MC系列运动控制器

指令手册

2020年09月 V1.7

版权声明

地址：宁波市鄞州区新材料国际创新中心 7 号楼 5 层

邮编：315000

商务电话：0574-8717 5756

产品咨询热线：0574-8717 5756

技术支持电话：4008-111-999

技术支持手机：136 1116 0213

传真：010-5898 1558

网址：http://www.hollysys.com/

Email：PLC@hollysys.com

新浪微博：http://weibo.com/hollysysplc

HollySys

目　录

第1章　关于本书

1.1 文档更新

版本	日期	说明
1.0	2016.08.03	创建
1.1	2016.12.05	新增 4.3.6.2 HMC_SplineCam（样条凸轮）和 4.5.1.1 HMC_StewartControl（Stewart 机构控制）
1.2	2017.06.15	新增 4.3.6.8HMC_MoveFlexLink（飞剪/旋切运动）、4.4.9 多轴同步控制、4.5.1 Stewart 6 自由度并联机构、4.5.25 自由度 SCARA 机械臂、6.7 EtherCAT Touch Probe 功能
1.3	2017.12.25	新增 3.6.6 kaff 参数、4.3.5.5 HMC_SetGearPhaseOffset（设置电子齿轮相位延迟）、4.3.5.6HMC_GetGearPhaseOffset（获取电子齿轮相位延迟）、新增章节、4.4.1.8HMC_SetAddAxis（设定和运动轴）、4.4.1.9HMC_GetAddAxis（获取和运动叠加轴）、4.4.4 低速到位输出、4.5.3 4 自由度串联机械臂
1.4	2018.07.30	新增以下指令： 4.3.5.7 HMC_SetGearTransationSwitch（电子齿轮过渡段开关） 4.4.12 轴位置补偿功能 5.10 RTEX 总线轴高速位置锁存 5.11 RTEX_SetMotorActualPos（设定 RTEX 电机实际位置）
1.5	2019.01.18	新增指令： 4.6 自定义运动控制功能 5.12RTEX_Home（RTEX 伺服回零） 5.13RTEX_GetHomeStatus（获取 RTEX 伺服回零状态） 5.14RTEX_ ProfileSmoothStop（RTEX_PP 模式减速停止） 5.15RTEX_ ProfileHardStop（RTEX_PP 模式急停） 9.1.6CANRecv（接收 CAN 数据） 9.1.7CANSend（发送 CAN 数据） 9.1.8CANConfig（设置 CAN 参数） 9.1.9CANFilterEnable（使能 CAN 滤波参数） 9.1.10CANFilterDisable（不使能 CAN 滤波参数） 9.1.11CANClrRecvFIFO（清空 CAN 接收 FIFO） 9.1.12TCP_CONNECT（建立 TCP 通信连接） 9.1.13TCP_SEND（TCP 数据发送） 9.1.14TCP_RECV（TCP 数据接收） 9.1.15UDP_CONNECT（创建 UDP 通信） 9.1.16UDP_SEND（UDP 数据发送） 9.1.17UDP_RECV（UDP 数据接收） 9.1.18ETH_DISCONNECT（关闭以太网通信连接）
1.6	2019.08.11	新增指令： 4.4.8.1 HMC_SetCurveJerkLimit（设定曲线冲击上限） 4.4.8.2 HMC_GetCurveJerkLimit（读取曲线冲击上限） 4.4.7.6 HMC_SetMergeSpeedTolerRatio（设定速度波动抑制比） 4.4.7.7 HMC_ GetMergeSpeedTolerRatio（读取速度波动抑制比）
1.7	2020.09.01	新增指令： 7.1CANopenReadDict（获取 CANopen 从节点对象字典参数）

版本	日期	说明
		7.2CANopenWriteDict（更新 CANopen 从节点对象字典参数） 7.3CANopenNodeNMT（发送 NMT 命令）CANopenNodeNMT（发送 NMT 命令） 7.4CANopenResetNodeDiag（清除从节点诊断信息）

1.2 文档用途

本文档通过对运动控制相关指令的详细描述与示例，让用户熟悉并掌握，并根据控制要求进行相应的编程。

1.3 阅读对象

本文档适用于具备一定的运动控制工作经验，对伺服或步进控制有一定了解的工程开发人员。

1.4 重要信息

在本手册中，使用以下标识明确相应信息：

- 危险图标，标识该操作有造成物理伤害或人身伤亡的潜在威胁。

- 电击图标，标识该操作有造成电击伤害的潜在威胁。

- 警告图标，标识该操作有造成软硬件设备故障或损坏的潜在威胁。

- 重要图标，标识需要理解的操作或功能的重要信息。

1.5 产品文档目录

AutoThink V3.1 用户手册_工程组态

MC 系列运动控制器系统手册

MC 系列运动控制器指令手册

第2章　概述

运动控制是自动化的一个分支，它使用通称为伺服机构的一些设备如液压泵、线性执行机构或者伺服电机来控制机器的位置或速度。本文主要对伺服电机的控制进行说明，被控目标是伺服驱动器或伺服电机系统。在上位控制器中，将其抽象成一个“轴”的概念。运动控制就是对轴的位置、速度等参数进行实时的控制管理，使其按照预期的运动轨迹和规定的运动参数进行运动。

一个轴包含一组轴参数，用户可通过实时查看轴参数了解该轴对应伺服驱动器的运行状态，也可通过修改轴参数值实时改变伺服驱动器的运行状态。系统提供多种运动控制算法，通过给对应轴投放运动控制算法指令，驱动该轴对应伺服驱动器进行相应动作，最终带动机械设备完成运动控制。在后续的“轴参数”章节会详细描述各个轴参数的含义和使用方法，“HMC 运动指令库”章节详细给出各个控制算法指令的功能描述和使用方法。

本章节主要描述在使用运动控制指令之前，需提前了解的整体相关知识，在此基础上，能更好理解运动控制算法指令，编写程序实现控制方案。

2.1　系统控制模式

对于运动控制，轴内核处理机制一直贯穿始终，无论是步进电机还是伺服电机，都需要控制器/驱动器/执行器在算法和控制方式上的支持。MC 系列运动控制器提供两种典型控制模式：位置开环模式（如步进轴类型）和位置闭环模式（如伺服轴类型），实现对轴不同模式下的控制。

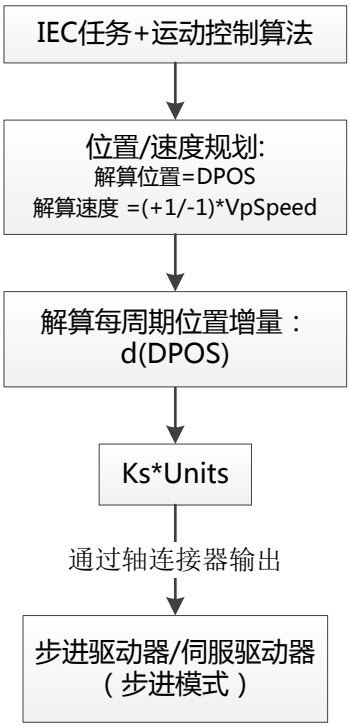


图 1　位置开环控制框图

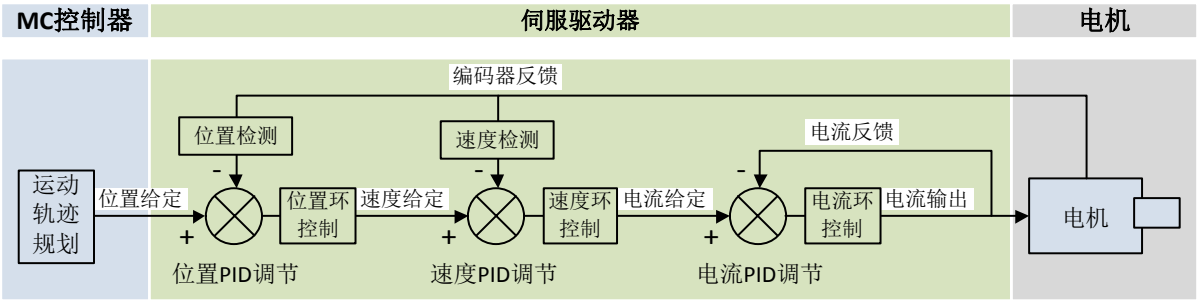


图 2 位置开环应用示意图

- 开环是针对 MC 系列运动控制器而言。

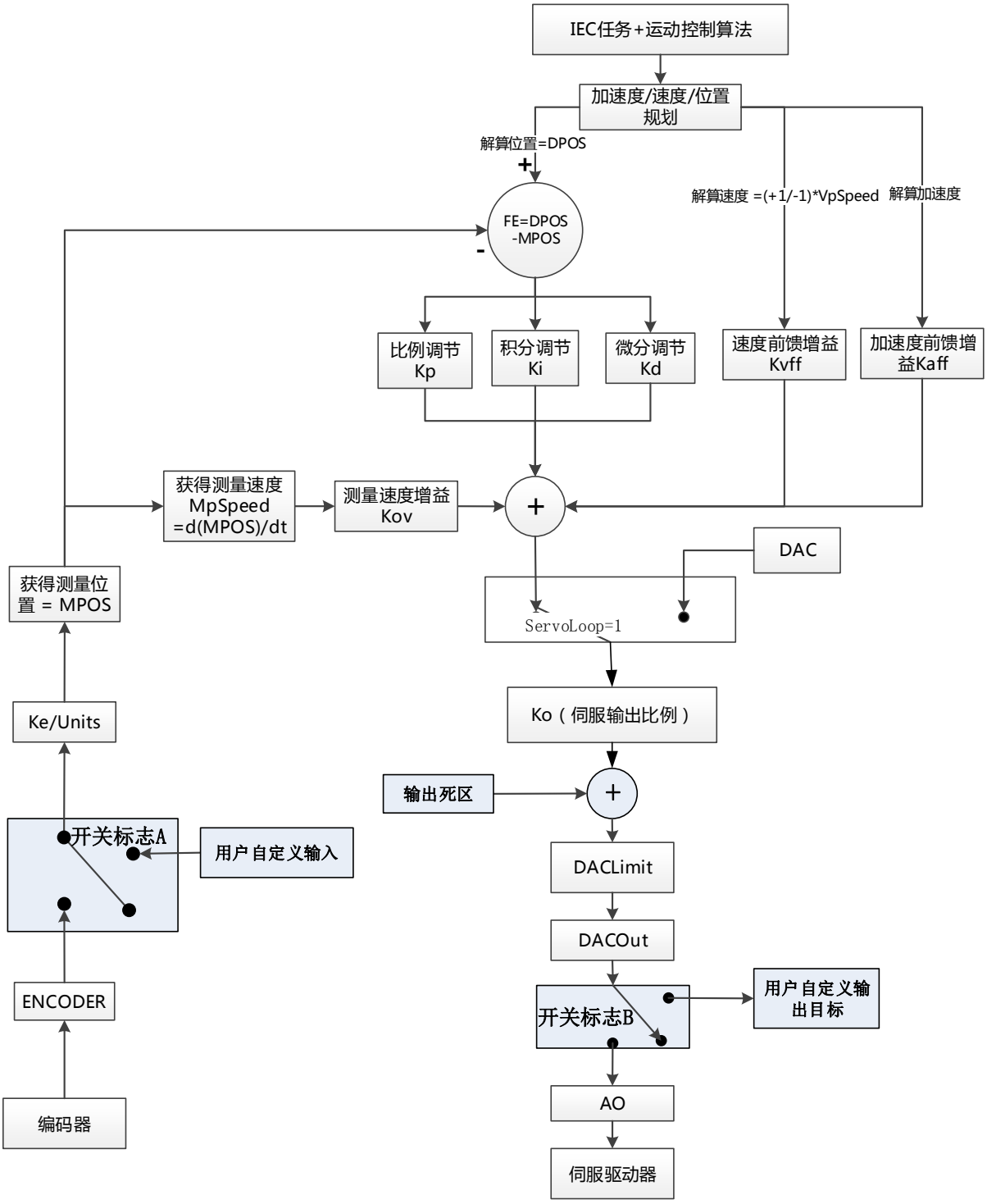

IEC任务+运动控制算法
加速度/速度/位置规划
解算位置=DPOS
+
解算速度 =(+1/-1)*VpSpeed
解算加速度
FE=DPOS -MPOS
-
比例调节 Kp
积分调节 Ki
微分调节 Kd
速度前馈增益 Kvff
加速度前馈增益Kaff
获得测量速度 MpSpeed =d(MPOS)/dt
测量速度增益 Kov
+
DAC
获得测量位置 = MPOS
ServoLoop=1
Ke/Units
Ko（伺服输出比例）
输出死区
+
开关标志A
用户自定义输入
DACLimit
DACOut
ENCODER
开关标志B
用户自定义输出目标
AO
编码器
伺服驱动器

图 3 位置闭环控制框图

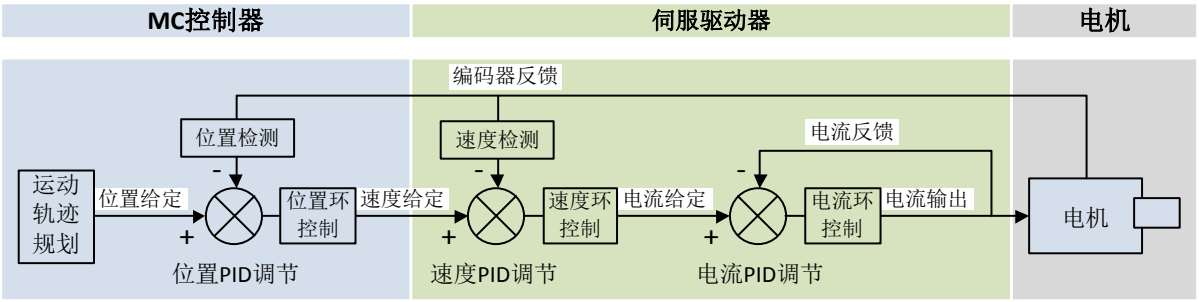


图 4 位置闭环应用示意图

- 闭环是针对 MC 系列运动控制器而言。

2.2 指令调度机制

HMC 库中的指令，从指令执行时的调度机制来看，可以分为轴运动缓存类指令和非轴运动缓存类指令。

2.2.1 轴运动缓存指令

每个轴有 64 级运动指令缓存(指令队列 / FIFO)，未被执行的轴运动缓存类指令会被暂时存入指令队列中，在后续的某个时间段执行。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，当前指令执行完毕后，会被从指令队列中移除（出队），以使后面的指令能够继续存入指令缓存（入队）。

用户在 IEC 任务中发起运动控制指令投送请求，在运动控制内核层，算法硬实时任务对该投送请求进行处理，如果满足投送条件则把指令放进指令队列，并执行指令队列中排在队首的运动控制指令，如果位于队首的指令执行完毕，则从缓存中移出。如图 5 所示。

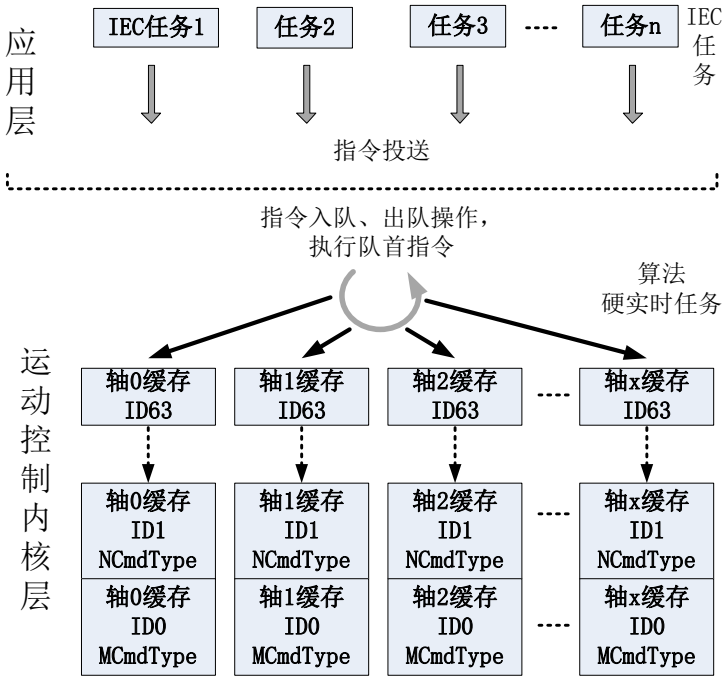


图 5 轴运动控制缓存类指令投送、处理过程

在 IEC 任务中调用轴运动缓存类指令时，若该指令所使用到的所有轴的指令缓存均未满，则该指令即刻被放到相关轴的运动指令缓存中，等待执行，指令投送完毕后函数立即返回，继续执行该指令后面的语句。当缓存中排在其前面的指令被执行完毕后，该指令即被调用执行。因此，轴运动缓存类指令对应的函数仅仅负责指令的投放，投放完成后立即返回，不管指令是否执行完成，指令可在后续某个时间段执行，这样的函数一般返回一个“指令 ID”，系统提供一些查询函数，这些查询函数可以根据“指令 ID”实时查询该指令的状态。

若在 IEC 任务中该指令投送时所使用到的某个轴的指令缓存已满，则投送该指令的函数会发生阻塞等待，直至满足指令投送的条件时，指令投送成功，函数才返回，继续后面的语句。

当前指令在执行过程中，若开启了 merge 功能，则会对指令队列中后面的指令进行前瞻，以计算 merge 所需要的运动参数。

- 轴运动缓存类指令示例（ST 语言）

  先在 AT 中启动任务 1，给轴 0 投送轴运动缓存指令 HMC_Move。再启动任务 2，查询任务 1 中 HMC_Move 指令的运动状态，根据此状态执行相应的程序。

  其中 HMC_Move(0,1000)为轴运动缓存指令。

  任务 1 程序：

```
a := 0;
id0 := HMC_Move(0, 1000);          (*该指令表示让轴 0 向正向运动 1000（用户单位），在 0 号轴指令缓存未满的情况下，指令立即被投入 0 号轴的运动缓存中，函数即刻返回，返回值为该指令的 ID*)
a := a+1;       (*a 已经变成了 1，但是 HMC_Move(0, 1000)所代表的功能尚未完成*)
```

  任务 2 程序：

```
IF HMC_GetCmdState(id0) = 2 Then          (*查询任务 1 中 HMC_Move 指令是否执行完成*)
```

```
...
```

2.2.2 非轴运动缓存指令

对于非轴运动缓存类指令，在 IEC 任务中执行该类指令时，指令会等到功能完成之后才返回，再继续执行后面的程序，否则将一直阻塞等待。

- 非轴运动缓存类指令示例（ST 语言）

  在 AT 中启动任务 1，给轴 0 一个运动指令。

  其中 HMC_WaitCmdFinish(id0)为非轴运动缓存指令。

  任务 1 程序：

```
a := 0;
id0 := HMC_Move(0, 1000);      (*该指令表示让轴 0 向正向运动 1000（用户单位），在 0 号轴指令缓存未满的情况下，指令立即被投入 0 号轴的运动缓存中，函数即刻返回，返回值为该指令的 ID*)
HMC_WaitCmdFinish(id0);        (*等待直到运动指令（id0）运行完成后才返回。当函数返回时，HMC_Move(0, 1000)已执行完成。*)
a := a+1;                      (*在该语句执行之前，HMC_WaitCmdFinish(id0)所代表的功能已经执行完成。*)
```

2.3 速度加减速规划

2.3.1 梯形曲线

在此速度规划方式下，速度曲线按梯形曲线变化。速度会经历三个阶段：匀加速阶段、匀速阶段、匀减速阶段。当位移较短时，可能不存在匀速阶段。

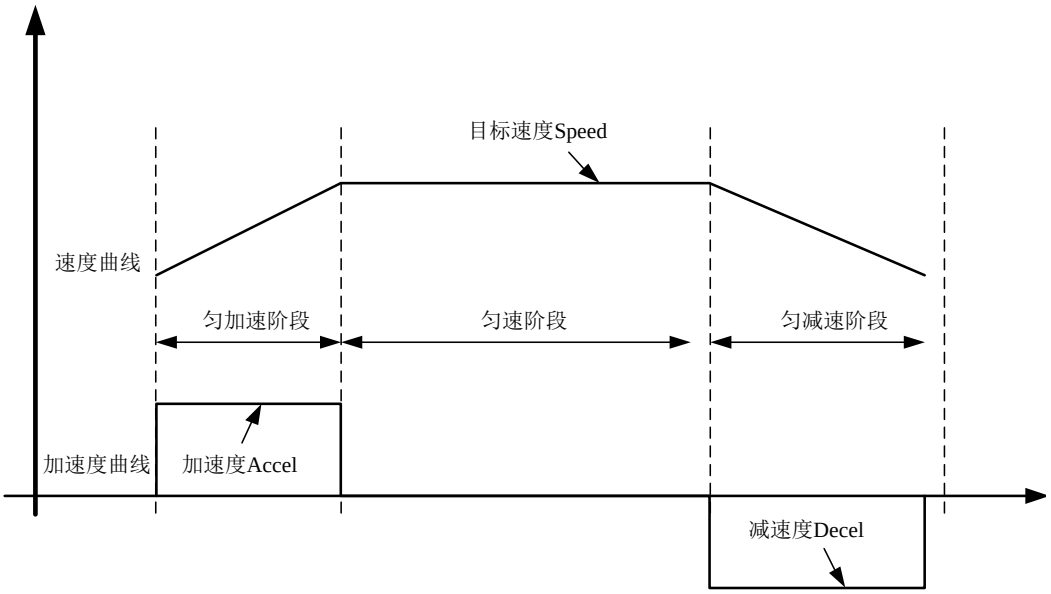


图 6 速度示意图

通过调用 HMC_SetJerkMode 指令把轴参 JerkMode 设为 0，设置当前速度加减速为梯形加减速模式，该轴参数默认为 0。

在指令运行过程中，支持在线修改加速度、减速度、目标速度等参数，且实时生效。在运动过程中修改加速度值和目标速度值，立刻生效。

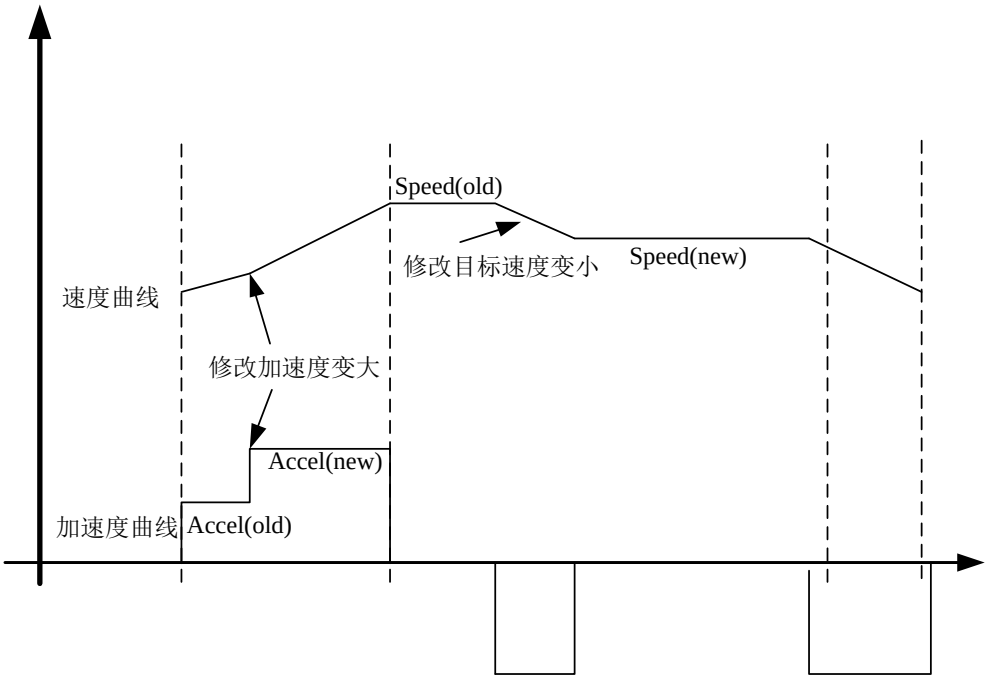


图 7 速度示意图

■ 注意事项

以下注意事项仅针对指令运行过程中动态修改参数的情况，其它情况不存在以下限制：

在指令减速阶段，不可把减速度改小，不然会造成减速距离不足，指令在结束时速度会从一定值直接减为零。

2.3.2 S 形曲线

常规的梯形加减速由于存在加速度的跃变，在加速/减速阶段的起始与终止时刻会引起动载荷的跃变，从而造成冲击和振动。采用 S 形加减速可有效的解决这一问题。S 形加减速在加速/减速阶段的加速度/减速度是按照一定的变化率（Jerk）均匀变化的，由于动负载与加速度成正比关系，从而可精准的控制动负载的变化率，抑制因动载荷的跃变而造成的冲击与振动。

完整的 S 形加减过程速分为七段：加加速、匀加速、减加速、匀速、加减速、匀减速、减减速。在加加速/减加速/加减速/减减速阶段，加速度变化率为Jerk，其余阶段为 0。匀速段/匀加速段/匀减速段不一定存在，取决于实际的运动参数设定。

使用 MC 系列控制器所提供的 S 形加减速功能，可通过设定合适的加减速变化率（轴参Jerk）来控制设备启停时的冲击。匀加速/匀减速阶段的加速度与减速度的最大值可通过设定轴参 Accel/Decel 实现，该值决定了动载荷的最大值，从而可以控制设备运行中所需的最大驱动扭矩。

通过调用 HMC_SetJerkMode 指令把轴参 JerkMode 设为 1，设置当前速度加减速为 S 形加减速模式。

在指令执行过程中可实时调整 Accel、Decel、Jerk、Speed 参数，立即生效。

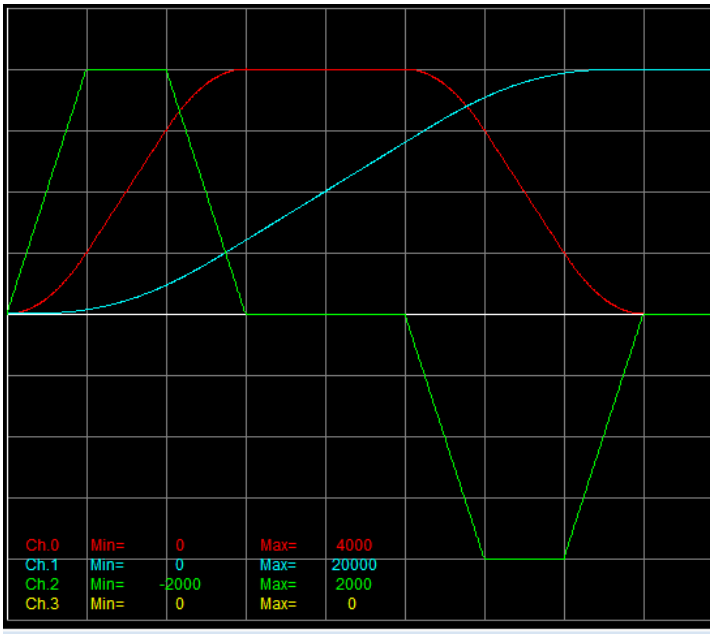


图 8 完整的 S 形加减速曲线

红色：速度曲线

绿色：加速度曲线

青色：位移曲线

在指令运行过程中，支持在线修改加加速度、加速度、减速度、目标速度等参数，且实时生效。以下为几种典型情况事例：

（1） 改变加加速度 Jerk

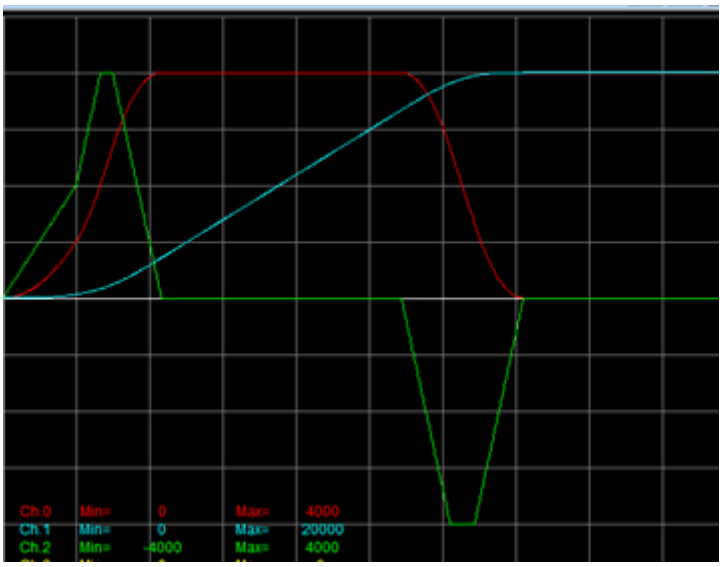


图 9 运动轨迹示意图

红色：速度曲线

绿色：加速度曲线

青色：位移曲线

（2） 改变最大加速度 Accel

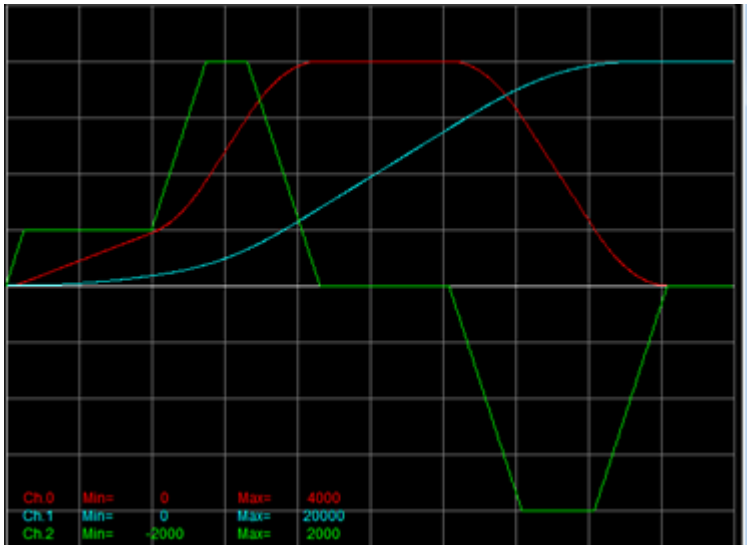


图 10 运动轨迹示意图

红色：速度曲线

绿色：加速度曲线

青色：位移曲线

（3） 改变最大减速度

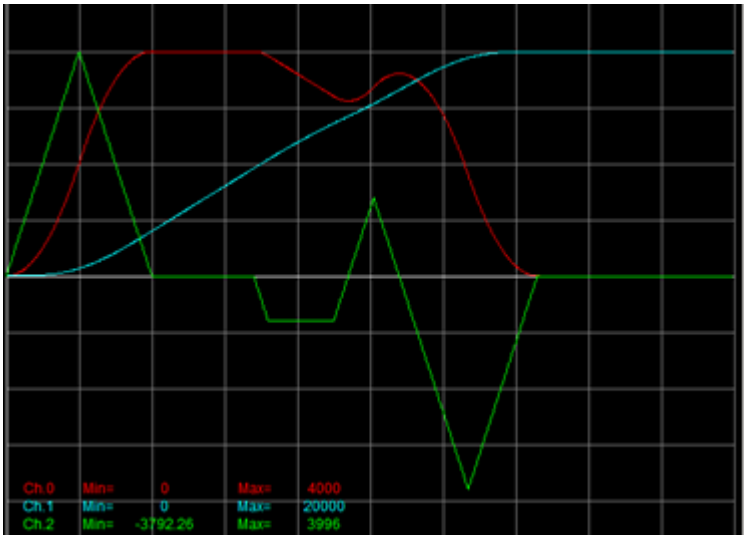


图 11 运动轨迹示意图

红色：速度曲线

绿色：加速度曲线

青色：位移曲线

（4） 改变最大速度 Speed

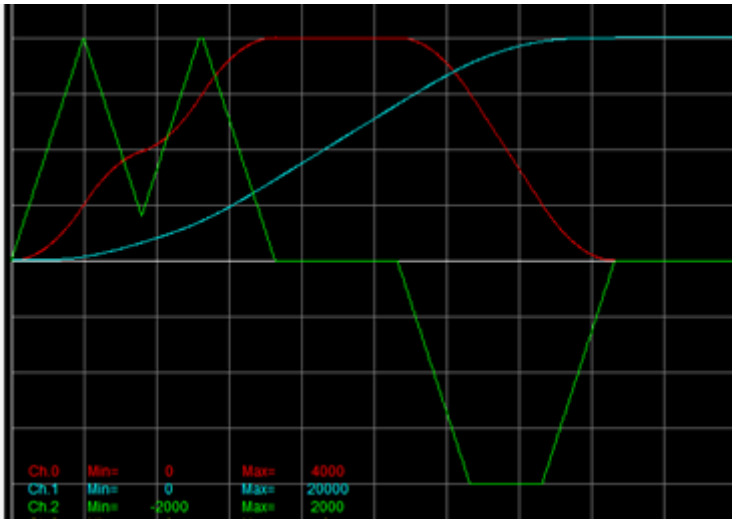


图 12 运动轨迹示意图

红色：速度曲线

绿色：加速度曲线

青色：位移曲线

■ 注意事项

以下注意事项仅针对指令运行过程中动态修改参数的情况，其它情况不存在以下限制：

（1）在指令减速阶段，不可把减速度改小，不然会造成减速距离不足，指令在结束时速度会从一定值直接减为零。

（2）指令以 S 形加减速执行过程中，可动态修改 Jerk，但当把 Jerk 修改得过小时，也可能会造成指令最终减速距离不足，结果同上。

第3章　轴参数

3.1 轴参数总体说明

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。

3.2 基本参数（Basic）

3.2.1 AxisID（轴号）

- 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

- 含义描述

  轴号，表示各组轴参数的轴号。

3.2.2 AxisType（轴类型）

- 基本属性

类型	EmAxisType
读写属性	只读
设置函数	HMC_SetAxisType
初始值	Virtual（0）：虚轴

类型	EmAxisType
取值范围	Unused（-1）：该轴未使用 Virtual（0）：虚轴 Stepper（10）：直连步进轴（脉冲+方向） Stepper_CW_CCW（11）：直连步进轴（正反脉冲） Stepper_AB（12）：直连步进轴（AB 相脉冲） Servo_AO（20）：直连伺服轴（增量式编码器+AO 输出） Encoder（30）：编码器轴（增量式编码器，AB 相计数） Encoder_PLS_DIR（31）：编码器轴（脉冲方向计数） ENCODER_CW_CCW（32）：编码器轴（正反脉冲计数） ENCODER_SSI（33）：SSI 绝对值编码器轴 （该类轴（30~39）不接受运动控制指令，仅仅跟踪编码器值） RTEX_Position（40）：RTEX 总线连接轴，位置控制（仅 MC1002R 支持） RTEX_Speed（41）：RTEX 总线连接轴，速度控制（仅 MC1002R 支持） RTEX_Torque（42）：RTEX 总线连接轴，转矩控制（仅 MC1002R 支持） RTEX_PP（43）：RTEX 总线连接轴，Profile Position 控制（该模式仅用于 RTEX 伺服原点回归） EtherCAT_Position（50）：EtherCAT 总线连接轴，位置控制（仅 MC1002E 支持） EtherCAT_Speed（51）：EtherCAT 总线连接轴，速度控制（仅 MC1002E 支持） EtherCAT_Torque（52）：EtherCAT 总线连接轴，转矩控制（暂不支持）
相关章节	4.7.1 HMC_SetAxisType（设置轴类型）

■ 含义描述

轴类型，每 10 个为一大类，例如类型 10~19 代表的大类为直连步进轴。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，参见章节 4.7.1 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。

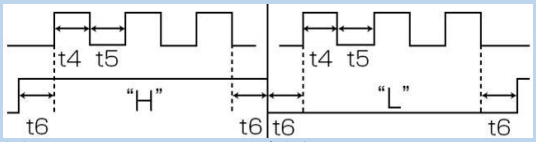
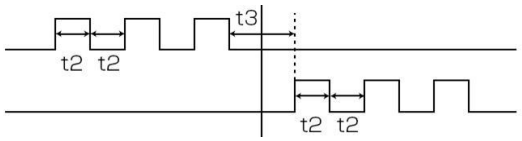
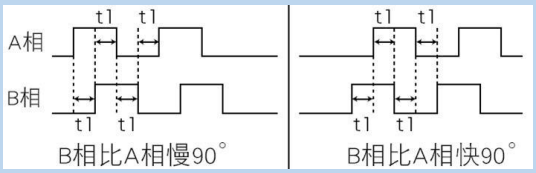
轴类型的不同，会影响该轴的输入、输出方式，各轴类型的输入、输出信号见下表。更改轴类型后，对应端子的输入、输出信号会发生变化，请确认无误后，再进行对应操作。

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
Stepper	10	直连	无	脉冲+方向差分信号输出，详见图 13	Z+、Z-为编码器输入
Stepper_CW_CCW	11	直连	无	正负脉冲差分信号输出，详见图 14	Z+、Z-为编码器输入

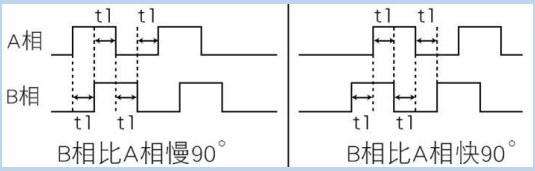
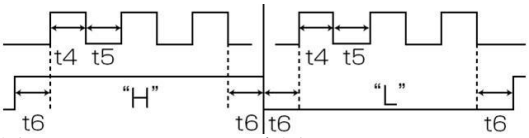
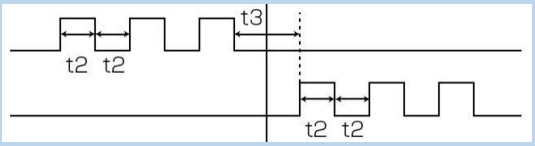
轴类型标示	轴类型代码	与控制器连接方式	输入	输出	备注
Stepper_AB	12	直连	无	AB 相正交脉冲差分信号输出，详见图 15	输出脉冲数量方面：Stepper_AB 是 Stepper 的 4 倍频，Z+、Z-为编码器输入
Servo_AO	20	直连	AB 相差分信号输入	使用 AO：使用对应轴号的 AO 口输出，例如 AO5 对应的是轴 5 的输出	-
Encoder	30	直连	AB 相差分信号输入，详见图 16	无	运动控制指令不允许投放到该轴
Encoder_PLS_DIR	31	直连	脉冲+方向差分信号输入，详见图 17	无	运动控制指令不允许投放到该轴
ENCODER_CW_CCW	32	直连	正反向脉冲差分信号输入，详见图 18	无	运动控制指令不允许投放到该轴
ENCODER_SSI	33	直连	SSI 绝对值编码器轴	无	运动控制指令不允许投放到该轴
RTEX_Position	40	RTEX 总线连接	通过 RTEX 总线接收端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 RTEX 总线发送端输出指令位置	-
RTEX_Speed	41	RTEX 总线连接	通过 RTEX 总线接收端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 RTEX 总线发送端输出指令速度	-
RTEX_Torque	42	RTEX 总线连接	通过 RTEX 总线接收端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 RTEX 总线发送端输出指令转矩	-
RTEX_PP	43	RTEX 总线连接	通过 RTEX 总线接收端获得编码器反馈位置（轴的位置信息），不作位置闭环控制	无	该模式仅用于 RTEX 伺服的原点回归功能（RTEX_Home 指令）
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），	通过 EtherCAT 总线输出指令转矩	-

轴类型标示	轴类型代码	与控制器连接方式	输入	输出	备注
			servoLoop = 1 时作位置闭环控制运算		

■ 步进模式

轴类型	脉冲输出形态	信号名称	正向脉冲输出	负向脉冲输出
AxisType=10 （脉冲+方向输出）	脉冲序列+符号	Step Direction	 图 13 AxisType=10 示意图	
AxisType=11 （正负脉冲输出）	正脉冲序列+负脉冲序列	Step Direction	 图 14 AxisType=11 示意图	
AxisType=12 （AB 相脉冲输出）	90° 相位差 2 相脉冲(A 相+B 相)	Step Direction	 图 15 AxisType=12 示意图	
说明	（1）信号输出类型为差分信号； （2）脉冲序列输出最高频率 2 MHz，占空比 50%，AB 相输出时相位差为 90°； （3）最小时间宽度：t1 为 0.125 μs，t2、t4、t5 为 0.25 μs，t3、t6 为 0.5 μs。			

■ 编码器模式

增量式编码器	脉冲输入形态	信号名称	编码器输入信号（正向）	编码器输入信号（负向）
AxisType=30 （AB 相编码器轴）	90° 相位差 2 相脉冲（A 相+B 相）	Enc.A Enc.B Enc.Z	 图 16 AxisType=30 示意图	
AxisType=31 （脉冲+方向编码器轴）	脉冲序列+符号	Enc.A Enc.B Enc.Z	 图 17 AxisType=31 示意图	
AxisType=32 （正负脉冲编码器轴）	正脉冲序列+负脉冲序列	Enc.A Enc.B Enc.Z	 图 18 AxisType=32 示意图	
说明	（1）信号输入类型为差分信号； （2）脉冲输入最高频率 6 MHz（单相），占空比 50%，AB 相输入时相位差为 90°； （3）最小时间宽度：t1 为 0.125 μs，t2、t4、t5 为 0.25 μs，t3、t6 为 0.5 μs；			

增量式编码器	脉冲输入形态	信号名称	编码器输入信号（正向）	编码器输入信号（负向）
	（4）Enc.Z 相用来接收编码器每转一圈时产生的 Z 相脉冲。			

SSI 总线绝对值编码器	说明
AxisType=33 （SSI 总线绝对值编码器轴）	（1）在设定轴类型为 SSI 编码器轴之前，可使用 HMC_SetSSIEncoderBit、HMC_SetSSIEncoderDataType、HMC_SetSSIEncoderFreq、HMC_SetSSIEncoderTp 配置相关参数； （2）初始化 Mpos 为 SSI 编码器绝对位置的时机： 使用 HMC_SetAxisType 成功设定该轴类型为 SSI 轴类型时； 轴类型已经为 SSI 轴类型，使用 HMC_SetSSIEncoderBit、HMC_SetSSIEncoderDataType 更改编码器位数/编码器码制成功时； （3）初始化绝对位置后，后续采用初始位置+每个伺服周期位置增量的方式计算 Mpos； 如果需要 MPos 的值始终与编码器绝对位置一致，则可设定该轴为循环行程（RepMode=1），且循环行程的行程长度与编码器的位置数据范围一致（如 12 位编码器，在 Units、Ke 均为 1 的情况下，RepMode=1 时应设置 RepDist = 2^12； 在设置 RepMode=2、RepDist=编码器位置数据范范围/2 的情况下，可实现该轴 Mpos 区间长度=编码器位置数据区间长度，并且 Mpos 的范围区间是关于零点对称的（如 12 位编码器，在 Units、Ke 均为 1 的情况下，应设置 RepMode=2、RepDist=2^12/2）； 如果设定该轴为有限行程，则如果编码器未发生数据溢出时，Mpos 与编码器绝对位置一致；当编码器发生数据溢出时，则控制器仍按实际增量位移计算 Mpos，而此时编码器内部的绝对位置数据则因数据溢出而发生了跳变； （4）支持 SSI 轴的模块：MC1008-B02 支持 4 个 SSI 轴（轴号 0、1、2、3），MC1004-B01 支持 2 个 SSI 轴（轴号 0、1），MC1008-A01/MC1008-A02/MC1008-B01/MC1004-A01/MC1002R-A01/MC1002E-A01 不支持。

- 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, 20);  (*Servo_AO 的轴类型枚举值为 20*)
IF 20 = Axis1.AxisType THEN
Axis1.Speed := 4000;
END_IF
```

3.2.3 Units（单位转换因子）

- 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	4.7.3 HMC_SetUnits（设置用户单位）

- 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，参见章节 4.7.3 HMC_SetUnits（设置用户单位）。

3.3 状态参数（Status）

3.3.1 MCmdType（当前指令类型）

- 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
相关章节	2.2 指令调度机制

- 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。

表 1 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	-
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	-

值	名称	说明	备注
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	-

值	名称	说明	备注
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	-

3.3.2 NCmdType（下一条指令类型）

- 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
相关章节	2.2 指令调度机制

- 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。

3.3.3 AxisStatus（轴状态）

- 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 位含义见表 2。
相关章节	3.3.4 AxisErrMask（轴状态掩码）

■ 含义描述轴

当前运行状态，为 0 时表示该轴无异常。不为 0 时，按位解析如下：

表 2 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。

■ 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 位与，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
```

3.3.4 AxisErrMask（轴状态掩码）

- 基本属性

类型	UDINT
读写属性	读写
设置函数	-
初始值	1
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	3.3.3 AxisStatus（轴状态）

- 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。

- 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）*)
```

3.3.5 LimitState（超限状态）

- 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	3.4.5 RepMode（位置行程模式） 3.7.2 FsLimitMode/RsLimitMode（前/后限位时指令撤销方式） 3.7.3 FHLimitIn/RHLimitIn（前/后向硬限位通道号） 3.7.4 FSLimit/RSLimit（前/后向软限位边界值） 10.3 有限行程轴超限处理

- 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。

- 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

- 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。

3.4 位置相关参数（Positions）

3.4.1 MPos（反馈位置）

- 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	3.4.2 DPos（本周期目标位置） 3.4.5 RepMode（位置行程模式） 3.4.6 RepDist（循环行程长度） 4.2.1.4 HMC_DefPos（设置当前位置）

- 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，如伺服轴、RTEX 总线轴、EtherCAT 总线轴、编码器轴，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。

轴类型为直连步进轴时，其 MPos 根据当前轴实际已发送的脉冲数计算。

（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。

对于 SSI 编码器轴类型，在设置轴类型后，或通过 HMC_SetSSIEncoderBit 设置 SSI 编码器位数，或通过 HMC_ SetSSIEncoderDataType 设置编码器轴码制时，系统会自动使用绝对值编码器当前位置初始化轴的 Mpos，但后续仍采用增量计算。

3.4.2 DPos（本周期目标位置）

- 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	3.4.5 RepMode（位置行程模式） 3.4.6 RepDist（循环行程长度）

- 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：

（1）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；

（2）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，伺服轴的 Dpos 会出现在循环行程区间之外；

（3）对于伺服轴（轴类型为 20~29）、RTEX_Speed 轴（轴类型为 41）、EtherCAT_Speed 轴（轴类型为 51），当 ServoLoop 为 0 时，Dpos 实时等于 Mpos。

3.4.3 FE（偏差）

- 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	3.4.4 StopFETol（反馈位置是否到达目标位置的死区） 3.7.1 FELimits（随动误差限制）

- 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。

3.4.4 StopFETol（反馈位置是否到达目标位置的死区）

- 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	3.4.3 FE（偏差）

- 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。

3.4.5 RepMode（位置行程模式）

- 基本属性

类型	SINT

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	3.4.6 RepDist（循环行程长度） 3.4.1 MPos（反馈位置）

■ 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。

3.4.6 RepDist（循环行程长度）

■ 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

类型	LREAL
相关章节	3.4.5 RepMode（位置行程模式）

■ 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。

3.4.7 EndPos（指令目标位置）

■ 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	-

■ 含义描述

当前运动指令的目标位置，用户单位。

系统按照以下原则计算和处理轴参数 EndPos：

（1）当一条指令刚开始执行时，系统将更新一次 EndPos 参数，显示当前运动指令的目标位置；

示例 1：假设一开始轴 0 的 EndPos 为 10000，那么下面指令刚开始执行时，轴 0 的 EndPos 将会变为 20000。

```
HMC_Move(0,10000);    (*轴 0 正向位移 10000*)
```

示例 2：假设一开始轴 0~2 的 EndPos 为 0，那么下面指令刚开始执行时，轴 0 的 EndPos 将会变为 50000，轴 1 的 EndPos 将会变为 30000，轴 2 的 EndPos 将会变为 40000。

```
HMC_Move2D(0,1,2,30000,40000); (*轴 1 与轴 2 做直线插补，轴 1 位移 30000，轴 2 位移 40000，轴 0 为合轴。*)
```

示例 3：假设一开始轴 0~2 的 EndPos 为 0，那么下面指令刚开始执行时，轴 0 的 EndPos 将会变为 2*π*10000，轴 1 的 EndPos 仍为 0，轴 2 的 EndPos 仍为 0。

```
HMC_MoveCircle(0,1,2,3,0,0,10000,0);   (*轴 1 与轴 2 做圆弧插补，轴 1 为 X 轴，轴 2 为 Y 轴，轴 0 为合轴。圆心坐标为（10000，0），以当前点（0，0）为起点顺时针画一整圆，那么最终 X 轴和 Y 轴又回到起点（0，0）.*)
```

（2）对于联动类指令 Gear、Cam、Superimpose，系统每个伺服周期更新一次 EndPos，轴 EndPos 永远等于 DPos；

（3）对于持续运动指令 HMC_Forward、HMC_Reverse，系统每 1000 个伺服周期更新一次 EndPos，更新时机为 Remain≤500 个伺服周期位移量，EndPos 增加量为 1000 个伺服周期位移量；

（4）HMC_DefOrigin、HMC_DefPos 和 Home 等指令完成后，系统将更新 EndPos；

（5）当指令被 Cancel 时，系统在 Cancel 完成时将更新 EndPos；

（6）对于循环行程的轴，EndPos 的值始终在[0,RepDist)或[-RepDist,RepDist)之间；

（7）对于有限行程的轴，当行程超限系统完成超限处理后将更新 EndPos。

3.4.8 Remain（剩余位置）

- 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	-

- 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。

3.5 速度相关参数（Velocity Profile）

3.5.1 Speed（目标速度）

- 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	-

- 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。

3.5.2 Accel（加速度）

- 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	-

- 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。

3.5.3 Decel（减速度）

- 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	-

- 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。

3.5.4 MpSpeed（反馈速度）

- 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	-

- 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）伺服轴（轴类型为 20~29）、编码器轴（轴类型为 30~39）、RTEX_Position 轴（轴类型为 40）、RTEX_Speed（轴类型为 41）、RTEX_Torque 轴（轴类型为 42）、RTEX_PP 轴（轴类型为 43）、EtherCAT_Position 轴（轴类型为 50）、EtherCAT_Speed（轴类型为 51）、EtherCAT_Torque（轴类型为 52），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。

3.5.5 VpSpeed（解算速度）

- 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	3.5.6 Direction（解算速度的方向）

- 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。

3.5.6 Direction（解算速度的方向）

- 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	3.5.5 VpSpeed（解算速度）

- 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。

3.5.7 Merge（融合功能模式）

- 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	4.4.7 运动前瞻

- 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。

3.5.8 CreepSpeed（原点回归爬行速度）

- 基本属性

类型	LREAL
读写属性	读写
设置函数	-

类型	LREAL
初始值	200
取值范围	正浮点数
相关章节	4.2.1.1 HMC_Home（原点回归）

- 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。

3.5.9 FastDecel（快速减速度）

- 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
相关章节	10.3 有限行程轴超限处理

- 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）。

3.5.10 Jerk（加加速度）

- 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	4.4.7.2 HMC_ClcJerk（冲击运算）

- 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。

- 不支持的模块版本

  MC1008-A01

3.5.11 JerkMode（加速模式）

- 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	-

- 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。

- S 形加减速不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-B02、MC1004-A01、MC1002R-A01、MC1002E-A01

3.5.12 CmdStopMode（指令结束判断方式）

- 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	-

- 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。

（a）DPOS 已经到达目标位置

（b）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。

3.5.13 MoveAbsMode（循环行程轴绝对运动时方向选取方式）

- 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	-

- 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);
```

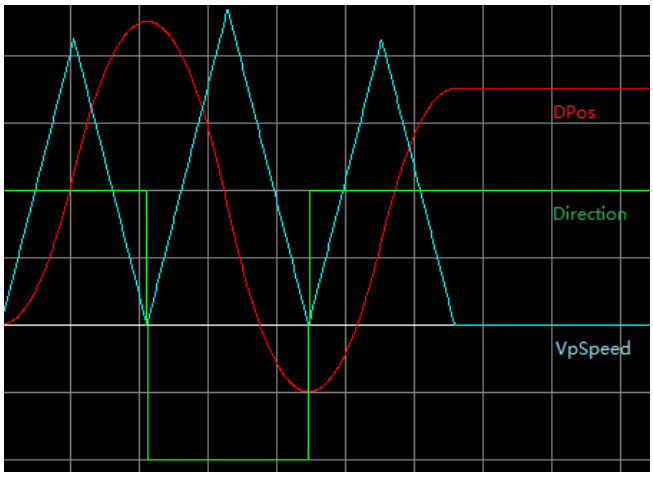


图 19 MoveAbsMode=0 在循环行程内确认方向

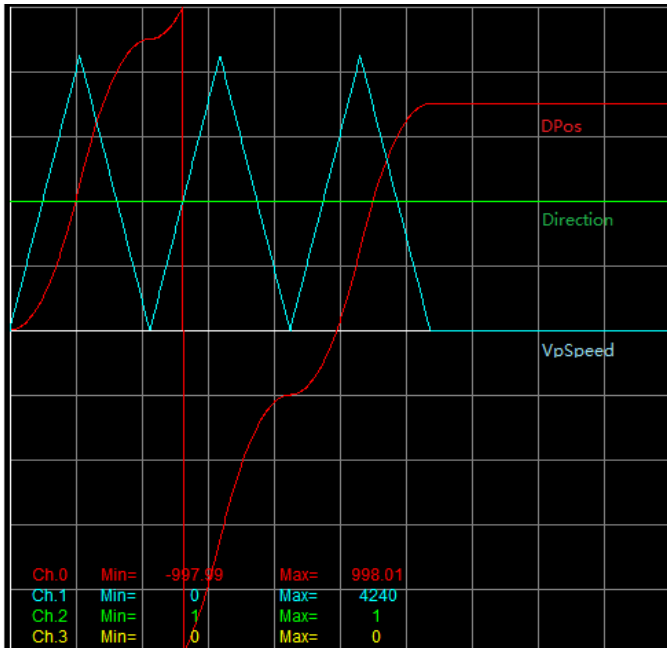


图 20 MoveAbsMode=1 取最短距离方向

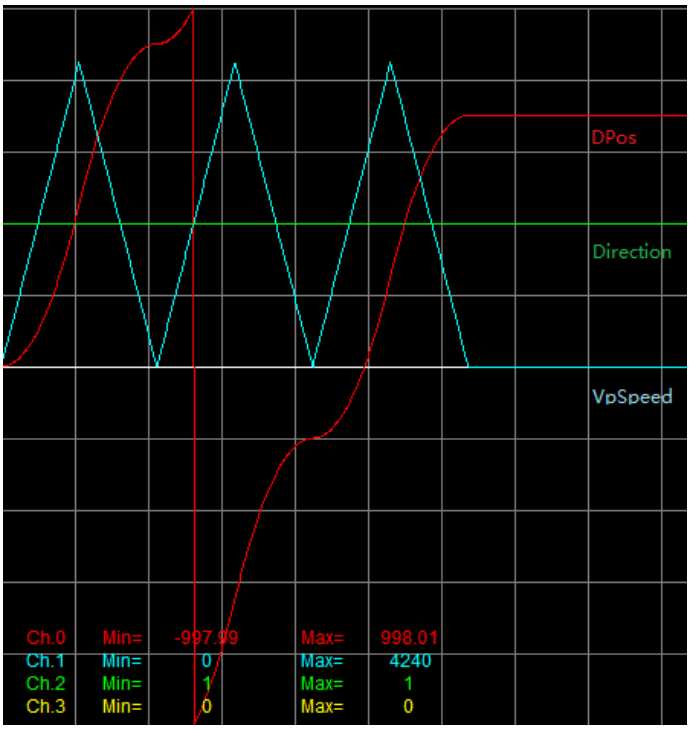


图 21 MoveAbsMode=0 始终正方向

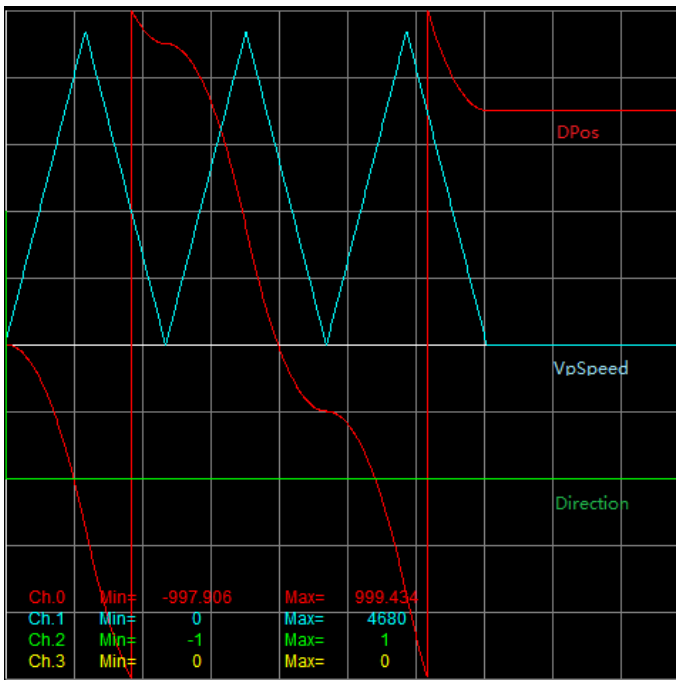


图 22 MoveAbsMode=0 始终负方向

3.5.14 CancelMode（Cancel 时指令停止模式）

- 基本属性

类型	USINT

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	3.5.15 CancelPos（Cancel 时停止位置） 4.2.4.1 HMC_Cancel（撤销指令）

■ 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。

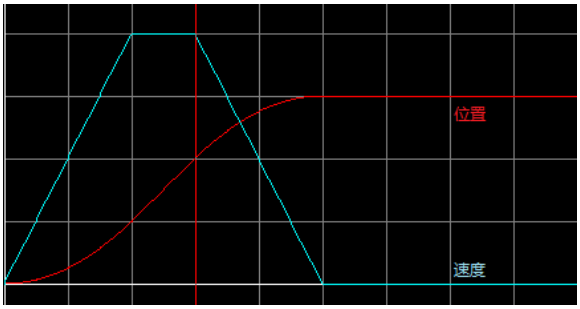


图 23 指令停止示意图

（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：

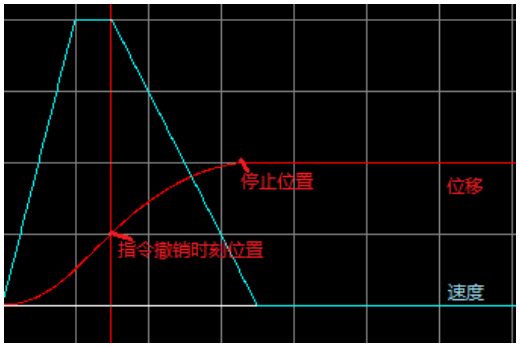


图 24 指令停止示意图

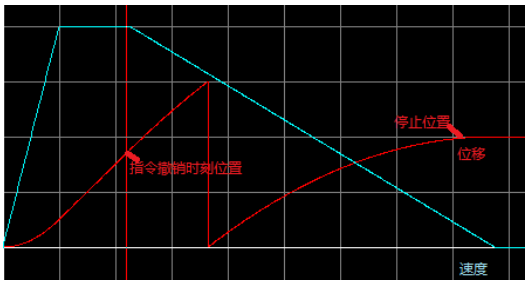


图 25 指令停止示意图

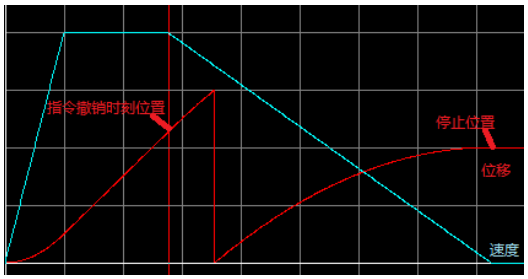


图 26 指令停止示意图

（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。

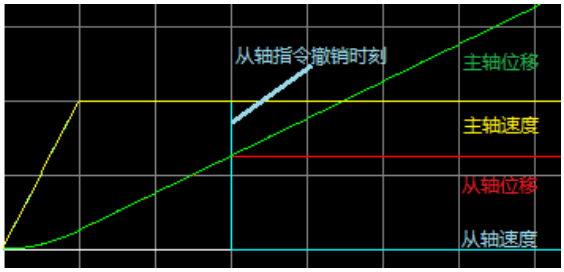


图 27 指令停止示意图

- 不支持的模块版本

  MC1008-A01

3.5.15 CancelPos（Cancel 时停止位置）

- 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	3.5.14 CancelMode（Cancel 时指令停止模式）

- 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。

- 不支持的模块版本

MC1008-A01

3.6 系数因子参数（Gains）

3.6.1 Kp（PID 的比例系数）

- 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
相关章节	2.1 系统控制模式

- 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。详见图 3。

3.6.2 Ki（PID 的积分增益）

- 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
相关章节	2.1 系统控制模式

- 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))。详见图 3。

3.6.3 Kd（PID 的微分增益）

- 基本属性

类型	LREAL

类型	LREAL
读写属性	读写
设置函数	-
初始值	0
取值范围	浮点数
相关章节	2.1 系统控制模式

- 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)。详见图 3。

3.6.4 Kov（测量速度增益）

- 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
相关章节	2.1 系统控制模式

- 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。详见图 3。

3.6.5 Kvff（速度前馈增益）

- 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
相关章节	2.1 系统控制模式

- 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)。详见图 3。

3.6.6 Kaff（加速度前馈增益）

- 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
相关章节	2.1 系统控制模式

- 含义描述

加速度前馈增益，加速度前馈=Kaff*解算加速度，详见图 3。

3.6.7 Ko（PID 输出放大因子）

- 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
相关章节	2.1 系统控制模式

- 含义描述

伺服输出比例，PID 计算后伺服输出值的放大因子。详见图 3。

3.7 限幅、限位参数（Limits）

3.7.1 FELimits（随动误差限制）

- 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	-

- 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。

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

■ 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	10.3 有限行程轴超限处理

■ 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。

3.7.3 FHLimitIn/RHLimitIn（前/后向硬限位通道号）

■ 基本属性

类型	INT
读写属性	只读
设置函数	HMC_HwLimitConfig
初始值	0
取值范围	-1：未启用； DI 通道号：快速、慢速或虚拟 DI 通道号
相关章节	10.3 有限行程轴超限处理

■ 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。

3.7.4 FSLimit/RSLimit（前/后向软限位边界值）

■ 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	10.3 有限行程轴超限处理

- 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。

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

- 基本属性

类型	DINT
读写属性	读写
设置函数	-
初始值	DACLimitLow：-32,768 DACLimitHigh：32,767
取值范围	轴类型为 20 时取值范围：-32,768~32,767；其他轴类型时取值范围：-2147483648~2147483647 需满足 DACLimitHigh>DACLimitLow
相关章节	2.1 系统控制模式

- 含义描述

DACOut 的上限/下限，确保 DACOut 的输出不会超过设定范围。详见图 3。

3.8 原点和保持参数（Home&Hold）

3.8.1 HomeState（原点回归状态）

- 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 HMC_HomeEx 函数的 Mode 参数相关。 0：尚未开始寻找原点或者已经完成了一个找原点过程 1：该功能块处于第一阶段运行找原点状态（找 DI（MODE=0、1、4 或 5）或 Z 相（MODE=2 或 3）） 2：该功能块处于第二阶段运行找原点状态（找到 DI（MODE=0、1、4 或 5）或 Z 相（MODE=2 或 3）后减速到 0） 3：该功能块处于第三阶段运行找原点状态（找到 DI（MODE=0、1、

类型	USINT
	4 或 5）并减速到 0 后，反向运动找 DI 下降沿或 Z 相） 4：该功能块处于第四阶段运行找原点状态（找到 DI（MODE=0、1、4 或 5）并减速到 0 后，反向运动找到 DI 下降沿或 Z 相，减速到 0）
相关章节	4.2.1.1 HMC_Home（原点回归） 4.2.1.2 HMC_HomeEx（高级原点回归）

- 含义描述

原点回归状态，用于指示 HMC_Home、HMC_HomeEx 在原点回归过程所处状态。原点回归过程需经历 0、1、2、3、4 等 5 个状态，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。

3.8.2 HomeIn（原点回归所用的 DI 通道）

- 基本属性

类型	INT
读写属性	只读
设置函数	HMC_Home、HMC_HomeEx
初始值	-1
取值范围	HMC_Home 时取值范围 0~63（DI 通道号） HMC_HomeEx 时取值范围 0~11（快速 DI 通道） -1 时表示未使用原点回归功能
相关章节	4.2.1.1 HMC_Home（原点回归） 4.2.1.2 HMC_HomeEx（高级原点回归）

- 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。

3.8.3 HomeCapIn（原点回归使用的高速捕获通道）

- 基本属性

类型	INT
读写属性	只读
设置函数	HMC_HomeEx
初始值	-1
取值范围	当前控制器型号所支持的高速捕获通道数目，MC1008、MC1004 为 8 路，MC1002R、MC1002E 为 2 路
相关章节	4.2.1.2 HMC_HomeEx（高级原点回归）

- 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。

3.8.4 HoldSpeed（强制速度）

- 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	4.4.6.1 HMC_HoldConfig(设置强制速度开关)

- 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 4.4.6.1 HMC_HoldConfig(设置强制速度开关)。

3.8.5 HoldIn（速度强制开关 DI 通道号）

- 基本属性

类型	INT
读写属性	只读
设置函数	HMC_HoldConfig
初始值	-1
取值范围	0~63（DI 通道号） -1 表示关闭该功能
相关章节	4.4.6.1 HMC_HoldConfig(设置强制速度开关)

- 含义描述

速度强制功能的触发源 DI 通道号。

3.9 输入相关参数（Input）

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

- 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	4.7.4 HMC_SetJerkMode（设置梯形/S 形加速度）

- 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（伺服轴、编码器轴、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”，可查看“系统控制模式”章节中的位置闭环框图了解 Ke 的作用方式。

3.9.2 EncoderValue（内部编码器值）

- 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	-

- 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。

3.9.3 FeedBackSrcMode（轴位置反馈源模式）

- 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 20、30 模式时为编码器输入，40、41、50、51 时为总线输入 1：反馈输入源为用户自定义输入
相关章节	4.7.7 HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址） 3.9.4 FeedBackSrcValue（轴反馈自定义输入实时值）

- 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，例如编码器轴、伺服轴、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。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

3.9.4 FeedBackSrcValue（轴反馈自定义输入实时值）

- 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	4.7.7 HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址） 3.9.3 FeedBackSrcMode（轴位置反馈源模式）

- 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。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

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

- 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	4.7.7 HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址） 3.9.3 FeedBackSrcMode（轴位置反馈源模式）

- 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。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1008-B02、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A01

3.10 输出相关参数（Output）

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

- 基本属性

类型	DINT
读写属性	只读

类型	DINT
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	4.7.6 HMC_SetKs（设置 Ks）

- 含义描述

步进输出比例 Ks 的分子/分母，通过 HMC_SetKs 设置。该参数仅对步进轴（10~19）、RTEX_Position（40）、EtherCAT_Position（50）有效。参见章节 4.7.6 HMC_SetKs（设置 Ks）。

3.10.2 PulseNum（步进轴周期输出脉冲数）

- 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	-

- 含义描述

每个伺服周期，步进轴输出的脉冲数。

3.10.3 ServoLoop（闭环输出开关）

- 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	2.1 系统控制模式

- 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。

3.10.4 DAC（速度模式开环输出值）

- 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	2.1 系统控制模式

- 含义描述

ServoLoop=0 时的 DACOut 的输出值。

3.10.5 DACOut（速度模式输出值）

- 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	2.1 系统控制模式

- 含义描述

当前轴的速度量输出结果（直连速度模式（20），该速度量转换为 AO 模拟量来控制伺服驱动器速度；RTEX_Speed（41）/EtherCAT_Speed（51），该速度量直接通过总线给伺服驱动器）。

3.10.6 PosOffset（DACOut 正向死区值）

- 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	直连轴：-32,768~32,767 总线轴：-2,147,483,648~2,147,483,647

类型	DINT
相关章节	2.1 系统控制模式

■ 含义描述

伺服轴输出正向死区值。

当伺服轴计算的输出值 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。

对于直连轴，DACOut 取值范围为-32,768~32,767。(-32768，32767)线性对应输出电压(-10，10）V。

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。

■ 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。

■ 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

3.10.7 NegOffset（DACOut 负向死区值）

■ 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	直连轴：-32768~32767 总线轴：-2147483648~2147483647
相关章节	2.1 系统控制模式

■ 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。

3.10.8 ZeroWindow（DACOut 零点间隙死区）

■ 基本属性

类型	DINT
读写属性	读写
设置函数	-

类型	DINT
初始值	0
取值范围	直连轴：-32768~32767 总线轴：-2147483648~2147483647
相关章节	2.1 系统控制模式

- 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）。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

3.10.9 OutDstMode（DACOut 输出目标模式）

- 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	4.7.8 HMC_SetOutDstAddr（设置用户自定义输出地址）

- 含义描述

对于速度模式的轴类型，如伺服轴 20、RTEX 总线速度模式 41，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 20 模式目标输出为对应 AO 通道，41、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：20 轴类型时，AO 通道不与轴关联，AO 通道变为普通模拟量输出通道；对于 41、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。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

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

- 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	4.7.8 HMC_SetOutDstAddr（设置用户自定义输出地址）

- 含义描述

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

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

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

- 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	4.7.8 HMC_SetOutDstAddr（设置用户自定义输出地址）

- 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。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-B02、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A01

3.11 预留参数（Reserved）

3.11.1 SystemValue*

- 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

类型	SystemValue0、SystemValue1：UDINT SystemValue2、SystemValue3：LREAL
取值范围	-
相关章节	-

■ 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	-	-

第4章　HMC 运动指令库

4.1 系统启动指令

4.1.1 HMC_DriveEnable（伺服使能）

- 函数原型

  UDINT HMC_DriveEnable(USINT bMode)

- 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）对于直连脉冲型的伺服驱动器，DriveEnable 信号对应的引脚直接与驱动器上的伺服使能信号相连接；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（例如直连伺服轴 DACOut 为 0，直连步进轴也停止发脉冲）。

- 输入参数

输入参数	类型	参数描述
bMode	USINT	0：使能禁止 1：使能开启

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

- 补充说明

  （1）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。

  （2）在 AutoThink 菜单栏点击使能按钮，也可控制伺服使能状态切换；

  （3）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。

- 应用举例

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

4.1.2 HMC_GetDriveEnableState（获取伺服使能状态）

- 函数原型

  USINT HMC_GetDriveEnableState ()

- 功能说明

  获取伺服总使能（DriveEnable）状态。

- 返回值

返回值	类型	参数描述
0	USINT	禁用
1	USINT	使能

- 指令类型

  非轴运动缓存指令。

- 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 RTEX_GetDriveEnableState、EtherCAT_ GetDriveEnableState。

- 应用举例

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

4.2 单轴运动指令

4.2.1 原点搜寻及位置定义

4.2.1.1 HMC_Home（原点回归）

- 函数原型

  UDINT HMC_Home (USINT ucAxisID, USINT ucHomeMode, USINT ucDiChl)

- 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。

- 参数说明

输入参数	类型	参数描述
ucAxisID	USINT	轴号

输入参数	类型	参数描述
ucHomeMode	USINT	0：仅使用 DI 信号正向定位原点 1：仅使用 DI 信号负向定位原点 2：仅使用本轴 Z 相信号正向定位原点 3：仅使用本轴 Z 相信号负向定位原点 4：使用 DI 及本轴 Z 相信号正向定位原点 5：使用 DI 及本轴 Z 相信号负向定位原点 注：详见应用示例
ucDiChl	USINT	原点回归时使用的 DI 通道号：0~63（快速/ 慢速/ 虚拟 DI）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 注意事项

（1）对于 RTEX 和 EtherCAT 总线轴类型，由于总线类型的驱动器再生脉冲口未提供 Z 相信号输出，所以涉及到 Z 相信号的原点回归功能不可用。

（2）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。

- 应用示例

（1）模式 0 原点回归（ucHomeMode = 0）

仅使用 DI 信号正向定位原点。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）。

特殊情形：如果一开始 DI 为 1，则直接进入阶段 3。

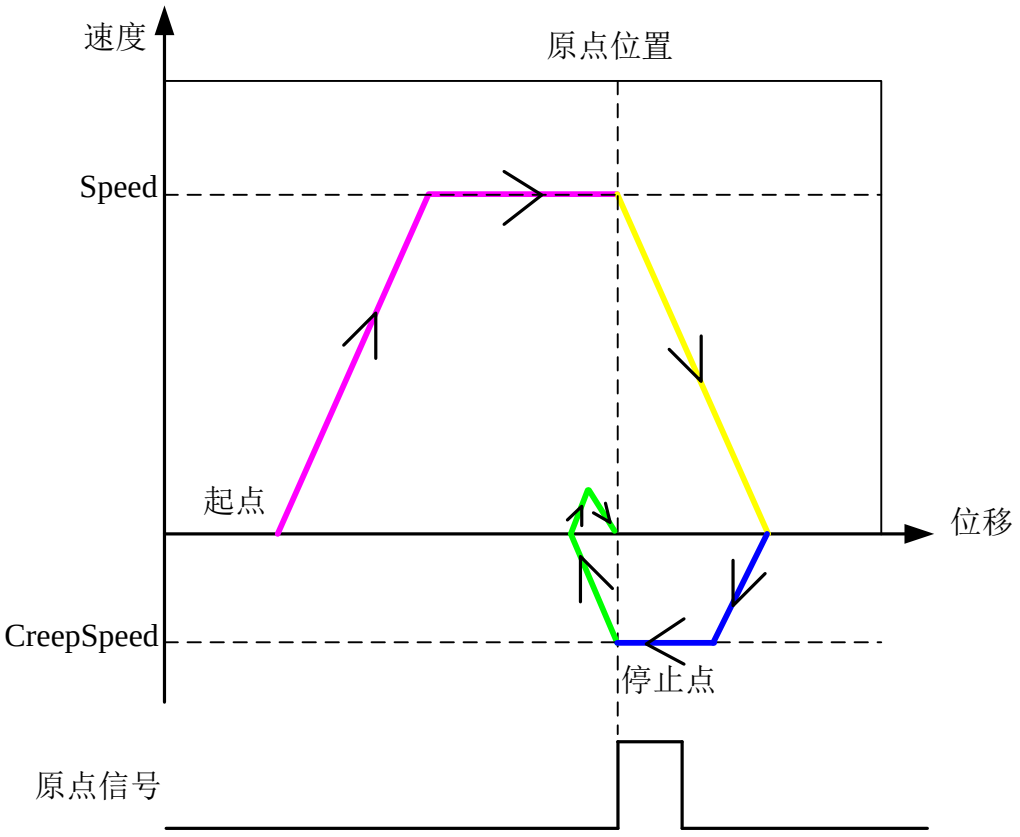


图 28 模式 0 原点回归

（紫色、黄色、蓝色、绿色 对应回归阶段 1、2、3、4）

示例程序如下（ST 语言）：

```
    Axis0.Speed := 1000;          (*设定原点回归速度(阶段 1 速度)*)
    Axis0.CreepSpeed := 100;          (*设定爬行速度(阶段 3 速度)*)
    Axis0.Accel := 2000;              (*设定加速度*)
    Axis0.Decel := 2000;              (*设定减速度*)
    HMC_Home(Axis0.AxisID, 0, 0);     (*向 Axis0 轴投送原点回归指令(模式 0，使用 0 号快速
DI 通道)*)
```

（2）模式 1 原点回归（ucHomeMode = 1）

仅使用 DI 信号负向定位原点。轴以 Speed 设定的速度负向运动（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）。

特殊情形：同模式 0。

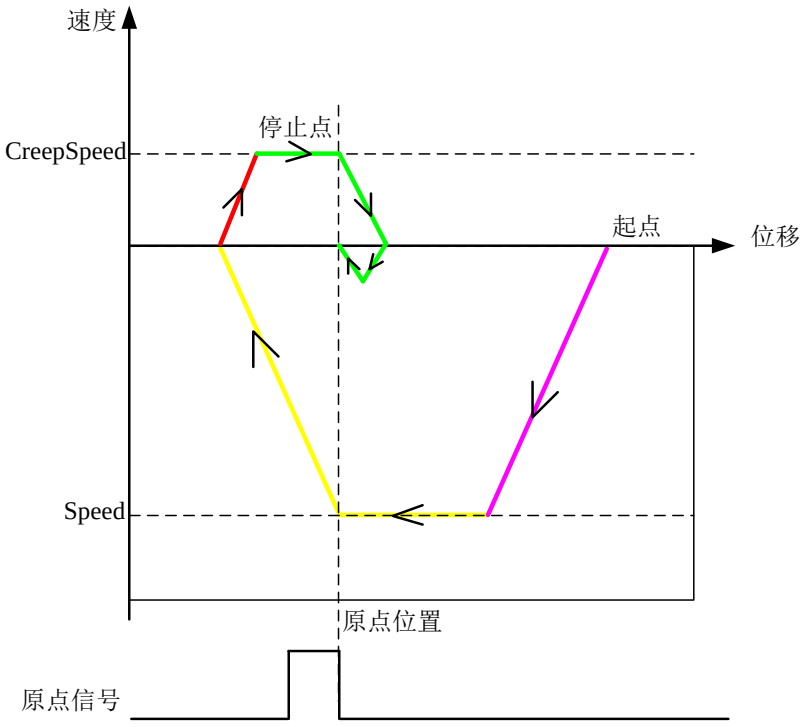


图 29 模式 1 原点回归

（紫色、黄色、红色、绿色 对应回归阶段 1、2、3、4）

示例程序如下（ST 语言）：

```
    Axis0.Speed := 1000;                (*设定原点回归速度(阶段 1 速度)*)
    Axis0.CreepSpeed := 100;            (*设定爬行速度(阶段 3 速度)*)
    Axis0.Accel := 2000;                (*设定加速度*)
    Axis0.Decel := 2000;                (*设定减速度*)
    HMC_Home(Axis0.AxisID, 1, 0);       (*向 Axis0 轴投送原点回归指令(模式 1，使用 0 号快速
DI 通道)*)
```

（3）模式 2 原点回归（ucHomeMode = 2）

仅使用本轴 Z 相信号正向定位原点。轴以 CreepSpeed 设定的速度正向运动（阶段 1），当 Z 相为 1 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 2）。

特殊情形：如果一开始 Z 相为 1，则直接进入阶段 2。

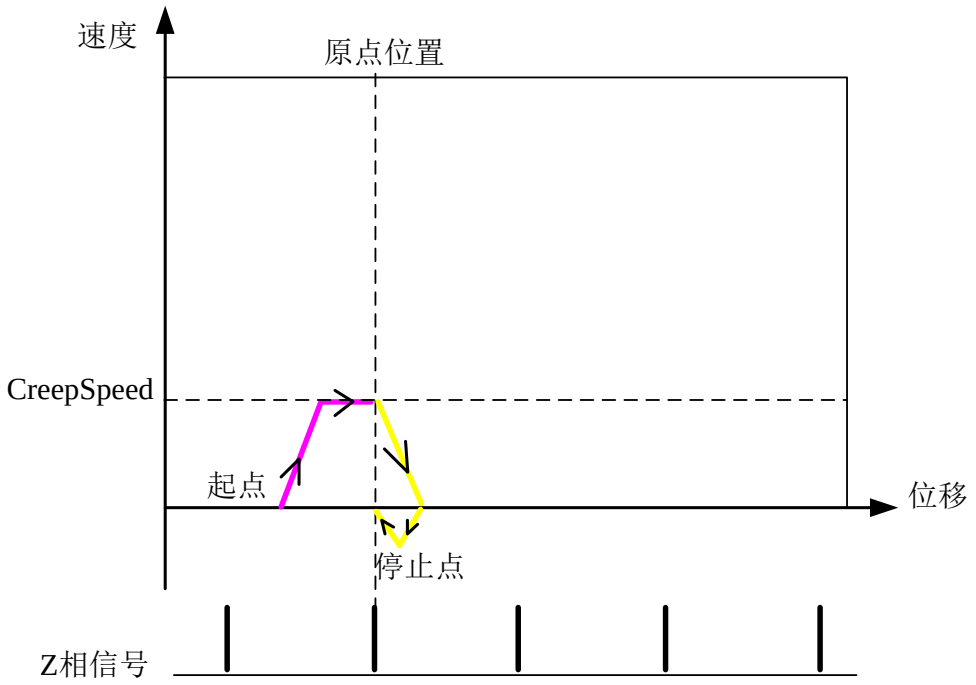


图 30 模式 2 原点回归（紫色、黄色 对应回归阶段 1、2）

示例程序如下（ST 语言）：

```
    Axis0.Speed := 1000;                (*设定原点回归速度(阶段 1 速度)*)
    Axis0.CreepSpeed := 100;            (*设定爬行速度(阶段 3 速度)*)
    Axis0.Accel := 2000;                (*设定加速度*)
    Axis0.Decel := 2000;                (*设定减速度*)
    HMC_Home(Axis0.AxisID, 2, 0);       (*向 Axis0 轴投送原点回归指令(模式 2,使用 Axis0 轴 Z 相通道)*)
```

（4）模式 3 原点回归（ucHomeMode = 3）

仅使用本轴 Z 相信号负向定位原点。轴以 CreepSpeed 设定的速度负向运动（阶段 1），当 Z 相为 1 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 2）。

特殊情形：同模式 2。

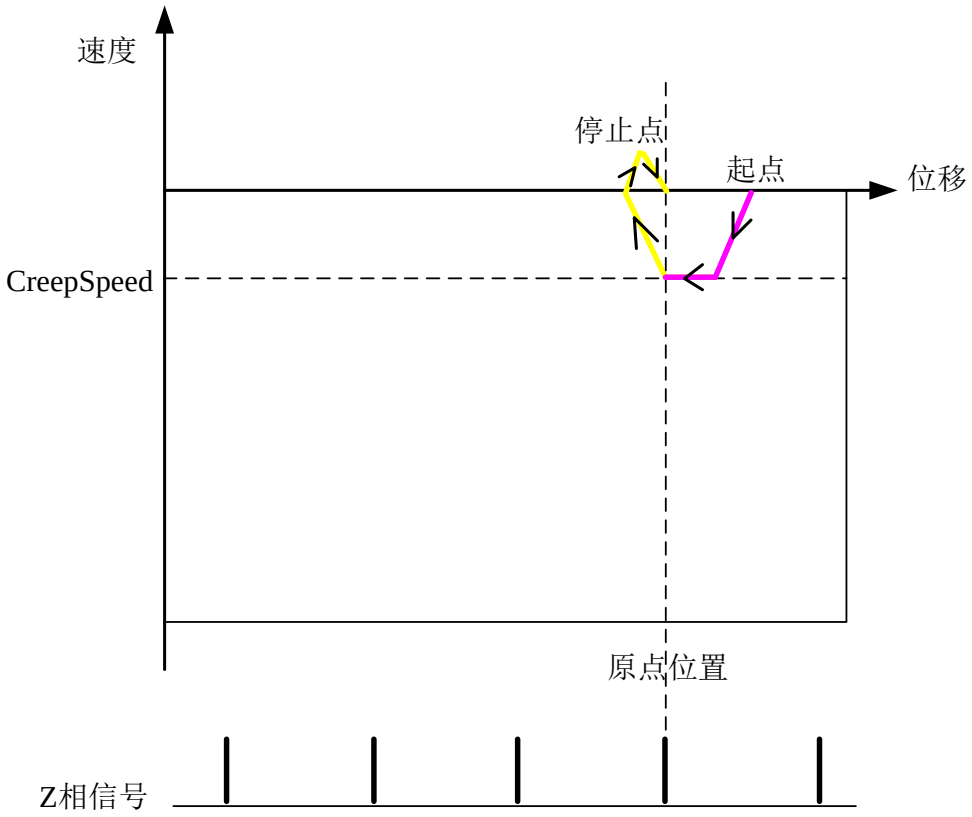


图 31 模式 3 原点回归

（紫色、黄色 对应回归阶段 1、2）

示例程序如下（ST 语言）：

```
    Axis0.Speed := 1000;              (*设定原点回归速度(阶段 1 速度)*)
    Axis0.CreepSpeed := 100;          (*设定爬行速度(阶段 3 速度)*)
    Axis0.Accel := 2000;              (*设定加速度*)
    Axis0.Decel := 2000;              (*设定减速度*)
    HMC_Home(Axis0.AxisID, 3, 0);     (*向 Axis0 轴投送原点回归指令(模式 3,使用 Axis0 轴 Z
相通道)*)
```

（5）模式 4 原点回归（ucHomeMode=4）

使用 DI 及本轴 Z 相信号正向定位原点。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 设定的速度反向运动（阶段 3），遇到 DI 下降沿后且遇到 Z 相为 1 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）。

特殊情形：如果一开始 DI 为 1，则直接进入阶段 3。

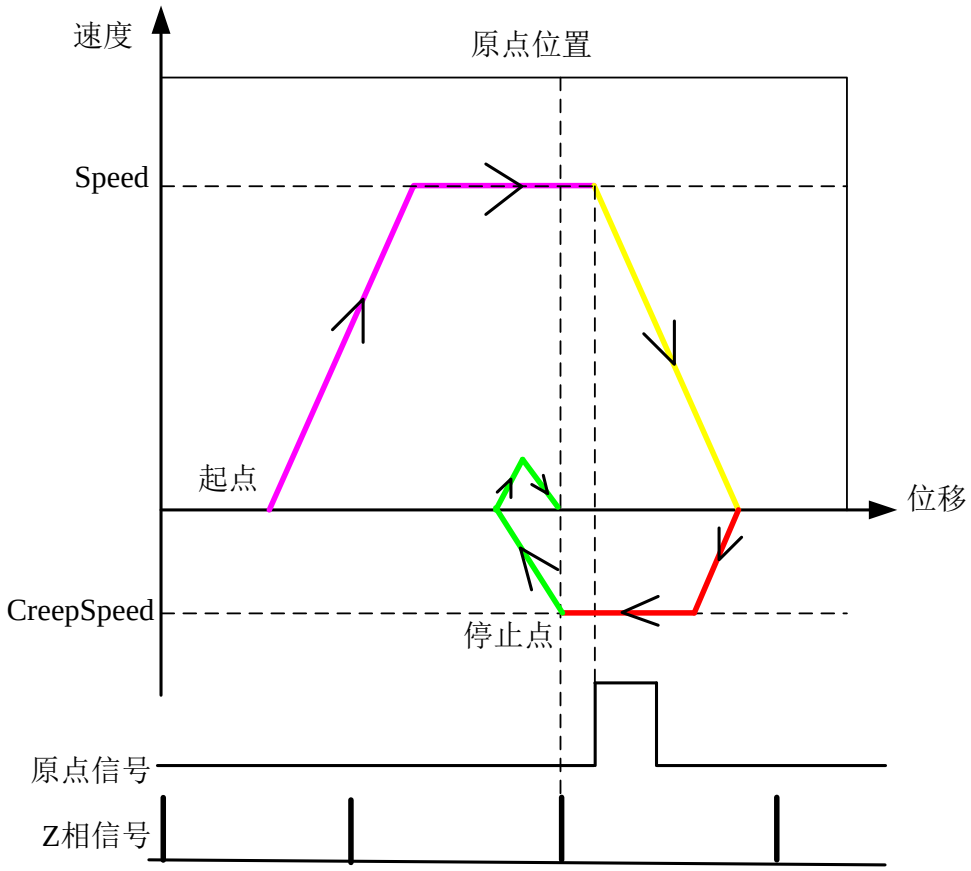


图 32 模式 4 原点回归

（紫色、黄色、红色、绿色 对应回归阶段 1、2、3、4）

示例程序如下（ST 语言）：

```
    Axis0.Speed := 1000;              (*设定原点回归速度(阶段 1 速度)*)
    Axis0.CreepSpeed := 100;          (*设定爬行速度(阶段 3 速度)*)
    Axis0.Accel := 2000;              (*设定加速度*)
    Axis0.Decel := 2000;              (*设定减速度*)
    HMC_Home(Axis0.AxisID, 4, 0);     (*向 Axis0 轴投送原点回归指令(模式 4，使用 0 号快速
DI 及本轴 Z 相通道)*)
```

（6）模式 5 原点回归（ucHomeMode=5）

使用 DI 及本轴 Z 相信号负向定位原点。轴以 Speed 设定的速度负向运动（阶段 1），当遇到 DI 上升沿时减速停止（阶段 2），然后以 CreepSpeed 设定的速度反向运动（阶段 3），遇到 DI 下降沿后且遇到 Z 相为 1 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）。

特殊情形：同模式 4。

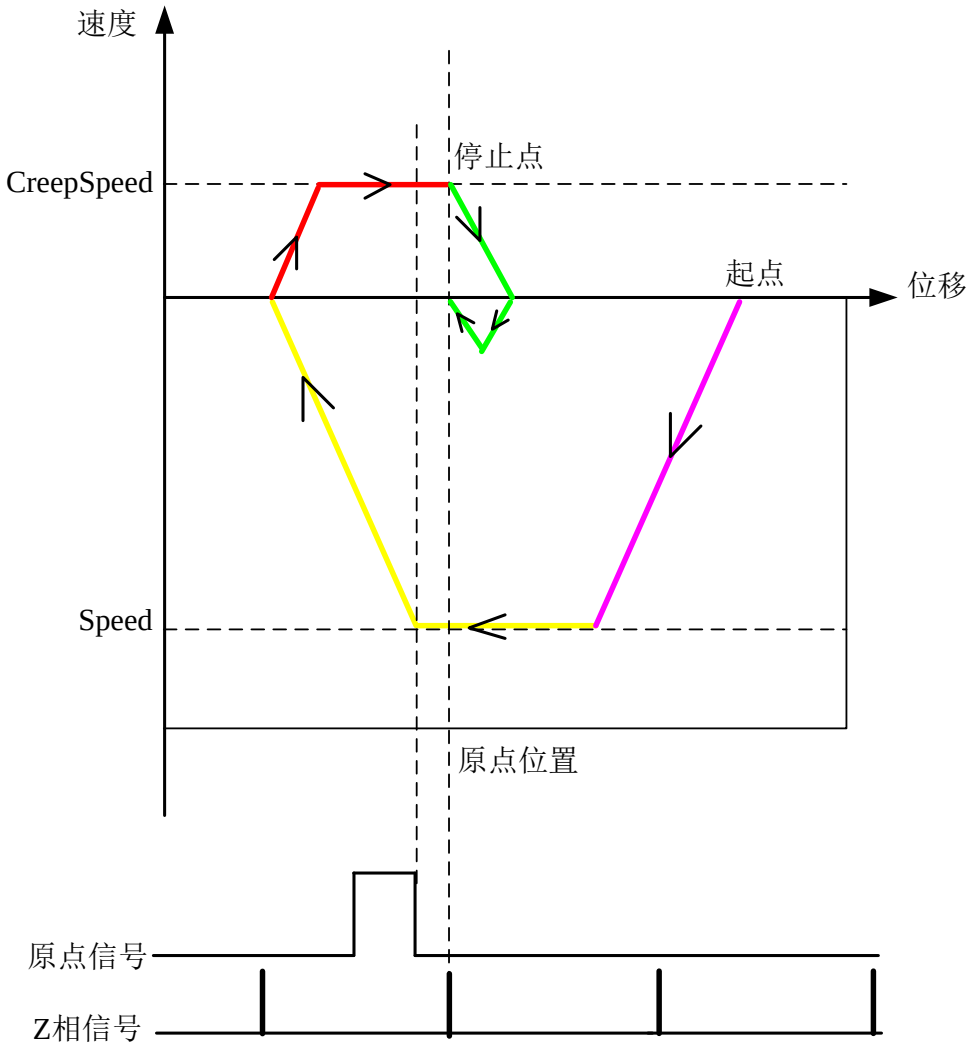


图 33 模式 5 原点回归

（紫色、黄色、红色、绿色 对应回归阶段 1、2、3、4）

示例程序如下（ST 语言）：

```
    Axis0.Speed := 1000;                (*设定原点回归速度(阶段 1 速度)*)
    Axis0.CreepSpeed := 100;            (*设定爬行速度(阶段 3 速度)*)
    Axis0.Accel := 2000;                (*设定加速度*)
    Axis0.Decel := 2000;                (*设定减速度*)
    HMC_Home(Axis0.AxisID, 5, 0);       (*向 Axis0 轴投送原点回归指令(模式 5，使用 0 号快速
DI 及本轴 Z 相通道)*)
```

4.2.1.2 HMC_HomeEx（高级原点回归）

- 函数原型

  UDINT HMC_HomeEx(USINT ucAxisID,

USINT ucHomeMode,

USINT ucDiChl,

USINT ucCaptureChl)

■ 功能

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 DI 通道，Z 相信号从原点回归轴 DB9 接口的 Z 相接口获取。

与 HMC_Home 的不同在于：该指令使用高速捕获通道来捕获原点位置，原点精确位置的获得通过硬件完成，消除了原点信号触发时刻与原点回归程序处理时刻的延时所引入的偏差，因此更加精准。

■ 参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucHomeMode	USINT	0：仅使用 DI 信号正向定位原点 1：仅使用 DI 信号负向定位原点 2：仅使用本轴 Z 相信号正向定位原点 3：仅使用本轴 Z 相信号负向定位原点 4：使用 DI 及本轴 Z 相信号正向定位原点 5：使用 DI 及本轴 Z 相信号负向定位原点 注：详见应用示例。
ucDiChl	USINT	原点回归时使用的快速 DI 通道号 MC1008&MC1004&MC1002R&MC1002E(0~11) MC1004L（0~3）
ucCaptureChl	USINT	原点回归时使用的高速捕获通道号

■ 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

■ 指令类型

轴运动缓存指令。

■ 注意事项

（1）对于 RTEX 和 EtherCAT 总线轴类型，由于总线类型的驱动器再生脉冲口未提供 Z 相信号输出，所以涉及到 Z 相信号的原点回归功能不可用该指令。

■ 应用举例

参见 HMC_Home 示例。

4.2.1.3 HMC_DefOrigin（设置原点）

■ 函数原型

UDINT HMC_DefOrigin(USINT ucAxisID, LREAL dMpos)

- 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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dMpos	LREAL	将会被定义为原点的位置（当前坐标系下）

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其他值	UDINT	错误码

- 指令类型

非轴运动缓存指令。

- 应用举例

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

4.2.1.4 HMC_DefPos（设置当前位置）

- 函数原型

UDINT HMC_ DefPos (USINT ucAxisID, LREAL dMpos)

- 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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dMpos	LREAL	当前位置指定值

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其他值	UDINT	错误码

- 指令类型

非轴运动缓存指令。

- 应用举例

（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*)
```

4.2.2 单向运动

4.2.2.1 HMC_Forward（正向运动）

- 函数原型

  UDINT HMC_Forward(USINT ucAxisID)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 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。

- 应用举例

  示例 1：

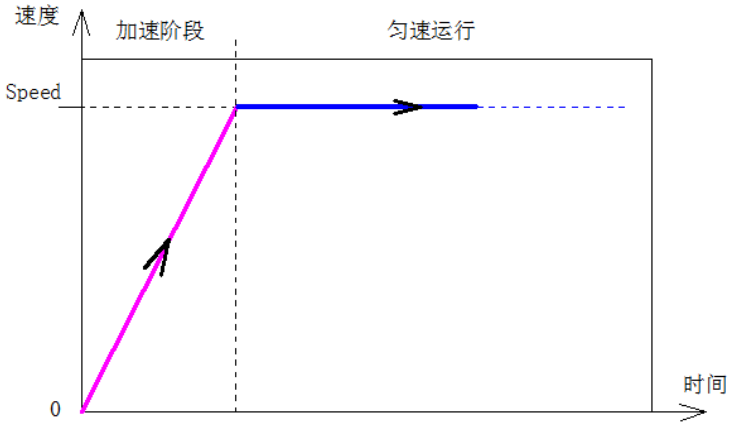


图 34 HMC_Forward

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*)
```

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*)
```

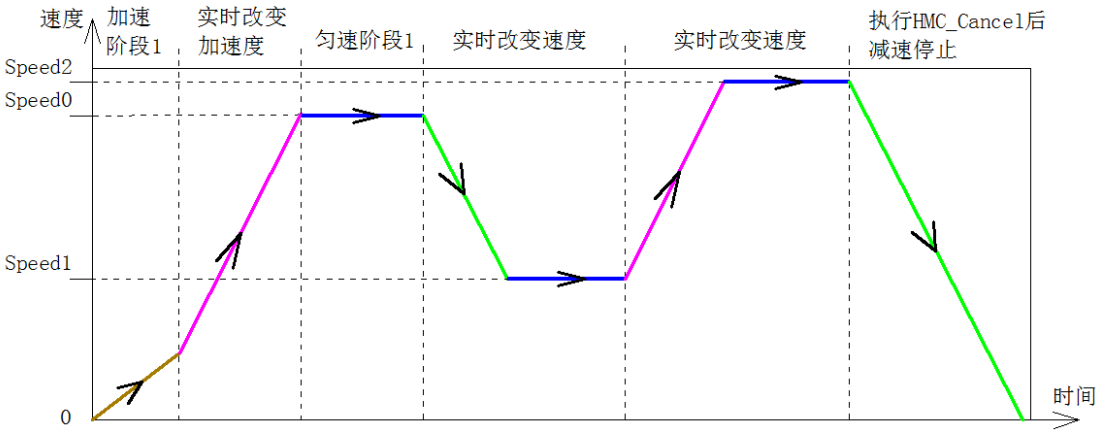


图 35 HMC_Forward 运动中实时调整速度、加速度

4.2.2.2 HMC_Reverse（反向运动）

- 函数原型

  UDINT HMC_ Reverse(USINT ucAxisID)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 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。

- 应用举例

  参见 HMC_Forward。

4.2.2.3 HMC_ForwardEx（高级正向运动）

- 函数原型

  UDINT HMC_ForwardEx(USINT ucAxisID, LREAL dSpeed)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dSpeed	LREAL	执行该运动时的目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 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。

- 应用举例

  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）*)
```

  示例 2：实时修改速度、加减速

  可参见 HMC_Forward 中示例。

- 不支持的模块版本

  MC1008-A01

4.2.2.4 HMC_ReverseEx（高级负向运动）

- 函数原型

  UDINT HMC_ReverseEx(USINT ucAxisID, LREAL dSpeed)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dSpeed	LREAL	执行该运动时的目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 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。

- 应用举例

  参见 HMC_ForwardEx。

- 不支持的模块版本

  MC1008-A01

4.2.3 点位运动

4.2.3.1 HMC_Move（相对运动）

- 函数原型

  UDINT HMC_Move(USINT ucAxisID, LREAL dDistance)

- 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，详见章节 3.5.10 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。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dDistance	LREAL	增量位移（正负均可）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 应用举例

  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。

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

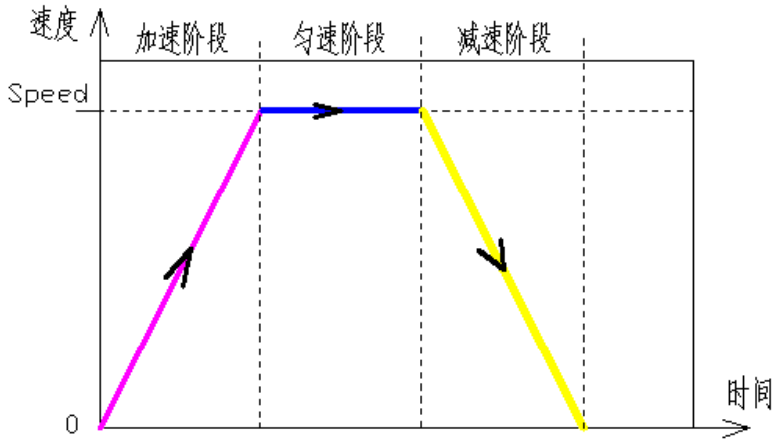


图 36 梯形加减速曲线

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*)
```

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。

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

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*)
```

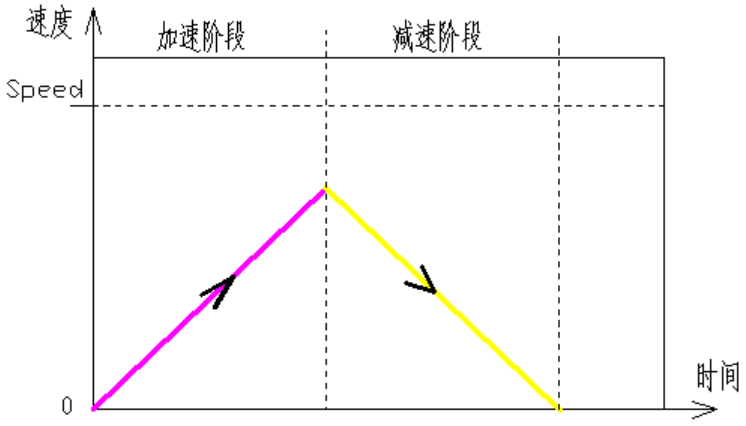


图 37 三角形加减速曲线

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*)
```

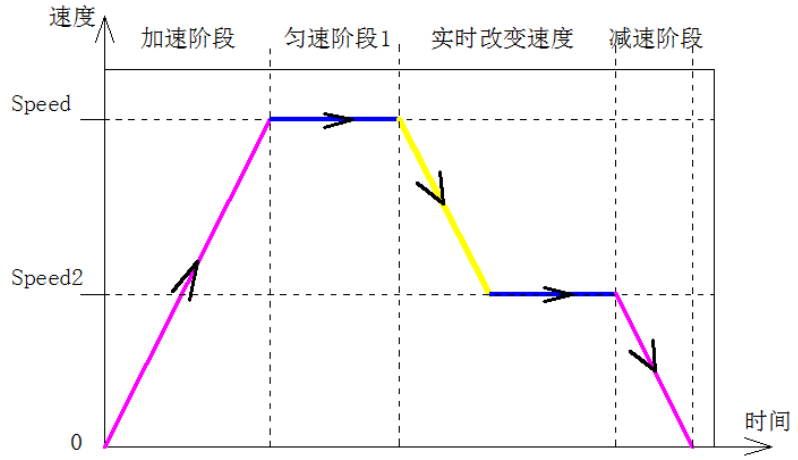


图 38 运动中实时修改速度参数

4.2.3.2 HMC_MoveAbs（绝对运动）

- 函数原型

  UDINT HMC_MoveAbs(USINT ucAxisID, LREAL dDistance)

- 功能说明

  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。

  相比而言，HMC_MoveAbs 在绝对坐标下使用较为方便，HMC_Move 适合在增量坐标下使用。

  详细内容可参考 HMC_Move 中功能说明部分。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dDistance	LREAL	目标位置

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 应用举例

  这里重点举例说明 HMC_MoveAbs 与 HMC_Move 的区别，其它方面的示例可参见 HMC_Move。

  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*)
```

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。

4.2.3.3 HMC_MoveEx（高级相对运动）

- 函数原型

  UDINT HMC_MoveEx(USINT ucAxisID, LREAL dDistance, LREAL dSpeed)

- 功能说明

  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 中功能说明部分。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dDistance	LREAL	位移（正负均可）
dSpeed	LREAL	执行该运动时的目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 应用举例

  示例程序如下（ST 语言），其余示例可参见 HMC_Move。

```
    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）*)
```

- 不支持的模块版本

  MC1008-A01

4.2.3.4 HMC_MoveAbsEx（高级绝对运动）

- 函数原型

  UDINT HMC_MoveAbsEx(USINT ucAxisID, LREAL dDistance, LREAL dSpeed)

- 功能说明

  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。

  详细内容可参考 HMC_MoveAbs 中功能说明部分。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dDistance	LREAL	目标位置
dSpeed	LREAL	执行该运动时的目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 应用举例

  参见 HMC_MoveAbs。

- 不支持的模块版本

  MC1008-A01

4.2.4 指令修正

4.2.4.1 HMC_Cancel（撤销指令）

- 函数原型

  UDINT HMC_Cancel(USINT ucAxisID, USINT ucCancelMode)

- 功能说明

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。

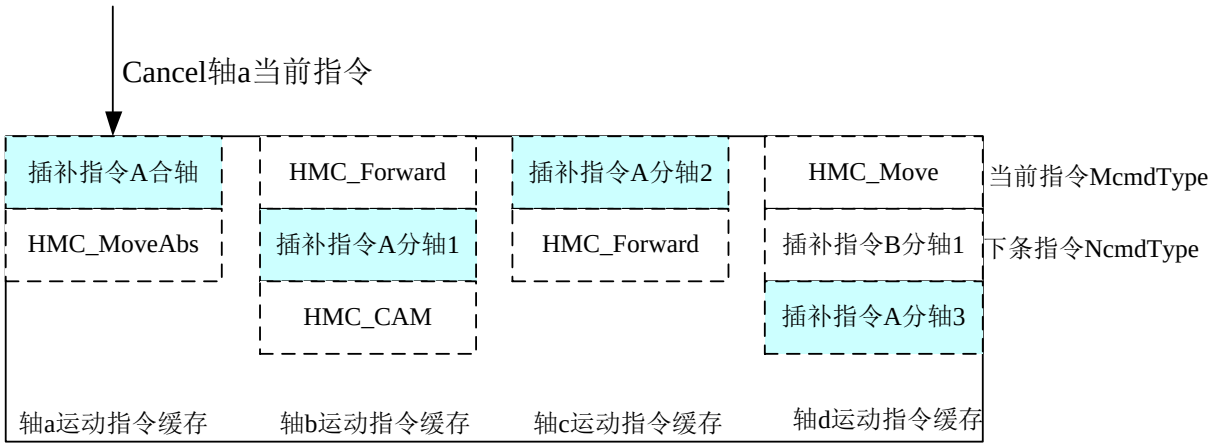


图 39 运动指令缓存状态示意图

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：

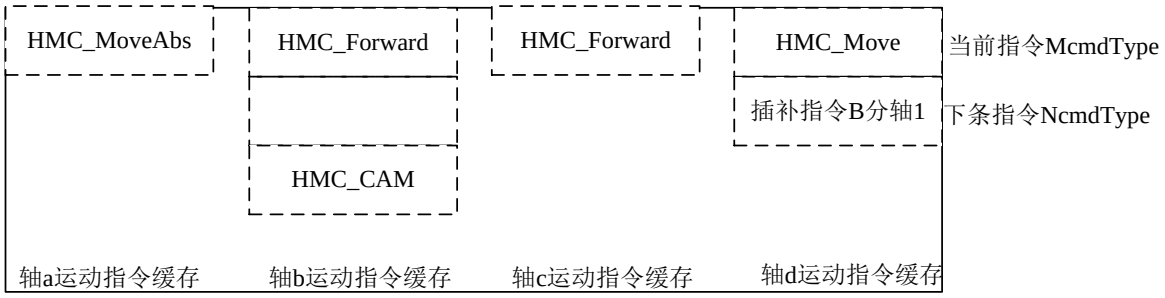


图 40 运动指令缓存状态示意图

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。

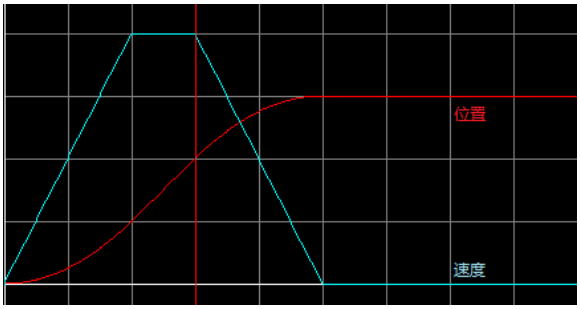


图 41 运动轨迹示意图

（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：

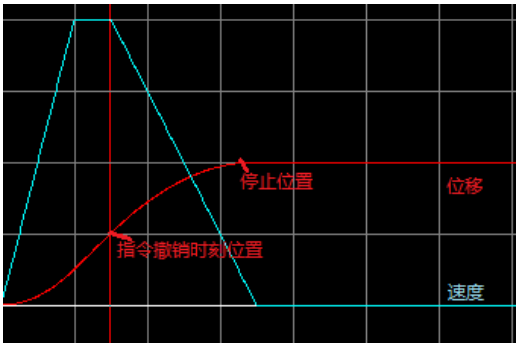


图 42 运动轨迹示意图

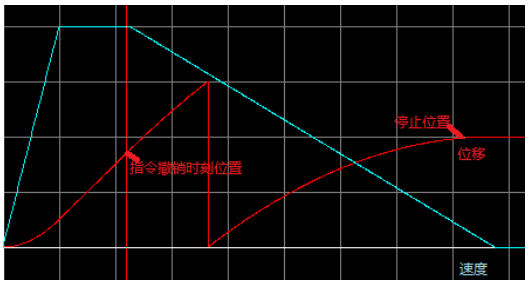


图 43 运动轨迹示意图

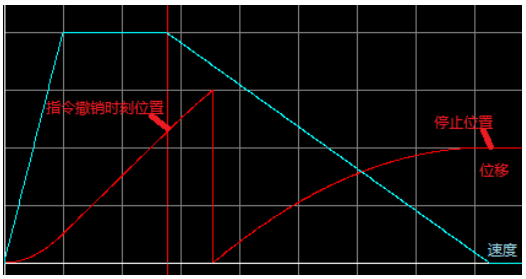


图 44 运动轨迹示意图

（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。

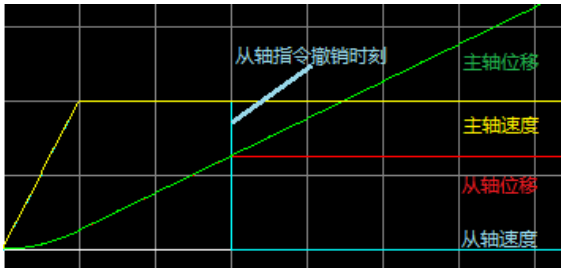


图 45 运动轨迹示意图

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucCancelMode	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

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其他值	UDINT	错误码

- 指令类型

非轴运动缓存指令

- 应用举例

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);
```

4.2.4.2 HMC_MoveModify（目标位置变更）

- 函数原型

UDINT HMC_MoveModify (USINT ucAxisID, LREAL dDpos)

- 功能说明

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。

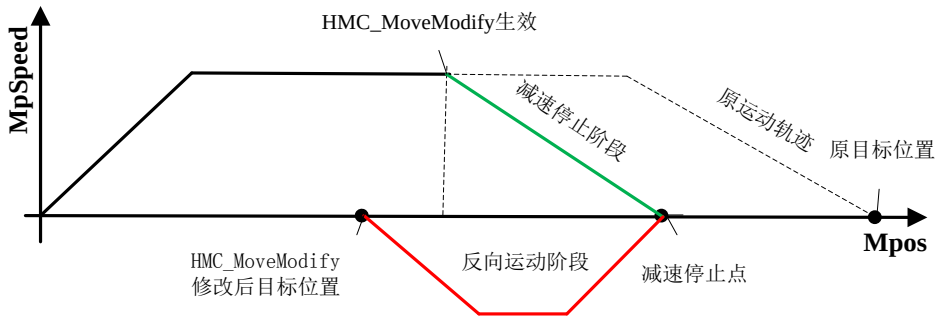


图 46 运动示意图

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。

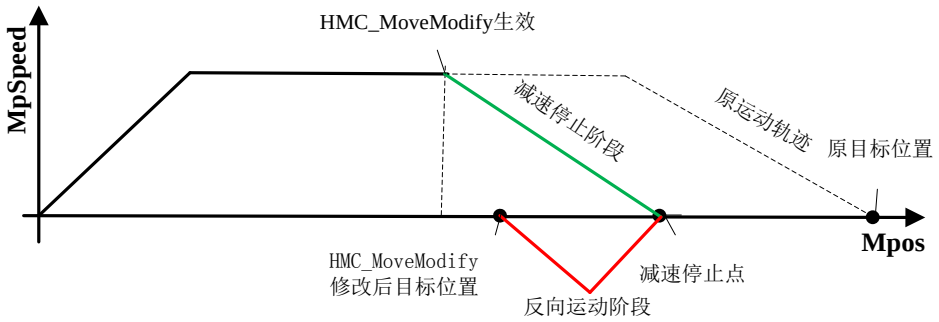


图 47 运动示意图

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。

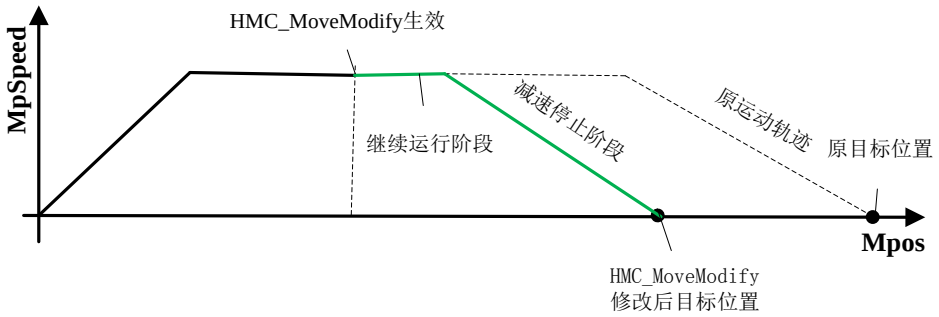


图 48 设定位置在原目标位置之前

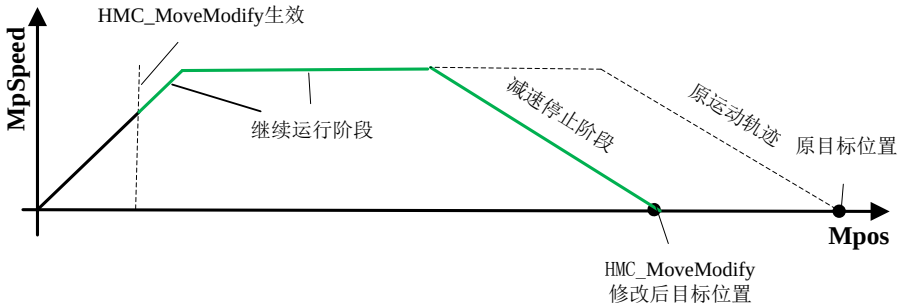


图 49　设定位置在原目标位置之前（加速阶段执行 HMC_MoveModify，存在匀速段）

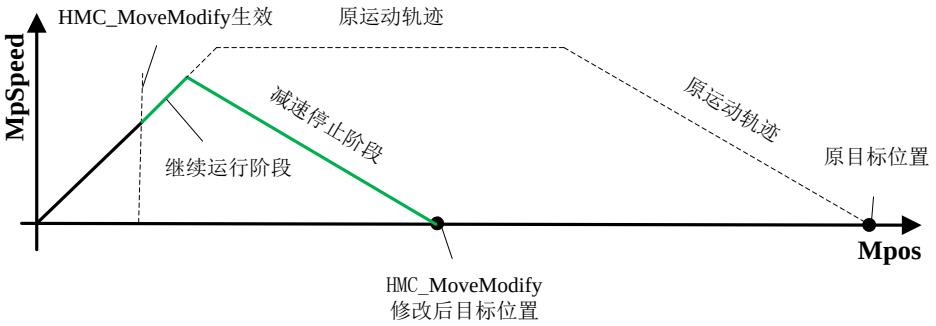


图 50　设定位置在原目标位置之前（加速阶段执行 HMC_MoveModify，无匀速段）

（b）HMC_MoveModify 所指定的终点位置在原目标位置之后。

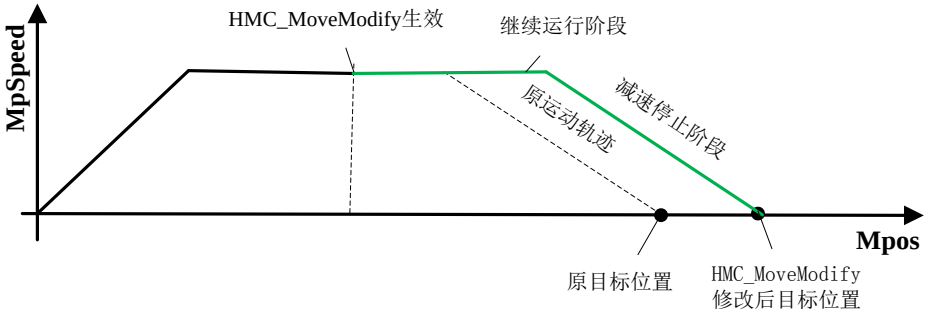


图 51　设定位置在原目标位置之后（匀速阶段执行 HMC_MoveModify）

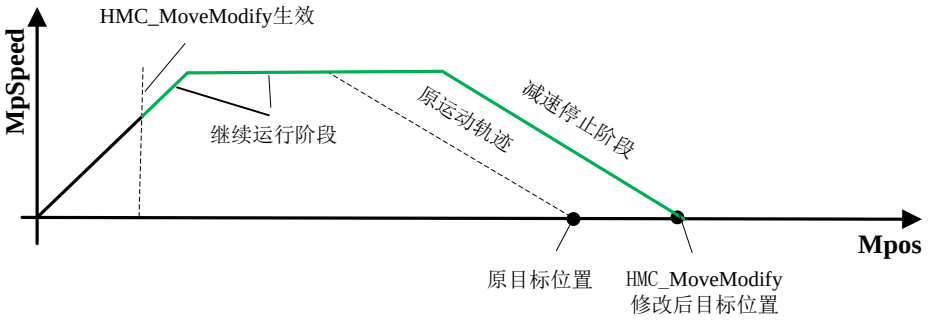


图 52 设定位置在原目标位置之后（加速阶段执行 HMC_MoveModify，存在匀速段）

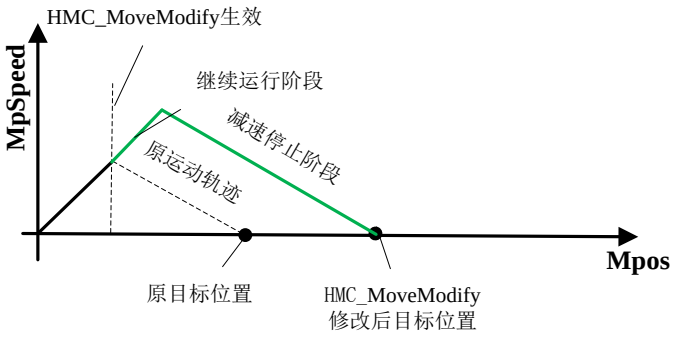


图 53 设定位置在原目标位置之前（加速阶段执行 HMC_MoveModify，无匀速段）

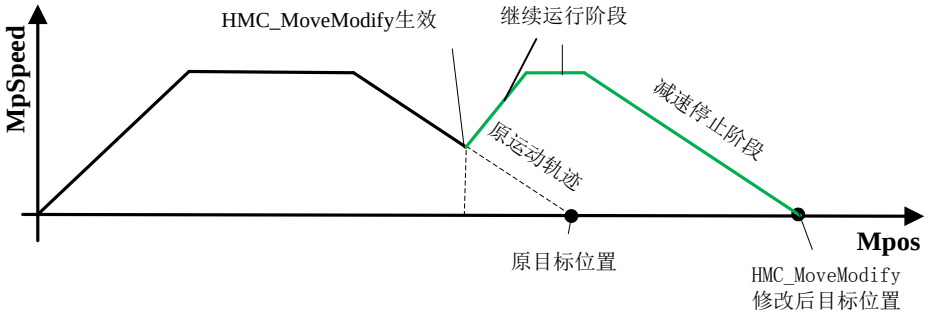


图 54 设定位置在原目标位置之后（减速阶段执行 HMC_MoveModify）

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)
```

- 输入参数

输入参数	类型	参数描述

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dDpos	LREAL	修改当前指令的位移

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFE	UDINT	当前指令不是 Move 或 MoveAbs，忽略本次修改申请
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

- 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。

- 应用举例：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。

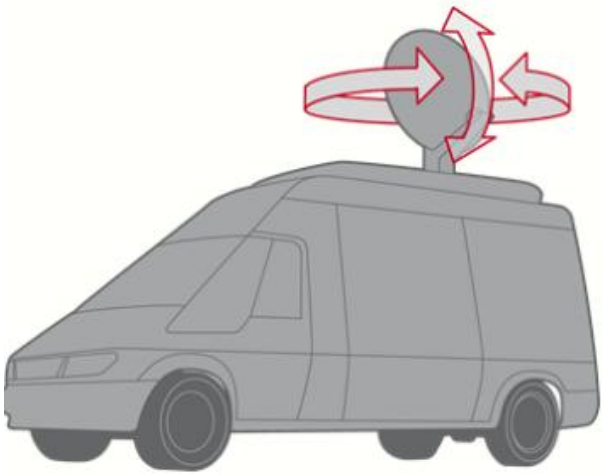


图 55 卫星信号接收车示意图

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
```

- 不支持的模块版本

  MC1008-A01

4.2.4.3 HMC_MoveModifyEx（高级目标位置变更）

- 函数原型

  UDINT HMC_MoveModifyEx(USINT ucAxisID, LREAL dDpos, USINT ucModifyMode)

- 功能说明

  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。

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。

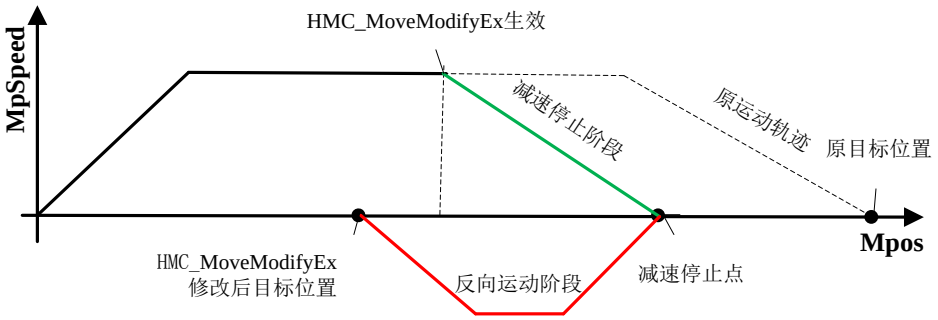


图 56 运动示意图

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。

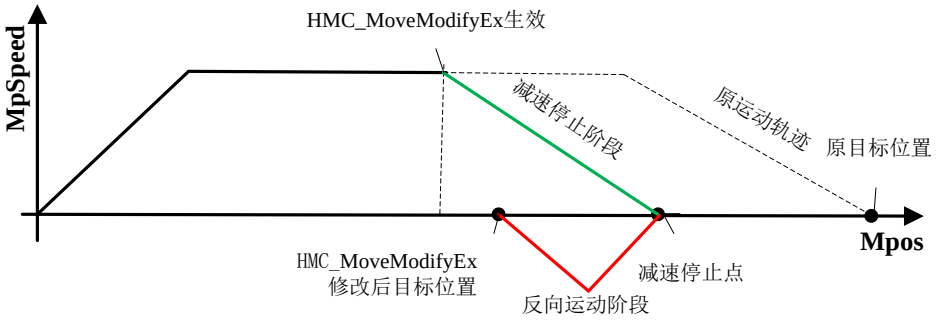


图 57 运动示意图

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。

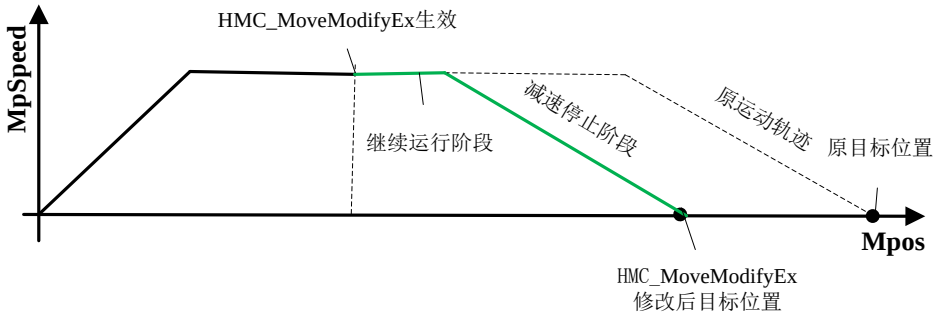


图 58 设定位置在原目标位置之前（匀速阶段执行 HMC_MoveModifyEx）

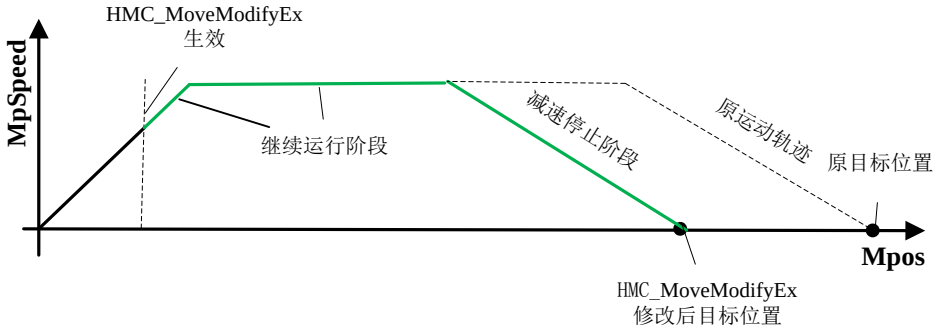


图 59 设定位置在原目标位置之前（加速阶段执行 HMC_MoveModifyEx，存在匀速段）

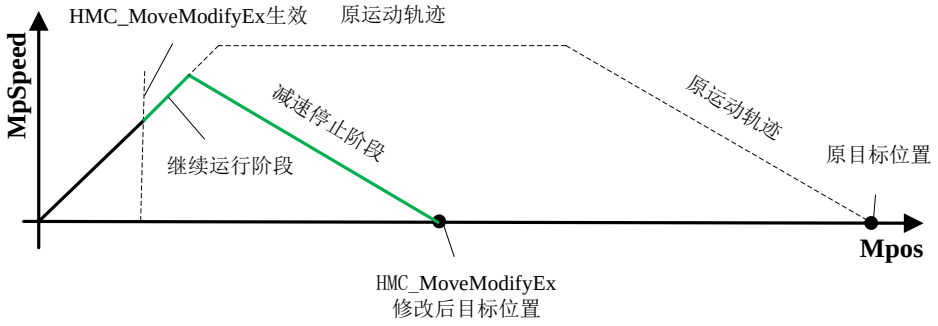


图 60　设定位置在原目标位置之前（加速阶段执行 HMC_MoveModifyEx，无匀速段）

（b）HMC_MoveModifyEx 所指定的终点位置在原目标位置之后。

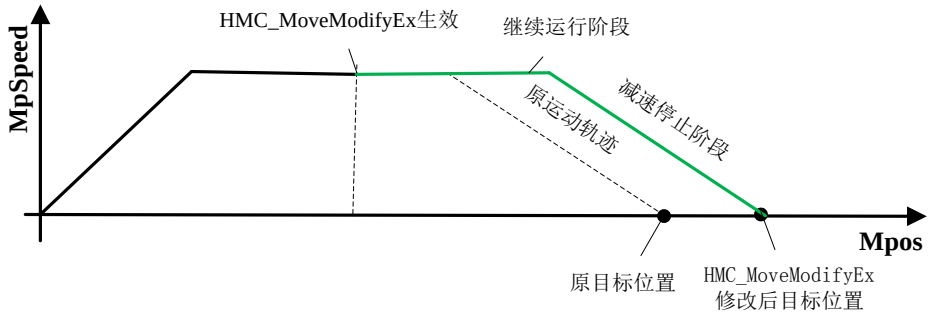


图 61　设定位置在原目标位置之后（匀速阶段执行 HMC_MoveModifyEx）

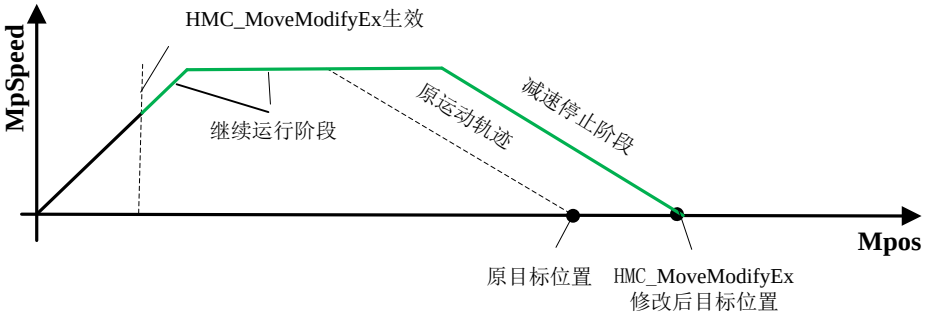


图 62　设定位置在原目标位置之后（加速阶段执行 HMC_MoveModifyEx，存在匀速段）

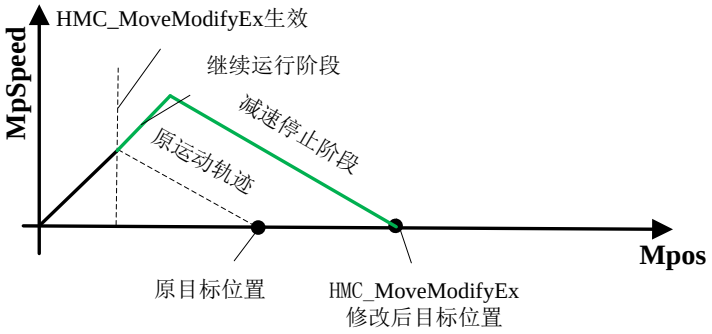


图 63 设定位置在原目标位置之前（加速阶段执行 HMC_MoveModifyEx，无匀速段）

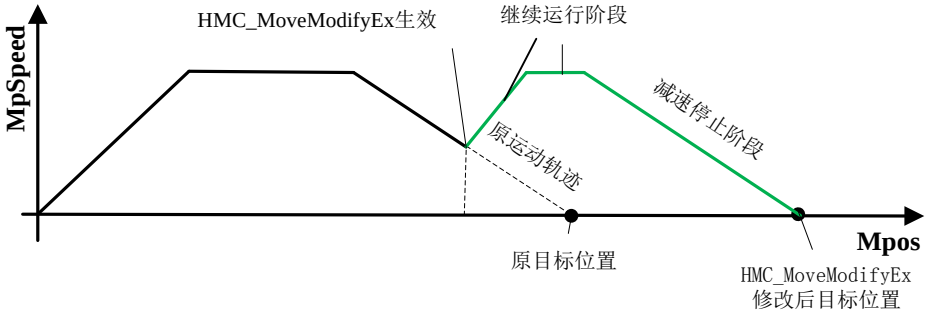


图 64 设定位置在原目标位置之后（减速阶段执行 HMC_MoveModifyEx）

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 *)
```

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	待修改轴轴号
dDpos	LREAL	修改当前指令的位移
ucModifyMode	USINT	0：绝对位置修改模式 1：增量位置修改模式

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFE	UDINT	当前指令不是 Move 或 MoveAbs，忽略本次修改申请
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

- 补充说明

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

- 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（图 65 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。

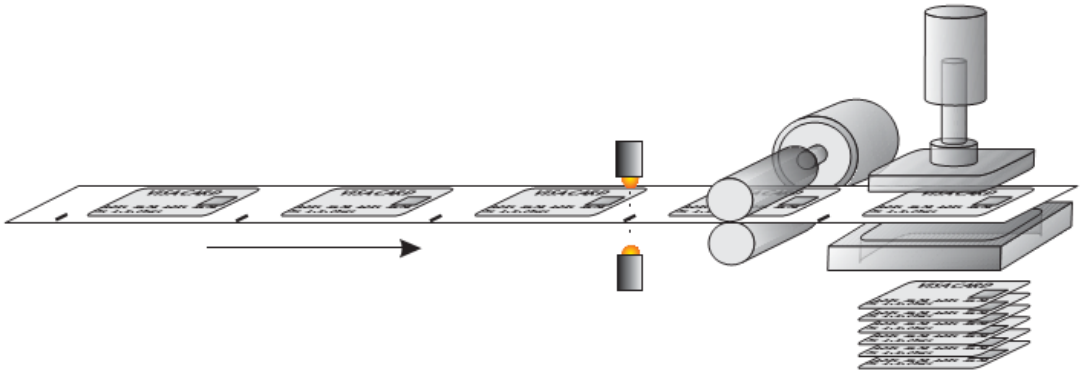


图 65 传输示意图

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：

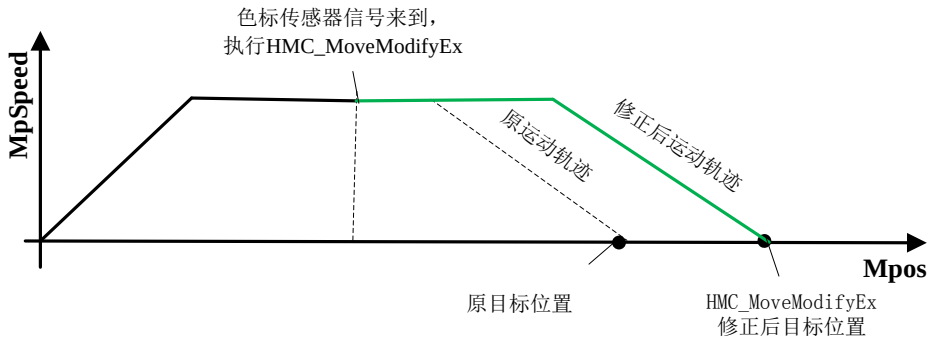


图 66 传送带轨迹示意图

- 不支持的模块版本

  MC1008-A01

4.3 多轴运动指令

4.3.1 线性插补

4.3.1.1 HMC_Move2D(2 轴直线插补)

- 函数原型

  UDINT HMC_Move2D (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  LREAL dDistanceX,

  LREAL dDistanceY)

- 功能说明

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，详见章节 4.7.2 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。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
dDistanceX	LREAL	轴 X 运动距离
dDistanceY	LREAL	轴 Y 运动距离

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  以一个平面切割轮廓为三角形的物料为例，三角形如下：

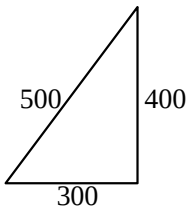


图 67 三角形轮廓

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：

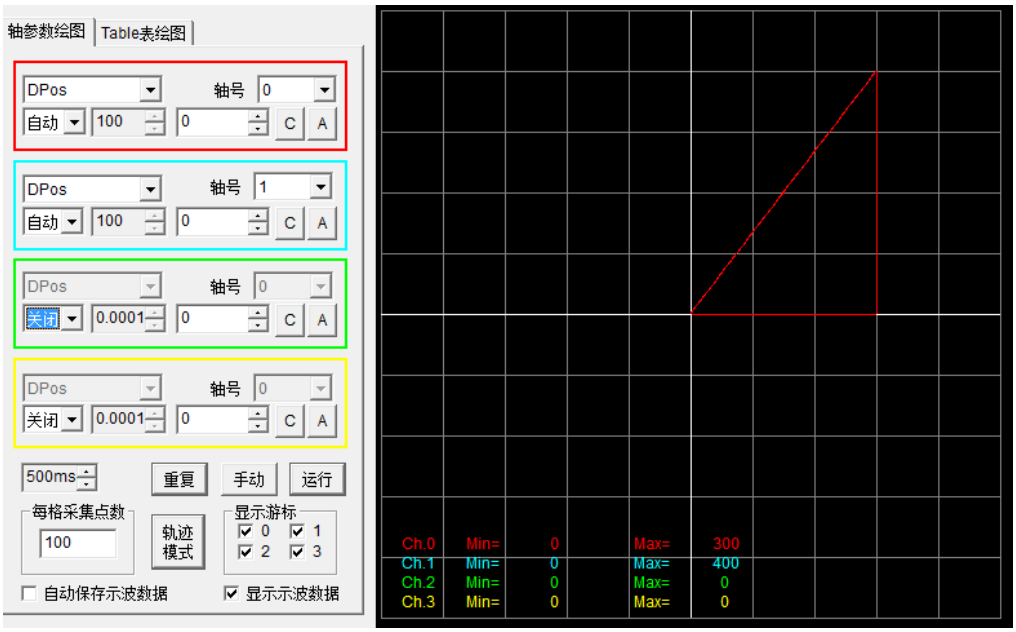


图 68 2 轴直线插补实现三角形轮廓轨迹

4.3.1.2 HMC_Move2DEx(高级 2 轴直线插补)

- 函数原型

  UDINT HMC_Move2DEx (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  LREAL dDistanceX,

  LREAL dDistanceY,

  LREAL dSpeed)

- 功能说明

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。

详细内容可参考章节 4.3.1.1 HMC_Move2D(2 轴直线插补)中功能说明部分。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
dDistanceX	LREAL	轴 X 运动距离

输入参数	类型	参数描述
dDistanceY	LREAL	轴 Y 运动距离
dSpeed	LREAL	合轴目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

```
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）*)
```

- 不支持的模块版本

  MC1008-A01

4.3.1.3 HMC_MoveAbs2D(2 轴绝对值直线插补)

- 函数原型

  UDINT HMC_MoveAbs2D (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  LREAL dDposX,

  LREAL dDposY)

- 功能说明

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。

详细内容可参考章节 4.3.1.1 HMC_Move2D(2 轴直线插补)中功能说明部分。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
dDposX	LREAL	轴 X 目标位置
dDposY	LREAL	轴 Y 目标位置

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  以一个平面切割轮廓为三角形的物料为例，三角形如下：

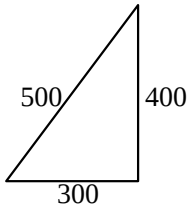


图 69 三角形轮廓

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：

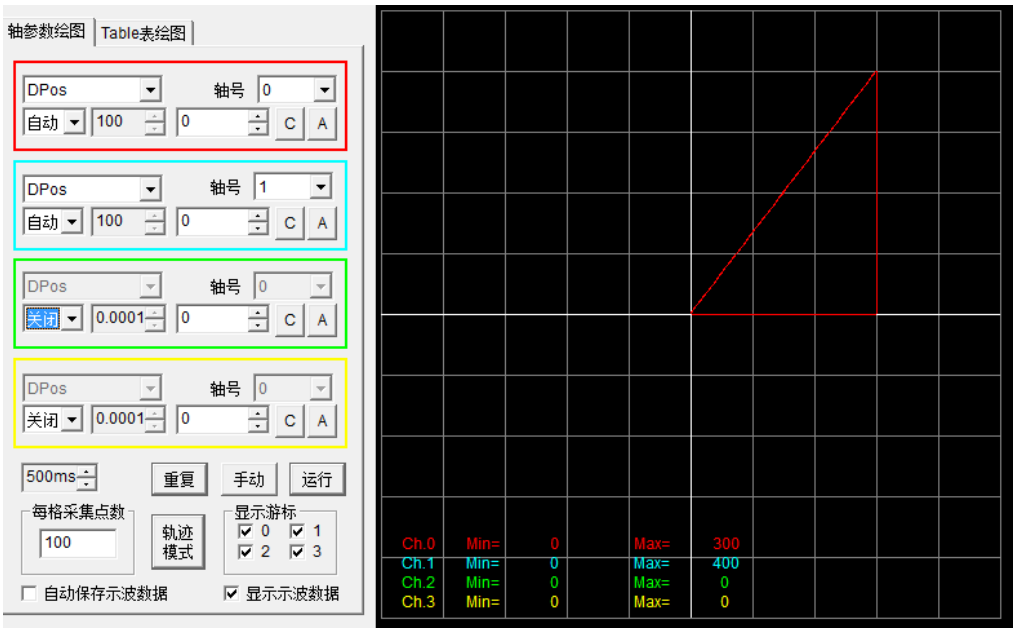


图 70　2 轴直线插补实现三角形轮廓轨迹

- 不支持的模块版本

  MC1008-A01

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

- 函数原型

  UDINT HMC_MoveAbs2DEx (USINT ucAxisID

  USINT ucAxisX,

  USINT ucAxisY,

  LREAL dDposX,

  LREAL dDposY,

  LREAL dSpeed)

- 功能说明

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。

详细内容可参考章节 4.3.1.1 HMC_Move2D(2 轴直线插补)中功能说明部分。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号

输入参数	类型	参数描述
dDposX	LREAL	轴 X 目标位置
dDposY	LREAL	轴 Y 目标位置
dSpeed	LREAL	合轴目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  参见 HMC_MoveAbs2D。

- 不支持的模块版本

  MC1008-A01

4.3.1.5 HMC_Move3D(3 轴直线插补)

- 函数原型

  UDINT HMC_Move3D (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  USINT ucAxisZ,

  LREAL dDistanceX,

  LREAL dDistanceY,

  LREAL dDistanceZ)

- 功能说明

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 4.7.2 HMC_HwLimitConfig（设置硬限位开关）。

不支持 Merge 功能。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
dDistanceX	LREAL	轴 X 运动距离
dDistanceY	LREAL	轴 Y 运动距离
dDistanceZ	LREAL	轴 Z 运动距离

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

轴运动缓存指令

- 应用举例

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。

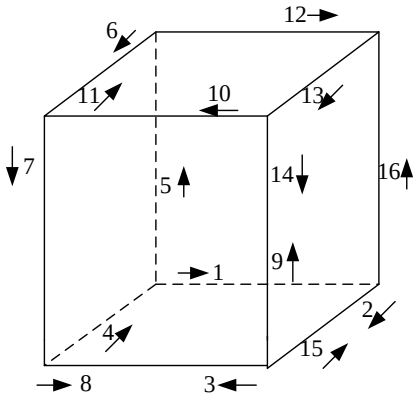


图 71 正方体示意图

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：

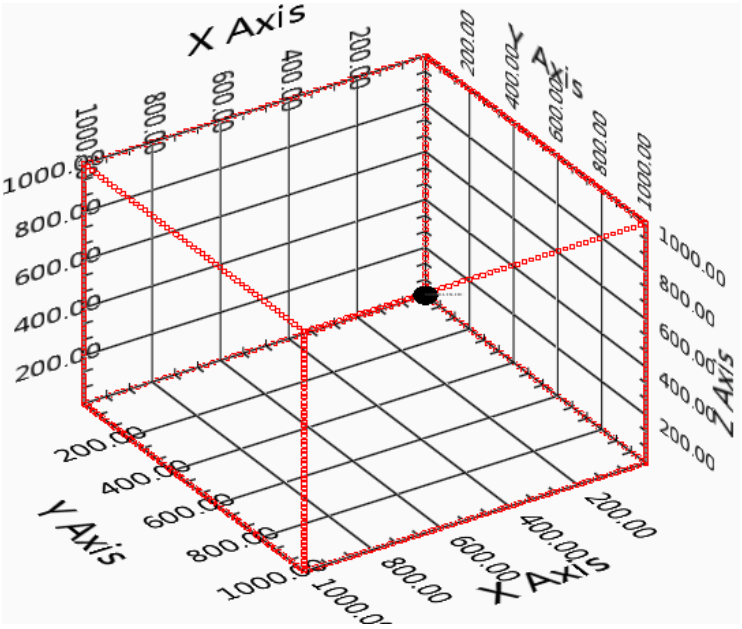


图 72 三维示意图

4.3.1.6 HMC_Move3DEx(高级 3 轴直线插补)

- 函数原型

  UDINT HMC_Move3DEx (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  USINT ucAxisZ,

LREAL dDistanceX,

LREAL dDistanceY,

LREAL dDistanceZ,

LREAL dSpeed)

- 功能说明

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。

详细内容可参考章节 4.3.1.5 HMC_Move3D(3 轴直线插补)中功能说明部分。

不支持 Merge 功能。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
dDistanceX	LREAL	轴 X 运动距离
dDistanceY	LREAL	轴 Y 运动距离
dDistanceZ	LREAL	轴 Z 运动距离
dSpeed	LREAL	合轴目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

轴运动缓存指令

- 应用举例

```
    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）*)
```

- 不支持的模块版本

MC1008-A01

4.3.1.7 HMC_MoveAbs3D(3 轴绝对值直线插补)

- 函数原型

  UDINT HMC_MoveAbs3D (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  USINT ucAxisZ,

  LREAL dDposX,

  LREAL dDposY,

  LREAL dDposZ)

- 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。

  详细内容可参考章节 4.3.1.5 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。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
dDposX	LREAL	轴 X 目标位置
dDposY	LREAL	轴 Y 目标位置
dDposZ	LREAL	轴 Z 目标位置

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  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。

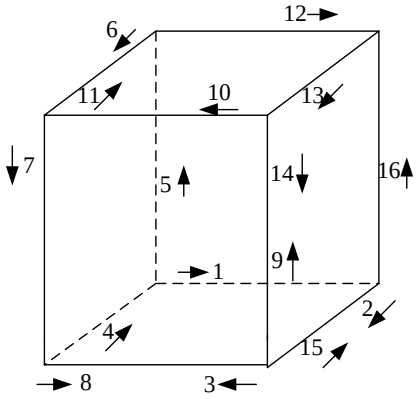


图 73 正方体示意图

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：

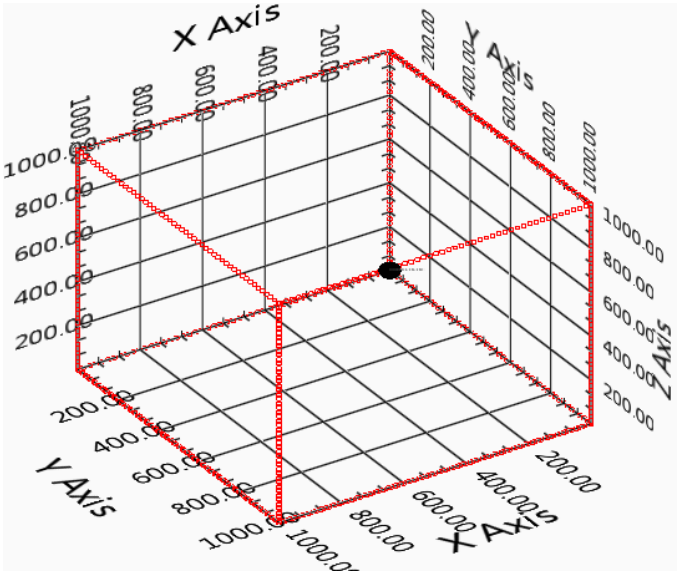


图 74 三维示意图

- 不支持的模块版本

  MC1008-A01

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

- 函数原型

  UDINT HMC_MoveAbs3DEx (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  USINT ucAxisZ,

  LREAL dDposX,

  LREAL dDposY,

  LREAL dDposZ,

  LREAL dSpeed)

- 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。

  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。

  详细内容可参考章节 4.3.1.5 HMC_Move3D(3 轴直线插补)中功能说明部分。

  不支持 Merge 功能。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号

输入参数	类型	参数描述
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
dDposX	LREAL	轴 X 目标位置
dDposY	LREAL	轴 Y 目标位置
dDposZ	LREAL	轴 Z 目标位置
dSpeed	LREAL	合轴目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  参考 HMC_MoveAbs3D

- 不支持的模块版本

  MC1008-A01

4.3.1.9 HMC_MoveND(N 轴直线插补)

- 函数原型

  UDINT HMC_MoveND (USINT ucAxisID,

  POINTER TO USINT pAxis,

  POINTER TO LREAL pDistance,

  USINT ucLineNum,

  USINT ucSelfNum)

- 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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
pAxis	POINTER TO USINT	插补分运动轴号数组首地址
pDistance	POINTER TO LREAL	插补分运动轴运动距离首地址
ucLineNum	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	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

输入参数	类型	参数描述
		取值范围：ucSelfNum>=0

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

```
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*)
```

4.3.1.10 HMC_MoveNDEx(高级 N 轴直线插补)

- 函数原型

  UDINT HMC_MoveNDEx (USINT ucAxisID,

  POINTER TO USINT pAxis,

  POINTER TO LREAL pDistance,

  USINT ucLineNum,

  USINT ucSelfNum,

  LREAL dSpeed)

- 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。

  详细内容可参考章节 4.3.1.9 HMC_MoveND(N 轴直线插补)中功能说明部分。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
pAxis	POINTER TO USINT	插补分运动轴号数组首地址
pDistance	POINTER TO LREAL	插补分运动轴运动距离首地址
ucLineNum	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	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	LREAL	合轴 ucAxisID 的目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  参考 HMC_MoveND。

- 不支持的模块版本

  MC1008-A01

4.3.1.11 HMC_MoveAbsND (N 轴绝对值直线插补)

- 函数原型

  UDINT HMC_MoveAbsND (USINT ucAxisID,

  POINTER TO USINT pAxis,

  POINTER TO LREAL pDpos,

  USINT ucLineNum,

  USINT ucSelfNum)

- 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。

详细内容可参考章节 4.3.1.9 HMC_MoveND(N 轴直线插补)中功能说明部分。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
pAxis	POINTER TO USINT	插补分运动轴号数组首地址
pDpos	POINTER TO LREAL	插补分运动轴目标位置首地址
ucLineNum	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，pDpos 的前 ucLineNum 个元素即为线性插补分轴的目标距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucLineNum>=1
ucSelfNum	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，pDpos 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的目标距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucSelfNum>=0

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  参考 HMC_MoveND

- 不支持的模块版本

  MC1008-A01

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

- 函数原型

  UDINT HMC_MoveAbsNDEx (USINT ucAxisID,
  POINTER TO USINT pAxis,
  POINTER TO LREAL pDpos,
  USINT ucLineNum,
  USINT ucSelfNum,
  LREAL dSpeed)

- 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。

详细内容可参考章节 4.3.1.9 HMC_MoveND(N 轴直线插补)中功能说明部分。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
pAxis	POINTER TO USINT	插补分运动轴号数组首地址
pDpos	POINTER TO LREAL	插补分运动轴目标位置首地址
ucLineNum	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，pDpos 的前 ucLineNum 个元素即为线性插补分轴的目标距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucLineNum>=1
ucSelfNum	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，pDpos 从 ucLineNum 个元素开始往后的 ucSelfNum 个元素即为独立插补分轴的目标距离，从前到后依次与 pAxis 指定的轴号对应 取值范围：ucSelfNum>=0
dSpeed	LREAL	合轴 ucAxisID 的目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

  参考 HMC_MoveND

- 不支持的模块版本

  MC1008-A01

4.3.2 圆弧插补

4.3.2.1 HMC_MoveCircle（圆弧插补）

- 函数原型

  UDINT HMC_ MoveCircle (USINT ucAxisID,

  USINT ucAxisX,

USINT ucAxisY,

USINT ucMode,

LREAL dAx,

LREAL dAy,

LREAL dBx,

LREAL dBy)

■ 功能说明

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。

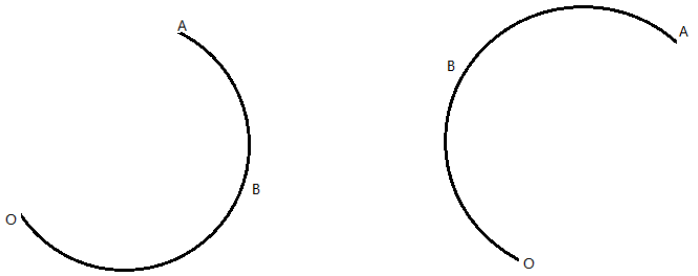


图 75 运动轨迹

（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，如图 76 左图；否则即为圆心到 A 点射线与圆弧的交点为终点，如图 76 右图。如果 A 点为（0,0）则逆时针画一整圆。

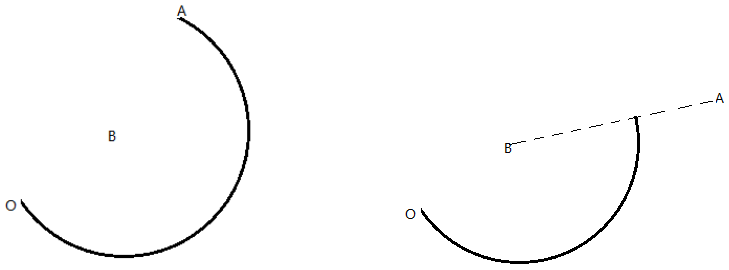


图 76 运动轨迹

（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，如图 77 左图；否则即为圆心到 A 点射线与圆弧的交点为终点，如图 77 右图。如果 A 点为（0,0）则顺时针画一整圆。

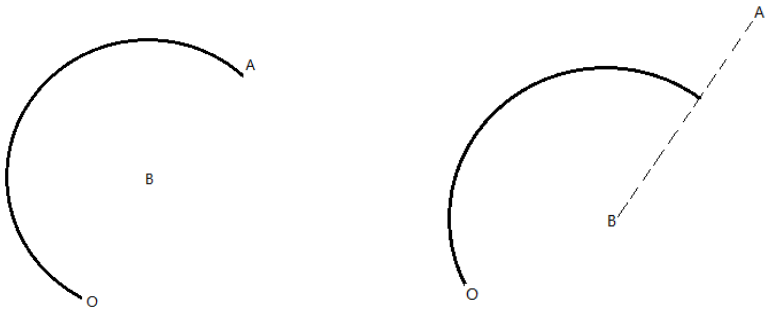


图 77 运动轨迹

支持 Merge 功能。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucMode （O 为当前点）	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	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	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	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	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）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令

- 应用举例

切割如下工件。

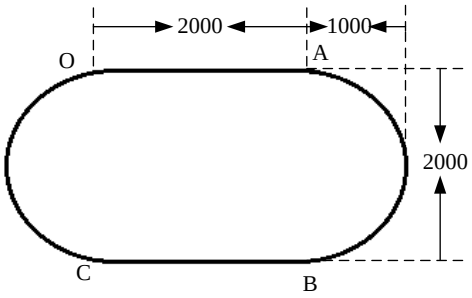


图 78 工件尺寸图

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：

图 79 轨迹示意图

4.3.2.2 HMC_MoveCircleEx（高级圆弧插补）

- 函数原型

  UDINT HMC_MoveCircleEx(USINT ucAxisID,

  USINT ucAxisX,

USINT ucAxisY,

USINT ucMode,

LREAL dAx,

LREAL dAy,

LREAL dBx,

LREAL dBy,

LREAL dSpeed)

- 功能说明

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 中功能说明部分。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucMode （O 为当前点）	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	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	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	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	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	LREAL	合轴目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

轴运动缓存指令

- 应用举例

  参考 HMC_MoveCircle。

- 不支持的模块版本

  MC1008-A01

4.3.2.3 HMC_MoveSpherical（球弧插补）

- 函数原型

  UDINT HMC_MoveSpherical (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  USINT ucAxisZ,

  USINT ucMode,

  LREAL dAx,

  LREAL dAy,

  LREAL dAz,

  LREAL dBx,

  LREAL dBy,

  LREAL dBz)

- 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。

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。

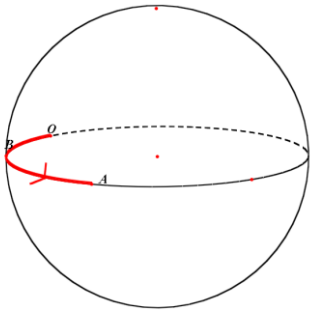


图 80　运动轨迹

（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。

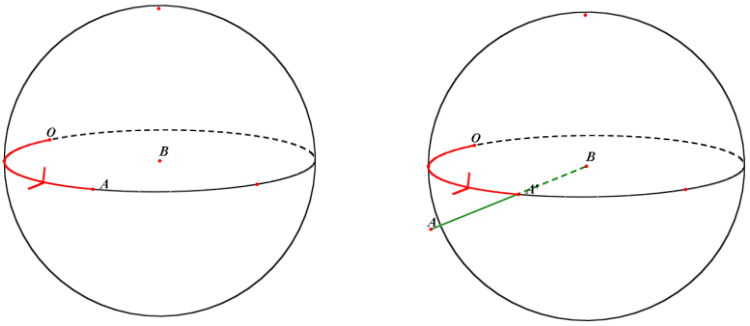


图 81　运动轨迹

（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。

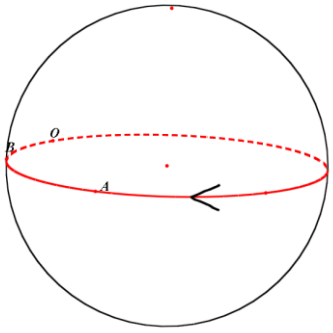


图 82　运动轨迹

（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。

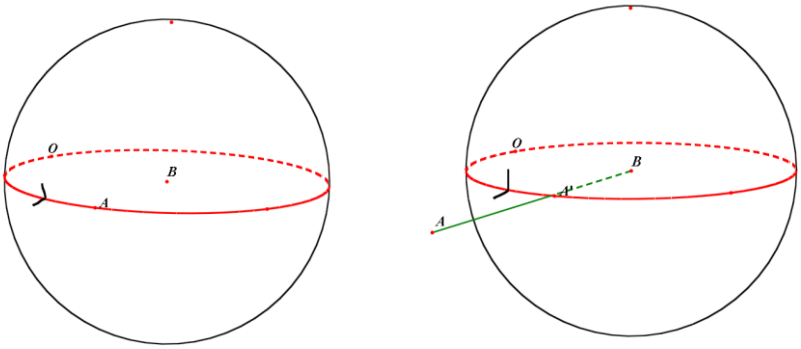


图 83 运动轨迹

■ 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
ucMode （O 为当前点）	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	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	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	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	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	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）

输入参数	类型	参数描述
dBZ	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）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令属性

  非阻塞指令，轴运动缓存指令

- 应用举例

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：

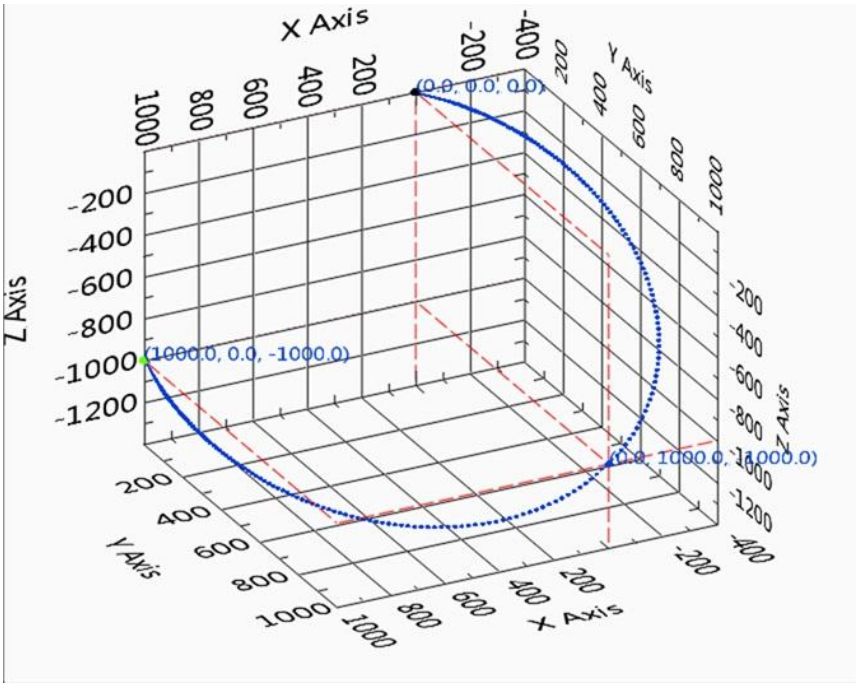


图 84 三维示意图

4.3.2.4 HMC_ MoveSphericalEx（高级球弧插补）

- 函数原型

  UDINT HMC_MoveSphericalEx (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  USINT ucAxisZ,

USINT ucMode,

LREAL dAx,

LREAL dAy,

LREAL dAz,

LREAL dBx,

LREAL dBy,

LREAL dBz,

LREAL dSpeed)

■ 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。

■ 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
ucMode （O 为当前点）	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	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	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	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）

输入参数	类型	参数描述
dBx	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	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	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	LREAL	合轴目标速度

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令属性

  非阻塞指令，轴运动缓存指令

- 应用举例

  参考 HMC_MoveSpherical。

- 不支持的模块版本

  MC1008-A01

4.3.3 螺旋线插补

4.3.3.1 HMC_MoveHelical（螺旋线插补）

- 函数原型

  UDINT HMC_ MoveHelical (USINT ucAxisID,

  USINT ucAxisX,

  USINT ucAxisY,

  USINT ucAxisZ,

  USINT ucMode,

  LREAL dAx,

  LREAL dAy,

  LREAL dBx,

  LREAL dBy,

  LREAL dDistanceZ)

- 功能说明

  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。

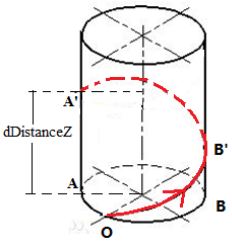


图 85 运动轨迹

（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。

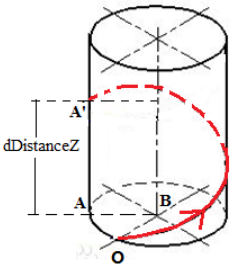


图 86 运动轨迹

（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。

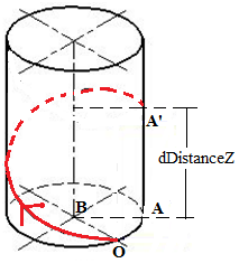


图 87 运动轨迹

（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：

$$dDistance\_AxisID = dCircleLength$$

$$\frac{dSpeed\_AxisZ}{dSpeed\_AxisID} = \frac{dDis\tan ce\_AxisZ}{dDis\tan ce\_AxisID}$$

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。

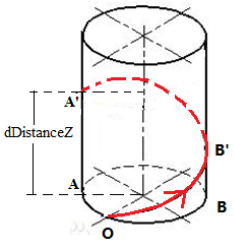


图 88 运动轨迹

（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。

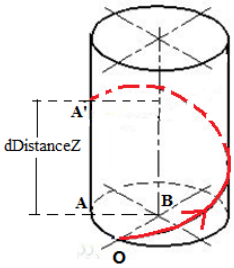


图 89 运动轨迹

（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。

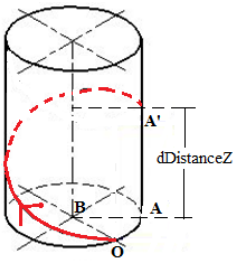


图 90 运动轨迹

■ 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
ucMode （O 为当前点）	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	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	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	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	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	LREAL	Z 轴位移

■ 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

■ 指令属性

非阻塞指令，轴运动缓存指令

■ 应用举例

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：

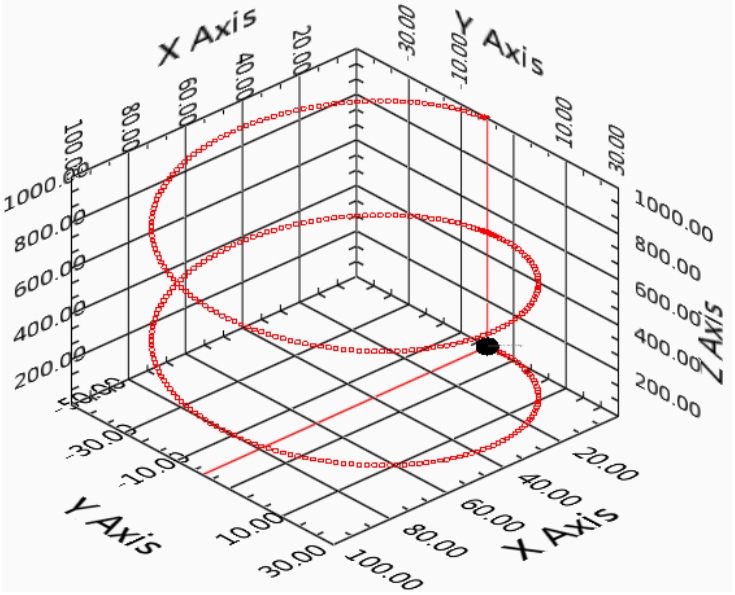


图 91 三维示意图

4.3.3.2 HMC_MoveHelicalEx（高级螺旋线插补）

■ 函数原型

UDINT HMC_ MoveHelicalEx (USINT ucAxisID,

USINT ucAxisX,

USINT ucAxisY,

USINT ucAxisZ,

USINT ucMode,

LREAL dAx,

LREAL dAy,

LREAL dBx,

LREAL dBy,

LREAL dDistanceZ,

LREAL dSpeed)

■ 功能说明

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 中功能说明部分。

■ 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
ucAxisX	USINT	插补轴 X 的轴号
ucAxisY	USINT	插补轴 Y 的轴号
ucAxisZ	USINT	插补轴 Z 的轴号
ucMode （O 为当前点）	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	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	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	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	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	LREAL	Z 轴位移
dSpeed	LREAL	合轴 ucAxisID 的目标速度

■ 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

■ 指令属性

非阻塞指令，轴运动缓存指令

- 应用举例

  参考 HMC_MoveHelical 示例。

- 不支持的模块版本

  MC1008-A01

4.3.4 样条插补

4.3.4.1 HMC_MoveSplineND（样条插补）

- 函数原型

  UDINT HMC_ MoveSplineND (USINT ucAxisID,

  POINTER TO USINT pAxis,

  USINT ucEndPointAxisNum,

  USINT ucFollowAxisNum,

  POINTER TO LREAL pArrPos,

  UDINT uiSplineNum,

  POINTER TO LREAL pArrSplineData,

  POINTER TO LREAL pArrSlenthData,

  POINTER TO LREAL pArrJerkData)

- 功能说明

  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。

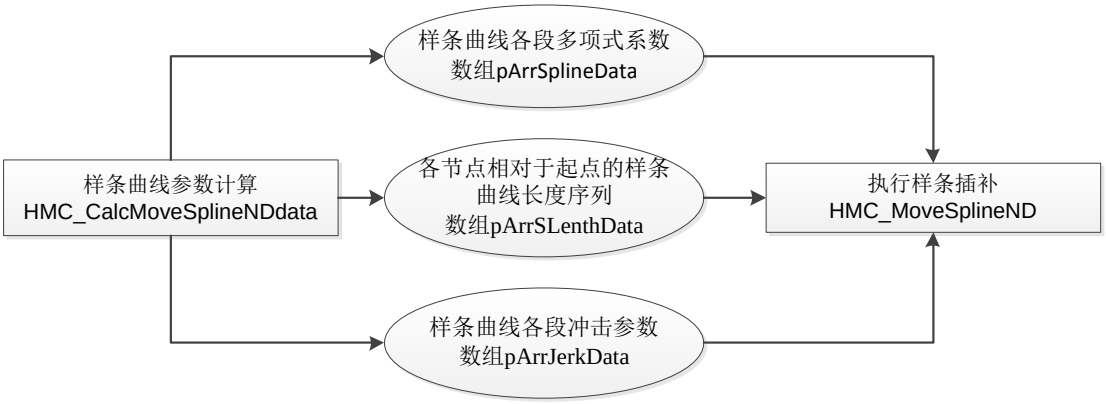


图 92 样条插补功能

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。

■ 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
pAxis	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	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	USINT	跟随轴的个数。
pArrPos	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	UDINT	pArrPos 中的样点个数，1~1000 个。
pArrSplineData	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	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	POINTER TO LREAL	数组首地址，HMC_CalcMoveSplineNDdata

输入参数	类型	参数描述
		中计算得到的 样条曲线各段冲击参数。

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0XFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 补充说明

（1）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；

（2）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。

- 应用示例

  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，则路径规划如图 93 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。

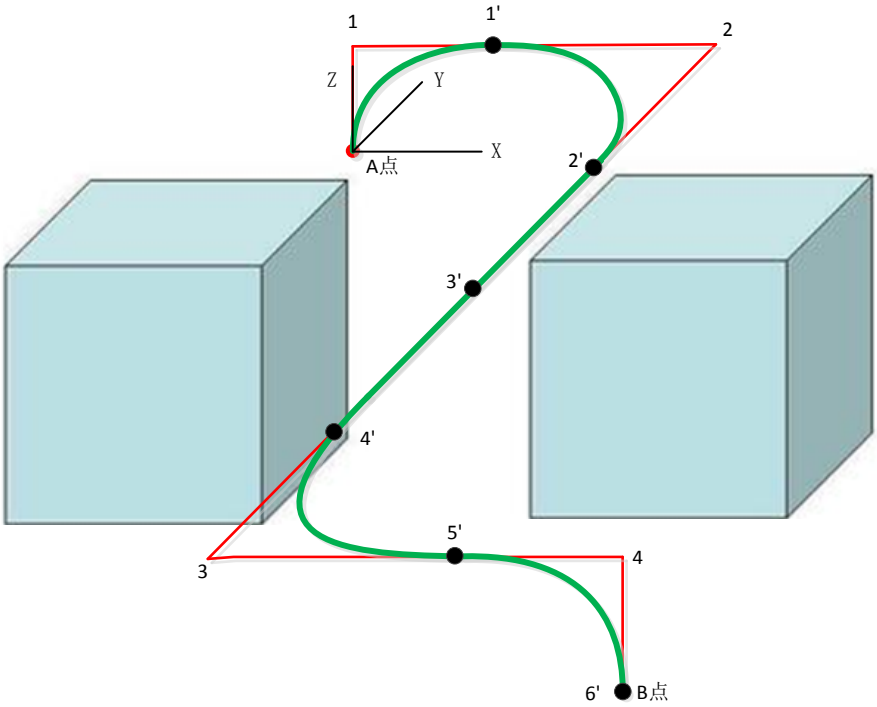


图 93 样条插补示意图

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：

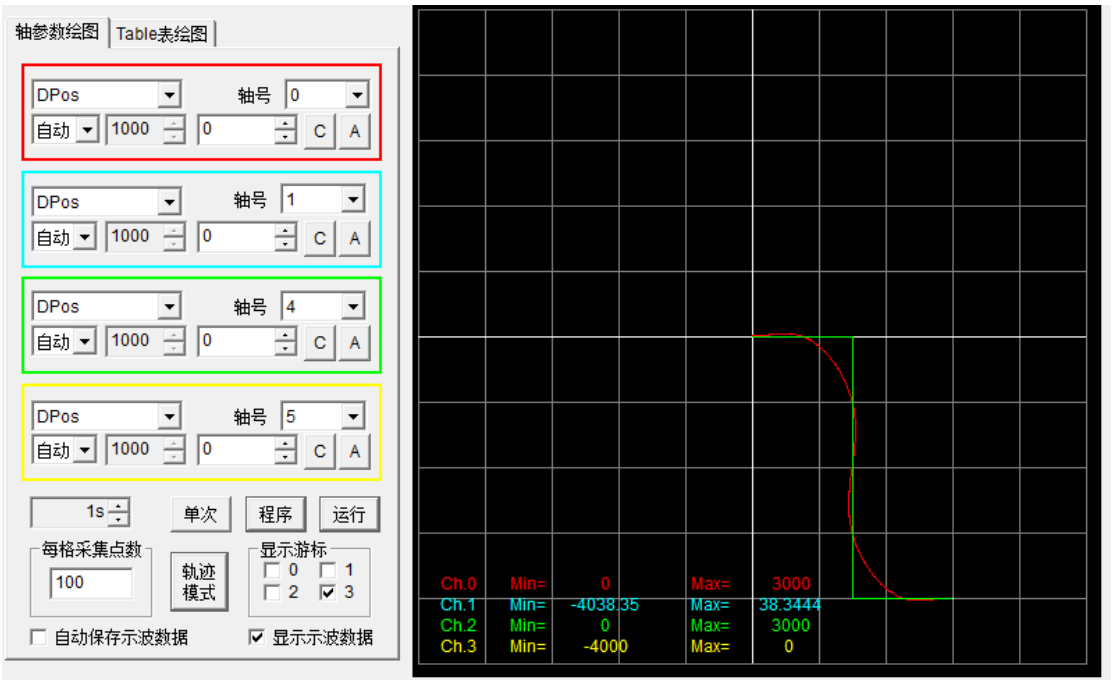


图 94　运动轨迹

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）如图 95 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。

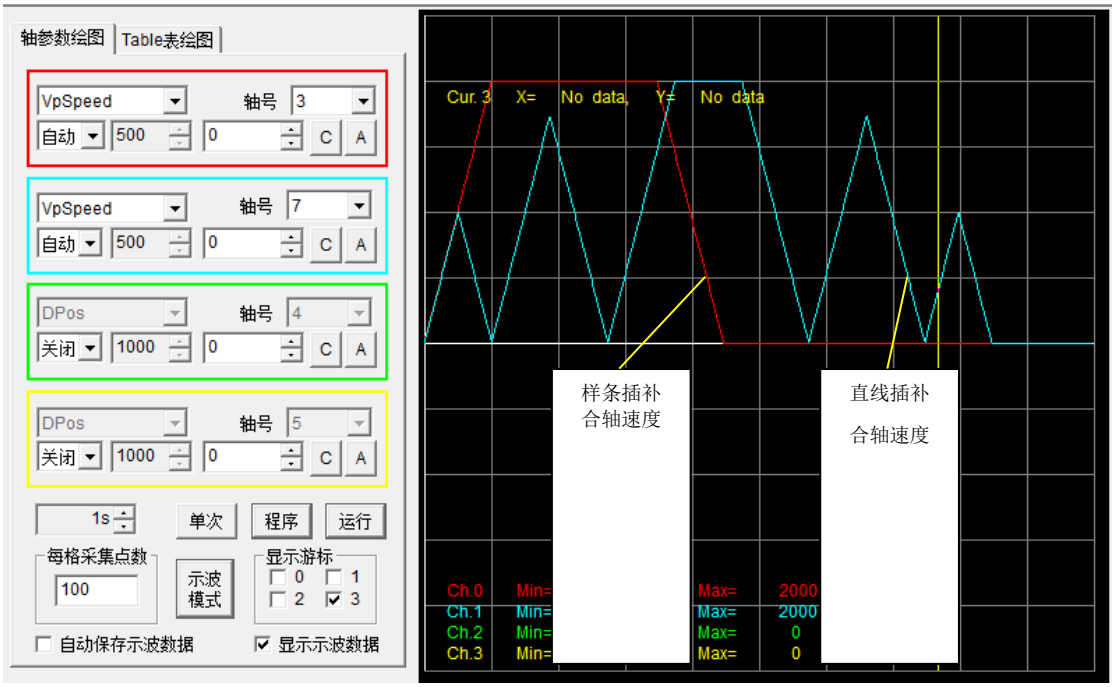


图 95　运动轨迹

（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 96 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。

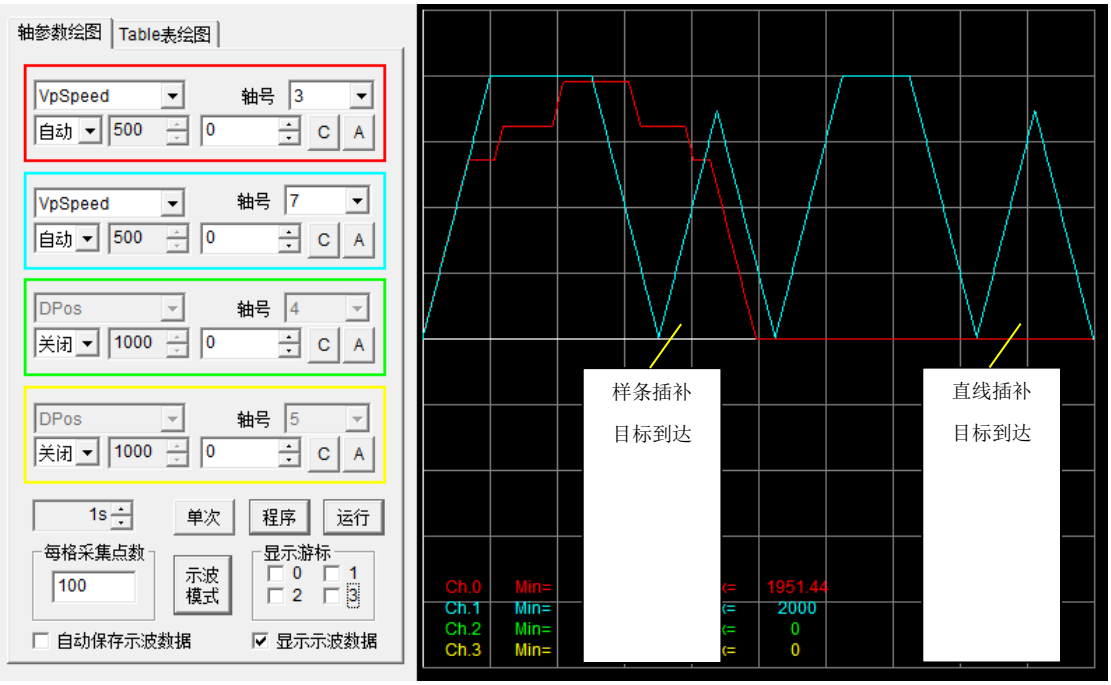


图 96 运动轨迹

（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。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.3.4.2 HMC_MoveSplineNDEx（高级样条插补）

- 函数原型

```
UDINT HMC_MoveSplineNDEx(USINT ucAxisID,
POINTER TO USINT pAxis,
USINT ucEndPointAxisNum,
USINT ucFollowAxisNum,
POINTER TO LREAL pArrPos,
UDINT uiSplineNum,
POINTER TO LREAL pArrSplineData,
POINTER TO LREAL pArrSlenthData,
POINTER TO LREAL pArrJerkData,
```

LREAL dSpeed)

- 功能说明

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）。

可参见章节 4.3.4.1 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。

- 参数

输入参数	类型	参数描述
ucAxisID	USINT	插补合运动轴号
pAxis	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	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	USINT	跟随轴的个数。
pArrPos	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	UDINT	pArrPos 中的样点个数，1~1000 个。
pArrSplineData	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	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	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	LREAL	合轴目标速度（插补轮廓速度）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	生成数据成功
0XFFFFFFFF	UDINT	函数参数错误

- 指令类型

轴运动缓存指令。

- 应用示例

参见 HMC_MoveSplineND。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.3.4.3 HMC_CalcMoveSplineNDdata（生成样条数据）

- 函数原型

UDINT HMC_CalcMoveSplineNDdata(POINTER TO LREAL pArrPos,

UDINT uiSplineNum,

USINT ucEndPointAxisNum,

USINT ucFollowAxisNum,

POINTER TO LREAL pArrSplineData,

UDINT uiSplineDataSize,

POINTER TO LREAL pArrSLenthData,

POINTER TO LREAL pArrJerkData)

- 功能说明

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。

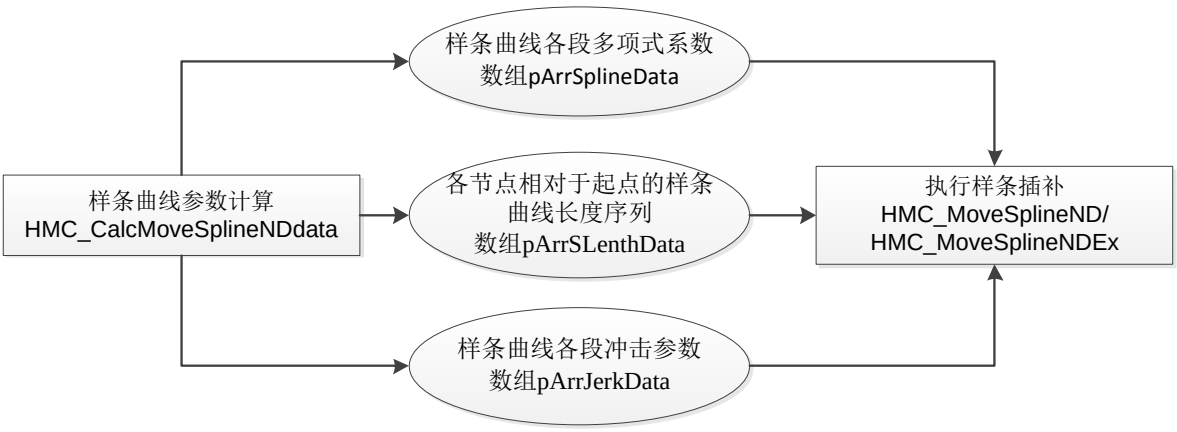


图 97 样条插补功能

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。

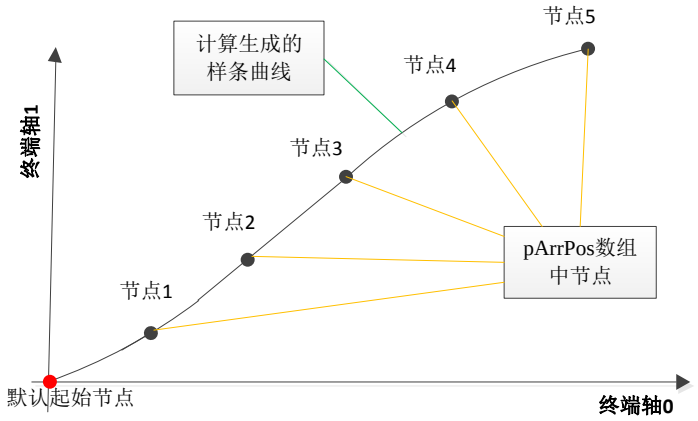


图 98 2 轴样条插补

（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。

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。

■ 参数

输入参数	类型	参数描述
pArrPos	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	UDINT	pArrPos 中的样点个数，1~1000 个。
ucEndPointAxisNum	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	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。

输入参数	类型	参数描述
pArrSplineData	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	UDINT	pArrSplineData 数组长度。
pArrSlenthData	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	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

- 返回值

返回值	类型	参数描述
0	UDINT	成功
非零	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 补充说明

（1）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。

（2）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。

（3）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。

- 应用示例

  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：

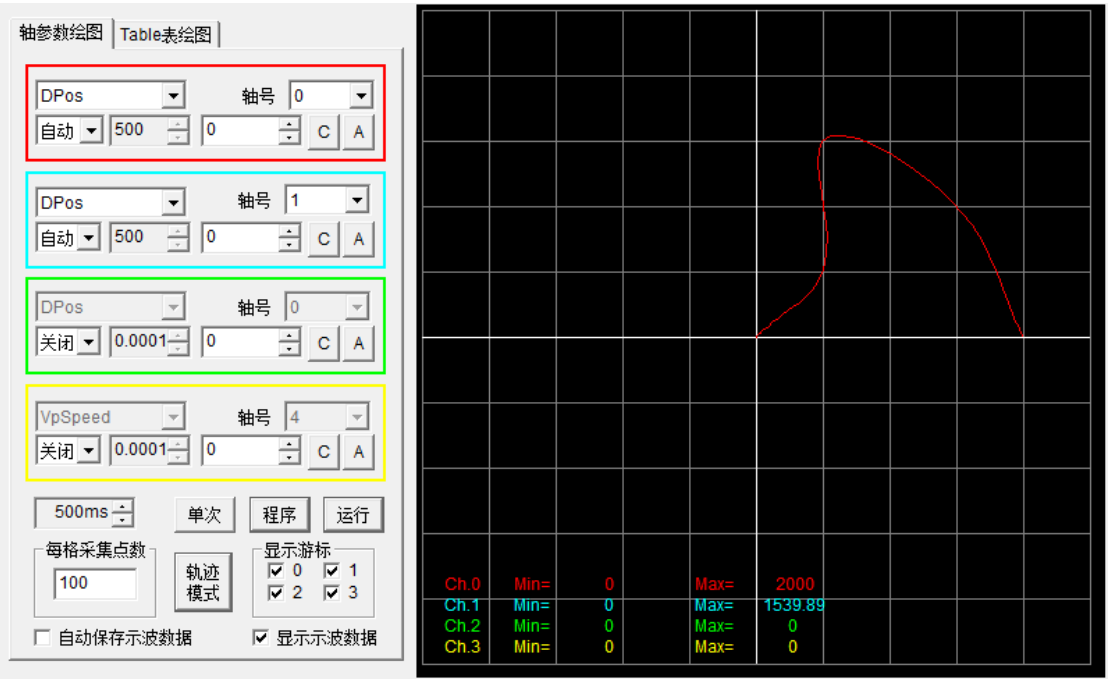


图 99 运动轨迹

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：

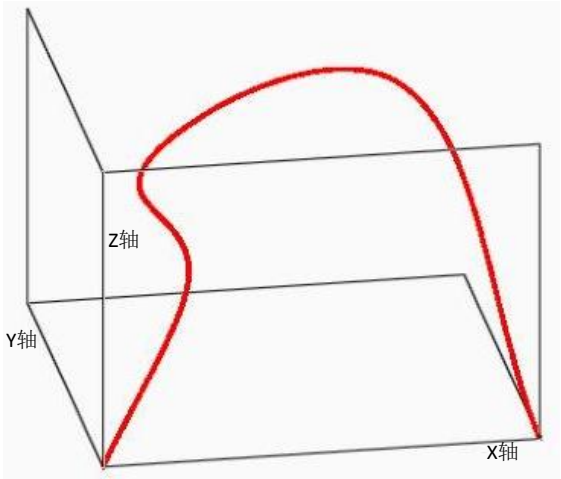


图 100 三维轨迹

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.3.5 电子齿轮

4.3.5.1 HMC_Gear（电子齿轮）

- 函数原型

  UDINT HMC_Gear(USINT ucSlaveAxis,

  USINT ucMasterAxis,

  DINT iRatioNumerator,

  DINT iRatioDenominator)

- 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。

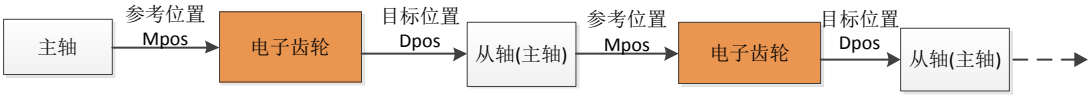


图 101 串联式电子齿轮结构图

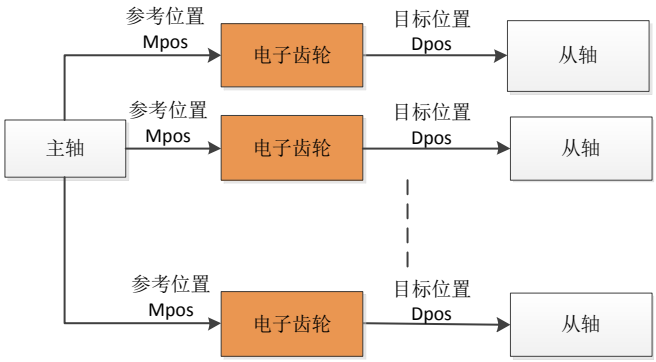


图 102 并联式电子齿轮结构图

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。

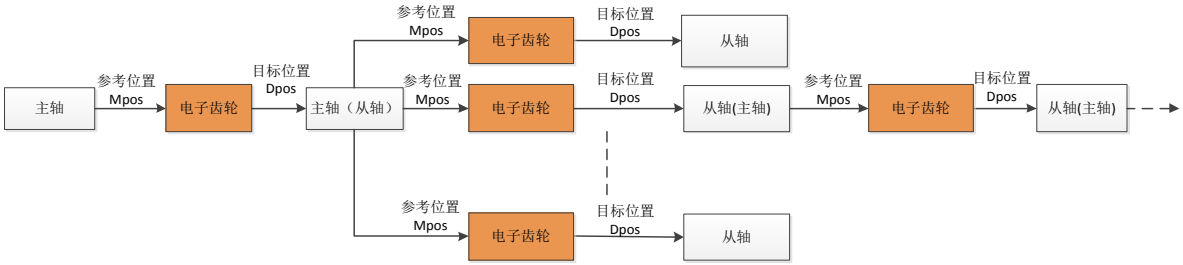


图 103 复合式电子齿轮结构图

使用电子齿轮后，从轴 Dpos 增量与主轴 Mpos 增量满足如下关系：

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

- 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
iRatioNumerator	DINT	齿轮比分子
iRatioDenominator	DINT	齿轮比分母（不可为 0）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 补充说明

（1） 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；

（2） 从数学关系上看，HMC_Gear 和 HMC_Superimpose 均是 HMC_SuperimposeEx 的特例；

（3） HMC_Gear 和 HMC_Superimpose 配合使用可实现 HMC_SuperimposeEx 的所有功能。

■ 应用举例

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。

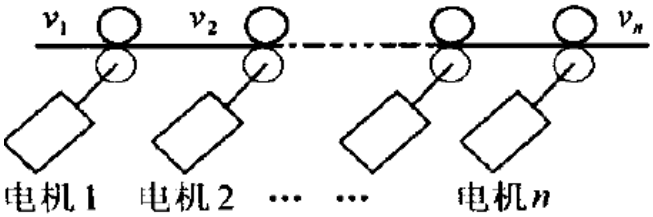


图 104 连续工艺过程装备传动示意图

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。

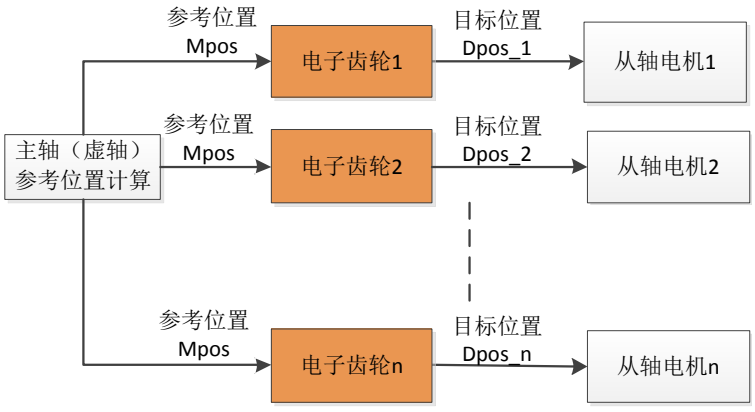


图 105 连续工艺过程装备传动电子齿轮实现方式

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）。

4.3.5.2 HMC_SetGearRatio（设置电子齿轮比例）

- 函数原型

  UDINT HMC_SetGearRatio(USINT ucSlaveAxis,

  DINT iRatioNumerator,

  DINT iRatioDenominator)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号
iRatioNumerator	DINT	齿轮比分子
iRatioDenominator	DINT	齿轮比分母

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其他值	UDINT	错误码

- 指令类型

非轴运动缓存指令。

4.3.5.3 HMC_GetGearMaster（获取电子齿轮主轴）

- 函数原型

INT HMC_GetGearMaster(USINT ucSlaveAxis)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
电子齿轮主轴号	INT	成功
-1	INT	当前未配置电子齿轮或函数参数错误

- 指令类型

非轴运动缓存指令。

4.3.5.4 HMC_GetGearRatio（获取电子齿轮比例）

- 函数原型

LREAL HMC_GetGearRatio(USINT ucSlaveAxis)

- 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。

- 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
电子齿轮比例	LREAL	-

- 指令类型

  非轴运动缓存指令。

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

- 函数原型

  UDINT HMC_SetGearPhaseOffset(USINT ucAxisID, DINT iPhaseOffset)

- 功能说明

  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 106 所示：

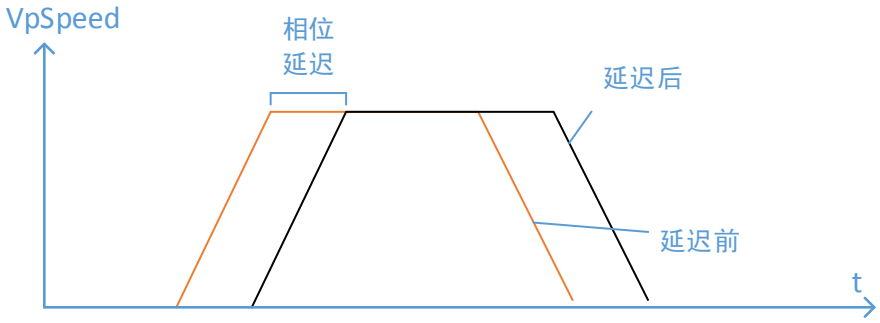


图 106 从轴相位延迟轨迹

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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	从轴轴号
iPhaseOffset	DINT	相位延迟量，0~128，单位为伺服周期

- 返回值

返回值	类型	参数描述
成功：0 错误：错误码	UDINT	--

- 指令类型

  非轴运动缓存指令。

- 补充说明

  必须在电子齿轮指令执行前调用 HMC_SetGearPhaseOffset 设定好所需的相位延迟，无法在电子齿轮指令执行过程中动态调整相位延迟。

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

- 函数原型

  DINT HMC_GetGearPhaseOffset (USINT ucAxisID)

- 功能说明

  获取电子齿轮相位延迟设定量。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	从轴轴号，0~63

- 返回值

返回值	类型	参数描述
相位延迟量	DINT	单位为伺服周期

- 指令类型

  非轴运动缓存指令。

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

- 函数原型

  UDINT HMC_SetGearTransationSwitch(USINT ucAxisID, BOOL ucValue)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucValue	BOOL	0：关闭 1：开启

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

4.3.6 电子凸轮

4.3.6.1 HMC_Cam（电子凸轮）

- 函数原型

UDINT HMC_Cam(USINT ucSlaveAxis,

USINT ucMasterAxis,

POINTER TO LREAL pArrPos,

UDINT uiPosNum,

LREAL dScale,

USINT ucMode,

LREAL dTrigMes)

■ 功能说明

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。

■ 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）。

■ 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

■ 指令类型

轴运动缓存指令。

■ 补充说明

（1）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！

（2）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）。

（3）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。

（4）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)。

■ 应用举例

（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);
```

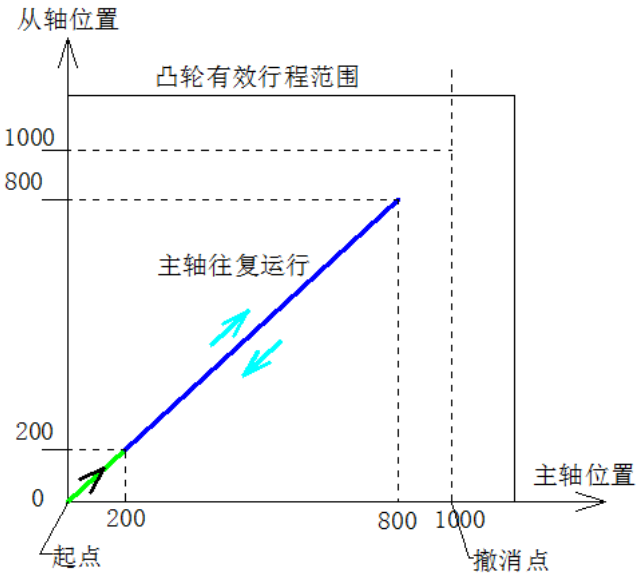


图 107 单次立即启动凸轮联动轨迹

（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);
```

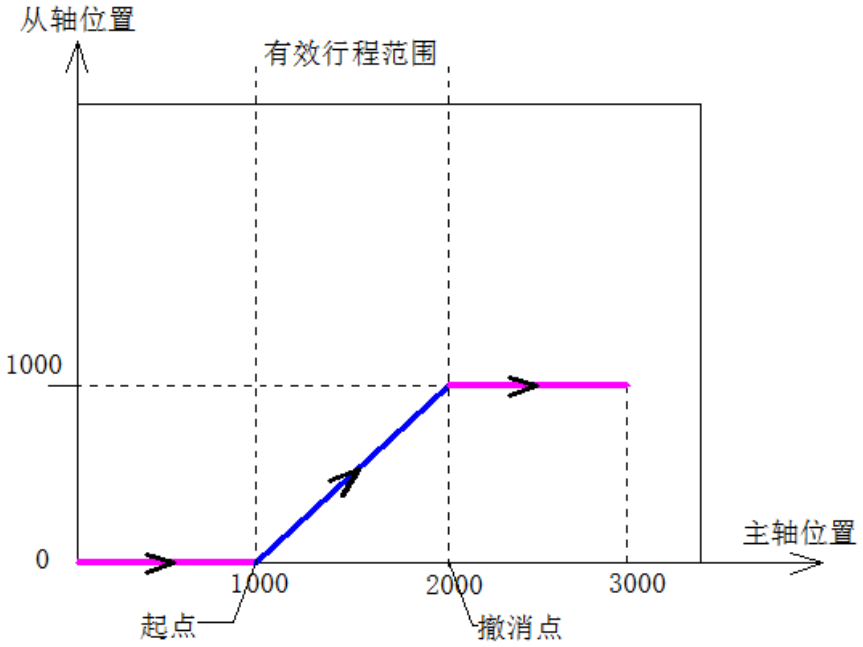


图 108 单次固定位置启动凸轮联动轨迹

（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);
```

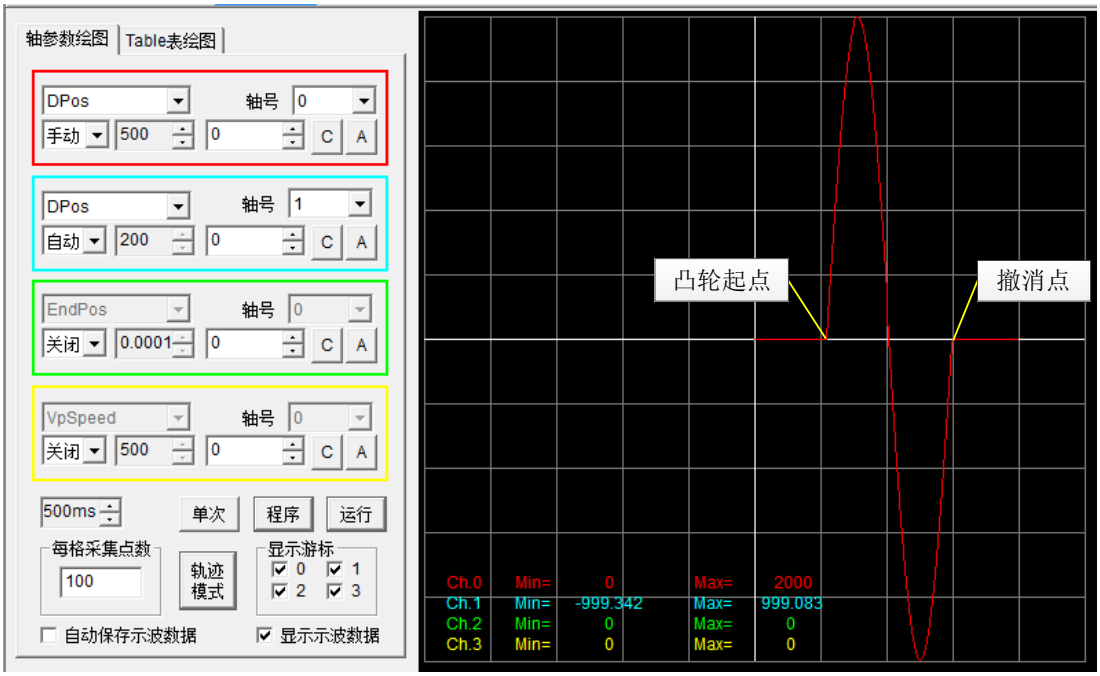


图 109 单次事件启动凸轮联动轨迹

（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);
```

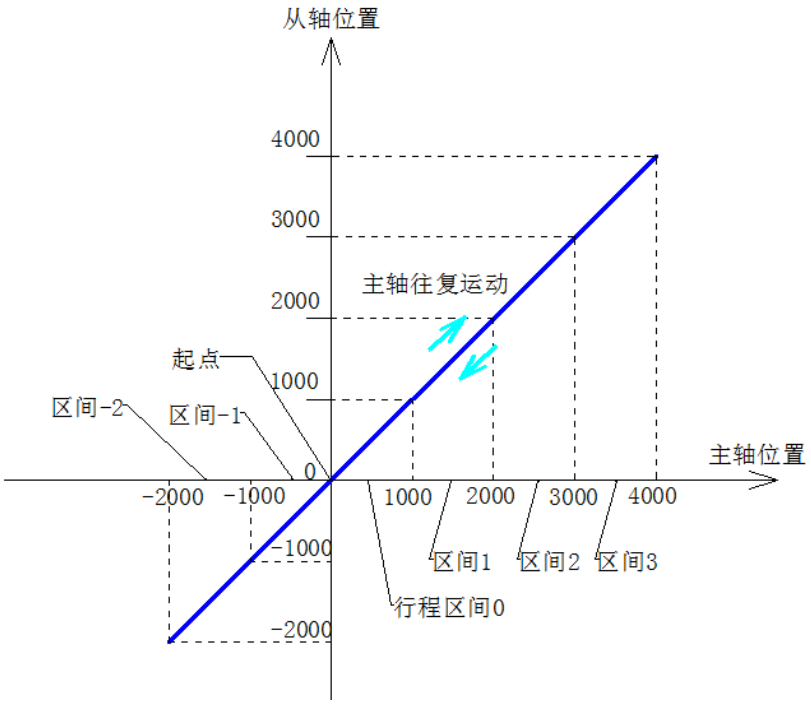


图 110　循环立即启动凸轮联动轨迹

（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);
```

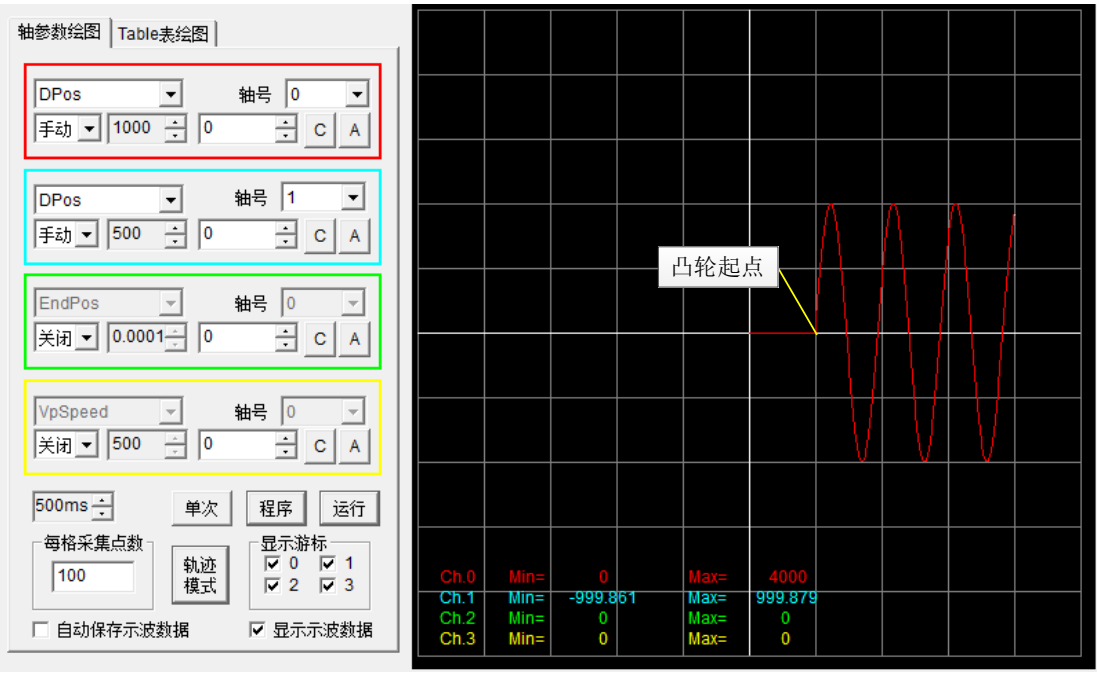


图 111 循环固定位置启动凸轮联动轨迹

（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。

4.3.6.2 HMC_SplineCam（样条凸轮）

■ 函数原型

UDINT HMC_SplineCam(USINT ucSlaveAxis,

USINT ucMasterAxis,

POINTER TO LREAL pArrPos,

UDINT uiPosNum,

LREAL dScale,

USINT ucMode,

LREAL dTrigMes)

- 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 4.3.6.1 HMC_Cam（电子凸轮）。
- 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）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

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

- 函数原型

  UDINT HMC_GetCamMasterPosIndex(USINT ucAxisID)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
位置索引	UDINT	当前正在运行的凸轮的主轴位置索引（在凸轮主轴位置数组中的区间索引）

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01

4.3.6.4 HMC_ClcMoveLinkData（追剪运算）

- 函数原型

  UDINT HMC_ClcMoveLinkData(LREAL dSlaveDistance,

  LREAL dMasterDistance,

  LREAL dDistanceAcc,

  LREAL dDistanceDec,

  POINTER TO LREAL pArrPos,

  UDINT uiPosNum)

- 功能说明

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，主从轴曲线如图 112 所示。该运动带有分离可变的加减速阶段。

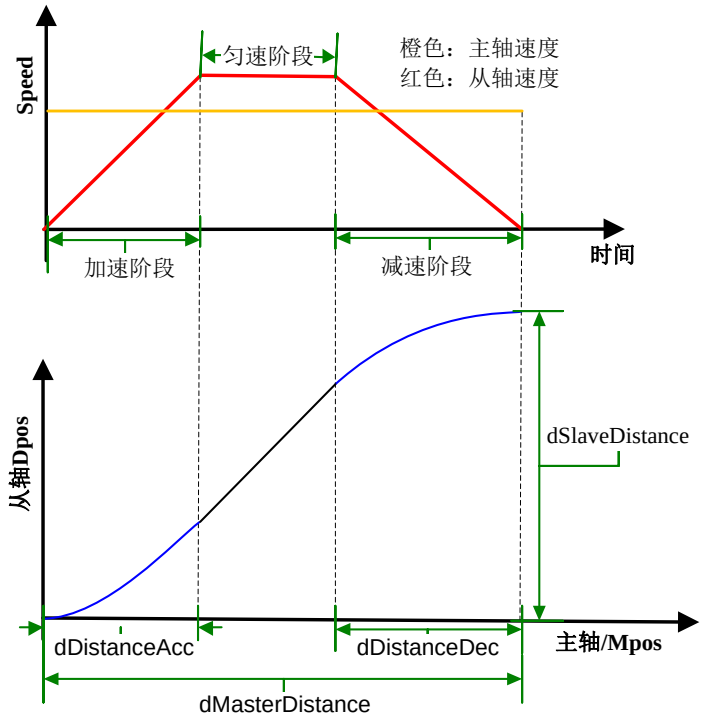


图 112 主/从轴示意图

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。

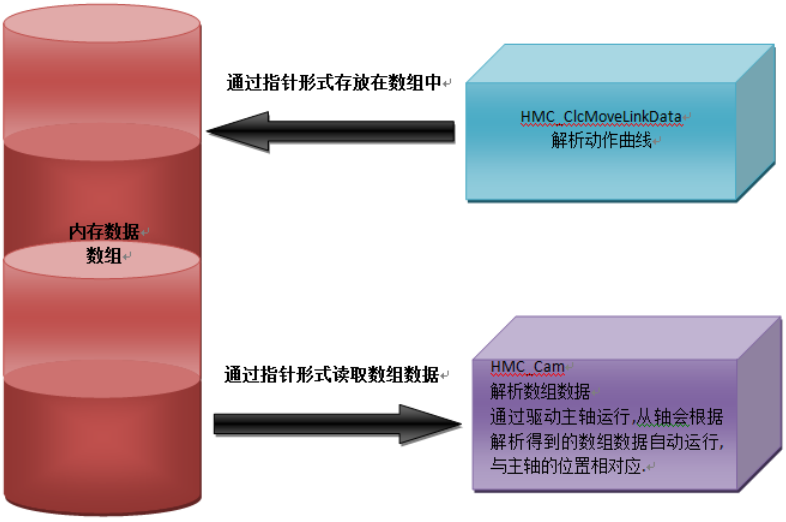


图 113 HMC_ClcMoveLinkData 与 HMC_Cam 数据流

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。

- 输入参数

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

- 返回值

返回值	类型	参数描述
0	UDINT	生成数据成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

非轴运动缓存指令。

- 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，图 112 中凸轮从轴将无匀速阶段，此时凸轮主从轴曲线如下：

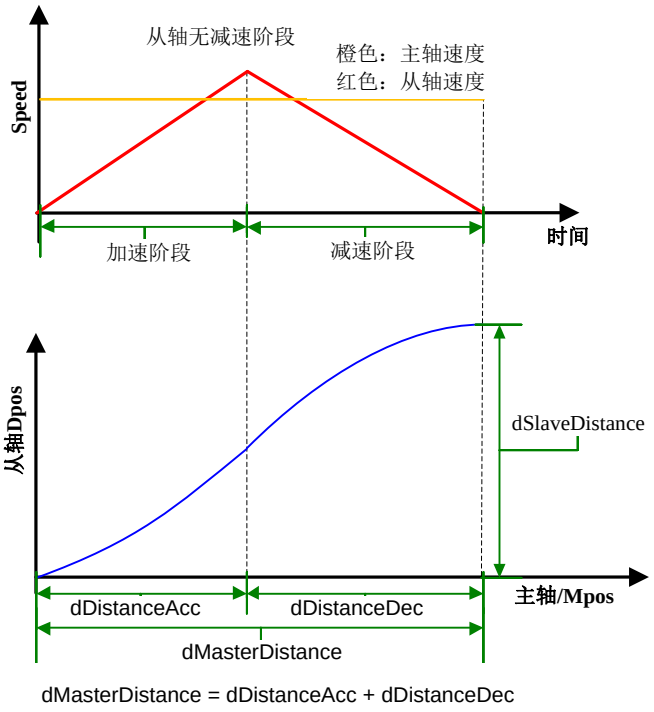


图 114 主/从轴示意图

（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，图 112 中的凸轮主从轴曲线如下：

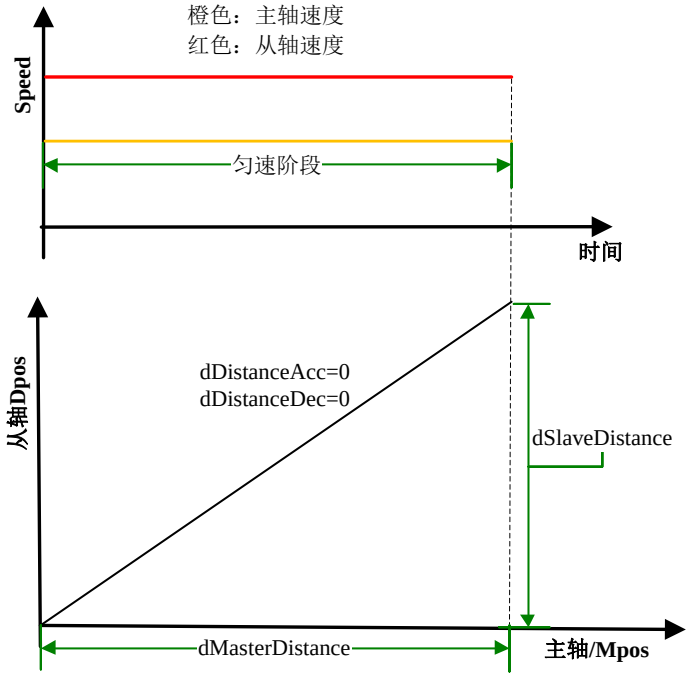


图 115 主/从轴示意图

（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。图 112 中的凸轮主从轴曲线如下：

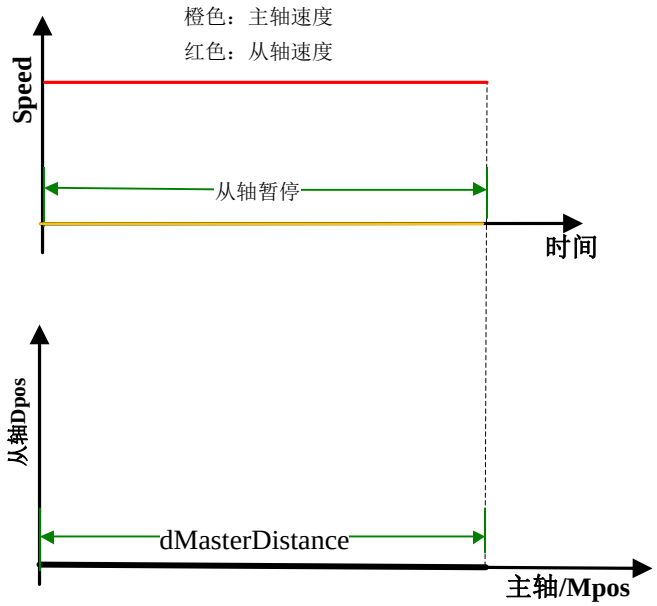


图 116 主/从轴示意图

■ 应用举例

（1）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。

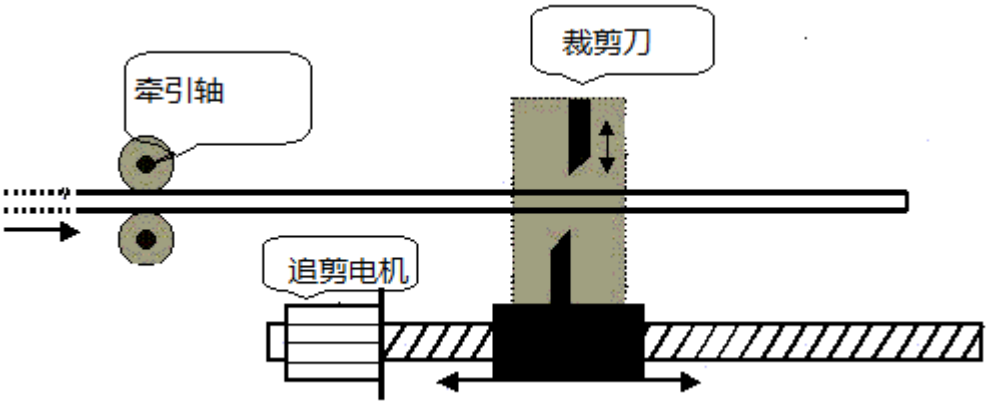


图 117 追剪方案示意图

满足追剪要求的速度曲线如图 118 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。

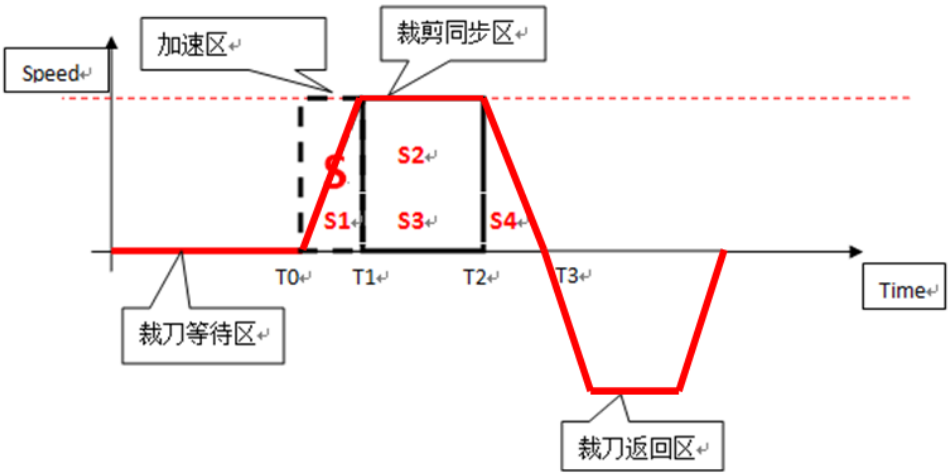


图 118 追剪速度规划图

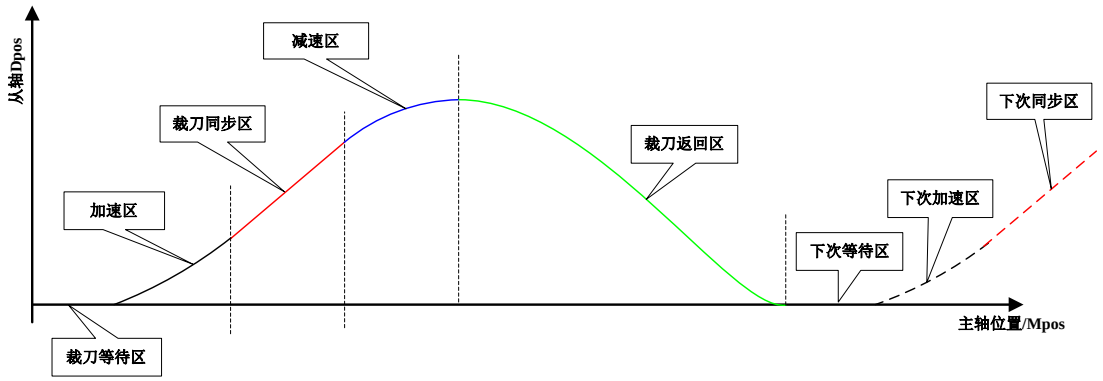


图 119 追剪主从轴位移曲线

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 *)
```

- 不支持的模块版本

  MC1008-A01

4.3.6.5 HMC_ClcFlexLinkData（飞剪/旋切运算）

- 函数原型

  UDINT HMC_ClcFlexLinkData(LREAL dSlaveBaseDistance,
  LREAL dSlaveSuperimpose,
  LREAL dSlaveBaseDistance,
  LREAL dSlaveSuperimpose,
  LREAL dMasterDistance,
  LREAL dSuperimposeIn,
  LREAL dSuperimposeOut,
  LREAL dSuperimposeAcc,
  LREAL dSuperimposeDec,
  POINTER TO LREAL pArrPos,
  UDINT uiPosNum)

■ 功能说明

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，主从轴曲线如图 120 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。

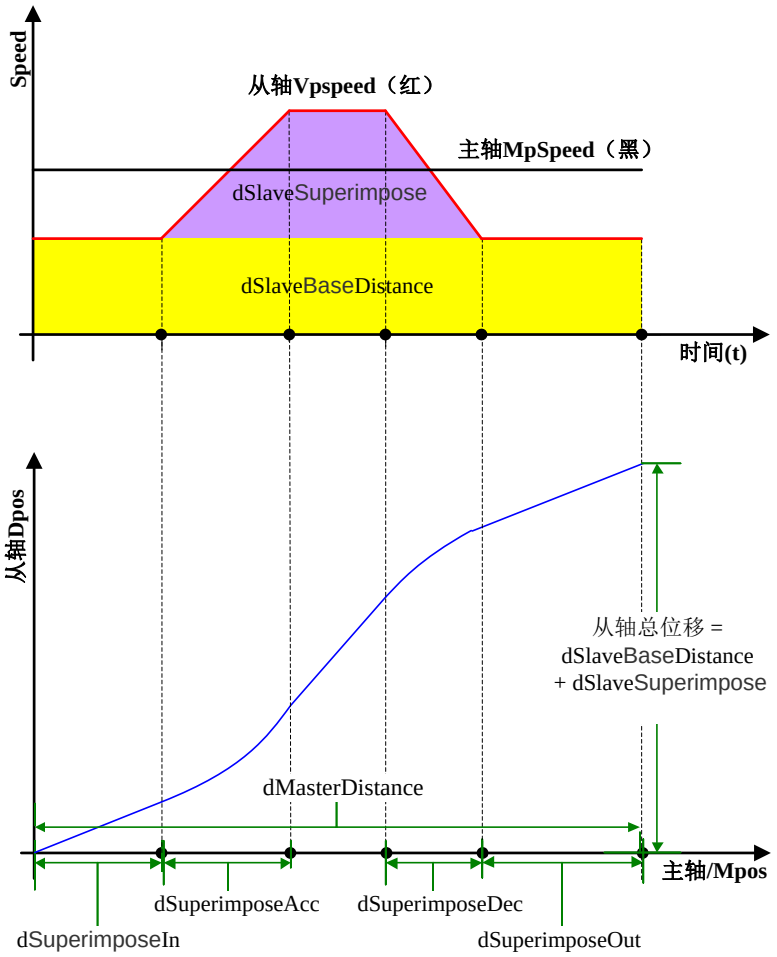


图 120 主/从轴示意图

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。

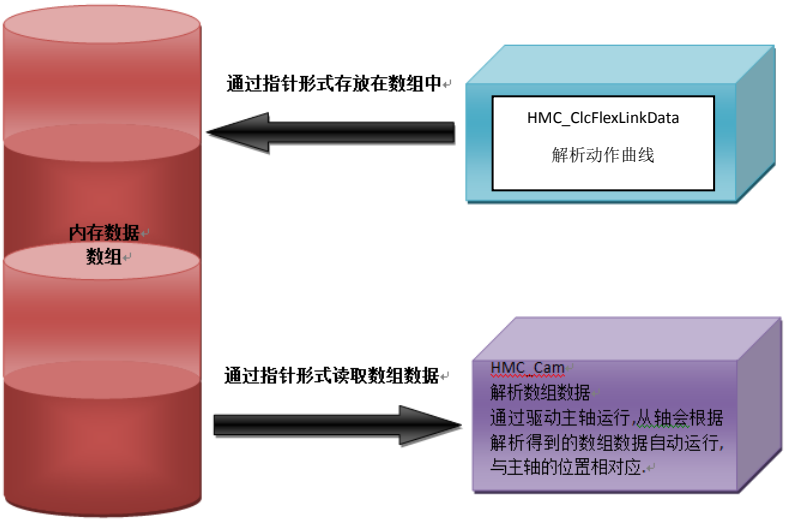


图 121 HMC_ClcFlexLinkData 与 HMC_Cam 数据流

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。

- 输入参数

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

- 返回值

返回值	类型	参数描述
0	UDINT	生成数据成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

非轴运动缓存指令。

■ 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。

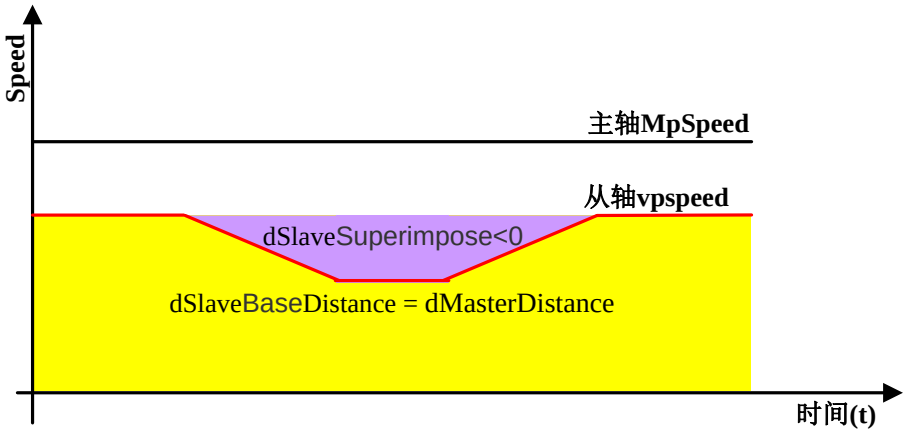


图 122 从轴叠加距离为负时的情形

（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。

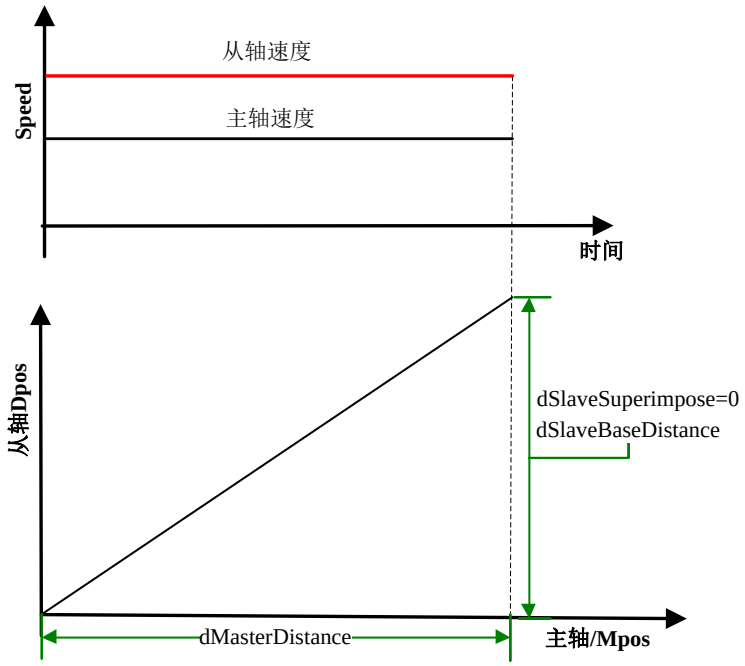


图 123 运动示意图

（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。

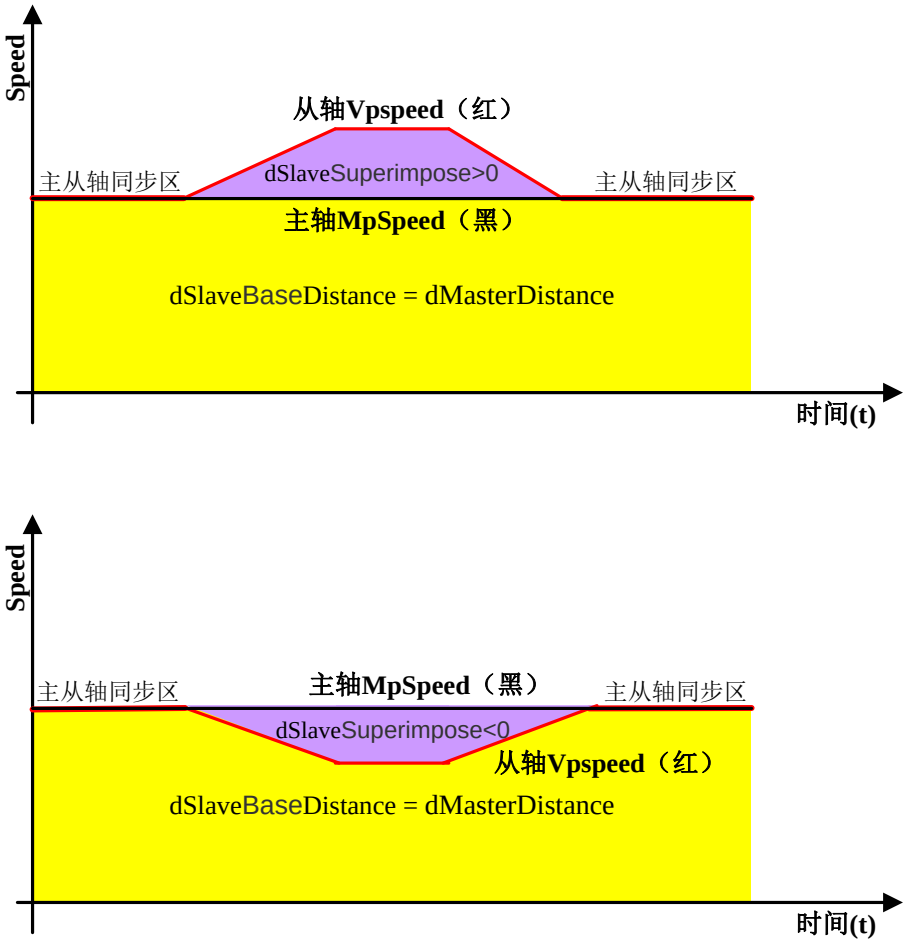


图 124　运动示意图

（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。

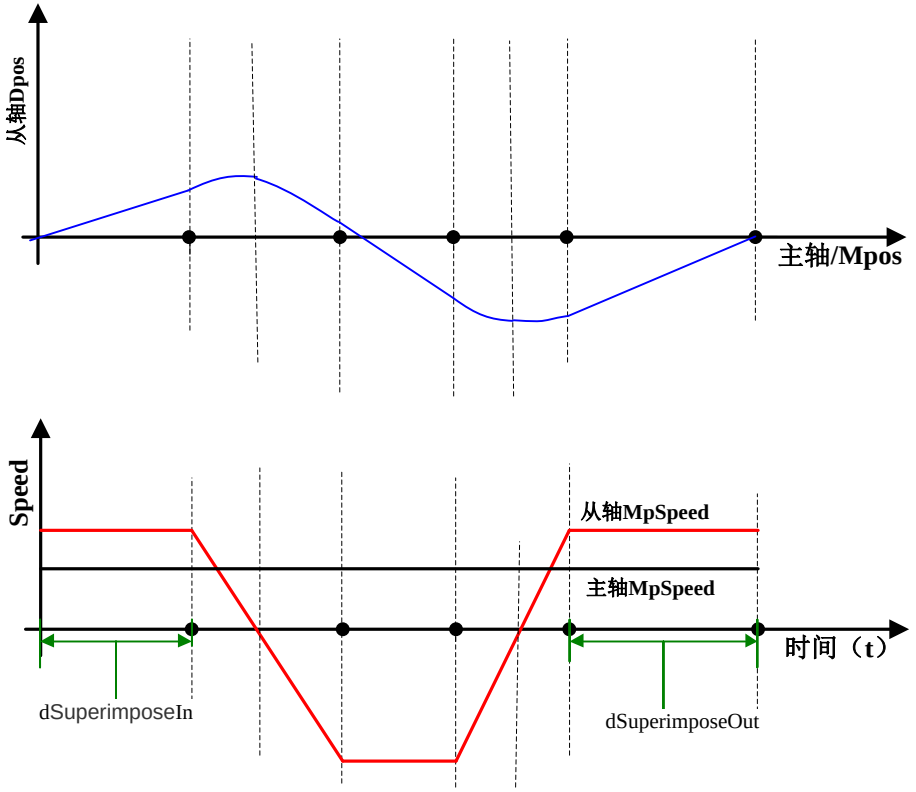


图 125　主/从轴示意图

■ 应用举例

（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。

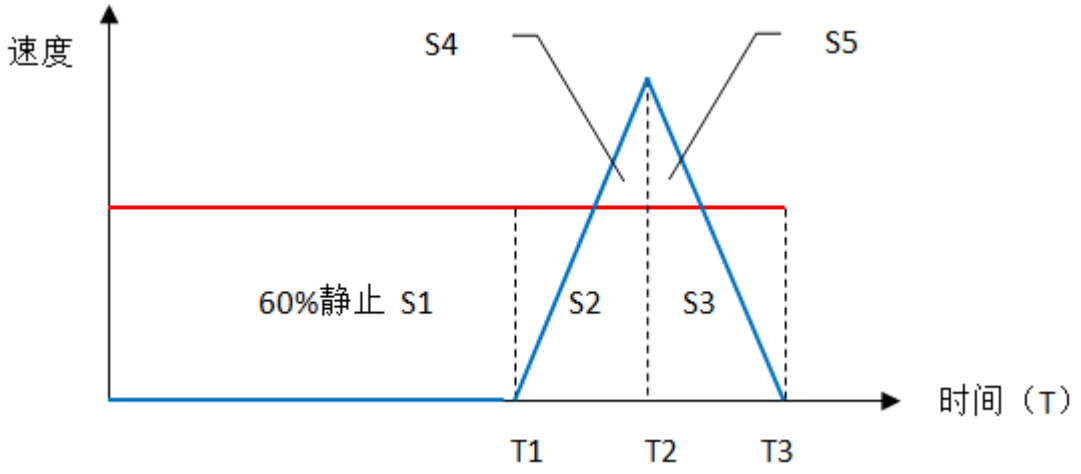


图 126　运动示意图

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：

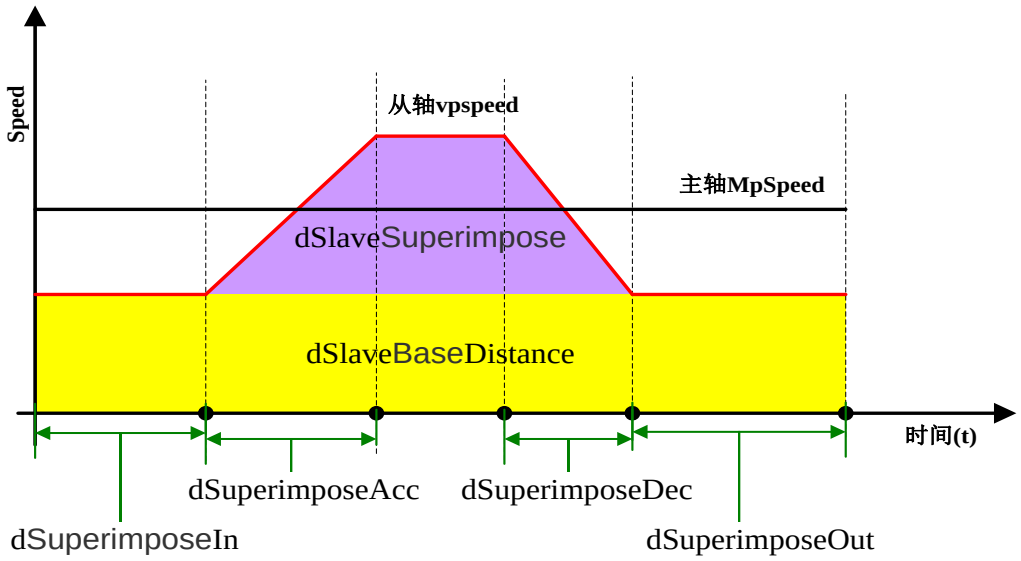


图 127 运动示意图

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）：

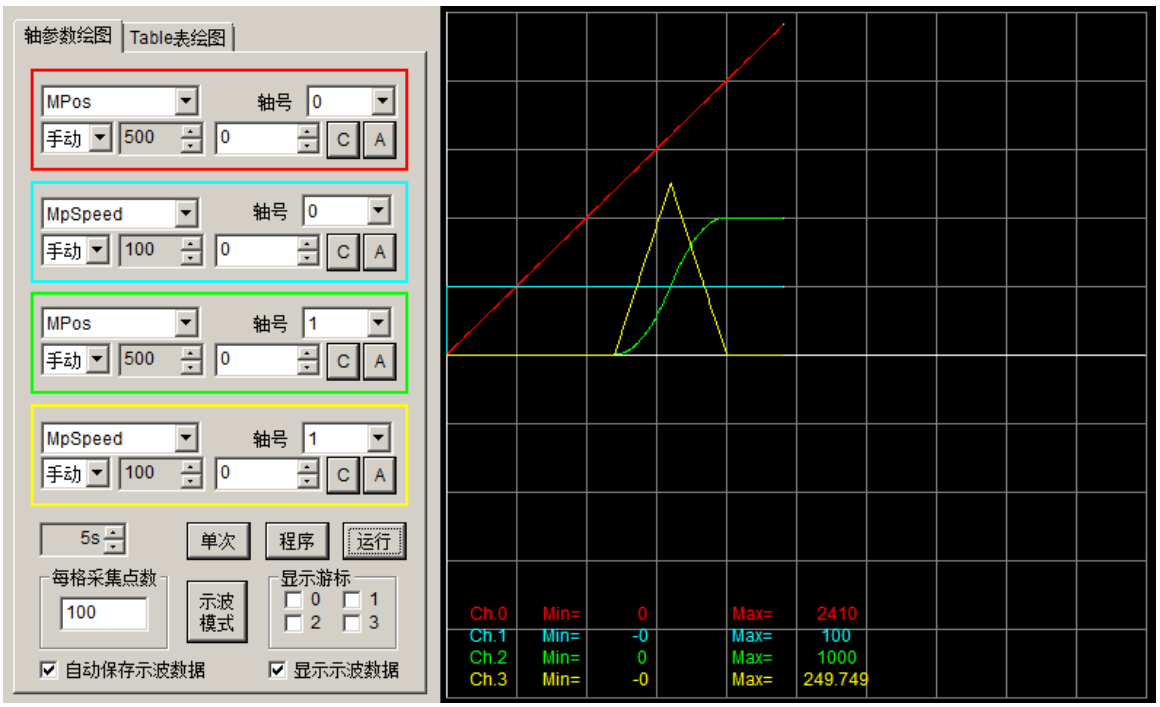


图 128　示波器显示结果

（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。

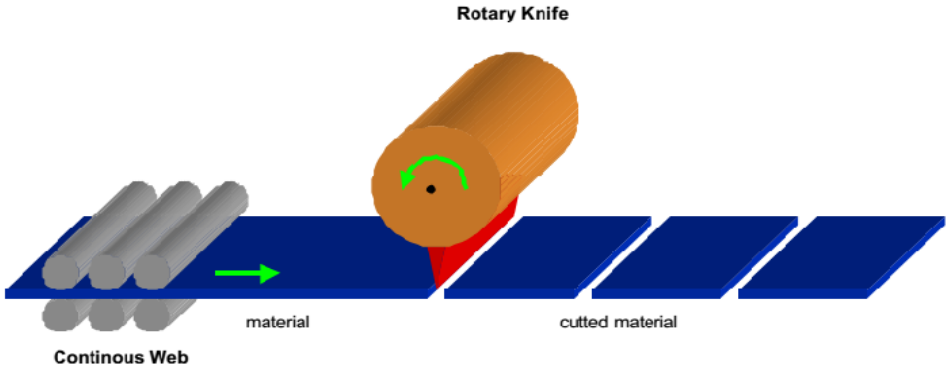


图 129 飞剪/旋切示意图

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;
```

1）情形一：短料剪切（L<剪刀周长）L=400 mm

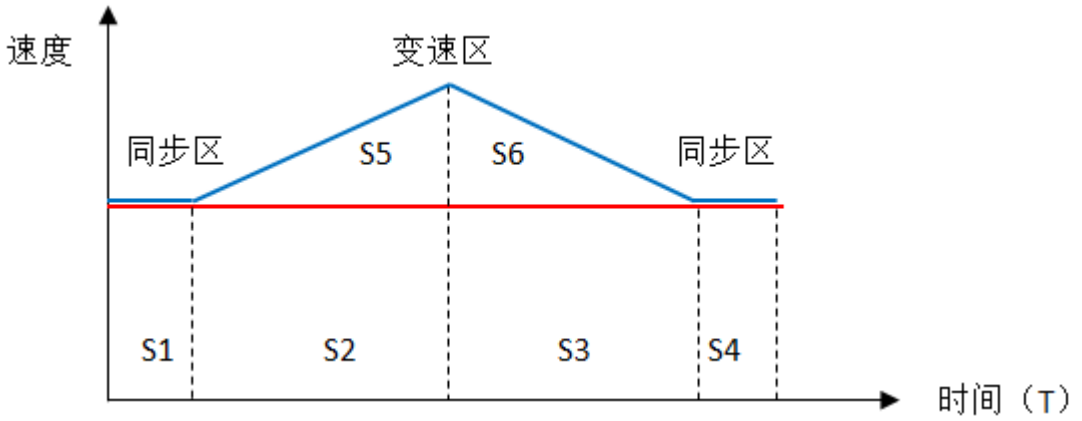


图 130 短料剪切示意图（红色代表主轴，蓝色代表从轴）

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

图 131 Arraympos 变量定义

运行任务。示波器显示：

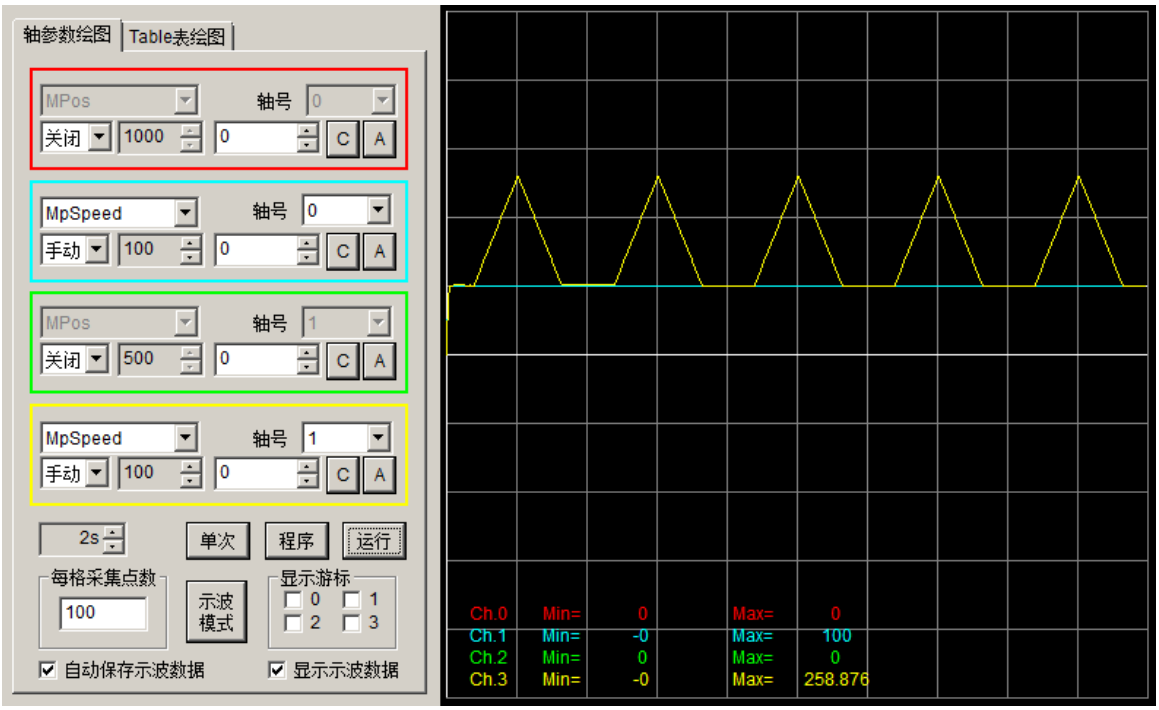


图 132 示波器显示结果

2）情形二：中料剪切（L>=剪刀周长）L=800 mm

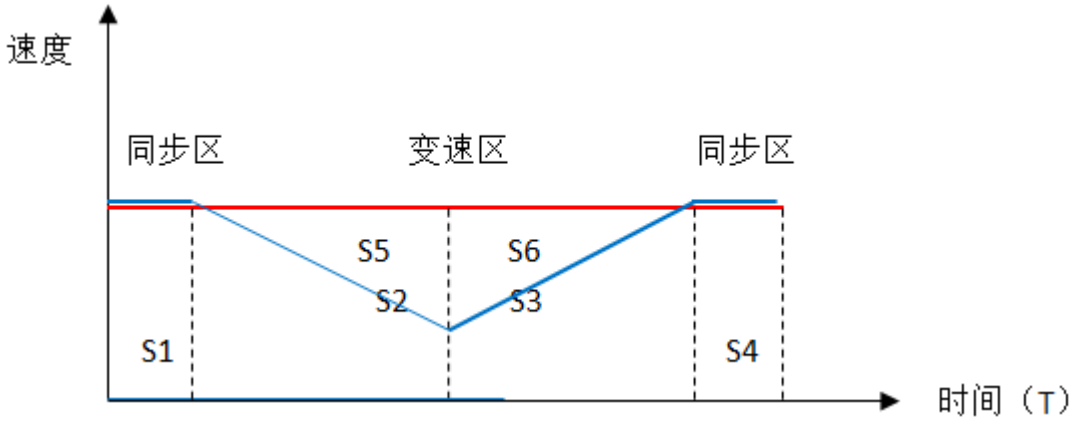


图 133 中料剪切示意图

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：

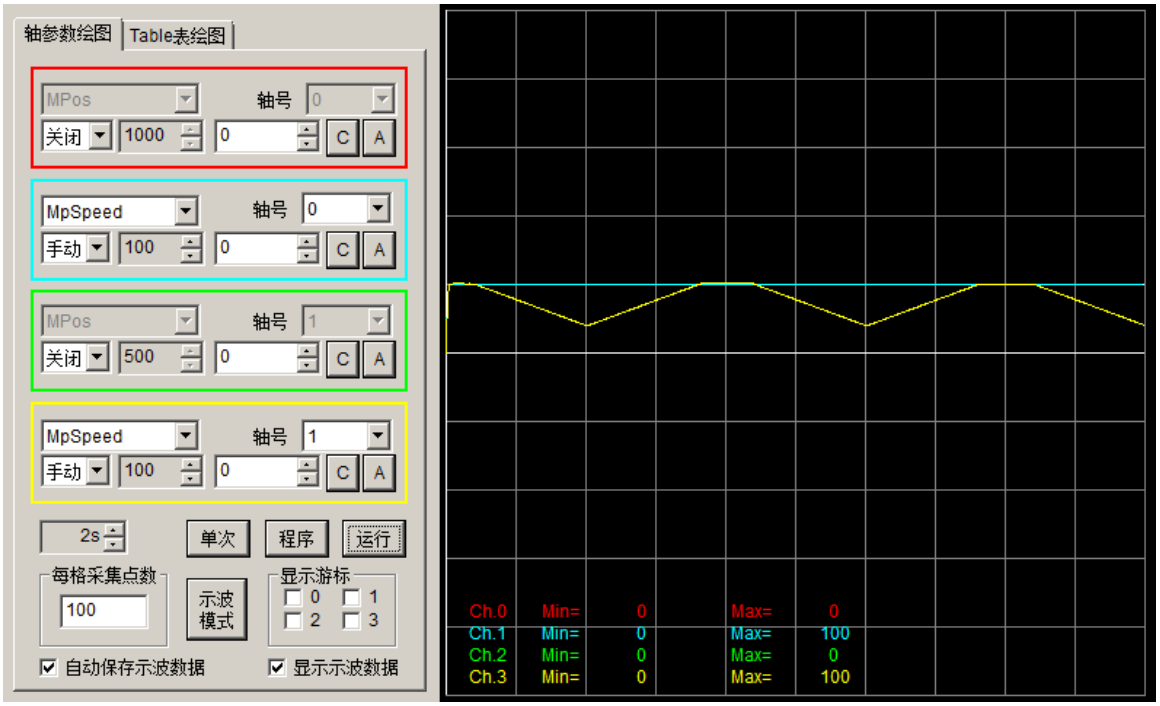


图 134 示波器显示结果

3）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。

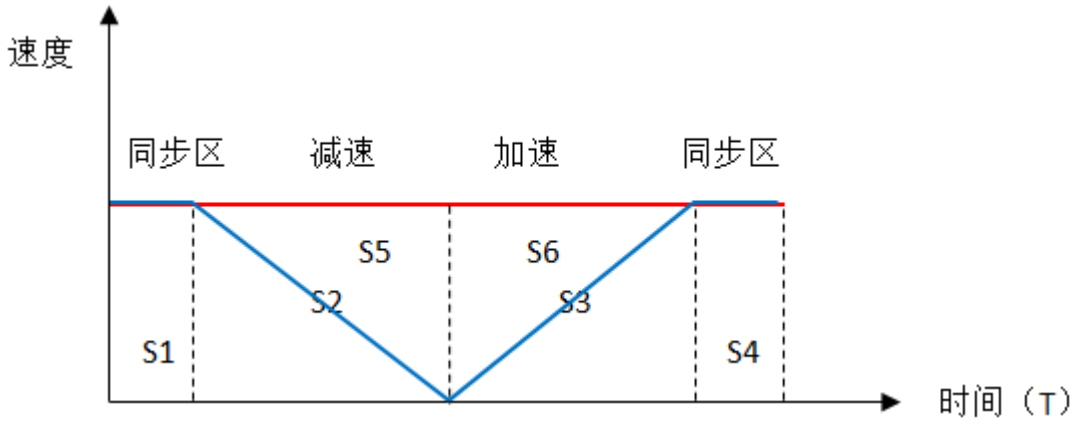


图 135 长料剪切与中料剪切分界点计算

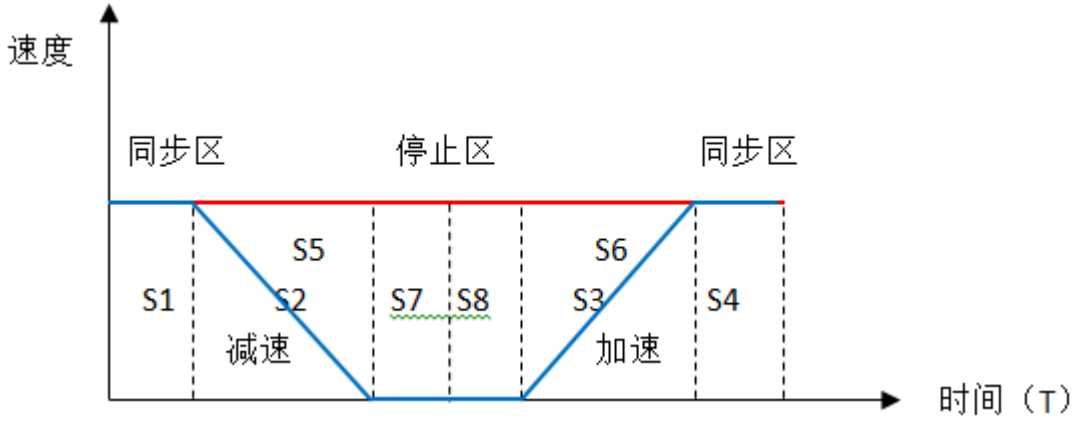


图 136 长料剪切示意图

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：

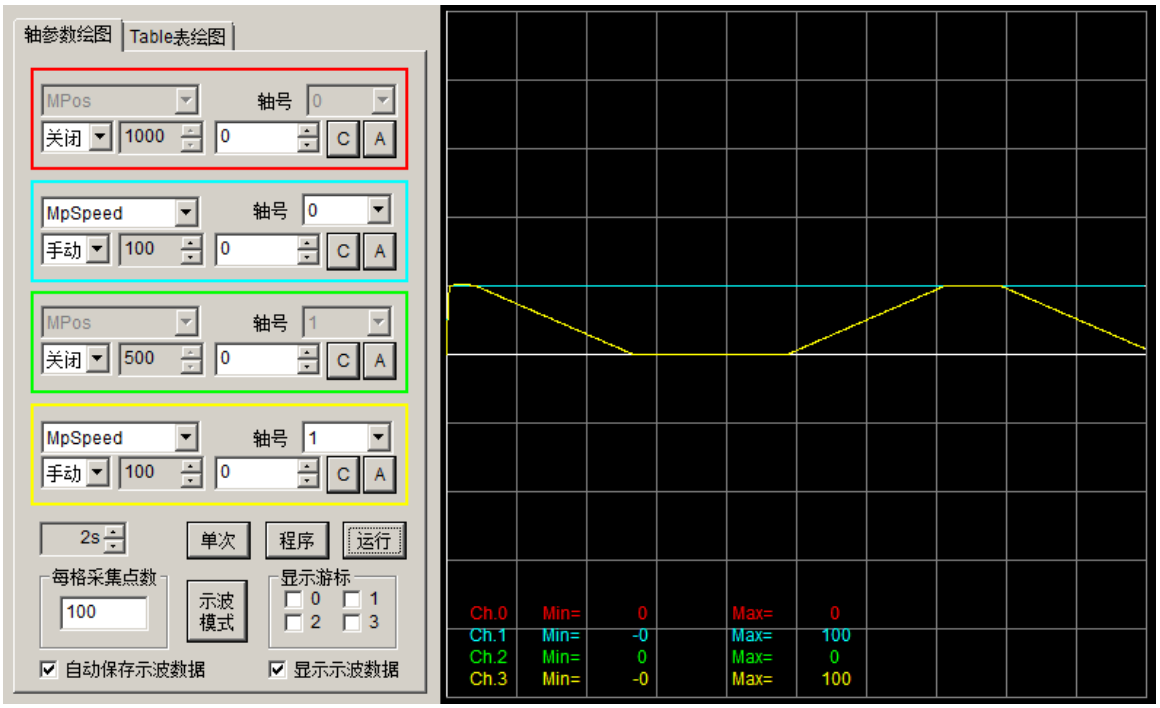


图 137 示波器显示结果

- 不支持的模块版本

  MC1008-A01

4.3.6.6 HMC_CancelREPCam（循环凸轮条件撤销控制）

- 函数原型

  UDINT HMC_CancelREPCam(USINT ucSlaveAxis, USINT ucCancelCycMode)

- 功能说明

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。

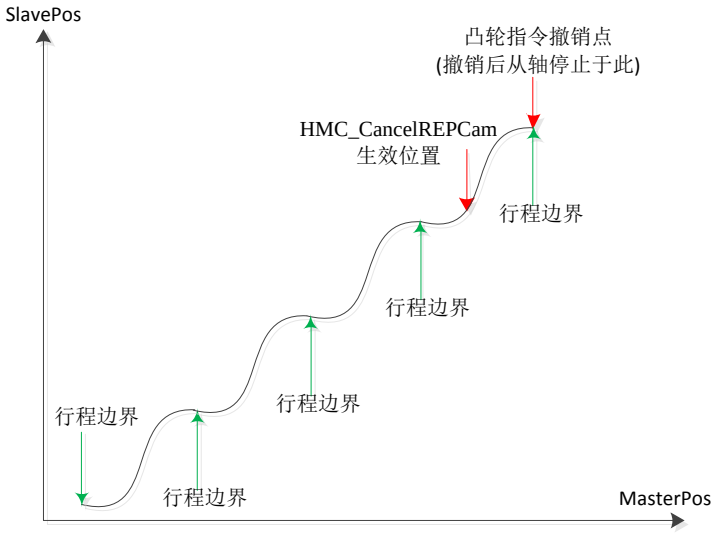


图 138　循环凸轮执行 HMC_CancelREPCam 指令示意图

- 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

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令

- 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。

- 应用举例

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*)
```

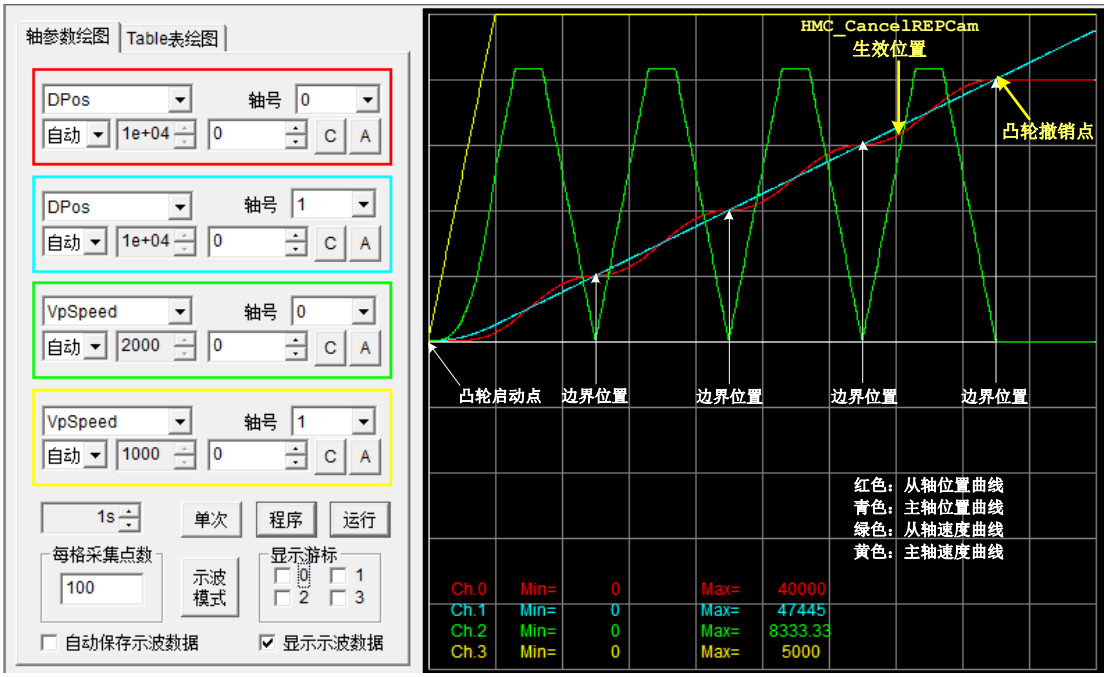


图 139　循环凸轮执行 HMC_CancelREPCam 指令运行结果图

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.3.6.7　HMC_MoveLink（追剪运动）

- 函数原型

  UDINT HMC_MoveLink（LREAL dSlaveDistance,

LREAL dMasterDistance,

LREAL dMasterDistAcc,

LREAL dMasterDistDec,

LREAL dSRatio,

USINT ucSlaveAxis,

USINT ucMasterAxis,

USINT ucMode,

LREAL dTrigMes）

■ 功能说明

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章节 4.3.6.4 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

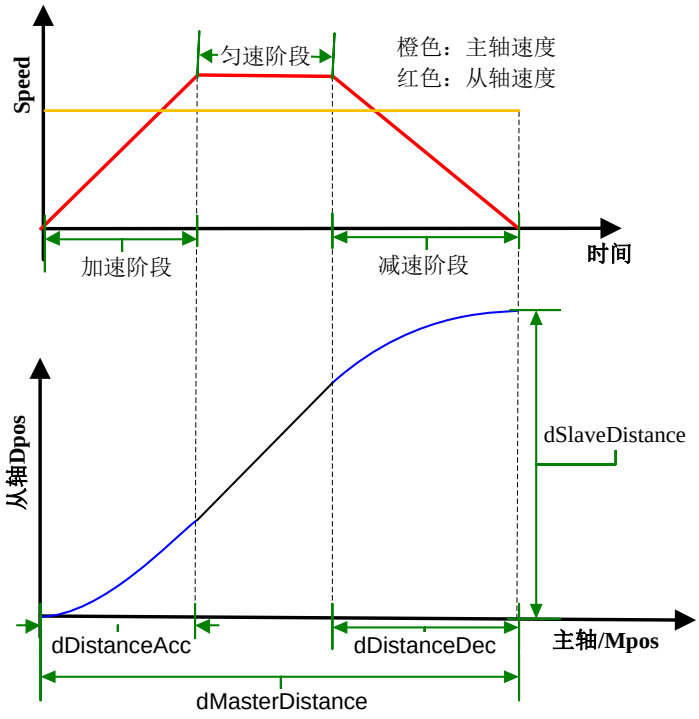


图 140 运动示意图

（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：

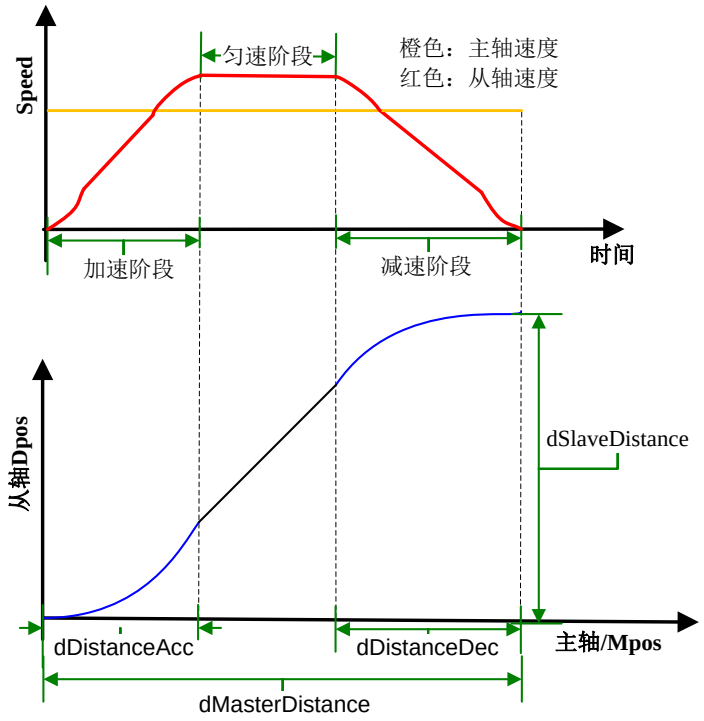


图 141　运动示意图

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。

■　输入参数

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

输入参数	类型	参数描述
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）

- 指令类型

  轴运动缓存指令

- 应用举例

  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，因此应用举例可参数 HMC_ClcMoveLinkData。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.3.6.8 HMC_MoveFlexLink（飞剪/旋切运动）

- 函数原型

  UDINT HMC_MoveFlexLink(LREAL dSlaveBaseDist, LREAL dSlaveAddDist, LREAL dMasterSumDist, LREAL dSlaveBeforeDist, LREAL dSlaveAfterDist, LREAL dMasterAccDist, LREAL dMasterDecDist, LREAL dSRatio，USINT ucSlaveAxis，USINT ucMasterAxis，USINT ucMode，LREAL dTrigMes)

- 功能说明

  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。

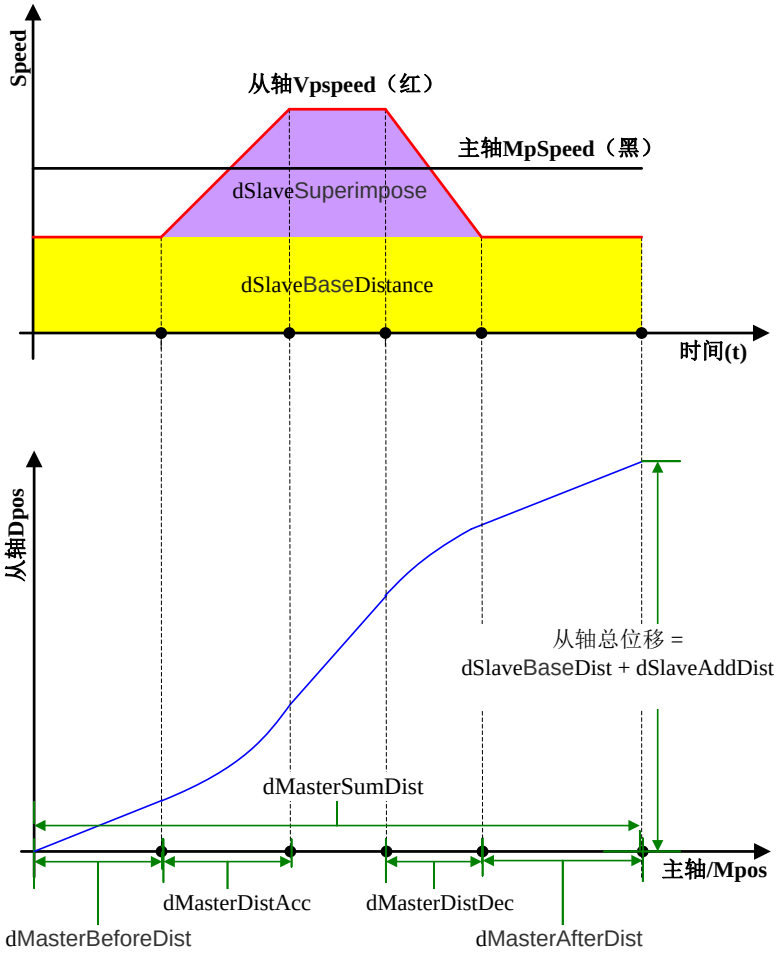


图 142　主/从轴示意图

■　输入参数

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 ∈ (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

输入参数	类型	含义
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）

- 返回值

返回值	类型	含义
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 补充说明

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

4.4 高级应用指令

4.4.1 运动叠加

4.4.1.1 HMC_Superimpose（叠加）

- 函数原型

  UDINT HMC_Superimpose(USINT ucSlaveAxis, USINT ucMasterAxis0, USINT ucMasterAxis1)

- 功能说明

  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 $\Delta \mathrm{Mpos}_{\mathrm{M0}}$、$\Delta \mathrm{Mpos}_{\mathrm{M1}}$，则从轴 Dpos 增量位移为：

$$\Delta \mathrm{Mpos}_{\mathrm{S}} = \Delta \mathrm{Mpos}_{\mathrm{M0}} + \Delta \mathrm{Mpos}_{\mathrm{M1}}$$

  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。

- 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号
ucMasterAxis0	USINT	第一个主轴轴号
ucMasterAxis1	USINT	第二个主轴轴号

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 应用举例

  示例 1：H 形同步齿形带并联机构运动控制

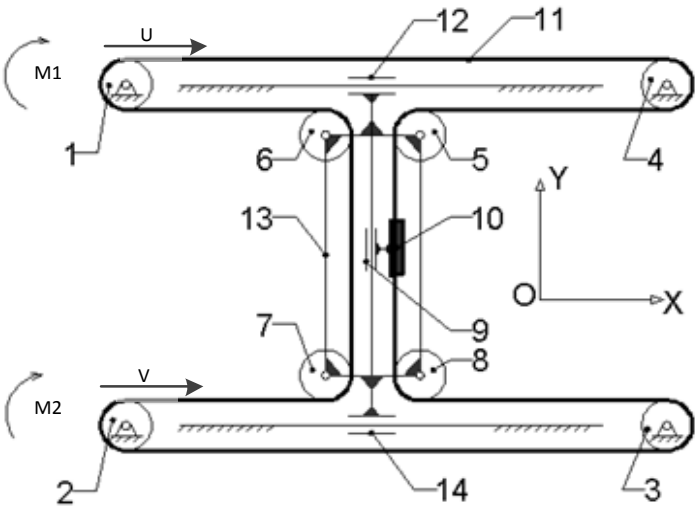


图 143 H 形并联机构简图

- 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)*)
```

4.4.1.2 HMC_SuperImposeEx（高级叠加）

- 函数原型

  UDINT HMC_SuperImpose(USINT ucSlaveAxis,

  USINT ucMasterAxis0,

  USINT ucMasterAxis1,

  DINT iRatioNumerator0,

  DINT iRatioDenominator0,

  DINT iRatioNumerator1,

  DINT iRatioDenominator1)

- 功能说明

  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 $\Delta \mathrm{Mpos}_{\mathrm{M0}}$、$\Delta \mathrm{Mpos}_{\mathrm{M1}}$，则从轴 Dpos 增量位移为：

$$\Delta \mathrm{Dpos}_{\mathrm{S}} = \left(\frac{\mathrm{iRatioNumerator0}}{\mathrm{iRatioDenominator0}}\right) \times \Delta \mathrm{Mpos}_{\mathrm{M0}} + \left(\frac{\mathrm{iRatioNumerator1}}{\mathrm{iRatioDenominator1}}\right) \times \Delta \mathrm{Mpos}_{\mathrm{M1}}$$

  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）。

- 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号
ucMasterAxis0	USINT	第一个主轴轴号
ucMasterAxis1	USINT	第二个主轴轴号
iRatioNumerator0	DINT	第一个主轴的叠加比例的分子
iRatioDenominator0	DINT	第一个主轴的叠加比例的分母
iRatioNumerator1	DINT	第二个主轴的叠加比例的分子
iRatioDenominator1	DINT	第二个主轴的叠加比例的分母

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 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。

■ 应用举例

示例 1：坐标变换

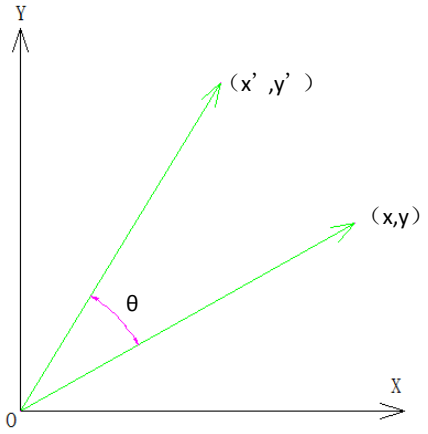


图 144 坐标变换示意图

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$\theta = 45^{\circ}$)：

```
(**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：

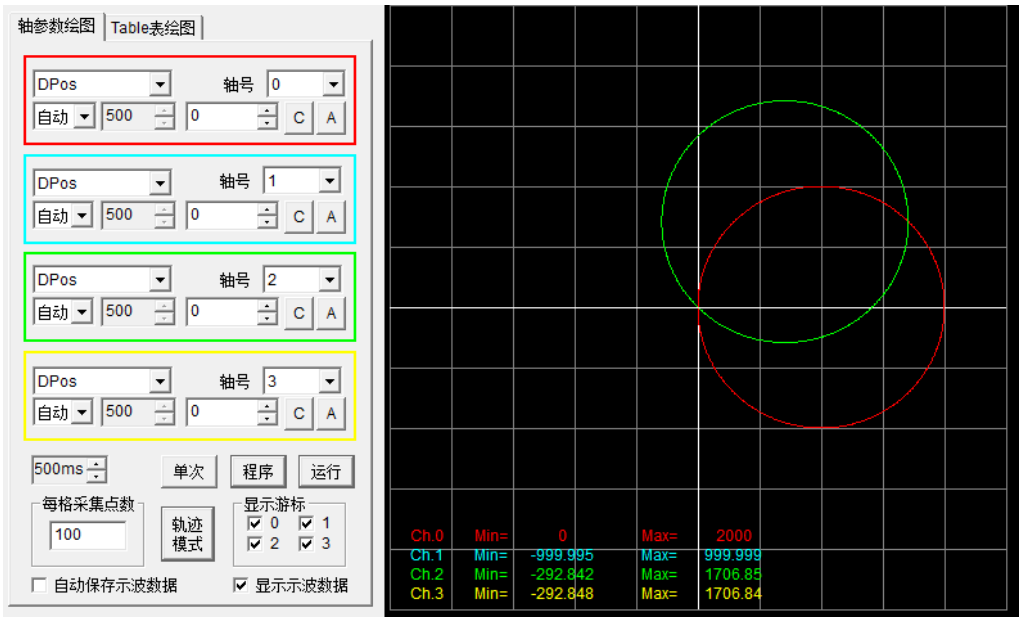


图 145 HMC_SuperImposeEx 实现坐标变换

示例 2：十字形同步齿形带并联机构运动控制

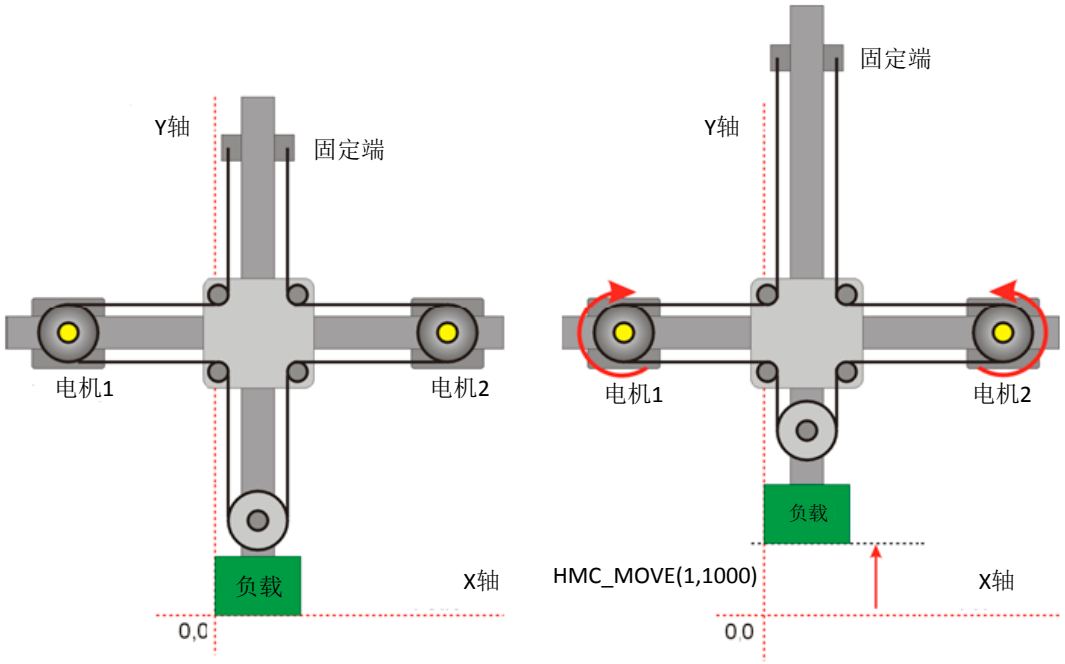


图 146 十字形同步齿形带并联机构

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*)
```

4.4.1.3 HMC_SetSuperimposeRatio（设置叠加比例）

- 函数原型

  UDINT HMC_SetSuperimposeRatio(USINT ucSlaveAxis,

  DINT iRatioNumerator0,

  DINT iRatioDenominator0,

  DINT iRatioNumerator1,

  DINT iRatioDenominator1)

- 功能说明

  设置叠加比例，关于叠加详见 HMC_Superimpose、HMC_SuperImposeEx。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。

- 输入参数

输入参数	类型	参数描述
ucSlaveAxis	USINT	从轴轴号
iRatioNumerator0	DINT	第一个主轴的叠加比例的分子

输入参数	类型	参数描述
iRatioDenominator0	DINT	第一个主轴的叠加比例的分母
iRatioNumerator1	DINT	第二个主轴的叠加比例的分子
iRatioDenominator1	DINT	第二个主轴的叠加比例的分母

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 应用举例

  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*)
```

4.4.1.4 HMC_GetSuperimposeMaster1（获取叠加主轴 1）

- 函数原型

INT HMC_GetSuperimposeMaster1(USINT ucSlaveAxis)

- 功能描述

  获取叠加的第一个主轴。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
叠加的第一个主轴	INT	成功
-1	INT	当前未进行叠加运动或函数参数错误

- 指令类型

  非轴运动缓存指令。

4.4.1.5 HMC_GetSuperimposeMaster2（获取叠加主轴 2）

- 函数原型

  INT HMC_GetSuperimposeMaster2(USINT ucSlaveAxis)

- 功能说明

  获取叠加的第二个主轴。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
叠加的第二个主轴	INT	成功
-1	INT	当前未进行叠加运动或函数参数错误

- 指令类型

  非轴运动缓存指令。

4.4.1.6 HMC_GetSuperimposeRatio1（获取叠加主轴 1 比例）

- 函数原型

  LREAL HMC_GetSuperimposeRatio1(USINT ucSlaveAxis)

- 功能说明

  获取叠加的第一个主轴比例。

- 输入参数

输入参数	类型	参数描述

输入参数	类型	参数描述
ucAxisID	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
叠加的第一个主轴比例	LREAL	成功

- 指令说明

非轴运动缓存指令。

4.4.1.7 HMC_GetSuperimposeRatio2（获取叠加主轴 2 比例）

- 函数原型

LREAL HMC_GetSuperimposeRatio2(USINT ucSlaveAxis)

- 功能说明

获取叠加的第二个主轴比例。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
叠加的第二个主轴比例	LREAL	成功

- 指令类型

非轴运动缓存指令。

4.4.1.8 HMC_SetAddAxis（设定和运动轴）

- 函数原型

UDINT HMC_SetAddAxis(USINT ucBaseAxisID, USINT ucAddAxisID)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucBaseAxisID	USINT	基础轴轴号，0~63
ucAddAxisID	USINT	叠加轴轴号（有效关联轴号：0~63；255 表示取消关

输入参数	类型	参数描述
		联。）

- 返回值

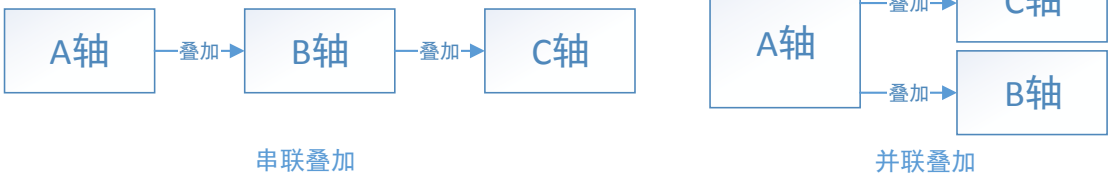
返回值	类型	参数描述
成功：0 错误：错误码	UDINT	--

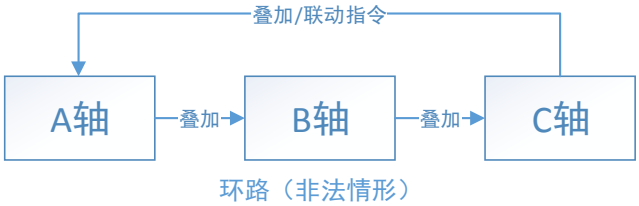
- 指令类型

非轴运动缓存指令。

- 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。

4.4.1.9 HMC_GetAddAxis（获取和运动叠加轴）

- 函数原型

UDINT HMC_GetAddAxis(USINT ucBaseAxisID)

- 功能说明

用于获取和运动叠加轴轴号，255 表示未关联任何叠加轴。

- 输入参数

输入参数	类型	参数描述
ucBaseAxisID	USINT	基础轴轴号，0~63

- 返回值

返回值	类型	参数描述
叠加轴轴号	UDINT	255 表示未关联任何叠加轴

- 指令类型

  非轴运动缓存指令。

4.4.2 高速脉冲捕获

4.4.2.1 HMC_CaptureConfig（设置高速捕获）

- 函数原型

  UDINT HMC_CaptureConfig (USINT ucChlID,

  USINT bTrigEdge,

  USINT ucTrigType,

  USINT ucTrigID,

  USINT ucCaptureAxis,

  USINT bWinModeEnable,

  LREAL dWinStart,

  LREAL dWinEnd)

- 功能说明

  该函数主要用来配置高速捕获通道。高速捕获功能主要用来实现通过外部触发信号（高速 DI 或 Z 相）快速获取待捕获轴当前位置的功能，该过程由底层硬件（FPGA）完成，不存在软件处理时的滞后误差，所以具有很高的捕获精度。

  用户通过指定待捕获轴、捕获信号的类型等参数，配置高速捕获通道。配置成功后，高速捕获功能立即生效。当捕获信号产生时，高速捕获功能被触发，此时待捕获轴的 Mpos 值被记录下来，之后可通过 HMC_CaptureGetMpos 指令获取此位置。

  在高速捕获功能使用过程中，可通过 HMC_CaptureGetState 指令实时查询高速捕获状态，以判断高速捕获通道是否被触发。

  当高速捕获功能的窗口模式（bWinModeEnable）禁用时，在任意时刻捕获信号到来之时都将触发高速捕获功能进行位置捕获；当窗口模式（bWinModeEnable）使能时，仅当待捕获轴的 Mpos 值在 dWinStart 与 dWinEnd 之间（窗口范围之内），捕获信号到来时高速捕获功能才会被触发。在该窗口区间之外时，即使捕获信号到来，高速捕获功能也不会被触发。

- 输入参数

输入参数	类型	参数描述
ucChlID	USINT	高速捕获通道号 MC1008&MC1004&MC1004L（0~7） MC1002R&MC1002E (0~1)
bTrigEdge	USINT	捕获触发信号的有效沿，0：上升沿，1：下降沿
ucTrigType	USINT	捕获触发信号类型，0：轴接口 Z 相信号，1：高速 DI 信号
ucTrigID	USINT	捕获触发信号的通道号，即高速 DI 通道号或 Z 相所在轴号
ucCaptureAxis	USINT	待捕获轴号，当捕获信号有效时获取该轴 MPos 值
bWinModeEnable	USINT	高速捕获窗口使能，0：禁止，1：窗口内使能，2：

输入参数	类型	参数描述
		窗口外使能
dWinStart	LREAL	捕获窗口开启位置，待捕获轴的 Mpos
dWinEnd	LREAL	捕获窗口关闭位置，待捕获轴的 Mpos

- 返回值

返回值	类型	参数描述
0	UDINT	配置成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 补充说明

（1） MC1008/MC1004 支持 8 路独立的高速捕获功能，MC1002R/MC1002E 支持 2 路高速捕获。

（2） 选择 DI 作为触发捕获信号时，只可选择 DI0~DI11（高速 DI）；选择 Z 相作为触发捕获信号时，可选择处于空闲状态的 Z 相接口。

（3） 高速捕获待捕获轴可为任意轴类型。当待捕获轴为伺服轴（20）、增量编码器（30/31/32）、SSI 编码器（33）或安川绝对值编码器时，捕获过程由硬件 FPGA 完成，不存在软件滞后偏差，精度较高；虚轴、总线轴类型捕获过程由软件完成，存在时间滞后（最大一个伺服周期），捕获精度稍差。

（4） 由于 DI 输入较 Z 相输入偏慢，因此当用作高速设备时建议选择 Z 相作为捕获触发信号，当使用低速设备时使用 DI 输入。

（5） 两路独立的高速捕获可同时捕获同一个轴。

（6） 对于高速捕获的窗口，需满足 dWinStart≤dWinEnd，可正可负。

- 应用举例

假如某设备外接一数字量传感器，算法要获取滑块到达该传感器时的精确位置，可配置一高速捕获通道，当检测到传感器有效时，捕获此时 Mpos 值。

```
axis0.AxisType := 30;     (*设定轴 0 为编码器轴*)


cap_chID := 0;            (*选择高速捕获通道 0*)
cap_type := 0;            (*捕获信号选择 Z 相输入*)
cap_trigID := 0;       (*选择轴 0、Z 相作为捕获信号*)
cap_edge := 0;         (*选择轴 0 的 Z 相上升沿触发高速捕获*)
cap_axis := 0;         (*捕获轴 0 的位置*)
cap_win_mode := 1;     (*使能窗口内有效*)
cap_winstart := -100;(*左窗口位置*)
cap_winend := 1000;    (*右窗口位置*)
```

```
    cap_status := HMC_CaptureConfig (cap_chID, cap_edge, cap_type, cap_trigID,
cap_axis, cap_win_mode, cap_winstart, cap_winend);


    while capture_state <> 3 do      (*等待捕获完成*)
    capture_state := HMC_CaptureGetState(cap_chID);       (*查询捕获状态*)
    end_while


    capture_pos := HMC_CaptureGetMpos(cap_chID);       (*读取捕获值*)
```

4.4.2.2 HMC_CaptureGetMpos（获取高速捕获 Mpos）

- 函数原型

  LREAL HMC_CaptureGetMpos(USINT ucChlID)

- 功能说明

  返回 ucChlID 指定高速捕获通道获取到的 Mpos。

- 输入参数

输入参数	类型	参数描述
ucChlID	USINT	高速捕获通道号

- 返回值

返回值	类型	参数描述
Mpos 值	LREAL	捕获到的 Mpos 值

- 指令类型

  非轴运动缓存指令。

4.4.2.3 HMC_CaptureGetState（获取高速捕获状态）

- 函数原型

  USINT HMC_CaptureGetState(USINT ucChlID)

- 功能说明

  返回某个高速捕获通道的状态，参见返回值参数描述。

- 输入参数

输入参数	类型	参数描述
ucChlID	USINT	高速捕获通道号

- 返回值

返回值	类型	参数描述
0	USINT	正在配置
1	USINT	不在窗口期（等待窗口打开，或者窗口已经关闭，此状态时窗口模式应该被使能）
2	USINT	正在等待捕获信号

返回值	类型	参数描述
3	USINT	空闲，捕获完成或未被使用

- 指令类型

  非轴运动缓存指令。

4.4.2.4 HMC_CaptureGetError（获取高速捕获错误）

- 函数原型

  UDINT HMC_CaptureGetError(USINT ucChlID)

- 功能说明

  返回某个高速捕获通道的错误码。

- 输入参数

输入参数	类型	参数描述
ucChlID	USINT	高速捕获通道号

- 返回值

返回值	类型	参数描述
0	UDINT	正确
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

4.4.3 高速到位输出

4.4.3.1 HMC_SwitchConfig(设置高速输出)

- 函数原型

  UDINT HMC_SwitchConfig(USINT ucChIID,

  USINT ucOutputMode,

  USINT ucOutputID,

  USINT ucTrigAxis,

  USINT bSwitchValue,

  POINTER TO LREAL pMpos,

  UDINT uiMposNum)

- 功能说明

  该指令主要完成对高速到位输出的配置。高速到位输出功能（简称 Switch）可以设置一系列目标位置，当被选择的伺服轴或编码器轴到达当前目标值时，被选择的输出通道电平发生翻转。其中，输出通道的首个电平值由 bSwitchValue 指定。该系列目标值被顺序执行。

  该指令可实现 DO 输出快速响应轴接口位置的功能，无需组态其它逻辑，操作简单、响应快速。

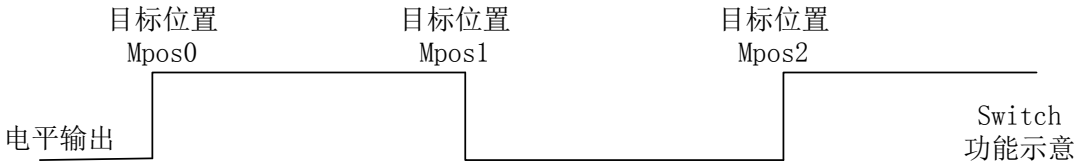


图 147 功能示意图

- 输入参数

输入参数	类型	参数描述
ucChIID	USINT	高速输出通道号 MC1008&MC1004&MC1004L（0~3） MC1002R&MC1002E (0)
ucOutputMode	USINT	输出类型选择，0：DO 输出；1：Axis Z 相输出
ucOutputID	USINT	输出位置选择，使用该轴 Z 相或该 DO 输出。
ucTrigAxis	USINT	触发轴，switch 比较该轴与目标 Mpos 的值决定输出电平
bSwitchValue	USINT	首次输出电平：0 或 1
pMpos	POINTER TO LREAL	目标位置存放地址，一般为一个数组的地址，该数组存放目标 Mpos 设定值。
uiMposNum	UDINT	目标 Mpos 序列的个数

- 返回值

返回值	类型	参数描述
0	UDINT	配置成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 补充说明

（1） 针对不同的模块，通道数有所不同，即 ucChIID 数不同。对于 MC1008、MC1004、MC1004L 共有 4 路高速到位输出通道，MC1002R、MC1002E 只有 1 路高速到位输出通道。

（2） 针对不同的模块，可选输出位置不同。对于 MC1008,共有 8 路轴接口或 8 路 DO 输出可供选择；对于 MC1004,共有 4 路轴接口和 8 路 DO 可供选择；对于 MC1002,共有 2 路轴接口和 8 路 DO 通道可供选择。

（3） 该指令 ucTrigAxis 触发轴只支持伺服轴、编码器轴、步进轴类型，不支持虚轴、总线轴类型。

（4） 该指令可设置一系列目标位置，该位置存于数组中，并将数组的首地址写入 pMpos 参数，Switch 将依次执行。该命令阻塞执行，即未到达当前目标位置时，后续目标位置将不响应。当目标位置被执行，该命令才被释放。

（5） 在位置数组的设置中，应遵循数据递增方向同电机运行方向保持一致，否则会产生异常。例如当前位置为 0，电机将向负向运行，则目标位置设定为-10、-100、-500 可工作正常。但设定为-10、10000、-500，则功能块会出现异常。

（6） 在配置高速到位输出前，触发轴当前位置 Mpos 必须在触发位置 Mpos 序列的起始位置之前。不允许在配置高速到位输出后，使用 Defpos 定义轴当前位置位于触发位置 Mpos 序列的起始位置之前。

（7） 当采集轴为绝对值编码器轴时，命令输出与实际位置会存在一些偏差。例如设定目标位置为 1000，相临两次从编码器获得的数据分别为 990、1100 时，在 1100 位置会产生输出跳变。

（8） 在 Switch 命令执行过程中，不得人为改变 Mpos 的值。例如，在 Switch 命令未完成之前对 DefPos 的执行会导致该功能的异常。如执行了修改 Mpos 的相关指令，应执行一次 HMC_CloseSwitch 命令，关闭高速到位输出并重新配置。

■ 应用举例

在某激光雕刻应用中，要求在钢板上每隔 100 单位的距离打长度为 100 单位的矩形孔。从 0 位开始，当到达 100 时激光打开，到达 200 时激光关闭，到达 300 时激光打开….直到到达 1000。

在上述应用中，如使用 HMC_SwitchConfig 指令，用户只需关心电机的运行，无需关注激光的开闭。该指令会快速及时的响应被采集轴的位置。

```
FOR i:=0 TO 9 BY 1 DO          (*填充目标位置设定值*)
target_pos[i] := (i+1)*100;
END_FOR


axis0.AxisType := 20;          (*设定轴 0 为伺服轴*)
(*轴 0 定位到起始位置...略...*)
ChIID := 0;                    (*选择 switch 通道 0*)
OutputMode := 1;               (*使用 Z 相输出*)
OutputID := 0;                 (*使用轴 0 的 Z 相作为 switch 的输出*)
TrigAxis := 0;                 (*采集轴 0 的当前位置*)
SwitchValue := 0;              (*首个输出为低电平*)
ppMpos := ADR (target_pos);    (* 获取目标位置数组的地址*)
MposNum := 10;                 (*设定设置的有效目标地址个数共 10 个*)


(*配置 SwitchConfig*)
switch_ack :=HMC_SwitchConfig(ChIID, OutputMode, OutputID, TrigAxis,
SwitchValue, ppMpos, MposNum);
(*轴 0 开始位移...略...*)
```

4.4.3.2 HMC_GetSwitchState(获取高速输出状态)

■ 函数原型

USINT HMC_GetSwitchState(USINT ucChIID)

■ 功能说明

获取指定 ID 的 SwitchConfig 的当前状态。

- 输入参数

输入参数	类型	参数描述
ucChIID	USINT	switchconfig 通道号

- 返回值

返回值	类型	参数描述
0	USINT	当前通道正在工作
3	USINT	当前通道空闲，表示已完成或未配置

- 指令类型

  非轴运动缓存指令

4.4.3.3 HMC_GetSwitchNum(获取已输出数目)

- 函数原型

  UDINT HMC_GetSwitchNum(USINT ucChIID)

- 功能说明

  获取指定 ID 的 SwitchConfig 已执行的段数。

- 输入参数

输入参数	类型	参数描述
ucChIID	USINT	switchconfig 通道号

- 返回值

返回值	类型	参数描述
已完成执行的段数	UDINT	已完成执行的段数

- 指令类型

  非轴运动缓存指令

4.4.3.4 HMC_SwitchTZConfig(设置高速 TZ 输出)

- 函数原型

  UDINT HMC_SwitchTZConfig(USINT ucChIID,
  USINT ucOutputAxis,
  USINT ucTrigAxis,
  USINT ucTimeMode,
  LREAL dTime,
  USINT bSwitchValue,
  POINTER TO LREAL pMpos,
  UDINT uiMposNum)

- 功能说明

SwitchTZ 功能是 Switch 功能的增强，即到达指定位置使指定轴的 Z 相输出产生响应。当时间使能被禁止时，该功能与相当于 Switch 使用 Z 相输出。开启时间使能 (ucTimeMode:= 1) 后，被采集轴（由 ucTrigAxis 设定）到达指定位置（pMpos 所指的数组设定）后，输出与 bSwitchValue 相同的电平值。电平持续 dTime 时间后恢复至相反电平，继续等待下一个目标位置，直到完成 uiMposNum 段的输出。

该指令与 Switch 功能类似，主要完成在指定位置完成固定时间的电平输出。AT 工程无需关心具体的操作逻辑，由底层硬件完成电平输出与当前位置的完美结合。

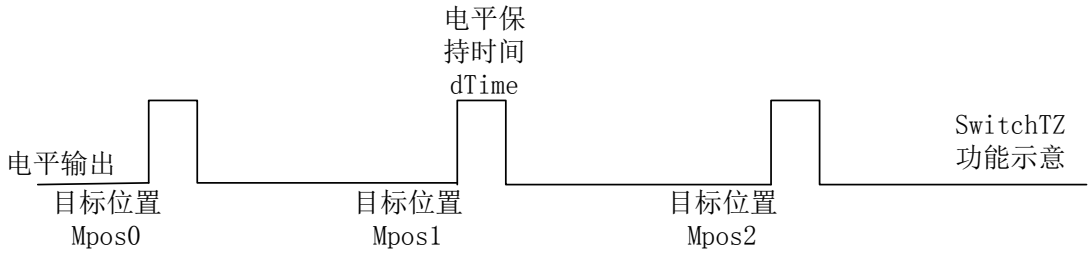


图 148 功能示意图

■ 输入参数

输入参数	类型	参数描述
ucChIID	USINT	高速输出通道号 MC1008（0~1） 其它型号不支持
ucOutputAxis	USINT	输出轴号（使用该轴 Z 相进行输出）
ucTrigAxis	USINT	触发轴，switch 比较该轴与目标 Mpos 的值决定输出电平
ucTimeMode	USINT	时间模式设置。0：关闭；1：开启
dTime	LREAL	输出电平保持时间，单位：us，有效范围：1～100,000，分辨率：0.1 μs。
bSwitchValue	USINT	时间模式关闭时：首个输出电平值（0 或 1） 时间模式开启时：输出脉冲电平值（0 或 1）
pMpos	POINTER TO LREAL	目标位置存放地址，一般为一个数组的地址，该数组存放目标 Mpos 设定值。
uiMposNum	UDINT	目标 Mpos 序列的个数

■ 返回值

返回值	类型	参数描述
0	UDINT	配置成功
其它	UDINT	错误码

■ 指令类型

非轴运动缓存指令。

■ 补充说明

（1） MC 系列产品中，只有 MC1008 支持该功能，其它模块不支持。

（2） 该指令只支持伺服轴、编码器轴、步进轴的采集。不支持虚轴、总线轴使用。

（3） 该指令可设置一系列目标位置，该位置存于数组中，并将数组的首地址写入 pMpos 参数，SwitchTZ 将依次执行。该命令阻塞执行，即未到达当前目标位置时，后续目标位置将不响应。只有最后一个目标位置被执行，该命令才可释放。

（4） 在位置数组的设置中，应遵循数据递增方向同电机运行方向保持一致，否则会产生异常。例如当前位置为 0，电机将向负向运行，则目标位置设定为-10、-100、-500 可工作正常。但设定为-10、10000、-500，则功能块会出现异常。

（5） 在位置设定中应注意有效电平维持时间同两次有效位置的间隔。例如电机速度为 V，电平有效时间设为 T，当前位置为 S0,下一有效位置为 S1，则应保证 S1-S0>V*T。否则，如果下一刻的目标位置在 T 时间内到达，SwitchTZ 模块将不响应。严重时发生阻塞。

（6） 在配置高速到位输出前，触发轴当前位置 Mpos 必须在触发位置 Mpos 序列的起始位置之前。不允许在配置高速到位输出后，使用 Defpos 定义轴当前位置位于触发位置 Mpos 序列的起始位置之前。

（7） 当采集轴为绝对值编码器轴时，命令输出与实际位置会存在一些偏差。例如设定目标位置为 1000，相临两次从编码器获得的数据分别为 990、1100 时，在 1100 位置会产生输出跳变。

（8） 在 SwitchTZ 命令执行过程中，不得人为改变 Mpos 的值。例如，在 SwitchTZ 命令未完成之前对 DefPos 的执行会导致该功能的异常。如执行了修改 Mpos 的相关指令，应执行一次 HMC_CloseSwitchTZ 命令，关闭高速到位输出并重新配置。

4.4.3.5 HMC_GetSwitchTZState(获取高速 TZ 输出状态)

- 函数原型

  USINT HMC_GetSwitchTZState (USINT ucChIID)

- 功能说明

  获取指定 ID 的 SwitchTZConfig 的当前状态。

- 输入参数

输入参数	类型	参数描述
ucChIID	USINT	switchTZConfig 通道号

- 返回值

返回值	类型	参数描述
0	USINT	当前通道正在工作
3	USINT	当前通道空闲，表示已完成或未配置

- 指令类型

  非轴运动缓存指令。

  仅 MC1008 支持。

4.4.3.6 HMC_GetSwitchTZNum(获取 TZ 已输出数目)

- 函数原型

  UDINT HMC_GetSwitchTZNum(USINT ucChIID)

- 功能说明

获取指定 ID 的 SwitchConfig 已执行的段数。

- 输入参数

输入参数	类型	参数描述
ucChIID	USINT	switchTZconfig 通道号

- 返回值

返回值	类型	参数描述
已完成执行的段数	UDINT	已完成执行的段数

- 指令类型

非轴运动缓存指令。

仅 MC1008 支持。

4.4.3.7 HMC_CloseSwitch(关闭高速输出)

- 函数原型

UDINT HMC_CloseSwitch(USINT ucChIID)

- 功能说明

关闭对 CloseSwitch 的操作，常用于 CloseSwitch 阻塞或需要重新配置时。

- 输入参数

输入参数	类型	参数描述
ucChIID	USINT	switchconfig 通道号

- 返回值

返回值	类型	参数描述
0	UDINT	关闭成功
其它	UDINT	错误码

- 指令类型

非轴运动缓存指令。

4.4.3.8 HMC_CloseSwitchTZ(关闭高速 TZ 输出)

- 函数原型

UDINT HMC_CloseSwitchTZ(USINT ucChIID)

- 功能说明

关闭对 SwitchTZ 的操作，常用于 SwitchTZ 阻塞或需要重新配置时。

- 输入参数

输入参数	类型	参数描述
ucChIID	USINT	switchTZconfig 通道号

- 返回值

返回值	类型	参数描述
0	UDINT	关闭成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

  仅 MC1008 支持。

4.4.4 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，包括直连、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。

4.4.4.1 HMC_NormalSwitch1Dconfig（设置一维低速输出）

- 功能说明

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。

- 输入参数

输入参数	类型	含义
ucChIID	USINT	普通到位输出通道号，0~7 通道号与 HMC_NormalSwitch2DConfig 共用
ucOutputID	USINT	输出位置选择，0～63
ucTrigAxis	USINT	采集轴，switch 比较该轴与目标位置值决定输出电平
ucTrigPosType	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	USINT	输出方式：

输入参数	类型	含义
		0：电平翻转 1：可设脉宽的脉冲
bSwitchValue	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	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	POINTER TO LREAL	目标位置存放地址，该数组存放目标位置设定值
uiPosNum	UDINT	目标 Mpos 序列的个数

- 返回值

返回值	类型	参数描述
0	UDINT	配置成功
其它	UDINT	错误码

4.4.4.2 HMC_NormalSwitch2DConfig（设置二维低速输出）

- 功能说明

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。

- 输入参数

输入参数	类型	含义
ucChIID	USINT	普通到位输出通道号，0~7 通道号与 HMC_NormalSwitch1DConfig 共用
ucOutputID	USINT	输出位置选择，0～63
ucTrigAxisX	USINT	采集轴 X，switch 比较该轴与 X 目标位置值决定输出电平
ucTrigAxisY	USINT	采集轴 Y，switch 比较该轴与 Y 目标位置值决定输出电平
ucTrigPosType	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	USINT	输出方式： 0：电平翻转

输入参数	类型	含义
		1：可设脉宽的脉冲
bSwitchValue	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	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	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	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	UDINT	目标 Mpos 序列的个数
dThreshold	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	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）

- 返回值

返回值	类型	参数描述
0	UDINT	配置成功
其它	UDINT	错误码

4.4.4.3 HMC_GetNormalSwitchNum（获取已输出数目）

- 功能说明

  获取指定 ID 的低速到位输出已执行的段数。

- 输入参数

输入参数	类型	含义
ucChIID	USINT	低速输出通道号，0~7

- 返回值

返回值	类型	参数描述
已完成执行的段数	UDINT	已完成执行的段数

4.4.4.4 HMC_GetNormalSwitchState（获取低速输出状态）

- 功能说明

  获取指定 ID 的低速输出的当前状态。

- 输入参数

输入参数	类型	含义
ucChIID	USINT	低速输出通道号，0~7

- 返回值

返回值	类型	含义
0	USINT	当前通道正在工作
3	USINT	当前通道空闲，表示已完成或未配置

4.4.4.5 HMC_CloseNormalSwitch（关闭低速输出）

- 功能说明

  关闭该 ID 的低速到位输出。

- 输入参数

输入参数	类型	含义
ucChIID	USINT	低速输出通道号，0~7

- 返回值

返回值	类型	参数描述
0	UDINT	关闭成功
其它	UDINT	错误码

4.4.5 示波器功能指令

4.4.5.1 HMC_SetSampleVar（设置采样变量）

- 函数原型

  UDINT HMC_ SetSampleVar (USINT ucSampleChl,

  POINTER TO BYTE pVar0,

  POINTER TO BYTE pVar1,

  POINTER TO BYTE pVar2,

  POINTER TO BYTE pVar3,

  UDINT uiVarType0,

  UDINT uiVarType1,

  UDINT uiVarType2,

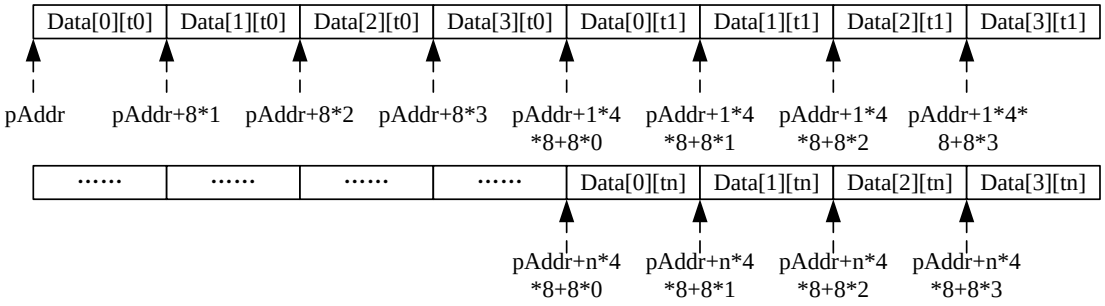
  UDINT uiVarType3)

- 功能说明

  MC1004 和 MC1008 型号控制器支持 4 路高速采样通道，MC1002R 和 MC1002E 型号控制器支持 1 路高速采样通道。每个采样通道有 4 个子通道，最多可以采集 4 个数据。

  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：



Data[i][ti]:第i个子通道在第i次采样时刻的采样数据

图 149 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。

- 输入参数

输入参数	类型	参数描述
ucSampleChl	USINT	采样通道号 MC1008&MC1004 (0~3) MC1002R&MC1002E (0)
pVar0	POINTER TO BYTE	待采集的第 0 子通道数据的地址
pVar1	POINTER TO BYTE	待采集的第 1 子通道数据的地址
pVar2	POINTER TO BYTE	待采集的第 2 子通道数据的地址
pVar3	POINTER TO BYTE	待采集的第 3 子通道数据的地址
uiVarType0	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	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	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	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

- 返回值

返回值	类型	参数描述

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 应用举例

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

图 150 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 1 ms、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。

4.4.5.2 HMC_SetSamplePara（设置采样参数）

- 函数原型

  UDINT HMC_SetSamplePara (USINT ucSampleChl,

  UDINT uiTime,

  UDINT uiNum,

  POINTER TO LREAL pRecordArea)

- 功能说明

  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，MC 控制器支持 0.25、0.5、1、2、4 ms 伺服周期。

  采样功能详细描述请查看 HMC_ SetSampleVar。

- 输入参数

输入参数	类型	参数描述
ucSampleChl	USINT	采样通道
uiTime	UDINT	记录间隔时间。不可为 0，单位：伺服周期
uiNum	UDINT	记录点数

输入参数	类型	参数描述
pRecordArea	POINTER TO LREAL	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）

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 应用举例

  参考 HMC_ SetSampleVar

4.4.5.3 HMC_GetSampleData（获取采样数据）

- 函数原型

  LREAL HMC_GetSampleData (USINT ucSampleChl, USINT ucChannel, UDINT uiDataNum)

- 功能说明

  获取指定通道的某子通道的序号为 uiDataNum 的采样数据。

- 输入参数

输入参数	类型	参数描述
ucSampleChl	USINT	采样通道
ucChannel	USINT	子通道号（0~3）
uiDataNum	UDINT	数据号（该子通道的第 uiDataNum 个数据），数据号从 0 开始

- 返回值

返回值	类型	参数描述
采样数据	LREAL	-
其它	LREAL	错误码

- 指令类型

  非轴运动缓存指令

4.4.5.4 HMC_TriggerSample（触发采样）

- 函数原型

  UDINT HMC_ TriggerSample (USINT ucSampleChl)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucSampleChl	USINT	采样通道

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 应用举例

  参考 HMC_ SetSampleVar

4.4.5.5 HMC_DisableSample（禁止采样）

- 函数原型

  UDINT HMC_ DisableSample (USINT ucSampleChl)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucSampleChl	USINT	采样通道

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.4.5.6 HMC_TriggerScope（触发示波器）

- 函数原型

  UDINT HMC_TriggerScope ()

- 功能说明

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。

- 输入参数

  无

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 应用举例

  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*)
```

4.4.5.7 HMC_GetSampleNum（获取已采样数目）

- 函数原型

  UDINT HMC_GetSampleNum (USINT ucSampleChl)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucSampleChl	USINT	采样通道

- 返回值

返回值	类型	参数描述

返回值	类型	参数描述
已采样数目	UDINT	-
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

4.4.5.8 HMC_SetSampleSingleVar（设置单个采样变量）

- 函数原型

  UDINT HMC_SetSampleSingleVar (USINT ucSampleChl,

  POINTER TO BYTE pVar,

  UDINT uiVarType)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucSampleChl	USINT	采样通道
pVar	POINTER TO BYTE	待采集的数据的地址
uiVarType	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

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

4.4.6 运动保持功能指令

4.4.6.1 HMC_HoldConfig(设置强制速度开关)

- 函数原型

  UDINT HMC_ HoldConfig (USINT ucAxisID, INT sDIChan)

- 功能说明

  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：

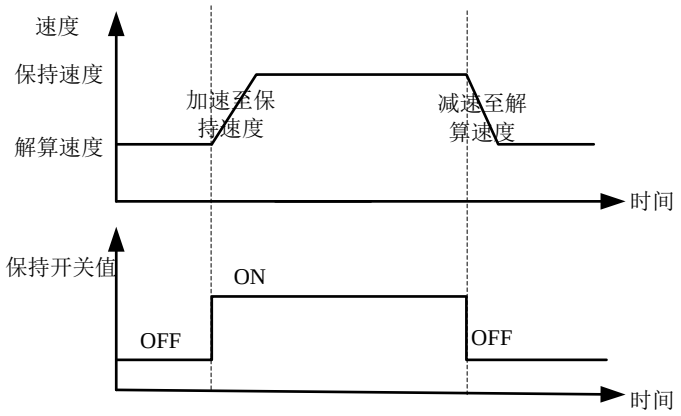


图 151　保持速度大于解算速度

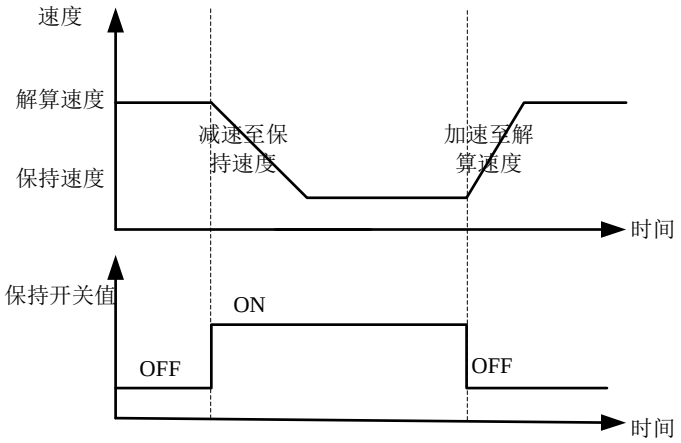


图 152　保持速度小于解算速度

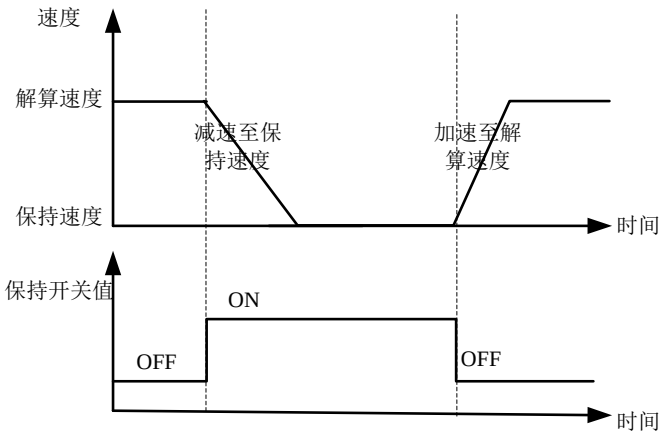


图 153 保持速度等于 0

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
sDiChan	INT	关联的 DI 通道（0~63）参数为-1 时表示取消 HoldSpeed（强制 Speed）开关。当取消 HoldSpeed 开关后，原来关联的 DI 通道的值将不再影响该轴的 HoldSpeed 状态，也就是即便当前处于 HoldSpeed 状态也会立即解除

- 返回值

返回值	类型	参数描述
0	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

非轴运动缓存指令。

- 应用举例

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*)
```

4.4.7 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。

（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。

（a）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。

（b）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

（c）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：

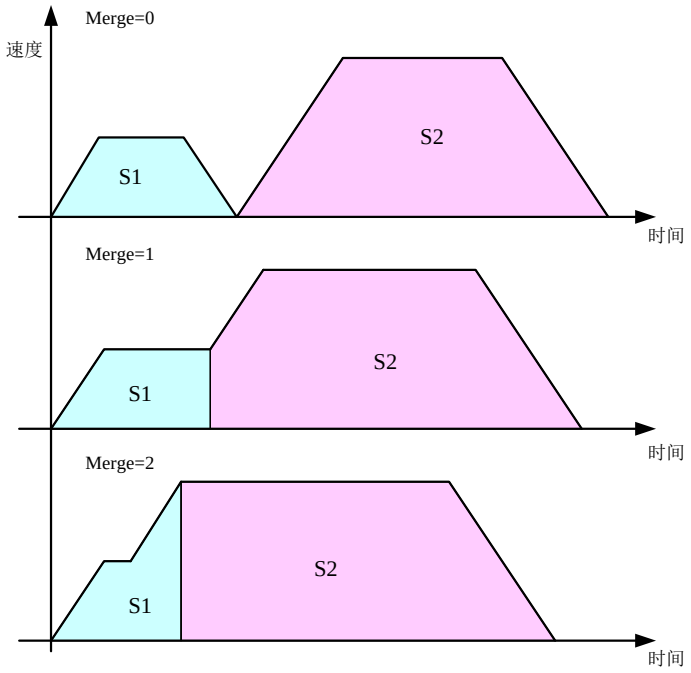


图 154 Merge 功能示意图

（c）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。

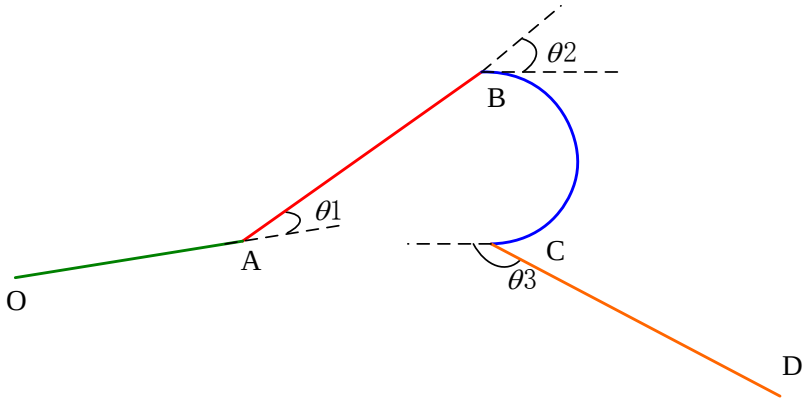


图 155 运动轨迹示意图

$\theta 1$ 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 156 所示：

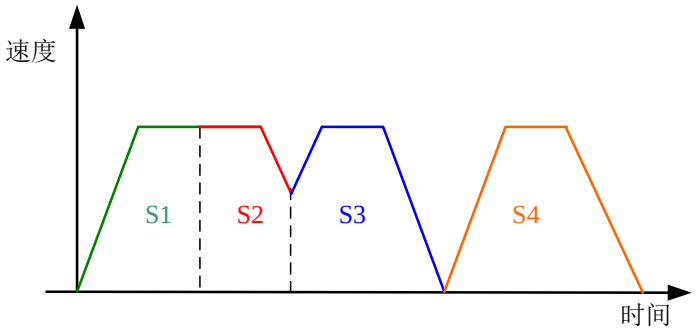


图 156 合轴速度示意图

（2）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。

（a）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。

（b）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。

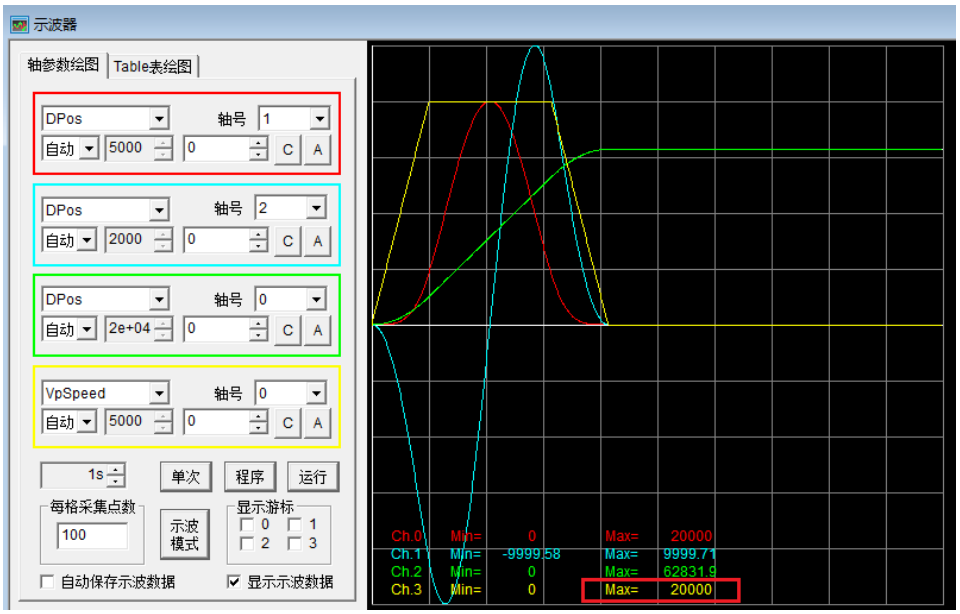


图 157 运动轨迹示意图

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。

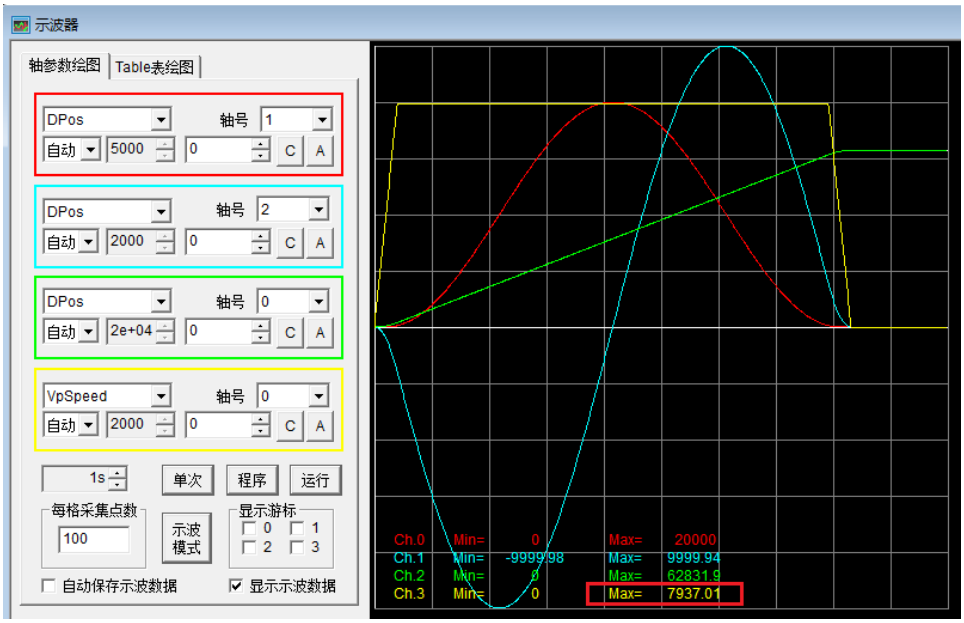


图 158 运动轨迹示意图

4.4.7.1 HMC_SetMerge（设置融合功能）

- 函数原型

  UDINT HMC_ SetMerge (USINT ucAxisID, USINT ucMerge)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucMerge	USINT	Merge 模式，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

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其他值	UDINT	错误码

- 指令类型

非轴运动缓存指令

■ 应用举例

单轴指令 Merge（两轴指令的举例见设置融合角章节）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）：

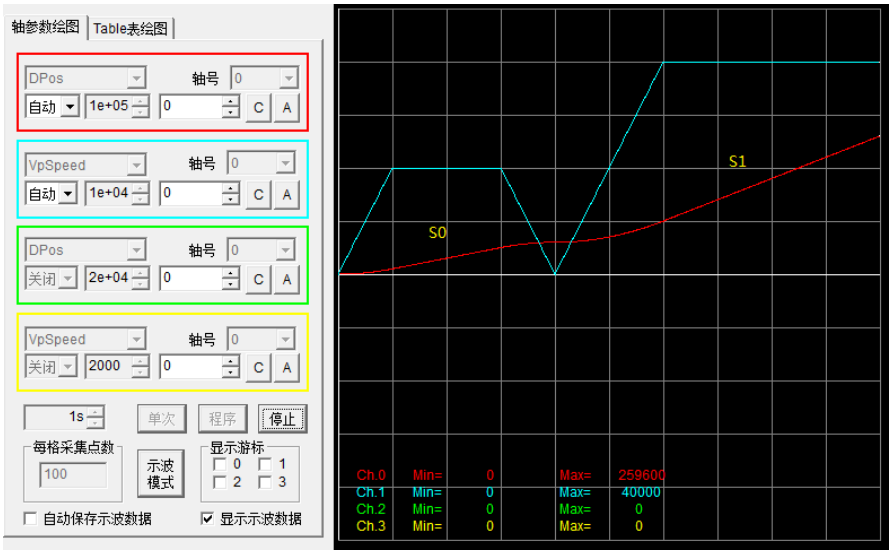


图 159 Merge 为 0 示意图

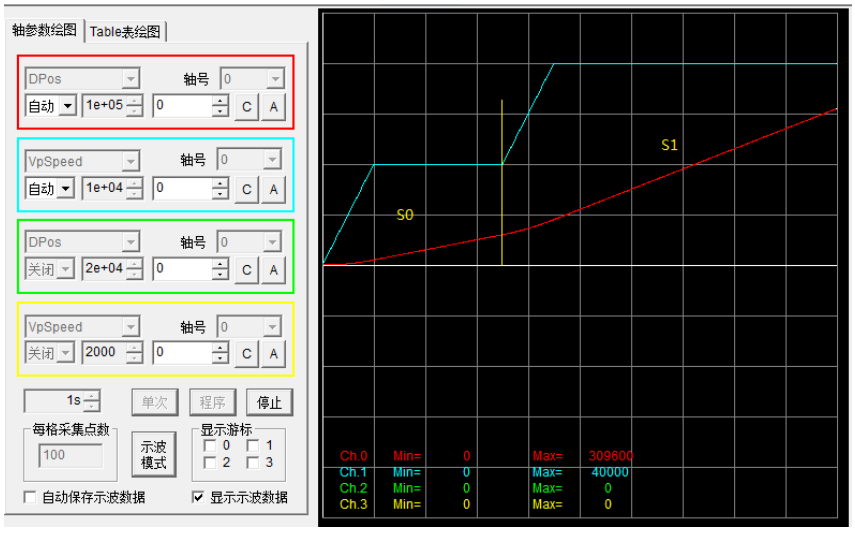


图 160 Merge 为 1 示意图

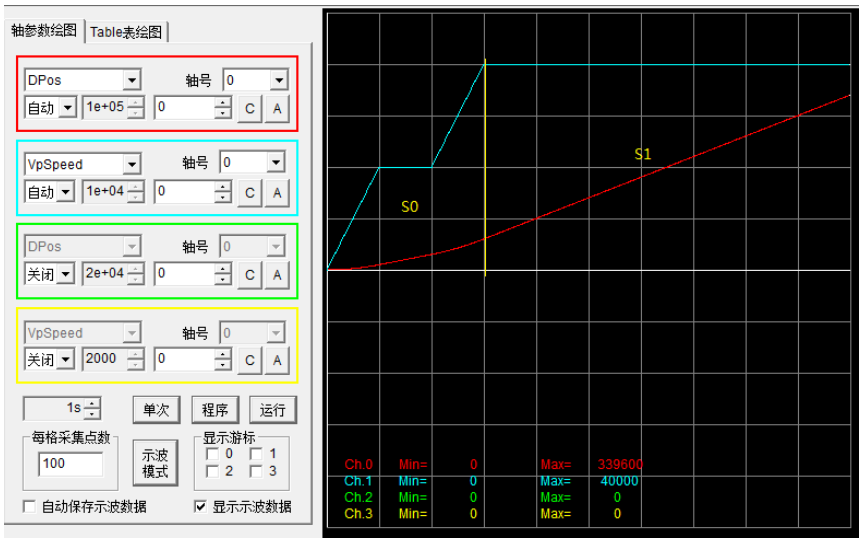


图 161 Merge 为 2 示意图

- 不支持的模块版本

  MC1008-A01

4.4.7.2 HMC_ClcJerk（冲击运算）

- 函数原型

  LREAL HMC_ClcJerk (LREAL dCurvature, LREAL dSpeed)

- 功能说明

  根据 Speed 和曲率计算 Jerk。

  $Jerk=dCurvature^2 * dSpeed^3$

- 输入参数

输入参数	类型	参数描述
dCurvature	LREAL	曲率（非负）
dSpeed	LREAL	速度（非负）

- 返回值

返回值	类型	参数描述
其他值	LREAL	返回 Jerk 值
-1	LREAL	参数错误

- 指令类型

  非轴运动缓存指令

- 应用举例

  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。

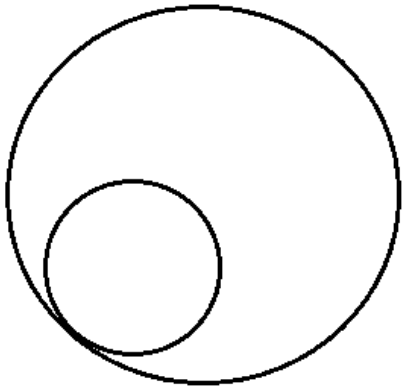


图 162 不同轨迹圆

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：

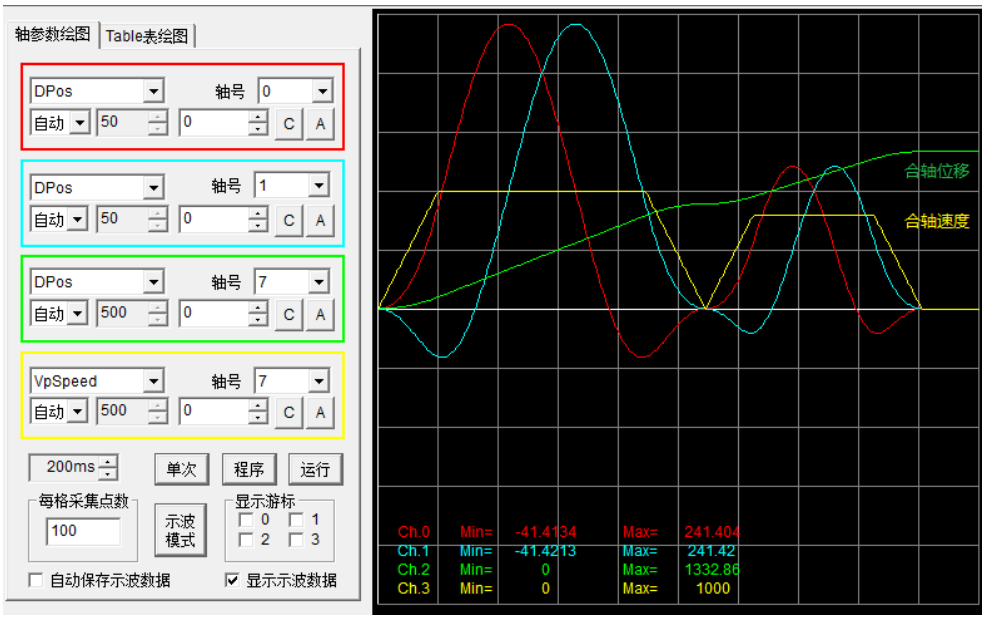


图 163 运动轨迹示意图

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。

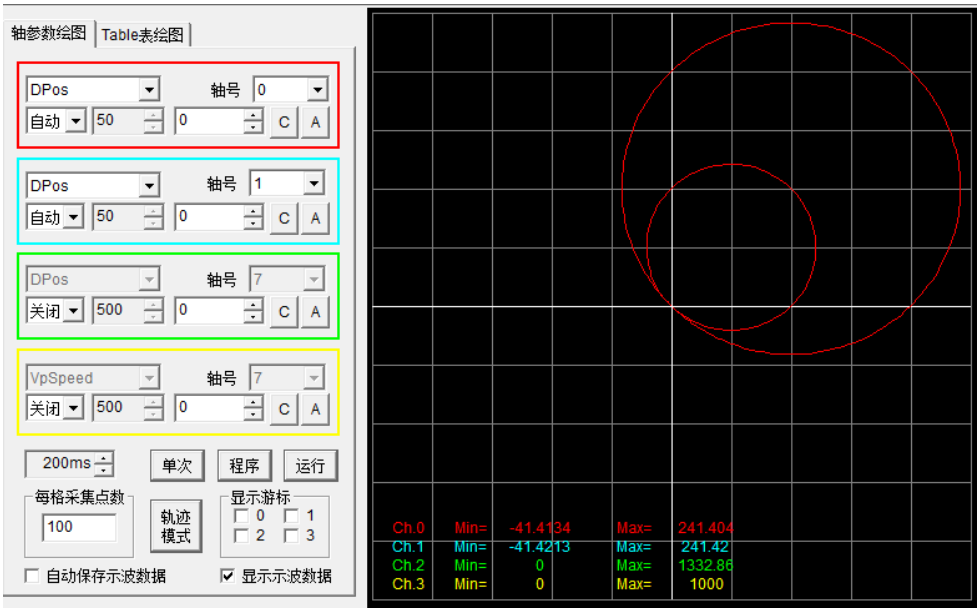


图 164 运动轨迹示意图

- 不支持的模块版本

  MC1008-A01

4.4.7.3 HMC_SetMergeAngle（设置融合角度）

- 函数原型

UDINT HMC_SetMergeAngle (USINT ucAxisID, LREAL dDecelAngle, LREAL dStopAngle)

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dDecelAngle	LREAL	设置二维指令的减速角，对合轴设置有效，不可为负
dStopAngle	LREAL	设置二维指令的停止角，对合轴设置有效，不可为负

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 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。

- 应用举例

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);
```

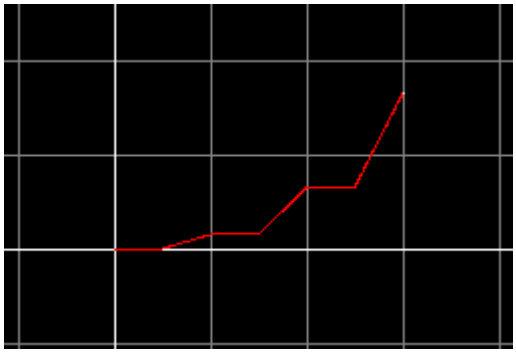


图 165 Merge 在二维指令下的情况

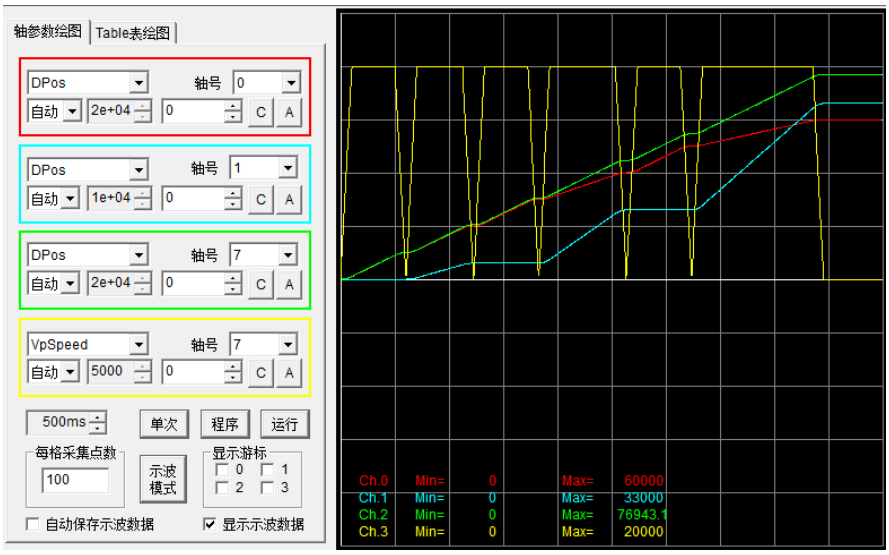


图 166 Merge 为 0 时的情况

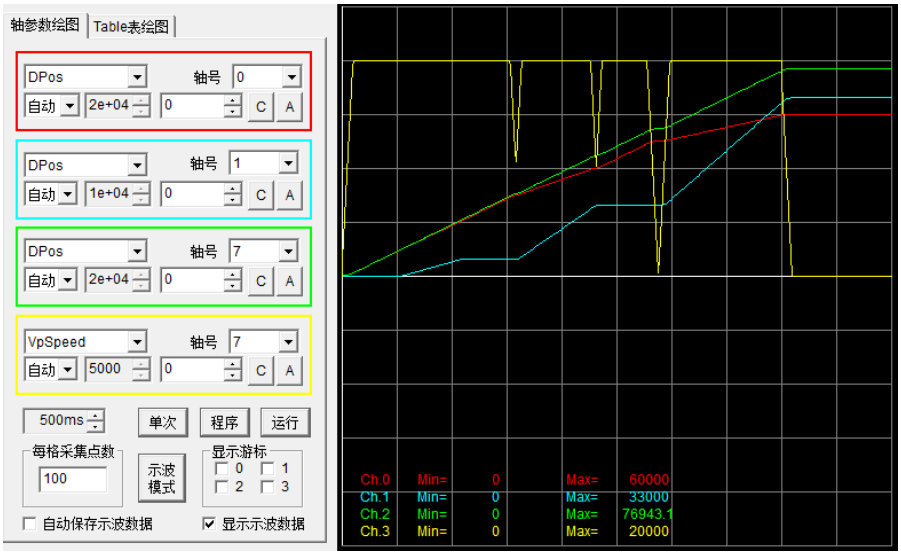


图 167 Merge 为 1 时的情况

- 不支持的模块版本

  MC1008-A01

4.4.7.4 HMC_GetMergeStopAngle（获取融合停止角）

- 函数原型

  LREAL HMC_ GetMergeStopAngle (USINT ucAxisID)

- 功能说明

  获取插补运动停止角。

- 输入参数

输入参数	类型	参数描述

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
运动停止角	LREAL	弧度

- 指令类型

  非轴运动缓存指令

- 不支持的模块版本

  MC1008-A01

4.4.7.5 HMC_GetMergeDecelAngle（获取融合减速角）

- 函数原型

  LREAL HMC_ GetMergeDecelAngle (USINT ucAxisID)

- 功能说明

  获取插补运动减速角。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
运动减速角	LREAL	弧度

- 指令类型

  非轴运动缓存指令

- 不支持的模块版本

  MC1008-A01

4.4.7.6 HMC_SetMergeSpeedTolerRatio（设定速度波动抑制比）

- 函数原型

  UDINT HMC_ SetMergeSpeedTolerRatio (USINT ucAxisID, LREAL dTolerRatio)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dTolerRatio	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

- 返回值

返回值	类型	参数描述
0	UDINT	成功
错误码	UDINT	参数错误

- 指令类型

  非轴运动缓存指令。

4.4.7.7 HMC_ GetMergeSpeedTolerRatio（读取速度波动抑制比）

- 函数原型

  LREAL HMC_ GetMergeSpeedTolerRatio (USINT ucAxisID)

- 功能说明

  读取当前的速度波动抑制比。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
速度波动抑制比	LREAL	当前速度波动抑制比

- 指令类型

  非轴运动缓存指令。

4.4.8 曲线限速

4.4.8.1 HMC_SetCurveJerkLimit（设定曲线冲击上限）

- 函数原型

  UDINT HMC_SetCurveJerkLimit (USINT ucAxisID, LREAL dLimitVal)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dLimitVal	LREAL	设定的上限值，非负

- 返回值

返回值	类型	参数描述
0	UDINT	成功
错误码	UDINT	参数错误

- 指令类型

  非轴运动缓存指令

4.4.8.2 HMC_GetCurveJerkLimit（读取曲线冲击上限）

- 函数原型

  LREAL HMC_GetCurveJerkLimit (USINT ucAxisID)

- 功能说明

  读取设定的曲线插补物理冲击的上限值。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
冲击上限值	LREAL	曲线插补物理冲击上限值 指令参数错误时返回 0

- 指令类型

  非轴运动缓存指令

4.4.9 切向追踪

4.4.9.1 HMC_MoveTang（切向追踪）

- 函数原型

  UDINT HMC_MoveTang (USINT ucAxisC, USINT ucAxisX, USINT ucAxisY, LREAL dOffset)

- 功能说明

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$[-\pi, \pi)$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：

1）加大 C 轴加减速/Speed；

2）减小 XY 轴插补速度以降低切向角变化速率；

3）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：

1）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；

2）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。

3）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：

1）优化 XY 插补路径。比如使用样条插补轨迹。

2）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。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号
ucAxisX	USINT	X 方向运动轴轴号
ucAxisY	USINT	Y 方向运动轴轴号
dOffset	LREAL	刀具旋转轴偏移量（用户单位）

- 返回值

返回值	类型	参数描述
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

轴运动缓存指令。

- 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。

- 应用举例

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：

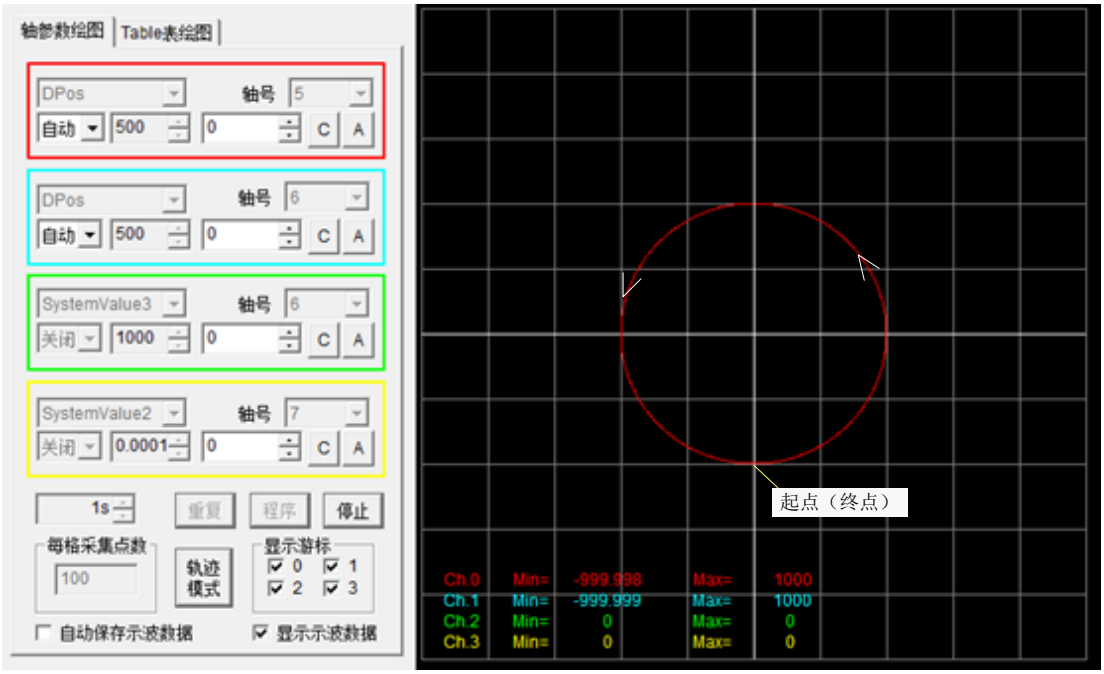


图 168 XY 轴插补轨迹

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）：

$$\text{切向角变化的最大速度} = \frac{\frac{\text{速度}}{\text{半径}}}{2*\pi} * \text{行程长度} = \frac{\frac{1000}{1000}}{2*\pi} * 20000 = 3{,}183.09886$$

$$\text{切向角变化的加速度} = \frac{\frac{\text{加速度}}{\text{半径}}}{2*\pi} * \text{行程长度} = \frac{\frac{2000}{1000}}{2*\pi} * 20000 = 6{,}366.19772$$

$$\text{切向角变化的减速度} = \frac{\frac{\text{减速度}}{\text{半径}}}{2*\pi} * \text{行程长度} = \frac{\frac{2000}{1000}}{2*\pi} * 20000 = 6{,}366.19772$$

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

由图 169 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。

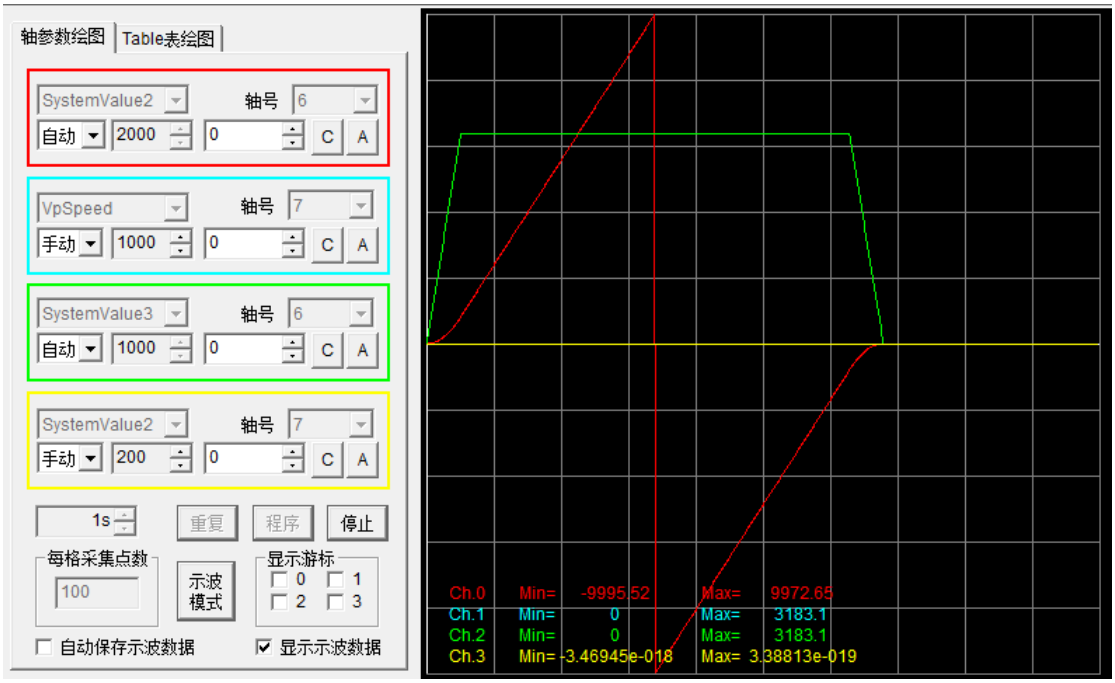


图 169 圆弧插补切向追踪实时数据（C 轴加减速、最大速度 Speed 足够）

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。

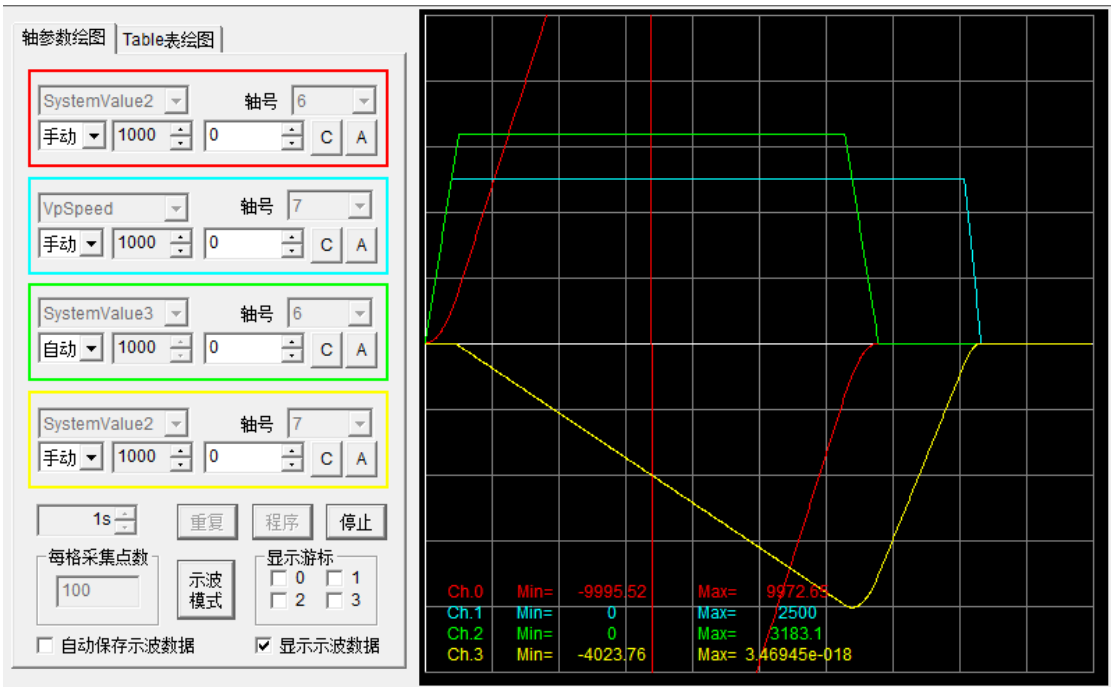


图 170　圆弧插补切向追踪实时数据（C 轴最大速度 Speed 过小）

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。

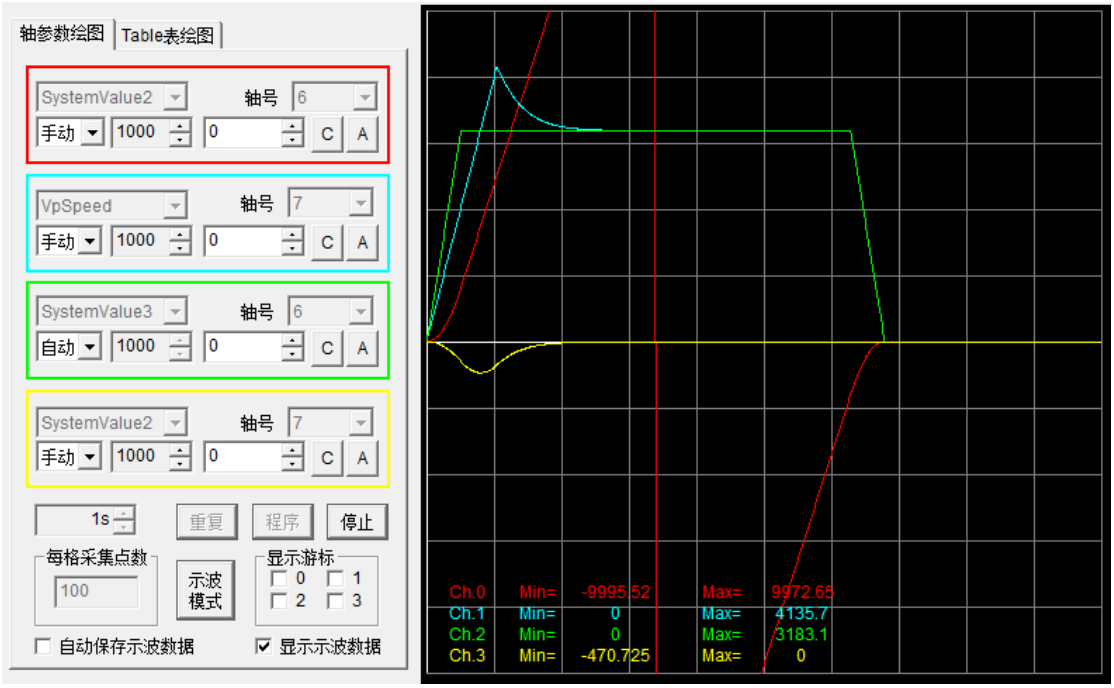


图 171 圆弧插补切向追踪实时数据（C 轴加速能力不足）

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。

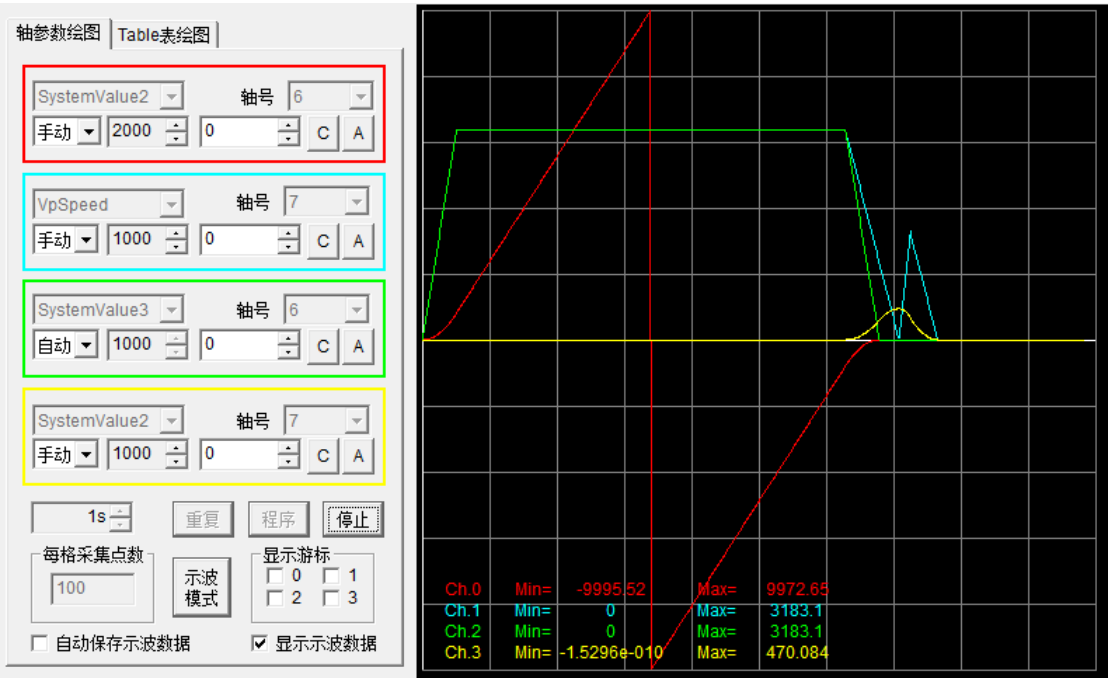


图 172　圆弧插补切向追踪实时数据（C 轴减速能力不足）

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 173 所示：

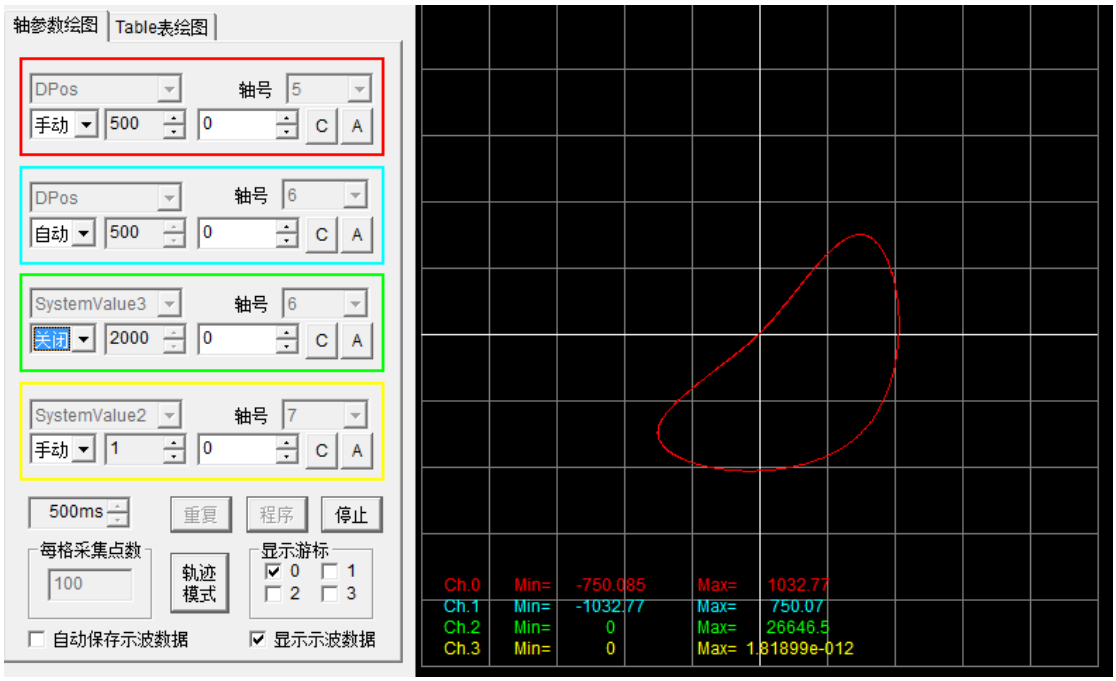


图 173　XY 轴 2D 样条插补轨迹

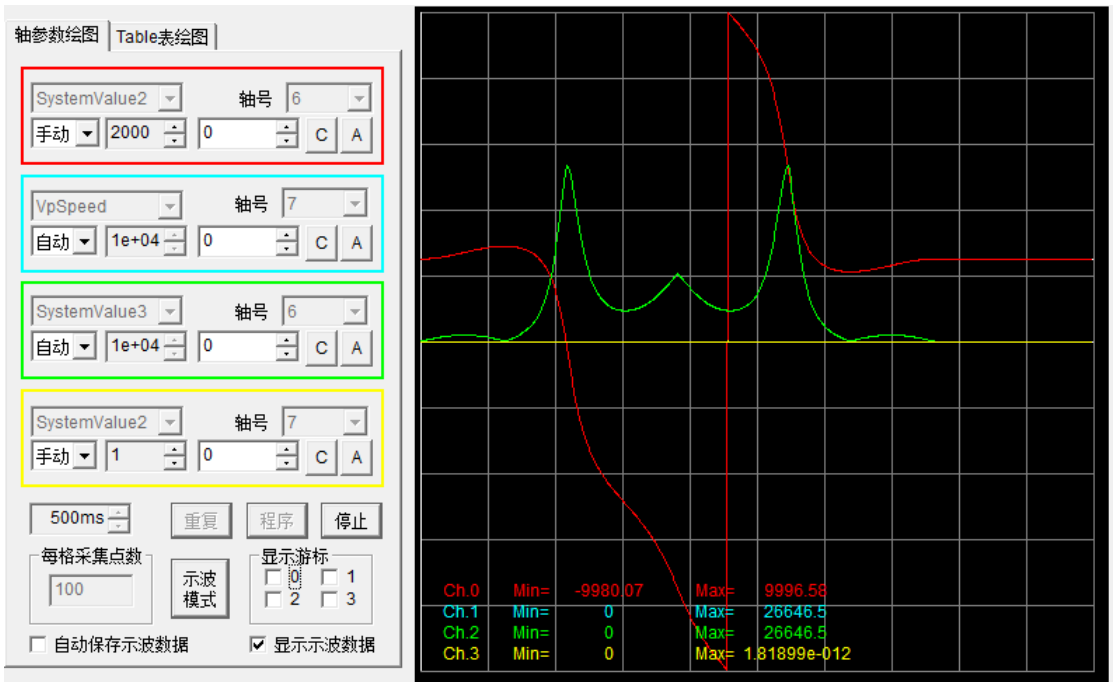


图 174　样条插补切向追踪实时数据

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：

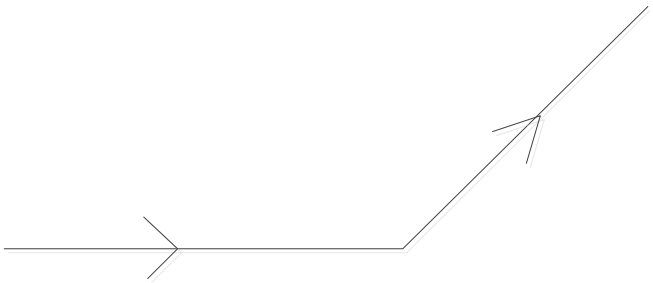


图 175 XY 轴运行轨迹

C 轴参数：accel = decel = 10000，speed = 10000。运行轨迹如下：

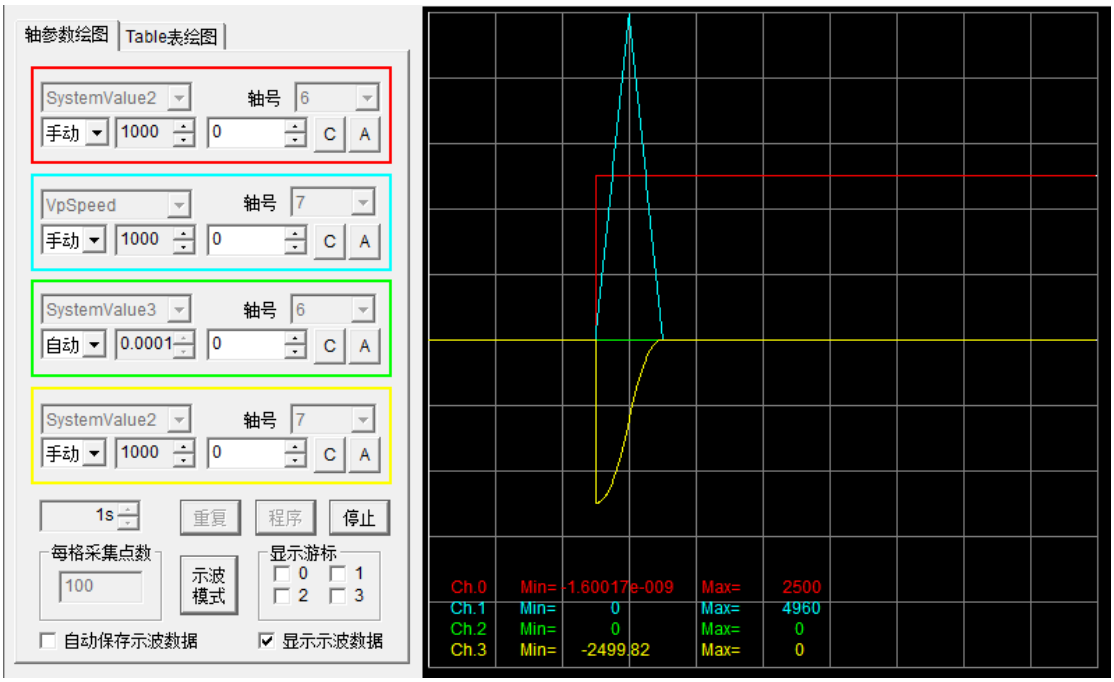


图 176 切向追踪实时数据（折线，Merge 开启）

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：

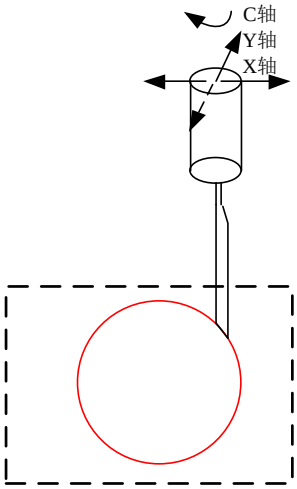


图 177 切割示意图

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：

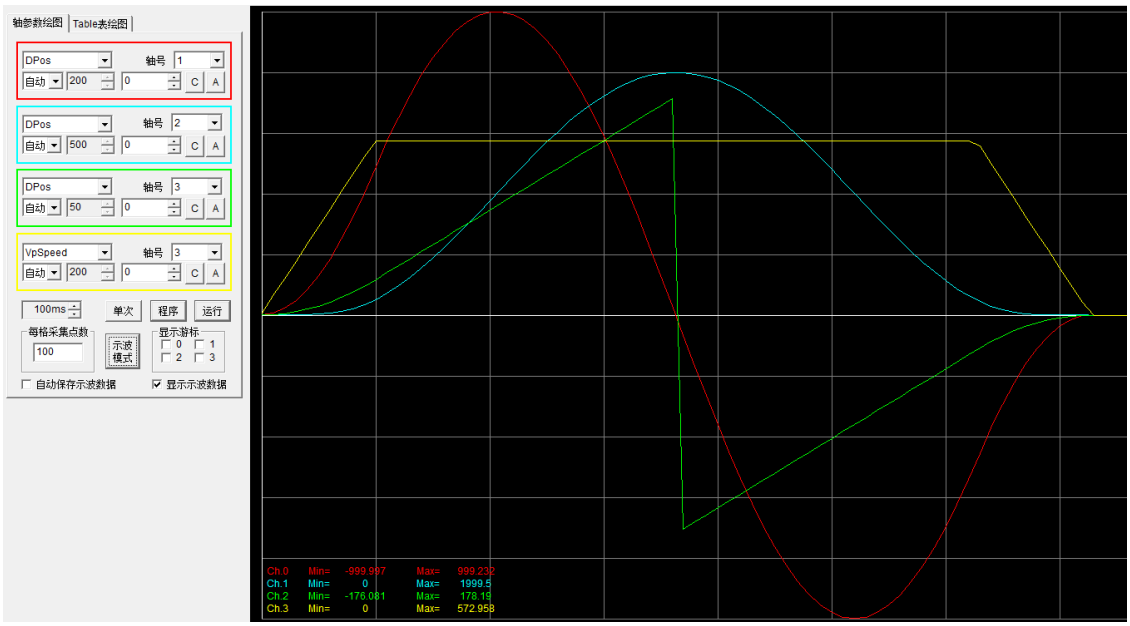


图 178 运动示意图

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。

4.4.9.2 HMC_MoveTangConfigCornerMode（切向追踪拐角提刀设定）

■ 函数原型

UDINT HMC_MoveTangConfigCornerMode (USINT ucAxisC,

USINT ucCornerMode,

LREAL dCornerAngelLimit,

USINT ucAxisZ,

LREAL dSafePosZ,

USINT ucSafePosZMode)

■ 功能说明

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。

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:

$$\text{merge 停止角} \leq \text{拐角提刀角度阈值}$$

（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。

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）：

MC 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：

$$\text{merge 停止角} \leq \text{拐角提刀角度阈值}$$

（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）。

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。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号
ucCornerMode	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	LREAL	拐角抬刀角度阈值（单位：弧度）
ucAxisZ	USINT	拐角自动抬刀 Z 轴轴号 注：ucCornerMode 为 2 时生效。
dSafePosZ	LREAL	拐角自动抬刀 安全提刀高度。 注：ucCornerMode 为 2 时生效。
ucSafePosZMode	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。

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 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。

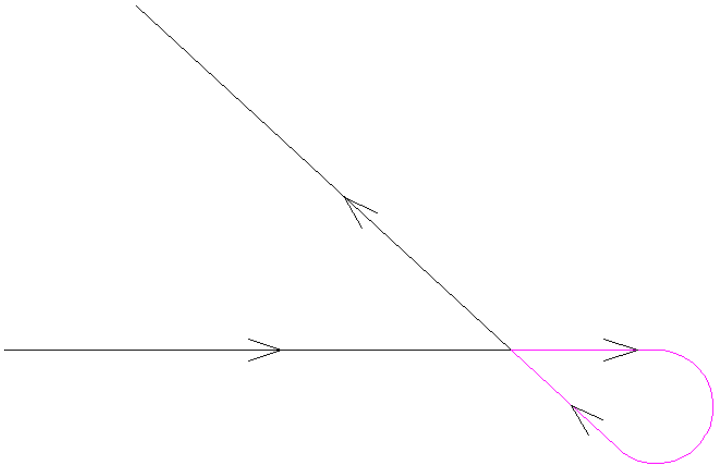


图 179 圆弧工艺路径示意图

（2）带抬刀的工艺路径

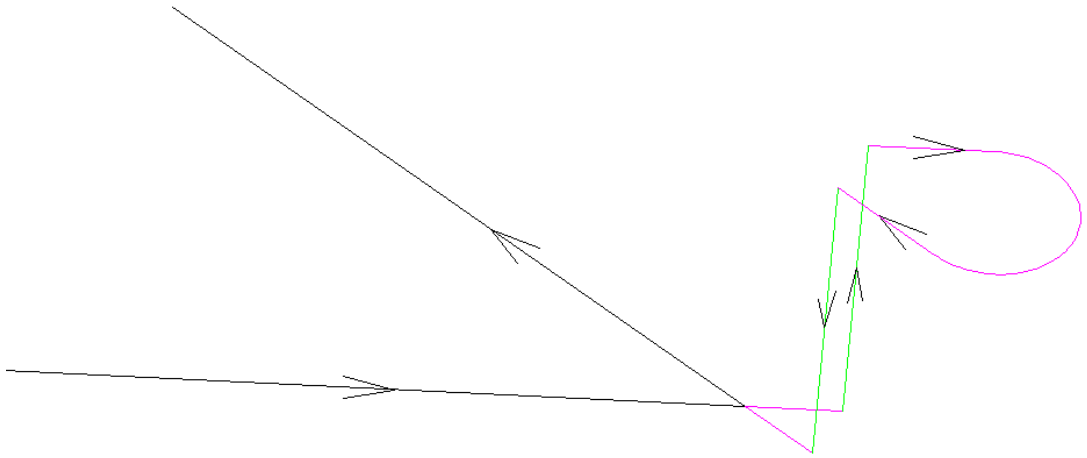


图 180 带抬刀的工艺路径示意图

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。

■ 应用举例

（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$\frac{\pi}{4}$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$\frac{\pi}{4}$处自动抬刀，设定拐角抬刀的角度阈值为$\frac{\pi}{4} - 10^{-6}$，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 插补轨迹如图 181 所示，各阶段运动过程曲线如图 182 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);
```

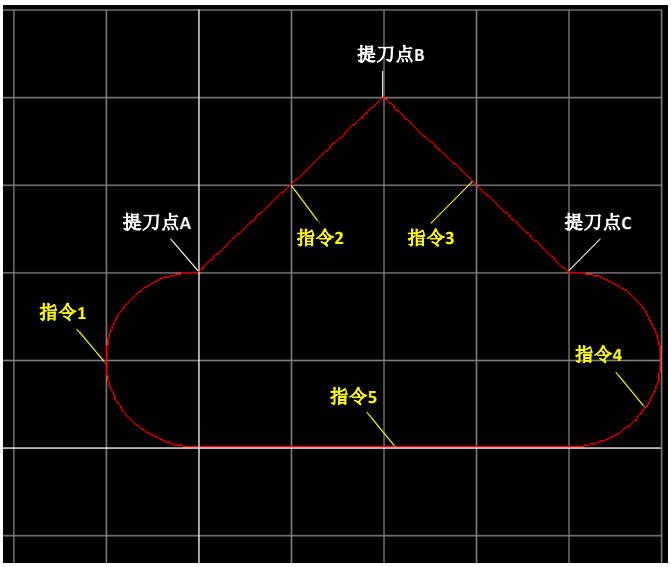


图 181　切向追踪主轴 XY 插补轨迹

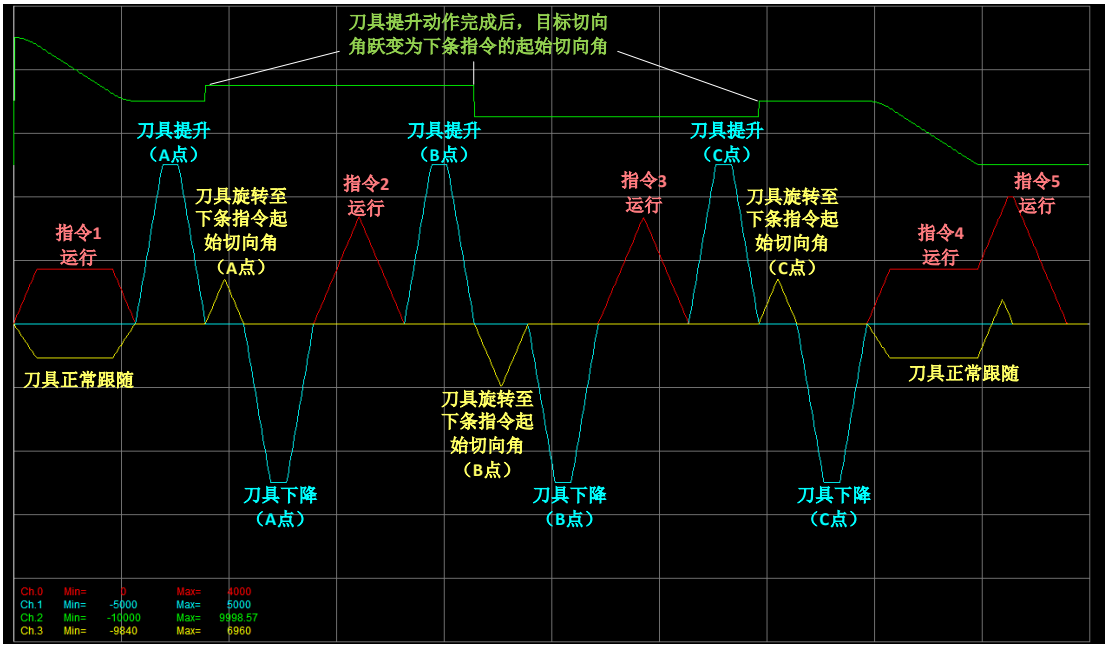


图 182 切向追踪自动提刀模式运动过程曲线（主轴 XY 执行 圆弧+Move2d）

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，则执行效果如图 183 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);
```

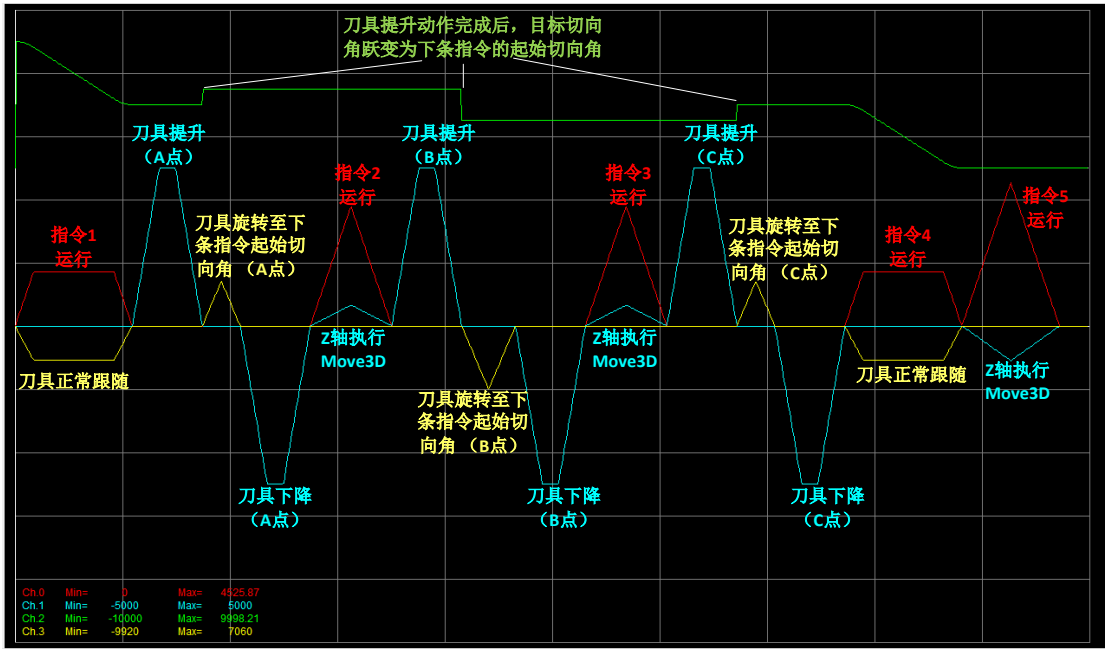


图 183 切向追踪自动提刀模式运动过程曲线（XYZ 执行 圆弧+Move3d）

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 插补轨迹如图 184 所示，各阶段运动过程曲线如图 185 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
```

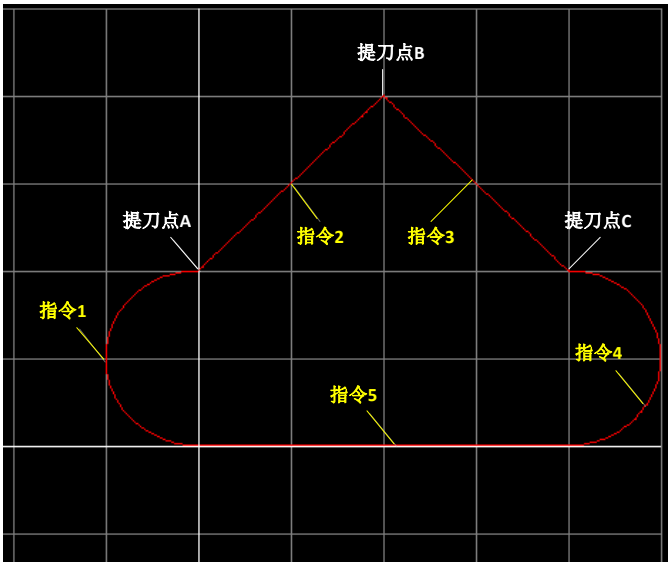


图 184 切向追踪主轴 XY 插补轨迹

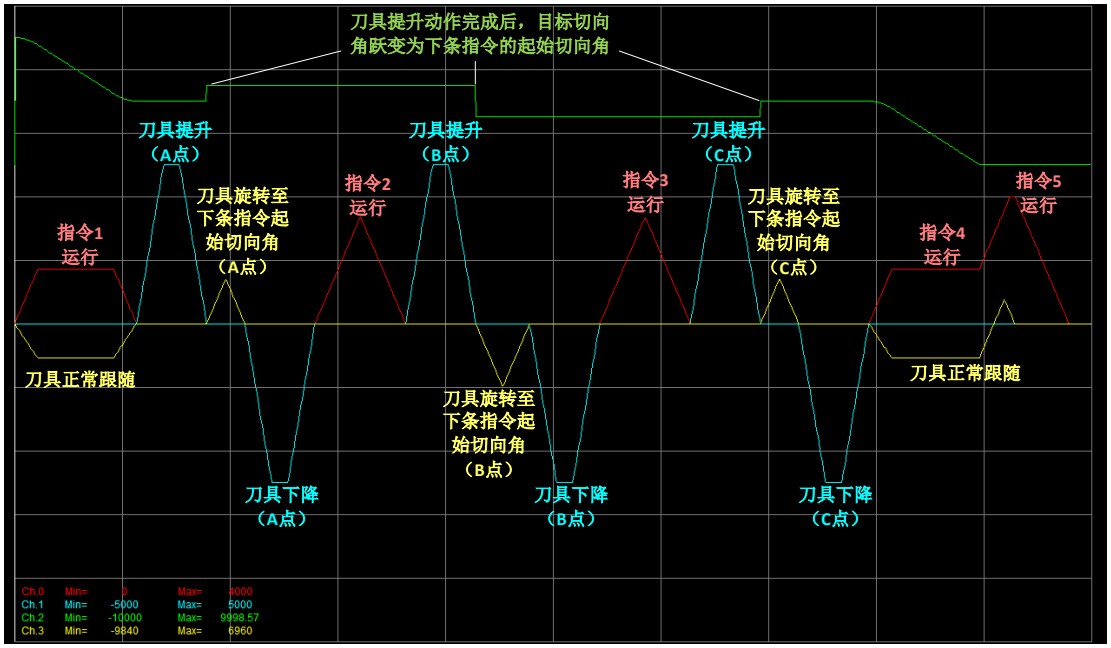


图 185 切向追踪手动提刀模式运动过程曲线（主轴 XY 执行 圆弧+Move2d）

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

4.4.9.3 HMC_GetMoveTangCornerState（获取切向追踪拐角抬刀状态）

- 函数原型

USINT HMC_GetMoveTangCornerState(USINT ucAxisC)

- 功能说明

  获取切向追踪拐角抬刀状态。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号

- 返回值

返回值	类型	参数描述
拐角抬刀状态	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

- 指令类型

  非轴运动缓存指令

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

- 补充说明

  当前指令不为切向追踪指令时，返回错误。

4.4.9.4 HMC_MoveTangCornerFollowNextCmd（C 轴定位至下一条指令的起始切向角）

- 函数原型

  UDINT HMC_MoveTangCornerFollowNextCmd (USINT ucAxisC)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

非轴运动缓存指令。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

- 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。

4.4.9.5 HMC_MoveTangCornerContinue（切向追踪继续运行）

- 函数原型

UDINT HMC_MoveTangCornerContinue (USINT ucAxisC)

- 功能说明

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。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

非轴运动缓存指令。

- 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

- 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。

4.4.9.6 HMC_GetMoveTangDestDirection（获取切向追踪目标方向位置）

- 函数原型

LREAL HMC_ GetMoveTangDestDirection (USINT ucAxisC)

- 功能说明

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$[-\pi, \pi)$的切向角度。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号

- 返回值

返回值	类型	参数描述
目标切向角	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。

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.4.9.7 HMC_GetMoveTangDestVelocity（获取切向追踪目标速度）

- 函数原型

  LREAL HMC_GetMoveTangDestVelocity (USINT ucAxisC)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号

- 返回值

返回值	类型	参数描述
目标速度	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

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.4.9.8 HMC_ GetMoveTangDeviation（获取切向追踪方向偏差）

- 函数原型

  LREAL HMC_ GetMoveTangDeviation (USINT ucAxisC)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	刀具旋转轴轴号

- 返回值

返回值	类型	参数描述
方向偏差	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。

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.4.9.9 HMC_GetMoveTangMasterAxisX（获取切向追踪主轴 X）

- 函数原型

  INT HMC_GetMoveTangMasterAxisX(USINT ucAxisC)

- 功能说明

  获取当前切向追踪指令的 X 轴。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
轴号/错误	INT	X 轴轴号/错误返回-1

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.4.9.10 HMC_GetMoveTangMasterAxisY（获取切向追踪主轴 Y）

- 函数原型

  INT HMC_GetMoveTangMasterAxisY(USINT ucAxisC)

- 功能说明

  获取当前切向追踪指令的 Y 轴。

- 输入参数

输入参数	类型	参数描述
ucAxisC	USINT	从轴轴号

- 返回值

返回值	类型	参数描述
轴号/错误	INT	Y 轴轴号/错误返回-1

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.4.10 PWM 激光控制

近年来，激光切割应用已被广泛使用于各种加工，例如金属、非金属或复合材料等。PWM 类功能块可根据运动轨迹合速度实时调整激光输出的频率与占空比，达到精确控制激光密度的目的，从而实现高质量的切割效果。

提供 5 种控制模式：

（1）手动控制

该模式下，用户设定的手动占空比、手动频率直接作为最终输出值。

（2）固定频率变占空比

该模式下，输出的频率固定，由用户设定。根据关联轴的合速度实时调整输出占空比，可实现均匀切割。

（3）固定脉宽变频率

该模式下，输出脉宽固定由用户设定，根据关联轴的合速度实时调整输出频率，可实现均匀切割。

（4）固定占空比变频率

该模式下，输出占空比固定由用户设定，根据关联轴的合速度实时调整输出频率，可实现均匀切割。

（5）随动控制占空比和频率

该模式下，根据关联轴的合速度实时调整输出占空比、频率。可实现均匀切割。

使用 PWM 功能时，请按以下步骤：

（1）先使用 HMC_PWMConfig 配置基本参数。

（2）根据控制模式调用对应的在线控制功能块，使得 PWM 进入输出状态。此时可在线调整 PWM 输出，也可通过 Enable 参数暂停 PWM 输出。

（3）需要重新设置基本参数或关闭 PWM 输出时，使用 HMC_PWMClose。

图 186 状态转移图

可以看出中，通过 HMC_PWMConfig 使得状态机从初始状态进入蓝色的过渡状态，一旦进入蓝色状态，其控制模式确定，但此时 PWM 通道还不进行输出；再根据当前控制模式选择对应的控制函数使得状态机从过渡状态进入黄色输出状态，处于输出状态时。

（4）还可通过 PWM 的全局变量在线实时调整 PWM 输出，或观察 PWM 状态及实时输出等参数，如图 187 所示。

图 187 PWM 参数查看与调整

4.4.10.1 HMC_PWMConfig（PWM 配置）

■ 函数原型

UDINT HMC_PWMConfig (BYTE ucPWMChannel, BYTE ucControlMode, BYTE ucOutputMode, BYTE ucOutputID, POINTER TO BYTE pArrLinkAxis, BYTE ucLinkAxisNum, BYTE ucAxisSpeedSrc)

■ 功能说明

对 PWM 功能进行离线配置。仅当状态机为初始状态时（MachineState=0），才可成功调用此功能块进行相关参数配置，其他状态下都不支持。所以一旦 PWM 成功调用 HMC_PWMConfig 退出初始状态后，该函数涉及的配置项则不允许修改，若需修改，必须先通过 HMC_PWMClose 关闭 PWM 通道后重新配置。

该功能块主要配置控制模式、输出方式和通道、随动控制时速度关联轴等信息。

■ 输入参数

输入参数	类型	参数描述
ucPWMChnl	BYTE	PWM 通道号（0~1）
ucControlMode	BYTE	控制模式 0：手动 1：固定频率变占空比 2：固定脉宽变频率

输入参数	类型	参数描述
		3：固定占空比变频率 4：速度随动控制占空比和频率
ucOutputType	BYTE	输出方式 0：Z 相输出 1：DO 输出
ucOutputID	BYTE	DO 通道号或 Z 相输出轴号
pAryLinkAxis	POINTER TO BYTE	速度关联的轴
ucLinkAxisNum	BYTE	速度关联轴个数
ucAxisSpeedSrc	BYTE	速度来源 0：运动指令计算速度 VpSpeed 1：实际反馈速度 MpSpeed

- 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1499：速度关联轴指针所指地址有超出 M、N 区 1553：当前选择的输出轴的 Z 相已被另一路 PWM 占用 1555：当前控制模式不为初始状态 0 1564：当前选择 DO 输出通道已被其它功能占用（如 Switch、StepperDO 等）

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A01

4.4.10.2 HMC_PWMManual（PWM 手动控制）

- 函数原型

  UDINT HMC_PWMManual (BOOL bEnable, BYTE ucPWMChnl, LREAL dManualFreq, LREAL dManualDutyCycle)

- 功能说明

  设置 PWM 手动控制模式时相关控制参数。当 bEnable 为 0 时，PWM 输出停止；bEnable 为 1 时，直接输出用户设定的手动输出频率、手动输出占空比。

- 输入参数

输入参数	类型	参数描述
bEnable	BOOL	0-PWM 输出停止，1-使能 PWM 输出
ucPWMChnl	BYTE	PWM 通道号（0~1）
dManualFreq	LREAL	手动输出频率（单位 Hz）

输入参数	类型	参数描述
dManualDutyCycle	LREAL	手动输出占空比（取值范围 0.0~1.0）

- 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1555：当前控制模式错误

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A001

4.4.10.3 HMC_PWMFixedFreq（PWM 固定频率变占空比控制）

- 函数原型

  UDINT HMC_PWMFixedFreq (BOOL bEnable, BYTE ucPWMChnl, LREAL dFixedFreq, POINTER TO LREAL pDutyCycleTable, UDINT uiDutyCycleTableNum)

- 功能说明

  设置 PWM 固定频率随动调整占空比控制模式的相关控制参数。

  当 bEnable 为 0 时，PWM 输出停止；bEnable 为 1 时，输出频率为用户设定的 dFixedFreq，根据速度关联轴的合速度查表计算得到输出占空比。

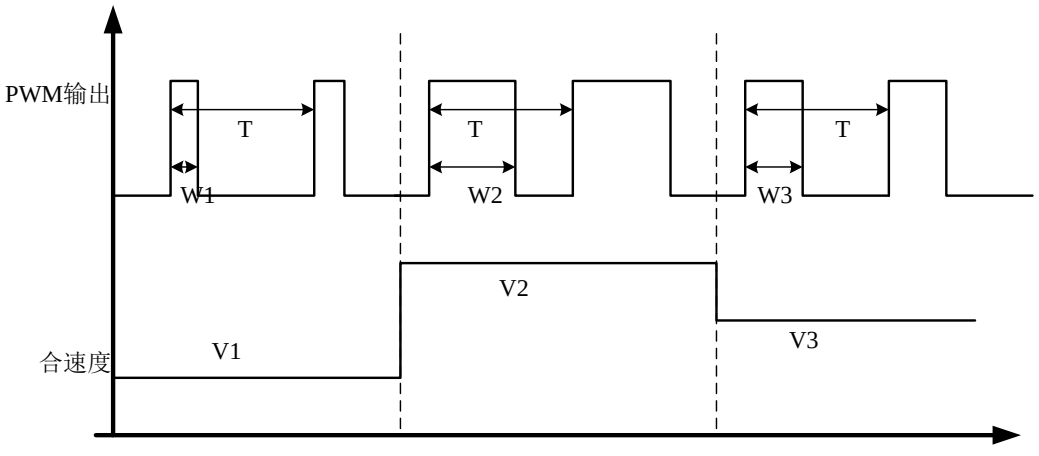


图 188 PWM 输出与速度示意图

- 输入参数

输入参数	类型	参数描述
bEnable	BOOL	0：PWM 输出停止 1：使能 PWM 输出

输入参数	类型	参数描述
ucPWMChnl	BYTE	PWM 通道号（0~1）
dFixedFreq	LREAL	设定频率值（单位 Hz）
pDutyCycleTable	POINTER TO LREAL	占空比与速度关系。在二维数组 pDutyCycleTable[2][uiNum]中，pMapTable[0]为占空比数组，pMapTable[1]为速度数组，两者一一对应，描述出占空比相对于速度的变化曲线
uiDutyCycleTableNum	UDINT	占空比与速度数组的长度，即点个数

- 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1499：占空比与速度映射表所占地址有超出 M、N 区 1555：当前控制模式错误

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A001

- 应用举例

  使用激光器切割金属材料：

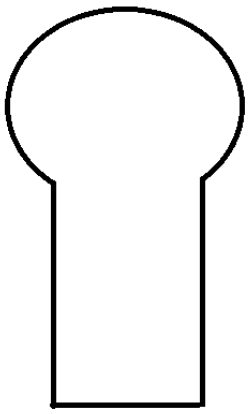


图 189 切割形状示意图

在切割过程中，为了保证切割轨迹精准，在圆弧段和短直线段调整速度小一些，在两个长的直线段速度大一些，为保证切割效率，开启速度前瞻。因此整个切割过程中，合运动轨迹的速度会实时变化，指令连接处会出现加速、减速阶段。

由于整个切割过程中，合运动轨迹速度会变化，为了保证激光切割的均匀，激光输出功率应跟随合运动速度实时调整，以达到均匀切割得效果。如速度减速阶段，激光输出功率也应跟随减小，避免出现过融现象；当速度加速阶段，激光输出功率也应跟随调大，避免出现欠融现象。

程序如下：

```
AxisAry[0] := 0;      (*合运动轴*)

(*占空比与速度随动调整数组*)
MapTable[0,0] := 0;
MapTable[0,1] := 0.2;
MapTable[0,2]:=0.6;
MapTable[0,3]:=0.8;
MapTable[0,4] := 1.0;

MapTable[1,0]:=0;
MapTable[1,1] := 1000;
MapTable[1,2]:=2000;
MapTable[1,3]:=3000;
MapTable[1,4] := 4000;

axis0.Accel:=3000;
axis0.Decel:=3000;

ConfigTemp := HMC_PWMConfig(0,1,1,0,ADR(AxisAry),1,0);  (*先配置 PWM 基本参数，控制模式配置为固定频率调整占空比*)
ControlTemp := HMC_PWMFixedFreq (1 ,0,dFixFreq,ADR(MapTable),5);   (*再通过实时控制函数，开启 PWM 输出，设置固定频率、占空比与速度关联关系*)

HMC_SetMerge (0, 1);
HMC_MoveCircleEx (0,1,2,3,2000,0,1000,1000,2000);    (*切割圆弧段*)
HMC_Move2DEx (0,1,2,0,-4000,3000);          (*切割长线段 1*)
HMC_Move2DEx (0,1,2,-2000,0,2000);          (*切割短线段*)
HMC_Move2DEx (0,1,2,0,4000,3000);           (*切割长线段 2*)
```

使用 AT 软件示波器记录合运动速度与 PWM 实时输出的占空比、频率关系：

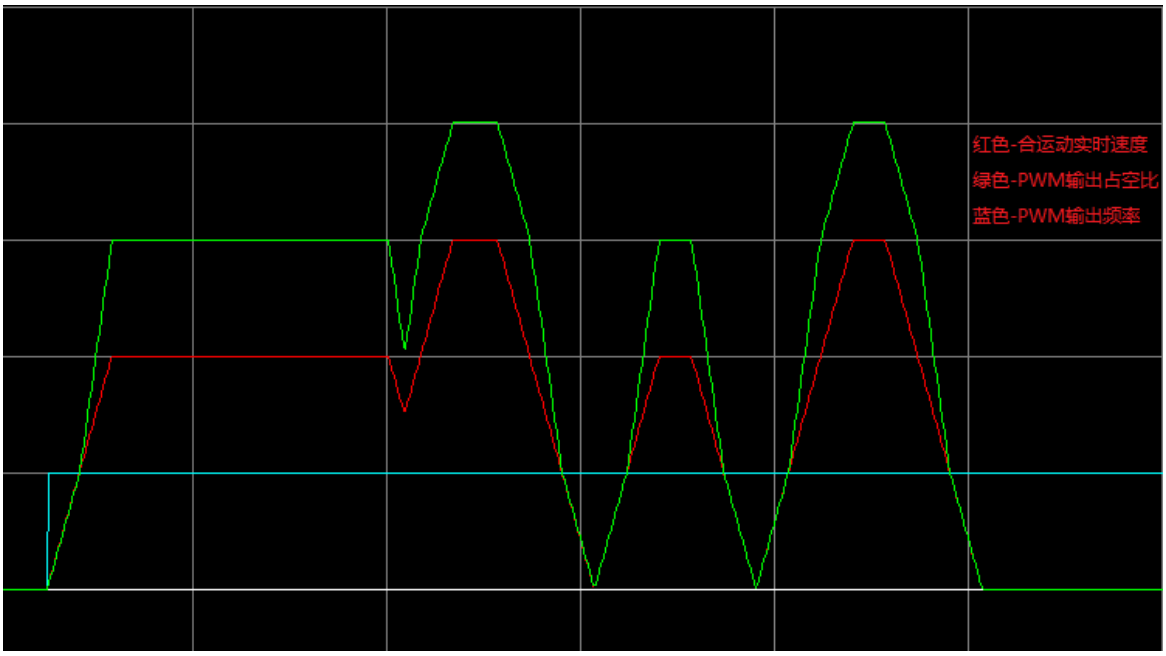


图 190 运动轨迹示意图

可以看出，合运动轨迹（红色）在整个切割过程中的指令连接处出现加减速，在速度发生变化时，为了保证切割均匀，避免过融和欠融，需通过 PWM 调整输出占空比以控制激光器输出功率。在速度加速阶段，PWM 输出占空比也跟随增加；在速度减速阶段，PWM 输出占空比也跟随减小。

4.4.10.4 HMC_PWMFixedPulse（PWM 固定脉宽变频率控制）

■ 函数原型

UDINT HMC_PWMFixedPulse (BOOL bEnable, BYTE ucPWMChnl, UDINT uiFixedPulseWidth, POINTER TO LREAL pFreqTable, UDINT uiFreqTableNum)

■ 功能说明

设置 PWM 固定脉宽随动调整频率控制模式的相关控制参数。

当 bEnable 为 0 时，PWM 输出停止；bEnable 为 1 时，输出脉宽为用户设定的 uiFixedPulseWidth，根据速度关联轴的合速度查表计算得到输出频率。

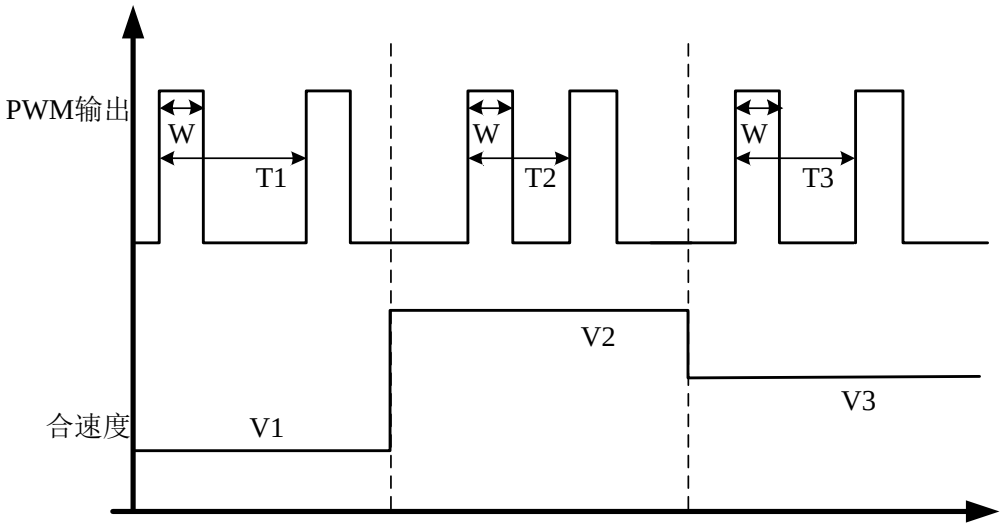


图 191 PWM 输出与速度示意图

- 输入参数

输入参数	类型	参数描述
bEnable	BOOL	0：PWM 输出停止 1：使能 PWM 输出
ucPWMChnl	BYTE	PWM 通道号（0~1）
uiFixedPulseWidth	UDINT	设定脉宽（单位：ns）
pFreqTable	POINTER TO LREAL	频率与速度关系。在二维数组 pFreqTable[2][uiNum]中，pFreqTable[0]为频率比数组，pFreqTable[1]为速度数组，两者一一对应，描述出频率相对于速度的变化曲线
uiFreqTableNum	UDINT	频率与速度数组的长度，即点个数

- 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1499：频率与速度映射表所占地址有超出 M、N 区 1555：当前控制模式错误

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A001

4.4.10.5 HMC_PWMFixedDutyCycle（PWM 固定占空比变频率控制）

■ 函数原型

UDINT HMC_PWMFixedDutyCycle (BOOL bEnable, BYTE ucPWMChnl, LREAL dFixedDutyCycle, POINTER TO LREAL pFreqTable, UDINT uiFreqTableNum)

■ 功能说明

设置 PWM 固定占空比随动调整频率控制模式的相关控制参数。

当 bEnable 为 0 时，PWM 输出停止；bEnable 为 1 时，输出占空比为用户设定的 dFixedDutyCycle，根据速度关联轴的合速度查表计算得到输出频率。

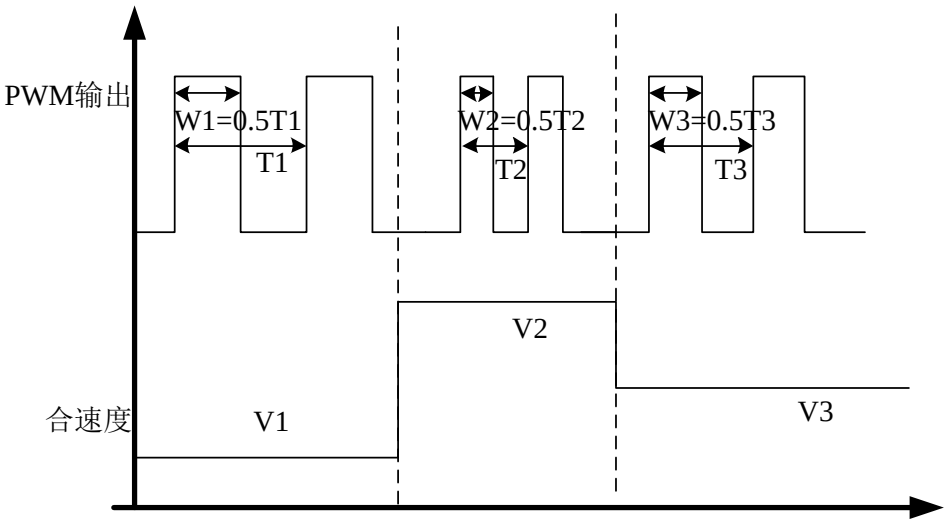


图 192 PWM 输出与速度示意图

■ 输入参数

输入参数	类型	参数描述
bEnable	BOOL	0：PWM 输出停止 1：使能 PWM 输出
ucPWMChnl	BYTE	PWM 通道号（0~1）
dFixedDutyCycle	LREAL	设定占空比（取值范围 0.0~1.0）
pFreqTable	POINTER TO LREAL	频率与速度关系。在二维数组 pFreqTable[2][uiNum]中，pFreqTable[0]为频率数组，pFreqTable[1]为速度数组，两者一一对应，描述出频率相对于速度的变化曲线
uiFreqTableNum	UDINT	频率与速度数组的长度，即点个数

■ 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1499：频率与速度映射表所占地址有超出 M、N 区

返回值	类型	参数描述
		1555：当前控制模式错误

■ 指令类型

非轴运动缓存指令。

■ 不支持的模块版本

MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A001

4.4.10.6 HMC_PWMChangeFreqAndDutyCycle（PWM 根据速度调整频率和占空比控制模式）

■ 函数原型

UDINT HMC_PWMChangeFreqAndDutyCycle (BOOL bEnable, BYTE ucPWMChnl, POINTER TO LREAL pFreqTable, UDINT uiFreqTableNum, POINTER TO LREAL pDutyCycleTable, UDINT uiDutyCycleTableNum)

■ 功能说明

设置 PWM 速度随动调整频率、占空比控制模式的相关控制参数。

当 bEnable 为 0 时，PWM 输出停止；bEnable 为 1 时，根据速度关联轴的合速度查频率速度映射表计算得到输出频率，根据速度关联轴的合速度查占空比速度映射表计算得到输出占空比。

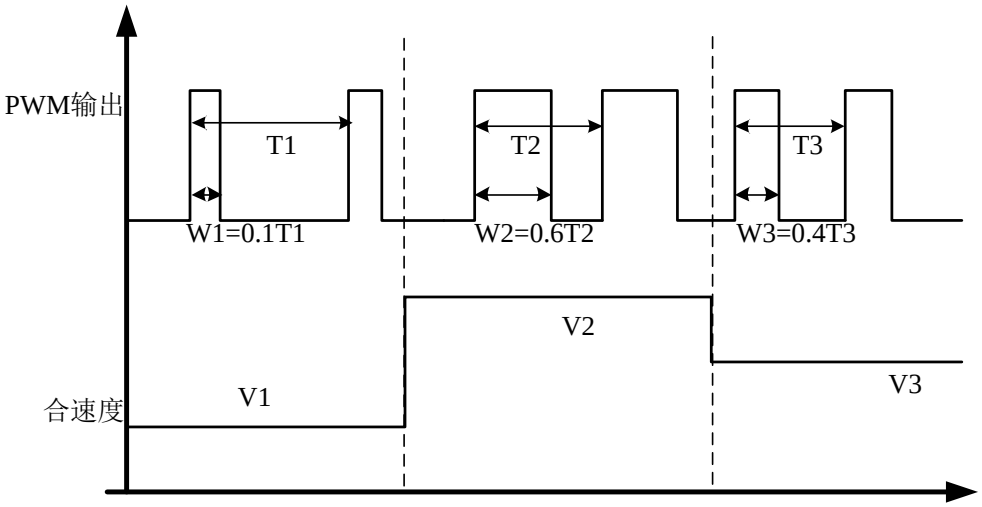


图 193 PWM 输出与速度示意图

■ 输入参数

输入参数	类型	参数描述
bEnable	BOOL	0：PWM 输出停止 1：使能 PWM 输出
ucPWMChnl	BYTE	PWM 通道号（0~1）
pFreqTable	POINTER TO LREAL	频率与速度关系。在二维数组

输入参数	类型	参数描述
		pFreqTable[2][uiNum]中，pFreqTable[0]为频率数组，pFreqTable[1]为速度数组，两者一一对应，描述出频率相对于速度的变化曲线
uiFreqTableNum	UDINT	频率与速度数组的长度，即点个数
pDutyCycleTable	POINTER TO LREAL	占空比与速度关系。在二维数组 pDutyCycleTable[2][uiNum] 中，pMapTable[0]为占空比数组，pMapTable[1]为速度数组，两者一一对应，描述出占空比相对于速度的变化曲线
uiDutyCycleTableNum	UDINT	占空比与速度数组的长度，即点个数

- 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1499：占空比与速度映射表，或频率与速度映射表所占地址有超出 M、N 区 1555：当前控制模式错误

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A001

4.4.10.7 HMC_PWMSetDOCompensation（设置 DO 输出脉宽补偿值）

- 函数原型

  UDINT HMC_PWMSetDOCompensation (BYTE ucPWMChnl, DINT iDOCompensation)

- 功能说明

  当输出方式为 DO 输出时，可通过此功能块对输出计算的结果进行补偿，在原有计算的输出脉宽宽度基础上叠加补偿值。

- 输入参数

输入参数	类型	参数描述
ucPWMChnl	BYTE	PWM 通道号（0~1）
iDOCompensation	DINT	DO 输出补偿值（单位：ns）

- 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1555：当前控制模式错误，不为 1 或 3

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A001

4.4.10.8 HMC_PWMClose（PWM 关闭）

- 函数原型

  UDINT HMC_PWMClose (BYTE ucPWMChnl)

- 功能说明

  关闭指定 PWM 通道。使得状态机复位到初始状态，输出停止。

- 输入参数

输入参数	类型	参数描述
ucPWMChnl	BYTE	PWM 通道号（0~1）

- 输出参数

返回值	类型	参数描述
0	UDINT	成功
其它值	UDINT	0xffffffff：函数参数错误 1555：当前控制模式错误

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A001

4.4.11 多轴同步控制

4.4.11.1 HMC_GantrySynchro（龙门同步控制）

- 函数原型

  UDINT HMC_ GantrySynchro (USINT ucMasterAxis，USINT* pSlaveAxis，USINT ucSlaveAxisNum，LREAL lrPosWaring， LREAL lrPosErr)

- 功能说明

  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。
- 输入参数

输入参数	类型	含义
UCMASTERAXIS	USINT	龙门组主动同步轴号（需小于任一从动轴号）
PSLAVEAXIS	POINTER TO USINT	龙门组从动同步轴号数组
UCSLAVEAXISNUM:	USINT	龙门组从动同步轴个数（≤7）
UCMODE	USINT	龙门同步模式： 0：轴间开环同步 1：轴间闭环同步
LRPOSWARING	LREAL	轴间偏差报警阈值
LRPOSERR	LREAL	轴间偏差停车保护阈值

- 返回值

返回值	类型	含义
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  轴运动缓存指令。

- 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。
- 应用示例

  如图 194 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。

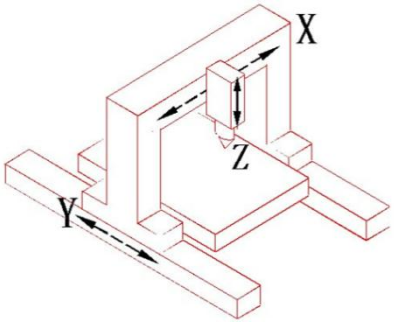


图 194 结构示意图

```
    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 195 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

图 195 参数示意图

4.4.11.2 HMC_GantryInit（龙门同步回参考点位置对齐初始化）

- 函数原型

UDINT HMC_ GantryInit (USINT* pAxis，USINT ucAxisNum，USINT ucMode，USINT ucHomeMode， USINT* pDiChl，USINT ucCaptureChl， LREAL dReferPointPos， LREAL dPosErr)

■ 功能说明

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：

1. 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；

2. 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；

3. 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。

■ 输入参数

输入参数	类型	含义
PAXIS	POINTER TO USINT	龙门组同步轴号数组
UCAXISNUM	USINT	龙门组同步轴个数（≤8）
UCMODE	USINT	龙门同步回参考点模式： 0：Home 方式 1：HOMEEX 方式
UCHOMEMODE	USINT	龙门同步回参考点的信号源模式，可参见 Home（0~5）
PDICHL	POINTER TO USINT	回参考点使用的 DI 信号（不可重复） Mode 为 0 时：DI 可为高速、低速、虚拟 DI Mode 为 1 时：DI 只能为高速 DI
UCCAPTURECHL	USINT	高速回参考点使用的高速捕捉通道号
DREFERPOINTPOS	LREAL	参考点绝对位置值
DPOSERR	LREAL	轴间偏差阈值

■ 返回值

返回值	类型	含义
轴指令 ID	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

■ 指令类型

轴运动缓存指令。

■ 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。

■ 应用示例

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。

```
    pAxisAry[0] := 1;
    pAxisAry[1] := 2;
    ucAxisNum := 2;
    ucMode := 1;
    ucHomeMode := 0;
    pDIChl[0] := 0;
    pDIChl[1] := 1;
    ucCaptureChl := 0;
    dReferPointPos := -1000;
    dPosErr := 10;

    HMC_GantryInit (ADR(pAxisAry) , ucAxisNum, ucMode, ucHomeMode,
ADR(pDIChl) , ucCaptureChl, dReferPointPos, dPosErr);
```

4.4.11.3 HMC_SetGantryAxisPID（设置龙门同步轴补偿 PID 参数）

■ 函数原型

UDINT HMC_ SetGantryAxisPID (USINT ucAxisID， LREAL dKpGain，LREAL dKiGain，LREAL dKdGain)

■ 功能说明

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。

■ 输入参数

输入参数	类型	含义
UCAXISID	USINT	龙门组轴号
DKPGAIN	LREAL	动态补偿 PID 比例系数（默认值为 1）
DKIGAIN	LREAL	动态补偿 PID 积分系数（默认值为 0）
DKDGAIN	LREAL	动态补偿 PID 微分系数（默认值为 0）

■ 返回值

返回值	类型	含义

返回值	类型	含义
0	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

- 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。

4.4.11.4 HMC_GetGantryAxisPID（读取龙门同步轴补偿 PID 参数）

- 函数原型

  UDINT HMC_ GetGantryAxisPID (USINT ucAxisID， LREAL dKpGain，LREAL dKiGain，LREAL dKdGain)

- 功能说明

  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。

- 输入参数

输入参数	类型	含义
UCAXISID	USINT	龙门组轴号

- 输入输出参数

输入输出参数	类型	含义
DKPGAIN	LREAL	动态补偿 PID 比例系数
DKIGAIN	LREAL	动态补偿 PID 积分系数
DKDGAIN	LREAL	动态补偿 PID 微分系数

- 返回值

返回值	类型	含义
0	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

- 补充说明

  轴号不为龙门轴时，返回错误。

4.4.11.5 HMC_GantryGroupDefPos（设置龙门同步轴组当前位置）

- 函数原型

  UDINT HMC_ GantryGroupDefPos (USINT* pAxis，USINT ucAxisNum，LREAL dMpos)

- 功能说明

  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。

- 输入参数

输入参数	类型	含义
PAXIS	POINTER TO USINT	龙门组同步轴
UCAXISNUM	USINT	龙门组同步轴个数（≤8）
DMPOS	LREAL	预定义位置

- 返回值

返回值	类型	含义
0	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

- 补充说明

  当前指令不为龙门同步指令时，返回错误。

4.4.11.6 HMC_GetGantryInitState（读取龙门同步控制函数当前状态）

- 函数原型

  UDINT HMC_GetGantryInitState (USINT* pAxis，USINT ucAxisNum)

- 功能说明

  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。

- 输入参数

输入参数	类型	含义
PAXIS	POINTER TO USINT	龙门组同步轴
UCAXISNUM	USINT	龙门组同步轴个数

- 返回值

返回值	类型	含义
1：Homing 2：ReturnBack 3：Defpos 4：Compsate 5：Finish	UDINT	龙门初始化指令执行状态
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

4.4.12 轴位置补偿功能

4.4.12.1 HMC_SetAxisCompenPos（轴位置补偿量设定）

- 函数原型

  UDINT HMC_ SetAxisCompenPos (USINT ucAxisID, LREAL dCompenVal)

- 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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dCompenPos	LREAL	位置补偿值

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

4.4.12.2 HMC_SetAxisCompenPara（轴位置补偿运动参数设定）

- 函数原型

  UDINT HMC_SetAxisCompenPara (USINT ucAxisID, USINT ucJerkMode, LREAL dVmax, LREAL dAccel, LREAL dDecel, LREAL dJerk)

- 功能说明

  设定轴位置补偿运动的运动学参数，实时生效。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dJerkMode	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	LREAL	位置补偿动作最大速度
dAccel	LREAL	位置补偿动作加速度
dDecel	LREAL	位置补偿动作减速度
dJerk	LREAL	S 形加减速加加速度

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

4.4.12.3 HMC_TrigAxisCompens（触发位置补偿）

- 函数原型

  UDINT HMC_TrigAxisCompens (USINT ucAxisID)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

4.4.12.4 HMC_StopAxisCompens（停止位置补偿）

- 函数原型

  UDINT HMC_StopAxisCompens (USINT ucAxisID)

- 功能说明

  立即停止正在进行中的轴位置补偿动作。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

4.4.12.5 HMC_GetAxisCompenState（获取轴位置补偿状态）

- 函数原型

  UDINT HMC_GetAxisCompenState (USINT ucAxisID)

- 功能说明

  获取轴补偿状态。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	补偿动作执行完毕
1	UDINT	补偿动作进行中
0xFFFFFFFF	UDINT	轴参数错误

- 指令类型

  非轴运动缓存指令。

4.5 机器人控制指令

4.5.1 Stewart 6 自由度并联机构

4.5.1.1 HMC_StewartControl（Stewart 机构控制）

- 函数原型

  UDINT HMC_StewartControl (POINTER TO USINT UserSpaceAxisID, POINTER TO USINT DriveSpaceAxisID, POINTER TO LREAL StewartPara)

- 功能说明

  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 196 所示，有关 Gough-Stewart 平台的详细信息参见补充说明。

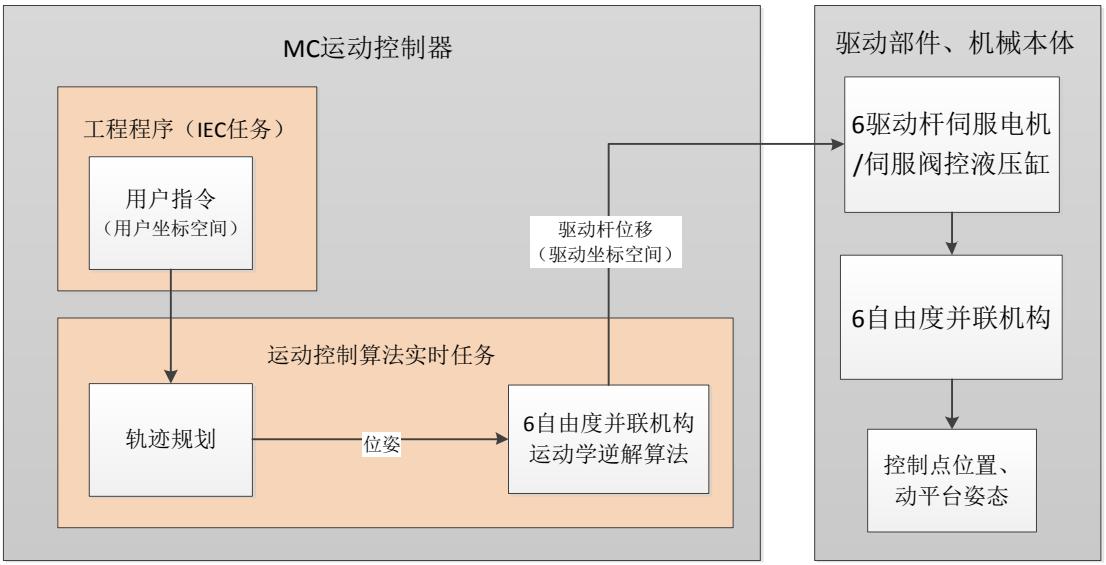
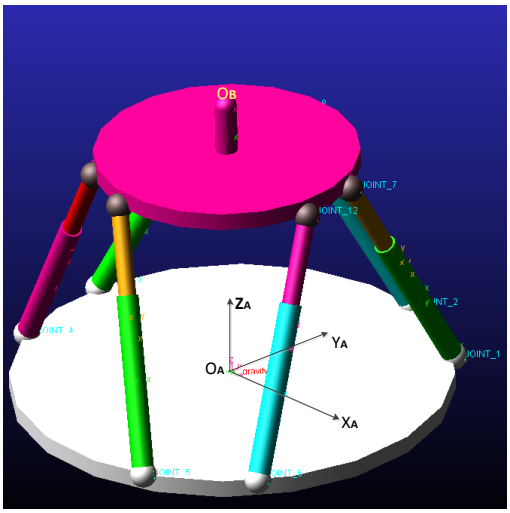


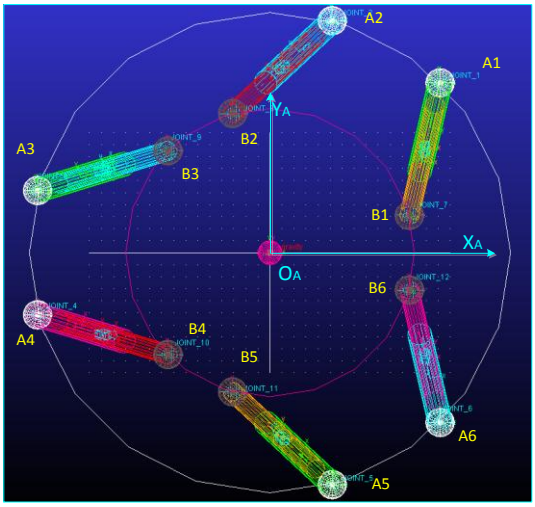
图 196　六自由度并联机构控制系统框图

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）

图 197　并联六自由度平台坐标与机构参数示意图

如图 197 所示，目标控制点O_B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$O_A - X_A Y_A Z_A$如下：

（1）坐标原点位于静平台铰点外接圆圆心O_A；

（2）Z_A轴垂直与静平台平面向上；

（3）X_A轴位于底面，且垂直下铰点A_1和A_6的连线；

（4）Y_A轴方向可由右手法则确定。

设x、y、z为目标控制点O_B在$O_A - X_A Y_A Z_A$中的位置坐标，φ_x、φ_y、φ_z为动平台在$O_B - X_B Y_B Z_B$中的姿态坐标。则动平台在绝对坐标系$O_A - X_A Y_A Z_A$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l_{init}。

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$\mathrm{l_{init}}$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$\mathrm{O_A - X_A Y_A Z_A}$中的位姿，把驱动坐标空间各轴的位移设定为初始杆长$\mathrm{l_{init}}$，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$l_{\min}$与$l_{\max}$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$O_A - X_A Y_A Z_A$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。

■ 输入参数

输入参数	类型	参数描述
UserSpaceAxisID	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$\emptyset_x \emptyset_y \emptyset_z$。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	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	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R_0 2、静平台铰点 短边对应圆心角度θ_0（$0° \le \theta_0 < 60°$） 3、动平台铰点外接圆半径 R_1 4、动平台铰点 短边对应圆心角度θ_1（$0° \le \theta_1 < 60°$） 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}

- 输入参数长度单位为用户单位，角度单位为角度制（°）。

■ 返回值

返回值	类型	参数描述
成功：指令 ID 错误：16#FFFFFFFF	UDINT	--

■ 指令类型

轴运动缓存指令。

■ 补充说明

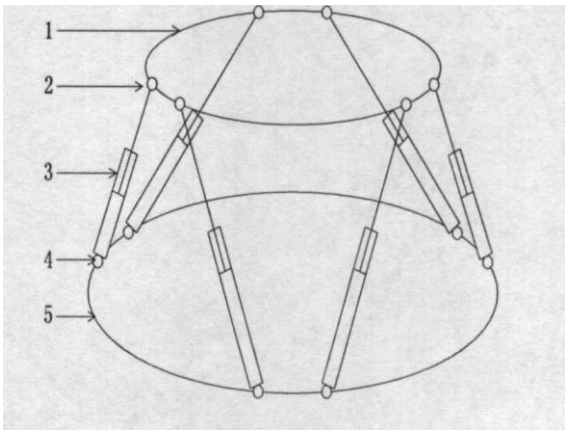


图 198 并联六自由度平台的机构原理

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图 198 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。

■ 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 USINT		FALSE
0002	DriveSpaceAxisID		驱动轴号数组	ARRAY[0..5] OF USINT		FALSE
0003	StewartPara		机构参数	ARRAY[0..7] OF LREAL		FALSE
0004	Err		返回值	DWORD	0	FALSE

```
0001 (*投送StewartControl指令*)
0002 Err := HMC_StewartControl (ADR(UserSpaceAxisID) ,ADR(DriveSpaceAxisID),ADR(StewartPara));
0003
```

图 199 程序示例

□ 变量定义

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

图 200 变量定义

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

图 201 变量定义

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]		静平台半径	LREAL	20000	FALSE
0002	StewartPara[1]		静平台短边角度	LREAL	30	FALSE
0003	StewartPara[2]		动平台半径	LREAL	10000	FALSE
0004	StewartPara[3]		动平台短边角度	LREAL	30	FALSE
0005	StewartPara[4]		控制点距离动平台...	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

图 202 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

```
0001 (*X轴位置调整*)
0002 HMC_MoveAbs(10,Dis_x);
0003 (*Y轴位置调整*)
0004 HMC_MoveAbs(11,Dis_y);
0005 (*Z轴位置调整*)
0006 HMC_MoveAbs(12,Dis_z);
0007 (*X轴姿态调整*)
0008 HMC_MoveAbs(13,Ang_x);
0009 (*Y轴姿态调整*)
0010 HMC_MoveAbs(14,Ang_y);
0011 (*Z轴姿态调整*)
0012 HMC_MoveAbs(15,Ang_z);
0013 (*也可投送球弧或样条指令，来规划用户轴的位置和姿态*)
0014
```

图 203 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		驱动轴数组第一个元素轴号	BYTE	0	FALSE

```
0001 (*撤销StewartControl指令*)
0002 HMC_Cancel (AxisID ,1);
0003
```

图 204 程序示例

4.5.2 5 自由度 SCARA 机械臂

4.5.2.1 HMC_Scara5Control（5 自由度 SCARA 机械臂控制）

- 函数原型

```
UDINT HMC_Scara5Control(
POINTER TO USINT arrBaseCoordAxisID,
POINTER TO USINT arrJointCoordAxisID,
POINTER TO LREAL arrScaraLinkLength)
```

- 功能说明

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 205 所示。

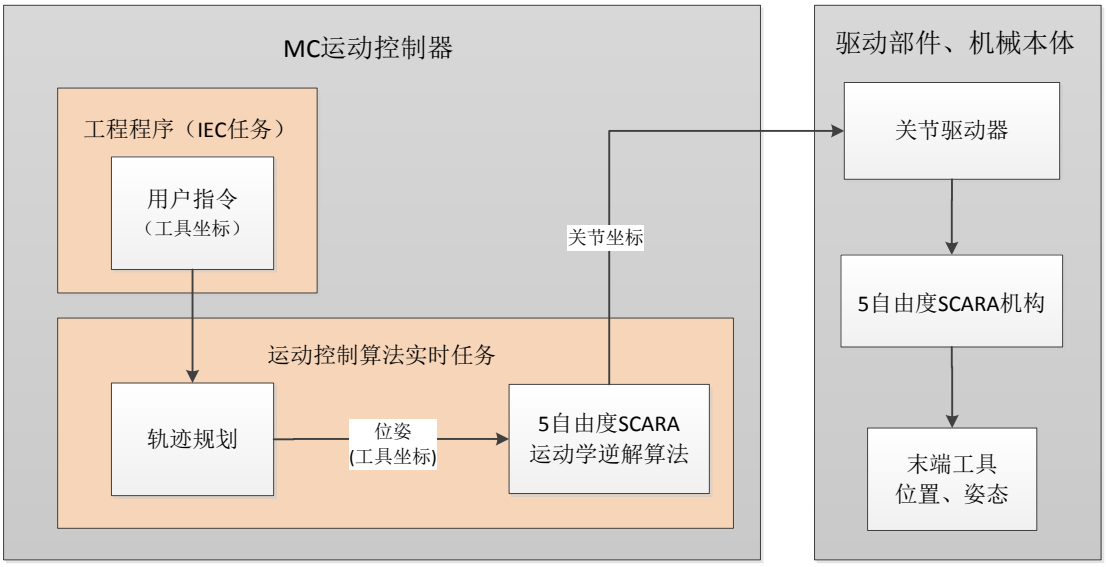


图 205　5 自由度 SCARA 串联机构控制系统框图

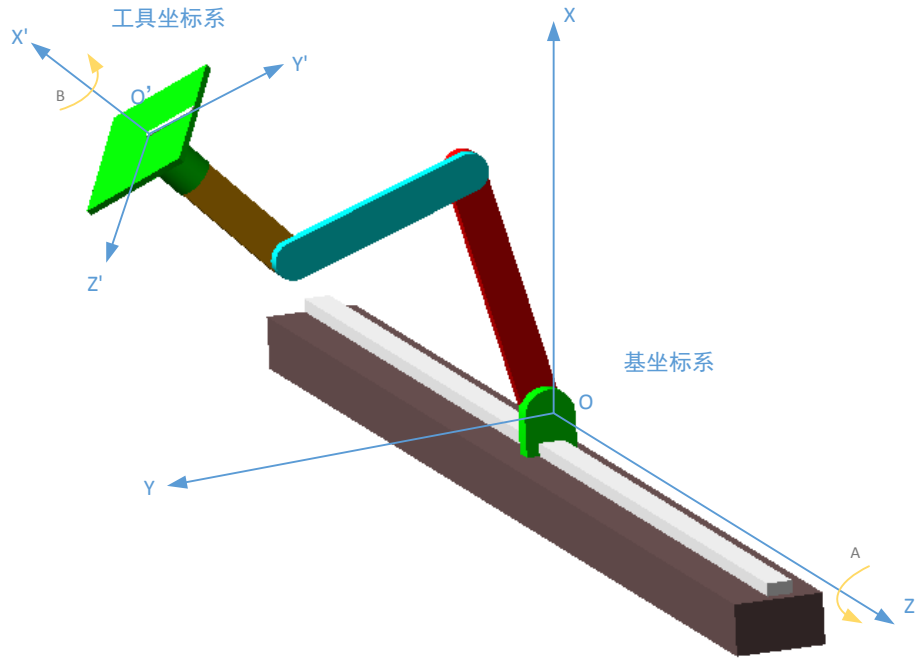


图 206　5 自由度 SCARA 机械手基坐标系、工具坐标系定义（笛卡尔坐标）

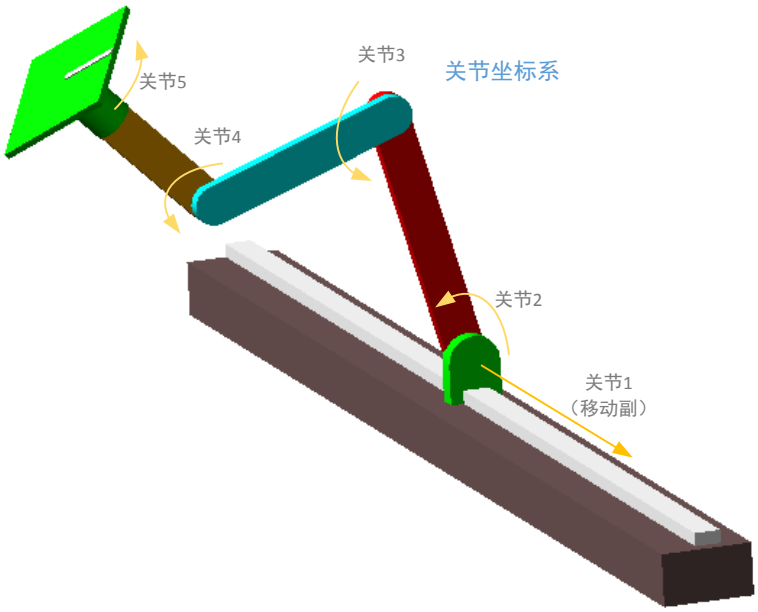


图 207　5 自由度 SCARA 机械手关节坐标系定义（广义坐标）

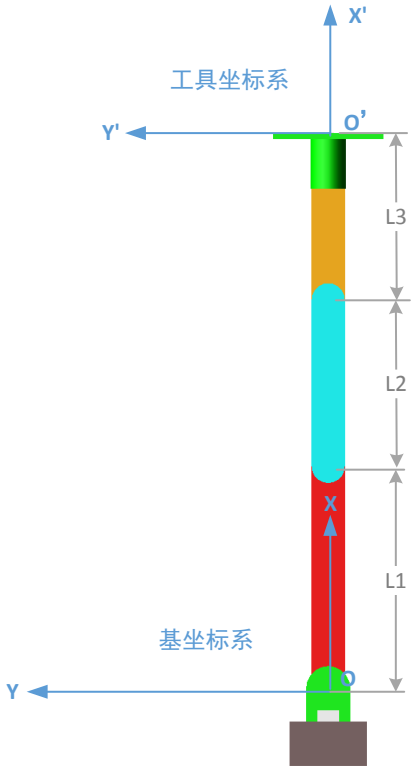


图 208　5 自由度 SCARA 机械手关节角零点位姿

定义基坐标系、工具坐标系、关节坐标系如图 206、图 207 所示，关节角零点位置如图 208 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 206 中 A 轴、B 轴。

关节角的零点位置如图 208 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 208 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）。

■ 输入参数

输入参数	类型	参数描述
arrBaseCoordAxisID	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 206、图 207 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	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 206、图 207 所示 物理输出轴，驱动机构 5 个关节运动
arrScaraLinkLength	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 208 中 L1、L2、L3

■ 返回值

返回值	类型	参数描述
成功：指令 ID	UDINT	--

返回值	类型	参数描述
错误：16#FFFFFFFF		

- 指令类型

  轴运动缓存指令。

- 使用示例

  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
0001	Err0		返回值	DWORD
0002	UserAxis		用户基坐标轴数组	ARRAY[0..4] OF BYTE
0003	DriveAxis		关节驱动轴数组	ARRAY[0..4] OF BYTE
0004	LinkLen		杆长数组	ARRAY[0..2] OF LREAL

```
0001 (*投送Scara5Control指令*)
0002 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）

Scara5Control.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）

Scara5Control.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）

Scara5Control.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
0001	UserAxis		用户基坐标轴数组	ARRAY[0..4] OF BYTE
0002	ToolCoor		工具坐标	ARRAY[0..4] OF LREAL

```
0001 (*用户基坐标投送MoveAbsND指令，给定用户基坐标系中的工具坐标*)
0002 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
0001	AxisID		驱动轴最小轴号	BYTE	0
0002	Modex		撤销模式	BYTE	1

```
0001 (*撤销指令*)
0002 HMC_Cancel(AxisID,Modex);
```

4.5.2.2 HMC_ Scara5ForwardKinematics（5 自由度 SCARA 运动学正解）

- 函数原型

  UDINT HMC_Scara5ForwardKinematics(
  POINTER TO LREAL arrScaraLinkLength,
  POINTER TO LREAL arrToolCoordinate,
  POINTER TO LREAL arrJointAngle)

- 功能说明

  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）。

- 输入参数

输入参数	类型	参数描述
arrScaraLinkLength	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 208 中 L1、L2、L3
arrToolCoordinate	POINTER TO LREAL	工具坐标数组，共 5 个元素，保存正解结果
arrJointAngle	POINTER TO LREAL	关节坐标数组，共 5 个元素

- 返回值

返回值	类型	参数描述

返回值	类型	参数描述
成功：0 错误：16#FFFFFFFF	UDINT	--

■ 指令类型

非轴运动缓存指令。

4.5.2.3 HMC_Scara5ReverseKinematics（5 自由度 SCARA 运动学逆解）

■ 函数原型

UDINT HMC_Scara5ReverseKinematics(

POINTER TO LREAL arrScaraLinkLength,

POINTER TO LREAL arrToolCoordinate,

POINTER TO LREAL arrReferenceJointAngle,

POINTER TO LREAL arrJointAngle)

■ 功能说明

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：

（1）检验关键点的工具坐标是否位于可达工作空间之内；

（2）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。

■ 输入参数

输入参数	类型	参数描述
arrScaraLinkLength	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 208 中 L1、L2、L3
arrToolCoordinate	POINTER TO LREAL	工具坐标数组，共 5 个元素，即（$x, y, z, \varphi_A, \varphi_B$）
arrReferenceJointAngle	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	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

■ 返回值

返回值	类型	参数描述
成功：0	UDINT	当用户输入参数错误 或 工具坐标超出机构的可达

返回值	类型	参数描述
错误：16#FFFFFFFF		工作空间时，返回错误

- 指令类型

  非轴运动缓存指令。

4.5.3 4 自由度串联机械臂

4.5.3.1 HMC_Serial4Control（4 自由度串联机械臂控制）

- 函数原型

  UDINT HMC_Serial4Control (

  POINTER TO USINT arrToolCoordAxisID,

  POINTER TO USINT arrJointCoordAxisID,

  POINTER TO LREAL arrLinkLength)

- 功能说明

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。

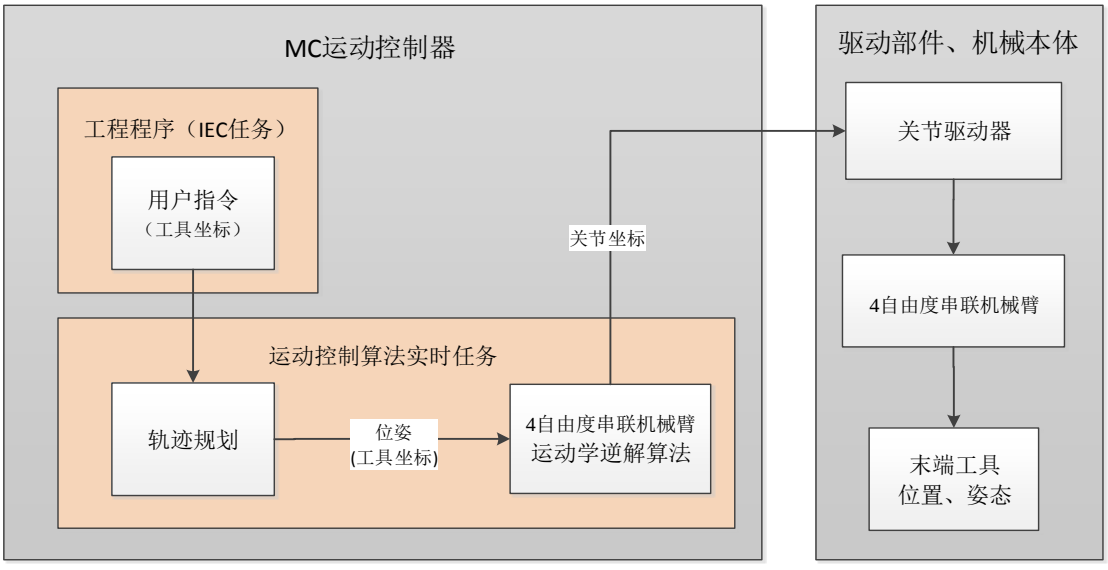
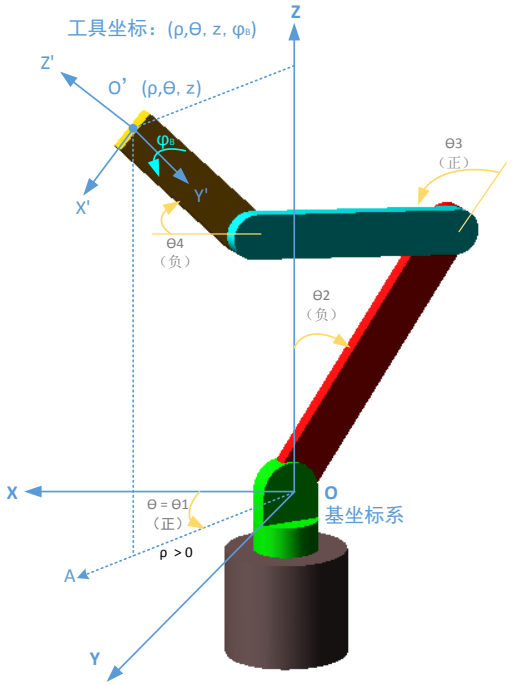
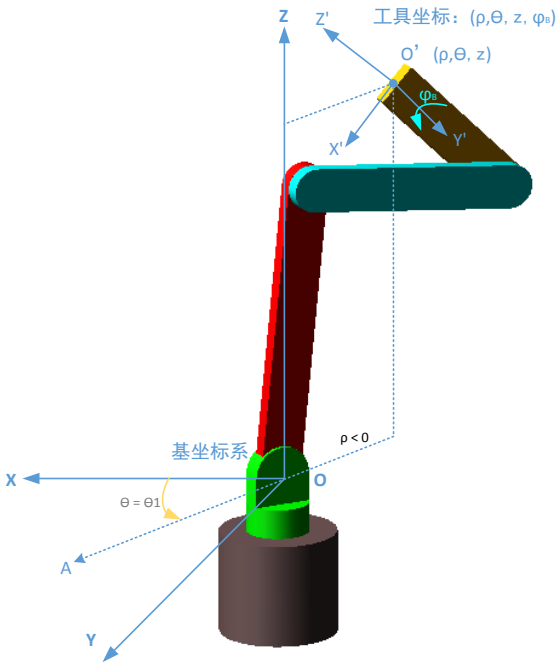


图 209 4 自由度串联机械臂控制系统框图



（a）



（b）

图 210　4 自由度机械手 工具坐标

- 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；
- 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。

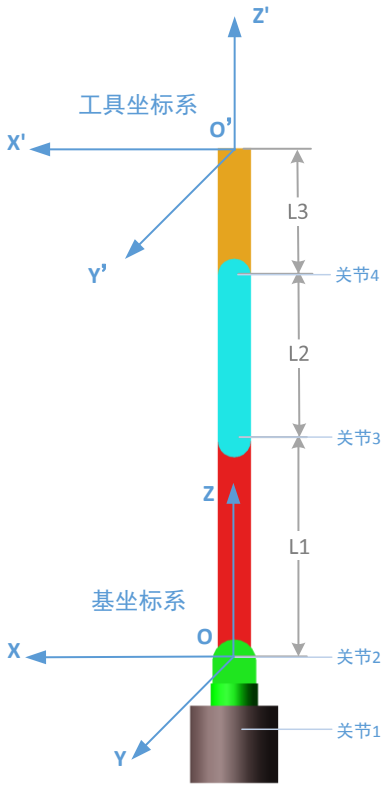


图 211 自由度串联机械臂关节角零点位姿

定义工具坐标系 O’-X’ Y’ Z’，如图 210、图 211 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 210（a）、图 210（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 210 所示。各关节角零点位置如图 211 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 211 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 211 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）。

■ 输入参数

输入参数	类型	参数描述
arrToolCoordAxisID	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（$\rho, \theta, z, \varphi_B$）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	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 211 所示 物理输出轴，驱动 4 个关节运动
arrLinkLength	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 211 中 L1、L2、L3

■ 返回值

返回值	类型	参数描述
成功：指令 ID 错误：16#FFFFFFFF	UDINT	--

■ 指令类型

轴运动缓存指令。

■ 使用示例

指令使用步骤如下：

第1步 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*)
```

第2步 定义各关节轴的零点在如图 211 所示的位置（关节轴实际位置不必在零位）

第3步 向关节轴中投送 HMC_Serial4Control 指令

```
HMC_Serial4Control(arrToolCoordAxisID , arrJointCoordAxisID , arrLinkLength);
```

第4步 使用工具坐标轴编程，坐标为圆柱坐标系

（1） 移动到图 210（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° *)
```

（2） 移动到图 210（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° *)
```

（3） 如果需要经过一系列中间点，可使用样条插补指令。

4.5.3.2 HMC_ Serial4ForwardKinematics（4 自由度串联机械臂运动学正解）

- 函数原型

```
UDINT HMC_ Serial4ForwardKinematics(
POINTER TO LREAL arrLinkLength,
POINTER TO LREAL arrToolCoordinate,
POINTER TO LREAL arrJointAngle)
```

- 功能说明

  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）。

- 输入参数

输入参数	类型	参数描述
arrLinkLength	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 211 中 L1、L2、L3
arrToolCoordinate	POINTER TO LREAL	工具坐标数组，共 4 个元素（$\rho, \theta, z, \varphi_B$）。保存正解结果
arrJointAngle	POINTER TO LREAL	关节坐标数组，共 4 个元素

- 返回值

返回值	类型	参数描述
成功：0 错误：16#FFFFFFFF	UDINT	--

4.5.3.3 HMC_ Serial4ReverseKinematics（4 自由度串联机械臂运动学逆解）

- 函数原型

```
UDINT HMC_Serial4ReverseKinematics(
POINTER TO LREAL arrLinkLength,
POINTER TO LREAL arrToolCoordinate,
POINTER TO LREAL arrReferenceJointAngle,
POINTER TO LREAL arrJointAngle)
```

- 功能说明

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：

（1）检验关键点的工具坐标是否位于可达工作空间之内；

（2）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。

- 输入参数

输入参数	类型	参数描述
arrLinkLength	POINTER TO LREAL	连接杆杆长数组，共 3 个元素，依此对应图 211 中 L1、L2、L3
arrToolCoordinate	POINTER TO LREAL	工具坐标数组，共 4 个元素（$\rho, \theta, z, \varphi_B$）
arrReferenceJointAngle	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

输入参数	类型	参数描述
arrJointAngle	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

- 返回值

返回值	类型	参数描述
成功：0 错误：16#FFFFFFFF	UDINT	当用户输入参数错误或工具坐标超出机构的可达工作空间时，返回错误。

- 指令类型

  非轴运动缓存指令。

4.6 自定义运动控制功能

4.6.1 HMC_SetDposSwitch（Dpos 规划开关切换）

- 函数原型

  UDINT HMC_SetDposSwitch (USINT ucAxisID, USINT ucDposSwitch)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucDposSwitch	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

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

4.6.2 HMC_SetDposVal（Dpos 规划）

- 函数原型

  UDINT HMC_SetDposVal (USINT ucAxisID, LREAL dSetDposVal)

■ 功能说明

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）。

■ 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dSetDposVal	LREAL	设定的 Dpos 值

■ 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

■ 指令类型

非轴运动缓存指令。

■ 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

图 212 程序示例

```
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）：

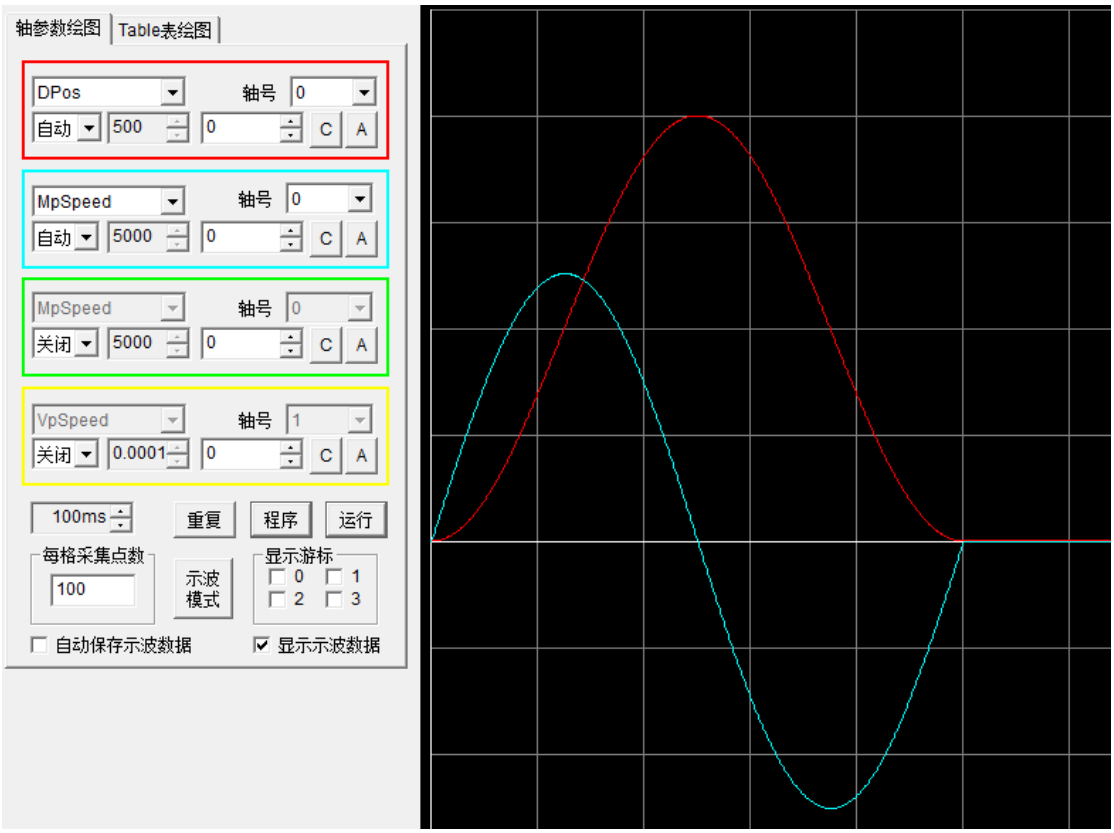


图 213 轴 0 Dpos 及 MpSpeed

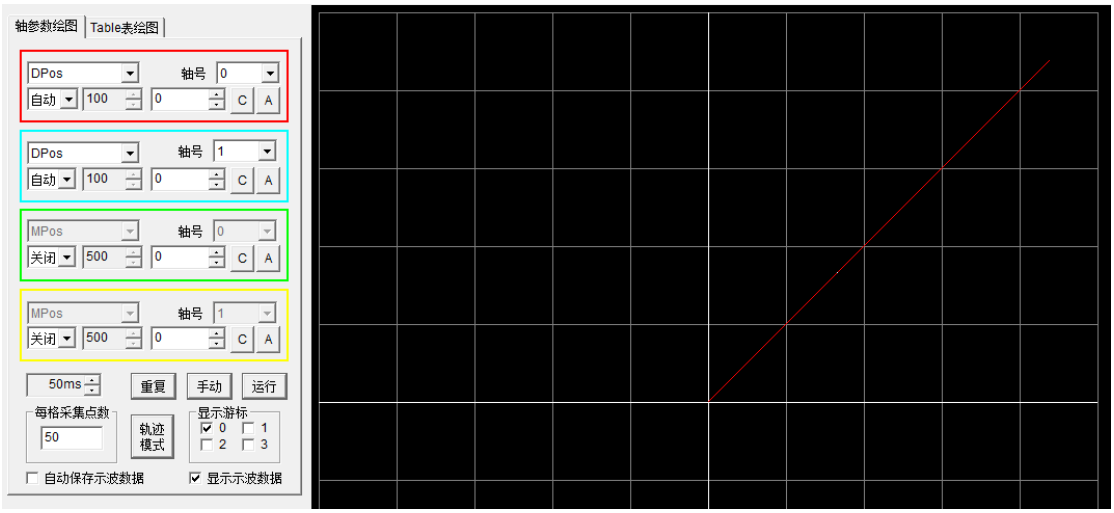


图 214 轴 0、轴 1 合运动轨迹

4.6.3 HMC_SynchroToAlm（同步到算法任务）

- 函数原型

  UDINT HMC_SynchroToAlm ()

- 功能说明

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）。

- 输入参数

  无

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功

- 指令类型

  非轴运动缓存指令。

4.7 轴参数设定指令

4.7.1 HMC_SetAxisType（设置轴类型）

- 函数原型

  UDINT HMC_SetAxisType (USINT ucAxisID, EmAxisType ucAxisType)

- 功能说明

  设置指定轴的轴类型，各轴类型含义详见轴参数 AxisType。

  （1）轴号 0~63 的不同区间范围所支持的轴类型不同。

  （a）0≤轴号<控制器支持的直连轴总数

  不同型号控制器支持的直连轴总数不同，如 MC1008 型号控制器的直连轴总数为 8，MC1004 型号控制器的直连轴总数为 4，MC1002E 型号控制器的直连轴总数为 2。

  轴号在此范围时，可设置轴类型为该控制器型号所支持的任意轴类型。

  （b）直连轴总数≤轴号<（直连轴总数+控制器支持的手轮总数）

  不同型号控制器支持的手轮总数不同，MC1008 型号控制器的直连轴总数为 2，其他型号控制器的直连轴总数为 1。

  轴号在此范围时，可设置轴类型为虚轴、未使用轴、三种编码器轴类型。若为总线型号控制器，也支持该总线类型对应的总线轴类型。

  （c）（直连轴总数+控制器支持的手轮总数）≤轴号<64

  轴号在此范围时，可设置轴类型为虚轴或未使用轴，若为总线型号控制器，也支持该总线类型对应的总线轴类型。

  各控制器型号的不同轴号范围支持的轴类型请参见《MC 系列运动控制器系统手册》中的附录 3 MC 控制器支持的轴类型。

  （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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
EmAxisType	ucAxisType	轴类型

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 应用举例

```
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*)
```

- 不支持的模块版本（直连步进轴：脉冲从 Do0~3 输出）

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A01

- 不支持的模块版本（增量式编码器：脉冲从 DI0~3 输入）

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.7.2 HMC_HwLimitConfig（设置硬限位开关）

- 函数原型

  UDINT HMC_ HwLimitConfig (USINT ucAxisID, USINT ucLimitType, INT sDiChan)

- 功能说明

  设置指定轴的硬限位开关，限位功能仅对有限行程轴有效。

  该函数配置指定某路 DI 为轴的前向或后向硬限位开关，当前（后）向限位 DI 通道为高电平时，认为该轴到达了前（后）向限位位置，系统将根据 FsLimitMode/RsLimitMode 值来进行有限行程限位处理。

  限位开关触发后，轴的运动方式详见章节 10.3 有限行程轴超限处理。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucLimitType	USINT	0：设置前向硬限位 1：设置后向硬限位
sDiChan	INT	关联的 DI 通道（0~63） 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

输入参数	类型	参数描述
		位 DI 通道，即使当前处于限位状态也会立即解除

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 应用举例

  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);
```

4.7.3 HMC_SetUnits（设置用户单位）

- 函数原型

  UDINT HMC_SetUnits (USINT ucAxisID, LREAL dUnits)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dUnits	LREAL	用户单位，不可为 0 或负数

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 应用举例

  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。

4.7.4 HMC_SetJerkMode（设置梯形/S 形加速度）

- 函数原型

  UDINT HMC_SetJerkMode(USINT ucAxisID, USINT ucJerkMode)

- 功能说明

  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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucJerkMode	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。

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01、MC1002E-A01

4.7.5 HMC_SetKe（设置 Ke）

- 函数原型

  UDINT HMC_ SetKe (USINT ucAxisID, DINT iKeNumerator, DINT iKeDenominator)

- 功能说明

  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。

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”，参见章节 3.2.3 Units（单位转换因子）。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
iKeNumerator	DINT	Ke 分子
iKeDenominator	DINT	Ke 分母

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 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。

- 应用举例

  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(0,30);
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*)
```

4.7.6 HMC_SetKs（设置 Ks）

- 函数原型

  UDINT HMC_ SetKs (USINT ucAxisID, DINT iKsNumerator, DINT iKsDenominator)

- 功能说明

  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。

  该参数仅对步进轴有效。该参数和 Units 有类似的作用，(Units*Ks)共同构成了步进轴的“单位转换因子”，其单位为“线/用户单位”，参见章节 3.2.3 Units（单位转换因子）。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

输入参数	类型	参数描述
iKsNumerator	DINT	Ks 分子
iKsDenominator	DINT	Ks 分母

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 补充说明

  Ks 需＞0。

- 应用举例

  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(0,10);
HMC_SetUnits(0,1);
HMC_SetKs(0,8192,360);
HMC_Move(0,180);      (*驱动转盘旋转 180 度*)
```

4.7.7 HMC_SetFeedBackSrcAddr（设置用户自定义反馈输入地址）

- 函数原型

  UDINT HMC_SetFeedBackSrcAddr (USINT ucAxis, POINTER TO LREAL pAddr, USINT DataType)

- 功能说明

  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，如编码器轴、伺服轴、总线轴等。

- 输入参数

输入参数	类型	参数描述
ucAxis	USINT	轴号
pAddr	POINTER TO LREAL	用户自定义变量地址
DataType	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

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 应用举例

  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 (0,30);
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*)
```

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.7.8 HMC_SetOutDstAddr（设置用户自定义输出地址）

- 函数原型

  UDINT HMC_SetOutDstAddr (USINT ucAxis, POINTER TO DINT pAddr, USINT DataType)

- 功能说明

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，如伺服轴、总线速度模式等。

- 输入参数

输入参数	类型	参数描述
ucAxis	USINT	轴号
pAddr	POINTER TO DINT	用户自定义变量地址
DataType	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、

输入参数	类型	参数描述
		LREAL=8

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他值	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 应用举例

  设置伺服轴轴 0 的输出目标为系统变量 diOutPut。

```
HMC_SetOutDstAddr(0, ADR(diOutPut), 6);
```

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.7.9 HMC_SetSSIEncoderBit (设置 SSI 编码器位数)

- 函数原型

  UDINT HMC_SetSSIEncoderBit (USINT ucAxisID, USINT ucSSIEncoderBit)

- 功能说明

  设置 SSI 编码器位数，默认值为 12。

  如果该轴类型已经为 SSI 轴类型（33），则该指令执行成功后会重新初始化 Mpos/Dpos 值为当前 SSI 编码器的绝对位置。

  在使用 SSI 绝对值编码器时，应按以下步骤进行操作

  （1）硬件接线

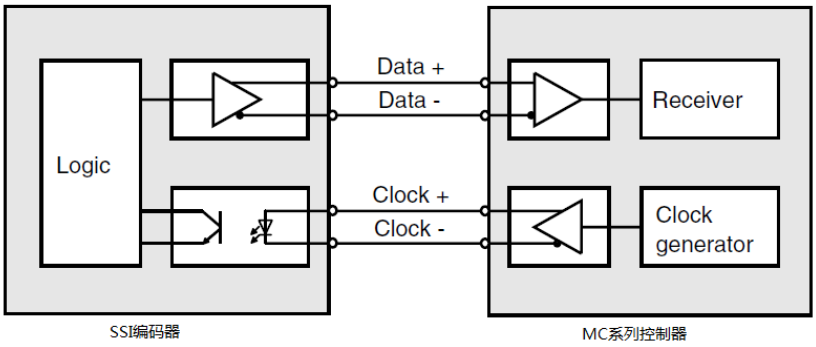


图 215 硬件接线示意图

  （2）软件操作

  步骤 1：配置 SSI 编码器相关参数

  根据编码器实际规格，配置对应的 SSI 编码器相关参数

配置项	相关函数	实时修改是否初始化位置
编码器位数	HMC_SetSSIEncoderBit	是
编码器码制	SetSSIEncoderDataType	是
SSI 总线时钟频率	HMC_SetSSIEncoderFreq	否
SSI 总线数据锁存时间	HMC_SetSSIEncoderTp	否

步骤 2：设置关联轴的轴类型为 SSI 编码器轴类型

参数配置成功后，设置轴类型 HMC_SetAxisType(AxisID, 33)。

轴类型成功为 SSI 编码器轴类型后，会根据读取到当前编码器的实际位置，并依此初始化当前轴位置 MPos。再此之后，则在此初始位置基础上增量更新每个伺服周期位置增量，计算轴的实时位置值。

其它注意事项：在线修改 SSI 配置参数

轴类型成功为 SSI 编码器轴类型后，位置采集更新过程中，仍可调用步骤 1 的函数重新配置 SSI 编码器相关参数时，且实时生效，但根据不同参数，会决定是否影响重新初始化轴的实际位置 MPos，即上表格第三列。行程设置

如果需要 MPos 的值始终与编码器绝对位置一致，则可设定该轴为循环行程（RepMode=1），且循环行程的行程长度与编码器的位置数据范围一致，如 12 位编码器，在 Units、Ke 均为 1 的情况下，RepMode=1 时应设置 RepDist = 2^12。

在设置 RepMode=2、RepDist=（编码器位置数据范范围/2）的情况下，可实现该轴 Mpos 区间长度=编码器位置数据区间长度，并且 Mpos 的范围区间是关于零点对称的，如 12 位编码器，在 Units、Ke 均为 1 的情况下，应设置 RepMode=2、RepDist = 2^12/2。

如果设定该轴为有限行程，则如果编码器未发生数据溢出时，Mpos 与编码器绝对位置一致；当编码器发生数据溢出时，则控制器仍按实际增量位移计算 Mpos（不跳变），而此时编码器内部的绝对位置数据则因数据溢出而发生了跳变。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucSSIEncoderBit	USINT	编码器位数(默认值 12) 范围：1~127

- 返回值

返回值	类型	参数描述
0	UDINT	设定成功
其它	UDINT	错误码

- 指令类型

非轴运动缓存指令

- 补充说明

MC1008 支持 4 个 SSI 轴，MC1004 支持 2 个 SSI 轴，MC1002R/MC1002E 暂不支持。支持 SSI 轴的轴号，MC1008 限定为 0、1、2、3，MC1004 限定为 0、1。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-XXX、MC1002E-XXX

- 应用举例

  使用 SSI 单圈绝对值编码器其规格：

接口电气参数	
传感器	
供电电压	5 (+0.4) V DC [1]
功耗（无负载）	< 40 mA
电源反极性保护	yes
测量范围	360°
分辨率/编码	12位/格雷码
线性度(25℃)	< 1.5°
重复精度	≤ 0.4°
数据刷新速率	典型100 µs
符合RoHS标准，据	EU标准 2002/95/EG
符合CE标准，据	EN 61000-6-2, EN 55011 Class B
SSI接口	
时钟脉冲速度	100 kHz ... 750 kHz
输出驱动	RS485
单稳态时间/最大	16 µs / 20 µs
输出短路保护	有 [2]
允许负载/通道	典型60 Ohm (对应RS485)

图 216 SSI 单圈绝对值编码器规格

则编写程序如下：

```
SSIEncoderAxisID := 0;        (*SSI 绝对值编码器关联轴 0*)

HMC_SetSSIEncoderBit (SSIEncoderAxisID ,12);       (*设置位数*)
HMC_SetSSIEncoderDataType (SSIEncoderAxisID ,1);  (*设置编码方式*)
HMC_SetSSIEncoderFreq (SSIEncoderAxisID ,100);     (*设置总线时钟频率*)
HMC_SetSSIEncoderTp (SSIEncoderAxisID ,100);       (*设置数据锁存时间*)

HMC_SetAxistype(SSIEncoderAxisID, 33);      (*正确配置参数后，设置轴类型为 SSI*)
```

4.7.10 HMC_SetSSIEncoderDataType (设置 SSI 编码器码制)

- 函数原型

  UDINT HMC_ SetSSIEncoderDataType (USINT ucAxisID,

  USINT ucSSIEncoderDataType)

- 功能说明

  配置 SSI 编码器轴码制。0 为二进制编码，1 为格雷码；默认值为 1。

  如果该轴类型已经为 SSI 轴类型（33），则该指令执行成功后会重新初始化 Mpos/Dpos 值为当前 SSI 编码器的绝对位置。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

输入参数	类型	参数描述
ucSSIEncoderDataType	USINT	码制（0 为二进制编码；1 为格雷码。默认值 1）

- 返回值

返回值	类型	参数描述
0	UDINT	设定成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-XXX、MC1002E-XXX

- 应用举例

  参考 HMC_SetSSIEncoderBit 章节

4.7.11 HMC_SetSSIEncoderFreq (设置 SSI 总线时钟频率)

- 函数原型

  UDINT HMC_ SetSSIEncoderFreq (USINT ucAxisID, LREAL dSSIEncoderFreq)

- 功能说明

  配置 SSI 总线时钟频率。默认值为 200 kHz。

  如果该轴类型已经为 SSI 轴类型（33），则可在线修改时钟频率，实时生效，且不影响当前 Mpos/Dpos 值。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
dSSIEncoderFreq	LREAL	时钟频率(单位 KHZ，默认值 200KHZ) 范围：40~10000

- 返回值

返回值	类型	参数描述
0	UDINT	设定成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-XXX、MC1002E-XXX

- 备注

将频率换算为周期，如果周期不是 0.1us 的整数倍，则就近取整。再折算为频率，如果两次设置频率相同，则返回错误码。

- 应用举例

  参考章节 4.7.9 HMC_SetSSIEncoderBit (设置 SSI 编码器位数)。

4.7.12 HMC_SetSSIEncoderTp (设置 SSI 总线数据锁存时间)

- 函数原型

  UDINT HMC_SetSSIEncoderTp (USINT ucAxisID, USINT ucSSIEncoderTp)

- 功能说明

  配置 SSI 总线数据锁存时间。默认值为 35 μs。

  如果该轴类型已经为 SSI 轴类型（33），则可在线修改该参数，实时生效，且不影响当前 Mpos/Dpos 值。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucSSIEncoderTp	USINT	数据锁存时间（单位 us，默认值 35 μs）

- 返回值

返回值	类型	参数描述
0	UDINT	设定成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-XXX、MC1002E-XXX

- 应用举例

  参考 4.7.9 HMC_SetSSIEncoderBit (设置 SSI 编码器位数)章节。

4.7.13 HMC_SetCmdBufferLoadMode（设置轴指令加载模式）

- 函数原型

  UDINT HMC_SetCmdBufferLoadMode (USINT ucAxisID, USINT ucCmdLoadMode)

- 功能说明

  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）。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucCmdLoadMode	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。

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.7.14 HMC_GetCmdBufferLoadMode（获取轴指令加载模式）

- 函数原型

  USINT HMC_GetCmdBufferLoadMode (USINT ucAxisID)

- 功能说明

  获取轴指令的加载模式。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
指令加载模式	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	USINT	错误码

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1008-C01、MC1004-A01、MC1004-B01、MC1002R-A01

4.7.15 HMC_SetCmdStopMaxTime (设置指令结束最大等待时间)

- 函数原型

UDINT HMC_SetCmdStopMaxTime (UDINT uiTime)

- 功能说明

设置指令结束最大等待时间。

- 输入参数

输入参数	类型	参数描述
uiTime	UDINT	时间（ms）

- 返回值

返回值	类型	参数描述
-	UDINT	0：成功； 其它值：失败。

- 指令类型

非轴运动缓存指令。

- 不支持的模块版本

MC1008-D03 以下版本、MC1004-C03 以下版本、MC1004L-A04 以下版本、MC1002R-B02 以下版本、MC1002E-A04 以下版本

4.8 状态读取指令

4.8.1 HMC_GetSysErr（获取系统错误码）

- 函数原型

UDINT HMC_GetSysErr ()

- 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。

- 返回值

返回值	类型	参数描述
0	UDINT	没有错误
其他值	UDINT	当前发生系统错误

- 指令类型

非轴运动缓存指令。

4.8.2 HMC_GetUserErrID（获取用户错误码）

- 函数原型

  UDINT HMC_ GetUserErrID (UDINT uiErrIndex)

- 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。

- 输入参数

输入参数	类型	参数描述
uiErrIndex	UDINT	错误索引 取值范围： 0~（HMC_GetUserErrNum()-1）

- 返回值

返回值	类型	参数描述
用户错误码	UDINT	-

- 指令类型

  非轴运动缓存指令。

4.8.3 HMC_GetUserErrNum（获取用户错误数）

- 函数原型

  UDINT HMC_GetUserErrNum ()

- 功能

  返回已发生的用户错误数。

  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。

- 返回值

返回值	类型	参数描述
已发生的用户错误数量	UDINT	-

- 指令类型

  非轴运动缓存指令。

4.8.4 HMC_GetUserErrMes（获取用户错误码附加信息）

- 函数原型

UDINT HMC_GetUserErrMes (UDINT uiErrIndex)

- 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。

- 输入参数

输入参数	类型	参数描述
uiErrIndex	UDINT	错误索引 取值范围：0~（HMC_GetUserErrNum()-1）

- 返回值

返回值	类型	参数描述
用户错误码附加信息	UDINT	-

- 指令类型

非轴运动缓存指令。

4.8.5 HMC_GetAllAxisErr（获取所有轴的总体异常状态）

- 函数原型

UDINT HMC_ GetAllAxisErr ()

- 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))。

- 返回值

返回值	类型	参数描述
0	UDINT	所有轴正常
其他值	UDINT	轴异常

- 指令类型

非轴运动缓存指令。

4.8.6 HMC_GetCmdErr（获取轴指令错误）

- 函数原型

UDINT HMC_GetCmdErr (UDINT uiAxisCmdID)

- 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。

- 输入参数

输入参数	类型	参数描述
uiAxisCmdID	UDINT	轴指令 ID

- 返回值

返回值	类型	参数描述
0	UDINT	没有错误
其他值	UDINT	轴指令 ID 不在轴缓存中或指令 ID 非法

- 指令类型

非轴运动缓存指令。

4.8.7 HMC_GetCmdState（获取轴指令状态）

- 函数原型

USINT HMC_GetCmdState (UDINT uiAxisCmdID)

- 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

- 输入参数

输入参数	类型	参数描述
uiAxisCmdID	UDINT	轴指令 ID

- 返回值

返回值	类型	参数描述
0	USINT	等待运行：已经被投放到运动缓存中但尚未开始执行
1	USINT	运行中
2	USINT	已经完成或无指令
3	USINT	正在投送：正在向运动缓存投送
其他值	USINT	错误码（0xFF 为用户参数错误）

- 指令类型

非轴运动缓存指令。

4.8.8 HMC_WaitCmdFinish（等待指令完成）

- 函数原型

  USINT HMC_WaitCmdFinish (UDINT uiAxisCmdID)

- 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。

- 输入参数

输入参数	类型	参数描述
uiAxisCmdID	UDINT	轴指令 ID

- 返回值

返回值	类型	参数描述
0	USINT	该指令运行完成
其他值	USINT	错误码（0xFF 为轴指令 ID 非法）

- 指令类型

  非轴运动缓存指令。

4.8.9 HMC_WaitCmdLoaded（等待指令装载）

- 函数原型

  USINT HMC_WaitCmdLoaded (UDINT uiAxisCmdID)

- 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。

- 输入参数

输入参数	类型	参数描述
uiAxisCmdID	UDINT	轴指令 ID

- 返回值

返回值	类型	参数描述
0	USINT	该指令开始运行
其他值	USINT	错误码（0xFF 为轴指令 ID 非法）

- 指令类型

  非轴运动缓存指令。

4.8.10 HMC_WaitAxisIdle（等待轴空闲）

- 函数原型

  USINT HMC_WaitAxisIdle (USINT uiAxis)

- 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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	USINT	该轴空闲
其他值	USINT	错误码（0x17 为轴号超范围）

- 指令类型

  非轴运动缓存指令。

4.8.11 HMC_GetCmdBufferState（获取轴指令缓存状态）

- 函数原型

  UDINT HMC_GetCmdBufferState (USINT uiAxis)

- 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。

- 输入参数

输入参数	类型	参数描述
ucAxis	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	指令缓存未满，可以正常投放指令
1	UDINT	指令缓存已满，若投放指令则会出现阻塞现象
2	UDINT	指令缓存被锁定（当前有指令正在投放进该缓存中）
0xFFFFFFFF	UDINT	函数参数错误

- 指令类型

  非轴运动缓存指令。

- 不支持的模块版本

  MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01、MC1002R-A01

4.9 故障清除指令

4.9.1 HMC_ClearAxisErr（消除轴错误）

- 函数原型

  UDINT HMC_ ClearAxisErr (USINT uiAxis)

- 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。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	成功
其他值	UDINT	错误码（0xFFFFFFFF 为轴号超范围）

- 指令类型

  非轴运动缓存指令。

4.9.2 HMC_ClearAllErr（消除所有轴错误）

- 函数原型

  UDINT HMC_ClearAllErr ()

- 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。

- 输入参数

  无

- 返回值

返回值	类型	参数描述
0	UDINT	成功

- 指令类型

  非轴运动缓存指令。

第5章　RTEX 总线指令库

本节指令专用于支持松下 RTEX 总线的控制器。

5.1 RTEX_Read（读驱动器信息）

- 功能说明

  通过外部库的方式获取从站信息。

- 输入参数

输入参数	类型	参数描述
StaNo	USINT	从站号
ParaID	UDINT	参数 ID
DataLen	USINT	读取的数据长度
pValue	POINTER TO BYTE	存放读出信息的地址

- 传参说明

参数	值	参数描述	举例
StaNo	0~31	从站号	0
ParaID	0x00000+Class*100+ParNum	读 Para0~8 类参数信息	725：为 Pr7.25
	0x10000+Type_Code	读 system ID 信息	0x10001：读取生产厂家名称（Panasonic）
	0x20000+Index*0x1000	读 Alarm 信息	0x21000：读取 1 号历史报警信息
	0x30000+Index*0x1000+Type_Code	读 Monitor 信息	0x30005：读取实时速度
DataLen	给下来的数据空间大小	system ID 为 20 字节 除了 system ID 外都为 4 字节	0x04
pValue	存放读出信息的首地址	-	-

- system ID 的 Type_Code

Type_Code	值	ParaID（十进制）
VENDOR_NAME	0x01	65537
DEVICE_TYPE	0x05	65541
DRIVER_MODE_NO	0x12	65554
DRIVER_SERIAL_NO	0x13	65555
SERVODVSFV	0x14	65556
DRIVER_TPPE	0x15	65557
MOTOR_DODEL_NO	0x22	65570
MOTOR_SERIAL_NO	0x23	65571

Type_Code	值	ParaID（十进制）
VENDOR_ID_EX	0x31	65585
MODEL_ID_EX	0x32	65586

- Alarm 的 Index

  取值范围：0~14。如表 3 所示。

表 3 当前与历史报警列表

Index		Alarm Code	Warning Code
当前的	0	代表当前报警	代表当前警示
历史的	1	代表倒数第 1 个报警	0
	2	代表倒数第 2 个报警	0
	…	…	…
	14	代表倒数第 14 个报警	0

- 报警未发生时，Alarm Code 为 0。

- Monitor 的 Index 和 Type_Code

  详见附录 2 Monitor 的 Index 和 Type_Code 列表。

- DataLen

  表示调用者为读取结果而预先开辟的空间，一般由读取的内容决定。

  驱动器参数为单字节、双字节、四字节等数据类型，长度分别为 1、2、4 字节。

  如果读取 sytem ID 命令，则需要预先开辟大于等于 20 字节的空间，以便存放字符串，一般取值 20 即可。

- 返回值

返回值	类型	参数描述
0	UINT	成功
0x1000	UINT	读取的类别不存在
0x2000	UINT	数据长度不对
0x3000	UINT	形参指针为空
0x4000	UINT	没有对应的从站号
0x5000	UINT	系统没有进入运行状态
0x6000	UINT	正在执行复位
0x7000	UINT	协议栈异常
0x8000	UINT	只允许制造商读取的值
0x9000	UINT	超时

返回值	类型	参数描述
0x9100	UINT	缓冲区已满
其他值	UINT	参见附录 1 RTEX 指令错误码

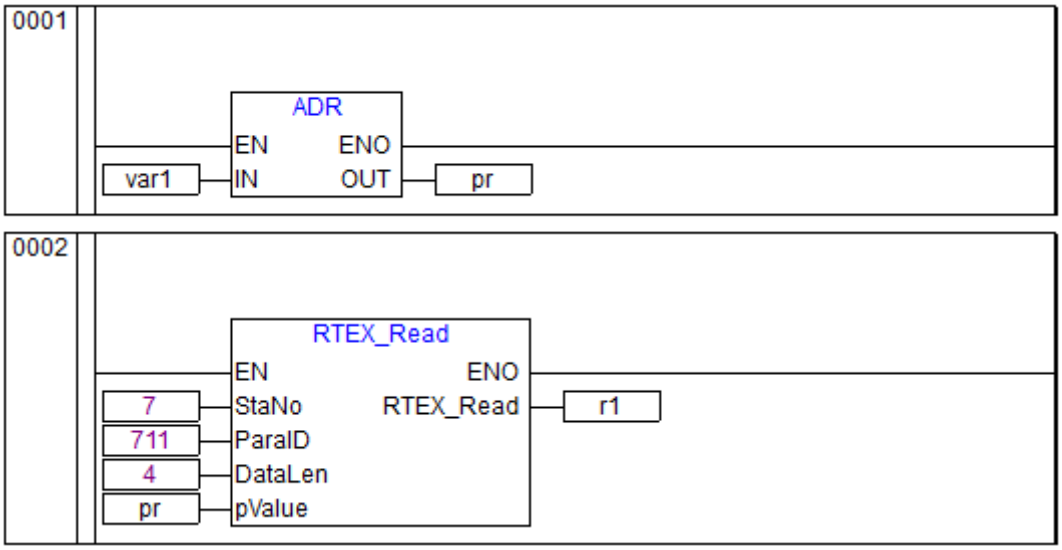
- 示例：读取 7 号站 Pr7.11 参数值。

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	var1			DINT	0	FALSE
0002	pr			POINTER TO BYTE	0	FALSE
0003	r1			UINT	0	FALSE

编程语言：程序

梯形图（LD）



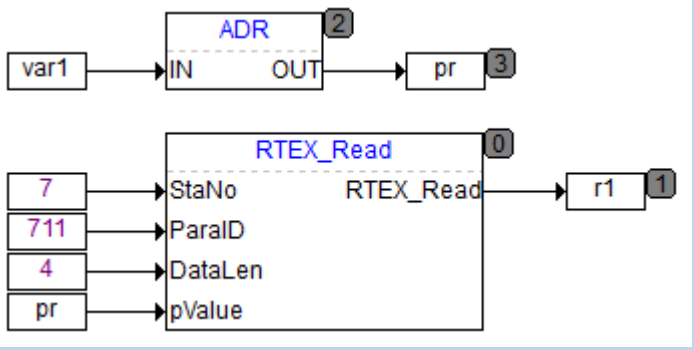
7 是站号
711 为 Pr7.11 参数（输入值的计算为 7*100+11）
var1 中为运行结果，比如该事例的运行结果为 500,000，即 Pr7.11 参数值为 500,000
运行结果：

0001	var1			DINT	500000	FALSE

结构化文本（ST）

```
0001 pr :=ADR ( var1);
0002 r1 := RTEX_Read (7 , 711, 4, pr);
```

连续功能图（CFC）



5.2 RTEX_Write（写驱动器参数）

- 功能说明

  通过外部函数库的形式给总线从站写参数。

- 输入参数

输入参数	类型	参数描述
StaNo	USINT	从站号
ParaID	UDINT	参数 ID
Value	DINT	参数值

- 传参说明

参数	值	参数描述
StaNo	0~31	从站号
ParaID	Class*100+ParNum	写 Para0~8 类参数信息
	0x1000	存储所有的参数到 EEPROM
	0x2000	将 C 类参数生效
Value	参数值	Class*100+ParNum：根据不同参数取值范围不同 ParaID 为 0x1000 或 0x2000 时，此参数无意义

- 返回值

返回值	类型	参数描述
0	UINT	成功
0x1000	UINT	读取的类别不存在
0x4000	UINT	没有对应的从站号
0x5000	UINT	系统没有进入运行状态
0x6000	UINT	正在执行复位
0x7000	UINT	协议栈异常
0x9000	UINT	超时
0x9100	UINT	缓冲区已满
其他值	UINT	参见附录 1 RTEX 指令错误码

- 备注：阻塞指令，非轴运动缓存指令。

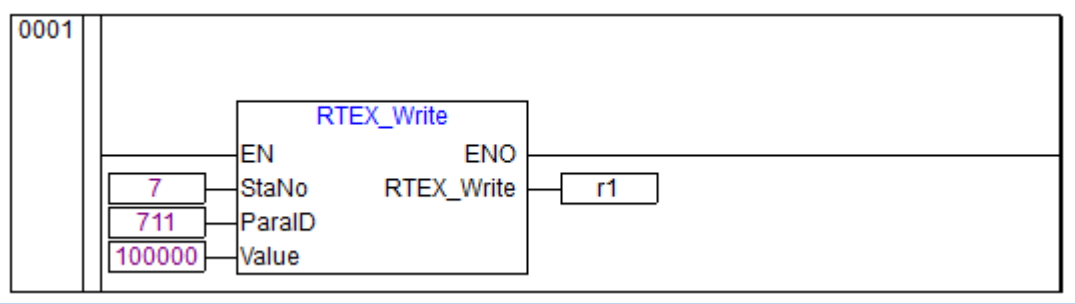
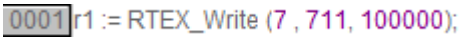
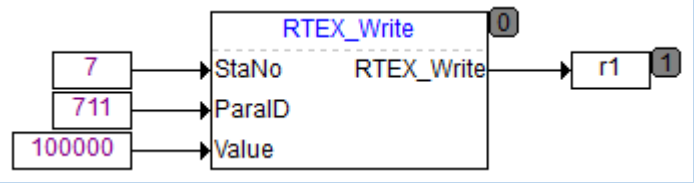
  C 类参数说明详见松下手册“SX-DSV02203_R2_2E”。

- 示例：修改 7 号站 Pr7.11 参数值为 100,000。

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	r1			UINT	0	FALSE

编程语言	程序

变量定义	
梯形图（LD）	 7 是站号 711 为 Pr7.11 参数（值的计算为 7*100+11） 运行结果为：将 Pr7.11 参数值写为 100,000
结构化文本（ST）	
连续功能图（CFC）	

5.3 RTEX_DriveErrorClear（清除驱动器错误）

- 功能说明

  通过外部库的方式清除告警信息。

- 输入参数

输入参数	类型	参数描述
StaNo	USINT	从站号
ParaID	UDINT	参数 ID

- 传参说明

参数	值	参数描述
StaNo	0~31	从站号
ParaID	0x001	清除当前报警
	0x011	清除历史报警
	0x021	清除外部报警

- 返回值

返回值	类型	参数描述
0	UINT	成功
0x1000	UINT	读取的类别不存在
0x4000	UINT	没有对应的从站号

返回值	类型	参数描述
0x5000	UINT	系统没有进入运行状态
0x6000	UINT	正在执行复位
0x7000	UINT	协议栈异常
0x9000	UINT	超时
0x9100	UINT	缓冲区已满
其他值	UINT	参见附录 1 RTEX 指令错误码

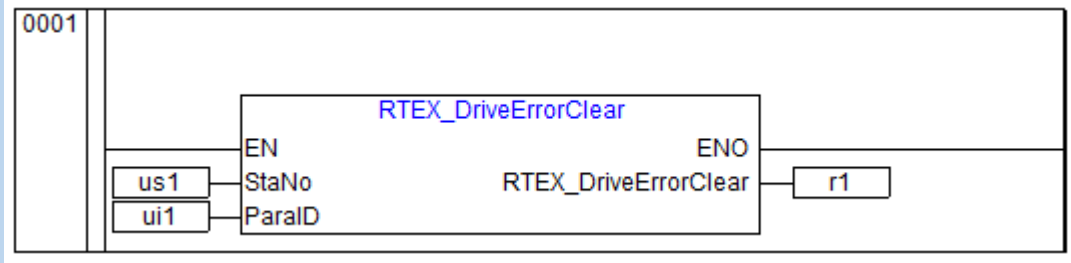
- 备注：阻塞指令，非轴运动缓存指令。

  清除指令，需要在电机处于 CP、CV 等运动控制模式下才有效。

- 示例：清除 7 号站当前报警。

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	us1			USINT	7	FALSE
0002	ui1			UDINT	1	FALSE
0003	r1			UINT	0	FALSE

编程语言	程序
梯形图（LD）	 运行结果为：将 7 号站的当前警报清除
结构化文本（ST）	0001 r1 := RTEX_DriveErrorClear (us1 , ui1);
连续功能图（CFC）	

5.4 RTEX_BusReset（复位协议主栈）

- 功能说明

  通过外部库的方式复位协议主站及其所有从站。为了避免驱动器复位时电机产生大的冲击，在使用此指令时，必须先清除所有轴缓存指令，使所有轴停止运动。

- 返回值

返回值	类型	参数描述
0	UINT	成功
0x5000	UINT	系统没有进入运行状态
0x6000	UINT	正在执行复位
0x7000	UINT	协议栈异常
0x9000	UINT	超时
0x9100	UINT	缓冲区已满
其他值	UINT	参见附录 1 RTEX 指令错误码

- 不支持的模块版本

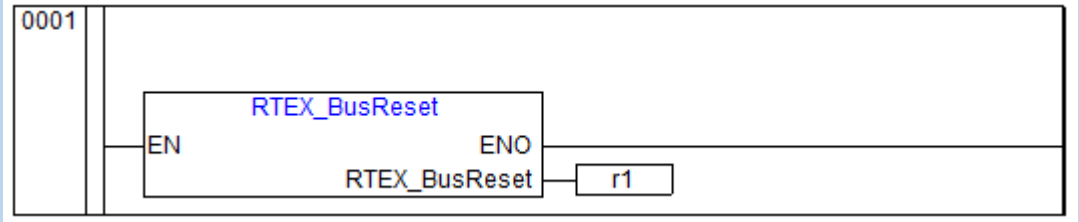
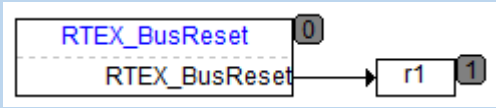
  MC1002R-A01

- 备注：由于总线断线或从站驱动器断电等原因导致协议栈异常时，可使用该指令对议栈主站进行复位。

- 示例：复位协议主站。

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	r1			UINT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	
连续功能图（CFC）	

5.5 RTEX_GetDriveState（获取驱动器状态）

- 功能说明

  通过外部库的方式获取从站状态标识。

- 输入参数

输入参数	类型	参数描述
StaNo	USINT	从站号
pFlagValue	POINTER TO UINT	读取结果存放地址

- 返回值

返回值	类型	参数描述
0	UINT	成功
0x3000	UINT	形参指针为空
0x4000	UINT	没有对应的从站号
0x5000	UINT	系统没有进入运行状态
0x6000	UINT	正在执行复位
0x7000	UINT	协议栈异常
0x9000	UINT	超时
0x10000	UINT	缓冲区已满
其他值	UINT	参见附录 1 RTEX 指令错误码

- 接收内容

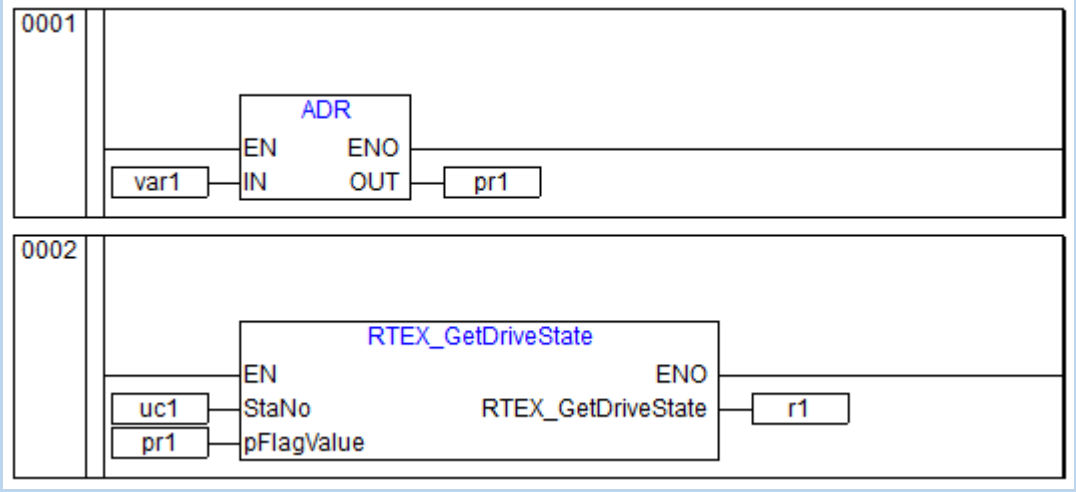
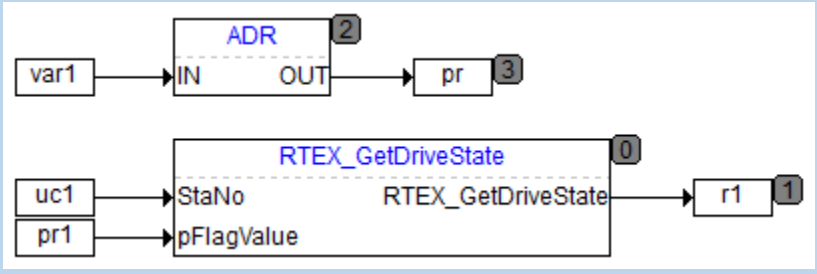
字节号	1	2
Bit 7	Servo_Active	E-STOP
Bit 6	Servo_Ready	SI-MON4/EX-SON
Bit 5	Alarm	SI-MON3/EXT3
Bit 4	Warning	SI-MON2/EXT2
Bit 3	Torque_Limited	SI-MON1/EXT1
Bit 2	Homing_Complete	HOME
Bit 1	In_Progress	POT/NOT
Bit 0	In_Position	NOT/POT

- 示例：获取 7 号站驱动器状态。

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	var1			UINT	960	FALSE
0002	pr1			POINTER TO UINT	207644652	FALSE
0003	uc1			BYTE	7	FALSE
0004	r1			UINT	0	FALSE

编程语言	程序

变量定义	
梯形图（LD）	 运行结果：r1 为 0 表示获取成功。 var1=960 转换为十六位二进制 0000 0011 1100 0000，各个位对应接收内容参见 RTEX_GetDriveState 的“接收内容”说明。
结构化文本（ST）	0001 pr1 := ADR (var1); 0002 r1 := RTEX_GetDriveState (uc1 , pr1);
连续功能图（CFC）	

5.6 RTEX_DriveEnable（伺服使能）

- 功能说明

  打开/关闭伺服使能分开关，可以独立控制每个从站轴的使能状态。

- 输入参数

输入参数	类型	参数描述
ucStackNo	BYTE	从站号
bMode	BOOL	使能值 1：使能 0：不使能（默认）

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他	UDINT	设置失败

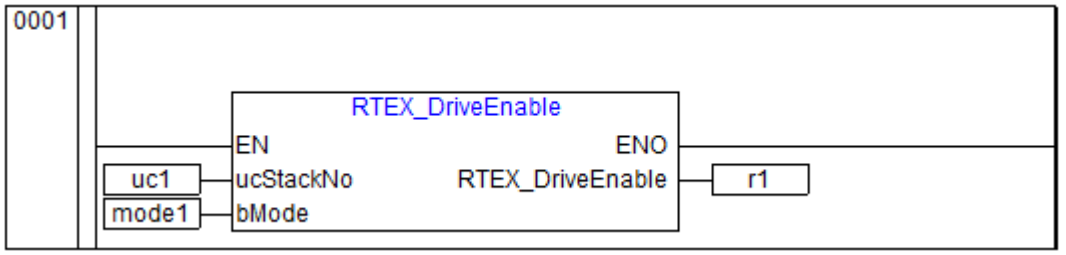
- 备注：阻塞指令，非轴运动缓存指令。只有伺服使能总开关开启的情况下，伺服使能分开关才能生效。使用 HMC_SetAxistype 设定总线轴类型后，需要延时一定时间（大约几毫秒）再使用 RTEX_DriveEnable 打开从站伺服使能，否则从站可能会报错。
- 示例：设定 7 号站伺服使能分开关为打开。

变量定义

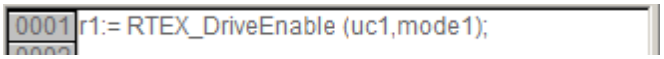
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	uc1			BYTE	7	FALSE
0002	mode1			BOOL	TRUE	FALSE
0003	r1			UDINT	0	FALSE

编程语言：程序

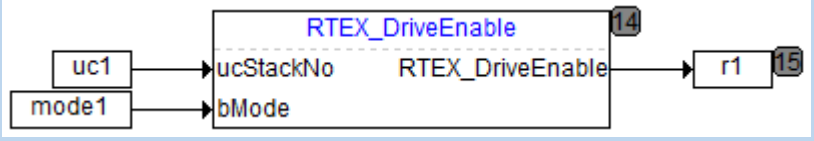
梯形图（LD）



结构化文本（ST）



连续功能图（CFC）



5.7 RTEX_GetDriveEnableState（获取伺服使能状态）

- 功能说明

  获取某个从站轴的伺服使能状态。

- 输入参数

输入参数	类型	参数描述
ucStackNo	BYTE	从站号

- 返回值

返回值	类型	参数描述
0	USINT	伺服使能关闭
1	USINT	伺服使能开启

- 备注：阻塞指令，非轴运动缓存指令。

  只有当伺服使能总开关（通过 HMC_DriveEnable 设定）与对应从站伺服使能分开关同时开启的情况下，从站的伺服使能状态才为开启状态，否则为关闭状态。

■ 示例：获取 7 号站伺服使能分开关状态。

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	uc1			BYTE	7	FALSE
0002	r1			USINT	0	FALSE

编程语言：程序

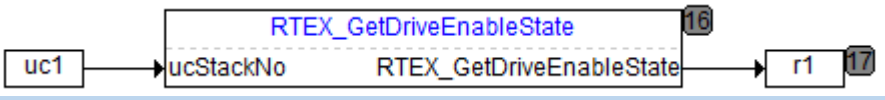
梯形图（LD）

r1 为伺服使能总开关&总线驱动器使能分开关。
运行结果：r1 为 1，即为伺服使能总开关和总线驱动器使能分开关均为 1。

结构化文本（ST）

```
0001 r1:= RTEX_GetDriveEnableState (uc1 );
```

连续功能图（CFC）



5.8 RTEX_DriveResetAll（复位所有驱动器）

■ 功能说明

通过外部库的方式复位所有从站驱动器。为了避免驱动器复位时电机产生大的冲击，在使用此指令时，必须先清除所有轴缓存指令，使所有轴停止运动。

■ 返回值

返回值	类型	参数描述
0	UINT	成功
0x5000	UINT	系统没有进入运行状态
0x6000	UINT	正在执行复位
0x7000	UINT	协议栈异常
0x9000	UINT	超时
0x9100	UINT	缓冲区已满
其他值	UINT	参见附录 1 RTEX 指令错误码

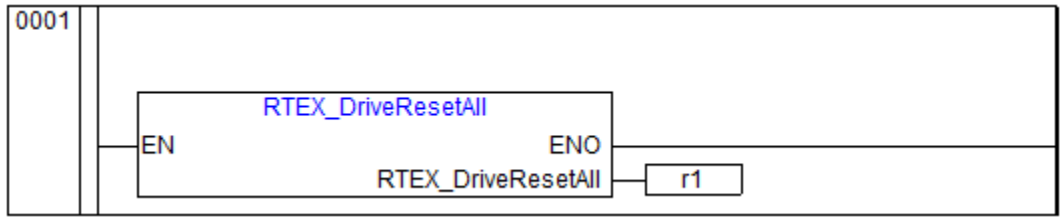
■ 备注：在进行参数配置，并执行写 EEPROM 之后，可以用此指令复位伺服器，达到参数生效的目的。也可以采用掉电，再上电的方式达到参数生效的目的。

■ 示例：复位驱动器。

变量定义

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	r1			UINT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	0001 r1 := RTEX_DriveResetAll ();
连续功能图（CFC）	

5.9 RTEX_ClearAbsEncoderData（清除绝对值编码器多圈数据）

- 函数原型

  UDINT RTEX_ClearAbsEncoderData(USINT ucAxisID)

- 功能说明

  对 RTEX 伺服配套的绝对值编码器多圈数据清零，执行完毕后需断电重启驱动器。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

5.10 RTEX 总线轴高速位置锁存

该功能可实现 RTEX 总线轴的高精度位置锁存，锁存由硬件完成，无软件延时偏差。锁存的触发源可选择驱动器的 latch 功能所对应的 EXT IO 引脚，也可选择由编码器的 Z 相零脉冲触发。

每个驱动器支持 2 路高速位置锁存通道。

5.10.1 RTEX_StartAixsPosLatch（启动 RTEX 轴位置锁存）

- 函数原型

UDINT RTEX_StartAixsPosLatch(USINT ucAxisID, USINT ucLatchNo, USINT ucTriggerSignal)

- 功能说明

配置并启动 RTEX 总线轴位置锁存功能，每个驱动器支持 2 路锁存通道。

注意：在配置完锁存功能后，应在 IEC 中循环调用 RTEX_CheckAixsPosLatchStatus 检查锁存状态，当检测到锁存完成时，立即调用 RTEX_GetLatchedAixsPos 读取锁存到的位置。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucLatchNo	USINT	锁存通道编号，1 或 2
ucTriggerSignal	USINT	锁存触发信号源： 0：Z 相 ■ 全闭环时外部位移传感器的 Z 相 ■ 串行通讯类型的绝对式外部位移传感器时返回指令错误（005Ah） 1：EXT1 的逻辑上升沿 2：EXT2 的逻辑上升沿 3：EXT3 的逻辑上升沿 4~8：禁止使用 ■ 选择时返回指令错误（0032h） 9：EXT1 的逻辑下升沿 10：EXT2 的逻辑下升沿 11：EXT3 的逻辑下升沿 12~15：禁止使用 ■ 选择时返回指令错误（0032h） 有关 EXT 信号的引脚定义，请参见松下 A6N 伺服驱动器手册

- 返回值

返回值	类型	参数描述
0	UDINT	配置成功
0xFFFFFFFF	UDINT	参数错误

- 指令类型

非轴运动缓存指令。

5.10.2 RTEX_CancelAixsPosLatch（撤销 RTEX 轴位置锁存）

- 函数原型

UDINT RTEX_CancelAixsPosLatch (USINT ucAxisID, USINT ucLatchNo)

- 功能说明

撤销指定的位置锁存通道。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucLatchNo	USINT	锁存通道编号，1 或 2

- 返回值

返回值	类型	参数描述
0	UDINT	配置成功
0xFFFFFFFF	UDINT	参数错误或锁存通道未启动

- 指令类型

非轴运动缓存指令。

5.10.3 RTEX_CheckAixsPosLatchStatus（获取位置锁存状态）

- 函数原型

UDINT RTEX_CheckAixsPosLatchStatus (USINT ucAxisID, USINT ucLatchNo)

- 功能说明

返回指定的总线轴高速位置锁存通道的状态。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucLatchNo	USINT	锁存通道编号，1 或 2

- 返回值

返回值	类型	参数描述
高速锁存通道状态	UDINT	0：等待锁存触发信号 1：锁存完成 其他：参数错误或锁存通道未启动

- 指令类型

非轴运动缓存指令。

5.10.4 RTEX_GetLatchedAixsPos（获取锁存位置）

- 函数原型

LREAL RTEX_GetLatchedAixsPos(USINT ucAxisID, USINT ucLatchNo)

- 功能说明

返回指定轴的高速锁存通道锁存的位置。

- 输入参数

输入参数	类型	参数描述

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucLatchNo	USINT	高速位置锁存通道号，1 或 2

- 返回值

返回值	类型	参数描述
锁存到的 Mpos 值（发生错误时返回 0）	LREAL	锁存到的 Mpos 值(发生错误时返回 0)

- 指令类型

  非轴运动缓存指令。

- 补充说明

  获取该值时请首先使用 RTEX_CheckAixsPosLatchStatus 指令检查锁存通道的状态，确认已锁存动作已完成。

5.11 RTEX_SetMotorActualPos（设定 RTEX 电机实际位置）

- 函数原型

  UDINT RTEX_SetMotorActualPos(USINT ucAxisID, DINT iPos)

- 功能说明

  设定 RTEX 电机当前实际位置（编码器位置）。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
iPos	DINT	设定位置

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

5.12 RTEX_Home（RTEX 伺服回零）

- 函数原型

  UDINT RTEX_Home(USINT ucAxisID, USINT ucHomeMode, USINT ucHomeDir)

- 功能说明

  调用 RTEX 伺服内部原点回归功能。原点回归的加减速、第一阶段查找速度、爬行速度对应轴参中相应参数。

该指令只能在轴类型为 RTEX_PP（43）时使用。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号
ucHomeMode	USINT	回归模式，详见松下 RTEX 伺服文档。以下为 A6N 支持的模式 1：DI+Z 相 2：DI 3：Z 相 4：正反限位+DI 6：正反限位+Z 相
ucHomeDir	USINT	查找方向，详见松下 RTEX 伺服文档。以下为 A6N 支持的模式： 0：正方向查找原点 1：负方向查找原点（ucHomeMode 为 2 时不支持）

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

非轴运动缓存指令。

- 补充说明

（1） 使用该指令前，首先设定轴类型为 43（RTEX_PP），开启伺服总使能（HMC_DriveEnable）和从站分使能（RTEX_DriveEnable）。轴类型为 43（RTEX_PP）时，不允许调用除 RTEX_Home/RTEX_ProfileSmoothStop/RTEX_ProfileHardStop 之外的运动控制指令；

（2） 该指令配置成功后即退出，原点回归过程中可通过 RTEX_GetHomeStatus 指令查看原点回归状态；

（3） 原点回归动作执行过程中，如果发生意外情况，可调用 RTEX_ProfileSmoothStop（减速停止）或 RTEX_ProfileHardStop（急停）终止原点回归操作。但这样会导致 DPOS 不等于 Mpos 的，所以请务必确保 RTEX_Home 最终执行成功后再切换为所需的轴类型；

（4） RTEX_Home 指令动作完成后，切换轴类型至需要的轴类型（如 40）；

（5） 最好把驱动器单位设定为指令单位/s（设定 Pr7.25=1），这样可以对原点回归速度进行更为精细的设定。

5.13 RTEX_GetHomeStatus（获取 RTEX 伺服回零状态）

- 函数原型

UDINT RTEX_ GetHomeStatus (USINT ucAxisID)

- 功能说明

通过该指令可获取 RTEX 总线原点回归状态。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	USINT	原点回归未完成
1	USINT	原点回归完成

- 指令类型

  非轴运动缓存指令。

5.14 RTEX_ ProfileSmoothStop（RTEX_PP 模式减速停止）

- 函数原型

  UDINT RTEX_ ProfileSmoothStop (USINT ucAxisID)

- 功能说明

  RTEX_PP 模式减速停止，主要用于终止 RTEX_Home 动作。

  RTEX 原点回归动作执行过程中，如果发生意外情况，可调用 RTEX_ProfileSmoothStop 终止原点回归操作。但这样会导致 DPOS 不等于 Mpos 的，所以请务必确保 RTEX_Home 最终执行成功后再切换为所需的轴类型。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

5.15 RTEX_ ProfileHardStop（RTEX_PP 模式急停）

- 函数原型

  UDINT RTEX_ ProfileHardStop (USINT ucAxisID)

- 功能说明

  RTEX_PP 模式急停，主要用于终止 RTEX_Home 动作。

RTEX 原点回归动作执行过程中，如果发生意外情况，可调用 RTEX_ProfileHardStop 终止原点回归操作。但这样会导致 DPOS 不等于 Mpos 的，所以请务必确保 RTEX_Home 最终执行成功后再切换为所需的轴类型。

- 输入参数

输入参数	类型	参数描述
ucAxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
0	UDINT	执行成功
0xFFFFFFFF	UDINT	执行失败

- 指令类型

  非轴运动缓存指令。

第6章　EtherCAT 指令库

本节指令专用于支持 EtherCAT 总线的控制器。

6.1 EtherCAT_BusReset（复位 EtherCAT 总线）

- 功能说明

通过外部库的方式复位 EtherCAT 总线，整个过程是先复位 comX 协议板卡、重新配置从站、初始化协议栈，最终达到整个 EtherCAT 总线复位的目的。

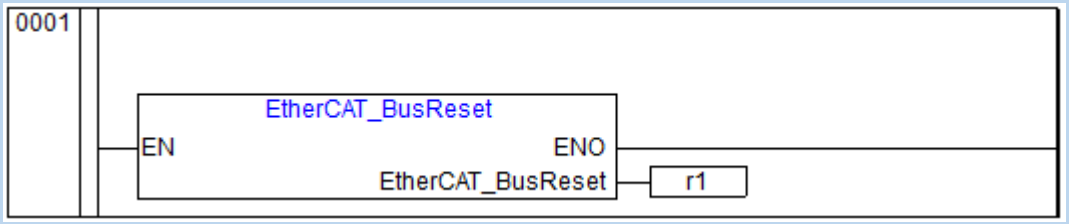
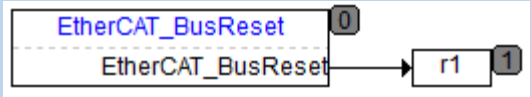
- 返回值

返回值	类型	参数描述
0	UDINT	复位 EtherCAT 总线成功
0x800A0001	UDINT	comX 通讯通道无效（内部错误，用户无需关注）
0x800C0011	UDINT	comX 通讯通道状态错误（内部错误，用户无需关注）
0x800A000A	UDINT	EtherCAT 配置文件无效（内部错误，用户无需关注）
0x800A0004	UDINT	EtherCAT 配置文件加载失败（内部错误，用户无需关注）
0x800A0019	UDINT	复位 comX 通讯通道失败
0x800C0020	UDINT	复位 comX 通讯通道超时

- 示例：复位 EtherCAT 总线。

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	r1			UDINT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	0001 r1 :=EtherCAT_BusReset();
连续功能图（CFC）	

6.2 EtherCAT_SlaveFaultReset（清除从站错误）

■ 功能说明

通过外部函数库的形式清除 EtherCAT 从站显示的错误码，整个过程是先通过 EtherCAT 协议消息清除从站显示的错误码，再对从站进行初始化，最终达到从站准备就绪，可以进入使能状态的目的。

■ 输入参数

输入参数	类型	参数描述
usSlaveAddress	UINT	从站号

■ 传参说明

参数	值	参数描述
usSlaveAddress	0x0001~0x37FF	从站号

■ 返回值

返回值	类型	参数描述
0	UDINT	清除从站错误码执行失败
1	UDINT	清除从站错误码执行成功

■ 备注

清除从站错误码执行成功表示命令执行成功，不代表从站错误被清除成功。部分从站错误无法清除，详见从站设备说明书。

清除从站错误码执行失败的原因可能是：从站号错误或者 EtherCAT 总线通信中断。

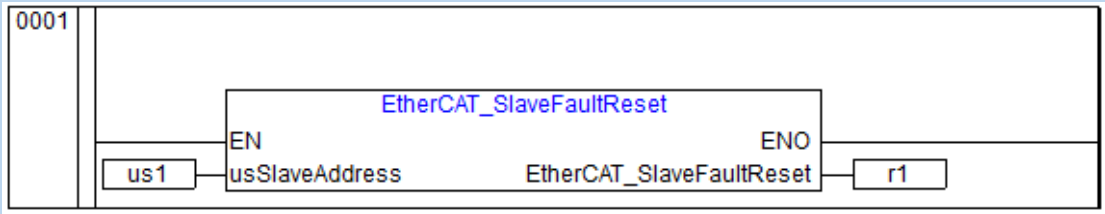
■ 示例：清除 16#100 号从站错误。

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	us1			UINT	16#100	FALSE
0002	r1			UDINT	0	FALSE

编程语言：程序

梯形图（LD）

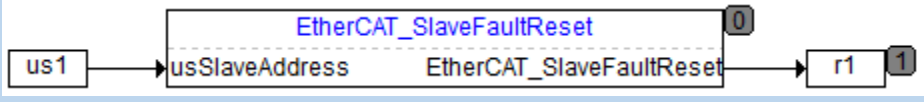


运行结果为：将 16#100 号站的错误清除

结构化文本（ST）

```
0001 r1 :=EtherCAT_SlaveFaultReset(us1);
```

连续功能图（CFC）



6.3 EtherCAT_DriveEnable（伺服使能）

- 功能说明

  打开/关闭伺服使能分开关，可以独立控制每个从站轴的使能状态。

- 输入参数

输入参数	类型	参数描述
usSlaveAddress	UINT	从站号（0x0001~0x37FF）
bMode	BOOL	使能值 1：使能 0：不使能（默认）

- 返回值

返回值	类型	参数描述
0	UDINT	设置成功
其他	UDINT	设置失败

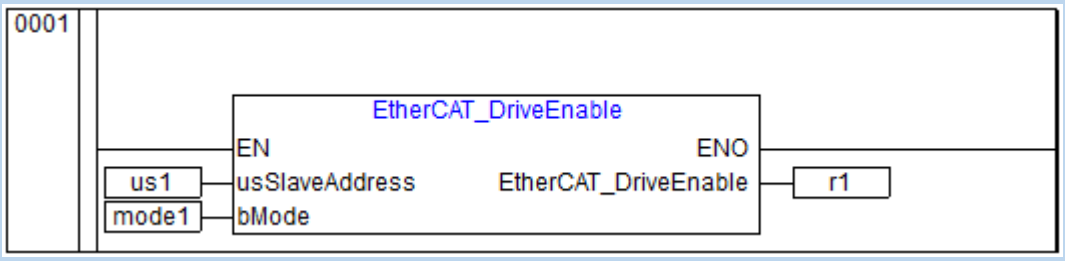
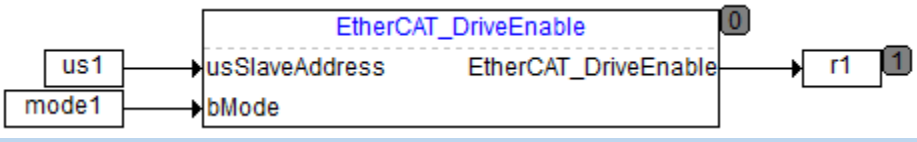
- 备注：阻塞指令，非轴运动缓存指令。

只有伺服使能总开关开启的情况下，伺服使能分开关才能生效。使用 HMC_SetAxistype 设定总线轴类型后，需要延时一定时间再使用 EtherCAT_DriveEnable 打开从站伺服使能，否则从站可能会报错。

- 示例：设定 16#100 号站伺服使能分开关为打开。

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	us1			UINT	16#100	FALSE
0002	mode1			BOOL	TRUE	FALSE
0003	r1			UDINT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	0001 r1 :=EtherCAT_DriveEnable(us1, mode1);
连续功能图（CFC）	

6.4 EtherCAT_GetDriveEnableState（获取伺服使能状态）

- 功能说明

  获取某个从站轴的伺服使能状态。

- 输入参数

输入参数	类型	参数描述
usSlaveAddress	UINT	从站号（0x0001~0x37FF）

- 返回值

返回值	类型	参数描述
0	USINT	伺服使能关闭
1	USINT	伺服使能开启

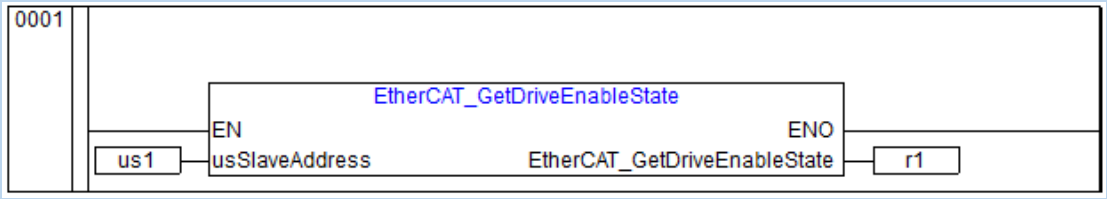
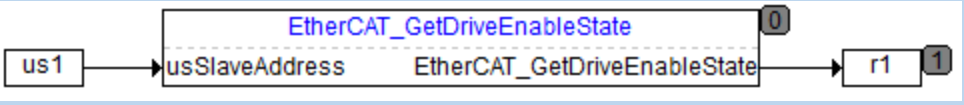
- 备注：阻塞指令，非轴运动缓存指令。

  只有当伺服使能总开关与对应从站伺服使能分开关同时开启的情况下，从站的伺服使能状态才为开启状态，否则为关闭状态。

- 示例：获取 16#100 号站驱动器状态。

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	us1			UINT	16#100	FALSE
0002	r1			UDINT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	0001 r1 :=EtherCAT_GetDriveEnableState(us1);
连续功能图（CFC）	

6.5 EtherCAT_Read（读从站参数）

- 功能说明

  读指定从站、指定参数的参数长度和参数内容。

- 输入参数

输入参数	类型	参数描述

输入参数	类型	参数描述
ulNodeId	UDINT	从站号
ulIndex	UDINT	参数对应的对象索引
ulSubIndex	UDINT	参数对应的对象子索引
pucDataCnt	POINTER TO BYTE	指向参数长度的指针
pucData	POINTER TO BYTE	指向参数内容的指针

- 返回值

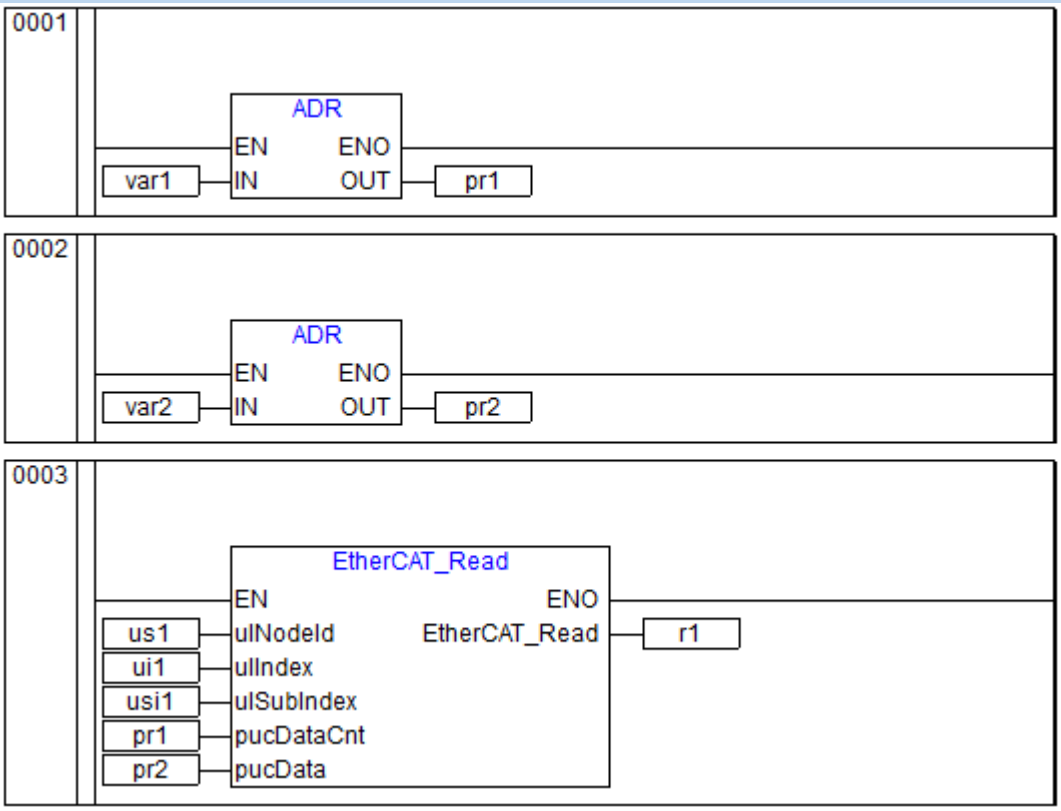
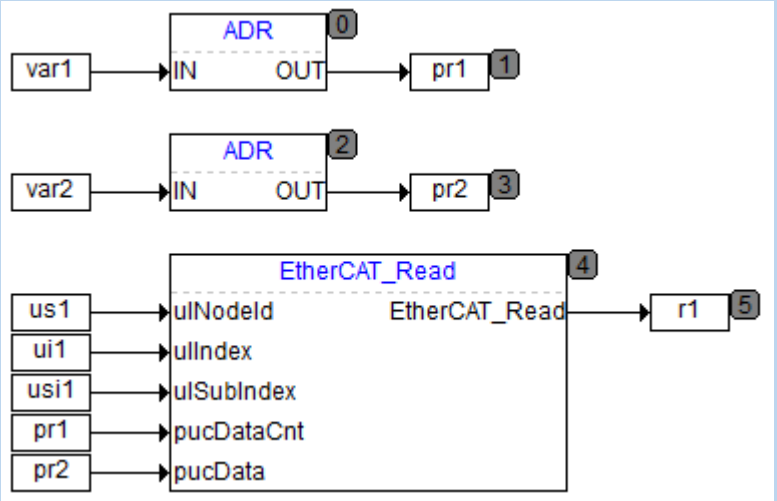
返回值	类型	参数描述
0	UDINT	读取成功
其他	UDINT	读取失败 详见附录 3 EtherCAT 读写参数错误码

- 备注：如果在 IEC 周期任务中调用 EtherCAT_Read 指令，则要求任务周期必须设置成大于任务执行的实际时间。
- 示例：读取 16#100 号站的设备名称（Omron 为例）。

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	var1			BYTE	0	FALSE
0002	pr1			POINTER TO BYTE	0	FALSE
0003	var2			STRING(160)	''	FALSE
0004	pr2			POINTER TO BYTE	0	FALSE
0005	us1			UDINT	16#100	FALSE
0006	ui1			UDINT	16#1008	FALSE
0007	usi1			UDINT	16#00	FALSE
0008	r1			UDINT	0	FALSE

编程语言	程序

变量定义	
梯形图（LD）	 16#100：从站号 16#1008：对象索引 16#00：对象子索引 运行结果： var1=20，表示参数长度占用 20 字节 var2=R88D-KN04H-ECT-L，表示设备名称
结构化文本（ST）	0001 pr1 :=ADR(var1); 0002 pr2 :=ADR(var2); 0003 r1 :=EtherCAT_Read(us1, ui1, usi1, pr1, pr2);
连续功能图（CFC）	

6.6 EtherCAT_Write（写从站参数）

- 功能说明

  写指定从站、指定参数的参数内容。

- 输入参数

输入参数	类型	参数描述
ulNodeId	UDINT	从站号
ulIndex	UDINT	参数对应的对象索引
ulSubIndex	UDINT	参数对应的对象子索引
ucDataCnt	BYTE	待写入参数的长度
pucData	POINTER TO BYTE	指向待写入参数内容的指针

- 返回值

返回值	类型	参数描述
0	UDINT	写入成功
其他	UDINT	写入失败 详见附录 3 EtherCAT 读写参数错误码

- 备注

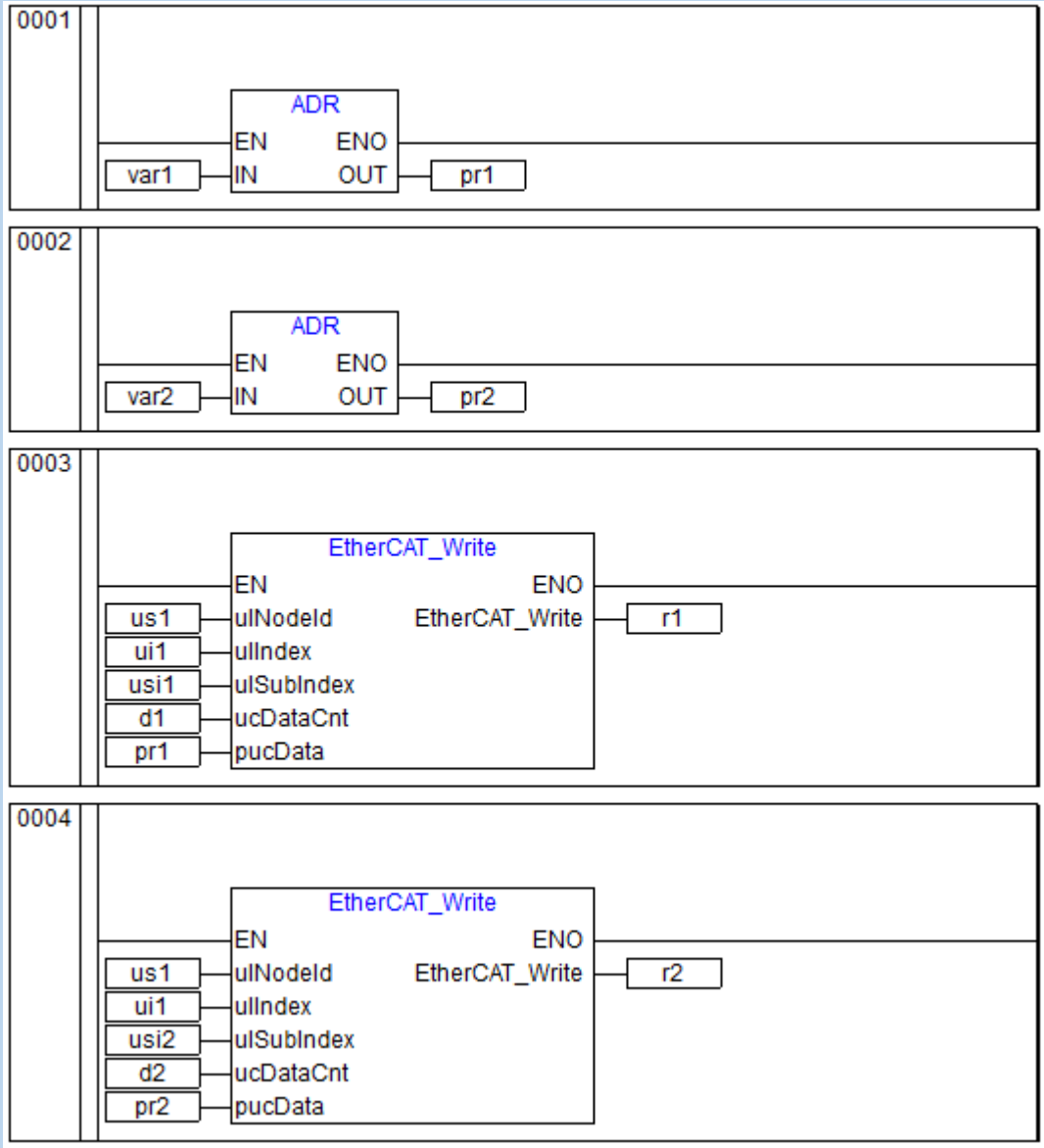
  待写入参数内容的实际长度与 ucDataCnt 的值一定要严格保持一致。

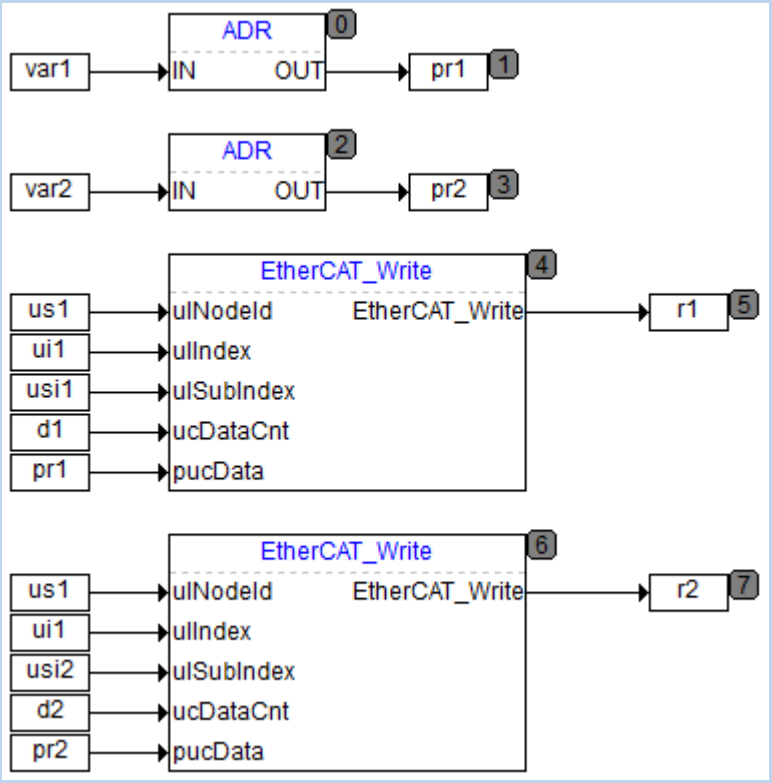
  如果在 IEC 周期任务中调用 EtherCAT_Write 指令，则要求任务周期必须设置成大于任务执行的实际时间。

- 示例：设置 16#100 号从站的齿轮比（Omron 为例）。

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	var1			DINT	1048576	FALSE
0002	pr1			POINTER TO BYTE	0	FALSE
0003	var2			DINT	10000	FALSE
0004	pr2			POINTER TO BYTE	0	FALSE
0005	us1			UDINT	16#100	FALSE
0006	ui1			UDINT	16#6091	FALSE
0007	usi1			UDINT	16#01	FALSE
0008	d1			BYTE	4	FALSE
0009	r1			UDINT	0	FALSE
0010	usi2			UDINT	16#02	FALSE
0011	d2			BYTE	4	FALSE
0012	r2			UDINT	0	FALSE

变量定义	
编程语言	程序
梯形图（LD）	 16#100：从站号 16#6091：对象索引 16#01：对象子索引 16#02：对象子索引 4：参数长度占用 4 字节 运行结果：设置从站 10000 个脉冲转一圈。
结构化文本（ST）	0001 pr1 :=ADR(var1); 0002 pr2 :=ADR(var2); 0003 r1 :=EtherCAT_Write(us1, ui1, usi1, d1, pr1); 0004 r2 :=EtherCAT_Write(us1, ui1, usi2, d2, pr2);

变量定义	
连续功能图（CFC）	

6.7 EtherCAT Touch Probe 功能

该功能可实现 EtherCAT 总线驱动器的高精度位置捕获，捕获由硬件完成，无软件延时偏差。捕获的触发源可选择驱动器的 touch probe 功能所对应的 IO 输入，也可选择由编码器的 Z 相 0 脉冲触发。

6.7.1 EtherCAT_CaptureConfig（Touch Probe 功能配置）

- 函数原型

  UDINT EtherCAT_CaptureConfig (USINT AxisID, UINT Mode)

- 功能说明

  配置 EtherCAT 总线伺服从站高速捕获功能。

  注意：在配置完 capture 功能后，最好在 IEC 中调用 EtherCAT_CaptureGetStatus 循环扫描捕获状态，当检测到捕获完成时，立即调用 EtherCAT_CaptureGetMpos 读取所需的捕获位置。

- 输入参数

输入参数	类型	参数描述
AxisID	USINT	轴号
Mode	UINT	捕获模式，见下表

伺服 Mode 参数配置如下表（具体参见所用伺服驱动器 touch probe 功能的相关对象，index: 0x60B8）：

捕获通道	Bit	参数描述
Touch probe1	0	0：关闭捕获通道 1 1：开启捕获通道 1
	1	0：单次触发 1：多次触发
	2	0：通道 1 的 input 信号触发（touch probe1 input） 1：Z 相 0 脉冲触发
	3	保留
	4	0：关闭上升沿捕获 1：开启上升沿捕获
	5	0：关闭下降沿捕获 1：开启下降沿捕获
	6	保留
	7	保留

捕获通道	Bit	参数描述
Touch probe2	8	0：关闭捕获通道 2 1：开启捕获通道 2
	9	0：单次触发 1：多次触发
	10	0：通道 2 的 input 信号触发（touch probe2 input） 1：Z 相 0 脉冲触发
	11	保留
	12	0：关闭上升沿捕获 1：开启上升沿捕获
	13	0：关闭下降沿捕获 1：开启下降沿捕获
	14	保留
	15	保留

- 返回值

返回值	类型	参数描述
0	UDINT	配置成功
其它	UDINT	错误码

- 指令类型

  非轴运动缓存指令。

- 应用示例

  当轴 0 对应的伺服驱动器的 touch probe1 针脚输入上升沿时，捕获该轴的位置值。

  程序如下：

序号	变量名	变量说明	变量类型	初始值	掉电保护
0001	AxisID	待捕获轴ID	USINT	0	FALSE
0002	CaptureMode	capture模式	UINT	0	FALSE
0003	CaptureFinishStatus	capture完成时状态	WORD	0	FALSE
0004	CaptureChannel	capture通道。1：touch probe1 ； 2：touch probe2	USINT	0	FALSE
0005	CapturePosID	capture位置ID。1：上升沿所捕获的位置 ； 2：下降沿所捕获的位置	USINT	0	FALSE
0006	Err	错误码	UDINT	0	FALSE
0007	CapturePos	捕获的位置	LREAL	0	FALSE

```
0001
0002 AxisID := 0;(*待捕获轴*)
0003 CaptureMode := 16#1; (*capture模式配置：使用通道1，单次触发，触发信号来自touch probe1 输入，仅捕获上升沿。*)
0004 CaptureFinishStatus := 16#3; (*capture完成时状态。*)
0005 Err := EtherCAT_CaptureConfig (AxisID ,CaptureMode); (*配置capture：配置成功时返回0*)
0006 IF 0 = Err THEN (*配置成功*)
0007    WHILE TRUE DO
0008       IF CaptureFinishStatus = ( UINT_TO_WORD (EtherCAT_CaptureGetState (AxisID)) AND CaptureFinishStatus ) THEN (*查询capture是否完成*)
0009          CaptureChannel := 1; (*capture通道。1：touch probe1 ； 2：touch probe2*)
0010          CapturePosID := 1; (*capture位置ID。1：上升沿所捕获的位置 ； 2：下降沿所捕获的位置*)
0011          CapturePos := EtherCAT_CaptureGetMpos(AxisID, CaptureChannel, CapturePosID); (*获取捕获到的位置*)
0012          EXIT; (*获取到捕获位置后退出循环*)
0013       END_IF
0014    END_WHILE
0015 END_IF
```

图 217 程序示例

6.7.2 EtherCAT_CaptureGetStatus（获取 Touch Probe 状态）

- 函数原型

  UINT EtherCAT_CaptureGetStatus(USINT AxisID)

- 功能说明

  返回指定从站高速捕获功能的状态。

- 输入参数

输入参数	类型	参数描述
AxisID	USINT	轴号

- 返回值

返回值	类型	参数描述
高速捕获功能状态 （指令调用失败时返回 0）	UINT	高速捕获功能状态，见下表

返回值状态含义如下表（具体参见所用伺服驱动器 touch probe 功能的相关对象，index: 0x60B9）。

捕获通道	Bit	参数描述
Touch probe1	0	0：捕获通道 1 关闭 1：获通道 1 开启捕
	1	0：捕获通道 1 上升沿捕获未完成 1：捕获通道 1 上升沿捕获完成
	2	0：捕获通道 1 下降沿捕获未完成 1：捕获通道 1 下降沿捕获完成
	3	保留
	4	保留
	5	保留

捕获通道	Bit	参数描述
	6	测试用
	7	测试用

捕获通道	Bit	参数描述
Touch probe2	8	0：捕获通道 2 关闭 1：获通道 2 开启
	9	0：捕获通道 2 上升沿捕获未完成 1：捕获通道 2 上升沿捕获完成
	10	0：捕获通道 2 下降沿捕获未完成 1：捕获通道 2 下降沿捕获完成
	11	保留
	12	保留
	13	保留
	14	测试用
	15	测试用

- 指令类型

  非轴运动缓存指令。

6.7.3 EtherCAT_CaptureGetMpos（获取 Touch Probe 捕获位置）

- 函数原型

  LREAL EtherCAT_CaptureGetMpos(USINT AxisID, USINT Channel ,USINT PosID)

- 功能说明

  返回指定从站高速捕获通道获取到的 Mpos。

- 输入参数

输入参数	类型	参数描述
AxisID	USINT	轴号
Channel	USINT	通道号 1：通道 1（touch probe1） 2：通道 2（touch probe2）
PosID	USINT	欲获取的位置 ID 1：上升沿 2：下降沿（某些驱动器不支持下降沿捕获，如松下 A5B）

- 返回值

返回值	类型	参数描述
捕获到的 Mpos 值（指令调用失败时返回 0）	LREAL	捕获到的 Mpos 值

- 指令类型

  非轴运动缓存指令。

- 补充说明

  获取该值时请首先使用 EtherCAT_CaptureGetStatus 指令检查捕获状态，确认要获取的值已经捕获完成。

第7章　CANopen 协议相关指令库

7.1 CANopenReadDict（获取 CANopen 从节点对象字典参数）

- 功能说明

  读取 CANopen 远程节点对象字典。

- 输入参数

输入项	数据类型	含义	初始值
EN_R	BOOL	上升沿触发	0
NodeID	BYTE	远程节点：1~126	0
Index	WORD	参数索引：0x1000~0x9FFF	0
SubIndex	BYTE	参数子索引	0

- 输出参数

输出项	数据类型	含义	初始值
Done	BOOL	读完成 0：未完成 1：成功完成	0
Busy	BOOL	读操作进行中 0：未开始或已结束 1：进行中	0
Error	BOOL	错误标志 0：无错误 1：有错误	0
ErrorID	DWORD	错误码	0
ReadLen	WORD	读取的对象字典参数大小 1：读取的对象字典参数大小为 1 字节 2：读取的对象字典参数大小为 2 字节 4：读取的对象字典参数大小为 4 字节	0
ReadData	DWORD	读取的对象字典参数值	0

7.2 CANopenWriteDict（更新 CANopen 从节点对象字典参数）

- 功能说明

  写 CANopen 远程节点对象字典。

- 输入参数

输入项	数据类型	含义	初始值
EN_R	BOOL	上升沿触发	0

输入项	数据类型	含义	初始值
NodeID	BYTE	远程节点：1~126	0
Index	WORD	参数索引：0x1000~0x9FFF	0
SubIndex	BYTE	参数子索引	0
DataLen	WORD	写入的对象字典值参数大小： 1：写入的对象字典参数大小为 1 字节 2：写入的对象字典参数大小为 2 字节 4：写入的对象字典参数大小为 4 字节	0
WriteData	DWORD	写入的对象字典参数值	0

- 输出参数

输出项	数据类型	含义	初始值
Done	BOOL	写完成 0：未完成 1：成功完成	0
Busy	BOOL	写操作进行中 0：未开始或已结束 1：进行中	0
Error	BOOL	错误标志 0：无错误 1：有错误	0
ErrorID	DWORD	错误码	0

7.3 CANopenNodeNMT（发送 NMT 命令）

- 功能说明

向 CANopen 远程节点设备发送 NMT 命令。

NMT 指令发送完成，仅表示 CANopen 主站已向指定的节点发送 NMT 命令，不代表该点已经按照命令字完成操作。

- 输入参数

输入项	数据类型	含义	初始值
EN_R	BOOL	上升沿触发	0
NodeID	BYTE	远程节点： 0：广播所有连接从站 1~126:指定对应从站	0
Command	BYTE	NMT 命令字设定值： 0x01：启动远程节点 0x02：停止远程节点 0x80：节点进入预运行状态 0x81：应用复位 0x82：通信复位	0

- 输出参数

输出项	数据类型	含义	初始值
Done	BOOL	命令已发送完成 0：未完成 1：完成	0
Busy	BOOL	正在发送中 0：未开始或已结束 1：进行中	0
Error	BOOL	错误标志 0：无错误 1：有错误	0
ErrorID	DWORD	错误码	0

7.4 CANopenResetNodeDiag（清除从节点诊断信息）

- 功能说明

  清除指定节点的诊断信息。

- 输入参数

输入项	数据类型	含义	初始值
EN_R	BOOL	上升沿触发	0
NodeID	BYTE	远程节点：1~126	0

- 输出参数

输出项	数据类型	含义	初始值
Done	BOOL	清除完成 0：未完成 1：完成	0
Busy	BOOL	正在清除中 0：未开始或已结束 1：进行中	0
Error	BOOL	错误标志 0：无错误 1：有错误	0
ErrorID	DWORD	错误码	0

第8章　I/O 操作指令库

8.1 SetInvertDI（设置 DI 反转标志）

- 功能：DI 状态掩码，32 个 bit 位与本体 DI 的输入通道对应。uDIMask 的某个指定 bit 位与所对应的 DI 通道位的异或结果作为上位机、LED 屏的显示结果。DI 状态掩码只对本体物理 DI 有效，对扩展 DI 以及虚拟 DI 掩码无效。
- 输入参数

输入参数	类型	参数描述
uDIMask	UDINT	0：所显示的 DI 状态与实际输入的电平状态一致 1：所显示的 DI 状态与实际输入的电平状态相反

- 返回值

返回值	类型	参数描述
0xFFFFFFFF	UDINT	函数参数错误

- 不支持的模块版本

  MC1008-A01

- 备注：阻塞指令，非轴运动缓存指令。

8.2 GetInvertDI（获取 DI 反转标志）

- 功能：获取 DI 状态掩码值。
- 返回值

返回值	类型	参数描述
DI 的状态掩码值	UDINT	成功
0xFFFFFFFF	UDINT	函数参数错误

- 不支持的模块版本

  MC1008-A01

- 备注：阻塞指令，非轴运动缓存指令。

第9章　系统库

9.1 通讯指令

9.1.1 GENERATE_CRC（CRC 校验计算）

- 功能：产生 16 位的 CRC 校验值。
- 输入参数

输入参数	类型	参数描述
TBL	WORD	需要校验的数据在 M 区存放的偏移 若设置为 200，则需要校验的数据保存在%MB200 开始的地址中 （默认值 0）
byteCounter	WORD	需要校验的数据长度（byte） 0 为非法值（默认值）

- 输出参数

输出参数	类型	参数描述
CRC_Code	WORD	校验结果 16 位 （默认值 0）
FINISH	BOOL	是否完成校验 FALSE：未完成校验（默认值） TRUE：完成校验
Error	BYTE	错误信息 0：正确（默认值） 1：输入参数 byteCounter 为 0 2：输入参数 TBL 超出 M 区范围

9.1.2 ComConfig（设置串口通讯参数）

- 功能：该功能目前仅支持在线配置各通道通讯参数。
- 输入参数

输入参数	类型	参数描述
EN_R	BOOL	上升沿使能 FALSE：无效（默认值） TRUE：上升沿有效
Port	USINT	端口号 0：COM1，支持 RS232（默认值） 1：COM2，支持 RS422/485 2：COM3，支持 RS485 3：COM4，支持 RS485
ComType	USINT	通信类型

输入参数	类型	参数描述
		0：RS485（默认值） 1：RS422 2：RS232
Protocol	USINT	协议类型 0：自由口 1：Modbus 主 2：Modbus 从
Parity	USINT	校验方式 0：无校验（默认值） 1：偶校验 2：奇校验
DataWidth	USINT	数据宽度 8：数据宽度 8 位（默认值）
Stop	USINT	停止位 1：停止位为 1（默认值） 2：停止位为 2
BaudRate	UDINT	波特率（bps） 9600：波特率 9600 19200：波特率 19,200 38400：波特率 38,400（默认值） 57600：波特率 57,600 115200：波特率 115,200 以下波特率 COM1 口不支持，但其余 COM 口支持。 500K：波特率 500,000 1M：波特率 1,000,000 1.125M：波特率 1,125,000 1.25M：波特率 1,250,000 1.5M：波特率 1,500,000

- Protocol 参数要求：（1）软件方面。仅在 AutoThink V3.1.4B2 之前版本中支持。（2）固件方面。仅限于 MC1008-A01、MC1008-A02、MC1008-B01、MC1004-A01。

■ 输出参数

输出参数	类型	参数描述
Q	BOOL	输出端 TRUE：本次任务周期该条指令执行完毕（默认值 FALSE）
Err	USINT	错误码 0：设置成功或未进行设置操作（默认值） 1：ComType 错误 2：Port 错误 3：Protocol 错误 4：Parity 错误 5：DataWidth 错误 6：Stop 错误 7：BaudRate 错误

9.1.3 ComRecv（接收串口数据）

- 功能：串口接收数据。
- 输入参数

输入参数	类型	参数描述
EN_R	BOOL	高电平有效 （默认值 FALSE）
DataLenth	UDINT	期望接收的数据长度（byte） （默认值 0）
DataOffset	UDINT	数据存放地址所在 M 区偏移（byte） （默认值 0）
Port	USINT	端口号 0：COM1，支持 RS232（默认值） 1：COM2，支持 RS422/485 2：COM3，支持 RS485 3：COM4，支持 RS485

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出 TRUE：本次任务周期该条指令执行完毕 （默认值 FALSE）
Error	USINT	错误 0：无错误（默认值） 1：输入参数配置错误 2：本端口没有配置为自由口协议
RecvDataBytes	UDINT	实际接收的字节数（byte） （默认值 0）

9.1.4 ComSend（发送串口数据）

- 功能：串口发送数据。
- 输入参数

输入参数	类型	参数描述
EN_R	BOOL	上升沿使能 （默认值 FALSE）
DataLenth	UDINT	期望发送的数据长度（byte） 最大支持 255 bytes （默认值 0）
DataOffset	UDINT	数据存放地址所在 M 区的偏移（byte） （默认值 0）
Port	USINT	端口号 0：COM1，支持 232（默认值） 1：COM2，支持 422/485 2：COM3，支持 485 3：COM4，支持 485

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出 TRUE：本次任务周期该条指令执行完毕 （默认值 FALSE）
Error	USINT	错误 0：无错误（默认值） 1：输入参数配置错误 2：本端口没有配置为自由口协议
SendDataBytes	UDINT	实际发送字节数（byte） （默认值 0）

9.1.5 ComClearRecvFIFO（清空串口接收数据）

- 功能：清空串口接收数据。
- 输入参数

输入参数	类型	参数描述
En_R	BOOL	上升沿使能 （默认值 FALSE）
Port	USINT	端口号 0：COM1，支持 232（默认值） 1：COM2，支持 422/485 2：COM3，支持 485 3：COM4，支持 485

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出 TRUE：本次任务周期该条指令执行完毕 （默认值 FALSE）
Err	USINT	错误 0：无错误（默认值） 1：错误

9.1.6 CANRecv（接收 CAN 数据）

- 功能：单次读取 CAN 数据，如有滤波设置，则读到的是滤波后的数据。
- 输入参数

输入参数	数据类型	参数描述
IN	BOOL	高电平使能
Port	BYTE	端口号 0：CAN0

- 输出参数

输出参数	数据类型	参数描述

输出参数	数据类型	参数描述
Done	BOOL	TRUE：接收到数据且没有错误 FALSE：未接收到数据
Busy	BOOL	TRUE：正在接收 FALSE：空闲状态
Error	BOOL	TRUE：处理过程有错误产生，通过 ErrorID 项查看对应的错误码 FALSE：无错误
ErrorID	WORD	0：无错误 1：端口号错误 2：功能块未配为 CAN 自由口协议 3：接收到的数据长度大于 8 字节 21：硬件接收 FIFO 已满，此时认为是网络风暴，控制器会自动复位 CAN 硬件，并恢复用户正在使用的参数设置 Bit7 为 1 表示 CAN 总线发生错误，此时 Bit0～Bit4 共计 5 个 bit 位分别表示如下五种错误： Bit0：CRC 错误（即 CRC ERR） Bit1：形式错误（即 Form ERR） Bit2：填充错误（即 Stuff ERR） Bit3：位错误（即 Bit ERR） Bit4：应答错误（即 ACK ERR）
EXT	BOOL	TRUE：扩展帧 FALSE：标准帧
RTR	BOOL	TRUE：远程帧 FALSE：数据帧
MsgID	DWORD	消息 ID 标准帧：低 11 位有效 扩展帧：低 29 位有效
DataLen	BYTE	接收到的数据长度
Byte1	BYTE	第 1 字节数据
Byte2	BYTE	第 2 字节数据
Byte3	BYTE	第 3 字节数据
Byte4	BYTE	第 4 字节数据
Byte5	BYTE	第 5 字节数据
Byte6	BYTE	第 6 字节数据
Byte7	BYTE	第 7 字节数据
Byte8	BYTE	第 8 字节数据

9.1.7 CANSend（发送 CAN 数据）

- 功能：单次发送数据。
- 输入参数

输入参数	数据类型	参数描述
EN_R	BOOL	上升沿使能

输入参数	数据类型	参数描述
Port	BYTE	端口号 0：CAN0
EXT	BOOL	TRUE：扩展帧 FALSE：标准帧
RTR	BOOL	TRUE：远程帧 FALSE：数据帧
MsgID	DWORD	消息 ID 标准帧：低 11 位有效 扩展帧：低 29 位有效
DataLen	BYTE	发送的数据长度，0~8
Byte1	BYTE	第 1 字节数据
Byte2	BYTE	第 2 字节数据
Byte3	BYTE	第 3 字节数据
Byte4	BYTE	第 4 字节数据
Byte5	BYTE	第 5 字节数据
Byte6	BYTE	第 6 字节数据
Byte7	BYTE	第 7 字节数据
Byte8	BYTE	第 8 字节数据

- 输出参数

输出参数	数据类型	参数描述
Done	BOOL	TRUE：发送成功
Busy	BOOL	TRUE：正在发送 FALSE：空闲状态
Error	BOOL	TRUE：处理过程有错误产生，通过 ErrorID 项查看对应的错误码 FALSE：无错误
ErrorID	WORD	错误码： 0：发送成功 1：端口号错误 2：功能块未配为 CAN 自由口协议 3：发送的数据长度大于 8 字节

9.1.8 CANConfig（设置 CAN 参数）

- 功能：设置波特率。
- 输入参数

输入参数	数据类型	参数描述
EN_R	BOOL	上升沿使能
Port	BYTE	端口号 0：CAN0
BaudRate	DWORD	波特率：1000、800、500、250、125、100、50，单位：kbps

- 输出参数

输出参数	数据类型	参数描述
Done	BOOL	TRUE：设置成功
Busy	BOOL	TRUE：正在设置 FALSE：空闲状态
Error	BOOL	TRUE：处理过程有错误产生，通过 ErrorID 项查看对应的错误码 FALSE：无错误
ErrorID	WORD	错误码： 0：设置成功 1：端口号错误 2：设置失败 3：不支持的波特率参数

9.1.9 CANFilterEnable（使能 CAN 滤波参数）

- 功能：使能某一组硬件滤波参数，最大支持 4 组。
- 输入参数

输入参数	数据类型	参数描述
EN_R	BOOL	上升沿使能
Port	BYTE	端口号 0：CAN0
Group	BYTE	参数组序号，最大 4 组，0~3
MaskEXT	BOOL	是否使能 EXT 项，默认 TRUE
EXT	BOOL	TRUE：扩展帧；FALSE：标准帧
MaskRTR	BOOL	是否使能 RTR 项，默认 TRUE
RTR	BOOL	TRUE：远程帧；FALSE：数据帧
MaskMsgID	BOOL	是否使能 MSGID 项，默认 TRUE
MsgID	DWORD	消息 ID 标准帧：低 11 位有效 扩展帧：低 29 位有效 如果 EXT_MASK 项未使能，则 MSGID 小于 2048 按标准帧格式处理，大于等于 2048 按扩展帧格式处理

- 输出参数

输出参数	数据类型	参数描述
Done	BOOL	TRUE：使能成功
Busy	BOOL	TRUE：正在设置 FALSE：空闲状态
Error	BOOL	TRUE：处理过程有错误产生，通过 ErrorID 项查看对应的错误码 FALSE：无错误
ErrorID	WORD	错误码 0：设置成功 1：端口号错误

输出参数	数据类型	参数描述
		2：设置失败 3：不支持的组序号

9.1.10 CANFilterDisable（不使能 CAN 滤波参数）

- 功能：不使能某一组滤波参数，最大支持 4 组。
- 输入参数

输入参数	数据类型	参数描述
EN_R	BOOL	上升沿使能
Port	BYTE	端口号 0：CAN0
Group	BYTE	参数组序号，最大 4 组，0~3

- 输出参数

输出参数	数据类型	参数描述
Done	BOOL	TRUE：设置成功
Busy	BOOL	TRUE：正在设置 FALSE：空闲状态
Error	BOOL	TRUE：处理过程有错误产生，通过 ErrorID 项查看对应的错误码 FALSE：无错误
ErrorID	WORD	错误码 0：成功 1：端口号错误 2：设置失败 3：不支持的组序号

9.1.11 CANClrRecvFIFO（清空 CAN 接收 FIFO）

- 功能：清空接收 FIFO。
- 输入参数

输入参数	数据类型	参数描述
EN_R	BOOL	上升沿使能
Port	BYTE	端口号 0：CAN0

- 输出参数

输出参数	数据类型	参数描述
Done	BOOL	TRUE：清空成功
Busy	BOOL	TRUE：正在清空 FALSE：空闲状态
Error	BOOL	TRUE：处理过程有错误产生，通过 ErrorID 项查看对应的错误码

输出参数	数据类型	参数描述
		FALSE：无错误
ErrorID	WORD	错误码 0：成功 1：端口号错误 2：清空失败

9.1.12 TCP_CONNECT（建立 TCP 通信连接）

- 功能：建立 TCP 协议通信连接。
- 输入参数

输入参数	数据类型	参数描述	初始值
EN_R	BOOL	上升沿：CPU 启动连接操作。当 EN_R 为 FALSE 时，输出显示连接的当前状态	0
ConnectID	BYTE	Socket 连接编号 1~5	0
Active	BOOL	TRUE：创建主动客户端连接 FALSE：创建被动服务器连接	0
IPaddr1	BYTE	四字节 IP 地址，由高到低依次为 IPaddr1~IPaddr4	0
IPaddr2	BYTE		0
IPaddr3	BYTE		0
IPaddr4	BYTE		0
RemotePort	WORD	远程设备上的端口号。远程端口号范围为 1 到 49151。对于被动连接，使用零	0
LocalPort	WORD	本地设备上的端口号。本地端口号范围为 1 到 49151，但存在一些限制，具体请详见附录 5 以太网自由协议库指令共用参数中的 LocalPort 参数	0

- 输出参数

输出参数	数据类型	参数描述	初始值
Done	BOOL	如果连接处于激活状态并准备发送或接收	0
Busy	BOOL	如果连接过程仍在进行中	0
Error	BOOL	如果连接不可用。ErrorID 显示其中一种错误代码，用于指示存在的问题	0
ErrorID	WORD	如果指令置位 Error 输出，ErrorID 输出会显示错误代码 如果指令置位 Busy 或 Done 输出，ErrorID 为零（无错误） 有关详细信息，请参见附录 6 以太网自由协议通信指令错误码	0
LinkIPaddr1	BYTE	已连接的远程设备 IP 地址的四个八位字节。	默认隐藏

输出参数	数据类型	参数描述	初始值
LinkIPaddr2	BYTE	LinkIPaddr1 是 IP 地址的最高有效字节，LinkIPaddr4 是 IP 地址的最低有效字节	
LinkIPaddr3	BYTE		
LinkIPaddr4	BYTE		
LinkPort	WORD	已连接远程设备端口号	

连接操作是异步的，可能需要几次扫描才能完成。连接操作待决时，Busy 输出值为 TRUE。当 CPU 完成操作时，指令置位 Done 或 Error 输出。如果发生错误，则 ErrorID 输出会显示错误代码。

指令处于繁忙状态时不得更改 TCP_CONNECT 的输入参数。CPU 需要凭借这一点了解这是启动连接过程的调用的延续。

您将连接 ID(ConnectID)输入分配给连接，然后当发送、接收或断开连接时使用此 ConnectID 引用该连接。

Active 输入位确定这是主动连接（Active 设置为 TRUE）还是被动连接（Active 设置为 FALSE）。

当 Active 输入设置为 TRUE 时，CPU 会创建主动连接（客户端），则 CPU 尝试联系并创建到指定 IP 地址和远程端口号(RemotePort)的连接。CPU 打开本地端口(LocalPort)以从远程设备接收消息。

当 Active 输入设置为 FALSE 时，CPU 会创建被动（服务器）连接。在这种情况下，CPU 打开请求的本地端口(LocalPort)并接受来自远程设备的连接请求。如果要接受来自任何远程 IP 地址的连接请求，应将 IP 地址设为 0.0.0.0。如果 IP 地址不为零，则 CPU 只接受来自指定 IP 地址的连接请求。对于被动连接，CPU 会忽略远程端口号(RemotePort)，RemotePort 可以设置为零。

您可以随时调用 TCP_CONNECT 指令以确定连接的当前状态。将 EN_R 输入设置为 FALSE 并提供有效的连接 ID(ConnectID)，TCP_CONNECT 返回以下内容：

- Busy，如果连接过程仍在进行中。
- Done，如果连接处于激活状态并准备发送或接收。
- Error，如果连接不可用。ErrorID 显示其中一种错误代码，用于指示存在的问题。
- LinkIPaddr1，输出当前连接设备 IP 地址第 1 位。
- LinkIPaddr2，输出当前连接设备 IP 地址第 2 位。
- LinkIPaddr3，输出当前连接设备 IP 地址第 3 位。
- LinkIPaddr4，输出当前连接设备 IP 地址第 4 位。
- LinkPort，输出当前连接设备端口号。

请注意：

- 主动连接可能最多需要 30 秒的时间来确定远程设备是否允许连接。被动连接显示 Busy 状态，直到远程设备尝试连接到 CPU。
- 主动连接 EN_R 保持为高电平时，如果远程设备断开设备连接，则 CPU 自动尝试重新连接到远程设备。
- 连接关闭后，CPU 不会自动尝试重新连接到设备，主动连接和被动连接皆如此。

9.1.13 TCP_SEND（TCP 数据发送）

- 功能：使用 TCP 协议进行以太网数据发送。
- 输入参数

输入参数	数据类型	参数描述	初始值
EN_R	BOOL	上升沿：CPU 启动发送操作	0
ConnectID	BYTE	连接 ID(ConnectID)是此发送操作所用连接的编号。使用您为 TCP_CONNECT 操作选择的 ConnectID	0
DataLen	WORD	请求发送数据长度，1~64K	0
DataAddr	POINTER TO BYTE	DataAddr 是指向发送数据存储位置的指针	0

- 输出参数

输出参数	数据类型	参数描述	初始值
Done	BOOL	当发送操作完成且没有错误时，指令置位 Done 输出	0
Busy	BOOL	当发送操作正在进行时，指令置位 Busy 输出	0
Error	BOOL	当发送操作完成但发生错误时，指令置位 Error 输出	0
ErrorID	WORD	如果指令置位 Error 输出，ErrorID 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，ErrorID 为零（无错误） 有关详细信息，请参见附录 6 以太网自由协议通信指令错误码	0
SentLen	WORD	SentLen 是实际发送的字节数。仅当指令置位 Done 或 Error 输出时 SentLen 才有效。如果指令置位 Done 输出，则整条消息信息发送完成	0

当发生以下情况时，TCP_SEND 指令启动发送指定数量的字节的操作：

- 程序通过将 EN_R 输入设置为 TRUE 来调用指令。
- 连接当前未用于执行其它发送操作。

EN_R 输入使用上升沿触发器，以便指令不启动意外的发送操作。TCP_SEND 处于繁忙状态时，程序会忽略 EN_R 输入。Done、Busy 和 Error 输出及 ErrorID 输出字节显示各调用的 TCP_SEND 状态。

发送操作完成后，指令显示调用一次 TCP_SEND 的 Done 或 Error 状态。此后，TCP_SEND 通过错误代码作出响应(错误描述信息：没有待决操作)，(如果通过将 EN_R 输入设置为 FALSE 进行调用)。如果 EN_R 输入设置为 TRUE，则程序会启动另一个发送操作。下面描述了输入和输出参数之间的关系。

- EN_R 设置为 TRUE 以便开始执行消息发送操作。Busy 设置为 TRUE。
- 消息发送完成。Done 置位，Busy 清零。
- EN_R 为 FALSE，无任何消息发送操作正在执行。
- EN_R 再次设置为 TRUE，因此开始执行另一消息发送操作。Busy 设置为 TRUE。
- 消息发送完成。Done 置位，Busy 在一个扫描周期内清零。

9.1.14 TCP_RECV（TCP 数据接收）

- 功能：使用 TCP 协议进行以太网数据接收。
- 输入参数

输入数据	数据类型	参数描述	初始值
IN	BOOL	高电平使能持续接收	0
ConnectID	BYTE	连接 ID(ConnectID)是此发送操作所用连接的编号。使用您为 TCP_CONNECT 操作选择的 ConnectID	0
MaxLen	WORD	MaxLen 是要接收的最大字节数（例如，DataPtr 中缓冲区的大小（1 到 64K））	0
DataAddr	POINTER TO BYTE	DataAddr 是指向接收数据存储位置的指针	0

- 输出参数

输出数据	数据类型	参数描述	初始值
Done	BOOL	当接收操作完成且没有错误时，指令置位 Done 输出。当指令置位 Done 输出时，RcvdLen 输出有效	0
Busy	BOOL	当接收操作正在进行时，指令置位 Busy 输出	0
Error	BOOL	当接收操作完成但发生错误时，指令置位 Error 输出	0
ErrorID	WORD	如果指令置位 Error 输出，ErrorID 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，ErrorID 为零（无错误） 有关详细信息，请参见附录 6 以太网自由协议通信指令错误码	0
RcvdLen	WORD	RcvdLen 是实际接收的字节数。仅当指令置位 Done 或 Error 输出时 RcvdLen 才有效。如果指令置位 Done 输出，则指令接收整条消息。如果指令置位 Error 输出，则消息超出缓冲区大小(MaxLen)并被截断	0

TCP_RECV 指令具有 IN（使能）输入,由高电平触发。第一次执行 TCP_RECV 指令后，状态位显示指令处于繁忙状态。对 TCP_RECV 的后续调用会显示繁忙状态，直至 CPU 通过指定连接接收数据。

CPU 通过指定连接接收消息后，下一次执行 TCP_RECV 指令时，会执行以下任务：

- 将消息数据复制到程序的数据区(DataAddr)。
- 将 RcvdLen 输出设置为接收的字节数。
- 置位 Done 输出，清除 Busy 和 Error 输出，且将 ErrorID 输出字节值设置为零。

使用 TCP 协议时，TCP_RECV 指令返回自程序上次调用 TCP_RECV 指令后 CPU 通过指定连接接收到的所有字节。程序必须足够频繁地调用 TCP_RECV 指令以正确地描绘消息，因为 TCP 充当“流”协议。在 TCP 协议中没有消息描述（没有开始或结束标记）。因此，CPU 并不知晓消息何时开始或结束。

例如，让我们假设存在一个 TCP 客户端接连不断地将 10 条 10 字节的消息发送给 CPU，而且在此期间程序不调用 TCP_RECV 指令。程序在 CPU 接受所有 10 条消息后调用 TCP_RECV 指令时，TCP_RECV 指令以一条 100 个字节的接收消息返回此数据。程序负责足够频繁地调用 TCP_RECV 指令，以便按发送消息的原样接收每条消息。

9.1.15 UDP_CONNECT（创建 UDP 通信）

- 功能：使用 UDP 协议创建被动连接。
- 输入参数

输入数据	数据类型	参数描述	初始值
EN_R	BOOL	上升沿触发连接	0
ConnectID	BYTE	Socket 连接编号 1~5	0
LocalPort	WORD	本地设备上的端口号。本地端口号范围为 1 到 49151，但存在一些限制，具体请详见附录 5 以太网自由协议库指令共用参数中的 LocalPort 参数	0

- 输出参数

输出数据	数据类型	参数描述	初始值
Done	BOOL	当连接操作完成且没有错误时，指令置位 Done 输出	0
Busy	BOOL	当连接操作正在进行时，指令置位 Busy 输出	0
Error	BOOL	如果连接不可用。ErrorID 显示其中一种错误代码，用于指示存在的问题	0
ErrorID	WORD	如果指令置位 Error 输出，ErrorID 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，ErrorID 为零（无错误） 有关详细信息，请参见附录 6 以太网自由协议通信指令错误码	0

UDP_CONNECT 指令只需要连接 ID 和本地端口号即可创建连接。一个 UDP 连接可以将消息发送到任意数量的其它设备，因为 IP 地址和远程端口会随每个 UDP_SEND 指令一起提供。仅当需要多个本地端口时，才需要多个 UDP 连接。不能将同一本地端口号用于多个 UDP 连接。所有本地端口号必须唯一。

连接操作是异步的，可能需要几次扫描才能完成。没有与远程设备建立主动连接，也没有等待另一台设备连接到该 CPU。当连接操作待决时，Busy 输出置位。当连接操作完成时，程序置位 Done 输出。仅当输入参数发生问题或没有可用的被动连接时，程序才置位 Error 输出。如果程序将 Error 位置位，ErrorID 输出字节将包含错误代码。

指令处于繁忙状态时，不得更改 UDP_CONNECT 的参数，CPU 借此可了解这是启动连接过程的调用的延续。

您可以调用 UDP_CONNECT 指令以确定连接的当前状态。将 EN_R 输入设置为 FALSE 并提供有效的连接 ID(ConnectID)，UDP_CONNECT 指令返回以下内容：

- 如果连接处于激活状态并准备发送或接收，指令置位 Done 输出。这仅表示您可以使用该连接，并不表示存在任何远程设备。
- 如果连接仍在进行，指令置位 Busy 输出。
- 如果连接不可用，指令置位 Error 输出。ErrorID 输出字节显示其中一种错误代码，用于指示存在的问题。

9.1.16 UDP_SEND（UDP 数据发送）

- 功能：UDP_SEND 指令将来自请求的缓冲区位置(DataAddr)的请求的字节数(DataLen)传输到通过 IP 地址(IPaddr1 – IPaddr4)和端口(RemPort)指定的设备。该指令仅用于 UDP 协议和通过 UDP_CONNECT 创建的连接。
- 输入参数

输入数据	数据类型	参数描述	初始值
EN_R	BOOL	上升沿：CPU 启动发送操作	0
ConnectID	BYTE	连接 ID(ConnectID)是此发送操作所用连接的编号（连接过程中通过 UDP_CONNECT 定义）	0
DataLen	WORD	发送数据长度，1~64K	0
DataAddr	POINTER TO BYTE	DataAddr 是指向发送数据存储位置的指针	0
IPaddr1	BYTE	四字节 IP 地址，由高到低依次为 IPaddr1~ IPaddr4	0
IPaddr2	BYTE		0
IPaddr3	BYTE		0
IPaddr4	BYTE		0
RemotePort	WORD	远程设备上的端口号。远程端口号范围为 1 到 49151。对于被动连接，使用零	0

- 输出参数

输出数据	数据类型	参数描述	初始值
Done	BOOL	当发送操作完成且没有错误时，指令置位 Done 输出	0
Busy	BOOL	当发送操作正在进行时，指令置位 Busy 输出	0
Error	BOOL	当发送操作完成但发生错误时，指令置位 Error 输出	0
ErrorID	WORD	如果指令置位 Error 输出，ErrorID 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，ErrorID 为零（无错误） 有关详细信息，请参见附录 6 以太网自由协议通信指令错误码	0

当发生以下情况时，UDP_SEND 指令启动发送指定数量的字节的操作：

- 程序通过将 EN_R 输入设置为 TRUE 来调用指令。
- 连接当前未用于执行其它发送操作。

EN_R 输入由电平触发。建议对 EN_R 输入使用上升沿触发器，以便指令不启动意外的发送操作。UDP_SEND 处于繁忙状态时，程序会忽略 EN_R 输入。Done、Busy 和 Error 输出及 ErrorID 输出字节显示各调用的 UDP_SEND 状态。

发送操作完成后，指令显示调用一次 UDP_SEND 的 Done 或 Error 状态。此后，UDP_SEND 通过错误代码作出响应，这意味着没有待决操作（如果通过将 EN_R 输入设置为 FALSE 进行调用）。如果 EN_R 输入设置为 TRUE，则程序会启动另一个发送操作。下面描述输入和输出参数之间的关系。

- EN_R 设置为 TRUE 以便开始执行消息发送操作。Busy 设置为 TRUE。
- 消息发送完成。Done 置位，Busy 清零。

- EN_R 为 FALSE，但无任何消息发送操作正在执行。
- EN_R 再次设置为 TRUE，因此开始执行另一消息发送操作。Busy 设置为 TRUE。
- 消息发送完成。Done 置位，Busy 在一个扫描周期内清零。

9.1.17 UDP_RECV（UDP 数据接收）

- 功能：UDP_RECV 指令通过现有连接检索数据。该指令仅用于 UDP 协议以及通过 UDP_CONNECT 创建的连接。
- 输入参数

输入数据	数据类型	参数描述	初始值
IN	BOOL	高电平使能	0
ConnectID	BYTE	CPU 将连接 ID(ConnectID)用于此接收操作（UDP_CONNECT 连接过程中定义）	0
MaxLen	WORD	MaxLen 是要接收的最大字节数（例如，DataAddr 中缓冲区的大小（1 到 64K））	0
DataAddr	POINTER TO BYTE	DataAddr 是指向接收数据存储位置的指针	0

- 输出参数

输出数据	数据类型	参数描述	初始值
Done	BOOL	当接收操作完成且没有错误时，指令置位 Done 输出	0
Busy	BOOL	当接收操作正在进行时，指令置位 Busy 输出	0
Error	BOOL	当接收操作完成但发生错误时，指令置位 Error 输出	0
ErrorID	WORD	如果指令置位 Error 输出，ErrorID 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，ErrorID 为零（无错误） 有关详细信息，请参见附录 6 以太网自由协议通信指令错误码	0
RcvdLen	WORD	RcvdLen 是实际接收的字节数。仅当指令置位 Done 或 Error 输出时 RcvdLen 才有效。如果指令置位 Done 输出，则指令接收整条消息。如果指令置位 Error 输出，则消息超出缓冲区大小(MaxLen)并被截断	0
SentIPaddr1	BYTE	发送消息的远程设备 IP 地址的四个八位字节。SentIPaddr1 是 IP 地址的最高有效字节，SentIPaddr4 是 IP 地址的最低有效字节	0.0.0.0
SentIPaddr2	BYTE		
SentIPaddr3	BYTE		
SentIPaddr4	BYTE		
SentPort	WORD	发送消息的远程设备的端口号	0

UDP_RECV 指令具有 IN（使能）输入，由高电平使能触发。第一次执行 UDP_RECV 指令后，状态输出显示指令处于繁忙状态。对 UDP_RECV 的后续调用显示繁忙状态，直至 CPU 通过指定连接接收数据。

CPU 通过指定连接接收消息后，下一次执行 UDP_RECV 指令时，会执行以下任务：

- 将消息数据复制到程序的数据区(DataAddr)。
- 将返回的 RcvdLen 设置为接收的字节数。
- 将 SentIPaddr1~SentIPaddr4 地址设置为发送消息的远程设备的地址。

- 将远程端口号(SentPort)设置为远程设备的端口。
- 置位 Done 输出，清除 Busy 和 Error 输出，且将 ErrorID 输出字节值设置为零。

9.1.18 ETH_DISCONNECT（关闭以太网通信连接）

- 功能：ETH_DISCONNECT 指令用于终止现有通信连接。
- 输入参数

输入数据	数据类型	参数描述	初始值
EN_R	BOOL	上升沿触发关闭连接	0
ConnectID	BYTE	CPU 使用连接 ID(ConnectID)标识要终止的连接（连接过程中定义）	0

- 输出参数

输出数据	数据类型	参数描述	初始值
Done	BOOL	当断开连接操作完成且没有错误时，指令置位 Done 输出	0
Busy	BOOL	当断开连接操作正在进行时，指令置位 Busy 输出	0
Error	BOOL	当断开连接操作完成但发生错误时，指令置位 Error 输出	0
ErrorID	WORD	如果指令置位 Error 输出，ErrorID 输出会显示错误代码。如果指令置位 Busy 或 Done 输出，ErrorID 为零（无错误） 有关详细信息，请参见附录 6 以太网自由协议通信指令错误码	0

9.2 任务指令

9.2.1 TaskStart（运行任务）

- 功能：运行指定的任务。
- 输入参数

输入参数	类型	参数描述
uiTaskID	UDINT	任务 ID 号

- 返回值

返回值	类型	参数描述
0	DINT	成功
-1	DINT	失败

9.2.2 TaskStop（停止任务）

- 功能：停止指定的任务。
- 输入参数

输入参数	类型	参数描述
uiTaskID	UDINT	任务 ID 号

- 返回值

返回值	类型	参数描述
0	DINT	成功
-1	DINT	失败

9.2.3 TaskSuspend（挂起任务）

- 功能：暂停指定的任务。
- 输入参数

输入参数	类型	参数描述
uiTaskID	UDINT	任务 ID 号

- 返回值

返回值	类型	参数描述
0	DINT	成功
-1	DINT	失败

9.2.4 TaskStatus（返回任务状态）

- 功能：返回任务的当前状态。
- 输入参数

输入参数	类型	参数描述
uiTaskID	UDINT	任务 ID 号

- 返回值

返回值	类型	参数描述
1	DINT	运行
2	DINT	挂起
3	DINT	停止
-1	DINT	失败

9.3 文件操作

- “/media/sd/”、“/media/flash/”目录方式仅在 V1.1.0_R 之前的固件版本中适用，固件版本 V1.1.0_R 及以后版本上，目录方式为：“sd/”、“flash/”。关于控制器固件版本信息详见附录 4 控制器固件版本信息。
- 对“sd/”目录下的文件进行打开（FileOpen）、查找（FindFirstFile、FindNextFile）、扫描（FileListScan）、删除（FileRemove）等操作前，请确保 SD 卡已正确插入卡槽。

9.3.1 FileOpen（打开文件）

- 功能：根据用户设定的方式，打开文件。
- 输入参数

输入参数	类型	参数描述
Name	STRING	文件名，且只支持全路径方式。 “sd/”为 SD 卡目录 “flash/”为 Flash 目录 例如，sd/A.txt 或 flash/A.txt 文件命名规则： 1）最大支持 63 个字符 2）支持字母 a-z、A~Z，不区分大小写 3）支持数字 0~9 4）支持空格（注意：不要将空格作为文件名的结尾） 5）支持! # $ % & () - @ ^ _ ` { } ~ '
uiAccess	UDINT	文件操作方式 0：读写（默认以读写方式打开文件） 1：只读（以只读方式打开文件） 2：只写（以只写方式打开文件） 8：创建（创建并打开文件） 16：清空（清除文件中的内容并打开文件） 32：追加（将光标定位到文件末尾并打开文件） 注意： 1）除了追加方式，其他方式都会将光标定位到文件头 2）写入操作会对光标后的内容进行覆盖 3）通过 FileLseek 指令可以移动光标

- 返回值

返回值	类型	参数描述
文件访问 ID	DINT	成功
-1	DINT	失败

- 文件系统中单个文件最大支持 1.99 GB。

9.3.2 FileClose（关闭文件）

- 功能：关闭文件。
- 输入参数

输入参数	类型	参数描述
iFd	DINT	FileOpen 返回的文件 ID

- 返回值

返回值	类型	参数描述
0	DINT	成功

返回值	类型	参数描述
-1	DINT	失败

9.3.3 FileRead（读文件）

- 功能：读文件。
- 输入参数

输入参数	类型	参数描述
iFd	DINT	FileOpen 返回的文件 ID
pcBuf	POINTER TO USINT	读出来的数据的存放地址
uiSize	UDINT	期望读取文件的大小（byte）

- 返回值

返回值	类型	参数描述
实际读取文件的大小	DINT	成功
-1	DINT	失败

9.3.4 FileWrite（写文件）

- 功能：写文件。
- 输入参数

输入参数	类型	参数描述
iFd	DINT	FileOpen 返回的文件 ID
pcBuf	POINTER TO USINT	待写数据的存放地址
uiSize	UDINT	期望写入数据的大小（byte）

- 返回值

返回值	类型	参数描述
实际写入数据的大小	DINT	成功
-1	DINT	失败

- 在使用 FileRead/FileWrite 读写文件过程中，如操作返回-1，在确认输入参数无误的情况下，请关闭并重新打开文件，则可继续进行读写操作。

9.3.5 FileLseek（移动文件读/写指针）

- 功能：移动文件读写指针。
- 输入参数

输入参数	类型	参数描述
iFd	DINT	FileOpen 返回的文件 ID
iOffset	DINT	移动文件指针的偏移量（相对于用户设定的定位模

输入参数	类型	参数描述
		式）
uiFrom	UDINT	定位模式 0：定位到文件头 1：定位到文件当前位置 2：定位到文件尾

- 返回值

返回值	类型	参数描述
文件当前的读写位置	DINT	成功
-1	DINT	失败

9.3.6 FileListScan（扫描目录文件）

- 功能：扫描指定目录下包含的所有文件。
- 输入参数

输入参数	类型	参数描述
PathName	STRING	待扫描的目录名，最大支持 63 个字符，且只支持全路径方式。 “sd/”为 SD 卡目录 “flash/”为 Flash 目录
pFileNameList	POINTER TO USINT	存储扫描到的文件名列表（只显示文件名，不含路径）
cFileNum	SINT	期望扫描的文件个数，最大支持 32 个文件
cFileNameLen	SINT	为每一个文件名开辟的存储空间大小（包括结束符“\0”最大支持 64 个字符），推荐设置为 64

- 返回值

返回值	类型	参数描述
扫描到文件的个数	SINT	成功
-1	SINT	失败

- 需要根据输入参数的 cFileNum、cFileNameLen 两项来计算需要分配给存储扫描结果的数组的大小。比如：cFileNum 为 16、cFileNameLen 为 64，则存储扫描结果的数组至少为 1024 字节。

9.3.7 FindFirstFile（查找第一个文件）

- 功能：获得遍历目录下的第一个文件名，并返回文件遍历句柄。
- 输入参数

输入参数	类型	参数描述
PathName	STRING	待扫描的目录名，最大支持 63 个字符，且只支持全

输入参数	类型	参数描述
		路径方式。 “sd/”为 SD 卡目录 “flash/”为 Flash 目录
FileName	STRING	返回遍历到的文件名（不含路径，最大支持 63 个字符）

- 返回值

返回值	类型	参数描述
文件遍历句柄	DINT	成功
-1	DINT	失败

9.3.8 FindNextFile（查找下一个文件）

- 功能：遍历目录，获取指定目录的下一个文件名。
- 输入参数

输入参数	类型	参数描述
iFileFindHandle	DINT	文件遍历句柄（FindFirstFile 的返回值）
FileName	STRING	返回遍历到的文件名（不含路径，最大支持 63 个字符）

- 返回值

返回值	类型	参数描述
0	SINT	成功
-1	SINT	失败

9.3.9 FindClose（查找结束）

- 功能：关闭文件遍历句柄。
- 输入参数

输入参数	类型	参数描述
iFileFindHandle	DINT	文件遍历句柄（FindFirstFile 的返回值）

- 返回值

返回值	类型	参数描述
0	SINT	成功
-1	SINT	失败

9.3.10 FileRemove（删除文件）

- 功能：删除文件。
- 输入参数

输入参数	类型	参数描述

输入参数	类型	参数描述
Name	STRING	文件名，最大支持 63 个字符，且只支持全路径方式。 “sd/”为 SD 卡目录 “flash/”为 Flash 目录

- 返回值

返回值	类型	参数描述
0	DINT	成功
-1	DINT	失败

9.3.11 FlashWrite（写入掉电保持数据）

- 功能：文件 ID 取值 0~15，分别对应 Flash 上的 16 个文件。用户通过调用此函数可以完成将指定地址 1 KB 大小的数据保存在 Flash 对应的文件中。
- 输入参数

输入参数	类型	参数描述
iFileID	DINT	FileOpen 返回文件 ID
pDataAddr	POINTER TO USINT	掉电保护区地址

- 返回值

返回值	类型	参数描述
0	DINT	成功
-1	DINT	失败

9.3.12 FlashRead（读取掉电保持数据）

- 功能：文件 ID 取值 0~15，分别对应 Flash 上的 16 个文件。用户通过调用此函数可以完成将 Flash 上指定文件的数据恢复至数据区。
- 输入参数

输入参数	类型	参数描述
iFileID	DINT	FileOpen 返回文件 ID
pDataAddr	POINTER TO USINT	掉电保护区地址

- 返回值

返回值	类型	参数描述
0	DINT	成功
-1	DINT	失败

9.3.13 文件操作示例

MC 控制器通过系统指令库中的函数提供了一系列的文件操作指令，支持对以下两个目录中的文件进行读写、扫描、遍历等：

- flash/：此目录挂载用户 Flash，最大容量 16 MB。
- sd/：此目录挂载 SD 卡，最大容量 32 GB。

除此之外的其它目录均不支持。

- SD 卡采用 FAT32 文件系统，Flash 采用 JFFS2 文件系统，在 Flash 或 SD 卡上进行文件操作时，对应的文件命名规则应该与其文件系统命名规则一致。

9.3.13.1 文件读写示例

文件读写操作涉及到的函数指令如表 4 所示。

表 4 文件读写操作指令

序号	指令名称	说明
1	FileOpen	打开/创建文件
2	FileRead	读文件
3	FileWrite	写文件
4	FileLSeek	移动读写位置
5	FileClose	关闭文件
6	FileRemove	删除文件

1. 示例 1：写入、读取、比较文件内容

功能说明：对“flash/”目录下的“001.txt”文件进行读写操作，先写入 500 字节的数据，然后再读出刚写入的数据，并比较是否正确（ST 语言）：

第1步 操作步骤

（1） AutoThink 组态工程，添加一个 500 ms 周期任务，使用 ST 语言，该任务的组态逻辑在后边描述。

（2） 编译、下装并在线监视。

（3） 使能 EnSetName 开关，得到文件名。

（4） 通过写变量的方式，将文件操作方式变量 FileOprAccess 修改为 8，即需要创建该文件。

（5） 使能 EnOpen 开关，打开文件。

（6） 使能 EnClose 开关，关闭打开的文件。

（7） 通过写变量的方式，将文件操作方式变量 FileOprAccess 修改为 0，即需要以读写方式操作文件。

（8） 使能 EnOpen 开关，打开文件。

（9） 修改需要写入到文件的数据（即 WriteDataArr 数组），通过写变量的方式修改 WriteDataArr 数组中的数据即可。

（10） 使能 EnWrite 开关，开始写文件，可通过 WriteLen 变量来查看实际写成功的字节数。

（11） 修改文件当前的操作位置，即通过写变量方式将 SeekOffset 变量修改为 0，然后使能 EnSeek，即可将文件当前的操作位置调整到文件头。

（12） 使能 EnRead 开关，读取文件，读出的数据存储在 ReadDataArr 数组中，ReadLen 为实际读取的文件大小。

（13） 观察 ReadDataArr 数组中前 500 字节的数据是否与 WriteDataArr 数组中的前 500 字节数据相同。

（14） 使能 EnClose 开关，关闭打开的文件。

第2步 组态逻辑

为了支持周期运行任务对文件进行多次重复操作，工程组态时特意将单个文件操作独立出来。

表 5 变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值
1	EnSetName	-	设置文件名	BOOL	0
2	strFileName	-	存储文件名的字符串	STRING(63)	''
3	EnOpen	-	打开使能	BOOL	0
4	FileOprAccess	-	文件操作方式	UDINT	0
5	Fd	-	FileOpen 返回的文件 ID	DINT	-1
6	EnSeek	-	定位使能	BOOL	0
7	SeekMode	-	定位模式	UDINT	0
8	SeekOffset	-	期望定位的偏移	DINT	-1
9	FileOffset	-	文件当前操作的位置	DINT	-1
10	EnRead	-	读使能	BOOL	0
11	ExpectReadLen	-	期望读取文件的大小	UDINT	1024
12	ReadLen	-	实际读取文件的大小	DINT	-1
13	ReadDataArr	-	读取出来的文件数据的存储数组	ARRAY[0..1024] OF USINT	1025(0)
14	EnWrite	-	写使能	BOOL	0
15	ExpectWriteLen	-	期望写入的数据长度	UDINT	500
16	WriteLen	-	实际写成功的数据长度	DINT	-1
17	WriteDataArr	-	待写数据存储数组	ARRAY[0..2048] OF USINT	2049(0)
18	pReadBuf	-	指向读取数据存储数组的指针	POINTER TO USINT	0
19	pWriteBuf	-	指向待写数据存储数组的指针	POINTER TO USINT	0

序号	变量名	直接地址	变量说明	变量类型	初始值
20	i	-	临时控制循环的变量	UDINT	0
21	EnClose	-	关闭使能	BOOL	0
22	RetClose	-	关闭操作的返回值	DINT	-1

```
(* 设置文件名： flash/001.txt *)
IF EnSetName THEN
EnSetName := 0; (* 一次使能只设置一次 *)
strFileName := 'flash/001.txt'; (* 设置期望操作的文件名 *)
END_IF;

(* 打开文件 *)
IF EnOpen THEN
EnOpen := 0; (* 一次使能只打开一次 *)
Fd := FileOpen(strFileName, FileOprAccess);
END_IF;

(* 写文件 *)
IF EnWrite THEN
EnWrite := 0; (* 一次使能只写一次 *)
pWriteBuf := ADR(WriteDataArr);
WriteLen := FileWrite(Fd, pWriteBuf, ExpectWriteLen); (* 将 WriteDataArr 中的数据写入文件中 *)
END_IF;

(* 设置文件当前操作位置 *)
IF EnSeek THEN
EnSeek := 0; (* 一次使能只定位一次 *)
FileOffset := FileLseek(Fd, SeekOffset, SeekMode);
END_IF;

(* 读文件 *)
IF EnRead THEN
EnRead := 0; (* 一次使能只操作一次 *)

(* 读之前先清空读取缓存 *)
```

```
FOR i := 0 TO ExpectReadLen BY 1 DO
ReadDataArr[i] := 0;
END_FOR;

pReadBuf := ADR(ReadDataArr);
ReadLen := FileRead(Fd, pReadBuf, ExpectReadLen);
END_IF;

(* 关闭打开的文件 *)
IF EnClose THEN
EnClose := 0; (* 一次使能只操作一次 *)
RetClose := FileClose(Fd); (* 成功关闭文件时，将文件句柄设置为初始值 *)
IF 0 = RetClose THEN
Fd := -1;
END_IF;
END_IF
```

2. 示例 2：删除文件

功能说明：删除“flash/”目录下的“001.txt”文件（ST 语言）。

第1步 操作步骤

（1） AutoThink 组态工程，添加一个 500 ms 周期任务，使用 ST 语言，该任务的组态逻辑在后边描述。

（2） 编译、下装并在线监视。

（3） 使能 EnSetName 开关，得到文件名。

（4） 使能 EnRemove 开关，删除文件，RmRet 为删除操作的返回值，如果为 0 则表示删除成功，-1 表示失败。

第2步 组态逻辑

表 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
1	EnSetName	-	设置文件名	BOOL	0
2	strFileName	-	期望删除的文件名	STRING(63)	''
3	EnRemove	-	删除使能	BOOL	0
4	RmRet	-	删除操作的返回值	DINT	-1

```
(* 设置文件名： flash/001.txt *)
IF EnSetName THEN
EnSetName := 0; (* 一次使能只设置一次 *)
```

```
(* 获取期望操作的文件名 *)
strFileName := 'flash/001.txt';
END_IF;

(* 删除文件 *)
IF EnRemove THEN
EnRemove := 0; (* 一次使能只操作一次 *)
RmRet := FileRemove(strFileName);
END_IF;
```

9.3.13.2 文件扫描示例

MC 控制器提供了文件扫描的功能，可以扫描用户指定目录下存在的文件，并将文件名保存在用户指定的数组中。

文件扫描涉及的函数指令如表 7。

表 7 文件扫描操作指令

序号	指令名称	说明
1	FileListScan	扫描目录文件

1. 示例

以下是扫描“flash/”目录的示例（ST 语言）。

第1步 操作步骤

（1） AutoThink 组态工程，添加一个 500 ms 周期任务，使用 ST 语言，该任务的组态逻辑在后边描述。

（2） 编译、下装并在线监视。

（3） 向“flash/”目录上传几个文件。

（4） 使能 EnSetName 开关，设置扫描目录名。

（5） 使能 EnScan 开关，开始扫描，扫描结果存储在 FileNameArr 数组中，FileNum 为实际扫描到的文件个数。

第2步 组态逻辑

表 8 变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值
1	EnSetName	-	设置扫描目录名	BOOL	0
2	strScanPath	-	期望扫描的目录名	STRING(63)	‘’
3	EnScan	-	使能扫描操作	BOOL	0
4	FileNum	-	实际扫描到的文件数	SINT	-1

序号	变量名	直接地址	变量说明	变量类型	初始值
5	FileNameArr	-	存储扫描结果的数组	ARRAY[0..2048] OF USINT	2049(0)
6	ExpectScanFileNum	-	预计扫描的文件数	SINT	32
7	ScanFileNameLen	-	为每一个文件名开辟的存储空间大小	SINT	64
8	pFileScanResult	-	指向扫描结果存储数组的指针	POINTER TO USINT	0
9	i	-	临时控制循环的变量	INT	0

```
    (* 设置扫描目录 *)
    IF EnSetName THEN
    EnSetName := 0; (* 一次使能只操作一次 *)
    strScanPath := ' flash/'; (* 获取期望扫描的目录 *)
    END_IF;

    (* 扫描操作 *)
    IF EnScan THEN
    EnScan := 0; (* 一次使能只操作一次 *)

    (* 清零存储扫描结果的数组 *)
    FOR i := 0 TO 2048 BY 1 DO
    FileNameArr[i] := 0;
    END_FOR;

    pFileScanResult := ADR(FileNameArr);
    FileNum := FileListScan(strScanPath, pFileScanResult, ExpectScanFileNum,
ScanFileNameLen);
    END_IF
```

9.3.13.3 文件遍历示例

MC 控制器提供了文件遍历的功能，可以逐个查找用户指定目录下的所有文件。

具体的操作包括三个步骤：

- 打开遍历文件句柄
- 遍历查找文件
- 关闭遍历文件句柄

文件遍历涉及的函数指令如表 9 所示。

表 9 文件遍历操作指令

序号	指令名称	说明
1	FindFirstFile	查找第一个文件
2	FindNextFile	查找下一个文件
3	FindClose	查找结束

1. 示例

第1步 操作步骤

（1） AutoThink 组态工程，添加一个 500 ms 周期任务，使用 ST 语言，该任务的组态逻辑在后边描述。

（2） 编译、下装并在线监视。

（3） 向“flash/”目录上传几个文件。

（4） 使能 EnSetName 开关，设置遍历目录名。

（5） 使能 EnFind 开关，开始遍历并得到第一个文件名，遍历结果存储在 FileNameArr 数组中，Fd 为 FindFirstFile 返回的遍历句柄，如果为-1 表示遍历失败。

（6） 使能 EnFindNext 开关，继续下一个文件的遍历，遍历结果存储在 FileNameArr 数组中，iFindRet 为当前遍历的返回值，如果为-1 表示遍历失败。

（7） 再次使能 EnFindNext 开关，继续下一个文件的遍历，直至 iFindRet 为-1。

（8） 遍历完成后，使能 EnClose 开关，关闭遍历句柄，结束遍历操作。

第2步 组态逻辑

表 10 变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值
1	EnSetName	-	设置遍历的目录名	BOOL	0
2	strScanPath	-	期望遍历的目录名	STRING(63)	‘’
3	EnFind	-	使能获得遍历目录下的第一个文件名	BOOL	0
4	Fd	-	FindFirstFile 返回的遍历句柄	DINT	-1
5	strFileName	-	当前遍历操作返回的文件名的存储字符串	STRING(63)	‘’
6	EnFindNext	-	使能遍历获取下一个文件名	BOOL	0
7	iFindRet	-	FindNextFile 的返回值	SINT	-1
8	EnClose	-	使能关闭遍历句柄	BOOL	0
9	RetClose	-	关闭遍历句柄的返回值	SINT	-1

序号	变量名	直接地址	变量说明	变量类型	初始值
10	i	-	临时控制循环的变量	INT	0

```
(* 设置遍历目录 *)
IF EnSetName THEN
EnSetName := 0; (* 一次使能只操作一次 *)
strScanPath := ' flash/'; (* 获取期望遍历的目录 *)
END_IF;

IF EnFind THEN
EnFind := 0; (* 一次使能只操作一次 *)
Fd := FindFirstFile(strScanPath, strFileName);
END_IF;

IF EnFindNext THEN
EnFindNext := 0; (* 一次使能只操作一次 *)
strFileName := ''; (* 对返回查询结果的字符串清空 *)

iFindRet := FindNextFile(Fd, strFileName);
END_IF;

IF EnClose THEN
EnClose := 0; (* 一次使能只操作一次 *)
RetClose := FindClose(Fd);
IF 0 = RetClose THEN
iFindRet := -1;
Fd := -1;
END_IF;
END_IF
```

9.4 时间相关指令

9.4.1 GetTickCnt（获取开机计数）

- 功能：获取控制器自开机以来的时间。单位：μs，时间范围：0~232/10 μs（≈7 分 9 秒 496 毫秒 729.6 微秒）。若超过上限，则从 0 开始计数。
- 返回值

返回值	类型	参数描述
控制器开机以来的所经过的时间，溢出后从零重新计时（μs）	UDINT	成功

9.4.2 TimeDelay（时间延时）

- 功能：函数执行 uiTimeDelay 毫秒后返回，以达到时间延时的效果。
- 输入参数

输入参数	类型	参数描述
uiTimeDelay	UDINT	延时时间（单位：ms）

- 返回值

返回值	类型	参数描述
0	DINT	成功
-1	DINT	失败

9.4.3 SET_RTC（设置实时时钟）

- 功能：设置控制器的实时时钟。
- 输入参数

输入参数	类型	参数描述
EN_R	BOOL	使能 FALSE：无效（默认值） TRUE：上升沿有效
Year	WORD	年 1971~2036（默认值 2011）
Month	BYTE	月 1~12（默认值 1）
Day	BYTE	日 1~31（默认值 1）
Hour	BYTE	时 0~23（默认值 0）
Minute	BYTE	分 0~59（默认值 0）
Second	BYTE	秒 0~59（默认值 0）

- 输出参数

输出参数	类型	参数描述
Q	BOOL	设置完成 FALSE：未完成（默认值） TRUE：已完成
Error	BYTE	参数设置错误 0：设置 RTC 时间成功（默认值）

输出参数	类型	参数描述
		非 0，表示设置 RTC 时间时出现错误 1：设置年超出规定范围 2：设置月不合理 3：设置日不合理 4：设置时不合理 5：设置分不合理 6：设置秒不合理

9.4.4 GET_RTC（读取实时时钟）

- 功能：读取控制器的实时时钟。
- 输入参数

输入参数	类型	参数描述
IN	BOOL	使能 FALSE：无效（默认值） TRUE：有效

- 输出参数

输出参数	类型	参数描述
Q	BOOL	是否读出数据 FALSE：读取失败（默认值） TRUE：读取成功
DateTime	DT	日期/时间 DT#1971-01-01-00:00:00~ DT#2036-12-31-23:59:59 （默认值 DT#1970-01-01-00:00:00）
Year	WORD	年 1971~2036（默认值 0）
Month	BYTE	月 1~12（默认值 0）
Day	BYTE	日 1~31（默认值 0）
Hour	BYTE	时 0~23（默认值 0）
Minute	BYTE	分 0~59（默认值 0）
Second	BYTE	秒 0~59（默认值 0）
Week	BYTE	星期 1~7（默认值 0）

9.5 模块信息指令

9.5.1 sysGetModel（获取模块型号）

- 功能：获取模块型号。
- 输入参数

输入参数	类型	参数描述
IN	BOOL	高电平使能（默认值 FALSE）
Slot	BYTE	槽位号（有效值为 1）
DataOffset	DWORD	M 区的偏移（模块型号以字符串格式保存在指定偏移位置，默认值 0）

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出（默认值 FALSE）
Err	BYTE	错误（默认值 0） 1：槽号 slot 错误 2：DataOffset 偏移配置错误 129：返回的字符串超出 M 区范围 130：返回信息错误
RetStrLen	BYTE	模块型号字符串长度（默认值 0）

- 型号字符串格式：“Model：MC1008”

- 执行完此功能块后，Q 引脚输出为 1 代表执行完成。执行完成后，Err 如果非零，代表执行错误。

9.5.2 sysGetModuleSN（获取模块序列号）

- 功能：获取模块序列号。
- 输入参数

输入参数	类型	参数描述
EN_R	BOOL	使能 FALSE：无效（默认值） TRUE：上升沿有效
Slot	BYTE	槽位号（有效值为 1）
DataOffset	DWORD	M 区的偏移（序列号以字符串格式将保存在指定偏移位置）

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出（默认值 FALSE）
Err	BYTE	错误（默认值 0） 0：正确

输出参数	类型	参数描述
		1：槽号 slot 错误 2：DataOffset 偏移配置错误 129：返回的字符串超出 M 区范围 130：返回信息错误或读取过程中暂时未得到正确数据（最终读取成功后会变为 0）
RetStrLen	BYTE	模块序列号字符串长度（默认值 0）

- 序列号字符串格式：“SN：xxxxxx/yyyyyy”（模块序列号/模板序列号）

9.5.3 sysGetEthMAC（获取 Mac 地址）

- 功能：获取 Mac 地址。
- 输入参数

输入参数	类型	参数描述
EN_R	BOOL	使能 FALSE：无效（默认值） TRUE：上升沿有效
EthID	BYTE	网卡 ID 号（有效值为 0） 0：第 1 块网卡

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出（默认值 FALSE）
Err	BYTE	错误（默认值 0） 1：EthID 配置错误 128：获取 MAC 失败
MAC1	BYTE	MAC 地址第 1 部分
MAC2	BYTE	MAC 地址第 2 部分
MAC3	BYTE	MAC 地址第 3 部分
MAC4	BYTE	MAC 地址第 4 部分
MAC5	BYTE	MAC 地址第 5 部分
MAC6	BYTE	MAC 地址第 6 部分

- 执行成功后，Mac 地址字符串将保存在相对于 M 区偏移为 DataOffset 的内存地址上字符串格式，例如：“00:01:02:03:04:06”。

9.5.4 sysGetCPUModuleVern（获取控制器的版本信息）

- 功能：获取控制器的版本信息。
- 输入参数

输入参数	类型	参数描述

输入参数	类型	参数描述
EN_R	BOOL	使能 FALSE：无效（默认值） TRUE：上升沿有效

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出（默认值 FALSE）
Err	BYTE	错误（默认值 0） 128：获取版本失败
RTSVern	DWORD	RTS 版本 build 号 高 16 位为快核 RTS 版本 build 号，低 16 为慢核 RTS 版本 build 号
OSVern	DWORD	OS 版本 build 号 高 16 位为快核 OS 版本 build 号，低 16 为慢核 OS 版本 build 号
FPGAVern	WORD	FPGA 版本 build 号
HWVern	WORD	硬件版本号 返回值为 0～25，对应 A~Z
FWVern	WORD	模块固件版本号 所示的为“MC1008-A01”中的“01”

9.5.5 sysGetCheckTime（获取检验时间）

- 功能：获取检测时间。
- 输入参数

输入参数	类型	参数描述
EN_R	BOOL	使能 FALSE：无效（默认值） TRUE：上升沿有效
Slot	BYTE	槽位号（有效值为 1）

- 输出参数

输出参数	类型	参数描述
Q	BOOL	输出（默认值 FALSE）
Err	BYTE	错误（默认值 0） 0：正确 1：Slot 配置错误 129：获取失败
Year	WORD	年（默认值 0）
Month	BYTE	月（默认值 0）
Day	BYTE	日（默认值 0）
Hour	BYTE	时（默认值 0）

输出参数	类型	参数描述
Minute	BYTE	分（默认值 0）

第10章　异常处理

10.1 运动控制函数输入参数错误

在 IEC 程序中，运动控制函数输入参数错误时，将返回异常（0xFFFFFFFF）。

10.2 轴状态异常

轴参数 AxisStatus 中的异常，用户在 IEC 程序中可通过为 AxisErrMask 赋值的方式来配置异常处理。

用户在 IEC 程序中可以通过函数 HMC_ClearAxisErr 和 HMC_ClearAllErr 进行轴异常清除。

10.3 有限行程轴超限处理

对于有限行程轴，当轴发生超限时，包括硬限位超限和软限位超限，系统会对相关轴进行紧急停止，具体停止方式由参数 FsLimitMode/RsLimitMode 决定。

（1）FsLimitMode/RsLimitMode=0 自动撤销当前指令

遇到前/后向限位后自动撤销当前指令。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。

（a）单轴指令

正在执行单轴指令的轴遇到超限时，将 cancel 该指令，使用 FastDecel 快速停止当前轴的运动。即使在轴减速停止期间，即便不再发生超限（例如用户此时修改了 FSLimit/RSLimit 的值），当前指令仍将继续被 cancel。

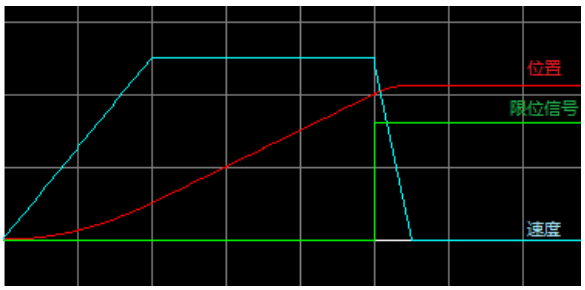


图 218　运动轨迹示意图

（b）插补合轴

对于正在执行插补指令的合轴超限，将 cancel 整个插补指令，并使用 FastDecel 快速停止当前轴的运动（若 FastDecel 小于 Decel，按照 cancel 方式以 Decel 停止，插补分轴也将按照插补关系快速停止。即使在轴减速停止期间，即便不再发生超限（例如用户此时修改了 FSLimit/RSLimit 的值），当前指令仍将继续被 cancel。

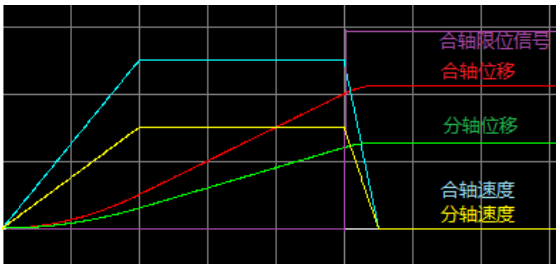


图 219 运动轨迹示意图

（c）插补分轴

对于正在执行插补指令的分轴超限，将 cancel 整个插补指令，并使用自身 FastDecel 快速停止当前分轴的运动，如下左图，分轴先与合轴减速停止；但若合轴 cancel 解算给分轴的实时减速度大于分轴 FastDecel，则按照合轴解算速度，如下右图，分轴选择合轴解算速度，两轴同时减速为 0。即使在轴减速停止期间，即便不再发生超限（例如用户此时修改了 FSLimit/RSLimit 的值），当前指令仍将继续被 cancel。

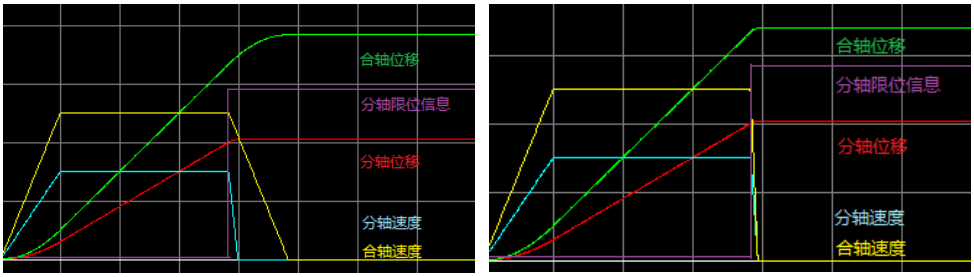


图 220 运动轨迹示意图

（d）联动指令从轴

对于正在执行联动指令的从轴， 将 cancel 该联动指令，并使用 FastDecel 快速停止当前从轴的运动。即使在轴减速停止期间，即便不再发生超限（例如用户此时修改了 FSLimit/RSLimit 的值），当前指令仍将继续被 cancel。

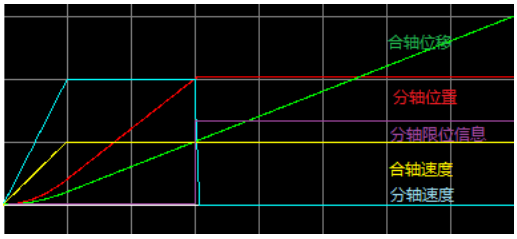


图 221 运动轨迹示意图

（2）FsLimitMode/RsLimitMode=1 手动撤销当前指令

遇到前/后向限位后手动撤销当前指令。当前指令在遇到函数 HMC_Cancel 之前是不会被自动撤销的，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。（目前暂时不支持该模式）

10.4 运行过程中的系统错误

用户在 IEC 程序中可以通过调用 HMC_GetSysErr 来获取系统错误，当发生系统错误时，运动控制任务将不再执行，同时也会将错误记录到控制器的错误记录文件里。OLED 屏上也会显示“ERR 错误码”。

10.5 运行过程中的用户错误

在运行过程中也可能发生用户错误（如由于设置错误，同时触发前/后向硬限位），用户在 IEC 程序中可以通过调用函数 HMC_GetUserErrNum、HMC_GetUserErrID、HMC_GetUserErrMes 来获取，详见这些函数的说明。同时也会将错误记录到控制器的错误记录文件里。

系统还提供了一些函数可以查询指令的状态和异常情况等，例如 HMC_GetCmdErr、HMC_GetCmdState、HMC_CaptureGetState、HMC_CaptureGetError 等，详见这些指令的说明。

10.6 指令间 Merge 处理

（1）目前仅支持一维、二维指令和 N 轴直线插补指令的 Merge 处理，插补指令的 Merge 以合轴 Merge 参数为准，不支持螺旋线、球弧、样条 Merge；

（2）一维指令（即单轴指令）中，只有当前指令为 HMC_Move、HMC_MoveEx、HMC_MoveAbs、HMC_MoveAbsEx 时，Merge 有效；

（3）二维指令（即两轴插补指令）中，只有当前指令为 HMC_Move2D、HMC_Move2DEx、HMC_MoveAbs2D、HMC_MoveAbs2DEx、HMC_MoveCircle、HMC_MoveCircleEx 时，Merge 有效；

（4）N 轴指令，支持 HMC_MoveND、HMC_MoveAbsND 指令之间 merge；

（5）二维指令中，若插补的合轴、分轴所对应的轴号序列发生变化时，Merge 无效；

（6）二维指令、一维指令之间相互连接时（包含二维指令的合轴与一维指令连接），Merge 无效；

（7）若当前指令正在执行 HMC_MoveModify，HMC_SetMerge 参数设置失败；

（8）Merge 在深度前瞻性后仍无法达到正常 Merge 速度的非正常使用时，Merge 无效。如 Merge 的下条指令为 HMC_MoveEx(1)时，Merge 无效，正常减速；

（9）二维指令中，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）；

（10）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；

（11）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。

10.7 轴指令 ID 处理

对于非阻塞的单轴指令，当轴运动缓存指令的移动位移为 0（如 HMC_Move(0)）时，可以正常返回正确的指令 ID，若执行连续多个位移为 0 的非阻塞指令时，返回同一个指令 ID 且只有最后一条指令（即最后一个 HMC_Move(0)）ID 有效。

对于非阻塞的多轴指令，当轴运动缓存指令的合位移为0时，可以正常返回正确的指令ID，若执行连续多个位移为 0 的非阻塞指令时，返回同一个指令 ID 且只有最后一条指令 ID 有效。

附录 1 RTEX 指令错误码

分类	错误码	原因
指令头文件相关	0011h	节点地址（MAC-ID）不匹配
	0012h	C/R bit 为 1（即使有指令） 32 字节模式下，Sub_Chk 为 0
指令代码、控制模式相关	0021h	周期指令未定义
	0022h	周期指令正常时，非周期指令未定义 组合（控制模式和非周期指令）错误 32 字节模式下，子指令未定义
	002Eh	组合（通讯周期、半闭环/全闭环、16/32 字节模式、控制模式）错误 控制模式切换时间＜2 ms 在 PP 模式下的位置锁存或原点回归操作（Type_Code=12h、13h、31h、32h、33h）运行期间，切换控制模式 在执行非周期指令（Busy=1）时，切换控制模式 在速度控制（CV）/转矩控制（CT）模式下，执行原点复位指令（□4h）Type_Code=1□h/2□h
参数相关	0031h	Type_Code/Sub_Type_Code 未定义
	0032h	非周期性数据/子指令数据（非 Type_Code/Sub_Type_Code）超出设定范围
	0033h	周期性数据（Command_Data1）超出设定范围
	0034h	前馈数据（Command_Data3、Sub_Command_Data2/3）超出设定范围
不可执行 1（一般项）	0041h	对只读数据区进行写操作
	0042h	执行报警清除指令（该报警不可清除，且未发布警告）
	0043h	在非全闭环控制模式下或未检测到外部位置传感器错误时，执行外部位置传感器错误清除指令
	0045h	伺服开启时，在属性 C 参数验证模式下执行复位指令
	0046h	按照驱动器抑制输入（POT/NOT）减速并停止后，执行方向指令 POT/NOT 在按照驱动器抑制输入（POT/NOT）减速期间，启动 profile operation（Type_Code=31h，32h，33h 除外）
不可执行 2（原点返回相关）	0051h	编码器处于增量模式时，执行原点返回指令、多圈计数数据清除指令
	0052h	在绝对位置模式下的周期位置控制（CP）（包括全闭环控制）运行期间，执行原点回归指令（□4h）、Type_Code=1□h 在绝对位置模式下的轮廓位置控制模式（PP）（包括全关闭控制）运行期间，执行 profile homing 指令（轮廓位置模式下的原点回归）
	0053h	在绝对位置模式下的周期位置控制模式（CP）（包括全闭环控制）运行期间，执行原点回归指令（□4h）的实际位置设置/指令位置设置操作（Type_Code=21h，22h）
	0055h	全闭环控制模式下，执行原点回归指令、多圈位置数据清除指令
	0056h	伺服开启条件下，执行原点回归指令、多圈计数数据清除指令

分类	错误码	原因
	0057h	伺服关闭条件下，执行原点返回指令、Type_Code=1□h
	0058h	外部输入没有分配给正确的锁存端口时，以该外部输入作为触发源执行 Type_Code
	0059h	在轮廓位置锁存/轮廓位置原点回归（Type_Code=12h、13h、31h、32h、33h）期间，执行原点回归指令（□4h） 在轮廓位置控制模式/轮廓连续运转模式（Type_Code=10h，11h，20h）运行期间，执行初始化模式（Type_Code=1□h，31h）的原点回归指令（□4h）
	005Ah	设置 Z 相为位置锁存触发信号（当位置反馈信号来自外部绝对位置传感器时）
不可执行 3（硬件因素相关）	0061h	控制电压不足时，EEPROM 写禁止
不可执行 4（进行中）	0101h	正在执行之前的指令
	0102h	伺服驱动器正在访问编码器，不接受指令
	0103h	伺服驱动器正在访问外部位置传感器，不接受指令
	0104h	在轮廓位置控制模式（PP）运行期间，更改 Type_Code
不可执行 5（访问限制）	0201h	写参数或 EEPROM 写禁止，不接受指令 Pr.7.23 RTEX 功能扩展设定 2 的 bit 0 为 1 时，发布写参数指令或 EEPROM 写指令

附录 2 Monitor 的 Index 和 Type_Code 列表

Type_Code		名称		Index	单位	描述
旧代码	新代码					
101h	01h	位置微分（滤波后）	PERR	0、1、2	指令单位	位置控制模式：位置微分（滤波后） 全闭环控制模式：外部检测位置微分（滤波后） 速度/转矩控制模式：未定义 注意：建议使用索引 index=0（尽管索引为 1 或 2 时，返回值相同）
102h	02h	编码器分辨率	-	0	pulse/r	所接电机的编码器分辨率
104h	04h	内部指令位置（滤波后）	MPOS	0	指令单位	内部指令位置（滤波后）
105h	05h	实际速度	ASPD	0	由参数 Pr.7.25 确定	电机实际速度 速度单位可通过参数 Pr.7.25（“RTEX 速度单位设定”）进行设置 Pr.7.25 | 单位 0 | r/min 1 | 指令单位/s
106h	06h	转矩	TRQ	0	0.1%	指令转矩
-	07h	实际位置	APOS	0	指令单位	电机实际位置 在全闭环模式下，该位置为外部位移传感器的检测值
-	08h	内部指令位置（滤波前）	IPOS	0	指令单位	内部指令位置（滤波前）
-	09h	锁存位置 1	LPOS1	0	指令单位	锁存通道 CH1 锁存的电机实际位置
-	0Ah	锁存位置 2	LPOS2	0	指令单位	锁存通道 CH2 锁存的电机实际位置
-	0Ch	指令速率（滤波后）	MSPD	0	设置 Pr.7.25	指令速度（滤波后） 速度单位可通过参数 Pr.7.25（RTEX 速度单位设定）进行设置 Pr.7.25 | 单位 0 | r/min 1 | 指令单位/s 转矩控制模式下，数值未定义
-	0Dh	外部位置传感器检测位置	EXPOS	0	脉冲（外部位置传感器）	外部位置传感器检测位置
111h	11h	再生负载比率	-	0	0.1%	再生负载过载保护值/报警发生阈值
112h	12h	过载比例	-	0	0.1%	实际负载/额定电机负载
-	21h	逻辑输入信	-	0	-	输入信号逻辑电平

Type_Code		名称		Index	单位	描述
		号				
-	22h	逻辑输出信号	-	0	-	输出信号逻辑电平
-	23h	逻辑输入信号（扩展部分）	-	0	-	输入信号逻辑电平（扩展部分）
-	24h	逻辑输出信号（扩展部分）	-	0	-	输出信号逻辑电平（扩展部分）
-	25h	物理输入信号	-	0	-	输入信号物理电平
-	26h	物理输出信号	-	0	-	输出信号物理电平
131h	31h	负载惯量比	-	0	%	（负载惯量/电机转子惯量）*100，与 Pr.0.04 中的值相同
132h	32h	自动电机识别	-	0	-	0：无效 1：有效
133h	33h	不运转原因	-	0	-	造成电机不运转的原因序号
134h	34h	报警标志	-	0	-	造成电机不运转的原因序号 对应的数据位设为 1，以激活标志（显示报警状态）
201h	41h	机械角度(单转数据)	-	0	脉冲	造成电机不运转的原因序号 CCW（逆时针）旋转时，该数值增大 机械角度：0~（编码器精度-1） 注意：使用增量编码器时，控制电源开启后，在检测 Z 相前，Monitor_Data 始终保持为 FFFFFFFF[Hex]
202h	42h	电角度	-	0	-	电机的电角度 CCW（逆时针）旋转时，该数值增大 电角度范围：0~1FF[Hex]
-	43h	多圈计数数据	-	-	圈	绝对编码器的多圈计数值
-	61h	上电累计时间	-	-	30 分钟	伺服驱动器的控制电源累计开启时间 单次少于 30 分钟的上电不记录（因为上电记录时间记录以 30 分钟为单位）
-	62h	伺服驱动器温度	-	-	℃	伺服驱动器内部温度
-	63h	编码器温度	-	-	℃	编码器内部温度 仅适用于 20-bit 的编码器（不支持的编码器，温度显示为 0）
-	64h	浪涌电阻继电器动作次数	-	-	周期	浪涌电流抑制电阻继电器运行周期 在最大值 40,000,000 h 时，发生溢出 单次少于 30 分钟的上电不记录（因为上电记录时间记录以 30 分钟为单位）

Type_Code		名称		Index	单位	描述
-	65h	动态制动器动作次数	-	-	周期	动态制动继电器动作次数 在最大值 40,000,000 h 时，发生溢出 单次少于 30 分钟的上电不记录（因为上电记录时间记录以 30 分钟为单位）
-	66h	风扇运行时间	-	-	30 分钟	冷却风扇的运行时间 单次少于 30 分钟的上电不记录（因为上电记录时间记录以 30 分钟为单位） 没有风扇时，显示为 0
-	67h	风扇期望寿命	-	-	0.1%	风扇期望寿命百分比 单次少于 30 分钟的上电不记录（因为上电记录时间记录以 30 分钟为单位） 没有风扇时，显示为 0
-	68h	电容期望寿命	-	-	0.1%	主电源电容器期望寿命百分比 单次少于 30 分钟的上电不记录（因为上电记录时间记录以 30 分钟为单位）
-	69h	PN 端电压	-	-	V	主电源 PN 电压
401h	71h	RTEX 累计通讯错误	-	0	周期	RTEX 通讯错误总数 在最大值 FFFFh 时，发生溢出 重启伺服驱动器或重置控制电源时，计数清零
411h	81h	编码器累计通讯错误	-	0	周期	编码器间通讯错误总数 在最大值 FFFFh 时，发生溢出 重启伺服驱动器或重置控制电源时，计数清零
412h	82h	编码器累计通讯数据错误	-	0	周期	编码器通讯数据错误总数（累计） 在最大值 FFFFh 时，发生溢出 重启伺服驱动器或重置控制电源时，计数清零
-	83h	外部位置传感器累计通讯错误	-	0	周期	外部位置传感器通讯错误总数（累计） 在最大值 FFFFh 时，发生溢出 重启伺服驱动器或重置控制电源时，计数清零
-	84h	外部位置传感器累计通讯数据错误	-	0	周期	外部位置传感器通讯数据错误总数（累计） 在最大值 FFFFh 时，发生溢出 重启伺服驱动器或重置控制电源时，计数清零

- 关于指令错误（0031h），请联系设备生产商。
- Index 错误时，返回指令错误（0032h）。
- 旧代码类型适用于 A4N 的主命令；新代码类型适用于 A5N 的主命令和从命令（配合主命令使用时，高 4 位须设为 0）。建议使用新代码类型。
- 详见松下手册“SX-DSV02203_R2_2E”。

附录 3 EtherCAT 读写参数错误码

错误码	含义
0xC065000F	错误的从站号
0xC0650012	错误的对象索引
0xC0650013	总线不在通讯状态
0xC0650022	从站不允许读写参数
0xC0650023	从站不支持邮箱通信
0xC0650024	从站不支持 COE
0xC0650028	SDO 协议超时
0xC065002E	不支持访问对象
0xC065002F	对只写对象进行读操作
0xC0650030	对只读对象进行写操作
0xC0650031	对象在对象字典中不存在
0xC0650032	对象在 PDO 中不存在
0xC0650037	参数的数据类型、长度不匹配
0xC0650038	参数的数据类型不匹配，长度超大
0xC0650039	参数的数据类型不匹配，长度不足
0xC065003A	对象子索引不存在
0xC065003B	参数值超限
0xC065003C	参数值超上限
0xC065003D	参数值超下限
0xC065003E	最大值低于参数下限
0xC065003F	其他错误
0xC0650044	未知错误
0xFFFFFFFF	内部错误

附录 4　控制器固件版本信息

序号	控制器型号	固件版本
1	MC1008	A01、A02、B01、B02、C01、C02、D01、V1.0.0_R、V1.1.0_R、V1.2.0_R、V1.3.0_R
2	MC1004	A01、B01、B02、C01、V1.0.0_R、V1.1.0_R、V1.2.0_R、V1.3.0_R
3	MC1002R	A01、A02、V1.0.0_R、V1.1.0_R、V1.2.0_R、V1.3.0_R
4	MC1002E	A01、A02、V1.0.0_R、V1.1.0_R、V1.2.0_R、V1.3.0_R
5	MC1004L	A01、A02、V1.0.0_R、V1.1.0_R、V1.2.0_R、V1.3.0_R
6	MC1008L	V1.0.0_R、V1.1.0_R、V1.3.0_R

附录 5 以太网自由协议库指令共用参数

共用参数	参数说明
EN_R	输入用于发起操作。由电平触发。应通过上升沿指令将 EN_R 输入连接到库指令，以便操作仅启动一次。指令为 Busy 时程序会忽略 EN_R 输入
Active	Active 输入用于指定连接指令是创建主动客户端连接(Active=TRUE)还是创建被动服务器连接(Active=FALSE)。在主动连接中，本地 CPU 启动到远程设备的通信。在被动连接中，本地 CPU 等待远程设备启动通信
Done	当操作完成且没有错误时，指令置位 Done 输出。如果指令置位 Done 输出，Busy、Error 和 ErrorID 输出为零。仅当 Done 输出置位时，其它输出（例如，接收到的字节数）才有效
Busy	Busy 输出指示正在进行操作。通过将 EN_R 设为 TRUE 启动操作时，指令置位 Busy 输出。对于对指令的所有后续调用，Busy 输出保持置位，直到操作完成
Error	Error 输出指示操作完成但有错误。如果指令置位 Error 输出，则 Done 和 Busy 输出将设置为 FALSE。如果指令置位 Error 输出，则 ErrorID 输出会指明错误原因。如果 Error 输出置位，所有其它输出均无效
ConnectID	ConnectID 编号是连接的标识符。通过 TCP_CONNECT 或 UDP_CONNECT 创建连接时，可以选择 1 到 5 范围内的任何值，每个连接必须具有唯一的 ConnectID。因为程序需要使用 ConnectID 指定后续发送、接收和断开操作所需的连接 特别注意：同一个 ConnectID 相关的操作不允许在多个任务中调用
IPaddr1~IPaddr4	这些是远程设备 IP 地址的四个八位字节。IPaddr1 是 IP 地址的最高有效字节，IPaddr4 是 IP 地址的最低有效字节。例如：对于 IP 地址 192.168.0.250，设置以下值： IPaddr1=192 IPaddr2=168 IPaddr3=0 IPaddr4=250 IP 地址不能为以下值： 0.0.0.0（针对主动连接） 任何广播 IP 地址（例如，255.255.255.255） 任何多播地址 本地 CPU 的 IP 地址 可以将 IP 地址 0.0.0.0 用于被动连接。通过选择 IP 地址 0.0.0.0，CPU 接受来自任何远程 IP 地址的连接。如果为被动连接选择一个非零的 IP 地址，CPU 将仅接受来自指定地址的连接
RemotePort	远程设备上的端口号。端口号可用于 TCP 和 UDP 协议，从而路由设备内的消息 远程端口号的规则如下： 有效端口号范围为 1 到 49151 建议采用的端口号范围为 2000 到 49151 对于被动连接，CPU 会忽略远程端口号（可以将其设置为零）
LocalPort	是本地 CPU 上的端口号。端口号可用于 TCP 和 UDP 协议，从而路由设备内的消息。对于所有被动连接，本地端口号必须唯一 本地端口号的规则如下： 有效端口号范围为 1 到 49151 不能使用端口号 20、21、25、80、102、135、161、162、443、502、1200 以及 34962 至 34964。这些端口具有特定用途 建议采用的端口号范围为 2000 到 49151 对于被动连接，本地端口号必须唯一（不重复）

附录 6　以太网自由协议通信指令错误码

错误码	错误信息描述
0	无错误
1	输入 ConnectID 错误
2	输入 IP 地址非法
3	输入的远程端口号非法
4	输入的本地端口号非法
5	正在尝试使用 TCP_CONNECT 指令连接时，输入参数发生变化
6	正在尝试使用 TCP_CONNECT 指令创建主动连接，该路连接输入信息与其它路连接信息重复
7	正在尝试使用 TCP_CONNECT 指令创建被动连接，该路连接输入信息与其它路连接信息重复
8	连接已关闭
9	连接非 TCP 类型
10	建立连接失败
11	连接对象拒绝连接
12	接收到非设定 IP 地址的连接请求
13	以太网协议链路未配置以太网自由协议功能
14	在使用 TCP 或 UDP 自由协议发送或接收数据时，输入数据长度或地址信息错误
15	TCP_SEND 发送数据发生失败
16	TCP_RECV 接收数据发生失败
17	连接非 UDP 类型
18	正在使用 ETH_DISCONNECT 指令关闭已关闭的链路
19	UDP_CONNECT 连接输入信息与其它路连接信息重复
20	正在尝试使用 UDP_CONNECT 连接时，输入参数发生变化
21	TCP_CONNECT 设置套接字选项过程发生错误
22	TCP_CONNECT 绑定端口过程失败
23	TCP_CONNECT 链路监听 listen 过程失败
24	UDP_CONNECT 设置套接字选项过程发生错误
25	UDP_CONNECT 绑定端口过程失败
26	多个任务正在同时进行同一路连接操作

附录 7 CANopen 协议通讯指令错误码

表 11 错误码表 1

错误码	错误码描述
0x00	无错误
0x01	NodeID 错误
0x02	Index 错误
0x03	SubIndex 错误*/
0x04	写对象字典参数长度错误
0x05	NMT 命令字错误
0x1105	对象字典读写操作，从站无响应
0x1106	从站配置过程出错，无法进行对象字典读写操作
0x1107	主站信息存储内存错误
0x1108	主站状态错误
0x1109	主站配置错误
0x110A	主站初始化错误
0x110B	主站对象字典读写错误
0x110C	从站节点不存在
0x110D	从站状态错误
0x110E	从站配置错误
0x110F	从站 PDO 参数配置过程发生错误
0x1110	从站 SDO 读写响应过程发生错误
0x1111	从站 SDO 读写过程发生错误*/
0x1112	系统发生错误，无法进行对象字典读写操作
0x1113	链路协议方式配置错误
0x1114	从站设备类型匹配错误
0x1115	从站厂商代码匹配错误
0x1116	从站产品代码匹配错误
0x1117	从站产品版本匹配错误
0x1118	从站心跳异常
0x1119	CAN 链路数据发送失败

错误码	错误码描述
0x111A	CAN 链路接收溢出
0x111B	CAN 总线发生错误

表 12 错误码表 2

Abort code	Description
0x05030000	Toggle bit not alternated.
0x05040000	SDO protocol timed out.
0x05040001	Client/server command specifier not valid or unknown.
0x05040002	Invalid block size (block mode only).
0x05040003	Invalid sequence number (block mode only).
0x05040004	CRC error (block mode only).
0x05040005	Out of memory.
0x06010000	Unsupported access to an object.
0x06010001	Attempt to read a write only object.
0x06010002	Attempt to write a read only object.
0x06020000	Object does not exist in the object dictionary.
0x06040041	Object cannot be mapped to the PDO.
0x06040042	The number and length of the objects to be mapped would exceed PDO length.
0x06040043	General parameter incompatibility reason.
0x06040047	General internal incompatibility in the device.
0x06060000	Access failed due to an hardware error.
0x06070010	Data type does not match, length of service parameter does not match
0x06070012	Data type does not match, length of service parameter too high
0x06070013	Data type does not match, length of service parameter too low
0x06090011	Sub-index does not exist.
0x06090030	Value range of parameter exceeded (only for write access).
0x06090031	Value of parameter written too high.
0x06090032	Value of parameter written too low.
0x06090036	Maximum value is less than minimum value.
0x08000000	general error
0x08000020	Data cannot be transferred or stored to the application.

Abort code	Description
0x08000021	Data cannot be transferred or stored to the application because of local control.
0x08000022	Data cannot be transferred or stored to the application because of the present device state.
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error

宁波和利时智能科技有限公司

地址：宁波市鄞州区新材料国际创新中心7号楼5层

邮编：315000

电话：0574-8717 5756

传真： 010-5898 1558

http://www.hollysys.com