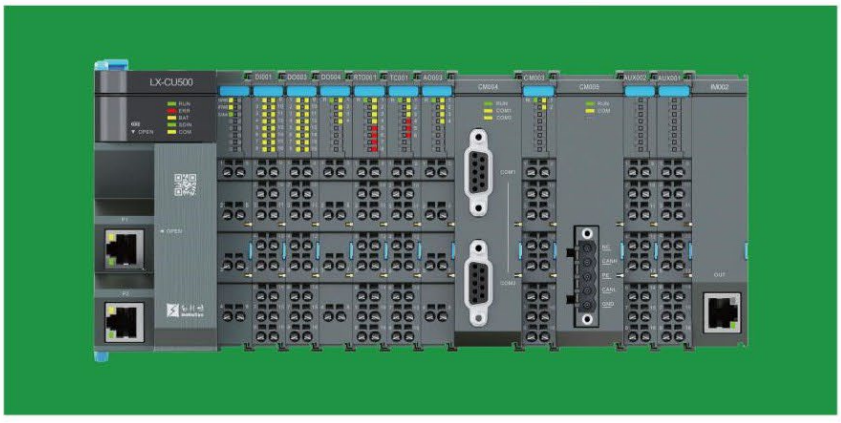


LX系列可编程控制器
运动指令手册

版本：2025.08

LX 系列可编程控制器
运动指令手册

2025.08

V1.2.1

版权声明

手册内容，包括文字、图表、标志、标识、商标、产品型号、软件程序、版面设计及其它内容等，均受《中华人民共和国著作权法》、《中华人民共和国商标法》、《中华人民共和国专利法》及与之适用的国际公约中有关著作权、商标权、专利权或其他财产所有权法律的保护，为北京和利时智能技术有限公司专属所有或持有。

由于本手册中所描述的设备有多种使用方法，用户以及设备使用责任人必须保证每种方法的可容许性。对由使用或错误使用这些设备造成的任何直接或间接损失，北京和利时智能技术有限公司将不负法律责任。

由于实际应用时的不确定因素，北京和利时智能技术有限公司不承担直接使用本手册中提供的数据的责任。

本手册仅供商业用户阅读，在未得到北京和利时智能技术有限公司书面授权的情况下，无论出于何种目的和原因，不得以任何形式（包括电子、机械或其它形式）传播或复制本手册的任何内容。违者我公司将依法追究其相关责任。

已核对本手册中的内容、图表与所述硬件设备相符，但误差难以避免，并不能保证完全一致。同时，会定期对手册的内容、图表进行检查、修改和维护，恕不另行通知。

HollySys、和利时、  的字样和徽标均为北京和利时智能技术有限公司的商标或注册商标。手册中涉及到的其他商标或注册商标属于它们各自的拥有者。

地址：北京经济技术开发区地盛中路 2 号院

邮编：100176

商务电话：010-5898 1588

产品咨询热线：400-811-1999

技术支持电话：010-58981514

传真：010-5898 1558

和利时集团网址：http://www.hollysys.com/

FA 业务官网：https://fa.hollysys.com/

Email：PLC@hollysys.com

新浪微博：http://weibo.com/hollysysplc

目录

第1章 前言

1.1 本文档用途

本文档将介绍 LX 系统中运动控制指令的使用，包含功能介绍、指令的参数说明、组态示例等内容。有关基本指令的内容请参见《LX 系列可编程控制器基本指令手册》。使用本手册时，请结合《LX 系列可编程控制器编程手册》一起使用，帮助用户更灵活的组态。

1.2 阅读对象

本手册供具有自动化及控制系统专业知识的工程师或相关专业技术人员使用。

1.3 对象产品

本手册适用的产品对象如下：

- LX 系列 PLC 产品

1.4 使用声明

- 本手册内容都按规定经过测试，图表与所述软件和硬件产品相符，但误差不可避免，并不能保证完全一致。如果您有关于改进或更正本手册内容的任何建议，请及时告知，我们将在下个版本进行修正。
- 因产品迭代更新，本手册中的相关参数和信息有可能会发生变更。如有问题，请咨询技术支持。
- 由于本手册中所描述的设备有多种使用方法，用户以及设备使用责任人必须保证每种方法的可容许性。对由使用或错误使用这些设备造成的任何直接或间接损失，和利时公司将不负法律责任。
- 由于实际应用时的不确定因素，和利时公司不承担直接使用本手册中提供的数据的责任。

1.5 安全注意事项

1.5.1 安全声明

（1） 使用本产品前，请先阅读并遵守本安全注意事项。

（2） 为保障人身和设备安全，在使用本产品时，请严格遵守产品上标识信息及手册中的安全注意事项。

（3） 手册中的“注意”、“小心”、“警告”和“危险”事项，并不代表所应遵守的所有安全事项，只作为所有安全注意事项的补充。

（4） 本产品应在符合设计规格要求的环境下使用，否则可能造成故障，因未遵守相关规定引发的设备损坏不在产品质量保证范围之内。

（5） 对于任何未按本手册规定操作或不符合要求的人员操作导致的人身安全事故和财产损失，和利时公司概不负责。

1.5.2 警告提示

为了您的人身安全以及避免财产损失，必须注意本手册中的警告提示。以下警告提示根据危险等级由高到低表示。当存在多个危险等级时，仅提示最高等级，如果最高等级为导致人身伤害的警告提示，则同时可能存在导致财产损失的警告。

- 人身安全提示：用一个警告三角  表示。
- 财产损失提示：不带警告三角。

 危险
表示如果不按规定操作，将会导致死亡或者严重的人身伤害。

 警告
表示如果不按规定操作，可能导致死亡或者严重的人身伤害。

 小心
表示如果不按规定操作，可能导致轻微的人身伤害。

注意
表示如果不按规定操作，可能导致财产损失。

第2章 文档指南

2.1 相关手册

本产品的手册种类如下表所示，请根据使用目的选读。

手册名称	用途	内容
LX 系列可编程控制器模块手册	了解 LX 系统硬件模块相关信息	对所有硬件模块进行说明，内容包含： 模块功能 硬件接口 接线 指示灯 技术参数
LX 系列可编程控制器通信手册	指导用户搭建 LX 系统通信网络	对 LX 系统所支持的通信网络进行说明，内容包含： 通讯协议介绍以及功能使用
LX 系列可编程控制器编程手册	了解 FA-AutoThink 编程软件的使用	对 FA-AutoThink 编程软件的界面、功能及相关操作进行说明，内容包含： 软件界面介绍 工程创建 编程 在线调试功能
LX 系列可编程控制器基本指令手册	了解 LX 系统基本控制指令的使用	对基本控制指令的使用进行说明，内容包含： 指令功能 参数 示例
LX 系列可编程控制器 PLCopen 指令手册	了解 LX 系统 PLCopen 运动控制指令的使用	对 PLCopen 运动指令的使用进行说明，内容包含： 参数 功能 示例 使用注意事项 参见

2.2 使用约定

在本手册中使用以下符号：

-  提示：该符号表示有用的技巧或建议，以便更高效地使用产品。

2.3 关于手册获取

如需获得电子版 PDF 手册文件，可通过以下方式获取：

登陆和利时公司官网（https://cn.hollysys.com/other/download.html）下载。

第3章　手册修订记录

手册版本	修订日期	修订内容
V1.0.0	2023 年 7 月	创建
V1.1.0	2023 年 10 月	新增指令： HMC_SynchroToAlm（同步到算法任务） 删除指令： HMC_SetSSIEncoderBit (设置 SSI 编码器位数)、HMC_SetSSIEncoderDataType (设置 SSI 编码器码制)、HMC_SetSSIEncoderFreq(设置 SSI 总线时钟频率)、HMC_SetSSIEncoderTp (设置 SSI 总线数据锁存时间)、 EtherCAT 协议相关指令
V1.1.1	2023 年 12 月	增加参数表默认值和掉电保护
V1.1.2	2024 年 8 月	增加全文指令类型说明
V1.2.0	2025 年 3 月	更新版权声明信息
V1.2.1	2025 年 8 月	新增 HMC 运动指令库使用说明

第4章　指令系统概述

4.1　指令列表

指令列表

指令库	指令	名称	功能简介
基本参数	AxisID	轴号	轴号，表示各组轴参数的轴号。
	AxisType	轴类型	轴类型，每 10 个为一大类。
	Units	单位转换因子	设置指定轴的单位转换因子轴参 Units。
状态参数	MCmdType	当前指令类型	轴的当前指令类型。
	NCmdType	下一条指令类型	轴的下一条指令类型。
	AxisStatus	轴状态	当前运行状态。
	AxisErrMask	轴状态掩码	轴状态异常掩码。
	LimitState	超限状态	描述当前超限状态。
位置相关参数	MPos	反馈位置	当前测量反馈位置。
	DPos	本周期目标位置	当前目标位置。
	FE	偏差	实时的随动误差。
	StopFETol	反馈位置是否到达目标位置的死区	判断速度模式时 MPos 是否到达目标位置。
	RepMode	位置行程模式	设置该轴的位置行程模式。
	RepDist	循环行程长度	设置循环行程长度。
	Remain	剩余位置	离指令目标位置的剩余位移。
速度相关参数	Speed	目标速度	设定指令解算的目标速度。
	Accel	加速度	设置加速度。
	Decel	减速度	设置加速度。
	MpSpeed	反馈速度	实时测量到的反馈速度。
	VpSpeed	解算速度	运动指令实时解算出来的速度数值。
	Direction	解算速度的方向	运动指令实时解算出来的速度方向。
	Merge	融合功能模式	设置速度融合功能。

	CreepSpeed	原点回归爬行速度	设置爬行速度。
	FastDecel	快速减速度	设置快速减速度。
	Jerk	加加速度	设置加加速度。
	JerkMode	加速模式	通过加加速度来改变加减速度和速度。
	CmdStopMode	指令结束判断方式	判断伺服轴的运动指令是否执行结束时的模式。
	MoveAbsMode	循环行程轴绝对运动时方向选取方式	设置循环行程轴绝对运动时方向。
	CancelMode	Cancel 时指令停止模式	设置 Cancel 时指令停止模式。
	CancelPos	Cancel 时停止位置	在指令 Cancel 完成时轴将到达的位置。
系数因子参数	Kp	PID 的比例系数	比例系数运算
	Ki	PID 的积分增益	积分增益运算
	Kd	PID 的微分增益	微分增益运算
	Kov	测量速度增益	测量速度增益
	Kvff	速度前馈增益	速度前馈增益
	Kaff	加速度前馈增益	加速度前馈增益
	Ko	PID 输出放大因子	伺服输出比例，PID 计算后伺服输出值的放大因子。
限幅限位参数	FELimits	随动误差限制	设置 FE（随动误差）的正常范围。
	FsLimitMode/RsLimitMode	前/后限位时指令撤销方式	设置前/后限位时指令撤销方式。
	FHLimitIn/RHLimitIn	前/后向硬限位通道号	设置前/后向硬限位行程开关所对应的 DI 通道号。
	FSLimit/RSLimit	前/后向软限位边界值	设置前/后向软限位边界值
	DACLimitHigh/DACLimitLow	DAC 输出上限/下限	设置 DACOut 的上限/下限。
原点和保持参数	HomeState	原点回归状态	了解当前原点回归情况。
	HomeIn	原点回归所用	原点回归所用的 DI 通道号。

		的 DI 通道	
	HomeCapIn	原点回归使用的高速捕获通道	HMC_HomeEx 所使用的高速捕获通道号，在投放 HMC_HomeEx 指令时设置。
	HoldSpeed	强制速度	当速度强制开关被触发时，轴会从当前速度加减速至 HoldSpeed。
	HoldIn	速度强制开关 DI 通道号	速度强制功能的触发源 DI 通道号。
输入相关参数	KeNumerator/KeDenominator	编码器输入比例 Ke 的分子/分母	伺服轴或编码器轴的“单位转换因子”。
	EncoderValue	内部编码器值	当该轴通过编码器反馈位置信息时，该轴参数表示内部编码器值。
	FeedBackSrcMode	轴位置反馈源模式	设定轴反馈值 Mpos 输入源方式。
	FeedBackSrcValue	轴反馈自定义输入实时值	配置某参数地址作为此时轴的反馈输入源。
	FeedBackDataType	轴反馈自定义输入变量数据类型	用户自定义输入变量的数据类型。
输出相关参数	KsNumerator/KsDenominator	步进输出比例 Ks 的分子/分母	步进输出比例 Ks 的分子/分母，通过 HMC_SetKs 设置。
	PulseNum	步进轴周期输出脉冲数	每个伺服周期，步进轴输出的脉冲数。
	ServoLoop	闭环输出开关	设置该开关决定位置闭环是否生效。
	DAC	速度模式开环输出值	ServoLoop=0 时的 DACOut 的输出值。
	DACOut	速度模式输出值	当前轴的速度量输出结果。
	PosOffset	DACOut 正向死区值	伺服轴输出正向死区值。
	NegOffset	DACOut 负向死区值	伺服轴输出负向死区值。
	ZeroWindow	DACOut 零点间隙死区	DACOut 零点间隙死区。
	OutDstMode	DACOut 输出	设置 OutDstMode 来指定输出值 DACOut 的最终输

		目标模式	出目标。
	OutDstValue	DACOut 输出自定义地址实时值	OutDstMode=1 时，OutDstValue 实时显示用户自定义变量数值。
	OutDstDataType	DACOut 输出自定义变量数据类型	OutDstValue 实时显示用户自定义变量数值。
系统启动指令	HMC_DriveEnable	伺服使能	控制与控制器相连的所有伺服驱动器使能信号的开启与关闭。
	HMC_GetDriveEnableState	获取伺服使能状态	获取伺服总使能（DriveEnable）状态。
原点搜寻及位置定义	HMC_Home	原点回归	通过检测原点开关信号或伺服电机编码器 Z 相信号为该轴确定原点。
	HMC_HomeEx	高级原点回归	通过检测原点开关信号或伺服电机编码器 Z 相信号为该轴确定原点。
	HMC_DefOrigin	设置原点	改变某轴坐标系的原点，将参数 dMpos 所标定的原坐标系下的位置设置为新的坐标系原点。
	HMC_DefPos	设置当前位置	移动坐标系，将当前位置定义为 dMpos。
单向运动	HMC_Forward	正向运动	持续正向运动，运动速度由轴参数 Speed 决定。
	HMC_Reverse	反向运动	持续负向运动，运动速度由轴参数 Speed 决定。
	HMC_ForwardEx	高级正向运动	运动过程中可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。
	HMC_ReverseEx	高级反向运动	运动过程中可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。
点位运动	HMC_Move	相对运动	让某个轴按照预先设定的速度及加减速参数运动到相对于本指令开始执行时刻所在位置的相对位移处。
	HMC_MoveAbs	绝对运动	让某个轴按照预先设定的加速度、速度及加速度参数运动到坐标位置 dDistance 处。
	HMC_MoveEx	高级相对运动	该指令执行时首先修改轴参数 Speed=函数参数 dSpeed，然后以速度 Speed 运动到相对于本指令开始执行时刻所在位置的相对位移处。
	HMC_MoveAbsEx	高级绝对运动	轴参数 Speed=函数参数 dSpeed，让某个轴按照本指令设定的目标速度运动到绝对位置。
指令修正	HMC_Cancel	撤销指令	撤销指定轴运动缓存中的指令。
	HMC_MoveModify	目标位置变更	实时修改 HMC_Move/HMC_MoveAbs 指令的目标位置，修改后的目标位置为 dDpos。

	HMC_MoveModifyEx	高级目标位置变更	实时修改 HMC_Move/HMC_MoveAbs 指令的目标位置。
两轴直线插补	HMC_Move2D	2 轴直线插补	二维平面上运动轮廓为直线的插补功能。
	HMC_Move2DEx	高级 2 轴直线插补	二维平面上运动轮廓为直线的插补功能。
	HMC_MoveAbs2D	2 轴绝对值直线插补	二维平面上运动轮廓为直线的插补功能。
	HMC_MoveAbs2DEx	高级 2 轴绝对值直线插补	二维平面上运动轮廓为直线的插补功能。
三轴直线插补	HMC_Move3D	3 轴直线插补	三维空间上运动轮廓为直线的插补功能。
	HMC_Move3DEx	高级 3 轴直线插补	三维空间上运动轮廓为直线的插补功能。
	HMC_MoveAbs3D	3 轴绝对值直线插补	三维空间上运动轮廓为直线的插补功能。
	HMC_MoveAbs3DEx	高级 3 轴绝对值直线插补	三维空间上运动轮廓为直线的插补功能。
多轴直线插补	HMC_MoveND	N 轴直线插补	可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。
	HMC_MoveNDEx	高级 N 轴直线插补	可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。
	HMC_MoveAbsND	N 轴绝对值直线插补	可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。
	HMC_MoveAbsNDEx	高级 N 轴绝对值直线插补	可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。
平面圆弧插补	HMC_MoveCircle	圆弧插补	二维平面上运动轮廓为圆弧的插补功能。
	HMC_MoveCircleEx	高级圆弧插补	二维平面上运动轮廓为圆弧的插补功能。
球面插补	HMC_MoveSpherical	球弧插补	三维空间上运动轮廓为球弧的插补功能。
	HMC_MoveSphericalEx	高级球弧插补	三维空间上运动轮廓为球弧的插补功能。
螺旋线插补	HMC_MoveHelical	螺旋线插补	三轴共同完成圆柱螺旋线轨迹。
	HMC_MoveHelicalEx	高级螺旋线插补	三轴共同完成圆柱螺旋线轨迹。
样条插补	HMC_MoveSplineND	样条插补	以当前点为起点，执行 N 轴样条插补运动。
	HMC_MoveSplineNDEx	高级样条插补	以当前点为起点，执行 N 轴样条插补运动。
	HMC_CalcMoveSplineNDdata	生成样条数据	生成 N 轴插补的三次样条曲线。
电子齿轮	HMC_Gear	电子齿轮	实现无轴传动。
	HMC_SetGearRatio	设置电子齿轮比例	实时调用该指令动态改变运转中电子齿轮比例。

	HMC_GetGearMaster	获取电子齿轮主轴	获取正在运行的 HMC_Gear 主轴号。
	HMC_GetGearRatio	获取电子齿轮比例	获取当前正在运行的 HMC_Gear 指令齿轮比。
	HMC_SetGearPhaseOffset	设置电子齿轮相位延迟	可调整电子齿轮从轴的跟随相位。
	HMC_GetGearPhaseOffset	获取电子齿轮相位延迟	获取电子齿轮相位延迟设定量。
	HMC_SetGearTransationSwitch	电子齿轮过渡段开关	开启/关闭电子齿轮平滑功能。
电子凸轮	HMC_Cam	电子凸轮	通过选取目标运动轨迹（二维）上的离散坐标点，使用电子凸轮功可以通过主从轴联动的方式逼近目标运动轨迹。
	HMC_SplineCam	样条凸轮	采用折线连接的方式把主从轴位置点连接起来生成轨迹。
	HMC_GetCamMasterPosIndex	获取电子凸轮主轴位置索引	该获取当前正在运行的凸轮主轴位置索引。
	HMC_ClcMoveLinkData	追剪运算	按照设定的关系生成主从轴位置数据。
	HMC_ClcFlexLinkData	飞剪/旋切运算	按照设定的关系生成主从轴位置数据。
	HMC_CancelREPCam	循环凸轮条件撤销控制	对正在运行的循环凸轮联动指令使用条件撤销功能。
	HMC_MoveLink	追剪运动	集成了凸轮和追剪运算功能，控制从轴跟随主轴按一定规律运动。
	HMC_MoveFlexLink	飞剪/旋切运动	具有主从轴位置及加减速规划的凸轮功能。
运动叠加	HMC_Superimpose	叠加	可实现两个主轴增量位移的简单叠加。
	HMC_SuperImposeEx	高级叠加	可实现两个主轴增量位移的线性叠加。
	HMC_SetSuperimposeRatio	设置叠加比例	设置叠加比例。
	HMC_GetSuperimposeMaster1	获取叠加主轴 1	从输入到输出所执行的动作及信号获取叠加的第一个主轴。
	HMC_GetSuperimposeMaster2	获取叠加主轴 2	获取叠加的第二个主轴。
	HMC_GetSuperimposeRatio1	获取叠加主轴 1 比例	获取叠加的第一个主轴比例。
	HMC_GetSuperimposeRatio2	获取叠加主轴 2 比例	获取叠加的第二个主轴比例。
	HMC_SetAddAxis	设定和运动轴	用于完成二轴运动求和。
	HMC_GetAddAxis	获取和运动叠	用于获取和运动叠加轴轴号，255 表示未关联任何

		加轴	叠加轴。
低速到位输出	HMC_NormalSwitch1Dconfig	设置一维低速输出	配置一维低速输出。
	HMC_NormalSwitch2DConfig	设置二维低速输出	配置二维低速输出。
	HMC_GetNormalSwitchNum	获取已输出数目	获取指定 ID 的低速到位输出已执行的段数。
	HMC_GetNormalSwitchState	获取低速输出状态	获取指定 ID 的低速输出的当前状态。
	HMC_CloseNormalSwitch	关闭低速输出	关闭该 ID 的低速到位输出。
示波器功能指令	HMC_SetSampleVar	设置采样变量）	对系统的轴参数、I/O 数据、中间/系统变量等进行数据采集。
	HMC_SetSamplePara	设置采样参数	配置指定采样通道的采样间隔时间、采样点数、采样数据存储区地址。
	HMC_GetSampleData	获取采样数据	获取指定通道的某子通道的序号为 uiDataNum 的采样数据。
	HMC_TriggerSample	触发采样	触发指定采样通道开始执行采样。
	HMC_DisableSample	停止采样	禁止指定采样通道执行采样。
	HMC_TriggerScope	触发示波器	触发示波器开始执行
	HMC_GetSampleNum	获取已采样数目	获取指定采样通道当前已完成的采样数目。
	HMC_SetSampleSingleVar	设置单个采样变量	配置指定采样通道的 0 号子通道的待采样数据来源。
运动保持功能指令	HMC_HoldConfig	设置强制速度开关	设置强制速度开关。
运动前瞻	HMC_SetMerge	设置融合功能	设置轴参数 Merge 模式。
	HMC_ClcJerk	冲击运算	根据 Speed 和曲率计算 Jerk。
	HMC_SetMergeAngle	设置融合角度	设置轴的融合角度。
	HMC_GetMergeStopAngle	获取融合停止角	获取插补运动停止角。
	HMC_GetMergeDecelAngle	获取融合减速角	获取插补运动减速角。
	HMC_SetMergeSpeedTolerRatio	设定速度波动抑制比	设定运动前瞻过程中的速度波动抑制比参数。
	HMC_GetMergeSpeedTolerRatio	读取速度波动抑制比	读取当前的速度波动抑制比。
曲线限速	HMC_SetCurveJerkLimit	设定曲线冲击	设定曲线插补过程中物理冲击的上限值。

		上限	
	HMC_GetCurveJerkLimit	读取曲线冲击上限	读取设定的曲线插补物理冲击的上限值。
切向追踪	HMC_MoveTang	切向追踪	自动实时调整刀具方向与切割轨迹的切向方向。
	HMC_MoveTangConfigCornerMode	切向追踪拐角提刀设定	可执行拐角提刀功能，以保护刀具，保证加工效果。
	HMC_GetMoveTangCornerState	获取切向追踪拐角抬刀状态	获取切向追踪拐角抬刀状态。
	HMC_MoveTangCornerFollowNextCmd	C 轴定位至下一条指令的起始切向角	可驱动 C 轴定位至下一条指令的起始切向角。
	HMC_MoveTangCornerContinue	切向追踪继续运行	在拐角暂停等待抬刀的状态下，使用该指令可恢复切向追踪继续运行。
	HMC_GetMoveTangDestDirection	获取切向追踪目标方向位置	获取切向追踪目标方向位置。
	HMC_GetMoveTangDestVelocity	获取切向追踪目标速度	获取切向追踪目标切向角的变化速度。
	HMC_GetMoveTangDeviation	获取切向追踪方向偏差	获取切向追踪 C 轴（ucAxisC，刀具旋转轴）当前指令位置与目标切向角的偏差。
	HMC_GetMoveTangMasterAxisX	获取切向追踪主轴 X	获取当前切向追踪指令的 X 轴。
	HMC_GetMoveTangMasterAxisY	获取切向追踪主轴 Y	获取当前切向追踪指令的 Y 轴。
多轴同步控制	HMC_GantrySynchro	龙门同步控制	对主轴和从轴进行动态同步。
	HMC_GantryInit	龙门同步回参考点位置对齐初始化	龙门同步回参考点位置对齐初始化。
	HMC_SetGantryAxisPID	设置龙门同步轴补偿 PID 参数	设置龙门组内指定龙门轴的动态纠偏 PID 参数。
	HMC_GetGantryAxisPID	读取龙门同步轴补偿 PID 参数	读取龙门组内指定龙门轴的动态纠偏 PID 参数。
	HMC_GantryGroupDefPos	设置龙门同步轴组当前位置	对龙门组内所有龙门轴的位置重定义。
	HMC_GetGantryInitState	读取龙门同步控制函数当前	读取该指令当前执行状态。

		状态	
轴位置补偿功能	HMC_SetAxisCompenPos	轴位置补偿量设定	设定轴位置补偿量。
	HMC_SetAxisCompenPara	轴位置补偿运动参数设定	设定轴位置补偿运动的运动学参数，实时生效。
	HMC_TrigAxisCompens	触发位置补偿	触发轴位置补偿动作。
	HMC_StopAxisCompens	停止位置补偿	立即停止正在进行中的轴位置补偿动作。
	HMC_GetAxisCompenState	获取轴位置补偿状态	获取轴补偿状态。
Stewart 6 自由度并联机构	HMC_StewartControl	Stewart 机构控制	该指令完成 Gough-Stewart 6 自由度并联机构的运动学变换。
5 自由度 SCARA 机械臂	HMC_Scara5Control	5 自由度 SCARA 机械臂控制	该指令完成 5 自由度 SCARA 机构的运动学变换。
	HMC_Scara5ForwardKinematics	5 自由度 SCARA 运动学正解	该指令完成 5 自由度 SCARA 机构的运动学正解。
	HMC_Scara5ReverseKinematics	5 自由度 SCARA 运动学逆解	该指令完成 5 自由度 SCARA 机构的运动学逆解。
4 自由度串联机械臂	HMC_Serial4Control	4 自由度串联机械臂控制	该指令完成 4 自由度串联机械臂的运动学变换。
	HMC_Serial4ForwardKinematics	4 自由度串联机械臂运动学正解	该指令完成 4 自由度串联机械臂的运动学正解。
	HMC_Serial4ReverseKinematics	4 自由度串联机械臂运动学逆解	该指令完成 4 自由度串联机械臂的运动学逆解。
自定义运动控制功能	HMC_SetDposSwitch	Dpos 规划开关切换	该指令切换指定轴的 Dpos 规划开关。
	HMC_SetDposVal	Dpos 规划	设置 Dpos 规划。
	HMC_SynchroToAlm	同步到算法任务	该指令可用于与 MC 内部的算法硬实时任务的同步。
轴参数设定指令	HMC_SetAxisType	设置轴类型	设置指定轴的轴类型。
	HMC_HwLimitConfig	设置硬限位开关	设置指定轴的硬限位开关。
	HMC_SetUnits	设置用户单位	设置指定轴的单位转换因子轴参 Units。

	HMC_SetJerkMode	设置梯形/S 形加速度	设置 JerkMode。
	HMC_SetKe	设置 Ke	设置轴参数 Ke 的分子（轴参数 KeNumerator）和分母（轴参数 KeDenominator）。
	HMC_SetKs	设置 Ks	设置轴参数步进输出比例 Ks 的分子（轴参数 KsNumerator）和分母（轴参数 KsDenominator）。
	HMC_SetFeedBackSrcAddr	设置用户自定义反馈输入地址	配置轴实际位置 Mpos 的用户自定义输入源。
	HMC_SetOutDstAddr	设置用户自定义输出地址	设置位置闭环输出值 DACOut 的自定义输出地址。
	HMC_SetSSIEncoderBit	设置 SSI 编码器位数	设置 SSI 编码器位数。
	HMC_SetSSIEncoderDataType	设置 SSI 编码器码制	配置 SSI 编码器轴码制。
	HMC_SetSSIEncoderFreq	设置 SSI 总线时钟频率	配置 SSI 总线时钟频率。
	HMC_SetSSIEncoderTp	设置 SSI 总线数据锁存时间	配置 SSI 总线数据锁存时间。
	HMC_SetCmdBufferLoadMode	设置轴指令加载模式	设置轴指令的加载模式。
	HMC_GetCmdBufferLoadMode	获取轴指令加载模式	获取轴指令的加载模式。
	HMC_SetCmdStopMaxTime	设置指令结束最大等待时间	设置指令结束最大等待时间。
状态读取指令	HMC_GetSysErr	获取系统错误码	获取系统最近一次发生的系统错误。
	HMC_GetUserErrID	获取用户错误码	返回用户错误码。
	HMC_GetUserErrNum	获取用户错误数	返回已发生的用户错误数。
	HMC_GetUserErrMes	获取用户错误码附加信息	返回用户错误码附加信息。
	HMC_GetAllAxisErr	获取所有轴的总体异常状态	获得所有轴的总体异常状态。
	HMC_GetCmdErr	获取轴指令错	根据轴指令 ID，返回该轴指令错误。

		误	
	HMC_GetCmdState	获取轴指令状态	根据轴指令 ID，返回该轴指令状态。
	HMC_WaitCmdFinish	等待指令完成	等待直到“某运动指令运行完成后”才返回。
	HMC_WaitCmdLoaded	等待指令装载	等待直到“某运动指令（uiAxisCmdID）装载后” 即指令刚开始执行才返回。
	HMC_WaitAxisIdle	等待轴空闲	等待直到“某轴（ucAxisID）的运动指令队列为空”才返回，否则一直处于阻塞状态，不返回。
	HMC_GetCmdBufferState	获取轴指令缓存状态	获取某轴（ucAxis）的指令缓存状态。
故障清除指令	HMC_ClearAxisErr	消除轴错误	清除某个轴的错误。
	HMC_ClearAllErr	消除所有轴错误	清除所有轴的错误。

4.2 指令使用说明

使用指令前，请先了解指令内容中涉及的相关信息，便于您更好的理解和使用指令。

信息项	说明
指令/功能块	表示指令名称，例如：AxisID
名称	表示指令的中文名称，例如，加法指令
FB/FUN	表示指令是功能块（FB）还是函数（FUN） 功能块型指令：可以被程序或 FB 调用。 函数型指令：可以被程序、FB、FUN 的任意一个调用。
功能	对指令功能进行说明
参数	对指令的输入、输出以及中间变量等参数信息进行说明。
示例	指令的应用示例，通过在 LD 和 ST 中的编程示例进行说明。
使用注意事项	说明指令使用时的注意事项，也包含特殊场景下的应用以及异常情况的发生。
参见	使用该指令需要参考的其他内容信息

第5章 HMC 运动指令库

HMC 运动指令库是和利时自研的运动控制指令，包括了单轴、多轴、电子齿轮、电子凸轮等多种轴运动控制。通过 HMC 运动指令，可以对轴参数进行读写操作，极大的方便了用户编程。

HMC 运动指令根据是否会在指令队列中缓存分为如下两大类：

- 非轴运动缓存类指令：指令不会放在轴指令队列里进行缓存。
- 轴运动缓存类指令：指令会放在轴指令队列里进行缓存。

- 说明：缓存队列最多支持缓存 64 条指令，缓存满之后再投送指令会阻塞 IEC 程序运行。

用户编程时可根据实际规划选择合适的指令，并根据指令类型对程序运行过程中可能产生的情况做相应处理。

5.1 AXIS（轴参数）

每个轴对应一组同样的轴参数，用户通过查看轴参数实时值可了解轴的当前运行状态，也可通过直接修改或运控控制指令间接修改轴参数，改变轴的运行状态。

轴参数按照读写属性可分为“只读”和“读写”两种。“只读”参数在组态时不能被赋值，也不能通过 AutoThink 来“写变量”；“读写”参数在组态时可以被赋值，也可以通过 AutoThink 写变量。“只读”参数又分为两种：第一种只反映轴的一些状态（例如 LimitState）或当前计算或测量出来的值（如 VpSpeed、MpSpeed），此类参数是由控制算法解算刷新；第二种是用户可通过设置函数设置的只读参数（如 AxisType）。

在 AutoThink 中，轴参数以功能块的方式组织，轴参数是其成员，每一个轴的所有轴参数构成一个功能块变量，该变量支持用户改名。举例来说，用户将以类似 Axis0.Speed 的方式访问轴 0 的 Speed，如果用户将轴 0 的功能块变量名改为 AxisLaser，则可以用 AxisLaser.Speed 的方式来访问轴 0 的 Speed。

AutoThink 中将所有轴的轴参数定义在数据区的某个固定区域中，为防止数据冲突，AutoThink 不允许用户在该区中再定义其他变量。

5.1.1 基本参数（Basic）

5.1.1.1 AxisID（轴号）

5.1.1.1.1 基本属性

类型	USINT
读写属性	只读
设置函数	-
初始值	0
取值范围	0~63
相关章节	无

5.1.1.1.2 含义描述

轴号，表示各组轴参数的轴号。

5.1.1.2 AxisType（轴类型）

5.1.1.2.1 基本属性

类型	EmAxisType
读写属性	只读
设置函数	HMC_SetAxisType
初始值	Unused（-1）：该轴未使用
取值范围	Unused（-1）：该轴未使用 Virtual（0）：虚轴 EtherCAT_Position（50）：EtherCAT 总线连接轴，位置控制 EtherCAT_Speed（51）：EtherCAT 总线连接轴，速度控制 EtherCAT_Torque（52）：EtherCAT 总线连接轴，转矩控制 EtherCAT_Encoder（53）：EtherCAT 总线连接轴，编码器计数（编码器类型的轴不接受运动控制指令，仅仅跟踪编码器值）
相关章节	HMC_SetAxisType（设置轴类型）

5.1.1.2.2 含义描述

通过函数 HMC_SetAxisType 设置轴类型，不同型号的控制器，同一型号控制器的不同轴号，所支持的轴类型不同，参见 HMC_SetAxisType（设置轴类型）。

轴类型的不同，会影响该轴支持的指令类型。例如，编码器轴类型，不支持运动控制指令。

轴类型的不同，会影响该轴的输入、输出方式，各轴类型的输入、输出信号见下表。更改轴类型后，对应端子的输入、输出信号会发生变化，请确认无误后，再进行对应操作。

轴类型标示	轴类型代码	与控制器连接方式	输入	输出	备注
Unused	-1	未启用轴	无	无	运动控制指令不允许投放到该轴；算法不对此类型轴进行解算，可节约算法解算时间
Virtual	0	控制器内置	无	无	一般用作插补指令的合轴或中间解算轴
EtherCAT_Position	50	EtherCAT 总线连接	通过 EtherCAT 总线获得编码器反馈位置（轴的位置信息），不作位置闭环控制	通过 EtherCAT 总线输出指令位置	-
EtherCAT_Speed	51	EtherCAT 总线连接	通过 EtherCAT 总线获得编码器反馈位置（轴的位置信息），servoLoop = 1 时作位置闭环控制运算	通过 EtherCAT 总线输出指令速度	-
EtherCAT_Torque	52	EtherCAT 总线连接	通过 EtherCAT 总线获得编码器反馈位置（轴的位置信息），servoLoop = 1 时作位置闭环控制运算	通过 EtherCAT 总线输出指令转矩	-
EtherCAT_Encoder	53	EtherCAT 总线连接	通过 EtherCAT 总线扩展的 ECI 模块获得编码器反馈位置		

5.1.1.2.3 示例

因为 AxisType 是枚举类型，所以可按照如下方式使用。

```
Result := HMC_SetAxisType(1, 51); (*EtherCAT_Speed 的轴类型枚举值为 51*)
```

```
IF51 = Axis1.AxisType THEN
Axis1.Speed := 4000;
END_IF
```

5.1.1.3 AxisState（轴 PLCopen 状态）

5.1.1.3.1 基本属性

类型	MC_AXIS_STATE
读写属性	只读
设置函数	-
初始值	0
取值范围	mcDisabled(0)：未使能状态 mcErrorstop(1)：故障停机状态 mcStopping(2)：停止状态 mcStandstill(3)：使能状态 mcDiscreteMotion(4)：离散运动 mcContinuousMotion(5)：连续运动 mcSynchronizedMotion(6)：同步运动 mcHoming(7)：原点回归
相关章节	PLCopen 状态机

5.1.1.3.2 含义描述

AxisState 表示当前轴状态，状态符合 PLCopen 标准单轴运动控制状态机，轴状态会随着控制器指令的变化而变化，具体参见 PLCopen 状态机章节。

5.1.1.4 Units（单位转换因子）

5.1.1.4.1 基本属性

类型	LREAL
读写属性	只读
设置函数	HMC_SetUnits
初始值	1

取值范围	正浮点数
相关章节	HMC_SetUnits（设置用户单位）

5.1.1.4.2 含义描述

单位转换因子（只可为正值），单位：线/用户单位。

设置指定轴的单位转换因子轴参 Units，该参数只可为正值，单位：线/用户单位，“线”是指轴走一个脉冲对应的刻度单位。

通过设置轴参 Units，可将编码器反馈的计数值或脉冲输出值转换为用户方便使用的单位数值，例如 mm、角度等。

Units 在系统中的默认值是 1，表示用户单位是“线”，因此 Speed、Accel、Decel、FastDecel 等运动参数的单位均是“线/秒”，或者“线/（秒*秒）”；当 Units 被修改后，这些运动参数的单位自然变为了“用户单位/秒”，或者“用户单位/（秒*秒）”，但是这些运动参数的数值需要用户根据工程需要手动修改。

该参数设置方式，参见 HMC_SetUnits（设置用户单位）。

5.1.2 状态参数（Status）

5.1.2.1 MCmdType（当前指令类型）

5.1.2.1.1 基本属性

类型	EmCmdType
读写属性	只读
设置函数	-
初始值	HMC_McIdle（空）
取值范围	系统的所有运动控制指令类型

5.1.2.1.2 含义描述

轴的当前指令类型。每个轴有 64 级运动指令缓存(指令队列 / FIFO)，位于指令队列中队首位置的指令为当前正在被执行的指令。

系统支持的所有运动控制指令类型

值	名称	说明	备注
0	HMC__McIdle	空闲	无运动指令
1	HMC__ERR1	预留 1	系统预留
2	HMC__ERR2	预留 2	系统预留
3	HMC__ERR3	预留 3	系统预留
4	HMC__ERR4	预留 4	系统预留
5	HMC__ERR5	预留 5	系统预留
6	HMC__Home	原点回归	-
7	HMC__HomeB	原点回归（阻塞）	暂不支持
8	HMC__HomeEx	高级原点回归	暂不支持
9	HMC__HomeExB	高级原点回归（阻塞）	暂不支持
10	HMC__Move	相对运动	-
11	HMC__MoveB	相对运动（阻塞）	暂不支持
12	HMC__MoveEx	高级相对运动	-
13	HMC__MoveExB	高级相对运动（阻塞）	暂不支持
14	HMC__MoveAbs	绝对运动	-
15	HMC__MoveAbsB	绝对运动（阻塞）	暂不支持
16	HMC__MoveAbsEx	高级绝对运动	-
17	HMC__MoveAbsExB	高级绝对运动（阻塞）	暂不支持
18	HMC__Forward	正向运动	-
19	HMC__ForwardEx	高级正向运动	-
20	HMC__Reverse	反向运动	-
21	HMC__ReverseEx	高级反向运动	-
22	HMC__Move2D	2 轴直线插补	-
23	HMC__Move2DB	2 轴直线插补（阻塞）	暂不支持
24	HMC__Move2DEx	高级 2 轴直线插补	-
25	HMC__Move2DExB	高级 2 轴直线插补（阻塞）	暂不支持
26	HMC__Move3D	3 轴直线插补	-
27	HMC__Move3DB	3 轴直线插补（阻塞）	暂不支持
28	HMC__Move3DEx	高级 3 轴直线插补	-
29	HMC__Move3DExB	高级 3 轴直线插补（阻塞）	暂不支持
30	HMC__MoveND	N 轴直线插补	-
31	HMC__MoveNDB	N 轴直线插补（阻塞）	暂不支持
32	HMC__MoveNDEx	高级 N 轴直线插补	-

33	HMC__MoveNDExB	高级 N 轴直线插补（阻塞）	暂不支持
34	HMC__MoveCircle	圆弧插补	-
35	HMC__MoveCircleB	圆弧插补（阻塞）	暂不支持
36	HMC__MoveCircleEx	高级圆弧插补	-
37	HMC__MoveCircleExB	高级圆弧插补（阻塞）	暂不支持
38	HMC__MoveHelical	螺旋线插补	-
39	HMC__MoveHelicalB	螺旋线插补（阻塞）	暂不支持
40	HMC__MoveHelicalEx	高级螺旋线插补	-
41	HMC__MoveHelicalExB	高级螺旋线插补（阻塞）	暂不支持
42	HMC__MoveSpherical	球弧插补	-
43	HMC__MoveSphericalB	球弧插补（阻塞）	暂不支持
44	HMC__MoveSphericalEx	高级球弧插补	-
45	HMC__MoveSphericalExB	高级球弧插补（阻塞）	暂不支持
46	HMC__Gear	电子齿轮	-
47	HMC__Superimpose	叠加	-
48	HMC__SuperimposeEx	高级叠加	-
49	HMC__Cam	电子凸轮	-
50	HMC__MoveAbs2D	2 轴绝对值直线插补	-
51	HMC__MoveAbs2DEx	高级 2 轴绝对值直线插补	-
52	HMC__MoveAbs3D	3 轴绝对值直线插补	-
53	HMC__MoveAbs3DEx	高级 3 轴绝对值直线插补	-
54	HMC__MoveAbsND	N 轴绝对值直线插补	-
55	HMC__MoveAbsNDEx	高级 N 轴绝对值直线插补	-
56	HMC__MoveTang	切向追踪	-
57	HMC__MoveSplineND	样条插补	-
59	HMC__MoveSplineNDEx	高级样条插补	-
62	HMC__MoveLink	追剪运动	-
63	HMC__StewartControl	Stewart 机构控制	
64	HMC__SplineCam	样条凸轮	
65	HMC__Scara5Control	5 自由度 SCARA 机械臂控制	
66	HMC__MoveFlexLink	飞箭/旋切运动	
67	HMC__GantrySynchro	龙门同步控制	
68	HMC__GantryInit	龙门同步回参考点位置对齐初始化	
69	HMC__Serial4Control	4 自由度串联机械臂控制	
71	SMC__SyncMoveVelocity	周期同步速度控制	

72	MC__TorqueControl	力矩控制	

5.1.2.2 NCmdType（下一条指令类型）

5.1.2.2.1 基本属性

类型	EmCmdType
读写属性	只读
设置函数	-
初始值	HMC_McIdle（空）
取值范围	系统的所有运动控制指令类型

5.1.2.2.2 含义描述

轴的下一条指令类型。每个轴有 64 级运动指令缓存(指令队列 / FIFO)，位于指令队列中队首位置指令的下一条指令。

5.1.2.3 AxisStatus（轴状态）

5.1.2.3.1 基本属性

类型	UDINT
读写属性	只读
设置函数	-
初始值	0
取值范围	0 表示正常； 其他数值，转换为 2 进制数后按位解析，各 Bit 位含义见轴状态解析表。
相关章节	AxisErrMask（轴状态掩码）

5.1.2.3.2 含义描述

当前轴运行状态，为 0 时表示该轴无异常。不为 0 时，按位解析如下：

轴状态解析表

Bit	含义	消除方式	可能原因
0	随动误差 FE 连续 10 次超限 FELimit	状态锁定，需要用户清除	1、轴编码器线未连接； 2、电机驱动器接线错误导致伺服使能未开； 3、电机驱动器报错导致伺服使能关闭； 4、轴 FELimit 参数设置过小； 5、轴 PID 参数不合理。
1	随动误差 FE 超限 1 次警告	连续 1000 次不超限后该警告，自动解除	1、轴 FELimit 参数设置过于临界； 2、轴 PID 参数不合理。
2	轴参数设定错误	状态锁定，需要用户清除	轴参数设置非法值（例如 Speed、CreepSpeed 为负，Accel、Decel 为 0 或负）。
3	到达正向或反向限位，包括软限位和硬限位	限位条件不满足后，自动清除	有限形成轴发生满足以下某条件： 1、轴 Dpos≥FSLimit（正向软限位）； 2、轴 Dpos≤RSLimit（反向软限位）； 3、轴 FHLimitIn 关联 DI 为 1（正向硬限位）； 4、轴 RHLimitIn 关联 DI 为 1（反向硬限位）；
4	与总线轴通讯错误	MC1002R、MC1002E 等总线型控制器支持	1、总线断线或输入输出连接错误； 2、配置总线从站数与实际连接总线从站数不一致； 3、总线主站配置参数与从站参数不同； 4、配置从站型号和实际从站型号不同； 5、总线从站报错（错误信息查看从站使用手册）； （AT 中的全局变量 RtexDiagGroup 或 EtherCATDiagGroup 包含总线部分详细诊断信息）
5	龙门轴偏差超限报警	偏差消除后自动清除	龙门轴存在轴间偏差超报警阈值
6	龙门轴偏差超限停车	手动消除	龙门轴存在轴间偏差超紧急停车阈值
7~31	未使用	预留	预留

- 备注：当发生轴状态错误时，排除导致错误的原因后，可使用 HMC_ClearAxisErr 清除该轴状态错误，也可使用 HMC_ClearAllErr 清除所有轴状态错误。如果是与总线轴通讯错误，清除错误前需使用 Rtex_BusReset()或 EtherCAT_BusReset()复位相应总线协议栈。

5.1.2.3.3 示例

轴 0 到达正向或反向限位，关闭伺服使能开关。程序如下：

IF (UDINT_TO_DWORD (Axis0.AxisStatus) AND 8) >0 THEN　(*轴 0 AxisStatus 与 8 位，判断是否到达正向或反向限位*)

HMC_DriveEnable(0);(*关闭伺服使能开关*)

END_IF

5.1.2.4 AxisErrMask（轴状态掩码）

5.1.2.4.1 基本属性

类型	UDINT
读写属性	读写
设置函数	-
初始值	0
取值范围	0 表示各种 AxisStatus 值均不进行异常处理 其他值设置方式参考 AxisStatus 的各 bit 位含义，可设置一个或多个 bit 位为 1
相关章节	AxisStatus（轴状态）

5.1.2.4.2 含义描述

轴状态异常掩码，32 个 bit 位与 AxisStatus 的 32 个 bit 位对应。当 AxisErrMask 的某个 bit 位为 1 时，轴状态 AxisStatus 的对应 bit 位如果也为 1 时，认为出现系统异常，此时系统处理方式为：将 DriveEnable 信号禁用，将 ServoLoop 置为 0。

5.1.2.4.3 示例

轴 0 的 AxisStatus Bit0 和 Bit4 为 1 时，断开伺服使能开关，将闭环输出开关 ServoLoop 置为 0（开环）。程序如下：

Axis0.AxisErrMask=17;　(*设置轴 0 的 AxisErrMask 参数 Bit0 和 Bit4 为 1，则在轴 0 的 AxisStatus Bit0 和 Bit4 为 1 时，断开伺服使能开关，将闭环输出开关 ServoLoop 置为 0（开环）*)

5.1.2.5 LimitState（超限状态）

5.1.2.5.1 基本属性

类型	USINT
读写属性	只读
设置函数	-
初始值	0
取值范围	LimitState 为 0 表示正常，即未发生任何超限。 不为 0 时，先转换为 2 进制数，各 bit 位为 1 时含义如下： Bit 0：超过前向硬限位 Bit 1：超过后向硬限位 Bit 2：超过前向软限位 Bit 3：超过后向软限位 Bit 4：软限位相关参数错误（前向软限位≤后向软限位） Bit 5：各个状态冲突错误（超越前向软/硬限位，同时也超越后向软/硬限位）
相关章节	RepMode（位置行程模式） FsLimitMode/RsLimitMode（前/后限位时指令撤销方式） FHLimitIn/RHLimitIn（前/后向硬限位通道号） FSLimit/RSLimit（前/后向软限位边界值）

5.1.2.5.2 含义描述

轴为有限行程时（RepMode=0），该轴参数用来描述当前超限状态。

5.1.2.5.3 取值

Bit	说明	值	备注
0	到达正向硬限位	1	正向硬限位关联 DI 通道为 1
1	到达反向硬限位	2	反向硬限位关联 DI 通道为 1
2	到达正向软限位	4	Dpos≥FSLimit
3	到达反向软限位	8	Dpos≤RSLimit
4	软限位相关参数错误	16	正向软限位≤反向软限位
5	同时到达正向和反向限位	32	包括软限位和硬限位
6~7	未使用	-	预留

5.1.2.5.4 示例

轴 0 到达正向软限位时，反向位移 100。

程序如下：

```
IF (USINT_TO_WORD (Axis0.LimitState) AND 4) >0 THEN    (*轴 0 LimitState 与 4 位与，判断是否到达正向软限位*)
HMC_Move(0,-100);   (*轴 0 反向位移 100*)
END_IF
```

■ 备注

（1）轴到达正向限位（含软限位和硬限位）时，投送加重正向限位的指令将被立即撤销，轴不正向运动。此时投送减缓正向限位的指令可投送成功，轴可向减缓正向限位的方向运动；

（2）轴到达反向限位（含软限位和硬限位）时，投送加重反向限位的指令将被立即撤销，轴不反向运动。此时投送减缓反向限位的指令可投送成功，轴可向减缓反向限位的方向运动；

（3）对于正在执行单轴指令的轴超限，将撤销该指令，并使用 Max(FastDecel,Decel)快速停止当前轴的运动；

（4）对于正在执行插补指令的合轴超限，将撤销整个插补指令，并使用合轴的 Max(FastDecel,Decel)快速停止当前轴的运动；

（5）对于正在执行插补指令的分轴超限，将撤销整个插补指令，超限分轴使用 Max(FastDecel 分轴,Decel 分轴，合轴分解的减速度)快速停止当前分轴的运动。合轴和未超限的分轴不受影响，继续运动；

（6）对于执行联动指令的从轴超限，将撤销该指令，并使用 Max(FastDecel,Decel)快速停止当前从轴的运动；

（7）轴同时到达正向和反向限位（含软限位和硬限位）时，轴指令停止解算但不撤销。当恢复为一种限位时将按照上面几种情况处理。

5.1.3 位置相关参数（Positions）

5.1.3.1 MPos（反馈位置）

5.1.3.1.1 基本属性

类型	LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	浮点数范围
相关章节	DPos（本周期目标位置） RepMode（位置行程模式） RepDist（循环行程长度） HMC_DefPos（设置当前位置）

5.1.3.1.2 含义描述

当前测量反馈位置，对于有实际输入信号的轴类型，MPos 反映了当前实际位置。用户单位。

（1） Mpos 取值原则

轴类型有实际输入信号时，如 EtherCAT 总线轴，其 Mpos 来自于编码器的实时测量值；但若当 FeedBackSrcMode=1 时，Mpos 来自用户自定义输入源。

轴类型为虚轴时，其 MPos 等于 DPos。

（2） Mpos 计算方式

Mpos 均采用增量计算方法。将前后两周期的编码器反馈值的增量累积到上周期 MPos 值上，得到本周期 MPos 值。

5.1.3.2 DPos（本周期目标位置）

5.1.3.2.1 基本属性

类型	LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	浮点数范围
相关章节	RepMode（位置行程模式） RepDist（循环行程长度）

5.1.3.2.2 含义描述

当前目标位置，是算法实时计算出来的本周期目标位置，据此得到最终的输出信号，用户单位。

系统按照以下几个原则处理该轴参数：

- 系统运行过程中实时更新该轴参数，该参数的值由运动指令实时解算；
- 在循环行程模式下，MPos 始终在[0,RepDist)或[-RepDist,RepDist)之间，Dpos 和 Mpos 之间保持实际的距离，这样 Dpos 大部分情况下也在[0,RepDist)或[-RepDist,RepDist)之间，一些个别情况，例如 RepDist 值设置的比较小的时候，总线轴的 Dpos 会出现在循环行程区间之外；
- 对于 EtherCAT_Speed 轴（轴类型为 51），当 ServoLoop 为 0 时，Dpos 实时等于 Mpos。

5.1.3.3 FE（偏差）

5.1.3.3.1 基本属性

类型	LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	浮点数范围
相关章节	StopFETol（反馈位置是否到达目标位置的死区）

	FELimits（随动误差限制）

5.1.3.3.2 含义描述

实时的随动误差，FE=DPOS-MPOS。

伺服模式时，会根据 FE 进行 PID 运算，得到最终输出值；对于总线位置模式时，该 FE 不参与 PID 运算，但用来进行偏差报警。当轴类型为非伺服轴、非总线轴时，该值始终为 0。

5.1.3.4 StopFETol（反馈位置是否到达目标位置的死区）

5.1.3.4.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	10
取值范围	正的浮点数
相关章节	FE（偏差）

5.1.3.4.2 含义描述

FE 误差容忍范围，用于判断速度模式时 MPos 是否到达目标位置。

当轴类型为速度模式时，即包括伺服轴、总线速度轴类型，如果 CmdStopMode=1，则判断指令是否结束时，除了需满足 DPos 解算完成到达 EndPos，还需满足 Abs(FE)<StopFETol，即认为 Mpos 到达指令目标位置；如果 CmdStopMode=0 则只需满足 DPos 解算完成到达 EndPos，就认为指令结束。

5.1.3.5 RepMode（位置行程模式）

5.1.3.5.1 基本属性

类型	SINT
读写属性	读写

设置函数	-
初始值	2
取值范围	0：有限行程，此时 RepDist 无意义，FsLimit 和 RsLimit 为软限位，硬限位通过函数 HMC_HwLimitConfig 来配置 1：循环行程，FsLimit/RsLimit、FHLimitIn/RHLimitIn 无意义，[0,RepDist)为循环行程上下限值 2：循环行程，FsLimit/RsLimit、FHLimitIn/RHLimitIn 无意义，[-RepDist,RepDist)为循环行程上下限值
相关章节	RepDist（循环行程长度） MPos（反馈位置）

5.1.3.5.2 含义描述

设置该轴的位置行程模式：循环行程/有限行程。

（1） 有限行程定义

表示轴只在一个限定的范围内运动，不能超过该限定的范围，例如由滚珠丝杆带动的运动物体，只能在有限的行程范围内运动。

当用户设置 RepMode=0 时，代表该轴为有限行程，此时 RepDist 无意义，该轴存在行程限位。

FsLimit 和 RsLimit 表示软限位，而硬限位通道（FHLimitIn/RHLimitIn）通过函数 HMC_HwLimitConfig 来配置。当有限行程的轴超过硬限位或者软限位时，系统的处理方式详见“有限行程轴超限处理”章节。

（2） 循环行程定义

轴在不断转动，其位置在不断的重复，没有限位，例如钟表的运动。

当用户设置 RepMode=1 或 RepMode=2 时，代表该轴为循环行程。其中 RepMode= 1 时，[0,RepDist)为循环行程上下限值， RepMode=2 时，[-RepDist,RepDist)为循环行程上下限值。

循环行程时，软限位 FsLimit/RsLimit 和硬限位通道（FHLimitIn/RHLimitIn）无意义。

（3） 轴类型设置后自动修改行程方式

当用户通过函数 HMC_SetAxisType 设置某个轴类型后，系统会自动修改该轴的行程模式。当设置为虚轴或编码器轴时，系统将设置 RepMode=2；当用户设置为其他轴类型时，系统将设置 RepMode=0。

5.1.3.6 RepDist（循环行程长度）

5.1.3.6.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	10,000,000,000
取值范围	正浮点数
相关章节	RepMode（位置行程模式）

5.1.3.6.2 含义描述

循环行程模式（RepMode 为 1 或 2 时）时有效：EndPos 和 MPOS 显示的数值范围限值。伺服轴时 DPos 有可能会有短时间超出循环行程的情况。

当用户设置 RepMode=1 时，代表该轴为循环行程，[0,RepDist]为循环行程上下限值；当用户设置 RepMode=2 时，代表该轴为循环行程，[-RepDist,RepDist]为循环行程上下限值。

5.1.3.7 Remain（剩余位置）

5.1.3.7.1 基本属性

类型	LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	浮点数范围
相关章节	-

5.1.3.7.2 含义描述

当前运动指令执行过程中，离指令目标位置的剩余位移，用户单位。系统运行过程中实时更新该轴参数，Remain=EndPos-DPos。

5.1.4 速度相关参数（Velocity Profile）

5.1.4.1 Speed（目标速度）

5.1.4.1.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	2000
取值范围	正浮点数
相关章节	-

5.1.4.1.2 含义描述

设定指令解算的目标速度，用户单位/秒。该参数可以实时修改，该值只可为正或 0，如果设置为负值，系统自动取其绝对值，同时在 AxisStatus 的 Bit 2 位报一个警告。

轴正在执行运动指令时，Speed 可以被实时修改，系统处理方法如下：

（1） 执行单轴指令时，Speed 可以实时修改，且实时生效，指令将按照新的 Speed 进行运动解算；

（2） 执行插补指令且是插补合轴时，Speed 可以实时修改，实时生效，合轴将按照新的 Speed 进行运动解算；因合轴分轴插补关系不变，这些实时解算出来的运动量自然会被分解到分轴上去；

（3） 执行插补指令且是插补分轴时，Speed 可以被实时修改，但不会影响该插补分轴的运动；

（4） 执行联动指令且该轴此时是联动从轴时，Speed 可以被实时修改，但不会影响该轴的运动。

5.1.4.2 Accel（加速度）

5.1.4.2.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-

初始值	200,000
取值范围	正浮点数
相关章节	-

5.1.4.2.2 含义描述

加速度，用户单位/（秒*秒）。

该值只可为正，如果设置为负值，系统自动取其绝对值；如果为 0，该参数恢复系统默认值；对于非法值，同时在 AxisStatus 的 Bit 2 位报一个警告。

某个轴正在执行运动指令时，Accel 可以被实时修改，系统处理方法如下：

（1） 执行单轴指令时，Accel 可以实时修改，实时生效，指令将按照新的 Accel 进行运动解算；

（2） 执行插补指令且是插补合轴时，Accel 可以实时修改，实时生效，合轴将按照新的 Accel 进行运动解算；因合轴分轴插补关系不变，这些实时解算出来的运动量自然会被分解到分轴上去；

（3） 执行插补指令且是插补分轴时，Accel 可以被实时修改，但不会影响该插补分轴的运动；

（4） 执行联动指令且该轴此时是联动从轴时，Accel 可以被实时修改，但不会影响该轴的运动。

5.1.4.3 Decel（减速度）

5.1.4.3.1 基本属性

类型	**LREAL**
读写属性	读写
设置函数	-
初始值	200,000
取值范围	正浮点数
相关章节	-

5.1.4.3.2 含义描述

减速度，用户单位/（秒*秒）。

该值只可为正，如果设置为负值，系统自动取其绝对值；如果为 0，该参数恢复系统默认值；对于非法

值，同时在 AxisStatus 的 Bit 2 位报一个警告。

某个轴正在执行运动指令时，Decel 可以被实时修改，系统处理方法如下：

（1）执行单轴指令时，Decel 可以实时修改，实时生效，指令将按照新的 Decel 进行运动解算；

（2）执行插补指令且该轴是插补合轴时，Decel 可以实时修改，实时生效，合轴将按照新的 Accel 进行运动解算；因合轴分轴插补关系不变，这些实时解算出来的运动量自然会被分解到分轴上去；

（3）执行插补指令且该轴是插补分轴时，Decel 可以被实时修改，但不会影响该插补分轴的运动；

（4）执行联动指令（该轴此时是联动从轴）时，Decel 可以被实时修改，但不会影响该轴的运动。

5.1.4.4 MpSpeed（反馈速度）

5.1.4.4.1 基本属性

类型	LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	浮点数
相关章节	-

5.1.4.4.2 含义描述

实时测量到的反馈速度，测速周期为系统伺服周期，单位同 Speed。

MpSpeed 的计算原则为：

（1） EtherCAT_Position（轴类型为 50）、EtherCAT_Speed（轴类型为 51）、EtherCAT_Torque（轴类型为 52）、EtherCAT_Encoder（轴类型为 53），MpSpeed 是根据来自编码器的实时测量位置值计算得到，即此时反映了 MPos 的变化速度；

（2） 对于其他轴，该值在数值上等于 VpSpeed；

（3） MpSpeed 的正负值表示当前轴的实际运动方向。

5.1.4.5 VpSpeed（解算速度）

5.1.4.5.1 基本属性

类型	LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	正浮点数
相关章节	Direction（解算速度的方向）

5.1.4.5.2 含义描述

运动指令实时解算出来的速度数值，解算周期为系统伺服周期，单位同 Speed。其反映了 Dpos 的变化速度值，该参数仅表示速度值，不包含方向，即为正值。

5.1.4.6 Direction（解算速度的方向）

5.1.4.6.1 基本属性

类型	SINT
读写属性	只读
设置函数	-
初始值	1
取值范围	1：表示正在向正向运动 -1：表示正在向负向运动
相关章节	VpSpeed（解算速度）

5.1.4.6.2 含义描述

运动指令实时解算出来的速度方向，反映了 DPos 的速度变化方向。

5.1.4.7 Merge（融合功能模式）

5.1.4.7.1 基本属性

类型	USINT
读写属性	只读
设置函数	HMC_SetMerge
初始值	0
取值范围	该值只可为 0、1、2、3、4 0：Merge 关闭，每个指令块单独完成各自的运动 1：Merge 打开，当前指令块结束时目标速度为该指令块的 Speed 2：Merge 打开，当前指令块结束时目标速度为下一指令块的 Speed 3：Merge 打开，当前指令块结束时目标速度为该指令块 Speed 和下条指令块 Speed 最小值 4：Merge 打开，当前指令块结束时目标速度为该指令块 Speed 和下条指令块 Speed 最大值
相关章节	运动前瞻

5.1.4.7.2 含义描述

设置速度融合功能，决定是否开启前瞻预处理的速度规划策略。Merge（融合功能）是将轴运动控制指令缓存中一连串运动指令连贯起来，使负载运动速度连续，告别走走停停，可以显著提升效率。

5.1.4.8 CreepSpeed（原点回归爬行速度）

5.1.4.8.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	200
取值范围	正浮点数
相关章节	HMC_Home（原点回归）

5.1.4.8.2 含义描述

爬行速度，用户单位/秒。

用于原点回归往回找原点阶段，在此阶段系统会将 CreepSpeed 作为目标速度进行运动解算，用户在此阶段修改 CreepSpeed 将实时生效。

该值只可为正或 0，如果给负值，取其绝对值，同时在 AxisStatus 的 Bit 2 位报一个警告。

5.1.4.9 FastDecel（快速减速度）

5.1.4.9.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	5,000,000
取值范围	正浮点数

5.1.4.9.2 含义描述

快速减速度，用户单位/（秒*秒），用于有限行程轴发生软超限或硬超限后的自动紧急停止。详见有限行程轴超限处理章节。

该值只可为正（如果为负值，取其绝对值；如果为 0，该参数恢复系统默认值；对于非法值，同时在 AxisStatus 的 Bit 2 位报一个警告）。

5.1.4.10 Jerk（加加速度）

5.1.4.10.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	5,000,000,000
取值范围	正浮点数
相关章节	HMC_ClcJerk（冲击运算）

5.1.4.10.2 含义描述

加加速度（加速度或减速度的导数），用户单位/（秒*秒*秒）。

该值只可为正。如果为负值，取其绝对值；如果为 0，该参数恢复系统默认值；对于非法值，在 AxisStatus 的 Bit 2 位报一个警告。

5.1.4.11 JerkMode（加速模式）

5.1.4.11.1 基本属性

类型	USINT
读写属性	只读
设置函数	-
初始值	0
取值范围	0：S 形运动禁用 1：S 形运动使能
相关章节	-

5.1.4.11.2 含义描述

S 形运动使能/禁用（是否使用 Jerk）。

当不使用 Jerk 时，将按照加速度 Accel 和减速度 Decel 进行加速或减速运动，此时速度曲线表现为梯形；当使用 Jerk 时，将按照 Jerk 作为加加速度 Jerk 来改变加（减）速度和速度来进行运动，一般情况下，速度曲线表现为“S”形。

5.1.4.12 CmdStopMode（指令结束判断方式）

5.1.4.12.1 基本属性

类型	USINT
读写属性	读写
设置函数	-
初始值	0

取值范围	0 1
相关章节	-

5.1.4.12.2 含义描述

判断伺服轴的运动指令是否执行结束时的模式。

（1） CmdStopMode=0

系统判断当前指令运算结束的标准是：DPOS 已经到达目标位置 EndPos。即达到该条件后，认为当前指令运算结束，立即开始执行下一条指令。

（2） CmdStopMode=1

系统判断当前指令运算结束必须同时满足以下两个条件，才会执行下一条指令。

- DPOS 已经到达目标位置
- MPOS 到达目标位置或者 DPOS 到达目标位置已经足够长时间（默认 500ms，也可通过 HMC_SetCmdStopMaxTime 指令修改此时间）。MPos 到达目标位置是指 Abs(FE)<StopFETol。例如系统控制周期为 1 ms，则 DPOS 到达目标位置 500 ms 后，即使 MPOS 没有到达指定位置，系统也认为该指令结束了。

5.1.4.13 MoveAbsMode（循环行程轴绝对运动时方向选取方式）

5.1.4.13.1 基本属性

类型	**USINT**
读写属性	读写
设置函数	-
初始值	0
取值范围	0、1、2、3、4
相关章节	-

5.1.4.13.2 含义描述

仅对循环行程轴（RepMode=1 或 RepMode=2）的绝对运动指令有效，例如 HMC_MoveAbs、

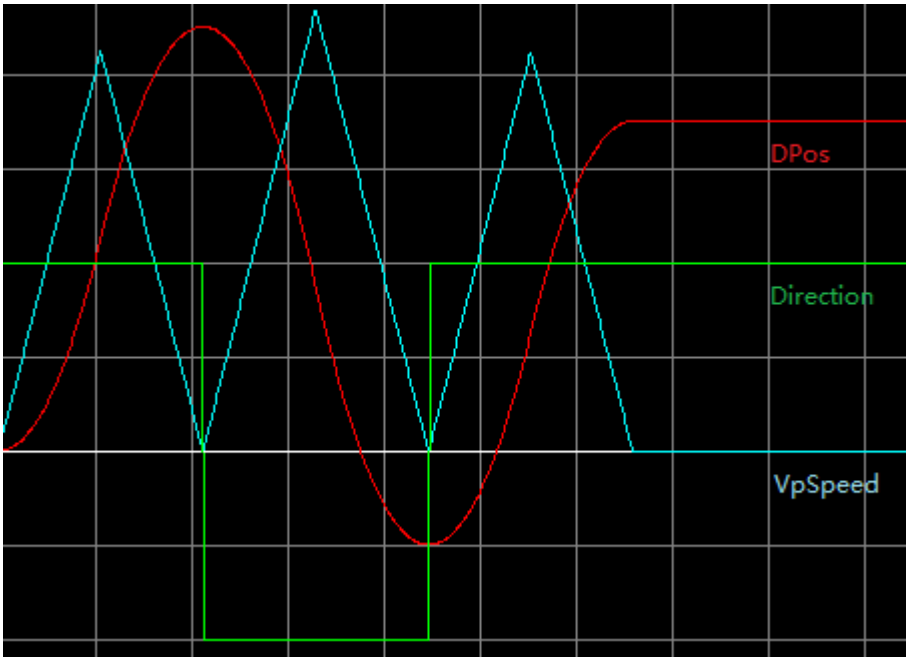
HMC_MoveAbs2DEx、HMC_MoveAbsND。

- MoveAbsMode=0：在本次循环行程内确认方向
- MoveAbsMode=1：算法选取距离最短的方向，正负向运动等距离时，选取正方向
- MoveAbsMode=2：按照正方向运动
- MoveAbsMode=3：按照负方向运动
- MoveAbsMode=4：按照当前运动方向运动。即 MoveAbs 之前的 Direction 方向即为本次 MoveAbs 的方向

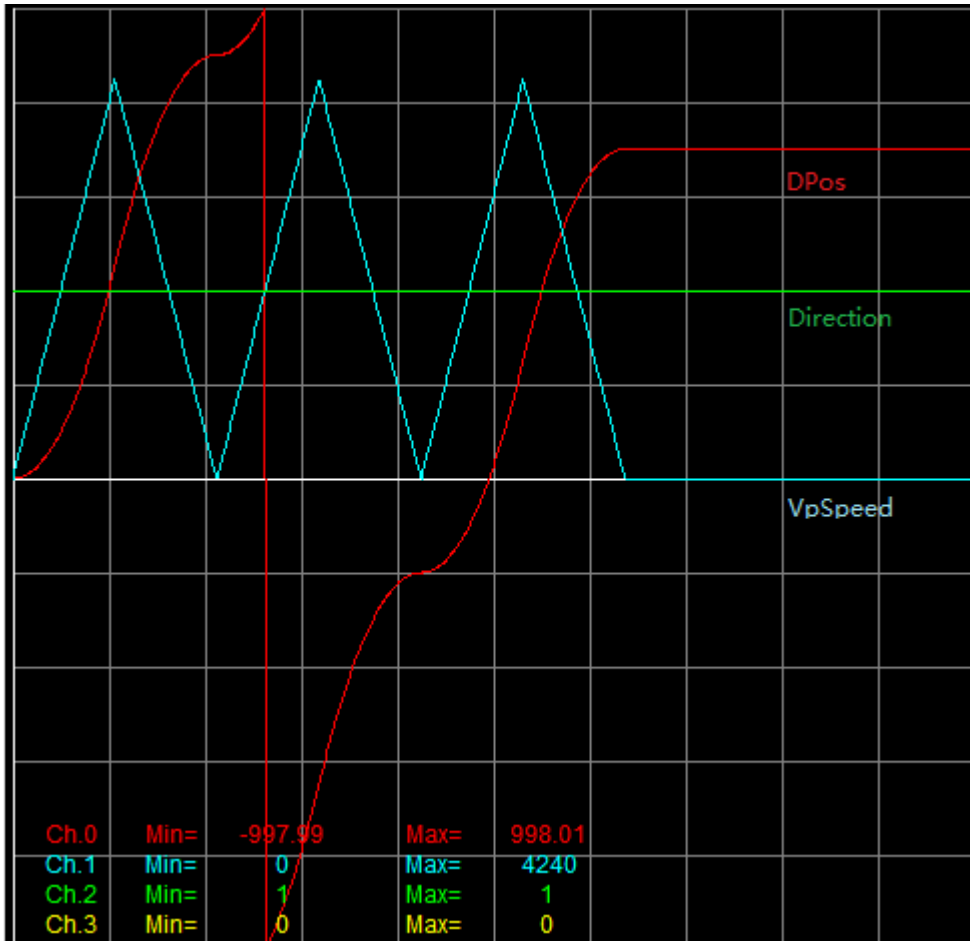
以下例详细描述以上几个模式时，方向确定方法。

编写如下程序，然后分别修改 MoveAbsMode，查看轴 0 的运行情况。

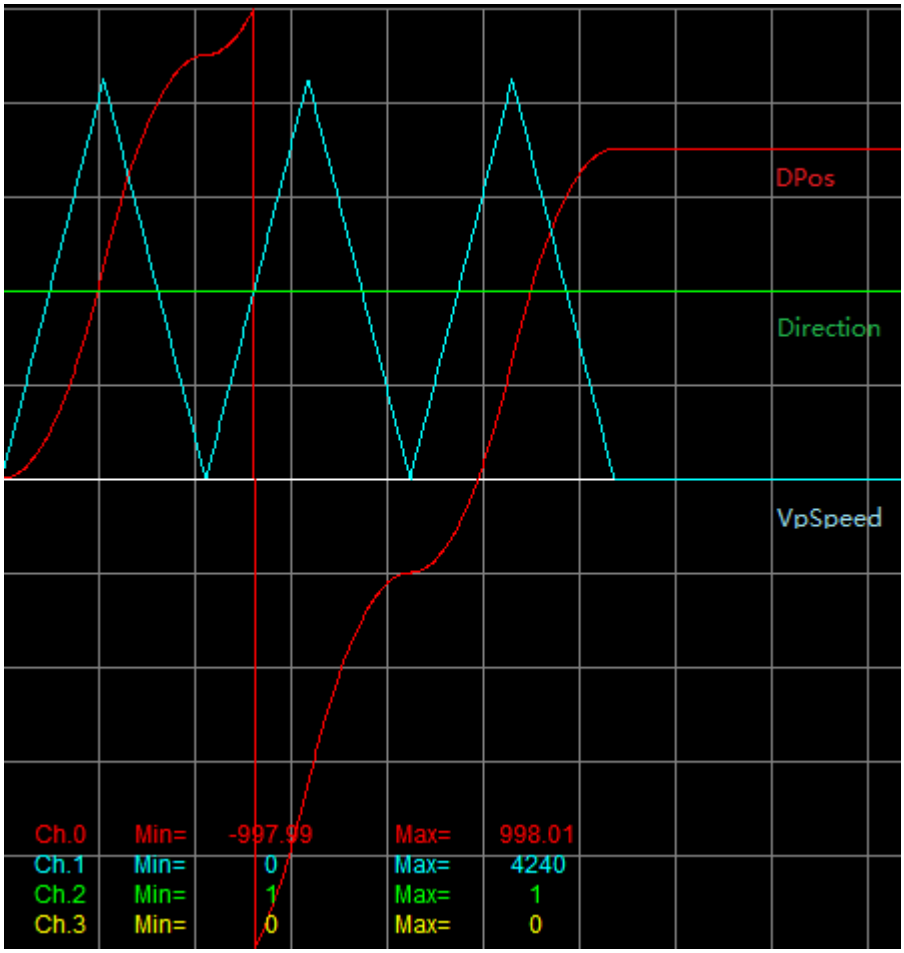
```
axis0.RepMode := 2;
axis0.RepDist := 1000;
hmc_defpos(0,0);
hmc_moveabs(0, 900);
hmc_moveabs(0, -200);
hmc_moveabs(0, 700);
```



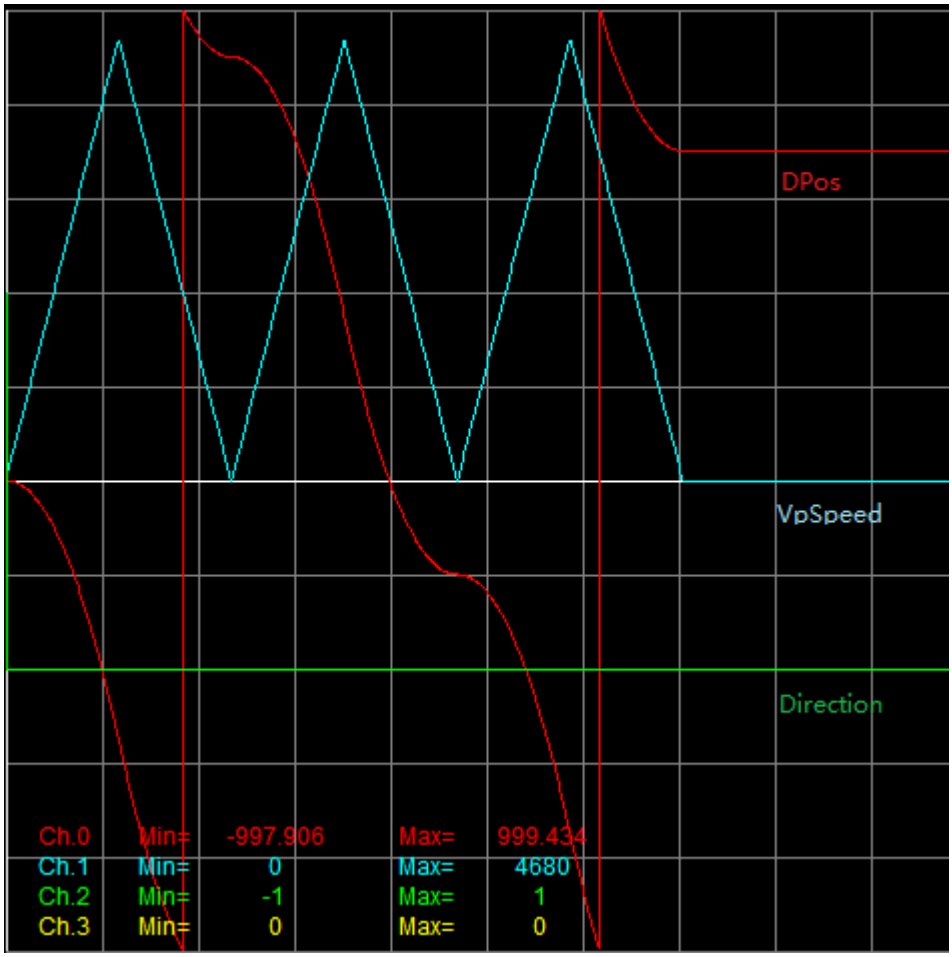
MoveAbsMode=0 在循环行程内确认方向



MoveAbsMode=1 取最短距离方向



MoveAbsMode=2 始终正方向



MoveAbsMode=3 始终负方向

5.1.4.14 CancelMode（Cancel 时指令停止模式）

5.1.4.14.1 基本属性

类型	USINT
读写属性	读写
设置函数	-
初始值	0
取值范围	0、1、2
相关章节	CancelPos（Cancel 时停止位置） HMC_Cancel（撤销指令）

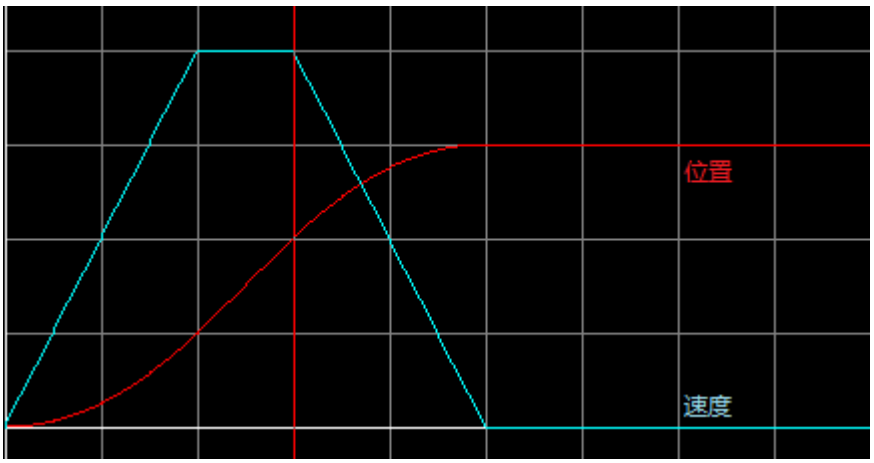
5.1.4.14.2 含义描述

Cancel 时指令停止模式。

指令撤销过程中，在接受到 Cancel 指令后，轴会以当前速度开始进行减速直到停止。该减速停止过程中具体停止方式，由轴参数 CancelMode 指定。具体如下：

（1） CancelMode=0 自然停止

默认值，按照 Decel 自然停止。此模式支持所有指令类型。

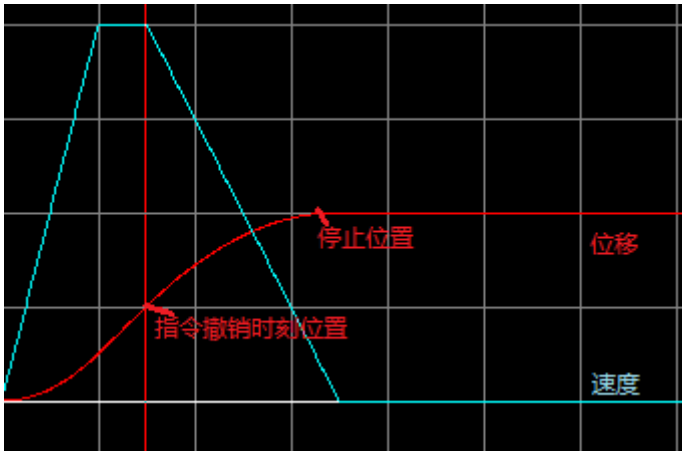


指令停止示意图

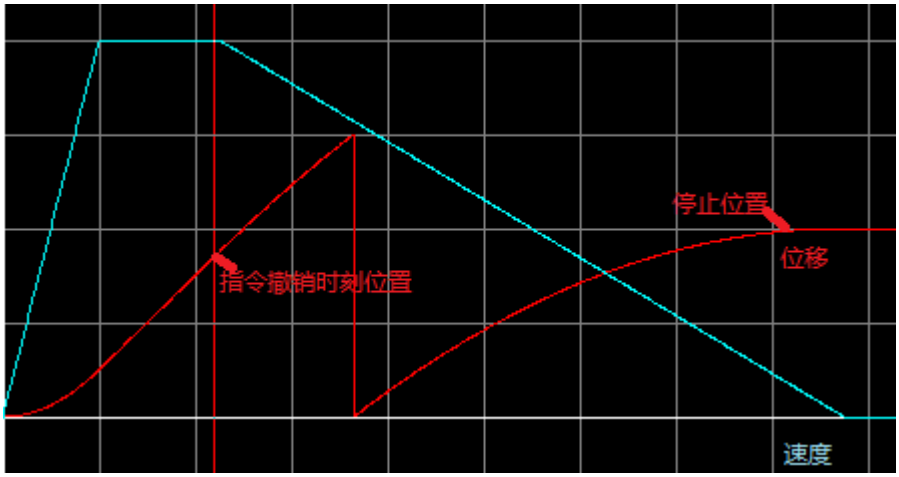
（2） CancelMode=1 指定位置停止

按照不超过减速度 Decel 停到指定位置，该位置由参数 CancelPos 指定。此模式仅对循环行程模式(RepMode=1 或 2)且当前指令为单向运动（HMC_Forward、HMC_Reverse）的情况生效。若单向运动指令，但不满足循环行程模式，则按照 CancelMode=0 处理。

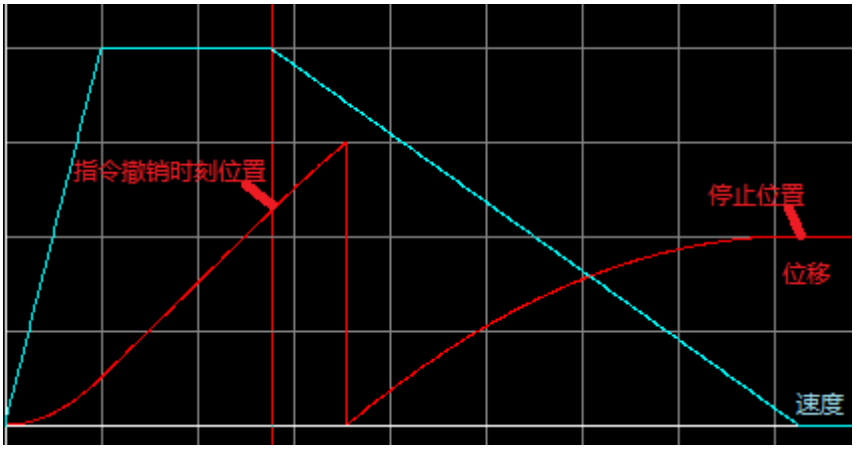
根据当前位置和 CancelPos 的相对关系，会有三种情况：1、当前位置<停止位置，且两者距离位移差足够当前速度以减速度停止为 0；2、当前位置<停止位置，但两者距离位移差不够当前速度以减速度停止为 0；3、当前位置>停止位置。以上三种情况分别对应如下三图：



指令停止示意图



指令停止示意图

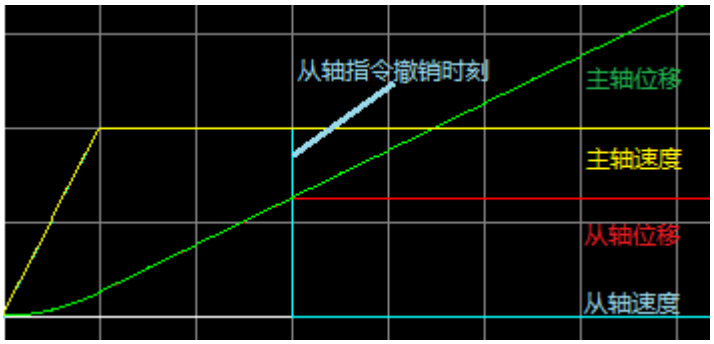


指令停止示意图

(3) CancelMode=2 立即停止

立即停止，速度从当前速度直接减速为 0。此模式仅对正在执行联动指令的轴生效，若状态不满足，

则默认按照 CancelMode=0 处理。



指令停止示意图

5.1.4.15 CancelPos（Cancel 时停止位置）

5.1.4.15.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	该值应在循环行程的上下限之间
相关章节	CancelMode（Cancel 时指令停止模式）

5.1.4.15.2 含义描述

循环行程轴且指令为 HMC_Forward 或 HMC_Reverse，同时 CancelMode=1 时，该参数有效，该参数表示在指令 Cancel 完成时轴将到达的位置。但 CancelPos 超行程限时，按照 CancelMode=0 处理。

5.1.5 系数因子参数（Gains）

5.1.5.1 Kp（PID 的比例系数）

5.1.5.1.1 基本属性

类型	LREAL
读写属性	读写

设置函数	-
初始值	1
取值范围	浮点数

5.1.5.1.2 含义描述

当控制方式为位置闭环模式，需要经过 PID 模型运算，Kp 即 PID 模型的比例系数，其作用与比例项，比例项作用=Kp*Fe。

5.1.5.2 Ki（PID 的积分增益）

5.1.5.2.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	浮点数

5.1.5.2.2 含义描述

当控制方式为位置闭环模式，需要经过 PID 模型运算，Ki 即 PID 模型的积分增益，其作用与积分项，积分项作用=Ki*(∑(Fe*dt))。

5.1.5.3 Kd（PID 的微分增益）

5.1.5.3.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	浮点数

5.1.5.3.2 含义描述

当控制方式为位置闭环模式，需要经过 PID 模型运算，Kd 即 PID 模型的微分增益，其作用与微分项，微分项作用= Kd*(d(Fe)/dt)。

5.1.5.4 Kov（测量速度增益）

5.1.5.4.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	浮点数

5.1.5.4.2 含义描述

当控制方式为位置闭环模式，会在 PID 计算输出基础上叠加测量速度增益项。测量速度增益作用=Kov*MpSpeed。

5.1.5.5 Kvff（速度前馈增益）

5.1.5.5.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	浮点数

5.1.5.5.2 含义描述

当控制方式为位置闭环模式，会在 PID 计算输出基础上叠加速度前馈增益项。速度前馈增益作用=Kvff*((+1/-1)*VpSpeed)。

5.1.5.6 Kaff（加速度前馈增益）

5.1.5.6.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	浮点数

5.1.5.6.2 含义描述

加速度前馈增益，加速度前馈=Kaff*解算加速度。

5.1.5.7 Ko（PID 输出放大因子）

5.1.5.7.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	1
取值范围	浮点数

5.1.5.7.2 含义描述

伺服输出比例，PID 计算后伺服输出值的放大因子。

5.1.6 限幅限位参数（Limits）

5.1.6.1 FELimits（随动误差限制）

5.1.6.1.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	1000
取值范围	正的双精度浮点数（64 位）
相关章节	-

5.1.6.1.2 含义描述

FE（随动误差）的正常范围。当 Abs(FE)>FELimit 时，则认为 FE 超限。当 FE 超限时，系统将会将轴参数 AxisStatus 中 bit0 置 1，并关闭伺服使能。如果要屏蔽 FE 超限关闭伺服使能动作，可以把 AxisErrMask 的 bit0 置 0。

5.1.6.2 FsLimitMode/RsLimitMode（前/后限位时指令撤销方式）

5.1.6.2.1 基本属性

类型	USINT
读写属性	只读
设置函数	-
初始值	0
取值范围	0：遇到前/后向限位后自动撤销当前指令 1：遇到前/后向限位后手动撤销当前指令(暂不支持)
相关章节	-

5.1.6.2.2 含义描述

表示前/后向限位控制模式，对软限位及硬限位均有效。该参数仅对有限行程的轴是有效的，对于循环

行程的轴无效。

当一个有限行程轴超限时，会开始紧急停止动作。当前指令如何处理由该轴相关参数决定，详见“异常处理”章节中“有限行程轴超限处理”。

0：遇到前/后向限位后自动撤销当前指令。需要特别说明的是，对插补指令而言，无论合轴还是分轴遇到限位时，整个插补指令都将被撤销。

1：遇到前/后向限位后手动撤销当前指令。需要特别说明的是，当前指令在遇到函数 HMC_Cancel 之前是不会被撤销的，在轴完成紧急停止后，当原运动指令解算出来的 DPos 值加重超限时，轴继续停止不动；当原运动指令解算出来的 DPos 值减缓超限时，轴将按照 DPos 值运动。

这里的“加重超限”指：当超越前向软/硬软限位时，当前指令解算的 DPos 值越来越大；后者当超越后向软/硬软限位时，当前指令解算的 DPos 值越来越小。直观来讲，就是超限的越来越严重。“减缓超限”的含义刚好与之相反。

5.1.6.3 FHLimitIn/RHLimitIn（前/后向硬限位通道号）

5.1.6.3.1 基本属性

类型	UDINT
读写属性	只读
设置函数	HMC_HwLimitConfig
初始值	4294967295
取值范围	未启用：4294967295 其他值：DI 通道号
相关章节	-

5.1.6.3.2 含义描述

设置前/后向硬限位行程开关所对应的 DI 通道号。

循环行程时（RepMode =1 或 2）无意义。

5.1.6.4 FSLimit/RSLimit（前/后向软限位边界值）

5.1.6.4.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	RSLimit：-1e100 FSLimit：1e100
取值范围	双精度浮点数（64 位）。需确保 RSLimit< FSLimit。
相关章节	-

5.1.6.4.2 含义描述

有限行程的前/后向软限位界限设定。循环行程时（RepMode =1 或 2）无意义。

5.1.6.5 DACLimitHigh/DACLimitLow（DAC 输出上限/下限）

5.1.6.5.1 基本属性

类型	DINT
读写属性	读写
设置函数	-
初始值	DACLimitLow：-2147483648 DACLimitHigh：2147483647
取值范围	双精度浮点数（64 位） 需满足 DACLimitHigh>DACLimitLow
相关章节	-

5.1.6.5.2 含义描述

DACOut 的上限/下限，确保 DACOut 的输出不会超过设定范围。

5.1.7 原点和保持参数（Home&Hold）

5.1.7.1 HomeState（原点回归状态）

5.1.7.1.1 基本属性

类型	USINT
读写属性	只读
设置函数	-
初始值	0
取值范围	以下描述与 HMC_Home 或 MC_Home 指令的 HomeMode 参数相关。 0：尚未开始寻找原点 1：原点回归快速查找减速点状态 2：原点回归找到减速点后快速停止状态 3：原点回归慢速查找原点状态 4：原点回归慢速查找 Z 信号状态 5：找到原点信号后慢速停止状态 6：运动至原点信号位置状态 7：查找到限位开关快速停止状态 8：碰到限位开关反向快速查找原点状态 9：高速找到原点 DI 后减速到低速 10：反向快速查找到原点后停止状态 11：原点回归完成状态 12：原点回归指令被打断
相关章节	HMC_Home（原点回归） MC_Home（原点回归）

5.1.7.1.2 含义描述

原点回归状态，用于指示 HMC_Home、MC_Home 在原点回归过程所处状态。原点回归过程需经历 0-12 状态，各个状态含义见上标取值范围。用户可通过查看该参数，清楚了解当前原点回归情况。

5.1.7.2 HomeIn（原点回归所用的 DI 通道）

5.1.7.2.1 基本属性

类型	UDINT
读写属性	只读
设置函数	HMC_Home、HMC_HomeEx（暂不支持）
初始值	4294967295
取值范围	取值为 4294967295 时表示未使用原点回归功能 其他值表示 DI 通道号
相关章节	HMC_Home（原点回归）

5.1.7.2.2 含义描述

原点回归所用的 DI 通道号，该参数值是在投放 HMC_Home、HMC_HomeEx 指令时设置。

5.1.7.3 HomeCapIn（原点回归使用的高速捕获通道）

5.1.7.3.1 基本属性

类型	INT
读写属性	只读
设置函数	HMC_HomeEx（暂不支持）
初始值	-1
取值范围	暂不支持

5.1.7.3.2 含义描述

HMC_HomeEx 所使用的高速捕获通道号，在投放 HMC_HomeEx 指令时设置。

5.1.7.4 HoldSpeed（强制速度）

5.1.7.4.1 基本属性

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	正浮点数
相关章节	HMC_HoldConfig(设置强制速度开关)

5.1.7.4.2 含义描述

当速度强制开关被触发时，轴会从当前速度加减速至 HoldSpeed。一种常用的使用方法是设置 HoldSpeed=0，当 Hold 触发源（HoldIn）为 1 时，Speed 被强制成 0，这样可以实现“暂停”功能。

速度强制功能详细描述可参考章节 HMC_HoldConfig(设置强制速度开关)。

5.1.7.5 HoldIn（速度强制开关 DI 通道号）

5.1.7.5.1 基本属性

类型	UDINT
读写属性	只读
设置函数	HMC_HoldConfig
初始值	4294967295
取值范围	4294967295 时表示关闭该功能 其他值表示 DI 通道号
相关章节	HMC_HoldConfig(设置强制速度开关)

5.1.7.5.2 含义描述

速度强制功能的触发源 DI 通道号。

5.1.8 输入相关参数（Input）

5.1.8.1 KeNumerator/KeDenominator（编码器输入比例 Ke 的分子/分母）

5.1.8.1.1 基本属性

类型	DINT
读写属性	只读
设置函数	HMC_SetKe
初始值	1
取值范围	DINT 整数：分子、分母均不为 0，但可异号
相关章节	HMC_SetKe（设置 Ke）

5.1.8.1.2 含义描述

Ke 的分子和分母。参数 Ke 仅对有实际位置反馈的轴类型起作用（总线轴），可为负。该参数和 Units 有类似的作用，(Ke /Units)共同构成了伺服轴或编码器轴的“单位转换因子”，其单位为“线/用户单位”。

5.1.8.2 EncoderValue（内部编码器值）

5.1.8.2.1 基本属性

类型	DINT
读写属性	只读
设置函数	-
初始值	0
取值范围	-2147483648～+2147483647
相关章节	-

5.1.8.2.2 含义描述

当该轴通过编码器反馈位置信息时，该轴参数表示内部编码器值，每个伺服周期进行刷新。根据该值

增量计算得到实际位置值 MPos。

5.1.8.3 FeedBackSrcMode（轴位置反馈源模式）

5.1.8.3.1 基本属性

类型	USINT
读写属性	读写
设置函数	-
初始值	0
取值范围	0：反馈输入源为轴实际物理连接输入，如 50、51 时为编码器输入 1：反馈输入源为用户自定义输入
相关章节	HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址） FeedBackSrcValue（轴反馈自定义输入实时值）

5.1.8.3.2 含义描述

设定轴反馈值 Mpos 输入源方式，仅对有实际输入的轴类型生效，例如总线轴类型等。

通过设置 FeedBackSrcMode=0 使得 Mpos 输入源为轴实际物理连接输入；通过设置 FeedBackSrcMode=1，使得 MPos 输入源为用户自定义输入，该用户自定义地址通过 HMC_SetFeedBackSrcAddr 函数设置。

5.1.8.4 FeedBackSrcValue（轴反馈自定义输入实时值）

5.1.8.4.1 基本属性

类型	LREAL
读写属性	只读
设置函数	HMC_FeedBackSrcAddr
初始值	0
取值范围	浮点数
相关章节	HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址） FeedBackSrcMode（轴位置反馈源模式）

5.1.8.4.2 含义描述

对于有实际反馈输入的轴，可通过设置 FeedBackSrcMode=1，调用 HMC_SetFeedBackSrcAddr 配置某参数地址作为此时轴的反馈输入源，即此时该轴的 Mpos 根据该地址对应参数来计算。FeedBackSrcValue 实时显示关联的自定义参数值。

5.1.8.5 FeedBackDataType（轴反馈自定义输入变量数据类型）

5.1.8.5.1 基本属性

类型	USINT
读写属性	只读
设置函数	HMC_SetFeedBackSrcAddr
初始值	0
取值范围	USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、REAL=7、LREAL=8
相关章节	HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址） FeedBackSrcMode（轴位置反馈源模式）

5.1.8.5.2 含义描述

FeedBackDataType 表示用户自定义输入变量的数据类型。FeedBackSrcValue 实时显示关联的自定义参数值。

5.1.9 输出相关参数（Output）

5.1.9.1 KsNumerator/KsDenominator（步进输出比例 Ks 的分子/分母）

5.1.9.1.1 基本属性

类型	DINT
读写属性	只读

设置函数	HMC_SetKs
初始值	1
取值范围	DINT 整数：分子、分母均不为 0，且同号
相关章节	HMC_SetKs（设置 Ks）

5.1.9.1.2 含义描述

步进输出比例 Ks 的分子/分母，通过 HMC_SetKs 设置。该参数仅对 EtherCAT_Position（50）有效。参见 HMC_SetKs（设置 Ks）。

5.1.9.2 PulseNum（步进轴周期输出脉冲数）

5.1.9.2.1 基本属性

类型	UDINT
读写属性	只读
设置函数	-
初始值	0
取值范围	0xffffffff
相关章节	-

5.1.9.2.2 含义描述

每个伺服周期，步进轴输出的脉冲数。

5.1.9.3 ServoLoop（闭环输出开关）

5.1.9.3.1 基本属性

类型	USINT
读写属性	读写
设置函数	-
初始值	0

取值范围	0：表示开环，模拟量输出（DACOut）采用 DAC 参数所给定的数值 1：表示闭环，模拟量输出（DACOut）为 PID 运算的结果
相关章节	-

5.1.9.3.2 含义描述

闭环输出开关，该参数仅对位置闭环模式的轴类型生效，如伺服轴 20、RTEX 总线速度模式 41，EtherCAT 总线速度模式 51 等。通过设置该开关决定位置闭环是否生效：

（1） ServoLoop=0 表示开环

该轴处于位置开环模式。此时 Mpos、MpSpeed 来自于编码器的测量值，而 DPos 跟踪 MPos。输出值 DACOut 采用 DAC 参数所给定的数值，即开环模式。

（2） ServoLoop=1 表示闭环

该轴处于位置闭环控制模式。Dpos、VpSpeed 为运动控制指令实时计算值，Mpos、MpSpeed 是来自于编码器的测量值，通过对 DPos 和 MPos 进行 PID 计算，PID 将计算结果更新至输出值 DACOut，达到位置闭环的控制效果。

需要说明的是：当系统出现异常时，会通过轴参数 AxisStatus 来显示，当 AxisStatus 和 AxisErrMask 的逻辑与运算非零时，系统将进行异常处理，此时，ServoLoop 将被系统强制修改为 0。

5.1.9.4 DAC（速度模式开环输出值）

5.1.9.4.1 基本属性

类型	**DINT**
读写属性	读写
设置函数	-
初始值	0
取值范围	-2,147,483,648~2,147,483,647
相关章节	-

5.1.9.4.2 含义描述

ServoLoop=0 时的 DACOut 的输出值。

5.1.9.5 DACOut（速度模式输出值）

5.1.9.5.1 基本属性

类型	DINT
读写属性	只读
设置函数	-
初始值	0
取值范围	-2,147,483,648~2,147,483,647
相关章节	-

5.1.9.5.2 含义描述

当前轴的速度量输出结果（EtherCAT_Speed（51），该速度量直接通过总线给伺服驱动器）。

5.1.9.6 PosOffset（DACOut 正向死区值）

5.1.9.6.1 基本属性

类型	DINT
读写属性	读写
设置函数	-
初始值	0
取值范围	总线轴：-2,147,483,648~2,147,483,647
相关章节	-

5.1.9.6.2 含义描述

伺服总线轴输出正向死区值。

当伺服总线轴计算的输出值 DACOut 为正时，则在此 DACOut 基础上叠加正向死区值 PosOffset。DACOut 为 0 时，固定输出（PosOffset+NegOffset）/2。即通过设置死区后，最终输出的 DACOut 永远在伺服执行机构的电压死区范围外，避免执行机构长期处于死区影响控制效果的现象。

对于伺服总线轴，DACOut 取值范围为-2,147,483,648~2,147,483,647。需根据执行机构情形进行计算并设置。

5.1.9.6.3 应用举例

若伺服轴执行机构存在死区，其死区对应电压为（-1，2）V，则需设置 NegOffset=-3278，PosOffset=6553。

5.1.9.7 NegOffset（DACOut 负向死区值）

5.1.9.7.1 基本属性

类型	DINT
读写属性	读写
设置函数	-
初始值	0
取值范围	总线轴：-2147483648~2147483647
相关章节	-

5.1.9.7.2 含义描述

伺服轴输出负向死区值。

当伺服轴计算的输出值 DACOut 为负时，则在此 DACOut 基础上叠加负向死区值 NegOffset。DACOut 为 0 时，固定输出（PosOffset+NegOffset）/2。即通过设置死区后，最终输出的 DACOut 永远在伺服执行机构的电压死区范围外，避免执行机构长期处于死区影响控制效果的现象。

5.1.9.8 ZeroWindow（DACOut 零点间隙死区）

5.1.9.8.1 基本属性

类型	DINT
读写属性	读写
设置函数	-
初始值	0
取值范围	总线轴：-2147483648~2147483647
相关章节	-

5.1.9.8.2 含义描述

DACOut 零点间隙死区。DACOut 位于死区窗口[-ZeroWindow, ZeroWindow]时，输出 NegOffset 和 PosOffset 的中值（即（PosOffset+NegOffset）/2）。

5.1.9.9 OutDstMode（DACOut 输出目标模式）

5.1.9.9.1 基本属性

类型	USINT
读写属性	读写
设置函数	-
初始值	0
取值范围	0：轴计算的 DACOut 输出值的目标为实际物理连接。 1：轴计算的 DACOut 输出值的目标为用户自定义变量。
相关章节	HMC_SetOutDstAddr（设置用户自定义输出地址）

5.1.9.9.2 含义描述

对于速度模式的轴类型，如 EtherCAT 总线速度模式 51 等，可以通过设置 OutDstMode 来指定输出值 DACOut 的最终输出目标。

（1） OutDstMode=0

轴计算的 DACOut 输出目标为实际物理连接，如 51 时目标输出为总线输出通道。

（2） OutDstMode=1

轴计算的 DACOut 输出目标为用户自定义变量，自定义变量值通过 OutDstValue 显示。通过 HMC_SetOutDstAddr 设置用户自定义变量。

此时物理输出通道的状态如下：51 模式下，为了安全起见，此时会置总线驱动器速度为 0，使总线驱动器停止转动。

5.1.9.10 OutDstValue（DACOut 输出自定义地址实时值）

5.1.9.10.1 基本属性

类型	LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	-
相关章节	HMC_SetOutDstAddr（设置用户自定义输出地址）

5.1.9.10.2 含义描述

OutDstMode=1 时，OutDstValue 实时显示用户自定义变量数值。

5.1.9.11 OutDstDataType（DACOut 输出自定义变量数据类型）

5.1.9.11.1 基本属性

类型	USINT
读写属性	只读
设置函数	HMC_ SetOutDstAddr
初始值	0
取值范围	USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、REAL=7、LREAL=8

相关章节	HMC_SetOutDstAddr（设置用户自定义输出地址）

5.1.9.11.2 含义描述

设置 OutDstMode=1 时，调用 HMC_SetOutDstAddr 配置轴的 DACOut 输出至用户自定义量，OutDstDataType 表示用户自定义变量的数据类型。OutDstValue 实时显示用户自定义变量数值。

5.1.10 预留参数（Reserved）

5.1.10.1 SystemValue*

5.1.10.1.1 基本属性

类型	SystemValue0、SystemValue1：UDINT SystemValue2、SystemValue3：LREAL
读写属性	只读
设置函数	-
初始值	0
取值范围	-
相关章节	-

5.1.10.1.2 含义描述

用于系统 Debug，或者记录系统的故障信息。目前有部分固定轴号的预留轴参数已经使用，其具体含义如下：

轴号	SystemValue0	SystemValue1	SystemValue2	SystemValue3
0	算法解算时间	算法的小版本号	-	-
1	算法调度间隔时间	最后一次系统错误码	-	-
2	算法最大解算时间		-	-
3	最后一次用户错误码	最后一次用户错误信息	-	-
4	运动控制算法、协议模块的总执行时间	运动控制算法、协议模块总执行时间超出伺服周期的累计次数	-	-

5.2 系统启动指令

5.2.1 HMC_DriveEnable（伺服使能）

5.2.1.1 功能说明

控制与控制器相连的所有伺服驱动器使能信号的开启与关闭。

（1） 对于总线类型伺服驱动器，DriveEnable 信号通过总线通信发出。

（2） DriveEnable 禁止时，运动控制器各轴停止输出。

5.2.1.2 参数

参数	声明	数据类型	说明
bMode	Input	USINT	0：使能禁止 1：使能开启
HMC_DriveEnable	Output	UDINT	0 设置成功 0xFFFFFFFF 函数参数错误

5.2.1.3 指令类型

非轴运动缓存指令。

5.2.1.4 补充说明

- 该指令控制与控制器相连的所有驱动器的伺服使能状态，为伺服使能总开关。对于总线轴类型，还可实现伺服分使能控制，即单独控制某个轴的使能状态，详见总线相关指令 RTEX_DriveEnable、EtherCAT_DriveEnable。
- 在 AutoThink 菜单栏点击使能按钮，也可控制伺服使能状态切换；
- 当某个轴出现某些异常时，其 AxisStatus 对应的 bit 位会被置 1，如果 AxisErrMask 对应的 bit 位也为

1，此时异常处理程序可能会自动禁用 DriveEnable 信号。

5.2.1.5 示例

以下示例通过 HMC_DriveEnable 指令实现伺服使能开启与关闭。

```
HMC_DriveEnable (1);    (*开启伺服使能*)
HMC_DriveEnable (0);    (*关闭伺服使能*)
```

5.2.2 HMC_GetDriveEnableState（获取伺服使能状态）

5.2.2.1 功能说明

获取伺服总使能（DriveEnable）状态。

5.2.2.2 参数

参数	声明	数据类型	说明
HMC_GetDriveEnableState	Output	USINT	0：禁用 1：启用

5.2.2.3 指令类型

非轴运动缓存指令。

5.2.2.4 补充说明

对于总线轴类型，还可获取某个轴的伺服分使能状态，详见 EtherCAT_ GetDriveEnableState。

5.2.2.5 示例

以下示例通过 HMC_GetDriveEnableState 指令获取伺服总使能（DriveEnable）状态。

State:=HMC_GetDriveEnableState (); (*获取伺服总使能状态*)

5.3 单轴运动指令

5.3.1 原点搜寻及位置定义

5.3.1.1 HMC_Home（原点回归）

5.3.1.1.1 功能

设备在上电后正常工作前一般首先需要寻找机械零点，本函数通过检测原点开关信号或伺服电机编码器 Z 相信号为该轴确定原点，原点回归方式由参数 ucHomeMode 确定。原点开关信号的获取使用的是控制器的某个快速/慢速 DI 通道，也可用虚拟 DI 通道模拟产生；Z 相信号从原点回归轴 DB9 接口的 Z 相接口获取。

慢速 DI 由于响应频率较低，原点回归精度不高，可使用快速 DI 通道提高回归精度。受安装精度及振动因素影响，编码器 Z 相信号比常用的开关传感器（如接近开关、机械式行程开关等）信号具有更高的精度与可靠性，因此在精度要求较高的场合，可使用 原点开关+Z 相信号相结合的方式进行原点回归。

5.3.1.1.2 参数说明

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucHomeMode	Input	USINT	支持 CIA402 规约中如下回零模式：1/2/3/5/7/11/17/18/19/21/23/27/33/34/35 注：原点回归模式说明详见 PLCopen 运动控制指令中的 MC_Home 指令
ucDiChl	Input	UDINT	原点回归时使用的 DI 通道号范围为 0-2097151
HMC_Home	Output	UDINT	-

5.3.1.1.3 指令类型

轴运动缓存指令。

5.3.1.1.4 注意事项

- Z 信号相关的 Home 模式只支持 EtherCAT 总线伺服轴。
- HMC_Home 设计采用查询机制获取触发信号，查询周期为一个伺服周期，故触发信号高电平持续时间≥一个伺服周期时，可在每个查询周期下捕获到触发信号。如果触发信号高电平持续时间<一个伺服周期时，不能保证每个查询周期下都能捕获到原点信号，影响原点回归正常工作。

5.3.1.1.5 示例

参见 PLCopen 指令 MC_Home 示例说明。

5.3.1.2 HMC_DefOrigin（设置原点）

5.3.1.2.1 功能

移动坐标系，改变某轴坐标系的原点，将参数 dMpos 所标定的原坐标系下的位置设置为新的坐标系原点。

5.3.1.2.2 输入参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴 ID
dMpos	Input	LREAL	将会被定义为原点的位置
HMC_DefOrigin	Output	UDINT	返回值

5.3.1.2.3 指令类型

非轴运动缓存指令。

5.3.1.2.4 示例

以下示例通过 HMC_DefOrigin 指令实现设置轴的原点。

HMC_DefOrigin(Axis0.AxisID,1000); (*将轴 Axis0 坐标为 1000 的位置设定为新的坐标原点*)

5.3.1.3 HMC_DefPos（设置当前位置）

5.3.1.3.1 功能

移动坐标系，将当前位置定义为 dMpos。

在轴高速运动的时候调用该函数，由于控制器处理时序的问题可能会产生偏差，此时要确保原点位置定义精确，可采用 HMC_DefOrigin 指令实现。

5.3.1.3.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dMpos	Input	LREAL	当前位置指定值
HMC_DefPos	Output	UDINT	返回值

5.3.1.3.3 指令类型

非轴运动缓存指令。

5.3.1.3.4 示例

以下示例通过 HMC_DefPos 指令实现设置轴当前位置。

(1) 示例 1：

HMC_DefPos(Axis0.AxisID,0); (*将轴 Axis0 当前坐标位置设定为坐标原点*)

(2) 示例 2：

HMC_DefPos(Axis0.AxisID,1000); (*将轴 Axis0 当前坐标位置设定为 1000*)

5.3.2 单向运动

5.3.2.1 HMC_Forward（正向运动）

5.3.2.1.1 功能

持续正向运动，运动速度由轴参数 Speed 决定，加减速由轴参数 Accel、Decel 确定。在运动过程中，可以实时修正 Speed、Accel、Decel 参数，实时生效。

5.3.2.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_Forward	Output	UDINT	返回值

5.3.2.1.3 指令类型

轴运动缓存指令。

5.3.2.1.4 补充说明

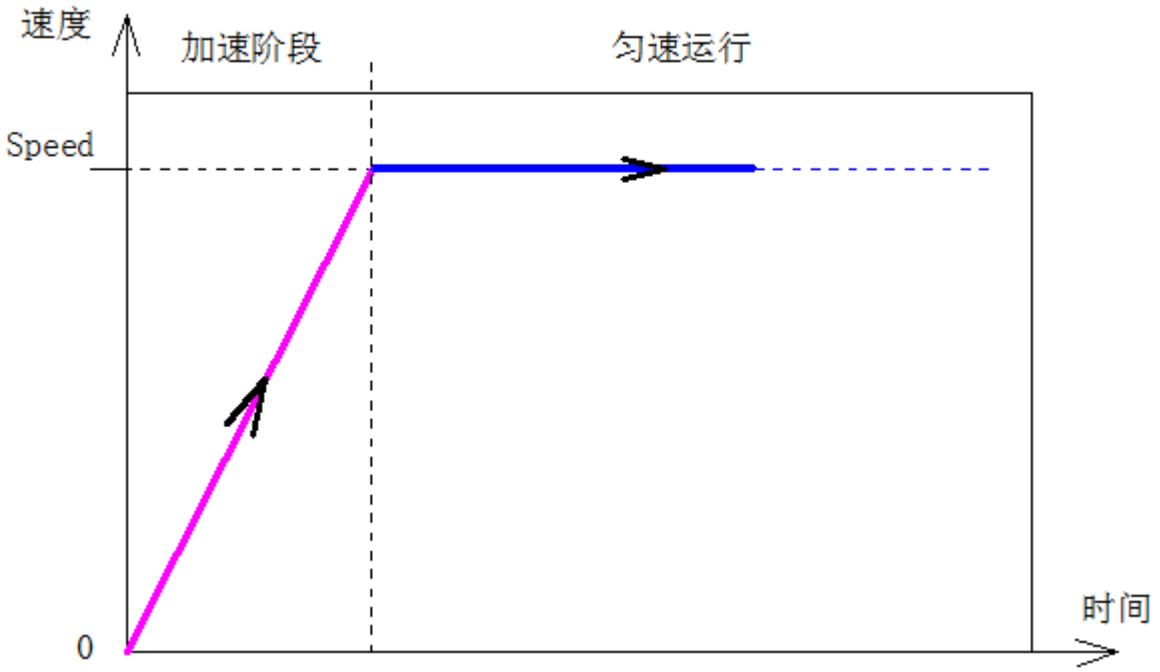
运动过程中如果需要撤消停止，可使用 HMC_Cancel 指令；

运动过程中若遇到正限位，将减速停止，参见限位时的处理。

5.3.2.1.5 示例

以下示例通过 HMC_Forward 指令实现轴的正向运动。运动速度由轴参数 Speed 决定，加减速由轴参数 Accel、Decel 确定。在运动过程中，可以实时修正 Speed、Accel、Decel 参数，实时生效。

（1） 示例 1：



HMC_Forward

示例程序如下（ST 语言），运行过程的速度-时间曲线如上图所示：

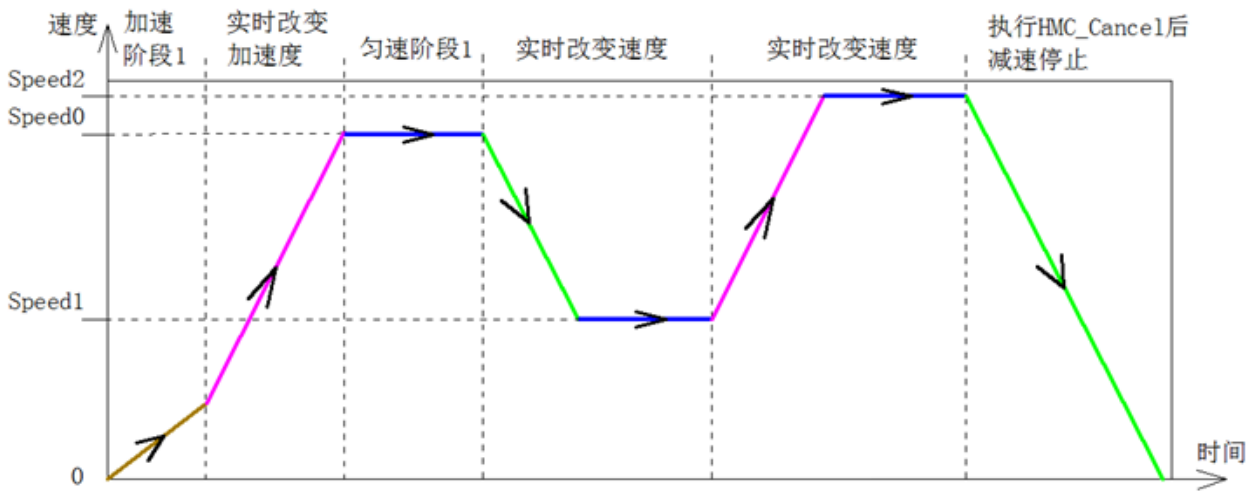
```
Axis0.Speed := 1000;      (*设定运动速度*)
Axis0.Accel := 2000;      (*设定加速度*)
Axis0.Decel := 2000;      (*设定减速度*)
HMC_Forward(Axis0.AxisID); (*向 Axis0 轴投送 HMC_Forward 指令*)
```

(2) 示例 2：实时调整速度、加速度，撤消停止

示例程序如下（ST 语言）：

```
Axis0.Speed := 2000;      (*设定运动速度*)
Axis0.Accel := 1000;      (*设定加速度*)
Axis0.Decel := 2000;      (*设定减速度*)

HMC_Forward(Axis0.AxisID); (*向 Axis0 轴投送 HMC_Forward 指令*)
TimeDelay(500);           (*延时 500 毫秒*)
```

```
Axis0.Accel := 2000;      (*改变加速度，实时生效*)
TimeDelay(1500);          (*延时 1500 毫秒*)

Axis0.Speed := 1000;      (*降低速度，实时生效*)
TimeDelay(1500);          (*延时 1500 毫秒*)

Axis0.Speed := 2500;      (*增大速度，实时生效*)
TimeDelay(1500);          (*延时 1500 毫秒*)
HMC_Cancel(Axis0.AxisID,1); (*撤消 Axis0 轴当前 HMC_Forward 指令*)
```



HMC_Forward 运动中实时调整速度、加速度

5.3.2.2 HMC_Reverse（反向运动）

5.3.2.2.1 功能

持续负向运动，运动的速度由轴参数 Speed 决定。在运动过程中，可以实时修正 Speed、Accel、Decel 等参数，实时有效。

5.3.2.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_Reverse	Output	UDINT	返回值

5.3.2.2.3 指令类型

轴运动缓存指令。

5.3.2.2.4 补充说明

运动过程中如果需要撤消停止，可使用 HMC_Cancel 指令；

运动过程中若遇到负限位，将减速停止，参见限位时的处理。

5.3.2.2.5 示例

参见 HMC_Forward（正向运动）。

5.3.2.3 HMC_ForwardEx（高级正向运动）

5.3.2.3.1 功能

该指令执行时首先修改轴参数 Speed=函数参数 dSpeed，然后以速度 Speed 持续向前运动。运动过程中可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。

5.3.2.3.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dSpeed	Input	LREAL	执行该运动时的目标速度

HMC_ForwardEx	Output	UDINT	返回值

5.3.2.3.3 指令类型

轴运动缓存指令。

5.3.2.3.4 补充说明

运动过程中如果需要撤消停止，可使用 HMC_Cancel 指令；

运动过程中若遇到正限位，将减速停止，参见限位时的处理。

5.3.2.3.5 示例

以下示例通过 HMC_ForwardEx 指令轴的正向运动，实现该指令执行时首先修改轴参数 Speed=函数参数 dSpeed，然后以速度 Speed 持续向前运动。运动过程中可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。

(1) 示例 1：

示例程序如下（ST 语言）：

```
Axis0.Speed := 1000;        (*设定运动速度*)
Axis0.Accel := 2000;        (*设定加速度*)
Axis0.Decel := 2000;        (*设定减速度*)
HMC_ForwardEx(Axis0.AxisID,2000);  (*向 Axis0 轴投送 HMC_Forward 指令,将以速度 2000 运行（而非 1000）*)
```

(2) 示例 2：

实时修改速度、加减速，可参见 HMC_Forward（正向运动）中示例。

5.3.2.4 HMC_ReverseEx（高级反向运动）

5.3.2.4.1 功能

该指令执行时首先修改轴参数 Speed=函数参数 dSpeed，然后以速度 Speed 持续负向运动。运动过程中可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。

5.3.2.4.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dSpeed	Input	LREAL	执行该运动时的目标速度
HMC_ReverseEx	Output	UDINT	返回值

5.3.2.4.3 指令类型

轴运动缓存指令。

5.3.2.4.4 补充说明

运动过程中如果需要撤消停止，可使用 HMC_Cancel 指令；

运动过程中若遇到负限位，将减速停止，参见限位时的处理。

5.3.2.4.5 示例

参见 HMC_ForwardEx（高级正向运动）。

5.3.3 点位运动

5.3.3.1 HMC_Move（相对运动）

5.3.3.1.1 功能说明

让某个轴按照预先设定的速度及加减速参数运动到相对于本指令开始执行时刻所在位置的相对位移处（正负均可）。运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在运动过程中，可以实时修改 Speed、Accel、Decel 等轴参数，立即生效。

HMC_Move 指令是 HMC 运动控制系统中最典型的运动控制函数，可控制单轴独立运动。没有插补或同步关系的轴可通过简单的执行 HMC_Move 进行独立的运动控制。

HMC_Move 的第一个参数是轴号，第二个参数是从当前位置开始该轴要移动的位移，正负代表运动方向。位移单位可通过轴参 Units、Ke、Ks 进行调整，详见相关轴参数。

（1） 在匀加速、匀减速运动情况下，如果设定的目标速度 Speed 能够达到且存在匀速段，则运动过程中的速度与时间曲线分为三段（加速段、匀速段、减速段），呈现为梯形；否则分为两段（加速段、减速段），呈现为三角形。

（2） 指令执行过程中，对于有限行程，对应的轴参有如下关系：

Endpos = Dpos+Remain

对于循环模式，Move 指令完成过程中，Endpos、Remain、Dpos、Mpos 的关系详见 RepMode、RepDist。

（3） 在位置开环模式下（如步进轴），当满足 Dpos=Endpos、Remain=0 时，指令执行完成。在位置闭环模式下（如伺服轴），判断执行是否完成还可能需要看 Mpos 是否到位，详见轴参数 CmdStopMode、StopFETol。

（4） 当 Jerk 模式使能时，指令实际执行的最大加减速会受 Jerk 功能限制，详见 Jerk（加加速度）。

（5） Merge 功能关闭时，HMC_Move 指令执行完成时，速度为 0。当 Merge 功能打开且满足 Merge 生效条件时，多条指令可以被连接在一起连续不间断执行，以提高效率，详见 Merge（融合功能模式）。

5.3.3.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dDistance	Input	LREAL	增量位移（正负均可）
HMC_Move	Output	UDINT	返回值

5.3.3.1.3 指令类型

轴运动缓存指令。

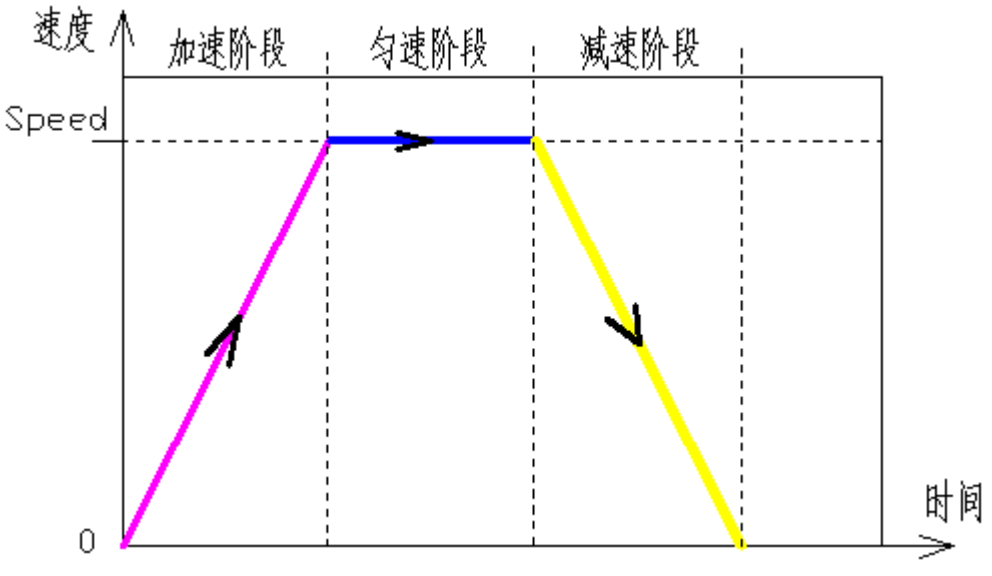
5.3.3.1.4 示例

以下示例通过 HMC_Move 指令实现轴的相对运动。

（1） 示例 1：梯形速度曲线

当设定的运动学参数满足下述关系式时，指令执行过程的速度曲线为梯形。

$$dDistance > \frac{Speed^2}{2}\left(\frac{1}{Accel}+\frac{1}{Decel}\right)$$



梯形加减速曲线

示例程序如下（ST 语言）：

```
Axis0.Speed := 1000;          (*设定运动速度*)
Axis0.Accel := 2000;          (*设定加速度*)
Axis0.Decel := 2000;          (*设定减速度*)
HMC_Move(Axis0.AxisID，4000); (*向 Axis0 轴投送 HMC_Move 指令,运动增量距离 4000*)
```

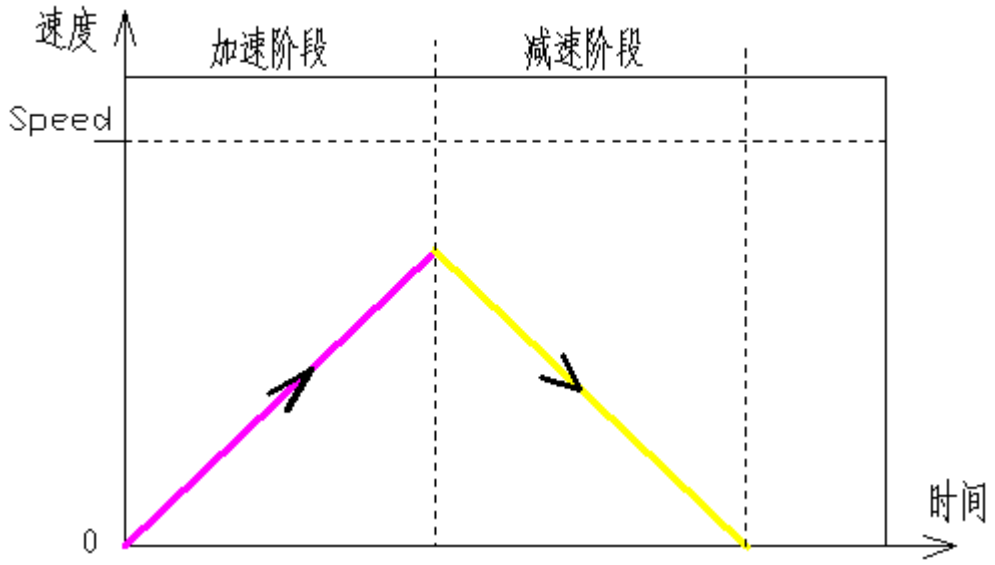
(2) 示例 2：三角形速度曲线

当设定的运动学参数满足下述关系式时，指令执行过程的速度曲线为三角形。

$$\text{dDistance} \leq \frac{Speed^2}{2}\left(\frac{1}{Accel}+\frac{1}{Decel}\right)$$

示例程序如下（ST 语言）：

```
Axis0.Speed := 2000;          (*设定运动速度*)
Axis0.Accel := 1000;          (*设定加速度*)
Axis0.Decel := 1000;          (*设定减速度*)
HMC_Move(Axis0.AxisID，1000); (*向 0 号轴投送 HMC_Move 指令,运动增量距离 1000*)
```

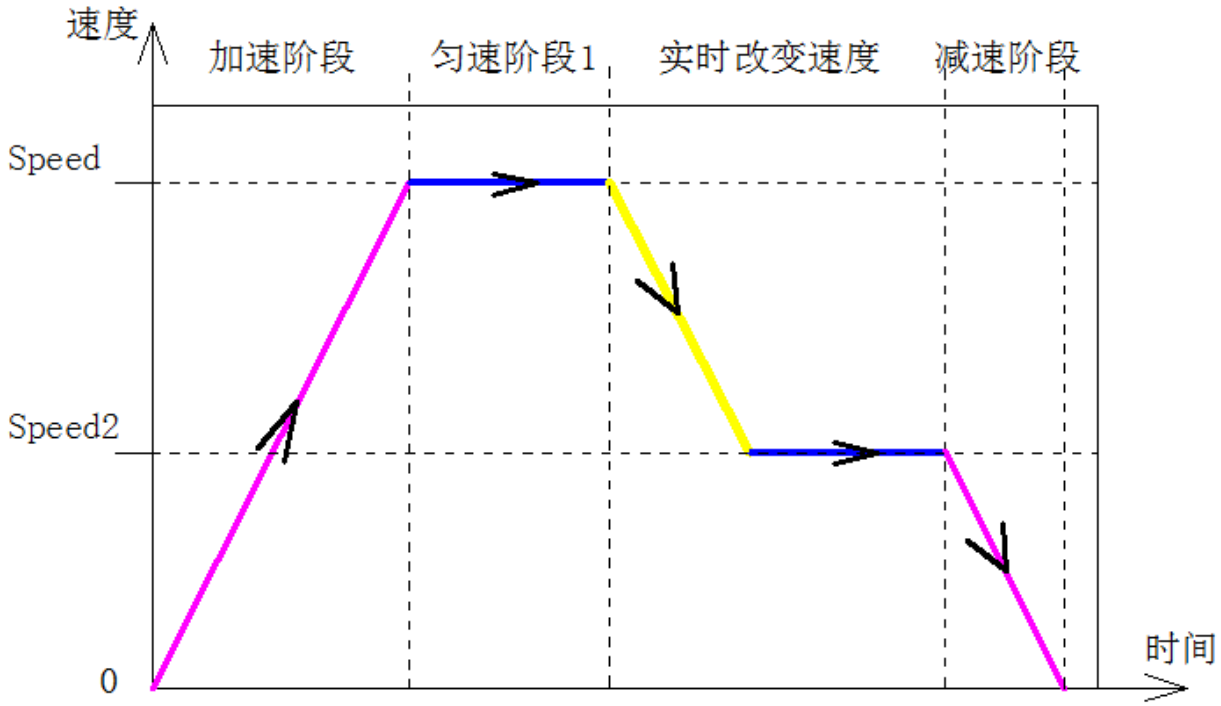


三角形加减速曲线

(3) 示例 3：运动中实时修改参数

在运动过程中，可以实时修改 Speed、Accel、Decel 等轴参数，立即生效。程序如下（ST 语言）：

```
Axis0.Speed := 1000;    (*设定运动速度*)

Axis0.Accel := 1000;    (*设定加速度*)

Axis0.Decel := 1000;    (*设定减速度*)

HMC_Move(Axis0.AxisID，3000);      (*向 0 号轴投送 HMC_Move 指令*)

TimeDelay(2000);        (*延时等待 2000 毫秒*)

Axis0.Speed := 500;     (*速度降为 500*)
```



运动中实时修改速度参数

5.3.3.2 HMC_MoveAbs（绝对运动）

5.3.3.2.1 功能

让某个轴按照预先设定的加速度、速度及加速度参数运动到坐标位置 dDistance 处。运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在运动过程中，可以实时修正 Speed、Accel、Decel 等轴参数，实时生效。

5.3.3.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dDistance	Input	LREAL	增量位移（正负均可）
HMC_MoveAbs	Output	UDINT	返回值

5.3.3.2.3 指令类型

轴运动缓存指令。

5.3.3.2.4 示例

以下示例通过 HMC_MoveAbs 指令实现轴的绝对运动。

（1） 示例（1）：使轴 0 依次运动 1000、3000、-1000、2000 的增量位移，采用 HMC_Move 与 HMC_MoveAbs 两种方式实现的程序如下（ST 语言）：

- HMC_Move 实现程序：

```
Axis0.Speed := 1000;          (*设定运动速度*)
Axis0.Accel := 2000;          (*设定加速度*)
Axis0.Decel := 2000;          (*设定减速度*)

HMC_Move(Axis0.AxisID，1000);  (*向 0 号轴投送 HMC_Move 指令,运动增量距离 1000*)
HMC_Move(Axis0.AxisID，3000);  (*向 0 号轴投送 HMC_Move 指令,运动增量距离 3000*)
HMC_Move(Axis0.AxisID，-1000); (*向 0 号轴投送 HMC_Move 指令,运动增量距离-1000*)
HMC_Move(Axis0.AxisID，2000);  (*向 0 号轴投送 HMC_Move 指令,运动增量距离 2000*)
```

- HMC_MoveAbs 实现程序（假设起点为 0 点）：

```
Axis0.Speed := 1000;        (*设定运动速度*)
```

Axis0.Accel := 2000;　　(*设定加速度*)

Axis0.Decel := 2000;　　(*设定减速度*)

HMC_MoveAbs (Axis0.AxisID，0 + 1000); (*向 0 号轴投送 HMC_MoveAbs 指令,运动至绝对位置 1000 处*)

HMC_ MoveAbs (Axis0.AxisID，0 + 1000 + 3000);　(*向 0 号轴投送 HMC_MoveAbs 指令,运动绝对位置 4000 处*)

HMC_ MoveAbs (Axis0.AxisID，0 + 1000 + 3000 + (-1000));　(*向 0 号轴投送 HMC_MoveAbs 指令,运动绝对位置 3000 处*)

HMC_ MoveAbs (Axis0.AxisID，0 + 1000 + 3000 + (-1000) + 2000);　(*向 0 号轴投送 HMC_MoveAbs 指令,运动绝对位置 5000 处*)

(2) 示例 2：使轴 0 依次运行至绝对位置坐标 1000、3000、-1000、2000，分别采用 HMC_Move 与 HMC_MoveAbs 两种方式实现的程序如下（ST 语言）：

□ HMC_Move 实现程序(假设起点为 0 点)：

Axis0.Speed := 1000;　(*设定运动速度*)

Axis0.Accel := 2000;　(*设定加速度*)

Axis0.Decel := 2000;　(*设定减速度*)

HMC_Move(Axis0.AxisID，1000 - 0);　　(*向 0 号轴投送 HMC_Move 指令,运动增量距离 1000*)

HMC_Move(Axis0.AxisID，3000 - 1000);　　(*向 0 号轴投送 HMC_Move 指令,运动增量距离 2000*)

HMC_Move(Axis0.AxisID，- 1000 - 3000);　　(*向 0 号轴投送 HMC_Move 指令,运动增量距离-1000*)

HMC_Move(Axis0.AxisID，2000 – (-1000));　　(*向 0 号轴投送 HMC_Move 指令,运动增量距离 2000*)

□ HMC_MoveAbs 实现程序：

```
Axis0.Speed := 1000;    (*设定运动速度*)
Axis0.Accel := 2000;    (*设定加速度*)
Axis0.Decel := 2000;    (*设定减速度*)

HMC_MoveAbs (Axis0.AxisID，1000);    (*向 0 号轴投送 HMC_MoveAbs 指令,运动至绝对位置 1000 处*)
HMC_ MoveAbs (Axis0.AxisID，3000);    (*向 0 号轴投送 HMC_MoveAbs 指令,运动绝对位置 3000 处*)
HMC_ MoveAbs (Axis0.AxisID，-1000);    (*向 0 号轴投送 HMC_MoveAbs 指令,运动绝对位置-1000 处*)
HMC_ MoveAbs (Axis0.AxisID，2000);    (*向 0 号轴投送 HMC_MoveAbs 指令,运动绝对位置 2000 处*)
```

通过上述例子可见，在增量坐标下使用 HMC_Move 较为适合，在绝对坐标下使用 HMC_MoveAbs 则较为方便。

5.3.3.3 HMC_MoveEx（高级相对运动）

5.3.3.3.1 功能

该指令执行时首先修改轴参数 Speed=函数参数 dSpeed，然后以速度 Speed 运动到相对于本指令开始执行时刻所在位置的相对位移处（正负均可）。在运动过程中，可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。

5.3.3.3.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号

dDistance	Input	LREAL	增量位移（正负均可）
dSpeed	Input	LREAL	执行该运动时的目标速度
HMC_MoveEx	Output	UDINT	返回值

5.3.3.3.3 指令类型

轴运动缓存指令。

5.3.3.3.4 示例

以下示例通过 HMC_MoveEx 指令实现轴的高级相对运动，该指令执行时首先修改轴参数 Speed=函数参数 dSpeed，然后以速度 Speed 运动到相对于本指令开始执行时刻所在位置的相对位移处（正负均可）。在运动过程中，可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。详细内容可参考 HMC_Move（相对运动）中功能说明部分。

```
Axis0.Speed := 1000;    (*设定运动速度*)
Axis0.Accel := 2000;    (*设定加速度*)
Axis0.Decel := 2000;    (*设定减速度*)
HMC_MoveEx(Axis0.AxisID，10000,2000);    (*向 0 号轴投送 HMC_Move 指令,将以最大速度 2000 运行（而非 1000）*)
```

5.3.3.4 HMC_MoveAbsEx（高级绝对运动）

5.3.3.4.1 功能

轴参数 Speed=函数参数 dSpeed，让某个轴按照本指令设定的目标速度运动到绝对位置。在运动过程中，可以实时修正 Speed、Accel、Decel 等轴参数，改变运动过程，实时有效。

5.3.3.4.2 输入参数

参数	声明	数据类型	说明

ucAxisID	Input	USINT	轴号
dDistance	Input	LREAL	增量位移
dSpeed	Input	LREAL	执行该运动时的目标速度
HMC_MoveAbsEx	Output	UDINT	返回值

5.3.3.4.3 指令类型

轴运动缓存指令。

5.3.3.4.4 示例

参考 HMC_MoveAbs（绝对运动）。

5.3.4 指令修正

5.3.4.1 HMC_Cancel（撤销指令）

5.3.4.1.1 功能

撤销指定轴运动缓存中的指令，支持撤销当前指令、撤销下一条指令、撤销该轴所有指令三种模式。

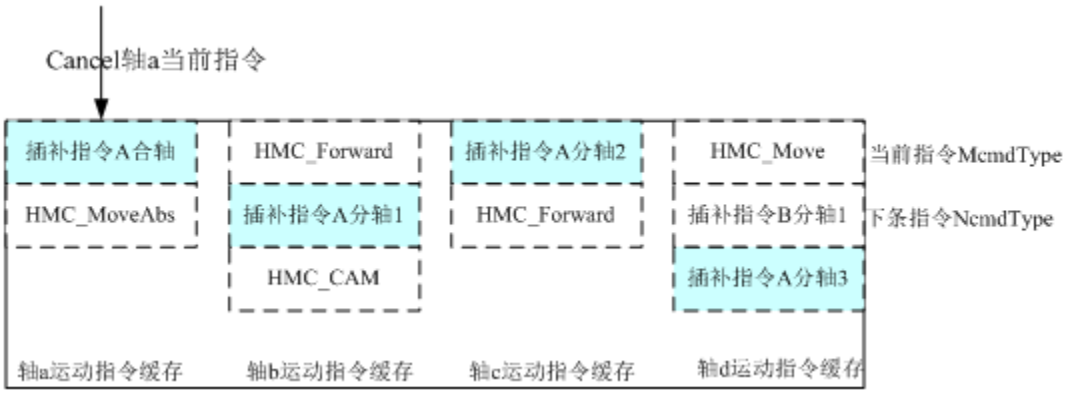
（1） 不同指令类型 Cancel 响应方式

（a） 单轴指令

撤销该轴当前或所有指令。

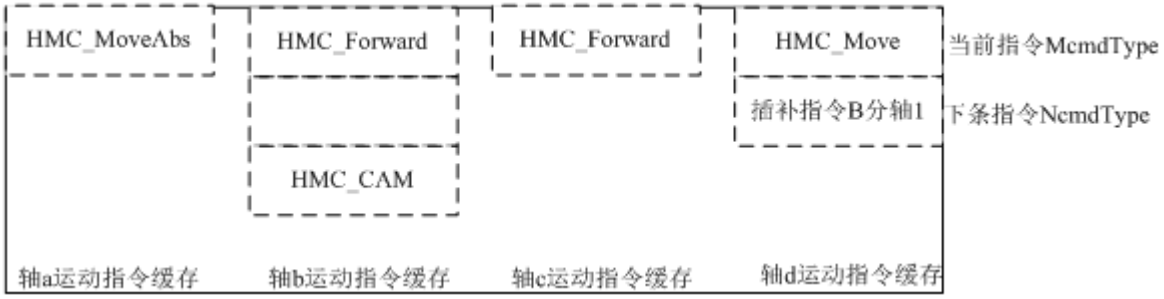
（b） 插补合轴

撤销插补合轴指令时，会撤销与该插补指令相关的所有轴的该插补指令，即包括合轴、分轴，且无论该指令在各插补分轴是否为当前指令。如当前各轴的运动指令缓存状态如下，现欲对轴 a 撤销当前指令。



运动指令缓存状态示意图

则调用 Cancel 指令后，轴 a 立即响应，无论其从轴的该指令是否处于当前指令。待 Cancel 执行完成后，各轴的运动指令缓存状态如下：



运动指令缓存状态示意图

- 注意：Cancel 插补指令后，会直接删除合轴、分轴的该指令缓存为空，但此时不会调整后续指令上移，而是等待执行到此指令时，判断为空则直接执行后续指令，对实际执行效率无影响。

(c) 插补分轴

Cancel 指令对插补分轴无效。

(d) 联动主轴

对于联动指令主轴执行 Cancel，仅对主轴生效，对从轴无影响。

(e) 联动从轴

对于联动指令从轴执行 Cancel，仅对从轴生效，对主轴无影响。

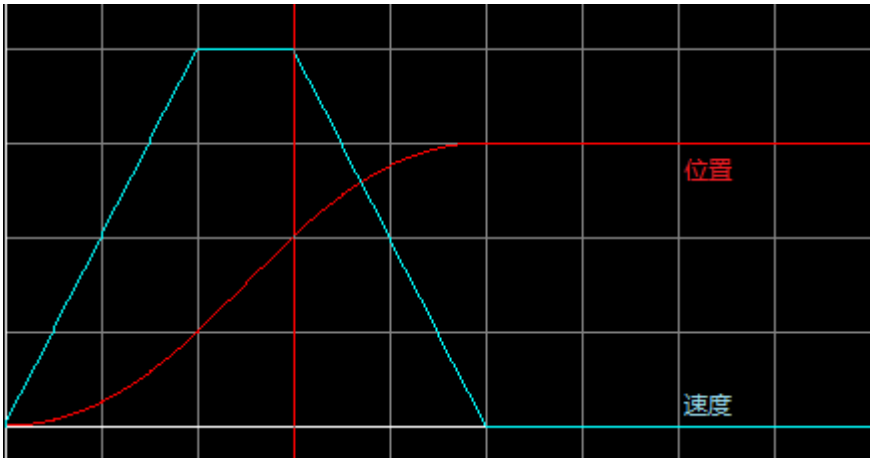
(2) 指令撤销时轴的停止方式

指令撤销过程中，在接受到 Cancel 指令后，轴会以当前速度开始进行减速，直到停止。但该减速

停止过程中具体停止方式，如实际减速度、最终停止时刻位置等可由轴参数 CancelMode、CancelPos 指定。具体如下：

（a） CancelMode=0 自然停止

默认值，按照 Decel 自然停止。此模式支持所有指令类型。

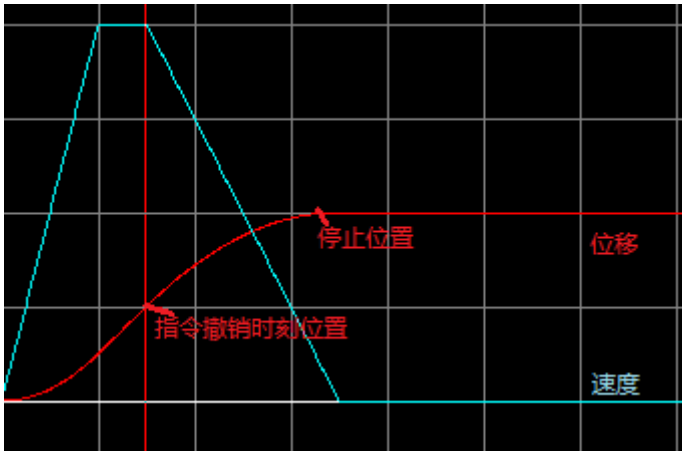


运动轨迹示意图

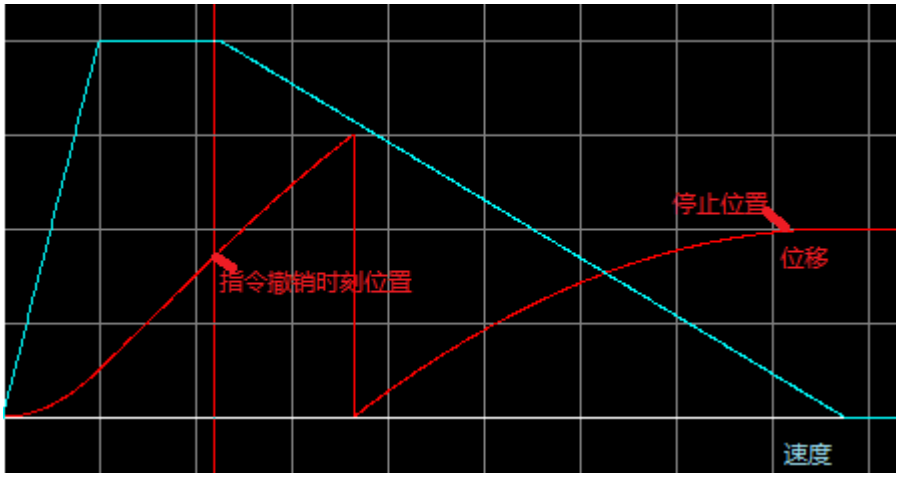
（b） CancelMode=1 指定位置停止

按照不超过减速度 Decel 停到指定位置，该位置由参数 CancelPos 指定。此模式仅对循环行程模式(RepMode=1 或 2)且当前指令为单向运动（HMC_Forward、HMC_Reverse）的情况生效。若单向运动指令，但不满足循环行程模式，则按照 CancelMode=0 处理。

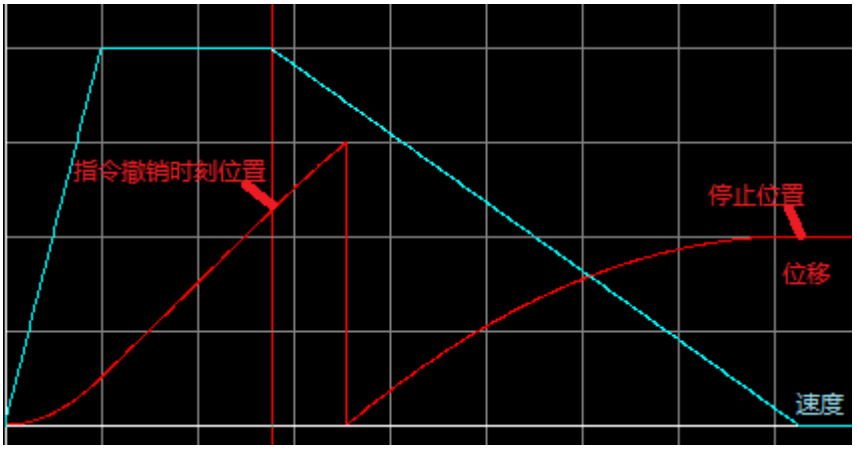
根据当前位置和 CancelPos 的相对关系，会有三种情况：1、当前位置<停止位置，且两者距离位移差足够当前速度以减速度停止为 0；2、当前位置<停止位置，但两者距离位移差不够当前速度以减速度停止为 0；3、当前位置>停止位置。以上三种情况分别对应如下三图：



运动轨迹示意图



运动轨迹示意图

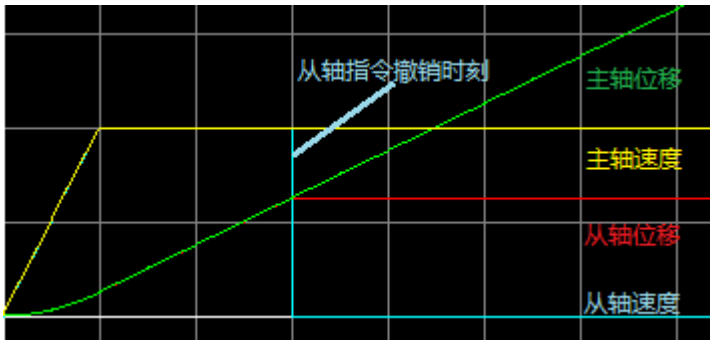


运动轨迹示意图

(c) CancelMode=2 立即停止

立即停止，速度从当前速度直接减速为 0。此模式仅对正在执行联动指令的轴生效，若状态不

满足，则默认按照 CancelMode=0 处理。



运动轨迹示意图

5.3.4.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucCancelMode	Input	USINT	0：撤销下一条指令 1：撤销当前指令 2：撤销该轴所有指令
HMC_Cancel	Output	UDINT	返回值

5.3.4.1.3 指令类型

非轴运动缓存指令。

5.3.4.1.4 示例

轴 0 单向正向运动 10s 之后，以 Decel 减速停止。

```
Axis0.Speed := 100000;
Axis0.Accel :=100000;
Axis0.Decel :=100000;
Axis0.CancelMode:=0;
HMC_Forward(0);
TimeDelay (10000);
HMC_Cancel(0, 1);
```

5.3.4.2 HMC_MoveModify（目标位置变更）

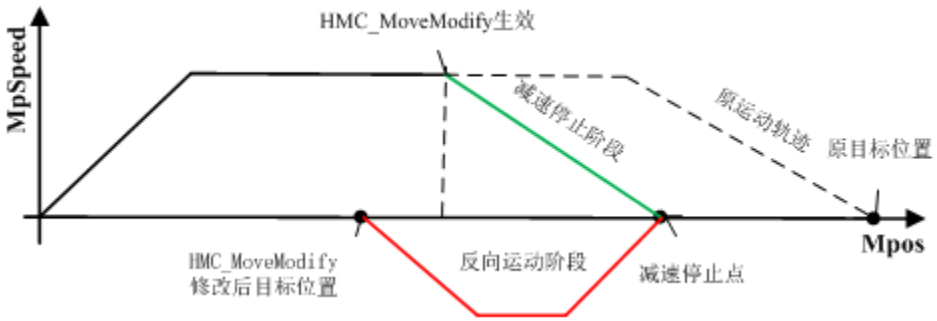
5.3.4.2.1 功能

实时修改 HMC_Move/HMC_MoveAbs 指令的目标位置，修改后的目标位置为 dDpos。

执行 HMC_MoveModify 时，若当前轴指令队列中没有指令，则直接投送 MoveAbs 指令；若有指令但且当前正在执行的指令不是 Move 或 MoveAbs，则忽略本次修改请求。

在执行 HMC_MoveModify 时，存在以下几种处理方式：

（1） 若轴的运动方向是远离 HMC_MoveModify 所指定的终点位置的方向，则该轴将减速直至停止，然后反向运动到 HMC_MoveModify 所设定的位置。

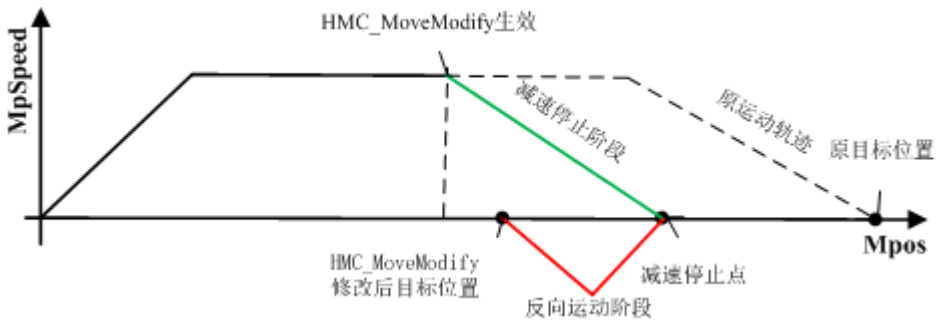


运动示意图

示例程序：

```
p1 := HMC_Move(0, 10000);
p2:= TimeDelay(15); (* 此时的位置值 axis0.dpos 已经超过 5000 并继续远离 5000 *)
p3:= HMC_MoveModify(0, 5000)
```

（2） 若轴的运动方向是朝向 HMC_MoveModify 所指定的终点位置的方向，但没有足够的减速距离，则当前轴将减速停止，然后反向运动到 HMC_MoveModify 所设定的位置。



运动示意图

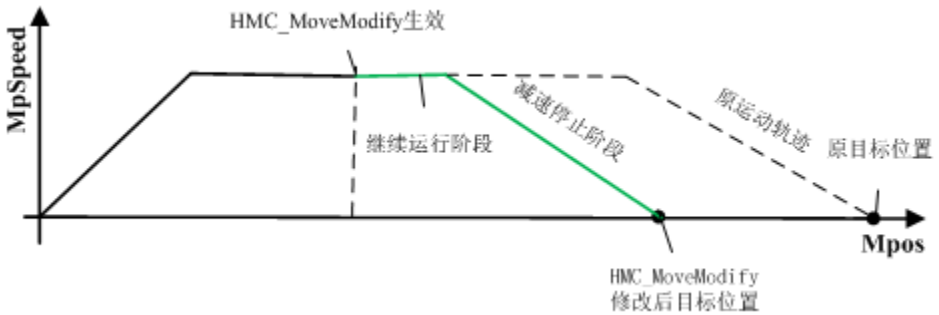
示例程序：

```
p1 := HMC_Move(0, 10000);
p2 := TimeDelay(10); (* 此时的位置值 axis0.dpos 已接近 6000，没有足够的减速距离使其减速停止 *)
p3 := HMC_MoveModify(0, 6100)
```

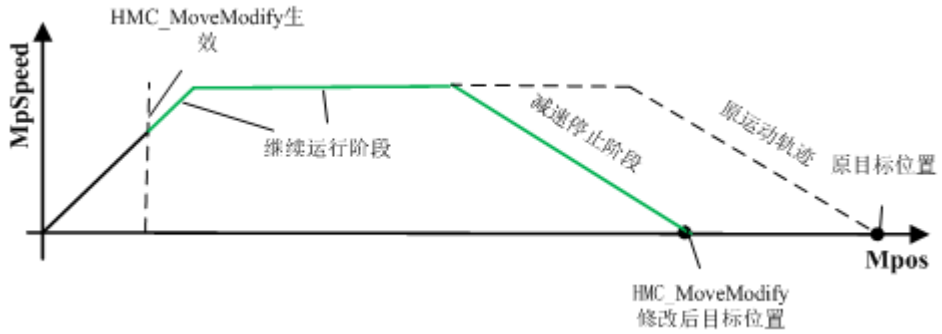
（3） 若轴的运动方向是朝向HMC_MoveModify所指定的终点位置的方向，且有足够的减速距离，则当前轴将继续运动至减速点，然后减速停止到 HMC_MoveModify 所设定的位置。

由于 HMC_MoveModifyEx 执行时机的不同以及修改后的终点位置的不同，“继续运行阶段”的运动过程不尽相同，并且匀速阶段不一定存在。以下列出几种典型的运动过程示意图：

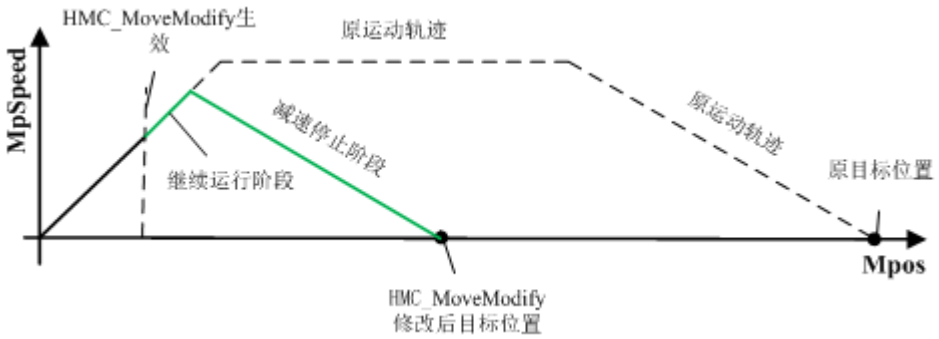
（a） HMC_MoveModify 所指定的终点位置在原目标位置之前。



设定位置在原目标位置之前

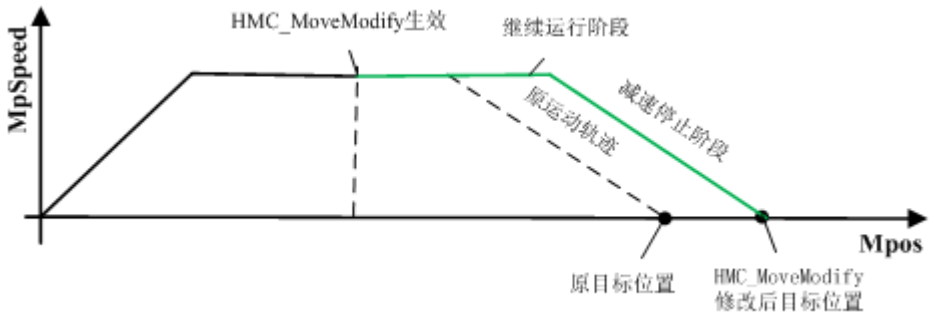


设定位置在原目标位置之前（加速阶段执行 HMC_MoveModify，存在匀速段）

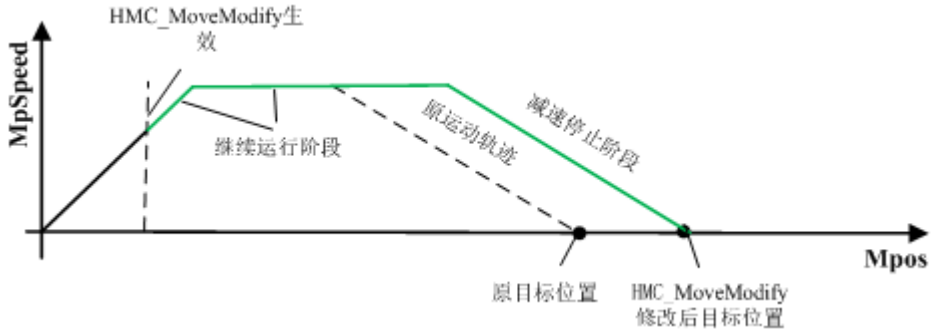


设定位置在原目标位置之前（加速阶段执行 HMC_MoveModify，无匀速段）

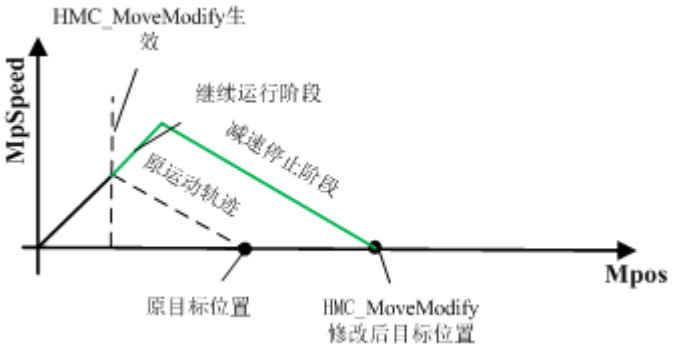
（b） HMC_MoveModify 所指定的终点位置在原目标位置之后。



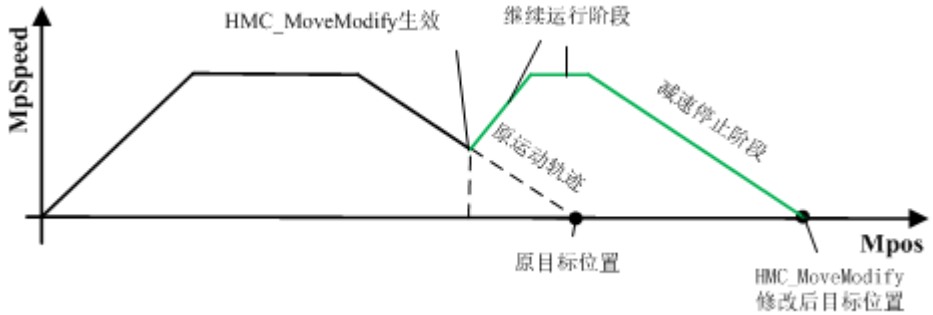
设定位置在原目标位置之后（匀速阶段执行 HMC_MoveModify）



设定位置在原目标位置之后（加速阶段执行 HMC_MoveModify，存在匀速段）



设定位置在原目标位置之前（加速阶段执行 HMC_MoveModify，无匀速段）



设定位置在原目标位置之后（减速阶段执行 HMC_MoveModify）

示例程序：

```
p1 := HMC_Move(0, 10000);
p2 := TimeDelay(2); (* 此时的位置值 axis0.dpos 远小于 5000，有足够的剩余减速距离 *)
p3 := HMC_MoveModify(0, 5000)
```

5.3.4.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dDpos	Input	LREAL	修改当前指令的位移
HMC_MoveModify	Output	UDINT	返回值

5.3.4.2.3 指令类型

非轴运动缓存指令。

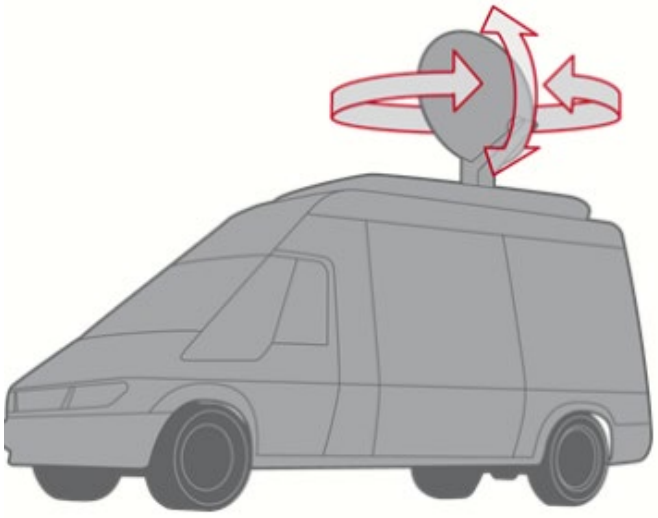
5.3.4.2.4 补充说明

若当前 Merge 打开，则在投送 HMC_MoveModify 指令时将会自动关闭 Merge。

5.3.4.2.5 示例

卫星信号接收车。

在进行野外测控火箭发射时的运行状态和接收卫星信号时，车载卫星接收器要根据上位机电脑系统发过来的数据，对接收天线的水平方向和上下俯仰方向进行实时的调整，以准确跟踪测控目标的状态。那么我们控制器中的 HMC_MoveModify 函数能很好的满足这个需求。可在此指令执行过程中不断加载修正目标位置，完成随动追踪任务。



卫星信号接收车示意图

设上位机电脑系统发送过来的水平矫正位置为 g_rHorizontalAdjPos，垂直俯仰矫正位置为 g_rVerticalAdjPos，假设水平方向转 360°指令脉冲为 X，垂直方向转 360°指令脉冲数为 Y。具体程序如下：

```
(**** 参数设定 *****)
Axis0UnitReval := HMC_SetUnits(0, X/3600);   (* 设定 AXIS0 单位为 0.1 度 *)
Axis1UnitReval := HMC_SetUnits(1, Y/3600);   (* 设定 AXIS1 单位为 0.1 度 *)
AxisEnableReval := HMC_DriveEnable(1);

axis0.Speed := 100;
axis0.Accel := 1000;
axis0.Decel := 1000;
axis1.Speed := 100;
axis1.Accel := 1000;
axis1.Decel := 1000;

(****后续程序计算，循环扫描*****)
WHILE true DO
IF bStart=1 THEN  (*设备运行*)
(*矫正水平方向*)
Axis0ModifyReval := HMC_MoveModify(0, g_rHorizontalAdjPos);

(*矫正垂直方向*)
Axis1ModifyReval := HMC_MoveModify(1, g_rVerticalAdjPos);

END_IF

IF bStop=1 THEN  (*设备停止*)
Axis0StopReVal := HMC_Cancel(0, 2);
```

```
Axis1StopReVal := HMC_Cancel(1, 2);
RESULT := HMC_MoveAbs(0, 0);
RESULT := HMC_MoveAbs(1, 0);
RESULT := HMC_WaitAxisIdle(0);
RESULT := HMC_WaitAxisIdle(1)
END_IF
END_WHILE
```

5.3.4.3 HMC_MoveModifyEx（高级目标位置变更）

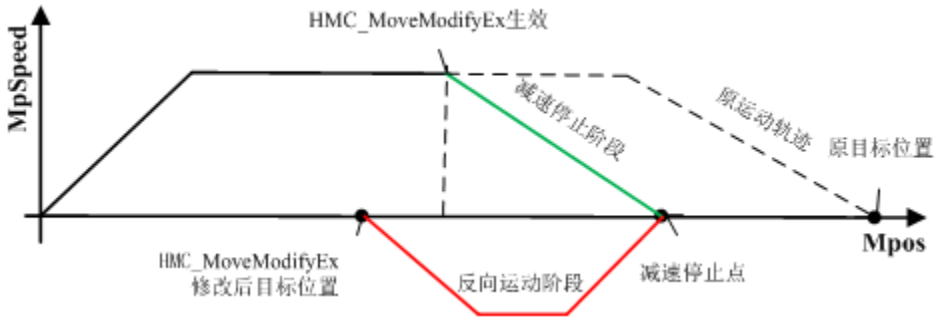
5.3.4.3.1 功能

实时修改 HMC_Move/HMC_MoveAbs 指令的目标位置，修改方式由 ucModifyMode 决定：当 ucModifyMode=0 时，本指令与 HMC_MoveModify 相同，为绝对位置修改模式，修改后的目标位置为 dDpos；当 ucModifyMode=1 时，为增量位置修改模式，修改后的目标位置为当前 dPos + dDpos。

执行 HMC_MoveModifyEx 时，若当前轴指令队列中没有指令，则 ucModifyMode=0 时投送 HMC_MoveAbs 指令， ucModifyMode=1 时投送 HMC_Move 指令；若当前轴指令队列中有指令，且正在执行的指令是 Move 或 MoveAbs，则修改当前指令的位移；若当前轴有指令，且正在执行的指令不是 Move 或 MoveAbs，则忽略本次修改请求。

以下示例通过 HMC_MoveModifyEx 指令实现高级位置目标修改。执行 HMC_MoveModifyEx 时，存在以下几种处理方式：

（1） 若轴的运动方向是远离 HMC_MoveModifyEx 所指定的终点位置的方向，则该轴将减速直至停止，然后反向运动到 HMC_MoveModifyEx 所设定的位置。匀速阶段执行 HMC_MoveModify 的运动过程示意图如下。

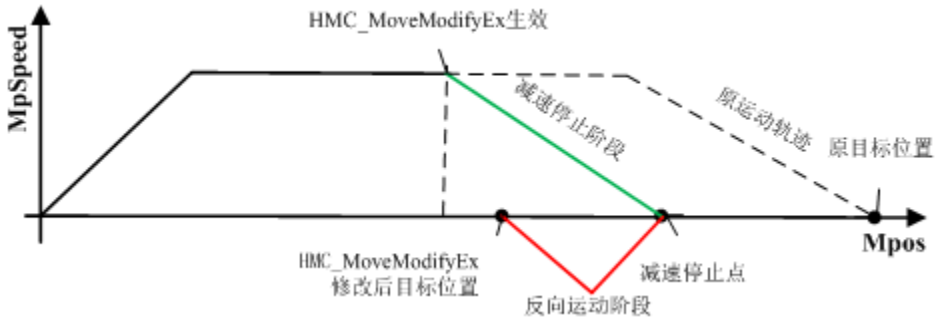


运动示意图

示例程序：

```
p1 := HMC_Move(0, -10000);
p2 := TimeDelay(10);    (* 此时的位置值已经超过 5000 并继续远离 5000 *)
p3 := HMC_MoveModifyEx(0, 5000, 1);
```

（2） 若轴的运动方向是朝向 HMC_MoveModifyEx 所指定的终点位置的方向， 但没有足够的减速距离，则当前轴将减速停止，然后反向运动到 HMC_MoveModifyEx 所设定的位置。匀速阶段执行 HMC_MoveModify 的运动过程示意图如下。



运动示意图

示例程序：

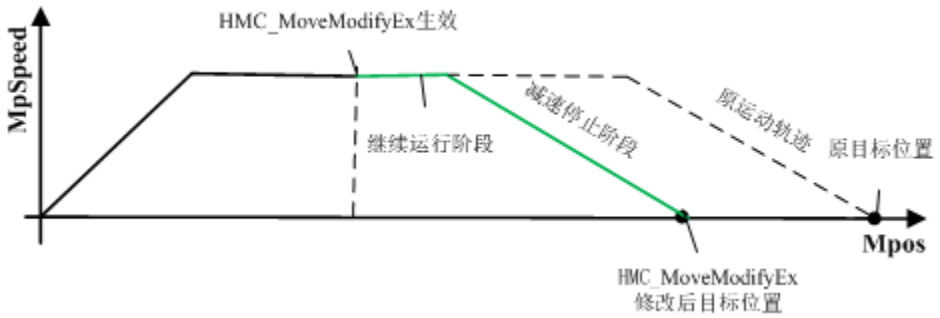
```
p1 := HMC_Move(0, 10000);
p2 := TimeDelay(10);
p3 := HMC_ MoveModifyEx(0,50,1);  (* 增量位置值太小，不足以减速 *)
```

（3） 若轴的运动方向是朝向 HMC_MoveModifyEx 所指定的终点位置的方向， 且有足够的减速距离，则当前轴将继续运动至减速点，然后减速停止到 HMC_MoveModifyEx 所设定的位置。

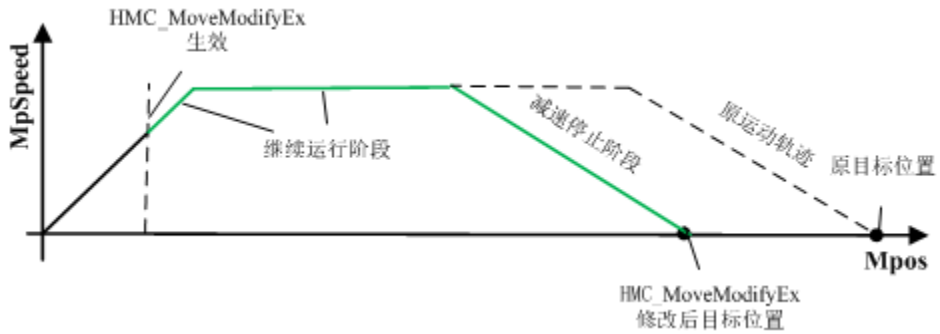
由于 HMC_MoveModifyEx 执行时机的不同以及修改后的终点位置的不同，“继续运行阶段”的运动

过程不尽相同，并且匀速阶段不一定存在。以下列出几种典型的运动过程示意图：

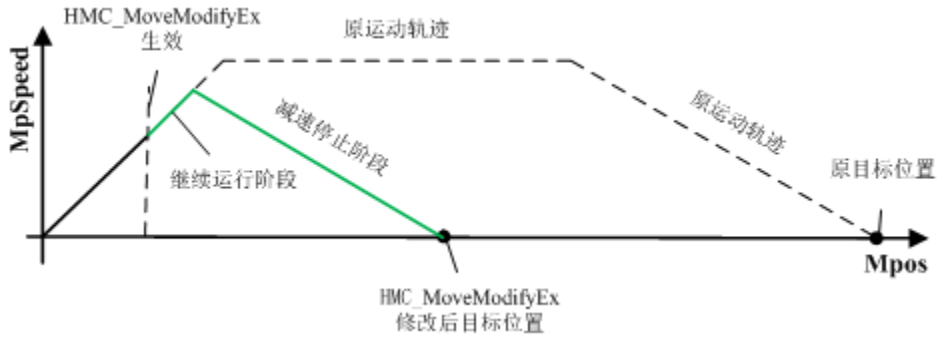
（a） HMC_MoveModifyEx 所指定的终点位置在原目标位置之前。



设定位置在原目标位置之前（匀速阶段执行 HMC_MoveModifyEx）

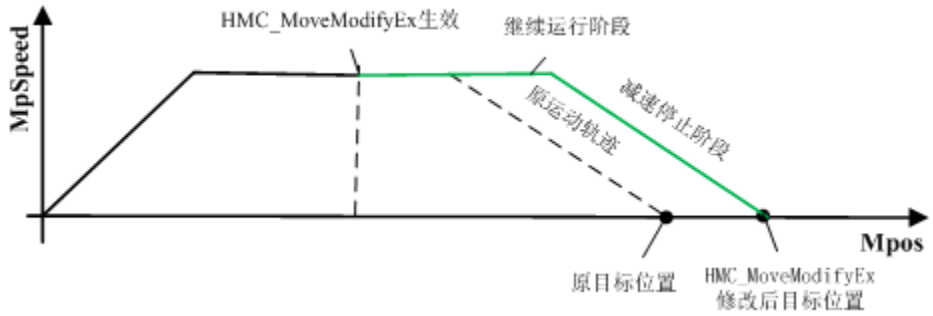


设定位置在原目标位置之前（加速阶段执行 HMC_MoveModifyEx，存在匀速段）

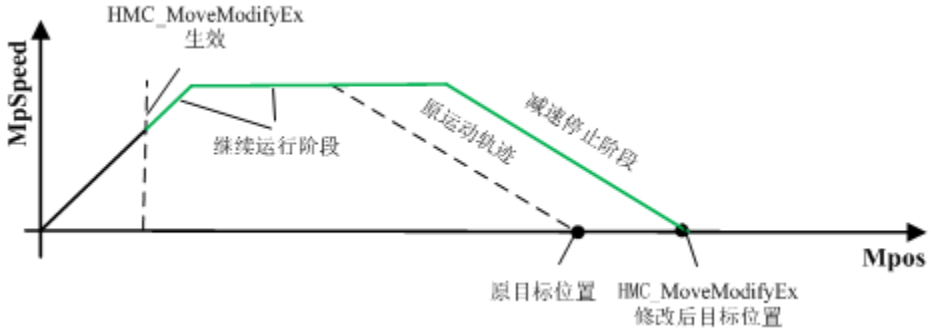


设定位置在原目标位置之前（加速阶段执行 HMC_MoveModifyEx，无匀速段）

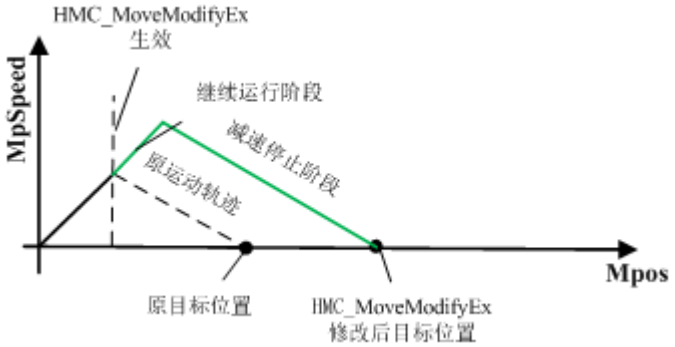
（b） HMC_MoveModifyEx 所指定的终点位置在原目标位置之后。



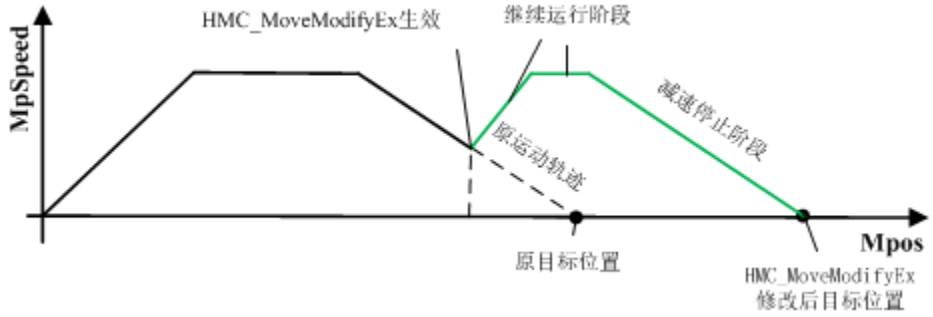
设定位置在原目标位置之后（匀速阶段执行 HMC_MoveModifyEx）



设定位置在原目标位置之后（加速阶段执行 HMC_MoveModifyEx，存在匀速段）



设定位置在原目标位置之前（加速阶段执行 HMC_MoveModifyEx，无匀速段）



设定位置在原目标位置之后（减速阶段执行 HMC_MoveModifyEx）

示例程序：

```
p1 := HMC_Move(0, 10000);
p2 := TimeDelay(2);
p3 := HMC_MoveModifyEx(0, 5000, 1);      (* 此时增量位置为 5000，有足够时间减速 *)
```

5.3.4.3.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dDpos	Input	LREAL	修改当前指令的位移
ucModifyMode	Input	USINT	0：绝对位置修改模式 1：增量位置修改模式
HMC_MoveModify	Output	UDINT	返回值

5.3.4.3.3 指令类型

非轴运动缓存指令。

5.3.4.3.4 补充说明

若当前 Merge 打开，则在投送 HMC_MoveModifyEx 指令时自动关闭 Merge。

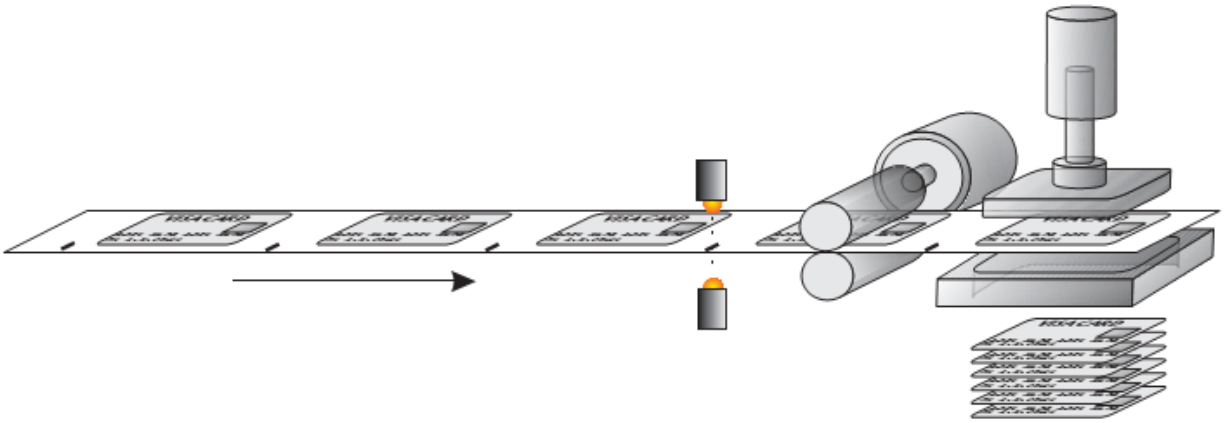
5.3.4.3.5 应用举例

包装机械在运行过程中，由于输送带机械振动将导致材料偏移、传送速度发生变化、包装材料拉伸，加之包装材料上的色标定位之间存在累计误差等因素的影响，将极大影响封切质量。

通过使用高速捕捉功能，配以目标位置变更的修正功能，在每一个色标定位周期进行补偿，可有效抑制累计误差，将误差控制到允许范围内。举例如下。

夹送辊输送一印刷的卡片，卡片在印刷的时候印上了一个色标块（下图中两图案中黑色的方块）。要求把卡片模切下来，模切的位置刚好在卡片中心，不能有累计误差。实现原理：我们在夹送辊的左方加装一个色标传感器，用以检测色标块的到来，当色标块到来时使用高速捕获功能捕获色标块的精确位置，每次行

走的距离以色标块的位置为参考偏移行走合适的距离。由于是同一个印刷版印出来的，所以色标和卡的图案位置是固定的。这样每次走过的距离都是以色标的位置作为参考，消除了累计的误差。色标传感器接到快速 DI 通道 0 上，使用高速捕获通道 0 捕获色标位置。



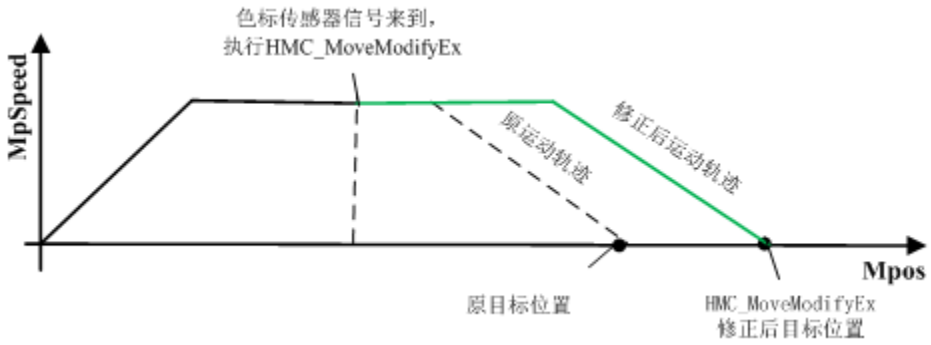
传输示意图

我们假设两卡片图案之间的间距是 300 mm，要模切的位置距离色标的位置是 offset（位置可以调整）。具体程序如下：

```
Start_MPOS := 0.0;
End_MPOS := 300.00;
Offset := 30.0;

Axis0TypeReval := HMC_SetAxisType(Axis0.AxisID, 10); (*设置轴类型*)
Axis0UnitReval := HMC_SetUnits(Axis0.AxisID, 1);    (*设置用户单位*)

AxisEnableReval := HMC_DriveEnable(1);              (*开启伺服使能*)

axis0.Speed := 100;
axis0.Accel := 1000;
axis0.Decel := 1000;

WHILE true DO          (* 后续程序计算，循环扫描 *)
```

```
IF bStart = 1 THEN      (*运行按钮按下*)
Axis0DefReval := HMC_DefOrigin(Axis0.AxisID, Axis0.MPos);     (* 当前位置清零 *)
Config_result := HMC_CaptureConfig(0, 0, 1, 0, 0, 1, Start_MPOS,
 End_MPOS);      (* 配置高速捕获*)

Axis0moveReval := HMC_Move(0, 300);      (* 走一个卡片图案间距 *)

(* 读取高速捕捉通道 0 的状态，等待高速捕获完成。当色标传感器有信号时，高速捕获将会完成，此时可获得色标传感器的精确位置 *)
WHILE (HMC_CaptureGetState(0) <> 3) DO
Reg_pos := HMC_CaptureGetMpos(0);
END_WHILE

Axis0ModifyReval := HMC_MoveModifyEx(0,offset,1);
(* 通过高速捕获到的精确位置，以色标传感器位置为参考修正位移，保证能准确定位到卡片中心位置。例如捕获到位置为 280， Offset 为 30，则补偿后的绝对目标位置为 310，补偿了 10 的误差 *)
Idlereval := HMC_WaitAxisIdle(0);     (* 等待 MoveModify 补偿完毕，再进行下一循环 *)

(****模切卡片（略）****)
END_IF
END_WHILE
```

传送带轨迹：



传送带轨迹示意图

5.4 多轴运动指令

5.4.1 线性插补

5.4.1.1 两轴直线插补

5.4.1.1.1 HMC_Move2D(2 轴直线插补)

5.4.1.1.1.1 功能

二维平面上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的两分轴相对位置（以指令起点时刻位置为坐标原点的坐标位置）为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是根据上一次运动的终点位置做增量运动，即每一步都是增量模式运动。

插补合轴的目标位置在指令开始执行时刻位置基础上，增加以两分轴增量位置为直角边的直角三角形的斜边长度。插补运动合轴运动速度由对应轴参数 Speed 设定，加减速分别由合轴轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。

插补分轴的目标位置在指令开始执行时刻位置基础上，增量该轴指定的相对运动距离。插补运动分轴的运行速度、加减速度与该分轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

插补分轴遇到限位时，选取合轴分解计算的分轴减速度和自身快减速度两者的最大值进行最优保护停止，详见 HMC_HwLimitConfig（设置硬限位开关）。

支持 Merge 功能。多条指令均为支持 Merge 功能的 2 轴插补指令时，如 HMC_Move2D、

HMC_MoveCircl，且合轴、X 轴、Y 轴都一致时，满足 Merge 生效条件，开启 Merge 功能可以提高指令执行效率。

5.4.1.1.1.2 参数

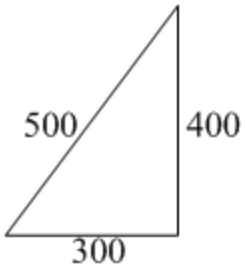
参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
dDistanceX	Input	LREAL	轴 X 运动距离
dDistanceY	Input	LERAL	轴 Y 运动距离
HMC_Move2D	Output	UDINT	返回值

5.4.1.1.1.3 指令类型

轴运动缓存指令。

5.4.1.1.1.4 示例

以下示例通过 HMC_Move2D 指令实现二维平面上运动轮廓为直线的插补功能。以一个平面切割轮廓为三角形的物料为例，三角形如下：



三角形轮廓

编写程序如下：

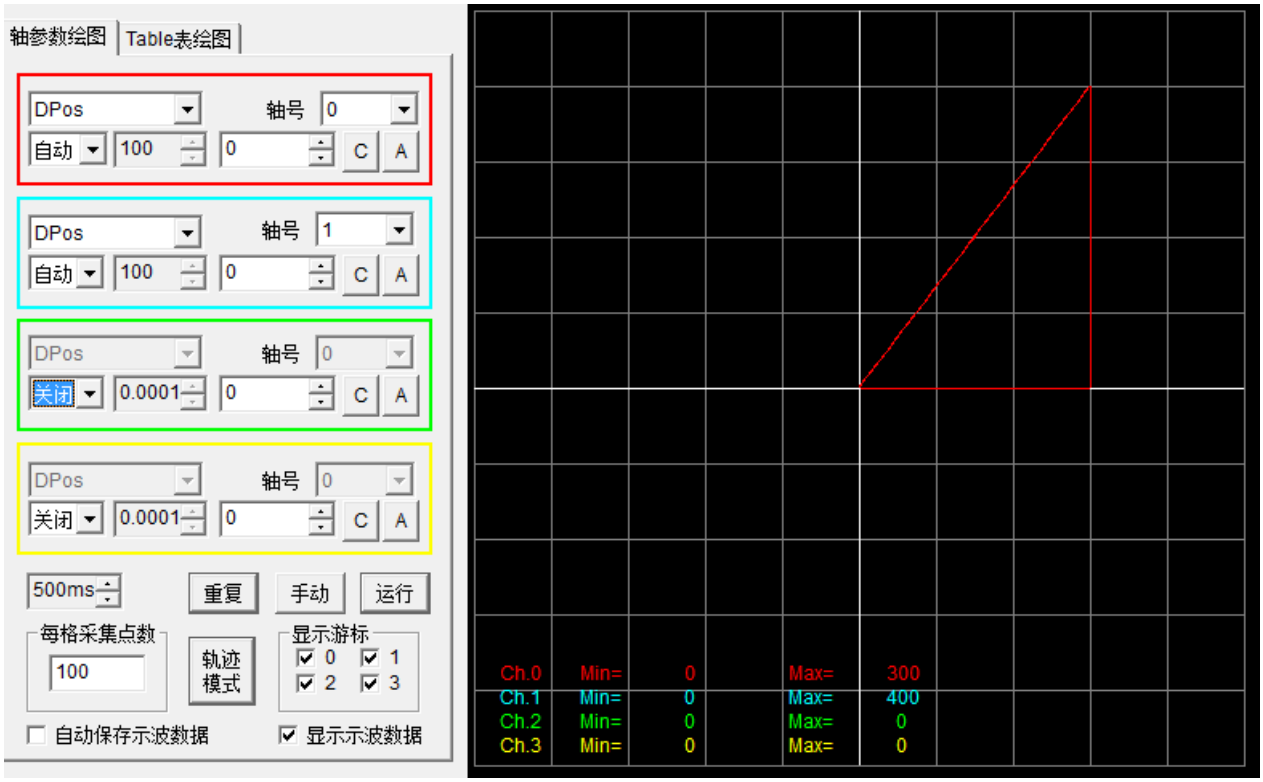
```
(*分别设置使用的三个轴的轴类型*)
HMC_SetAxisType (10 ,0);
HMC_SetAxisType (0 ,10);
HMC_SetAxisType (1,10);
```

HMC_Move2D(10,0,1,300,0); (*X 轴正向移动 300，Y 轴静止。第 1 条直角边*)

HMC_Move2D(10,0,1,0,400); (*X 轴保持静止，Y 轴正向移动 400。第 2 条直角边*)

HMC_Move2D(10,0,1,-300,-400); (*X 轴负向移动 300，Y 轴负向移动 400。斜边*)

通过 AT 示波器观察实际运行轨迹如下：



2 轴直线插补实现三角形轮廓轨迹

5.4.1.1.2 HMC_Move2DEx(高级 2 轴直线插补)

5.4.1.1.2.1 功能

二维平面上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的两分轴相对位置（以指令起点时刻位置为坐标原点的坐标位置）为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是根据上一次运动的终点位置做增量运动，即每一步都是增量模式运动。

此指令执行前自动设定合轴的轴参数 Speed = 函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_Move2D(2 轴直线插补)中功能说明部分。

5.4.1.1.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
dDistanceX	Input	LREAL	轴 X 运动距离
dDistanceX	Input	LERAL	轴 Y 运动距离
dSpeed	Input	LREAL	合轴运动速度
HMC_Move2DEx	Output	UDINT	返回值

5.4.1.1.2.3 指令类型

轴运动缓存指令。

5.4.1.1.2.4 示例

以下示例通过 HMC_Move2DEx 指令实现高级 2 轴直线插补，此指令执行前自动设定合轴的轴参数 Speed=函数参数 dSpeed，合轴以该速度为目标速度进行解算。详细内容可参考 HMC_Move2D(2 轴直线插补)中功能说明部分。

```
Axis10.Speed := 1000;        (*设定运动速度*)


HMC_Move2DEx(10，0，1，30000, 40000,2000);    (*投送 HMC_ Move2DEx 指令,合轴轴 10 将以最大速度 2000 运行（而非 1000）*)
```

5.4.1.1.3 HMC_MoveAbs2D(2 轴绝对值直线插补)

5.4.1.1.3.1 功能

二维平面上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的两分轴绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是绝对位置，绝对位置指的是相对于该轴原点的位置。

详细内容可参考 HMC_Move2D(2 轴直线插补)中功能说明部分。

5.4.1.1.3.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
dDposX	Input	LREAL	轴 X 运动距离
dDposY	Input	LERAL	轴 Y 运动距离
dSpeed	Input	LREAL	合轴运动速度
HMC_Move2DEx	Output	UDINT	返回值

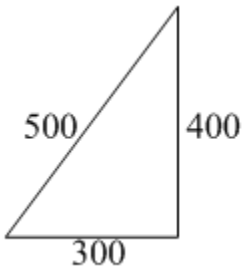
5.4.1.1.3.3 指令类型

轴运动缓存指令。

5.4.1.1.3.4 示例

以下示例通过 HMC_MoveAbs2D 指令实现 2 轴绝对值直线插补。以本指令开始执行时刻为起点，以指令输入参数设置的两分轴绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是绝对位置，绝对位置指的是相对于该轴原点的位置。

以一个平面切割轮廓为三角形的物料为例，三角形如下：



三角形轮廓

编写程序如下：

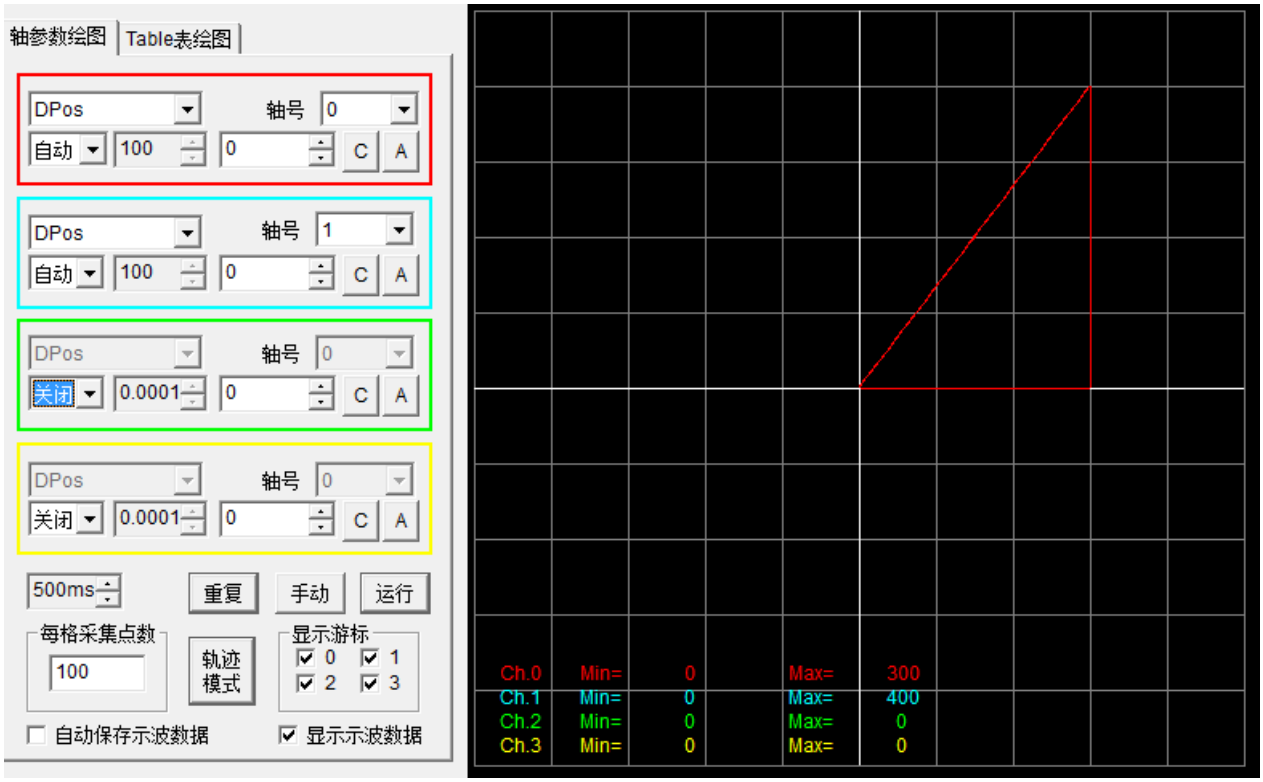
```
(*分别设置使用的三个轴的轴类型*)
HMC_SetAxisType(10,0);
HMC_SetAxisType(0,10);
HMC_SetAxisType(1,10);
```

HMC_MoveAbs2D(10,0,1,300,0);　　(*X Y 轴运行至（300,0）点。第 1 条直角边*)

HMC_MoveAbs2D(10,0,1,300,400);(*X Y 轴运行至（300,400）点。第 2 条直角边*)

HMC_MoveAbs2D(10,0,1,0,0);　　(*X Y 轴运行至（0,0）点。斜边*)

通过 AT 示波器观察实际运行轨迹如下：



2 轴直线插补实现三角形轮廓轨迹

5.4.1.1.4　　HMC_MoveAbs2DEx(**高级 2 轴绝对值直线插补**)

5.4.1.1.4.1　　功能

二维平面上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的两分轴绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是绝对位置，绝对位置指的是相对于该轴原点的位置。

此指令可设定合轴的轴参数 Speed=函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_Move2D(2 轴直线插补)中功能说明部分。

5.4.1.1.4.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
dDposX	Input	LREAL	轴 X 运动距离
dDposY	Input	LERAL	轴 Y 运动距离
dSpeed	Input	LREAL	合轴运动速度
HMC_Move2DEx	Output	UDINT	返回值

5.4.1.1.4.3 指令类型

轴运动缓存指令。

5.4.1.1.4.4 应用举例

以下示例通过 HMC_MoveAbs2DEx 指令实现高级 2 轴绝对直线插补，此指令执行前自动设定合轴的轴参数 Speed=函数参数 dSpeed，合轴以该速度为目标速度进行解算。详细内容可参考 HMC_MoveAbs2D(2 轴绝对值直线插补)中功能说明部分。

```
Axis10.Speed := 1000;        (*设定运动速度*)


HMC_MoveAbs2DEx(10，0，1，30000, 40000,2000);   (*投送 HMC_ MoveAbs2DEx 指令,合轴轴 10 将以最大速度 2000 运行（而非 1000）*)
```

5.4.1.2 三轴直线插补

5.4.1.2.1 HMC_Move3D(3 轴直线插补)

5.4.1.2.1.1 功能

三维空间上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的三分轴相对位置（以指令起点时刻位置为坐标原点的坐标位置）为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是根据上一次运动的终点位置做增量运动，即每一步都是增量模式运动。

插补合轴的目标位置在指令开始执行时刻位置基础上，增加起始点至终点的连线长度。插补运动合轴

运动速度由对应轴参数 Speed 设定，加减速分别由合轴轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。

插补分轴的目标位置在指令开始执行时刻位置基础上，增量该轴指定的相对运动距离。插补运动分轴的运行速度、加减速度与该轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

插补分轴遇到限位时，选取合轴分解计算的分轴减速度和自身快减速度两者的最大值进行最优保护停止，详见 HMC_HwLimitConfig（设置硬限位开关）。

不支持 Merge 功能。

5.4.1.2.1.2 参数

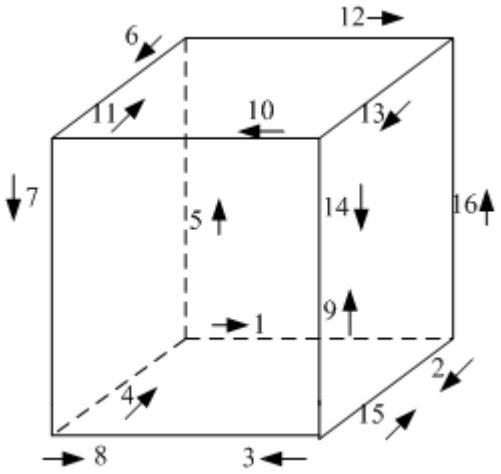
参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	插补合运动轴号
ucAxisX	INPUT	USINT	插补轴 X 的轴号
ucAxisY	INPUT	USINT	插补轴 Y 的轴号
ucAxisZ	INPUT	USINT	插补轴 Z 的轴号
dDistanceX	INPUT	LREAL	轴 X 运动距离
dDistanceY	INPUT	LREAL	轴 Y 运动距离
dDistanceZ	INPUT	LREAL	轴 Z 运动距离
HMC_Move3D	OUTPUT	UDINT	轴指令 ID

5.4.1.2.1.3 指令类型

轴运动缓存指令。

5.4.1.2.1.4 示例

画一个空间的边长为 1000 的正立方体，允许同一条边重复绘制，预期按如下顺序依次画各条边，数字代表轨迹顺序，箭头代表轨迹方向，最终完成正方体的绘制。



正方体示意图

编写程序如下：

（*起始位置为原点*）

HMC_Move3D (5 ,0,1,2, 1000, 0, 0);　(*轨迹 1*)

HMC_Move3D (5 ,0,1,2, 0, 1000, 0);　(*轨迹 2*)

HMC_Move3D (5 ,0,1,2, -1000, 0, 0);　(*轨迹 3*)

HMC_Move3D (5 ,0,1,2, 0, -1000, 0);　(*轨迹 4*)

HMC_Move3D (5 ,0,1,2, 0,0, 1000);　(*轨迹 5*)

HMC_Move3D (5 ,0,1,2, 0, 1000, 0);　(*轨迹 6*)

HMC_Move3D (5 ,0,1,2, 0, 0, -1000);　(*轨迹 7*)

HMC_Move3D (5 ,0,1,2, 1000, 0, 0);　(*轨迹 8*)

HMC_Move3D (5 ,0,1,2, 0, 0, 1000);　(*轨迹 9*)

HMC_Move3D (5 ,0,1,2, -1000, 0, 0);　(*轨迹 10*)

HMC_Move3D (5 ,0,1,2, 0, -1000, 0);　(*轨迹 11*)

HMC_Move3D (5 ,0,1,2, 1000, 0, 0);　(*轨迹 12*)

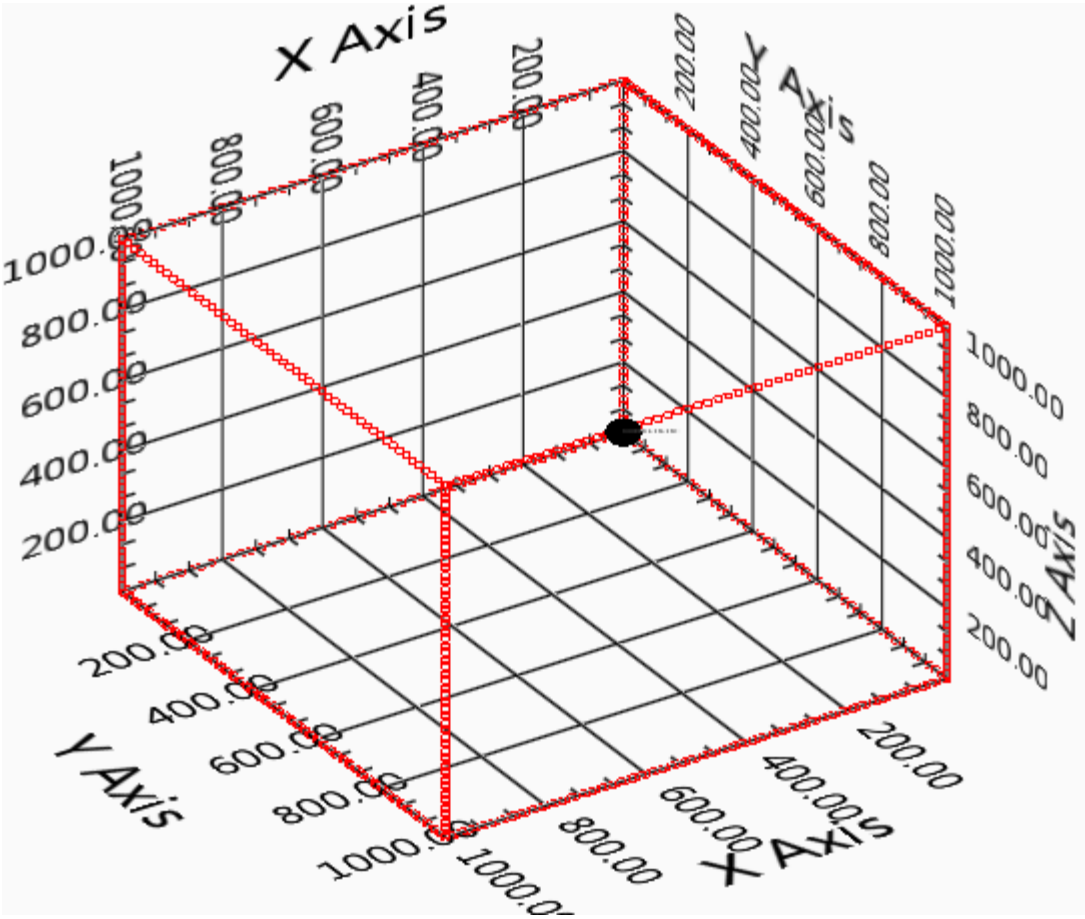
HMC_Move3D (5 ,0,1,2, 0, 1000, 0);　(*轨迹 13*)

HMC_Move3D (5 ,0,1,2, 0, 0, -1000);　　(*轨迹 14*)

HMC_Move3D (5 ,0,1,2, 0, -1000, 0);　　(*轨迹 15*)

HMC_Move3D (5 ,0,1,2, 0, 0, 1000);　　(*轨迹 16*)

运行以上程序，由于目前 AT 示波器不支持绘制三维轨迹，因此通过在示波器记录三个分轴的 Dpos 值，并将记录数据导出，在三维图形工具中导入记录数据：



三维示意图

5.4.1.2.2　　HMC_Move3DEx(**高级 3 轴直线插补**)

5.4.1.2.2.1　　功能

三维空间上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的三分轴相对位置（以指令起点时刻位置为坐标原点的坐标位置）为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是根据上一次运动的终点位置做增量运动，即每一步都是增量模式运动。

此指令可设定合轴的轴参数 Speed=函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_Move3D(3 轴直线插补)中功能说明部分。

不支持 Merge 功能。

5.4.1.2.2.2 参数

参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	插补合运动轴号
ucAxisX	INPUT	USINT	插补轴 X 的轴号
ucAxisY	INPUT	USINT	插补轴 Y 的轴号
ucAxisZ	INPUT	USINT	插补轴 Z 的轴号
dDistanceX	INPUT	LREAL	轴 X 运动距离
dDistanceY	INPUT	LREAL	轴 Y 运动距离
dDistanceZ	INPUT	LREAL	轴 Z 运动距离
dSpeed	INPUT	LREAL	合轴目标速度
HMC_Move3DEx	OUTPUT	UDINT	轴指令 ID

5.4.1.2.2.3 指令类型

轴运动缓存指令。

5.4.1.2.2.4 示例

```
Axis10.Speed := 1000;        (*设定运动速度*)

HMC_Move3DEx(10，0，1，2，30000, 40000,5000，2000);      (*投送 HMC_ Move3DEx 指令,合轴轴 10 将以最大速度 2000 运行（而非 1000）*)
```

5.4.1.2.3 HMC_MoveAbs3D(3 **轴绝对值直线插补**)

5.4.1.2.3.1 功能说明

三维平面上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的三分轴绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是绝对位置，绝对位置指的是相对于该轴原点的位置。

详细内容可参考 HMC_Move3D(3 轴直线插补)中功能说明部分。

本指令目前暂不支持 Merge 功能。

5.4.1.2.3.2 参数

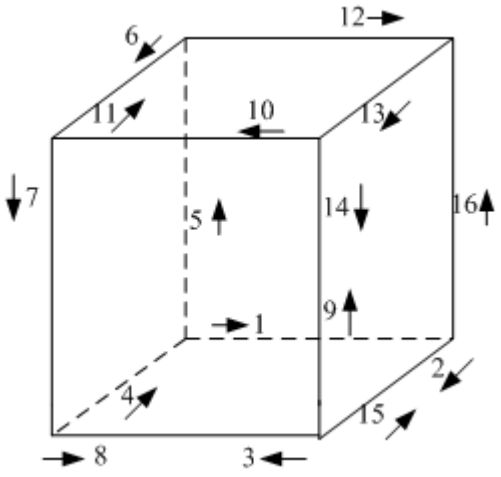
参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	插补合运动轴号
ucAxisX	INPUT	USINT	插补轴 X 的轴号
ucAxisY	INPUT	USINT	插补轴 Y 的轴号
ucAxisZ	INPUT	USINT	插补轴 Z 的轴号
dDposX	INPUT	LREAL	轴 X 目标位置
dDposY	INPUT	LREAL	轴 Y 目标位置
dDposZ	INPUT	LREAL	轴 Z 目标位置
HMC_MoveAbs3D	OUTPUT	UDINT	轴指令 ID

5.4.1.2.3.3 指令类型

轴运动缓存指令。

5.4.1.2.3.4 示例

画一个空间的边长为 1000 的正立方体，允许同一条边重复绘制，预期按如下顺序依次画各条边，数字代表轨迹顺序，箭头代表轨迹方向，最终完成正方体的绘制。



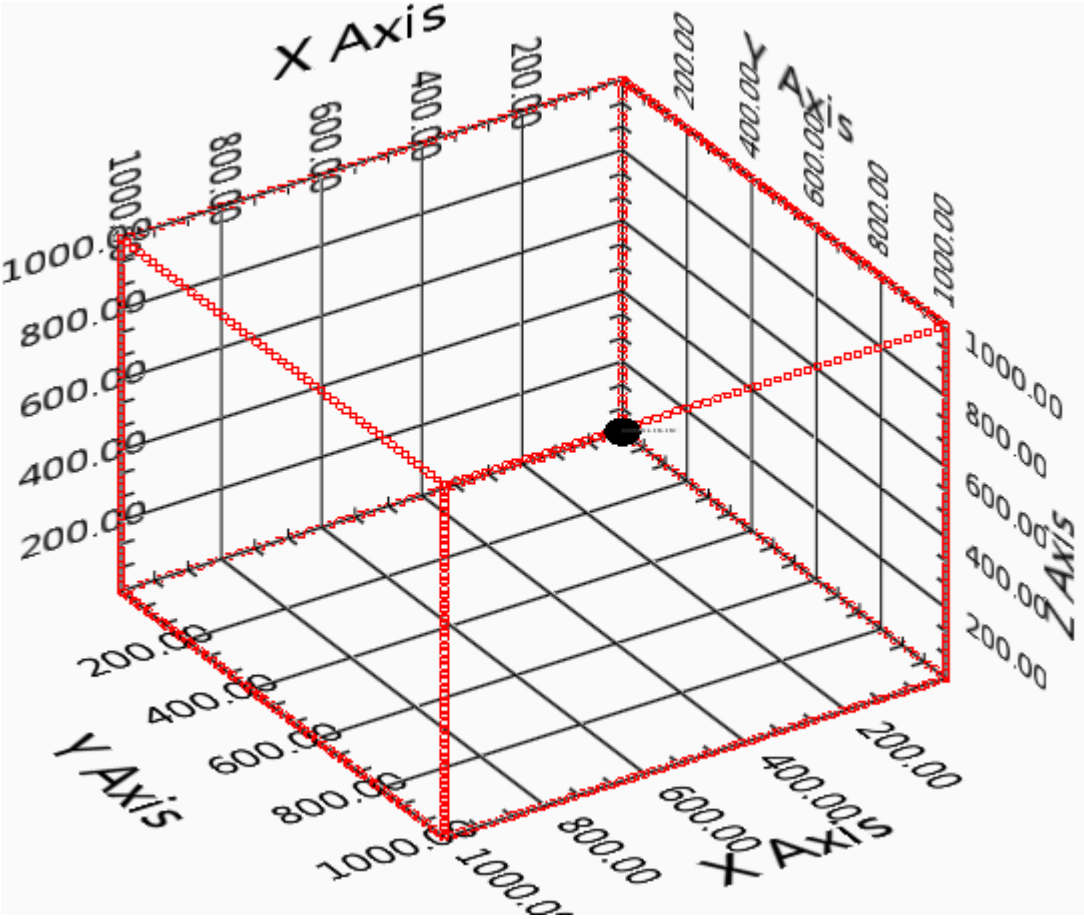
正方体示意图

编写程序如下：

（*起始位置为原点*）

```
HMC_MoveAbs3D (5,0,1,2,1000,0,0);        (*轨迹 1*)
HMC_MoveAbs3D (5,0,1,2,1000,1000,0);     (*轨迹 2*)
HMC_MoveAbs3D (5 ,0,1,2,0,1000,0);       (*轨迹 3*)
HMC_MoveAbs3D (5,0,1,2,0,0,0);           (*轨迹 4*)
HMC_MoveAbs3D (5,0,1,2,0,0,1000);        (*轨迹 5*)
HMC_MoveAbs3D (5,0,1,2,0,1000,1000);     (*轨迹 6*)
HMC_MoveAbs3D (5,0,1,2,0,1000,0);        (*轨迹 7*)
HMC_MoveAbs3D (5,0,1,2,1000,1000,0);     (*轨迹 8*)
HMC_MoveAbs3D (5,0,1,2,1000,1000,1000);  (*轨迹 9*)
HMC_MoveAbs3D (5,0,1,2,0,1000,1000);     (*轨迹 10*)
HMC_MoveAbs3D (5,0,1,2,0,0,1000);        (*轨迹 11*)
HMC_MoveAbs3D (5,0,1,2,1000,0,1000);     (*轨迹 12*)
HMC_MoveAbs3D (5,0,1,2,1000,1000,1000);  (*轨迹 13*)
HMC_MoveAbs3D (5,0,1,2,1000,1000,0);     (*轨迹 14*)
HMC_MoveAbs3D (5,0,1,2,1000,0,0);        (*轨迹 15*)
HMC_MoveAbs3D (5,0,1,2,1000,0,1000);     (*轨迹 16*)
```

运行以上程序，由于目前 AT 示波器不支持绘制三维轨迹，因此通过在示波器记录三个分轴的 Dpos 值，并将记录数据导出，在三维图形工具中导入记录数据：



三维示意图

5.4.1.2.4 HMC_MoveAbs3DEx(**高级 3 轴绝对值直线插补**)

5.4.1.2.4.1 功能

三维平面上运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的三分轴绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是绝对位置，绝对位置指的是相对于该轴原点的位置。

此指令可设定合轴的轴参数 Speed = 函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_Move3D(3 轴直线插补)中功能说明部分。

不支持 Merge 功能。

5.4.1.2.4.2 参数

参数	声明	数据类	说明

		型	
ucAxisID	INPUT	USINT	插补合运动轴号
ucAxisX	INPUT	USINT	插补轴 X 的轴号
ucAxisY	INPUT	USINT	插补轴 Y 的轴号
ucAxisZ	INPUT	USINT	插补轴 Z 的轴号
dDposX	INPUT	LREAL	轴 X 目标位置
dDposY	INPUT	LREAL	轴 Y 目标位置
dDposZ	INPUT	LREAL	轴 Z 目标位置
dSpeed	INPUT	LREAL	合轴目标速度
HMC_MoveAbs3D	OUTPUT	UDINT	轴指令 ID

5.4.1.2.4.3 指令类型

轴运动缓存指令。

5.4.1.2.4.4 示例

参考 HMC_MoveAbs3D(3 轴绝对值直线插补)。

5.4.1.3 多轴直线插补

5.4.1.3.1 HMC_MoveND(N 轴直线插补)

5.4.1.3.1.1 功能

可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的线性插补分轴相对位置（以指令起点时刻位置为坐标原点的坐标位置）为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是根据上一次运动的终点位置做增量运动，即每一步都是增量模式运动。

插补合轴的目标位置在指令开始执行时刻位置基础上，增加起始点至终点的连线长度。插补运动合轴运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。

插补分轴的目标位置在指令开始执行时刻位置基础上，增量该轴指定的相对运动距离。插补运动分轴的运行速度、加减速度与该轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

N 轴插补指令的分轴分为两种类型：线性插补分轴和独立插补分轴。线性插补分轴决定插补合运动的实际轨迹，独立插补分轴与合运动轨迹无关，独立插补分轴可实现与合轴或分轴的同步运动，即指令结束时刻，合轴和线性插补分轴完成指令运动距离时，独立插补分轴也同时完成运动距离。

各个线性插补分轴的运动距离决定了合轴的运动距离，合轴运动距离等于各个线性插补分轴运动距离的平方之和再开方。线性插补分轴速度由合轴速度、线性插补分轴运动距离与合轴运动距离的比值决定。

独立插补分轴的运动距离不影响合轴的运动距离，因此称为独立轴，可实现与合轴或插补线性轴的同步运动。独立轴的速度由合轴速度、独立插补分轴运动距离与合轴运动距离的比值决定。

N 轴线性插补中合运动与分运动关系详细描述如下：

数组 pAxis 和 pDistance 分别代表分轴轴号和各分轴运动距离。LN 为线性插补分轴个数（ucLineNum 记为 LN），SN 为独立插补分轴个数（ucSelfNum 记为 SN），即 LN+SN 为分轴总个数，因此 pAxis 和 pDistance 分别指向数组的元素个数 N 应满足：N≥LN+SN。

合运动轴 ucAxisID 的运动距离为 D，速度为 Speed；

LN 个线性插补轴的运动距离为 L(0)，…，L(LN-1)，速度为：SpeedL(0)，…，SpeedL(LN-1)；

SN 个独立插补轴的运动距离为 S(0)，…，S(SN-1)，速度为：SpeedS(0)，…，SpeedS(SN-1)；

合轴运动距离由线性插补分轴运动距离决定，具体如下：

$$D^2 = L(0)^2 + L(1)^2 + \ldots + L(LN-1)^2$$

线性插补分轴速度与合轴速度关系如下：

$$SpeedL(i) = \frac{L(i)}{D} \times Speed \quad (0 \le i \le LN-1)$$

独立插补分轴速度与合轴速度关系如下：

$$SpeedS(j) = \frac{S(j)}{D} \times Speed \quad (0 \le j \le SN-1)$$

本指令不支持 Merge 功能。

5.4.1.3.1.2 参数

参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	插补合运动轴号
pAxis	INPUT	POINTER TO USINT	插补分运动轴号数组首地址
pDistance	INPUT	POINTER TO LREAL	插补分运动轴运动距离首地址
ucLineNum	INPUT	USINT	线性插补轴个数。pAixs 的前 ucLineNum 个元素即为线性插补分轴的轴号，pDistance 的前 ucLineNum 个元素即为线性插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucLineNum>=1
ucSelfNum	INPUT	USINT	独立插补轴个数。pAixs 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的轴号，pDistance 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucSelfNum>=0
HMC_MoveND	OUTPUT	UDINT	轴指令 ID

5.4.1.3.1.3 指令类型

轴运动缓存指令。

5.4.1.3.1.4 示例

```
FOR i:=0 TO 4 DO
 AxisArry[i] := i;
END_FOR      (*数组 AxisArry 前 5 个元素为插补分轴的轴号，依次为 0~4*)

DistanceArry[0] := 10000;
DistanceArry[1] := 20000;
DistanceArry[2] := 30000;
DistanceArry[3] := 180;
DistanceArry[4] := 120;

HMC_MoveND (15 ,ADR(AxisArry), ADR(DistanceArry), 3, 2);      (*前 3 个分轴为线性插补分轴，轴序号为 0、1、2，接下来 2 个分轴为独立插补分轴，轴序号为 3、4*)
```

5.4.1.3.2 HMC_MoveNDEx(高级 N 轴直线插补)

5.4.1.3.2.1 功能

可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。以本指令开始执行时刻为起点，以指令输入参数设置的线性插补分轴相对位置（以指令起点时刻位置为坐标原点的坐标位置）为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是根据上一次运动的终点位置做增量运动，即每一步都是增量模式运动。

此指令可设定合轴的轴参数 Speed=函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_MoveND(N 轴直线插补)中功能说明部分。

5.4.1.3.2.2 参数

参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	插补合运动轴号
pAxis	INPUT	POINTER TO USINT	插补分运动轴号数组首地址
pDistance	INPUT	POINTER TO LREAL	插补分运动轴运动距离首地址
ucLineNum	INPUT	USINT	线性插补轴个数。pAixs 的前 ucLineNum 个元素即为线性插补分轴的轴号，pDistance 的前 ucLineNum 个元素即为线性插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucLineNum>=1
ucSelfNum	INPUT	USINT	独立插补轴个数。pAixs 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的轴号，pDistance 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucSelfNum>=0
dSpeed	INPUT	LREAL	合轴 ucAxisID 的目标速度
HMC_MoveND	OUTPUT	UDINT	轴指令 ID

5.4.1.3.2.3 指令类型

轴运动缓存指令。

5.4.1.3.2.4 示例

```
FOR i:=0 TO 4 DO
   AxisArry[i] := i;
```

```
END_FOR      (*数组 AxisArry 前 5 个元素为插补分轴的轴号，依次为 0~4*)

DistanceArry[0] := 10000;
DistanceArry[1] := 20000;
DistanceArry[2] := 30000;
DistanceArry[3] := 180;
DistanceArry[4] := 120;
HMC_MoveNDEX (15 ,ADR(AxisArry), ADR(DistanceArry), 3, 2,1000);
```

5.4.1.3.3 HMC_MoveAbsND (N 轴绝对值直线插补)

5.4.1.3.3.1 功能

可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。以指令输入参数设置的线性插补分轴的绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是绝对位置，绝对位置指的是相对于该轴原点的位置。

详细内容可参考 HMC_MoveND(N 轴直线插补)中功能说明部分。

5.4.1.3.3.2 参数

参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	插补合运动轴号
pAxis	INPUT	POINTER TO USINT	插补分运动轴号数组首地址
pDpos	INPUT	POINTER TO LREAL	插补分运动轴目标位置首地址
ucLineNum	INPUT	USINT	线性插补轴个数。pAixs 的前 ucLineNum 个元素即为线性插补分轴的轴号，pDistance 的前 ucLineNum 个元素即为线性插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucLineNum>=1
ucSelfNum	INPUT	USINT	独立插补轴个数。pAixs 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的轴号，pDistance 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucSelfNum>=0
HMC_MoveND	OUTPUT	UDINT	轴指令 ID

5.4.1.3.3.3 指令类型

轴运动缓存指令。

5.4.1.3.3.4 示例

参考 HMC_MoveND(N 轴直线插补)。

5.4.1.3.4 HMC_MoveAbsNDEx (高级 N 轴绝对值直线插补)

5.4.1.3.4.1 功能说明

可以实现任意 N 维坐标系统的运动轮廓为直线的插补功能。以指令输入参数设置的线性插补分轴的绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。此指令的目标位置都是绝对位置，绝对位置指的是相对于该轴原点的位置。

此指令可设定合轴的轴参数 Speed=函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_MoveND(N 轴直线插补)中功能说明部分。

5.4.1.3.4.2 参数

参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	插补合运动轴号
pAxis	INPUT	POINTER TO USINT	插补分运动轴号数组首地址
pDpos	INPUT	POINTER TO LREAL	插补分运动轴目标位置首地址
ucLineNum	INPUT	USINT	线性插补轴个数。pAixs 的前 ucLineNum 个元素即为线性插补分轴的轴号，pDistance 的前 ucLineNum 个元素即为线性插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucLineNum>=1
ucSelfNum	INPUT	USINT	独立插补轴个数。pAixs 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的轴号，pDistance 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的运动距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucSelfNum>=0
dSpeed	INPUT	LREAL	合轴 ucAxisID 的目标速度
HMC_MoveND	OUTPUT	UDINT	轴指令 ID

5.4.1.3.4.3 指令类型

轴运动缓存指令。

5.4.1.3.4.4 示例

参考 HMC_MoveND(N 轴直线插补)。

5.4.2 圆弧插补

5.4.2.1 平面圆弧插补

5.4.2.1.1 HMC_MoveCircle（圆弧插补）

5.4.2.1.1.1 功能

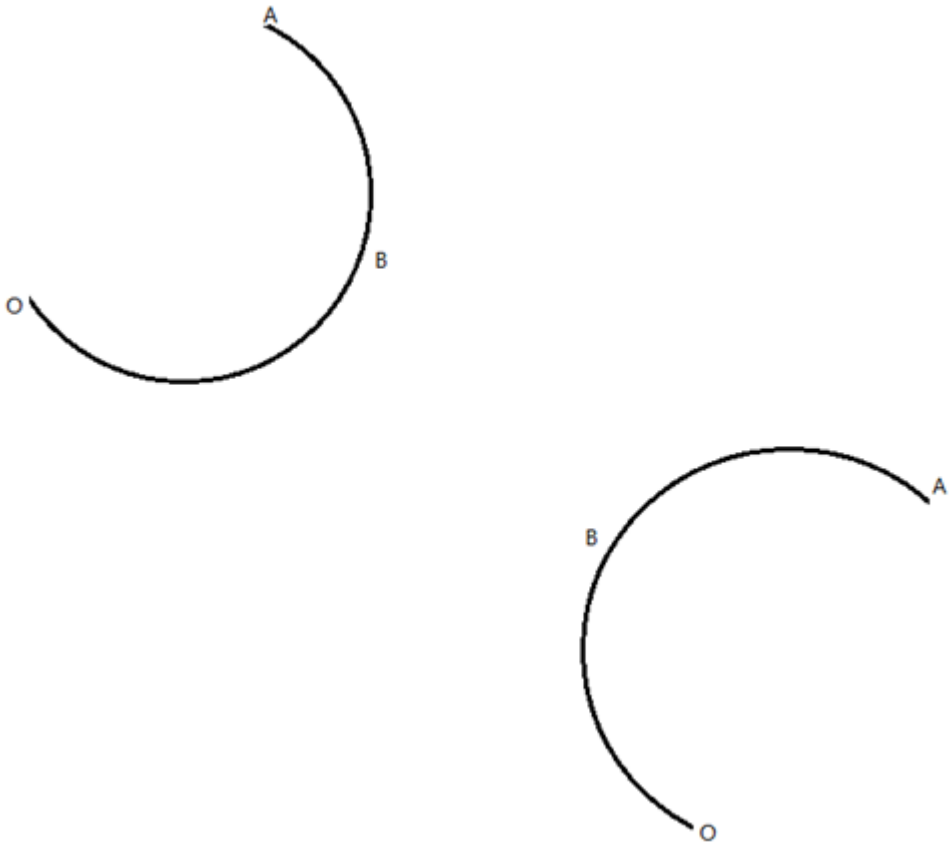
二维平面上运动轮廓为圆弧的插补功能。

插补合轴的目标位置在指令开始执行时刻位置基础上，增加指令轨迹圆弧长度。插补运动合轴运动速度由对应轴参数 Speed 设定，加减速分别由合轴轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。插补运动分轴的运行速度、加减速度与该轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

本指令支持三种模式：

（1） ucMode=0

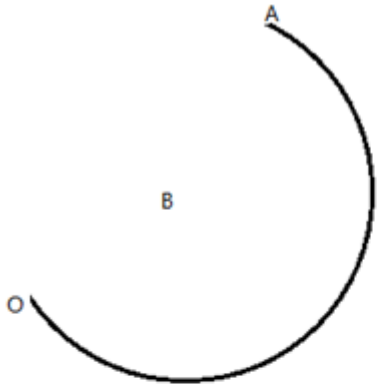
以圆弧指令开始时刻位置为起点，A 为终点，B 为中间点，一段圆弧。A、B 点的坐标都是以 O 点为坐标原点。如果 O、B、A 三点一线且 B 在中间时，按照直线处理。

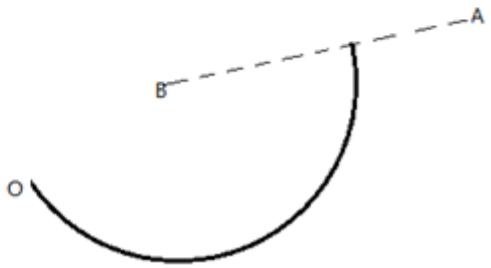


运动轨迹

（2） ucMode=1

以圆弧指令开始时刻为起点，B 为圆心，A 为终点，逆时针方向一段圆弧。A、B 点的坐标都是以 O 点为坐标原点。起点和圆心即可确定圆弧，如果 A 是圆上一点，则 A 就为指令终点，如下图左图；否则即为圆心到 A 点射线与圆弧的交点为终点，如下图右图。如果 A 点为（0,0）则逆时针画一整圆。

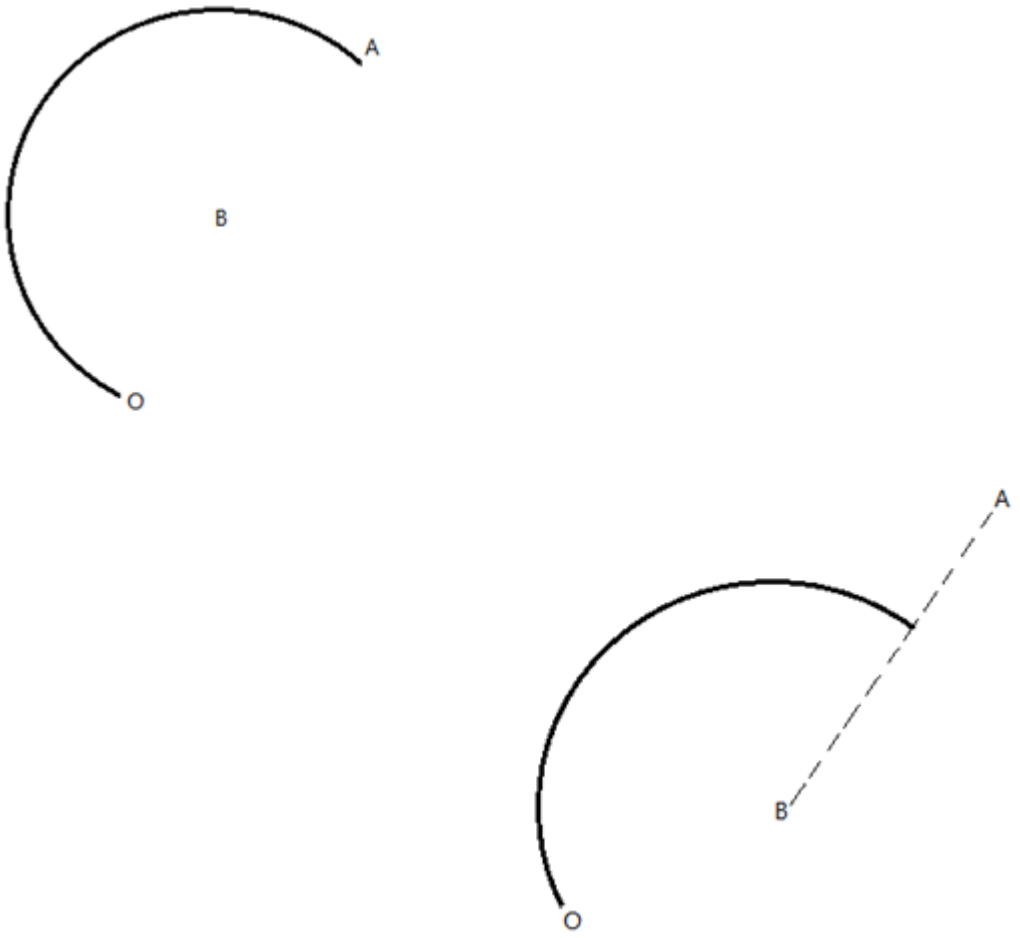




运动轨迹

（3） ucMode=3

以圆弧指令开始时刻为起点，B 为圆心，A 为终点，顺时针方向一段圆弧。A、B 点的坐标都是以 O 点为坐标原点。起点和圆心即可确定圆弧，如果 A 是圆上一点，则 A 就为指令终点，如下图左图；否则即为圆心到 A 点射线与圆弧的交点为终点，如下图右图。如果 A 点为（0,0）则顺时针画一整圆。



运动轨迹

支持 Merge 功能。

5.4.2.1.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
ucMode（O 为当前点）	Input	USINT	0：A 为终点，B 为中间某点的一段圆弧；O、B、A 三点一线且 B 在中间时，按照直线处理 1：A 为终点，B 为圆弧的圆心，且按照逆时针方向走 3：A 为终点，B 为圆弧的圆心，且按照顺时针方向走
dAx	Input	LREAL	A 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，A 的 X 坐标）
dAy	Input	LREAL	A 点在 Y 轴位置-O 点在 Y 轴的位置

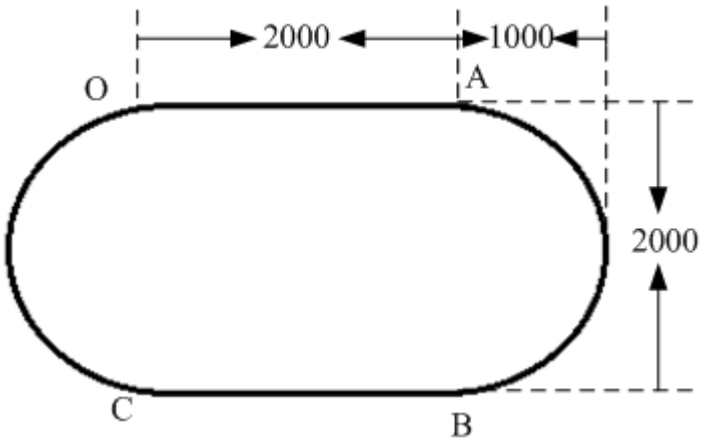
			（将 O 视为坐标系原点时，A 的 Y 坐标）
dBx	Input	LREAL	B 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，B 的 X 坐标）
dBy	Input	LREAL	B 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，B 的 Y 坐标）
HMC_MoveCircle	Output	UDINT	轴指令 ID

5.4.2.1.1.3 指令类型

轴运动缓存指令。

5.4.2.1.1.4 示例

切割如下工件。



工件尺寸图

以上图 O 点为起始点，编写程序如下：

```
HMC_Move2D(6,0,1,2000,0);
HMC_MoveCircle(6,0,1,3,0,-2000,0,-1000);        (*以当前点 A 为起始点，终点和圆心都以 A 点为原点*)
HMC_Move2D(6,0,1,-2000,0);
HMC_MoveCircle(6,0,1,3,0,2000,0,1000);          (*以当前点 C 为起始点，终点和圆心都以 C 点为原点*)
```

运行程序，在 AT 示波器中得到轨迹如下：

轨迹示意图

5.4.2.1.2 HMC_MoveCircleEx（高级圆弧插补）

5.4.2.1.2.1 功能

二维平面上运动轮廓为圆弧的插补功能。

插补合轴的目标位置在指令开始执行时刻位置基础上，增加指令轨迹圆弧长度。插补运动合轴运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。插补运动分轴的运行速度、加减速度与该轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

此指令可设定合轴的轴参数 Speed = 函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_MoveCircle（圆弧插补）中功能说明部分。

5.4.2.1.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号

ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
ucMode（O 为当前点）	Input	USINT	0：A 为终点，B 为中间某点的一段圆弧；O、B、A 三点一线且 B 在中间时，按照直线处理 1：A 为终点，B 为圆弧的圆心，且按照逆时针方向走 3：A 为终点，B 为圆弧的圆心，且按照顺时针方向走
dAx	Input	LREAL	A 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，A 的 X 坐标）
dAy	Input	LREAL	A 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，A 的 Y 坐标）
dBx	Input	LREAL	B 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，B 的 X 坐标）
dBy	Input	LREAL	B 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，B 的 Y 坐标）
dSpeed	Input	LREAL	合轴目标速度
HMC_MoveCircle	Output	UDINT	轴指令 ID

5.4.2.1.2.3 指令类型

轴运动缓存指令。

5.4.2.1.2.4 示例

参考 HMC_MoveCircle（圆弧插补）。

5.4.2.2 球面插补

5.4.2.2.1 HMC_MoveSpherical（球弧插补）

5.4.2.2.1.1 功能

三维空间上运动轮廓为球弧的插补功能。球弧即空间某个平面的圆弧，或者说球面与某个平面相交的空间圆上的一段圆弧。

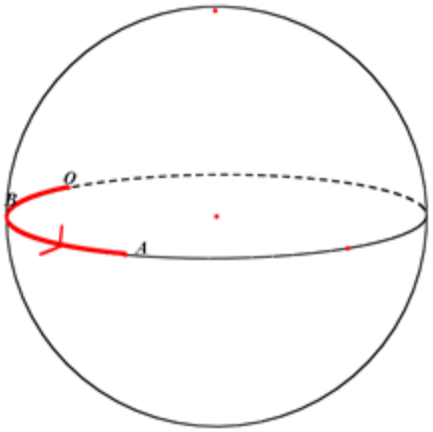
插补合轴的目标位置在指令开始执行时刻位置基础上，增加指令轨迹球弧长度。插补运动合轴运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。插补运动分轴的运行速度、加减速度与该轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

本指令支持四种模式：

（1） ucMode=0

以指令开始时刻为起点 O，B 为中间点，A 为终点，三点确定一段球弧。A、B 点的坐标都是以 O 点为坐标原点。如果 O、B、A 三点一线且 B 点为中间点时，按照直线处理。

下图是 O、B、A 三点确定的空间一段球弧，运动方向为 O->B->A，红色曲线为球弧运动轨迹曲线，箭头表示方向。

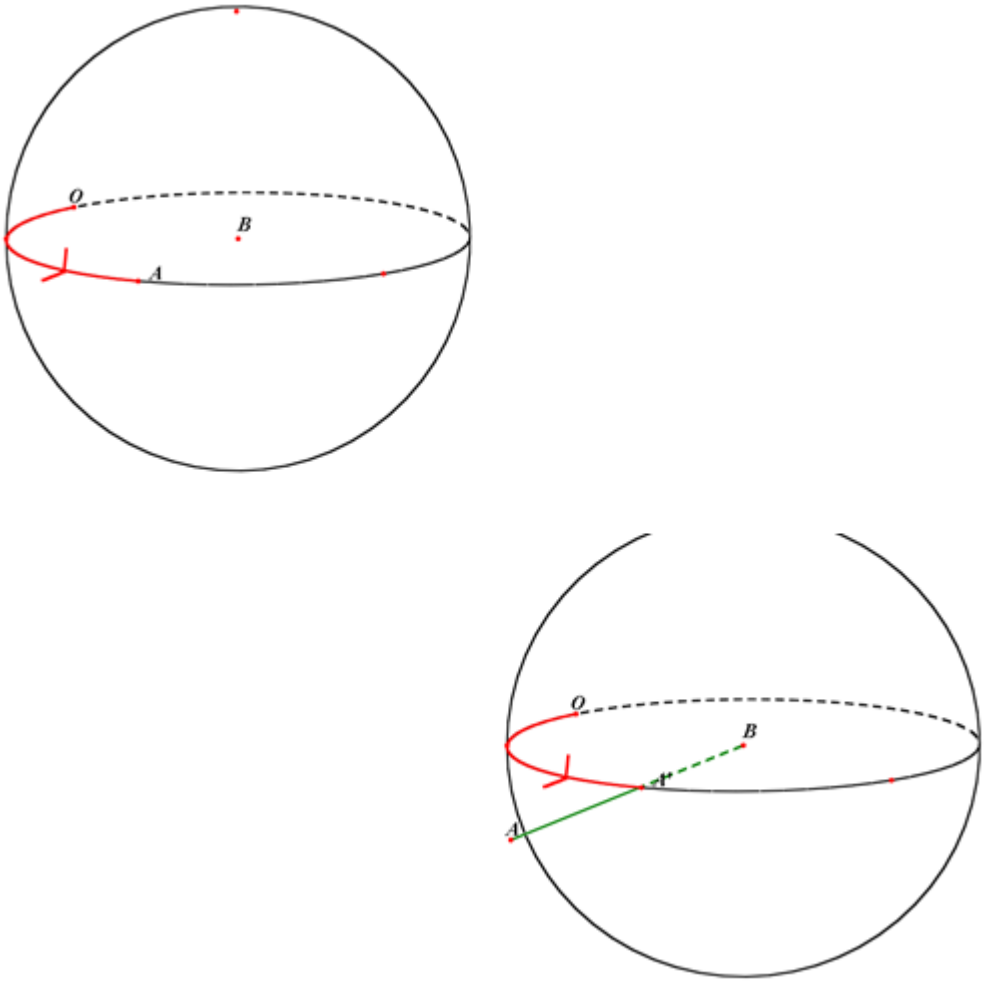


运动轨迹

（2） ucMode=1

以指令开始时刻为起点 O，B 为空间圆弧的圆心，A 为终点，按照 O 到 A 的最短距离为运动方向，确定一段空间圆弧。

下图是以 O 为起点、B 为球弧圆心、A 为终点的空间一段球弧，当 A 点不在球弧上时，AB 连线与球弧交点则为终点，红色曲线为球弧运动轨迹曲线，箭头表示方向。

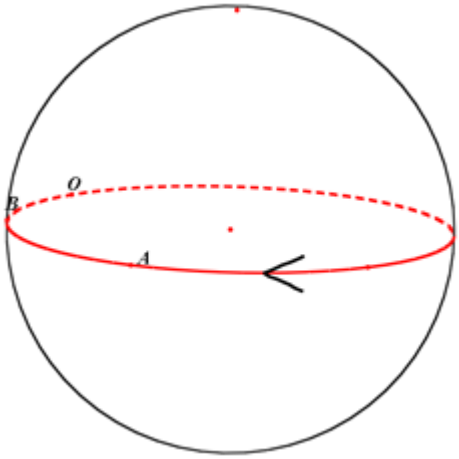


运动轨迹

(3) ucMode=2

以指令开始时刻为起点 O，A、B 为中间点，方向为 O->A->B，运动完整圆弧。

下图是 O、B、A 三点确定的空间球弧，运动方向为 O->A->B，红色曲线即三点确定的完整球弧为运动轨迹曲线，箭头表示方向。

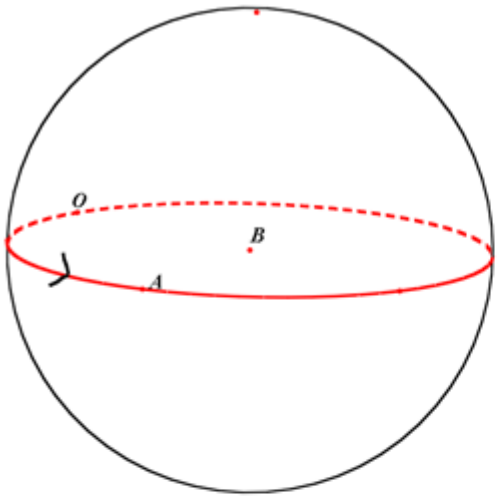


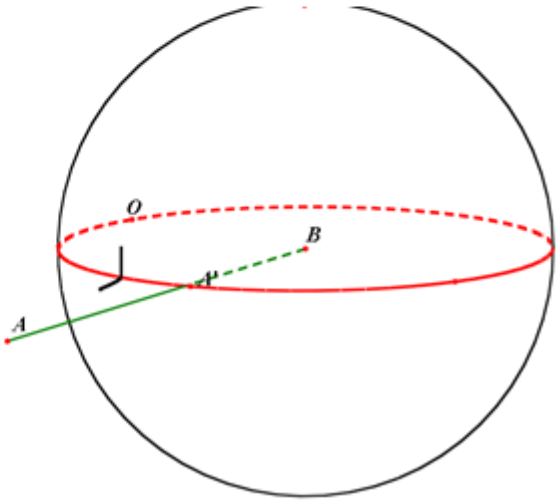
运动轨迹

（4） ucMode=3

以指令开始时刻为起点 O，B 为空间圆弧的圆心，A 为中间点，按照 O 到 A 的最短距离为运动方向，运动完整圆弧。

下图是以 O 为起点、B 为球弧圆心、A 为中间点的空间一段球弧，按照 O 到 A 的最短距离为运动方向，即为逆时针方向。红色曲线为球弧运动轨迹曲线，箭头表示方向。





运动轨迹

5.4.2.2.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
ucAxisZ	Input	USINT	插补轴 Z 的轴号
ucMode（O 为当前点）	Input	USINT	0：A 为终点，B 为中间某点的一段圆弧，O、B、A 三点一线且 B 点为中间点时，按照直线处理 1：A 为终点，B 为空间圆弧的圆心的一段圆弧，系统确定方向方法：使 O 到 A 的最短距离（等距特殊情况系统自动处理）。注意：用户设置时 OBA 三点不可共线

			2：A、B 为中间某点，运动完整圆弧，方向为 O->A->B 3：A 为中间某点，B 为空间圆弧的圆心，运动完整圆弧，系统确定方向方法：使 O 到 A 的最短距离（等距特殊情况系统自动处理）。注意：用户设置时 OBA 三点不可共线
dAx	Input	LREAL	A 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，A 的 X 坐标）
dAy	Input	LREAL	A 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，A 的 Y 坐标）
dAZ	Input	LREAL	A 点在 Z 轴位置-O 点在 Z 轴的位置 （将 O 视为坐标系原点时，A 的 Z 坐标）
dBx	Input	LREAL	B 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，B 的 X 坐标）
dBy	Input	LREAL	B 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，B 的 Y 坐标）
dBZ	Input	LREAL	B 点在 Z 轴位置-O 点在 Z 轴的位置 （将 O 视为坐标系原点时，B 的 Z 坐标）
HMC_MoveSpherical	Output	UDINT	轴指令 ID

5.4.2.2.1.3 指令类型

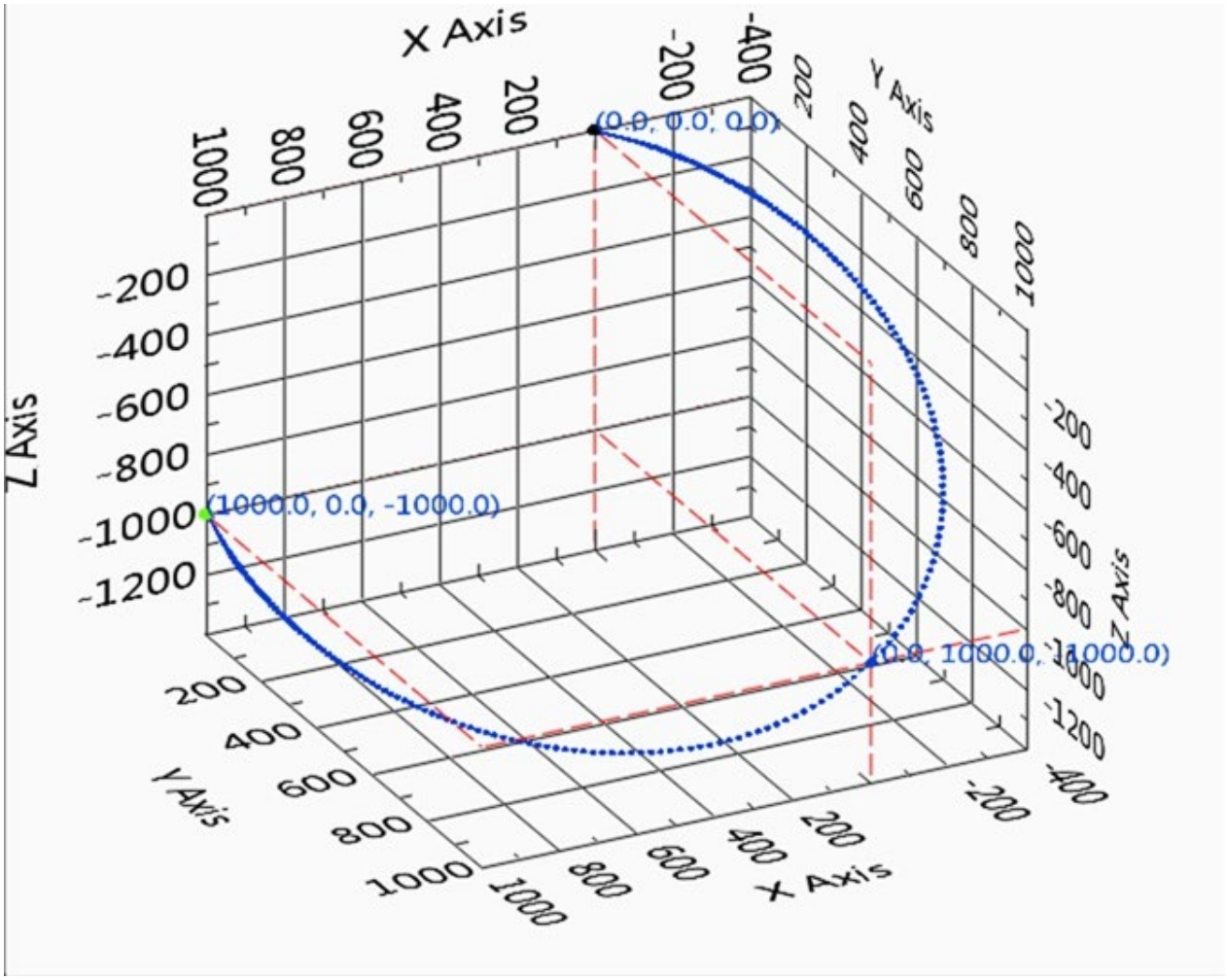
非阻塞指令，轴运动缓存指令

5.4.2.2.1.4 示例

以(0, 0, 0)为起始点，经过(0, 1000, -1000)点，终点为(1000, 0, -1000)点，使用模式 0，编写程序如下：

HMC_MoveSpherical (0,1,2,3,0,1000,0,-1000,0,1000,-1000);

AT 示波器记录各分轴 Dpos，将数据导出，通过第三方软件读取数据绘制三维图形如下：



三维示意图

5.4.2.2.2 HMC_MoveSphericalEx（高级球弧插补）

5.4.2.2.2.1 功能说明

三维空间上运动轮廓为球弧的插补功能。球弧即空间某个平面的圆弧，或者说球面与某个平面相交的空间圆上的一段圆弧。

插补合轴的目标位置在指令开始执行时刻位置基础上，增加指令轨迹球弧长度。插补运动合轴运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。插补运动分轴的运行速度、加减速度与该轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

此指令可设定合轴的轴参数 Speed=函数参数 dSpeed，合轴以该速度为目标速度进行解算。

本指令支持四种模式，详细可见 HMC_MoveSpherical（球弧插补）。

5.4.2.2.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
ucAxisZ	Input	USINT	插补轴 Z 的轴号
ucMode（O 为当前点）	Input	USINT	0：A 为终点，B 为中间某点的一段圆弧，O、B、A 三点一线且 B 点为中间点时，按照直线处理 1：A 为终点，B 为空间圆弧的圆心的一段圆弧，系统确定方向方法：使 O 到 A 的最短距离（等距特殊情况系统自动处理）。注意：用户设置时 OBA 三点不可共线 2：A、B 为中间某点，运动完整圆弧，方向为 O->A->B 3：A 为中间某点，B 为空间圆弧的圆心，运动完整圆弧，系统确定方向方法：使 O 到 A 的最短距离（等距特殊情况系统自动处理）。注意：用户设置时 OBA 三点不可共线
dAx	Input	LREAL	A 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，A 的 X 坐标）
dAy	Input	LREAL	A 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，A 的 Y 坐标）
dAZ	Input	LREAL	A 点在 Z 轴位置-O 点在 Z 轴的位置 （将 O 视为坐标系原点时，A 的 Z 坐标）
dBx	Input	LREAL	B 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，B 的 X 坐标）
dBy	Input	LREAL	B 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，B 的 Y 坐标）
dBZ	Input	LREAL	B 点在 Z 轴位置-O 点在 Z 轴的位置 （将 O 视为坐标系原点时，B 的 Z 坐标）
dSpeed	Input	LREAL	合轴目标速度
HMC_MoveSphericalEx	Output	UDINT	轴指令 ID

5.4.2.2.2.3 指令类型

非阻塞指令，轴运动缓存指令。

5.4.2.2.2.4 示例

参考 HMC_MoveSpherical（球弧插补）。

5.4.3 螺旋线插补

5.4.3.1 HMC_MoveHelical（螺旋线插补）

5.4.3.1.1 功能

螺旋线插补运动。ucAxisID 为合运动轴，在坐标架 O-XYZ 下，ucAxisX、ucAxisY 两轴执行平面圆弧插补运动，ucAxisZ 轴执行 Z 相线性运动，三轴共同完成圆柱螺旋线轨迹。

插补合轴的目标位置根据不同模式计算方法不同。插补运动合轴运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。插补运动分轴的运行速度、加减速度与该轴设定的速度参数 Speed、Accel、Decel 等无关。分轴运动速度由合轴速度参数 VpSpeed 实时关联计算。

该指令支持 6 种模式：

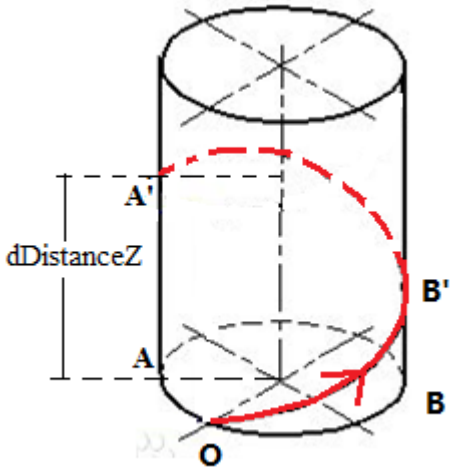
（1） ucMode=0

平面圆弧是以 A 为终点，B 为中间点。合轴是 XYZ 轴合运动，合轴的目标位置在指令开始执行时刻位置基础上，增量以平面圆弧长度、Z 轴运动位置长度为直角边的三角形斜边长度。合轴与分轴位置、速度关系如下：

$$dDistance\_AxisID^2 = dCircleLength^2 + dDistance\_AxisZ^2$$

$$\frac{dSpeed\_AxisZ}{dSpeed\_AxisID} = \frac{dDistance\_AxisZ}{dDistance\_AxisID}$$

下图为该模式运动示意图，红色曲线为三轴联动的空间轨迹曲线。

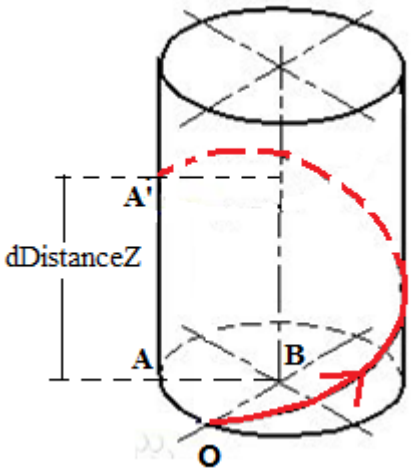


运动轨迹

（2） ucMode=1

平面圆弧以 B 为圆心，A 为终点，按照逆时针方向。合轴是 XYZ 轴合运动，合轴与分轴位置、速度关系与模式 0 一致。

下图为该模式运动示意图，红色曲线为三轴联动的空间轨迹曲线。

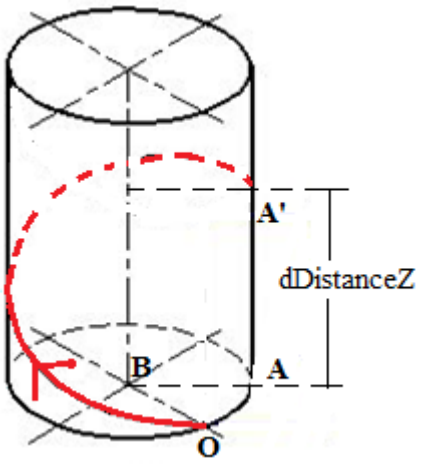


运动轨迹

（3） ucMode=3

平面圆弧以 B 为圆心，A 为终点，按照顺时针方向。合轴是 XYZ 轴合运动，合轴与分轴位置、速度关系与模式 0 一致。

下图为该模式运动示意图，红色曲线为三轴联动的空间轨迹曲线。



运动轨迹

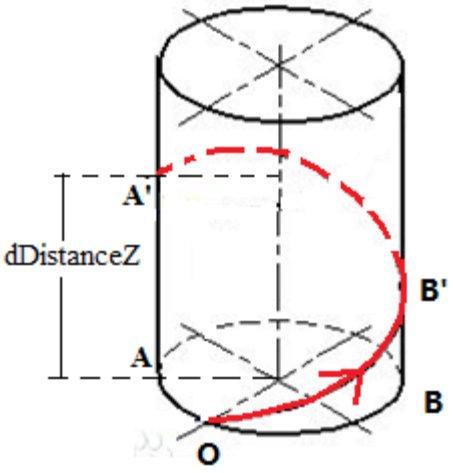
(4) ucMode=4

平面圆弧是以 A 为终点，B 为中间点。合轴是 XY 轴合运动，合轴的目标位置在指令开始执行时刻位置基础上，增量以平面圆弧长度。合轴与分轴位置、速度关系如下：

$$dDistance\_AxisID = dCircleLength$$

$$\frac{dSpeed\_AxisZ}{dSpeed\_AxisID} = \frac{dDistance\_AxisZ}{dDistance\_AxisID}$$

下图为该模式运动示意图，红色曲线为三轴联动的空间轨迹曲线。



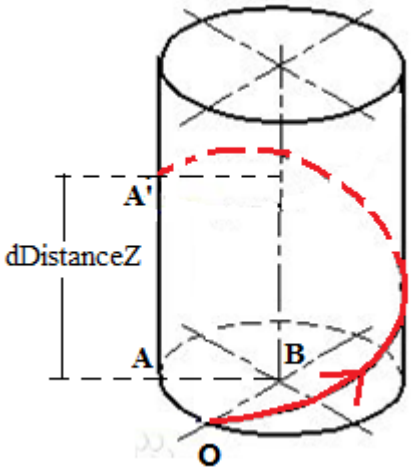
运动轨迹

(5) ucMode=5

平面圆弧以 B 为圆心，A 为重点，按照逆时针方向。合轴是 XY 轴合运动，合轴与分轴位置、速度

关系与模式 4 一致。

下图为该模式运动示意图，红色曲线为三轴联动的空间轨迹曲线。

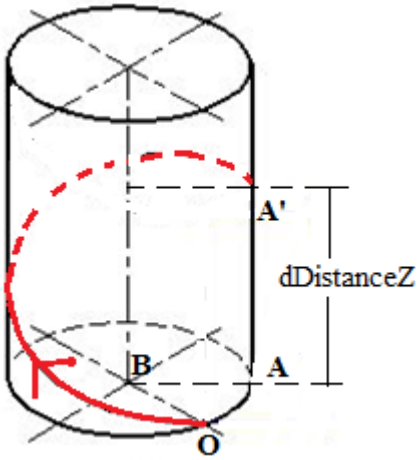


运动轨迹

(6) ucMode=7

平面圆弧以 B 为圆心，A 为重点，按照顺时针方向。合轴是 XY 轴合运动，合轴与分轴位置、速度关系与模式 4 一致。

下图为该模式运动示意图，红色曲线为三轴联动的空间轨迹曲线。



运动轨迹

5.4.3.1.2 参数

5.4.3.1.3 指令类型

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
ucAxisZ	Input	USINT	插补轴 Z 的轴号
ucMode（O 为当前点）	Input	USINT	0：A 为终点，B 为中间某点的一段圆弧；ucAxisID 的速度 dSpeed 是三轴的合运动的速度。注意：用户设置时 OBA 三点不可共线 1：A 为终点，B 为圆弧的圆心，且按照逆时针方向走；ucAxisID 的速度 dSpeed 是三轴的合运动的速度 3：A 为终点，B 为圆弧的圆心，且按照顺时针方向走；ucAxisID 的速度 dSpeed 是三轴的合运动的速度 4：A 为终点，B 为中间某点的一段圆弧；ucAxisID 的速度 dSpeed 是 X、Y 轴合运动的速度。注意：用户设置时 OBA 三点不可共线 5：A 为终点，B 为圆弧的圆心，且按照逆时针方向走；ucAxisID 的速度 dSpeed 是 X、Y 轴合运动的速度 7：A 为终点，B 为圆弧的圆心，且按照顺时针方向走；ucAxisID 的速度 dSpeed 是 X、Y 轴合运动的速度
dAx	Input	LREAL	A 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，A 的 X 坐标）
dAy	Input	LREAL	A 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，A 的 Y 坐标）
dBx	Input	LREAL	B 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，B 的 X 坐标）
dBy	Input	LREAL	B 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，B 的 Y 坐标）
dDistanceZ	Input	LREAL	Z 轴位移
HMC_MoveHelical	Output	UDINT	轴指令 ID

非阻塞指令，轴运动缓存指令。

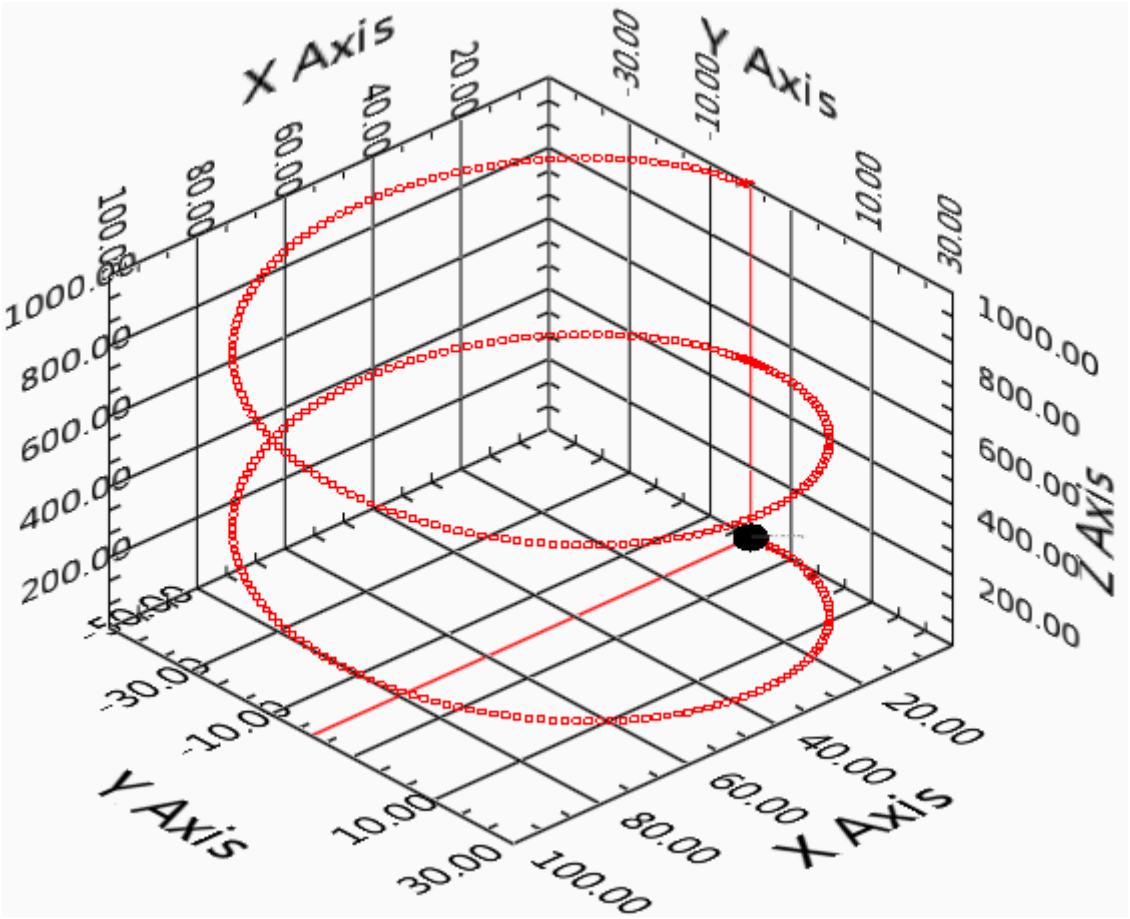
5.4.3.1.4 示例

螺纹铣削运动轨迹为一螺旋线，可通过数控机床的三轴联动来实现，工件水平面进行 X、Y 轴的圆弧插补，工件的垂直方向进行 Z 轴线性运动。在 XOY 平面完成整圆运动，即沿螺纹孔旋转 360 度后，在 Z 轴方向上升一个螺距。可利用螺旋线插补铣削螺纹。

例如，要加工一个直径为 100 mm，螺距是 500 mm，高度为 1000 mm 的螺纹。

```
FOR i:=1 TO 2 BY 1 DO
    HMC_MoveHelical (0, 1, 2, 3, 7, 0, 0, 50, 0, 500);
END_FOR
```

AT 示波器记录各分轴 Dpos，将数据导出，通过第三方软件读取数据绘制三维图形：



三维示意图

5.4.3.2 HMC_MoveHelicalEx（高级螺旋线插补）

5.4.3.2.1 功能

螺旋线插补运动。ucAxisID 为合运动轴，在坐标架 O-XYZ 下，ucAxisX、ucAxisY 两轴执行平面圆弧插补运动，ucAxisZ 轴执行 Z 相线性运动，三轴共同完成圆柱螺旋线轨迹。

此指令可设定合轴的轴参数 Speed = 函数参数 dSpeed，合轴以该速度为目标速度进行解算。

详细内容可参考 HMC_MoveHelical（螺旋线插补）中功能说明部分。

5.4.3.2.2 参数

5.4.3.2.3 指令类型

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号
ucAxisX	Input	USINT	插补轴 X 的轴号
ucAxisY	Input	USINT	插补轴 Y 的轴号
ucAxisZ	Input	USINT	插补轴 Z 的轴号
ucMode（O 为当前点）	Input	USINT	0：A 为终点，B 为中间某点的一段圆弧；ucAxisID 的速度 dSpeed 是三轴的合运动的速度。注意：用户设置时 OBA 三点不可共线 1：A 为终点，B 为圆弧的圆心，且按照逆时针方向走；ucAxisID 的速度 dSpeed 是三轴的合运动的速度 3：A 为终点，B 为圆弧的圆心，且按照顺时针方向走；ucAxisID 的速度 dSpeed 是三轴的合运动的速度 4：A 为终点，B 为中间某点的一段圆弧；ucAxisID 的速度 dSpeed 是 X、Y 轴合运动的速度。注意：用户设置时 OBA 三点不可共线 5：A 为终点，B 为圆弧的圆心，且按照逆时针方向走；ucAxisID 的速度 dSpeed 是 X、Y 轴合运动的速度 7：A 为终点，B 为圆弧的圆心，且按照顺时针方向走；ucAxisID 的速度 dSpeed 是 X、Y 轴合运动的速度
dAx	Input	LREAL	A 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，A 的 X 坐标）
dAy	Input	LREAL	A 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，A 的 Y 坐标）
dBx	Input	LREAL	B 点在 X 轴位置-O 点在 X 轴的位置 （将 O 视为坐标系原点时，B 的 X 坐标）
dBy	Input	LREAL	B 点在 Y 轴位置-O 点在 Y 轴的位置 （将 O 视为坐标系原点时，B 的 Y 坐标）
dDistanceZ	Input	LREAL	Z 轴位移
dSpeed	Input	LREAL	合轴 ucAxisID 的目标速度
HMC_MoveHelical	Output	UDINT	轴指令 ID

非阻塞指令，轴运动缓存指令。

5.4.3.2.4 示例

参考 HMC_MoveHelical（螺旋线插补）示例。

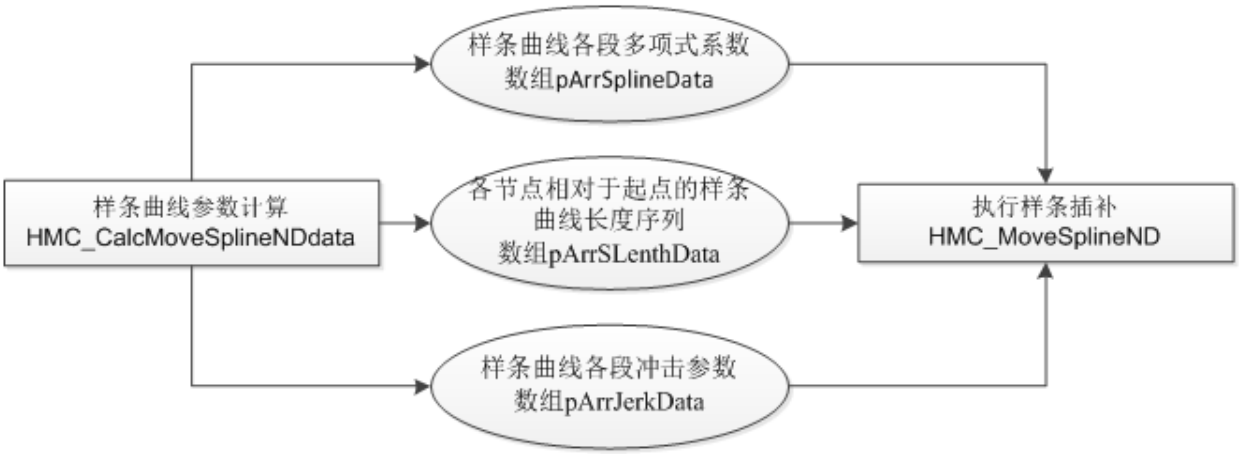
5.4.4 样条插补

5.4.4.1 HMC_MoveSplineND（样条插补）

5.4.4.1.1 功能

以当前点为起点，执行 N 轴样条插补运动（相对坐标/增量坐标）。

该指令使用 HMC_CalcMoveSplineNDdata 函数计算所得到的样条曲线的相关参数(pArrSplineData、pArrSplineData、pArrSlenthData)，通过指定合轴轴号 ucAxisID 及各个插补分轴轴号数组 pAxis、样条曲线节点坐标数组 pArrPos，实现 N 轴样条插补功能。



样条插补功能

使用 N 轴样条插补指令最常用的是实现 2 维或 3 维样条插补，可如下设置参数：

（1） 2 轴样条插补：跟随轴 ucFollowAxisNum = 0，端点轴 ucEndPointAxisNum=2；

（2） 3 轴样条插补：跟随轴 ucFollowAxisNum = 0，端点轴 ucEndPointAxisNum=3。

执行上述 2 轴或 3 轴样条插补时，插补合轴按照端点轴合运动的轨迹运动，端点轴速度的合速度近似等于插补合轴速度 VpSpeed。通过设定合轴 Speed 参数，可控制插补轮廓速度。

pAxis 数组中放置端点轴（EndPointAxis）与跟随轴（FollowAxis）的轴号，排列顺序为先端点轴后跟随轴。

pArrPos 数组中的坐标由各节点在端点轴与跟随轴上的坐标分量组成，端点轴与跟随轴的数目由 ucEndPointAxisNum 与 ucFollowAxisNum 确定。数组中坐标排列顺序：首先是各端点轴坐标，然后是各跟随轴坐标。可定义该数组为二维数组，这样数组元素 pArrPos[i,j]便表示第 i 个轴在第 j 个节点上的坐标分量。

端点轴与跟随轴的区别（如果仅使用常规的 2 轴/3 轴样条插补，则跟随轴数目为 0，此段可不作了解）：类似于 N 轴直线插补（HMC_MoveND）的线性轴与独立轴，作样条插补运动时，插补合轴按照各端点轴合运动的轨迹进行，而不受跟随轴的影响。各端点轴速度的合速度约等于插补合轴速度 VpSpeed，各跟随轴的平均速度约为 （跟随轴总位移/合轴总位移）*合轴 VpSpeed，瞬时速度与样条曲线在该点的瞬时斜率有关。端点轴的个数只能为 2 或 3，而跟随轴个数可为 0，总轴数不能超过系统最大轴数（64）。

插补合轴运动速度由对应轴参数 Speed 设定，加减速分别由轴参 Accel、Decel 设定。在插补运动过程中，可以实时修改合轴 Speed、Accel、Decel 等轴参数，立即生效。插补运动分轴的运行速度、加减速度与该轴自身的 Speed、Accel、Decel 等无关, 由合轴实时关联计算得到。

5.4.4.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号
pAxis	Input	POINTER TO USINT	插补分运动轴号数组首地址。 排列顺序为先端点轴后跟随轴。
ucEndPointAxisNum	Input	USINT	端点轴的个数。端点轴合运动的速度近似为插补合轴 Speed。 取值范围：2 或 3
ucFollowAxisNum	Input	USINT	跟随轴的个数。
pArrPos	Input	POINTER TO LREAL	样条曲线节点坐标，2 维数组 pArrPos[length1][length2]首地址。数组第一维长度 length1 为坐标轴个数（维度），第二维长度 length2 为样条曲线节点数目。有如下关系： length1 = n length2 = uiSplineNum 其中 n = ucEndPointAxisNum +ucFollowAxisNum
uiSplineNum	Input	UDINT	pArrPos 中的样点个数，1~1000 个。
pArrSplineData	Input	POINTER TO LREAL	数组首地址，HMC_CalcMoveSplineNDdata 中计算得到的样条曲线各段的

			多项式系数参数。
pArrSlenthData	Input	POINTER TO LREAL	数组首地址，HMC_CalcMoveSplineNDdata 中计算得到的 各节点相对于起点的样条曲线长度序列。
pArrJerkData	Input	POINTER TO LREAL	数组首地址，HMC_CalcMoveSplineNDdata 中计算得到的 样条曲线各段冲击参数。
HMC_MoveSplineND	Output	UDINT	轴指令 ID

5.4.4.1.3 指令类型

轴运动缓存指令。

5.4.4.1.4 补充说明

- pArrJerkData 为 HMC_CalcMoveSplineNDdata 中计算得出的每段样条函数所对应的最大冲击计算参数，插补运动过程中会根据 pArrJerkData 与合轴速度实时速度 VpSpeed 计算实时冲击数据，如果大于合轴轴参中的 Jerk 限定值，则将降低合轴速度，以保证运动过程中冲击不超限；
- 在转折较为急剧的节点附近，实际插补速度可能会比设定的合轴插补速度 Speed 略大一些，但上述中的处理措施可保证此处的冲击不会超过用户设定的 Jerk 值。

5.4.4.1.5 示例

示例 1：2 轴样条插补

参见 HMC_CalcMoveSplineNDdata 中示例 1。

示例 2：3 轴样条插补

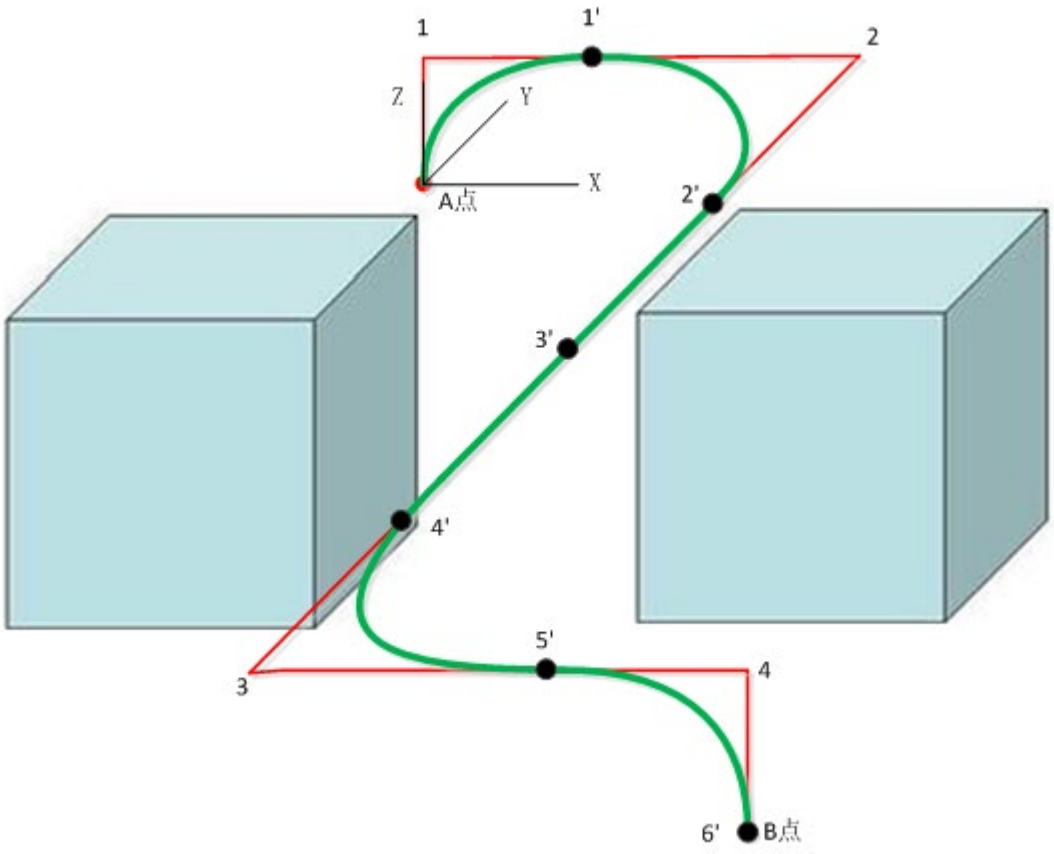
参见 HMC_CalcMoveSplineNDdata 中示例 2。

示例 3：避障取放物料

三轴直角坐标机械手需把物料从位置 A 搬运至位置 B，然后从 B 返回 A，周而复始。由于 AB 之间有一些障碍物，因此需规划合适的路径避开障碍。

如果采用常规的三轴直线插补，则路径规划如下图中红色曲线所示。由图可知在每个转折点 1、2、3、4 均需减速至 0，然后开始下一段运动，大大降低了搬运效率。

若使用样条插补，以 A 点为当前起点运动至 B 点，在路径上取节点 1'、2'、3'、4'、5'、6'，则样条插补轨迹示意图如下。由于样条插补轨迹的方向变化是平滑的，不会像直线插补那样在转折点处减速至 0，因此可以大大提高搬运效率及运动的平稳性。



样条插补示意图

设 A 点为坐标原点，B 点坐标为（3000,-4000,0）。

若采用三轴直线插补，则从 A 点运行至 B 点经过的停止点 1、2、3、4 到 B 点，坐标依次为（0,0,500）、（1500,0,500）、（1500,-4000,500）、（3000,-4000,500）、（3000,-4000,0）。

从 A 点运行至 B 点的三轴直线插补程序如下：

(****首先定义大小为 pArrPos[3][5]的节点数组，然后对目标位置数组赋值****)

pArrPos[0][0] := 0;　　(*第 1 个目标点在轴 X 轴上的坐标分量*)

pArrPos[0][1] := 1500;　　(*第 2 个目标点在轴 4(X 轴)上的坐标分量*)

pArrPos[0][2] := 1500;　　(*第 3 个目标点在轴 4(X 轴)上的坐标分量*)

```
pArrPos[0][3] := 3000;     (*第 4 个目标点在轴 4(X 轴)上的坐标分量*)
pArrPos[0][4] := 3000;     (*第 5 个目标点在轴 4(X 轴)上的坐标分量*)

pArrPos[1][0] := 0;     (*第 1 个目标点在轴 5(Y 轴)上的坐标分量*)
pArrPos[1][1] := 0;     (*第 2 个目标点在轴 5(Y 轴)上的坐标分量*)
pArrPos[1][2] := -4000;  (*第 3 个目标点在轴 5(Y 轴)上的坐标分量*)
pArrPos[1][3] := -4000;  (*第 4 个目标点在轴 5(Y 轴)上的坐标分量*)
pArrPos[1][4] := -4000;  (*第 5 个目标点在轴 5(Y 轴)上的坐标分量*)

pArrPos[2][0] := 500;      (*第 1 个目标点在轴 6(Z 轴)上的坐标分量*)
pArrPos[2][1] := 500;      (*第 2 个目标点在轴 6(Z 轴)上的坐标分量*)
pArrPos[2][2] := 500;      (*第 3 个目标点在轴 6(Z 轴)上的坐标分量*)
pArrPos[2][3] := 500;      (*第 4 个目标点在轴 6(Z 轴)上的坐标分量*)
pArrPos[2][4] := 0;     (*第 5 个目标点在轴 6(Z 轴)上的坐标分量*)

(*定义分轴轴号数组 pArrAxis[3]*)
pArrAxis[0] := 4;       (*X 轴轴号*)
pArrAxis[1] := 5;       (*Y 轴轴号*)
pArrAxis[2] := 6;       (*Z 轴轴号*)
 (*定义合轴轴号 AxisNo_3d *)
AxisNo_3d := 7;
(*执行三轴直线插补指令*)
HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][0], pArrPos[1][0],
pArrPos[2][0]);
HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][1], pArrPos[1][1],
```

pArrPos[2][1]);

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][2], pArrPos[1][2], pArrPos[2][2]);

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][3], pArrPos[1][3], pArrPos[2][3]);

HMC_MoveABS3d(AxisNo_3d,pArrAxis[0],pArrAxis[1],pArrAxis[2], pArrPos[0][4], pArrPos[1][4], pArrPos[2][4]);

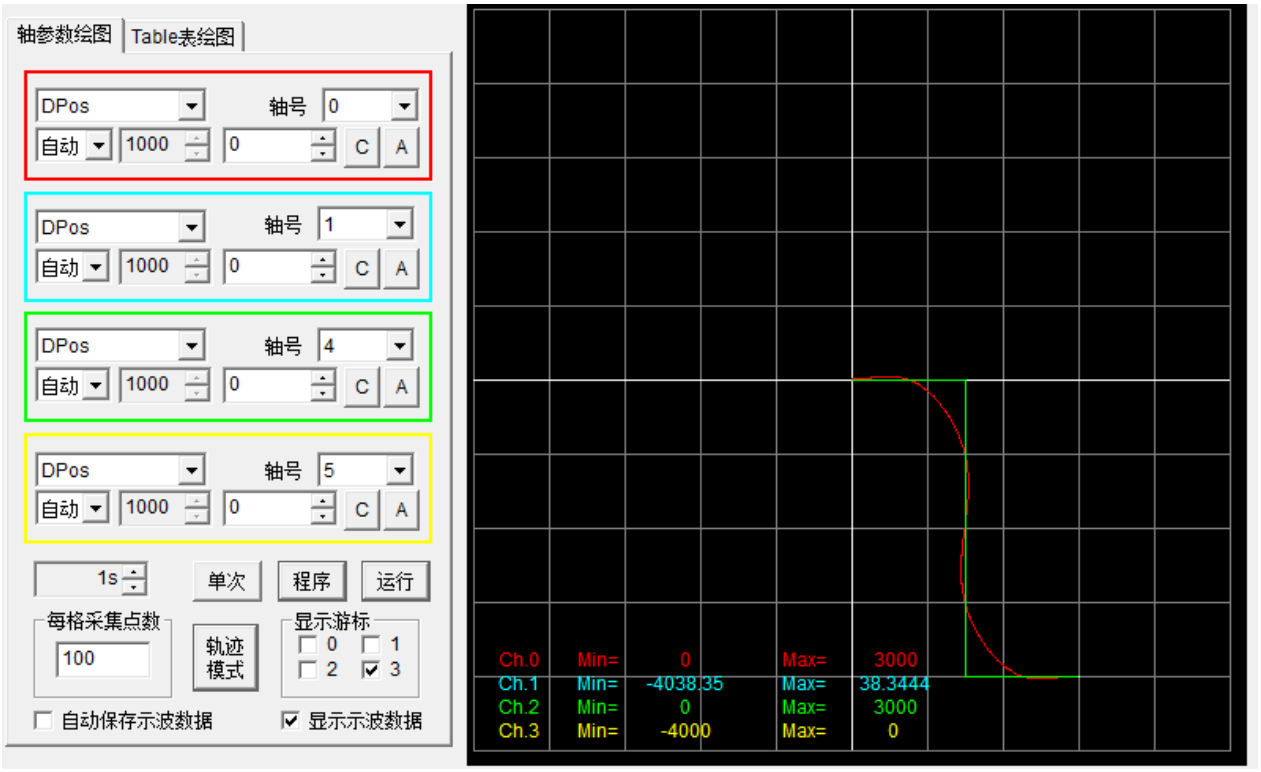
若采用样条插补，从 A 点运行至 B 点，在路径上取节点 1'、2'、3'、4'、5'、6'，存入节点位置数组 pArrPos 中。1'、2'、3'、4'、5'、6'的坐标依次为（750,0,500）、（1500,-1000,500）、（1500,-2000,500）、（1500,-3000,500）、（2250,-4000,500）、（3000,-4000,0）。

从 A 点运行至 B 点的样条插补程序如下：

```
(****首先定义大小为 pArrPos[3][6]的节点数组，然后对节点数组赋值****)

pArrPos[0][0] := 750;      (*第 1 个目标点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][1] := 1500;   (*第 2 个节点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][2] := 1500;   (*第 3 个节点点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][3] := 1500;   (*第 4 个节点点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][4] := 2250;   (*第 5 个节点点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][5] := 3000;   (*第 6 个节点点在轴 0(X 轴)上的坐标分量*)


pArrPos[1][0] := 0;      (*第 1 个节点点在轴 1(Y 轴)上的坐标分量*)

pArrPos[1][1] := -1000;  (*第 2 个节点点在轴 1(Y 轴)上的坐标分量*)

pArrPos[1][2] := -2000;  (*第 3 个节点点在轴 1(Y 轴)上的坐标分量*)

pArrPos[1][3] := -3000;  (*第 4 个节点点在轴 1(Y 轴)上的坐标分量*)

pArrPos[1][4] := -4000;  (*第 5 个节点点在轴 1(Y 轴)上的坐标分量*)

pArrPos[1][5] := -4000;  (*第 6 个节点点在轴 1(Y 轴)上的坐标分量*)
```

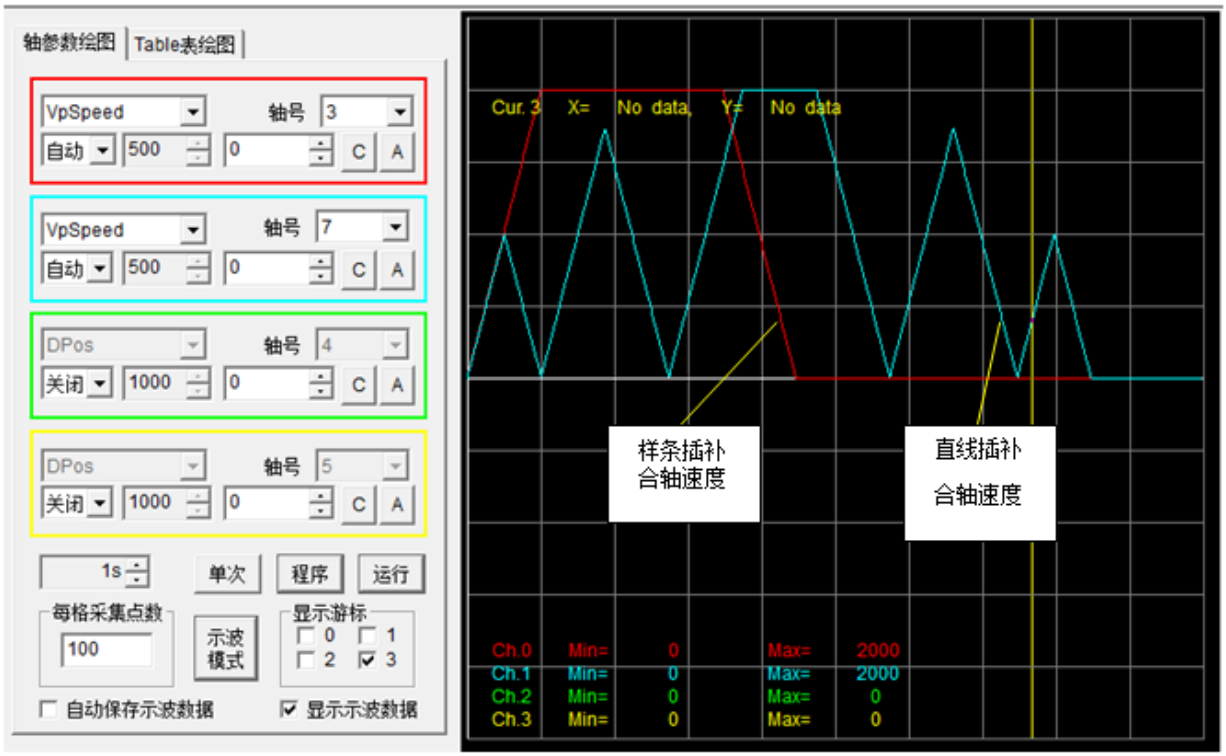
```
pArrPos[2][0] := 500;      (*第 1 个节点点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][1] := 500;      (*第 2 个节点点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][2] := 500;      (*第 3 个节点点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][3] := 500;      (*第 4 个节点点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][4] := 500;      (*第 5 个节点点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][5] := 0;     (*第 6 个节点点在轴 2(Z 轴)上的坐标分量*)

(*定义数组 pArrSplineData[72], pArrSlenthData[6], pArrJerkData [6]，
保存计算得到的样条参数*)
(*调用函数生成样条曲线参数*)
HMC_CalcMoveSplineNDdata(pArrPos,6,3,0, pArrSplineData,72, pArrSlenthData, pArrJerkData);

(*定义分轴轴号数组 pArrAxis[3]*)
pArrAxis[0] := 0;      (*第 1 个端点轴轴号(X 轴)*)
pArrAxis[1] := 1;      (*第 2 个端点轴轴号(Y 轴)*)
pArrAxis[2] := 2;      (*第 3 个端点轴轴号(Z 轴)*)

(*定义合轴轴号 AxisNo_Spline *)
AxisNo_Spline := 3;

(*执行样条插补指令*)
HMC_MoveSplineND(AxisNo_Spline,pArrAxis,3,0, pArrPos,6, pArrSplineData, pArrSlenthData,
pArrJerkData);
```

在 AT 中运行，直线插补轨迹（绿色）与样条插补轨迹（红色）的 XY 平面俯视图如下：



运动轨迹

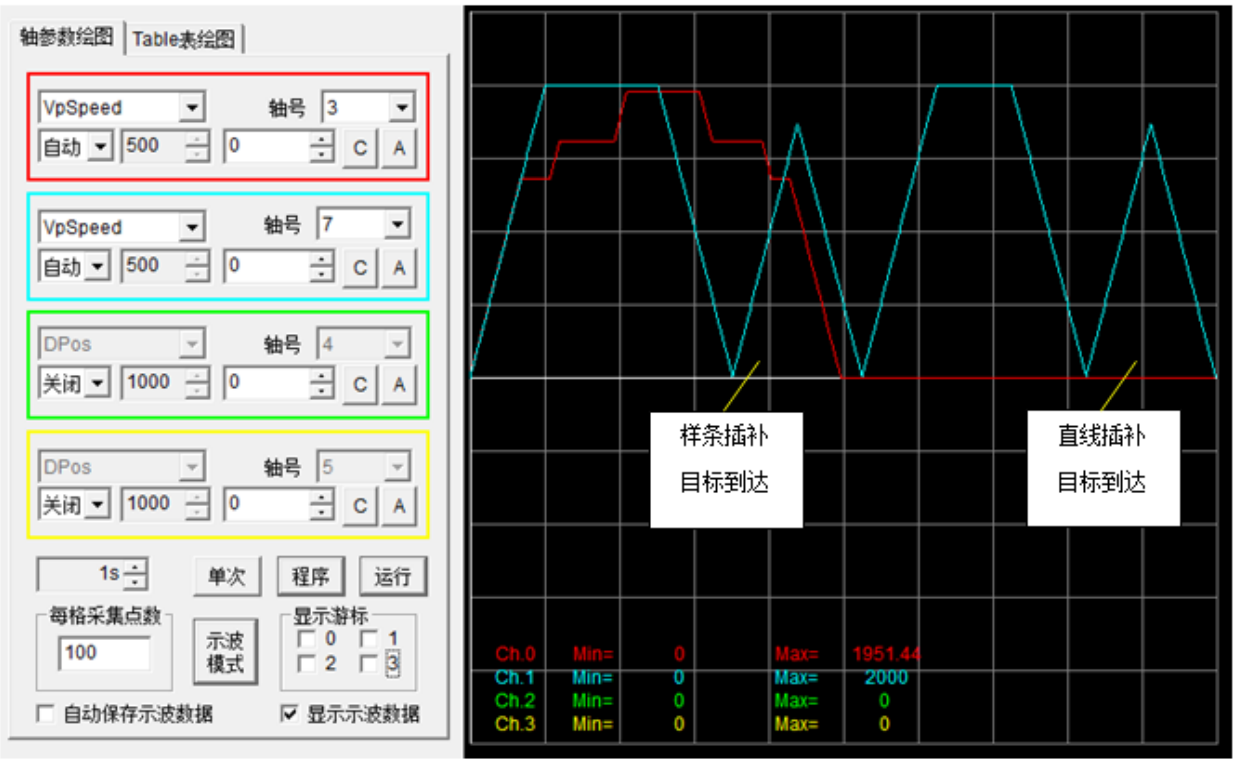
直线插补与样条插补采用相同的加减速及最大速度参数，在 AT 中运行，运行过程中直线插补速度曲线（青色）与样条插补速度曲线（红色）如下图所示。可以看出，直线在每个转折点 1、2、3、4 均需减速至 0，然后开始下一段运动，大大降低了搬运效率。样条插补轨迹整个运动过程比较平滑，不会像直线插补那样在转折点处减速至 0，因此可以大大提高效率以及运动的平稳性。



运动轨迹

（红色：样条插补合轴速度，青色：直线插补合轴速度）

为限制样条插补在转折较为急剧之处的冲击，设置样条插补中合轴轴参 Jerk 限制值为 5000，AT 中运行速度曲线如图所示（直线插补速度曲线（青色）与样条插补速度曲线（红色））。由图可知，在转折较为急剧之处，通过降低该段的最大速度，保证运行中的冲击不超过设定值 5000。



运动轨迹

（红色：样条插补合轴速度，青色：直线插补合轴速度）

若要从 B 点运行至 A 点，则只需把样条节点 1'、2'、3'、4'、5'、6'顺序倒置，存入 pArrayPos 数组中，即可生成回程的样条参数模型。这里不再详述。

5.4.4.2 HMC_MoveSplineNDEx（高级样条插补）

5.4.4.2.1 功能

以当前点为起点，执行 N 轴样条插补运动（相对坐标/增量坐标）。

可参见 HMC_MoveSplineND（样条插补）指令，该指令与 HMC_MoveSplineND 的区别在于：执行该指令时会把插补合轴轴参 Speed 修改为函数参数中的 dSpeed 值，以该值为插补合速度进行样条插补运动。

5.4.4.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	插补合运动轴号
pAxis	Input	POINTER TO USINT	插补分运动轴号数组首地址。 排列顺序为先端点轴后跟随轴。
ucEndPointAxisNum	Input	USINT	端点轴的个数。端点轴合运动的速度近似为插补合轴 Speed。 取值范围：2 或 3
ucFollowAxisNum	Input	USINT	跟随轴的个数。
pArrPos	Input	POINTER TO LREAL	样条曲线节点坐标，2 维数组。第一维长度 length1 为坐标轴个数（维度），第二维长度 length2 为样条曲线节点数目。有如下关系： length1 = n length2 = uiSplineNum 其中 n = ucEndPointAxisNum +ucFollowAxisNum
uiSplineNum	Input	UDINT	pArrPos 中的样点个数，1~1000 个。
pArrSplineData	Input	POINTER TO LREAL	数组首地址，HMC_CalcMoveSplineNDdata 中计算得到的样条曲线各段的多项式系数参数。
pArrSlenthData	Input	POINTER TO LREAL	数组首地址，HMC_CalcMoveSplineNDdata 中计算得到的 各节点相对于起点的样条曲线长度序列。
pArrJerkData	Input	POINTER TO LREAL	数组首地址，HMC_CalcMoveSplineNDdata 中计算得到的 样条曲线各段冲击参数。
dSpeed	Input	LREAL	合轴目标速度（插补轮廓速度）
HMC_MoveSplineNDEx	Output	UDINT	轴指令 ID

5.4.4.2.3 指令类型

轴运动缓存指令。

5.4.4.2.4 示例

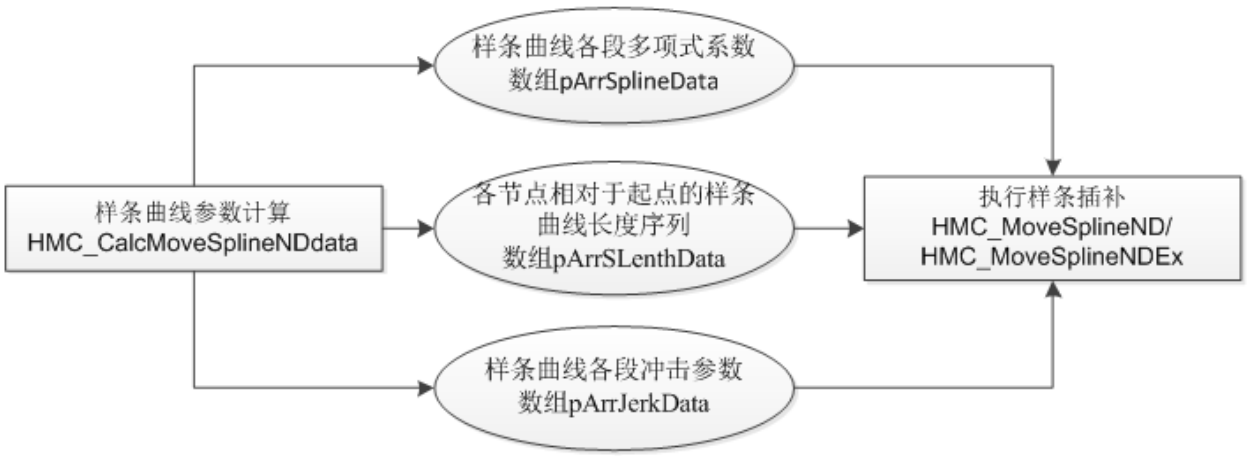
参见 HMC_MoveSplineND（样条插补）。

5.4.4.3 HMC_CalcMoveSplineNDdata（生成样条数据）

5.4.4.3.1 功能

样条曲线由于具有二阶连续可导性，可有效的减小插补运动过程中因方向变化而引入的速度/加速度跃变，从而降低机械冲击，实现平滑的运动效果。

该函数生成 N 轴插补的三次样条曲线，计算得到的样条曲线参数被保存在用户指定的数组中，用户不必关心该数组中的具体内容，后续使用样条插补指令 HMC_MoveSplineND / HMC_MoveSplineNDEx 调用该数组中的数据，即可实现 N 轴样条插补功能。



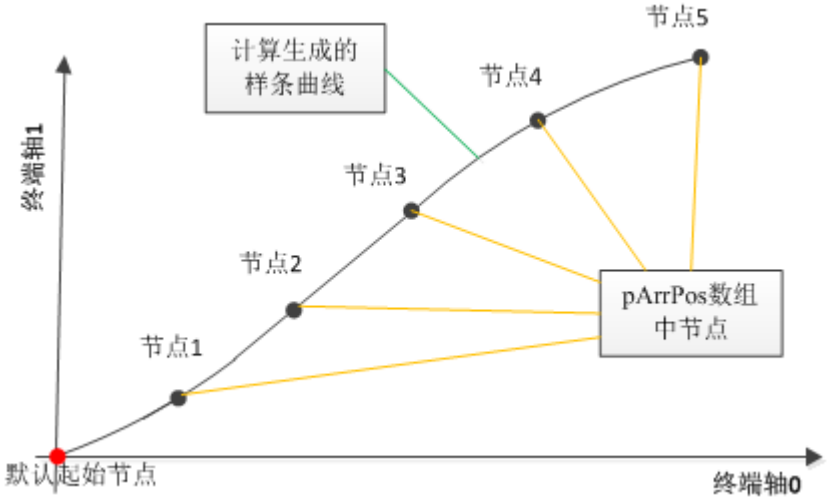
样条插补功能

函数根据用户指定的样条曲线节点坐标 pArrPos（起点默认为原点）、节点个数 uiSplineNum，计算生成样条曲线的参数。参数包括各段的多项式系数、各节点相对于起点的样条曲线长度序列、样条曲线各段冲击参数，分别保存在用户指定的数组 pArrSplineData、pArrSLenthData、pArrJerkData 中。

pArrPos 数组中的坐标由各节点坐标在端点轴（EndPointAxis）与跟随轴（FollowAxis）上的分量组成，端点轴与跟随轴的数目由 ucEndPointAxisNum 与 ucFollowAxisNum 确定。数组中坐标排列顺序：首先是各端点轴坐标，然后是跟随轴坐标。建议定义该数组为二维数组，这样数组元素 pArrPos[i,j]便表示第 j 个节点在 i 号轴上的坐标分量。详见示例。

使用 N 轴样条插补功能最常用的是实现 2 维或 3 维样条插补，可如下设置参数：

（1） 2 轴样条插补： 跟随轴 ucFollowAxisNum = 0，端点轴 ucEndPointAxisNum=2。



2 轴样条插补

（2） 3 轴样条插补： 跟随轴 ucFollowAxisNum = 0，端点轴 ucEndPointAxisNum=3。

使用样条插补指令 HMC_MoveSplineND / HMC_MoveSplineNDEx 作插补运动时，插补合轴按照各端点轴合运动的轨迹进行，而不受跟随轴的影响。各端点轴速度的合速度约等于插补合轴速度 Speed。

端点轴与跟随轴的概念可类比 N 轴直线插补（HMC_MoveND）的线性轴与独立轴（如果仅使用常规的 2 轴/3 轴样条插补，可不作了解），详见 HMC_MoveSplineND（样条插补）。

函数首先计算出样条曲线长度序列 pArrSLenthData，然后使用该长度序列计算各个分轴（端点轴、跟随轴）的样条曲线模型参数。样条长度序列 pArrSLenthData 由节点坐标在端点轴上的分量计算得到，表示样条插补指令 HMC_MoveSplineND / HMC_MoveSplineNDEx 中合轴在各节点处的位移，合轴速度 speed 近似等于各端点轴合运动的速度。

5.4.4.3.2 参数

参数	声明	数据类型	说明
pArrPos	Input	POINTER TO LREAL	样条曲线节点坐标，2 维数组 pArrPos[length1][length2]首地址。数组第一维长度 length1 为坐标轴个数（维度），第二维长度 length2 为样条曲线节点数目。有如下关系： length1 = n length2 = uiSplineNum 其中 n = ucEndPointAxisNum +ucFollowAxisNum
uiSplineNum	Input	UDINT	pArrPos 中的样点个数，1~1000 个。
ucEndPointAxisNum	Input	USINT	端点轴的个数，样条曲线长度序列 pArrSlenthData 由节点坐标在

			端点轴上的分量计算得到。 端点轴合运动的速度近似为插补合轴 Speed。 取值范围：2 或 3
ucFollowAxisNum	Input	USINT	跟随轴的个数。跟随轴的样条曲线模型是根据样条曲线长度序列 pArrSlenthData 和样条节点在跟随轴上的坐标分量计算得到的。
pArrSplineData	Input	POINTER TO LREAL	用户指定的一维数组 pArrSplineData[length]首地址，用于保存计算得到的样条曲线各段的多项式系数参数数据。应满足： 数组长度 length >= n*uiSplineNum*4 其中 n = ucEndPointAxisNum +ucFollowAxisNum
uiSplineDataSize	Input	UDINT	pArrSplineData 数组长度。
pArrSlenthData	Input	POINTER TO LREAL	用户指定的一维数组 pArrSlenthData[length]首地址，用于保存计算得到的 各节点相对于起点的样条曲线长度序列。应满足： 数组总长度 >= uiSplineNum
pArrJerkData	Input	POINTER TO LREAL	用户指定的数组首地址，用于保存计算得到的 样条曲线各段冲击参数。应满足： 数组长度 length >= uiSplineNum
HMC_CalcMoveSplineNDdata	Output	UDINT	轴指令 ID

5.4.4.3.3 指令类型

非轴运动缓存指令。

5.4.4.3.4 补充说明

- 各节点之间的空间距离尽量不要相差过大，否则在相差较大的节点附近的数值计算偏差可能会影响精度。
- 当样条曲线上各个节点在经过原点的直线上时，生成的样条曲线为直线。
- 如果只有一个节点，则生成的样条曲线为连接原点与该节点的线段。因此 N 轴样条插补可完成 2 轴、3 轴、N 轴直线插补的功能。

5.4.4.3.5 示例

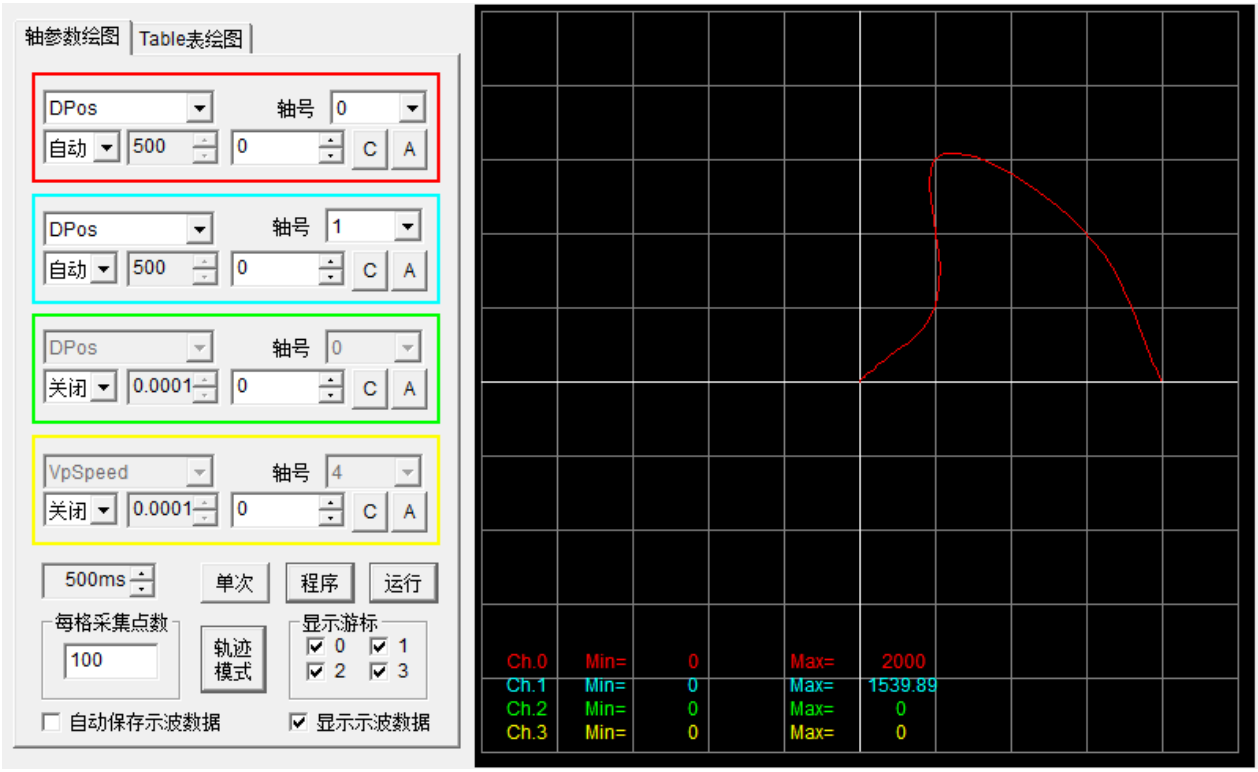
示例 1：2 轴样条插补

端点轴数量为 2 个，跟随轴数量为 0 个。2 个端点轴轴号为 0、1，分别对应 X、Y 轴。节点数目为 3

个，节点坐标分别为（500,500）、（500,1500）、（1500,1000）、（2000,0）。使用 HMC_CalcMoveSplineNDdata 生成样条曲线参数，然后使用该参数调用 HMC_MoveSplineND 执行样条插补指令(当前点为原点)，生成样条曲线轨迹。程序如下：

```
(****首先定义大小为 pArrPos[2][3]的节点数组，然后对节点数组赋值****)
pArrPos[0][0] := 500;      (*第 1 个节点在轴 0(X 轴)上的坐标分量*)
pArrPos[0][1] := 500;      (*第 2 个节点在轴 0(X 轴)上的坐标分量*)
pArrPos[0][2] := 1500;   (*第 3 个节点在轴 0(X 轴)上的坐标分量*)
pArrPos[0][3] := 2000;   (*第 4 个节点在轴 0(X 轴)上的坐标分量*)

pArrPos[1][0] := 500;      (*第 1 个节点在轴 1(Y 轴)上的坐标分量*)
pArrPos[1][1] := 1500;   (*第 2 个节点在轴 1(Y 轴)上的坐标分量*)
pArrPos[1][2] := 1000;   (*第 3 个节点在轴 1(Y 轴)上的坐标分量*)
pArrPos[1][3] := 0;      (*第 4 个节点在轴 1(Y 轴)上的坐标分量*)

(*定义数组 pArrSplineData[32], pArrSlenthData[4], pArrJerkData [4]，保存计算得到的样条参数*)
(*调用函数生成样条曲线参数*)
HMC_CalcMoveSplineNDdata(pArrPos, 4, 2, 0,  pArrSplineData, 32, pArrSlenthData, pArrJerkData);

(*定义轴号数组 pArrAxis[2]*)
pArrAxis[0] := 0;    (*第 1 个端点轴轴号*)
pArrAxis[1] := 1;    (*第 2 个端点轴轴号*)

(*执行样条插补指令*)
HMC_MoveSplineND(2,pArrAxis,2,0, pArrPos,4,
pArrSplineData, pArrSlenthData, pArrJerkData);
```

AT 示波器运行结果如下：



运动轨迹

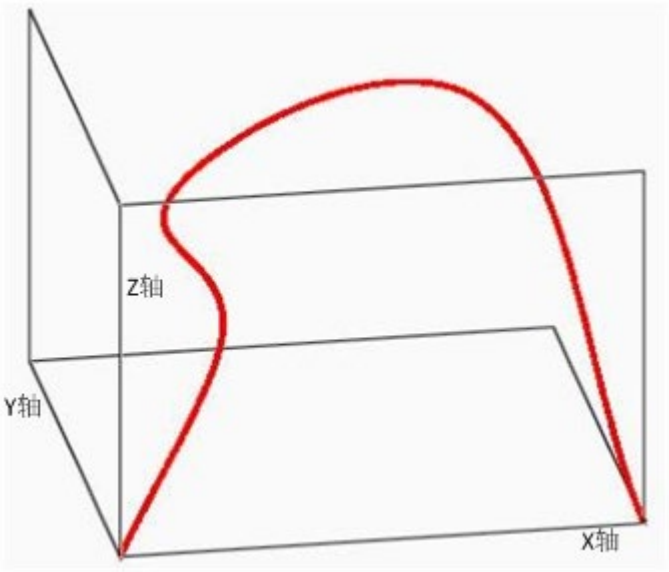
示例 2：3 轴样条插补曲线

端点轴数量为 3 个，跟随轴数量为 0 个。3 个端点轴轴号为 0、1、2，分别对应 X、Y 轴。节点数目为 4 个，节点坐标分别为（500,500,500）、（500,1500,500）、（1500,1000,1000）、（2000,0,0）。使用 HMC_CalcMoveSplineNDdata 生成样条曲线参数，然后使用该参数调用 HMC_MoveSplineND 执行样条插补指令(当前点为原点)，生成样条曲线轨迹。程序如下：

```
(****首先定义大小为 pArrPos[3][3]的节点数组，然后对节点数组赋值****)

pArrPos[0][0] := 500;      (*第 1 个节点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][1] := 500;      (*第 2 个节点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][2] := 1500;   (*第 3 个节点在轴 0(X 轴)上的坐标分量*)

pArrPos[0][3] := 2000;   (*第 4 个节点在轴 0(X 轴)上的坐标分量*)


pArrPos[1][0] := 500;      (*第 1 个节点在轴 1(Y 轴)上的坐标分量*)
```

```
pArrPos[1][1] := 1500;  (*第 2 个节点在轴 1(Y 轴)上的坐标分量*)
pArrPos[1][2] := 1000;  (*第 3 个节点在轴 1(Y 轴)上的坐标分量*)
pArrPos[1][3] := 0;     (*第 4 个节点在轴 1(Y 轴)上的坐标分量*)

pArrPos[2][0] := 500;     (*第 1 个节点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][1] := 500;     (*第 2 个节点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][2] := 1000;  (*第 3 个节点在轴 2(Z 轴)上的坐标分量*)
pArrPos[2][3] := 0;     (*第 4 个节点在轴 2(Z 轴)上的坐标分量*)

(*定义数组 pArrSplineData[48], pArrSlenthData[4], pArrJerkData [4]，保存计算得到的样条参数*)
(*调用函数生成样条曲线参数*)
HMC_CalcMoveSplineNDdata(pArrPos,4,3,0, pArrSplineData,48, pArrSlenthData, pArrJerkData);

(*定义轴号数组 pArrAxis[3]*)
pArrAxis[0] := 0;   (*第 1 个端点轴轴号*)
pArrAxis[1] := 1;   (*第 2 个端点轴轴号*)
pArrAxis[2] := 2;   (*第 3 个端点轴轴号*)

(*执行样条插补指令*)
HMC_MoveSplineND(3,pArrAxis,3,0, pArrPos,4, pArrSplineData, pArrSlenthData, pArrJerkData);
```

运行结果如下：



三维轨迹

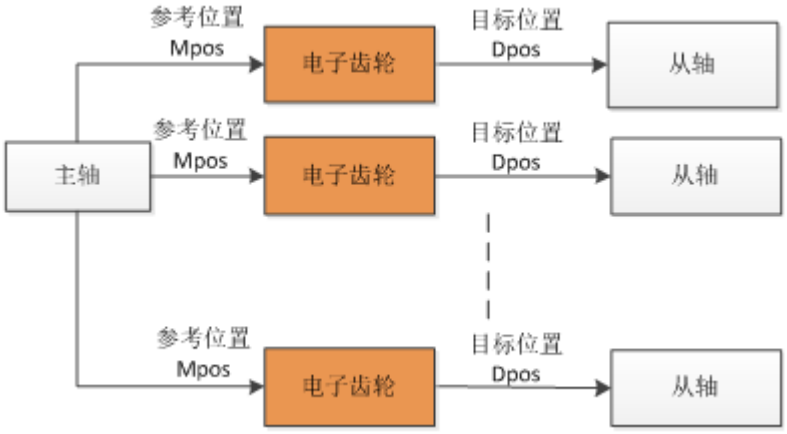
5.4.5 电子齿轮

5.4.5.1 HMC_Gear（电子齿轮）

5.4.5.1.1 功能说明

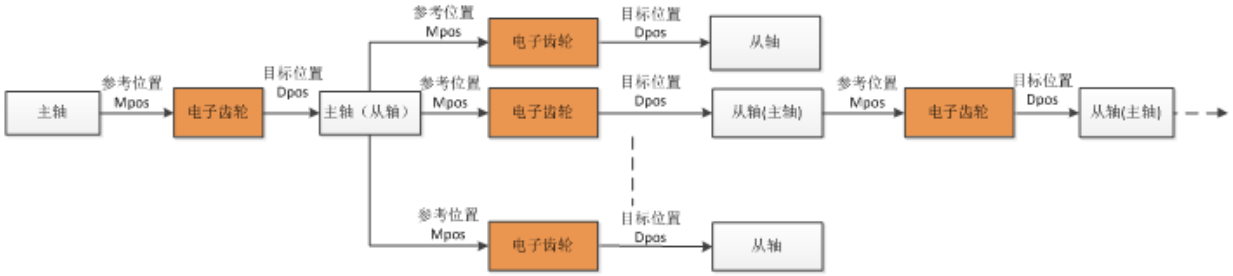
电子齿轮主要用于实现无轴传动。无轴传动技术具有传动精度高、结构简单、传动比范围宽、调整方便等优点，因此可以代替传统的机械传动实现准确的传动关系。在印染、纺织、造纸、齿轮加工机床内联传动等要求实现多轴同步传动的领域，无轴传动技术有着广阔的应用前景。

串联式电子齿轮结构图



并联式电子齿轮结构图

电子齿轮使用中有串联式与并联式两种基本结构，如上图所示。串联式结构中从轴作为下一级电子齿轮的主轴，并联式结构中各从轴拥有共同的主轴。实际使用中可根据需要选用，或使用该两种基本结构组合构建复合式电子齿轮结构。



复合式电子齿轮结构图

使用电子齿轮后，从轴 Dpos 增量与主轴 Mpos 增量满足如下关系：

$$\Delta Dpos_S = \Delta Mpos_M * \left(\frac{iRatioNumerator}{iRatioDenominator}\right)$$

5.4.5.1.2 参数

参数	声明	数据类型	说明
ucSlaveAxis	Input	USINT	从轴轴号
ucMasterAxis	Input	USINT	主轴轴号
iRatioNumerator	Input	DINT	齿轮比分子
iRatioDenominator	Input	DINT	齿轮比分母（不可为 0）

HMC_Gear	Output	UDINT	轴指令 ID：成功 0xFFFFFFFF：函数参数错误

5.4.5.1.3 指令类型

轴运动缓存指令。

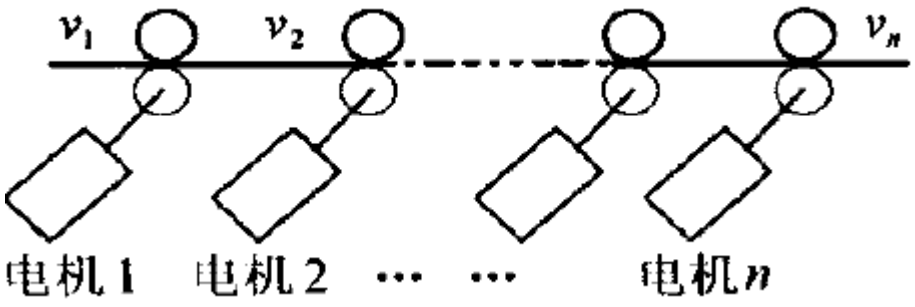
5.4.5.1.4 补充说明

- 在该指令执行过程中，可以通过调用 HMC_SetGearRatio 函数动态修改齿轮比例；
- 从数学关系上看，HMC_Gear 和 HMC_Superimpose 均是 HMC_SuperimposeEx 的特例；
- HMC_Gear 和 HMC_Superimpose 配合使用可实现 HMC_SuperimposeEx 的所有功能。

5.4.5.1.5 示例

示例 1：连续工艺过程装备无轴控制技术

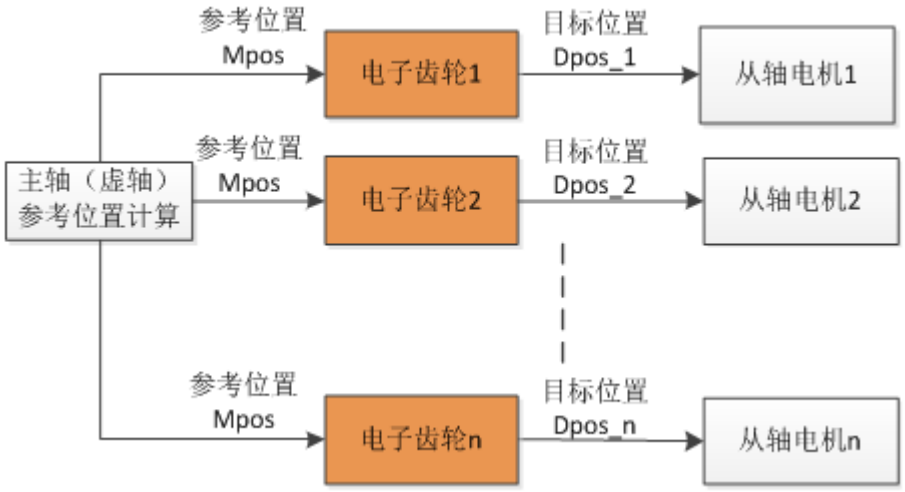
许多连续工艺过程的工业设备，如热连轧机和冷连轧机、造纸机、聚合物加工生产线、纺织染整机械等等，对传动系统的要求是相似的，即采用多单元分布传动。单元数量往往多至几十个，各单元由单独电动机驱动; 从工艺过程和保持 2 个单元间给定张力的条件出发，各单元机之间的速度关系必须保持严格不变,必须在某一精度下长时间地稳定所加工带材(金属、纸张、聚合胶片、纺织物等) 的线速度;各单元机通过同步传动，保持加工时的适度张力，避免加工过程中带材伸长或产生折皱。



连续工艺过程装备传动示意图

这类设备的传动结构如上图所示，通常采用并联式电子齿轮箱结构,通过共同加工的带材负载以及通过公用的速度给定和各个分部速度比值控制，使分部系统相互耦合起来。

主轴为虚轴，用于产生公共参考位置，通过对该公共参考位置进行电子齿轮变换产生各从轴目标位移，驱动各从轴按预定位置及速度关系同步运转。



连续工艺过程装备传动电子齿轮实现方式

如果某个分部内部还存在传动关系，则可在分部中构建子电子齿轮控制系统，与整个系统组成复合式电子齿轮控制系统。

示例 2：无轴传动实现螺纹车削

数控车床主轴电机为 0 号轴，轴向进给电机为 1 号轴，进给丝杠导程为l_f，现欲车削导程为l_x的螺纹，可用电子齿轮功能实现。

计算可得，加工该导程的螺纹时进给电机与主轴的传动比为：

$$\frac{l_x}{l_f}$$

把该数值化简为最简分数形式，设：

$$\frac{l_x}{l_f}=\frac{\text{iRatioNumerator}}{\text{iRatioDenominator}}$$

程序如下（ST 语言）（设主轴电机、轴向进给电机、径向进给电机分别为轴 0、1、2）：

```
(**设置轴类型为步进轴**)

HMC_SetAxisType(Axis0.AxisID，10);

HMC_SetAxisType(Axis1.AxisID，10);
```

HMC_SetAxisType(Axis2.AxisID，10);

(**设置各轴速度、加减速参数（略）**)

(**设置各轴速度、加减速参数（略）**)

(**运动至进刀点（略）**)

(**径向进给电机给定螺纹深度（略）**)

(*投送 HMC_Gear 指令，1 号轴（轴向进给电机）为从轴，0 号轴（主轴电机）为主轴，齿轮比为 iRatioNumerator/iRatioDenominator *)

HMC_Gear(Axis1.AxisID，Axis0.AxisID, iRatioNumerator, iRatioDenominator);

(*驱动主轴运转，轴向进给电机将在电子齿轮的驱动下同步运动，完成螺纹车削*)

示例 3：H 形同步齿形带并联机构运动控制（参见 HMC_Superimpose 中例 1）。

5.4.5.2 HMC_SetGearRatio（设置电子齿轮比例）

5.4.5.2.1 功能

设置齿轮比例，在 HMC_Gear 指令启动后，可实时调用该指令动态改变运转中电子齿轮比例，从而达到动态改变电子齿轮的效果。

该指令只对当前正在运行的 HMC_Gear 指令起作用。

5.4.5.2.2 参数

参数	声明	数据类型	说明

ucSlaveAxis	Input	USINT	从轴轴号
iRatioNumerator	Input	DINT	齿轮比分子
iRatioDenominator	Input	DINT	齿轮比分母
HMC_SetGearRatio	Output	UDINT	0：成功 其他值：错误码

5.4.5.2.3 指令类型

非轴运动缓存指令。

5.4.5.3 HMC_GetGearMaster（获取电子齿轮主轴）

5.4.5.3.1 功能

获取电子齿轮的主轴号。该指令只可获取当前正在运行的 HMC_Gear 指令主轴号。

5.4.5.3.2 输入参数

参数	声明	数据类型	说明
ucSlaveAxis	Input	USINT	从轴轴号
HMC_GetGearMaster	Output	INT	电子齿轮主轴号：成功 -1：当前未配置电子齿轮或函数参数错误

5.4.5.3.3 指令类型

非轴运动缓存指令。

5.4.5.4 HMC_GetGearRatio（获取电子齿轮比例）

5.4.5.4.1 功能说明

获取电子齿轮的齿轮比。该指令只可获取当前正在运行的 HMC_Gear 指令齿轮比。

5.4.5.4.2 参数

参数	声明	数据类型	说明
ucSlaveAxis	Input	USINT	从轴轴号
HMC_GetGearRatio	Output	LREAL	电子齿轮比例

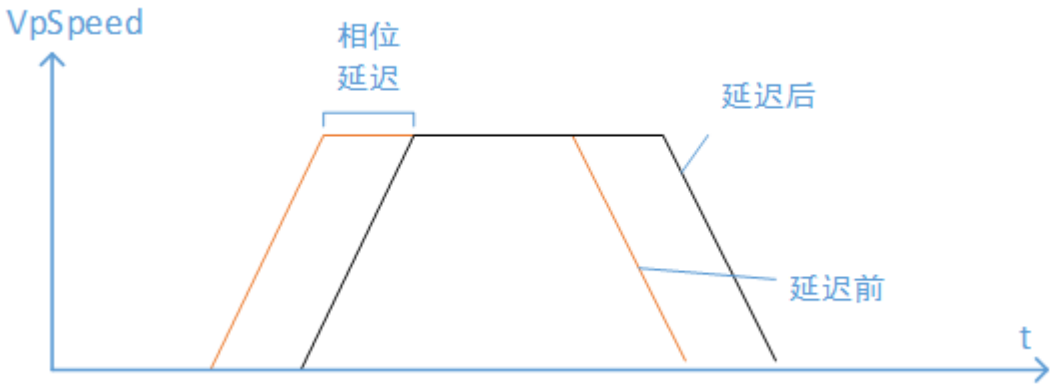
5.4.5.4.3 指令类型

非轴运动缓存指令。

5.4.5.5 HMC_SetGearPhaseOffset（设置电子齿轮相位延迟）

5.4.5.5.1 功能

该指令可调整电子齿轮从轴的跟随相位，即把原来的从轴轨迹在时间轴上作一个相位延迟。调整后的从轴轨迹比原轨迹滞后 iPhaseOffset 个伺服周期（最大为 128）。如图所示：



从轴相位延迟轨迹

可通过电子齿轮间接实现其它指令的相位延迟。如凸轮指令，可把凸轮主轴作为 1:1 电子齿轮的从轴，通过调整电子齿轮的相位延迟量间接实现凸轮的相位延迟。

5.4.5.5.2 参数

参数	声明	数据类	说明

		型	
ucAxisID	Input	USINT	从轴轴号
iPhaseOffset	Input	DINT	相位延迟量，0~128，单位为伺服周期
HMC_SetGearPhaseOffset	Output	UDINT	成功：0 错误：错误码

5.4.5.5.3 指令类型

非轴运动缓存指令。

5.4.5.6 HMC_GetGearPhaseOffset（获取电子齿轮相位延迟）

5.4.5.6.1 功能

获取电子齿轮相位延迟设定量。

5.4.5.6.2 输入参数

参数	声明	类型	参数描述
ucAxisID	Input	USINT	从轴轴号，0~63
HMC_GetGearPhaseOffset	Output	DINT	相位延迟量，单位为伺服周期

5.4.5.6.3 指令类型

非轴运动缓存指令。

5.4.5.7 HMC_SetGearTransationSwitch（电子齿轮过渡段开关）

5.4.5.7.1 功能

开启/关闭电子齿轮平滑功能。开关开启时，电子齿轮在启动阶段或电子齿轮比发生切换时，会自动插入一段平滑过渡过程，过渡过程采用轴自身的加减速，防止机械冲击。

5.4.5.7.2 输入参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucValue	Input	BOOL	0：关闭 1：开启
HMC_SetGearTransationSwitch	Output	UDINT	0：执行成功 0xFFFFFFFF：执行失败

5.4.5.7.3 指令类型

非轴运动缓存指令。

5.4.6 电子凸轮

5.4.6.1 HMC_Cam（电子凸轮）

5.4.6.1.1 功能

通过选取目标运动轨迹（二维）上的离散坐标点，使用电子凸轮功可以通过主从轴联动的方式逼近目标运动轨迹。逼近方式为“以折代曲”，即用连接上述离散坐标点的折线段去逼近目标曲线。

电子凸轮可完成与机械式凸轮相似的功能，而没有机械式凸轮设计难度大、加工成本高、运动高副易磨损等缺点，在许多场合是机械式凸轮的理想替代品。电子凸轮功能也可用来进行不规则曲线的逼近。与 HMC_ClcMoveLinkData 或 HMC_ClcFlexLinkData 指令配合使用，还可实现追剪、飞剪、旋切等应用场景所需的动作流程。

电子凸轮有主轴、从轴之分，凸轮主从轴位置预先存储在主从轴位置数组 pArrPos 中。pArrPos 为二维数组，第一维 pArrPos[0]为主轴位置，第二维 pArrPos[1]为从轴位置，二者为一一对应关系。凸轮启动时，根据不同的启动方式设定凸轮主轴起点位置；凸轮运行过程中，主轴运行正常的运动控制指令，控制器根据主轴当前 Mpos 值和 pArrPos 中数据计算出从轴的理论位置，驱动从轴进行相应的联动动作，同时根据当前主轴 Mpos 值计算判断是否满足凸轮撤消条件。

按主轴位置的边界范围划分，凸轮可分为：

（1） 单次凸轮——凸轮启动后，若凸轮主轴位置在有效行程范围内，则凸轮可保持正常联动运行（主轴单向、往复运动均可。当主轴在有效行程范围内往复运动时，从轴也会进行往复性的联动动作），主轴超过有效行程范围时凸轮指令将自动撤销。

（2） 循环凸轮—— 凸轮主轴位置没有边界范围限制，启动后如果不执行相应的撤销指令，凸轮功能将一直有效。

按凸轮启动条件的触发方式划分，凸轮可分为如下几种：

（1） 立即启动—— 凸轮指令投放后立即启动凸轮功能，启动位置为主轴当前 Mpos 位置；

（2） 固定位置启动（到位启动）—— 凸轮指令投放后，当凸轮主轴 Mpos 从负向跨过凸轮启动位置时，凸轮功能启动，启动位置为该固定位置；

（3） 事件启动—— 凸轮指令投放后，当设定的某高速捕获通道被触发时启动凸轮，启动位置为高速捕获通道所捕获到的主轴 Mpos 位置；

（4） DI 启动—— 凸轮指令投放后，当设定的某 DI 信号为 1 时启动凸轮，启动位置为程序检测到 DI 为 1 时主轴 Mpos 位置。

事件启动凸轮与 DI 启动凸轮启动方式类似，均为外部信号触发。不同的是事件启动凸轮可利用高速捕获通道记录下 DI 信号被触发瞬间的精确位置，因而可获得更为精准的启动位置，相比 DI 启动凸轮具有更高的精度。

凸轮主轴的一个行程长度 Delta 为凸轮主轴位置数组的最后一个元素值减去第一个元素值，即：Delta = pArrPos[0,uiPosNum-1] - pArrPos[0,0]，其中 uiPosNum 为位置个数。设凸轮起点为 Mpos0，则单次凸轮的有效行程范围为：[Mpos0，Mpos0+ Delta)，单次凸轮主轴位置超出该行程范围时，凸轮将撤消；凸轮主轴在有效行程范围内时，主轴可进行单向、往复运动，当主轴在有效行程范围内往复运动时，从轴也会进行往复性的联动动作。

对于循环凸轮，当凸轮主轴位置越过一个凸轮主轴行程长度 Delta 时，将会进入下一个凸轮行程，此时从轴将会以上个凸轮行程结束时的从轴位置为起点，在此基础上作增量位移计算，重复上一个凸轮行程的联动动作，周而复始。凸轮主轴也可做单向、往复运动。如果需要撤消循环凸轮，可执行 Hmc_Cancel 立即撤消，或执行 HMC_CancelREPCam 指令在凸轮主轴行程的整数倍处撤消，详细使用方法请参考相关指

令。

5.4.6.1.2 参数

参数	声明	数据类型	说明
ucSlaveAxis	Input	USINT	从轴轴号
ucMasterAxis	Input	USINT	主轴轴号
pArrPos	Input	POINTER TO LREAL	位置数组首地址（二维）。 在二维数组 pArrPos[2,uiPosNum]中，pArrPos[0]为主轴位置数组，pArrPos[1]为从轴位置数组。
uiPosNum	Input	UDINT	位置个数，应大于等于 3
dScale	Input	LREAL	凸轮从轴位置数据比例因子。凸轮从轴实际位置输出 = 从轴位置数组设定输出*dScale
ucMode	Input	USINT	0：单次立即启动； 1：单次固定位置启动； 2：单次事件启动； 3：单次 DI 启动； 4：循环立即启动； 5：循环固定位置启动； 6：循环事件启动； 7：循环 DI 启动。
dTrigMes	Input	LREAL	启动信息： 1、凸轮为位置启动时，该值表示主轴的启动位置； 2、凸轮为事件触发启动时，表示某高速捕获通道号； 3、凸轮为 DI 为 1 触发启动时，表示 DI 通道（0~63）。

5.4.6.1.3 指令类型

轴运动缓存指令。

5.4.6.1.4 补充说明

- 主轴位置数组 pArrPos[0]中的元素必须是单调递增的（不可有相同数），数据正负均可。凸轮运行过程中不能修改主从轴位置数据，否则可能会造成系统错误！
- 凸轮为事件触发启动时，dTrigMes 表示某高速捕获通道，所以应事先配置一个高速捕获通道来捕获启动（HMC_CaptureConfig 功能块中的 CaptureAxis 必须为凸轮主轴号）。

- 凸轮启动时的参考启动点、运行时用于计算的动态坐标点、撤消时的参考坐标点均以凸轮主轴的 Mpos 值为参考基准。当主轴为位置闭环控制时，由于主轴 Mpos 来自于外部检测装置，为避免在凸轮有效行程范围的边界处由于 Mpos 的小扰动所造成的凸轮意外撤消，撤消条件也会兼顾主轴 Dpos 值。
- 对于单次凸轮，当 Mpos 来自外部检测装置时，为避免小扰动所引起的凸轮异常撤消，应确保凸轮正常工作区间距离凸轮边界位置有一定余量（保证 Mpos 的扰动不会超出凸轮边界位置）。

5.4.6.1.5 示例

（1） 单次立即启动凸轮(ucMode = 0)

定义 ArrPos 为主从轴位置数组，示例程序如下（ST 语言）。通过选取直线上三个坐标点，该程序实现凸轮主轴与从轴联动运行的轨迹为倾角为 45 度的直线。

凸轮指令投放运行后，以当前主轴 Mpos 值为启动位置立即启动凸轮功能，向凸轮主轴投放 Hmc_Move 指令驱动凸轮主轴运动，此时凸轮从轴作相应的联动动作，当主轴在凸轮有效行程范围内往复运行时，从轴也作往复性的联动动作（主从轴起始位置均为 0）。

```
(**主从轴位置数组赋值**)
ArrPos[0,0] := 0;
ArrPos[1,0] := 0;       (*点 1 坐标*)
ArrPos[0,1] := 500;
ArrPos[1,1] := 500;     (*点 2 坐标*)
ArrPos[0,2] := 1000;
ArrPos[1,2] := 1000;    (*点 3 坐标*)

pArrPos := ADR(ArrPos);  (*取主从轴位置数组 ArrPos 首地址*)

HMC_Cam(Axis1.AxisID,Axis0.AxisID,pArrPos,3,1,0,0);    (*投放单次立即启动凸轮指令，0 轴为主轴，1 轴为从轴，凸轮主轴行程范围为*)
```

(*向主轴投送 Move 指令，驱动凸轮主轴往复运动。此时凸轮从轴将进行相应联动动作*)

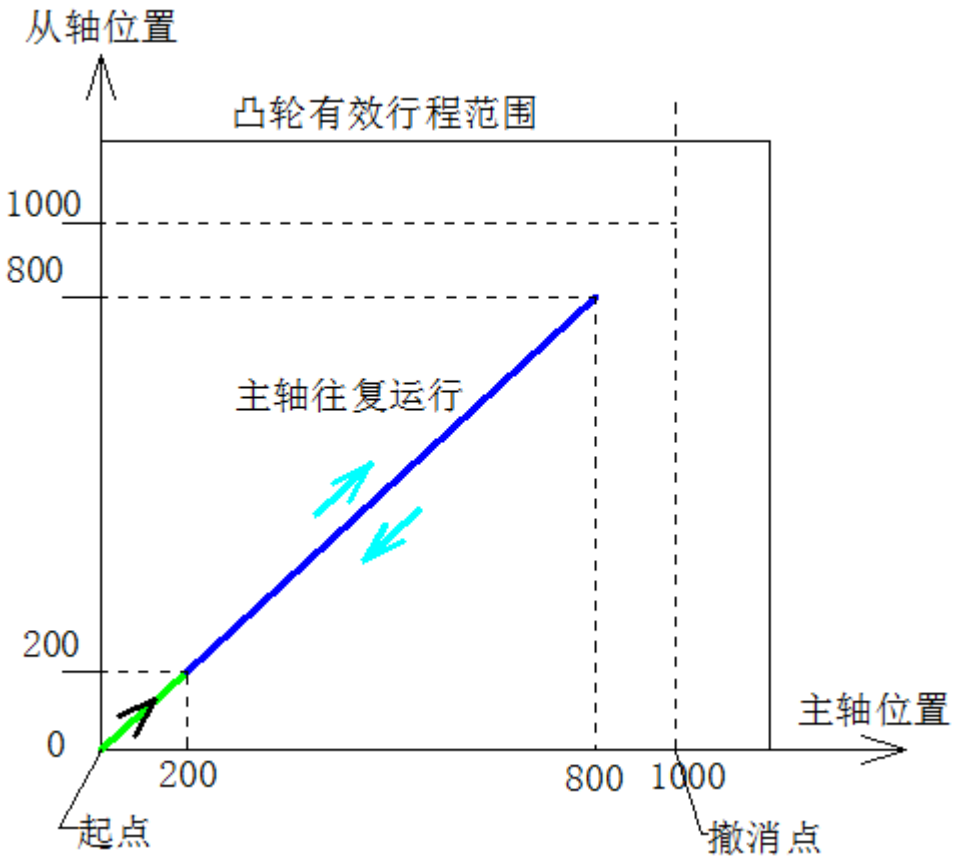
HMC_Move(Axis0.AxisID,800);

HMC_Move(Axis0.AxisID,-600);

HMC_Move(Axis0.AxisID,600);

HMC_Move(Axis0.AxisID,-600);

HMC_Move(Axis0.AxisID,600);



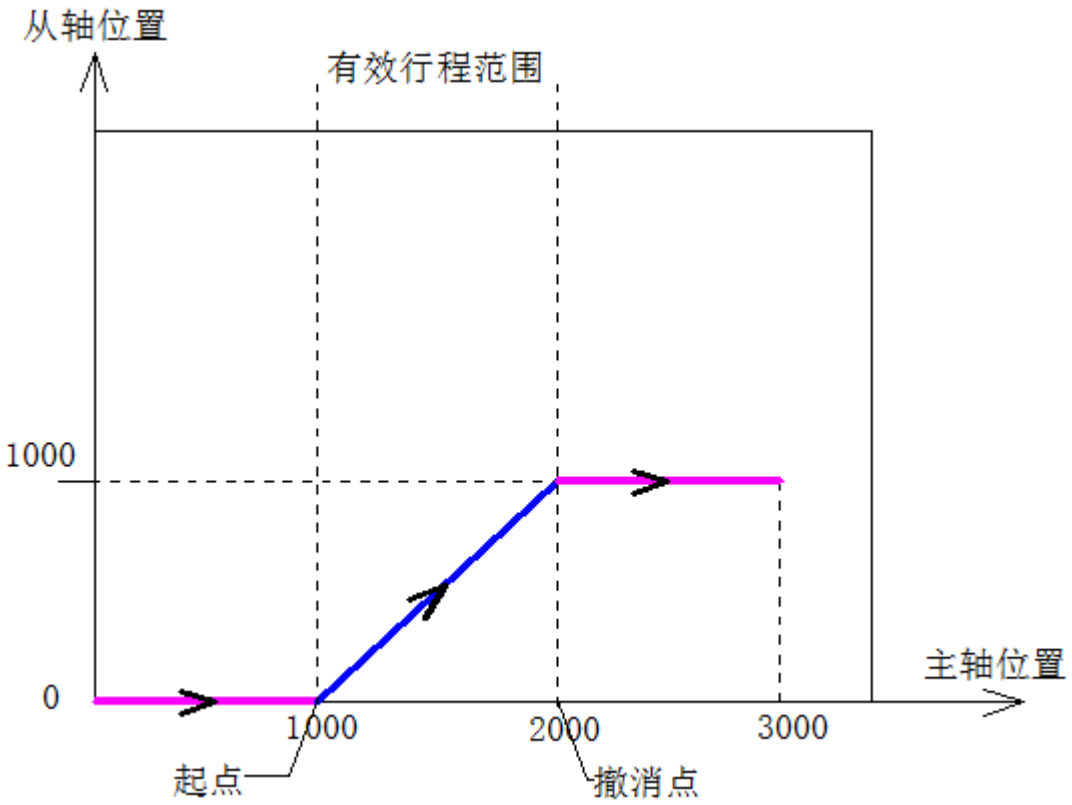
单次立即启动凸轮联动轨迹

(2) 单次固定位置启动凸轮(ucMode = 1)

示例程序如下（ST 语言），该程序凸轮主轴与从轴联动运行的轨迹为倾角为 45 度的直线。向凸轮主轴投放 Hmc_Move 指令驱动凸轮主轴运动，当凸轮主轴达到启动位置时，凸轮指令启动运行，凸轮从轴开始作相应的联动动作。当主轴越过凸轮有效行程边界时，凸轮功能自动撤销（主从轴起始位置均为 0）。

(**主从轴位置数组赋值**)

```
ArrPos[0,0] := 0;
ArrPos[1,0] := 0;       (*点 1 坐标*)
ArrPos[0,1] := 500;
ArrPos[1,1] := 500;     (*点 2 坐标*)
ArrPos[0,2] := 1000;
ArrPos[1,2] := 1000;    (*点 3 坐标*)

(*投放单次到位启动凸轮指令，0 轴为主轴，1 轴为从轴，启动位置为 1000*)
HMC_Cam (Axis1.AxisID,Axis0.AxisID,ADR(ArrPos2),101,1,1,1000);

(*向主轴投送 Move 指令，驱动凸轮主轴单向运动。单次凸轮将启动——运行——撤销*)
HMC_Move(Axis0.AxisID,3000);
```

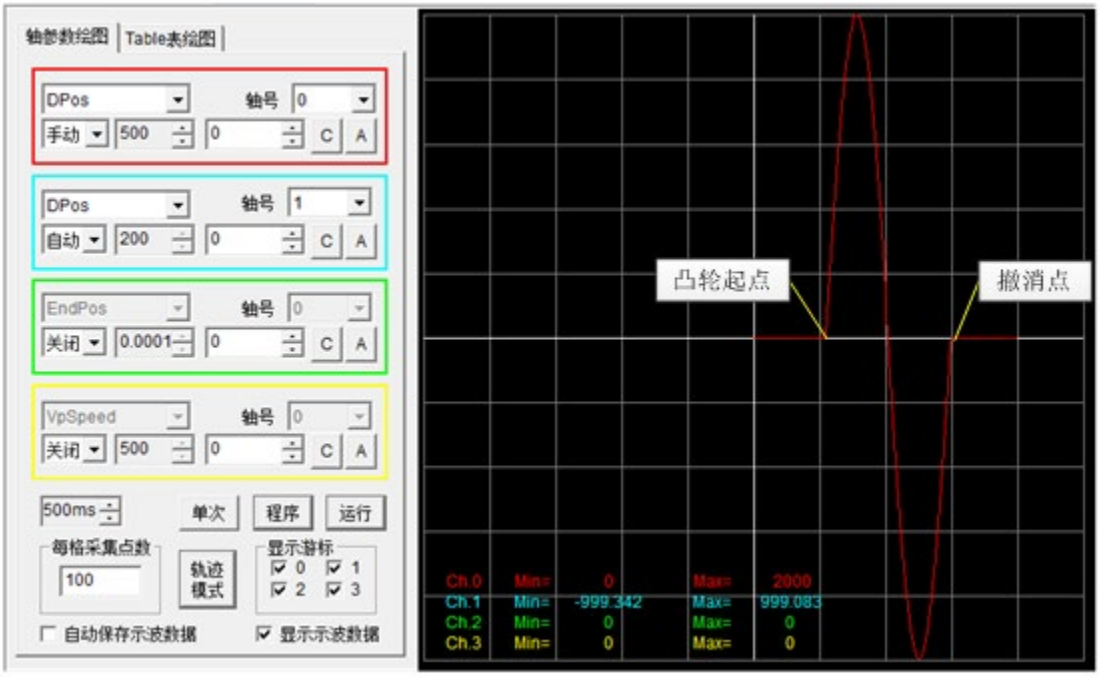


单次固定位置启动凸轮联动轨迹

（3） 单次事件启动凸轮(ucMode = 2)

示例程序如下（ST 语言），该程序实现凸轮主轴与从轴联动运行的轨迹为正弦曲线。向凸轮主轴投放 Hmc_Move 指令驱动凸轮主轴运动，当 0 号通道 DI 上升沿到来时，0 号高速捕获通道被触发，凸轮指令以捕获到的主轴位置为启动位置启动运行，凸轮从轴开始作相应的联动动作。当主轴越过凸轮有效行程边界时，凸轮功能自动撤销。

```
(**生成主从轴位置数组**)
delta := 3.1415926*2/100;
FOR n := 0 TO 100 DO
ArrPos2[0,n] := (delta*n)*150;
ArrPos2[1,n] := SIN(delta*n)*1000;
END_FOR

(*配置高速捕获通道：0 号通道 DI 触发，上升沿触发，待捕获轴为 0 号轴*)
HMC_CaptureConfig (0，0，1，0，Axis0.AxisID，0，0，0);

(*投放单次事件启动凸轮指令，0 轴为主轴，1 轴为从轴，高速捕获通道号为 0*)
HMC_Cam (Axis1.AxisID,Axis0.AxisID,ADR(ArrPos2),101,1,2,0);

(*向主轴投送 Move 指令，驱动凸轮主轴单向运动。*)
HMC_Move(Axis0.AxisID,2000);
```



单次事件启动凸轮联动轨迹

（4） 单次 DI 启动凸轮(ucMode = 3)

示例程序如下（ST 语言），该程序实现凸轮主轴与从轴联动运行的轨迹为正弦曲线。当 0 号通道 DI 上升沿到来时，凸轮指令以当前主轴 Mpos 位置为启动位置启动运行。此时若向凸轮主轴投放 Hmc_Move 指令驱动凸轮主轴运动，凸轮从轴便会开始作相应的联动动作。

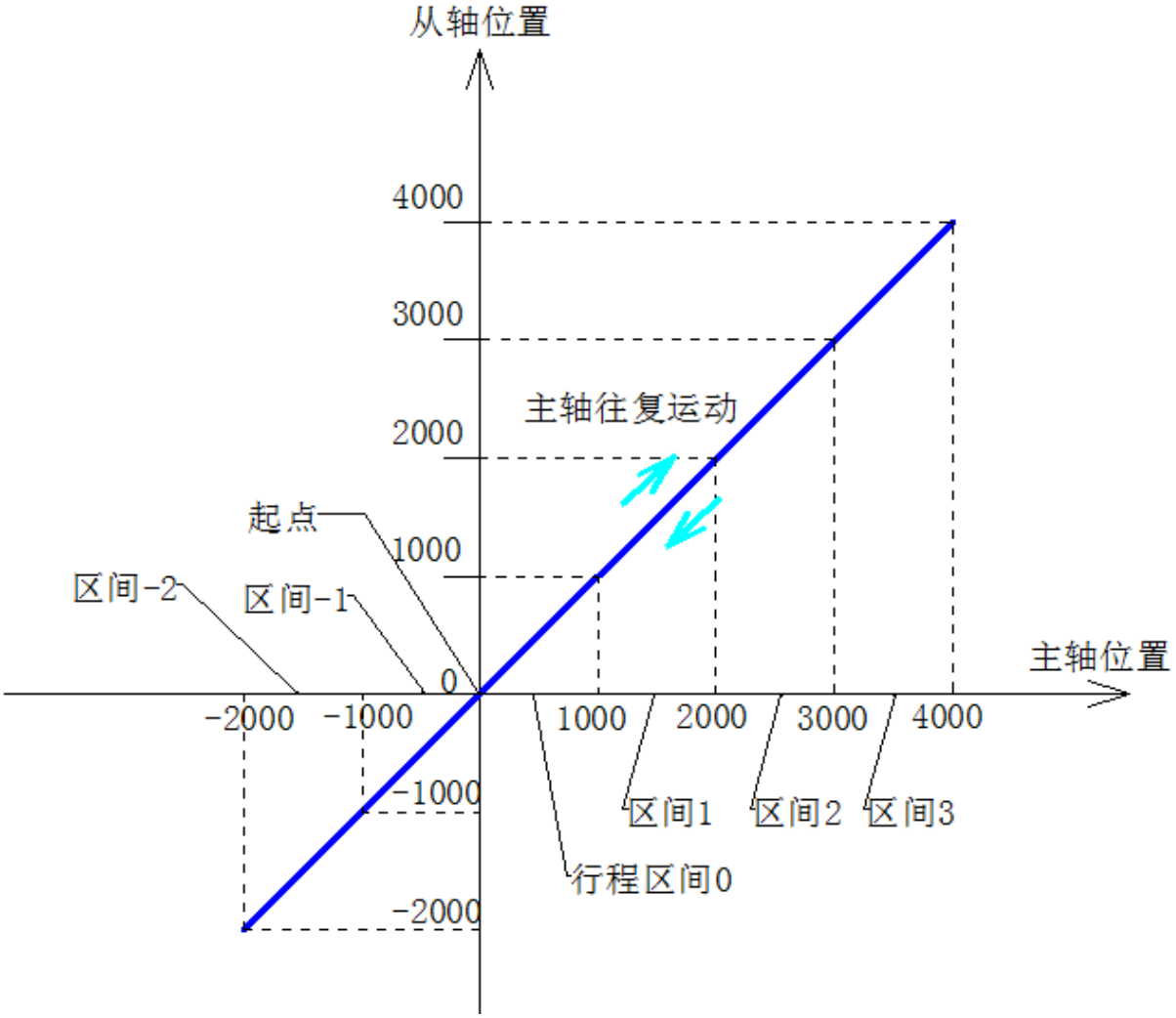
```
(**生成主从轴位置数组**)
delta := 3.1415926*2/100;
FOR n := 0 TO 100 DO
ArrPos2[0,n] := (delta*n)*150;
ArrPos2[1,n] := SIN(delta*n)*1000;
END_FOR

(*投放单次 DI 启动凸轮指令，0 轴为主轴，1 轴为从轴，DI 道号为 0*)
HMC_Cam (Axis1.AxisID,Axis0.AxisID,ADR(ArrPos2),101,1,3,0);
```

（5） 循环立即启动凸轮(ucMode = 4)

示例程序如下（ST 语言），该程序凸轮主轴与从轴联动运行的轨迹为倾角为 45 度的直线。

凸轮指令投放运行后，以当前主轴 Mpos 值为启动位置立即启动凸轮功能，向凸轮主轴投放 Hmc_Move 指令驱动凸轮主轴往复运动，此时凸轮从轴作相应的联动动作，当主轴一个凸轮行程结束时，进入下一个凸轮行程，从轴以上一个凸轮行程的终点为起点作增量计算，执行周期性的联动动作（主从轴起始位置均为 0）。

```
(**主从轴位置数组赋值**)
ArrPos[0,0] := 0;
ArrPos[1,0] := 0;       (*点 1 坐标*)
ArrPos[0,1] := 500;
ArrPos[1,1] := 500;     (*点 2 坐标*)
ArrPos[0,2] := 1000;
ArrPos[1,2] := 1000;    (*点 3 坐标*)

HMC_Cam(Axis1.AxisID,Axis0.AxisID，ADR(ArrPos),3,1,0,0); (*投放单次立即启动凸轮指令，0 轴为主轴，1 轴为从轴，凸轮主轴行程范围为*)

(*向主轴投送 Move 指令，驱动凸轮主轴往复运动。此时凸轮从轴将进行相应联动动作*)
HMC_Move(Axis0.AxisID,4000);
HMC_Move(Axis0.AxisID,-6000);
HMC_Move(Axis0.AxisID,6000);
HMC_Move(Axis0.AxisID,-4000);
```



循环立即启动凸轮联动轨迹

（6） 循环固定位置启动凸轮(ucMode = 5)

示例程序如下（ST 语言），该程序凸轮主轴与从轴联动运行的轨迹为倾角为正弦曲线。向凸轮主轴投放 Hmc_Move 指令驱动凸轮主轴运动，当凸轮主轴达到启动位置时，凸轮指令启动运行，凸轮从轴开始作相应的联动动作（主从轴起始位置均为 0）。

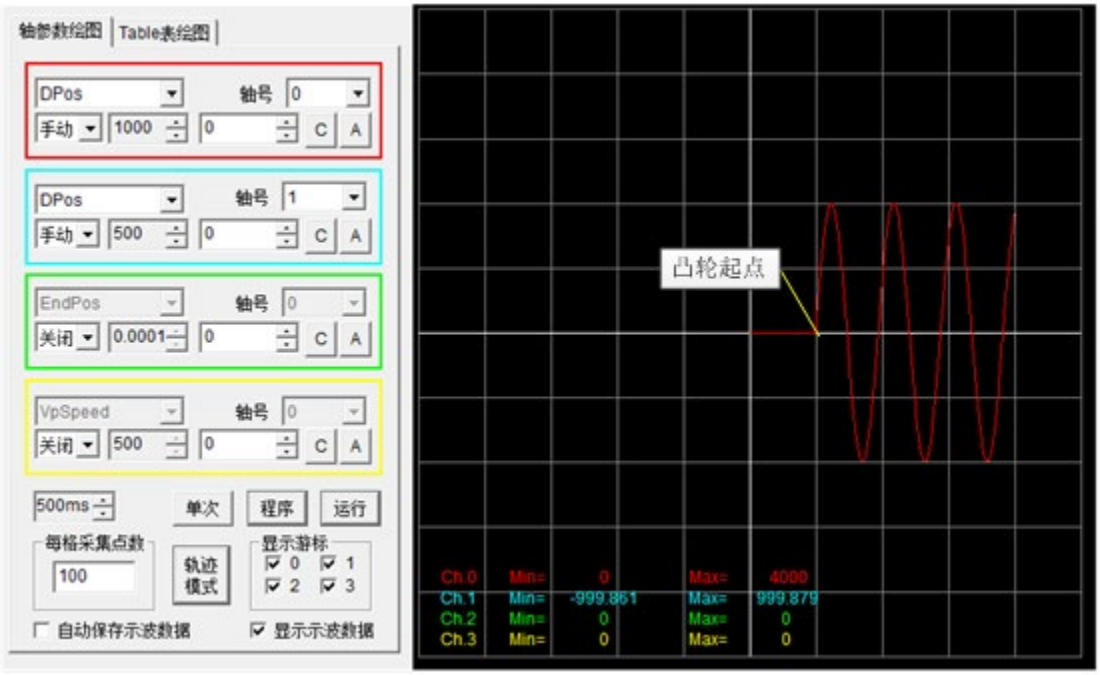
```
(**生成主从轴位置数组**)
delta := 3.1415926*2/100;
FOR n := 0 TO 100 DO
ArrPos2[0,n] := (delta*n)*150;
ArrPos2[1,n] := SIN(delta*n)*1000;
END_FOR
```

(*投放单次事件启动凸轮指令，0 轴为主轴，1 轴为从轴，固定启动位置为 1000*)

HMC_Cam (Axis1.AxisID,Axis0.AxisID,ADR(ArrPos2),101,1,5,1000);

(*向主轴投送 Move 指令，驱动凸轮主轴运动。*)

HMC_Move(Axis0.AxisID,4000);



循环固定位置启动凸轮联动轨迹

（7） 循环事件启动凸轮(ucMode = 6)

除启动后撤消条件不同之外，其余与单次事件启动凸轮类似，可参考（3）中示例。

（8） 循环 DI 启动凸轮(ucMode = 7)

除启动后撤消条件不同之外，其余与单次 DI 启动凸轮类似，可参考（4）中示例。

5.4.6.2 HMC_SplineCam（样条凸轮）

5.4.6.2.1 功能说明

该指令功能与 HMC_Cam 功能类似，区别在于 HMC_SplineCam 使用样条曲线拟合的方式把主从轴位

置点连接起来，而 HMC_Cam 采用的是折线连接的方式。因此，样条凸轮生成的轨迹更为顺滑，在运动过程中没有速度的跃变。

其它相关的说明及指令示例可参见 HMC_Cam（电子凸轮）。

5.4.6.2.2 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号
ucMasterAxis	USINT	主轴轴号
pArrPos	POINTER TO LREAL	位置数组首地址（二维） 在二维数组 pArrPos[2,uiPosNum]中，pArrPos[0]为主轴位置数组，pArrPos[1]为从轴位置数组
uiPosNum	UDINT	位置个数，应大于等于 3
dScale	LREAL	凸轮从轴位置数据比例因子。凸轮从轴实际位置输出=从轴位置数组设定输出* dScale
ucMode	USINT	0：单次立即启动 1：单次固定位置启动 2：单次事件启动 3：单次 DI 启动 4：循环立即启动 5：循环固定位置启动 6：循环事件启动 7：循环 DI 启动
dTrigMes	LREAL	启动信息： 1、凸轮为位置启动时，该值表示主轴的启动位置 2、凸轮为事件触发启动时，表示某高速捕获通道号 3、凸轮为 DI 为 1 触发启动时，表示 DI 通道（0~63）

5.4.6.2.3 指令类型

轴运动缓存指令。

5.4.6.3 HMC_GetCamMasterPosIndex（获取电子凸轮主轴位置索引）

5.4.6.3.1 功能

获取当前正在运行的凸轮主轴位置索引（当前凸轮主轴位置在凸轮主轴位置数组中的区间索引，应为 0 到 uiPosNum-1）。如果当前指令不为凸轮，函数返回 0。

5.4.6.3.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BOOL	轴号

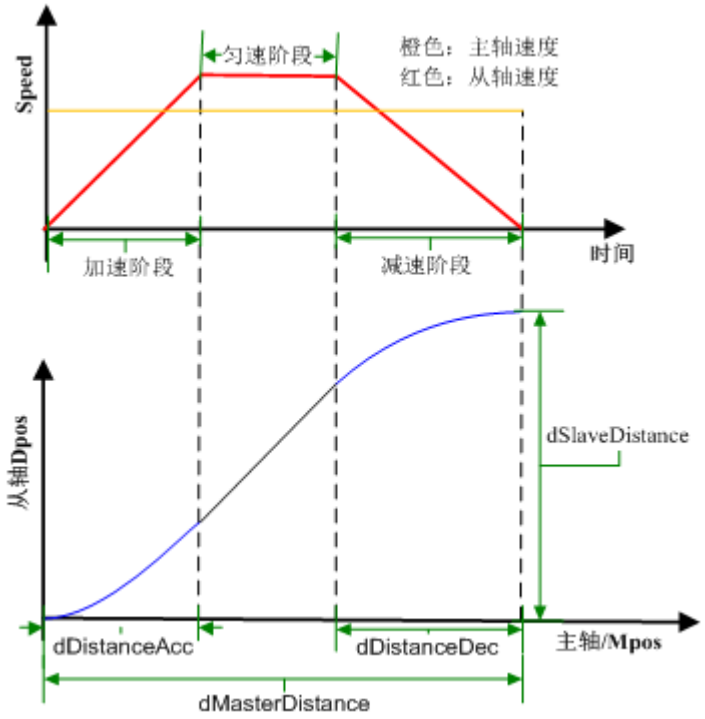
5.4.6.3.3 指令类型

非轴运动缓存指令。

5.4.6.4 HMC_ClcMoveLinkData（追剪运算）

5.4.6.4.1 功能

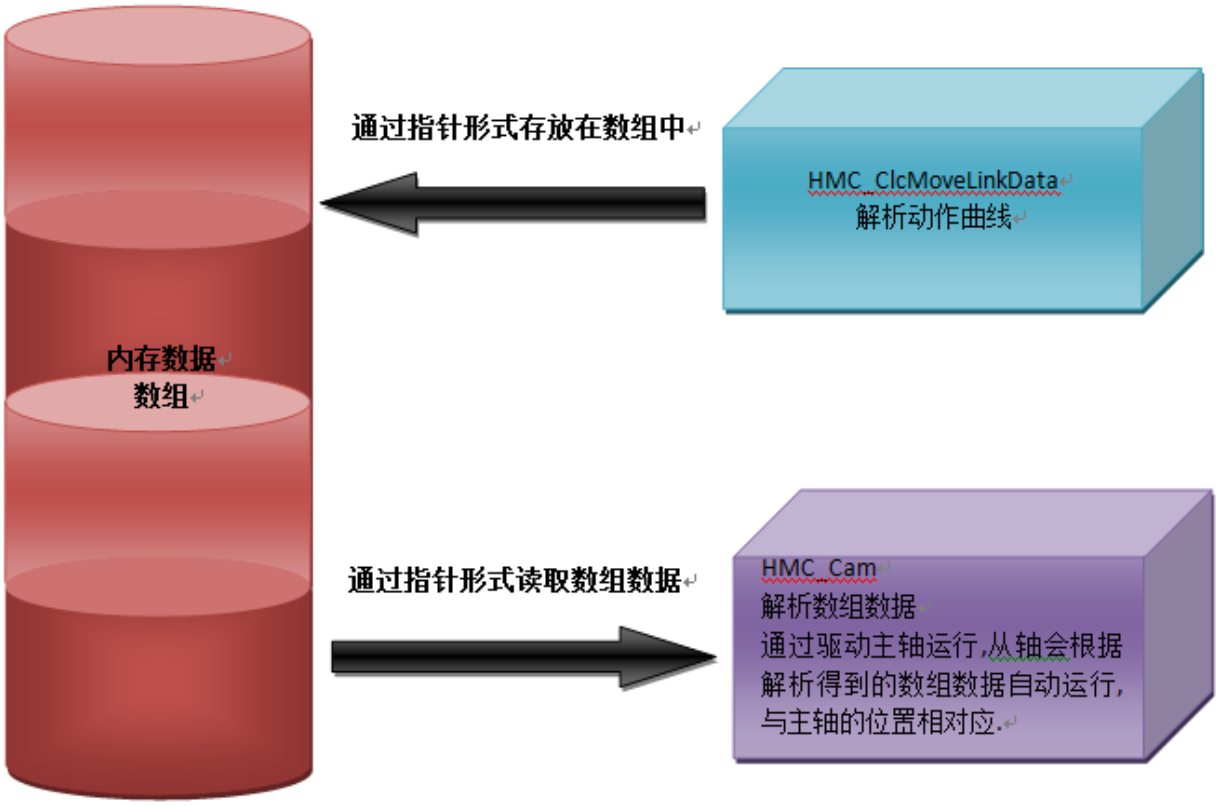
该指令可按照设定的关系生成主从轴位置数据，位置数据保存在数组中，可被凸轮指令使用，控制从轴跟随主轴按一定规律运动。使用该指令所生成的主从轴位置数据，当凸轮主轴匀速运动时，主从轴曲线如下图所示。该运动带有分离可变的加减速阶段。



主/从轴示意图

通过设定从轴位移 dSlaveDistance、主轴位移 dMasterDistance、从轴加速阶段主轴位移 dDistanceAcc、从轴减速阶段主轴位移 dDistanceDec，HMC_ClcMoveLinkData 可生成满足上图所示的主从轴位置数据，数据保存在数组 pArrPos 中，主/从轴位置个数由 uiPosNum 参数指定。

在 HMC_ClcMoveLinkData 与 HMC_Cam 指令配合使用时，使用数组传递主从轴位置数据。首先将曲线通过 HMC_ClcMoveLinkData 指令运算的主从轴位置数据放在数组中，然后通过 HMC_Cam 指令调用该数组数据，最后驱动主轴运行，从轴通过 HMC_Cam 指令解析数组中位置数据实现对主轴位置的跟随。



HMC_ClcMoveLinkData 与 HMC_Cam 数据流

该指令可用在追剪场景中，配合凸轮指令 HMC_Cam，快速生成追剪所需的动作流程，无需进行复杂的中间计算与编程。

5.4.6.4.2 参数

输入参数	类型	参数描述
dSlaveDistance	LREAL	从轴运动距离
dMasterDistance	LREAL	主轴运动距离（必须为正）
dDistanceAcc	LREAL	在从轴加速阶段主轴移动的正向距离，非负
dDistanceDec	LREAL	在从轴减速阶段主轴移动的正向距离，非负
pArrPos	POINTER TO LREAL	位置数组（二维） 在二维数组 pArrPos[2][uiPosNum]中，pArrPos[0]为主轴位置数组，pArrPos[1]为从轴位置数组
uiPosNum	UDINT	位置个数，应大于等于 6

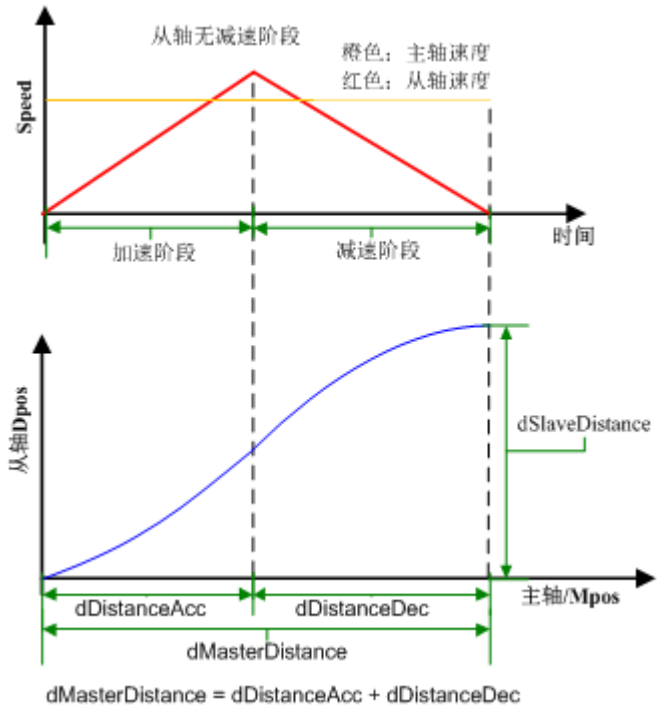
5.4.6.4.3 指令类型

非轴运动缓存指令。

5.4.6.4.4 补充说明

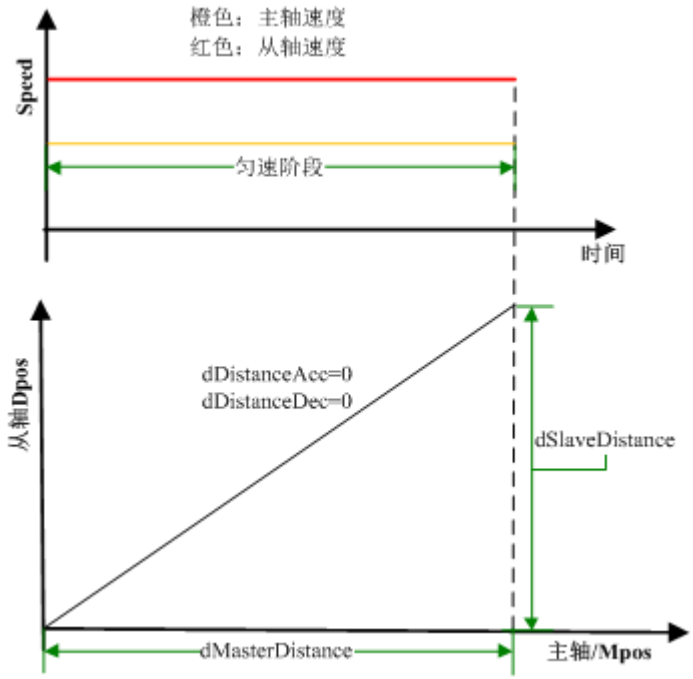
(1) 该函数中的距离均采用用户单位。需满足 dDistanceAcc + dDistanceDec ≤ dMasterDistance，否则返回参数错误。

(2) 当 dDistanceAcc + dDistanceDec = dMasterDistance 时，下图中凸轮从轴将无匀速阶段，此时凸轮主从轴曲线如下：



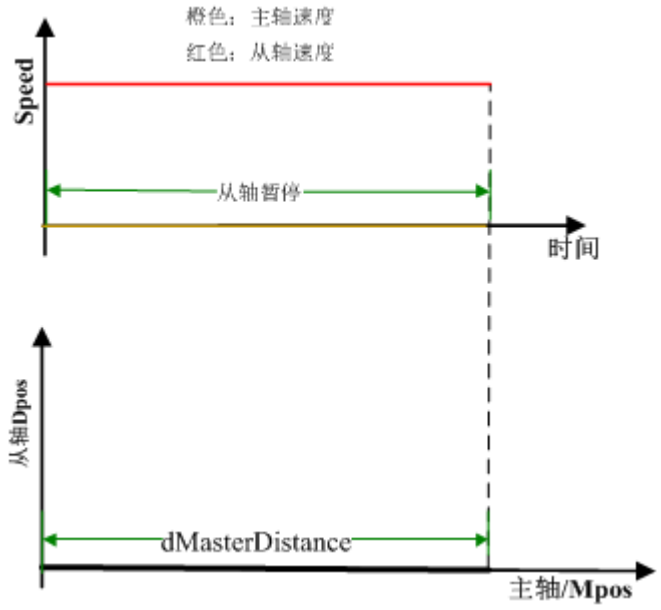
主/从轴示意图

(3) 当 dDistanceAcc、dDistanceDec 均为零时，即无分离的加、减速时，此时的联动等效于电子齿轮，下图中的凸轮主从轴曲线如下：



主/从轴示意图

（4） 若 dSlavetDistance 为 0，当凸轮主轴在设定区域运动时，从轴将暂停等待。下图中的凸轮主从轴曲线如下：



主/从轴示意图

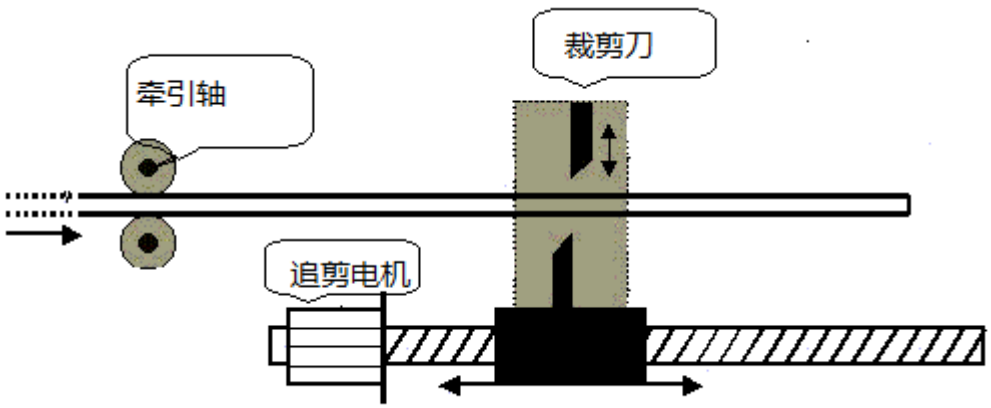
5.4.6.4.5 示例

■ 追剪

在传统的定长度物料加工系统中，一般是由送料伺服电机将物料输出设定的长度后停止送料电机，然后驱动加工机构（如裁刀等）进行加工，加工完毕后才能继续送料。传统的此类设备有钢板校平轧断机、各类管材的定长裁断、纺织机械的落砂装置等。由于加工中送料动作需要间歇性停止，因此此类设备生产效率低下，且只能用在定长输出的物料可以间歇停止的场合。在无缝钢管生产线、送料装置为挤出机等场合，或对生产效率有较高要求的应用场景，设备不允许间歇停止，上述传统设备便无法满足客户需求，由此需要引入追剪方案。

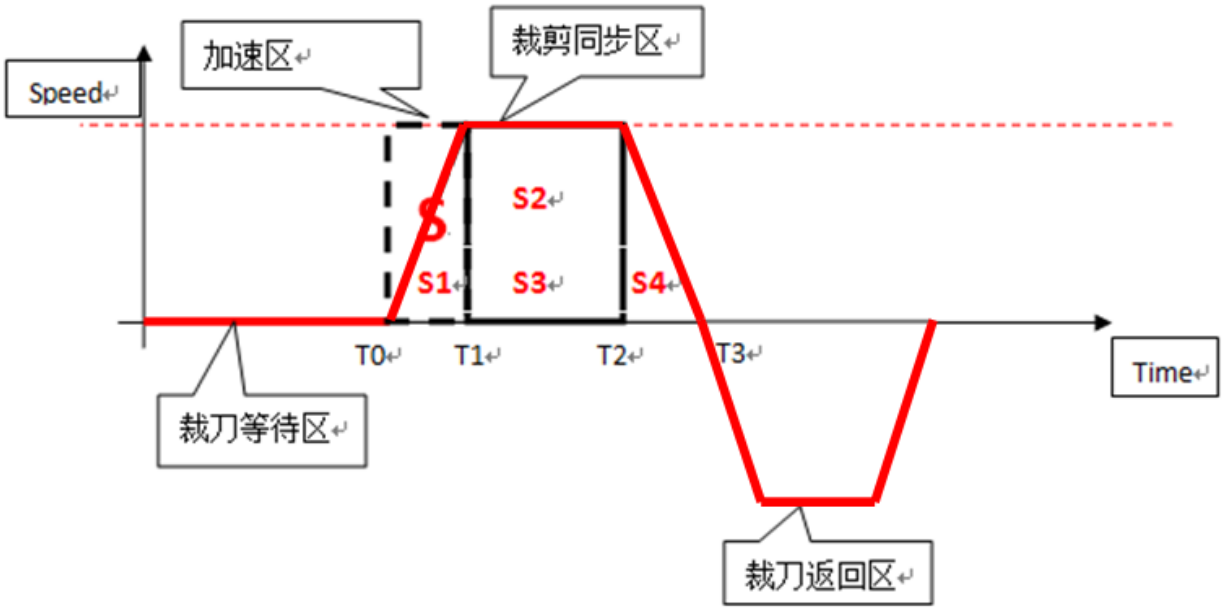
追剪就是在物料剪裁过程中，物料保持不间歇的传送，而裁断装置做往复运动，通过对裁断装置进行速度及位置规划，使得剪裁装置与物料速度达到同步时，所要剪裁的物料长度刚好满足预定要求，此时进行物料裁剪，剪裁动作完成后返回等待位置等待一定时间，之后再次追踪物料同步裁剪，周而复始。以下举例说明。

物料通过牵引轴从左至右匀速运行，裁刀从初始位置开始追踪物料到达同步区后进行裁剪动作，裁剪完毕后，裁刀快速返回至初始位置，等待一段时间后继续下一个动作循环。

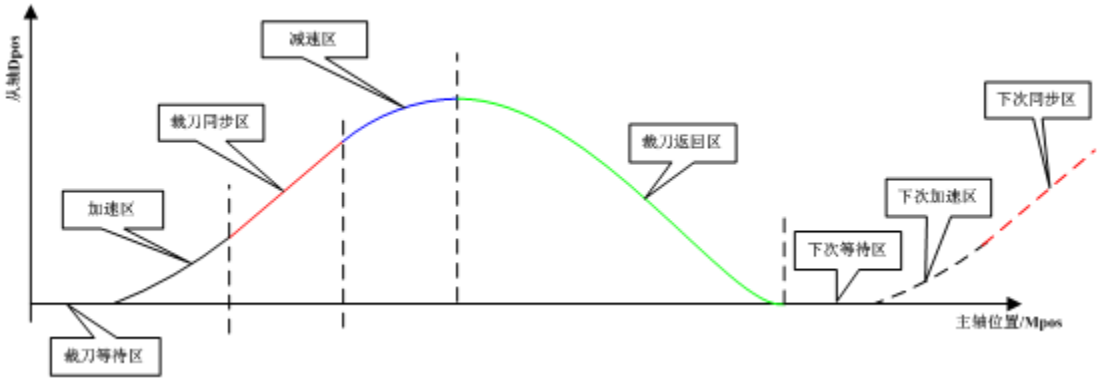


追剪方案示意图

满足追剪要求的速度曲线如下图所示，途中红色虚线为牵引轴速度，红色实线为裁剪轴速度。通过 HMC_ClcMoveLinkData 及 HMC_Cam 指令即可实现该追剪动作流程。设牵引轴为系统的主轴，追剪电机为系统从轴。



追剪速度规划图



追剪主从轴位移曲线

裁剪长度设定 100 m，裁刀移动行程 3 m，丝杠导程 30 mm，追剪电机转速 2000 r/min，速比 1:1。将运动轨迹分成以下几部分：等待区、加速区、同步区、减速区、返回区。在常速区域可以细化曲线来配合裁剪刀动作。设在加速区（T0~T1 时刻）裁剪轴的行走距离是 S1，牵引轴行走距离是 S，计算可知：S=2S1；在同步区（T1~T2 时刻），裁刀速度 裁剪轴行走距离 S2 = 牵引轴行走距离 S3；若加减速时间相等，则在减速区（T2~T3 时刻）裁剪轴行走距离 S4=S1，牵引轴行走距离为 S。

由已知参数得知裁剪轴速度=2000*30=60,000 mm/min=1000 mm/s。设加减速时间为 1 s，则计算可得：

S=1000 mm

S1=S4=500 mm

S2=S3=2000 mm

裁刀等待区域的主轴位移 = 总裁剪长度 - 裁刀正向动作区主轴位移 -裁刀反向动作区主轴位移。

本例设定裁刀正向动作与反向动作所用的加减速、匀速速度大小相同，则有裁刀正反向动作时间相等，因此：

裁刀反向动作区主轴位移 = 裁刀正向动作区主轴位移 = S+S2+S = 4000 (mm)

裁刀正向动作区从轴位移 = S1+S2+S4 = 3000 (mm)

裁刀反向动作区从轴位移 = -（裁刀正向动作区从轴位移 = -3000 (mm)

裁刀等待区域的主轴位移 = 100000 – 4000 - 4000 = 92000(mm)

裁刀等待区域的从轴位移 = 0

程序如下：

TASK1(任务 1):初始化轴参数及追剪数据

(**** 使用 HMC_SetUnits 把用户单位转换为 mm（略）****)

(****使用 ADR 取位置数组首地址****)

```
add1 := ADR(Pos);
add2 := ADR(Pos1);
add3 := ADR(Pos2);
add4 := ADR(Pos3);
add5 := ADR(Pos4);
```

(**** 生成主从轴位置数组 ****)

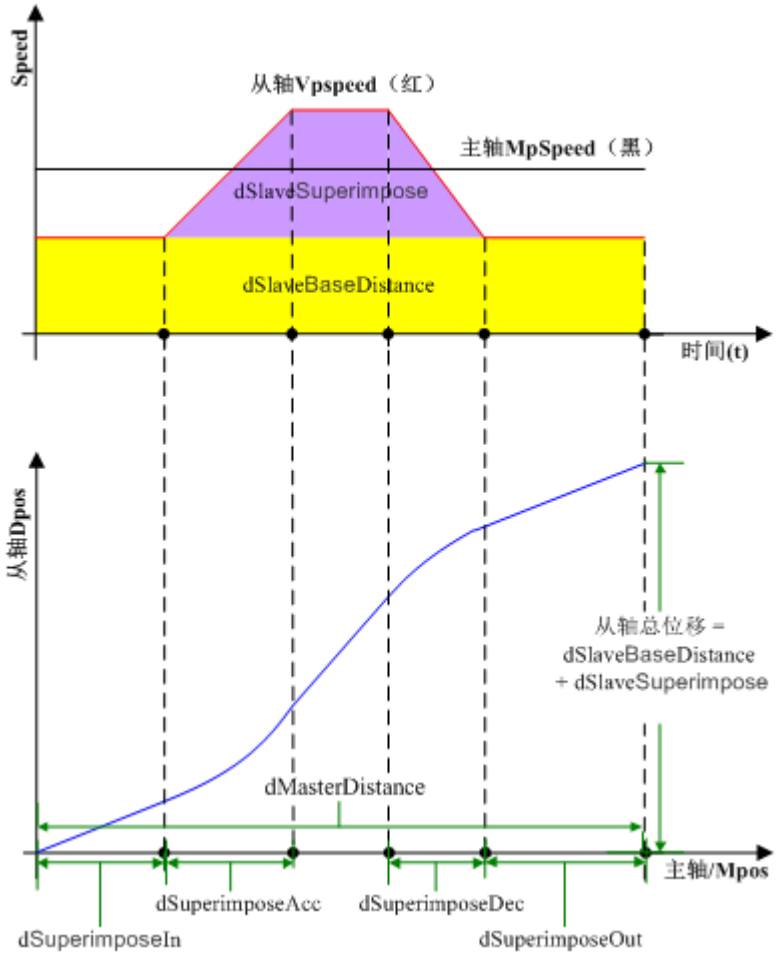
```
HMC_ClcMoveLinkData(0, 92000, 0, 0, add1, PosNum);          (* 裁剪等待区 *)
HMC_ClcMoveLinkData(500, 1000, 1000, 0, add2, PosNum);      (*裁剪加速区 *)
HMC_ClcMoveLinkData(2000, 2000, 0, 0, add3, PosNum);        (* 裁剪同步区 *)
HMC_ClcMoveLinkData(500, 1000, 0, 1000, add4, PosNum);      (* 裁剪减速区 *)
```

```
HMC_ClcMoveLinkData(-3000, 4000, 1000, 1000, add2, PosNum); (* 裁剪返回区 *)

(**** 使用各阶段主从轴位置数组投放凸轮指令****)
While 1 do                              (* 循环投送追剪凸轮*)
HMC_Cam(1, 0, add1, PosNum, 1, 0, 0);           (* 裁剪等待区 *)
HMC_Cam(1, 0, add2, PosNum, 1, 0, 0);           (* 裁剪加速区 *)
Cut_Enable := HMC_Cam(1, 0, add3, PosNum, 1, 0, 0);  (* 裁剪同步区 *)
HMC_Cam(1, 0, add4, PosNum, 1, 0, 0);           (* 裁剪减速区 *)
HMC_Cam(1, 0, add5, PosNum, 1, 0, 0);           (* 裁剪返回区 *)

(**** 在同步区进行裁剪动作 ****)
Camload_Ok := HMC_WaitCmdLoaded(Cut_Enable);       (* 在同步区驱动裁剪 *)
%QX0.0:= 1;      (* 裁刀动作。假设裁刀动作为开关量控制。 *)

(****同步区结束后收回裁刀，裁剪结束 ****)
CamFinish_Ok := HMC_WaitCmdFinish(Cut_Enable);   (* 同步区结束后立即停止裁剪 *)
%QX0.0 := 0;     (* 裁刀收回 *)
End_While;
(**** 注意:在实际应用过程中,为了保护裁刀可以将同步区再细化。 ****)
TASK2(任务 2):
HMC_Forward(0);     (*主轴位移启动 *)
```

5.4.6.5 HMC_ClcFlexLinkData（飞剪/旋切运算）

5.4.6.5.1 功能

该指令可按照设定的关系生成主从轴位置数据，位置数据保存在数组中，可被凸轮指令使用，控制从轴跟随主轴按一定规律运动。使用该指令所生成的主从轴位置数据，当凸轮主轴匀速运动时，主从轴曲线如下图所示。从轴的跟随运动由恒速距离（dSlaveBasetDistance）和带分离加减速的变速距离（dSlaveSuperimpose）叠加而成。

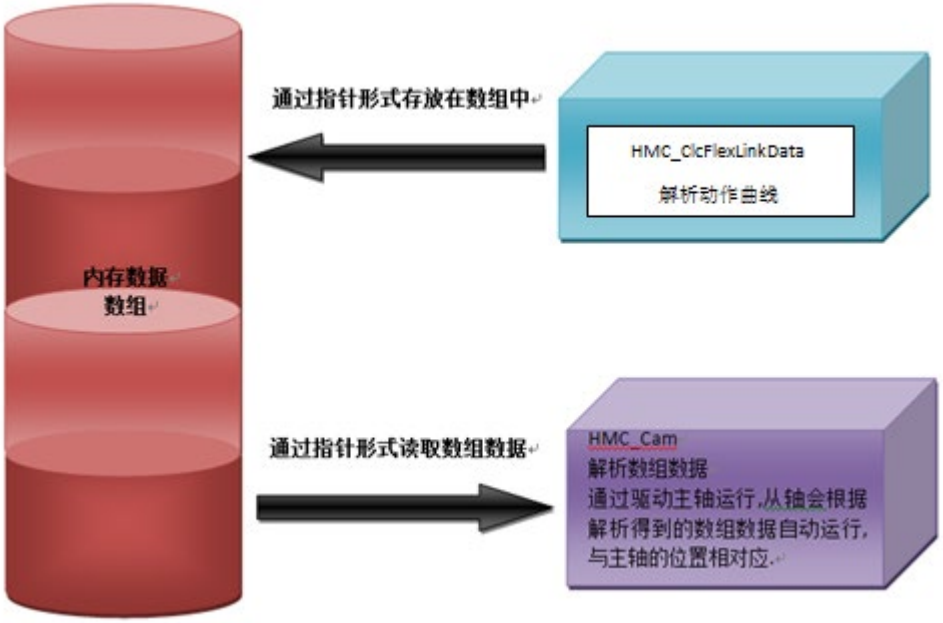


主/从轴示意图

通过设定所需的参数（见输入参数列表），HMC_ClcMoveLinkData 即可生成满足上图所示的主从轴位置数据，数据保存在数组 pArrPos 中，主/从轴位置个数由 uiPosNum 参数指定。

在 HMC_ClcFlexLinkData 与 HMC_Cam 指令配合使用时，使用数组传递主从轴位置数据。首先将曲

线通过 HMC_ClcFlexLinkData 指令运算的主从轴位置数据放在数组中，然后通过 HMC_Cam 指令调用该数组数据，最后驱动主轴运行，从轴通过 HMC_Cam 指令解析数组中位置数据实现对主轴位置的跟随。



HMC_ClcFlexLinkData 与 HMC_Cam 数据流

该指令可用在飞剪/旋切场景中，配合凸轮指令 HMC_Cam，快速生成追剪所需的动作流程，无需进行复杂的中间计算与编程。

5.4.6.5.2 参数

输入参数	类型	参数描述
dSlaveBaseDistance	LREAL	从轴恒速跟随距离
dSlaveSuperimpose	LREAL	从轴叠加运动距离
dMasterDistance	LREAL	主轴运动距离（必须为正）
dSuperimposeIn	LREAL	叠加运动（dSlaveSuperimpose）开始时主轴已完成正向距离，非负
dSuperimposeOut	LREAL	叠加运动（dSlaveSuperimpose）结束时主轴剩余正向距离，非负
dSuperimposeAcc	LREAL	叠加运动加速阶段主轴移动的正向距离，非负
dSuperimposeDec	LREAL	叠加运动减速阶段主轴移动的正向距离，非负
pArrPos	POINTER TO LREAL	位置数组（二维） 在二维数组 pArrPos[2][uiPosNum]中，pArrPos[0]为主轴位置数组， pArrPos[1]为从轴位置数组
uiPosNum	UDINT	位置个数，应大于等于 6

5.4.6.5.3 指令类型

非轴运动缓存指令。

5.4.6.5.4 补充说明

（1） 主轴运动距离（dMasterDistance）严格为正，否则返回参数错误。

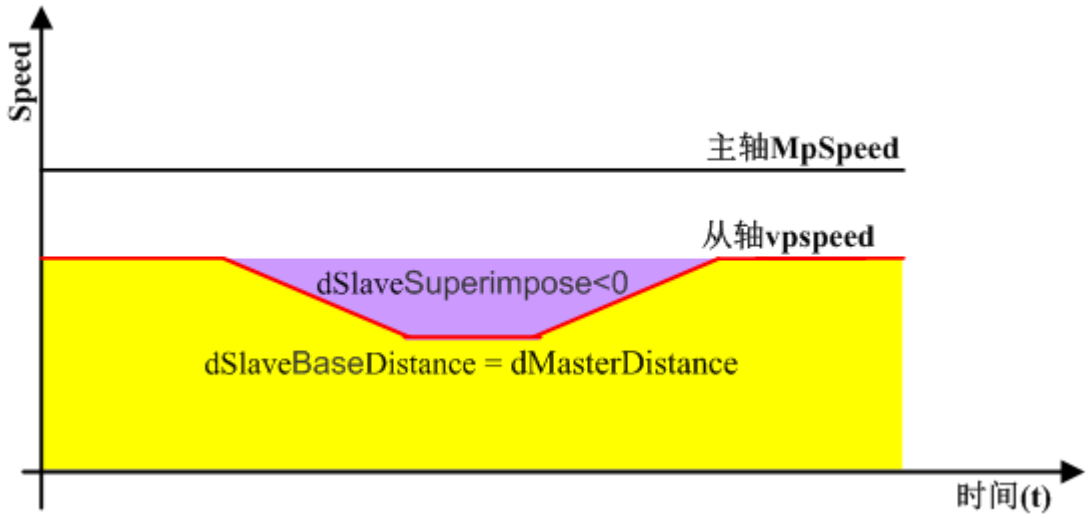
从轴位移等于 dSlaveBaseDistance 与 dSlaveSuperimpose 之和。

（2） 应满足：

$$dSuperimposeIn + dSuperimposeOut + dDistanceAcc + dDistanceDec \leq dMasterDistance$$

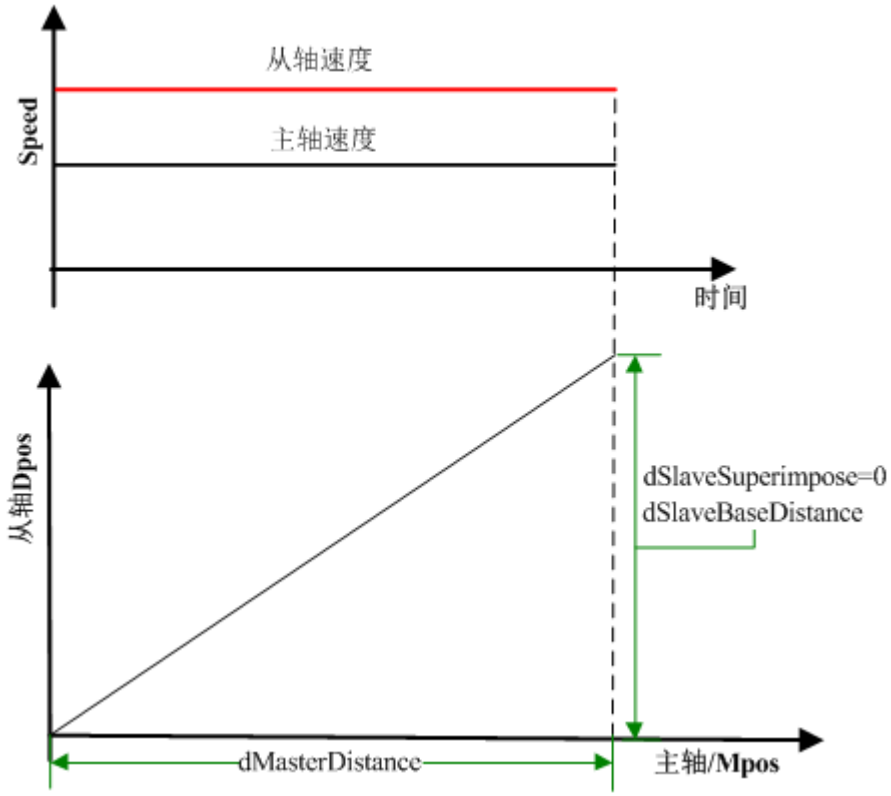
否则返回参数错误（0xFFFFFFFF）。

（3） dSlaveBaseDistance、dSlaveSuperimpose 正负均可，若两个参数值均为 0 时，在主轴未执行完 dMasterDistance 之前，从轴一直处于等待状态，与 HMC_MoveLink 的暂停等待类似。



从轴叠加距离为负时的情形

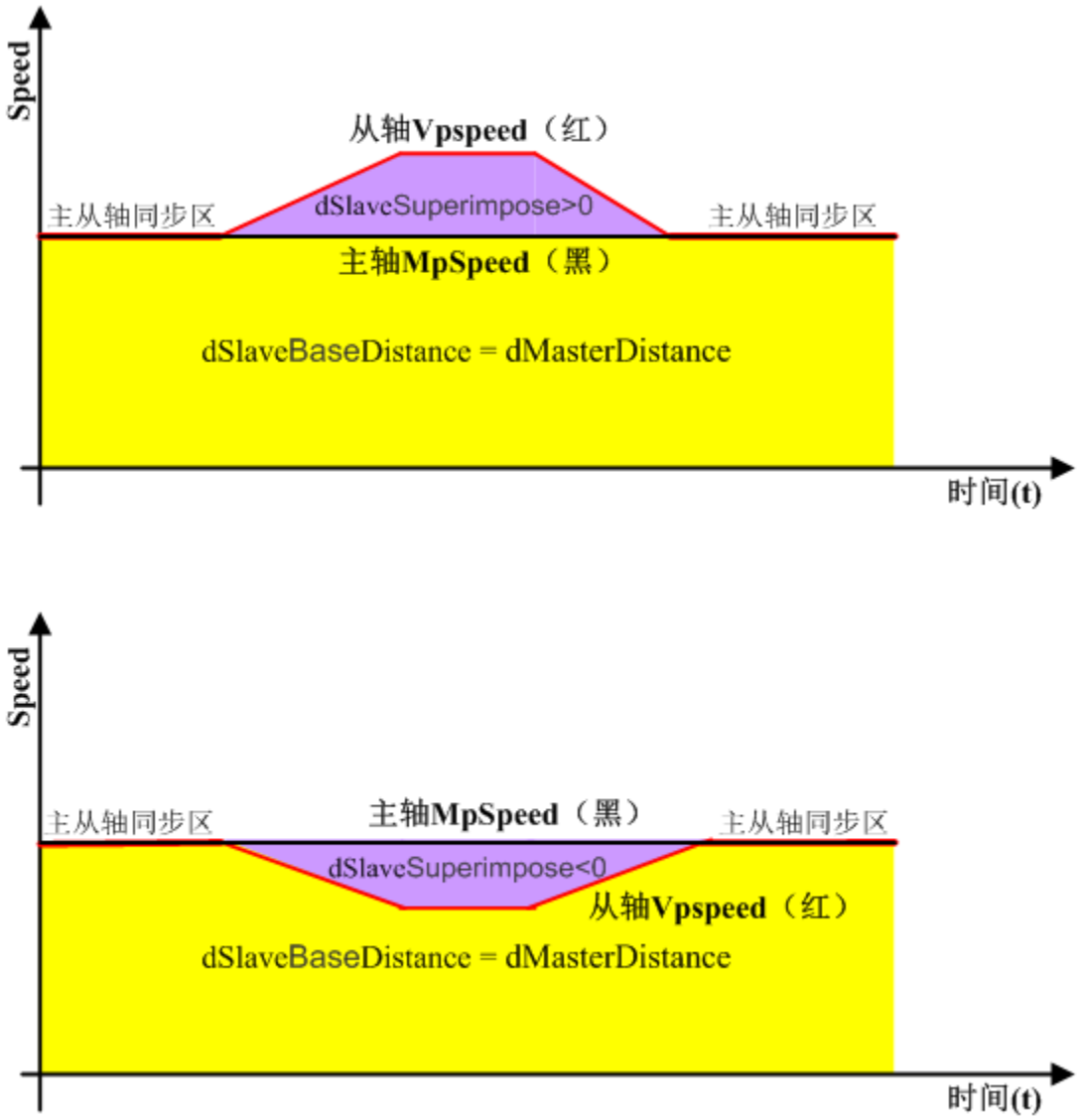
（4） dSlaveSuperimpose 为 0 时，主从轴将按照一定的速度比例关系运动，参数 dSuperimposeIn、dSuperimposeOut、dSuperimposeAcc、dSuperimposeDec 的设置将失效。此时的联动等效于电子齿轮。



运动示意图

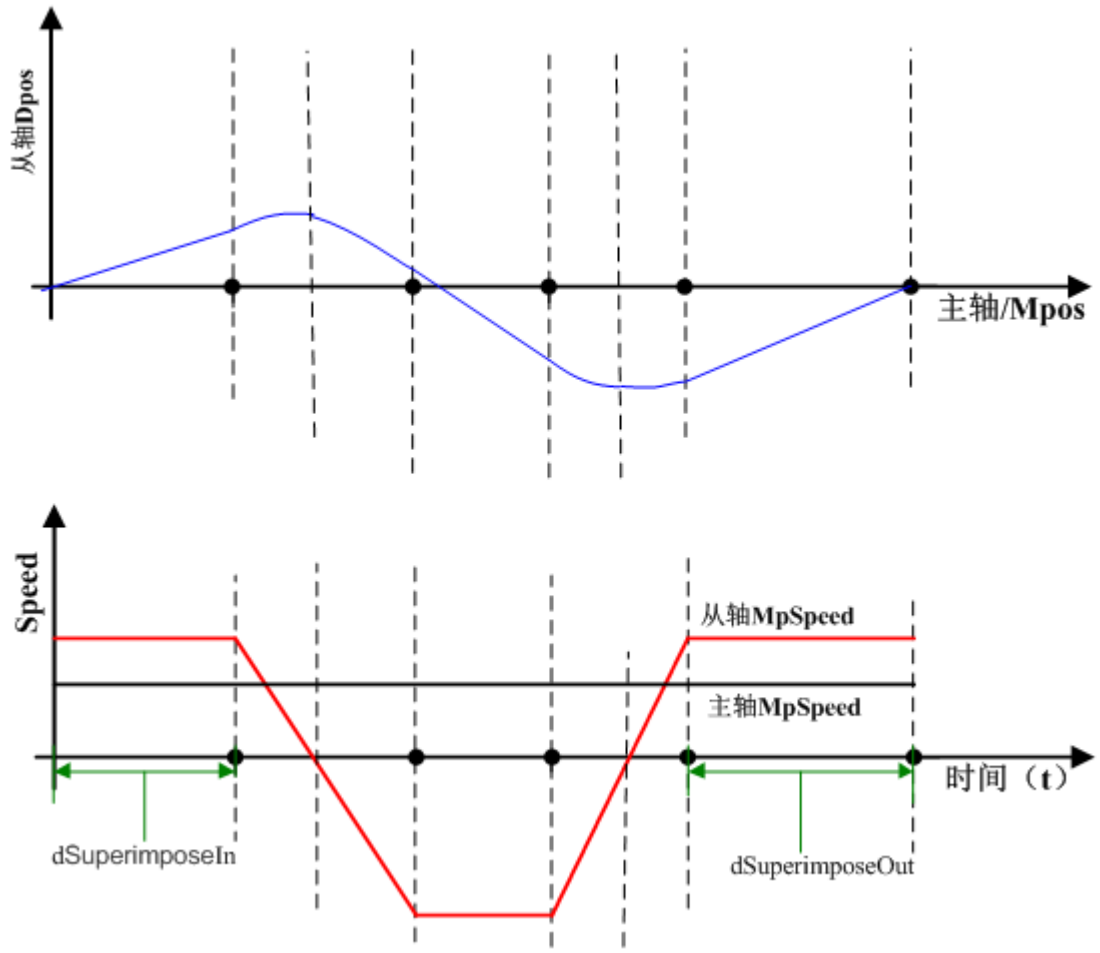
（5） 当从轴恒速跟随位移与主轴位移相等时，即：

dSlaveBaseDistance = dMasterDistance，在从轴叠加区域之外，从轴与主轴的运动将保持同步。



运动示意图

（6） 当 dSlaveBaseDistance + dSlaveSuperimpose = 0，且 dSlaveBaseDistance 不为 0 时，通过叠加补偿后从轴位置回到了起始点，此情况也可用在需通过反转返回到触发位置的情况。

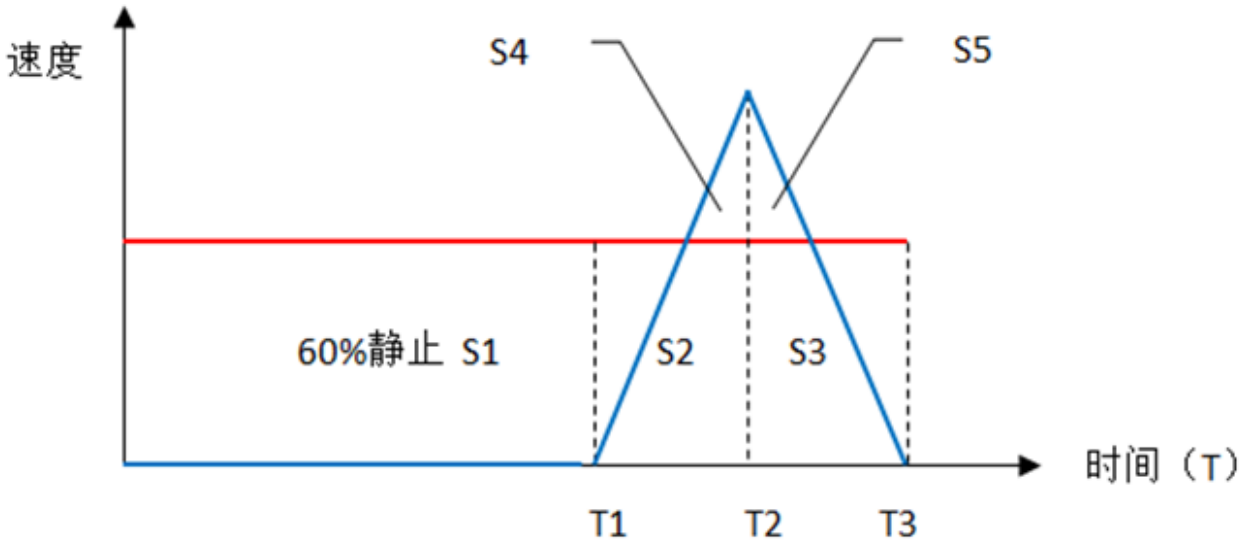


主/从轴示意图

5.4.6.5.5 示例

（1） 生成指定的联动动作

在一个循环动作中，有 40%的距离是平滑的加减速，余下的部分保持静止不动，现使用 HMC_ClcFlexLinkData 生成该动作流程。

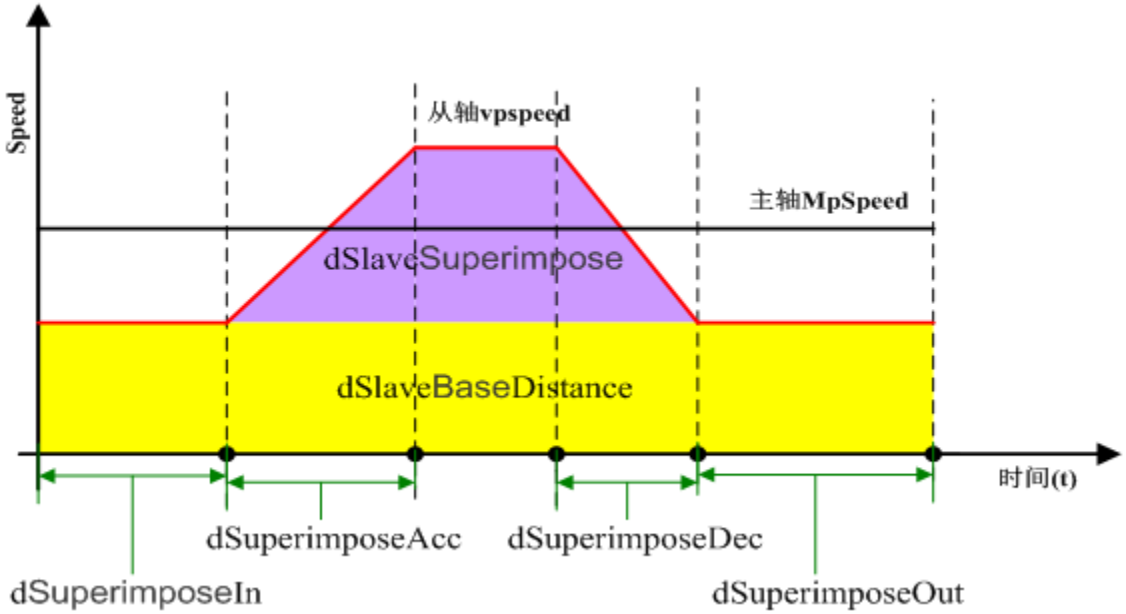


运动示意图

假设主轴的移动距离为 2000mm，从轴的移动距离为 1000mm，加速和减速距离相等。根据以上的数据可计算 HMC_ClcFlexLinkData 所需的参数值。

如上图所示：我们把主轴的面积分成 3 个部分，面积 S1（占主轴面积的 60%），也就是 2000*0.6=1200，主轴在 T1-T2 时间内位移为 S2 面积，而从轴在 T1-T2 时间内为加速过程 S4；主轴在 T2-T3 时间内位移为 S3 面积表示，而从轴在 T2-T3 时间内为减速过程 S5。假设从轴加减速时间相同，则有：从轴位移 S4=S5=500（蓝色三角形一半），主轴位移 S2=S3=(2000-1200)/2=400。

对照 HMC_ClcFlexLinkData 函数的原始图形，由于从轴只有加减速，没有基准速度。所以下图中黄色部分面积，即 dSlaveBaseDistance=0。而从轴紫色部分的面积，即为从轴加减速和匀速的面积，由于从轴没有匀速部分，只有加减速，故而：



运动示意图

dSlaveBaseDistance=0；

dSlaveSuperimpose=1000；

dSuperimposeIn=S1=1200；

dSuperimposeOut=0；

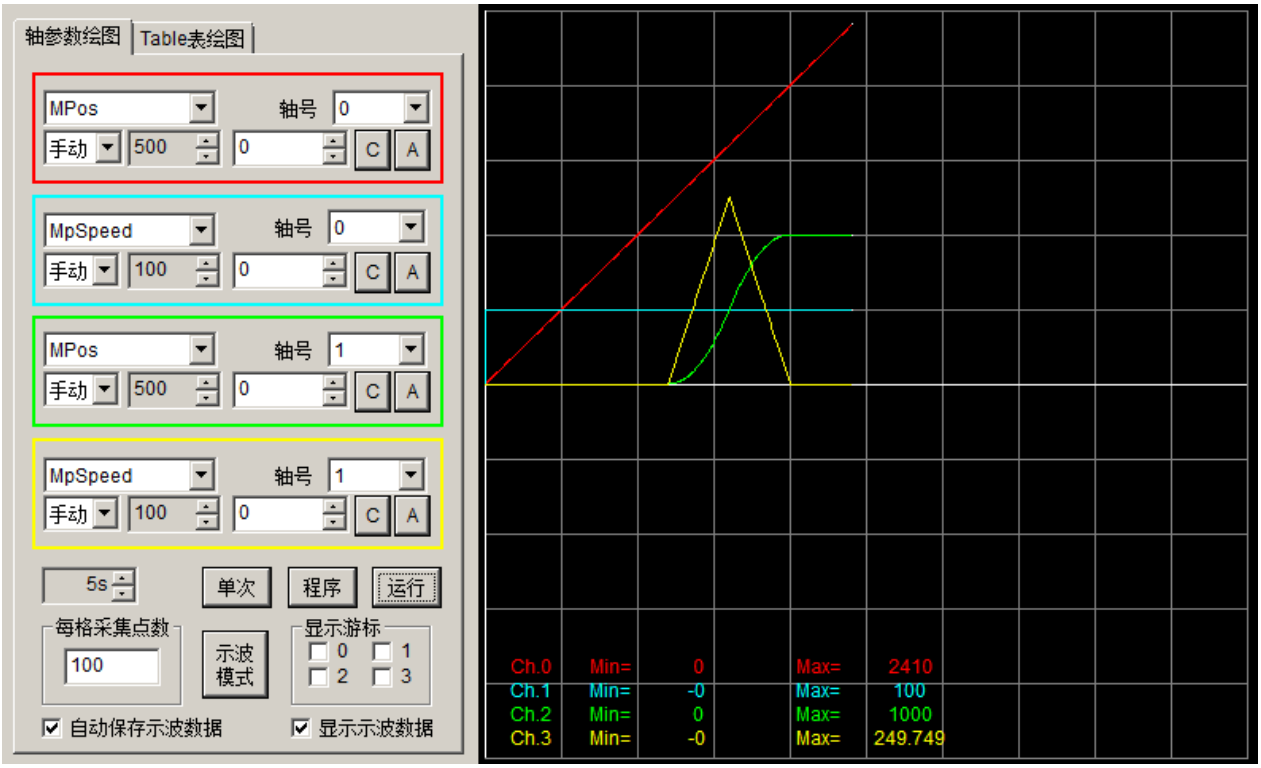
dSuperimposeAcc=dSuperimposeDec=S2=400。

具体程序如下：

```
(****初始化程序如下 ****)
(*使用 HMC_SetUnits 把用户单位转换为 mm（略）*)
DR_RESULT := HMC_DriveEnable(1);      (* 信号使能操作 *)
X_TYPE := HMC_SetAxisType(0, 0);      (* 设置 axis0 属性 *)
X_UNIT := HMC_SetUnits(0, 100);       (* 假设 100 个脉冲对应 1 mm *)
AXIS0.SPEED := 100;                   (* 单位 mm/s *)
AXIS0.ACCEL := 1000;
AXIS0.DECEL := 1000;
```

```
Y_TYPE := HMC_SetAxisType(1, 0);      (* 设置 axis1 属性 *)
Y_UNIT := HMC_SetUnits(1, 100);
AXIS1.SPEED := 100;     (* 单位 mm/s *)
AXIS1.ACCEL := 1000;
AXIS1.DECEL := 1000;
Pt_array := ADR(arraympos);
NumOfPos := 100;

(* 主要程序如下 *)
HMC_ClcFlexLinkData(0, 1000, 2000, 1200, 0, 400, 400, Pt_array, NumOfPos);
fff := HMC_TriggerScope();     (* 示波器使能 *)
camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0); (* 投送凸轮，循环立即启动 *)
X_RESULT := HMC_Forward(0);    (* 主轴启动 *)
```

在控制器中运行结果如下（青色与黄色曲线分别为主轴与从轴速度）：



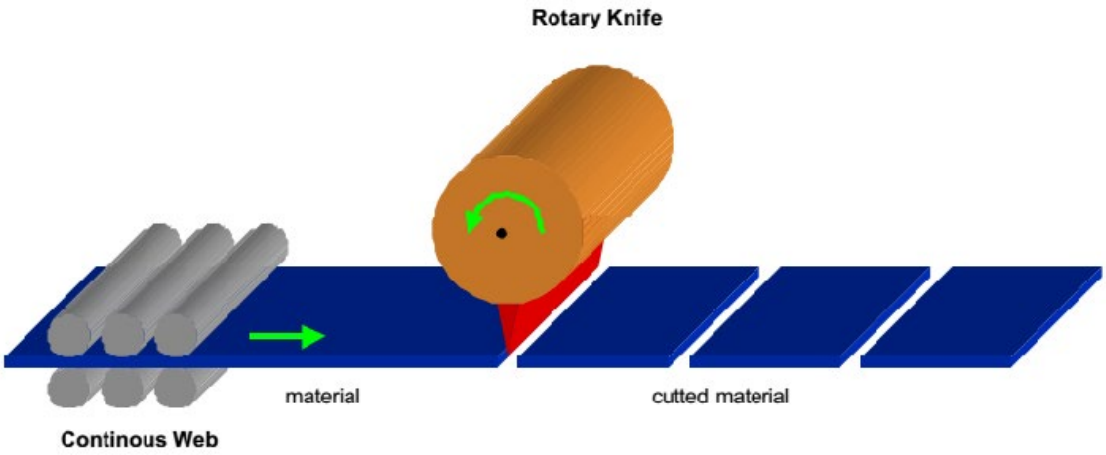
示波器显示结果

（2） 飞剪/旋切运算

飞剪常用于包装、造纸、轧钢等生产线上。在连轧钢坯车间或小型型钢车间里，它安放在轧制线的后部，将轧件切成定尺寸或仅切头切尾。在冷、热带钢车间的横剪机组、重剪机组、镀锌机组和镀锡机组里都配置有各种不同类型的飞剪，将带钢剪成定尺或裁成规定重量的钢卷。广泛采用飞剪有利于使造纸、包装、轧钢生产线迅速向高速化、连续化方向发展。因此，它是高速生产线发展的重要环节之一。

旋切工艺流程中，切刀的动作有同步区和非同步区。切刀剪切材料的区域为同步区，在剪切过程中，刀头与被剪切的材料同步运动，以保证剪切质量，同时保护刀具；在剪切完成后，根据剪切的长度不同，刀辊做加减速运动来调整刀头做下一次剪切，即为非同步区。

使用 HMC_ClcFlexLinkData 指令，我们无需使用复杂的数学表达式去求解位置点，底层的算法由控制器自动完成，只需要规划好同步、加速、减速的距离，便可自动生成飞剪/旋切运动所需的主从轴位置数据，然后在 HMC_Cam 中使用该位置数据执行即可。



飞剪/旋切示意图

假定：刀辊的周长为 600 mm，剪切同步区长度为 150 mm（两边各 75 mm）。

易知：dSuperimposeIn = dSuperimposeOut = 75mm；

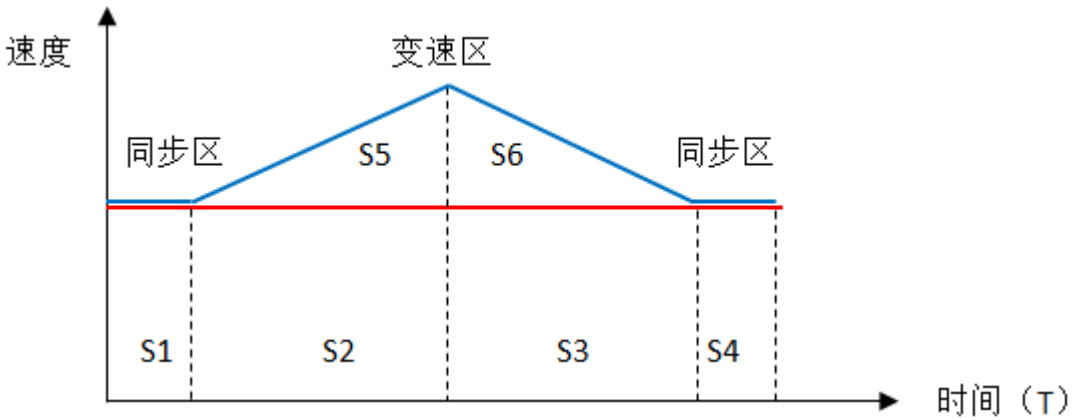
由于存在同步区，故有：dSlaveBaseDistance = dMasterDistance = 待剪切料长度。

根据所需要的剪切长度的不同，可分为短料剪切、中料剪切、长料剪切三种情况，可通过调整从轴叠加距离 dSlaveSuperimpose 来实现（刀辊周长= dSlaveBaseDistance+dSlaveSuperimpose）。

初始化程序如下：

```
DR_RESULT := HMC_DriveEnable(1);      (* 信号使能操作 *)
X_TYPE := HMC_SetAxisType(0, 0);      (* 设置 axis0 属性 *)
X_UNIT := HMC_SetUnits(0, 100);       (* 假设 100 个脉冲对应 1 mm *)
AXIS0.SPEED := 100;                   (* 单位 mm/s *)
AXIS0.ACCEL := 1000;
AXIS0.DECEL := 1000;
Y_TYPE := HMC_SetAxisType(1, 0);      (* 设置 axis1 属性 *)
Y_UNIT := HMC_SetUnits(1, 100);
AXIS1.SPEED := 100;                   (* 单位 mm/s *)
AXIS1.ACCEL := 1000;
AXIS1.DECEL := 1000;
Pt_array := ADR(arraympos)        (* 取得数组指针，数组维数应为[2,NumOfpos] *)
NumOfPos := 100;
```

(a) 情形一：短料剪切（L<剪刀周长）L=400 mm



短料剪切示意图（红色代表主轴，蓝色代表从轴）

由于剪切材料的时候主轴和从的线速度要同步，在这里我们取同步的长度为 150 mm，这样以切断点为中心，两边同步距离就是 75 mm，S1=S4=75 mm。我们假设从轴加速和减速区的距离相等，那么相对应的主轴 S2=S3=(400-150)/2=125 mm，而从轴整个位移，即蓝线所包含的面积就是剪刀的周长即 600 mm。S5+S6=(600-400)=200 mm。S5+S6 区域为蓝线区域减红线区域。

故有：

dSuperimposeIn = dSuperimposeOut = S1= S4=75mm

dSlaveBaseDistance = dMasterDistance = L=400 mm

dSlaveSuperimpose= S5+S6=200mm

dSuperimposeAcc=dSuperimposeDec=S2= S3=125mm。

具体凸轮程序如下：

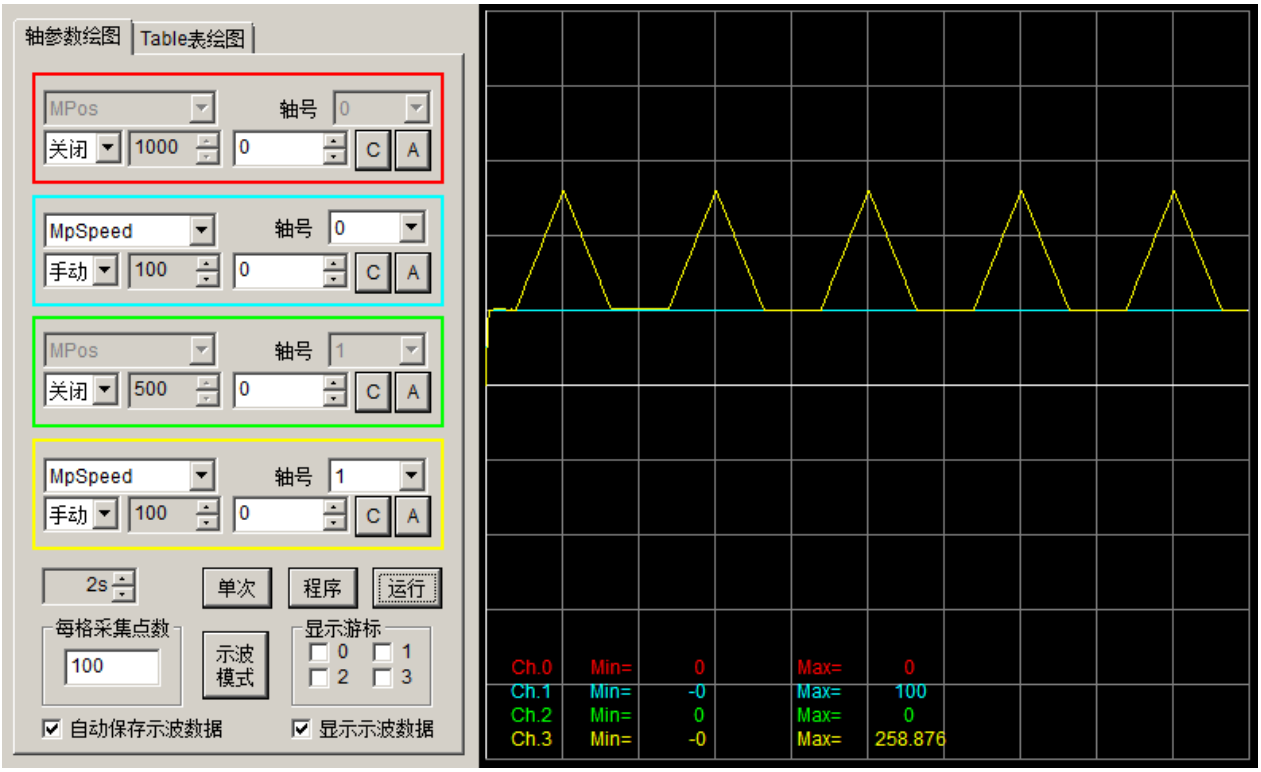
```
FlexLinkReVal := HMC_ClcFlexLinkData(400, 200, 400, 75, 75, 125, 125, Pt_array, NumOfPos);

fff := HMC_TriggerScope();     (* 示波器使能 *)

camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0); (* 投凸轮，循环立即启动 *)

X_RESULT := HMC_Forward(0);    (* 主轴启动 *)
```

AT 工程中 arraympos 数组维数应为[2,NumOfpos]。

变量名	直接地址	变量说明	变量类型	在线值	掉电保护
arraympos	%MD0		ARRAY[0..1,0..1000] OF LREAL		FALSE

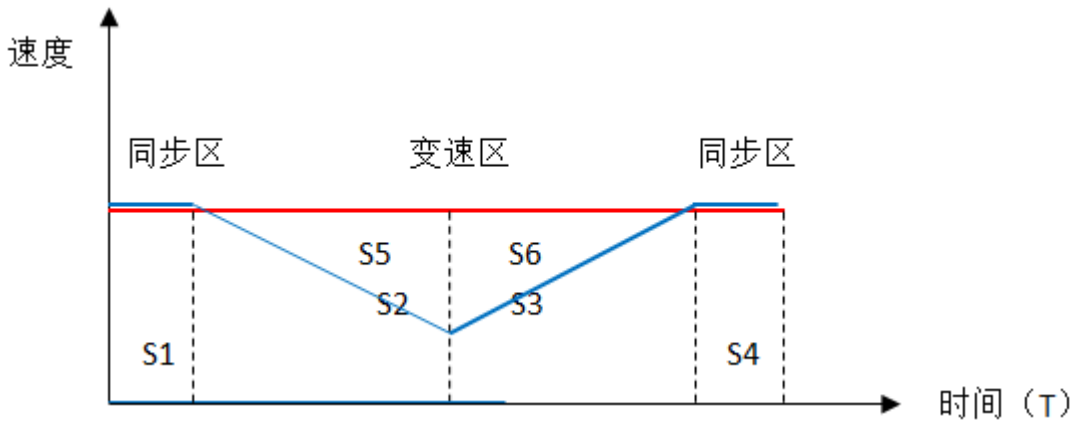
Arraympos 变量定义

运行任务，示波器显示：



示波器显示结果

(b) 情形二：中料剪切（L>=剪刀周长）L=800 mm



中料剪切示意图

红色区域面积(剪切料长/主轴位移)：S1+S2+S3+S4=800 mm；

蓝色区域面积(刀辊周长/从轴位移)：S1+(S2-S5)+(S3-S6)+S4=600；

S2 = S3 = (800-S1-S4)/2 = 325 mm；

S5+S6 = 800-600 = -200 mm；

S5+S6 区域为红线区域减蓝线区域。

故有：

dSuperimposeIn = dSuperimposeOut= S1= S4=75mm

dSlaveBaseDistance = dMasterDistance = L=800 mm

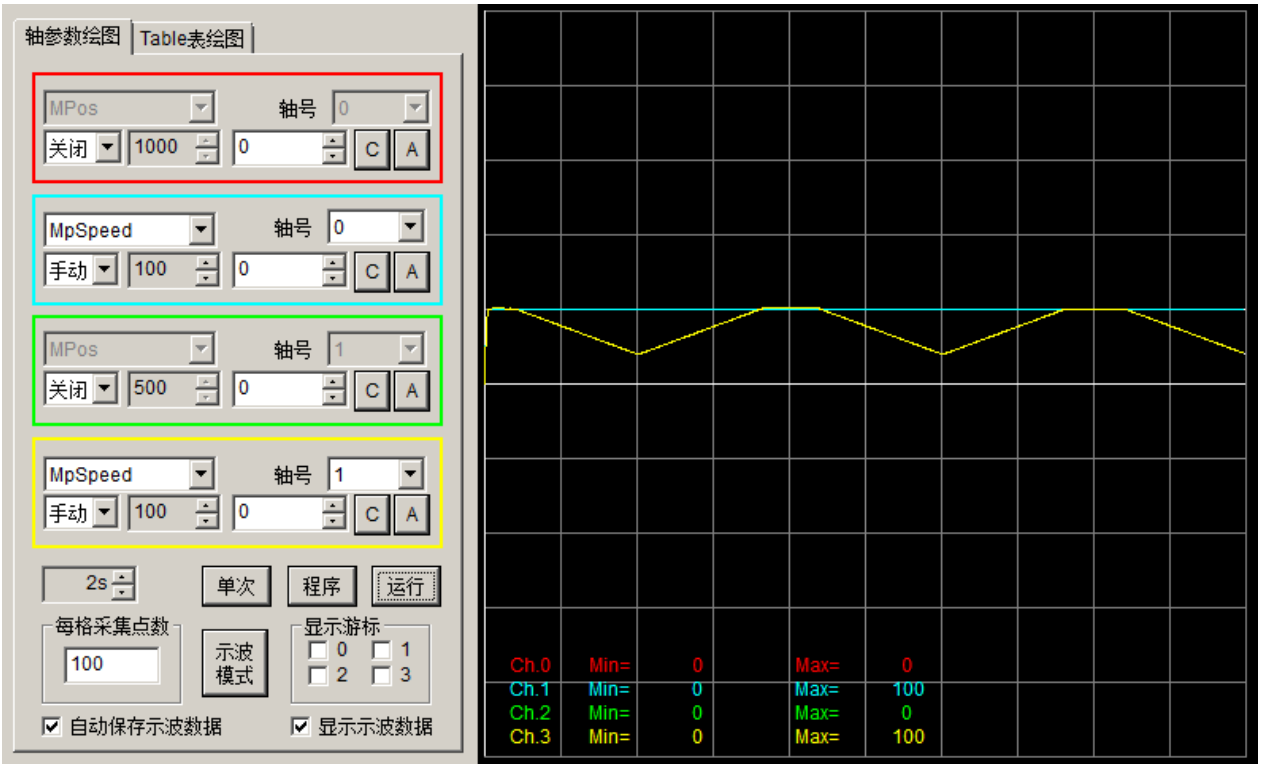
dSlaveSuperimpose = S5+S6 = -200mm

dSuperimposeAcc=dSuperimposeDec=S2= S3=325mm。

具体的凸轮运动程序：

```
FlexLinkReVal := HMC_ClcFlexLinkData(800, -200, 800, 75, 75, 325, 325, Pt_array, NumOfPos);

fff := HMC_TriggerScope();    (* 示波器使能 *)

camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0); (*投凸轮，循环立即启动*)

X_RESULT := HMC_Forward(0);    (* 主轴启动 *)
```

运行任务，示波器显示如下：



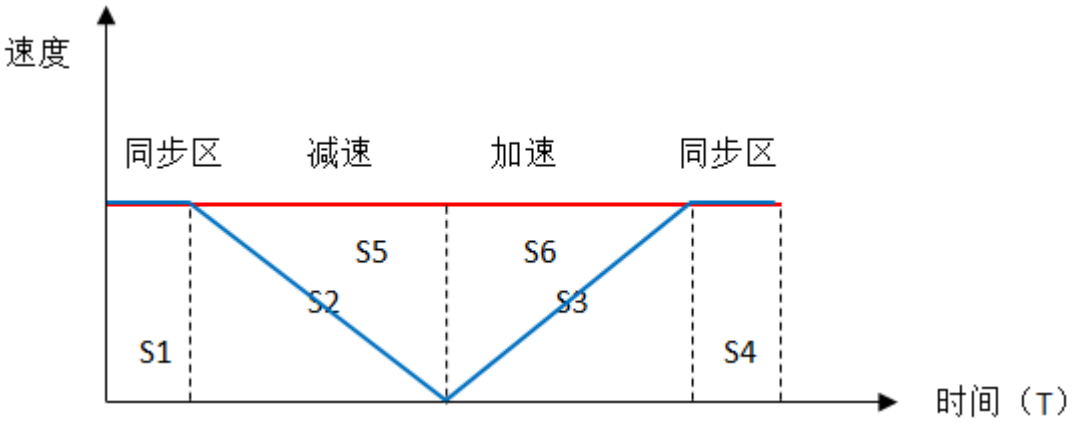
示波器显示结果

(c) 情形三：长料剪切

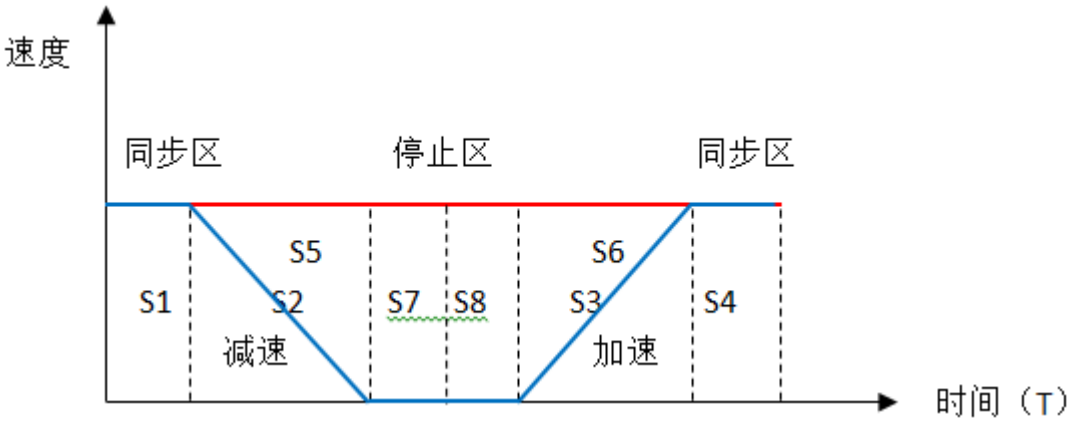
先计算刀辊刚好停止时（长料剪切与中料剪切分界点），剪切的长度(红色包围的面积)：

L0=S1+S2+S3+S4=(S1+S4)+2*(S5+S6)=(600-150)*2+150=1050 mm

当 L>1050 mm 时，刀辊出现停止等待区域，属长料剪切。



长料剪切与中料剪切分界点计算



长料剪切示意图

假设 L=1500 mm，那么刀辊停止等待时，主轴走过的距离:

S7+S8= L- L0=1500-1050=450 mm

故有：

dSuperimposeIn = dSuperimposeOut= S1= S4=75mm

dSlaveBaseDistance = dMasterDistance = L=1500 mm

dSlaveSuperimpose =刀辊周长- L = 600-1500 = -900mm

dSuperimposeAcc = dSuperimposeDec = (L - (S1+S4) - (S7+S8))/2 = (1500 – 150 – 450)/2

= 450

具体的运动程序如下：

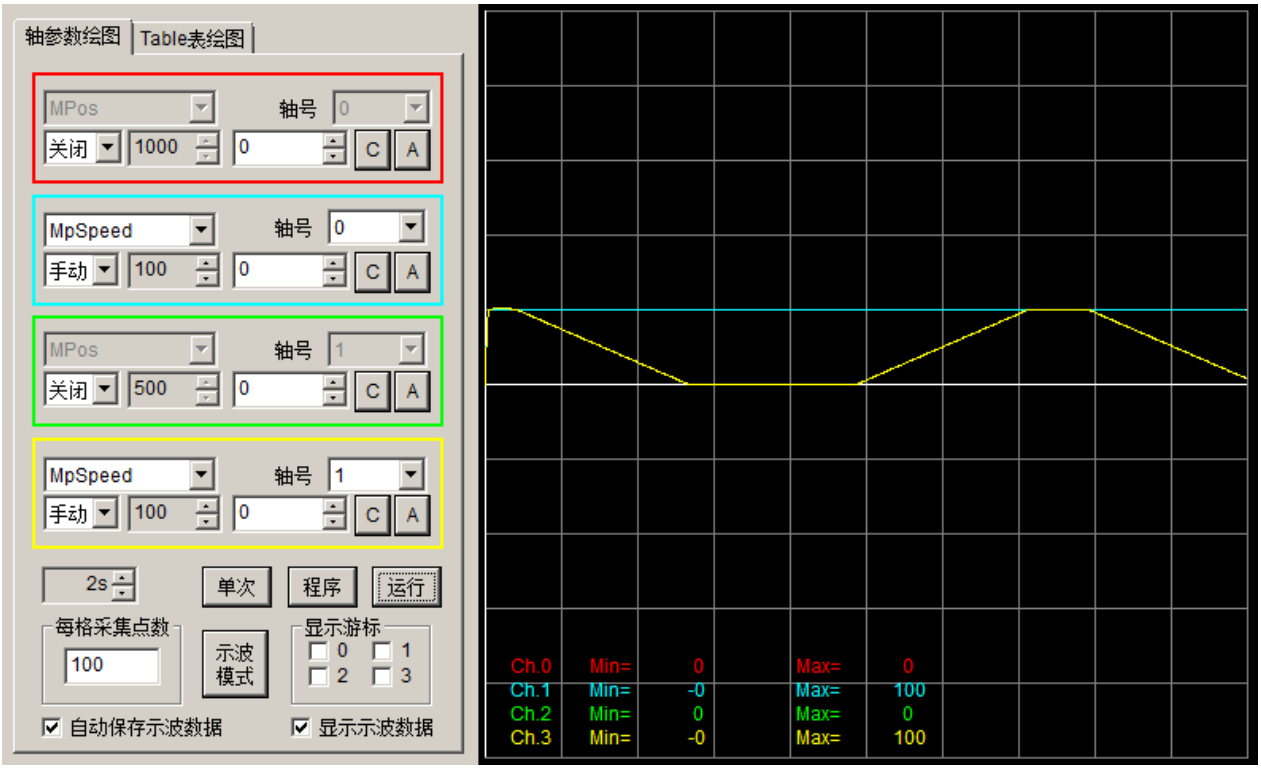
FlexLinkReVal := HMC_ClcFlexLinkData(1500, -900, 1500, 75, 75, 450, 450, Pt_array, NumOfPos);

fff := HMC_TriggerScope(); (* 示波器使能 *)

X_RESULT := HMC_Forward(0); (* 主轴启动 *)

camid := HMC_Cam(1, 0, Pt_array, NumOfPos, 1, 4, 0);

运行任务，示波器显示：



示波器显示结果

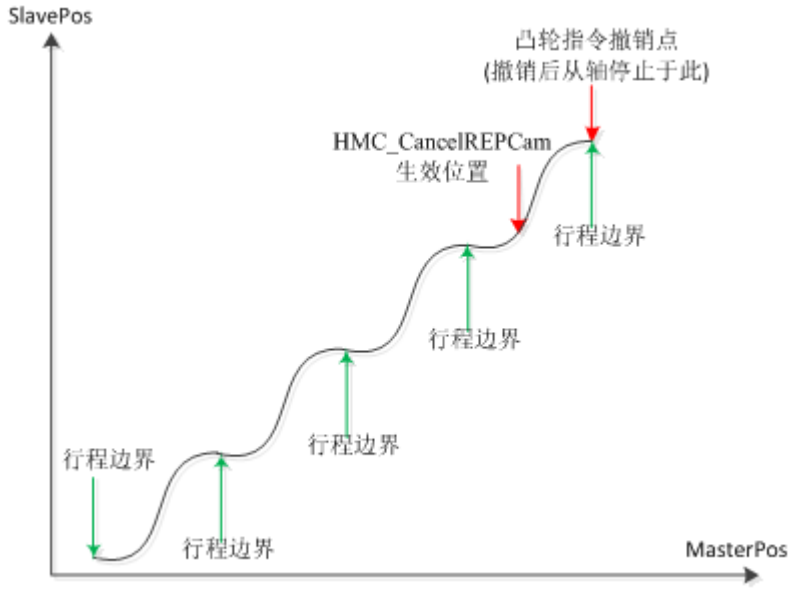
5.4.6.6 HMC_CancelREPCam（循环凸轮条件撤销控制）

5.4.6.6.1 功能

对正在运行的循环凸轮联动指令使用条件撤销功能，功能生效时，循环凸轮指令将在完成当前凸轮完整行程后撤销。

与 HMC_Cancel 指令直接撤销当前指令不同，HMC_CancelREPCam 会等待循环凸轮指令完成当前凸轮完整行程后才撤销当前凸轮指令，凸轮从轴会跟随主轴运行至凸轮主轴行程边界处，撤销后凸轮从轴准确停止在凸轮主轴行程边界处所对应的从轴位置处。

HMC_CancelREPCam 有两种模式：使能撤销与禁止撤销。HMC_CancelREPCam 生效后，会等待循环凸轮指令完成当前凸轮完整行程后才撤销当前凸轮指令，在凸轮主轴运行至行程边界之前，尚可撤回已生效的 HMC_CancelREPCam 指令，只需调用 HMC_CancelREPCam(ucSlaveAxis,0)即可。



循环凸轮执行 HMC_CancelREPCam 指令示意图

5.4.6.6.2 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号
ucCancelCycMode	USINT	撤销模式。 1：使能撤销 0：禁止撤销

5.4.6.6.3 指令类型

非轴运动缓存指令。

5.4.6.6.4 补充说明

HMC_CancelREPCam 生效后，凸轮从轴会跟随主轴运行至凸轮主轴行程边界处，然后直接停止，若此时凸轮从轴速度不为 0，则会造成较大冲击。请设计合适的主从轴位置曲线，确保在行程边界处从轴速度为 0 或较小。

- 注：可采用 HMC_ClcMoveLinkData、HMC_ClcFlexLinkData 指令自动生成主从轴位置数组来满足行程边界处从轴速度为 0 的要求。

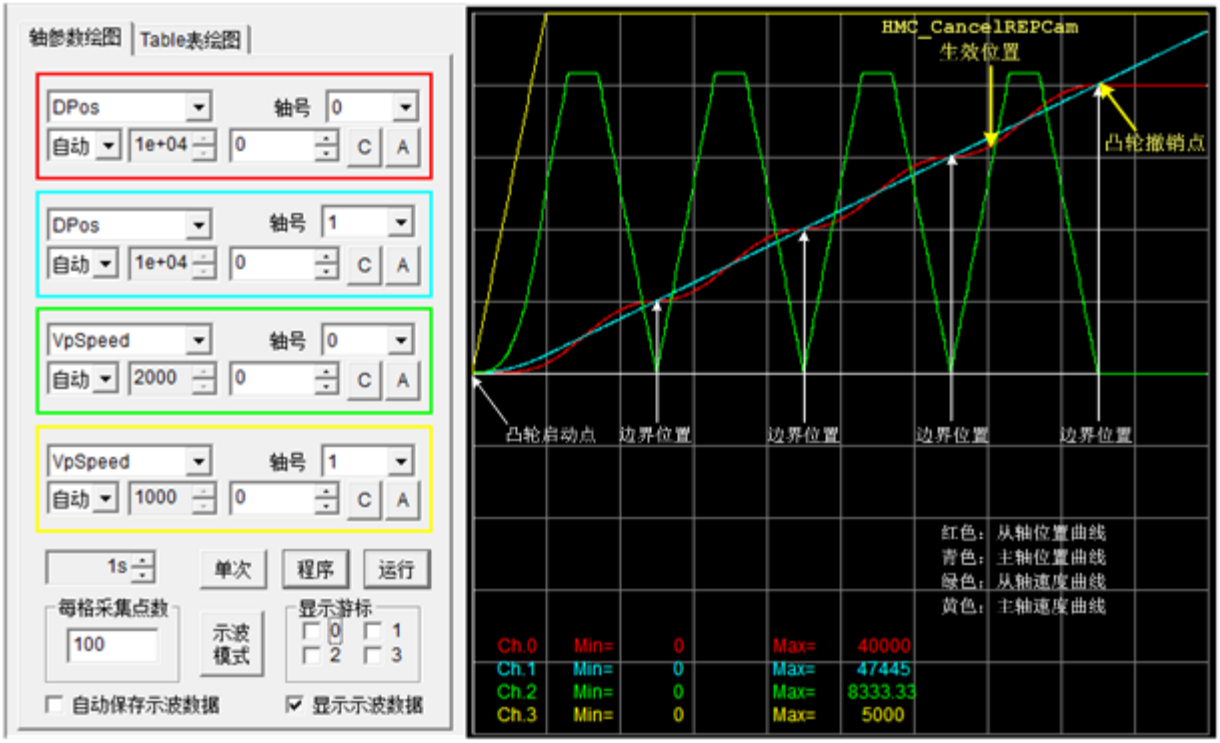
5.4.6.6.5 示例

使用 HMC_ClcMoveLinkData 生成主从轴位置数组，保存在用户自定义数组 arrPos 中，然后使用 arrPos 中的主从轴位置数组投放 HMC_Cam 凸轮指令，模式为循环凸轮，启动方式为立即启动。使用 Hmc_Move 驱动主轴运动，延时 7 秒后使用 HMC_CancelREPCam 撤销循环凸轮。

程序如下：

```
MasterAxisID := 1;      (*主轴轴号*)

SlaveAxisID := 0;       (*从轴轴号*)


HMC_TriggerScope();     (*程序触发示波器开始示波采样*)


HMC_ClcMoveLinkData(10000,10000,4000,4000,ADR(arrPos),100);    (*生成主从轴位置数组，保存在用户自定义数组 arrPos 中*)


HMC_Cam(SlaveAxisID,MasterAxisID,ADR(arrPos),100,1,4,0);        (*使用 arrPos 中的主从轴位置数组投放凸轮指令，模式为循环凸轮，启动方式为立即启动*)


Hmc_Move(MasterAxisID,200000);    (*驱动主轴运动*)


TimeDelay(7000);    (*延时 7 秒*)
```

HMC_CancelREPCam(SlaveAxisID,1); (*撤销循环凸轮*)



循环凸轮执行 HMC_CancelREPCam 指令运行结果图

5.4.6.7 HMC_MoveLink（追剪运动）

5.4.6.7.1 功能

该指令集成了凸轮和追剪运算功能，控制从轴跟随主轴按一定规律运动，可用在追剪场景中，快速生成追剪所需的动作流程，无需进行复杂的中间计算与编程。

该运动带有分离可变的加减速阶段。各个阶段参数设定方式可参考 HMC_ClcMoveLinkData（追剪运算），此处不再重复。

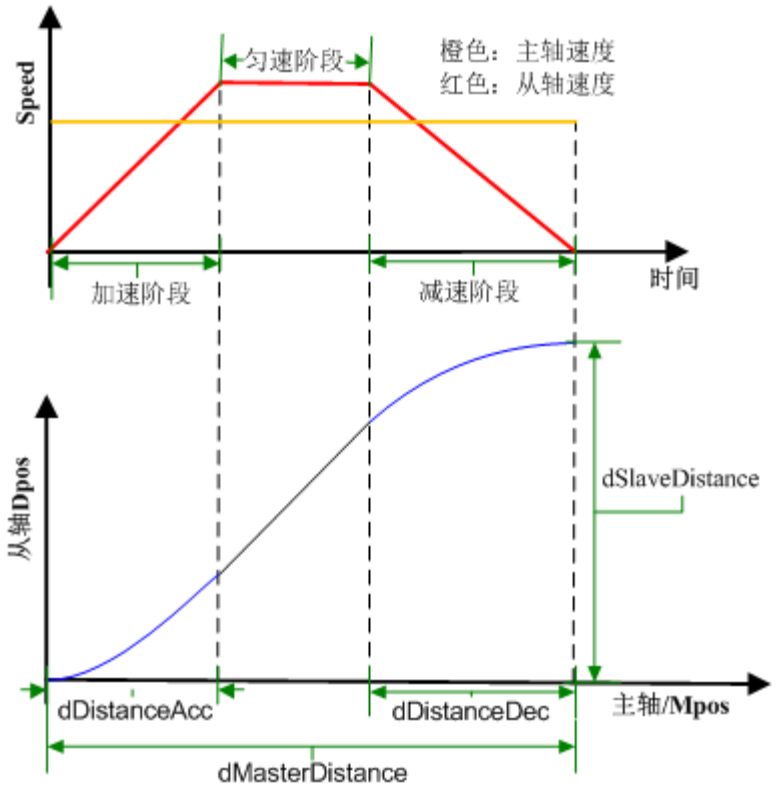
该功能块支持加减速阶段按梯形或 S 形曲线加减速两种方式。且不采用计算生成主从轴位置数组功能块，然后再线性插的方式控制从轴跟随主轴；而是根据设定好的加减速阶段距离、加减速方式等，选择梯形、S 形加减速模型，之后实时根据主轴位置计算从轴位置。这种方法可避免因主从轴位置数组选取点数少时，带来的速度阶梯。

加减速阶段规划方式：

（1） 梯形加减速 dSRatio=0

当 dSRatio=0 时，算法将对从轴的速度按照梯形曲线加减速规划，根据梯形模型、主轴位置实时计算本周期从轴的目标位置。

梯形加减速时，以主轴匀速运动为例，主、从轴的速度、位移曲线示意图如下



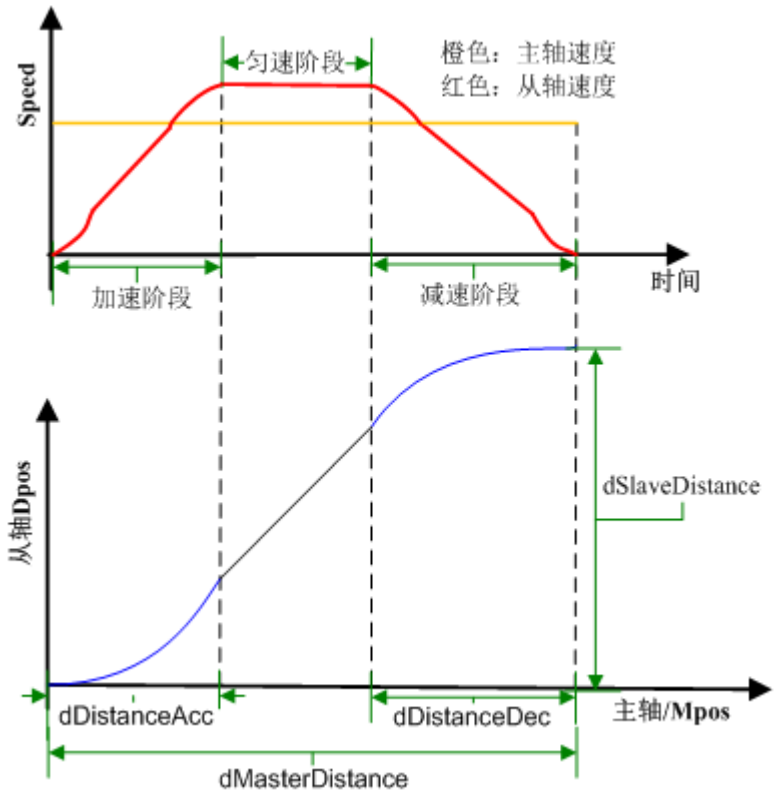
运动示意图

（2） S 型曲线加减速 dSRatio 在(0,1]

当 dSRatio 在(0,1]范围时，算法将对从轴的速度按照 S 形曲线加减速规划，根据 S 型模型、主轴位置实时计算本周期从轴的目标位置。

dSRatio 表示 S 形加减速在整个加速/减速过程所占比例，以加速阶段为例，加加速阶段所用时间占整个加速阶段时间的比例为 dSRatio/2，匀加速阶段所用时间占整个加速阶段时间的比例为 1-dSRatio，减加速阶段所用时间占整个加速阶段时间的比例为 dSRatio/2。

以 dSRatio=0.5，主轴匀速运动为例，主、从轴的速度、位移曲线示意图如下：



运动示意图

当设置 dSRatio=1 时，加减速阶段就没有匀加速/匀减速阶段，加速阶段的加加速和减加速比例分别为 0.5，减速阶段类似。

5.4.6.7.2 参数

输入参数	类型	参数描述
dSlaveDistance	LREAL	从轴运动距离
dMasterDistance	LREAL	主轴运动距离（必须为正）
dMasterDistAcc	LREAL	在从轴加速阶段主轴移动的正向距离，非负
dMasterDistDec	LREAL	在从轴减速阶段主轴移动的正向距离，非负
dSRatio	LREAL	从轴加速/减速阶段中 S 形加减速所占的比例 取值范围：[0,1]
ucSlaveAxis	USINT	从轴轴号
ucMasterAxis	USINT	主轴轴号
ucMode	USINT	0：单次立即启动 1：单次固定位置启动 2：单次事件启动

		3：单次 DI 启动 4：循环立即启动 5：循环固定位置启动 6：循环事件启动 7：循环 DI 启动 13：循环固定位置启动，主轴负向运动时启动 29：单次固定位置启动，主轴负向运动时启动
dTrigMes	LREAL	启动信息： 1：凸轮为位置启动时，该值表示主轴的启动位置 2：凸轮为事件触发启动时，表示某高速捕捉通道号 3：凸轮为 DI 为 1 触发启动时，表示 DI 通道（0~63）

5.4.6.7.3 指令类型

轴运动缓存指令。

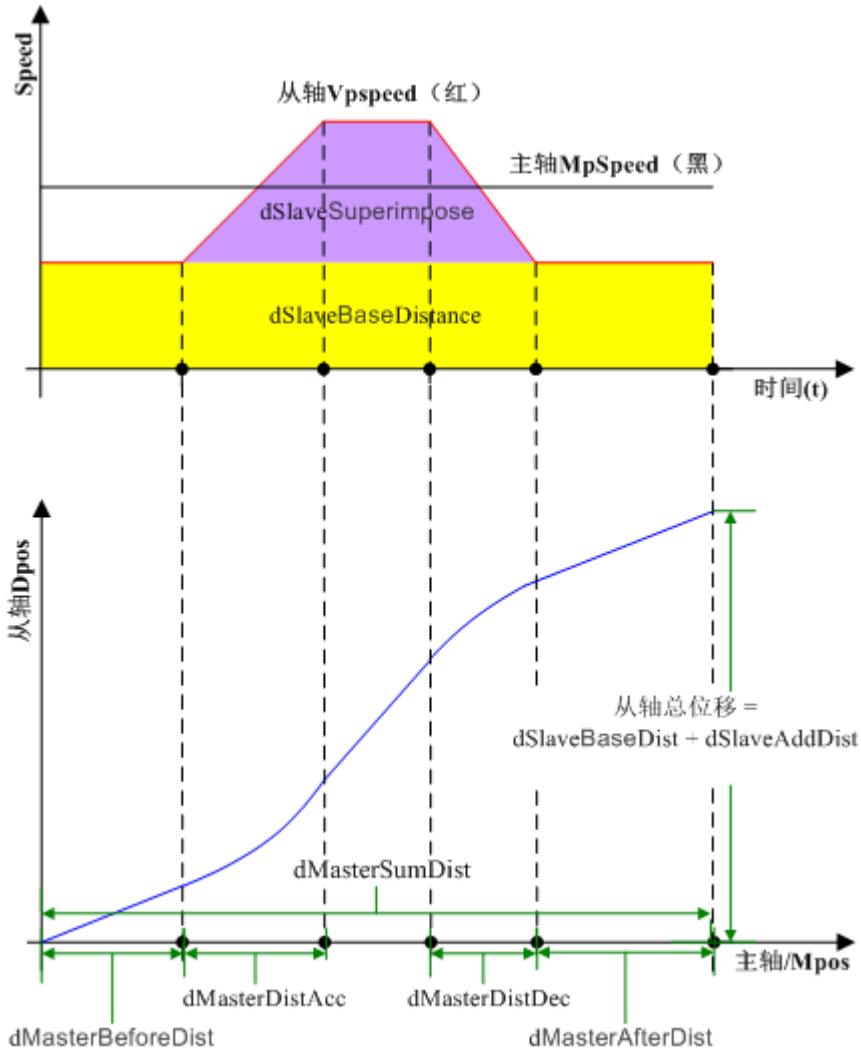
5.4.6.7.4 示例

功能块各个参数及算法方式同 HMC_ClcMoveLinkData，区别仅在于加速方式可选，因此应用举例可参考 HMC_ClcMoveLinkData。

5.4.6.8 HMC_MoveFlexLink（飞剪/旋切运动）

5.4.6.8.1 功能

完成具有主从轴位置及加减速规划的凸轮功能（dSRatio 设置 0 时为梯形加减速；S 形加减速所占的比例可通过 dSRatio 参数调整），可用在追剪、飞剪、旋切场景。



主/从轴示意图

5.4.6.8.2 参数

输入参数	类型	含义
dSlaveBaseDist	LREAL	从轴恒速跟随距离，正负均可
dSlaveAddDist	LREAL	从轴叠加运动总距离，正负均可
dMasterSumDist	LREAL	主轴运动总距离（必须为正）
dMasterBeforeDist	LREAL	叠加运动（dSlaveAddDist）开始时主轴已完成正向距离，非负
dMasterAfterDist	LREAL	叠加运动（dSlaveAddDist）结束时主轴剩余正向距离，非负
dMasterDistAcc	LREAL	从轴叠加运动加速阶段主轴移动的正向距离，非负
dMasterDistDec	LREAL	从轴叠加运动减速阶段主轴移动的正向距离，非负
dSRatio	LREAL	dSRatio=0 时，从轴为梯形加减速；$dSRatio \in (0,1]$ 时，从轴执行 S 形加减速，S 形加减速

		所占的比例为由 dSRatio 确定 取值范围：[0,1]
ucSlaveAxis	USINT	从轴轴号
ucMasterAxis	USINT	主轴轴号
ucMode	USINT	0：单次立即启动 1：单次固定位置启动 2：单次事件启动 3：单次 DI 启动 4：循环立即启动 5：循环固定位置启动 6：循环事件启动 7：循环 DI 启动 13：循环固定位置启动，主轴负向运动时启动 29：单次固定位置启动，主轴负向运动时启动
dTrigMes	LREAL	启动信息： （1）凸轮为位置启动时，该值表示主轴的启动位置 （2）凸轮为事件触发启动时，表示某高速捕捉通道号 （3）凸轮为 DI 为 1 触发启动时，表示 DI 通道（0~63）

5.4.6.8.3 指令类型

轴运动缓存指令。

5.4.6.8.4 补充说明

当 dSlaveBaseDist、dMasterBeforeDist、dMasterAfterDist 均为 0 时，即与 HMC_MoveLink 功能等价。

5.5 高级应用指令

5.5.1 运动叠加

5.5.1.1 HMC_Superimpose（叠加）

5.5.1.1.1 功能

该函数可实现两个主轴增量位移的简单叠加，叠加结果通过从轴输出。叠加功能生效后，设两个主轴增量位移分别为 ΔMpos_M0、ΔMpos_M1，则从轴 Dpos 增量位移为：

ΔMpos_S=ΔMpos_M0+ΔMpos_M1

使用叠加功能时，两个主轴执行正常运动控制指令，从轴跟随两个主轴位置的叠加结果作相应的联动动作。

5.5.1.1.2 参数

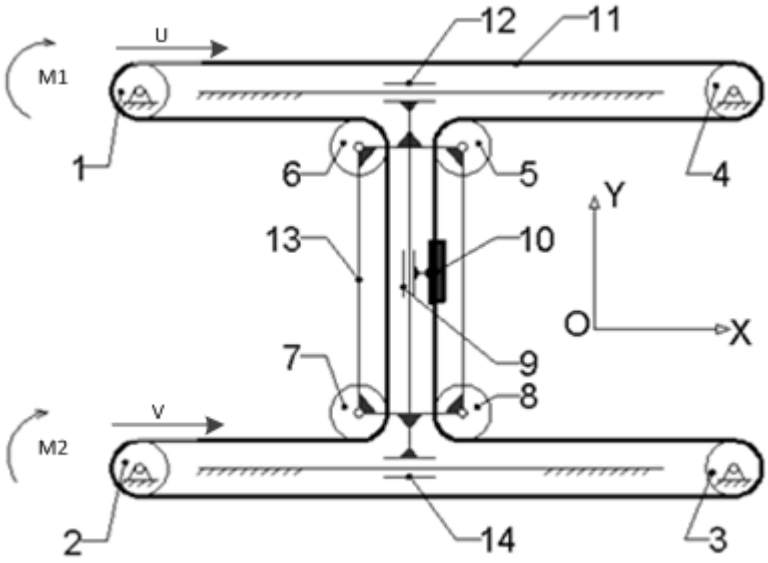
参数	声明	数据类型	说明
ucSlaveAxis	Input	USINT	从轴轴号
ucMasterAxis0	Input	USINT	第一个主轴轴号
ucMasterAxis1	Input	USINT	第二个主轴轴号
HMC_Superimpose	Output	UDINT	轴指令 ID：成功 0xFFFFFFFF：函数参数错误

5.5.1.1.3 指令类型

轴运动缓存指令。

5.5.1.1.4 示例

示例 1：H 形同步齿形带并联机构运动控制



H 形并联机构简图

- 1、2、3、4——静轮（固定于基座，选择 1、2 做驱动轮 M1、M2）
- 5、6、7、8——动轮
- 9、12、14——移动副
- 10——滑块
- 11——同步齿形带
- 13——动平台

设用户坐标空间为 X-Y，电机坐标空间为 U-V。通过机构学分析，可得出该机构的正反解运动学方程如下：

正解：

$$\begin{cases} \Delta x = (-\Delta u + \Delta v)/2 \\ \Delta y = -(\Delta u + \Delta v)/2 \end{cases}$$

逆解：

$$\begin{cases} \Delta u = -(\Delta x + \Delta y) \\ \Delta v = \Delta x - \Delta y \end{cases}$$

现欲实现在用户坐标空间 X-Y 中编写正常的用户程序，驱动电机在电机坐标空间 U-V 运行至相应的位

置。根据该机构运动学反解模型，用 HMC_Superimpose 与 HMC_Gear（电子齿轮）功能实现该并联机构控制的程序如下 (设 X-Y 对应主轴 0、1；U-V 对应从轴 2、3（电机轴）；10、11 号轴用来完成电子齿轮变换功能（实现 X、Y 轴换向）。)：

```
(**主轴设置为虚轴。**)
HMC_SetAxisType(Axis0.AxisID，0);
HMC_SetAxisType(Axis1.AxisID，0);

(**完成电子齿轮变换功能的 10、11 号轴设置为虚轴。**)
HMC_SetAxisType(Axis10.AxisID，0);
HMC_SetAxisType(Axis11.AxisID，0);

(**从轴（电机轴）设置为步进脉冲输出轴。**)
HMC_SetAxisType(Axis2.AxisID，10);
HMC_SetAxisType(Axis3.AxisID，10);

(**设定主轴运动参数**)
Axis0.Speed := 1000;    (*设定 0 轴运动速度*)
Axis0.Accel := 2000;    (*设定 0 轴加速度*)
Axis0.Decel := 2000;    (*设定 0 轴减速度*)
Axis1.Speed := 1000;    (*设定 1 轴运动速度*)
Axis1.Accel := 2000;    (*设定 1 轴加速度*)
Axis1.Decel := 2000;    (*设定 1 轴减速度*)

(*投送 HMC_Gear 指令，10 号轴为从轴，0 号轴为主轴，齿轮比为-1（X 轴换向）*)
```

HMC_Gear(Axis10.AxisID，Axis0.AxisID,-1,1);

(*投送 HMC_Gear 指令，11 号轴为从轴，1 号轴为主轴，齿轮比为-1（Y 轴换向）*)

HMC_Gear(Axis11.AxisID，Axis1.AxisID,-1,1);

(*投送 HMC_SuperImpose 指令，2 号轴为从轴，10、11 号轴为主轴。*)

HMC_Superimpose(Axis2.AxisID,Axis10.AxisID，Axis11.AxisID);

(*投送 HMC_SuperImpose 指令，3 号轴为从轴，0、11 号轴为主轴。*)

HMC_Superimpose(Axis3.AxisID,Axis0.AxisID，Axis11.AxisID);

(*之后可向主轴 0、1 投放所需的运动控制指令，驱动主轴运动，从轴 2、3 将驱动电机运动至相应位置。*)

有关该机构更为简单的控制实现方法，可参考 HMC_SuperImposeEx 中十字形并联机构运动控制示例。

示例 2：同步跟踪点胶

待点胶工件通过传送带传送，直角坐标机构负责完成工件表面的点胶，点胶轨迹为不规则几何曲线。为提高生产效率，要求在传送带保持正常运转的情况下，点胶头同步完成传送带上工件的点胶。

直角坐标机构 X、Y、Z 对应的轴号分别为 0、1、2，安装时保证直角坐标机构的 X 轴与传送带平行。轴 4 用于 X 方向叠加运动。同步动作开始时，通过使用 HMC_Superimpose 指令，把轴 4 的运动叠加在传送带轴（轴 3）的运动之上，叠加结果通过 X 轴（轴 0）电机输出。

通过运动叠加，在同步运动开始之后，用户只需对轴 4、1、2（对应 X、Y、Z）进行编程即可，点胶时只需按照点胶轨迹的原始几何曲线控制轴 4、1、2 进行插补运动，而无需考虑传送带的附加运动，X 轴方向的同步跟踪动作由叠加功能自动完成。

(*设定轴类型。Stepper 轴——直角坐标 X 轴:轴 0，Y 轴:轴 1，Z 轴:轴 2，传送带轴:轴 3。虚轴——用于 X 方向叠加运动的轴：轴 4*)

HMC_SetAxisType(Axis0.AxisID，10);

HMC_SetAxisType(Axis1.AxisID，10);

HMC_SetAxisType(Axis2.AxisID，10);

HMC_SetAxisType(Axis3.AxisID，10);

HMC_SetAxisType(Axis4.AxisID，0);　(*用于 X 方向叠加运动的轴设定为虚轴*)

(*设定各轴速度、加减速参数。(略)*)

(*检测工件是否到来，没有到来时等待(略)*)

(*工件到来，机械手首先加速到传送带相同的速度(略)*)

(*投送 HMC_Superimpose 指令，0 号轴(X 轴)为从轴，3 号轴（传送带轴）、4 号轴（圆弧插补轨迹计算轴）为主轴。*)

HMC_ Superimpose(Axis0.AxisID，Axis3.AxisID，Axis4.AxisID);

(*之后控制机械手运动时对应的 X、Y、Z 轴轴号分别为：4、1、2，无需考虑传送带的附加运动，X 轴方向的同步跟踪动作由叠加功能自动完成*)

(*首先移动机械手到工件涂胶起始点 (略)*)

(*按照点胶轨迹的原始几何曲线控制轴 4、1、2 进行插补运动(略)*)

5.5.1.2 HMC_SuperImposeEx（高级叠加）

5.5.1.2.1 功能

该函数可实现两个主轴增量位移的线性叠加，叠加结果通过从轴输出。叠加功能生效后，设两个主轴

增量位移分别为 ΔMpos_M0、ΔMpos_M1，则从轴 Dpos 增量位移为：

$$\triangle Dpos_S = \left(\frac{iRatioNumerator0}{iRatioDenominator0}\right) \times \triangle Mpos_{M0} + \left(\frac{iRatioNumerator1}{iRatioDenominator1}\right) \times \triangle Mpos_{M1}$$

支持用户动态修改叠加比例（详见示例 3：电子齿轮箱实现斜齿轮滚齿加工）。

5.5.1.2.2 参数

参数	声明	数据类型	说明
ucSlaveAxis	Input	USINT	从轴轴号
ucMasterAxis0	Input	USINT	第一个主轴轴号
ucMasterAxis1	Input	USINT	第二个主轴轴号
iRatioNumerator0	Input	DINT	第一个主轴的叠加比例的分子
iRatioDenominator0	Input	DINT	第一个主轴的叠加比例的分母
iRatioNumerator1	Input	DINT	第二个主轴的叠加比例的分子
iRatioDenominator1	Input	DINT	第二个主轴的叠加比例的分母
HMC_SuperImposeEx	Output	UDINT	轴指令 ID：成功 0xFFFFFFFF：函数参数错误

5.5.1.2.3 指令类型

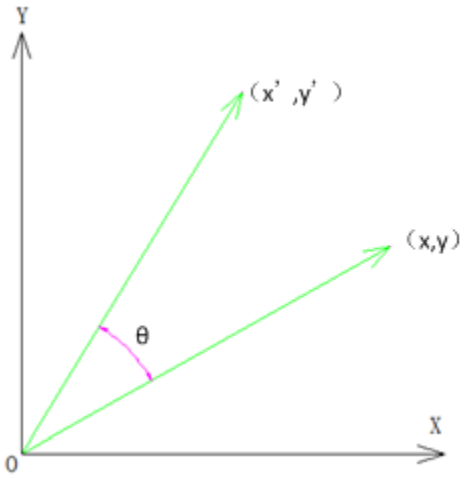
轴运动缓存指令。

5.5.1.2.4 补充说明

- 从数学关系上看，HMC_SuperImpose 和 HMC_Gear 均是 HMC_SuperImposeEx 的特例；
- 在该指令执行过程中，可以调用 HMC_SetSuperImposeRatio 函数动态修改叠加比例。

5.5.1.2.5 示例

示例 1：坐标变换



坐标变换示意图

轴 0、1 的合运动轨迹坐标点(x,y)与轴 2、3 的合运动轨迹坐标点(x',y')存在如下关系：(x,y)逆时针旋转 θ 得到(x',y')。由坐标变理论可知：

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

使用 HMC_SuperImposeEx 可实现该两组坐标的变换。程序如下(设 θ=45)°：

```
(**设定主轴运动参数**)
Axis0.Speed := 1000;    (*设定 0 轴运动速度*)
Axis0.Accel := 2000;    (*设定 0 轴加速度*)
Axis0.Decel := 2000;    (*设定 0 轴减速度*)
Axis1.Speed := 1000;    (*设定 1 轴运动速度*)
Axis1.Accel := 2000;    (*设定 1 轴加速度*)
Axis1.Decel := 2000;    (*设定 1 轴减速度*)

(*设置当前点为零点*)
HMC_DefPos(Axis0.AxisID,0);
HMC_DefPos(Axis1.AxisID,0);
HMC_DefPos(Axis2.AxisID,0);
```

HMC_DefPos(Axis3.AxisID,0);

(*投送 HMC_SuperImpose 指令，2 号轴为从轴，0、1 号轴为主轴。Sin45°=Cos45°≈707/1000*)

HMC_SuperImposeEx(Axis2.AxisID, Axis0.AxisID, Axis1.AxisID, 707, 1000, -707, 1000);

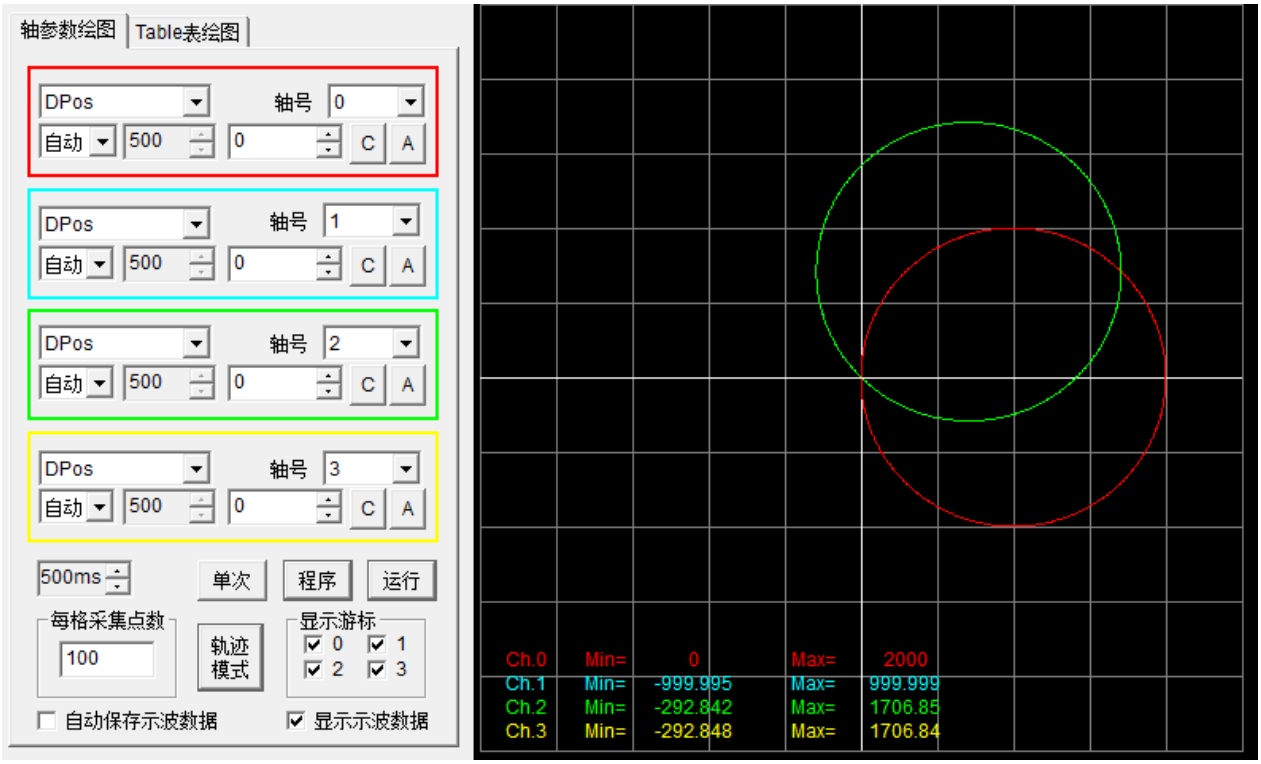
(*投送 HMC_SuperImpose 指令，3 号轴为从轴，0、1 号轴为主轴。*)

HMC_SuperImposeEx(Axis3.AxisID, Axis0.AxisID, Axis1.AxisID, 707, 1000, 707, 1000);

(*之后向主轴 0、1 投放圆弧插补指令指令，驱动主轴运动（红色），从轴 2、3 将输出变换后的结果（绿色）。*)

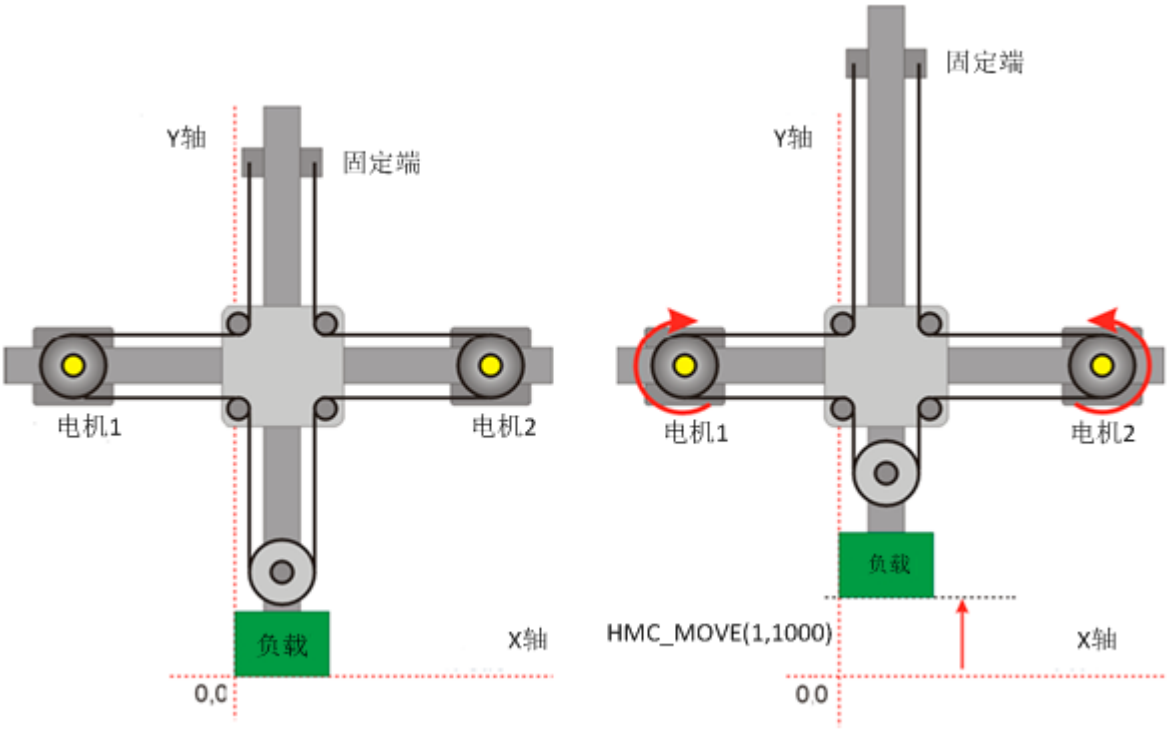
HMC_MoveCircle(10, Axis0.AxisID, Axis1.AxisID, 1, 0, 0, 1000, 0);

AT 中执行结果如下：



HMC_SuperImposeEx 实现坐标变换

示例 2：十字形同步齿形带并联机构运动控制



十字形同步齿形带并联机构

设用户坐标空间为 X-Y，电机坐标空间为 U-V。通过机构学分析，可得出该机构的正反解运动学方程如下：

正解：

$$\begin{cases}\Delta x = 0.5 * \Delta u + 0.5 * \Delta v \\ \Delta y = 0.5 * \Delta u - 0.5 * \Delta v\end{cases}$$

反解：

$$\begin{cases}\Delta u = \Delta x + \Delta y \\ \Delta v = \Delta x - \Delta y\end{cases}$$

现欲实现在用户坐标空间 X-Y 中编写正常的用户程序，驱动电机在电机在电机坐标空间 U-V 运行至相应的位置。用 HMC_SuperimposeEx 实现的程序如下(设 X-Y 对应主轴 0、1，U-V 对应从轴 2、3)：

```
(**主轴设置为虚轴。**)
HMC_SetAxisType(Axis0.AxisID, 0);
HMC_SetAxisType(Axis1.AxisID, 0);
(**从轴设置为步进脉冲输出轴。**)
HMC_SetAxisType(Axis2.AxisID, 10);
```

```
HMC_SetAxisType(Axis3.AxisID, 10);

(**设定主轴运动参数**)
Axis0.Speed := 1000;    (*设定 0 轴运动速度*)
Axis0.Accel := 2000;    (*设定 0 轴加速度*)
Axis0.Decel := 2000;    (*设定 0 轴减速度*)
Axis1.Speed := 1000;    (*设定 1 轴运动速度*)
Axis1.Accel := 2000;    (*设定 1 轴加速度*)
Axis1.Decel := 2000;    (*设定 1 轴减速度*)

(*投送 HMC_SuperImpose 指令，2 号轴为从轴，0、1 号轴为主轴。*)
HMC_SuperimposeEx(Axis2.AxisID, Axis0.AxisID, Axis1.AxisID, 1, 1, 1, 1);

(*投送 HMC_SuperImpose 指令，3 号轴为从轴，0、1 号轴为主轴。*)
HMC_SuperimposeEx(Axis3.AxisID, Axis0.AxisID, Axis1.AxisID, 1, 1, -1, 1);

(*之后可向主轴 0、1 投放所需的运动控制指令，驱动主轴运动，从轴 2、3 将驱动电机运动至相应位置。*)
```

示例 3：电子齿轮箱实现斜齿轮滚齿加工

数控滚齿机和插齿机以电子齿轮箱取代了原始的机械式内联传用动链，与常规的机械内联传动链相比，采用电子齿轮箱的数控齿轮加工机床传动链缩短，提高了传动刚度和传动精度；各向运动轴既可单独动作也可多轴联动，加工过程由程序控制，运动灵活、定位准确、精度高、效率高，能加工普通齿轮加工机床无法加工的零件。

HMC_SuperImposeEx 指令可实现数控滚齿机和插齿机所需的电子齿轮箱功能，以下以滚齿机为例分析。

首先针对滚齿机床各轴运动特征，建立无轴传动电子齿轮箱控制的数学模型。滚齿机床各运动轴为：

主运动轴 B 轴（滚刀主轴）、工件轴（C 轴）、X 轴（径向进给轴）、Y 轴（窜刀轴）和 Z 轴（轴向进给轴）。

设滚刀（B 轴）头数为 z_B，转速为 n_B；工件（C 轴）齿数为 z_C。在用“差动法”加工斜齿轮或采用“对角滚切法”加工齿轮时，机床工件主轴与机床刀具主轴之间不仅要有准确的速比关系，在滚刀轴有 Z 向进给或 Y 向连续窜刀运动时，工件主轴还要完成对 Z 轴或 Y 轴的准确跟随，使滚刀与工件之间保持严格的展成运动。选取工件轴作为电子齿轮箱的从动轴，其运动速度可描述为:

$$n_C = K_B \frac{z_B}{z_C} n_B + K_Z \frac{\sin\beta}{\pi m_n z_C} v_Z + K_Y \frac{\cos\lambda}{\pi m_n z_C} v_Y$$

式中，v_Y、v_Z 分别为 Y 轴、Z 轴的移动速度，mm／min；β 为斜齿轮的螺旋角；λ 为刀具的安装角；m_n 为齿轮的法面模数； K_B、K_Z、K_Y 为系数。

由上式可知，滚齿加工时，C 轴不仅与 B 轴速度有关，Y 轴和 Z 轴的运动也会产生 C 轴的附加运动。假设 Y 轴与 Z 轴均采用伺服电机+滚珠丝杠的传动方式实现，则上述公式可简化为(最简分数形式):

$$n_C = \frac{i_{Bn}}{i_{Bd}} n_B + \frac{i_{Zn}}{i_{Zd}} n_Z + \frac{i_{Yn}}{i_{Yd}} n_Y$$

n_B、n_C、n_Y、n_Z 分别为 B、C、Y、Z 轴的驱动电机转速。

设 B、C、X、Y、Z 轴的驱动电机对应的轴号分别为 0、1、2、3、4，用 HMC_SuperImposeEx 指令实现该运动关系的示例程序如下（ST 语言）：

```
(**设置 B、C、X、Y、Z 轴为闭环伺服轴。**)
HMC_SetAxisType(Axis0.AxisID, 20);
HMC_SetAxisType(Axis1.AxisID, 20);
HMC_SetAxisType(Axis2.AxisID, 20);
HMC_SetAxisType(Axis3.AxisID, 20);
HMC_SetAxisType(Axis4.AxisID, 20);

(**用于中间运算的轴 5 设置为虚轴。**)
HMC_SetAxisType(Axis5.AxisID, 0);

(**设定 B、C、X、Y、Z 轴运动参数（略）**)
```

(**调整设备至加工起始位置（略）**)

(*投送 HMC_SuperImpose 指令，5 号轴为从轴，B 轴（0）、Z 轴（4）为主轴。*)

HMC_SuperimposeEx(Axis5.AxisID, Axis0.AxisID, Axis4.AxisID, iBn, iBd, iZn, iZd);

(*投送 HMC_SuperImpose 指令，C 轴（1）为从轴，Y 轴（3）、5 号轴为主轴。*)

HMC_SuperimposeEx(Axis1.AxisID, Axis3AxisID, Axis5.AxisID, iYn, iYd, 1, 1);

(*通过 X 轴调整合适的径向进给量，然后驱动 B/Y/Z 轴运动，从轴 C 将按上述关系运动，实现斜齿轮范成加工*)

5.5.1.3 HMC_SetSuperimposeRatio（设置叠加比例）

5.5.1.3.1 功能

设置叠加比例，关于叠加详见 HMC_Superimpose（叠加）、HMC_SuperImposeEx（高级叠加）。在 HMC_Superimpose 或 HMC_SuperimposeEx 指令启动后，可以实时调用该指令，可以动态改变运转中的叠加比例，从而达到动态改变叠加比例的效果。

该指令只允许当前正在运行 HMC_Superimpose 或 HMC_SuperimposeEx 指令时设置。

5.5.1.3.2 参数

参数	声明	数据类型	说明
ucSlaveAxis	Input	USINT	从轴轴号
iRatioNumerator0	Input	DINT	第一个主轴的叠加比例的分子
iRatioDenominator0	Input	DINT	第一个主轴的叠加比例的分母
iRatioNumerator1	Input	DINT	第二个主轴的叠加比例的分子
iRatioDenominator1	Input	DINT	第二个主轴的叠加比例的分母
HMC_SetSuperimposeRatio	Output	UDINT	0：成功

			其他值：错误码

5.5.1.3.3 指令类型

非轴运动缓存指令。

5.5.1.3.4 示例

示例程序如下：

```
Axis0.Speed := 1000;    (*设定运动速度*)
Axis0.Accel := 2000;    (*设定加速度*)
Axis0.Decel := 2000;    (*设定减速度*)
Axis1.Speed := 1000;    (*设定运动速度*)
Axis1.Accel := 2000;    (*设定加速度*)
Axis1.Decel := 2000;    (*设定减速度*)

(*投送 HMC_SuperImpose 指令，3 号轴为从轴，0、1 号轴为主轴。*)
HMC_SuperimposeEx(Axis3.AxisID, Axis0.AxisID, Axis1.AxisID, 1, 1, 1, 1);

Cmd_move0 := HMC_Move(Axis0.AxisID, 1000);      (*向 0 号轴投送 HMC_Move 指令,运动增量距离 1000*)
Cmd_move1 := HMC_Move(Axis1.AxisID, 2000);      (*向 0 号轴投送 HMC_Move 指令,运动增量距离 2000*)
HMC_WaitCmdFinish(Cmd_move0);    (* 等待指令完成 *)
HMC_WaitCmdFinish(Cmd_move1);    (* 等待指令完成 *)

HMC_ SetSuperimposeRatio(Axis3.AxisID, 1, 1, -1, 1);   (*改变叠加比例*)
```

```
HMC_Move(Axis0.AxisID, 1000);  (*向 0 号轴投送 HMC_Move 指令,运动增量距离 1000*)
HMC_Move(Axis1.AxisID, -1000);  (*向 0 号轴投送 HMC_Move 指令,运动增量距离-1000*)
```

5.5.1.4 HMC_GetSuperimposeMaster1（获取叠加主轴 1）

5.5.1.4.1 功能

获取叠加的第一个主轴。

5.5.1.4.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	从轴轴号
HMC_GetSuperimposeMaster1	Output	INT	叠加的第一个主轴:成功 -1:当前未进行叠加运动或函数参数错误

5.5.1.4.3 指令类型

非轴运动缓存指令。

5.5.1.5 HMC_GetSuperimposeMaster2（获取叠加主轴 2）

5.5.1.5.1 功能

获取叠加的第二个主轴。

5.5.1.5.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	从轴轴号
HMC_GetSuperimposeMaster2	Output	INT	叠加的第二个主轴：成功

			-1：当前未进行叠加运动或函数参数错误

5.5.1.5.3 指令类型

非轴运动缓存指令。

5.5.1.6 HMC_GetSuperimposeRatio1（获取叠加主轴 1 比例）

5.5.1.6.1 功能

获取叠加的第一个主轴比例。

5.5.1.6.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	从轴轴号
HMC_GetSuperimposeRatio1	Output	LREAL	叠加的第一个主轴比例：成功 0xFFFFFFFF：当前未进行叠加运动或函数参数错误

5.5.1.6.3 指令类型

非轴运动缓存指令。

5.5.1.7 HMC_GetSuperimposeRatio2（获取叠加主轴 2 比例）

5.5.1.7.1 功能

获取叠加的第二个主轴比例。

5.5.1.7.2 参数

参数	声明	数据类型	说明

ucAxisID	Input	USINT	从轴轴号
HMC_GetSuperimposeRatio2	Output	LRAEL	叠加的第二个主轴比例：成功 0xFFFFFFFF：当前未进行叠加运动或函数参数错误

5.5.1.7.3 指令类型

非轴运动缓存指令。

5.5.1.8 HMC_SetAddAxis（设定和运动轴）

5.5.1.8.1 功能

该指令用于完成二轴运动求和。指令调用成功后，在后续的运动过程中，有：基础轴位移=基础轴自身指令位移+叠加轴位移。

上式中位移均为 Dpos 增量。

当 ucAddAxisID=255 时，表示取消 ucBaseAxisID 轴与其他轴的和运动。可以通过 HMC_GetAddAxis(ucBaseAxisID)指令查询叠加轴轴号。

5.5.1.8.2 参数

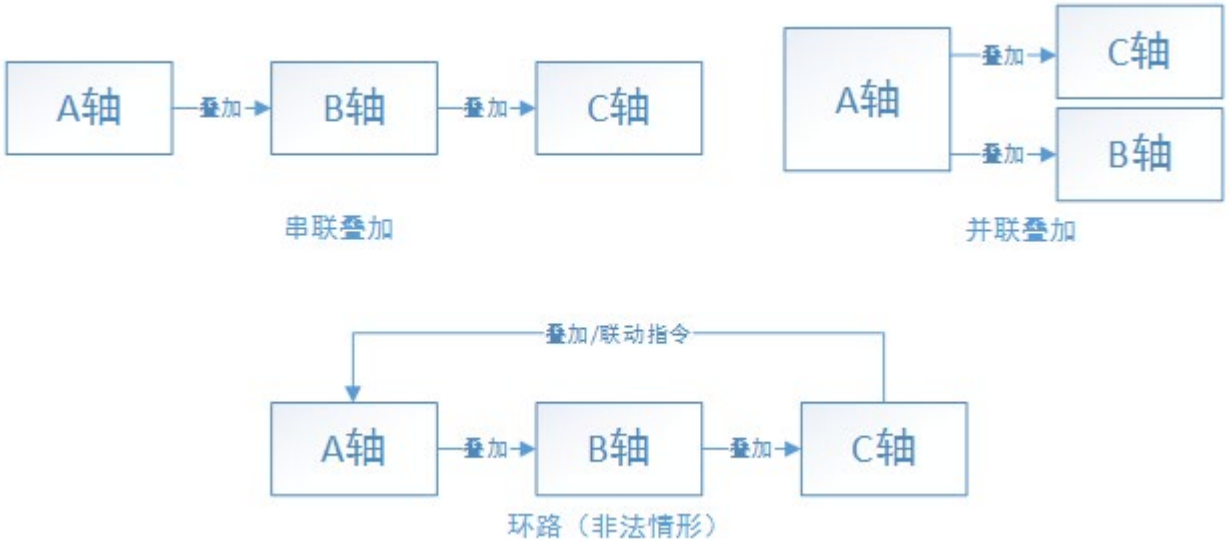
参数	声明	数据类型	说明
ucBaseAxisID	Input	USINT	基础轴轴号，0~63
ucAddAxisID	Input	USINT	叠加轴轴号（有效关联轴号：0~63；255 表示取消关联。）
HMC_SetAddAxis	Output	UDINT	成功：0 错误：错误码

5.5.1.8.3 指令类型

非轴运动缓存指令。

5.5.1.8.4 补充说明

（1） 叠加指令使用时不可形成环路，否则会导致轴速度发散失控。下图依此为正常的串/并联开路情形、异常的环路情形。因异常环路的多样性，检测比较困难，用户使用时需自行保证。



（2） 指令调用成功后，基础轴的 Dpos/Mpos 反映的是轴实际的位置，MpSpeed 反映的是实际叠加后的速度，但 VpSpeed 为基础轴自身指令速度，而非叠加后的值。

5.5.1.9 HMC_GetAddAxis（获取和运动叠加轴）

5.5.1.9.1 功能

用于获取和运动叠加轴轴号，255 表示未关联任何叠加轴。

5.5.1.9.2 参数

参数	声明	数据类型	说明
ucBaseAxisID	Input/Output	USINT	基础轴轴号，0~63
HMC_GetAddAxis	Output	DINT	叠加轴轴号：成功 255：未关联任何叠加轴

5.5.1.9.3 指令类型

非轴运动缓存指令。

5.5.2 低速到位输出

低速到位输出与高速到位输出功能类似，但是位置比较的精准度低于高速到位输出功能，位置比较精度与伺服周期和轴的速度都有关系，适用于对控制精度要求不高的应用场景。

低速到位输出支持更丰富的子功能：

- 输入

  支持所有轴类型，包括总线、虚轴等；

  信号源比较支持目标位置 Dpos 和实际反馈位置 Mpos。

- 输出

  可通过任意类型 DO 输出，包括高速、低速、虚拟 DO；

  信号输出方式支持电平翻转和指定宽度的脉冲输出。

- 位置比较

  比较位置序列随意设置，但必须符合轴的位置变换，否则会出现一直等待某点判断中；

  支持一维位置比较、二维位置比较。

5.5.2.1 HMC_NormalSwitch1Dconfig（设置一维低速输出）

5.5.2.1.1 功能

配置一维低速输出。按照设置的位置比较序列，与轴的位置进行比较，当轴的位置在本周期经过目标比较位置时，则激活输出。

支持目标位置序列不严格单调。按目标位置序列数组从前到后比较，若当前待比较位置一直不能触发，则一直处于判断该目标点状态，不会对后续位置进行比较。

5.5.2.1.2 参数

参数	声明	数据类型	说明
ucChIID	Input	USINT	普通到位输出通道号，0~7 通道号与 HMC_NormalSwitch2DConfig 共用
ucOutputID	Input	UDINT	输出位置选择，0～65535
ucTrigAxis	Input	USINT	采集轴，switch 比较该轴与目标位置值决定输出电平
ucTrigPosType	Input	USINT	位置比较源方式： 0：比较采集轴的 Mpos 1：比较采集轴的 Dpos
ucOutputType	Input	USINT	输出方式： 0：电平翻转 1：可设脉宽的脉冲
bSwitchValue	Input	USINT	当 ucOutputType=0 时，表示第一个位置输出电平值：0 或 1 当 ucOutputType=1 时，表示输出脉冲电平值：0 或 1
uiPluseWidth	Input	UDINT	当 ucOutputType=1 时，该参数有效，表示脉冲宽度，单位 ms（设置值需≥1 个伺服周期，分辨率正负 1 个伺服周期。）
pPos	Input	POINTER TO LREAL	目标位置存放地址，该数组存放目标位置设定值
uiPosNum	Input	UDINT	目标 Mpos 序列的个数
HMC_NormalSwitch1DConfig	Output	UDINT	0:配置成功 其他：错误码

5.5.2.1.3 指令类型

非轴运动缓存指令。

5.5.2.2 HMC_NormalSwitch2DConfig（设置二维低速输出）

5.5.2.2.1 功能

配置二维低速输出，按照设置的位置比较序列进行比较，当两轴位置对应的二维坐标进入目标位置相等区域并选取较优位置，激活输出。具体的位置判定方法如下：

- 当前点与目标点的距离小于等于位置比较阈值 dThreshold，认为进入目标位置相等区域，是触发输出的必要条件；

- 进入相等区域后，尽量选择在距离目标点距离最近处输出；通过连续比较与目标点距离，确认逼近还是远离，从而寻找最优时机；
- 当比较信号源为轴的实际反馈位置时，需做防抖处理，使用防抖动阈值 dDisturbValue 参数；
- 只要进入过相等区域就一定会触发输出。

综合以上多项处理，输出的时机不一定在距离目标点最近处，但在其附近，这取决于用户设定的判断阈值和轴的速度等多项因素。

5.5.2.2.2 参数

参数	声明	数据类型	说明
ucChIID	Input	USINT	普通到位输出通道号，0~7 通道号与 HMC_NormalSwitch2DConfig 共用
ucOutputID	Input	UDINT	输出位置选择，0～65535
ucTrigAxisX	Input	USINT	采集轴 X，switch 比较该轴与 X 目标位置值决定输出电平
ucTrigAxisY	Input	USINT	采集轴 Y，switch 比较该轴与 Y 目标位置值决定输出电平
ucTrigPosType	Input	USINT	位置比较源方式： 0：比较采集轴的 Mpos 1：比较采集轴的 Dpos
ucOutputType	Input	USINT	输出方式： 0：电平翻转 1：可设脉宽的脉冲
bSwitchValue	Input	USINT	当 ucOutputType=0 时，表示第一个位置输出电平值：0 或 1　ucOutputType=1 时，表示输出脉冲电平值：0 或 1
uiPluseWidth	Input	UDINT	当 ucOutputType=1 时，该参数有效，表示脉冲宽度，单位 ms（设置值需≥1 个伺服周期，分辨率正负 1 个伺服周期。）
pPosX	Input	POINTER TO LREAL	X 目标位置存放地址，该数组存放目标位置设定值。当为二维位置比较时，该数组描述 X 轴位置值
pPosY	Input	POINTER TO LREAL	Y 目标位置存放地址，当为二维位置比较时该参数有效，该数组描述 Y 轴位置值，与 pMposX 共同描述二维比较目标位置
uiPosNum	Input	UDINT	目标 Mpos 序列的个数
dThreshold	Input	LREAL	位置比较偏差阈值，判断是否到达目标位置相等区域(≥0)
dDisturbValue	Input	LREAL	防抖动阈值。位置比较源为 Mpos，且反馈位置来自物理输入时才有效。反馈位置与目标位置进行比较时，用来进行防抖动处理。（≥0）

HMC_NormalSwitch1DConfig	Output	UDINT	0:配置成功，其他：错误码

5.5.2.2.3 指令类型

非轴运动缓存指令。

5.5.2.3 HMC_GetNormalSwitchNum（获取已输出数目）

5.5.2.3.1 功能

获取指定 ID 的低速到位输出已执行的段数。

5.5.2.3.2 参数

参数	声明	数据类型	说明
ucChIID	Input	USINT	低速输出通道号，0~7
HMC_GetNormalSwitchNum	Output	UDINT	已完成执行的段数

5.5.2.3.3 指令类型

非轴运动缓存指令。

5.5.2.4 HMC_GetNormalSwitchState（获取低速输出状态）

5.5.2.4.1 功能

获取指定 ID 的低速输出的当前状态。

5.5.2.4.2 参数

参数	声明	数据类型	说明
ucChIID	Input	USINT	低速输出通道号，0~7

HMC_GetNormalSwitchState	Output	UDINT	返回值 0：当前通道正在工作 返回值 3：当前通道空闲，表示已完成或未配置

5.5.2.4.3 指令类型

非轴运动缓存指令。

5.5.2.5 HMC_CloseNormalSwitch（关闭低速输出）

5.5.2.5.1 功能

关闭该 ID 的低速到位输出。

5.5.2.5.2 参数

参数	声明	数据类型	说明
ucChIID	Input	USINT	低速输出通道号，0~7
HMC_CloseNormalSwitch	Output	UDINT	返回值 0：关闭成功 其他值：错误码

5.5.2.5.3 指令类型

非轴运动缓存指令。

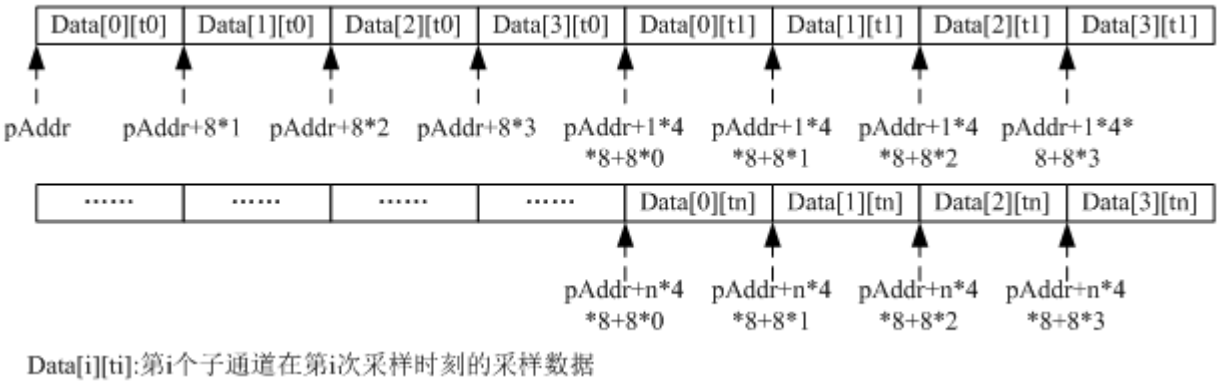
5.5.3 示波器功能指令

5.5.3.1 HMC_SetSampleVar（设置采样变量）

5.5.3.1.1 功能

数据采样功能可以对系统的轴参数、I/O 数据、中间/系统变量等进行数据采集，按照配置的数据采样间隔时间进行采样，将采样获得的实时数据记录存放至用户指定区域，该指定区域必须位于 M 区内。

无论待采样数据其自身数据类型是整形还是浮点，采样数据最终都按照双精度浮点 LREAL 类型存放至用户指定区域，一个通道的四个子通道数据按照时间先后顺序在用户指定区域存放顺序如下：



采样存放示意图

可以根据以上存储格式解析查看记录数据。当仅使用部分子通道时，如只使用 0、1 通道，则 1 号子通道数据后紧促排列存储 0 号子通道下一采样时刻的值，中间无空闲。

本功能块作用是配置待采样数据来源，最多支持采集 4 个变量。

采样功能须结合系统其它几个采样相关功能块一起使用。先调 SetSampleVar 配置待采样数据，再调用 SetSamplePara 设置存储区域、采样间隔时间、采样总点数，最后还需通过 HMC_TriggerSample 触发采样执行。采样执行过程中，可通过 HMC_ GetSampleNum 查看当前已经完成的采样数目，采样执行完成后，可通过 HMC_ GetSampleData 查看某一子通道指定采样时刻序号的采样值。其它采样相关功能块请查看对应章节。

5.5.3.1.2 参数

参数	声明	数据类型	说明
ucSampleChl	Input	USINT	采样通道号
pVar0	Input	POINTER TO BYTE	待采集的第 0 子通道数据的地址
pVar1	Input	POINTER TO BYTE	待采集的第 1 子通道数据的地址
pVar2	Input	POINTER TO BYTE	待采集的第 2 子通道数据的地址
pVar3	Input	POINTER TO BYTE	待采集的第 3 子通道数据的地址
uiVarType0	Input	UDINT	待采集的第 0 子通道数据类型，数据类型为：USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、F32=7、LREAL=8

uiVarType1	Input	UDINT	待采集的第 1 子通道数据类型，数据类型为：USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、F32=7、LREAL=8
uiVarType2	Input	UDINT	待采集的第 2 子通道数据类型，数据类型为：USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、F32=7、LREAL=8
uiVarType3	Input	UDINT	待采集的第 3 子通道数据类型，数据类型为：USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、F32=7、LREAL=8
HMC_SetSampleVar	Output	UDINT	0:成功 其它：错误码

5.5.3.1.3 指令类型

非轴运动缓存指令。

5.5.3.1.4 示例

对轴 0 的 Mpos（LREAL）和 VpSpeed（LREAL）进行采样，采样时间间隔分别为 1 倍伺服周期和 2 倍伺服周期，伺服周期为 1 ms，采样数都为 3000。

```
Axis0.Speed:=2000;
HMC_Forward(0);

HMC_SetSampleVar (0, ADR(axis0.MPos), ADR(axis0.VpSpeed), ADR(Temp), ADR(Temp), 8, 8, 7, 7);      (*使用 0 号通道的子通 0、1 进行采样，2、3 通道采样 Temp 变量，类型为 F32*)
HMC_SetSamplePara(0,1,3000,ADR(RecordAry0));      (*设置 0 号通道的采样间隔时间为 1 个伺服周期,采样点数 3000 个/子通道，采集数据存放到数组 RecordAry0*)

HMC_SetSampleVar (1, ADR(axis0.MPos), ADR(axis0.VpSpeed), ADR(Temp), ADR(Temp), 8, 8, 7, 7);
HMC_SetSamplePara(1, 10, 3000, ADR(RecordAry1));      (*设置 1 号通道的采样间隔时间为 10 个伺服周期,采样点数 3000 个/子通道，采集数据存放到数组 RecordAry1*)

HMC_TriggerSample(0);      (*触发采样通道 0*)
HMC_TriggerSample(1);      (*触发采样通道 1*)
```

查看数组 RecordAry0 和 RecordAry1 数组存储的采样结果如下：

变量名	直接地址	在线值
RecordAry0[0,0]	%MD0	5238
RecordAry0[0,1]	%MD8	2000
RecordAry0[1,0]	%MD16	5240
RecordAry0[1,1]	%MD24	2000
RecordAry0[2,0]	%MD32	5242
RecordAry0[2,1]	%MD40	2000
RecordAry0[3,0]	%MD48	5244
RecordAry0[3,1]	%MD56	2000
RecordAry0[4,0]	%MD64	5246
RecordAry0[4,1]	%MD72	2000
RecordAry0[5,0]	%MD80	5248
RecordAry0[5,1]	%MD88	2000
RecordAry0[6,0]	%MD96	5250
RecordAry0[6,1]	%MD104	2000
RecordAry0[7,0]	%MD112	5252
RecordAry0[7,1]	%MD120	2000

变量名	直接地址	在线值
RecordAry1[0,0]	%MD5000	8320
RecordAry1[0,1]	%MD5008	2000
RecordAry1[1,0]	%MD5016	8340
RecordAry1[1,1]	%MD5024	2000
RecordAry1[2,0]	%MD5032	8360
RecordAry1[2,1]	%MD5040	2000
RecordAry1[3,0]	%MD5048	8380
RecordAry1[3,1]	%MD5056	2000
RecordAry1[4,0]	%MD5064	8400
RecordAry1[4,1]	%MD5072	2000
RecordAry1[5,0]	%MD5080	8420
RecordAry1[5,1]	%MD5088	2000
RecordAry1[6,0]	%MD5096	8440
RecordAry1[6,1]	%MD5104	2000
RecordAry1[7,0]	%MD5112	8460
RecordAry1[7,1]	%MD5120	2000

存储示意图

可以看出 RecordAry0 中存储的 Mpos 采样间隔的差值为 2，符合速度为 2000、伺服周期为 1ms、采样间隔时间为 1 个伺服周期；RecordAry1 中存储的 Mpos 采样间隔的差值为 20，符合速度为 2000、伺服周期为 1 ms、采样间隔时间为 10 个伺服周期。

5.5.3.2 HMC_SetSamplePara（设置采样参数）

5.5.3.2.1 功能

配置指定采样通道的采样间隔时间、采样点数、采样数据存储区地址。需注意，采样间隔时间 uiTime 的单位为伺服周期，uiTime=2，表示采样间隔为 2 个伺服周期，LX 控制器支持 0.25、0.5、1、2、4 ms 伺服周期。

5.5.3.2.2 参数

参数	声明	数据类型	说明
ucSampleChl	Input	USINT	采样通道
uiTime	Input	USINT	记录间隔时间。不可为 0，单位：伺服周期
uiNum	Input	UDINT	记录点数
pRecordArea	Input	POINTER TO BYTE	采样记录数组的起始地址（采样记录数组只允许定义在 M 区，系统不允许记录区域大小超过 M 区；用户在定义记录数组时应计算好大小，避免越界情况发生）
HMC_SetSamplePara	Output	UDINT	0：成功 其他：错误码

5.5.3.2.3 指令类型

非轴运动缓存指令。

5.5.3.2.4 示例

参考 HMC_SetSampleVar（设置采样变量）。

5.5.3.3 HMC_GetSampleData（获取采样数据）

5.5.3.3.1 功能

获取指定通道的某子通道的序号为 uiDataNum 的采样数据。

5.5.3.3.2 参数

参数	声明	数据类型	说明
ucSampleChl	Input	USINT	采样通道
ucChannel	Input	USINT	子通道号（0~3）
uiDataNum	Input	UDINT	数据号（该子通道的第 uiDataNum 个数据），数据号从 0 开始
HMC_GetSampleData	Output	LREAL	采样数据

			其他：错误码

5.5.3.3.3 指令类型

非轴运动缓存指令。

5.5.3.4 HMC_TriggerSample（触发采样）

5.5.3.4.1 功能

触发指定采样通道开始执行采样。待采样通道的相关参数配置完毕后，HMC_TriggerSample 用来触发采样动作的执行。

5.5.3.4.2 参数

参数	声明	数据类型	说明
ucSampleChl	Input	USINT	采样通道
HMC_TriggerSample	Output	UDINT	0：成功 其它：错误码

5.5.3.4.3 指令类型

非轴运动缓存指令。

5.5.3.4.4 示例

参考 HMC_SetSampleVar（设置采样变量）。

5.5.3.5 HMC_DisableSample（停止采样）

5.5.3.5.1 功能

禁止指定采样通道执行采样。对于正在执行采样的通道，通过本函数可以停止采样，若需要再次触发

采样，必须重新配置采样和触发采样。

5.5.3.5.2 参数

参数	声明	数据类型	说明
ucSampleChl	Input	USINT	采样通道
HMC_DisableSample	Output	UDINT	0：成功 其它：错误码

5.5.3.5.3 指令类型

非轴运动缓存指令。

5.5.3.6 HMC_TriggerScope（触发示波器）

5.5.3.6.1 功能

触发示波器开始执行。

示波器功能需配合 AT 的示波器软件使用，仅在触发方式为“程序”时该功能才起作用。示波器窗口中，选择触发方式为“程序”，切换“停止/运行”按钮进入运行模式后，通过调用此功能块，可触发算法进行采样，同时示波器软件根据采样数据绘制过程曲线。

编写 IEC 程序，当触发条件满足时调用 HMC_TriggerScope，可实现满足某条件后立刻采样的功能。

5.5.3.6.2 参数

参数	声明	数据类型	说明
HMC_TriggerScope	Output	UDINT	0：成功 其它：错误码

5.5.3.6.3 指令类型

非轴运动缓存指令。

5.5.3.6.4 示例

轴 0 投放持续正转指令，当轴 0 位置达到 5000 时，利用示波器获取轴 0 的速度和位置信息。

设置 AT 触发方式为“程序”，切换“停止/运行”按钮进入运行模式。调用如下程序所在 IEC 任务：

```
HMC_Forward (0);
WHILE axis0.Mpos < 5000 DO
;
END_WHILE
HMC_TriggerScope();      (*触发示波器采样*)
```

5.5.3.7 HMC_GetSampleNum（获取已采样数目）

5.5.3.7.1 功能

获取指定采样通道当前已完成的采样数目。完成该采样通道下的 1 个子通道的采样，则已采样数目累加 1。

即如果 HMC_SetSamplePara 中设置采样点数为 N，且 HMC_ SetSampleVar 设置四个子通道采样均有效，则当该通道完成采样后，通过 HMC_GetSampleNum 获取的已采样数目应为 4*N。

5.5.3.7.2 参数

参数	声明	数据类型	说明
ucSampleChl	Input	USINT	采样通道
HMC_GetSampleNum	Output	UDINT	已采样数目 其它：错误码

5.5.3.7.3 指令类型

非轴运动缓存指令。

5.5.3.8 HMC_SetSampleSingleVar（设置单个采样变量）

5.5.3.8.1 功能

配置指定采样通道的 0 号子通道的待采样数据来源，即仅采集单个变量。该函数仅使用 0 号子通道，且同时将 1~3 号子通道的待采样数据均设置为空，即后续 3 个子通道均不进行采样。

5.5.3.8.2 参数

参数	声明	数据类型	说明
ucSampleChl	Input	USINT	采样通道
pVar	Input	POINTER TO BYTE	待采集的数据的地址
uiVarType	Input	UDINT	待采集的数据类型，数据类型为： USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、F32=7、LREAL=8
HMC_SetSampleSingleVar	Output	UDINT	0：成功 其它：错误码

5.5.3.8.3 指令类型

非轴运动缓存指令。

5.5.4 运动保持功能指令

5.5.4.1 HMC_HoldConfig(设置强制速度开关)

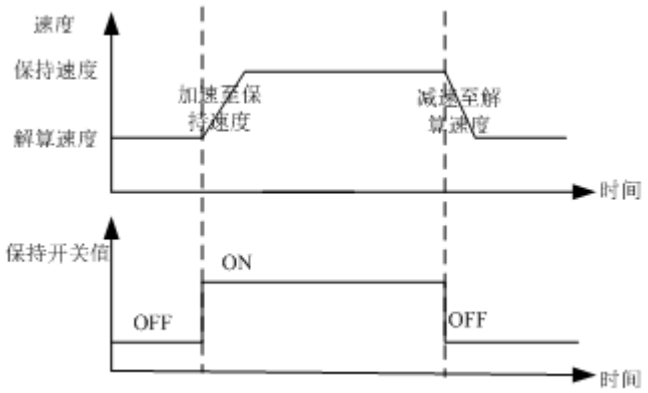
5.5.4.1.1 功能

速度强制功能是指某个轴关联的速度保持开关 DI 通道被触发即值为 1 时，该轴会从当前速度按照

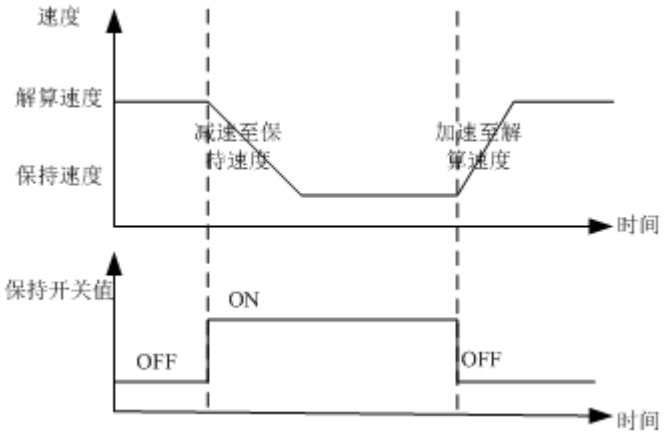
Accel 或 Decel 加减速到 HoldSpeed 设定速度；当 DI 通道解除触发状态即值为 0 时，该轴又按照 Accel 或 Decel 加减速到指令解算速度。

该算法块用于配置轴的速度强制开关为指定 DI，sDiChan 参数指定 DI 通道号。当参数 sDiChan 等于-1 时，表示关闭此功能。

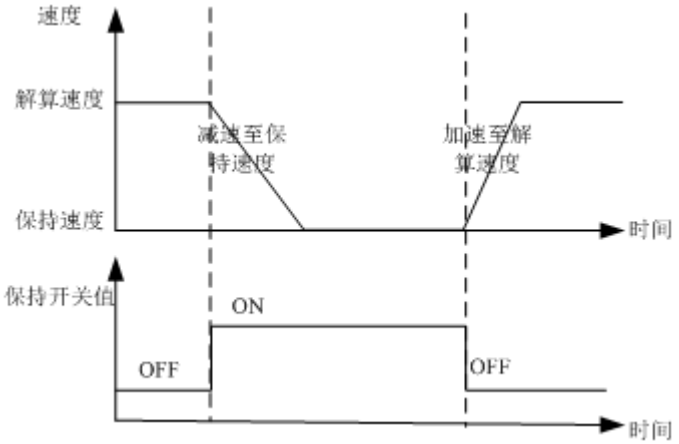
以保持速度为 0，保持速度大于运行速度，保持速度小于运行速度，三种常见情况下轴的速度变化如下：



保持速度大于解算速度



保持速度小于解算速度



保持速度等于 0

5.5.4.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
sDiChan	Input	UDINT	DI 通道号
HMC_HoldConfig	Output	UDINT	返回值

5.5.4.1.3 指令类型

非轴运动缓存指令。

5.5.4.1.4 示例

轴在正常运动过程中，当某条件满足时，如按下暂停按钮，需使得该轴暂停运动，保持静止；当暂停按钮弹起，保持条件解除时，该轴再恢复之前运动。

该按钮状态可通过 DI 检测，使用第 10 路 DI，编写程序如下：

```
axis0.HoldSpeed:=0;            (*设置保持速度为 0*)
HMC_HoldConfig (0, 10);        (*配置第 10 路 DI 为轴 0 速度保持开关*)
HMC_ForwardEx (0, 10000);      (*投入运动指令，轴 0 以 10000 速度运动*)
```

(*当现场的暂停按钮按下时，第 10 路 DI 检测为 1，算法检测到该 DI 为 1 时，从当前速度减速至保持速度 0，实现该轴暂停；当现场的暂停按钮弹起时，第 10 路 DI 为 0，算法检测到该 DI 为 0，从速度 0 加速至设定速度 10000*)

5.5.5 运动前瞻

本章节是运动前瞻功能，该功能是对轴运动持续不断地追踪、统计、分析，在计划时间内得出解算结果，是对轴运动存在的多影响因素的综合考虑。

控制器提供段内梯形或 S 形加减速控制功能，即对每一段都进行加减速处理，指令间速度独立。因此各个指令段连接处，一定会经历速度先减速为零，然后再下条指令开始后再重新进行加速的过程，宏观上，速度是持续的加减速、加减速。实际应用时，不仅影响加工效率，而且在长度极短的连续微线段加工时，会对机床产生巨大的加速度冲击，造成机床振动，甚至导致工件过切。因此需要更加高级的速度控制方法。

系统的速度前瞻功能，一方面可以对指令进行整体规划，即对各段速度进行整体规划，再配合指令段内的加减速控制，可以使机床保持高速运行提高效率，使负载运动更加流畅，告别停停走走，系统通过 Merge 速度融合功能实现；另一方面，再保证高速运行基础上为了限制机械冲击和过切等，还需进行减速识别，通过提前识别轨迹变化，从而按照安全的减速度提前减速，系统通过减速/停止融合功能、抑制冲击功能实现。整体来看，速度前瞻功能既可提升整机效率，也可减少冲击增加柔性，降低零部件磨损，增加设备使用寿命。

系统的运动前瞻功能包括运动两大类，1-融合功能，其包括速度融合、减速/停止融合功能，2-抑制冲击功能。

5.5.5.1 Merge（融合功能）

Merge 功能是将控制器运动缓存中一连串运动指令连贯起来，使负载运动更加流畅，告别停停走走。其针对指令间的处理：开启 Merge 功能后实现了速度衔接，指令间速度不再减速为 0；而另一方面，为了避免指令间夹角过大时如果速度较大会造成冲击，引入了减速/停止角融合功能。

- 生效范围
  - Merge 功能是否能够成功开启，除了依赖 Merge 开关外，还依赖于指令类型、指令涉及轴的统一等条件，如下：

- 目前仅支持一维指令（单轴指令）、二维指令（两轴插补指令）、N 轴插补指令的 Merge 处理。且支持 S 形加减速 Merge；
- 单轴指令中，只支持 HMC_Move、HMC_MoveEx、HMC_MoveAbs、HMC_MoveAbsEx 指令；
- 两轴插补指令中，只支持 HMC_Move2D、HMC_Move2DEx、HMC_MoveAbs2D、HMC_MoveAbs2DEx、HMC_MoveCircle、HMC_MoveCircleEx 指令；
- N 轴插补指令中，只支持 HMC_MoveND、HMC_MoveAbsND、HMC_MoveNDEx、HMC_MoveAbsNDEx 指令；
- 两轴/N 轴插补指令中，若插补的合轴、分轴所对应的轴号序列发生变化时 Merge 无效；
- 维度（轴数）不同的指令之间相互连接时 Merge 无效；
- 若当前指令正在执行 HMC_MoveModify，HMC_SetMerge 参数设置失败；
- 二维、N 维指令中，不允许第 N 段、N+1 段的指令转角 θ 过大，θ 为 Merge 所连接的两相邻指令之间的连接角。指令转角 θ 小于减速角（减速角由 HMC_SetMergeAngle 函数设置）时可以正常 Merge；减速角 θ 大于等于停止角（停止角由 HMC_SetMergeAngle 函数设置）时 Merge 无效，指令转角 θ 大于等于减速角且小于停止角时，当前指令以小于 Speed 的速度结束（按比例）；
- 如果 Merge 功能处于开启状态，在投放指令 HMC_MoveModify 后 Merge 自动关闭；用户需手动重新开启 Merge 功能；
- 如果 Merge 功能处于开启状态，在使用 Cancel 撤销指令后 Merge 自动关闭；用户需手动重新开启 Merge 功能。

■ Merge 功能开启

通过 HMC_SetMerge 函数设置 Merge 功能是否开启及开启方式。

- Merge=0

  Merge 关闭，当前指令块结束时的速度为 0。

- Merge=1

  Merge 打开，当前指令块结束时目标速度为该指令块的 Speed

- Merge=2

  Merge 打开，当前指令块结束时目标速度为下一指令块的 Speed

- Merge=3

  Merge 打开，当前指令块结束时目标速度为该指令块 Speed 和下条指令块 Speed 最小值

- Merge=4

  Merge 打开，当前指令块结束时目标速度为该指令块 Speed 和下条指令块 Speed 最大值

■ Merge 的主要设计准则如下：

- 1 D 指令衔接处，如果本条指令剩余位移不足，则提前输出下一条指令的部分位移，防止每条指令的最后一个周期速度跃变；

- 2 D 插补指令衔接处，不强制插补轨迹必须经过两条指令的衔接点，如果本条指令剩余位移不足，则提前输出下一条指令的部分位移；

- 对于改变速度 speed 的 EX 指令：

1） 尽量保证所有 EX 指令衔接点预设的速度够达到，如果只能保证部分衔接点的预设速度，则优先保证速度较低的衔接点，不能达到预设速度的衔接点需保证实际运行速度小于其预设速度。

2） 必须保证 VpSpeed 的变化过程始终满足预设的梯形/S 形规律；

3） 在满足上述准则所需约束条件的前提下，各指令在各自的生命周期内保证尽可能多的时间以自身的 speed 为目标速度运动。

- 对于存在拐角限速/Jerk 限速的指令：

1）在逐级前瞻计算的过程中判断当前前瞻的衔接点是否需要拐角限速，如果需要则计算限制速度。

尽量保证所有指令衔接处的限制速度速度够达到，如果只能保证部分衔接点的限制速度，则优先保证限制速度较低的衔接点，不能达到限制速度的衔接点需保证实际运行速度小于其限制速度。

2）运动过程中必须保证 VpSpeed 的变化过程始终满足预设的梯形/S 形规律；

3） 在满足上述准则所需约束条件的前提下，各指令在各自的生命周期内保证尽可能多的时间以自身的 speed 为目标速度运动。

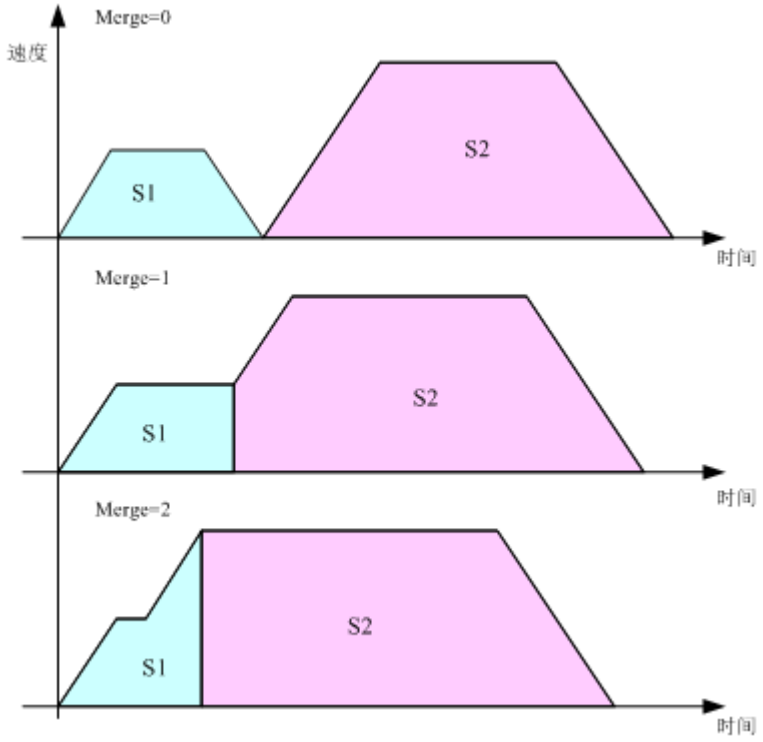
□ 如下条件可作为前瞻计算的结束判据

1) 前瞻已达到指令缓存队列尾部，则前瞻结束；

2) 前瞻计算尚未达到指令缓存队列尾部，但当前条件下可以保证：在前瞻范围之外的指令即使出现任意低速的限制速度，都可以在位移条件满足的前提下，确保能够按照预设的梯形/S形加减速规律到达该速度。则此时可以结束当前前瞻计算。

这样可以有效减小前瞻计算深度，节省计算时间。

例如当前指令速度为 1000，下条指令目标速度为 2000，五种模式下计算结果如下：



Merge 功能示意图

5.5.5.2 减速/停止融合功能

减速/停止融合功能解决的问题是：当指令间夹角过大时，如果仍以较大速度运行，会在夹角处产生较大的机械冲击，轨迹偏离。

控制器会对指令间轨迹变化的夹角进行提前识别，比较其与减速/停止角的大小关系，提前决定是否进行减速，保证在指令连接处平稳过渡。

- 指令间夹角<减速角

  减速/停止角对速度不做限制，Merge 融合功能决定指令结束时的速度。

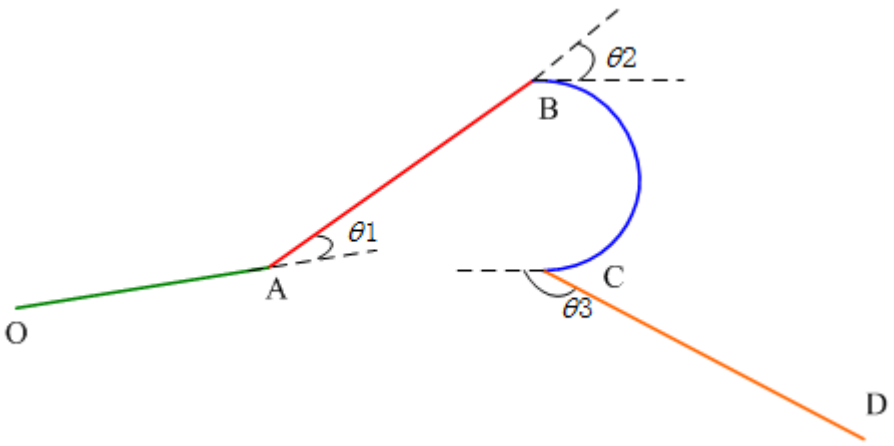
- 减速角<指令间夹角<停止角

  对速度做限制，限制后速度为 Merge 融合决定指令结束速度的 k 倍，该比例系数 k 属于[0,1]范围内，$k = \frac{\text{停止角} - \text{指令间夹角}}{\text{停止角} - \text{减速角}}$。

- 停止角<指令间夹角

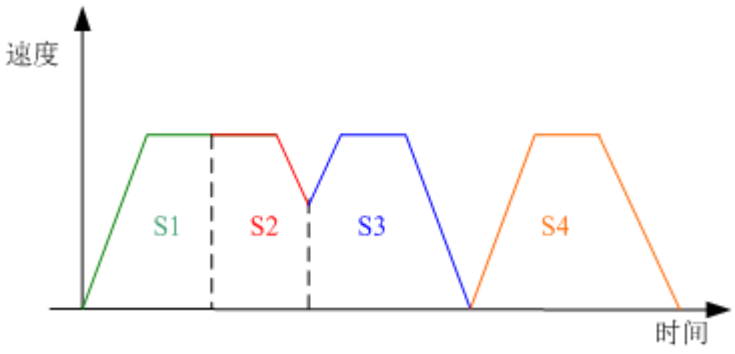
  指令结束速度为 0。

如下，在二维平面走如下轨迹，依次经过 O->A->B->C->D 点执行 4 条指令后完成整个轨迹。假设，4 条指令间不通过其它方式修改合轴的 Speed 参数。



运动轨迹示意图

$\theta 1$ 为指令 1 和指令 2 间的夹角，$\theta 2$、$\theta 3$ 依次为后续相邻指令间的夹角。假设角度满足关系：$\theta 1$ < 减速角< $\theta 2$ <停止角<$\theta 3$ ，则合轴的速度变化如图所示：



合轴速度示意图

5.5.5.3 抑制冲击功能

抑制冲击功能主要解决的问题是：运动轨迹的曲率较大时，如果速度过大会带来较大的惯性离心力，造成机械冲击损坏工件，甚至会导致轨迹偏离失准。

- 生效范围

区别与 Merge 只作用于多条指令之间，抑制冲击功能不仅支持多条 2 维指令之间，还对单条指令生效，具体如下：

单条指令，无需开启 Merge

- 平面圆弧插补指令、球弧插补指令；
- 螺旋线插补指令；
- 样条插补指令；

多条指令之间，需开启 Merge

- 2 轴直线插补（HMC_Move2D、HMC_Move2DEx）和平面圆弧插补（HMC_MoveCircle、HMC_MoveCircleEx）任意组合，插补的合轴、分轴所对应的轴号序列必须相同。

- 抑制冲击

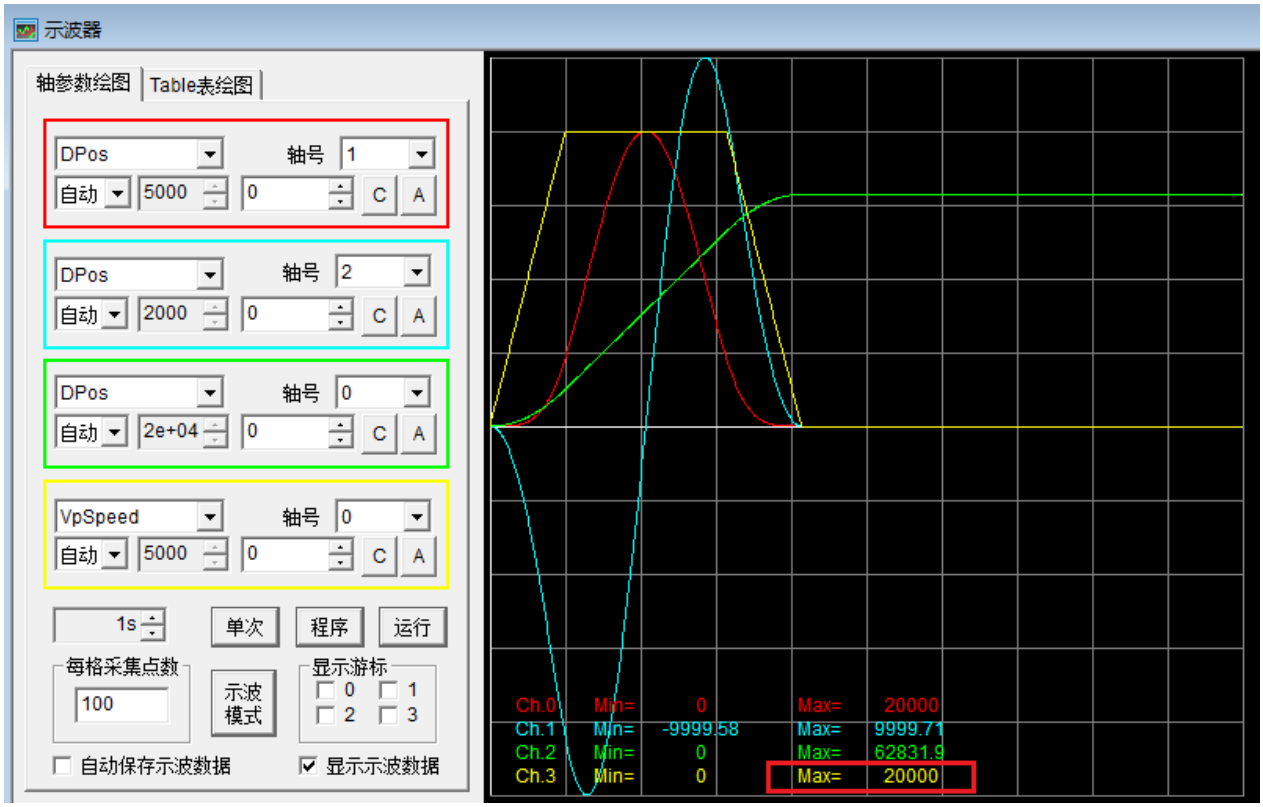
计算指令轨迹的曲率 ρ，然后根据曲率、Jerk 计算出最大速度。

$$V_{max} = f(\rho, Jerk)$$

与指令当前的目标速度比较，是否需要对其进行限制减速，以保证全部轨迹过程内的机械冲击限制在合理范围内，同时保证轨迹的精准。

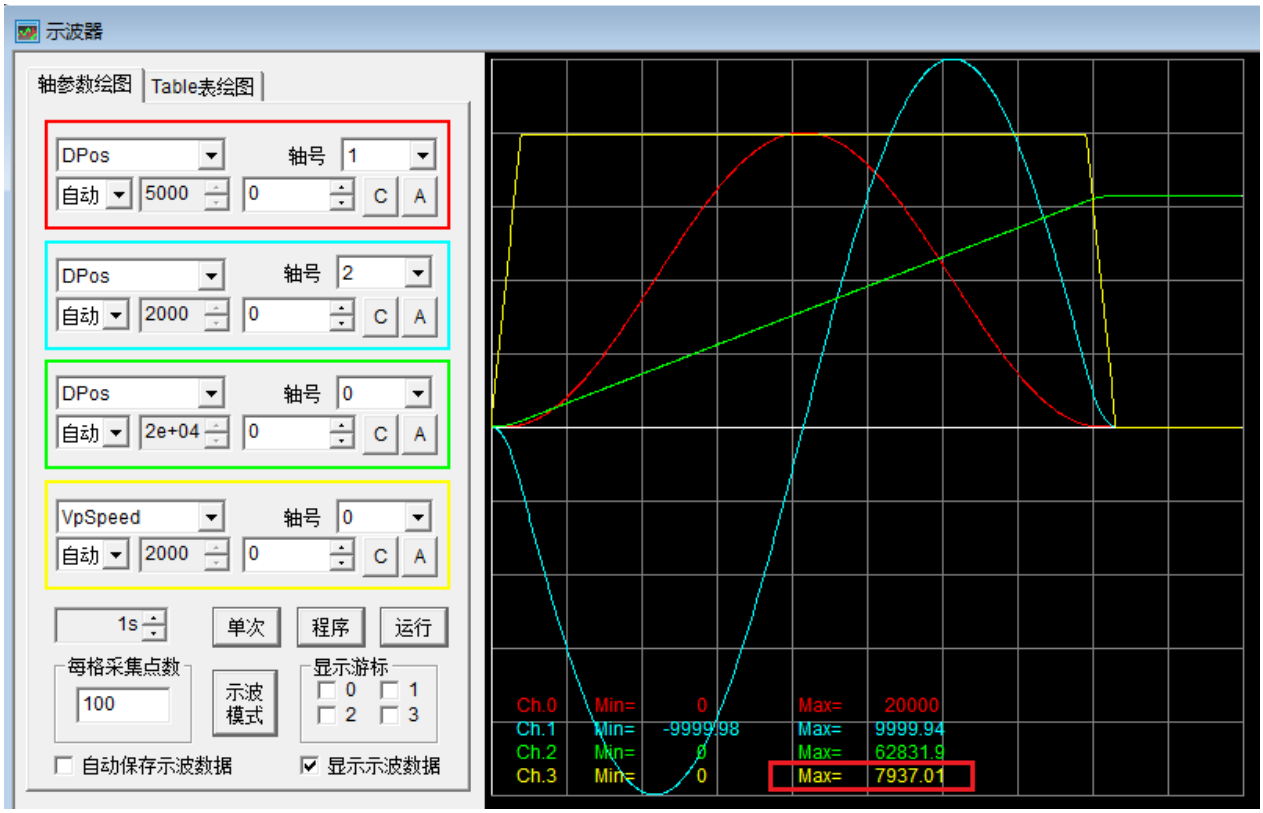
如执行一个圆弧指令，设定相同的 Speed=20000，但设定不同的 Jerk 值，实际运行时合轴的速度发生变化。

Jerk=5000000000 默认值时，合轴速度达到 20000，指令花费较短时间就解算完成。



运动轨迹示意图

修改 Jerk=5000 时，合轴速度最高仅达到 7937.01，指令花费较长时间解算完成。

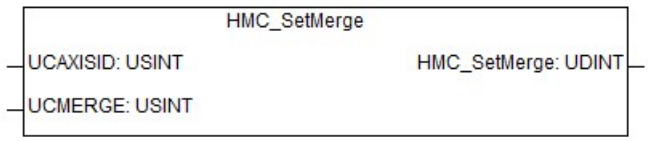


运动轨迹示意图

5.5.5.4 HMC_SetMerge（设置融合功能）

5.5.5.4.1 功能

设置轴参数 Merge 模式。Merge 相关详细描述见轴参数 Merge 和本章节的运动前瞻功能描述部分。



HMC_SetMerge 功能块

5.5.5.4.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号

ucMerge	Input	USINT	Merge 模式，该值只可为 0、1、2、3、4
HMC_SetMerge	Output	UDINT	返回值

5.5.5.4.3 指令类型

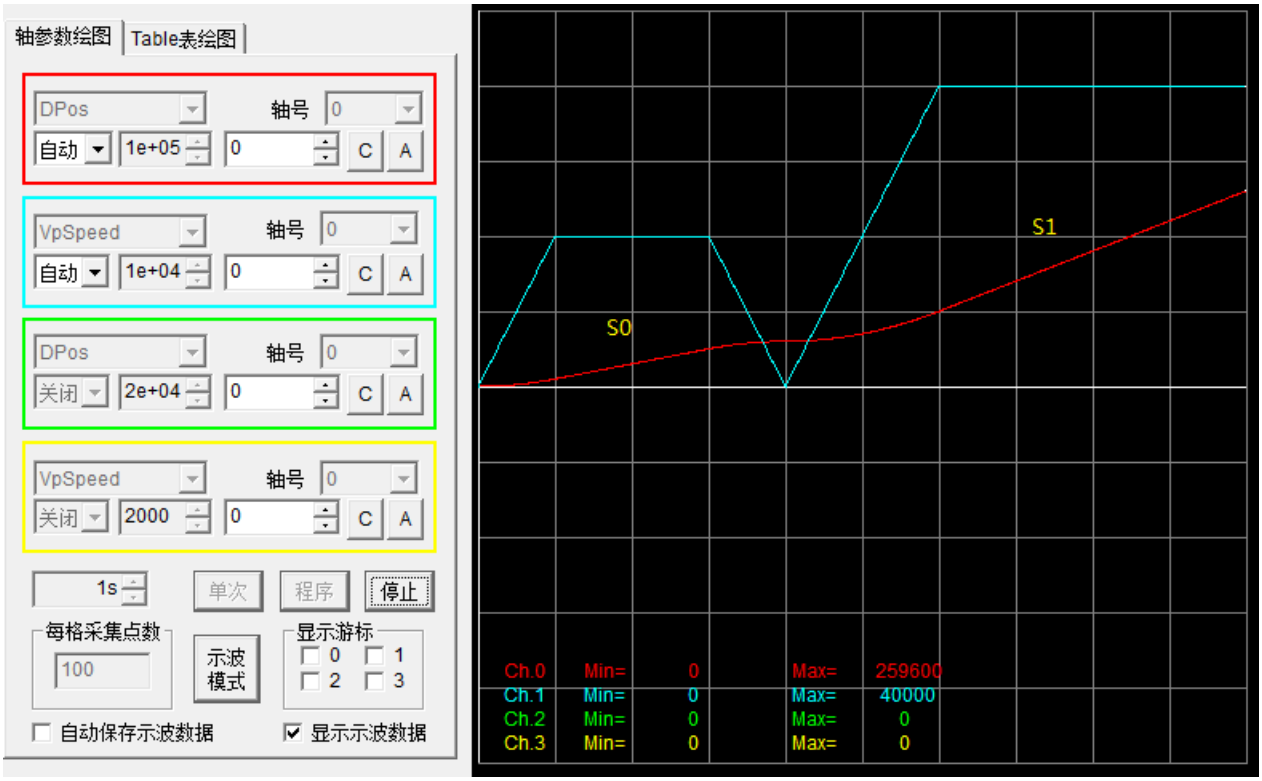
非轴运动缓存指令。

5.5.5.4.4 示例

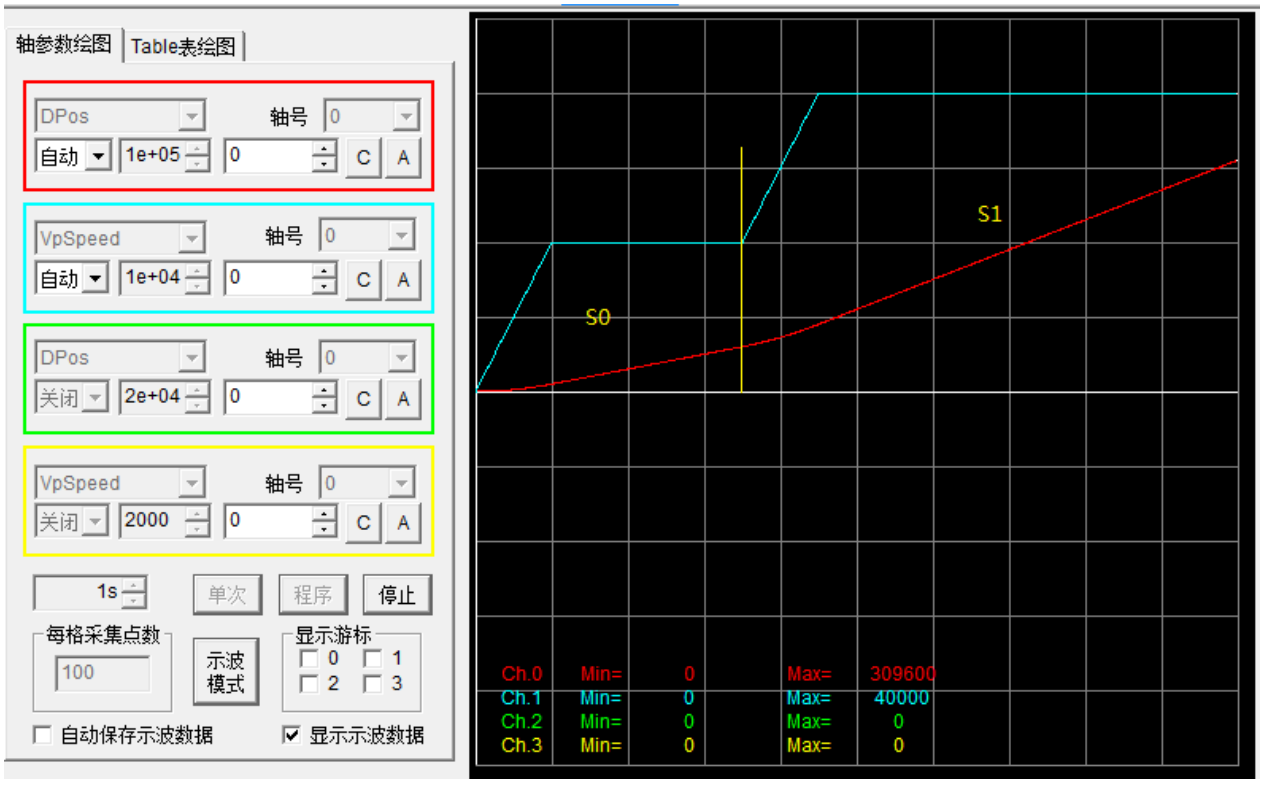
程序如下：

```
axis0.Speed := 20000;      (* 轴 0 速度为 20,000（用户单位/秒） *)
HMC_Move(0, 60000);      (* 指令 0 向前运动 60000（用户单位） *)
HMC_ForwardEx(0, 40000); (* 指令 1：将 axis0.Speed 置为 40,000（用户单位/秒）然后持续向前运动*)
```

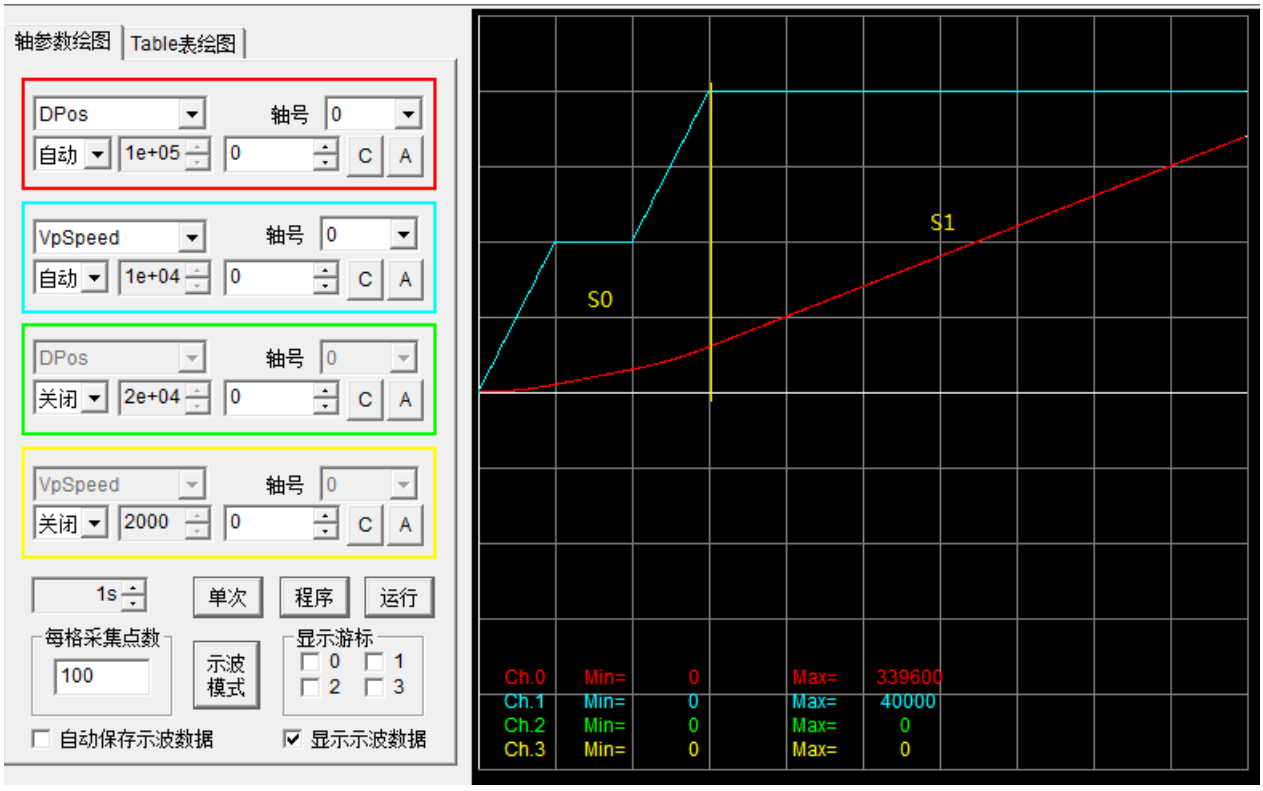
当 axis0.Merge 为 0、1、2 时，曲线形状分别如下所示（其中 S0 均为 60000 用户单位）：



Merge 为 0 示意图



Merge 为 1 示意图



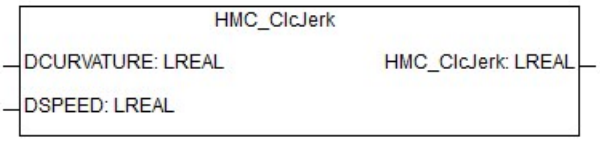
Merge 为 2 示意图

5.5.5.5 HMC_ClcJerk（冲击运算）

5.5.5.5.1 功能

根据 Speed 和曲率计算 Jerk。

Jerk=dCurvature² * dSpeed³



HMC_ClcJerk 功能块

5.5.5.5.2 参数

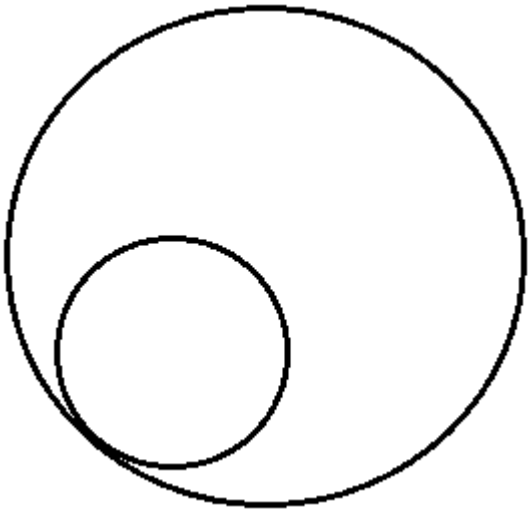
参数	声明	数据类型	说明
dCurvature	Input	LREAL	曲率（非负）
dSpeed	Input	LREAL	速度（非负）
HMC_ClcJerk	Output	LREAL	其他值：返回 Jerk 值 -1：参数错误

5.5.5.5.3 指令类型

非轴运动缓存指令。

5.5.5.5.4 示例

要走如下轨迹，从两圆的交点处为起始点，先经过大圆达到起始点后，在经过小圆最终到达起始点。假设设备能承受的最大 Jerk=100000，希望速度为 1000。



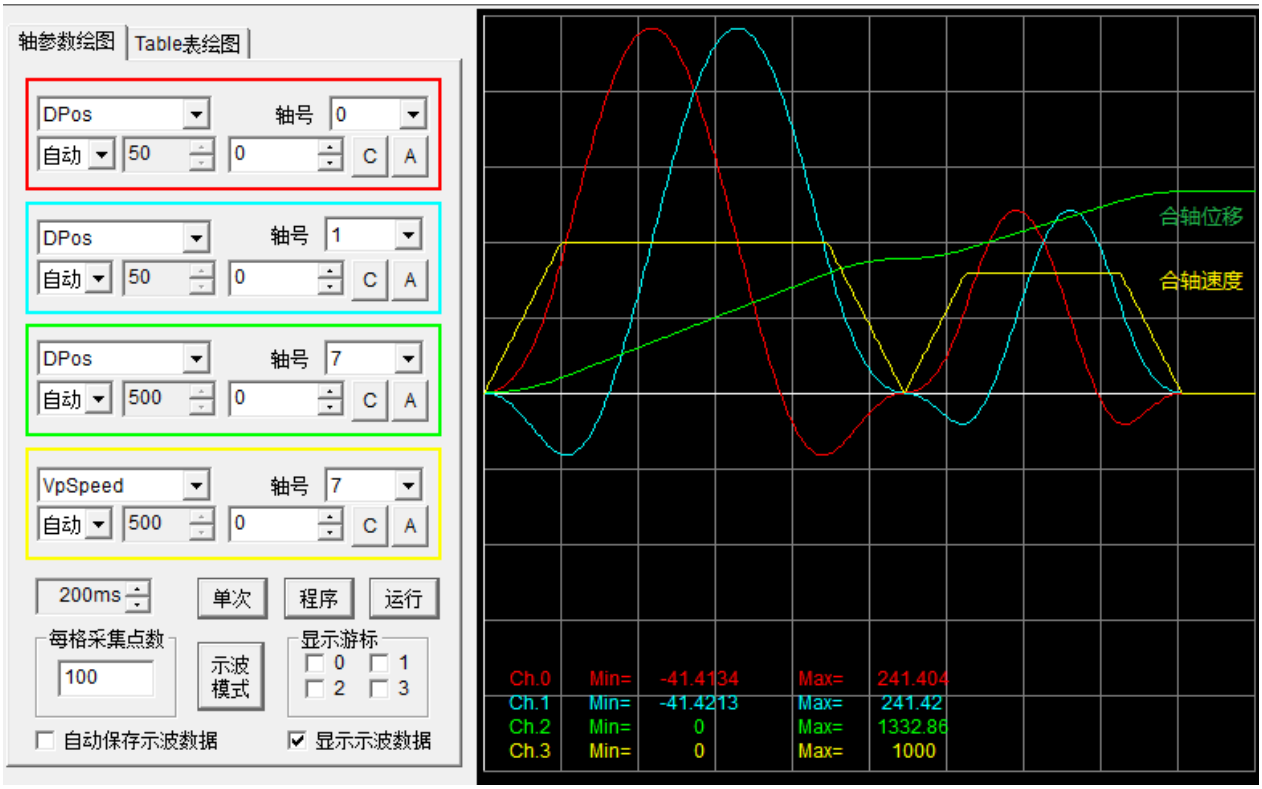
不同轨迹圆

考虑 Jerk 与速度和曲率有关系，而以上两个不同轨迹圆的曲率不同，如果设置 Jerk 过小，会导致实际运行速度受限影响效率，如果设置过大，超过设备能承受的最大 Jerk，则会影响设备安全。故对两个圆分别计算出 Jerk 后，与最大 Jerk 比较，若大于系统承受最大 Jerk 则设置为最大 Jerk 否则按照计算得到的 Jerk。故编写程序如下：

```
Axis7.Speed := 1000;
Axis7.Accel := 5000;
Axis7.Decel:= 5000;

HMC_MoveCircle(7, 0, 1, 1, 0, 0, 100, 100);     (*第一个大圆*)
Jerk1 := HMC_ClcJerk(1/(100*SQRT(2)), 1000);    (*冲击运算函数*)
MaxJerk := 100000;
IF Jerk1 > MaxJerk THEN (*与设备所能承受的最大冲击比较，原则为：安全第一效率第二*)
   axis7.Jerk := MaxJerk;
ELSE
   axis7.Jerk := Jerk1;
END_IF

HMC_MoveCircle(7, 0, 1, 1, 0, 0, 50, 50);
Jerk2 := HMC_ClcJerk(1/(50*SQRT(2)), 1000);     (* 冲击运算函数 *)
```

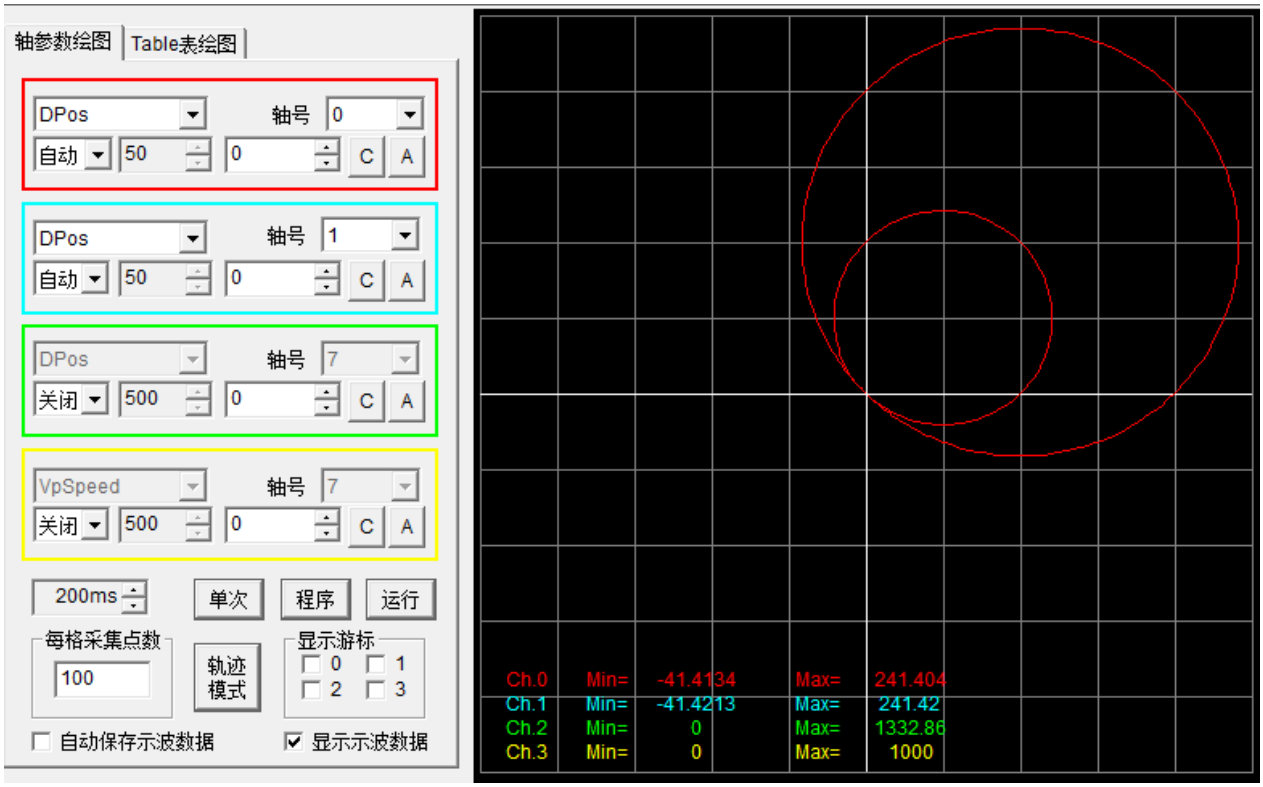
```
MaxJerk := 100000;
IF Jerk2 > MaxJerk THEN（*与设备所能承受的最大冲击比较，原则为：安全第一效率第二*）
   axis7.Jerk := MaxJerk;
ELSE
   axis7.Jerk := Jerk2;
END_IF
```

执行后结果如下：



运动轨迹示意图

从上图可以看出，虽然设置了同样的指令速度 Speed=1000，但小圆轨迹执行时的实际运行速度并未达到 1000，这是因为因为小圆的曲率较大，以 1000 速度运行时需要设备承受的冲击超过了设备能承受的最大冲击 100000,此时设置 Jerk=100000，则限制了实时结算速度。

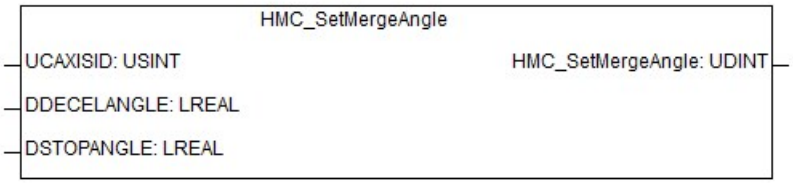


运动轨迹示意图

5.5.5.6 HMC_SetMergeAngle（设置融合角度）

5.5.5.6.1 功能

设置轴的融合角度。



HMC_SetMergeAngle 功能块

5.5.5.6.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号

dDecelAngle	Input	LREAL	设置二维指令的减速角，对合轴设置有效，不可为负
dStopAngle	Input	LREAL	设置二维指令的停止角，对合轴设置有效，不可为负
HMC_SetMergeAngle	Output	UDINT	0：设置成功 其他值：错误码

5.5.5.6.3 指令类型

非轴运动缓存指令。

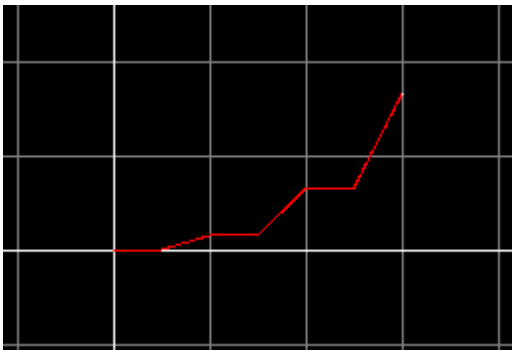
dDecelAngle≤π/6，dStopAngle≤π/2，且 dDecelAngle≤dStopAngle。

dDecelAngle 默认值为 π/6，dStopAngle 默认值为 π/2。

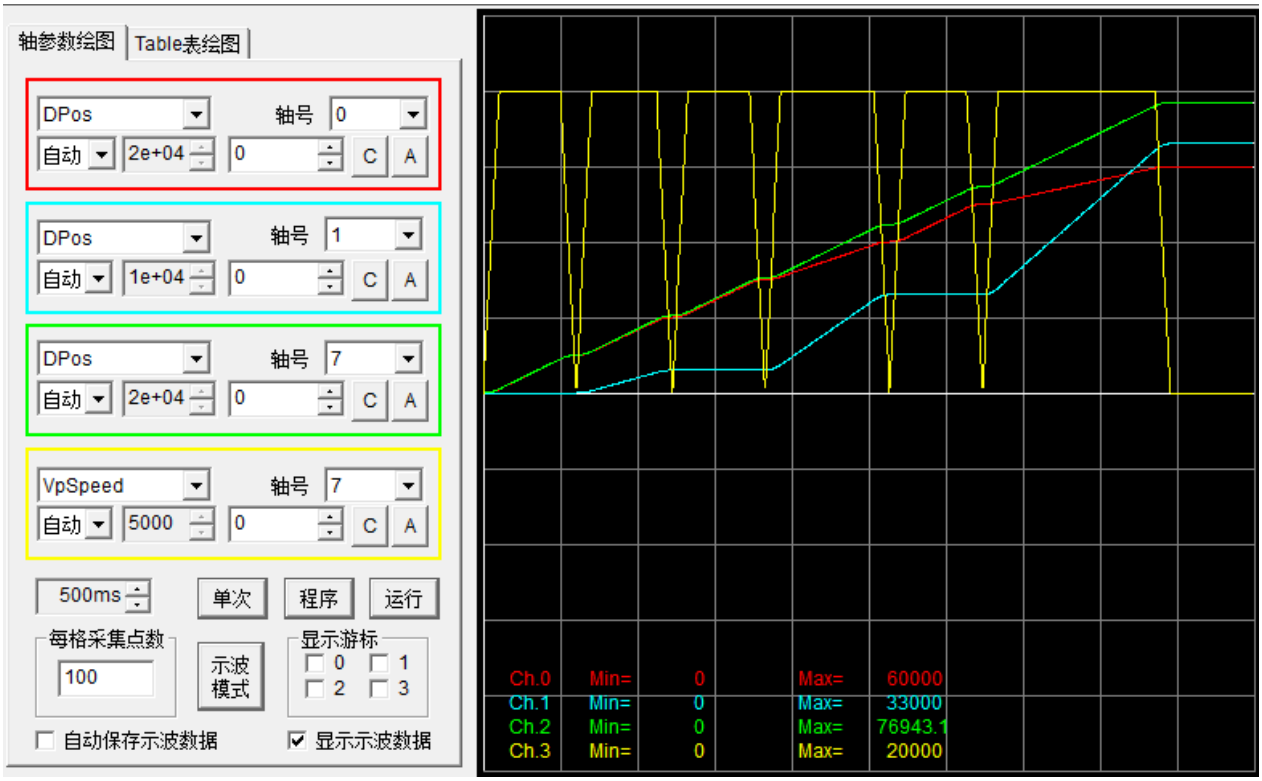
5.5.5.6.4 示例

两轴指令 Merge，程序如下：

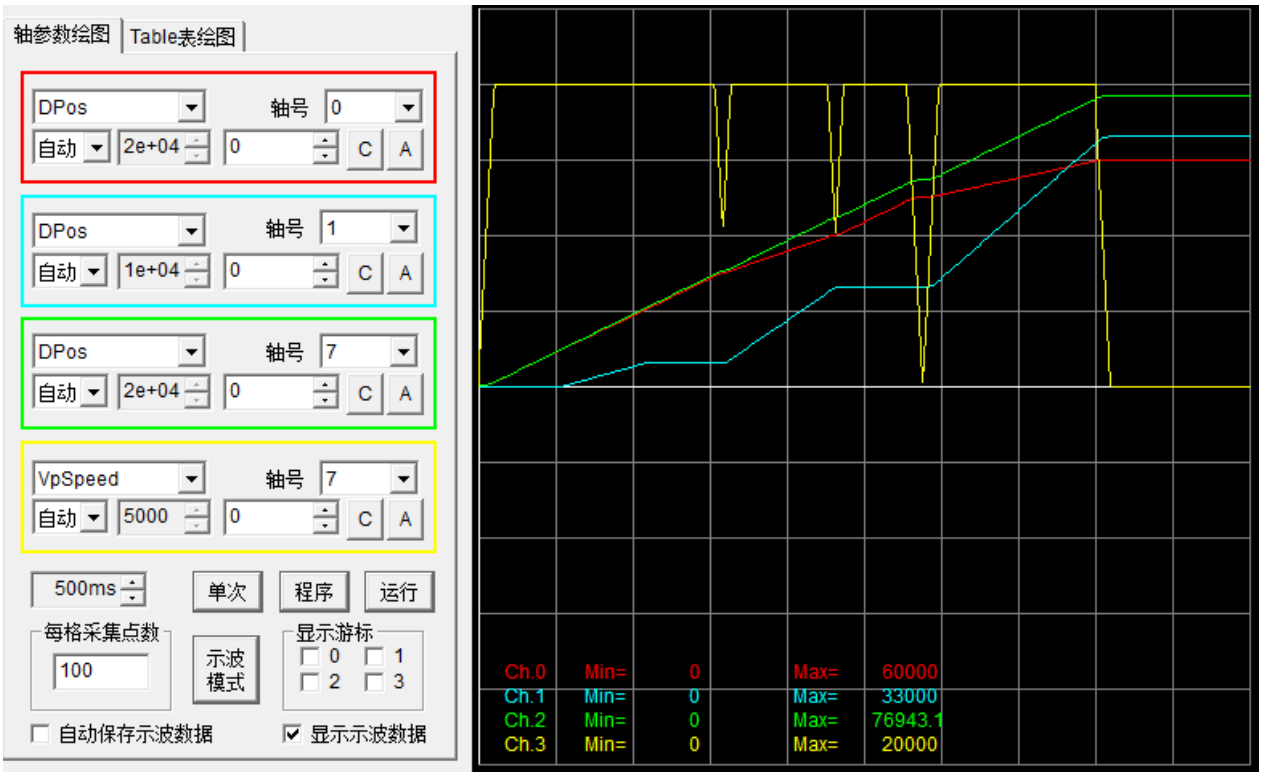
```
axis0.Speed := 20000;      (*轴 0 速度为 20,000（用户单位/秒）*)
axis0.Accel := 200000;
axis0.Decel := 200000;
HMC_Move2D(7, 0, 1, 10000, 0);
HMC_Move2D(7, 0, 1, 10000, 3000);
HMC_Move2D(7, 0, 1, 10000, 0);
HMC_Move2D(7, 0, 1, 10000, 10000);
HMC_Move2D(7, 0, 1, 10000, 0);
HMC_Move2D(7, 0, 1, 10000, 20000);
```



Merge 在二维指令下的情况



Merge 为 0 时的情况



Merge 为 1 时的情况

5.5.5.7 HMC_GetMergeStopAngle（获取融合停止角）

5.5.5.7.1 功能

获取插补运动停止角。

HMC_GetMergeStopAngle 功能块

5.5.5.7.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_GetMergeStopAngel	Output	LREAL	弧度

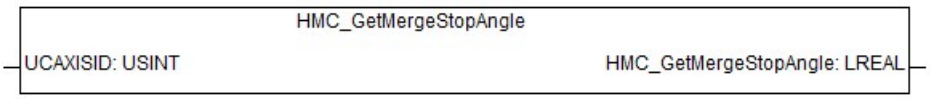
5.5.5.7.3 指令类型

非轴运动缓存指令。

5.5.5.8 HMC_GetMergeDeceAngle（获取融合减速角）

5.5.5.8.1 功能

获取插补运动减速角。



HMC_GetMergeDeceAngle 功能块

5.5.5.8.2 参数

参数	声明	数据类	说

		型	明
ucAxisID	Input	UINT	轴号
HMC_GetMergeDecelAngel	Output	LREAL	弧度

5.5.5.8.3 指令类型

非轴运动缓存指令。

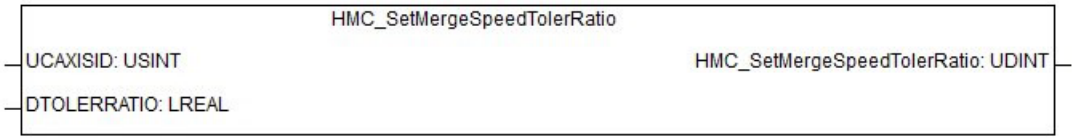
5.5.5.9 HMC_SetMergeSpeedTolerRatio（设定速度波动抑制比）

5.5.5.9.1 功能

该指令设定运动前瞻过程中的速度波动抑制比参数，速度波动抑制功能允许在用户设定的误差比率范围内抑制速度波动。

在运动前瞻过程中，为降低冲击，运行速度受曲率限速以及拐角限速机制的约束，因此速度常常会有一定波动。当使用小折线逼近曲线时，如果用户提供的原始点位数据精度过低，可能会造成曲率限速及拐角限速机制计算出的约束速度频繁上下跳动，从而造成速度高频波动。

速度波动抑制功能可解决上述问题，减小乃至消除这种波动。速度波动抑制比 dTolerRatio 越大，抑制作用越强，但过大时可能会增加系统冲击。因此，在可以接受的速度波动范围内，尽量取 dTolerRatio 为较小值。如果开启 S 形加减速并设置合适的加加速度，则可在同样的 dTolerRatio 下达到更好的波动抑制效果。



HMC_SetMergeSpeedTolerRatio 功能块

5.5.5.9.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号

dTolerRatio	Input	LREAL	速度波动抑制比[0,1] 值越大抑制作用越强，但过大可能会增加系统冲击
HMC_SetMergeSpeedTolerRatio	Output	UDINT	返回值 0：成功，错误码：参数错误

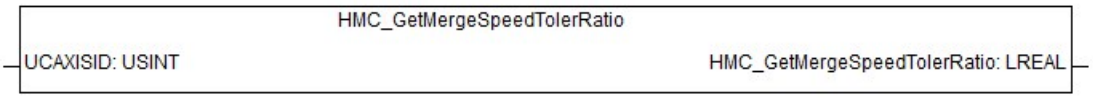
5.5.5.9.3 指令类型

非轴运动缓存指令。

5.5.5.10 HMC_GetMergeSpeedTolerRatio（读取速度波动抑制比）

5.5.5.10.1 功能说明

读取当前的速度波动抑制比。



HMC_GetMergeSpeedTolerRatio 功能块

5.5.5.10.2 输入参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_GetMergeSpeedToleRatio	Output	LREAL	当前速度波动抑制比

5.5.5.10.3 指令类型

非轴运动缓存指令。

5.5.6 曲线限速

5.5.6.1 HMC_SetCurveJerkLimit（设定曲线冲击上限）

5.5.6.1.1 功能

该指令设定曲线插补过程中物理冲击的上限值，由此可限定曲线的插补速度。

通过合理设定该参数，可使插补过程中曲线的大曲率处（如小圆圆弧）有较大的降速，而小曲率处（如大圆圆弧）降速较少或不降速。

可通过设备在线调试找到满足具体设备的冲击参数值，也可通过 HMC_ClcJerk 指令计算出在特定曲率（半径的倒数）、特定速度下的冲击值，作为冲击上限值。

5.5.6.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dLimitVal	Input	LREAL	设定的上限值，非负
HMC_SetCurveJerkLimit	Output	UDINT	0：成功 错误码：参数错误

5.5.6.1.3 指令类型

非轴运动缓存指令。

5.5.6.2 HMC_GetCurveJerkLimit（读取曲线冲击上限）

5.5.6.2.1 功能

读取设定的曲线插补物理冲击的上限值。

5.5.6.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_GetCurveJerkLimit	Output	LREAL	曲线插补物理冲击上限值 指令参数错误时返回 0

5.5.6.2.3 指令类型

非轴运动缓存指令。

5.5.7 切向追踪

5.5.7.1 HMC_MoveTang（切向追踪）

5.5.7.1.1 功能

该功能块根据 X、Y 运动轨迹，实时计算出二者合运动轨迹与 X 轴正向夹角，然后驱动切刀旋转轴 C 轴跟踪至目标角度，可用在物料切割场景，自动实时调整刀具方向与切割轨迹的切向方向（切割速度矢量方向）保持一致，达到最好的切割效果。可设置刀具倾角的偏移量 dOffset，以调节刀具切割时的前角、后角角度。

该指令运行过程中，C 轴（刀具旋转轴）工作在循环行程模式下，模式 RepMode 为 2(对称区间)。使用该指令前，用户需首先把循环行程长度（[-Repdist ,Repdist)）设为该轴旋转一圈所对应的用户单位下的位移量(假设为 RoundLength)。即：

Repdist = RoundLength/2

这样，C 轴的绝对位置[-Repdist ,Repdist)就对应于[-π,π)的切向角度。

指令执行过程中实时计算出 X 轴（轴号 ucAxisX）、Y 轴（轴号 ucAxisY）合运动轨迹的切向方向与 X 轴正向所成的角度 TangAngle，即 C 轴的目标切向角；同时结合目标切向角的变化速度（角速度）。实时调整 C 轴目标位置，驱动 C 轴（轴号 ucAxisC）定位至目标切向角，实现对两轴合运动轨迹角度的最优跟随。

使用切向追踪功能时，用户最好先估算出刀具旋转轴 C 轴的最大加减速能力，以及最大运行速度，这些参数可根据电机的额定转矩、额定转速等参数，结合负载情况以及设备机械结构对振动、扭矩的要求估算得到。根据上述计算把 C 轴的加减速、speed 参数设定为设备所允许的最大值（或乘以一定的安全余量系数）。在切向追踪执行过程中，算法会按照不超过 C 轴设定的加减速及 speed 的运动参数驱动 C 轴至目标切向角度。

若 C 轴的最大加减速能力小于实际切向角变化的加/减速度，或最大运行速度小于实际切向角变化的最大速度，则 MoveTang 运行过程中 C 轴不能实现无偏差跟随切向角度。

在指令执行过程中，当目标切向角发生跃变或目标切向角的变化速度（角速度）发生跃变时，刀具旋转轴 C 将以轴参中设定的加减速及速度朝目标位置运动，而不会发生因运行速度的跃变而造成的冲击。但此处 C 轴的实际角度位置与目标切削角之间将存在一定的跟随偏差。

设备调试时可用 HMC_ GetMoveTangDeviation 实时获取切向角跟随偏差，查看切向追踪实际运行效果。如果跟随偏差超出应用所要求的范围，则可用 HMC_GetMoveTangDestDirection 和 HMC_GetMoveTangDestVelocity 指令分别获取目标切向角位置、切向角变化速度，通过对比 C 轴 DPOS 变化曲线与 VpSpeed 变化曲线，可分析出偏差产生的具体原因，一般可分为以下 3 类：

（1） C 轴加减速/Speed 过小，无法跟随切向角变化。解决措施：

- 加大 C 轴加减速/Speed；
- 减小 XY 轴插补速度以降低切向角变化速率；
- 若 XY 轴插补速度不可降低，且 C 轴加减速/Speed 已达到 C 轴电机扭矩极限或机械结构极限，则考虑更换更大扭矩的电机或变更机械结构。

（2） 切向角位置 发生跃变。（XY 插补轨迹存在一阶不可导点，如两条折线）。解决措施：

- 优化 XY 插补路径。比如对于折线路径，可在折线之间插入圆弧；
- 如果需要的就是折线轨迹，则可加入辅助进退刀路径，使刀具首先从一段折线的延长线切出，再从另一段折线的延长线切入。
- 加大 C 轴加减速/Speed 或 减小跃变点处 XY 轴插补速度 可降低该偏差的影响范围，但无法根本消除。

（3） 切向角变化速度 发生跃变。解决措施：

- 优化 XY 插补路径。比如使用样条插补轨迹。
- 如果插补路径不可更改，则可加大 C 轴加减速/Speed 或 减小跃变点处 XY 轴插补速度。

5.5.7.1.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
ucAxisX	Input	USINT	X 方向运动轴轴号
ucAxisY	Input	USINT	Y 方向运动轴轴号
dOffset	Input	LREAL	刀具旋转轴偏移量（用户单位）
JMC_MoveTang	Output	UDINT	轴指令 ID：成功 0Xffffffff：函数参数错误

5.5.7.1.3 指令类型

轴运动缓存指令。

5.5.7.1.4 补充说明

(1) 该指令一旦被投送成功，则一直生效，直到执行 cancel 指令撤销为止。

(2) 切向追踪支持 X、Y 轴为任意指令类型。

(3) 刀具旋转轴 C 轴的最大速度、加减速由其自身轴参设定值决定，应根据运动情况及该轴自身的特性设定合适的值（在轴自身能承受的范围内保证该值足够大），以确保 C 轴能够快速完成跟随切向角的动作。

(4) 当两段运动轨迹的衔接处发生急剧的角度变化时，此时程序会自动选择最短的路径追踪定位切向角度，不会出现打转现象。

(5) 在指令执行过程中，当目标切向角或目标切向角的变化速度（角速度）发生跃变时，刀具旋转轴 C 将以轴参中设定的加减速及速度朝目标位置运动，而不会发生因运行速度的跃变而造成冲击。但此处 C 轴的实际角度位置与目标切削角之间将存在一定的跟随偏差。

（6） X/Y 轴遇到限位时，在减速停止过程中切向角会按照实际减速轨迹计算。在限位停止动作完成后，若行程超限状态尚未解除，并且后续所投送到主轴 XY 中的指令不是朝 X/Y 限位的反方向运行的（减轻位置超限方向），则按照指令使用异常处理，出于安全考虑，此时刀具旋转轴 C 轴将保持原位不动。

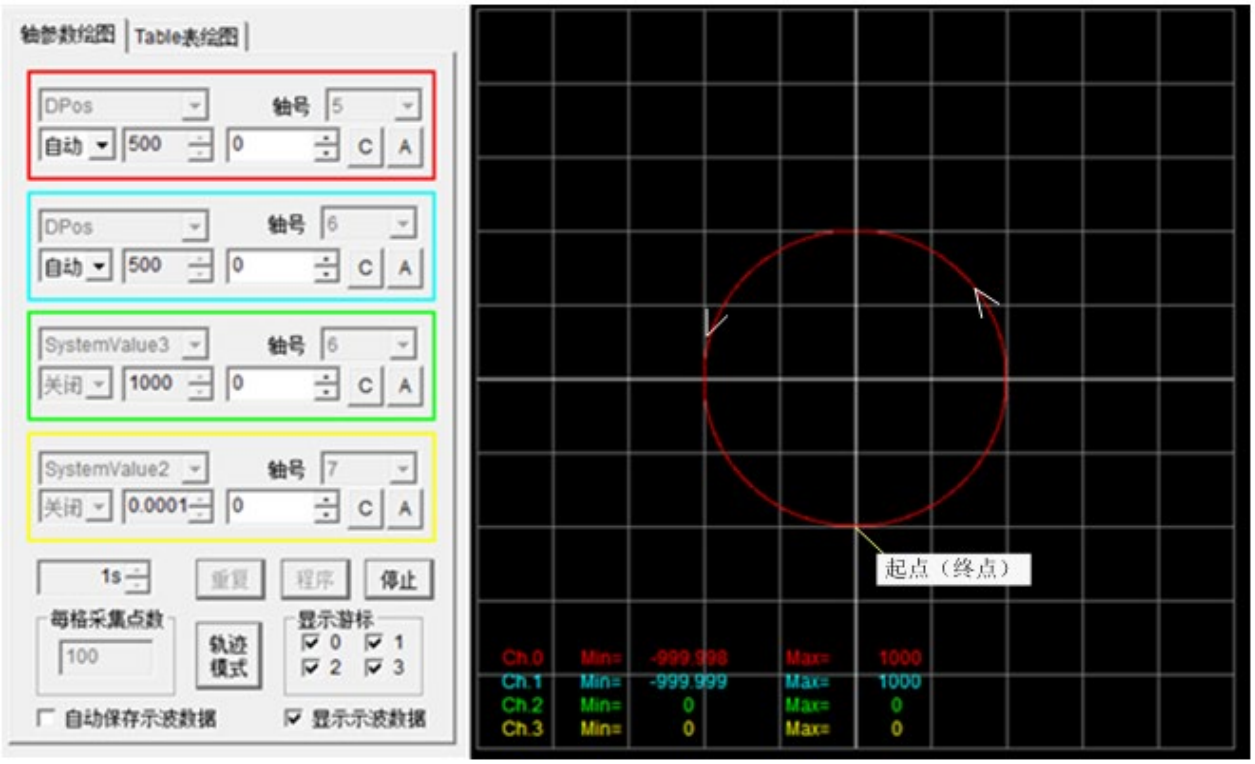
5.5.7.1.5 示例

为说明 C 轴的参数设定对跟随偏差的影响，现对一些典型情况的处理举例如下。设切向追踪刀具旋转轴 C 轴为 7 号轴，跟随 5、6 号轴的插补轨迹的切向角。设定 7 号轴为循环模式 2，循环行程区间为[-10000,10000)。

首先投送指令 hmc_movetang(7,5,6,0)，主轴 5、6 执行如下不同的指令。

1. 主轴 XY 执行圆弧插补

投送 hmc_movecircle(4,5,6,1,0,0,0,1000)执行圆弧插补，半径为 1000。插补 Speed = 1000，accel = decel = 2000。圆弧轨迹如下：



XY 轴插补轨迹

可算得执行圆弧插补过程中一些关键的运动参数（C 轴单位下）：

切向角变化的最大速度 = $\frac{\frac{速度}{半径}}{2*\pi} * 行程长度 = \frac{\frac{1000}{1000}}{2*\pi} * 20000 = 3,183.09886$

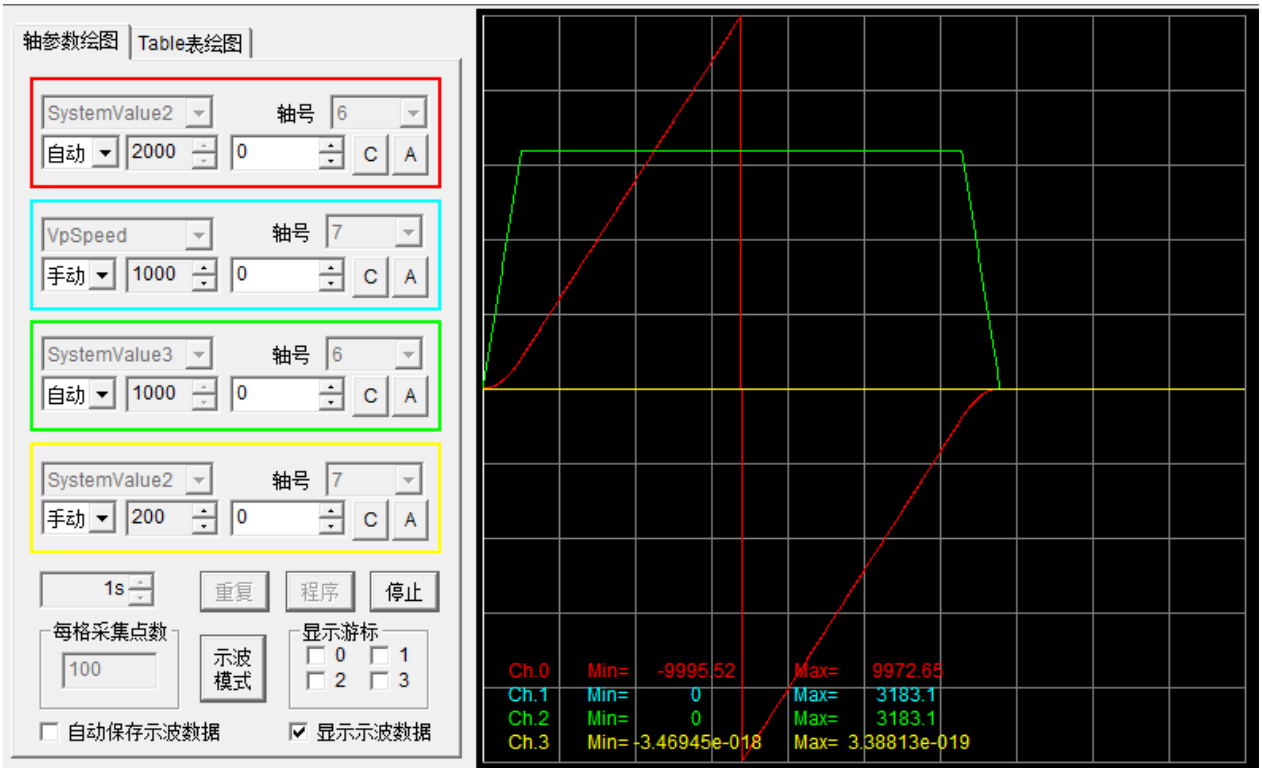
切向角变化的加速度 = $\frac{\frac{加速度}{半径}}{2*\pi} * 行程长度 = \frac{\frac{2000}{1000}}{2*\pi} * 20000 = 6,366.19772$

切向角变化的减速度 = $\frac{\frac{减速度}{半径}}{2*\pi} * 行程长度 = \frac{\frac{2000}{1000}}{2*\pi} * 20000 = 6,366.19772$

C 轴加减速及速度参数若大于以上计算参数，便可实现无差跟随。不同的 C 轴参数的运动效果如下图所示，红色曲线为 moveTang 执行过程中的实时目标角度位置，青色为刀具旋转轴 C 轴 VpSpeed，绿色为目标切向角的变化速度，黄色为 moveTang 执行过程中指令解算位置与理论目标角度位置的差值（随动偏差）。

（1） accel = decel =10000，speed = 10000

由下图可知此时 C 轴可实现无差跟随，C 轴 VpSpeed 曲线与切向角的变化速度曲线完全重合。红色曲线的跃变是由于循环行程边界处的折算。



圆弧插补切向追踪实时数据（C 轴加减速、最大速度 Speed 足够）

红色：目标切向角

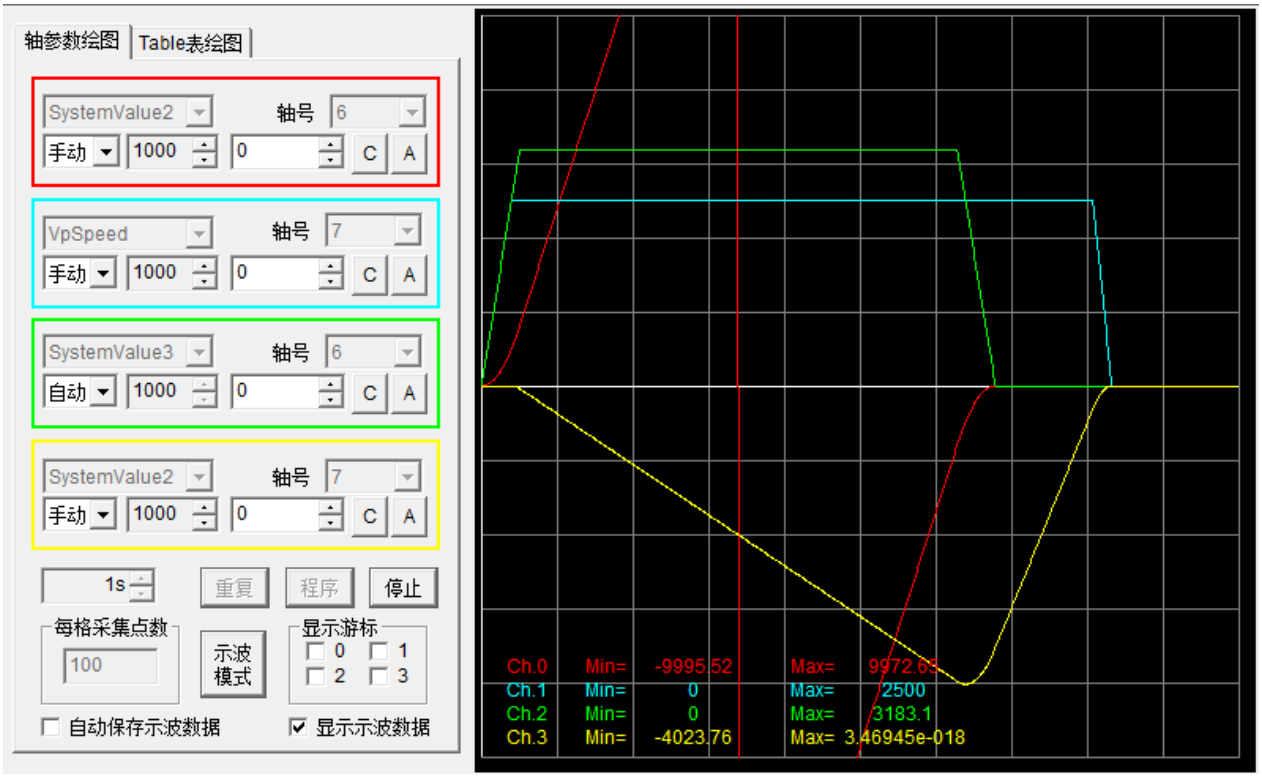
青色：刀具旋转轴 VpSpeed

绿色：目标切向角变化速度

黄色：解算位置与目标切向角的差值

(2)　accel = decel =10000，speed = 2500

此时 C 轴最大速度小于切向角最大变化速度，无法实现无差跟随。



圆弧插补切向追踪实时数据（C 轴最大速度 Speed 过小）

红色：目标切向角

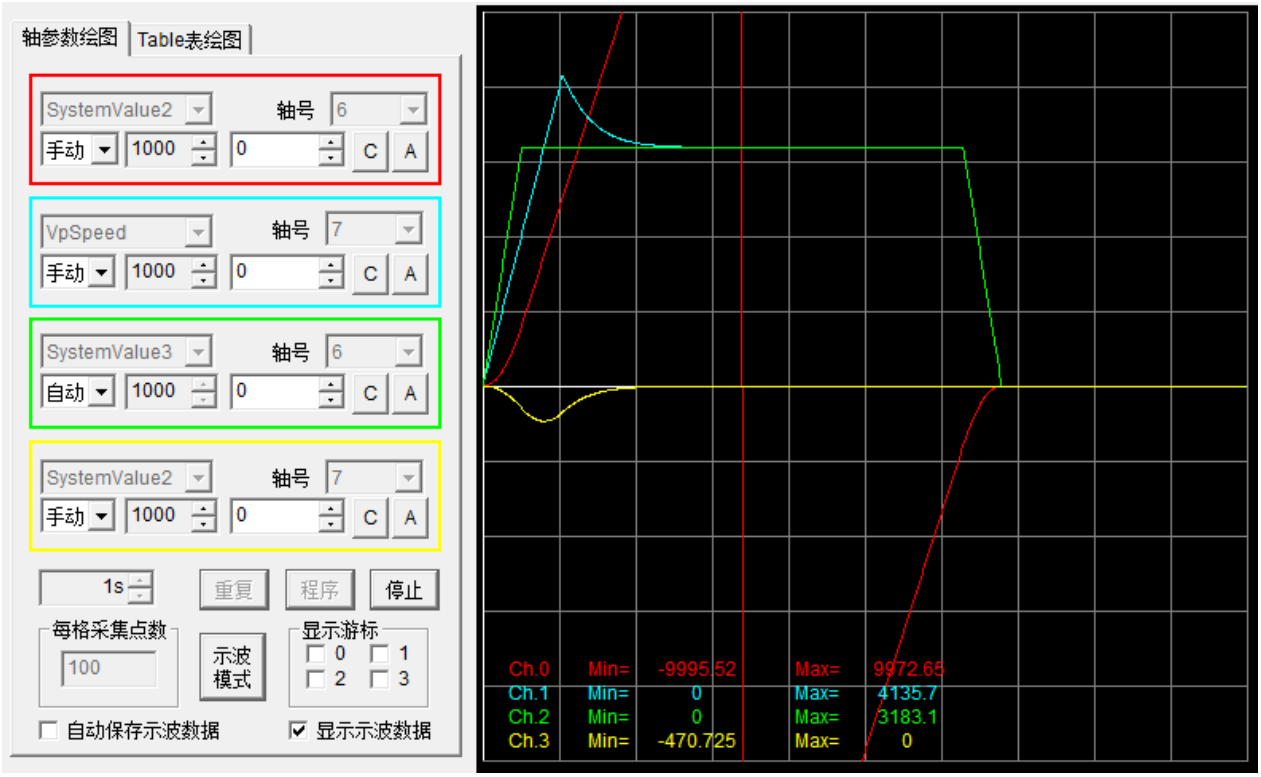
青色：刀具旋转轴 VpSpeed

绿色：目标切向角变化速度

黄色：解算位置与目标切向角的差值

(3)　accel = 4000，decel =10000，speed = 10000

此时 C 轴最大加速度小于切向角变化的最大加速度，加速阶段无法实现无差跟随。



圆弧插补切向追踪实时数据（C 轴加速能力不足）

红色：目标切向角

青色：刀具旋转轴 VpSpeed

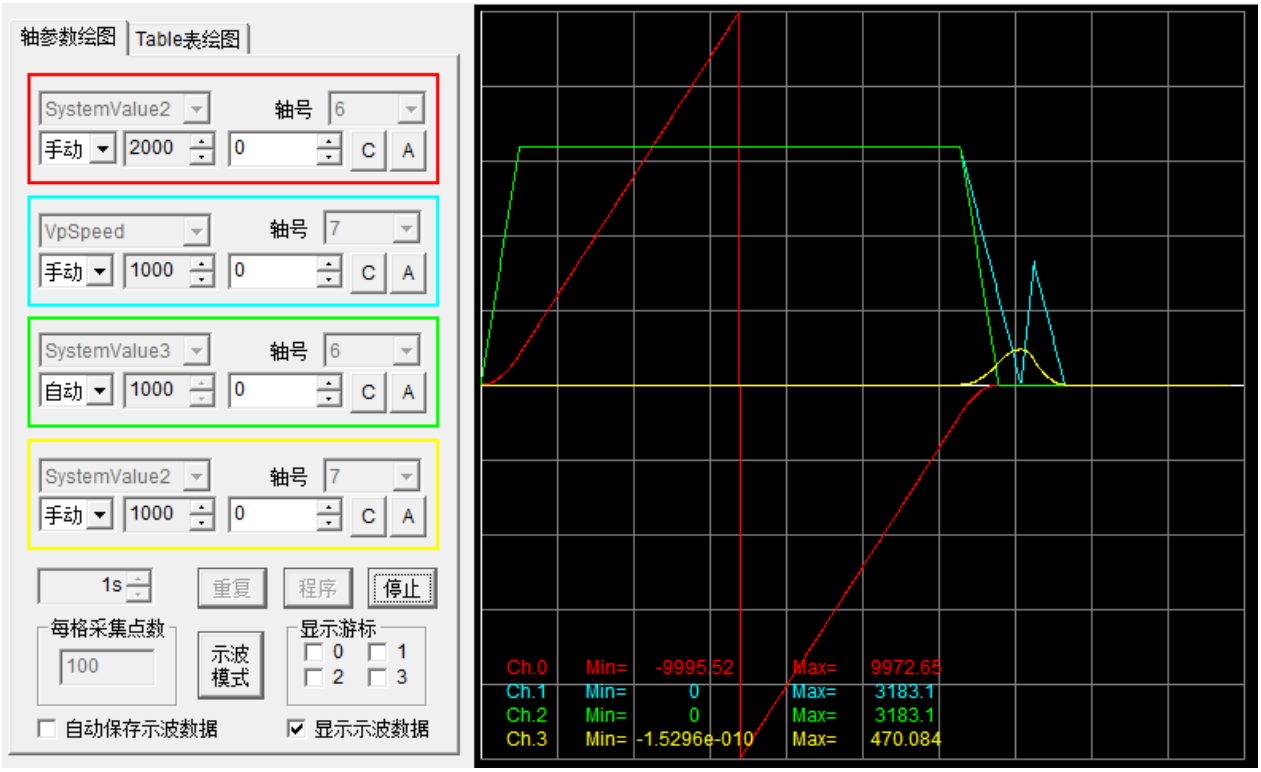
绿色：目标切向角变化速度

黄色：解算位置与目标切向角的差值

(4)　accel = 10000，decel =4000，speed = 10000

此时 C 轴最大减速度小于切向角变化的最大减速度，减速阶段无法实现无差跟随。

因此最后减速阶段 C 轴首先减速至 0（已经超过目标位置），再反向定位到目标位置。



圆弧插补切向追踪实时数据（C 轴减速能力不足）

红色：目标切向角

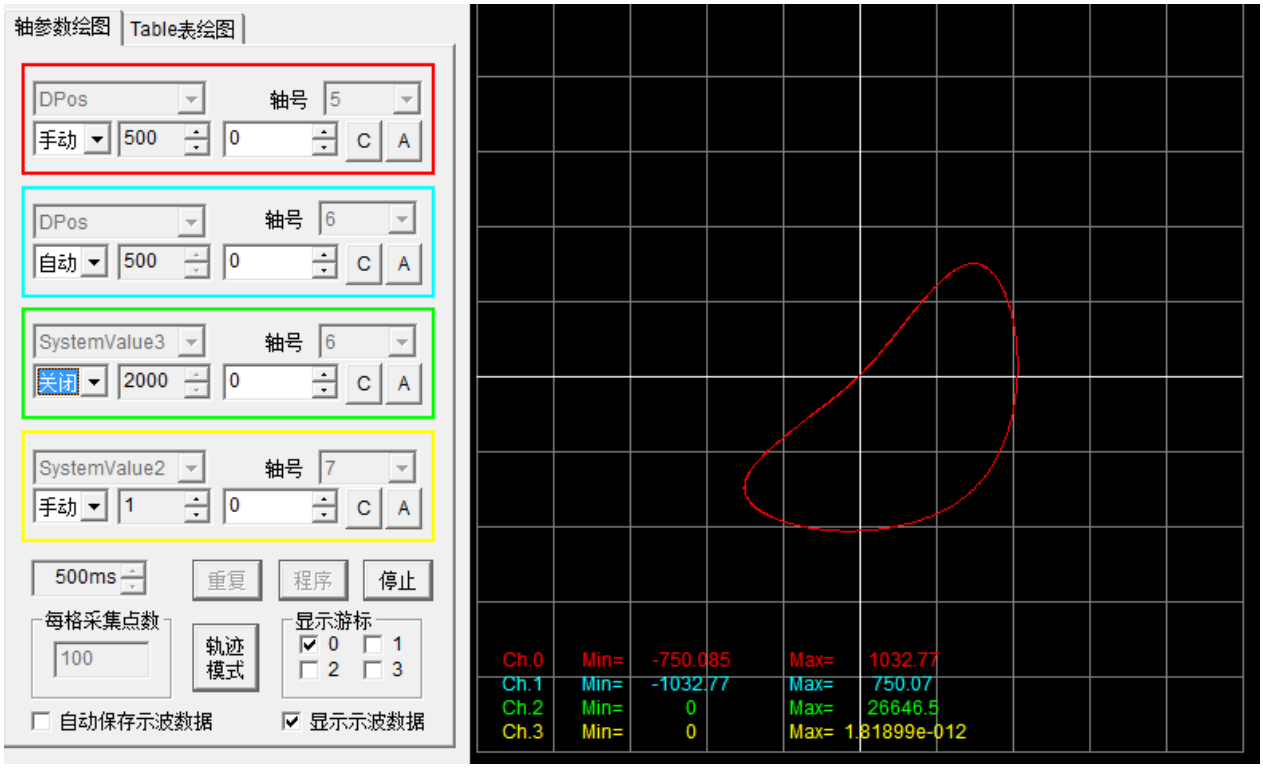
青色：刀具旋转轴 VpSpeed

绿色：目标切向角变化速度

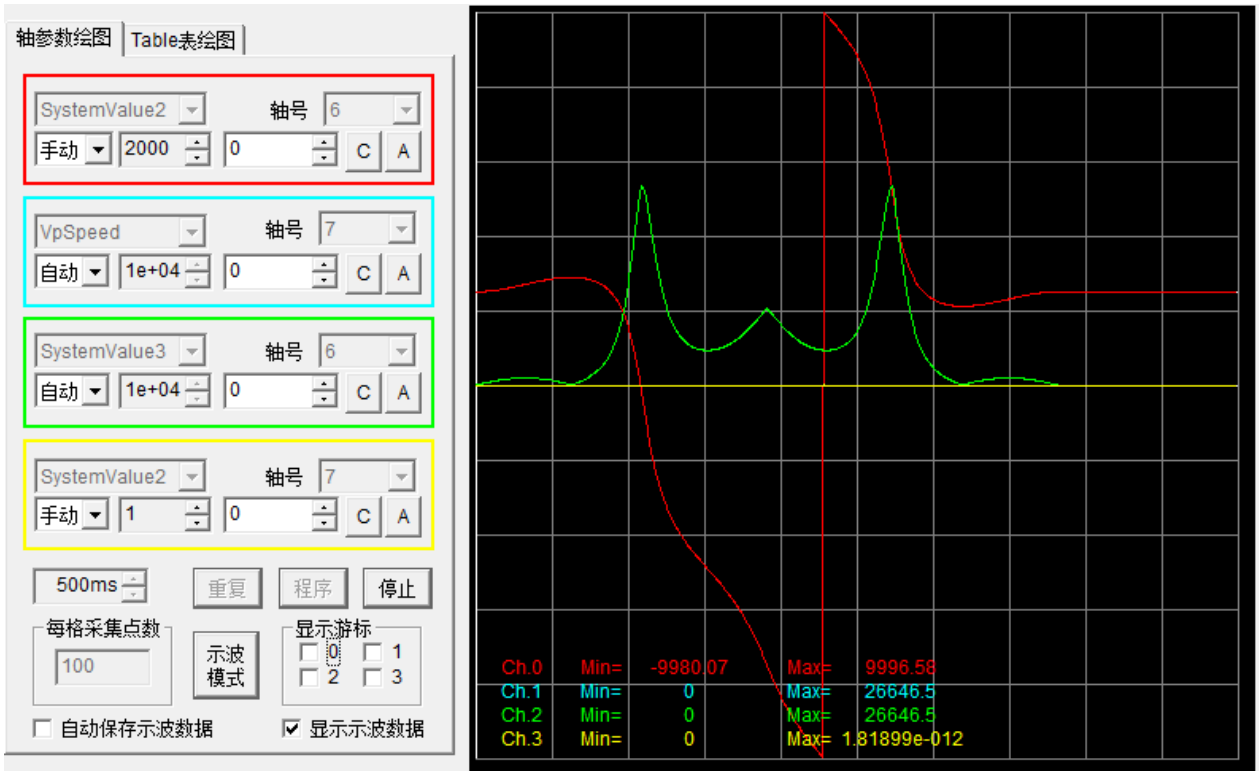
黄色：解算位置与目标切向角的差值

2. 主轴 XY 执行样条插补

样条插补过程中的切向角度变化的速度及加减速均为变值，最大加减速及最大速度无法精确计算得出。只要 C 轴加减速及速度参数大于样条插补过程中的切向角度变化的速度及加减速，便可实现无差跟随。如图所示：



XY 轴 2D 样条插补轨迹



样条插补切向追踪实时数据

红色：目标切向角

青色：刀具旋转轴 VpSpeed

绿色：目标切向角变化速度

黄色：解算位置与目标切向角的差值

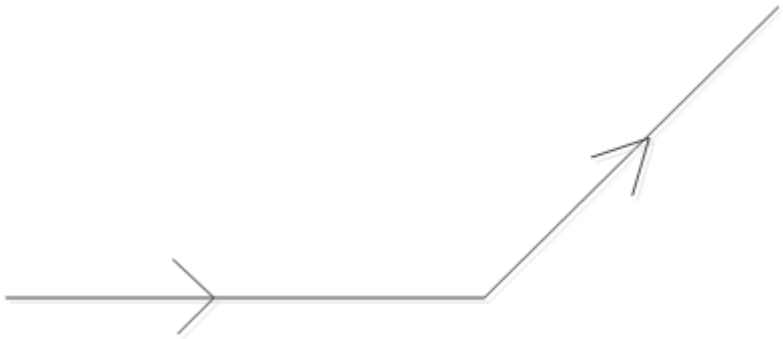
3. 曲线衔接处的处理

在曲线衔接处，若目标切向角或目标切向角变化速度发生阶跃变化，这时 C 轴将按照自身设定的加减速及 speed 朝目标切向角运行，所以不会产生运动冲击，但此处 C 轴实际位置与目标切向角位置会产生一定的偏差，且该偏差不可避免。

设切向追踪刀具旋转轴 C 轴为 7 号轴，跟随 5、6 号轴的合位移的切向角。设定 7 号轴为循环模式 2，循环行程区间为[-10000,10000)。首先投送指令 hmc_movetang(7,5,6,0)。

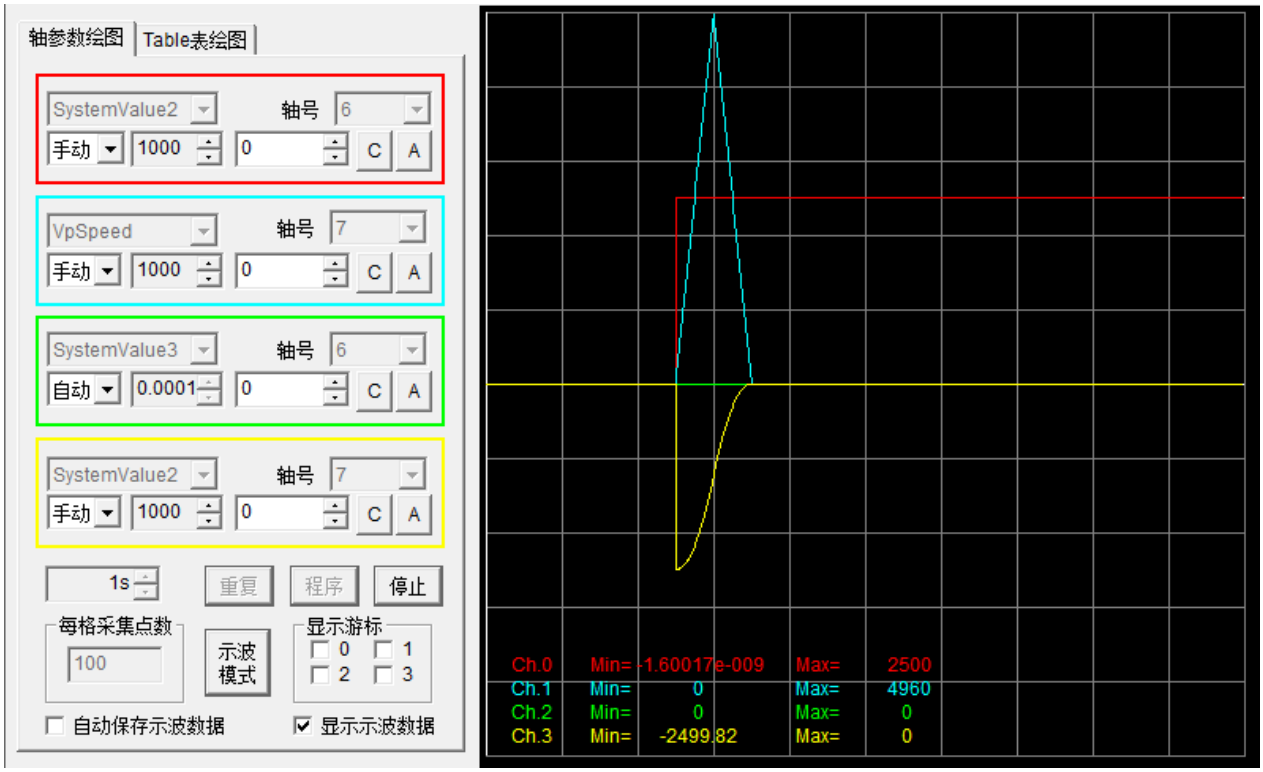
这里仅举例说明直线之间的衔接。折线处由于目标角度位置发生阶跃变化，这时 C 轴将以自身的加减速朝目标位置及目标速度运行，MoveTang 刀具旋转轴无法无差跟随。

投送指令：hmc_move2d(4,5,6,2000,000); hmc_move2d(4,5,6,2000,2000); 轨迹如下：



XY 轴运行轨迹

C 轴参数：accel = decel = 10000，speed = 10000。运行轨迹如下：



切向追踪实时数据（折线，Merge 开启）

红色：目标切向角

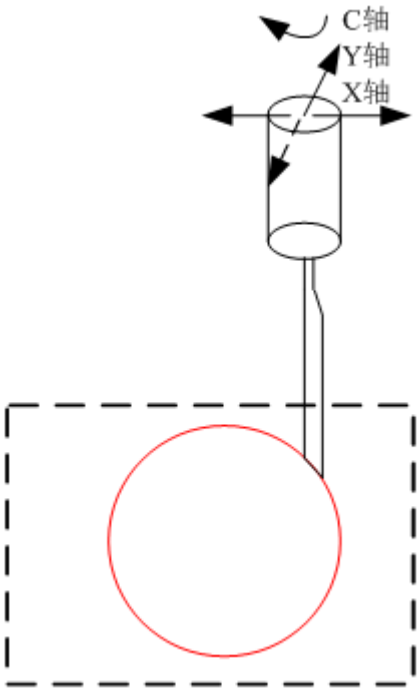
青色：刀具旋转轴 VpSpeed

黄色：解算位置与目标切向角的差值

4. 切割应用场景举例

使用切刀切割一个平面圆弧，为保证切割效果，降低切刀磨损程度，应使得在整个切割过程中实时调整切刀刀刃跟随圆弧轨迹的切线方向变化。

X、Y 轴驱动切刀在 XY 平面执行圆弧插补运动，切割物料；C 轴驱动切刀进行旋转，控制切刀方向跟随切割轨迹实时调整角度：

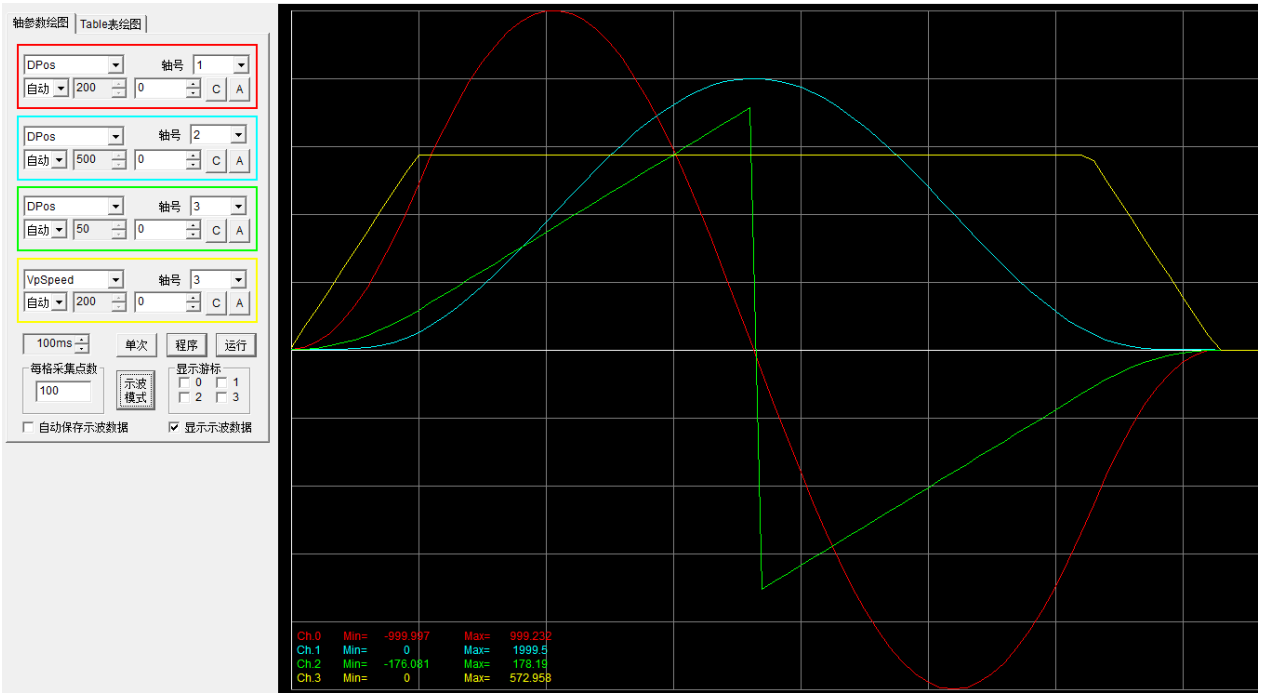


切割示意图

则 AT 中编写程序如下：

```
XAxisID := 1;       (*圆弧插补分轴 X*)

YAxisID := 2;       (*圆弧插补分轴 Y*)

CAxisID := 3;       (*切刀旋转轴 C*)


Axis3.RepMode := 2;     (*设置切刀旋转轴为循环行程*)

Axis3.RepDist := 180;         (*循环行程范围为-180~180，则 C 轴位置显示为旋转角度*)


HMC_MoveCircle(0,XAxisID,YAxisID,1,0,0,0,1000);  (*X、Y 轴执行圆弧插补切割*)

HMC_MoveTang(CAxisID,XAxisID,YAxisID,0);        (*C 轴跟随 X、Y 轴轨迹旋转，调整切刀角度*)
```

实际执行结果如下：



运动示意图

可以看出，当圆弧插补轨迹执行至 1/4 圆弧时，切刀角度为 90 度，圆弧插补继续执行至 1/2 圆弧时，切刀角度为 180 度，圆弧插补继续执行至 3/4 圆弧时，切刀角度为-90 度，圆弧插补执行结束整圆时，切刀角度为 0 度。整个插补过程中，切刀旋转角度实时跟随合运动轨迹与 X 轴正向夹角，及刀具方向与切割轨迹的切向方向（切割速度矢量方向）保持一致，达到最好的切割效果。

5.5.7.2 HMC_MoveTangConfigCornerMode（切向追踪拐角提刀设定）

5.5.7.2.1 功能

当两条指令间衔接处具有角度的跃变时（如折线），可执行拐角提刀功能，以保护刀具，保证加工效果.

提供以下三种抬刀模式供用户选择，按照自动化程度由高到低分别为：自动抬刀模式、手动抬刀模式、不抬刀模式。

5.5.7.2.1.1 自动抬刀模式 ucCornerMode=2

1. 模式介绍

提前判断若指令之间衔接处的转折角大于预设的提刀角度阈值，则会自动执行如下抬刀动作流程：

等待插补主轴 XY、刀具升降轴 Z 停止—>Z 轴提升至安全高度—>C 轴旋转至下条指令的起始切向角—>Z 轴下降至原位置—>XYZ 轴恢复运行

支持任意插补指令之间的衔接。由于非插补类指令的起始切向角不可预判，若 XY 的指令队列中存在非插补类指令，则在该指令的衔接处按照不抬刀处理。

支持以下两种情形：

主轴 XY 为同一插补指令的两个分轴。典型应用为 XY 轴执行二维平面插补指令。

主轴 XY、刀具提升轴 Z 为同一插补指令的三个分轴。典型应用为 XYZ 轴执行三维空间插补指令。

为保证在满足拐角提刀的指令衔接处 XY 轴可靠停止，当 merge 开启时，需保证:

merge 停止角 ≤ 拐角提刀角度阈值

2. 配置流程

（1） 使用 HMC_MoveTang (ucAxisC，ucAxisX， ucAxisY， dOffset) 投送切向追踪指令；

（2） 使用 HMC_MoveTangConfigCornerMode (ucAxisC，ucCornerMode，dCornerAngleLimit，ucAxisZ，dSafePosZ，ucSafePosZMode) 配置拐角抬刀模式为自动抬刀模式， 同时配置 拐角角度阈值、刀具升降轴 Z 轴轴号、Z 轴抬刀高度、Z 轴抬刀高度坐标模式（增量高度/绝对高度）；

（3） 往主轴 XY 中投送二维平面插补指令，或往轴 XYZ 中投送三维空间插补指令。

3. 补充说明

（1） HMC_MoveTang 指令运行过程中可调用 HMC_GetMoveTangCornerState(ucAxisC)获取实时运行状态（正常运行状态、拐角抬刀状态）；

（2） 若在拐角抬刀状态下使用 HMC_Cancel 撤销 HMC_MoveTang 指令，则指令会同时控制 C 轴、Z 轴按照其自身的减速度停止。

（3） 由于在拐角抬刀状态下 XYZ 指令队列中的指令处于暂停状态，出于安全考虑，撤销完成后并不自动恢复 XYZ 轴中指令队列中的指令运行，若需要恢复 XYZ 中指令队列中后续指令的运行，用户需调用 HMC_SetCmdBufferLoadMode(ucAxisID，ucCmdLoadMode)把指令加载模式设定为正常

加载模式（0）；

（4） 若在拐角抬刀状态下 Z 轴遇到限位，则 Z 轴将以 MAX{decle、FastDecel}减速停止，出于安全考虑，HMC_MoveTang 自动切换至手动抬刀模式。

5.5.7.2.1.2 手动抬刀模式 ucCornerMode=1

1. 模式介绍

当指令之间衔接处的转折角大于预设的拐角提刀角度阈值时，XY 轴自动停止，等待用户在 IEC 程序中处理 Z 轴抬刀及 C 轴方向调整动作，抬刀动作流程完成后 XY 轴继续运行后续指令。手动模式下一个完整的拐角抬刀的处理流程如下（黄色部分为用户 IEC 程序）：

LX 控制器：

检测到拐角抬刀条件满足，插补主轴 XY 停止等待。

IEC 周期任务中（用户编写程序）：

（1） 调用 HMC_GetMoveTangCornerState 查询到 MoveTang 处于手动抬刀模式下的拐角抬刀等待状态；

（2） 使用 HMC_Move 提升 Z 轴至安全高度；

（3） 等待 Z 轴提升动作完成；

（4） 调用 HMC_MoveTangCornerFollowNextCmd 旋转 C 轴至下条指令的起始切向角；

（5） 等待 C 轴旋转动作完成；

（6） 使用 HMC_Move 驱动 Z 轴下降至原位置；

（7） 等待 Z 轴下降动作完成；

（8） 调用 HMC_MoveTangCornerContinue 恢复 XY 轴运行后续指令。

支持任意插补指令之间的衔接。由于非插补类指令的起始切向角不可预判，若 XY 的指令队列中存在非插补类指令，则在该指令的衔接处按照不抬刀处理。

支持情形：主轴 XY 为同一插补指令的两个分轴。典型应用为 XY 轴执行二维平面插补指令。

为保证在满足拐角提刀的指令衔接处 XY 轴可靠停止，当 merge 开启时，需保证：

merge 停止角 ≤ 拐角提刀角度阈值

2. 配置流程

（1） 使用 HMC_MoveTang (ucAxisC，ucAxisX， ucAxisY， dOffset) 投送切向追踪指令；

（2） 使用 HMC_MoveTangConfigCornerMode (ucAxisC，ucCornerMode，dCornerAngleLimit，ucAxisZ，dSafePosZ，ucSafePosZMode) 配置拐角抬刀模式为手动抬刀模式， 同时配置 拐角角度阈值；

（3） 在一个周期性 IEC 任务中，调用 HMC_GetMoveTangCornerState(ucAxisC)反复查询切向追踪执行状态。若返回值为 手动抬刀模式下的拐角停止状态（2），则执行以下 IEC 程序：

（4） 使用 HMC_Move(ucAxisZ,dSafePos)驱动 Z 轴抬刀至安全平面；

（5） 等待 Z 轴到达安全平面后，调用 HMC_MoveTangCornerFollowNextCmd(AxisC)，C 轴会运行至下条指令的起始切向角位置；

（6） 等待 C 轴到达下条指令的起始切向角位置后，使用 HMC_Move(ucAxisZ，-dSafePos)驱动 Z 轴落刀至原位置；

（7） 等待 Z 轴到达原位置后，调用 HMC_MoveTangCornerContinue (AxisC)恢复切向追踪主轴的运行。

（8） 往主轴 XY 中投送插补指令；

3. 补充说明

（1） 与自动抬刀模式不同，手动抬刀模式不支持 主轴 XY、刀具提升轴 Z 作为同一插补指令的三个分轴的情形；

（2） 若在拐角停止状态下使用 HMC_Cancel 撤销 HMC_MoveTang 指令，由于在拐角抬刀状态下 XY 指令队列中的指令处于暂停状态，出于安全考虑，撤销完成后并不自动恢复 XY 轴中指令队列中的指令运行，若需要恢复 XY 中指令队列中后续指令的运行，用户需调用 HMC_SetCmdBufferLoadMode(ucAxisID，ucCmdLoadMode)把指令加载模式设定为正常加载模式（0）。

5.5.7.2.1.3 不抬刀模式 ucCornerMode=0

不作任何处理，拐角处插补轴 XY 不会停止等待。

5.5.7.2.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
ucCornerMode	Input	USINT	拐角抬刀模式（0：不抬刀； 1：拐角手动抬刀； 2：拐角自动抬刀）
dCornerAngleLimit	Input	LREAL	拐角抬刀角度阈值（单位：弧度）
ucAxisZ	Input	USINT	拐角自动抬刀 Z 轴轴号 注：ucCornerMode 为 2 时生效。
dSafePosZ	Input	LREAL	拐角自动抬刀 安全提刀高度。 注：ucCornerMode 为 2 时生效。
ucSafePosZmode	Input	USINT	拐角自动提刀 安全提刀高度 坐标模式（0:增量坐标，1:绝对坐标）。 注：ucCornerMode 为 2 时生效。
HMC_MoveTangConfigConrnerMode	Output	USINT	0：设置成功 其他：错误码

5.5.7.2.3 指令类型

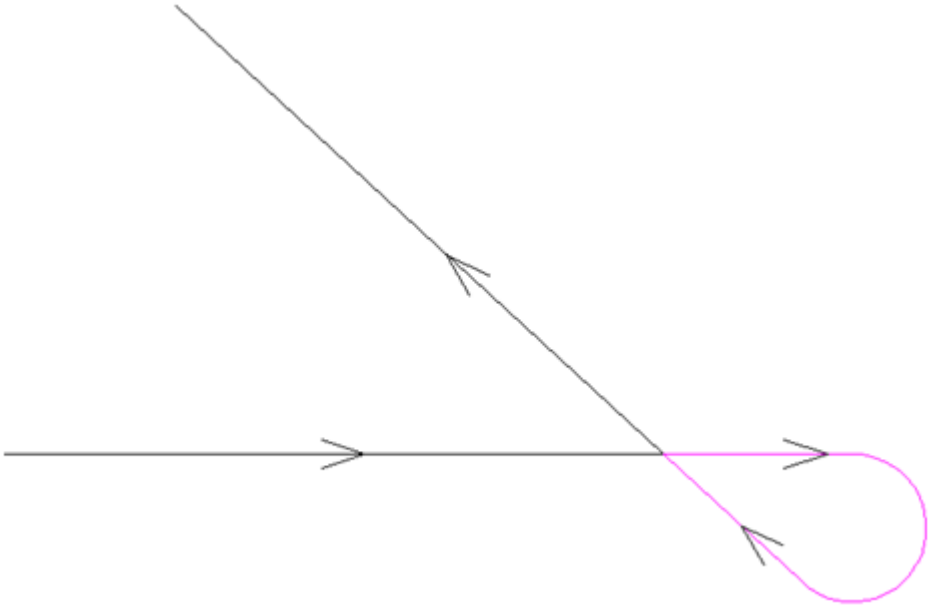
非轴运动缓存指令。

5.5.7.2.4 补充说明

如果拐角抬刀动作不能满足用户要求，用户可在拐角处添加工艺路径，以满足加工要求。

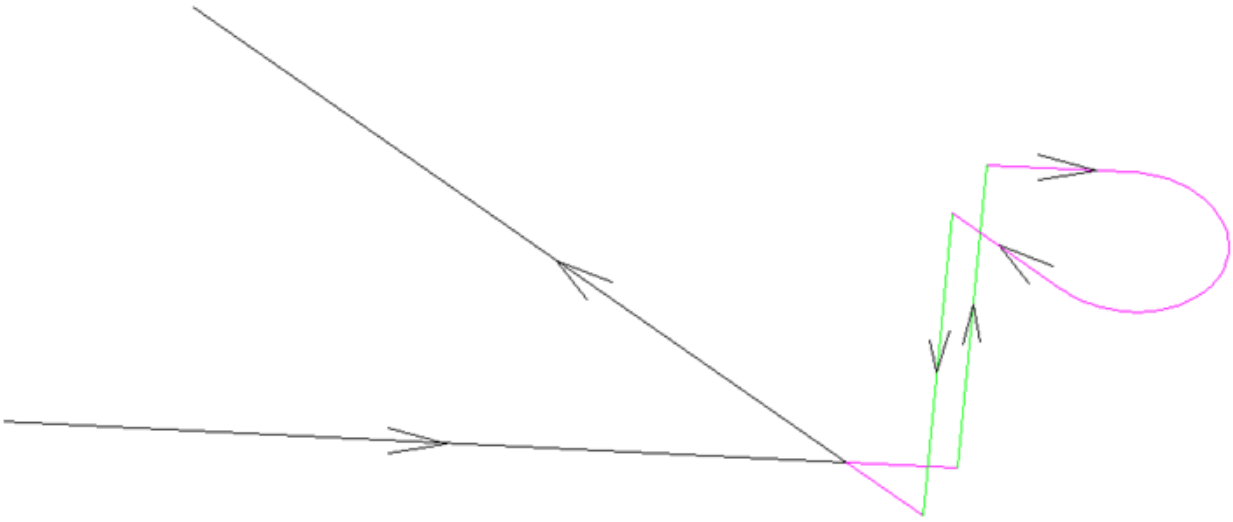
（1） 圆弧工艺路径

在工艺允许的情况下，可在拐角处加入圆弧辅助工艺路径，提高加工效率，避免频繁抬刀,同时亦可提高拐角处的切割质量。黑色折线为加工路径，紫色部分工艺路径。



圆弧工艺路径示意图

（2） 带抬刀的工艺路径



带抬刀的工艺路径示意图

如果用户希望在拐角处刀具的加工路线延出一些，以保证拐角处的切割质量，同时又希望尽可能的降低因工艺路径的引入而对切割原料造成的浪费，则可在（1）中工艺路径中插入 Z 向抬刀路径，如绿色部分。

5.5.7.2.5 示例

（1） 示例 1：自动抬刀

插补指令轨迹如下，在切割过程中为保护设备，需在指令拐角大于等于 π/4 时抬刀。现在分别以自动抬刀和手动抬刀两种方式为例，实现抬刀保护功能。

投送 HMC_MoveTang 指令，为控制设备在拐角大于等于 π/4 处自动抬刀，设定拐角抬刀的角度阈值为 π/4-10-6，10-6 为安全余量，因程序中可能会存在浮点数小量偏差，余量大小用户可根据精度要求酌情选取。ST 语言程序如下，主轴 XY 插补轨迹如下图 1 所示，各阶段运动过程曲线如下图 2 所示。

```
AxisXY_ID:=4;(*插补合轴轴号*)

AxisX_ID:=5; (*X 轴轴号*)

AxisY_ID:=6; (*Y 轴轴号*)

AxisZ_ID:=7; (*Z 轴轴号*)

AxisC_ID:=8; (*C 轴轴号*)

tangentAngleOffset:=0; (*切向角偏移量*)

cornerMode :=2; (*拐角自动提刀模式*)

cornerAngleLimit := ATAN(1) - 1E-6; (*拐角抬刀的角度阈值为 45 度, 1E-6 为安全余量*)

safePosZ:= 4000; (*拐角抬刀安全高度*)

SafePosZMode:= 0; (*拐角抬刀安全高度为增量坐标*)


HMC_MoveTang(AxisC_ID, AxisX_ID, AxisY_ID, tangentAngleOffset); (*投送切向追踪指令*)

HMC_MoveTangConfigCornerMode(AxisC_ID, cornerMode, cornerAngleLimit, AxisZ_ID, safePosZ, SafePosZMode);  (*配置拐角提刀模式*)


Axis4.Merge := 1; (*开启 merge*)

HMC_SetMergeAngle (AxisXY_ID, ASIN(0.5), cornerAngleLimit); (*配置 merge 减速角、停止角*)
```

(**向主轴投送插补指令（圆弧+Move2d）**)

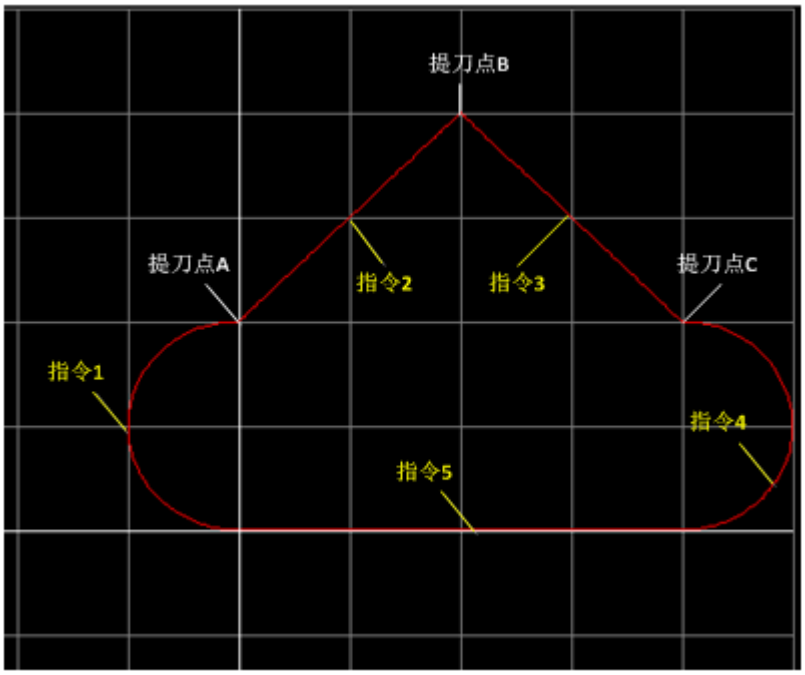
HMC_MoveCircle(AxisXY_ID, AxisX_ID, AxisY_ID,3,0,2000,0,1000);

HMC_Move2d(AxisXY_ID, AxisX_ID, AxisY_ID,2000,2000);

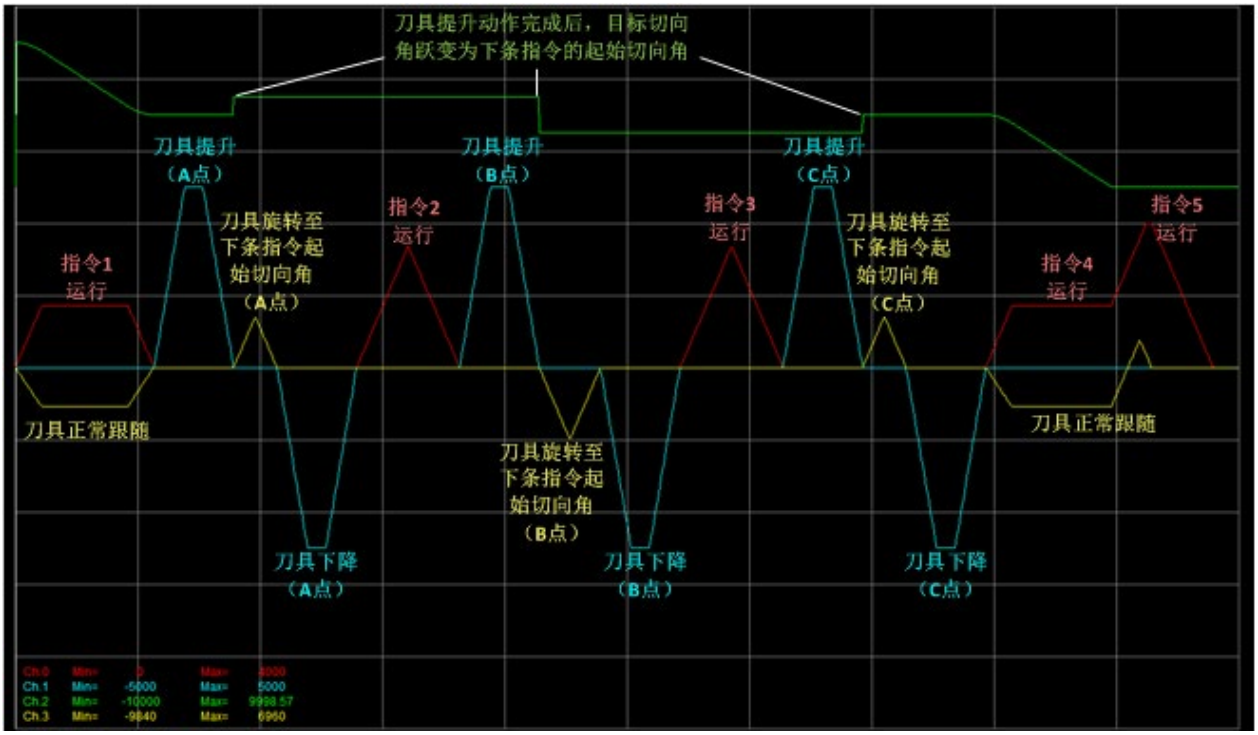
HMC_Move2d(AxisXY_ID, AxisX_ID, AxisY_ID,2000,-2000);

HMC_MoveCircle(AxisXY_ID, AxisX_ID, AxisY_ID,3,0,-2000,0,-1000);

HMC_Move2d(AxisXY_ID, AxisX_ID, AxisY_ID,-4000,0);



切向追踪主轴 XY 插补轨迹



切向追踪自动提刀模式运动过程曲线（主轴 XY 执行 圆弧+Move2d）

绿色：目标切向角

红色：XY 插补合轴 MpSpeed 曲线

黄色：刀具旋转轴 C 轴 MpSpeed 曲线

青色：刀具升降轴 Z 轴 MpSpeed 曲线

从上图可以看出，指令衔接处，当目标切向角变化大于设定抬刀阈值时（前四条指令间拐角均大于抬刀阈值），插补 X、Y 轴会自动停止（红色曲线为插补合轴速度曲线），刀具提升轴 Z 轴开始提到至设定位置（青色曲线为刀具提升轴速度），然后切刀旋转轴旋转至下条指令的起始切向角，刀具提升轴 Z 轴再下降至原位，下条插补指令继续恢复执行。指令衔接处目标切向角变换小于设定抬刀阈值时（第四、五条指令间拐角为 0 度），则不进行抬刀动作，不停止插补轴运动。

若把上面的 move2d 指令用 Move3d 代替，即 XYZ 轴执行圆弧+3 轴插补指令，则执行效果如下图所示。由图可知 Z 轴作为 3 轴插补分轴的插补运动与 Z 轴的提刀动作分阶段运行，没有冲突。

(**向主轴投送插补指令(圆弧+Move3d)**)

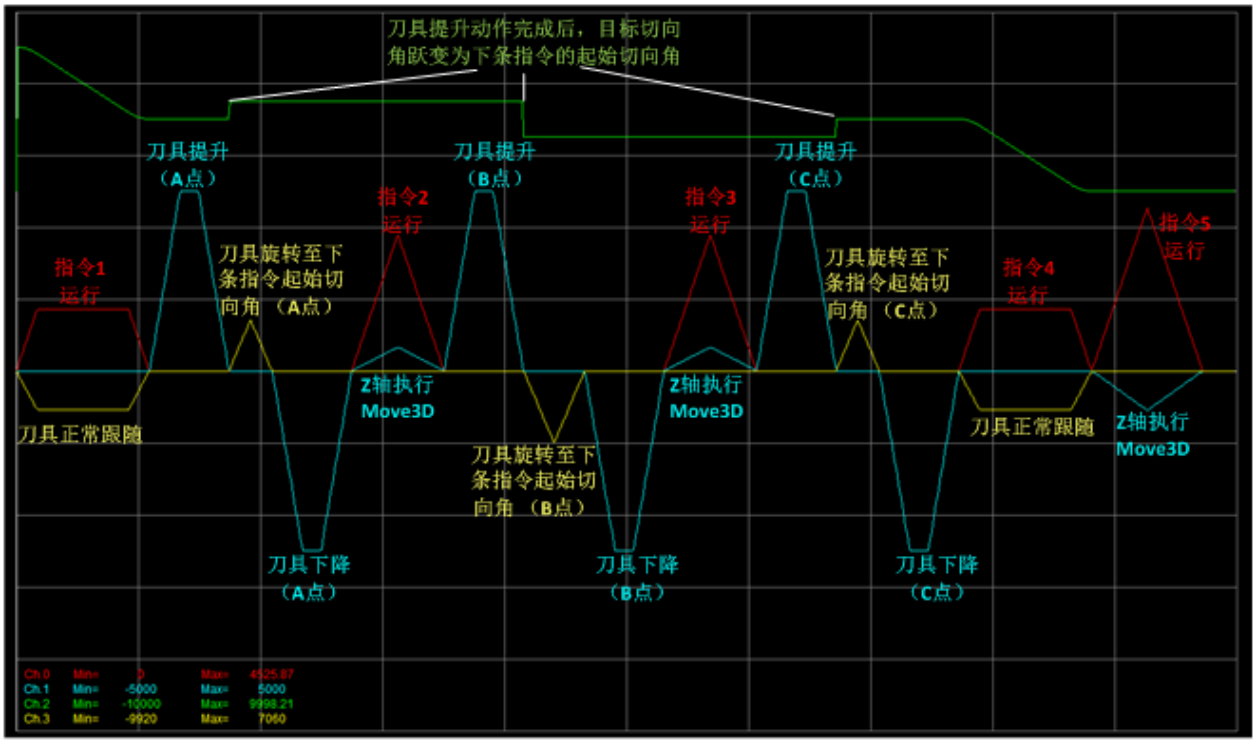
HMC_MoveCircle(AxisXY_ID, AxisX_ID, AxisY_ID,3,0,2000,0,1000);

HMC_Move3d(AxisXY_ID, AxisX_ID, AxisY_ID, AxisZ_ID,2000,2000,500);

HMC_Move3d(AxisXY_ID, AxisX_ID, AxisY_ID, AxisZ_ID,2000,-2000,500);

HMC_MoveCircle(AxisXY_ID, AxisX_ID, AxisY_ID,3,0,-2000,0,-1000);

HMC_Move3d(AxisXY_ID, AxisX_ID, AxisY_ID, AxisZ_ID,-4000,0,-1000);



切向追踪自动提刀模式运动过程曲线（XYZ 执行 圆弧+Move3d）

绿色：目标切向角

红色：XYZ 插补合轴 MpSpeed 曲线

黄色：刀具旋转轴 C 轴 MpSpeed 曲线

青色：刀具升降轴 Z 轴 MpSpeed 曲线

(2) 示例 2：手动抬刀

XY 插补路径同（1）。往 C 轴中投送 HMC_MoveTang 指令，设定为手动抬刀模式，其它参数同（1）中设置。ST 程序如下。主轴 XY 插补轨迹如下图 1 所示，各阶段运动过程曲线如下图 2 所示。由图可知与（1）中自动提刀模式下的结果完全相同。

任务 1 程序：

AxisXY_ID:=4;(*插补合轴轴号*)

AxisX_ID:=5; (*X 轴轴号*)

AxisY_ID:=6; (*Y 轴轴号*)

AxisZ_ID:=7; (*Z 轴轴号*)

AxisC_ID:=8; (*C 轴轴号*)

tangentAngleOffset:=0; (*切向角偏移量*)

cornerMode :=1; (*拐角手动提刀模式*)

cornerAngleLimit := ATAN(1) - 1E-6; (*拐角抬刀的角度阈值为 45 度, 1E-6 为安全余量*)

HMC_MoveTang(AxisC_ID, AxisX_ID, AxisY_ID, tangentAngleOffset); (*投送切向追踪指令*)

HMC_MoveTangConfigCornerMode(AxisC_ID, cornerMode, cornerAngleLimit, 0, 0, 0);(*配置拐角提刀模式*)

Axis4.Merge := 1; (*开启 merge*)

HMC_SetMergeAngle (AxisXY_ID, ASIN(0.5), cornerAngleLimit); (*配置 merge 减速角、停止角*)

(**向主轴投送插补指令**)

HMC_movecircle(AxisXY_ID, AxisX_ID, AxisY_ID,3,0,2000,0,1000);

HMC_move2d(AxisXY_ID, AxisX_ID, AxisY_ID,2000,2000);

HMC_move2d(AxisXY_ID, AxisX_ID, AxisY_ID,2000,-2000);

HMC_movecircle(AxisXY_ID, AxisX_ID, AxisY_ID,3,0,-2000,0,-1000);

HMC_move2d(AxisXY_ID, AxisX_ID, AxisY_ID,-4000,0);

任务 2 程序（周期任务）：

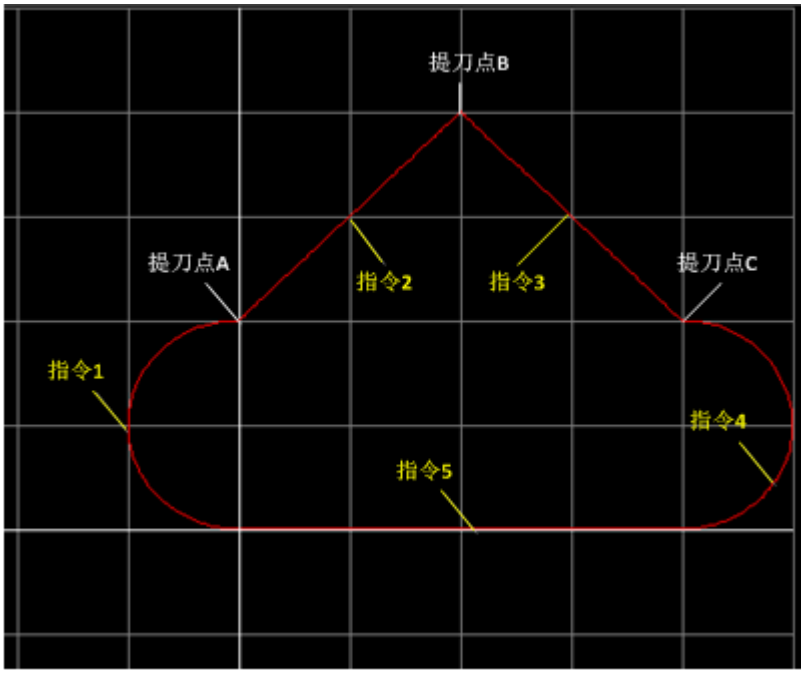
ManualModeState_Pause:=2; (*拐角抬刀等待状态*)

safePosZ:= 4000; (*拐角抬刀安全高度*)

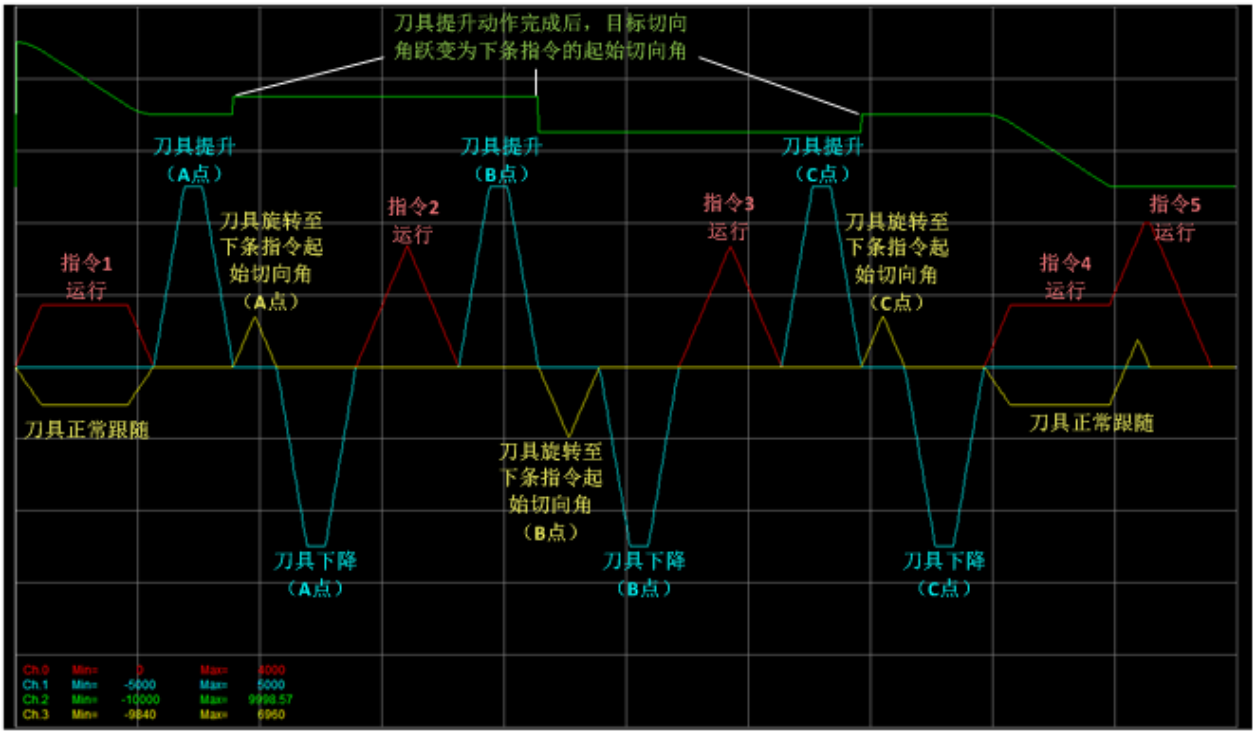
IF (HMC_GetMoveTangCornerState(AxisC_ID) = ManualModeState_Pause) THEN

(*主轴 XY 处于拐角停止状态，等待提刀处理*)

```
    CmdID_AxisZ := HMC_Move(AxisZ_ID,safePosZ);(*Z 轴提刀*)

    HMC_WaitCmdFinish(CmdID_AxisZ);(*等待 Z 轴提刀完成*)
    HMC_MoveTangCornerFollowNextCmd(AxisC_ID);  (*C 轴运动至下一条指令的起
始切向角方向*)

    WHILE (HMC_GetMoveTangDestDirection(AxisC_ID) <> Axis8.dPos ) DO
(*等待 C 轴动作完成*)
        ;
    END_WHILE

    CmdID_AxisZ := HMC_Move(AxisZ_ID,-safePosZ);(*Z 轴落刀*)

    HMC_WaitCmdFinish(CmdID_AxisZ);(*Z 轴落刀完成*)
    HMC_MoveTangCornerContinue(AxisC_ID) ; (*拐角提刀动作全部完成，插补轴恢复
运行*)
  END_IF
```



切向追踪主轴 XY 插补轨迹



切向追踪手动提刀模式运动过程曲线（主轴 XY 执行 圆弧+Move2d）

绿色：目标切向角

红色：XY 插补合轴 MpSpeed 曲线

黄色：刀具旋转轴 C 轴 MpSpeed 曲线

青色：刀具升降轴 Z 轴 MpSpeed 曲线

5.5.7.3 HMC_GetMoveTangCornerState（获取切向追踪拐角抬刀状态）

5.5.7.3.1 功能

获取切向追踪拐角抬刀状态。

5.5.7.3.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_GetMoveTangCornerState	Output	USINT	0：拐角不抬刀； 1：拐角手动抬刀，主轴 XY 正常运行状态； 2：拐角手动抬刀，主轴 XY 遇到拐角并暂停，等待工程程序处理抬刀； 3：拐角自动抬刀，主轴 XY 正常运行状态； 4：拐角自动抬刀，主轴 XY 遇到拐角并暂停，正在进行自动抬刀处理。 255：错误

5.5.7.3.3 指令类型

非轴运动缓存指令

5.5.7.3.4 补充说明

当前指令不为切向追踪指令时，返回错误。

5.5.7.4 HMC_MoveTangCornerFollowNextCmd（C 轴定位至下一条指令的起始切向角）

5.5.7.4.1 功能

用于切向追踪拐角手动抬刀模式，在拐角暂停等待抬刀的状态下，使用该指令可驱动 C 轴定位至下一条指令的起始切向角。

5.5.7.4.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_MoveTangCornerFollowNextCmd	Output	UDINT	0：设置成功 其他：错误码

5.5.7.4.3 指令类型

非轴运动缓存指令。

5.5.7.4.4 补充说明

若当前指令不为手动抬刀模式下的切向追踪指令，则返回错误。

5.5.7.5 HMC_MoveTangCornerContinue（切向追踪继续运行）

5.5.7.5.1 功能

用于切向追踪拐角手动抬刀模式，在拐角暂停等待抬刀的状态下，使用该指令可恢复切向追踪继续运行。

5.5.7.5.2 参数

参数	声明	数据类	说明

		型	
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_MoveTangCornerContinue	Output	UDINT	0：设置成功 其他：错误码

5.5.7.5.3 指令类型

非轴运动缓存指令。

5.5.7.5.4 补充说明

若当前指令不为手动抬刀模式下的切向追踪指令，则返回错误。

5.5.7.6 HMC_GetMoveTangDestDirection（获取切向追踪目标方向位置）

5.5.7.6.1 功能

该指令获取切向追踪的目标切向角，单位为 C 轴（ucAxisC，刀具旋转轴）用户单位。该指令执行时，C 轴当前指令需是 HMC_MoveTang 指令。

此时 C 轴工作在循环行程模式下，模式为 2(对称区间)，因此该指令返回的目标切向角范围为[-Repdist ,Repdist)，对应于[-π,π)的切向角度。

5.5.7.6.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_GetMoveTangDestdirection	Output	LREAL	切向追踪指令的目标切向角。若该指令调用错误时返回 0，并报用户错误。

5.5.7.6.3 指令类型

非轴运动缓存指令。

5.5.7.7 HMC_GetMoveTangDestVelocity（获取切向追踪目标速度）

5.5.7.7.1 功能

该指令获取切向追踪目标切向角的变化速度，单位为 C 轴（ucAxisC，刀具旋转轴）用户单位/S。该指令执行时，C 轴当前指令需是 HMC_MoveTang 指令。

5.5.7.7.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_GetMoveTangDestVelocity	Output	LREAL	切向追踪指令的目标切向角的变化速度。函数错误时返回 0，并报用户错误

5.5.7.7.3 指令类型

非轴运动缓存指令。

5.5.7.8 HMC_GetMoveTangDeviation（获取切向追踪方向偏差）

5.5.7.8.1 功能

该指令获取切向追踪 C 轴（ucAxisC，刀具旋转轴）当前指令位置与目标切向角的偏差，单位为 C 轴用户单位。该指令执行时，C 轴当前指令需是 HMC_MoveTang 指令。

5.5.7.8.2 参数

参数	声明	数据类	说明

		型	
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_GetMoveTangDestViation	Output	LREAL	切向追踪指令当前指令位置与目标切向角的偏差。函数错误时返回0，并报用户错误。

5.5.7.8.3 指令类型

非轴运动缓存指令。

5.5.7.9 HMC_GetMoveTangMasterAxisX（获取切向追踪主轴 X）

5.5.7.9.1 功能

获取当前切向追踪指令的 X 轴。

5.5.7.9.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_GetMoveTangMasterAxisX	Output	INT	X 轴轴号：成功 -1：错误

5.5.7.9.3 指令类型

非轴运动缓存指令。

5.5.7.10 HMC_GetMoveTangMasterAxisY（获取切向追踪主轴 Y）

5.5.7.10.1 功能

获取当前切向追踪指令的 Y 轴。

5.5.7.10.2 参数

参数	声明	数据类型	说明
ucAxisC	Input	USINT	刀具旋转轴轴号
HMC_GetMoveTangMasterAxisY	Output	INT	Y 轴轴号：成功 -1：错误

5.5.7.10.3 指令类型

非轴运动缓存指令。

5.5.8 多轴同步控制

5.5.8.1 HMC_GantrySynchro（龙门同步控制）

5.5.8.1.1 功能

一个龙门组最多支持 8 个龙门轴，其中包括一个龙门主轴，其余轴作为从轴，即最多支持 7 个从轴。主轴运动缓存中不显示龙门同步指令。对主轴投送运动控制指令，对从轴投送龙门同步控制指令并指定主轴，龙门同步控制指令会对主轴和从轴进行动态同步。

龙门同步控制指令通过开环和闭环两种方式结合实现对一组龙门轴的动态同步控制：

- 开环同步：将龙门主轴解算的目标位置同步至各个龙门从轴，作为从轴的目标位置。从前端指令保证轴间位置同步；
- 闭环同步：动态监测各龙门轴的实际反馈位置值，通过轴间交叉耦合算法对各个龙门进行动态补偿，实现动态纠偏以消除各龙门轴轴间的偏差。闭环同步要求各龙门轴必须带有位置反馈值。
- 在动态同步过程中，还支持轴间实时偏差监测，设置两组阈值：
- 轴间偏差报警阈值：实时监测龙门组内两两轴间偏差，当发生偏差超出报警阈值时，在对应轴参 AxisStatus 的 bit5 置 1，当偏差恢复时，轴参 AxisStatus 的 bit5 自动恢复。
- 轴间偏差停车保护阈值：实时监测龙门组内两两轴间偏差，当发生偏差超出报警阈值时，在对应

轴参 AxisStatus 的 bit6 置 1，并关闭伺服使能开关，无论 AxisErrMask 的 bit6 位是否为 1。当偏差恢复时，轴参 AxisStatus 的 bit6 不自动恢复，需手动清除。

- 可使用 HMC_Cancel 指令撤消龙门同步指令，轴号为从轴数组中任一轴号；
- 主轴轴号必须小于从轴数组中任一轴号；
- 从轴数组轴号必须从小到大排列。

5.5.8.1.2 参数

参数	声明	数据类型	说明
UCMASTERAXIS	INPUT	USINT	龙门组主动同步轴号（需小于任一从动轴号）
PSLAVEAXIS	INPUT	POINTER TO USINT	龙门组从动同步轴号数组
UCSLAVEAXISNUM	INPUT	USINT	龙门组从动同步轴个数（≤7）
UCMODE	INPUT	USINT	龙门同步模式： 0：轴间开环同步 1：轴间闭环同步
LRPOSWARING	INPUT	LREAL	轴间偏差报警阈值
LRPOSERR	INPUT	LREAL	轴间偏差停车保护阈值
HMC_GantrySynchro	OUTPUT	UDINT	轴指令 ID：成功 0xFFFFFFFF：函数参数错误

5.5.8.1.3 指令类型

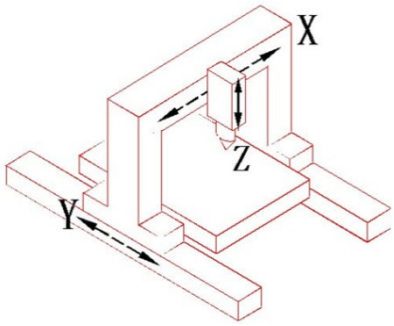
轴运动缓存指令。

5.5.8.1.4 补充说明

- 同一龙门组内，所有龙门轴类型必须一致；
- 当选择龙门同步模式为开环模式时，轴类型为带实际输出的轴类型即可，如直连步进轴、伺服轴、总线轴等；当选择为闭环模式时，轴类型为带实际输出且有实际反馈的位置闭环控制轴类型，即直连伺服轴、总线速度控制模式轴类型；
- 龙门轴的循环模式必须为有限行程，即 RepMode=0。

5.5.8.1.5 示例

如图结构，其中 Y 轴为龙门双驱，可以在工作平台执行一个圆弧轨迹。



结构示意图

```
ucMaster := 1;              (*龙门主轴号*)

pSlaveAxisAry[0] := 2;     (*龙门从轴号*)

ucSlaveNum := 1;           (*一个龙门从轴*)

ucMode := 1;               (*轴间闭环同步*)

lrPosWaring := 100;

lrPosErr := 500;

HMC_GantrySynchro (ucMaster,ADR(pSlaveAxisAry),ucSlaveNum,ucMode,lrPosWaring,lrPosErr); (*投送龙门指令，使得轴 0 和轴 1 建立龙门关系*)

HMC_MoveCircle (10,0,1,1,0,0,100,100);   (*轴 0 为 X 轴，轴 1 为 Y 轴，轴 2 与轴 1 为龙门关系*)
```

指令投送后如图所示：

Basic				
AxisID	USINT	0	1	2
AxisType	EmAxisType	Servo_AO	Servo_AO	Servo_AO
Units	LREAL	1	1	1
Status				
MCmdType	EmCmdType	HMC__MoveCircle	HMC__MoveCircle	HMC__GantrySynchro
NCmdType	EmCmdType	HMC__McIdle	HMC__McIdle	HMC__McIdle
AxisStatus	UDINT	0	0	0
AxisErrMask	UDINT	1	1	1

参数示意图

5.5.8.2 HMC_GantryInit（龙门同步回参考点位置对齐初始化）

5.5.8.2.1 功能

龙门在开始运行时，须完成对位以及回原点的动作。对位需由安装在龙门各轴侧的对位检测器配合完成，对位检测器的机械位置安装必须准确，因为这是龙门可以校正平行对准的唯一地方。整个龙门初始化过程如下：

- 各轴依次单独回到参考点，在一轴回参考点过程中，另一龙门轴同步跟随；
- 所有轴完成回参考点后，同时对所有龙门轴进行位置重定义；
- 计算轴间偏差，当偏差超出阈值时，指令结束；偏差在阈值范围内时，对各轴进行补偿移位，消除偏差实现对位。

龙门轴回参考点与回原点功能类似，模式支持 Home 低速和 HomeEx 高速两种方式；选择 HomeEx 高速时必须配合使用高速 DI。

5.5.8.2.2 参数

输入参数	声明	类型	参数描述
PAXIS	INPUT	POINTER TO USINT	龙门组同步轴号数组
UCAXISNUM	INPUT	USINT	龙门组同步轴个数（≤8）
UCHOMEMODE	INPUT	USINT	龙门同步回参考点的原点回归模式，可参见 MC_Home 指令中的原点回

			归模式说明（龙门同步初始化支持模式 1/2/3/5/7/11/17/18/19/21/23/27）
PDICHL	INPUT	POINTER TO UDINT	回参考点使用的 DI 信号（不可重复）
DREFERPOINTPOS	INPUT	LREAL	参考点绝对位置值
DPOSERR	INPUT	LREAL	轴间偏差阈值
HMC_GantryInit	OUTPUT	UDINT	轴指令 ID：成功 0xFFFFFFFF：函数参数错误

5.5.8.2.3 指令类型

轴运动缓存指令。

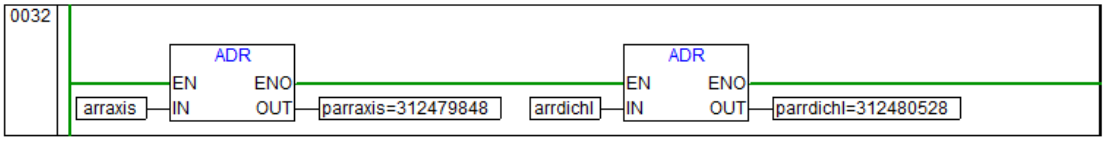
5.5.8.2.4 补充说明

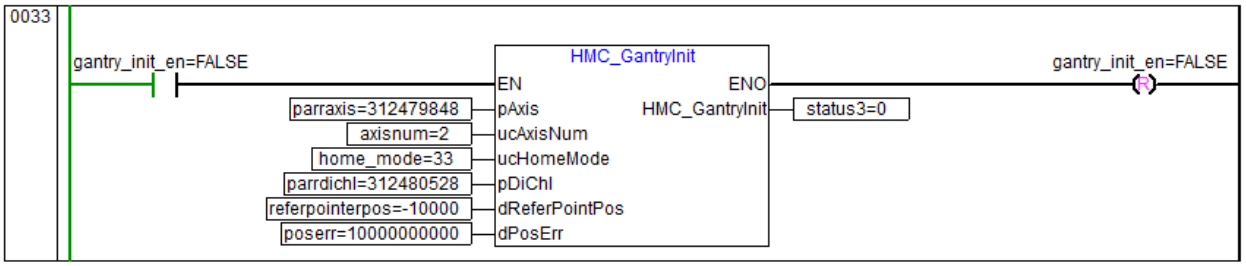
- 龙门轴回参考点使用的原点信号源不可重复；
- 同一龙门组内，所有龙门轴类型必须一致；
- 当选择龙门同步模式为开环模式时，轴类型为带实际输出的轴类型即可，如直连步进轴、伺服轴、总线轴等；当选择为闭环模式时，轴类型为带实际输出且有实际反馈的位置闭环控制轴类型，即直连伺服轴、总线速度控制模式轴类型；
- 龙门轴的循环模式必须为有限行程，即 RepMode=0。
- 可使用 HMC_Cancel 指令撤消龙门初始化指令，轴号必须为轴数组中最小轴号；
- 轴数组轴号顺序无要求。

5.5.8.2.5 示例

龙门双驱结构停机重启后，两龙门轴处于自然连接状态，存在偏差。此时可先通过龙门初始化对位指令，使得两轴依次回到对位参考点，对轴位置进行初始化，以此获取龙门轴间偏差后，对两轴消除偏差，完成对龙门轴的位置初始化以及对齐。在此之后再对两轴投送龙门同步控制指令，建立龙门关联关系；之后对龙门主轴投送运动控制指令，实现龙门轴动态同步跟随控制。

变量名	直接地址	变量说明	变量类型	初始值
gantry_init_en			BOOL	FALSE
arraxis			ARRAY[0..1] OF USINT	
arrdichl			ARRAY[0..1] OF UDINT	
referpointerpos			LREAL	-10000
poserr			LREAL	10000000000
home_mode			USINT	33
status3			UDINT	0
parraxis			POINTER TO USINT	0
parrdichl			POINTER TO UDINT	0






```
(*龙门同步初始化*)
IF gantry_init_en = TRUE THEN
    status3 := HMC_GantryInit(ADR(arraxis), 2, home_mode, ADR(arrdichl), referpointerpos, poserr);
    gantry_init_en := FALSE;
END_IF
```

龙门组同步轴号数组：

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	arraxis[0]		龙门同步轴0	USINT	0	FALSE
0002	arraxis[1]		龙门同步轴1	USINT	1	FALSE

回参考点使用的 DI 信号：

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	arrdichl[0]		轴0 DI信号	UDINT	38	FALSE
0002	arrdichl[1]		轴1 DI信号	UDINT	39	FALSE

5.5.8.3 HMC_SetGantryAxisPID（设置龙门同步轴补偿 PID 参数）

5.5.8.3.1 功能

配合龙门同步控制指令 HMC_GantrySynchro 使用。设置龙门组内指定龙门轴的动态纠偏 PID 参数。

若指定轴号当前不为龙门同步指令，则设置不成功。

5.5.8.3.2 参数

输入参数	说明	类型	参数描述
UCAXISID	INPUT	USINT	龙门组轴号
DKPGAIN	INPUT	LREAL	动态补偿 PID 比例系数（默认值为 1）
DKIGAIN	INPUT	LREAL	动态补偿 PID 积分系数（默认值为 0）
DKDGAIN	INPUT	LREAL	动态补偿 PID 微分系数（默认值为 0）
HMC_SetGantryAxisPID	OUTPUT	UDINT	0：成功 0xFFFFFFFF：函数参数错误

5.5.8.3.3 指令类型

非轴运动缓存指令。

5.5.8.3.4 补充说明

不调用此指令设置 PID 时，各轴动态补偿 PID 为默认值。

5.5.8.4 HMC_GetGantryAxisPID（读取龙门同步轴补偿 PID 参数）

5.5.8.4.1 功能

配合龙门同步控制指令 HMC_GantrySynchro 使用。读取龙门组内指定龙门轴的动态纠偏 PID 参数。

5.5.8.4.2 参数

输入参数	说明	类型	参数描述
UCAXISID	INPUT	USINT	龙门组轴号
DKPGAIN	INPUT/OUTPUT	LREAL	动态补偿 PID 比例系数
DKIGAIN	INPUT/OUTPUT	LREAL	动态补偿 PID 积分系数
DKDGAIN	INPUT/OUTPUT	LREAL	动态补偿 PID 微分系数
HMC_GetGantryAxisPID	OUTPUT	UDINT	0：成功 0xFFFFFFFF：函数参数错误

5.5.8.4.3 指令类型

非轴运动缓存指令。

5.5.8.4.4 补充说明

轴号不为龙门轴时，返回错误。

5.5.8.5 HMC_GantryGroupDefPos（设置龙门同步轴组当前位置）

5.5.8.5.1 功能

配合龙门同步控制指令 HMC_GantrySynchro 使用。对龙门组内所有龙门轴的位置重定义。

5.5.8.5.2 参数

输入参数	说明	类型	参数描述
PAXIS	INPUT	POINTER TO USINT	龙门组同步轴
UCAXISNUM	INPUT	USINT	龙门组同步轴个数（≤8）
DMPOS	INPUT	LREAL	预定义位置
HMC_GantryGroupDefPos	OUTPUT	UDINT	0：成功 0xFFFFFFFF：函数参数错误

5.5.8.5.3 指令类型

非轴运动缓存指令。

5.5.8.5.4 补充说明

当前指令不为龙门同步指令时，返回错误。

5.5.8.6 HMC_GetGantryInitState（读取龙门同步控制函数当前状态）

5.5.8.6.1 功能

配合龙门同步会对位参考点初始化对齐指令 HMC_GantryInit 使用。读取该指令当前执行状态。

5.5.8.6.2 参数

输入参数	说明	类型	参数描述
PAXIS	INPUT	POINTER TO USINT	龙门组同步轴
UCAXISNUM	INPUT	USINT	龙门组同步轴个数
HMC_GetGantryInitState	OUTPUT	UDINT	表示龙门初始化指令执行状态： 1：Homing 2：ReturnBack 3：Defpos 4：Compsate 5：Finish 0xFFFFFFFF：函数参数错误

5.5.8.6.3 指令类型

非轴运动缓存指令。

5.5.9 轴位置补偿功能

5.5.9.1 HMC_SetAxisCompenPos（轴位置补偿量设定）

5.5.9.1.1 功能说明

设定轴位置补偿量。

- 注意：补偿量的设定需放在 HMC_TrigAxisCompens 指令之前，才能在触发下次补偿动作时生效。该值的改变并不影响正在进行中的补偿动作。

5.5.9.1.2 参数

参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	轴号
dCompenPos	INPUT	LREAL	位置补偿值
HMC_SetAxisCompenPos	OUTPUT	UDINT	0：执行成功 0xFFFFFFFF 执行失败

5.5.9.1.3 指令类型

非轴运动缓存指令。

5.5.9.2 HMC_SetAxisCompenPara（轴位置补偿运动参数设定）

5.5.9.2.1 功能

设定轴位置补偿运动的运动学参数，实时生效。

5.5.9.2.2 参数

参数	声明	数据类型	说明

ucAxisID	INPUT	USINT	轴号
dJerkMode	INPUT	USINT	轴位置补偿动作采用的加减速类型： 0：梯形加减速 1：S 形加减速
dVmax	INPUT	LREAL	位置补偿动作最大速度
dAccel	INPUT	LREAL	位置补偿动作加速度
dDecel	INPUT	LREAL	位置补偿动作减速度
dJerk	INPUT	LREAL	S 形加减速加加速度
HMC_SetAxisCompenPara	OUTPUT	UDINT	0：执行成功 HMC_SetAxisCompenPara：执行失败

5.5.9.2.3 指令类型

非轴运动缓存指令。

5.5.9.3 HMC_TrigAxisCompens（触发位置补偿）

5.5.9.3.1 功能

触发轴位置补偿动作。每运行一次该指令，都会使用当前设定的轴位置补偿量进行一次位置补偿动作。如果调用该指令时当前补偿动作尚未结束，则放弃剩余尚未完成的补偿，立即开始新的补偿动作。可调用 HMC_GetAxisCompenState 指令查询轴的补偿状态，以确认当前补偿动作是否已经完成。

5.5.9.3.2 参数

输入参数	说明	类型	参数描述
ucAxisID	INPUT	USINT	轴号
HMC_TrigAxisCompens	OUTPUT	UDINT	0：执行成功 0xFFFFFFFF：执行失败

5.5.9.3.3 指令类型

非轴运动缓存指令。

5.5.9.4 HMC_StopAxisCompens（停止位置补偿）

5.5.9.4.1 功能

立即停止正在进行中的轴位置补偿动作。

5.5.9.4.2 参数

输入参数	说明	类型	参数描述
ucAxisID	INPUT	USINT	轴号
HMC_StopAxisCompens	OUTPUT	UDINT	0：执行成功 0xFFFFFFFF：执行失败

5.5.9.4.3 指令类型

非轴运动缓存指令。

5.5.9.5 HMC_GetAxisCompenState（获取轴位置补偿状态）

5.5.9.5.1 功能说明

获取轴补偿状态。

5.5.9.5.2 参数

输入参数	说明	类型	参数描述
ucAxisID	INPUT	USINT	轴号
HMC_GetAxisCompenState	OUTPUT	UDINT	0：补偿动作执行完毕 1：补偿动作进行中 0xFFFFFFFF：轴参数错误

5.5.9.5.3 指令类型

非轴运动缓存指令。

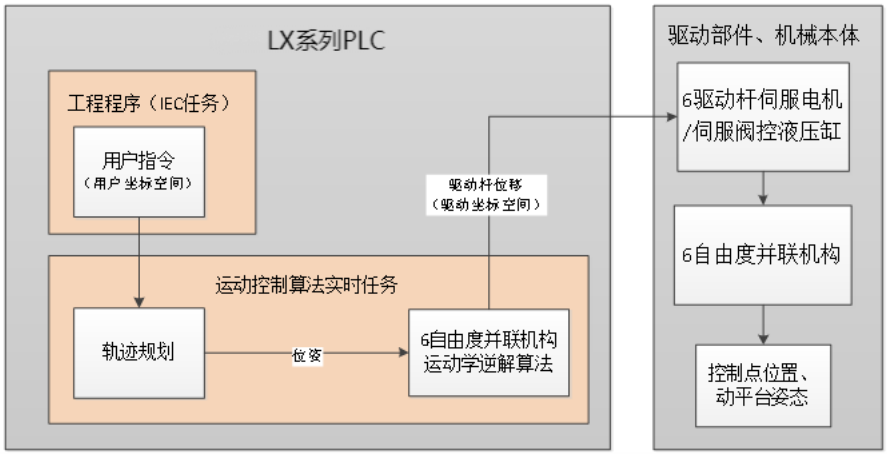
5.6 机器人控制指令

5.6.1 Stewart 6 自由度并联机构

5.6.1.1 HMC_StewartControl（Stewart 机构控制）

5.6.1.1.1 功能

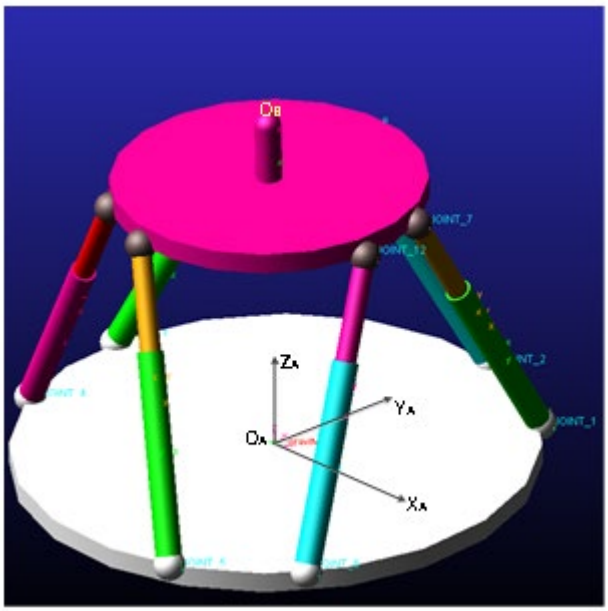
该指令完成 Gough-Stewart 6 自由度并联机构的运动学变换（由用户坐标空间到驱动坐标空间）。指令生效后，便可在用户坐标空间（直角坐标系）编写用户工程程序，控制器自动完成机构的运动学逆解变换，在驱动坐标空间内控制 6 个驱动杆运动。过程如图所示（有关 Gough-Stewart 平台的详细信息参见补充说明部分）。



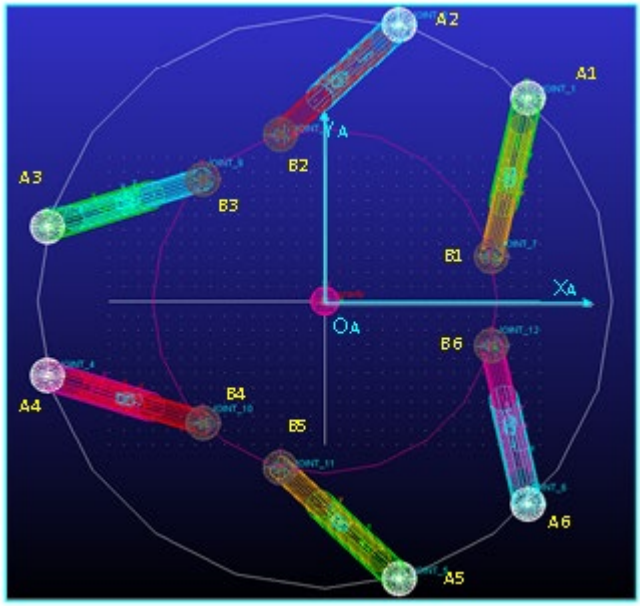
六自由度并联机构控制系统框图

姿态描述采用 RPY 角，即绕固定坐标系的旋转定义方式，旋转顺序为 XYZ。

在 Gough-Stewart 平台 6 个驱动杆回零操作完成，并且 6 个驱动杆实际杆长均等于 StewartPara 数组中的 l_{init} 参数时，调用该指令完成 6 自由度并联平台的初始化，配置 6 自由度并联机构的机构参数、驱动坐标空间的轴号（6 个）、用户坐标空间的轴号（6 个）。



(a)



(b)

并联六自由度平台坐标与机构参数示意图

如上图所示，目标控制点 OB 位于通过动平台外接圆圆心并与该圆所在平面垂直的直线上，距离动平台平面的距离为 H。

定义固定于静平台的绝对坐标系 OA-XA YA ZA 如下：

（1）坐标原点位于静平台铰点外接圆圆心 OA；

（2）ZA 轴垂直与静平台平面向上；

（3）XA 轴位于底面，且垂直下铰点 A1 和 A6 的连线；

（4）YA 轴方向可由右手法则确定。

设 x、y、z 为目标控制点 OB 在 OA-XA YA ZA 中的位置坐标，φx 、φy 、φz 为动平台在 OB-XBYBZB 中的姿态坐标。则动平台在绝对坐标系 OA-XA YA ZA 内的位姿坐标为：

$$\boldsymbol{X} = [\mathrm{x}, \mathrm{y}, \mathrm{z}, \varphi_x, \varphi_y, \varphi_z]^T$$

调用该指令初始化完成后，用户坐标空间内的各轴坐标被自动初始化为初始位姿：

$$\boldsymbol{X}_{init} = [0,0,z_{\mathrm{init}},0,0,0]^T$$

驱动坐标空间的各轴坐标则被初始化为初始杆长参数 linit。

使用该指令时需注意以下几点：

（1）在 6 个驱动杆实际杆长均等于 StewartPara 数组中的 linit 参数时方可调用该函数，否则初始位姿会产生偏差；

（2）可使用 HMC_Cancel(AxisID,CancelMode)撤销该指令，撤销时传入的轴号必须为 6 个驱动轴中的最小轴号；

（3）指令调用成功后，首次被执行时会自动把用户坐标空间的各轴的位移设定为动平台在绝对坐标系 OA-XA YA ZA 中的位姿，把驱动坐标空间各轴的位移设定为初始杆长 linit，之后在此初始位姿基础上运动；

（4）指令首次被执行时会把驱动坐标空间及用户坐标空间各轴的 RepMode 设定为有限行程，并根据驱动杆长参数 lmin 与 lmax 自动设定各驱动轴的软限位参数。在指令运行过程中不允许修改 RepMode 轴参，否则可能会发生坐标转换错误；

（5）调用该指令初始化成功后，用户在绝对坐标系 OA-XA YA ZA 内编程，不允许调用 HMC_DefPos 或 HMC_DefOrigin 指令重新设定用户坐标空间或驱动坐标空间内的各轴坐标，否则坐标转换时会产生严重的偏差。如果用户需要使用相对用户坐标系，可在工程程序中自行进行坐标偏移换算；

（6）该指令一旦调用成功，在用户坐标空间内不允许使用 HOME 指令，因 HOME 指令在回零完成之时会有 DPOS 和 Mpos 的跃变，从而造成坐标转换后驱动轴输出的跃变。

（7）该并联机构在可达工作空间内存在奇异位形，在奇异位形附近机构力学性能较差，甚至失去承载

能力。请根据机构参数，合理规划工作空间，避开奇异位形及机械干涉位形。

5.6.1.1.2 参数

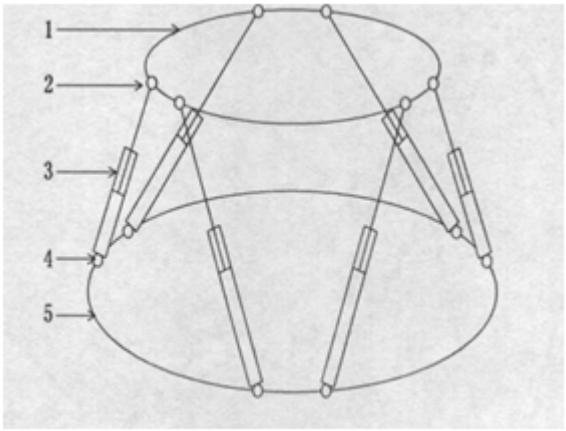
参数	声明	数据类型	说明
UserSpaceAxisID	INPUT	POINTER TO USINT	用户坐标空间轴号数组，共 6 个元素，依次对应平移轴 XYZ 与旋转轴 φx φy φz 。单位为度 一般为虚拟轴，用户对该 6 个轴进行编程，对动平台位姿进行规划
DriveSpaceAxisID	INPUT	POINTER TO USINT	驱动轴轴号数组，共 6 个元素，依次对应驱动杆 1~6。如图所示，AiBi 对应编号为 i 的驱动杆 物理输出轴，驱动机构 6 个驱动杆运动
StewartPara	INPUT	POINTER TO LREAL	Gough-Stewart 机构参数数组，共 8 个元素，依次为： 1、静平台铰点外接圆半径 R0 2、静平台铰点 短边对应圆心角度 θ0（0°≤θ0<60°） 3、动平台铰点外接圆半径 R1 4、动平台铰点 短边对应圆心角度 θ1（0°≤θ1<60°） 5、目标控制点距离动平台各铰点所在平面的距离 H（目标控制点位于通过动平台外接圆圆心并与该圆所在平面垂直的直线上)。 H≥0 6、驱动杆初始杆长 l_{init} 7、驱动杆最短杆长 l_{min} 8、驱动杆最长杆长 l_{max}
HMC_StewartControl	OUTPUT	UDINT	指令 ID：成功 16#FFFFFFFF：错误

- 输入参数长度单位为用户单位，角度单位为角度制（°）。

5.6.1.1.3 指令类型

轴运动缓存指令。

5.6.1.1.4 补充说明



并联六自由度平台的机构原理

1：动平台

2：上球铰

3：液压缸/电动缸

4：下球铰

5：静平台

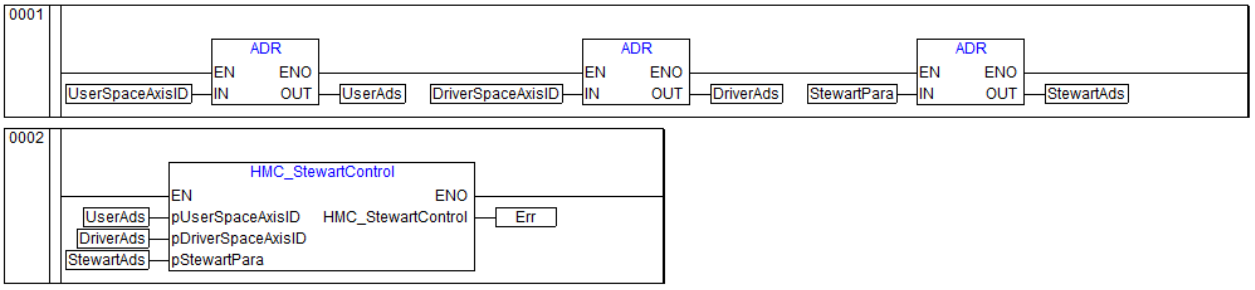
六自由度并联平台的机构原理如上图所示，系统由动、静平台和 6 个伺服电动缸(或伺服阀控液压缸)组成，各支链电动缸与动、静平台分别由球铰连接（实际使用中为消除局部自由度，平台的上铰点通常使用虎克铰连接）。一般静平台固定于基座，通过控制支链电动缸的伸缩运动，动平台可以实现空间运动，完成空间六自由度的位置及姿态调整。并联机构本身的刚度大，当使用液压缸驱动时，特别适用于重载情况下对各种空间运动的模拟。

5.6.1.1.5 使用示例

任务一：投送 StewartControl 指令

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	UserSpaceAxisID		用户轴号数组	ARRAY[0..5] OF BOOL		FALSE
0002	DriverSpaceAxisID		驱动轴号数组	ARRAY[0..5] OF BOOL		FALSE
0003	StewartPara		机构参数	ARRAY[0..7] OF LREAL		FALSE
0004	Err		返回值	DWORD	0	FALSE

LD 程序：



ST 程序：

```
1   (*投送StewartControl指令*)
2   Err  := HMC_StewartControl(ADR(UserSpaceAxisID), ADR(DriveSpaceAxisID), ADR(StewartPara));
```

■ 变量定义

（1） 用户轴号数组：（不允许存在轴号重复或与驱动轴号重复的情况）

Stewart_ST.UserSpaceAxisID

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	UserSpaceAxisID[0]		X轴_位置	USINT	10	FALSE
0002	UserSpaceAxisID[1]		Y轴_位置	USINT	11	FALSE
0003	UserSpaceAxisID[2]		Z轴_位置	USINT	12	FALSE
0004	UserSpaceAxisID[3]		X轴_姿态	USINT	13	FALSE
0005	UserSpaceAxisID[4]		Y轴_姿态	USINT	14	FALSE
0006	UserSpaceAxisID[5]		Z轴_姿态	USINT	15	FALSE

变量定义

（2） 驱动轴号数组：（不允许存在编码器轴类型或未使用轴类型）

Stewart_ST.DriveSpaceAxisID

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	DriveSpaceAxisID[0]		驱动杆1	USINT	0	FALSE
0002	DriveSpaceAxisID[1]		驱动杆2	USINT	1	FALSE
0003	DriveSpaceAxisID[2]		驱动杆3	USINT	2	FALSE
0004	DriveSpaceAxisID[3]		驱动杆4	USINT	3	FALSE
0005	DriveSpaceAxisID[4]		驱动杆5	USINT	4	FALSE
0006	DriveSpaceAxisID[5]		驱动杆6	USINT	5	FALSE

变量定义

（3） 机构参数：（当机构参数设置不合理时，指令投送不成功）

Stewart_ST.StewartPara

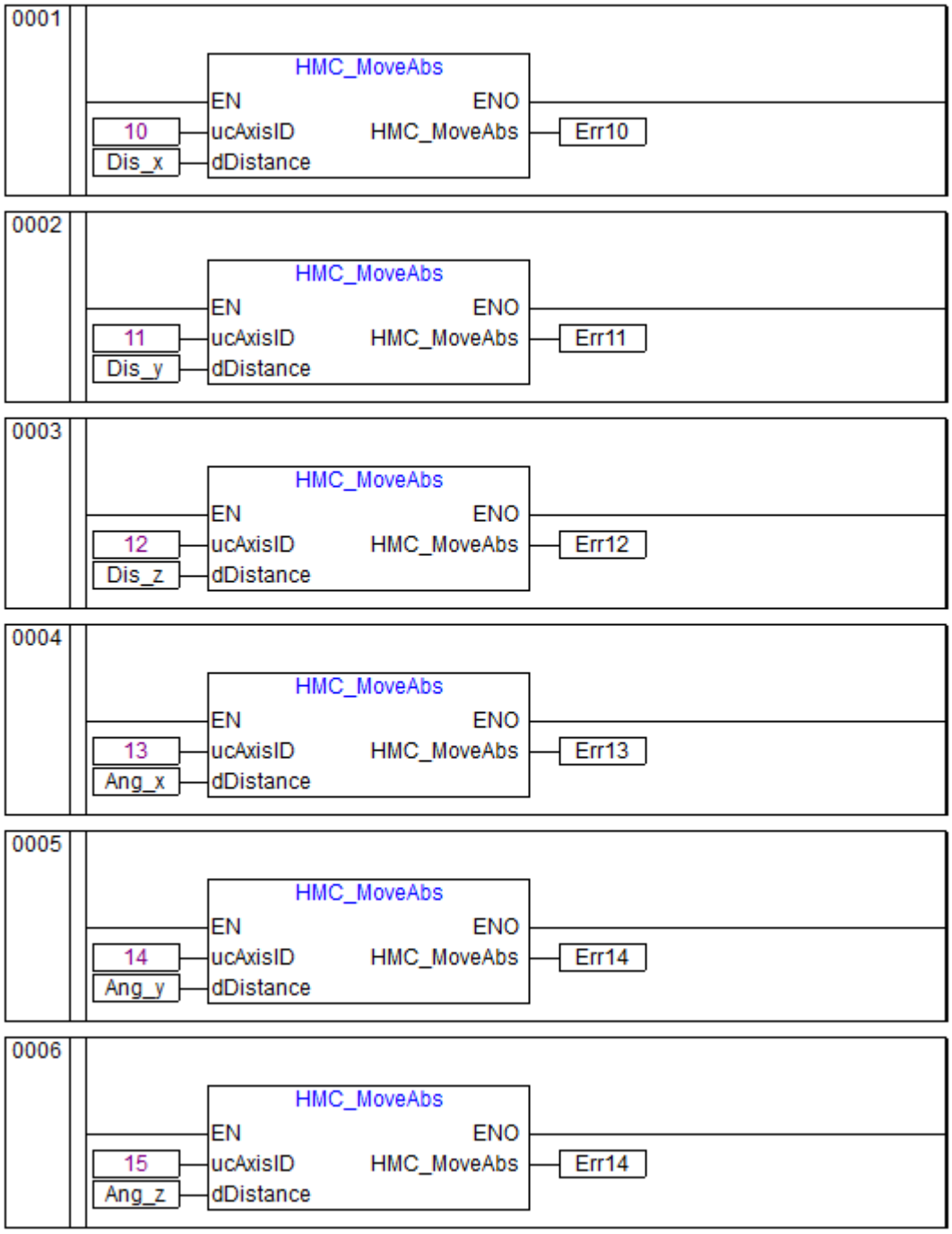
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	StewartPara[0]		动平台半径	LREAL	10000	FALSE
0002	StewartPara[1]		动平台短边角度	LREAL	30	FALSE
0003	StewartPara[2]		静平台半径	LREAL	20000	FALSE
0004	StewartPara[3]		静平台短边角度	LREAL	30	FALSE
0005	StewartPara[4]		控制点距离动平台高度	LREAL	1000	FALSE
0006	StewartPara[5]		初始杆长	LREAL	20000	FALSE
0007	StewartPara[6]		最小杆长	LREAL	10000	FALSE
0008	StewartPara[7]		最大杆长	LREAL	30000	FALSE

变量定义

任务二：用户轴中投送运动控制指令（单轴绝运动、6 轴绝对值直线插补(N 轴插补)、圆弧、球弧、样条等指令），驱动轴将会调整位置和姿态。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Dis_x		X轴坐标位置	LREAL	1000	FALSE
0002	Dis_y		Y轴坐标位置	LREAL	2000	FALSE
0003	Dis_z		Z轴坐标位置	LREAL	3000	FALSE
0004	Ang_x		X轴姿态_角度	LREAL	10	FALSE
0005	Ang_y		Y轴姿态_角度	LREAL	-20	FALSE
0006	Ang_z		Z轴姿态_角度	LREAL	30	FALSE

- LD 程序：


0001
HMC_MoveAbs
EN
ENO
10
ucAxisID
HMC_MoveAbs
Err10
Dis_x
dDistance
0002
HMC_MoveAbs
EN
ENO
11
ucAxisID
HMC_MoveAbs
Err11
Dis_y
dDistance
0003
HMC_MoveAbs
EN
ENO
12
ucAxisID
HMC_MoveAbs
Err12
Dis_z
dDistance
0004
HMC_MoveAbs
EN
ENO
13
ucAxisID
HMC_MoveAbs
Err13
Ang_x
dDistance
0005
HMC_MoveAbs
EN
ENO
14
ucAxisID
HMC_MoveAbs
Err14
Ang_y
dDistance
0006
HMC_MoveAbs
EN
ENO
15
ucAxisID
HMC_MoveAbs
Err14
Ang_z
dDistance

程序示例

■ ST 程序

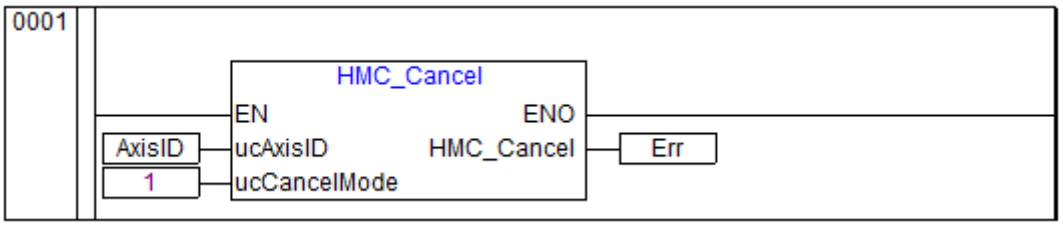
```
1   (*X轴位置调整*)
2   HMC_MoveAbs(10, Dis_x);
3   (*X轴位置调整*)
4   HMC_MoveAbs(11, Dis_y);
5   (*X轴位置调整*)
6   HMC_MoveAbs(12, Dis_z);
7   (*X轴位置调整*)
8   HMC_MoveAbs(13, Ang_x);
9   (*X轴位置调整*)
10  HMC_MoveAbs(14, Ang_y);
11  (*X轴位置调整*)
12  HMC_MoveAbs(15, Ang_z);
13  (*也可以投送球弧或样条指令，来规划用户轴的位置和姿态*)
14
```

程序示例

任务三：撤销 StewartControl 指令（AxisID 必需是驱动轴号数组中的最小轴号）

序号	变量名	直接地址	变量说明	变量类型	初始值 △	掉电保护
0001	AxisID		驱动轴数组第一个轴号	BYTE	0	FALSE

- LD 程序：



程序示例

- ST 程序：

```
1   (*撤销StewartControl指令*)
2   HMC_Cancel(AxisID, 1);
```

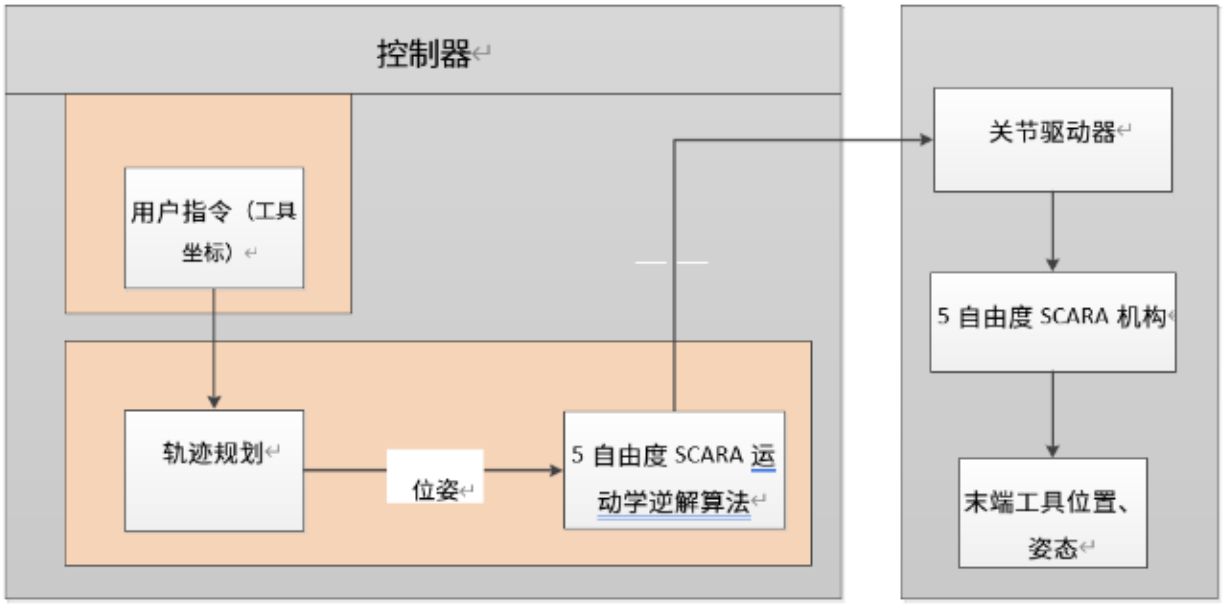
程序示例

5.6.2 5 自由度 SCARA 机械臂

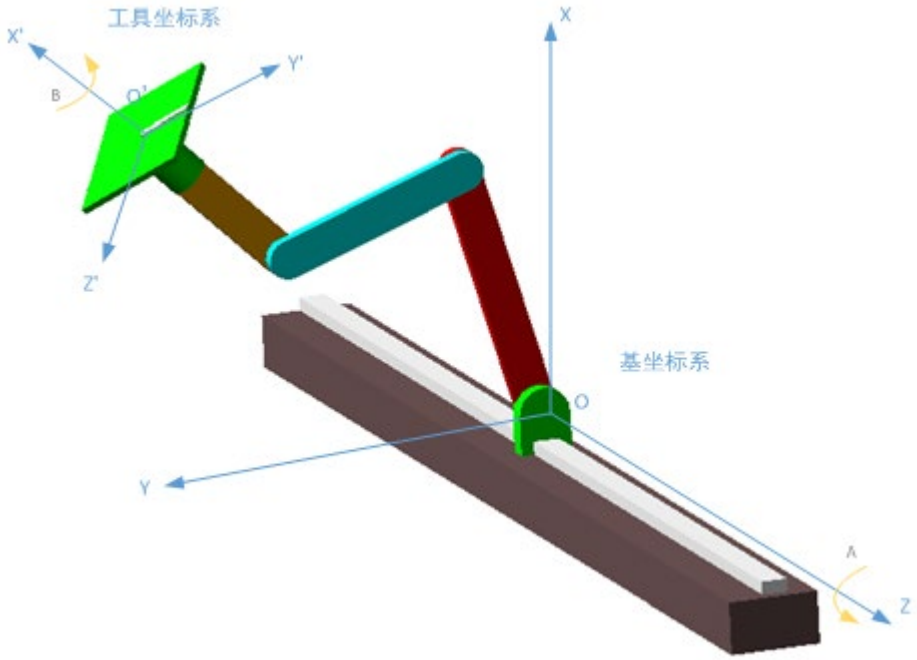
5.6.2.1 HMC_Scara5Control（5 自由度 SCARA 机械臂控制）

5.6.2.1.1 功能

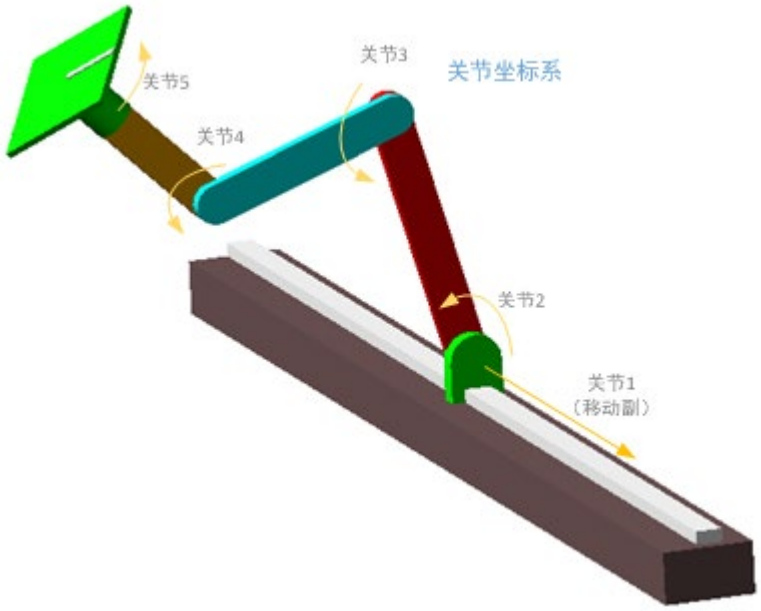
该指令完成 5 自由度 SCARA 机构的运动学变换（由基坐标系到关节坐标系）。指令生效后，便可在基坐标系（直角坐标系）中编写用户工程程序，控制器自动完成机构的运动学逆解变换，驱动各关节运动。过程如图所示。



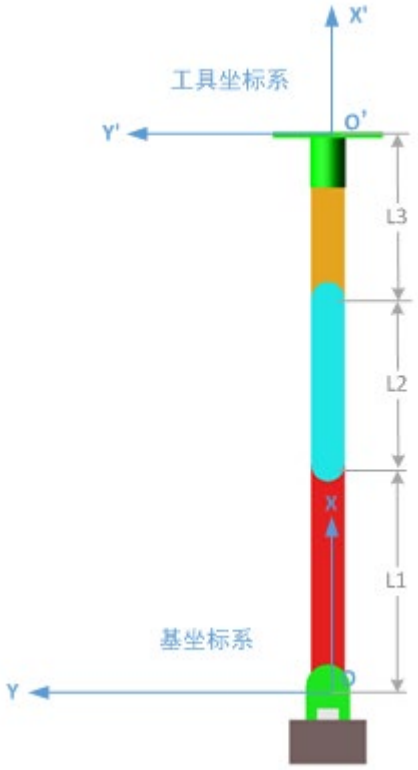
5 自由度 SCARA 串联机构控制系统框图



5 自由度 SCARA 机械手基坐标系、工具坐标系定义（笛卡尔坐标）



5 自由度 SCARA 机械手关节坐标系定义（广义坐标）



5 自由度 SCARA 机械手关节角零点位姿

定义基坐标系、工具坐标系、关节坐标系如图 5 自由度 SCARA 机械手基坐标系、工具坐标系定义（笛卡尔坐标）_、5 自由度 SCARA 机械手关节坐标系定义（广义坐标）所示，关节角零点位置如图 5 自由度 SCARA 机械手关节角零点位姿所示。

定义工具坐标：工具坐标系 在 基坐标系 中的位置和姿态描述机构目标位姿，即机构的工具坐标。 共 5 个坐标分量，即（x,y,z,φ_A,φ_B）。

姿态描述采用欧拉角，即绕动坐标系的旋转定义方式，旋转顺序为 Z-X'，分别对应图 5 自由度 SCARA 机械手基坐标系、工具坐标系定义（笛卡尔坐标）中 A 轴、B 轴。

关节角的零点位置如图 5 自由度 SCARA 机械手关节角零点位姿中所示。在回零操作完成后，只要零点位置未发生改变，则可在任意位置调用该指令。

使用该指令时需注意以下几点：

（1） 长度单位为用户单位，角度单位为角度制（°）；

（2） 务必保证关节角的零点位置如图 5 自由度 SCARA 机械手关节角零点位姿中所示；

(3) 可使用 HMC_Cancel(AxisID,CancelMode)撤销该指令，撤销时传入的轴号必须为 5 个驱动轴中的最小轴号；

(4) 指令调用成功后，首次被执行时，会根据 当前关节角 初始化 工具坐标，之后在此初始位姿基础上运动；

(5) 指令首次被执行时会把驱动坐标空间及用户坐标空间各轴的 RepMode 设定为有限行程，用户可根据机构具体结构设定机构各关节角的软限位参数。在指令运行过程中不允许修改 RepMode，否则可能会发生坐标转换错误。

(6) 可根据机构实际情况设定机构各关节角的限位值；

(7) 调用该指令初始化成功后，用户在基坐标系内编程，不允许调用 HMC_DefPos 或 HMC_DefOrigin 指令重新设定基坐标系或关节坐标系内的各轴坐标值，否则坐标转换时会产生严重的偏差。如果用户需要使用相对坐标系，可在工程程序中自行进行坐标偏移换算；

(8) 该指令运行时不允许对基坐标系中的轴使用 HOME 指令，因 HOME 指令在回零完成之时会有 DPOS 和 Mpos 的跃变，从而造成坐标转换后关节轴输出的跃变；

(9) 机构存的可达工作空间是有限的，如果规划的路径超出了其可达工作空间，出于安全考虑，指令会被撤销。请根据机构参数，合理规划工作空间，确保工作空间位于机构可达工作空间之内（可用 HMC_ Scara5ReverseKinematics 指令辅助校验规划的工具坐标是否合法）。

5.6.2.1.2 参数

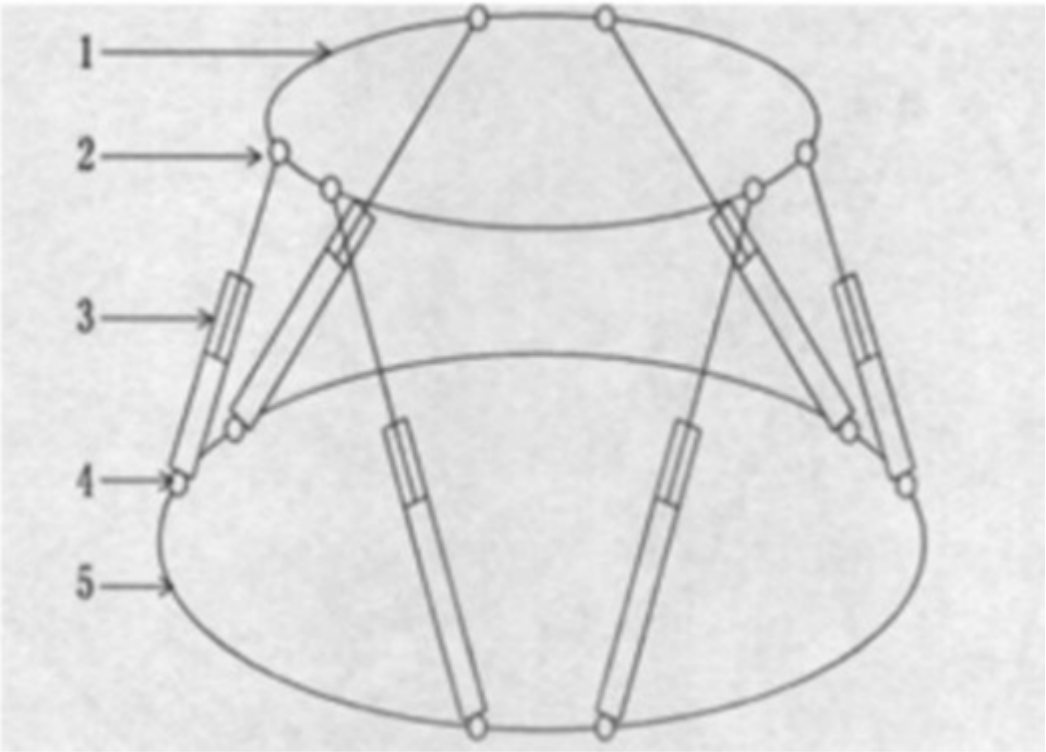
参数	声明	数据类型	说明
arrBaseCoordAxisID	INPUT	POINTER TO USINT	基坐标系轴号数组，共 5 个元素，依次对应平移轴 X、Y、Z 与旋转轴 A、B，如图 5 自由度 SCARA 机械手基坐标系、工具坐标系定义（笛卡尔坐标）、5 自由度 SCARA 机械手关节坐标系定义（广义坐标）所示。单位为度 一般为虚拟轴，用户对该 5 个轴进行编程，对工具坐标系的位姿进行规划
arrJointCoordAxisID	INPUT	POINTER TO USINT	关节坐标系各关节轴轴号数组，共 5 个元素，依次对应关节 1~5。如图 5 自由度 SCARA 机械手基坐标系、工具坐标系定义（笛卡尔坐标）、5 自由度 SCARA 机械手关节坐标系定义（广义坐标）所示。 物理输出轴，驱动机构 5 个关节运动
arrScaraLinkLength	INPUT	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 5 自由度 SCARA 机械手关节

			角零点位姿中 L1、L2、L3
HMC_Scara5Control	OUTPUT	UDINT	指令 ID：成功 16#FFFFFFFF：错误

5.6.2.1.3 指令类型

轴运动缓存指令。

5.6.2.1.4 补充说明



- 1：动平台
- 2：上球铰
- 3：液压缸/电动缸
- 4：下球铰
- 5：静平台

六自由度并联平台的机构原理如上图所示，系统由动、静平台和 6 个伺服电动缸(或伺服阀控液压缸)组

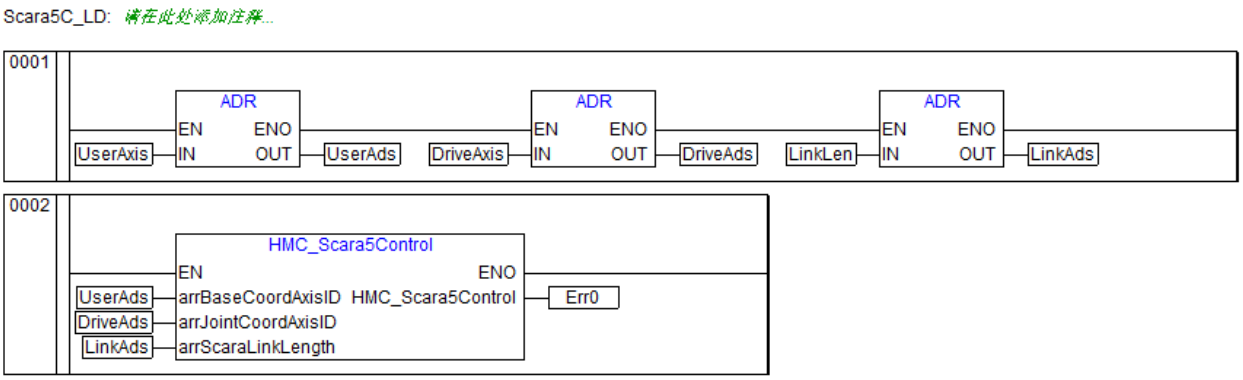
成，各支链电动缸与动、静平台分别由球铰连接（实际使用中为消除局部自由度，平台的上铰点通常使用虎克铰连接）。一般静平台固定于基座，通过控制支链电动缸的伸缩运动，动平台可以实现空间运动，完成空间六自由度的位置及姿态调整。并联机构本身的刚度大，当使用液压缸驱动时，特别适用于重载情况下对各种空间运动的模拟。

5.6.2.1.5 示例

任务一：投送 Scara5Control 指令

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	UserAxis		用户基坐标轴数组	ARRAY[0..4] OF BYTE		FALSE
0002	DriveAxis		关节驱动轴数组	ARRAY[0..4] OF BYTE		FALSE
0003	LinkLen		杆长数组	ARRAY[0..2] OF LREAL		FALSE
0004	Err0		返回值	UDINT	0	FALSE

- LD 程序：



- ST 程序

```
1   (*投送Scara5Control指令*)
2   Err0 := HMC_Scara5Control(ADR(UserAxis), ADR (DriveAxis), ADR (LinkLen));
```

- 变量定义

用户基坐标轴号数组：（不允许存在轴号重复或与关节驱动轴号重复的情况）

Scara5C_ST.UserAxis

序号	变量名	直接地址	变量说明	变量类型	初始值
0001	UserAxis[0]		用户基坐标X	BYTE	6
0002	UserAxis[1]		用户基坐标Y	BYTE	7
0003	UserAxis[2]		用户基坐标Z	BYTE	8
0004	UserAxis[3]		用户旋转坐标A	BYTE	9
0005	UserAxis[4]		用户旋转坐标B	BYTE	10

关节驱动轴号数组：（不允许存在编码器轴类型或未使用轴类型）

Scara5C_ST.DriveAxis

序号	变量名	直接地址	变量说明	变量类型	初始值
0001	DriveAxis[0]		关节1驱动轴	BYTE	1
0002	DriveAxis[1]		关节2驱动轴	BYTE	2
0003	DriveAxis[2]		关节3驱动轴	BYTE	3
0004	DriveAxis[3]		关节4驱动轴	BYTE	4
0005	DriveAxis[4]		关节5驱动轴	BYTE	5

杆长参数：（当杆长设置不合理时，指令投送不成功）

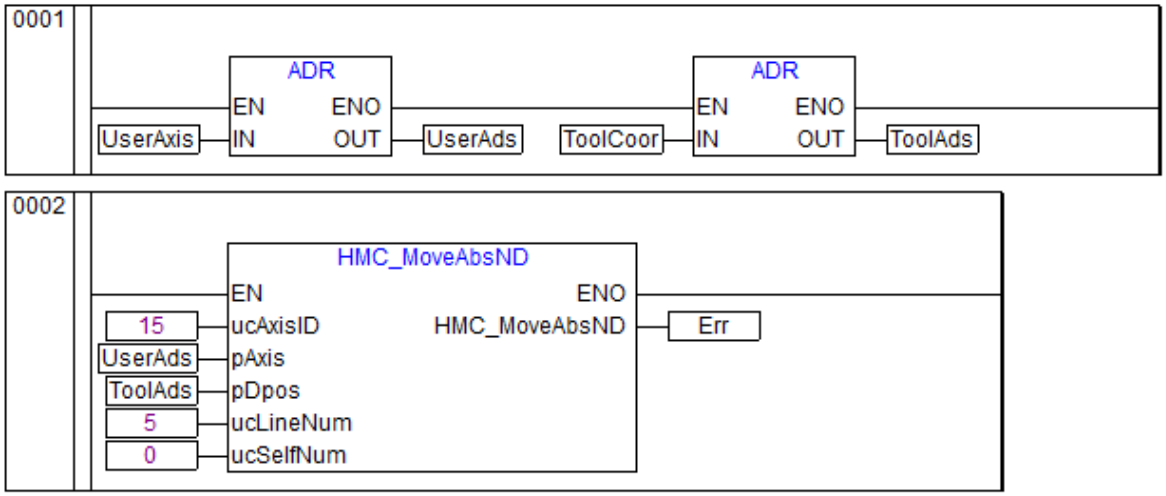
Scara5C_ST.LinkLen

序号	变量名	直接地址	变量说明	变量类型	初始值
0001	LinkLen[0]		杆1长度	LREAL	10000
0002	LinkLen[1]		杆2长度	LREAL	11000
0003	LinkLen[2]		杆3长度	LREAL	12000

任务二：用户基坐标轴中投送运动控制指令(MoveAbsND)，给定用户基坐标系中的工具坐标。

序号	变量名	直接地址	变量说明	变量类型
0002	ToolCoor		用户基坐标轴数组	ARRAY[0..4] OF LREAL
0001	UserAxis		工具坐标	ARRAY[0..4] OF BYTE

- LD 程序：



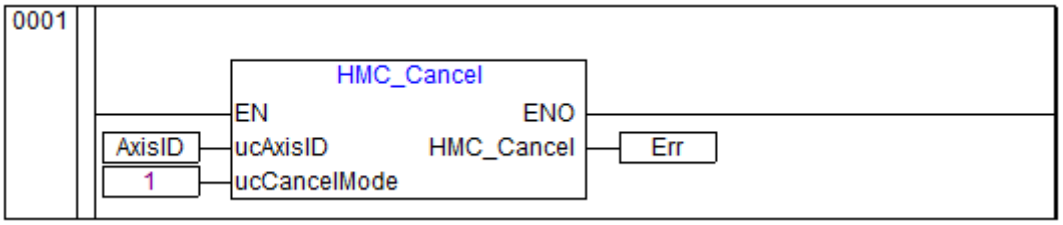
■ ST 程序：

```
1  (*用户基坐标投送MoveAbsND指令，给定用户基坐标中的工具坐标*)
2  HMC_MoveAbsND(15, ADR(UserAxis), ADR(ToolCoor), 5, 0);
```

任务三：撤销 Scara5Control 指令（AxisID 必需是驱动轴号数组中的最小轴号）

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	AxisID		驱动轴数组第一个轴号	BYTE	0	FALSE

■ LD 程序：



■ ST 程序：

```
1   (*撤销StewartControl指令*)
2   HMC_Cancel(AxisID, 1);
```

5.6.2.2 HMC_Scara5ForwardKinematics（5 自由度 SCARA 运动学正解）

5.6.2.2.1 功能

该指令完成 5 自由度 SCARA 机构的运动学正解（由关节坐标系到基坐标系）。

5.6.2.2.2 参数

参数	声明	数据类型	说明
arrScaraLinkLength	INPUT	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 5 自由度 SCARA 机械手关节角零点位姿中 L1、L2、L3
arrToolCoordinate	INPUT	POINTER TO LREAL	工具坐标数组，共 5 个元素，保存正解结果
arrJointAngle	INPUT	POINTER TO LREAL	关节坐标数组，共 5 个元素
HMC_Scara5ForwardKinematics	OUTPUT	UDINT	0：成功 16#FFFFFFFF：错误

5.6.2.2.3 指令类型

非轴运动缓存指令。

5.6.2.3 HMC_Scara5ReverseKinematics（5 自由度 SCARA 运动学逆解）

5.6.2.3.1 功能

该指令完成 5 自由度 SCARA 机构的运动学逆解（由基坐标系到关节坐标系）。当输入的工具坐标超出机构的可达工作空间时，返回错误。

该函数可用在如下方面：

- 检验关键点的工具坐标是否位于可达工作空间之内；
- 实现关节坐标空间编程。

关节坐标空间编程可避免直角坐标系下编程时关键点之间的运动轨迹可能穿过机构不可达工作空间的问题。

如果仅要求点到点之间的定位运动，对中间运动轨迹不作要求，则可在基坐标系中规划好关键位置点的工具坐标，然后使用该指令对关键点进行合法性检查，将合法的关键点的工具坐标转换为关节坐标，此后便可在关节坐标系内编程。

5.6.2.3.2 参数

参数	声明	数据类型	说明
arrScaraLinkLength	INPUT	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 5 自由度 SCARA 机械手关节角零点位姿中 L1、L2、L3
arrToolCoordinate	INPUT	POINTER TO LREAL	工具坐标数组，共 5 个元素，即（x,y,z,φA,φB）
arrReferenceJointAngle	INPUT	POINTER TO LREAL	关节坐标数组（参照值），共 5 个元素，依次为关节 1~5 的旋转角度 由于逆解的结果不止一个，选择距离该参照位置 较近的一组解作为逆解结果
arrJointAngle	INPUT	POINTER TO LREAL	关节坐标数组，共 5 个元素，保存逆解结果，依次为关节 1~5 的旋转角度
HMC_Scara5ReverseKinematics	OUTPUT	UDINT	0：成功 16#FFFFFFFF：错误

5.6.2.3.3 指令类型

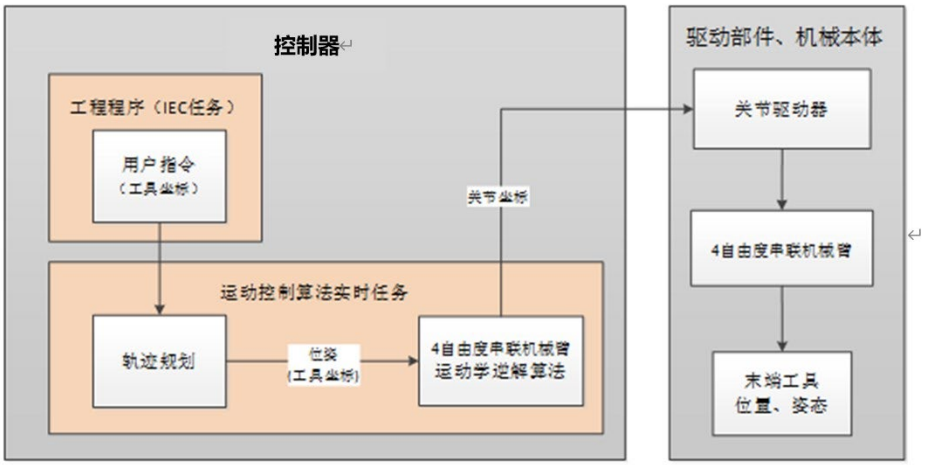
非轴运动缓存指令。

5.6.3 4 自由度串联机械臂

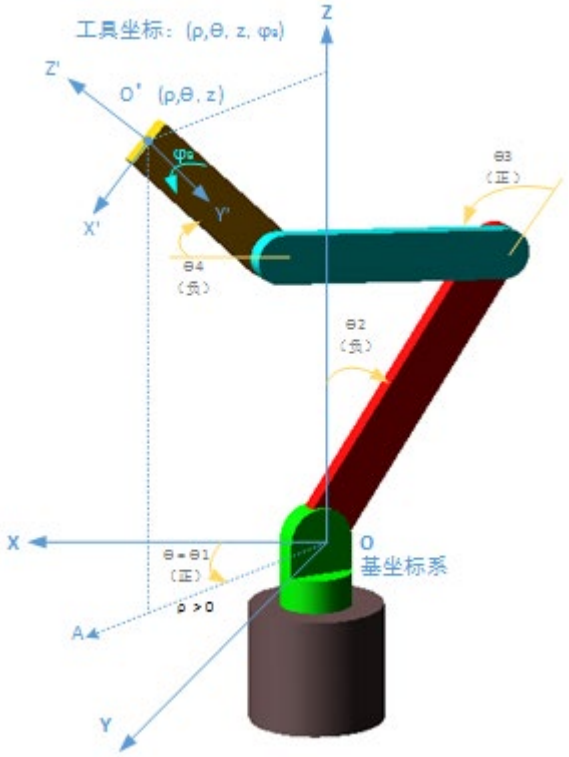
5.6.3.1 HMC_Serial4Control（4 自由度串联机械臂控制）

5.6.3.1.1 功能

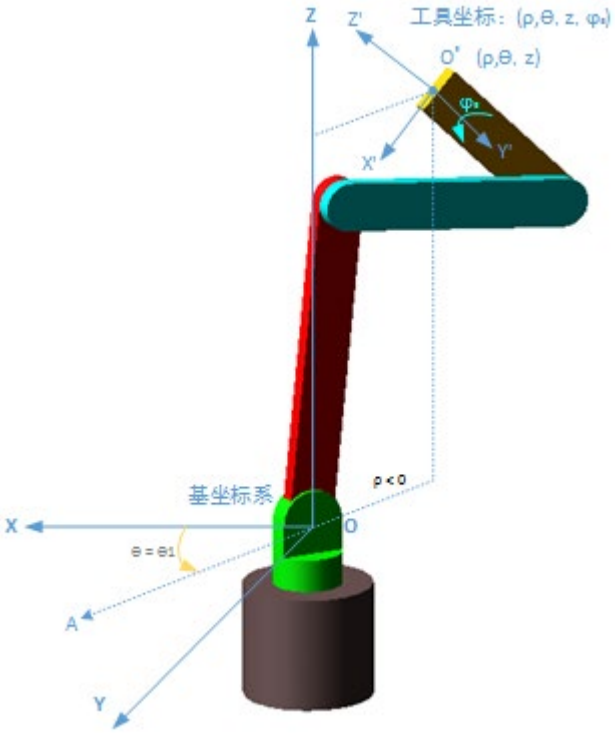
该指令完成 4 自由度串联机械臂的运动学变换（由工具坐标到关节坐标）。指令生效后，用户便可使用工具坐标（圆柱坐标）编写工程程序，控制器自动完成机构的运动学逆解变换，驱动各关节运动。过程如下图所示。



4 自由度串联机械臂控制系统框图



（a）

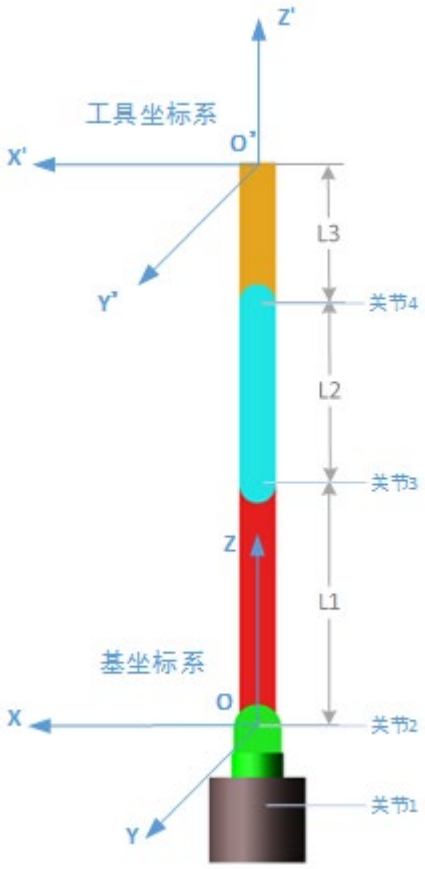


（b）

4 自由度机械手工具坐标

（1） 图（a）为 4 自由度机械手 工具坐标（圆柱坐标，ρ > 0 的情形）、关节坐标示意；

（2） 图（b）为 4 自由度机械手 工具坐标（圆柱坐标，ρ < 0 的情形）、关节坐标示意。



自由度串联机械臂关节角零点位姿

定义工具坐标系 O'-X'Y'Z'，如图 4 自由度机械手工具坐标、图自由度串联机械臂关节角零点位姿所示。

定义基坐标系为圆柱坐标系，因此工具坐标采用圆柱坐标描述，记为（ρ,Θ,z,φ_B）。其中 φ_B 为末端执行工具绕 Y'轴的旋转角度，旋转正方向参照右手定则。Θ 为关节 1 的旋转角度，ρ 为极坐标半径。为全面的描述末端工具的位姿，这里对 ρ 进行了扩展，可正可负，对应的位姿分别如图 4 自由度机械手_工具坐标（a）、图 4 自由度机械手_工具坐标（b）所示。

关节 1、2、3、4 的旋转角度称为关节坐标，记作（Θ_1,Θ_2,Θ_3,Θ_4），单位为度，如图 4 自由度机械手工具坐标所示。各关节角零点位置如图自由度串联机械臂关节角零点位姿所示。

在回零操作完成后，只要关节角零点位置未发生改变，则可在任意位置调用该指令，而不必调整到图自由度串联机械臂关节角零点位姿中位置。

使用该指令时需注意以下几点：

（1） 长度单位为用户单位，角度单位为角度制（°）；

（2） 务必保证关节角的零点位置在图自由度串联机械臂关节角零点位姿中位置；

（3） 可使用 HMC_Cancel(AxisID,CancelMode)撤销该指令，撤销时传入的轴号必须为 4 个关节轴中的最小轴号；

（4） 指令调用成功后，首次被执行时，会根据 当前关节角 初始化 工具坐标，之后在此初始位姿基础上运动；

（5） 指令首次被执行时会把关节坐标轴及工具坐标轴的 RepMode 设定为有限行程，在指令运行过程中不允许修改 RepMode，否则可能会发生坐标转换错误；

（6） 可根据机构实际情况设定机构各关节角的软限位值，触发限位后指令自动撤销；

（7） 调用该指令初始化成功后，用户使用 工具坐标（圆柱坐标） 编程；

（8） 不允许调用 HMC_DefPos 或 HMC_DefOrigin 指令重新设定工具坐标的坐标值，或对工具坐标轴使用 HOME 指令，否则坐标转换时会产生严重的偏差。如果用户需要自定义相对坐标系，可在工程程序中自行进行坐标偏移换算；

（9） 机构存的可达工作空间是有限的，如果规划的路径超出了其可达工作空间，出于安全考虑，指令会被撤销。请根据机构参数，合理规划工作空间，确保工作空间位于机构可达工作空间之内（可用 HMC_Serial4ReverseKinematics 指令校验规划的工具坐标是否合法）。

5.6.3.1.2 参数

参数	声明	数据类型	说明
arrToolCoordAxisID	INPUT	POINTER TO USINT	工具坐标对应的轴号数组，共 4 个元素，依次对应圆柱坐标中的（ρ,Θ,z,φB）所使用的坐标轴 一般为虚拟轴，用户在这 4 个轴中使用圆柱坐标编程，规划工具坐标
arrJointCoordAxisID	INPUT	POINTER TO USINT	关节轴轴号数组，共 4 个元素，依次对应关节 1~4。如图自由度串联机械臂关节角零点位姿所示 物理输出轴，驱动 4 个关节运动
arrLinkLength	INPUT	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图自由度串联机械臂关节角零点位姿中 L1、L2、L3

HMC_Serial4Control	OUTPUT	UDINT	0：成功 16#FFFFFFFF：错误

5.6.3.1.3 指令类型

轴运动缓存指令。

5.6.3.1.4 使用示例

指令使用步骤如下：

（10） 参数定义

```
(*杆长参数数组*)
arrLinkLength[0] := 200; ( *L1*)
arrLinkLength[1] := 150; ( *L2*)
arrLinkLength[2] := 100; ( *L3*)

(*关节轴轴号数组*)
arrJointCoordAxisID[0] := 0; ( *关节 1*)
arrJointCoordAxisID[1] := 1; ( *关节 2*)
arrJointCoordAxisID[2] := 2; ( *关节 3*)
arrJointCoordAxisID[3] := 3; ( *关节 4*)

(*工具坐标轴号数组*)
arrToolCoordAxisID[0] := 10;(*工具坐标 ρ 对应轴号*)
arrToolCoordAxisID[1] := 11;(*工具坐标 Θ 对应轴号*)
arrToolCoordAxisID[2] := 12;(*工具坐标 z 对应轴号*)
arrToolCoordAxisID[3] := 13;(*工具坐标 φ_B 对应轴号*)
```

（11） 定义各关节轴的零点在如图自由度串联机械臂关节角零点位姿所示的位置（关节轴实际位置不必在零位）

（12） 向关节轴中投送 HMC_Serial4Control 指令

HMC_Serial4Control(arrToolCoordAxisID , arrJointCoordAxisID , arrLinkLength);

（13） 使用工具坐标轴编程，坐标为圆柱坐标系

- 移动到图 4 自由度机械手工具坐标（a）中位姿，（ρ,Θ,z,φ_B）=（100, 30°,250,45°）

HMC_MoveAbs(arrToolCoordAxisID[0], 100)；(*移动到 ρ = 100*)

HMC_MoveAbs(arrToolCoordAxisID[1], 30)； (*移动到 Θ = 30°*)

HMC_MoveAbs(arrToolCoordAxisID[2], 250)；(*移动到 z = 250*)

HMC_MoveAbs(arrToolCoordAxisID[3], 45)； (*末端工具绕 Y’轴旋转 φ_B = 45°*)

- 移动到图 4 自由度机械手工具坐标（b）中位姿，（ρ,Θ,z,φ_B）=（-100, 30°,250,45°）

HMC_MoveAbs(arrToolCoordAxisID[0], -100); (*移动到 ρ = -100*)

HMC_MoveAbs(arrToolCoordAxisID[1], 30)； (*移动到 Θ = 30°*)

HMC_MoveAbs(arrToolCoordAxisID[2], 250)；(*移动到 z = 250*)

HMC_MoveAbs(arrToolCoordAxisID[3], 45)； (*末端工具绕 Y’轴旋转 φ_B = 45°*)

- 如果需要经过一系列中间点，可使用样条插补指令。

5.6.3.2 HMC_Serial4ForwardKinematics（4 自由度串联机械臂运动学正解）

5.6.3.2.1 功能

该指令完成 4 自由度串联机械臂的运动学正解（由关节坐标到工具坐标）。

5.6.3.2.2 参数

参数	声明	数据类型	说明

arrLinkLength	INPUT	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图自由度串联机械臂关节角零点位姿中 L1、L2、L3
arrToolCoordinate	INPUT	POINTER TO LREAL	工具坐标数组，共 4 个元素（ρ,Θ,z,φB）
arrReferenceJointAngle	INPUT	POINTER TO LREAL	关节坐标数组（参照值），共 4 个元素，依次为关节 1~4 的旋转角度 由于逆解的结果不止一个，选择距离该参照位置 较近的一组解作为逆解结果
arrJointAngle	INPUT	POINTER TO LREAL	关节坐标数组，共 4 个元素，保存逆解结果，依次为关节 1~4 的旋转角度
HMC_Serial4ReverseKinematics	OUTPUT	UDINT	0：成功 16#FFFFFFFF：错误

5.6.3.3 HMC_Serial4ReverseKinematics（4 自由度串联机械臂运动学逆解）

5.6.3.3.1 功能

该指令完成 4 自由度串联机械臂的运动学逆解（由工具坐标到关节坐标）。当输入的工具坐标超出机构的可达工作空间时，返回错误。

该函数可用在如下方面：

- 检验关键点的工具坐标是否位于可达工作空间之内；
- 实现关节坐标空间编程。

关节坐标编程可避免工具坐标编程时的诸多问题，如关键点之间的运动轨迹可能穿过机构不可达工作空间的问题、奇异位形问题等。

如果仅要求点到点之间的定位运动，对中间运动轨迹不作要求，则可首先规划好关键位置点的工具坐标，然后使用该指令对关键点进行合法性检查，将合法的关键点的工具坐标转换为关节坐标，此后便可使用关节坐标编程。

5.6.3.3.2 参数

参数	声明	数据类型	说明

arrLinkLength	INPUT	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图自由度串联机械臂关节角零点位姿中 L1、L2、L3
arrToolCoordinate	INPUT	POINTER TO LREAL	工具坐标数组，共 4 个元素（ρ,Θ,z,φB）
arrReferenceJointAngle	INPUT	POINTER TO LREAL	关节坐标数组（参照值），共 4 个元素，依次为关节 1~4 的旋转角度 由于逆解的结果不止一个，选择距离该参照位置 较近的一组解作为逆解结果
arrJointAngle	INPUT	POINTER TO LREAL	关节坐标数组，共 4 个元素，保存逆解结果，依次为关节 1~4 的旋转角度
HMC_Serial4ReverseKinematics	OUTPUT	UDINT	0：成功 16#FFFFFFFF：错误

5.6.3.3.3 指令类型

非轴运动缓存指令。

5.7 自定义运动控制功能

5.7.1 HMC_SetDposSwitch（Dpos 规划开关切换）

5.7.1.1 功能

该指令切换指定轴的 Dpos 规划开关，开关设定为 0 时 Dpos 由 MC 内部运动控制算法规划，为 1 时 Dpos 由用户通过 HMC_SetDposVal 指令规划。

Dpos 规划开关设定为 1 时，该轴运动缓存中的运动指令不再执行，也无法通过 HMC_Cancel 清除，因此最好在 Dpos 规划开关切换为 1 前清空该轴的运动缓存，并在开关设定为 1 期间不再往其中投送运动指令。

5.7.1.2 参数

参数	声明	数据类型	说明

ucAxisID	INPUT	USINT	轴号
ucDposSwitch	INPUT	USINT	Dpos 规划开关 0：Dpos 由内部运动控制算法规划 1：Dpos 由用户规划
HMC_SetDposSwitch	OUTPUT	UDINT	0：成功 16#FFFFFFFF：错误

5.7.1.3 指令类型

非轴运动缓存指令。

5.7.2 HMC_SetDposVal（Dpos 规划）

5.7.2.1 功能

Dpos 规划开关设定为 1 时，该指令设定轴的 Dpos 为指定值；Dpos 规划开关设定为 0 时，该指令无作用。

使用该指令设定 Dpos 后，如果想要等到设定的 Dpos 生效后再执行下一步动作，可在其后调用 HMC_SynchroToAlm 指令，HMC_SynchroToAlm 会与 MC 内部的硬实时任务有一个同步操作，这样直到设定的 Dpos 值被硬实时任务处理生效后才退出指令（硬实时任务每个伺服周期执行一次）。

因此，如果想要使用 HMC_SetDposVal 达到每个伺服周期更新一次 Dpos 的目的，可在一个循环中反复调用 HMC_SetDposVal，并在其后调用 HMC_SynchroToAlm 指令，同时使系统的负荷尽量降低（把用不到的轴全部关闭（轴类型设定为-1），并尽可能的减少处于运行状态的 IEC 任务的数量）。

5.7.2.2 参数

参数	声明	数据类型	说明
ucAxisID	INPUT	USINT	轴号
dSetDposVal	INPUT	LREAL	设定的 Dpos 值
HMC_SetDposVal	OUTPUT	UDINT	0：成功 16#FFFFFFFF：错误

5.7.2.3 指令类型

非轴运动缓存指令。

5.7.2.4 示例

设定 Dpos 规划开关为 1，合理规划 Dpos，使 0 轴和 1 轴位置同步，并且速度曲线满足正弦曲线规律。代码如下：

（1） 初始化，任务 1。

程序：

```
(*关闭用不到的轴*)
FOR AxisID:=2 TO 15  BY 1 DO
HMC_SetAxisType(AxisID, -1);
END_FOR
HMC_SetAxisType(0, 0); (*0 轴为虚拟轴*)
HMC_SetAxisType(1, 0); (*1 轴为虚拟轴*)
HMC_SetDposSwitch(0,1);  (*0 轴 Dpos 规划开关设为 1*)
HMC_SetDposSwitch(1,1);  (*1 轴 Dpos 规划开关设为 1*)
HMC_DefPos(0,0);  (*0 轴位置初始化*)
HMC_DefPos(1,0);  (*0 轴位置初始化*)
```

（2） Dpos 规划，任务 2。

程序：

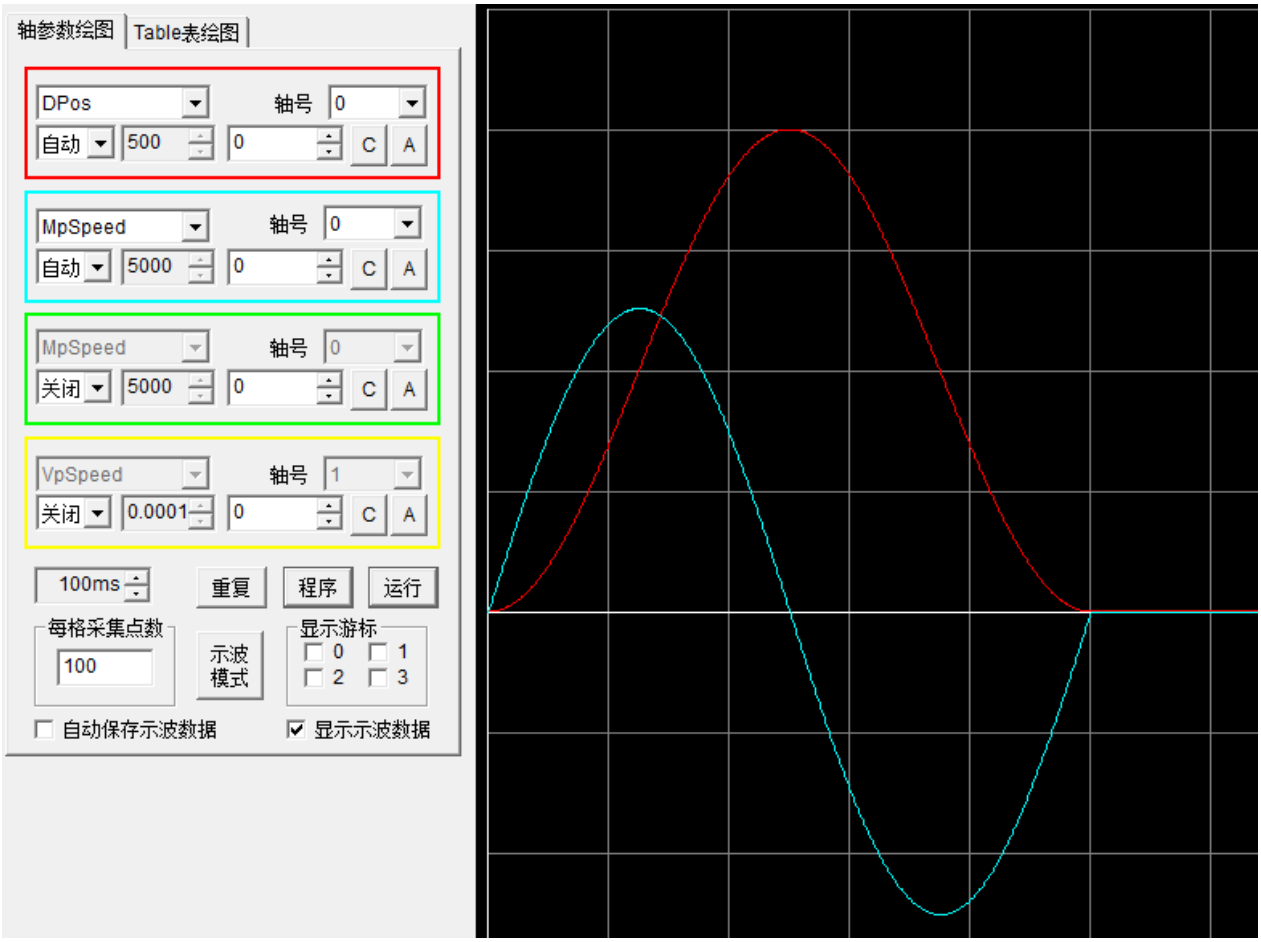
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Dpos			LREAL	0	FALSE
0002	count			DINT	0	FALSE
0003	MaxNum		运行的伺服周期数	DINT	500	FALSE
0004	PI		π	LREAL	3.1415926	FALSE
0005	A		Dpos幅值	LREAL	1000	FALSE

```
count := 0;
WHILE TRUE DO
count := count + 1;
Dpos := A * ( 1 - COS(2*PI*count/MaxNum) ); (*目标位置计算*)
HMC_SetDposVal(0, Dpos);(*0 轴 Dpos 规划*)
HMC_SetDposVal(1, Dpos);(*1 轴 Dpos 规划*)
HMC_SynchroToAlm (); (*同步到算法硬实时任务*)
IF count = MaxNum THEN (*控制运行次数，如果不需要控制次数也可一直循环*)
EXIT;
END_IF
END_WHILE
```

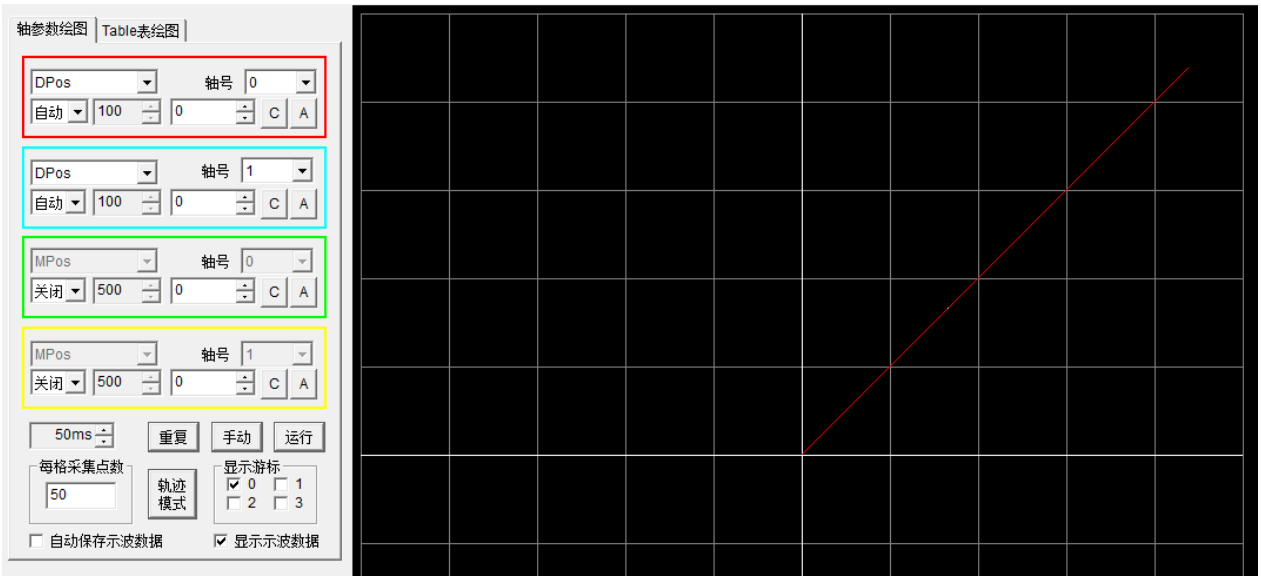
首先运行初始化任务 1，然后在终端输入如下指令：

```
HMC_TriggerScope();TaskStart(2);
```

运行结果如下（伺服周期 1ms）：



轴 0 Dpos 及 MpSpeed



轴 0、轴 1 合运动轨迹

5.7.3 HMC_SynchroToAlm（同步到算法任务）

5.7.3.1 功能

该指令可用于与 MC 内部的算法硬实时任务的同步。调用该指令后，会等到硬实时任务执行过一次后才退出指令（硬实时任务每个伺服周期执行一次）。

5.7.3.2 参数

参数	声明	数据类型	说明
0	OUTPUT	UDINT	成功

5.7.3.3 指令类型

非轴运动缓存指令。

5.8 轴参数操作指令

5.8.1 HMC_SetAxisType（设置轴类型）

5.8.1.1 功能

设置指定轴的轴类型，各轴类型含义详见轴参数 AxisType。

1. 轴号范围 0~63。
2. 当该轴的轴指令缓存中有指令时，调用此函数设置轴类型不成功。
3. 轴类型设置成功后，系统会自动根据不同的轴类型会关联修改轴参数 RepMode。

当用户设置某个轴类型为虚轴或编码器轴时，系统将设置 RepMode=2；当用户设置某个轴类型为其他轴类型时，系统将设置 RepMode=0。

5.8.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
EmAxisType	Input	EMAXISTYPE	轴类型
HMC_SetAxisType	Output	UDINT	0：设置成功 其他值：错误代码

5.8.1.3 指令类型

非轴运动缓存指令。

5.8.1.4 示例

```
HMC_SetAxisType(0,10);      (*设置轴 0 为直连步进轴*)
HMC_SetAxisType(1,20);      (*设置轴 1 为直连伺服轴*)
HMC_SetAxisType(60,-1);     (*设置轴 60 为未使用轴*)
```

5.8.2 HMC_HwLimitConfig（设置硬限位开关）

5.8.2.1 功能

设置指定轴的硬限位开关，限位功能仅对有限行程轴有效。

5.8.2.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucLimitType	Input	USINT	0：设置前向硬限位（常开） 1：设置后向硬限位（常开） 2：设置前向硬限位（常闭）

			3：设置后向硬限位（常闭）
sDiChan	Input	UDINT	关联的 DI 通道（0-2097151） 当 sDiChan 为-1 时，表示取消硬限位开关，仅对有限行程轴有效，指定轴将不再关联原来的前（后）向限位 DI 通道，即使当前处于限位状态也会立即解除
HMC_HwLimitConfig	Output	UDINT	0:设置成功，其他值：错误代码

5.8.2.3 指令类型

非轴运动缓存指令。

5.8.2.4 示例

配置第 5 路 DI 为轴 0 的前向限位开关，第 6 路 DI 为轴 0 的后向限位开关。

```
HMC_HwLimitConfig (0 ,0, 5);
HMC_HwLimitConfig (0 ,1, 6);
```

5.8.3 HMC_SetUnits（设置用户单位）

5.8.3.1 功能

设置指定轴的单位转换因子轴参 Units，该参数只可为正值，单位：线/用户单位，”线”是指轴走一个脉冲对应的刻度单位。

通过设置轴参 Units，可将编码器反馈的计数值或脉冲输出值转换为用户方便使用的单位数值，例如 mm、角度等。

5.8.3.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dUnits	Input	LREAL	用户单位，不可为 0 或负数
HMC_SetUnits	Output	UDINT	0:设置成功

			其他值：错误代码

5.8.3.3 指令类型

非轴运动缓存指令。

5.8.3.4 示例

例如某轴走 10000 个脉冲正好转一圈，即转一圈对应 10000 线；假如说，通过机械结构的设计，该轴每转一圈（10000 线）带动终端走 10 mm，那么如下两种方法是等价的，都可以让终端走 50 mm（即电机转动 5 圈）：

方法 1：

```
HMC_SetUnits(0,1000); (*10000 线/10mm=1000 线/mm*)
HMC_Move(0,50); (/*终端走 50 mm*/)
```

方法 2：

```
HMC_SetUnits(0,1);
HMC_Move(0,50000);      (/*电机走 50,000 线；终端走 50 mm*/)
```

上述两种方法中选择哪一种取决于用户的使用习惯。

5.8.4 HMC_SetJerkMode（设置梯形/S 形加速度）

5.8.4.1 功能

设置 JerkMode。JerkMode 为 0 时运动过程速度曲线为梯形加减速曲线，JerkMode 为 1 时为 S 形加减速曲线。

5.8.4.2 参数

参数	声明	数据类	说明

		型	
ucAxisID	Input	USINT	轴号
ucJerkMode	Input	USINT	JerkMode（默认值 0）。 0：运动过程速度曲线为梯形加减速曲线； 1：运动过程速度曲线为 S 形加减速曲线。
HMC_SetJerkMode	Output	UDINT	0:设置成功 其他值：错误代码

5.8.4.3 指令类型

非轴运动缓存指令。

5.8.5 HMC_SetKe（设置 Ke）

5.8.5.1 功能

设置轴参数 Ke 的分子（轴参数 KeNumerator）和分母（轴参数 KeDenominator），参考轴参数 KeNumerator 和 KeDenominator 的说明。

5.8.5.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
iKeNumerator	Input	DINT	Ke 分子
iKeDenominator	Input	DINT	Ke 分母
HMC_SetKe	Output	UDINT	0:设置成功 其他值：错误代码

5.8.5.3 指令类型

非轴运动缓存指令。

5.8.5.4 补充说明

Ke 可设置为负，设置为负可对实际反馈速度及位置方向取反（部分低版本固件不支持）。

Ke 不可为 0。

5.8.5.5 示例

转盘由一个电机驱动控制旋转，另有编码器连接至该转盘，测量其旋转角度。

转盘旋转一周，该编码器输出 8192 个脉冲。期望在软件上直接显示该转盘的旋转角度值，由于 8192/360 无法整除，为了达到最高精度，此时可通过设置 Ke 实现。

```
HMC_SetAxisType(0,50);
HMC_SetUnits(0,1);
HMC_SetKe(0,360,8192);      (*配合 Units，使得该轴单位为角度，Mpos 直接显示角度*)
```

5.8.5.6 使用注意事项

该参数仅对伺服轴和编码器轴生效。该参数和 Units 有类似的作用，(Ke /Units)共同构成了伺服轴或编码器轴的“单位转换因子”，其单位为“线/用户单位”，参见 Units（单位转换因子）

5.8.6 HMC_SetKs（设置 Ks）

5.8.6.1 功能

设置轴参数步进输出比例 Ks 的分子（轴参数 KsNumerator）和分母（轴参数 KsDenominator），参考轴参数 KsNumerator 和 KsDenominator 的说明。

5.8.6.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号

iKsNumerator	Input	DINT	Ks 分子
iKsDenominator	Input	DINT	Ks 分母
HMC_SetKs	Output	UDINT	0:设置成功 其他值：错误代码

5.8.6.3 指令类型

非轴运动缓存指令。

5.8.6.4 补充说明

Ks 需＞0。

5.8.6.5 示例

转盘由一个步进电机驱动控制旋转。该步进电机输出 8192 个脉冲可驱动转盘旋转 1 圈。期望在软件上直接显示该转盘的旋转角度值，由于 8192/360 无法整除，为了达到最高精度，此时可通过设置 Ks 实现。

```
HMC_SetAxisType(0,50);
HMC_SetUnits(0,1);
HMC_SetKs(0,8192,360);
HMC_Move(0,180);    (*驱动转盘旋转 180 度*)
```

5.8.7 HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址）

5.8.7.1 功能

配置轴实际位置 Mpos 的用户自定义输入源。当轴参数 FeedBackSrcMode=1 时，该轴的实际位置 Mpos 由用户自定义输入源增量计算更新，此时 Mpos 与该轴的实际物理输入无任何关系。

当轴 Mpos 计算来自用户自定义输入源时，依据自定义输入源的变化增量累积计算 Mpos，轴参 Units 同样参与 Mpos 计算。

- 注意：该功能仅对有实际物理输入的轴类型生效，如总线轴等。

5.8.7.2 参数

参数	声明	数据类型	说明
ucAxis	Input	USINT	轴号
pAddr	Input	POINTER TO LREAL	用户自定义变量地址
DataType	Input	USINT	用户自定义变量的数据类型：USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、F32=7、LREAL=8
HMC_SetFeedBackSrcAddr	Output	UDINT	0:设置成功 其他值：错误代码

5.8.7.3 指令类型

非轴运动缓存指令。

5.8.7.4 示例

现有一位置传感器，其输出信号为电压信号，待测量位置在 0~100000，对应位置传感器输出电压信号 0~10V。

将位置传感器输出的电压信号连接至第 1 路 AI。用轴 0 实时计算该位置信息。

```
HMC_SetAxisType (0,50);
axis0.FeedBackSrcMode := 1;
HMC_SetUnits(0, 10.0/100000.0);   (*设置位置和电压信号转换系数为单位*)
HMC_SetFeedBackSrcAddr(0, ADR(AI0), 4);      (*设置第 1 路 AI 为轴 0 的 Mpos 计算源*)
HMC_Defpos(0, AI0*(100000/10));   (*初始化轴 0 的位置为当前测量的绝对位置，后续在此基础上增量计算位置值*)
```

5.8.8 HMC_SetOutDstAddr（设置用户自定义输出地址）

5.8.8.1 功能

设置位置闭环输出值 DACOut 的自定义输出地址。位置闭环模式时，计算的输出值 DACOut 输出给该轴的物理输出通道，如伺服 20 模式时，输出给该轴对应的 AO 通道；如总线速度模式时，输出给该轴对应的总线输出数据区。但当轴参数 OutDstMode=1 时，则停止将 DACOut 值输出至该轴物理输出通道，而是输出给该轴的自定义输出地址。

当 OutDstMode=1 时，如果为直连伺服轴，此时算法不修改该轴对应 AO 通道值，则此时若用户通过 AT 在线修改或 IEC 程序修改 AO 通道值会生效；对于总线速度模式轴，算法会立即输出速度 0，将对应驱动器停止。

- 注意：该功能仅位置闭环模式的轴类型生效，如总线轴、总线速度模式等。

5.8.8.2 参数

参数	声明	数据类型	说明
ucAxis	Input	USINT	轴号
pAddr	Input	POINTER TO DINT	用户自定义变量地址
DataType	Input	USINT	用户自定义变量的数据类型：USINT=1、SINT=2、UINT=3、INT=4、UDINT=5、DINT=6、F32=7、LREAL=8
HMC_SetOutDstAddr	Output	UDINT	0:设置成功 其他值：错误代码

5.8.8.3 指令类型

非轴运动缓存指令。

5.8.8.4 示例

设置伺服轴轴 0 的输出目标为系统变量 diOutPut。

```
HMC_SetOutDstAddr(0, ADR(diOutPut), 6);
```

5.8.9 HMC_SetCmdBufferLoadMode（设置轴指令加载模式）

5.8.9.1 功能

设置轴指令的加载模式。主要用于切向追踪拐角抬刀模式下的辅助处理。

在切向追踪指令处于拐角暂停抬刀状态下时，若使用 HMC_Cancel 撤销指令，由于在此时 XY 指令队列中的指令处于暂停状态，出于安全考虑，撤销完成后并不自动恢复 XY 轴中指令队列中的指令运行，若需要恢复 XY 中指令队列中后续指令的运行，用户需调用 HMC_SetCmdBufferLoadMode 把指令加载模式设定为正常加载模式（0）。

5.8.9.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
ucCmdLoadMode	Input	USINT	0：正常加载模式； 1：当前指令运行完成后，不加载下一条指令
HMC_SetCmdBufferLoadMode	Output	UDINT	0:设置成功 其他值：错误代码

5.8.9.3 指令类型

非轴运动缓存指令。

5.8.9.4 使用注意事项

在切向追踪指令处于拐角暂停抬刀状态下时，若使用 HMC_Cancel 撤销指令，由于在此时 XY 指令队列中的指令处于暂停状态，出于安全考虑，撤销完成后并不自动恢复 XY 轴中指令队列中的指令运行，若需要恢复 XY 中指令队列中后续指令的运行，用户需调用 HMC_SetCmdBufferLoadMode 把指令加载模式设定为正常加载模式（0）。

5.8.10 HMC_GetCmdBufferLoadMode（获取轴指令加载模式）

5.8.10.1 功能

获取轴指令的加载模式。

5.8.10.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_GetCmdBufferLoadMode	Output	USINT	0:正常加载模式 1：当前指令运行完成后，不加载下一条指令 255：错误代码

5.8.10.3 指令类型

非轴运动缓存指令。

5.8.11 HMC_SetCmdStopMaxTime (设置指令结束最大等待时间)

5.8.11.1 功能

设置指令结束最大等待时间。

5.8.11.2 参数

参数	声明	数据类型	说明
uiTime	Input	UDINT	时间（ms）

HMC_SetCmdStopMaxTime	Output	UDINT	0：成功 其他值：失败

5.8.11.3 指令类型

非轴运动缓存指令。

5.8.12 HMC_GetAxisSlaveAddr（获取轴对应的站地址）

5.8.12.1 功能

获取指定轴号对应的从站地址，输入为要获取地址的轴的轴号，轴号在设置时必须和实际组态设备轴号一致，输出为对应的从站站号，如果轴号设置为未配置的轴号或者超过轴号最大取值范围（0~63），则输出为 0xFFFF 。

5.8.12.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_GetAxisSlaveAddr	Output	UDINT	从站站号

5.8.12.3 指令类型

非轴运动缓存指令。

5.8.12.4 示例

以下示例通过 HMC_GetAxisSlaveAddr 指令获取轴号为 2 的从站地址，当 EN 使能时，该指令启动获取从站地址功能，并将获取到的从站地址结果返回到 SlaveAddr 变量，获取轴号为 2 的从站的地址为 1004，与实际组态配置的从站地址一致。

变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAddr		获取到的从站地址	UINT	1004	FALSE
0022	SlaveAxisID		轴号	USINT	2	FALSE

LD 程序实现：

ST 程序实现：

5.8.12.5 使用注意事项

输入的轴号要根据实际组态配置进行设置，才能获取到正确的从站地址。

5.8.13 HMC_GetAxisTorque（获取轴扭矩）

5.8.13.1 功能

获取指定轴号对应的从站实际扭矩值，输入为要获取扭矩值的轴的轴号，轴号在设置时必须和实际组态设备轴号一致，输出为对应从站设备的实际扭矩值，即 EtherCAT 对象字典 0x6077（Torque actual value）的值，单位为千分之一额定扭矩。如果设置轴号超过最大取值范围（0~63）或获取从站扭矩值出错，则输出为 0 。

5.8.13.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号

HMC_GetAxisTorque	Output	INT	从站实际扭矩值，单位为千分之一额定扭矩

5.8.13.3 指令类型

非轴运动缓存指令。

5.8.13.4 示例

以下示例通过 HMC_GetAxisTorque 指令获取轴号为 2 的从站的实际扭矩值，当 EN 使能时，该指令启动获取从站扭矩功能，并将获取到的从站实际扭矩值返回到 ActualTorque 变量，获取到的从站扭矩值与从站对象字典 0x6077（Torque actual value）的值一致。

变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAddr		获取到的从站地址	UINT	1004	FALSE
0022	SlaveAxisID		轴号	USINT	2	FALSE

LD 程序实现：

ST 程序实现：

5.8.13.5 使用注意事项

输入的轴号要根据实际组态配置进行设置，才能获取到对应的正确的从站实际扭矩值。

5.8.14 HMC_AxisPotInput（获取轴正向硬限位状态）

5.8.14.1 功能

获取指定轴号从站对应的轴正向硬限位状态，输入要获取的轴的轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴号的从站当前正向硬限位状态，状态为 1 表示当前位置已到达正向硬限位点，状态为 0 表示当前位置未到达正向硬限位点，如果设置轴号超过最大取值范围（0~63），输出为 0 。

获取到的正向硬限位状态与轴参数 LimitState（超限状态）的 bit0 含义相同。

5.8.14.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_AxisPotInput	Output	BOOL	从站正向硬限位状态 1：到达正向硬限位 0：未到达正向硬限位或轴号出错

5.8.14.3 指令类型

非轴运动缓存指令。

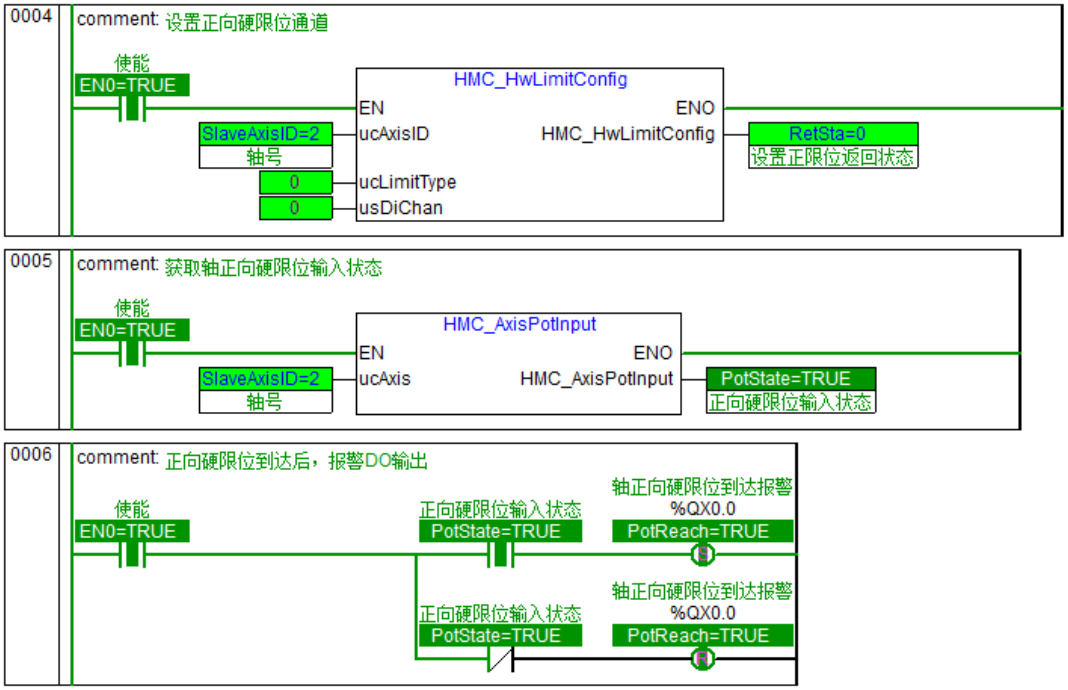
5.8.14.4 示例

以下示例通过 HMC_AxisPotInput 指令获取轴号为 2 的从站的正向硬限位状态，在获取之前，需要先指定正向硬限位通道，示例使用 HMC_HwLimitConfig 指令设置轴正向硬限位通道号为 0，当轴正向运动到达正向硬限位点时，HMC_AxisPotInput 指令获取到正向硬限位状态 PotState 为 1，报警 DO 输出。

变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAxisID		轴号	USINT	2	FALSE

LD 程序实现：



ST 程序实现：

```
10
11  (*设置正向硬限位通道*)
12  IF EN0 THEN
13      HMC_HwLimitConfig(SlvaAxisID,0,0);
14  END_IF
15
16  (*获取轴正向硬限位输入状态，状态为TRUE后，报警DO输出*)
17  IF EN0 THEN
18      PotState := HMC_AxisPotInput(SlvaAxisID);
19      IF PotState THEN
20          PotReach := TRUE;
21      ELSE
22          PotReach := FALSE;;
23      END_IF
24  END_IF
25
26  (*设置负向硬限位通道*)
27  IF EN0 THEN
28      HMC_HwLimitConfig(SlvaAxisID,1,1);
29  END_IF
```

EN0 = TRUE
SlvaAxisID = 2

EN0 = TRUE
PotState = TRUE　　SlvaAxisID = 2
PotState = TRUE
PotReach = TRUE

PotReach = TRUE

EN0 = TRUE
SlvaAxisID = 2

5.8.14.5 使用注意事项

正向硬限位只在有限行程时有效，即轴参数 RepMode=0 时有效，还需要设置通道号。

5.8.15 HMC_AxisNotInput（获取轴负向硬限位状态）

5.8.15.1 功能

获取指定轴号从站对应的轴负向硬限位状态，输入要获取的轴的轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴号的从站当前负向硬限位状态，状态为 1 表示当前位置已到达负向硬限位点，状态为 0 表示当前位置未到达负向硬限位点，如果设置轴号超过最大取值范围（0~63），输出为 0 。

获取到的负向硬限位状态与轴参数 LimitState（超限状态）的 bit1 含义相同。

5.8.15.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_AxisNotInput	Output	BOOL	从站负向硬限位状态 1：到达负向硬限位 0：未到达负向硬限位或轴号出错

5.8.15.3 指令类型

非轴运动缓存指令。

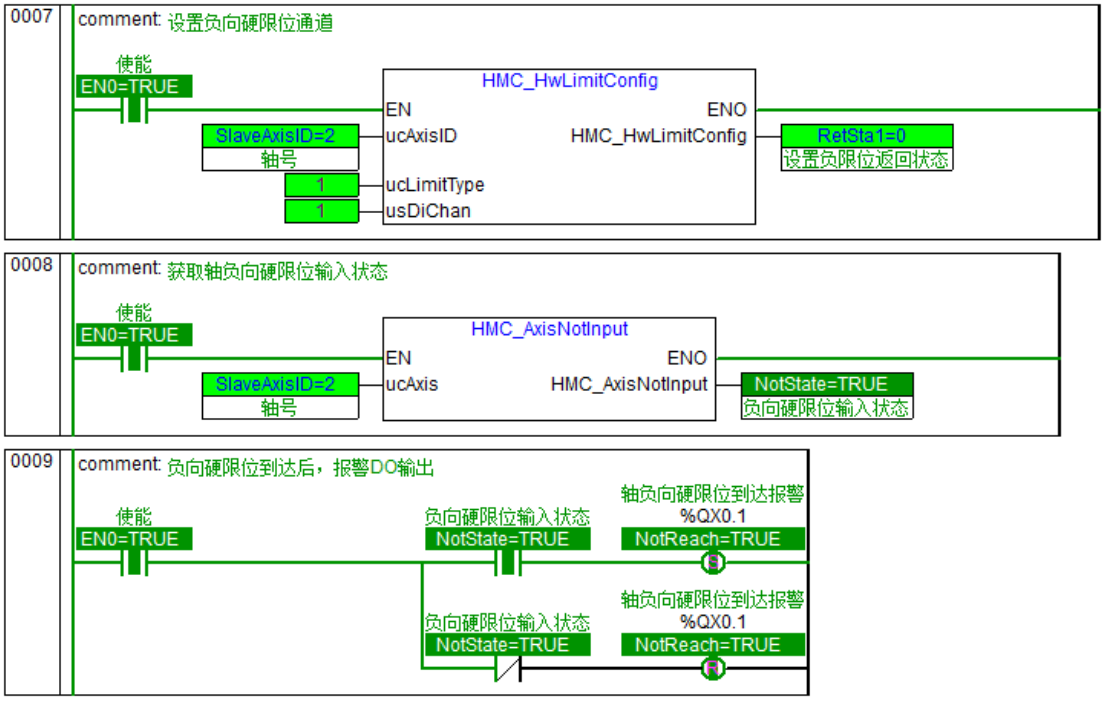
5.8.15.4 示例

以下示例通过 HMC_AxisNotInput 指令获取轴号为 2 的从站的负向硬限位状态，在获取之前，需要先指定负向硬限位通道，示例使用 HMC_HwLimitConfig 指令设置轴负向硬限位通道号为 1，当轴反向运动到达负向硬限位点时，HMC_AxisNotInput 指令获取到负向硬限位状态 NotState 为 1，报警 DO 输出。

变量定义：

0026	PotState		正向硬限位输入状态	BOOL	FALSE	FALSE
0027	NotState		负向硬限位输入状态	BOOL	FALSE	FALSE
0028	PotReach	%QX0.0	轴正向硬限位到达报警	BOOL	FALSE	FALSE
0029	NotReach	%QX0.1	轴负向硬限位到达报警	BOOL	FALSE	FALSE

LD 程序实现：



ST 程序实现：

```
25
26    (*设置负向硬限位通道*)
27    IF EN0 THEN
28        HMC_HwLimitConfig(SlvaAxisID,1,1);
29    END_IF
30
31    (*获取轴负向硬限位输入状态，状态为TRUE后，报警DO输出*)
32    IF EN0 THEN
33        NotState := HMC_AxisNotInput(SlvaAxisID);
34        IF NotState THEN
35            NotReach := TRUE;
36        ELSE
37            NotReach := FALSE;;
38        END_IF
39    END_IF
```

EN0 = TRUE
SlvaAxisID = 2

EN0 = TRUE
NotState = TRUE
NotState = TRUE
NotReach = TRUE

NotReach = TRUE

SlvaAxisID = 2

5.8.15.5 使用注意事项

负向硬限位只在有限行程时有效，即轴参数 RepMode=0 时有效，还需要设置通道号。

5.8.16 HMC_AxisStandStillStatus（获取轴停止状态）

5.8.16.1 功能

获取指定轴号从站是否处于停止状态，输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴当前是否处于停止状态，状态为 1 表示当前轴处于停止状态，即当前轴无运动控制指令，状态为 0 表示当前轴处于非停止状态，如果设置轴号超过最大取值范围（0~63），输出也为 0 。

5.8.16.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_AxisSlaveErrorStatus	Output	BOOL	从站当前的错误状态 1：有错误 0：无错误

5.8.16.3 指令类型

非轴运动缓存指令。

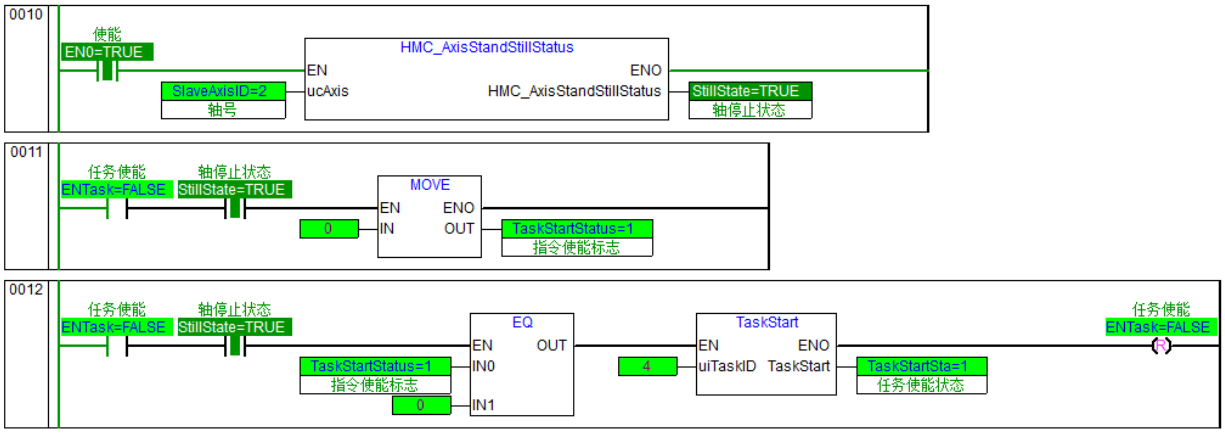
5.8.16.4 示例

以下示例通过 HMC_AxisStandStillStatus 指令获取轴号为 2 的从站的轴停止状态，当 EN 使能时，该指令启动获取从站轴停止状态功能，并将获取到的状态结果返回到 StillState 变量。该示例是在判断轴 2 从站轴停止状态为 1 时，即该轴当前无运动控制指令时，调用 TaskStart 启动任务 4，当任务 4 运行运动控制指令后，通过 HMC_AxisStandStillStatus 指令获取到的 StillState 状态为 0。

变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAxisID		轴号	USINT	2	FALSE

LD 程序实现：



ST 程序实现：

5.8.17 HMC_AxisSlaveErrorStatus（获取轴从站错误状态）

5.8.17.1 功能

获取指定轴号所关联从站的错误状态，输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴当前的错误状态，状态为 1 表示当前轴处于错误状态，状态为 0 表示当前轴无异常，如果设置轴号超过最大取值范围（0~63），输出也为 0 。

5.8.17.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_AxisSlaveErrorStatus	Output	BOOL	从站当前的错误状态。 1：有错误 0：无错误

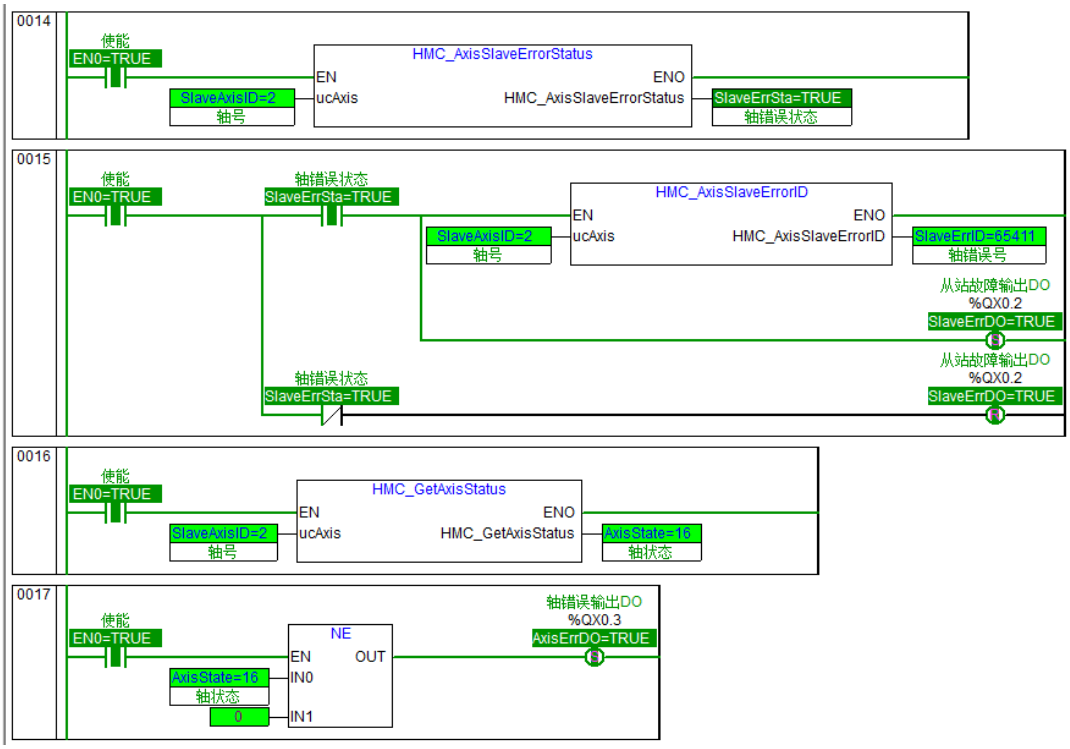
5.8.17.3 示例

以下示例通过相关指令获取轴号为 2 的从站错误状态、错误 ID 和轴状态。该示例使用欧姆龙伺服作为从站，从站轴号设为 2，在 EtherCAT 通信正常时，拔插 EtherCAT 通讯网线，伺服会报故障 0x83，此时通过 HMC_AxisSlaveErrorStatus 指令可以获取到从站错误状态为 1，通过 HMC_AxisSlaveErrorID 指令可以获取到从站错误码为 65411（0xFF83），与伺服报的故障码一致，通过 HMC_GetAxisStatus 可以获取到轴状态为 16，即总线从站报错。报故障后，对应的 DO 输出置位。

变量定义：

0033	SlaveErrSta		轴错误状态	BOOL	TRUE	FALSE
0034	SlaveErrID		轴错误号	UDINT	65411	FALSE
0035	SlaveErrDO	%QX0.2	从站故障输出DO	BOOL	TRUE	FALSE
0036	AxisState		轴状态	UDINT	16	FALSE
0037	AxisErrDO	%QX0.3	轴错误输出DO	BOOL	TRUE	FALSE

LD 程序实现：



ST 程序实现：

```
(*获取从站错误状态、错误码、轴状态*)
IF EN0 THEN
    SlaveErrSta := HMC_AxisSlaveErrorStatus(SlvaAxisID);

    IF SlaveErrSta THEN
        SlaveErrID := HMC_AxisSlaveErrorID(SlvaAxisID);
        SlaveErrDO := TRUE;
    ELSE
        SlaveErrDO := FALSE;;
    END_IF

    AxisState := HMC_GetAxisStatus(SlvaAxisID);

    IF AxisErrDO <> 0 THEN
        AxisErrDO := TRUE;
    ELSE
        AxisErrDO := FALSE;;
    END_IF

END_IF
```

EN0 = TRUE
SlaveErrSta = TRUE　　SlvaAxisID = 2
SlaveErrSta = TRUE
SlaveErrID = 65411　　SlvaAxisID = 2
SlaveErrDO = TRUE
SlaveErrDO = TRUE
AxisState = 16　　SlvaAxisID = 2
AxisErrDO = TRUE
AxisErrDO = TRUE
AxisErrDO = TRUE

5.8.18 HMC_AxisSlaveErrorID（获取轴从站错误号）

5.8.18.1 功能

获取指定轴号所关联从站的错误号，输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴当前的错误码，当该轴处于错误状态时，返回当前轴对应从站的实际错误码，如果设置轴号超过

最大取值范围（0~63），输出为 0，获取轴关联从站错误号出现错误时，返回 0xFFFFFFFF 。

具体的错误码信息参见对应从轴手册的故障描述。

5.8.18.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisStatus	Output	UDINT	从站当前的错误号

5.8.18.3 指令类型

非轴运动缓存指令。

5.8.18.4 示例

参见 HMC_AxisSlaveErrorStatus（获取轴从站错误状态）示例。

5.8.19 HMC_GetAxisStatus（获取轴参 AxisStatus）

5.8.19.1 功能

获取指定轴号从站的轴参数 AxisStatus（轴状态），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的轴状态信息，如果设置轴号超过最大取值范围（0~63），输出为 0 。

获取的轴状态信息与轴参数 AxisStatus（轴状态）一致，具体内容参见轴参数章节描述。

5.8.19.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisStatus	Output	UDINT	轴参 AxisStatus（轴状态）值

5.8.19.3 指令类型

非轴运动缓存指令。

5.8.19.4 示例

参见 HMC_AxisSlaveErrorStatus（获取轴从站错误状态）示例。

5.8.20 HMC_GetAxisMcmdType（获取轴参 McmdType）

5.8.20.1 功能

获取指定轴号从站的当前指令类型，输入为要获取轴参数的指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的当前指令类型（轴参数 McmdType），如果设置轴号超过最大取值范围（0~63）或当前轴指令为空，输出都为 0 。

获取到的轴状态信息与轴参数 McmdType（当前指令类型）一致，具体内容参见轴参数章节描述。

5.8.20.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisMcmdType	Output	UDINT	轴参 McmdType（当前指令类型）值

5.8.20.3 指令类型

非轴运动缓存指令。

5.8.20.4 示例

以下示例通过 HMC_GetAxisMcmdType 指令获取轴号为 2 的从站当前的指令类型，当 EN 使能时，该指令启动获取从站当前指令类型功能，并将获取到的指令类型结果返回到 CCmdType 变量，执行后获取到

的当前指令类型为 10:HMC_MOVE，与实际组态的逻辑配置一致。

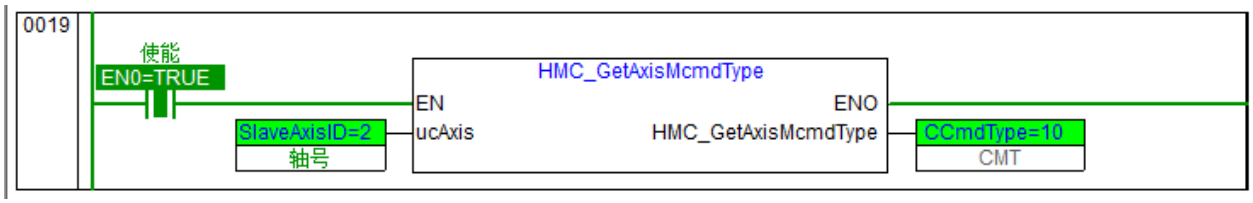
轴参数	类型	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
Basic									
AxisID	USINT	0	1	2	3	4	5	6	7
VarName	STRING	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
AxisType	EmAxisType	Unused	Unused	EtherCAT_...	Unused	Unused	Unused	Unused	Unused
Units	LREAL	1	1	1	1	1	1	1	1
Status									
MCmdType	EmCmdType	HMC__McIdle	HMC__McIdle	HMC__Move	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle
NCmdType	EmCmdType	HMC__McIdle	HMC__McIdle	HMC__Forward	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle
AxisStatus	UDINT	0	0	0	0	0	0	0	0
AxisErrMask	UDINT	0	0	0	0	0	0	0	0
LimitState	USINT	0	0	0	0	0	0	0	0

变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAxisID		轴号	USINT	2	FALSE

0038	CCmdType			UDINT	10	FALSE
0039	NCmdType			UDINT	18	FALSE

LD 程序实现：



ST 程序实现：

```
78  (*获取指定轴当前指令类型*)
79  IF EN0 THEN
80
81      CCmdType := HMC_GetAxisMcmdType(SlvaAxisID);
82
83  END_IF
```

EN0 = TRUE

CCmdType = 10　　SlvaAxisID = 2

5.8.21　HMC_GetAxisNcmdType（获取轴参 NcmdType）

5.8.21.1　功能

获取指定轴号从站的下一个指令类型，输入为要获取轴参数的指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的下一个指令类型（轴参数 NcmdType），如果设置轴号超过最大取值范围（0~63）或当前轴指令为空，输出为 0 。

获取到的轴状态信息与轴参数 NcmdType（下一条指令类型）一致，具体内容参见轴参数章节描述。

5.8.21.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisNcmdType	Output	UDINT	轴参 NcmdType（下一条指令类型）值

5.8.21.3 指令类型

非轴运动缓存指令。

5.8.21.4 示例

以下示例通过 HMC_GetAxisNcmdType 指令获取轴号为 2 的从站下一条指令类型，当 EN 使能时，该指令启动获取从站下一条指令类型功能，并将获取到的指令类型结果返回到 NCmdType 变量，执行后获取到的下一条指令类型为 18:HMC_FORWARD，与实际组态的逻辑配置一致。

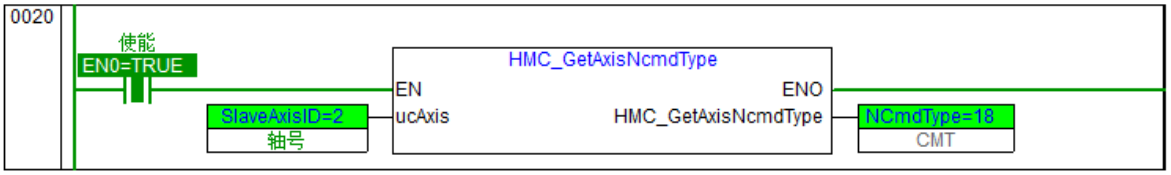
轴参数	类型	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
Basic									
AxisID	USINT	0	1	2	3	4	5	6	7
VarName	STRING	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
AxisType	EmAxisType	Unused	Unused	EtherCAT_...	Unused	Unused	Unused	Unused	Unused
Units	LREAL	1	1	1	1	1	1	1	1
Status									
MCmdType	EmCmdType	HMC__McIdle	HMC__McIdle	HMC__Move	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle
NCmdType	EmCmdType	HMC__McIdle	HMC__McIdle	HMC__Forward	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle	HMC__McIdle
AxisStatus	UDINT	0	0	0	0	0	0	0	0
AxisErrMask	UDINT	0	0	0	0	0	0	0	0
LimitState	USINT	0	0	0	0	0	0	0	0

变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAxisID		轴号	USINT	2	FALSE

0038	CCmdType			UDINT	10	FALSE
0039	NCmdType			UDINT	18	FALSE

LD 程序实现：



ST 程序实现：

```
84
85   (*获取指定轴下一条指令类型*)
86   IF EN0 THEN
87
88       NCmdType := HMC_GetAxisNcmdType(SlvaAxisID);
89
90   END_IF
```

EN0 = TRUE

NCmdType = 18　　SlvaAxisID = 2

5.8.22　HMC_GetAxisType（获取轴参 AxisType）

5.8.22.1　功能

获取指定轴号从站的轴参数 AxisType（轴类型），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的轴类型，如果设置轴号超过最大取值范围（0~63），输出为 0 。

获取的轴状态信息与轴参数 AxisType（轴类型）一致，具体内容参见轴参数章节描述。

5.8.22.2　参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisType	Output	UDINT	轴参 AxisType（轴类型）值

5.8.22.3　指令类型

非轴运动缓存指令。

5.8.22.4　示例

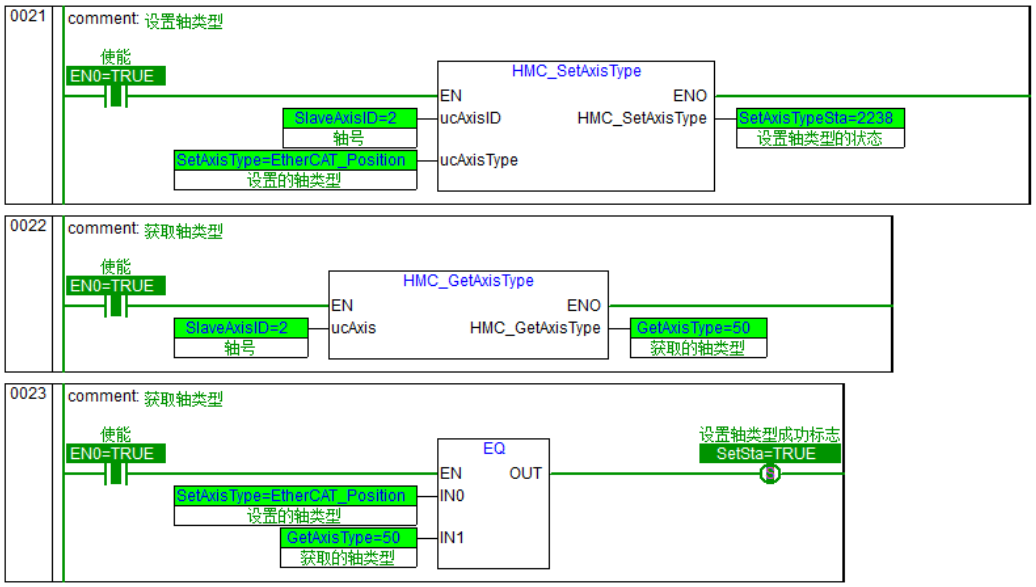
以下示例通过 HMC_GetAxisType 指令获取轴号为 2 的轴类型，当 EN 使能时，该指令启动获取从站轴类型功能，并将获取到的轴类型结果返回到 AxisType 变量，执行后获取到的下一条指令类型为 50:

EtherCAT_Position，与实际组态的逻辑配置一致。

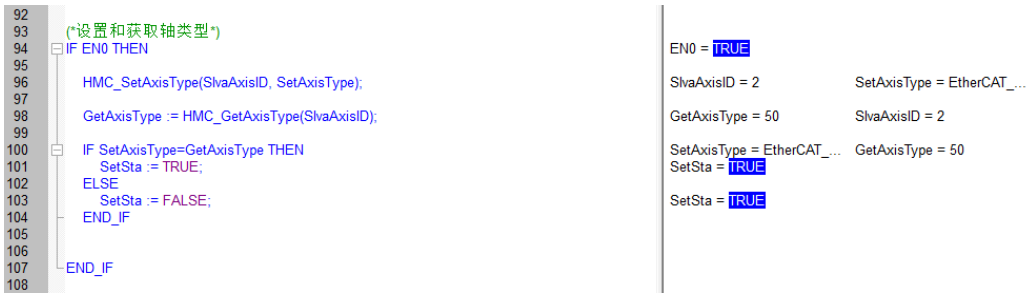
变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAxisID		轴号	USINT	2	FALSE
0040	SetAxisType		设置的轴类型	EMAXISTYPE	EtherCAT_...	FALSE
0041	GetAxisType		获取的轴类型	DINT	50	FALSE
0042	SetAxisTypeSta		设置轴类型的状态	UDINT	2238	FALSE
0043	SetSta		设置轴类型成功标志	BOOL	TRUE	FALSE

LD 程序实现：



ST 程序实现：



5.8.23 HMC_GetAxisLimitState（获取轴参 LimitState）

5.8.23.1 功能

获取指定轴号从站的轴参数 LimitState（超限状态），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的超限状态，如果设置轴号超过最大取值范围（0~63），输出为 0 。

获取的轴状态信息与轴参数 LimitState（超限状态）一致，具体内容参见轴参数章节描述。

5.8.23.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisLimitState	Output	UDINT	轴参 LimitState（超限状态）值

5.8.23.3 指令类型

非轴运动缓存指令。

5.8.23.4 示例

以下示例通过 HMC_GetAxisLimitState 指令获取轴号为 2 的从站的限位状态，在获取之前，需要先指定正向硬限位通道，示例使用 HMC_HwLimitConfig 指令设置轴正向硬限位通道号为 0，当轴正向运动到达正向硬限位点时，HMC_GetAxisLimitState 指令获取到的限位状态变量 LimitState 为 1，此时报警 DO 输出并将电机反转 10000 个脉冲。

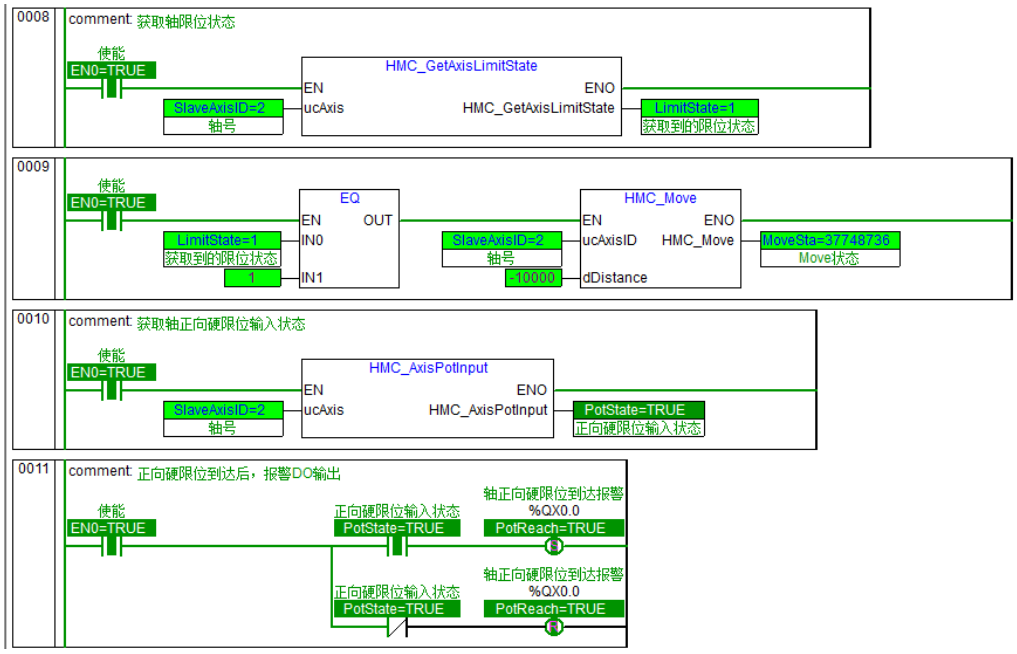
变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAxisID		轴号	USINT	2	FALSE

0026	PotState		正向硬限位输入状态	BOOL	FALSE	FALSE
0027	NotState		负向硬限位输入状态	BOOL	FALSE	FALSE
0028	PotReach	%QX0.0	轴正向硬限位到达报警	BOOL	FALSE	FALSE
0029	NotReach	%QX0.1	轴负向硬限位到达报警	BOOL	FALSE	FALSE

0044	LimitState		获取到的限位状态	USINT	1	FALSE
0045	MoveSta		Move状态	UDINT	37748736	FALSE

LD 程序实现：



ST 程序实现：

```
10
11  (*设置正向硬限位通道*)                                  EN0 = TRUE
12  IF EN0 THEN                                             SlvaAxisID = 2
13      HMC_HwLimitConfig(SlvaAxisID,0,0);
14  END_IF
15
16  (*获取轴正向硬限位输入状态，状态为TRUE后，报警DO输出*)
17  IF EN0 THEN                                             EN0 = TRUE
18      LimitState := HMC_GetAxisLimitState(SlvaAxisID);(*获取限位状态*)   LimitState = 1   SlvaAxisID = 2
19      IF LimitState=1 THEN                                LimitState = 1
20          MoveSta := HMC_Move(SlvaAxisID,-10000);         MoveSta = 33554433   SlvaAxisID = 2
21      END_IF
22
23      PotState := HMC_AxisPotInput(SlvaAxisID);           PotState = TRUE   SlvaAxisID = 2
24      IF PotState THEN                                    PotState = TRUE
25          PotReach := TRUE;                               PotReach = TRUE
26      ELSE
27          PotReach := FALSE;;                             PotReach = TRUE
28      END_IF
29  END_IF
```

5.8.24 HMC_GetAxisUnits（获取轴参 Units）

5.8.24.1 功能

获取指定轴号从站的轴参数 Units（单位转换因子），输入为指定轴号，轴号在设置时必须和实际组态

的轴号一致，输出为对应轴的单位转换因子，如果设置轴号超过最大取值范围（0~63），输出为 0 。

获取的轴状态信息与轴参数 Units（单位转换因子）一致，具体内容参见轴参数章节描述。

5.8.24.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisUnits	Output	UDINT	轴参 Units（单位转换因子）值

5.8.24.3 指令类型

非轴运动缓存指令。

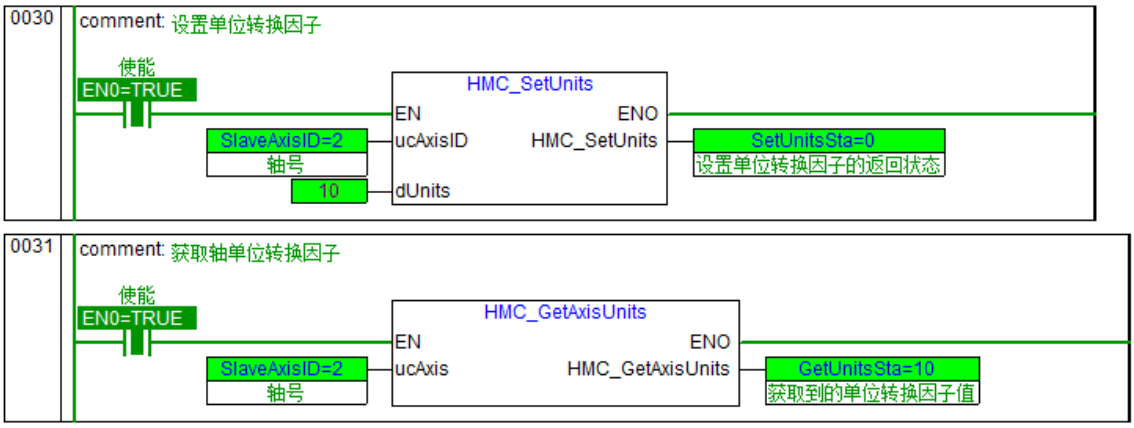
5.8.24.4 示例

以下示例通过 HMC_GetAxisUnits 指令获取轴号为 2 的从站的单位转换因子，当 EN 使能时，该指令启动获取从站已设置的用户单位转换因子，并将获取到的从站地址结果返回到 SlaveUnits 变量，获取到的转换因子与实际逻辑组态写入的一致。

变量定义：

0020	ENO		使能	BOOL	TRUE	FALSE
0021	SlaveAxisID		轴号	USINT	2	FALSE
0046	SetUnitsSta		设置单位转换因子的返回状态	UDINT	0	FALSE
0047	GetUnitsSta		获取到的单位转换因子值	LREAL	10	FALSE

LD 程序实现：



ST 程序实现：

```
114
115     (*单位转换因子*)
116
117   IF EN0 THEN
118       SetUnitsSta := HMC_SetUnits(SlvaAxisID,10);
119       GetUnitsSta := HMC_GetAxisUnits(SlvaAxisID);
120   END_IF
121
```

EN0 = TRUE
SetUnitsSta = 0　　SlvaAxisID = 2
GetUnitsSta = 10　　SlvaAxisID = 2

5.8.25 HMC_SetAxisSpeed (**设定轴参** Speed)

5.8.25.1 **功能**

设定指定轴号从站的轴参数 Speed（目标速度），输入为指定轴号和需要设定的目标速度，轴号在设置时必须和实际组态的轴号一致，设定的目标速度是负值时，会取其绝对值赋值给轴参 Speed，设置成功返回 0，如果设置轴号超过最大取值范围（0~63），返回为 0xFFFFFFFF。

具体内容参见轴参数 Speed（目标速度）章节描述。

5.8.25.2 **参数**

参数	**声明**	**数据类型**	**说明**
ucAxisID	Input	USINT	轴号
dSpeed	Input	LREAL	设定的目标速度
HMC_SetAxisSpeed	Output	UDINT	0：设置成功 0xFFFFFFFF：失败

5.8.25.3 指令类型

非轴运动缓存指令。

5.8.25.4 示例

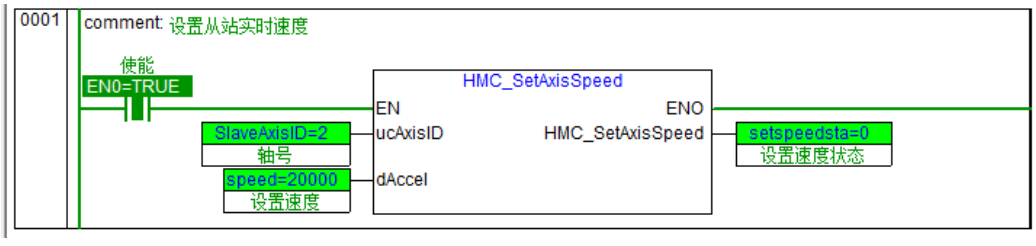
以下示例通过 HMC_SetAxisSpeed 指令设置轴号为 2 的从站的目标速度，当 EN 使能时，该指令启动设置从站目标速度功能，设置的速度实时生效，读取伺服电机实际转速，与设置值一致。

轴参数	类型	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	0	0	3638165.0...	0	0	0	0	0
Remain	LREAL	0	0	1180	0	0	0	0	0
Velocity Profile									
Speed	LREAL	10000	2000	20000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	200000	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	200000	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	18000	0	0	0	0	0
VpSpeed	LREAL	0	0	20000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	5000000000	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

变量定义：

0001	speed		设置速度	LREAL	20000	FALSE
0002	setspeedsta		设置速度状态	UDINT	0	FALSE
0003	ENO		使能	BOOL	TRUE	FALSE
0004	SlaveAxisID		轴号	USINT	2	FALSE

LD 程序实现：



ST 程序实现：

```
1   (*设置实际速度*)
2
3   IF EN0 THEN
4       setspeedsta := HMC_SetAxisSpeed(AxisNum,speed);
5   END_IF
```

EN0 = TRUE
setspeedsta = 0　　AxisNum = 2　　speed = 20000

5.8.26 HMC_SetAxisAccel（设定轴参 Accel）

5.8.26.1 功能

设定指定轴号从站的轴参数 Accel（加速度），输入为指定轴号和需要设定的加速度，轴号在设置时必须和实际组态的轴号一致，设定的加速度是负值时，会取其绝对值赋值给轴参 Accel，设置成功返回 0，如果设置轴号超过最大取值范围（0~63），返回为 0xFFFFFFFF。

具体内容参见轴参数 Accel（加速度）章节描述。

5.8.26.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dAccel	Input	LREAL	设定的加速度
HMC_SetAxisAccel	Output	UDINT	0：设置成功 0xFFFFFFFF：失败

5.8.26.3 指令类型

非轴运动缓存指令。

5.8.26.4 示例

以下示例通过 HMC_SetAxisAccel 指令设置轴号为 2 的从站加速度，当 EN 使能时，该指令启动设置从站加速度的功能，设置成功后，监视轴参数值与实际组态逻辑设置值一致。

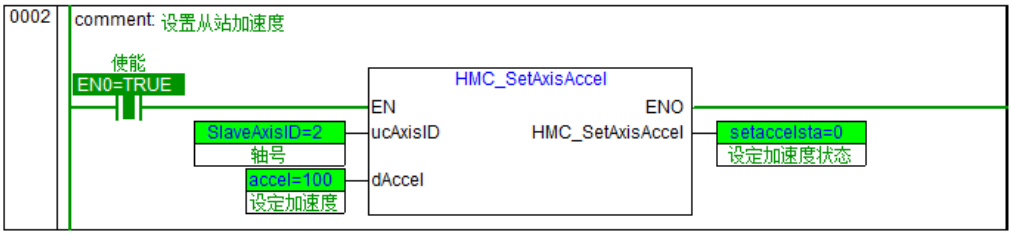
轴参数	类型	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	???	???	???	???	???	???	???	???
Remain	LREAL	???	???	???	???	???	???	???	???
Velocity Profile									
Speed	LREAL	10000	2000	3000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	2000	0	0	0	0	0
VpSpeed	LREAL	0	0	3000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	100	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

变量定义：

0003	ENO		使能	BOOL	TRUE	FALSE
0004	SlaveAxisID		轴号	USINT	0	FALSE

0005	accel		设定加速度	LREAL	0	FALSE
0006	setaccelsta		设定加速度状态	UDINT	0	FALSE

LD 程序实现：



ST 程序实现：

```
(*设置加速度*)

IF EN0 THEN
    setaccelsta := HMC_SetAxisAccel(AxisNum,accel);
END_IF
```

EN0 = TRUE
setaccelsta = 0　　AxisNum = 2　　accel = 100

5.8.27　HMC_SetAxisDecel（设定轴参 Decel）

5.8.27.1　功能

设定指定轴号从站的轴参数 Decel（减速度），输入为指定轴号和需要设定的减速度，轴号在设置时必须和实际组态的轴号一致，设定的减速度是负值时，会取其绝对值赋值给轴参 Decel，设置成功返回 0，如

果设置轴号超过最大取值范围（0~63），返回为 0xFFFFFFFF。

具体内容参见轴参数 Decel（减速度）章节描述。

5.8.27.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dDecel	Input	LREAL	设定的减速度
HMC_SetAxisDecel	Output	UDINT	0：设置成功 0xFFFFFFFF：失败

5.8.27.3 指令类型

非轴运动缓存指令。

5.8.27.4 示例

以下示例通过 HMC_SetAxisDecel 指令设置轴号为 2 的从站减速度，当 EN 使能时，该指令启动设置从站减速度的功能，设置成功后，监视轴参数值与实际组态逻辑设置值一致。

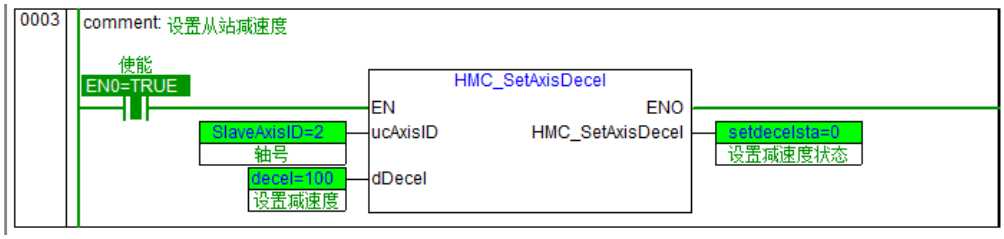
轴参数	类型	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	???	???	???	???	???	???	???	???
Remain	LREAL	???	???	???	???	???	???	???	???
Velocity Profile									
Speed	LREAL	10000	2000	3000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	2000	0	0	0	0	0
VpSpeed	LREAL	0	0	3000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	100	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

变量定义：

0003	ENO		使能	BOOL	TRUE	FALSE
0004	SlaveAxisID		轴号	USINT	0	FALSE

0007	decel		设置减速度	LREAL	0	FALSE
0008	setdecelsta		设置减速度状态	UDINT	0	FALSE

LD 程序实现：



ST 程序实现：

```
13  (*设置减速度*)
14
15  IF EN0 THEN
16      setdecelsta := HMC_SetAxisDecel(AxisNum,decel);
17  END_IF
18
```

EN0 = TRUE
setdecelsta = 0　　AxisNum = 2　　decel = 100

5.8.28 HMC_SetAxisJerk（设定轴参 Jerk）

5.8.28.1 功能

设定指定轴号从站的轴参数 Jerk（加加速度），输入为指定轴号和需要设定的加加速度，轴号在设置时必须和实际组态的轴号一致，设定的加加速度是负值时，会取其绝对值赋值给轴参 Jerk，设置成功返回 0，如果设置轴号超过最大取值范围（0~63），返回为 0xFFFFFFFF。

具体内容参见轴参数 Jerk（加加速度）章节描述。

5.8.28.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dJerk	Input	LREAL	设定的加加速度
HMC_SetAxisJerk	Output	UDINT	0：设置成功 0xFFFFFFFF：失败

5.8.28.3 指令类型

非轴运动缓存指令。

5.8.28.4 示例

以下示例通过 HMC_SetAxisJerk 指令设置轴号为 2 的从站加加速度，当 EN 使能时，该指令启动设置从站加加速度的功能，设置成功后，监视轴参数值与实际组态逻辑设置值一致。

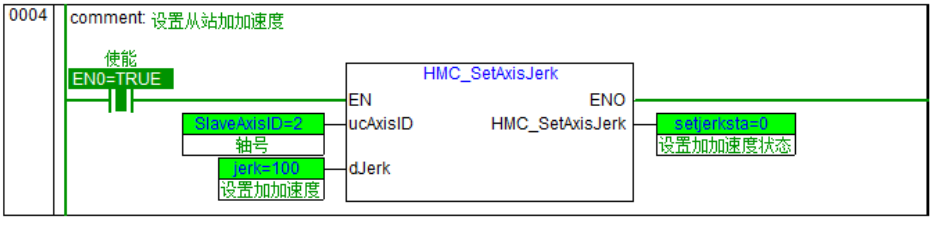
轴参数	类型	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
EndPos	LREAL	???	???	???	???	???	???	???	???
Remain	LREAL	???	???	???	???	???	???	???	???
Velocity Profile									
Speed	LREAL	10000	2000	3000	2000	2000	2000	2000	2000
Accel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
Decel	LREAL	200000	200000	100	200000	200000	200000	200000	200000
MpSpeed	LREAL	0	0	2000	0	0	0	0	0
VpSpeed	LREAL	0	0	3000	0	0	0	0	0
Direction	SINT	1	1	1	1	1	1	1	1
Merge	USINT	0	0	0	0	0	0	0	0
CreepSpeed	LREAL	200	200	200	200	200	200	200	200
FastDecel	LREAL	5000000	5000000	5000000	5000000	5000000	5000000	5000000	5000000
Jerk	LREAL	5000000000	5000000000	100	5000000000	5000000000	5000000000	5000000000	5000000000
JerkMode	USINT	0	0	0	0	0	0	0	0

变量定义：

0003	ENO		使能	BOOL	TRUE	FALSE
0004	SlaveAxisID		轴号	USINT	0	FALSE

0009	jerk		设置加加速度	LREAL	100	FALSE
0010	setjerksta		设置加加速度状态	UDINT	0	FALSE

LD 程序实现：



ST 程序实现：

```
18
19    (*设置加加速度*)
20
21  IF EN0 THEN
22      setjerksta := HMC_SetAxisJerk(AxisNum,jerk);
23  END_IF
```

EN0 = TRUE
setjerksta = 0　　AxisNum = 2　　jerk = 100

5.8.29 HMC_SetAxisDAC（设定轴参 DAC）

5.8.29.1 功能

设定指定轴号从站的轴参数 DAC（速度模式开环输出值），输入为指定轴号和需要设定的速度模式开环输出值，轴号在设置时必须和实际组态的轴号一致，设置成功返回 0，如果设置轴号超过最大取值范围（0~63），返回为 0xFFFFFFFF。

具体内容参见轴参数 DAC（速度模式开环输出值）章节描述。

5.8.29.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
iAxisDAC	Input	DINT	设定的速度模式开环输出值
HMC_SetAxisDAC	Output	UDINT	0：设置成功 0xFFFFFFFF：失败

5.8.29.3 指令类型

非轴运动缓存指令。

5.8.29.4 示例

以下示例通过 HMC_SetAxisDAC 指令设置轴号为 2 的从站速度开环输出值，当 EN 使能时，该指令启动设置从站的速度开环输出值功能，设置成功后，监视轴参数值与实际组态逻辑设置值一致。

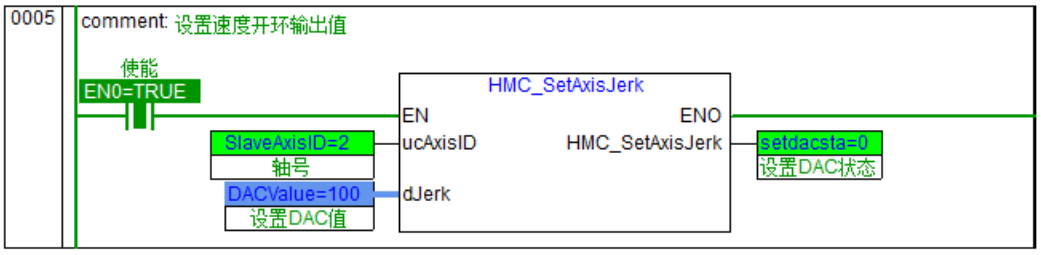
轴参数	类型	Axis0	Axis1	Axis2	Axis3	Axis4	Axis5	Axis6	Axis7
HoldIn	UDINT	65535	65535	65535	65535	65535	65535	65535	65535
Input									
KeNumerator	DINT	1	1	1	1	1	1	1	1
KeDenominator	DINT	1	1	1	1	1	1	1	1
EncoderValue	DINT	0	0	6051849	0	0	0	0	0
FeedBackSrcMode	USINT	0	0	0	0	0	0	0	0
FeedBackSrcValue	LREAL	0	0	0	0	0	0	0	0
FeedBackDataType	USINT	0	0	0	0	0	0	0	0
Output									
KsNumerator	DINT	1	1	1	1	1	1	1	1
KsDenominator	DINT	1	1	1	1	1	1	1	1
PulseNum	UDINT	0	0	3	0	0	0	0	0
ServoLoop	USINT	0	0	0	0	0	0	0	0
DAC	DINT	0	0	100	0	0	0	0	0
DACOut	DINT	0	0	0	0	0	0	0	0

变量定义：

0003	ENO		使能	BOOL	TRUE	FALSE
0004	SlaveAxisID		轴号	USINT	0	FALSE

0011	DACValue		设置DAC值	DINT	100	FALSE
0012	setdacsta		设置DAC状态	UDINT	0	FALSE

LD 程序实现：



ST 程序实现：

```
24
25   (*设置速度开环输出值*)
26
27   IF EN0 THEN
28       setdacsta := HMC_SetAxisDAC(AxisNum,DACValue);
29   END_IF
```

EN0 = TRUE　setdacsta = 0　AxisNum = 2　DACValue = 100

5.8.30 HMC_SetAxisCreepSpeed（设定轴参 CreepSpeed）

5.8.30.1 功能

设定指定轴号从站的轴参数 CreepSpeed（原点回归爬行速度），输入为指定轴号和需要设定的原点回

归爬行速度，轴号在设置时必须和实际组态的轴号一致，设定原点回归爬行速度是负值时，会取其绝对值赋值给轴参 CreepSpeed，设置成功返回 0，如果设置轴号超过最大取值范围（0~63），返回为 0xFFFFFFFF。

具体内容参见轴参数 CreepSpeed（原点回归爬行速度）章节描述。

5.8.30.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
dCreepSpeed	Input	LREAL	设定的原点回归爬行速度值
HMC_SetAxisDAC	Output	UDINT	0：设置成功 0xFFFFFFFF：失败

5.8.30.3 指令类型

非轴运动缓存指令。

5.8.30.4 示例

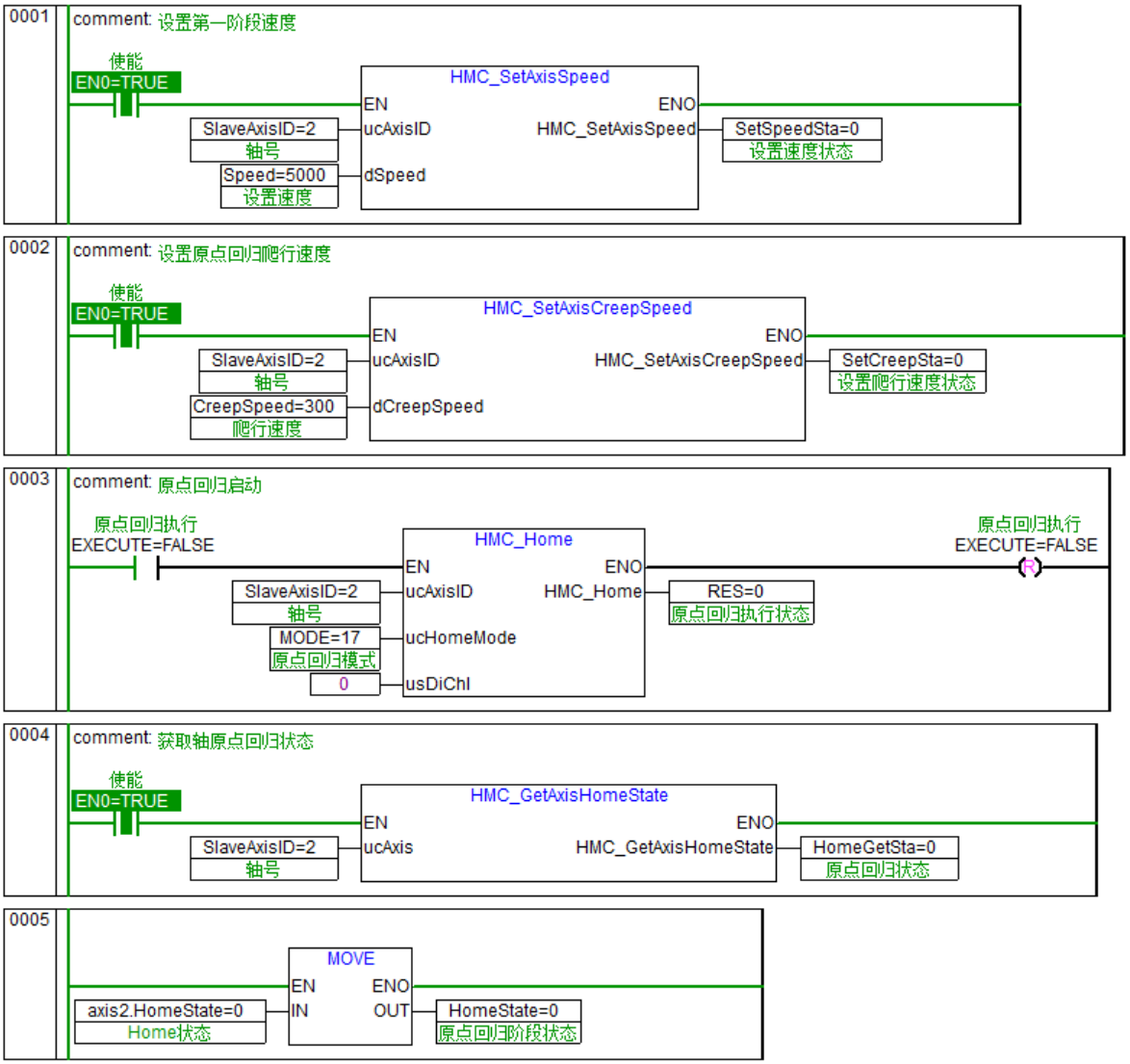
以下示例对轴 2 的伺服系统进行原点回归操作，原点回归模式设置为 0，DI 通道设置为 0 通道，通过 HMC_SetAxisCreepSpeed 指令设置原点回归第三阶段的爬行速度，设置完成后调用 HMC_Home 指令启动原点回归，原点回归执行期间使用 HMC_GetAxisHomeState 指令获取原点回归状态。

模式 17 原点回归过程：原点回归启动后轴以 Speed 设定的速度正向运动（阶段 1），当遇到 DI 上升沿时减速停止（阶段 2），然后以 CreepSpeed 设定的爬行速度反向运动（阶段 3），遇到 DI 下降沿时，设定此处为原点位置并开始减速停止，然后再返回到原点位置（阶段 4），最终完成原点回归过程（阶段 6）。

变量定义：

序号△	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	MODE		原点回归模式	BYTE	17	FALSE
0002	RES		原点回归执行状态	UDINT	0	FALSE
0003	EXECUTE		原点回归执行	BOOL	FALSE	FALSE
0004	HomeState		原点回归阶段状态	USINT	0	FALSE
0005	ENO		使能	BOOL	TRUE	FALSE
0006	SlaveAxisID		轴号	USINT	2	FALSE
0007	SetCreepSta		设置爬行速度状态	UDINT	0	FALSE
0008	CreepSpeed		爬行速度	LREAL	300	FALSE
0009	HomeGetSta		原点回归状态	USINT	0	FALSE
0010	done			BOOL	FALSE	FALSE
0011	targetspeed		目标速度	LREAL	0	FALSE
0012	Speed		设置速度	LREAL	5000	FALSE

LD 程序实现：



ST 程序实现：

```
(*原点回归*)
IF EN0 THEN
    HMC_SetAxisSpeed(SlaveAxisID, Speed);
    HMC_SetAxisCreepSpeed(SlaveAxisID, CreepSpeed); (*设置爬行速度*)

    IF EXECUTE THEN
        HMC_Home(SlaveAxisID, MODE, 0); (*原点回归执行*)
        EXECUTE := FALSE;
    END_IF

    HomeGetSta := HMC_GetAxisHomeState(SlaveAxisID); (*获取原点回归状态*)

END_IF
```

```
EN0 = TRUE
SlaveAxisID = 2          Speed = 5000
SlaveAxisID = 2          CreepSpeed = 300

EXECUTE = FALSE
SlaveAxisID = 2          MODE = 17
EXECUTE = FALSE

HomeGetSta = 0           SlaveAxisID = 2
```

5.8.31 HMC_GetAxisHomeState(**获取轴参** HomeState)

5.8.31.1 **功能**

获取指定轴号从站的轴参数 HomeState（原点回归状态），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的原点回归状态，如果设置轴号超过最大取值范围（0~63），输出为 0 。

获取的轴状态信息与轴参数 HomeState（原点回归状态）一致，具体内容参见轴参数章节描述。

5.8.31.2 **参数**

参数	**声明**	**数据类型**	**说明**
ucAxisID	Input	USINT	轴号
HMC_GetAxisHomeState	Output	USINT	原点回归状态值，具体参见轴参数 HomeState（原点回归状态）

5.8.31.3 **指令类型**

非轴运动缓存指令。

5.8.31.4 **示例**

参见 HMC_SetAxisCreepSpeed（设定轴参 CreepSpeed） 示例。

5.8.32 HMC_GetAxisHardLimitCfg (**获取轴限位配置**)

5.8.32.1 **功能**

获取指定从站的轴硬限位配置数据，输入为指定从站的轴号和前后限位的选择标志，轴号在设置时必须和实际组态的轴号一致，前后限位的选择标志即选择当前要获取的是前限位的配置信息还是后限位的配置信息，输出为获取到的硬限位配置数据，包括限位通道号和限位逻辑值。获取成功返回 0，失败返回-1。

5.8.32.2 **参数**

参数	**声明**	**数据类型**	**说明**
ucAxisID	Input	USINT	轴号
ucSelect	Input	USINT	设定的前后限位标志 0：前限位，1：后限位
pHardLimitChl	Output	POINTER TO UDINT	限位通道号
pHardLimitLogic	Output	POINTER TO USINT	限位逻辑值 0：常开，1：常闭
HMC_GetAxisHardLimitCfg	Output	UDINT	0：设置成功 -1：设置错误

5.8.32.3 **指令类型**

非轴运动缓存指令。

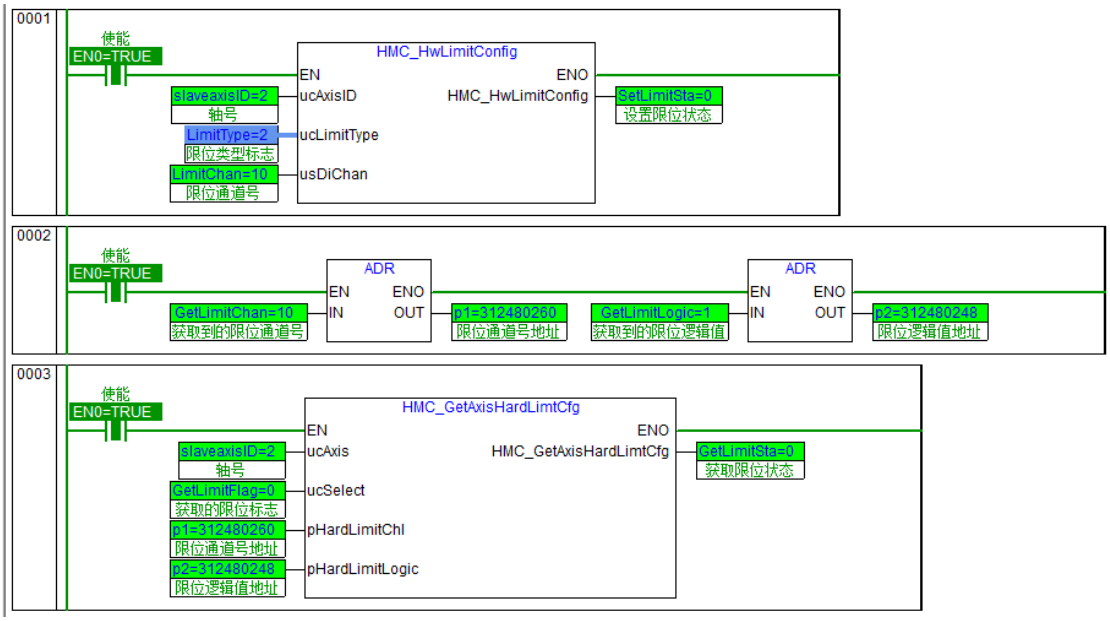
5.8.32.4 **示例**

以下示例通过 HMC_GetAxisHardLimitCfg 指令获取轴号为 2 的从站的硬限位配置数据，EN0 使能之后，该指令启动获取从站硬限位配置数据的功能。

变量定义：

序号	变量名	直接地址	变量说明	变量类型	在线值	掉电保护
0001	ENO		使能	BOOL	TRUE	FALSE
0002	slaveaxisID		轴号	USINT	2	FALSE
0003	LimitType		限位类型标志	USINT	2	FALSE
0004	LimitChan		限位通道号	UDINT	10	FALSE
0005	SetLimitSta		设置限位状态	UDINT	0	FALSE
0006	GetLimitChan		获取到的限位通道号	UDINT	10	FALSE
0007	GetLimitLogic		获取到的限位逻辑值	USINT	1	FALSE
0008	p1		限位通道号地址	POINTER TO UDINT	312480260	FALSE
0009	p2		限位逻辑值地址	POINTER TO SINT	312480248	FALSE
0010	GetLimitSta		获取限位状态	UDINT	0	FALSE
0011	GetLimitFlag		获取的限位标志	USINT	0	FALSE

LD 程序实现：



ST 程序实现：

```
IF EN0 THEN

    SetLimitSta := HMC_HwLimitConfig(slaveaxisID,LimitType,LimitChan); (*配置硬限位*)
    GetLimitSta := HMC_GetAxisHardLimtCfg(slaveaxisID,GetLimitFlag,ADR(GetLimitChan),ADR(GetLimitLogic)); (*获取硬限位配置信息*)

END_IF
```

5.8.33 HMC_GetAxisVpSpeed (**获取轴参** VpSpeed)

5.8.33.1 **功能**

获取指定轴号从站的轴参数 VpSpeed（解算速度），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的运动指令实时解算出来的速度值，单位为用户单位/秒，如果设置轴号超过最

大取值范围（0~63），输出为-1 。

获取的轴状态信息与轴参数 VpSpeed（解算速度）一致，具体内容参见轴参数章节描述。

5.8.33.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisMpSpeed	Output	LREAL	轴参数 VpSpeed（解算速度）值

5.8.33.3 指令类型

非轴运动缓存指令。

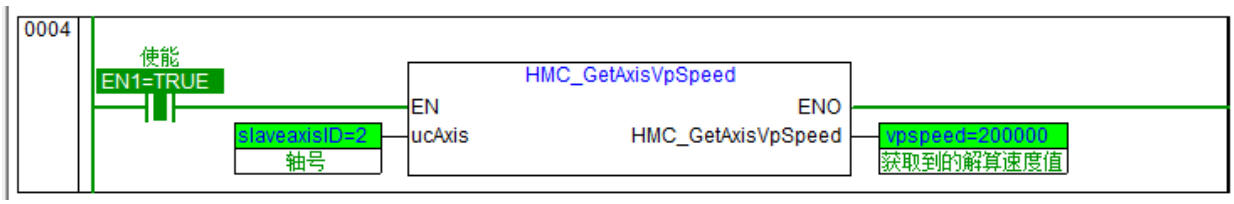
5.8.33.4 示例

以下示例通过 HMC_ GetAxisVpSpeed 指令获取轴号为 2 的从站的实时目标解算速度。

变量定义：

0004	EN1		使能	BOOL	TRUE	FALSE
0005	vpspeed		获取到的解算速度值	LREAL	200000	FALSE
0006	mpspeed		获取到的反馈速度值	LREAL	202000	FALSE

LD 程序实现：



ST 程序实现：

```
vpspeed := HMC_GetAxisVpSpeed(slaveaxisID);
```

vpspeed = 200000　　slaveaxisID = 2

5.8.34 HMC_GetAxisMpSpeed (**获取轴参** MpSpeed)

5.8.34.1 **功能**

获取指定轴号从站的轴参数 MpSpeed（反馈速度），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴的实时测量到的反馈速度，单位为用户单位/秒，如果设置轴号超过最大取值范围（0~63），输出为-1 。

获取的轴状态信息与轴参数 MpSpeed（反馈速度）一致，具体内容参见轴参数章节描述。

5.8.34.2 **参数**

参数	**声明**	**数据类型**	**说明**
ucAxisID	Input	BYTE	轴号
HMC_GetAxisDpos	Output	LREAL	轴参数 MpSpeed（反馈速度）值

5.8.34.3 **指令类型**

非轴运动缓存指令。

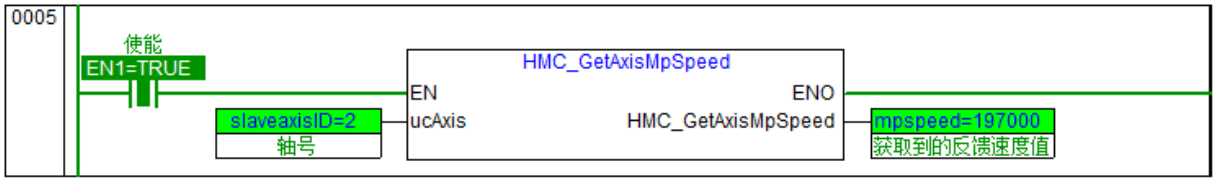
5.8.34.4 **示例**

以下示例通过 HMC_ GetAxisMpSpeed 指令获取轴号为 2 的从站的实时测量的反馈速度。

变量定义：

0004	EN1		使能	BOOL	TRUE	FALSE
0005	vpspeed		获取到的解算速度值	LREAL	200000	FALSE
0006	mpspeed		获取到的反馈速度值	LREAL	202000	FALSE

LD 程序实现：



ST 程序实现：

```
mpspeed := HMC_GetAxisMpSpeed(slaveaxisID);
```

mpspeed = 204000　　slaveaxisID = 2

5.8.35　HMC_GetAxisDpos (获取轴参 Dpos)

5.8.35.1　功能

获取指定轴号从站的轴参数 Dpos（本周期目标位置），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴解算出来的本周期目标位置，单位为用户单位，如果设置轴号超过最大取值范围（0~63），输出为-1 。

获取的轴状态信息与轴参数 Dpos（本周期目标位置）一致，具体内容参见轴参数章节描述。

5.8.35.2　参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisDpos	Output	LREAL	轴参数 Dpos（本周期目标位置）值

5.8.35.3　指令类型

非轴运动缓存指令。

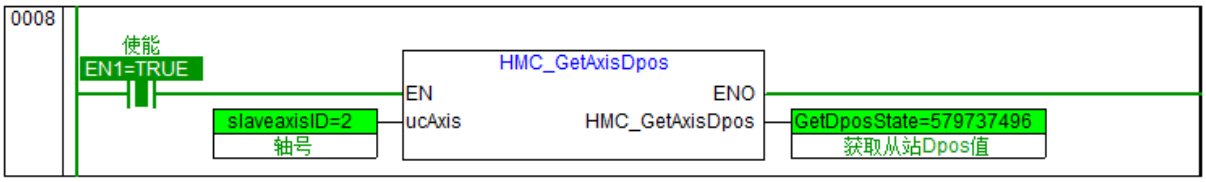
5.8.35.4　示例

以下示例通过 HMC_GetAxisDpos 指令获取轴号为 2 的从站的本周期目标位置。

变量定义：

0003	slaveaxisID		轴号	USINT	2	FALSE
0004	EN1		使能	BOOL	TRUE	FALSE
0028	GetDposState		获取从站Dpos值	LREAL	579737496	FALSE

LD 程序实现：



ST 程序实现：

```
14  GetDposState := HMC_GetAxisDpos(slaveaxisID);
```

GetDposState = 579737496 slaveaxisID = 2

5.8.36 HMC_GetAxisMpos (获取轴参 Mpos)

5.8.36.1 功能

获取指定轴号从站的轴参数 Mpos（反馈位置），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴实时反馈的位置，单位为用户单位，如果设置轴号超过最大取值范围（0~63），输出为-1 。

获取的轴状态信息与轴参数 Mpos（反馈位置）一致，具体内容参见轴参数章节描述。

5.8.36.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisMpos	Output	LREAL	轴参数 Mpos（反馈位置）值

5.8.36.3 指令类型

非轴运动缓存指令。

5.8.36.4 示例

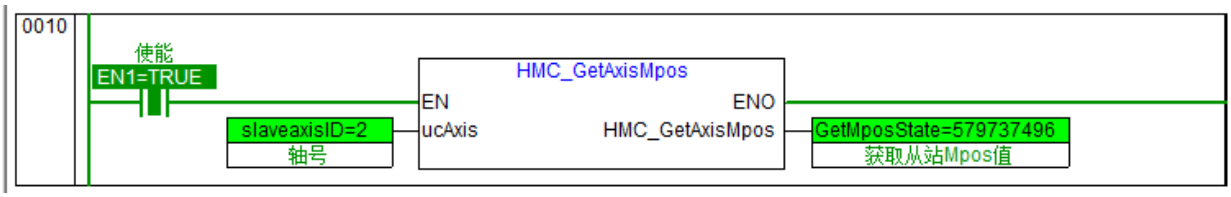
以下示例通过 HMC_GetAxisMpos 指令获取轴号为 2 的从站的反馈位置。

变量定义：

0003	slaveaxisID		轴号	USINT	2	FALSE
0004	EN1		使能	BOOL	TRUE	FALSE

0029	GetMposState		获取从站Mpos值	LREAL	579737496	FALSE

LD 程序实现：



ST 程序实现：

```
15  GetMposState := HMC_GetAxisMpos(slaveaxisID);
```

GetMposState = 579737496 slaveaxisID = 2

5.8.37 HMC_GetAxisEndPos (获取轴参 EndPos)

5.8.37.1 功能

获取指定轴号从站的轴参数 EndPos（指令目标位置），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出当前运动指令的目标位置，单位为用户单位，如果设置轴号超过最大取值范围（0~63），输出为-1 。

获取的轴状态信息与轴参数 EndPos（指令目标位置）一致，具体内容参见轴参数章节描述。

5.8.37.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisEndPos	Output	LREAL	轴参数 EndPos（指令目标位置）） 值

5.8.37.3 指令类型

非轴运动缓存指令。

5.8.37.4 示例

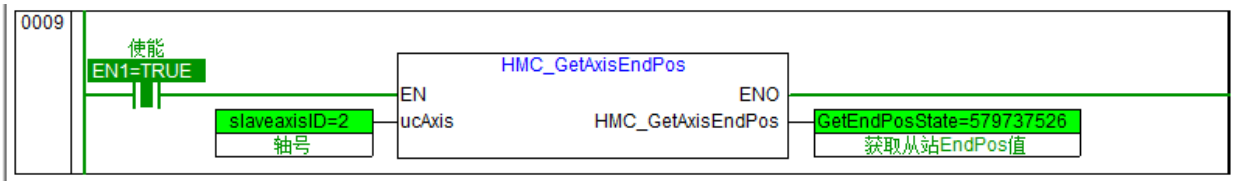
以下示例通过 HMC_GetAxisEndPos 指令获取轴号为 2 的从站的当前运动指令目标位置。

变量定义：

0003	slaveaxisID		轴号	USINT	2	FALSE
0004	EN1		使能	BOOL	TRUE	FALSE

0030	GetEndPosState		获取从站EndPos值	LREAL	579737526	FALSE

LD 程序实现：



ST 程序实现：

```
16  GetEndPosState := HMC_GetAxisEndPos(slaveaxisID);
```

GetEndPosState = 579737526 slaveaxisID = 2

5.8.38 HMC_GetAxisRemain (获取轴参 Remain)

5.8.38.1 功能

获取指定轴号从站的轴参数 Remain（剩余位置），输入为指定轴号，轴号在设置时必须和实际组态的轴号一致，输出为对应轴离指令目标位置的剩余位移，单位为用户单位，如果设置轴号超过最大取值范围（0~63），输出为-1 。

获取的轴状态信息与轴参数 Remain（剩余位置）一致，具体内容参见轴参数章节描述。

5.8.38.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	BYTE	轴号
HMC_GetAxisRemain	Output	LREAL	轴参数 Remain（剩余位置）值

5.8.38.3 指令类型

非轴运动缓存指令。

5.8.38.4 示例

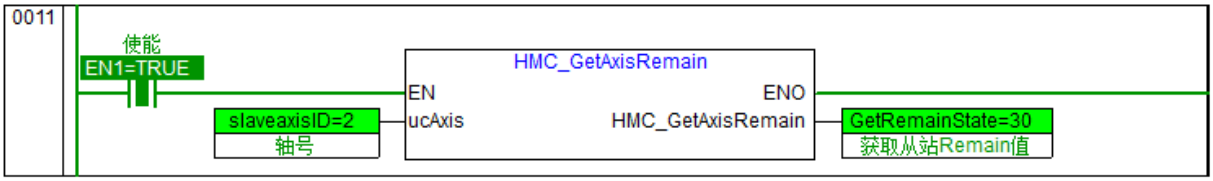
以下示例通过 HMC_GetAxisRemain 指令获取轴号为 2 的从站的剩余位置。

变量定义：

0003	slaveaxisID		轴号	USINT	2	FALSE
0004	EN1		使能	BOOL	TRUE	FALSE

0031	GetRemainState		获取从站Remain值	LREAL	30	FALSE

LD 程序实现：



ST 程序实现：

```
17  GetRemainState := HMC_GetAxisRemain(slaveaxisID);
```

GetRemainState = 30　　slaveaxisID = 2

5.9 状态读取指令

5.9.1 HMC_GetSysErr（获取系统错误码）

5.9.1.1 功能说明

获取系统最近一次发生的系统错误，若未发生系统错误，则返回 0。

算法实时运行时，会对检测出的异常报错误，错误类型包括系统错误和用户错误。

正常运行时不会产生系统错误，除非算法所用数据被外部程序模块篡改，导致进入一些不可能进入的代码分支，从而产生系统错误。

系统错误是严重的错误类型，所以当产生系统错误时，会自动关闭 watchdog，且算法停止解算（“超时错误”除外），排查错误原因后，需要重启控制器才可恢复。

系统错误中的“超时错误”比较特殊，错误 ID 为 1025，该错误是指算法模块单次运行所耗时间超出当前伺服周期，当发生该系统错误时，只是产生报警，不会关闭 wachdog，算法仍会正常解算。

5.9.1.2 参数

参数	声明	数据类型	说明
HMC_GetSysErr	Output	UDINT	0：没有错误 其他值：当前发生系统错误

5.9.1.3 指令类型

非轴运动缓存指令。

5.9.2 HMC_GetUserErrNum（获取用户错误数）

5.9.2.1 功能

返回已发生的用户错误数。

5.9.2.2 参数

参数	声明	数据类型	说明
HMC_GetUserErrNum	Output	UDINT	已发生的用户错误数量

5.9.2.3 指令类型

非轴运动缓存指令。

5.9.2.4 使用注意事项

当历史未发生用户错误时，返回值为 0，若发生过用户错误，目前只持者返回 1，暂不支持累计错误数量。

5.9.3 HMC_GetUserErrID（获取用户错误码）

5.9.3.1 功能

返回用户错误码。

用户错误是因为参数设置非法导致，如循环模式时 RepDist 设置小于 0，采样功能块设置采样通道号超出系统支持的采样通道范围等，即可以看出，用户错误是因为用户设置参数错误人为导致。

发生用户错误时，系统不会有其它保护动作，如停止解算、关闭伺服使能等。

目前只能返回当前用户错误码，暂不支持按索引返回错误码。

5.9.3.2 参数

参数	声明	数据类型	说明
uiErrIndex	Input	UDINT	错误索引取值范围：0~（HMC_GetUserErrNum()-1）
HMC_GetUserErrID	Output	UDINT	-

5.9.3.3 指令类型

非轴运动缓存指令。

5.9.4 HMC_GetUserErrMes（获取用户错误码附加信息）

5.9.4.1 功能

返回用户错误码附加信息。通过附加信息结合固件版本其它信息，可以确定产生该用户错误的函数。

目前只能返回当前用户错误码附加信息，暂不支持按索引返回错误码附加信息。

5.9.4.2 参数

参数	声明	数据类型	说明
uiErrIndex	Input	UDINT	错误索引 取值范围：0~（HMC_GetUserErrNum()-1）
HMC_GetUserErrMes	Output	UDINT	用户错误码附加信息

5.9.4.3 指令类型

非轴运动缓存指令。

5.9.5 HMC_GetAllAxisErr（获取所有轴的总体异常状态）

5.9.5.1 功能

获得所有轴的总体异常状态，当 64 个轴中存在超过 1 个以上轴的轴状态 AxisStatus 与其掩码 AxisErrMask 进行逻辑与运算结果不为 0，则认为存在轴异常。即如下表达式非 0 时，认为所有轴总体异常，该函数返回-1（0xFFFFFFFF）：((Axis0.AxisStatus and Axis0.AxisErrMask) or (Axis1.AxisStatus and Axis1.AxisErrMask) or …… or (Axis63.AxisStatus and Axis63.AxisErrMask))。

5.9.5.2 参数

参数	声明	数据类型	说明
HMC_GetAllAxisErr	Output	UDINT	0：所有轴正常 其他值：轴异常

5.9.5.3 指令类型

非轴运动缓存指令。

5.9.6 HMC_GetCmdErr（获取轴指令错误）

5.9.6.1 功能

根据轴指令 ID，返回该轴指令错误。没有错误时返回 0，轴指令 ID 不存在时返回 0xffffffff，指令 ID 存在错误时，返回错误类型。

轴指令 ID 是投放运动控制指令时，该函数的返回值。

5.9.6.2 参数

参数	声明	数据类型	说明
uiAxisCmdID	Input	UDINT	轴指令 ID
HMC_GetCmdErr	Output	UDINT	0：没有错误 其他值：轴指令 ID 不在轴缓存中或指令 ID 非法

5.9.6.3 指令类型

非轴运动缓存指令。

5.9.7 HMC_GetCmdState（获取轴指令状态）

5.9.7.1 功能

根据轴指令 ID，返回该轴指令状态。包括以下状态：

- 0：等待运行
- 1：运行中
- 2：已经完成或无指令
- 3：正在投送
- 其它值：参数错误

5.9.7.2 参数

参数	声明	数据类型	说明
uiAxisCmdID	Input	UDINT	轴指令 ID
HMC_GetCmdState	Output	USINT	0：等待运行：已经被投放到运动缓存中但尚未开始执行； 1：运行中； 2：已经完成或无指令； 3：正在投送：正在向运动缓存投送； 其他值：错误码（0xFF 为用户参数错误）

5.9.7.3 指令类型

非轴运动缓存指令。

5.9.8 HMC_WaitCmdFinish（等待指令完成）

5.9.8.1 功能

等待直到“某运动指令运行完成后”才返回。在该指令未执行完成之前，HMC_WaitCmdFinish 会一直处于阻塞，不返回。

5.9.8.2 参数

参数	声明	数据类型	说明
uiAxisCmdID	Input	UDINT	轴指令 ID
HMC_WaitCmdLoaded	Output	USINT	0：该指令运行完成 其他值：错误码（0xFF 为轴指令 ID 非法）

5.9.8.3 指令类型

非轴运动缓存指令。

5.9.9 HMC_WaitCmdLoaded（等待指令装载）

5.9.9.1 功能

等待直到“某运动指令（uiAxisCmdID）装载后” 即指令刚开始执行才返回。在该指令装载之前，HMC_WaitCmdLoaded 会一直处于阻塞，不返回。指令装载指该指令成功投送至轴运动缓存中。

5.9.9.2 参数

参数	声明	数据类型	说明
uiAxisCmdID	Input	UDINT	轴指令 ID
HMC_WaitCmdLoaded	Output	USINT	0：该指令运行完成 其他值：错误码（0xFF 为轴指令 ID 非法）

5.9.9.3 指令类型

非轴运动缓存指令。

5.9.10 HMC_WaitAxisIdle（等待轴空闲）

5.9.10.1 功能

等待直到“某轴（ucAxisID）的运动指令队列为空”才返回，否则一直处于阻塞状态，不返回。

5.9.10.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_WaitAxisIdle	Output	USINT	0：该轴空闲 其他值：错误码（0x17 为轴号超范围）

5.9.10.3 指令类型

非轴运动缓存指令。

5.9.11 HMC_GetCmdBufferState（获取轴指令缓存状态）

5.9.11.1 功能

获取某轴（ucAxis）的指令缓存状态。判断当前轴缓存是否未满，并返回该轴缓存状态；主要是为了防止“程序阻塞不往下执行”的现象发生。

5.9.11.2 参数

参数	声明	数据类型	说明
ucAxis	Input	USINT	轴号
HMC_GetCmdBufferState	Output	UDINT	0：指令缓存未满，可以正常投放指令 1：指令缓存已满，若投放指令则会出现阻塞现象 2：指令缓存被锁定（当前有指令正在投放进该缓存中） 0xFFFFFFFF：函数参数错误

5.9.11.3 指令类型

非轴运动缓存指令。

5.10 故障清除指令

5.10.1 HMC_ClearAxisErr（消除轴错误）

5.10.1.1 功能

清除某个轴的错误，设置某个轴 AxisStatus=0。

清除错误前，需先解除错误才可清除成功，否则无法清除错误。

5.10.1.2 参数

参数	声明	数据类型	说明
ucAxisID	Input	USINT	轴号
HMC_ClearAxisErr	Output	UDINT	0：成功 其他值：错误码（0xFFFFFFFF 为轴号超范围）

5.10.1.3 指令类型

非轴运动缓存指令。

5.10.2 HMC_ClearAllErr（消除所有轴错误）

5.10.2.1 功能

清除所有轴的错误，设置所有轴的 AxisStatus=0。

清除错误前，需先解除错误才可清除成功，否则无法清除错误。

5.10.2.2 参数

参数	声明	数据类型	说明
HMC_ClearAllErr	Output	UDINT	0：成功

5.10.2.3 指令类型

非轴运动缓存指令。

北京和利时智能技术有限公司

Beijing HollySys Intelligent Technologies Co., Ltd..

地址：北京经济技术开发区地盛中路2号院

邮编：100176

电话：010-5898 1588

热线：400-811-1999

传真：010-5898 1558

和利时集团网址：http://www.hollysys.com/

FA业务官网：https://fa.hollysys.com/