开放式控制系统软件
用户手册
V2.8

杭州芯控智能科技有限公司

HANGZHOU CORE CONTROL ROBOTICS CO.,LTD

感谢购买本公司产品，本手册为安全使用本公司产品而需要遵守的内容，在使用之前，请务必仔细阅读相关手册，并且在理解该内容的前提下正确使用。

有关本产品的详细功能，请用户通过相关说明书充分理解其规格。

我们试图在本手册中描述尽可能多的情况。然而，对于那些不必做的和不可能做的情况，由于种种原因，我们没有描述。因此对于那些在手册中没有描述的情况，可以视为“不可能”的情况。

- **版权声明**

商标声明

- **版本更新记录**

变更日期	版本号	变更说明
2022.11.08	V2.8	更新 i5 系列控制柜 IP 默认地址信息
2022.08.25	V2.7	更新软件描述
2022.08.10	V2.6	添加 4.12 OPC UA 通讯内容
2022.07.15	V2.5	添加 4.11 TCP/IP 通讯内容
2022.05.06	V2.4	整合轴组配置、更新示教库添加简化运动指令说明、添加混合连续轨迹程序、工具调用方式、附加轴跟踪算法说明
2022.04.18	V2.3	添加控制系统总览内容、更新适用机器人图片
2022.04.13	V2.2	添加常用 MC 指令详解、完整例程示例
2022.04.11	V2.1	更新 4、6 轴机器人示教库使用说明
2021.11.15	V2.0	添加示教库使用说明
2021.11.15	V1.2	添加 BLEND 运动控制指令说明
2021.04.11	V1.1	更新运动控制指令说明
2021.03.01	V1.0	初版发布

目 录

一、控制系统介绍

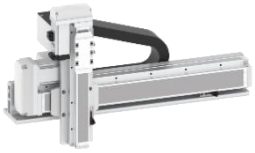
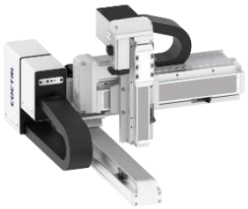
1.1 基本情况介绍

C²Cube运动控制器/柜搭载杭州芯控智能科技有限公司Corecontroller开放式系统软件开发，用户可以引用许多标准的功能函数库。采用高级语言的编程方式，易于PLC 厂家和用户开发自己专有的功能块和指令库，借用已有的类似控制程序，形成行业特点的“工艺包”，可显著提高用户编程效率。

1.2 支持算法总览

C²Cube 运动控制器/柜集成了市面常见型制机器人的运动控制算法，可实现多种复杂协同动作控制，总览如下，：

- **机器人运控算法：**
  1. 2 轴/3 轴/4 轴悬臂及龙门坐标机器人运动算法库；
  2. 耦合及非耦合式 3 轴/4 轴 SCARA、6 轴机器人运动算法库；
  3. AGV、RGV、复合机器人运动算法库；

<table>
<tr><td colspan="2">2 轴/3 轴/4 轴悬臂及龙门坐标机器人：</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td colspan="2">2 轴十字型悬臂坐标机器人</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>3 轴悬臂坐标机器人</td><td>4 轴悬臂坐标机器人</td></tr>
<tr><td colspan="2">注:龙门型机器人为长跨距悬臂机器人加辅助滑轨组成</td></tr>
</table>

SCARA 机器人、6 轴机器人：	
	
3/4 轴 SCARA（L1-一二轴线性式）机器人	3/4 轴 SCARA（L2-二轴线性式）机器人
	
3/4 轴 SCARA（L3-三轴线性式）机器人	6 轴机器人
AGV、复合机器人：	
	
二维码导航式 AGV	AGV+六轴复合机器人

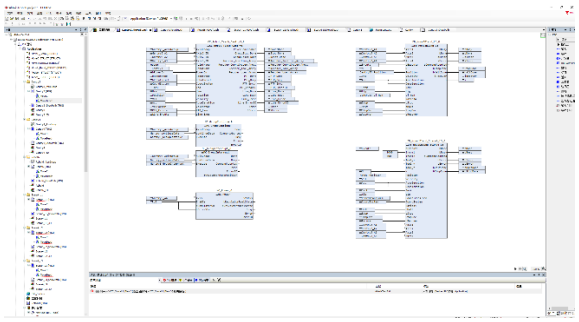
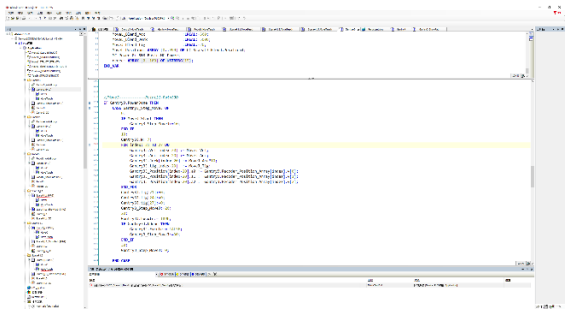
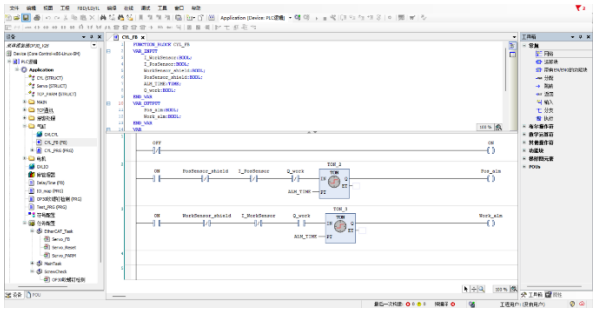
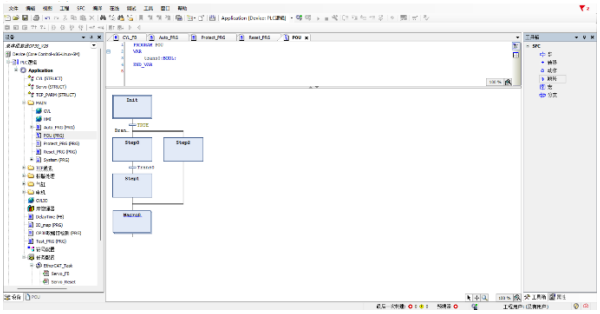
- **轨迹控制功能：**
  1. 单轴/轴组 PTP、插补、Blend 轨迹、高速电子凸轮；
  2. 坐标/关节机器人 MovJ、MovL、MovC；
  3. 梯型、二次曲线型、S 曲线型加/减速；
- **数据交互：**
  1. EtherCAT 伺服总线；
  2. RS232/RS485、TCP/IP、OPC UA；
  3. Web/HMI 访问系统变量；

4. 分布式 I/O;
5. 视觉、力觉传感器接入；

◆ **编程方式：**

1. IEC61131-3 标准化编程；
2. 顺序功能图（CFC）、功能块图（FBD）、梯形图（LD）、PLCopen；
3. 结构化文本（ST）；
4. 机器人式动作点位示教；

相关界面示例：

常用编程界面：	
	
FBD、SFC 功能块图编程	ST 结构化文本编程
	
LD 梯形图编程	CFC 顺序功能图

机器人式动作点位示教界面：

4 轴机器人示教界面（以 4 轴为例展示）

1.3 多台控制拓扑关系

C²Cube 运动控制器/柜可作为主控，实现单一控制器（主控单元）控制多台机器人或多伺服轴独立/协同动作，同时可直接用作视觉相机控制器，完成全站、全线完全控制，完整控制系统拓扑关系如下：

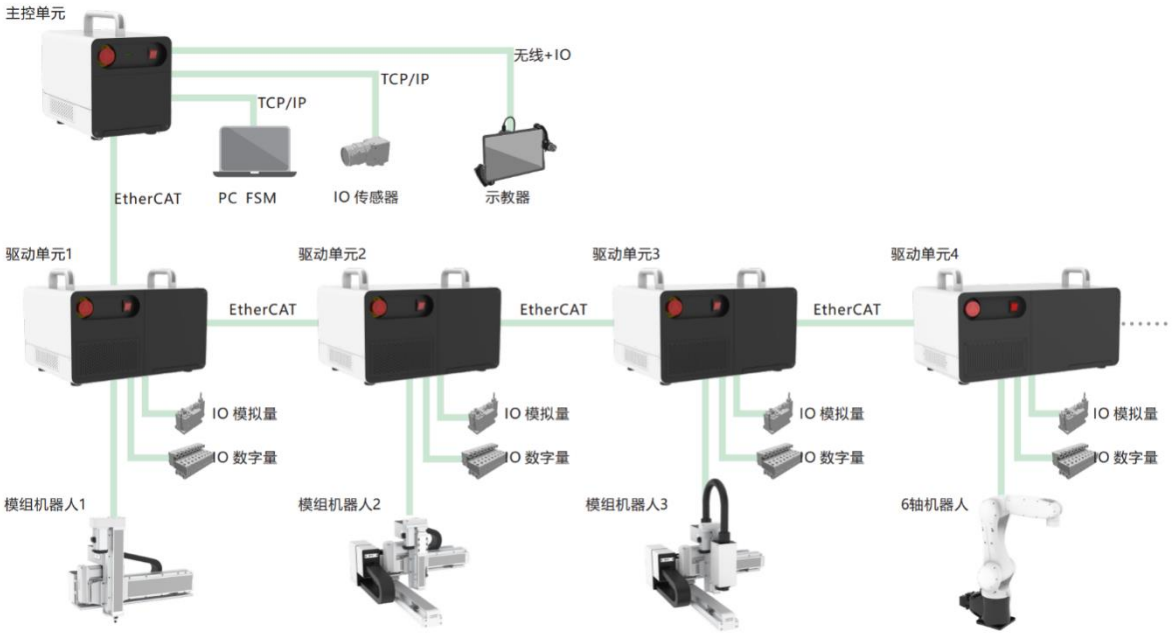


图 1

控制系统无需升级，可直接添加机器人本体及伺服轴进行拓展控制，方便设备后续升级。

二、运动控制程序的组成

2.1 用户程序结构

C²Cube 运动控制器是基于多任务操作系统开发的控制器，系统以多任务执行的模式运行多个功能模块，各任务间独立执行。

编写用户程序时，用户可根据应用系统中各任务的业务类型和紧急程度，分成若干个程序组织单元组成，同时可对每个任务指定不同的执行触发条件，或指定相应的执行时间间隔（也称执行周期），以让应用系统的控制响应达到最佳状态。

2.1.1 用户程序组成

C²Cube 运动控制器/柜可“同时”执行几个任务，每个任务（任务配置）由若干个用户程序组织单元（简称 POU）构成。按照 POU 执行特性要求，可分为若干个任务组，并配置其执行特性，没有列入任务配置的 POU 将不会被执行。另外用户工程中，还有一些支撑用户程序的对象，如库函数、全局变量 GVL、功能块 FB、凸轮定义 CAM 曲线、多轴插补轨迹定义 CNC 曲线等等。

2.1.2 任务类型

通过任务配置可以将用户程序按执行要求，分成若干个任务组，每个任务组可以设置不同的执行触发条件、执行时间间隔、优先级等。

C²Cube 运动控制器/柜常见的任务有：EtherCAT 任务、CANopen 任务、HSIO 高速中断任务、主循环任务等，其中，运动控制相关的用户程序主体，需安排在 EtherCAT 任务下执行。

运动控制功能的实时处理，均在在 EtherCAT 任务运中完成，该任务是一个执行时间间隔短（1-4ms）、优先级最高的一个时钟中断型的任务，一旦满足时间条件，它可以无条件地中断其他的任务，直到该任务配置下所有的 POU 执行完毕，才会退出。

2.2 建立用户程序的准备过程

2.2.1 软件环境与控制器连接

编程设备可以通过以太网（可经过集线器、交换机等）、WLAN（控制柜自建 AP）与控制器/柜相连接，使用 Corecontrol 软件需配置相应的网络 IPV4 地址。

C10-19 系列控制柜、MC200 系列控制器默认 IP 设置如下：

Windows 系统：

默认以太网 IP 地址配置为 192.168.1.143，子网掩码：255.255.0.0；

默认 WLANIP 地址配置为 192.168.0.100，子网掩码：255.255.0.0；

C10-i5 系列控制柜默认 IP 设置如下：

Linux 系统：

默认以太网 IP 地址配置为 192.168.1.143，子网掩码：255.255.0.0；

默认 WLANIP 地址配置为 192.168.0.100，子网掩码：255.255.0.0；

Windows 系统：

默认以太网 IP 地址配置为 192.168.1.143，子网掩码：255.255.0.0；

默认 WLANIP 地址配置为 192.168.0.102，子网掩码：255.255.0.0；

图 2

确认控制器是否能够连接

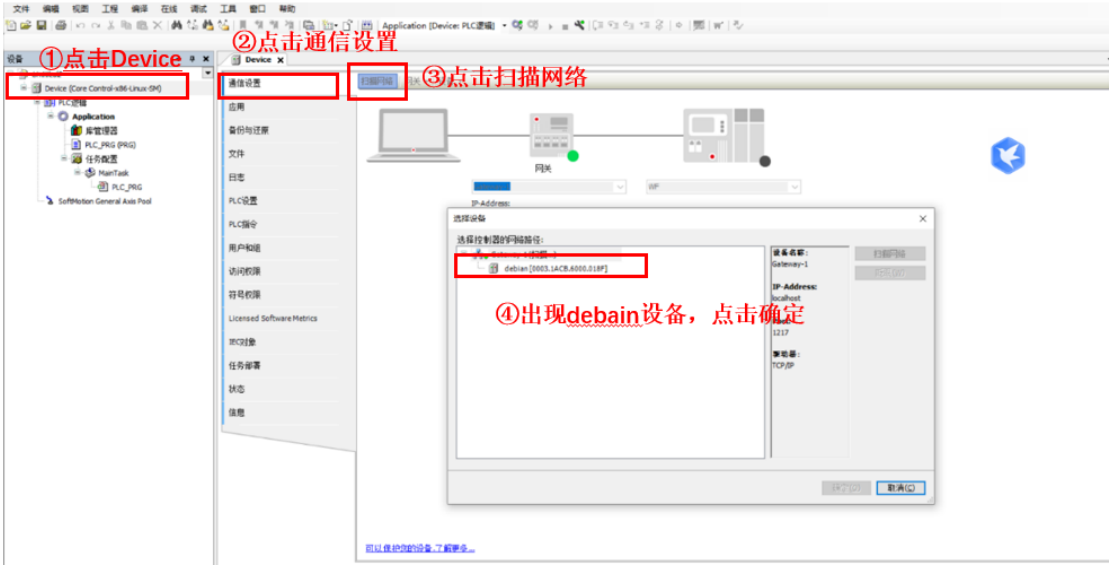


图 3

2.2.2 伺服以及 IO 连接

Corecontrol 软件与实际设备关联，选择先加载设备描述文件（XML 文件），软件即可自动扫描出现设备。

首先要添加 EtherCAT Master SoftMotion 主站，配置主站的网卡信息，自动扫描设备。

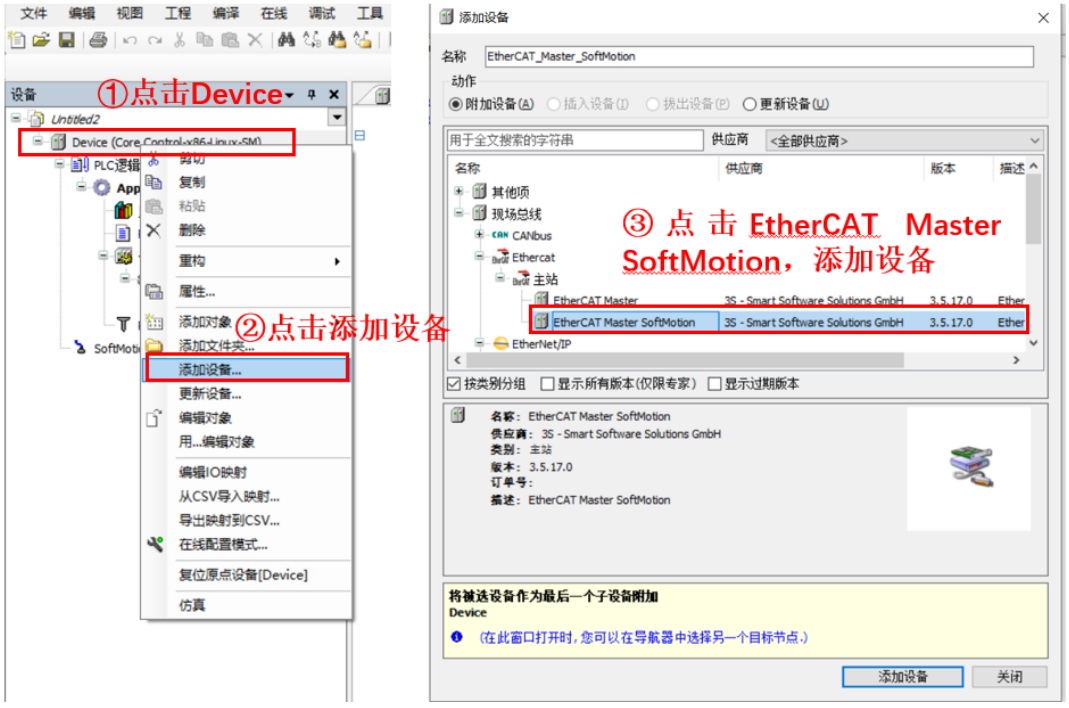


图 4

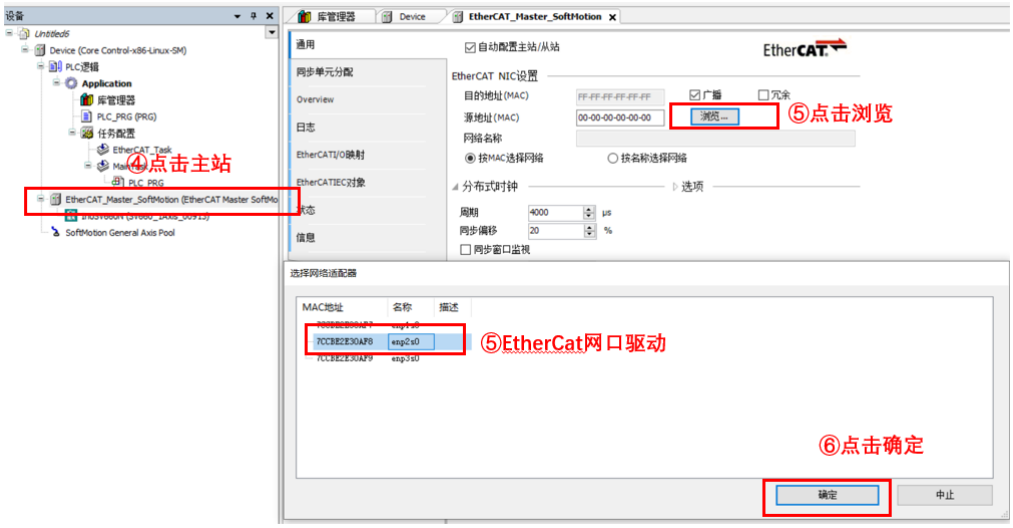


图 5

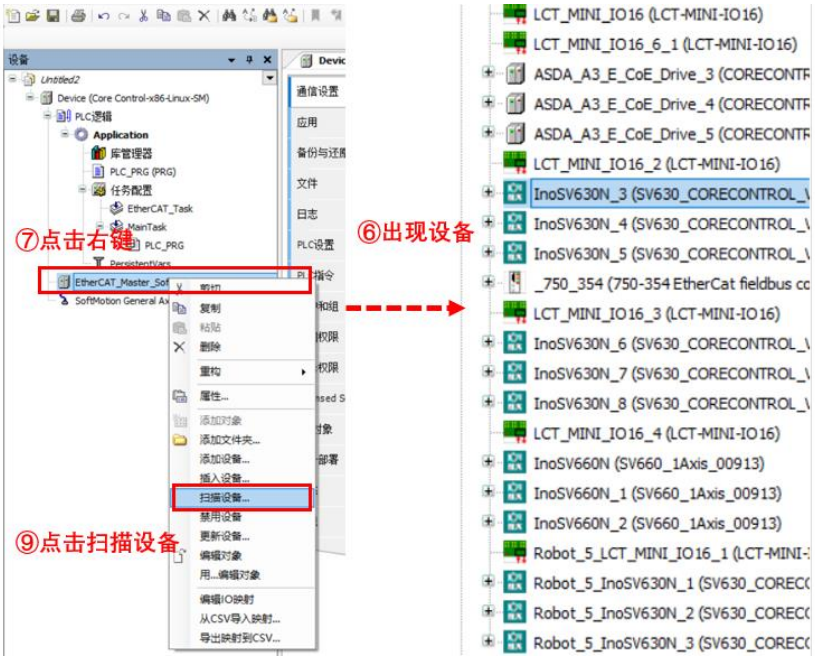


图 6

2.2.3 添加 XML 文件

在扫描出设备前，需要添加设备描述文件（XML 文件）。

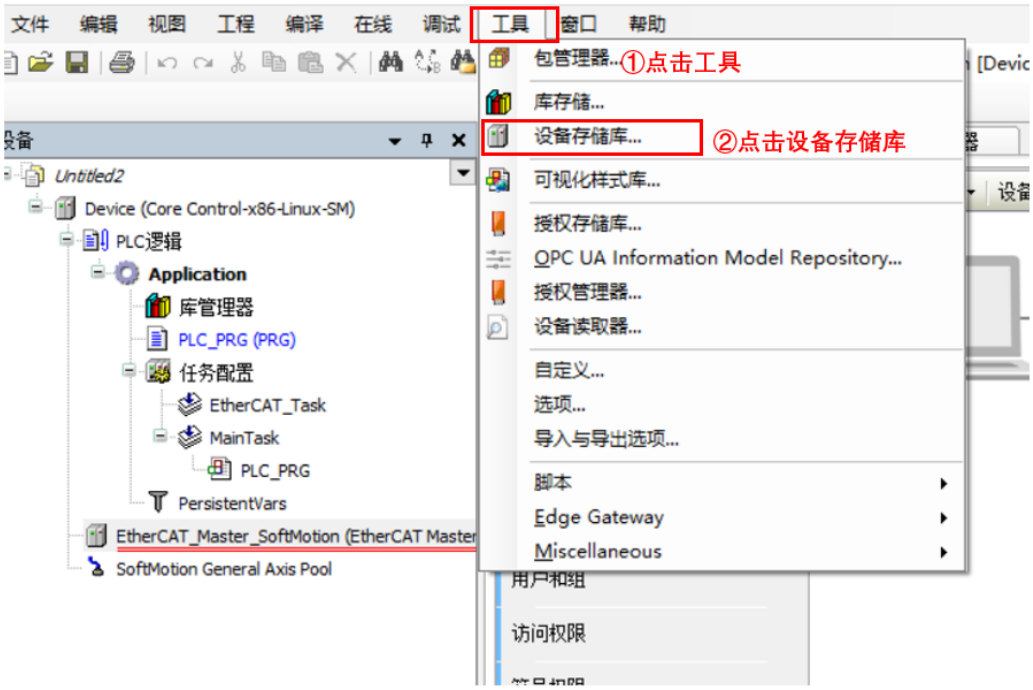


图 7

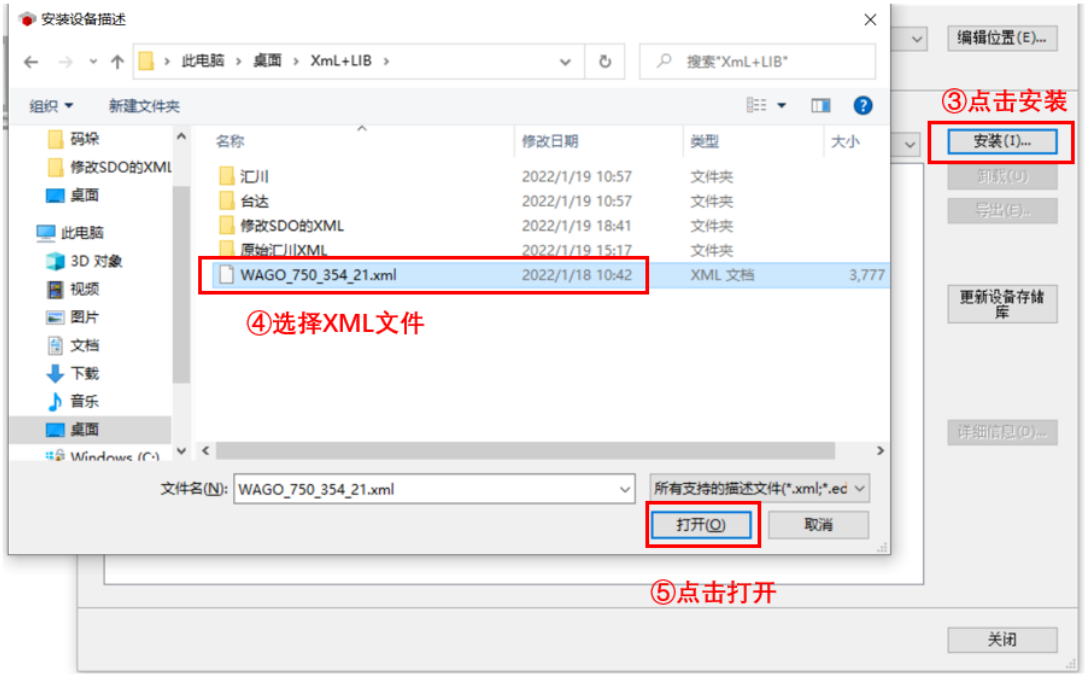


图 8

2.2.4 添加外部库文件以及加载外部库

在使用此软件时，若需要引用外部库文件时，需先加载外部库，然后在添加，才可以使用。


文件 编辑 视图 工程 编译 在线 调试 工具 窗口 帮助
包管理器...
①点击工具
库存储...
设备存储库...
②点击库存储
可视化样式库...
授权存储库...
OPC UA Information Model Repository...
授权管理器...
设备读取器...
自定义...
选项...
导入与导出选项...
脚本
Edge Gateway
Miscellaneous
Untitled2
Device (Core Control-x86-Linux-SM)
PLC逻辑
Application
库管理器
PLC_PRG (PRG)
任务配置
EtherCAT_Task
MainTask
PLC_PRG
PersistentVars
EtherCAT_Master_SoftMotion (EtherCAT Master
SoftMotion General Axis Pool
用户和组
访问权限

图 9

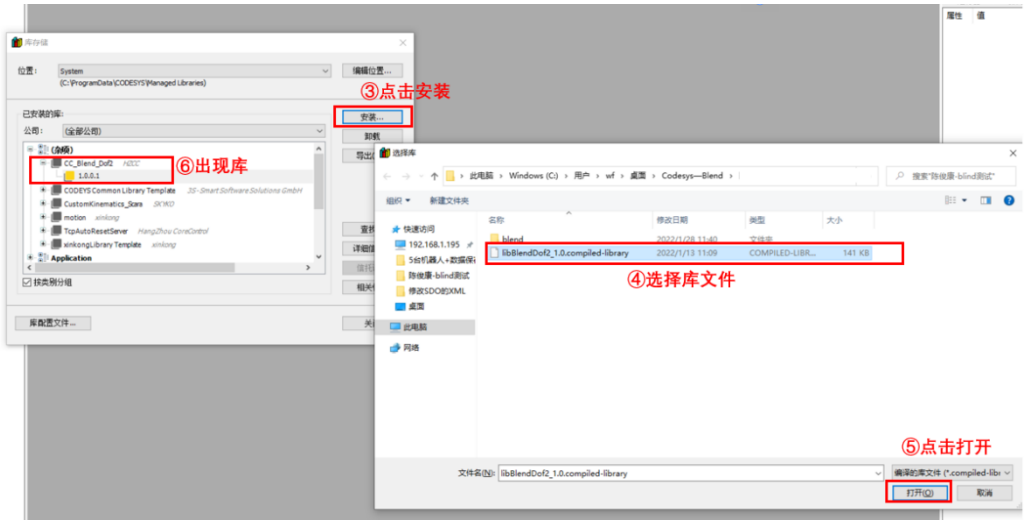

③点击安装
⑥出现库
④选择库文件
⑤点击打开
CC_Blend_Dof2
1.0.0.1
libBlendDof2_1.0.compiled-library
打开(O)
取消

图 10

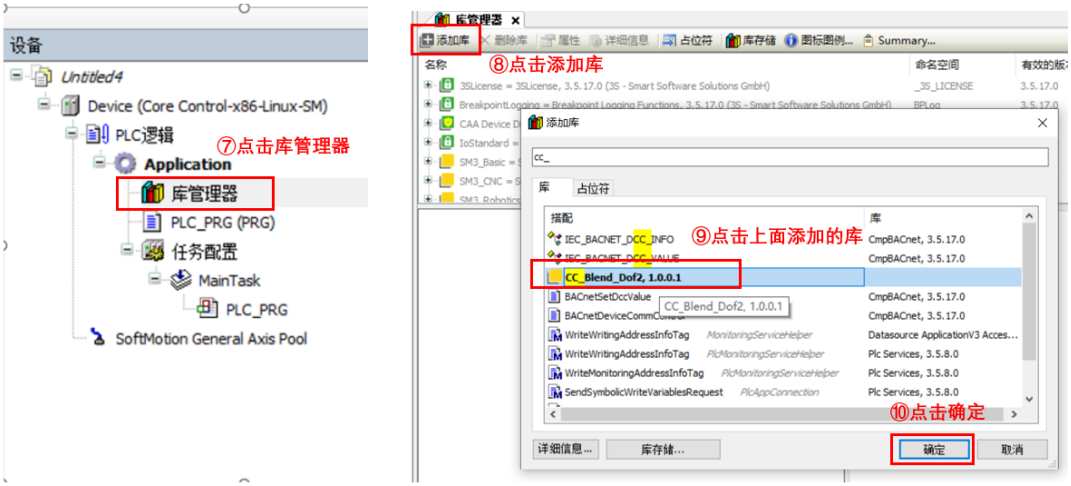

设备
Untitled4
Device (Core Control-x86-Linux-SM)
PLC逻辑
⑦点击库管理器
Application
库管理器
PLC_PRG (PRG)
任务配置
MainTask
PLC_PRG
SoftMotion General Axis Pool
库管理器
添加库
删除库
属性
详细信息
占位符
库存储
图标图例...
Summary...
⑧点击添加库
名称
命名空间
添加库
cc_
库
占位符
描配
⑨点击上面添加的库
CC_Blend_Dof2, 1.0.0.1
CC_Blend_Dof2, 1.0.0.1
⑩点击确定
详细信息...
库存储...
确定
取消

图 11

2.2.5 关键运动库明细

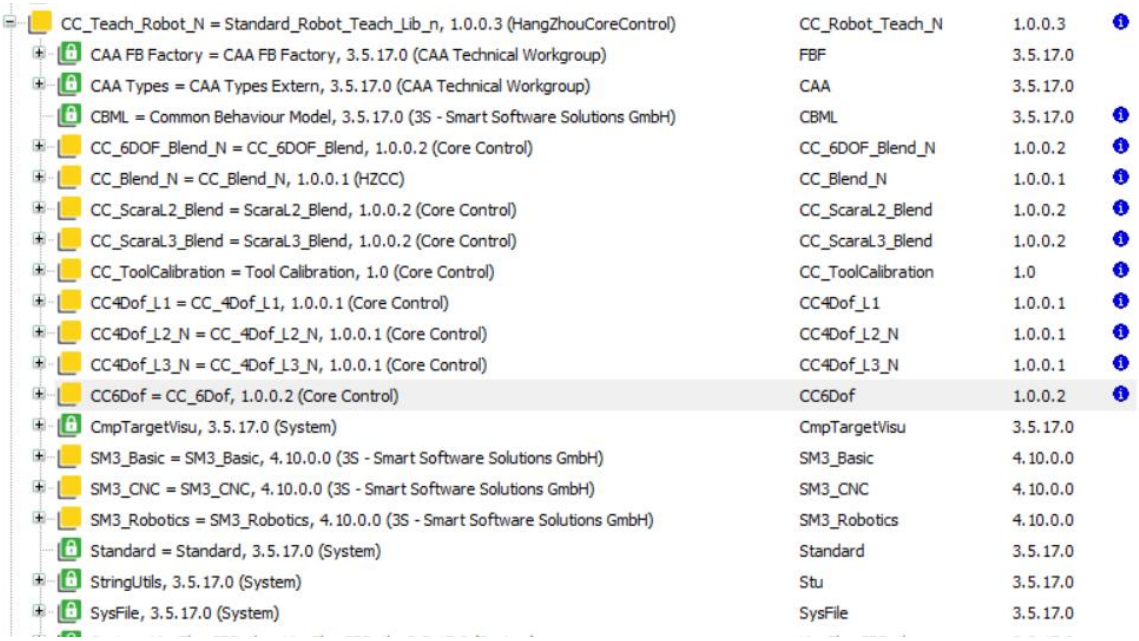


图 12

图为完整的示教库：CC_Teach_Robot_N，用于三轴桁架、四轴桁架、Scara 机器人、六轴机器人的点动、示教记录、ptp 运动、直线运动、圆弧运动、工具坐标系标定等。

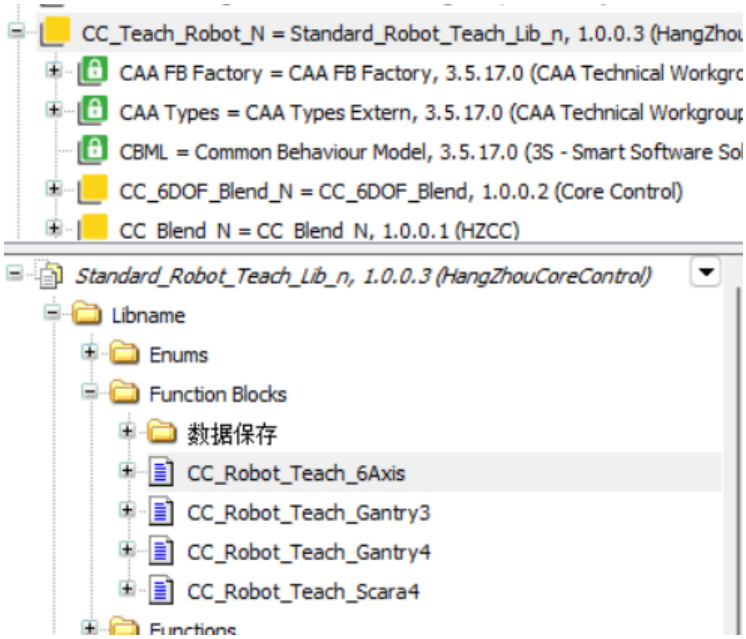


图 13

图中为各个模型的示教界面功能块。除此之外，还包含以下几个重要运动功能库如下：

2.2.5.1 六轴机器人 Blend 库

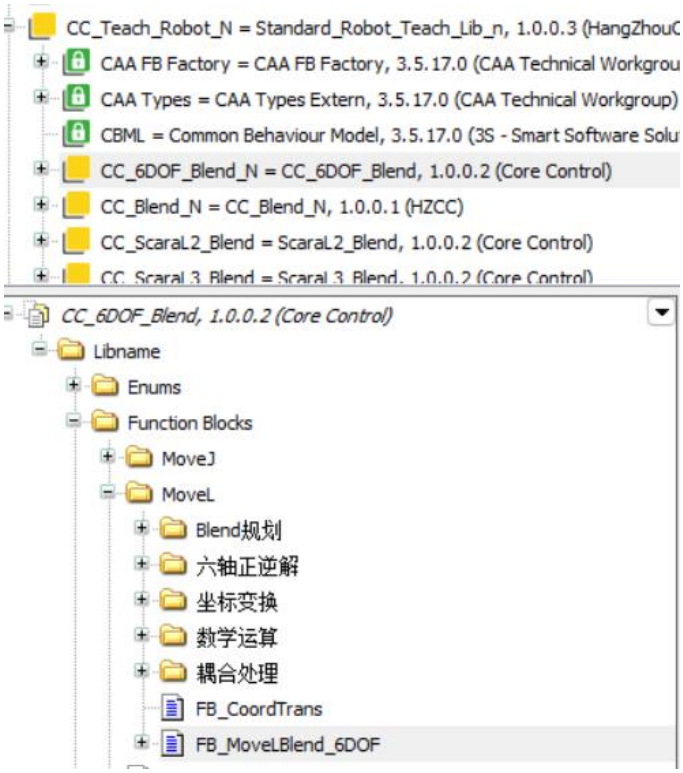


图 14

2.2.5.2 通用 Blend 库

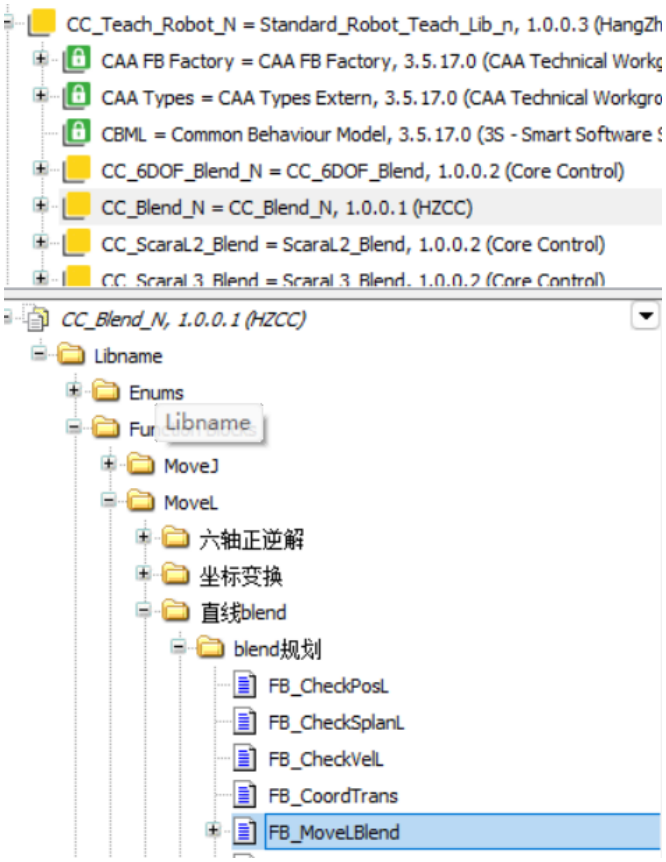


图 15

2.2.5.3 Scara_L2 机器人 Blend 库

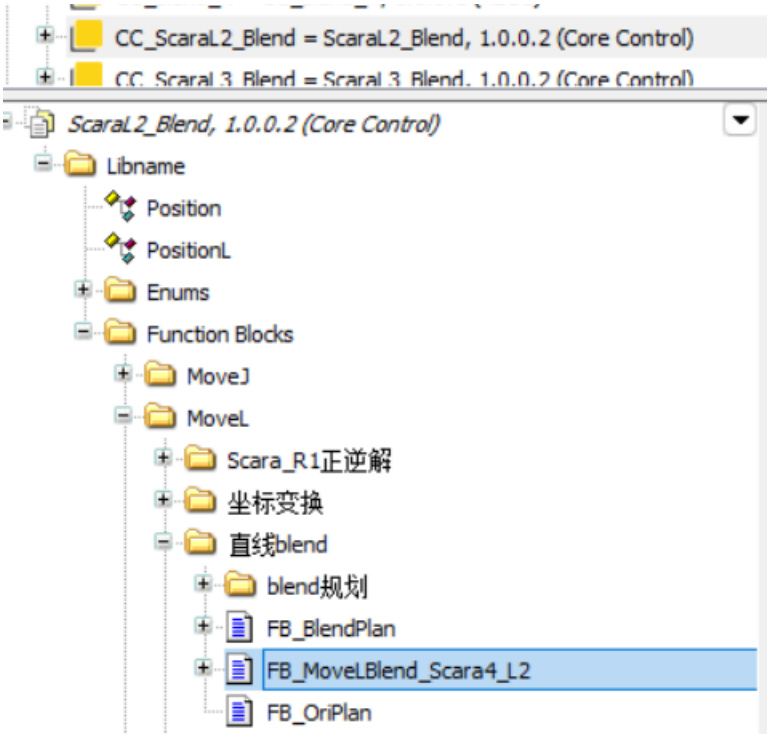


图 16

2.2.5.4 Scara_L3 机器人 Blend 库

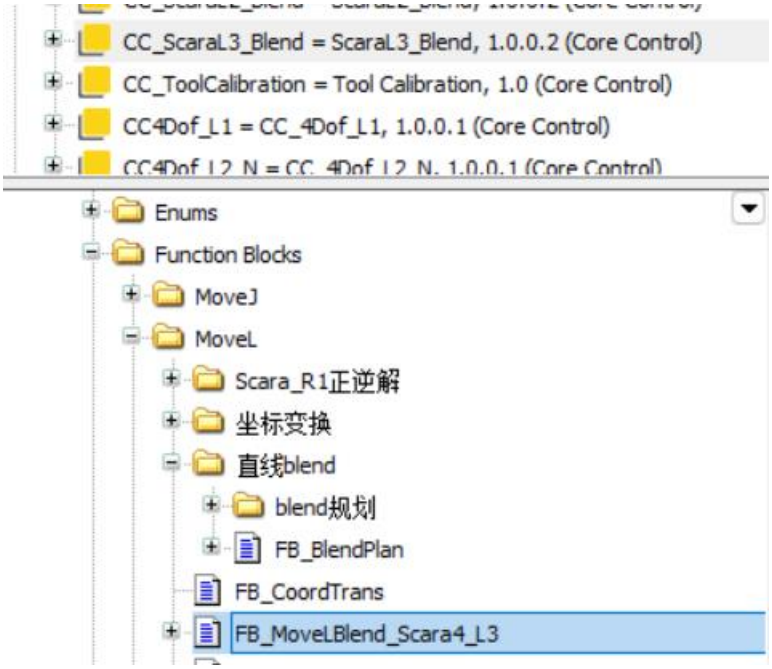


图 17

2.2.5.5 Scara_L1 模型库

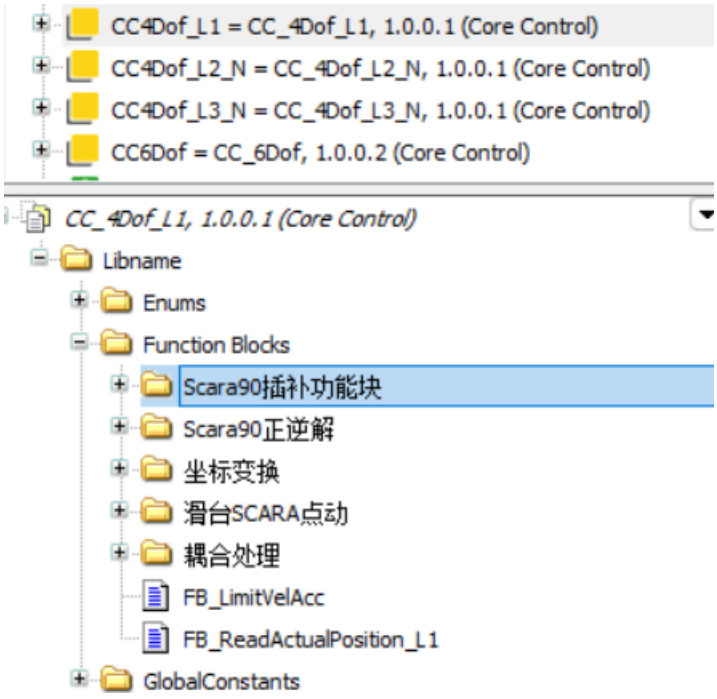


图 18

包含 L1 模型的圆弧、PTP、直线运动控制块。

2.2.5.6 Scara_L2 模型库

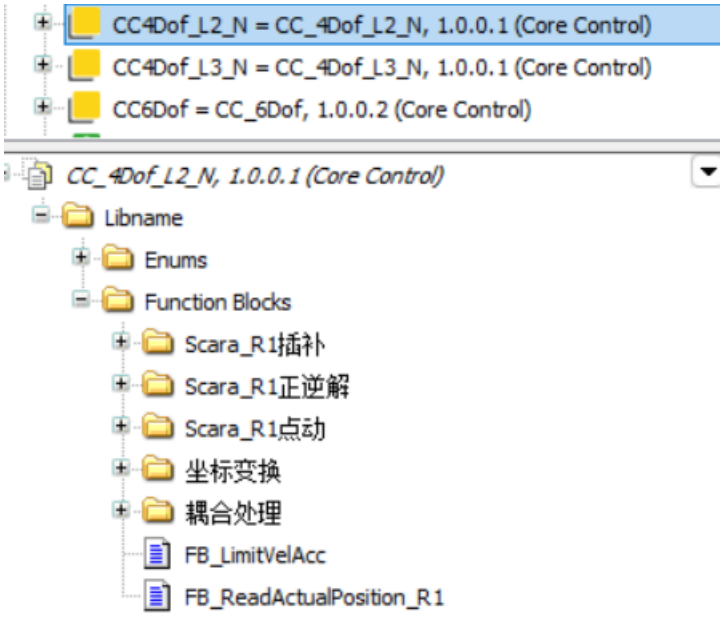


图 19

包含 L2 模型的圆弧、PTP、直线运动控制块。

2.2.5.7 Scara_L3 模型库

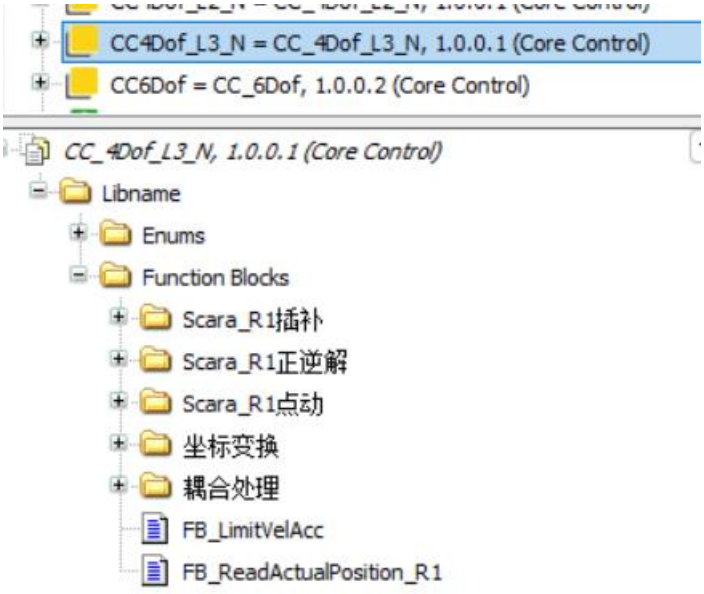


图 20

包含 L3 模型的圆弧、PTP、直线运动控制块。

2.2.5.8 6 轴模型库

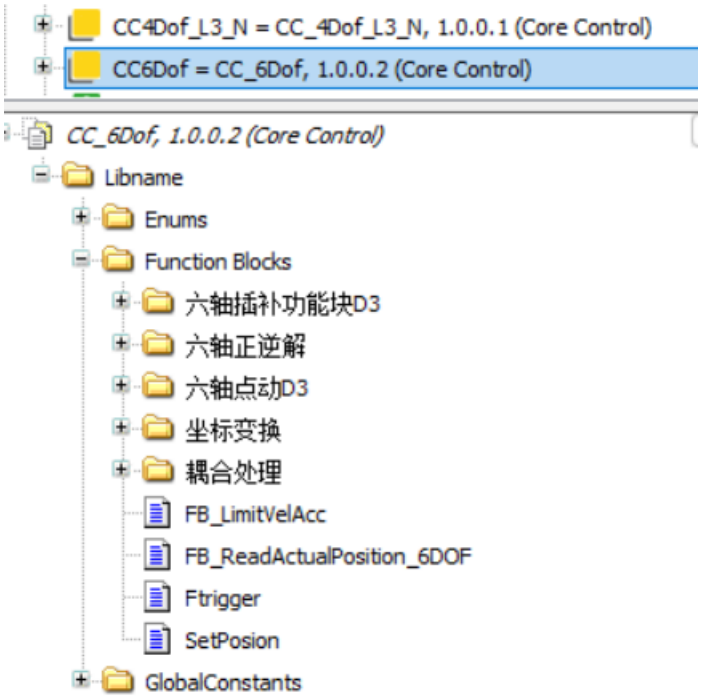


图 21

包含六轴机器人模型的圆弧、PTP、直线运动控制块。

2.2.5.9 坐标系标定库

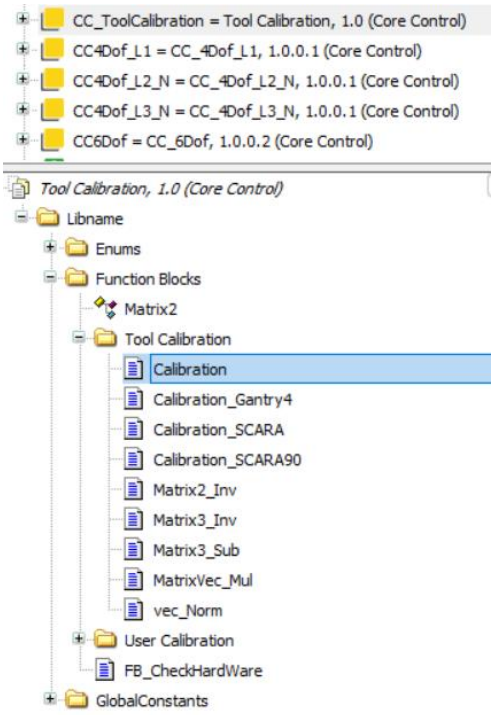


图 22

包含工具坐标系标定功能块等。

2.3 创建一个简单的运动控制工程

2.3.1 创建新工程

运行编程软件 CorecontrolIDE，新建一个用户工程，在下面的画面中

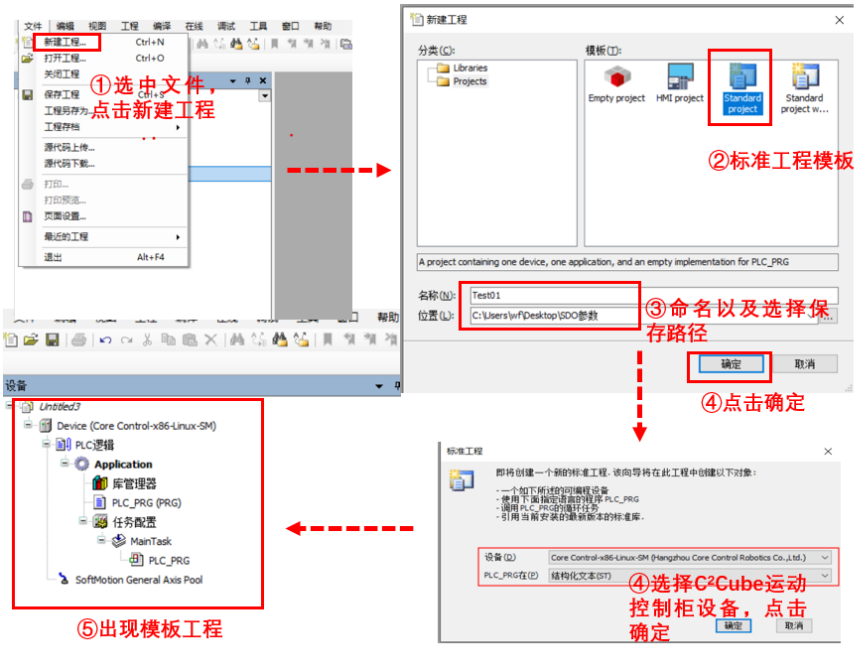


图 23

2.3.2 扫描硬件设备添加实轴

根据 2.2.2 Corecontrol 控制器与伺服以及 IO 连接，扫描出硬件设备，添加实轴

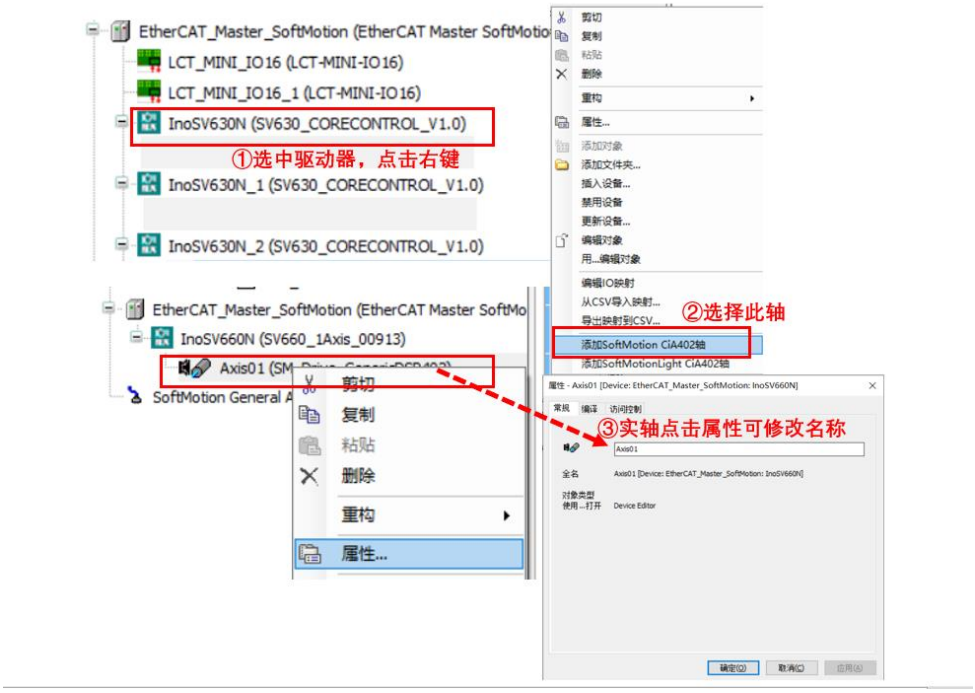


图 24

2.3.3 电机参数设置

为了精确地控制运动位置，控制器必需准确计算伺服电机的位置，根据应用系统的运转特性、行程特点，选择“轴类型与限制”，以便控制器内部对读电机编码器反馈信息进行计算，得到准确位置，避免编码器脉冲数累积溢出造成的错误。

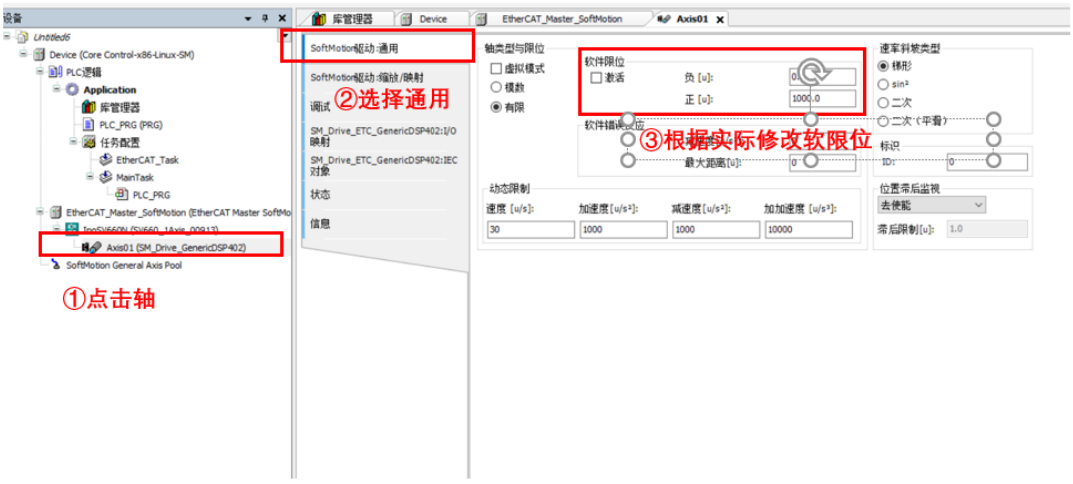


图 25

例如普通增量式编码器为 20bit 分辨率，即每圈有 1048576 个脉冲数，伺服电机经过变比为 10:1 机械减速机构后，驱动导程为 6.2mm 的丝杆（即丝杆转动 1 圈，丝杆滑块运动 6.2mm）运动，设定如下图：

图 26

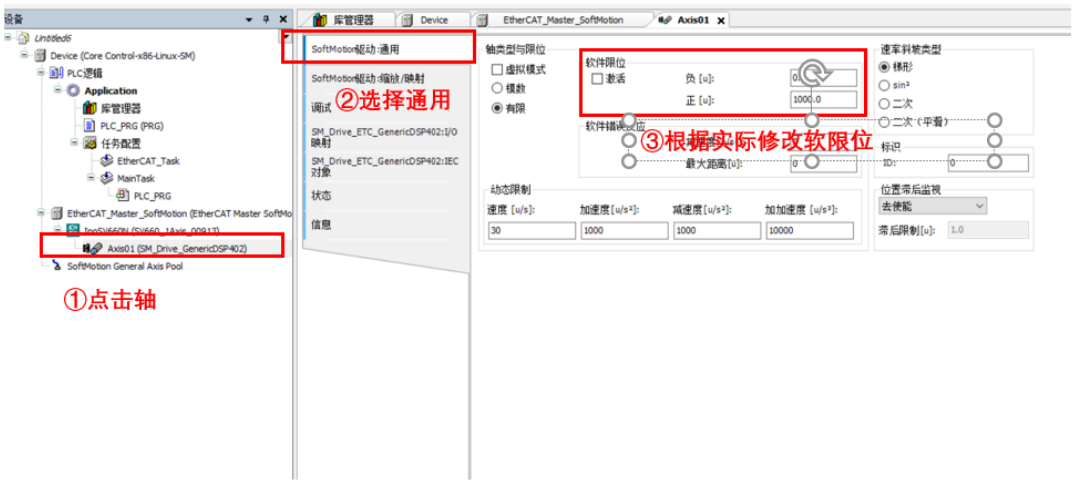


图 27

2.3.4 编写运动控制程序以及 IO 映射

运动控制程序编写，首先设置 EtherCAT 的任务优先级为 1，循环任务，间隔时间 4ms

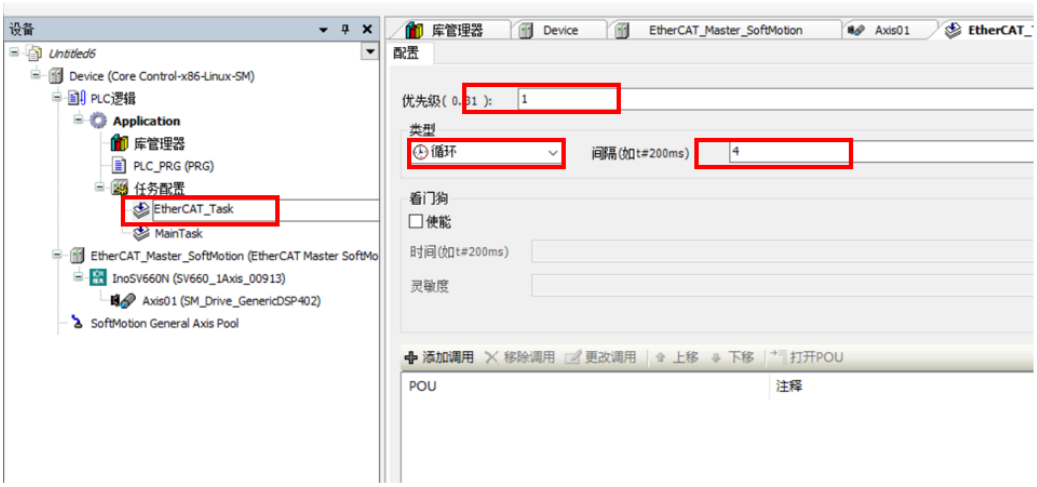


图 28

例：在声明区域声明 MC_Power、MC_MoveAbsolute、bStart。在代码编写区域对代码进行编写，将此 POU 加载到 EtherCAT 的任务中。

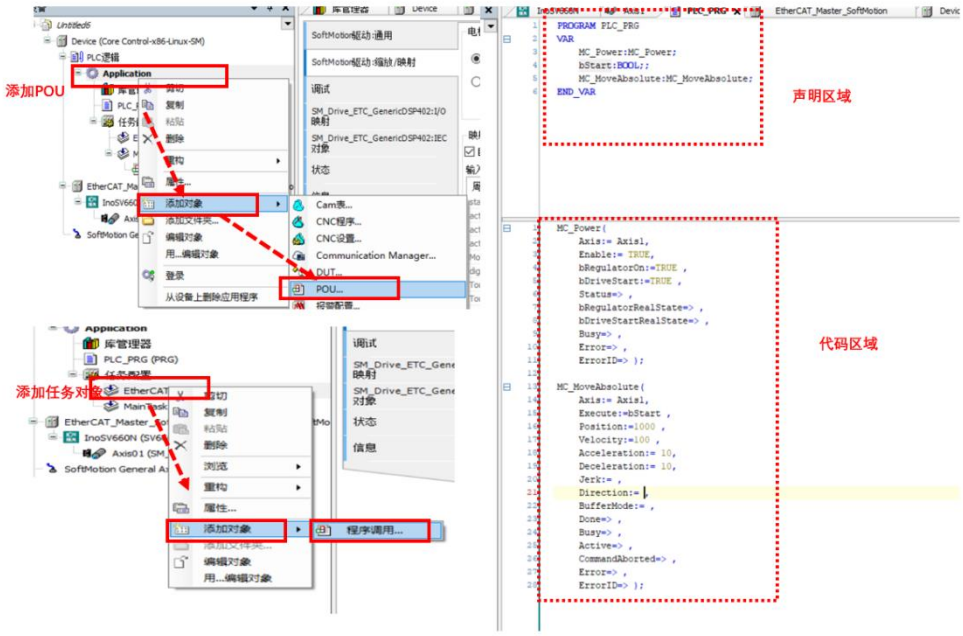


图 29

在变量区域声明变量，将变量与硬件 IO 映射，

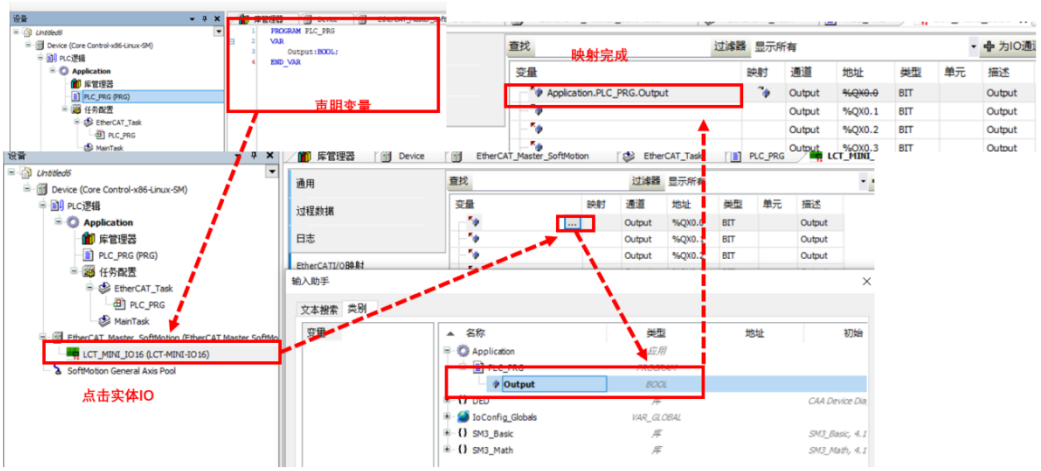


图 30

2.3.5 用户程序编辑排查错误

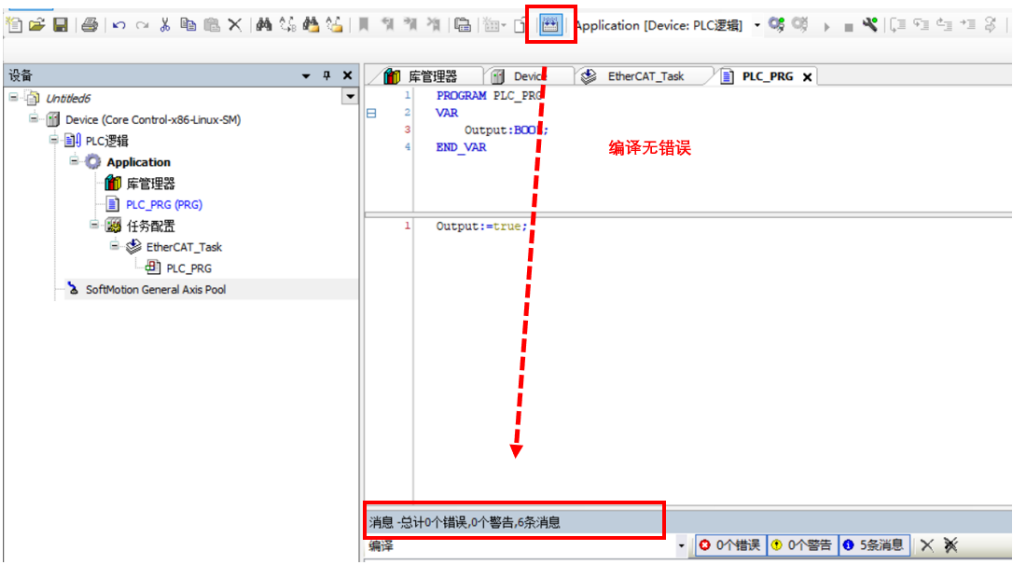


图 31

若有编写错误，上图中会列出错误类型与原因，双击其中的错误描述，光标会跳转到对应的程序编辑窗口，便于修订；逐一处理后，再进行编译，直到所有编译问题排除。最后将用户程序下载

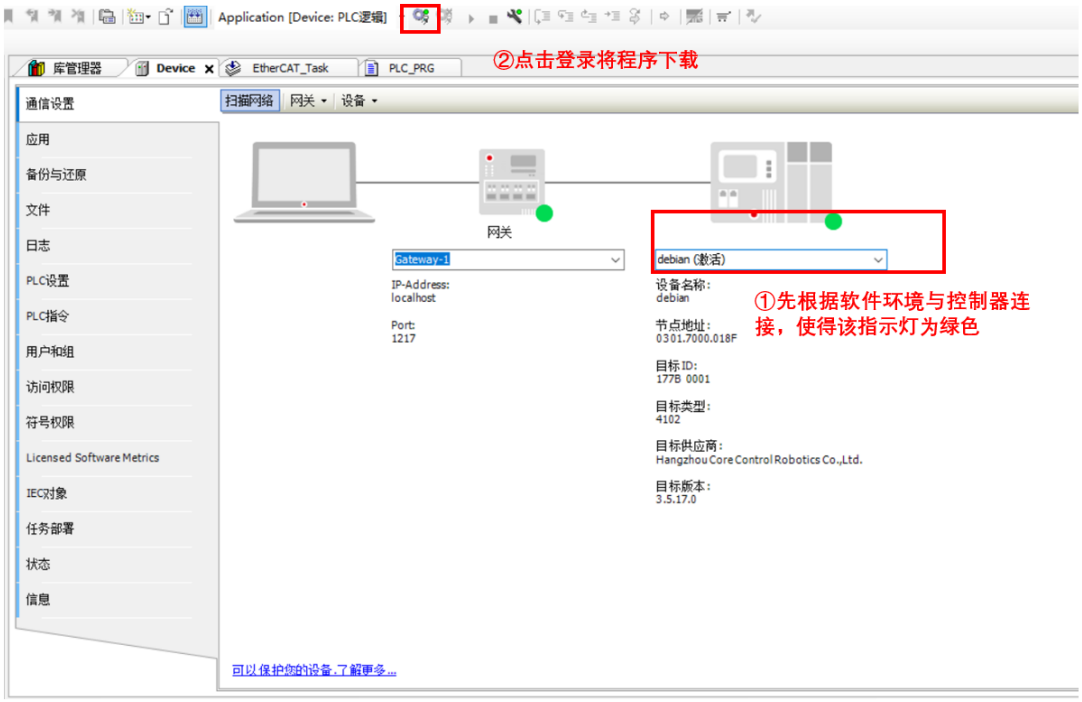


图 32

2.3.6 监控用户程序的运行

在监控画面，可以直观看到程序在执行情况

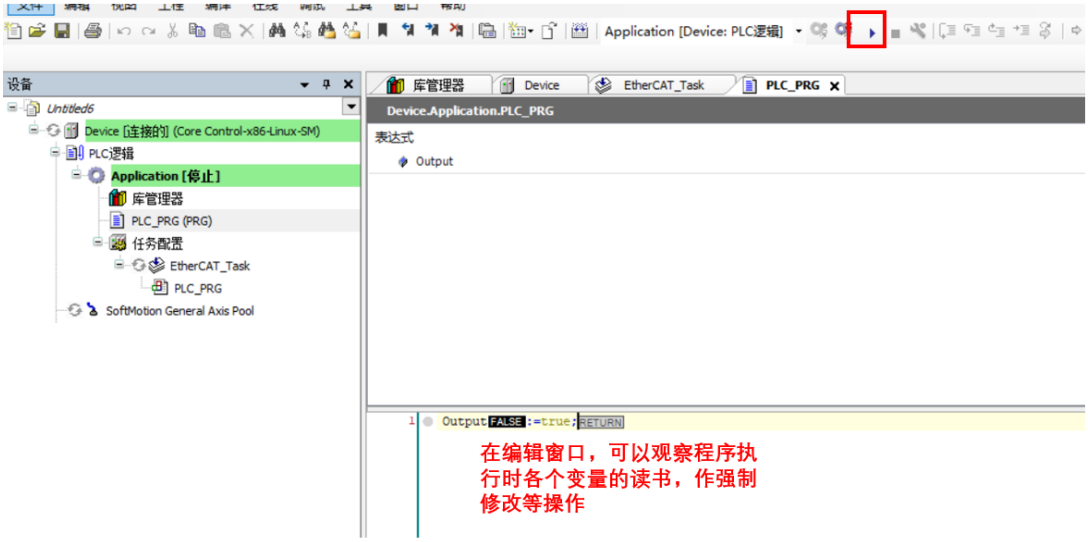


图 33

2.3.7 虚轴设置方式

在无实轴时，仍需要轴进行调试的时候，可以用虚轴进行调试，虚轴的设置方法如下：

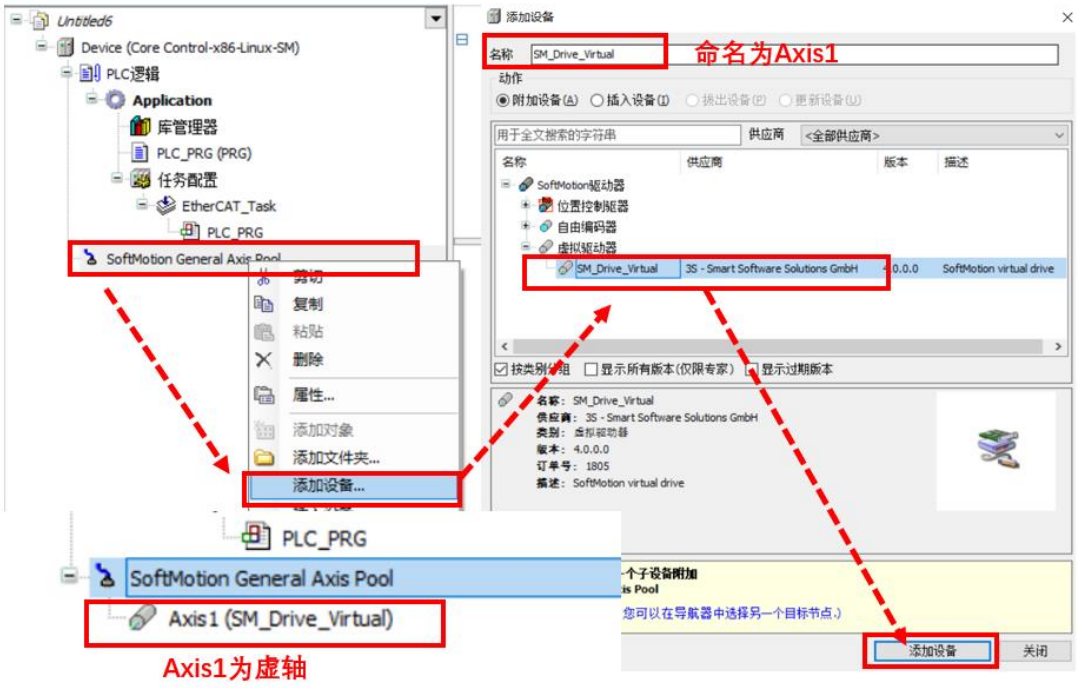


图 34

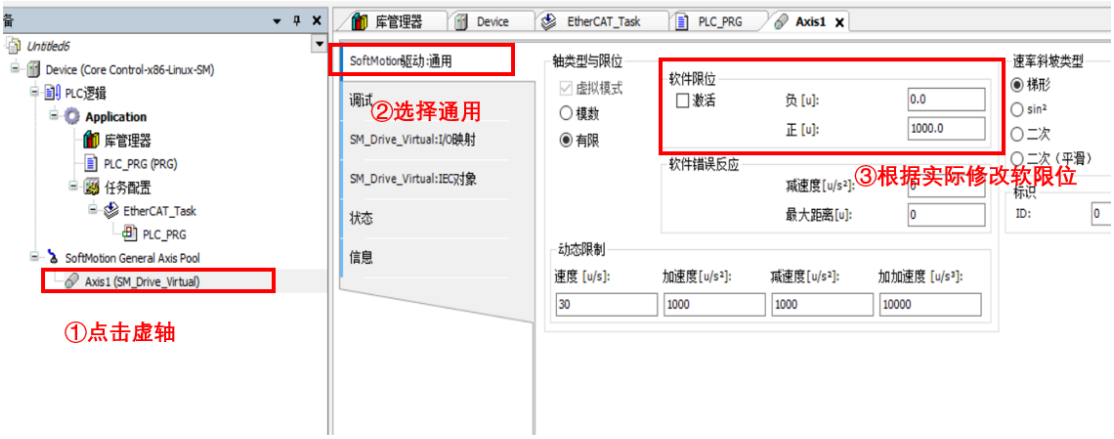


图 35

2.3.8 机器人运动学模型设置

在控制各伺服轴做机器人式协同动作时，需要配置相关机器人运动学模型或使用轴组模型控制。

C²Cube 运动控制器算法库已集成 3/4 轴 scara、6 轴机器人运行学模型于相应示教功能块，配置相应功能块即实现相应模型配置。

2-4 轴悬臂式/龙门式机器人采用轴组模型控制，相关设置详见 3.2.2-4 机器人示教功能块配置说明。

同时我们也可以添加自定义机器人模型，具体操作如下：

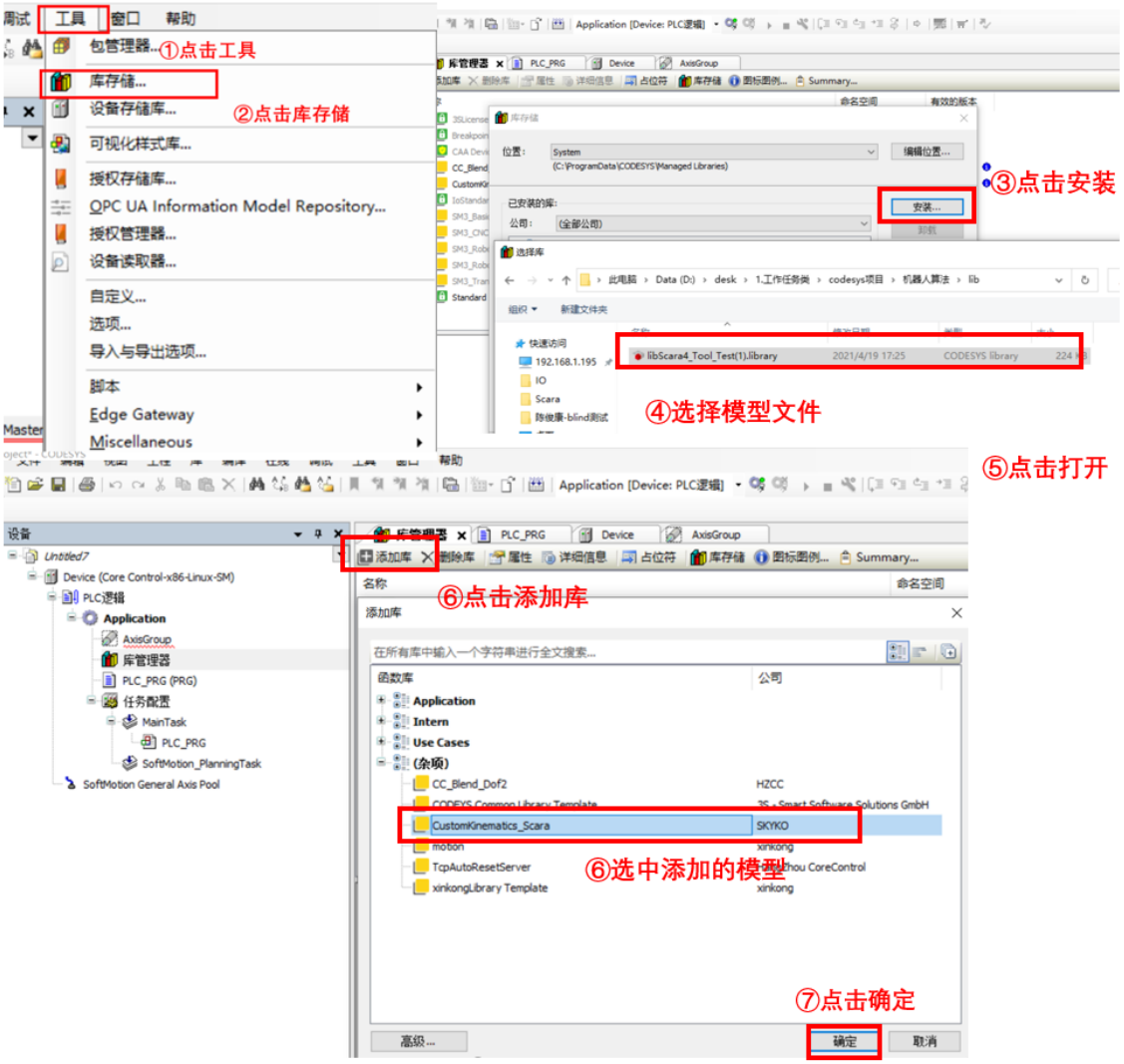


图 36

2.3.9 功能块调用方式

功能块调用方式分两种，可以于 CFC 顺序功能图中拖拽调用，也可以在 ST 结构语言中声明调用，两者效果一致。

本控制器中 CFC 顺序功能图、ST 结构语言编程方式与常用方式一致，以下仅做简单介绍。

以 6 轴示教功能块为例说明：

CFC 语言调用：如下图所示。

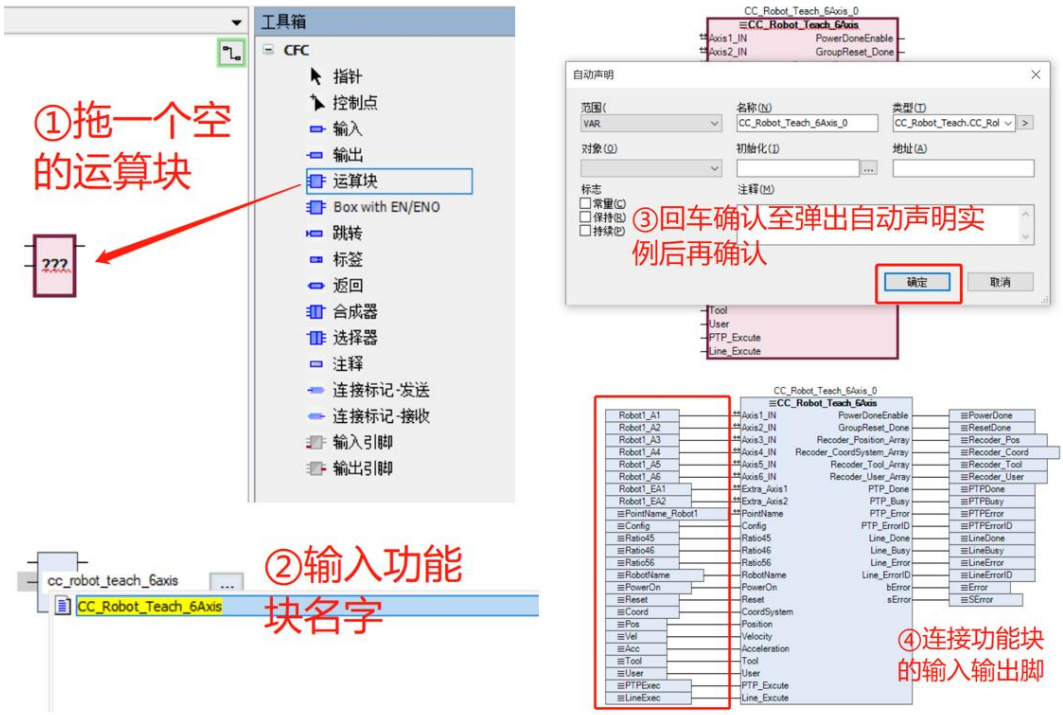


图 37

ST 语言调用：如下图所示。

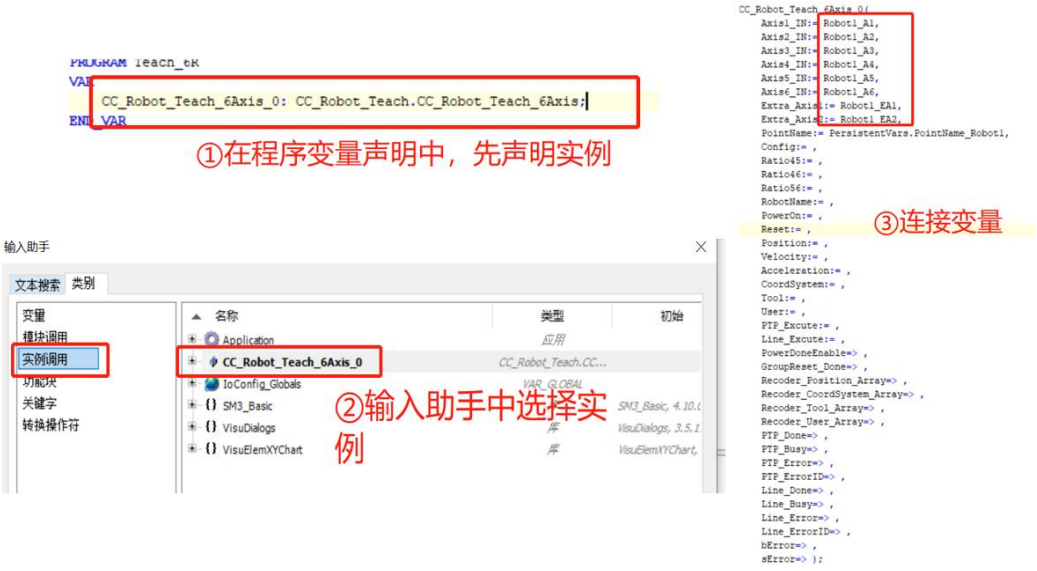


图 38

功能块的引脚较多，推荐使用 CFC 语言会方便。

三、机器人示教库添加及使用

3.1 机器人示教库简介

控制器通过”CC_Robot_Teach”机器人示教库，实现对 2-4 轴龙门/悬臂式模组机器人（轴组控制）、3-4 轴 SCARA 机器人（运动学模型控制）、6 轴标准工业机器人（运动学模型控制）的示教控制。配合示教界面使用，实现包括机器人单轴点动、单轴绝对、单轴相对、多轴绝对、多轴相对和点到点的点位动作示教方式、点位总表管理、限位设置及零点标定、工具坐标系标定、用户坐标系标定等功能。

3.2 示教库功能块介绍及添加说明

3.2.1 3 轴龙门/悬臂式模组机器人示教功能块

1) 指令形式

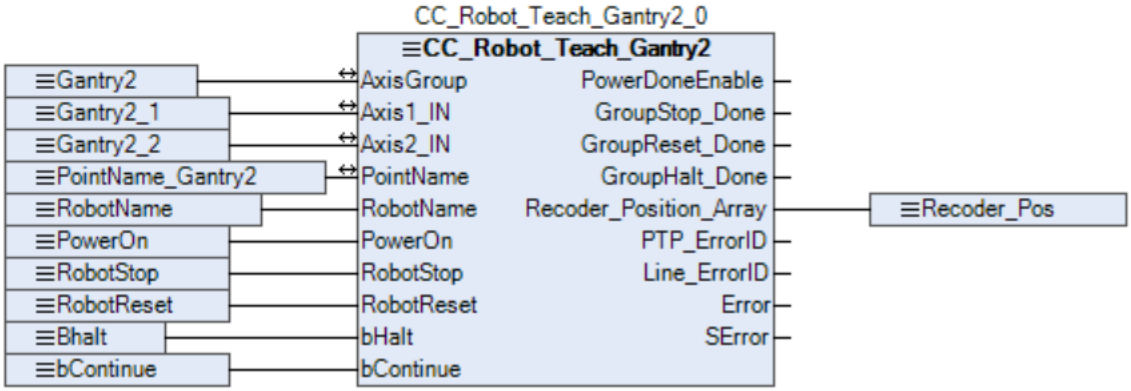


图 39

2) 相关变量说明

	类型	默认值	描述
输入输出			
AxisGroup	AXIS_GROUP_REF_SM3	————	机器人轴组
Axis1_IN	AXIS_REF_SM3	————	机器人第一轴
Axis2_IN	AXIS_REF_SM3	————	机器人第二轴
PointName	ARRAY[0..100] OF WSTRING（15）	————	机器人示教点位别名

输入			
RobotName	STRING	————————'	机器人注释名
PowerOn	BOOL	FALSE	程序模式：TRUE:使能； FALSE:下使能
RobotStop	BOOL	FALSE	程序模式：机器人停止
RobotReset	BOOL	FALSE	程序模式:机器人报错复位
bHalt	BOOL	FALSE	程序模式：机器人运动暂停
bContinue	BOOL	FALSE	程序模式：机器人继续运动
输出			
PowerDoneEnable	BOOL	FALSE	机器人已使能
GroupStop_Done	BOOL	FALSE	停止完成
GroupReset_Done	BOOL	FALSE	复位完成
Recoder_Position_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————————	示教点位的坐标值
PTP_ErrorID	SMC_ERROR	0	点到点运动错误代码
Line_ErrorID	SMC_ERROR	0	直线运动错误代码
Error	BOOL	FALSE	功能块报错
SError	WSTRING	“”	功能块报错描述

3) 功能说明

RobotName 不能和其他的机器人示教块重合；

暂停 bHalt 由上边沿触发；

继续 bContinue 由上边沿触发；

4) 2 轴机器人轴组参数配置

2 轴机器人运动控制通过轴组实现，使用前需进行轴组配置，配置方式如下：

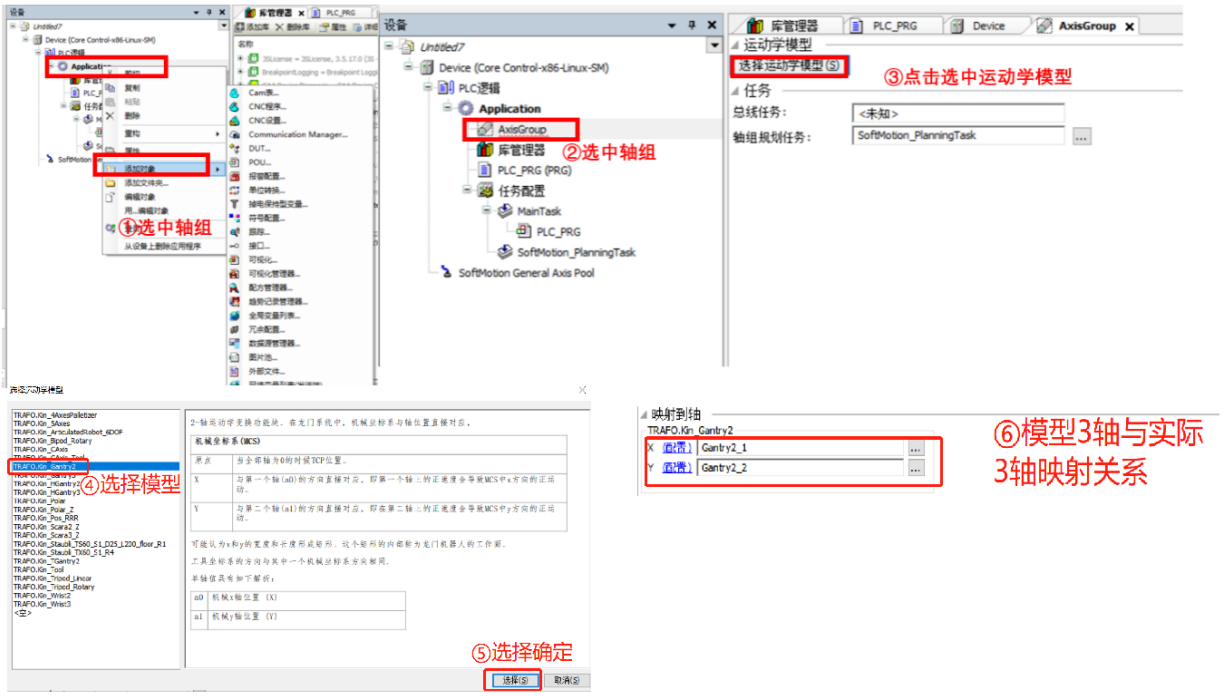


图 40

3.2.2 3 轴龙门/悬臂式模组机器人示教功能块

1) 指令形式

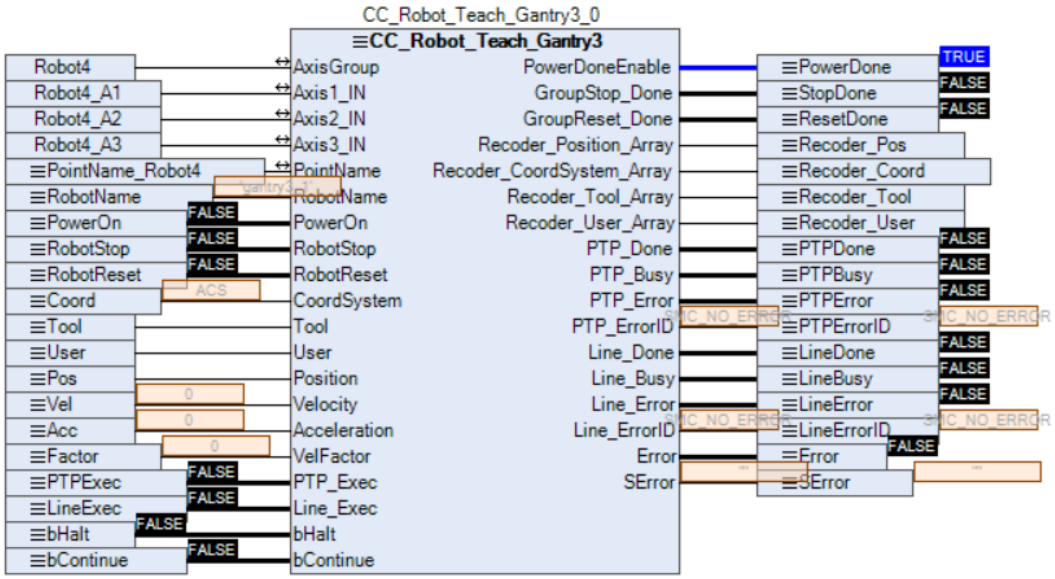


图 41

2) 相关变量说明

	类型	默认值	描述
输入输出			
AxisGroup	AXIS_GROUP_REF_SM3	————	机器人轴组
Axis1_IN	AXIS_REF_SM3	————	机器人第一轴
Axis2_IN	AXIS_REF_SM3	————	机器人第二轴
Axis3_IN	AXIS_REF_SM3	————	机器人第三轴
PointName	ARRAY[0..100] OF WSTRING（15）	————	机器人示教点位别名
输入			
RobotName	STRING	————'	机器人注释名
PowerOn	BOOL	FALSE	程序模式：TRUE:使能；FALSE:下使能
RobotStop	BOOL	FALSE	程序模式：机器人停止
RobotReset	BOOL	FALSE	程序模式:机器人报错复位
CoordSystem	SMC_COORD_SYSTEM	ACS	程序模式：运动坐标系
Tool	MC_COORD_REF	————	程序模式：工具坐标系
User	MC_COORD_REF	————	程序模式：用户坐标系
Position	SMC_POS_REF	————	程序模式：运动目标点坐标
Velocity	LREAL	0	程序模式：速度
Acceleration	LREAL	0	程序模式：加速度
VelFactor	LREAL	0	程序模式：速度因子
PTP_Exec	BOOL	FALSE	程序模式：点到点运行启动
Line_Exec	BOOL	FALSE	程序模式：直线运行启动
bHalt	BOOL	FALSE	程序模式：机器人运动暂停
bContinue	BOOL	FALSE	程序模式：机器人继续运动
输出			
PowerDoneEnable	BOOL	FALSE	机器人已使能
GroupStop_Done	BOOL	FALSE	停止完成
GroupReset_Done	BOOL	FALSE	复位完成
Recoder_Position_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	示教点位的坐标值
Recoder_CoordSystem_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_R	————	示教点位对应的坐标系

	EF		
Recoder_Tool_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	工具坐标系
Recoder_User_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	用户坐标系
PTP_Done	BOOL	FALSE	点到点运动完成
PTP_Busy	BOOL	FALSE	点到点运动未完成
PTP_Error	BOOL	FALSE	点到点运动错误
PTP_ErrorID	SMC_ERROR	0	点到点运动错误代码
Line_Done	BOOL	FALSE	直线运动完成
Line_Busy	BOOL	FALSE	直线运动未完成
Line_Error	BOOL	FALSE	直线运动错误
Line_ErrorID	SMC_ERROR	0	直线运动错误代码
Error	BOOL	FALSE	功能块报错
SError	WSTRING	“”	功能块报错描述

3) 功能块注意事项

RobotName 不能和其他的机器人示教块重合；

PTP 运动速度由 VelFactor 这输入参数控制；

Line 运动速度由 Velocity，Accelerate，VelFactor 这几个参数共同控制；

暂停 bHalt 由上边沿触发；

继续 bContinue 由上边沿触发；

4) 3 轴机器人轴组参数配置

3 轴机器人运动控制通过轴组实现，使用前需进行轴组配置，配置方式如下：

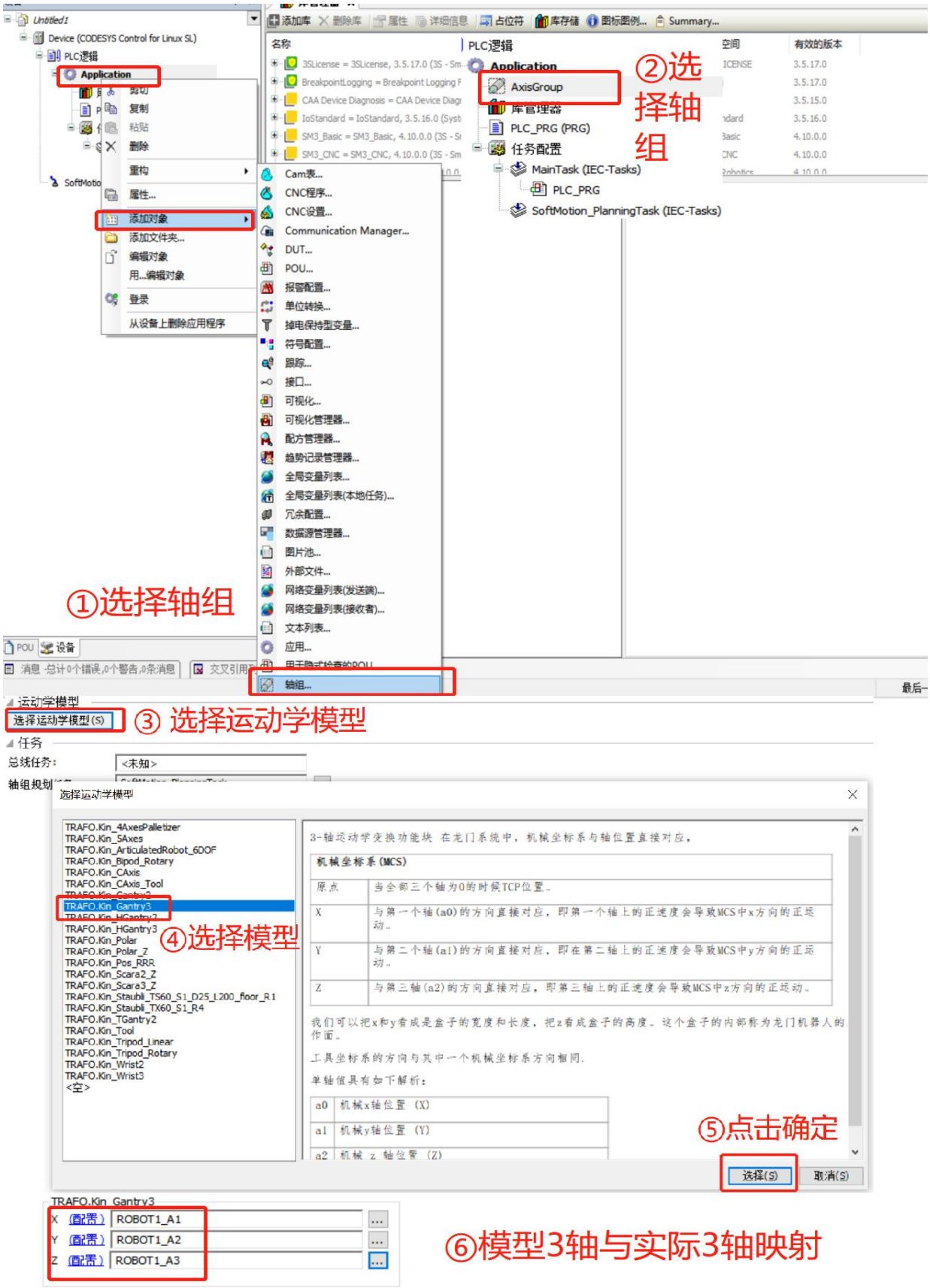

①选择轴组
②选择轴组
③ 选择运动学模型
④选择模型
⑤点击确定
⑥模型3轴与实际3轴映射
选择运动学模型(S)
选择运动学模型
TRAFO.Kin_Gantry3
选择(S)
取消(S)
ROBOT1_A1
ROBOT1_A2
ROBOT1_A3

图 42

3.2.3 4 轴龙门/悬臂式模组机器人示教功能块

1) 指令形式

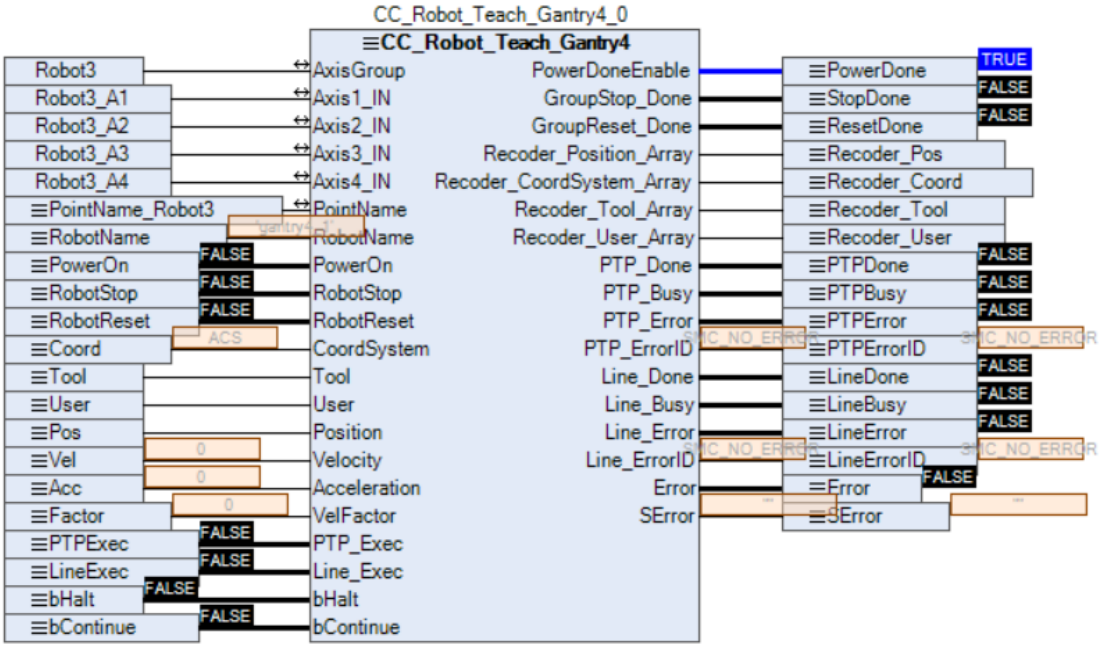


图 43

2) 相关变量说明

	类型	默认值	描述
输入输出			
AxisGroup	AXIS_GROUP_REF_SM3	————	机器人轴组
Axis1_IN	AXIS_REF_SM3	————	机器人第一轴
Axis2_IN	AXIS_REF_SM3	————	机器人第二轴
Axis3_IN	AXIS_REF_SM3	————	机器人第三轴
Axis4_IN	AXIS_REF_SM3	————	机器人第四轴
PointName	ARRAY[0..100] OF WSTRING（15）	————	机器人示教点位别名
输入			
RobotName	STRING	————'	机器人注释名
PowerOn	BOOL	FALSE	程序模式：TRUE:使能；FALSE:下使能
RobotStop	BOOL	FALSE	程序模式：机器人停止
RobotReset	BOOL	FALSE	程序模式:机器人报错复位
CoordSystem	SMC_COORD_SYSTEM	ACS	程序模式：运动坐标系
Tool	MC_COORD_REF	————	程序模式：工具坐标系
User	MC_COORD_REF	————	程序模式：用户坐标系

Position	SMC_POS_REF	————	程序模式：运动目标点坐标
Velocity	LREAL	0	程序模式：速度
Acceleration	LREAL	0	程序模式：加速度
VelFactor	LREAL	0	程序模式：速度因子
PTP_Exec	BOOL	FALSE	程序模式：点到点运行启动
Line_Exec	BOOL	FALSE	程序模式：直线运行启动
bHalt	BOOL	FALSE	程序模式：机器人运动暂停
bContinue	BOOL	FALSE	程序模式：机器人继续运动
输出			
PowerDoneEnable	BOOL	FALSE	机器人已使能
GroupStop_Done	BOOL	FALSE	停止完成
GroupReset_Done	BOOL	FALSE	复位完成
Recoder_Position_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	示教点位的坐标值
Recoder_CoordSystem_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	示教点位对应的坐标系
Recoder_Tool_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	工具坐标系
Recoder_User_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	用户坐标系
PTP_Done	BOOL	FALSE	点到点运动完成
PTP_Busy	BOOL	FALSE	点到点运动未完成
PTP_Error	BOOL	FALSE	点到点运动错误
PTP_ErrorID	SMC_ERROR	0	点到点运动错误代码
Line_Done	BOOL	FALSE	直线运动完成
Line_Busy	BOOL	FALSE	直线运动未完成
Line_Error	BOOL	FALSE	直线运动错误
Line_ErrorID	SMC_ERROR	0	直线运动错误代码
Error	BOOL	FALSE	功能块报错
SError	WSTRING	“”	功能块报错描述

3) 功能块注意事项

RobotName 不能和其他的机器人示教块重合；

PTP 运动速度由 VelFactor 这输入参数控制；

Line 运动速度由 Velocity，Accelerate，VelFactor 这几个参数共同控制；

暂停 bHalt 由上边沿触发；

继续 bContinue 由上边沿触发；

4) 4 轴机器人轴组参数配置

4 轴机器人运动控制通过轴组实现，使用前需进行轴组配置，配置方式如下：

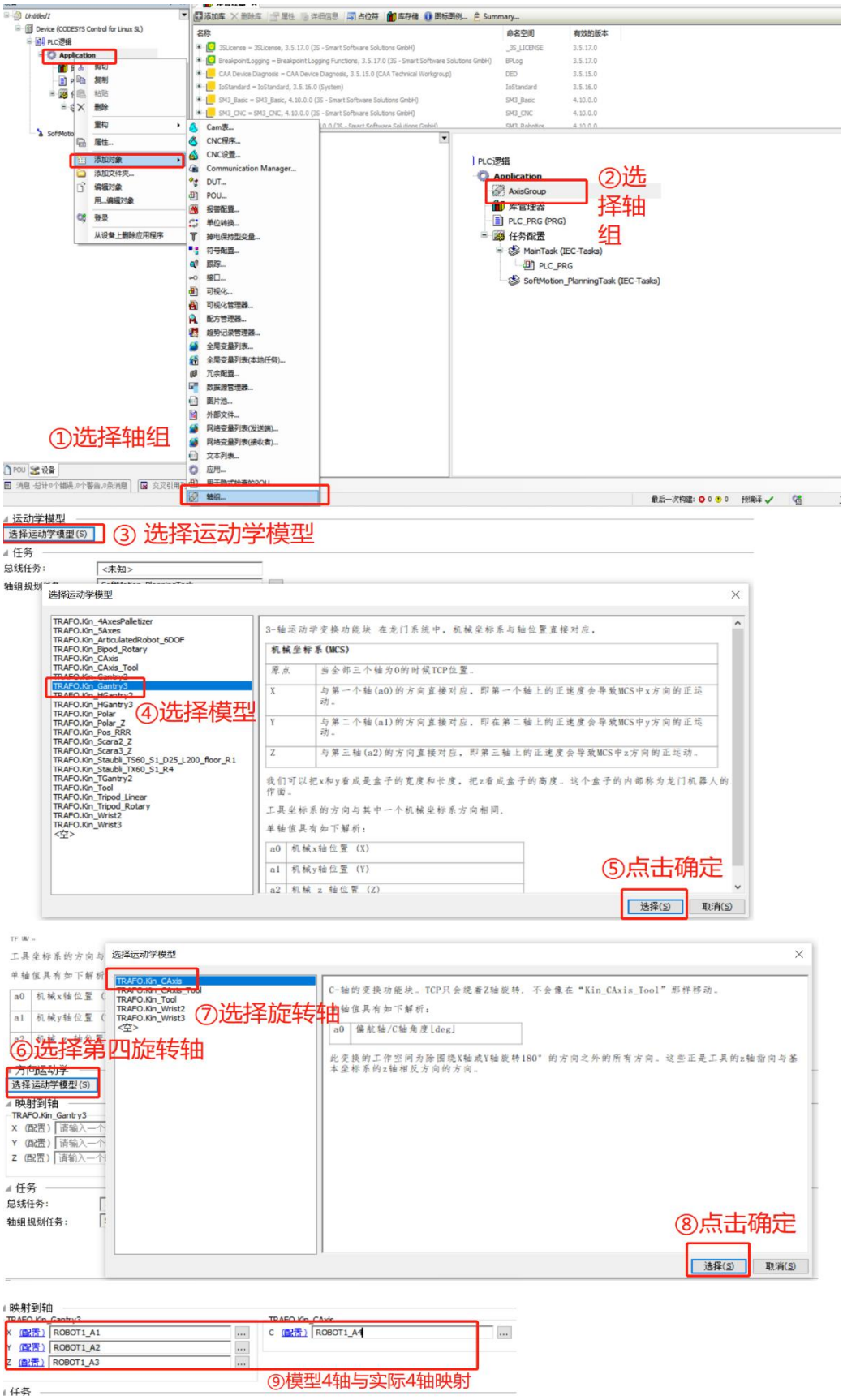

①选择轴组
②选择轴组
③ 选择运动学模型
④选择模型
⑤点击确定
⑥选择第四旋转轴
⑦选择旋转轴
⑧点击确定
⑨模型4轴与实际4轴映射

图 44

3.2.4 4 轴 SCARA 机器人示教功能块

1) 指令形式

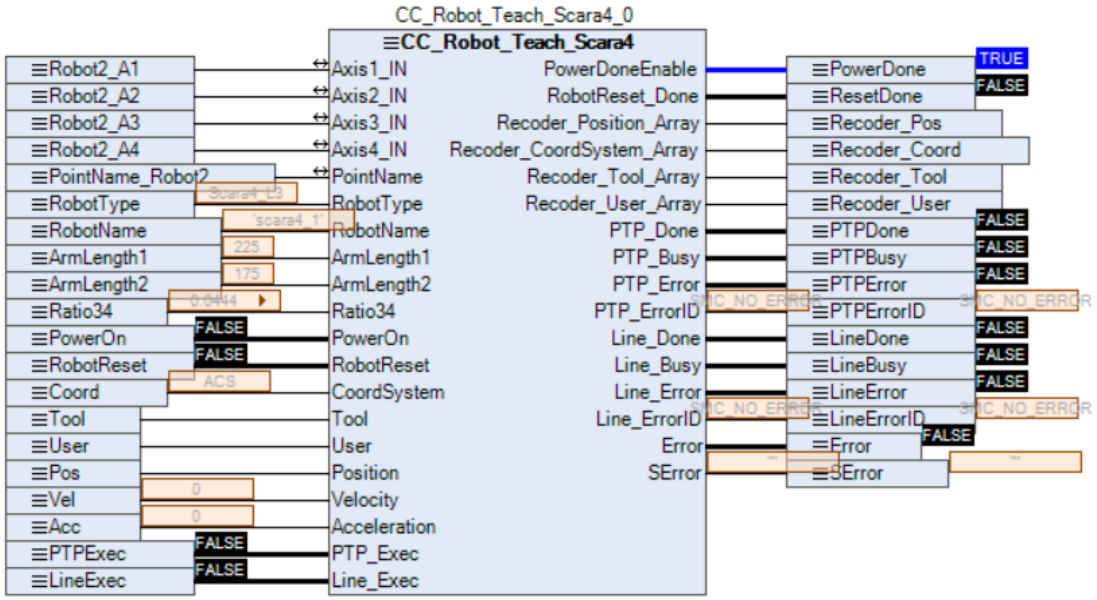


图 45

2) 相关变量说明

	类型	默认值	描述
输入输出			
Axis1_IN	AXIS_REF_SM3	————	机器人第一轴
Axis2_IN	AXIS_REF_SM3	————	机器人第二轴
Axis3_IN	AXIS_REF_SM3	————	机器人第三轴
Axis4_IN	AXIS_REF_SM3	————	机器人第四轴
PointName	ARRAY[0..100] OF WSTRING（15）	————	机器人示教点位别名
输入			
RobotType	RobotType	0	机器人类型：1：1 轴 2 轴为线性轴的机器人；2:2 轴为线性轴的机器人；3:3 轴为线性轴的机器人
RobotName	STRING	————	机器人注释名
ArmLength1	LREAL	0	机器人关节 1 长度
ArmLength2	LREAL	0	机器人关节 2 长度
Ratio34	LREAL	0	机器人三四轴耦合比
PowerOn	BOOL	FALSE	程序模式：TRUE:使能；FALSE:下使能

RobotReset	BOOL	FALSE	程序模式:机器人报错复位
CoordSystem	SMC_COORD_SYSTEM	ACS	程序模式：运动坐标系
Tool	MC_COORD_REF	————	程序模式：工具坐标系
User	MC_COORD_REF	————	程序模式：用户坐标系
Position	SMC_POS_REF	————	程序模式：运动目标点坐标
Velocity	LREAL	0	程序模式：速度
Acceleration	LREAL	0	程序模式：加速度
PTP_Exec	BOOL	FALSE	程序模式：点到点运行启动
Line_Exec	BOOL	FALSE	程序模式：直线运行启动
输出			
PowerDoneEnable	BOOL	FALSE	机器人已使能
RobotReset_Done	BOOL	FALSE	复位完成
Recoder_Position_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	示教点位的坐标值
Recoder_CoordSystem_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	示教点位对应的坐标系
Recoder_Tool_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	工具坐标系
Recoder_User_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	用户坐标系
PTP_Done	BOOL	FALSE	点到点运动完成
PTP_Busy	BOOL	FALSE	点到点运动未完成
PTP_Error	BOOL	FALSE	点到点运动错误
PTP_ErrorID	SMC_ERROR	0	点到点运动错误代码
Line_Done	BOOL	FALSE	直线运动完成
Line_Busy	BOOL	FALSE	直线运动未完成
Line_Error	BOOL	FALSE	直线运动错误
Line_ErrorID	SMC_ERROR	0	直线运动错误代码
Error	BOOL	FALSE	功能块报错
SError	WSTRING	“”	功能块报错描述

3) 功能说明

参数配置：

RobotName 不能和其他的机器人示教块重合；

PTP 运动速度由 VelFactor 这输入参数控制；

Line 运动速度由 Velocity，Accelerate，VelFactor 这几个参数共同控制；

暂停 bHalt 由上边沿触发；

继续 bContinue 由上边沿触发；

耦合比参数：

Scara_L1 、Scara_L2：三轴关节（输出端）转 1°，带动四关节（输出端）转 n°或-n°，耦合比设为 n 或-n；

Scara_L3：四关节（输出端）转 1°，带动三关节（输出端）转 n°或-n°，耦合比设为 n 或-n；

上述耦合比默认 0 值，即设备无耦合结构；

3.2.5 6 轴标准工业机器人示教功能块

1) 指令形式

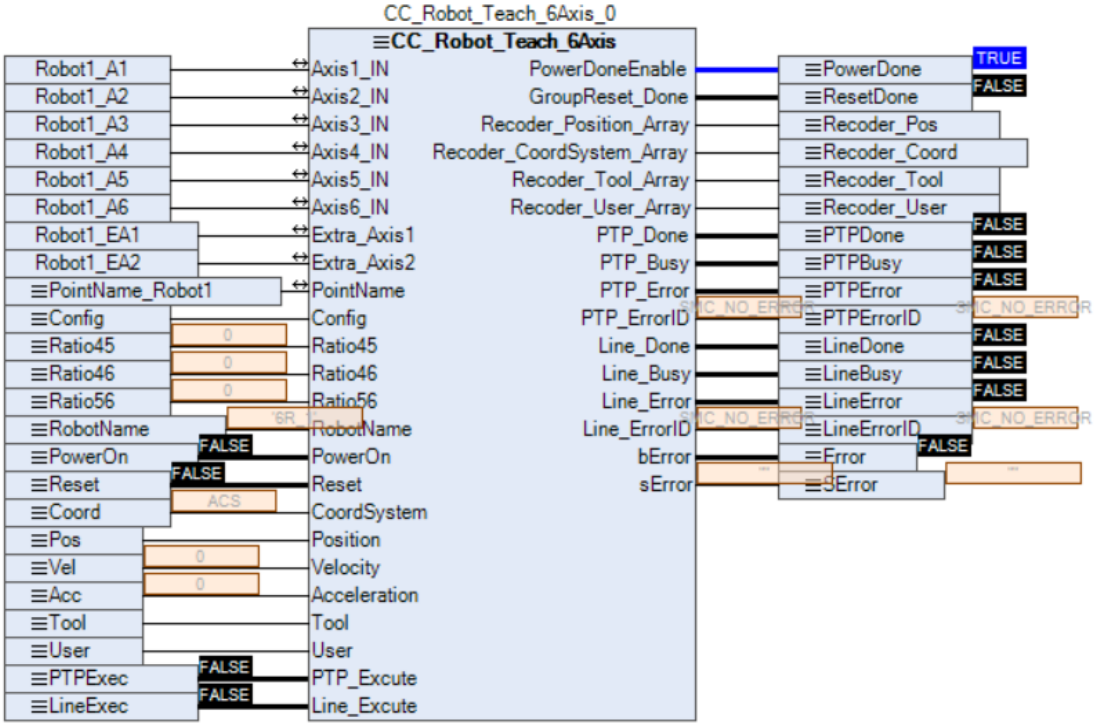


图 46

2) 相关变量说明

	类型	默认值	描述
输入输出			
Axis1_IN	AXIS_REF_SM3	————	机器人第一轴
Axis2_IN	AXIS_REF_SM3	————	机器人第二轴
Axis3_IN	AXIS_REF_SM3	————	机器人第三轴
Axis4_IN	AXIS_REF_SM3	————	机器人第四轴
Axis5_IN	AXIS_REF_SM3	————	机器人第五轴
Axis6_IN	AXIS_REF_SM3	————	机器人第六轴
Extra_Axis1	AXIS_REF_SM3	————	机器人外部一轴
Extra_Axis2	AXIS_REF_SM3	————	机器人外部二轴
PointName	ARRAY[0..100] OF WSTRING（15）	————	机器人示教点位别名
输入			
Config	SMC_TrafoConfig_ArticulatedRobot_3DOF	————	机器人 DH 参数
Ratio45	LREAL	0	机器人四五轴耦合比
Ratio46	LREAL	0	机器人四六轴耦合比
Ratio56	LREAL	0	机器人五六轴耦合比
RobotName	STRING	————	机器人注释名
PowerOn	BOOL	FALSE	程序模式：TRUE:使能；FALSE:下使能
Reset	BOOL	FALSE	程序模式:机器人报错复位
CoordSystem	SMC_COORD_SYSTEM	ACS	程序模式：运动坐标系
Tool	MC_COORD_REF	————	程序模式：工具坐标系
User	MC_COORD_REF	————	程序模式：用户坐标系
Position	SMC_POS_REF	————	程序模式：运动目标点坐标
Velocity	LREAL	0	程序模式：速度
Acceleration	LREAL	0	程序模式：加速度
PTP_Exec	BOOL	FALSE	程序模式：点到点运行启动
Line_Exec	BOOL	FALSE	程序模式：直线运行启动
输出			
PowerDoneEnable	BOOL	FALSE	机器人已使能
RobotReset_Done	BOOL	FALSE	复位完成

Recoder_Position_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	示教点位的坐标值
Recoder_CoordSystem_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	示教点位对应的坐标系
Recoder_Tool_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	工具坐标系
Recoder_User_Array	ARRAY [0..100] OF SM3_Robotics.SMC_POS_REF	————	用户坐标系
PTP_Done	BOOL	FALSE	点到点运动完成
PTP_Busy	BOOL	FALSE	点到点运动未完成
PTP_Error	BOOL	FALSE	点到点运动错误
PTP_ErrorID	SMC_ERROR	0	点到点运动错误代码
Line_Done	BOOL	FALSE	直线运动完成
Line_Busy	BOOL	FALSE	直线运动未完成
Line_Error	BOOL	FALSE	直线运动错误
Line_ErrorID	SMC_ERROR	0	直线运动错误代码
Error	BOOL	FALSE	功能块报错
SError	WSTRING	“”	功能块报错描述

3) 功能块注意事项

RobotName 不能和其他的机器人示教块重合；

PTP 运动速度由 VelFactor 这输入参数控制；

Line 运动速度由 Velocity，Accelerate，VelFactor 这几个参数共同控制；

暂停 bHalt 由上边沿触发；

继续 bContinue 由上边沿触发；

耦合比参数定义：

四关节（输出端）转 1°，带动五关节（输出端）转 n°或-n°，耦合比 Ratio45 设为 n 或-n；

四关节（输出端）转 1°，带动六关节（输出端）转 n°或-n°，耦合比 Ratio46 设为 n 或-n；

五关节（输出端）转 1°，带动六关节（输出端）转 n°或-n°，耦合比 Ratio56 设

为 n 或-n；

正负方向有是否同向旋转决定啊，同向旋转为正，反之为负；

上述耦合比默认 0 值，即设备无耦合结构；

3.3 示教界面使用说明

3.3.1 整体界面展示

所有机器人示教界面布局一致，仅显示轴数存在差异，功能一致。

所有涉及机器人控制模型的相关算法需要进行使用，必须配置相应机器人类型的示教库功能块。

完成相应示教功能块后，可在 CORECONTROL 可视化界面打开相应机器人示教操作界面，或通过 web 端(控制器 IP 地址：192.138.1.143:8080/仿真控制器 IP 地址：127.0.0.1：8080）打开窗口显示示教界面。

以 6 轴机器人示教界面为例展示如下：

图 47

3.3.2 状态栏按键介绍

状态栏如图所示：

图 48

从左到右依次介绍所示按钮：

1) 使能按键

用于功能块控制伺服轴上下使能，关断抱闸。选中状态下做浅色显示，未使能

状态为深色显示，手动或程序控制机器人动作前需选中。

2) 急停按键

用于紧急状态下下使能，停止机器人动作，配置时需关联伺服驱动器 STO 信号。故障解除后需解除急停并复位。

3) 暂停按键

用于正常生产状态下暂时停止机器人动作，不执行下使能动作，解除暂停后可从停止点继续执行当前指令动作。

4) 启动按键

用于解除机器人暂停，与暂停按键互锁，选中后清除暂停按键状态，使机器人从停止点继续执行当前指令动作。

5) 坐标系选择按键

用于切换机器人坐标系，辅助实现机器人各种轨迹运动通过下拉可选择：ACS-关节坐标系、MCS-基坐标系、TCS-工具坐标系、PCS-用户坐标系，选用工具坐标系或用户坐标系时，工具坐标或用户坐标需选择具体工具或用户坐标。

6) 工具坐标选择按键

用于配合切换机器人工具坐标系，选用工具坐标系时，配合选择切换不同工具坐标，共预留 8 项工具坐标选择项，工具坐标系需提标定。

7) 用户坐标选择按键

用于配合切换机器人用户坐标系，选用用户坐标系时，配合选择切换不同用户坐标，共预留 8 项用户坐标选择项，用户坐标系需提标定。

8) 速度设置按键

用于调整机器人手动示教速度，设置范围 0-100％，速度设置默认“1%”，点击窗口可手动输入数值，实际运动最高速度、加速度、加加速度为所有轴动态限制的最小值的百分比。

9) 通讯状态显示

用于显示 ETHCAT 总线通讯通断。

10) 系统版本

用于显示当前示教程序版本。

3.3.3 动作示教窗口功能

动作示教窗口实现功能包括：点位移动及记录、各轴零位设置，窗口布局如下：

图 49

“轴”显示窗口根据实时选择坐标系，以大写字母“A”加数字显示机器人关节轴，以“X/Y/Z”或“RX/RY/RZ”等显示笛卡尔坐标系轴及相应欧拉角。ACS 坐标系下，从上到下轴项分别对应机器人的六个轴，对应“+”按键轴按正方向旋转，“-”按键轴按负方向旋转。当坐标系切换为笛卡尔坐标系时，点动方向则变为沿当前直角坐标系的“X”、“Y”、“Z”方向直线运动和对应欧拉角方向的旋转运动。

“当前位置”窗口根据所选坐标系不同，实时显示当下机器人末端点的直角坐标值或各轴关节角度值。

3.3.3.1 移动示教

沿“MCS-基坐标”/“ACS-关节坐标系”执行下述 1-7 模式运动时，坐标系选择“MCS-基坐标”/“ACS-关节坐标系”，工具坐标及用户坐标选项可随意，速度设置默认“1%”。

沿“TCS-工具坐标系”/“PCS-用户坐标系”执行下述 1-7 模式运动时，坐标系选择“TCS-工具坐标系”/“PCS-用户坐标系”，工具坐标及用户坐标选项选定相应坐标，不得为无，速度设置默认“1%”。

具体运动控制如下：

1) 单轴点动

移动方式下拉窗口切换 “单轴点动”，动作示教窗口“JOG 使能”按键选中，点动相应轴项后“+”/“-”按键实现相应单轴点动动作，松开按键机器人停止移动。如下图所示

图 50

2) 单轴绝对移动

移动方式下拉窗口切换 “单轴绝对”，动作示教窗口“JOG 使能”按键自动屏蔽灰

化，在各轴 “目标位置”窗口输入目标点坐标，点动相应轴项后“+”按键实现相应单轴点动动作，到达指定点位或松开按键机器人停止移动。如下图所示：

图 51

3) 单轴相对运动

移动方式下拉窗口切换 “单轴相对”，动作示教窗口“JOG 使能”按键自动屏蔽灰化，在各轴 “移动距离”窗口输入移动距离，点动相应轴项后“+”按键，机器人相应单轴运动至“当前位置”加上对应“移动距离”的位置处，如移动过程中松开按键机器人停止移动，继续点动“+”按键，目标位置将更新为新“当前位置”加上对应“移动距离”的位置处。如下图所示：

图 52

4) 多轴绝对移动

移动方式下拉窗口切换 “多轴绝对”，动作示教窗口“JOG 使能”按键自动屏蔽灰化，在各轴“目标位置”窗口输入目标点坐标，点动“移动”按键实现相应多轴点动动作，到达指定点位或松开按键机器人停止移动。如下图所示：

图 53

5) 多轴相对运动

移动方式下拉窗口切换 “多轴相对”，动作示教窗口“JOG 使能”按键自动屏蔽灰化，在各轴 “移动距离”窗口输入移动距离，点动 “移动”按键，机器人整体运动至“当前位置”加上对应“移动距离”的位置处，如移动过程中松开按键机器人停止移动，继续点动“+”按键，目标位置将更新为新“当前位置”加上对应“移动距离”的位置处。如下图所示：

图 54

6) 点到点移动

移动方式下拉窗口切换 “点到点”，动作示教窗口“JOG 使能”按键自动屏蔽灰化，点位选择下拉窗口切换目标点位， “目标位置”窗口自动显示目标点坐标，点动“移动”按键实现相应多轴点动动作，到达指定点位或松开按键机器人停止移动。如下图所示：

图 55

7) 程序控制移动

程序模式是示教功能块开放接口给编程人员使用的运动模式，为防止程序控制机器人动作过程中修改点位，起锁定作用，启动程序前需将运动方式切换为“程序控制”。

3.3.3.2 点位记录

点击“点位记录”按键，记录当前坐标系下，机器人末端坐标值。已记录点于“点位总表”窗口显示，已记录点位坐标可经过重新示教覆盖或于“点位总表”窗口手动修改，详见“3.3.4-点位总表窗口功能”。

3.3.3.3 零 位设置

控制器控制机器人时需先对机器人所有轴设置零位，使用 3.3.3-a 所述，控制机器人单轴至零位机械刻度或编码器值符合处停止，依次点击各轴前“零位设置”按键，设置零位。

3.3.4 点位总表窗口功能

点位总表窗口实现功能包括：点位坐标值显示及修改、点位注释名添加修改、点位在点状态显示、点表文件形式导入导出，窗口布局如下：

	标签		注释	X	Y	Z	A	B	C
0	Point 0		原点	-122.85	33.60	-30.00	30.00	-30.00	30.00
1	Point 1		抓取点	-125.00	33.60	-45.00	20.00	20.00	15.00
2	Point 2		放料点	-125.00	23.99	-66.36	150.64	13.80	-117.47
3	Point 3			0.00	0.00	0.00	0.00	0.00	0.00
4	Point 4			0.00	0.00	0.00	0.00	0.00	0.00
5	Point 5			0.00	0.00	0.00	0.00	0.00	0.00
6	Point 6			0.00	0.00	0.00	0.00	0.00	0.00
7	Point 7			0.00	0.00	0.00	0.00	0.00	0.00
8	Point 8			0.00	0.00	0.00	0.00	0.00	0.00
9	Point 9			0.00	0.00	0.00	0.00	0.00	0.00
10	Point 10			0.00	0.00	0.00	0.00	0.00	0.00

图 56

3.3.4.1 点位坐标值显示及修改

点位总表窗口显示所有 3.3.3 功能中执行“点位记录”动作的点位各轴或欧拉角坐标值，未记录点参数默认为“0”，点表记录点数上限 200 点位，点表右侧点击拖动显示。

点击点位后坐标值表格可手动填写坐标值，填写完成需点击“点位保存”，将数据上载保存。如下图所示：

	标签		注释	X	Y	Z	A	B	C
0	Point 0		原点	-122.85	33.60	-30.00	30.00	-30.00	30.00
1	Point 1		抓取点	-125.00	33.60	-45.00	20.00	20.00	15.00
2	Point 2		放料点	-125.00	25	-66.36	150.64	13.80	-117.47
3	Point 3			0.00	0.00	0.00	0.00	0.00	0.00
4	Point 4			0.00	0.00	0.00	0.00	0.00	0.00
5	Point 5			0.00	0.00	0.00	0.00	0.00	0.00
6	Point 6			0.00	0.00	0.00	0.00	0.00	0.00
7	Point 7			0.00	0.00	0.00	0.00	0.00	0.00
8	Point 8			0.00	0.00	0.00	0.00	0.00	0.00
9	Point 9			0.00	0.00	0.00	0.00	0.00	0.00
10	Point 10			0.00	0.00	0.00	0.00	0.00	0.00

图 57

3.3.4.2 点位注释名添加修改

点击各点在点位总表窗口中相应“注释”项可添加修改各点注释名，“注释”名支持中英文输入，填写完成需点击“点位保存”，将数据上载保存。如下图所示：

图 58

3.3.4.3 点位在点状态显示

各点在点位总表窗口中设高光显示项，如机器人当前末端点与已示教点记录坐标值一致（误差0.03mm范围内），高光项显示绿色标志，表明在当前点位。如下图所示：

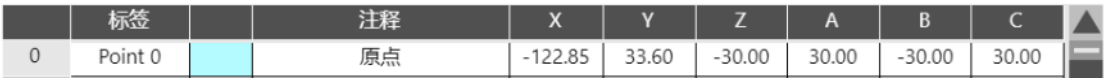


图 59

3.3.4.4 点表文件形式导入导出

点击“文件导出”可将点表以 EXCEL 文件形式导出至制定存储位置，点击“文件导入”可将指定位置点表 EXCEL 文件导入覆盖当前系统存储执行的点表，导入点表格式需与导出文件格式一致，否则无法读取。

3.3.5 限位设置窗口功能

限位设置窗口实现功能包括：机器人各轴软限位范围设置与激活、各轴当前位置显示、各轴转速显示。窗口布局如下：

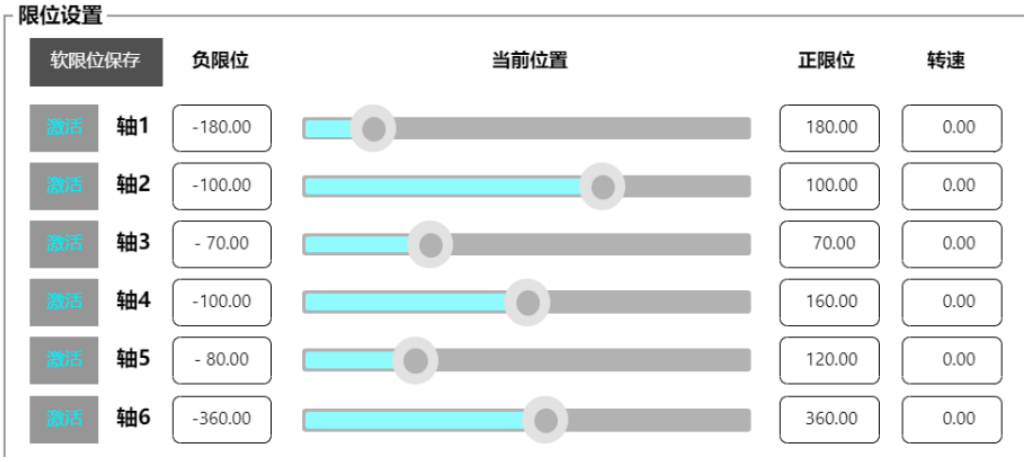


图 60

3.3.5.1 机器人各轴软限位范围设置与激活

“负限位”“正限位”下窗口选中填写完各轴相应正负限位范围，设点范围需小于等于 CORECONTROL 底层设定范围（见 2.3.3 操作）。设定完成需执行“软限位保存”，超出可设定范围触发报错。

图 61

轴限位设定后可通过轴前限位激活按钮选择激活相应轴软限位，如无激活，实际限位参照底层软限位执行，触发软限位机器人停止，不下使能。

3.3.5.2 各轴当前位置显示

显示轴当前所处位置在整体限位范围内大致位置。

3.3.5.3 轴转速显示

实时同步电机驱动器采集电机转速参数，MCS、TCS、PCS 直角坐标系下显示速度参数单位为 mm/s,ACS 关节坐标系显示速度参数单位为°/s。

3.3.6 工具坐标系示教窗口功能

工具坐标示教窗口实现功能包括：工具坐标系六点标定、工具坐标系手动输入、已有工具坐标系查询。窗口布局如下：

图 62

3.3.6.1 工具坐标系 6 标定

点击“示教图示”，可弹窗展示示教点位，具体如下:

图 63

工具坐标下拉窗口选择待示教工具坐标（预设 1-8 项），选择“6 点标定”，控制机器人按照图纸移动至点，点击相应点后窗口并点击执行“记录”，完成一次示教，已记录点后窗口显示“已记录”文字，依次完成两次示教记录。已记录点可重新覆盖，6 点记录完成后，再点击“记录”自动生成工具坐标。标定过程如下图所示：

图 64

注意：所有型制机器人工具坐标系均采用六点标定，标定步骤同上

3.3.6.2 工具坐标系手动输入

工具坐标下拉窗口选择待示教工具坐标（预设 1-8 项），选择“手动输入”，于弹窗相应轴项后窗口点击输入偏移值，完成后点击“标定”生成工具坐标。弹窗如下所示：

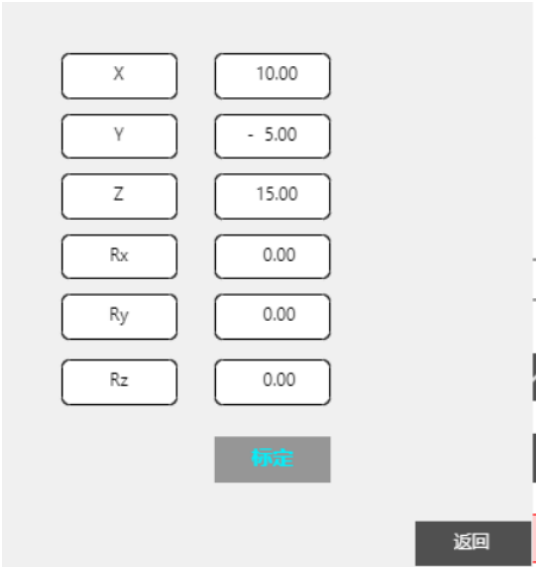


图 65

3.3.6.3 工具坐标系查询

工具坐标下拉窗口选择待已教工具坐标，选择“查询”，于弹窗显示相应轴项偏移值，查询窗口不支持坐标修改。

图 66

3.3.7 用户坐标系示教窗口功能

用户坐标系示教窗口实现功能包括：用户坐标系三点标定、工用户坐标系手动输入、已有用户坐标系查询。窗口布局如下：

3.3.7.1 用户坐标系三点标定

点击“示教图示”，可弹窗展示示教点位，具体如下:

图 67

用户坐标下拉窗口选择待示教用户坐标（预设 1-8 项），选择“3 点标定”，控制机器人按照图纸移动至点，点击相应点后窗口并点击执行“记录”，完成一次示教，已记录点后窗口显示“已记录”文字，依次完成 3 次示教记录。已记录点可重新覆盖，3 点记录完成后，再点击“记录”自动生成用户坐标。标定过程如下图所示：

图 68

3.3.72 用户坐标系手动输入

用户坐标下拉窗口选择待示教用户坐标（预设 1-8 项），选择“手动输入”，于弹窗相应轴项后窗口点击输入偏移值，完成后点击“标定”生成用户坐标。弹窗如下所示：

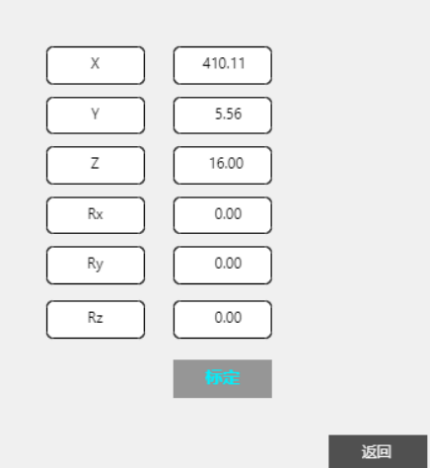


图 69

3.3.7.3 用户坐标系查询

用户坐标下拉窗口选择待已教用户坐标，选择“查询”，于弹窗显示相应轴项偏移值，查询窗口不支持坐标修改。

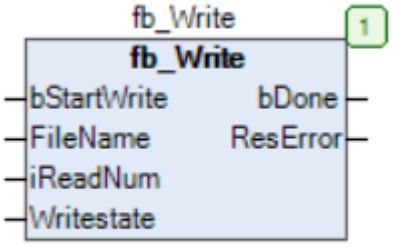
四、常用 FB 指令详解

4.1 数据保存调用

4.1.1 文件形式数据保存

名称：fb_Write

1) 指令格式

指令	名称	图形表现	ST 表现
fb_Write	写 csv 格式数据		fb_Write(bStartWrite:= , FileName:= , iReadNum:= , Writestate:= , bDone=> , ResError=>);

2) 相关变量

◆ **输入变量**

输入变量	名称	数据类型	有效范围	初始值	描述
bStartWrite	执行条件	BOOL	TRUE，FALSE	FALSE	输入一个上升沿将启动功能块的处理
FileName	文件名称	String	—	—	例'D:\Data.csv'
iReadNum	写、读取文件个数	Int	0-100	0	写多少个数据，（写成 read 是说读也要一样的数据），100 个数组，数据可以扩充
Writestate	写数据内容	ParamStruct	自定义		写入的数据数组，最大 100 个 ParamStruct

◆ 输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
bDone		BOOL	TRUE，FALSE	FALSE	写入数据完成，置为 TRUE
ResError		BOOL	TRUE，FALSE	FALSE	数据报错

3) 功能说明

此功能块是写 csv 格式数据，bStartWrite 为输入一个上升沿将启动功能块的处理，在 FileName 输入存储文件的路径。bDone 说明当次写入数据完成。ResError 继承 SysFileOpen 的报错。Writestate 读取出的数据数组，最大容量为 100 个（容量可以自己扩充）。

CSV 格式文件下的数据保存。是根据数据格式 WtiteDataFun 对应的。

4) 注意事项

注意在 WtiteDataFun，读数据与写数据类型、数量要一致

```
FUNCTION WtiteDataFun : BOOL
VAR_INPUT
    hFile: RTS_IEC_HANDLE:= RTS_INVALID_HANDLE;
    WriteData: ParamStruct;
END_VAR
VAR
    udiBytesWrite: __XWORD;
    udiWriteError: RTS_IEC_RESULT;
    strWriteData: STRING(255);
    cycleNum: DINT;
    num: DINT;
    byteWriteBuffer: ARRAY[0..60] OF BYTE;
    sting:STRING;
END_VAR
```

```
IF hFile <> RTS_INVALID_HANDLE THEN

    WtiteDataFun:= FALSE;
    strWriteData:= CONCAT(INT_TO_STRING(WriteData.LableNum),',');
    strWriteData:= CONCAT(strWriteData,INT_TO_STRING(WriteData.CoordSystem));
    strWriteData:= CONCAT(strWriteData,',');
    strWriteData:= CONCAT(strWriteData,LREAL_TO_STRING(WriteData.X_Pos));
    strWriteData:= CONCAT(strWriteData,',');
    strWriteData:= CONCAT(strWriteData,LREAL_TO_STRING(WriteData.Y_Pos));
    strWriteData:= CONCAT(strWriteData,',');
    strWriteData:= CONCAT(strWriteData,LREAL_TO_STRING(WriteData.Z_Pos));
    strWriteData:= CONCAT(strWriteData,',');
    strWriteData:= CONCAT(strWriteData,LREAL_TO_STRING(WriteData.Rx));
    strWriteData:= CONCAT(strWriteData,',');
    strWriteData:= CONCAT(strWriteData,LREAL_TO_STRING(WriteData.Ry));
    strWriteData:= CONCAT(strWriteData,',');
    strWriteData:= CONCAT(strWriteData,LREAL_TO_STRING(WriteData.Rz));
    strWriteData:= CONCAT(strWriteData,',');
    strWriteData:=CONCAT(strWriteData,';');

    StrTrimA(ADR(strWriteData));
    cycleNum:= StrLenA(ADR(strWriteData));
    FOR num:= 0 TO cycleNum DO
        byteWriteBuffer[num]:= strWriteData[num];
    END_FOR
    udiBytesWrite:= SysFileWrite(hFile:= hFile, pbyBuffer:= ADR(byteWriteBuffer), ulSize:= SIZEOF(byteWriteBuffer), pResult:= ADR(udiWriteError));
    WtiteDataFun:= TRUE;
END_IF
```

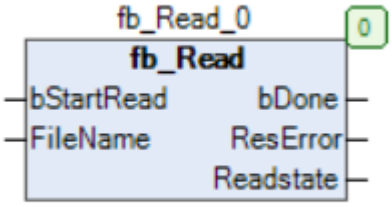
按照自定义的数据格式，修改横线处。

图 70

4.1.2 文件形式数据读取功能块

名称：fb_Read

1) 指令格式

指令	名称	图形表现	ST 表现
fb_Read	读 csv 格式数据		fb_Read(bStartRead:= , FileName:= , bDone=> , ResError=> , Readstate=>);

2) 相关变量说明

◆ **输入变量**

输入变量	名称	数据类型	有效范围	初始值	描述
bStartRead	执行条件	BOOL	TRUE，FALSE	FALSE	输入一个上升沿将启动功能块的处理
FileName	文件名称	String	—	—	例'D:\Data.csv'

◆ **输出变量**

输入变量	名称	数据类型	有效范围	初始值	描述
bDone		BOOL	TRUE，FALSE	FALSE	读取数据完成，置为 TRUE
ResError		BOOL	TRUE，FALSE	FALSE	数据报错
Readstate		ParamStruct			读取数据的数组，例程最大容量 100 个 ParamStruct

3) 功能说明

此功能块是读 csv 格式数据，bStartRead 为输入一个上升沿将启动功能块的处理，在 FileName 输入存储文件的路径。bDone 说明当次读取数据完成。ResError 继承 SysFileOpen 的报错。Readstate 读取出的数据数组，例程的容量为 100 个，可以自行扩充容量。

CSV 格式文件下的数据保存。是根据数据格式 ReadDataFun 对应的。

4) 注意事项

注意在 ReadDataFun，读数据与写数据类型、数量要一致

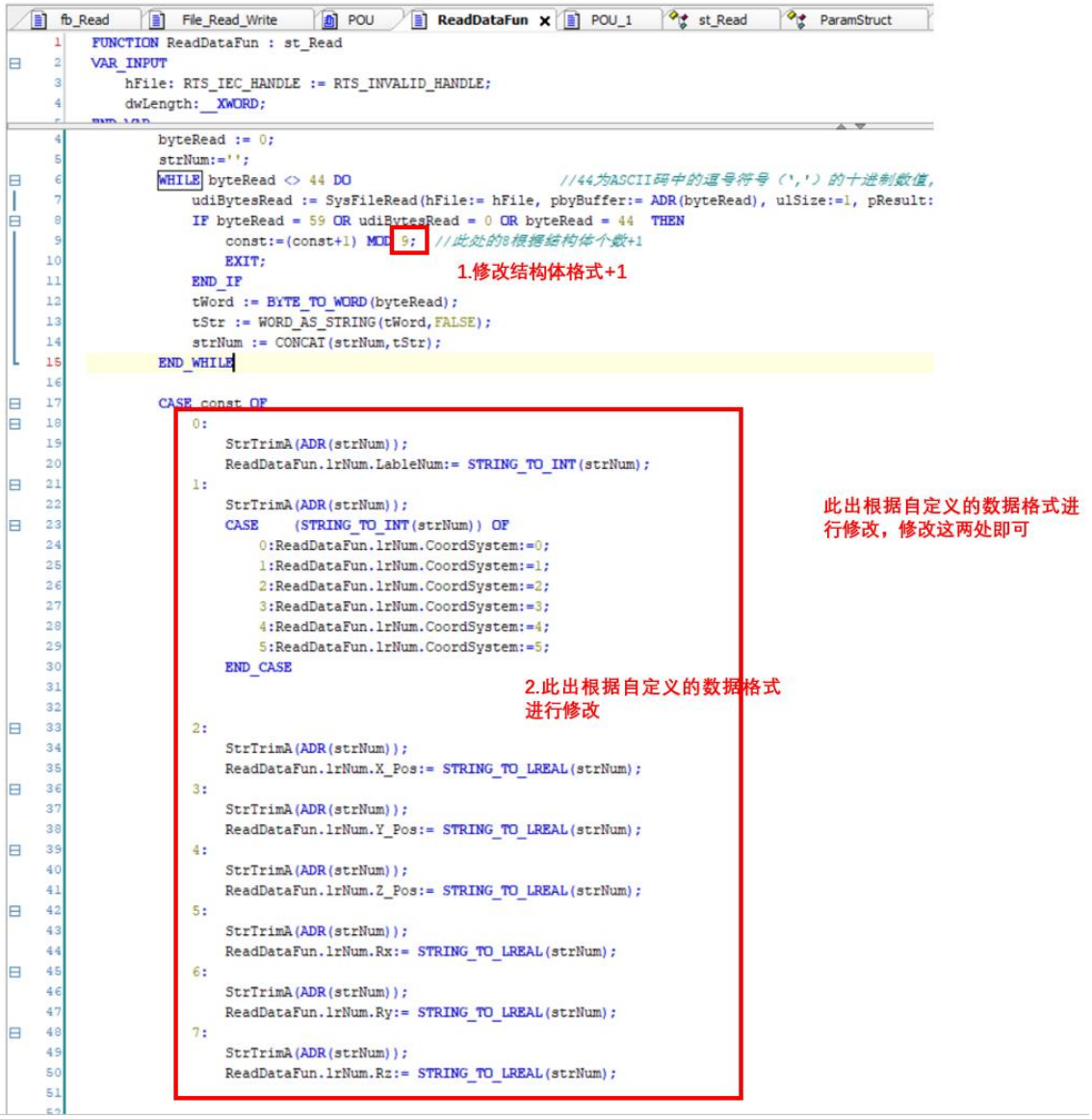


图 71

4.1.3 数据保持型功能

名称：PersistentVars

1) 功能说明

控制器断电情况下数据保存

2) 使用方式

1.确定控制器是否具有保持型 PersistentVars 功能。

2.安装并加载 CmpRetain 库

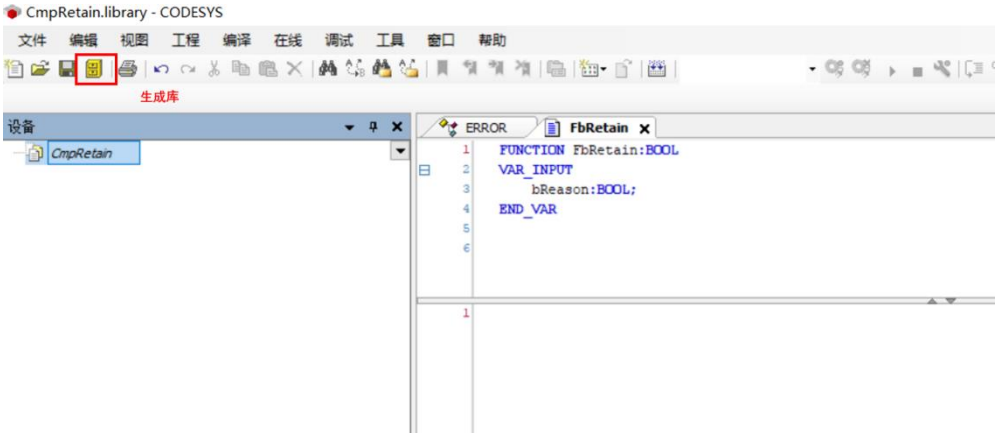


图 72

3.添加 CmpRetain 生成的 COCTRL Common Library Template 3.4.17.0 （COCTRL）

图 73

4.在工程加入此函数语句，该语句实现保存数据功能，PersistentVars 内定义的变量，当 bReason 为常为 True，则根据该 POU 的任务周期（4ms）保存一次，调试人员也可根据实际现场情况加入时间逻辑控制，更改调试保存时间。

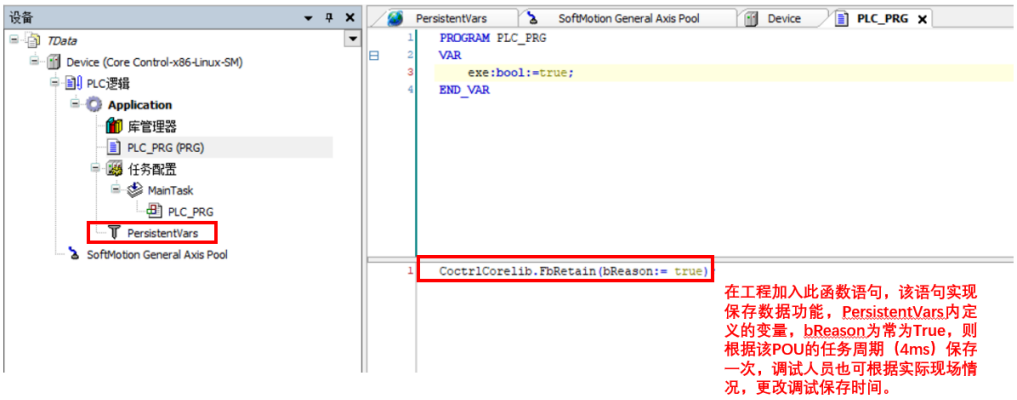


图 74

2) 注意事项

1) 不要在 PersistentVars 内一次定义大量的连续数组类型，容易出现如下报错，一旦出现报错，即使删除也有时会出现无法清除报警信息。在编写选项中，选择清除，切勿选择清除全部，如果清除也不可，那只能选择清除全部，会丢失保持数据。

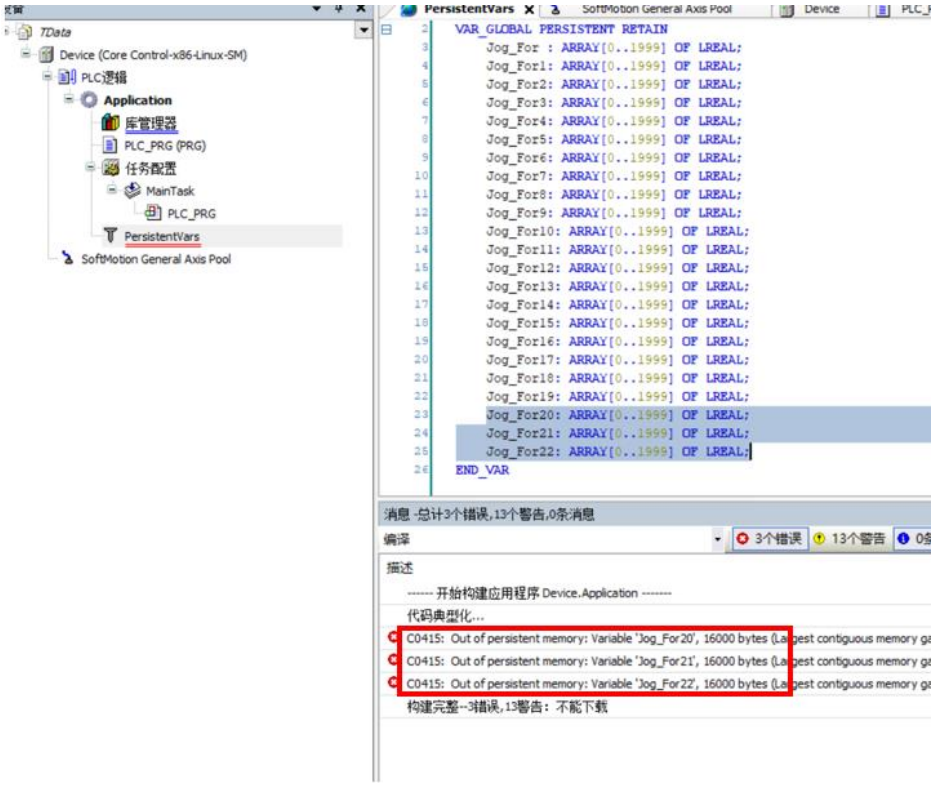


图 75

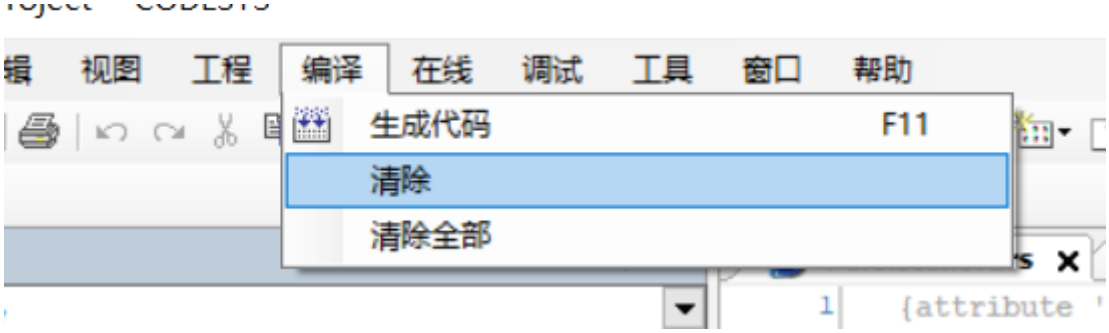


图 76

2) PersistentVars 的容量

目前的存储方式可以存储量很大，目前测试 72M 数据是可行的，连续声明大量变量易出现问题，需要清除全部。会出现数据丢失。

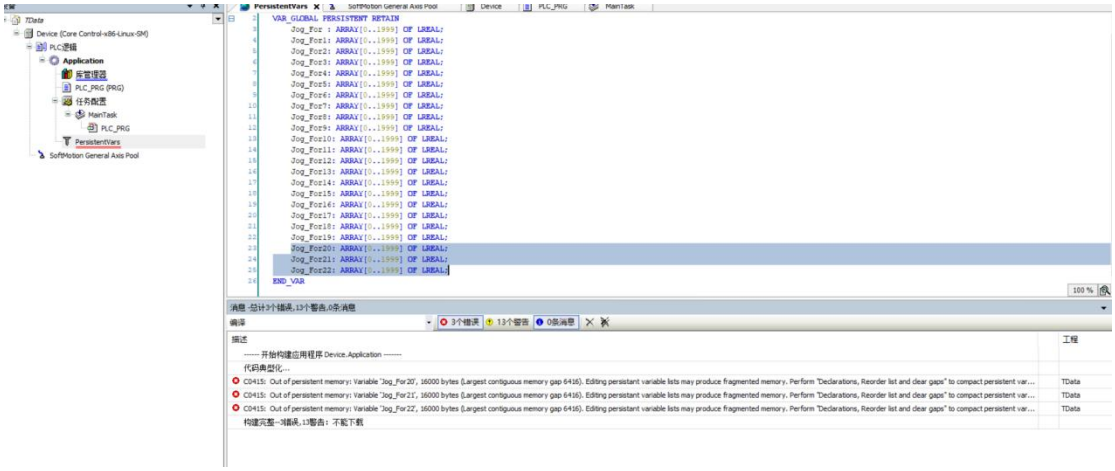


图 77

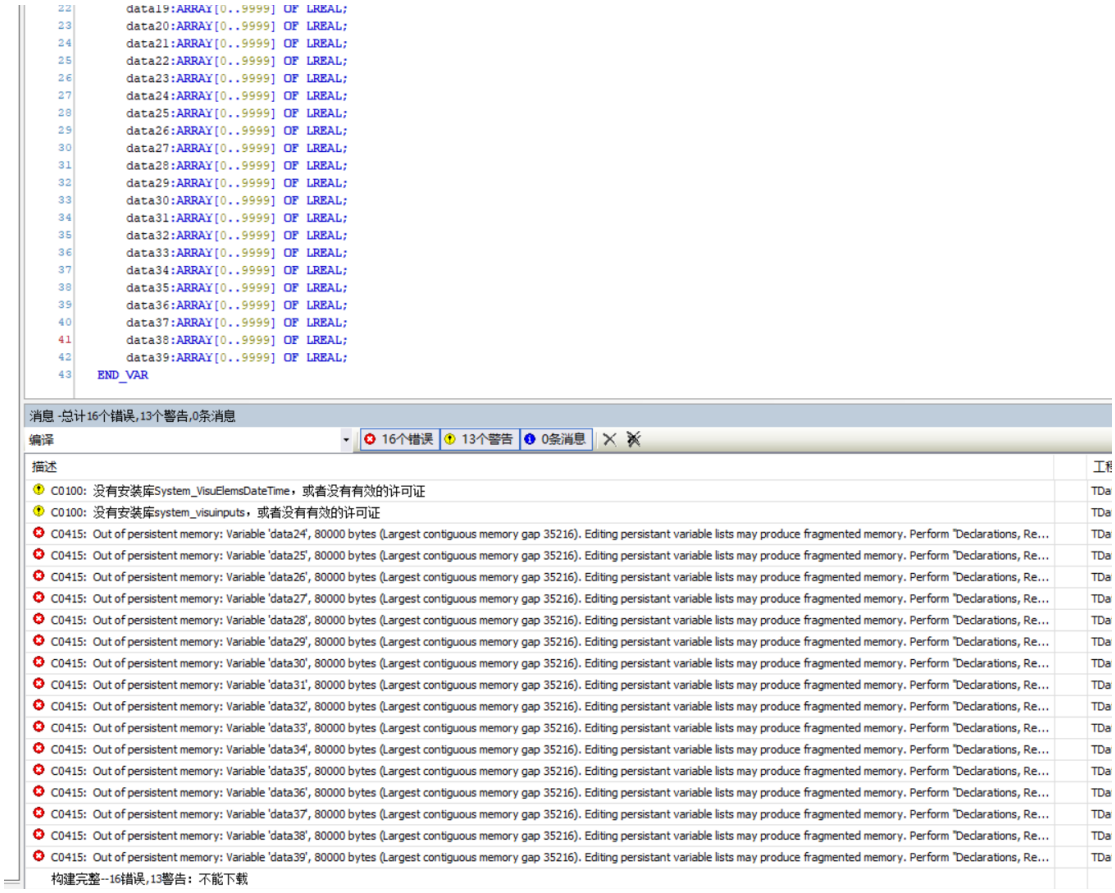


图 78

3) 修改以下方式会可能导致 PersistentVars 数据丢失

a) 出现报错，需要清除时，点击清除全部，数据初始化，数据全部丢失。

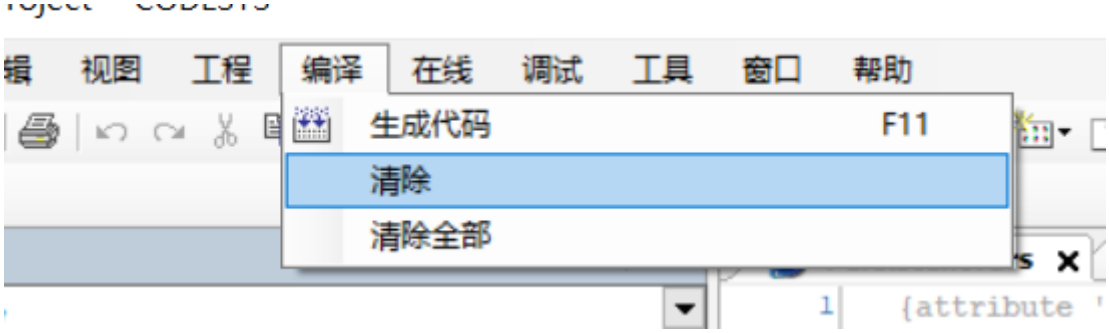


图 79

b) 修改工程名字，过程文件与工程文件不同名字时候，数据回全部丢失。

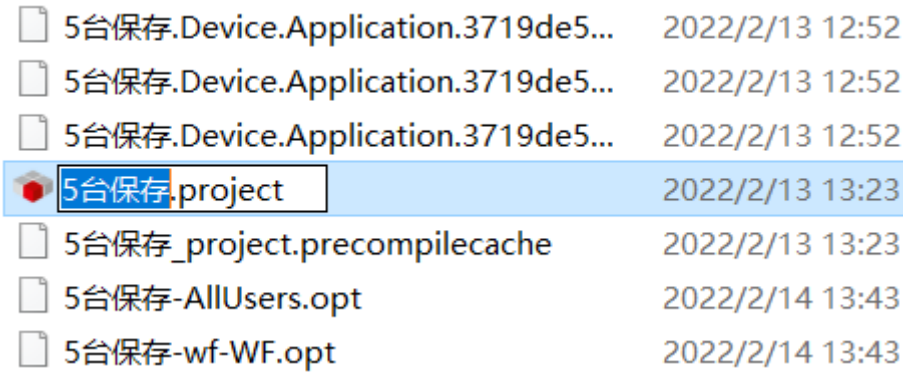


图 80

c) 在已经编译生成的文件后，修改变量名称或修改变量数据类型被修改的变量会丢失值，为 0，未被修改的继续保持值。，将 input1 改为 input1_1，将 input3 的变量类型 INT 改为 DWORD。数据清 0.

Device.Application.PersistentVars

表达式	类型	值	准
input1	INT	2020	
input2	BOOL	TRUE	
input3	INT	2021	
input4	REAL	1999.99988	
input5	LREAL	3337.8889	
input6	STRING	'1234567890'	
input7	DWORD	9999	
input8	BYTE	254	

图 81

Device.Application.PersistentVars

表达式	类型	值	准
input1_1	INT	0	
input2	BOOL	TRUE	
input3	DWORD	0	
input4	REAL	1999.99988	
input5	LREAL	3337.8889	
input6	STRING	'1234567890'	
input7	DWORD	9999	
input8	BYTE	254	

图 82

4.2 各型制机器人的简化关节移动指令

1) 功能说明

本控制器已针对 ST 语言中关节移动功能块（MoveJ 运动）做优化集成，使用下述指令时需安装 3.2 章所述各相应类型示教功能块。

2) 程序示例

演示程序以 6 轴机器人为例（其它类型机器使用方法差异见说明），在 ST 语言中，使用关节指令编写程序如下：

```
IF bStartMovej THEN

    IF CC_Robot_Teach_6Axis_0.MoveJ(Pos:= GVL_Teach_Robot1.Recoder_Pos[1], Vel:= 10, Acc:= 100) THEN

        bStartMovej := FALSE;
    END_IF
END_IF
```

图 83

变量说明：

bStartMovej：程序启动信号；

CC_Robot_Teach_6Axis_0：6 轴机器人示教功能块的实例化对象；GVL_Teach_Robot1.Recoder_Pos：6 轴机器人示教界面的记录点；

程序说明：

6 轴和 4 轴 scara 类型机器人，函数需要输入三个参数，分别是 POS(目标位置)，Vel(速度)，Acc（加速度）。2/3/4 轴龙门或悬臂式机器人，函数只需输入两个参数，分别是 POS(目标位置)，VelF(速度因子)。

MoveJ 是 BOOL 类型，到达结束点返回 TRUE，所以使用 IF 语句编写程序。到达目标位置，复位控制信号。

MoveJ 为 PTP 动作指令，精确执行至目标点，相邻 MoveJ 动作之间存在停顿，连续动作（Blend 动作）见 4.6 关节移动连续轨迹控制。

3) 注意事项：

执行 MoveJ 指令的点须为关节坐标 ACS 下记录的点。

4.3 各型制机器人的简化直线移动指令

1) 功能说明

本控制器已针对 ST 语言中关节移动功能块（MoveL 运动）做优化集成，使用下述指令时需安装 3.2 所述各相应类型示教功能块。

2) 程序示例

演示程序以 6 轴机器人为例（其它类型机器使用方法一致），使用直线指令。在 ST 中编写程序：

```
IF bStartMoveL THEN
    IF CC_Robot_Teach_6Axis_0.MoveL(GVL_Teach_Robot1.Recoder_Pos[2],10,100,1,
        GVL_Teach_Robot1.Recoder_Tool[1],GVL_Teach_Robot1.Recoder_User[3])THEN
        bStartMoveL := FALSE;
    END_IF
END_IF
```

图 84

变量说明：

bStartMoveL : 程序启动信号;

CC_Robot_Teach_6Axis_0 : 6 轴机器人示教功能块的实例化对象;

GVL_Teach_Robot1.Recoder_Pos : 6 轴机器人示教界面的记录点;

GVL_Teach_Robot1.Recoder_User : 6 轴机器人示教界面示教的用户坐标系;

程序说明：

函数需要输入 5 个参数，分别是目标位置（示教的 Point2），速度 10，加速度 100，坐标系选项 1（0：ACS、1：MCS、2：PCS、3：TCS）,工具坐标系 1（已示教的 1 号工具坐标），用户坐标系 3（已示教的 3 号用户坐标系）。上述参数均为示例，可根据具体情况修改。

本例坐标系选项填写 1，该程序执行基坐标系下动作，工具坐标系及用户坐标系选用无效，如需搭配工具执行动作，坐标系选项填写 3，再于工具坐标选项中选择想调用的工具坐标。

MoveL 是 BOOL 类型，到达结束点返回 TRUE，所以使用 IF 语句编写程序。到达目标位置，复位控制信号。

MoveL 为 PTP 动作指令，精确执行至目标点，相邻 MoveJ 动作之间存在停顿，连续动作（Blend 动作）见 4.7 直线移动连续轨迹控制及 4.8 直线与圆弧混合轨迹控制。

3) 注意事项：

MoveL 指令支持 MCS,PCS 等笛卡尔坐标系,不支持 ACS 坐标系，执行 MoveL 指令的点须为 MCS 或 PCS 下的记录点。

笛卡尔坐标系下，如点位示教记录时使用工具与程序调用工具不一致，程序执行时机器人将以默认无工具状态（末端法兰中心处）执行动作。

4.4 各型制机器人的简化圆弧移动指令

1) 功能说明

本控制器已针对 ST 语言中关节移动功能块（MoveC 运动）做优化集成，不需额外调用功能块，使用下述指令时需安装 3.2.1-4 所述示教功能块。

2) 程序示例

演示程序以 6 轴机器人为例，使用圆弧指令。在 ST 中编写程序如下：

```
IF bStartMoveC THEN
    IF CC_Robot_Teach_6Axis_0.MoveC(GVL_Teach_Robot1.Recoder_Pos[1],GVL_Teach_Robot1.Recoder_Pos[2],10,100,1,
        GVL_Teach_Robot1.Recoder_Tool[1],GVL_Teach_Robot1.Recoder_User[3]) THEN
        bStartMoveC := FALSE;
    END_IF
END_IF
```

图 85

轨迹示意：

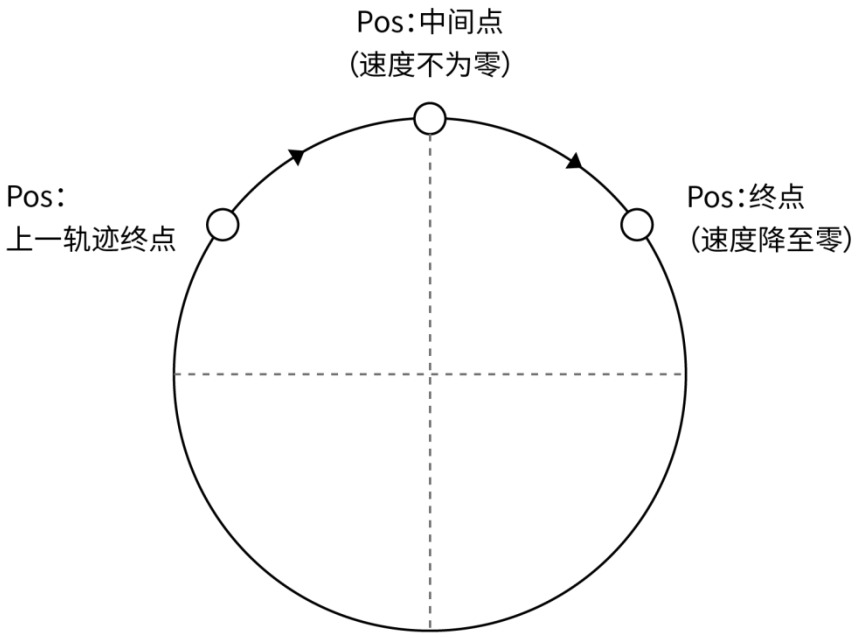


图 86

变量说明：

bStarCicle2：演示程序启动控制信号；

CC_Robot_Teach_Gantry3_0：3 轴机器人示教功能块的实例化对象；

GVL_Teach_Robot4.Recoder_Pos：3 轴机器人示教界面的记录点；

GVL_Teach_Robot4.Recoder_User：3 轴机器人示教界面示教的用户坐标系；

程序说明：

函数固定需要 6 个输入参数，中间点（示教记录点 Point1），结束点（示教记录

点 Point2），速度 10，加速度 100，坐标系选项 1（0：ACS、1：MCS、2：PCS、3：TCS），工具坐标系 1（已示教的 1 号工具坐标），用户坐标系 1（已示教的 1 号用户坐标系）。上述参数均为示例，可根据具体情况修改。

本例坐标系选项填写 1，该程序执行基坐标系下动作，工具坐标系及用户坐标系选用无效，如需搭配工具执行动作，坐标系选项填写 3，再于工具坐标选项中选择想调用的工具坐标。

MoveC 是 BOOL 类型，到达结束点返回 TRUE，所以使用 IF 语句编写程序。到达结束点,复位启动信号。

空间任意三点可确定圆形轨迹，本指令默认执行三点间劣弧部分，限定圆心角小于 180°。

MoveC 为 PTP 动作指令，精确执行至目标点，相邻 MoveC 动作之间存在停顿，连续动作（Blend 动作）见 4.8 混合连续轨迹控制。

3) 注意事项：

MoveL 指令支持 MCS,PCS,不支持 ACS 坐标系，执行 MoveL 指令的点须为 MCS 或 PCS 下的记录点。

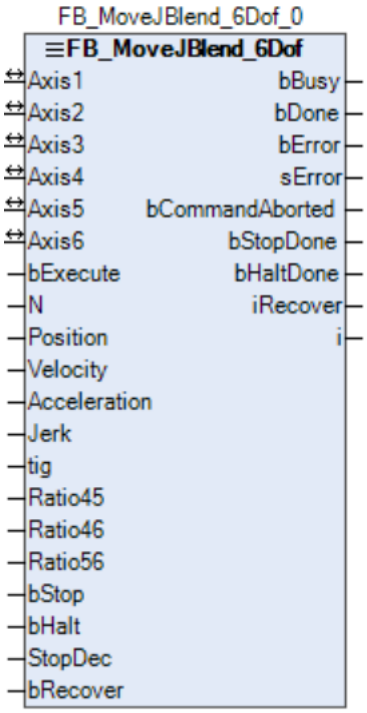
笛卡尔坐标系下，如点位示教记录时使用工具与程序调用工具不一致，程序执行时机器人将以默认无工具状态（末端法兰中心处）执行动作。

4.5 各型制机器人的关节移动连续轨迹控制

4.5.1 所有型制机器人适用

名称：FB_MoveJBlend_6Dof

1) 指令格式

指令	名称	图形表现
FB_MoveJBlend_6Dof	关节移动 Blend 指令	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
Axis1	SM3_Basic.AXIS_REF_SM3		映射轴 1
Axis2	SM3_Basic.AXIS_REF_SM3		映射轴 2
Axis3	SM3_Basic.AXIS_REF_SM3		映射轴 3
Axis4	SM3_Basic.AXIS_REF_SM3		映射轴 4
Axis5	SM3_Basic.AXIS_REF_SM3		映射轴 5

Axis6	SM3_Basic.AXIS_REF_SM3		映射轴 6

◆ 输入变量

输入变量	数据类型	初始值	描述
bExecute	Bool		上升沿启动运动
N	Uint		Blend 运动的点数
Position	Array[0..999]of trafo.AXISPOS_REF		Blend 点的位置值
Velocity	Array[0..999]of LREAL		Blend 点的速度值
Acceleration	Array[0..999]of LREAL		Blend 点的加速度值
Jerk	Array[0..999]of LREAL		Blend 点加加速度
tig	Array[0..999]of LREAL		Blend 运动的半径
Ratio45	LREAL		四五轴耦合比
Ratio46	LREAL		四六轴耦合比
Ratio56	LREAL		五六轴耦合比
bStop	Bool		运动停止指令
bHalt	Bool		上升沿触发运动暂
StopDec	LREAL	1000	运动停止减速度
bRecover	Bool		上升沿从暂停恢复

◆ 输出变量

输出变量	数据类型	初始值	描述
bBusy	Bool		指令正在执行置位 true
bDone	Bool		运动完成置位 true
bError	Bool		异常发生时置位 true
sError	String(81)		异常信息
bCommandAborted	Bool		指令被中断
bStopDone	Bool		运动停止完成标志
bHaltDone	Bool		运动暂停完成标志
iRecover	Int		运动从暂停恢复起始点
i	Int	0	运动所运动到的当前点

3) 功能说明

本功能块为关节运动连续轨迹控制指令。

适用范围：3 轴龙门/悬臂式模组机器人、4 轴龙门/悬臂式模组机器人、Scara（L1-L3）机器人、6 轴机器人。

参数配置:

映射轴：填入运动控制中被控对象的轴名，根据对应关系配置，如下图所示。

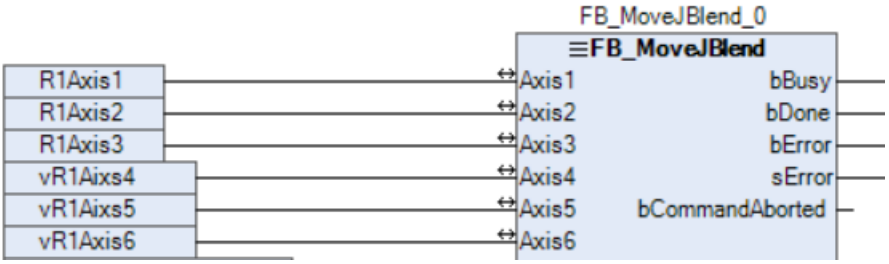


图 87

本功能块基于 6 轴机器人开发，其他运动模型轴数不满 6 轴，缺少的轴需要补充配置虚轴。以 3 轴龙门/悬臂式模组机器人示例，轴配置如下图所示。

图 88

点参数：N,Position,Velocity,Acceleration,Deceleration,Jerk,tig。N 为参与 Blend 的点数量。速度、加速度、加加速度、半径等参数需要根据数量配置到数组的对应元素。

点位参数设定时主要有两种方式：

1.点位参数（速度、坐标系等）各异，不同点位需要一一赋值。可直接在声明区对数组进行赋值。ST 程序如下图所示：

```
tig: ARRAY [0..999] OF LREAL:=[0,20,30,20,30];
```

图 89

图为将 Blend 运动的前四个点的半径设定为 20，30，20，30。其他参数均可参照该方式设置；

注意：Blend 参数数组的第一个元素[0]不要指定，从数组元素[1]开始。

2.点位参数（速度、坐标系等）基本相同时，可使用 For 循环用示教点（数组）赋值。详见程序示例。

耦合参数：Ration45,Ration46,Ration56。根据机械实际情况配置，如无耦合则不需要配置，具体定义参见 3.2.3-4 机器人示教功能块耦合比定义。

启停完成参数：bExecute,bDone,bBusy，bHalt,bStop，bRecover 等。上升沿触发 bExecute 可启动开始运动，运动过程中 bBusy 置为 True。运动完成后，bDone 置为 True,bBusy 置为 False。

上升沿触发 bHalt，机器人会暂停运动，bHaltDone 置为 True。上升沿触发 bRecover,机器人继续运动，bHaltDone 置为 False。

bStop 一般是以紧急停止或特殊处理为主要目的停止功能，bStop 为 True 期间，轴全部保持 Stopping 状态，无法执行其他运动指令，bStop 为 False 后，轴转变为 StandStill 状态。

4) 程序示例

以 3 轴龙门/悬臂机器人连续轨迹编程为例

MoveJ Blend 程序示例

```
//MoveJ-----------Point21-Point27
IF Gantry3.PowerDone THEN
    CASE Gantry3_Step_MoveJ OF
        0:
        IF MoveJ_Start THEN
            Gantry3_Step_MoveJ:=10;
        END_IF
        10:
        Gantry3J.N:=7;
        FOR index:=21 TO 27 DO
            Gantry3J.Vel[index-20] := MoveJ_Vel;
            Gantry3J.Acc[index-20] := MoveJ_Acc;
            Gantry3J.Jerk[index-20] := MoveJ_Acc*10;
            Gantry3J.tig[index-20]  := MoveJ_Tig;
            Gantry3J_Position[index-20].a0 := Gantry3.Recoder_Position_Array[index].v[0];
            Gantry3J_Position[index-20].a1 := Gantry3.Recoder_Position_Array[index].v[1];
            Gantry3J_Position[index-20].a2 := Gantry3.Recoder_Position_Array[index].v[2];
        END_FOR
        Gantry3J.tig[21]:=0;
        Gantry3J.tig[24]:=0;
        Gantry3J.tig[27]:=0;
        Gantry3_Step_MoveJ:=20;
        20:
        Gantry3J.Excute:= TRUE;          此管脚置TRUE，运动开始
        IF Gantry3J.bDone THEN
            Gantry3J.Excute:= FALSE;
            Gantry3_Step_MoveJ:=30;
        END_IF
        30:
        Gantry3_Step_MoveJ:=0;
    END_CASE
END_IF
```

图 90

程序说明：

第 0 步：MoveJ_Start 置 TRUE 后，进入第 10 步；

第 10 步：Gantry3J.N 指定龙门/悬臂机器人本段连续轨迹中包含点位数，程序示例赋值 7，即本段轨迹点数为 7；

For 循环的作用是把示教的点位（Gantry3.Recoder_Position_Array）及设定的速度（MoveJ_Vel）、加速度（MoveJ_Acc）、加加速度（MoveJ_Acc*10，此处可单独建变量进行赋值，建议加加速度为加速度 10 倍关系）、Blend 转弯半径（MoveJ_Tig）赋值到轨迹要运行的 7 个点位信息里（Gantry3J_Position、Gantry3J.Vel、Gantry3J.Acc、Gantry3J.Jerk、Gantry3J.tig）

需要注意：Blend 参数数组的第一个元素[0]不指定，为机器人当前点位，从数组元素[1]开始。本例中执行的是示教的 point21-27 点位，因素不为[1]，FOR 循环中

通过 index-20 运算，表示数组元素从 Gantry3J.Vel[1]开始。

Gantry3J.tig[21]\Gantry3J.tig[24]\Gantry3J.tig[27]置 0 表示该点处转弯半径为 0，即精准到达该点,作用与 MOVEJ 动作一致。

第 20 步：等待 Gantry3J.Excute 置 TRUE 代表轨迹开始运行；轨迹走完之后 Gantry3J.Excute 自动置 FALSE;

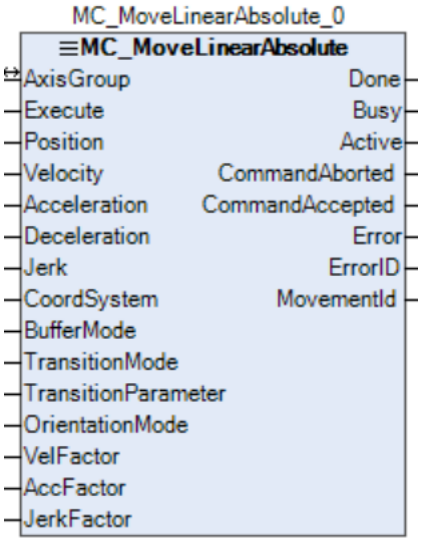
第 30 步：返回程序步 0，等待开始判断；

4.6 各型制机器人的直线移动连续轨迹控制

4.6.1 (2/3/4)轴龙门/悬臂式模组机器人适用

指令名称：MC_MoveLinearAbsolute

1) 指令格式

指令	名称	图形表现
MC_MoveLinearAbsolute	轴组直线运动功能块	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
AxisGroup	AXIS_GROUP_REF_SM3		映射轴组

◆ 输入变量

输入变量	数据类型	初始值	描述
Execute	Bool		上升沿开始运动
Position	SMC_POS_REF		指定坐标系中的结束位置
Velocity	LREAL		速度
Acceleration	LREAL		加速度
Deceleration	LREAL		减速度
Jerk	LREAL		加加速度
CoordSystem	SMC_COORD_SYSTEM		坐标系
BufferMode	MC_BUFFER_MODE		定义 FB 相对于前一
TransitionMode	MC_TRANSITION_MODE		在混合缓冲模式的情况下定义混合
TransitionParameter	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF LREAL		混合参数
OrientationMode	SMC_ORIENTATION_MODE		插补方向
VelFactor	LREAL		每个轴的最大速度乘
AccFactor	LREAL		每个轴的最大加速度
JerkFactor	LREAL		每个轴的最大加加速

◆ 输出变量

输出变量	数据类型	初始值	描述
Done	Bool		到达所有轴的指令终端位置
Busy	Bool		功能块没有完成
Active	Bool		轴开始运动
CommandAborted	Bool		指令被中断
CommandAccepted	Bool		指令被轴组接受
Error	Bool		功能块发生错误
ErrorID	SMC_ERROR		错误代码
MovementId	SMC_Movement_Id		运动唯一标识符

3) 功能说明

上述功能块命令轴组线性移动到指定坐标系中的指定绝对位置。设置参数可以实现连续轨迹控制。详细设置参考程序示例。

4） 程序示例

以 4 轴龙门型机器人连续轨迹编程为例，先在 CFC 中配置运动控制功能块，连续动作需将第一个功能块的 Active 连接到第二个功能块的 Execute。如需要添加工具坐标系及用户坐标系，需同时于 CFC 中配置工具/用户坐标系设定功能块，功能块说明详见 5.27-28，所有配置示例如下：

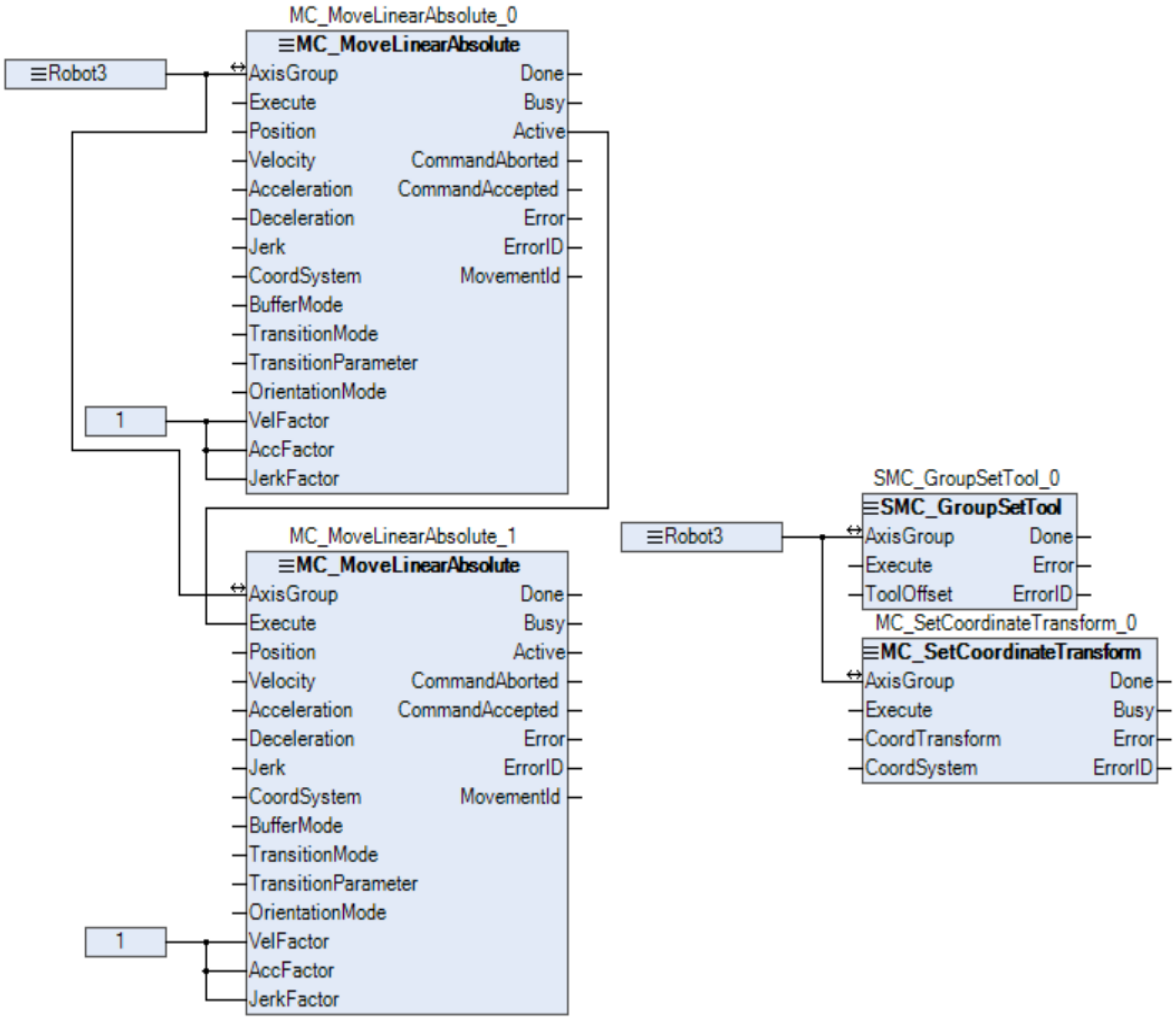


图 91

对应 ST 程序如下：

```
(*
    演示程序：连续运动2个点
    变量说明：
        bStarMoveL : 演示程序启动控制信号
        MC_MoveLinearAbsolute_0 : MC_MoveLinearAbsolute功能块的实例化1
        MC_MoveLinearAbsolute_1 : MC_MoveLinearAbsolute功能块的实例化2
        GVL_Teach_Robot3.Recoder_Pos : 对应示教界面的100个记录点的数组
        GVL_Teach_Robot3.Recoder_Tool : 对应示教界面的8个工具坐标的数组
        GVL_Teach_Robot3.Recoder_User : 对应示教界面的8个用户坐标的数组
*)
IF bStarMoveL THEN
    //----使用工具坐标
    //Tool 1
    SMC_GroupSetTool_0.ToolOffset := GVL_Teach_Robot3.Recoder_Tool[1];
    SMC_GroupSetTool_0.Execute := TRUE;
    //User 1
    MC_SetCoordinateTransform_0.CoordTransform := GVL_Teach_Robot3.Recoder_User[1];
    MC_SetCoordinateTransform_0.CoordSystem := SM3_Robotics.SMC_COORD_SYSTEM.PCS_1;
    MC_SetCoordinateTransform_0.Execute := TRUE;
    //工具和用户坐标设置完成
    IF SMC_GroupSetTool_0.Done AND MC_SetCoordinateTransform_0.Done THEN
        SMC_GroupSetTool_0.Execute := FALSE;
        MC_SetCoordinateTransform_0.Execute := FALSE;
        //----P1
        MC_MoveLinearAbsolute_0.Position := GVL_Teach_Robot3.Recoder_Pos[1]; //point1
        MC_MoveLinearAbsolute_0.Velocity := Vel; //Vel : 设置的速度
        //演示程序中加速度是速度的10倍，也可设置其他倍率
        MC_MoveLinearAbsolute_0.Acceleration := Vel * 10;
        MC_MoveLinearAbsolute_0.Deceleration := Vel * 10;
        //演示程序中加加速度是速度的100倍，也可设置其他倍率
        MC_MoveLinearAbsolute_0.Jerk := vel * 100;
        MC_MoveLinearAbsolute_0.CoordSystem := SM3_Robotics.SMC_COORD_SYSTEM.PCS_1; //用户坐标
        MC_MoveLinearAbsolute_0.BufferMode := 3; // 路径连接上一个FB
        MC_MoveLinearAbsolute_0.TransitionMode := 2; //距离混合
        MC_MoveLinearAbsolute_0.TransitionParameter[0] := 1.5;//blend参数,过渡半径,
        MC_MoveLinearAbsolute_0.TransitionParameter[1] := 1.5;//blend参数，最小曲率半径
        //----P2
        MC_MoveLinearAbsolute_1.Position := GVL_Teach_Robot3.Recoder_Pos[2]; //point2
        MC_MoveLinearAbsolute_1.Velocity := Vel; //Vel : 设置的速度
        //演示程序中加速度是速度的10倍，也可设置其他倍率
        MC_MoveLinearAbsolute_1.Acceleration := Vel * 10;
        MC_MoveLinearAbsolute_1.Deceleration := Vel * 10;
        //演示程序中加加速度是速度的100倍，也可设置其他倍率
        MC_MoveLinearAbsolute_1.Jerk := vel * 100;
        MC_MoveLinearAbsolute_1.CoordSystem := SM3_Robotics.SMC_COORD_SYSTEM.PCS_1; //用户坐标
        MC_MoveLinearAbsolute_1.BufferMode := 3; // 路径连接上一个FB
        MC_MoveLinearAbsolute_1.TransitionMode := 2; //距离混合
        MC_MoveLinearAbsolute_1.TransitionParameter[0] := 1.5;//blend参数,过渡半径,
        MC_MoveLinearAbsolute_1.TransitionParameter[1] := 1.5;//blend参数，最小曲率半径
        //----开始
        MC_MoveLinearAbsolute_0.Execute := TRUE;
        //MC_MoveLinearAbsolute_1.Done : 到达P2点，运动完成
        IF MC_MoveLinearAbsolute_1.Done THEN
            //复位控制信号
            MC_MoveLinearAbsolute_0.Execute := FALSE;
            bStarMoveL := FALSE;
        END_IF
    END_IF
END_IF
```

图 92

程序说明:

示例程序调用 3.2 所述示教库标定的工具坐标系 1，用户坐标系 1，程序调用范围至本段轨迹结束，不同轨迹可调用不同工具。

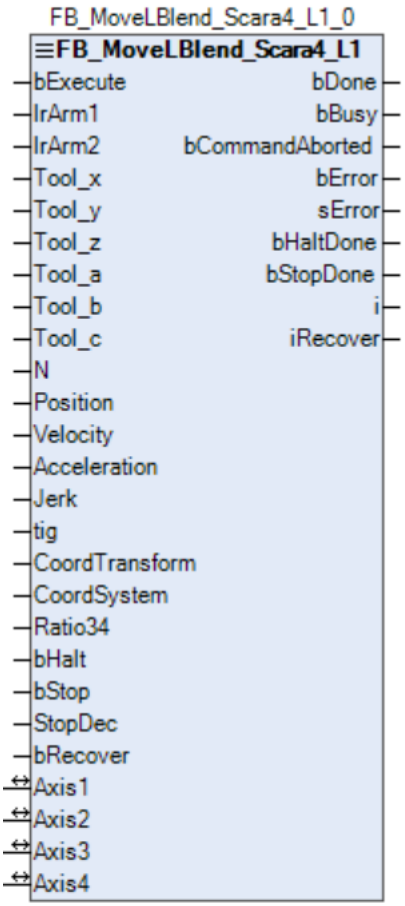
删除相应轨迹段 ST 程序调用工具坐标、用户坐标语句，当段轨迹执行时默认无工具坐标及用户坐标

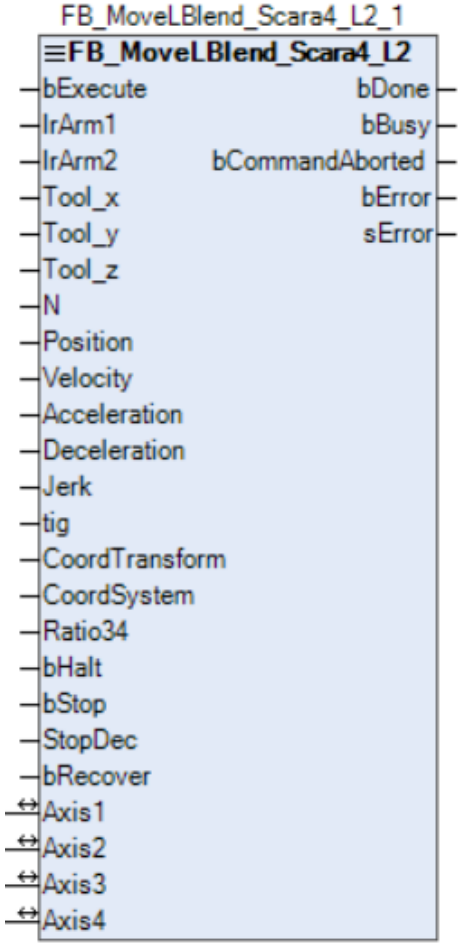
4.6.2 4 轴 SCARA（L1/L2/L3）型机器人适用

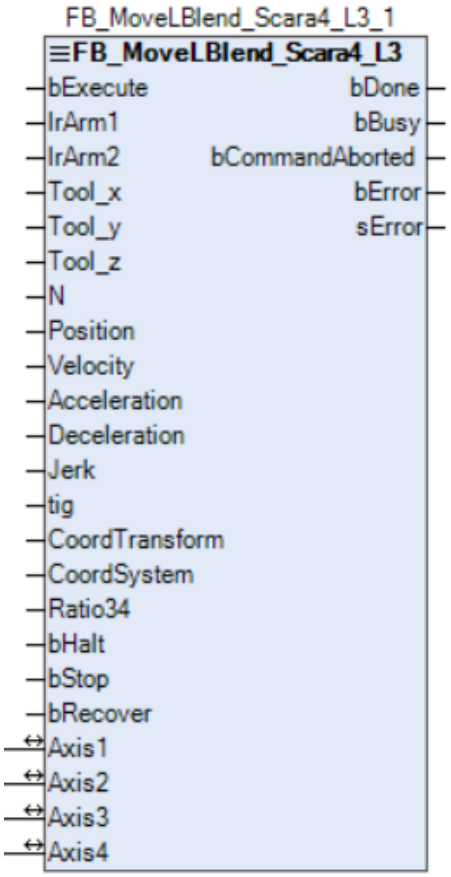
指令名称：FB_MoveLBlend_Scara4_L1

FB_MoveLBlend_Scara4_L2、FB_MoveLBlend_Scara4_L3

1) 指令格式

指令	名称	图形表现
FB_MoveLBlend_Scara4_L1	一二轴线性轴式 4 轴 Scara 机器人带 Blend 直线运动	

指令	名称	图形表现
FB_MoveLBlend_Scara4_L2	二轴线性轴式 4 轴 Scara 机器人带 Blend 直线运动	

指令	名称	图形表现
FB_MoveLBlend_Scara4_L3	三轴线性轴式 4 轴 Scara 机器人带 Blend 直线运动	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
Axis1	SM3_Basic.AXIS_REF_SM3		映射轴 1
Axis2	SM3_Basic.AXIS_REF_SM3		映射轴 2
Axis3	SM3_Basic.AXIS_REF_SM3		映射轴 3
Axis4	SM3_Basic.AXIS_REF_SM3		映射轴 4

◆ 输入变量

输入变量	数据类型	初始值	描述
bExecute	Bool		上升沿启动运动
lrArm1	LREAL		机器人大臂臂长
lrArm2	LREAL		机器人小臂臂长
Tool_X	LREAL		工具坐标 X 偏移
Tool_Y	LREAL		工具坐标 Y 偏移
Tool_Z	LREAL		工具坐标 Z 偏移
N	Uint		Blend 运动的点数
Position	Array[0..999]of position		Blend 点的位置值
Velocity	Array[0..999]of LREAL		Blend 点的速度值
Acceleration	Array[0..999]of LREAL		Blend 点的加速度值
Deceleration	Array[0..999]of LREAL		Blend 点的减速度值
Jerk	Array[0..999]of LREAL		Blend 点加加速度
tig	Array[0..999]of LREAL		Blend 运动的半径
CoordTransform	ARRAY [0..999] OF MC_COORD_REF		同 coordSystem，可不写入值
CoordSystem	ARRAY [0..999] OF SM3_Robotics.SMC_COORD_SYSTEM		Blend 点的坐标系
Ratio34	LREAL		三四轴耦合比
bHalt	Bool		上升沿触发暂停运动
bStop	Bool		运动停止指令
StopDec	LREAL	1000	运动停止减速度
bRecover	Bool		上升沿从暂停恢复

◆ 输出变量

输出变量	数据类型	初始值	描述
bBusy	Bool		指令正在执行置位 true
bDone	Bool		运动完成置位 true
bError	Bool		异常发生时置位 true
sError	String(81)		异常信息
bCommandAborted	Bool		指令被中断
bStopDone	Bool		停止指令完成
i	INT		正在执行的点位数
iRecover	INT		正在执行的点位数

3) 功能说明

上述功能块为二轴线性轴式 4 轴 Scara 机器人带 Blend 的直线运动指令、三轴线性轴式 4 轴 Scara 机器人带 Blend 的直线运动指令。指令效果如下：

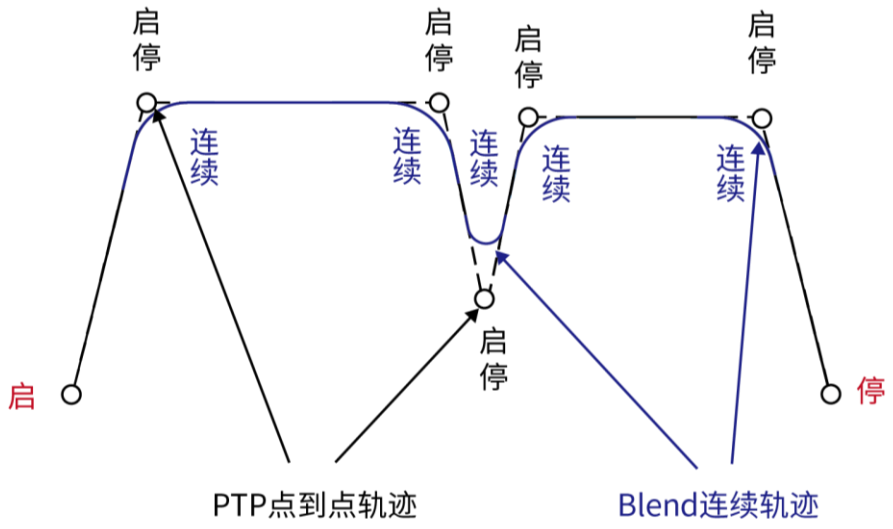


图 93

参数配置：

臂长参数：lrArm1、lrArm2 填入 Scara 大臂和小臂的臂长，单位 mm。

工具坐标参数：Tool_x、Tool_y、Tool_z 填入 Scara 机器人的工具坐标系的标定

偏移结果，如果无工具，可不进行配置。

坐标系：CoordSystem，Blend运动点所处的坐标系。可设定为整数型。设定为0为关节坐标系 ACS，1 为基坐标系 MCS。

4）程序示例

以 4 轴 SCARA（L2）型机器人连续轨迹编程为例

MoveL Blend 程序示例

```
//MoveL_Blend------------Point21-Point30

IF Scara4_L2L_Blend.bDone THEN
    CASE Scara4_L2_Step_L_Blend OF
        0:
        IF MoveL_Blend_Start THEN
            Scara4_L2_Step_L_Blend:=10;
        END_IF
        10:
        Scara4_L2L_Blend.N:=7;
        FOR index:=21 TO 27 DO
            Scara4_L2L_Blend.Vel[index-20] := MoveL_Blend_Vel;
            Scara4_L2L_Blend.Acc[index-20] := MoveL_Blend_Acc;
            Scara4_L2L_Blend.Jerk[index-20] := MoveL_Blend_Acc*10;
            Scara4_L2L_Blend.tig[index-20]  := MoveL_Blend_tig;
            Scara4_L2L_Blend.CoordSystem[index-20]:=Scara4_L2.Recoder_CoordSystem_Array[index];
            Scara4_L2L_Blend_Position[index-20].X := Scara4_L2.Recoder_Position_Array[index].v[0];
            Scara4_L2L_Blend_Position[index-20].Y := Scara4_L2.Recoder_Position_Array[index].v[1];
            Scara4_L2L_Blend_Position[index-20].Z := Scara4_L2.Recoder_Position_Array[index].v[2];
            Scara4_L2L_Blend_Position[index-20].A := Scara4_L2.Recoder_Position_Array[index].v[3];
        END_FOR
        Scara4_L2L_Blend.tig[21]:=0;
        Scara4_L2L_Blend.tig[24]:=0;
        Scara4_L2L_Blend.tig[27]:=0;
        Scara4_L2_Step_L_Blend:=20;
        20:
        Scara4_L2L_Blend.Excute:= TRUE;        此管脚置TRUE，运动开始
        IF Scara4_L2L_Blend.bDone THEN
            Scara4_L2L_Blend.Excute:= FALSE;
            Scara4_L2_Step_L_Blend:=30;
        END_IF
        30:
        Scara4_L2_Step_L_Blend:=0;
    END_CASE
END_IF
```

图 94

程序说明：

第 0 步：Movel_blend_Start 置 TRUE 后，进入第 10 步；

第 10 步：Scara4_L2L_Blend.N 指定 SCARA 机器人本段连续轨迹中包含点位数，程序示例赋值 7，即本段轨迹点数为 7；

For 循环的作用是把示教的点位（Scara4_L2.Recoder_Position_Array）及设定的速度（MoveL_Blend _Vel）、加速度（MoveL_Blend_Acc）、加加速度（MoveL_Blend_Acc*10，此处可单独建变量进行赋值，建议加加速度为加速度 10 倍关系）、Blend 半径（MoveL_Blend_Tig）赋值到轨迹要运行的 7 个点位信息里（Scara4_L2L_Blend_Position、Scara4_L2L_Blend.Vel、Scara4_L2L_Blend.Acc、Scara4_L2L_Blend.Jerk、Scara4_L2L_Blend.tig）。

需要注意：Blend 参数数组的第一个元素[0]不指定，为机器人当前点位，从数组元素[1]开始。本例中执行的是示教的 point21-27 点位，因素不为[1]，FOR 循环中通过 index-20 运算，表示数组元素从 Scara4_L2L_Blend.Vel[1]开始。

Scara4_L2L_Blend.tig[21]\Scara4_L2L_Blend.tig[24]\Scara4_L2L_Blend.tig[27]置 0 表示该点处转弯半径为 0，即精准到达该点，即精准到达该点,作用与 MOVEJ 动作一致。

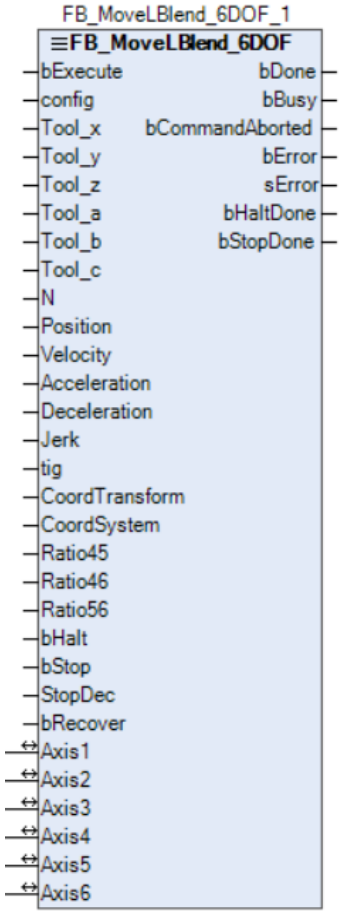
第 20 步：等待 Scara4_L2L_Blend.Excute 置 TRUE 代表轨迹开始运行，轨迹走完之后 Scara4_L2L_Blend.Excute 自动置 FALSE;

第 30 步：返回程序步 0，等待开始判断。

4.6.3 6 轴机器人适用

指令名称：FB_MoveLBlend_6DOF

1) 指令格式

指令	名称	图形表现
FB_MoveLBlend_6DOF	六轴机器人带 Blend 直线运动指令	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
Axis1	SM3_Basic.AXIS_REF_SM3		映射轴 1
Axis2	SM3_Basic.AXIS_REF_SM3		映射轴 2
Axis3	SM3_Basic.AXIS_REF_SM3		映射轴 3
Axis4	SM3_Basic.AXIS_REF_SM3		映射轴 4
Axis5	SM3_Basic.AXIS_REF_SM3		映射轴 5

Axis6	SM3_Basic.AXIS_REF_SM3		映射轴 6

◆ 输入变量

输入变量	数据类型	初始值	描述
bExecute	Bool		上升沿启动运动
config	SM3_CNC.SMC_TrafoConfig_ArticulatedRobot_6DOF		六轴机器人 D-H 参数
Tool_X	LREAL		工具坐标 X 偏移
Tool_Y	LREAL		工具坐标 Y 偏移
Tool_Z	LREAL		工具坐标 Z 偏移
Tool_A	LREAL		工具坐标 A 偏移
Tool_B	LREAL		工具坐标 B 偏移
Tool_C	LREAL		工具坐标 C 偏移
N	Uint		Blend 运动的点数
Position	Array[0..999]of position		Blend 点的位置值
Velocity	Array[0..999]of LREAL		Blend 点的速度值
Acceleration	Array[0..999]of LREAL		Blend 点的加速度值
Deceleration	Array[0..999]of LREAL		Blend 点的减速度值
Jerk	Array[0..999]of LREAL		Blend 点加加速度
tig	Array[0..999]of LREAL		Blend 运动的半径
CoordTransform	ARRAY [0..999] OF MC_COORD_REF		
CoordSystem	ARRAY [0..999] OF SM3_Robotics.SMC_COORD_SYSTEM		Blend 点的坐标系
Ratio45	LREAL		四五轴耦合比
Ratio46	LREAL		四六轴耦合比
Ratio56	LREAL		五六轴耦合比
bStop	Bool		运动停止指令
bHalt	Bool		上升沿触发运动暂停
StopDec	LREAL	1000	运动停止减速度

bRecover	Bool		上升沿从暂停恢复

◆ 输出变量

输出变量	数据类型	初始值	描述
bBusy	Bool		指令正在执行置位 true
bDone	Bool		运动完成置位 true
bError	Bool		异常发生时置位 true
sError	String(81)		异常信息
bCommandAborted	Bool		指令被中断
bStopDone	Bool		运动停止完成标志
bHaltDone	Bool		运动暂停完成标志

3) 功能说明

本功能块为六轴机器人带 Blend 的直线运动指令。指令效果如下：

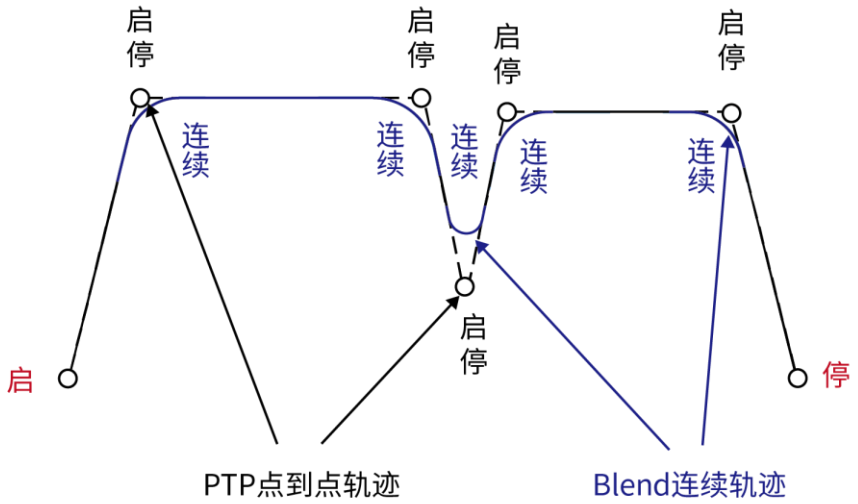


图 95

Config: D-H 参数根据当前六轴机器人的实际参数进行配置。

例如：

```
Config: SM3_CNC.SMC_TrafoConfig_ArticulatedRobot_6DOF :=
(a1 := 30 ,a2 := 270 , a3 := 79,d1 := 345 ,d3 := 0 ,d4 := 320 ,d6 := 110 ,
 q3_max_deg := 360,q3_min_deg := -360 ,q5_max_deg := 3600,q5_min_deg := -3600);
```

图 96

4） 程序示例

MoveL Blend 程序示例

```
//MoveL_Blend------------Point21-Point30

IF Robot6.PowerDone THEN
    CASE Robot6_Step_MoveL_Blend OF
        0:
        IF MoveL_Blend_Start THEN
            Robot6_Step_MoveL_Blend:=10;
        END_IF
        10:
        Robot6L_Blend.N:=7;
        FOR index:=21 TO 27 DO
            Robot6L_Blend.Vel[index-20] := MoveL_Blend_Vel;
            Robot6L_Blend.Acc[index-20] := MoveL_Blend_Acc;
            Robot6L_Blend.Jerk[index-20] := MoveL_Blend_Acc*10;
            Robot6L_Blend.tig[index-20]  := MoveL_Blend_tig;
            Robot6L_Blend.CoordSystem[index-20]:=Robot6.Recoder_CoordSystem_Array[index];
            Robot6L_Blend.Position[index-20].X := Robot6.Recoder_Position_Array[index].v[0];
            Robot6L_Blend.Position[index-20].Y := Robot6.Recoder_Position_Array[index].v[1];
            Robot6L_Blend.Position[index-20].Z := Robot6.Recoder_Position_Array[index].v[2];
            Robot6L_Blend.Position[index-20].A := Robot6.Recoder_Position_Array[index].v[3];
            Robot6L_Blend.Position[index-20].B := Robot6.Recoder_Position_Array[index].v[4];
            Robot6L_Blend.Position[index-20].C := Robot6.Recoder_Position_Array[index].v[5];
        END_FOR
        Robot6L_Blend.tig[21]:=0;
        Robot6L_Blend.tig[24]:=0;
        Robot6L_Blend.tig[27]:=0;
        Robot6_Step_MoveL_Blend:=20;
        20:
        Robot6L_Blend.Excute:= TRUE;        此管脚置TRUE，运动开始
        IF Robot6L_Blend.bDone THEN
            Robot6L_Blend.Excute:= FALSE;
            Robot6_Step_MoveL_Blend:=30;
        END_IF
        30:
        Robot6_Step_MoveL_Blend:=0;
    END_CASE
END_IF
```

图 97

程序说明：

第 0 步：Movel_blend_Start 置 TRUE 后，进入第 10 步；

第 10 步：Robot6L_Blend.N 指定六轴机器人本段连续轨迹中包含点位数，程序示例赋值 7，即本段轨迹点数为 7；

For 循环的作用是把示教的点位（Robot6.Recoder_Position_Array）及设定的速度（MoveL_Blend _Vel）、加速度（MoveL_Blend_Acc）、加加速度（MoveL_Blend_Acc*10，此处可单独建变量进行赋值，建议加加速度为加速度 10 倍关系）、Blend 半径（MoveL_Blend_Tig）赋值到轨迹要运行的点位信息里（Robot6L_Blend_Position、Robot6L_Blend.Vel、Robot6L_Blend.Acc、Robot6L_Blend.Jerk、Robot6L_Blend.tig）。

需要注意：Blend 参数数组的第一个元素[0]不指定，为机器人当前点位，从数组元素[1]开始。本例中执行的是示教的 point21-27 点位，因素不为[1]，FOR 循环中通过 index-20 运算，表示数组元素从 Robot6L_Blend.Vel[1]开始。

Robot6L_Blend.tig[21]\ Robot6L_Blend.tig[24]\ Robot6L_Blend.tig[27]置 0 表示该点处转弯半径为 0，即精准到达该点，即精准到达该点,作用与 MOVEJ 动作一致。

第 20 步：等待 Robot6L_Blend.Excute 置 TRUE 代表轨迹开始运行，轨迹走完之后 Scara4_L2L_Blend.Excute 自动置 FALSE;

第 30 步：返回程序步 0，等待开始判断。

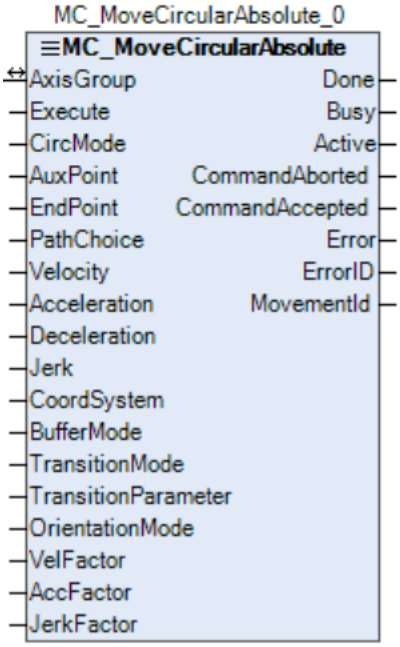
4.7 各型制机器人直线与圆弧混合连续轨迹控制

4.7.1 (2/3/4)轴龙门/悬臂式模组机器人适用

涉及指令名称：MC_MoveLinearAbsolute，指令介绍详见 4.6.1 直线移动连续轨迹控制；

涉及指令名称：MC_MoveCircularAbsolute

1) 指令格式

指令	名称	图形表现
MC_MoveCircularAbsolute	轴组的圆弧运动指令	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
AxisGroup	AXIS_GROUP_REF_SM3		映射轴组

◆ 输入变量

输入变量	数据类型	初始值	描述
Execute	Bool		上升沿启动运动
CircMode	LREAL		弧段定义方式
AuxPoint	SMC_POS_REF		辅助点
EndPoint	SMC_POS_REF		结束点
PathChoice	MC_CIRC_PATHCHOICE		顺时针或逆时针
Velocity	LREAL		速度
Acceleration	LREAL		加速度
Deceleration	LREAL		减速度
Jerk	LREAL		加加速度
CoordSystem	SMC_COORD_SYSTEM		坐标系
BufferMode	MC_BUFFER_MODE		定义 FB 相对于前一个块的时间顺序。
TransitionMode	MC_TRANSITION_MODE		在混合缓冲模式的情况下定义混合
TransitionParameter	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF		混合参数
OrientationMode	SMC_ORIENTATION_MODE		插补方向
VelFactor	LREAL		每个轴的最大速度乘以该因子，该因子必须在]0, 1] 范围内。
AccFactor	LREAL		每个轴的最大加速度乘以该因数，该因数必须在]0, 1] 范围内。
JerkFactor	LREAL		每个轴的最大加加速度乘以该因子，该因子必须在]0, 1] 范围内。

◆ 输出变量

输出变量	数据类型	初始值	描述
Done	Bool		到达所有轴的指令终端位置
Busy	Bool		功能块没有完成
Active	Bool		轴开始运动
CommandAborted	Bool		指令被中断
CommandAccepted	Bool		指令被轴组接受
Error	Bool		功能块发生错误
ErrorID	SMC_ERROR		错误代码
MovementId	SMC_Movement_Id		运动唯一标识符

3) 功能说明

该功能块是命令轴组向指定坐标系中的绝对位置进行圆周运动。

4) 程序示例

a) 连续整圆轨迹：

先在 CFC 中配置功能块，功能块介绍详见 4.5.1，示例程序中坐标系选择 MCS，速度、加速度和加加速度因子设置成 1。第一个功能块的 Active 连接到第二个功能块的 Execute。如需要添加工具坐标系及用户坐标系，需同时于 CFC 中配置工具/用户坐标系设定功能块，功能块说明详见 5.27-28，所有配置示例如下：

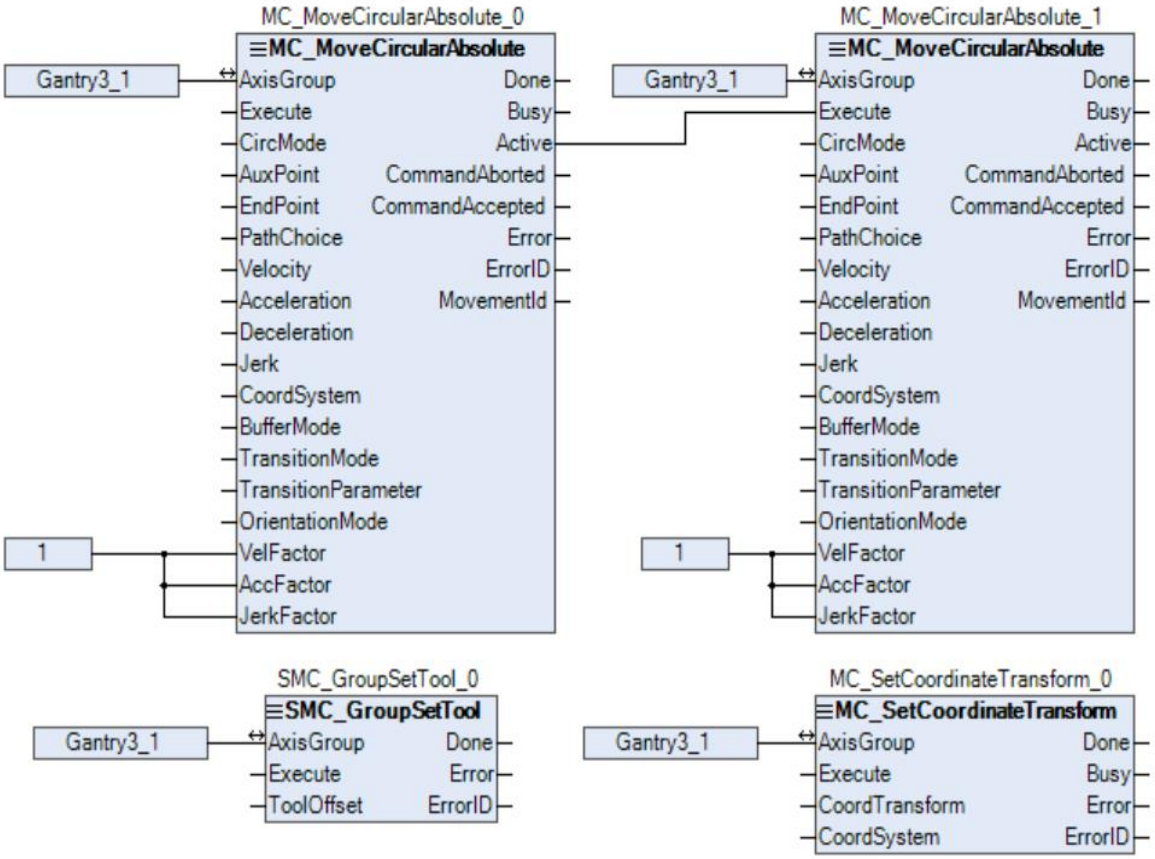


图 98

然后在 ST 语句中编写如下程序：

```
(*
    演示程序：连续2个半圆，组合成一个整圆轨迹
    变量说明：
        bStarCicle : 演示程序启动控制信号
        MC_MoveCircularAbsolute_0 : MC_MoveCircularAbsolute功能块的实例化1
        MC_MoveCircularAbsolute_1 : MC_MoveCircularAbsolute功能块的实例化2
        GVL_Teach_Robot4.Recoder_Pos : 对应示教界面的100个记录点的数组
*)
IF bStarCicle THEN
    //----使用工具和用户坐标
    //Tool 1
    SMC_GroupSetTool_0.ToolOffset := GVL_Teach_Robot3.Recoder_Tool[1];
    SMC_GroupSetTool_0.Execute := TRUE;
    //User 1
    MC_SetCoordinateTransform_0.CoordTransform := GVL_Teach_Robot3.Recoder_User[1];
    MC_SetCoordinateTransform_0.CoordSystem := SM3_Robotics.SMC_COORD_SYSTEM.PCS_1;
    MC_SetCoordinateTransform_0.Execute := TRUE;
    //工具和用户坐标设置完成
    IF SMC_GroupSetTool_0.Done AND MC_SetCoordinateTransform_0.Done THEN
        SMC_GroupSetTool_0.Execute := FALSE;
        MC_SetCoordinateTransform_0.Execute := FALSE;

        MC_MoveCircularAbsolute_0.AuxPoint := GVL_Teach_Robot4.Recoder_Pos[10]; //point10
        MC_MoveCircularAbsolute_0.EndPoint := GVL_Teach_Robot4.Recoder_Pos[11]; //point11
        MC_MoveCircularAbsolute_0.Velocity := Vel; //Vel : 设置的速度
        MC_MoveCircularAbsolute_0.CoordSystem := SM3_Robotics.SMC_COORD_SYSTEM.PCS_1;
        //演示程序中加速度是速度的10倍，也可设置其他倍率
        MC_MoveCircularAbsolute_0.Acceleration := Vel * 10;
        MC_MoveCircularAbsolute_0.Deceleration := Vel * 10;
        //演示程序中加加速度是速度的100倍，也可设置其他倍率
        MC_MoveCircularAbsolute_0.Jerk := vel * 100;
        MC_MoveCircularAbsolute_0.BufferMode := 3; // 路径连接上一个FB
        MC_MoveCircularAbsolute_0.TransitionMode := 1; //速度混合
        MC_MoveCircularAbsolute_0.TransitionParameter[0] := 1;//blend参数0-1，加速与减速比
        MC_MoveCircularAbsolute_0.TransitionParameter[1] := 1;//blend参数0-1，最小曲率比

        MC_MoveCircularAbsolute_1.AuxPoint := GVL_Teach_Robot4.Recoder_Pos[12]; //point12
        MC_MoveCircularAbsolute_1.EndPoint := GVL_Teach_Robot4.Recoder_Pos[13]; //point13
        MC_MoveCircularAbsolute_1.Velocity := Vel; //Vel : 设置的速度
        MC_MoveCircularAbsolute_1.CoordSystem := SM3_Robotics.SMC_COORD_SYSTEM.PCS_1;
        //演示程序中加速度是速度的10倍，也可设置其他倍率
        MC_MoveCircularAbsolute_1.Acceleration := Vel * 10;
        MC_MoveCircularAbsolute_1.Deceleration := Vel * 10;
        //演示程序中加加速度是速度的100倍，也可设置其他倍率
        MC_MoveCircularAbsolute_1.Jerk := vel * 100;
        MC_MoveCircularAbsolute_1.BufferMode := 3; // 路径连接上一个FB
        MC_MoveCircularAbsolute_1.TransitionMode := 1; //速度混合
        MC_MoveCircularAbsolute_1.TransitionParameter[0] := 1;//blend参数0-1
        MC_MoveCircularAbsolute_1.TransitionParameter[1] := 1;//blend参数0-1
        //开始
        MC_MoveCircularAbsolute_0.Execute := TRUE;
        //MC_MoveCircularAbsolute_1.Done : 运动完成
        IF MC_MoveCircularAbsolute_1.Done THEN
            //复位控制信号
            MC_MoveCircularAbsolute_0.Execute := FALSE;
            bStarCicle := FALSE;
        END_IF
    END_IF
END_IF
```

图 99

轨迹说明：

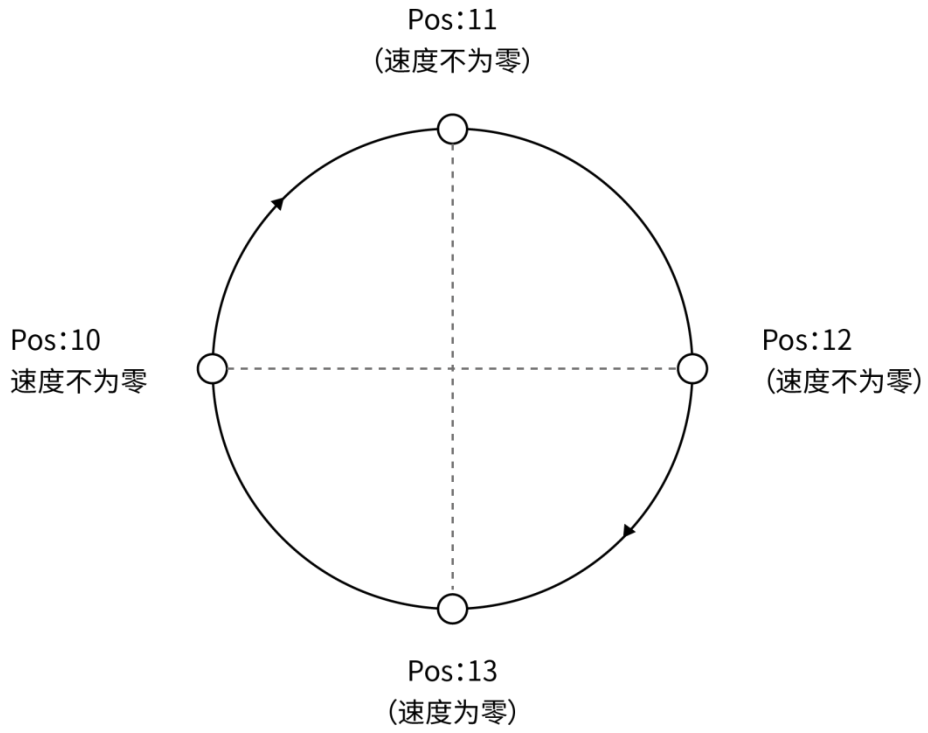


图 100

程序说明:

轨迹起始点和结束点都是上图中的 P13,顺时针运动。

示例程序调用 3.2 所述示教库标定的工具坐标系 1，用户坐标系 1，程序调用范围至本段轨迹结束，不同轨迹可调用不同工具。

删除相应轨迹段 ST 程序调用工具坐标、用户坐标语句，当段轨迹执行时默认无工具坐标及用户坐标

b) 圆弧直线混合轨迹示例程序

先在 CFC 中配置功能块，示例程序中，坐标系都设置成 MCS,速度、加速度、加加速度因子都设置成 1.前一个功能块的 active 连到下一个功能块的 Execute

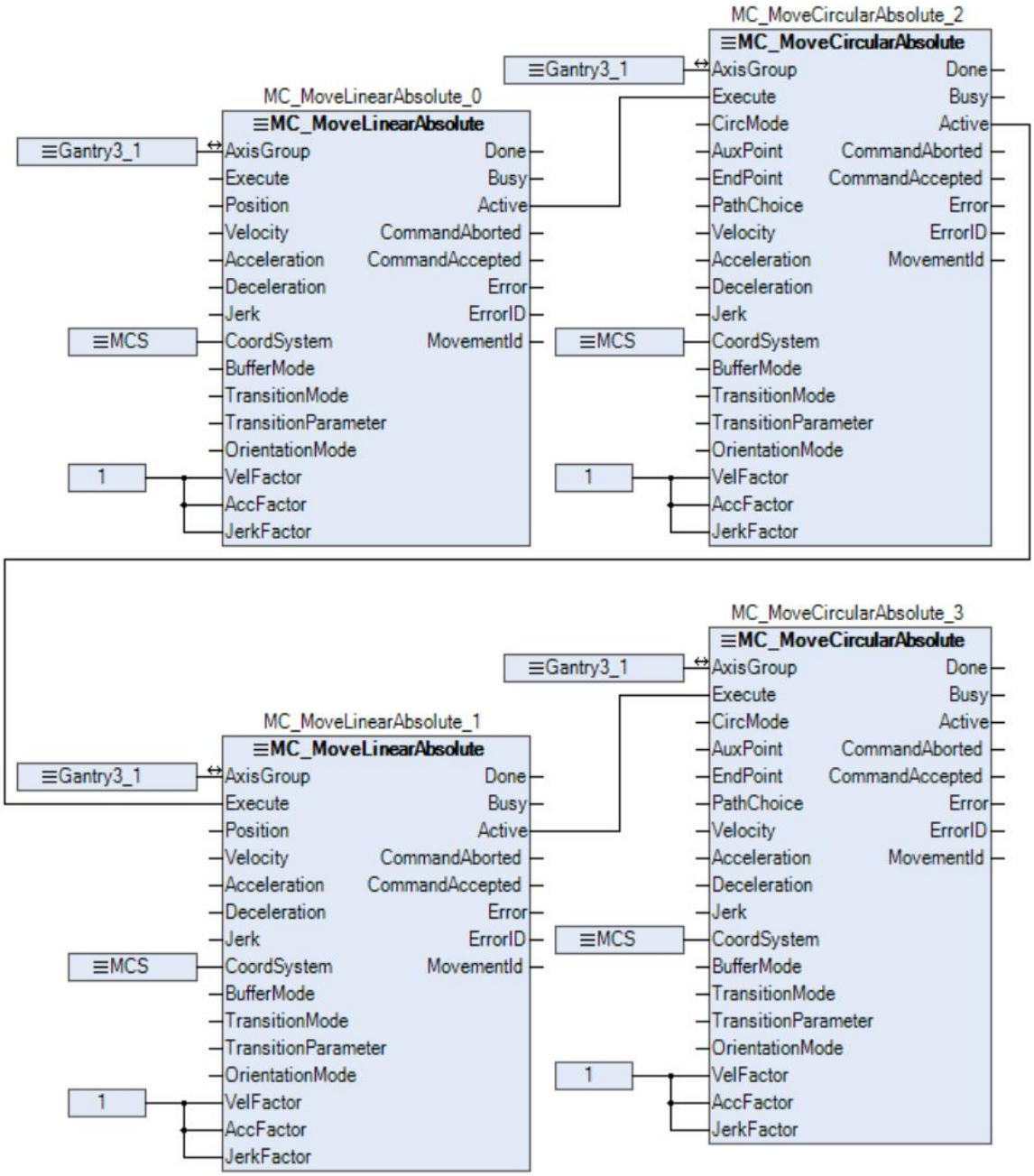


图 101

然后在 ST 中编写程序:

```
(*
    演示程序：直线，圆弧，直线，圆弧4段连续运动轨迹
    变量说明：
        bStarLC : 演示程序启动控制信号
        MC_MoveLinearAbsolute_0 : MC_MoveLinearAbsolute功能块的实例化1
        MC_MoveLinearAbsolute_1 : MC_MoveLinearAbsolute功能块的实例化2
        MC_MoveCircularAbsolute_2 : MC_MoveCircularAbsolute功能块的实例化1
        MC_MoveCircularAbsolute_3 : MC_MoveCircularAbsolute功能块的实例化2
        GVL_Teach_Robot4.Recoder_Pos : 对应示教界面的100个记录点的数组
*)
IF bStarLC THEN
    //直线1
    MC_MoveLinearAbsolute_0.Position := GVL_Teach_Robot4.Recoder_Pos[20];//point20
    MC_MoveLinearAbsolute_0.Velocity := Vel;//Vel : 设置的速度
    //演示程序中加速度是速度的10倍，也可设置其他倍率
    MC_MoveLinearAbsolute_0.Acceleration := Vel * 10;
    MC_MoveLinearAbsolute_0.Deceleration := Vel * 10;
    //演示程序中加加速度是速度的100倍，也可设置其他倍率
    MC_MoveLinearAbsolute_0.Jerk := Vel * 100;
    MC_MoveLinearAbsolute_0.BufferMode := 3;// 路径连接上一个FB
    MC_MoveLinearAbsolute_0.TransitionMode := 2;//距离混合
    MC_MoveLinearAbsolute_0.TransitionParameter[0] := 1.635;//blend参数,过渡半径,
    MC_MoveLinearAbsolute_0.TransitionParameter[1] := 1.635;//blend参数，最小曲率半径
    //圆弧1
    MC_MoveCircularAbsolute_2.AuxPoint := GVL_Teach_Robot4.Recoder_Pos[21]; //point21
    MC_MoveCircularAbsolute_2.EndPoint := GVL_Teach_Robot4.Recoder_Pos[22]; //point22
    MC_MoveCircularAbsolute_2.Velocity := Vel; //Vel : 设置的速度
    //演示程序中加速度是速度的10倍，也可设置其他倍率
    MC_MoveCircularAbsolute_2.Acceleration := Vel * 10;
    MC_MoveCircularAbsolute_2.Deceleration := Vel * 10;
    //演示程序中加加速度是速度的100倍，也可设置其他倍率
    MC_MoveCircularAbsolute_2.Jerk := vel * 100;
    MC_MoveCircularAbsolute_2.BufferMode := 3; // 路径连接上一个FB
    MC_MoveCircularAbsolute_2.TransitionMode := 2; //距离混合
    MC_MoveCircularAbsolute_2.TransitionParameter[0] := 1.635;//blend参数,过渡半径,
    MC_MoveCircularAbsolute_2.TransitionParameter[1] := 1.635;//blend参数，最小曲率半径
    //直线2
    MC_MoveLinearAbsolute_1.Position := GVL_Teach_Robot4.Recoder_Pos[23];//point23
    MC_MoveLinearAbsolute_1.Velocity := Vel;//Vel : 设置的速度
    //演示程序中加速度是速度的10倍，也可设置其他倍率
    MC_MoveLinearAbsolute_1.Acceleration := Vel * 10;
    MC_MoveLinearAbsolute_1.Deceleration := Vel * 10;
    //演示程序中加加速度是速度的100倍，也可设置其他倍率
    MC_MoveLinearAbsolute_1.Jerk := Vel * 100;
    MC_MoveLinearAbsolute_1.BufferMode := 3;// 路径连接上一个FB
    MC_MoveLinearAbsolute_1.TransitionMode := 2;//距离混合
    MC_MoveLinearAbsolute_1.TransitionParameter[0] := 1.635;//blend参数,过渡半径,
    MC_MoveLinearAbsolute_1.TransitionParameter[1] := 1.635;//blend参数，最小曲率半径
    //圆弧2
    MC_MoveCircularAbsolute_3.AuxPoint := GVL_Teach_Robot4.Recoder_Pos[24]; //point24
    MC_MoveCircularAbsolute_3.EndPoint := GVL_Teach_Robot4.Recoder_Pos[25]; //point25
    MC_MoveCircularAbsolute_3.Velocity := Vel; //Vel : 设置的速度
    //演示程序中加速度是速度的10倍，也可设置其他倍率
    MC_MoveCircularAbsolute_3.Acceleration := Vel * 10;
    MC_MoveCircularAbsolute_3.Deceleration := Vel * 10;
    //演示程序中加加速度是速度的100倍，也可设置其他倍率
    MC_MoveCircularAbsolute_3.Jerk := vel * 100;
    MC_MoveCircularAbsolute_3.BufferMode := 3; // 路径连接上一个FB
    MC_MoveCircularAbsolute_3.TransitionMode := 2; //距离混合
    MC_MoveCircularAbsolute_3.TransitionParameter[0] := 1.635;//blend参数,过渡半径,
    MC_MoveCircularAbsolute_3.TransitionParameter[1] := 1.635;//blend参数，最小曲率半径
    //开始
    MC_MoveLinearAbsolute_0.Execute := TRUE;
    IF MC_MoveCircularAbsolute_3.Done THEN //MC_MoveCircularAbsolute_3.Done : 运动完成
        //复位控制信号
        MC_MoveLinearAbsolute_0.Execute := FALSE;
        bStarLC := FALSE;
    END_IF
END_IF
```

图 102

轨迹说明：

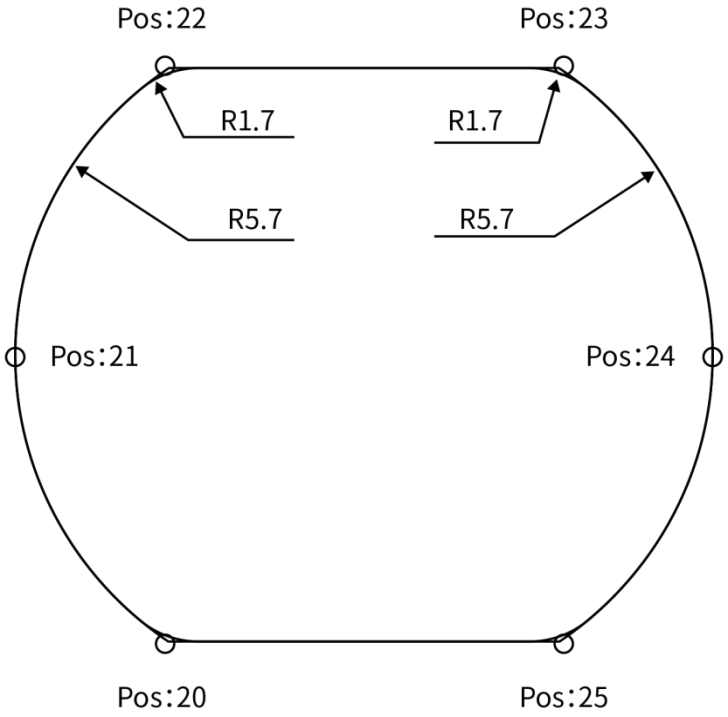


图 103

程序说明：

上图的轨迹示意图中，直线与圆弧有 R=1.7 的过渡段(blend 转弯半径 1.7mm)，示例程序机器人起始点和结束点都是 P25。

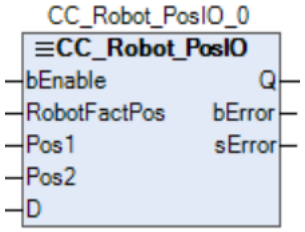
示例程序未调用工具坐标系或用户坐标系，具体调用方式与 4.7.1-a 连续整圆轨迹程序一致；

4.8 机器人点位跟踪输出

指令名称：CC_Robot_PosIO；

库：CC_Teach_Robot;

1) 指令格式

指令	名称	图形表现
CC_Robot_PosIO	机器人点位跟踪输出跟踪功能块	

2) 相关变量

◆ 输入变量

输入变量	数据类型	初始值	描述
bEnable	BOOL	FALSE	置 TRUE，开始跟踪，到达跟踪点后 Q 点置 TRUE； 置 FALSE，跟踪关闭，Q 点置 FALSE
RobotFactPos	AXIS_REF_SM3		机器人末端位置
Pos1	MC_COORD_REF		A 点位置，起始点
Pos2	MC_COORD_REF		B 点位置，目标点

D	LREAL	0	在 AB 线段中，离 B 点的距离，单位毫米。设置跟踪点

◆ 输出变量

输出变量	数据类型	初始值	描述
Q	BOOL	FALSE	在 bEnable 为 TRUE 后，到达跟踪点后置 TRUE
bError	BOOL	FALSE	功能块报错
sError	WSTRING		报错信息

3) 功能说明

上述功能块为跟踪机器人点位，输出控制信号，常用于连续轨迹动作中输出 I/O 或调用新功能块；

如下图所示，机器人从 A 点运动到 B 点，设置跟踪点 M。

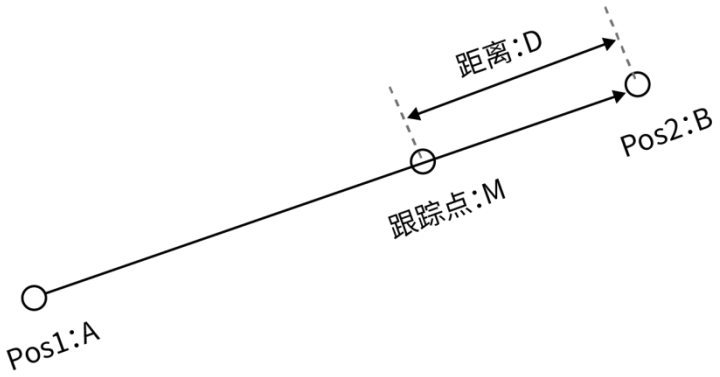


图 104

机器人在到达 M 点后，输出变量 Q 置 TRUE，直到输入脚 bEnable 为 FALSE，输出变量 Q 也恢复为 FALSE

4） 程序示例

以上图中的运动轨迹为例，编写示例程序：

先配置功能块：用 MC_GroupReadActualPosition 读取笛卡尔坐标下机器人的实时位置，把实时位置 Position 连接到 CC_Robot_PosIO 功能块的 RobotFactPos 输入脚。

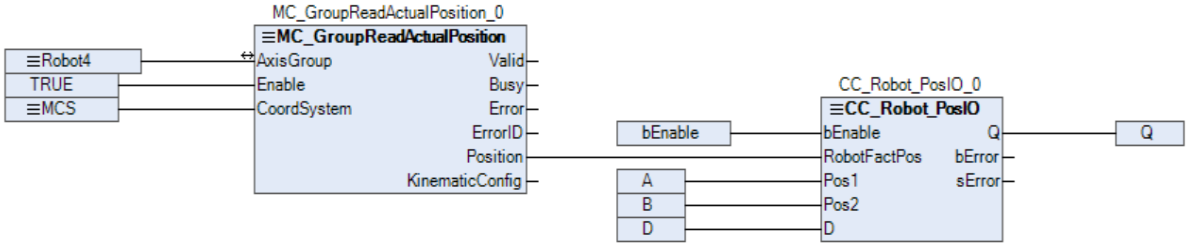


图 105

读取不是轴组的机器人的位置，比如 6 轴机器人，使用相应 FB_ReadActualPosition_6DOF 功能块，各型制机器人读取指令说明详见 4.9，使用 ST 语言写控制程序如下：

```
IF bStart THEN
    CASE iStep_IO OF
        0:
        //机器人上电使能
        IF GVL_Teach_Robot4.PowerDone THEN
            iStep_IO := 10;
        ELSE
            GVL_Teach_Robot4.PowerOn := TRUE;
        END_IF
        10:
        //机器人运动控制
        FB_MoveJBlend_Scara_L2_0.N := 2;
        FOR index := 1 TO 2 DO
            FB_MoveJBlend_Scara_L2_0.Position[index] := GVL_Teach_Robot4.Recoder_Pos[index];
            FB_MoveJBlend_Scara_L2_0.Velocity[index] := 10;
            FB_MoveJBlend_Scara_L2_0.Acceleration[index] := FB_MoveJBlend_Scara_L2_0.Velocity[index] * 10;
            FB_MoveJBlend_Scara_L2_0.Jerk[index] := FB_MoveJBlend_Scara_L2_0.Acceleration[index] * 100;
            FB_MoveJBlend_Scara_L2_0.tig[index] := 5;
        END_FOR
        FB_MoveJBlend_Scara_L2_0.bExecute := TRUE;
        iStep_IO := 20;
        20:
        //跟踪点位
        A := GVL_Teach_Robot4.Recoder_Pos[1];
        B := GVL_Teach_Robot4.Recoder_Pos[2];
        D := 10;
        bEnable := TRUE;
        IF Q THEN
            GVL_IO.Q1 := TRUE;
            GVL_IO.Q2 := TRUE;
            wstr := "机器人经过跟踪点";
            bEnable := FALSE;
        END_IF
        IF FB_MoveJBlend_Scara_L2_0.bDone THEN
            GVL_IO.Q1 := FALSE;
            GVL_IO.Q2 := FALSE;
            iStep_IO := 30;
        END_IF
        30://运动结束，关闭启动信号
        iStep_IO := 0;
        bStart := FALSE;
    END_CASE
END_IF
```

图 106

其中 bStart 为启动信号；

case 语句使程序按照程序步顺序执行，iStep_IO 就是程序步变量；

第 0 步：先机器人上电使能，引脚使用的是示教库功能块，详见 3.2 示教库功能块介绍及添加说明;

第 10 步：机器人的关节移动连续轨迹控制，详见 4.2 各型制机器人的关节移动连续轨迹控制。

第 20 步：启动点位跟踪，A 点与 B 点是示教点的 Point1 和 Point2，距离设置 10，然后通过输出点 Q，可以控制多个 IO 点或者其他的指令，示例中 IO 的 Q1，Q2 输出，同时字符串显示；在机器人到达了目标点 B 后，关闭了 Q1 和 Q2;

第 30 步，运动结束，关闭启动信号。

5) 注意事项

输出脚 bError 为 TRUE 为发生报错；SError 输出报错信息文本，有 2 种情况：“输入的两点重合”，即 AB 两点重合、“输入的两点间距过小”，即距离 D 大于 AB 两点间距。

本功能块仅检测可能执行到的轨迹中的点位并执行输出，连续轨迹（Blend）运动中因存在转弯圆弧半径，输入检测距离 D 须大于 tig 引脚设点转弯半径值（tig 设定详见 4.2/4.3 连续轨迹控制说明）。相应设置情况示例如下：

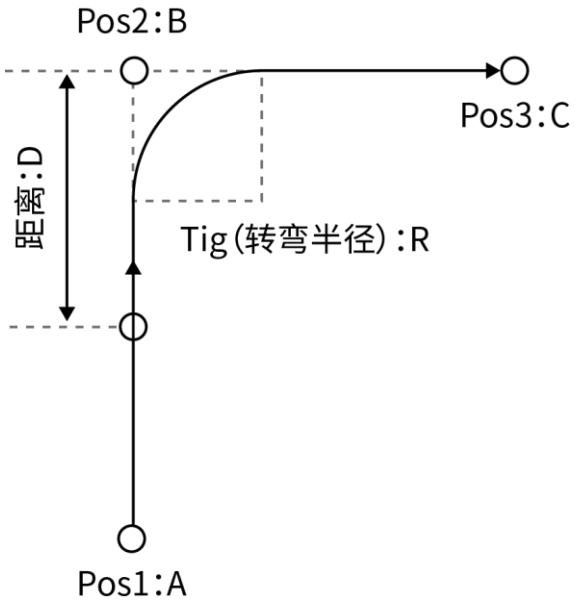


图 107

错误设置情况下，因机器人末端实际不经过检测点，功能块不报错但失效，相应设置情况示例如下：

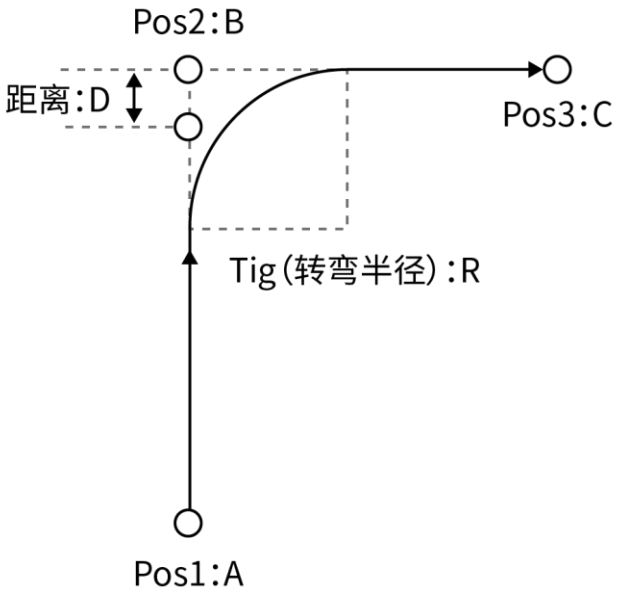


图 108

4.9 各型制机器人末端位置读取

4.9.1 4 轴 SCARA（L1）型机器人适用

指令名称：FB_ReadActualPosition_Scara4_L1

1) 指令格式

指令	名称	图形表现
FB_ReadActualPosition_Scara4_L1	四轴机器人位置读取功能块	FB_ReadActualPosition_Scara4_L1_0 FB_ReadActualPosition_Scara4_L1 Axis1 Position Axis2 Axis3 Axis4 IrArm1 IrArm2 Tool_x Tool_y Tool_z Tool_a Tool_b Tool_c CoordTransform CoordSystem Ratio34

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
Axis1	SM3_Basic.AXIS_REF_SM3		映射轴 1
Axis2	SM3_Basic.AXIS_REF_SM3		映射轴 2
Axis3	SM3_Basic.AXIS_REF_SM3		映射轴 3
Axis4	SM3_Basic.AXIS_REF_SM3		映射轴 4

◆ 输入变量

输入变量	数据类型	初始值	描述
lrArm1	LREAL	0	关节 1 长度/mm
lrArm2	LREAL	0	关节 2 长度/mm
Tool_X	LREAL	0	工具坐标 X 偏移
Tool_Y	LREAL	0	工具坐标 Y 偏移
Tool_Z	LREAL	0	工具坐标 Z 偏移
Tool_A	LREAL	0	工具坐标 A 偏移
Tool_B	LREAL	0	工具坐标 B 偏移
Tool_C	LREAL	0	工具坐标 C 偏移
CoordTransform	DWORD	MC_COORD_REF	用户坐标系
CoordSystem	LREAL	SMC_COORD_SYSTEM	坐标系选择
Ratio34	LREAL	0	3 轴与 4 轴的耦合比

◆ 输出变量

输出变量	数据类型	初始值	描述
Position	BOOL	MC_COORD_REF	机器人实时位置

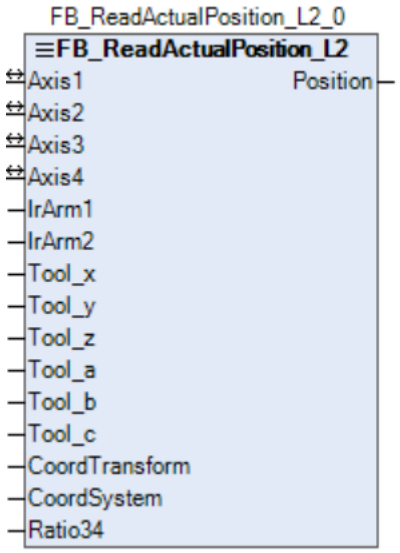
3) 功能说明

本功能块适用 4 轴 scara 且只有 1 和 2 轴为线性轴的机器人，读取实时位置。输出引脚 Position 接入需输入机器人末端位置的功能块。

4.9.2 4 轴 SCARA（L2）型机器人适用

指令名称：FB_ReadActualPosition_L2

1) 指令格式

指令	名称	图形表现
FB_ReadActualPosition_L2	四轴机器人位置读取功能块	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
Axis1	SM3_Basic.AXIS_REF_SM3		映射轴 1
Axis2	SM3_Basic.AXIS_REF_SM3		映射轴 2
Axis3	SM3_Basic.AXIS_REF_SM3		映射轴 3
Axis4	SM3_Basic.AXIS_REF_SM3		映射轴 4

◆ 输入变量

输入变量	数据类型	初始值	描述
lrArm1	LREAL	0	关节 1 长度/mm
lrArm2	LREAL	0	关节 2 长度/mm
Tool_X	LREAL	0	工具坐标 X 偏移
Tool_Y	LREAL	0	工具坐标 Y 偏移
Tool_Z	LREAL	0	工具坐标 Z 偏移
Tool_A	LREAL	0	工具坐标 A 偏移
Tool_B	LREAL	0	工具坐标 B 偏移
Tool_C	LREAL	0	工具坐标 C 偏移
CoordTransform	DWORD	MC_COORD_REF	用户坐标系
CoordSystem	LREAL	SMC_COORD_SYSTEM	坐标系选择
Ratio34	LREAL	0	3 轴与 4 轴的耦合比

◆ 输出变量

输出变量	数据类型	初始值	描述
Position	BOOL	MC_COORD_REF	机器人实时位置

3) 功能说明

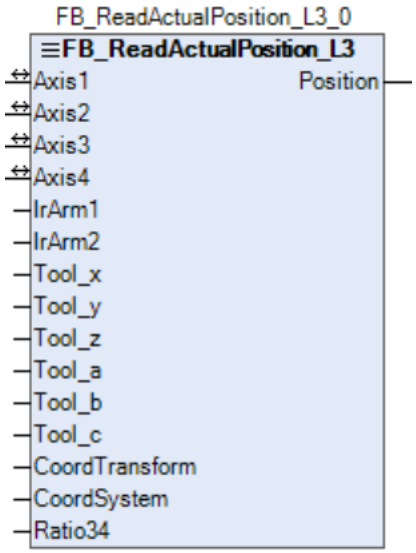
本功能块适合 4 轴 scara 并且只有第 2 轴为线性轴的机器人，读取实时位置。

输出引脚 Position 接入需输入机器人末端位置的功能块。

4.9.3 4 轴 SCARA（L3）型机器人适用

指令名称：FB_ReadActualPosition_L3

1) 指令格式

指令	名称	图形表现
FB_ReadActualPosition_L3	四轴机器人位置读取功能块	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
Axis1	SM3_Basic.AXIS_REF_SM3		映射轴 1
Axis2	SM3_Basic.AXIS_REF_SM3		映射轴 2
Axis3	SM3_Basic.AXIS_REF_SM3		映射轴 3
Axis4	SM3_Basic.AXIS_REF_SM3		映射轴 4

◆ 输入变量

输入变量	数据类型	初始值	描述
lrArm1	LREAL	0	关节 1 长度/mm
lrArm2	LREAL	0	关节 2 长度/mm
Tool_X	LREAL	0	工具坐标 X 偏移
Tool_Y	LREAL	0	工具坐标 Y 偏移
Tool_Z	LREAL	0	工具坐标 Z 偏移
Tool_A	LREAL	0	工具坐标 A 偏移
Tool_B	LREAL	0	工具坐标 B 偏移
Tool_C	LREAL	0	工具坐标 C 偏移
CoordTransform	DWORD	MC_COORD_REF	用户坐标系
CoordSystem	LREAL	SMC_COORD_SYSTEM	坐标系选择
Ratio34	LREAL	0	3 轴与 4 轴的耦合比

◆ 输出变量

输出变量	数据类型	初始值	描述
Position	BOOL	MC_COORD_REF	机器人实时位置

3) 功能说明

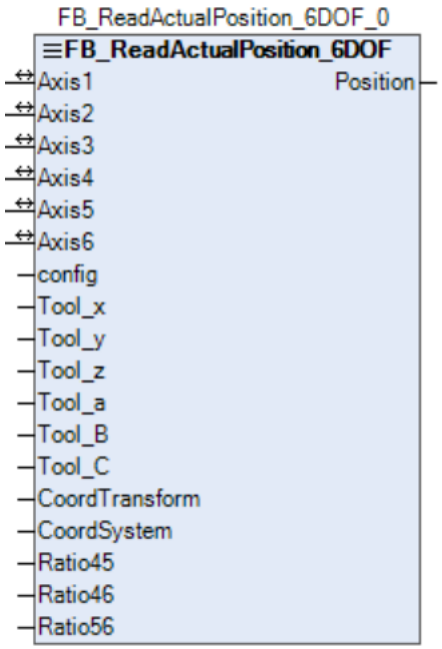
本功能块适合 4 轴 scara 并且只有第 3 轴为线性轴的机器人，读取实时位置。

输出引脚 Position 接入需输入机器人末端位置的功能块。

4.9.4 6 轴型机器人适用

指令名称：FB_ReadActualPosition_6DOF

1) 指令格式

指令	名称	图形表现
FB_ReadActualPosition_6DOF	六轴机器人位置读取功能块	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
Axis1	SM3_Basic.AXIS_REF_SM3		映射轴 1
Axis2	SM3_Basic.AXIS_REF_SM3		映射轴 2
Axis3	SM3_Basic.AXIS_REF_SM3		映射轴 3
Axis4	SM3_Basic.AXIS_REF_SM3		映射轴 4
Axis5	SM3_Basic.AXIS_REF_SM3		映射轴 5
Axis6	SM3_Basic.AXIS_REF_SM3		映射轴 6

◆ 输入变量

输入变量	数据类型	初始值	描述
config	LREAL	0	机器人 DH 参数
Tool_X	LREAL	0	工具坐标 X 偏移
Tool_Y	LREAL	0	工具坐标 Y 偏移
Tool_Z	LREAL	0	工具坐标 Z 偏移
Tool_A	LREAL	0	工具坐标 A 偏移
Tool_B	LREAL	0	工具坐标 B 偏移
Tool_C	LREAL	0	工具坐标 C 偏移
CoordTransform	DWORD	MC_COORD_REF	用户坐标系
CoordSystem	LREAL	SMC_COORD_SYSTEM	坐标系选择
Ratio45	LREAL	0	4 轴与 5 轴的耦合比
Ratio46	LREAL	0	4 轴与 6 轴的耦合比
Ratio56	LREAL	0	5 轴与 6 轴的耦合比

◆ 输出变量

输出变量	数据类型	初始值	描述
Position	BOOL	MC_COORD_REF	机器人实时位置

3) 功能说明

本功能块适合 6 轴机器人，读取实时位置。

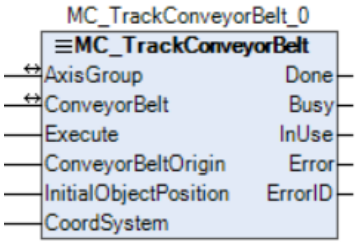
输出引脚 Position 接入需输入机器人末端位置的功能块。

4.10 附加轴同步运动控制

4.10.1 传送带（单一线性轴）跟踪

指令名称：MC_TrackConveyorBelt

1）指令格式

指令	名称	图形表现
MC_TrackConveyorBelt	传送带同步功能块	

2)相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
AxisGroup	AXES_Group_REF		轴组
ConveyorBelt	AXIS_REF		映射轴

◆ 输入变量

输入变量	数据类型	初始值	描述
Execute	BOOL		功能块启动
ConveyorBeltOrigin	MC_COORD_REF		传送带原点
InitialObjectPosition	MC_COORD_REF		工件初始位置
CoordSystem	SMC_COORD_SYSTEM		坐标系

◆ 输出变量

输出变量	数据类型	初始值	描述
Done	BOOL		完成后置 TRUE
Busy	BOOL		进行中置 TRUE
Inuse	BOOL		
Error	BOOL		发生错误 TRUE
ErrorID	SMC_ERROR		输出错误代码

3) 功能说明

本功能块用于在传送带皮带跟踪直线移动的对象，同时还进行 MCS 坐标系到 PCS 坐标系的动态计算。它不启动运动，但完成坐标系的变换。PCS 坐标系标定详见 3.3.7 用户坐标系示教窗口功能。

参数配置：

ConveyorBelt 是传送带皮带的驱动轴，可以是虚轴、编码器、实轴等。

ConveyorBeltOrigin 是规定传送带皮带相对于 MCS 坐标系的原点，实际上填入以传送带为参照标定的 PCS 的标定结果。

InitialObjectPosition 是传送带上物体相对于 PCS 原点的初始位置，实际上是对 PCS 的再一次偏移。

CoordSystem 为使用的坐标系，约定的坐标系需要是 PCS

用户坐标系的标定使用示教界面标定窗口完成后，得出的标定结果填入 ConveyorBeltOrigin。

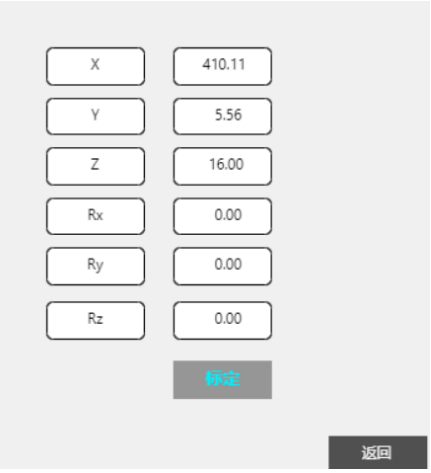


图 109

如需将工件在传送带起始感应到的位置作为 PCS 原点，在 InitialObjectPosition 填入工件起始点相对于标定的 PCS 原点的偏移值。即对 PCS 再进行一次以工件起始

点为原点的偏移。如起始点和原点重合或者不需要将起始点设置为 PCS 原点则填入(X:=0, Y:=0)即可。变量定义示例如下：

```
VAR
    MC_TrackConveyorBelt_0: SM3_Robotics.MC_TrackConveyorBelt;
    BeltOrigin: SM3_Robotics.MC_COORD_REF:=(X:=410.11, Y:=5.56, Z:=16,A:=0,B:=0,C:=0);
    InitialPos: SM3_Robotics.MC_COORD_REF:=(X:=0, Y:=0);
END VAR
```

使用说明：

启动：将 Execute 置 TRUE，并且 Done 反馈 TRUE 后，跟踪开启。此时将建立 CoordSystem 指定的 PCS 坐标系（注意：不需要额外使用 MC_SetCoordinateTransform 功能块切换为工件坐标系），并且坐标系跟着传送带轴运动方向实时进行偏移。 这时再使用其他运动功能块 MC_MoveLinearAbsolute,MC_MoveCircularAbsolute,MC_MoveDirectAbsolute 等运动机器人，机器人会相对于传送带静止进行轨迹运动。此时运动功能块的坐标系需要指定为 MC_TrackConveyorBelt 所设定的坐标系一致的 PCS 坐标系.如下所示：

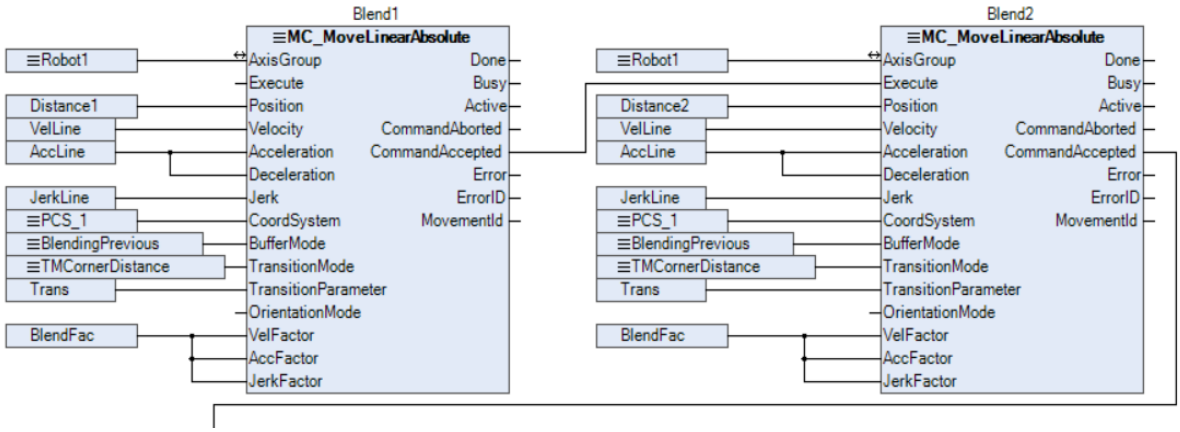


图 110

停止：不需要跟踪时，将 Execute 置为 FALSE。机器人运动指令的坐标系指定为 MCS 或 ACS 等（不能再为 PCS），机器人即会停止跟踪，按正常轨迹运动。

保护：在功能块开启时，机器人会跟随传送带移动。为避免传送带速度过快或其他故障导致功能块未及时关闭，机器人运动超过其运动范围，需要加限制强制关闭功能块。

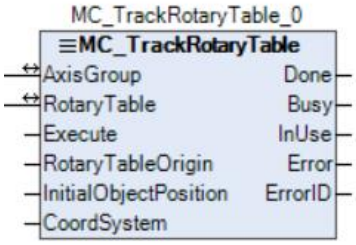
```
IF (Actual_Y >=300 OR Actual_Y <= -300)
    AND (Conveyor.MC_TrackConveyorBelt_0.Done) THEN
   bOnBelt:=FALSE;
   Teach_Robot1.bEmergStop:=TRUE;
END_IF
```

图 111

4.10.2 旋转平台（单一旋转轴）跟踪

指令名称：MC_TrackRotaryTable

1）指令格式

指令	名称	图形表现
MC_TrackRotaryTable	旋转平台跟踪功能块	

2)相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
AxisGroup	AXES_Group_REF		轴组
RotaryTable	AXIS_REF		映射轴

◆ 输入变量

输入变量	数据类型	初始值	描述
Execute	BOOL		功能块启动
RotaryTableOrigin	MC_COORD_REF		旋转台原点
InitialObjectPosition	MC_COORD_REF		工件初始位置
CoordSystem	SMC_COORD_SYSTEM		坐标系

◆ 输出变量

输出变量	数据类型	初始值	描述
Done	BOOL		完成后置 TRUE
Busy	BOOL		进行中置 TRUE
Inuse	BOOL		
Error	BOOL		发生错误 TRUE
ErrorID	SMC_ERROR		输出错误代码

3) 功能说明

本功能块用于旋转平台物体的跟踪同步。它将旋转平台驱动轴作为主轴，轴组同步主轴。与 4.8.1 说明 MC_TrackConveyorBelt 功能块被控对象时直线移动的传送带类似，MC_TrackRotaryTable 功能块被控对象时旋转平台。它也主要是完成从 MCS 到 PCS 的坐标变换，并不启动运动。

参数设置：

输入输出参数 RotaryTable 是旋转平台的驱动轴，可以是虚轴、编码器、实轴等。

RotaryTableOrigin 用于设置旋转平台的初始位置。

InitialObjectPosition 设置旋转平台上被控对象的位置，它是相对于选择平台的位置。

CoordSystem 为使用的坐标系，约定的坐标系需要是 PCS。

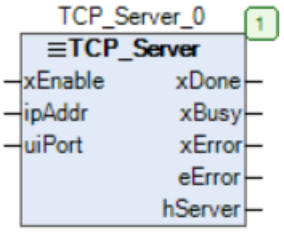
4.11 TCP/IP 通讯

控制器通过”CAA Net Base Services Library”库,实现 TCP/IP 通讯.

4.11.1 服务器

指令名称：TCP_Server ;

1）指令格式

指令	名称	图形表现
TCP_Server	TCP_服务器	

2)相关变量

◆ 输入变量

输入变量	数据类型	初始值	描述
xEnable	BOOL		功能块启动
ipAddr	IP_ADDR		服务器 IP 地址
uiPort	UINT		服务器端口号

◆ 输出变量

输出变量	数据类型	初始值	描述
xDone	BOOL		完成后置 TRUE
xBusy	BOOL		进行中置 TRUE
xError	BOOL		发生错误 TRUE
eError	ERROR		错误信息
hServer	CAA.HANDLE		

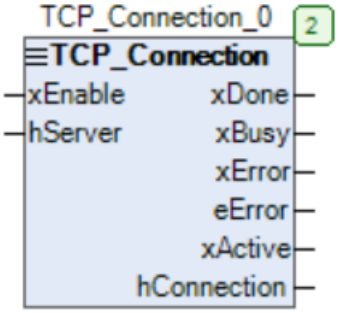
4) 功能说明

本功能块用于配置 TCP 服务器参数,配合 TCP_Connection 功能块使用.特别注意:输入 ipAddr 是结构体,需要把 IP 地址赋值给 sAddr.

4.11.2 TCP 连接

指令名称：TCP_Connection ;

1）指令格式

指令	名称	图形表现
TCP_Connection	服务器与客户端连接	

2)相关变量

◆ 输入变量

输入变量	数据类型	初始值	描述
xEnable	BOOL		功能块启动
hServer	CAA.HANDLE		与服务器的同名输出变量连接

◆ 输出变量

输出变量	数据类型	初始值	描述
xDone	BOOL		完成后置 TRUE
xBusy	BOOL		进行中置 TRUE
xError	BOOL		发生错误 TRUE
eError	ERROR		错误信息
xActive	BOOL		建立连接置 TRUE
hConnection	CAA.HANDLE		

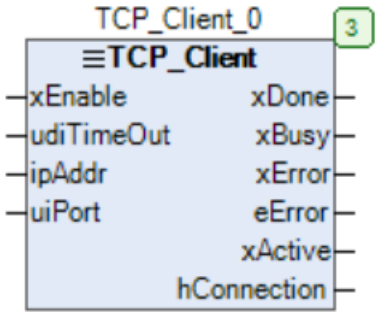
3) 功能说明

本功能块用于 TCP 服务器和客户端连接.

4.11.3 客户端

指令名称：TCP_Client ;

1）指令格式

指令	名称	图形表现
TCP_Client	TCP 客户端	

2)相关变量

◆ 输入变量

输入变量	数据类型	初始值	描述
xEnable	BOOL		功能块启动
udiTimeOut	UDINT		超时监控
ipAddr	IP_ADDR		服务器 IP 地址
uiPort	UINT		服务器端口号

◆ 输出变量

输出变量	数据类型	初始值	描述
xDone	BOOL		完成后置 TRUE
xBusy	BOOL		进行中置 TRUE
xError	BOOL		发生错误 TRUE
eError	ERROR		错误信息
xActive	BOOL		建立连接置 TRUE
hConnection	CAA.HANDLE		

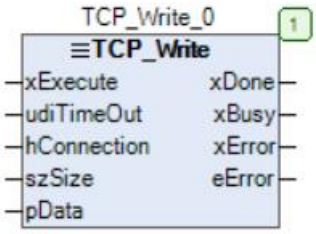
4) 功能说明

本功能块用于建立 TCP 客户端.特别注意:输入 ipAddr 是结构体,需要把 IP 地址赋值给 sAddr.

4.11.4 数据发送

指令名称：TCP_Write ;

1）指令格式

指令	名称	图形表现
TCP_Write	数据发送功能块	

2)相关变量

◆ 输入变量

输入变量	数据类型	初始值	描述
xExecute	BOOL		功能块启动,上升沿触发
udiTimeOut	UDINT		超时监控
hConnection	CAA.HANDLE		连接服务器或客户端的同名输出脚
szSize	CAA.SIZE		发送数据字节数,使用 sizeof 获取字节数
pData	CAA.PVOID		发送数据的地址

◆ 输出变量

输出变量	数据类型	初始值	描述
xDone	BOOL		完成后置 TRUE
xBusy	BOOL		进行中置 TRUE
xError	BOOL		发生错误 TRUE
eError	ERROR		错误信息

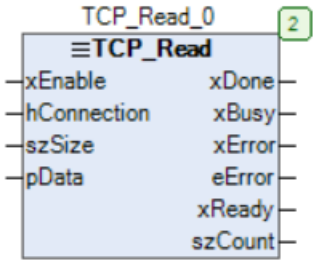
3) 功能说明

本功能块用于 TCP 数据发送.

4.11.5 数据接收

指令名称：TCP_Read ;

1）指令格式

指令	名称	图形表现
TCP_Read	数据接收功能块	

2)相关变量

◆ 输入变量

输入变量	数据类型	初始值	描述
xEnable	BOOL		功能块启动
hConnection	CAA.HANDLE		连接服务器或客户端的同名输出脚
szSize	CAA.SIZE		接收数据字节数,使用 sizeof 获取字节数
pData	CAA.PVOID		接收数据的地址

◆ 输出变量

输出变量	数据类型	初始值	描述
xDone	BOOL		完成后置 TRUE
xBusy	BOOL		进行中置 TRUE
xError	BOOL		发生错误 TRUE
eError	ERROR		错误信息
xReady	BOOL		
szCount	CAA.SIZE		

4) 功能说明

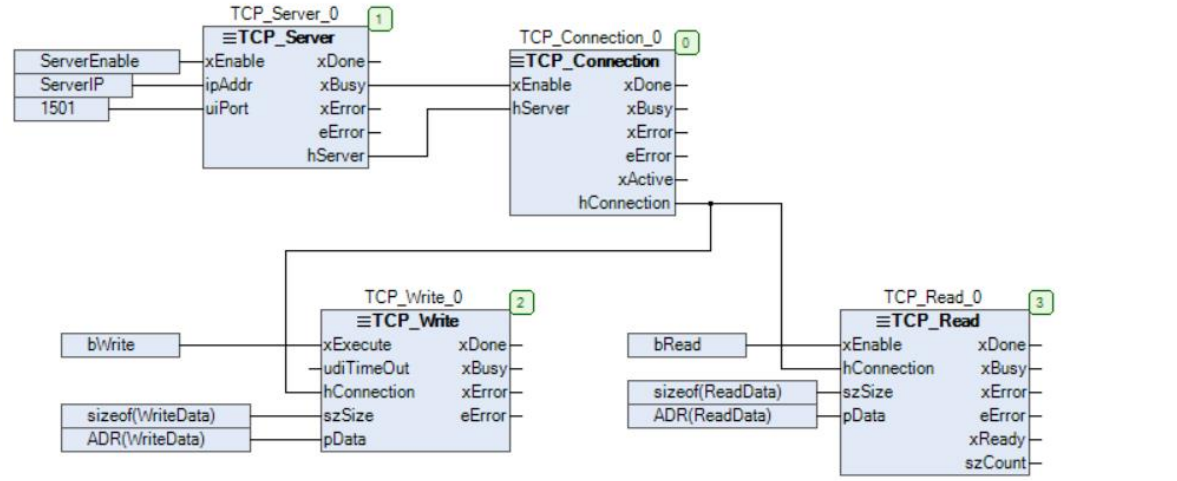
本功能块用于 TCP 数据接收.

4.11.6 示例程序

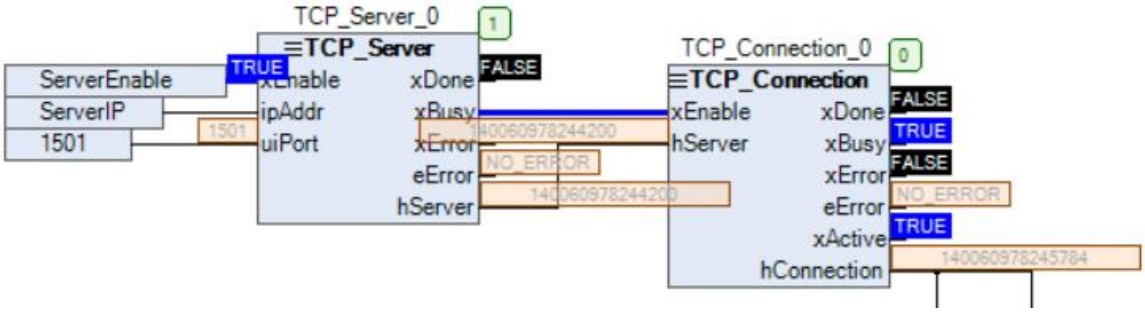
C²Cube 运动控制器做服务器,网络调试助手做客户端,发送接收字符串数据

程序如下图所示:

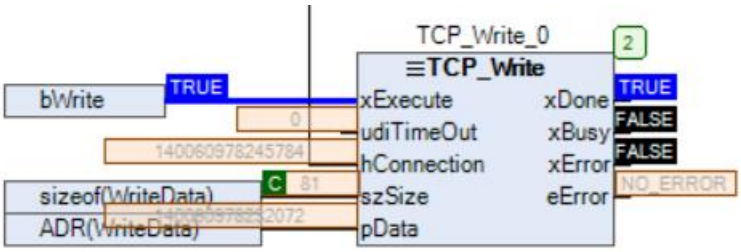
```
PROGRAM TCPIP
VAR
    //功能块实例化
    TCP_Server_0        : NBS.TCP_Server;
    TCP_Write_0         : NBS.TCP_Write;
    TCP_Read_0          : NBS.TCP_Read;
    TCP_Connection_0    : NBS.TCP_Connection;
    //功能块变量
    ServerIP            : NBS.IP_ADDR := (sAddr := '192.168.1.144'); //服务器IP地址
    ServerEnable        : BOOL; //开启服务器
    bWrite              : BOOL;//开启写入
    WriteData           : STRING;//写入数据
    bRead               : BOOL;//开始读取
    ReadData            : STRING;//读取数据
END_VAR
```



然后下载程序到 C²Cube 运动控制器, 然后 ServerEnable 置 True, 接着网络调试助手做客户端进行连接; 如下图所示连接成功.



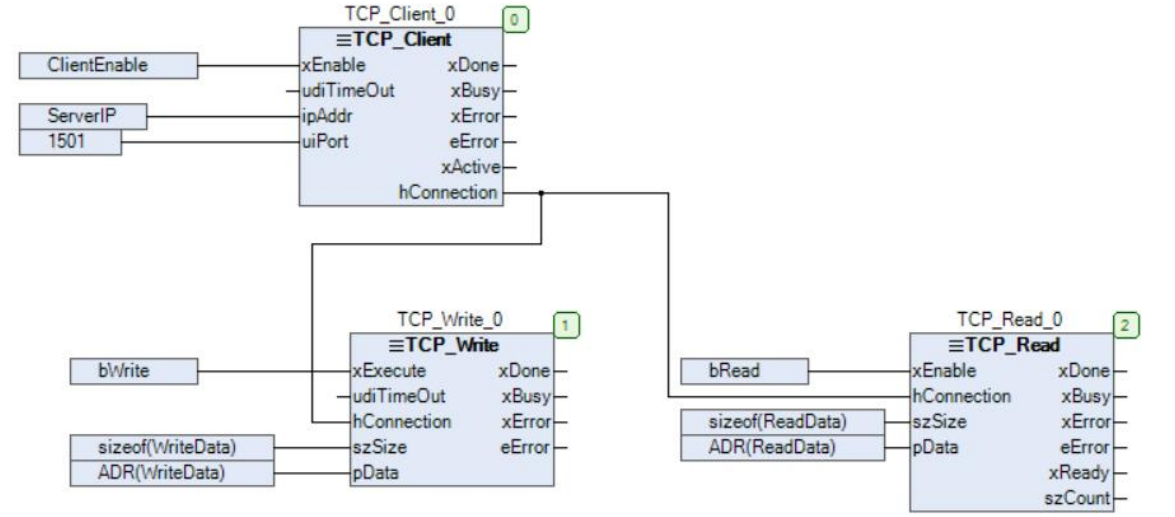
连接成功后,测试数据收发功能. 先 WriteData 赋值'C2CubeWriteTest',然后 bWrite 置 True,网络助手收到'C2CubeWriteTest',同时 TCP_Write 功能块的 xDone 变为 True,则发送成功.



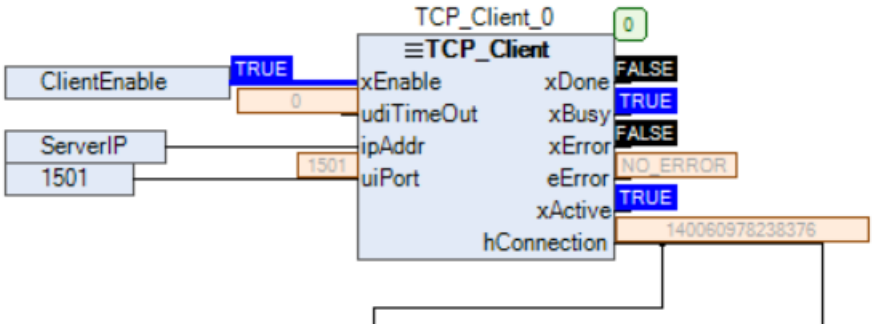
接着 bRead 置 True,网络助手发送 C2CubeReadTest, 然后查看 ReadData 变为'C2CubeReadTest',数据接收正常

C²Cube 运动控制器做客户端

```
PROGRAM TCPIP
VAR
    //功能块实例化
    TCP_Client_0        : NBS.TCP_Client;
    TCP_Write_0         : NBS.TCP_Write;
    TCP_Read_0          : NBS.TCP_Read;
    //功能块变量
    ServerIP            : NBS.IP_ADDR := (sAddr := '192.168.1.124'); //服务器IP地址
    ClientEnable        : BOOL; //开启服务器
    bWrite              : BOOL;//开启写入
    WriteData           : STRING;//写入数据
    bRead               : BOOL;//开始读取
    ReadData            : STRING;//读取数据
END_VAR
```



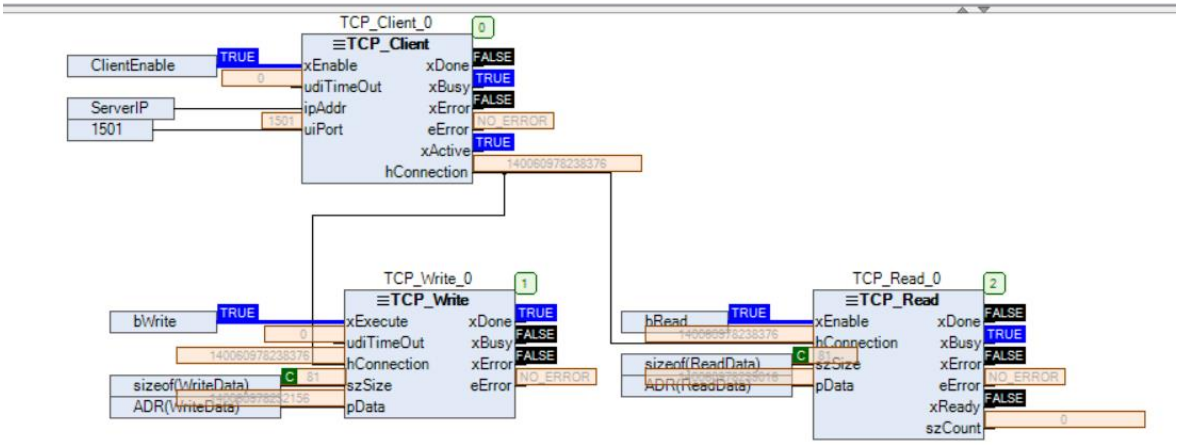
然后下载程序到 C²Cube 运动控制器,接着打开服务器,然后 ClientEnable 置 True, 如下图所示连接成功



连接成功后,测试数据收发功能. 先 WriteData 赋值'C2CubeWriteTest',然后 bWrite 置 True,服务器收到'C2CubeWriteTest',同时 TCP_Write 功能块的 xDone 变为 True,则发送成功.

接着 bRead 置 True,服务器发送 C2CubeReadTest, 然后查看 ReadData 变为'C2CubeReadTest',数据接收正常,如下图所示; 收发数据也可以使用其他的数据类型.

ClientEnable	BOOL	TRUE		开启服务器
bWrite	BOOL	TRUE		开启写入
WriteData	STRING	'C2CubeWriteTest'		写入数据
bRead	BOOL	TRUE		开始读取
ReadData	STRING	'C2CubeReadTest'		读取数据

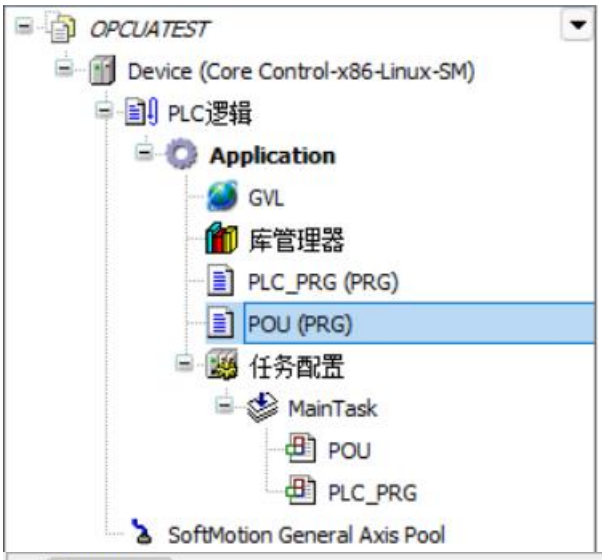


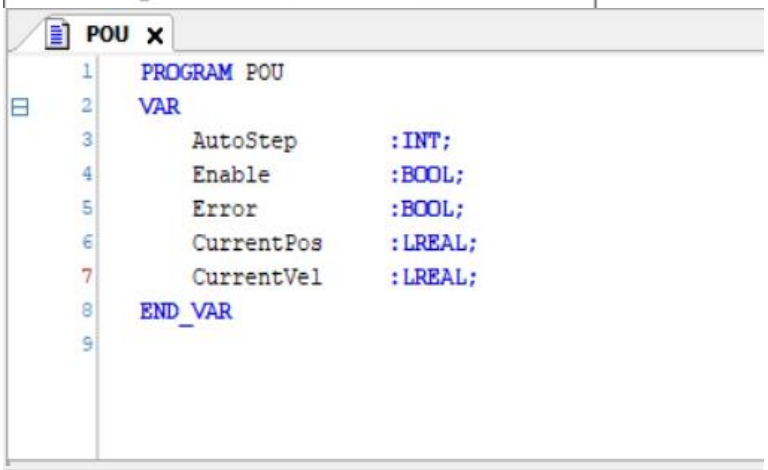
4.12 OPC UA 通讯

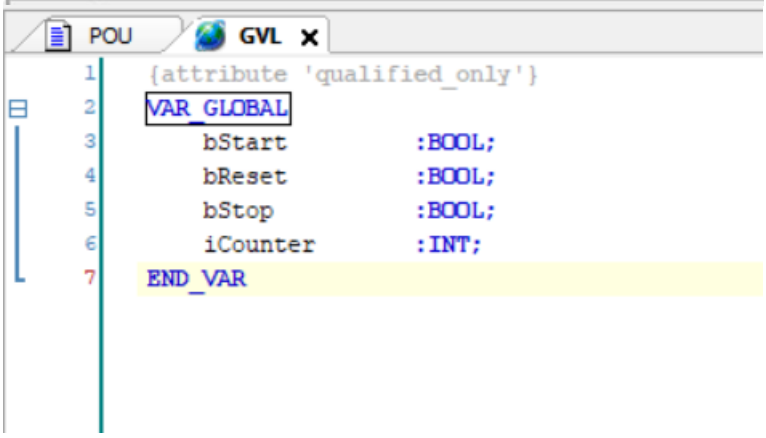
控制器支持 OPC UA 通讯处理，可建立 OPC UA 服务器并通过 TCP/IP 协议用于与触摸屏或其他 OPC UA 客户端通讯。

4.12.1 控制器侧建立 OPC UA 服务器

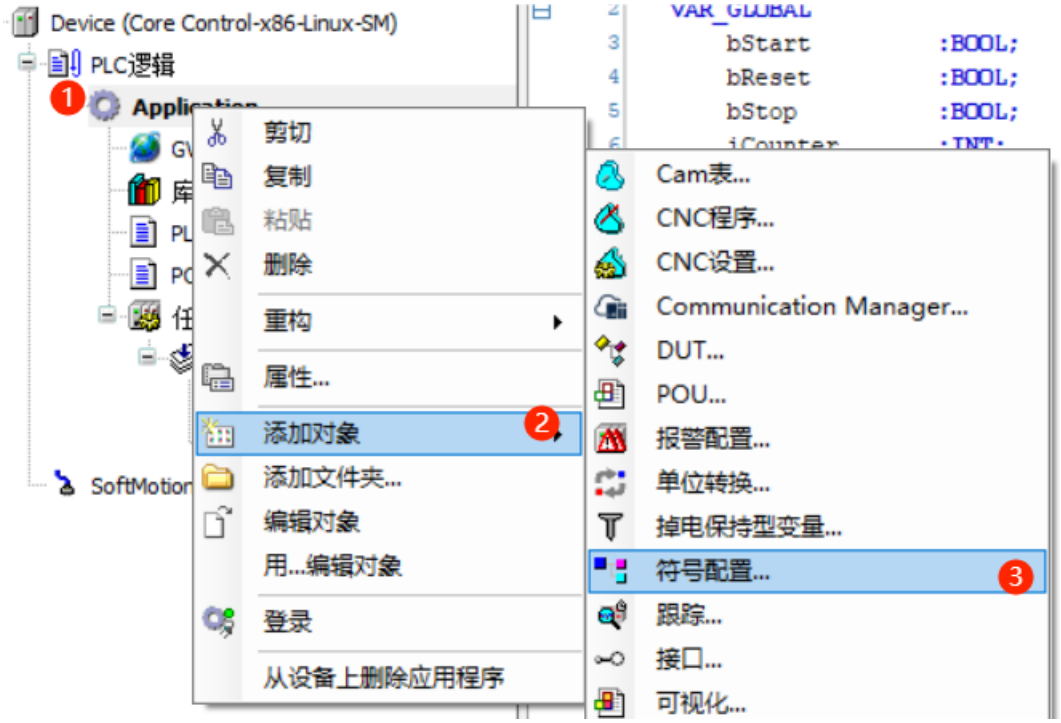
创建一个工程，新建一个 POU 在 POU 下声明局部变量，再建立一个全局变量 GVL 声明全局变量。（实际工程中新建的变量需要被程序引用，此处不做演示）。变量建立完成后在工具栏下点击“编译”-“生成代码”。



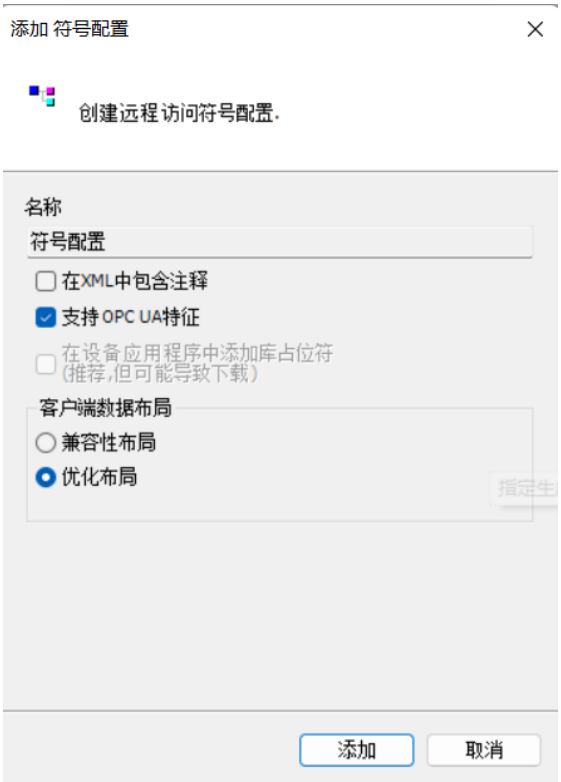




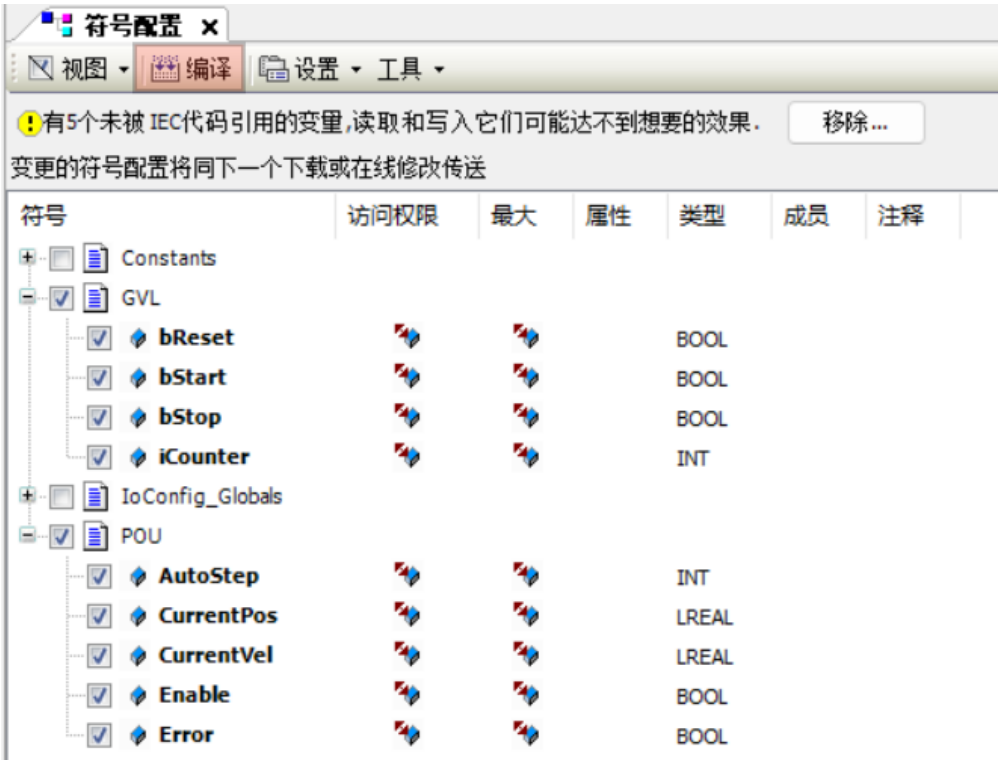
在工程 Application 对象下添加 “符号配置”对象。右键点击 Application-添加对象-符号配置。



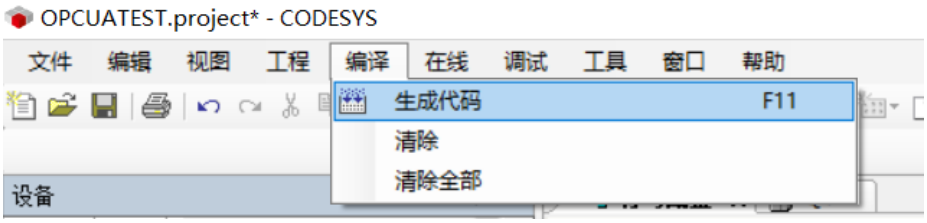
在弹出的符号配置对话框中勾选“支持 OPC UA 特征”，可选在 XML 中包含注释。

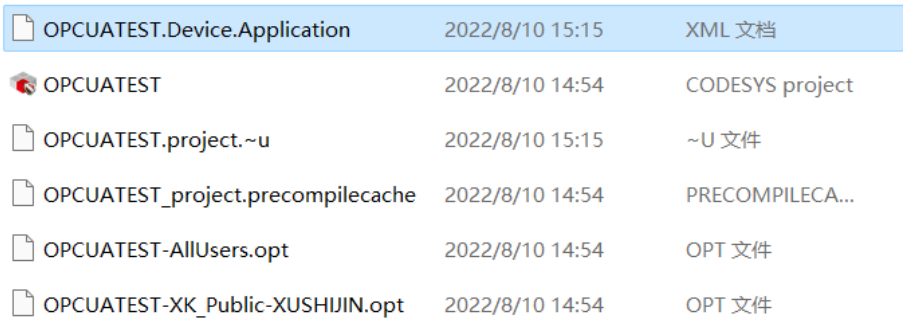


点击添加后，可在 Application 对象下找到生成的符号配置，双击打开可查看当前工程已经建立的全局变量和 POU 中建立的局部变量。点击符号配置框图下方的“编译”按钮，选中需要进行 OPC UA 通讯交换的变量，如图此处勾选上一步建立的全部变量。然后再次点击”编译”。



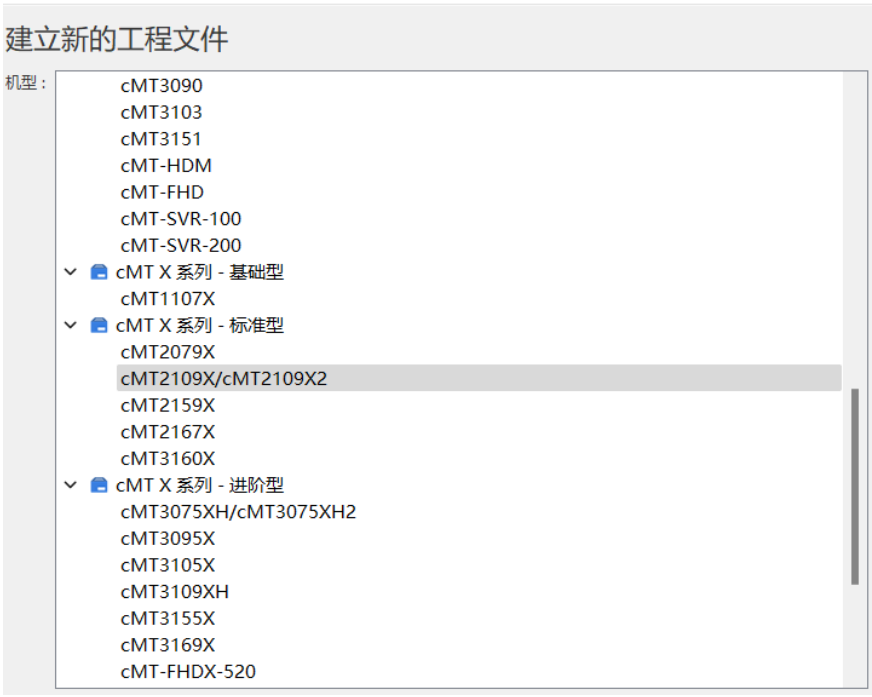
然后点击工具栏的“编译”-“生成代码”，即可在工程所在文件目录下生成一个 XML 文件。文件后缀名为“Device.Application”。



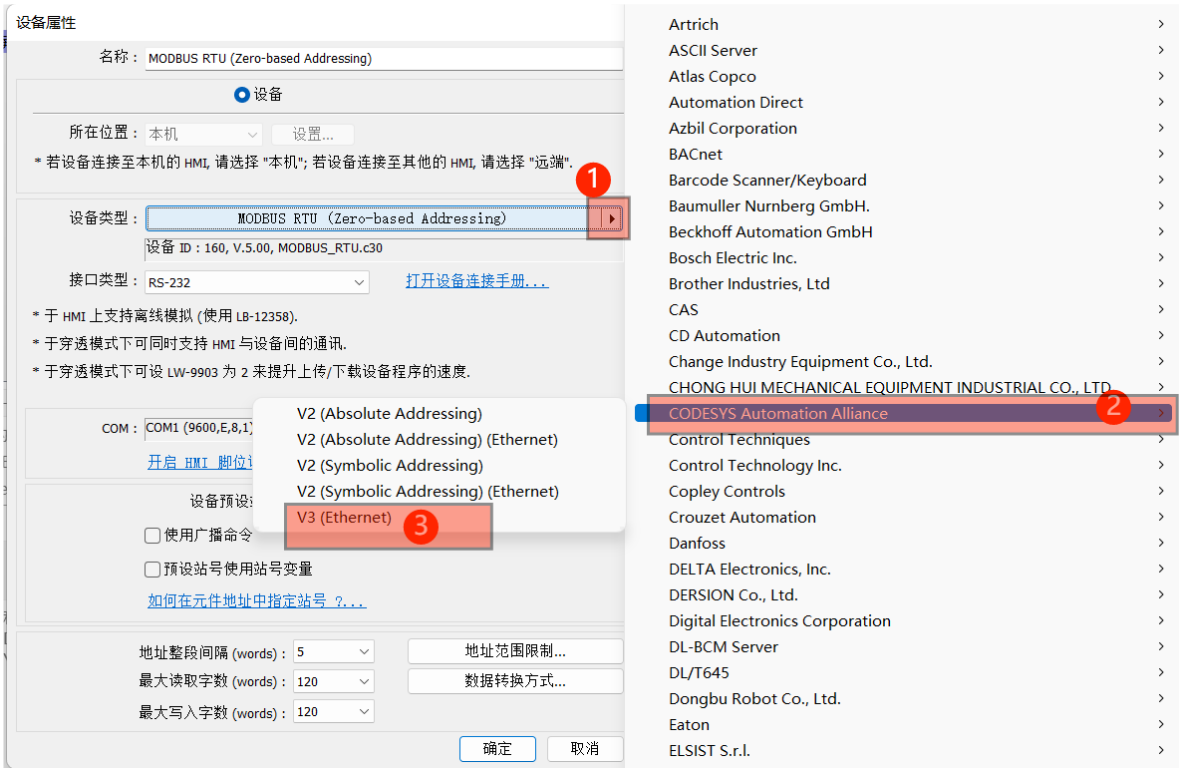


4.12.2 触摸屏侧建立 OPC UA 通讯

此处以威纶通型号 cM2109X/cM2109X2 触摸屏为例，打开威纶通 EBpro 软件，新建一个工程。选择对应的触摸屏型号。



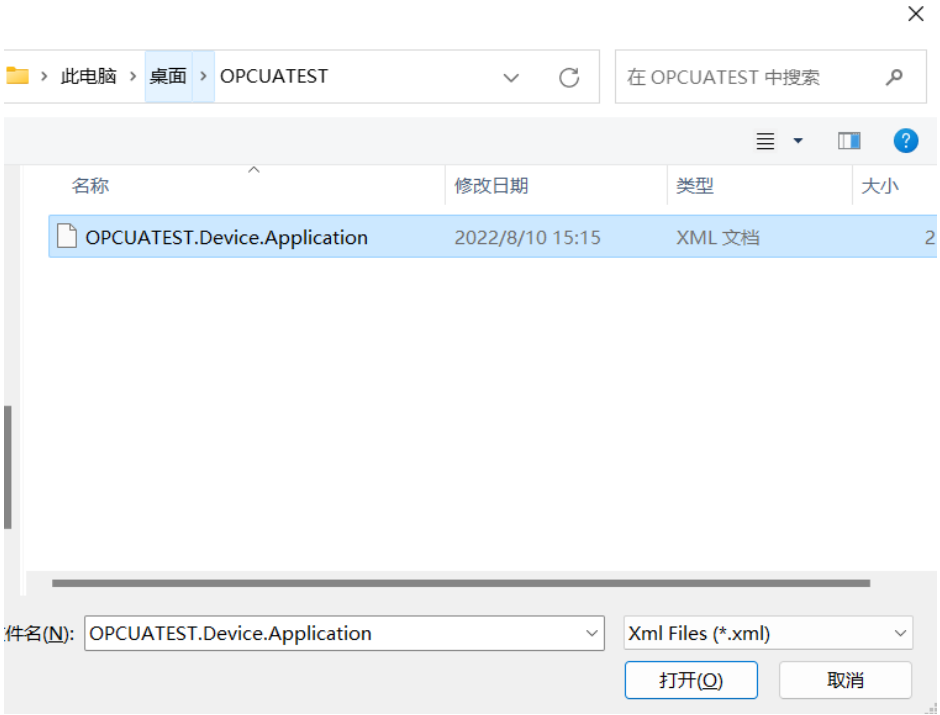
然后在系统参数设置对话框中点击“新增设备/服务器”。
在弹出的“设备属性”对话框中，点击“设备类型”按钮旁的小三角标。



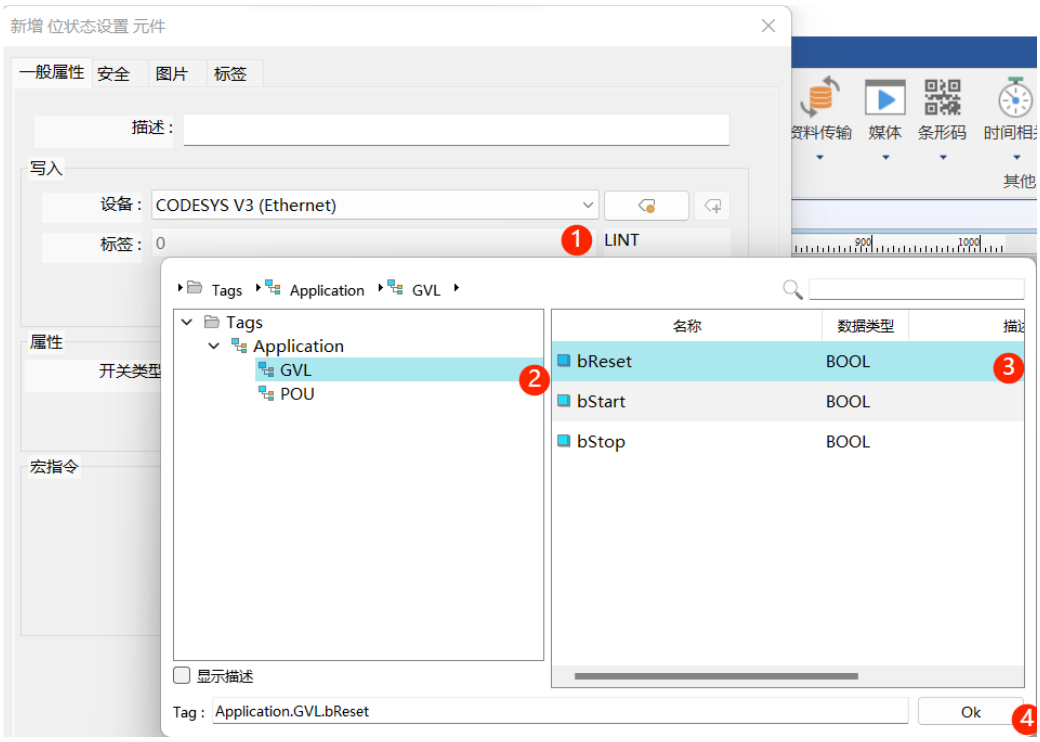
选择“CODESYS Automation Aliance”-“V3(Ethernet)”，可在对话框中设置控制器的 IP 地址和端口号。通讯协议修改为“V3 TCP/IP”。修改完成后确定退回到“系统参数设置”对话框。

选中添加的设备“CODESYS V3”，点击“导入标签”。然后在字符串类型选择对话框中选择类型方式，一般使用默认“一个字表示一个字符”即可，然后确定。

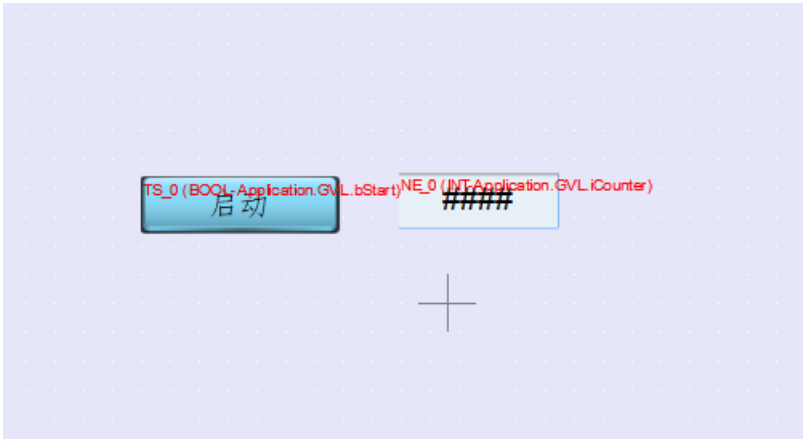
选择在 4.12.1 中在工程中生成的 XML 标签文件。点击打开后，会提示“成功导入标签”。至此控制器和触摸屏 OPC UA 通讯建立成功。



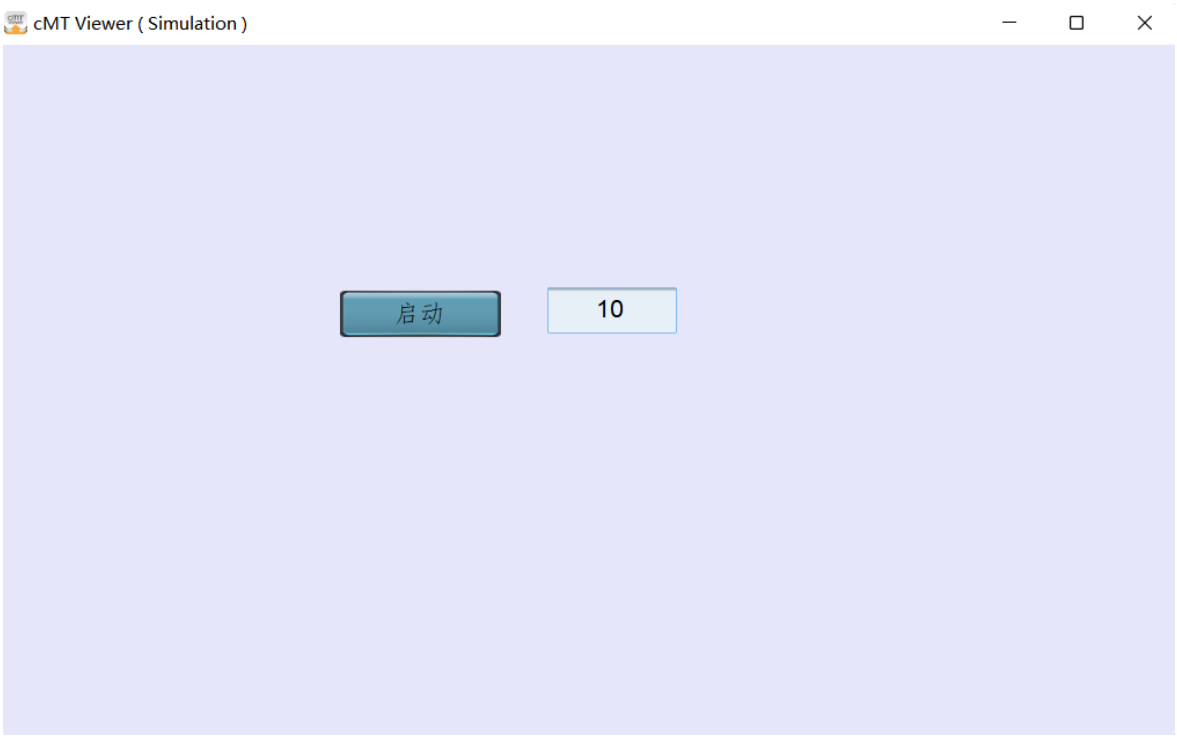
在 EBpro 中建立一个位状态切换元件，设置为工程中全局变量 GVL-bStart。



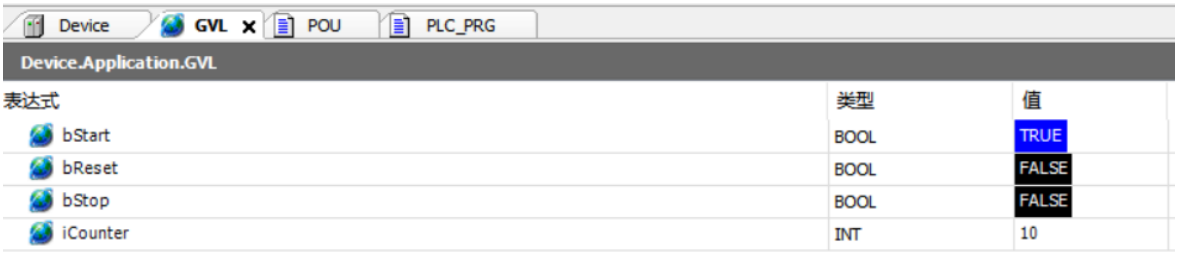
同样再创建一个数值元件，关联全局变量 GVL-iCounter。



将触摸屏工程下载进入实体触摸屏（或用在线模拟模式）。同时将 CODESYS 工程下载进入控制器。



在仿真项目中点击按钮，输入数值，然后在 CODESYS 中观察变量状态。



表达式	类型	值
bStart	BOOL	TRUE
bReset	BOOL	FALSE
bStop	BOOL	FALSE
iCounter	INT	10

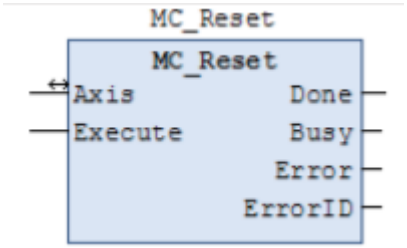
五、常用单轴指令详解

5.1 轴错误状态复位

名称：MC_Reset

功能简述：轴错误状态状态复位

1) 指令格式

指令	名称	图形表现	ST 表现
MC_Reset	轴错误状态复位指令		MC_Reset(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) 相关变量

◆ **输入输出变量**

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	——	——	映射到 AXIS_REF_SM3 的 1 个实例

◆ **输入变量**

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE，FALSE	FALSE	输入一个上升沿将启动功能块的处理

◆ 输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE，FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE，FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE，FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

在本功能块在轴通讯正常的情况下，把轴状态处于 errorstop 变为 Standstill, 把轴的异常状态变为正常可运行的状态；当出现轴 errorstop 无法复位，Axis.bCommunication 为 FLASE 状态，必须要重新建立主站和从站轴的通讯；在指令中的 Busy 标志位接通的时间非常短，使用时请注意；

◆ 时序图

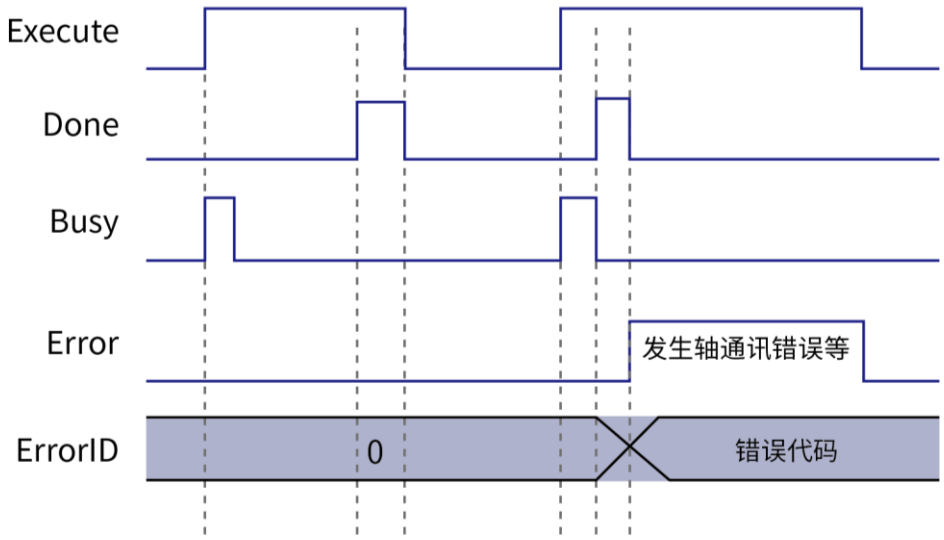


图 112

4) 注意事项

本功能块适合所有轴运动控制。

5.2 加速度轮廓

指令名称：MC_AccelerationProfile

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ AccelerationProfile	加速度轮廓指令		MC_AccelerationProfile(Axis:= , TimeAcceleration:= , Execute:= , ArraySize:= , AccelerationScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例
TimeAcceleration	轴加速时间和加速度描述	MC_TA_ REF			轴加速时间和加速度数据描述 , 加速数据由多组数据组成

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
ArraySize	动态数组	INT	数据范围	0	运行轮廓中使用的数组个数
AccelerationScale	综合因子	LREAL	“正数”+”0”	1	MC_TA_REF 中加速度或减速度的比例因子
Offset	偏移	LREAL		0	加减速度的总体偏移值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为时间段和加减速度的轮廓运动模型,运行模式为“Discrete Motion”, 按用户在“TimeAcceleration”变量中设定的数据运行。

本功能块运行状态为“Standstill”中,指令运行时的状态为“Discrete Motion”, 其他状态无法运行。

启动指令为“Execute”的上升沿启动,本指令在“Discrete Motion”重复运行速度都是在上一次的叠加,容易引起系统故障。

TimeAcceleration 为 MC_TA_REF 数据类型;

MC_TA_REF 具体描述如下:

成员	类型	初始值	描述
Number_of_pairs	INT	0	轮廓路径的段数
IsAbsolute	BOOL	TRUE	绝对运动 (TRUE) 和相对运动选择
MC_TA_Array	ARRAY[1..N] OF SMC_TA		时间和加速值的数组

SMC_TA 具体描述如下:

成员	类型	初始值	描述
delta_time	TIME	TIME#0ms	加速度段的时间
acceleration	LREAL	0	当前的加速度值

注:设置的加速度体现在速度的变化上,所有的加速度变化按 S 曲线的方式变化,从最终的结果变化为[起始加速度为 A 终止加速度为 B] (A+B)/2 的加速度数据体现在最终的速度上;

◆ 时序图

条件 MC_TA_Array 通过其他方式已经设置；轴必须处于 Standstill 状态指令才能运行；功能块的 Execute 必须有上升沿的条件；功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

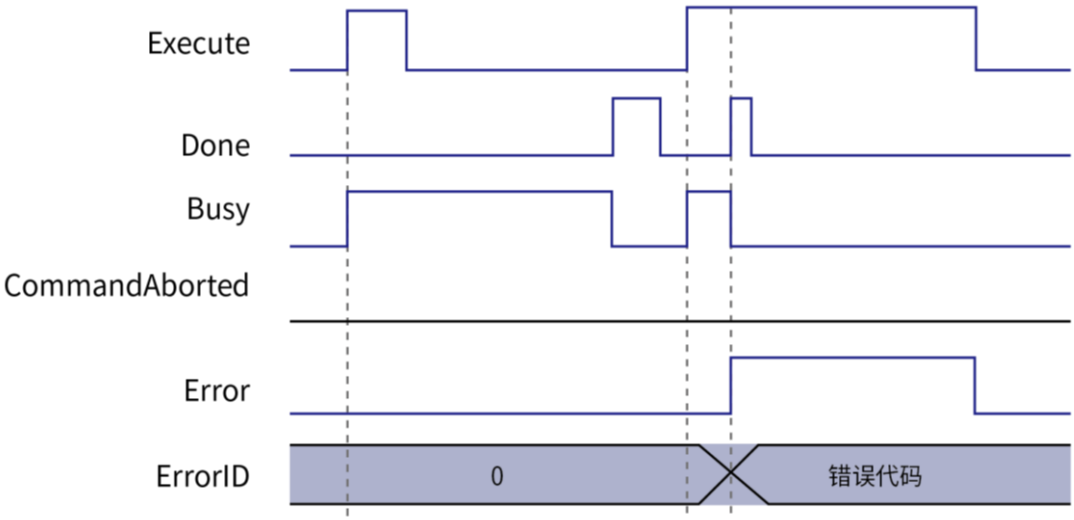


图 113

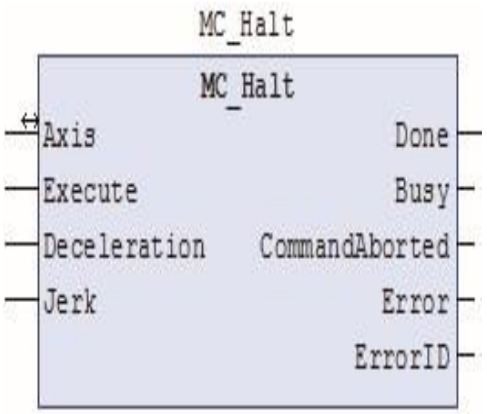
4) 注意事项

错误的出现为轴状态不是在 Standstill 中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

5.3 轴暂停

指令名称：MC_Halt

1) 指令格式

指令	名称	图形表现	ST 表现
MC_Halt	轴正常停止命令		MC_Halt(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Deceleration	减速度	LREAL	“正数”+”0	” 0	功能块的减速度 (u/S^2)
Jerk	跃度	LREAL	“正数”+”0	” 0	指定跃度［指令单位 /S^3］

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAborted	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为在正常运行的情况下停止一个轴的运动，而其它一个运动轴指令再运行时可以终止本指令的运行。

本功能块运行状态为运行状态 (Motion) 才能运行 , 其他状态无法运行。

启动指令为 Execute 的上升沿启动 ;

指令运行中的状态为 Discrete Motion, 运行完成后状态为 Standstill。

◆ 时序图

轴必须处于运行状态 (Motion) 指令才能运行；

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

功能块的 CommandAborted 表示指令被其他运动控制指令中断，此时标志位为 TRUE；

例程 : 在执行 MC_MoveVelocity 指令和 MC_Halt 指令在不同的时序操作中对应的标志位的变化 ; 对 CommandAborted 的处理描述如下图的时序描述。

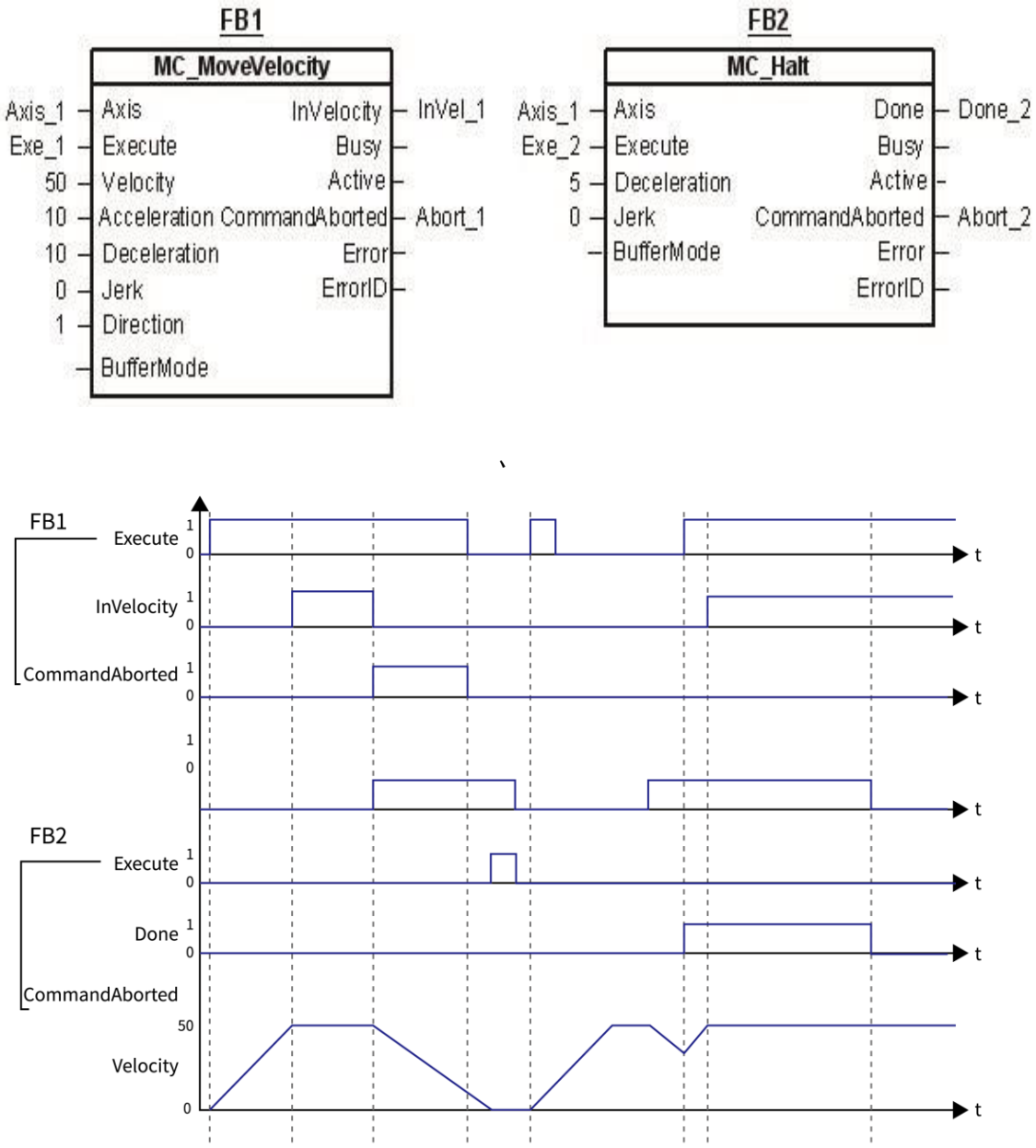


图 114

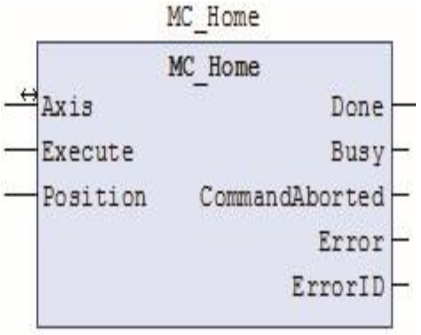
4) 注意事项

错误的出现为轴状态不是在 Standstill 中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

5.4 轴回零

指令名称：MC_Home

1) 指令格式

指令	名称	图形表现	ST 表现
MC_Home	轴回零指令		MC_Home(Axis:= Axis, Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Position	轴到达位置	LREAL	数据范围	0	代表轴位置的回零位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为回零操作 , Position 数据位轴的零点位置。

本功能块运行状态为 Standstill 中 , 指令运行时的状态为 homing, 其他状态无法运行。

启动指令为 Execute 的上升沿启动指令。

◆ 时序图

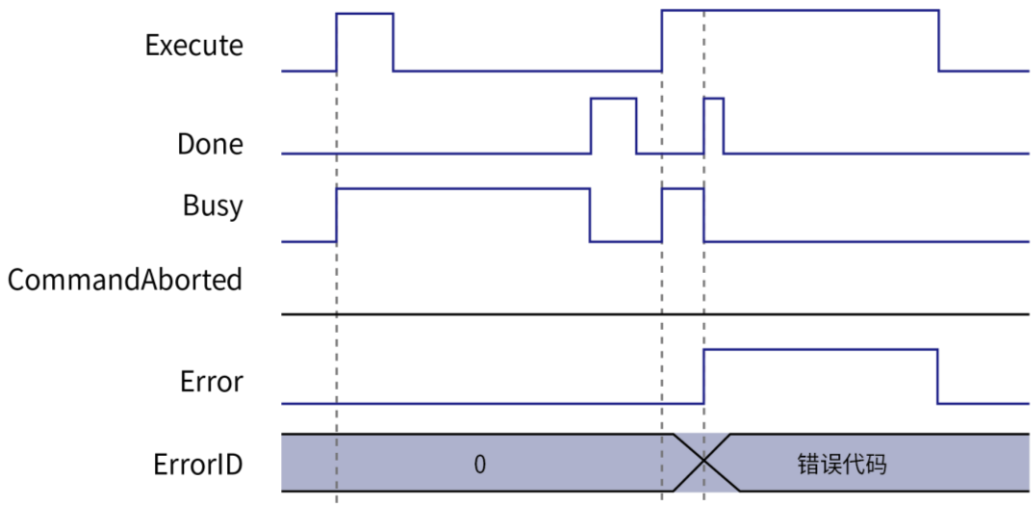
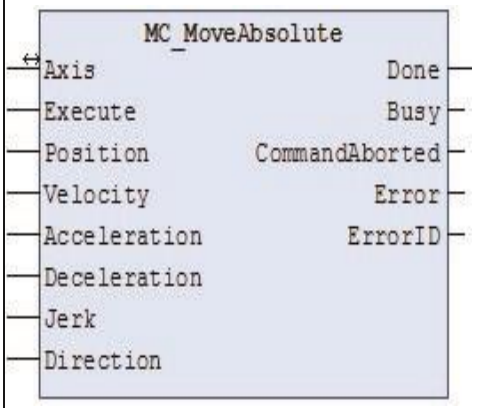


图 115

5.5 轴绝对位置控制

指令名称：MC_MoveAbsolute

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ MoveAbsolute	轴绝对位置控制指令		MC_MoveAbsolute(Axis:= , Execute:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Position	轴到达位置	LREAL	数据范围	0	此位置为轴的绝对位置数据
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度

Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值
Direction	指令极性	MC_ DIRECTION	Negative,shortest Positive, current,fastest	shortest	Negative: 反向移动 ; Shortest: 根据最短路径选择方向 ; Positive: 正向移动 ; Current: 按当前方向移动 ; Fastest: 自动选择最快的方向移动 ; (此功能轴在旋转模式下有效)

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE

ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为轴绝对定位指令,Position 数据为轴的绝对位置。

本指令运行前设置好相关的参数 , 加速度(Acceleration)、减速度 (Deceleration)、运行速度 (Velocity) 和加减速模式的跃度 (Jerk)；对加速度 (Acceleration) 或减速度 (Deceleration) 的赋值为 0

本功能块运行状态为 Standstill 中 , 指令运行时的状态为 Discrete Motion, 一个完整的运行过程一定要控制轴的不同运动状态。

启动指令为 Execute 的上升沿启动,本指令在 Discrete Motion 可以重复上升沿有效 , 每次都可以刷新最新的 Position 位置。

Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

梯形加减速动作：Velocity、Acceleration 和 Deceleration 有数据，而 Jerk 为 0，速度/位置曲线如下：

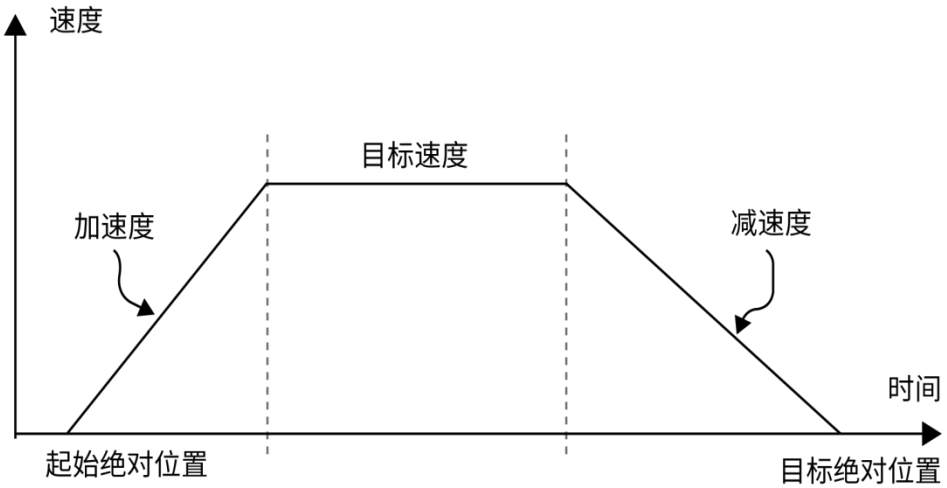


图 116

S 曲线加减速动作：Velocity、Acceleration、 Deceleration 和 Jerk 有数据，速度/位置曲线如下：

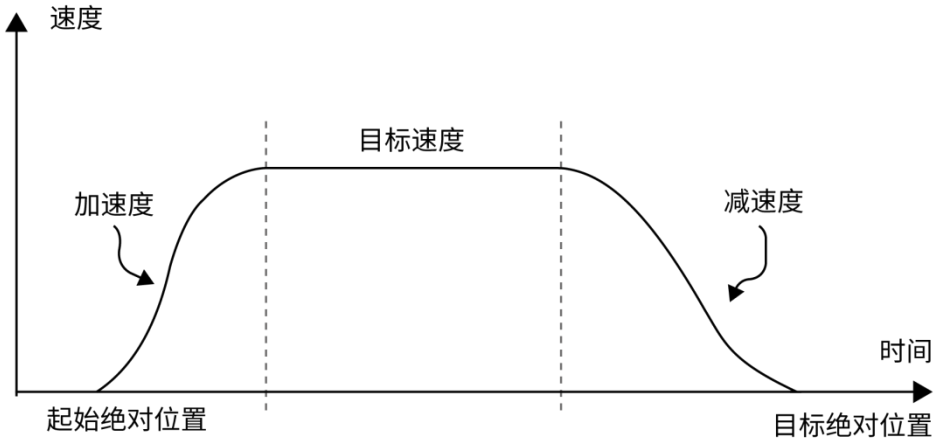


图 117

◆ 轴在周期模式下绝对定位

轴旋转周期设置为 360、Direction 设置为 Positive。

当 Position 对 360 的模值（Position/360 取余，比如 Position 为 380 则对 360 的模值为 20，Position 为 350 则对 360 模值为 350）>起始绝对位置时，此时轴朝正向运行 Position 对 360 的模值-起始绝对位置的距离。

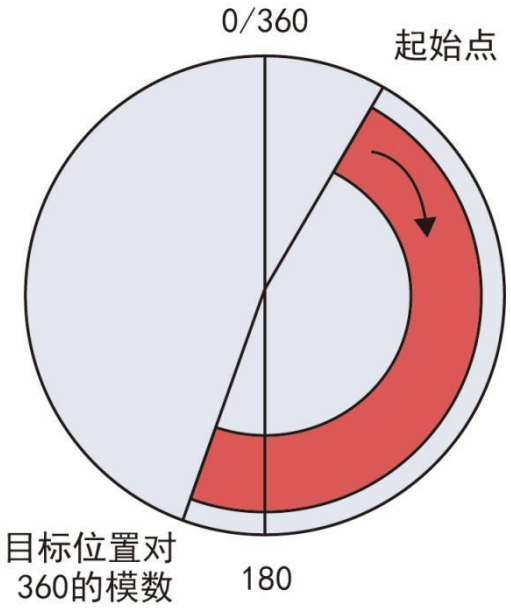


图 118

当 Position 对 360 的模值（Position/360 取余，比如 Position 为 380 则对 360 的模值为 20）< 起始绝对位置时，此时轴朝正向运行 360- 起始绝对位置 + Position 对 360 的模值的距离。

1) 轴旋转周期设置为 360、Direction 设置为 shortest 或者 fastest。Position 对 360 的模值为 XPosition 当 =0<XPosition- 起始绝对位置 <180 时，轴朝正向

运行 XPosition- 起始绝对位置的距离。

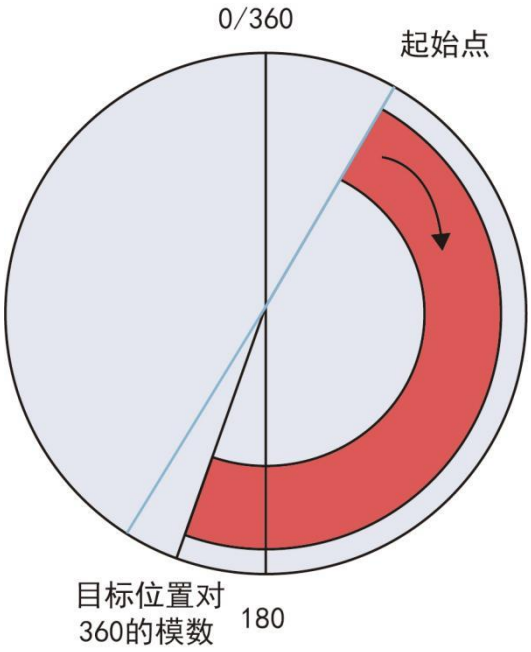


图 119

2) 当 180<XPosition- 起始绝对位置时，轴朝反向运行 360-XPosition+ 起始绝对位置的距离。

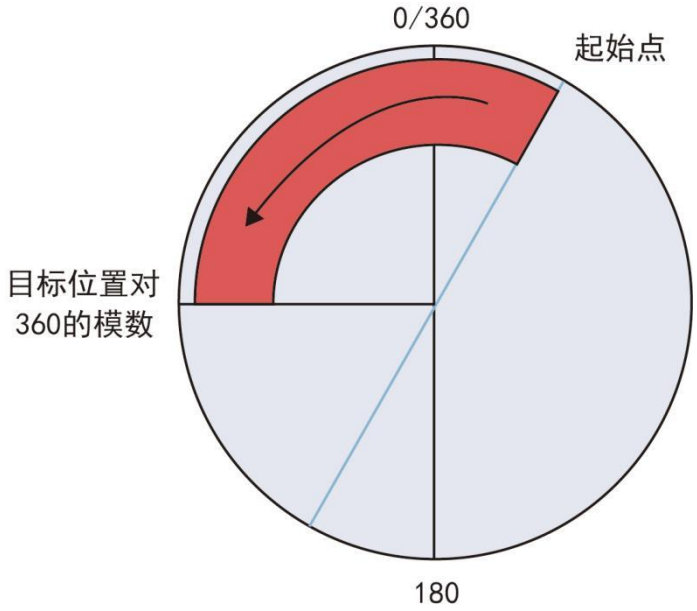


图 120

3) 当 XPosition< 起始绝对位置时，轴朝反向运行起始绝对位置 -XPosition 的距离。

MC_MoveAbsolute

目标位置对

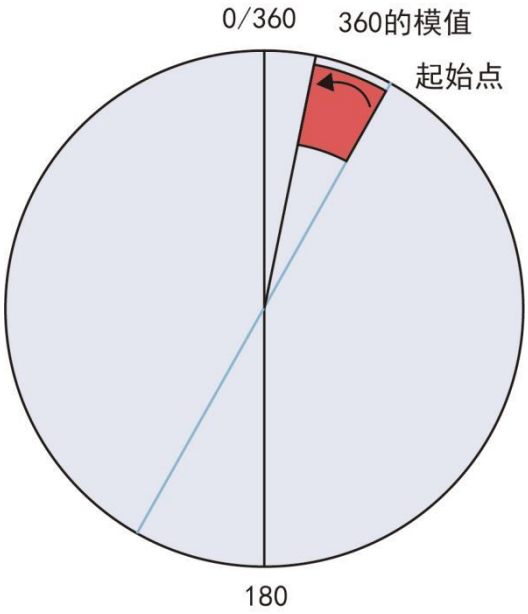


图 121

4) 轴旋转周期设置为 360、Direction 设置为 shortest 或者 Negative。Position 对 360 的模值为 XPosition 轴朝反向运行起始绝对位置 +360-XPosition 的距离。

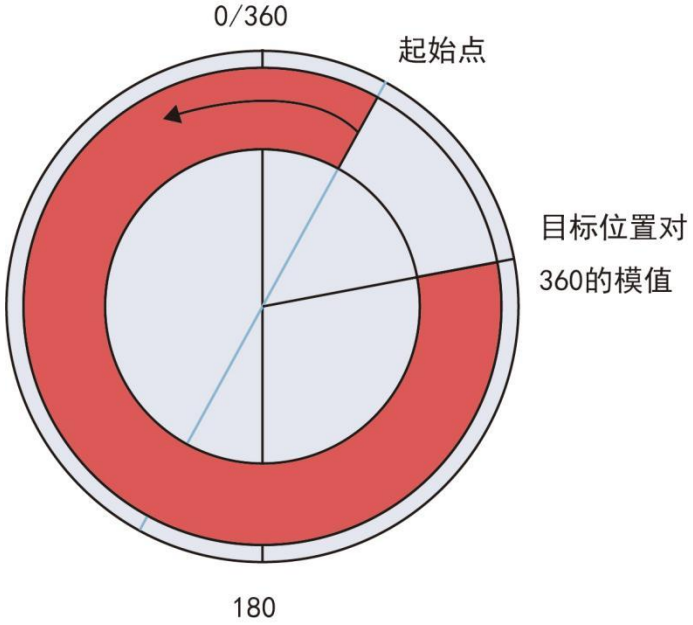


图 122

◆ 轴在线性模式下绝对定位

当目标绝对位置 > 起始位置，则正向移动目标绝对位置 - 起始位置距离，当目标位置 < 起始位置时则反向移动起始位置 - 目标位置的距离。线性模式下设定的运行方向不决定轴运行方向。

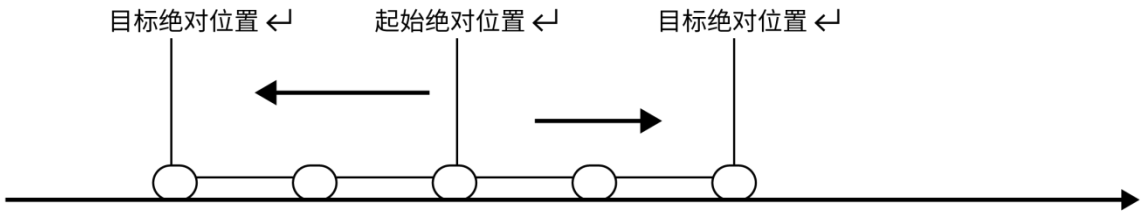


图 123

◆ 时序图

轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

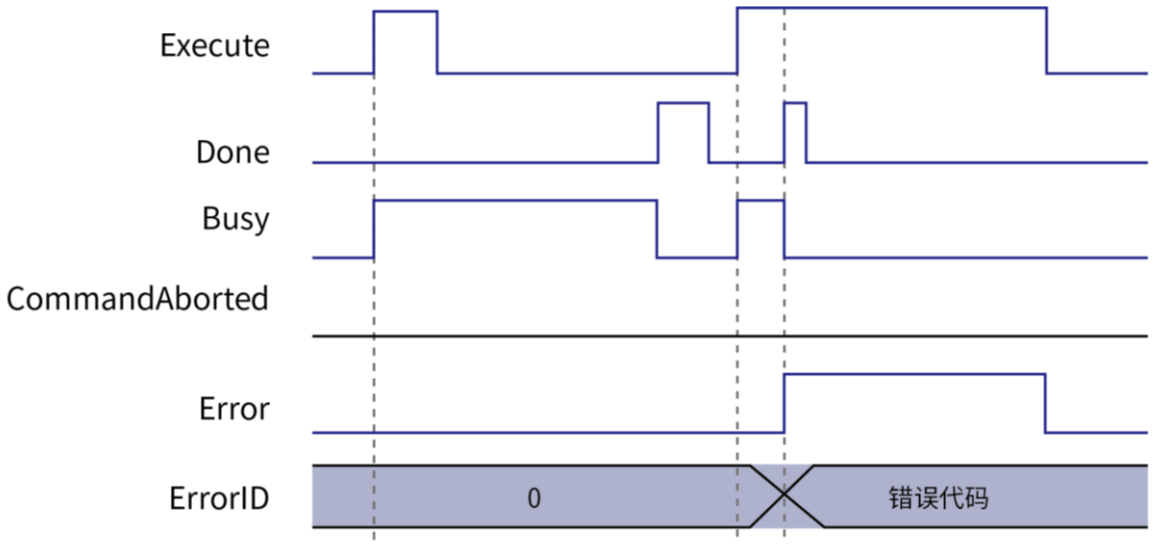


图 124

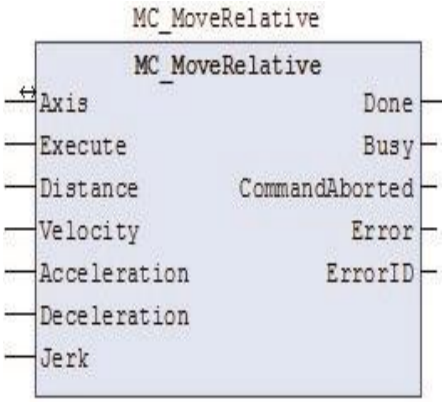
4) 注意事项

在运行过程中，一定要关注本指令运行的完整过程，从用户程序的设计角度避免其他指令中断此指令。

5.6 轴相对位置控制

指令名称：MC_MoveRelative

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ MoveRelative	轴相对定位指令		MC_MoveRelative(Axis:= , Execute:= , Distance:= , Velocity:= 10, Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Distance	运动相对位置	LREAL	数据范围	0	此数据为运动的相对位置
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值

Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块运行状态为 Standstill 中，指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。

轴按相对位置运行 (单位按轴设置), 相对位置由 Distance 指定；

本指令运行前设置好相关的参数 , 加速度(Acceleration)、减速度 (Deceleration)、运行速度 (Velocity) 和加减速模式的跃度 (Jerk)；对加速度 (Acceleration) 或减速度 (Deceleration) 的赋值为 0

启动指令为 Execute 的上升沿启动 , 本指令在 Discrete Motion 可以重复上升沿有效 , 每次都可以刷新最新的 Position 位置。Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

梯形加减速动作：Velocity、Acceleration 和 Deceleration 有数据，而 Jerk 为 0，速度/位置曲线如下：

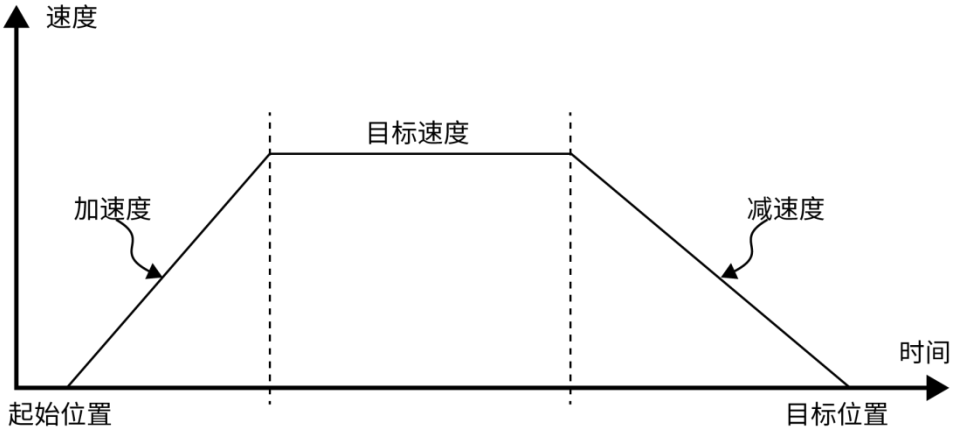


图 125

S 曲线加减速动作：Velocity、Acceleration、 Deceleration 和 Jerk 有数据，速度/位置曲线如下：

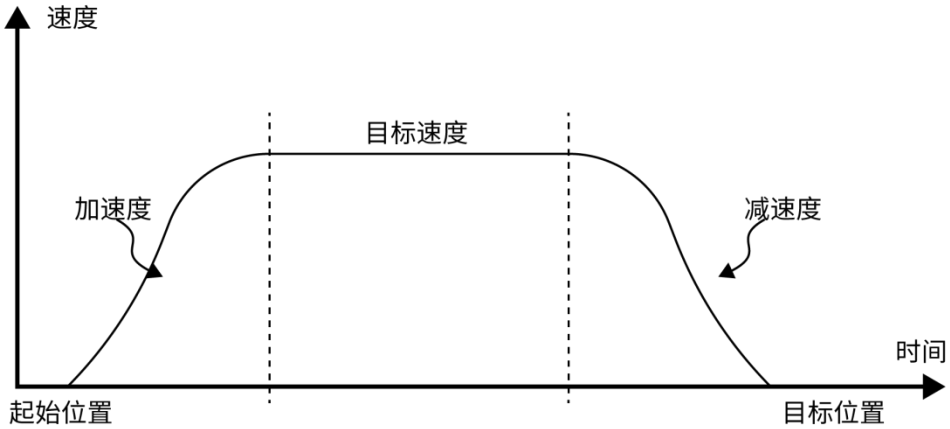


图 126

◆ 时序图

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

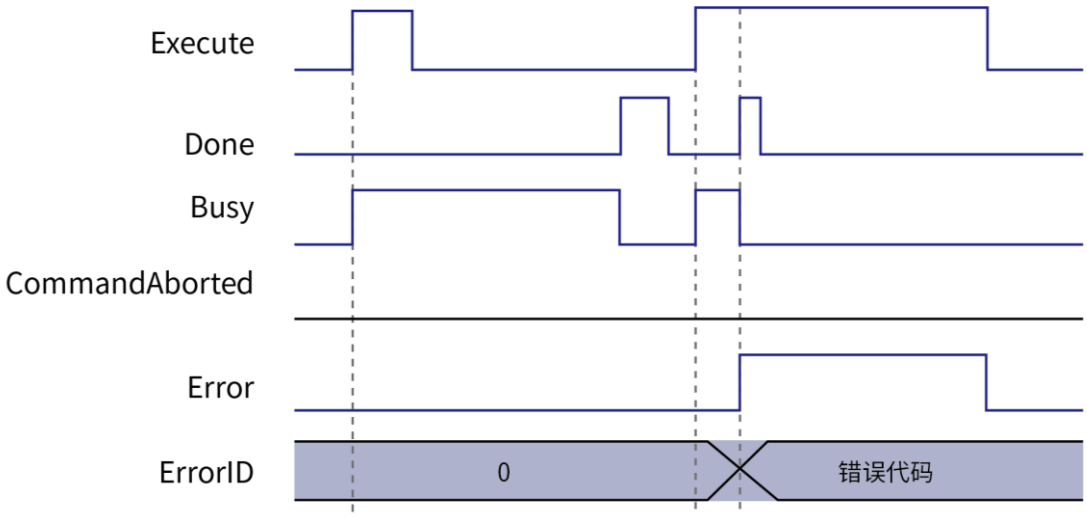


图 127

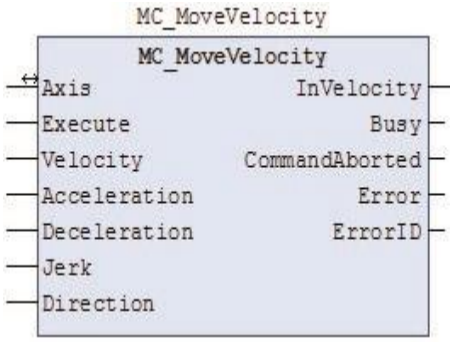
4) 注意事项

指令运行错误；在运行过程中，一定要关注本指令运行的完整过程，从用户程序的设计角度避免其他指令中断此指令。

5.7 速度控制

指令名称：MC_MoveVelocity

1) 指令格式

指令	名称	图形表现	ST 表现
MC_MoveVelocity	速度控制指令		MC_MoveVelocity(Axis:= , Execute:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , InVelocity=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Velocity	速度设定值	LREAL	数据范围	0	此数据为本指令的速度运行值
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值

Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值
Direction	运行方向	MC_Direction	positive, negative, current	current	为运行方向的指令操作

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InVelocity	达到设定速度的标志	BOOL	TRUE,FALSE	FALSE	设定的运行速度达到，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本指令使用驱动器位置控制模式下的进行模拟速度控制，在轴使能并指令有效的情况下，对 Velocity 的赋值能控制驱动器的速度。

◆ 时序图

功能块的 Execute 必须有上升沿的条件；

功能块的 InVelocity 表示指令的运行速度达到设定值；

功能块的 Busy 表示当前功能块正在执行中；

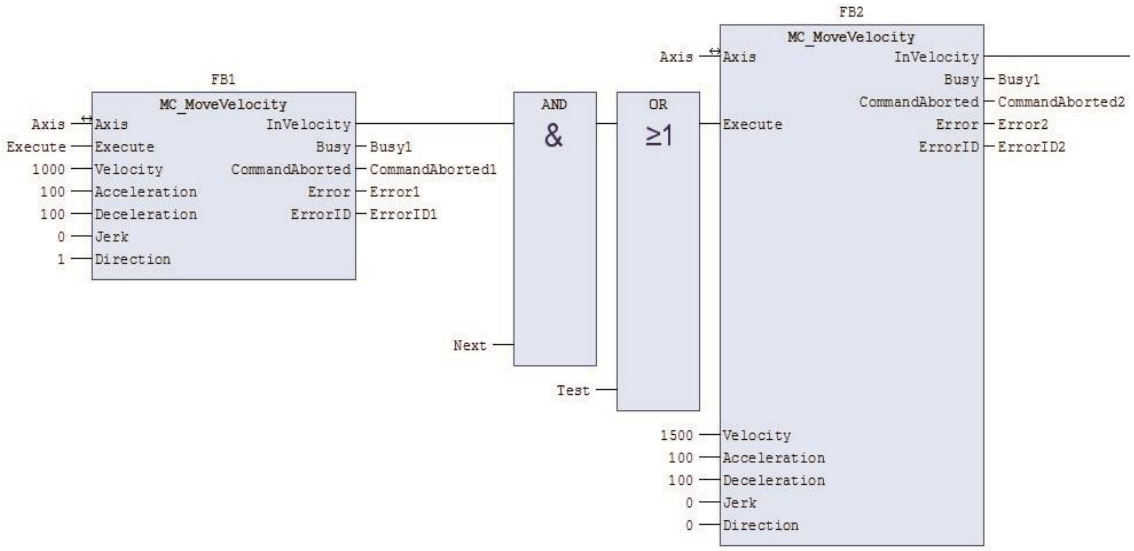

FB1
MC_MoveVelocity
Axis
Execute
1000
100
100
0
1
Axis
Execute
Velocity
Acceleration
Deceleration
Jerk
Direction
InVelocity
Busy
CommandAborted
Error
ErrorID
Busy1
CommandAborted1
Error1
ErrorID1
AND
&
Next
Test
OR
≥1
FB2
MC_MoveVelocity
Axis
Execute
1500
100
100
0
0
Axis
Velocity
Acceleration
Deceleration
Jerk
Direction
InVelocity
Busy
CommandAborted
Error
ErrorID
Busy1
CommandAborted2
Error2
ErrorID2

图 128

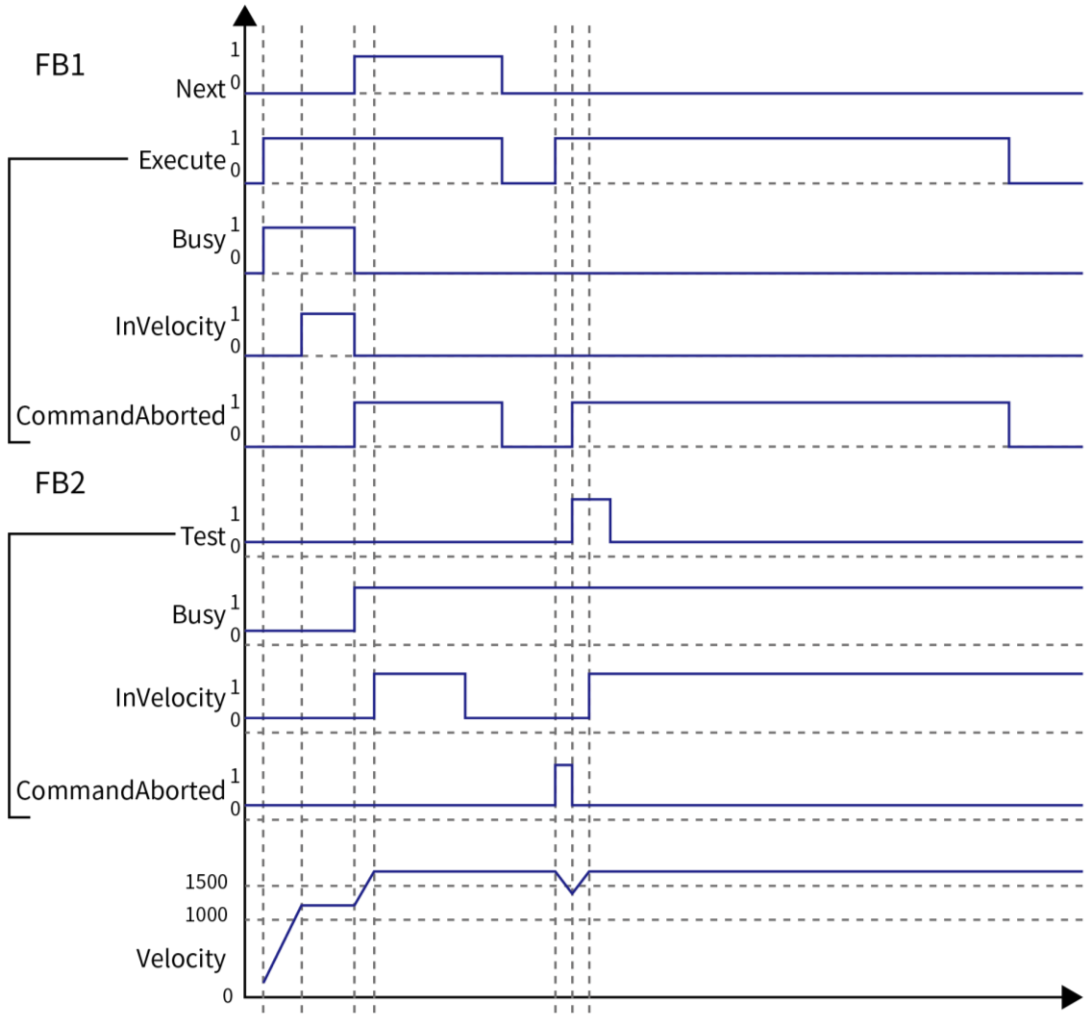
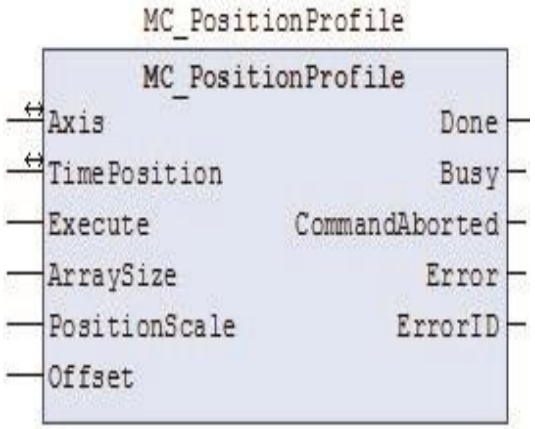

FB1
Next
Execute
Busy
InVelocity
CommandAborted
FB2
Test
Busy
InVelocity
CommandAborted
1500
1000
Velocity
0
1
0

图 129

5.8 位置轮廓

指令名称：MC_PositionProfile

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ PositionProfile	位置轮廓指令		MC_PositionProfile(Axis:= , TimePosition:= , Execute:= , ArraySize:= , PositionScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例
TimePosition	轴位置运行时间和位置描述	MC_TP_ REF			轴位置运行时间和位置数据描述 , 数据由多组数据组成

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
ArraySize	动态数组	INT	数据范围	0	运行轮廓中使用的数组个数

PositionScale	综合因子	LREAL	“正数”+”0”	1	MC_TP_REF 中位置的比例因子
Offset	偏移	LREAL		0	位置的总体偏移值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为时间段和位置的轮廓运动模型，运行模式为 Discrete Motion, 按用户在 TimePosition 变量中设定的数据运行。本功能块运行状态为 Standstill 中，指令运行时的状态为 Discrete Motion, 其他状态无法运行。

启动指令为 Execute 的上升沿启动，本指令在 Discrete Motion 重复运行。

TimePosition 为 MC_TP_REF 数据类型；

MC_TP_REF 具体描述如下：

成员	类型	初始值	描述
Number_of_pairs	INT	0	轮廓路径的段数
IsAbsolute	BOOL	TRUE	绝对运动 (TRUE) 和相对运动选择
MC_TP_Array	ARRAY[1..N] OF SMC_TP		时间和位置的数组

SMC_TP 具体描述如下：

成员	类型	初始值	描述
delta_time	TIME	TIME#0ms	位置段的时间
position	LREAL	0	当前的位置值

注：按设置的位置数据对应的速度有变化时都按 S 曲线进行相关调整。

◆ 时序图

条件 MC_TP_Array 通过其他方式已经设置才能运行位置轮廓曲线指令；

轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

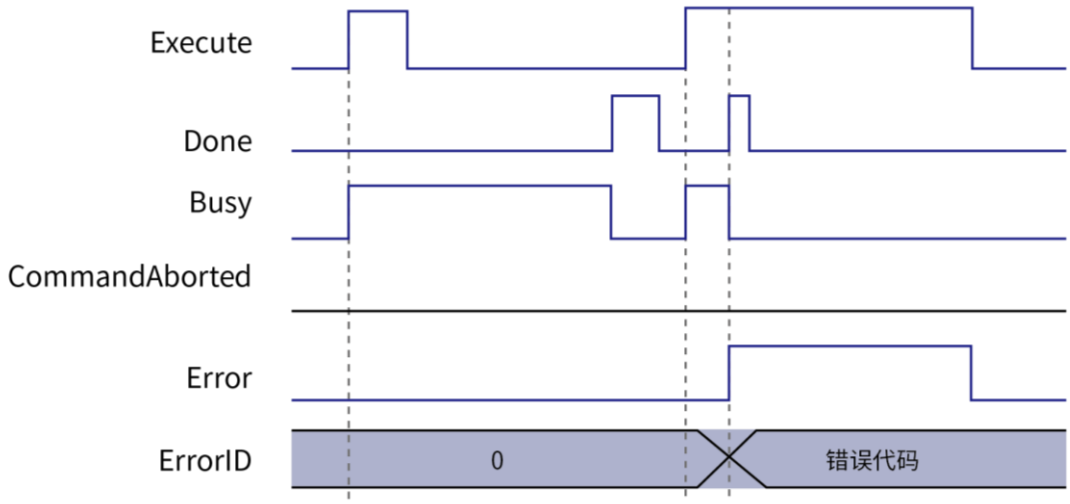


图 130

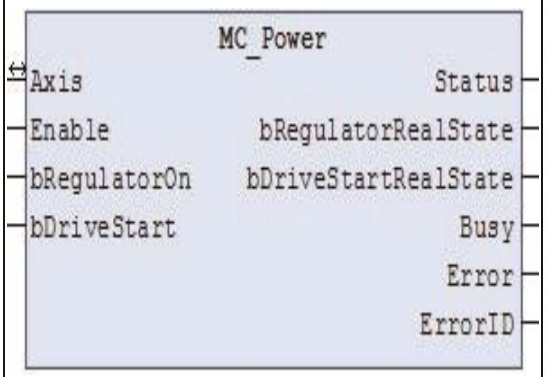
5) 注意事项

轴状态不是在 Standstill 中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

5.9 轴使能

指令名称：MC_Power

1) 指令格式

指令	名称	图形表现	ST 表现
MC_Power	轴使能指令		MC_Power(Axis:= , Enable:= , bRegulatorOn:= , bDriveStart:= , Status=> , bRegulatorRealState=> , bDriveStartRealState=> , Busy=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE,FALSE	FALSE	设置为 TRUE 则功能块开始处理
bRegulatorOn	使能状态	BOOL	TRUE,FALSE	FALSE	设置为 TRUE 则设置轴为使能状态
bDriveStart	允许驱动	BOOL	TRUE,FALSE	FALSE	设置为 TRUE 以关闭功能块的紧急停止处理

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	可运行状态	BOOL	TRUE,FALSE	FALSE	如果轴已经准备好运动，置为 TRUE
bRegulatorRealState	轴使能信号状态	BOOL	TRUE,FALSE	FALSE	当轴使能处于有效状态时，置为 TRUE
bDriveStartRealState	允许驱动状态	BOOL	TRUE,FALSE	FALSE	如果轴没有被快速停止机制中断，置为 TRUE
Busy	执行中	BOOL	TRUE,FALSE	FALSE	如果功能块的处理没有完成，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

只有在输入 Enable 为 TRUE 的时候，其它的输入才会被功能块处理。

如果功能块 MC_Power 已经被调用，并且 bRegulatorOn=FALSE，那么功能块将设置相关轴的轴状态（nAxisState）为 power_off 状态，表明驱动器还没有做好运动准备。

如果功能块 MC_Power 已经被调用，并且 bRegulatorOn=TRUE，如果此时轴没有错误发生，那么功能块将设置相关轴的轴状态（nAxisState）为 standstill 状态；如果有错误发生，将输出相应的错误状态。

如果 Enable，bRegulatorOn 以及 bDriveStart 都为 TRUE，但是输出 Status 在一定时间后仍为 FALSE，那么输出 Error 将会被置位。当在使能状态情况下产生一个硬件问题，可能会发生这种情况。

如果使能信号丢失（通常在操作模式下），相关轴的 nAxisState 将会被置位 ErrorStop 状态。

◆ 时序图

将 Enable 设为 TRUE，bRegulatorOn 设为 TRUE，bDriveStart 设为 TRUE, 表示正在受理指令的 Busy 变为 TRUE，然后轴进入使能 ON 状态，Status 状态变为 TRUE。

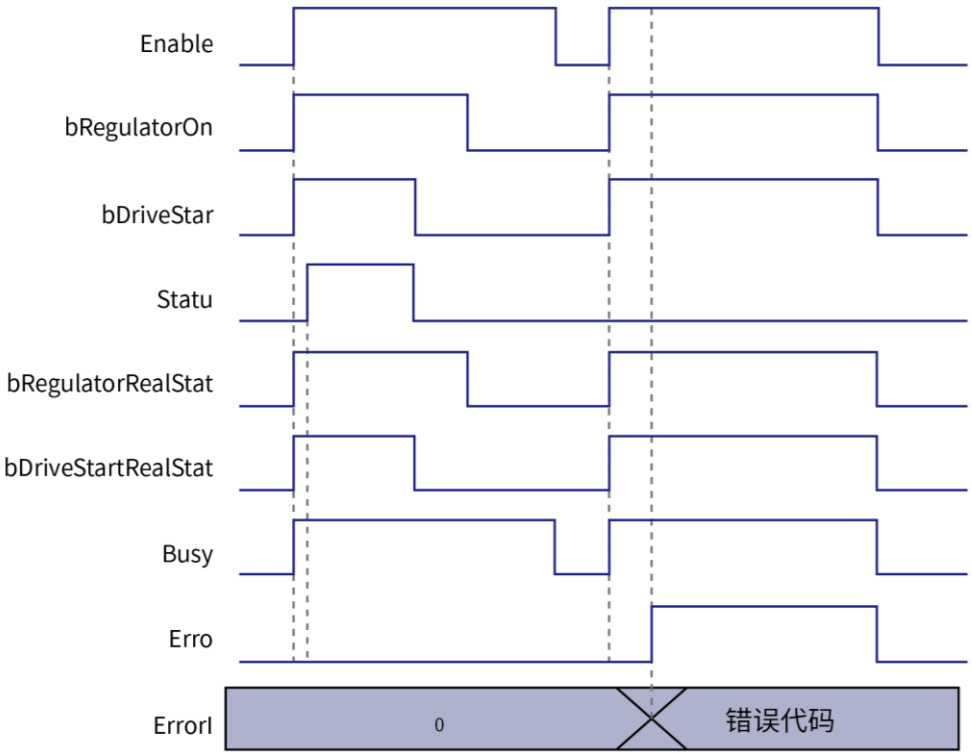


图 131

4) 注意事项

请勿在正在执行 MC_Power 指令的轴中，编写用于启动其它实例的 MC_Power 指令的程序。原则上 1 根轴只能设 1 个 MC_Power 指令。

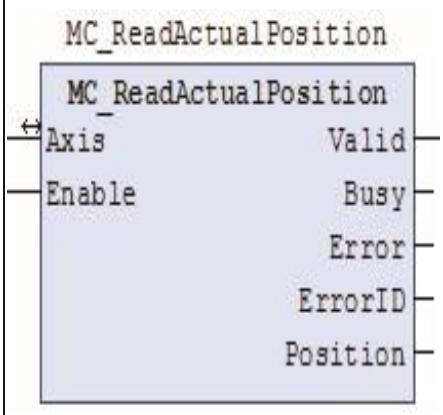
如果在正在执行 MC_Power 指令的轴中，启动其它实例的 MC_Power 指令，则优先执行后执行的 MC_Power 指令。

5.10 轴实际位置读取

指令名称：MC_ReadActualPosition

1) 指令格式

2) 相关变量

指令	名称	图形表现	ST 表现
MC_ ReadActualPosition	实际位置读取指令		MC_ReadActualPosition(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	位置数据可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的位置，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码
Position	获取到的轴位置	LREAL	轴位置	0	指令读出来的轴位置数据

3) 功能说明

通过本指令读取驱动器中的实际位置指令，指令为 Enable 电平使能效应。指令读取驱动器运行的实际位置，保存在自己定义的变量单元中。

指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Position 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中；

时序操作说明 ：

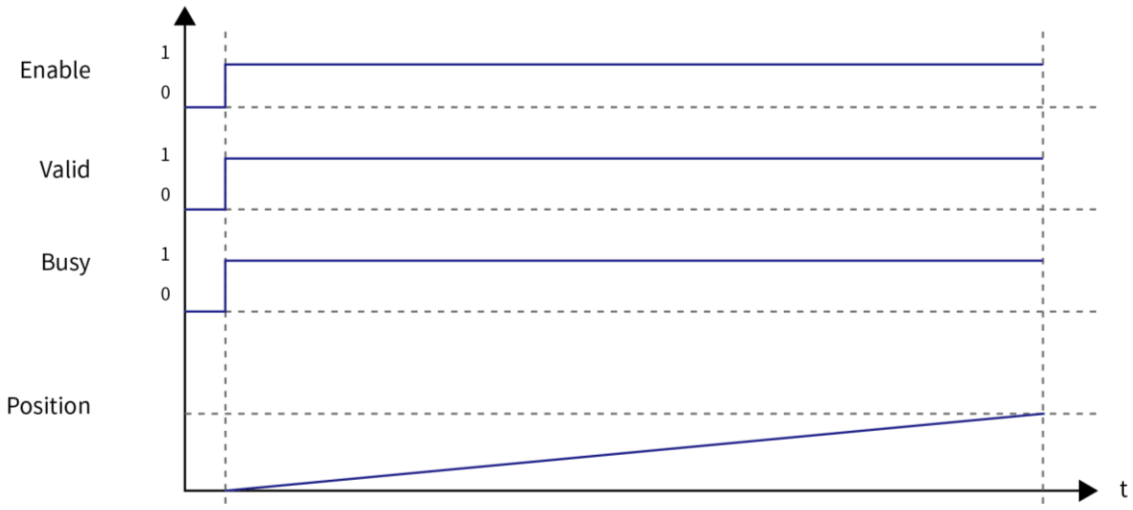
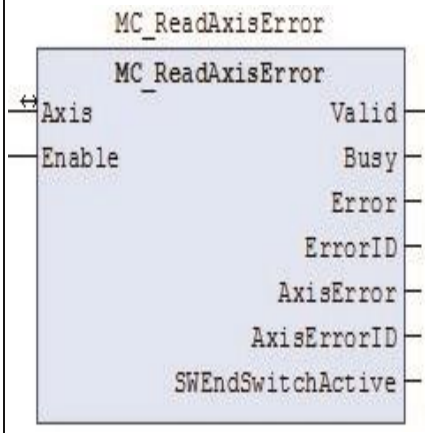


图 132

5.11 轴错误读取

指令名称：MC_ReadAxisError

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ReadAxisError	读轴的错误状态		MC_ReadAxisError(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , AxisError=> , AxisErrorID=> , SWEndSwitchActive=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	错误数据可获取标志	BOOL	TRUE,FALSE	FALSE	能获取轴的错误数据，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执

					行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
AxisError	轴错误标示	BOOL	TRUE,FALSE	FALSE	读出轴为错误，对应的标示置位
AxisErrorID	轴错误码	DWORD		0	读出轴为错误码
SWEndSwitchActive	软限位开关有效	BOOL	TRUE,FALSE	FALSE	在指令读取中，检查软限位开关状态

3) 功能说明

通过 MC_ReadAxisError 读取驱动器中的错误码，指令为 Enable 电平使能效应。指令读取轴的错误情况，保存在自己定义的变量单元中。

指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

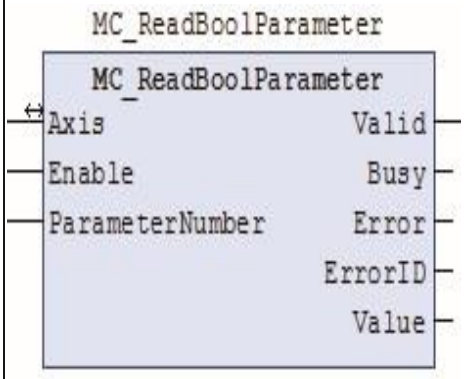
功能块的 Valid 表示读出的 AxisError 和 AxisErrorID 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中；

5.12 轴位参数读取

指令名称：MC_ReadBoolParamete

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ ReadBoolParameter	读取轴的位参数		MC_ReadBoolParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置
ParameterNumber	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	位置数据可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的位置，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
Value	获取到的轴位状态	BOOL	TRUE,FALSE	FALSE	指令读出来的轴位状态

3) 功能说明

通过 MC_ReadBoolParam 读取驱动器中的位数据状态，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；功能块的 Valid 表示读出的 Valid 为有效的位状态数据；功能块的 Busy 表示当前功能块正在执行中； 时序操作说明：

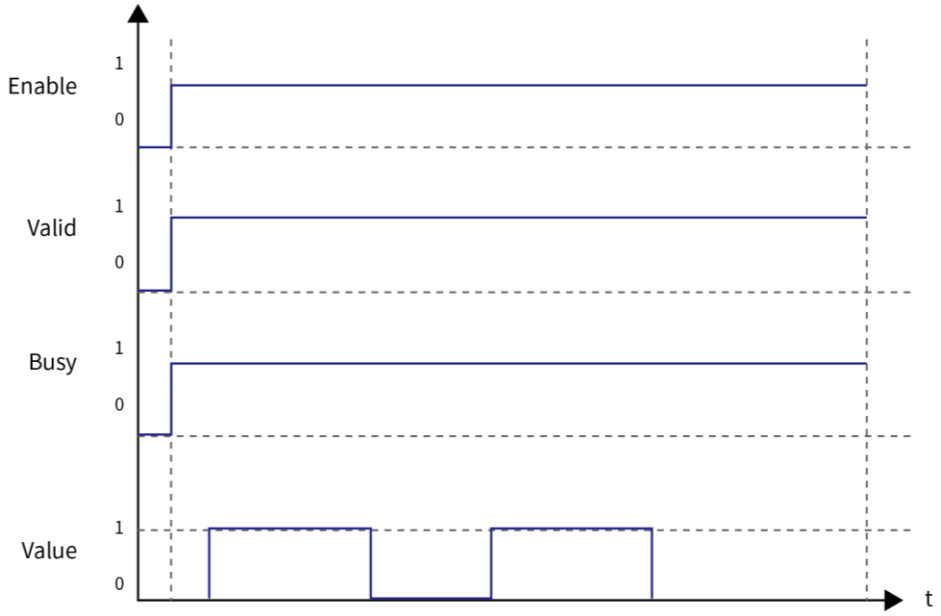
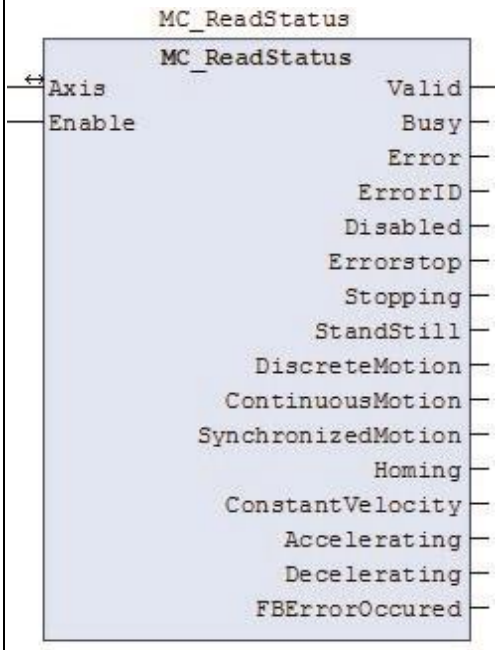


图 133

5.13 轴状态读取

指令名称：MC_ReadStatus

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ReadStatus	读轴的状态		MC_ReadStatus(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Disabled=> , Errorstop=> , Stopping=> , StandStill=> , DiscreteMotion=> , ContinuousMotion=> , SynchronizedMotion=> , Homing=> , ConstantVelocity=> , Accelerating=> , Decelerating=> , FBErrorOccured=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	错误数据可获取标志	BOOL	TRUE,FALSE	FALSE	能获取轴的错误数据，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
Disabled	轴没使能状态	BOOL	TRUE,FALSE	FALSE	轴在没使能状态为 TRUE；
Errorstop	轴错误状态	BOOL	TRUE,FALSE	FALSE	轴在错误运行状态为 TRUE；
Stoping	轴停止过程状态	BOOL	TRUE,FALSE	FALSE	轴在停止过程中为 TRUE
StandStill	轴标准状态	BOOL	TRUE,FALSE	FALSE	轴在标准 (能运行) 状态中为 TRUE
DiscreteMotion	轴离散运动状态	BOOL	TRUE,FALSE	FALSE	轴在离散运动状态中为 TRUE
ContinuousMotion	轴连续运动状态	BOOL	TRUE,FALSE	FALSE	轴在连续运动状态中为 TRUE
SynchronizedMotion	轴同步运行状态	BOOL	TRUE,FALSE	FALSE	轴在同步运动状态中为 TRUE

Homing	轴回原点状态	BOOL	TRUE,FALSE	FALSE	轴在回原点状态中为 TRUE
ConstantVelocity	轴运行速度到达	BOOL	TRUE,FALSE	FALSE	轴达到运行速度中为 TRUE
Accelerating	轴加速过程状态	BOOL	TRUE,FALSE	FALSE	轴加速过程状态为 TRUE
Dccelerating	轴减速过程状态	BOOL	TRUE,FALSE	FALSE	轴减速过程状态为 TRUE
FBErrorOccured	轴功能块错误出现标志	BOOL	TRUE,FALSE	FALSE	轴功能块错误标志为 TRUE

3) 功能说明

通过 MC_ReadStatus 对应轴的各种状态，指令为 Enable 电平使能效应。指令读取轴的状态数据，保存在自己定义的变量单元中。指令可以重复多次使用，互不影响。

◆ 时序说明

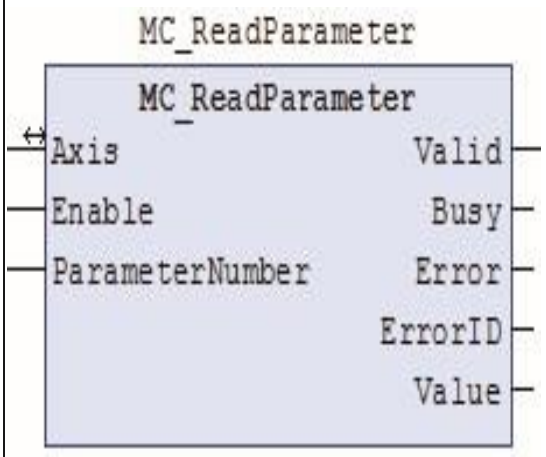
功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的后面个状态标志的各种数据；

功能块的 Busy 表示当前功能块正在执行中。

5.14 轴参数读取

指令名称：MC_ReadParameter

指令	名称	图形表现	ST 表现
MC_ ReadParameter	读取轴的参数		MC_ReadParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

1) 指令格式

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置
ParameterNumber	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	位置数据可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的位置，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码
Value	获取到的轴参数	LREAL		0	指令读出来的轴参数

3) 功能说明

通过 MC_ReadParam 读取驱动器中的位数据状态。指令读取驱动轴的参数，保存在自己定义的变量单元中。

指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Valid 为有效的位状态数据；

功能块的 Busy 表示当前功能块正在执行中；

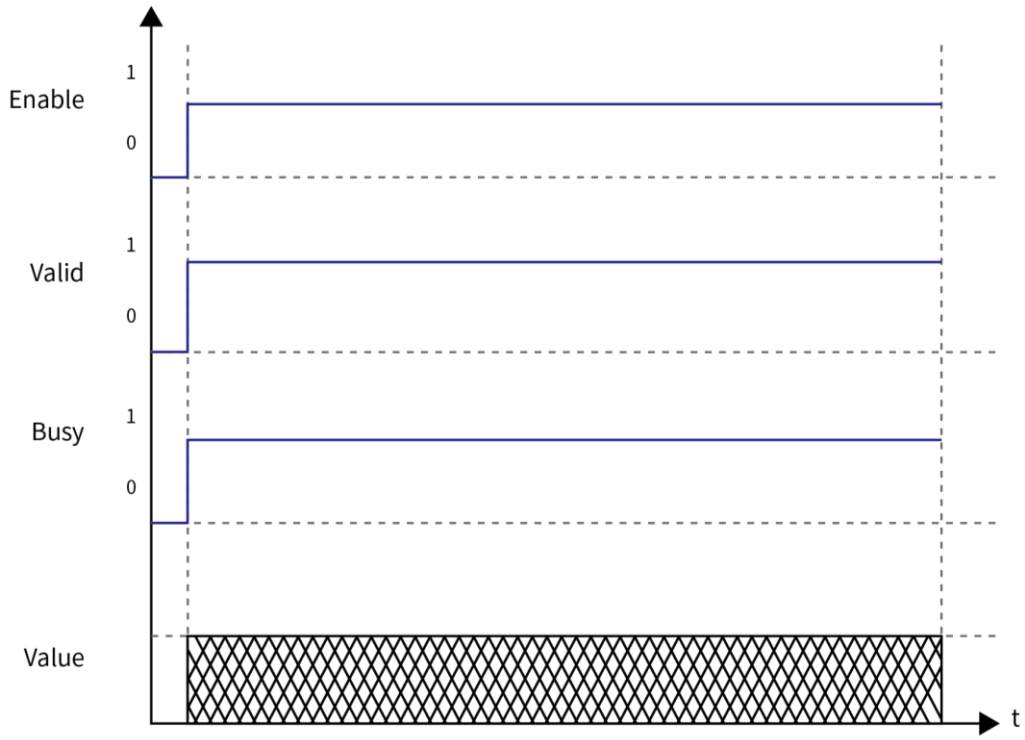
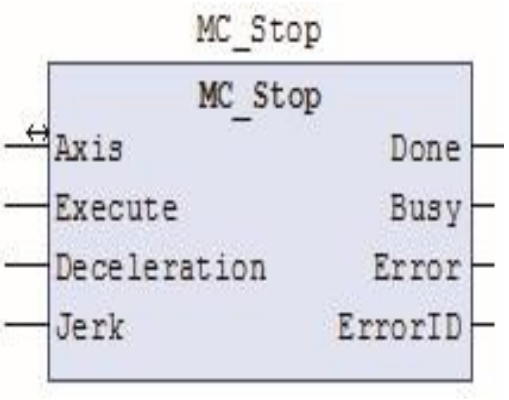


图 134

5.15 轴停止

指令名称：MC_Stop

1) 指令格式

指令	名称	图形表现	ST 表现
MC_Stop	轴停止命令		MC_Stop(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Deceleration	减速度	LREAL	“正数”+”0”	0	功能块的减速度 (u/S^2)
Jerk	加加速度	LREAL	“正数”+”0”	0	指定加加速度［指令单位/S^3］

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为在正常运行的情况下停止一个轴的运动，当轴处于 stopping 状态对本轴的任何指令都是无效的。

当轴状态处于 stopping 时 ,Execute 为 Flase 状态 ,Done 输出状态为 True, 轴状态变为 Standstill; 本功能块运行状态为运行状态 (Motion) 才能运行 , 其他状态无法运行。

启动指令为 Execute 的上升沿启动。

在 MC_Stop 有效执行的过程 Busy 有效时 , 再一次启动 MC_Stop 指令系统处于 Errorstop 状态

◆ 时序图

轴必须处于运行状态 (Motion) 指令才能运行；

功能块的 Execute 必须有上升沿的条件；功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

功能块的 CommandAborted 表示指令被其他运动控制指令中断，此时标志位为 TRUE；

例程 : 在执行 MC_MoveVelocity 指令和 MC_Stop 指令在不同的时序操作中对应的标志位的变化 ; 对 CommandAborted 的处理描述如下图的时序描述 .

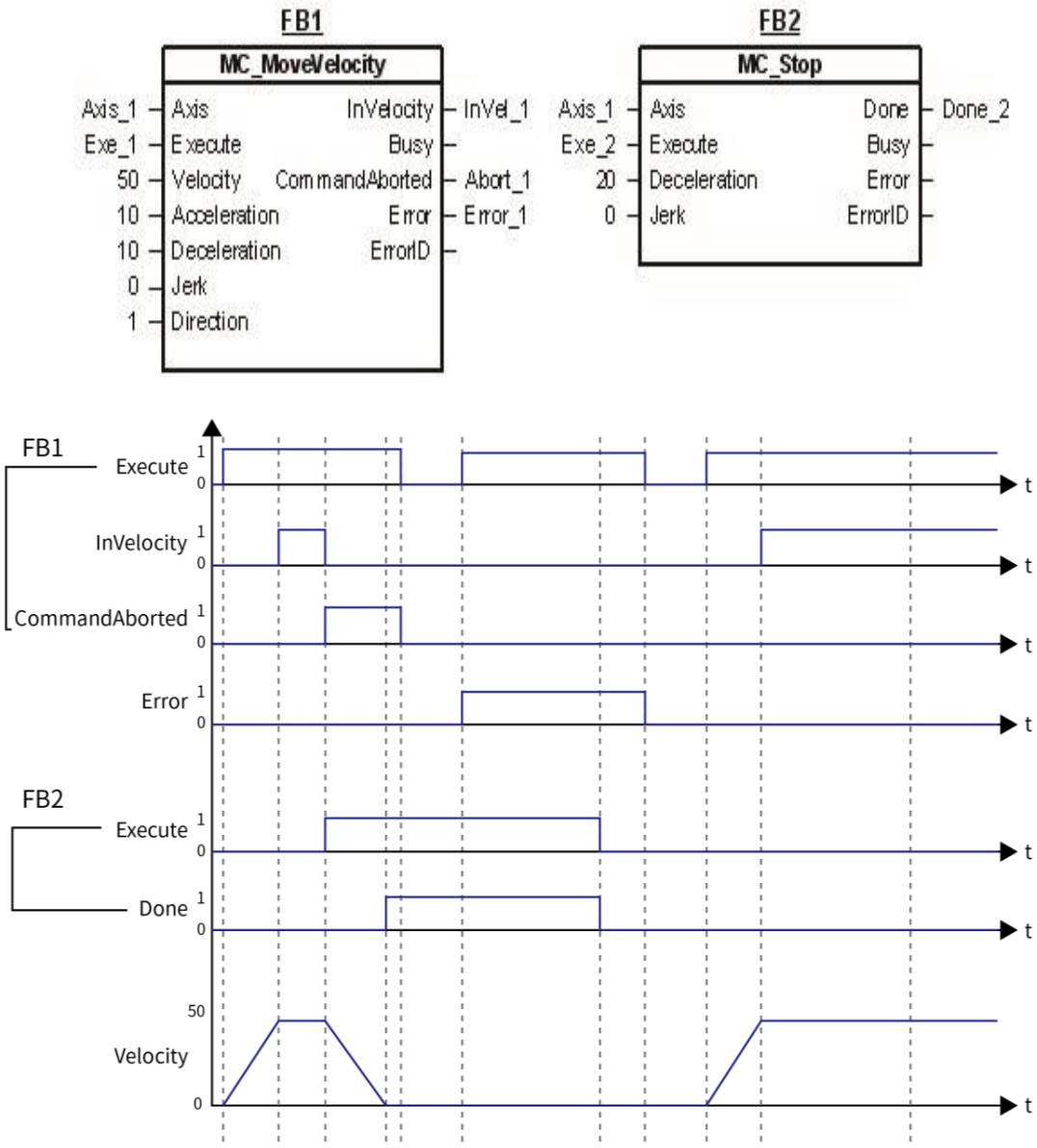


图 135

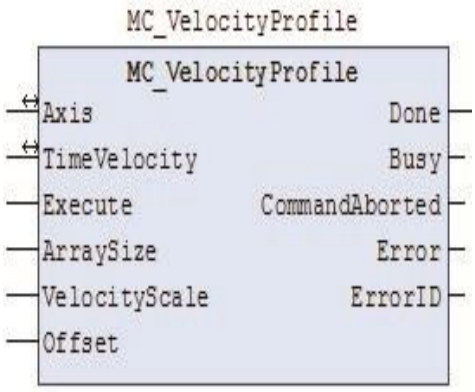
4) 注意事项

MC_Stop 有重复性的指令运行时，错误标志 Error 为 True,ErrorID 为 SMC_MS_AXI 错误；

5.16 速度轮廓

指令名称：MC_VelocityProfile

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ VelocityProfile	速度轮廓指令		MC_VelocityProfile(Axis:= , TimeVelocity:= , Execute:= , ArraySize:= , VelocityScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例
TimeVelocity	轴速度运行时间和速度描述	MC_TV_ REF			轴速度运行时间和速度数据描述 , 由多组数据组成

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
ArraySize	动态数组	INT	数据范围	0	运行轮廓中使用的数组个数
VelocityScale	速度因子	LREAL	“正数”，”0”	1	速度的比例因子
Offset	偏移	LREAL		0	速度值的总体偏移值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为时间段和速度的轮廓运动模型 , 运行模式为 Continuous Motion，按用户在 TimeVelocity 变量中设定的数据运行。

本功能块运行状态为 Standstill 中 , 指令运行时的状态为 Discrete Motion, 其他状态无法运行。

启动指令为 Execute 的上升沿启动 , 本指令在 Discrete Motion 重复运行。

TimeVelocity 为 MC_TV_REF 数据类型 ;

MC_TV_REF 具体描述如下 :

成员	类型	初始值	描述
Number_of_pairs	INT	0	轮廓路径的段数
IsAbsolute	BOOL	TRUE	绝对运动 (TRUE) 和相对运动选择
MC_TV_Array	ARRAY[1..N] OF SMC_TV		时间和速度的数组

SMC_TV 具体描述如下：

成员	类型	初始值	描述
delta_time	TIME	TIME#0ms	速度值段的时间
Velocity	LREAL	0	当前记录的速度值

注：整个速度的过程为 S 曲线加减速的方式，每段轮廓都速度都为叠加的计算方式；在指令重复运行时，速度也为叠加方式，在指令使用时避免速度超限的出现；重复运行一定把本轴的状态重回到 Standstill 状态。

◆ 时序图

条件 MC_TV_Array 通过其他方式已经设置才能运行位置轮廓曲线指令；

轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

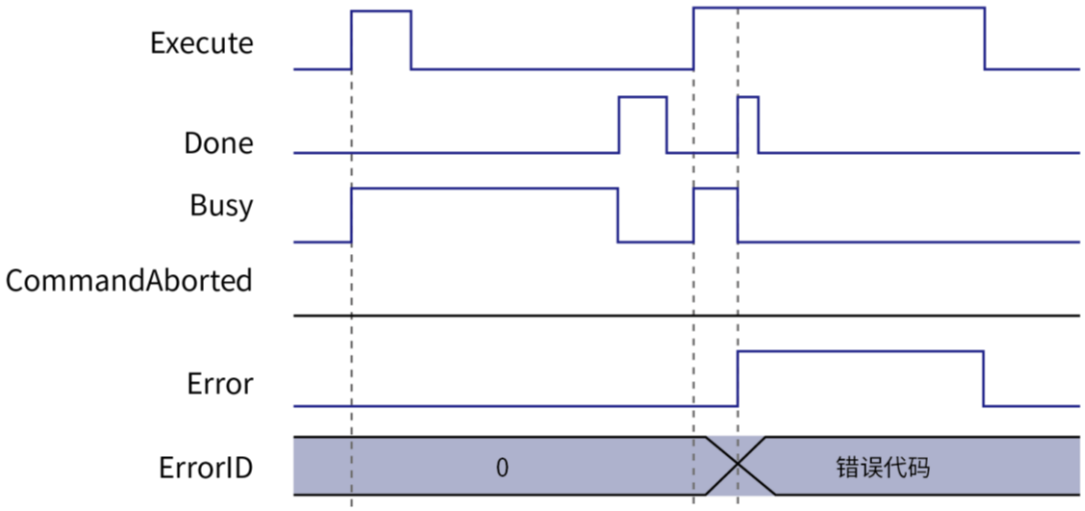


图 136

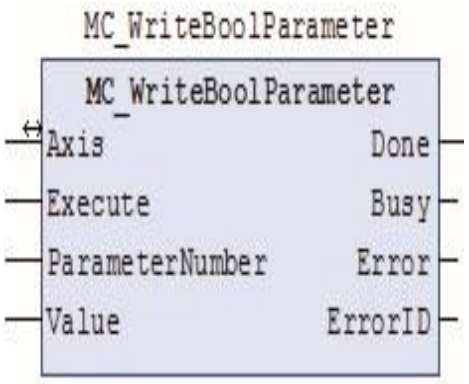
4) 注意事项

错误的出现为轴状态不是在 Standstill 中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

5.17 设置轴的位参数

指令名称 MC_WriteBoolParameter

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ WriteBoolParameter	设置轴的位参数		MC_WriteBoolParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作
ParameterNumber	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号
Value	设定值	BOOL	TRUE,FALSE	FALSE	设置位参数值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码

3) 功能说明

通过 MC_ WriteBoolParameter 设置轴的位参数，指令为 Execute 上升沿触发。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Execute 必须为上升沿触发条件；

功能块的 Done 表示设置操作成功；

功能块的 Busy 表示当前功能块正在执行中；

◆ 时序操作说明：

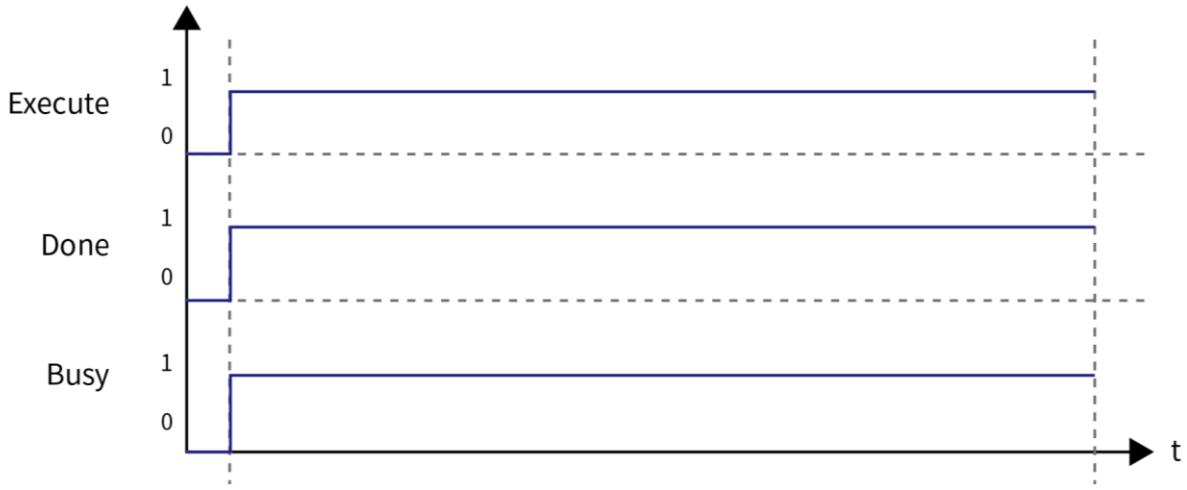
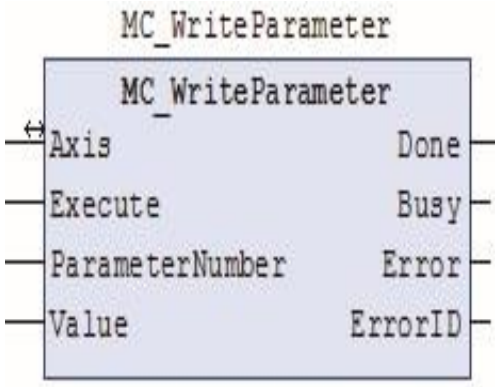


图 137

5.18 设置轴参数

指令名称;MC_WriteParameter

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ WriteParameter	设置轴参数		MC_WriteParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作
ParameterNumber	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号
Value	设定值	LREAL			设置位参数值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码

3) 功能说明

通过 MC_ WriteParameter 设置轴的位参数，指令修改驱动轴的参数，保存在自己定义的变量单元中。

指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Execute 必须为上升沿触发条件；

功能块的 Done 表示设置操作成功；

功能块的 Busy 表示当前功能块正在执行中；

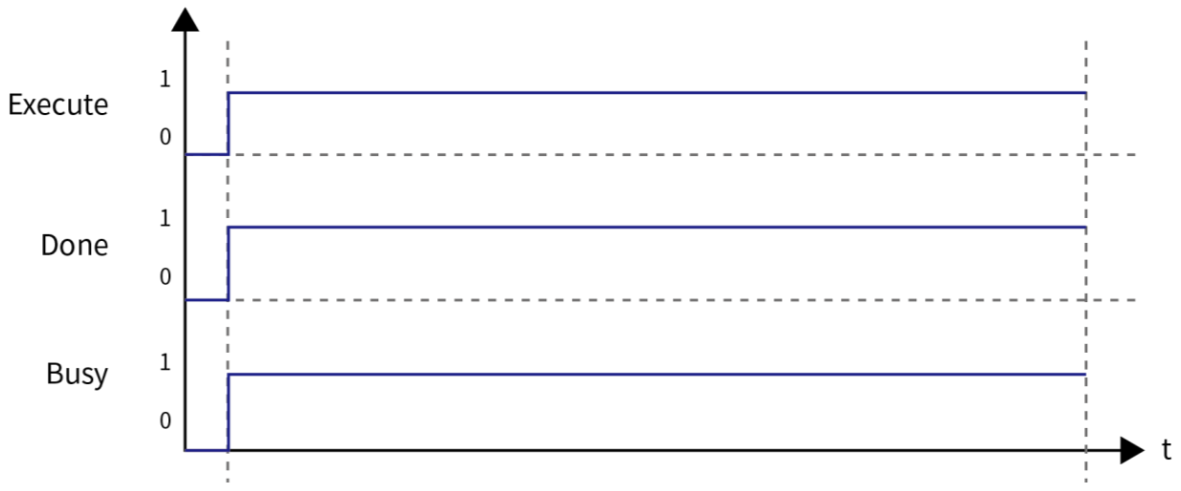
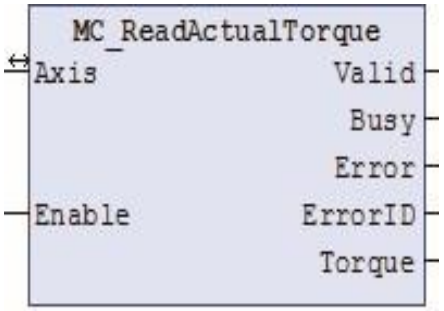


图 138

5.19 读取当前扭矩值

指令名称：MC_ReadActualTorque

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ReadActualTorque	当前力矩值读取指令		MC_ReadActualTorque0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Torque=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	当前力矩值可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的力矩值，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为

					TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
Torque	获取的当前力矩值	LREAL	力矩值	0	指令读出来的当前力矩数据

3) 功能说明

通过 MC_ReadActualTorque 读取驱动器中的当前力矩值指令，指令读取驱动器运行的当前力矩值，读取的当前力矩值保存在自己定义的变量单元中。指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Torque 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中；

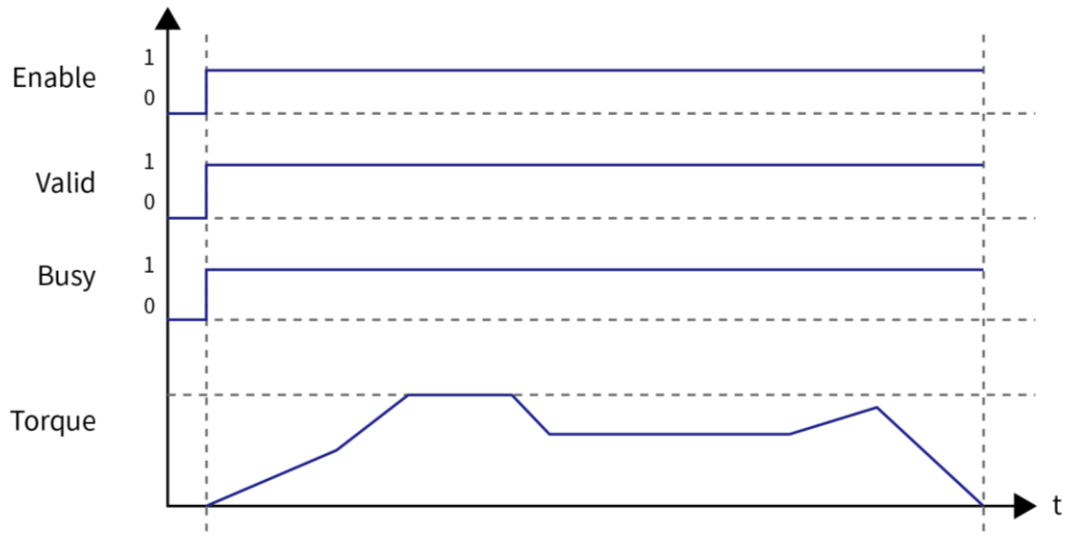
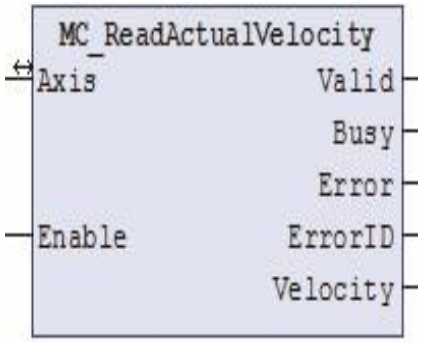


图 139

5.20 读取当前速度

指令名称：MC_ReadActualVelocity

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ ReadActualVelocity	当前速度读取指令		MC_ReadActualVelocity0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Velocity=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取当前轴速度

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	当前速度值可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的速度值，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
Velocity	获取的当前速度值	LREAL	速度值	0	指令读出来的当前速度数据

3) 功能说明

通过 MC_ReadActualVelocity 读取驱动器中的当前速度值指令，指令读取驱动器运行的当前速度值，读取的当前速度值保存在自己定义的变量单元中。指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Velocity 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中；

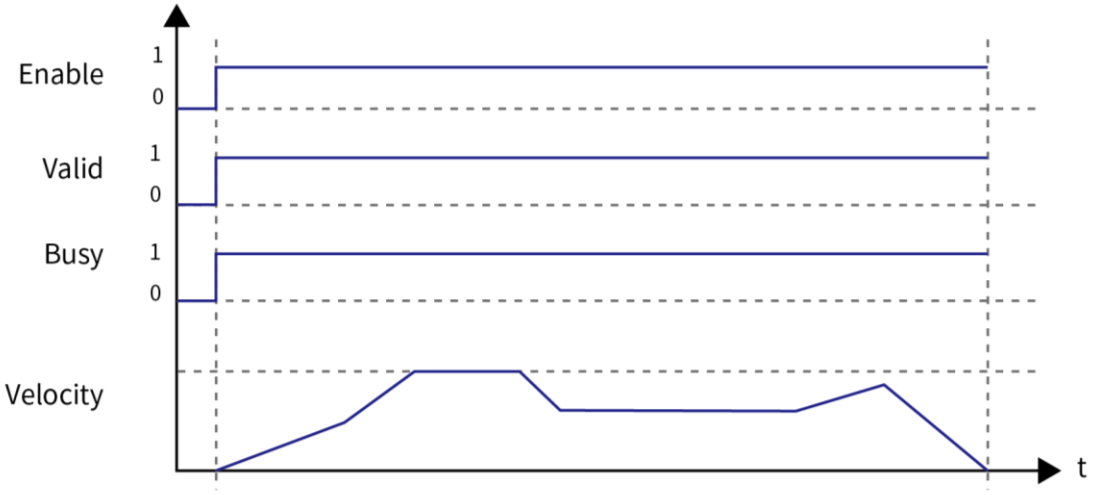
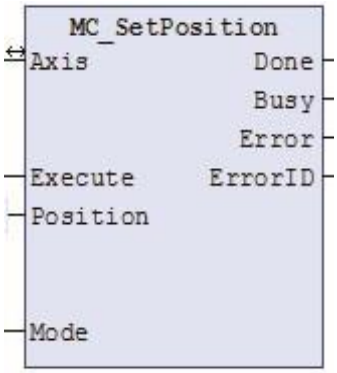


图 140

5.21 轴位置设置

指令名称：MC_SetPosition

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ SetPosition	轴位置设置		MC_SetPosition0(Axis:= , Execute:= , Position:= , Mode:= , Done=> , Busy=> , Error=> , ErrorID=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作
Position	轴位置数据	LREAL		0	位置数据
Mode	设定值	BOOL	TRUE,FALSE	FALSE	位置模式；TRUE：相对位置 (RELATIVE)；FALSE：绝对位置 (ABSOLUTE)；

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

通过 MC_ SetPosition 设置轴的位置参数，不产生任何位移，但形成了坐标偏移；将指令中的位置数据设为当前轴的位置数据，对设置位置数据操作不会产生任何位移移动，用于产生坐标系的位移。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Execute 必须为上升沿触发条件；

功能块的 Done 表示设置操作成功；

功能块的 Busy 表示当前功能块正在执行中；

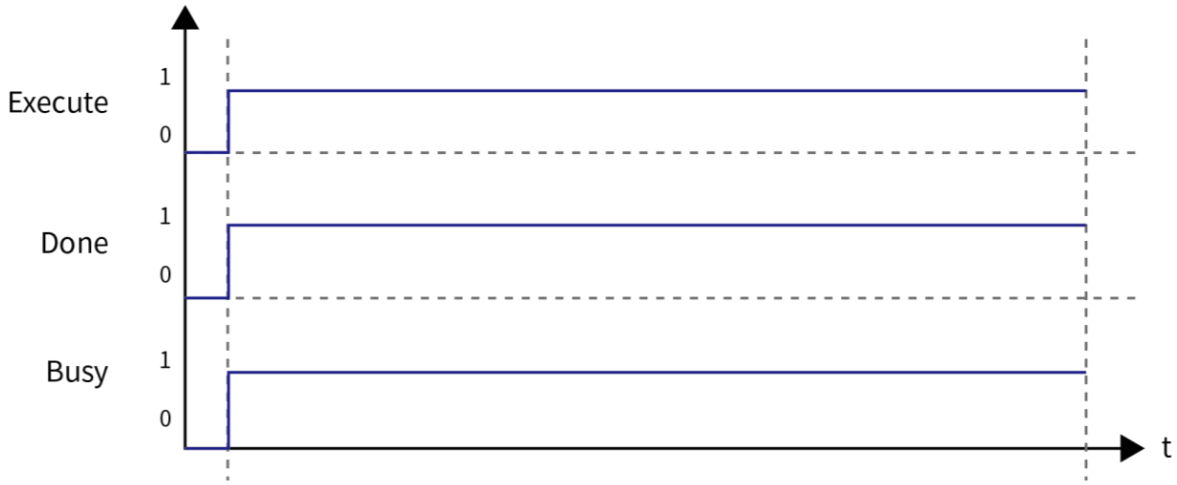
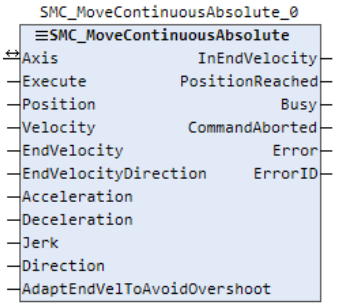


图 141

5.22 轴绝对位置连续控制

指令名称：MC_MoveContinuousAbsolute

1) 指令格式

指令	名称	图形表现	ST 表现
MC_MoveContinousAbsolute	轴绝对位置连续控制指令		

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Distance	运动相对位置	LREAL	数据范围	0	此数据为运动的相对位置
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
EndVelocity	运行结束速度	LREAL	数据范围	0	指令执行完成后的运行速度
EndVelocityDrection	结束速度	MC_Direction	positive,	Current	可以使用：positive,

	的方向		negative, current;		negative, current;不可使用 : shortest, fastest
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Direction	运行方向	shortest	数据范围	shortest	对于线性 / 直线轴：positive, negative; 对于旋转 / 圆周轴：positive, negative, current, shortest, fastest

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InEndVelocity	指令位置到达	BOOL	TRUE,FALSE	FALSE	轴指令执行位置到达，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块为轴绝对定位指令 , Distance 数据为轴的绝对位置。

本功能块运行状态为 Standstill 中 , 指令运行时的状态为 Discrete Motion, 一个完整的运行过程一定要控制轴的不同运动状态。

启动指令为 Execute 的上升沿启动 , 本指令在 Discrete Motion 可以重复上升沿有

效 , 每次都可以刷新最新的 Position 位置。

Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

轴按绝对位置连续运行 (单位按轴设置), 绝对位置由 Distance 指定，最后的运行速度 EndVelocity 运行；本指令运行前设置好相关的参数 , 加速度 (Acceleration)、减速度 (Deceleration) 和运行速度 (Velocity)；对加速度 (Acceleration) 或减速度 (Deceleration) 的赋值为 0

◆ 时序图

轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

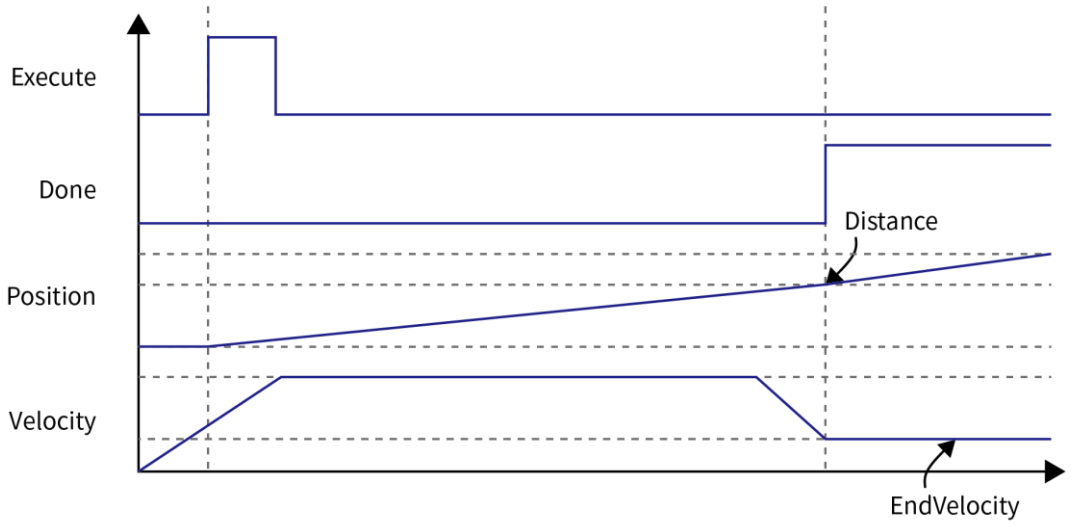


图 142

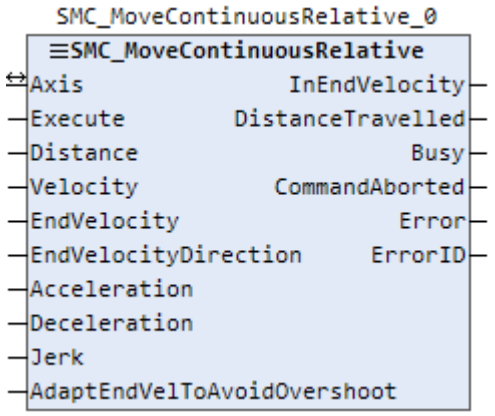
4) 注意事项

在运行过程中，一定要关注本指令运行的完整过程，从用户程序的设计角度避免其他指令中断此指令。

5.23 轴相对定位

指令名称：MC_MoveContinuousRelative

1) 指令格式

指令	名称	图形表现	ST 表现
MC_ MoveContinuousRelative	轴相对定位指令		

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Distance	运动相对位置	LREAL	数据范围	0	此数据为运动的相对位置
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
EndVelocity	运行结束速	LREAL	数据范围	0	指令执行完成后的

	度				运行速度
EndVelocityDirection	结束速度的方向	MC_Direction	positive, negative, current;	Current	可以使用：positive, negative, current; 不可使用 : shortest, fastest
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Direction	运行方向	shortest	数据范围	shortest	对于线性 / 直线轴：positive, negative; 对于旋转 / 圆周轴：positive, negative, current, shortest, fastest

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InEndVelocity	指令位置到达	BOOL	TRUE,FALSE	FALSE	轴指令执行位置到达，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE

ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码

3) 功能说明

本功能块运行状态为 Standstill 中， 指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。

启动指令为 Execute 的上升沿启动 , 本指令在 Discrete Motion 可以重复上升沿有效 , 每次都可以刷新最新的 Position 位置。

Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

轴按相对位置连续运行 (单位按轴设置), 绝对位置由 Distance 指定，最后的运行速度 EndVelocity 运行；本指令运行前设置好相关的参数 , 加速度 (Acceleration)、减速度 (Deceleration) 和运行速度 (Velocity)；对加速度 (Acceleration) 或减速度 (Deceleration) 的赋值为 0

◆ 时序图

功能块的 Execute 必须有上升沿的条件；功能块的 Done 表示指令正常执行完成；功能块的 Busy 表示当前功能块正在执行中；

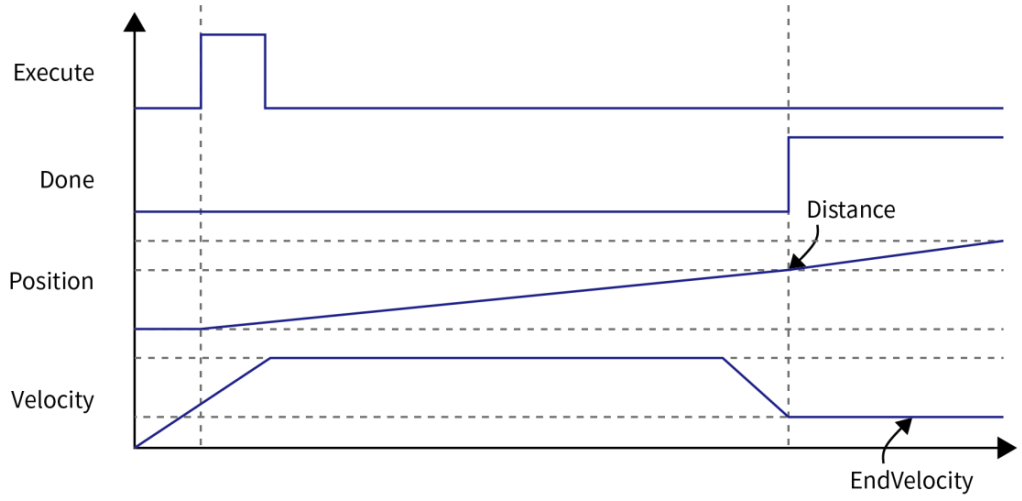


图 143

4) 注意事项

指令运行错误；在运行过程中，一定要关注本指令运行的完整过程，从用户程序的设计角度避免其他指令中断此指令。

5.24 轴点动

指令名称：MC_Jog

1) 指令格式

指令	名称	图形表现	ST 表现
MC_Jog	轴点动指令		MC_Jog(Axis:= , JogForward:= , JogBackward:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);

2) 相关变量

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
JogForward	正向有效	BOOL	TRUE,FALSE	FALSE	设置为 TRUE 则开始正向移动；设置为 FALSE 则停止正向移动
JogBackward	负向有效	BOOL	TRUE,FALSE	FALSE	设置为 TRUE 则开始反向移动；设置为 FALSE 则停止反向移动
Velocity	目标速度	LREAL	正数或“0”	0	指定目标速度。单位：[指令单位 /s]
Acceleration	加速度	LREAL	正数或“0”	0	指定加速度。单位：[指令单位 /s]

Deceleration	减速度	LREAL	正数或“0”	0	指定减速度。单位：[指令单位 /s]

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE,FALSE	FALSE	接收指令后，置为 TRUE
CommandAborted	执行中断	BOOL	TRUE,FALSE	FALSE	指令中止时，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

3) 功能说明

根据指定 Velocity（目标速度）执行点动运行。

需要正向运行时，将 JogForward（正向运行有效）置为 TRUE；需要反向运行时，将 JogBackward（负向运行有效）置为 TRUE。同时将 JogForward（正向运行有效）和 JogBackward（负向运行有效）置为 TRUE，将不会有运动发生。

如果 MC_Jog 指令的指令速度设置值超过轴参数中的点动最高速度，则以点动最高速度执行。

◆ 时序图

在启动 JogForward（正向运行有效）或 JogBackward（负向运行有效）的同时，Busy（执行中）变为 TRUE。

在 JogForward（正向运行有效）或 JogBackward（负向运行有效）的下降沿开始减速并停止轴的同时，Busy（执行中）变为 FALSE。

利用其它指令中止本指令时，CommandAborted（执行中断）变为 TRUE，Busy（执行中）变为 FALSE。

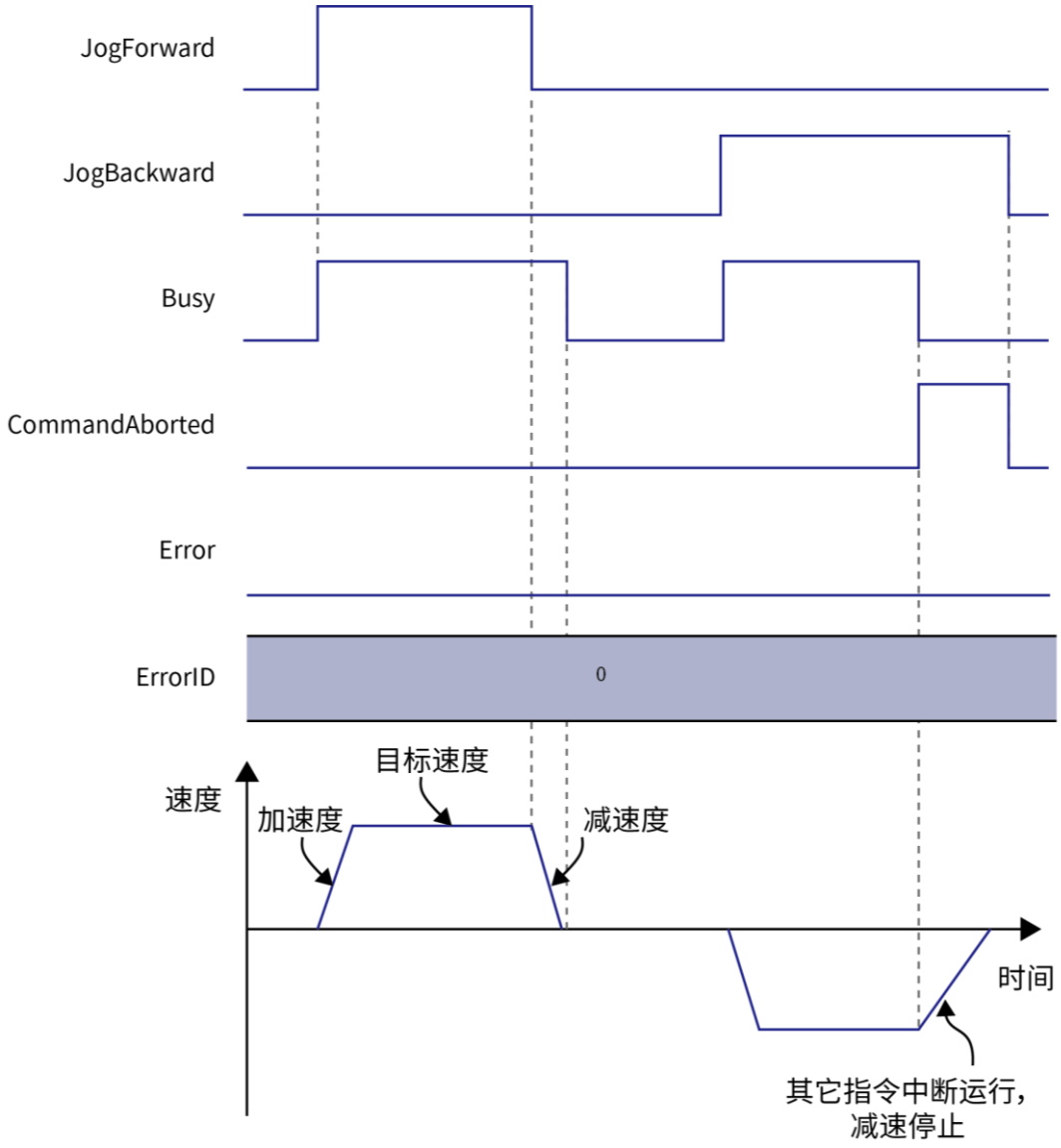


图 144

4) 注意事项

在执行本指令中发生异常时，Error（错误）变成 TRUE，轴停止动作。可查看 ErrorID（错误代码）的输出值，了解发生异常的原因。

◆ 发生异常时的时序图

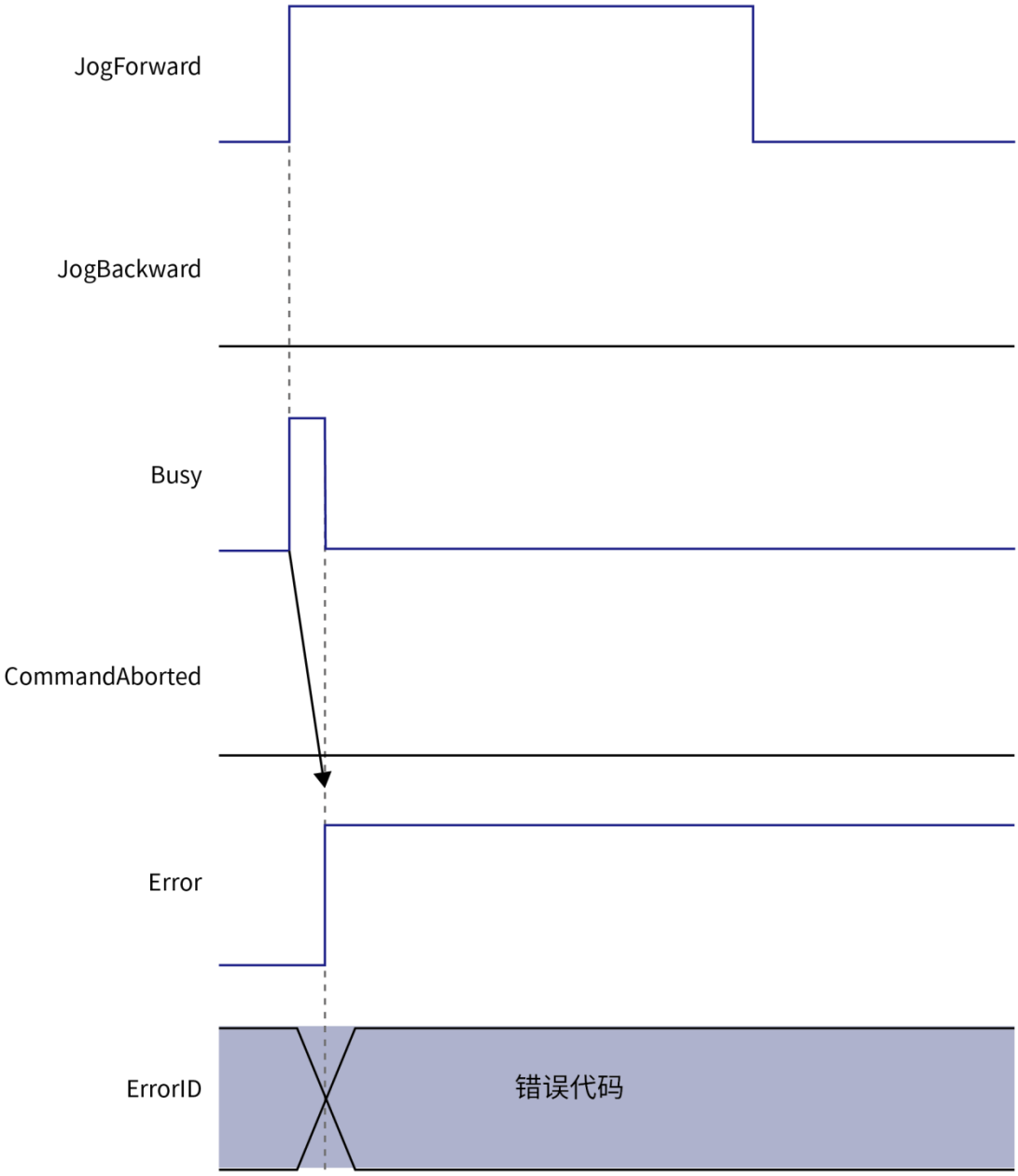
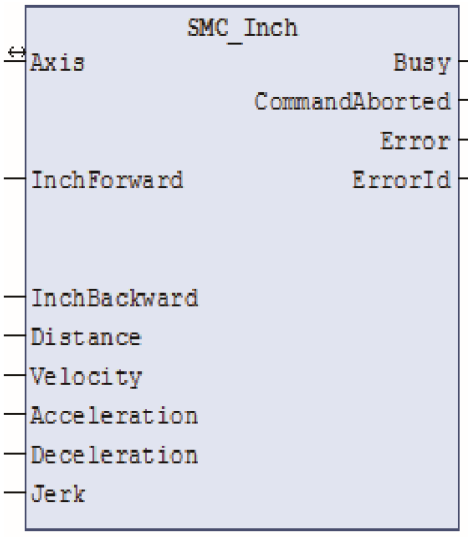

JogForward
JogBackward
Busy
CommandAborted
Error
ErrorID
错误代码

图 145

5.25 轴微动

指令名称：SMC_Inch

1) 指令格式

指令	名称	图形表现	ST 表现
SMC_Inch	轴微动指令		SMC_Inch0(Axis:= , InchForward:= , InchBackward:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InchForward	正向执行	BOOL	TRUE,FALSE	FALSE	如果 InchForward 为 TRUE，轴将按照 给定的速度运动 (Velocity, Acceleration, Deceleration) 以正向直到到达距离。输入必须被指定为 FALSE 然后再为 TRUE

					再次启动运动。 如果 InchForward 在到达位置之前被设置为 FALSE，那么轴将立即减速到 0 并且 Busy 将会被设置为 FALSE。 如果输入 InchBackward 在仿真情况下被设置为 TRUE，那么将不会有运动产生。
InchBackward	反向执行	BOOL	TRUE,FALSE	FALSE	如果 InchBackward 为 TRUE，轴将会按照给定的速度值进行运动 (Velocity, Acceleration, Deceleration) 以正向运动到 设定位置。然后输入必须被设置为 FALSE 然后再设置为 TRUE 启动另一个运动。 如果输入信号 InchForward 同时被设置为 TRUE，那么将不会有轴运动。
Distance	移动的距离	LREAL	数据范围	0	此数据为运动的距离
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3) 功能说明

轴单步运动控制，通过程序能实现一步接一步的单步控制。

本功能块运行状态为 Standstill 中， 指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。

Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

◆ 时序图

功能块的 InchForward/ InchBackward 必须有 TRUE/FALSE 的条件；

功能块的 Busy 表示当前功能块正在执行中；

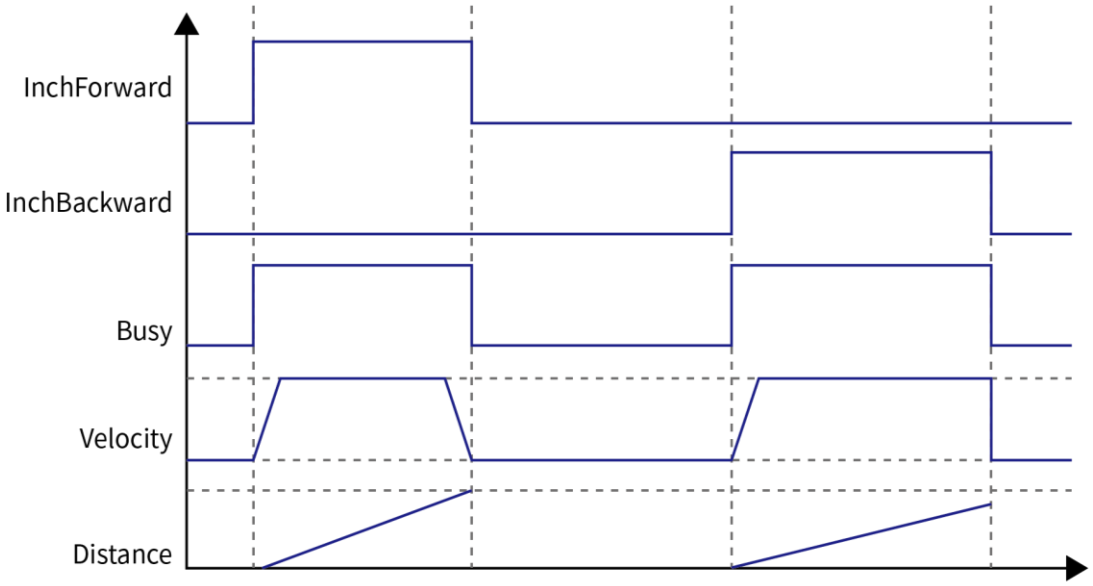
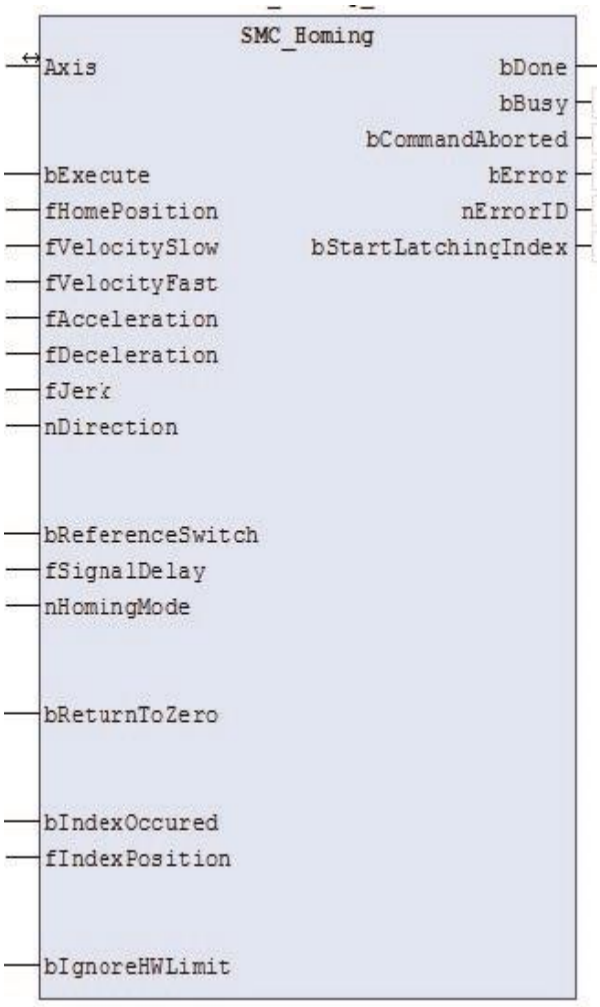


图 146

5.26 轴回零

指令名称：SMC_Homing

1) 指令格式

指令	名称	图形表现	ST 表现
SMC_Homing	轴回零		SMC_Homing0(Axis:= , bExecute:= , fHomePosition:= , fVelocitySlow:= , fVelocityFast:= , fAcceleration:= , fDeceleration:= , fJerk:= , nDirection:= , bReferenceSwitch:= , fSignalDelay:= , nHomingMode:= , bReturnToZero:= , bIndexOccured:= , fIndexPosition:= , bIgnoreHWLimit:= , bDone=> , bBusy=> , bCommandAborted=> , bError=> , nErrorID=> , bStartLatchingIndex=>);

2) 相关变量

◆ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	True 功能块执行，false 不执行功能块
fHomePosition	原点设定位置	LREAL		0	回零后原点设定位置，单位为用户标定后的单位
fVelocitySlow	慢速	LREAL		0	离开参考开关后慢速设定速度
fVelocityFast	快速	LREAL		0	离开参考开关置位时，快速设定速度
fAcceleration	加速度	LREAL		0	加速度设定值
fDeceleration	减速度	LREAL		0	减速度设定值
fJerk	加速度导数	LREAL		0	Jerk in [u/s3]
nDirection	回零方向	MC_DIRECTION		negative	回零开始方向，参考 MC_DIRECTION
bReferenceSwitch	参考开关	BOOL	TRUE,FALSE	FALSE	连接参考开关，TRUE: 参开开关触发， FALSE: 参考开关闭合
fSignalDelay	延迟	LREAL		0	参考开关的传输时间，用来补偿死区时间。 单位为秒。

nHomingMode	回零模式	SMC_HOMING_MODE			参考 SMC_HOMING_MODE

输入变量	名称	数据类型	有效范围	初始值	描述
bReturnTozero	返回零位	BOOL	TRUE,FALSE	FALSE	TRUE: 回零完成后轴运行到位置零（注意：如果 fHomePosition=10，则回零完成后轴位置变为 10，bReturnTozero 为 ture 则回零完成后轴反向走 10 个单位到 0 位）
bIndexOccured		BOOL	TRUE,FALSE	FALSE	True, 标志脉冲记录，回零模式为 FAST_ BSLOW_I_S_STOP，FAST_SLOW_I_S_ STOP 时生效
fIndexPosition		LREAL		0	标志脉冲时记录的位置
bIgnoreHWLimit	忽略硬限位	BOOL	TRUE,FALSE	FALSE	TRUE, 设定硬件限位开关使能为 false，如果相同的物理开关用于硬件限位开关和参考开关，那么硬件控制将被设置为假

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone		BOOL	TRUE,FALSE	FALSE	True, 回零完成
bBusy		BOOL	TRUE,FALSE	FALSE	True, 功能块正在生效中
bCommandAborted		BOOL	TRUE,FALSE	FALSE	True, 功能块被其他动作指令中断
Error		BOOL	TRUE,FALSE	FALSE	True, 错误发生
ErrorID		SMC_ERROR		0	错误代码，枚举型变量，查看帮助 smc_ error 查看具体报警代码
bStartLatchingIndex		BOOL	TRUE,FALSE	FALSE	由“bIndexOccured”和“fIndexPosition”共同作用产生

◆ 回零模式（SMC_HOMING_MODE）

枚举名称	类型	初 始值	描述
FAST_BSLOW_S_STOP	SMC_HOMING_MODE	0	按照设定方向以快速度走向原点开关，碰到原点开关后反向以慢速离开原点开关，离开后先执行 MC_setPosition 将当前位置设为 fHomePosition 设定值，然后执行 MC_stop
FAST_BSLOW_STOP_S	SMC_HOMING_MOD	1	按照设定方向以快速走向原点开关，碰到原点开关后反向以慢速离开原点开关，离开后先执行 MC_stop 将轴停止，然后执行 MC_setPosition，将当前位置设为 fHomePosition 设定值

FAST_BSLOW_I_S_STOP	SMC_HOMING_MOD	2	按照设定方向以快速走向原点开关，碰到原点开关后反向以慢速离开原点开关，bIndexOccured 信号到时先执行 MC_ setPosition 再执行 MC_stop
FAST_SLOW_S_STOP	SMC_HOMING_MOD	4	按照设定方向以快速走向原点开关，碰到原点开关后以慢速 离开原点开关，离开后先执行 MC_setPosition 将当前位置设为 fHomePosition 设定值，然后执行 MC_stop
FAST_SLOW_STOP_S	SMC_HOMING_MOD	5	按照设定方向以快速走向原点开关，碰到原点开关后以慢 速离开原点开关，离开后先执行 MC_stop，然后执行 MC_ setPosition 将当前位置设为 fHomePosition 设定值。
FAST_SLOW_I_S_STOP	SMC_HOMING_MOD	6	按照设定方向以快速走向原点开关，碰到原点开关后反向以慢速离开原点开关，bIndexOccured 信号到时先执行 MC_ setPosition 再执行 MC_stop

3) 功能说明

轴回零指令，与 MC_Home 有区别，MC_Home 在轴配置处设置回零方式，该指令为控制器控制的回零方式。

SMC_HOMING 通过 bExecute 的上升沿启动之后，轴将会按照速度 fVelocityFast 并以 nDirection 定义的方向开始运动，直到 bReferenceSwitch = FALSE。然后轴将会缓慢停止并按照相反的方向以速度 fVelocitySlow 离开参考开关。bReferenceSwitch = TRUE 后回零完成，使能回零指令后 bReferenceSwitch 的状态为 ON->OFF->ON，在 OFF->ON 的上升沿回零完成，设置参考位置。

参考位置 =fHomePostion+（（fSignalDelay*1000+1 个 DC 时钟周期）/1000）*fVelocitySlow 实际就是补偿了设置的 bReferenceSwitch 采样延迟和一个通讯周期位移延迟。

如果 bReturnToZero=TRUE，bReferenceSwitch 的状态在 OFF->ON 的上升沿将参考位置设置为 fHomePostion+（（fSignalDelay*1000+1 个 DC 时钟周期）/1000）

*fVelocitySlow，然后按照速度 fVelocityFast 运行到 0 位置。

注意：Done 完成信号后，轴位置设定为：fHomePosition。设定的时机跟 nHomingMode 有关（详情参考 SMC_HOMING_MODE）。下图为几种回零模式：

回零模式“0”时：

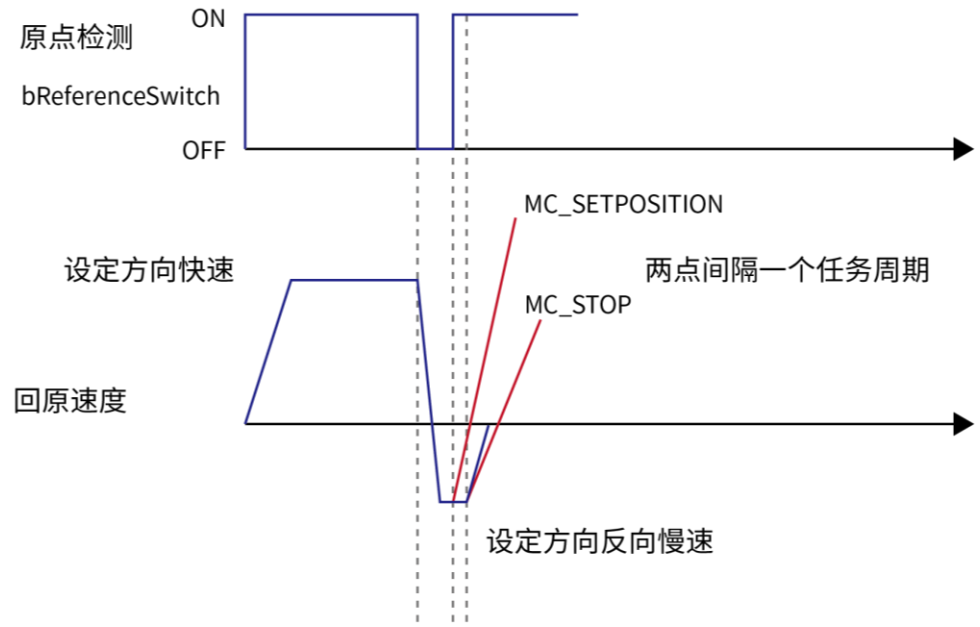


图 147

回零模式“1”时

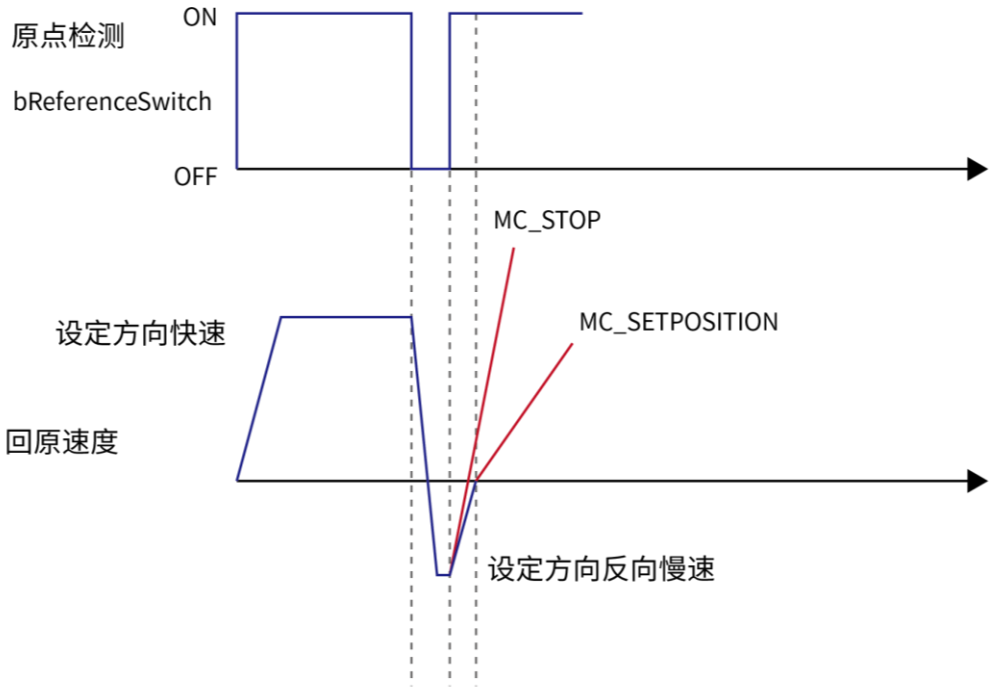


图 148

回零模式“4”时

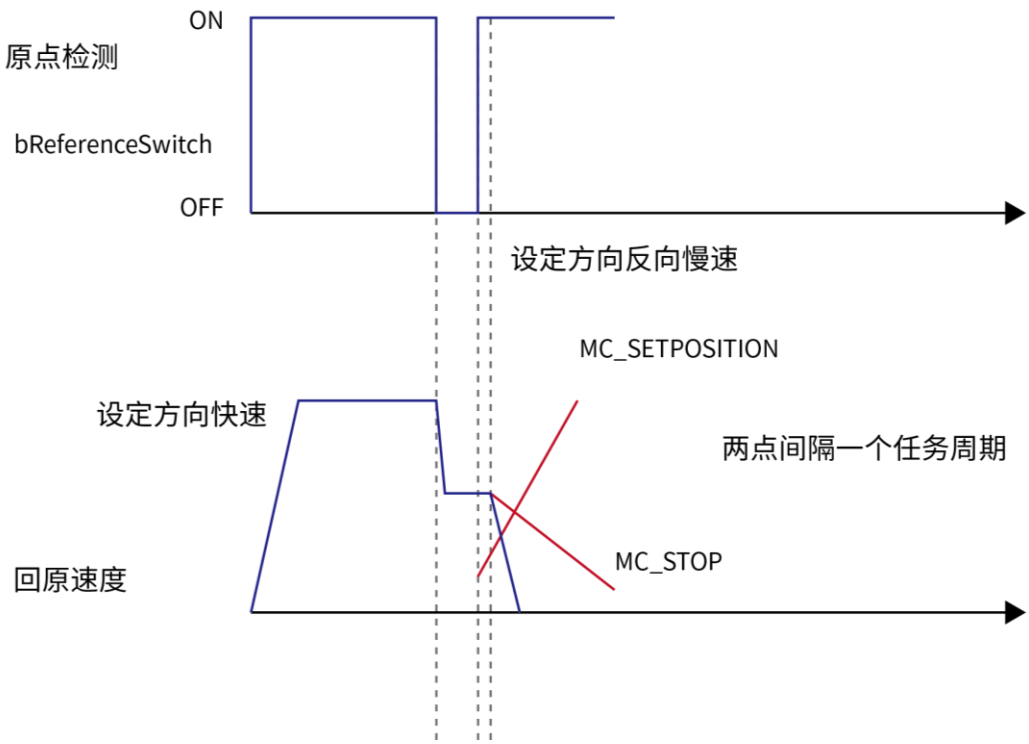


图 149

回零模式“5”时

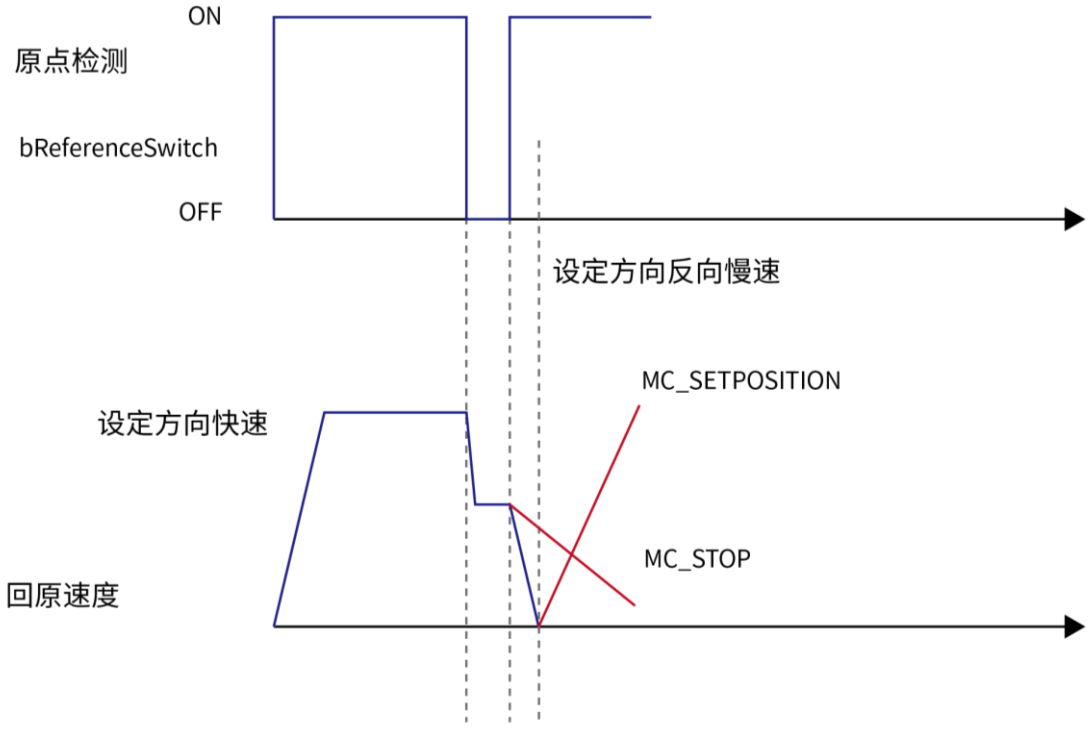


图 150

◆ 时序图

指令执行时 bReferenceSwitch TRUE 时

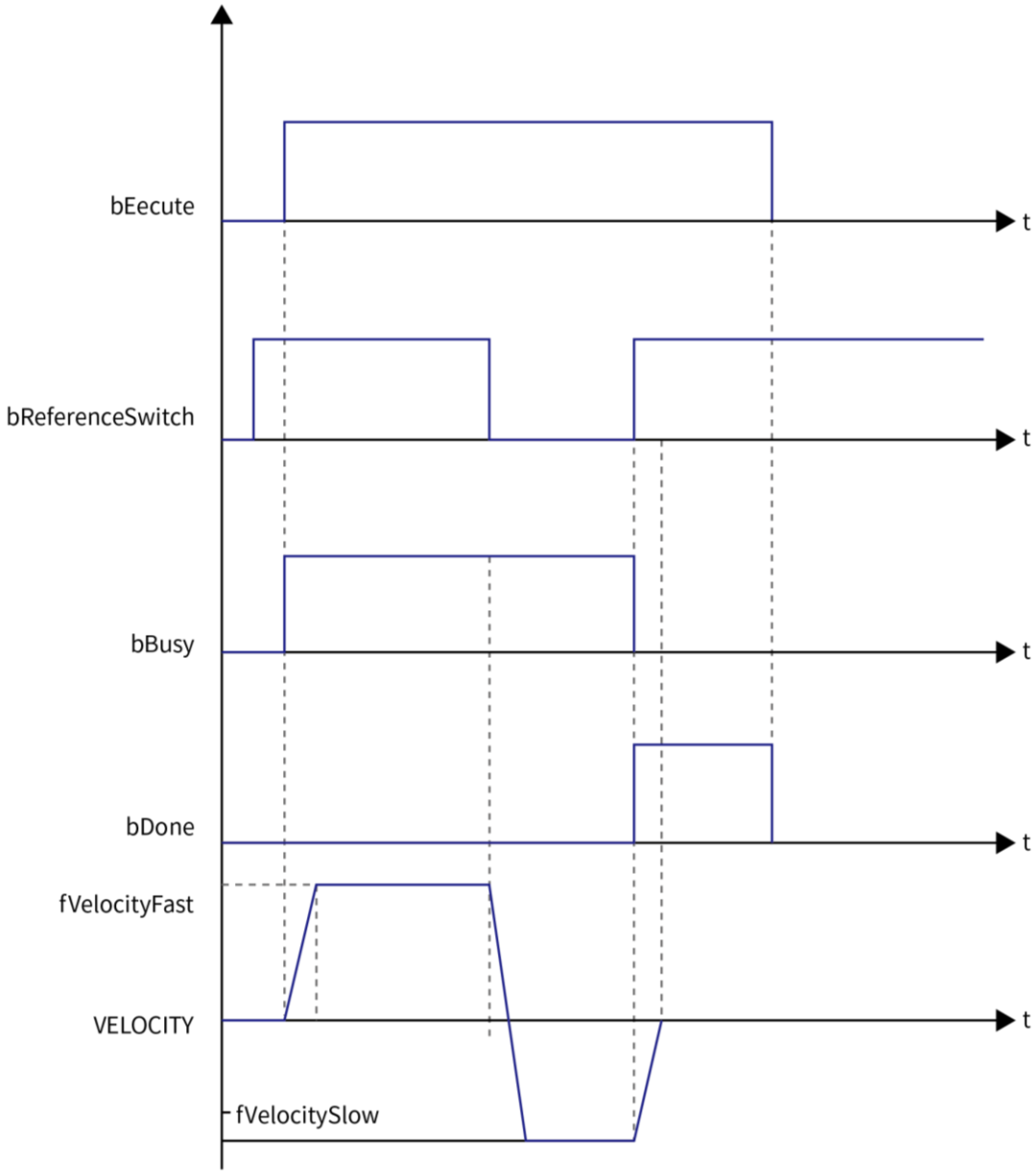


图 151

指令执行时 bReferenceSwitch FALSE 时

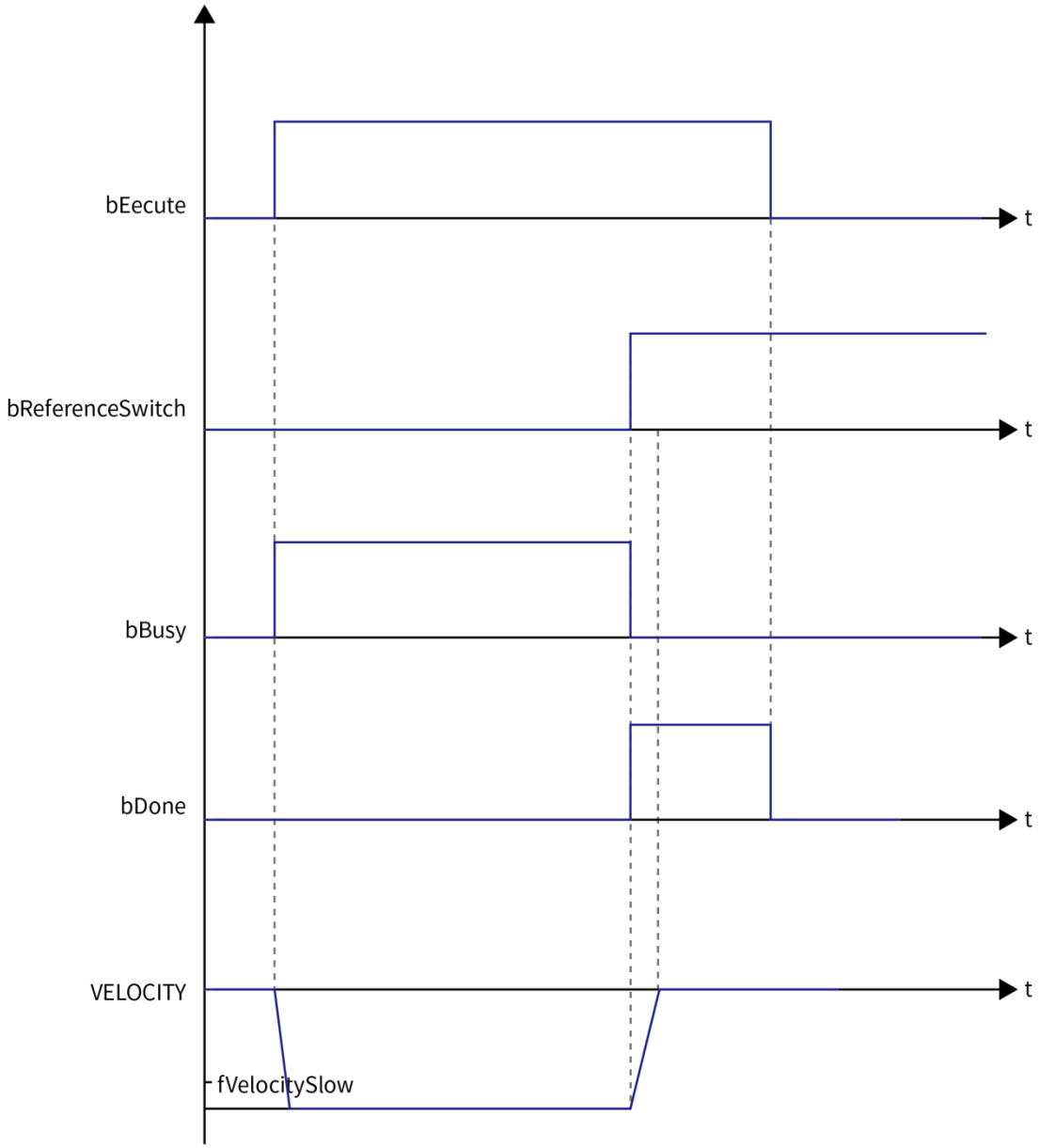


图 152

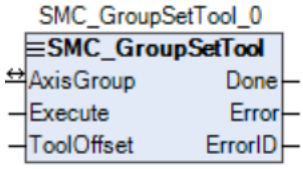
4) 注意事项

输入轴类型出错、轴有错误、轴没有使能速度或加速度无效，会导致指令报错。

5.27 工具坐标

名称：SMC_GroupSetTool

1) 指令格式

指令	名称	图形表现
SMC_GroupSetTool	轴组工具坐标设置	

2) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
AxisGroup	AXIS_GROUP_REF_SM3		映射轴组

◆ 输入变量

输入变量	数据类型	初始值	描述
Execute	BOOL		上升沿设置工具坐标
ToolOffset	SMC_POS_REF		工具偏移坐标

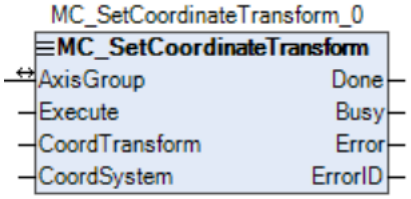
◆ 输出变量

输出变量	数据类型	初始值	描述
Done	Bool		轴组工具坐标设置完成
Error	Bool		功能块发生错误
ErrorID	SMC_ERROR		错误代码

5.28 用户坐标

名称：MC_SetCoordinateTransform

3) 指令格式

指令	名称	图形表现
MC_SetCoordinateTransform	轴组用户坐标设置	

4) 相关变量

◆ 输入输出变量

输入输出变量	数据类型	初始值	描述
AxisGroup	AXIS_GROUP_REF_SM3		映射轴组

◆ 输入变量

输入变量	数据类型	初始值	描述
Execute	BOOL		上升沿设置 Y 坐标
CoordTransform	MC_COORD_REF		用户坐标偏移值

CoordSystem	MC_COORD_SYSTEM		坐标系选择:PCS-1,PCS-2,MCS

◆ 输出变量

输出变量	数据类型	初始值	描述
Done	Bool		轴组用户坐标设置完成
Busy	Bool		功能块未完成
Error	Bool		功能块发生错误
ErrorID	SMC_ERROR		错误代码