

前言

感谢您选用 DCCE 网络化可编程控制器，手册提供了 DCCE 网络化可编程控制器必要的编程和使用信息。为了确保能够正确使用产品，请认真阅读手册。

手册内容已经过严格确认，与所描述的软硬件信息相符合。由于遗漏或疏忽可能造成某些地方的错误，发现后请及时反馈给我们，我们会在后续版本中进行更正。

安全指南

手册包括应该遵守的注意事项，以保证人身安全，保护产品和所连接的设备免受损坏。注意事项分为四类：“危险”、“警告”、“注意”和“提示”。

注意事项分为四类：“危险”、“警告”、“注意”和“提示”。

危险：表示不正确的操作将导致危险情况发生，造成严重的人身伤害；

警告：表示不正确的操作将导致危险情况发生，造成中度或轻微的人身伤害；

注意：表示不正确的操作将导致产品无法正常工作，造成该部分功能无法实现或设备损坏；

提示：表示使用建议的操作方法能使产品获得更好的性能或是对部分说明更详尽的补充；

手册用途

手册提供了 DCCE 网络化可编程控制器的产品功能、编程方法、通信方法、管理方法、指令集和程序示例等。根据手册说明对控制器编程，可减少因编程方法不当造成的损失。

适用产品范围

适用于 DCCE 网络化可编程控制器，包括 PEC/MAC/AIO/DIO/MEC/EDC 等系列硬件产品。

认证标准

- **CE 认证：**DCCE 网络化可编程控制器符合 CE 认证标准
- **ROHS 认证：**DCCE 网络化可编程控制器符合 ROHS 认证标准

相关专有名词

- **AIO：**Analog Input Output，模拟量信号输入输出
- **DIO：**Digital Input Output，数字量信号输入输出
- **DCCE：**DUT Computer Control Engineering Co.LTD，大连理工计算机控制工程有限公司
- **DUT：**Dalian University of Technology，大连理工大学
- **EPA：**Ethernet for Plant Automation，中国制定的第一个现场总线标准，被列为国际现场总线标准

IEC61158 第 14 类型

- **FBD：**Function Block Diagram，功能块图表，基于 IEC61131-3 标准的一种编程语言
- **LAD：**Ladder Diagram，梯形图表，基于 IEC61131-3 标准的一种编程语言
- **HMI：**Human Machine Interface，人机界面接口
- **IEC：**International Electronic Committee，国际电工委员会
- **MB+：**Modbus Plus，指在原有 RS485 总线基础上通过特有技术实现的一种速率可达 1Mbps 的总线
- **PEC：**Programmable Ethernet Controller，可编程以太网控制器
- **PLC_Config：**可编程控制器编程软件，DCCE 公司编写
- **TAC：**Temperature Acquisition Controller，温度采集控制器
- **MAC：**Master Controller，主控制器

目 录

前言	1
目 录	2
1 PLC_Config 和 DCCE 控制器的基本信息	9
1.1 PLC_Config 和 DCCE 控制系统简介	9
1.1.1 PLC_Config	9
1.1.2 DCCE 控制系统	9
1.2 PLC_Config 界面	10
1.3 快捷键	13
1.4 联机帮助	13
2 编程	14
2.1 控制程序作业	14
2.1.1 程序组成	14
2.1.2 程序资源	15
2.1.3 程序执行逻辑	15
2.2 项目元件及工作原理	16
2.2.1 状态符号表	17
2.2.2 交叉引用表	18
2.2.3 内存表管理	19
2.2.4 只读变量区管理表	19
2.2.5 趋势图	20
2.2.6 程序参数	22
2.3 编写程序	23
2.3.1 编写梯形图程序	23
2.3.2 编写功能块图程序	33
2.4 下载程序	43
2.4.1 编译检错	43
2.4.2 程序下载	44
2.5 上载程序	44
2.6 监控与仿真	44
2.6.1 设备的运行与停止	44
2.6.2 强制写入	45
2.6.3 在线调试	45
2.6.4 仿真设备运行	46
2.7 PLC_Config 软件操作	47
2.7.1 主程序与子程序	47
2.7.2 子程序参数编辑	49
2.7.3 连线操作	50
2.7.4 网络操作	51
2.7.5 指令参数编辑	52
2.7.6 功能块梯形图互转	54

2.8 系统工程创建步骤.....	58
2.8.1 创建工程.....	58
2.8.2 添加控制器.....	58
2.8.3 添加从设备.....	60
2.8.4 添加指令.....	60
2.8.5 添加变量.....	60
2.8.6 编写并下载程序.....	61
2.8.7 添加符号.....	63
2.8.8 在线监视.....	63
3 内存变量.....	64
3.1 变量区分类.....	64
3.2 变量使用方式.....	65
3.3 变量存储方式.....	66
3.4 变量寻址方式.....	67
3.5 特殊功能变量区.....	72
3.6 变量区操作方式.....	76
4 通信方式.....	77
4.1 串行通信服务.....	77
4.1.1 串口通信配置.....	77
4.1.2 串行通信参数.....	78
4.1.3 Modbus 通信.....	79
4.1.4 Profibus_DP 通信.....	81
4.2 以太网通信服务.....	84
4.2.1 以太网通信配置.....	84
4.2.2 以太网自由通信.....	85
4.2.3 EPA 协议通信.....	86
4.2.4 ModbusTCP/ModbusUDP 通信.....	90
4.3 转发通信.....	90
4.3.1 基本概念.....	90
4.3.2 功能说明.....	91
4.3.3 性能指标.....	91
4.3.4 配置转发参数.....	92
4.3.5 配置实例.....	94
4.3.6 网关配置.....	97
5 系统管理.....	100
5.1 分布式控制系统.....	100
5.2 指令网络通信.....	100
5.3 主从网络通信.....	102
5.3.1 从设备基本概念.....	102
5.3.2 从设备的种类与连接.....	102
5.3.3 参数配置.....	104

5.3.4 配置实例.....	110
5.3.5 从设备组态.....	113
5.4 扩展 IO 通信系统.....	114
6 可编程控制指令集.....	118
6.1 指令集概述.....	118
6.2 指令描述.....	120
6.3 梯形图指令.....	121
6.4 位逻辑指令.....	123
6.4.1 与和或.....	123
6.4.2 沿触发器.....	124
6.4.3 输出.....	125
6.4.4 置位和复位.....	127
6.4.5 触发指令.....	129
6.5 网络通信指令.....	130
6.5.1 EPA 通信	130
6.5.2 UDP 自由通信	137
6.5.3 TCP 自由通信	141
6.5.4 串行通信.....	146
6.5.5 串行自由通信.....	151
6.5.6 USS 通信	159
6.5.7 MPI 通信.....	166
6.6 比较指令.....	173
6.6.1 无符号整数.....	173
6.6.2 有符号整数.....	174
6.6.3 有符号长整数.....	176
6.6.4 实数.....	177
6.7 转换指令.....	179
6.7.1 实数与整数.....	179
6.7.2 整数与 BCD 码.....	180
6.7.3 长整数与实数.....	181
6.7.4 取整.....	182
6.7.5 长整数与整数.....	183
6.7.6 字与字节.....	184
6.8 计数器指令.....	186
6.8.1 计数器.....	186
6.8.2 高速计数指令	188
6.8.3 高速脉冲输出指令.....	197
6.9 实数运算指令.....	200
6.9.1 实数加减乘除.....	200
6.9.2 数学功能.....	201
6.9.3 比例积分微分 (PID)	209
6.10 整型数运算指令.....	218
6.10.1 整数加减乘除.....	218

6.10.2 递增和递减.....	223
6.11 逻辑操作指令.....	225
6.11.1 字取反和双字取反指令.....	225
6.11.2 与_或_异或.....	226
6.12 变量赋值指令.....	228
6.12.1 字节_字_双字_实数.....	228
6.12.2 字块.....	229
6.12.3 立即传送.....	232
6.13 程序控制指令.....	234
6.13.1 循环和跳转.....	234
6.13.2 子程序调用.....	235
6.13.3 G 代码调用.....	236
6.14 中断指令.....	240
6.14.1 中断允许和禁止指令.....	240
6.14.2 中断连接指令.....	240
6.14.3 中断分离指令.....	241
6.15 移位与循环指令.....	246
6.15.1 左移和右移.....	246
6.15.2 循环左移和循环右移.....	247
6.15.3 移位寄存器.....	249
6.15.4 交换.....	250
6.16 定时器指令.....	251
6.16.1 定时接通指令 (TON)	251
6.16.2 定时断开指令 (TOF)	252
6.16.3 保持定时接通指令 (TONR)	253
6.17 专用指令.....	254
6.17.1 工件检测.....	254
6.17.2 经济运行.....	256
6.17.3 存储.....	257
6.17.4 冒泡排序.....	258
6.17.5 数据校验.....	259
6.17.6 定位.....	263
6.17.7 往复运动.....	269
6.17.8 斜率检测.....	272
6.17.9 PH 检测.....	273
6.17.10 电导率检测.....	275
6.17.11 溶氧检测.....	277
6.17.12 Na 离子检测.....	279
6.17.13 SD 卡设置指令.....	282
6.17.14 气缸控制指令.....	283
6.17.15 IO 状态监控指令.....	284
7 运动控制指令集.....	286
7.1 运动控制指令表.....	286

7.2 运动控制状态机.....	287
7.3 运动控制参数.....	289
7.3.1 单轴参数配置.....	289
7.3.2 轴组参数配置.....	291
7.3.3 轴组运动模型参数配置.....	291
7.4 运动控制状态监控.....	295
7.5 运动控制指令概述.....	296
7.5.1 指令触发方式.....	296
7.5.2 指令引脚时序.....	297
7.5.3 指令引脚参数.....	298
7.5.4 加减速控制曲线.....	298
7.5.5 缓冲模式.....	301
7.6 单轴指令.....	302
7.6.1 绝对定位 MC_MoveAbsolute.....	302
7.6.2 相对定位 MC_MoveRelative.....	305
7.6.3 叠加定位 MC_MoveAdditive.....	308
7.6.4 重新定位 MC_MoveSuperImposed.....	310
7.6.5 恒速运行 MC_MoveVelocity.....	312
7.6.6 停止轴 MC_Stop.....	314
7.6.7 点动运行 MC_Jog.....	316
7.6.8 回原点 MC_Home.....	319
7.6.9 扩展回原点 MC_HomeEx.....	329
7.6.10 往复运行 MC_Reciprocator.....	331
7.6.11 复位轴 MC_Reset.....	335
7.6.12 读轴状态 MC_ReadStatus.....	337
7.6.13 读轴故障 MC_ReadAxisError.....	339
7.6.14 读轴参数 MC_ReadParameter.....	341
7.6.15 写轴参数 MC_WriteParameter.....	342
7.6.16 读轴布尔型参数 MC_ReadBoolParameter.....	343
7.6.17 写轴布尔型参数 MC_WriteBoolParameter.....	344
7.6.18 读轴位置 MC_ReadActualPosition.....	345
7.6.19 写轴位置 MC_WriteActualPosition.....	346
7.6.20 编码器轴配置 MC_CoderMap.....	347
7.6.21 立即停止配置指令 MC_EmgStopSet.....	350
7.7 多轴指令.....	351
7.7.1 凸轮运行 MC_CamIn.....	351
7.7.2 凸轮分离 MC_CamOut.....	354
7.7.3 齿轮运行 MC_GearIn.....	356
7.7.4 齿轮分离 MC_GearOut.....	359
7.7.5 相位偏移 MC_Phasing.....	361
7.7.6 创建凸轮表 CamTableCreate.....	363
7.7.7 删除凸轮表 CamTableDelete.....	364
7.7.8 写凸轮表数据 CamTableWriteData.....	365
7.7.9 读凸轮表数据 CamTableReadData.....	366

7.7.10 读从轴位置 CamTableReadSlavePos.....	368
7.7.11 多轴同步运行 MC_MultiAxesSyncMove.....	369
7.7.12 旋切耦合 MC_APF_RotaryCutIn	375
7.7.13 旋切初始化 MC_APF_RotaryCutInit	381
7.7.14 旋切脱离 MC_APF_RotaryCutOut.....	382
7.8 轴组指令.....	383
7.8.1 绝对直线运行 MC_MoveLinearAbsolute.....	383
7.8.2 相对直线运行 MC_MoveLinearRelative.....	386
7.8.3 绝对样条运行 MC_MoveSplineAbsolute.....	389
7.8.4 轴组点动运行 MC_GroupJog.....	392
7.8.5 轴组写实际位置 MC_GroupWriteActPosition.....	394
7.8.6 轴组恒速运行 MC_GroupMoveVelocity.....	396
7.8.7 停止轴组 MC_GroupStop	398
7.8.8 平面圆弧运行 MC_MoveCircular2D	399
7.8.9 连续曲线运行 MC_ContinuousCurve.....	402
7.8.10 读轴组位置 MC_GroupReadActualPosition.....	414
7.8.11 读轴组速度 MC_GroupReadActualVelocity.....	416
7.8.12 读取轴组故障 MC_GroupReadError	417
7.8.13 轴组复位 MC_GroupReset.....	418
7.8.14 读轴组状态 MC_GroupReadStatus.....	418
7.8.15 轴组使能 MC_GroupEnable	420
7.8.16 轴组禁能 MC_GroupDisable	421
7.8.17 空间圆弧运行 MC_MoveCircular3D.....	422
8 重点示例应用.....	433
8.1 PID 控温程序示例	433
8.2 电机往复运动程序示例.....	436
8.3 系统通信程序示例.....	437
8.4 四轴纺纱控制示例.....	440
8.5 三轴高速肥皂切割机控制示例.....	444
8.6 直角机器人编程示例.....	448
8.7 三轴机器人编程示例.....	451
附录 A 参数设置和说明	456
附录 B 特殊功能变量区 (SM)	458
(1) 厂家参数.....	458
(2) 设备参数.....	458
(3) 功能参数.....	459
(4) 通信参数.....	461
(5) IO 配置参数	462
(6) 从设备控制参数.....	463
(7) 指令状态参数.....	464
(8) 运动控制参数.....	470

附录 C 从设备资源和特性.....	471
(1) DIO1000 和 DIO2000.....	471
(2) TAC60 简单模式下映射的资源如下	471
(3) TAC3000	472
(4) DIO3000.....	473
(5) DIO3400.....	475
(6) PEC3000 和 PEC3400 和运动控制系列产品	477
(7) PEC5000 和 AIO5000 和 AIO4000 和 PEC9000	479
(8) PEC8000 和 PMC2000	481
(9) NCC2000	483
附录 D 指令码堆栈描述表.....	485
附录 E 运动控制错误码.....	488
附录 F 运动控制参数监控寄存器	490
附录 G 运动控制轴参数定义.....	492
附录 H 常见问题	493
附录 K 版本信息	495

1 PLC_Config 和 DCCE 控制器的基本信息

1.1 PLC_Config 和 DCCE 控制系统简介

1.1.1 PLC_Config

PLC_Config 编程软件是用于对 DCCE 网络化可编程控制器和运动控制器进行逻辑组态的编程开发软件。包括设备管理、通信配置、程序管理、状态符号表、趋势图、仿真运行等多项便于编程的功能模块，支持 IEC61131-3 标准的功能块、梯形图编程语言，具有逻辑指令，运算指令，定时/计数器指令，控制指令，中断指令，网络通信指令，专用指令，以及符合 PLCopen 标准的单轴指令，多轴指令，轴组指令等，实现了用户程序的优化编译、分布式下载、调试与加密。

PLC_Config 编程软件为用户管理控制器及其从设备、开发程序、监视变量等提供了良好的环境，PLC_Config 可以通过以太网或 RS485 串口两种通信方式和控制器连接。支持 PEC、MAC、AIO、DIO、PMC、EDC、MEC、iTC 等系列控制器。

PLC_Config 软件是基于 Windows 操作系统的应用软件，它支持 32 位的 Windows2000、Windows2003、XP、Windows7 等操作系统。使用 PLC_Config 时，建议用户的计算机系统要高于以下配置：

- CPU 主频 $\geq 1\text{GHz}$ ；
- 内存 $\geq 256\text{M}$ ；
- 显示器分辨率 $\geq 1024\times 768$ ；
- 硬盘剩余空间 $\geq 512\text{M}$ ；

如果用户需要通过串口对控制器编程，计算机需要具有 RS232 或 RS485 接口及 RS232/485 转换器。用户也可以使用 USB 转串口设备替代计算机串行口。

1.1.2 DCCE 控制系统

DCCE 网络化可编程控制系统主要包括 PEC/MAC/AIO/DIO/MEC/EDC 等系列硬件产品，以及编程开发软件 PLC_Config 和监控管理开发软件 DView，具备全分布式控制、多种网络集成、数据采集、远程监控等功能，为各类复杂工程应用提供低成本、高效能的通用解决方案，实现了工厂协同自动化。

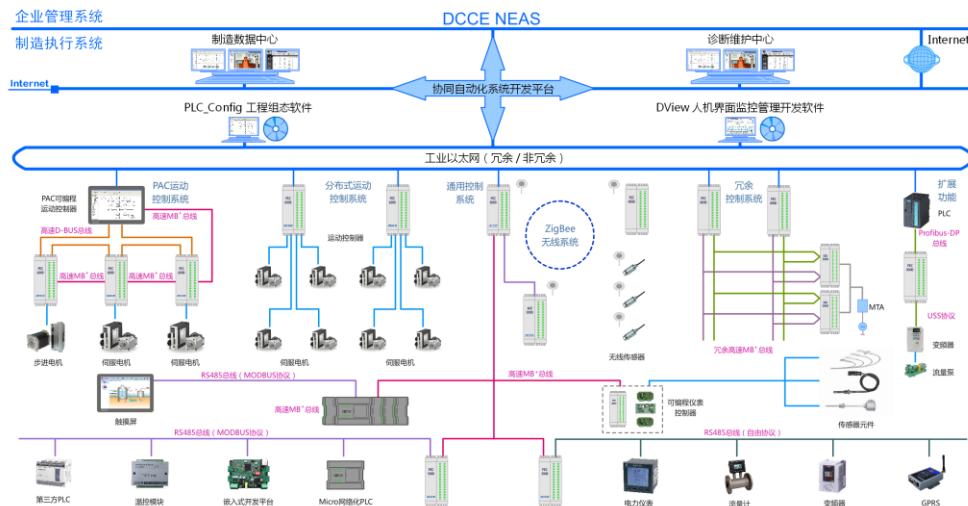


图 1.1.1 DCCE 网络化可编程控制系统

DCCE 控制系统如图 1.1.1 所示，系统支持多级从设备扩展和第三方设备接入，每级可扩展 16 个从设备，单台设备控制规模包括 10/20/30/40 多种 I/O 点数组组合，基于对主流控制网络的良好兼容性和分布式控制特点，可以实现大中小型可编程控制系统的各种功能。

系统支持 IEC61131-3 标准的梯形图、功能块编程，具备高速输入输出、PID 自整定控制等功能；实现了设备管理、通信管理、变量管理、画面管理、视频管理和制造执行数据管理。

系统提供了符合 PLCopen 标准的单轴，多轴，轴组三大类运动控制指令集，支持空间直线、空间圆弧、

NURBS 曲线插补等多种插补算法，支持梯形曲线，S 曲线，E 指数曲线等多种加减速控制曲线，支持直角模型、圆柱模型、四轴并联模型、三轴并联模型、三轴座椅模型、通用模型等多种动力学模型。

系统拥有丰富的行业参考案例，为工程人员提供了开放、灵活、标准的模块接口，极大地缩减了系统部署的时间与人力成本。网络级别的双重冗余，为系统的可靠运行提供强有力的保证。DCCE 网络化可编程控制器主要用于钢铁、冶金、电力、水泥、医药等流程行业以及纺织机械、塑料机械、电子产品制造、包装机械等装备制造中，为相关企业提升了设备自动化程度和工作效率，降低了物耗，节约了能耗，提升了产品竞争能力。

1.2 PLC_Config 界面

PLC_Config 编程软件作为 PLC 组态工程开发软件，主要完成程序的开发、编译、上下载和变量监控等功能。PLC_Config 元件包括菜单，工程树，符号状态表，交叉引用表等。

PLC_Config 的主界面如图 1.2.1 所示：



图 1.2.1 PLC_Config 主界面

表 1.2.1 主界面说明

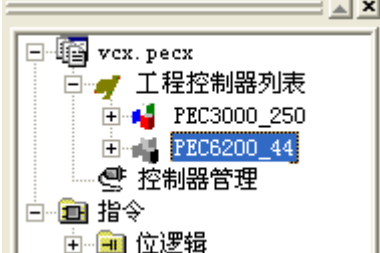
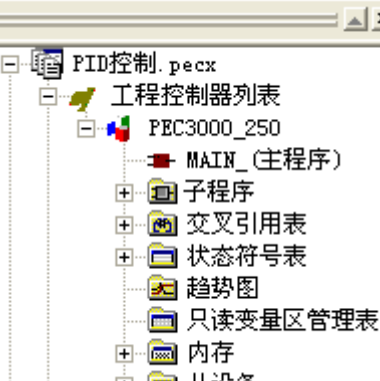
元件	说明	元件	说明
菜单	列出 PLC_Config 软件的所有功能	状态符号表	提供符号变量编辑、监控功能的变量表格
工程树	列出工程的主要信息和指令树	交叉引用表	用于查看当前程序变量引用情况的表格
工具条	列出 PLC_Config 的常用功能	趋势图	可观察变量在一个连续时间段内的变化情况
功能块编辑区	支持功能块语言的编程界面	输出栏	提供控制器上下线、程序编译、搜索结果等信息显示的窗口
梯形图编辑区	支持梯形图语言的编辑界面	状态栏	显示用户编辑的控制器的基本信息

下面是工程树、输出栏，两个较为复杂的常用元件的功能介绍，其他元件的使用说明，可参见《PLC_Config 使用说明书》。

(1) 工程树

工程树提供所有项目对象和为当前程序编辑器提供的所有指令的树型视图。工程的主要信息包含：工程名、控制器列表、主程序、子程序、交叉引用表、状态符号表、趋势图、只读变量区、内存表、从设备列表、控制器管理和指令树等内容。

表 1.2.2 工程树说明

树形结点	结点说明
	工程树：列出了工程控制器。其中彩色图标表明控制器在线，灰色图标表明控制器离线 指令树：提供 PLC_Config 支持的所有指令。您可通过双击指令结点的方式向程序编辑区中添加指令
	工程树结点下列出的内容包括： 主程序：控制器开始运行后首先执行主程序 子程序：控制器可通过调用或中断的方式执行子程序 交叉引用表：统计变量地址的使用情况 状态符号表：用于编辑和监控变量 趋势图：可显示出变量在某段时间内的变化情况 只读变量区管理表：可初始化只读变量区 内存：可对变量数据进行监控 从设备：用户通过从设备结点对从控制器信息进行配置

(2) 输出栏

输出栏提供了较为丰富的控制器信息，包括系统信息，编译，交叉引用和查找窗口。

➤ 系统信息

系统信息栏显示了控制器的上下线信息，以及这些控制器的 IP 地址和模块的串口地址，方便用户及时查看控制器的状态。

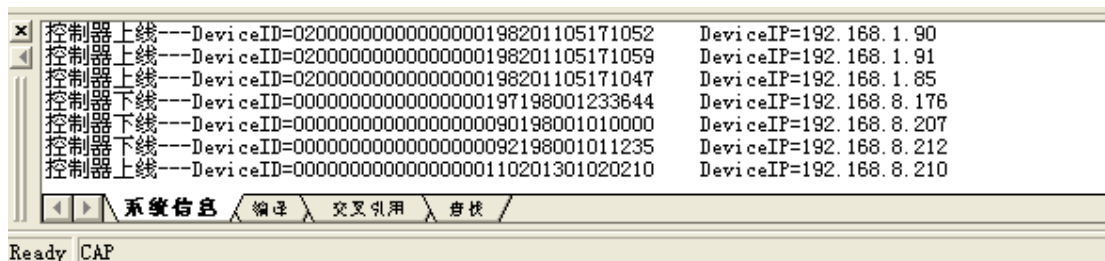


图 1.2.2 输出栏—系统信息

编译信息

在编译栏中，可以看到编译后的各项信息：编译是否成功、编译出错时的错误位置、编译警告信息等，双击错误位置可以定位到出错的功能块或梯形图。

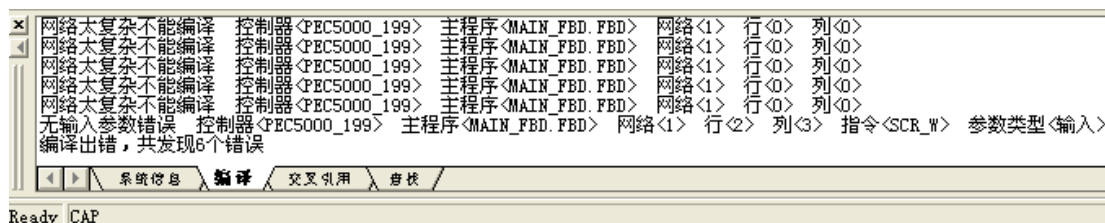


图 1.2.3 输出栏—编译

► 交叉引用

当需要查看工程中变量的引用情况时，在输出栏中选择“交叉引用”选项卡，当选中某一指令或变量时，在该处即可查看到选中指令的使用情况。具体现象如图 1.2.4 所示：



图 1.2.4 交叉引用查询

➤ 查找

系统主菜单“编辑->查找”，或点击键盘组合键 Ctrl+F，或点击工具条中查找  按钮，均可启动查找功能。

在打开的“查找参数”对话框中填写查找内容，在配置选项中选择查找参数的匹配项，在查找范围中选择查找该变量的范围，选择完成后点击查找按钮，如图 1.2.5 所示。

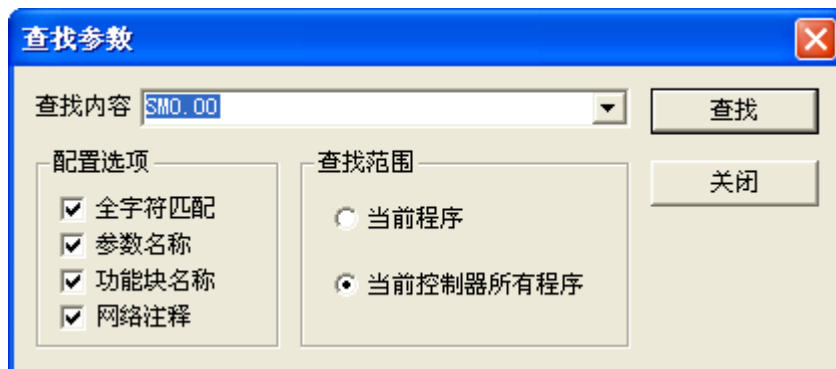


图 1.2.5 查找参数对话框

查找的结果将列在输出栏的“查找”一栏中。

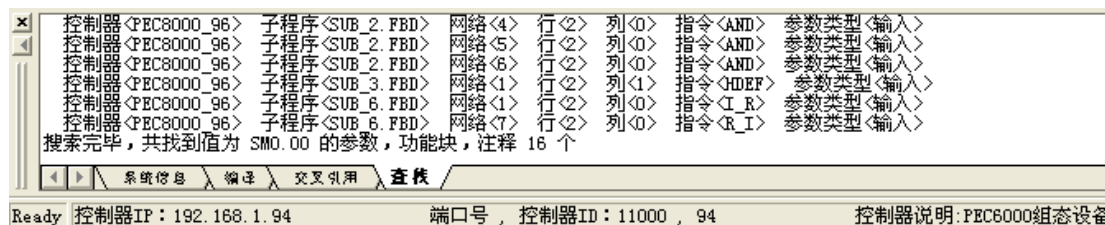


图 1.2.6 查找结果显示

查找过的变量会自动存储到查找参数对话框中，再次查找是可以通过“查找内容”后的下拉框选择已查找过的变量，方便了用户对变量的再次查找。

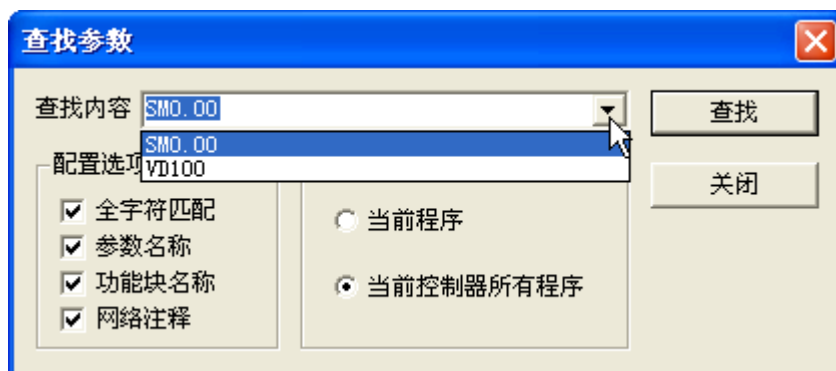


图 1.2.7 查找内容记录

1.3 快捷键

PLC_Config 软件提供多个快捷键，可方便用户有效地完成多项任务。表 1.3.1 中列出了所有本软件支持的快捷键。

表 1.3.1 组态软件中的快捷键

快捷键	功能	快捷键	功能
F1	打开帮助文档并直接链接到帮助主题	F12	跳转到该程序的子程序中
Shift+F3	跳转到上一个书签	Enter	光标在功能块参数位置时, 使该参数处于编辑状态, 光标在功能块参数位置且处于编译状态时, 结束编辑
F5	运行控制器	Home	删除网络行
Shift+F5	停止运行控制器	End	增加网络行
F6	编辑界面中启动/停止程序监控; 状态符号表界面中启动/停止状态表监控	BackSpace (退位键)	删除当前网络中元件或连线, 并向前移动一格
F7	编译所有程序	Delete (删除键)	删除当前网络内容
Shift+F7	编译当前页面程序	PageUp	显示上一屏的内容
↑ (上)	向上移动光标	Ctrl+↑ (上)	向上连线
↓ (下)	向下移动光标	Ctrl+↓ (下)	向下连线
← (左)	向左移动光标	Ctrl+← (左)	向左连线
→ (右)	向右移动光标	Ctrl+→ (右)	向右连线
Ctrl + Home	跳转到程序开始 (1 号网络位置)	Ctrl + End	跳转到程序结束 (最后一个网络位置)
Ctrl + C	复制指令或网络	Ctrl + V	粘贴指令或网络
Ctrl + X	剪切指令或网络	Ctrl + Z	撤销先前的操作
Ctrl + Y	恢复成撤销前状态	Ctrl + F	查找
Ctrl + G	跳转网络	Ctrl + H	替换
Ctrl + B	切换书签	C	常开触点
/	常闭触点	O	梯形图中为线圈, 功能块中为“或 (OR) ”
P	上升沿触发器	N	下降沿触发器
U	取反 (梯形图)	S	置位 (梯形图)
R	复位 (梯形图)	A	与 (AND) 块 (功能块)
=	输出块 (功能块)	--	--

1.4 联机帮助

用户使用过程中, 如果不确定某一个指令的使用方法, 使用鼠标左键选中功能块, 然后点击 F1 按钮, 进入帮助界面, 查看指令的具体信息和使用方法。

PLC_Config 的“帮助”菜单提供选项如表 1.4.1 所示。

表 1.4.1 帮助菜单

菜单项	说明
快捷键说明	提供帮助文档中快捷键说明列表
PLC_Config 软件编程手册	提供 PLC_Config 软件编程手册
DCCE 网络化可编程控制器用户编程手册	提供 DCCE 网络化可编程控制器用户编程手册
联系我们	登陆 DCCE 网站首页
关于	显示软件基本信息

通过帮助菜单可访问《快捷键说明》、《PLC_Config 软件编程手册》、《DCCE 网络化可编程控制器用户编程手册》等说明文档。为了方便用户使用每个说明文档都提供了目录、索引和搜索栏。

- (1) 目录和索引栏允许用户借助目录浏览程序 (显示包含的标题) 或可搜索索引浏览该帮助系统;
- (2) 搜索栏允许用户搜索某一具体的变量、指令等。

2 编程

2.1 控制程序作业

控制器通过程序控制逻辑的不断循环，读取和写入数据，与外围设备进行数据交换；控制器支持多种指令，具有稳定高效的控制逻辑，可实现运动控制、过程控制和逻辑控制等；控制器定义了多个变量区，包括：I, Q, XI, XQ, V, M, S, SM, L, T, C, AI, PAI, XAI, AQ, PAQ, XAQ, P, EI, EQ, EAI, EAQ 和 E 区等，其中的 V 区、P 区和 SM 区支持掉电保持功能，可用于存储重要的数据信息。另外，控制器拥有足够的用户程序存储区，保证了用户程序的永久保存。变量的定义和指令说明详见《访问内存变量》和《可编程指令集》章节。

2.1.1 程序组成

程序是一组为完成某种功能而按一定顺序由算法确定编排的指令序列。如图 2.1.1 所示，控制程序包含主程序、子程序和中断程序 3 个部分，用户可以根据实际控制过程的需要选择所要使用的程序。

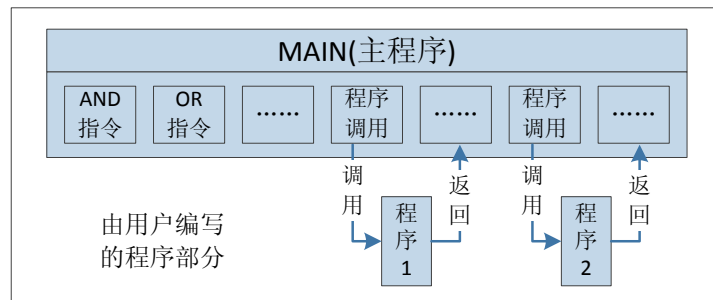


图 2.1.1 程序执行示意图

(1) 主程序

主程序是控制程序的基本程序，能够调用子程序，而不能被其他子程序所调用，是控制程序的中心部分。程序执行以周期扫描形式，从主程序的第一条指令开始顺序执行，循环往复。一个控制程序有且只有一个主程序。

(2) 子程序

子程序是用户程序的可选部分，只有当被调用的时候才能够执行。使用子程序可以将一段程序封装成功能独立的新的控制指令形式，增强了程序的重用性，有效地减少程序代码的冗余，并可以将很多功能独立程序应用于其他程序中，建立控制程序库。合理的使用子程序还可以减少控制程序的执行时间，提高运行速度。子程序由 CALL 指令调用，调用者可以是主程序或其他程序，编程软件 PLC_Config 将子程序直接显示成功能块格式，形式上相当于定义了新的功能块指令。子程序可嵌套调用，最多可嵌套 16 层。

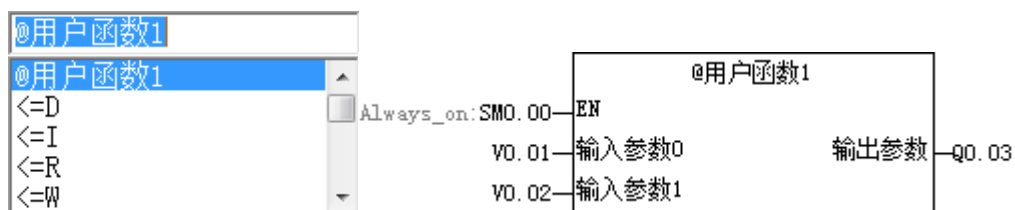


图 2.1.2 程序调用

(3) 中断程序

中断程序是用户程序的可选部分，只有当相应的中断事件发生时，中断程序才会被执行。中断程序由 ATCH 指令激活，与控制器内部的中断事件关联起来，由中断事件触发。中断事件到来时，控制器会打断当前正在执行的程序，转而执行中断程序，执行完成后再继续执行被打断的程序。

注意：中断程序有可能随时发生，故用户在中断程序中需要对使用的变量和主程序或其他程序中的变量进行仔细区分，防止变量使用重叠造成程序运行异常。相关的详细信息请参看第 6 章。

(4) 其他功能

除上述的程序组成部分外，控制器内的硬件或其它配置信息也是程序正常运行必不可少的部分，使用时

可根据实际情况进行选择配置。其中，程序可添加程序参数，无论主程序、子程序或中断程序均支持程序参数配置功能。

相关章节：程序参数

2.1.2 程序资源

不同控制器的程序存储区大小各不相同，其中 PEC/EDC/DEC/PMC/DIO/AIO 系列控制器，程序存储区容量不同对应的典型指令条数不同：用户程序存储区大小为 96K，可存储典型指令 3933 条；用户程序存储区大小为 256K，可存储典型指令 10488 条；用户程序存储区大小为 1M，可存储典型指令 41952 条；MAC 系列主控器的用户程序存储区大小为 2M，可存储典型指令 83904 条，具体程序存储区大小请参见各产品使用手册。

典型指令包括输入输出刷新指令，由于不同的指令占用的存储空间不同，典型存储指令以 AND 指令为例，其余资源如表 2.1.1 所示。

表 2.1.1 程序资源范围

名称	描述	大小
主程序	扫描周期中每次都运行的程序	1
子程序	扫描周期中可选运行的程序	1~128
中断程序	需要中断事件触发运行的程序	1~128
跳转/标号	用户程序中可跳转的标识	1~65535
PID 回路	支持的 PID 指令	0~15

2.1.3 程序执行逻辑

用户需要灵活运用程序，实现各种控制。下面从程序的运行机制，工作模式，存储加密三方面描述程序的控制逻辑。

(1) 运行机制

可编程控制器完成输入刷新、内部处理、程序执行、通信处理、输出刷新五个工作阶段，称为一个扫描周期。完成一次扫描后，又重新执行上述过程，这种周而复始的循环工作方式称为扫描工作方式。

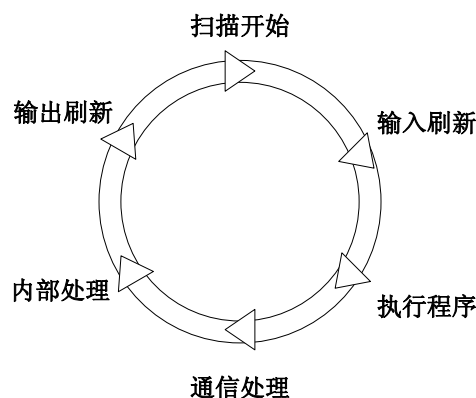


图 2.1.3 扫描周期

扫描周期中执行的任务依赖于控制器操作的模式，控制器有两种操作模式：停止模式、运行模式。停止模式和运行模式的主要区别是停止模式的时候不运行用户程序只执行 I/O 操作，而运行模式则运行控制程序且进行 I/O 操作。

● 输入刷新：

每次扫描周期开始时，先读取控制器物理输入到输入映像变量区（I 和 AI），然后把从设备输入值读取到本地输入映像变量区（XI 和 PAI）。

● 执行程序：

执行程序就是控制器从用户程序区读取指令，译码执行。从第一条指令开始，直到最后一条指令结束。在程序执行过程中，如果遇到立即输入输出指令则直接对物理量进行刷新，否则需要等到周期结束后进行输出刷新。

程序中调用的中断程序或子程序不会在扫描周期中自动执行，只有当中断事件触发或子程序调用时才被执行。

- 通信处理：

扫描周期内程序执行过程中，控制器定期进行通信处理，管理通信缓冲区收发报文，实现网络设备中共享变量区的刷新。

- 内部处理：

控制器每执行一条指令都对自身硬件、指令执行和通信状态进行检测，当发现问题时做出相应处理，并给出故障信息。

- 输出刷新：

在每个扫描周期的结尾，控制器会把更新后的输出映像变量区（Q 和 AQ），送给相应的物理输出，并把扩展设备的输出映像变量区的值（XQ 和 PAQ）通过相应通信操作发送给扩展设备。

控制器在 SM 区提供了程序执行时间信息，如表 2.1.2 所示。用户可根据需要进行查询。

表 2.1.2 程序运行扫描时间

SM 区	数据类型	功能说明	单位
SMD8	无符号四字节	上一次用户程序扫描执行时间	ms
SMD10	无符号四字节	用户程序扫描执行的最短时间	ms
SMD12	无符号四字节	用户程序扫描执行的最长时间	ms

(2) 工作模式

控制器具有两种工作模式：运行模式和停止模式，面板上设有相关的 LED 指示灯，特殊功能变量区 SM0.4 也能查询当前的运行模式。

运行模式下控制器自动运行用户已经下载到控制器中的程序，停止模式下控制器不执行程序，仅进行 I/O 刷新和通信处理。用户可以通过 PLC_Config 的  按钮来控制用户程序的运行或停止。SM0.4 也显示了用户程序的执行状态，1 为运行模式，0 为停止模式。

控制器支持程序单轮运行和多轮运行的调试功能，方便用户查找程序的错误或现场运行出现的问题。点击  按钮，所下载的运行程序仅运行 1 轮，并且运行程序的效果等同于设备刚刚上电运行第一轮程序的效果。点击  会提示输入需要调试的程序扫描轮数 n，程序会按照设置的扫描轮数进行程序扫描执行，并且运行程序的效果等同于设备刚刚上电运行 n 轮的效果。

(3) 程序存储和加密

用户可以利用编程软件上载和下载用户程序，用户程序保存在控制器内部单独的程序存储区中，可永久保存。

为保证用户程序的安全，控制器支持程序加密功能。在下载完程序后，用户只需要把特殊功能变量区 SM0.8 置位为 1，即可完成对用户程序的加密。加密后用户无法进行程序上载，但是可以下载新的程序，重新下载程序后加密功能自动解除，如果需要加密可再次置位 SM0.8。

警告：程序加密后用户将无法上载程序，如果用户需要上载程序，只能和公司技术人员联系！

2.2 项目元件及工作原理

PLC_Config 提供多种变量管理功能，如组态程序中的变量、状态符号表、交叉引用表、趋势图、只读变量区管理表等，利用这些功能可以实现对变量的操作、监控和管理。

在程序中使用符号，可使程序更易于理解或更改，符号分为全局符号和程序参数两种。全局符号在整个程序中均有效，该类型的符号在状态符号表中定义。无论状态符号表中的符号或是程序参数均需要符合以下要求。

- 1) 只能包含字母、汉字、下划线和数字；
- 2) 只能以字母、汉字或下划线开头；
- 3) 不能与绝对地址格式冲突，如无法定义全局符号为 VW3，或 TEMP 等绝对地址。

全局符号对于当前设备的所有程序均有效，状态符号表提供了全局符号的定义和配置功能。此外，软件

还提供了交叉引用表，通过交叉引用表可以收集并分析出程序中所有寄存器变量的使用情况，方便及时掌握程序状态，避免地址使用冲突，并且能够有效的提高程序运行效率。软件还提供了趋势图功能，可以在一定时间内监控变量的变化趋势，便于及时的跟踪程序运行状况。同时，只读变量区具有控制器掉电后保持数据的功能，您可以用此变量区保存配方数据等重要数据信息。

2.2.1 状态符号表

状态符号表提供变量的集中显示与管理功能，可在工程树的控制器结点下的“状态符号表”结点下查看到属于该控制器的所有状态符号表信息，每张表中均可以定义若干符号变量，如图 2.2.1 状态符号表所示。

序号	选择	符号	绝对地址	Modbus寄存器	数据类型	当前值(10进)	当前值(16进)	新数值	参数说明
1	☐	上电延迟启动时间	SMW212	1252	有符号双字节	10	A		
2	☐	内部程序版本号	SMW124	1164	有符号双字节	20482	5002		
3	☐	秒脉冲	SMW6	1046	有符号双字节	585	249		每秒加1
4	☐								
5	☒	进板数量	MW0	2848	有符号双字节	0	0	12	可配置
6	☐	进板使能	V0.0	16768	位变量	1	1		

图 2.2.1 状态符号表

状态符号表的各字段说明如表 2.2.1 状态符号表各字段的说明所示。其中除了当前值为只读项，其他项都可读可写，用户可根据需要添加一个新的符号变量，对其进行读写监控，也可修改已有符号变量的各种信息。此外，软件还为各种符号变量提供了多种编辑功能。

表 2.2.1 状态符号表各字段的说明

字段名	说明
序号	变量的索引顺序
选择	通过选择该变量来进行单独的写入操作
符号	定义的变量名称，在编程时可使用该符号名
绝对地址	对应设备的寄存器地址、区绝对地址 + 变量在区内的偏移地址组成，例如：I0.00，QW1 等
Modbus 地址	所使用变量可通过 Modbus 协议寻址的寄存器地址
数据类型	包含位变量、有符号双字节、无符号双字节、有符号四字节、无符号四字节、四字节浮点数共 6 种
当前值	启动监控时显示变量的当前数值，分 10 进制和 16 进制显示
新数值	在新数值区域中写入变量值后，可通过写入功能将其写入到设备中
参数说明	变量的注释，可自由填写

● 添加、修改

用户可选择任意一行，添加一个新的符号变量及其对应属性信息，也可修改已有的符号变量及其对应属性信息，需要注意符号名称不可重复，如果重复软件会及时通知。

● 多重值

用户可选中某行中的任意一列，右键选择“多重值”，便可快速定义多个地址连续、名称连续的符号变量。

● 添加 Modbus 变量

若只知道变量的 Modbus（寄存器）地址，用户可选中某行中的任意一列，右键选择“增加 ModBus 变量”，弹出 ModBus 地址设置对话框，如图 2.2.2 所示。软件会根据输入的 Modbus 地址反算出变量的绝对地址，并将其添加到表格中。

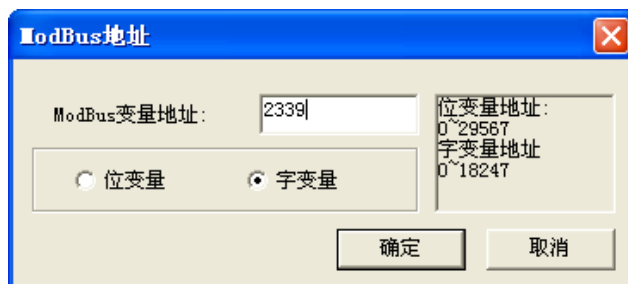


图 2.2.2 添加 ModBus 变量

图 2.2.3 添加 Modbus 变量效果

● 导入导出

软件提供变量表的导出和导入功能，可导出成 Excel 格式、Txt 格式。同时也支持将变量值按自定义的格式导出成文本格式。软件支持将西门子 STEP6-200 程序中的符号导入到状态符号表中，仅需要在西门子 STEP6-200 的符号表中选中所需的变量，使用 Ctrl+C 复制，再到 PLC_Config 的状态符号表中使用 Ctrl+V 粘贴。

● 监视功能

点击工具栏中的  按钮或“调试→监视”菜单项，PLC_Config 进入状态符号表监控状态。如果在与控制器通信过程中发生错误，则监控过程自动停止。

➤ 读取变量当前值

当状态符号表处于监控状态时，软件会定时与控制器通信读取变量的当前值，时间间隔为 250 毫秒。读取的变量内容仅为当前状态符号表所包含的变量。

➤ 写入全部

点击工具栏中的  按钮或“调试→全部写入”菜单项，软件会将当前状态符号表中用户所有设定的“新数值”写入到控制器的对应变量的中。写入完成或失败都会提示用户。

➤ 写入当前选中

点击工具栏中的  按钮或“调试→选中项写入”菜单项，软件会将当前状态符号表中用户选中项设定的“新数值”写入到控制器的对应变量的中。选中项即为图 2.2.4 中红色框所示。

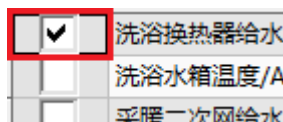


图 2.2.4 选中项例图

2.2.2 交叉引用表

PLC_Config 支持交叉引用表功能，该功能可以列出在组态程序中使用变量的情况，能够随时查看变量的引用情况。软件提供位应用、字应用和变量引用表三种引用表格式。程序编译完毕后，在工程树中打开交叉引用表，可选择查看字应用、位应用和变量引用表。

位应用和字应用表分别以位和字为单位显示变量地址的引用情况，如图 2.2.5 所示。

按位编址	15	14	13	12	11	10	9	8	7	6	5	4
V158.0	B	B	B	B	B	B	B	B	B	B	B	B
V159.0	B	B	B	B	B	B	B	B	B	B	B	B
M6.0	B	B	B	B	B	B	B	B	B	B	B	B
SMD.0												

按字编址	9	8	7	6	5	4	3	2	1
VW0									D
VW150	D	D							
MW0				W					
TO									T

图 2.2.5 位应用和字应用表

其中在字应用表中，“B”表示该地址被位变量使用，“W”表示该地址被字变量使用，“D”表示该地址被双字变量使用，“T”和“C”表示定时器和计数器地址的使用。双击某被显示字符的单元格时，软件会在下方的输出栏中显示出引用该地址的所有变量在程序中的具体位置，双击输出栏中的信息便可直接定位到程序的具体位置中，通过此功能可快速找到程序中的具体位置，一旦发现变量使用冲突，可及时予以纠正。

除位和字应用外，软件还支持变量引用表，双击打开变量引用表，可以查看到工程中每个变量的变量符号，及变量引用地址的所在程序、程序类型、网络号、行号、列号、指令名称及 IO 类型，如图 2.2.6 所示。

地址	符号	程序	程序类型	网络	行	列	指令名称	IO类型
IO.00	换热变频手/自/I	SUB_4.LAD	梯形图	1	0	0	- -	输入
IO.01	换热变频启动I	SUB_4.LAD	梯形图	1	0	1	- -	输入
IO.02	换热变频停止/I	SUB_4.LAD	梯形图	1	0	2	- -	输入
IO.03	手投运动0-1#/1-2#/I	SUB_4.LAD	梯形图	1	0	3	- -	输入
IO.03	手投运动0-1#/1-2#/I	SUB_4.LAD	梯形图	1	0	4	- /-	输入
IO.03	手投运动0-1#/1-2#/I	SUB_4.LAD	梯形图	1	0	5	- -	输入

图 2.2.6 变量引用表

变量引用表同样支持双击定位功能，双击某行的任一单元格，便可直接定位到该行变量所提示的程序处。

地址	符号	程序	程序类型	网络	行
VD100	电压码值转换	SUB_6.FBD	功能块	4	3
VD100	电压码值转换	SUB_3.FBD	功能块	1	3
VW100	移动停止位	SUB_5.FBD	功能块	1	3
VW101	输出位移	SUB_5.FBD	功能块	1	4
VD102	采集电压值	SUB_1.FBD	功能块	1	3
VD102	采集电压值	SUB_1.FBD	功能块	1	2

00码值，因此2000码/V，在当前采集的码值除以2000便是当前的电

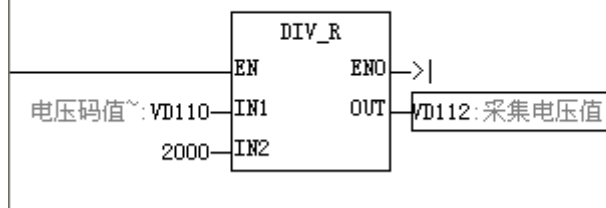


图 2.2.7 双击跳转定位变量

2.2.3 内存表管理

内存表可以对设备的变量区数据进行监控，将变量区内的数据通过表格以不同的形式显示出来，方便在不同的模式下进行观察。在内存表中可以根据需要修改变量值并写入到设备中；也可以将变量值导出或导入，或者将内存表中的值清空。内存表提供多种显示模式，可以随意进行切换，如表 2.2.2 所示。

表 2.2.2 显示模式说明

显示模式	说明
位变量	以位为单位显示，位的值只有 0 和 1 两种
无符号十进制	以无符号十进制数为单位显示
有符号十进制	以有符号十进制数为单位显示
浮点	以四字节浮点数为单位显示
十六进制	以十六进制数为单位显示
文本	显示文本格式，每个单元格的单位为字，显示 ASCII 码格式的文本，支持中文
双字	默认上显示方式（除浮点和文本外）均以字为单位，选择此项后将以双字为单位。字的长度为 16 位，双字长度为 32 位。
二进制显示转换	当以二进制显示时，选择此项后，显示值为 1 的单元格将背景色变为浅绿色，否则为白色。

2.2.4 只读变量区管理表

对于支持只读变量区（永久保持区：P 区）的控制器，PLC_Config 提供了只读变量区管理表，方便用户按指定类型管理地址连续的只读变量。只读变量具有掉电保持功能，您可以用此变量区保存配方数据。

PLC_Config 在工程树中显示只读变量区管理表结点，用户可以使用右键弹出菜单添加一个只读变量区管理表。也可以使用“编辑→只读变量区管理表→添加”菜单添加只读变量管理表。

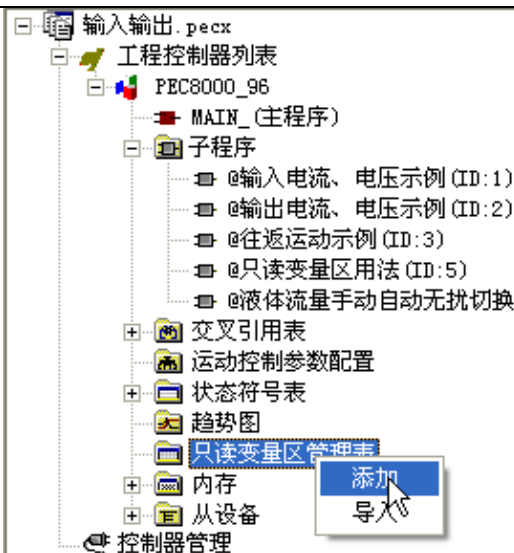


图 2.2.8 添加只读变量区管理表

添加管理表后需要点击  按钮或使用“编辑→只读变量区管理表→设置只读变量区”菜单来设置只读变量。弹出设置对话框如图 2.2.9 所示。

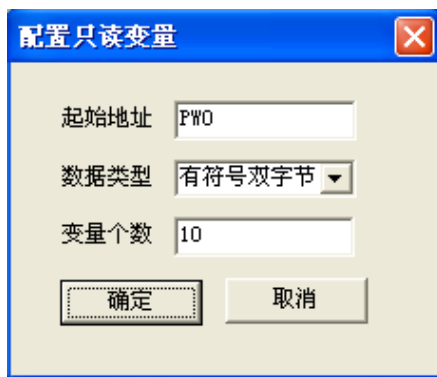


图 2.2.9 设置只读变量区

设置只读变量区的起始地址，数据类型和查看范围。点击确定后，根据设置填充管理表，如图 2.2.10 所示。

序号	选择	绝对地址	Modbus变量地址	数据类型
1	☐	PW0	0x3748 (14152)	有符号双字节
2	☒	PW1	0x3749 (14153)	有符号双字节
3	☒	PW2	0x374A (14154)	有符号双字节
4	☐	PW3	0x374B (14155)	有符号双字节
5	☐	PW4	0x374C (14156)	有符号双字节
6	☐	PW5	0x374D (14157)	有符号双字节
7	☐	PW6	0x374E (14158)	有符号双字节
8	☐	PW7	0x374F (14159)	有符号双字节
9	☐	PW8	0x3750 (14160)	有符号双字节
10	☐	PW9	0x3751 (14161)	有符号双字节

图 2.2.10 填充只读变量区管理表

只读变量数据的读取、写入、文本的导入导出功能均与符号状态表的使用方法一致。

2.2.5 趋势图

趋势图用于监控变量实时值，可以在一定时间内监控变量的变化趋势，便于及时地跟踪程序运行状况。在工程树的发展趋势图节点处的右键菜单中选择“添加（A）”，在趋势图界面的左下角添加需要被监控的变量，并配置监视时变量值显示的样式，通过点击“开始”按钮，对变量进行监控。

在监控时，趋势图部分显示出变量值的变化曲线，可以实时地修改所监控变量的当前值。此外，软件还

为趋势图提供了多种编辑功能。

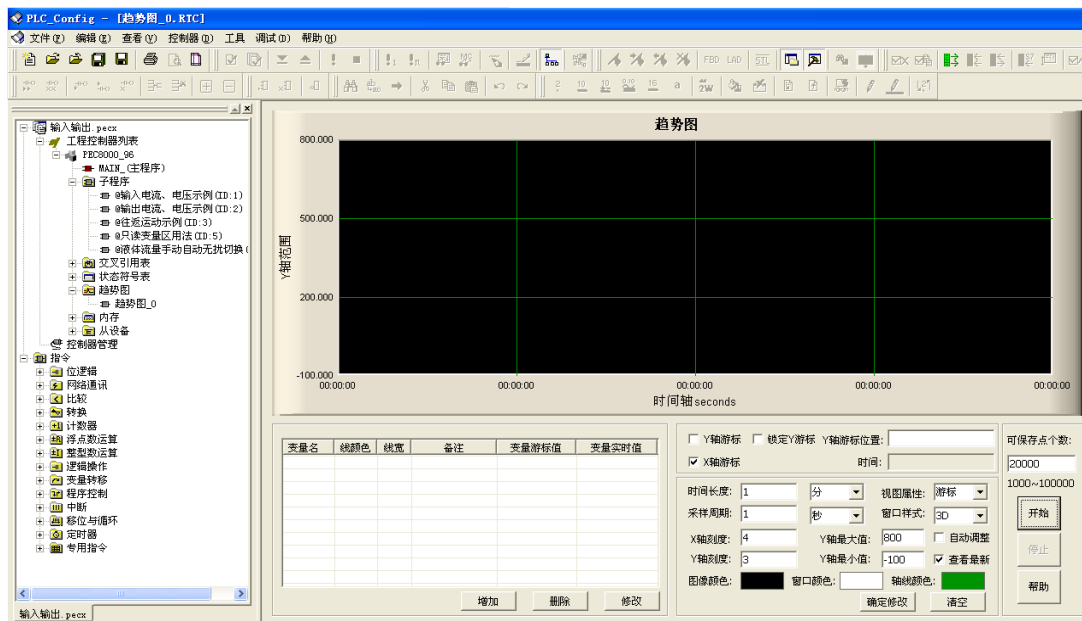


图 2.2.11 趋势图

● 趋势图的添加和删除

在工程树控制器结点下的“趋势图”上点击右键选择“添加”，或通过“编辑→趋势图→添加”菜单都可以添加趋势图。添加后会弹出一个“添加对话框”，通过该中为新增的趋势图填写文件名称。

在某个具体趋势图上点击右键选择“删除”即可删除选中的趋势图，选择“重命名”可以修改趋势图名称。

● 趋势图变量的导出、增删改

趋势图提供导出为 Excel 文件功能。通过  按钮导出文件，把图像上所有的点保存到 Excel 文件中。点击 **增加** 按钮，弹出添加变量的窗口，可以配置变量的属性。右键菜单提供修改和删除功能。每张趋势图最多可以添加并监控 8 个变量。

● 变量实时值监控和修改

在监控时，用户可以实时地修改所监控变量的当前值，在变量表格中右键该变量，选择“变量写值”菜单项即可实现写入操作。

● 趋势图控制面板

趋势图的右下方提供了一个控制面板，通过该控制面板可配置趋势图的时间长度，采样周期，视图属性，窗口样式，X、Y 轴刻度，Y 轴最大、最小值等，详细说明如表 2.2.3 所示。

表 2.2.3 趋势图控制面板说明

标题	功能
时间长度	表示默认情况下趋势图曲线显示区域的宽度
采样周期	每隔多长时间读取一次变量值
视图属性	曲线操作功能，可选择一下几种功能 游标：点击曲线会出现 X 或 Y 轴的游标，变量表显示变量所在游标的值 放大：框选一段曲线后放开鼠标，图像会将框选区域放大显示 XY 轴拖动：可将曲线上下左右拖动 X 轴拖动：可将曲线沿 X 轴拖动 Y 轴拖动：可将曲线沿 Y 轴拖动
窗口样式	显示曲线窗口的显示效果
X、Y 轴刻度	表示 X 和 Y 轴刻度线的数量
Y 轴最大、最小值	Y 轴的默认显示范围
自动调整	选择该项可使图像自动调整到合适的范围显示图像
查看最新	选择该项后图像的最右端将一直为当前时间

2.2.6 程序参数

在程序编辑窗口中为每个程序都提供了一个程序参数配置区，通过程序开头的程序参数配置区可为该程序添加输入、输出变量和内部变量（局部变量），这些变量统称为程序参数。

PLC_Config 中所有的程序，包括主程序、子程序和中断程序，都支持程序参数配置功能，其中主程序和中断程序只支持内部变量配置，子程序除了支持内部变量配置外，还支持输入、输出变量配置功能。

程序参数不需要编写参数的实际地址（即对应设备的变量地址），PLC_Config 软件在编译时会自动为程序参数编址，保证地址使用不重叠。图 2.2.12 中共添加了 5 个程序参数，包括 2 个输入参数、1 个输出参数和 2 个内部变量，添加程序参数时，需要定义参数名称，选择输入输出类型和数据类型。此外，也可为参数添加注释信息。

ID	名称	输入输出	数值类型	注释
0	输入参数0	输入	位变量	
1	输入参数1	输入	位变量	
2	输出参数0	输出	位变量	
3	i	内部变量	无符号双字节	
4	运算结果	内部变量	有符号双字节	
+ 添加参数				

图 2.2.12 程序参数配置示意图

(1) 利用程序参数进行编程

在编程时，可利用程序参数作为指令的输入输出引脚。为指令引脚填写参数时，会弹出指令参数编辑对话框，如图 2.2.13 所示。



图 2.2.13 指令参数编辑对话框

在参数编辑对话框中，可以输入符号、地址，也可以选择变量类型。其中符号一栏为下拉列表，参数名称可以直接输入有实际地址的变量区，也可以在下拉列表中选择已定义的程序参数或全局符号。下拉列表中显示的所有符号均为符合该类型的符号集合，是软件统计了程序参数和全局符号之后整理出来的。通过键盘输入对符号进行实时筛选，直到找到所需参数变量为止。输入完成后，点击回车键或“确定”按钮后完成输入。

(2) 程序的调用与传参

可通过添加程序功能块的方式调用子程序。添加程序功能块与添加普通功能块类似，在指令网络中按下“Enter”键，在弹出的指令列表中找到并选择所需要的指令，可将其添加到程序中，也可以在工程树的程序结点下找到相应的结点，然后通过右键菜单的“插入块”添加程序功能块，如图 2.2.14 所示。

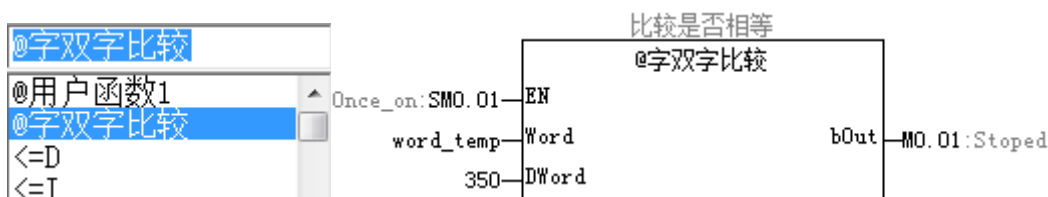


图 2.2.14 子程序的使用

子程序可添加输入、输出等类型的参数，这些参数在程序内部统称为“形参”，在调用程序时，外部需要

为其传递“实参”，实参可以是地址、其他变量或立即数。

2.3 编写程序

PLC_Config 软件支持两种编程语言，分为梯形图和功能块，此外，梯形图和功能块都可以转换为指令码显示。软件支持梯形图、功能块和指令码三种语言间的相互转换，转换时以单个网络为单位，如果某个网络转换失败，将显示成指令表格式，并通过网络注释提示转换失败的原因，具体的情况和相应的解决方法可参见随软件安装附带的《PLC_Config 使用说明》。

指令是组成程序的基本单位，用梯形图语言和功能块语言编写的程序都是由不同的指令组合而成，PLC_Config 软件提供 200 多条指令，其中包括位逻辑、整型运算、浮点运算、比较、移位、转换、循环、定时器、计数器、网络通讯和运动控制等，关于指令集的详细说明可参见指令章节。

2.3.1 编写梯形图程序

梯形图是使用最广泛的 PLC 图形编程语言。梯形图与继电器的电路很相似，具有直观易懂的优点，很容易被工厂熟悉继电器控制的电气人员掌握，特别适合数字量逻辑控制。

梯形图由触点、线圈和方框表示的指令组成。触点代表逻辑输入条件，例如外部的开关、按钮和内部条件等。线圈通常代表逻辑运算的结果，常用来控制外部的负载和内部标志位等。方框指令用来表示定时器、计数器或者数学运算等附加指令，如图 2.3.1 所示。

梯形图编程格式如下：

(1) 梯形图按行从上至下编写，每一行从左往右顺序编写。PLC 程序执行顺序与梯形图的编写顺序一致；

(2) 图左、右边垂直线称为起始母线、终止母线。每一逻辑行必须从起始母线开始画起，终止于继电器线圈或终止母线（有些 PLC 终止母线可以省略）；

(3) 梯形图的起始母线与线圈之间一定要有触点，而线圈与终止母线之间则不能有任何触点。

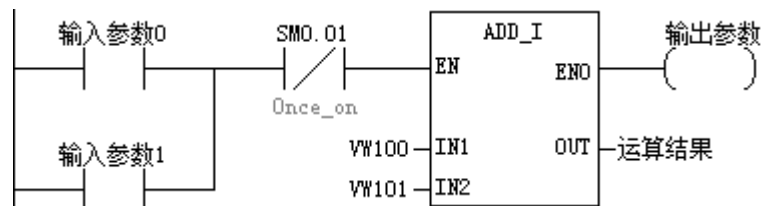


图 2.3.1 梯形图编程语言

其中符号意义如下：

- | |— **触点**，代表逻辑输入条件，如开关、按钮、内部条件等。
- () **线圈**，通常代表逻辑输出结果，如灯、马达启动器、内部输出条件等。
- **块指令**，具有一定算法能力的处理单元，例如定时器、计数器、数学运算指令等。

2.3.1.1 建立梯形图程序工程

(1) 新建工程

双击 PLC_Config 图标  打开 PLC_Config 软件，在编程软件界面中，通过“文件→新建工程”或点击  按钮来创建工程，在新建工程对话框中添加工程名称，路径，也可根据需要选择添加工程备注。

(2) 打开工程

通过“文件→打开工程”或点击  按钮来打开工程，选择工程文件后，点击“打开”按钮；或双击工程文件  “.pecx”，直接进入工程。

(3) 添加设备

当 PLC_Config 中有一个工程后，用户可以进行以下操作：

- 在工程树中的“设备管理”选项中选择需要添加的工程设备，双击该设备，将其添加到工程中；
- 单击设备列表中的设备，选中从设备列表，为该设备配置从设备；

通过“添加设备”选项为工程添加离线设备，当设备被添加到工程中后，用户就可以对该设备进行编程了。

2.3.1.2 LAD 中建立网络

在 LAD 中建立网络时，需要遵循一定的编程原则：

- (1) 放置触点的规则：每个网络必须以一个触点开始；网络不能以触点终止；
- (2) 放置线圈的规则：网络不能以线圈开始；线圈用于终止逻辑网络；
- (3) 放置方框的规则：如果方框有 ENO，功率流扩充至方框外；这意味着可以在方框后放置更多的指令。在网络的同级线路中，可以串联若干个含有 ENO 的方框。如果方框没有 ENO，则不能在其后放置任何指令；
- (4) 网络尺寸限制：用户可以将程序编辑器窗口视作划分为单元格的网格（单元格是可放置指令、为参数指定数值或绘制线段的区域）。在网格中，一个单独的网络最多能垂直扩充 32 个单元格或水平扩充 32 个单元。

2.3.1.3 LAD 中输入指令

打开工程后，软件左侧工程树的下方会显示出指令树，即为当前控制器当前编程语言所支持的所有编程指令。



图 2.3.2 指令树图例

(1) 添加指令

➤ 通过指令树添加指令

在网络内左键单击空白位置，然后展开指令组，双击想要添加的指令，即可将其添加到用户指定的位置上。

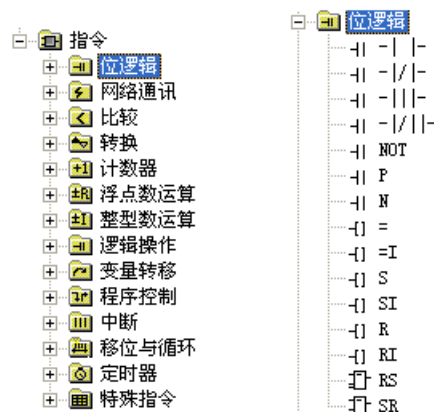


图 2.3.3 指令图例

➤ 键盘键添加指令

在网络内左键单击空白位置，点击“Enter”键直接插入新指令。



图 2.3.4 键盘添加指令图例

➤ 右键菜单添加指令

在需要添加指令的程序段或网络中右键，在弹出的菜单中选择“插入指令”，找到需要的指令，按“Enter”键即可。

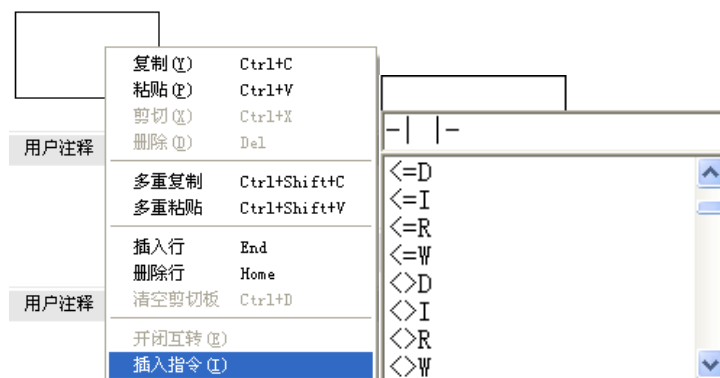


图 2.3.5 右键菜单插入指令图例

➤ 常用指令

在编辑区的右侧，PLC_Config 为用户提供了常用的指令快捷键，方便用户使用。

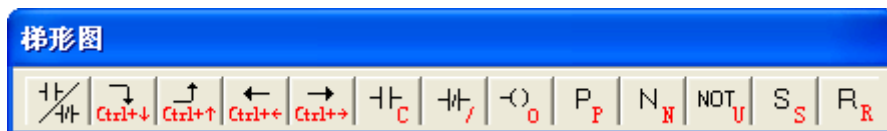


图 2.3.6 梯形图常用指令快捷键

表 2.3.1 梯形图常用指令快捷键表

指令	说明	指令	说明
	按“C”键直接添加常开指令		按“R”键直接添加复位命令
	按“/”键直接添加常闭指令		按“Ctrl+↑”键直接添加连接线
	按“O”键直接添加输出指令		按“Ctrl+↓”键直接添加连接线
	按“P”键直接添加上升沿触发器		按“Ctrl+←”键直接添加连接线
	按“N”键直接添加下降沿触发器		按“Ctrl+→”键直接添加连接线
	按“U”键直接添加取反指令		常开、常闭开关转换
	按“S”键直接添加置位命令	--	--

(2) 移动指令

选中一个或多个指令，在选中的指令上按住鼠标左键，通过拖拽操作将选中的指令移动到目标位置。在移动指令时，需要注意：

- 指令不能跨网络移动，若要跨网络移动需要使用粘贴功能；
- 指令不能移动到网络编辑区以外的部分；

- 不能跨越连接关系进行移动；
- 指令可以与连接关系重合，但不推荐使用此种方式；
- 指令和指令不能重合。

梯形图指令在移动时，指令本身不移动，只显示一个表示移动位置的黄色矩形框。松开左键后，指令会移动到该位置。在鼠标左键选中指令前，按住“Ctrl”键表示复制该指令，但不复制参数。

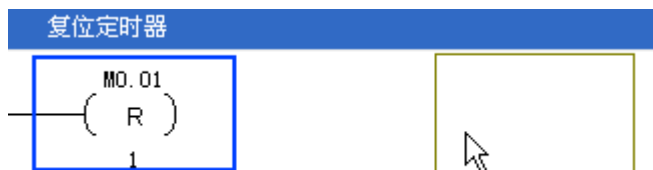


图 2.3.7 指令的移动图例

软件提供指令、网络的复制粘贴功能，并且可以在两个 PLC_Config 进程之间进行复制粘贴。

(3) 复制粘贴指令

复制快捷键：Ctrl+C

粘贴快捷键：Ctrl+V

(4) 指令多重复制

PLC_Config 支持多重复制功能，可快速地添加多个变量地址连续的相同指令，地址的连续规则可自由配置。

多重复制快捷键：Ctrl+Shift+C

多重粘贴快捷键：Ctrl+Shift+V

多重复制分为如下几步：

- 多重复制调用

在网络中，选中需要多重复制的指令后单击鼠标右键，在右键菜单中选择“多重复制”选项，弹出多重复制对话框。

多重复制也可以通过快捷键来配置，选中需要多重复制的指令，同时按下 Ctrl+Shift+ C 键，弹出多重复制对话框。

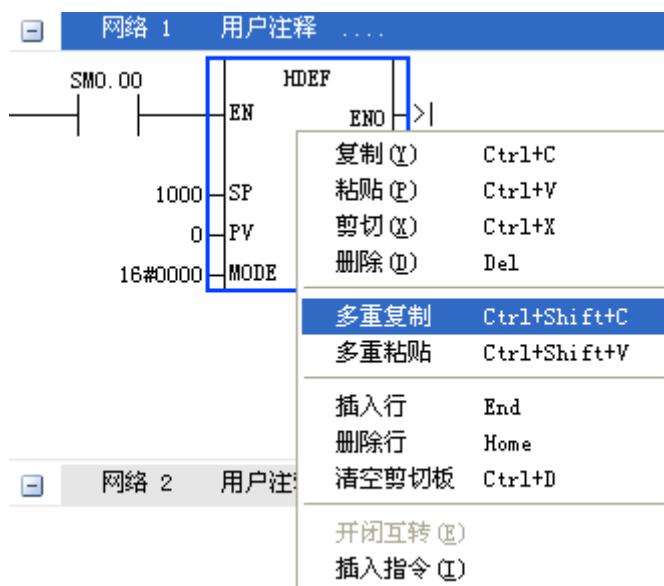


图 2.3.8 指令的多重复制图例

- 配置多重复制信息

在多重复制对话框中配置多重复制信息，其中，只有变量偏移量列是可以修改的，功能块信息、引脚信息、变量地址与类型列均为只读部分。

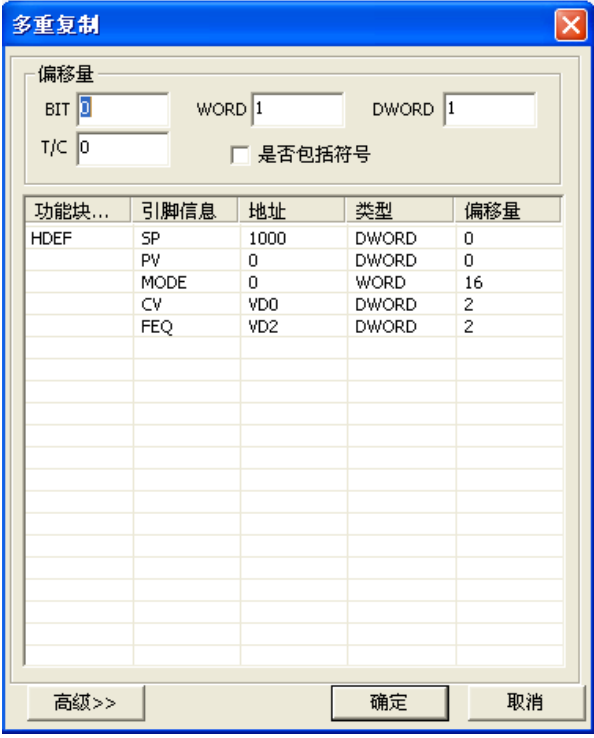


图 2.3.9 多重复制对话框

表 2.3.2 偏移量说明

偏移量名称	说明
Bit	位变量，可统一修改参数类型为 bit 的位变量参数
Word	字变量，可统一修改参数类型为 word 的字变量参数
Dword	双字变量，可统一修改参数类型为 dword 的字变量参数
T/C	T 区和 C 区的统一修改

当选择“高级”配置时，会在对话框中增加“符号名设置”栏，可以选择变量符号的复制增加方式。

注意：在选择“高级”配置时，需要在“是否包括符号”前打勾。

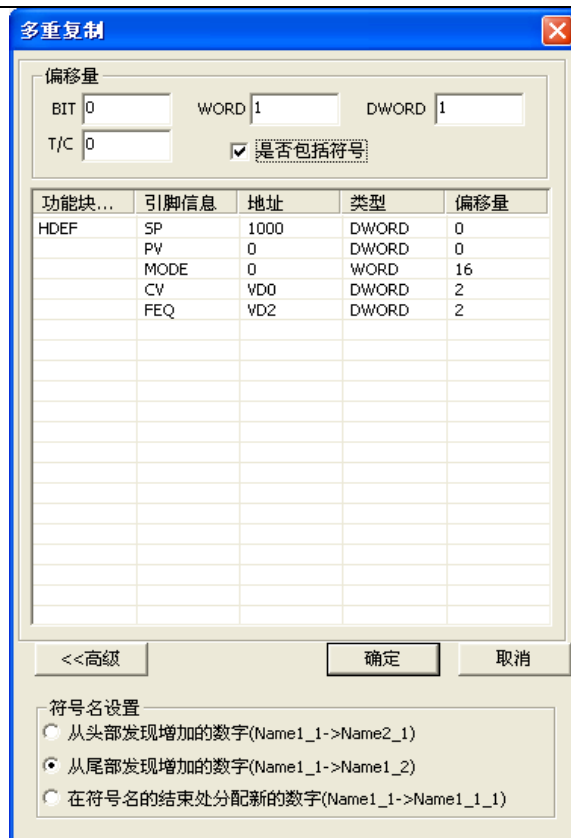


图 2.3.10 多重复制对话框

► 多重粘贴

指令被复制以后，用户可以在网络的空白处单击鼠标右键，选择“多重粘贴”选项，将指令粘贴到所需要的位置。粘贴时参数引脚也会按照配置的增量信息进行增加。使用快捷键 **Ctrl+Shift+ V** 也可以达到同样的目的，如图 2.3.11 所示。

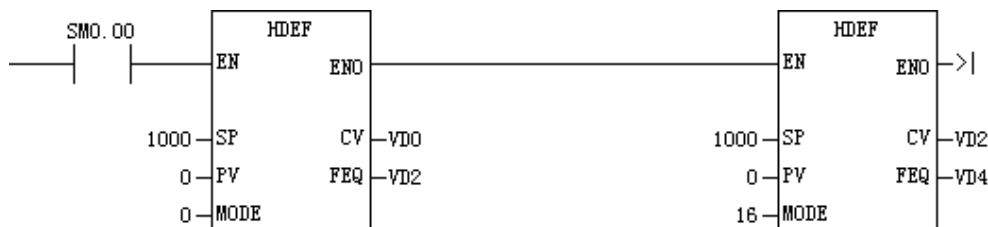


图 2.3.11 多重粘贴结果

多重复制支持对多个指令进行操作，过程与单个指令完全相同。

2.3.1.4 LAD 中输入地址

当在 LAD 中输入一条指令时，参数用问号表示，例如 (???) 或 (????)。问号表示参数未赋值。用户可以在输入元素时为该元素的参数指定一个常量、绝对值、符号、变量地址等。如果有任何参数未赋值，程序将不能正确编译。

欲指定一个常量数值（例如 100）或一个绝对地址（例如 I0.1），只需在指令地址区域中键入所需的数值。（用鼠标双击或 ENTER 键选择键入的地址区域。）



图 2.3.12 参数编辑器

当输入的地址异常时，会弹出错误提示对话框。输入异常情况有如下几种情况：

(1) 当地址变量的类型输入错误时，按下回车键或点击确定后会弹出提示对话框，提醒用户需要重新输入地址变量。

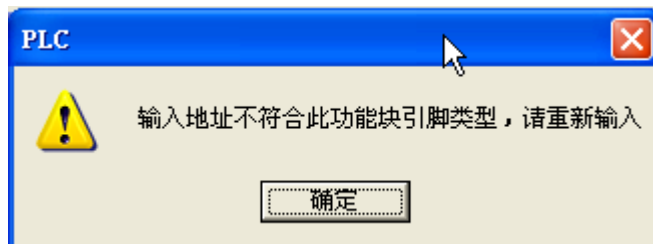


图 2.3.13 输入地址不符

(2) 在指令参数的编辑对话框中，如果用户在“符号”栏输入了符号名，生效后软件将自动记录该符号。

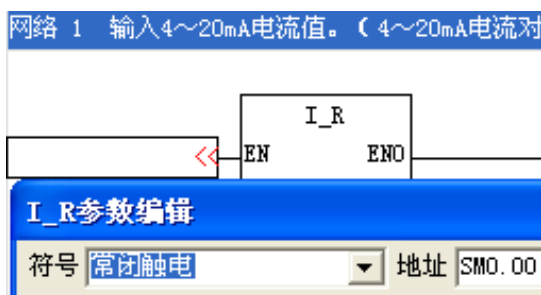


图 2.3.14 参数编辑器图例

打开符号状态表，可以查看该符号。

符号	绝对地址	Modbus变量地址	数据类型
常闭触点	SM0.00	0x1140 (4416)	位变量

图 2.3.15 状态符号表图例

(3) 当添加的变量符号已存在时，按下回车键或点击确定后会弹出警告对话框，提示该符号已经定义，并指出定义的位置，要求用户重新定义。在程序下方的编译信息中会输出符号定义的位置，双击即可找到该符号。

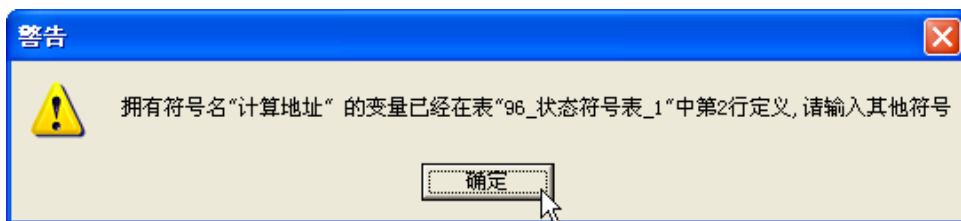


图 2.3.16 输入变量已存在

(4) 当输入的符号不符合变量的符号定义规则时，按下回车键或点击确定后会弹出警告对话框，提示错误原因。

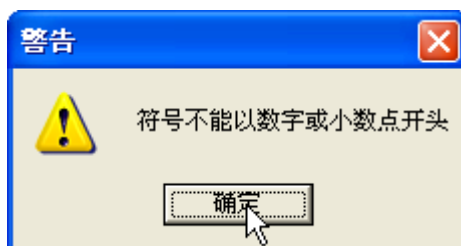


图 2.3.17 符号不符定义规则

(5) 当输入的变量地址不符合变量定义规则时，按下回车键或点击确定后弹出警告对话框，提示用户所输入的地址不合法。



图 2.3.18 输入地址不合法

(6) 当符号栏中输入地址变量时，按下回车键或点击确定后弹出警告对话框，提示用户符号名不合法。

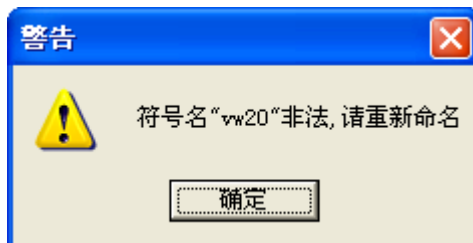


图 2.3.19 输入符号名非法

(7) 快速定义程序参数

在指令参数的编辑对话框中，如果用户在“地址”栏输入符号名，生效后如果软件认为该地址信息不符合该控制器的变量地址格式，则将其视为添加程序参数。



图 2.3.20 快速定义程序参数

2.3.1.5 LAD 中使用查找和替换功能

(1) 查找

若在程序编辑时需要查看变量的引用位置，可以通过系统主菜单“编辑->查找”，或键盘组合键“Ctrl+F”，或点击工具条中查找按钮 打开“查找”对话框，查找指令或变量。在查找对话框中输入要查找的内容，并选择查找参数的匹配项，点击  查找后，结果会在输出栏的查找选项中显示。

查找功能支持查找参数、搜索当前选中与端点查找功能。

➤ 查找参数

在打开的“查找参数”对话框中填写查找内容，在配置选项中选择查找参数的匹配项，在查找范围中选择查找该变量的范围，选择完成后点击查找按钮。

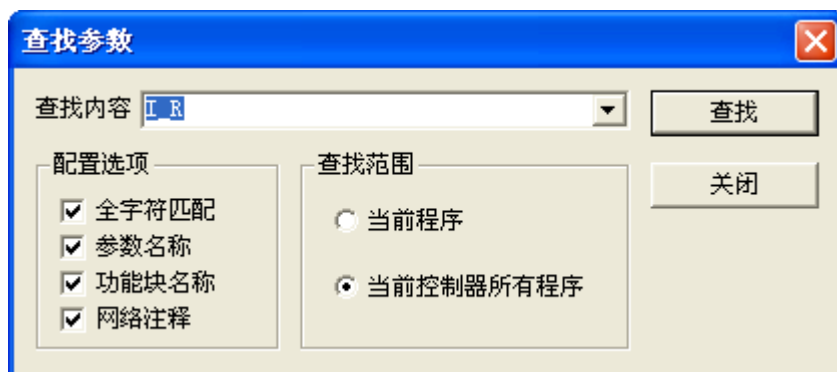


图 2.3.21 查找参数对话框

查找的结果将列在输出栏的查找选项中。

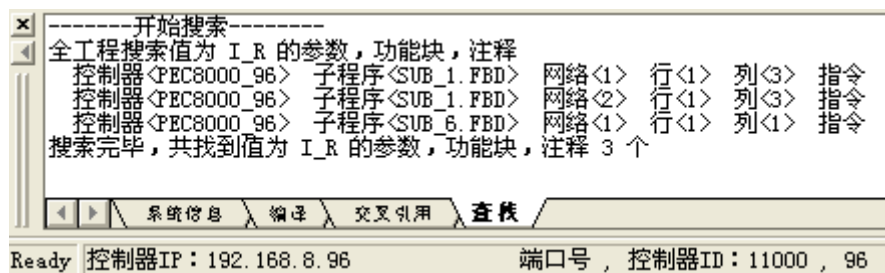


图 2.3.22 查找结果显示

➤ 查找端点

端点查找是指选中指令引脚。将鼠标放置在指令端点处，点击鼠标右键，在右键弹出菜单中选择查找该端点，在输出栏查找选项中会显示该变量引用的地址，指令名称，输入输出类型。

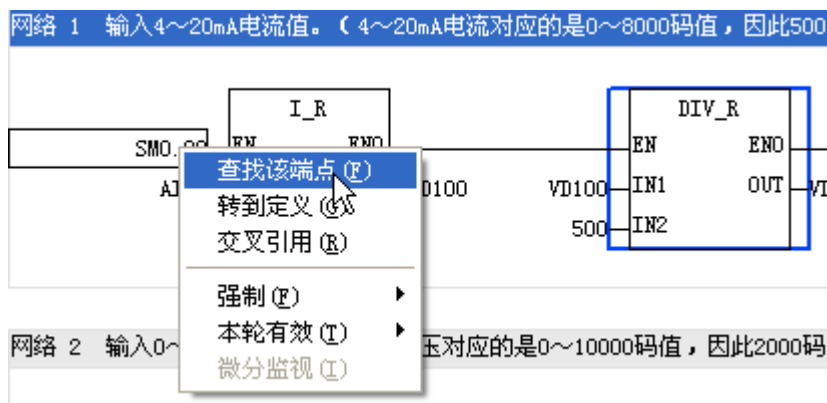


图 2.3.23 端点查找

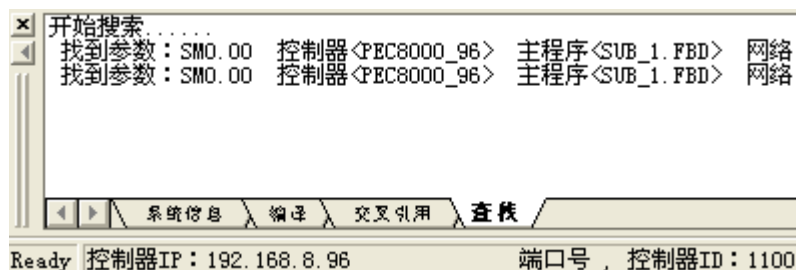


图 2.3.24 端点查找结果

端点查找不同于“查找参数”对话框查找，端点查找的查找范围为当前程序段，而“查找参数”对话框的查找范围是整个程序。

➤ 搜索当前选中

搜索当前选中，是指当选中指令或某一指令引脚变量时，按下“Ctrl+F”，在查找内容处会自动添加当前选中内容，点击“查找”按钮，在输出栏“查找”中列出相关变量的应用信息，能够方便用户查找功能块（梯形图）指令及其引脚变量的引用情况。

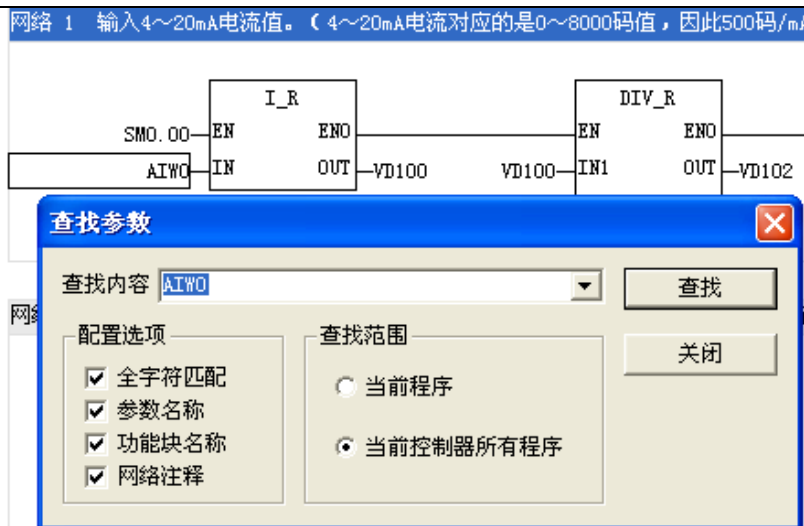


图 2.3.25 选定查找指令或参数图例

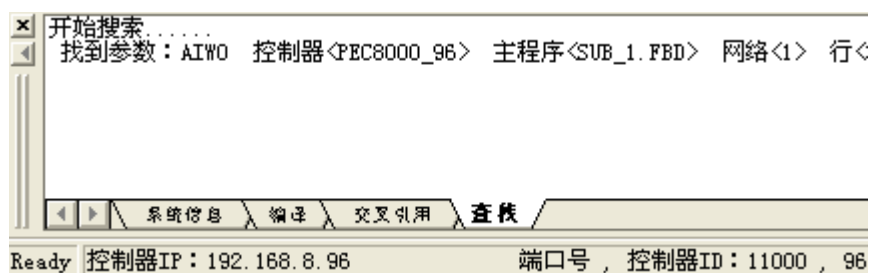


图 2.3.26 输出栏显示查找信息图例

在查找栏中，可以将工程中所有关于搜索变量（指令）的引用信息一次列出，双击其中的信息可定位到变量引用的梯形图引脚中。

(2) 替换

在程序编辑过程中，可以通过替换功能进行变量的替换。替换可分为单独变量替换和区间替换两个子功能。

➤ 单独变量替换功能

您可以通过以下三种方法进行替换操作：

- ①单击编辑工具条中替换  按钮
- ②使用快捷方式 Ctrl+H
- ③点击“编辑→替换”菜单

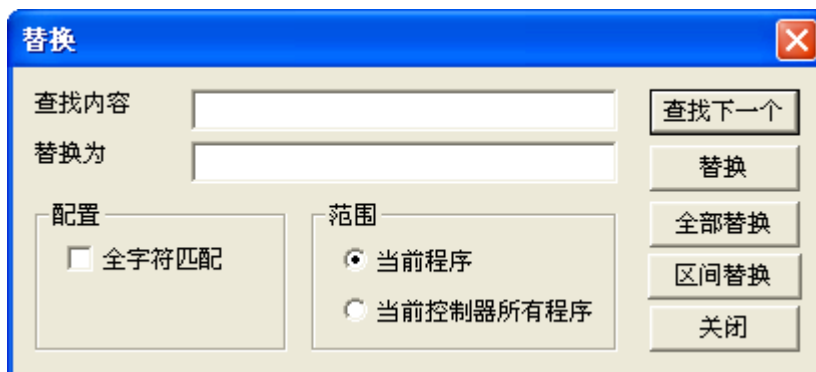


图 2.3.27 替换对话框

在查找替换对话框中填入您要查找、替换的内容，并选择替换范围，是否全字符匹配，然后点击功能按钮即可。

➤ 区间替换

在工程中有大量的连续变量需要修改成其他类型的变量时，可通过变量区间替换来完成，该功能可以为用户节省时间，避免在修改过程中出现遗漏变量的现象。具体步骤如下：

通过编辑菜单打开替换对话框，也可通过快捷键“Ctrl+H”打开替换对话框。

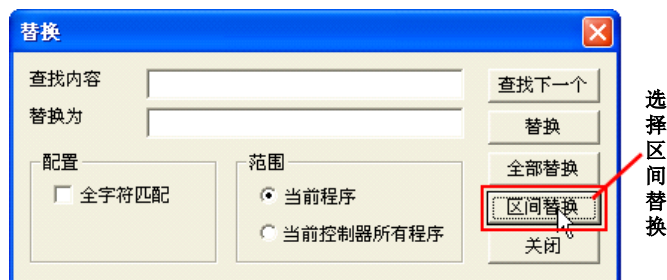


图 2.3.28 替换对话框图例

点击区间替换对话框，填写需要更改的变量区间，以及替换变量名称，在“范围”选项中选择替换变量是针对当前程序段还是对于当前控制器中的所有程序。

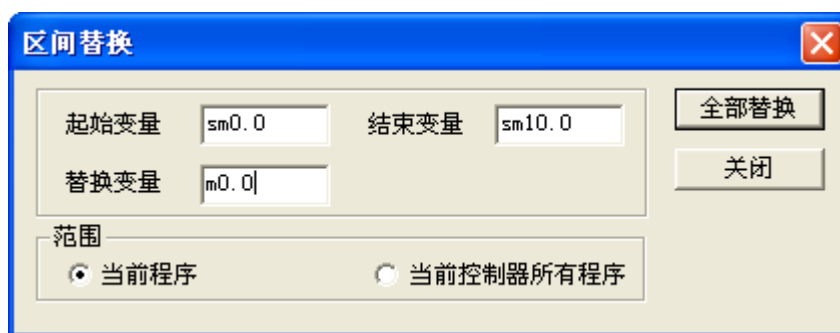


图 2.3.29 区间替换对话框

2.3.2 编写功能块图程序

功能块图用来描述功能块和程序的行为特征，还可以在顺序功能流程图中描述步骤、动作和转变的行为特征。功能块的使用类似与布尔代数的图形逻辑符号来表示控制逻辑，有数字电路基础的人很容易掌握。功能块用于类似于与门、或门的方框来表示逻辑运算关系，方框的左侧为逻辑运算的输入变量，右侧为输出的变量，输入输出的小圆圈表示“非”运算，方框被“导线”连接在一起，信号自左向右流动。

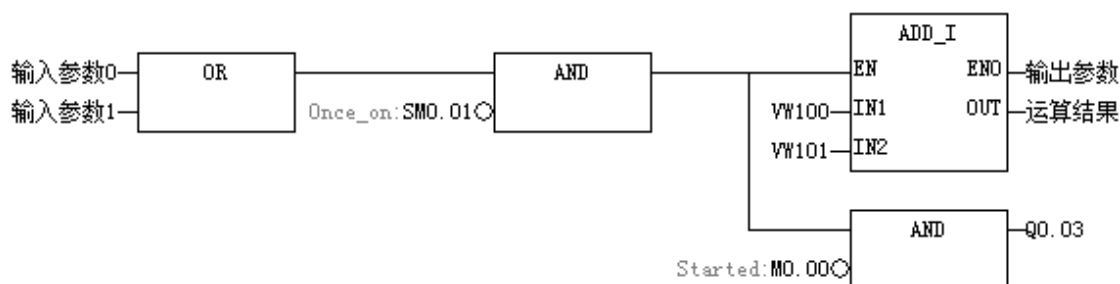


图 2.3.30 功能块编程语言

功能块用矩形块来表示，除特殊功能块外，每一个功能块的左侧应至少有一个输入端，在右侧应至少有一个输出端。功能块的类型名称通常写在块内的上部，输入输出参数的名称写在块内的输入输出点的相应地方。与梯形图编程语言的区别在于功能块是完全基于块指令的编程方式，功能块有自己的连线方式，不会像梯形图一样出现相交的情况。

2.3.2.1 建立功能块程序工程

(1) 新建工程

双击 PLC_Config 图标  打开 PLC_Config 软件，在编程软件界面中，通过“文件→新建工程”或点击  按钮来创建工程，在新建工程对话框中添加工程名称，路径，也可根据需要选择添加工程备注。

(2) 打开工程

通过“文件→打开工程”菜单或  按钮来打开工程，选择工程文件后，点击“打开”按钮。或双击工程文件  “.pecx”，直接进入工程。

(3) 添加设备

当 PLC_Config 中有一个工程后，用户可以进行以下操作：

- 在工程树中的“设备管理”选项中选择需要添加的工程设备，双击该设备，将其添加到工程中；
- 单击设备列表中的设备，选中从设备列表，为该设备配置从设备；
- 通过“添加设备”选项为工程添加离线设备，当设备被添加到工程中后，用户就可以对该设备进行编程了。

2.3.2.2 FBD 中建立网络

在 FBD 中建立网络时，需要遵循一定的编程原则：

- (1) 放置功能块的规则：放置功能块要按照功能块的执行顺序摆放，执行顺序为从左至右，从上至下。
- (2) 功能块连线的规则：功能块连线要按照功能块的执行顺序摆放，执行顺序为从左至右，从上至下。在一个网络中所有的功能块都需要用“导线”连接成一个独立的程序，在一个网络中不可以出现两个独立的程序。“导线”的连接不能出现交叉的情况。

(3) 网络尺寸限制：用户可以将程序编辑器窗口视作划分为单元格的网格（单元格是可放置指令、为参数指定数值或绘制线段的区域）。在网格中，一个单独的网络最多能垂直扩充 32 个单元格或水平扩充 32 个单元。

2.3.2.3 FBD 中输入指令



图 2.3.31 指令树图例

(1) 添加指令

- 通过指令树添加指令

在网络内左键单击空白位置，然后展开指令组，双击想要添加的指令，即可将其添加到用户指定的位置上。

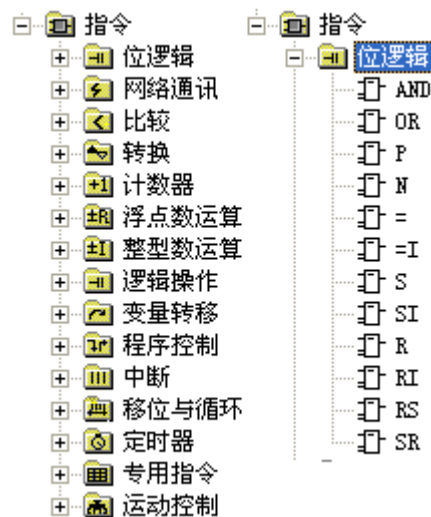


图 2.3.32 指令图例

➤ 键盘键添加指令

在网络内左键单击空白位置，点击“Enter”键直接插入新指令。

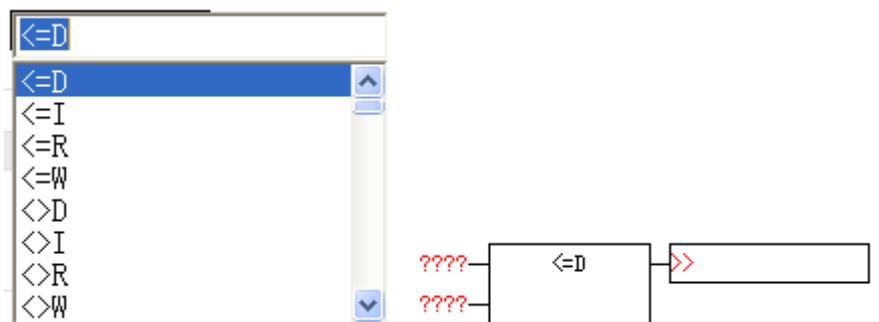


图 2.3.33 键盘添加指令图例

➤ 右键菜单添加指令

在需要添加指令的程序段或网络中右键，在弹出的菜单中选择“插入指令”，找到需要的指令，按“Enter”键即可。



图 2.3.34 右键菜单插入指令图例

➤ 常用指令

在编辑区的右侧，PLC_Config 为用户提供了常用的指令快捷键，方便用户使用。

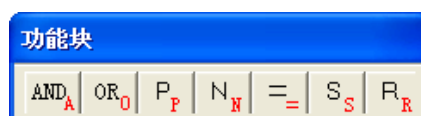


图 2.3.35 功能块常用指令快捷键

表 2.3.3 功能块常用指令快捷键表

指令	说明	指令	说明
AND _A	按“A”键直接添加 AND 指令	=	按“=”键直接添加=指令
OR _O	按“O”键直接添加 OR 指令	S _S	按“S”键直接添加 S 指令
P _P	按“P”键直接添加 P 指令	R _R	按“R”键直接添加 R 指令
N _N	按“N”键直接添加 N 指令	--	--

(2) 移动指令

选中一个或多个指令，在选中的指令上按住鼠标左键，通过拖拽操作将选中的指令移动到目标位置。

在移动指令时，需要注意：

- 指令不能跨网络移动，若要跨网络移动需要使用粘贴功能；
- 指令不能移动到网络编辑区以外的部分；
- 不能跨越连接关系进行移动；
- 指令可以与连接关系重合，但不推荐使用此种方式；
- 指令和指令不能重合。

梯形图指令在移动时，指令本身不移动，只显示一个表示移动位置的黄色矩形框。松开左键后，指令会移动到该位置。在鼠标左键选中指令前，按住“Ctrl”键表示复制该指令，但不复制参数。

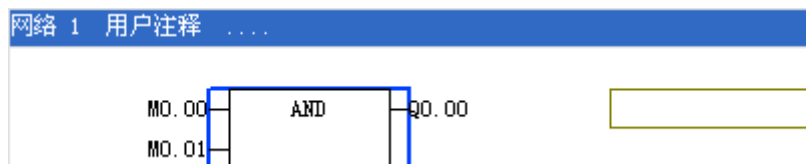


图 2.3.36 指令的移动图例

软件提供指令、网络的复制粘贴功能，并且可以在两个 PLC_Config 进程之间进行复制粘贴。

(3) 复制粘贴指令

复制快捷键：Ctrl+C

粘贴快捷键：Ctrl+V

(4) 指令多重复制

PLC_Config 支持多重复制功能，可快速地添加多个变量地址连续的相同指令，地址的连续规则可自由配置。

多重复制快捷键：Ctrl+Shift+C

多重粘贴快捷键：Ctrl+Shift+V

多重复制分为如下几步：

- 多重复制调用

在网络中，选中需要多重复制的指令后单击鼠标右键，在右键菜单中选择“多重复制”选项，弹出多重复制对话框。

多重复制也可以通过快捷键来配置，选中需要多重复制的指令，同时按下 Ctrl+Shift+ C 键，弹出多重复制对话框。

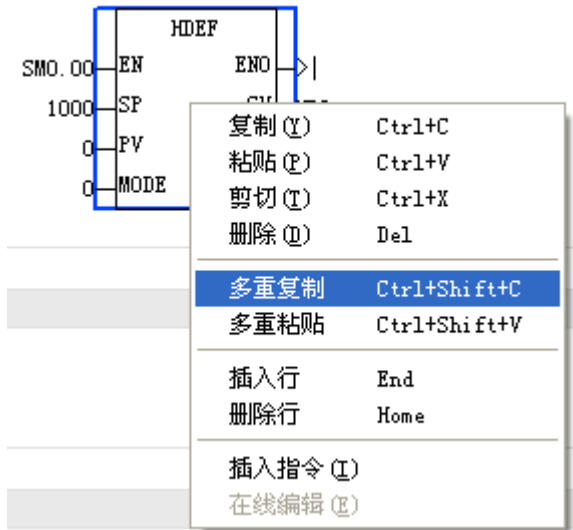


图 2.3.37 指令的多重复制图例

➤ 配置多重复制信息

在多重复制对话框中配置多重复制信息，其中，只有变量偏移量列是可以修改的，功能块信息、引脚信息、变量地址与类型列均为只读部分。

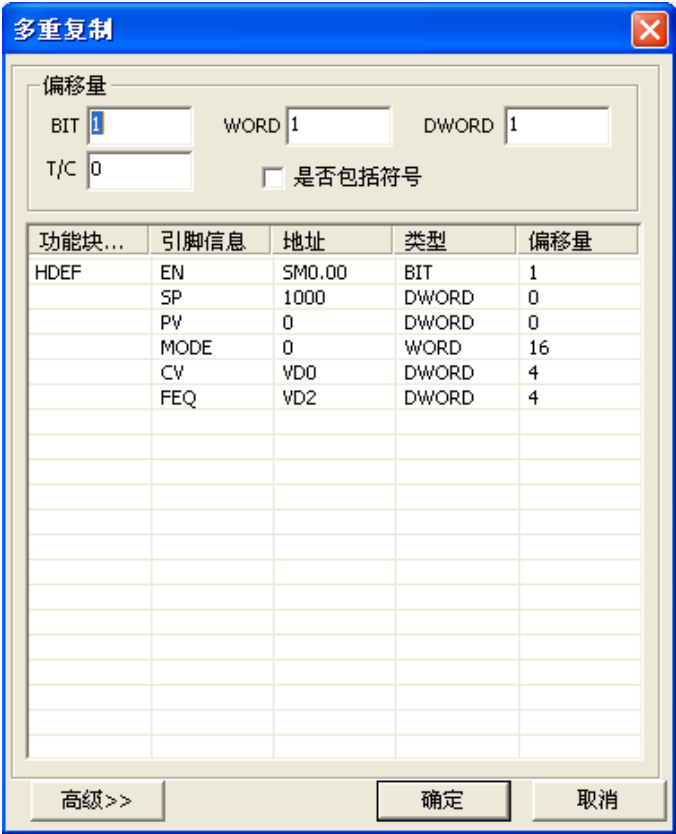


图 2.3.38 多重复制对话框

表 2.3.4 偏移量说明

偏移量名称	说明
Bit	位变量，可统一修改参数类型为 bit 的位变量参数
Word	字变量，可统一修改参数类型为 word 的字变量参数
Dword	双字变量，可统一修改参数类型为 dword 的字变量参数
T/C	T 区和 C 区的统一修改

当选择“高级”配置时，会在对话框中增加“符号名设置”栏，可以选择变量符号的复制增加方式。

注意：在选择“高级”配置时，需要在“是否包括符号”前打勾。

[illegible]

图 2.3.39 多重复制对话框

► 多重粘贴

指令被复制以后，用户可以在网络的空白处单击鼠标右键，选择“多重粘贴”选项，将指令粘贴到所需要的位置。粘贴时参数引脚也会按照配置的增量信息进行增加。使用快捷键 Ctrl+Shift+ V 也可以达到同样的目的，如图 2.3.40 所示。



图 2.3.40 多重粘贴结果

多重复制支持对多个指令进行操作，过程与单个指令完全相同。

2.3.2.4 FBD 中输入地址

当在 LAD 中输入一条指令时，参数用问号表示，例如 (???) 或 (????)。问号表示参数未赋值。用户可以在输入元素时为该元素的参数指定一个常量、绝对值、符号、变量地址等。如果有任何参数未赋值，程序将不能正确编译。

欲指定一个常量数值（例如 100）或一个绝对地址（例如 I0.1），只需在指令地址区域中键入所需的数值。（用鼠标双击或 Enter 键选择键入的地址区域）。

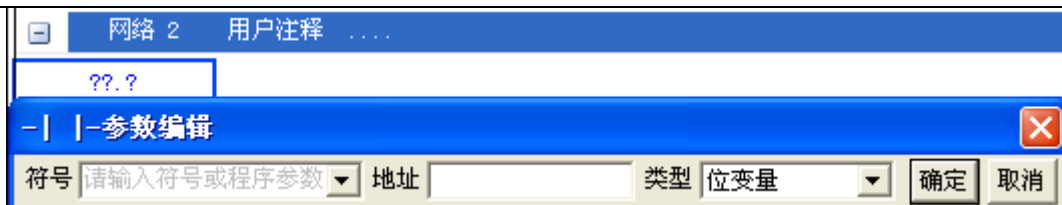


图 2.3.41 参数编辑器

当输入的地址异常时，会弹出错误提示对话框。输入异常情况有如下几种情况：

(1) 当地址变量的类型输入错误时，按下回车键或点击确定后会弹出提示对话框，提醒用户需要重新输入地址变量。

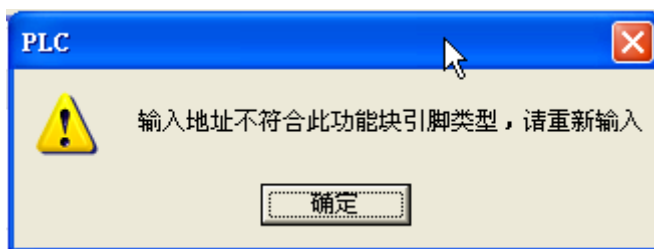


图 2.3.42 输入地址不符

(2) 在指令参数的编辑对话框中，如果用户在“符号”栏输入了符号名，生效软件将自动记录该符号。



图 2.3.43 参数编辑器图例

打开符号状态表，可以查看该符号。

符号	绝对地址	Modbus变量地址	数据类型
常闭触点	SM0.00	0x1140 (4416)	位变量

图 2.3.44 状态符号表图例

(3) 当添加的变量符号已存在时，按下回车键或点击确定后会弹出警告对话框，提示该符号已经定义，并指出定义的位置，要求用户重新定义。在程序下方的编译信息中会输出符号定义的位置，双击即可找到该符号。

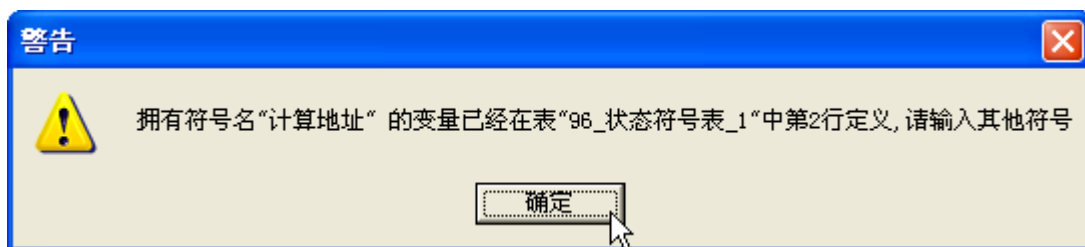


图 2.3.45 输入变量已存在

(4) 当输入的符号不符合变量的符号定义规则时，按下回车键或点击确定后会弹出警告对话框，提示错误原因。

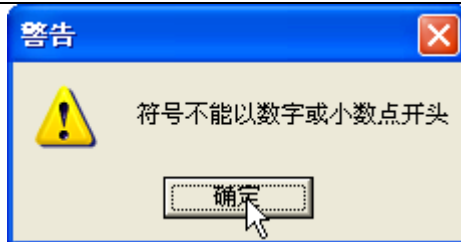


图 2.3.46 符号不符定义规则

(5) 当输入的变量地址不符合变量定义规则时，按下回车键或点击确定后弹出警告对话框，提示用户所输入的地址不合法。



图 2.3.47 输入地址不合法

(6) 当符号栏中输入地址变量时，按下回车键或点击确定后弹出警告对话框，提示用户符号名不合法。

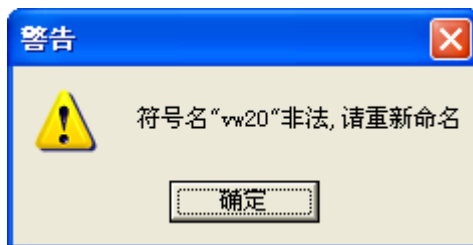


图 2.3.48 输入符号名非法

(7) 快速定义程序参数

在指令参数的编辑对话框中，如果用户在“地址”栏输入符号名，生效后如果软件认为该地址信息不符合该控制器的变量地址格式，则将其视为添加程序参数。



图 2.3.49 快速定义程序参数

2.3.2.5 FBD 中使用查找和替换功能

(1) 查找

若在程序编辑时需要查看变量的引用位置，可以通过系统主菜单“编辑->查找”，或键盘组合键“Ctrl+F”，或点击工具条中查找按钮 打开“查找”对话框，查找指令或变量。在查找对话框中输入要查找的内容，并选择查找参数的匹配项，点击查找后，结果会在输出栏的查找选项中显示。

查找功能支持查找参数、搜索当前选中与端点查找功能。

➤ 查找参数

在打开的“查找参数”对话框中填写查找内容，在配置选项中选择查找参数的匹配项，在查找范围中选择查找该变量的范围，选择完成后点击查找按钮。

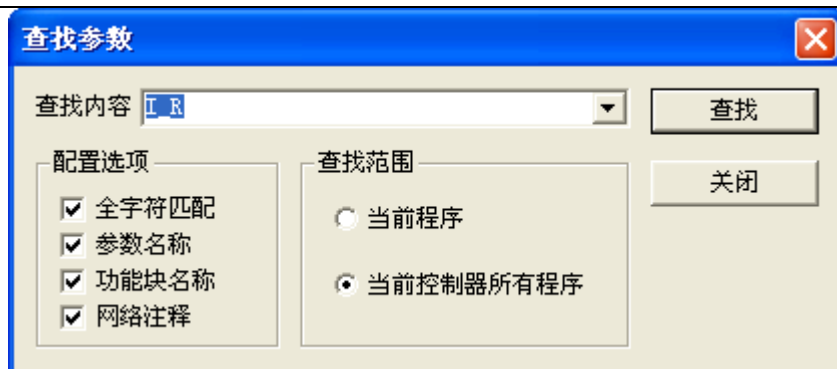


图 2.3.50 查找参数对话框

查找的结果将列在输出栏的查找选项中。

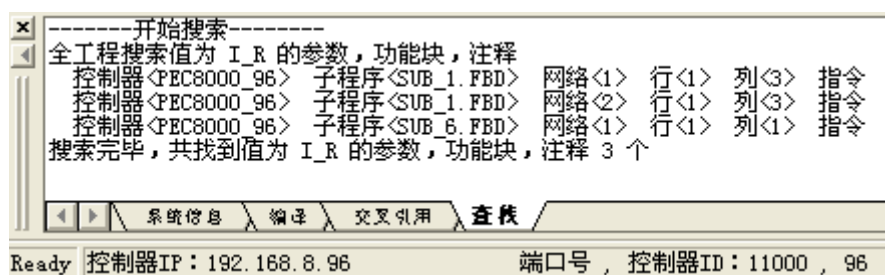


图 2.3.51 查找结果显示

➤ 查找端点

端点查找是指选中指令引脚。将鼠标放置在指令端点处，点击鼠标右键，在右键弹出菜单中选择查找该端点，在输出栏查找选项中会显示该变量引用的地址，指令名称，输入输出类型。

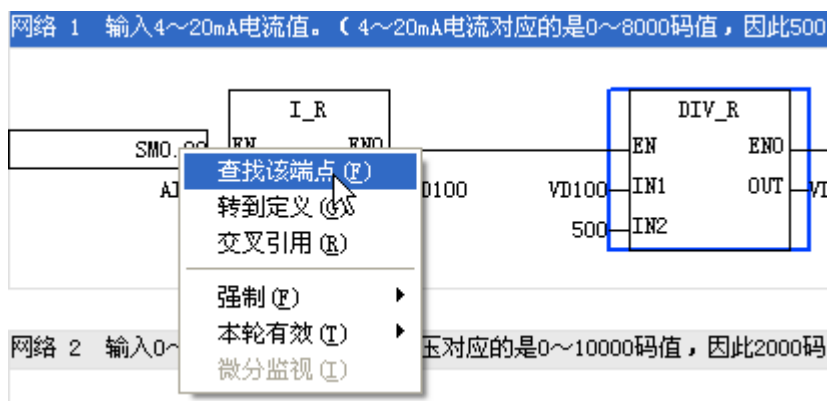


图 2.3.52 端点查找

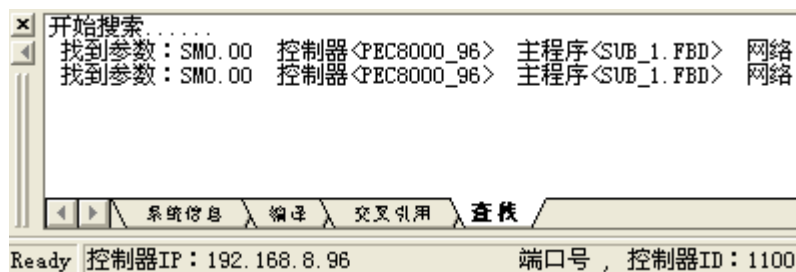


图 2.3.53 端点查找结果

端点查找不同于“查找参数”对话框查找，端点查找的查找范围为当前程序段，而“查找参数”对话框的查找范围是整个程序。

➤ 搜索当前选中

搜索当前选中，是指当选中指令或某一指令引脚变量时，按下“Ctrl+F”，在查找内容处会自动添加当前选中内容，点击“查找”按钮，在输出栏“查找”中列出相关变量的应用信息，能够方便用户查找功能块（梯形

图) 指令及其引脚变量的引用情况。

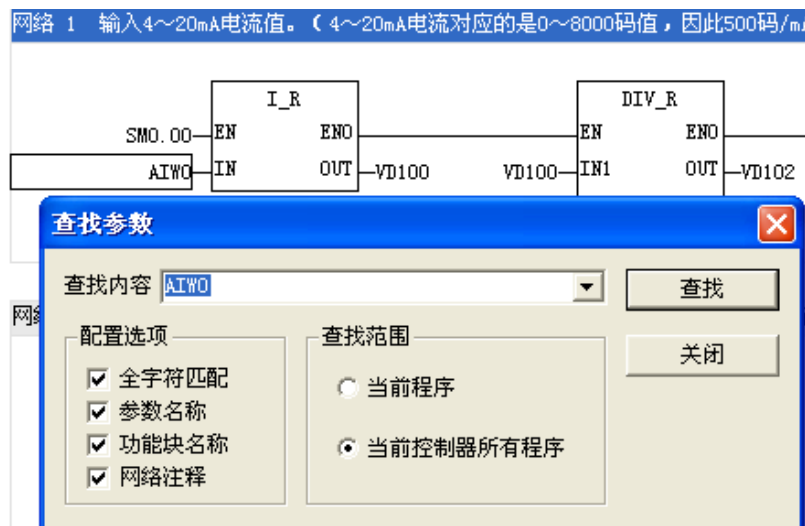


图 2.3.54 选定查找指令或参数图例

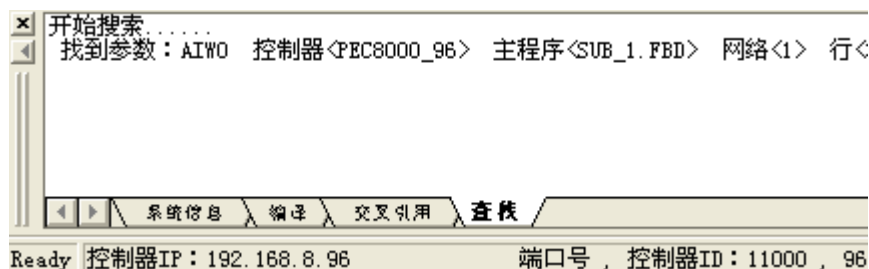


图 2.3.55 输出栏显示查找信息图例

在查找栏中，可以将工程中所有关于搜索变量（指令）的引用信息一次列出，双击其中的信息可定位到变量引用的梯形图引脚中。

(2) 替换

在程序编辑过程中，可以通过替换功能进行变量的替换。替换可分为单独变量替换和区间替换两个子功能。

➤ 单独变量替换功能

您可以通过以下三种方法进行替换操作：

- ①单击编辑工具条中替换  按钮
- ②使用快捷方式 Ctrl+H
- ③点击“编辑→替换”菜单

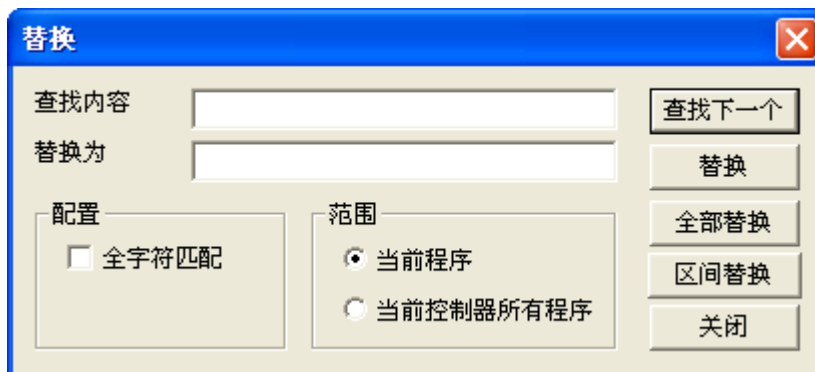


图 2.3.56 替换对话框

在查找替换对话框中填入您要查找、替换的内容，并选择替换范围，是否全字符匹配，然后点击功能按钮即可。

➤ 区间替换

在工程中有大量的连续变量需要修改成其他类型的变量时，可通过变量区间替换来完成，该功能可以为用户节省时间，避免在修改过程中出现遗漏变量的现象。具体步骤如下：

通过编辑菜单打开替换对话框，也可通过快捷键“Ctrl+H”打开替换对话框。

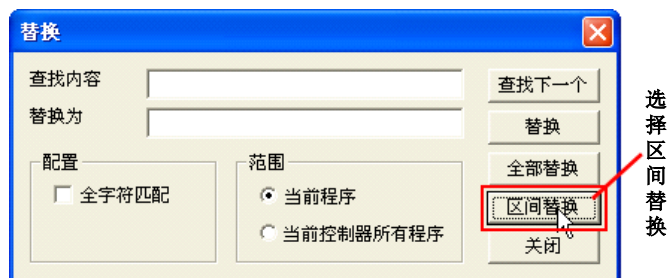


图 2.3.57 替换对话框图例

点击区间替换对话框，填写需要更改的变量区间，以及替换变量名称，在“范围”选项中选择替换变量是针对当前程序段还是对于当前控制器中的所有程序。

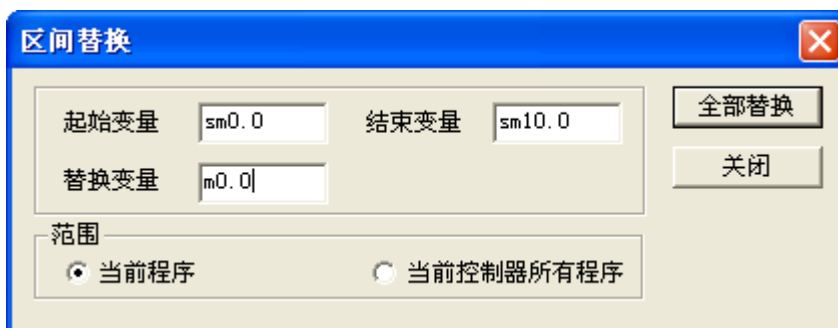


图 2.3.58 区间替换对话框

2.4 下载程序

当控制器与 PC 机连接成功后，用户可以将工程下载到 PLC 中，在下载之前，可以通过编译功能检查程序的正确性，通过输出栏提示程序错误的原因及相应位置。双击提示信息可直接定位到程序的错误位置，方便及时检查并完善程序逻辑。

2.4.1 编译检错

编译分为错误检查与生成下载信息两个过程，同时分成编译当前与编译全部两种方式。

点击  按钮或使用“设备→编译当前”菜单来编译当前显示的程序。

点击  按钮或使用“设备→编译全部”菜单来统一编译当前设备下所有程序。

表 2.4.1 列出了在编写组态程序时可能出现的错误，及解决方案，可作为参考。

表 2.4.1 编译出错信息

错误信息	说明及解决方法
参数编址错误或者立即数越界.....	指令参数填入了错误的变量，可能为错误的变量地址，或地址超出该变量区范围，查看指令说明，修改该变量
无输入参数错误	指令输入没有写入任何变量或立即数，写入正确的输入参数
无输出参数错误	指令输出没有写入任何变量或立即数，写入正确的输出参数
无头部参数错误	指令头顶参数没有写入任何变量，写入正确的头顶参数
目的 IP 地址与本地设备 IP 地址相同	设备不会为自己发送网络数据，修改目的 IP
PID 通道只支持 16 路	PID 回路号越界，超过了限制范围
网络太复杂不能编译	网络中有两个或多个彼此独立的程序结构； 解决方法：确保每个网络最多只有一套彼此相连的逻辑结构。如果想要多个参与逻辑控制，最好用变量传递的方法，在多网络中实现
网络逻辑错误，无法编译	梯形图存在回环，也可以说某梯形图指令无法确定能流的流向； 解决方法：将梯形图拆分，确保所有指令的能流流向
FOR-NEXT 不匹配	FOR 指令和 NEXT 指令不匹配，检查指令使之相匹配
此网络为转换后无法用梯形图表示的网络	功能块转换成梯形图时，某些功能块网络结构无法用梯形图表示。 解决方法：重新转换为功能块，并修改该网络的逻辑
此网络为转换后无法用功能块表示的网络	梯形图转功能块时，某些梯形图网络结构无法用功能块表示。 解决方法：重新转换为梯形图，并修改该网络的逻辑

2.4.2 程序下载

在程序执行编译且无错误后，通过  按钮可将编译完成的程序下载到设备中。下载时，软件会根据设备类型提示选择下载网络注释、程序指令注释和状态符号表。

提示：

- (1) 状态符号表中没有符号名的变量不会被下载到设备中。
- (2) 下载的变量信息含列：符号名、绝对地址、数据类型和参数说明。
- (3) 从设备上载的变量在状态符号表中的位置不变。

2.5 上载程序

用户可以通过点击  按钮将 PLC 中的程序上载到 PLC_Config 中。上载时，软件会根据设备类型提示用户选择上载网络注释、子程序指令注释和状态符号表。

2.6 监控与仿真

PLC_Config 支持程序监控、状态符号表监控以及趋势图等功能，通过这些功能可以提供程序状态的显示与跟踪，方便快速地掌握设备的实时状态和程序的运行效果，并帮助及时纠正程序的错误和偏差。

此外，在没有实际设备的情况下，PLC_Config 软件还可以使用仿真功能，模拟真实设备运行，调试程序。仿真功能提供灯板和模拟量修改界面，可快速的配置外部输入条件，达到模拟现实的效果。

2.6.1 设备的运行与停止

程序下载后，设备会自动进入运行状态。用户也可以通过以下两种方式进入运行状态：点击  按钮使其变为凹陷  状态；选择菜单“调试→全速运行”。

停止控制器运行有以下四种方式：点击  按钮使其恢复为弹起  状态；取消勾选“调试->全速运行”菜单；使用停止按钮  强制停止程序运行；点击“调试→停止”菜单。

在设备运行时，可以对程序运行轮次进行设定，通过菜单“调试→首次上电运行一轮”或工具栏中  按钮将程序设置“首次上电运行一轮”；若需要程序运行多轮，可通过菜单“调试→设置运行轮次”或工具栏中  按钮设置程序执行轮次。

2.6.2 强制写入

在线监控的状态下，用户可以强制改变变量的数值，如图 2.6.1 所示。

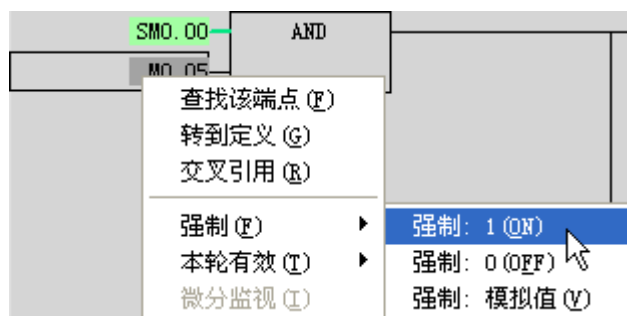


图 2.6.1 强制写入变量

在指令引脚上点击右键，选择“强制”，在其子菜单中，可以选择强制写入 1、0 或模拟值。

若引脚变量为常数，则引脚参数不可被强制写入，此时弹出提示框，“该参量无法被强制写入！”。



图 2.6.2 不可写入提示

2.6.3 在线调试

在线调试功能包括本轮有效、微分监视和在线编辑功能，在线调试功能既可以使设备在调试过程中不影响设备在整条生产线的工作，又可以帮助工程师快速定位组态程序运行出现的问题。

● 本轮有效

在程序处于监视状态时，可通过鼠标右键单击功能块引脚弹出的菜单选择本轮 1 有效和本轮 0 有效。当选择本轮有效功能时，程序会对当前运行状态进行判断：当主循环未开始执行时，立刻执行本轮效功能，在本轮循环结束后，变量值恢复为初始值；若主循环在已开始执行时，则本轮有效功能在主循环下一轮执行，当本轮有效执行后变量立即恢复为初始值。

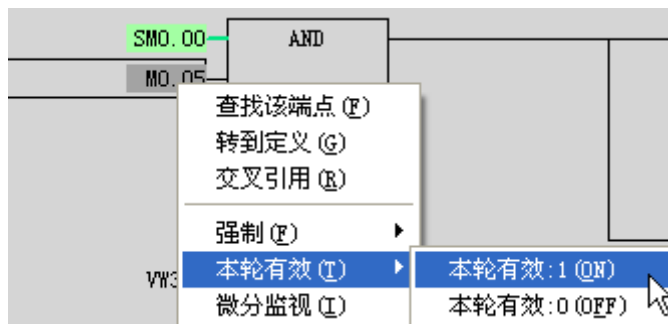


图 2.6.3 本轮有效写入

注意：只有位变量才可以使用本轮有效功能。

● 在线编辑

在程序处于监视状态时，程序处于不可修改状态。在当前程序段内任意网络内点击鼠标右键，在右键菜单中选择“在线编辑”，此时，该网络的背景颜色变为白色后该网络内的程序则处于可修改状态，修改程序后，再点击下载按钮将组态工程下载到设备中，在程序的下载过程中设备可以正常运行。下载完成后监控，当前程序网络处于不可编译的灰色状态。



图 2.6.4 在线编辑程序

- 微分监视

在程序处于监视状态时，选择某一指令，单击右键弹出的菜单中选择“微分监视”，打开“微分监视”对话框。在地址处可以修改变量名称，通过“边沿”选择监视上升沿变化次数或下降沿变化次数，点击“开始”按钮开始进行监视。

当该功能处于监视状态时，配置选项栏为只读状态，“微分监视”对话框中添加计数部分，当程序在运行过程中出现一次上升沿，颜色改变一次，由初始的黑色变为白色，当再次出现上升沿时，颜色由白色变为黑色，计数由 0 改变为 1，再由 1 改变为 2。

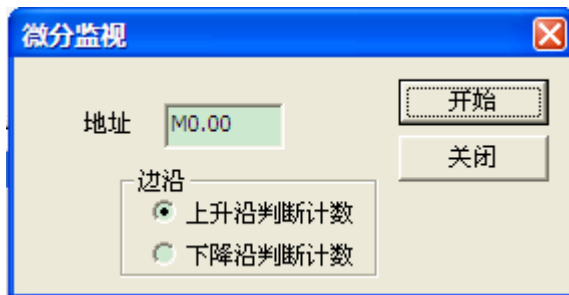


图 2.6.5 微分监视功能

提示：只有运动控制系列产品、PEC2000 和 EDC8000 等新模具，且设备版本为 5.0 以上时可用，在程序监控状态下，右键点击位变量，在右键菜单中才会出现“微分监视”选项。

- 状态符号表和趋势图的监控

符号变量表提供变量的集中显示与管理功能，趋势图用于监控变量实时值，可以在一定时间内监控变量的变化趋势，具体用法请参见“程序元件及工作原理”章节。

2.6.4 仿真设备运行

仿真运行是指对当前程序中编辑的组态程序在仿真器中模拟运行，在不具备真实设备的情况下调试组态程序。启动仿真后，软件自动将当前组态设备的程序下载到仿真器中，同时启动程序监控。

仿真运行期间，如下操作将针对仿真器。这些操作不影响真实设备，无论设备是否在线，都可以通过仿真运行功能调试组态程序。

- (1) 组态程序监控；

- (2) 状态符号表监控；
- (3) 实时曲线监控；
- (4) 只读变量区监控；
- (5) 通信命令：程序运行、停止、首次上电运行轮次、设置运行轮次。

启动仿真的方法是点击工具栏中按钮。仿真期间按钮为按下状态，关闭仿真期间按钮为弹起状态。启动仿真运行后，PLC_Config 软件将弹出设备的灯板界面，如图 2.6.6 所示。

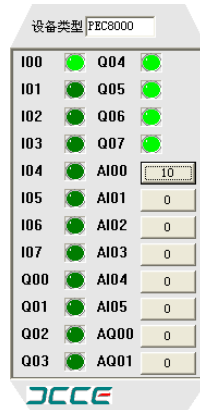


图 2.6.6 仿真设备灯板

灯板界面中能够显示设备的数字量输入输出灯与模拟量输入输出，与真实设备灯板略有不同。可以点击仿真界面上的小灯图标来写入数字量，将当前状态值切换到该灯当前状态的相反值；点击模拟量按钮后弹出输入值对话框界面，可以写入相应整型数字，点击确定后写入该模拟量值。

2.7 PLC_Config 软件操作

2.7.1 主程序与子程序

2.7.1.1 替换主程序

每个控制器都会拥有一个主程序，PLC_Config 支持替换主程序的功能，用户可以使用以下 2 种方法来替换主程序：

- 右键点击主程序，选择“替换主程序”
- 软件主菜单，编辑→程序块→导入并替换主程序



图 2.7.1 替换主程序选项图例

2.7.1.2 添加删除子程序

PLC_Config 支持添加子程序功能，用户可以通过以下两种方式添加子程序：

- 通过“编辑→程序块→导入并添加子程序”菜单
- 在“子程序”目录树上单击鼠标右键，选择“添加”，来添加一个子程序

在弹出的创建新程序对话框中填入程序名称，并选择相应的程序类型、之后点击“确定”即可。子程序名称不可为空。



图 2.7.2 添加子程序图例

点击某子程序结点，按键盘 Delete 键，或在右键菜单中选择删除，即可将该子程序删除。

2.7.1.3 导入子程序

在子程序结点右键菜单中选择“导入”，在本地选择功能块或梯形图程序，即可实现导入功能。

2.7.1.4 子程序的调用

子程序调用可分为直接插入子程序指令和 CALL 指令调用两种。

(1) 直接插入子程序指令

在网络空白处点击“Enter”键，可浏览到当前程序所包含的子程序与其他指令，选择需要的子程序即可将其添加到程序中。

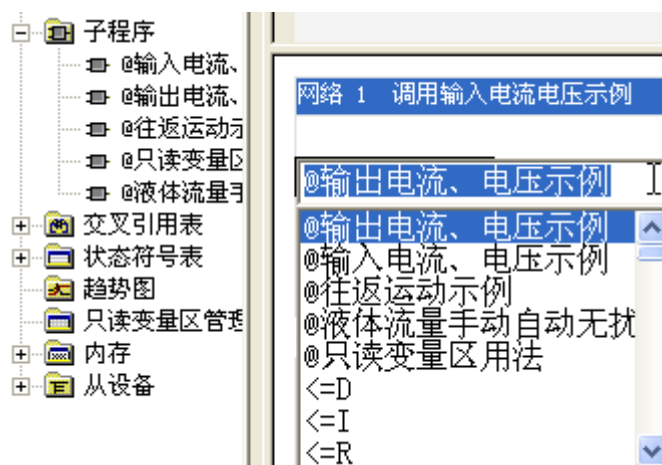


图 2.7.3 子程序的调用

(2) CALL 指令调用

在网络空白处添加 CALL 指令，在符号一栏中选择指定子程序，确定后 CALL 指令将自动变成子程序功能块。

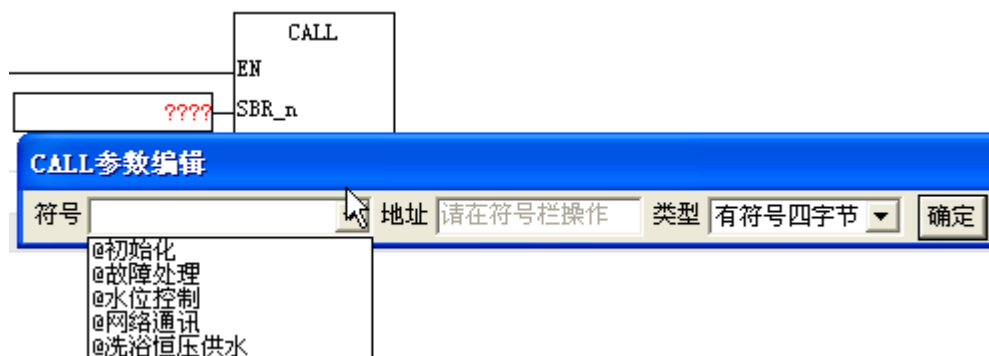


图 2.7.4 CALL 指令调用

2.7.1.5 程序统计

程序统计功能可以统计出当前子程序的所有指令数量。通过菜单“查看→程序统计 (R)”选项查看。

所含程序块名	网络总数	功能块数	字节总数
MAIN (主程序)	9 个	0 个	10 字节
输入电流、电...	3 个	7 个	215 字节
输出电流、电...	7 个	34 个	920 字节
往返运动示例	4 个	12 个	365 字节
只读变量区用法	2 个	10 个	310 字节
液体流量手动...	9 个	16 个	525 字节
总计	34 个	79 个	2345 字节

图 2.7.5 功能块属性表单图例

2.7.2 子程序参数编辑

在网络上方的参数编辑区域中，可以对已添加的子程序参数进行添加、删除或修改参数操作，参数编辑区如图 2.7.6 所示。

ID	名称	输入输出	数值类型
0	电压模拟值	内部变量	位变量
1	电流模拟值	内部变量	位变量
2	电压源输入值	内部变量	位变量
+ 添加参数			

图 2.7.6 子程序参数栏图例

2.7.2.1 添加和删除参数

双击参数表格中最后一行的“添加参数”，或点击右键菜单中的“添加参数”菜单即可实现添加参数功能。添加参数时会弹出如图所示的对话框，用户需要填写必要的参数信息。

图 2.7.7 子程序增加参数

参数信息的具体含义如表所示。

表 2.7.1 参数各项信息说明

名称	当前参数的名称
输入输出类型	参数的 I/O 类型，可以选择输入、输出、输入输出及内部变量， 输入：该参数用作程序的输入，在程序调用前需向该参数赋值。 输出：该参数用作程序的输出，在程序调用后将该参数的值输出到变量。 输入输出：程序执行前将该参数赋值，程序调用结束将该参数值输出到变量。 内部变量：仅在程序内部使用的局部变量。（主程序添加参数只能为内部变量）
数值类型	参数的数值类型，用户可以选择位变量、有符号字、无符号字、有符号双字、无符号双字，浮点
注释	介绍该参数的意义

选择某参数后，点击“Delete”键，或点击右键菜单中的“删除参数”选项，即可将此参数删除。

注意：同一段程序内的参数名称不可相同，也不可为空。

2.7.2.2 修改参数

在程序参数表格中双击某参数，或单击选中该参数后，点击右键菜单的“修改参数”菜单项，即可打开修改参数的对话框。

2.7.2.3 上下移动

用户可移动程序参数的上下顺序，从而调整程序功能块的显示样式。选择某参数后，点击右键菜单中的“向上移动”或“向下移动”来完成移动操作。

参数仅能够在相同的输入输出类型中移动，如果某输出类型的参数 Out1 上移时，成为了第一个输出类型的参数，但上面还有一个输入类型的参数 In1，这时 Out1 也不能够继续上移了。

2.7.2.4 复制粘贴

PLC_Config 提供程序参数的复制粘贴功能，可以通过程序参数表格的右键菜单实现该操作，或选择某参数后通过键盘组合键 Ctrl+C 和 Ctrl+V 来进行复制粘贴操作。

2.7.3 连线操作

2.7.3.1 建立连接线

功能块编程语言中，具有能流作用的位输入输出引脚可以连线。该类型的引脚在无参数时默认显示成“<<”“>>”“>”“>”。



图 2.7.8 选中一个引脚

选中一个引脚，并在其上双击鼠标左键，若鼠标变为十字光标说明此操作成功；或按住键盘 Ctrl 键，用鼠标单击该引脚。

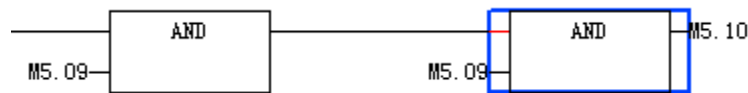


图 2.7.9 连接成功

在另一个功能块指令管脚上按上述步骤做同样操作，即可实现连接操作。

梯形图程序语言中，连线的最小单元是每个网格中的横线和竖线，由多个网格的横线和竖线共同组成指令引脚间的连接关系。在一个网格中，可点击工具栏中的“   

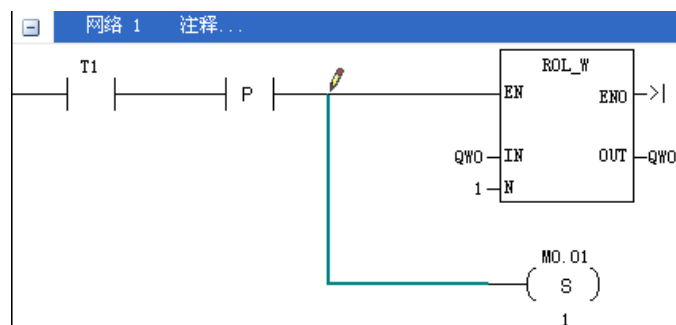


图 2.7.10 梯形图程序连线操作

2.7.3.2 删除连接线

在网络内选中该连线，当连线高亮时，证明其已被选中，点击“Delete”键即可删除该连线。

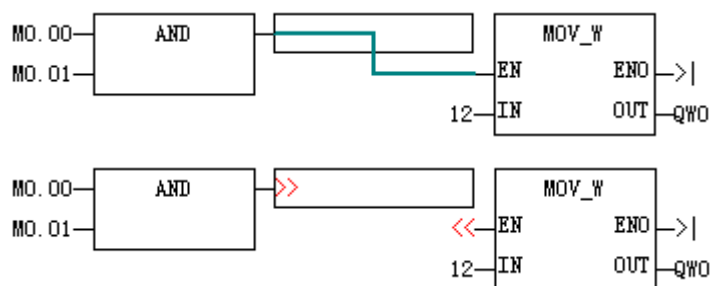


图 2.7.11 删除连线图例

2.7.4 网络操作

2.7.4.1 添加插入删除网络

您可以通过网络工具栏或“编辑→网络”菜单来添加和删除网络。

(1) 末尾添加网络

 按钮或“编辑→网络→末尾添加网络”。在所有网络的末尾增加一个新的网络。若是没有网络的空程序，则添加一个新的网络。

(2) 删除末尾网络

 按钮或“编辑→网络→删除末尾网络”。删除所有网络中最后一个网络。若仅有一个网络时，则清空该功能块程序。

(3) 后插网络

 按钮或“编辑→网络→后插网络”。在当前选中的网络后添加一个新的网络。

(4) 前插网络

 按钮或“编辑→网络→前插网络”。在当前选中的网络前添加一个新的网络。

(5) 删除当前网络

 按钮或“编辑→删除”。删除当前选中的网络。

2.7.4.2 编辑网络注释

选中一个网络，在该网络的标题栏上双击鼠标左键，在弹出的对话框中填入注释信息。

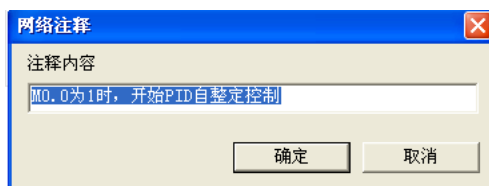


图 2.7.12 添加网络注释

2.7.4.3 网络的选择

单击网络的注释区，可实现选择该网络，此时进行的复制、删除、剪切等操作将针对该网络。

2.7.4.4 书签功能

您可以通过编辑工具栏或“编辑→书签”菜单来进行书签相关操作。

(1) 切换书签

 或“编辑→书签→切换书签”。若被选中的网络未添加书签，标题栏将变色为浅绿色；若此网络已经添加书签，则会删除该书签。

(2) 上一个书签

 或“编辑→书签→上一个书签”。跳转到上一个被标记为书签的网络，若之前不存在书签，则返回最

后一个网络开始向上查找。

(3) 下一个书签

 或“编辑→书签→下一个书签”。跳转到下一个被标记为书签的网络，若之后不存在书签，则返回第一个网络开始向下查找。

(4) 清除全部书签

 或“编辑→网络→清除全部书签”。删除所有网络的书签标记。

2.7.4.5 插入和删除行

选中一个网络，单键盘上的 End 键或通过“编辑→网络→插入行”菜单，您可以在网络底部插入一行，网络最多可为 32 行。

选中一个网络，单击键盘上的 Home 键或通过“编辑→网络→删除行”菜单，您可以删除网络最后一行，若最后一行存在指令，则无法删除。

2.7.4.6 网络跳转

您可以通过三种方式跳转到指定网络：

- (1) 点击工具栏中  按钮
- (2) 通过 Ctrl + G 快捷键
- (3) 通过“编辑→转到”菜单

在弹出的“跳转网络”对话框中填入您要跳转到的网络号即可。

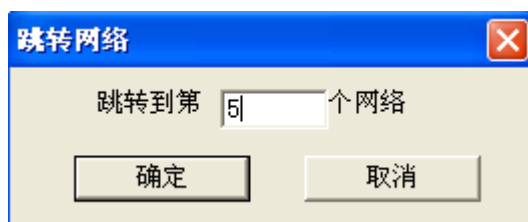


图 2.7.13 跳转到指定网络

2.7.5 指令参数编辑

2.7.5.1 AND 和 OR 的引脚增删

功能块编程语言中，AND 和 OR 指令的输入参数数量可变，范围在 1~8 个之间。可以通过功能块工具栏或“编辑→功能块”菜单来进行功能块相关操作。

(1) 增加输入

 按钮或“编辑→功能块→增加输入”。仅对 AND/OR 块有效，为 AND/OR 块增加一个输入引脚，输入引脚最多为 8 个。

(2) 删除输入

 按钮或“编辑→功能块→删除输入”。仅对 AND/OR 块有效，为 AND/OR 块删除一个输入引脚，输入引脚最少为 1 个。

2.7.5.2 位输入取反

功能块编程语言中，位输入变量可以将输入的数值取反处理。

 按钮或“编辑→功能块→切换求反”。

2.7.5.3 跳转到符号

选择某参数后，点击右键菜单中的“转到定义”菜单项，即可定位到状态符号表的具体位置。



图 2.7.14 转到定义图例

符号	绝对地址	Modbus变量地址	数据类型
	AIWO	0x0000 (0)	有符号双字节
	VD100	0x0984 (2436)	四字节浮点数
	VD102	0x0986 (2438)	四字节浮点数
s1	VW50	0x0952 (2386)	有符号双字节

图 2.7.15 跳转到状态符号表图例

2.7.5.4 交叉引用显示

当您需要查看某一指令所有参数的引用情况时，可以打开输出栏的“交叉引用”栏，然后鼠标单击该指令，输出栏即会列出该指令的所有参数被引用的情况，双击输出栏的某一条信息时可以直接定位到具体的引用位置。

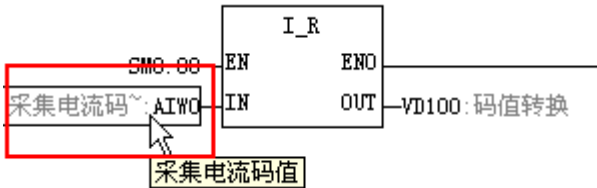


图 2.7.16 选择变量查询

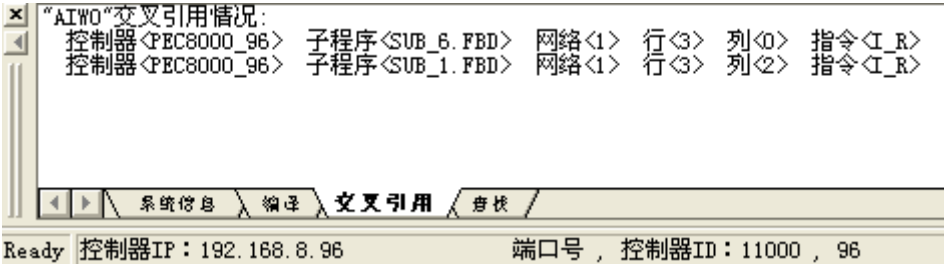


图 2.7.17 变量查询结果

2.7.6 功能块梯形图互转

PLC_Config 支持功能块、梯形图语言编程，支持指令表语言的程序显示，您可以在编写完程序后，通过转换功能将组态程序转变为您更熟悉的程序语言，方便您更好地完成组态程序。

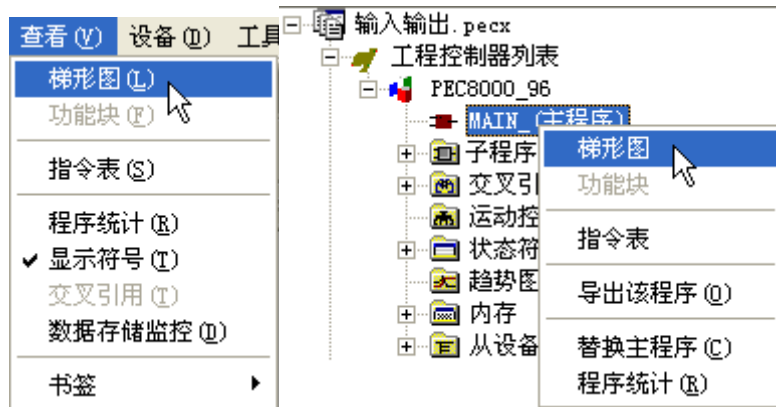


图 2.7.18 更改程序类型

- 通过“查看→梯形图”菜单或 **LAD** 按钮，可将程序转换为梯形图语言。
- 通过“查看→功能块”菜单或 **FBD** 按钮，可将程序转换为功能块语言。
- 通过“查看→显示指令表”菜单或 **STL** 按钮，可将程序转换为指令表显示。

您也可以右键点击工程树中待转换文件，选择程序的显示方式。

如图 2.7.19 中，为梯形图语言转为功能块语言的样例。

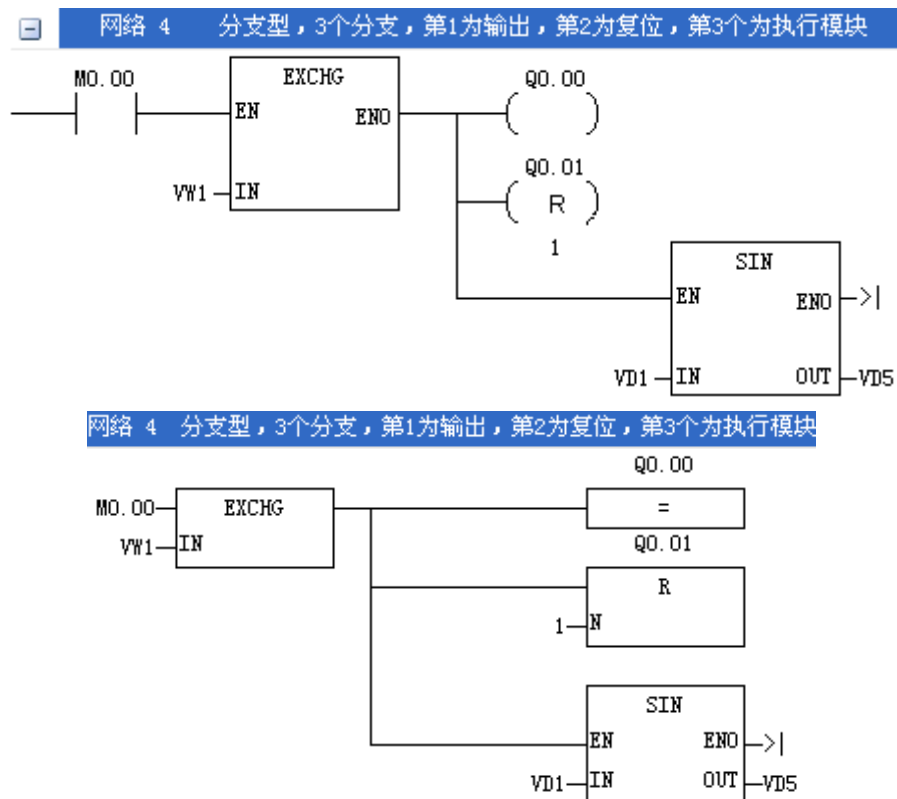


图 2.7.19 梯形图与功能块互转

表 2.7.2 三种编程语言对比

指令	样式
梯形图	
功能块	
指令表	<pre> LD SMO.00 A MO.01 S= TEMP7.15 MOV_W TEMP7.15 VW1 TEMP7.15 VW3 A TEMP7.15 S= Q0.00 </pre>

提示：由于两种语言处理逻辑和方式不同，部分情况下可能出现无法转换的现象，在转换过程中会提示用户。无法转换的情况与详细原因，请查看表 2.7.4 或上接表 2.7.5。

表 2.7.3 不能转换网络举例

原网络	
转换得到网络	<pre> LD MO.00 LD VO.00 S= TEMP7.15 MOV_W TEMP7.15 VW1 TEMP7.15 VW3 A TEMP7.15 ALD S= Q0.00 </pre>
注释信息	网络 5 转换失败，执行模块与AND块连线关系无法转换为梯形图！

不能转换的网络均可以通过反向转换恢复为原来的网络。

表 2.7.4 功能块不能转换为梯形图

原功能块网络	提示信息	不能互转原因
	转换失败，该网络无法用梯形图表示！多输出与多输入结构冲突。	功能块中出现环形结构，或出现无法用指令表（STL）描述的结构，无法用梯形图表示这种网络。 解决方案：拆分成多个小网络进行转换。
	转换失败，执行模块与AND块连线关系无法转换为梯形图！	执行类模块（一般带 EN 输入）连接在 AND 的第 2/3/4... 个引脚，无法用梯形图表示这种网络。 解决方案 1：将执行类模块连接到 AND 的第 1 个输入引脚。 解决方案 2：拆分成多个小网络进行转换。

上接表 2.7.4 功能块不能转换为梯形图

原功能块网络	提示信息	不能互转原因
	转换失败，分支结构连接 OR 功能块时无法转换为梯形图	功能块的分支结构中，支路的能流输出传递给 OR 块的输入，无法用梯形图表示这种网络。 解决方案：拆分成多个小网络进行转换。
	转换失败，请检查分支结构的最后一个支路！	分支结构的最后一个支路连接在 AND 的第 2/3/4...个引脚，无法用梯形图表示这种网络。 解决方案 1：将最后一个支路连接到 AND 第 1 个输入引脚。 解决方案 2：拆分成多个小网络进行转换。
		分支结构的最后一个支路连接在 RS/SR/CTU/CTD/CTUD 的输入，无法用梯形图表示这种网络。 解决方案：拆分成多个小网络进行转换。
	转换失败，P/N 模块不能摆在这个位置！	P/N 在转换为梯形图后将会出现在并联结构中，这在梯形图中是无法编译通过的。 解决方案 1：尝试使用其他结构来实现需要的功能。 解决方案 2：拆分成多个小网络进行转换。
	转换失败，梯形图网络中只允许同时有一个 RS/SR/CTU/CTD/CTUD 指令！	功能块网络中允许存在多个 RS/SR/CTU/CTD/CTUD 指令，但在梯形图中每个网络只能存在一条该种指令。 解决方案：拆分成多个小网络进行转换。
限于篇幅，该图形仅为示例，实际编程红色部分应为直线)	网络过大，请拆分成小网络转换！	转换得到的梯形图网络超过 32*32 的上界，无法显示。 解决方案：拆分成多个小网络进行转换。

上接表 2.7.5 梯形图不能转换为功能块

原梯形图网络	提示信息	不能互转原因
	存在 WPC、PS、STORE 等特殊指令，无法转换	这 3 条指令在功能块与梯形图中的引脚定义不同，暂时无法直接转换。 解决方案：手动转换。
	转换错误，该网络结构无法转换为功能块	梯形图的并联结构中出现执行模块（如 MOV_W 等），无法用功能块表示这种网络。 解决方案 1：手动转换。 解决方案 2：拆分成多个小网络进行转换。
	转换失败，网络过大（宽度：x 高度：y），请尝试拆分成小网络再转换 提示：x、y 为功能块网络期望的宽度和高度	梯形图转换为功能块后，功能块网络尺寸超过 32*32 的上界，无法显示。 解决方案：依据注释中给出的此网络预期大小，将此网络拆分成多个小网络进行转换。
	网络过大，功能块密集摆放，请手动调整	转换得到的功能块网络能够编译通过，但摆放较密集，可能出现部分指令与连线交叉的现象。 解决方案 1：手动调整。 解决方案 2：拆分成多个小网络进行转换。

提示：在编程过程中，PLC_Config 提供了许多提示信息，例如指令引脚变量输入不正确变红、变量多重输出、指令格式不正确等提示信息。这些提示信息能够帮助确认程序编写是否正确，提高编码的准确性。

2.8 系统工程创建步骤

在 PLC_Config 中，每一个实际的应用案例都可称作工程，工程包含数据库、I/O 设备、人机界面、网络应用等组态信息和运行数据。PLC_Config 以工程为单位对控制器进行管理，支持新建工程、打开工程、保存工程、工程另存为、最近打开工程、关闭工程等操作。

2.8.1 创建工程

打开 PLC_Config 软件后，创建工程。

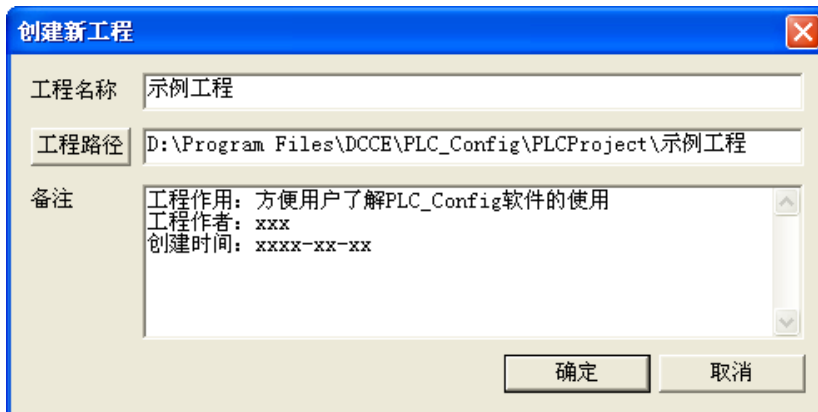


图 2.8.1 新建 PLC_Config 工程

2.8.2 添加控制器

PLC_Config 将控制器分为两类：网络控制器、工程控制器。在控制器管理中可显示出当前网络内的所有控制器情况，此时这些控制器即为“网络控制器”，网络控制器不会保存到工程文件中。用户选择自己所需的控制器，双击将其添加到 PLC_Config 的工程树中，此时添加在工程树中的控制器即为“工程控制器”，工程控制器的信息将被保存在工程文件中，下次打开工程时仍可看到该控制器信息。

2.8.2.1 添加网络控制器

双击主界面左侧工程树中的“控制器管理”结点，弹出如图 2.8.2 所示的“控制器管理”对话框。



图 2.8.2 控制器管理对话框图例

网络控制器列表中将网络中已存在的控制器自动扫描上线，双击该控制器实现将其添加到工程的操作。如果当前不存在任何物理控制器，用户可手动添加控制器，如图 2.8.3 所示。

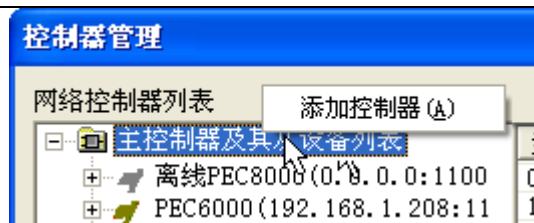


图 2.8.3 添加控制器

控制器添加到工程后，关闭控制器管理对话框，即可在软件左侧工程树中看到控制器结点。

PLC_Config 软件支持以太网和串口的通讯模式，若采用串口通信模式，配置方式见 4.1 串行通信服务章节。

2.8.2.2 添加工程控制器

与网络控制器的区别在于，工程控制器指被添加到工程中的控制器，PLC_Config 支持的编程、程序上下载、监控等功能均针对工程控制器提供，当工程关闭时，所有被添加到工程中的控制器信息将被保存，而网络控制器无此功能。

- 双击或者通过菜单“控制器→添加控制器”添加控制器

在网络控制器中双击选择要添加的控制器，弹出“控制器名称”对话框，如图 2.8.4 所示。为其命名后确定，该控制器将被添加到工程中。



图 2.8.4 添加控制器名称

- 删除工程控制器

加入到工程中的控制器将出现在“工程控制器列表”中，如果不再需要此控制器，可以在工程控制器列表中右键点击该控制器将其删除。

- 改控制器通讯接口和控制器类型

使用设置通信接口功能，可以改变当前工程控制器所对应的 IP 地址或域名地址，从而实现快速将当前程序下载到不同地址的控制器中。

在工程树中，选择需要修改通信接口的控制器，点击鼠标右键，在右键菜单中选择“设置通信接口”，在弹出的控制器地址对话框中填写控制器信息。如图 2.8.5 所示，可通过修改域名或 IP 地址来修改控制器地址。

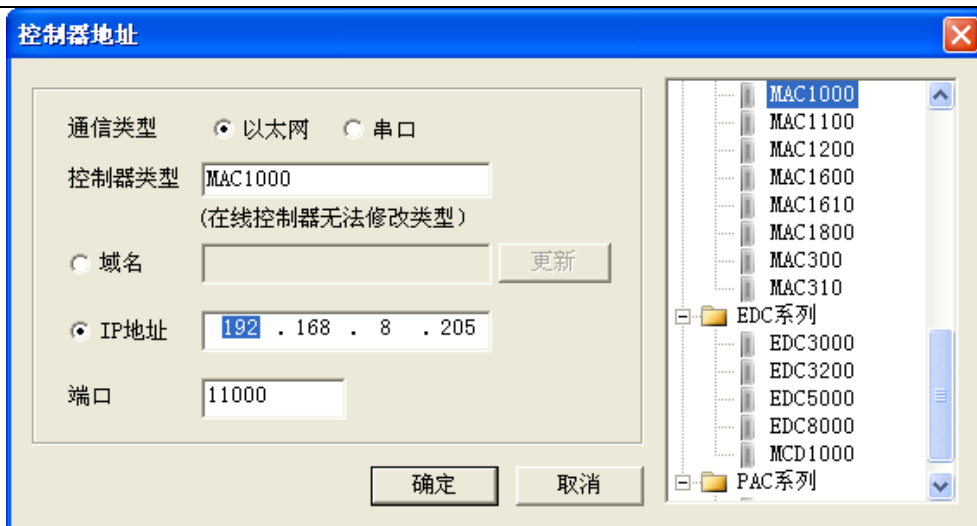


图 2.8.5 设置通信接口

当离线模式下，可以设置控制器类型。在打开的控制器地址对话框中，通过下拉框选择要修改的控制器类型。点击确定完成修改。

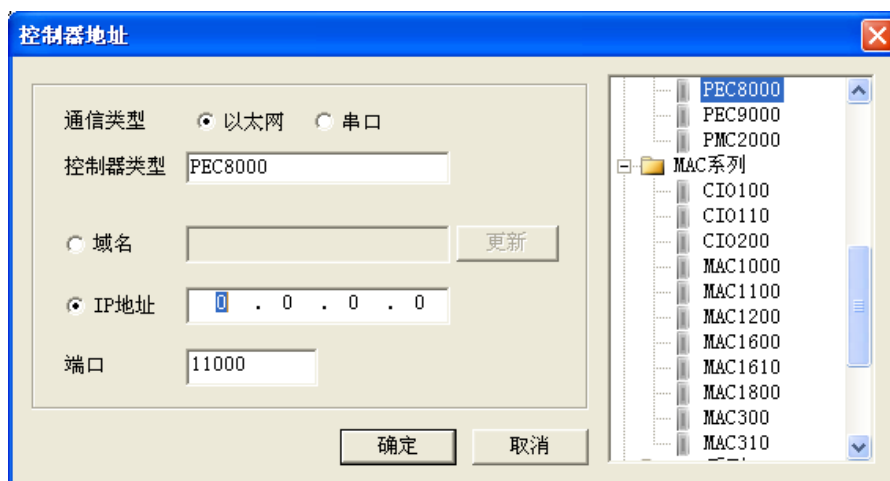


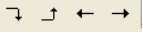
图 2.8.6 设置控制器类型

2.8.3 添加从设备

如果需要添加的为主从网络通信，则可以在工程中添加从设备，第三方设备要求支持 MODBUS 协议。具体如何添加从设备与如何配置从设备信息请参见 5.3 主从网络通信章节。

2.8.4 添加指令

添加指令的方式有两种：通过指令树双击指令或快捷方式（回车键或指令快捷工具条）来完成。在梯形图编程语言中，向同一网络添加指令时，该指令会自动向前添加连接线。在功能块编程语言中，具有能流作用的位输入输出引脚可以连线。

添加连接线的的方式：选中一个引脚，在该引脚处双击鼠标左键，当鼠标变为十字光标时，再双击另一指令的引脚；或按住键盘 Ctrl 键，用鼠标单击指令引脚；通过快捷键“Ctrl+↑”、“Ctrl+↓”、“Ctrl+←”、“Ctrl+→”或通过使用  快捷工具条。

2.8.5 添加变量

在指令引脚处添加变量的方式有两种：一是通过鼠标左键双击指令引脚，二是在选中引脚处按下回车键。无论是哪一种添加方式，都会弹出参数编辑助手，在地址栏中填写变量名称，在符号栏中为变量填写符号，类型处选择变量的类型。

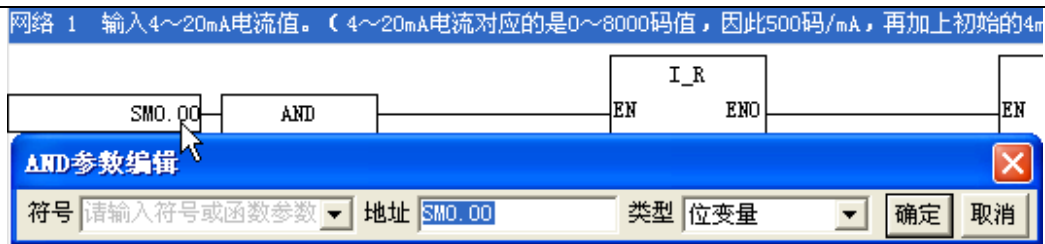


图 2.8.7 参数编辑助手

在编辑变量时，要根据指令引脚类型添加变量，如 0 中，AND 指令的引脚需要添加的变量类型为位变量，若在地址处添加变量为“VD100”，点击确定弹出错误提示对话框，提示“输入的地址不合法”，请重新输入。

提示：

1) 添加参数引脚处输入的地址与功能块引脚类型不符，按下回车键或点击确定后弹出提示对话框，提示“输入地址不符合此功能块引脚类型，请重新输入”。

2) 添加的变量符号已存在，按下回车键或点击确定后弹出警告对话框，提示该符号定义的位置，请重新输入其他符号。

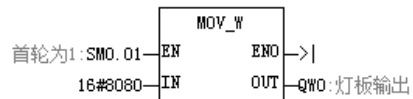
3) 符号不能以数字或小数点开头。

4) 若符号栏中输入地址变量，按下回车键或点击确定后弹出警告对话框，提示“符号名“***”不合法，请重新命名”。

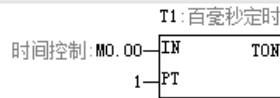
2.8.6 编写并下载程序

点击软件左侧工程树控制器下的 MAIN（主程序）结点进入编程页面，用户可根据编程习惯使用不同的编程语言，程序结点的右键菜单提供切换功能。下面举例一个简单的跑马灯程序。

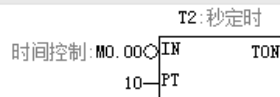
网络 1 将初始值16#8080写入QW0



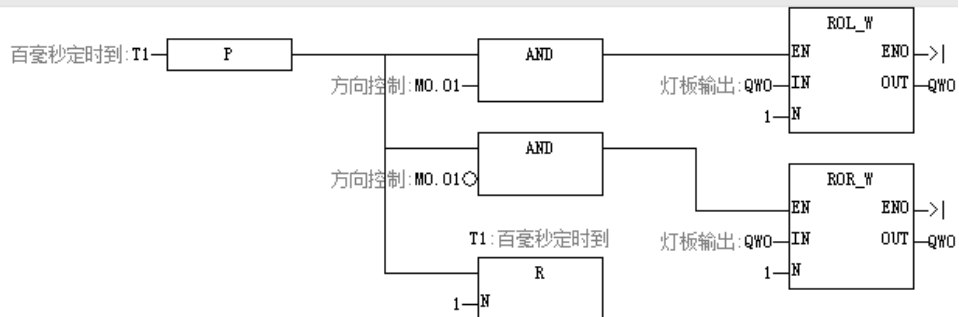
网络 2 M0.0为1的时候，选择100ms定时器



网络 3 M0.0为0的时候，选择1s定时器



网络 4 以100ms为周期，开始循环移位。M0.1为1时，左移；为0时右移。



网络 5 以1s为周期，开始循环移位。M0.1为1时，左移；为0时右移。

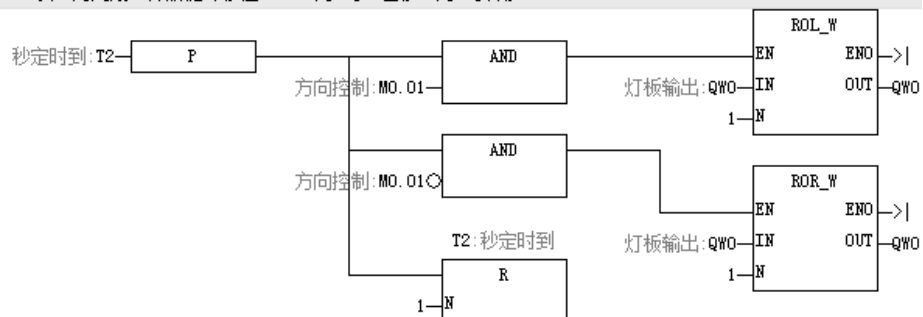


图 2.8.8 跑马灯程序编写示例

程序编写完成后，点击  编译按钮，编译当前工程。软件下方的输出栏会提示出编译结果。

编译无误的程序可直接下载控制器中，点击  下载按钮进行下载操作。



图 2.8.9 下载状态图例

2.8.7 添加符号

用户可将定义好的符号变量写入到软件的状态符号表中，双击工程树中的状态符号表结点进入到状态符号表页面。

序号	选择	符号	绝对地址	Modbus寄存器地址	数据类型
1	☐	采集电流码值	AIW0	0	无符号双字节
2	☐	码值转换	VD100	2436	无符号四字节
3	☐	电流中间值	VD102	2438	四字节浮点数
4	☐	采集电流值	VD104	2440	四字节浮点数
5	☐	采集电压码值	AIW1	1	有符号双字节
6	☐	电压码值转换	VD110	2446	四字节浮点数
7	☐	采集电压值	VD112	2448	四字节浮点数
8	☐				

图 2.8.10 编辑变量符号

2.8.8 在线监视

为了方便对程序进行监控，PLC_Config 提供了在线监视功能。点击 在线监视按钮后程序处于监控状态。

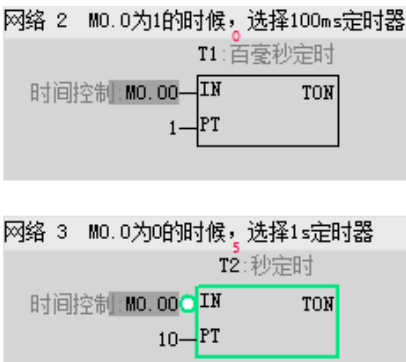


图 2.8.11 程序监视状态图例

如果当前控制器不在线，还可以通过仿真功能模拟控制器运行，点击 仿真按钮，进入在线仿真状态。软件提供控制器的仿真画面，可以观察到灯板状态。

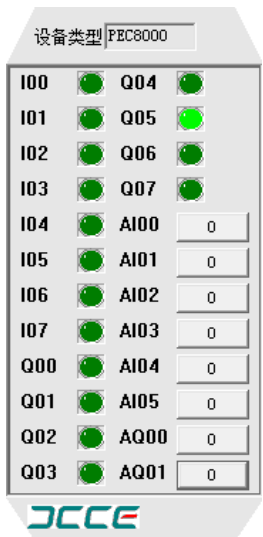


图 2.8.12 仿真控制器灯板

3 内存变量

变量区是为程序提供的一种内存资源，每个变量都有唯一的地址，可通过变量地址直接存取数据，编写程序前需要合理地划分所要使用的变量资源，从而提高内存的利用率和程序执行效率。

3.1 变量区分类

DCCE 控制器定义了 23 个变量区，分别是 I, Q, XI, XQ, V, M, S, SM, L, T, C, AI, PAI, XAI, AQ, PAQ, XAQ, P, EI, EQ, EAI, EAQ 和 E 区，按照访问的数据类型可分为字访问和位访问，按照操作权限可分为只读和读写两种。

变量区的存取支持位、字和双字 3 种数据类型，如表 3.1.1 所示。

表 3.1.1 基本数据类型说明表

数据类型		描述	位数
位	BOOL	位变量	1
字	WORD	无符号整型数	16
	INT	有符号整型数	16
双字	DWORD	无符号长整型数	32
	DINT	有符号长整型数	32
	REAL	实数	32

控制器的内存变量以字为单位，地址连续排列。不支持字节变量（一个字节由 8 个位数据组成，其中第 0 位为最低位，第 7 位为最高位），如果使用字节变量，也要存储在字变量中。可以 1 个字节变量存储于 1 个字变量中，仅使用字变量的低 8 位；也可以 2 个字节变量存储于 1 个字变量中，分别存于字变量高低字节中。

各种变量类型定义如下：

(1) **字变量**：区标识符 + W + 字地址

字变量包括有符号整型数，无符号整型数两种类型，范围分别是-32768~32767 和 0~65535，字变量主要用于模拟量输入输出和数据处理，支持数据传送、算数运算、逻辑运算、定时计数、程序控制、循环移位等操作。

其中，区标识符如表 3.1.2 所示，分别是 V, S, L, M, I, Q 等。W 表示字变量。字地址表示变量存储位置，从 0 开始。例如：VW3，表示变量定义在 V 区中地址为 3 的区域中。

(2) **双字变量**：区标识符 + D + 字地址

双字变量包括有符号长整型数，无符号长整型数和实数三种类型，每种变量表示范围分别是-2147483648~2147483647、0~4294967295 和-3.402823E+38~3.402823E+38。有符号和无符号双字变量和字变量用法相同，主要用于 32 位的整型运算和数据处理。实数类型的双字变量主要用于数值范围较宽的运算和处理，如实数运算、PID 控制等。

其中，区标识符与字变量定义相同。D 表示变量类型是双字变量。字地址表示该双字变量存储的位置。双字变量占用 2 个字变量，分别是字地址指定的字变量和下一个字地址指定的字变量，双字变量低位存于低地址，高位存于高地址。可以通过字变量存取双字变量的高位和低位。例如：VD4，表示双字变量存于字变量 4 和 5 中，VW4 和 VW5 分别存取该双字变量的低位和高位。

(3) **位变量**：区标识符+字地址 + “.” + 位号

位变量的数据类型为 BOOL（布尔）型，仅有 0 和 1 两个值，主要用于能流控制、位逻辑运算、使能、状态位存储和数字量输入输出的处理等。

区标识符与字变量定义相同。字地址与字变量定义相同。位号是字变量中的该位的位置，范围是 0~15。例如：V3.11 表示是 V 区字变量地址为 3，位偏移为 11 的位变量操作数，如图 3.1.1 所示。

字变量和双字变量里面存储的数据类型由处理程序定义，控制器中提供了对有符号整型数，无符号整型数，有符号长整型数，无符号长整型数，实数的处理指令。

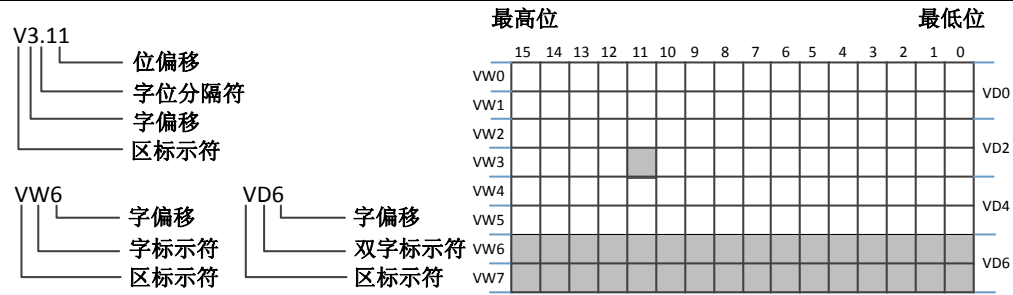


图 3.1.1 变量寻址示意图

不同控制器变量区的定义和功能基本相同，大小略有不同。表 3.1.2 为的变量区功能说明表。

表 3.1.2 变量区功能说明表

标示符	名称	描述	区大小
I	开关输入	开关量输入寄存器，程序运行周期开始刷新 I 区	4 字
Q	开关输出	开关量输出寄存器，程序运行周期结束刷新 Q 区	4 字
V	用户变量	程序主要变量区，前 1925 个字有掉电保持功能，用户可以通过 PLC_Config 进行配置	65535 字
M	内存变量	同 V 区用法相同，后 123 个字有掉电保持功能	512 字
S	顺序控制	以位变量应用为主，主要用于程序的顺序控制	16 字
L	局部变量	以位变量应用为主，作为头参数存放指令当前状态	16 字
SM	特殊功能	具有特殊功能，存放设备参数，系统参数和特殊指令参数	1024 字
AI	模拟输入	模拟输入数值，二进制有符号数	16 字
AQ	模拟输出	模拟输出数值，二进制有符号数	16 字
XI	扩展输入	通过网络映射的从设备开关量输入值	256 字
XQ	扩展输出	通过网络映射的从设备开关量输出值	256 字
PAI	扩展模拟输入	通过网络映射的从设备模拟量输入整数值	256 字
XAI	变换模拟输入	PAI 值经标度变换生成的浮点数	512 字
PAQ	扩展模拟输出	通过网络映射的从设备模拟量输出整数值	256 字
XAQ	变换模拟输出	PAQ 值经标度变换生成的浮点数	512 字
T	定时器	位变量表示时间到否，字变量表示定时器的过程值	1024 个
C	计数器	位变量表示计数到否，字变量表示当前计数值	128 个
P	永久保存	存储在 FLASH 中，用于保存初始值或用户设定值	65535 字
EI	扩展数字输入	扩展设备数字量输入映射区	32 字
EQ	扩展数字输出	扩展设备数字量输出映射区	32 字
EAI	扩展模拟输入	扩展设备模拟量输入映射区	192 字
EAQ	扩展模拟输出	扩展设备模拟量输出映射区	128 字
E	扩展数据变量	存储在外扩 RAM 中，用法与 V 区相同	1048576 字

3.2 变量使用方式

变量主要由指令处理，以实现各种运算和控制功能，也可以通过网络与其他设备共享。因此，变量访问方式可分为指令读写，RS-485 读写和以太网读写三种。

指令读写：指令输入可以是常量也可以是变量，但输出必须是变量，用于存放指令执行结果。根据指令要求的变量类型，选择相应变量。通过指令的输入、输出引脚读写变量，实现各种逻辑控制。

RS-485 读写：RS-485 总线以主从方式组成网络，每个从设备都有唯一地址。控制器支持 Modbus 协议、Profibus-DP、MPI 和自由通信协议。可以通过 Modbus 协议对从设备变量区进行读写，实现主从设备之间的变量映射。

以太网读写：以太网是对等网络，控制器通过以太网读写指令实现设备之间变量映射。以太网读写指令支持 EPA、UDP、TCP 以及 ModbusTCP/UDP 等协议。编程软件使用 EPA 协议管理网络中的设备，完成控制程序开发。

3.3 变量存储方式

控制器的不同变量区存储方式不同，使用前请先确认所需功能，再选择适合的变量区。

- 电池保持区（M、V 区）：

M 区后 123 个字（MW389~MW511）掉电保持，默认为全保持。V 区前 1925 个字（VW0~VW1924）具有掉电保持功能，默认不保持，用户可以在 PLC_Config 控制器参数中设置并分配 V 区保持区域。当供电电池电量不足时，请用户更换电池，否则保持的数据将全部清零。控制器配有电池开关，用户可根据需要选择是否开启电池保持功能。由于电池的电量有限，从电池开启算起，M、V 区保持可持续 1 年左右的时间。

控制器支持 V 区保持功能在编程界面中选择“控制器→控制器参数→设置 V 区保持”对 V 区保持进行配置。每一段保持区包括起始地址和设置长度两个配置选项，用户可自由配置所要保持的数据起始地址和长度。如图 3.3.1 所示，设置完成后点击下载按钮，完成对 V 区保持的设置工作。

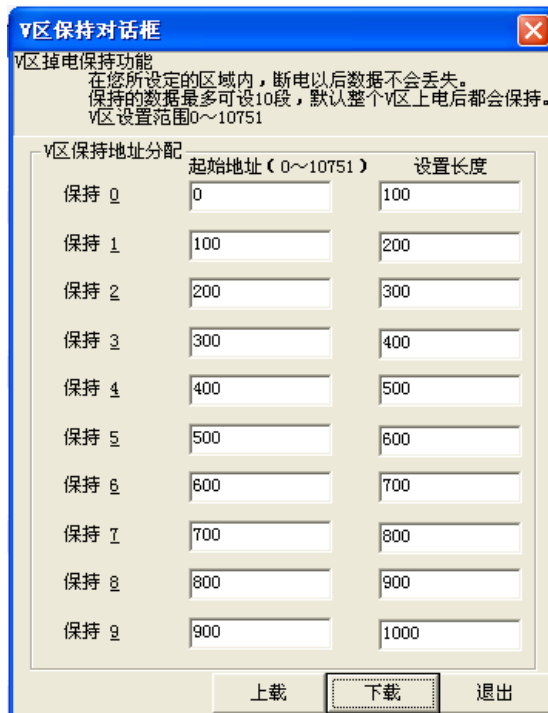


图 3.3.1 V 区数据保持

- 永久保持区（P 区）：

P 区可以用作重要数据的永久保存，可存储程序执行中不变的数据。P 区位于控制器的 FLASH 中，数据掉电不丢失，支持位、字和双字操作。由于 FLASH 芯片的擦写寿命约 10 万次，故修改 P 区数据后，需要带电等待 10 分钟后才能生效，否则重新上电后数据保持不变，或擦除过程中断电导致数据丢失。设计时也对 P 区进行了特殊保护，P 区变量不允许放置在指令的输出引脚。可通过 PLC_Config 软件或操作面板通过网络协议对 P 区进行读写操作，目前不支持 Modbus 协议位命令。

- 特殊功能变量区（SM 区）：

控制器使用 SM 区开放了系统的一些功能，提供了和用户程序之间信息交互的方法，可以通过 SM 区使用一些特殊功能，如：首次扫描为 1 标志和运行状态等。SM 区的前 256 个字变量是只读的，后 256 个字变量和 P 区相同，位于控制器的 FLASH 中，具有永久保存能力。SM 区提供了控制器的厂家参数、设备参数、指令参数、通信参数和系统参数等。

注意：由于 PID 指令的自整定参数需要掉电保持，占用了 SM 区后 256 个字变量区的部分区域，故 SM 区中可供用户自定义使用的只有 SMW347~SMW511。

- 扩展数据变量区（E 区）：

E 区作为扩展数据变量区，与 V 区的使用方法一致，用于存储程序运行过程中的变量和中间结果。E 区

大小为 1M 字，可用于存储大量的扩展数据。由于 E 区位于控制器的外扩 RAM 中，数据掉电丢失，所以只能作为临时数据存储区，不能作为数据永久保存区。E 区不支持 Modbus 寻址。

- 其他变量区：

其他变量区，可保存过程中数据，掉电丢失，用于存储实时数据、控制逻辑、物理输入输出等。

3.4 变量寻址方式

变量区的寻址方式可分为直接寻址和间接寻址两种，直接寻址可对位、字和双字寻址。间接寻址只能寻址字和双字变量。

(1) 变量的直接寻址：

直接寻址方式是用户直接访问各变量区，支持对位、字和双字操作。

- I：数字量输入变量区寻址

用于存储物理数字量输入点，在程序扫描运行周期的开始刷新 I 区。支持按位，字，双字来存取 I 区。

表 3.4.1 I 区寻址

寻址方式	寻址格式	举例
位	I[起始字地址].[位地址]	I0.0
字	IW[起始字地址]	IW1
双字	ID[起始字地址]	ID2

- Q：数字量输出变量区寻址

用于存储物理数字量输出点，在程序扫描运行周期的结尾刷新 Q 区。支持按位，字，双字来存取 Q 区。

表 3.4.2 Q 区寻址

寻址方式	寻址格式	举例
位	Q[起始字地址].[位地址]	Q0.0
字	QW[起始字地址]	QW1
双字	QD[起始字地址]	QD2

- XI：扩展输入映像变量区寻址

用于存储扩展设备的物理数字量输入点，程序扫描运行周期的开始刷新 XI 区。支持按位，字，双字来存取 XI 区。

表 3.4.3 XI 区寻址

寻址方式	寻址格式	举例
位	XI[起始字地址].[位地址]	XI0.0
字	XIW[起始字地址]	XIW4
双字	XID[起始字地址]	XID2

- XQ：扩展输出映像变量区寻址

用于存储扩展设备的物理数字量输出点，程序扫描运行周期的结尾刷新 XQ 区。支持按位，字，双字来存取 XQ 区。

表 3.4.4 XQ 区寻址

寻址方式	寻址格式	举例
位	XQ[起始字地址].[位地址]	XQ0.0
字	XQW[起始字地址]	XQW4
双字	XQD[起始字地址]	XQD2

- V：用户变量区寻址

用于存储程序运行过程中控制逻辑操作的中间结果，如工序，任务等数据，V 区带有掉电保持功能，默认为全保持。支持按位，字，双字来存取。

表 3.4.5 V 区寻址

寻址方式	寻址格式	举例
位	V[起始字地址].[位地址]	V0.0
字	VW[起始字地址]	VW4
双字	VD[起始字地址]	VD2

- M：位变量区寻址

用于存储程序运行过程中数据存储的中间操作状态或者其他的控制信息，M 区掉电丢失，建议位控制变量选用 M 区。尽管名称为位存储区，但是仍然按位，字，双字来存取。

表 3.4.6 M 区寻址

寻址方式	寻址格式	举例
位	M[起始字地址].[位地址]	M0.0
字	MW[起始字地址]	MW4
双字	MD[起始字地址]	MD2

- S：顺序控制变量区寻址

用于组织机器操作等，目前功能和 V 区保持一致，但掉电丢失。支持按位，字，双字来存取。

表 3.4.7 S 区寻址

寻址方式	寻址格式	举例
位	S[起始字地址].[位地址]	S0.0
字	SW[起始字地址]	SW1
双字	SD[起始字地址]	SD2

- SM：特殊功能变量区寻址

用于传递控制器和用户程序之间控制信息，使用 SM 区选择和控制器的一些特殊功能。支持按位，字，双字来存取。有关特殊功能变量区的详细信息请参见附录 B 特殊功能变量区（SM）。

表 3.4.8 SM 区寻址

寻址方式	寻址格式	举例
位	SM[起始字地址].[位地址]	SM0.0
字	SMW[起始字地址]	SMW4
双字	SMD[起始字地址]	SMD2

- L：局部变量区寻址

用于暂存参数。通常的上升沿、下降沿触发器或部分指令头顶参数可以用该区来寻址。支持按位，字，双字来存取。

表 3.4.9 L 区寻址

寻址方式	寻址格式	举例
位	L[起始字地址].[位地址]	L0.0
字	LW[起始字地址]	LW4
双字	LD[起始字地址]	LD2

- T：定时器变量区寻址

用于累计时间增量。定时器精度（时基增量）是 100ms、10ms 和 1ms。控制器提供了 3 种类型的定时器：接通定时器、断开定时器和保持接通定时器。

定时器位和当前计数值的读取依赖于所用的指令，位操作相关指令可读取定时器位，字操作相关指令可读取定时器当前计数值。如图 3.4.1 所示，AND 指令读取的是定时器位，而 MOV_W 指令读取的是定时器当前计数值。

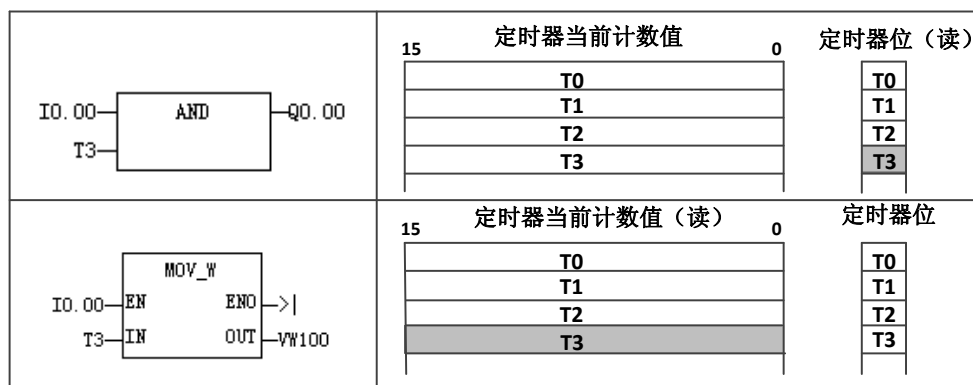


图 3.4.1 定时器区寻址

- C: 计数器变量区寻址

计数器用来累计输入端脉冲从低到高变化次数，控制器提供了 3 种类型的计数器：增计数器，减计数器和增减计数器。

计数器位或者当前计数值的读取依赖于所用的指令，位操作相关的指令可读取计数器位，字操作相关指令可读取计数器当前计数值。如图 3.4.2 所示，AND 指令读取的是计数器位，MOV_W 指令读取的是计数器当前计数值。

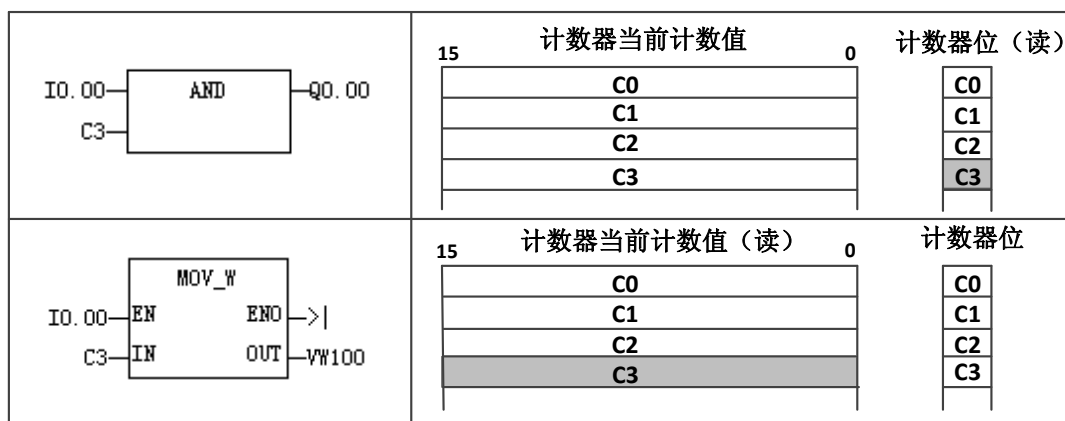


图 3.4.2 计数器区寻址

- AI: 模拟输入变量区寻址

用于存储物理模拟量输入数值，程序运行扫描周期的开始刷新 AI 区。模拟量输入为只读数据，支持按字来存取。

表 3.4.10 AI 区寻址

寻址方式	寻址格式	举例
字	AIW[起始字地址]	AIW1

- PAI: 扩展模拟量输入变量区寻址

用于存储扩展设备的物理模拟量输入数值，程序运行扫描周期的开始刷新 PAI 区。支持按位，字，双字来存取 PAI 区。每个 PAIW 字对应一个 XAID 双字，以此类推。例如：PAIW1 对应 XAID2，PAIW255 对应 XAID510。

表 3.4.11 PAI 区寻址

寻址方式	寻址格式	举例
位	PAI[起始字地址].[位地址]	PAIW0.0
字	PAIW[起始字地址]	PAIW4
双字	PAID[起始字地址]	PAID2

- XAI：校准后扩展模拟量输入变量区寻址

用于存储经过工程校准值转换后的扩展设备的物理模拟量输入数值，XAI 区采用双字存储，数据类型为浮点数，且起始地址只能为偶地址才有意义。

表 3.4.12 XAI 区寻址

寻址方式	寻址格式	举例
双字	XAID[起始字地址]	XAID0, XAID2

- AQ：模拟输出变量区寻址

用于存储物理模拟量输出数值，程序扫描运行周期的结尾刷新 AQ 区。支持按字，双字来存取变量区。

表 3.4.13 AQ 区寻址

寻址方式	寻址格式	举例
字	AQW[起始字节地址]	AQW4
双字	AQD[起始字节地址]	AQD2

- PAQ：扩展模拟量输出变量区寻址

用于存储扩展设备的物理模拟量输出数值，在程序扫描运行周期的结尾刷新 PAQ 区。支持按位，字，双字来存取变量区。每个 PAQW 字对应一个 XAQD 双字，以此类推。例如：PAQW1 对应 XAQD2, PAQW255 对应 XAQD510。

表 3.4.14 PAQ 区寻址

寻址方式	寻址格式	举例
位	PAQ[起始字地址].[位地址]	PAQ0.0
字	PAQW[起始字地址]	PAQW4
双字	PAQD[起始字地址]	PAQD2

- XAQ：校准后扩展模拟量输出变量区寻址

用于存储经过工程校准值转换后的扩展设备的物理模拟量输出数值，XAQ 区采用双字存储。只能按浮点来存取变量区，且起始地址只能为偶地址才有意义。

表 3.4.15 XAQ 区寻址

寻址方式	寻址格式	举例
浮点	XAQD[起始字地址]	XAQD2

- P：永久保存变量区寻址

用于保存配方等需要永久保存的数据。支持可按位，字，双字来存取，写该变量区需要操作 FLASH，故建议不要频繁擦写。

表 3.4.16 P 区寻址

寻址方式	寻址格式	举例
位	P[起始字地址].[位地址]	P0.0
字	PW[起始字地址]	PW0
双字	PD[起始字地址]	PD0

- 常数

指令中除支持上述变量区外，同时支持使用常数。常数可以是字或双字，但无法指定常数为无符号整数、有符号整数或长整数来存储，控制器以二进制方式存取常数。在 PLC_Config 中，常数可以用十进制或十六进制格式来输入或显示，具体格式如下：

十进制：[十进制值]或 10#[十进制值]；

十六进制：16#[十六进制值]；

表 3.4.17 是常数表示方法举例：

表 3.4.17 常数表示方法举例

数据	表示方法
1000	1000 或 10#1000 或 16#3E8
1.2345	10#1.2345 或 1.234
-300.1	10#-300.1 或-300.1

控制器不支持数据类型的检查。举例来说，减法指令可以把 VW100 的值作为一个有符号整数使用，而一条变量赋值指令把 VW100 中的数据作为一个无符号整数使用。故，在 PLC_Config 中变量区的数据类型由使用它的指令决定。

- 本地 I/O 和扩展 I/O 寻址

不同类型的控制器具有不同的 I/O 类型和数量，统一遵循的规则如下：

- 数字量输入对应 I 变量区，从 I0.0 开始，I0.1，I0.2 等依次递增；
- 数字量输出对应 Q 变量区，从 Q0.0 开始，Q0.1，Q0.2 等依次递增；
- 模拟量输入对应 AI 变量区，从 AIW0 开始，AIW1，AIW2 等依次递增；
- 模拟量输出对应 AQ 变量区，从 AQW0 开始，AQW1，AQW2 等依次递增。

扩展 I/O 的寻址与控制器管理的扩展设备相关，每个控制器能管理的最大扩展设备数量为 16 个。在简单模式下，对每个扩展设备管理的开关量输入、开关量输出、模拟量输入和模拟量输出数量和映射变量区都是固定的，映射示意图如表 3.4.18 所示：

表 3.4.18 扩展设备变量映射示意

控制器内部变量	扩展设备变量
XI（扩展设备映射）	扩展设备 DI
XQ（扩展设备映射）	扩展设备 DQ
PAI（扩展设备映射）	扩展设备 AI
XAI（扩展设备映射）	
PAQ（扩展设备映射）	扩展设备 AQ
XAQ（扩展设备映射）	

每个扩展设备在控制器中映射的变量按照表 3.4.19 寻址标准划分。

表 3.4.19 扩展设备映射变量

索引	映射变量区范围			
	DI	DQ	AI	AQ
0 号扩展设备	XI0.0～ XI15.15	XQ0.0～ XQ15.15	PAIW0～PAIW15 XAID0～XAID31	PAQW0～PAQW15 XAQD0～XAQD31
1 号扩展设备	XI16.0～ XI31.15	XQ16.0～ XQ31.15	PAIW16～PAIW31 XAID32～XAID63	PAQW16～PAQW31 XAQD32～XAQD63
.....				
15 号扩展设备	XI240.0～ XI255.15	XQ240.0～ XQ255.15	PAIW240～PAIW255 XAID480～XAID511	PAQW240～PAQW255 XAQD480～XAQD511

提示：如果控制器内部无相应的 I/O 类型，则相应的变量区用作中间变量区使用；扩展设备管理相关的变量信息请参见第 5 章节系统管理部分。

- 变量的间接寻址

除直接使用变量地址外，控制器还支持变量的间接寻址，间接寻址使用指针来存取变量中的数据，只能使用 V 区和 M 区定义指针变量，同时只能寻址 V 区和 M 区的变量。间接寻址中，指针变量为 32 位的整型数值，存储在双字变量中，用户可以通过修改指针实现对数据块变量区中的各个变量的访问，但不支持位变量寻址。

注意：建议尽量少使用指针变量开发程序，或者使用指针变量时在程序中做越界检查，以免指针变量改变了其他变量而影响程序的正常运行。

- 建立指针：

通过变量地址的指针操作，可以实现变量的间接寻址。指针为双字值，存放变量的地址，为了生成指针必须使用双字传送指令(MOV_DW)，将变量地址移到指针变量中，如图 3.4.3，将变量 MW500 的地址赋值给指针变量 MD0。使用“&”符号表示取某一变量的地址，而不是它的值。

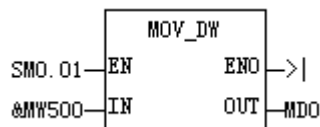


图 3.4.3 建立指针的程序表达式

- 使用指针来存取数据：

操作数前面加上“*”号表示该操作数为一个指针指向的数据。如图 3.4.4 所示 MD0 为指向 MW500 地址的指针，MD0 加 1 后，变为指向 MW501 地址的指针，*MD0 则代表 MD0 所指向 MW501 地址中的数据。输出操作数也可以用指针存取。

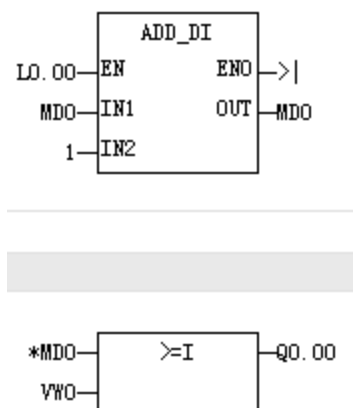


图 3.4.4 使用指针存取数据

- 修改指针：

指针为 32 位的值，所以采用双字指令来修改指针的值，例如双字加减指令。

提示：使用指针时要根据访问数据的长度不同调整指针的步长：当存取字时，使用加法或递增指令使指针加 1；当存取双字时，使用加法或递增指令使指针值加 2。

3.5 特殊功能变量区

SM 区提供了控制器和用户程序之间传递信息的方法，使用 SM 区选择和控制在一些特殊功能。SM 区的前 256 个字变量对外部通信来说是只读的，后 256 个字变量和 P 区相同，位于控制器的 FLASH 中，具有永久保存能力。支持位、字和双字操作。SM 区存储了控制器的厂家参数、资源参数、功能参数、通信参数、IO 配置参数、从设备配置参数和指令状态参数等。

(1) 厂家参数

控制器的厂家参数包括：生产日期、程序版本和控制器类型等系统参数，用户不可修改。

表 3.5.1 控制器厂家参数

SM 区	描述	SM 区	描述
SMW15	设备的 DEVICEID 号	SMD70	控制器生产序号
SMW72	生产日期的年	SMW73	生产日期的月
SMW74	生产日期的日	SMW80	程序副版本号
SMW81	产品扩展功能标志位	SMW124	程序主版本号

(2) 设备参数

控制器为用户提供了以下设备参数：始终和首次扫描为 1 标志、1ms、100ms 和 1s 定时器，秒和分钟脉冲，实时时钟和电池电压值等设备信息，如表 3.5.2 所示。可以使用这些数据控制程序运行。

表 3.5.2 设备参数

SM 区	描述	SM 区	描述
SM0.0	始终为 1	SM0.1	首次扫描为 1
SM0.8	指令加密位	SMW1	每 100ms 加 1
SMW6	秒定时器	SM7.0	秒脉冲
SM7.1	分钟脉冲	SMW52	实时时钟的年
SMW53	实时时钟的月	SMW54	实时时钟的日
SMW55	实时时钟的时	SMW56	实时时钟的分
SMW57	实时时钟的秒	SMW58	实时时钟的星期
SMD167	毫秒定时器	--	--

(3) 功能参数

功能参数部分存放的是永久保存区、电池电压报警、工作模式和用户程序执行状态等参数。

表 3.5.3 功能参数

SM 区	说明	SM 区	说明
SM0.6	用户程序扫描时钟	SM0.4	用户程序运行状态标志
SMW2	硬件拨码开关地址	SMD8	上次程序执行时间
SMD10	程序执行的最短时间	SMD12	程序扫描最长时间
SM14.0	设备参数读取错误	SM14.2	控制参数读取错误
SM14.3	描述参数读取错误	SM14.4	系统参数未初始化错误
SM14.8	下载第一个指令错误	SM14.9	下载最后一个指令错误
SM14.10	下载出现未定义指令	SM14.11	下载写入 Flash 错误标志
SMW47	指令运行时间百分比	*SM59.0	定时器兼容标志
*SM68.0	PEC8K 温度补偿控制路数	SMW79	模块软件地址
*SMW93~100	用户使用的模拟量数值	SMW148	ADC 每一轮的转换时间
*SMW149	时钟同步报文发送间隔	*SM164.1	电池电压报警标示位
*SM164.2	工作模式切换标志位	SMW198	程序中的功能块指令总数
*SMW212	程序延迟运行时间	*SMW253	功能块同步间隔参数
*SMW347~511	用户永久保存区	--	--

提示：标有*号标识的变量可修改，用户可在程序中利用指令（例如 MOV_W）进行设置来配合相关功能使用。

通信参数部分存放的是 Modbus 通信、串口通信和以太网通信的配置和状态参数。

表 3.5.4 通信参数

SM 区	说明	SM 区	说明
SMW16	串口 1 错误处理次数(只读)	SMW17	串口 2 错误处理次数(只读)
SMW40	串口 1 从设备最大等待时间	SMW46	通信实时处理与分时处理
SM67.0	串口通信模式错误	SM67.1	Modbus 功能号溢出标志
SM67.2	Modbus 读通道数量溢出	SM67.3	Modbus 写单个变量溢出
SM67.4	Modbus 写多个变量通道数量溢出	SM67.6	Modbus 扫描从控制器命令串行口号溢出标志
SM67.10	Modbus 转发错误标志	SM67.12	ASCII 到 RTU 转换错误标志
SMW71	串口 1 从设备通信最大错误次数	SMW89	串口 1 作为主口时出错次数
SMW90	串口 2 作为主口时出错次数	SM104.0	自适应状态通信方式
SM104.1	自适应状态下以太网带宽	*SMW158	串口 1 通信收发切换延时
*SMW159	串口 2 通信收发切换延时	*SM165.0~1	以太网通信模式
SM165.2	下以太网通信方式	*SMW179	串行写指令兼容老模式
*SMW181	以太网错误间隔时间	SMW246	串口 2 从设备通信最大错误次数
SMW248	串口 2 从设备通信最大等待时间	SMW247	串口 2 从设备通信时间间隔
*SMW254	设备所处转发状态	SM255.0	转发协议不符
SM255.1	TCP 链接数量超过限制	SM255.2	UDP 链接数量超过限制
SM255.3	同一 IP 地址出现两次链接	--	--

IO 配置参数部分存放 DI、DQ、AI 和 AQ 的配置和状态参数。

表 3.5.5 IO 配置参数

SM 区	说明	SM 区	说明
SMW75	PEC8000 测周法测量频率	SMW76	PEC8000 测周法采样次数
*SMW108	DQ、AQ 屏蔽功能选择	*SMW125	非冻结模式 AQW7 输出值
SMW131	AI0 通道报警上限值	SMW132	AI0 通道报警下限值
SMW133	AI1 通道报警上限值	SMW134	AI1 通道报警下限值
SMW135	AI2 通道报警上限值	SMW136	AI2 通道报警下限值
SMW137	AI3 通道报警上限值	SMW138	AI3 通道报警下限值
SMW139	AI4 通道报警上限值	SMW140	AI4 通道报警下限值
SMW141	AI5 通道报警上限值	SMW142	AI5 通道报警下限值
SMW143	AI6 通道报警上限值	SMW144	AI6 通道报警下限值
SMW145	AI7 通道报警上限值	SMW146	AI7 通道报警下限值
SMW147	AI 通道报警状态	*SM164.0	PEC8000 的 AQ 输出兼容
*SMW169	输出冻结是否使能选择	*SMW170	非冻结模式 QW0 输出值
*SMW171	非冻结模式 QW1 输出值	*SMW172	非冻结模式 AQW0 输出值
*SMW173	非冻结模式 AQW1 输出值	*SMW174	非冻结模式 AQW2 输出值
*SMW175	非冻结模式 AQW3 输出值	*SMW176	非冻结模式 AQW4 输出值
*SMW177	非冻结模式 AQW5 输出值	*SMW178	非冻结模式 AQW6 输出值
SMW194	DQ0 在输出中是否编程	SMW195	DQ1 在输出中是否编程
SMW196	AQW 在输出中是否编程	*SMW213	DI 防抖时间
*SMD214~244	AQ/DQ 读取命令时间间隔	--	--

从设备配置参数部分存放从设备的配置和状态参数。

表 3.5.6 从设备控制参数

SM 区	说明	SM 区	说明
SM0.9	上电后从设备全部扫描完毕为 1	SM14.1	从设备参数读取错误
SMW19	16 个从设备的状态	SMW42	从设备通信模式
SM162.0	从设备转发命令是否使能	SMW126	从设备在线灯点亮(拷机用)
*SMW129	从设备自动恢复扫描控制位	SMW130	从设备自恢复扫描间隔
SMW161	从设备转发命令保留时间	SMW162	从设备转发命令刷新间隔
SMW247	从设备通信时间间隔	--	--

指令状态参数部分存放高速输入、输出的计数状态，WPC 工件检测指令的控制参数、模式控制和使用标志，以太网读、写指令的标志，PID 的自整定参数、状态位和控制参数，自由通信状态和定时器时基控制等需要与指令配合使用的一些特殊功能变量。

表 3.5.7 指令状态参数标志位

SM 区	说明	SM 区	说明
SMW23	HQ0 的 S 曲线最大加速度	*SMW24	HQ0 的 S 曲线起始频率
*SM25.0	写入第一路坐标变量	*SM25.1	写入第二路坐标变量
*SMD26	HQ0 的定位指令坐标值	SMW28	HQ1 的 PLSE 执行结束阈值
*SMW29	飞升曲线法自整定控制位	SMD30	HQ1 已输出的脉冲个数
SMW32	HQ1 的 S 曲线加/减速时间	SMW33	HQ1 的 S 曲线最大加速度
*SMW34	HQ1 的 S 曲线起始频率	*SM35.0	写入第一路坐标变量
*SM35.1	写入第二路坐标变量	*SMD36	HQ1 的定位指令坐标值
SMW38	HQ1 的 PLSE 结束阈值	*SMW39	AREA 起控范围参数使能
SMW43	第一个 WPC 工件长度	SMW44	第二个 WPC 工件长度
SMW45	第一个 WPC 生产线总距离	SMW48	WPC 指令距离判断模式
SMD49	第一个 WPC 距离判断模式速度值	SMW51	PID 自整定错误标识
SM60.0	SR 触发器工作在不定状态	SM60.1	RS 触发器工作在不定状态
SM61.0	调用网络读指令数量溢出	SM61.2	读指令目的设备 IP 冲突
SM61.3	网络读指令发送数据错误	SM61.4	网络读指令响应数据错误

上接表 3.5.8 指令状态参数标志位

SM 区	说明	SM 区	说明
SM62.0	调用网络写指令数量溢出	SM62.2	写指令目的设备 IP 冲突
SM62.3	网络写指令发送数据错误	SM62.4	网络写指令响应数据错误
SM62.0	高速计数器 0 增计数模式大于等于设定值	SM62.1	高速计数器 0 复位标志位
SM62.2	高速计数器 0 计数等于预定值	SM62.3	高速计数器 0 计数大于预定值置位
SM62.4	高速计数器 1 增计数模式大于等于设定值	SM62.5	高速计数器 1 复位标志位
SM62.6	高速计数器 1 计数等于预定值	SM62.7	高速计数器 1 计数大于预定值
*SM62.8	AB 相计数时方向控制位	SM62.10	高速计数器 2 计数大于预定值
SM62.11	高速计数器 2 复位标志位	SM62.14	高速计数器 3 计数大于预定值
SM62.15	高速计数器 3 复位标志位	SM64.1	PID 回路选择越界
SM64.2	PID 功能块调解方式	SM64.3	PID 的微分项设置出错
SM64.4	PID 的积分参数设置出错	SM64.5	PID 所处阶段
SM64.6	PID 的比例度设置出错	SM64.7	PID 的采样时间设置出错
SMD65	第二个 WPC 距离判断模式速度值	SMW69	清除输出值使能标志位
SMW78	第二个 WPC 生产线总距离	SMW82~88	PID 相位偏移设置值
*SMW103	PID 整定完后的变速积分	*SMW105	无过冲自整定输出方式
*SMW106	自整定 P、I、D 参数	*SMW107	PWM 输出时间间隔
*SMW109	自整定时滤波使能位	*SMW110	自整定时滤波次数
*SMW111	PWM 周期自定义模式	*SMD112	自整定时的 Kp 系数
*SMD114	自整定时的 Ti 系数	*SMD116	自整定时的 Td 系数
*SMD118	自整定时的 TD 值	*SMW120	P_PID 公式选择
*SMD150	微分系数	*SMW152~155	P_PID 公式算法选择字
*SM166.0~1	以太网写指令模式选择	SM180.0	子程序调用深度超过 16 层
*SMW182	以太网读指令通信最小间隔	*SMW183	以太网读指令超时时间
*SMW184	以太网写指令通信最小间隔	*SMW185	WPC 第二个指令的使用指示
*SMW186	PS 第二个指令的使用指示	*SMW187	高测周法和测频法分割点
*SMW188	高速计数器 0 滤波次数	*SMW189	高速计数器 1 滤波次数
SMW190	定时 0 中断定时时间	SMW191	定时 1 中断定时时间
SMW192	定时 2 中断定时时间	SMW193	定时 3 中断定时时间
SM199.0	自由通信串口 1 发送完毕信号	SM199.1	自由通信串口 2 发送完毕信号
SM200.1	自由通信接收功能中止	SM200.5	1 表示收到结束字符
SM200.8~15	8~15 位与 0~7 位意义相同	*SM201.1	检测起始接收
*SM201.4	检测空闲状态	*SM201.5	检测结束字符
*SM201.6	检测起始字符	*SM201.8~15	8~15 位与 0~7 位意义相同
*SMW202	自由通信串口信息开始字符	*SMW203	自由通信串口信息结束字符
*SMW208	自由通信串口 1 接收的最大字符数	*SMW209	自由通信串口 2 接收的最大字符数
*SM210.6	串口 1 使用自由通信协议	*SM210.7	串口 2 使用自由通信协议
*SM210.8	自由通信串口 1 启动被动接收模式	*SM210.9	自由通信串口 2 启动被动接收模式
*SM211.0	串口 1 的 Recv 功能块 TBL 没有进行正确的配置	*SM211.2	串口 1 未使能自用通信格式
*SM211.3	自由通信串口 1 不是主口	*SM211.4	自由通信串口 1 启动模式错误
*SM211.5	自由通信串口 1 终止模式错误	*SM211.8	串口 2 的 Recv 功能块 TBL 没有进行正确的配置
*SM211.10	串口 2 未使能自由通信格式	*SM211.11	自由通信串口 2 不是主口
*SM211.12	自由通信串口 2 启动模式错误	*SM211.13	自由通信串口 2 终止模式错误
SMD256	PID0 自整定参数的 PB 值	SMD258	PID0 自整定参数的 TI 值
SMD260	PID0 自整定参数的 TD 值	SMD262	PID0 自整定采样时间值
SMD264~318	PID1~PID7 的自整定参数值	*SMW320	PID 自整定的使能位
*SMW321	PID 是否使用自整定参数	*SMW322	使用自整定参数
SMW323~338	飞升曲线法整定出的斜率	SMW339~346	无过冲自整定百分比值

3.6 变量区操作方式

PLC_Config 利用组态程序中的变量、状态符号表、交叉引用表、趋势图、只读变量区管理表等元件实现对变量的操作、监控和管理，有效的控制程序的运行，掌握控制器运行的最新动态。有关程序元件的具体使用方法详见 2.2 小节。

4 通信方式

控制器拥有以太网和 RS-485 通信接口，兼容 Modbus、USS、EPA、Profibus-DP 等主流控制网络协议。控制器采用多种通信协议实现对等网络和主从设备之间的通信。支持标准的 Modbus 命令号，实现变量区的访问和主从设备通信。可利用 Modbus、Profibus-DP、USS 等串口通信协议与第三方设备进行访问和管理。控制器利用标准的 EPA 协议实现与 PLC_Config 编程软件的通信，完成内部参数配置、程序上下载和变量监控等功能。

4.1 串行通信服务

DCCE 网络化可编程控制器支持 ModbusRTU、ModbusASCII 主站和从站通信，实现从设备配置和资源映射功能。可利用 Profibus-DP 协议作为从站与西门子 S7-300 通信，实现与西门子 S7-300 的 I/O 数据交互。此外，控制器提供了 MPI、USS 指令和自由通信指令，以指令的方式实现和变频器等第三方设备的数据发送和接收。

4.1.1 串口通信配置

PLC_Config 软件支持以太网和串口两种通讯模式，若采用串口通信模式，则需要在软件主界面菜单栏的“工具”中点击软件配置，如图 4.1.1 所示。



图 4.1.1 调出软件配置界面

在软件配置界面中选择通讯配置，如图 4.1.2 所示。点击“启用串口”，其他参数则根据用户需求进行配置，配置好相关参数后，软件串口接收功能即可正常使用。如果不需要使用串口功能，则将“启用串口”选项取消即可。

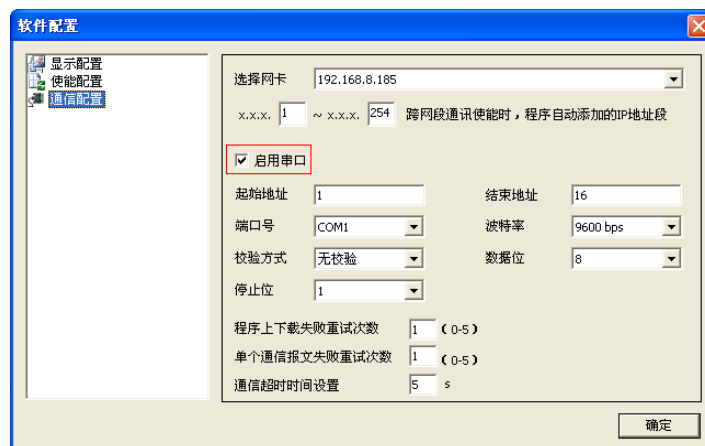


图 4.1.2 软件配置界面

如果软件已配置开启串口功能，正常使用时仅在控制器管理界面中进行配置即可。点击控制器管理界面中的“串口设置”按钮打开对话框。

当已知控制器的端口、校验位、停止位、波特率及数据位信息时，可在“串口设置”对话框中对控制器信息进行配置，同时可以为控制器配置起始与结束地址，点击确定后，该“串口设置”对话框自动关闭，在左侧网络控制器中会按照配置信息添加扫描控制器。若在未知控制器信息的前提下，可以通过点击“串口设置”对话框中的“全部扫描”按钮，PLC_Config 会根据串口配置信息组合情况对当前所有的控制器进行扫描。

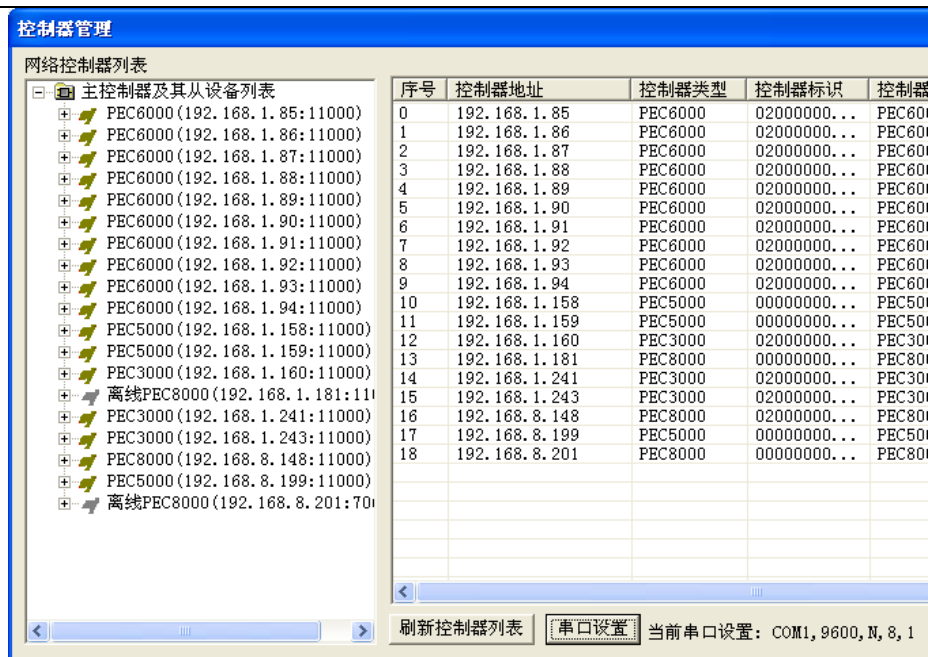


图 4.1.3 控制管理界面

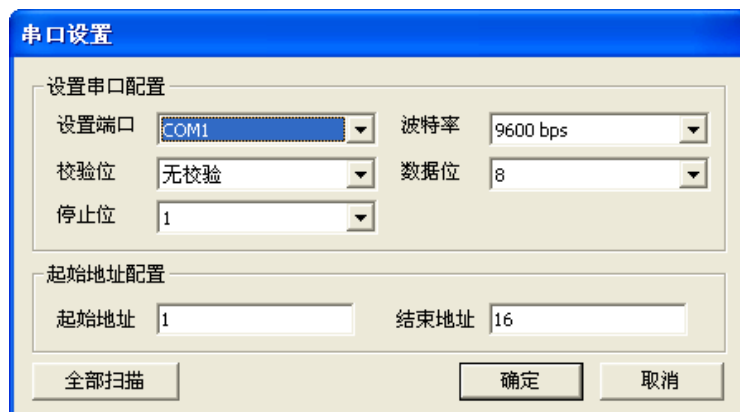


图 4.1.4 串口设置对话框图例

4.1.2 串行通信参数

DCCE 网络化可编程控制器中与串口通信相关的通信参数控制字包括：通信波特率、数据位、停止位、校验方式、通信协议、主从模式和报文传送时间间隔等控制字，使用控制器前需要配置通信参数控制字，可通过 PLC_Config 编辑软件 and 用户版调试软件读取和修改通信参数。

通信协议：可配置成 ModbusRTU 协议，ModbusASCII 协议，Profibus-DP 协议，MPI 协议以及 IO 协议。MAC 系列可编程控制器串口 1 默认为 IO 协议，其他设备默认为 Modbus RTU 协议。

校验方式：包括无校验、奇校验、偶校验、以及 Mark 或 Space 校验。DCCE 网络化可编程控制器的校验方式控制字默认为 0。

通信波特率：可配置为 1200bps、2400bps、4800bps、9600bps、19200bps、38400bps、57600bps、115200bps、500Kbps、1Mbps。DCCE 网络化可编程控制器的通信波特率控制字默认为 115200bps。

数据位：分为 8 位数据位（如果此时有奇偶校验，则第 9 位为奇偶校验位）和 7 位数据位（若数据位为 7 位时，最高位为奇偶校验位）。DCCE 网络化可编程控制器的数据位控制字默认为 8 位数据位。

停止位：包括 1 个停止位模式和 2 个停止位模式（2 个停止位时，奇偶校验无效）。DCCE 网络化可编程控制器的停止位控制字默认为 0。

报文传送时间间隔：串行通信时，串口接收两条报文的最小时间间隔，最大值为 255，单位 ms。DCCE 网络化可编程控制器的报文传送时间间隔控制字默认为 4。

主从模式控制字：将串口配置为从口，此时串口的主要任务是被动接收数据，根据接收到的命令，执行

相应命令。串口配置为主口，设备可以主动向从设备发送数据。DCCE 网络化可编程控制器串口 1 的主从模式控制字默认为 1，串口 2 的主从模式控制字默认为 0。

提示：控制器支持双主口通信。

4.1.3 Modbus 通信

控制器支持 ModbusRTU 和 ModbusASCII 通信协议。控制器支持标准的 Modbus 命令号和变量区 Modbus 地址，实现了变量区的访问和主从设备通信。

● Modbus RTU

RTU 协议以总线上不少于 3.5 个字节传输时间的通信空闲间隔作为数据接收的开始和结束标志，中间的格式按顺序依次为站地址、功能码、数据和 CRC 校验码。主机发送和从机响应相同。

协议格式如下：



注意：其中 1b 代表 1 个字节 (byte)，数据信息部分长度不定。

- 控制器地址为总线上设备的唯一标识；
- 功能码为此次命令的功能标识，指示从设备需要执行哪些行为；
- 数据信息格式与功能码相对应，是功能码的具体执行部分；
- 错误校验是对发送命令正确与否的检验。

● Modbus ASCII

ASCII 协议以冒号“:”为数据帧传输的开始标志；以回车<CR>换行<LF>符为结束标志，对应的 ASCII 码为 0D、0A。中间的格式按顺序依次为站地址、功能码、数据和 LRC 校验码。主机发送和从机响应相同。

协议格式如下：



注意：其中 1b 代表 1 个字节 (byte)，数据信息部分长度不定。

ASCII 格式中，每 1 个字节的高低 8 位数据，都需要用 2 字节的 ASCII 码表示，高 8 位在前，低 8 位在后。

LRC 校验码是累加和校验值，将站地址、功能码和数据的所有 ASCII 码相加进位丢弃，得到的八位二进制数取其负数，再转换成两个 ASCII 码，附在数据串后面。

● Modbus 命令号

控制器支持 1、2、3、4、5、6、15、16、17 号 Modbus 命令号，如表 4.1.1 所示。

表 4.1.1 Modbus 功能码

命令号	实际操作
1	读多个开关量输出状态
2	读多个开关量输入状态
3	读多个系统参数或模拟量输入
4	同 3 号功能
5	强制单个开关量输出
6	设置单个系统参数或模拟量输出
15	强制多个开关量输出
16	设置多个系统参数或模拟量输出
17	读取设备系统信息

● 变量区 Modbus 地址

控制器为各变量区分配了唯一的 Modbus 地址，所有变量区都支持字访问和位访问操作。变量区的 Modbus 地址索引表如表 4.1.2 和表 4.1.3 所示：

表 4.1.2 支持字访问的变量区索引表（支持功能号 3、4、6、16）

变量区名称	内容	字数量	Modbus 地址		操作
			十进制	十六进制	
AI	模拟输入	16	0~15	0000~000F	只读
XAI ^[1]	变换模拟输入	512	16~527	0010~020F	只读
PAI	扩展模拟输入	256	528~783	0210~030F	只读
T	定时器	128	784~911	0310~038F	只读
C	计数器	128	912~1039	0390~040F	只读
SM ^[2]	特殊功能	512	1040~1551	0410~060F	读写
XAQ ^[3]	变换模拟输出	512	1552~2063	0610~080F	读写
AQ	模拟量输出值	16	2064~2079	0810~081F	读写
PAQ	扩展模拟输出	256	2080~2335	0820~091F	读写
V	用户变量	512	2336~2847	0920~0B1F	读写
M	内存变量	512	2848~3359	0B20~0D1F	读写
S	顺序控制	16	3360~3375	0D20~0D2F	读写
L	局部变量	16	3376~3391	0D30~0D3F	读写
V ^[4]	用户扩展变量	4608	3392~8000	0D40~1F40	读写
I	开关输入	4	13632~13635	3540~3543	只读
XI	扩展输入	256	13636~13891	3544~3643	只读
XQ	扩展输出	256	13892~14147	3644~3743	读写
Q	开关输出	4	14148~14151	3744~3747	读写
P	永久保存	4096 或 2048	14152~18247 或 14152~16199	0x3748~0x4747 或 0X3748~0X3F47	读写
RAM	指令 RAM	14504	18248~32751	0x4748~0x7FEF	只读

提示：

[1]XAI 区每个通道的数据类型为浮点型，占 2 个字变量。

[2]对于 SM 区的前 256 个字为只读，后 256 个字作为永久存储区，可读可写，部分作为控制器内部使用。

[3]XAQ 区每个通道的数据类型为浮点型，占 2 个字变量。

[4]用户扩展变量区 V 延续用户变量区的寻址，对应于 VW[512]~VW[5119]。

表 4.1.3 支持位访问的变量索引表（支持功能号 1、2、5、15）

变量区名称	内容	字数量	Modbus 地址		操作
			十进制	十六进制	
I	开关输入	4	0~63	0000~003F	只读
XI	扩展输入	256	64~4159	0040~103F	只读
T	定时器	8	4160~4287	1040~10BF	只读
C	计数器	8	4288~4415	10C0~113F	只读
SM ^[1]	特殊功能	512	4416~12607	1140~313F	读写
XQ	扩展输出	256	12608~16703	3140~413F	读写
Q	开关输出	4	16704~16767	4140~417F	读写
V	用户变量	512	16768~24959	4180~617F	读写
M ^[2]	内存变量	256	24960~29055	6180~717F	读写
S	顺序控制	16	29056~29311	7180~727F	读写
L	局部变量	16	29312~29567	7280~737F	读写

提示：

[1]对于 SM 区的前 256 个字为只读，后 256 个字作为永久存储区，可读可写。

[2]对于 M 区的位变量，由于 Modbus 地址的限制，只支持前 256 个字的位寻址。

- Modbus 从站通信

控制器作为 Modbus 从站时，需将通信串口配置为从口、通信协议配置为 ModbusRTU 或 ModbusASCII。

通信波特率，校验位，停止位，数据位，通信报文时间间隔等参数需和主站配置保持一致。

主从通信，控制器实时响应主站发送的 Modbus 报文。编程软件或其他支持 Modbus 协议的主设备都可通过 Modbus 命令访问控制器的变量。

● Modbus 主站通信

控制器作为 Modbus 主站时，需将通信串口配置为主口、通信协议配置为 ModbusRTU 或 ModbusASCII。通信波特率，校验位，停止位，数据位，通信报文时间间隔等参数需和从站配置保持一致。

主从通信时，控制器可通过串行通信指令访问或控制串行总线上的其他设备，也可作为主站控制器与 DCCE 控制器或其他支持 Modbus 协议的第三方设备构成分布式控制系统。

注意：(1) 进行 Modbus 通信时，控制器地址代表 Modbus 站地址。(2) 控制器与其他设备通信时，Modbus 站地址不能相同。(3) 通过 Modbus 协议进行资源映射时，需要注意数据区大小端模式，DCCE 网络化可编程采用小端模式，西门子设备和昆仑动态触摸屏均采用大端模式，使用时请注意。

4.1.4 Profibus_DP 通信

Profibus 是目前市场占有率较大、应用较广泛的一种数字式、串行、多点的通信协议标准，用于工业控制和仪表设备如传感器、执行机构和控制器之间的数据通信。Profibus 总线数据传输速率最快为 12MBit/s；输入输出字节最多为 244Byte；一条总线上最多支持 126 个工作站；每个工作站可具体分为 32 个段。

Profibus 从站，负责对主站的请求做出响应，不参与总线控制，通过.GSD 文件定义从设备的相关属性。

控制器可通过 Profibus_DP 协议与西门子 S6-300 通信。在使用 Profibus_DP 通信协议时，支持的波特率类型为：9600bps 和 19200bps，需将通信串口配置为从口、通信协议配置为 Profibus_DP，通信参数配置为 8 个数据位，1 个停止位，偶校验。

控制器作为 S7-300 的从设备，可将其变量区的数据映射给 S7-300。用户需要使用 STEP7 和 PLC_Config 软件，实现控制器变量区的映射功能，大工计控为 STEP7 提供了.GSD 文件，描述了控制器实现变量刷新所需要的必要参数。在 STEP7 软件中添加 DCCE.GSD 文件，并通过 PLC_Config 软件配置映射的变量区，实现主从设备之间的资源映射。

DCCE 提供的.GSD 文件支持 9 种映射方式，包括：1 字节、1 字、2 字、3 字、4 字、8 字、12 字和 16 字及 32 字映射方式。

控制器通过 PLC_Config 编程软件配置映射的变量区，可映射的变量区有 12 种，包括：AQ、M、PAQ、Q、S、V、XAQ、AI、PAI、XAI、I、XI，最少支持 1 个字节的映射，最多支持 80 个字的映射。

● 标准的 Profibus 从站运行状态机

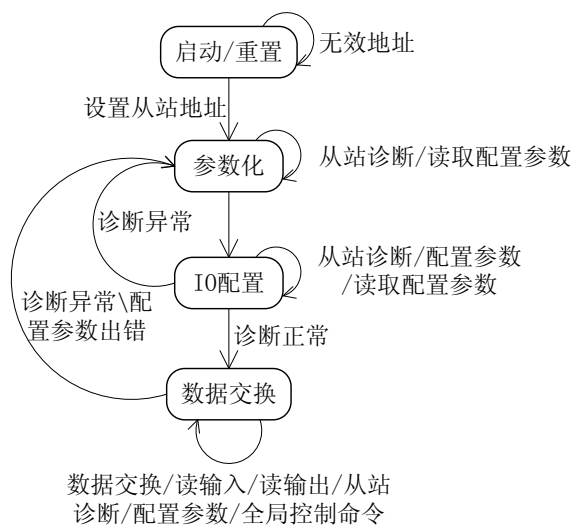


图 4.1.5 Profibus 从站运行状态机

● 报文格式

一条 Profibus 报文最大可以包含 255 个字节，其中包含 11 个字节的报文头，以及 244 个字节的信息。除了数据交换报文的报文头为 9 个字节外，其余所有的 Profibus 报文的报文头都为 11 个字节。

报文格式如下所示：

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
1b	1b	1b	1b	1b	1b	1b	1b	1b	Var	1b	1b

注意：其中 1b 代表 1 个字节（byte），DU 部分长度不定。

表 4.1.4 Profibus 报文参数说明

项	长度（字节）	描述
SD	1	起始标识（用来区分报文格式）
LE	1	网络数据长度：(DU)+DA+SA+FC+DSAP+SSAP
LEr	1	重复长度字节，用于查错。
SD	1	目的地址
DA	1	源地址
FC	1	帧控制（标识当前报文的类型/属性） 例如：请求、确认或响应。
DSAP	1	目的 SAP
SSAP	1	源 SAP
DU	1 ~ 32 或 1 ~ 244	数据单元，网络传输数据
FCS	1	帧校验字节（报文中指定长度数据的 ASCII 字节和）
ED	1	结束标识（通常是 16H）

Profibus 的主站和从站之间的数据通信使用 SAP（Service Access Points）来进行。

● Profibus 配置实例

(1) 打开 STEP 7 软件，新建 S6-300 Station 工程。选择 Hardware，在弹出的 HW_Config 界面中添加 DCCE.GSD 文件。

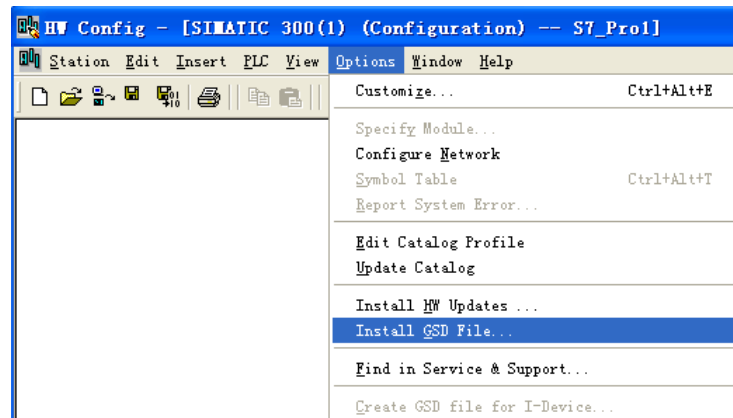


图 4.1.6 STEP7 软件配置

当 GSD 文件添加完成之后，依次点击右侧“PROFIBUS DP -> Additional Field Devices -> I/O -> DCCE->PEC”。在 PEC 下拉列表中，用户可以选择 9 种 IO 映射方式，如图 4.1.7 所示。

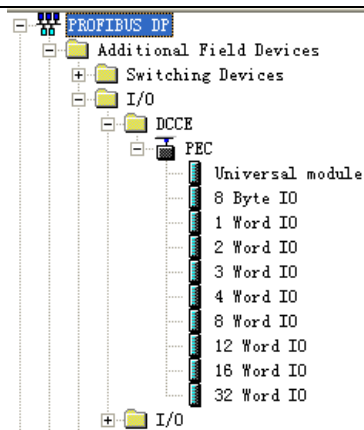


图 4.1.7 STEP7 软件配置

在硬件组态中添加 S6-300 设备, 控制器地址为 1, 通信波特率为 9600bps。在组态中添加一个 PROFIBUS DP 总线, 选中 PROFIBUS 线, 添加 PEC 设备, 作为 S7-300 设备的从设备, 添加后如图 4.1.8 所示。

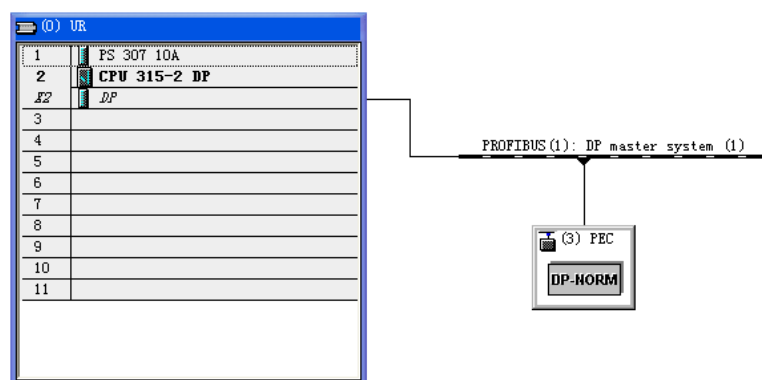


图 4.1.8 STEP7 软件配置

双击 PEC 图标, 将 PROFIBUS 地址设置为 3, 波特率设置为 9600bps; 根据需要配置 IO, IO 配置如图所示。配置完成后点击 Download 按钮, 将程序下载到指定的控制器中。

(3) PEC				
S...	DP ID	Order Number / Designation	I Address	Q Address
1	115	4 Word IO	256...263	256...263
2				

图 4.1.9 STEP7 软件配置

打开 PLC_CONFIG 组态软件, 新建工程。将需要与西门子通信的设备添加到工程中, 在设备参数界面中配置串口通信参数。具体配置信息如图 4.1.10。

主从设置	从
通信协议	Profibus
通信波特率	9600 bps
通讯基本设置	偶, 8, 1
数据包间隔时间 (ms)	4

图 4.1.10 串口参数配置

(2) 设置 Profibus 参数, 配置映射变量区。如图 4.1.11 所示, 将 S6-300 设备的 Q256~Q263 变量区中的数据映射到 PEC6000 的 MW0~MW3 中, 并将控制器的 MW10~MW13 变量区中的数据映射到 S6-300 设备的 I256~I263 中。配置完毕后, 点击下载按钮, 将配置信息下载至 PEC6000 中。



图 4.1.11 配置映射变量区

(3) 将控制器硬件地址设置为所需要的 PROFIBUS ID, S7-300 的 DP 端口默认 ID 为 2。PROFIBUS 协议控制器硬件地址大于 2, 例程设置为 3。

4.2 以太网通信服务

控制器以太网支持标准的 UDP/TCP 自由通信协议、EPA 通信协议和 ModbusTCP/ModbusUDP 通信协议。

控制器的 EPA 通信端口号默认为 11000, 支持 10M、100M 和自适应 3 种以太网通信速率配置模式, 支持全双工和半双工 2 种工作方式。用户可通过用户版调试软件或 SM 区进行上述参数配置。通过 SMW165 的第 0 位和第 1 位配置以太网通信速率, SM165.0~SM165.1 为 0 表示为 10M 模式, 为 1 表示为 100M 模式, 为 2 表示自适应模式, 默认为自适应模式。

当控制器处于非自适应模式时, 可通过 SM165.2 配置工作方式, SM165.2 为 1 表示全双工, 为 0 表示半双工。控制器处于自适应模式时, 可通过以太网状态位 SM104.0 和 SM104.1 查看以太网工作状态, SM104.0 为 0 表示工作方式为半双工, 为 1 表示工作方式为全双工; SM104.1 为 0 表示通信速率为 10M, 为 1 表示通信速率为 100M。

提示: 双网口控制器增加了两个以太网工作状态位, 分别是 SM104.2 和 SM104.3。具体表示方法见特殊变量区说明。

4.2.1 以太网通信配置

使用以太网通信模式时, 需要为控制器设置一个 IP 地址, 默认为 192.168.1.181。用户可以通过调试软件对设备的 IP 地址进行设置, 当进行 IP 地址的设置时, 需要先确认网络中该 IP 地址不会被其他设备所使用, 否则会导致更改后的设备无法上线。

以太网通信模式中, 使用设置通信接口功能, 可以改变当前工程控制器所对应的 IP 地址或域名地址, 从而实现快速将当前程序下载到不同地址的控制器中。

用户可以在工程树中设置工程设备的通信接口。选择需要修改通信接口的控制器, 点击鼠标右键, 在右键菜单中选择“设置通信接口”, 在弹出的控制器地址对话框中填写控制器信息。如图 4.2.1 所示, 可通过修改域名或 IP 地址来修改控制器地址。

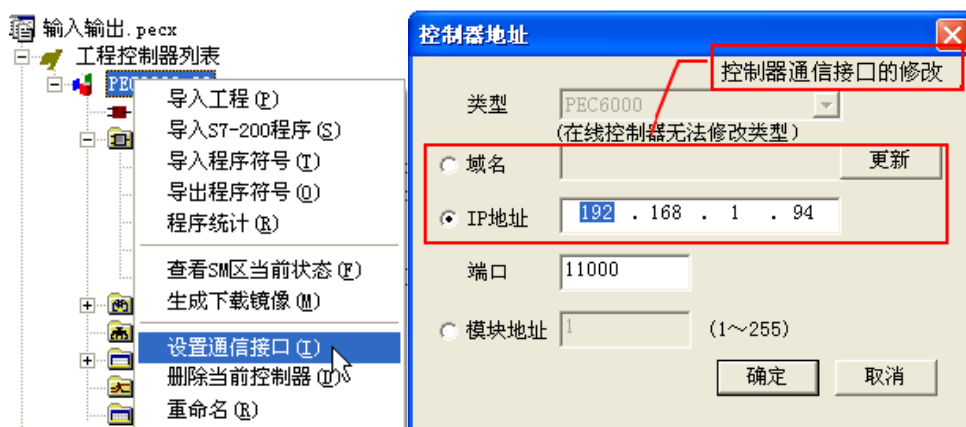


图 4.2.1 工程树通信配置

同时, 以太网通讯时, 要注意网关的配置。详细网关配置参见 4.2.2 以太网自由通信中的同网段和跨网段网关配置部分。

4.2.2 以太网自由通信

以太网自由通信报文带有 UDP 或 TCP 报文头，数据段用户自定义。控制器使用 UDP 和 TCP 自由通信指令实现以太网自由通信，调用自由通信指令时，通过指令引脚配置目的 IP 和目的端口号等参数。

● UDP 自由通信

控制器提供的 UDP 自由通信指令包括：NXMT_RCV，NXMT_SEND，NET_XMT。通过上述指令可实现同网段或跨网段的 UDP 通信。

➤ 同网段 UDP 通信

当使用 UDP 自由通信指令进行同一网段内的 UDP 通信时，必须保证进行通信的设备具有相同的网络号。例如：设备 A 的 IP 地址网络为 192.168.8.123，则与其通信设备的 IP 地址应设置为 192.168.8.xxx（IP 尾号范围为 1~254）。

➤ 跨网段 UDP 通信

当使用 UDP 自由通信指令进行跨网段通信时，需要对网关进行配置，以图 4.2.2 中的网络结构为例，A 设备通过路由器可以跨网关向 B 设备和 PC 机发送数据报文，实现跨网段通信。

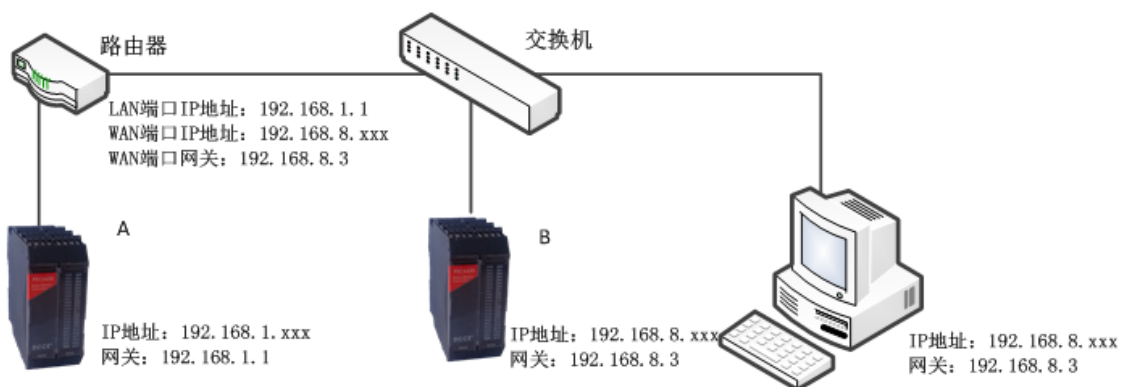


图 4.2.2 跨网段通信

➤ 应用示例：

调用 NET_XMT 指令，配置 IP 引脚为目的端 IP 地址、PORT 引脚存储目的端端口号。该指令上升沿触发。

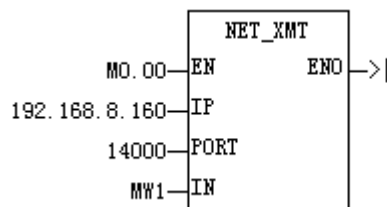


图 4.2.3 UDP 自由通信指令

NET_XMT 指令的目的端可以是 PC 机，也可以是控制器。

● TCP 自由通信

DCCE 网络化可编程控制器提供的 TCP 自由通信指令包括：TCP_LINK、TCP_SEND 和 TCP_RCV。控制器通过 TCP_LINK 指令与同一网络上的其他设备建立 TCP 连接，控制器最多可支持 2 个连接。

➤ 应用示例：

调用 TCP_LINK 指令建立 TCP 连接，与 IP 为 192.168.8.160 的 PC 机建立 TCP 连接，连接建立成功时 LINK 引脚置 1。ID 引脚表示 TCP 连接编号。在同一工程中最多可用两个 TCP_LINK 指令。

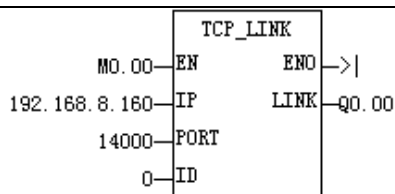
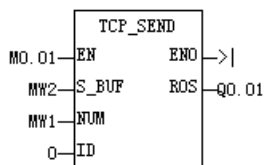


图 4.2.4 建立 TCP 连接

调用 TCP_SEND 或 TCP_RCV 指令在已建立好的 TCP 链接上完成数据的收发功能。ID 引脚表示接收和发送数据的 TCP 连接编号。



§ 3 注释...

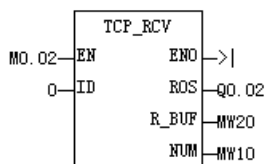


图 4.2.5 发送接收数据

4.2.3 EPA 协议通信

DCCE 网络化可编程控制器使用标准的 EPA 协议实现与 PLC_Config 编程软件的通信，完成内部参数配置、程序上下载和变量监控等功能。

- EPA 报文

EPA 报文头由 8 部分组成，表 4.2.1 描述了各部分含义。控制器提供了读写变量区的报文格式，可利用 EPA 变量读写报文访问各变量区，实现对控制器的监控。

表 4.2.1 EPA 报文头

参数名	数据类型	偏移 (字节)	字节长度	描述
ServiceID	Unsigned8	0	1	报文类别码
Reserved	Ocetstring	1	3	保留
Length	Unsigned16	4	2	报文长度
MessageID	Unsigned16	6	2	消息号
SourceAppID	Unsigned16	8	2	源功能块标识
DestinationAppID	Unsigned16	10	2	变量区标识
DestinationObjectID	Unsigned16	12	2	目的对象标识
SubIndex	Unsigned16	14	2	对象的子索引

提示：EPA 协议支持 14 个报文，为了支持多级设备转发，定义了 FEP 协议。

- 变量区读

通过 EPA 协议可访问 18 个变量区，每个变量区都支持单个变量、多个变量和区变量（512 个字节）的读操作，对应的变量区标识如表 4.2.2 所示。

表 4.2.2 变量区标识

变量区标识	区域	变量区标识	区域
101	I 区	110	AI 区
102	Q 区	111	AQ 区
103	V 区	112	XI 区
104	L 区	113	XQ 区
105	M 区	114	XAI 区
106	S 区	115	XAQ 区
107	SM 区	116	PAI 区
108	T 区	117	PAQ 区
109	C 区	119	P 区

➤ 单个变量读命令

表 4.2.3 单个变量读命令

字节偏移	数据类型	描述
1	Char	0x0C 读服务
2~4	Char	保留
5~6	Short	报文长度
7~8	Short	报文标识
9~10	Short	Source AppID
11~12	Short	Destination AppID 变量区标识
13~14	Short	Destination ObjectID 区偏移区偏移, $1 \leq \text{ObjectID} \leq (\text{各个区最大值}-2)$, 表示从此区的第 ObjectID 个字开始取两个字回应。读取 T 区和 C 区变量时则填充定时器和计数器个数偏移。
15~16	Short	SubIndex: 2

正响应

表 4.2.4 单个变量读命令正响应

字节偏移	数据类型	描述
13~16	由解析确定	响应从变量区的指定偏移开始读取 4 个字节数据回复给上位机, 由上位机 PLC_Config 根据用户需求来解析数据, 数据单位可以使位、字、双字或者实数。若为读取定时器和计数器区变量的响应, 第 13 个字节为定时到或计数到标志, 15 和 16 个字节为 1 个 Int 型数据, 表示定时器或计数器的当前值。

➤ 区变量读命令

表 4.2.5 区变量读命令

字节偏移	数据类型	描述
1	Char	0x0C 读服务
2~4	Char	保留
5~6	Short	报文长度
7~8	Short	报文标识
9~10	Short	Source AppID
11~12	Short	Destination AppID 必须为 0
13~14	Short	Destination ObjectID 0
15~16	Short	SubIndex ≥ 0 , 表示从此区的第 SubIndex+1 个字开始取 512 个字节回应。读取 T 区和 C 区变量时则填充定时器和计数器个数偏移。

正响应：

表 4.2.6 区变量读命令正响应

字节数	数据类型	说明
13~524	由解析确定	响应从变量区的指定偏移开始读取 512 个字节数据回复给上位机，由上位机 PLC_Config 根据用户需求来解析数据，数据单位可以使位、字、双字或者实数。对于不够 512 个字节的区，后面的数据填充为随机值。若读取 T 区和 C 区变量，每个定时器或计数器占 8 个字节，举例说明：若区为定时器，SubIndex 为 20，则第 13 个字节为第 20 个定时器(定时器由 0 开始)定时到标志，15 和 16 个字节为 1 个 int 型数据，表示第 20 个定时器的当前值。第 21 个字节为第 21 个定时器定时到标志，23 和 24 个字节为 1 个 int 型数据，表示第 21 个定时器的当前值，依次类推。计数器同定时器。

➤ 多个变量读命令

表 4.2.7 多个变量读命令

字节偏移	数据类型	描述
1	Char	0x0C 读服务
2~4	Char	保留
5~6	Short	报文长度
7~8	Short	报文标识
9~10	Short	Source AppID
11~12	Short	Destination AppID 必须为 0;
13~14	Short	Destination ObjectID 区偏移，1=< ObjectID <=(各个区最大值-2)，表示从此区的第 ObjectID 个字开始取两个字回应。读取 T 区和 C 区变量时则填充定时器和计数器个数偏移。
15~16	Short	SubIndex: 读取变量区个数

正响应：

表 4.2.8 多个变量读命令正响应

字节数	数据类型	说明
13~524	由解析确定	响应自区的偏移开始一共回 512 个字节，具体的位、字、双字、实数据由上位机自行解析，对于不够 512 个字节的区，后面的数据填充为随机值。读取 T 区和 C 区变量，每个定时器或计数器占 8 个字节，举例说明：若区为定时器，SubIndex 为 20，则第 13 个字节为第 20 个定时器(定时器由 0 开始)定时到标志，15 和 16 个字节为 1 个 Int 型数据，表示第 20 个定时器的当前值。第 21 个字节为第 21 个定时器定时到标志，23 和 24 个字节为 1 个 Int 型数据，表示第 21 个定时器的当前值，依次类推。计数器同定时器。

● 变量区写

通过 EPA 协议可访问 18 个变量区，每个变量区都支持单个变量、多个变量和区变量（512 个字节）的写操作。

➤ 单字变量写命令

表 4.2.9 单字变量写命令

字节偏移	数据类型	描述
1	Char	0x0D 写服务
2~4	Char	保留
5~6	Short	报文长度
7~8	Short	报文标识
9~10	Short	Source AppID
11~12	Short	Destination AppID 变量区标识
13~14	Short	Destination ObjectID 区偏移 1=< ObjectID <=(各个区最大值-2)，表示从此区的第 ObjectID 个字开始取两个字回应。定时器 T 区和计数器 C 区为各个定时器和计数器个数偏移
15~16	Short	SubIndex: 必须为 0;
17~18	由上位机写入时确定	ObjectID 偏移开始一共写入 2 个字节，有符号整型或无符号整型由上位机写入时确定。写 T 区和 C 区变量时为定时器或计数器的当前值。

➤ 双字变量写命令

表 4.2.10 双字变量写命令

字节偏移	数据类型	描述
1	Char	0x0D 写服务
2~4	Char	保留
5~6	Short	报文长度
7~8	Short	报文标识
9~10	Short	Source AppID
11~12	Short	Destination AppID 变量区标识
13~14	Short	Destination ObjectID 区偏移，ObjectID >=1，表示从此区的第 ObjectID 个字开始写入相应的区，注意不可超过每个区的最大值。不能写入 T 区和 C 区变量。
15~16	Short	SubIndex: 必须为 0;
17~20	由上位机写入时确定	ObjectID 偏移开始一共写入 4 个字节，有符号双字整型、无符号双字整型或浮点型由上位机写入时确定。不能写入 T 区和 C 区变量。

➤ 位变量写命令

表 4.2.11 位变量写命令

字节偏移	数据类型	描述
1	Char	0x0D 写服务
2~4	Char	保留
5~6	Short	报文长度
7~8	Short	报文标识
9~10	Short	Source AppID
11~12	Short	Destination AppID 变量区标识
13~14	Short	Destination ObjectID 区偏移，ObjectID >=1，表示从此区的第 objID 个字开始写入相应的区，注意不可超过每个区的最大值。
15~16	Short	SubIndex 位偏移，SubIndex >=1，表示从此区偏移字的第 SubIndex 个位开始置位。写 T 区和 C 区变量时为各个定时器或计数器的到达位。
17	Unsigned8	ObjectID 偏移开始一共写入 1 个字节，值必须为 0 或 1。

➤ 多字变量写命令

表 4.2.12 多字变量写命令

字节偏移	数据类型	描述
1	Char	Service ID 0x0D 写服务
2~4	Char	保留
5~6	Short	报文长度
7~8	Short	报文标识
9~10	Short	Source AppID
11~12	Short	Destination AppID 变量区标识
13~14	Short	Destination ObjectID 区偏移, ObjectID ≥ 1 , 表示从此区的第 ObjectID 个字开始写入相应的区, 注意不可超过每个区的最大值。若写 T 区和 C 区变量时则填充各个定时器和计数器个数偏移。
15~16	Short	SubIndex 0
17~	由上位机写入时确定	本命令包长度由写入数据确定, 字节个数为 16+待写入的字变量个数*2。

4.2.4 ModbusTCP/ModbusUDP 通信

DCCE 网络化可编程控制器支持 ModbusTCP/ModbusUDP 通信协议。进行 TCP 方式通信时, 通信设备间需要建立 TCP 连接, 控制器可接收连接请求, 对接收的报文进行相应的处理; 也可调用 TCP_LINK 指令主动发起连接, 访问其他设备资源。使用 UDP 方式通信时, 不需要建立连接, 可接收报文或通过指令主动发送报文, 实现与设备之间的通信。控制器的 EPA 通信端口号默认为 11000, MODBUS TCP/UDP 通信端口号默认为 502。

4.3 转发通信

转发是指将来自工业控制网络中特定设备的请求或者响应, 根据给定的配置, 发往目标设备的过程。主控制器支持网桥的功能, 把上位机访问从设备的命令通过转发的方式传递给从设备, 将从设备响应的数据回传给上位机。

目前仅 PEC 系列控制器支持转发操作, 转发类型包括串口转串口、以太网转串口等方式, 下文将详细描述这些功能的特点以及配置使用方法。

4.3.1 基本概念

● 转发接口:

转发时用于接收、发送转发数据的接口, 目前 PEC 系列控制器的两个串口以及一个以太网接口均支持作转发接口。

● 透明转发:

将来自一个转发接口中的数据, 不进行任何处理, 根据转发配置表, 发送到另一个转发接口的过程。

● Modbus 协议转发:

将来自转发接口的一种 Modbus 协议, 根据转发配置进行协议转换成另一种 Modbus 协议后, 发送到另一个转发接口的过程。

● 转发配置表:

转发配置表用来描述设备在收到转发数据后, 将会发生的行为, 包含协议之间转换, 描述了转发中使用的转发接口及其配置。

● Modbus 地址映射表:

控制器支持变量区地址在 32768~65535 之间的数据访问, 并将这些请求映射到从设备上的相应区域。Modbus 地址映射表定义了为 16 个从设备分配的地址空间。

● 源转发接口:

通信过程中数据报文分为两种, 一种为数据请求报文, 另一种为数据响应报文。对于源转发端口和目的转发接口可以是控制器内部的以太网接口或者串口。源接口为请求报文的流入接口, 响应报文的流出接口。

- 目的转发接口：

根据源转发接口中数据报文的定义，目的接口为请求报文的流出接口，响应报文的流入接口。

4.3.2 功能说明

公司产品支持 3 大类转发功能，透明转发、Modbus 协议转发以及 Modbus 地址转发。

- 透明转发：

在不进行任何处理的情形下，将源转发接口中收到的数据发往目的接口。如将来自串口 1 的数据发送给以太网接口。

- Modbus 协议转发：

将来自源转发接口中的一种 Modbus 协议，转换成目的接口支持的 Modbus 协议后发送数据。如将来自以太网的 Modbus TCP 协议转换成为 Modbus ASCII 后，通过串口 1 发送给目标设备。

- Modbus 地址转发：

启用 Modbus 地址映射时，主控制器将请求根据配置表，发送给相应从设备。如主控制器收到地址为 50000 的 Modbus 请求时，且 8 号从设备配置的 Modbus 地址区间为 50000~60000，那么主控制器将请求转发给 8 号从设备。

不配置 Modbus 地址映射时，设备也支持 Modbus 地址转发，表 4.3.1 为默认的转发地址表，如果设备收到地址范围在 32768~65535 之间的请求后，转发给相应的从设备。

表 4.3.1 转发寻址方式

从设备编号	Modbus 地址		地址分配
	十进制	十六进制	
0	32768~34815	8000~87FF	0~2047
1	34816~36863	8800~8FFF	0~2047
2	36864~38911	9000~97FF	0~2047
3	38912~40959	9800~9FFF	0~2047
4	40960~43007	A000~A7FF	0~2047
5	43008~45055	A800~AFFF	0~2047
6	45056~47103	B000~B7FF	0~2047
7	47104~49151	B800~BFFF	0~2047
8	49152~51199	C000~C7FF	0~2047
9	51200~53247	C800~CFFF	0~2047
10	53248~55295	D000~D7FF	0~2047
11	55296~57343	D800~DFFF	0~2047
12	57344~59391	E000~E7FF	0~2047
13	59392~61439	E800~EFFF	0~2047
14	61440~63487	F000~F7FF	0~2047
15	63488~65535	F800~FFFF	0~2047

地址分配为从设备中的所能访问的绝对地址，实际地址为上位机监控所输入的绝对地址，每个从设备仅支持变量地址为 0~2047 的变量，变量地址大于 2047 的变量将不被支持。

4.3.3 性能指标

处于转发模式下的设备支持多种转发类型，如以太网转串口转发，串口转以太网转发以及串口转串口转发。当配置合理时，处于转发状态中的设备可以同时支持所有转发类型。转发操作会消耗一定的系统资源，因此使用转发时存在以下限制：

- 使用协议相关转发时，最多支持 20 个转发表项，构造 20 条转发链路；
- 如果转发操作的一端是以太网，那么一个 IP 只能存在一个链接（通过 IP: PORT 标识）；
- 如果转发操作中使用了 TCP/IP，那么转发系统中最多只能存在 2 个 TCP/IP 链接；
- 如果转发中使用了 UDP，那么配置中最多只能存在 3 个 UDP 链接；
- 如果在启动转发的同时，设备也启动了 Modbus TCP Server，那么能够连接到设备的 Modbus TCP Client 的数量最多为 6 个。Modbus TCP Client 个数 = 6 - 转发中使用的 TCP/IP 链接数；
- 产品作为 Modbus TCP Server 时，监听来自 500~600 的端口的 TCP 链接。如果通过配置软件，配置了其中一些端口作为转发的监听端口，那么这些端口将不再处理 Modbus TCP Client 的请求，而是处理转发

请求；

- 协议相关转发、透明转发以及 Modbus 地址转发，设备在某一时刻只能支持 3 者之一；
- 为了防止协议转发通信与主从通信相互影响，规定源转发接口如果为串口，那么该口必须为从口；目的转发接口如果为串口，那么目的串口为主口。这样保证了数据请求总是由主口流出，从口流入；
- 透明转发中使用的串口强制为从口。

4.3.4 配置转发参数

进行转发时，需要对转发参数进行配置。默认情形下公司产品的转发功能处于关闭状态。正确配置 SM 区、转发配置表以及 Modbus 地址映射表后，设备进入转发状态，将指定的报文转发给目标设备。

● SM 区配置

转发的开启、关闭以及转发类型的选择通过 SMW254 进行控制，其值和意义如表 4.3.2 所示。

表 4.3.2 转发类型描述表

SM 区	描述	
SMW254	0	默认 Modbus 地址映射转发状态
	1	设备处于透明转发状态
	2	设备处于 Modbus 协议转发状态
	3	设备处于 Modbus 地址映射转发状态

当设备处于透明转发或者 Modbus 协议转发状态（SMW254 的值为 1 或者 2）时，设备收到请求或者响应后，会根据转发表中的配置选择一条合理的路径转发报文，如果不存在这样的路径，那么设备根据该报文的类型将报文交给特定的处理程序。例如当前请求来自以太网且其协议为 TCP/IP，那么该报文将被默认认为一条 Modbus TCP 请求，由下位机 Modbus TCP Server 处理后返回给请求设备。

当设备处于 Modbus 地址转发状态时（SMW254 被设置为 3），设备根据 Modbus 地址映射表查找与请求的 Modbus 地址相符合的从设备，并将请求转换后发送给该从设备。

转发系统开启后，SMW255 描述了转发系统所处的状态，其值和意义如表 4.3.3 所示。

表 4.3.3 转发状态描述表

SM 区	描述
SM255.0	表示转发类型以及转发接口的协议不符（如系统处在透明转发状态下，但是转发表中存在 Modbus 协议转发的表项）
SM255.1	TCP 链接数量超过限制
SM255.2	UDP 链接数量超过限制
SM255.3	表示同一 IP 地址，在表中出现了两次链接（如转发接口配置中，出现了 IP 相同，端口不同的表项）

提示：

- (1) 转发选择控制字 SMW254 掉电丢失，设备重启后应重新设置；
- (2) 进行转发时，需将从设备通信模式控制字 SMW42 设置为 1。

● 配置转发表

转发表是数据转发的依据，转发表不但规定了转发中使用的接口协议，还规定了数据的流向以及控制器内部处理数据的方式，因此正确配置转发表是十分重要的。图 4.3.1 为配置转发表的配置界面。

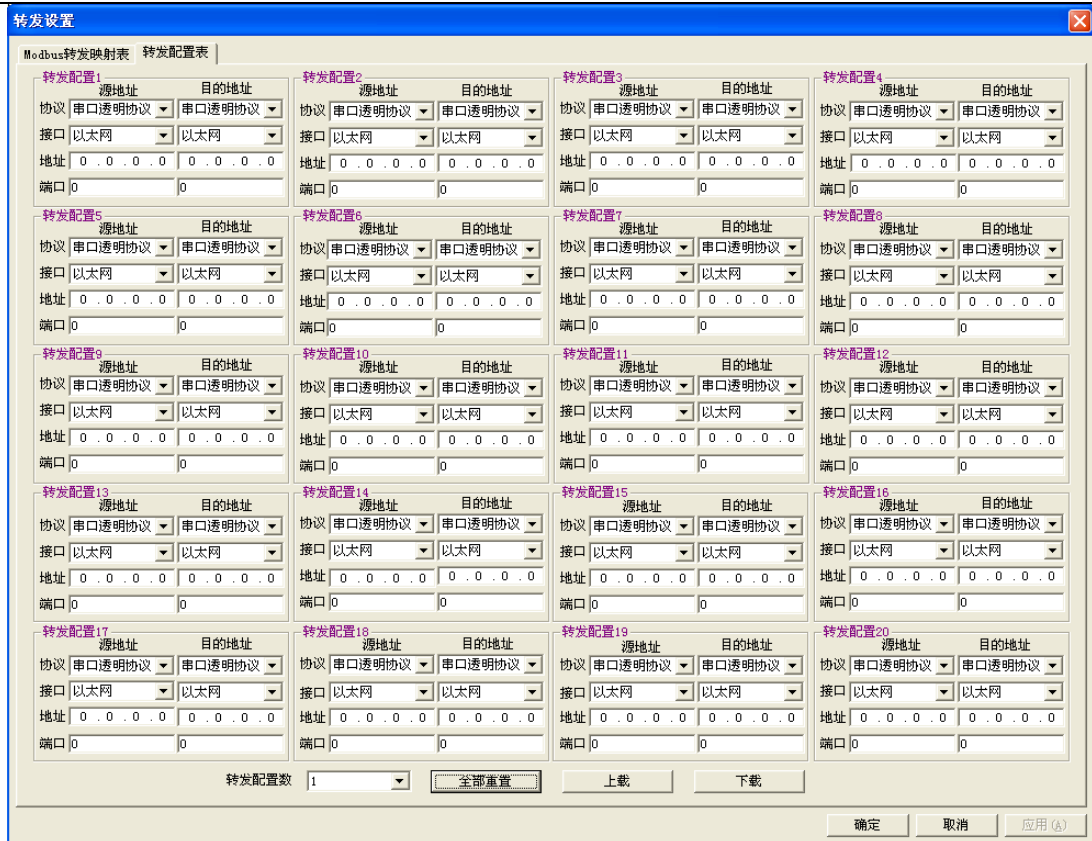


图 4.3.1 转发表配置

在三个转发应用中，透明转发和 Modbus 协议转发需要配置转发表。转发表中包含的项目如图 4.3.2 所示。

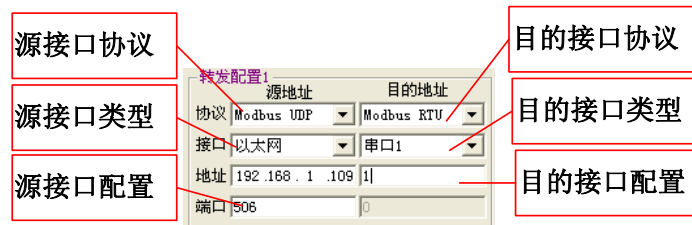


图 4.3.2 转发表配置

公司产品转发模型，所用的通信都被看成请求、应答式。数据请求从源接口流入，从目的接口流出。数据应答从目的接口流入，源接口流出。

- 接口类型：

请求流入接口的类型，可以设置为 COM1、COM2 以及以太网。

- 源接口配置：

源接口配置参数。如果源接口类型为以太网，那么该参数为设备监听的 IP 地址以及端口号；如果源接口类型为串口，且转发类型为 Modbus 协议转发，那么该参数表示串口监听的 Modbus 站地址号；如果转发类型为透明转发，该参数没有意义。

- 源接口协议类型：

表示源接口与请求设备间使用的通信协议，可以设置为 Modbus 协议，也可以设置为 RAW 协议（RAW 协议指没有任何格式的原数据）。

- 目的接口类型：

请求流出接口的类型，可以设置为 COM1、COM2 以及以太网。

- 目的接口配置：

其配置参数以及意义与源接口配置一致，可参考源接口配置。

提示：

- (1) 对于转发配置表中未使用的串口，不受转发操作的影响；
- (2) 对于 Modbus 协议转发来说，若转发串口处于主口状态，转发操作不影响主口下面配置的从设备；若转发串口处于从口状态，转发操作不影响目的接口为转发设备的 Modbus 请求；
- (3) 端口 502 为 Modbus TCP 专用端口，不可作为转发监听端口。

● Modbus 地址映射表

如果设备处于 Modbus 从设备转发状态，那么需要根据 16 个从设备对应的 Modbus 地址区间正确配置 Modbus 地址映射表。其配置界面如图 4.3.3 所示。

配置转发表时，需保证分给每个从设备的地址区间交集为空，以免造成转发错误。正确配置 Modbus 地址映射表后，将 SMW254 设置为 3（将设备状态设置为 Modbus 地址转发状态）。

每个从设备对应一个地址区间，如 10 号从设备对应的地址区间为 53248~55295 之间，当主控制器接收到地址在 53248~55295 之间的 Modbus 请求时，会将请求发送给 10 号从设备处理。

访问从设备变量区时，发送给主控制器的变量地址为 53248+从设备变量地址偏移，如访问从设备 10 的 VW0 数值，对应的 Modbus 地址为 53248+2336（VW0 地址）=55584。

从设备号	开始地址	终止地址
从设备0	52755	34815
从设备1	34816	36863
从设备2	36864	38911
从设备3	38912	40959
从设备4	40960	43007
从设备5	43008	45055
从设备6	45056	47103
从设备7	47104	49151
从设备8	49152	51199
从设备9	51200	53247
从设备10	53248	55295
从设备11	55296	57343
从设备12	57344	59391
从设备13	59392	61439
从设备14	61440	63487
从设备15	63488	65535

图 4.3.3 Modbus 地址映射表

4.3.5 配置实例

通过实例对用户描述转发的使用。

● 透明转发

已知工业现场中的通信模型如图 4.3.4 所示。

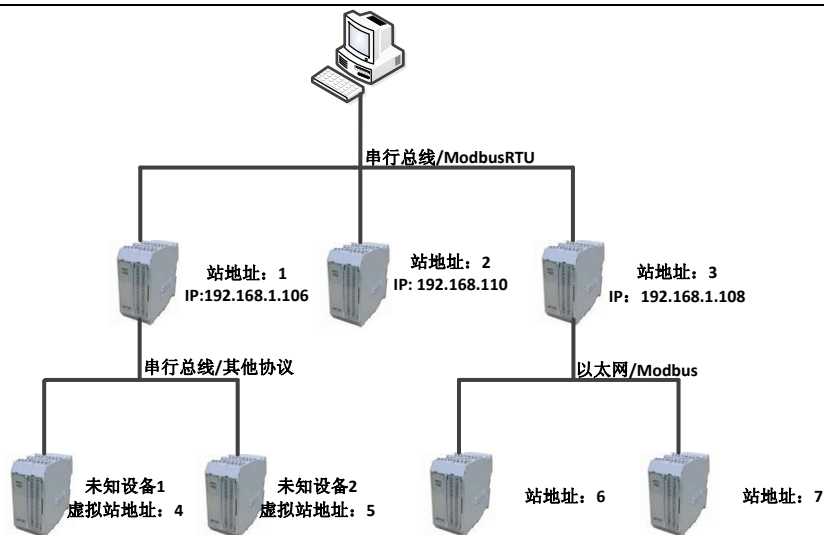


图 4.3.4 工业网络结构

未知设备 1 需要访问站地址 6 中的资源完成某项工作。网络中 IP 地址为 192.168.1.108 的设备支持转发操作，且未知设备 1 与站地址为 6 的设备都支持 Modbus RTU 协议。

➤ 配置转发表

为了实现完成上述任务，需要使用设备 192.168.1.108 以及 192.168.1.106 的转发功能。未知设备 1 请求站地址为 6 的设备中的资源，因此数据请求需要通过从 192.168.1.106 与未知设备 1 之间的串行总线，经过工业以太网，流入 192.168.1.108 与站地址为 6 的设备间的串行总线。Modbus RTU 请求的流程如下所示。

未知设备串口→设备 192.168.1.106 的串口 1→设备 192.168.1.106 的以太网→设备 192.168.1.108 的以太网→设备 192.168.1.108 的串口 1→站地址为 6 的设备。

配置 192.168.1.106 的转发配置表，如图 4.3.5 所示。其参数意义如下：

- 源转发接口：串口 1；
- 源转发协议：透明处理协议，对接收到的数据不做任何处理；
- 源接口配置：1，在透明转发下该参数没有意义；
- 目的转发接口：以太网；
- 目的转发协议：透明处理协议，对接收到的数据不做任何处理；
- 目的接口配置：转发下一站地址为 192.168.1.108：507。

转发配置1	
源地址	目的地址
协议 TCP透明协议	串口透明协议
接口 以太网	串口1
地址 192.168.1.106	0
端口 507	0

图 4.3.5 192.168.1.106 转发表配置

配置 192.168.1.108 的转发配置表，如图 4.3.6 所示。其参数意义如下：

- 源转发接口：以太网；
- 源转发协议：透明处理协议，即对接收到的数据不做任何处理；
- 源接口配置：监听 IP 地址为 192.168.1.108，监听端口为 507；
- 目的转发接口：串口 1；
- 目的转发协议：透明处理协议，即对接收到的数据不做任何处理；
- 目的接口配置：0，在透明转发下该参数没有意义。

图 4.3.6 192.168.1.108 转发表配表

- 启动转发

透明转发过程中以太网的部分通过 TCP/IP 进行通信，启动 TCP/IP 客户端之前，应先启动 TCP/IP 服务端。因此按照以下步骤启动系统：

- 将设备 192.168.1.108 的 SMW254 设置为 1，表示启动透明转发；
- 将设备 192.168.1.106 的 SMW254 设置为 1，表示启动透明转发，这时设备 192.168.1.106 会主动向设备 192.168.1.108: 507 发起 TCP/IP 链接；
- 如果 TCP 链接成功，此时未知设备 1 与站地址为 6 的设备可以通过 Modbus 进行通信。

- Modbus 协议转发

工业网络拓扑图如图 4.3.4 所示，设备 192.168.1.110 为网络上 Modbus TCP Server，站地址为 5 的设备需要 IP 为 192.168.1.110 的设备采集的数据进行逻辑运算，并将运算的结果反馈给设备 192.168.1.110。

- 配置转发表

这种模型类似串行通信中主从设备的资源映射机制，通过协议相关转发，将 Modbus RTU 协议转发为 Modbus TCP。

接线时，站地址为 5 的设备的串口 1 通过串行总线与设备 192.168.1.108 的串口 1 相连，因此转发的数据流如下所示：

Modbus RTU 请求→设备 192.168.1.108 的串口 1→设备 192.168.1.108 以太网→设备 192.168.1.110。来自站地址为 5 的设备的 Modbus RTU 请求被设备 192.168.1.108 转换成为 Modbus TCP 请求发送给设备 192.168.1.110。

设备 192.168.1.108 的转发表配置如图 4.3.7 所示。

图 4.3.7 192.168.1.108 配置

其参数值及意义如下：

- 源接口类型：串口 1；
- 源接口协议：Modbus RTU 协议；
- 源接口配置：2 号，即所有目的站地址为 2 的请求都被发送到目的接口；
- 目的接口类型：以太网；
- 目的接口协议：Modbus TCP；
- 目的接口配置：192.168.1.110: 507，表示由串口 1 接收到的目的站地址为 7 的请求，在经过处理后都被发送到设备 192.168.1.110 的 507 端口。

- 启动转发

例程中只有一个设备作为协议转发设备，因此将 SMW254 设置为 2 即可（2 表示系统使用协议转发配置进行转发操作），但是在启动转发之前应保证设备 192.168.1.110 的 Modbus TCP Server 已经正常启动。

● Modbus 地址转发

Modbus 地址转发主要应用在主从设备间，外部设备可以通过主控制器访问主从网络中的从设备。当 SMW254 为 0 时，使用默认的地址映射，如果 SMW254 为 3 时，使用地址映射表中的地址映射。

如一主从网络中，主控制器带有 3 个用于采样的串口从设备，上位机通过 INTOUCH 组态软件监视系统中数据的变化。INTOUCH 不仅需要采集主控制器中数据，同时还需要采集从设备中的数据。

例如上位机需要访问第一个从设备 0~3000(MW152)地址区域的数据,第 2 个从设备 0~13631(VW10751)地址区间的数据，第 3 个从设备 0~13631 (VW10751) 的地址区间，那么其地址映射表可以按图 4.3.8 方式配置。

从设备0 开始地址: 32768 终止地址: 35767	从设备1 开始地址: 35768 终止地址: 49398	从设备2 开始地址: 49399 终止地址: 63029	从设备3 开始地址: 0 终止地址: 0
从设备4 开始地址: 0 终止地址: 0	从设备5 开始地址: 0 终止地址: 0	从设备6 开始地址: 0 终止地址: 0	从设备7 开始地址: 0 终止地址: 0
从设备8 开始地址: 0 终止地址: 0	从设备9 开始地址: 0 终止地址: 0	从设备10 开始地址: 0 终止地址: 0	从设备11 开始地址: 0 终止地址: 0
从设备12 开始地址: 0 终止地址: 0	从设备13 开始地址: 0 终止地址: 0	从设备14 开始地址: 0 终止地址: 0	从设备15 开始地址: 0 终止地址: 0

图 4.3.8 Modbus 地址映射表配置

图中仅显示了 0~10 从设备的配置，11~15 号从设备配置与 3~10 号从设备配置相同，我们不关心从设备编号（图中的 Handle 号）为 3~15 的从设备。

编号 0 的转发映射地址为 32768~35767 共 3000 个字变量。例如 INTOUCH 想要读取 VW0 的数据，配置 INTOUCH 时，将访问变量地址值设置为 $35105 = 32768$ （映射起始地址）+ 2336（VW0 地址偏移）+ 1。

编号 1 的转发映射地址为 35768~49398 共 13631 个字变量。例如 INTOUCH 想要读取 MW0 的数据，配置 INTOUCH 时，将访问变量的地址值设置为 $38617 = 35768$ （映射起始地址）+ 2848（VW0 地址偏移）+ 1。

编号 2 的转发映射地址为 49399~63029 共 13631 个字变量。例如 INTOUCH 想要读取 VW500 的数据，配置 INTOUCH 时，将访问变量的地址值设置为 $52236 = 49399$ （映射起始地址）+ 2836（VW0 地址偏移）+ 1。

4.3.6 网关配置

为了控制器能够更好地与局域网络内不同网段的设备通信，下位机提供了默认网关以及子网掩码的配置。

- 配置默认网关
- 在控制管理器中，配置默认网关的界面如图 4.3.9 所示。

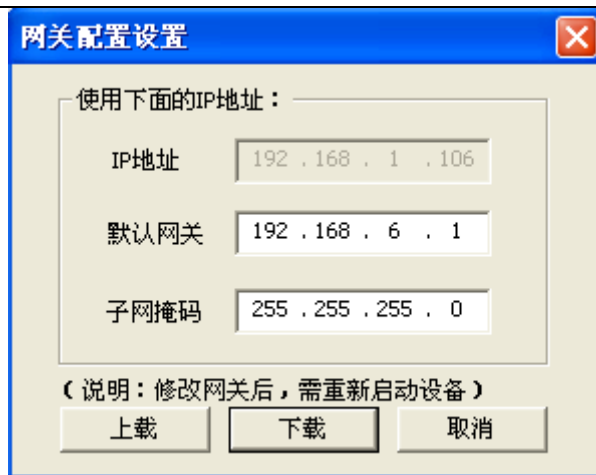


图 4.3.9 默认网关配置

其中

Device IP：要配置设备的 IP 地址；

Gateway：设备默认网关；

SubMask：设备子网掩码。

配置完默认网关以及子网掩码后，点击下载后将配置写入设备 Flash 中，但是这时系统并没有使用新配置进行 IP 层数据交换，只用当设备重启后或者点击 StartRTE 后才会使用新配置进行通信。

● 子网掩码配置示例

现场应用中存在两块设备，一块设备 IP 为 192.168.1.110，另一块设备 IP 为 192.168.10.10，两块设备在同一个局域网中。IP 为 192.168.1.110 的设备为主控模块，通过以太网获取扩展设备 192.168.10.10 的 IO 数据，并将 IO 数据反馈给上位机。

默认情形下，两块设备的子网掩码为 255.255.255.0，即两块设备位于不同的网络中，虽然在它们同在一个物理局域网中。这种情形下修改两设备的子网掩码，将两设备划入同一个局域网中即可。

设备 192.168.1.110 的配置如图 4.3.10 所示。

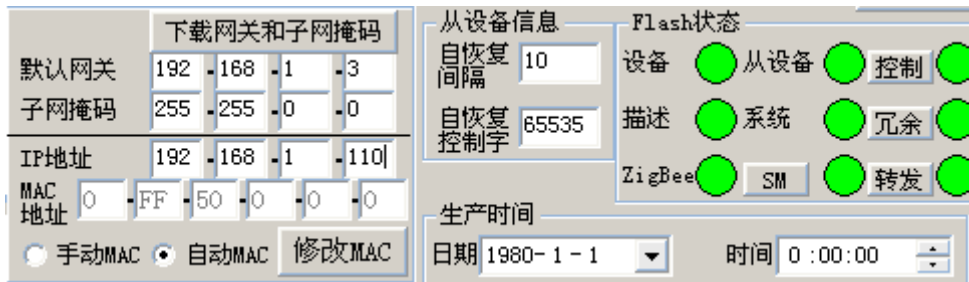


图 4.3.10 子网掩码配置图 1

设备 192.168.10.10 的配置如图 4.3.11 所示。

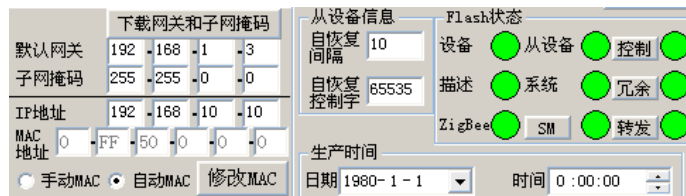


图 4.3.11 子网掩码配置图 2

正确配置设备子网掩码后，将两块设备重新上电。此时，主控制器 192.168.1.110 通过如 NETR、NETW 等功能块与设备 192.168.10.10 进行资源映射。

● 网关配置实例

上位机 PC(IP:192.168.1.103)所在网络为 192.168.1.0/255.255.255.0，设备(IP:192.168.5.106)所在网络为

192.168.5.0/255.255.255.0，两设备位于不同的网络中，因此不能通信。网络中存在路由设备 192.168.1.109，它的路由表中存在到网络 192.168.5.0/255.255.255.0 的路由项。因此，为了 PC 能够和设备通信分别配置 PC 和设备的默认网关。

PC 默认网关以及子网掩码配置如图 4.3.12 所示。

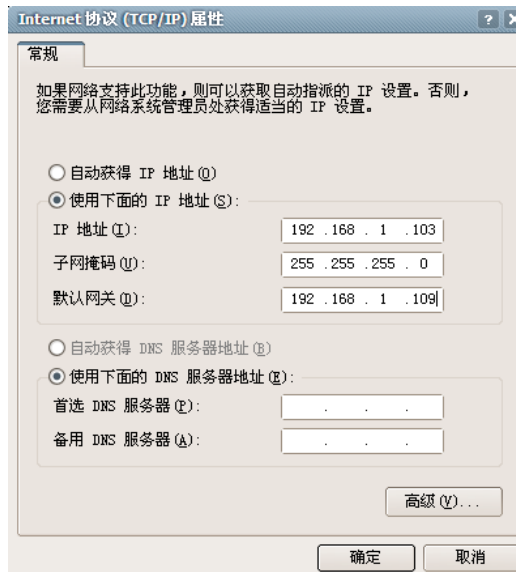


图 4.3.12 PC 默认网关配置

设备默认网关以及子网掩码配置如图 4.3.13 所示。

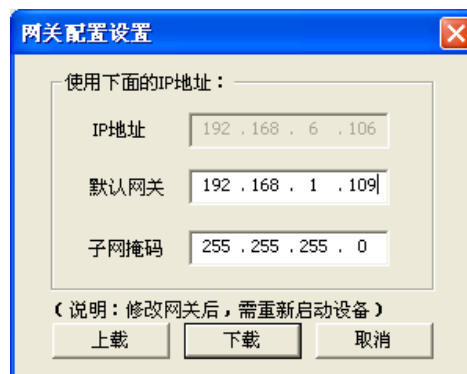


图 4.3.13 设备默认网关配置

5 系统管理

典型的计算机控制系统通常可以分为设备层、控制层、监控层、管理层四个层次结构，构成了一个分布式的工业网络控制系统，其中设备层负责将物理信号转换成数字或标准的模拟信号，控制层完成对现场工艺过程的实时监测与控制，监控层通过对多个控制设备的集中管理，以完成监控生产运行过程的目的，管理层实现对生产数据进行管理、统计和查询。

多个控制器可组成分布式控制系统，对生产数据进行分布运算，实时汇总。用户通过收集生产过程数据，及时优化工艺与过程管理，达到提高生产效率、节约能源、降低生产成本的目的。分布式控制系统组成如图 5.1 所示。

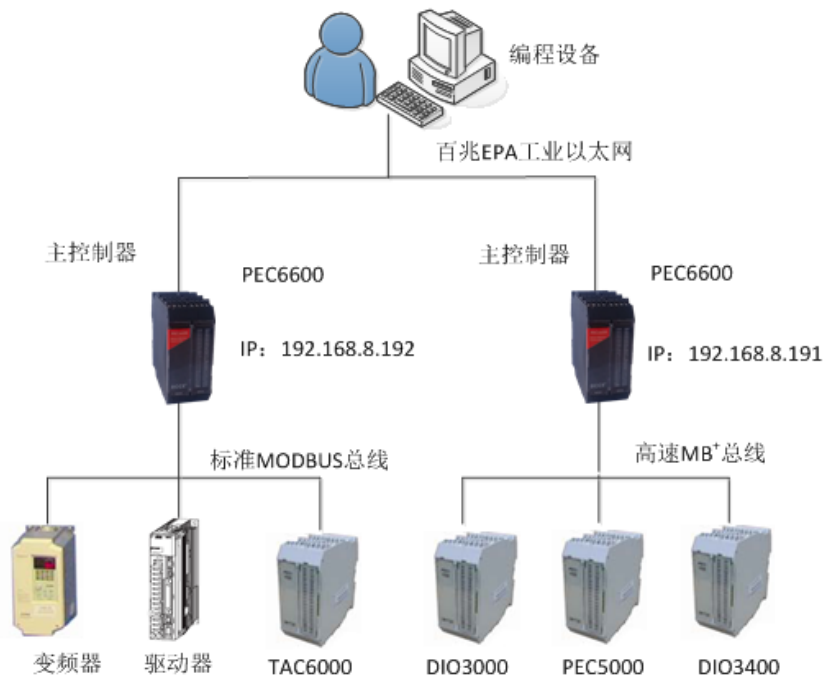


图 5.1 分布式控制系统

分布式控制系统由编程设备、主控制器和扩展从设备 3 部分组成，用户通过编程设备对系统中的主控制器进行操作和管理；两个主控制器之间进行变量的映射、通信校验和同步运行；主控制器可利用串口的高速 MB+ 总线和标准 Modbus 总线扩展从设备，完成对整个系统的分布式控制。

5.1 分布式控制系统

分布式控制系统是由多个控制器分别控制生产过程中的多个控制回路，同时又可集中获取数据、集中管理和集中控制的自动控制系统。

系统支持多级从设备扩展，每级可扩展 16 个从设备，单台设备控制规模包括 10/20/30/40 多种 I/O 点数组合，基于对主流控制网络的良好兼容性和分布式控制特点，可满足大中小型控制系统的通用控制需求。

编程设备对系统中的控制器进行操作和管理。实际使用 PLC_Config 组态软件作为编程软件，进行现场设备的程序编写、开发与实时监控，实现系统的分布式控制。

主控设备是系统的控制核心，负责处理从设备采集的数据，记录设备的运行参数，进而控制整个系统。主控设备之间可以通过以太网实现变量映射，通信校验与同步运行功能。

从设备作为主控设备的扩展，用以增加系统的 I/O 点数，负责采集系统数据，分担主控设备的任务。

5.2 指令网络通信

指令通信网络系统分为以太网和串行网络，分别通过以太网读写指令和串行读写指令实现变量的映射，通信校验和同步运行等功能。

(1) 变量映射

- 以太网变量映射

控制器支持 12 种以太网读写指令，用于多个以太网对等控制器之间的变量映射和系统的协同控制。例如，控制器可通过以太网字读指令获取指定 IP 地址的控制器变量信息。

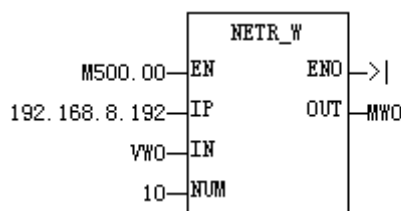


图 5.2.1 以太网字读指令

如图 5.2.1 所示，控制器运行以太网字读指令读取 IP 地址为 192.168.8.192 的控制器 VW0~VW9。读取后将其放入到本地控制器的 MW0~MW9 中，供本地控制器使用。

● 串行变量读写

控制器支持 4 种串行读写指令，用于控制器通过串行通信实现设备间的变量映射。例如，控制器可通过串行字读指令获取指定 Modbus 站地址的设备变量信息。

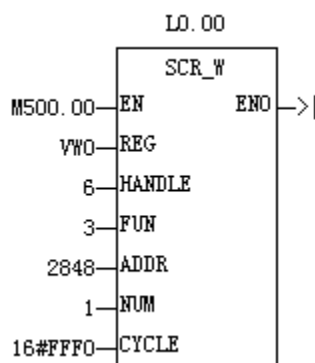


图 5.2.2 串行字读指令

如图 5.2.2 所示，控制器运行串行字读指令读取 Modbus 站地址为 6 的串行设备的一个 Modbus 地址为 2848 的字变量，并将读取到的数据放入本地的 VW0 中，可供本地控制器使用。

(2) 通信校验

用户可根据实际情况自行设计通信校验方法，以防止错误报文的处理，影响系统正常工作。例如，发送设备可修改发送报文，在报文的起始和结束处添加检测字符，接收设备根据接收报文的起始字符与结束字符判断是否为所需报文，若为所需报文，对其进行相应的处理。此外，控制器还提供了数据校验码生成和检查指令，发送设备可通过数据校验码生成指令对发送数据进行校验，生成校验码重新打包后再发送，接收设备可通过数据校验码检查指令，判断接收报文是否正确，若正确对其进行相应的处理。

(3) 同步运行

● 以太网同步运行

以太网指令网络中通信设备之间需要设置同步周期。同步周期通过以太网通信指令控制字配置，如表 5.2.1 所示。

表 5.2.1 以太网通信指令控制字

SM 区	描述	默认值
*SMW182	以太网读指令通信最小间隔时间	200ms
*SMW183	以太网读指令最大超时时间	400ms
*SMW184	以太网写指令通信最小间隔时间	1s

● 串行同步运行

串行网络中控制器之间需要设置同步周期，实现网络的同步运行。同步周期通过 PLC_Config 软件中的设备参数配置界面或用户版调试软件配置，串口通信控制字如表 5.2.2 所示。

表 5.2.2 串口通信参数配置

控制字	描述	默认值
串口 1 通信控制字	串口 1 从设备通信最大错误次数	10 次
	串口 1 从设备通信时间间隔	100ms
	串口 1 从设备通信最大等待时间	1000ms
串口 2 通信控制字	串口 2 从设备通信最大错误次数	10 次
	串口 2 从设备通信时间间隔	100ms
	串口 2 从设备通信最大等待时间	1000ms

5.3 主从网络通信

控制器支持从设备管理功能，主从设备之间可以进行资源映射，用以扩展物理输入输出；用户程序仅对主控制器中的变量进行操作，即可完成对从设备的资源管理，控制从设备的物理输入输出。

控制器可通过串口的方式扩展从设备，主从设备自动完成通信操作，资源映射简单、快捷，提供了多种资源映射模式。

5.3.1 从设备基本概念

从设备是指和主控制器通过串口通信的方式连接，通过参数配置或用户程序处理，将自身的资源映射到控制器内部，交由控制器管理的设备。

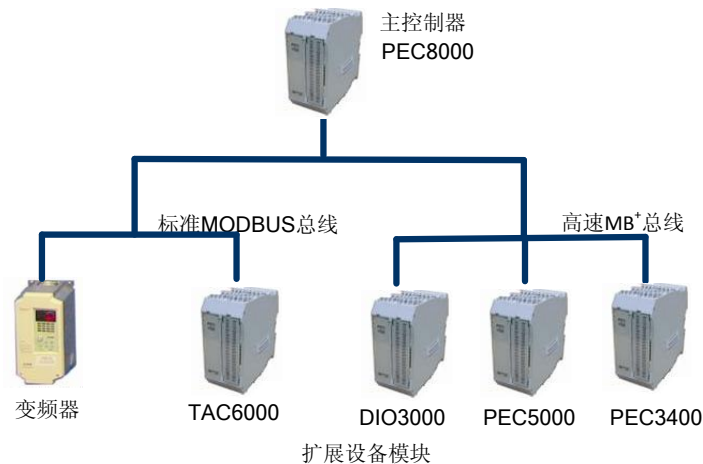


图 5.3.1 扩展从设备配置

图 5.3.1 中的 DIO3000、PEC3400、TAC6000、变频器作为 PEC8000 的从设备。经过配置后，用户通过操作主控制器 PEC8000 中的资源，即可完成对上述从设备操作，从而控制与从设备连接的现场资源。例如通过调节变频器的输出频率，控制电机的转速。

使用从设备有以下优点：

(1) 提升控制的性能。由于每个控制器所拥有的资源有限，而工业现场的控制任务千变万化，虽然可以选择使用不同的控制器来完成控制任务，但是在复杂的控制系统尤其是点数较多的控制系统中，单独的控制器往往无法满足现场的需求，因此需要扩展控制器的资源，而利用通信扩展其他设备则是比较理想的方式之一；

(2) 与其他设备互联。很多工业现场的控制系統往往由多种控制器协作完成，为了完成统一的控制任务，这些控制器之间要互相传输数据，故控制器可以利用通过将其他设备配置为从设备的方式进行数据传递；

(3) 分布式控制可降低系统运行风险。通过扩展从设备，可以将控制系统的任务进行模块化处理，不再是集中在一个控制器中，在主控制器出现故障时，扩展从设备仍然可以执行自己的任务而不受影响，这样可以有效地避免现场由于主控制器死机导致整个系统瘫痪的事故发

5.3.2 从设备的种类与连接

(1) 从设备的种类

根据设备生产厂家的不同，将控制器支持的从设备分为两大类，公司产品和第三方产品。

- 公司产品

公司产品是指 DCCE 公司生产的控制器、数据采集模块或控温模块等产品。管理这些设备时，使用较为简单的参数配置即可将用户所需要的资源映射到主控制器内，使用起来较为方便，具体支持的设备种类请参见附录 D。

- 第三方产品

其他非 DCCE 公司生产的产品均称之为第三方产品，用户在使用时需要详细阅读相关设备的说明以及控制器的配置方法，根据不同的设备进行配置。

(2) 连接

- 通信接口

控制器通过串行通信的方式管理从设备，每个控制器具有两路串口，均有主从模式的配置，配置为主口模式时才能与从设备通信。支持 1200bps~115200bps 的标准速率通信，支持两路串口同时配置为主口来管理从设备。

- MB⁺高速总线

对于部分公司产品，控制器支持一种速率高达 500Kbps 和 1Mbps 的 MB⁺总线，当配置为此种速率时，将极大地提高主从设备的响应时间，用户只需要在配置通信参数时按照图 5.3.2 修改为 1Mbps 的速率即可。

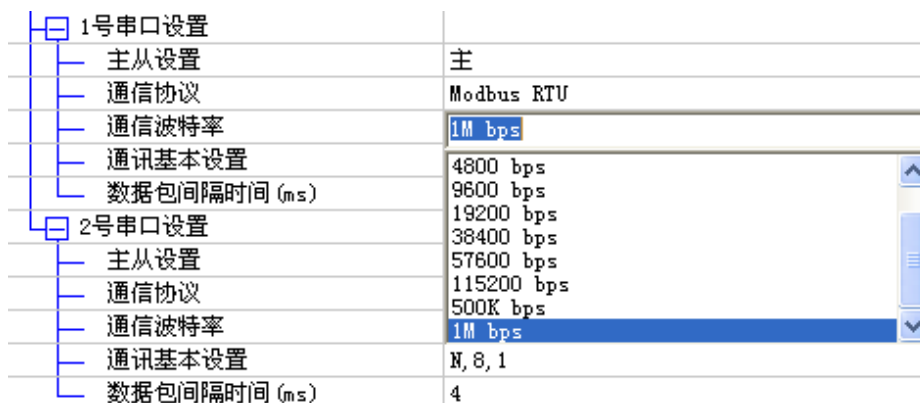


图 5.3.2 MB⁺总线配置

提示：

(1) 有关 Modbus 协议的相关信息请参看第 4 章的 Modbus 协议介绍；

(2) 当选择为高速 MB⁺总线通信时，主控制器最大扩展从设备为 13 个，超过 13 个从设备时要在总线上加匹配电阻，范围为 33 欧姆~270 欧姆，典型值为 80 欧姆。

- 设备编号

为方便用户的管理，主控制器为每个从设备进行了编号，作为从设备的索引，0~15 对应 16 个从设备，用户可随意配置控制器编号，每个编号下的从设备管理资源和方法均相同。

- 设备上线和下线

扩展从设备有上线和下线两个状态。只有在从设备上线的情况下主控制器才能与从设备通信，并对其进行各种操作。

主控制器需要在从设备编号下配置好相关参数，根据配置的参数与从设备通信，判断从设备是否上线。从设备上线后，若主控制器与其连续通信错误超过一定次数，主控制器会认为从设备下线。用户可以在设备参数界面中查看从设备通信最大错误次数，根据现场情况自行设置调整。从设备上线和下线的上位机界面如图 5.3.3 所示。



图 5.3.3 从设备上线和下线

● 扫描从设备

为了更好地管理从设备的上线和下线，主控制器提供了 4 种对从设备的扫描方式。

(1) 上电自动扫描

用户配置好从设备的相关参数后，主控制器上电后会根据参数自动扫描配置的从设备，检查从设备是否在线。

(2) 手动扫描

用户配置好从设备的相关参数后，使用“手动扫描从设备”强制扫描从设备，检查从设备是否在线。

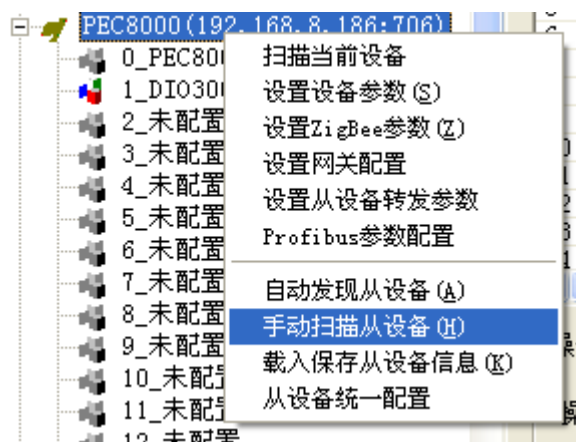


图 5.3.4 手动扫描

(3) 自恢复扫描从设备

用户配置好从设备的相关参数后，如果主控制器运行过程中从设备下线或上电没有扫描到从设备，主控制器会以固定时间间隔扫描从设备，查看从设备是否在线。用户可以在设备参数中配置查询间隔和是否允许主控制器自恢复扫描。

(4) 自动发现从设备

此扫描功能只对公司产品起作用。在主从设备连接好后，用户只需要执行“自动发现从设备”命令，主控制器就会自动发现从设备。



图 5.3.5 自动发现从设备

在自动发现从设备完毕后，从设备图标呈咖啡色。此时主从设备还不会进行通信，用户需要确认扫描到的从设备确实与用户配置的从设备相符，下载从设备的配置参数后，主从设备才能正常通信。

5.3.3 参数配置

与从设备相关的参数包含主从通信参数、公共参数、资源映射参数 3 部分，用户可以通过配置这些参数

来使主从设备通信达到最优。

(1) 主从通信参数

主控制器通过 MB+总线协议最多可扩展 16 个从设备, 用标准 Modbus 协议, 支持同时管理公司设备和第三方设备同时扩展, 用户可以通过 PLC_Config 中的设备参数界面配置从设备通信参数, 配置示例如图 5.3.6 所示。

从设备自恢复扫描控制字	65535
从设备自恢复扫描时间间隔 (s)	10
从设备通信最大允许错误次数	10
从设备通信时间间隔 (ms)	100
从设备通信最大等待时间 (ms)	1000
1号串口设置	
主从设置	从
通信协议	Modbus RTU
通信波特率	9600 bps
通讯基本设置	N, 8, 1
数据包间隔时间 (ms)	4
2号串口设置	
主从设置	从
通信协议	Modbus RTU
通信波特率	9600 bps
通讯基本设置	N, 8, 1
数据包间隔时间 (ms)	4

图 5.3.6 从设备相关参数

从设备自恢复扫描是系统设备上线管理的重要部分, 自恢复扫描控制字共 16 位, 0~15 位对应 0~15 号的从设备, 相应位为 1 则允许从设备自恢复扫描, 为 0 则不允许从设备自恢复扫描, 默认值为 65535。

从设备自恢复扫描时间间隔为进行一次自恢复扫描所用时间, 单位为秒, 默认值为 10。只有当主控设备的通信串口配置“主”口时才能与从设备通信。主从设备的通信协议、校验位、数据位、停止位、波特率等参数要保持一致。

主从设备连续通信错误次数超过从设备最大允许错误次数后, 主设备会认为此从设备下线。默认值为 10 次。

从设备通信时间间隔是指主设备与从设备通信一次所需时间, 用户可以通过调整此参数来控制主从设备的数据交换速度。默认值为 100ms。

从设备通信最大等待时间是指主设备从发送命令到接收从设备响应所需要的最大等待时间, 若超过此时间后, 主设备仍未接收到响应数据, 则认为本次通信失败。默认值为 1000ms。

控制器的从设备通信时间间隔和从设备通信最大等待时间可根据通信波特率的不同进行合理配置。从设备通信时间间隔和从设备通信最大等待时间设置较小时会影响控制器的程序执行效率。故在实际应用中, 需要根据对程序执行效率的要求适当调整从设备通信时间间隔和从设备通信最大等待时间。例如, 以 PEC6000 作为主设备, PEC3000 作为从设备, 主设备中运行 1000 个 AND 功能块, 在不同波特率下得出的从设备通信时间间隔和从设备通信最大等待时间极限值如表 5.3.1 所示。

表 5.3.1 从设备通信参数配置

波特率	从设备通信时间间隔	从设备通信最大等待时间
9600bps	15ms	150ms
115200bps	5ms	50ms
500Kbps	1ms	1ms
1Mbps	1ms	1ms

报文传送时间间隔是主控制器接收从设备响应时, 数据包中两个相邻字节的最大间隔时间。当相邻字节的间隔时间超过此设定值时, 则认为报文传送结束, 控制器开始处理数据。根据波特率不同, 报文传送时间间隔不同, 单位为 ms。在 Modbus RTU 协议中规定以总线上不少于 3.5 个字节传输时间的通讯空闲间隔作为数据接收的开始和结束标志。由此可根据波特率计算出报文传送时间间隔, 表 5.3.2 给出了不同波特率对应的报文传送时间间隔。使用过程中可根据实际情况调整报文传送时间间隔。

表 5.3.2 报文传送时间间隔默认值

波特率 (bps)	报文传送时间间隔 (ms)	波特率 (bps)	报文传送时间间隔 (ms)
1200	32	38400	2
2400	16	57600	2
4800	8	115200	2
9600	4	500000	内部自动处理
19200	2	1000000	内部自动处理

提示：当通信数据包字节数较大时，用户可以适当增大报文传送时间间隔。

(2) 公共参数

公共参数是主从设备通信的基本参数，其中设备标识和设备描述是主控制器的设备参数，当前操作从设备须对应主控制器管理的设备编号为 0~15。

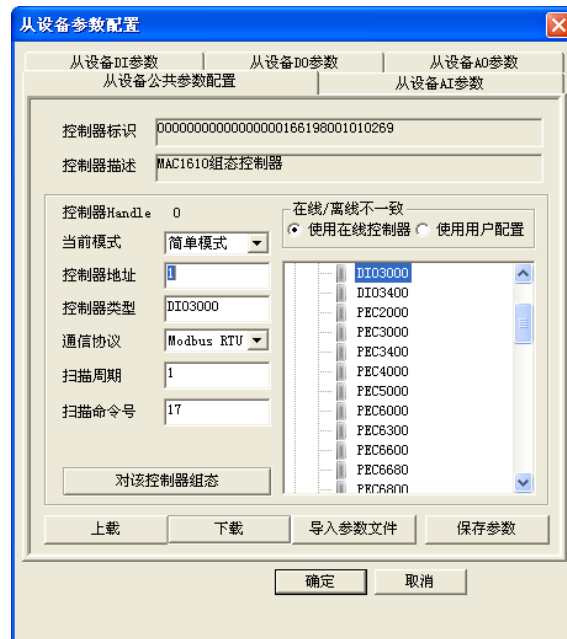


图 5.3.7 公共参数

当前模式是主从设备通信的模式选择。模式共分为 3 种：自定义模式、简单模式和完全模式。当从设备为其他设备时，只能选择自定义模式，当从设备为公司产品时可选择三种模式中的一种进行通信。使用变频器等第三方厂家设备时应使用自定义模式，根据设备需求进行所需变量的映射。自定义模式下需要用户配置所有需要映射的 AI、DI、AQ、DQ 区参数。简单模式下无需用户配置变量区参数，从设备会自动映射 I/O 数据到主控制器相应变量区中，一般情况下建议使用此种配置。完全模式下用户也无需配置变量区参数，此时从设备除映射 I/O 数据外还映射其他的内部变量区，主控制器内相关的变量区资源均会被映射完全。关于简单模式和完全模式的资源映射情况，请用户参见附录 C。

控制器地址为 Modbus 协议中主控制器与从设备通信的唯一标识，通信时主控制器以此来区分不同的从设备。通信地址与设备编号无关，不同的设备编号下可配置相同的地址，此时两个设备编号下映射的资源均为同一个从设备的资源，当主控制器内为一个从设备分配的映射资源小于用户编程所需时，可以采用此种方式来映射从设备内的其余资源。可选范围为 1~255。

设备类型是指从设备的种类，可以配置公司产品 DIO2000、TAC6000、DIO3000 等或非公司产品“其他设备”，当用户想取消原来的从设备类型配置时，可选择配置为“未组态设备”。

通信协议支持 Modbus RTU 或 Modbus ASCII，根据实际连接的从设备来选择。

扫描周期是指运行几轮的用户程序后进行一次资源映射，此参数和用户的程序运行周期有关。举例，若用户编写的用户程序运行周期为 2ms，配置的扫描周期为 10，则每 20ms 主控制器会执行一次对从设备的读命令，并根据配置进行数据刷新。注意此参数和从设备通信时间间隔不同，从设备通信时间间隔是对此串口上所有的从设备发送命令进行管理的最小时间间隔，扫描周期是指从设备所有读命令轮询一次的周期，默认

为 1。

扫描命令表示主控制器查询从设备是否上线的 Modbus 命令功能号，对于公司产品请设置为 17，其他设备可根据支持的 Modbus 命令功能号进行选择，一般为 1、2、3、4。当选择为 1 或 2 时，命令中的变量起始地址和读取个数会按照此从设备 DI 参数或 DQ 参数的第一个命令来配置，当选择为 3 或 4 时，命令中的变量起始地址和读取个数会按照从设备 AI 参数或 AQ 参数的第一个命令来配置。

提示：

①当从设备类型配置为“其他设备”时，建议选择简单模式，下载后即可正常通信进行资源映射，无需配置从设备 AI、AQ、DI、DQ 区参数；

②从设备的地址部分和设备编号没有任何联系，用户可以在范围内随意配置；

③TAC6000、TAC3000、DIO2000、DIO1000 下通信模式固定为简单模式，用户不能进行修改；

④当从设备类型为“其他设备”时，需要配置 AI、AQ、DI 或 DQ 参数才能正确发送扫描命令。

(3) 资源映射参数

控制器提供 3 种资源映射模式：自定义模式、简单模式和完全模式。当从设备为第三方厂家设备时，只能选择自定义模式，当从设备为公司设备时可选择三种模式中的一种进行通信。

● AI 参数

每个主控制器最多可以映射的从设备 AI 资源为 16 个字变量和 16 个命令，通过 AI 映射可读取从设备的变量区资源。图 5.3.8 为 AI 参数的参数配置例程。

从设备公共参数配置		从设备AI参数	
读功能号	3	3、4	
写功能号	不能写		
通道数量	8		
命令数量	1		
通道参数		寄存器偏移	设备地址
通道0			
寄存器地址	0	PAIW0, XAID0	528, 16
数据类型	无符号双...		
输入零点	-2000.00...		
输入满度	8000.000000		
输出零点	-200.000000		
输出满度	800.000000		
通道1			
寄存器地址	1	PAIW1, XAID2	529, 18
数据类型	无符号双...		
输入零点	-2000.00...		
输入满度	8000.000000		

默认I/O参数 刷新寄存器偏移

上载 下载 导入参数文件 保存参数

确定 取消

图 5.3.8 AI 参数配置例程

读功能号为 3 或 4，其余不起作用。

通道数量表示主控制器读取从设备的 AI 通道数。每个通道中的变量地址为读取的从设备 AI 区对应的 Modbus 地址。PAI 和 XAI 是读取后的通道数据在主控制器中变量映射区，其中 PAI 区存储的是对应从设备 AI 区对应的 Modbus 地址中的原始数据，XAI 对应相应的 PAI 转换数据，转换对应如图 5.3.9 所示。例如，PAI 输入的码值为-2000~8000，对应-200 度~800 度，如上图通道 0 中的输入零点、输入满度、输出零点、输出满度所示，当读入的 PAIW0 中为 2000 时，对应的 XAID0 为 200.0 度，注意 XAI 的存取方式为双字且为实数。Modbus 地址为对应的主控制器中的映射区 PAI 和 XAI 的 Modbus 地址。

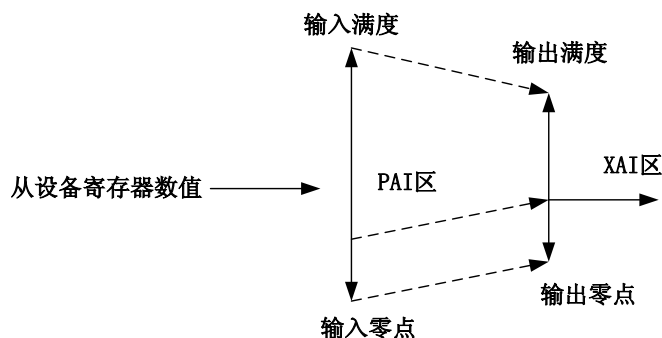


图 5.3.9 PAI 和 XAI 的转换关系

数据类型支持有符号双字节、无符号双字节、有符号四字节、无符号四字节和浮点数，当为有符号四字节、无符号四字节或实数时需要占用两个字变量。

命令数量对应为读取从设备 AI 区的 Modbus 命令个数，变量地址为 Modbus 读命令中的起始地址，返回变量个数为 Modbus 读命令中所要读取的变量个数。

例程中的配置功能为主控制器每次读取从设备变量地址为 0~7 的变量，读取后的结果放入到主控制器 PAIW0~PAIW7 中，同时将其数据值缩小 10 倍放入 XAID0~XAID7 中。

● AQ 参数

每个主控制器最多可以映射的从设备 AQ 资源为 16 个字变量和 16 个命令，通过 AQ 映射可读取或写入从设备变量区资源。图 5.3.10 为 AQ 参数的参数配置例程。

该图显示了从设备参数配置窗口，包含从设备公共参数配置、从设备 DI 参数、从设备 DO 参数和从设备 AO 参数。配置项包括读功能号、写功能号、通道数量、命令数量、通道参数、寄存器地址、数据类型、输入零点、输出满度、输出零点等。

图 5.3.10 AQ 参数配置例程

读功能号为 3 或 4，写功能号为 6 或 16。

通道数量为主控制器 Modbus 的读取或写入 AQ 通道的数量。每个通道中的变量地址为读取或写入的从设备 Modbus 地址。变量区偏移对应的 PAQ 和 XAQ 是读取后的通道数据在主控制器中的变量区映射，其中 PAQ 存储的是对应从设备 Modbus 变量地址中的原始数据，XAQ 对应相应的 PAQ 转换数据，转换对应原理同 AI 部分。Modbus 地址为对应的主控制器中的映射变量 PAQ 和 XAQ 的 Modbus 地址。

数据类型支持有符号双字节、无符号双字节、有符号四字节、无符号四字节和浮点数，当为有符号四字节、无符号四字节或实数时需要占用两个变量。

命令数量对应为读取或写入从设备 AQ 变量的总的 Modbus 命令个数，起始变量地址为 Modbus 读命令中的起始地址，返回变量个数为 Modbus 读或写命令中的读取或写入的变量个数。

AQ 参数部分映射的从设备变量是完全受主控制器中的变量数据控制。例如，例程中的配置为利用 3 号命令读取从设备变量地址为 2064 和 2065 的数据，读取完毕后与 PAQW0 和 PAQW1 变量中的数据进行比较，如果相同则不进行写入操作，如果不同则利用 6 号命令将变化的数据写入到从设备对应变量的地址中，如果两个变量比较都不同，则利用 6 号命令将变量地址为 2064 和 2065 的变量均写入数据，写入完毕后再按照一定的时间间隔再次读取从设备变量地址为 2064 和 2065 的数据再次比较。

● DI 参数

每个主控制器最多可以映射的从设备 DI 资源为 256 个位变量和 32 个命令，DI 映射功能仅为读取从设备的变量资源。图 5.3.11 为 DI 参数的参数配置例程。



图 5.3.11 DI 参数配置例程

读功能号为 1 或 2 可选，其余不起作用。

命令数量对应为读取从设备 DI 变量的总的 Modbus 命令个数，起始变量地址为 Modbus 读命令中的起始地址，返回变量个数为 Modbus 读命令中的读取变量个数。

例程中的配置功能为主控制器每次读取从设备起始变量地址为 1 开始，连续读取 16 个变量，读取后的结果放入到主控制器 XIO.0~XIO.15 的变量中。

● DQ 参数

每个主控制器最多可以映射的从设备 DQ 资源为 256 个位变量和 32 个命令，DQ 映射功能为读取和写入从设备变量资源。图 5.3.12 为 DQ 参数的参数配置例程。

读功能号	写功能号	命令数量	命令参数	寄存器偏移	设备地址
1	15	1	命令0		
			起始寄存...	16704	XQ0.0 ~ XQ0.15
			返回寄存...	16	12808 ~ 1...

图 5.3.12 DQ 参数配置

读功能号为 1 或 2 可选，写功能号为 5 或 15 可选。

命令数量对应为读取或写入从设备 DQ 变量的 Modbus 命令个数，起始变量地址为 Modbus 读命令中的起始地址，返回变量个数为 Modbus 读命令中读取或写入的变量个数。

从设备变量中 DQ 参数部分完全受主控制器中的变量数据控制。例如，例程中的配置为利用 1 号命令读取从设备变量地址为 0 开始，连续读 16 个位变量，读取完毕后与 XQ0.0~XQ0.15 进行比较，如果相同则不进行写入操作，如果不同则利用 5 号命令将变化的数据写入到从设备对应变量中，如果两个变量比较都不同，则利用 5 号命令将变量地址为 0~15 的变量均写入数据，写入完毕后再按照一定的时间间隔再次读取从设备变量地址为 0~15 的数据再次比较。

5.3.4 配置实例

● 公司产品

公司产品中多数都支持从设备功能，其中，PEC、MAC、DEC、EDC、MAC、AIO、DIO、PMC 等系列产品都可作为主从设备中的主设备或从设备。以 PEC8000 为从设备的配置步骤如下：

(1) 首先确定用户要配置的从设备在主控制器中的编号，双击将要配置的从设备，软件会弹出从设备参数配置界面，如图 5.3.13 所示。

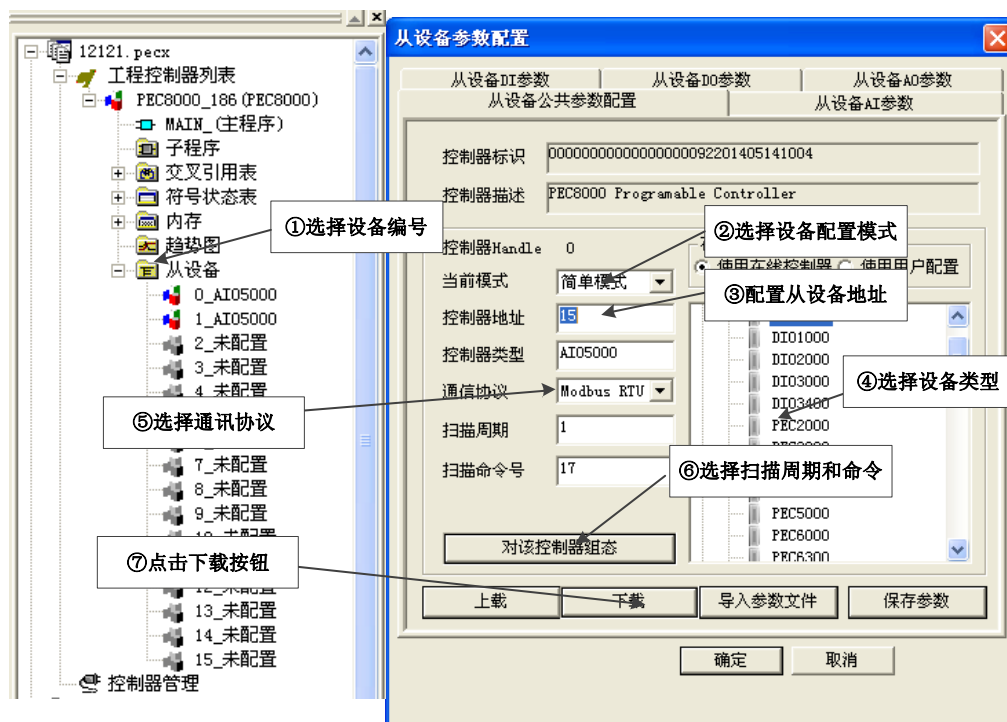


图 5.3.13 配置界面及参数

(2) 按照上图的顺序依次配置从设备的参数，将 PEC8000 的参数配置完毕后，如果从设备 PEC8000 已与主设备相应的串口连接，可采用手动扫描从设备或从设备自恢复扫描使从设备上线。



图 5.3.14 设备上线

(3) 设备上线后会看到设备的通信灯闪烁指示主从通信状态，此时用户可以根据需要对从设备资源进行映射，具体简单模式的映射资源请参看附录 C 从设备资源和特性。

(4) 简单模式下无需用户配置变量参数，从设备会自动映射 I/O 数据到主控制器相应变量中。

● 第三方产品

第三方产品是指支持标准 Modbus 协议的其他控制器，这类产品可通过 Modbus 协议以从设备的身份和 DCCE 控制器进行串口通信，实现资源映射。以台达的变频器作为从设备，配置步骤如下：

(1) 首先确定用户要配置的从设备在主控制器中的编号，图 5.3.15 为配置编号为 0 的从设备，双击将要配置的从设备，软件会弹出从设备参数配置界面。

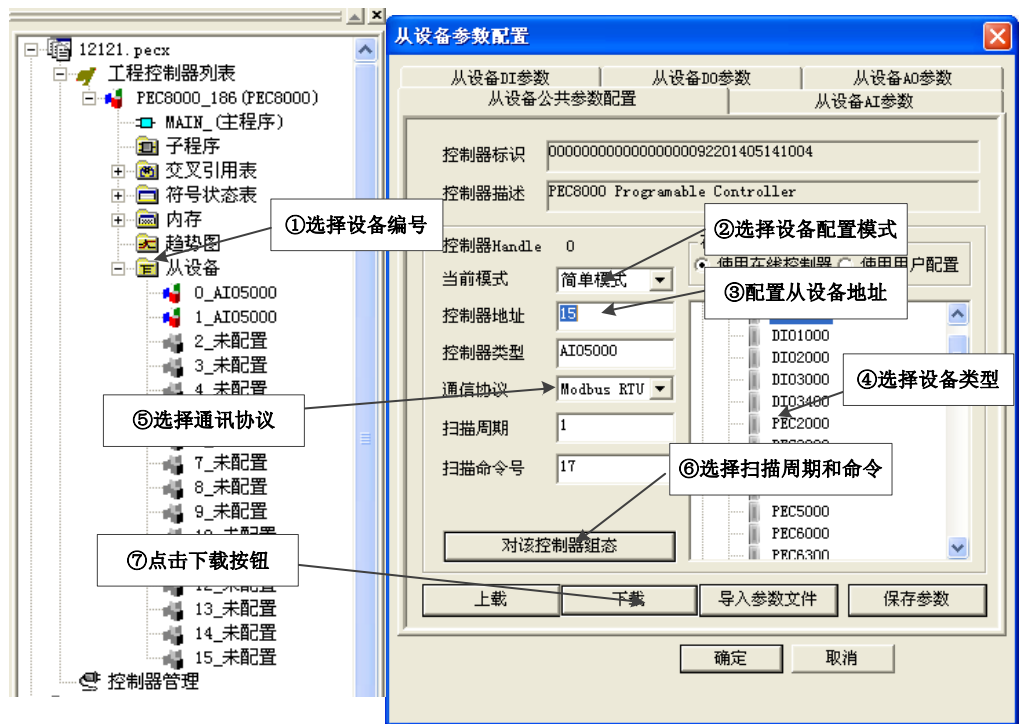


图 5.3.15 配置界面及参数

(2) 按照上图的顺序依次操作，完成了变频器从设备公共参数的配置，接下来根据变频器提供的变量说明配置 AI 参数。

从设备DI参数		从设备DO参数		从设备AO参数	
从设备公共参数配置				从设备AI参数	
	读功能号	3	3、4		
	写功能号	不能写			
	通道数量	2			
	命令数量	1			
日	通道参数		寄存器偏移	设备地址	
日	通道0				
├	寄存器地址	8848	PAIW0, XAID0	528, 16	
├	数据类型	无符号双...			
├	输入零点	-2000.00...			
├	输入满度	8000.000000			
├	输出零点	-200.000000			
├	输出满度	800.000000			
日	通道1				
├	寄存器地址	8849	PAIW1, XAID2	529, 18	
├	数据类型	无符号双...			
├	输入零点	-2000.00...			
├	输入满度	8000.000000			
日	命令参数				
日	命令0				
├	起始寄存...	8848			
├	返回寄存...	2			

图 5.3.16 AI 参数

上述参数配置完后，将变频器变量地址为 8448 和 8449 的变量映射到主控制器的 PAIW0 和 PAIW1，读取功能号为 3 号。主控制器内这 2 个变量数值会根据变频器内的变量变化而变化，从而用于主控制器编程和确定变频器在线状态。

(3) 映射变频器的 AQ 资源。

从设备公共参数配置			从设备AI参数	
从设备DI参数		从设备DO参数	从设备AO参数	
	读功能号	3		
	写功能号	6		
	通道数量	2		
	命令数量	1		
	通道参数		寄存器偏移	设备地址
[-]	通道0			
	寄存器地址	8192	PAQW0 , XAQD0	2080 , 1552
	数据类型	无符号双字节		
	输入零点	4.000000		
	输入满度	20.000000		
	输出零点	800.000000		
	输出满度	4000.000000		
[-]	通道1			
	寄存器地址	8193	PAQW1 , XAQD2	2081 , 1554
	数据类型	无符号双字节		
	输入零点	4.000000		
	输入满度	20.000000		
[-]	命令参数			
	命令0			
	起始寄存...	8192		
	返回寄存...	2		

图 5.3.17 AQ 参数

上述参数配置完后，将变频器变量地址为 8192 和 8193 的变量映射到主控制器的 PAQW0 和 PAQW1，读取功能号为 3 号，写入功能号为 6 号。变频器内变量的数值会根据主控制器内的变量数值变化而变化，当 PAQW0 或 PAQW1 的数值改变时，主控制器会自动进行检查并将变化的数值写入到变频器内的相应变量。

(4) 配置完毕后采用手动扫描从设备或从设备自恢复扫描使从设备上线，设备上线后会看到设备的通信灯闪烁指示主从通信状态，此时用户已经可以根据需要对从设备资源进行映射。



图 5.3.18 变频器上线

5.3.5 从设备组态

DCCE 网络化可编程控制器集控制单元和 I/O 单元于一身，PLC_Config 支持对设备类型为 DCCE 控制器的从设备进行直接组态编程。点击图 5.3.19 中的“对该设备组态”进入从设备编程界面。

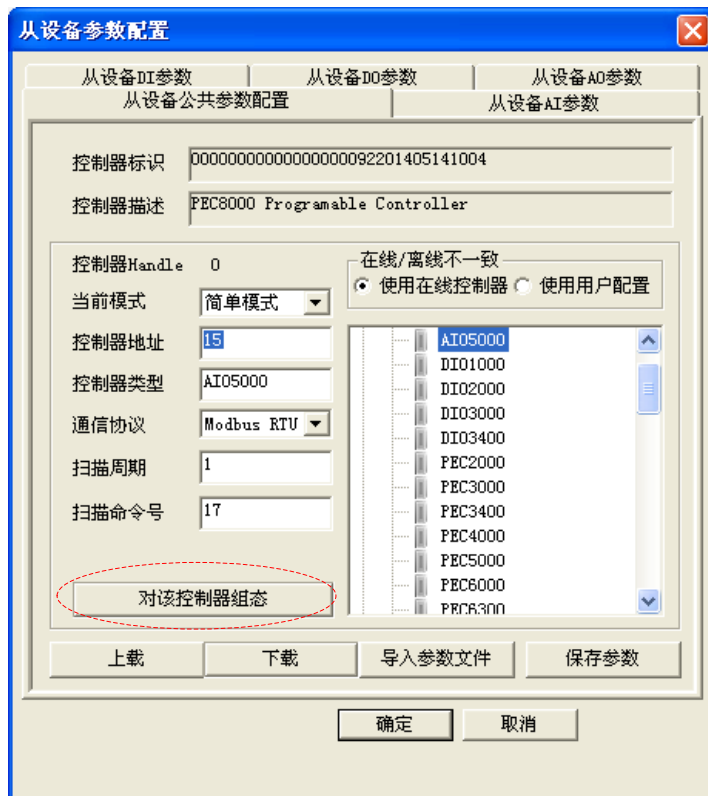


图 5.3.19 从设备编程调用方式

5.4 扩展 IO 通信系统系统

MAC 系列主控器可以与 EIO 系列产品组成环网通信资源映射系统，环网通讯可以很大的提高资源映射的效率和降低主控器的 CPU 占用率。

PLC_Config 为扩展 IO 设备提供了“IO 设备配置监控界面”，如图 5.4.1 所示，利用该界面用户可通过“手动分配”或“自动分配”按钮实现 IO 设备的配置。当主控器和扩展 IO 设备的物理连接完成，可通过“自动分配”按钮，控制主控器扫描 IO 设备、分配节点地址和完成资源映射，该界面会显示出自动分配的设备信息和设备状态，编写组态工程可参照主控器映射地址表，控制扩展 IO 设备的 IO 资源；若主控器的组态工程已写好，可通过“手动分配”按钮，手动填写 IO 设备的设备类型、物理地址和映射地址，可根据组态工程，配置 IO 设备映射地址，这样更换 IO 设备就无需修改组态工程，更方便用户使用。



图 5.4.1 IO 设备配置监控界面

编号（只读）：指设备的节点地址，范围为：1~16；

IO 地址（可读可写）：指设备的物理地址，范围为：1~16；

IO 类型：指 IO 设备的型号，设备类型是根据选择的设备类型或者是扫描读回来的报文显示出来的。在选择其中的一个类型的设备时，IO 板资源映射和主控制器资源映射会根据设备类型默认配置一个映射地址。

IO 资源（只读）：IO 板资源映射是根据选择的设备类型或者是扫描读回来的报文显示出来的。

主控制器映射地址（可读可写）：主控制器资源映射是根据设备的类型进行不同区域的映射。

设备状态（只读）：设备的状态栏中有两种底色，其中绿色底色的代表通信正常，红色底色代表通信故障或者是未通信。

开始监控（按钮）：点击“开始监控”按钮时，PLC_Config 软件会定时 2S 循环发送读取 IO 映射信息的报文，同时刷新 IO 配置界面的信息。

停止监控（按钮）：在点击“停止监控”按钮时，PLC_Config 软件会停止发送读取 IO 映射信息的报文，同时停止刷新 IO 配置界面信息。

自动扫描按钮（按钮）：在点击“自动分配”按钮时，PLC_Config 软件会发送一条自动扫描命令，之后定时 2S 循环读取 IO 设备的映射信息，此功能可以完成 IO 设备的自动分配和刷新功能。在设备未完成自动分配时，“自动分配”按钮会被禁用，自动分配完成之后会变成可用状态。

(1) 参数配置

主控制器通过环网通讯协议最多可扩展 16 个从设备，用户可以通过 PLC_Config 中的设备参数界面配置从设备通信参数，配置示例图 5.4.2 所示。

在配置完成主控器的模式后，给主控制器模块切换到设置状态（电源端子中 SET 端和 V-端短接），然后给主控制器以及扩展 IO 模块重新上电，主控制器会将从设备的波特率统一改成与主设备相同的波特率，配置完成后会有四个灯闪烁，有几个灯长亮。长亮的灯代表修改扩展 IO 模块的波特率成功的地址。

1号串口设置	
主从设置	主
通信协议	扩展IO
通信波特率	1M bps
通讯基本设置	N, 8, 1
数据包间隔时间 (ms)	2
从设备通信最大允许错误次数	10
从设备通信时间间隔 (ms)	30
从设备通信最大等待时间 (ms)	60

图 5.4.2 参数配置图

(2) 自动扫描

在 IO 设备的波特率配置成功后，将主控制器模块调整到运行状态，双击工程树中的“IO 映射管理”栏，弹出“IO 设备配置监控界面”，如图 5.4.3 所示，点击“自动扫描”按钮，在等待 15S 后即可完成 IO 设备的自动分配，自动分配在进行地址映射的时候同一种类型的设备在主设备的映射地址是连续的。

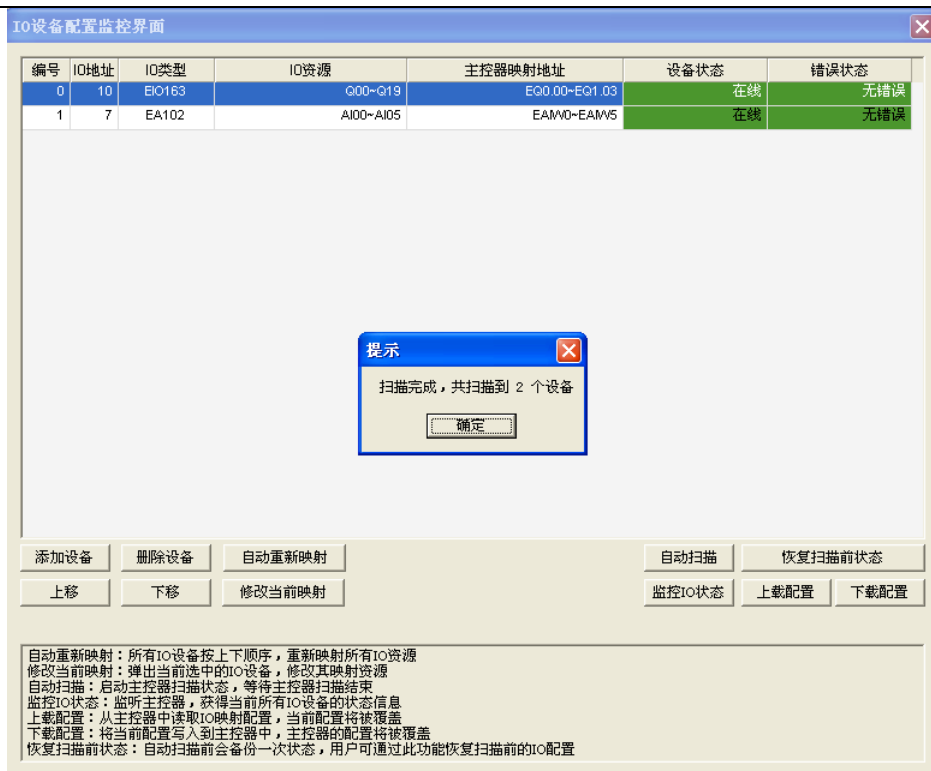


图 5.4.3 自动分配配置图

(3) 手动分配地址

当自动分配的地址与组态工程编写的映射区不同时, 可以通过手动分配修改 IO 设备的映射地址, 点击“修改当前映射”按钮软件会弹出“配置通道映射”界面, 如图 5.4.4 所示, 只需要在界面中配置首地址的字偏移和位偏移和映射长度。在配置完成所有的 IO 设备映射地址后, 点击“下载配置”按钮, 软件会检测 IO 设备的映射资源是否正确, 如果正确会将配置的映射信息下载到主控制器完成新的地址资源映射。

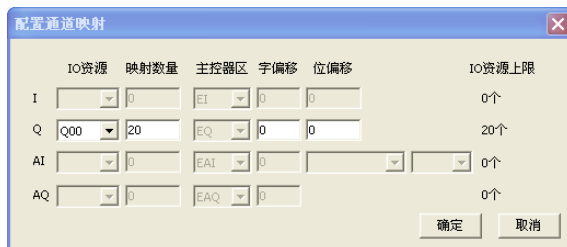


图 5.4.4 手动分配配置图

(4) 统一配置波特率

MAC 主控制器支持 IO 扩展通信协议, 由于 IO 设备默认为 1M 波特率通信; 为了方便现场波特率的修改, 支持主控制器对 IO 设备波特率的统一修改。

步骤如下:

- ①将主控制器与 IO 设备通过底板总线相连接;
- ②将接线端子的 SET 和 V-端短接, 对所有设备上电;
- ③观察主设备串口灯闪烁, 代表主控制器正在逐个发送配置波特率报文, 待配置完毕, 主设备的 I20~I23 灯闪烁, 代表波特率统一配置完毕;
- ④主设备的 I00~I15 灯分别代表物理地址为 1~16 的 IO 设备, 对应的 I 灯亮, 代表该地址的设备波特率配置成功。

(5) IO 设备故障表

根据工程经验，总结了主控器由于设备出错，或者误操作导致的错误情况进行分析，给现场工程师查找问题提供一个依据。

表 5.4.1 错误码参照表

错误类型		现象	处理方式
设备掉线，通信出现问题		设备报错,其他设备正常通信	转为点对点通信
(点对点通信) 添加设备地址存在的设备		两个设备地址冲突，该地址设备报错	清除该设备 IO 输出
(点对点通信) 添加设备地址不存在的设备		该设备不会上线，其他设备正常通信	不做处理
(环网通信) 添加设备地址存在或者不存在的设备		该设备不会上线，其他设备正常通信	不做处理
(上电前) 换一个地址存在的设备，但类型不对的设备		该地址设备报错	转为点对点通信，每隔 10 轮,扫描一次
(上电前) 添加设备地址存在的设备		两个设备地址冲突，该地址设备报错	清除该设备 IO 输出
(上电前) 添加设备地址不存在的设备		该地址设备通过功能块告知该设备地址	其他设备正常走环网通信
一个设备下线	添加该地址设备（类型对）	设备上线	重新扫描回到环网模式
	添加该地址设备（类型不对）	该地址设备报错，转为点对点通信	每隔 10 轮,扫描一次
	添加存在的设备地址（类型对或者不对）	两个设备地址冲突，该地址设备报错	清除该设备 IO 输出
	添加设备地址不存在的设备（类型对或者不对）	不会报错，该设备不会上线	其他设备正常通信

6 可编程控制指令集

6.1 指令集概述

控制器共支持 193 个可编程控制指令，共分为 14 类。具体类别为：位逻辑指令、网络通信指令、比较指令、转换指令、计数器指令、实数运算指令、整数运算指令、逻辑操作指令、变量赋值指令、程序控制指令、中断指令、移位与循环指令、定时器指令和专用指令。

- **位逻辑指令**实现与、或、置位和复位等位运算功能。
- **网络通信指令**实现 Modbus、MPI、UDP、TCP、自由通信等协议的网络通信。
- **比较指令**实现字、整数、长整和实数的比较功能。
- **转换指令**实现实数与整数、整数与 BCD 码、长整与实数和长整与整数的相互转换功能。
- **计数器指令**实现边沿计数、高速输入和高速输出的计数功能。
- **实数运算指令**实现实数运算功能，如 PID 运算、四则运算等。
- **整数运算指令**实现整数数学计算、整数递增和递减等运算功能。
- **逻辑操作指令**实现字、双字的取反、与、或和异或等逻辑功能。
- **变量赋值指令**实现变量的字、双字、实数、字块的赋值和转移功能，以及字的立即传送功能。
- **程序控制指令**实现程序的循环、跳转和子程序调用功能。
- **中断指令**实现中断的允许、禁止、连接和分离功能。
- **移位与循环指令**实现字、双字的左移、右移、循环左移和循环右移功能，以及寄存器移位和交换功能。
- **定时器指令**实现 10ms、100ms 时基的定时接通、断开、保持定时接通等定时功能和中断功能。
- **专用指令**实现工件检测、数据校验、经济运行、存储、冒泡排序、斜率检测、定位、往复运动、PH 检测、电导率检测、溶氧检测、Na 离子检测、气缸控制和 IO 监控等特殊功能。

表 6.1.1 指令索引表

标识符	说明	标识符	说明
AND	位与	TAN	正切
OR	位或	LN	自然对数
P	上升沿	EXP	自然幂指数
N	下降沿	ARC_SIN	反正弦
=	输出	ARC_COS	反余弦
=I	立即输出	ARC_TAN	反正切
S	置位	ABS	绝对值
SI	立即置位	I_PID	增量式 PID
R	复位	P_PID	位置式 PID
RI	立即复位	ADD_I	整数加
RS	复位优先触发	ADD_DI	长整加
SR	置位优先触发	SUB_I	整数减
NETR_B	EPA 位读	SUB_DI	长整减
NETR_B_P	EPA 位确认读	MUL	整数乘
NETR_W	EPA 字读	MUL_I	整数短乘
NETR_W_P	EPA 字确认读	MUL_DI	长整乘
NETR_D	EPA 双字读	DIV	整数除
NETR_D_P	EPA 双字确认读	DIV_I	整数整除
NETW_B	EPA 位写	DIV_DI	长整整除
NETW_B_P	EPA 位确认写	INC_I	整数加 1
NETW_W	EPA 字写	INC_W	字加 1
NETW_W_P	EPA 字确认写	INC_DW	双字加 1
NETW_D	EPA 双字写	DEC_I	整数减 1
NETW_D_P	EPA 双字确认写	DEC_W	字减 1
SCR_B	串行位读	DEC_DW	双字减 1

SCR_W	串行字读	INV_W	字取反
SCW_B	串行位写	INV_DW	双字取反
SCW_W	串行字写	WAND_W	字与
XMT	串行自由发送	WAND_DW	双字与
RCV	串行自由接收	WOR_W	字或
SC_AFC	串行自由发送接收	WOE_DW	双字或
USS_INIT	USS 初始化	WXOR_W	字异或
USS_WPM_W	USS 字写	WXOR_DW	双字异或
USS_WPM_DW	USS 双字写	MOV_B	字节赋值
USS_WPM_R	USS 实数写	MOV_W	字赋值
USS_RPM_W	USS 字读	MOV_DW	双字赋值
USS_RPM_DW	USS 双字读	MOV_R	实数赋值
USS_RPM_R	USS 实数读	BLKMOV_W	字块赋值
USS_CTRL	USS 控制	BLKMOV_DW	双字块赋值
NET_XMT	UDP 自由发送	MOV_WIR	立即传送读
NXMT_SEND	UDP 自由确认发送	MOV_WIW	立即传送写
NXMT_RCV	UDP 自由确认接收	FMOV_W	多点变量赋值
UDP_AFC	UDP 自由发送接收	BSET_W	字批量赋值
TCP_LINK	TCP 连接	BSET_DW	双字批量赋值
TCP_SEND	TCP 自由发送	FOR	循环头
TCP_RCV	TCP 自由接收	NEXT	循环尾
TCP_AFC	TCP 自由发送接收	JMP	跳转头
X_GET_BIT	MPI 位读	LBL	跳转尾
X_GET_BYTE	MPI 字节读	GPROGRUN	G 代码运行控制
X_GET_WORD	MPI 字读	CALL	子程序调用
X_GET_DWORD	MPI 双字读	ENI	中断允许
X_PUT_BIT	MPI 位写	DISI	中断禁止
X_PUT_BYTE	MPI 字节写	ATCH	中断连接
X_PUT_WORD	MPI 字写	DTCH	中断分离
X_PUT_DWORD	MPI 双字写	SHL_W	字左移
==W、==I	相等	SHL_DW	双字左移
<>W、<>I	不等于	SHR_W	字右移
>=W、>=I	大于等于	SHR_DW	双字右移
<=W、<=I	小于等于	ROL_W	字循环左移
>W、>I	大于	ROL_DW	双字循环左移
<W、<I	小于	ROR_W	字循环右移
==D、==R	相等	ROR_DW	双字循环右移
<>D、<>R	不等于	SHRB	移位寄存器
>=D、>=R	大于等于	SWAP	字节交换
<=D、<=R	小于等于	EXCHG	字交换
>D、>R	大于	TON	定时接通
<D、<R	小于	TONR	定时断开
B_W	字节转字	TOF	保持定时接通
W_B	字转字节	WPC	工件检测
R_I	实数整数转换	PS	经济运行
I_R	整数实数转换	STORE	数据存储
I_DI	整数长整转换	B_SortW	字冒泡排序
DI_I	长整整数转换	B_SortD	双字冒泡排序

DI_R	长整实数转换	B_SortF	实数冒泡排序
R_DI	实数长整转换	CKSUM_G	数据校验码生成
BCD_I	BCD 整数转换	CKSUM_C	数据校验码检查
I_BCD	整数 BCD 转换	ZRN	原点回归
ROUND	四舍五入	DRVA	绝对定位
TRUNC	取整	DRVI	相对定位
CTU	上升计数	RSI	读取当前值
CTD	下降计数	PLSV	旋转方向脉冲输出
CTUD	上升下降计数	PLSE	脉冲输入输出
HDEF	高速计数	SLC	斜率检测
PLS	高速脉冲输出	PH	PH 检测
ADD_R	实数加	DD	电导率检测
SUB_R	实数减	OXM	溶氧检测
MUL_R	实数乘	Na	Na 离子检测
DIV_R	实数除	RECIPROCATATE	往复运动
SQRT	开方	PumpManual	气缸控制
SIN	正弦	IOMonitor	IO 监控
COS	余弦	--	--

6.2 指令描述

控制器支持梯形图和功能块两种编程方式，且梯形图方式下除位逻辑指令中与、或指令用常开、常闭指令替换外，支持所有的功能块指令，但在两种编程方式下指令显示方式会不同。

如图 6.2.1 所示，一个典型的功能块指令由指令名称、操作数名称、操作数（指令引脚）、能流组成，梯形图指令由能流、操作数组成。

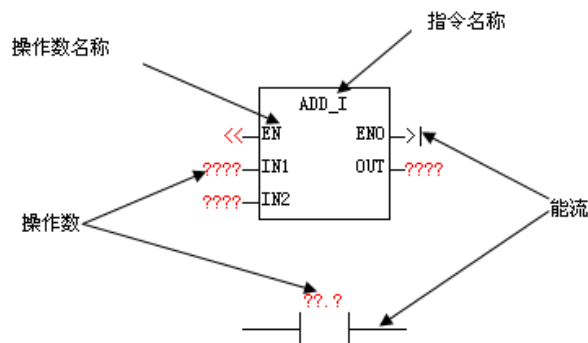


图 6.2.1 指令描述

操作数名称用于表示操作数的含义，便于用户理解操作数功能，同一指令的不同操作数的名称各不相同。操作数又称指令引脚，作为指令的输入输出接口，位于指令左侧的操作数称为输入操作数（输入引脚），位于指令右侧的操作数称为输出操作数（输出引脚），每个操作数都有自己支持的变量区类型和数据类型，可参见每个指令的操作数说明，表 6.2.1 进行了举例说明。能流是指令的通用操作数，多数作为指令的使能引脚，通过能量的流动，指令的输入能流会传递给输出能流，该输出能流可作为下一指令的输入能流，实现指令之间能流互通。

指令的输入能流的操作数名称为 EN，输出能流的操作数名称为 ENO，在下文中的功能块说明中将不再累述。

表 6.2.1 操作数指示说明

操作数	操作数有效区域	数据类型
IN1、IN2	IW、QW、VW、LW、MW、SW、SMW、T、C、AIW、AQW、XIW、XQW、PAIW、PAQW、*MD、*VD、常数	INT
OUT	QW、VW、LW、MW、SW、SMW、AQW、XQW、PAQW、*MD、*VD	INT
EN	I、Q、V、L、M、S、SM、T、C、AI、AQ、XI、XQ、PAI、PAQ	BOOL

ENO	Q、V、L、M、S、SM、AQ、XQ、PAQ	BOOL
???	I、Q、V、L、M、S、SM、T、C、AI、AQ、XI、XQ、PAI、PAQ	BOOL
????	QW、VW、LW、MW、SW、SMW、AQW、XQW、PAQW、*MD、*VD	INT

注意：

1、操作数使用 T 区或 C 区时，若操作数的数据类型为 BOOL，表示该操作数为定时器或计数器到达标志位，若操作数的数据类型为 WORD，表示该操作数为定时器或计数器的当前计数值。

2、控制器的 CPU 均为 32 位处理器，故根据 32 位实数的规定，实数的有效位数为 6~7 位。

编程时可使用绝对地址或变量符号两种方式填写指令参数。在未填写任何参数时，指令的引脚会显示成默认状态。

<< 能流输入，可输入位变量地址、位类型符号或连线，可以取反

>> 能流输出，可输入位变量地址、位类型符号或连线

>| 能流输出，与>>不同的是，该类型的参数可以不填写

???.? 位地址，可输入位变量地址、位类型符号

???? 非位地址，可输入字、双字类型的变量地址或符号

在指令的使用中，指令的触发方式可分为高电平触发和上升沿触发两种。

● 高电平触发

对于高电平触发的指令，当使能引脚为“1”时，指令将一直处于工作状态，直到使能引脚为“0”，指令停止工作。使用电平触发的指令时，若只想让指令工作一个循环周期，可通过上升沿触发指令控制使能引脚。

● 上升沿触发

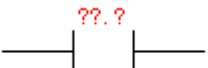
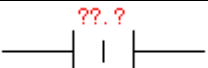
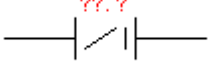
对于上升沿触发的指令，当使能引脚的上升沿产生时指令使能，执行逻辑功能。上升沿触发时，则输出的状态值只在执行完成的一个周期内有效。

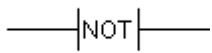
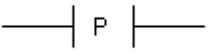
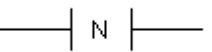
6.3 梯形图指令

梯形图指令是一种与电路图类似的图形化编程语言，很多方面与由继电器和电子元件组成的硬件回路相像，非常直观地将程序与现场控制回路结合起来，简单易学，使用方便。

在梯形图编辑模式下，控制器提供了触点指令和线圈指令。此外，梯形图模式下还支持所有的功能块指令，功能块指令将在本章节的后续内容中详细介绍。

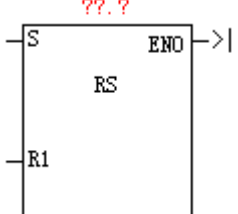
● 触点指令

常开指令	
	控制能流的导通和关闭，当???.?值为 1 时，常开触点闭合，能流导通，为 0 时，常开触点断开，能流截止。
常闭指令	
	反向控制能流的导通和关闭，当???.?值为 1 时，常闭触点闭合，能流截止，为 0 时，常闭触点断开，能流导通。
立即常开指令	
	控制能流的立即导通和关闭，与常开指令相比，该指令的头顶参数若为物理输入 I 区，可直接从物理输入点取值，实现立即控制功能。
立即常闭指令	
	反向控制能流的立即导通和关闭，与常闭指令相比，该指令的头顶参数若为物理输入 I 区，可直接从物理输入点取值，实现立即控制功能。
取反指令	
	控制能流的取反输出，将输入能流取反给输出能流。

			
上升沿触发指令		作为上升沿触发开关，当输入能流由低到高时，输出能流接通一个扫描周期。	
			
下降沿触发指令		作为下降沿触发开关，在输入由高到低跳变以后，输出能流接通一个扫描周期。	
			
操作数	变量区	数据类型	数据范围
???	位变量	BOOL	0 或 1

● 线圈指令

输出指令	输出能流状态，将输出逻辑值写入指定变量区。
立即输出指令	立即输出能流状态，与输出指令相比，该指令的头顶参数若为物理输出 Q 区，可直接给物理输出点，实现立即输出功能。
置位指令	实现置位功能，若输入能流为导通状态，则以指定位变量为起始位，将指定数量的位变量全部置位，若输入能流为关闭状态，则位变量的状态保持不变。 ???.?为起始位变量。 ????为置位数量，其数据范围为 1~255。
立即置位指令	实现立即置位功能，与置位指令相比，该指令只能对物理输出 Q 区进行置位操作。 ???.?为起始位变量，只能使用 Q 区位变量。 ????为置位数量，其数据范围为 1~255。
复位指令	实现复位功能，若输入能流为导通状态，则以指定位变量为起始位，将指定数量的位变量全部复位，若输入能流为关闭状态，则位变量的状态保持不变。 ???.?为起始位变量。 ????为复位数量，其数据范围为 1~255。
立即复位指令	实现立即复位功能，与复位指令相比，该指令只能对物理输出 Q 区进行复位操作。 ???.?为起始位变量，只能使用 Q 区位变量。 ????为复位数量，其数据范围为 1~255。
置位优先触发指令	实现置位优先触发功能，当 S1 为 1 时，无论 R 为 1 或 0，触发输出都为 1；当 S1 为 0 时，若 R 为 0 则输出状态不变，若 R 为 1 则输出为 0。 ???.?为记录转换状态的位变量。
复位优先触发指令	实现复位优先触发功能，当 R1 为 1 时，无论 S 为 1 或 0，

		触发输出都为 0；当 R1 为 0 时，若 S 为 0 则输出状态不变，若 S 为 1 则输出为 1。 ???为记录转换状态的位变量。	
操作数	变量区	数据类型	数据范围
???	位变量	BOOL	0 或 1
???	字变量、常数	WORD	1~255

● 详细说明

在梯形图编译模式下，使用上述指令时，请在指令树的位逻辑子树中寻找。梯形图程序中，触点指令作为控制逻辑的输入条件，控制能流的开关。线圈指令作为逻辑输出的结果，输出过程中的变化结果。

梯形图指令采用堆栈处理方式执行，编译完毕后会形成对应的指令码执行。指令码堆栈描述表请参见附录 D 指令码堆栈描述表。

● 应用示例

梯形图指令解析顺序按照由左上向右下的顺序解析执行，对应指令码如图 6.3.1 所示：

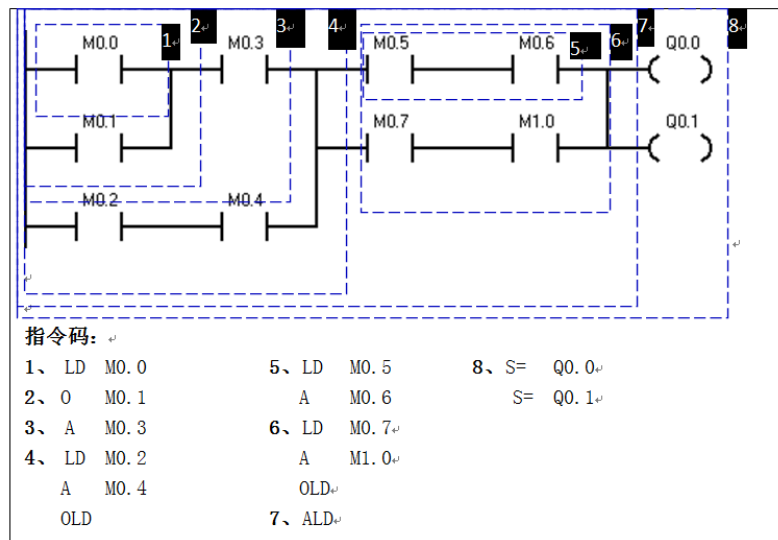


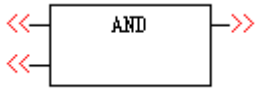
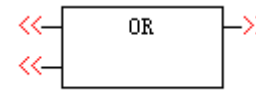
图 6.3.1 梯形图顺序解析示意图

6.4 位逻辑指令

位逻辑指令只要用于实现位操作的与、或、沿触发器、输出、置位、复位和 RS 触发器等功能。

6.4.1 与和或

● 指令参数

与指令		控制位逻辑的与、或运算，对输入的位变量进行与、或操作。 两输入与输出的关系如下： OUT = IN1 & IN2(逻辑与)，OUT = IN1 IN2(逻辑或) <table><tr><td>IN1</td><td>IN2</td><td>OUT</td><td>IN1</td><td>IN2</td><td>OUT</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>						IN1	IN2	OUT	IN1	IN2	OUT	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	1	1	1	1	1	1	1
IN1	IN2							OUT	IN1	IN2	OUT																										
0	0							0	0	0	0																										
0	1							0	0	1	1																										
1	0	0	1	0	1																																
1	1	1	1	1	1																																
																																					
或指令																																					
																																					
参数名	变量区	数据类型		说明																																	
IN	位变量	BOOL		要赋值的参数																																	
OUT	位变量	BOOL		赋值后的结果																																	

● 详细说明

与、或指令的输入引脚均可调，最少支持 1 个输入引脚，最多支持 8 个输入引脚，可通过 PLC_Config 软件添加删减输入引脚。该类指令的输入引脚还支持取反功能。

● 应用示例

如图 6.4.1，分别利用 AND 和 OR 指令对不同的变量区操作，将输出结果送入物理输出变量区，控制 LED 灯亮灭。并通过点击 PLC_Config 软件的  增加输入量、点击  减少输入量，实现不同输入变量的与或运算。



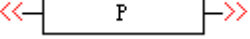
图 6.4.1 与或指令示例

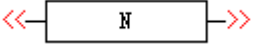
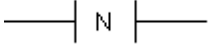
左侧的例程中，网络 1 的 AND 指令只有 1 个输入引脚，利用 M0.00 控制 Q0.00 的输出状态；网络 2 中，AND 指令有 2 个输入引脚，输入变量分别为 M0.01 和 M0.02，结果送入变量区 Q0.01；网络 3 中，将 AND 指令的一个输入引脚进行了取反操作，当 M0.03 为 0 时，对应的输入引脚为 1，此时，若 M0.02 为 1，则输出变量区 Q0.02 为 1；网络 4 中，AND 指令有 8 个输入引脚，当输入端不全为 1 时，Q0.03 为 0，反之为 1。

右侧的例程中，网络 1 的 OR 指令只有 1 个输入引脚，利用 M0.00 控制 Q0.00 的输出状态；网络 2 中，OR 指令有 2 个输入引脚，输入变量分别为 M0.01 和 M0.02，结果送入变量区 Q0.01，当 M0.01 或 M0.02 中有一个为 1 时，Q0.01 都为 1；网络 3 中，将 OR 指令的一个输入引脚进行了取反操作，当 M0.03 为 0 时，对应的输入引脚为 1，此时，无论 M0.02 为 1 或 0，输出变量区 Q0.02 都为 1；网络 4 中，OR 指令有 8 个输入引脚，只要有一个输入引脚为 1 时，Q0.03 就为 1。

6.4.2 沿触发器

● 指令参数

上升沿触发指令 (P)		作为上升沿触发开关，当输入变量由 0 跳变为 1 时，输出变量将维持一个扫描周期的高电平输出。输入变量的跳变时间必须维持一个扫描周期的时间，否则沿会丢失。
		
下降沿触发指令 (N)		作为下降沿触发开关，当输入变量由 1

			跳变为 0 时，输出变量将维持一个扫描周期的高电平输出。输入变量的跳变时间必须维持一个扫描周期的时间，否则沿会丢失。
参数名	变量区	数据类型	说明
IN	位变量	BOOL	要赋值的参数
OUT	位变量	BOOL	赋值后的结果

- 详细说明

沿触发指令主要用于实现一些指令的单周期运行，可将电平触发指令改为上升沿或下降沿触发方式。控制器最多支持 512 个上升沿和下降沿指令。

- 应用示例

如图 6.4.2 所示，利用上升沿指令作为整数加指令使能开关，实现变量 MW10 的上升沿加 1 功能；利用下降沿指令作为整数加指令使能开关，实现变量 MW20 的下降沿加 1 功能。

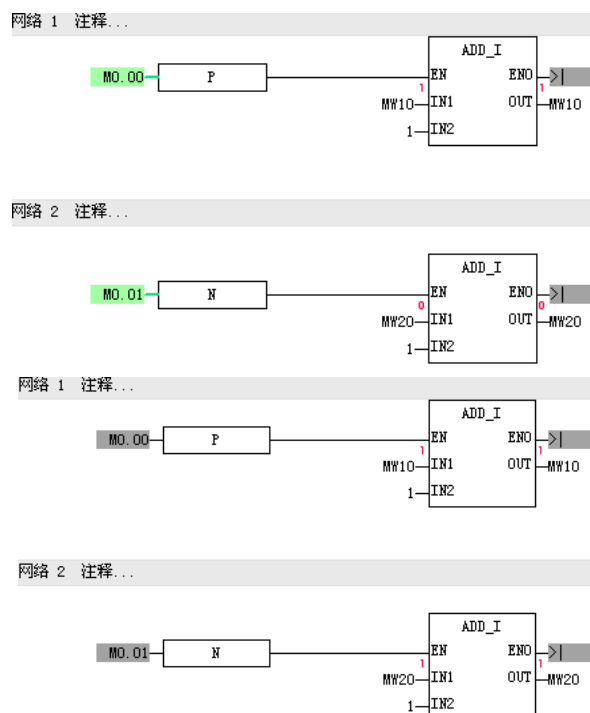


图 6.4.2 沿触发指令示例

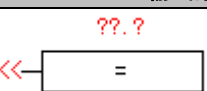
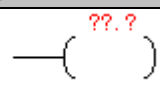
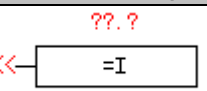
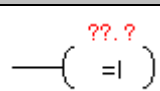
程序运行前，M0.00，M0.01 均为 0，网络 1 和网络 2 中的整数加指令不执行，MW10 和 MW20 不自加。

当 M0.00 和 M0.01 为 1 时，产生上升沿脉冲，网络 1 中，上升沿指令输出 1，整数加指令使能一次，MW10 自加 1 次，变为 1；网络 2 中，下降沿指令输出为 0，整数加指令不使能，MW20 不自加，仍为 0。

当 M0.00 和 M0.01 重新变为 0 时，产生下降沿脉冲，网络 1 中，上升沿指令输出 0，整数加指令不使能，MW10 不自加，仍为 1；网络 2 中，下降沿指令输出为 1，整数加指令使能一次，MW20 自加 1 次，变为 1。

6.4.3 输出

- 指令参数

输出指令 (EQU)		实现输出功能，将输出逻辑值写入指定变量区。	
			
立即输出指令 (EQUI)		实现立即输出功能，与输出指令相比，该指令的头顶参数若为物理输出 Q 区，可直接给物理输出点。	
			
参数名	变量区	数据类型	说明

IN	位变量	BOOL	要赋值的参数
???	位变量	BOOL	立即输出指令时为过程映像存储区 Q 区

- 详细说明

输出指令主要用于实现逻辑输出功能，将输入的位变量赋给头顶参数对应的位变量。若要求立即输出到物理输出点，则可选择立即输出指令，该指令将自动调用物理刷新函数，实现立即输出功能。

- 应用示例

如图 6.4.3 所示，调用 1 个立即输出指令和 1 个输出指令，实现 Q0.00 和 Q0.01 的输出功能，为了区分立即输出和正常周期输出的不同，本示例通过 FOR 指令做了一个周期延时，FOR 指令后面的循环体是 1000 个 AND 指令，循环体共执行 3000 次。

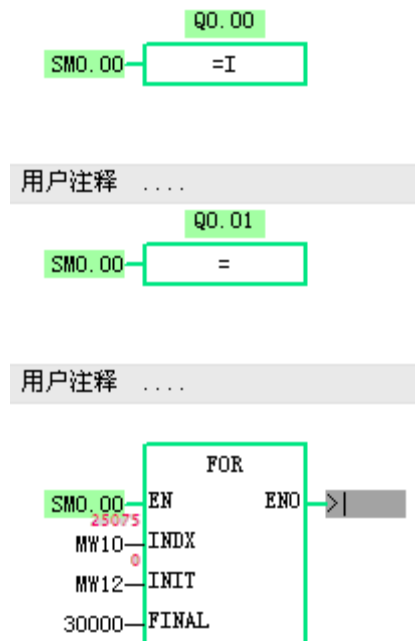


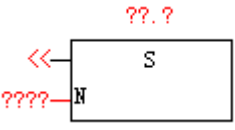
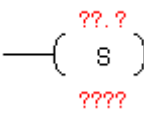
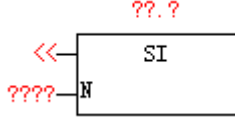
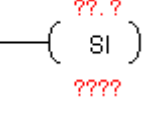
图 6.4.3 输出指令示例

程序首次运行时，可以观察到变量 Q0.00 和 Q0.01 都为 1，但用万用表测量其对应的物理输出时，Q0.00 对应的物理输出点立即输出，而 Q0.01 对应的物理输出点要等到 FOR 循环完成，整个程序完成 1 周期运算后才输出，说明立即输出功能是不会受程序的运行周期所影响的，使用时请根据需要选择。

6.4.4 置位和复位

6.4.4.1 置位和立即置位指令（S 和 SI）

● 指令参数

置位指令 (S)			实现置位功能，若输入使能为 1，则以指定位变量为起始位，将指定数量的位变量全部置位，若输入使能为 0，则位变量的状态保持不变。
			
参数名	变量区	数据类型	
EN	位变量	BOOL	指令使能位
N	字变量、常数	WORD	置位个数 1~65535
???	位变量	BOOL	起始位变量
立即置位指令 (SI)			实现立即置位功能，该指令可立即对 Q 区变量进行置位操作，不受功能块执行影响。
			
操作数	变量区	数据类型	
EN	位变量	BOOL	指令使能位
N	字变量、常数	WORD	立即置位个数 1~65535
???	位变量	BOOL	起始位变量，只能使用 Q 区位变量

● 详细说明

置位/立即置位指令主要用于实现逻辑控制中的置位功能，通过该指令可对多个连续的位变量进行置位操作，适用于开关量控制场合。立即置位指令只对 Q 区有效，可直接操作物理输出点，实现立即置位。

● 应用示例

如图 6.4.4 所示，利用置位指令点亮 Q0.00~Q0.01 对应的 2 个 LED 灯，利用立即置位指令点亮 Q0.03~Q0.07 对应的五个 LED 灯。

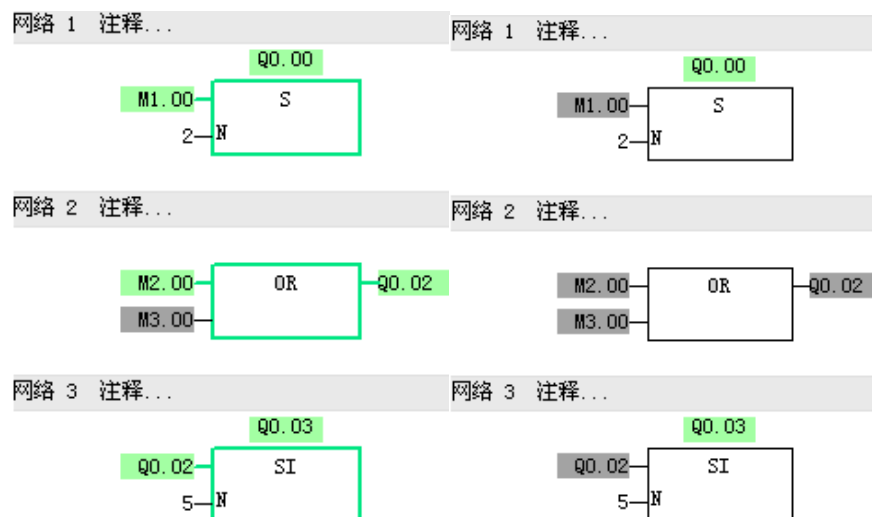


图 6.4.4 置位指令示例

程序运行前，M1.00、M2.00 为 0，Q0.00~Q0.07 对应的 LED 不亮。

程序运行后，强制 M1.00 为 1，S 指令置位，Q0.00~Q0.01 对应的 LED 灯点亮；同时，M1.00 与 M2.00 进行或运算后，输出到 Q0.02，Q0.02 为 1，SI 指令置位 Q0.03~Q0.07，Q0.03~Q0.07 对应的 LED 被点亮。

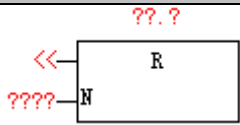
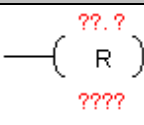
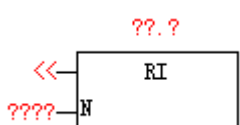
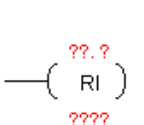
然后强制 M1.00 为 0，S 指令能够保持上一次置位状态，Q0.00~Q0.1 对应的 LED 仍点亮；强制 M2.00

为 0，或运算后 Q0.02 为 0，SI 指令能够保持上一次置位状态，Q0.03-Q0.07 对应的 LED 仍点亮。

若验证立即置位功能，可参见 6.4.3 章节的应用示例。

6.4.4.2 复位和立即复位指令（R 和 RI）

● 指令参数

复位指令（R）				
				实现复位功能，若输入使能为 1，则以指定位变量为起始位，将指定数量的位变量全部复位，若输入使能为 0，则位变量的状态保持不变。
参数名	变量区	数据类型	说明	
EN	位变量	BOOL	使能位	
N	字变量、常数	WORD	复位个数 1~65535	
???	位变量	BOOL	起始位变量	
立即复位指令（RI）				
				实现立即复位功能，与复位指令相比，该指令可立即对 Q 区变量进行复位操作，不受功能块执行影响。
参数名	变量区	数据类型	说明	
EN	位变量	BOOL	使能位	
N	字变量、常数	WORD	立即复位个数 1~65535	
???	位变量	BOOL	起始位变量，只能使用 Q 区位变量	

● 详细说明

复位/立即复位指令主要用于实现逻辑控制中的复位功能，通过该指令可对多个连续的位变量进行复位操作，该指令还可以对定时器和计数器进行复位，若复位的起始位变量是 C 或 T 时，通过复位指令可将其的状态位和当前值都清零，若 N 大于 1，则可以同时复位多个 C 或 T。立即置位指令只对 Q 区有效，可直接操作物理输出点，实现立即复位。

● 应用示例

如图 6.4.5 所示，利用置位指令点亮 Q0.01~Q0.05 对应的 5 个 LED 灯，利用复位指令熄灭 Q0.01~Q0.05 对应的 5 个 LED 灯；并利用立即复位指令对 M1.00 复位，观察是否能够复位成功。

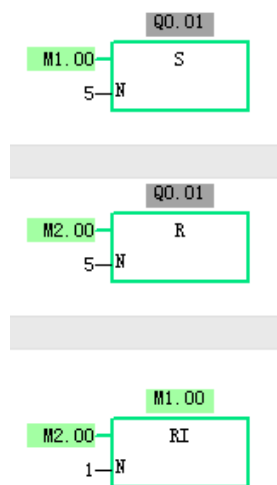


图 6.4.5 复位指令示例

程序运行前，M1.00、M2.00 为 0，Q0.01~Q0.05 为 0，Q0.01~Q0.05 对应的 LED 灯不亮。

程序运行后，强制 M1.00 为 1，Q0.01~Q0.05 被置位，Q0.01~Q0.05 对应的 LED 灯点亮。强制 M1.0 为 0，

Q0.1~Q0.05 保持置位状态，Q0.01~Q0.05 对应的 LED 灯点亮。强制 M2.00 为 1，Q0.1~Q0.05 被复位，Q0.01~Q0.05 对应的 LED 灯熄灭。但是强制 M2.00 为 1 后，立即复位指令中的 M1.00 没有复位，原因是立即复位不能对除本地数字量以外的其它变量复位。

6.4.5 触发指令

● 指令参数

置位优先触发 (SR)																
??? <<S1 <<R SR >	实现置位优先触发功能，当 S1 为 1 时，无论 R 为 1 或 0，触发输出都为 1；当 S1 为 0 时，若 R 为 0 则输出状态不变，若 R 为 1 则输出为 0。 输入与输出的关系如下： <table><tr><th>S1</th><th>R</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>不改变</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	S1	R	OUT	0	0	不改变	0	1	0	1	0	1	1	1	1
S1	R	OUT														
0	0	不改变														
0	1	0														
1	0	1														
1	1	1														
??? -S1 ENO > SR -R																
参数名	变量区	数据类型	说明													
S1	位变量	BOOL	置位使能位													
R	位变量	BOOL	复位使能位													
ENO	位变量	BOOL	能流输出													
???	位变量	BOOL	记录转换状态的位变量													

复位优先触发 (RS)																
??? <<S <<R1 RS >	实现复位优先触发功能，当 R1 为 1 时，无论 S 为 1 或 0，触发输出都为 0；当 R1 为 0 时，若 S 为 0 则输出状态不变，若 S 为 1 则输出为 1。 输入与输出的关系如下： <table><tr><th>S</th><th>R1</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>不改变</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	S	R1	OUT	0	0	不改变	0	1	0	1	0	1	1	1	0
S	R1	OUT														
0	0	不改变														
0	1	0														
1	0	1														
1	1	0														
??? -S ENO > RS -R1																
参数名	变量区	数据类型	说明													
S	位变量	BOOL	置位使能位													
R1	位变量	BOOL	复位使能位													
ENO	位变量	BOOL	能流输出													
???	位变量	BOOL	记录转换状态的位变量													

● 详细说明

触发器指令主要用于实现逻辑控制中的优先触发功能，可根据实际需要选择置位优先或复位优先。一般适用于多个不同优先级的控制信号控制 1 个输出信号的场合。

● 应用示例

如图 6.4.6 所示，利用置位优先触发指令实现 M1.00 和 M2.00 两个位变量对 Q0.08 的触发控制，并利用复位优先触发指令实现 M4.00 和 M5.00 两个位变量对 M1.00 的触发控制，最终通过 M4.00 和 M5.00 两个位变量控制 Q0.08 对应的 LED 灯的亮灭。

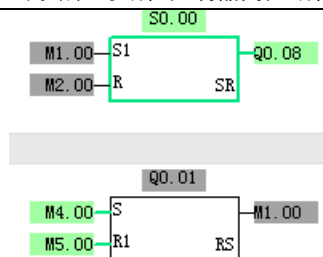


图 6.4.6 RS 触发器指令示例

程序运行前，M1.00、M2.00、M4.00、M5.00、Q0.08 为 0，Q0.08 对应的 LED 灯不亮。

程序运行后，强制 M4.00 为 1，网络 2 中，M1.00 输出为 1，网络 1 中，置位优先，虽然 M2.00=0，但由于 M1.00 为 1，所以 Q0.08 置位，Q0.08 对应的 LED 灯点亮。网络 2 中，强制 M5.00 为 1，RS 复位触发器使 M1.00 复位；网络 1 中，M1.00、M2.00 均为 0，输出保持原来状态，Q0.8 对应的 LED 点亮。

6.5 网络通信指令

网络通信指令按照通信方式可分为以太网和串口两类，主要用于设备与设备之间的对等通信。

在以太网通信方式下，DCCE 可编程控制器利用标准的 EPA 通信协议，支持了 EPA 通信指令，通过该类指令可直接访问本公司其他控制器的变量区。此外，为了与其他公司的 PLC 通信，控制器利用标准的 UDP/TCP 通信协议，支持了 UDP 自由通信指令和 TCP 自由通信指令，实现与其他 PLC 的资源映射。

在串口通信方式下，DCCE 可编程控制器利用标准的 Modbus 通信协议，支持了串行通信指令，实现与串行总线上的从设备或其他设备之间的资源映射。此外，控制器还支持了串行自由通信指令、USS 通信指令和 MPI 通信指令，多种协议的支持，使得控制器能够更灵活的接入不同的工控系统。

6.5.1 EPA 通信

EPA 通信指令利用标准的 EPA 通信协议，该类指令基于 EPA 工业以太网通信规范的读写服务，为每个网络通信区分配了 512 个字节的数据区，所以每条通信报文长度不得超出 512 个字节。

本公司产品中，仅具有以太网功能的控制器支持 EPA 通信指令。

6.5.1.1 位读和确认读指令

● 指令参数

位读指令(NETR_B)			实现位读功能，向指定 IP 设备发送 EPA 位读报文，读取一段指定地址和长度的位变量。
位确认读指令(NETR_B_P)			
参数名	变量区	数据类型	说明
EN	变量	BOOL	使能端，高电平触发
IP	直接写 IP 地址，如 192.168.1.241	DWORD	目的 IP 地址
IN	位变量	BOOL	目标设备的位变量起始地址
NUM	字变量、常数	WORD	连续读取的位变量数量，最大为 4095
Timeout	字变量、常数	WORD	超时时间，单位毫秒
ENO	位变量	BOOL	输出能流
OUT	位变量	BOOL	读取到本地的位变量起始地址
Err	位变量	BOOL	错误标志位

- 详细说明

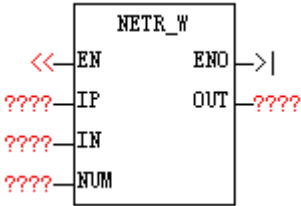
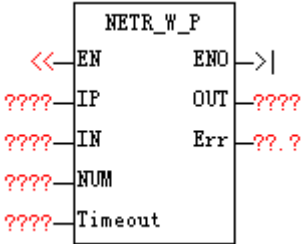
位读指令主要用于对等设备之间的通信，控制器可通过该指令访问网络中的其他控制器。根据指定的 IP 地址，以 IN 为起始地址，从指定设备中读取连续 NUM 个变量位，读取后的结果存储到从 OUT 开始的连续 NUM 个位变量中。

NUM 最大可以是 4095，如果 NUM>4095 指令将不执行。

位确认读指令在位读指令的基础上添加了超时报警功能，如果超时时间内未收到正确回复，则置位错误标志位。

6.5.1.2 字读和确认读指令

- 指令参数

字读指令(NETR_W)		实现字读功能，向指定 IP 设备发送 EPA 字读报文，读取一段指定地址和长度的字变量。	
			
字确认读指令(NETR_W_P)		实现字确认读功能，与字读指令相比，该指令带有超时报警功能，若响应时间超时则置位错误标志。	
			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
IP	直接写 IP 地址，如 192.168.1.241	DWORD	目的 IP 地址
IN	字变量	WORD	目标设备的字变量起始地址
NUM	字变量、常数	WORD	连续读取的字变量数量，最大为 255
Timeout	字变量、常数	WORD	超时时间，单位毫秒
ENO	位变量	BOOL	输出能流
OUT	字变量	WORD	无符号整数
Err	位变量	BOOL	错误标志位

- 详细说明

字读指令主要用于对等设备之间的通信，控制器可通过该指令访问网络中的其他控制器。根据指定的 IP 地址，以 IN 为起始地址，从指定设备中读取连续 NUM 个字变量，读取后的结果存储到从 OUT 开始的连续 NUM 个字变量中。

NUM 最大可以是 255，如果 NUM>255 指令将不执行。

字确认读指令在字读指令的基础上添加了超时报警功能，如果超时时间内未收到正确回复，则置位错误标志位。

6.5.1.3 双字读和确认读指令

● 指令参数

双字读指令(NETR_D)			
NETR_D <<EN ENO-> ????IP OUT-???? ????IN ????NUM			
双字确认读指令(NETR_D_P)			
NETR_D_P <<EN ENO-> ????IP OUT-???? ????IN Err-???.? ????NUM Timeout			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
IP	直接写 IP 地址, 如 192.168.1.241	DWORD	目的 IP 地址
IN	双字变量	DWORD	目标设备的双字变量起始地址
NUM	字变量、常数	WORD	连续读取的双字变量数量，最大为 127
Timeout	字变量、常数	WORD	超时时间，单位毫秒
ENO	位变量	BOOL	输出能流
OUT	双字变量	DWORD	读取到本地的双字变量起始地址
Err	位变量	BOOL	错误标志位

● 详细说明

双字读指令主要用于对等设备之间的通信，控制器可通过该指令访问网络中的其他控制器。根据指定的 IP 地址，以 IN 为起始地址，从指定设备中读取连续 NUM 个双字变量，读取后的结果存储到从 OUT 开始的连续 NUM 个双字变量中。

NUM 最大可以是 127，如果 NUM>127 指令将不执行。

双字确认读指令在双字读指令的基础上添加了超时报警功能，如果超时时间内未收到正确回复，则置位错误标志位。

6.5.1.4 位写和确认写指令

● 指令参数

位写指令(NETW_B)			
		实现位写功能，向指定 IP 设备发送 EPA 位写报文，写入一段指定地址和长度的位变量。	
位确认写指令(NETW_B_P)			
		实现位确认写功能，与位写指令相比，该指令带有超时报警功能，若响应时间超时则置位错误标志。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
IP	直接写 IP 地址，如 192.168.1.241	DWORD	目的 IP 地址
IN	位变量	BOOL	本地的位变量起始地址
NUM	字变量、常数	WORD	连续写入的位变量数量，最大为 4095
Timeout	字变量、常数	WORD	超时时间，单位毫秒
ENO	位变量	BOOL	输出能流
OUT	位变量	BOOL	写入到本地的位变量起始地址
Err	位变量	BOOL	错误标志位

● 详细说明

位写指令主要用于对等设备之间的通信，控制器可通过该指令访问网络中的其他控制器。根据指定的 IP 地址，将本地从 IN 起始的连续 NUM 个位变量写入到目标设备中的从 OUT 起始的连续 NUM 个变量当中。

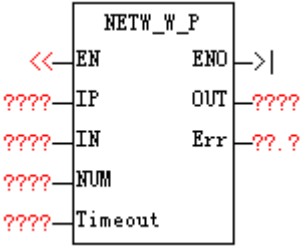
NUM 最大可以是 4095，如果 NUM>4095 指令将不执行。

位确认读指令在位读指令的基础上添加了超时报警功能，如果超时时间内未收到正确回复，则置位错误标志位。

6.5.1.5 字写和确认写指令

● 指令参数

字写指令(NETW_W)			
		实现字写功能，向指定 IP 设备发送 EPA 字写报文，写入一段指定地址和长度的字变量。	
字确认写指令(NETW_W_P)		实现字确认写功能，与字写指令相比，该指令	

		带有超时报警功能，若响应时间超时则置位错误标志。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
IP	直接写 IP 地址，如 192.168.1.241	DWORD	目的 IP 地址
IN	字变量	WORD	本地的字变量起始地址
NUM	字变量、常数	WORD	连续写入的字变量数量，最大为 255
Timeout	字变量、常数	WORD	超时时间，单位毫秒
ENO	位变量	BOOL	输出能流
OUT	字变量	WORD	无符号整数
Err	位变量	BOOL	错误标志位

- 详细说明

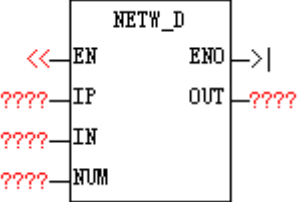
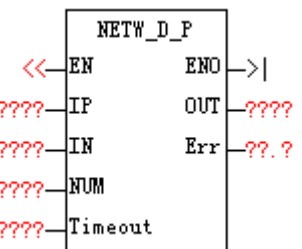
字写指令主要用于对等设备之间的通信，控制器可通过该指令访问网络中的其他控制器。根据指定的 IP 地址，将本地从 IN 起始的连续 NUM 个字变量写入到目标设备中的从 OUT 起始的连续 NUM 个字变量当中。

NUM 最大可以是 255，如果 NUM>255 指令将不执行。

字确认写指令在字写指令的基础上添加了超时报警功能，如果超时时间内未收到正确回复，则置位错误标志位。

6.5.1.6 双字写和确认写指令

- 指令参数

双字写指令(NETW_D)				实现双字写功能，向指定 IP 设备发送 EPA 双字写报文，写入一段指定地址和长度的双字变量。	
双字确认写指令(NETW_D_P)				实现双字确认写功能，与双字写指令相比，该指令带有超时报警功能，若响应时间超时则置位错误标志。	
参数名	变量区	数据类型	说明		
EN	位变量	BOOL	使能端，电平触发		
IP	直接写 IP 地址，如 192.168.1.241	DWORD	目的 IP 地址		
IN	双字变量	DWORD	本地的双字变量起始地址		
NUM	字变量、常数	WORD	连续写入的双字变量数量，最大为 127		
Timeout	字变量、常数	WORD	超时时间，单位毫秒		
ENO	位变量	BOOL	输出能流		
OUT	双字变量	DWORD	写入到本地的双字变量起始地址		
Err	位变量	BOOL	错误标志位		

● 详细说明

双字写指令主要用于对等设备之间的通信，控制器可通过该指令访问网络中的其他控制器。根据指定的 IP 地址，将本地从 IN 起始的连续 NUM 个双字变量写入到目标设备中的从 OUT 起始的连续 NUM 个双字变量当中。

NUM 最大可以是 127，如果 NUM>127 指令将不执行。

双字确认写指令在双字写指令的基础上添加了超时报警功能，如果超时时间内未收到正确回复，则置位错误标志位。

➤ EPA 读指令使用说明

使用 EPA 读指令和其他控制器通信时，指令使能后会根据设定好的网络读取时间间隔，周期性读取指定设备的数据信息。通过 SM 区可配置该类指令的最小通信时间间隔和超时时间间隔，如表 6.5.1 所示，SMW182 作为 EPA 读指令的最小通信时间间隔，即每两次网络读取数据命令之间的最小时间间隔，单位为 10ms，可以通过调整这个参数来控制读取数据的频率。SMW183 为 EPA 读指令的最大超时时间间隔，单位为 10ms。SMW182 和 SMW183 设置后均可以永久保存，切勿频繁更改。

➤ EPA 写指令使用说明

EPA 写指令支持 3 种写入模式，通过 SMW166 配置，0 为直接写入模式，在使用期间会一直不停地向目标设备写入指定的数据，建议此模式下用上升沿来进行控制；1 为自动判断写入模式，第一次写入前会读取目标设备中的数据，并与要写入的数据进行比较，不同时才写入，否则不写入，使能期间会检查要写入的数据是否改变，改变后才写入，并且写入后会进行读取检查，自动判断是否写入正确，若不正确则再次写入正确数据，直到写入正确为止；2 为兼容 PEC7000 模式，写入数据前比较，不同时才写入，此模式下不会进行自动检查。

写指令有两个控制参数，如表 6.5.1 所示，SMW166 控制写入模式，SMW184 控制写入的最小时间间隔，单位为 10ms，修改完毕后永久保存。

模式 1 只能在非 PEC7000 的设备之间进行通信，不能用于 PEC7000 等非兼容性设备。

表 6.5.1 以太网通信指令 SM 区控制参数

SMW166	EPA 写指令模式选择，0 为直接写入模式，1 为写入前和写入后读取验证模式，2 为兼容 PEC7000 模式，其余为保留，默认为模式 1
SMW182	EPA 读指令最小通信时间间隔，单位为 10ms，默认为 200ms
SMW183	EPA 读指令最大超时时间间隔，单位为 10ms，默认为 400ms
SMW184	EPA 写指令最小通信时间间隔，单位为 10ms，默认为 1 秒

提示：EPA 通信指令为电平触发，即当 EN 端为 1 时，只要原设备映射变流区中的数据发生变化，目标设备的对应的映射区就会同步更新。

注意：

①使用 EPA 读写指令时，原设备和目标设备的 IP 地址网段和通信端口号必须相同，否则指令将不能正确执行。

②一个工程中最多只能同时使用 64 个 EPA 通信指令，当个数大于 64 时，64 个之后的指令将不执行，并且会把 SM62.0 置 1。

③具体的 EPA 读写指令错误标示详见附录 C 中的功能参数。

● 应用示例

(1) EPA 读写指令例程：

如图 6.5.1 所示，控制器以 300ms 为通信时间间隔，通过位读指令对 IP 地址为 192.168.1.59 的设备进行读操作。从指定设备的 M0.00 为起始的变量区中读取连续 16 个位变量，读取后的结果存储到从 M100.0 开始的连续 16 个位变量中。其中设置 SMW182 为 30，即每 300ms 读一次数据，设置 SMW183 为 200，即通信超时时间是 2000ms。通过字写指令对 IP 地址为 192.168.8.193 的设备进行写操作。将本设备中从 VW0 开始的连续 10 个字变量写入到目标设备的 VW100 到 VW109 共 10 个字变量中。其中 SMW184 为 30，即写最小通信时间间隔为 300ms。

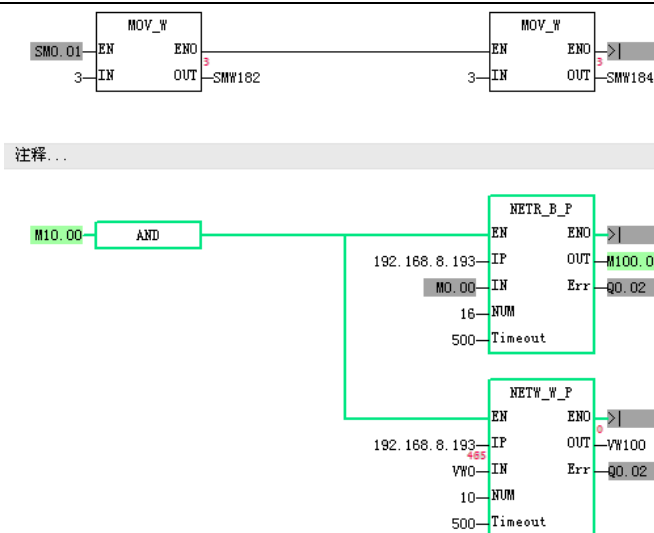


图 6.5.1 UDP 写指令例程

(2) EPA 确认读写指令例程:

每 0.3 秒将本设备的 M10.00 的数据写入 IP 为 192.168.8.192 设备的 V0.0 开始的 16 个位地址中, 同时将 IP 为 192.168.8.192 的从 V0.00 开始的 16 位数据读取到本设备的 Q0.0 开始的 16 个位地址中。

源程序如下: 设置 SMW184 为 3, 每 30ms 写入一次数据, 超时时间为 50ms;

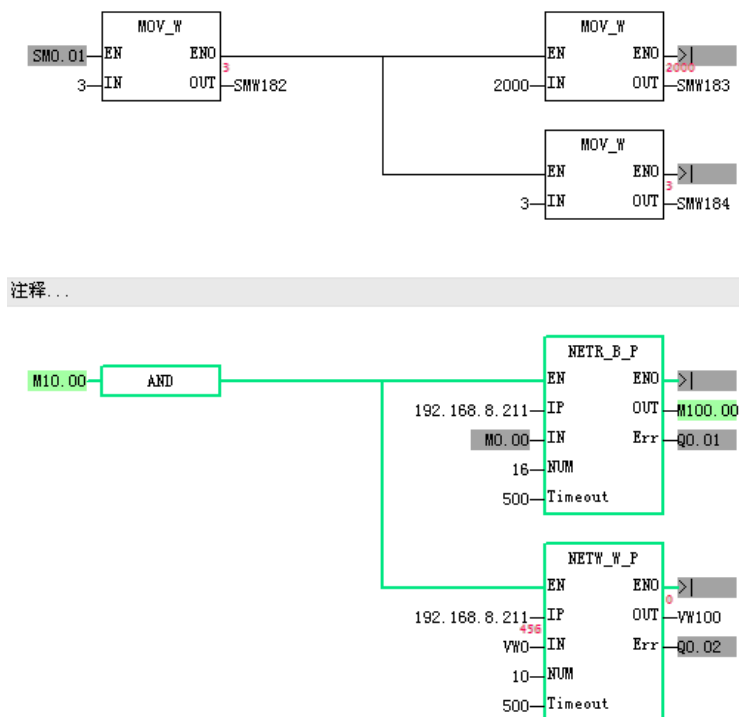


图 6.5.2 UDP 读写指令成功例程

在 UDP 读写指令的通信过程中, 如果在设置的超时时间内没有通信成功, 错误标志引脚 Err 会输出高电平, 表示通信失败, 如图 6.5.3 所示。

注意: 当错误标志引脚 Err 置 1 后, 如果重新使能 M2.00, Err 引脚不会被复位, 只有重新下载指令时才能被复位。

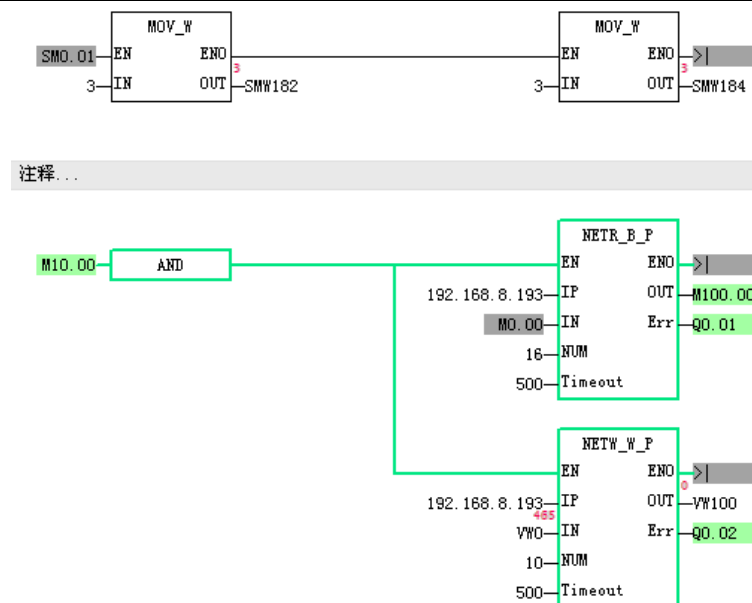


图 6.5.3 UDP 读写指令失败例程

6.5.2 UDP 自由通信

6.5.2.1 UDP 自由发送指令(NET_XMT)

● 指令参数

UDP 自由发送指令(NET_XMT)			
		实现 UDP 自由发送功能，向指定 IP 设备发送指定字节数的自由格式报文	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
IP	常量直接写 IP 地址	DWORD	目的 IP 地址，目标设备可以是控制器或者电脑，例如“192.168.1.105”
PORT	字变量	WORD	目标设备以太网端口，范围 0~65535
IN	字变量	WORD	发送数据缓冲区，第 1 个字变量存储发送长度
ENO	位变量	BOOL	输出能流

● 详细说明

NET_XMT 指令通过输入引脚配置目的 IP 和目的端口号，配置一块连续的变量区作为报文缓冲区。指令的发送字节数由 IN 起始的第 1 个字变量（设第一个字变量为 N）决定，最大支持 512 字节数据，以第 2 个字变量为起始，连续取 N 个字变量，每个字变量中低字节为发送的报文数据。

注意：

- ①在同一工程中最多可同时使用 16 个 NET_XMT 指令。
- ②NET_XMT 指令不使用 EPA 协议通信，而是直接发送用户配置的报文，报文格式由用户自定。
- ③原控制器和目的控制器的 IP 地址需要在同一网段内。
- ④通过 NET_XMT 指令与其他控制器通信时，若采用标准协议报文读写目标设备变量区通信端口号应设置为 11000，若采用自定义报文，端口号不限，但目标设备必须调用一个 NXMT_RCV 指令接收报文。

● 应用示例

如图 6.5.4 所示，通过 NET_XMT 指令对 IP 地址为 192.168.8.194 的设备进行 MW0 的写入操作，写入数据为 0x0B0B。采用上升沿触发方式使能指令，以 MW100 为发送数据缓存区的起始变量区，发送报文字节

长度为 18，发送内容如图 6.5.5 所示。

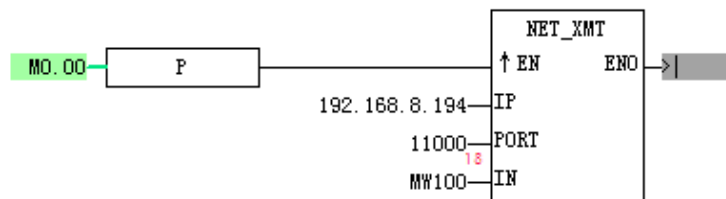


图 6.5.4 UDP 自由通信指令例程

绝对地址	Modbus变量地址	数据类型	当前值(10进制)	当前值(16进制)	新数值
MW100	0x0B84 (2948)	有符号双字节	18	12	18
MW101	0x0B85 (2949)	有符号双字节	13	D	0x0d
MW102	0x0B86 (2950)	有符号双字节	0	0	0
MW103	0x0B87 (2951)	有符号双字节	0	0	00
MW104	0x0B88 (2952)	有符号双字节	0	0	00
MW105	0x0B89 (2953)	有符号双字节	18	12	18
MW106	0x0B8A (2954)	有符号双字节	0	0	00
MW107	0x0B8B (2955)	有符号双字节	1	1	01
MW108	0x0B8C (2956)	有符号双字节	0	0	00
MW109	0x0B8D (2957)	有符号双字节	0	0	00
MW110	0x0B8E (2958)	有符号双字节	0	0	00
MW111	0x0B8F (2959)	有符号双字节	105	69	0x69
MW112	0x0B90 (2960)	有符号双字节	0	0	00
MW113	0x0B91 (2961)	有符号双字节	1	1	01
MW114	0x0B92 (2962)	有符号双字节	0	0	00
MW115	0x0B93 (2963)	有符号双字节	0	0	00
MW116	0x0B94 (2964)	有符号双字节	0	0	00
MW117	0x0B95 (2965)	有符号双字节	11	B	11
MW118	0x0B96 (2966)	有符号双字节	11	B	11

图 6.5.5 UDP 自由通信报文内容

6.5.2.2 UDP 自由确认发送指令(NXMT_SEND)

● 指令参数

UDP 自由确认发送指令(NET_SEND)			
		实现 UDP 自由确认发送功能，向指定 IP 设备发送指定字节数的自由格式报文，并能够接收处理响应报文。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，上升沿触发
IP	常量直接写 IP 地址	DWORD	目的 IP 地址，目标设备可以是控制器或者电脑，例如“192.168.1.105”。
PORT	字变量	WORD	目标设备以太网端口，范围 0~65535
IN	字变量	WORD	发送数据缓冲区，第 1 个字变量存储发送长度
ENO	位变量	BOOL	输出能流
ROS	位变量	BOOL	成功接收到响应报文标志位
OUT	字变量	WORD	响应报文缓冲区

● 详细说明

NXMT_SEND 指令同 NET_XMT 指令功能相似，同样可直接发送用户配置的报文，报文格式由用户自定。此外，该指令在实现将本地变量的数据发送到指定 IP 和端口的设备后，还对指定设备的响应报文进行了处理，若接收到响应报文，ROS 置位，并把收到的数据存储在 OUT 缓冲区中。该指令发送接收使用同一端口号 13000，故使用该指令时，该端口不能被其他含以太网接收功能的指令占用。

注意：NXMT_SEND 指令在同一工程中最多可同时使用 16 个。

- 应用示例

如图 6.5.6 所示, 通过 NXMT_SEND 指令对 IP 地址为 192.168.8.194 的设备进行 MW0 的写入操作, 写入数据为 0x0B0B。采用上升沿触发方式使能指令, 以 MW100 为发送数据缓存区的起始变量区, 发送报文字节长度为 18, 发送内容如图 6.5.5 所示。

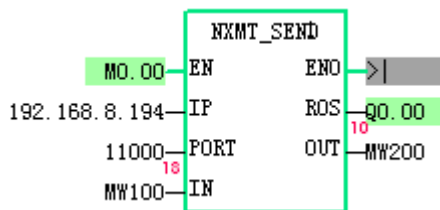


图 6.5.6 UDP 自由确认写指令例程

6.5.2.3 UDP 自由确认接收指令(NXMT_RCV)

- 指令参数

UDP 自由确认接收指令(NXMT_RCV)			
		实现 UDP 自由确认接收功能, 接收所有目的端口号为本地端口的报文, 并获取远端设备的 IP 和端口号。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端, 电平触发
L_PORT	字变量、常数	WORD	本地接收数据的端口号
ENO	位变量	BOOL	输出能流
ROS	位变量	BOOL	接收成功标志位
OUT	字变量	WORD	接收到的报文缓冲区
R_IP	IP 地址	DWORD	远端设备的 IP, 远端设备可以是控制器或者电脑
R_PORT	字变量	WORD	远端设备的端口号

- 详细说明

NET_RCV 指令可以单独使用, 用于接收从本机 L_PORT 端口收到的数据, 收到数据后 ROS 置位, 并把数据存放在 OUT 开始的缓冲区中, OUT 的首字节为接受数据的长度。R_IP 为发来消息的远端设备 IP, R_PORT 为远端设备的端口。

IP 地址存储为点分十进制格式。如 192.168.8.170 的点分十进制表示为 AA08A8C0。

注意: 以太网确认接收指令, 在同一工程中只能调用 1 个, 否则指令将不会执行。

- 应用示例

如图 6.5.7 所示, 功能块将监听 15000 端口所接收到的数据, 数据存放在 MW200 中。收到数据长度为 6 个字节, 数据存储在 MW201-MW206 中。MD300 存储的是远端设备 IP, MW302 为远端设备端口号。图中显示的是 2852694208 为有符号四字节, 将其改为点分十进制格式为 AA08A8C0, 其对应的 IP 为 192.168.8.170。

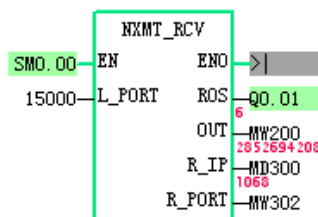


图 6.5.7 UDP 自由确认写指令例程

6.5.2.4 UDP 自由发送接收指令 (UDP_AFC)

● 指令参数

UDP 自由通讯(UDP_AFC)			
实现 UDP 发送、接收任意格式的数据信息			
操作数	变量区	数据类型	说明
EN	位变量	BOOL	使能位，高电平触发
TR	位变量	BOOL	发送请求标志位，上升沿触发，检测到上升沿发送一次数据。发送成功时发送完成标志置位
IP	IP 地址, 如 192.168.1.241	DWORD	目的 IP 地址
DP	字变量, 常数	WORD	目的端口号
LP	字变量, 常数	WORD	本地端口号
SBUF	字变量	WORD	发送缓存区起始地址, 第一个字为发送的数据长度
RBUF	字变量	WORD	接收缓存区起始地址, 第一个字为接收的数据长度
ENO	位变量	BOOL	能流输出
EOS	位变量	BOOL	UDP 发送完成标志, 为 1 时数据发送成功
EOR	位变量	BOOL	UDP 接收完成标志, 为 1 时表示当前读缓存区收到数据信息, 用户取走数据后需要手动清除完成标志, 否则会出现数据冲突错误
ERR	字变量	WORD	错误状态字 0x0001 SOCKET 建立失败 0x0002 UDP 端口打开失败 0x0004 指定的 IP 为设备自身的 IP 0x0008 指定的发送缓存区地址类型错误 0x0010 指定的发送缓存区地址类型错误 0x0020 UDP 数据发送失败 0x0040 接收到的数据被覆盖 0x0080 接收到的数据长度越界 0x0100 发送数据长度越界
RIP	字变量	WORD	收到数据的 IP 地址
RPORT	字变量	WORD	收到数据的端口号

● 详细说明

UDP 自由通信指令通过用户指定的本地端口，完成任意格式数据的收发操作。当用户收到报文后会在 RIP 和 RPORT 引脚上给出数据的 IP 地址和端口信息。其中接收和发送缓存区的第一个字为接收数据和发送数据的长度。目前支持的传输数据最大长度为 512 个字节。

EOR 引脚为接收完成引脚，用户在数据后应当将 EOR 引脚手动清 0，表示对应的数据信息已经被使用。若下一条数据到来时，EOR 引脚仍为 1 就会在 ERR 引脚给出数据覆盖提示信息。

提示：706、707、11000、12000、13000 端口为设备保留端口，请不要使用。

● 应用实例

如图 6.5.8 所示通过 UDP_AFC 功能块与 IP 地址为 192.168.8.168 端口为 13200 的电脑进行 UDP 通信，本地接收发送端口为 12300 端口。

将要发送的数据存到 SBUF 所指定的变量区中，第一个字为发送的长度信息，将 TR 引脚置 1 向电脑端发送一次数据。发送请求是沿触发信号每个上升沿发送一次数据。下图向电脑端发送 VW200 开始 1 个字节

的数据信息，发送时低字节在前高字节在后。发送完成后 EOS 引脚置位。若发送失败 ERR 引脚会给出错误信息。

RBUF 引脚为接收数据缓存区地址，第一个字为接收数据的字节数，图中显示设备从电脑端收到长度为 5 个字节的数据信息存放在 VW301 开始的地方，先收到的数据存放在低字节中，接收完成后 EOR 引脚会置位。当接收出现错误时，ERR 引脚会提示响应的错误信息。同时在 RIP 和 RPORT 引脚会显示数据发送的 IP 和端口号。

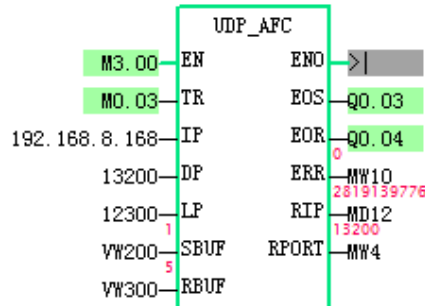


图 6.5.8 UDP 自由通信

6.5.3 TCP 自由通信

6.5.3.1 TCP 连接指令(TCP_LINK)

● 指令参数

TCP 连接指令(TCP_LINK)			
		实现 TCP 连接功能，与指定 IP 的设备建立 TCP 连接，设备可以是控制器或者电脑。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能引脚，高电平触发
IP	直接写 IP 地址，如 192.168.1.241	DWORD	目的 IP 地址，可以是电脑或控制器
PORT	字变量、常数	WORD	目的端口
ID	字变量、常数	WORD	链接标识，为 0 或 1
ENO	位变量	BOOL	输出能流
LINK	位变量	BOOL	链接成功标志

● 详细说明

TCP 连接指令会绑定指定的 IP 和端口号与目标设备进行 3 次握手通信，建立 TCP 连接。当需要与目标设备建立 TCP 连接时，只需要在本控制器中调用 TCP_LINK 指令，配置好参数即可进行正常通信，目标设备无需主动建立 TCP 连接。

注意：

- ①TCP 连接指令 TCP_LINK，在同一工程中最多可调用 2 个，ID 只能为 0 或 1，ID 大于 1 时，指令将不执行。
- ②当使用两个 TCP_LINK 指令时，目的的 IP 地址不能相同。
- ③同一个工程中的 TCP_LINK 指令 ID 不能相同。
- ④两个 TCP_LINK 指令不能指向同一个目标设备。
- ⑤当目标设备是本公司内部产品时，PORT 引脚只能为 502~512（包括 502 和 512），非本公司产品无限制。

● 应用示例

如图 6.5.9 所示，通过 TCP_LINK 指令与 IP 地址为 192.168.8.152 的电脑建立 TCP 连接，绑定的端口号为 14000，ID 为 0，LINK 置位说明该连接已经成功建立。

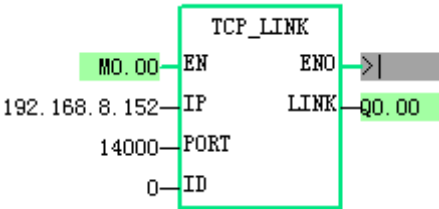


图 6.5.9 TCP 连接指令例程

6.5.3.2 TCP 发送指令(TCP_SEND)

● 指令参数

TCP 自由发送指令(TCP_SEND)			
		实现 TCP 自由发送功能，通过已经建立好的 TCP 连接，向指定 IP 设备发送指定字节数的自由格式报文。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
S_BUF	字变量	WORD	发送缓冲区
NUM	字变量、常数	WORD	发送缓冲区长度，单位字节，最多 512 个字节
ID	字变量、常数	WORD	标识使用哪个 TCP 连接
ENO	位变量	BOOL	输出能流
ROS	位变量	BOOL	发送成功标志

● 详细说明

TCP 发送指令使用前需要先进行 TCP 连接，通过 ID 选择指定的 TCP 连接端口后进行自由发送，发送数据的起始变量地址为 S_BUF，长度为 NUM。如果发送成功 ROS 置位。

注意：

- ①调用 TCP 发送指令前必须先调用 TCP 连接指令。
- ②TCP_SEND 指令在同一工程中最多可同时使用 32 个。

● 应用示例

如图 6.5.10 所示，使用 TCP_LINK 指令连接到 IP 地址为 192.168.8.152 的电脑。通过 TCP_SEND 指令发送 MW100 开始的 10 个字节的数据，使能指令后，ROS 被置位，说明发送已经成功。

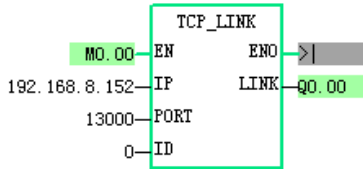


图 2 注释...

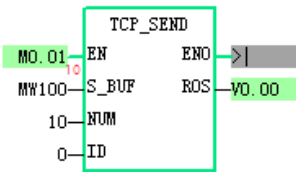


图 6.5.10 TCP 发送指令例程

在电脑上打开 sockit 软件，选择服务器选项，在网络设置中将 TCP 地址配置成电脑的 IP 地址，端口号为 TCP_LINK 指令中的端口号，点击 TCP 侦听；然后使能 TCP_LINK 指令，如果输出端 LINK 置位，说明连接成功。此时，在 sockit 软件中的当前连接窗口就会发现与电脑连接的设备了。在 sockit 软件的收发记录中，记录了从控制器发过来的数据。如图 6.5.11 所示。

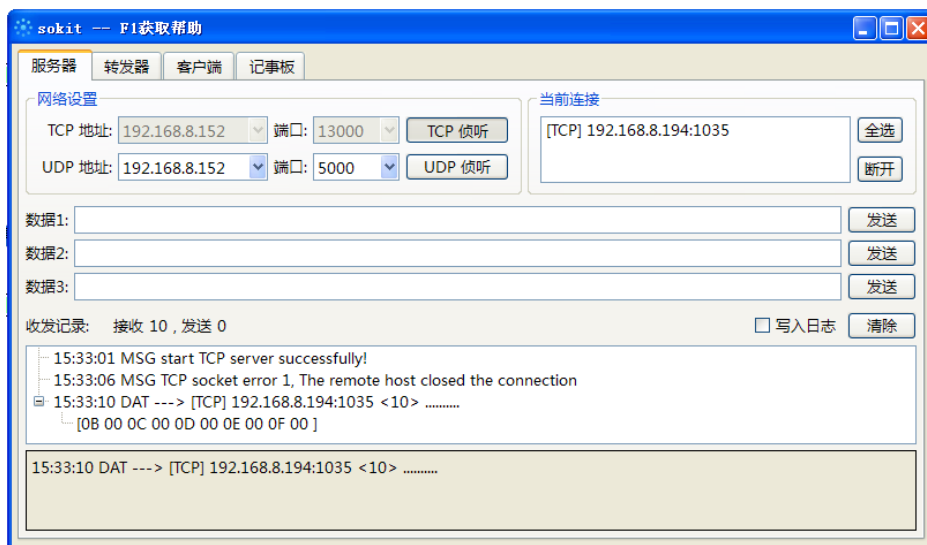


图 6.5.11 sockit 软件接收数据显示示例

6.5.3.3 TCP 接收指令(TCP_RCV)

● 指令参数

TCP 自由接收指令(TCP_RCV)			
		实现 TCP 自由接收功能，通过已经建立好的 TCP 连接，接收该连接上的自由格式报文。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
ID	字变量、常数	WORD	标识该使用哪个 TCP 连接
ENO	位变量	BOOL	输出能流
ROS	位变量	BOOL	接收成功标志
R_BUF	字变量	WORD	接收缓冲区
NUM	字变量、常数	WORD	缓冲区长度，单位为字节

● 详细说明

TCP 接收指令用于使用指定 ID 的 TCP 连接进行接收数据，将收到的数据存储在 R_BUF 中，数据长度为 NUM。如果接收成功 ROS 置位。

注意：

- ①调用 TCP 接收指令前必须先调用 TCP 连接指令。
- ②同一个 ID 号只能调用一个 TCP_RCV 指令。
- ③一个 TCP_RCV 指令最多只能接收 255 个字节，超过的部分将被舍弃。

● 应用示例

如图 6.5.12 所示，使用 TCP_LINK 指令连接到 IP 地址为 192.168.8.152 的电脑。TCP_RCV 指令接收从电脑发过来的数据，并将接收到的数据存储在 MW10 中，长度存储在 MW20 中。ROS 为 1，说明接收成功。

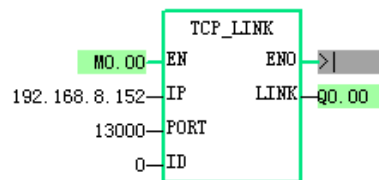


图 2 注释...

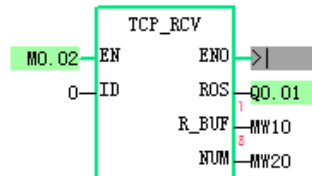


图 6.5.12 TCP 接收指令例程

在电脑上打开 sockit 软件，选择服务器选项，在网络设置中将 TCP 地址配置成电脑的 IP 地址，端口号为 TCP_LINK 指令中的端口号，点击 TCP 侦听；然后使能 TCP_LINK 指令，如果输出端 LINK 置位，说明连接成功。此时，在 sockit 软件中的当前连接窗口就会发现与电脑连接的设备了。利用 sockit 软件发送 8 个字节的数据，在 sockit 软件的收发记录中，记录了软件发送的数据。如图 6.5.13 所示。

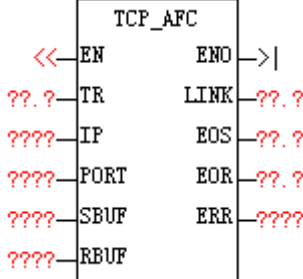


图 6.5.13 sockit 软件发送数据显示示例

注意：控制器接收 sockit 发送的数据时，按照小端模式存储。例如，在此示例中，发送的前四个数据 0100，发送到控制器中后，会将该数据按照高位在前，低位在后的原则，变成 0001，存储在 MW10 中。

6.5.3.4 TCP 发送接收 (TCP_AFC)

● 指令参数

TCP 自由通讯(TCP_AFC)			
		实现 TCP 发送、接收任意格式的数据信息	
操作数	变量区	数据类型	说明
EN	位变量	BOOL	使能位，高电平触发
TR	位变量	BOOL	发送请求标志位，上升沿触发，LINK 表示为 1 时检测到上上升沿发送一次数据。发送成功时发送完成标志置位，否则置位错误标志，此时用户需重新发送当前数据
IP	目的 IP 地址，如 192.168.1.241	DWORD	目的 IP 地址
PORT	字变量，常数	WORD	目的端口号
SBUF	字发送缓冲区	WORD	发送缓存区起始地址，第一个字为发送的数据长度
RBUF	字变量	WORD	接收缓存区起始地址，第一个字为接收的数据长度
ENO	位变量	BOOL	能流输出
LINK	位变量	BOOL	TCP 连接建立标志，为 1 时已建立连接
EOS	位变量	BOOL	TCP 发送完成标志，为 1 时数据发送成功
EOR	位变量	BOOL	TCP 接收完成标志，为 1 时表示当前读缓存区收到数据信息，用户在取完数据后需手动清除这个标志位，否则会有数据冲突错误
ERR	字变量	WORD	TCP 发送错误状态字 0x0001 输入的 IP 地址为本机 IP 0x0002 发送数据长度超过最大长度 0X0004 指定的发送缓存区地址 0x0008 接收报文长度超过最大长度 0x0010 接收缓存区地址错误 0x0020 TCP 接收到的数据被覆盖 0x0040 未建立 TCP 连接 0x0080 TCP 连接个数超过限制 0x0100 缓存区长度不足 0x0200 数据响应超时 0x0400 TCP 连接已存在 0x0800 socket 不支持此操作

● 详细说明

TCP 自由通信指令会绑定指定的 IP 和端口号与目标设备进行 3 次握手通信，建立 TCP 连接。

该指令可实现 TCP 连接建立以及 TCP 数据接收发送功能。能够主动检测对方设备是否在线如果对方设备掉线会主动进行掉线重连操作。其中接收和发送缓存区的第一个字为接收数据和发送数据的长度。目前支持的传输数据最大长度为 512 个字节。

EOR 引脚为接收完成引脚，用户在数据后应当将 EOR 引脚手动清 0，表示对应的数据信息已经被使用。若下一条数据到来时，EOR 引脚仍为 1 就会在 ERR 引脚给出数据覆盖提示信息。

提示：

①为了实现对方设备离线检测功能，当用户不进行 TCP 收发操作时，设备会发送一条数据为[00]的报文，根据目标设备回复的响应报文来判断目标设备是否在线。

②指令最多可调用 8 个，更换本地端口后需要重新使能功能块使能引脚，新端口才会生效。

- 应用示例

如图所示通过 TCP_AFC 功能块与 IP 地址为 192.168.8.168 端口为 530 的电脑进行 TCP 通信，功能块使能后设备与电脑建立 TCP 连接。

将要发送的数据存到 SBUF 所指定的变量区中，第一个字为发送的长度信息，将 TR 引脚置 1 向电脑端发送一次数据。发送请求是沿触发信号每个上升沿发送一次数据。下图向电脑端发送 VW1 开始 4 个字节的数据信息，发送时低字节在前高字节在后。发送完成后 EOS 引脚置位。若发送失败 ERR 引脚会给出错误信息。

RBUF 引脚为接收数据缓存区地址，第一个字为接收数据的字节数，图中显示设备从电脑端收到长度为 5 个字节的数据信息存放在 VW101 起始的变量区中，先收到的数据存放在低字节中，接收完成后 EOR 引脚会置位。当接收出现错误时，ERR 引脚会提示响应的错误信息。

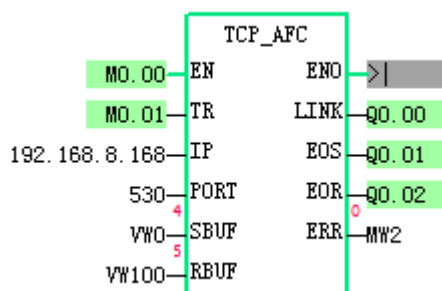


图 6.5.14 TCP 自由通信示例

6.5.4 串行通信

串行通信指令能够把内部指定的一段变量区与扩展设备指定的一段变量区关联起来，即该指令能够把扩展设备的变量资源映射到内部数据区中。

扩展设备可以是当前设备的从设备，也可以是与当前设备位于同一条通信总线上支持 Modbus 协议的设备。指令通过两种模式完成对上述两种设备资源的映射：从设备映射模式下，指令可以将在线从设备中的资源映射到主设备中；Modbus 总线设备映射模式下，指令能够将与当前设备位于同一总线上且支持 Modbus 协议的指定控制器的地址资源映射到当前设备中。

指令的两种工作模式通过 CYCLE 引脚的高 12 位进行区别，其具体意义在下文描述。

- 参数说明

6.5.4.1 串行位读指令 (SCR_B)

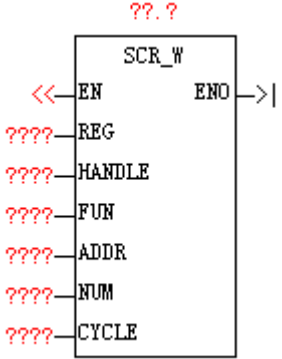
- 指令参数

串行位读指令(SCR_B)			
从扩展设备中读取一段位变量。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
REG	位变量	BOOL	目的位变量起始地址
HANDLE	字变量、常数	WORD	指令运行行为从设备模式时，表示从设备编号；指令运行在总线设备模式时，表示总线设备地址

FUN	字常数	WORD	Modbus 功能号，只可能是 1 或 2
ADDR	字变量、常数	WORD	扩展设备源位变量区的起始地址
NUM	字常数	WORD	位变量数量，最大为 1920
CYCLE	字变量、常数	WORD	运行模式和执行频率设定字。当 CYCLE 的高 12 位不全为 1 时，设定为从设备模式，此时由 CYCLE 字设定指令执行频率，该值是程序扫描次数；当 CYCLE 的高 12 位为 1 时，设定为总线设备模式，此时 CYCLE 的低 4 位设定为总线设备连接到主设备串口号，0 为串口 0，1 为串口 1
???	位变量	BOOL	头参数“???”是通信成功标志位，该值为 1，表示通信成功，为 0 表示通信失败
ENO	位变量	BOOL	输出能流

6.5.4.2 串行字读指令 (SCR_W)

● 指令参数

串行字读指令(SCR_W)			
		从扩展设备中读取一段字变量。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
REG	字变量	WORD	目的字变量起始地址
HANDLE	字变量、常数	WORD	指令运行行为从设备模式时，表示从设备编号；指令运行在总线设备模式时，表示总线设备地址
FUN	常数	WORD	Modbus 功能号，只能是 3 或 4
ADDR	字变量、常数	WORD	扩展设备源字变量区的起始地址
NUM	常数	WORD	字变量数量，最大为 120
CYCLE	字变量、常数	WORD	无符号整数
???	位变量	BOOL	头参数“???”是通信成功标志位，该值为 1，表示通信成功，为 0 表示通信失败
ENO	位变量	BOOL	输出能流

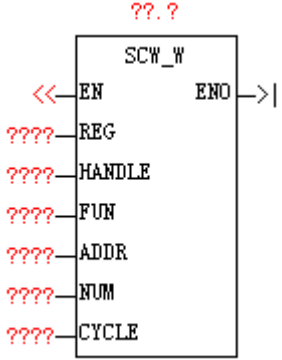
6.5.4.3 串行位写指令 (SCW_B)

● 指令参数

串行位读指令(SCW_B)			
		向扩展设备中写入一段位变量。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
REG	位变量	BOOL	源位变量起始地址
HANDLE	字变量	WORD	指令运行行为从设备模式时，表示从设备编号；指令运行在总线设备模式时，表示总线设备地址
FUN	字常数	WORD	Modbus 功能号，只能是 5 或 15
ADDR	字变量、常数	WORD	扩展设备目的位变量区的起始地址
NUM	常数	WORD	位变量数量。当 FUN 为 5 时，该指令只能操作 1 个位变量，FUN 为 15 时最多可操作 1920 个位变量
CYCLE	字变量、常数	WORD	运行模式和执行频率设定字。当 CYCLE 的高 12 位不全为 1 时，设定为从设备模式，此时由 CYCLE 字设定指令执行频率，该值是程序扫描次数；当 CYCLE 的高 12 位为 1 时，设定为总线设备模式，此时 CYCLE 的低 4 位设定总线设备连接到主设备串口号，0 为串口 0，1 为串口 1。
??.	位变量	BOOL	头参数“??.”是通信成功标志位，该值为 1，表示通信成功，为 0 表示通信失败。
ENO	位变量	BOOL	输出能流

6.5.4.4 串行字写指令 (SCW_W)

● 指令参数

串行字写指令(SCW_W)			
			
向扩展设备中写入一段字变量			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能端，电平触发
REG	字变量	WORD	源位变量起始地址
HANDLE	字变量、常数	WORD	指令运行行为从设备模式时，表示从设备编号；指令运行在总线设备模式时，表示总线设备地址
FUN	常数	WORD	Modbus 功能号，只能是 6 或 16
ADDR	字变量、常数	WORD	扩展设备目的字变量区的起始地址
NUM	常数	WORD	字变量数量。当 FUN 为 6 时，该指令只能操作 1 个字变量，FUN 为 16 时最多可操作 120 个字变量
CYCLE	字变量、常数	WORD	运行模式和执行频率设定字。当 CYCLE 的高 12 位不全为 1 时，设定为从设备模式，此时由 CYCLE 字设定指令执行频率，该值是程序扫描次数；当 CYCLE 的高 12 位为 1 时，设定为总线设备模式，此时 CYCLE 的低 4 位设定总线设备连接到主设备串口号，0 为串口 0，1 为串口 1。 头参数“??.”是通信成功标志位，该值为 1，表示通信成功，为 0 表示通信失败
??.	位变量	BOOL	头参数“??.”是通信成功标志位，该值为 1，表示通信成功，为 0 表示通信失败。
ENO	位变量	BOOL	输出能流

● 详细说明

串行通信指令是对控制器管理从设备的一种扩展，该类指令能够根据配置对控制器中已经发现的从设备进行变量区映射，映射的方式由 Modbus 协议的 1, 2, 3, 4, 5, 6, 15, 16 号功能来实现。控制器最多支持 256 个串行通信指令，但由于下位机串口通信队列有限，256 个串行通信指令不能同时使能，最多可同时使能 48 个，若需要使用更多的通信指令可采用分时使能的方式实现。

指令进行扩展设备资源映射时，可以访问从设备资源也可访问同一 Modbus 总线上的设备。指令执行时，通过 CYCLE 区分这两种模式。

➤ 从设备映射模式：

当 CYCLE 高 12 位不全为 1 时，设备处于从设备资源映射模式，此时 CYCLE 表示资源刷新的周期。

当扩展设备为当前设备的从设备时，以前写指令存在第一次运行无法向从设备数据区写 0 的问题，故改进的串行通信写指令在每次执行串行通信写之前，将要写的数据区映射上来，以确定是否要写。在每次写之后，对写下去的数据进行读取确认，以确保正确写入。读的功能号会根据位操作或字操作进行自适应，位变量操作自动适应为 1 号或 2 号 Modbus 读命令，字变量操作自动适应为 3 号或 4 号 Modbus 读命令。

为保持和以前功能的兼容性，默认 SMW179 为 0，使用以前的写入方式，要使用改进的串行通信写指令则使 SMW179 为 1 即可，如表 6.5.2 所示。需要注意的是，当使用串行通信相关写指令的时候，需要确保用户程序至少执行两轮以上。即 CYCLE 的值至少需要大于 1。如果仅执行一轮，将无法保证数据正确写入相应变量。

表 6.5.2 串行通信指令 SM 区控制参数

SMW179	0 表示串行通信写指令写之前和写之后不读取所写变量； 1 表示串行通信写指令写之前和写之后均读取所写变量； 默认为 0。

➤ Modbus 总线设备映射模式：

当 CYCLE 高 12 位全为 1 时，设备运行在 Modbus 总线设备映射模式，CYCLE 的低 4 位为 0 表示扩展设备与串口 0 相连，低 4 位为 1 表示扩展设备与串口 1 相连。

指令运行在当前模式下时，扩展设备可以是当前设备的从设备，也可以不是当前设备的从设备。

当串行通信指令使用该模式进行通信时，每个 EN 的上升沿会触发一次当前设备与扩展设备的资源映射。上升沿过后，EN 应保持有效直到扩展设备响应或者当前命令超时。如果资源映射成功，那么功能块的头顶参数会被置 1；如果资源映射失败或者通信超时，那么头顶参数被置 0。

指令的超时时间使用串口的超时时间（通过 PLC_Config 软件读取或者设置该时间），在使用时需要保证 EN 引脚新的上升沿发生在指令超时后或在本次请求成功后，以确保头顶参数正确反应当前指令的状态。

● 应用示例

以串行字写指令为例说明串行通信网络块的法。

(1) 示例 1

如图 6.5.15 所示，程序功能为将本设备 MW100 中的字数据，写入到扩展从设备 Modbus 变量地址为 2336 的变量中，写入 SMW179 为 1，即写入前后均进行读检查，每运行 1000 次本指令检查一次是否需要写入，扩展从设备的设备编号为 0。

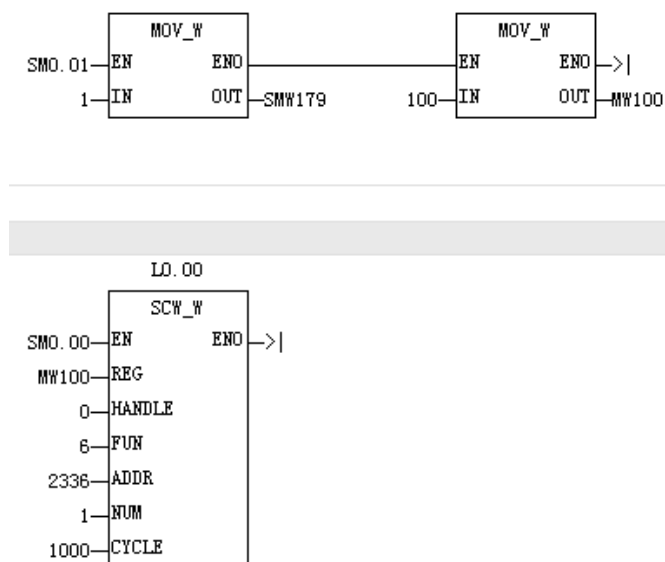


图 6.5.15 串行通信例程

示例 1 中使用了从设备资源映射模式（CYCLE 引脚高 12 位不全为 1），当前模式下 HANDLE 表示从设备编号，此处为 0 即 0 号从设备。

(2) 示例 2

如图 6.5.16 所示，程序执行与示例 1 中相同的功能，唯一的区别是，该示例中访问的设备不是当前设备的从设备，功能块程序如下所示。

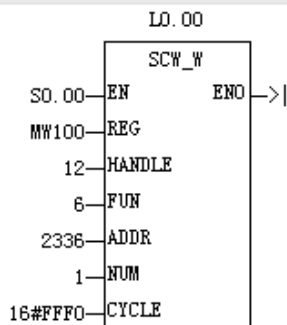
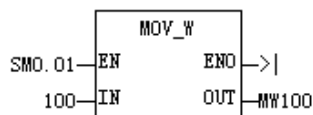


图 6.5.16 Modbus 总线设备资源映射

其中，CYCLE 为 0xFFF0，其高 12 位为 1，功能块运行在 Modbus 总线设备映射模式下，CYCLE 低 4 位为 0，表示扩展设备连接在当前设备的串口 0。HANDLE 指定了扩展 Modbus 站地址，此处为 12。

当 S0.0 使能时，串行通信指令执行一次资源映射，S0.0 一直保持有效直到扩展设备响应或者当前请求超时。

提示：

①功能块运行在从设备映射模式下时，所有的串行通信指令均是在从设备在线的状态下才能执行，从设备不在线则指令不会被执行；

②功能块运行在从设备映射模式下时，请不要映射从设备 AQ 和 DQ 变量到串行通信写指令，以免造成读写冲突使映射变量写入异常；

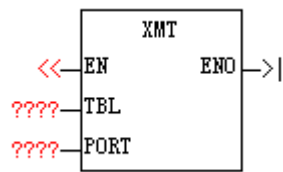
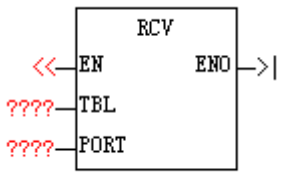
③功能块运行在从设备映射模式下时，建议将 CYCLE 参数设置为 0，使用上升沿指令触发 EN 来使用串行网络读指令，每个上升沿触发一次读命令，可有效避免数据阻塞；

④指令运行在 Modbus 总线设备映射模式下时，为了保证头顶参数能够正确的反映当前通信的状况，需要保持 EN 有效直到命令超时或者响应完毕。

6.5.5 串行自由通信

6.5.5.1 自由发送和自由接收指令

● 指令参数

串行自由发送指令（XMT）		 实现串行自由发送功能，将指定的数据不加任何报文头直接发送到串行总线上。	
串行自由接收指令（RCV）			
		实现串行自由接收功能，从串行总线上接收自定义格式数据，不做任何处理直接传送给指定变量区。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
TBL	字变量	WORD	XMT 指令时，发送缓冲区，第一个字为缓冲区

			长度；RCV 指令时，接收缓冲区，第一个字母用来保存接收报文字符数量，起始字符与接收字符不计入总数
PORT	常数：0 或 1	WORD	通信端口号，为 0 或 1
ENO	位变量	BOOL	输出能流

● 详细说明

在串行总线上传递的自由通信数据，语法、语义和语序全由用户指定，报文格式透明。

自由通信指令可触发串口发送完成中断和接收完成中断，对应的中断号可参见 6.14 章节，若通过中断指令使能相应的中断号，当发送完成或接收完成后会执行相应的中断函数。

端口配置为自由通信端口时，必须为主口，使用自由通信指令之前必须对相应的变量区进行配置。当端口只被用来做接收时，需要配置为被动接收模式，被动接收模式下只负责接收数据，不能进行数据发送，同时不能再用于其他串行通信。具体参看 SMW210 的配置。

提示：

①使用 XMT 指令时，一定要调用 RCV 指令。

②接收和发送指令的缓冲区以字为单位，故实际的接收和发送的字节数是每个字变量的低 8 位。

③配置为被动接收模式后，此端口不能再用于其它串行通信。

④接收和发送中断在自由通信完成后会自动产生，如果此时已经关联了中断程序，会自动进入中断程序执行，否则忽略。

⑤XMT 指令在每个程序中只能使用一次。

➤ 相关变量说明：

(1) SMW201:

位序从右向左表示由低到高，低八位用于配置端口 0，高八位用于配置端口 1。下面以低八位为例说明。

SMW201.7 ~ SMW201.0							
Reserve	SC	EC	0	0	0	0	0

该变量区用于配置 RCV 指令的接收模式，EC 位用于配置接收终止检测模式，SC 位用于配置接收启动检测模式，其余位变量保留，默认为 0，请勿修改。

位变量	0	1
EC	不使用 SMW203 检测结束字符	使用 SMW203 检测接收结束字符
SC	不使用 SMW202 检测起始字符	使用 SMW202 检测接收起始字符

(2) SMW202:

存放匹配起始字符，低字节对应端口 0，高字节对应端口 1。当配置为接收启动检测模式时，接收到与起始字符相同的数据后，程序开始向 RCV 的 TBL 区写入数据。

(3) SMW203:

存放匹配结束字符，低字节对应端口 0，高字节对应端口 1。当配置为接收终止检测模式时，接收到与终止字符相同的数据后，此次接收数据过程结束。

(4) SMW208 (端口 0) /SMW209 (端口 1):

存放最大接收字符长度。用户可以通过这个区配置 RCV 指令的最大接收字符数目。接收过程中将舍弃超过这一数目的所有数据。默认为 200，最大为 512 个字节。

(5) SMW210:

SMW210 为串口通信配置变量，结构如下：

...	210.7	210.6	210.5	210.4	210.3	210.2	210.1	210.0

SM210.0~SM210.5 用于显示自由通信运行状态。

SM210.6/SM210.7 用于配置自由通信端口，SM210.6 为 1 表示串口 0 为自由通信端口，SM210.7 为 1 表

示串口 1 为自由通信端口。SM210.8 为 1 表示 0 口启动自由通信被动接收模式，SM210.9 为 1 表示 1 口启动自由通信被动接收模式，高 6 位保留。

(6) SMW199:

自由通信状态变量:

SM199.0 为 1 表示 0 端口发送完毕信号;

SM199.1 为 1 表示 1 端口发送完毕信号。

(7) SMW211:

自由通信错误标志变量，低 8 位表示端口 0，高 8 位表示端口 1。以下以端口 0 为例，介绍每位所代表的含义。

SM211.0: 置位，表示 Recv 功能块 TBL 没有进行正确的配置;

SM211.1: 置位，表示通信使用不存在的端口;

SM211.2: 置位，表示通信使用端口没有使能自由通信格式;

SM211.3: 置位，表示用户指定的自由通信口不是主口;

SM211.4: 置位，表示启动模式错误;

SM211.5: 置位，表示终止模式错误。

➤ TBL 区说明

TBL 仅支持 V 区、L 区、M 区、S 区、XAQ 区、AQ 区和 PAQ 区。如果用户配置其它区，或者使用长度超出该区范围，则指令直接返回，无法得到正确的结果。

(1) XMT 指令 TBL 区

TBL 指针指向的第一个字，表示将要发送的数据长度。TBL 第二个字开始为发送信息区。发送 TBL 区结构（从低地址开始）:

2字节计数	…信息区 …

信息区按字寻址。每一个字地址的低八位，存放需要发送的数据，高八位无效。

高8位保留	有效数据	…	…	高8位保留	有效数据

(2) RCV 功能块 TBL 区

TBL 指针指向的第一个字，表示接收到的数据长度。

第二个字存放接收到的起始字符，若没有配置接收启动检测模式，该字存放的内容无效。

从 TBL 区第三个字地址开始是信息区，每一个字地址低八位存放接收到的数据，高八位无效。信息区长度等于 TBL 指针指向的第一个字。

TBL 区最后一个字符为结束字符，若没有配置接收结束检测模式，该字为 0。接收 TBL 区结构（从低地址开始）:

计数	起始字符	… …	结束字符

➤ 工作流程

(1) 发送数据过程:

发送过程可发送一个或者多个字节的缓冲区，暂时最多支持 255 个字节。

当发送指令将数据添加到串口发送队列中的时候，会产生一个发送完毕中断。（端口 0 对应中断号 16，端口 1 对应中断号 18）

(2) 接收数据过程:

当用户准备使用自由通信接收数据时，需要首先配置相应的通信口为自由通信口。

配置接收启动检测模式。若启动模式未配置，则默认为无需起始字符检测。如果 SM201.1 = 0，SM201.4 = 0，SM201.6 = 1 时，表示使用的接收启动检测模式，SMW202 中存放需要检测的起始字符。

配置接收终止检测模式，程序会在后台进入等待接收状态，其它组态程序照常运行。当通信端口有数据到来时，判断启动模式是否匹配，若匹配则进入自由通信数据接收过程，否则继续等待。当接收数据与结束

字符相匹配时，将数据存入到缓冲区中，接收过程结束。SM201.5 = 1 时，表示接收终止检测模式，SMW203 中存放匹配结束字符。

数据接收结束的时候会产生一个中断，中断号为 17（端口 0）和 19（端口 1），用户可根据需要，编写中断程序。此外，用户可强制 SMW201.7 为 0 时，结束接收过程。

● 应用示例：

(1) XMT 指令应用示例

想要使用 XMT 功能块，首先需要配置通信端口为自由通信口，而可以配置为自由通信口的端口必须为主口。

1) 打开控制器参数对话框，确认想要使用自由通信的端口为主口。

1号串口设置	
主从设置	主
通信协议	Modbus RTU
通信波特率	9600 bps
通讯基本设置	N, 8, 1
数据包间隔时间 (ms)	4
从设备通信最大允许错误次数	10
从设备通信时间间隔 (ms)	100
从设备通信最大等待时间 (ms)	1000

图 6.5.17 自由通信端口配置

2) 配置端口为自由通信模式。

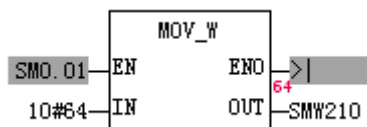


图 6.5.18 自由通信控制字配置

3) 将需要发送的包，写入发送缓冲区 TBL 中。

例如，想要发送数据包“E2 81 A0 A0”，使用 MW100 作为发送缓冲区。

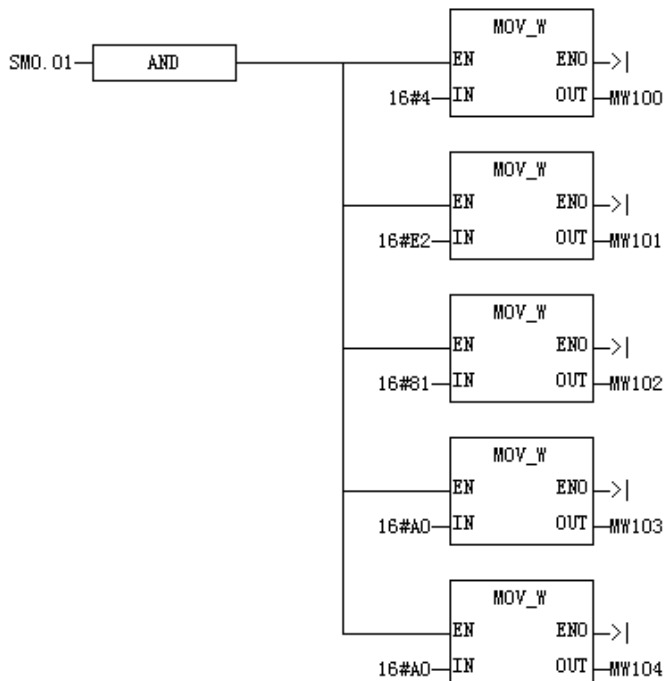


图 6.5.19 发送数据配置

其中，MW100 的第一个字存放的是想要发送数据包的长度，最长支持 255。

4) 发送数据

通过 SM0.1 控制，控制器上电只发送一次数据。

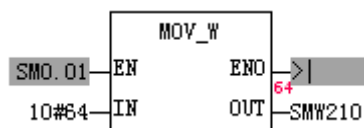


图 6.5.20 发送数据指令

(2) RCV 指令应用示例

使用 RCV 功能块也需要配置通信端口为自由通信模式，与 XMT 不同的是，RCV 功能块还需配置接收启动模式，也可以选择性的配置接收终止模式。

- 1) 确认使用的通信口为主口（参考 XMT 指令配置）
- 2) 配置通信端口使用自由通信模式（参考 XMT 指令配置）

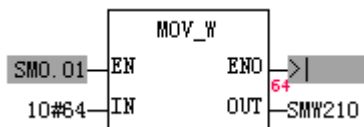


图 6.5.21 发送数据

3) 使用 SMW201 配置接收、启动模式。

对于该变量，如果用户不进行任何配置，程序默认接收模式为任意字符检测启动接收。

本例即使用任意字符检测启动接收，即配置完毕后，只要通信端口中有数据到来，就直接拷贝到接收缓冲区中。

SMW208 保存端口 0 最大接收长度，如果不对其进行配置，默认为 200 个字节，最大为 512 个字节。本例程配置为 20，即每次有数据到来的时候仅接收开始的 20 个字节数据，20 个字节之后的数据全都会被丢掉。

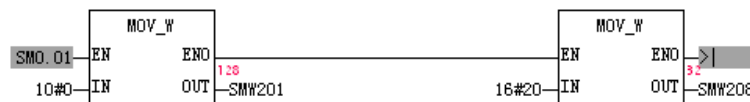


图 6.5.22 配置接收模式和最大接收长度

当然，也可以使用起始字符检测，SMW202 存放起始字符，配置如下：



图 6.5.23 配置接收模式和起始字符检测

4) 调用 RCV 接收功能块。使用 VW1000 作为接收缓冲区。

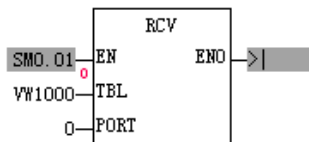


图 6.5.24 配置接收缓冲区

执行 RCV 功能块的时候会根据 SMW201 中配置的启动、终止模式，配置一个全局的接收模式变量。所以 RCV 功能块的调用，必须在接收启动模式和接收终止模式配置之后，否则接收模式就不能被正确的配置。

因为全局接收模式控制变量只有一个，所以一般情况下在整个工程中 RCV 功能块只需要执行一次。第二次执行 RCV 功能块会导致覆盖第一次的配置。

当端口有数据到来的时候，数据就会被拷贝到 VW1000 中。

- 工程示例

如图 6.5.25 示, 程序功能为配置 0 号端口为自由通信口, 利用定时中断每 3 秒钟发送 16 进制数据 0xE2, 0x81, 0xA0, 0xA0 到 0 号串行通信口, 接收数据放在 VW1000 起始的变量区中, 其中 VW1000 存储接收字节数, VW1001 开始存储接收到的数据, 数据变量中低 8 位为有效数据。

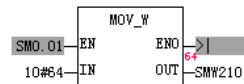
1) 程序结构



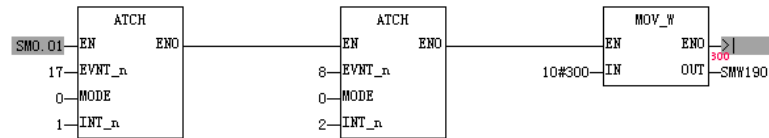
图 6.5.25 程序结构

2) 主程序

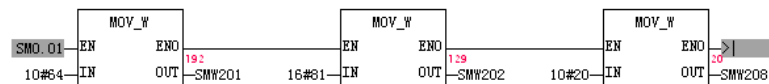
网络 1 配置端口0为自由通讯口



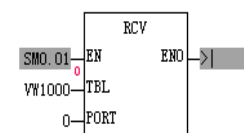
网络 2 连接中断，设置定时器中断超时时间



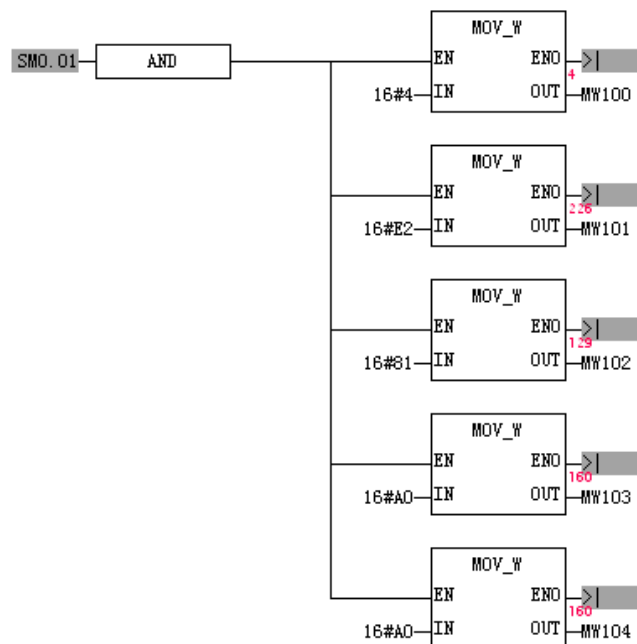
网络 3 配置接收模式寄存器，使用起始检测字符，最大接收长度20



网络 4 执行接收配置



网络 5 构造发送数据包



网络 6 注释...

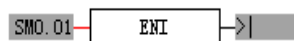
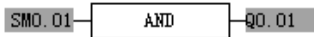


图 6.5.26 主程序

3) 中断子程序 RecvOK

每次进接收中断都置位Q0.1，显示接收是否完毕



注释...

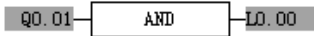


图 6.5.27 中断子程序 RecvOK

4) 中断子程序 Timer



图 6.5.28 中断子程序 Timer

6.5.5.2 自由发送接收(SC_AFC)指令

● 指令参数

串口自由通讯(SC_AFC)			
		实现串口发送、接收任意格式的数据信息	
操作数	变量区	数据类型	说明
EN	位变量	BOOL	使能位，高电平触发
REQ	位变量	BOOL	发送请求标志位，上升沿触发，检测到上升沿发送一次数据。
SBUF	字变量	WORD	发送缓存区起始地址，第一个字为发送的数据长度 512
RBUF	字变量	WORD	接收缓存区起始地址，第一个字为接收的数据长度 512
PORT	字变量	WORD	串口端口号
ENO	位变量	BOOL	能流输出
BUSY	位变量	BOOL	串口使用标志，为 1 时数据串口正在使用
DONE	位变量	BOOL	串口发送完成标志，为 1 时数据发送成功
ERR	字变量	WORD	串口发送错误状态字 0x0001 发送长度越界 0x0002 发送和接收缓存区为常数 0x0004 发送端口号为变量 0x0008 端口号不存在 0x0010 配置端口不是主口 0x0020 发送缓存区队列溢出

● 详细说明

串口自由通信指令可触发串口发送完成中断和接收完成中断，对应的中断号可参见串口中断向量表，若通过中断指令使能相应的中断号，当发送完成或接收完成后会执行相应的中断函数。

提示：

- ①每个串口只能调用一个串口自由通信指令；
- ②接收和发送中断在自由通信完成后会自动产生，如果此时已经关联了中断程序，会自动进入中断程序执行，否则

忽略。

③此端口不能再用于其它串行通信

④接收和发送指令的缓冲区以字节为单位，故实际的接收和发送的字节数是每个字变量低 8 位和高 8 位。

- 应用示例

如图 6.5.29 所示，该例程实现自由通讯报文的发送，SBUF 引脚中 VW0 代表发送串口报文长度，VW1 开始为发送缓存区，发送缓存区发送的数据是以字节格式发送，例如：VW1 中的数据先发送 VW1 中的低字节，再发送 VW1 中的高字节数据。RBUF 引脚中 VW600 代表接收串口报文长度，VW601 开始为接收缓存区，接收缓存区存储的数据是以字节格式存储，例如：VW601 中低字节存储的是先收到的数据，高字节存储的是后收到的数据。

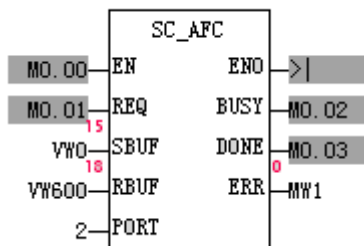


图 6.5.29 自由发送接收示例

表 6.5.3 串口中断向量表

中断号	MODE	中断事件	描述
16	0	串口 1 发送	串口 1 自由通信发送发送结束产生中断
17	0	串口 1 接收	串口 1 自由通信接收接收到报文产生中断
18	0	串口 2 发送	串口 2 自由通信发送发送结束产生中断
19	0	串口 2 接收	串口 2 自由通信接收接收到报文产生中断
20	0	串口 3 发送	串口 2 自由通信发送发送结束产生中断
21	0	串口 3 接收	串口 2 自由通信接收接收到报文产生中断

6.5.6 USS 通信

PEC 系列可编程控制器产品支持 USS 指令。

6.5.6.1 USS 初始化指令 (USS_INIT)

- 指令参数

USS 初始化指令 (USS_INIT)			
		用于使能 USS 通信，初始化所要操作的驱动器。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能控制位，1 为有效
PKWN	字变量、常数	WORD	参数通道的长度 (0, 3~120)
PZDN	字变量、常数	WORD	过程数据的长度 (0, 2, 4, 6, 8)
Active	字变量、常数	WORD	指示激活哪个驱动 (0~31)
ENO	位变量	BOOL	能流输出位
Error	字变量、常数	WORD	错误字

- 详细说明

USS 初始化指令负责配置当前 USS 命令所要操作的驱动器的地址、USS 报文长度等基本信息，是所有 USS 通信前必须调用的指令。

如果使用 USS 命令前没有进行初始化，在命令运行过程中将会返回一个执行错误编号，具体故障对应的编号请参看本小节后面的附表。

PKWN 引脚负责初始化 USS 报文中的 PKW，长度单位是字。根据 USS 协议的定义 PKWN 可以初始化为 0 或者 3~120 个字。

PZDN 引脚负责初始化 USS 报文中的 PZD，长度单位是字。根据 USS 协议的定义 PZDN 可以初始化为 0、2、4、6、8 中的任意一个。

Active 引脚负责初始化 USS 通信中需要使用的驱动器，Active 为一个双字，每一位代表驱动器的一个编号，从编号 0 到编号 31。根据 USS 协议定义，USS 通信中的目的驱动器仅支持 32 个（0~31），对应 Active 引脚的第 0~31 位。某一位被置 1 则激活该位对应编号的驱动器，此时 USS 读写功能块才可与该编号的驱动器进行通信。否则与该驱动器进行通信会返回 USS_DRIVER_NOT_INUSE 错误。

Error 引脚输出功能块执行的结果，为 0 时表示正常，否则输出一个错误号。用户可以根据错误号在本小节后面的附表中查到对应的错误说明。

● 应用示例

如图 6.5.30 所示，与编号为 2 和编号为 5 的驱动器通信，通信所使用的报文包含 4 个 PKW、不需要 PZD，则需要对 USS 协议进行如下初始化：

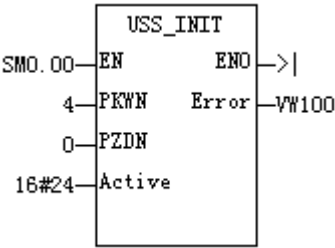


图 6.5.30 USS 初始化指令例程

6.5.6.2 USS 写指令

● 指令参数

USS 字写指令 (USS_WPM_W)	
USS_WPM_W <<—EN ENO—> ????—Type Error—???? ????—Drive ????—Param ????—IND ????—Port ????—DB_Buffer ????—Value	向目标设备写入一个字。
USS 双字写指令 (USS_WPM_DW)	向目标设备写入一个双字。

USS_WPM_DW <<EN ENO-> ????Type Error-???? ????Drive ????Param ????IND ????Port ????DB_Buffer ????Value			
USS 实数写指令 (USS_WPM_R)			
USS_WPM_R <<EN ENO-> ????Type Error-???? ????Drive ????Param ????IND ????Port ????DB_Buffer ????Value		向目标设备写入一个实数。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能控制位，1 为有效
Type	字变量、常数	WORD	Type: 修改方式选择。 3: 修改参数值（双字）; 8: 修改参数值（矩阵，双字）; 11: 修改参数值（矩阵，双字）并保存在 EEPROM 中; 13: 修改参数值（双字）并保存在 EEPROM 中; 14: 修改参数值（字）并保存在 EEPROM 中
Drive	字变量、常数	WORD	驱动地址。有效地址为 0~31
Parma	字变量、常数	WORD	参数号码
IND	字变量、常数	WORD	写参数的索引值
Port	常数	WORD	串口号，为 0 或 1
DB_Buffer	字变量、常数	WORD	16 字节缓存区地址，存储向 Micro Master 驱动发送的命令的执行结果
Value ^[1]	字变量	WORD	写入参数的数值
Value ^[2]	双字变量、常数	DWORD	写入参数的数值
Value ^[3]	双字变量、常数	REAL	写入参数的数值
ENO	位变量	BOOL	输出能流
Error	字变量	WORD	错误字

提示：USS 写指令可分为字、双字、实数三种类型，各指令的输入输出操作数名称都相同，只有 Value 引脚对应的数据类型不同。

● 详细说明

USS_WPM_x 系列指令负责向通过 USS 协议向驱动器写入变量，共有三种类型：

USS_WPM_W：写入字变量；

USS_WPM_DW：写入双字变量；

USS_WPM_R：写入实数变量；

根据 USS 协议的定义，USS 通信中的目的驱动器最多支持 32 个，驱动器地址为 0~31。

关于 Param 参数号、IND 索引的给定值，请参看具体目的驱动器的相关说明手册。因为通信目的变量不同，参数号的意义也不相同。目前功能块所支持的参数号最大为 8999。

Port 引脚为指定使用 USS 协议进行写通信的具体端口，为 0 或 1。

ENO 引脚在写命令返回响应或者超时后输出为 1，其余状态下输出 0。

Error 引脚输出功能块执行的结果，为 0 时表示正常，否则输出一个错误代码。用户可以根据错误代码在本小节后面的附表中查到对应的错误说明。

6.5.6.3 USS 读指令

● 指令参数

USS 字读指令(USS_RPM_W)			
USS_RPM_W <<—EN ENO—> ????—Type Error—???? ????—Drive Value—???? ????—Param ????—IND ????—Port ????—DB_Buffer		从目标设备读取一个字。	
USS 双字读指令 (USS_RPM_DW)			
USS_RPM_DW <<—EN ENO—> ????—Type Error—???? ????—Drive Value—???? ????—Param ????—IND ????—Port ????—DB_Buffer		从目标设备读取一个双字。	
USS 实数读指令(USS_RPM_R)			
USS_RPM_R <<—EN ENO—> ????—Type Error—???? ????—Drive Value—???? ????—Param ????—IND ????—Port ????—DB_Buffer		从目标设备读取一个实数。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能控制位，1 为有效
Type	字变量、常数	WORD	读的方式选择。 1：请求参数值； 4：请求参数描述部分； 6：请求参数值（矩阵） 9：请求矩阵包含的变量个数
Drive	字变量、常数	WORD	驱动地址。有效地址为 0~31

Parma	字变量、常数	WORD	参数号码
IND	字变量、常数	WORD	读参数的索引值
Port	常数	WORD	串口号，为 0 或 1
DB_Buffer	字变量	WORD	16 字节缓存区地址，用于存储向 Micro Master 驱动发送的命令的执行结果。
ENO	位变量	BOOL	输出能流
Error	字变量	WORD	错误字
Value ^[1]	字变量、常数	WORD	读取参数的数值
Value ^[2]	双字变量、常数	DWORD	读取参数的数值
Value ^[3]	双字变量、常数	REAL	读取参数的数值

● 详细说明

USS_RPM_x 系列指令负责向通过 USS 协议从驱动器读取变量，共有三种类型：

USS_RPM_W：读入字变量；

USS_RPM_DW：读入双字变量；

USS_RPM_R：读入实数变量；

根据 USS 协议的定义，USS 通信中的目的驱动器最多支持 32 个，驱动器地址为 0~31。

关于 Param 参数号、IND 索引的给定值，请参看具体目的驱动器的相关说明手册。因为通信目的变量不同参数号的意义也不相同。目前功能块所支持的参数号最大为 8999。

Port 引脚指定使用 USS 协议进行写通信的具体端口。对于当前下位机可以选择 0 号串口和 1 号串口两种。

Value 引脚输出读命令返回的读取值。

ENO 引脚在读命令返回响应或者超时后输出为 1，其余状态下输出 0；

Error 引脚输出功能块执行的结果，为 0 时表示正常，否则输出一个错误代码。用户可以根据错误代码在本小节后面的附表中查到对应的错误说明。

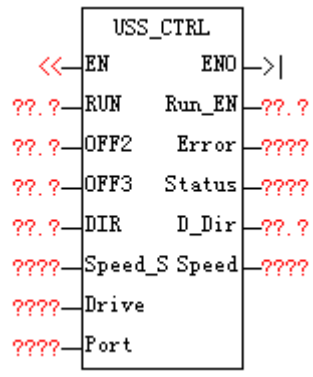
提示：

①读命令有字、双字、实数三种类型可以选择，在实际应用过程中具体使用哪种类型需要用户根据读取的 Param 区自行决定。例如：Param 所指定的参数是一个字，则使用 USS_RPM_W 功能块，如果 Param 所指定的参数是一个实数，则使用 USS_RPM_R 功能块。

②Param 对应的参数类型，用户请参看相关的驱动器手册。USS_RPM_x 系列指令仅负责将读取的值转换为指定的类型。

6.5.6.4 USS 控制指令 (USS_CTRL)

● 指令参数

USS 控制指令(USS_CTRL)			
		向目标设备发送带 PZD 的 USS 报文来控制变频器运行。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能控制位，1 为有效
RUN	位变量	BOOL	控制运行状态，有 0 和 1 两种状态，当 1 时发送 USS 报文控制驱动器

OFF2	位变量	BOOL	斜率停机
OFF3	位变量	BOOL	自由停机
DIR	位变量	BOOL	控制电机运行方向
Speed_S	双字变量、常数	REAL	电机运行频率设定
Drive	字变量、常数	WORD	驱动器地址。有效地址为 0~31
ENO	位变量	BOOL	能流传输位
Run_EN	位变量	BOOL	驱动器运行状态，0 为停止，1 为正在运行
Error	字变量	WORD	功能块执行过程中的错误字
Status	字变量	WORD	驱动器响应报文中的状态字
D_Dir	位变量	BOOL	驱动器实际运动方向
Speed	双字变量	REAL	驱动器实际运行频率

● 应用示例

(1) 使用 USS 协议命令前需要将连接有目的驱动器的串口配置为主口，如图 6.5.31 所示。

1号串口设置	
主从设置	主
通信协议	USS
通信波特率	9600 bps
通讯基本设置	偶, 8, 1
数据包间隔时间 (ms)	4
从设备通信最大允许错误次数	10
从设备通信时间间隔 (ms)	100
从设备通信最大等待时间 (ms)	1000

图 6.5.31 USS 通信参数配置图

波特率与基本通信参数需要与具体与之通信的驱动器保持一致，我们这里使用的是：(9600，偶，8，1)。

(2) 编写 USS 初始化功能块

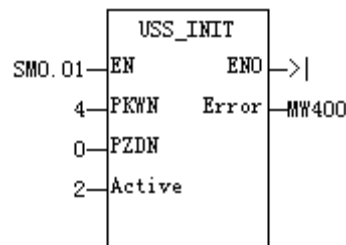


图 6.5.32 USS 初始化功能块配置

配置 PKWN 长度为 4 个字，PZDN 长度为 0。

假设驱动器地址为 x，则 Active 引脚的具体算法为：

$\text{Active} = (0x00000001 \ll x)$

本范例通信的驱动器地址为 1，则 Active 为 2。

(3) 编写 USS 实数写功能块

本范例将写入的参数区为 1001、索引值为 0 的浮点变量。编写功能块如下：

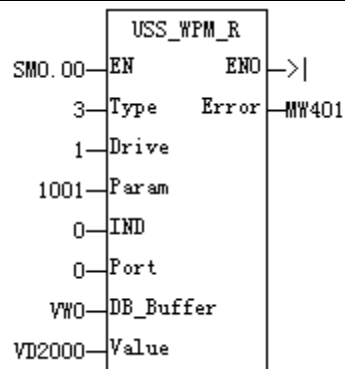


图 6.5.33 USS 实数写功能块编写示意图

Port 引脚使用在步骤 1 中配置过的通信口。

因为需要写入实数，故 Type 使用 3，代表修改参数值（双字）。

(4) 编写 USS 实数读功能块

本范例将刚刚写入到 1001、索引值为 0 的浮点变量再次读取回来。

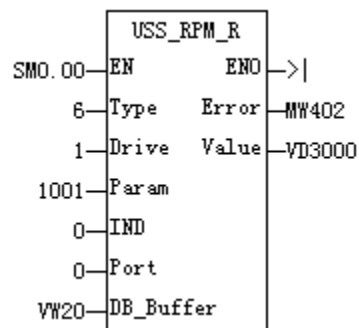


图 6.5.34 USS 实数读功能块编写示意图

(5) 编写 USS 驱动装置控制功能块

当使用 CTRL 功能块前，需要编写 USS_INIT 功能块，配置 PKWN=0，PZDN=2。

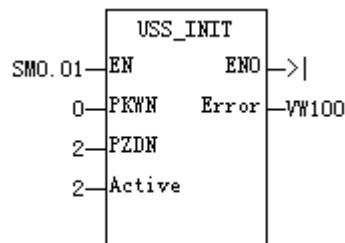


图 6.5.35 USS 通信初始化配置

然后编写 CTRL 功能块。RUN，OFF2 和 OFF3 同一时刻只能有一位为 1。当 EN (M0.5=1) 使能后，使能 RUN (M0.1=1)，被控运动器的运行频率为 Speed_S (VD200)中指定

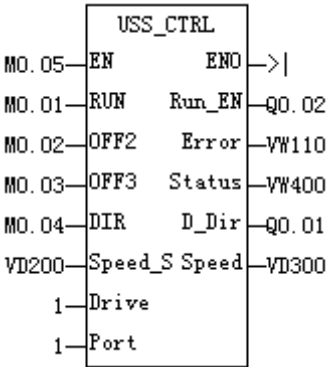


图 6.5.36 USS 驱动装置控制功能块编写示意图

提示：

- ①USS 通信功能块使用同步通信模型，同一时刻最多只有一个功能块使用串口进行通信，其他功能块查询到串口被占用直接返回。
- ②当组态工程中存在多个 USS 通信功能块的时候，排列在前面的 USS 通信功能块被执行的次数比排列在后面的功能块多。
- ③如果希望每个 USS 通信的功能块都能够等概率的被执行，则需要使用定时器，或者通过 USS 功能块执行完后在 ENO 引脚输出的上升沿来进行控制。

表 6.5.4 USS 通信错误代码

代码	描述	备注
0	没有错误	-
1	驱动器没有响应	驱动器是否在线或通信配置是否正确
2	BCC 校验错误	-
5	命令错误	确认 USS 读写功能块的 TYPE 输入正确
6	响应报文中的驱动器地址错误	-
8	串口忙	-
9	频率设定值超出最大限制	修改频率设定值
10	USS 报文长度错误	
15	请求修改的参数地址（Parma）无意义	修改参数地址
19	所操作的驱动器未被激活	编写 USS_INIT 功能块激活驱动器
20	收到负响应报文	-
23	USS 通信串口没有意义	只能是常数 0 或 1
25	USS 报文响应超时	-

6.5.7 MPI 通信

6.5.7.1 MPI 位读写指令

● 指令参数

MPI 位读指令 (X_GET_BIT)		MPI 位写指令 (X_PUT_BIT)		说明
参数名	变量区	数据类型		
X_GET_BIT <<EN ENO-> ??? REQ RET_VAL-??? ??? CONT BUSY-??? ??? DEST_ID ??? VAR_ADDR ??? RD ??? LEN ??? ID X_PUT_BIT <<EN ENO-> ??? REQ RET_VAL-??? ??? CONT BUSY-??? ??? DEST_ID ??? VAR_ADDR ??? SD ??? LEN ??? ID				

EN	位变量	BOOL	指令使能引脚，高电平触发
REQ	位变量	BOOL	通信使能位，高电平触发
CONT	位变量	BOOL	持续读命令控制位
DEST_ID	字变量	WORD	目标设备的 MPI 地址
VAR_ADDR	位变量	BOOL	目标设备（服务器端）的数据区
RD	位变量	BOOL	数据接收缓冲区
SD	位变量	BOOL	数据发送缓冲区
LEN	字变量	WORD	获取数据的长度，范围 1~76
ID	字变量，常数	WORD	指令编号，范围 0~255
ENO	位变量	BOOL	输出能流
RET_VAL	字变量	WORD	保留
BUSY	位变量	BOOL	通信进程，为 1 表示正在发送；为 0 表示发送完成

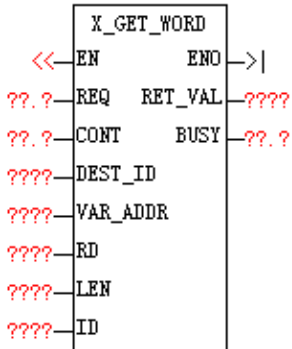
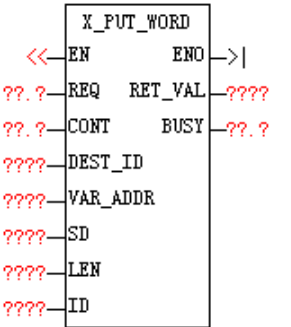
6.5.7.2 MPI 字节读写指令

● 指令参数

MPI 字节读指令(X_GET_BYTE)		MPI 字节写指令 (X_PUT_BYTE)		字节读指令读取目标设备中的一段字节变量；字节写指令向目标设备中写入一段字节变量。
X_GET_BYTE <<--EN ENO--> ???.?REQ RET_VAL--???.? ???.?CONT BUSY--???.? ????--DEST_ID ????--VAR_ADDR ????--RD ????--LEN ????--ID		X_PUT_BYTE <<--EN ENO--> ???.?REQ RET_VAL--???.? ???.?CONT BUSY--???.? ????--DEST_ID ????--VAR_ADDR ????--SD ????--LEN ????--ID		
参数名	变量区	数据类型	说明	
EN	位变量	BOOL	指令使能位，高电平触发	
REQ	位变量	BOOL	通信使能位，高电平触发	
CONT	位变量	BOOL	持续读命令控制位	
DEST_ID	字变量	WORD	目标设备 MPI 地址	
VAR_ADDR	字变量	WORD	目标设备（服务器端）的数据区	
RD	字变量	WORD	数据接收缓冲区	
SD	字变量	WORD	数据发送缓冲区	
LEN	字变量	WORD	获取数据的长度，范围 1~76	
ID	字变量，常数	WORD	指令编号，范围 0~255	
ENO	位变量	BOOL	输出能流	
RET_VAL	字变量	WORD	保留	
BUSY	位变量	BOOL	通信进程，为 1 表示正在发送；为 0 表示发送完成	

6.5.7.3 MPI 字读写指令

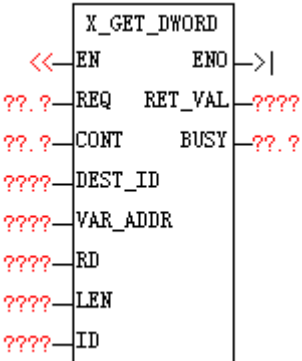
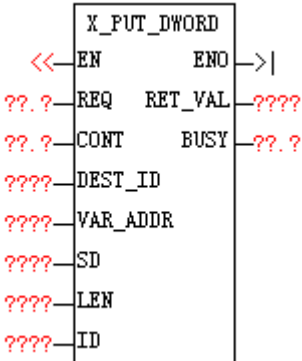
● 指令参数

MPI 字读指令(X_GET_WORD)		MPI 字写指令 (X_PUT_WORD)		字读指令读取目标设备中的一段字变量；字写指令向目标设备中写入一段字变量。
				
参数名	变量区	数据类型	数据范围	
EN	位变量	BOOL	指令使能位，高电平触发	
REQ	位变量	BOOL	通信使能位，高电平触发	
CONT	位变量	BOOL	持续读命令控制位	
DEST_ID	字变量	WORD	目标设备 MPI 地址	
VAR_ADDR	字变量	WORD	目标设备（服务器端）的数据区	
RD	字变量	WORD	数据接收缓冲区	
SD	字变量	WORD	数据发送缓冲区	
LEN	字变量	WORD	获取数据的长度，范围 1~38	
ID	字变量，常数	WORD	指令编号，范围 0~255	
ENO	位变量	BOOL	输出能流	
RET_VAL	字变量	WORD	保留	
BUSY	位变量	BOOL	通信进程，为 1 表示正在发送；为 0 表示发送完成	

字读指令读取目标设备中的一段字变量; 字写指令向目标设备中写入一段字变量。

6.5.7.4 MPI 双字读写指令

● 指令参数

MPI 双字读指令 (X_GET_DWORD)		MPI 双字写指令 (X_PUT_DWORD)		双字读指令读取目标设备中的一段双字变量；双字写指令向目标设备中写入一段双字变量。
				
参数名	变量区	数据类型	说明	
EN	位变量	BOOL	指令使能位，高电平触发	
REQ	位变量	BOOL	通信使能位，高电平触发	
CONT	位变量	BOOL	持续读命令控制位	
DEST_ID	字变量	WORD	目标设备 MPI 地址	
VAR_ADDR	双字变量	DWORD	目标设备（服务器端）的数据区	
RD	双字变量	DWORD	数据接收缓冲区	
SD	双字变量	DWORD	数据发送缓冲区	
LEN	字变量	WORD	获取数据的长度，范围 1~19	
ID	字变量	WORD	指令编号，范围 0~255	

双字读指令读取目标设备中的一段双字变量; 双字写指令向目标设备中写入一段双字变量。

ENO	位变量	BOOL	输出能流
RET_VAL	字变量	WORD	保留
BUSY	位变量	BOOL	通信进程，为 1 表示正在发送；为 0 表示发送完成

- 详细说明

通过 MPI 读写指令，控制器可代替 S7-300，利用 MPI 协议与 S7-200 进行读写通信，读写 S7-200 中 M 区、V 区、I 区、或 Q 区的变量。

MPI 读写指令共分为 4 类，包括位读写指令、字节读写指令、字读写指令、双字读写指令。其中，位读写指令只能读写 1 个 BIT 的数据，即 LEN 为 1；字节读写指令最多能读写 76 个 BYTE 的数据；字读写指令最多能读写 38 个 WORD 的数据；双字读写指令最多能读取 19 个 DWORD 的数据。

注意：

①使用 MPI 指令时，需要先进行串口选择，当 SMW201=0x4000 时，使用串口 1 作为 MPI 的通信接口；当 SMW201=0x0040 时，使用串口 0 作为 MPI 的通信接口；当未对 SMW201 进行配置时，默认选择串口 1 作为 MPI 的通信接口；当 SMW201=0x4040，指令将不会执行；

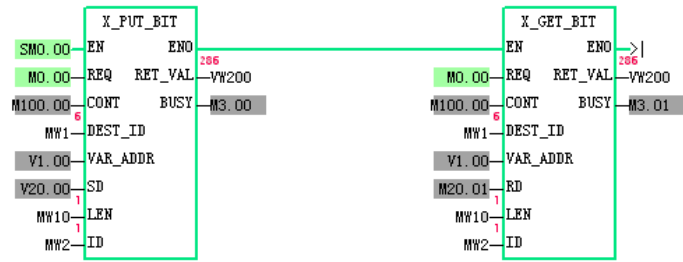
②MPI 支持的波特率有 9600 和 19200。

③支持 MPI 读写指令的控制器：运动控制系列产品、PEC2000、EDC8000 和 MAC 系列主控器。

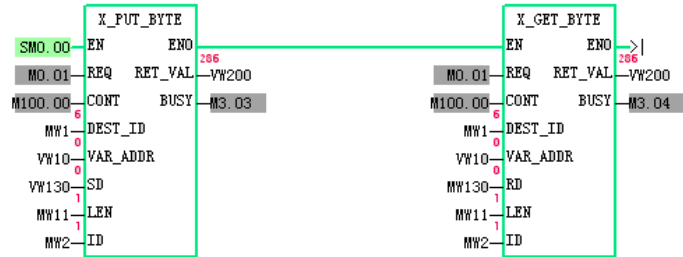
- 应用示例

如图 6.5.37 所示，本示例使用 EDC8000 作为通信控制器，控制 X_GET_BIT、X_PUT_BIT、X_GET_BYTE、X_PUT_BYTE、X_GET_WORD、X_PUT_WORD、X_GET_DWORD 和 X_PUT_DWORD 共八个指令，每个指令各两个，共十六个指令同时执行，对 S7-200 进行读取数据和写入数据。

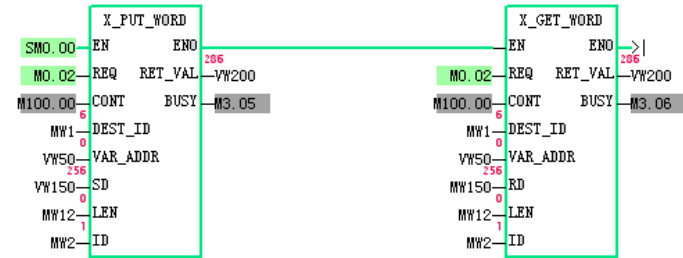
在本工程中，首先向 SMW201 中写入 16 进制数据 0x4000，使用串口 1 作为 MPI 通信端口。用户使用时，也可以在 MPI 指令之前使用 MOV_W 指令将此 16 进制数直接赋值给 SMW201。



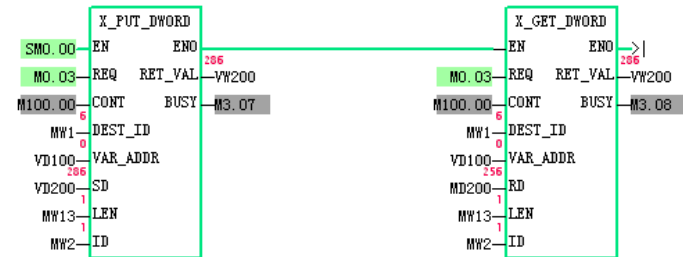
网络 2 用户注释



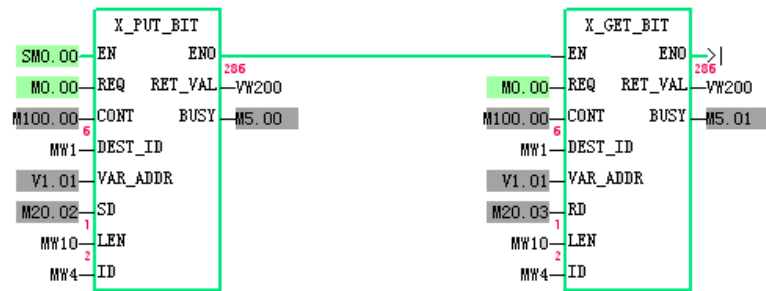
网络 3 用户注释



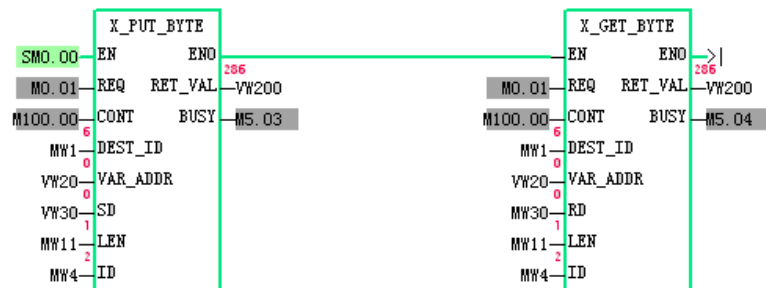
网络 4 用户注释



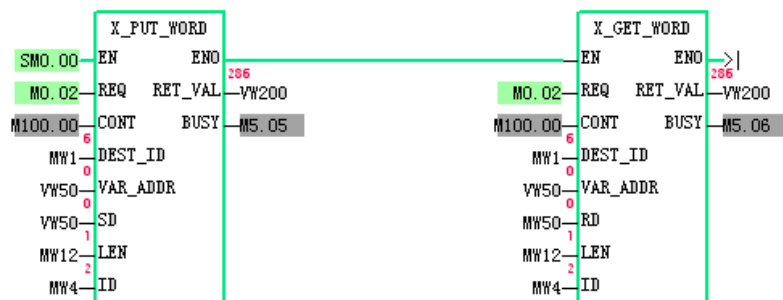
网络 5 用户注释



网络 6 用户注释



网络 7 用户注释



网络 8 用户注释

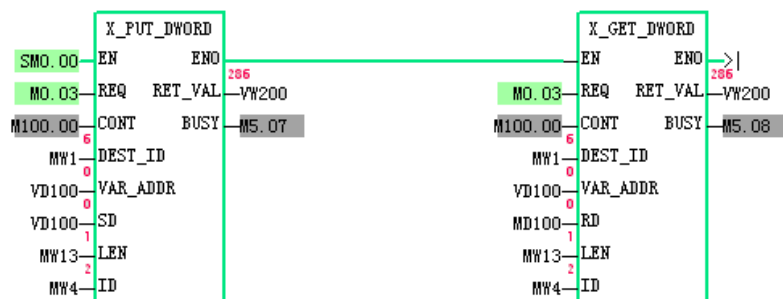


图 6.5.37 MPI 通信例程

准备工作

首先将 EDC8000 的侧串配置成主口，波特率为 19200，校验位为偶校验，数据位为 8 位，停止位为 1 位，通信协议为 MPI，如图 6.5.38 所示。

然后配置 S7-200 的波特率和站地址，波特率为 19200，站地址为 2~32，EDC8000 的地址可以通过拨码开关进行设定，范围同为 1~32，两者的地址不能相同。

控制器参数				
控制器ID号	00000000000000000000189198001010000		程序版本	1.0.00版
控制器序号	0189198001010000		CPU类型	保留
13	主从设置	从		
14	通信协议	Modbus RTU		
15	通信波特率	1M bps		
16	通讯基本设置	N, 8, 1		
17	数据包间隔时间 (ms)	2		
18	从设备通信最大允许错误次数	10		
19	从设备通信时间间隔 (ms)	100		
20	从设备通信最大等待时间 (ms)	1000		
21	2号串口设置			
22	主从设置	主		
23	通信协议	MPI		
24	通信波特率	19200 bps		
25	通讯基本设置	偶, 8, 1		
26	数据包间隔时间 (ms)	2		
27	从设备通信最大允许错误次数	10		
28	从设备通信时间间隔 (ms)	100		
29	从设备通信最大等待时间 (ms)	1000		

➤ 操作步骤:

- (1) 使能 MPI 指令前，需要对 SMW201 进行串口配置；
- (2) EN 高电平为指令使能；
- (3) REQ 高电平为通信使能，运行时，需要先使能 EN，再使能 REQ；
- (4) DEST_ID 为 S7-200 的站地址，此处为 6，该地址通过 STEP 7 MicroWIN 软件配置，其地址可以为 2~32；
- (5) VAR_ADDR 为 S7-200 被写入的地址；
- (6) SD 为 EDC8000 写入 S7-200 的数据的源地址；
- (7) LEN 为写入的长度，S7-200 只支持读写一个位变量区，所以使用 X_PUT_BIT 和 X_GET_BIT 指令时，LEN 必须为 1；
- (8) ID 为指令编号，相同的指令 ID 不能相同；
- (9) BUSY 为高电平表示此指令正在运行；
- (10) CONT 和 RET_VAL 功能保留；

注意：

- 第 172 页

6.6 比较指令

比较指令主要用于实现逻辑控制中的比较功能，包括无符号整数、有符号整数、长整数和实数的比较功能。其中，比较类型共有 6 种，分别为等于、不等于、大于等于、小于等于、大于、小于。

6.6.1 无符号整数

● 指令参数

等于指令			比较输入的两个无符号整数，如果两个输入相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
不等于指令			比较输入的两个无符号整数，如果两个输入不相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于等于指令			比较输入的两个无符号整数，如果输入变量 1 大于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
小于等于指令			比较输入的两个无符号整数，如果输入变量 1 小于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于指令			比较输入的两个无符号整数，如果输入变量 1 大于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
小于指令			比较输入的两个无符号整数，如果输入变量 1 小于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
参数名	变量区	数据类型	说明
IN1	字变量、常数	WORD	要输入的变量
IN2	字变量、常数	WORD	要输入的变量
OUT	位变量	BOOL	输出变量比较的结果

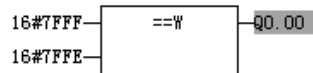
● 详细说明

无符号整数比较指令主要用于实现逻辑控制中的无符号整数比较功能，通过该指令可对两个无符号整数进行比较操作，比较输出的结果可作为控制信号。

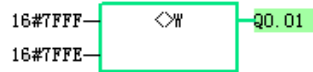
● 应用示例

如图 6.6.1 所示，例程中完成了 16 进制数 7FFF 和 7FFE 的 6 种无符号整数比较运算。网络 1，判断两数不相等，输出为 0；网络 2，判断两数不相等，输出为 1；网络 3，判断输入变量 1 大于等于输入变量 2，输出为 1；网络 4，判断输入变量 1 不小于等于输入变量 2，输出为 0；网络 5，判断输入变量 1 大于输入变量 2，输出为 1；网络 6，判断输入变量 1 不小于输入变量 2，输出为 0。

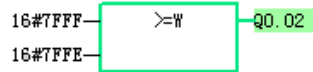
网络 1 用户注释



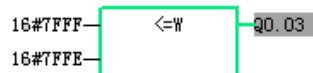
网络 2 用户注释



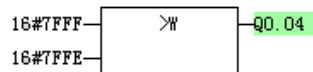
网络 3 用户注释



网络 4 用户注释



网络 5 用户注释



网络 6 用户注释

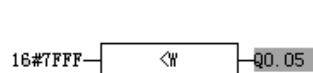
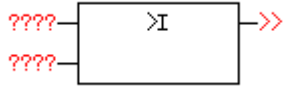
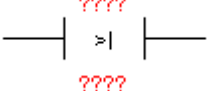
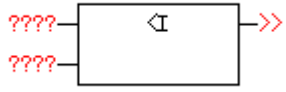
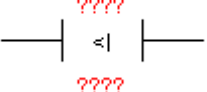


图 6.6.1 无符号整型字指令例程

6.6.2 有符号整数

● 指令参数

等于指令		
		比较输入的两个有符号整数，如果两个输入相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
不等于指令		
		比较输入的两个有符号整数，如果两个输入不相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于等于指令		
		比较输入的两个有符号整数，如果输入变量 1 大于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
小于等于指令		
		比较输入的两个有符号整数，如果输入变量 1 小于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于指令		
		比较输入的两个有符号整数，如果

			输入变量 1 大于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
小于指令			
			比较输入的两个有符号整数，如果输入变量 1 小于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
参数名	变量区	数据类型	说明
IN1	字变量、常数	INT	要输入的变量
IN2	字变量、常数	INT	要输入的变量
OUT	位变量	BOOL	输出变量比较的结果

● 详细说明

有符号整数比较指令主要用于实现逻辑控制中的有符号整数比较功能，通过该指令可对两个有符号整数进行比较操作，比较输出的结果可作为控制信号。

● 应用示例

如图 6.6.2 所示，例程中完成了 16 进制数 7FFF 和 7FFE 的 6 种有符号整数比较运算。网络 1，判断两数不相等，输出为 0；网络 2，判断两数不相等，输出为 1；网络 3，判断输入变量 1 大于等于输入变量 2，输出为 1；网络 4，判断输入变量 1 不小于等于输入变量 2，输出为 0；网络 5，判断输入变量 1 大于输入变量 2，输出为 1；网络 6，判断输入变量 1 不小于输入变量 2，输出为 0。

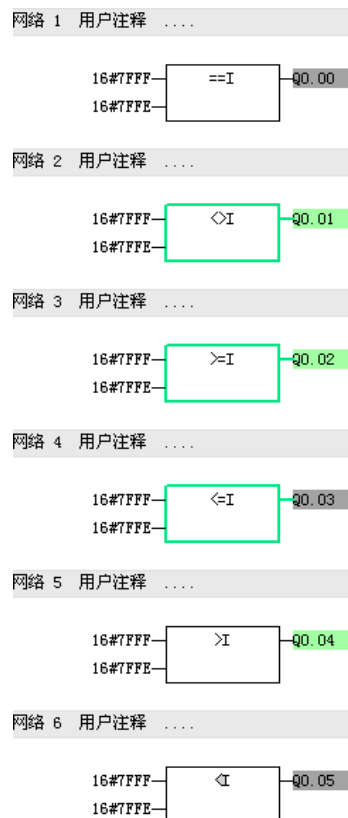


图 6.6.2 有符号整数指令例程

6.6.3 有符号长整数

● 指令参数

等于指令			比较输入的两个有符号长整数，如果两个输入相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
不等于指令			比较输入的两个有符号长整数，如果两个输入不相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于等于指令			比较输入的两个有符号长整数，如果输入变量 1 大于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
小于等于指令			比较输入的两个有符号长整数，如果输入变量 1 小于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于指令			比较输入的两个有符号长整数，如果输入变量 1 大于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
小于指令			比较输入的两个有符号长整数，如果输入变量 1 小于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
参数名	变量区	数据类型	说明
IN1	双字变量、常数	DINT	要输入的变量
IN2	双字变量、常数	DINT	要输入的变量
OUT	位变量	BOOL	输出变量比较的结果

● 详细说明

有符号长整数比较指令主要用于实现逻辑控制中的有符号长整数比较功能，通过该指令可对两个有符号长整数进行比较操作，比较输出的结果可作为控制信号。

● 应用示例

如图 6.6.3 所示，例程中完成了 16 进制数 7FFFFFFF 和 7FFFFFFE 的 6 种有符号长整数比较运算。网络 1，判断两数不相等，输出为 0；网络 2，判断两数不相等，输出为 1；网络 3，判断输入变量 1 大于等于输入变量 2，输出为 1；网络 4，判断输入变量 1 不小于等于输入变量 2，输出为 0；网络 5，判断输入变量 1 大于输入变量 2，输出为 1；网络 6，判断输入变量 1 不小于输入变量 2，输出为 0。

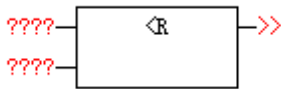
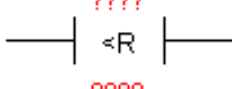


图 6.6.3 有符号长整数指令例程

6.6.4 实数

● 指令参数

等于指令		
		比较输入的两个实数，如果两个输入相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
不等于指令		
		比较输入的两个实数，如果两个输入不相等则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于等于指令		
		比较输入的两个实数，如果输入变量 1 大于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
小于等于指令		
		比较输入的两个实数，如果输入变量 1 小于等于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
大于指令		
		比较输入的两个实数，如果输入变量 1 大于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。

小于指令			比较输入的两个实数，如果输入变量 1 小于输入变量 2，则输出为 1，否则为零。输出引脚是 BOOL 型变量。
			
参数名	变量区	数据类型	说明
IN1	双字变量、常数	REAL	要输入的变量
IN2	双字变量、常数	REAL	要输入的变量
OUT	位变量	BOOL	输出变量比较的结果

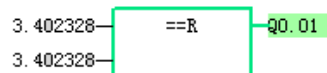
- 详细说明

实数比较指令主要用于实现逻辑控制中的实数比较功能，通过该指令可对两个实数进行比较操作，比较输出的结果可作为控制信号。

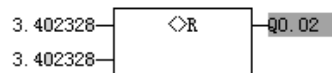
- 应用示例

如图 6.6.4 所示，例程中完成了 3.402328 和 3.402328 的 6 种实数比较运算。网络 1，判断两数相等，输出为 1；网络 2，判断两数相等，输出为 0；网络 3，判断输入变量 1 大于等于输入变量 2，输出为 1；网络 4，判断输入变量 1 小于等于输入变量 2，输出为 1；网络 5，判断输入变量 1 不大于输入变量 2，输出为 0；网络 6，判断输入变量 1 不小于输入变量 2，输出为 0。

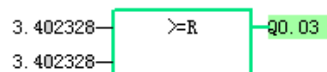
网络 1 用户注释



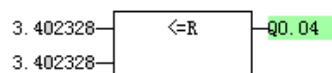
网络 2 用户注释



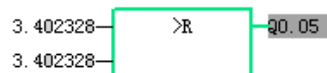
网络 3 用户注释



网络 4 用户注释



网络 5 用户注释



网络 6 用户注释

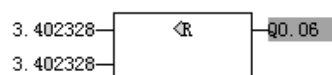


图 6.6.4 实数指令例程

6.7 转换指令

转换指令主要用于实现不同数据类型之间的强制转换功能，包含了实数、有符号整数、BCD 码、长整数变量之间的类型转换以及取整和四舍五入功能。其中，转换类型共有 6 种，分别为实数与整数、整数与 BCD 码、长整数与实数、长整数与整数转换、取整和四舍五入。

6.7.1 实数与整数

● 指令参数

实数整数转换指令 (R_I)			
		实现实数到有符号整数的转换，将输入的实数转换为 16 位有符号整数，结果送入 OUT。带有四舍五入功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入的变量
OUT	字变量	INT	输出变量转换的结果
ENO	位变量	BOOL	输出能流

整数实数转换指令 (I_R)			
		实现有符号整数到实数的转换，将输入的有符号整数 (IN) 转换为实数、结果送入 OUT。	
操作数	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	字变量、常数	INT	要输入的变量
OUT	双字变量	REAL	输出变量转换的结果
ENO	位变量	BOOL	输出能流

● 详细说明

实数与整数的转换主要用于浮点数和整数之间的类型转换，其中实数转换成整数时，会自动对数据进行四舍五入。

● 应用示例

如图 6.7.1 所示，该例程将实数 3.5678 转换为 16 位有符号整数，将整数 9 转换为实数表示。程序运行后，强制 Q0.1 为 1，使能 R_I 和 I_R 指令，其中，R_I 输出结果为 4，I_R 输出实数为 9.000000。

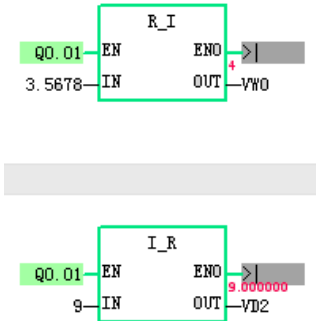
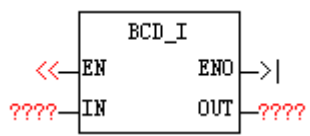
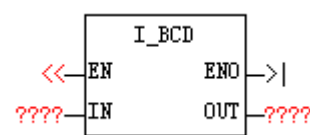


图 6.7.1 实数与整数转换指令例程

6.7.2 整数与 BCD 码

● 指令参数

BCD 码整数转换指令 (BCD_I)		实现 BCD 码到整数的转换，将输入的 BCD 码转换为整数，结果送入 OUT。输入 IN 的范围是 0-39321。	
			
整数实数转换指令 (I_BCD)		实现整数到 BCD 码的转换，将输入的整数转换为 BCD 码，结果送入 OUT。输入 IN 的范围是 0-9999。	
			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	字变量、常数	WORD	要输入变量的值
OUT	字变量	WORD	输出变量转换的结果
ENO	位变量	BOOL	输出能流

● 详细说明

BCD 码 (Binary-Coded Decimal) 是用 4 位二进制数来表示一个十进制数。所以用户在使用 BCD 码时建议使用状态符号表，采用十六进制的方式查看 BCD 码，这样更加直观。BCD 和整数转换指令就是将 BCD 码和实际存储时的实数之间的转换。例如 BCD 码 100 表示的是十进制的 100，但实际上存储该变量存储的是 0x100，使用 BCD_I 模块后才能转换成真正的 100，这时候输出端变量存储的是 0x64，即 100。反之，如果在输入端输入 100，使用 I_BCD 模块后，使用 16 进制方式查看输出变量，显示为 0x100。

BCD 码在某些场合使用比较方便，比如时间、金额等数字显示，可以直接打印，但是计算机中的 BCD 码不能直接参与计算，例如加减乘除这类的运算，只有通过转换，将 BCD 码转换成真正的二进制数据才能进行计算。

● 应用示例

如图 6.7.2 所示，该例程实现了 BCD 码与整数转换功能。网络 1：将输入端的十进制整数 100 转换成对应的 BCD 码实际存储值，使用 16 进制方式查看 VW1 变量即为 0x100；网络 2：与程序段 1 相反，通过调用 BCD_I 模块输出得到了 BCD 码 256 所要表示的真正整数的大小。

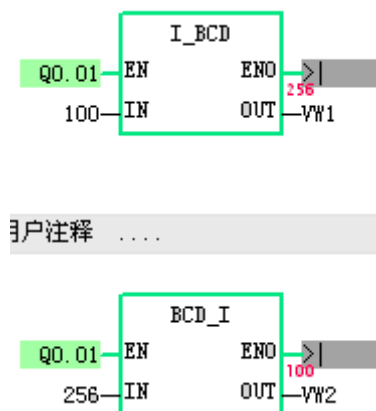


图 6.7.2 整数与 BCD 码转换指令例程

6.7.3 长整数与实数

● 指令参数

长整数实数转换指令 (DI_R)			
		实现长整数到实数的转换，将输入的 32 位有符号整数 (IN) 转换为实数、结果送入 OUT。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	DINT	输入变量的值
OUT	双字变量	REAL	输出变量转换的值
ENO	位变量	BOOL	输出能流
实数长整数转换指令 (R_DI)			
		实现实数到长整数的转换，将输入的实数 (IN) 转换为 32 位有符号整数、结果送入 OUT。带有四舍五入功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	输入变量的值
OUT	双字变量	DINT	输出变量转换的值
ENO	位变量	BOOL	输出能流

● 详细说明

实数与长整数的转换主要用于浮点数和长整数之间的类型转换，其中实数转换成长整数模块自动对数据进行四舍五入。

● 应用示例

如图 6.7.3 所示，该例程实现实数与长整数的转换，网络 1 将 32 位有符号整数 2147483647 转换为实数为 2147483648.000000。网络 2 将实数 100.0 转换为 32 位有符号整数 100。

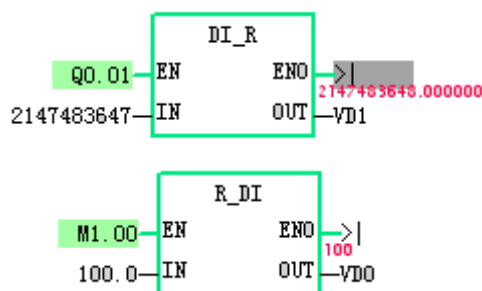
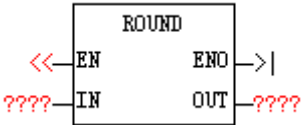
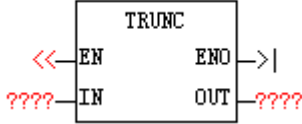


图 6.7.3 实数长整数转换指令例程

注意：受系统系统计算精度限制，有符号数转换为实数时推荐取 6 位有效数字。

6.7.4 取整

● 指令参数

四舍五入指令（ROUND）		实现四舍五入功能。将输入的实数（IN）转换为 32 位有符号整数，结果送入 OUT。如果大于 0.5 就进一位。	
			
取整指令（TRUNC）		实现取整功能，将输入的实数（IN）转换为 32 位有符号整数，结果送入 OUT。小数部分舍去。	
			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值
OUT	双字变量	DINT	输出变量取整的结果
ENO	位变量	BOOL	输出能流

● 详细说明

取整指令的输入端可以是任意 32 位实数，两个指令的取整方式不同，可以根据不同场合选择使用四舍五入取整还是舍去小数部分取整。

● 应用示例

如图 6.7.4 所示，该例程实现取整和四舍五入功能，网络 1 将实数 3.7895 转换为 32 为有符号整数 4，小数部分四舍五入。网络 2 将实数 3.7895 转换为 32 为有符号整数 3，舍去小数部分。

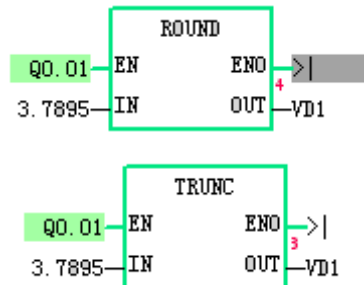
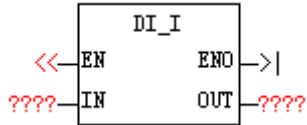
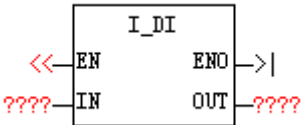


图 6.7.4 取整指令例程

6.7.5 长整数与整数

● 指令参数

长整数整数转换指令 (DI_I)			
		将输入的 32 位有符号整数 (IN) 转换为 16 位有符号整数，结果送入 OUT。如果要转换的数据太大，那么自动截断。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	DINT	要输入变量的值
OUT	字变量	INT	输出变量转换的结果
ENO	位变量	BOOL	输出能流
整数长整数转换指令 (I_DI)			
		将输入的 16 位有符号整数 (IN) 转换为 32 位有符号整数，结果送入 OUT。	
参数名	有效范围	数据类型	说明
EN	位变量	BOOL	输入能流
IN	字变量、常数	INT	要输入变量的值
OUT	双字变量	DINT	输出变量转换的结果
ENO	位变量	BOOL	输出能流

● 详细说明

长整数转换为整数指令会对越界输入变量进行处理，当输入的数大于 32767 会自动截断为-32766，当输入的数小于 -32767 时会输出 16 位整型数。

● 应用示例

如图 6.7.5 所示，该例程实现长整数与整数的转换。

网络 1，没有超过 16 位最大值，所以输出为 16 位有符号整数 32767。

网络 2，超过了最大 16 位最大值，自动截断，输出为-32766。

网络 3，输入为 16 位有符号整数 32767，可以输出到 VD4，得到 32 为有符号整数 32767。

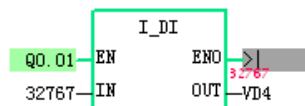
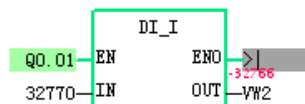
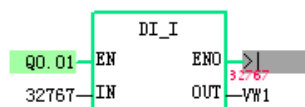
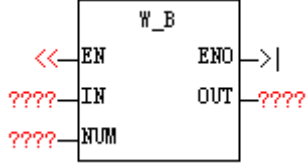
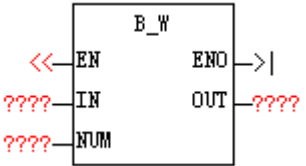


图 6.7.5 长整与整数转换指令例程

6.7.6 字与字节

● 指令参数

字转换字节指令 (W_B)			
		实现多个字变量拆分成字节变量	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位，高电平触发
IN	字变量、常数	INT	需要拆分字变量的首地址
NUM	字变量、常数	WORD	需要拆分字变量的个数(1~65535)
OUT	字变量	INT	拆分后存储字变量的首地址
ENO	位变量	BOOL	输出能流
字节转换字指令 (B_W)			
		实现多个字节变量合并成字变量	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位，高电平触发
IN	字变量、常数	INT	需要合并字节变量的首地址
NUM	字变量、常数	WORD	需要合并字节变量的个数(1~65535)
OUT	字变量	INT	合并后存储字变量的首地址
ENO	位变量	BOOL	输出能流

● 详细说明

该指令实现多个字节变量的合并。

● 应用示例

(1) 字节转字

如图 6.7.6 所示,该指令实现将 MW10 开始的 10 个变量,按照先放低字节在放高字节的顺序存入到 MW20 开始的字变量中。

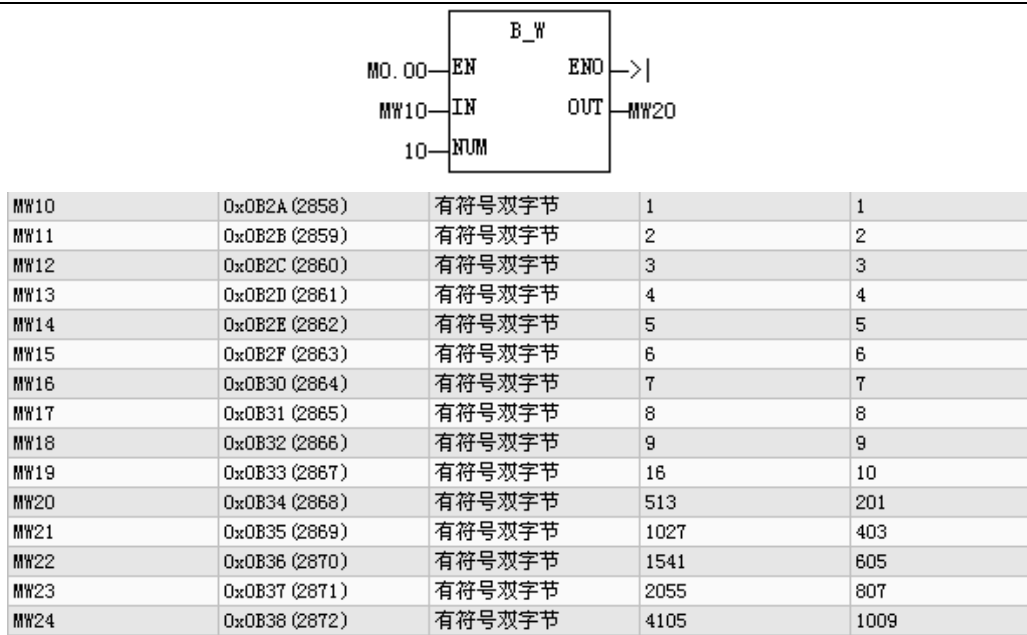


图 6.7.6 字节转字示例

(2) 字转字节

如图 6.7.7 所示,该指令实现将 MW20 开始的 5 个变量,按照先放低字节在放高字节的顺序存入到 MW30 开始的字变量中。

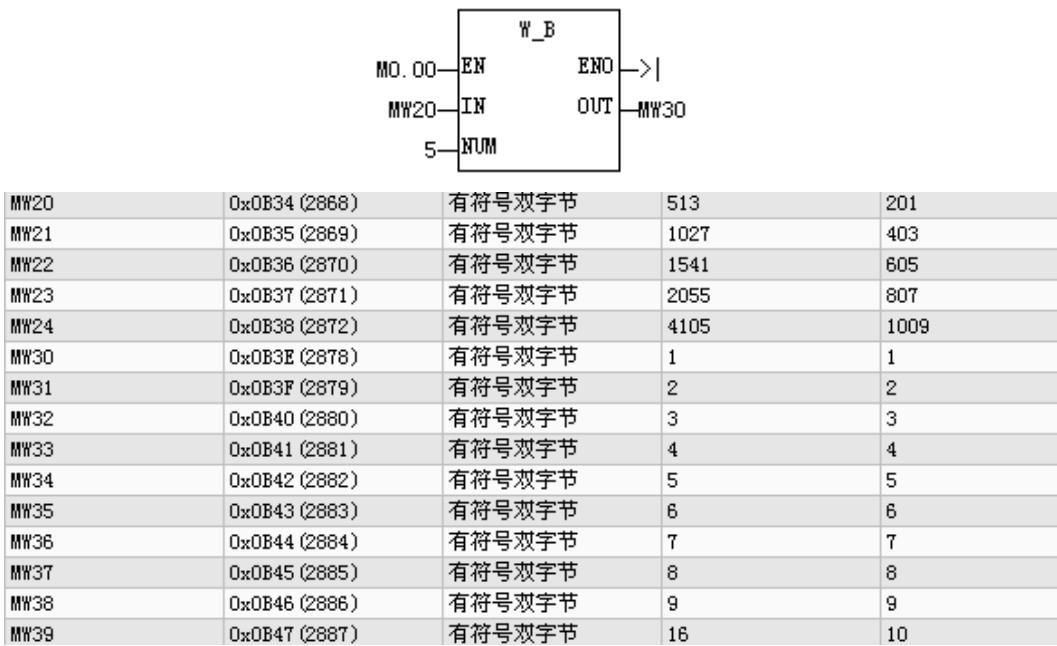


图 6.7.7 字转换字节例程

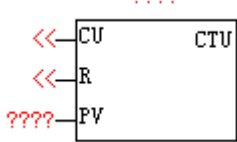
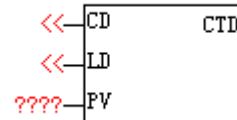
6.8 计数器指令

计数器指令主要用于实现逻辑控制中的增减计数功能，包括上升计数、下降计数和上升下降计数功能。

6.8.1 计数器

6.8.1.1 上升计数或下降计数器指令

● 指令参数

上升计数器指令(CTU)			
		实现加计数功能，在脉冲输入信号的上升沿，若当前的计数值小于 32767，计数值加 1。	
参数名	变量区	数据类型	说明
CU	位变量	BOOL	脉冲输入信号，在上升沿时，计数值加 1
R	位变量	BOOL	复位信号，为 1 时计数器被复位
PV	字变量、常数	INT	预设值
????	C 区	INT	显示当前计数值和计数器位
下降计数器指令(CTD)			
		实现减计数功能，在脉冲输入信号的上升沿，若当前的计数值大于 0，计数值减 1。	
参数名	变量区	数据类型	说明
CD	位变量	BOOL	脉冲输入信号，在上升沿时，计数值减 1
LD	位变量	BOOL	复位信号，为 1 时计数器被复位，将预设值装载到当前计数值中，计数器位变为 0
PV	字变量、常数	INT	预设值
????	C 区	INT	显示当前计数值和计数器位

● 详细说明

每个计数器都有一个 16 位的字和一个二进制位，计数器的字用来存储它的当前计数值，计数器触点的状态由它的位状态决定。用计数器地址（C 和计数器号，例如 C10）来访问当前计数值和计数器位。

上升计数器指令在脉冲输入信号 CU 的上升沿，计数值加 1。PV 为预设值，当前计数值大于等于预设值时，计数器位置位，计数器仍计数，直到计数值到达 32767 时停止计数。当复位信号 R 为 1 的时计数器复位，当前计数值被清零，计数器位清零。

下降计数器指令在脉冲输入信号 CD 的上升沿，计数值减 1。PV 为预设值，当前计数值为 0 时，计数器位置位，停止计数。当复位信号 LD 为 1 的时，将 PV 装载到当前计数值中，计数器位清零。可使用 R 指令对上升/下降计数器指令复位。

● 应用示例

如图 6.8.1 所示，例程中，采用下降计数器指令说明计数器指令的使用方法。当复位信号 I0.01(LD)为 1 时，C1 初始化为 PV 值，计数器位 Q0.0 清零。当复位信号 I0.01(LD)为 0 时，每在脉冲输入信号 I0.00(CD)的上升沿，C1 计数值减 1。当计数值为 0 时计数器位 Q0.0 输出高电平并保持至 LD 置位。

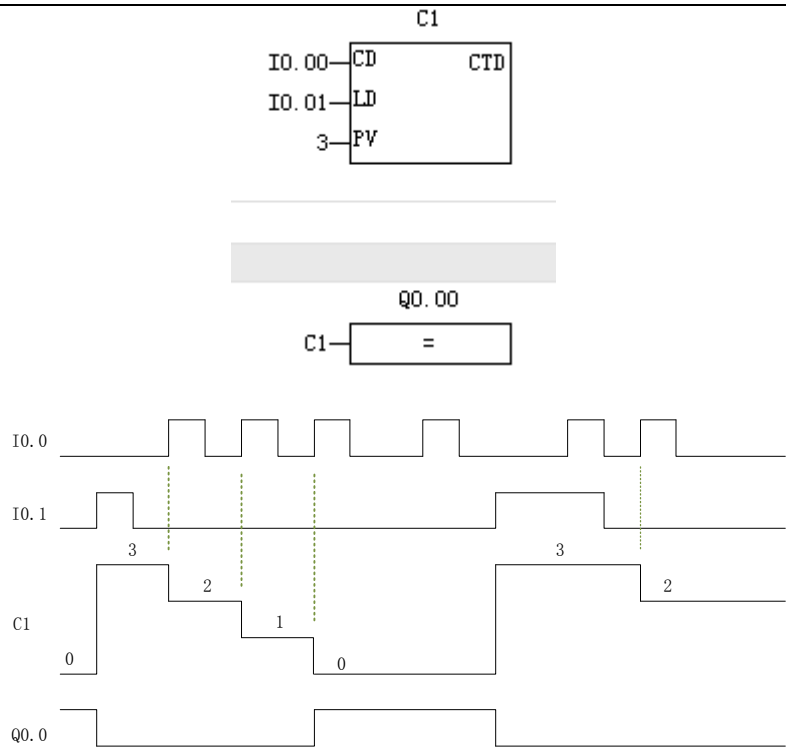


图 6.8.1 下降计数器例程

6.8.1.2 上升下降计数器指令

● 指令参数

上升下降计数器指令(CTUD)			
<<— <<— <<— ????— CUCTUD CD R PV			
实现加减计数功能，在脉冲输入信号 CU 的上升沿，若当前的计数值小于 32767，计数值加 1。在脉冲输入信号 CD 的上升沿，若当前的计数值大于 0，计数值减 1。			
参数名	变量区	数据类型	说明
CU	位变量	BOOL	上升脉冲输入信号，在上升沿时，计数值加 1
CD	位变量	BOOL	下降脉冲输入信号，在上升沿时，计数值减 1
R	位变量	BOOL	复位信号，为 1 时计数器被复位
PV	字变量、常数	INT	预设值
????	C 区	INT	显示当前计数值和计数器位

● 详细说明

上升下降计数器指令，在 CU 的上升沿，计数值加 1。在 CD 的上升沿，计数值减 1。当前计数值大于等于预设值时，计数器位置位，计数器仍计数。当复位信号 R 为 1 的时计数器复位，当前计数值被清零，计数器位清零。可使用 R 指令复位上升下降计数器指令。

计数器的当前计数值为 32767 时，下一个 CU 的上升沿将使计数值变化为-32768。在当前计数值为-32768 时，下一个 CD 的上升沿将使计数值到达 32767。

● 应用示例

如图 6.8.2 所示，例程中，采用上升下降计数器指令说明计数器指令的使用方法。当 I0.0 触发上升沿时，C48 计数。当计数值达到 4 时，Q0.0 置位；当 I0.1 产生上升沿时，计数值减 1，直到计数小于 4 时，Q0.0 复位，I0.0 产生上升沿后，数据增加直至变为 4，Q0.0 重新置位。当 I0.2 置位时，当前计数值清零，且 Q0.0 复位；

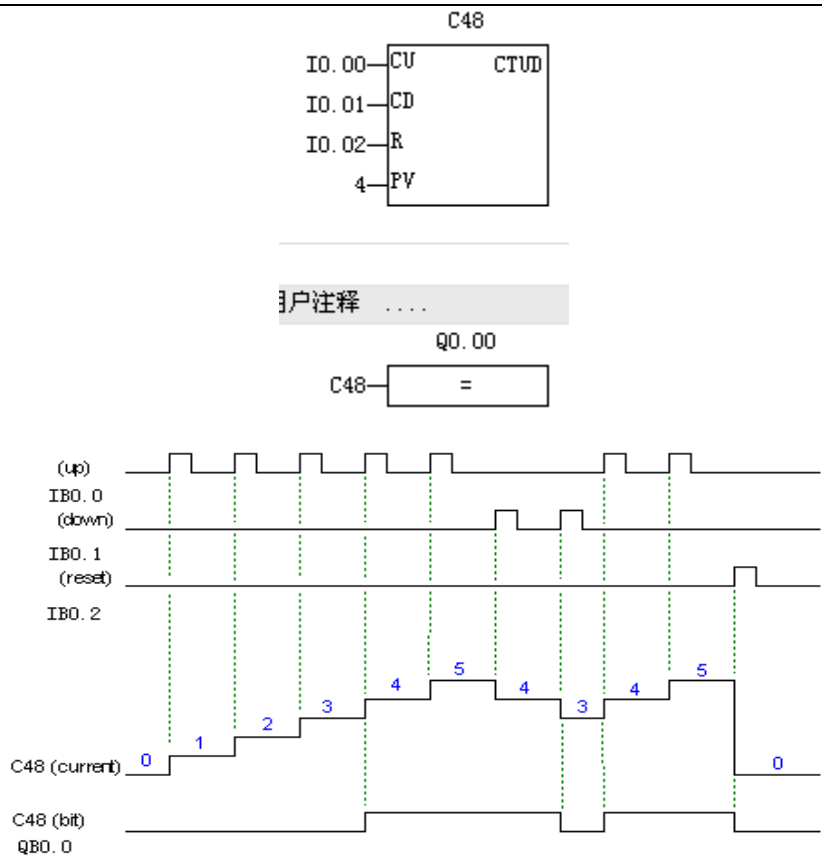


图 6.8.2 上升下降计数器例程

6.8.2 高速计数指令

● 指令参数

高速计数指令(HDEF)			
HDEF <<—EN ????—SP ????—PV ????—MODE ENO—> CV—???? FEQ—????		实现高速输入计数和计频功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
SP	双字变量、常数	DINT	设定脉冲值
PV	双字变量、常数	DINT	脉冲计数起始值
MODE	字变量、常数	WORD	高速计数模式
ENO	位变量	BOOL	输出能流
CV	双字变量	DINT	当前脉冲计数个数
FEQ	双字变量	REAL	显示当前采集脉冲的频率

● 详细说明

高速计数指令通常用来统计编码器的输出脉冲，根据频率值确定速度，计算出控制输出数字量，对电机或其他位置控制器进行控制。使能位被置“1”后，根据控制模式 MODE 开始采集控制器的高速数字量输入，CV 值会根据设置的增减模式加 1 或减 1，同时 FEQ 值给出当前采集脉冲的频率。

➤ 高速计数物理引脚说明

具有高速输入引脚的控制器支持 HDEF 指令如表 6.8.1 所示。

表 6.8.1 控制器对应高速输入引脚及频率范围

控制器	高速 DI 物理引脚	最低频率	最高频率
PEC8000/PMC2000	I04~I07	1Hz	20KHz
PEC3000/DIO3000	I20~I23	1Hz	20KHz
EDC8000	I00~I07	1Hz	1MHz
运动控制器及 MAC1610	HI0~HI7	0Hz	1MHz
DEC 系列控制器	I00、I01、I02、I03	10 Hz	1MHz

➤ 高速计数测频方式

PEC3000、DIO3000、PEC8000、PMC2000 的高速计数器测频方法可分为测频和测周两种。可通过 SMW187、SMW188、SMW189 配置测频方式和滤波次数，其中 SMW187 为测频和测周的分界频率值，当输入脉冲的频率值高于分界频率值时使用测频法计算频率，低于分界频率值时使用测周法计算频率，SMW188 和 SMW189 分别对应两路高速输入的滤波次数，可通过设置此参数来避免现场的干扰。

➤ 高速计数模式说明

HDEF 指令支持输入方式：内部控制单相计数、外部控制单相计数、增减时钟双相计数、脉冲+方向模式、正反向模式和 AB 相正交计数。

提示：

①使用外部控制单相计数时，指令的使能引脚 EN 是不起作用的，这一点请用户注意。

②可通过设置 SM63.8 改变输入的 AB 相信号相序。

表 6.8.2 PEC8000 和 PMC2000 高速计数模式一览表

引脚	计数模式	方向	MODE	描述	计数器
I04	内部	增	0x0300	内部控制单相增计数	3
		减	0x0301	内部控制单相减计数	
I05	内部	增	0x0200	内部控制单相增计数	2
		减	0x0201	内部控制单相减计数	
I06	内部	增	0x0100	内部控制单相增计数	1
		将	0x0101	内部控制单相减计数	
	外部	增	0x0500	外部控制单相计数，I04 为 0 时增计数	
		减		外部控制单相计数，I04 为 1 时减计数	
		增	0x0501	外部控制单相计数，I04 为 0 时增计数，I02 为 1 时复位	
I06	外部	减	0x0501	外部控制单相计数，I04 为 1 时减计数，I02 为 1 时复位	1
		增	0x0502	外部控制单相计数，I04 为 0 时增计数，I02 为 1 时复位，I00 为 1 启动	
		减		外部控制单相计数，I04 为 1 时减计数，I02 为 1 时复位，I00 为 1 启动	
I07	内部	增	0x0000	内部控制单相增计数	0
		减	0x0001	内部控制单相减计数	
	外部	增	0x0400	外部控制单相计数，I05 为 0 时增计数	
		减		外部控制单相计数，I05 为 1 时减计数	
		增	0x0401	外部控制单相计数，I05 为 0 时增计数，I03 为 1 时复位	
		减		外部控制单相计数，I05 为 1 时减计数，I03 为 1 时复位	
		增	0x0402	外部控制单相计数，I05 为 0 时增计数，I03 为 1 时复位，I01 为 1 启动	
		减		外部控制单相计数，I05 为 1 时减计数，I03 为 1 时复位，I01 为 1 启动	
I04+ I06	时钟双相	增	0x1002	I06 为上升沿时，输入加 1，为增计数	1
		减		I04 为下降沿时，输入减 1，为减计数	
	AB 相	增	0xFF02	I04 为 A 相时钟，I06 为 B 相时钟，I04 上升沿先	

I05+ I07	时钟双相	增	0x1000	于 I06 上升沿 90 度到来, 为增计数	0
		减		I04 为 A 相时钟, I06 为 B 相时钟, I04 上升沿后于 I06 上升沿 90 度到来, 为减计数	
	AB 相	增	0xFF03	I07 为上升沿时, 输入加 1, 为增计数	0
		减		I05 为下降沿时, 输入减 1, 为减计数	
I06+ I07	时钟双相	增	0xFF00	I05 为 A 相时钟, I07 为 B 相时钟, I05 上升沿先于 I07 上升沿 90 度到来, 为增计数	0
		减		I05 为 A 相时钟, I07 为 B 相时钟, I05 上升沿后于 I07 上升沿 90 度到来, 为减计数	
	AB 相	增	0xFF01	I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿先于 I06 上升沿 90 度到来, 为增计数, 4 倍速模式	
		减		I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿后于 I06 上升沿 90 度到来, 为减计数, 4 倍速模式	
		增	0xFF04	I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿先于 I06 上升沿 90 度到来, 为增计数, 2 倍速模式	
		减		I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿后于 I06 上升沿 90 度到来, 为减计数, 2 倍速模式	

表 6.8.3 PEC3000 和 DIO3000 高速计数模式一览表

引脚	计数模式	方向	MODE	描述	计数器
I20	内部	增	0x0300	内部控制单相增计数	3
		减	0x0301	内部控制单相减计数	
I21	内部	增	0x0200	内部控制单相增计数	2
		减	0x0201	内部控制单相减计数	
I22	内部	增	0x0000	内部控制单相增计数	0
		将	0x0001	内部控制单相减计数	
	外部	增	0x0400	外部控制单相计数, I20 为 0 时增计数	
		减		外部控制单相计数, I20 为 1 时减计数	
		增	0x0401	外部控制单相计数, I20 为 0 时增计数, I18 为 1 时复位	
		减		外部控制单相计数, I20 为 1 时减计数, I18 为 1 时复位	
I22	外部	增	0x0402	外部控制单相计数, I20 为 0 时增计数, I18 为 1 时复位, I16 为 1 启动	0
		减		外部控制单相计数, I20 为 1 时减计数, I18 为 1 时复位, I16 为 1 启动	
I23	内部	增	0x0100	内部控制单相增计数	1
		减	0x0101	内部控制单相增计数	
	外部	增	0x0500	外部控制单相计数, I21 为 0 时增计数	
		减		外部控制单相计数, I21 为 1 时减计数	
		增	0x0501	外部控制单相计数, I21 为 0 时增计数, I19 为 1 时复位	
		减		外部控制单相计数, I21 为 1 时减计数, I19 为 1 时复位	
		增	0x0502	外部控制单相计数, I21 为 0 时增计数, I19 为 1 时复位, I17 为 1 启动	
		减		外部控制单相计数, I21 为 1 时减计数, I19 为 1 时复位, I17 为 1 启动	

I20+ I22	时钟双相	增	0x1000	I22 为上升沿时, 输入加 1, 为增计数	0
		减		I20 为下降沿时, 输入减 1, 为减计数	
	AB 相	增	0xFF02	I20 为 A 相时钟, I22 为 B 相时钟, I20 上升沿先于 I22 上升沿 90 度到来, 为增计数	
		减		I20 为 A 相时钟, I22 为 B 相时钟, I20 上升沿后于 I22 上升沿 90 度到来, 为减计数	
I21+ I23	时钟双相	增	0x1002	I23 为上升沿时, 输入加 1, 为增计数	1
		减		I21 为下降沿时, 输入减 1, 为减计数	
	AB 相	增	0xFF03	I21 为 A 相时钟, I23 为 B 相时钟, I21 上升沿先于 I23 上升沿 90 度到来, 为增计数	
		减		I21 为 A 相时钟, I23 为 B 相时钟, I21 上升沿后于 I23 上升沿 90 度到来, 为减计数	
I22+ I23	AB 相	增	0xFF00	I23 为 A 相时钟, I22 为 B 相时钟, I23 上升沿先于 I22 上升沿 90 度到来, 为增计数	0
		减		I23 为 A 相时钟, I22 为 B 相时钟, I23 上升沿后于 I22 上升沿 90 度到来, 为减计数	
I22+ I23	AB 相	增	0xFF01	I23 为 A 相时钟, I22 为 B 相时钟, I23 上升沿先于 I22 上升沿 90 度到来, 为增计数, 4 倍速模式	0
		减		I23 为 A 相时钟, I22 为 B 相时钟, I23 上升沿后于 I22 上升沿 90 度到来, 为减计数, 4 倍速模式	
		增	0xFF04	I23 为 A 相时钟, I22 为 B 相时钟, I23 上升沿先于 I22 上升沿 90 度到来, 为增计数, 2 倍速模式	
		减		I23 为 A 相时钟, I22 为 B 相时钟, I23 上升沿后于 I22 上升沿 90 度到来, 为减计数, 2 倍速模式	

表 6.8.4 PEC3000 和 DIO3000 普通 DI 高速计数模式一览表

引脚	方向	MODE	描述
I00	增	0x2000	内部控制单相增计数
	减	0x2001	内部控制单相减计数
I01	增	0x2100	内部控制单相增计数
	减	0x2101	内部控制单相减计数
I02	增	0x2200	内部控制单相增计数
	将	0x2201	内部控制单相减计数
I03	增	0x2300	内部控制单相增计数
	减	0x2301	内部控制单相减计数
I04	增	0x2400	内部控制单相增计数
	减	0x2401	内部控制单相减计数
I05	增	0x2500	内部控制单相增计数
	减	0x2501	内部控制单相减计数
I06	增	0x2600	内部控制单相增计数
	减	0x2601	内部控制单相减计数
I07	增	0x2700	内部控制单相增计数
	减	0x2701	内部控制单相减计数
I12	增	0x2800	内部控制单相增计数
	减	0x2801	内部控制单相减计数
I13	增	0x2900	内部控制单相增计数
	减	0x2901	内部控制单相减计数
I14	增	0x2A00	内部控制单相增计数
I14	减	0x2A01	内部控制单相减计数

I15	增	0x2B00	内部控制单相增计数
	减	0x2B01	内部控制单相减计数

表 6.8.5 EDC8000 高速计数模式一览表

引脚	计数模式	方向	MODE	描述	计数器
I01	内部	增	0x0200	内部控制单相增计数	编码器 0
		减	0x0201	内部控制单相减计数	
	外部	增	0x0600	外部控制单相计数, I00 为 0 时增计数	编码器 0
		减		外部控制单相计数, I01 为 1 时减计数	
I03	内部	增	0x0300	内部控制单相增计数	编码器 1
		减	0x0301	内部控制单相减计数	
	外部	增	0x0700	外部控制单相计数, I02 为 0 时增计数	编码器 1
		减		外部控制单相计数, I02 为 1 时减计数	
I04	内部	增	0x0030	内部控制单相增计数	3
		将	0x0031	内部控制单相减计数	
I05	内部	增	0x0020	内部控制单相增计数	1
		减	0x0021	内部控制单相减计数	
	外部	增	0x0500	外部控制单相计数, I04 为 0 时增计数	
		减		外部控制单相计数, I04 为 1 时减计数	
I06	内部	增	0x0010	内部控制单相增计数	0
		减	0x0011	内部控制单相减计数	
	外部	增	0x0400	外部控制单相计数, I07 为 0 时增计数	
		减		外部控制单相计数, I07 为 1 时减计数	
I07	内部	增	0x0000	内部控制单相增计数	2
		减	0x0001	内部控制单相减计数	
I00 + I01	AB 相	增	0xFF04	I00 为 A 相时钟, I01 为 B 相时钟, I00 上升沿先于 I01 上升沿 90 度到来, 为增计数, 单倍速模式	编码器 0
		减		I00 为 A 相时钟, I01 为 B 相时钟, I00 上升沿后于 I01 上升沿 90 度到来, 为减计数, 单倍速模式	
		增	0xFF05	I00 为 A 相时钟, I01 为 B 相时钟, I00 上升沿先于 I01 上升沿 90 度到来, 为增计数, 4 倍速模式	
I00 + I01	AB 相	减	0xFF05	I00 为 A 相时钟, I01 为 B 相时钟, I00 上升沿后于 I01 上升沿 90 度到来, 为减计数, 4 倍速模式	编码器 0
I04+ I05	AB 相	增	0xFF02	I05 为 A 相时钟, I04 为 B 相时钟, I05 上升沿先于 I04 上升沿 90 度到来, 为增计数, 单倍速模式	1
		减		I05 为 A 相时钟, I04 为 B 相时钟, I05 上升沿后于 I04 上升沿 90 度到来, 为减计数, 单倍速模式	
		增	0xFF03	I05 为 A 相时钟, I04 为 B 相时钟, I05 上升沿先于 I04 上升沿 90 度到来, 为增计数, 4 倍速模式	
		减		I05 为 A 相时钟, I04 为 B 相时钟, I05 上升沿后于 I04 上升沿 90 度到来, 为减计数, 4 倍速模式	
I06+ I07	AB 相	增	0xFF00	I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿先于 I06 上升沿 90 度到来, 为增计数, 单倍速模式	0
		减		I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿后于 I06 上升沿 90 度到来, 为减计数, 单倍速模式	

		增	0xFF01	I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿先于 I06 上升沿 90 度到来, 为增计数, 4 倍速模式	
		减		I07 为 A 相时钟, I06 为 B 相时钟, I07 上升沿后于 I06 上升沿 90 度到来, 为减计数, 4 倍速模式	
I02+ I03	AB 相	增	0xFF06	I03 为 A 相时钟, I02 为 B 相时钟, I03 上升沿先于 I02 上升沿 90 度到来, 为增计数, 单倍速模式	编码器 1
		减		I03 为 A 相时钟, I02 为 B 相时钟, I03 上升沿后于 I02 上升沿 90 度到来, 为减计数, 单倍速模式	
		增	0xFF07	I03 为 A 相时钟, I02 为 B 相时钟, I03 上升沿先于 I02 上升沿 90 度到来, 为增计数, 4 倍速模式	
		减		I03 为 A 相时钟, I02 为 B 相时钟, I03 上升沿后于 I02 上升沿 90 度到来, 为减计数, 4 倍速模式	

表 6.8.6 运动控制器及 MAC1610 高速计数模式一览表

引脚	MODE (16 进制)	模式	描述
HI0	0x0000	单相增计数模式	HI0 增计数
	0x0001	单相减计数模式	HI0 减计数
HI1	0x0002	单相增计数模式	HI1 增计数
	0x0003	单相减计数模式	HI1 减计数
HI2	0x0004	单相增计数模式	HI2 增计数
	0x0005	单相减计数模式	HI2 减计数
HI3	0x0006	单相增计数模式	HI3 增计数
	0x0007	单相减计数模式	HI3 减计数
HI4	0x0008	单相增计数模式	HI4 增计数
	0x0009	单相减计数模式	HI4 减计数
HI5	0x000A	单相增计数模式	HI5 增计数
	0x000B	单相减计数模式	HI5 减计数
HI6	0x000C	单相增计数模式	HI6 增计数
	0x000D	单相减计数模式	HI6 减计数
HI7	0x000E	单相增计数模式	HI7 增计数
	0x000F	单相减计数模式	HI7 减计数
HI0+HI1	0x0100	脉冲+方向模式	HI0 为脉冲, HI1 为方向; HI1 为 0 增计数, HI1 为 1 减计数
HI2+HI3	0x0101	脉冲+方向模式	HI2 为脉冲, HI3 为方向; HI3 为 0 增计数, HI3 为 1 减计数
HI4+HI5	0x0102	脉冲+方向模式	HI4 为脉冲, HI5 为方向; HI5 为 0 增计数, HI5 为 1 减计数
HI6+HI7	0x0103	脉冲+方向模式	HI6 为脉冲, HI7 为方向; HI7 为 0 增计数, HI7 为 1 减计数
HI0+HI1	0x0200	AB 相模式	HI0 为 A 相, HI1 为 B 相, 相位差 90°, A 相超前 B 相增计数, 否则减计数, 4 倍频模式
HI2+HI3	0x0201	AB 相模式	HI2 为 A 相, HI3 为 B 相, 相位差 90°, A 相超前 B 相增计数, 否则减计数, 4 倍频模式
HI4+HI5	0x0202	AB 相模式	HI4 为 A 相, HI5 为 B 相, 相位差 90°, A 相超前 B 相增计数, 否则减计数, 4 倍频模式
HI6+HI7	0x0203	AB 相模式	HI6 为 A 相, HI7 为 B 相, 相位差 90°, A 相超前 B 相增计数, 否则减计数, 4 倍频模式
HI0+HI1	0x0300	正向+反向脉冲模式	HI0 为正向, HI1 为反向, 正向增计数, 反向减计数

HI2+HI3	0x0301	正向+反向脉冲模式	HI2 为正向, HI3 为反向, 正向增计数, 反向减计数
HI4+HI5	0x0302	正向+反相脉冲模式	HI4 为正向, HI5 为反向, 正向增计数, 反向减计数
HI6+HI7	0x0303	正向+反向脉冲模式	HI6 为正向, HI7 为反向, 正向增计数, 反向减计数

表 6.8.7 PEC8000/PMC2000/PEC3000/DIO3000 高速计数对应 SM 区标志位

计数器	SM 区	描述
0	SM63.0	计数到达预定值则该位置 1, 高速计数复位时, 该位自动清零
	SM63.1	复位标志位, 当该位为 1, 计数值恢复为初始值, 频率清零, 当该位为 0, 则持续计数
	SM63.2	计数等于预定值则该位置 1、 不等于时该位清零
	SM63.3	计数大于预定值则该位置 1, 小于等于时该位清零
1	SM63.4	计数到达预定值则该位置 1, 高速计数复位时, 该位自动清零
	SM63.5	复位标志位, 当该位为 1, 计数值恢复为初始值, 频率清零, 当该位为 0, 则持续计数
	SM63.6	计数等于预定值则该位置 1, 不等于时该位清零
	SM63.7	计数大于预定值则该位置 1, 小于等于时该位清零
2	SM63.14	计数大于预定值则该位置 1, 小于等于时该位清零, 复位标志置位时不清零
	SM63.15	复位标志位, 当该位为 1, 计数值恢复为初始值, 频率清零, 当该位为 0, 则持续计数
3	SM63.10	计数大于预定值则该位置 1, 小于等于时该位清零, 复位标志置位时不清零
	SM63.11	复位标志位, 当该位为 1, 计数值恢复为初始值, 频率清零, 当该位为 0, 则持续计数
所有计数器	SM63.8	AB 相计数时方向控制位, 默认为 0, 置位 1 后, A 相和 B 相位置互换, 对所有的 AB 相模式同时有效。
	SMW187	高速计数测周法和测频法分割点, 默认为 1Hz, 对高速计数器 0 和 1 有效
	SMW188	高速计数的滤波次数, 默认为 10 次滤波, 最大为 50 次, 最小为 4 次, 对高速计数器 0 有效
	SMW189	高速计数的滤波次数, 默认为 10 次滤波, 最大为 50 次, 最小为 4 次, 对高速计数器 1 有效

表 6.8.8 EDC8000 高速计数对应 SM 区标志位

计数器	SM 区	描述
2	SM63.2	计数等于预定值则该位置 1、 不等于时该位清零
	SM63.3	增计数模式 $CV \geq PV$ 置 1, 小于等于时该位清零; 减计数模式 $CV \leq PV$ 置 1, 大于等于时该位清零;
0	SM63.6	计数等于预定值则该位置 1、 不等于时该位清零
	SM63.7	增计数模式 $CV \geq PV$ 置 1, 小于等于时该位清零; 减计数模式 $CV \leq PV$ 置 1, 大于等于时该位清零;
1	SM63.10	计数等于预定值则该位置 1、 不等于时该位清零
	SM63.11	增计数模式 $CV \geq PV$ 置 1, 小于等于时该位清零; 减计数模式 $CV \leq PV$ 置 1, 大于等于时该位清零;
3	SM63.14	计数等于预定值则该位置 1、 不等于时该位清零
	SM63.15	增计数模式 $CV \geq PV$ 置 1, 小于等于时该位清零; 减计数模式 $CV \leq PV$ 置 1, 大于等于时该位清零;

表 6.8.9 高速计数分钟计频方式

物理输入点		SM 区	描述
PEC8000/PMC2000	PEC3000/DIO3000		

DI07	DI22	SM35.0	该位置 1，高速计数器输出频率为分钟频率
DI06	DI23	SM35.1	该位置 1，高速计数器输出频率为分钟频率
DI05	DI20	SM35.2	该位置 1，高速计数器输出频率为分钟频率
DI04	DI21	SM35.3	该位置 1，高速计数器输出频率为分钟频率
——	DI00	SM35.4	该位置 1，高速计数器输出频率为分钟频率
——	DI01	SM35.5	该位置 1，高速计数器输出频率为分钟频率
——	DI02	SM5.6	该位置 1，高速计数器输出频率为分钟频率
——	DI03	SM35.7	该位置 1，高速计数器输出频率为分钟频率
——	DI04	SM35.8	该位置 1，高速计数器输出频率为分钟频率
——	DI05	SM35.9	该位置 1，高速计数器输出频率为分钟频率
——	DI06	SM35.10	该位置 1，高速计数器输出频率为分钟频率
——	DI07	SM35.11	该位置 1，高速计数器输出频率为分钟频率
——	DI12	SM35.12	该位置 1，高速计数器输出频率为分钟频率
——	DI13	SM35.13	该位置 1，高速计数器输出频率为分钟频率
——	DI14	SM35.14	该位置 1，高速计数器输出频率为分钟频率
——	DI15	SM35.15	该位置 1，高速计数器输出频率为分钟频率

● 应用示例

(1) 内部控制单向计数

内部控制单向计数是根据 HDEF 功能块的模式设置 HDEF 功能块的增减计数功能，如图 6.8.3 所示，以 PEC8000 为例，模式为 0 为 DI7 的内部控制单向增计数，模式 1 为内部控制单向减计数。详细的模式与计数引脚参见《高速模式一览表》。

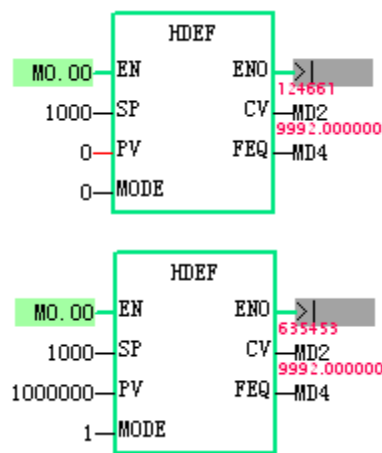


图 6.8.3 内部控制单向计数例程

(2) 外部控制单向计数

外部控制单向计数是 HDEF 功能块的增减计数是根据一个外部输入的高低电平决定的，如图 6.8.4 所示，以 PEC8000 为例，当模式设置为 0x0400 时，DI5 输入低电平时 HDEF 为增计数，DI5 输入高电平时 HDEF 为减计数。详细的模式与计数引脚参见《高速模式一览表》。

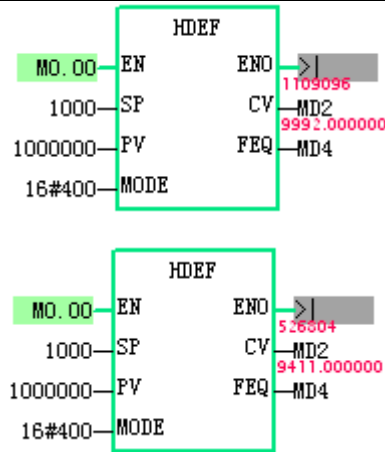


图 6.8.4 外部控制单向计数例程

(3) AB 相计数模式

AB 相计数是根据 A 相与 B 相的相位差计数，如图 6.8.5 所示，以 PEC8000 为例，模式为 0xFF03 时，I05 为 A 相时钟，I07 为 B 相时钟，I05 上升沿先于 I07 上升沿 90 度到来，为增计数。I05 为 A 相时钟，I07 为 B 相时钟，I05 上升沿后于 I07 上升沿 90 度到来，为减计数。详细的模式与计数引脚参见《高速模式一览表》。

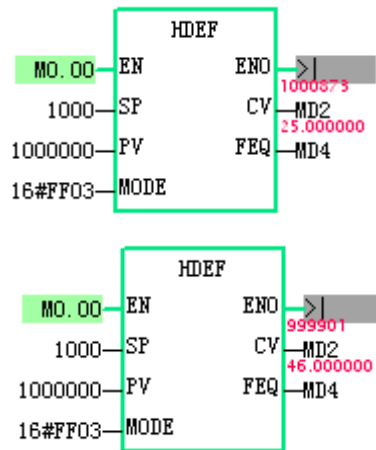
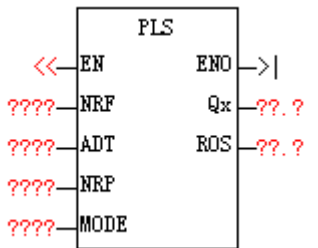


图 6.8.5 AB 相计数例程

6.8.3 高速脉冲输出指令

● 指令参数

高速脉冲输出指令(PLS)			
 实现不同频率的高速脉冲输出功能;支持十级变速输出模式、PWM 占空比可调输出模式和 S 曲线输出模式。可通过 MODE 引脚选择高速输出模式。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
NRF	双字变量、常数	DWORD	正常输出频率, 单位为 Hz
ADT	双字变量、常数	DWORD	十级变速输出模式时为输出加减速时间, 单位为 ms; PWM 占空比可调输出模式时为占空比, 0~100 代表 0%~100%; S 曲线输出模式时为加减速脉冲个数; NRP: 每轮输出的脉冲个数; 只有十级变速输出模式和 S 曲线输出模式有效
NRP	双字变量、常数	DWORD	每轮输出的脉冲个数; 只有十级变速输出模式和 S 曲线输出模式有效
MODE	字变量、常数	WORD	模式选择。低字节为方式选择, 0 表示方式十级变速输出模式, 1 表示方式 PWM 占空比可调输出模式, 2 表示方式 S 曲线输出模式, 其余保留; 高字节为触发方式选择, 0 为电平触发, 1 为上升沿触发输出, 其余保留; PWM 占空比可调输出模式不支持上升沿触发
Qx	Q 区	BOOL	高速输出的物理引脚
ROS	位变量	BOOL	输出完毕标志位
ENO	位变量	BOOL	输出能流

注: 运动控制器仅支持 PWM 占空比可调输出模式。

● 详细说明

PLS 指令支持十级变速输出、PWM 占空比可调输出和 S 曲线输出, 共 3 种模式。十级变速和 S 曲线输出模式均为占空比为 50% 的脉冲串输出方式, 输出过程均分为 3 个阶段: 加速、恒速、减速; PWM 占空比可调输出模式为占空比可调的等频率输出方式。在同一工程中最多可同时使用 2 个 PLS 指令。

➤ 高速输出物理引脚说明

不是所有的控制器都支持高速输出功能, 如表 6.8.10 所示, 列出了支持高速输出的控制器, 并标出了每个控制器的高速输出对应物理输出引脚以及所能达到的频率范围。

表 6.8.10 控制器对应高速输出引脚及频率范围

控制器		高速 DQ 物理引脚	最低频率	最高频率
PEC8000/PMC2000		Q0.6、Q0.7	600Hz	20KHz
PEC3000/DIO3000		Q0.0、Q0.1	600Hz	20KHz
EDC8000		Q0.7、Q0.8	200Hz	200KHz
运动控制器	PEC6000/ MAC1600/ PEC6300/PEC6600	Q0.12~Q0.15	0Hz	500KHz

	PEC6680	Q08~Q15		
	PEC6900	Q20~Q23		
DEC 系列控制器		Q0.0、Q0.1	1Hz	200KHz

➤ 十级变速输出模式

在该模式下，加减速阶段采用十级变速方式，即加速阶段由起始频率经过 10 次加速输出达到目标频率，减速阶段由目标频率经过 10 次减速输出达到起始频率。经过十级变速可使输出的脉冲平缓的上升下降，减小惯性冲击。该模式下，通过相应的物理输出可产生等占空比的方波脉冲。使用时需要注意以下几点：

(1) 分十级变速。NRF(Normal Run Frequency)为正常输出时的频率（最大频率）。为使误差最小，NRF 最好为 10 的整数倍。

(2) ADT(Accelerate Decelerate Time)为加减速时间。此数的大小依赖于 NRF 的设置，此数不宜过小。单位是 ms。

(3) 用户配置时应确保 $N = ADT \times NRF \times 252 / (7.381 \times 10^6) \geq 1$ 。

(4) NRP(Normal Run Pulse)为总脉冲个数。用户配置时，此数也不宜过小，应确保大于 $18 \times N$ ($N = ADT \times NRF \times 252 / (7.381 \times 10^6)$)。

(5) EN 为指令的使能位。EN 为 1 时指令运行，EN 为 0 时功能块停止运行。若触发方式为上升沿触发 (MODE 的高字节为 1) 时，每次 EN 从 0 到 1 的跳变，指令从初始动作开始运行，完成 1 轮脉冲后自动停止，直到下个上升沿到来再次进行脉冲输出。

(6) 支持起始频率设置，其中，SMW156 为第一路高速输出的起始频率，SMW157 为第二路高速输出的起始频率。当用户设置相应的起始频率时，加速过程中变速输出会从相应的起始频率开始而不是从 0 开始增加，减速过程中也会减速到相应的起始频率后停止。默认为 0，单位为 Hz。

(7) 运行状态标志位：

SM5.0: 频率值低于起始频率 ($NRF < 10$)

SM5.1: 频率和时间设置不正确

SM5.2: 脉冲个数设定不正确

(8) 该模式下，对于 PEC8A00、PEC3A00、PEC3A70、EDC8000 无法输出小于 1KHz 的频率。

➤ PWM 占空比可调输出模式

在该模式下，可产生一个占空比、频率可调的脉冲输出，占空比和输出频率可根据参数配置，ADT 参数代表占空比，0~100 代表 0%~100%，该模式只工作在电平触发的方式下，EN 使能后，对应的物理输出引脚将一直输出。

➤ S 曲线输出模式

S 曲线实际上不是一种固定算法的加减速形式，由于其加减速段的速度呈 S 形而得名，常见的 S 曲线有抛物线型和三角程序型，PLS 指令中实现抛物线方式的 S 曲线。S 曲线的核心思想是让加速度不产生突变，而是由零逐渐增大到目标加速度，并在撤销加速度时也采用逐渐减小加速度的方式。图 6.8.6 给出了抛物线 S 曲线的速度、加速度曲线。

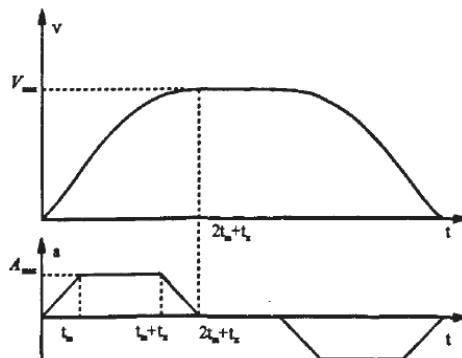


图 6.8.6 抛物线 S 曲线的速度和加速度曲线图

S 曲线的加速段主要分为三个子阶段：加加速阶段、匀加速阶段、减加速阶段。现设 $V_{\max}$ 为最大速度， $A_{\max}$ 为最大加速度， $a1$ 为加加速阶段的加速度， $a2$ 为匀加速阶段的加速度， $a3$ 为减加速阶段的加速度， $v1$ 为加加速阶段的速度， $v2$ 为匀加速阶段的最高速度， $v3$ 为减加速阶段的速度。 t_m 为加速阶段的时间和减加速阶段的时间， t_x 为匀加速阶段的时间和减加速阶段的时间，那么其各段的加速度满足下列等式。

$$a1 = (A_{\max} / t_m) \times t$$

$$a2 = A_{\max}$$

$$a3 = A_{\max} - (A_{\max} / t_m) \times t$$

模块是以脉冲为单位进行曲线计算的，即在加速和减速阶段输出一个脉冲的同时将当前的速度代入 S 曲线方程计算下一个脉冲的速度，通过此方法来完成加速和减速阶段的脉冲输出。脉冲输出时，由于按照 S 曲线方程计算出的起始频率可能会很小，输出这个脉冲会占用很长时间，加大了 S 曲线的加减速时间。为了避免这种情况，在 SM 区开放了变量区，用户可以根据需要设置功能块的起始频率。在实际应用中加加速和减加速段的时间越短越好，而 t_x 越大加减速时间越短，取 $t_x = 2 \times t_m$ ，那么加减速时间 $T_{up} = T_{dn} = t_x + 2 \times t_m = 4 \times t_m$ 。

➤ 参数说明

NRF：常速运行段的脉冲频率。

ADT：加速段和减速段的脉冲个数，此值愈大加减速时间越长，反之亦然。它的设置值必须满足 $2 \times ADT < NRP$ ，否则功能块将不进行指令输出。另外在 SM 区可以查看加减速时间，详见表 6.8.11。

NRP：每一轮输出的脉冲个数。

表 6.8.11 高速脉冲输出 SM 区说明

输出模式	SM 区	描述
十级变速	SM5.0	频率值低于下限频率 (NRF<10)
	SM5.1	频率和时间设置不正确
	SM5.2	脉冲个数设定不正确
	SMW156	高速十级变速输出 Q0.6(PEC8000 和 PMC2000)/Q0.0(PEC3000 和 DIO3000) 起始频率
	SMW157	高速十级变速输出 Q0.7(PEC8000 和 PMC2000)/Q0.1(PEC3000 和 DIO3000) 起始频率
S 曲线	第一路高速输出的 SM 区说明 Q0.7(PEC8000)/Q0.0(PEC3000)	
	SMD20	本轮中已输出的脉冲个数(只读)
	SMW22	S 曲线加/减速时间 T_{up}/T_{dn} ，单位为毫秒(只读)
	SMW23	S 曲线最大加速度 $A_{\max}$ (只读)
	SMW24	S 曲线加减速段的起始频率，默认为 2Hz(读写)
	第二路高速输出的 SM 区说明 Q0.6(PEC8000)/Q0.1(PEC3000)	
	SMD30	本轮中已输出的脉冲个数(只读)
	SMW32	S 曲线加/减速时间 T_{up}/T_{dn} ，单位为毫秒(只读)
	SMW33	S 曲线最大加速度 $A_{\max}$ (只读)
	SMW34	S 曲线加减速段的起始频率，默认为 2Hz(读写)
	SMD30	本轮中已输出的脉冲个数(只读)

● 应用示例

如图 6.8.7 所示，PEC3000 控制器调用 PLS 指令，配置为十级变速输出模式，通过高速物理输出引脚 Q00 输出频率为 3000Hz 的脉冲，将 Q00 接到高速脉冲输入 I22 上，利用高速计数指令采集当前的输出频率和脉冲个数。

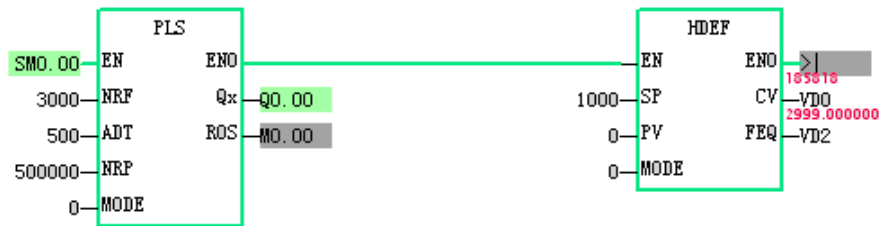


图 6.8.7 高速输出指令例程

运行后，高速脉冲输出 3000Hz 脉冲，Q0.0 有脉冲输出，接入到 I22 后，由高速计数 1 工作在模式 0 下进行计数，显示频率为 2999.000000Hz，在计数值达到 1000 时 SM63.0 置位。图 6.8.8 为 Q0.0 输出波形。

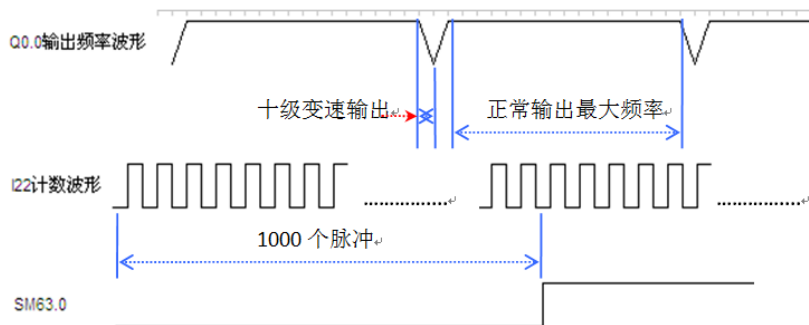


图 6.8.8 高速脉冲输入输出波形图

6.9 实数运算指令

实数运算指令主要用于实现逻辑控制中的浮点数运算功能，包括实数的加减乘除、对数、指数、开方、绝对值、三角函数和反三角函数等数学运算功能。

6.9.1 实数加减乘除

● 指令参数

实数加指令 (ADD_R)	实现实数之间的加法运算；对两个 32 位实数进行加法运算得到 32 位的结果，放入 OUT 中。 $IN1+IN2=OUT$
实数减指令 (SUB_R)	实现实数之间的减法运算；对两个 32 位实数进行减法运算得到 32 位的结果，放入 OUT 中。 $IN1-IN2=OUT$
实数乘指令 (MUL_R)	实现实数之间的乘法运算；对两个 32 位实数进行乘法运算得到 32 位的结果，放入 OUT 中。 $IN1*IN2=OUT$
实数除指令 (DIV_R)	实现实数之间的除法运算；对两个 32 位实数进

			行除运算得到 32 位的结果，放入 OUT 中。 $IN1/IN2=OUT$
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN1/IN2	双字变量、常数	REAL	要输入的变量的值
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量运算的值

- 详细说明

实数加减乘除指令主要用于实现简单的数学运算，需要注意根据 32 位实数的规定，实数的有效位数为 6~7 位，所以在实数操作时请避免将一个很大的数和很小的数相加或相减，这样有可能会将小的数丢失。

- 应用示例

如图 6.9.1 所示，求代数式 $[a \times b + (c - d)] / 2$ 的值，根据例子中给出的程序计算 $[300 \times 500 + (600 - 200)] / 2$ ，结果为 75200。

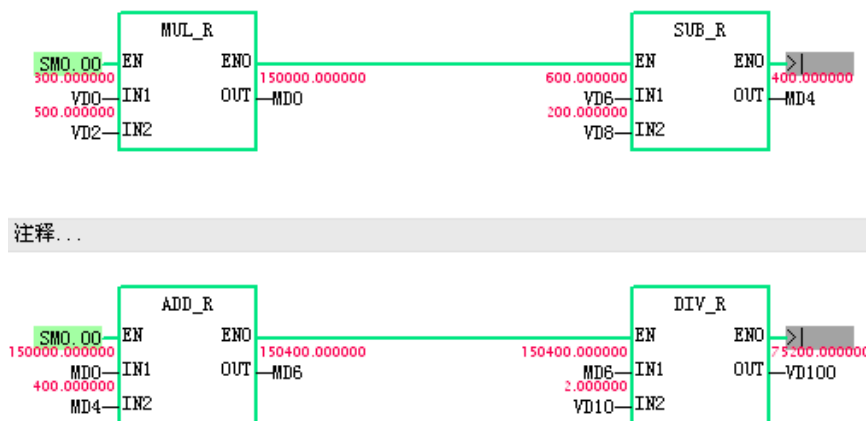


图 6.9.1 实数加减乘除指令例程

6.9.2 数学功能

6.9.2.1 开方指令 (SQRT)

- 指令参数

开方指令			
			实现开方运算功能，对 32 位实数进行开方运算得到 32 位的结果。 $OUT = \sqrt{IN}$
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量开方的值

- 详细说明

开方指令主要用于实现数学运算中的开方功能，需要注意该指令的输入变量只能是正实数或 0，若输入变量是负实数则输出为 0，并将 SM0.11（运算错误状态标志位）置 1，表示当前运算错误。

● 应用示例

如图 6.9.2 所示，例程计算了 10000 的平方根。

运算过程： $VD1 = \sqrt{10000} = 100.000000$

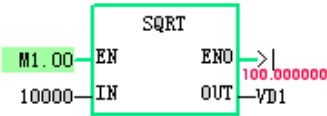


图 6.9.2 开方指令例程

6.9.2.2 正弦指令（SIN）

● 指令参数

正弦指令		实现正弦运算功能；对 32 位实数进行正弦运算得到 32 位的结果。 OUT = SIN(IN)	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值，单位“rad”
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量的值 -1≤OUT≤1

● 详细说明

正弦指令主要用于实现三角函数中的正弦运算功能，根据三角函数运算法则，可得出该指令的输入变量是全体实数，输出变量在-1 到 1 之间。使用时需要注意输入变量以弧度为单位。

● 应用示例

如图 6.9.3 所示，例程完成了 30°、45°、90°的正弦值的计算。例程中，对使能端 EN 进行了取反操作，即下载后立即使能指令。

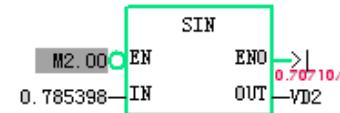
30°≈0.523599（弧度）；sin30°≈0.500000

45°≈0.785398（弧度）；sin45°≈0.707107

90°≈1.570796（弧度）；sin90°≈1.000000



用户注释



用户注释

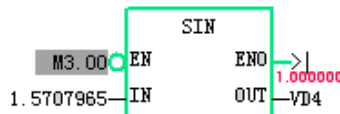


图 6.9.3 正弦指令例程

6.9.2.3 反正弦指令(ARC_SIN)

- 指令参数

反正弦指令 (ARC_SIN)			
		实现反正弦运算功能，对 32 位实数进行反正弦运算得到 32 位的结果。 $OUT = \text{ARCSIN}(IN)$	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值 $-1 \leq IN \leq 1$
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量的值，单位“rad”， $-\pi/2 \leq OUT \leq +\pi/2$

- 详细说明

反正弦指令主要用于实现三角函数中的反正弦运算功能，该指令的输出结果是以弧度制表示的角度，输出范围在正负 $\pi/2$ 以内（其中 $\pi = 3.1414\dots$ ）。需要注意该指令的输入变量数据范围是： $-1 \leq IN \leq 1$ ，若输入变量在允许范围以外，则输出为 0，并将 SM0.11（运算错误状态标志位）置 1，表示当前运算错误。

- 应用示例

如图 6.9.4 所示，例程完成 $\arcsin 30^\circ$ 、 $\arcsin 45^\circ$ 、 $\arcsin 90^\circ$ 的计算。

$\sin 30^\circ \approx 0.500000$ ； $\arcsin 30^\circ \approx 0.523599$ （弧度）

$\sin 45^\circ \approx 0.707107$ ； $\arcsin 45^\circ \approx 0.785398$ （弧度）

$\sin 90^\circ \approx 1.000000$ ； $\arcsin 90^\circ \approx 1.570796$ （弧度）

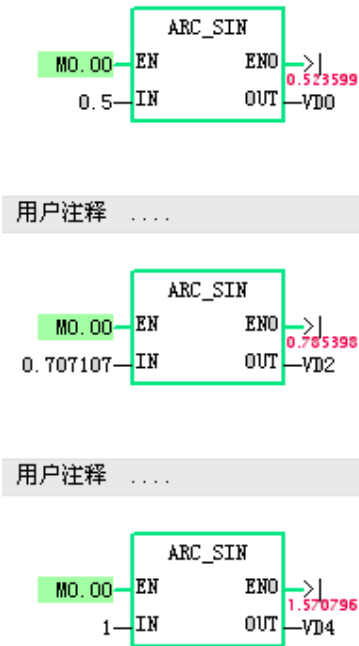


图 6.9.4 反正弦指令例程

6.9.2.4 余弦指令 (COS)

● 指令参数

余弦指令			
		实现余弦运算功能；对 32 位实数进行余弦运算得到 32 位的结果。 OUT=COS(IN)	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值，单位 “rad”
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量运算的值

● 详细说明

余弦指令主要用于实现三角函数中的余弦运算功能，根据三角函数运算法则，可得出该指令的输入变量是全体实数，输出变量在-1 到 1 之间。使用时需要注意输入变量以弧度为单位。

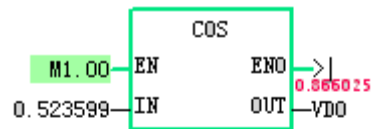
● 应用示例

如图 6.9.5 所示，例程完成 30°、45°、90°的余弦值的计算。

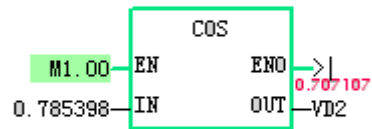
30°≈0.523599（弧度）；cos30°≈0.866025

45°≈0.785398（弧度）；cos45°≈0.707107

90°≈1.570796（弧度）；cos90°≈0.000000



用户注释



用户注释

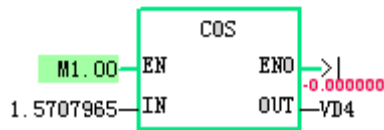
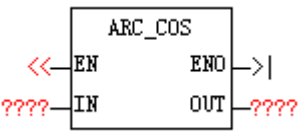


图 6.9.5 余弦指令例程

6.9.2.5 反余弦指令 (ARC_COS)

● 指令参数

反余弦指令 (ARC_COS)			
		实现三角程序反余弦的运算：	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	输入变量的值 $-1 \leq IN \leq 1$
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量的值 单位 “rad”， $0 \leq OUT \leq +\pi$

● 详细说明

反余弦指令主要用于实现三角函数中的反余弦运算功能，该指令的输出结果是以弧度制表示的角度，输出范围在 0 到 π 之间（其中 $\pi = 3.1414\dots$ ）。需要注意该指令的输入变量数据范围是： $-1 \leq IN \leq 1$ ，若输入变量在允许范围以外，则输出为 0，并将 SM0.11（运算错误状态标志位）置 1，表示当前运算错误。

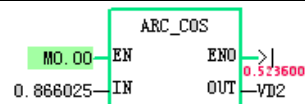
● 应用示例

如图 6.9.6 所示，例程完成 $\arccos 30^\circ$ 、 $\arccos 45^\circ$ 、 $\arccos 90^\circ$ 的值计算。

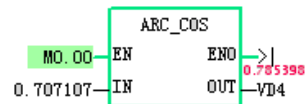
$\cos 30^\circ \approx 0.866025$ ； $\arccos 30^\circ \approx 0.523599$ （弧度）

$\cos 45^\circ \approx 0.707107$ ； $\arccos 45^\circ \approx 0.785398$ （弧度）

$\cos 90^\circ \approx 0.000000$ ； $\arccos 90^\circ \approx 1.570796$ （弧度）



用户注释



用户注释

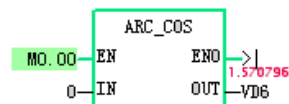


图 6.9.6 反余弦指令例程

6.9.2.6 正切指令 (TAN)

● 指令参数

正切指令 (TAN)			
		实现三角程序正切的运算；对 32 位实数进行正切运算得到 32 位的结果。 $OUT = \tan(IN)$	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值 $\neq (2k+1)\pi/2$ ($k = 0, 1, 2 \dots$)
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量的值

● 详细说明

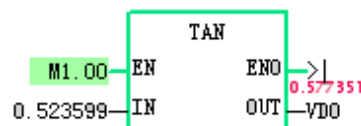
正切指令主要用于实现三角函数中的正切运算功能，根据三角函数运算法则，可得出该指令的输入变量的余弦值不能为 0。使用时需要注意输入变量以弧度为单位。若输入变量在允许范围以外，则输出为 0，并将 SM0.11（运算错误状态标志位）置 1，表示当前运算错误。

● 应用示例

如图 6.9.7 所示，例程完成 30°、45° 正切值的计算。

30° ≈ 0.523599 (弧度)；tan30° ≈ 0.577351

45° ≈ 0.785398 (弧度)；tan45° ≈ 1.000000



用户注释

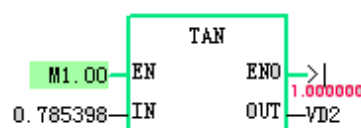
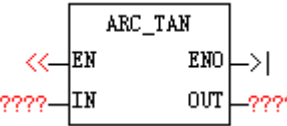


图 6.9.7 正切指令例程

6.9.2.7 反正切指令 (ARC_TAN)

● 指令参数

反正切指令 (ARC_TAN)		实现三角程序反正切的运算；对 32 位实数进行反正切运算得到 32 位的结果。 OUT=ARCTAN(IN)	
			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值
OUT	双字变量	REAL	输出变量的值 $-\pi/2 < OUT < +\pi/2$
ENO	位变量	BOOL	输出能流

● 详细说明

反正切指令主要用于实现三角函数中的反正切运算功能，该指令的输出结果是以弧度制表示的角度，输出范围在 $-\pi/2$ 到 $+\pi/2$ 之间（不包括 $\pm\pi/2$ ）。

● 应用示例

如图 6.9.8 所示，例程完成 $\arctan 30^\circ$ 、 $\arctan 45^\circ$ 的计算。

$\tan 30^\circ \approx 0.577351$ ； $\arctan 30^\circ \approx 0.523599$ （弧度）

$\tan 45^\circ \approx 1.000000$ ； $\arctan 45^\circ \approx 0.785398$ （弧度）

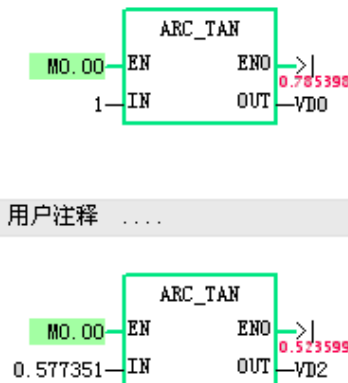
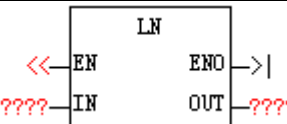


图 6.9.8 反切指令例程

6.9.2.8 对数指令 (Ln)

● 指令参数

对数指令 (Ln)		实现以 e 为底数的对数的运算；对 32 位实数进行自然对数运算得到 32 位的结果。 OUT=ln(IN)	
			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能流
IN	双字变量、常数	REAL	要输入变量的值
ENO	位变量	BOOL	输出能流
OUT	双字变量	REAL	输出变量运算的值

● 详细说明

对数指令主要用于实现数学中的对数运算功能，根据对数运算法则，可得出该指令的输入变量必须大于 0。若输入变量在允许范围以外，则输出为 0，并将 SM0.11（运算错误状态标志位）置 1，表示当前运算错误。

- 应用示例

如图 6.9.9 所示，例程完成 1、10 的自然对数的计算。

计算结果：

$\ln 1 = 0.000000$

$\ln 10 = 2.302585$

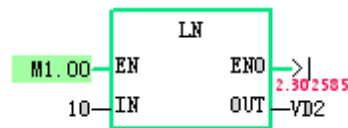
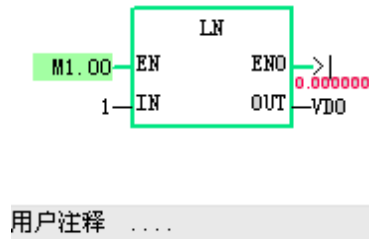


图 6.9.9 对数指令例程

6.9.2.9 幂指数指令 (EXP)

- 指令参数

幂指数指令 (EXP)			
		实现以 E 为底数的幂指数的运算；对 32 位实数进行自然幂指数运算得到 32 位的结果。 $OUT = e^{IN}$	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	REAL	要输入变量的值 $-89 \leq IN \leq 89$
ENO	位变量	BOOL	输出能量流
OUT	双字变量	REAL	输出变量运算的值

- 详细说明

幂指数指令主要用于实现数学中的幂指数运算功能，由于输出变量是 32 位实数，故当输入变量过大时会导致输出溢出，所以输入变量的数据范围限定为： $-89 \leq IN \leq 89$ 。若输入变量在允许范围以外，则输出为 0，并将 SM0.11（运算错误状态标志位）置 1，表示当前运算错误。

- 应用示例

如图 6.9.10 所示，例程计算 1、10 的幂指数。计算结果： $OUT = e^1 = 2.718282$ ； $OUT = e^{10} = 22026.464844$ 。

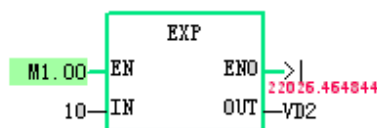
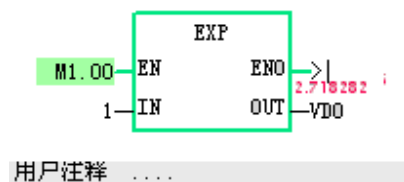
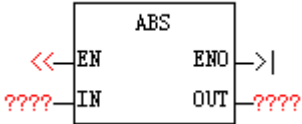


图 6.9.10 幂指数指令例程

6.9.2.10 绝对值指令 (ABS)

- 指令参数

绝对值指令 (ABS)			可以完成一个实数的绝对值运算: 对 32 位实数进行自然幂指数运算得到 32 位的结果。 $OUT= IN$
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	REAL	要输入变量的值
ENO	位变量	BOOL	输出能量流
OUT	双字变量	REAL	输出变量运算的值

● 详细说明

绝对值指令主要用于实现数学中的绝对值运算功能, 根据绝对值运算法则, 可得出该指令的输入变量为全体实数。输出变量为非负实数。

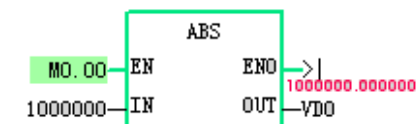
● 应用示例

如图 6.9.11 所示, 例程计算 1000000、0、-1000000 的绝对值。

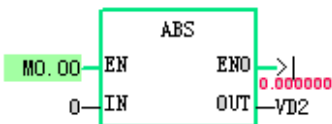
计算结果: $ABS(1000000) = 1000000$;

计算结果: $ABS(0) = 0$;

计算结果: $ABS(-1000000) = 1000000$;



用户注释



用户注释

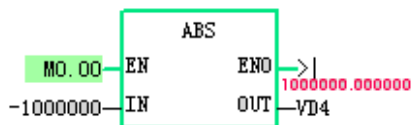


图 6.9.11 绝对值指令例程

6.9.3 比例积分微分 (PID)

可编程控制器具有带自整定功能的 PID 控制指令, 可以根据被控对象自动生成控制参数; 提供多种 PID 控制方式的选择, 为用户提供更为专业的控制方案; 实现精确的温度控制, 控制精度可达 $\pm 0.1^\circ\text{C}$ 。

每个控制器最多支持 16 个 PID 指令。按照控温方式分类, PID 指令可分为增量式 PID 指令和位置式 PID 指令, 其中位置式 PID 指令的 0~7 号 PID 支持控制参数自整定功能, 自整定方式分为继电器反馈自整定 (有超调) 和飞升曲线自整定 (无超调) 两种方式。

PID 控制参数, 包括比例系数、积分系数和微分系数。在不同系统 (温度系统、流量系统、压力系统等) 中参数不同, 其参数可根据以下三种方法获得:

① 先设置为不同系统的经验值, 然后根据调节效果修改。

②建立系统的数学模型，用仿真的方式计算出各参数的数值，再用于实际中。

③利用控制器提供的 PID 参数自整定功能，可以根据现场对象自整定出一套参数，供 PID 指令使用。

6.9.3.1 位置式 PID 指令

● 指令参数

位置式 PID 指令 (P_PID)			
根据配置的参数进行位置式 PID 控制。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	PID 运算的使能控制位，该位为 1，PID 开始运算，该位为零，PID 停止运算，PID 所有输出清零
SP	双字变量、常数	REAL	运算设定值，即目标设定值
PB	双字变量、常数	REAL	比例度
TI	双字变量、常数	REAL	积分项，单位为秒
TD	双字变量、常数	REAL	微分项，单位为秒
TS	双字变量、常数	REAL	采样时间，单位为秒
AREA	双字变量、常数	REAL	起控范围，单位和 SP 相同
LOOP	常数	WORD	回路号，0~15，最多支持 16 个 PID 指令
LOW_I	双字变量、常数	REAL	PID 模拟输入的零点
HIGH_I	双字变量、常数	REAL	PID 模拟输入的满度
LOW_O	双字变量、常数	REAL	PID 模拟输出的零点
HIGH_O	双字变量、常数	REAL	PID 模拟输出的满度
IN	双字变量	REAL	PID 的模拟输入值
ENO	位变量	BOOL	能流传输位
PWM_O	位变量	BOOL	作为 PWM 输出时的数字量输出位
PID_O	双字变量	REAL	作为模拟输出时是模拟输出量，作为 PWM 输出时是高电平持续时间，单位是 ms

● 详细说明

位置式 PID 控制的输出与整个过去的状态有关，用到了误差的累加值，位置式的输出直接对应对象的输出，适用于执行机构不带积分部件的对象，如电液伺服阀等。位置式 PID 的 0~7 路支持自整定算法，自整定方式共支持继电反馈自整定（有超调）和飞升曲线自整定（无超调）两种方式。

6.9.3.2 增量式 PID 指令

● 指令参数

增量式 PID 指令 (I_PID)	根据配置的参数进行增量式 PID 控制。

参数名	变量区	数据类型	说明
EN	位变量	BOOL	PID 运算的使能控制位，该位为 1，PID 开始运算，该位为零，PID 停止运算，PID 所有输出清零
SP	双字变量、常数	REAL	运算设定值，即目标设定值
PB	双字变量、常数	REAL	比例度
TI	双字变量、常数	REAL	积分项，单位为秒
TD	双字变量、常数	REAL	微分项，单位为秒
TS	双字变量、常数	REAL	采样时间，单位为秒
AREA	双字变量、常数	REAL	起控范围，单位和 SP 相同
LOOP	常数	WORD	回路号，0~15，最多支持 16 路
LOW_I	双字变量、常数	REAL	PID 模拟输入的零点
HIGH_I	双字变量、常数	REAL	PID 模拟输入的满度
LOW_O	双字变量、常数	REAL	PID 模拟输出的零点
HIGH_O	双字变量、常数	REAL	PID 模拟输出的满度
IN	双字变量	REAL	PID 的模拟输入值
ENO	位变量	BOOL	能流传输位
PWM_O	位变量	BOOL	作为 PWM 输出时的数字量输出位
PID_O	双字变量	REAL	作为模拟输出时是模拟输出量，作为 PWM 输出时是高电平持续时间，单位是 ms

● 详细说明

增量式 PID 在实际应用中不如位置式 PID 广泛，主要是当执行机构（如步进电机）无法承受输出量的突变时使用。其输出只与当前拍和前两拍的误差有关，控制输出的是控制量增量，并无积分作用，因此该方法适用于执行机构带积分部件的对象，如步进电机等。由于增量式 PID 每次输出只是需要的增量而不是位置量的绝对数值，避免了位置式 PID 输出会剧烈波动的问题，同时也从根本上避免了积分饱和问题。

➤ PID 基本原理

PID 控制是连续系统中技术最成熟、应用最广泛的一种调节方式。PID 的实质是根据输入的偏差值，按比例、积分、微分的函数关系进行运算，其运算结果用于输出进行控制。其模拟系统中的表达式如下：

$$U(t) = K_p \left[e(t) + \frac{1}{T_i} \int e(t) dt + T_d \frac{de(t)}{dt} \right]$$

其中 $U(t)$ ：输出信号

$e(t)$ ：偏差信号，等于测量值与设定值之差

K_p ：比例系数

T_i ：积分时间

T_d : 微分时间

(1) 比例作用对控制性能的影响

比例增益 K_p 的引入是为了及时地反映控制系统的偏差信号，一旦系统出现了偏差，比例调节作用立即产生调节作用，使系统偏差快速向减小的趋势变化。当比例增益增大的时候，PID 控制器可以加快调节，但是过大的比例增益会使调节过程出现较大的超调量，从而降低系统的稳定性，在某些严重的情况下，甚至可能造成系统不稳定。

(2) 积分作用对控制性能的影响

积分作用的引入是为了使系统消除稳态误差，提高系统的无差度，以保证实现对设定值的无静差跟踪。假设系统已经处于闭环稳定状态，此时的系统输出和误差量保持为常值 U_0 和 e_0 ，则由公式可知，只有当且仅当动态误差 $e(t) = 0$ 时，控制器的输出才为常数。因此，从原理上看，只要控制系统存在动态误差，积分调节就产生作用，直至无差，积分作用就停止，此时积分调节输出为常值。积分作用的强弱取决于积分时间常数 T_i 的大小， T_i 越小，积分作用越强，反之则积分作用弱。积分作用的引入会使系统稳定性下降，动态响应变慢。

(3) 微分作用对控制性能的影响

微分作用的引入，主要是为了改善控制系统的响应速度和稳定性。微分作用能反映系统偏差的变化律，预见偏差变化的趋势，因此能产生超前的控制作用。直观而言，微分作用能在偏差还没有形成之前，就已经消除偏差。因此，微分作用可以改善系统的动态性能。微分作用的强弱取决于微分时间 T_d 的大小， T_d 越大，微分作用越强，反之则越弱。在微分作用合适的情况下，系统的超调量和调节时间可以被有效的减小。从滤波器的角度看，微分作用相当于一个高通滤波器，因此它对噪声干扰有放大作用，而这是我们在设计控制系统时不希望看到的。所以我们不能过强地增加微分调节，否则会对控制系统抗干扰产生不利的影响。此外，微分作用反映的是变化率，当偏差没有变化时，微分作用的输出为零。

➤ PID 指令参数说明

为了下面方便位置式 PID 实现描述，先对一部分参数进行一下说明：

PB——比例度系数，按照通常理解设置成了比例度，即比例系数 $K_p = 100 / PB$ ，对应公式中的 $100 / K_p$ ，

TI——积分常数

TD——微分常数

TS——采样周期

SP——设定值

IN——输入值，即被控对象的测量值

LOW_{in}——输入最低值

HIGH_{in}——输入最高值

LOW_{out}——输出最低值

HIGH_{out}——输出最高值

控制器最多支持 16 路 PID。PID 路数选择是由指令的回路号 (LOOP) 决定的，所以 LOOP 的数据范围在 0~15 之间，若超出此范围指令将不工作。

PB、TI、TD 三个参数分别是 PID 运算的比例、积分、微分项。其中 PB 作为比例度，是比例的倒数，即比例 $K_p = 100 / PB$ 。所以 PB 越大，PID 调节越缓慢。

SP 为控制设定值，PID 被设计成反作用控制方式，如果 PID 的 IN 不等于 SP，PID 指令会根据算法调节 PID_O 来反作用于控制对象，从而调节 IN 的值，直到 IN=SP 为止。SP 与 IN 要为相同量纲的数据值。

TS 是 PID 的采样时间，即 PID 多长时间进行一次采样运算。对于本身具有模拟量输入的控制器，当 TS 设置为 0 时内部自动同步采样周期，使计算采样时间和 AD 转换周期保持一致。

AREA 是起控范围，即 PID 的 IN 在 SP-AREA 和 SP+AREA 范围内进行 PID 调节；如果 IN 小于 SP-AREA，置 PID 输出为最高输出值 HIGH_O，即全功率输出；如果 IN 大于 SP+AREA，置 PID 输出为 PID_O 最低输出值 LOW_O。SMW39 作为起控范围控制变量区，SMW39 的每个位对应一路 PID，相应位为 1 则禁止对应

编号 PID 的起控范围功能，为 0 则使能对应编号 PID 的起控范围功能，默认为 0。

LOW_I、HIGH_I 作为输入的最小、最大值，即 PID 的 IN 能够达到的范围。

LOW_O、HIGH_O 作为输出的最小、最大值，即 PID 的输出能够达到的范围，例如，控制器的模拟输出码值是 0~5000 对应 0~20mA 电流输出。使用模拟量输出作为 PID 的输出控制信号时，可用一个浮点变量（例如 MD100）记录 PID 输出值，然后用 R_I 指令将该变量转换成有符号整数付给 AQ 区，进行输出控制。这时 PID 的输出范围应定为 LOW_O=0，HIGH_O=5000。

IN 是 PID 的模拟量输入，该输入变量类型为实数，故控制器的模拟量输入需要经过转换后输入，并且该输入的采集对象是由 PID 的输出信号控制的，从而完成 PID 运算控制。

PWM_O 是当 PID 作为 PWM 控制输出时的输出位选择，该位只能是 Q0.0~Q0.15 中的一个位。通过 PID 运算可得出输出高电平时间，则 PID 会根据指定的输出周期检测输出信号的高低电平变化，可通过 SMW107 配置输出周期，如表 6.9.1 所示。当使用模拟量输出时，无需配置该引脚。

PID_O 在不同的输出模式下作用不同，作为模拟输出时是模拟输出量，在 PWM 控制输出模式下，PID_O 作为 PWM 输出的高电平持续时间了；在模拟量输出模式下，该值作为模拟量输出值，转换后付给对应的 AQ 区。

提示：

①无模拟量输入的产品由于内部无 AD 转换，不支持 TS 设置为 0。

②当使用 PWM 输出时，输出周期根据 SMW107 配置的模式选择 TS 或 HIGH_O，请在使用时注意区分。

③若选择使用模拟量输出，PWM_O 引脚不可填写输出变量区。

➤ PWM 控制输出模式

PID 指令的 PWM 控制输出模式可通过 SMW107 进行配置。模式配置如表 6.9.1 所示。

表 6.9.1 PWM 控制输出模式配置表

变量区	功能	描述
SMW107	PID 为 PWM 输出时的模式控制寄存器，默认为 0	0: PID 控制 100ms 检查一次 DQ 输出，输出周期为采样时间 TS，最小值为由 LOW_O 变为 0，最大值由 HIGH_O 变为 TS 1: PID 控制 10ms 检查一次 DQ 输出，输出周期为采样时间 TS，最小值为由 LOW_O 变为 0，最大值由 HIGH_O 变为 TS 2: PID 控制 100ms 检查一次 DQ 输出，输出周期为采样时间 TS，最小值为 LOW_O，最大值为 HIGH_O 3: PID 控制 10ms 检查一次 DQ 输出，输出周期为采样时间 TS，最小值为 LOW_O，最大值为 HIGH_O 4: PID 控制 1ms 检查一次 DQ 输出，输出周期为 HIGH_O，最小值为由 LOW_O 变为 0，最大值为 HIGH_O

➤ 关于 P_PID 的算法选择

P_PID 目前支持 3 种算法选择，每路 PID 都可以使用不同的 PID 算法，通过 SMW120 和 SMW152~SMW155 进行 PID 的算法选择。

表 6.9.2 PID 算法选择配置表

变量区	功能	描述
SMW120	整型数，位置式 PID 公式选择寄存器，默认为 0。	0 表示使用不完全微分输出 PID 公式，需要配合 SMW152~SMW155 为不同编号的 PID 选择不同的公式。
		1 表示使用正常算法
		4 表示使用无积分处理的正常算法
		其余保留
SMW152 ~ SMW155	整型数，当寄存器 SMW120 为 0 时 PID 算法选择寄存	16 个 PID 中每个编号的 PID 对应 4 位作为算法选择，其中 SMW152.0~SMW152.3 对应

	器。默认均为 0。	0 号 PID、SMW152.4~SMW152.7 对应 1 号 PID.....依次类推， SMW155.12~SMW155.15 对应 15 号 PID。 目前算法选择为 0 是对应不完全微分输出 2 算法，1 为正常算法，2 为纯 K_p 、 T_i 、 T_d 作用的通用算法，3 为无积分项的正常算法，4 为无积分项的不完全微分输出 2 算法，5 为不完全微分输出 1 算法，6~15 目前保留。

➤ P_PID 自整定功能说明

一般的 PID 控制参数 K_p 、 T_i 和 T_d 需要经验或现场实验人工确定，麻烦且控制效果并不一定理想，通过自整定方法可以自动确定 PID 的控制参数，简单便捷且控制精度高。

控制器的位置式 PID 指令支持自整定功能。并且只对 LOOP 号为 0~7 的 PID 指令起作用。位置式 PID 共支持继电器反馈自整定（有超调）和飞升曲线自整定（无超调）两种方式。

PID 自整定功能需要设置 SM 区的标志位来实现，整定完的参数也保存在 SM 区中，掉电永不丢失。

➤ 算法说明

（1）继电器反馈自整定算法（有过冲）

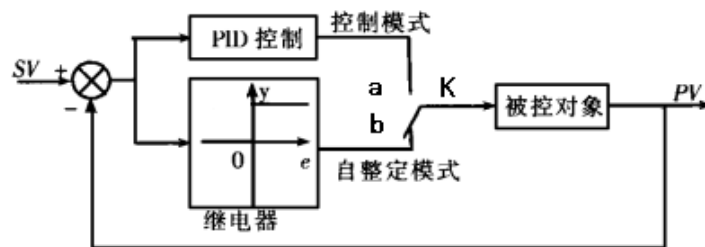


图 6.9.12 继电器反馈自整定系统框图

继电器反馈自整定方法基本原理是通过设定继电器滞环的参数，使系统产生等幅振荡，然后在继电器反馈下观察过程的极限振荡环，由极限振荡环获得过程的临界信息，然后由临界信息算出 PID 的控制参数。

（2）飞升曲线自整定（无过冲）

继电器反馈自整定的方法可以自动计算出 PID 的控制参数，但是由于在自整定时必须要求被控对象经过振荡过程，此过程中会出现被控对象的测量值大于设定值的情况，而这种情况在一些工业场合是不允许出现的，会发生危险或使被控对象的性质发生变化而失去控制的意义（比如化学反应等）。此外，在一些存在大滞后特性的被控对象中，采用继电器反馈自整定的方法所需周期太长（几十个小时以上），同时整定出来的参数控制稳定性也较差。

基于上述需求，可以采用飞升曲线法来确定 PID 控制参数，即在被控对象从冷态上升至设定值的过程中建立数学模型，获取被控对象的模型参数，从而计算出 PID 的控制参数，达到稳定上升不超调的目的。

➤ PID 自整定步骤：

首先确定要进行自整定的位置式 PID 指令（P_PID）的 LOOP 号，并在此指令的输入和输出端填写合适的变量和常数，其中 PB、TI 和 TD 值需要填写变量，不能填写常数；

确保调节回路已经连接正确并使能 PID 运行，此时可以根据相应的 LOOP 号将 SMW320 的相应位置为 1，0~7 号 PID 分别对应 SMW320 的 0~7 位，表示对应的 PID 自整定使能。例如 LOOP=3，则 SMW320 应设置为 0x08，同时将 SMW321 设置为 0x08，表示整定完毕后自动使用整定完毕的 PID 参数，SMW321 的 0~7 位表示相应的 LOOP 号 PID 是否使用自整定 PID 参数。如表 6.9.3 所示。

表 6.9.3 自整定控制变量一览表

SMW320	0~7 位分别代表 0~7 路 PID 自整定的使能位, 1 为使能此路 PID 自整定, 0 为不使能此路 PID 自整定, 其余位保留
SMW321	0~7 位分别代表 0~7 路 PID 是否使用自整定参数, 1 为使用此路 PID 自整定参数, 0 为不使用此路 PID 自整定参数, 其余位保留
SMW322	无过冲控温使能控制位, 0~7 位分别代表 0~7 路 PID 的无过冲控温使能位, 1 为使能此路 PID 无过冲控温, 0 为不使能此路 PID 无过冲控温, 其余位保留

此时 PID 会开始根据设定参数开始自整定, 根据控制对象的特性整定时间不同。

整定完毕后 SMW320 会自动清掉相应的自整定使能位, 下次自整定必须再次置 SMW320 的相应位。

整定完的参数保存在 SM 区中, 具体信息可参看以下 SM 区说明。如果想修正自整定完毕的参数, 可直接在 SM 区相应位置修改, 修改完毕的数据会自动保存。

➤ PID 自整定相关变量

表 6.9.4 有过冲自整定相关 SM 区变量一览表

SMW106	0 表示 PID 自整定时整定 P、I、D 共 3 个参数, 1 表示 PID 自整定时整定 P、I 共 2 个参数, D 参数自动赋值为 0
SMW109	有过冲自整定时滤波使能和无过冲自整定控制字, 2 为使用无过冲自整定方法, 1 为有过冲自整定滤波使能, 0 为有过冲自整定滤波禁止。默认为有过冲自整定滤波使能。
SMW110	自整定时的滤波次数, 可通过它来消除自整定计算时过零点的抖动, 默认为 3 次
SMD112	实数, 自整定时的 Kp 系数, 默认为 0.6, 建议不要修改
SMD114	实数, 自整定时的 Ti 系数, 默认为 0.5, 建议不要修改
SMD116	实数, 自整定时的 Td 系数, 默认为 0.1, 建议不要修改
SMD118	实数, 自整定时的 d 值, 默认为 0.5, 建议不要修改
SMD150	实数, 不完全微分公式中的微分系数, 默认为 0.125, 建议不要修改

表 6.9.5 无过冲自整定相关 SM 区变量一览表

SMW105	整型数, 0~7 位对应无过冲自整定初始输出是否为全部输出, 对应位为 0 则从相应的 SMW339~SMW346 去取相应的百分比作为初始输出, 对应位为 1 则初始为全部输出, 默认为全部输出。
SMW109	低 8 位为有过冲自整定滤波使能和无过冲自整定控制位, 2 为使用无过冲自整定方法, 1 为有过冲自整定滤波使能, 0 为有过冲自整定滤波禁止。默认为 1, 有过冲自整定滤波使能。 高 8 位对应 PID 上电自动整定标志, 即 SM109.8~SM109.15 对应 Loop 为 0~7 的 PID 上电第一次是否自动进行自整定, 对应位为 1 则相应 Loop 的 PID 自动进行自整定, 为 0 则上电第一次不自动进行自整定, 默认为 0
SMD323~ SMD337	实数, SMD323 对应 PID0 的无过冲自整定记录参数值, SMD325 对应 PID1 的无过冲自整定记录参数值……以此类推, SMD337 对应 PID7 的自整定参数值, 本参数为自动生成, 切勿手动修改!
SMW339~ SMW346	整型数, 无过冲自整定初始输出百分比和整定过程中百分比记录值, 在 SMW105 的 0~7 位对应位为 0 时取 SMW339~SMW346 相应的数值作为初始输出百分比, 0~100 对应 0%~100%, 自整定完毕后根据 Loop 为 0~7 将整定过程中最后实际用到的百分比数值写入对应的变量中以备查看。

表 6.9.6 各路 PID 的自整定参数保存位置一览表

SMD256	实数, PID0 自整定参数的 PB 值
SMD258	实数, PID0 自整定参数的 TI 值
SMD260	实数, PID0 自整定参数的 TD 值
SMD262	实数, PID0 自整定参数的采样时间值
SMD264 ~SMD318	实数, PID1~PID7 的自整定参数值, 以上述 PID0 类推, 每个 PID 占用 4 个实数

➤ 相位偏移功能说明:

P_PID 和 I_PID 支持 PWM 输出时的相位偏移功能, 此功能是为了避免现场使用大功率被控设备时, 所有的 DQ 同时输出会造成电网负荷过重的情况。目前只支持 0~7 路 PID 的相位偏移, 且设置时不要大于 TS 的数值, 如果 8 路同时控温时只需要设置 7 路偏移即可, 即第 7 路 PID 无需设置, 自动默认为 0。

相位偏移对应 SM 区的 SMW82~SMW88, 单位为 10ms。举例说明, 如果 TS 设置为 1, 即为 1 秒, 计算输出的 0 号 PID 的 PWM 值为 100, 即 1 秒钟输出 100ms, 若不加偏移, 则此 PID 会在周期的最后 100ms 输出为 1, 前 900ms 输出为 0, 若设置 SMW82 为 10, 则会在前 800ms 输出为 0, 800ms~900ms 输出为 1, 900ms~1000ms 输出为 0, 而如果此时 1 号 PID 的输出也恰好为 100ms 时, 则就会通过相位偏移错开输出。

SMW82~SMW88	SMW82 对应第 0 路的偏移、以此类推.....SMW88 对应第 6 路的偏移, 单位为 10ms。

● 应用示例

(1) PID 有过冲自整定例程:

本例程主要通过 PID 的有过冲自整定法对某控制对象进行自整定并通过整定的参数使其温度控制在 100 度。

源程序如下所示:

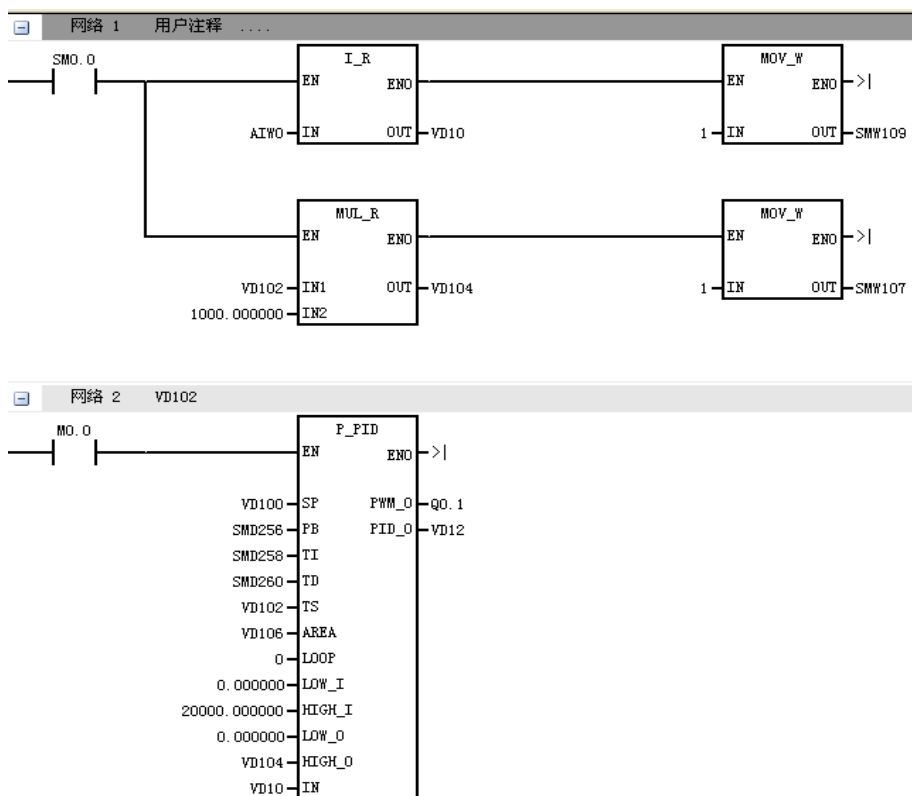


图 6.9.13 PID 有过冲自整定例程

网络 1: 用来对有过冲自整定的各控制参数进行配置, 其中通过 I_R 功能块将输入 AIW0 的有符号整数转换成实数存到 VD10 中, 并将 SMW109 配置为 1 表示有过冲自整定且滤波使能模式, SMW107 也配置为 1 表示设备 10ms 检查一次 DQ 输出。

网络 2: 用来对 PID 自整定时参数进行配置, 其中变量 VD100 为控制设定值 SP 提供了设定值 1000。变量 SMD256、SMD258、SMD260 分别用来存放 PID0 自整定的比例、积分、微分参数 PB、TI、TD, 以便自整定后利用这些参数直接控制温度。变量 VD106 为起控范围 AREA 提供了设定值 800。LOW_I、HIGH_I 这两个参数给出了 PID 的 IN 能够达到的范围。LOW_O、HIGH_O 这两个参数给出了 PID 输出能够达到的范围。PWM_O 接 Q0.1、是当 PID 作为 PWM 控制输出的输出位。

操作步骤如下：

步骤 1：连接被控对象到嵌入式设备 PEC8000，利用 PLC_Config 将编写的功能块指令程序下载到 PEC8000 中。

步骤 2：检查调节回路是否已经连接正确，若已连好，则在状态符号表中根据相应的 LOOP 号将 SMW320 的相应位置为 1，0~7 号 PID 分别对应 SMW320 的 0~7 位，表示对应的 PID 自整定使能、本程序中选用的是 0 号 PID，所以需要在状态符号表中将 SMW320、SMW321 置 1 并将 SMW322 置 0、这样就可以进行 PID 的有过冲自整定。

步骤 3：置位 M0.0 使能 PID 指令，PID 开始进入有过冲自整定。此时 Q0.1 应该为刚开始全速输出，温度到达设定值之后又全部熄灭，反复经过三轮过后即可整定完毕，进入到 PID 控制阶段。整个过程其监视曲线如下所示，红色曲线为温度的变化情况。

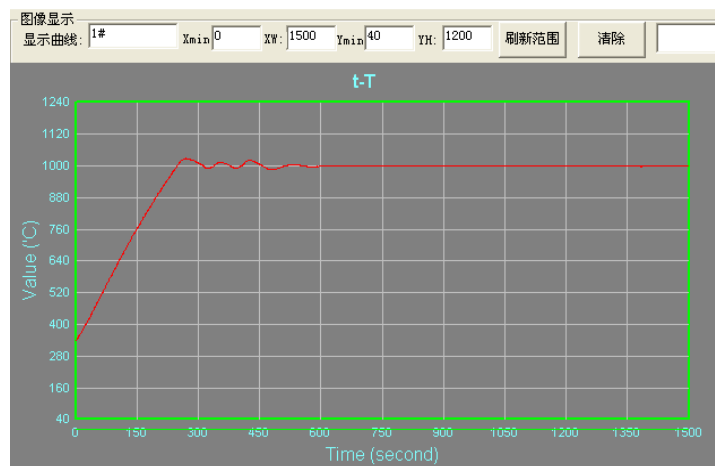


图 6.9.14 PID 有过冲自整定过程监视曲线

(2) PID 无过冲自整定例程：

本例程主要通过 PID 的无过冲自整定法对某控制对象进行自整定并通过整定的参数使其温度控制在 100 度。

程序如图 6.9.15 所示：

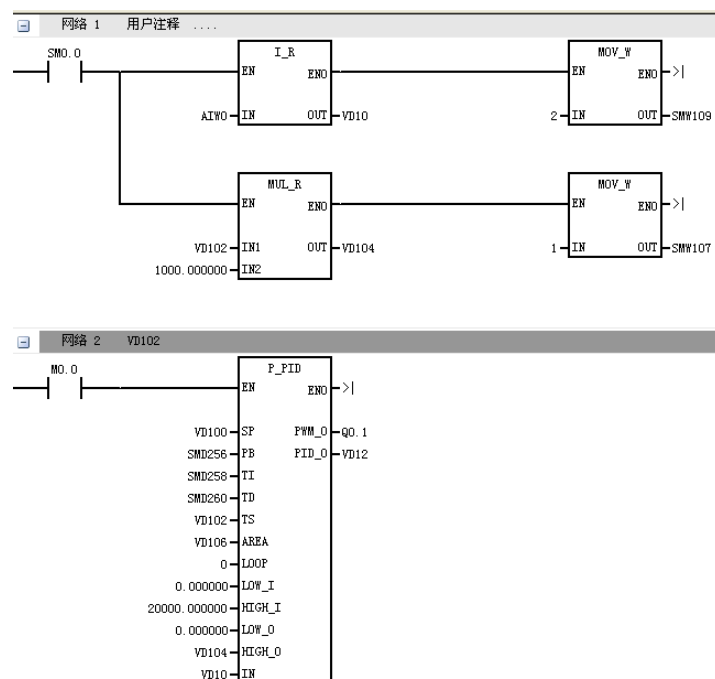


图 6.9.15 PID 无过冲自整定例程

网络 1：用来对无过冲自整定的各控制参数进行配置，其中通过 I_R 功能块将输入 AIW0 的有符号整数

转换成实数存到 VD10 中,并将 SMW109 配置为 2 表示无过冲自整定模式,SMW07 也配置为 1 表示设备 10ms 检查一次 DQ 输出。

网络 2: 用来对 PID 自整定时的参数进行配置,其中变量 VD100 为控制设定值 SP 提供了设定值 1000。变量 SMD256、SMD258、SMD260 分别用来存放 PID0 自整定的比例、积分、微分参数 PB、TI、TD,以便自整定后利用这些参数直接控制温度。变量 VD106 为起控范围 AREA 提供了设定值 800。LOW_I、HIGH_I 这两个参数给出了 PID 的 IN 能够达到的范围。LOW_O、HIGH_O 这两个参数给出了 PID 输出能够达到的范围。PWM_O 接 Q0.1、是当 PID 作为 PWM 控制输出的输出位。

操作步骤如下:

步骤 1: 连接被控对象到嵌入式设备 PEC8000, 利用 PLC_Config 将编写的功能块程序下载到 PEC8000 中。

步骤 2: 检查调节回路是否已经连接正确,若已连好,则在状态符号表中根据相应的 LOOP 号将 SMW320 的相应位置为 1, 0~7 号 PID 分别对应 SMW320 的 0~7 位, 表示对应的 PID 自整定使能。本程序中选用的是 0 号 PID, 所以需要在状态符号表中将 SMW320、SMW321 置 1 并将 SMW322 置 1, 这样就可以进行 PID 的无过冲自整定。

步骤 3: 置位 M0.0 使能 PID 指令, PID 开始进入无过冲自整定。此时 Q0.1 应该为刚开始全速输出, 温度到达设定值约 2/3 时整定完毕, 开始进入控制状态。

整个过程其监视曲线如图 6.9.16 所示, 红色曲线为温度的变化情况:

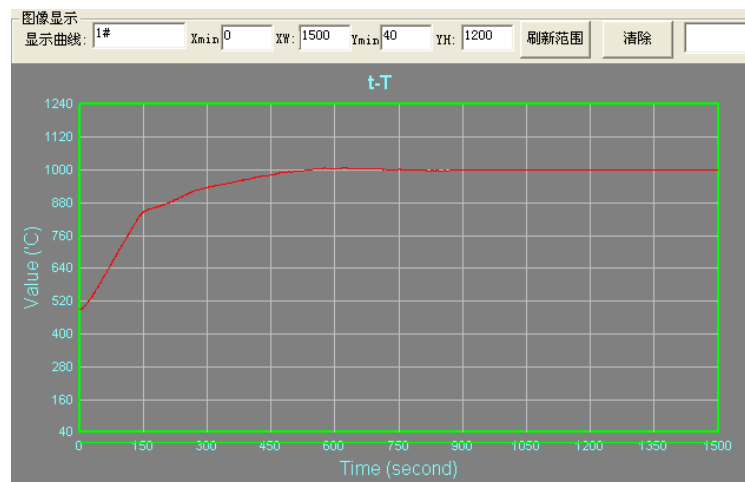


图 6.9.16 PID 无过冲自整定过程监视曲线

6.10 整型数运算指令

整形数运算指令主要用于实现整形数的四则运算和递增/递减功能,其中四则运算包含了 10 个指令,分别为整数加指令、整数减指令、长整数加指令、长整数减指令、整数乘指令、整数除指令、长整数乘指令、长整数除指令、整数短乘指令、整数短除指令。递增/递减功能包含了 6 个指令,字加 1、整数加 1、字减 1、整数减 1,双字加 1,双字减 1 功能。

6.10.1 整数加减乘除

6.10.1.1 整数加和减指令 (ADD_I 和 SUB_I)

● 指令参数

整数加指令 (ADD_I)			
			实现 16 位整数的加法运算。对两个 16 位整数进行加法运算得到 16 位的结果。 $IN1+IN2=OUT$
整数减指令 (SUB_I)			
			实现 16 位整数的减法运算。对两个 16 位整数进行加法运算得到 16 位的结果。 $IN1-IN2=OUT$
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	字变量、常数	INT	输入变量的值
ENO	位变量	BOOL	输出能量流
OUT	字变量	INT	输出变量运算的值

● 详细说明

整数加减指令主要用于 16 位变量与 16 位变量的加减运算后得到的结果为 16 位变量的简单数学运算。

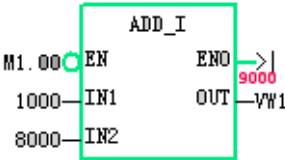
● 应用示例

如图 6.10.1 所示，该例程完成算式 1000+8000-1000 的运算。例程中，EN 端进行了取反操作，下载完程序后立即使能。

程序运行过程：

1000+8000=VW1

VW1-1000=VW2=8000



用户注释

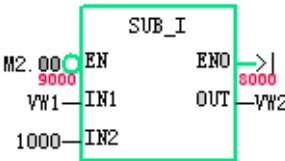


图 6.10.1 整加/减指令例程

6.10.1.2 长整加和减指令 (ADD_DI 和 SUB_DI)

● 指令参数

长整加指令（ADD_DI）		实现双字整数的加法运算；对两个 32 位整数进行加法运算得到 32 位的结果，放入输出中。 IN1+IN2=OUT。	
ADD_DI <<EN ENO-> ????IN1 OUT-???? ????IN2			
长整减指令（SUB_DI）		实现双字整数的减法运算；SUB_DI 对两个 32 位整数进行减法运算得到 32 位的结果，放入输出中。 IN1-IN2=OUT。	
SUB_DI <<EN ENO-> ????IN1 OUT-???? ????IN2			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	双字变量、常数	DINT	输入变量的值
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DINT	输出变量运算的值

● 详细说明

整数加减指令主要用于 32 位变量与 32 位变量的加减运算后得到的结果为 32 位变量的简单数学运算。

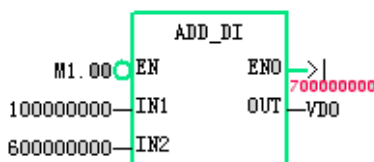
● 应用示例

如图 6.10.2 所示，该例程完成算式 $1000000000+600000000-2000000000$ 的运算。例程中，EN 端进行了取反操作，下载完程序后立即使能。

程序运行过程：

$1000000000+600000000=VD0$

$VD0-2000000000=VD2=500000000$



用户注释

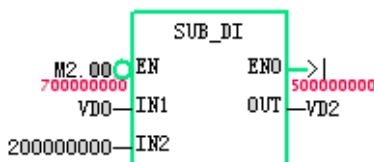


图 6.10.2 长整加/减指令例程

6.10.1.3 整数乘和除指令 (MUL 和 DIV)

● 指令参数

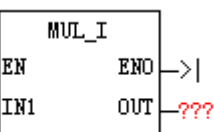
整数乘指令 (MUL)		实现两个 16 位整数的乘法运算；对两个 16 位整数进行乘法运算得到 32 位的结果。 IN1×IN2=OUT	
MUL <<EN IN1 IN2 ENO OUT > ???? ????			
整数除指令 (DIV)		实现两个 16 位整数的除法运算；对两个 16 位整数进行除法运算得到 32 位的结果。 IN1/IN2=OUT	
DIV <<EN IN1 IN2 ENO OUT > ???? ????			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	字变量、常数	INT	输入变量的值
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DINT	输出变量运算的值

6.10.1.4 长整数乘和除指令 (MUL_DI 和 DIV_DI)

● 指令参数

长整乘指令（MUL_DI）		实现长整数的乘法运算；对两个 32 位长整数进行乘法运算得到 32 位长整数的结果。 IN1×IN2=OUT	
MUL_DI <<EN ENO-> ????IN1 OUT-???? ????IN2			
长整整除指令（DIV_DI）		实现长整数的除法运算；对两个 32 位长整数进行除法运算得到 32 位长整数的结果。 IN1/IN2=OUT	
DIV_DI <<EN ENO-> ????IN1 OUT-???? ????IN2			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	双字变量、常数	DINT	输入变量的值
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DINT	输出变量运算的值

6.10.1.5 短整数乘和除指令 (MUL_I 和 DIV_I)

整数短乘指令 (MUL_I)			
		实现两个 16 位整数的乘法运算；对两个 16 位整数进行乘法运算得到 16 位的结果。 $IN1 \times IN2 = OUT$	
整数整除指令 (DIV_I)			
		实现两个 16 位整数的除法运算；对两个 16 位整数进行除法运算得到 16 位的结果。 $IN1 / IN2 = OUT$	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	字变量、常数	INT	输入变量的值
ENO	位变量	BOOL	输出能量流
OUT	字变量	INT	输出变量运算的值

● 详细说明

这一组乘除指令根据输入输出的不同规格划分出了三种，短指令的输入输出都是 16 位有符号整数；普通指令的输入为 16 位有符号整数，输出为 32 位有符号长整数；长指令的输入和输出都是 32 位有符号整数。

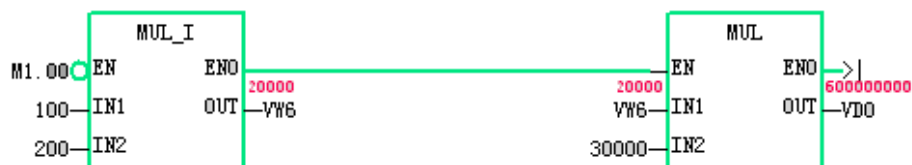
● 应用示例

如图 6.10.3 所示，该例程完成算式 $100 \times 200 \times 30000 / 3 \times 2$ 的运算。例程中，EN 端进行了取反操作，下载完程序后立即使能。

程序运行过程: $100 \times 200 = VW6 = 20000$

$$VW6 \times 30000 = VD0 = 600000000$$

VD0 / 3=VD2=200000000

$$VD2 \times 2 = VD4 = 400000000$$


用户注释



图 6.10.3 整数乘除指令例程

6.10.2 递增和递减

6.10.2.1 字递增和递减指令 (INC_W 和 INC_I 和 DEC_W 和 DEC_I)

● 指令参数

字加 1 指令 (INC_W)			
		实现对输入的无符号字加 1 操作；INC_W 对输入（IN）进行加 1 后，将结果保存到输出 OUT。 IN+1=OUT	
整数加 1 指令 (INC_I)			
		实现对输入的有符号字加 1 操作；INC_I 对输入（IN）进行加 1 后，将结果保存到输出 OUT。 IN+1=OUT	
字减 1 指令 (DEC_W)			
		实现对输入的无符号字减 1 操作；DEC_W 对输入（IN）进行减 1 后，将结果保存到输出 OUT。 IN-1=OUT	
整数减 1 指令 (DEC_I)			
		实现对输入的有符号字减 1 操作；DEC_I 对输入（IN）进行减 1 后，将结果保存到输出 OUT。 IN-1=OUT	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	输入变量的值
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	输出变量运算的值

● 详细说明

递增/递减模块提供了有符号和无符号两类整数的自增和自减功能，用户可以根据需要使用。

● 应用示例

如图 6.10.4 所示，该程序运行后，程序段 1 把有符号数-32768 加 1，运行结果为-32767；

程序段 2 把字 1000 加 1，运行结果为 1001；

程序段 3 把有符号数-32767 减 1，运行结果为-32768；

程序段 4 把字 1000 减 1，运行结果为 999。

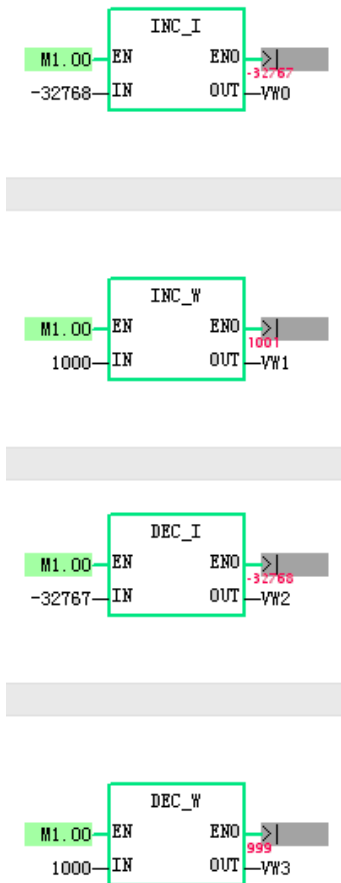


图 6.10.4 字递增和递减指令例程

6.10.2.2 双字递增和递减指令（INC_DW 和 DEC_DW）

● 指令参数

双字加 1 指令（INC_DW）		实现双字的加 1 运算；INC_DW 对输入（IN）进行加 1 后，将结果保存到输出 OUT。 IN+1=OUT	
双字减 1 指令（DEC_DW）		实现双字的加 1 运算；DEC_DW 对输入（IN）进行减 1 后，将结果保存到输出 OUT。 IN-1=OUT	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	DWORD	输入变量的值
OUT	双字变量	DWORD	输出变量运算的值
ENO	位变量	BOOL	输出能量流

● 详细说明

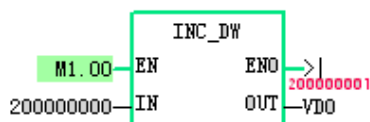
双字的递增/递减模块为用户提供了有符号长整数的自增自减功能。

● 应用示例

如图 6.10.5 所示，该程序运行后：

程序段 1，将双字 200000000 加 1，运行结果为 200000001。

程序段 2，将双字 400000000 减 1，运行结果为 399999999。



用户注释

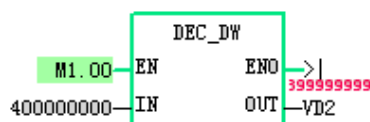


图 6.10.5 双字递增和递减指令例程

6.11 逻辑操作指令

逻辑操作指令主要用于实现字或双字逻辑运算功能，完成 16 位字或 32 位双字的取反、与、或和异或功能。具体包含字取反、双字取反、字与、双字与、字或、双字或、字异或、双字异或指令。

6.11.1 字取反和双字取反指令

- 指令参数

字取反指令 (INV_W)			
实现字的取反功能。			
操作数	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	输入的字
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	输出取反结果
双字取反指令 (INV_DW)			
实现双字的取反功能。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	DWORD	输入的双字
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	输出取反结果

- 详细说明

取反指令分为字和双字两种类型，支持 16 位字和 32 位双字的逻辑取反。

- 应用示例

如图 6.11.1 所示，该程序运行后，INV_W 指令将 16 位数 0x7FFF 取反，得到 16 位数 0x8000，转换成 10 进制数为 32768；INV_DW 指令将 32 位数 0xFFFFFFFF 取反，得到 16 进制数 0x80000000，转换为无符号 10 进制数为 2147483648。

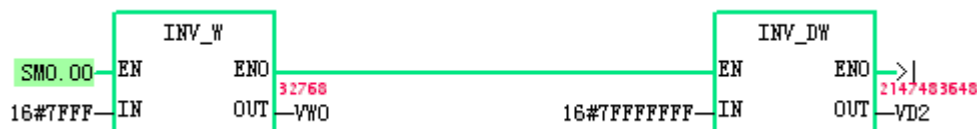


图 6.11.1 取反指令示例

6.11.2 与_或_异或

- 指令参数

6.11.2.1 字与和双字与指令 (WAND_W 和 WAND_DW)

- 指令参数

字与指令 (WAND_W)			
		实现字与功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	字变量、常数	WORD	输入的参数
ENO	位变量	BOOL	输出能量流
OUT	字变量、	WORD	输出逻辑与结果
双字与指令 (WAND_DW)			
		实现双字与功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	双字变量、常数	DWORD	输入参数
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	输出逻辑与结果

6.11.2.2 字或和双字或指令 (WOR_W 和 WOR_DW)

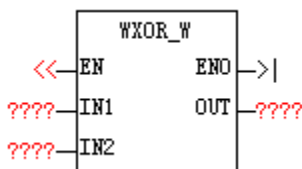
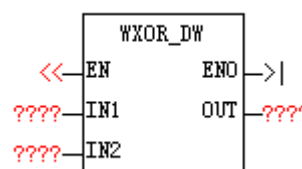
- 指令参数

字或指令 (WOR_W)			
		实现字或功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	字变量、常数	WORD	输入参数
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	输出逻辑或结果
双字或指令 (WOR_DW)		实现双字或功能。	

			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	双字变量、常数	DWORD	输入参数
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	输出逻辑或结果

6.11.2.3 字异或和双字异或指令 (WXOR_W 和 WXOR_DW)

- 指令参数

字异或指令 (WXOR_W)			
			
实现字的异或功能。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	字变量、常数	WORD	输入参数
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	输出逻辑异或结果
双字异或指令 (WXOR_DW)			
			
实现双字的异或功能。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN1/IN2	双字变量、常数	DWORD	输入参数
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	输出逻辑异或结果

- 详细说明

与/或/异或指令主要用于实现字或双字的与/或/异或运算，使用时需要注意该类指令的输入、输出参数都是无符号数，通过输出引脚显示的输出结果是以十进制表示的，可通过符合状态表观察十六进制输出结果。

- 应用示例

如图 6.11.2 所示，该程序运行后，WAND_W 指令将 16 位字 0xFF 和 0x00 做与，运行结果为 0；WAND_W 指令将 32 位字 0xFFFF 和 0x0000 做与，运行结果为 0；WOR_W 指令将 16 位字 0xFF 和 0x00 做或，运行结果为 255；WOR_DW 指令将 32 位字 0xFFFF 和 0x0000 做或，运行结果为 65535；WXOR_W 指令将 16 位字 0xF0 和 0x0F 做异或，运行结果为 255；WXOR_DW 指令将 32 位字 0xFF00 和 0x00FF 做异或，运行结果为

65535。

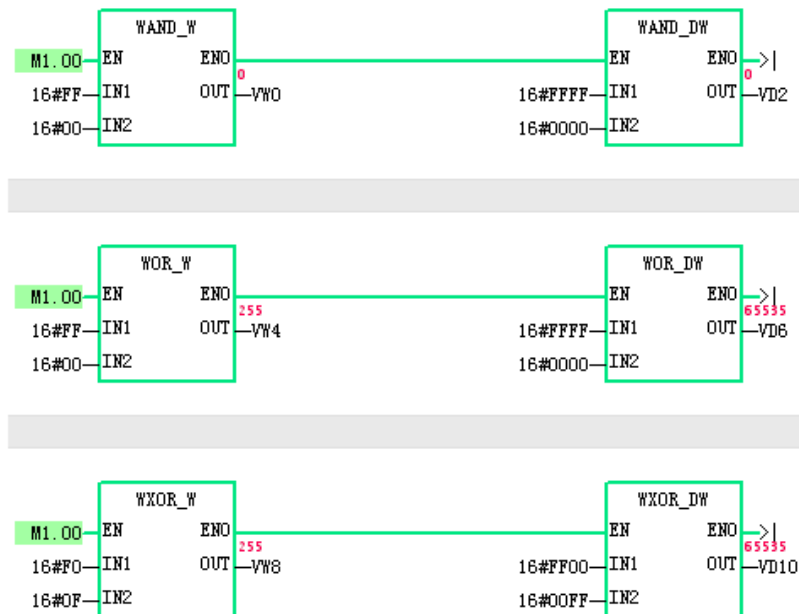


图 6.11.2 异或指令示例

6.12 变量赋值指令

变量赋值指令主要完成变量的转移和赋值操作。

6.12.1 字节_字_双字_实数

● 指令参数

字节赋值指令 (MOV_B)			
		实现位变量的数值转移	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位，高电平触发
IN	位变量	BOOL	要赋值的位变量
ENO	位变量	BOOL	赋值后的结果
OUT	位变量	BOOL	能流输出
字赋值指令 (MOV_W)			
		实现字的赋值功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	要赋值的参数
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	赋值后的结果
双字赋值指令 (MOV_DW)			
		实现双字的赋值功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流

IN	双字变量、常数	DWORD	要赋值的参数
ENO	位变量	BOOL	能流输出
OUT	双字变量	DWORD	赋值后的结果
实数赋值指令 (MOV_R)		实现实数的赋值功能。	
MOV_R <<—EN ENO—> ????—IN OUT—????			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	REAL	要赋值的参数
ENO	位变量	BOOL	输出能量流
OUT	双字变量	REAL	赋值后的结果

- 详细说明

该组指令用于单个变量的赋值。

- 应用示例

如图 6.12.1 所示，M1.00 为模块使能端，IN 端口直接输入常数，结果由 OUT 端口输出。

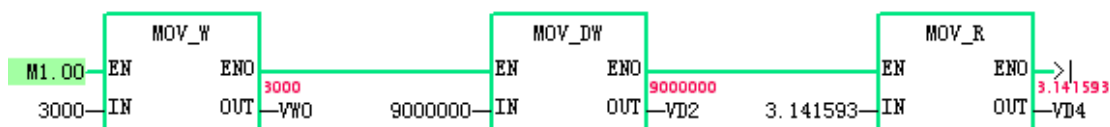


图 6.12.1 移动指令示例

该程序运行后，MOV_W 将字 3000 赋给 VW0，运行结果 VW0=3000；MOV_DW 指令将双字 9000000 赋给 VD2，运行结果 VD2=9000000；MOV_R 指令将实数 3.141593 赋给 VD4，运行结果 VD4=3.141593。

6.12.2 字块

6.12.2.1 单字块赋值指令 (BLKMOV_W)

- 指令参数

单字块赋值指令 (BLKMOV_W)		实现单字块的赋值功能。	
BLKMOV_W <<—EN ENO—> ????—IN OUT—???? ????—N			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	赋值字的首地址
N	字变量、常数	WORD	赋值字的长度 ≤4096
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	输出结果
双字块赋值指令 (BLKMOV_DW)		实现双字块的赋值功能。	
BLKMOV_DW <<—EN ENO—> ????—IN OUT—???? ????—N			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	DWORD	赋值双字的首地址

N	字变量、常数	WORD	赋值双字的长度 ≤ 4096
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	输出结果

● 详细说明

字块移动指令把输入 IN 开始的 N 个字 ($N \times 2$ 字节) 移动至输出 OUT 开始的 N 个字地址中 ($N \times 2$ 字节), 并且不改变 IN 的值。输入为常数, 则将 N 个常数移动到输出变量区。输入为变量, 则从输入变量地址开始, 将 $N \times 2$ 个字变量的数值放入到 OUT 中。

双字块移动指令把输入从 IN 开始的 N 个双字数据 ($N \times 4$ 字节) 移动至输出 OUT 开始的 N 个双字地址中 ($N \times 4$ 字节), 并且不改变 IN 的值。若输入为常数, 则将 N 个常数移动输出变量区。若输入为变量, 则从输入变量地址开始, 将 $N \times 2$ 个双字变量的数值放入到 OUT 中。

● 应用示例

如图 6.12.2 所示, 通过字块和双字块指令对 V 区和 M 区进行赋值。

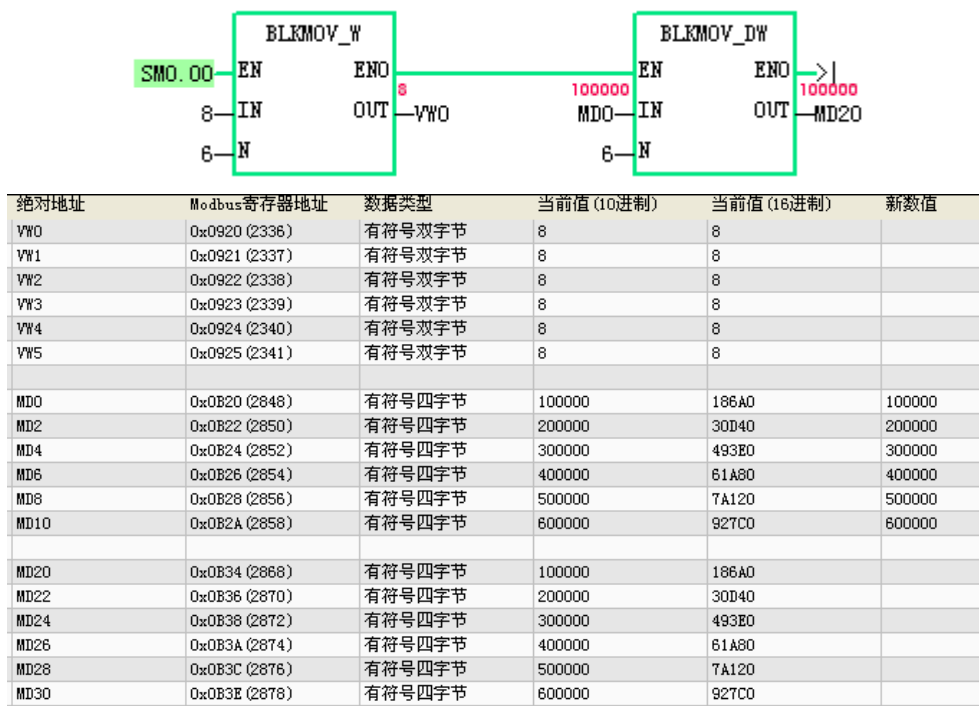


图 6.12.2 字块移动指令示例

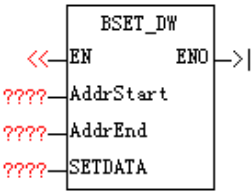
程序运行后, BLKMOV_W 将字 8 赋给 VW0~MW5, 运行结果 VW0~MW5 都为 8; BLKMOV_DW 指令将 MD0~MD10 的值赋给 MD20~MD30, 运行结果 MD0=MD20, 依次类推。

提示: 注意输入长度 N 与输入变量区的起始之和不能超过输入变量区的大小, 例如双字块移动指令的输入为 MD500, N=10, 则 $500+10 \times 2=520$ 已经超过了 M 区的最大值 512, 会造成指令执行错误。

6.12.2.2 字和双字批量赋值 (BSET_W 和 BSET_DW)

● 指令参数

BSET_W << EN ENO >> ???? AddrStart ???? AddrEnd ???? SETDATA		对连续多个字变量赋值相同的数据。 将从 AddrStart 到 AddrEnd 字地址数据全部赋值为 SETDATA。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流

AddrStart	字变量	WORD	赋值字首地址
AddrEnd	字变量	WORD	赋值字尾地址
SETDATA	字变量、常数	WORD	赋值字数据
ENO	位变量	BOOL	输出能量流
		对连续多个双字变量赋值相同数据。 将从从 AddrStart 到 AddrEnd 双字地址数据全部赋值为 SETDATA。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
AddrStart	双字变量	DWORD	赋值双字首地址
AddrEnd	双字变量	DWORD	赋值双字尾地址
SETDATA	双字变量、常数	DWORD	赋值字数据
ENO	位变量	BOOL	输出能量流

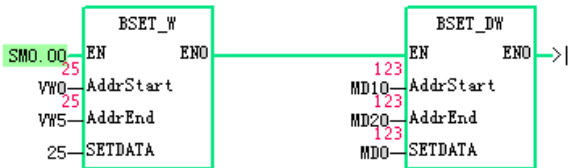
● 详细说明

单字块赋值指令把从 AddrStart 到 AddrEnd 字地址中数据全部赋值为 SETDATA, AddrStart 与 AddrEnd 必须为同一字寄存器区地址。

双字块赋值指令把从 AddrStart 到 AddrEnd 双字地址中数据全部赋值为 SETDATA, AddrStart 与 AddrEnd 必须为同一双字寄存器区地址。

● 应用示例

如图 6.12.3 所示, 通过批量赋值指令对 V 区和 M 区进行赋值。



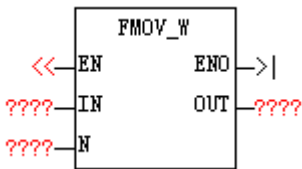
绝对地址	Modbus寄存器地址	数据类型	当前值 (10进制)	当前值 (16进制)	新数值
VW0	2336	有符号双字节	25	19	25
VW1	2337	有符号双字节	25	19	25
VW2	2338	有符号双字节	25	19	25
VW3	2339	有符号双字节	25	19	25
VW4	2340	有符号双字节	25	19	25
VW5	2341	有符号双字节	25	19	25
MD0	2848	有符号四字节	123	7B	123
MD10	2858	有符号四字节	123	7B	123
MD12	2860	有符号四字节	123	7B	123
MD14	2862	有符号四字节	123	7B	123
MD16	2864	有符号四字节	123	7B	123
MD18	2866	有符号四字节	123	7B	123
MD20	2868	有符号四字节	123	7B	123

图 6.12.3 批量赋值指令例程

该程序运行后, BSET_W 将字 8 赋给 VW0~MW5, 运行结果 VW0~MW5 都为 8; BSET_DW 指令将 MD0 的值赋给 MD10~MD20。

6.12.2.3 多点变量赋值指令 (FMOV_W)

● 指令参数

		将字赋值到多个连续的变量地址中。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	源赋值字的地址
N	字变量、常数	WORD	目的赋值变量地址个数，最大 128
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	目的赋值变量首地址

- 详细说明

多点变量赋值指令可以对同一变量区内连续的变量进行统一赋值。

- 应用示例

如图 6.12.4 所示，通过多点赋值指令对 V 区和 M 区进行赋值。

```

    graph LR
      SMO00[SMO.00] -- EN --> FMOV_W1[FMOV_W]
      F5_1[5] -- IN --> FMOV_W1
      F5_N1[5] -- N --> FMOV_W1
      FMOV_W1 -- OUT --> VWO[VWO]
      VWO -- EN --> FMOV_W2[FMOV_W]
      MW0[MW0] -- IN --> FMOV_W2
      F5_N2[5] -- N --> FMOV_W2
      FMOV_W2 -- OUT --> MW10[MW10]
  
```

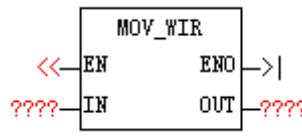
绝对地址	Modbus寄存器地址	数据类型	当前值 (10进制)	当前值 (16进制)	新数值
VW0	0x0920 (2336)	有符号双字节	5	5	
VW1	0x0921 (2337)	有符号双字节	5	5	
VW2	0x0922 (2338)	有符号双字节	5	5	
VW3	0x0923 (2339)	有符号双字节	5	5	
VW4	0x0924 (2340)	有符号双字节	5	5	
MW0	0x0B20 (2848)	有符号双字节	10	A	10
MW10	0x0B2A (2858)	有符号双字节	10	A	
MW11	0x0B2B (2859)	有符号双字节	10	A	
MW12	0x0B2C (2860)	有符号双字节	10	A	
MW13	0x0B2D (2861)	有符号双字节	10	A	
MW14	0x0B2E (2862)	有符号双字节	10	A	

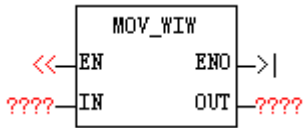
图 6.12.4 多点变量赋值指令示例

程序运行后，第一个 FMOV_W 指令将字 5 赋给 VW0~VW4，运行结果 VW0~VW5 都为 5；第二个 FOMV_W 指令将 MW0 中的值 10 赋给 MW10~MW14，运行结果 MW10~MW14 都为 10。

6.12.3 立即传送

- 指令参数

立即传送读指令 (MOV_WIR)			
		读物理输入，并立即刷新输入变量区。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量	WORD	待读取数据
ENO	位变量	BOOL	输出能量流
OUT	字变量、常数	WORD	存储被读取数据
立即传送写指令(MOV_WIW)		从内存中读取数据，并立即写入到物理输出点。	

			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	写入的数据
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	输出数据

注意：立即传送读和立即传送写指令会立即刷新输入输出变量区，但不会立即刷新灯板，即当使用立即传送读和立即传送写指令时，如果想要确定物理输入输出点是否有值，需要利用其他辅助手段（例如万用表）去检查而无法从灯板上直观的看到现象。

- 详细说明

该组指令用于立即读写物理输入输出。

- 应用示例

如图 6.12.5 所示，利用立即读指令控制开出。

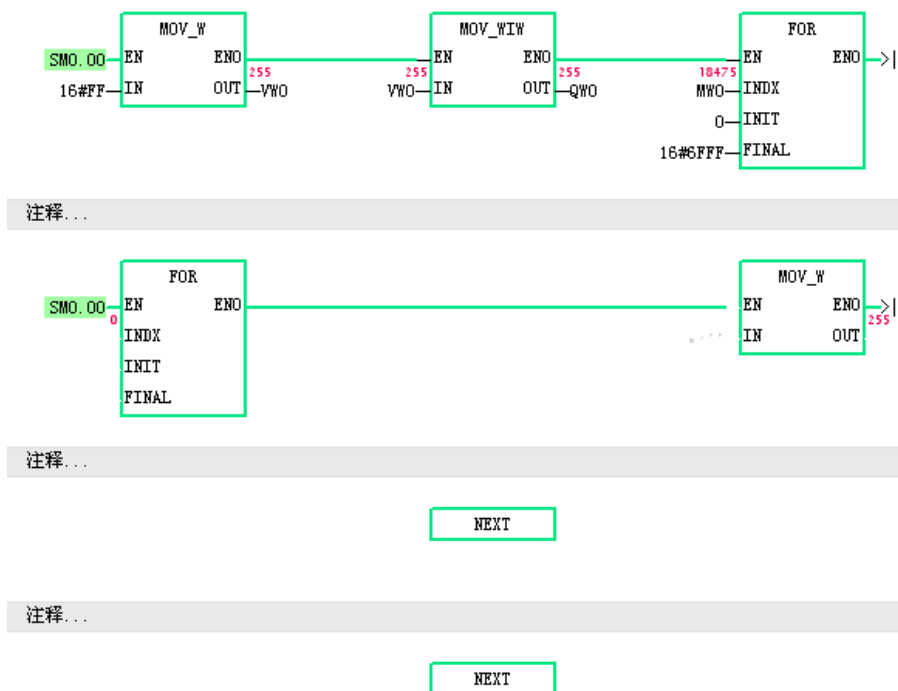


图 6.12.5 立即写指令示例

程序运行后，FOR 循环产生 10s 延时；立即输出指令在程序未执行完一个周期时输出，Q0.0~Q0.15 全部输出。如果将示例中的 MOV_WIW 指令用 MOV_W 指令替换，则替换后的 MOV_W 指令会延时 10 后输出，10s 后 Q0.0~Q0.15 全部输出。

如图 6.12.6 所示，利用立即读、写指令点亮 Q0.0~Q0.15。

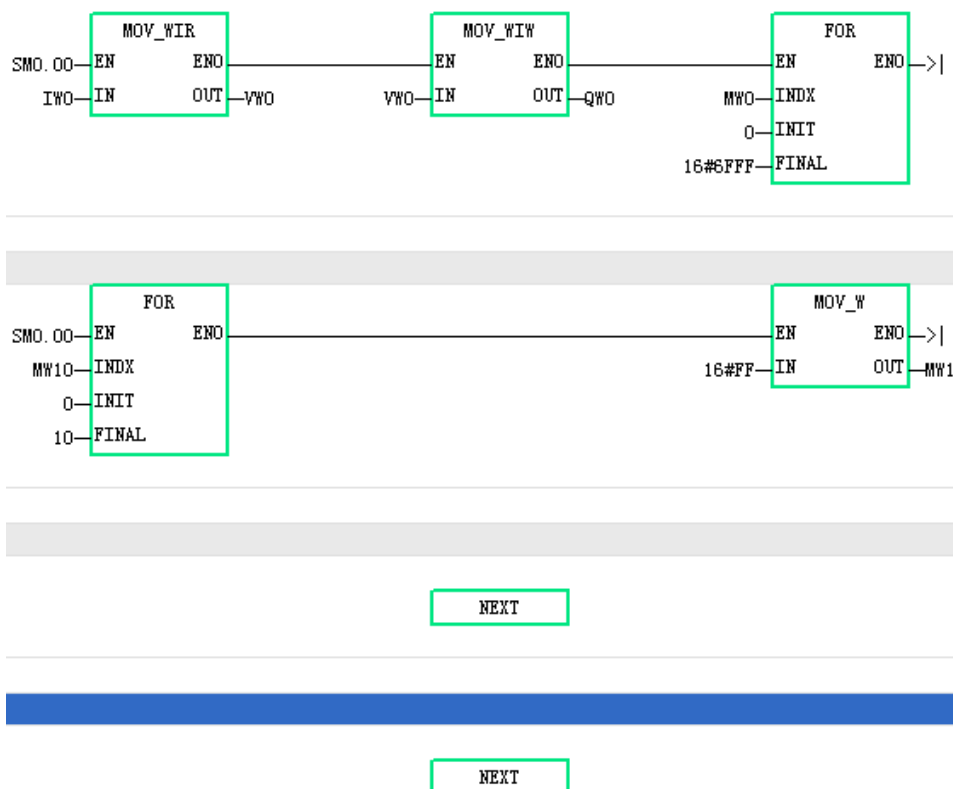


图 6.12.6 立即读指令示例

程序运行后，FOR 循环产生 10s 延时，立即读指令在程序未执行完一个周期时读输入的值，立即输出指令在程序未执行完一个周期时输出，Q0.0~Q0.15 全部输出。如果将示例中的 MOV_WIR 和 MOV_WIW 指令用 MOV_W 指令替换，则替换后的 MOV_W 指令会延时 12 后输出，12s 后 Q0.0~Q0.15 全部输出。

6.13 程序控制指令

程序控制指令主要用于控制程序的循环、跳转和子程序调用功能。其中，循环指令包含循环头指令和循环尾指令，跳转指令包含跳转头指令和跳转尾指令。

6.13.1 循环和跳转

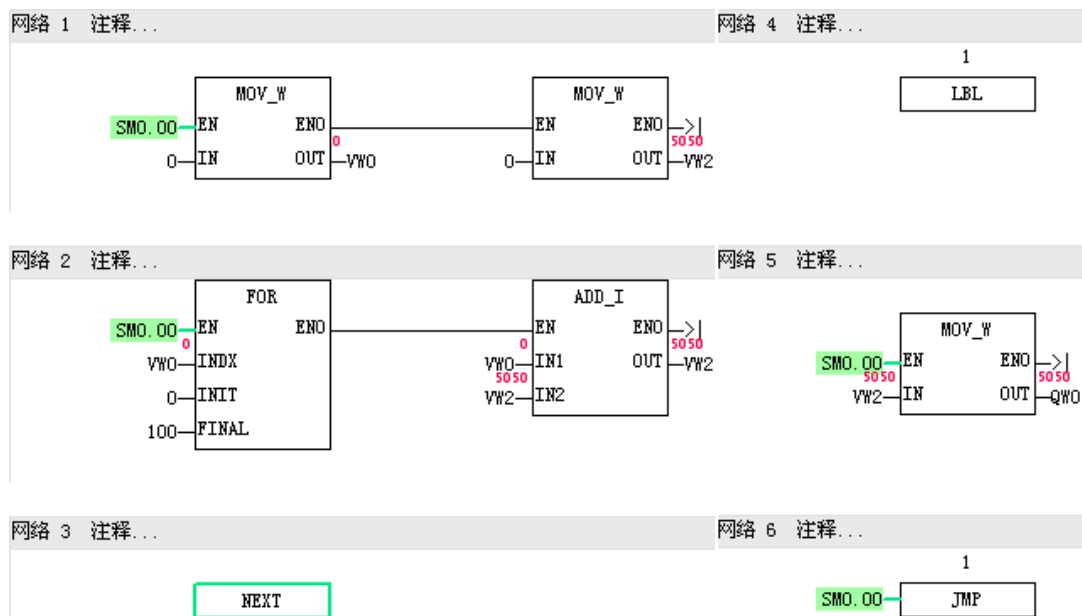
● 指令参数

循环头指令（FOR）		执行从 INIT 开始到 FINAL 次循环。	
FOR <<—EN ENO—> ????—INDX ????—INIT ????—FINAL			
循环尾指令（NEXT）		运行到此指令时结束循环。	
NEXT			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
INDX	字变量	INT	当前循环次数
INIT	字变量、常数	INT	循环初值
FINAL	字变量、常数	INT	循环终值
ENO	位变量	BOOL	输出能量流
跳转头指令（JMP）		JMP 指令和 LBL 指令成对执行，实现跳转功能。	

FOR 循环功能块，执行从 INIT 开始到 FINAL 次循环，当前循环数在 INDX 中给出来。循环体执行的范围是从 FOR 开始，到 NEXT 结束之间的指令。循环结束后，接着从 NEXT 指令开始运行。每一个 FOR 必须对应一个 NEXT，否则指令编译不能通过。另外 FOR 循环指令支持多层嵌套，最深可以达到 255 层。

● 应用示例

如图 6.13.1 所示，程序运行后，利用 FOR 循环指令进行 100 次循环，计算 1~100 自然数的和。网络 2 到网络 4 为循环体，加数 VW0 接到 INDX，VW0 每循环一次加 1，VW0 和 VW2 的和送入 VW2，循环 100 次实现 1~100 自然数的相加。循环结束后，执行网络 5 到网络 7，该段程序为死循环，程序停留在此位置，同时将结果送入 QW0 显示。



6.13.2 子程序调用

子程序调用 (CALL)		实现子程序调用功能	
参数名	变量区	数据类型	说明

EN	位变量	BOOL	输入能量流
???	常数 (1~255)	DWORD	子程序号

● 详细说明

CALL 指令完成对子程序调用的功能，在任何程序块中，都可以用该指令调用指定编号的子程序，该指令不能调用主程序。

例如 SBR_n=1 则在 EN 使能的情况下可以调用 1 号子程序。

通过 CALL 指令和子程序的编写，可以实现整个工程的模块化编程。

子程序支持最大 16 层嵌套，SBR_n 标号范围为 1~255。如果子程序嵌套超过 16 层，SM180.0 会被置 1，提示用户嵌套层数超过最大限制。

● 应用示例

如图 6.13.2 所示，程序运行后，使能 M0.0，主程序中使用 CALL 指令调用编号为 1 的子程序，在子程序 1 中点亮 Q0.0。

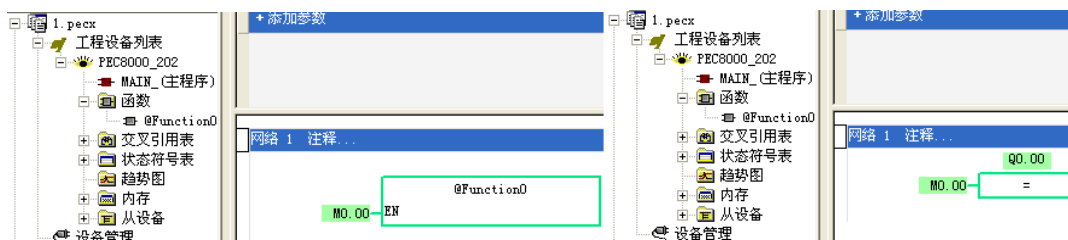


图 6.13.2 子程序调用指令示例

6.13.3 G 代码调用

● 指令参数

GPROGRUN $\ll$ EN SBR_n START ENO DONE TYPE $\rightarrow$??? ???					
启动或停止 G 代码执行。					
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
SBR_n	常数	WORD	正整数	无	G 代码指令程序 ID
Start	位变量	BOOL	0 或 1	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
TYPE	字变量	WORD	非负整数	功能块运行完毕类型，1 为 M02 程序停止，2 为 M30 程序结束，3 为进给保持停止，4 为打断停止	

● 详细说明

通过调用 GPROGRUN 功能块，用户可以控制 G 代码的启动和停止。

在执行 G、M、S、T 代码时，需要固定功能的 PLC 控制 I/O 点，完成 PLC 与 G 代码指令间的通信。

采用 V 区 VW20000~VW21999 作为 G 代码指令 PLC 控制 I/O 点。其中 VW20000~VW20999 为 G 代码指令输出给 PLC 控制点，如表 6.13.1 所示，即执行 NC 代码指令时产生的 PLC 控制信号，PLC 可根据这些信号完成主轴启动、冷却液的开关等功能；VW21000~VW21999 为 PLC 输出给 G 代码指令控制点，如表 6.13.2 所示，通过这些 I/O 信号，控制 G 代码指令的执行，如进给保持、单段执行、复位、急停等功能。

表 6.13.1 NC 代码指令输出给 PLC 控制点定义

地址	符号	说明	使能条件
V20000.0	AUTOST	自动运行方式选择中	V21001.0 置 1
V20000.2	MANUALST	手动方式选择中	V21001.2 置 1
V20000.4	STEPST	单段执行中	V21001.4 置 1
V20000.6	STARTST	启动运行中	V21000.0 置 1
V20000.7	HOLDST	进给保持中	V21000.1 置 1 或单段执行完成
V20000.9	RSTST	复位中	V21000.2 置 1 或 M30 程序结束
V20001.0	MSTFIN	MST 代码分配完成标志	M 代码转换为 BCD 码完成
V20001.1	MFLG	M 代码读取完成标志	M 代码读取完成
V20001.2	TFLG	T 代码读取完成标志	T 代码读取完成
V20001.3	SFLG	S 代码读取完成标志	S 代码读取完成
V20002.0~V20002.15	MBCD	M 指令对应 BCD 码	M 代码 BCD 码转换完成
VW20004	TNUM	T 指令值	获取 T 代码刀号完成
V20005.0	TTABFLG	刀具表更新完成标志	V21003.0 置 1, 刀具表更新完成
VD20006	SSPD	S 指令值	获取 S 代码指令值完成

表 6.13.2 PLC 输出给 NC 代码指令控制点

地址	符号	说明	使能条件
V21000.0	START	启动 G 代码指令	PLC 中将 V21000.0 置 1
V21000.1	HOLD	进给保持	PLC 中将 V21000.1 置 1
V21000.2	RST	G 代码指令复位	PLC 中将 V21000.2 置 1
V21001.0	AUTO	自动运行	PLC 中将 V21001.0 置 1
V21001.2	MANUAL	手动模式, 停止 NC 代码指令执行	PLC 中将 V21001.2 置 1
V21001.4	STEP	单段执行 G 代码指令	PLC 中将 V21001.4 置 1
V21002.0	MSTDONE	辅助功能完成标志	PLC 中将 V21002.0 置 1
V21003.0	TTABUPD	刀具表更新	PLC 中将 V21003.0 置 1

G 代码指令地址主要由字母和数字组成, 控制器使用 G 代码指令地址含义如表 6.13.3 所示。

表 6.13.3 G 代码指令字地址表

功能	地址	说明
顺序号	N	当前 G 代码段地址, 由 1 开始自动生成
准备功能	G	控制器运动方式指令, 如 G00 定位、G01 直线插补、G02 顺圆插补
坐标	X,Y,Z,A,B,C	运动指令坐标 XYZ: 直线轴绝对坐标 ABC: 旋转轴绝对坐标
圆弧半径	R	G02/G03 圆弧插补时, 指定圆弧半径
圆弧中心	I,J,K	圆心中, 起点到圆心距离
进给功能	F	进给速度指定
辅助功能	M	辅助动作代码
主轴功能	S	主轴转速指定代码
刀具功能	T	刀具选择
暂停	F	暂停时间指定
偏置号码	H,D	D: 刀具半径补偿刀具号 H: 刀具长度补偿刀具号
参考点选择	E	G30 使用时, 指定返回参考点 2、3 或 4
子程序地址	L	M98 指定子程序地址

控制器支持使用的 G 代码如表 6.13.4 所示。

表 6.13.4 G 代码一览表

G 代码	组号	说明
G00*	01	定位
G01	01	直线插补
G02	01	顺圆插补 CW
G03	01	逆圆插补 CCW
G04	00	暂停
G09	00	准停指令
G17*	02	XOY 平面
G18	02	XOZ 平面
G19	02	YOZ 平面
G20	06	英制尺寸
G21*	06	公制尺寸
G25	20	刀具半径补偿直线方式过渡
G26	20	刀具半径补偿圆弧方式过渡
G28	00	返回机械原点
G29	00	机械原点返回
G30	00	返回第 2~4 参考点
G40*	07	取消刀具半径补偿
G41	07	刀具半径补偿（左侧）
G42	07	刀具半径补偿（右侧）
G43	08	刀具长度补偿+
G44	08	刀具长度补偿-
G49*	08	刀具长度补偿取消
G53	00	机械坐标系定位
G54*	14	工件坐标系 1 选择
G55	14	工件坐标系 2 选择
G56	14	工件坐标系 3 选择
G57	14	工件坐标系 4 选择
G58	14	工件坐标系 5 选择
G59	14	工件坐标系 6 选择
G90*	03	绝对编程方式
G91	03	相对编程方式
G92	00	工件坐标系设置

注意

[1] 同组 G 代码不能在同一段出现，否则报错处理；

[2] 00 组 G 代码为非模态代码，仅当前段有效；

[3] 带*号 G 代码为初始默认的 G 代码；

[4] 非 00 组 G 代码为模态代码，可省略，即 G 代码被指定后一直有效，直至同组的另一个 G 代码被指定才无效；

● 时序图

图 6.13.3 为 GPROGRUN 功能块指令的引脚时序和速度变化图。当 Enable 为“1”时，V21001.0 置“1”，V21000.0 置“1”，G 代码程序开始执行；当执行程序结束指令 M02 或 M30 时，Done 置“1”，当 Enable 为“0”时，Done 置“0”。TYPE 可以查看 G 代码结束类型。

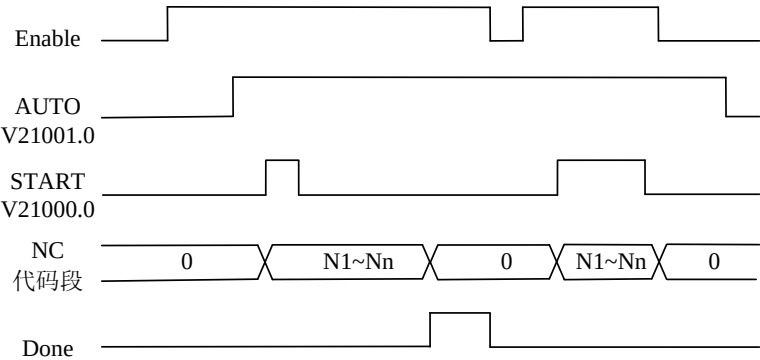


图 6.13.3 GPROGRUN 功能块指令时序图

● 应用示例

PLC_Config 子程序中添加 G 代码程序如图 6.13.4 和图 6.13.5 所示。在主程序中配置 GPROGRUN 的输入引脚 SBR_n 为 G 代码程序 ID=1 如图 6.13.6 所示。将 M1.0 置“1”启动 NC 代码功能。

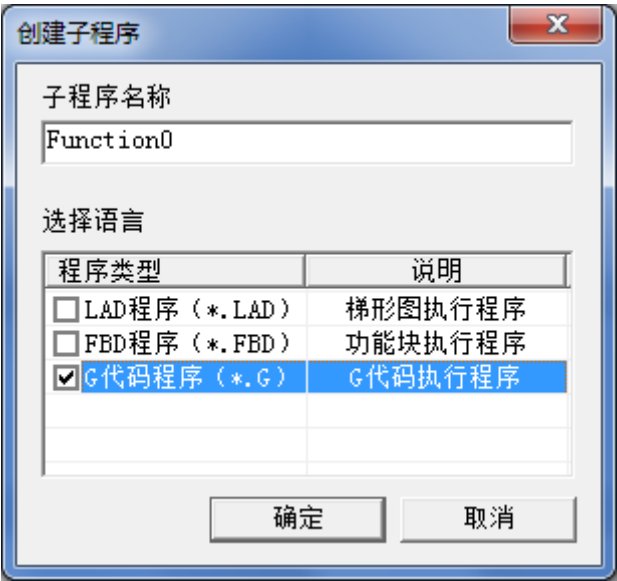


图 6.13.4 添加 G 代码程序

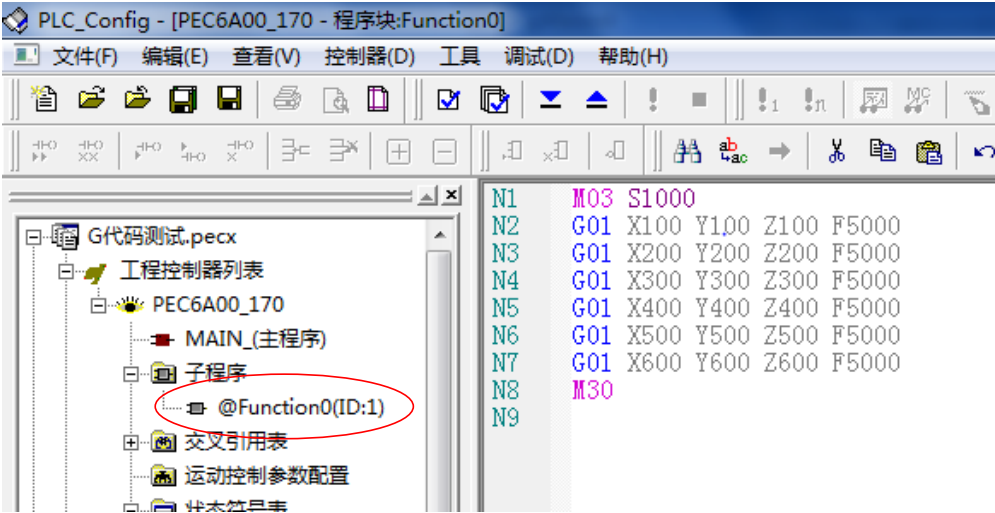


图 6.13.5 G 代码指令程序编写

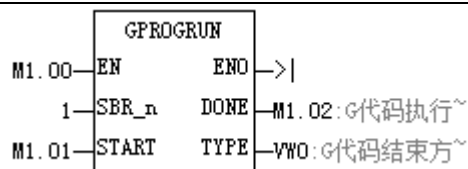


图 6.13.6 G 代码功能启动

如图 6.13.7 所示，将 M10.0 置“1”，自动运行 V21001.0 被置“1”。将 M11.01 置“1”后，V20000.0 输入一个脉冲信号，开始执行 G 代码指令。

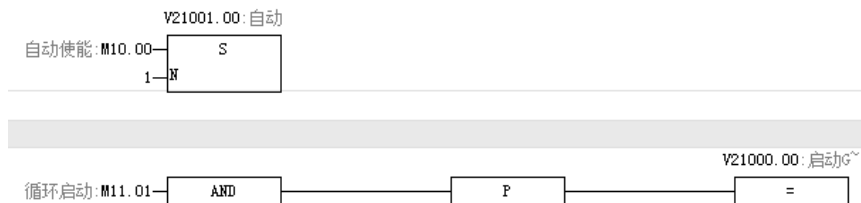


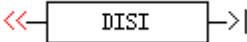
图 6.13.7 G 代码指令执行

6.14 中断指令

控制器提供了多个应用中断，可以有效提高事件响应的实时性。其中包括 I/O 中断、定时中断和自由通信中断。定时中断分为定时器指令中断和匹配定时中断。使用时需要通过相应的中断指令实现各种中断功能，控制器支持的中断指令包括中断使能、中断禁能、中断连接、中断分离指令。

6.14.1 中断允许和禁止指令

● 指令参数

中断允许指令（ENI）			作为中断的总使能开关，使能时允许执行所有中断。
中断禁止指令（DISI）			作为中断的总禁能开关，使能时禁止执行所有中断。
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
ENO	位变量	BOOL	输出能量流

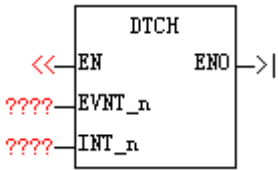
6.14.2 中断连接指令

● 指令参数

中断连接指令 (ATCH)			
		连接中断事件号和中断子程序。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能标志位
INT	常数	WORD	调用的中断子程序号
EVNT	常数	WORD	触发的中断事件号
MODE	常数	WORD	中断响应方式

6.14.3 中断分离指令

● 指令参数

中断分离指令 (DTCH)			
		分离中断事件号和中断子程序。	
参数名	变量区	数据类型	数据范围
EN	位变量	BOOL	使能标志位
EVNT_n	常数(0~19)	WORD	调用的中断子程序号
INT_n	常数(1~128)	WORD	触发的中断事件号

● 详细说明

中断指令可分为两类，中断允许/禁止指令主要用于使能或禁止中断事件；中断连接/分离指令主要用于连接或分离中断事件号和中断子程序。

不同的中断事件号 Event（即中断源）对应不同的中断。每一个中断事件通过 ENI 指令使能，通过 ATCH 指令建立连接，将 INT_n 引脚指定中断服务程序的标号与 EVNT_n 引脚指令的中断事件号绑定。中断事件通过 DTCH 指令取消连接。当中断事件触发后，主程序被打断，进入 ATCH 指定的中断程序开始运行，当中断程序执行完毕后返回到主程序继续执行。

如果需要在中断程序中立即响应物理输入或输出，需要使用立即输出（=I）、立即置位（SI）、立即复位（RI）或立即传送指令实现。

提示：

(1) 在中断连接或使能时，最好使用 SM0.1 来进行控制，使得在用户程序初始化时仅运行一次中断使能和连接，防止中断队列发生混乱。

(2) 注意 I/O 中断中只能同时使用上升沿、下降沿、上升下降沿或高速计数的一种。

(3) 当 A 中断处理中，B 中断来时，只能进入队列等待处理，不会打断原有中断执行体。

(4) 多个中断事件之间采用先进先出的方式执行。

(4) 控制器提供的 I/O 中断的最小时基不能小于 1ms。

➤ 中断事件说明

不同类型控制器支持的中断个数不同，以 PEC8000 控制器为例，该控制器提供了 22 个中断，可以有效提高事件响应的实时性。其中包括 10 个 I/O 中断，8 个定时中断和 4 个自由通信中断。8 个定时中断分为 4 个定时器指令中断和 4 个匹配定时中断。如表 6.14.2 所示。

I/O 中断主要包括外部开关量输入中断、高速输入中断和高速输出中断。开关量输入中断主要包括上升沿和下降沿中断，高速输入中断是指当前值等于预先设定值时产生的中断，高速输出中断是在高速输出完成时产生的中断。

定时器中断有 8 个，其中 4 个匹配定时中断分别对应中断号 8~11，匹配设定值分别存储在 SMW190、SMW191、SMW192、SMW193 四个变量区中，当计数值与匹配值相等时触发中断程序。如表 6.14.1 所示。4 个定时器指令中断分别对应中断号 12~15，使用定时器指令中断时需要调用定时器指令（TON 或 TOF）与中断连接指令配合使用。

表 6.14.1 定时匹配变量表

SM 区	描述
SMW190	定时 0 中断定时时间，单位为 10ms
SMW191	定时 1 中断定时时间，单位为 10ms
SMW192	定时 2 中断定时时间，单位为 10ms
SMW193	定时 3 中断定时时间，单位为 10ms

定时器 T124、T125、T126、T127 则允许用户配置为接通延时定时器(TON)或断开延时定时器(TOF)，一

一旦中断允许，当定时器当前值等于预设值时，触发与其连接的中断程序。

串行口接收发送中断是为自由格式通信指令的串行通信及时响应而设计，主要包括串口 1 发送中断，串口 1 接收中断，串口 2 发送中断和串口 2 接收中断。

表 6.14.2 可编程控制器和 MAC 系列主控器中断事件一览表

中断号	MODE	中断事件	描 述
0	0	I00 上升沿	I00 引脚上升沿产生中断
	1	I00 下降沿	I00 引脚下降沿产生中断
	2	高速计数 0	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I00 上升下降沿	I00 引脚上升沿或下降沿都产生中断
1	0	I01 上升沿	I01 引脚上升沿产生中断
	1	I01 下降沿	I01 引脚下降沿产生中断
	2	高速计数 1	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I01 上升下降沿	I01 引脚上升沿或下降沿都产生中断
2	0	I02 上升沿	I02 引脚上升沿产生中断
	1	I02 下降沿	I02 引脚下降沿产生中断
	2	高速计数 2	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I02 上升下降沿	I02 引脚上升沿或下降沿都产生中断
3	0	I03 上升沿	I03 引脚上升沿产生中断
	1	I03 下降沿	I03 引脚下降沿产生中断
	2	高速计数 3	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I03 上升下降沿	I03 引脚上升沿或下降沿都产生中断
4	0	I04 上升沿	I04 引脚上升沿产生中断
	1	I04 下降沿	I04 引脚下降沿产生中断
	2	高速计数 4	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I04 上升下降沿	I04 引脚上升沿或下降沿都产生中断
5	0	I05 上升沿	I05 引脚上升沿产生中断
	1	I05 下降沿	I05 引脚下降沿产生中断
	2	高速计数 5	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I05 上升下降沿	I05 引脚上升沿或下降沿都产生中断
6	0	I06 上升沿	I06 引脚上升沿产生中断
	1	I06 下降沿	I06 引脚下降沿产生中断
	2	高速计数 6	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I06 上升下降沿	I06 引脚上升沿或下降沿都产生中断
7	0	I07 上升沿	I07 引脚上升沿产生中断
	1	I07 下降沿	I07 引脚下降沿产生中断
	2	高速计数 7	高速增计数 $CV \geq SP$ (当前值=预设值) 产生中断 高速减计数 $CV \leq SP$ (当前值=预设值) 产生中断
	3	I07 上升下降沿	I07 引脚上升沿或下降沿都产生中断
8	0	定时 0	定时中断 0 时间到产生中断
9	0	定时 1	定时中断 1 时间到产生中断
10	0	定时 2	定时中断 2 时间到产生中断
11	0	定时 3	定时中断 3 时间到产生中断
12	0	定时器 124	T124 定时时间到产生中断
13	0	定时器 125	T125 定时时间到产生中断
14	0	定时器 126	T126 定时时间到产生中断
15	0	定时器 127	T127 定时时间到产生中断

16	0	串口 1 发送	串口 1 自由通信发送指令 (XMT) 发送结束产生中断
17	0	端口 1 接收	串口 1 自由通信接收指令 (RCV) 接收到报文产生中断
18	0	串口 2 发送	串口 2 自由通信发送指令 (XMT) 发送结束产生中断
19	0	串口 2 接收	串口 2 自由通信接收指令 (RCV) 接收到报文产生中断
20	0	高速输出 1	PLS 指令 Q0.7(PEC8000)/Q0.0(PEC3000)引脚高速脉冲输出完成产生中断
21	0	高速输出 2	PLS 指令 Q0.6(PEC8000)/Q0.1(PEC3000)引脚高速脉冲输出完成产生中断

表 6.14.3 运动控制器中断事件一览表

中断号	MODE	中断事件	描述
0	0	HI0 上升沿	HI0 引脚上升沿产生中断
	1	HI0 下降沿	HI0 引脚下降沿产生中断
	2	高速计数 0	HI0 作为高速计数输入, 在单相计数模式、脉冲+方向模式、AB 相模式和正向+反向脉冲模式下, 当 CV=SP (当前值=预设值) 时, 产生中断
	3	HI0 上升下降沿	HI0 引脚上升下降沿产生中断
1	0	HI1 上升沿	HI1 引脚上升沿产生中断
	1	HI1 下降沿	HI1 引脚下降沿产生中断
	2	高速计数 1	HI1 作为高速计数输入, 在单相计数模式下, 当 CV=SP (当前值=预设值) 时, 产生中断
	3	HI1 上升下降沿	HI1 引脚上升下降沿产生中断
2	0	HI2 上升沿	HI2 引脚上升沿产生中断
	1	HI2 下降沿	HI2 引脚下降沿产生中断
	2	高速计数 2	HI2 作为高速计数输入, 在单相计数模式、脉冲+方向模式、AB 相模式和正向+反向脉冲模式下, 当 CV=SP (当前值=预设值) 时, 产生中断
	3	HI2 上升下降沿	HI2 引脚上升下降沿产生中断
3	0	HI3 上升沿	HI3 引脚上升沿产生中断
	1	HI3 下降沿	HI3 引脚下降沿产生中断
	2	高速计数 3	HI3 作为高速计数输入, 在单相计数模式下, 当 CV=SP (当前值=预设值) 时, 产生中断
	3	HI3 上升下降沿	HI3 引脚上升下降沿产生中断
4	0	HI4 上升沿	HI4 引脚上升沿产生中断
	1	HI4 下降沿	HI4 引脚下降沿产生中断
	2	高速计数 4	HI4 作为高速计数输入, 在单相计数模式、脉冲+方向模式、AB 相模式和正向+反向脉冲模式下, 当 CV=SP (当前值=预设值) 时, 产生中断
	3	HI4 上升下降沿	HI4 引脚上升下降沿产生中断
5	0	HI5 上升沿	HI5 引脚上升沿产生中断
	1	HI5 下降沿	HI5 引脚下降沿产生中断
	2	高速计数 5	HI5 作为高速计数输入, 在单相计数模式下, 当 CV=SP (当前值=预设值) 时, 产生中断
	3	HI5 上升下降沿	HI5 引脚上升下降沿产生中断
6	0	HI6 上升沿	HI6 引脚上升沿产生中断
	1	HI6 下降沿	HI6 引脚下降沿产生中断
	2	高速计数 6	HI6 作为高速计数输入, 在单相计数模式、脉冲+方向模式、AB 相模式和正向+反向脉冲模式下, 当 CV=SP (当前值=预设值) 时, 产生中断
	3	HI6 上升下降沿	HI6 引脚上升下降沿产生中断
7	0	HI7 上升沿	HI7 引脚上升沿产生中断
	1	HI7 下降沿	HI7 引脚下降沿产生中断
	2	高速计数 7	HI7 作为高速计数输入, 在单相计数模式下, 当 CV=SP (当前值=预设值) 时, 产生中断

	3	HI7 上升下降沿	HI7 引脚上升下降沿产生中断
8	0	时基匹配 0	时基匹配 0 定时时间到产生中断
9	0	时基匹配 1	时基匹配 1 定时时间到产生中断
10	0	时基匹配 2	时基匹配 2 定时时间到产生中断
11	0	时基匹配 3	时基匹配 3 定时时间到产生中断
12	0	定时器 124	T124 定时时间到产生中断
13	0	定时器 125	T125 定时时间到产生中断
14	0	定时器 126	T126 定时时间到产生中断
15	0	定时器 127	T127 定时时间到产生中断
16	0	串口 1 发送	串口 1 自由通信发送指令 (XMT) 发送结束产生中断
17	0	串口 1 接收	串口 1 自由通信接收指令 (RCV) 接收到报文产生中断
18	0	串口 2 发送	串口 2 自由通信发送指令 (XMT) 发送结束产生中断
19	0	串口 2 接收	串口 2 自由通信接收指令 (RCV) 接收到报文产生中断

● 应用示例

中断指令在工程中的应用很广泛，通过 ATCH 指令将中断服务程序 (INT_n) 与中断源 (EVENT_n) 连接起来。这样，当运行过程中产生与中断源相应的事件的时候，就会进入预先设定的中断服务程序。下面列举了不同中断类型的示例。

(1) 外部 I/O 中断示例

控制器支持有 8 个外部 I/O 口中断，其中 0 号和 1 号中断信号可用作高速计数中断，也可用作普通 I/O 中断。下面以 3 号中断事件为例说明外部 I/O 中断服务程序的使用方法。无外部 I/O 的控制器不支持此类中断。

程序功能为外部 DI3 来一个上升沿，MW100 自加 1。

首先，在子程序选项中添加中断服务程序。

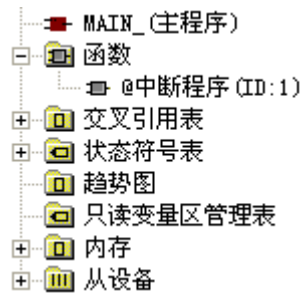


图 6.14.1 添加中断程序

在主程序中连接中断事件到中断服务程序（注意每一个中断事件只能连接一个中断服务程序），在 MW100 中记录进入中断服务程序的次数。

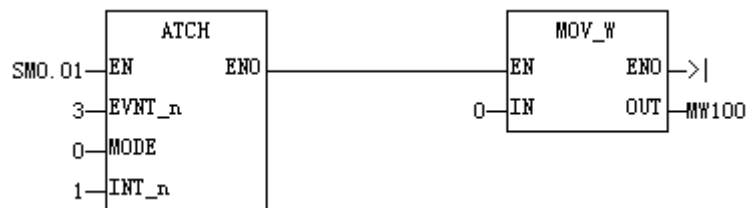


图 6.14.2 连接中断事件

打开全局中断使能标志，若没有使能该指令，则无法进入任何中断服务程序，SM0.1 表示程序第一轮扫描时运行此指令，其余时候不运行此指令。

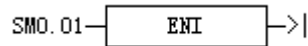


图 6.14.3 中断使能

编写中断服务程序，选中程序库中的中断程序，在其程序段 1 中添加如下指令。每进一次上升沿中断，MW100 自加 1。

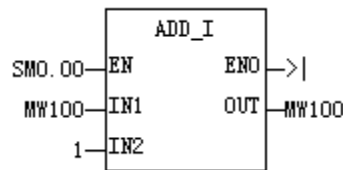


图 6.14.4 中断服务程序

(2) 定时器中断

控制器包括 4 个定时中断和 4 个定时器中断，下面以定时中断 1 为例，说明定时器中断程序使用方法。程序功能为每 1 秒钟 MW100 自加 1。

首先，添加中断服务程序（参考上例），在主程序中连接中断事件到中断服务程序。定时中断 3 的事件号为 11。

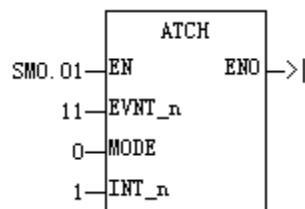


图 6.14.5 连接中断事件

使用定时中断一定要设定定时时间。定时器 3 对应的定时变量为 SMW193，单位是 10ms。若向 SMW193 中写入 100，表示定时时间为 1s。

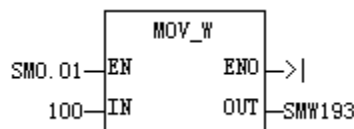


图 6.14.6 设置中断时间

打开全局中断使能标志，若没有使能该指令，则无法进入任何中断服务程序，SM0.1 表示程序第一轮扫描时运行此指令，其余时候不运行此指令。

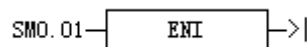


图 6.14.7 中断使能

编写中断服务程序，选中程序库中的中断程序，在其程序段 1 中添加如下指令。每进一次上升沿中断，MW100 自加 1。

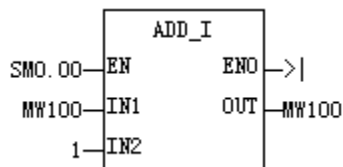


图 6.14.8 中断服务程序

(3) 自由通信中断

自由通信中断在网络通信指令章节中详细说明，在此就不加赘述。示例可参见 6.5.5 章节。

6.15 移位与循环指令

移位与循环指令主要用于变量的移位操作，实现了左移/右移，循环左移/循环右移，变量区移位、交换功能。具体包含字左移指令、双字左移指令、字右移指令、双字右移指令、字循环左移指令、双字循环左移指令、字循环右移指令、双字循环右移指令、移位寄存器指令、字节交换指令、字交换指令。

6.15.1 左移和右移

6.15.1.1 字和双字左移指令（SHL_W 和 SHL_DW）

- 指令参数

字左移指令（SHL_W）			
		实现字的左移功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	待移位的字
N	字变量、常数	WORD	移位次数，≤16 次
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	移位后的输出结果

双字左移指令（SHL_DW）			
		实现双字的左移功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	DWORD	待移位的字
N	字变量、常数	WORD	移位次数，≤32 次
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	移位后的输出结果

6.15.1.2 字和双字右移指令（SHR_W 和 SHR_DW）

- 指令参数

字右移指令（SHR_W）			
		实现字的右移功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	待移位的字
N	字变量、常数	WORD	移位次数，≤16 次
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	移位后的输出结果

双字右移指令（SHR_DW）			
实现双字的右移功能。			

参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	DWORD	待移位的双字
N	字变量、常数	WORD	移位次数, ≤16 次
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	移位后的输出结果

- 详细说明

左移/右移指令完成字和双字的逻辑移位, 逻辑移位时由零补足。

- 应用示例

如图 6.15.1 所示, 演示了逻辑左移和右移后数据的变化。程序运行后, VW0 左移 15 位, 右端补零, 结果 32768 赋给 VW1; VD10 右移 15 位, 左端补零, 结果 1 赋给 VD12; 其他同类移位指令同理。

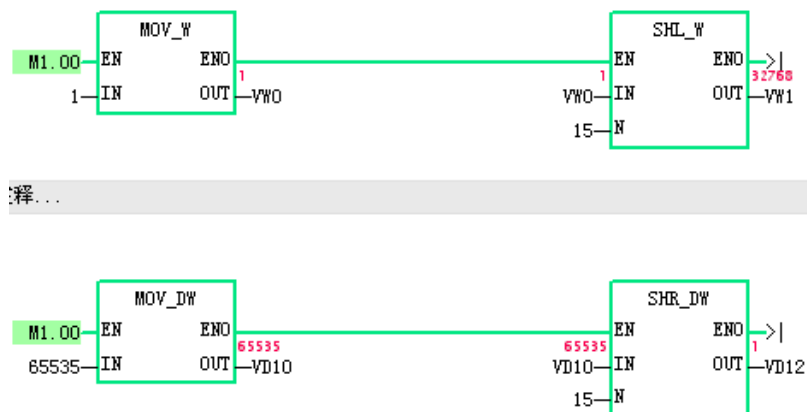


图 6.15.1 移位指令示例

6.15.2 循环左移和循环右移

6.15.2.1 字和双字循环左移指令 (ROL_W 和 ROL_DW)

- 指令参数

字循环左移指令 (ROL_W)			
实现字的循环左移功能。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	待移位的字
N	字变量、常数	WORD	移位次数, ≤16 次
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	移位后的输出结果
双字循环左移 (ROL_DW)		实现双字的循环左/右移功能。	

参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	DWORD	待移位的字
N	字变量、常数	WORD	移位次数, ≤32 次
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	移位后的输出结果

6.15.2.2 字和双字循环右移指令 (ROR_W 和 ROR_DW)

- 指令参数

字循环右移指令 (ROR_W)			
实现字的循环右移功能。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量、常数	WORD	待移位的字
N	字变量、常数	WORD	移位次数, ≤16 次
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	移动后的结果数据无符号整数
双字循环右移 (ROR_DW)			
实现双字的循环右移功能。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	双字变量、常数	DWORD	待移位的字
N	字变量、常数	WORD	移位次数, ≤32 次
ENO	位变量	BOOL	输出能量流
OUT	双字变量	DWORD	移位后的结果

- 详细说明

循环左移/右移指令完成字和双字的循环移位, 循环移位时由移出位补足。

- 应用示例

如图 6.15.2 所示, 演示了循环左移和右移后数据的变化。程序运行后, VW0 循环右移 5 位, 结果 32771 赋给 VW2; VD10 循环左移 5 位, 结果 14 赋给 VD12; 其他同类循环移位指令同理。

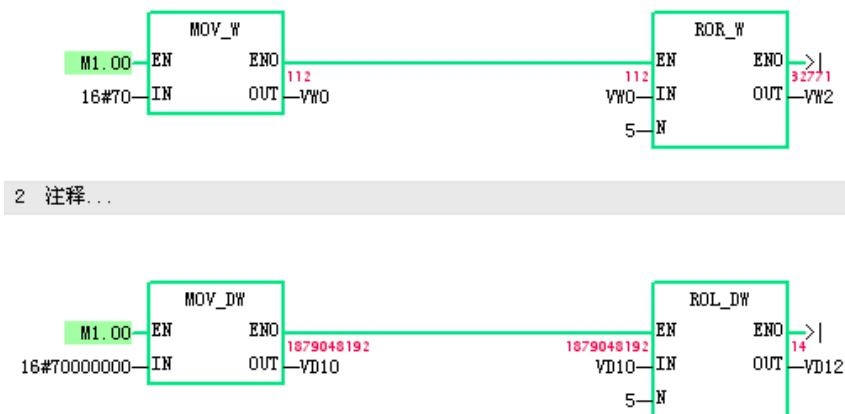


图 6.15.2 循环移位指令示例

6.15.3 移位寄存器

● 指令参数

移位寄存器 (SHRB)			
		将一个数值移入移位寄存器中。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
DATA	位变量	BOOL	待移位的数据
S_BIT	位变量	BOOL	移位寄存器的最低位
N	字变量, 常数	WORD	移位寄存器的长度和移位方向
OUT	位变量	BOOL	输出能量流

● 详细说明

使用移位寄存器指令可以通过配置移位长度和方向简化一些复杂操作。移位寄存器指令将一个数值移入移位寄存器中。每个扫描周期，整个移位寄存器移动一位。其中，S_BIT 指定移位寄存器的最低位，N 指定移位寄存器的长度和位移方向（正向移位=N，反向移位=-N）， $|N| \leq 64$ 。SHRB 指令移出的每一位都被放入溢出标志位（SM0.7）。

● 应用示例

如图 6.15.3 所示，演示了移位寄存器指令的执行过程。

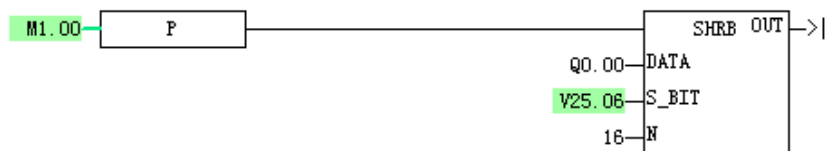


图 6.15.3 移位寄存器指令示例

(1) 初始状态，S_BIT 指向移位寄存器 V25.6，Q0.0 初始阶段值为 0， $N > 0$ ，长度为 16，数据正向移动。V25.6~V26.5 中的初始值如图 6.15.4 所示。溢出位寄存器 SM0.7 的初始值为 0。

V25.6	V25.7	V25.8													V26.5
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

图 6.15.4 移位寄存器初始值

(2) 如使能 M1.0，运行一次功移位指令，移位寄存器的值正向移动一位，结果如图 6.15.5 所示。并将

移位寄存器中 V25.4 的值赋给溢出位寄存器 SM0.7，即 SM0.7 的值由 0 变为 1。

V25.6	V25.7	V25.8													V26.5
0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

图 6.15.5 运行一次移位指令

(3) 重新使能 M1.0，移位寄存器会再次移动一位，并将溢出值放入 SM0.7 中。图 6.15.6 为执行 3 次程序后的结果。溢出位寄存器 SM0.7 的值为 1。

V25.6	V25.7	V25.8													V26.5
0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0

图 6.15.6 运行三次移位指令

(4) 将 Q0.0 的值变为 1，功能块指令再次运行 5 次，结果如图 6.15.7 所示，SM0.7 的值为 0。

V25.6	V25.7	V25.8													V26.5
1	1	1	1	1	0	0	0	0	1	0	1	0	1	0	1

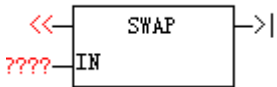
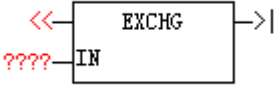
图 6.15.7 再次运行五次移位指令

当 $N < 0$ 时，与上面情况类似，移位寄存器反向移动，即从 V26.5 向左移动 16 位。

提示：建议根据需要利用上升沿指令来控制移位寄存器的移位操作。

6.15.4 交换

● 指令参数

字节交换指令（SWAP）			将 16 位字的低八位进行低 4 位与高 4 位交换，字的高 8 位清 0。
字交换指令（EXCHG）			
		将 16 位字的高八位和低八位互换。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
IN	字变量	WORD	待交换字
ENO	位变量	BOOL	输出能量流

● 详细说明

交换指令完成字节序和字序的位置交换，字节交换指令和字交换指令的输入变量都是 16 位字变量，其中，字节交换指令只对低字节进行。

● 应用示例

如图 6.15.8，程序运行后，SWAP 指令将 VW0 的低四位与高四位互换，16 进制数 1234 中的 3 和 4 互换，1 和 2 清零，运行结果为 VW0=67；EXCHG 指令将 VW1 的高八位与低八位互换，16 进制数 1234 中的 12 和 34 互换，运行结果 VW1=13330。

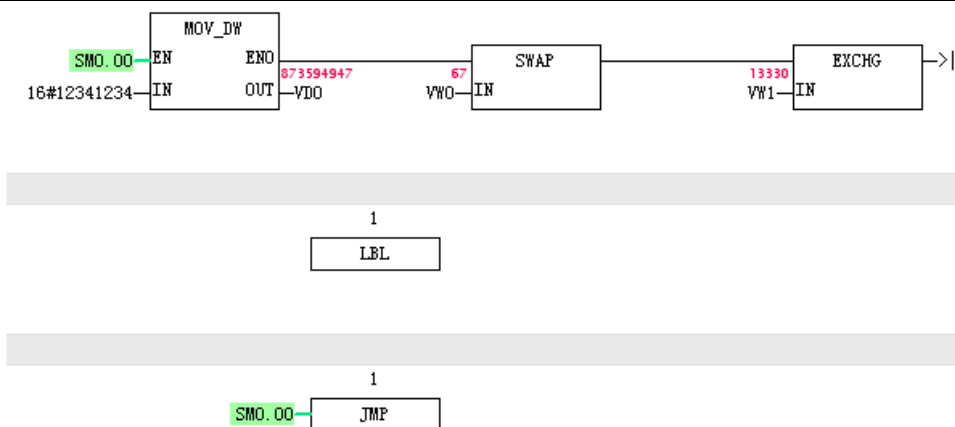


图 6.15.8 交换指令示例

6.16 定时器指令

定时器指令主要完成定时器计时相关的功能。包含了定时器接通指令，定时器断开指令，保持定时接通指令。

每个定时器都有一个 16 位的字和一个二进制位，定时器的字用来存储它的当前计时数值，定时器接通的状态由它的位状态决定。用定时器地址（T 和定时器号，例如 T10）来访问当前计时数值和定时器位。

具体说明如下：

- (1) 所有定时器(T0~T127)均可以配置分为 3 种定时器指令，通过标号区分。
- (2) 所有定时器均以 100ms 为时基，其中当 SM59.0 配置为 1 时，T80~T119 的时基缩短为 10ms。
- (3) T124、T125、T126、T127 支持定时器中断。
- (4) 定时器符号 T 在作为位操作时为定时器到达目标值的标志位，作为字操作时为当前计时数值，使用时会自动进行区分。

表 6.16.1 定时器兼容标志

SM 区	描 述
SM59.0	定时器兼容标志，默认为 0 (1) 为 1: T80~T119 时基为 10ms T752~T1007 时基为 10ms T1008~T1023 时基为 1ms T0~T79 时基为 100ms T120~T751 时基为 100ms (2) 为 0: T0~T1007 时基为 100ms T1008~T1023 时基为 1ms

6.16.1 定时接通指令 (TON)

● 指令参数

定时器接通指令 (TON)		实现定时接通功能，在脉冲输入信号 IN 为高电平时，定时器开始计数，当到达延时时间 PT 时，定时器输出高电平。	
参数名	变量区	数据类型	说明
IN	位变量	BOOL	开关量输入
PT	字变量、常数	INT	延时时间
????	T 区	INT	定时器的有效范围是 T0~T1023

● 详细说明

定时器接通指令在脉冲输入信号 IN 为高电平时，定时器开始计数，当到达延时时间 PT 时，定时器输出高电平。当 IN 输入低电平时，定时器计数值清零且输出低电平。当 PT 为 0 时定时器立即接通，定时器位标

志输出高电平。PT 为延时时间，其单位与所选择的定时器时基有关。

- 应用示例

如图 6.16.1 所示，给出了定时器配置的具体例子。在 T1 时刻 I0.0 输入高电平，定时器启动，开始计数。在 T2 时刻 I0.0 输入低电平时定时器停止工作并清空计数值。由于在 T2 时刻计数值没有达到 PT 设定值，所以 Q0.0 没有输出。在 T3 时刻 I0.0 输入高电平，定时器启动，经过 t5 的时间计数值与 PT 值相等，Q0.0 输出高电平。在 T4 时刻 I0.0 输入低电平，定时器复位，Q0.0 复位。

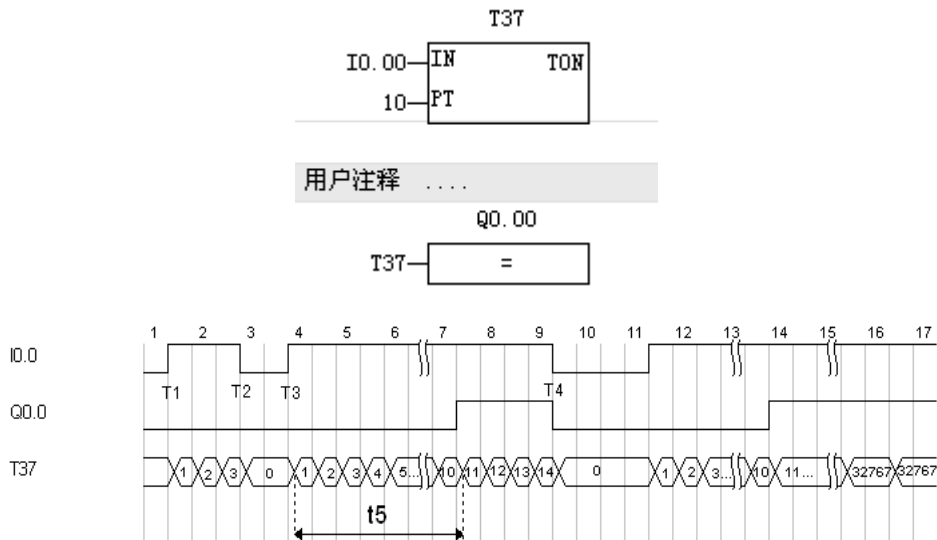


图 6.16.1 接通延时定时器示例

6.16.2 定时断开指令 (TOF)

- 指令参数

定时断开指令 (TOF)			
		实现定时断开功能，当 IN 为高电平时，定时器输出高电平。当 IN 输入低电平时，定时器启动计数，当到达设定时间 PT，定时器输出低电平。	
参数名	变量区	数据类型	说明
IN	位变量	BOOL	开关量输入
PT	字变量、常数	INT	延时时间
???	T 区	INT	定时器有效范围是 T0~T1023

- 详细说明

定时器断开指令在脉冲输入信号 IN 为高电平时，定时器输出保持高电平。当 IN 输入低电平时，定时器启动计数，当到达设定时间 PT，定时器输出低电平。当 PT 为 0 时定时器立即断开，定时器位标志输出低电平。PT 为延时时间，其单位与所选择的定时器时基有关。

- 应用示例

如图 6.16.2 所示，给出了定时器配置的具体例子。在 T1 时刻 I0.00 输入高电平，Q0.00 输出高电平；在 T2 时刻 I0.00 输入低电平，定时器启动计数；在 T3 时刻 I0.00 再次输入高电平时，定时器复位，T4 时刻定时器再次计数，经过 t5 的时间计数值与 PT 值相等，Q0.0 输出低电平。

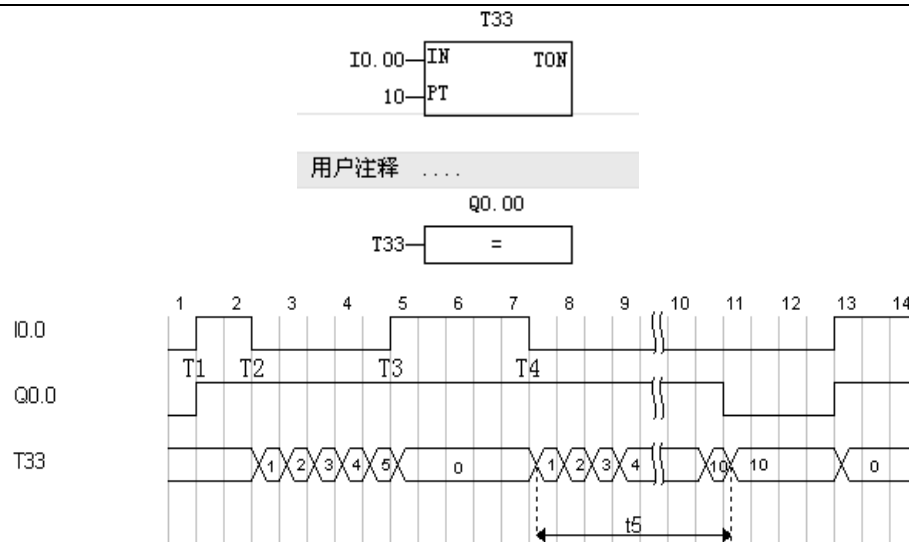


图 6.16.2 断开延时定时器示例

6.16.3 保持定时接通指令 (TONR)

● 指令参数

保持定时接通指令 (TONR)			
		实现保持定时接通功能，当 IN 输入高电平时定时器开始计数，当 IN 输入低电平时定时器停止计数并保值当前计数值。当计数值等于 PT 时，定时器输出。直到用户配置其复位。	
参数名	变量区	数据类型	说明
IN	位变量	BOOL	开关量输入
PT	字变量、常数	INT	延时时间
???	T 区	INT	定时器范围是 T0~T1023

● 详细说明

保持定时接通指令与定时接通指令相比，其输入引脚 IN 输入低电平时，定时器输出低电平，计时数值不会被清零，当重新使能 IN 时定时器的计数值可继续累加计时。

● 应用示例

如图 6.16.3 所示，在 T1 时刻 I0.0 输入高电平时定时器启动，在 T2 时刻 I0.0 输入低电平时定时器停止并保持当前计数值。T3 时刻定时器启动。当计数值（T6）等于 PT 值时 Q0.0 输出高电平。在 T4 时刻 Q0.0 输出低电平并且定时器计数值复位。在 T5 时刻定时器继续计数。

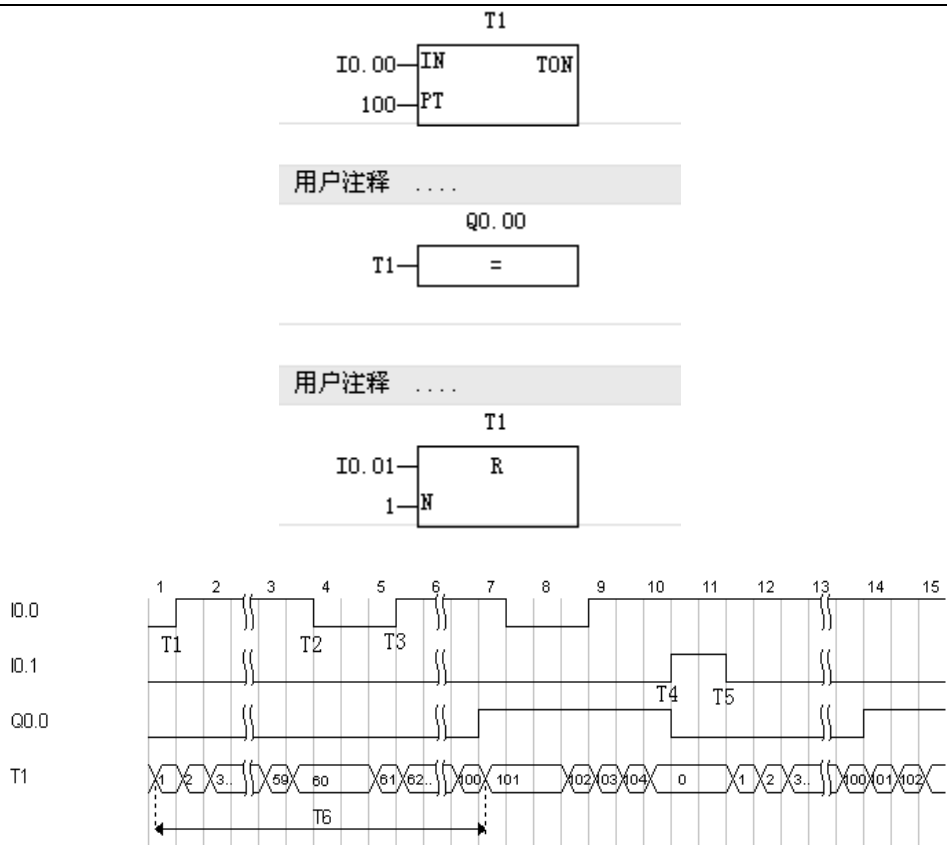


图 6.16.3 有记忆接通延时定时器示例

6.17 专用指令

专用指令提供了工业控制领域常用的一些指令和新功能。用户可以使用专用指令实现一些复杂的算法。目前支持的专用指令包括：工件检测、数据校验、经济运行、存储、冒泡排序、斜率检测、定位、往复运动、PH 检测、电导率检测、溶氧检测和 Na 离子检测等特殊指令。

6.17.1 工件检测

● 指令参数

工件检测指令（WPC） ??.			
WPC <<EN END-> <<IC LS-??. <<OC BLOCK-??. ????INTVL NUM-???? ????S_TIME ????R ????MN ????LN ????AREA			
检测工件在生产线上是否丢失或出现阻塞现象。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	指令使能位
IC	位变量	BOOL	检测工件是否进入生产线
OC	位变量	BOOL	检测处理完毕的工件是否退出生产线
INTVL	字变量、常数	WORD	工件在生产线上的总停留时间，单位为秒
S_TIME	字变量、常数	WORD	工件进入或退出时的检测时间，单位为秒

R	位变量	BOOL	指令复位参数
MN	字变量、常数	WORD	生产线上最多停留的工件数，最大值 511
LN	字变量、常数	WORD	连续丢失的最大工件数量
AREA	字变量、常数	WORD	工件退出或丢失的微调时间，单位为秒
ENO	位变量	BOOL	输出能量流
LS	位变量	BOOL	工件是否丢失输出标志
BLOCK	位变量	BOOL	工件是否阻塞输出标志
NUM	字变量	WORD	记录总的工件丢失个数
???.?	位变量	BOOL	指令状态指示参数

● 详细说明

WPC 指令用于检测工件传输装置中的工件丢失或阻塞情况，通过参数配置能够指示出当前传输装置的工作状态，有时间判断和距离判断两种模式，可通过 SMW48 设置，该功能块参数说明如下：

工件进入检测信号 IC(In Check)用于检测工件是否进入生产线，当 IC 信号检测到工件进入信号并持续 S_TIME（单位为秒）时间时，认为此工件进入生产线。

INTVL(Interval)参数为工件在生产线上的总停留时间或设定的生产线长度，AREA 用于对此参数进行微调。时间判断模式下，二者单位均为秒；距离判断模式下与设定的速度单位相同，如速度为 2 厘米/秒，则二者单位均为厘米。

工件退出检测信号 OC(Out Check)用于检测处理完毕的工件是否退出生产线。时间模式下，当 OC 信号检测到工件退出信号，持续了 S_TIME 时间，且此工件在生产线上停留的总时间大于(INTVL-AREA)时，认为此工件正常退出生产线。当检测到此工件在生产线上停留时间大于(INTVL+AREA)时，认为此工件丢失。距离模式下当检测到此工件运行距离大于 INTVL+AREA 时则认为丢失。

MN(Max Number)为生产线上最多停留的工件数量、最大值为 512。

当出现工件丢失情况时，LS (Lost State) 输出为 1，直到功能块复位或功能块不使能，LS 输出为 0。

LN(Lost Number)为连续丢失的最大工件数量。当连续丢失的工件数量大于 LN 时，阻塞参数 BLOCK 输出为 1，直到指令复位或重新有工件退出。

NUM(Number)用于记录总的工件丢失个数，直到指令复位。

R(Reset)参数为功能块复位参数，此位为 1 时功能块输出参数 LS、BLOCK 和 NUM 复位为 0。

???.?为功能块状态指示参数，为 1 表示功能块运行正常，为 0 表示功能块内部运行错误，此时 MN 为 0 或内部索引值越界。

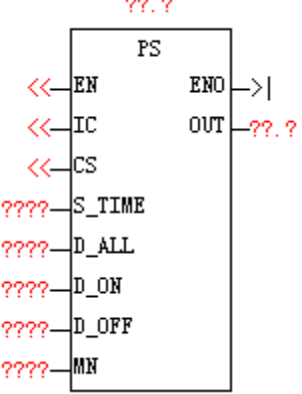
表 6.17.1 WPC 指令标识

SM 区	描 述
SMW48	WPC 指令使用距离判断模式标识，第一位对应第一个 WPC 指令，第二位对应第二个 WPC 指令，其余位保留，相应位为 1 启用距离判断模式，为 0 使用时间判断模式，默认为 0
SMD49	浮点数，第一个 WPC 指令使用距离判断模式时的速度值，单位和 INTVL 一致
SMD65	浮点数，第二个 WPC 指令使用距离判断模式时的速度值，单位和 INTVL 一致

提示：本指令最多支持 2 个，且通过???.?头顶参数区分，头顶参数必须为 L15.8 或 L15.9，否则指令将不被执行。

6.17.2 经济运行

● 指令参数

经济运行指令 (PS)			
 检测工件在生产线上的位置，自动控制是否启动锡炉。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	指令使能位
IC	位变量	BOOL	工件进入检测信号
CS	双字变量、常数	REAL	链条速度，以秒为时间单位
S_TIME	字变量、常数	WORD	工件检测时间，用于输入防抖处理
D_ALL	双字变量、常数	REAL	工件检测孔到锡炉的距离
D_ON	双字变量、常数	REAL	波峰启动点到锡炉的距离
D_OFF	双字变量、常数	REAL	锡炉到波峰关闭点的距离
MN	字变量、常数	WORD	生产线上工件的最大数量
ENO	位变量	BOOL	输出能量流
OUT	位变量	BOOL	波峰开启与关闭标志
???	位变量	BOOL	头顶参数，指示功能块在工作状态是否运行正常

● 详细说明

PS 指令用于检测工件在生产线上的位置，自动控制是否启动锡炉，用于节省能源。

EN 为指令使能位，为 1 则使能指令运行，0 则禁止指令运行。当禁止指令运行后，将清除输出位 OUT。

S_TIME 为工件检测时间，单位为 100ms，用于输入防抖处理。

IC(In Check)为工件进入检测信号，用于检测工件是否进入生产线。当 IC 信号检测到工件进入信号并持续 S_TIME 的时间时，认为此工件进入生产线。

D_ALL(Distance All)为工件检测孔到锡炉的距离。

D_ON(Distance ON)为波峰启动点到锡炉的距离。

D_OFF(Distance OFF)锡炉到波峰关闭点的距离。

CS(Chain Speed)为链条速度，以秒为时间单位，用于和设定的 D_ALL、D_ON、D_OFF 等距离参数一起，判断波峰的开启与关闭，四者长度单位应保持一致，例如 CS 的单位为 1.6 厘米/秒，则 D_All、D_ON、D_OFF 的单位应该为厘米。进入锡炉前，当运行的工件与锡炉的距离小于 D_ON 时，启动波峰。进入锡炉后，当运行的工件与锡炉的距离大于 D_OFF，且下一个未到达锡炉的运行工件与锡炉的距离大于 D_ON 时，波峰关闭。

MN(Max Number)为生产线上工件的最大数量，用于程序内部处理队列数据。最大值为 255。

OUT 为波峰开启的控制位，输出为 1 时波峰开启，输出为 0 时波峰关闭。

???.?为头顶参数，指示功能块在工作状态是否运行正常。1 为正常，0 为异常。当生产线上工件最大数量 MN 为 0、链速 CS≤0、内部队列索引值越界或寻址错误时为异常状态。

检测示意图如图 6.17.1 所示。

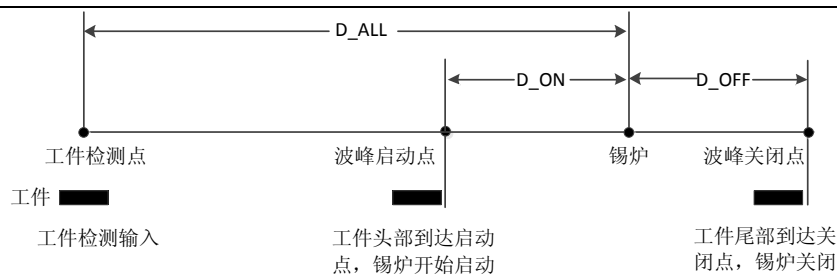


图 6.17.1 工件检测示例

提示：本指令最多支持 2 个，且通过???.?头顶参数区分，头顶参数必须为 L15.0 或 L15.1，否则指令将不被执行。

6.17.3 存储

● 指令参数

存储指令 (STORE)			
STORE << EN ENO > << RESET OUT <??? ???? REG ???? NUM ???? INTVAL ???.? T_FLAG 存储用户数据。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能控制位
RESET	位变量	BOOL	复位信号
REG	字变量	WORD	所要存储数据的变量起始地址
NUM	字变量、常数	WORD	要存储的变量数量
INTVAL	字变量、常数	WORD	需要存储的时间间隔
T_FLAG	位变量	BOOL	是否需要存储时间戳标志
ENO	位变量	BOOL	输出能量流
OUT	字变量	WORD	当前存储区域的变量索引值

● 详细说明

STORE 指令用于存储用户需要暂时保留的数据，将其保存在 RAM 中，利用电池可保证数据不丢失，此存储区域共分配 20K 字节共 10000 个变量空间，其中第 10000 个地址存储的是当前存储区已经使用的变量数量 R_NUM。

REG 为需要存储的区域变量起始地址，NUM 为需要存储的变量数量，INTVL 为每隔多长时间进行一次存储操作，单位为秒，时间戳标志 T_FLAG 为 0，例如如果 REG 填入 VW100，NUM 为 10，INTVL 为 60，则 EN 为 1 时运行此指令，并且每隔 60 秒钟将 VW100~VW109 的数据存入所分配的 RAM 区域，存储时以字为单位，只存储数据不存储时间戳，存储顺序如下：

表 6.17.2 第一次存储变量表

地址	20000		20009	20010		200119	20020	...	29999
数据	VW100		VW109	VW100		VW109	VW100	...	R_NUM
	第一次存储			第二次存储			...		

每次存储的数据按照存储的容量依次往下排列，其中地址一项为开放给用户的 Modbus 变量地址，只允许利用 ModbusRTU 和 ModbusTCP/UDP 协议的 3 号命令进行读取，地址范围为 20000~29999，共 10000 个变量。

如果时间戳标志 T_FLAG 为 1，则存储时会加入时间戳，存储时以字为单位，存储顺序如下：

表 6.17.3 第二次存储变量表

地址	20000	20001	20002	20003	20004	20005	20006		20016		29999
数据	实时 时钟 年	实时 时钟 月	实时 时钟 日	实时 时钟 时	实时 时钟 分	实时 时钟 秒	VW10 0		VW109		R_NUM
	第一次存储										

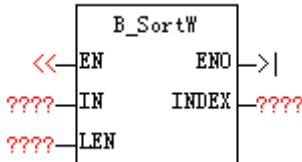
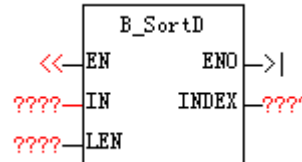
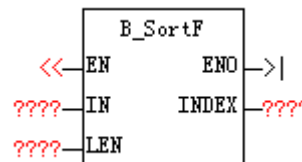
第二次存储时则从地址 20017 开始存储第二次存储时的实时时钟和 VW100~VW109 的数据。

OUT 值显示当前变量的索引值。为 0~9999 之间的数值，为 0 时表示存储区域全部空闲，为 9999 或当前 OUT+NUM>10000 时表示存储区域已全部存满，无法填入新数据。

当 RESET 为 1 时则清除所有保存的数据，并将当前变量索引值归 0。

6.17.4 冒泡排序

● 指令参数

字冒泡排序指令（B_SortW）		 实现多个字的排序功能，顺序号依次放入 INDEX 中。			
参数名	变量区			数据类型	说明
EN	位变量			BOOL	使能位，高电平触发
IN	字变量			WORD	排序字的起始地址
LEN	常数			WORD	排序个数
ENO	位变量	BOOL	输出能量流		
INDEX	字变量	WORD	结果数据在原顺序下的索引		
双字冒泡排序指令（B_SortD）		 实现多个双字的排序功能，顺序号依次放入 INDEX 中。			
操作数	变量区			数据类型	说明
EN	位变量			BOOL	使能位，高电平触发
IN	双字变量			DWORD	排序字的起始地址
LEN	常数			WORD	排序个数
ENO	位变量	BOOL	输出能量流		
INDEX	字变量	WORD	结果数据在原顺序下的索引		
实数冒泡排序指令（B_SortF）		 实现多个实数的排序功能，顺序号依次放入 INDEX 中。			
参数名	变量区			数据类型	说明
EN	位变量			BOOL	使能位，高电平触发
IN	双字变量			REAL	排序字的起始地址
LEN	常数			WORD	排序个数

ENO	位变量	BOOL	输出能量流
INDEX	字变量	WORD	结果数据在原顺序下的索引

- 详细说明

冒泡排序在排出顺序的同时会给出数据原位置的索引。

- 应用示例

如图 6.17.2, 向 VW100~VW109 中写入原数据串为 159, 42, 56, 212, 118, 33, 71, 0, 189, 169。使能 M1.00 后, 执行 B_SortW 指令, 对 VW100~VW109 中的数据进行排序, 排序后的结果为 0, 33, 42, 56, 71, 118, 159, 169, 189, 212。INDEX 中存储结果数据在原数据串中的位置, 运行后, VW200~VW209 的值为 7, 5, 1, 2, 6, 4, 0, 9, 8, 3。例如, VW100 中的数据是 0, 在原数据串中的位置是 7 (以 0 位置开始), 所以 VW200 为 7; VW104 中的数据是 71, 在原数据串中的位置是 6, 所以 VW204 为 6。



图 6.17.2 冒泡排序指令示例

提示：由于冒泡排序指令每次执行会改变原数据队列的数据排列, 建议使用上升沿来控制本指令的执行, 这样才能每次获得正确的输出索引。

6.17.5 数据校验

6.17.5.1 数据校验码生成指令 (CKSUM_G)

- 指令参数

数据校验码生成指令 (CKSUM_G)			
		计算从 IN 开始的 N 个字节(字)数据的校验码。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位, 高电平触发
IN	字变量	WORD	待校验数据首地址
N	字变量、常量	WORD	校验数据长度
TYPE	字变量	WORD	校验的类型 (0~5)
OUT	字变量	WORD	校验结果值
ENO	位变量	BOOL	输出能量流

- 详细说明

数据校验码生成指令根据 IN 指定的资源起始位置, 计算 N 个 (最多支持 512 字节数据校验) 字节(字)数据的校验码, 并根据指定的数据校验类型将数据校验码存入 OUT 指定的变量中。

校验码生成指令在处理数据时分为字节和字两种单位, 当以字节为单位时, CKSUM_G 指令在处理数据时使用每个变量的低 8 位, 并将生成的校验码放入两个连续字变量的低 8 位。

当以字为单位时, 取数据的方式为高字节在前, 低字节在后的顺序, CKSUM_G 指令在处理数据时读取每个字变量中的数据, 并将生成的校验码放入一个字变量中。(当进行 BCC、LRC 校验时, 校验结果将放入字变量的高字节, 例如 BCC 校验结果为 74、那么结果变量中存放的数据为 7400)

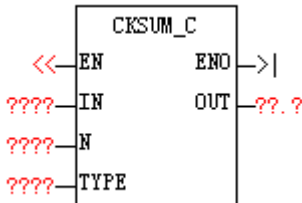
其中:

TYPE = 0, 字节校验, 表示生成 CRC16 校验码;

TYPE = 1, 字节校验, 表示生成 BCC 校验码;
 TYPE = 2, 字节校验, 表示生成 LRC 校验码;
 TYPE = 3, 字校验, 表示生成 CRC16 校验码;
 TYPE = 4, 字校验, 表示生成 BCC 校验码;
 TYPE = 5, 字校验, 表示生成 LRC 校验码。

6.17.5.2 数据校验码检查指令 (CKSUM_C)

● 指令参数

数据校验码检查指令(CKSUM_C)			
		根据不同的数据校验算法, 检验数据的有效性。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位, 高电平触发
IN	字变量	WORD	待校验数据的首地址
N	字变量, 常量	WORD	校验数据的长度
TYPE	字变量	WORD	校验的类型 (0~5)
OUT	位变量	BOOL	校验结果
ENO	位变量	BOOL	输出能量流

● 详细说明

数据校验码检查指令 CKSUM_C, 能够根据不同的数据校验算法对数据的有效性进行检验。目前数据校验码检查指令支持的校验算法有 CRC16, LRC, BCC 等。

校验码检查指令在处理数据时分为字节和字为单位, 当以字节为单位时, CKSUM_C 对从 IN 开始的 N 个字变量的低 8 位进行数据校验, 并将校验的结果放入用户指定的位变量中。

当以字为单位时, CKSUM_C 从对 IN 开始的 N 个字变量的字数据进行数据校验, 并将校验的结果放入用户指定的位变量中。

● 应用示例

示例 1: 字节数据校验

如图 6.17.3, 数据校验码生成指令计算从 IN 开始的 N 个 (最多支持 512 字节数据校验) 字节数据的校验码。数据校验码检查指令根据指定的数据校验算法, 检验给定的数据是否正确。

利用 CKSUM_G 指令对 MW0~MW3 中的数据进行校验, 并通过 CKSUM_C 指令获取 MW0~MW5 中的低 8 位数据, 检查数据的有效性。

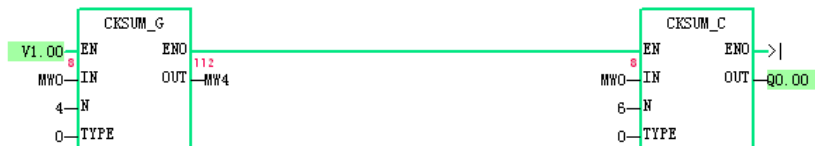


图 6.17.3 字节数据校验指令示例

注意这里 CKSUM_C 指令的 N 为 6, 包括 MW4 和 MW5 中的校验码, 将校验的结果放入 Q0.0 中。

如图 6.17.4, 对数据进行 CRC 字节校验时, 对 MW0~MW3 中的数据进行 CRC 校验, 取低八位数据, 忽略高八位数据, 其数据的校验顺序为 08、07、06、05, 其 CRC 校验结果为 0x7026, MW4 中存放的数据为 0x70, MW5 中存放的数据为 0x26。

MW0	2848		有符号双字节	8	8
MW1	2849		有符号双字节	7	7
MW2	2850		有符号双字节	6	6
MW3	2851		有符号双字节	5	5
MW4	2852		无符号双字节	112	70
MW5	2853		无符号双字节	38	26
Q0.0	16704		位变量	1	1

图 6.17.4 CRC 字节校验结果

如图 6.17.5，对数据进行 BCC 字节校验时，对 MW0~MW3 中的数据进行 BCC 校验，取低八位数据，忽略高八位数据，其数据的校验顺序为 08、07、06、05，其 BCC 校验结果为 0x0C，MW4 中存放的数据为 0x0C，MW5 中存放的数据为 0x00。

MW0	2848		有符号双字节	8	8
MW1	2849		有符号双字节	7	7
MW2	2850		有符号双字节	6	6
MW3	2851		有符号双字节	5	5
MW4	2852		无符号双字节	12	C
MW5	2853		无符号双字节	0	0
Q0.0	16704		位变量	1	1

图 6.17.5 BCC 字节校验结果

如图 6.17.6，对数据进行 LRC 字节校验时，对 MW0~MW3 中的数据进行 LRC 校验，取低八位数据，忽略高八位数据，其数据的校验顺序为 08、07、06、05，其 LRC 校验结果为 0xE6，MW4 中存放的数据为 0xE6，MW5 中存放的数据为 0x00。

MW0	2848		有符号双字节	8	8
MW1	2849		有符号双字节	7	7
MW2	2850		有符号双字节	6	6
MW3	2851		有符号双字节	5	5
MW4	2852		无符号双字节	230	E6
MW5	2853		无符号双字节	0	0
Q0.0	16704		位变量	1	1

图 6.17.6 LRC 字节校验结果

示例 2：字数据校验

对 16 进制数据 0x4708, 0x1207, 0x6906, 0x5405 进行数据校验可以通过如下过程实现：

如图 6.17.7，使用 CKSUM_G 指令对存放在数据区的数据进行校验，CKSUM_G 功能块使用 CRC16 方式（TYPE 引脚为 3，如需使用其他校验方法，改变 TYPE 即可）计算从 MW0 开始的 4 个字数据的校验码，并将校验结果放入在 MW4 中。数据校验码检查指令根据指定的数据校验算法，检验给定的数据是否正确。注意这里的 N 为 5，包括 MW4 的校验码，将校验的结果放入 Q0.0 中。

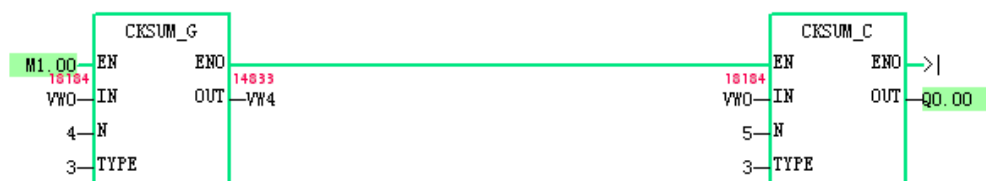


图 6.17.7 字数据校验指令示例

如图 6.17.8，对数据进行 CRC 字校验时，对 MW0~MW3 中的数据进行 CRC 校验，按高字节在前、低字节在后的顺序读取数据，其数据的校验顺序为 47、08、12、07、69、06、54、05，其 CRC 校验结果为 39F1，存放在 VW4 中。

VW0	0x0920 (2336)	有符号双字节	18184	4708	18184
VW1	0x0921 (2337)	有符号双字节	4615	1207	4615
VW2	0x0922 (2338)	有符号双字节	26886	6906	26886
VW3	0x0923 (2339)	有符号双字节	21509	5405	21509
VW4	0x0924 (2340)	有符号双字节	14833	39F1	
Q0.00	0x4140 (16704)	位变量	1	1	

图 6.17.8 CRC 字校验结果

如图 6.17.9，对数据进行 BCC 字校验时，对 VW0~VW3 中的数据进行 BCC 校验，按高字节在前、低字节在后的顺序读取数据，其数据的校验顺序为 47、08、12、07、69、06、54、05，其 BCC 校验结果为 64，

结果将存入变量区的高字节，即 VW4 中的数值为 0x6400。

VW0	0x0920 (2336)	有符号双字节	18184	4708	18184
VW1	0x0921 (2337)	有符号双字节	4615	1207	4615
VW2	0x0922 (2338)	有符号双字节	26886	6906	26886
VW3	0x0923 (2339)	有符号双字节	21509	5405	21509
VW4	0x0924 (2340)	有符号双字节	25600	6400	
Q0.00	0x4140 (16704)	位变量	1	1	

图 6.17.9 LRC 字校验结果

➤ 校验原理说明

(1) CRC 校验：

CRC(Cyclic Redundancy Check)既循环冗余码检验，CRC 校验的基本思想是利用线性编码理论，在发送端根据要传送的 k 位二进制码序列，以一定的规则产生一个校验用的监督码（既 CRC 码）r 位，并附在信息后边，构成一个新的二进制码序列数共(k+r)位，最后发送出去。在接收端，则根据信息码和 CRC 码之间所遵循的规则进行检验，以确定传送中是否出错。

16 位的 CRC 码产生的规则是先将要发送的二进制序列数左移 16 位（既乘以 2^{16} ）后，再除以一个多项式，所得到的余数既是 CRC 码，如式 6.17.1 所示，其中 B(X)表示 n 位的二进制序列数，G(X)为多项式，Q(X)为整数，R(X)是余数（既 CRC 码）。

$$\frac{B(x) \times 2^{16}}{G(x)} = Q(x) + \frac{R(x)}{G(x)} \quad (\text{式 6.17.1})$$

求 CRC 码采用模 2 加减运算法则，既不带进位和借位的按位加减，这种加减运算实际上就是逻辑上的异或运算，加法和减法等价，乘法和除法运算与普通代数式的乘除法运算一样，符合同样的规律。生成 CRC 码的多项式如下，其中 CRC-16 和 CRC-CCITT 产生 16 位的 CRC 码，而 CRC-32 则产生的是 32 位的 CRC 码。本文仅讨论 CRC-16 算法。

在 CRC-16 算法中，G(x) 如下所示。

$$G(x) = x^{16} + x^{15} + x^2 + 1$$

$$G(x) = x^{16} + x^{15} + 1 \quad (\text{式 6.17.2})$$

接收方将接收到的二进制序列数（包括信息码和 CRC 码）除以多项式，如果余数为 0，则说明传输中无错误发生，否则说明传输有误，关于其原理这里不再多述。

对于一个二进制序列数可以表示为：

$$B(x) = B_n \cdot 2^n + B_{n-1} \cdot 2^{n-1} + \dots + B_1 \cdot 2 + B_0 \quad (\text{式 6.17.3})$$

求此二进制序列数的 CRC 码时，先乘 2^{16} （既左移 16 位），再除以多项式 G(X)，所得的余数既是所要求的 CRC 码

$$\frac{B(x) \cdot 2^{16}}{G(x)} = \frac{B_n \cdot 2^{16}}{G(x)} \cdot 2^n + \frac{B_{n-1} \cdot 2^{16}}{G(x)} \cdot 2^{n-1} + \dots + \frac{B_1 \cdot 2^{16}}{G(x)} \cdot 2 + \frac{B_0 \cdot 2^{16}}{G(x)} \quad (\text{式:6.17.4})$$

可以设：

$$\frac{B(x) \cdot 2^{16}}{G(x)} = Q_n(x) + \frac{R_n(x)}{G(x)} \quad (\text{式 6.17.5})$$

其中， $Q_n(x)$ 为整数， $R_n(x)$ 为 16 位余数、将式 6.17.5 带入式 6.17.4 得：

$$\frac{B(x) \cdot 2^{16}}{G(x)} = Q_n(x) \cdot 2^n + \left[\frac{B_{n-1} \cdot 2^{16}}{G(x)} + 2 \cdot \frac{R_n(x)}{G(x)} \right] \cdot 2^{n-1} + \dots + \frac{B_1 \cdot 2^{16}}{G(x)} \cdot 2 + \frac{B_0 \cdot 2^{16}}{G(x)} \quad (\text{式 6.17.6})$$

再设：

$$\left[\frac{B_{n-1} \cdot 2^{16}}{G(x)} + 2 \cdot \frac{R_n(x)}{G(x)} \right] = Q_{n-1}(x) + \frac{R_{n-1}(x)}{G(x)} \quad (\text{式 6.17.7})$$

其中 $Q_{n-1}(x)$ 为整数, $R_{n-1}(x)$ 为 16 位二进制余数, 将式 6.17.7 带入式 6.17.6, 如上类推, 最后得到

$$\frac{B(x) \cdot 2^{16}}{G(x)} = Q_n(x) \cdot 2^n + Q_{n-1}(x) \cdot 2^{n-1} + \cdots + Q_0(x) \quad (\text{式 6.17.8})$$

根据 CRC 的定义, 十六位二进制数 $R_0(x)$ 既是 CRC 校验码。

(2) LRC 校验

Longitudinal Redundancy Check (LRC) 校验为一个字节的 2 进制数值。生成校验码时, 按字节累加待校验数据并且忽略第 9 比特进位, 将累加所得结果按位取反, 所得的结果即为 LRC 校验码。

在进行校验时, 按字节累加待校验数据以及 8 比特校验码, 忽略第 9 比特进位, 若所得结果为 0, 则校验成功; 若非 0, 则 LRC 校验失败。

(3) BCC 校验

BCC 校验又称为异或校验, 校验码为 8 比特的 2 进制数据。算法的核心思想为将待校验数据按字节异或生成校验码。

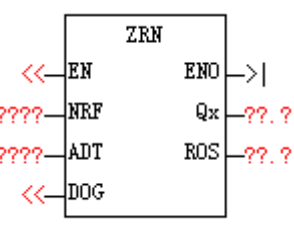
进行数据校验时, 按字节异或待校验数据和 8 比特的校验码, 若计算所得结果为 0, 则校验成功; 若干结果非 0, 则 BCC 校验失败。

6.17.6 定位

V3000 版本以上产品支持此指令。

6.17.6.1 原点回归指令 (ZRN)

● 指令参数

原点回归指令 (ZRN)			
		定位系统回归原点。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位, 高电平触发
NRF	双字变量、常数	DWORD	原点回归的起始运行频率, 单位为 Hz
ADT	双字变量、常数	DWORD	启动时的加速脉冲及当 DQG 为 1 时减速停止的脉冲个数
DOG	I 区位变量	BOOL	近点信号, 由 0 变为 1 时, 高速 Qx 减速至停止
ENO	位变量	BOOL	输出能量流
Qx	PEC8000 的 Q0.6、Q0.7 , DIO3000/PEC3000 的 Q0.0、Q0.1	BOOL	DQ 输出选择, 必须填写 Q0.6 或 Q0.7(PEC8000), Q0.0 或 Q0.1(PEC3000/DIO3000), 其余不支持
ROS	位变量	BOOL	指令执行结束标志

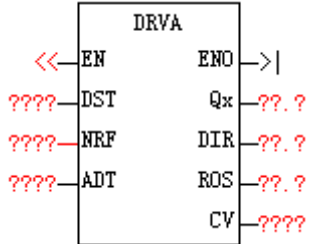
● 详细说明

ZRN 指令用于原点回归, 当执行绝对定位及相对定位指令后, 功能块要对其当前的坐标进行增减, 并将其保存到坐标变量区。当系统断电, 此信息也即随之消失, 所以每次上电要运行原点回归指令, 系统的原点与机械原点相对应。

系统在回归原点的过程中, 首先以 S 曲线模式加速到 NRF 频率, 接着以 NRF 在 Qx 上进行脉冲输出, 进行原点回归运动, 当 DQG 信号由 0 改变为 1 时, 说明此时已接近原点, 则 Qx 通过 S 曲线模式进行减速直至停止, 功能块结束运行, ROS 输出 1 表示脉冲输出完成。这里的 ADT 是 S 曲线加减速的脉冲个数。另外, 对于 S 曲线的一些参数, 可以通过 SM 区进行设置, 详见 SM 区说明文档。

6.17.6.2 绝对定位指令 (DRVA)

● 指令参数

绝对定位指令 (DRVA)			
		实现定位系统的绝对定位功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位, 高电平触发
DST	双字变量、常数	DWORD	目标坐标值
NRF	双字变量、常数	DWORD	运行频率, 单位为 Hz
ADT	双字变量、常数	DWORD	脉冲输出的开始阶段和结束阶段加减速脉冲个数
ENO	位变量	BOOL	能流输出
Qx	Q 区位变量	BOOL	脉冲输出, 必须填写 Q0.6 或 Q0.7(PEC8000), Q0.0 或 Q0.1(PEC3000), 其余不支持
DIR	Q 区的位变量	BOOL	方向输出, DST 为正数时输出 1, 为负数时输出 0
ROS	位变量	BOOL	指令执行结束标志
CV	双字变量、常数	DWORD	指令当前已输出脉冲个数

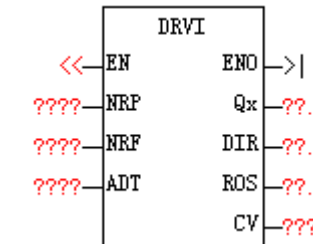
● 详细说明

DRVA 指令完成的功能是在高速 Qx 上以 NRF 的频率输出脉冲, 并且会根据方向位 DIR 改变高速 Qx 对应的坐标变量数值, 当前坐标等于 DST 设置的坐标时, 脉冲输出结束。CV 为当前已输出脉冲的个数, ROS 为脉冲输出结束标志。

为了避免系统启动或停止时引起的速度突变, 可以通过 ADT 设置加减速脉冲个数, 从而保证系统在启动和停止的过程中, 脉冲频率会以 S 曲线的方式进行加减速。而当该值为 0 时, 高速 Qx 不进行加减速, 直接以 NRF 频率输出。对于 S 曲线的一些参数, 可以通过 SM 区进行设置, 详见 SM 区说明文档。

6.17.6.3 相对定位指令 (DRVI)

● 指令参数

相对定位指令 (DRVI)			
		实现定位系统的相对定位功能。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位, 高电平触发
NRP	双字变量、常数	DWORD	将要输出的脉冲个数
NRF	双字变量、常数	DWORD	运行频率, 单位为 Hz
ADT	双字变量、常数	DWORD	脉冲输出的开始阶段和结束阶段加减速脉冲个数
ENO	位变量	BOOL	输出能量流
Qx	Q 区位变量	BOOL	DQ 输出选择, 必须填写 Q0.6 或 Q0.7(PEC8000), Q0.0 或 Q0.1(PEC3000), 其余不支持
DIR	Q 区的位变量	BOOL	脉冲方向信号, NRP 为正时输出 1, 为负时输出 0

ROS	位变量	BOOL	指令执行结束标志
CV	双字变量、常数	DWORD	功能块当前已输出脉冲个数

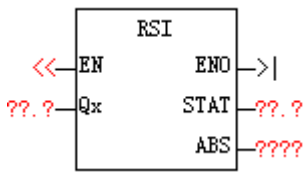
- 详细说明

DRVI 指令在高速 Qx 上以 NRF 频率输出 NRP 个脉冲,并且会根据方向位 DIR 改变高速 Qx 对应的坐标变量, CV 为当前已输出脉冲的个数, ROS 为脉冲输出结束标志, 当此位为 1 时, CV 值等于 NRP 的值。

为了避免系统启动或停止时引起的速度突变, 可以通过 ADT 设置加减速脉冲个数, 从而保证系统在启动和停止的过程中, 脉冲频率会以 S 曲线的方式进行加减速。而当该值为 0 时, 高速 Qx 不进行加减速, 直接以 NRF 频率输出。对于 S 曲线的一些参数, 可以通过 SM 区进行设置, 详见 SM 区说明文档。

6.17.6.4 读取当前值指令 (RSI)

- 指令参数

读取当前值指令 (RSI)			
		读取高速 Qx 的信息。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位, 高电平触发
Qx	Q 区位变量	BOOL	要读取的高速通道, 必须填写 Q0.6 或 Q0.7(PEC8000), Q0.0 或 Q0.1(PEC3000)
STAT	位变量	BOOL	高速 Qx 当前的状态, 1 为工作状态, 0 为空闲状态
ABS	双字变量	DWORD	Qx 当前的绝对坐标值
ENO	位变量	BOOL	输出能量流

- 详细说明

每一路高速 Qx 对应一个坐标变量, 它记录 Qx 当前的坐标值。同一时刻只允许有一组定位指令运行, 原点回归指令对坐标变量进行清零, 相对及绝对定位指令对坐标变量进行增减。

RSI 指令可以读取当前的坐标及高速 Qx 当前的状态, 当 STAT 为 1 时表示高速 Qx 为工作状态, 此时不能运行其它定位指令。

- 定位详细说明

每一路高速 DQ 对应一组定位指令(包括 ZRN、DRVA、DRVI、RSI)。为了方便用户在定位方面的应用, 这里为每路高速 DQ 建立一条坐标轴来管理其输出的脉冲个数。用户在每次上电时通过运行一次 ZRN 指令确定机械原点的位置, 这时系统会将对应的坐标变量清零, 作为该路高速 DQ 坐标轴的原点。坐标原点确定以后, 就可以运行定位指令了。

当运行绝对定位指令 DRVA 时, 如果目标坐标值 DST 的值为正数, 那么 DIR 输出为 1, 配合高速 Qx 控制电机正转, 同时坐标变量在高速 Qx 输出一个脉冲后加一; 如果目标坐标值 DST 为负数, 那么 DIR 输出 0, 配合高速 Qx 控制电机反转, 同时坐标变量在高速 Qx 输出一个脉冲后减一。

当运行相对坐标指令 DRVI 时, NRP 表示即将要输出的脉冲个数, 其值可正可负。当为正数时, 那么 DIR 输出为 1, 配合高速 Qx 控制电机正转, 同时坐标变量在高速 Qx 输出一个脉冲后加一; 当为负数时, 那么 DIR 输出 0, 配合高速 Qx 控制电机反转, 同时坐标变量在高速 Qx 输出一个脉冲后减一。

另外用户通过 RSI 指令, 可以读取某一路高速 Qx 当前的坐标值即正向方向累加后的脉冲个数。

表 6.17.4 高速 SM 区参数说明

Q0.7 (PEC8000 和 PMC2000) 或 Q0.0 (PEC3000 和 DIO3000) 定位 SM 区参数	
SMD20	已输出脉冲个数
SMW22	S 曲线加速时间(单位 ms)

SMW23	S 曲线最大加速度 Amax
SMW24	S 曲线的起始频率(默认为 2)
SM25.0	为 1 时, 将 SMD26 的值写入到第一路当前坐标变量中
SM25.1	为 1 时, 将 SMD36 的值写入到第二路当前坐标变量中
SMD26	定位指令坐标值
Q0.6 (PEC8000 和 PMC2000) 或 Q0.1 (PEC3000 和 DIO3000) 定位 SM 区参数	
SMD30	已输出脉冲个数
SMW32	S 曲线加速时间(单位 ms)
SMW33	S 曲线最大加速度 Amax
SMW34	S 曲线的起始频率(默认为 2)
SM35.0	为 1 时, 将 SMD26 的值写入到第一路当前坐标变量中
SM35.1	为 1 时, 将 SMD36 的值写入到第二路当前坐标变量中
SMD36	定位指令坐标值

● 应用示例

如图 6.17.10, 定位功能块为每一路高速输出定义了坐标系, 回原点指令可以在运行完成时将坐标清零, 绝对定位、相对定位指令可以对当前坐标进行增减, 而通过 RSI 指令可以读取当前坐标值。下面的程序就是利用定位指令, 从原点运行到坐标 50000 处, 再在坐标 50000 处返回到坐标 10000 处。

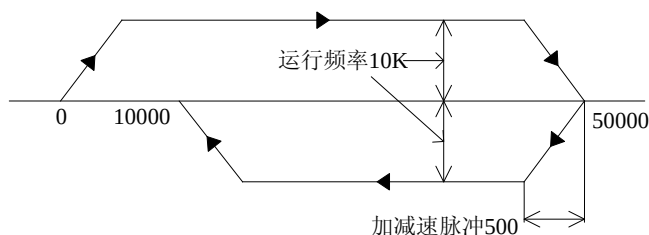


图 6.17.10 运行方向示意

如图 6.17.11 和图 6.17.12, 这里可以通过几个定位功能块完成上述功能。首先通过绝对定位指令以 5K 的频率正向输出 50000 个脉冲, 然后再通过绝对定位功能块, 反向输出 40000 个脉冲。当然为了避免速度突变, 在高速脉冲的启动与停止过程中, 通过 S 曲线的方式进行加减速, 其加减速脉冲个数为 500。

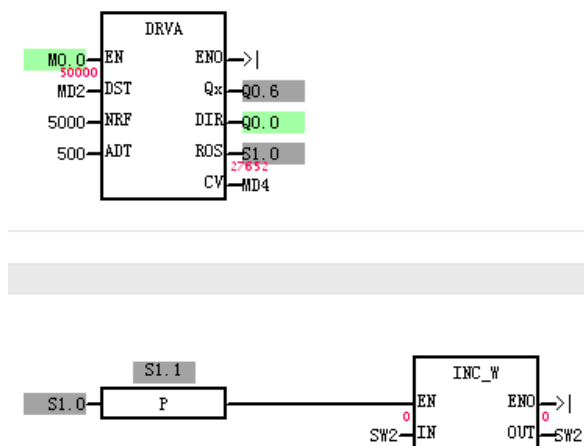


图 6.17.11 输出正向脉冲

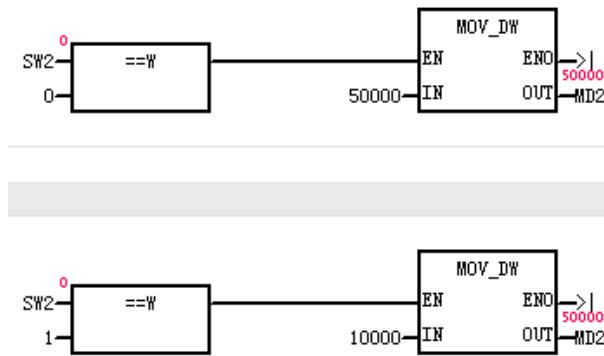


图 6.17.12 获取脉冲个数

6.17.6.5 旋转方向脉冲输出指令 (PLSV)

● 指令参数

旋转方向脉冲输出指令 (PLSV)			
PLSV <<—EN ENQ—> ????—NRF Qx—??. Dir—??. ROS—??. CV—???		输出一定数量的脉冲。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位，高电平触发
NRF	双字变量、常数	DWORD	脉冲输出频率，单位为 Hz
ENO	位变量	BOOL	输出能量流
Qx	Q 区位变量	BOOL	DQ 输出选择，PEC8000/PMC2000 必须为 Q0.6 或 Q0.7，PEC3000/DIO3000 必须是 Q0.0 或 Q0.1，其余不支持，最高输出频率为 50KHz
Dir	Q 区位变量	BOOL	旋转方向输出，用于控制电机的转向
CV	双字变量、常数	DWORD	功能块当前已输出脉冲个数
ROS	位变量	BOOL	指令执行结束标志

● 详细说明

PLSV 指令用于输出一定数量的脉冲，EN 为 1 时启动脉冲输出，EN 为 0 时停止脉冲输出。根据频率设置值的正负控制 Dir 的方向，当脉冲频率为正值时，Dir 为 ON、当脉冲频率为负值时，Dir 为 OFF。这里的 Qx 必须为高速输出，对于 PEC8000/PMC2000 只能为 Q0.6 和 Q0.7，而对于 PEC3000/DIO3000 只能为 Q0.0 和 Q0.1。

6.17.6.6 脉冲输入输出指令 (PLSE)

● 指令参数

脉冲输入输出指令 (PLSE)			
		根据输入参数输出一定数量的脉冲。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能位, 高电平触发
NRF	双字变量、常数	DWORD	脉冲输出频率, 单位为 Hz(浮点数)
NRP	双字变量、常数	DWORD	当前脉冲
NF	双字变量、常数	REAL	系数(浮点数), 频率系数, 输出频率=NRF×NF
NP	双字变量、常数	REAL	系数(浮点数)、脉冲系数、输出脉冲=NRP×NP
PV	双字变量、常数	REAL	脉冲输出起始值
ENO		BOOL	输出能量流
Qx	Q 区位变量	BOOL	DQ 输出选择, PEC8000/PMC2000 必须为 Q0.6 或 Q0.7PEC3000/DIO3000 必须是 Q0.0 或 Q0.1, 其余不支持, 最高输出频率为 50KHz。
Dir	位变量	BOOL	旋转方向输出, NRP>CV 时为正, NRP<CV 时为负
ROS	位变量	BOOL	指令执行结束标志, 指令执行结束后置 1

● 详细说明

PLSE 指令根据输入的 NRF(频率)、NRP(脉冲个数)输出一定数量的脉冲, 直至输出脉冲当前值 CV 与 NRP 相等。在 EN 有效的情况下, NRF 与 NRP 参数可以实时的改变(可以将 HDEF 指令的输出作为该指令的输入), 并且定位指令的坐标变量会随着 CV 值进行增减。

SMW28 和 SMW38 为指令执行结束控制变量。由于输入参数不断变化, 为了保证 ROS 的可靠性, 当 NRP 与 CV 相等时, 并不是马上置位 ROS 位, 而是等待程序运行 N 轮后, NRP 与 CV 值仍然相等, 则置位 ROS, 表示指令输出结束。这里的 N 即为 SMW28 和 SMW38 的值, 用户可以根据自己的需要进行修改。

表 6.17.5 PLSE 指令结束阈值

SM 区	描述
SMW28	Q0.7 (PEC8000 和 PMC2000) 或 Q0.0 (PEC3000 或 DIO3000) PLSE 指令执行结束阈值, 单位为指令运行轮次。
SMW38	Q0.6 (PEC8000 和 PMC2000) 或 Q0.1 (PEC3000 或 DIO3000) PLSE 指令执行结束阈值, 单位为指令运行轮次。

● 应用示例

如图 6.17.13, 设置脉冲频率为 1000, 脉冲个数为 2000, 频率和脉冲个数的倍率 NF、NP 为 1, 初值 PV 为 100, 那么当使能位为 1 时, Q0.06 会以 1000 的频率输出 1900 个脉冲, 由于频率是正值所以方向 Q0.0 输出, 当脉冲输出完成以后, 输出结束标志 S0.0 置位。

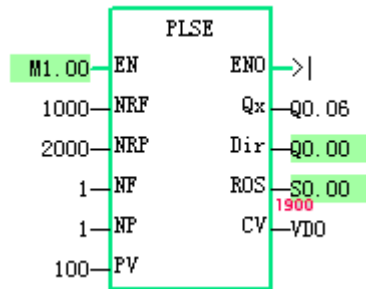


图 6.17.13 脉冲输入输出指令示例

6.17.7 往复运动

● 指令参数

往复运动指令 (RECIPROCATE)			
<pre> RECIPROCATE <<--EN ENO--> ????--NRF Qx--???.? ????--NF Dir--???.? ????--START ROS--???.? ????--END CV--???? ????--NP ???.?--REST ???.?--REC ???.?--ZRN </pre>			
本指令主要用于实现往复运动及回原点操作。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能控制位，电平信号
NRF	双字变量、常数	DWORD	往复运动的输出速度
NF	双字变量、常数	REAL	速度齿轮比(浮点数)，输出频率=NRF*NF
START	双字变量、常数	REAL	往复运动的起始位置
END	双字变量、常数	REAL	往复运动的结束位置
NP	双字变量、常数	REAL	位置齿轮比(浮点数)，起始输出脉冲点=START×NP，结束输出脉冲点=END×NP
REST	位变量	BOOL	原点复位开关，上升沿触发时进行回原点操作
REC	位变量	BOOL	往复运动开关，上升沿触发时进行往复运动
ZRN	位变量	BOOL	原点限位标志，当 ZRN 为 1 时，电机反向运动固定的脉冲数后停止
ENO	位变量	BOOL	输出能量流
Qx	Q 区位变量	BOOL	DQ 输出选择，PEC8000/PMC2000 必须为 Q0.6 或 Q0.7，PEC3000/DIO3000 必须是 Q0.0 或 Q0.1，其余不支持
Dir	位变量	BOOL	输出方向，由 START 运行到 END 时 Dir 置 0，反之 Dir 清 1
ROS	位变量	BOOL	输出完成标志，指令执行结束后置 1
CV	双字变量、常数	DWORD	当前输出的脉冲个数，每次换向该值都清零

● 详细说明

往复运动指令可实现设备的往复运动和回原点操作。往复运动过程中为了减少输出惯性对换向时带来的冲击影响，高速输出采用 S 曲线方式进行。该指令为高电平触发，当使能位 EN 为“1”时，可进行电机往复运动或回原点操作，否则电机不运行。

当设备重新上电且喷雾孔不在原点处时，需要先给原点复位开关（REST）一个上升沿信号，这时高速

输出通道以指定频率（默认为 500Hz）向原点方向输出，碰到原点附近的限位开关后改变输出方向，继续输出 400 个脉冲后停止输出，即当前位置为原点处。

当喷雾孔在原点处时，可以启动设备进行往复运动。需要先给往复运动开关（REC）一个电平信号，这时高速输出通道以设定频率（NRF）正向输出，输出 $END \times NP$ 个脉冲后，自动切换输出方向，反向输出 $(END-START) \times NP$ 个脉冲后再切换方向，周而复始的进行往复运动。

为了便于客户使用，回原点的频率可通过 SM 区读写，用户可根据实际需要修改。该指令在运行前需要通过 SM 区配置以下三个变量：

- (1) 回原点起始频率；
- (2) 执行回原点操作时，当运动杆碰到左限位后，需要反向输出的脉冲个数；
- (3) 回原点时的目标频率。

对应的配置方式见表 6.17.6:

表 6.17.6 回原点配置方式表

配置编号	输出引脚		默认值	单位
	Q0.6	Q0.7		
(1)	SMW34	SMW24	300	Hz
(2)	SMW36	SMW26	400	脉冲数
(3)	SMW38	SMW28	500	Hz

➤ 执行步骤

- ①设备上电时，获取输入引脚的值，并向对应的 SM 区中写入需要的参数；
- ②使能 REST 引脚，执行回原点操作，回原点完成后，确保 REST 引脚处于禁能状态；
- ③步骤 2 执行完成后，使能 REC 引脚，执行往复运动；
- ④当需要停止往复运动时，禁能 REC 引脚，自动执行回原点操作，回到原点位置停止输出。

➤ 注意事项

①设备需要回原点时，使能 REST 引脚。该引脚为沿触发，即使能后可以立即禁能而不会影响回原点操作的正常执行；

②设备需要执行往复运动时，使能 REC 引脚。该引脚为电平触发，即需要一直使能；

③当 REST 和 REC 引脚同时使能时，会执行回原点操作，因此在执行往复运动之前需要确保 REST 引脚已经被清零；

④在往复运动过程中，禁能 REC 引脚，则执行回原点操作直至回到原点位置；

⑤往复运动和回原点过程中，如果将 EN 使能引脚禁能，则设备立即停止在当前位置。

⑥往复运动最高输出频率为 16KHz，大于 16KHz 时偶尔会出现丢步现象。

⑦在一个工程中最多只能调用 2 个该指令。

● 应用示例

为了方便客户使用该指令，以下给出了一个往复运动的工程示例，该工程可以实现电机的往复运动以及回原点操作。

电机的参数如下：

转速 = $\text{频率} \times 60 / (360 / T) \times X$;

电机运行一圈的脉冲数 = $360 \times X / T$;

脉冲数 = $\text{运行长度} \times \text{电机运行一圈的脉冲数} / \text{电机运行一圈的长度}$;

其中 T 为固有步距值，X 为细分倍数。

通过上述公式可以得出：

- (1) 速度齿轮比 $NF = \text{频率} / \text{转速} = 6 \times X / T$;
- (2) 位置齿轮比 $NP = \text{脉冲数} / \text{运行长度} = (360 \times X / T) / \text{电机运行一圈的长度}$;

实际使用时，用户需要获得 NF 和 NP 的值。在 2.6 节中介绍了获取 NF 和 NP 的方式，仅供参考。

该节介绍指令的使用方式。

程序段 1 中，配置回原点参数。具体意义见功能说明章节。如图 6.17.14 所示。

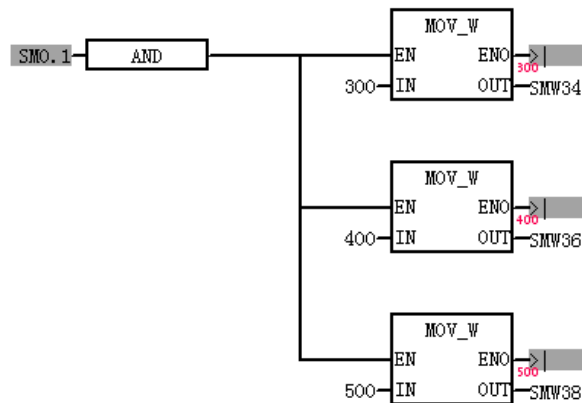


图 6.17.14 程序段 1 内容

程序段 2 中，配置往复运动的转速 MD194 和速度齿轮比 MD182。如图 6.17.15 所示。这里默认齿轮比为 1.0。

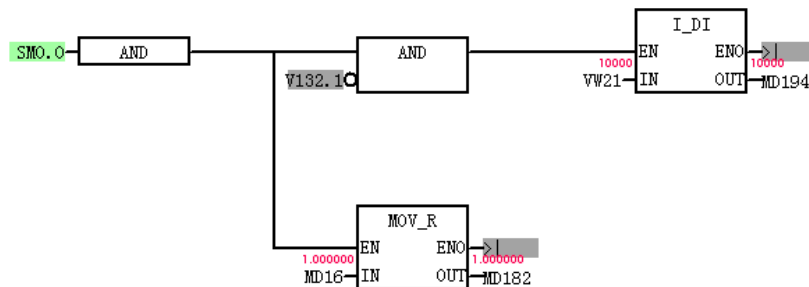


图 6.17.15 程序段 2 内容

子程序 3 中，配置往复运动的起始位置和目标位置，将位置齿轮比赋给 MD260（MD260 为往复运动指令的速度齿轮比 NP 的输入参数）。当 V132.1 为 0 时，从 VW24 中获取起始位置，从 VW26 中获取运动距离，则 VW24 与 VW26 的和即为运动的目标位置。如图 6.17.16 所示。这里默认齿轮比为 1.0。

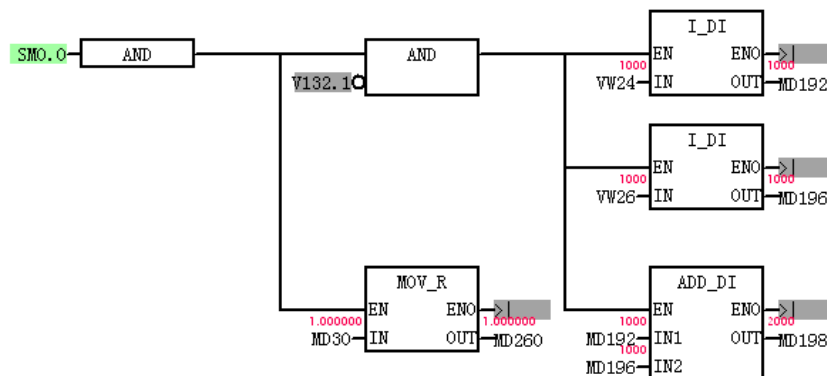


图 6.17.16 程序段 3 内容

程序段 4 中，为往复运动指令。该指令的输入参数从以上 3 个程序段中获得。V132.7 为回原点使能引脚，V132.0 为往复运动使能引脚，I0.2 为原点限位开关，Q0.6 为脉冲输出引脚，Q0.5 为方向引脚。

使能 V132.7 时，电机执行回原点操作，当电机碰到原点限位开关 I0.2 后，电机换向，并按照程序段 1 中的参数运行，输出 400 个脉冲后停止。如图 6.17.17 所示。

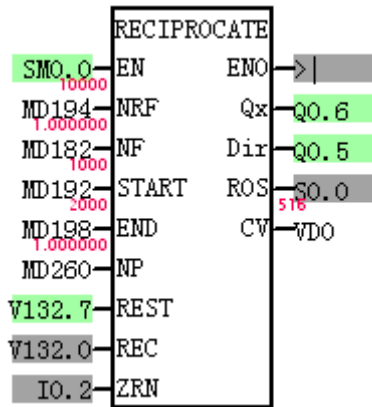


图 6.17.17 回原点运动过程

使能 V132.0 时，电机进行往复运动，直至 V132.0 复位，如图 6.17.18 所示。V132.0 复位后，电机执行回原点操作。

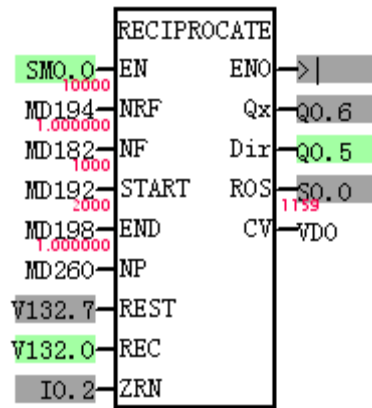


图 6.17.18 往复运动过程

6.17.8 斜率检测

● 指令参数

斜率检测指令 (SLC)			
???.? SLC <<—EN ENO—> ????—HANDLE ????—INTVAL ALM0—???.? ????—SLOPE ALM1—???.? ALM2—???.? ALM3—???.? ALM4—???.? ALM5—???.? ALM6—???.? ALM7—???.?			
检测 DUT6000、TAC3000 或 TAC6000 的输入温度斜率变化，并根据设置给出相应的报警。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
HANDLE	常数	WORD	所要操作从设备的序号 (0~15)
INTVAL	字变量、常数	WORD	斜率检测间隔，单位为秒
SLOPE	双字变量	REAL	报警斜率数值，单位为 0.1 度/秒
ALM0	位变量	BOOL	第 1 路控温通道的报警位

ALM1	位变量	BOOL	第 2 路控温通道的报警位
ALM2	位变量	BOOL	第 3 路控温通道的报警位
ALM3	位变量	BOOL	第 4 路控温通道的报警位
ALM4	位变量	BOOL	第 5 路控温通道的报警位
ALM5	位变量	BOOL	第 6 路控温通道的报警位
ALM6	位变量	BOOL	第 7 路控温通道的报警位
ALM7	位变量	BOOL	第 8 路控温通道的报警位
ENO	位变量	BOOL	输出能量流
STATE	位变量	BOOL	状态量

● 详细说明

SLC 指令用于检测作为从设备的 DUT6000、TAC3000 或 TAC6000 设备的输入温度升温斜率，并给出相应报警。每个功能块检测 1 个模块的 8 路控制通道，每路通道的报警输出对应 ALM0~ALM7，报警时输出为 1，不报警时输出为 0。

升温斜率单位为 0.1 度/秒，计算公式如下：

(本次采集通道温度值—上次采集通道温度值)/INTVAL

下列情况时报警输出为 1。在总控温使能及单路控温使能情况下，从常温状态开始，若相应 DQ 一直输出为 1 即全速加热，对应的通道升温斜率值小于所设定的阈值 SLOPE（单位为 0.1 度/秒）、若下次升温斜率值大于所设定的阈值。

下列情况之一时相应的报警输出清 0。(1) 总控温位为 0；(2) 总控温位为 1 但相应的通道单路控温位为 0。(3) 当通道已经进入稳定状态时（即从常温态开始相应 DQ 输出为 1，并且对应通道的升温斜率一直大于设定的阈值 SLOPE，直到下一次读入相应通道的 DQ 值为 0），相应的通道检测失效。

HANDLE 为需要检测的从设备，取值范围为 0~15。指令内部会自动根据配置的设备类型进行处理，非 DUT6000、TAC3000 或 TAC6000 模块将不作处理。

INTVAL 参数为斜率检测的时间间隔，单位为秒。在设置此参数时应符合实际的情况，一般建议设置在 1 分钟以上。

???.为指令运行状态指示，当下列状态之一时输出为 0。(1) 指令使能位为 0；(2) 从设备类型不为 TAC6000、DUT6000 或 TAC3000；(3) 从设备不在线；(4) 从设备总控温位为 0。正常情况下输出为 1。

6.17.9 PH 检测

● 指令参数

PH 值检测指令 (PH)			
测量水溶液中的 H ⁺ 含量。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
E	双字变量、常数	REAL	AI 通道的电压值，单位为毫伏
T	双字变量、常数	REAL	当前的温度值(通过 AI 通道采集)，单位为摄氏度
Ecal	双字变量、常数	REAL	传感器采集的电压值，单位为毫伏
Tcal	双字变量、常数	REAL	标定时的温度，单位为摄氏度
CPHcal	双字变量、常数	REAL	定标溶液对应的 H ⁺ 浓度值

MODE	字变量、常数	WORD	MODE: 模式选择。 0: 计算 PH 值; 1: 标定第一种定标溶液; 2: 标定第二种定标溶液; 3: 标定完成, 将标定参数存入 flash; 4: 恢复出厂参数; 5: 用户手动输入, 具体输入方式见下面的指令详细说明
Index	字变量、常数	WORD	测量通道选择
ENO	位变量	BOOL	输出能量流
PH	双字变量、常数	REAL	当前溶液对应的 PH 值
K0	双字变量、常数	REAL	计算 PH 时的斜率
E0	双字变量、常数	REAL	计算 PH 时的标定电压
T0	双字变量、常数	REAL	计算 PH 时的标定温度

● 详细说明

PH 指令用于测量水溶液的 PH 值。使用时应先在两种已知 PH 值的定标溶液中对 PH 电极进行定标（两点定标法），通过模式切换，获得电极在 PH 值为 PH1 的溶液中传感器的输出信号 E1、溶液温度 T1，以及在 PH 值为 PH2 的溶液中传感器的输出信号 E2、溶液温度 T2；而后将电极放入待测溶液中，控制器根据 A/D 传感器采集的输出信号 E 以及温度传感器采集的溶液温度 T，计算出当前溶液的 PH 值。

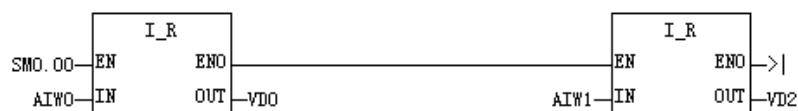
指令执行中，需要实时修改 MODE 的值，不同的 MODE 值所表示的意义不同。0 为计算 PH 值；1 为标定第一种定标溶液；2 为标定第二种定标溶液；3 为标定完成；4 为恢复出厂参数；5 为用户手动输入参数 K0、E0、T0，在此模式下，K0 是计算 PH 时的斜率，从 CPHcal 接口获得；E0 是计算 PH 时的标定电压，从 Ecal 接口获得，T0 是计算 PH 时的标定温度，从 Tcal 接口获得，这些值是计算 PH 时所必须用到的参数）。

● 应用示例

如图 6.17.19，程序中 AIW0，AIW1 分别接 PH 传感器信号调理板的输出端。AIW0 采集电压，AIW1 采集温度。调理板的具体使用方式见软仪表用户手册。

在网络 1 中，将采集的信号转换为浮点数据，并在网络 2 中将信号码值转换为真实值；

网络 1 PH 调理板输出的 MV 及温度



网络 2 转换数据为程序



图 6.17.19 电压和温度采集

如图 6.17.20，网络 3 中，将 PH 电极放入第一种 PH 值的定标溶液中，将 MODE 赋值为 1，将传感器在第一种定标溶液中的信号存储在 VD50 中、温度存储在 VD52 中、定标溶液的 PH 值存储在 VD54 中，然后将 MODE 赋值为 3，将第一种标定溶液的值存入 flash 中。

图 3 注释...

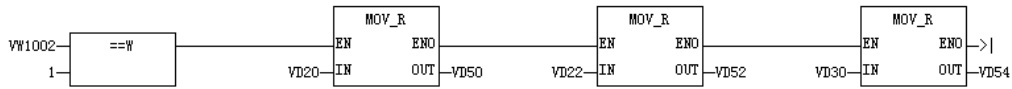


图 4 注释...

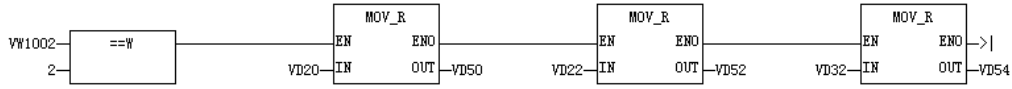


图 6.17.20 两种标定溶液值

在网络 4 中，将 PH 电极放入第二种 PH 值的定标溶液中，将 MODE 赋值为 2，如图 6.17.21，将传感器在第二种定标溶液中的信号存储在 VD50 中、温度存储在 VD52 中、定标溶液的 PH 值存储在 VD54 中，然后将 MODE 赋值为 3，将第一种标定溶液的值存入 flash 中。

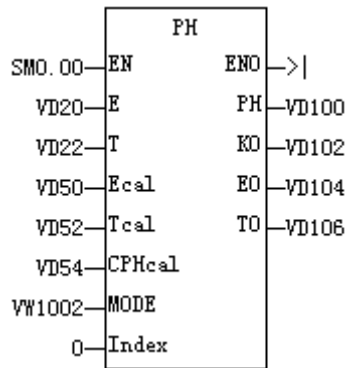


图 6.17.21 测量未知溶液的 PH 值

将传感器放入待测溶液中，将 MODE 赋值为 0，此时，PH 指令将根据当前传感器输出的信号 VD20、温度 VD22、及在定标溶液中获取的斜率计算出当前溶液的 PH 值。

注意：此例程中，标定溶液的数据不能一直存储在 VD50、VD52、VD54 中，因为在标定第二个溶液时，此三个寄存器中原来的第一个标定溶液的的参数会被覆盖。但参数已经存到 flash 里了。因此如果用户在使用时需要查看两种标定溶液的参数，需要另外编写指令存储。

提示：当用户在同一环境下，只需要标定一次，之后就可以正常检测了。

6.17.10 电导率检测

● 指令参数

电导率检测指令 (DD)			
DD <<—EN ENO—> ????—E DDS—???? ????—T ????—f1 ????—f2 ????—FS ????—K ????—Tp ????—MODE ????—Index			
测量液体的电导率			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流

E	双字变量、常数	REAL	测量时的电压
T	双字变量、常数	REAL	测量时的温度
f1	双字变量、常数	REAL	第一设定频率
f2	双字变量、常数	REAL	第二设定频率
FS	位变量	BOOL	滤波选择, 0 为不滤波, 1 为滤波
K	双字变量、常数	REAL	电极常数, 根据被测溶液不同而不同
Tp	双字变量、常数	REAL	温补系数, 一般溶液: 0.02; 酸类溶液: 0.016; 盐类溶液: 0.024; 碱类溶液: 0.019; 其他: 此时未加温补
MODE	字变量、常量	WORD	MODE: 测量模式。 0: 测量模式; 3: 使用 1K 上拉电阻时校准 1K 电阻 Ω; 4: 使用 1K 上拉电阻时校准 100K 电阻; 5: 使用 100K 上拉电阻时校准 100K 电阻; 6: 使用 100 上拉电阻时校准 200K 电阻; 7: 校准电阻完成标志; 8: 恢复默认参数。
Index	字变量、常量	WORD	测量通道选择
ENO	位变量	BOOL	输出能量流
DDS	双字变量、常数	REAL	计算溶液的电导率

● 详细说明

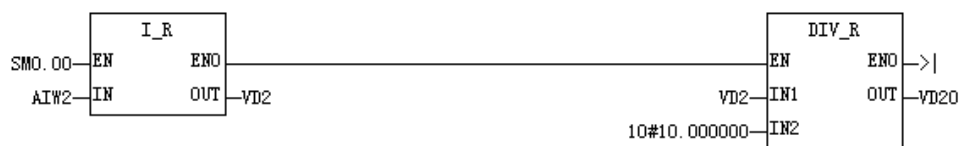
DD 指令只有 EDC8000 支持, 作用是测量液体的电导率。上电后, 通过激励源间断的给出两个频率, 激励电导池。从电导池分压得到的两个电压量经过缓冲、整流和滤波后输出给主控制器。主控制器通过得到的参数计算电阻值并进行校准, 最后利用校准完成后的电阻值、电极常数、温度以及温补系数求得电导率的值。

● 应用示例

将 EDC8000 控制器与电导率调理板连接, 调理板上的 Q1 和 Q2 分别与控制器的 Q3 和 Q2 连接, V0+ 和 V0- 接 AI 口, 例如 A0 和 B0, IN+ 和 IN- 接一个 100K 的电阻。程序中 AIW0, AIW1 分别接 A/D 传感器输出端, 温度传感器输出端。调理板的具体使用方式见软仪表用户手册。

如图 6.17.22, 网络 1 中采集电压信号, 并将其信号码值转换成真实值 (IN2 的数值视现场调理板的放大倍数而定); 网络 2 中采集温度信号, 并将其信号码值转换成真实值 (IN2 的数值视现场温度传感器的放大倍数而定);

网络 1 电导率调理板采集的输出电压值, 并将其转换成真实值



网络 2 电导率调理板采集的输出电压值, 并将其转换成真实值



图 6.17.22 电压和温度采集

先后给 MODE 赋值为 3、4、5、6，分别校准校准 100Ω、1K、10K、和 100K 电阻，校准完成后需要向 MODE 中写 7，保存校准值。最后向 MODE 中写入 0，进行电导率的计算。其中 FS 为有无滤波设置，FS 为 1 时进行滤波操作，为 0 时则不滤波，滤波时，测量速度比不滤波慢，但测量精度将比不滤波高。此模式用户可根据实际应用需要进行设置；K、Tp 为与用户电极及实际测量环境有关的量，用户根据实际使用情况进行设置。

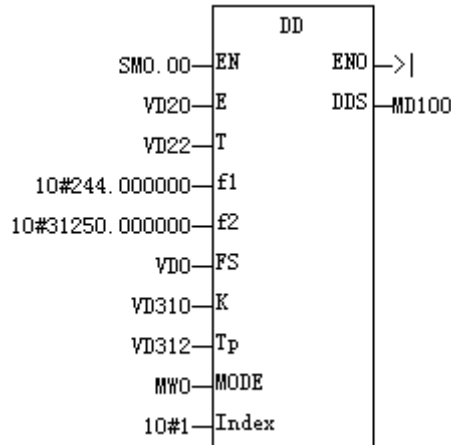


图 6.17.23 计算电导率的值

提示：当用户在同一环境下，只需要标定一次，之后就可以正常检测了。

6.17.11 溶氧检测

● 指令参数

溶氧检测指令 (OXM)			
		测量介质中的溶解氧含量。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
E	双字变量、常数	REAL	输入电压值（毫伏）
T	双字变量、常数	REAL	输入温度值
K	双字变量、常数	REAL	用户手动输入斜率（NA/mbar）
P	双字变量、常数	REAL	手动输入标准大气压
MODE	字变量、常数	WORD	模式选择。 0：测量模式的分压表示法； 1：测量模式的氧浓度表示法，单位为 ppb； 2：标定零点电流； 3：标定满度电流； 4：标定完成标志；

			5: 恢复出厂参数; 6: 用户手动输入模式, 输入参数有 K0、Izero、Ifull, 其中 K0 从 K 接口获得, Izero 从 E 接口得到, Ifull 从 T 接口得到, 这些值是计算 OXM 值所必须用到的参数)。
Index	字变量、常数	WORD	测量通道选择
OXM	双字变量	REAL	输出的溶氧值
K0	双字变量、常数	REAL	计算溶氧的斜率
Izero	双字变量、常数	REAL	溶氧电流零点
Ifull	双字变量、常数	REAL	溶氧电流满度
T0	双字变量、常数	REAL	测量溶氧时的标定温度
ENO	位变量	BOOL	输出能量流

● 详细说明

OXM 指令只有 EDC8000 支持, 作用是获得溶液中溶解氧的含量。测量前需要进行参数的标定, 需要标定的参数有电流零点, 电流满度, 以及温度满度。使用标定的参数结合大气压值计算出溶液溶解氧的含量。

● 应用示例

如图 6.17.24, 网络 1 中将从 NTC 传感器采集到的信号转换为浮点数据, 并在网络 2 和网络 3 中进行判断, 当 NTC 的值大于 100 时, 通过 AIW0 获取电压值; 当 NTC 的值小于 100 时, 通过 AIW5 获取电压值。

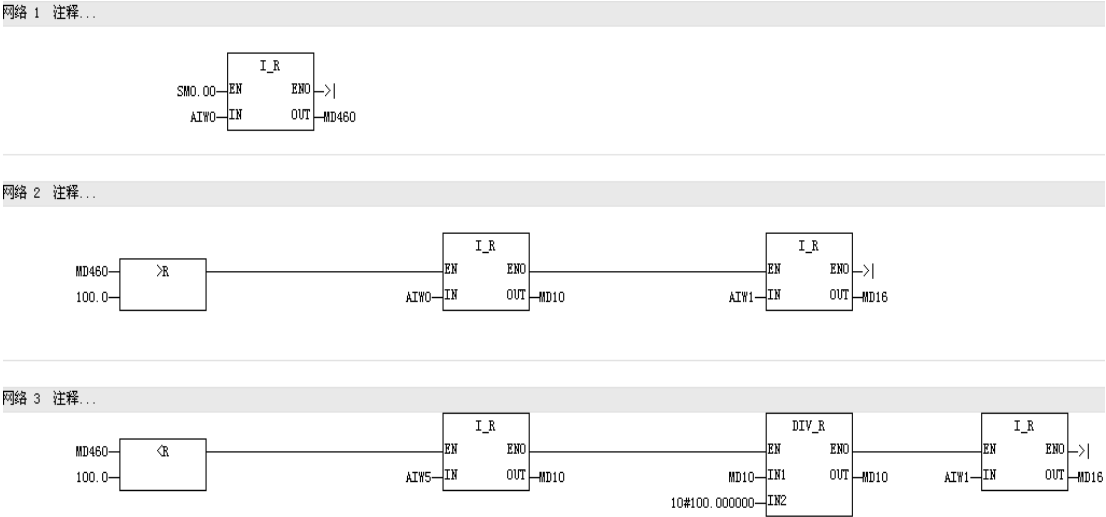


图 6.17.24 采集电压和温度

如图 6.17.25, 网络 4 中将获得的温度和电压转换为真实值, 利用获得的数据在网络 5 中计算溶液的溶氧值。

网络 4 注释...



网络 5 注释...

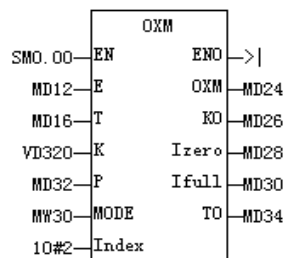


图 6.17.25 计算溶氧值

溶解氧测量步骤如下:

首先将 **MODE** 设置为 2, 标定电流零点。E 为零点电流码值。然后将 **MODE** 设置为 3, 标定电流满度。E 为满度电流码值, T 为满度温度码值。校准完成后, 将 **MODE** 设置为 4, 结束校准, 此时校准数据将保存到 FLASH 中, 数据掉电保存, 下次上电后即不用进行校准, 直接进行测量操作即可。若用户校准过程出现误操作, 想恢复出厂参数, 将 **MODE** 设置为 5, 即向 VW30 中写入 5, 即可恢复出厂时的标定参数。当电流零点和满度标定完成以后, 将 **MODE** 设置为 0, 进行溶解氧浓度的测量。

注意: **OXM** 指令采集溶液电压信号时, 需要使用 **NTC** 型传感器, 用户在选择传感器类型时, 需要特别注意。

提示: 当用户在同一环境下, 只需要标定一次, 之后就可以正常检测了。

6.17.12 Na 离子检测

● 指令参数

Na 离子检测指令 (Na)			
Na <<—EN ENO—> ????—E CNa—???? ????—T KO—???? ????—Ecal EO—???? ????—Tcal TO—???? ????—CNacal ????—MODE ????—Index 对含钠的水溶液中钠离子进行测量。			
参数名	变量区	数据类型	说明
EN	位变量	BOOL	输入能量流
E	双字变量、常数	REAL	传感器 AI 通道的电压值, 单位为毫伏
T	双字变量, 常数	REAL	当前温度值(通过 AI 通道采集), 单位为摄氏度
Ecal	双字变量, 常数	REAL	校准溶液的电压值, 通过 AI 通道读出。
Tcal	双字变量, 常数	REAL	校准时的温度, 单位为摄氏度
CNacal	双字变量, 常数	REAL	校准溶液对应的 Na ⁺ 浓度值 (ppb)
MODE	字变量、常数	WORD	模式选择。 0 为计算溶液 Na 离子浓度;

			1 为标定第一种定标溶液； 2 为标定第二种定标溶液； 3 为标定第三种定标溶液； 4 为采用三点标定的方法计算标定参数(采用三点标定时，计算取第一，第二和第三种定标溶液参数)； 5 为标定完成标志； 6 为恢复出厂参数； 7 为用户手动输入模式，输入参数有斜率 K0、E0、T0，其中 K0 从 CNacal 接口获得，E0 从 Ecal 接口获得，T0 从 Tcal 接口获得； 8 为采用两点标定待测方法计算标定参数（采用两点标定时，计算取第二和第三种定标溶液参数）。
Index	字变量、常数	WORD	测量路数选择
CNa	双字变量，常数	REAL	为当前溶液对应的钠离子值
E0	双字变量，常数	REAL	测量溶液时的标定电压
K0	双字变量，常数	REAL	测量溶液时斜率
T0	双字变量，常数	REAL	测量溶液时的标定温度
ENO	位变量	BOOL	输出状态指示

● 详细说明

Na 离子检测指令只有 EDC8000 设备（版本号在 V5006 以上）支持，用于对溶液中的 Na 离子浓度进行测量。初次使用时应对钠电极进行标定。标定步骤如下：

- (1) 将钠电极放入零钠水中，当电压稳定后，此时的电压值即为零钠时的钠电极输出电压。
- (2) 将钠电极放入第一种标定溶液中，当电压稳定后，此时的电压值即为第一种标定时的钠电极输出电压。
- (3) 将钠电极放入第二种标定溶液中，当电压稳定后，此时的电压值即为第二种标定时的钠电极输出电压。

指令将根据三种标定溶液中的数值，计算出钠离子浓度值。

● 应用示例

如图 6.17.26，程序中 AIW2，AIW1 分别接 Na 传感器信号调理板输出端，温度传感器输出端。

网络 1 中将采集的信号转换为浮点数据，并在网络 2 中将信号码值转换为真实值；

网络 1 Na离子调理板输出的MV及温度



网络 2 将获得的码值转换成真实值

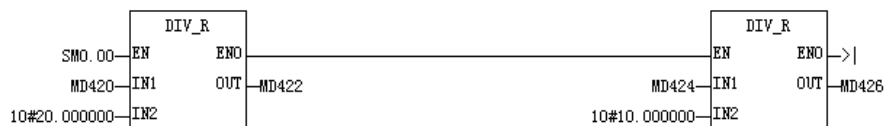


图 6.17.26 采集电压和温度

如图 6.17.27，网络 3 中，将 Na 离子电极放入第一种 Na 离子值的定标溶液中，将 MODE 赋值为 2，获取定标溶液一的 Na 离子值存储在 VD434 中，然后将 MODE 设置为 5，确认标定完成，将参数存储到 flash 中。

网络 3 模式2，标定第一种溶液



网络 4 模式3，标定第二种溶液

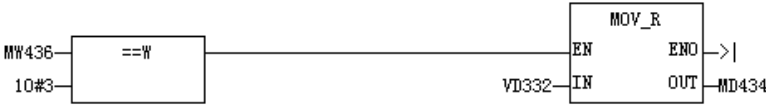
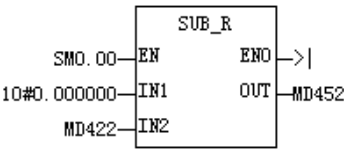


图 6.17.27 采集电压和温度

网络 4 中，将 Na 离子电极放入第二种 Na 离子值的定标溶液中，将 MODE 赋值为 3，获取定标溶液二的 Na 离子值存储在 VD434 中，然后将 MODE 设置为 5，确认标定完成，将参数存储到 flash 中。

如图 6.17.28，标定完成后，需要将 MODE 设置为 8，计算两点定标方式下的斜率值，然后将 MODE 设置为 5，确认标定完成。网络 5 中，将 Na 离子电极放入待测溶液中，将 MODE 赋值为 0，此时，Na 指令将根据当前传感器输出的电压、温度及在定标溶液中获取的斜率计算出当前溶液的 Na 离子值。

网络 5 将电压转换成正值



网络 6 计算Na离子浓度

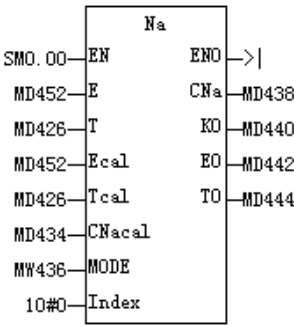
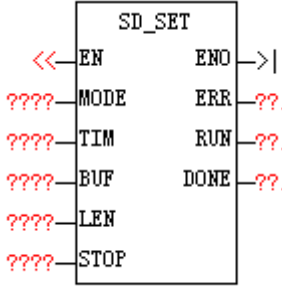


图 6.17.28 计算 Na 离子浓度

提示：当用户在同一环境下，只需要标定一次，之后就可以正常检测了。

6.17.13 SD 卡设置指令

● 指令参数

SD 卡设置指令(SD_SET)			
		SD 卡设置指令完成 SD 卡存储参数的配置	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	功能块使能信号，高电平触发
MODE	字变量、常数	WORD	数据存储模式设置。 模式 0：继续之前的数据存储； 模式 1：从头开始存储
TIM	字变量、常数	WORD	数据存储的最小时间间隔，单位为毫秒，最小 100ms
BUF	字变量	WORD	用户需要存储数据的起始地址
LEN	字变量、常数	WORD	用户需要存储的数据长度（最大长度 124 字）
STOP	字变量、常数	WORD	停止当前数据存储操作
ENO	位变量	BOOL	输出能量流
ERR	位变量	BOOL	SD 存储错误报警标志
RUN	位变量	BOOL	SD 卡运行状态输出
DONE	位变量	BOOL	设置成功标志

● 详细说明

SD 卡设置指令只有 MCD1000 设备支持，用于设置 SD 卡存储信息的参数。

具体设置参数如下

MODE：数据存数模式。为 0 时数据接着之前存储的数据进行存储，为 1 时数据指针清零重新进行数据存储。

TIM：为数据存储的最小时间间隔，单位为毫秒，其最小值为 100ms。

BUF：用户需要存储数据的起始地址

LEN：用户需要存储的数据长度

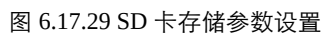
STOP：是否停止存储，为 1 时设备停止存储数据，为零时开始存储数据

RUN：SD 卡当前的状态信息，为 0 表示 SD 卡未被设置过，没有在运行。为 1 时表示 SD 卡已经设置过，并正在存储数据。

SET_DONE：返回设置成功信息，并开始存储数据。为 0 时，设置信息不正确保存失败，SD 卡无法保存数据。为 1 时 SD 卡数据设置成功，SD 卡开始存储数据信息。

● 应用示例

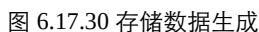
使用此功能块时首先调用功能块，并配置相应的输入引脚信息，如图 6.17.29 所示。



```

graph LR
    T1((T1)) -- EN --> ADD_I[ADD_I]
    VWO1[VWO] -- IN1 --> ADD_I
    1((1)) -- IN2 --> ADD_I
    ADD_I -- OUT --> AND[AND]
    VWO2[VWO] -- ==W --> AND
    AND --> MOV_W[MOV_W]
    MOV_W -- EN --> OUT1[>]
    MOV_W -- IN --> VWO3[VWO]
    MOV_W -- OUT --> OUT2[>]

```



6.17.14 气缸控制指令

PumpManual		
<<—EN		ENO—>
????—Pump		Pumping—???
???.?—Manual		ManualOn—???
???.?—PumpIn		ErrIn—???
???.?—PumpOut		ErrOut—???
???.?—Home		ErrReset—???
???.?—Motion		
???.?—Reset		
????—Time		

一个程序中共支持 32 个气缸指令。

第 283 页

ManualOn	位变量	BOOL	0 或 1	气缸手动动作中状态，为“1”时，表示气缸手动中。
ErrorIn	位变量	BOOL	0 或 1	当 Error 为“1”时，表示气缸回故障。
ErrorOut	位变量	BOOL	0 或 1	当 Error 为“1”时，表示气缸出故障。
ErrorReset	位变量	BOOL	0 或 1	当 Error 为“1”时，表示气缸故障可复位。

● 详细说明

通过 PumpManual 指令，实现气缸的基本动作、故障诊断、状态显示，并在故障状态下进行复位操作。

气缸指令为高电平触发，当使能位 En 为“1”时，可进行气缸操作，否则气缸操作无效。

在气缸输出时，当手动按钮上升沿和气缸回上升沿到来时，气缸停止输出；气缸停止输出时，当手动按钮上升沿和气缸出上升沿到来时，气缸输出。

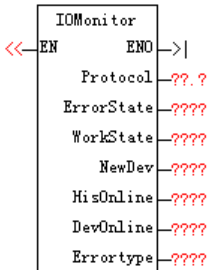
在气缸手动状态下，当手动按钮上升沿到来时，将气缸手动状态复位；在气缸不在手动状态下时，当手动按钮上升沿到来时，气缸手动状态置位。

气缸不在动点时，若气缸输出，则开始计时，计时时间到时，若气缸还在输出，则报气缸出故障；气缸不在原点时，若气缸停止输出，则开始计时，计时时间到时，若气缸还是停止输出，则报气缸回故障。

气缸出故障时，若气缸在动点，则气缸出故障可复位，此时，按复位按钮，可将气缸出故障复位；气缸回故障时，若气缸在原点，则气缸回故障可复位，此时，按复位按钮，可将气缸回故障复位。

6.17.15 IO 状态监控指令

● 指令参数

IO 运行状态监控指令			
		本指令主要用于监控 IO 设备的运行状态。	
参数名	变量区	数据类型	说明
EN	位变量	BOOL	使能控制位，电平信号
ENO	位变量	BOOL	输出能量流
Protocol	字变量	WORD	主串配置为 IO 协议，且底板串口为主口时，该位为 1；
ErrorState	字变量	WORD	每一位代表一个设备，该位为 1 表示有设备出错，该位为 0 时表示设备没有出错。
WorkState	字变量	WORD	显示当前工作状态： 1:扫描状态； 2:配置节点地址状态； 4:环网通信状态； 5:点对点通信状态；
NewDev	字变量	WORD	有新的地址设备在线，显示新设备在线状态；
HisOnline	字变量	WORD	历史物理地址存在设备，该寄存器为 16 位字变量，每一位代表一个物理地址；
DevOnline	字变量	WORD	在线设备，该位地址为 1 表示在线 0 表示不在线；
Errortype	字变量	WORD	首地址后 16 个变量，分别存储该地址设备的出错信息； 0: 未出错； 1: 设备通信错误； 2: 设备类型错误； 3: 设备 IO 映射错误； 4: AI 类型错误；

- 详细说明

IO 设备运行状态指示指令为高电平触发，当使能位 EN 为“1”时，可进行读取相关状态信息。

示例工程

- (1) 将 CIO100 设备通信协议设置为 IO 通信协议，波特率 1M，N，8，1；
- (2) 将 8 个 EIO3050 地址分配配置为 1~8，8 个 EIO3000 地址分配配置为 9~16，并且与 CIO100 主串口相连；
- (3) 打开 Config 软件打开 IO 设备配置对话框，点击扫描 IO 设备，等待扫描完成，看设备是否上线，IO 映射区域是否正常；
- (4) 下载 IO 设备运行状态指示指令；查看从设备的运行状态；
- (5) 将其中一个设备拿掉，查看是否转为点对点通讯模式，查看功能块的运行状态；
- (6) 将拿掉的设备重新上线，大约 10 秒内，查看是否重新恢复为环网通信模式。

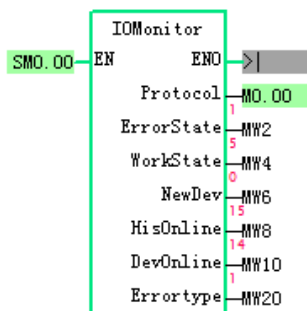


图 6.17.31 IOMonitor 使用示例工程

7 运动控制指令集

DCCE 运动控制指令，参考 PLCopen 标准，通过 PLC 指令的方式对机械运动部件的位置、速度等进行实时的控制管理，使其按照预期的运动轨迹和规定的运动参数进行运动，同时，运动指令还进行复杂的轨迹规划、多轴插补、加减速控制和动力学模型计算，使运动控制系统的精度更高、速度更快、运动更加平稳。

DCCE 运动控制指令包括单轴指令，多轴指令和轴组指令三大类。单轴指令主要应用于点位控制场合，对运动的终点位置和目标速度进行准确控制。多轴指令解决主从轴之间的协调运动关系，主要应用有电子齿轮和电子凸轮。轴组指令将多个轴组合成一个整体进行同步控制，提供轨迹插补和动力学模型，主要应用在多轴同步运行和具有轨迹规划的运动中。

DCCE 运动控制指令在对目标机构进行轨迹规划时，提供了空间直线插补、圆弧插补、NURBS 曲线插补等多种插补算法。在进行快速定位和减少系统抖动时，集成了梯形曲线，S 曲线，E 指数曲线等多种加减速控制曲线。同时，DCCE 运动控制指令还集成了直角模型、圆柱模型、四轴并联模型、三轴并联模型、二轴并联模型、三轴座椅模型、SCARA 模型、四轴码垛机器人模型、通用模型等丰富的动力学模型，用户可根据被控对象，选择满足系统需求的机器人模型进行控制，也可以使用通用模型自己编写模型控制函数，进行控制。

7.1 运动控制指令表

运动控制指令包括单轴指令，多轴指令和轴组指令三大类，涉及到轴和轴组的加减速控制、插补算法、运动执行、运动监控与管理。

单轴指令功能描述如表 7.1.1 所示。

表 7.1.1 单轴指令列表

指令名称	功能描述
绝对定位——MC_MoveAbsolute	控制轴运行到指定的绝对目标位置
相对定位——MC_MoveRelative	控制轴从当前位置运行指定的距离
叠加定位——MC_MoveAdditive	在轴的目标位置上叠加一段距离作为新的目标位置 控制轴运行到新的目标位置
重新定位——MC_MoveSuperimposed	在轴运行时，修改目标位置及目标速度
恒速运行——MC_MoveVelocity	控制轴以设定的速度恒速运行
停止轴——MC_Stop	停止轴运行
点动运行——MC_Jog	控制轴以设定的速度点动运行
回原点——MC_Home	控制轴回原点
扩展回原点——MC_HomeEx	控制轴回到距离原点指定偏移量的位置
往复运行——MC_Reciprocator	控制轴往复运行
轴复位——MC_Reset	复位轴错误状态
读轴状态——MC_ReadStatus	读取轴当前的运行状态
读轴故障——MC_ReadAxisError	读取轴的故障信息
读轴参数——MC_ReadParameter	读取轴参数
写轴参数——MC_WriteParameter	设置轴参数
读布尔型参数——MC_ReadBoolParameter	读取轴的布尔型参数
写布尔型参数——MC_WriteBoolParameter	设置轴的布尔型参数
读轴位置——MC_ReadActualPosition	读取轴当前的实际位置
写轴位置——MC_WriteActualPosition	设置轴当前的实际位置
编码器轴配置——MC_CoderMap	设置当前轴为编码器轴
立即停止配置指令——MC_EmgStopSet	读取 DI 状态实现轴立即停止运动

多轴指令功能描述如表 7.1.2 所示。

表 7.1.2 多轴指令列表

指令名称	功能描述
凸轮运行——MC_CamIn	控制凸轮按照凸轮表运行
凸轮分离——MC_CamOut	控制从轴脱离主轴，停止凸轮按照凸轮表运行

齿轮运行——MC_GearIn	控制齿轮按照电子齿轮比运行
齿轮分离——MC_GearOut	停止齿轮按照电子齿轮比运行
相位偏移——MC_Phasing	控制主轴相位偏移设定的距离
创建凸轮表——MC_CamTableCreate	创建凸轮表
删除凸轮表——MC_CamTableDelete	删除凸轮表
写凸轮表数据——MC_CamTableWriteData	将数据写入凸轮表
读凸轮表数据——MC_CamTableReadData	将凸轮表数据读出
读从轴位置——MC_CamTableReadSlavePos	根据主轴位置读取从轴的位置
旋切初始化——MC_APF_RotaryCutInit	根据旋切工艺参数创建或修改旋切表
旋切耦合——MC_APF_RotaryCutIn	控制旋切轴按照指定的旋切曲线进行运动
旋切分离——MC_APF_RotaryCutOut	解除旋切轴和进给轴之间的旋切关系，并使旋切轴停止在设定位置上
多轴同步运行——MC_MultiAxesSyncMove	控制多轴同时启动、同时停止、等时控制、恒速控制和直线运行

轴组功能描述如表 7.1.3 所示。

表 7.1.3 轴组指令列表

指令名称	功能描述
绝对直线运行——MC_MoveLinearAbsolute	控制轴组按照设定的参数直线运行到绝对位置
相对直线运行——MC_MoveLinearRelative	控制轴组按照设定的参数直线运行到相对位置
绝对样条运——MC_MoveSplineAbsolute	控制轴组按照设定的参数曲线运行到目标位置
轴组点动运行——MC_GroupJog	控制轴组按照设定的方向恒速点动运行
轴组恒速运行——MC_GroupMoveVelocity	控制轴组按照设定的方向恒速运行
停止轴组——MC_GroupStop	停止轴组运行
平面圆弧运行——MC_MoveCircular2D	控制轴组进行平面圆弧插补运行
连续曲线运行——MC_ContinuousCurve	控制轴组按照指定的连续时间和位置曲线运行
轴组写实际位置——MC_WriteActPosition	设置轴组当前的实际位置
读轴组位置——MC_GroupReadActualPosition	读取轴组的实际位置
读轴组速度——MC_GroupReadActualVelocity	读取轴组的实际速度
读轴组故障——MC_GroupReadError	读取轴组故障信息
轴组复位——MC_GroupReset	复位轴组状态
读轴组状态——MC_GroupReadStatus	读取轴组的运行状态
轴组使能——MC_GroupEnable	轴组使能
轴组禁能——MC_GroupDisable	轴组禁能
空间圆弧运行——MC_MoveCircular3D	控制轴组进行空间圆弧插补运行

7.2 运动控制状态机

运动控制为每个轴维护了一个状态机，标定了轴当前的运行状态。在轴进行运动过程中，会发生状态的跃迁。用户可以通过查询轴当前的运行状态，更好的对轴进行控制管理。图 7.2.1 为单轴和多轴运动控制中轴状态跃迁图，主要包括禁能状态 Disabled、停止状态 Stopping、离散运行状态 DiscreteMotion、连续运行状态 ContinuousMotion、暂停状态 StandStill、同步运行状态 Synchronized- Motion、错误状态 ErrorStop、回原点状态 Homing 等。

Disabled 状态，轴的运行控制被禁止，任何功能块都无法执行，此时只有通过修改运动控制参数的使能端为 TRUE，轴才被使能。

StandStill 状态，即停止状态。在轴初始化完成后，轴会进入停止状态，或者运行控制指令执行完成时，轴也会进入此状态。

Discrete Motion 状态，即离散运行状态。通过加速、恒速、减速三个阶段控制轴运动到目标位置，在运动过程中，轴处于离散运行状态。在到达目标位置后，速度为零，此时转为 StandStill 状态。

Continuous Motion 状态，即连续运行状态。控制单个轴连续运行，轴会连续运行，直至被其它指令打断。

Stopping 状态，即停止运行状态。在此状态中，轴会从当前速度进行减速运行，当速度减到零时，运动

ErrorStop 状态，即错误状态。在轴运行时，如果当前运行参数值，超过系统设置的极限值，轴会进入错误状态，此状态必须通过 MC_Reset 指令进行改变，否则功能块将无法执行。

Homing 状态，即回原点状态。在轴回参考点运动执行时，轴会进入 Homing 状态，当回到原点后，轴状态跃迁为 StandStill 状态。

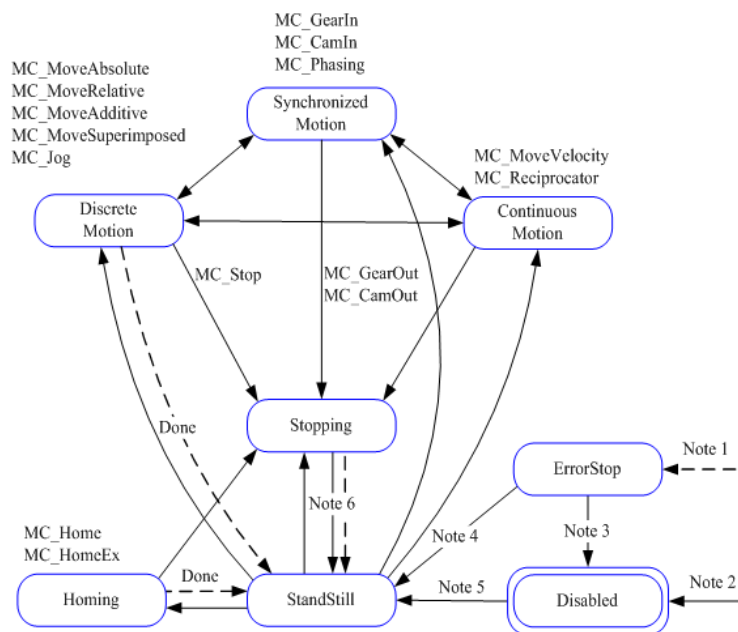


图 7.2.1 轴状态跃迁图

Note2: 进入 Dsiabled 状态触发条件为运动控制参数是否使用设置为 FALSE:

Note3: 由 ErrorStop 跃迁到 Disabled 状态触发条件为复位轴 MC_Reset, 且运动控制参数是否使用设置为 FALSE;

Note4: 由 ErrorStop 跃迁到 StandStill 状态触发条件为复位轴 MC_Reset, 且运动控制参数是否使用设置为 “1”;

Note5: 由 Disabled 跃迁到 StandStill 状态触发条件为运动控制参数是否使用设置为 TRUE;

Note6: 由 Stopping 跃迁到 StandStill 状态触发条件为停止轴 MC Stop 指令执行完成。

轴组状态是轴组指令运动执行过程中的运行、停止等状态信息。在轴组中的轴组合为一个整体，与轴组的运行状态一致。字变量、常数图 7.2.2 为轴组状态跃迁图，主要包括轴组禁能状态 GroupDisabled、轴组暂停状态 GroupStandby、轴组错误状态 GroupErrorStop、轴组停止状态 GroupStopping、轴组运行状态 GroupMoving 等状态。

GroupDisabled 状态：即轴组禁能状态。轴组的运行控制被禁止，任何轴组指令都无法执行，此时只有调用 MC_GroupEnable 指令，轴组才被使能。

GroupStandby 状态：即轴组停止状态。轴组初始化后，进入停止状态，或者轴组运行停止，也进入该状态。

GroupErrorStop 状态：即轴组错误状态。在轴组运行时，如果当前运行参数值，超过系统设置的极限值，轴组会进入错误状态。

GroupMoving 状态：即轴运行状态。在轴组执行绝对直线运行、绝对样条运行，圆弧曲线运行等多轴联动指令时，轴组就进入实时插补与控制的运动状态。

GroupStopping 状态：即轴停止状态。在此状态中，轴组会从当前速度进行减速运行，当速度减到零时，运行结束，轴组在执行轴组复位 MC_GroupReset 指令后状态跃迁为 GroupStandby 状态，若停止后出现错误，

则进入 GroupErrorStop 状态。

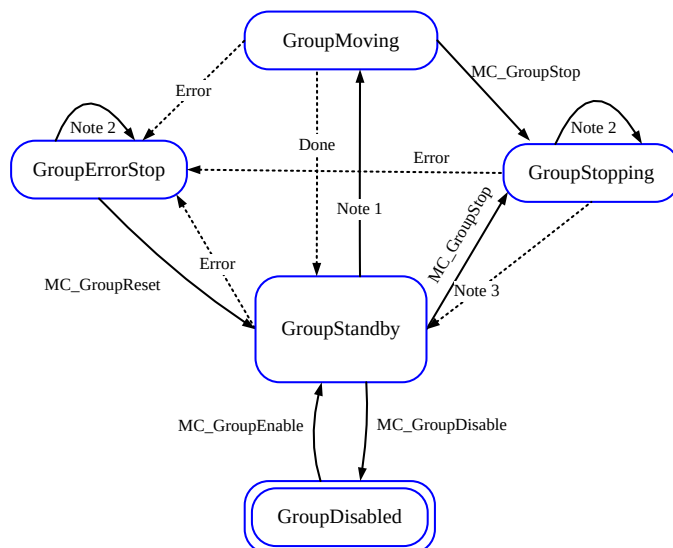


图 7.2.2 轴组状态跃迁图

Note1: 由 GroupStandby 跃迁到 GroupMoving 状态触发条件为所有轴组运动指令，如绝对执行运行 MC_MoveLinearAbsolute 指令、绝对样条运行 MC_MoveSplineAbsolute 指令、平面圆弧运行 MC_MoveCircular2D 指令等；

Note2: GroupStopping 和 GroupErrorStop 状态下，除了出现 Error 错误信息和执行 MC_GroupReset 指令外，调用其他轴组指令不会更改其状态；

Note3: 由 GroupStopping 跃迁到 GroupStandby 状态触发条件为停止轴组 MC_GroupStop 指令执行完成。

7.3 运动控制参数

使用运动控制类指令前需要对相应的轴或轴组进行运动控制参数配置。通过运动控制参数配置，能够配置轴和轴组的基本信息。

PLC_Config 软件提供运动控制参数设置，可在软件的工程树下直接进入。

运动控制参数配置完成后，需下载到设备中，重新上电后生效。同样，通过上载参数能够读取设备中的运动控制信息。

在“运动控制参数配置”界面中需根据工程的要求，进行轴和轴组的数量配置。程序会根据配置的轴和轴组的数量信息，生成对应数量的详细配置界面。

7.3.1 单轴参数配置

轴参数的功能、设定范围以及初始值如表 7.3.1 所示

表 7.3.1 单轴参数

参数名称	功能	设定范围	默认值
轴编号	轴编号用于表示轴的 ID	0,1,2,3...	根据选择轴的数量编号依次递增
是否使用	是否使用选项能够使能和禁能该轴	TRUE 或 FALSE	TRUE
单位	位移参数的单位，以 units 表示	脉冲，毫米，微米，纳米，度，英尺	脉冲
伺服单周脉冲	伺服电机旋转 1 周的脉冲个数	$1 \sim 1.8 \times 10^{19}$	10000 脉冲
伺服单周位移	伺服电机旋转 1 周的机构位移量	$1 \times 10^{-9} \sim 1.8 \times 10^{19}$	10000 units
最大速度	设置轴的最高速度，需为正实数	正实数	4×10^8 units/s
点动最大速度	单轴点动的最大速度，需为正实数	正实数	4×10^8 units/s
最大加速度	最大加速度。置 0 时没有加速度限制	≥ 0 的实数	4×10^8 units/s ²
最大减速度	最大减速度。置 0 时没有减速度限制	≥ 0 的实数	4×10^8 units/s ²
正向位置软件限制	软件限制的轴运行的正向最大位置	实数	4×10^9 units
负向位置软件限制	负向位置软件限制，需为实数	实数	-4×10^9 units
回原点动作模式	回原点动作模式设置，回原点指令 MC_Home	接近信号 OFF (反	接近信号 OFF (反

	执行前, 需在此处设置回原点动作模式	转), 接近信号 ON (反转), 接近信号 OFF, 接近信号 ON, 限位信号 OFF, 原点信号屏蔽, 只用限位信号, 只用原点信号, 无接近信号	转)
回原点开始方向	回原点运动的起始运动方向	正向、反向	正向
原点搜索方向	开始搜索原点的运动方向	正向、反向	正向
回原点速度	开始回原点运动的速度	正实数	10000 units/s
接近原点速度	接近原点时的速度	正实数	1000 units/s
回原点加速度	回原点加速度	正实数	10000 units/s ²
回原点减速度	回原点减速度	正实数	10000 units/s ²
原点位置偏移量	与原点位置的偏移量	实数	0 units
原点检测屏蔽距离	原点信号屏蔽模式中的原点检测屏蔽距离	正实数	10000 units
原点附近输入信号	原点附近输入信号为当轴运动到接近原点位置时的输入信号	所有 DI	I00
原点附近输入信号种类	常态下原点附近输入信号开闭状态	常开、常闭	常开
原点输入信号	原点输入信号为当轴运动到达原点位置时的输入信号	所有 DI	I01
原点输入信号种类	常态下原点输入信号开闭状态	常开、常闭	常开
正方向限位输入信号	轴运动到正方向最大位置时的输入信号	所有 DI	I02
正方向限位输入信号种类	常态下正方向限位输入信号开闭状态	常开、常闭	常开
负方向限位输入信号	轴运动到负方向最大位置时的输入信号	所有 DI	I03
负方向限位输入信号种类	常态下负方向限位输入信号种类	常开、常闭	常开

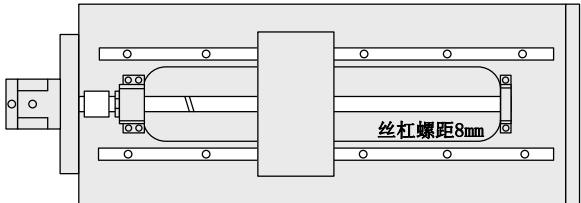
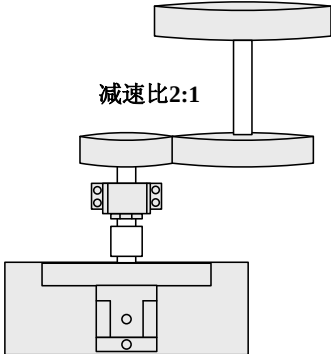
注 1: 在“运动控制参数配置”中, 需要对参数单位进行选择。可选的单位有脉冲、毫米、微米、纳米、度和英尺等。若单位选择“毫米”, 则表示位移的单位为毫米, 速度的单位为毫米/秒, 加速度的单位为毫米/秒², 减速度的单位为毫米/秒², 加加速度的单位为毫米/秒³, 运动控制中的其他相应参数单位均以该单位为基准。

注 2: 在“运动控制参数配置”中, 可以通过配置伺服单周脉冲和伺服单周位移, 实现与驱动器端对应的电子齿轮比功能。其中伺服单周脉冲表示为伺服电机旋转一周所需脉冲数, 伺服单周位移表示伺服电机旋转一周运动控制平台的实际位移, 位移的单位为运动控制中的单位。电子齿轮比为伺服单周脉冲与伺服单周位移的比值, 表示伺服电机运动单位位移量所需的脉冲数。计算公式如下:

$$\text{电子齿轮比} = \frac{\text{伺服单周脉冲}}{\text{伺服单周位移}}$$

表 7.3.2 给出了直线运动平台和旋转运动平台两种情况下电子齿轮计算和参数设置过程。

表 7.3.2 电子齿轮比设计实例

直线运动平台	旋转运动平台
	
已知: 滚珠丝杠螺距 $L=8\text{mm}$, 减速比 1:1, 伺服电机单周脉冲 $P=10000$, 指令单位为微米。 则: 伺服电机单周位移	已知: 减速比 2:1, 伺服电机单周脉冲 $P=10000$, 指令单位为度。 则: 伺服电机单周位移

$S = 1 \times 8 \times 1000 \times \frac{1}{1} = 8000 (um)$ 电子齿轮比 = $\frac{P}{S} = \frac{10000}{8000}$ 轴参数设置如下： 伺服单周脉冲：10000 伺服单周位移：8000	$S = 1 \times 360 \times \frac{1}{2}$ 电子齿轮比[1] = $\frac{P}{S} = \frac{10000}{360} \times \frac{2}{1} = \frac{20000}{360}$ 轴参数设置如下： 伺服单周脉冲：20000 伺服单周位移：360

注 1：计算电子齿轮比时，分子与分母尽可能为整数，若超过参数设定范围，请更换指令单位或约分为设定范围的整数。

7.3.2 轴组参数配置

在“运动控制参数配置”中，可对选择的轴组进行轴组参数的详细配置。轴组参数的功能、设定范围以及初始值如表 7.3.3 所示。

表 7.3.3 轴组参数

参数名称	功能	设定范围	初始值
轴组编号	轴组编号用于表示轴组的 ID	0,1,……	根据选择轴组的数量依次递增
是否使用	使能和禁能该轴组	TRUE 或 FALSE	FALSE
组成轴组的轴	添加到轴组中的轴的编号	0,1,……	0,1,2,3
最大插补速度	轴组运动的最大速度	正实数	4×10^8 units/s
最大插补加速度	轴组运动的最大加速度	正实数	4×10^8 units/s ²
最大插补减速度	轴组运动的最大减速度	正实数	4×10^8 units/s ²
运动模型	提供轴组运动模型种类	直角坐标系，圆柱坐标系，四轴并联机器人，三轴并联机器人，二轴并联机器人、三轴座椅，二轴座椅、SCARA 模型、通用模型	直角坐标系

7.3.3 轴组运动模型参数配置

运动控制参数配置中，提供运动模型的选择与配置。模型与轴组关联，每个轴组同一时间只能选择一种模型。轴组支持的模型有直角坐标系模型、圆柱坐标系模型、四轴并联机器人模型、三轴并联机器人模型、二轴并联机器人、三轴座椅，二轴座椅、SCARA 模型、通用模型等，默认为直角坐标系模型。选择完运动模型后，可对模型关联的参数进行详细配置。

● 直角坐标系模型

主要应用于空间垂直的三轴机械结构，提供 X, Y, Z 三个维度和一个旋转轴 A 的运动。用户输入空间位置 (X, Y, Z, A) 后，经过直角坐标系模型变换，转换为对应的 X, Y, Z, A 四个轴上的脉冲输出。

建立的直角坐标系符合右手定则，各轴对应关系如图 7.3.1 所示。

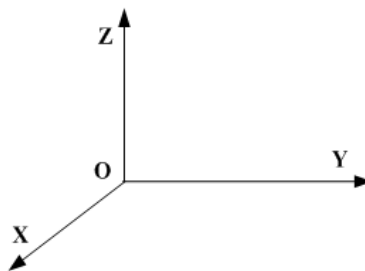


图 7.3.1 直接坐标系

● 圆柱坐标系模型

主要应用于圆形或类圆形绕轴心运动机构，提供 θ 、 ρ 、Z 三个维度和一个旋转轴 A 的运动。用户输

入空间位置 (X, Y, Z, A) 后, 经过圆柱坐标系模型变换, 转换为对应的 θ 、 ρ 、Z, A 四个轴上的脉冲输出。圆柱坐标系的对应关系如图 7.3.2 所示。

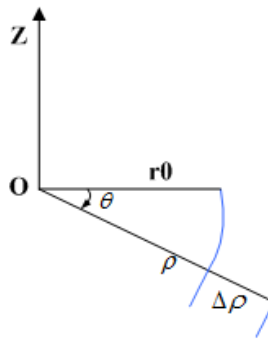


图 7.3.2 圆柱坐标系

圆柱坐标系模型参数的功能、设定范围以及初始值描述如表 7.3.4 所示。

表 7.3.4 圆柱坐标系模型参数

参数名称	功能	设定范围	初始值
初始半径值	初始半径值 r_0 ，如果实际半径 $r > r_0$ ，则输出的 r 值为正，如果 $r < r_0$ ，则输出的 r 值为负，如果 $r = r_0$ ，则输出的 r 值为 0。单位 mm	$0 \sim 1.8 \times 10^{19}$	0mm

● 四轴并联机器人模型

四轴并联机器人模型适用于机器人的支链和运动平台均对称均匀布置的四轴机器人。该模型实现空间三维平动和一维转动，在工作空间内无奇异位置且受力均匀。四轴并联机器人的实体如图 7.3.3 所示。

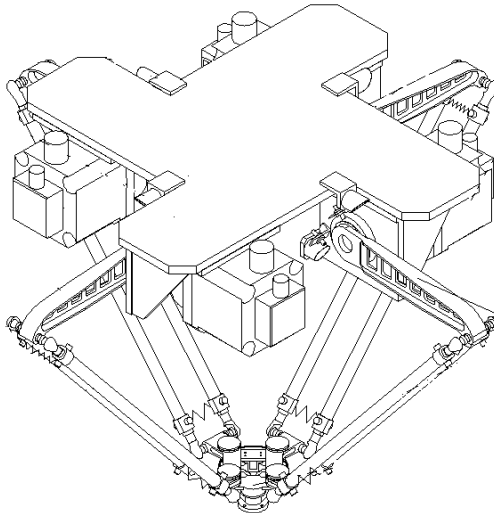


图 7.3.3 四轴并联机器人模型整体图

四轴并联机器人模型分解如图 7.3.4 所示。图中 R 为静平台的半径 (P_i 到 O 的距离), H 为动平台的长度 (C_3 与 C_4 之间距离), D 为动平台的宽度 (C_1 与 C_3 之间距离), H_1 为原点长边垂直距离 (B_2 到直线 C_1C_2 之间距离), D_1 为原点短边垂直距离 (B_2 到直线 C_2C_4 之间距离), M 为原点动平台静平台垂直距离 (P_i 与 B_i 之间的距离), L 为主动臂长度, l 为从动臂长度。

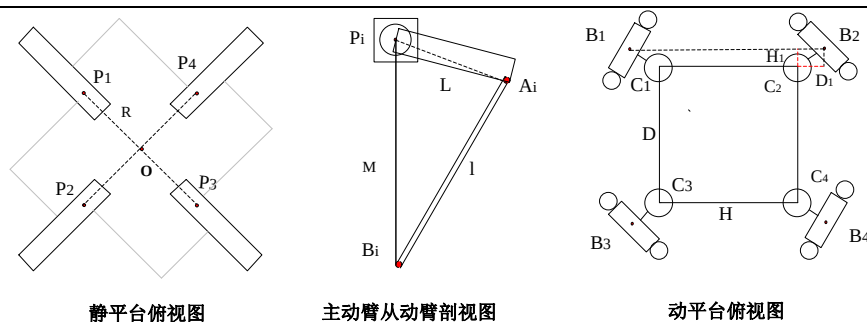


图 7.3.4 四轴并联机器人模型分解图

四轴并联机器人模型参数的功能、设定范围以及初始值描述如表 7.3.5 所示。

表 7.3.5 四轴并联机器人模型参数

参数名称	功能	设定范围	初始值
静平台半径 R	电机轴中心到静平台中心的距离, 单位 mm	正实数	275mm
动平台长 H	动平台上矩形的长边长, 单位 mm	正实数	100mm
动平台宽 D	动平台上矩形的短边长, 单位 mm	正实数	100mm
原点长边垂直距离 H1	原点时动平台上球角中心到矩形的长边的垂直距离, 单位 mm	正实数	28.13mm
原点短边垂直距离 D1	原点时动平台上球角中心到矩形的宽边的垂直距离, 单位 mm	正实数	28.13mm
原点动平台静平台垂直距离 M	原点位置时动平台与静平台的垂直距离, 单位 mm	正实数	650.36mm
主动臂长度 L	主动臂长度, 单位 mm	正实数	375mm
从动臂长度 I	从动臂长度, 单位 mm	正实数	845mm

● 三轴并联机器人模型

三轴并联机器人模型适用于机器人的支链和运动平台均对称均匀布置的三轴机器人。如图 7.3.5 所示, 该模型实现空间三维平动, 在工作空间内无奇异位置且受力均匀。

该模型提供三个电机的旋转运动。用户输入空间位置 (X, Y, Z) 后, 经过并联机器人模型变换, 转换为对应的三个轴上的脉冲输出。

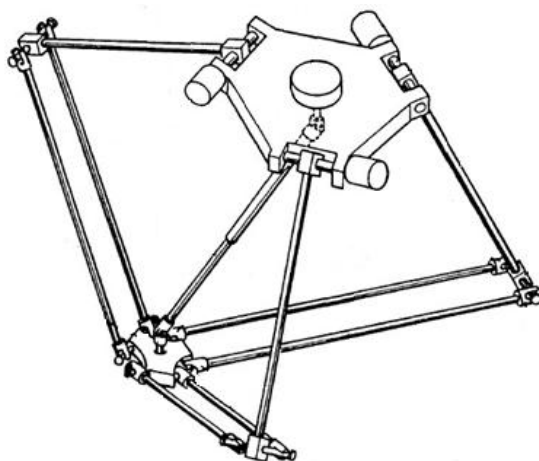


图 7.3.5 三轴并联机器人模型整体图

三轴并联机器人模型分解图如图 7.3.6 所示。(a) 为静平台俯视图, P_1 、 P_2 、 P_3 分别为主动臂, R 为静平台的半径。(b) 为主动臂从动臂剖视图, P_1 为某一条主动臂, L_0 为主动臂水平投影, h 为主动臂垂直投影, l 为从动臂长度。(c) 为动平台俯视图, a_1 、 a_2 、 a_3 分别为主动臂 P_3 、 P_1 、 P_2 到 X 轴正方向的夹角, r_0 为动平台半径。

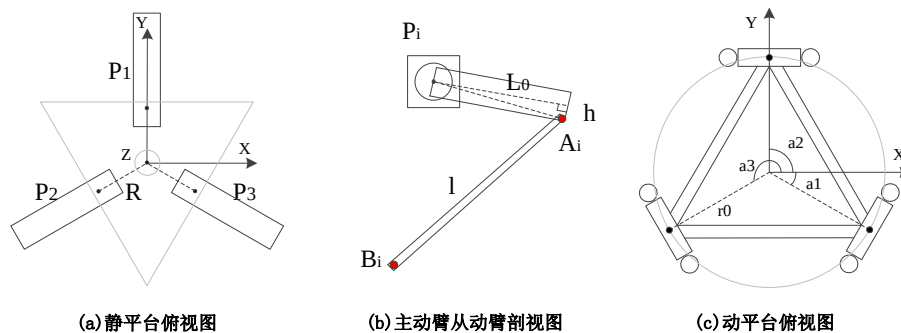


图 7.3.6 三轴并联机器人模型分解图

三轴并联机器人模型参数的功能、设定范围以及初始值描述如表 7.3.6 所示。

表 7.3.6 三轴并联机器人模型参数

参数名称	功能	设定范围	初始值
主动臂到角 a_1 、 a_2 、 a_3	主动臂在水平面投影线与 x 轴正向的到角, 单位度	-90~270	-30 度
动平台半径 r_0	动平台外圆半径, 单位 mm	正实数	50mm
静平台半径 R	静平台上电机安装半径, 单位 mm	正实数	200mm
主动臂水平投影 L_0	主动臂水平时, 球关节轴线相对于电机轴线的水平长度, 单位 mm	正实数	350mm
主动臂垂直投影 h	主动臂水平时, 球关节轴线相对于电机轴线的垂直高度, 单位 mm	正实数	35mm
从动臂长度 l	从动臂长度, 单位 mm	正实数	850mm

● 通用模型

通用模型由用户编写程序设定模型的变换关系, 操作灵活。用户编程写入空间位置 (该值由轴组中轴个数决定, 例如三轴输入 X, Y, Z), 按照所设定的变换关系, 转换为对应轴上的脉冲输出。以下为通用模型应用示例。

在“工程设备列表”中选择“函数”, 创建新程序, 如图 7.3.7 所示。

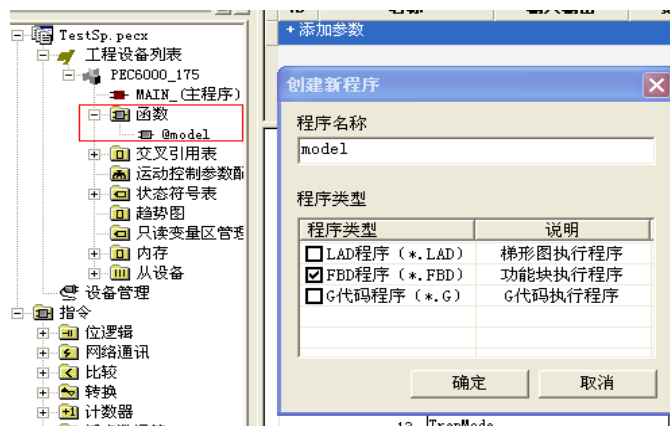


图 7.3.7 通用模型函数创建

在该程序中首先定义运动模型的输入变量与输出变量 (输入或输出变量个数与轴组中轴个数相同), 然后编写程序实现输入变量与输出变量的变换, 以建立通用模型。程序段编写可参图 7.3.8。

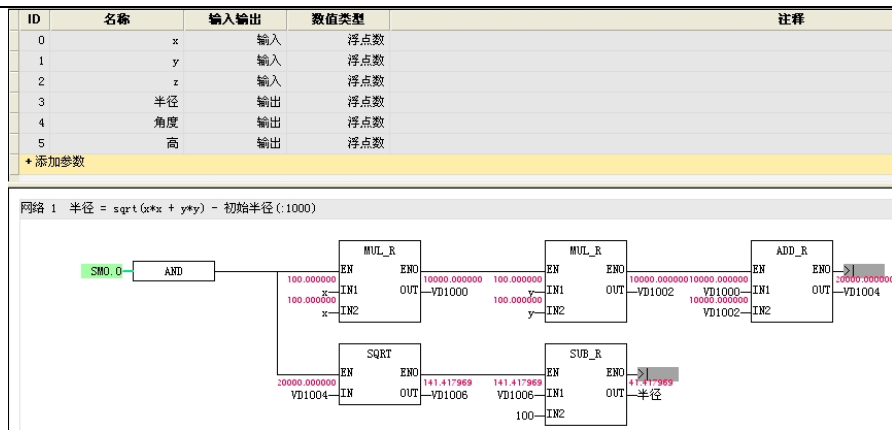


图 7.3.8 通用模型变换公式创建

在“运动控制参数配置”中的“运动模型”选项中选择“通用模型”，并选择所建立的模型转换函数。设置成功后，轴组运动类指令执行时都按照该模型进行转换。

注：在使用通用模型时，需注意以下几项：

- (1) 通用模型函数不支持嵌套调用；
- (2) 选择通用模型后，需要重新下载组态程序才能生效；
- (3) 模型转换函数只能含有输入输出参数，并且输入和输出数量相等；
- (4) 模型转换函数所有输入输出参数均为浮点数，且数量均等于轴组中轴的个数。

7.4 运动控制状态监控

运动控制状态监视，能够监视轴和轴组的状态信息。在 PLC_Config 软件中，选择“运动控制参数监控”可进入监视状态。

轴状态监视中监视的参数意义、参数类型如表 7.4.1 所示。

表 7.4.1 轴状态监视

参数名称	参数意义	参数类型
轴编号	表示轴的 ID	WORD
禁能状态	轴处于禁能状态	BOOL
停止状态	轴处于停止状态	BOOL
离散运动	轴处于离散运动(绝对、相对运动)状态	BOOL
连续运动	轴处于连续运动状态	BOOL
同步运动	轴处于同步运动状态	BOOL
回原点中	正在利用 MC_Home 或 MC_HomeEx 指令回原点	BOOL
减速停止中	正在利用 MC_Stop 指令进行减速停止	BOOL
减速停止中	正在利用 MC_Stop 指令进行减速停止	BOOL
异常停止	急停中和急停、异常停止	BOOL
准备就绪	轴准备就绪并处于停止状态	BOOL
被轴组使用	轴正在轴组中使用	BOOL
原点确定	原点确定后置位	BOOL
接近原点	正在接近原点	BOOL
达到速度上限	同步运动时轴速度达到设定的最大速度	BOOL
正方向	轴的运动方向为正	BOOL
负方向	轴的运动方向为负	BOOL
位置	轴运动的目标位置	REAL
速度	轴运动的目标速度	REAL
加速度	轴运动的目标加速度	REAL
减速度	轴运动的目标减速度	REAL
位置	运动中的实际位置	REAL
速度	运动中的实际速度	REAL
是否监视	是否对轴进行异常监视	BOOL
异常代码	异常发生时的异常代码	WORD

轴组状态监视中监视的参数意义、参数类型如表 7.4.2 所示。

表 7.4.2 轴组状态监视

参数名称	参数意义	数据类型
轴组编号	轴组编号用于表示轴组的 ID	WORD
禁能状态	轴组处于禁能状态	BOOL
停止状态	轴组处于停止状态	BOOL
运动中	正在向目标位置运动	BOOL
停止中	正在执行 MC_GroupStop 指令	BOOL
异常停止	急停中和急停、异常停止	BOOL
准备就绪	轴准备就绪并处于停止状态	BOOL
速度	轴组运行的目标速度	REAL
加速度	轴组运行的目标加速度	REAL
减速度	轴组运行的目标减速度	REAL
是否监视	是否对轴组进行异常监视	BOOL
异常代码	异常发生时的异常代码	WORD
动力学模型种类	0:直角坐标系,1:圆柱坐标系,2:四轴并联模型,3:三轴并联模型,4:三轴座椅模型,0xFFFF:通用模型	WORD
第一个轴编号	轴组中第一个轴的轴编号	WORD
第二个轴编号	轴组中第二个轴的轴编号	WORD
第三个轴编号	轴组中第三个轴的轴编号	WORD
第四个轴编号	轴组中第四个轴的轴编号	WORD

7.5 运动控制指令概述

本章主要从运动控制指令描述和编程两个方面进行介绍。指令编程主要介绍运动控制指令触发方式、引脚时序、指令参数初始化、速度时间曲线参数配置以及运动曲线等内容。在指令描述中分单轴指令、多轴指令和轴组指令，每一类指令从指令引脚说明、时序图、编程举例等方面进行详细说明。

7.5.1 指令触发方式

在指令的使用中，对于指令的触发方式，根据其使能引脚名称可分为高电平触发和上升沿触发两种。

● 高电平触发

如图 7.5.1 所示，高电平触发方式使能引脚的名称为 Enable。当 Enable 引脚为“1”时，指令使能，当 Enable 引脚为“0”时，指令禁能。

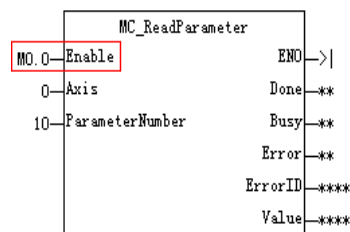


图 7.5.1 指令的高电平触发

● 上升沿触发

如图 7.5.2 所示，上升沿触发方式使能引脚的名称为 Execute。当 Execute 引脚的上升沿产生时，指令使能，读取输入变量，执行指令。其他情况均不执行指令体。运动类指令执行完成后，若 Execute 仍为“1”，则输出的状态值是有效的，若 Execute 被置为“0”，则输出的状态值只在执行完成的一个周期内有效。

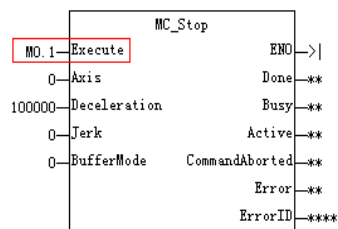


图 7.5.2 指令的上升沿触发

7.5.2 指令引脚时序

对于具有输出引脚 Done、Busy、CommandAborted 及 Error 的运动类指令，其输出引脚信号状态会受到能流输入引脚 Execute 的影响，下面以运动完全执行、运动被打断和运行出错三种情况为例进行说明。

运动完全执行后，指令输出引脚 Done 被置“1”。能流输入引脚 Execute 的信号状态影响输出引脚 Done 的显示持续时间，时序如图 7.5.3 和表 7.5.1 所示。

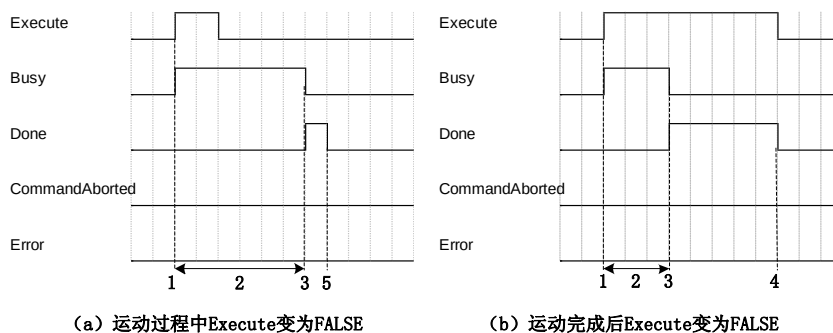


图 7.5.3 运动执行完成时的引脚时序

表 7.5.1 运动执行完成时的引脚信号变化

NO.	说明
1	能流输入引脚 Execute 上升沿时启动运动控制。根据编程情况，Execute 引脚在运动过程中可能被置“0”，或者保持为“1”
2	运动过程中 Busy 引脚为“1”
3	运动完成后，Busy 引脚变为“0”，Done 引脚变为“1”
4	如果 Execute 引脚在运动完成之前保持“1”，则 Done 引脚保持为“1”并随 Execute 一起变为“0”
5	如果 Execute 引脚在运动完成之前置为“0”，则 Done 引脚仅在一个执行周期内为“1”

运动过程中的指令被其他指令打断，指令输出引脚 CommandAborted 变为 TRUE。能流输入引脚 Execute 的信号状态影响输出引脚 CommandAborted 的显示持续时间，时序如图 7.5.4 和表 7.5.2 所示。

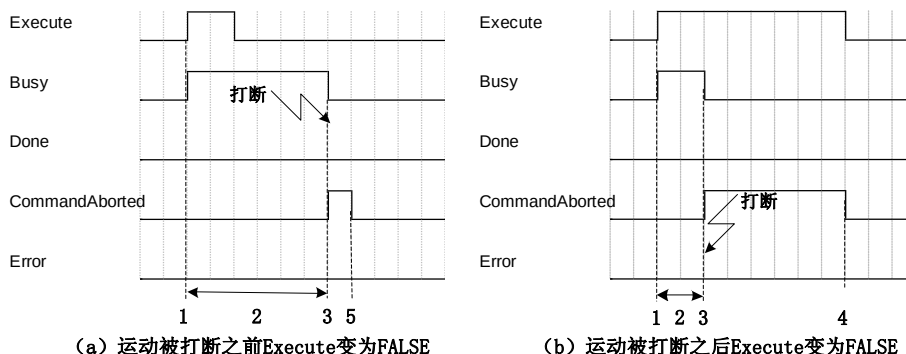


图 7.5.4 运动被打断时的引脚时序

表 7.5.2 运动被打断时的引脚信号变化

NO.	说明
1	能流输入引脚 Execute 上升沿时启动运动控制。根据编程情况，Execute 引脚在运动过程中可能

	被置为“0”，或者保持为“1”
2	运动过程中 Busy 引脚为“1”
3	运动过程中，当前指令被其他指令打断，Busy 引脚变为“0”，CommandAborted 引脚变为“1”
4	如果 Execute 引脚在运动被打断之前保持“1”，则 CommandAborted 引脚保持为“1”并随 Execute 一起变为“0”
5	如果 Execute 引脚在运动被打断之前置为“0”，则 CommandAborted 引脚仅在一个执行周期内为“1”

运动过程中运行出错时，指令输出引脚 Error 变为“1”。能流输入引脚 Execute 的信号状态影响输出引脚 Error 的显示持续时间，时序如图 7.5.5 和 0 所示。

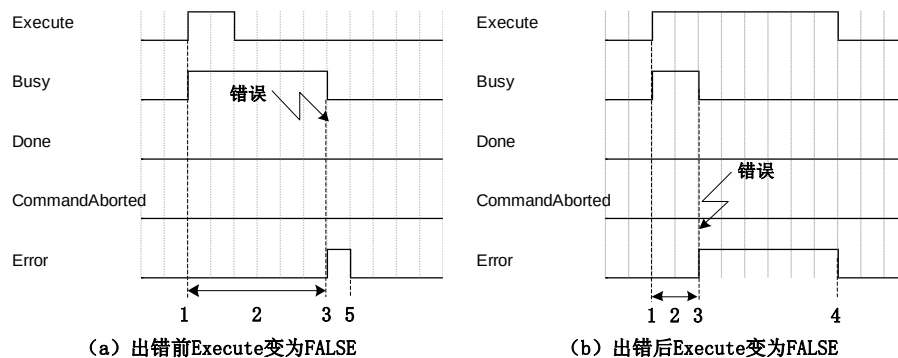


图 7.5.5 运行出错时的引脚时序

表 7.5.3 运行出错时的引脚信号变化

NO.	说明
1	能流输入引脚 Execute 上升沿时启动运动控制。根据编程情况，Execute 引脚在运动过程中可能被置为“0”，或者保持为“1”
2	运动过程中 Busy 引脚为“1”
3	运行出错时，Busy 引脚变为“0”，Error 引脚变为“1”
4	如果 Execute 引脚在运行出错之前保持“1”，则 Error 引脚保持为“1”并随 Execute 一起变为“0”
5	如果 Execute 引脚在运行出错之前置为“0”，则 Error 引脚仅在一个执行周期内为“1”

7.5.3 指令引脚参数

在指令参数初始化的过程中，指令左侧为输入引脚，指令右侧为输出引脚。输入引脚和输出引脚各自都包含自动分配引脚和非自动分配引脚。

输入自动分配引脚。指令中对于轴 ID、速度、加速度、位移、方向等引脚提供了默认的初始值，用户可根据实际需求更改这些输入值为其他常量或寄存器值。

输入非自动分配引脚。能流输入、地址参数、数组类型参数等引脚需要用户按照指令参数中的“操作数有效范围和长度属性”进行寄存器的分配，初始时用“???”表示。

输出自动分配引脚。对于状态输出引脚默认表示为“**”，但“**”表示的输出值仅可用于监视，如用户需要在寄存器中体现输出值，则需按照指令的“操作数有效范围和长度属性”进行寄存器分配。

输出非自动分配引脚。用户可根据实际需求分配寄存器，将当前指令的输出值使用在程序中的其他地方。

7.5.4 加减速控制曲线

单轴和轴组的加减速控制方式主要有：梯形曲线、指数曲线、S 形曲线和变加加速曲线。

● 梯形曲线

轴按照梯形曲线运动时，需要设定的运动参数有：加速度、减速度、目标速度。如果当前速度小于目标速度，则通过加速 - 恒速 - 减速三个阶段运行到目标位置；如果当前速度大于目标速度，则通过减速 - 恒速 - 加速三个阶段运行到目标位置；如果当前速度等于目标速度，则通过恒速 - 减速两个阶段运行到目标位置。如图 7.5.6 所示。

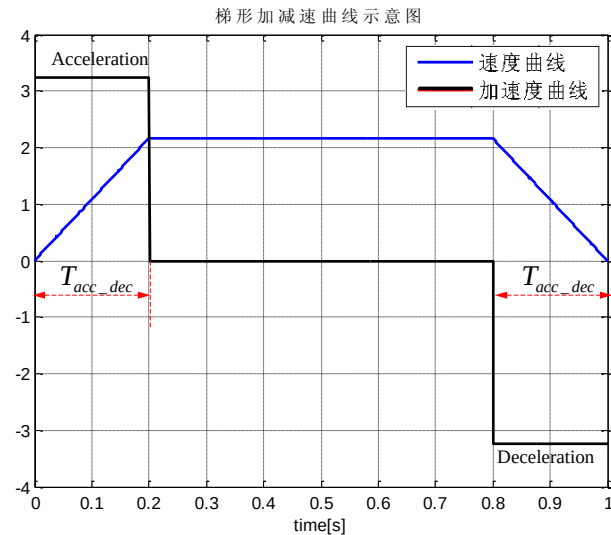


图 7.5.6 梯形曲线加减速控制

● 指数曲线

轴按照指数曲线运动时，需要设置运行目标速度和加减速时间，其中加减速时间为轴从初始速度到达目标速度的 63.2% 时的插补时间。如图 7.5.7 所示，指数曲线在轴加速开始和减速开始阶段速度变化较大，适用于启动速度要求较高的场合。

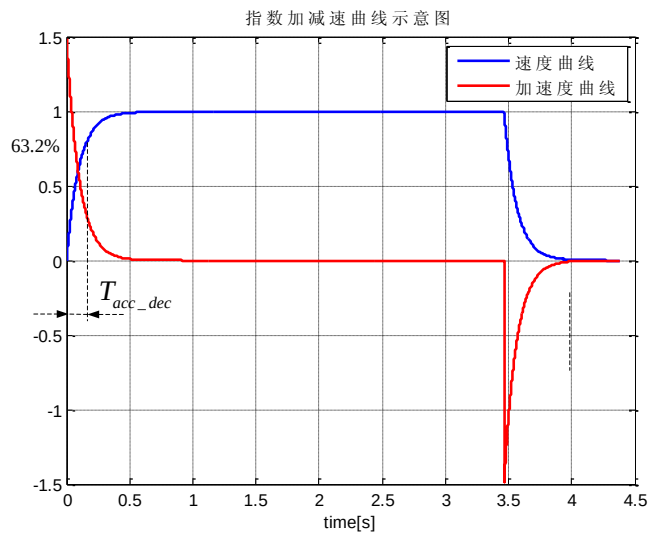


图 7.5.7 指数曲线加减速控制

● S 型曲线

轴按照 S 型曲线运动时，需要设定的运动参数有：加加（减减）速度、加速度、减速度、目标速度。如果当前速度小于目标速度，则通过加加速 - 恒加速 - 减加速 - 恒速 - 减加速 - 恒减速 - 减减速七个阶段运行到目标位置；如果当前速度大于目标速度，则通过减加速 - 恒减速 - 减减速 - 恒速 - 减加速 - 恒减速 - 减减速七个阶段运行到目标位置；如果当前速度等于目标速度，则通过恒速 - 减加速 - 恒减速 - 减减速四个阶段运行到目标位置。如图 7.5.8 所示。

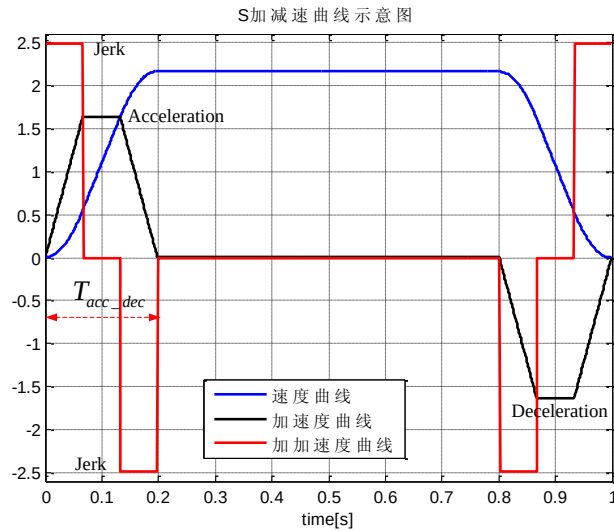


图 7.5.8 S 曲线加减速控制

● 变加加速曲线

轴按照变加加速运动时，需要设定目标速度和加减速时间如图 7.5.9 所示。为减少轴启动和停止的刚性冲击，对曲线加加速度进行加减速控制，保证加速度在启动和停止阶段平滑过渡。

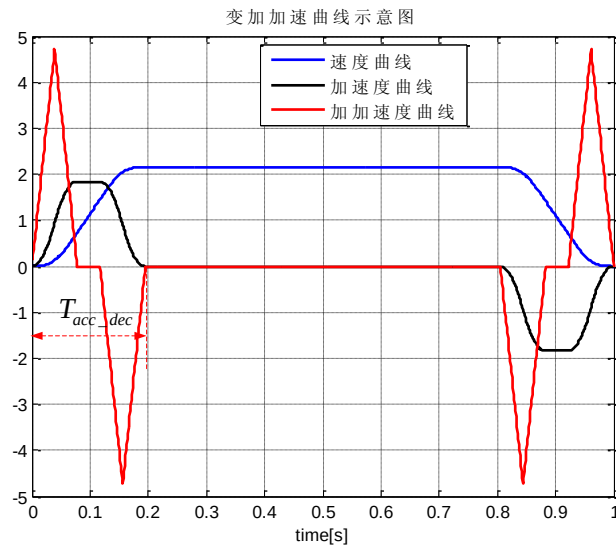


图 7.5.9 变加加速曲线加减速控制

针对运动类指令中的单轴指令和轴组指令，PLC_Config 软件提供了速度时间 VT 曲线配置功能。双击此类指令，在弹出的“VT 曲线配置”对话框中，用户可以通过设定“目标路程”、“预计时间”、“加速度”、“减速度”以及“加加速度”等选项，自动计算所需配置的速度，描绘出与设置值对应的 VT 曲线如图 7.5.10 所示。确定后，可将软件仿真的值分配到指令对应的引脚中。

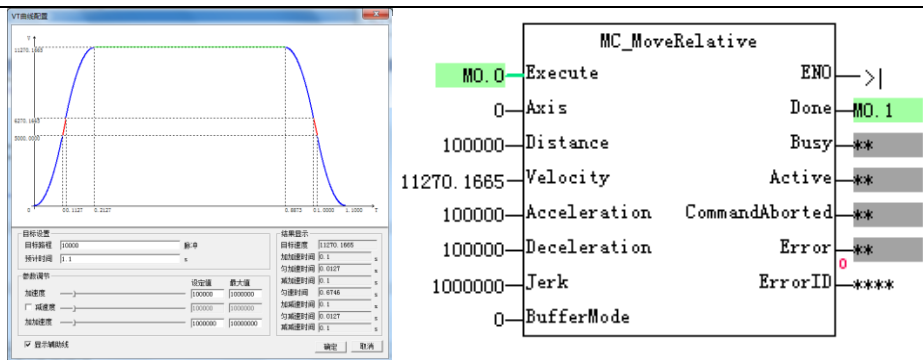


图 7.5.10 VT 曲线配置

7.5.5 缓冲模式

在多段曲线连续运行时，中止模式（默认模式）会先执行完第一段曲线，输出停止后，再执行下一段曲线。缓冲模式可以同时多段曲线进行预处理，在曲线连接处，会平滑的过渡到下一段曲线，中间不会停止如图 7.5.11 所示。两种模式通过指令的输入引脚 BufferMode 来进行选择，其中 0 为中止模式，其他值为缓冲模式。

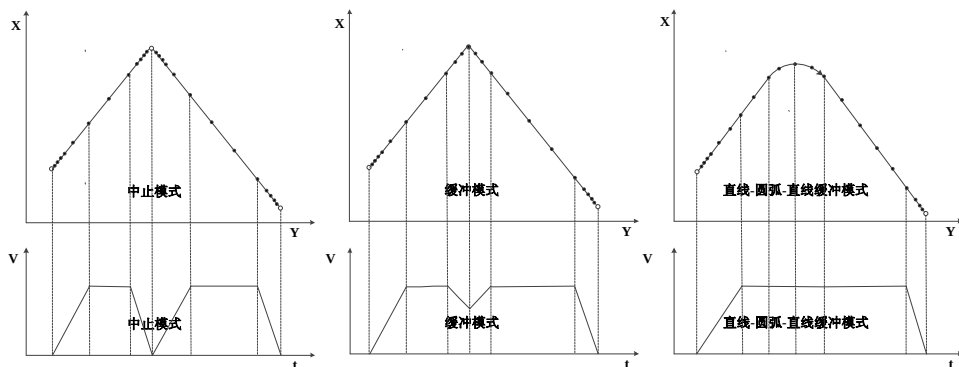


图 7.5.11 中止/缓冲模式示意图

缓冲模式将 BufferMode 值相同的指令绑定起来进行前瞻预处理，提前规划曲线衔接处轨迹并进行加减速控制。对于单轴指令缓冲模式还可以实现多个轴同步启动和同步停止控制。目前支持缓冲模式的指令如表 7.5.4 支持缓冲模式的指令表所示。

表 7.5.4 支持缓冲模式的指令表

单轴	轴组
MC_MoveAbsolute	MC_MoveLinearAbsolute
MC_MoveRelative	MC_MoveLinearRelative
--	MC_MoveCircular2D

7.5.5.1 单轴缓冲模式



MC_MoveAbsolute

<< ↑ Execute 双击配置曲线 ENO >|

0 Axis Done **

10000 Position Busy **

10000 Velocity Active **

100000 Acceleration CommandAborted **

100000 Deceleration Error **

0 Jerk ErrorID ****

0 Direction

0 BufferMode

第 302 页

CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）

● 详细说明

通过 MC_MoveAbsolute 指令，可以使轴运动到 Position 参数指定的目标位置。指令正在运行时，如果执行了控制相同轴的其他运动控制指令，此时该指令运行会被打断，CommandAborted 被置“1”。

当参数配置错误或指令执行错误时，Error 被置“1”，ErrorID 为错误码，根据错误码可查找到相应的错误，详见附录 E。

绝对定位指令支持缓冲模式，用来实现轴多段定位连续运动控制。指令使用过程中将 BufferMode 值相同的指令绑定起来进行前瞻预处理，提前规划曲线衔接处轨迹并进行加减速控制。对于 BufferMode 的 ID 大于 100 的指令，控制器会在程序扫描一轮后，同步插补输出脉冲，来实现多个轴同步启动和同步停止控制。关于缓冲模式使用请参考 7.5.5 缓冲模式章节。

● 时序图

图 7.6.1 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当位移达到目标值而且定位完成时 Done 被置“1”；如果有其它的指令打断当前指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID（错误码）值找到导致错误的原因。

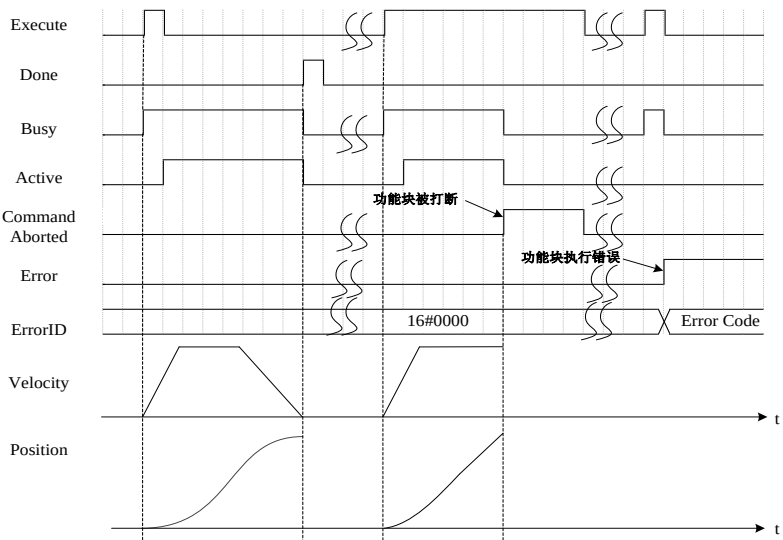


图 7.6.1 指令引脚时序和速度、位置变化图

● 应用示例

图 7.6.2 显示了在第一个绝对定位指令执行完成后调用第二个绝对定位指令的情况。第一个指令控制轴运动到目标位置 100000 units（同时速度减为 0）后，其 Done 引脚输出高电平信号使能第二个指令，控制轴 0 运动到目标位置 200000 units。

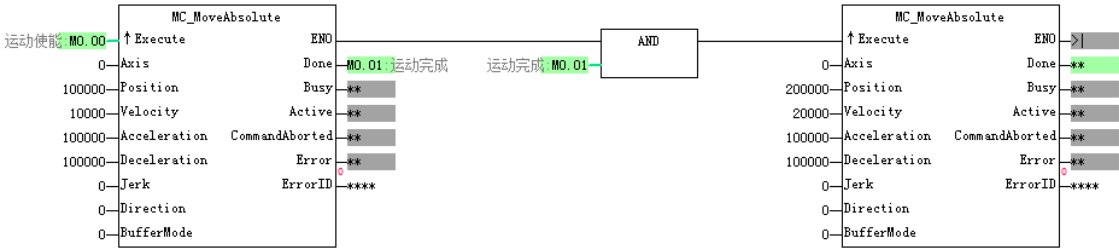


图 7.6.2 绝对定位运行

运动曲线如图 7.6.3 所示：

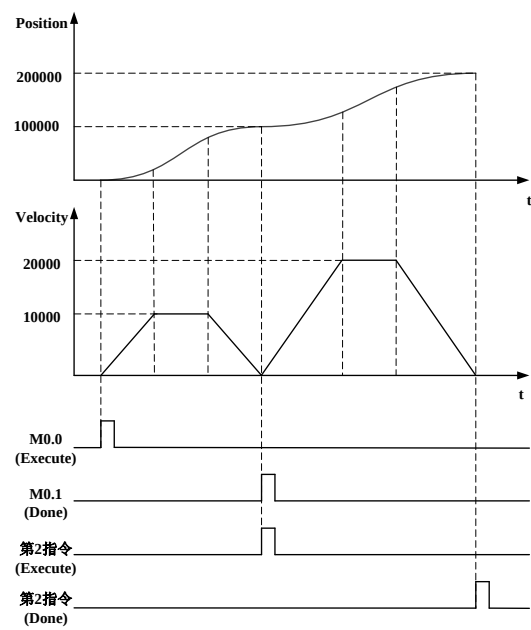


图 7.6.3 绝对定位运动曲线与时序图

图 7.6.4 显示了在第一个绝对定位指令执行过程中调用第二个绝对定位指令的情况。第一个指令控制轴 0 到目标位置 100000 units 之前，使能 M0.1 第二个指令打断第一个指令，第二个指令会控制轴 0 直接运动到目标位置 200000 units。

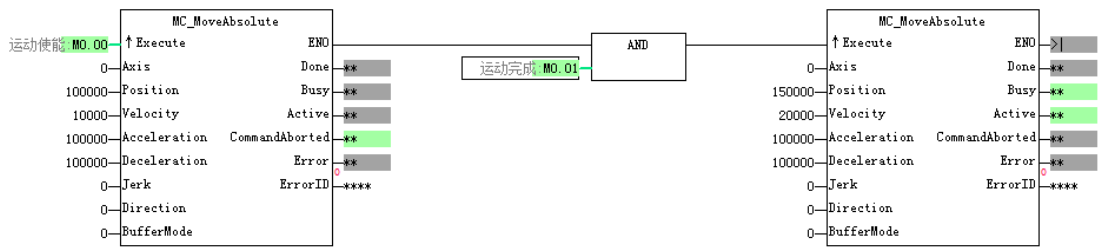


图 7.6.4 绝对定位运行被打断

运动曲线如图 7.6.5 所示：

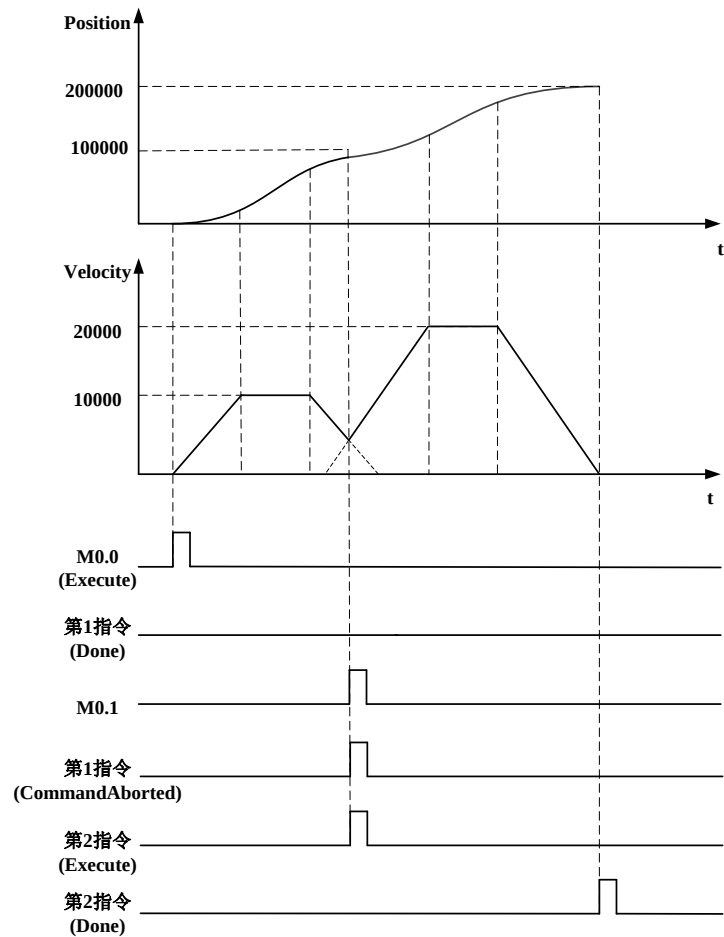


图 7.6.5 绝对定位被打断运行曲线与时序图

7.6.2 相对定位 MC_MoveRelative

● 指令参数

MC_MoveRelative << ↑ Execute 双击配置曲线 0 Axis 10000 Distance 10000 Velocity 100000 Acceleration 100000 Deceleration 0 Jerk 0 BufferMode EN0 > Done ** Busy ** Active ** CommandAborted ** Error ** ErrorID ****		控制轴从当前位置运动指定的距离，距离值为负时，表示向负方向运动； 指令使能后，轴的目标位置为轴的当前绝对位置加 Distance；速度以当前速度为起始速度，到达目标位置时速度为 0； 该指令支持“梯型曲线”和“S 型曲线”加减速控制方式。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
Distance	双字变量、常数	REAL	实数	10000	相对运行位置，单位 units
Velocity	双字变量、常数	REAL	正实数	10000	运行目标速度，单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；若为 0，则无加速过程，单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³

BufferMode	字变量、常数	WORD	非负整数	0	曲线运行模式选择 0: 中止模式, 此时 Done 输出时, 表示曲线运动停止 其他: 缓冲模式, 此时轴会持续运动一段时间后才停止
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时, 表示轴正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

通过 MC_MoveRelative 指令, 可以使轴运行 Distance 引脚参数指定的相对目标位置 (以当前位置作为起始位置)。指令正在运行时, 如果执行了控制相同轴的其他运动控制指令, 此时该指令运行会被打断, CommandAborted 被置“1”。

MC_MoveRelative 打断其它指令时, 在当前的位置叠加 Distance 距离做为目标位置, 继续运行。如前一个指令的目标位置为 1000 units, 而当前只运行到了 500 units 即被 MC_MoveRelative 指令打断, 如果 MC_MoveRelative 指令的 Distance 为 200, 那么轴会运行到 700 units 处。

当参数配置错误或指令执行错误时, Error 被置“1”, ErrorID 为错误码, 根据错误码可查找到相应的错误, 详见附录 E。

相对定位指令支持缓冲模式, 用来实现轴多段定位连续运动控制。指令使用过程中将 BufferMode 值相同的指令绑定起来进行前瞻预处理, 提前规划曲线衔接处轨迹并进行加减速控制。对于 BufferMode 的 ID 大于 100 的指令, 控制器会在程序扫描一轮后, 同步插补输出脉冲, 来实现多个轴同步启动和同步停止控制。关于缓冲模式使用请参考 7.5.5 缓冲模式章节。

● 时序图

图 7.6.6 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”, Active 在下一周期被置“1”; 当轴运动位移达到目标值而且定位完成时 Done 被置“1”; 如果有其它的指令打断此指令, CommandAborted 被置“1”, 同时 Busy 和 Active 被置“0”; 如果在指令执行期间有错误发生, 则 Error 被置“1”, 同时轴将停止, 可以通过 ErrorID(错误码)的值找到导致错误的原因。

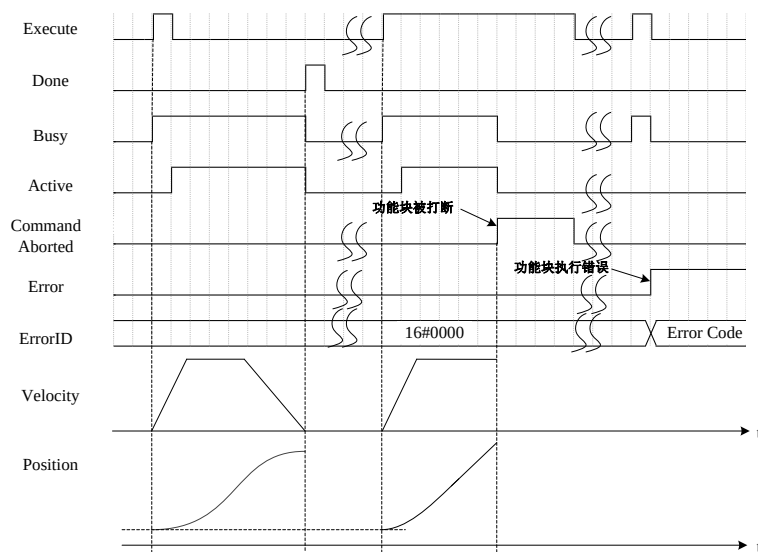


图 7.6.6 指令引脚时序和速度、位置变化图

● 应用示例

图 7.6.7 显示了在第一个指令执行完成后调用第二个指令的情况：第一个指令控制轴运动到目标位置 100000 units（同时速度减为 0）后，以当前位置为起始位置再运行 150000 units 的距离。

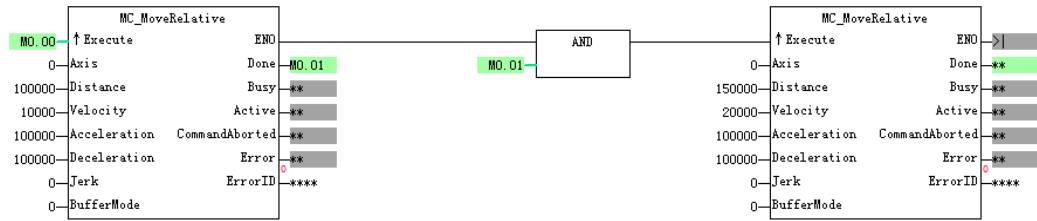


图 7.6.7 相对定位运行

运动曲线如图 7.6.8 所示：

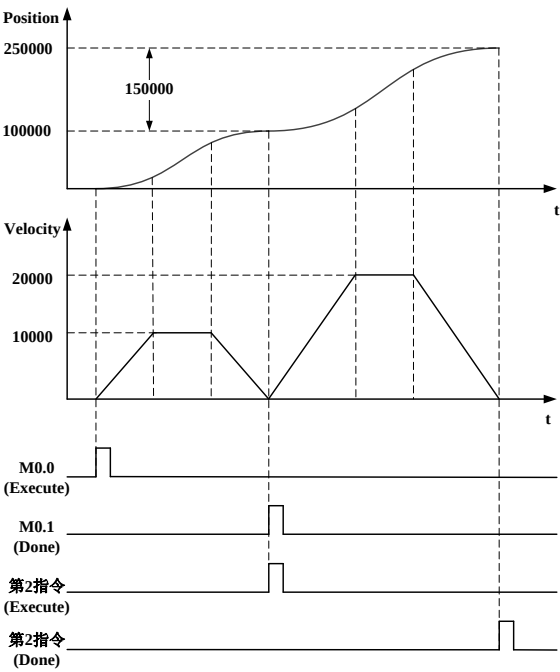


图 7.6.8 相对定位运行曲线与时序图

图 7.6.9 显示了在第一个指令执行过程中调用第二个指令的情况：第一个指令控制轴运动到目标位置 100000 units（同时速度减为 0）之前，被第二个指令打断，第二个指令会在当前实际位置基础上增加 150000 units 的距离作为目标位置。

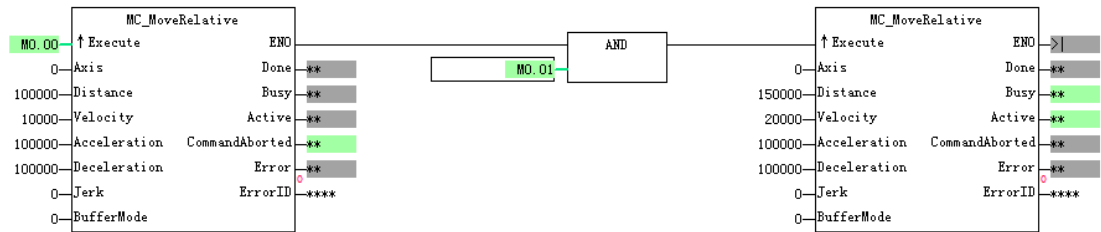


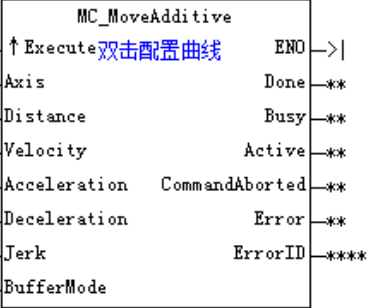
图 7.6.9 相对定位打断运行

运动曲线如图 7.6.10 所示：



7.6.3 叠加定位 MC_MoveAdditive

● 指令参数

 MC_MoveAdditive Execute 双击配置曲线 Axis Distance Velocity Acceleration Deceleration Jerk BufferMode Done Busy Active CommandAborted Error ErrorID			在轴的目标位置上叠加一段距离作为新的目标位置，控制轴运动到新的目标位置； 指令使能后，轴的目标位置为上个指令的目标位置加 Distance；速度以当前速度为起始速度，到达目标位置时速度为 0； 该指令支持“梯型曲线”和“S 型曲线”加减速控制方式。		
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
Distance	双字变量、常数	REAL	实数	10000	相对位置，单位 units
Velocity	双字变量、常数	REAL	正实数	10000	运行目标速度，单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；若为 0，则无加速过程，单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³

BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_MoveAdditive 指令，可以使轴的目标位置增加指定的距离。指令正在运行时，如果执行了控制相同轴的其他运动控制指令，此时该指令运行会被打断，CommandAborted 被置“1”。

MC_MoveAdditive 打断其它指令时，在上一个指令的目标位置上叠加 Distance 距离做为目标位置，继续运行。比如前一个指令的目标位置为 1000 units，而当前只运行到了 500 units 即被下一个块打断，如果这个块的 Distance 为 200 units，那么轴会运行到 1200 units 处。

当参数配置错误或指令执行错误时，Error 被置“1”，ErrorID 为错误码，根据错误码可查找到相应的错误，详见附录 E。

● 时序图

图 7.6.11 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当轴运行到目标位置上叠加了指定距离后的新目标位置时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

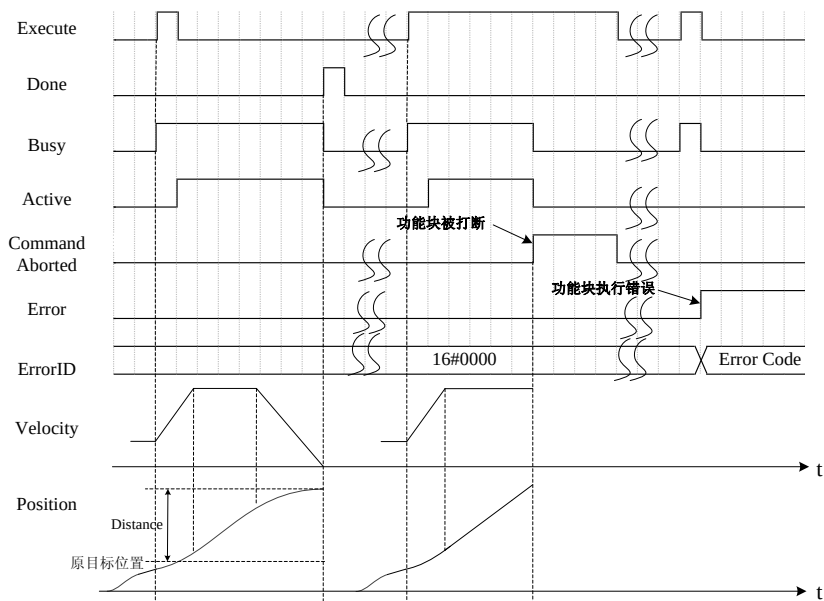


图 7.6.11 指令引脚时序和速度、位置变化图

● 应用示例

图 7.6.12 显示了在第一个 MC_Absolute 指令执行完成之前调用第二个指令的情况：第一个指令控制轴运动到目标位置 100000 units（同时速度减为 0）之前，使能 M0.1，MC_MoveAdditive 打断第一个指令，将目标脉冲数叠加 150000 units 到达 250000 units 的位置并以设定的速度运动，当轴运动到目标位置时速度减为 0，轴停止运动，Done 被置“1”。

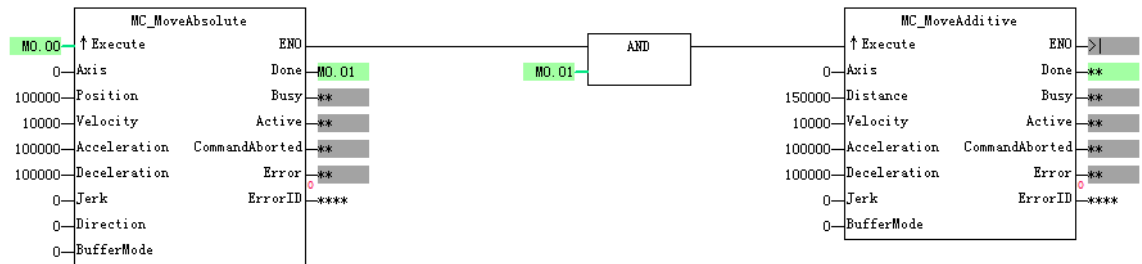


图 7.6.12 叠加定位运行

运动曲线如图 7.6.13 所示：

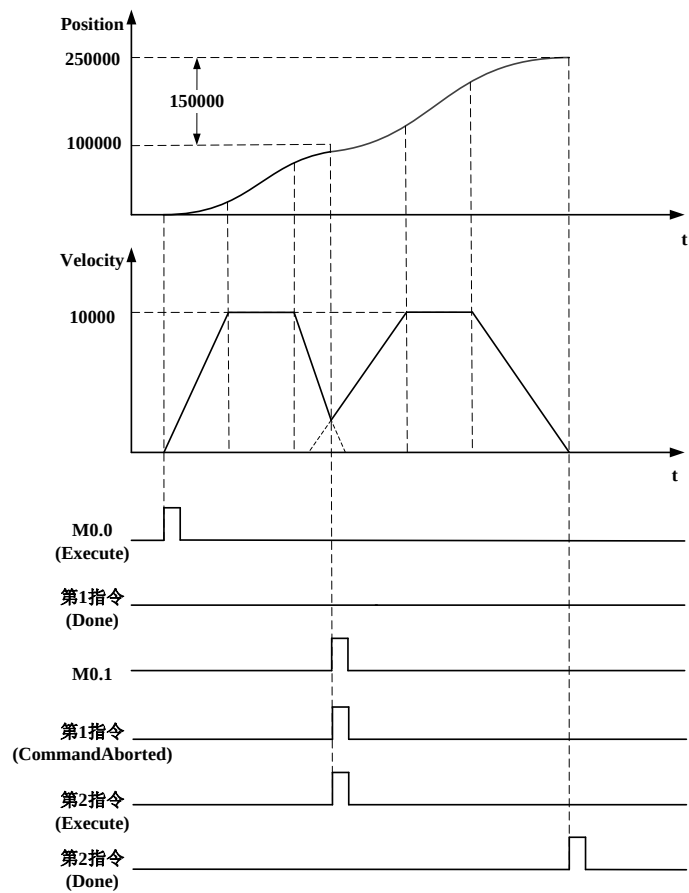


图 7.6.13 叠加定位运行曲线与时序图

7.6.4 重新定位 MC_MoveSuperImposed

● 指令参数

MC_MoveSuperImposed <<↑Execute双击配置曲线 ENO> 0AxisDone** 10000DistanceBusy** 10000VelocityDiffCommandAborted** 100000AccelerationError** 100000DecelerationErrorID**** 0Jerk 0BufferMode		在轴运动时，修改目标位置和目标速度； 指令使能后，轴的目标位置为当前实际位置加 Distance，轴的目标速度为当前实际速度加 VelocityDiff；速度以当前速度为起始速度，到达目标位置时速度为 0； 该指令支持“梯型曲线”和“S 型曲线”加减速控制方式。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID

Distance	双字变量、常数	REAL	正数、负数	10000	相对距离, 单位 units
VelocityDiff	双字变量、常数	REAL	正数	10000	运行速度偏移, 单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度; 若为 0, 则无加速过程, 单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度; 若为 0, 则无减速过程, 单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度; 若为 0, 速度曲线为梯形, 若非 0, 速度曲线为 S 形, 单位 units/s ³
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

通过 MC_MoveSuperImposed 指令, 可以改变轴的目标位置, 以当前实际位置加 Distance 引脚值的和为目标位置, 以当前速度加 VelocityDiff 引脚值的和为目标速度。指令正在运行时, 如果执行了控制相同轴的其他运动控制指令, 此时该指令运行会被打断, CommandAborted 被置“1”。

MC_MoveSuperImposed 打断其它指令时, 在当前位置上叠加 Distance 距离做为目标位置, 在当前的速度叠加 VelocityDiff 作为目标速度, 继续运行。

当参数配置错误或指令执行错误时, Error 被置“1”, ErrorID 为错误码, 根据错误码可查找到相应的错误, 详见附录 E。

● 时序图

图 7.6.14 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”; 当目标位置及速度改变完成时 Done 被置“1”; 如果有其它的指令打断此指令, CommandAborted 被置“1”, 同时 Busy 被置“0”; 如果在指令执行期间有错误发生, 则 Error 被置“1”, 同时轴将停止, 可以通过 ErrorID(错误码)的值找到导致错误的原因。

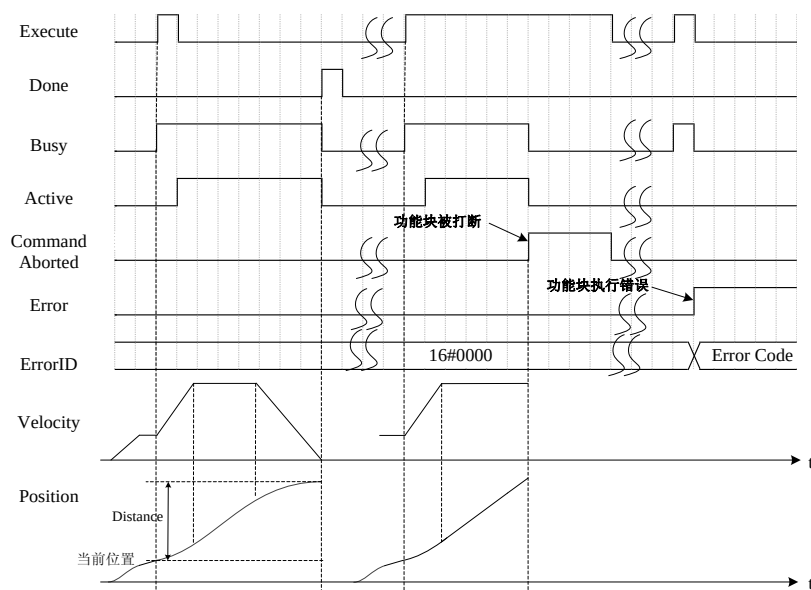


图 7.6.14 指令引脚时序和速度、位置变化图

● 应用示例

图 7.6.15 显示了在绝对定位运行过程中调用重新定位指令的情况：在绝对定位指令控制轴达到目标位置 100000 units（同时速度减为 0）之前，被重新定位指令打断，重新定位指令会修改目标位置以及目标速度，在轴的当前位置上增加 200000 units 的距离，将当前速度增加 10000 units，作为新的目标速度，然后运动到目标位置。

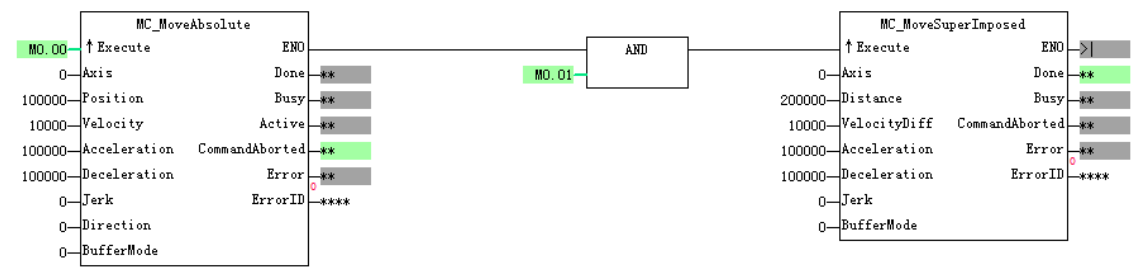


图 7.6.15 重新定位运行

运动曲线如图 7.6.16 所示：

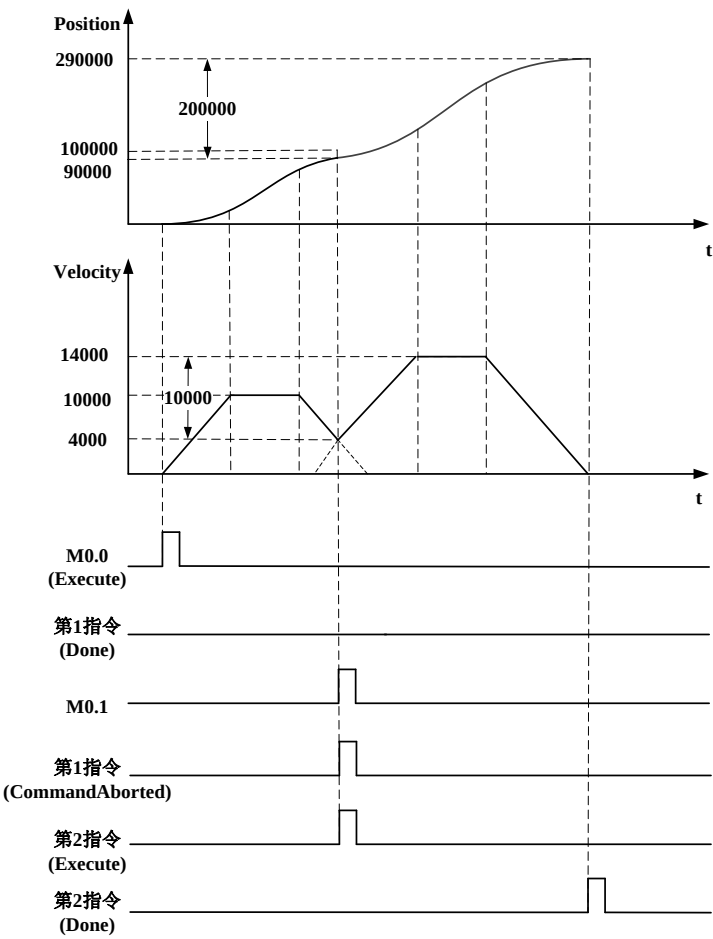


图 7.6.16 重新定位运行曲线与时序图

7.6.5 恒速运行 MC_MoveVelocity

● 指令参数

MC_MoveVelocity << ↑ Execute 双击配置曲线 ENO 0 Axis 10000 100000 100000 0 0 ?? 0 InVelocity Busy Active CommandAborted Error ErrorID ** ** ** ** ** ****					
控制轴以设定的速度恒速运动。 指令使能后，轴以当前速度为起始速度，按照设定的参数达到目标速度后恒速运动。 Direction: 为 0 正向运动，为 1 负向运动。 该指令支持“梯型曲线”和“S 型曲线”加减速控制方式。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
Velocity	双字变量、常数	REAL	正数	10000	运行目标速度,单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度; 若为 0, 则无加速过程, 单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度; 若为 0, 则无减速过程, 单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度; 若为 0, 速度曲线为梯形; 若非 0, 速度曲线为 S 形, 单位 units/s ³
Direction	字变量、常数	WORD	非负整数	0	0 为正方向, 1 为负方向
Continuous	位变量	BOOL	0 或 1	0	连续运行模式 0: 运动到正负软件限位后, 轴停止运动; 1: 运动到正负软件限位后, 轴当前位置被清零并继续运动
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
InVelocity	位变量	BOOL	0 或 1	为“1”时, 表示达到目标速度	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时, 表示轴正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

MC_MoveVelocity 指令使轴以恒定速度连续运行。如果当前速度小于目标速度, 则通过加速后恒速运行; 如果当前速度大于目标速度, 则通过减速后恒速运行。

指令正在运行时, 如果执行了控制相同轴的其他运动控制指令, 该指令运行会被打断, CommandAborted 被置“1”。

当参数配置错误或指令执行错误时, Error 被置“1”, ErrorID 为错误码, 根据错误码可查找到相应的错误, 详见附录 E。

● 时序图

图 7.6.17 为该指令的引脚时序和速度变化图。Busy 和 Execute 同时被置“1”, Active 在下一周期被置“1”; 当速度到达目标值时 InVelocity 被置“1”; 如果有其它的指令打断此指令, CommandAborted 被置“1”, 同时 Busy 和 Active 被置“0”; 如果在指令执行期间有错误发生, 则 Error 被置“1”, 同时轴将停止, 可以通过 ErrorID(错误码)的值找到导致错误的原因。

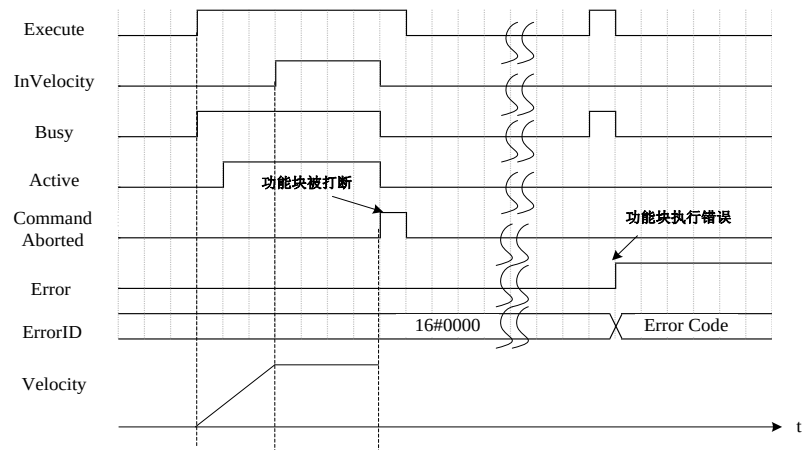


图 7.6.17 指令引脚时序和速度变化图

● 应用示例

图 7.6.18 中使能 M0.0 后，轴会以当前的方向、速度、加速度运动，当轴的当前速度达到目标速度的时候，InVelocity 被置“1”。

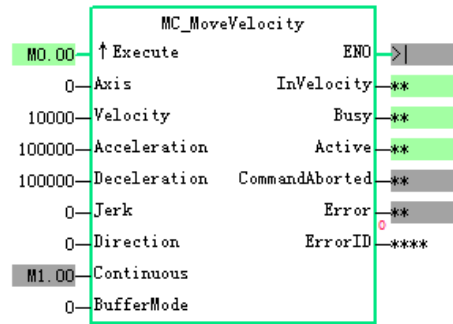


图 7.6.18 恒速运行

运动曲线如图 7.6.19 所示：

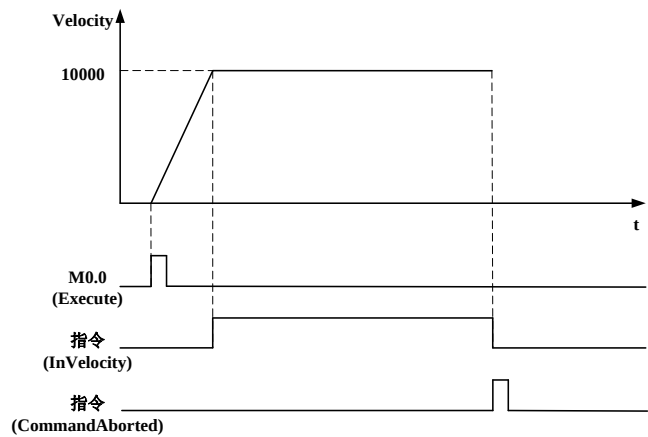
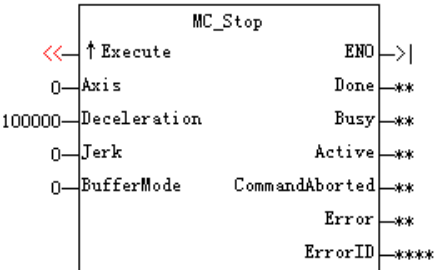


图 7.6.19 恒速运行曲线与时序图

7.6.6 停止轴 MC_Stop

● 指令参数

					
停止轴运动 减速度为 0 时，轴立即停止运动；减速度非 0 时，轴以当前速度为起始速度，减速停止运动。 该指令支持“梯型曲线”和“S 型曲线”加减速控制方式。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示轴已停止运动	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴正在减速停止中	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_Stop 指令，停止轴的运动。在当前速度不为零时，如果减速度为非零值（大于零），则轴减速停止；如果减速度值设为 0，轴立即停止。

参数配置错误或指令执行错误时，Error 被置“1”，ErrorID 为错误码，根据错误码可查找到相应的错误，详见附录 E。

● 时序图

图 7.6.20 为该指令的引脚时序图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当轴停止时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

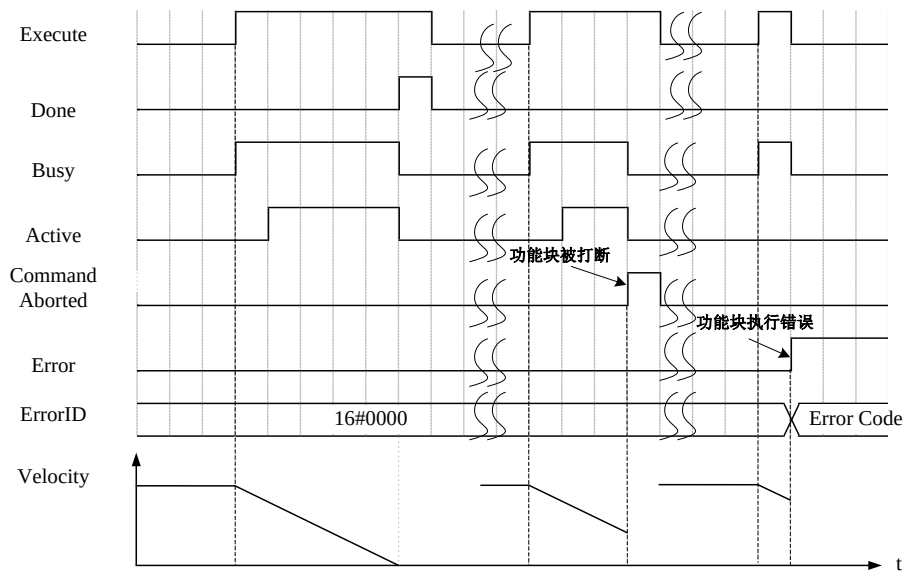


图 7.6.20 指令引脚时序和速度变化图

● 应用示例

图 7.6.21 中，当 M0.0 使能后，程序调用 MC_MoveVelocity 指令执行单轴的匀速运动。将 M0.1 置 “1” 后，程序调用 MC_Stop 指令，轴 0 减速停止。

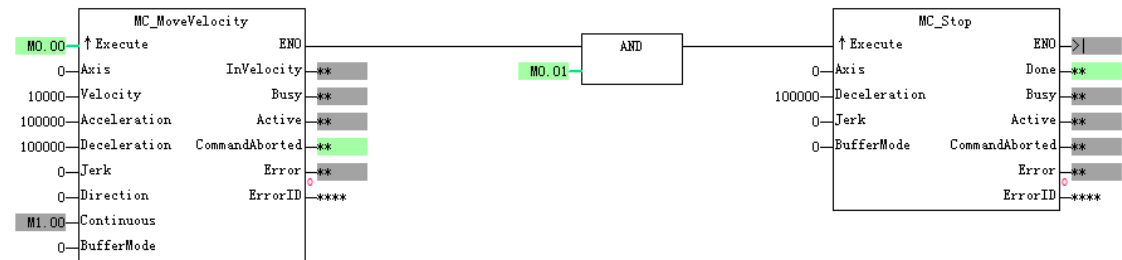


图 7.6.21 停止轴运行

运动曲线如图 7.6.22 所示：

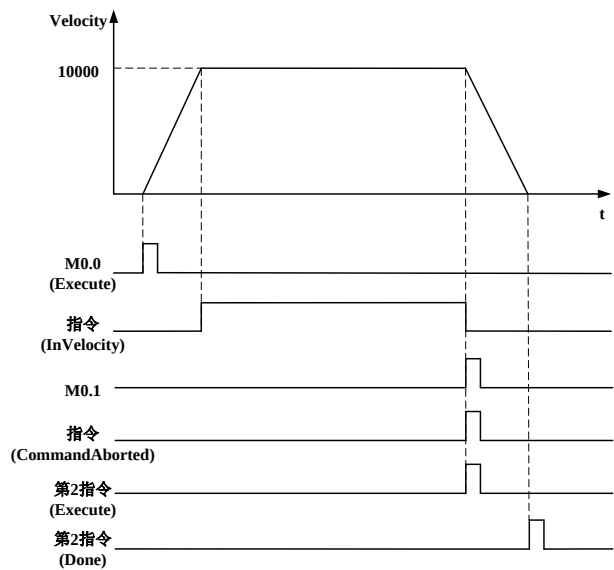
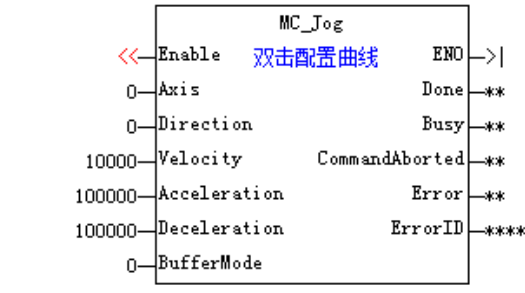


图 7.6.22 停止轴运行曲线与时序图

7.6.7 点动运行 MC_Jog

● 指令参数

		控制轴以设定的速度点动运行； Enable 为 “1” 时轴以当前速度为起始速度，达到目标速度后恒速运动。 Enable 为 “0” 时轴以当前速度为起始速度，减速停止运动。 Direction: 0 为正方向，1 为负方向。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
Direction	字变量、常数	WORD	非负整数	0	运动方向: 0 为正方向, 1 为负方向
Velocity	双字变量、常数	REAL	正数	10000	运行目标速度, 单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度; 若为 0, 则无加速过程, 单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度; 若为 0, 则无减速过程, 单位 units/s ²
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为 “1” 时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为 “1” 时, 表示指令正在执行	
CommandAborted	位变量	BOOL	0 或 1	为 “1” 时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为 “1” 时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

通过 MC_Jog 指令，控制轴点动运行。Enable 置 “1”，轴运动；Enable 置 “0”，轴停止运动。

指令正在运行时，如果执行了控制相同轴的其他运动控制指令，该指令运行会被终止，CommandAborted 引脚被置 “1”。参数配置错误或指令执行错误时，Error 被置 “1”，ErrorID 为错误码，根据错误码可查找到相应的错误，详见附录 E。

● 时序图

图 7.6.23 为该指令的引脚时序和速度变化图。当 Enable 置 “1” 时 Busy 被置 “1”，当 Direction 为 0 时，正向运行，当 Direction 为 1 时，反向运行；在运动过程中，若将 Enable 置 “0”，则轴开始减速运动，当运动停止时，Done 被置 “1”。如果有其它的指令打断此指令，CommandAborted 被置 “1”，同时 Busy 被置 “0”；如果在指令执行期间有错误发生，则 Error 被置 “1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

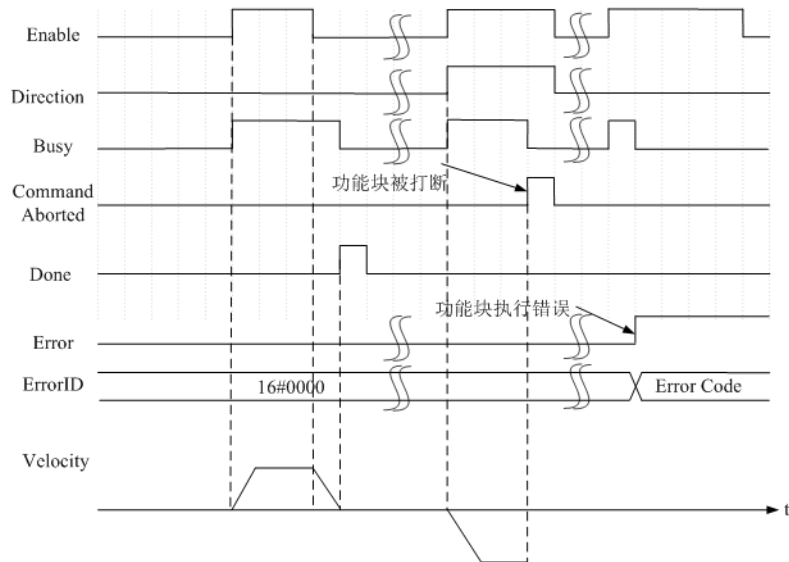


图 7.6.23 指令引脚时序和速度变化图

● 应用示例

图 7.6.24 中将 M0.0 置“1”，使能点动功能，M0.2 置“0”，正向点动，可以从 MC_ReadActualPosition 中看出轴的位置向正向运动。

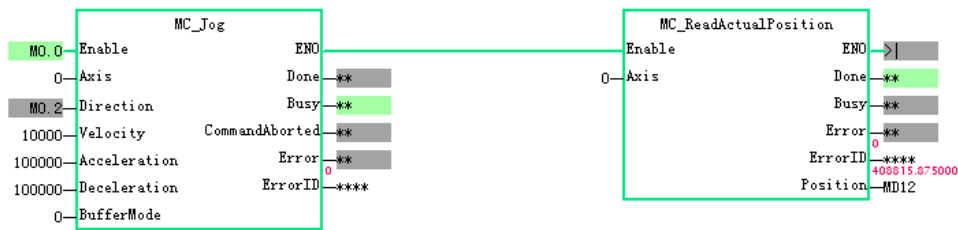


图 7.6.24 正向点动运行

运动曲线如图 7.6.25 所示：

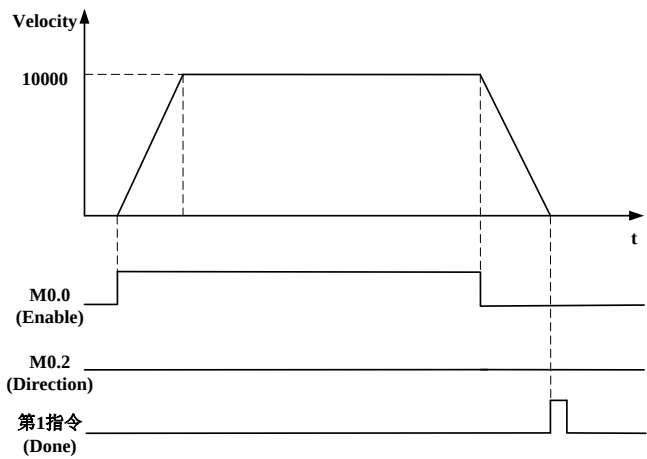


图 7.6.25 正向点动运行曲线与时序图

图 7.6.26 中将 M0.0 置“1”，使能点动功能，将 M0.2 置“1”，负向点动， MC_ReadActualPosition 指令可以读取轴的实时位置。

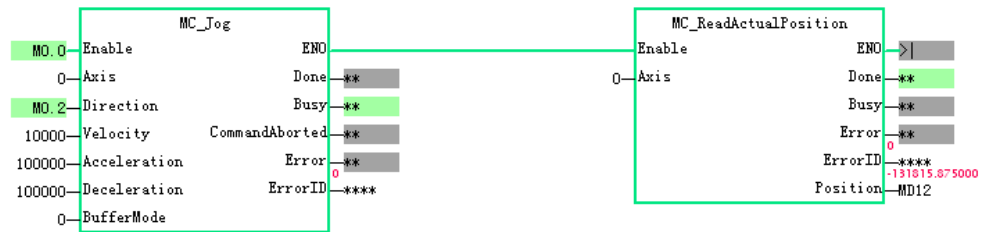


图 7.6.26 负向点动运行

运动曲线如图 7.6.27 所示。

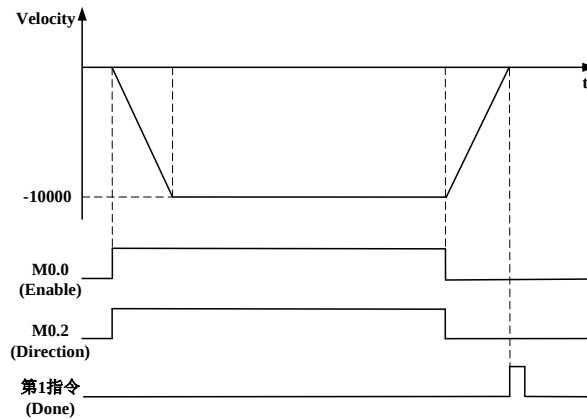


图 7.6.27 负向点动运行曲线与时序图

7.6.8 回原点 MC_Home

● 指令参数

		控制轴回原点。 指令使能后，轴按照设定的回原点速度与原点接近速度开始回原点；速度以当前速度为起始速度，到原点位置时速度为 0。 控制器共支持 9 种回原点模式，具体情况应根据机械结构中输入信号的个数、类型、排布进行合理选择。 回原点模式与参数请在运动控制参数配置界面配置并下载到控制器中。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_Home 指令，可以控制轴回原点。回原点现在支持 9 种模式，不同的模式对应着不同的回原点的方式。回原点模式及参数的配置可以在运动控制参数的单轴参数配置中进行。

表 7.6.1 表示各种回原点模式所对应的参数使用情况，其中“√”表示参数被使用，“__”表示参数未被使用。

表 7.6.1 回原点参数使用情况

参数 模式	信号				方向				运动参数			
	原点附近输入信号	原点输入信号	正方向限位输入信号	负方向限位输入信号	原点搜索方向	回原点开始方向	回原点速度	接近原点速度	回原点加速度	回原点加加速度	原点检测屏蔽距离	原点位置偏移量
模式 1: 接近信号 OFF(反转)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	—	✓
模式 2: 接近信号 ON(反转)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	—	✓
模式 3: 接近信号 OFF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	—	✓
模式 4: 接近信号 ON	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	—	✓
模式 5: 限位信号 OFF	—	✓	✓	✓	✓	✓	✓	✓	✓	✓	—	✓
模式 6: 原点信号屏蔽	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
模式 7: 只用限位信号	—	—	✓	✓	✓	✓	✓	✓	✓	✓	—	✓
模式 8: 只用原点信号	—	✓	—	—	✓	✓	—	✓	✓	✓	—	✓
模式 9: 无接近信号	—	✓	✓	✓	✓	✓	—	✓	✓	✓	—	✓

正（负）方向限位信号：在回原点的过程中，当限位信号由 OFF 变成 ON 时，轴反转继续回原点。

原点附近输入信号：在回原点的过程中，当原点附近输入信号由 OFF 变成 ON 时，轴开始减速。

原点输入信号：在回原点的过程中，回原点输入信号由 OFF 变成 ON 时，轴停止运动，记录原点位置。

回原点开始方向：回原点运动开始时，轴开始运动的方向。在模式 5 和模式 7 时，如果超出负限位，禁止将回原点开始方向设为负向；如果超出正限位，禁止将回原点开始方向设为正向。在模式 1，模式 2，模式 6 时，当原点附近输入信号为 ON 时，指令使能后，控制指定轴按照原点搜索方向的反方向和回原点速度开始运动，如图 7.6.28 所示：

原点搜索方向：正方向

回原点开始方向：正方向

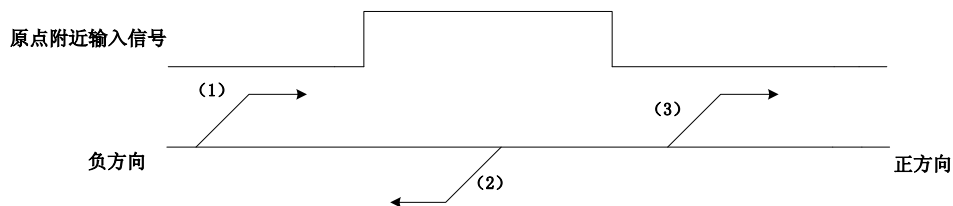


图 7.6.28 回原点开始方向

(1) (3) 原点附近输入信号为 OFF，所以轴开始运动方向为回原点开始方向；

(2) 原点附近输入信号为 ON，所以轴开始运动方向为原点搜索方向的反方向。

原点搜索方向：回原点输入被搜索到时的方向。

图 7.6.29 为回原点搜索方向为正向的示意图。

原点搜索方向：正方向

回原点开始方向：正方向

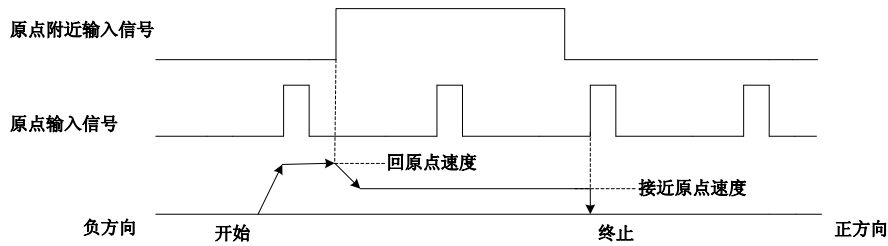


图 7.6.29 回原点搜索方向：正向

图 7.6.30 为回原点搜索方向为负向的示意图。

原点搜索方向：负方向

回原点开始方向：正方向

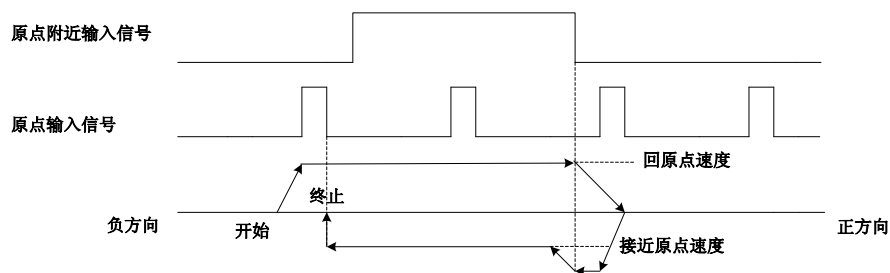


图 7.6.30 回原点搜索方向：负向

回原点动作模式 1：接近信号 OFF（反转）

图 7.6.31 为回原点模式 1 的示意图，原点搜索方向为正向，其中 a、b、c 为回原点方向为正方向的情况；d、e、f 为回原点方向为负方向的情况。当原点搜索方向为负向时，同理。

原点附近输入信号必须为 OFF，原点输入信号才有效。如果指令使能前检测到原点附近输入信号为 ON 时，当指令使能后，控制指定轴按照原点搜索方向的反方向和回原点速度开始运动。如 b 图开始回原点时，因有原点附近输入信号，故回原点方向变为负向。负方向时，遇到原点附近输入信号下降沿，改变速度方向，不改变速度大小；正方向时，遇到原点附近输入信号上升沿，变为接近原点速度。

原点搜索方向：正方向

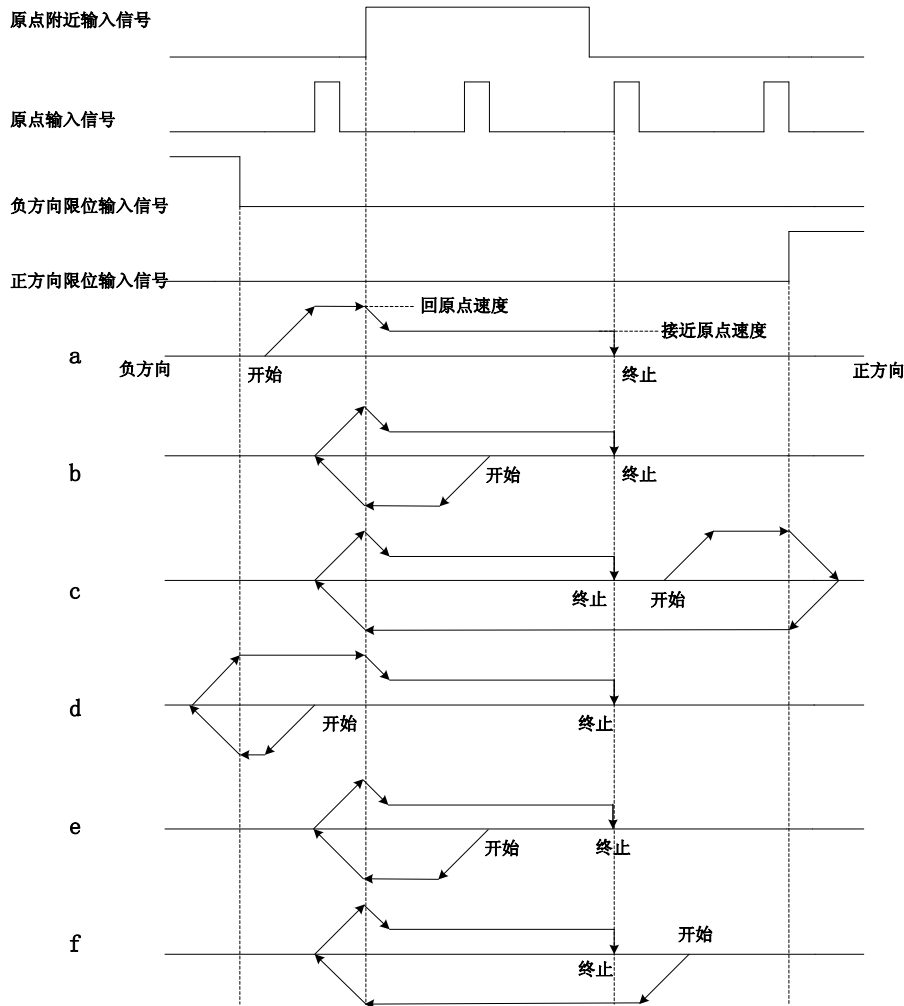


图 7.6.31 模式一的示意图

回原点动作模式 2：接近信号 ON（反转）

0 为回原点模式 2 的示意图，原点搜索方向为正向，其中 a、b、c 为回原点方向为正方向的情况；d、e、f 为回原点方向为负方向的情况。当原点搜索方向为负向时，同理。

原点附近输入信号必须为 ON，原点输入信号才有效。如果指令使能前检测到原点附近输入信号为 ON 时，当指令使能后，控制指定轴按照原点搜索方向的反方向和回原点速度开始运动。如 b 图开始回原点时，因有原点附近输入信号，故回原点方向变为负向。负方向时，遇到原点附近输入信号下降沿，改变速度方向，不改变速度大小；正方向时，遇到原点附近输入信号上升沿，变为接近原点速度。

原点搜索方向：正方向

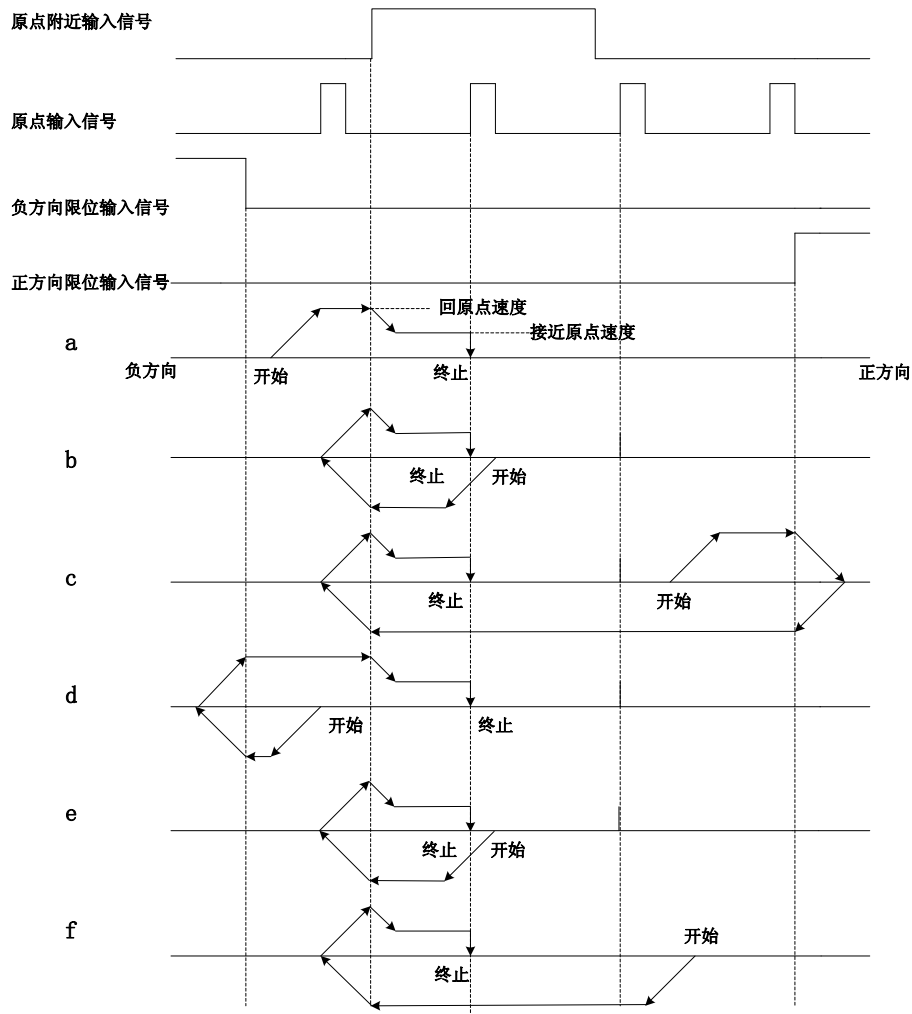


图 7.6.32 模式二的示意图

回原点动作模式 3：接近信号 OFF

图 7.6.33 为回原点模式 3 的示意图，原点搜索方向为正向，其中 a、b、c 为回原点方向为正方向的情况；d、e、f 为回原点方向为负方向的情况。与模式 1 的不同之处在于，负方向运动时，遇到原点附近输入信号的下降沿时，不改变方向。当原点搜索方向为负向时，同理。

原点搜索方向：正方向

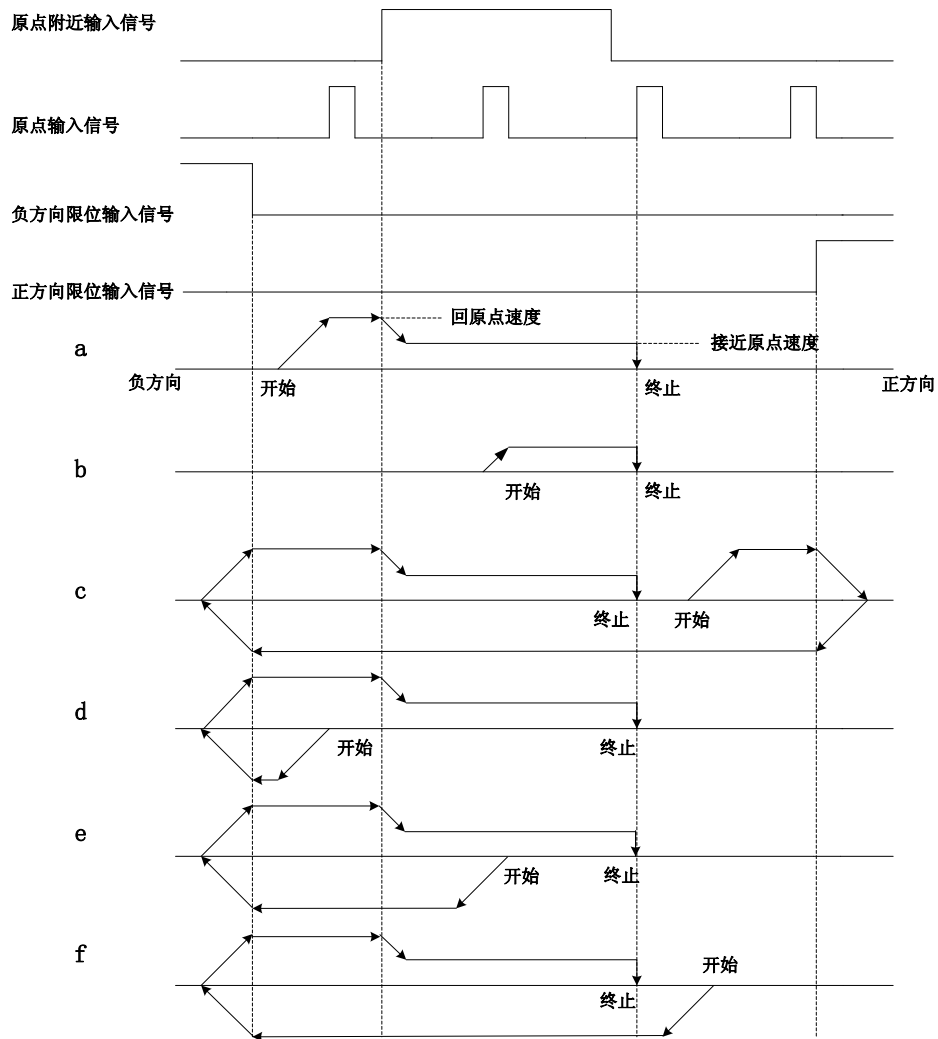


图 7.6.33 模式三的示意图

回原点动作模式 4：接近信号 ON

图 7.6.34 为回原点模式 4 的示意图，原点搜索方向为正向，其中 a、b、c 为回原点方向为正方向的情况；d、e、f 为回原点方向为负方向的情况。与模式 2 的不同之处在于，负方向运动时，遇到原点附近输入信号的下降沿时，不改变方向。当原点搜索方向为负向时，同理。

原点搜索方向：正方向

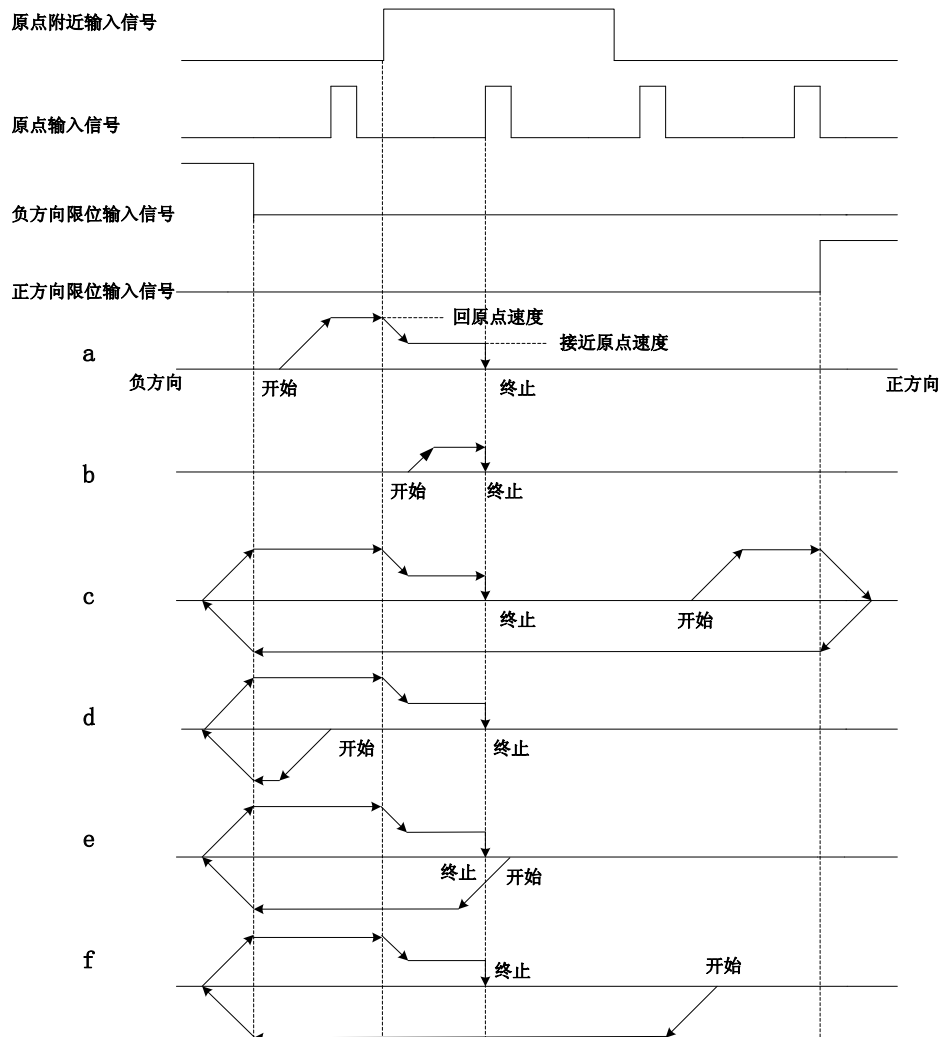


图 7.6.34 模式四的示意图

回原点动作模式 5：限位信号 OFF

图 7.6.35 为回原点模式 5 的示意图，原点搜索方向为正向，其中 a、b、c 为回原点方向为正向的情况；d、e 为回原点方向为负方向的情况。负方向限位输入信号为 ON 时，原点输入信号无效。负方向时，遇负方向限位输入信号下降沿，改变方向，速度大小不变；正向时，遇负方向限位输入信号上升沿，变为接近原点速度。当原点搜索方向为负向时，同理。

超过负限位时，禁止回原点开始方向设为负方向；

超过正限位时，禁止回原点开始方向设为正方向。

原点搜索方向：正方向

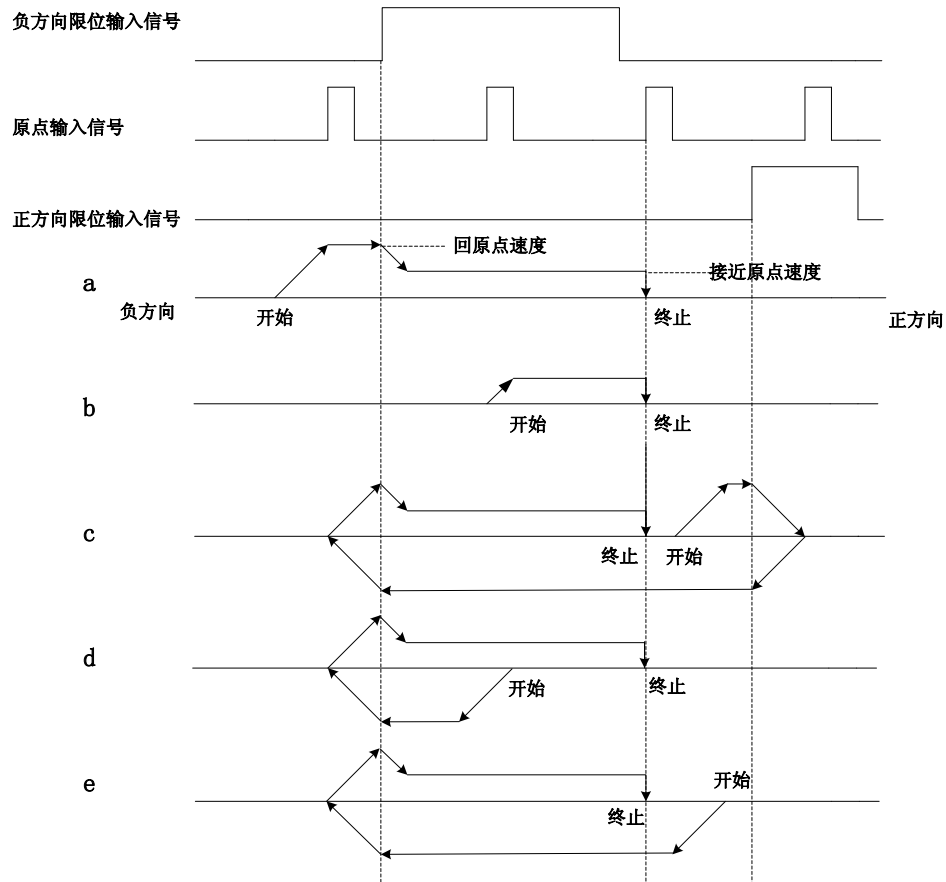


图 7.6.35 模式五的示意图

回原点动作模式 6：原点信号屏蔽

图 7.6.36 为回原点模式 6 的示意图，原点搜索方向为正向，其中 a、b、c 为回原点方向为正方向的情况；d、e、f 为回原点方向为负方向的情况。当原点搜索方向为负向时，同理。

如果指令使能前检测到原点附近输入信号为 ON 时，当指令使能后，控制指定轴按照原点搜索方向的反方向和回原点速度开始运动。屏蔽范围内，原点输入信号无法响应。

原点搜索方向：正方向

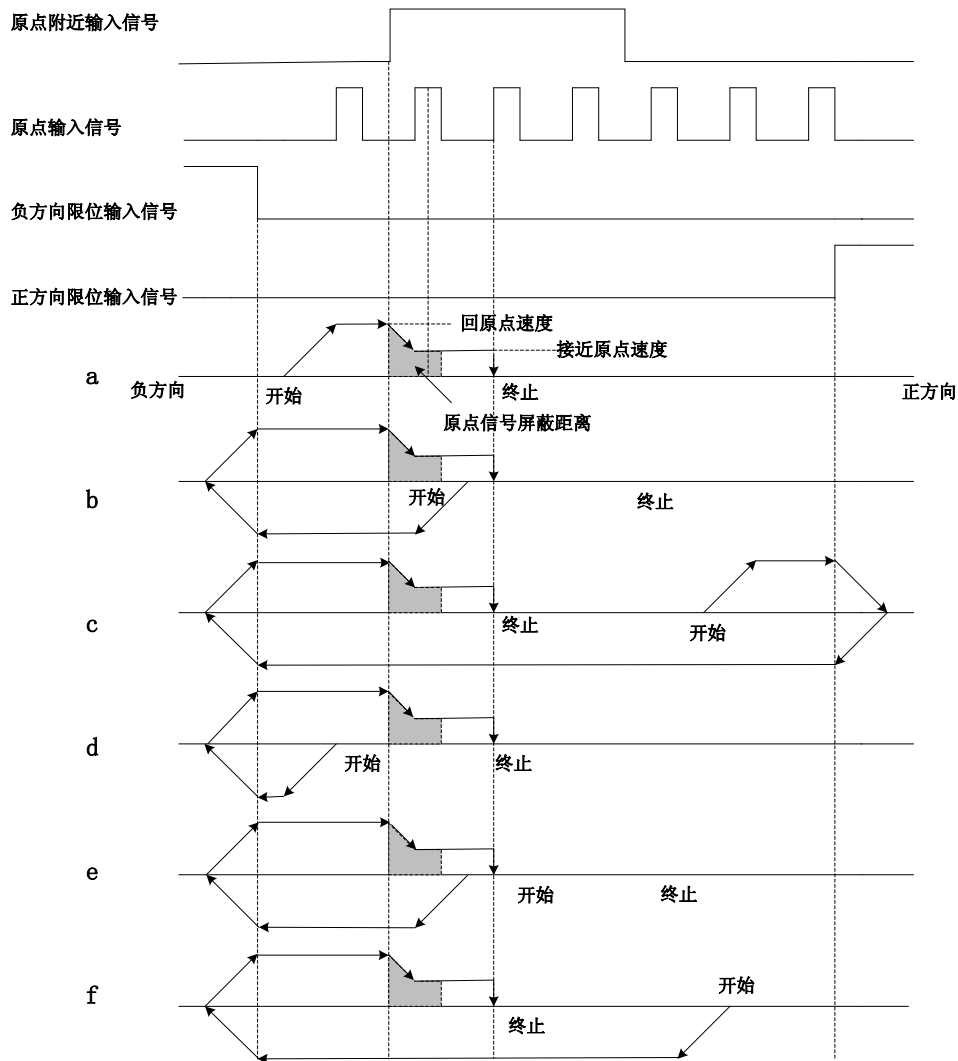


图 7.6.36 模式六的示意图

回原点动作模式 7：只用限位信号

图 7.6.37 为回原点模式 7 的示意图，原点搜索方向为正向，其中 a、b、c 为回原点方向为正方向的情况；d 为回原点方向为负方向的情况。正向时，遇负向限位输入信号下降沿和正向输入信号上升沿，改变方向。负向时，遇负向限位输入信号上升沿，改变方向同时速度大小变为接近原点速度，再遇负向限位输入信号下降沿，终止。当原点搜索方向为负向时，同理。

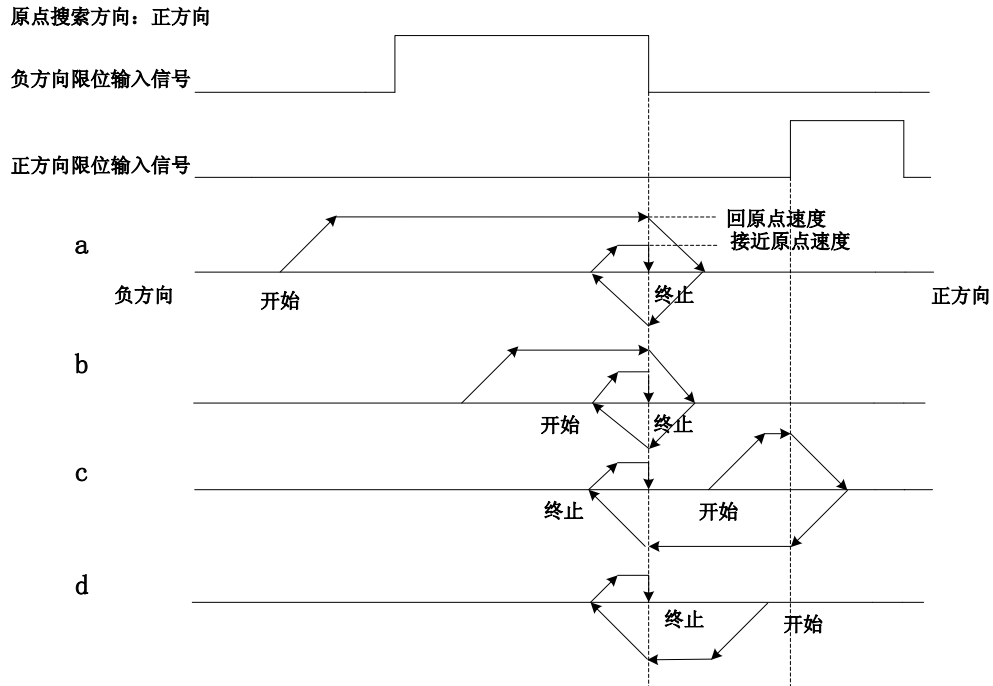


图 7.6.37 模式七的示意图

回原点动作模式 8：只用原点信号

图 7.6.38 为回原点模式 8 的示意图，原点搜索方向为正方向，其中 a、b、c 为回原点方向为正方向的情况，其中 b、c 两种情况由于循环机制，会在原点输入信号的上升沿终止；d、e、f 为回原点方向为负方向的情况，其中 d 情况由于循环机制，会在原点输入信号的上升沿终止。当原点搜索方向为负向时，同理。

原点搜索方向：正方向

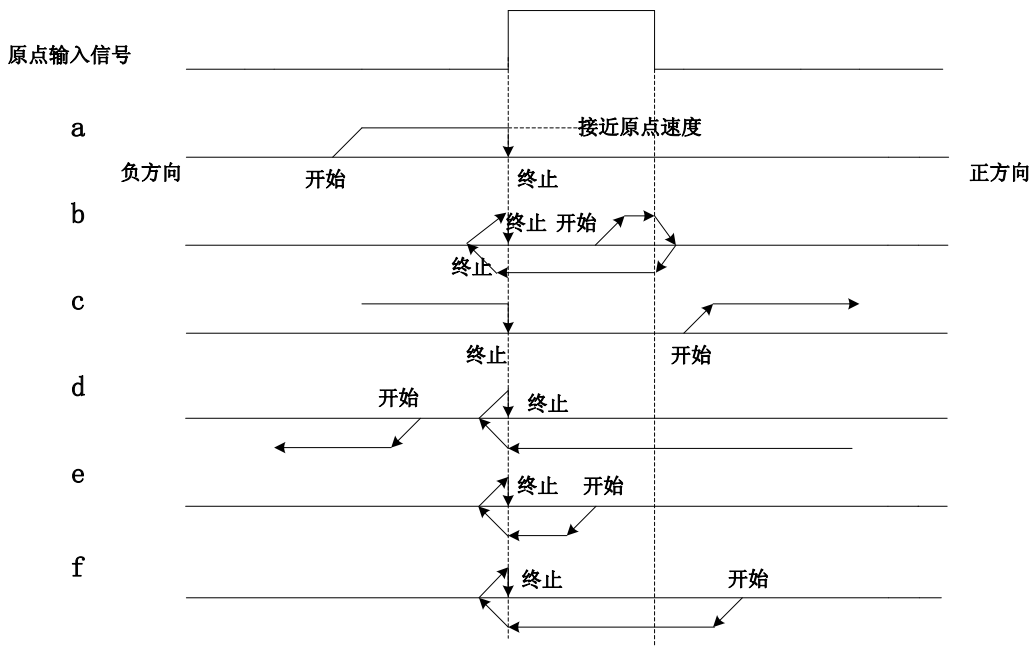


图 7.6.38 模式八的示意图

回原点动作模式 9：无接近信号

图 7.6.39 为回原点模式 9 的示意图，原点搜索方向为正方向，其中 a、b、c 为回原点方向为正方向的情况；d、e、f 为回原点方向为负方向的情况。无原点接近信号，回原点过程中，速度一直为接近原点速度。当原点搜索方向为负向时，同理。

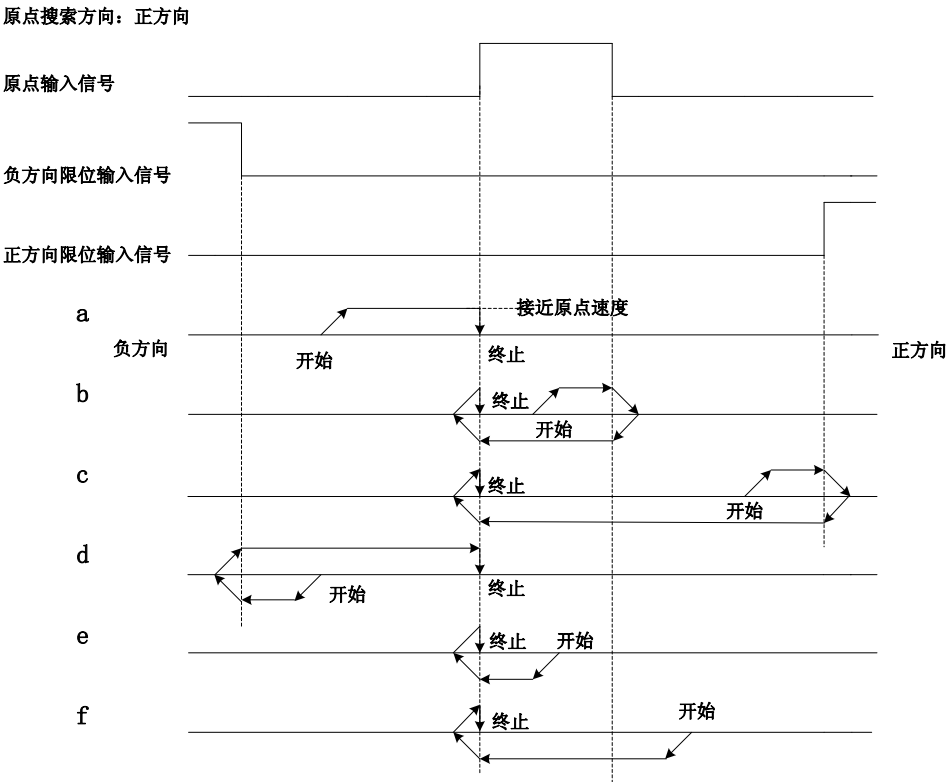


图 7.6.39 模式九的示意图

时序图

图 7.6.40 为该指令的引脚时序图。Busy 和 Execute 同时被置“1”；当轴回到原点时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

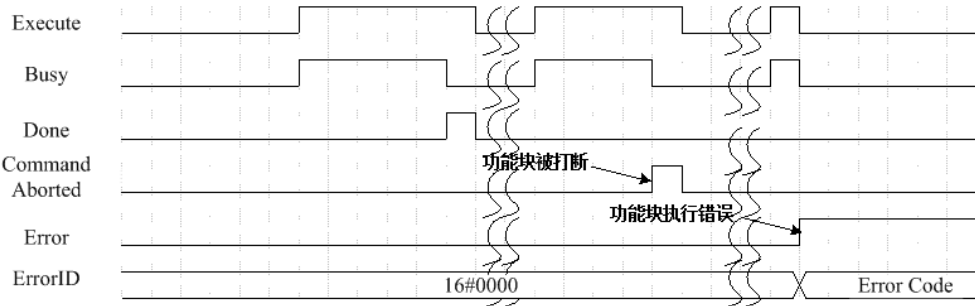


图 7.6.40 指令引脚时序图

编程举例

无

7.6.9 扩展回原点 MC_HomeEx

● 指令参数

MC_HomeEx		ENO	>
<< ↑ Execute	双击配置曲线	Done	**
0 Axis		Busy	**
10000 Position		CommandAborted	**
10000 Velocity		Error	**
100000 Acceleration		ErrorID	****
100000 Deceleration			
0 Jerk			

控制轴运动到离原点指定偏移量的绝对位置处。该指令，应在确认原点后执行。

指令使能后，轴从当前位置运动到距离原点为 Position 值的绝对位置；速度以当前速度为起始速度，到达目标位置时速度为 0。

该指令支持“梯型曲线”和“S 型曲线”加减速控制方式。

输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
Position	双字变量、常数	REAL	实数	10000	距离原点的偏移量, 单位 units
Velocity	双字变量、常数	REAL	正数	10000	运行目标速度, 单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度; 若为 0, 则无加速过程, 单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度; 若为 0, 则无减速过程, 单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度; 若为 0, 速度曲线为梯形, 若非 0, 速度曲线为 S 形, 单位 units/s ³
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为 “1” 时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为 “1” 时, 表示指令正在执行	
CommandAborted	位变量	BOOL	0 或 1	为 “1” 时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为 “1” 时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

通过 MC_HomeEx 指令, 可以控制轴回原点。回归到距离原点 Position 的偏移量的位置。指令正在运行时, 如果执行了控制相同轴的其他运动控制指令, 此时该指令运行会被打断, CommandAborted 被置 “1”。

● 时序图

图 7.6.41 为该指令的引脚时序图。Busy 和 Execute 同时被置 “1”; 当轴回到偏移原点 Position 的位置时 Done 被置 “1”; 如果有其它的指令打断此指令, CommandAborted 被置 “1”, 同时 Busy 被置 “0”; 如果在指令执行期间有错误发生, 则 Error 被置 “1”, 同时轴将停止, 可以通过 ErrorID(错误码)的值找到导致错误的原因。

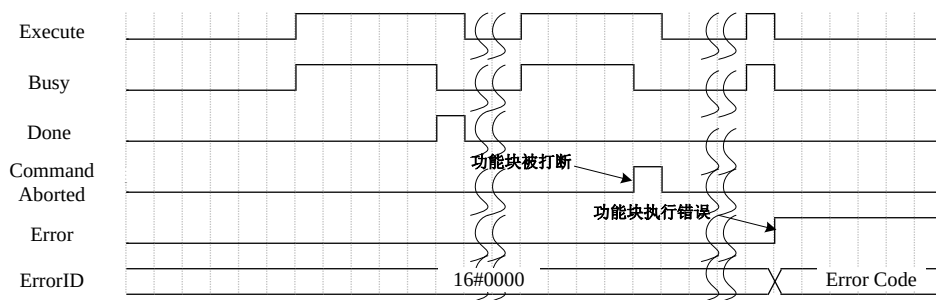


图 7.6.41 指令引脚时序图

● 应用示例

图 7.6.42 中将 M0.0 置为 “1”, 程序调用 MC_Absolute 指令运行到目标位置 (100000) 后停止。

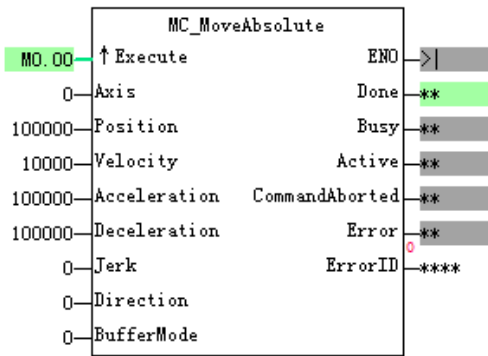


图 7.6.42 绝对定位运行

图 7.6.43 中将 M0.1 置为“1”，程序调用 MC_HomeEX 指令开始回原点，MC_HomeEX 指令的 Position 的引脚为 35000，轴则会运动到偏离原点 35000 的位置停止。

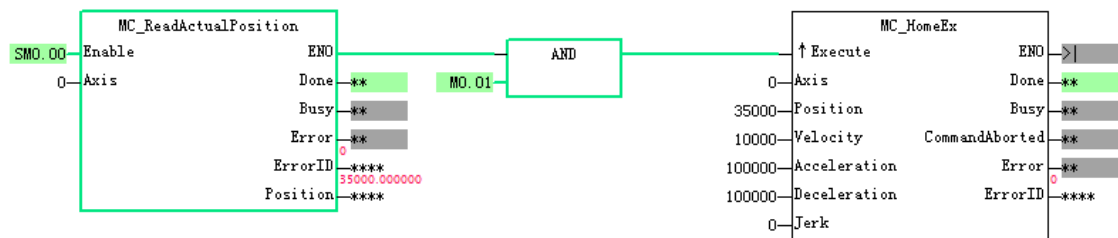


图 7.6.43 回原点运行

运动曲线如图 7.6.44 所示：

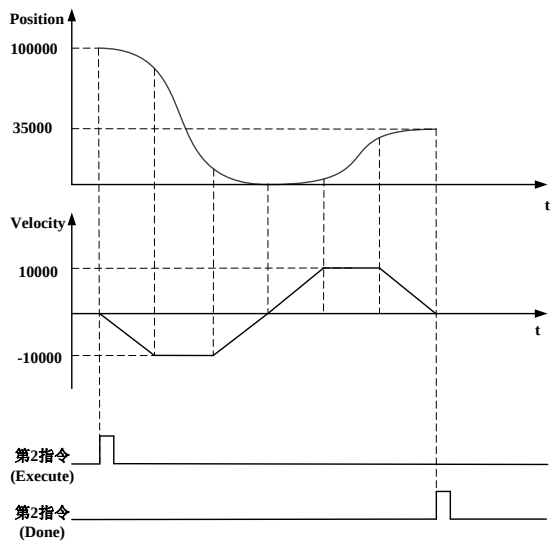


图 7.6.44 扩展回原点运行曲线与时序图

7.6.10 往复运行 MC_Reciprocator

- 指令参数

CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）

注 1：沿触发时，指令执行仅往复一次；持续高电平时，指令执行往复运动。

- 详细说明

往复运动指令 MC_Reciprocator，控制单轴按照给定的运动参数由起始位置和目标位置往复运动。指令正在运行时，如果执行了控制相同轴的其他运动控制指令，此时该指令运行会被打断，CommandAborted 被置“1”。PositionArray 输入为相对位置。

- 时序图

图 7.6.45 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当位移达到目标值而且定位完成时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

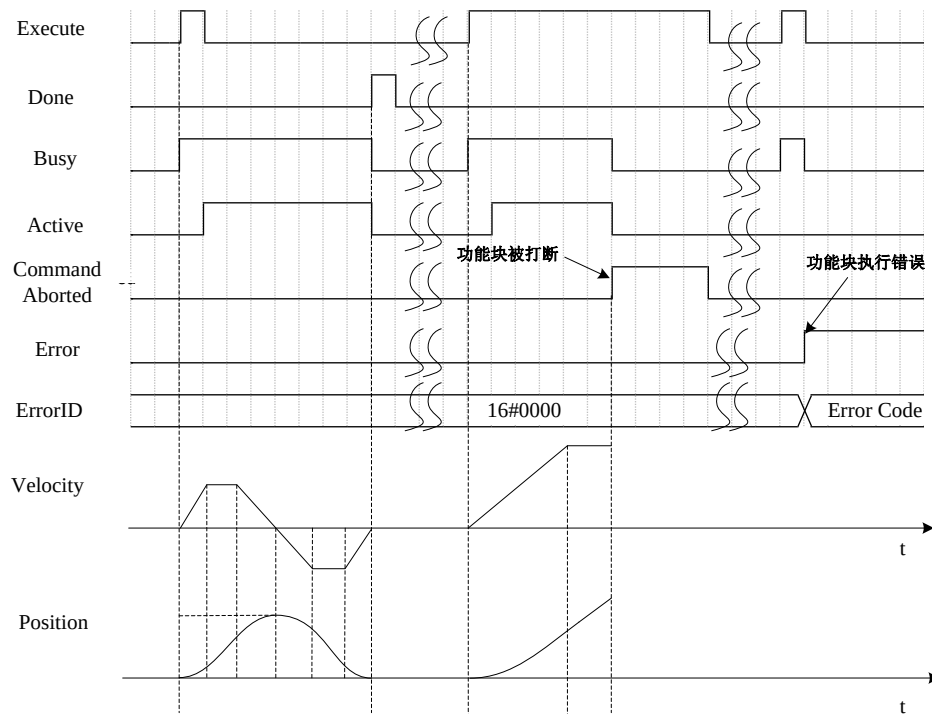


图 7.6.45 指令引脚时序和速度、位置变化图

- 应用示例

图 7.6.46 中将 M0.0 置为“1”，程序调用 MC_Reciprocator 指令，轴会按照图 7.6.47 指定的运动参数开始运动，设定的运动次数为 2 次，正向位置为 10000 脉冲，负向位置为-10000 脉冲，最后轴停止在 0 的位置。

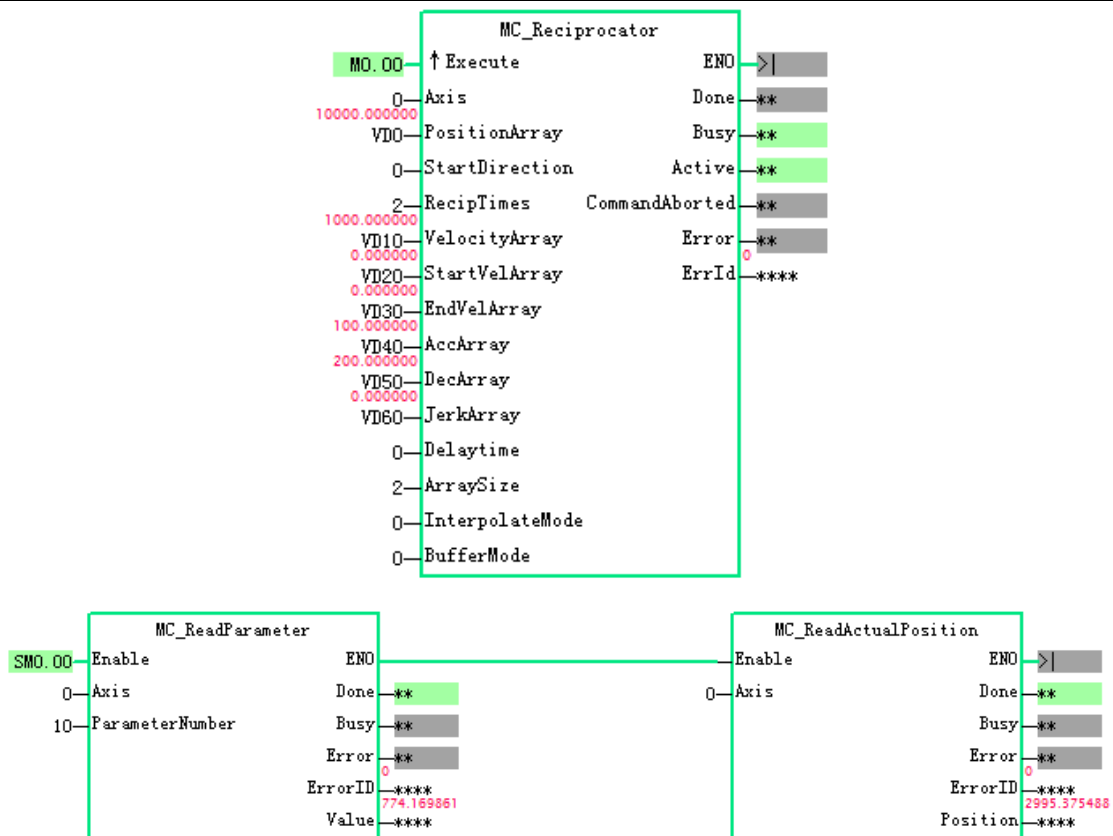


图 7.6.46 往复运行

绝对地	Modbus寄存器	数据类型	当前值(10进	当前值(16进制)	新数值	参数说明
VD0	0x0920(2336)	四字节浮点数	10000.000000		10000	第一段位置
VD2	0x0922(2338)	四字节浮点数	-10000.000000		-10000	第二段位置
VD10	0x092A(2346)	四字节浮点数	1000.000000		1000	第一段速度
VD12	0x092C(2348)	四字节浮点数	1000.000000		1000	第二段速度
VD20	0x0934(2356)	四字节浮点数	0.000000		0	第一段起始速度
VD22	0x0936(2358)	四字节浮点数	0.000000		0	第二段起始速度
VD30	0x093E(2366)	四字节浮点数	0.000000		0	第一段结束速度
VD32	0x0940(2368)	四字节浮点数	0.000000		0	第二段结束速度
VD40	0x0948(2376)	四字节浮点数	100.000000		100	第一段加速度
VD42	0x094A(2378)	四字节浮点数	200.000000		200	第二段加速度
VD50	0x0952(2386)	四字节浮点数	200.000000		200	第一段减速度
VD52	0x0954(2388)	四字节浮点数	100.000000		100	第二段减速度
VD60	0x095C(2396)	四字节浮点数	0.000000		0	第一段加加(减减)速度
VD62	0x095E(2398)	四字节浮点数	0.000000		0	第二段加加(减减)速度

图 7.6.47 往复运行参数

运动曲线如图 7.6.48 所示：

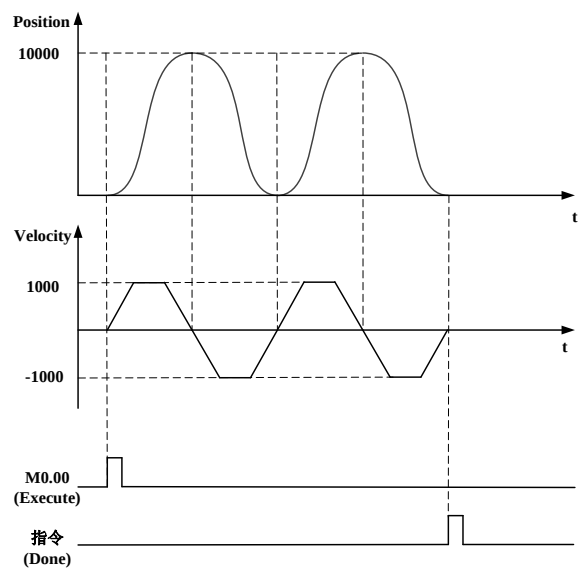


图 7.6.48 往复运行曲线与时序图

7.6.11 复位轴 MC_Reset

● 指令参数

MC_Reset <<Enable Axis ENO-> Done-** Busy-** Failure-** Error-** ErrorID-****			复位轴运行过程产生的错误。 轴运行出错后，调用此指令可以清除错误信息。 当 Error 为“1”时，说明复位失败，此时需重启系统。		
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示复位完成	
Busy	位变量	BOOL	0 或 1	保留	
Failure	位变量	BOOL	0 或 1	保留	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

当轴在运行过程中出错时，通过 MC_Reset 指令，将清除轴的错误状态，复位出错轴的指令 Error 及 ErrorID 引脚信息。

● 时序图

图 7.6.49 为该指令的引脚时序图。当 Enable 置“1”，复位完成时，Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

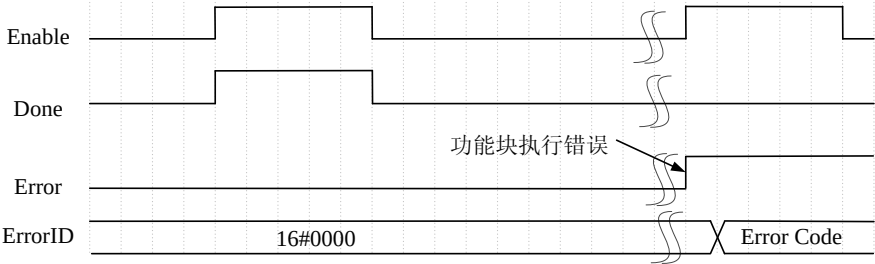


图 7.6.49 指令引脚时序图

● 应用示例

图 7.6.50 将软件限位修改为 90000，图 7.6.51 将 M0.1 置为 “1”，恒速输出至软件限位报错，然后将 M0.4 置为 “1”。

软件限位功能	无效
正向位置软件限制	90000.000000
负向位置软件限制	-4000000000.000000

图 7.6.50 修改正向位置软件限位

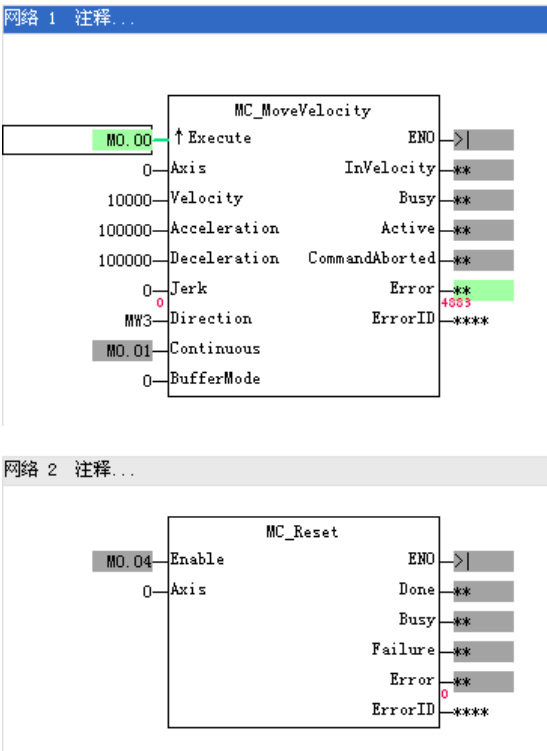


图 7.6.51 恒速输出至软件限位及轴复位

图 7.6.52 将 M0.4 置为 “1”，复位轴 0，将 MW3 置为 1，重新使能 M0.0 输出。

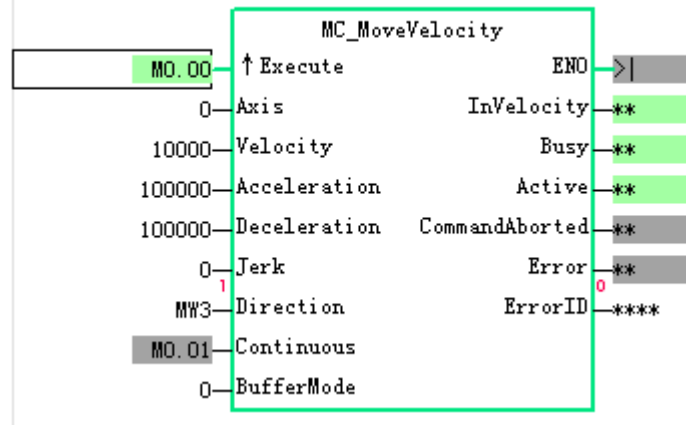


图 7.6.52 反向输出

7.6.12 读轴状态 MC_ReadStatus

● 指令参数

MC_ReadStatus << Enable 0 Axis ENO Done Busy Error ErrorID Errorstop Stopping StandStill DiscreteMotion ContinuousMotion SynchronizedMotion Homing ConstantVelocity Accelerating Decelerating > ** ** ** **** ** ** ** ** ** ** ** ** ** **					
读取轴当前状态 状态包括：错误停止状态、停止中状态、停止状态、离散运动状态、连续运动状态、同步运动状态、回原点状态、加速状态、减速状态、恒速状态。 当 Done 为“1”时，表示已经获取轴状态信息，并显示在对应的输出引脚上。					
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	保留	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	
Errorstop	位变量	BOOL	0 或 1	为“1”时，表示轴产生错误	
Stopping	位变量	BOOL	0 或 1	为“1”时，表示轴处于停止过程中	
StandStill	位变量	BOOL	0 或 1	为“1”时，表示轴停止	
DiscreteMotion	位变量	BOOL	0 或 1	为“1”时，表示轴离散运动	
ContinuousMotion	位变量	BOOL	0 或 1	为“1”时，表示轴连续运动	
SynchronizedMotion	位变量	BOOL	0 或 1	为“1”时，表示轴同步运动	
Homing	位变量	BOOL	0 或 1	为“1”时，表示轴回原点中	
ConstantVelocity	位变量	BOOL	0 或 1	为“1”时，表示轴以恒定速度运动	
Accelerating	位变量	BOOL	0 或 1	为“1”时，表示轴在加速运动	
Decelerating	位变量	BOOL	0 或 1	为“1”时，表示轴在减速运动	

● 详细说明

MC_ReadStatus 指令指示了轴当前的运行状态。通过调用 MC_ReadStatus 指令，读取当前正在运动的轴的详细状况，如轴停止、离散运行、连续运行、同步运行、回原点、加速、恒速、减速等状态。

● 时序图

图 7.6.53 为该指令的引脚时序图。Enable 置“1”时，当读取到轴运动状态正确时 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

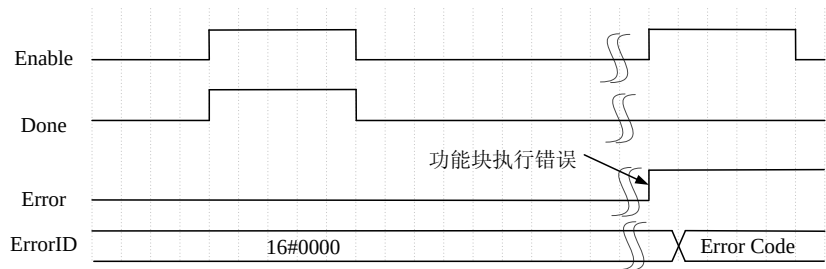


图 7.6.53 指令引脚时序图

● 应用示例

图 7.6.54 中使能 M0.0，执行 MC_MoveVelocity 指令和 MC_ReadStatus 指令，轴处于加速的连续运动，Accerelating 和 ContinuousMotion 被置“1”。

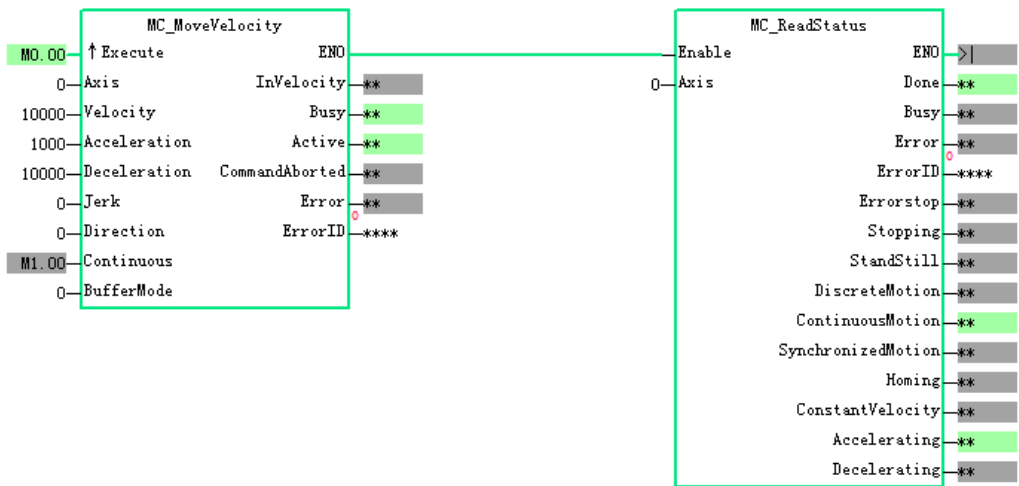


图 7.6.54 加速运动

图 7.6.55 中当轴加速到目标速度后，轴将进入匀速连续运动过程，ConstantVelocity 和 ContinuousMotion 均被置“1”。

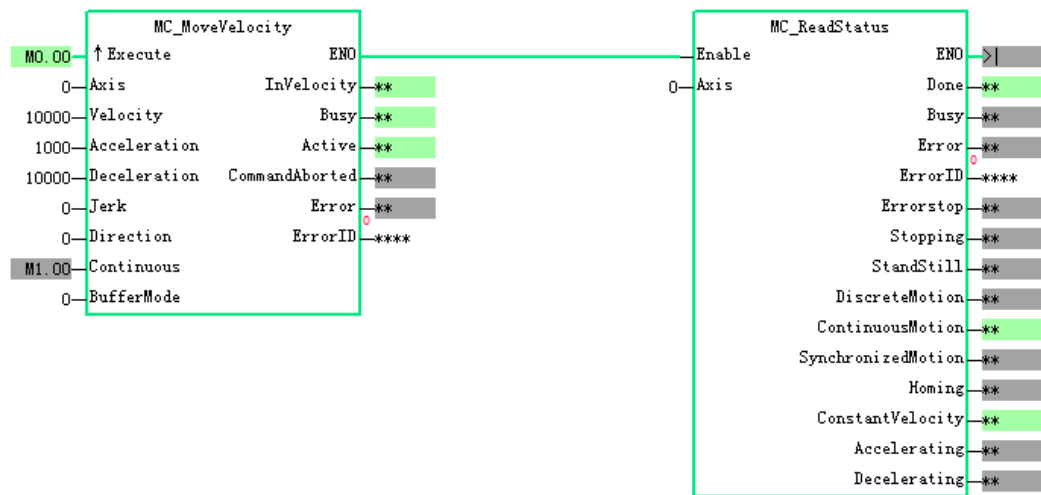


图 7.6.55 匀速运动

图 7.6.56 中使能 M0.1，程序调用 MC_Stop 指令，轴处于减速运动并将停止，Decerelating 和 Stopping 均被置“1”。当轴停止的时候 StandStill 被置“1”。

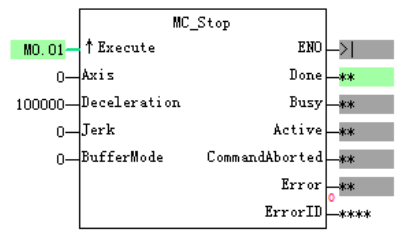
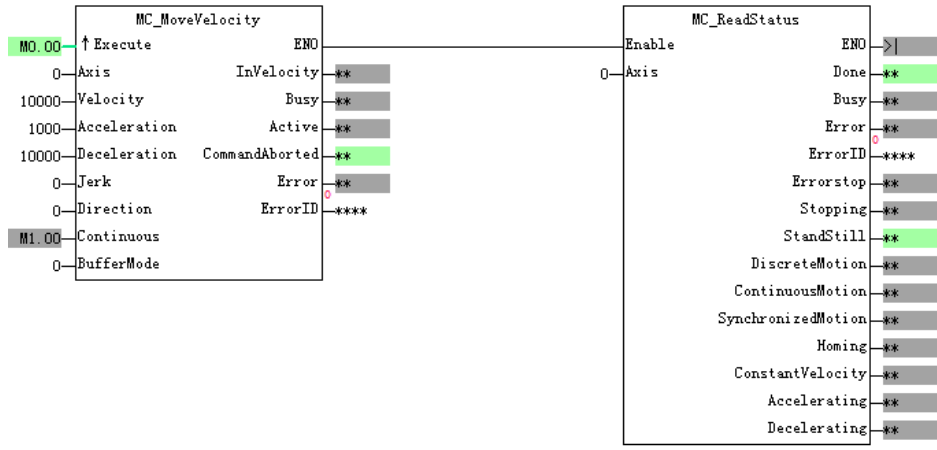


图 7.6.56 减速运动

7.6.13 读轴故障 MC_ReadAxisError

● 指令参数

MC_ReadAxisError <<Enable 0Axis ENO-> Done** Error** ErrorID**** 读取轴的故障信息 指令使能后，读取轴的错误信息，轴无错误时 ErrorID 为零，轴出错时 ErrorID 不为零，此时，可通过 ErrorID 查看异常处理表（附录 E），找出具体的错误信息。					
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发

Axis	字变量、常数	WORD	非负整数	0	轴 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	0 或 1	出错时的错误码（错误码见附录 E）	

- 详细说明

通过 MC_ReadAxisError 指令，读取轴上与指令无关的故障信息，例如没有将轴进行初始化就开始对轴进行操作。

- 时序图

图 7.6.57 为该指令的引脚时序图。当读取到轴上与指令无关的故障信息时 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

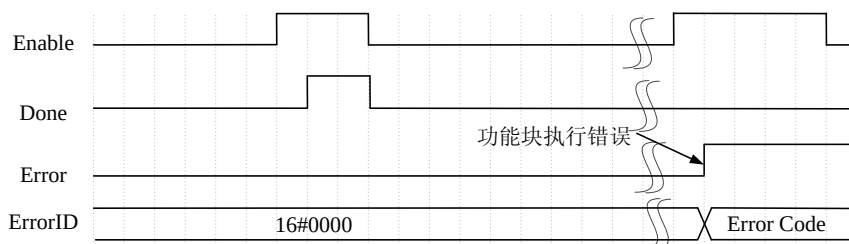


图 7.6.57 指令引脚时序图

- 应用示例

首先在“运动控制参数配置”的单轴配置中对于轴 0，将“是否使用”选择为“FALSE”，如图 7.6.58 所示，然后点击“全部下载”，并按照提示为设备重新上电。



图 7.6.58 轴参数配置

按照图 7.6.59 编辑指令并下载，启动程序监控后，将 M0.0 置为“1”，程序调用 MC_ReadAxisError 指令，读取的错误码为 4880，对应的 16 进制数为 0x1310，查找附录 E 中对应的错误说明为“轴状态为禁止”，这是因为轴参数配置中“是否使用”中选择了“FALSE”。

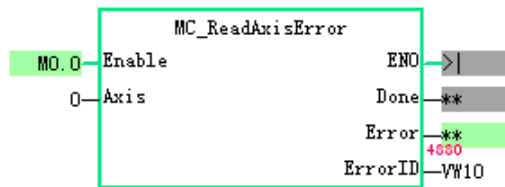


图 7.6.59 读取故障信息

7.6.14 读轴参数 MC_ReadParameter

● 指令参数

		读取轴参数。 ParameterNumber 为要读取参数的索引 ID（轴参数定义见附录 G），结果显示在 Value 引脚上。 轴出错时，读取的 Value 值为 0。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
ParameterNumber	字变量、常数	WORD	非负整数	10	读取的参数索引 ID，见附录 G
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	保留	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	
Value	双字变量	REAL	非负数	读取的值	

● 详细说明

通过 MC_ReadParameter 指令，读取 ParameterNumber 所指定 ID 号的参数值。不同的 ParameterNumber 对应着不同的意义（详见附录 G）。例如当 ParameterNumber 为 10 的时候则表示读取当前轴的实际速度值；当 ParameterNumber 为 12 时，读出的是运动系统允许的最大加速度的值。

● 时序图

图 7.6.60 为该指令的引脚时序图。Enable 置“1”时，当读取到轴参数正确时 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

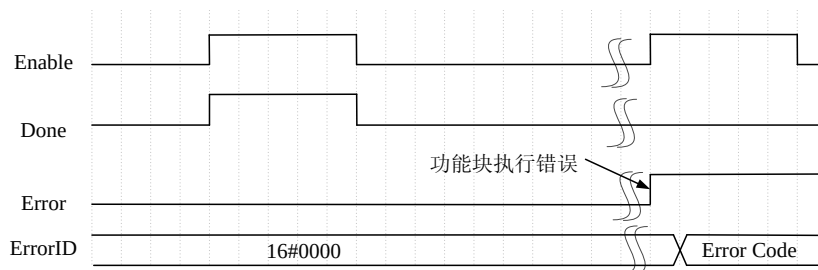


图 7.6.60 指令引脚时序图

● 应用示例

图 7.6.61 表示正在执行 MoveAbsolute 模块，由于目标速度为 10000，当前轴 0 从静止加速到目标速度，当接近目标位置的时候，轴 0 减速，使轴 0 到目标位置时，轴状态为静止状态。SM0.0 一直为“1”，则程序循环调用指令 MC_ReadParameter 读取当前轴 0 的运动状态，当 ParameterNumber 为 10，value 的值则代表当

前轴 0 运动的实际速度。

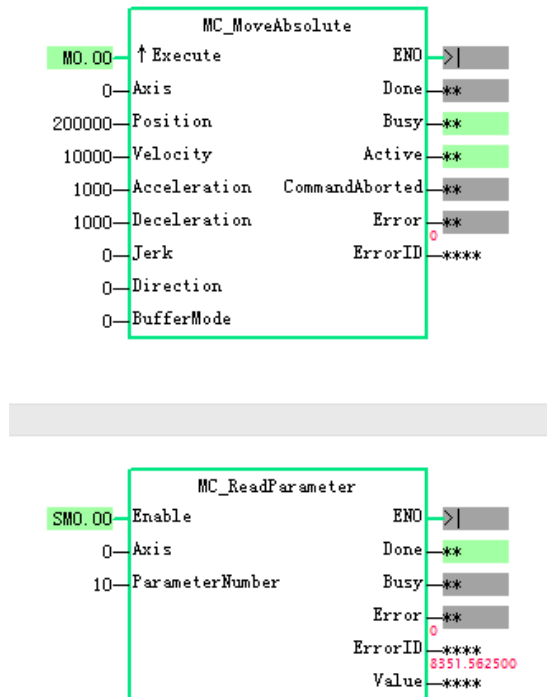


图 7.6.61 读取当前速度

7.6.15 写轴参数 MC_WriteParameter

● 指令参数

MC_WriteParameter Execute Axis ParameterNumber Value ENO Done Busy Error ErrorID 设置轴参数。 ParameterNumber 设置要写入参数的索引 ID（轴参数定义见附录 G），Value 为写入的参数值。 轴出错误时，写入的 Value 值无效。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
ParameterNumber	字变量、常数	WORD	非负整数	9	写入的参数，见附录 G
Value	双字变量、常数	REAL	实数	0	需写入的值
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	保留	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_WriteParameter 指令，修改 ParameterNumber 所指定 ID 号的参数值。不同的 ParameterNumber 对应着不同的意义（详见附录 G）。例如当 ParameterNumber 为 10 的时候则表示写入当前轴的实际速度值。

● 时序图

图 7.6.62 为该指令的引脚时序图。Execute 置“1”时，当修改轴的参数成功时 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

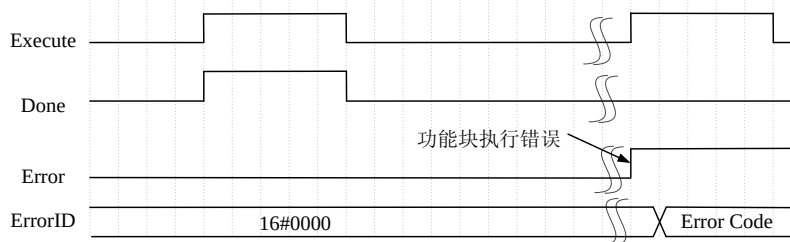


图 7.6.62 指令引脚时序图

● 应用示例

图 7.6.63 表示通过调用 MC_ReadParameter 来读取所设置的轴的参数，将 ParameterNumber 设置为 8，读取的为运动系统允许的最大速度值为 400000000。

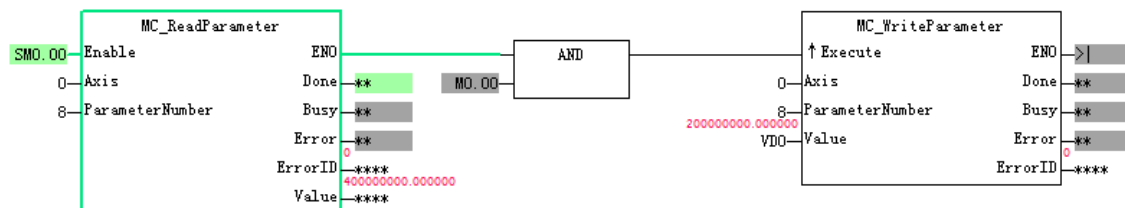


图 7.6.63 读取运动最大允许速度

图 7.6.64 中将 M0.0 置“1”后，调用 MC_WriteParameter 将运动系统允许的最大速度值设置为 200000000。

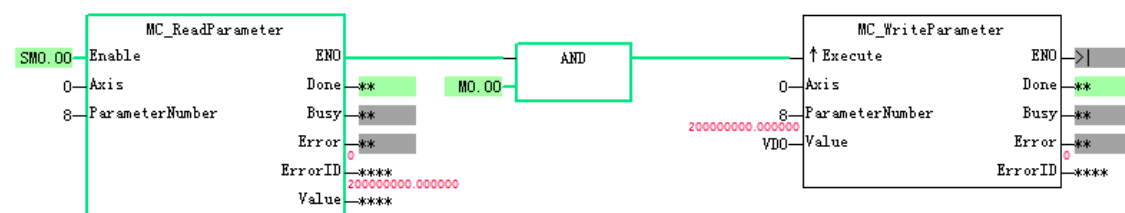


图 7.6.64 设置运动最大允许速度

7.6.16 读轴布尔型参数 MC_ReadBoolParameter

● 指令参数

MC_ReadBoolParameter << Enable ENO >> 0 Axis Done ** 4 ParameterNumber Busy ** Error ** ErrorID **** Value **		读取轴的布尔型参数 ParameterNumber 设置要读取的参数索引 ID（轴参数定义见附录 G），结果显示在 Value 引脚上。 轴出错时，此时读取的 Value 为“0”。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
ParameterNumber	字变量、常数	WORD	非负整数	4	读取的参数索引 ID，见附录 G
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	保留	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	
Value	位变量	BOOL	0 或 1	读取的值	

● 详细说明

通过 MC_ReadBoolParameter 指令,读取 ParameterNumber 所指定 ID 号的参数值。不同 ParameterNumber 对应着不同的意义(详见附录 G)。例如当 ParameterNumber 为 4 的时候,参数表示软件限制的正向位置的使能。

● 时序图

图 7.6.65 为该指令的引脚时序图。Enable 置“1”时,当读取到轴参数时 Done 被置“1”;如果在指令执行期间有错误发生,则 Error 被置“1”,可以通过 ErrorID(错误码)的值找到导致错误的原因。

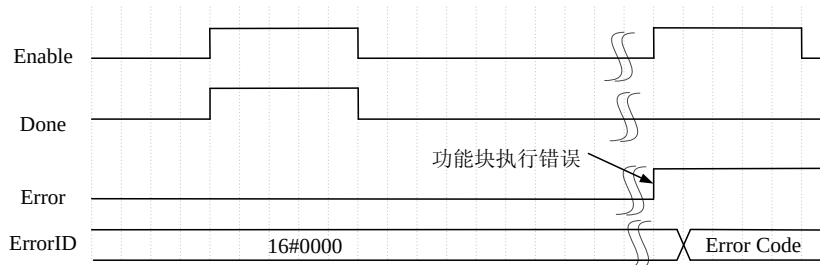


图 7.6.65 指令引脚时序图

● 应用示例

图 7.6.66 表示通过调用 MC_ReadBoolParameter 指令来读取当前轴的参数,将 ParameterNumber 设置为 4,表示软件限制正向位置的使能与否。

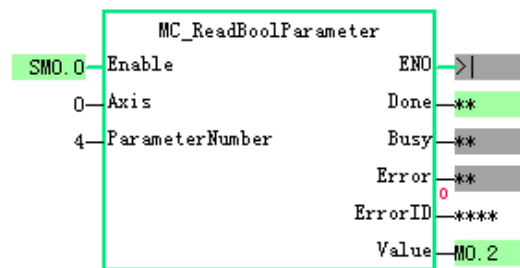
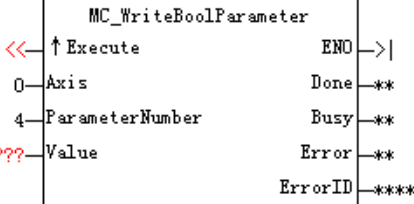


图 7.6.66 读取当前轴的 bool 型参数

7.6.17 写轴布尔型参数 MC_WriteBoolParameter

● 指令参数

		设置轴的布尔型参数 ParameterNumber 设置要写入的参数索引 ID (轴参数定义见附录 G), Value 为要写入的参数值。 轴出错时, 写入的 Value 值无效。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
ParameterNumber	字变量、常数	WORD	非负整数	4	写入的参数, 见附录 G
Value	位变量	BOOL	1 或 0	0	需写入的值
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	保留	
Error	位变量	BOOL	0 或 1	为“1”时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

通过 MC_WriteBoolParameter 指令，修改 ParameterNumber 所指定 ID 号的参数值。不同的 ParameterNumber 对应着不同的意义（详见附录 G）。例如当 ParameterNumber 为 4 的时，参数表示软件限制的正向位置的使能与否。

● 时序图

图 7.6.67 为该指令的引脚时序图。Execute 置“1”时，当修改轴的参数成功时 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

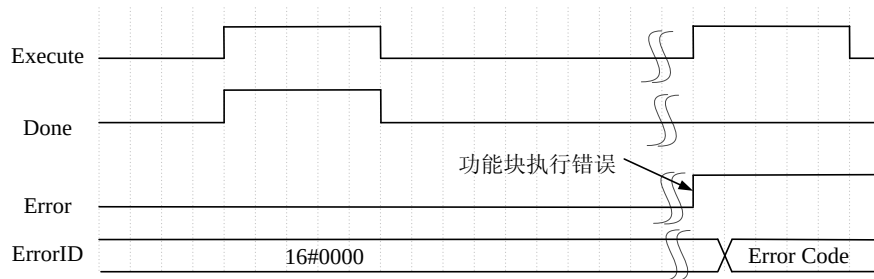


图 7.6.67 指令引脚时序图

● 应用示例

图 7.6.68 表示通过调用 MC_ReadBoolParameter 指令来读取当前轴的参数,将 ParameterNumber 设置为 4, 表示软件限制正向位置的使能与否。当 M0.1 置为“1”后，会修改软件限制正向位置的使能位。

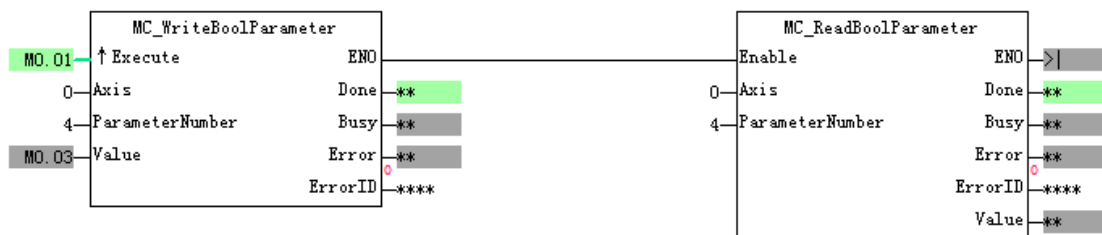


图 7.6.68 设置布尔型参数

7.6.18 读轴位置 MC_ReadActualPosition

● 指令参数

MC_ReadActualPosition << Enable 0 Axis ENO Done Busy Error ErrorID Position		读取轴的当前绝对位置 指令使能后，实时读取轴的当前绝对位置值，结果显示在 Position 引脚上。 轴出错时，读取的 Position 值为 0。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	保留	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	
Position	双字变量	REAL	实数	轴的当前位置，单位 units	

● 详细说明

通过 MC_ReadActualPosition 指令，读取轴的实际位置。单轴控制指令中，控制器对伺服器是以脉冲为

单位，而对轴的速度的控制也是以脉冲/秒为单位。MC_ReadActualPosition 以输出了多少脉冲来表示轴的实际位置。

● 时序图

图 7.6.69 为该指令的引脚时序图。Enable 置“1”时，当读取到轴位置后 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

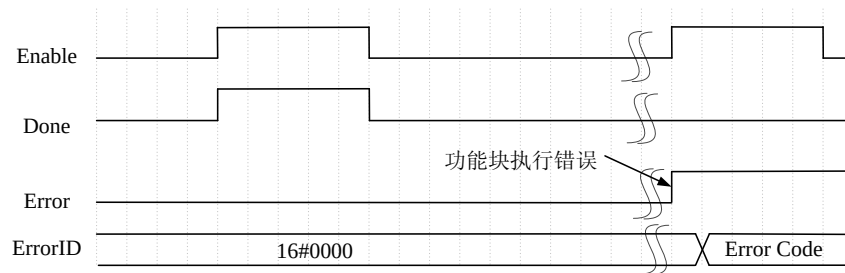


图 7.6.69 指令引脚时序图

● 应用示例

图 7.6.70 中，当 M0.0 置为“1”时，程序调用 MC_MoveAbsolute 和 MC_ReadActualPosition 指令，轴开始运动，可以从 MC_ReadActualPosition 指令的 Position 引脚读出轴当前的实时位置。

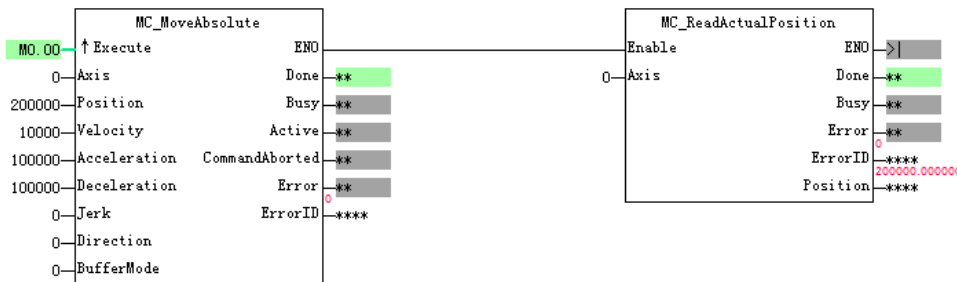


图 7.6.70 读取轴的当前位置

7.6.19 写轴位置 MC_WriteActualPosition

● 指令参数

MC_WriteActualPosition << ↑ Execute ENO → 0 Axis Done ** 0 Position Busy ** ??? ReferenceType Failure ** ??? Relative Error ** 0 ExecutionMode ErrorID **** 修改轴的当前绝对位置。 指令使能后，修改轴的当前绝对位置为 Position 值。 轴出错时，写入的 Position 值无效。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
Position	双字变量、常数	REAL	实数	0	轴的位置，单位 units
ReferenceType	位变量	BOOL	--	保留	保留
Relative	字变量	WORD	--	保留	保留
ExecutionMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Failure	位变量	BOOL	0 或 1	为“1”时，表示写位置失败	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_WriteActualPosition 指令，设置轴的实际位置。该指令仅在高电平时执行，且当前轴处于停止运动状态。

● 时序图

图 7.6.71 为该指令的引脚时序图。Busy 和 Enable 同时被置“1”；当轴的实际位置设置成功时 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

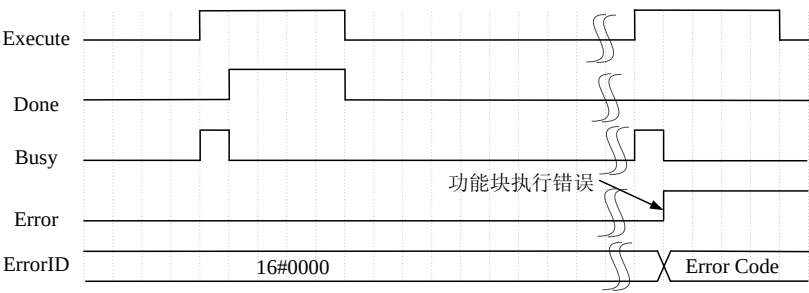


图 7.6.71 指令引脚时序图

● 应用示例

图 7.6.72 中将 M0.0 置为“1”，程序调用 MC_WriteActualPosition 指令，将轴 0 的位置设置为 Position 引脚对应的值，程序调用 MC_ReadActualPosition 指令，读取轴 0 的位置。

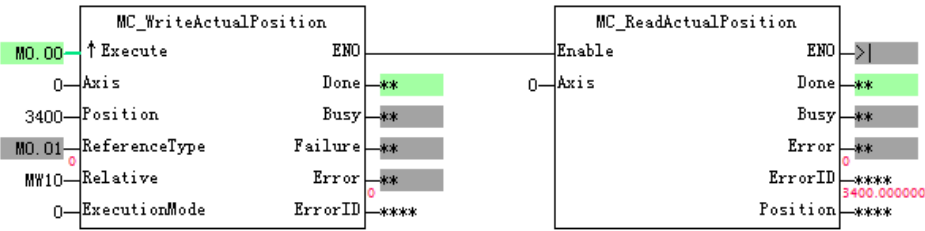


图 7.6.72 设置位置和读取位置

7.6.20 编码器轴配置 MC_CoderMap

● 指令参数

		建立或取消输出轴与高速计数之间的连接关系。 当 Enable 为“1”时，建立连接，输出轴会根据反馈计数的值，按照给定的倍比关系，输出对应的脉冲。 当 Enable 为“0”时，取消输出轴与反馈计数之间的关系。反馈计数发生变化时，轴不再跟随其输出。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID，该轴关联到反馈计数上
Mode	字变量、常数	WORD	非负整数	0	高速 DI 的接收模式，需与 HDEF 指令中的 Mode 引脚一致
Ratio	双字变量、常数	REAL	实数	1.0	输出轴与反馈计数之间的倍比关系
Type	字变量、常数	WORD	--	0	保留

输出	变量区	数据类型	数据范围	注解
ENO	位变量	BOOL	0 或 1	能流输出
Busy	位变量	BOOL	0 或 1	为“1”时，表示输出轴与反馈计数之间已建立联系
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）

● 详细说明

编码器轴为实现轴随输入脉冲信号情况而运动，当输入脉冲信号发生变化（频率或脉冲数）编码器轴会随着输入信号而做相应的运动，编码器轴运动会自动与输入信号保持一致。通过 MC_CoderMap 指令，建立或取消输出轴与高速计数之间的连接关系，使输出轴能根据反馈计数进行输出。

指令使能后，根据 HDEF 计算脉冲数，通过 Ratio 输出相应指令单位脉冲数，计算公式如下：

$$P_{out} = (P_{fbd} - P_{fbd1}) \times k \times \frac{P}{s}$$

其中：

P_{out} ——为实际输出脉冲数；

P_{fbd} ——为当前 HDEF 计数脉冲；

P_{fbd1} ——为上一周期 HDEF 计数脉冲；

k ——Ratio 输入倍比

P ——伺服单周脉冲

s ——伺服单周位移

图 7.6.73，若指令 Ratio 设置为 2，伺服单周脉冲 P 设置为 25600，伺服单周位移 s 设置为 4mm，则编码器轴 HDEF 采集到的 1 个脉冲对应轴实际输出脉冲个数 P_{out} 为：

$$P_{out} = 1 \times 2 \times \frac{25600}{4} = 12800$$

此时平台移动 2mm，若需要输入脉冲与输出脉冲保持 1:1 关系，可将 Ratio 设置为 s / P 。

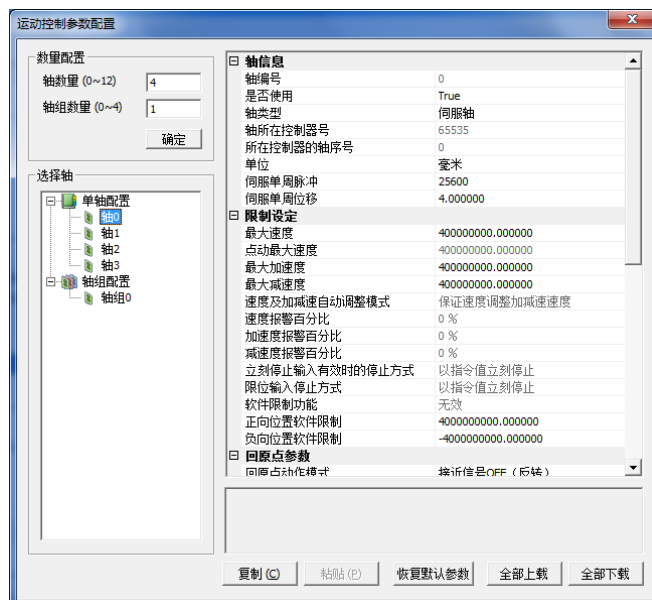


图 7.6.73 运动控制参数配置

当参数配置错误或指令执行错误时，Error 被置“1”，ErrorID 为错误码，根据错误码可查找到相应的错误，详见附录 E。

● 应用示例

图 7.6.73，将轴 0 运动控制参数单位设置为毫米，单周脉冲设置为 25600，单轴位移设置为 4。通过轴 1 单轴指令发脉冲，其中轴 1 高速脉冲输出接高速输入 HI0，轴 1 方向输出接高速输入 HI1，轴 0 高速输出接高速输入 HI2，轴 0 方向输出接高速输入 HI3。

(1) 按指令单位控制

此方式输入一个脉冲，平台移动 Ratio 毫米。如图 7.6.74，MC_CoderMap 指令高速 DI 的接收模式设置为 0x100，倍比关系 Ratio 设置为 2，即 1 个脉冲轴 0 运动 2mm。轴 1 调用 MC_MoveRelative 指令，用来产生 100 个脉冲。

使能 S1.0 将轴 0 由伺服轴切换为编码器轴，使能 M3.0 轴 1 产生 100 个脉冲，VD0 记录编码器轴输入脉冲数为 100，VD500 显示轴 0 运动位移为 $2 \times 100 = 200\text{mm}$ ，VD4 显示轴 0 实际输出脉冲为 $200 \times 25600 / 4 = 1280000$ （正负表示方向）。

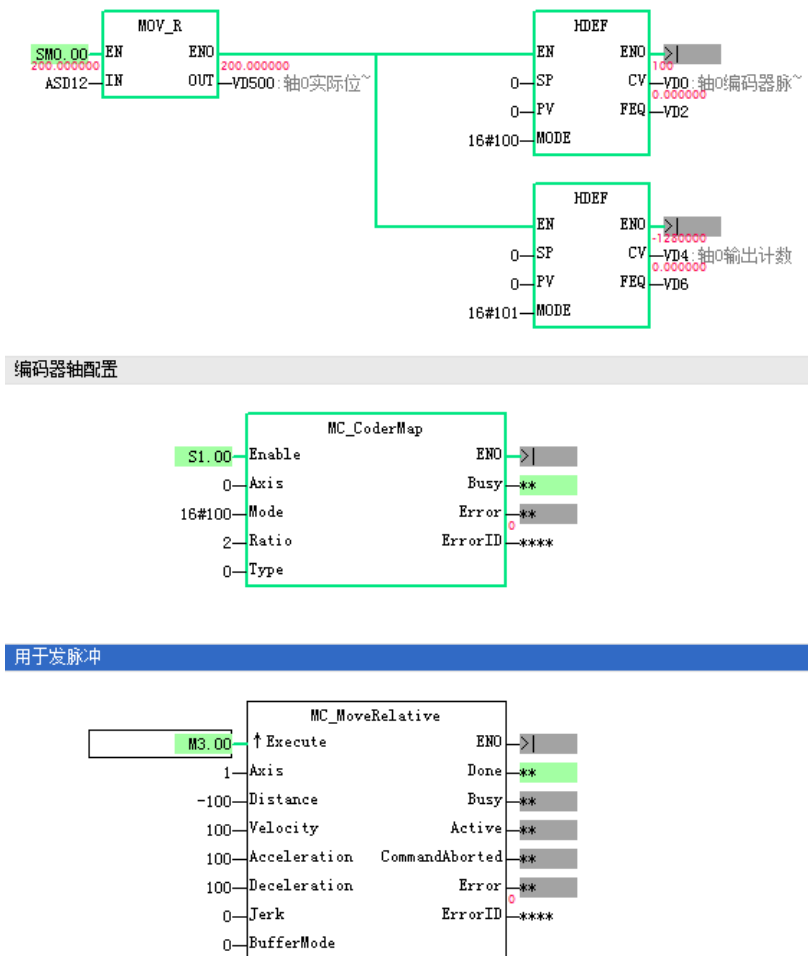


图 7.6.74 编码器轴按指令单位控制

(2) 按脉冲单位控制

此方式需要先保证输入脉冲与输出脉冲为 1:1 关系。如图 7.6.75 所示，MC_CoderMap 指令高速 DI 的接收模式设置为 0x100，倍比关系 Ratio 设置为 $2 \times 4 / 25600 = 0.0003125$ ，即输入 1 个脉冲轴 0 会输出 2 个脉冲。调用 MC_MoveRelative 指令，采用轴 1 发 640000 个脉冲。

使能 S1.0 将轴 0 由伺服轴切换为编码器轴，使能 M3.0 轴 1 产生 640000 个脉冲，VD0 记录编码器轴输入脉冲数为 640000，VD500 显示轴 0 运动位移为 $2 \times 640000 \times 4 / 25600 = 200\text{mm}$ ，VD4 显示轴 0 实际输出脉冲为 $2 \times 640000 = 1280000$ （正负表示方向）。

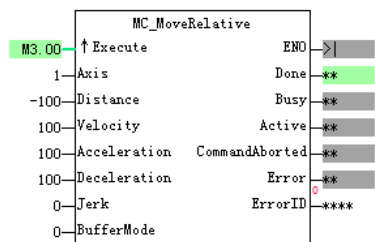
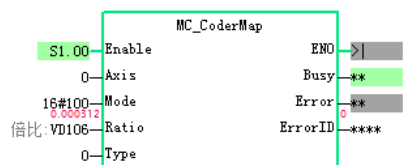
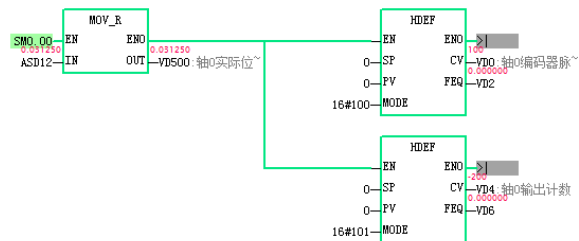
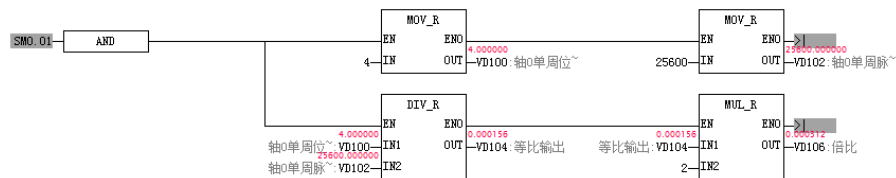


图 7.6.75 编码器轴按脉冲单位控制

7.6.21 立即停止配置指令 MC_EmgStopSet

● 指令参数

MC_EmgStopSet << Enable 0 Axis ?? ? DigitalInput 0 Mode ENO Done Busy Error ErrorID					
建立 DI 中断和轴之间的对应关系，当 DI 触发中断时立即停止对应轴脉冲输出。					
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
Axis	字变量、常数	WORD	非负整数	0	轴 ID
DigitalInput	I 区位变量	BOOL	0 或 1	0	外部中断信号输入引脚
Mode	字变量、常数	WORD	非负整数	0	中断触发模式： 0: 上升沿触发 1: 下降沿触发 3: 上升下降沿都可以触发
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	

Done	位变量	BOOL	0 或 1	为“1”时，表示轴立即停止操作完成
Busy	位变量	BOOL	0 或 1	为“1”时，表示轴 DI 已绑定，等待中断信号
Error	位变量	BOOL	0 或 1	为“1”时，表示绑定出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）

● 详细说明

立即停止配置指令将 DI 引脚和轴进行绑定，Enable 引脚置“1”后，获取 Mode 引脚的中断模式配置信息，并打开对应 DI 的外部 IO 中断。当 DI 有中断触发信号输入时，立刻停止其对应轴脉冲输出操作，实现立即停止脉冲输出功能。

当多路 DI 同时绑定同一个轴时，需要同时调用多个功能块为相同的轴指定不同的 DI 引脚。其中，任意引脚触发中断都会停止对应轴的脉冲输出。

● 时序图

图 7.6.76 为指令执行时序图。t1 时刻，功能块使能，CPU 配置 IO 中断模式，此时 Busy 信号被置“1”，在立即停止操作完成前 Busy 位一直为“1”。t2 时刻，中断模式为上升沿触发模式或上升下降沿触发模式，DI 引脚输入上升沿信号，此时 FPGA 停止对应轴脉冲输出操作，操作完成后 Done 引脚被置“1”。t5 时刻，中断模式配置为下降沿触发模式或上升下降沿触发模式，DI 引脚输入下降沿信号，FPGA 停止对应轴输出脉冲操作，操作完成后 Done 引脚被置“1”。

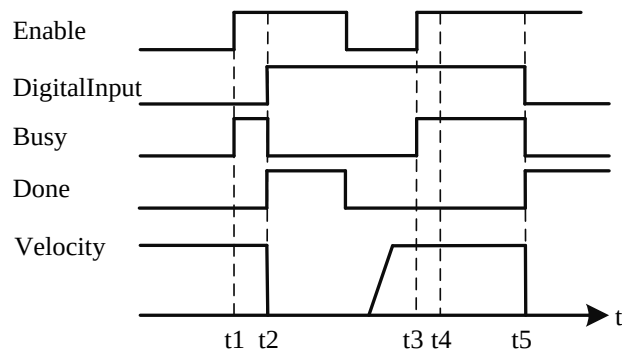


图 7.6.76 指令执行时序图

● 应用示例

图 7.6.77 为多个 DI 绑定同一个轴的示例工程。V0.0 置 1 后，轴 0 同时绑定正向限位 I1.0 和负向限位 I1.1 信号，当正向运动时，I1.0 上升沿到来，轴 0 会立即停止；当负向运动时，I1.1 上升沿到来，轴 0 会立即停止。

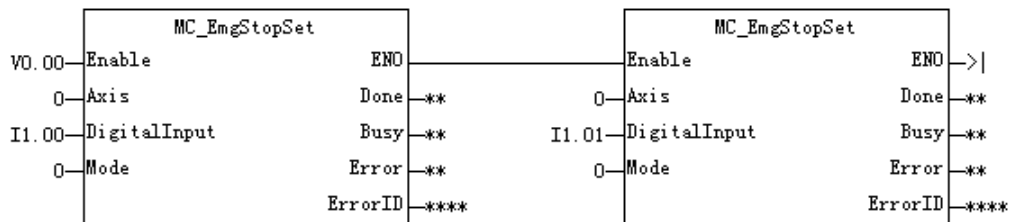


图 7.6.77 多个 DI 绑定同一个轴示例工程

7.7 多轴指令

7.7.1 凸轮运行 MC_CamIn

● 指令参数

控制凸轮按照凸轮表运动。 指令使能后，主轴与从轴按照设定的凸轮表中主轴与从轴位置开始同步运动，主轴保持恒速运动，从轴以当前速度为起始速度，随着主轴自动变速运动。 非周期执行时，凸轮表执行结束后，主轴与从轴的速度为 0。 当凸轮周期执行并且选择绝对位置时，凸轮表中的数据应大于 2 行。 周期执行时凸轮表中最后一行主轴与从轴的位置应与首行一样。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Master	字变量、常数	WORD	非负整数	0	主轴 ID
Slave	字变量、常数	WORD	非负整数	1	从轴 ID
Periodic	位变量	BOOL	0 或 1	0	周期运行标志
StartPosition	双字变量、常数	REAL	--	0	保留
MasterStartDistance	双字变量、常数	REAL	--	0	保留
MasterOffset	双字变量、常数	REAL	实数	0	主轴位置的偏移
SlaveOffset	双字变量、常数	REAL	实数	0	从轴位置的偏移
MasterScaling	双字变量、常数	REAL	实数	0	主轴位置偏移因子，默认为 1
SlaveScaling	双字变量、常数	REAL	实数	0	从轴位置偏移因子，默认为 1
StartMode	字变量、常数	WORD	非负整数	0	开始模式：绝对=0，相对=1
ReferenceType	字变量、常数	WORD	--	0	保留
Direction	字变量、常数	WORD	--	0	保留
CamTransition	字变量、常数	WORD	--	0	保留
CamTableID	字变量、常数	WORD	0~255	0	凸轮表的 ID
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
InCam	位变量	BOOL	0 或 1	为“1”时，表示开始凸轮耦合	
InSync	位变量	BOOL	0 或 1	为“1”时，表示进入同步运行状态	
EndOfProfile	位变量	BOOL	0 或 1	为“1”时，表示执行完倒数第二条记录	
Index	字变量	WORD	非负整数	正在执行凸轮表索引号	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_CamIn 指令，顺序执行凸轮表。在执行的过程中，主轴以当前的实际速度匀速运动，从轴则按照当前执行的凸轮表参数进行运动；当主轴达到凸轮表中设定的位置时，从轴也达到凸轮表中对应的目标位置。

凸轮周期执行时，执行完最后一条记录，会直接跳转到第一条记录执行。

凸轮在实际运动过程中目标位置计算如下：

$$\text{Actpos} = \text{CamPos} \times \text{Scale} + \text{Offset} \quad (\text{主从轴})$$

使用该指令时，如果主从轴的位置不在相应的偏移值内或从轴在主轴运动到目标位置时达不到既定的位置和速度，就会输出错误信息。

● 时序图

图 7.7.1 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；开始凸轮耦合时 InCam 被置“1”；接合完成时 InSync 被置“1”；执行完倒数第二条记录后 EndOfProfile 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID（错误码）的值找到导致错误的原因。

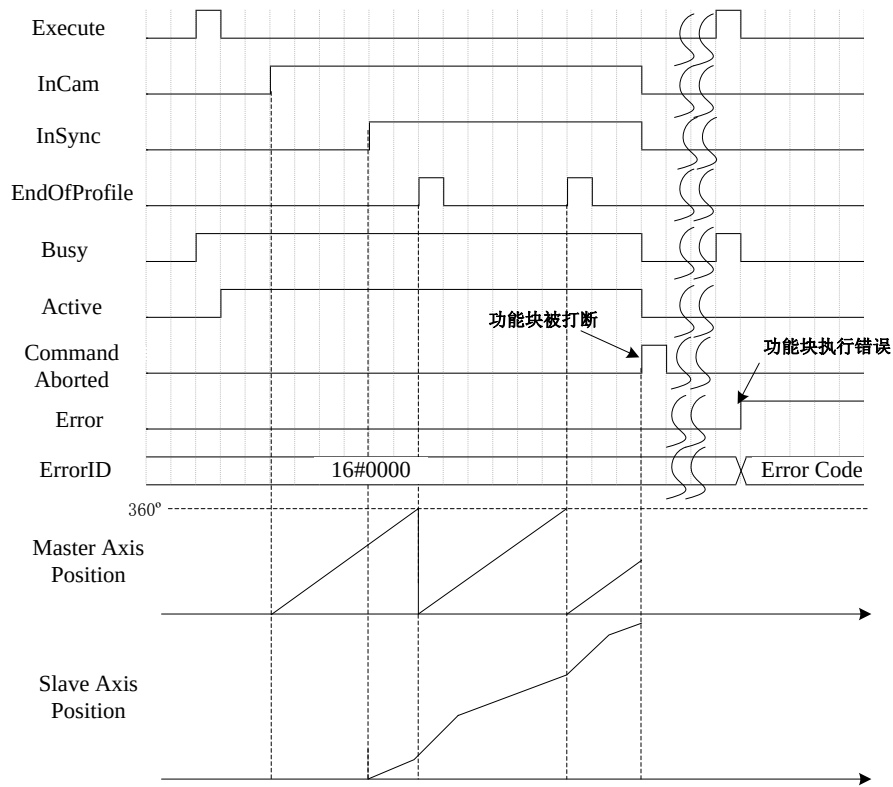


图 7.7.1 指令引脚时序和主、从轴位置变化图

● 应用示例

图 7.7.2 中将 M0.0 置为“1”，程序调用 CamTableCreate 指令，创建凸轮表；将 M0.1 置为“1”，程序调用 MC_MoveVelocity 指令和 MC_ReadActualPosition 指令，使主轴 0 轴按照一定的速度匀速运动，并实时的监控主轴和从轴的位置。

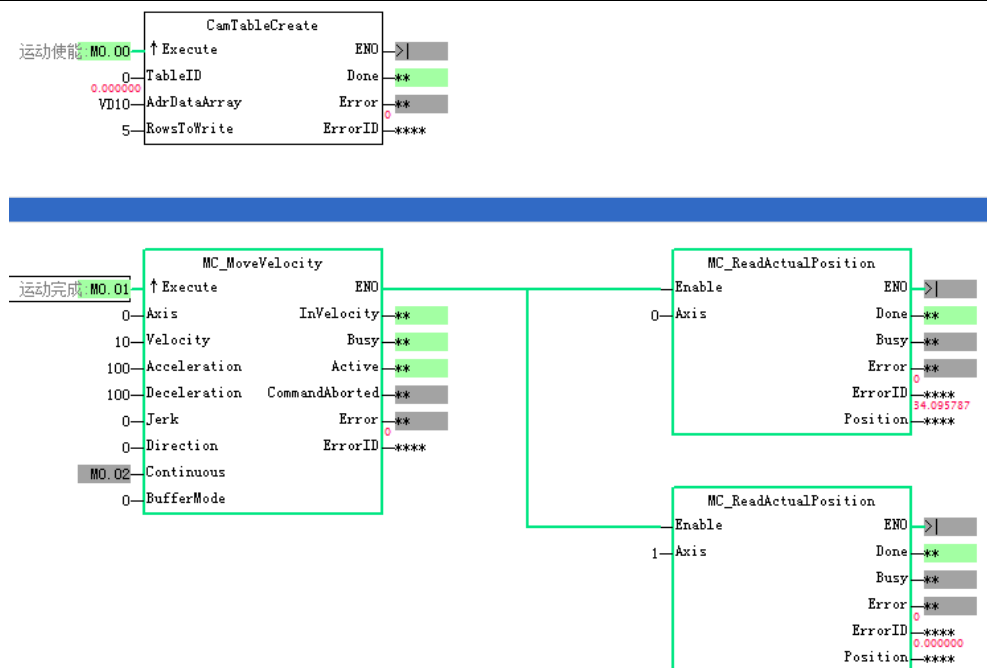


图 7.7.2 创建凸轮表，主轴匀速运动和主轴从轴实时位置的监控

图 7.7.3 中将 M0.3 置“1”，程序调用 MC_CamIn 指令，使主轴和从轴按照凸轮表中设定的位置运动。

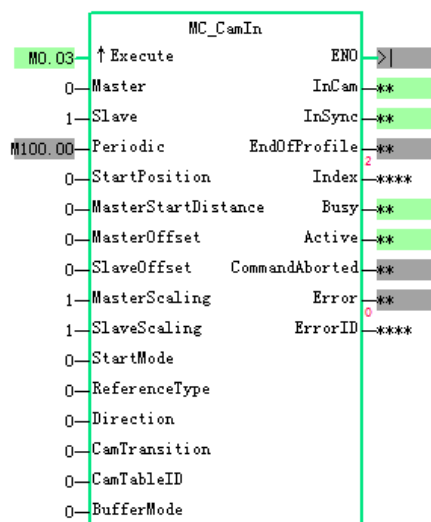


图 7.7.3 凸轮耦合运动

图 7.7.4 为在“状态符号表”中设置的凸轮表参数。

绝对地址	Modbus寄存器地址	数据类型	当前值(10)	新数值	参数说明
VD10	2346	四字节浮点数	0.000000	0	主轴第1个旋转角度
VD12	2348	四字节浮点数	0.000000	0	从轴第1个位置
VD14	2350	四字节浮点数	60.000000	60	主轴第2个旋转角度
VD16	2352	四字节浮点数	100.000000	100	从轴第2个位置
VD18	2354	四字节浮点数	90.000000	90	主轴第3个旋转角度
VD20	2356	四字节浮点数	150.000000	150	从轴第3个位置
VD22	2358	四字节浮点数	180.000000	180	主轴第4个旋转角度
VD24	2360	四字节浮点数	200.000000	200	从轴第4个位置
VD26	2362	四字节浮点数	360.000000	360	主轴第5个旋转角度
VD28	2364	四字节浮点数	190.000000	190	从轴第5个位置

图 7.7.4 凸轮表设定的参数

7.7.2 凸轮分离 MC_CamOut

● 指令参数

MC_CamOut ↑ Execute 1 Slave 100000 Deceleration 0 Jerk 0 OutMode ENO Done Busy CommandAborted Error ErrorID ENI ** ** ** ** **** 控制从轴脱离主轴，停止凸轮按照凸轮表运动。 指令使能后，主轴会继续执行到目标位置，到达目标位置后主轴停止运行，从轴以当前速度恒速运行。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Slave	字变量、常数	WORD	非负整数	1	从轴 ID
Deceleration	双字变量、常数	REAL	--	100000	保留
Jerk	双字变量、常数	REAL	--	100000	保留
OutMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

MC_CamOut 指令使从轴脱离主轴，停止凸轮表的执行，主轴会继续执行当前凸轮表位置，到达目标位置后停止，从轴会以当前的速度，匀速执行，直至被其他指令打断。主从轴分离成功时，Done 被置“1”。

● 时序图

图 7.7.5 为该指令的引脚时序和速度、位置变化图；Busy 和 Execute 同时被置“1”；当分离完成时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 被置“0”；如果指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID（错误码）的值找到导致错误的原因。

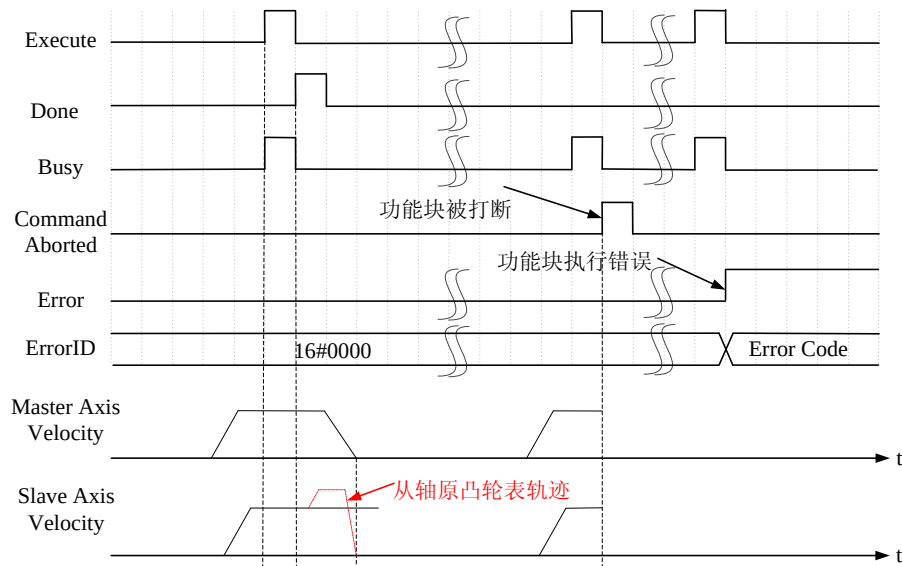


图 7.7.5 指令引脚时序和主、从轴速度变化图

● 应用示例

图 7.7.6 中将 M0.0 置为“1”，程序调用 CamTableCreate 指令，创建凸轮表；将 M0.1 置为“1”，程序调用 MC_MoveVelocity 指令和 MC_ReadActualPosition 指令，使主轴 0 轴按照一定的速度匀速运动，并实时的监控主轴和从轴的位置。

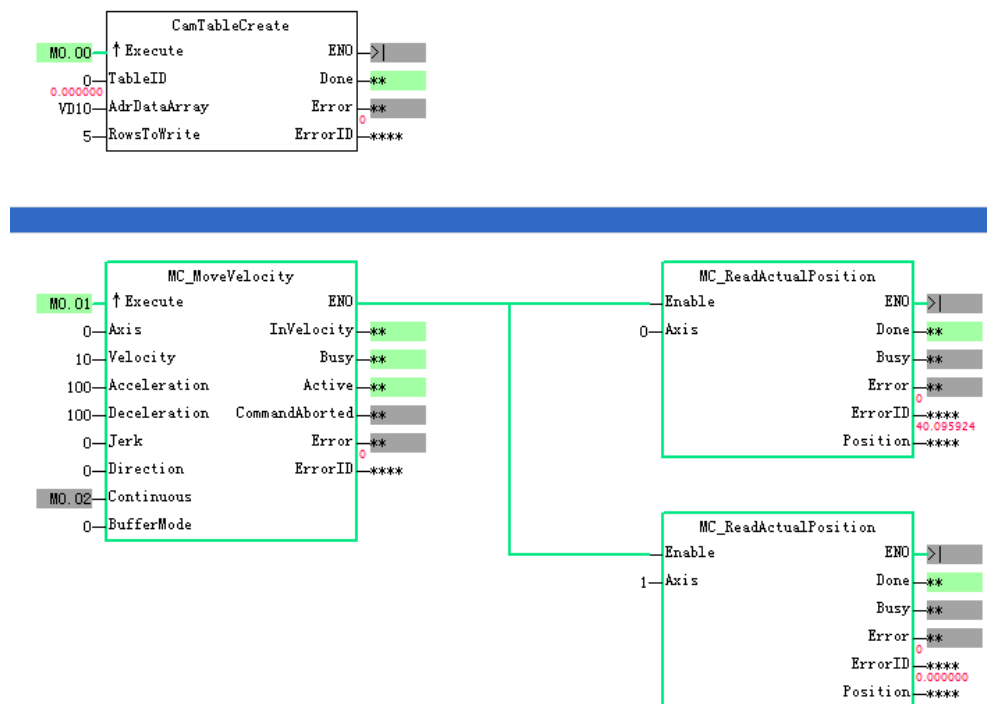


图 7.7.6 创建凸轮表，主轴匀速运动和主轴从轴实时位置的监控

图 7.7.7 中将 M0.3 位置“1”，程序调用 MC_CamIn 指令，使主轴和从轴按照凸轮表中设定的位置运动，在主轴和从轴运动过程中，将 M0.4 置为“1”，程序调用 MC_CamOut 指令，从轴凸轮表中分离并以当前的速度匀速运动下去。

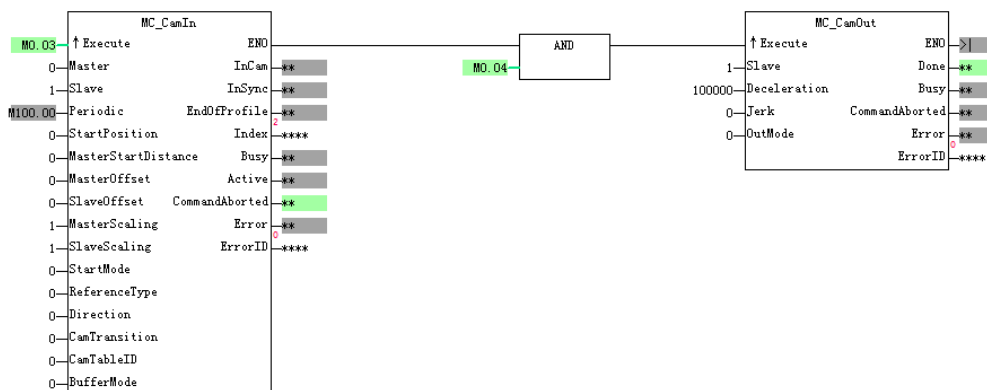


图 7.7.7 凸轮分离

图 7.7.8 为在“状态符号表”中设置的凸轮表参数。

绝对地址	Modbus寄存器地址	数据类型	当前值(10)	新数值	参数说明
VD10	2346	四字节浮点数	0.000000	0	主轴第1个旋转角度
VD12	2348	四字节浮点数	0.000000	0	从轴第1个位置
VD14	2350	四字节浮点数	60.000000	60	主轴第2个旋转角度
VD16	2352	四字节浮点数	100.000000	100	从轴第2个位置
VD18	2354	四字节浮点数	90.000000	90	主轴第3个旋转角度
VD20	2356	四字节浮点数	150.000000	150	从轴第3个位置
VD22	2358	四字节浮点数	180.000000	180	主轴第4个旋转角度
VD24	2360	四字节浮点数	200.000000	200	从轴第4个位置
VD26	2362	四字节浮点数	360.000000	360	主轴第5个旋转角度
VD28	2364	四字节浮点数	190.000000	190	从轴第5个位置

图 7.7.8 凸轮表设定的参数

7.7.3 齿轮运行 MC_GearIn

● 指令参数

控制齿轮按照电子齿轮比运动。 指令使能后，从轴以当前速度为起始速度，运动到与主轴速度成比例的速度时，与主轴同步运动，该比值为 RatioNumerator 与 RatioDenominator 的比值。 InGear 为“1”时，主轴与从轴齿轮耦合成功，之后两轴按照设定的比值同步运动。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Master	字变量、常数	WORD	非负整数	0	主轴 ID
Slave	字变量、常数	WORD	非负整数	1	从轴 ID
RatioNumerator	字变量、常数	WORD	正整数	0	速度比率分子，非正则比率为 1
RatioDenominator	字变量、常数	WORD	正整数	0	速度比率分母，非正则比率为 1
ReferenceType	字变量、常数	WORD	--	0	保留
MasterSyncPosition	双字变量、常数	REAL	--	0	保留
SlaveSyncPosition	双字变量、常数	REAL	--	0	保留
Velocity	双字变量、常数	REAL	--	10000	保留
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；如果为 0，则无加速过程，单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；如果为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
StartSync	位变量	BOOL	0 或 1	保留	
InGear	位变量	BOOL	0 或 1	为“1”时，表示该命令完成	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴处于运动过程中	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GearIn 指令，使主从轴按照一定的速率比进行运动；指令执行时，主轴以当前的实际速度匀速运动，从轴则以主轴的速度作参考，按照设定的参数值与主轴联动，最后速度值与主轴成比率，以这个速度匀速运动。从轴和主轴的速度比为 RatioNumerator 与 RatioDenominator 的比值。当从轴达到目标的速度后，匀速运动。

主从轴同步运行时，InGear 输出。主从轴的速率比，可以在运行中动态修改。

● 时序图

图 7.7.9 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当主轴和从轴处于同步运行时 InGear 被置“1”；如果有其它的指令打断此指令，CommandAborted

被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

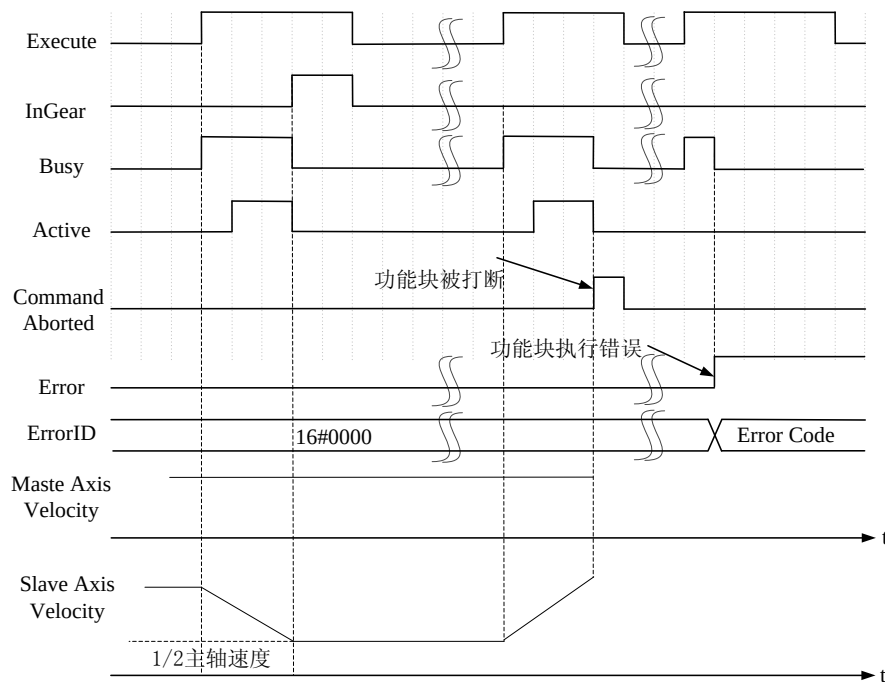


图 7.7.9 指令引脚时序和主、从轴速度变化图

● 应用示例

图 7.7.10 中将 M0.0 置为“1”，程序调用 MC_MoveVelocity 指令，控制主轴开始匀速运动，并实时的监视主轴和从轴的速度。

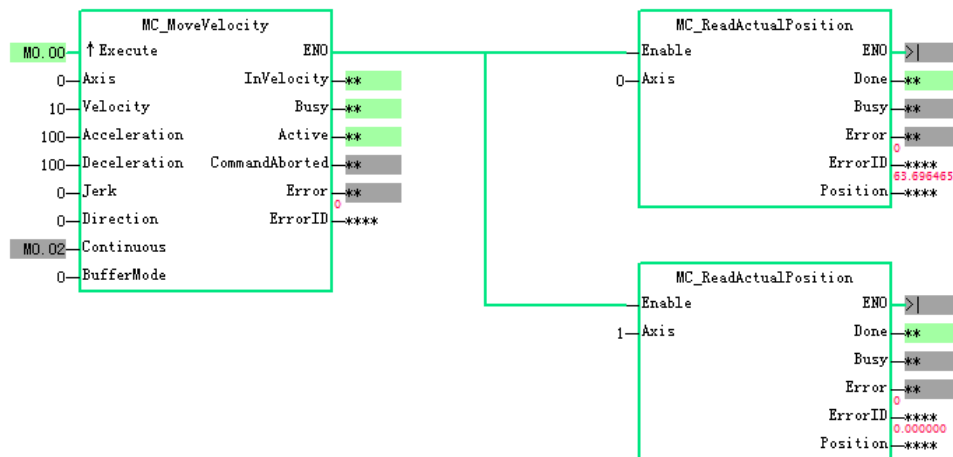


图 7.7.10 主轴匀速运动和主从轴的速度监控

图 7.7.11 中将 M0.1 置为“1”，程序调用 MC_GearIn 指令，控制从轴按照主轴的电子齿轮比开始运动。当达到目标速度后，InGear 被置“1”，从轴的速度为主轴速度的一半。

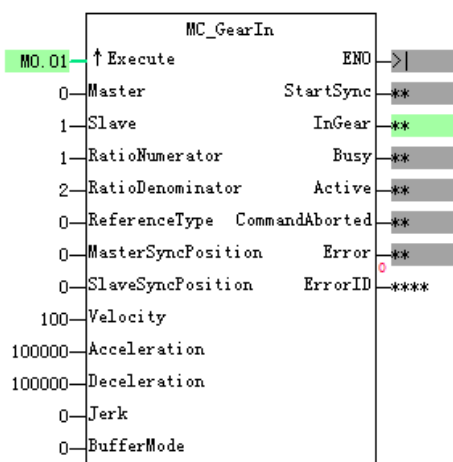


图 7.7.11 齿轮运行

7.7.4 齿轮分离 MC_GearOut

● 指令参数

		停止齿轮按照电子齿轮比运动。 指令使能后，从轴与主轴脱离齿轮耦合关系，并以当前速度恒速运动。之后主轴速度的改变，不影响从轴。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Slave	字变量、常数	WORD	非负整数	1	从轴 ID
Deceleration	双字变量、常数	REAL	--	100000	保留
Jerk	双字变量、常数	REAL	--	100000	保留
OutMode	双字变量、常数	REAL	--	0	保留
输入	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GearOut 指令，停止主从轴的速度关联；如果没有其他命令打断，从轴按照分离时的速度匀速运行；主从轴停止关联后，Done 被置“1”。

● 时序图

图 7.7.12 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当主轴和从轴停止按照设定的关系运行时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

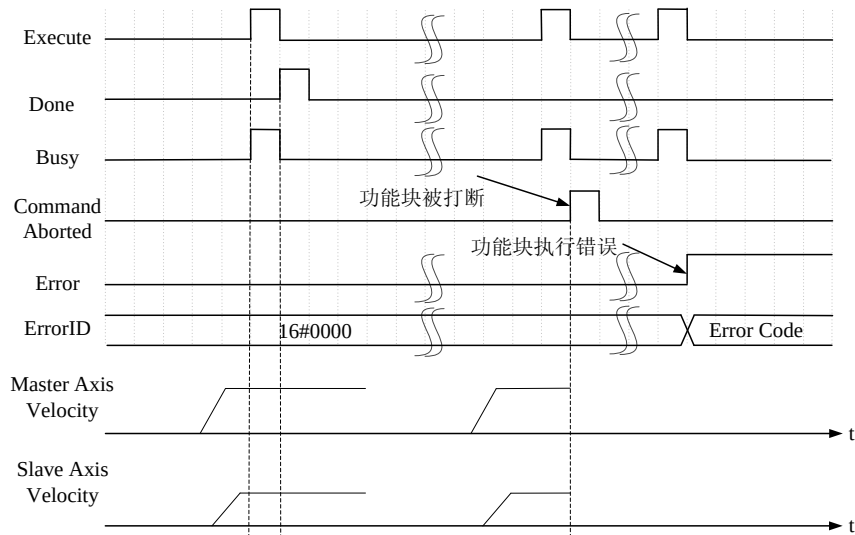


图 7.7.12 指令引脚时序和主、从轴速度变化图

● 应用示例

图 7.7.13 中将 M0.0 置为“1”，程序调用 MC_MoveVelocity 指令，控制主轴开始匀速运动，并实时的监视主轴和从轴的速度。

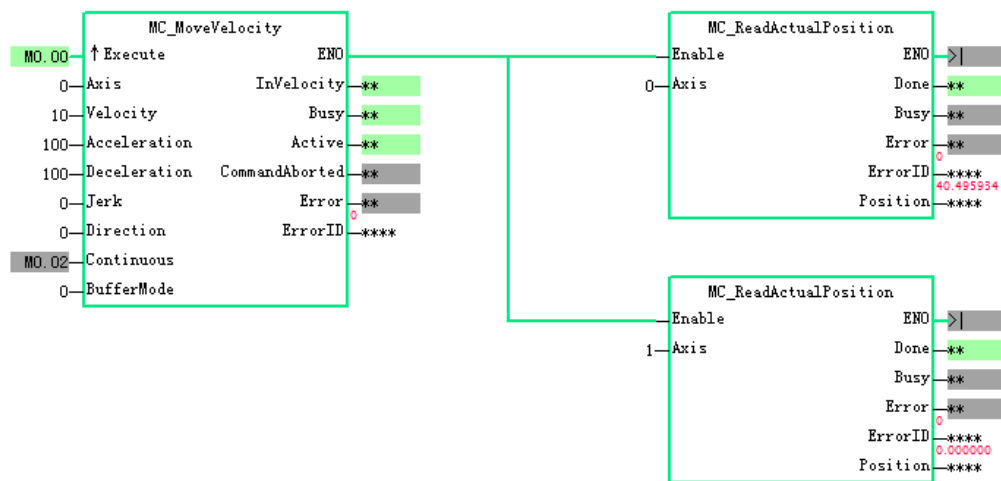


图 7.7.13 主轴匀速运动和主从轴速度监控

图 7.7.14 中将 M0.1 置为“1”，程序调用 MC_GearIn 指令，控制从轴按照主轴的电子齿轮比开始运动。当达到目标速度后，InGear 被置“1”，主轴与从轴速度相同。

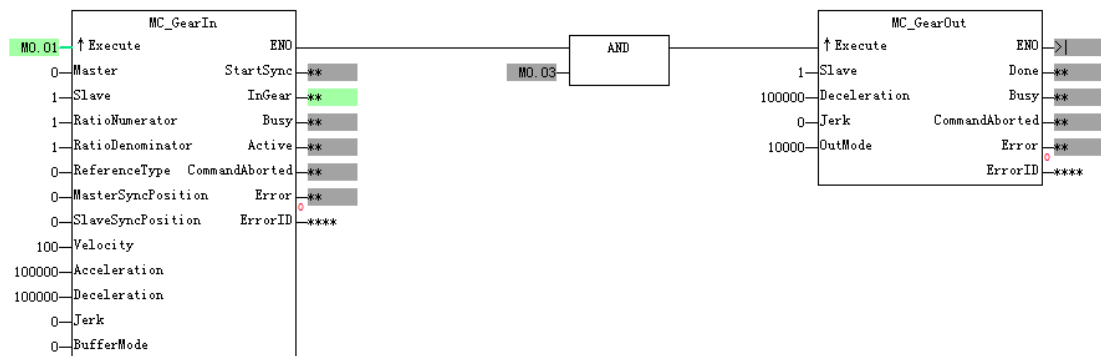


图 7.7.14 齿轮运行

图 7.7.15 中将 M0.3 置为“1”，程序调用 MC_GearOut 指令，控制从轴从电子齿轮耦合中分离，从轴会

以当前的速度继续匀速运动。

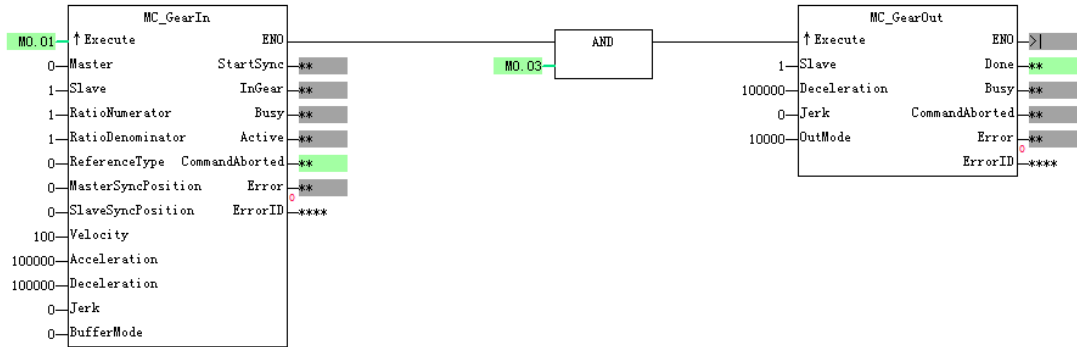


图 7.7.15 齿轮分离

7.7.5 相位偏移 MC_Phasing

● 指令参数

MC_Phasing ↑ Execute ENO 0 Master Done 1 Slave Busy 0 PhaseShift Active 10000 Velocity CommandAborted 100000 Acceleration Error 100000 Deceleration ErrorID 0 Jerk 0 BufferMode		控制主轴相位偏移设定的距离。 主轴目标位置为当前同步运动位置叠加 PhaseShift 值。 主轴目标速度为当前实际速度叠加 Velocity 值。 指令执行结束后，主轴速度恢复为指令开始执行叠加时的主轴的速度。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
Master	字变量、常数	WORD	非负整数	0	主轴 ID
Slave	字变量、常数	WORD	非负整数	1	从轴 ID
PhaseShift	双字变量、常数	REAL	非负数	0	相位偏移值（相对于主轴），units
Velocity	双字变量、常数	REAL	正数	10000	相位偏移时叠加速度，单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；如果为 0，则无加速过程，单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；如果为 0，则无减速过程，单位 units/s ²
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令在运行过程中	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_Phasing 指令，将主轴的位置相对同步运行位置偏移一个设置值；运动从当前的实际速度开始，运行到目标速度，最后再回到开始时的速度值，这一个变化过程所经过的距离就是设定的调整值。

● 时序图

图 7.7.16 为该指令的引脚时序和速度、位置变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期

被置“1”；当将主轴的位置相对实际运行位置偏移到设置值时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

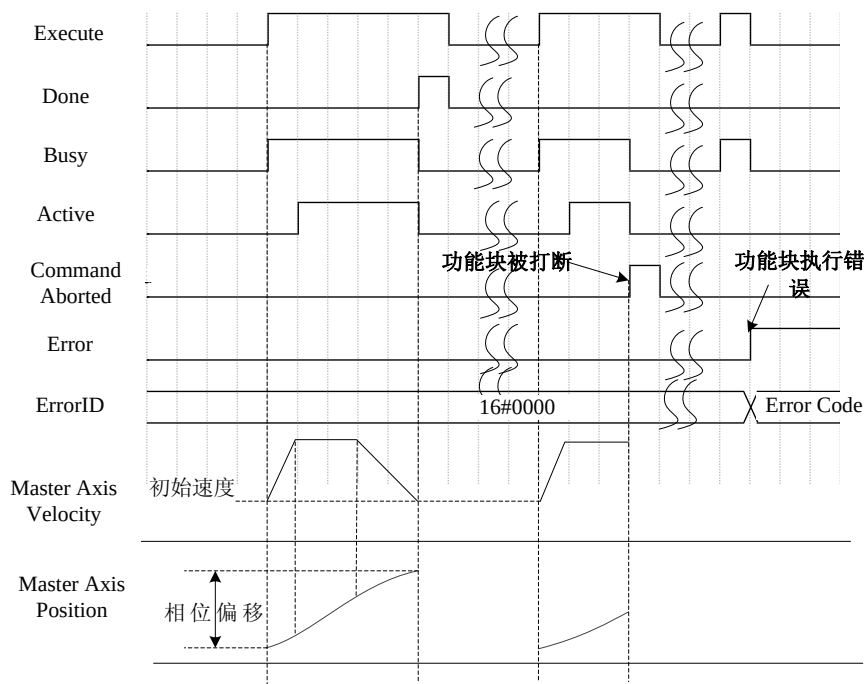


图 7.7.16 指令引脚时序和主轴速度、位置变化图

● 应用示例

图 7.7.17 中将 M0.0 置为“1”，程序调用 MC_MoveVelocity 指令，控制主轴开始匀速运动，并实时的监视主轴和从轴的速度。

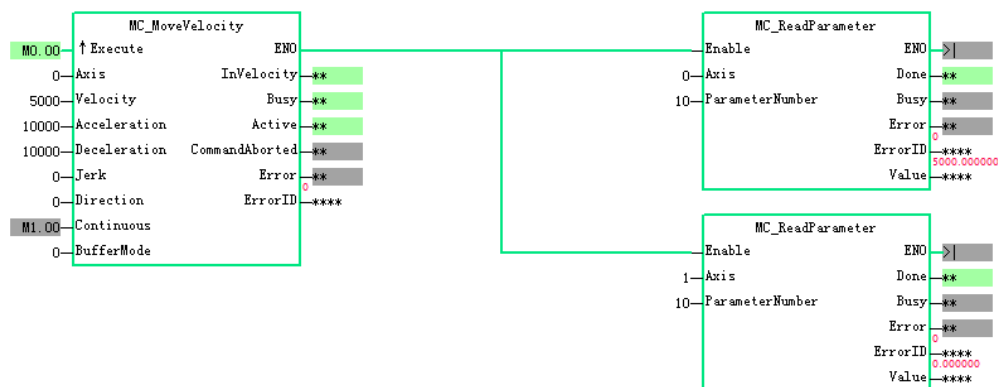


图 7.7.17 主轴匀速运动和主从轴速度监控

图 7.7.18 中将 M0.1 置为“1”，程序调用 MC_GearIn 指令，控制从轴按照主轴的电子齿轮比开始运动。当达到目标速度后，InGear 置“1”，从轴的速度为主轴速度的一半。然后将 M0.3 置为“1”，程序调用 MC_Phasing 指令，主轴按照给定的速度和相移量进行运动，主轴的速度先增加 MC_Phasing 指令 Velocity 的速度值后再减小到原来的速度值。

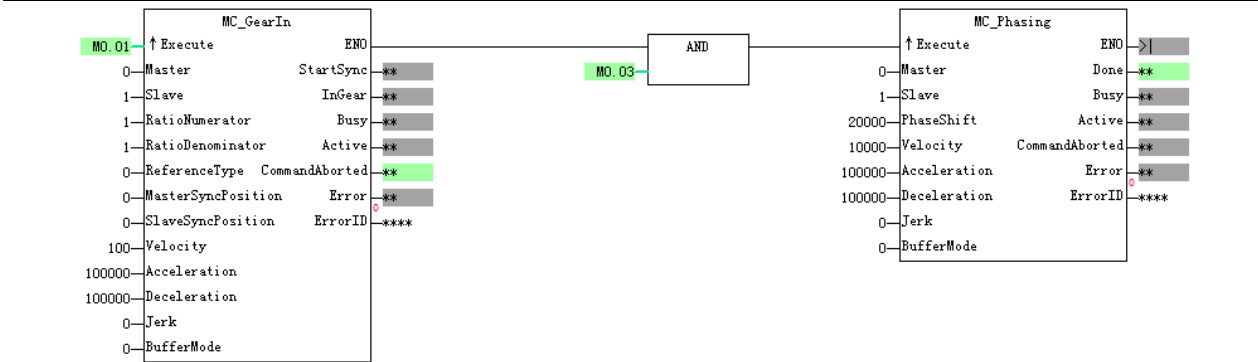


图 7.7.18 电子齿轮耦合以及主轴相位偏移

7.7.6 创建凸轮表 CamTableCreate

● 指令参数

CamTableCreate <<↑ Execute 0 TableID ????AdrDataArray 2 RowsToWrite ENO Done Error ErrorID → ** ** ****			创建凸轮表。 指令使能后，将创建一个新的凸轮表。 当凸轮周期执行并且选择绝对位置时，凸轮表中的数据应大于 2 行。 目前凸轮表最多可同时创建 3 个。		
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
TableID	字变量、常数	WORD	0~255	0	凸轮表的 ID
AdrDataArray	双字变量	REAL	实数	指定变量区中的值	存放凸轮表数据的起始地址
RowsToWrite	字变量、常数	WORD	非负整数	2	凸轮表的行数
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令在运行过程中	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 CamTableCreate 指令，创建一个指定大小和数据位置的凸轮表；凸轮表创建成功，Busy 被置“1”；否则会输出相应错误信息。

创建的凸轮表中 RowsToWrite 表示要执行的凸轮表的行数，AdrDataArray 存放的是凸轮表中主轴的位置和从轴的位置。

● 时序图

图 7.7.19 为该指令的引脚时序图。当创建凸轮表完成时 Busy 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

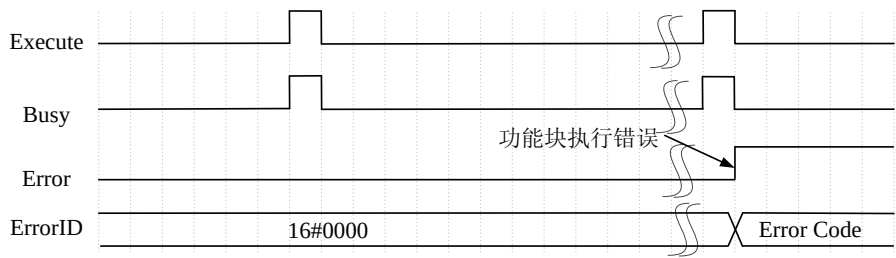


图 7.7.19 指令引脚时序图

● 应用示例

图 7.7.20 中将 M0.0 置为“1”，程序调用 CamTableCreate 指令，创建凸轮表。

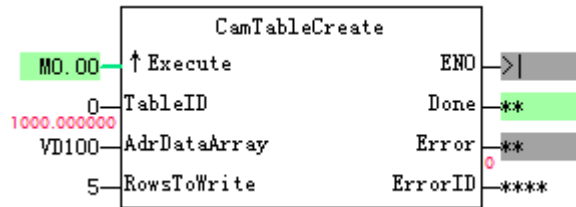


图 7.7.20 凸轮表的创建

图 7.7.21 为凸轮表设定的参数。VD100 中存放的是主轴第一个位置，VD102 中存放的是从轴第一个位置，VD104 中存放的是主轴第二个位置，VD106 中存放的是从轴第二个位置。

VD100	2438	四字节浮点数	1000.000000	1000
VD102	2438	四字节浮点数	2000.000000	2000
VD104	2440	四字节浮点数	3000.000000	3000
VD106	2442	四字节浮点数	4000.000000	4000

图 7.7.21 轮表设定的参数

7.7.7 删除凸轮表 CamTableDelete

● 指令参数

CamTableDelete Execute (上升沿触发) → TableID (0) → RowsToWrite (5) → ENO → Done → ** Error → ** ErrorID → ****					
删除凸轮表。 指令使能后，删除指定的凸轮表，如果指定的凸轮表不存在，指令报错。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
TableID	字变量、常数	WORD	0~255	0	凸轮表 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令在运行过程中	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 CamTableDelete 指令和设置的凸轮表 ID 删除指定 ID 的凸轮表。凸轮表删除成功，Busy 被置“1”，否则会输出相应错误信息。

● 时序图

图 7.7.22 为该指令的引脚时序图。当删除凸轮表完成时 Busy 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

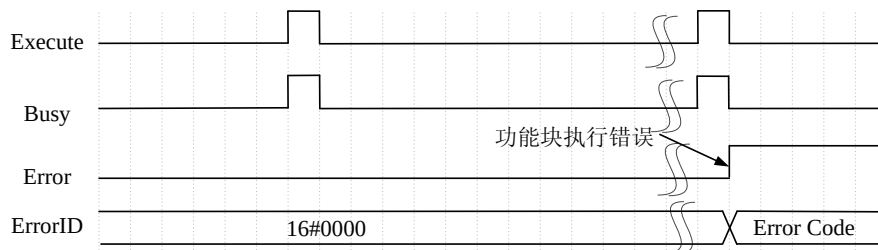


图 7.7.22 指令引脚时序图

● 应用示例

图 7.7.23 中将 M0.0 置为“1”，程序调用 CamTableCreate 指令，创建凸轮表。然后将 M0.1 置为“1”，程序调用 CamTableDelete 指令，删除创建的凸轮表，当删除成功后，Busy 被置“1”。

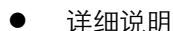


图 7.7.24 为凸轮表设定的参数。VD100 存放的是主轴的 1 号位置，VD102 中存放的是从轴的 1 号位置，VD104 存放的是主轴的 2 号位置，VD106 存放的是从轴的 2 号位置。

图 7.7.24 凸轮表设定的参数

7.7.8 写凸轮表数据 CamTableWriteData

● 指令参数



● 时序图

第 365 页

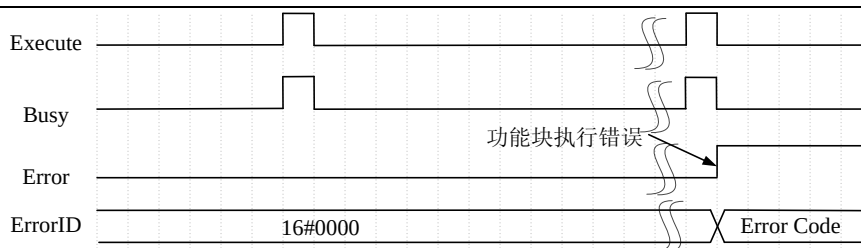


图 7.7.25 指令引脚时序图

● 应用示例

图 7.7.26 中将 M0.0 置为“1”，程序调用 CamTableCreate 指令，创建凸轮表。

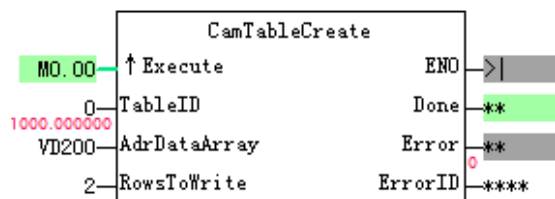


图 7.7.26 凸轮表的创建

图 7.7.27 中将 M0.1 置为“1”程序调用 CamTableWriteData 指令，从 CamTableWriteData 的 AdrDataArray 引脚中读取数据写入凸轮表中。

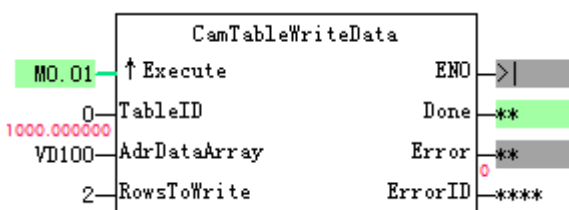


图 7.7.27 读取凸轮表的参数

图 7.7.28 为在“状态符号表”中，分别为调用 CamTableWriteData 指令写入的凸轮表参数和调用 CamTableCreate 指令时写入的凸轮表参数的参数。

VD100	2436	四字节浮点数	1000.000000	1000
VD102	2438	四字节浮点数	2000.000000	2000
VD104	2440	四字节浮点数	3000.000000	3000
VD106	2442	四字节浮点数	4000.000000	4000
VD200	2536	四字节浮点数	1000.000000	
VD202	2538	四字节浮点数	2000.000000	
VD204	2540	四字节浮点数	3000.000000	
VD206	2542	四字节浮点数	4000.000000	

图 7.7.28 凸轮表参数设置

7.7.9 读凸轮表数据 CamTableReadData

● 指令参数

CamTableReadData << ↑ Execute 0 TableID ???? AdrDataArray 2 RowsToRead ENO → Done ** Error ** ErrorID ****			读取凸轮表中的数据。 指令使能后，读取指定凸轮表中的数据，要读取的行数（RowsToRead）必须小于等于凸轮表的行数。		
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
TableID	字变量、常数	WORD	0~255	0	凸轮表 ID
AdrDataArray	双字变量	REAL	非负整数	指定变量区中的值	读取数据存放的起始地址

RowsToRead	字变量、常数	WORD	非负整数	2	读取的行数
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令在运行过程中	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

- 详细说明

通过 CamTableReadData 指令，将凸轮表中的数据读取到对应的寄存器。数据读取成功时，Busy 被置“1”，否则输出对应的错误信息。

- 时序图

图 7.7.29 为该指令的引脚时序图。当凸轮表的数据读出时 Busy 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

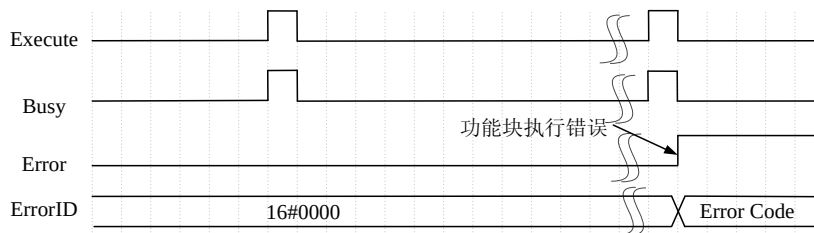


图 7.7.29 指令引脚时序图

- 应用示例

图 7.7.30 中将 M0.1 置为“1”，程序调用 CamTableCreate 指令，创建凸轮表。

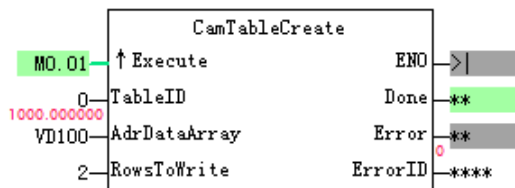


图 7.7.30 凸轮表的创建

图 7.7.31 中将 M0.0 置为“1”，程序调用 CamTableReadData 指令，从凸轮表中读取数据。

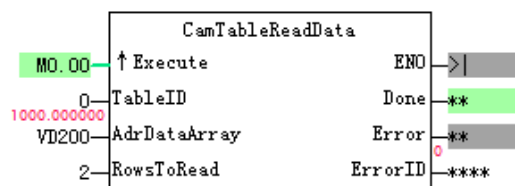


图 7.7.31 读取凸轮表的参数

图 7.7.32 为在“状态符号表”中，调用 CamTableCreate 指令写入的凸轮表数据和调用 CamTableReadData 指令读取的凸轮表数据。

VD100	2436	四字节浮点数	1000.000000	1000
VD102	2438	四字节浮点数	2000.000000	2000
VD104	2440	四字节浮点数	3000.000000	3000
VD106	2442	四字节浮点数	4000.000000	4000
VD200	2536	四字节浮点数	1000.000000	
VD202	2538	四字节浮点数	2000.000000	
VD204	2540	四字节浮点数	3000.000000	
VD206	2542	四字节浮点数	4000.000000	

图 7.7.32 凸轮表中的数据和从凸轮表读

7.7.10 读从轴位置 CamTableReadSlavePos

● 指令参数

CamTableReadSlavePos << ↑ Execute 0 TableID 0 MasterPos ENO → Done ** Error ** ErrorID **** SlavePos **** 根据主轴位置读取从轴的位置。 指令使能后，根据设定的主轴位置，读取当主轴到达指定位置处时，从轴到达的位置，指定的主轴位置必须在凸轮执行的位置中，否则报错，读取值无效。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
TableID	字变量、常数	WORD	0~255	0	凸轮表 ID
MasterPos	双字变量、常数	REAL	实数	0	主轴位置
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令在运行过程中	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	
SlavePos	双字变量	REAL	实数	从轴实时位置	

● 详细说明

通过 CamTableReadSlavePos 指令，根据主轴给出的位置值，读取当主轴运动到该位置时从轴运动到的位置值。读取数据成功时，Busy 被置“1”，否则输出对应的错误信息。

注意：使用该指令时，主轴必须有速度，并且运行该指令时的主从轴速度和位置值，应与实际运行该凸轮表时的主从轴速度和位置值相同。

● 时序图

图 7.7.33 为该指令的引脚时序图。当主轴位置读取从轴位置时 Busy 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

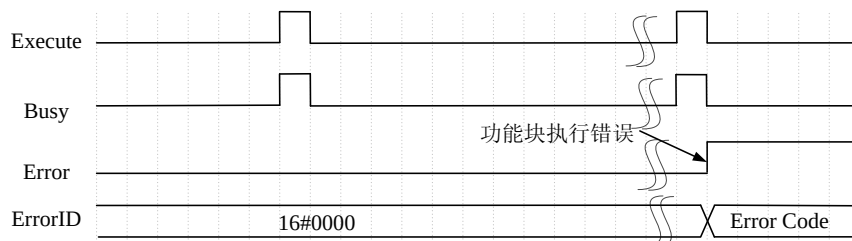


图 7.7.33 指令引脚时序图

● 应用示例

图 7.7.34 中将 M0.1 置为“1”，程序调用 CamTableCreate 指令，创建凸轮表；将 M0.2 置为“1”，程序调用 MC_MoveVelocity 指令，使主轴 0 轴按照一定的速度匀速运动。

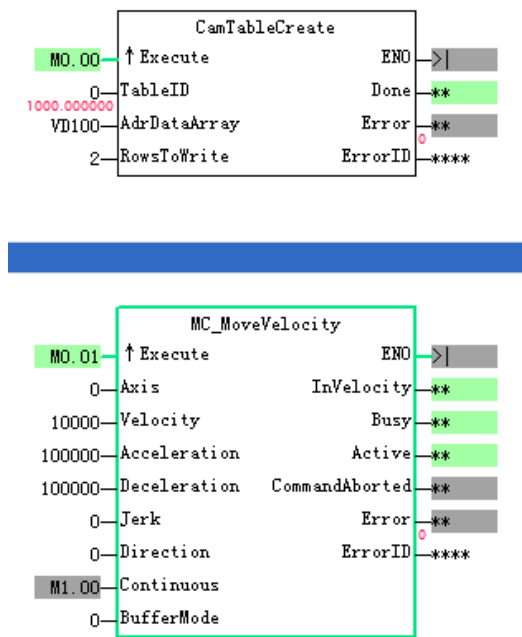


图 7.7.34 创建凸轮表，主轴匀速运动和主轴从轴实时位置的监控

图 7.7.35 中将 M0.3 位置“1”，程序调用 MC_CamIn 指令，使主轴和从轴按照凸轮表中设定的位置运动，同时程序调用 CamTableReadSlavePos 指令，通过主轴的位置计算得出从轴的位置并显示在 SlavePos 引脚上。

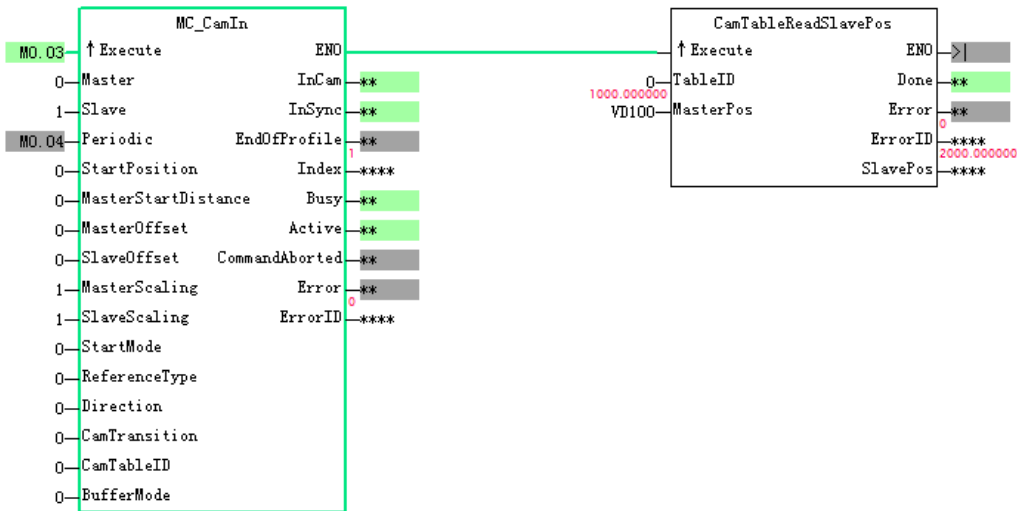


图 7.7.35 凸轮耦合运动

图 7.7.36 为设置的凸轮表参数以及通过 CamTableReadSlavePos 指令读回的从轴的位置。

VD100	2436	四字节浮点数	10000.000000	10000
VD102	2438	四字节浮点数	20000.000000	20000
VD104	2440	四字节浮点数	30000.000000	30000
VD106	2442	四字节浮点数	40000.000000	40000

图 7.7.36 凸轮表参数和读取的从轴位置

7.7.11 多轴同步运行 MC_MultiAxesSyncMove

- 指令参数

			控制多个轴运动到指定的绝对目标位置。 指令使能后，各轴从当前绝对位置运动到 Position 指定的绝对位置；速度以当前速度为起始速度，到达目标位置时速度为 0。 该指令可实现多轴同步启动、同步停止、等时控制与直线插补。 模式 0：多轴同步启动。控制多个轴同步启动，但非同步停止，仅在被其他指令打断时能够同步停止； 模式 1：自适应等时控制（指定时间）。控制多个轴同步启动，经过设定时间后同步停止，加减速控制所需的位置指令自动选取各轴位移最大值，以最优速度控制轴运动。 模式 2：指定主轴等时控制（指定时间）。控制方式与模式 1 相同，加减速控制所需的位置指令为主轴位置。 模式 3：指定主轴等时控制（指定速度）。控制多个轴跟随主轴同步启动，当主轴停止后，其他轴也同步停止。 模式 4：多轴直线插补（指定速度）。控制多个轴进行按照合速度进行直线插补，其中前 3 个轴对应直角坐标系 X-Y-Z 轴，其他均作为旋转轴控制。 模式 5：多轴直线插补（指定时间）。控制方式与模式 4 相同，直线插补采用固定时间梯形曲线加减速控制方式。		
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	FALSE	上升沿触发
MultiAxesEn	字变量、常数	WORD	非负整数	1	多轴 ID 使能，BIT0~BIT15 为“1”对应轴 0~15 使能
Master	字变量、常数	WORD	非负整数	0	主轴 ID。
Position	双字变量、常数	REAL	实数	10000	各轴目标位置
Velocity	双字变量、常数	REAL	实数	10000	各轴运行速度
Acceleration	双字变量、常数	REAL	实数	100000	各轴运行加速度
Deceleration	双字变量、常数	REAL	实数	100000	各轴运行减速度
Jerk	双字变量、常数	REAL	实数	0	各轴运行加加速度
TransitionMode	字变量、常数	WORD	0,1	0	同步运动模式： 模式 0：多轴同步启动 模式 1：自适应等时控制（指定时间） 模式 2：指定主轴等时控制（指定时间） 模式 3：指定主轴等时控制（指定速度） 模式 4：多轴直线插补（指定速度） 模式 5：多轴直线插补（指定时间）
TransitionParameter	双字变量、常数	REAL	正实数	0	模式 0,3,4：保留 模式 1,2,5：数组 0 为同步运行时间，单位 ms；数组 1 为加减速时间常数，单位 ms
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	

Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）
AxesDone	字变量	WORD	非负整数	各轴 Done 信号输出标志，BIT0~BIT15 为“1”对应轴 0~15 运动完成标志

● 详细说明

多轴同步运动控制指令主要完成多个单轴同时启动、同时停止与等时控制，可以通过 TransitionMode 进行选择。

多轴同步启动模式 0：各单轴执行与绝对定位——MC_MoveAbsolute 指令相同，Position 输入各轴指令位置；Velocity 输入各轴运动速度；Acceleration 输入各轴加速度；Deceleration 输入各轴减速度；Jerk 输入各轴加加速度；如图 7.7.37 所示，指令使能后通过 MultiAxesEn 同时启动相应轴。

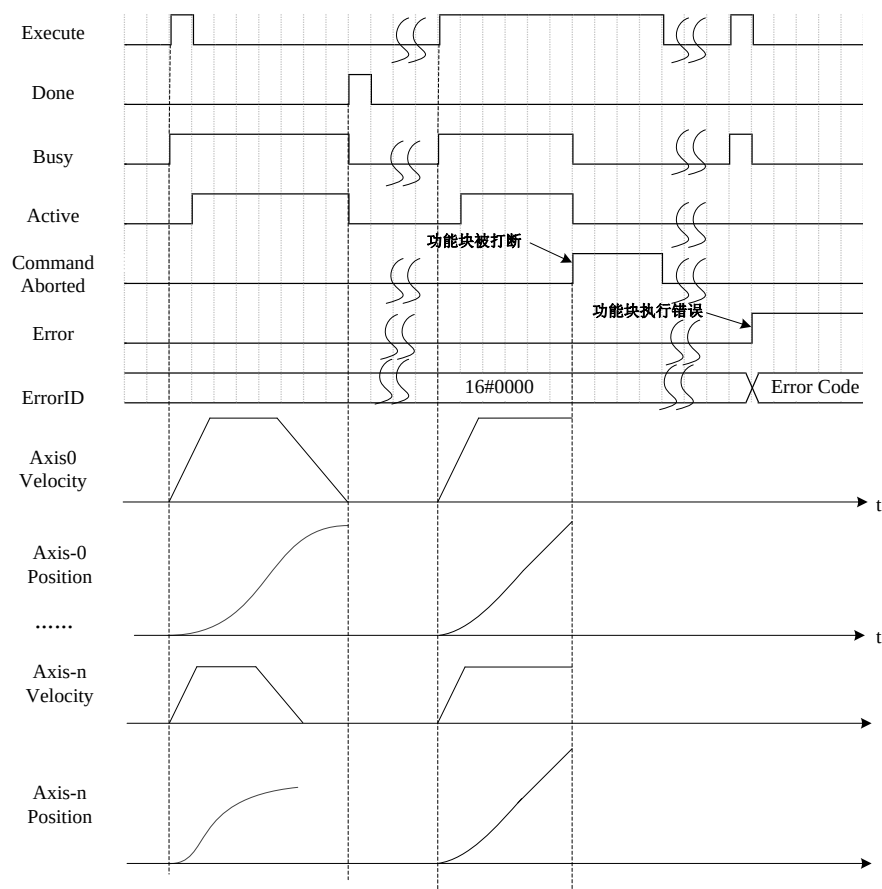


图 7.7.37 多轴同步启动模式时序图

自适应等时控制（指定时间）模式 1：等时控制自动选取位移最大值作为主轴，其他轴按照主从关系同步与主轴运行，Position 输入各轴指令位置；TransitionParameter 输入同步运行时间与加减速时间；指令使能后通过 MultiAxesEn 同时启动相应轴，当经过 TransitionParameter 设定时间后，各轴同时停止，Done 被置“1”。

指定主轴等时控制（指定时间）模式 2：与模式 1 相同，其中主轴可通过输入引脚 Master 设定。

指定主轴等时控制（指定速度）模式 3：与模式 2 相同，速度与加减速通过输入引脚 Velocity、Acceleration 和 Deceleration 设定。

多轴直线插补（指定速度）模式 4：通过多个轴联动，可实现直线插补，如图 7.7.38 所示，Position 输入各轴指令位置，Velocity 为合速度，Acceleration 为加速度，Deceleration 为减速度，Jerk 为加加速度。

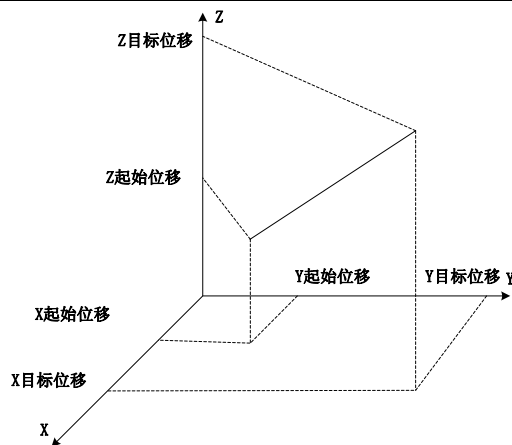


图 7.7.38 多轴直线插补

多轴直线插补（指定时间）模式 5：与模式 4 相同，采用固定时间 T 曲线加减速控制如图 7.7.39 所示。

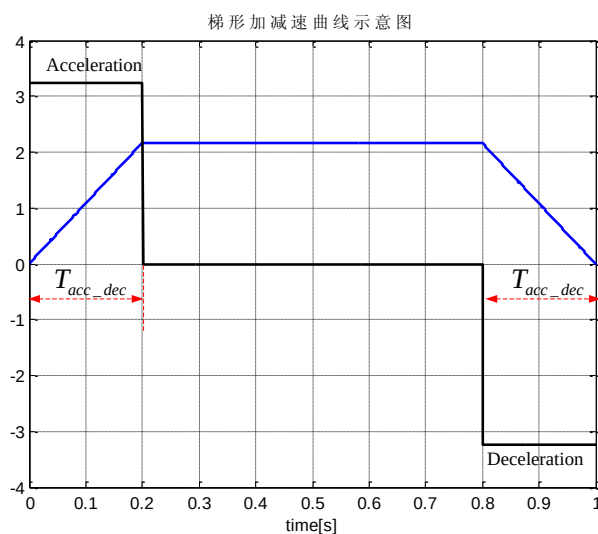


图 7.7.39 等时控制 T 曲线加减速控制

当单轴指令轴 ID 在 MC_MultiAxesSyncMove 中被使能后，MC_MultiAxesSyncMove 指令执行过程中当使能单轴指令（如 MC_Stop、MC_Jog）时，所有轴停止 CommandAborted 被置“1”。

当单轴指令轴 ID 未在 MC_MultiAxesSyncMove 中被使能后，MC_MultiAxesSyncMove 指令执行过程中当使能单轴指令（如 MC_Stop、MC_Jog）时，所有轴继续运行。

注意：

- (1) 模式 0~3, 当 Position、Velocity、Acceleration、Deceleration、Jerk 输入常数时，指定轴 0 位置，其他轴 Position 位置不变、Velocity、Acceleration、Deceleration、Jerk 均为 0；
- (2) 模式 4~5, MultiAxesEn 使能的前三个轴对应直角坐标系 X、Y、Z 轴，其它为旋转轴；
- (3) 模式 4~5, Velocity、Acceleration、Deceleration、Jerk 第 1 个数组指定直线插补的合速度、加速度、减速度和加加速度，其他数组无效。

● 应用示例

多轴同步运行位置、速度、加减速、时间参数设置如表 7.7.1 运动参数所示。

表 7.7.1 运动参数

参数说明	设定值
各轴运动目标位置	(100000,200000,300000,100000,100000,500000,100000,300000)
各轴运动速度	(10000,20000,10000,40000,30000,10000,20000,10000)
各轴加速度	(10000,10000,10000,10000,10000,10000)
各轴减速度	(10000,10000,10000,10000,10000,10000)

模式	0
启停时间	10000
加减速时间	500

如图 7.7.40 所示将运动控制参数轴数量设置为 8，其他为系统默认参数。

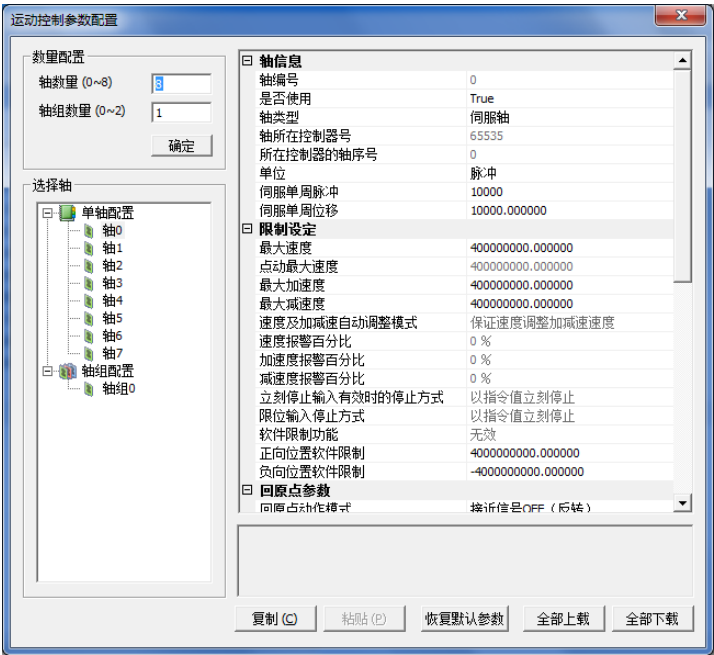


图 7.7.40 多轴同步运行控制参数配置

VD0~VD14 输入各轴指令目标位置，VD100~VD114 输入各轴速度，VD200~VD214 输入各轴加速度，VD300~VD314 输入各轴减速度，如图 7.7.41 所示。

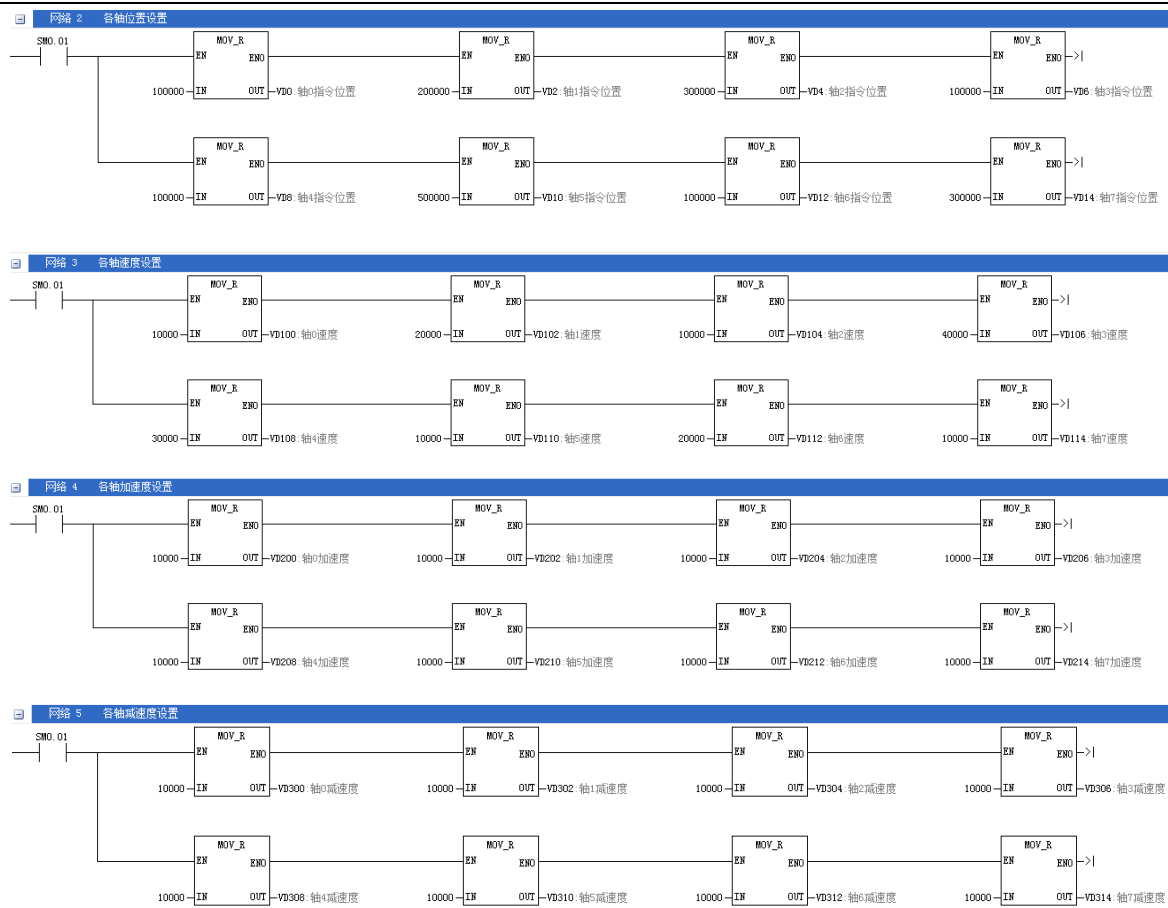


图 7.7.41 多轴运动参数设置

VW400 为模式选择，VD500 为启停时间，VD502 为加减速时间，如图 7.7.42 所示。

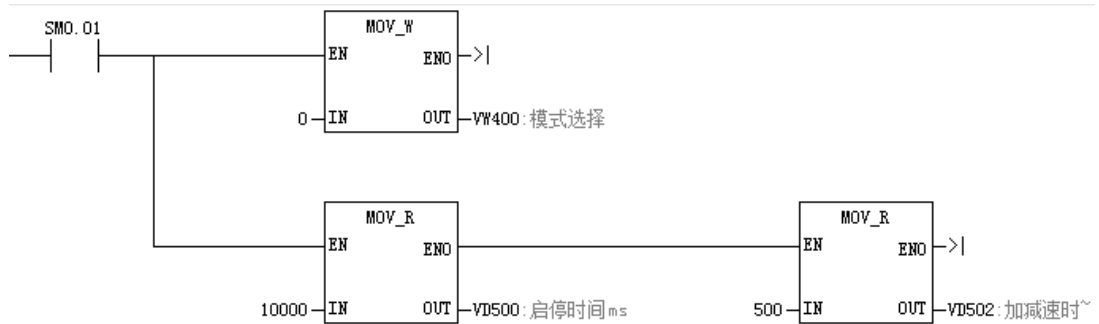


图 7.7.42 模式与时间设置

如图 7.7.43，将 M1.0 置“1”，根据 VW400 设置模式进行多轴同步运行。

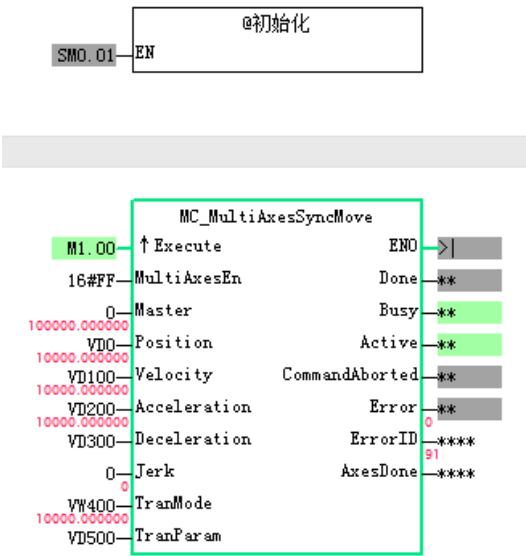


图 7.7.43 多轴同步运行

7.7.12 旋切耦合 MC_APF_RotaryCutIn

● 指令参数

			建立旋切关系 根据实际的应用需求，指定旋切轴和进给轴，以及对应的旋切关系表，当指令执行成功后，旋切轴按照指定的旋切曲线进行运动		
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
RotaryAxis	字变量、常数	WORD	非负整数	0	轴 ID
FeedAxis	字变量、常数	WORD	非负整数	1	轴 ID
RotaryCutID	字变量、常数	WORD	0~7	0	旋切表的 ID，使用同一个编号的旋切指令，运动参数一致。
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

在旋切工艺中，进给轴为主轴，旋切轴为从轴图 7.7.44 所示。主从轴根据辊半径的大小来计算各自运行的距离，进给轴运行的距离与被切割的材料相等。旋切轴根据切割长度，动态调节自己的运行速度（切割时必须保证刀头和被切割材料的线速度一致）。如果只有一把刀，则旋切轴运动一周的时间与被切割材料运动设定长度的时间一致。

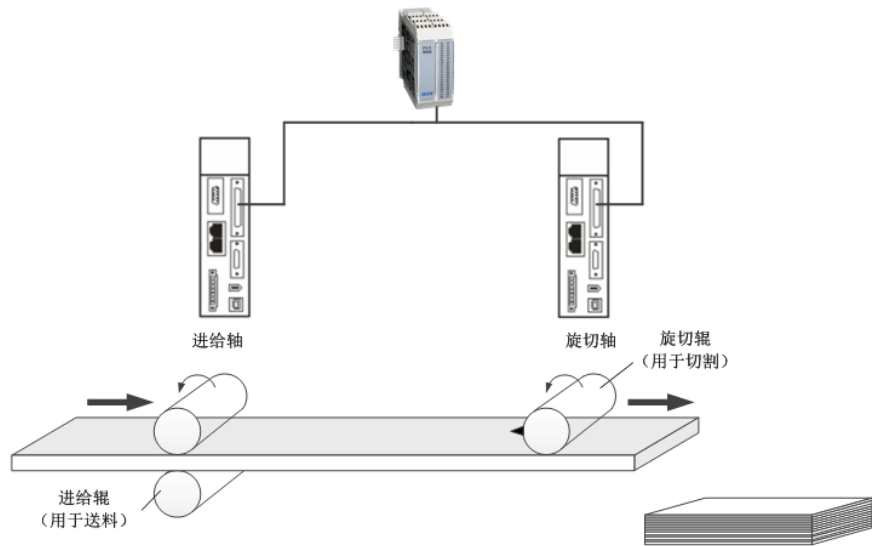


图 7.7.44 旋切工艺图

旋切功能涉及的参数如图 7.7.45 和表 7.7.2 所示：

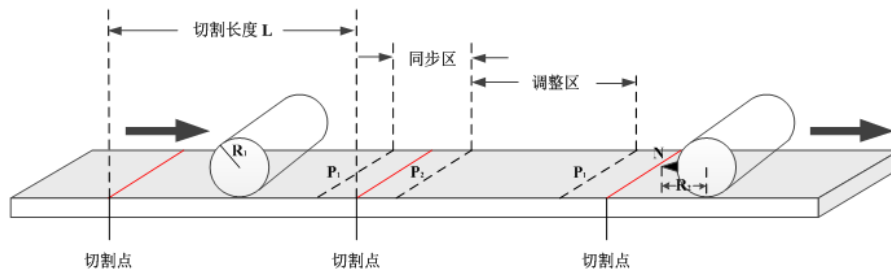


图 7.7.45 旋切工艺参数

表 7.7.2 旋切工艺参数说明

图中参数	实际意义	对应指令中的名称
L	被切割材料的长度	RotaryCurLen
R_1	进给轴的半径，即进给辊的半径长度	FeedAxisRadius
R_2	旋切轴的半径，即圆心到刀尖的距离	RotaryAxisRadius
N	旋切轴的刀头数，图中为 1	RotaryAxisKnifeNum
P_1	同步区的起始位置	FeedAxisSyncStartPos
P_2	同步区的结束位置	FeedAxisSyncStopPos

旋切功能是一种特殊的电子凸轮。有以下特点。

- (1) 用户可根据工艺要求自由设置切割长度，切割长度可以小于等于或大于切刀周长；
- (2) 在同步区内，进给轴与旋切轴（刀头）的线速度相等。在同步区进行切割；
- (3) 旋切可以支持多刀头的旋切辊；
- (4) 旋切功能启动后，旋切轴跟随进给轴的相位动作，进给轴可以做匀速、加速、减速、不规则运动等，旋切轴随动；

(5) 旋切功能结束后，旋切刀停止在用户设置的位置（一般在调整区）。原点位置设置在同步区内。

旋切曲线分为同步区和调整区两段。

同步区：进给轴和旋切轴按照相等的速度运动，即刀头的线速度和切割面的线速度相等。在同步区切割材料。

调整区：为了保证切割的长度和旋切轴的周期性执行，在运行时，需做相应的相位调整。根据切割材料的长度，可以分为下面三种情况。

➤ 短料切割

切割长度小于旋切辊周长（半径为刀头到旋切辊中心的距离）时，任一周期的旋切曲线如图 7.7.46 所示。

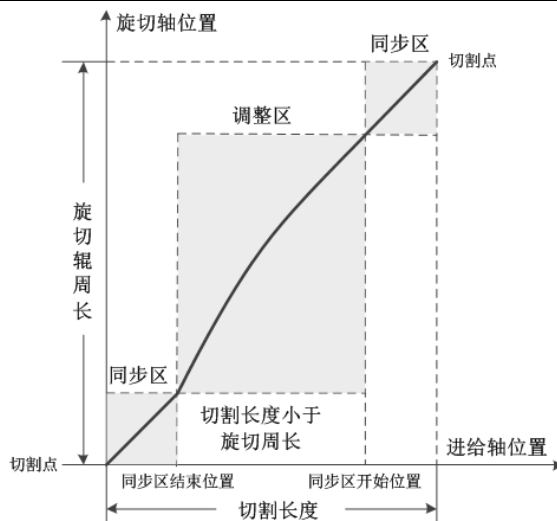


图 7.7.46 短料切割旋切轴运动曲线图

短料切割时，旋切轴必须在调整区内先加速，后减速至同步速度，设计时，调整区无恒速段。

➤ 等长切割

切割长度等于旋切辊周长（半径为刀头到旋切辊中心的距离）时，任一周期的旋切曲线如图 7.7.47 所示。

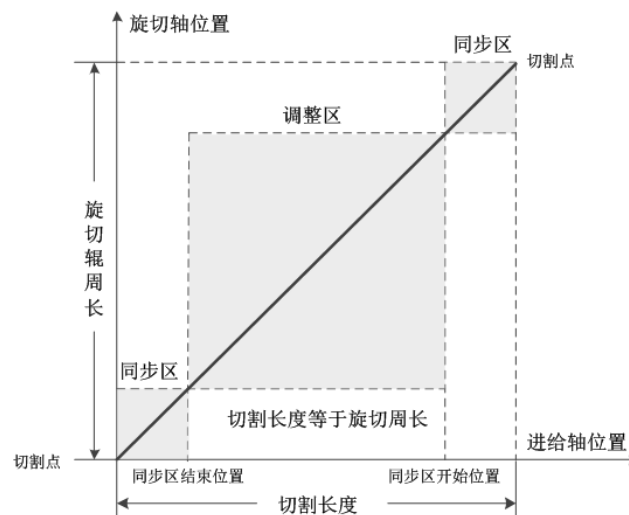


图 7.7.47 等长料切割旋切轴运动曲线图

等长切割时，在同步区与调整区内，进给轴和旋切轴一直保持速度同步，无需进行相位调整。

➤ 长料切割

切割长度大于旋切辊周长（半径为刀头到旋切辊中心的距离）时，任一周期的旋切曲线如图 7.7.48 所示。

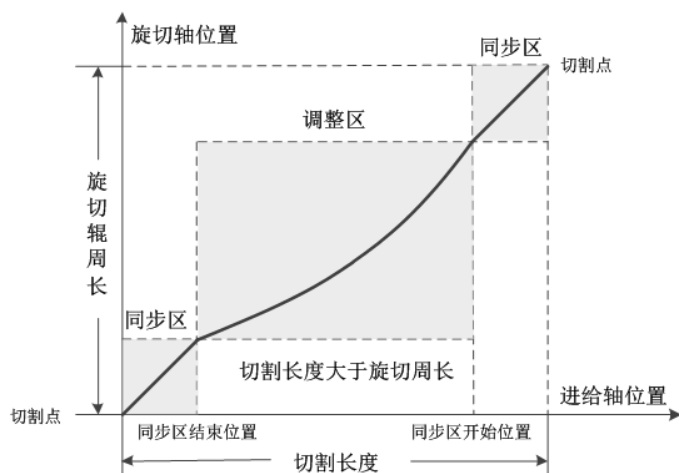


图 7.7.48 长料切割旋切轴运动曲线图

长料切割时，旋切轴必须在调整区内先减速，后加速至同步速度，设计时，调整区无恒速段。如果旋切长度远大于旋切辊周长（刀辊），则有可能减速至零，停留一段时间后，再加速至同步速度，切割越长，停留的时间越长。

➤ 进入曲线

旋切轴进入耦合的位置可以为同步区开始位置，也可以是上一次作业后的停止位置。原点位置为切割点位置，刀尖冲下，与切割材料（水平线）垂直如图 7.7.49 所示。停止位置为与原点位置呈任意角度的位置，这个位置可以通过指令进行设置，停止位置一般设置在调整区。

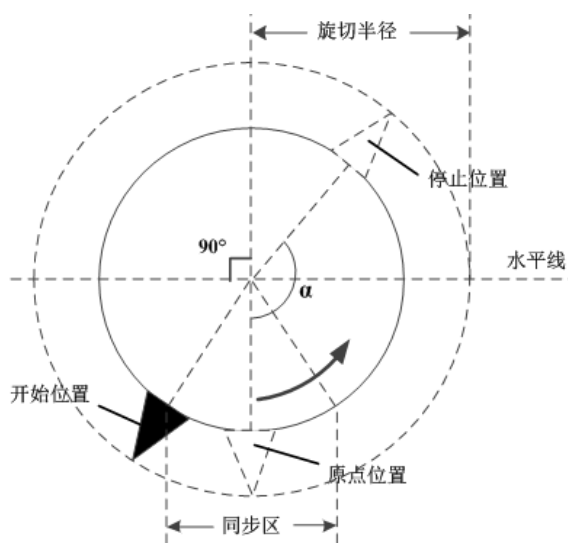


图 7.7.49 旋切轴切割运动曲线图

根据是否回原点，旋切轴有两种进入曲线，一种是回原点结束后，停在同步区开始位置，启动时，直接进入同步区；另一种是在上一次的停止位置启动，先经过一段调整区后，再进入同步区。进入曲线可以从同步区进入，如图 7.7.50 所示，也可以从调整区进入，如图 7.7.510 所示。

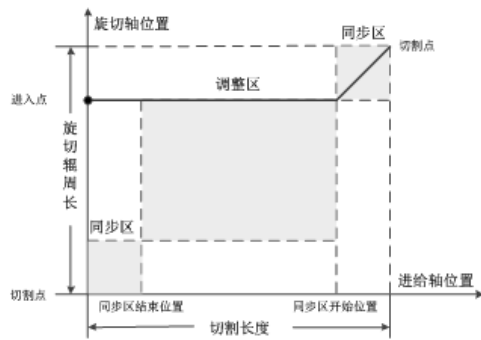


图 7.7.50 旋切轴从同步区进入

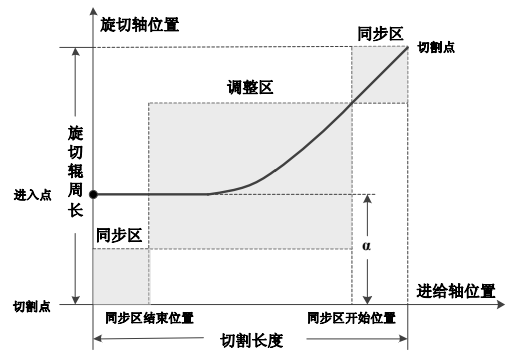


图 7.7.51 旋切轴从调整区进入

安装多刀头时，要求刀头和刀头的间距应该相等。

➤ 退出曲线

旋切轴脱离进给轴运行时，停止在设置的停止位置处，该位置与原点位置呈 α 角度。结束曲线如图 7.7.52 所示。

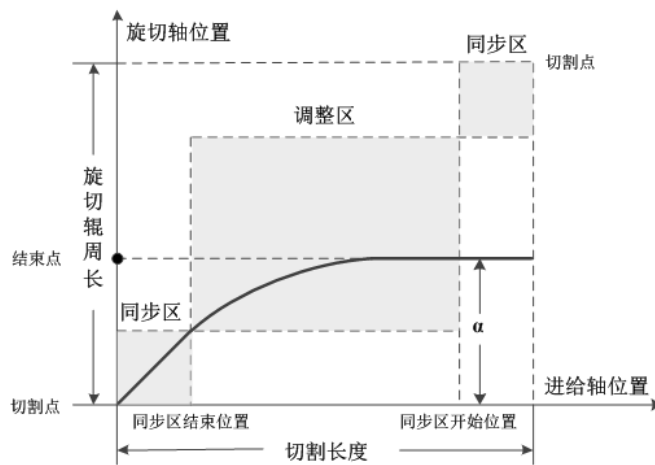


图 7.7.52 旋切轴退出运动曲线图

结束点是针对旋切轴而言的，进给轴仍然保持之前的运动状态。停止后，若不进行其他操作，下次继续进行旋切运动，则该停止点为下一次的进入点。

旋切功能的控制顺序如图 7.7.53 所示。

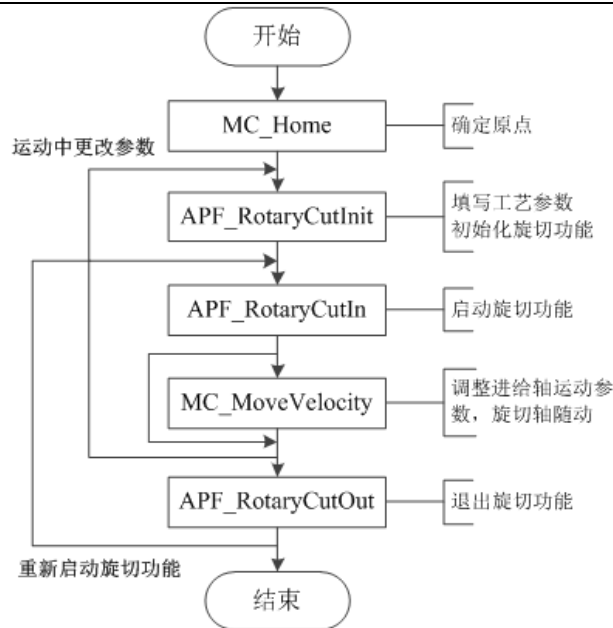


图 7.7.53 旋切控制流程图

旋切功能执行时，旋切轴只能被 APF_RotaryCutOut 和 MC_Stop 指令打断，其它指令无效

● 应用示例

在 PLC_Config 软件运动参数参数配置中将轴 0 和轴 1 系统单位设置为度，伺服单周脉冲设置为 10000，伺服单周位移设置为 360，其他为默认参数，点击全部下载。

旋切工艺参数如表 7.7.3 旋切范例工艺参数所示：

表 7.7.3 旋切范例工艺参数

参数名称	设定值
进给轴 ID	0
旋切轴 ID	1
旋切轴半径	30
旋切轴刀数	1
进给轴半径	30
材料切割长度	200
同步区起始位置	180
同步区结束位置	20
进给轴速度	200

(1) 旋切工艺参数设置

设置材料切割长度为 200，旋切轴半径为 30，旋切轴刀数为 1，进给轴半径为 30，同步区起始位置为 180，同步区结束位置为 20，如图 7.7.54 设置 APF_RotaryCutInit 功能块。M1.00 置“1”后，旋切表 ID=1 创建完成。

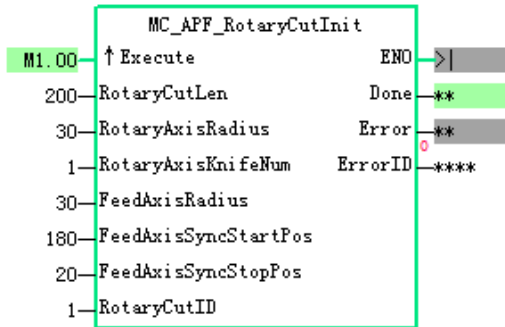


图 7.7.54 旋切工艺参数设置

(2) 建立旋切关系

如图 7.7.55 旋切轴 ID 为 1，进给轴 ID 为 0，旋切表 ID 为 1。M1.01 置“1”后，旋切关系建立完成。

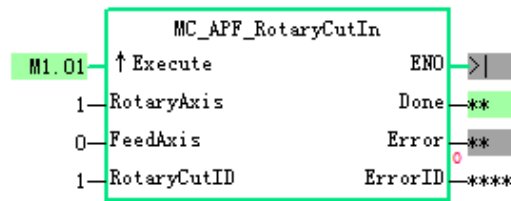


图 7.7.55 建立旋切关系

(3) 进给轴设置

如图 7.7.56 进给轴 ID 为 0，进给速度为 200，加速度为 100，减速度为 100，加加速度为 0，设置恒速运动 MC_MoveVelocity 功能块。M1.02 置“1”后，进给轴开始执行速度指令，此时旋切轴根据进给轴相位来执行旋切运动。

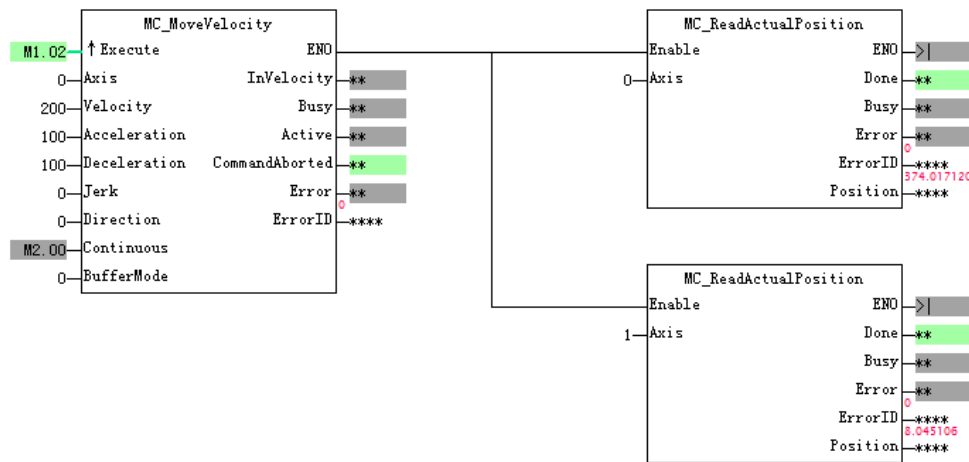


图 7.7.56 旋切轴根据进给轴相位执行旋切运动

(4) 解除旋切关系

如图 7.7.57 旋切轴 ID 为 1，旋切表 ID 为 1，旋切轴停止位置为 150，设置旋切脱离 APF_RotaryCutOut 功能块。M1.03 置“1”后，旋切轴开始脱离进给轴，脱离成功后旋切轴不再跟随进给轴动作，并停止在设定的位置 150 处。

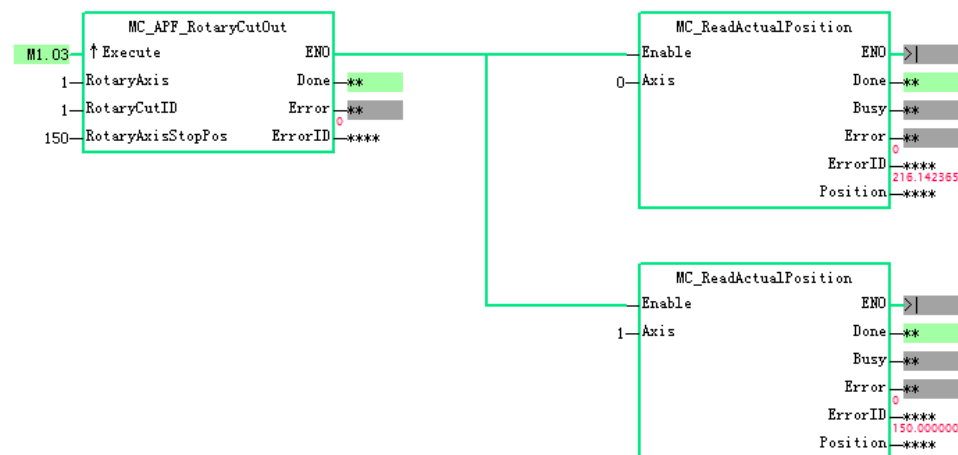


图 7.7.57 旋切关系解除

7.7.13 旋切初始化 MC_APF_RotaryCutInit

- 指令参数

MC_APF_RotaryCutInit << ↑ Execute 0 RotaryCutLen 0 RotaryAxisRadius 0 RotaryAxisKnifeNum 0 FeedAxisRadius 0 FeedAxisSyncStartPos 0 FeedAxisSyncStopPos 0 RotaryCutID ENO → Done ** Error ** ErrorID ****					
旋切参数初始化。 若旋切关系未建立，则初始化对应旋切表的切割长度、旋切轴半径、旋切刀头数，进给轴半径、同步区参数等。指令执行成功后，载入旋切参数。 若旋切关系已经建立，该指令则修改参数，修改参数结束后，新参数在下一个周期生效。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
RotaryCutLen	双字变量、常数	REAL	正数	100	材料的切割长度，单位 units
RotaryAxisRadius	双字变量、常数	REAL	正数	100	旋切轴半径，即旋切轴圆心到刀尖的距离，单位 units
RotaryAxisKnifeNum	字变量、常数	WORD	正数	1	安装在旋切轴上的刀头数
FeedAxisRadius	双字变量、常数	REAL	正数	100	进给轴半径，单位 units
FeedAxisSyncStartPos	双字变量、常数	REAL	正数	578	同步区开始位置，即同步区开始时对应的进给轴的位置，单位 units
FeedAxisSyncStopPos	双字变量、常数	REAL	非负数	50	同步区结束位置，即同步区结束时对应的进给轴的位置，单位 units
RotaryCutID	字变量、常数	WORD	0~7	0	旋切表的 ID，使用同一个编号的旋切指令，运动参数一致。
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

注意：

(1) 旋切轴半径、进给轴半径、切割长度、同步区开始位置、同步区结束位置的单位必须统一。若其中一个单位为毫米，则其他几个的单位也必须为毫米，否则会出错。

(2) 同步区起始位置 FeedAxisSyncStartPos 总大于同步区结束位置 FeedAxisSyncStopPos，如切料长 500，同步区起始位置为 480，同步区结束位置为 20。

(3) 同步区不能大于切割长度的一半。

7.7.14 旋切脱离 MC_APF_RotaryCutOut

● 指令参数

MC_APF_RotaryCutOut << ↑ Execute 0 RotaryAxis 0 RotaryCutID 0 RotaryAxisStopPos ENO → Done ** Error ** ErrorID ****					
解除旋切轴和进给轴之间的旋切关系，旋切轴不再跟随进给轴动作，并停止在设定的位置处 该指令不影响进给轴的运动					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
RotaryAxis	字变量、常数	WORD	非负整数	1	旋切轴 ID
RotaryCutID	字变量、常数	WORD	0~7	0	旋切表 ID，使用同一个编号

					的旋切指令，运动参数一致
RotaryAxisStopPos	双字变量、常数	REAL	非负数	50	旋切轴停止的位置，其值为与原点位置的距离，单位 units，该单位需与选择的旋切表中单位一致
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

7.8 轴组指令

7.8.1 绝对直线运行 MC_MoveLinearAbsolute

● 指令参数

MC_MoveLinearAbsolute << ↑ Execute 双击配置曲线 0 AxesGroup ???? Position 10000 Velocity 100000 Acceleration 100000 Deceleration 0 Jerk 0 CoordSystem 0 TranMode 0 TranParam 0 BufferMode ENO → Done ** Busy ** Active ** CommandAborted ** Error ** ErrorID ****					
控制轴组按照设定的空间直线参数运动到绝对位置。 指令使能后，轴组从当前绝对空间位置运动到 Position 指定的绝对空间位置；速度以当前速度为起始速度，到达目标位置时速度为 0。 该指令支持“梯型曲线”、“指数曲线”和“S 型曲线”加减速控制方式。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
Position	双字变量	REAL	实数	指定变量区中的值	绝对目标位置[数组]，单位 units；数组为 (2~n) 维，n 为轴组中轴个数
Velocity	双字变量、常数	REAL	正数	100000	运行目标速度，单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；若为 0，则无加速过程，单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	100000	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³
CoordSystem	字变量、常数	WORD	--	0	保留
TranMode	字变量、常数	WORD	0,1,11,12,13	0	加减速控制方式： 0：按照配置的运动参数运动，Jerk 为零时，梯形曲线，Jerk 不为零时，S 曲线； 1：指数曲线控制，对应指数加减速时间常数由 TranParam 设置； 11：直线插补时间固定的梯形曲线加减速控制； 12：直线插补时间固定的 S 曲线加减速控制； 13：直线插补时间固定的变加加速度曲线加减速控制；

					模式 11, 12, 13 时, 运动以 TranParam 数组 1 设定的时间动态计算运动参数运动, 运动过程中加减速时间由 TranParam 数组 2 设定
TranParam	双字变量、常数	REAL	非负整数	0	加减速控制参数: TranMode=0: 保留 TranMode=1: 指数加减速时间常数, 单位 ms TranMode=11、12、13: [数组 2] 数组 1: 运动到目标位置需要的时间, 单位:ms: 数组 2: 运动过程加减速时间常数, 单位 ms 数组 3: 缓冲模式过渡圆弧半径
BufferMode	字变量、常数	WORD	非负整数	0	曲线运行模式选择 0: 中止模式, 此时 Done 输出时, 表示曲线运动停止 其他: 缓冲模式, 此时轴会持续运动一段时间后才停止
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时, 表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

通过 MC_MoveLinearAbsolute 指令, 控制指定轴组 (AxesGroup 输入参数) 中的每个轴按着各自设置的目标位置 (Position 输入的各个参数) 运动, 进行空间直线运动, 运动从当前点开始至 Position 设定的位置终止。

TranMode 为加减速控制模式。目前支持梯形曲线加减速控制模式、S 曲线加减速控制模式、指数曲线加减速控制模式、固定时间的梯形曲线加减速控制模式、固定时间的 S 曲线加减速控制模式、固定时间的变加速度加减速控制。

模式 0: 梯形曲线加减速控制, 此模式需要设置运行目标速度 Velocity、加速度 Acceleration 和减速度 Deceleration; TranMode=11 时需要设置 TranParam 数组 1 整个直线插补时间和数组 2 加减速时间。

模式 1: 指数曲线加减速控制, 此模式需要设置运行目标速度 Velocity、TranParam 数组 1 加减速时间, 其中加减速时间为直线从初始速度到达目标速度的 63.2% 时的插补时间。

模式 0: S 曲线加减速控制, 此模式需要设置运行目标速度 Velocity、加速度 Acceleration、减速度 Deceleration 和加加速度 Jerk; TranMode=12 时需要设置 TranParam 数组 1 整个直线插补时间和数组 2 加减速时间。

模式 13: 变加速度曲线加减速控制, 此模式需要设置 TranParam 数组 1 整个直线插补时间和数组 2 加减速时间。

绝对直线运行指令支持缓冲模式, 用来实现多段直线连续运动控制。指令使能后会提前进行前瞻预处理, 规划曲线衔接处轨迹并进行加减速控制, 各段直线采用圆弧 (TranParam 数组 3 需要设定圆弧半径) 衔接以避免速度方向突变引起的电机抖动问题。关于缓冲模式使用请参考 7.5.5 缓冲模式章节。

● 时序图

图 7.8.1 为该指令的引脚时序和速度变化图。Busy 和 Execute 同时被置“1”, Active 在下一周期被置“1”;

当轴组达到目标位置时 Done 被置 “1”；如果有其它的指令打断此指令，CommandAborted 被置 “1”，同时 Busy 和 Active 被置 “0”；如果在指令执行期间有错误发生，则 Error 被置 “1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

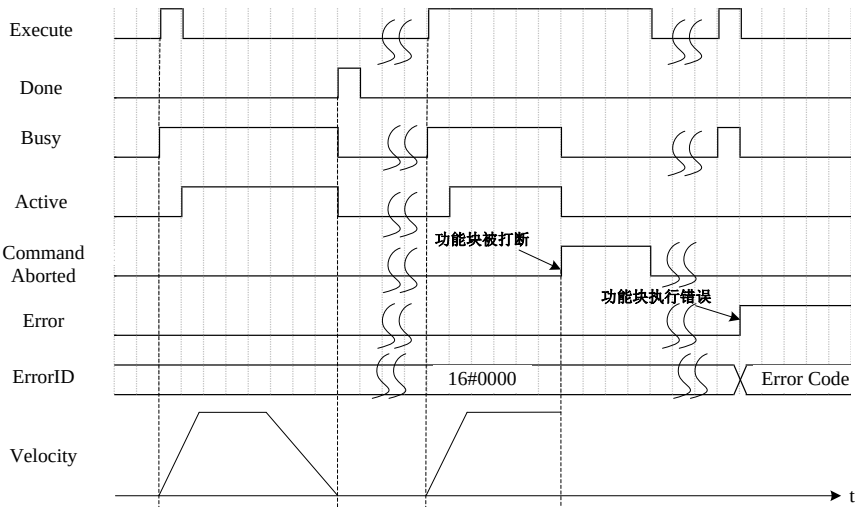


图 7.8.1 指令引脚时序和速度变化图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.2 为“状态符号表”中，轴组的直线运行设定的目标位置参数表。

VD200	2536	四字节浮点数	10000.000000	10000
VD202	2538	四字节浮点数	20000.000000	20000
VD204	2540	四字节浮点数	15000.000000	15000

图 7.8.2 轴组的直线运行位置参数

图 7.8.3 中运行直线插补指令，指令会按照设置的参数值使各个轴运动到指定的位置。图 7.8.4 为轴组运动位移曲线。

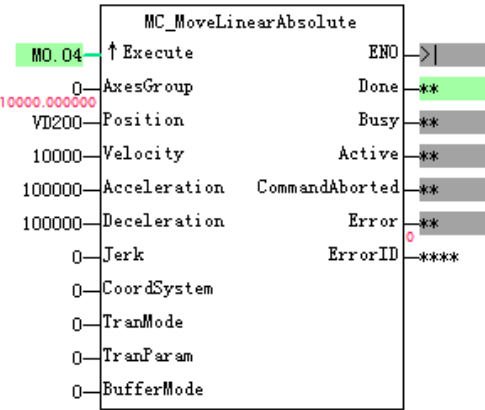


图 7.8.3 绝对直线运行

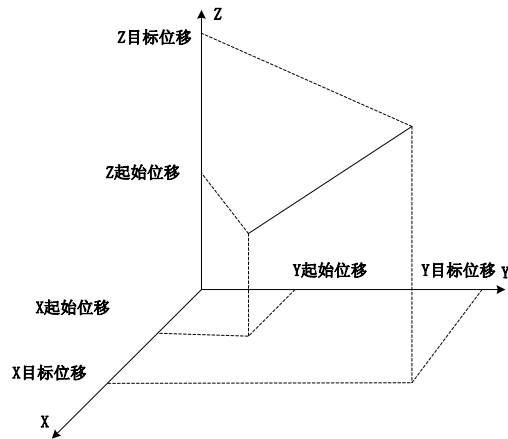


图 7.8.4 位移曲线

7.8.2 相对直线运行 MC_MoveLinearRelative

● 指令参数

MC_MoveLinearRelative << ↑ Execute 双击配置曲线 ENQ → 0 AxesGroup Done ** ???? Distance Busy ** 10000 Velocity Active ** 100000 Acceleration CommandAborted ** 100000 Deceleration Error ** 0 Jerk ErrorID **** 0 CoordSystem 0 TranMode 0 TranParam 0 BufferMode		控制轴组按照设定的空间直线参数运动到相对位置。 指令使能后，轴组的目标位置为轴组的当前绝对空间位置加上 Distance 值；速度以当前速度为起始速度，到达目标位置时轴组速度为 0。 该指令支持“梯型曲线”、“指数曲线”和“S 型曲线”加减速控制方式。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
Distance	双字变量	REAL	实数	指定变量区中的值	相对位置坐标[数组]，单位 units 数组为 (2~n) 维，n 为轴组中轴个数
Velocity	双字变量、常数	REAL	正数	10000	运行目标速度，单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；若为 0，则无加速过程，单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³
CoordSystem	字变量、常数	WORD	--	0	保留
TranMode	字变量、常数	WORD	非负整数	0	加减速控制方式： 0: 按照配置的运动参数运动，Jerk 为零时，梯形曲线，Jerk 不为零时，S 曲线； 1: 指数曲线控制，对应指数加减速时间常数由 TranParam 设置； 11: 直线插补时间固定的梯形曲线加减速控制； 12: 直线插补时间固定的 S 曲线加减速控制；

					13: 直线插补时间固定的变加速度曲线加减速控制; 模式 11, 12, 13 时, 运动以 TranParam 数组 1 设定的时间动态计算运动参数运动, 运动过程中加减速时间由 TranParam 数组 2 设定
TranParam	双字变量、常数	REAL	非负整数	0	加减速控制参数: TranMode=0: 保留 TranMode=1: 指数加减速时间常数, 单位 ms TranMode=11、12、13: [数组 2] 数组 1: 运动到目标位置需要的时间, 单位:ms; 数组 2: 运动过程加减速时间常数, 单位 ms
BufferMode	字变量、常数	WORD	非负整数	0	曲线运行模式选择 0: 中止模式, 此时 Done 输出时, 表示曲线运动停止 其他: 缓冲模式, 此时轴会持续运动一段时间后才停止
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	输出能流	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时, 表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

通过 MC_MoveLinearRelative 指令, 控制指定轴组 (AxesGroup 输入参数) 中的每个轴从当前的位置按照设定的运动参数向相对于当前位置的目标位置进行空间直线运动, 运动从当前点开始, 至 Distance 设定的相对位置终止。

TranMode 为加减速控制模式。使用方法与绝对直线运行相同。目前控制器支持梯形曲线加减速控制模式 (TranMode=0)、指数曲线加减速控制模式 (TranMode=1)、S 曲线加减速控制模式 (TranMode=0)、固定时间的梯形曲线加减速控制模式 (TranMode=11)、固定时间的 S 曲线加减速控制模式 (TranMode=12) 和固定时间的变加速度加减速控制 (TranMode=13)。

相对直线运行指令支持缓冲模式, 用来实现多段直线连续运动控制。指令使能后会提前进行前瞻预处理, 规划曲线衔接处轨迹并进行加减速控制, 各段直线采用圆弧 (TranParam 数组 3 需要设定圆弧半径) 衔接以避免速度方向突变引起的电机抖动问题。关于缓冲模式使用请参考 7.5.5 缓冲模式章节。

● 时序图

图 7.8.5 为该指令的引脚时序和速度变化图。Busy 和 Execute 同时被置“1”, Active 在下一周期被置“1”; 当轴组运动位移达到目标值而且定位完成时 Done 被置“1”; 如果有其它的指令打断此指令, CommandAborted 被置“1”, 同时 Busy 和 Active 被置“0”; 如果在指令执行期间有错误发生, 则 Error 被置“1”, 同时轴将停止, 可以通过 ErrorID(错误码)的值找到导致错误的原因。

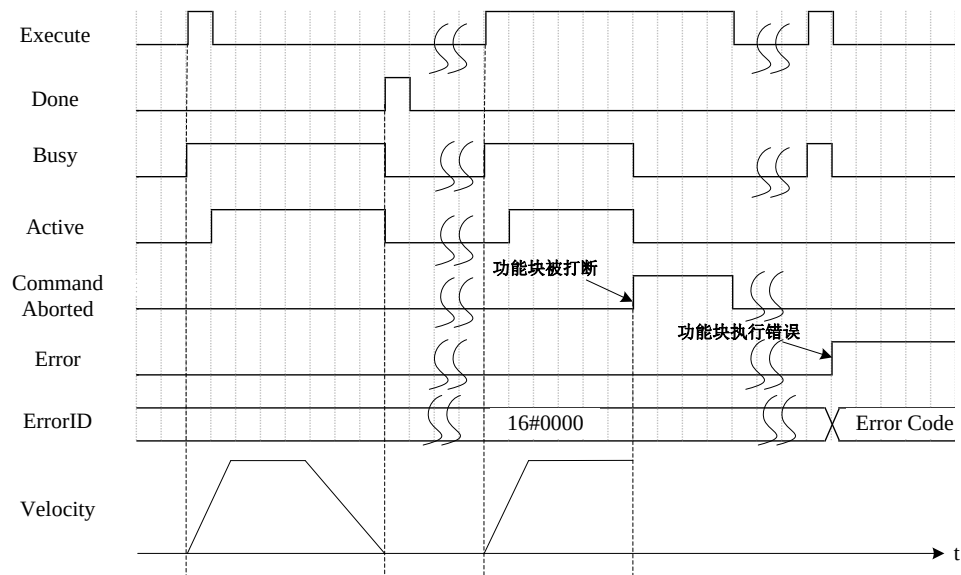


图 7.8.5 指令引脚时序和速度变化图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.6 为“状态符号表”中写入的轴组绝对直线运行的目标位置和相对直线运行的目标位置。

VD200	2536	四字节浮点数	10000.000000	10000
VD202	2538	四字节浮点数	20000.000000	20000
VD204	2540	四字节浮点数	15000.000000	15000
VD300	2636	四字节浮点数	10000.000000	10000
VD302	2638	四字节浮点数	10000.000000	10000
VD304	2640	四字节浮点数	10000.000000	10000

图 7.8.6 目标位置参数表

图 7.8.7 中将 M100.0 置为“1”，程序调用 MC_MoveLinearAbsolute 指令运动到指定的位置，当直线运动到达目标位置后，将 M100.1 置为“1”，程序调用 MC_MoveLinearRelative 指令，轴组开始执行相对运动指令。图 7.8.8 为轴组运动位移曲线。

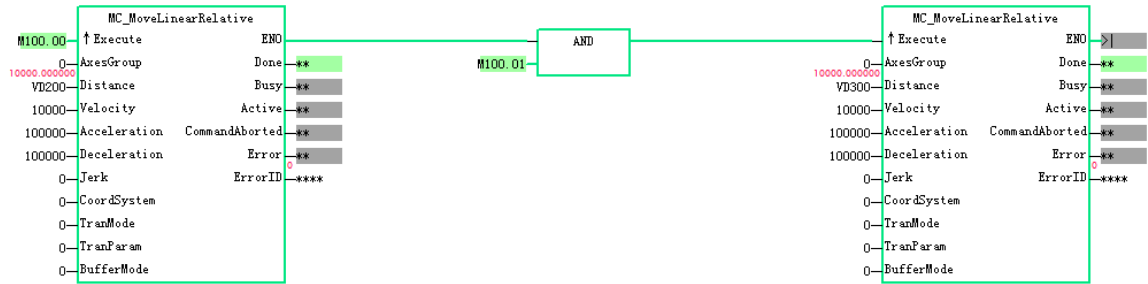


图 7.8.7 直线运动，相对运动和读取轴组位置

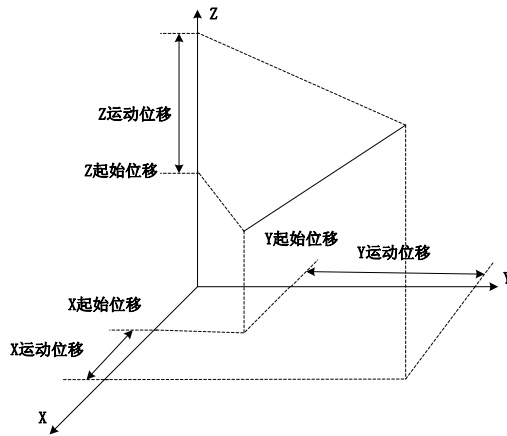


图 7.8.8 位移曲线

7.8.3 绝对样条运行 MC_MoveSplineAbsolute

● 指令参数

<pre> MC_MoveSplineAbsolute << ↑ Execute 双击配置曲线 ENO → 0-AxesGroup Done ** ???? PointsArray Busy ** 0-WeightsArray Active ** 1-ArraySize CommandAborted ** 0-MaxBow Error ** 10000-Velocity ErrorID **** 100000-Acceleration Time **** 100000-Deceleration 0-Jerk 0-CoordSystem 0-TranMode 0-TranParam 0-BufferMode </pre> 控制轴组按照设定的空间曲线参数运动到目标位置。指令使能后，轴组从当前绝对空间位置开始运动，并且按顺序经过空间每个型值点；速度以当前速度为起始速度，到达目标位置时速度为 0。 该指令支持“梯型曲线”和“S 型曲线”加减速控制方式。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
PointsArray	双字变量	REAL	实数	指定变量区中的值	型值点[数组]，每个点包含 X,Y,Z 三个值，如果添加有第四轴则还包含值 A；
WeightsArray	双字变量、常数	REAL	实数	0	权值[数值]，当第一个权值为 0 时，数组中元素均取 0 值，数组长度等于型值点个数加 2
ArraySize	字变量、常数	WORD	非负整数	1	型值点数组的长度
MaxBow	双字变量、常数	REAL	非负数	0	最大的弓高值
Velocity	双字变量、常数	REAL	正数	10000	运行目标速度，单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；若为 0，则无加速过程，单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³
CoordSystem	字变量、常数	WORD	--	0	保留
BufferMode	字变量、常数	WORD	--	0	保留

TranMode	字变量、常数	WORD	非负整数	0	加减速控制方式： 0: 按照配置的运动参数运动, Jerk 为零时, 梯形曲线, Jerk 不为零时, S 曲线; 12: 加加速度为恒定值的 S 曲线。 13: 加加速度变化的 S 曲线 模式 12,13 时, 运动以设定的时间 (TranParam 值), 动态计算运动参数运动, 运动过程中无恒速段
TranParameter	双字变量、常数	REAL	非负整数	0	TranMode 为 12 或 13 时指定运动到目标位置需要的时间, 单位: ms
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为 “1” 时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为 “1” 时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为 “1” 时, 表示轴正在运动	
CommandAborted	位变量	BOOL	0 或 1	为 “1” 时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为 “1” 时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	
Time	双字变量	DWORD	非负整数	指令执行所需时间, 单位 ms	

● 详细说明

通过 MC_MoveSplineAbsolute 指令, 对指定轴组中的各个轴同时进行控制, 每个轴按照指定的各个位置 (型值点的参数, 每点参数包含各个轴的位置) 进行运动, 运动的弓高误差来决定曲线的弯曲度, 最终各个轴会按照各个型值点设置的参数进行空间曲线运动。

● 时序图

图 7.8.9 为该指令的引脚时序和速度变化图。Busy 和 Execute 同时被置 “1”, Active 在下一周期被置 “1”; 当按照空间样条轨迹达到目标值而且定位完成时 Done 被置 “1”; 如果有其它的指令打断此指令, CommandAborted 被置 “1”, 同时 Busy 和 Active 被置 “0”; 如果在指令执行期间有错误发生, 则 Error 被置 “1”, 同时轴将停止, 可以通过 ErrorID(错误码)的值找到导致错误的原因。

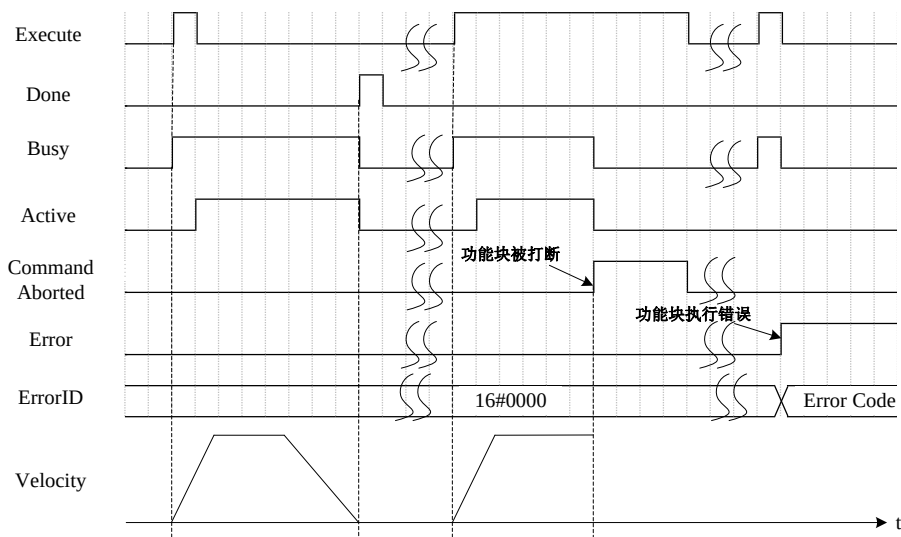


图 7.8.9 指令引脚时序和速度变化图

● 应用示例

在“运动控制参数配置”中, 增加一个轴组 0, 配置使能轴组 0, 并选择三个单轴 (0、1、2) 在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”, 下载参数到设备。图 7.8.10 为在“状态符号表”

中，写入的轴组样条运动的型值点的参数表。

VD200	2536	四字节浮点数	10000.000000	10000
VD202	2538	四字节浮点数	20000.000000	20000
VD204	2540	四字节浮点数	15000.000000	15000
VD206	2542	四字节浮点数	20000.000000	20000
VD208	2544	四字节浮点数	10000.000000	10000
VD210	2546	四字节浮点数	30000.000000	30000
VD212	2548	四字节浮点数	20000.000000	20000
VD214	2550	四字节浮点数	12000.000000	12000
VD216	2552	四字节浮点数	18000.000000	18000
VD218	2554	四字节浮点数	27000.000000	27000
VD220	2556	四字节浮点数	40000.000000	40000
VD222	2558	四字节浮点数	30000.000000	30000

图 7.8.10 轴组样条运动的型值点的参数表

图 7.8.11 中使能 M0.4，程序调用 MC_MoveSplineAbsolute 指令，轴按照设定的型值点开始样条运动。
图 7.8.12 为轴组的运动位移曲线。

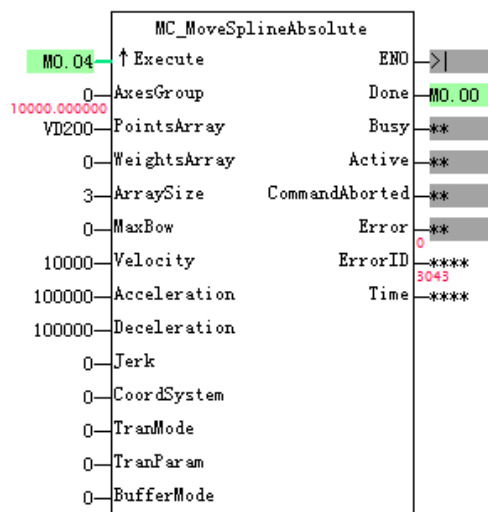


图 7.8.11 绝对样条运行

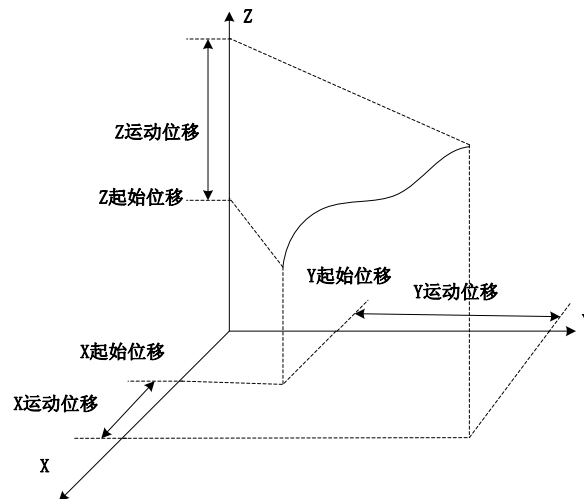


图 7.8.12 位移曲线

在进行样条曲线的轨迹编辑时，可以通过图形化拖拽的方式对样条曲线进行轨迹规划。在样条曲线的编辑状态下，双击指令中的“双击配置曲线”，可弹出 0 的 NURBS 曲线配置界面。在 NURBS 曲线配置界面中，通过设置型值点（曲线经过的点）的值，或者拖拽曲线中的型值点，来规划曲线。0 可以清晰的看出空间轨迹的样式，空间轨迹在三个平面上的投影，以及型值点的坐标等信息。确定轨迹后，可以将轨迹的控制参数下载到指令中，控制轴按照设定的轨迹进行插补运动。

图 7.8.13 中曲线的 4 个型值点的坐标分别为 (100, 100, 100, 0), (118, 150, 188, 107), (256, 240, 218, 283), (314, 216, 180, 360)。

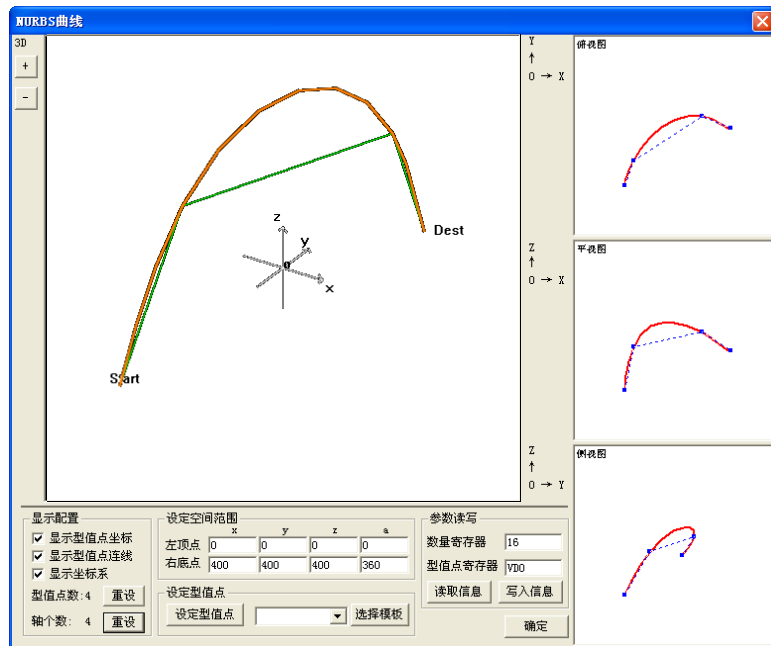


图 7.8.13 绝对样条曲线配置界面

7.8.4 轴组点动运行 MC_GroupJog

● 指令参数

<pre> MC_GroupJog << Enable 双击配置曲线 ENO > 0 AxesGroup Done ** 0 CoordinateUnit Busy ** 0 Direction Active ** 10000 Velocity CommandAborted ** 100000 Acceleration Error ** 100000 Deceleration ErrorID **** 0 BufferMode </pre>		控制轴组按照设定的方向恒速点动执行。 Enable 为“1”时轴组以当前速度为起始速度，达到目标速度后恒速运动。 Enable 为“0”轴组以当前速度为起始速度，减速至停止运动。 Direction: 0 为正方向，1 为负方向。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
CoordinateUnits	字变量、常数	WORD	非负整数	0	坐标单元, 0: X 轴, 2: Y 轴, 4: Z 轴, 8: U 轴, 超出范围坐标单元默认为 X 轴
Direction	字变量、常数	WORD	非负整数	0	0 为正方向, 其他为负方向
Velocity	双字变量、常数	REAL	正数	10000	运行目标速度, 单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度; 若为 0, 则无加速过程, 单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度; 若为 0, 则无减速过程, 单位 units/s ²
BufferMode	字变量、常数	WORD	非负整数	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时, 表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	

Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）

● 详细说明

通过 MC_GroupJog 指令，可以对指定轴组进行点动运行控制，运动的方向由 Coordinateunits 的参数决定：其中 0 为 X 轴方向，2 为 Y 轴方向，8 为 Z 轴方向，缺省值均为 X，direction 决定运动的正向还是负向，最终完成空间点动运动。

● 时序图

图 7.8.14 为该指令的引脚时序和速度变化图。当 Enable 为“1”时 Busy 被置“1”，当 Direction 为 0 时，正向运行，当 Direction 为 1 时，反向运行；在运动过程中，若将 Enable 置为“0”，则轴开始减速运动，当运动停止时，Done 输出一个脉冲。如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

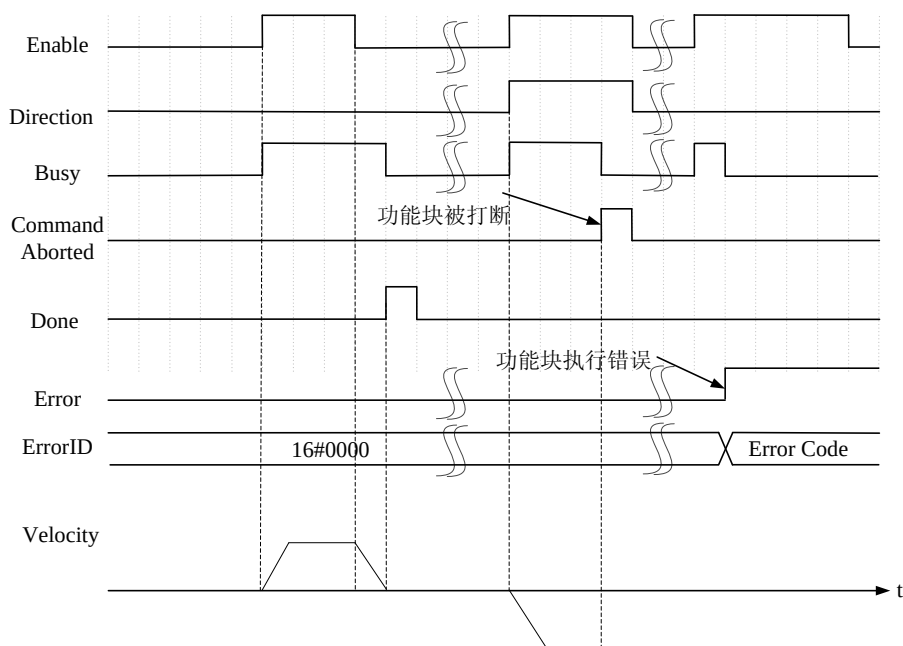


图 7.8.14 指令引脚时序和速度变化图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。

图 7.8.15 中将 M0.3 置为“1”，使能轴组点动指令，将 Coordinateunits 设置为初始值 0，为选择 X 方向点动，将 Direction 置为 0，程序使能 X 正方向点动，可以从实时位置中读取到 X 坐标增大。依次将 CoordinateUnit 设置为 2,4 代表分别选择 Y, Z 方向，置 Direction 为 0 或 1，轴组则会向 Y,Z 的正方向或负方向移动。图 7.8.16 为轴组的运动位移曲线。

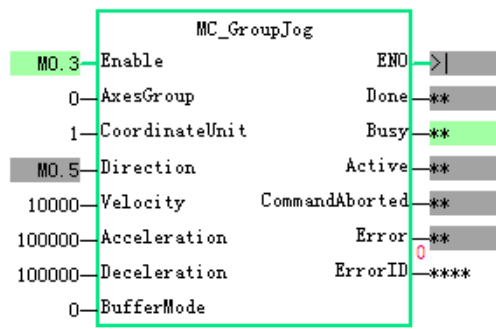


图 7.8.15 点动运行

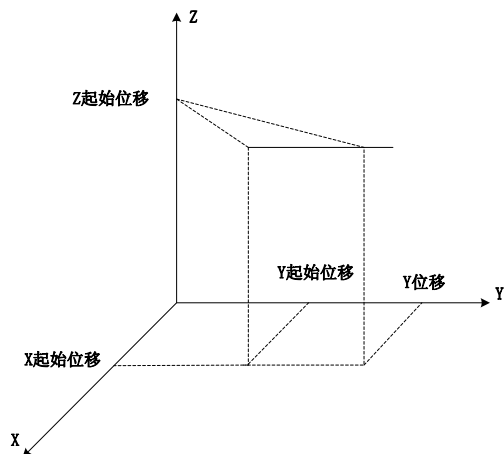


图 7.8.16 位移曲线

7.8.5 轴组写实际位置 MC_GroupWriteActPosition

● 指令参数

MC_GroupWriteActualPosition <<↑Execute 0—AxesGroup ???—Position 0—Mode 0—BufferMode ENO—> Done—** Busy—** Active—** CommandAborted—** Error—** ErrorID—****			修改轴组当前的绝对空间位置。 指令使能后，修改轴组当前绝对空间位置为 Position 指定的值。 轴组报错时，写入的 Position 无效。		
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
Position	双字变量	REAL	实数	指定变量区中的值	绝对目标位置[数组], 单位 units, 数组为 (2~n) 维 n 为轴组中轴个数
Mode	字变量、常数	WORD	--	0	保留
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为 “1” 时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为 “1” 时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为 “1” 时，表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为 “1” 时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为 “1” 时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GroupWriteActPosition 指令，修改轴组当前绝对空间位置为 Position 指定的值。

● 时序图

图 7.8.17 为该指令的引脚时序图。Busy 和 Execute 同时被置“1”；当轴组修改当前绝对空间位置为 Position 时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

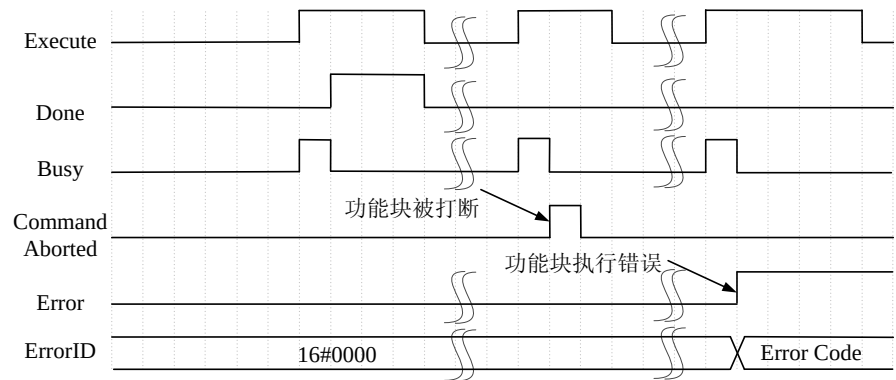


图 7.8.17 指令引脚时序图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.18 中为在“状态符号表”中，写入的轴组目标位置和原点偏移量。

VD200	2536	四字节浮点数	100.000000		100
VD202	2538	四字节浮点数	200.000000		200
VD204	2540	四字节浮点数	150.000000		150
VD300	2636	四字节浮点数	8000.000000		8000
VD302	2638	四字节浮点数	9000.000000		9000
VD304	2640	四字节浮点数	1000.000000		1000

图 7.8.18 轴组目标位置和原点偏移量

图 7.8.19 中将 M0.4 置为“1”，程序将执行 MC_MoveLinearAbsolute 指令，轴组执行运动到目标位置，将 M0.5 置为“1”，程序执行 MC_GroupWriteActPosition 指令，将 Position 引脚的值设置为原点位置。图 7.8.20 为轴组的运动位移曲线。

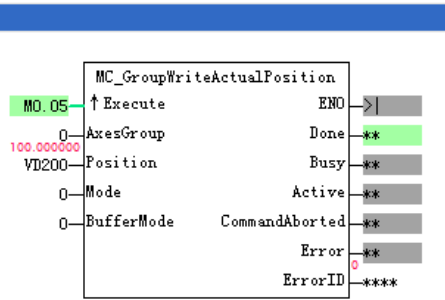
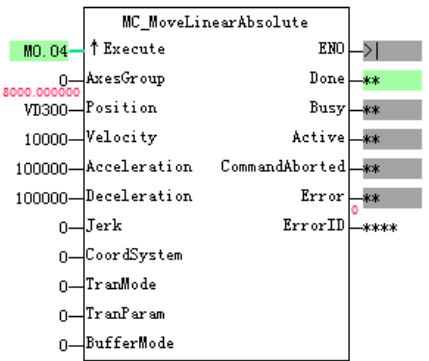


图 7.8.19 直线运动和回原点运行

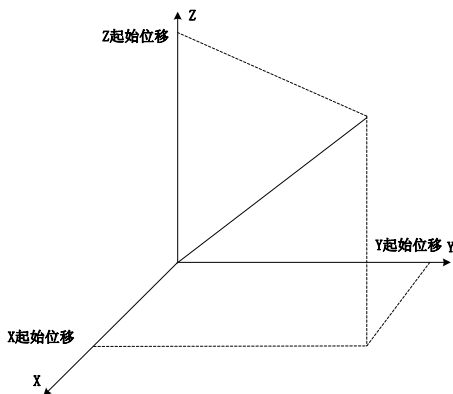


图 7.8.20 位移曲线

7.8.6 轴组恒速运行 MC_GroupMoveVelocity

● 指令参数

MC_GroupMoveVelocity << ↑ Execute 双击配置曲线 ENO → 0 AxesGroup InVelocity ** 0 CoordinateUnit Busy ** 10000 Velocity Active ** 100000 Acceleration CommandAborted ** 100000 Deceleration Error ** 0 Jerk Errorid **** 0 Direction 0 BufferMode		控制轴组按照设定的方向恒速运动。 指令使能后，轴组以当前速度为起始速度，按照设定的参数达到目标速度后恒速运动。 Coordinateunits: 设置轴组按照空间坐标运动，数值 0, 2, 4, 8, 分别为沿 X 轴, Y 轴, Z 轴, U 轴方向运动。 Direction: 设置轴组的运动方向，0 为正向，1 为负向。			
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
Coordinateunits	字变量、常数	WORD	非负整数	0	坐标单元，0: X 轴，2: Y 轴，4: Z 轴，8: A 轴，超出范围默认为 X 轴
Velocity	双字变量、常数	REAL	正数	10000	运行目标速度，单位 units/s

Acceleration	双字变量、常数	REAL	非负数	100000	加速度；若为 0，则无加速过程，单位 units/s^2
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s^2
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s^3
Direction	字变量、常数	WORD	非负整数	0	0 为正方向，其他为负方向
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
InVelocity	位变量	BOOL	0 或 1	为“1”时，表示达到目标速度	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GroupMoveVelocity 指令，实现对指定的轴组进行空间恒速运行控制；其中 Coordinateunits 为坐标轴选择，0 为 X 轴方向，2 为 Y 轴方向，4 为 Z 轴方向；Direction 引脚为方向选择，为 0 时轴组按相应坐标轴正向运行，为 1 时轴组按坐标轴负向运行。

● 时序图

图 7.8.21 为该指令的引脚时序和速度变化图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当速度到达目标值时 InVelocity 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

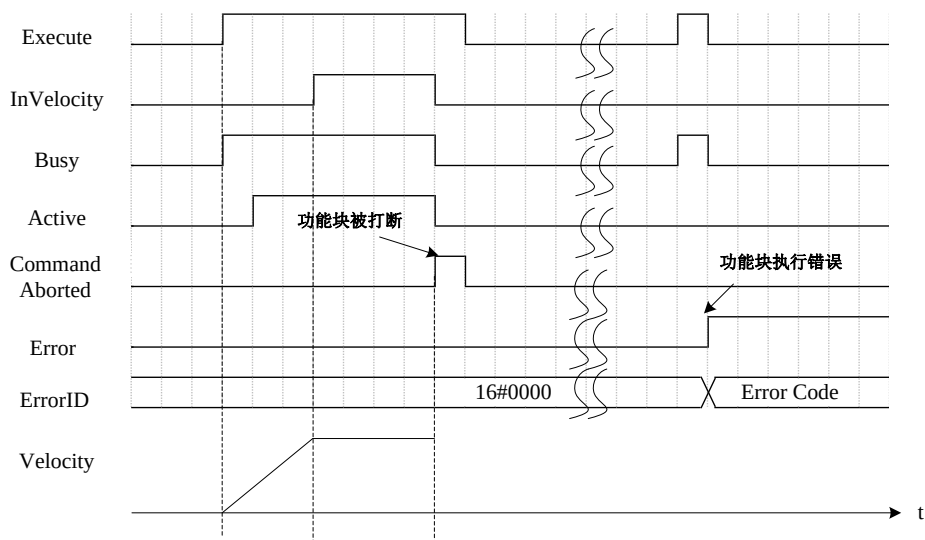


图 7.8.21 指令引脚时序和速度变化图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.22 中将 M0.3 置为“1”，程序执行 MC_GroupMoveVelocity 指令，将 Coordinateunits 置为初始值 0，表示 X 轴方向的点动。

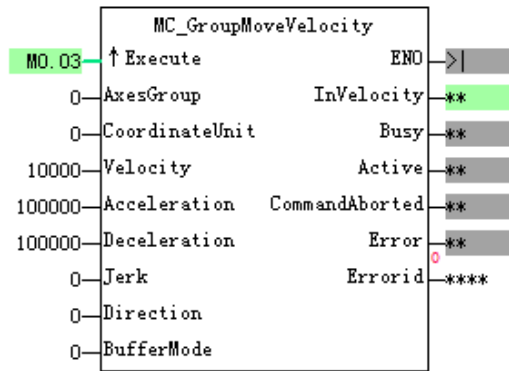


图 7.8.22 匀速运动

7.8.7 停止轴组 MC_GroupStop

● 指令参数

停止轴组运动。 Deceleration 为零时，轴组立即停止运动，非零时，轴组减速运动至停止。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	减减速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GroupStop 指令，停止轴组空间曲线的运行。

● 时序图

图 7.8.23 为该指令的引脚时序图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当轴组上的运动停止时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

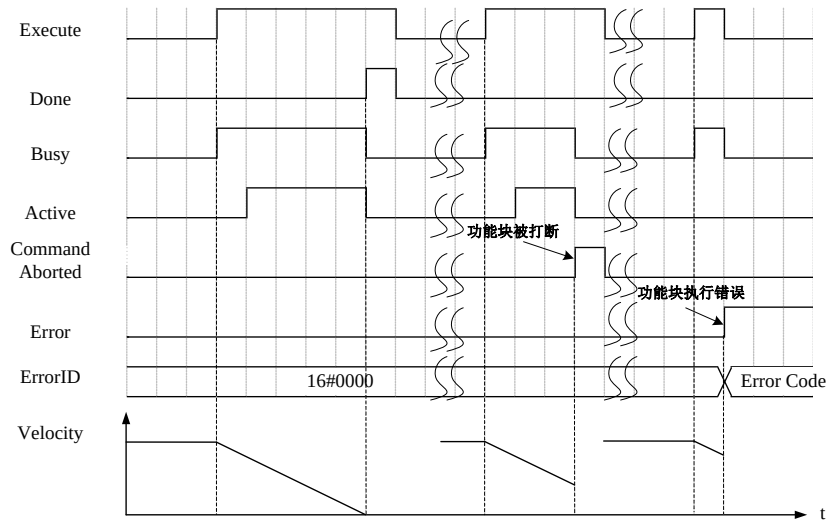


图 7.8.23 指令引脚时序和速度变化图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.24 为“状态符号表”中，写入的轴组运动的目标位置。

VD200	2536	四字节浮点数	100000.000000	100000
VD202	2538	四字节浮点数	200000.000000	200000
VD204	2540	四字节浮点数	150000.000000	150000

图 7.8.24 轴组运动的目标位置

图 7.8.25 中将 M0.3 置为“1”，程序调用 MC_MoveLinearAbsolute 指令，轴组开始直线运动，此时将 M0.4 置为“1”，程序调用 MC_GroupStop 指令，轴组开始减速停止运动。

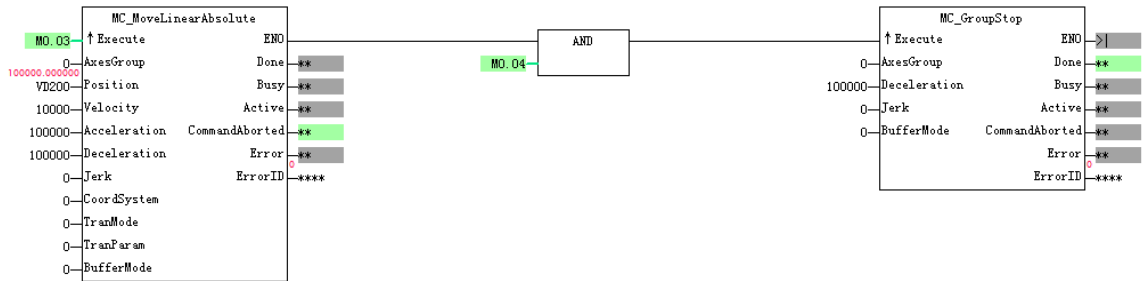


图 7.8.25 直线运动

7.8.8 平面圆弧运行 MC_MoveCircular2D

● 指令参数

MC_MoveCircular2D << ↑ Execute 双击配置曲线 ENO > 0—AxesGroup Done—** 0—CircMode Busy—** ????—AuxPoint Active—** ????—EndPoint CommandAborted—** 0—PathChoice Error—** 10000—Velocity ErrorID—**** 100000—Acceleration 100000—Deceleration 0—Jerk 0—TransitionMode 0—TransitionParameter 0—BufferMode 控制轴组进行平面圆弧插补运动。 指令使能后，控制指定轴组中的每个轴从当前的位置按照设定的运动参数，沿圆弧轨迹向目标位置插补运动，运动从当前点开始，至 EndPoint 设定的位置终止；速度以当前速度为起始速度，到达目标位置后轴组的速度为 0。					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
CircMode	字变量、常数	WORD	0~2	0	圆弧模式 0: 圆心模式 1: 边界点模式 2: 半径模式
AuxPoint	双字变量	REAL	实数	指定变量区中的值	数组: 圆心模式 0: 圆心坐标 边界点模式 1: 圆弧上任意一点坐标 半径模式 2: 第一个实数表示半径长度, 正值运行劣弧, 负值运行优弧;
EndPoint	双字变量	REAL	实数	指定变量区中的值	圆弧终点坐标
PathChoice	字变量、常数	WORD	0 或 1	0	圆弧插补方向选择: 0: 顺时针, 1: 逆时针
Velocity	双字变量、常数	REAL	正实数	10000	运行目标速度, 单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度; 若为 0, 则无加速过程, 单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度; 若为 0, 则无减速过程, 单位 units/s ²
Jerk	双字变量、常数	REAL	--	0	保留
TransitionMode	字变量、常数	WORD	0,1 或 2	0	插补平面选择: 0: XY 平面, 1: ZX 平面, 2: YZ 平面
TransitionParameter	双字变量、常数	REAL	0 或 1	0	插补方式选择: 0: 绝对圆弧, 1: 相对圆弧
BufferMode	字变量、常数	WORD	非负整数	0	曲线运行模式选择 0: 中止模式, 此时 Done 输出时, 表示曲线运动停止 其他: 缓冲模式, 此时轴会持续运动一段时间后才停止
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时, 表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	

Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）

● 详细说明

通过 MC_MoveCircular2D 指令，控制指定轴组（AxesGroup 输入参数）中的每个轴从当前的位置按照设定的运动参数向相对于当前位置的目标位置进行平面圆弧插补运动，运动从当前点开始，至 EndPoint 设定的相对位置终止。

CircMode 为圆弧插补的模式选择，0 为圆心模式，1 为边界点模式，2 为半径模式。

圆心模式：在圆心模式下 AuxPoint 存放的三维数组为运行的坐标，控制轴组以当前点为起点，以 EndPoint 为终点，以 AuxPoint 为圆心进行圆弧插补。在起点和终点重合情况下，会以 AuxPoint 为圆心进行整圆插补。

边界点模式：在边界点模式下 AuxPoint 存放的三维数组为运行的圆弧轨迹上的一点，控制轴组以当前点为起点，以 EndPoint 为终点，并以 AuxPoint 为运动轨迹上的一点进行圆弧插补。在起点、终点和边界点共线情况下，会按照直线插补方式从起点运动到终点。边界点模式无法加工整圆。

半径模式：在半径模式下 AuxPoint 存放的第一个数组元素为圆弧插补的半径（不为 0），其他数组元素保留，以当前点为起点，以 EndPoint 为终点，以 AuxPoint 中第一个数组元素为半径进行圆弧插补，其正值为劣弧插补，负值为优弧插补，半圆插补情况下，不区分正负。半径模式无法加工整圆。

PathChoice 为圆弧插补方向选择，即是按照顺时针方向或是逆时针方向进行圆弧插补。0 为顺时针，1 为逆时针

TransitionMode 为圆弧插补平面选择，即是在 XYZ 任意两轴平面进行圆弧插补。0 为 XY 平面，1 为 ZX 平面，2 为 YZ 平面。

TransitionParameter 为插补方式选择，即是按照绝对位置进行插补或是按照相对位置进行插补。0 为绝对位置插补，1 为相对位置插补。

平面圆弧运行指令支持缓冲模式，用来实现多段圆弧连续运动控制。指令使能后会提前进行前瞻预处理，规划曲线衔接处轨迹并进行加减速控制，各段圆弧采用圆弧（TranParam 数组 3 需要设定圆弧半径）衔接以避免速度方向突变引起的电机抖动问题。关于缓冲模式使用请参考 7.5.5 缓冲模式章节。

● 时序图

图 7.8.26 为该指令的引脚时序图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当平面圆弧插补运动完成时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

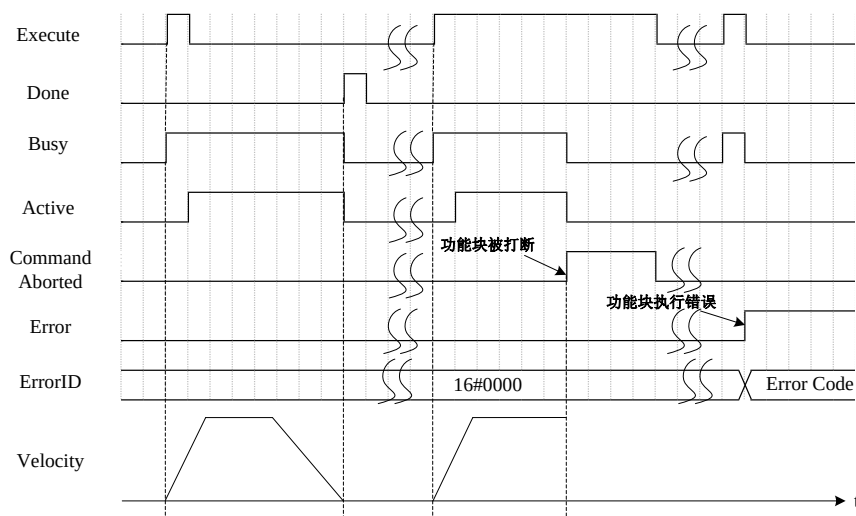


图 7.8.26 指令引脚时序和速度变化图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.27 为在“状态符号表”中，设置的运动轨迹点的目标位置。

VD100	2436	四字节浮点数	2000.000000	2000
VD102	2438	四字节浮点数	3000.000000	3000
VD104	2440	四字节浮点数	4000.000000	4000
VD106	2442	四字节浮点数	6000.000000	6000

图 7.8.27 运动轨迹点的目标位置

图 7.8.28 中将 M100.1 置为“1”，程序调用 MC_MoveCircular2D 指令，进行圆弧插补。插补模式选择 1，为边界点模式，即按照当前点（0,0）作为运动起点，以 EndPoint 作为运动的终点，以 AuxPoint 作为运动轨迹中的一点进行圆弧插补。图 7.8.29 为轴组运动位移曲线。

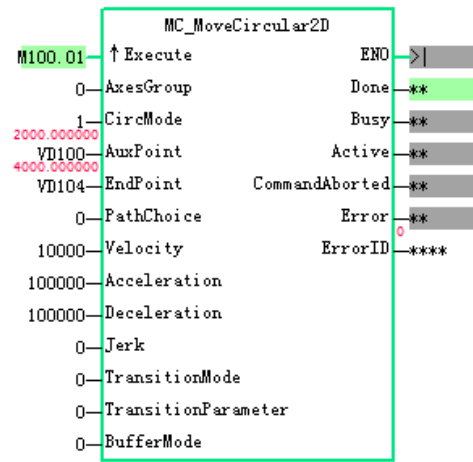


图 7.8.28 圆弧插补运行

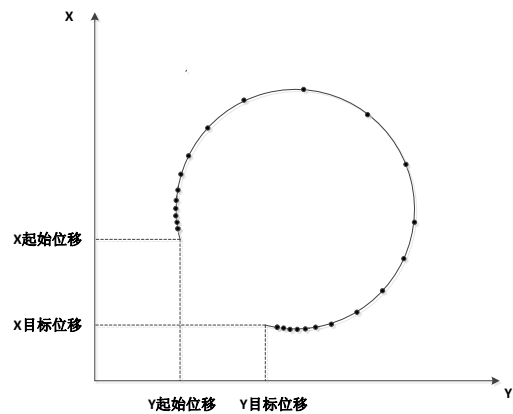
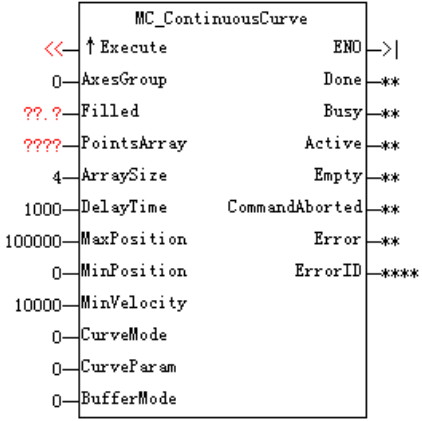


图 7.8.29 位移曲线

7.8.9 连续曲线运行 MC_ContinuousCurve

● 指令参数

			CurveMode 模式: 模式 0: 控制轴组按照指定的连续时间和位置连续运动。指令使能后, 轴组中的实轴从当前绝对位置开始运动, 按照设定的运动时间和位置自动拟合连续曲线运动; 速度以当前速度为起始速度, 到达目标位置时速度为 0。为确保曲线连续运动, 需不断补充新的数据, 当指令可填入新数据时, Empty 为“1”, 此时, 用户可通过装载新的运动时间和位置, 置 Filled 为“1”, 将信息送入指令, 指令获取新信息后, 继续运动。 模式 4、5: 控制轴组按照连续空间直线运动, 相邻两直线衔接处采用空间圆弧过渡。指令使能后, 轴组以当前位置开始运动, 按照设定的绝对空间直线目标位置, 以空间直线和空间圆弧连续运动; 速度以当前速度为起始速度, 到达目标位置时速度为 0。 模式 8: 连续直线圆弧运动, 控制轴组按照指定的连续位置和插补方式连续运动。指令使能后, 轴组中的实轴从当前绝对位置开始运动, 按直线或圆弧连续运动到目标位置, 各段曲线连接处可以采用直线过渡或圆弧过渡。目前圆弧插补方式仅支持 XY 平面圆弧插补。		
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
Filled	位变量	BOOL	0 或 1	指定变量区中的值	模式 0: 读取新运动轨迹控制标志 其他模式: 保留
PointsArray	双字变量	REAL	实数	指定变量区中的值	模式 0: 型值点[数组], 每个点包含两个值: 目标位置 S 和到达目标位置的时间 T 模式 4、5: 空间直线绝对目标位置[数组], 单位 units 模式 8: 绝对目标位置[数组], 单位 units
ArraySize	字变量、常数	WORD	非负整数	2~200	模式 0: 型值点数组的长度 模式 4、5: 空间直线总段数 模式 8: 直线与圆弧总段数
DelayTime	字变量、常数	WORD	非负整数	1000	模式 0: 当前曲线运动完成后至写入新运动轨迹参数所需延时, 单位 ms 模式 4、5、8: 曲线插补完成 Done 信号输出前的延时时间, 单位 ms
MaxPosition	双字变量、常数	REAL	实数	100000	模式 0: 运动过程中的最大位置, 必须大于最小位置, 单位 units 模式 4、5: 保留 模式 8: 半径值, 正为劣弧加工, 负为优弧加工, 0 采用 MinPosition 指定圆心坐标进行圆弧加工
MinPosition	双字变量、常数	REAL	实数	0	模式 0: 运动过程中的最小位置, 必须小于最大位置, 单位 units 模式 4、5: 保留 模式 8: 圆心增量坐标[数组], 单位 units

MinVelocity	双字变量、常数	REAL	非负整数	10000	模式 0: 运动过程中的最小速度 units/s 模式 4、8: 曲线运行目标速度, 单位 units/s 模式 5: 保留
CurveMode	字变量、常数	WORD	0, 4, 5	0	曲线模式: 0: 位置-时间曲线运动模式 4、5: 连续空间直线运动模式, 其中模式 4 为给定目标速度进行连续空间直线运动; 模式 5 为给定整个曲线插补时间进行连续空间直线运动 8: 连续直线圆弧运动模式
CurveParam	双字变量、常数	REAL	正实数	0	[数组 n] 模式 0: 保留 模式 4、5、8: 数组 1: 梯形曲线加减速时间常数, 单位[ms] 数组 2: 整个曲线插补时间, 单位[ms] 数组 3~n: 两段曲线连接处空间圆弧半径, 单位 units, 若为 0 则采用直线过渡
BufferMode	字变量、常数	WORD	0	0	模式 0: 保留 模式 4、5: 空间半径半径输入模式; 0: 每段圆弧半径必须指定; 1: 圆弧半径为 0 时, 自动选择前一个非 0 圆弧半径 模式 8: 数组 0: 0—每段圆弧半径必须指定; 1—圆弧半径为 0 时, 自动选择前一个非 0 圆弧半径 数组 2: 保留 数组 3~n: 曲线各段插补类型, 0 为直线, 1 为顺圆, 2 为逆圆
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时, 表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时, 表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时, 表示轴正在运动	
Empty	位变量	BOOL	0 或 1	为“1”时, 表示可以写入新的运动参数	
CommandAborted	位变量	BOOL	0 或 1	为“1”时, 表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时, 表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码 (错误码见附录 E)	

● 详细说明

目前 MC_ContinuousCurve 指令支持位置-时间曲线运动模式、连续空间直线运动模式和连续直线圆弧运动模式。位置-时间曲线运动模式为两个轴组成的轴组, 虚轴和实轴, 虚轴为时间轴, 实轴为控制运动的实际存在的单轴; 连续空间直线运动模式控制轴组按照连续空间直线运动, 相邻两直线衔接处采用空间圆弧过渡。连续直线圆弧运动模式控制轴组按照连续直线或圆弧运动, 相邻两段曲线衔接处可以采用圆弧过渡。

(1) 位置-时间曲线运动模式

通过 MC_MoveContinuousCurve 指令, 控制指定轴组 (AxesGroup 输入参数) 中的实轴从当前的位置按照设定的运动时间和位置自动拟合连续曲线运动。当指令判断可以写入新的运动轨迹参数时会将 Empty 置为

“1”，此时允许用户填入新的运动时间和位置。将 Filled 置为“1”，指令读入新的运动轨迹参数，之后控制实轴按照设定的运动时间和位置运动。

Filled 读取新运动轨迹控制标志，当用户将新的运动时间和位置写入到 PointArrays 变量区后，须将该引脚置为“1”，指令才会读取新设定的时间和位置运动。

DelayTime 为允许的延迟时间，在当前的运动完成时开始计时，用户在 DelayTime 的时间范围内，需将新的运动轨迹参数写入到 PointArrays 变量区中并将 Filled 置为“1”，程序才会控制实轴按照新设定的运动时间和位置运动；若未在 DelayTime 的时间范围内将 Filled 置为“1”，则指令运行完成，Done 置为“1”。

MaxPosition, MinPosition 和 MinVelocity 分别表示允许运动的最大位置，MinPosition 表示允许运动的最小位置，MinVelocity 表示允许运动的最小速度。

Empty 为写入新运动轨迹参数允许标志。Empty 为“1”时，用户可以写入新的时间和位置。

PointsArray 引脚存储的是型值点数组首地址，每个型值点包含目标位置和运动时间两个值。目前默认的为 4 个型值点，第一个和第四个点分别为整条曲线的起始点和终止点，而中间的两个点为控制运动曲线的型值点，建议使用 4 个或更多型值点，当型值点小于 4 个点时，程序将控制轴按照直线运动。

(2) 连续空间直线运动模式

通过 MC_MoveContinuousCurve 指令控制轴组按照 PointsArray 设定各段空间直线绝对目标位置，进行连续空间直线运动，相邻两空间直线衔接处采用空间圆弧过渡如图 7.8.30 所示。

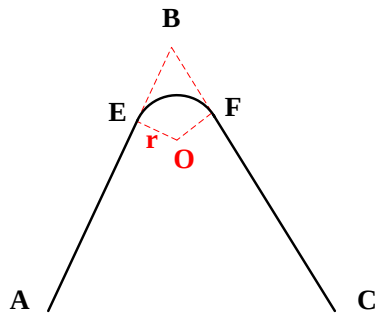


图 7.8.30 相邻两空间直线连接示意图

CurveMode 曲线模式 4 和 5 为连续空间直线运动模式，用于实现如“门”字型轨迹运动控制。其中，模式 4 为给定进给速度进行连续空间直线运动；模式 5 为给定整个曲线插补时间进行连续空间直线运动。

模式 4：根据 CurveParam 数组 1 加减速时间常数在第 1 段和最后一段空间直线进行加减速控制，中间轨迹按照 MinVelocity 进给速度匀速运动。如图 7.8.31 所示，其中 V_0 为初始速度， V_f 为 MinVelocity 进给速度， V_t 为末速度。

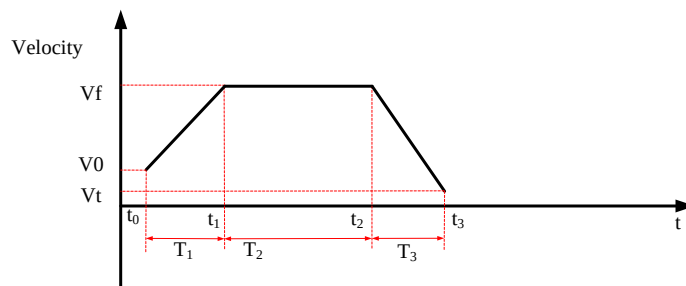


图 7.8.31 模式 4 梯形曲线加减速控制曲线

模式 5：根据 CurveParam 数组 1 加减速时间常数 T_{acc_dec} 和数组 2 整个曲线插补时间 T_{total} ，对整个曲线进行加减速控制，并计算实际的进给速度，MinVelocity 给定的进给速度无效。

当 $T_{total} > 2 \times T_{acc\_dec}$ 时，整个曲线有匀速段；

当 $T_{total} < 2 \times T_{acc\_dec}$ 时，整个曲线无匀速段，仅进行加速和减速控制，如图 7.8.32 所示，其中 V_{act} 为计

算的实际进给速度。

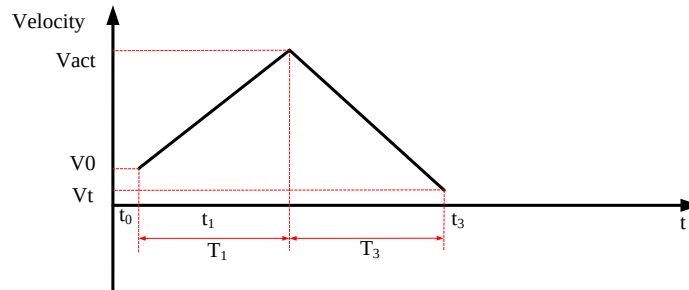


图 7.8.32 模式 5 梯形曲线加减速控制（无匀速段）

PointsArray 引脚存储的是模式 4 和 5 各段空间直线连续运动的绝对目标位置数组首地址，由 ArraySize 确定空间直线总段数。PointsArray 根据轴组设定的轴数依次存放第 1 段至第 ArraySize 段空间直线绝对目标位置坐标值。

MinVelocity 引脚存储的是模式 4 给定的进给速度。

CurveParam 引脚存储的是模式 4 和 5 运动控制参数数组首地址，数组大小为 n，其中数组 1 存储模式 4 和 5 梯形曲线加减速时间常数，数组 2 存储模式 5 整个曲线插补时间，数组 3~n 存储模式 4 和 5 空间圆弧半径值。

DelayTime 引脚存储的是曲线插补完成 Done 信号输出前的延时时间。

BufferMode 引脚存储的是模式 4 和 5 空间半径半径输入模式，为 0 时每段圆弧半径必须指定，为 1 时当圆弧半径为 0，自动选择前一个非 0 圆弧半径。

注意：

①梯形曲线加减速时间常数 CurveParam 第 1 数组为 0，加速段：初始速度直接加速到目标速度；减速段：目标速度直接减速到末速度；

②若空间圆弧半径过大，相邻两空间直线衔接处的空间圆弧会与下一个空间圆弧相交如图 7.8.33 所示为门字型轨迹 A→B→C→D 转角处空间圆弧相交情况，此时控制器会自动修改两段圆弧半径值，以保证运动连续性。

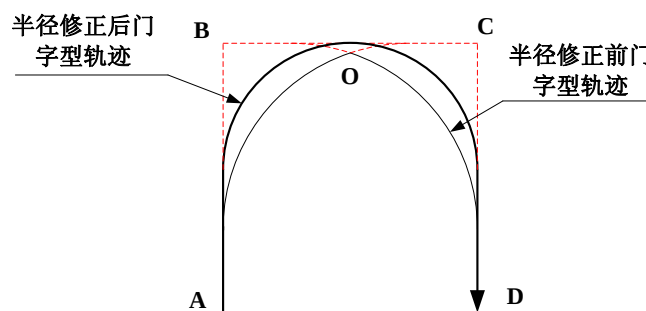


图 7.8.33 空间圆弧相交示意图

③BufferMode 为 0 时，每段圆弧半径必须指定，即需要指定 ArraySize-1 段空间圆弧半径，若半径为 0，则相邻两直线直接过渡。

④当轴组轴数设置为 4 时，第 4 轴按照设定的绝对目标位置，以 CurveParam 数组 1 设定的梯形曲线加减速时间常数运动，起始速度为 0，到达目标位置时速度为 0。指令起始位置与目标位置按照 PointsArray 各段空间直线第 4 轴位置增量为 0 处划分；第 4 轴插补实际起始位置为指令起始位置当前直线段与前一直线段衔接处的空间圆弧起点；第 4 轴插补实际结束位置为指令目标位置当前直线段与后一直线段衔接处的空间圆弧终点。

如图 7.8.34 各段直线坐标 A(0,0,0), B(0,100,0,0), C(100,100,0,70), D(100,0,0,70), E(200,0,0,120)。第 4 轴第 1 段指令起始位置为 B 点坐标 0，目标位置为 C 点坐标 70，实际插补起始位置为 E 点，插补结束位置

为 N 点，即在连续曲线按照 E→F→M→N 运动时，第 4 轴由起始位置 0 经过加减速控制后，同步运动到目标位置 70 处；第 4 轴第 2 段指令起始位置为 D 点坐标 70，目标位置为 E 点坐标 120，实际插补起始位置为 I 点，插补结束位置为 E 点，即在连续曲线按照 I→J→E 运动时，第 4 轴由起始位置 70 经过加减速控制后同步运动到目标位置 120 处。

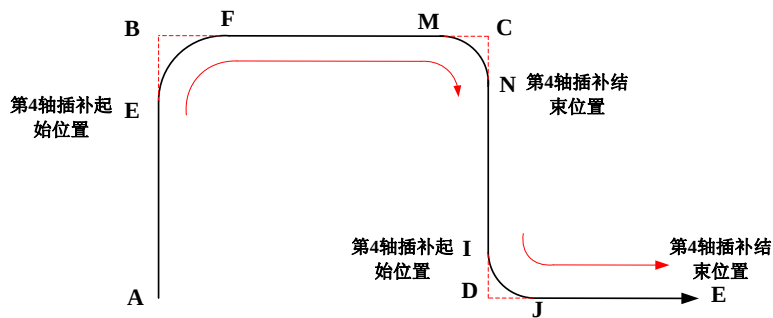


图 7.8.34 第 4 轴插补示意图

(3) 连续直线圆弧运动模式

通过 MC_MoveContinuousCurve 指令控制轴组按照 PointsArray 设定各段直线或圆弧绝对目标位置，进行连续直线圆弧运动，相邻两段曲线衔接处可以采用圆弧过渡。

MaxPosition 引脚存储每段圆弧半径值，正值为劣弧加工，负值为优弧加工，0 采用 MinPosition 指定圆心坐标进行圆弧加工，半径模式下无法加工整圆。

MinPosition 引脚存储每段圆弧圆心增量坐标（圆心绝对坐标 - 起点坐标），在 MaxPosition 为 0 时有效。

MinVelocity 引脚存储连续直线圆弧运行速度，单位 units/s。

CurveParam 引脚为数组输入，第 1 个数组为 T 曲线加减速时间常数，单位 ms；第 3~n 为每段曲线衔接处圆弧半径，若为 0 采用直线直接过渡。

BufferMode 引脚为数组输入，第 1 个数组为曲线衔接处圆弧设定方式，若为 0 则每段圆弧必须设定，若为 1 则每段圆弧为 0 时，会自动搜索前一个不为 0 的圆弧半径；数组 3~n 为每段曲线插补类型，0 为直线插补、1 为顺圆插补、2 为逆圆插补。

● 时序图

图 7.8.35 为该指令的引脚时序图。Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当 Empty 为“1”，表示可以写入新点进行插补，此时若 Filled 为“0”则经过 DelayTime 时间后，Done 被置“1”；若 Filled 为“1”，则读入新的型值点；当 Empty 变为“1”，表示可以写入新点进行插补。如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

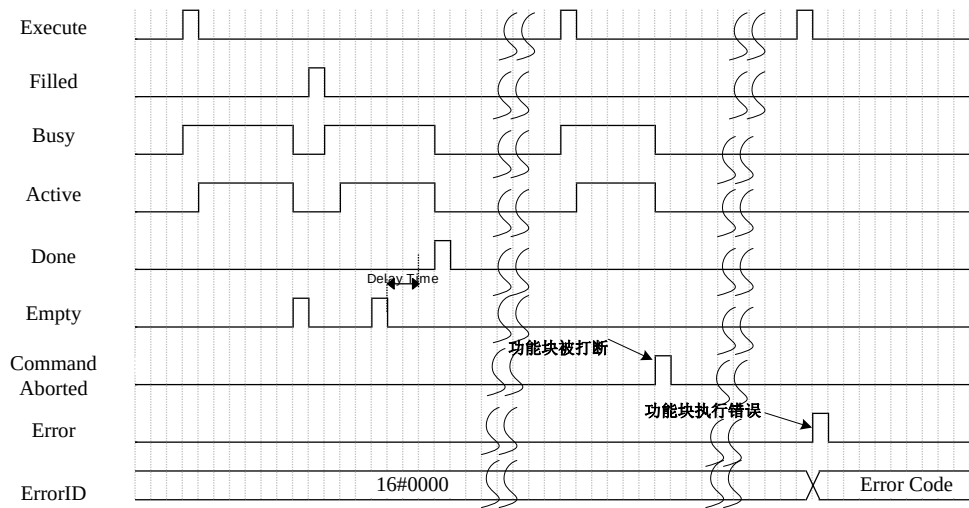


图 7.8.35 指令引脚时序图

● 应用示例

(1) 位置-时间曲线运动模式

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择两个单轴（0、1）在轴组中的 ID 分别为 0、1。保持默认运动模型“直角坐标系”，下载参数到设备。

图 7.8.36 VD100、VD104、VD108、VD112 为轴依次要达到的目标位置，VD102、VD106、VD110、VD114 为到达目标位置的时间。VW200 记录 MC_MoveContinuousCurve 指令读取运动轨迹参数的次数，初值为 2。

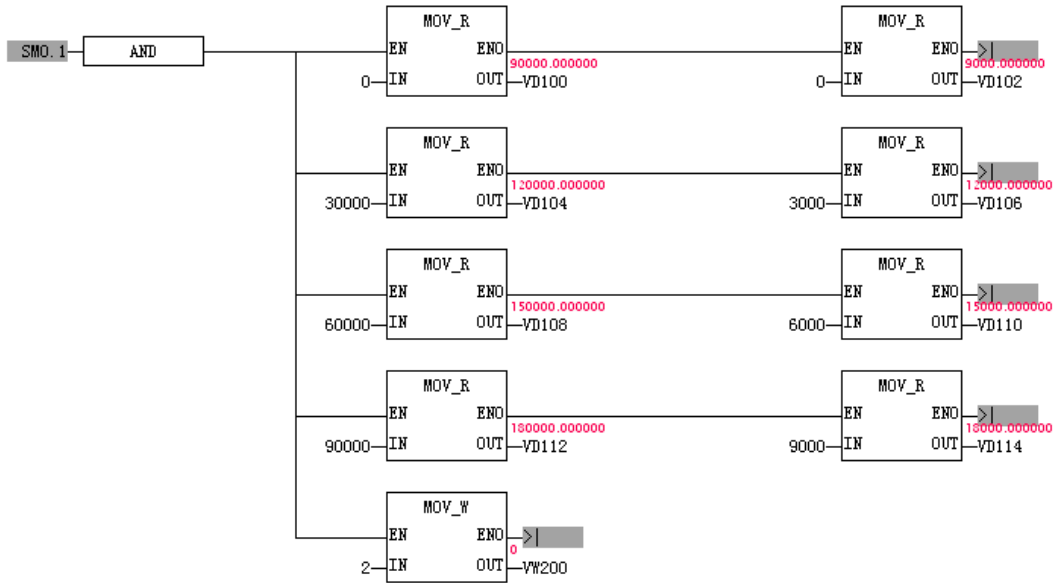


图 7.8.36 第一次曲线运行轨迹参数

图 7.8.37 中将 M0.0 置为“1”，程序调用 MC_MoveContinuousCurve 指令，轴按照当前读入的运动轨迹参数运行。Empty 输出为“0”时，表明指令可以读入下一次曲线运行轨迹参数。

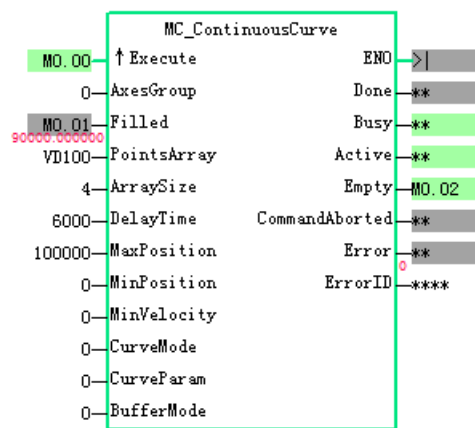


图 7.8.37 连续曲线运行

Filled 引脚分配变量 M0.1。当 M0.1 置“1”，MC_MoveContinuousCurve 指令读入新的运动轨迹参数后，程序复位 M0.1，如图 7.8.38 所示。Empty 引脚分配变量 M0.2。M0.2 产生一次上升沿，则 VW200 减 1，以此作为标记控制 MC_MoveContinuousCurve 指令读运动轨迹参数次数。调用 MC_ReadActualPosition 指令读轴位置。

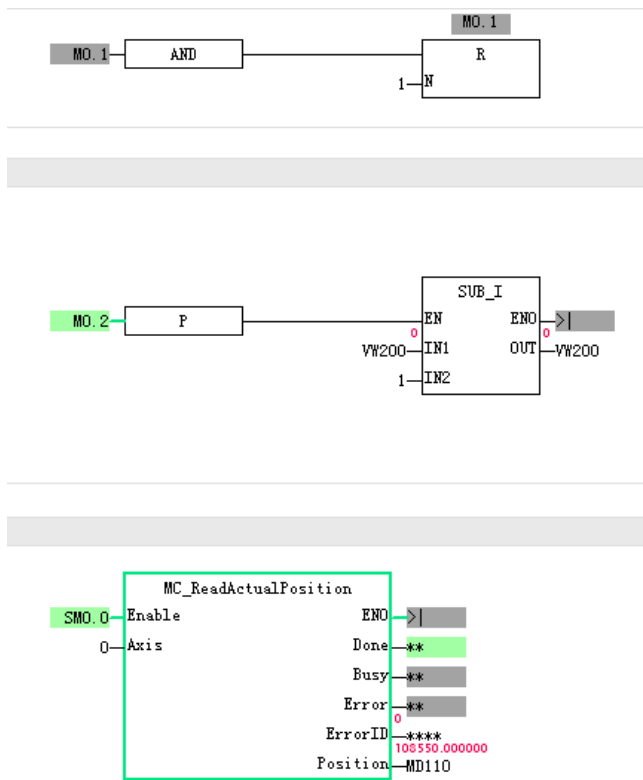


图 7.8.38 复位控制标志及读取轴位置

当 VW200 大于等于 1 并且 M0.2 为“1”时，写入新的运动轨迹参数，并置 M0.1 为“1”，如图 7.8.39 所示。

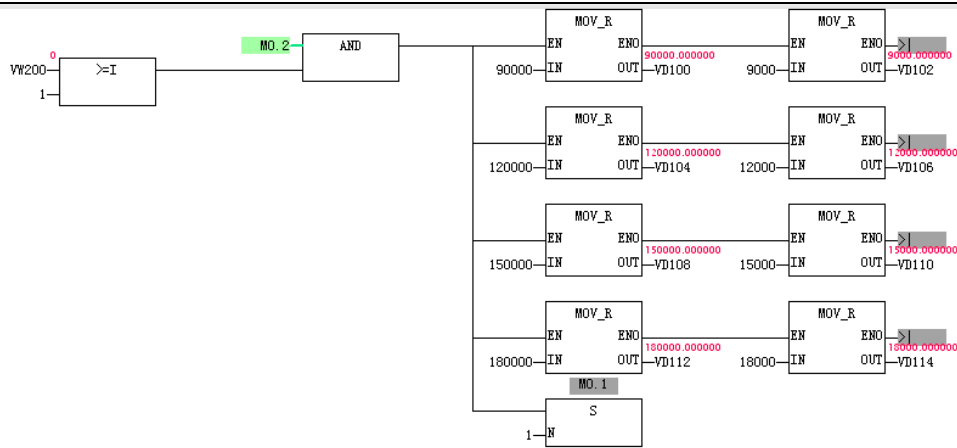


图 7.8.39 写入新的运动轨迹参数

(2) 连续空间直线运动模式

如图 7.8.40 所示为门字型指令轨迹，其运动方式为 $A \rightarrow B \rightarrow C \rightarrow D$ ，为减少机械冲击，各空间直线转角处采用空间圆弧连接。

起始位置 A 点坐标为 (0,0,0,0)，第 1 段空间直线 $A \rightarrow B$ 目标位置 B 点坐标 (0,0,100,0)，第 2 段空间直线 $B \rightarrow C$ 目标位置 C 点坐标 (100,100,100,0)，最后 1 段空间直线 $C \rightarrow D$ 目标位置 D 点坐标 (100,100,0,0)。

第 1 段空间圆弧半径 R 为 50，第 2 段空间圆弧半径 R 为 30，梯形曲线加减速时间常数为 500ms，选择模式 5，整个曲线插补时间 1000ms。

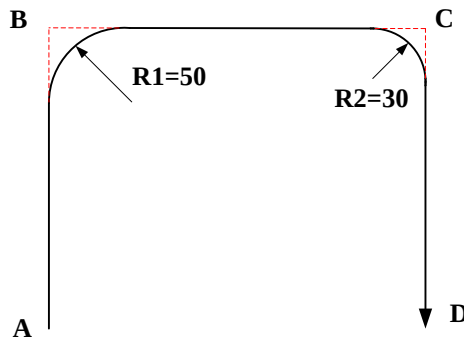


图 7.8.40 门字型轨迹

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择 4 个轴 (0、1、2、3) 在轴组中的 ID 分别为 0、1、2、3。保持默认运动模型“直角坐标系”，下载参数到设备。

如图 7.8.41 所示，VW0 存入直线段总数 3，VW1 存入 CurveMode 模式 5，VD200 存入模式 4 进给速度 10unit/s，VD300 梯形曲线加减速时间常数 500ms，VD302 存入曲线总插补时间 1000ms，VD304 存入第 1 段空间圆弧半径 50unit，VD306 存入第 2 段圆弧半径 30，VD100~VD122 依次存入第 1 段~第 3 段空间直线 B、C、D 点坐标。

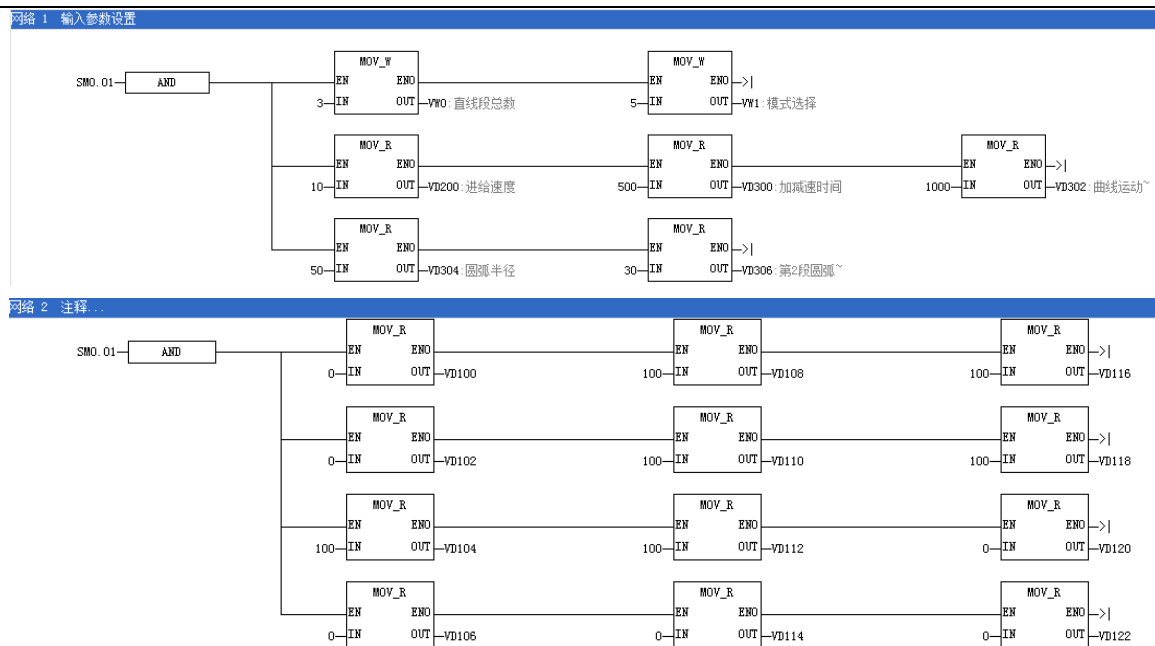


图 7.8.41 连续空间直线运动模式参数配置

图 7.8.42 中 MC_MoveContinuousCurve 指令 PointsArray 引脚输入空间直线连续运动的绝对目标位置数组首地址 VD100, MinVelocity 引脚输入进给速度 VD200, CurveMode 引脚输入曲线模式 VW1, CurveParam 引脚输入连续曲线运动控制参数数组首地址 VD300。将 M0.1 置为“1”, 程序使能 MC_MoveContinuousCurve 指令, 轴组以当前读入的运动轨迹参数按照门字型轨迹运行。

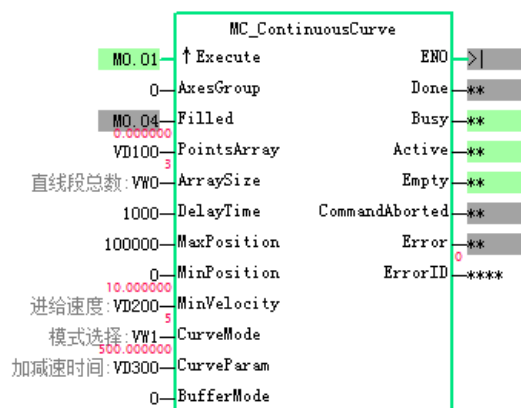


图 7.8.42 MC_MoveContinuousCurve 引脚配置

(3) 连续直线圆弧运动模式

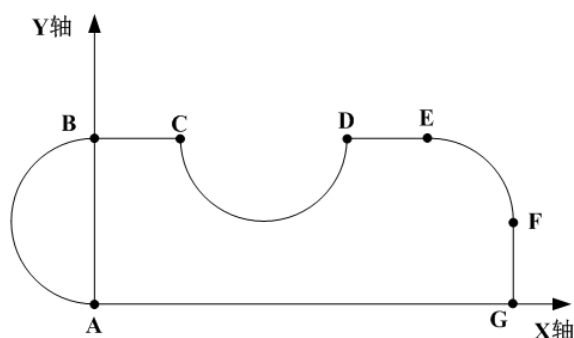


图 7.8.43 连续直线圆弧运动轨迹

采用 MC_ContinuousCurve 指令完成连续直线圆弧运动与控制。如图 7.8.43 所示为 4 段直线+3 段圆弧构成的轨迹图形。

设计实际运动过程为 $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow A$ ，如图 7.8.44 所示。首先运动控制器控制各轴由原点顺圆运动到 B；然后按直线+逆圆+直线+顺圆+直线+直线从 $B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow A$ 完成连续直线圆弧运动轨迹控制；

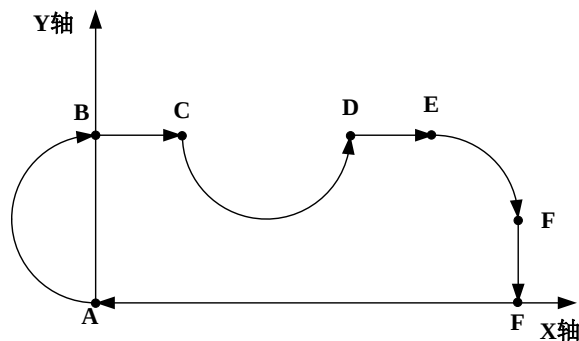


图 7.8.44 运动控制轨迹

运动参数设置如表 7.8.1 运动参数所示。

表 7.8.1 运动参数

参数说明	值
连续空间直线起点 A 坐标	(0, 0, 0, 0)
第 1 段顺圆终点 B 坐标	(0, 200, 0, 0)
第 2 段直线终点 C 坐标	(100, 200, 0, 0)
第 3 段逆圆终点 D 坐标	(300, 200, 0, 0)
第 4 段直线终点 E 坐标	(400, 200, 0, 0)
第 5 段顺圆终点 F 坐标	(500, 100, 0, 0)
第 6 段直线终点 G 坐标	(500, 0, 0, 0)
第 7 段直线终点 A 坐标	(0, 0, 0, 0)
第一个顺圆 A-B 圆心增量坐标	(0, 100, 0, 0)
第二个逆圆 C-D 圆心增量坐标	(100, 0, 0, 0)
第三个顺圆 E-F 圆心增量坐标	(0, -100, 0, 0)
直线圆弧转接处圆弧过渡半径	20
连续空间直线运动加减速时间常数（单位 ms）	200

VD0~VD54 输入 $B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow A$ 指令目标位置，如图 7.8.45 所示。

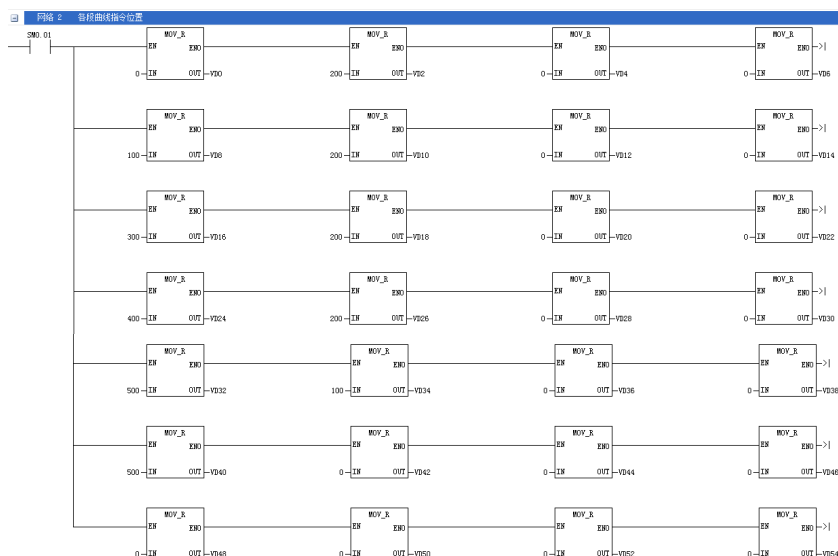


图 7.8.45 目标位置设置

VD100~VD106 输入顺圆 A-B 圆心增量坐标(0,100,0,0)，VD108~VD114 输入逆圆 C-D 圆心增量坐标

(100,0,0,0), VD116~VD122 输入顺圆 E-F 圆心增量坐标(0,-100,0,0), 如图 7.8.46 所示。

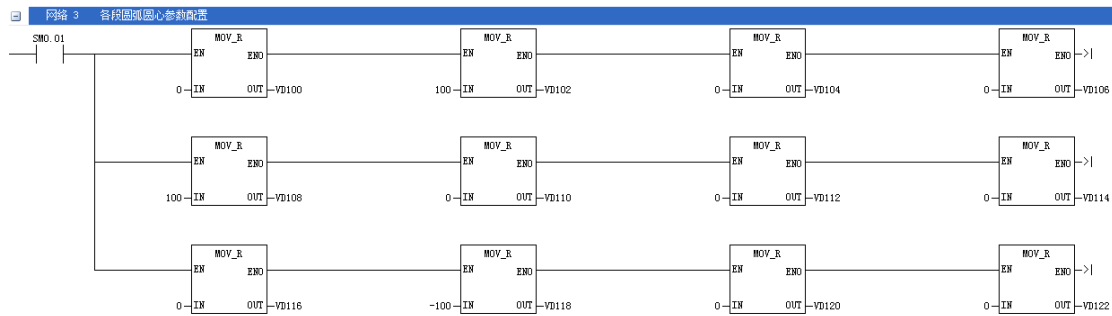


图 7.8.46 圆心增量坐标设置

VD500 输入加减速时间 200ms, VD504 输入连续 2 段曲线间过渡圆弧半径 20mm, VW1002~VW1008 输入每段曲线加工类型(0 表示直线、1 表示顺圆、2 表示逆圆), 如图 7.8.47 所示。

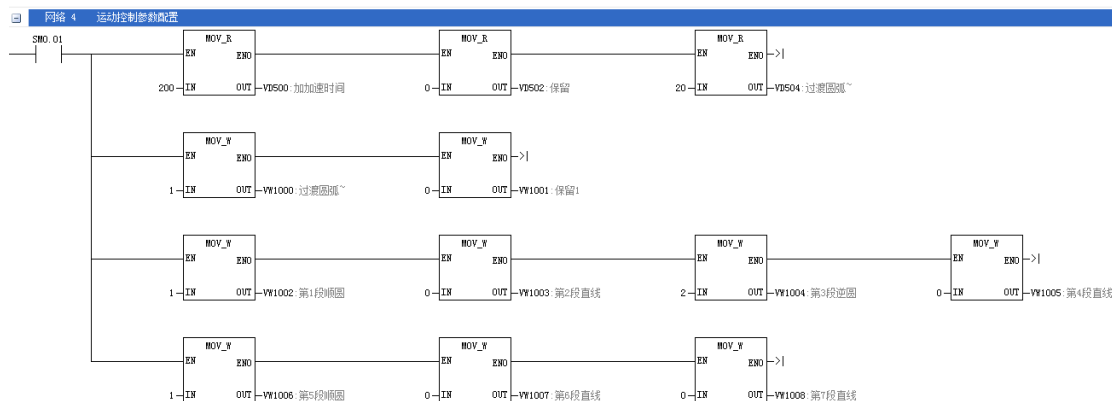


图 7.8.47 基本参数设置

如图 7.8.48 所示, 将 M1.0 置 1 使能运动指令。

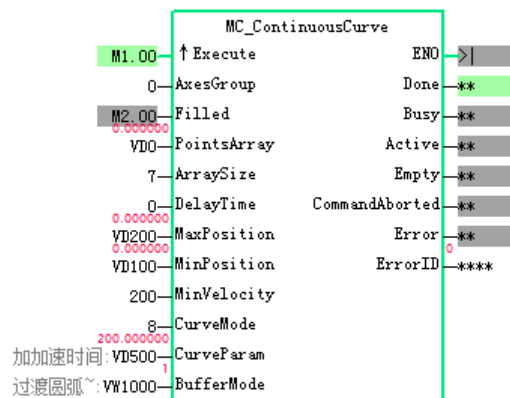


图 7.8.48 连续直线圆弧运动

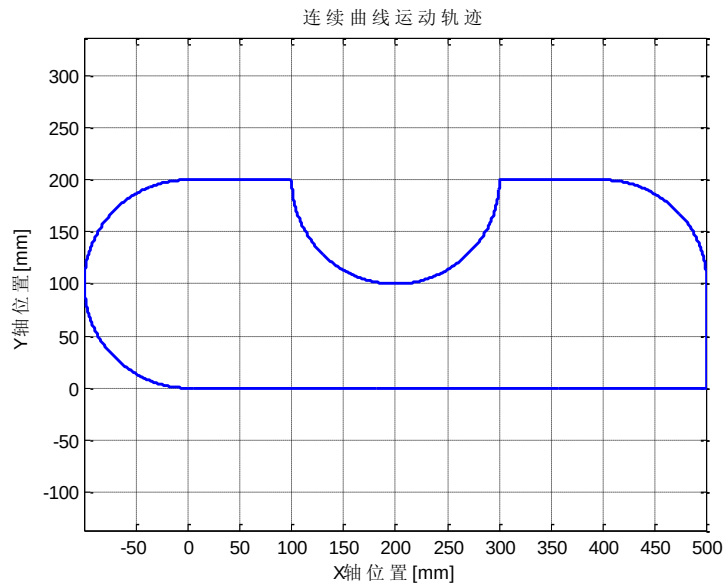


图 7.8.49 直线过渡连续直线圆弧运动轨迹

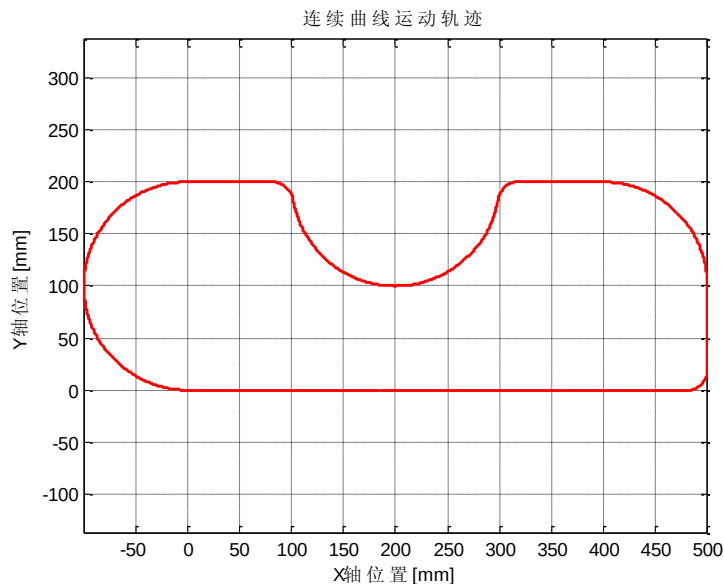


图 7.8.50 圆弧过渡连续直线圆弧运动轨迹

图 7.8.49 为连续 2 段曲线间采用直线过渡方式，图 7.8.50 为采用圆弧过渡方式。采用直线可以精确定位，但转角处速度突然换向导致速度变化，采用圆弧过渡运动轨迹平滑，可以保证整个加工过程速度平稳。实际应用中可以根据具体工艺要求选择直线过渡或圆弧过渡。

7.8.10 读轴组位置 MC_GroupReadActualPosition

● 指令参数

MC_GroupReadActualPosition <<Enable EN0 > 0AxesGroup Valid ** 0CoordSystem Busy ** Error ** ErrorID **** CommandPosition **** Position ****		读取轴组的绝对位置。 指令使能后实时读取轴组的目标位置与当前绝对位置。 轴组错误时,读取的 Position 和 CommandPosition 无效。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID

CoordSystem	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Valid	位变量	BOOL	0 或 1	为“1”时，表示状态被确定	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	
CommandPosition	双字变量	REAL	实数	轴组目标位置，单位 units	
Position	双字变量	REAL	实数	轴组当前位置，单位 units	

● 详细说明

通过 MC_GroupReadActualPosition 指令，读取空间指定轴组的目标位置值与当前的实际位置值。

● 时序图

图 7.8.51 为该指令引脚时序图。Busy 和 Enable 同时被置“1”；当位移值可获取时 Valid 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

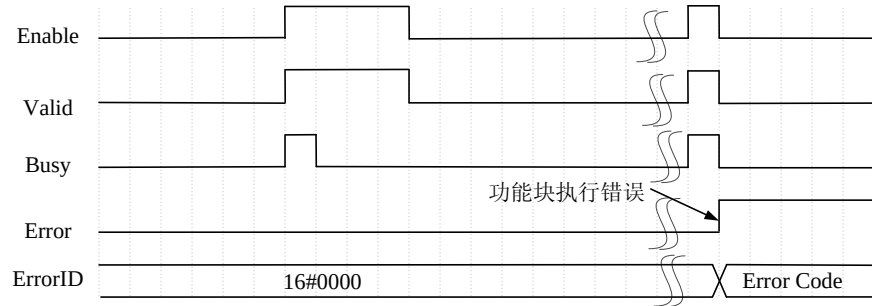


图 7.8.51 指令引脚时序图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.52 为“状态符号表”中，写入的直线运动的目标位置和监控的轴组实时位置。

VD200	2536	四字节浮点数	100000.000000		100000
VD202	2538	四字节浮点数	200000.000000		200000
VD204	2540	四字节浮点数	150000.000000		150000
VD300	2636	四字节浮点数	100000.000000		
VD302	2638	四字节浮点数	200000.000000		
VD304	2640	四字节浮点数	150000.000000		

图 7.8.52 目标位置和实时位置

图 7.8.53 中将 M0.3 置为“1”，程序调用 MC_MoveLinearAbsolute 指令，轴执行直线运动并且可以通过 VD300，VD302，VD304 监控轴组的实时位置。

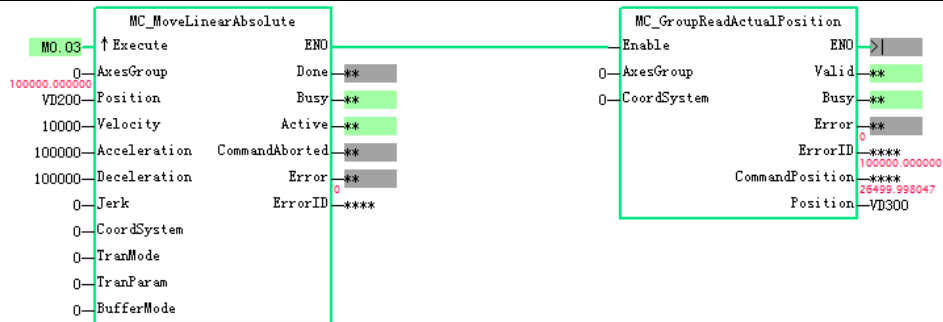


图 7.8.53 直线运动和实时位置读取

7.8.11 读轴组速度 MC_GroupReadActualVelocity

● 指令参数

<pre> MC_GroupReadActualVelocity <<-- Enable ENO --> 0-- AxesGroup Busy ** 0-- CoordSystem Error ** ErrorID **** Velocity **** PathVelocity **** </pre> 读取轴组的实际速度。 指令使能后实时读取轴组速度与轴组中各个轴的速度。 轴组出错误时，读取的 Velocity 和 PathVelocity 无效。					
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
CoordSystem	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	默认值	注解
ENO	位变量	BOOL	0 或 1	能流输出	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	
Velocity	双字变量	REAL	非负数	轴组各轴当前速度，单位 units/s	
PathVelocity	双字变量	REAL	非负数	轴组当前合速度，单位 units/s	

● 详细说明

通过 MC_GroupReadActualVelocity 指令，读取指定轴组中各个坐标方向的实际速度值和合速度值。从 Velocity 引脚开始的几个双字为读取的各个轴的实时速度，从 PathVelocity 读取的速度为轴组的实时速度。

● 时序图

图 7.8.54 为该指令的引脚时序图。Busy 和 Enable 同时被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

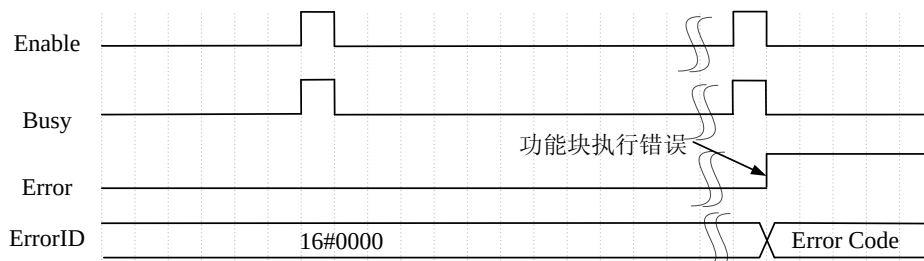


图 7.8.54 指令引脚时序图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.55 在“状态符号表”中，

写入的直线运动的目标位置和监控的轴组实时速度。

VD200	2536	四字节浮点数	100000.000000	100000
VD202	2538	四字节浮点数	200000.000000	200000
VD204	2540	四字节浮点数	150000.000000	150000
VD300	2636	四字节浮点数	3712.890381	
VD302	2638	四字节浮点数	7425.780762	
VD304	2640	四字节浮点数	5570.312012	
VD350	2686	四字节浮点数	9997.806641	

图 7.8.55 轴组运动的目标位置和实时速度

图 7.8.56 中将 M0.3 置为“1”，程序调用 MC_MoveLinearAbsolute 指令和 MC_GroupReadActualVelocity 指令，轴执行直线运动并且可以通过 VD300，VD302，VD304 监控各个单轴的实时速度，VD350 监控轴组的实时速度。

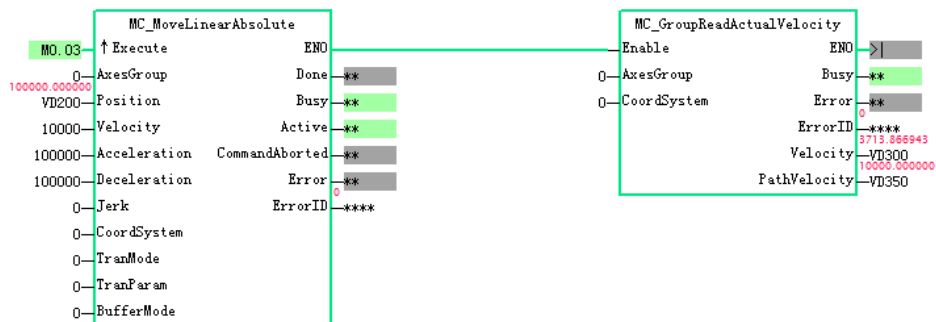


图 7.8.56 直线运动和实时速度读取

7.8.12 读取轴组故障 MC_GroupReadError

● 指令参数

MC_GroupReadError Enable 0 AxisGroup 0		ENO Valid Busy Error ErrorID		读取轴组的故障信息。 指令使能后，读取轴组的错误信息，轴组无错误时 ErrorID 为零，轴组错误时 ErrorID 不为零，具体的错误信息，请查看异常处理表（附录 E）。	
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxisGroup	字变量、常数	WORD	非负整数	0	轴组 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Valid	位变量	BOOL	0 或 1	为“1”时，表示状态被确定	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GroupReadError 指令，读取指定轴组的错误信息。当轴组出现故障后，可以读取轴组的故障信息。

● 时序图

图 7.8.57 为该指令的引脚时序图。Busy 和 Enable 同时被置“1”；当读轴组的故障信息可读时 Valid 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

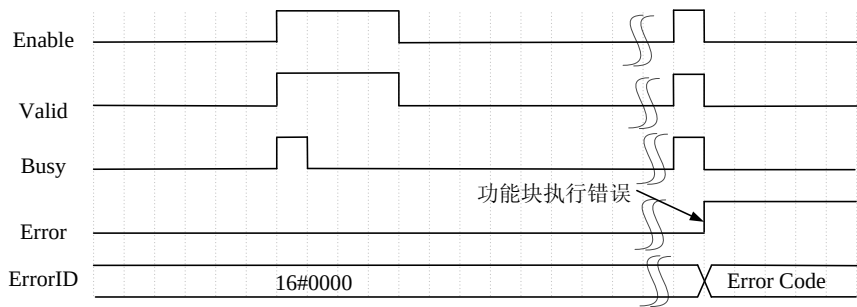


图 7.8.57 指令引脚时序图

7.8.13 轴组复位 MC_GroupReset

● 指令参数

		复位轴组错误状态。 轴组出错后，调用此指令可以清除轴组的错误信息，轴组的错误信息被清除后，轴组才可以继续运动。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
输入	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示复位完成	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Failure	位变量	BOOL	0 或 1	为“1”时，表示复位失败	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GroupReset 指令，对指定的轴组进行复位。当轴运动出现错误时，可将轴组进行复位，复位后，所有关于轴组的参数都会恢复到默认值

● 时序图

图 7.8.58 为该指令的引脚时序图。Busy 和 Enable 同时被置“1”；当复位完成时，Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID（错误码）的值找到导致错误的原因。

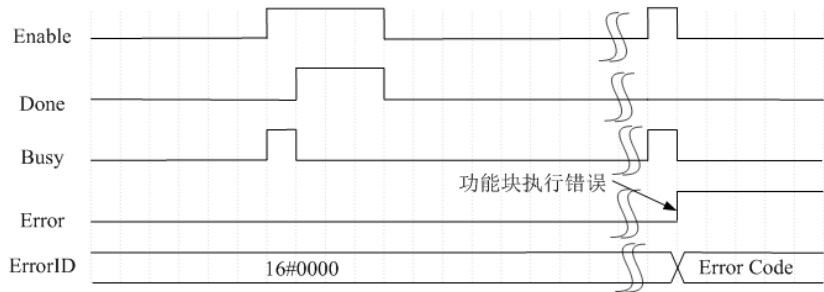


图 7.8.58 指令引脚时序图

7.8.14 读轴组状态 MC_GroupReadStatus

● 指令参数

读取轴组的运动状态和速度状态。 轴组运动状态：轴组运行状态、轴组回原点状态、轴组错误停止状态、轴组停止状态、轴组停止中状态、轴组禁能状态。 轴组速度状态：轴组加速状态、轴组减速状态、轴组恒速状态、到达目标位置状态。 当 Valid 为“1”时，表示已经获取到轴组上的状态信息，轴组的运动状态和速度状态会直接显示在右侧的输出引脚上。					
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Valid	位变量	BOOL	0 或 1	为“1”时，表示状态被确定	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
GroupMoving	位变量	BOOL	0 或 1	为“1”时，表示轴组处于运行状态	
GroupHoming	位变量	BOOL	--	保留	
GroupErrorStop	位变量	BOOL	0 或 1	为“1”时，表示轴组处于出错停止状态	
GroupStandby	位变量	BOOL	0 或 1	为“1”时，表示轴组处于静止状态	
GroupStopping	位变量	BOOL	0 或 1	为“1”时，表示轴组处于停止中状态	
GroupDisabled	位变量	BOOL	0 或 1	为“1”时，表示轴组处于禁能运行状态	
ConstantVelocity	位变量	BOOL	0 或 1	为“1”时，表示轴组处于恒速状态	
Accelerating	位变量	BOOL	0 或 1	为“1”时，表示轴组处于加速状态	
Decelerating	位变量	BOOL	0 或 1	为“1”时，表示轴组处于减速状态	
InPosition	位变量	BOOL	0 或 1	为“1”时，表示轴组到达目标位置	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_GroupReadStatus 指令，读取指定轴组的状态信息。其中 GroupMoving、GroupHoming、GroupErrorStop、GroupStandby、GroupStopping、GroupDisable 为一组状态，分别表示轴组处于运行状态、回原点状态、错误状态、停止状态、减速状态、禁能状态，其状态值为互斥的，即同一时刻，只有一个位为“1”；ConstantVelocity、Accelerating、Decelarating 为一组状态，分别表示轴组处于恒速、加速、减速状态。

● 时序图

图 7.8.59 为该指令的引脚时序图。Busy 和 Enable 同时被置“1”；当读取到轴组的运动状态时 Done 被置“1”；如果在指令执行期间有错误发生，则 Error 被置“1”，可以通过 ErrorID(错误码)的值找到导致错误的原因。

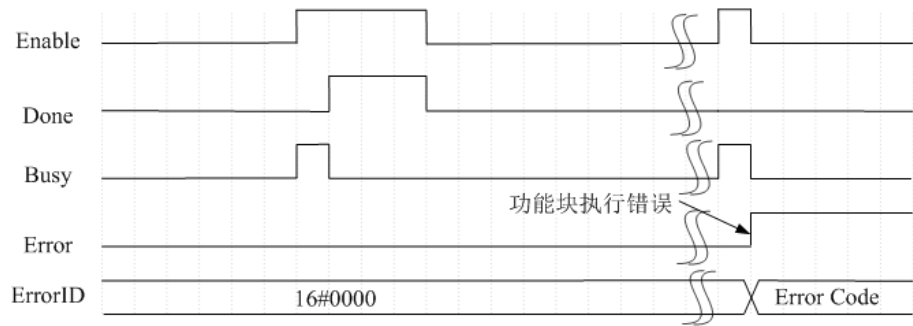


图 7.8.59 指令引脚时序图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。图 7.8.60 为“状态符号表”中，写入的轴组目标位置参数表。

VD200	2536	四字节浮点数	100000.000000	100000
VD202	2538	四字节浮点数	200000.000000	200000
VD204	2540	四字节浮点数	150000.000000	150000

图 7.8.60 轴组目标位置参数表

图 7.8.61 中程序调用 MC_GroupReadStatus 指令读取轴组状态，将 M0.5 置为“1”，程序调用 MC_MoveLinearAbsolute 指令，轴组开始直线运动。运动开始时，轴组开始加速运动 GroupMoving 引脚和 Accelerating 引脚被置“1”，当速度达到目标速度后，ConstantVelocity 引脚被置“1”，当轴组接近目标位置时，轴组开始减速运动，Decelerating 被置“1”，当轴组运动到目标位置后，轴组停止，GroupStandBy 和 InPosition 被置“1”。

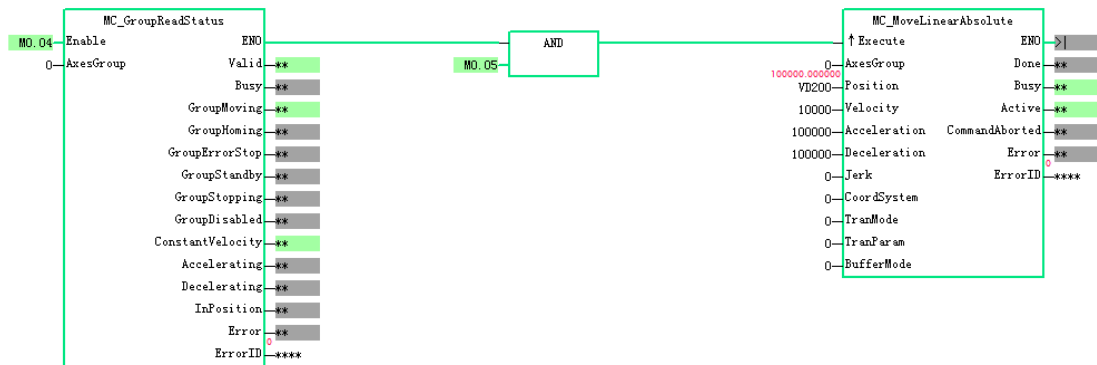


图 7.8.61 轴组直线运动和轴组状态读取

7.8.15 轴组使能 MC_GroupEnable

● 指令参数

MC_GroupEnable << Enable 0 AxesGroup ENO Done Busy CommandAborted Error ErrorID			轴组使能 轴组禁能（MC_GroupDisable）后，需调用轴组使能指令，轴组才可以继续运动		
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	

Done	位变量	BOOL	0 或 1	为“1”时，表示使能完成
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）

- 详细说明

MC_GroupEnable 指令为轴组使能指令，如果轴组没有使能，则所有关于轴组的指令都会报错，轴组默认为使能状态，MC_GroupEnable 配合 MC_GroupDisable 指令配合使用，当调用 MC_GroupDisable 后，若要使用轴组指令则需要首先调用 MC_GroupEnable 将轴组使能。

- 时序图

图 7.8.62 为该指令的引脚时序图。Busy 和 Enable 同时被置“1”；使能完成时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

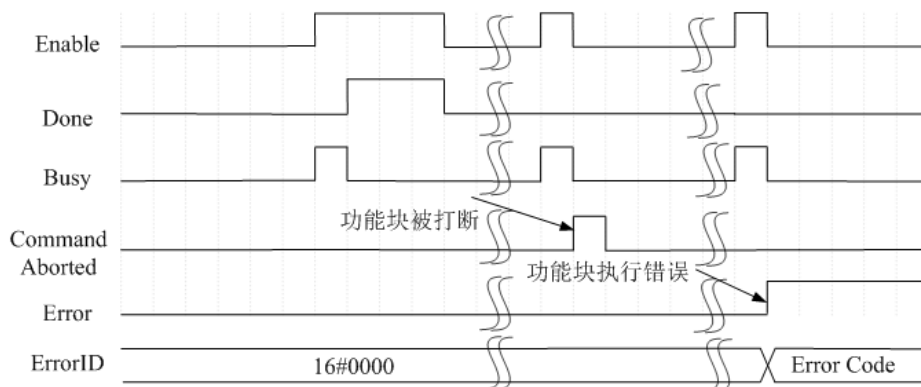


图 7.8.62 指令引脚时序图

- 应用示例

无

7.8.16 轴组禁能 MC_GroupDisable

- 指令参数

		轴组禁能。 指令使能后，指定的轴组禁能，一般在轴组停止状态时使用。如果轴组正在运行中，使能轴组禁能，轴组会报错，轴组被禁能后，需要调用轴组使能指令，轴组才可以继续运动。			
输入	变量区	数据类型	数据范围	默认值	注解
Enable	位变量	BOOL	0 或 1	0	高电平触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示轴组禁能完成	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

- 详细说明

MC_GroupDisable 指令为轴组禁能指令，当轴组禁能指令被调用后，调用的所有关于轴组的指令都会报

错。当出现某些情况，需要禁止轴组运动时，就可以调用 MC_GroupDisable 指令，这样就可以避免误操作轴组指令。

● 时序图

图 7.8.63 为该指令的引脚时序图。Busy 和 Enable 同时被置“1”；禁能完成时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

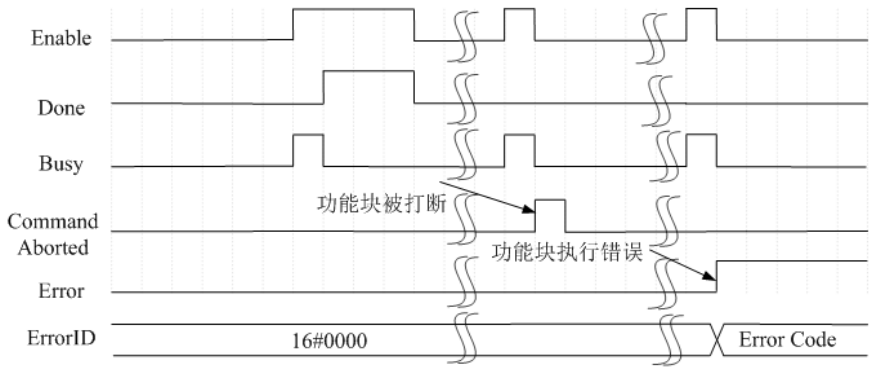


图 7.8.63 指令引脚时序图

● 应用示例

在“运动控制参数配置”中，增加一个轴组 0，配置使能轴组 0，并选择三个单轴（0、1、2）在轴组中的 ID 分别为 0、1、2。保持默认运动模型“直角坐标系”，下载参数到设备。

如图 7.8.64 所示，首先将 M100.0 置为“1”，程序调用 MC_GroupDisable 指令，将轴组 0 禁能。然后将 M0.1 置为“1”，程序调用 MC_GroupMoveVelocity 指令，轴组没有匀速运行，这是由于轴组 0 被禁能，所以 MC_GroupMoveVelocity 指令将报错。

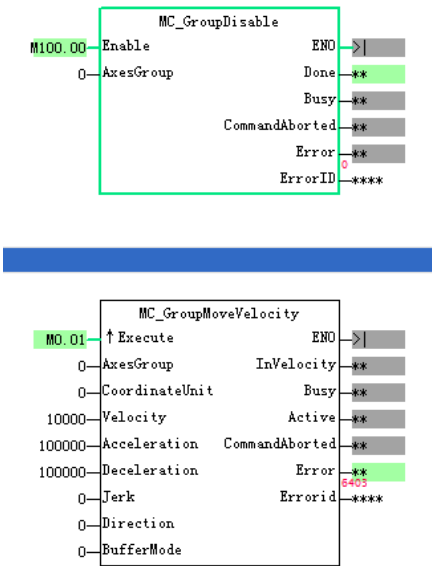


图 7.8.64 轴组禁能后让轴组匀速运行

7.8.17 空间圆弧运行 MC_MoveCircular3D

● 指令参数

MC_MoveCircular3D << ↑ Execute 双击配置曲线 ENO → 0 AxesGroup Done ** 0 CircMode Busy ** ??? EndPoint Active ** ??? AuxPoint1 CommandAborted ** ??? AuxPoint2 Error ** 0 PathChoice ErrorID **** 0 CircDirection 10000 Velocity 100000 Acceleration 100000 Deceleration 0 Jerk 0 TransitionMode 0 TransitionParameter 0 CoordSystem 0 BufferMode					
控制轴组进行空间圆弧插补运动。 指令使能后，控制指定轴组中的每个轴从当前的位置按照设定的运动参数，沿空间圆弧轨迹向目标位置插补运动，运动从当前点开始，至 EndPoint 设定的位置终止；速度以当前速度为起始速度，到达目标位置后轴组的速度为 0					
输入	变量区	数据类型	数据范围	默认值	注解
Execute	位变量	BOOL	0 或 1	0	上升沿触发
AxesGroup	字变量、常数	WORD	非负整数	0	轴组 ID
CircMode	字变量、常数	WORD	0,1 或 2	0	空间圆弧模式 0: 圆心模式 1: 边界点模式 2: 半径模式
EndPoint	双字变量	REAL	实数	指定变量区中的值	空间圆弧终点坐标[数组]，单位 units 数组为 (0—n) 维，n 为轴组中轴个数
AuxPoint1	双字变量	REAL	实数	指定变量区中的值	圆心模式 0: 圆心坐标[数组]，单位 units 边界点模式 1: 圆弧上任意一点坐标[数组]，单位 units 半径模式 2: 起点坐标(0,0,0)模为半径的圆弧平面法向向量坐标[数组]，单位 units 数组为 (0—n) 维，n 为轴组中轴个数
AuxPoint2	双字变量	REAL	实数	指定变量区中的值	圆心模式 0: 加工整圆时圆弧上任意一点坐标[数组]，单位 units 边界点模式 1: 加工整圆时沿圆弧插补方向 AuxPosition1 后的中间点[数组]，单位 units; 半径模式 2: 加工整圆时圆弧上任意一点坐标[数组]，单位 units 数组为 (0—n) 维，n 为轴组中轴个数
PathChoice	字变量、常数	WORD	0~3	0	空间圆弧插补指令与路径选择 0: 绝对指令劣弧插补; 1: 相对指令劣弧插补 2: 绝对指令优弧插补 3: 相对指令优弧插补
CircDirection	字变量、常数	WORD	0 或 1	0	圆心插补模式时，指定圆弧插补方向: 0: 顺时针插补方向 1: 逆时针插补方向
Velocity	双字变量、常数	REAL	非负数	10000	运行目标速度，单位 units/s
Acceleration	双字变量、常数	REAL	非负数	100000	加速度；若为 0，则无加速过程，

					单位 units/s ²
Deceleration	双字变量、常数	REAL	非负数	100000	减速度；若为 0，则无减速过程，单位 units/s ²
Jerk	双字变量、常数	REAL	非负数	0	加加速度；若为 0，速度曲线为梯形，若非 0，速度曲线为 S 形，单位 units/s ³
TransitionMode	字变量、常数	WORD	0 或 1	0	加减速控制方式选择 0：根据 Jerk 选择 S 形或梯形加减速； 1：指数加减速
TransitionParameter	双字变量、常数	REAL	0 或 1	0	指数加减速时间常数，单位 ms
CoordSystem	字变量、常数	WORD	--	0	保留
BufferMode	字变量、常数	WORD	--	0	保留
输出	变量区	数据类型	数据范围	注解	
ENO	位变量	BOOL	0 或 1	能流输出	
Done	位变量	BOOL	0 或 1	为“1”时，表示指令执行完毕	
Busy	位变量	BOOL	0 或 1	为“1”时，表示指令正在执行	
Active	位变量	BOOL	0 或 1	为“1”时，表示轴组正在运动	
CommandAborted	位变量	BOOL	0 或 1	为“1”时，表示被其他指令打断	
Error	位变量	BOOL	0 或 1	为“1”时，表示运行出错	
ErrorID	字变量	WORD	非负整数	出错时的错误码（错误码见附录 E）	

● 详细说明

通过 MC_MoveCircular3D 指令，控制指定轴组（AxesGroup 输入参数）中的每个轴从当前的位置按照设定的运动参数向目标位置进行空间圆弧插补运动，运动从当前点开始，至 EndPoint 设定的相对位置终止。

CircMode 为圆弧插补的模式选择：0 为圆心模式，1 为边界点模式，2 为半径模式。

圆心模式：在圆心模式下 AuxPoint1 存放的数组为圆心坐标，MC_MoveCircular3D 控制轴组以当前点为起点，以 EndPoint 为终点，以 AuxPoint1 为圆心根据 CircDirection 进行顺圆或逆圆插补。在起点和终点重合加工整圆情况下，需选取圆上任意一点 AuxPoint2 坐标后，才能以 AuxPoint1 为圆心进行整圆插补。圆心模式无法加工半圆，请采用边界点模式。

边界点模式：在边界点模式下 AuxPoint1 存放的数组为运行的圆弧轨迹上的一点，MC_MoveCircular3D 控制轴组以当前点为起点，以 EndPoint 为终点，并以 AuxPoint1 为运动轨迹上的一点进行圆弧插补。在起点和终点重合加工整圆情况下，需按圆弧插补方向 AuxPoint1 后选取圆上任意一点 AuxPoint2，才能进行整圆插补。边界点模式下若起始点与终点 EndPoint 和边界点 AuxPoint1 共线会报错。

半径模式：在半径模式下 AuxPoint1 存放数组为起点坐标(0, 0, 0)模为半径的圆弧平面法向量坐标，如图 7.8.65 所示。MC_MoveCircular3D 以当前点为起点，以 EndPoint 为终点，以 AuxPoint1 模为半径根据 PathChoice 进行优弧或劣弧插补。在起点和终点重合加工整圆情况下，需选取圆上任意一点 AuxPoint2 坐标后，才能以 AuxPoint1 模为半径进行整圆插补。

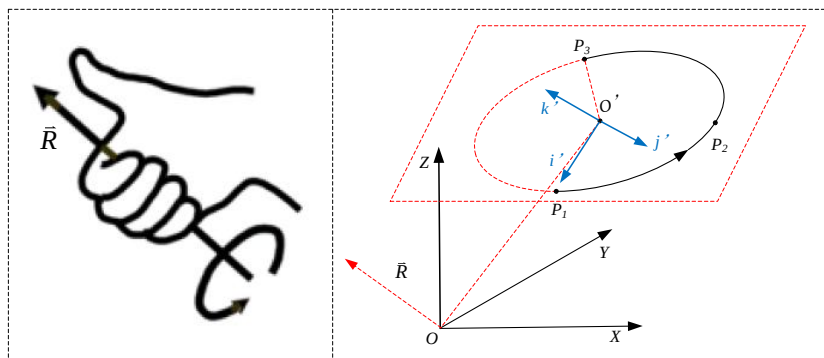


图 7.8.65 半径法向量

PathChioce 为圆弧插补指令与路径选择：即是按照顺时针方向或是逆时针方向进行劣弧或优弧插补。0 为绝对指令劣弧插补，1 为相对指令劣弧插补，2 为绝对指令优弧插补，3 为相对指令优弧插补。

CircDirection 为圆弧插补方向选择：即是按照顺时针方向或是逆时针方向进行圆弧插补。0 为顺时针，1 为逆时针。

TransitionMode 为加减速控制方式选择：即是按照 T 形曲线、S 形曲线或 E 形曲线进行加减速控制。TransitionMode 为 0 时：根据 Jerk 和 Acceleration、Deceleration 选择 S 形或 T 形，当 Jerk 为 0 时为 T 形曲线加减速控制方式，当 Jerk $\neq$ 0 且 Acceleration 与 Deceleration $\neq$ 0 时为 S 形曲线加减速控制，当 Jerk $\neq$ 0 且 Acceleration=0 或 Deceleration=0 时为 T 形曲线加减速控制。TransitionMode 为 1 时为 E 形曲线加减速控制，E 形加减速时间常数为 TransitionParameter 设定值。

TransitionParameter 为指数加减速时间常数，TransitionMode 为 1 时，空间圆弧插补过程按照即是按照 TransitionParameter 进行加减速控制。

● 时序图

如图 7.8.66 所示，Busy 和 Execute 同时被置“1”，Active 在下一周期被置“1”；当平面圆弧插补运动完成时 Done 被置“1”；如果有其它的指令打断此指令，CommandAborted 被置“1”，同时 Busy 和 Active 被置“0”；如果在指令执行期间有错误发生，则 Error 被置“1”，同时轴将停止，可以通过 ErrorID(错误码)的值找到导致错误的原因。

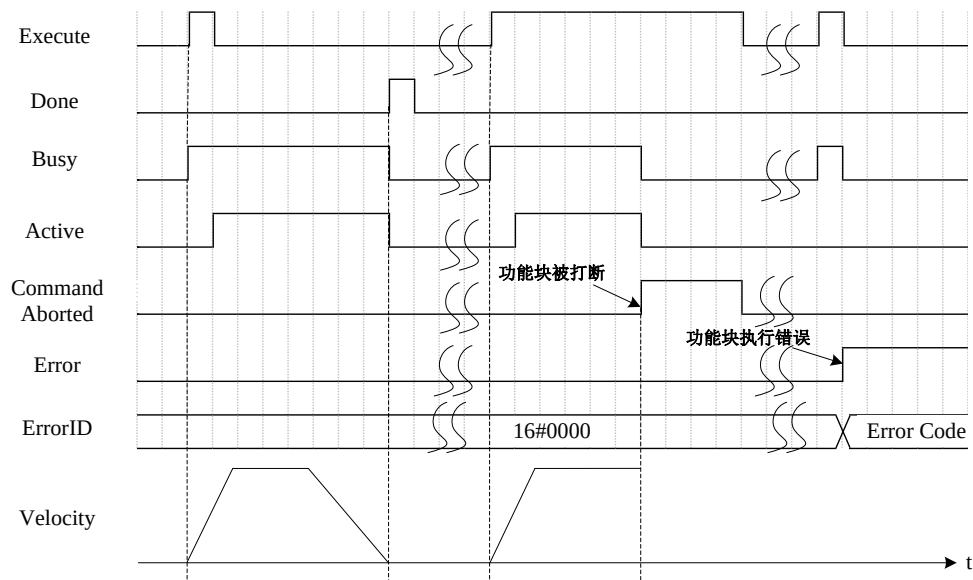


图 7.8.66 空间圆弧运动时序图

● 应用示例

(1) 圆心模式

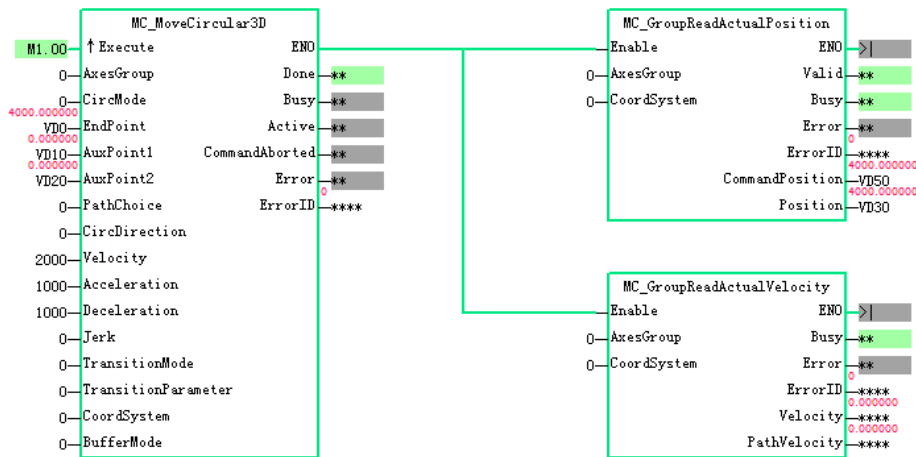


图 7.8.67 空间圆弧运行例程

①XY 平面

如图 7.8.67 所示，设置圆心坐标 (0,4000,0)，起点坐标 (0,0,0)，终点坐标 (4000,4000,0)。图 7.8.68 为 XY 平面顺时针和逆时针圆弧插补的位移曲线。

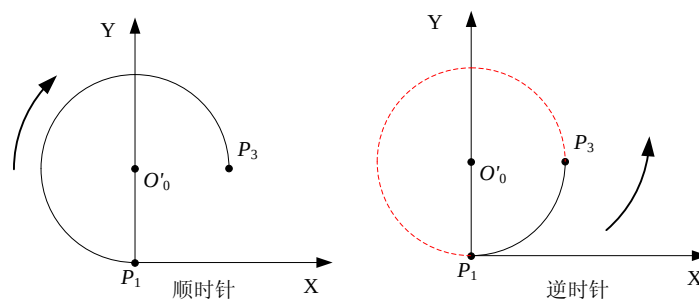


图 7.8.68 圆心模式 XY 圆弧插补

②XZ 平面

设置圆心坐标 (0,0,4000)，起点坐标 (0,0,0)，终点坐标 (4000,0,4000)。图 7.8.69 为 XZ 平面顺时针和逆时针圆弧插补的位移曲线。

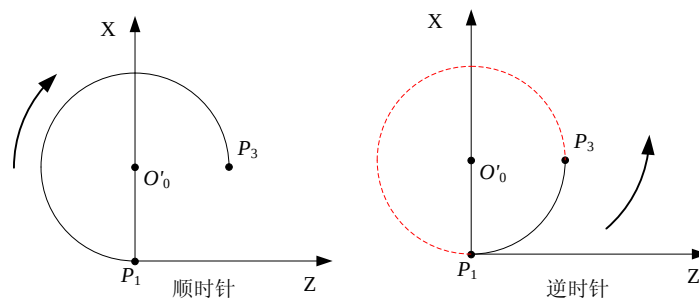


图 7.8.69 圆心模式 XZ 圆弧插补

③YZ 平面测试

设置圆心坐标 (0,0,4000)，起点坐标 (0,0,0)，终点坐标 (0,4000,4000)。图 7.8.70 为 YZ 平面顺时针和逆时针圆弧插补的位移曲线。

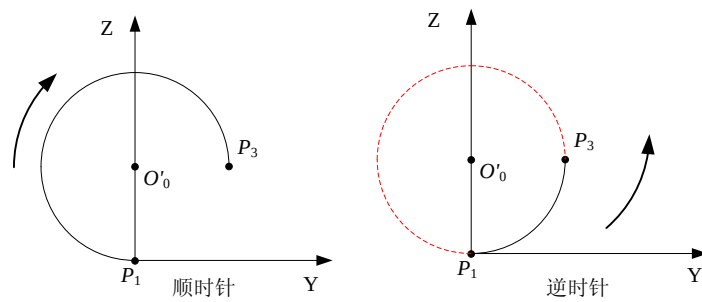


图 7.8.70 圆心模式 XZ 圆弧插补

④任意平面

设置圆心坐标 (2000,4000,1000)，起点坐标 (0,0,0)，终点坐标 (1000,2000,5000)。图 7.8.71 为任意平面空间圆弧插补的位移曲线。

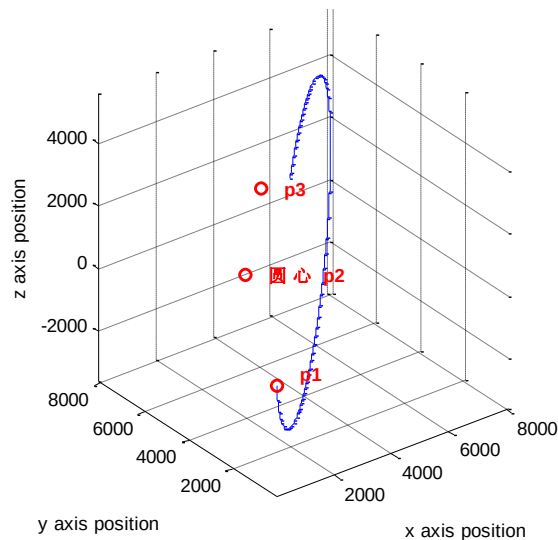


图 7.8.71 圆心模式任意平面空间圆弧插补

⑤整圆加工

圆心坐标 (2000,4000,1000)，起点坐标 (0,0,0)，终点坐标 (0,0,0)，边界点 2 坐标 (1000,2000,5500)。图 7.8.72 为整圆插补的位移曲线。

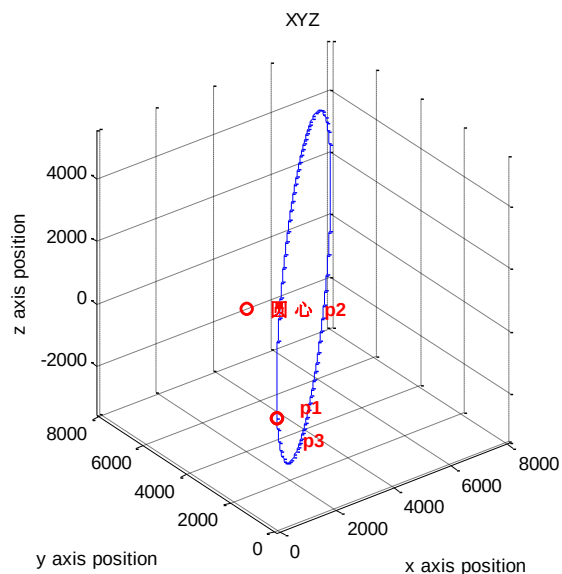


图 7.8.72 圆心模式整圆空间圆弧插补

(2) 边界点模式

①XY 平面

设置边界点坐标 (4000,4000,0)，起点坐标 (0,0,0)，终点坐标 (-4000,4000,0)。图 7.8.73 为 XY 平面边界点模式圆弧插补的位移曲线。

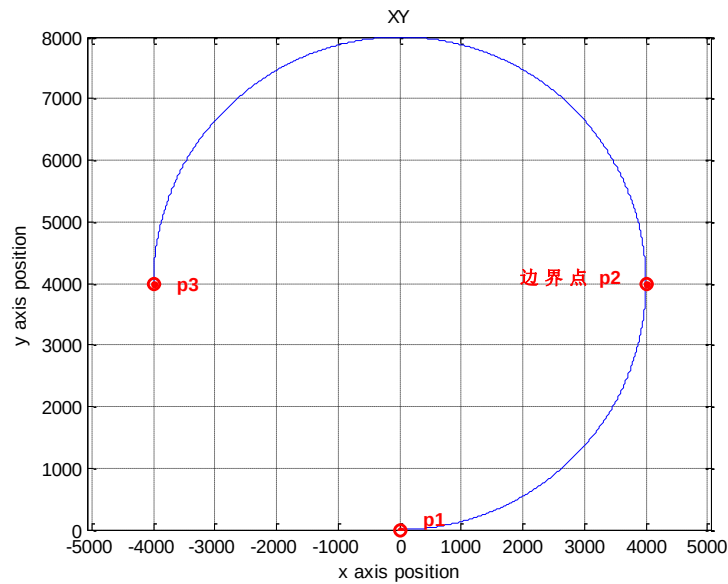


图 7.8.73 边界点模式 XY 平面圆弧插补

②YZ 平面

设置边界点坐标 (0,4000,4000)，起点坐标 (0,0,0)，终点坐标 (0,-4000,4000)。图 7.8.74 为 YZ 平面边界点模式圆弧插补的位移曲线。

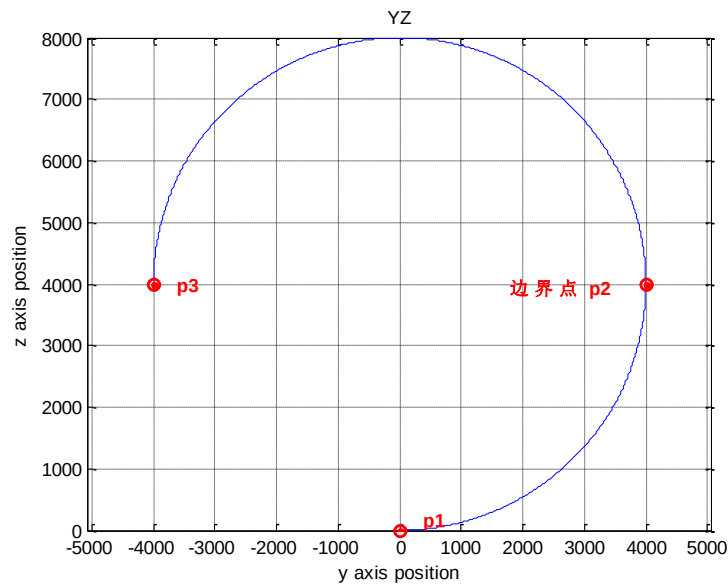


图 7.8.74 边界点模式 YZ 平面圆弧插补

③XZ 平面

设置边界点坐标 (4000,0,4000)，起点坐标 (0,0,0)，终点坐标 (-4000,0,4000)。图 7.8.75 为 XZ 平面边界点模式圆弧插补的位移曲线。

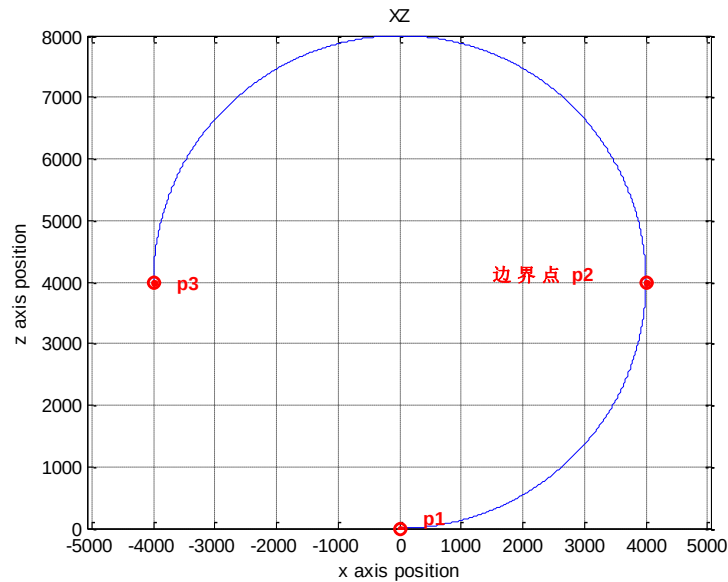


图 7.8.75 边界点模式 XZ 平面圆弧插补

④任意平面

设置边界点坐标 (4000,1000,4000)，起点坐标 (0,0,0)，终点坐标 (2000,3000,4000)。图 7.8.76 为任意平面边界点模式圆弧插补的位移曲线。

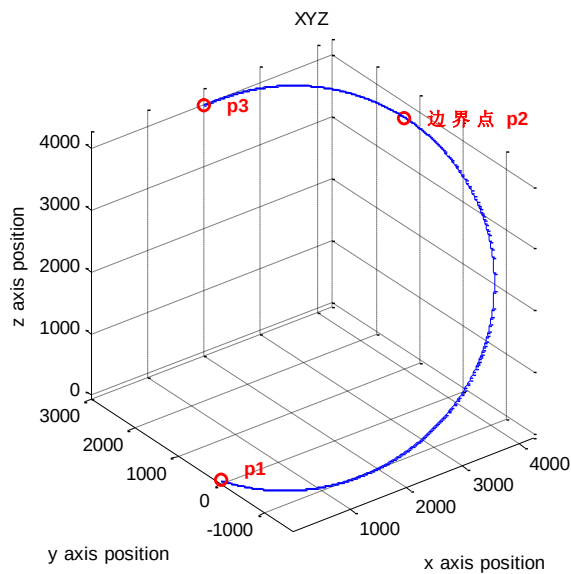


图 7.8.76 边界点模式任意平面圆弧插补

⑤整圆加工

设置边界点坐标 (4000,1000,4000)，起点坐标 (0,0,0)，终点坐标 (0,0,0)，边界点 2 坐标 (2000,3000,4000)。图 7.8.77 为边界点模式整圆插补的位移曲线。

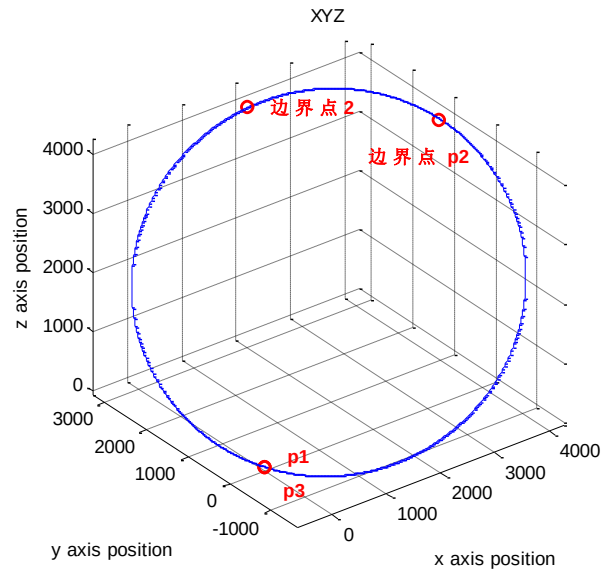


图 7.8.77 边界点模式整圆插补

(3) 半径模式

①XY 平面

设置半径向量坐标 (0,0,4000)，起点坐标 (0,0,0)，终点坐标 (4000,4000,0)。图 7.8.78 为 XY 平面半径模式圆弧插补的位移曲线。

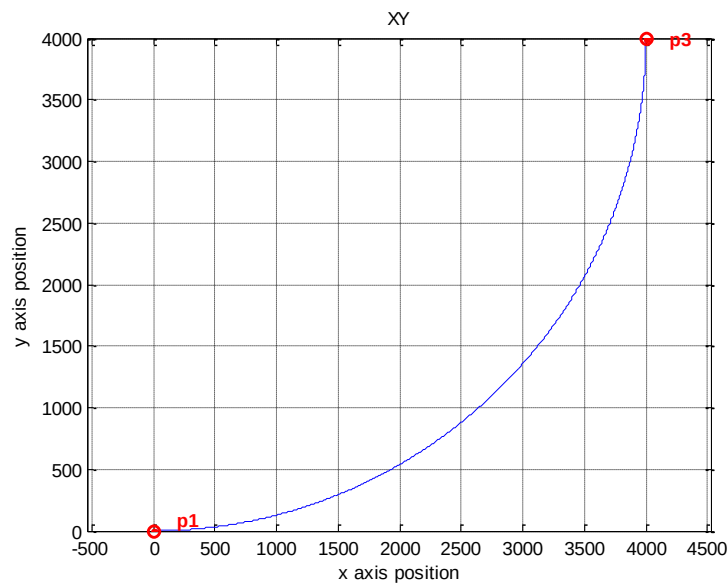


图 7.8.78 半径模式 XY 平面圆弧插补

②YZ 平面

设置半径向量坐标 (4000,0,0)，起点坐标 (0,0,0)，终点坐标 (0,4000,4000)。图 7.8.79 为 YZ 平面半径模式圆弧插补的位移曲线。

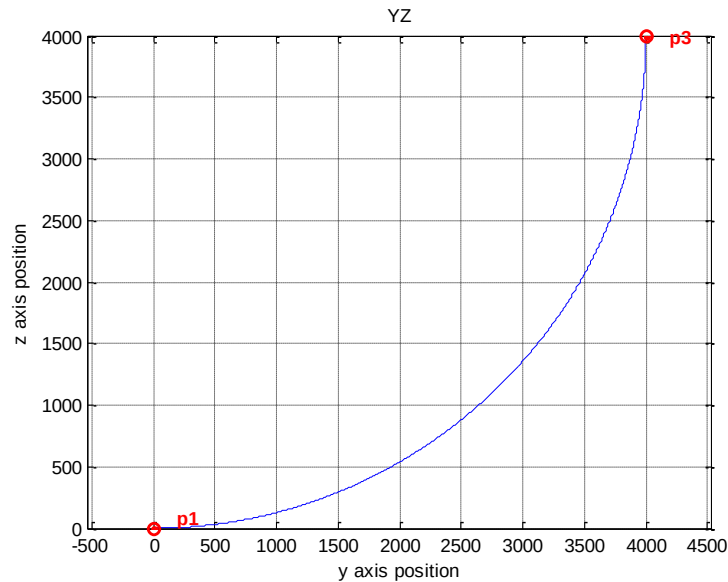


图 7.8.79 半径模式 YZ 平面圆弧插补

③XZ 平面

设置半径向量坐标 (0,4000,0)，起点坐标 (0,0,0)，终点坐标 (4000,0,4000)。图 7.8.80 为 XZ 平面半径模式圆弧插补的位移曲线。

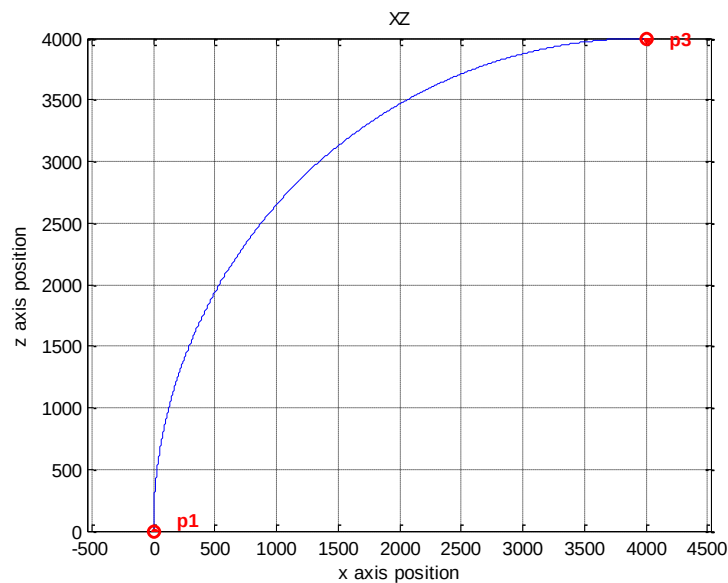


图 7.8.80 半径模式 XZ 平面圆弧插补

④任意平面

设置半径向量坐标 (5000,-2000,1000)，起点坐标 (0,0,0)，终点坐标 (2000,3000,4000)。图 7.8.81 为任意平面半径模式圆弧插补的位移曲线。

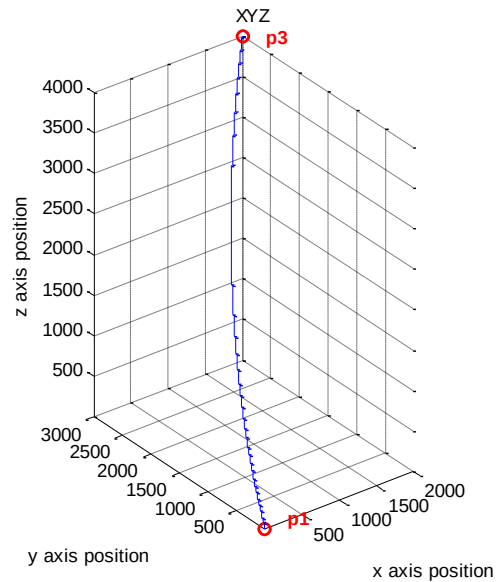


图 7.8.81 半径模式任意平面圆弧插补

⑤整圆加工

设置半径向量坐标 (5000,-2000,1000)，起点坐标 (0,0,0)，终点坐标 (0,0,0)，边界点 2 坐标 (2000,3000,4000)。图 7.8.82 为半径模式整圆插补的位移曲线。

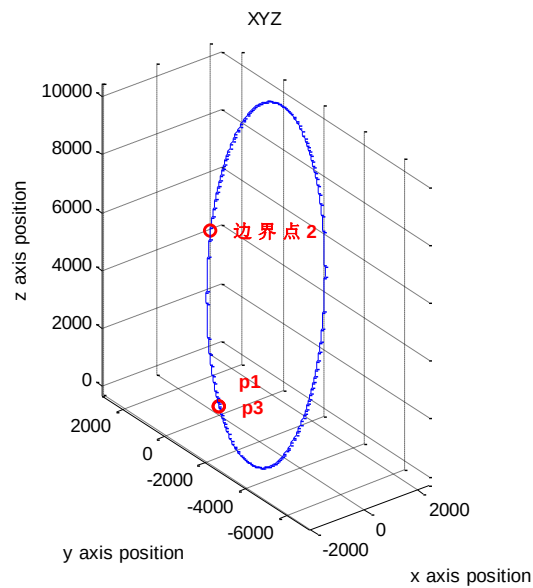


图 7.8.82 半径模式整圆插补

8 重点示例应用

DCCE 控制器提供了工程应用案例，包括 PID 控温程序、电机往复运动程序、系统通信程序、四轴纺纱控制程序、三轴高速肥皂切割机控制程序、直角机器人控制程序和三轴并联机器人控制程序。分别针对控温功能，高速输入输出功能、通信功能和运动控制功能对控制器进行深层次的讲解。

8.1 PID 控温程序示例

PID 调节又称比例-积分-微分反馈调节，PID 调节器一般用于闭环系统中控制输出。控制器通过 AI 传感器将温度、压力、流量等模拟量信号数字化，然后通过 PID 算法计算输出量，循环往复，从而达到精确控制的目的。

- 程序设计：

设计编程时，基于程序逻辑清晰、可读性好、便于管理的原则，采用主程序调用子程序的方式完成 PID 控制。

主程序负责整个控制系统的逻辑构架设计、参数初始化和子程序的衔接调度工作，使程序流程清晰明了；子程序按功能、用途、控制的方式分为数个小框架，每个框架包含相对单一的功能，便于维护管理。PID 例程程序中，主程序共调用两个子程序，子程序 1 配置 PID 指令的参数，子程序 2 进行 PID 控温。

主程序：上电调用子程序，如图 8.1.1 所示。SM0.0 始终为 1，控制器上电时使能两个子程序。

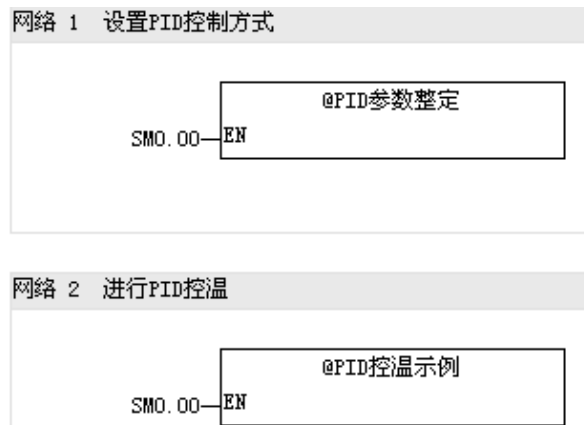


图 8.1.1 主程序

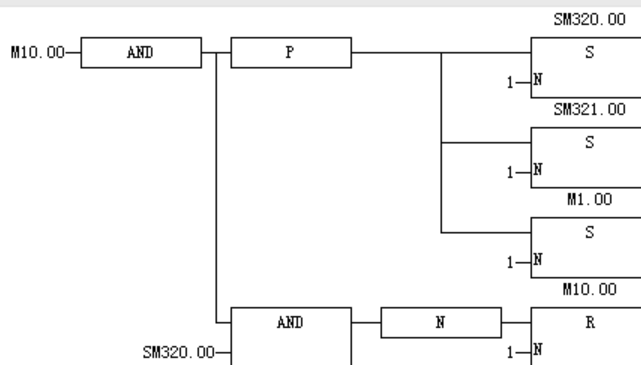
子程序 1：配置 PID 指令的参数。

网络 1 和 2 中，配置第 0 路 PID 指令的参数。其中 SM320.00 为第 0 路 PID 选择标志位；SM321.00 为自整定参数使能标志位；M1.00 为第 0 路 PID 的使能标志位。

在网络 1 中调用上升沿，置位 SM320.00、SM321.00 和 PID 指令的使能标志位 M1.00 开始自整定。自整定结束后，系统会给出自整定参数，同时 SM320.00 会自动复位，通过其下降沿，复位 M10.00。

为了确保下次自整定工作的正常运行，在网络 2 中对自整定功能进行了保护操作，当自整定过程尚未结束时，若手动复位了 M10.00，系统会复位 SM320.00，重新使能自整定。如图 8.1.2 所示。

网络 1 利用继电器反馈法进行PID第0通道参数整定。M10.0参数整定按钮，M1.0为PID功能块的使能标志位。



网络 2 利用继电器反馈法进行PID第0通道参数整定。在参数整定过程中如果想取消，将SM320.0复位

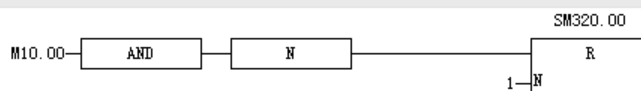
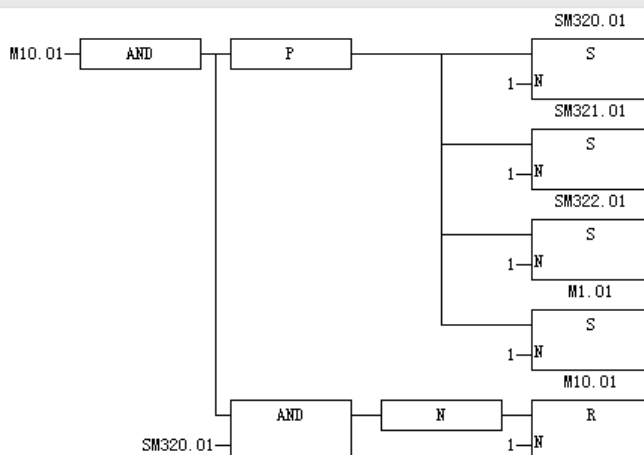


图 8.1.2 子函数 1 的网络 1 和 2

网络 3 和 4 中，配置了第 1 路 PID 指令的参数。与第 0 路不同的是，第 1 路选用了无过冲自整定（飞升曲线法，SMW109 设置为 2）方式，增加了 SM322.01 位的置位，使能无过冲方法。如图 8.1.3。其余与 PID 相关的 SM 区变量，请参见附录 B。

网络 3 利用飞升曲线法进行PID通道2参数整定（1）。增加特殊寄存器SM322.1置位。



网络 4 利用飞升曲线法进行PID第2通道参数整定（2）。在参数整定过程中如果想取消，将SM320.1复位

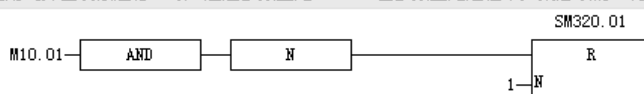


图 8.1.3 子函数 1 的网络 3 和 4

子函数 2：使能 PID 控温。

网络 1 中，把模拟量通道采集的整型温度值转换为实数，单位为 0.1°C 。例如第 0 路 AI 通道采集到的温度为 34.7°C ，则模拟量输入变量 AIW0 的值为 357。PID 指令的设定值 SP、采集值 IN 的引脚需要输入双字型数据，所以需将采集到的整型温度值转换为实数。如图 8.1.4 所示。

网络 1 将通道0的采集温度转换成浮点数（方便计算）

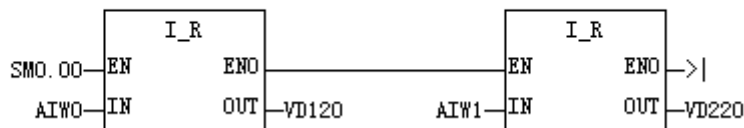


图 8.1.4 子函数 2 的网络 1

网络 2 中，实现第 0 路 PID 的控温操作，如图 8.1.5 所示。PID 指令各引脚详细说明请参见第 5 章节的实数运算指令部分。

注意：

(1) 设定值 SP，起控范围 AREA，采集值 IN 均需要大于零（若采集值为负，请先进行数据处理，使之为正），且均为相同单位的数据；

(2) PID 的输出引脚 PWM_O 只支持 Q 区变量，其余不支持。此输出点仅在 SP 值大于 IN 值时输出，且此引脚上的变量不可以用在程序中的其他输出引脚上。如程序中的 Q0.00 和 Q0.01 由 PID 算法控制，不得在程序内其余的输出引脚上使用，否则会导致 PID 不输出。

网络 2 PID 控温。VD124 用户设定目标温度变量，VD120 当前采集温度变量，Q0.0 控制输出点。

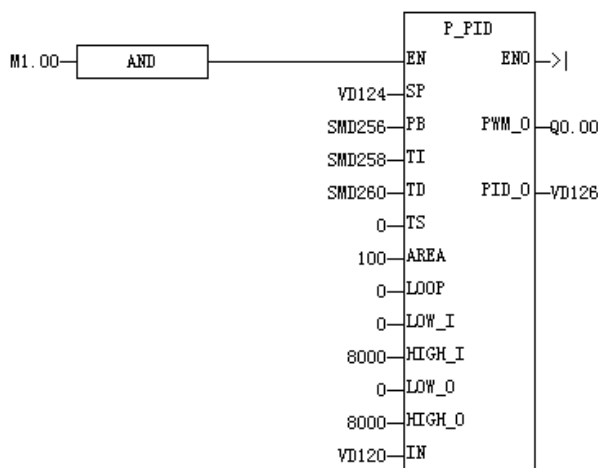


图 8.1.5 子函数 2 的网络 2

网络 3 中，实现第 1 路 PID 的控温操作。LOOP 表示所使用的 PID 通道数，此 LOOP 号不允许冲突，范围为 0~15，一共支持 16 路 PID 功能。如图 8.1.6。

网络 3 注释...

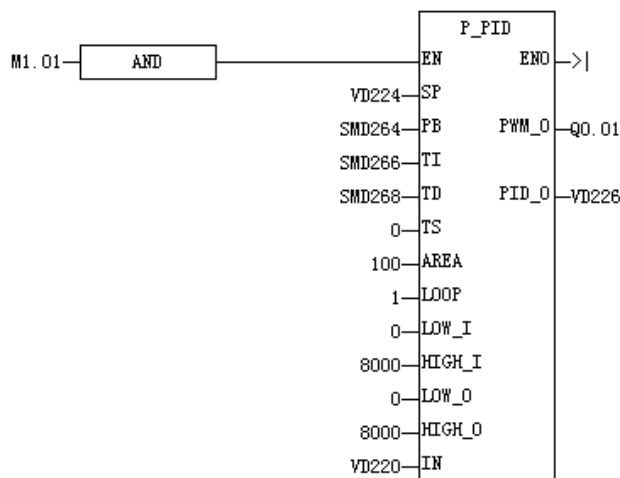


图 8.1.6 子函数 3 网络 3

- 程序功能说明：

例题中，PID 控温程序可达到两路温度的精确控制，并可对两路同时进行 PID 参数自整定。第 0 路为有过冲自整定，整定速度快；第 1 路为无过冲自整定，升降温速度缓慢。

当第一次加热时，程序内使能 M10.00 和 M10.01，运行 PID 功能块，Q0.00、Q0.01 开始输出，进入 PID 自整定阶段。根据现场情况不同，此阶段可能需十几分钟至几十分钟不等；自整定结束后，M10.00 和 M10.01 会自动复位，进入控温阶段，复位 M1.00 和 M1.01 即可停止控温。

8.2 电机往复运动程序示例

控制器可以利用高速数字量输入输出的功能，使用几个简单的指令通过编码器解析电机的实际位移，实现电机的往复运动功能。同时，可以设置电机的转速、位移等参数。

- 程序设计：

电机往复运动程序发送高频脉冲用以驱动电机，并接收由编码器反馈回的高频脉冲。考虑到程序循环，把切换方向写入到中断程序中，实现方向的立即切换，避免软延时。

- 主程序：

在主程序中分别进行了开中断、连接中断、高频脉冲输出以及高频脉冲计数的工作，如图 8.2.1 所示。

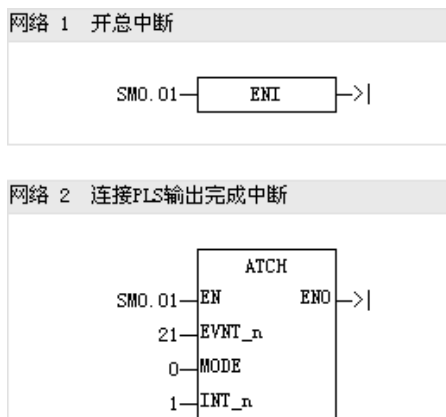


图 8.2.1 开总中断及中断连接

网络 1 中，利用 SMO.01 上电初始化的功能打开总中断；网络 2 中，连接中断事件号为 21、模式为 0 的中断事件，即 PEC8000 的 Q0.07 高速输出完成中断。其余中断事件请参考指令章节中的中断指令部分。

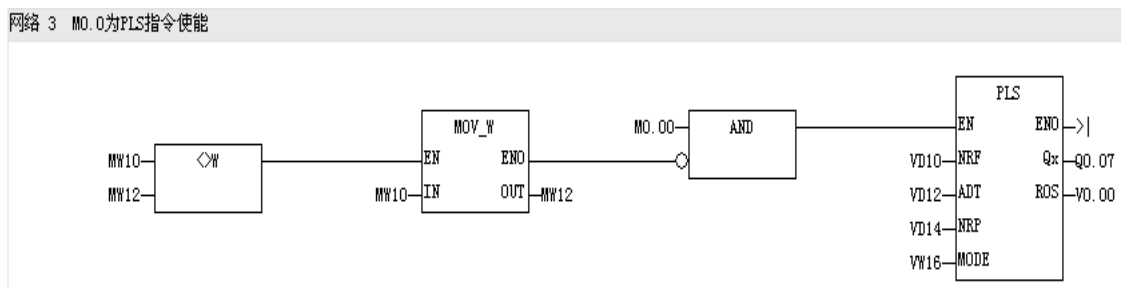


图 8.2.2 PLS 指令

网络 3 中，通过 M0.00 控制 PLS 指令，即电机的运动停止标志位。这里需要注意的是，程序采用高电平使能的 50%固定占空比输出模式，即 VW16 的值为 0，此模式下的速度曲线为梯形。对于每一路高速输出，仅支持一个 PLS 指令，不可重复使用。PLS 指令的详细说明请参考指令章节的计数器指令部分。

网络 4 高速计数

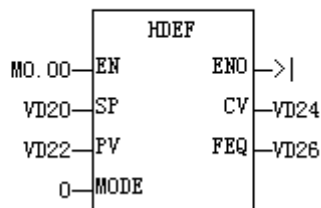


图 8.2.3 HDEF 指令

网络 4 中，利用 HDEF 高速计数指令进行编码器反馈脉冲计数。这里 MODE 配置为 0，即使用 I07 增计数模式。HDEF 指令的详细说明请参考指令章节的计数器指令部分。

- 中断程序

中断程序负责电机运行到位的换向控制，并记录往复次数。如图 8.2.4 所示，其中 Q0.01 为电机方向控制位。

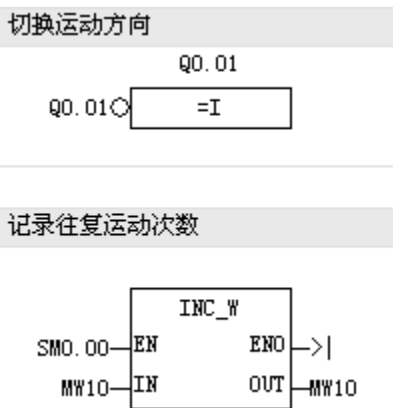


图 8.2.4 中断子函数

- 程序功能说明

电机往复运动程序可控制电机按照设定的脉冲数和频率进行 A、B 两点的往复运动。当使能 M0.00 时，Q0.07 按照 PLS 指令设定的脉冲数和频率输出高频脉冲用以驱动电机运动。当指定脉冲数输出完毕时，进入中断子程序切换方向，同时记录往复次数，重新使能 PLS 功能块，使得电机反向运动，从而实现电机的往复运动功能。

8.3 系统通信程序示例

本小节以简单模型和例子，说明了多级网络、不同协议之间的通信过程。引导客户使用 DCCE 可编程控制器通过以太网交换机与其他设备构成通信系统，进行信息的交互与传递。

- 通信系统搭建：

通信系统的搭建如图 8.3.1 所示，分为工程师站、工作站 1 和工作站 2，可达到下述功能：

- (1) 站内通信：通过标准 MODBUS 总线实现站内数据交互。
- (2) 站间通信：通过以太网通信，实现站间数据的高速交互。
- (3) 总站管理：工程师站通过以太网通信，实时监控多个工作站工作状态。

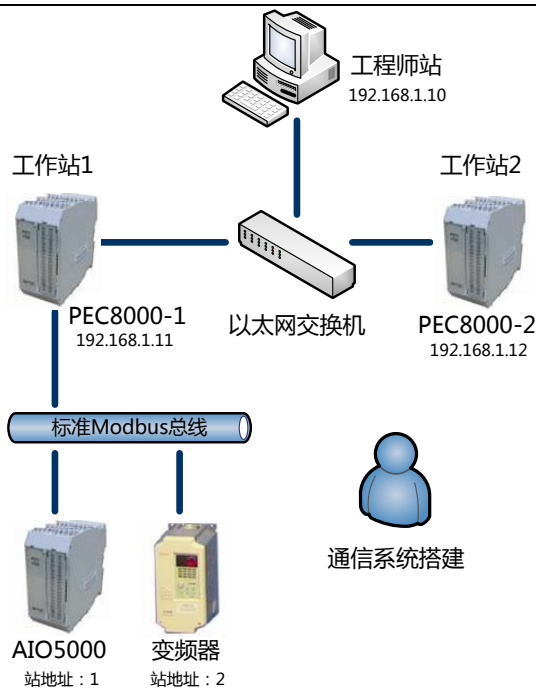


图 8.3.1 通信系统

- 上位机：搭载组态界面，负责整个系统的监视与控制。
- 以太网交换机：数据交换。
- PEC8000-1：一号站主控制器，与上位机和二号站进行以太网通信，与下位机进行 MODBUS 总线通信。
- PEC8000-2：二号站主控制器，与上位机和一号站通过以太网进行数据交互。
- AIO5000：从控制器，作为 PEC8000-1 的从设备，挂在 MODBUS 总线上。
- 变频器：作为 PEC8000-1 的从设备，挂在 MODBUS 总线上。
- 程序设计：

通信例程中应包含串口通信程序、以太网通信程序和自由通信程序。上位机通过 PLC_Config 编程软件或 DVIEW 组态软件同时对工作站 1 和工作站 2 进行编程及监控。

- 以太网通信程序：

工作站 1 与工作站 2 之间，通过以太网交换机，进行以太网通信。首先为各工作站分配 IP 地址，工作站 1 为 192.168.1.11，工作站 2 为 192.168.1.12，确定局域网内无 IP 地址冲突。

然后在设备管理中右键单击主控制器及从设备列表，选择添加设备，填写 IP 地址等以太网信息，单击确定，将控制器添加到设备管理列表中。双击列表内的两个控制器，将其添加到工程内，然后关闭设备管理窗口。此时，工程列表内已经有工作站 1 和工作站 2 的两个 PEC8000，如图 8.3.2 所示：

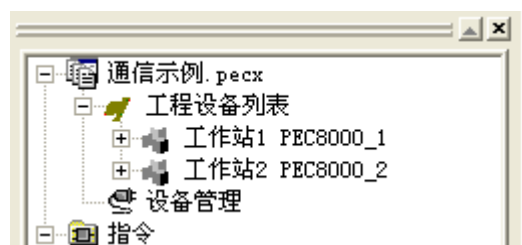


图 8.3.2 添加两个设备

添加控制器后，展开工作站 1 列表，在工作站 1 中编写以太网通信程序，如图 8.3.3 所示。

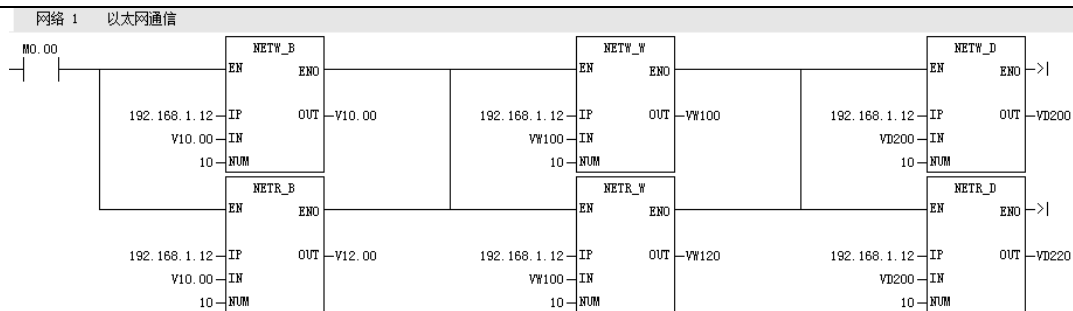


图 8.3.3 以太网通信

此程序负责读写 IP 地址为 192.168.1.12 的工作站 2 中的变量，以上图中的块 NETW_W 和 NETR_W 为例：NETW_W 为 UDP 字写功能块，完成把工作站 1 中 VW100 后连续 10 个字变量的值写入到工作站 2 的 VW100 往后的 10 个字变量中去；NETR_W 为 UDP 字读功能块，完成把工作站 2 中的 VW100 往后连续 10 个变量读到工作站的 VW120 往后的 10 个变量中。其余块为 UDP 位读和双字读功能块，使用方法相同。

同样，若要使工作站 2 读写工作站 1 的变量，可在工程设备列表内展开工作站 2 的列表，在工作站 2 内编写以太网通信程序，读写的 IP 地址为工作站 1 的 192.168.1.11。

● 串口通信程序：

工作站 1 和从设备 AIO5000 及变频器之间通过串口通信。首先为从设备分配 MODBUS 站地址，AIO5000 为 1，变频器为 2。主设备读写从设备物理输入输出可直接寻址，例如工作站 1 的 PEC8000 读写 AIO5000 的 Q0.0 和 AIW0，直接使用映射寄存器 XQ0.0 和 PAIW0 即可。如图 8.3.4 所示：

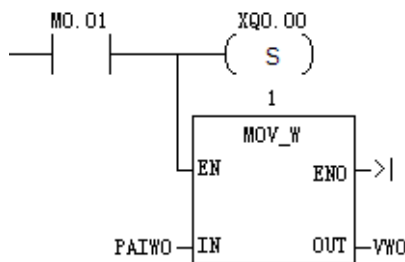


图 8.3.4 主设备直接使用从设备资源

详细的主从通信说明及配置请参见系统管理章节的主从网络通信。

对于读写从设备变量，或与非 DCCE 类其他产品进行通信时，可用串口通信功能实现。首先参见系统管理章节的主从网络通信，在从设备列表内配置从设备。正确配置后，从设备会出现在从设备列表内，如图 8.3.5 所示。

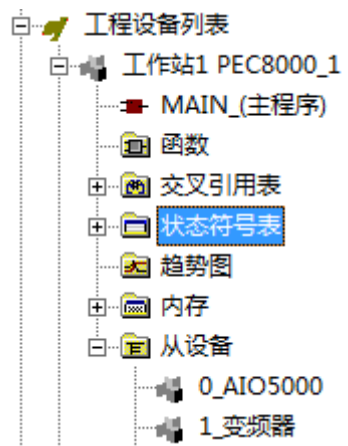


图 8.3.5 配置从设备

然后在主程序内编写与站地址为 1 的 AIO5000 通信程序如图 8.3.6 所示。

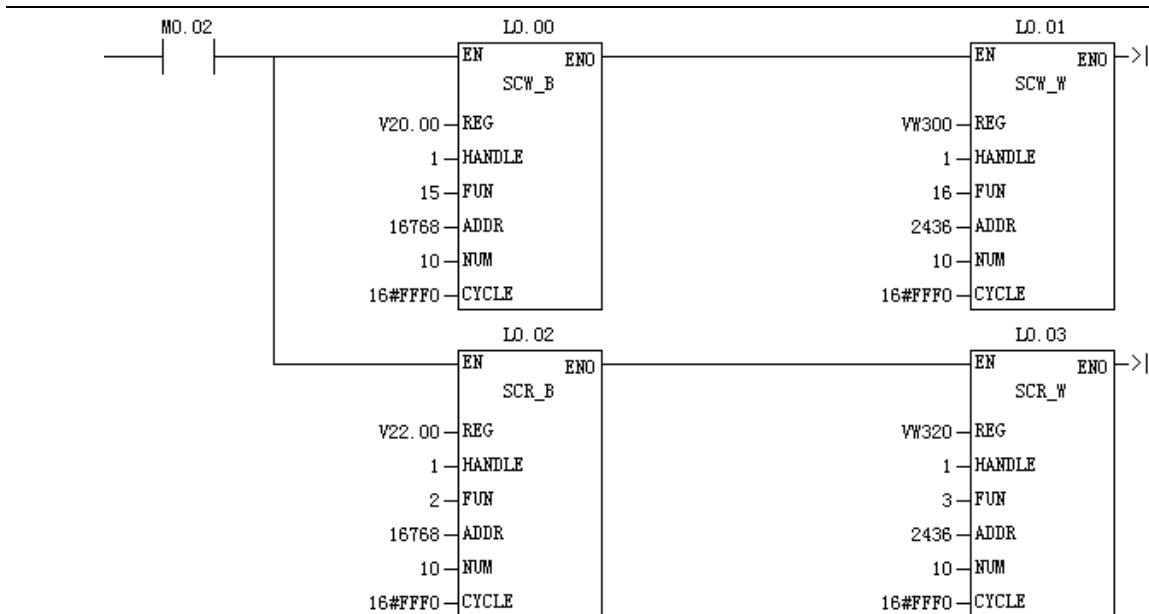


图 8.3.6 串口通信

SCW_B 是串行位写指令，把主设备 PEC8000 从 V20.00 位变量开始的，连续 10 个位变量写入 AIO5000 中 MODBUS 地址为 16768 (V0.0) 开始的连续 10 个位变量。SCR_B 是串行位读指令，负责把上述写入的变量从 AIO5000 读回到 PEC8000 中 V22.00 开始的 10 个位变量中。SCW_W 和 SCR_W 分别为串行字写、字读指令，功能与上述位指令相同，区别为负责读写字变量。头顶参数示意是否完成串口通信，当成功进行一次串口指令读写后，头顶参数置位。CYCLE 引脚为模式选择，CYCLE 为 FFF0，其高 12 位为 1，功能块运行在 Modbus 总线设备映射模式，CYCLE 低 4 位为 0，表示扩展设备连接在当前设备的串口 0。HANDLE 指定了 Modbus 站地址。所以本例中通过串口与变频器通讯时，只需把 HANDLE 号改为变频器站地址 2 即可。

● 转发通信：

对于工作站 2 若要访问工作站 1 的从设备 AIO5000，既可用到转发通信。对于转发通信，4.3 转发通信章节有相应的说明和配置实例，本章节不加详述。

8.4 四轴纺纱控制示例

某纺纱厂采用 PEC6000 控制的气流纺纱简化工程示例，其结构如图 8.4.1 所示，图中有四个电机，分别为 M1（卷绕）、M2（导纱）、M3（引纱）、M4（喂棉），其中 M2 为往复运动，其余均为恒速运动，每路电机对应 PEC6000 的一路高速输出（HQ0~HQ3）。工作流程如下：

- (1) 上电启动时，M2 电机需要进行回原点处理，使梭子在横杆的一端；
- (2) M4 电机启动，延时 100ms 后，M3 电机启动；
- (3) M2、M1 电机同时启动，M2 做往复运动，M1 恒速运动；
- (4) 在需要时，停止所有电机。

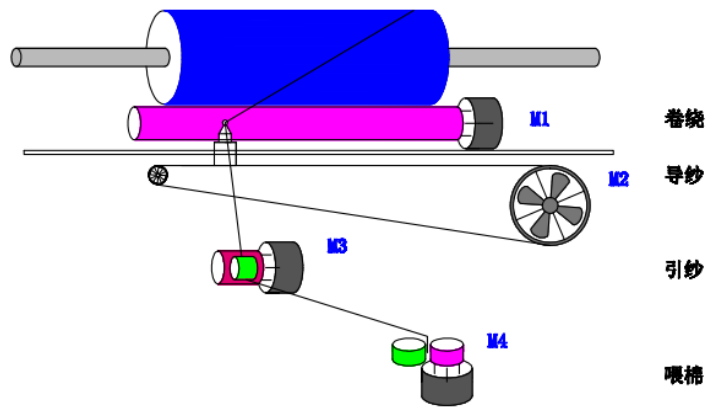


图 8.4.1 纺纱示意图

● 使用的指令

使用指令如表 8.4.1 使用指令所示。

表 8.4.1 使用指令

使用的指令	程序元素分类	说明
MC_MoveVelocity	运动控制指令	恒速运动指令
MC_Reciprocator	运动控制指令	往复运动指令
MC_MoveAbsolute	运动控制指令	绝对运行指令
MC_ReadActualPosition	运动控制指令	读取轴实际位置指令
MC_ReadParameter	运动控制指令	读取参数指令
MC_Stop	运动控制指令	停止轴运动指令
MC_Home	运动控制指令	回原点指令
AND	位逻辑指令	按位与指令
TON	定时器指令	定时器指令
MOV_R	变量转移指令	实数转移指令

● 使用寄存器区

使用变量区如表 8.4.2 使用变量区所示。

表 8.4.2 使用变量区

变量区	说明
SM0.0	始终为高电平全程监视轴的位置和速度
SM0.1	首次上电为高电平，为变量赋值开关
L0.0	回原点开关
L0.1	M3 电机和定时器启动开关
L0.2	M4 电机开关
L0.3	M1、M2 电机开关
L0.5	电机停止开关
VD100	M3 目标速度
VD102	M4 目标速度
VD104	M1 目标速度
VD106	M2 正向位置
VD108	M2 负向位置
VD110	M2 目标速度
VW204	电机运行方向

● 编写示例程序

上电后首先进行运动控制参数配置，所配置参数如图 8.4.2 所示。

回原点动作模式	无接近信号
原点输入信号类型	Z相
回原点开始方向	正向
原点搜索方向	正向
正向限制输入信号ON时的停止动作	反向并立刻停止
负向限制输入信号ON时的停止动作	反向并立刻停止
回原点速度	10000.000000
接近原点速度	1000.000000
回原点加速度	0.000000
回原点减速度	0.000000
回原点加加速度	0.000000
原点位置偏移量	0.000000
原点检测屏蔽距离	10000.000000
原点附近输入信号	I00
原点附近输入信号种类	常开
原点输入信号	I01
原点输入信号种类	常开
正方向限位输入信号	I02
正方向限位输入信号种类	常开
负方向限位输入信号	I03
负方向限位输入信号种类	常开

图 8.4.2 回原点运动控制参数配置

编写程序，使用回原点指令 MC_Home，使 M2 电机回原点，如图 8.4.3 所示。

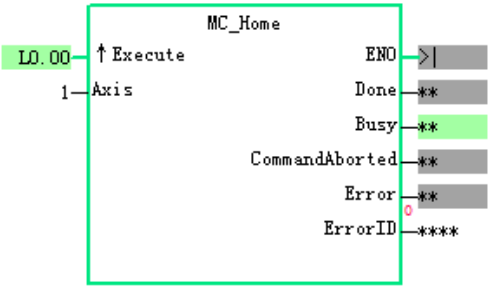
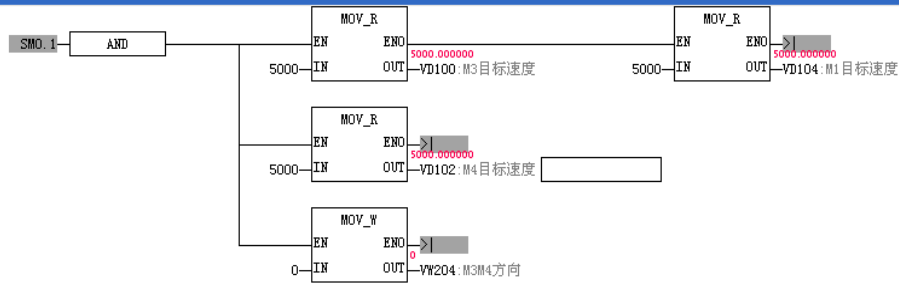


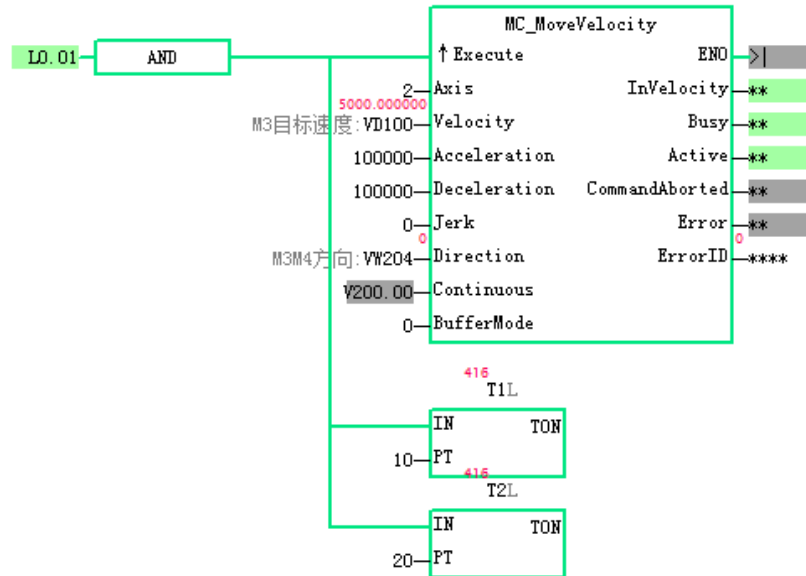
图 8.4.3 回原点

使能 L0.1、L0.2 启动 M3 和 M4 电机，如图 8.4.4 所示。

网络 9 M3/M4电机初始速度和方向



网络 1 注释...



网络 2 注释...

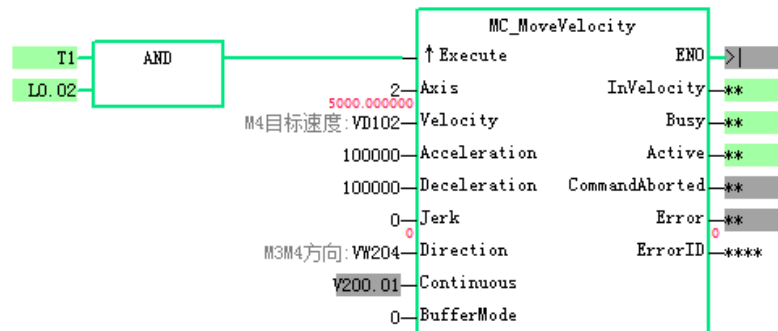
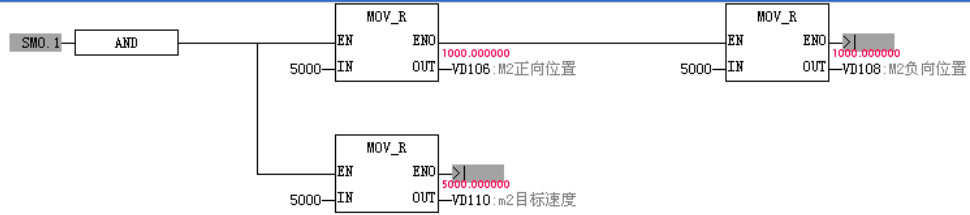


图 8.4.4 启动电机 M3、M4

使能 L0.3 启动 M1 和 M2 电机，如图 8.4.5 所示。

网络 12 M2赋初值



网络 3 M1、M2

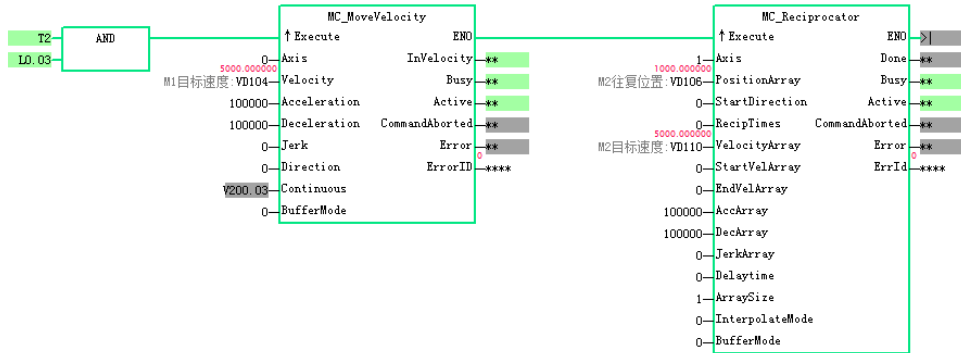


图 8.4.5 启动电机 M1、M2

通过 MC_Stop 指令，停止电机运行，如图 8.4.6 所示。

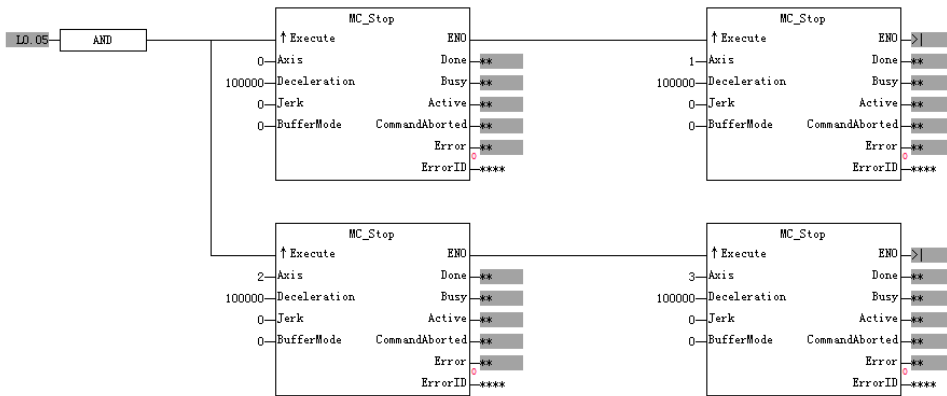


图 8.4.6 停止电机

通过读轴实际位置和速度指令来实时观察运行情况，如图 8.4.7 所示。

网络 15 读取M2实际位置和速度

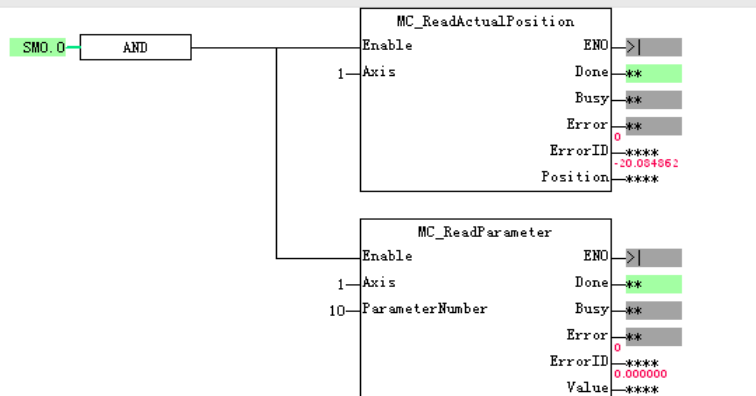


图 8.4.7 运行情况

8.5 三轴高速肥皂切割机控制示例

图 8.5.1 所示为高速肥皂切割机，能够切割任意长度的肥皂，设置好需要切割肥皂的长度，自动切割出高精度均匀等长的肥皂，然后将切割后的肥皂印花。图中轴 1 为主轴，轴 2、轴 3 为从轴，轴 1 为肥皂传送轴，轴 2 为肥皂切割轴，轴 3 为肥皂印花轴。

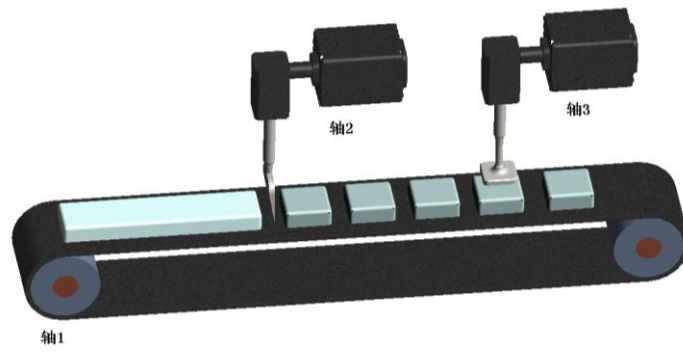


图 8.5.1 高速肥皂切割机示意

- 使用的指令

使用指令如表 8.5.1 使用指令所示。

表 8.5.1 使用指令

使用的指令	程序元素分类	说明
CamTableCreate	运动控制指令	凸轮表创建指令
MC_MoveVelocity	运动控制指令	恒速运动指令
MC_CamIn	运动控制指令	电子齿轮指令
MC_ReadActualPosition	运动控制指令	读取轴实际位置指令
MC_ReadParameter	运动控制指令	读取参数指令

- 使用变量区

使用变量区如表 8.5.2 使用变量区所示。

表 8.5.2 使用变量区

变量区	说明
SM0.0	始终为高电平全程监视轴的位置和速度
M0.0	凸轮表创建指令运行开关
M0.1	凸轮表创建指令运行开关
M0.2	恒速指令运行开关
M0.3	凸轮指令运行开关
M0.4	凸轮指令运行开关
MW13	记录凸轮指令执行凸轮表位置
MW15	记录凸轮指令执行凸轮表位置
VD0	存储主轴与切割轴凸轮表数据
VD100	存储主轴与印花轴凸轮表数据
VD1000	存储主轴实际位置
VD1002	存储主轴实际速度
VD1004	存储切割轴实际位置
VD1006	存储切割轴实际速度
VD1008	存储印花轴实际位置
VD1010	存储印花轴实际速度

- 编写示例程序

- 创建凸轮表

创建 2 张凸轮表，分别为主轴与切割轴之间的凸轮表，如图 8.5.2 和图 8.5.3 所示，主轴与印花轴之间的凸轮表，如图 8.5.4 和图 8.5.5 所示。

VD0	2336	四字节浮点数	0.000000	0
VD2	2338	四字节浮点数	1000.000000	1000
VD4	2340	四字节浮点数	100.000000	100
VD6	2342	四字节浮点数	0.000000	0
VD8	2344	四字节浮点数	1100.000000	1100
VD10	2346	四字节浮点数	1000.000000	1000
VD12	2348	四字节浮点数	1200.000000	1200
VD14	2350	四字节浮点数	0.000000	0
VD16	2352	四字节浮点数	2200.000000	2200
VD18	2354	四字节浮点数	1000.000000	1000
VD20	2356	四字节浮点数	2400.000000	2400
VD22	2358	四字节浮点数	0.000000	0
VD24	2360	四字节浮点数	3400.000000	3400
VD26	2362	四字节浮点数	1000.000000	1000

图 8.5.2 主轴与切割轴凸轮表数据

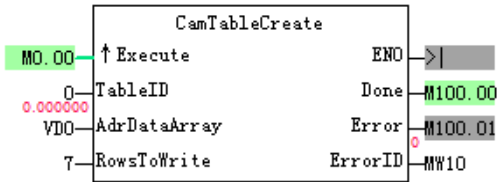


图 8.5.3 程创建主轴与切割轴凸轮表

VD100	2436	四字节浮点数	0.000000	0
VD102	2438	四字节浮点数	500.000000	500
VD104	2440	四字节浮点数	100.000000	100
VD106	2442	四字节浮点数	0.000000	0
VD108	2444	四字节浮点数	1100.000000	1100
VD110	2446	四字节浮点数	500.000000	500
VD112	2448	四字节浮点数	1200.000000	1200
VD114	2450	四字节浮点数	0.000000	0
VD116	2452	四字节浮点数	2200.000000	2200
VD118	2454	四字节浮点数	500.000000	500
VD120	2456	四字节浮点数	2400.000000	2400
VD122	2458	四字节浮点数	0.000000	0
VD124	2460	四字节浮点数	3400.000000	3400
VD126	2462	四字节浮点数	500.000000	500

图 8.5.4 主轴与印花轴凸轮表数据

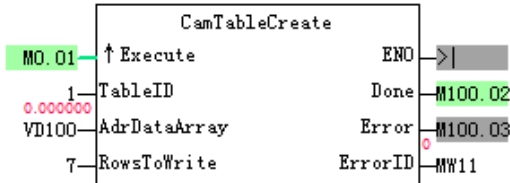


图 8.5.5 创建主轴与印花轴凸轮表

➤ 单轴恒速运行

如图 8.5.6 所示，将 M0.2 置位后使能恒速运动指令，启动主轴恒速运动；

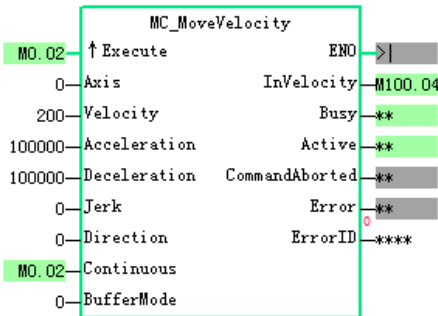


图 8.5.6 主轴恒速运动控制

➤ 主轴与从轴按照凸轮表运行

凸轮耦合指令实现的是让主轴与从轴同时到达相应的目标位置，目标位置由凸轮表给出，如图 8.5.7 所

示，使能 M0.3 主轴与切割轴实现凸轮耦合，如图 8.5.8 所示，使能 M0.4 主轴与印花轴实现凸轮耦合。

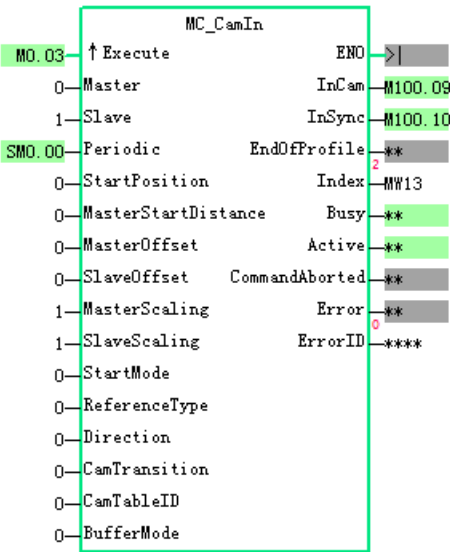


图 8.5.7 主轴与切割轴凸轮耦合控制指令

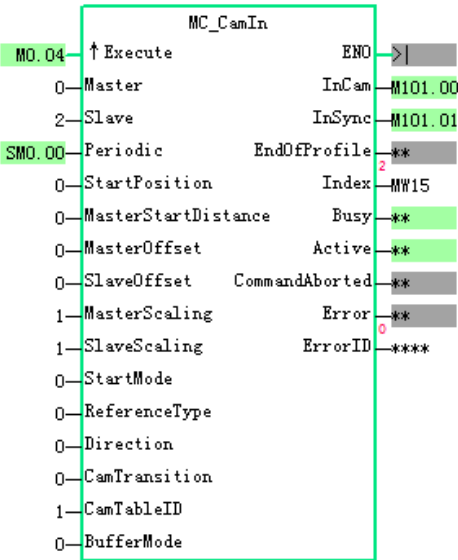


图 8.5.8 主轴与印花轴凸轮耦合控制指令

➤ 轴速度和位置监控

调用 MC_ReadActualPosition 指令和 MC_ReadParameter 指令，读取轴当前的位置和速度，如图 8.5.9、图 8.5.10 和图 8.5.11 所示。

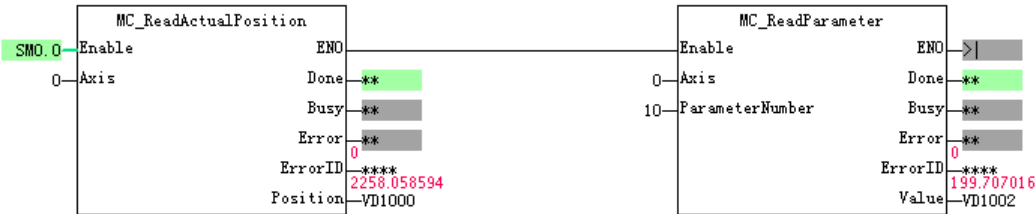


图 8.5.9 主轴速度与位置监控

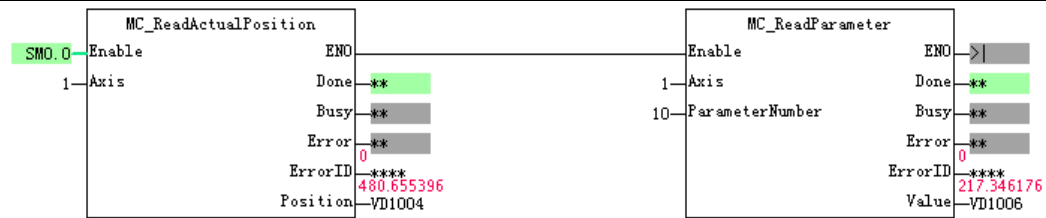


图 8.5.10 切割轴速度与位置监控

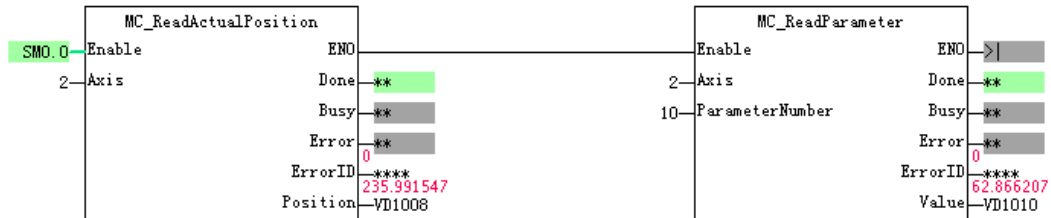


图 8.5.11 切割轴速度与位置监控

8.6 直角机器人编程示例

图 8.6.1 控制一个直角机器人按空间直线运动到指定位置，然后按指定的轨迹在空间中平滑的轨迹运行；同时对轴的速度和位置进行了监视。下面对示例程序分步介绍。

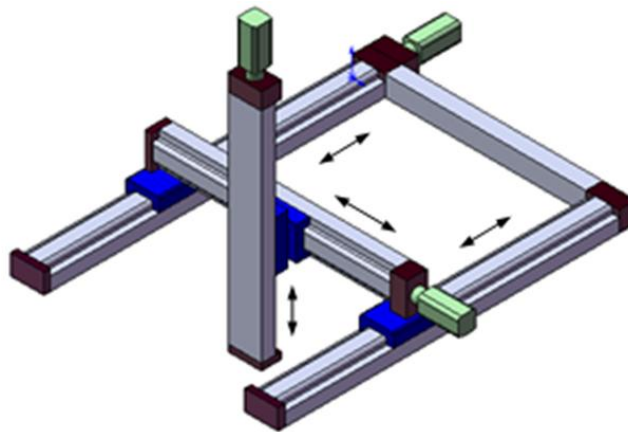


图 8.6.1 直角机器人示意图

● 使用的指令

使用指令如表 8.6.1 使用指令所示。

表 8.6.1 使用指令

使用的指令	程序元素分类	说明
MC_MoveLinearAbsolute	运动控制指令	空间直线运行指令
MC_SetKinTransform	运动控制指令	设置模型及设置轴参数指令
MC_MoveSplineAbsolute	运动控制指令	空间曲线（样条）指令
MC_GroupReadActualPosition	运动控制指令	轴组读取实际位置指令
MC_GroupReadActualVelocity	运动控制指令	轴组读取实际速度指令
AND	位逻辑指令	按位与指令

● 使用变量区

使用变量区如表 8.6.2 使用变量区所示。

表 8.6.2 使用变量区

变量区	说明
SM0.0	始终为高电平，下载程序后立即向轴组中添加 3 个轴，始终读取轴的实际位置和实际速度

M0.0	轴 0 添加成功标志位
M0.1	轴 0 正在添加标志位
M0.2	轴 0 添加错误标志位
M0.3	轴 1 添加能流标志位
M0.4	轴 1 正在添加成功标志位
M0.5	轴 1 正在添加标志位
M0.6	轴 1 添加错误标志位
M0.7	轴 2 添加成功标志位
M0.8	轴 2 正在添加标志位
M0.9	轴 2 添加错误标志位
M0.10	轴 2 添加能流标志位
M1.0	模型设置指令使能位
M1.1	模型设置指令设置成功标志位与空间直线使能位
M1.2	模型设置指令忙标志位
M1.3	模型设置指令正在设置标志位
M1.4	模型设置指令被打断标志位
M1.5	模型设置指令错误标志位
M1.6	模型设置指令能流标志位
M2.1	空间直线能流标志位
M2.2	空间直线运行成功标志位
M2.3	空间直线忙标志位
M2.4	空间直线执行标志位
M2.5	空间直线被打断标志位
M2.6	空间直线错误标志位
M3.0	空间曲线（样条）能流标志位
M3.1	空间曲线（样条）运行成功标志位
M3.2	空间曲线（样条）忙标志位
M3.3	空间曲线（样条）被打断标志位
M3.4	空间曲线（样条）运行错误标志位
M4.0	读取轴组实际位置进行标志位
M4.1	读取轴组实际位置错误标志位
M4.2	读取实际速度能流标志位
M4.3	读取实际速度进行标志位
M4.4	读取实际速度错误标志位
M4.5	读取状态确定标志
MW33	存储轴 0 添加轴组错误参数
MW34	存储轴 1 添加轴组错误参数
MW35	存储轴 2 添加轴组错误参数
MW37	存储模型设置错误参数
MW38	存储空间直线运行错误参数
MW39	存储空间曲线（样条）运行错误参数
MW40	存储读取轴组实际位置时发生的错误参数
MW41	存储读取轴组实际速度时发生的错误参数
VD0	模型设置起始地址
VD10	空间直线（样条）运行位置起始地址
VD100	空间曲线（样条）运行位置起始地址
VD200	空间曲线（样条）运行权值起始地址
VD500	存储空间曲线（样条）轨迹计算时间
VD600	存储空间曲线（样条）运行实际位置
VD700	空间曲线（样条）各个单轴运行速度起始地址
VD800	存储空间曲线（样条）的速度
VD900	存储空间曲线（样条）的目标位置

● 编写示例程序

➤ 轴和轴组参数配置及模型设置

在图 8.6.2 “运动控制参数配置”中添加 3 个轴，1 个轴组。在轴组信息的“是否使用”中选择“TRUE”，选择默认的直角坐标系模型。选择“全部下载”下载所配置的参数。

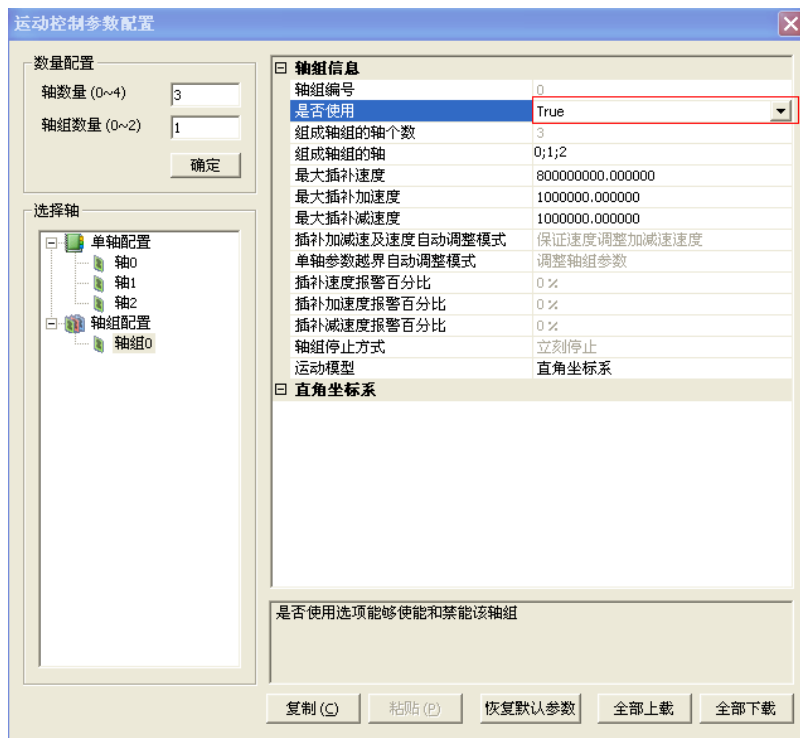


图 8.6.2 轴和轴组参数配置

➤ 空间直线运行

如图 8.6.3 所示，当模型及轴参数设置完成后置位 Done 引脚相应的位变量 M1.1，使能空间直线运行模块，控制直角机器人按直线运动到指定的空间位置。

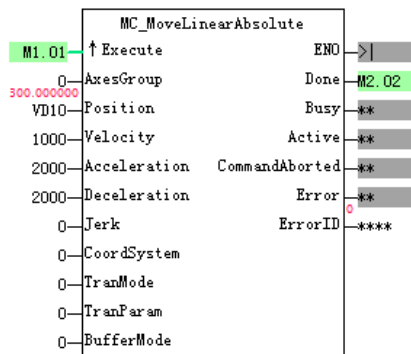


图 8.6.3 空间直线运行

➤ 空间曲线（样条）运动

如图 8.6.4 所示，当空间直线指令运行完成后 Done 引脚变量 M2.2 被置位，使能空间曲线（样条）指令，将控制各个轴按设定的目标值运动。

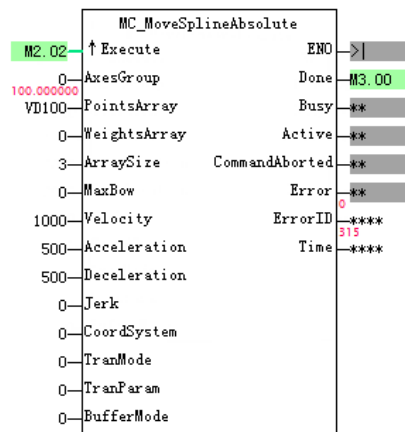


图 8.6.4 空间样条运行

➤ 轴速度和位置监控

如图 8.6.5 所示，将 SM0.0(始终为高电平)置为能流输入，对轴组的合成位置 and 各个轴的位置以及合速度进行监控。

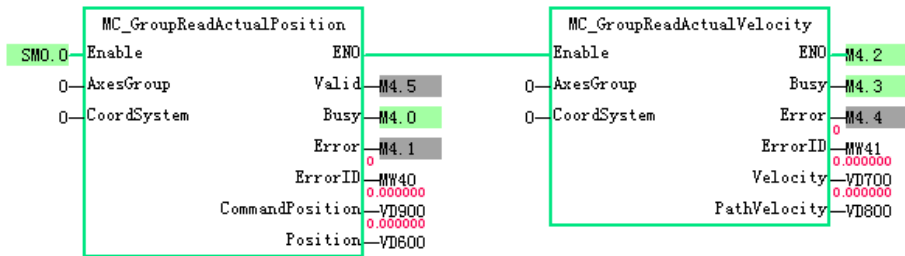


图 8.6.5 参数监控

➤ 此历程的状态符号表

如图 8.6.6 所示，在” 状态符号表” 中，写入的轴组运行目标位置参数表。

11	VD100	暂不支持	四字节浮点数	100.000000	100	曲线运动时第一点x坐标
12	VD102	暂不支持	四字节浮点数	100.000000	100	曲线运动时第一点y坐标
13	VD104	暂不支持	四字节浮点数	0.000000	0	曲线运动时第一点z坐标
14						
15	VD106	暂不支持	四字节浮点数	300.000000	300	曲线运动时第二点x坐标
16	VD108	暂不支持	四字节浮点数	450.000000	450	曲线运动时第二点y坐标
17	VD110	暂不支持	四字节浮点数	350.000000	350	曲线运动时第二点z坐标
18						
19	VD112	暂不支持	四字节浮点数	0.000000	0	曲线运动时第三点x坐标
20	VD114	暂不支持	四字节浮点数	0.000000	0	曲线运动时第三点y坐标
21	VD116	暂不支持	四字节浮点数	0.000000	0	曲线运动时第三点z坐标

图 8.6.6 目标位置参数

8.7 三轴机器人编程示例

本程序功能：通过视频踩点，控制三轴机器人从传送带上抓放货物。

三轴机器人是由电机、机器臂、动平台组成，由电机带动机器臂，从而控制动平台运行，动平台底部有吸盘，可以实现抓取放下货物，模型如图 8.7.1 所示。

当货物在传送带上运行，经过传送带上方的摄像头时，会触发一个光点开关，并计算出三轴动平台的抓货点，而放货点固定，从而确定了动平台轨迹。下面重点介绍动平台的轨迹运动控制。

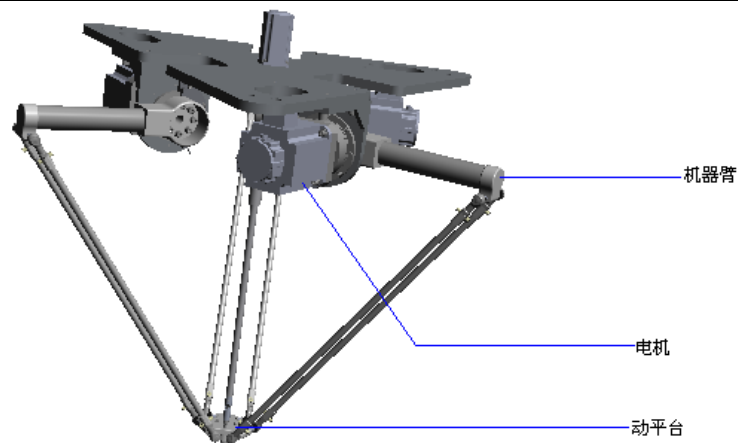


图 8.7.1 三轴机器人示意图

- 使用的指令

使用指令如表 8.7.1 使用指令所示。

表 8.7.1 使用指令

使用的指令	程序元素分类	说明
MC_MoveSplineAbsoluet	运动控制指令	自由样条指令
MOV_R	变量转移指令	实数转移指令
AND	位逻辑指令	与指令
P	位逻辑指令	上升沿触发器
TON	定时器指令	接通延时定时器
S	位逻辑指令	置位指令
SI	位逻辑指令	立即置位指令
R	位逻辑指令	复位指令
RI	位逻辑指令	立即复位指令

- 使用变量区

使用变量区如表 8.7.2 使用变量区所示。

表 8.7.2 使用变量区

变量区	说明
SM0.1	首次上电为高电平，设置抓放货物的轨迹速度模式等值
L10.0	抓放货轨迹赋值标志
L10.1	抓货启动标志
L10.2	抓货轨迹完成标志
L10.3	放货轨迹完成标志
L10.6	视频识别抓货点完成标志
VD500	当前点 x 坐标
VD502	当前点 y 坐标
VD504	当前点 z 坐标
VD506	抓货中间点 x 坐标
VD508	抓货中间点 y 坐标
VD510	抓货中间点 z 坐标位
VD512	抓货点 x 坐标
VD514	抓货点 y 坐标
VD516	抓货点 z 坐标
VD518	放货中间点 x 坐标
VD520	放货中间点 y 坐标
VD522	放货中间点 z 坐标
VD524	放货点 x 坐标
VD526	放货点 y 坐标

VD528	放货点 z 坐标
VD600	轨迹运行速度
VD602	轨迹运行加速度
VD604	轨迹运行减速度
VW606	样条模式选择
VD608	样条执行时间

- 编写示例程序
 - 轴和轴组参数配置及模型设置

如图 8.7.2 所示，在“运动控制参数配置”中添加 3 个轴，1 个轴组。在轴组信息的“是否使用”中选择“TRUE”，选择三轴并联机器人模型。选择“全部下载”下载所配置的参数。

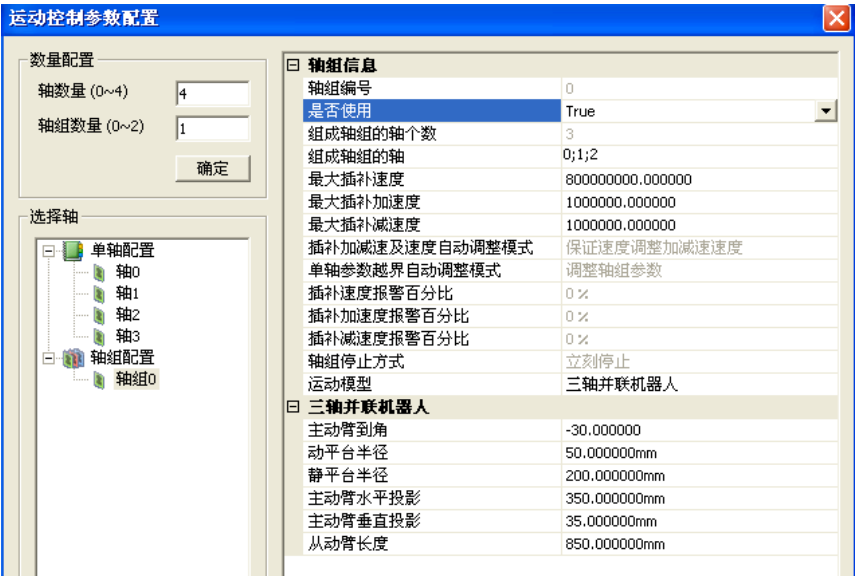


图 8.7.2 三轴并联机器人运动控制参数配置

- 设抓放货轨迹已由摄像头取得，直接赋值

抓获轨迹设置如图 8.7.3 所示。



图 8.7.3 位置赋值

- 启动抓货

抓放货轨迹由自由样条指令来实现，当抓货标志位使能，并且视频获取抓货点完成后，启动抓货行为，如图 8.7.4 所示。

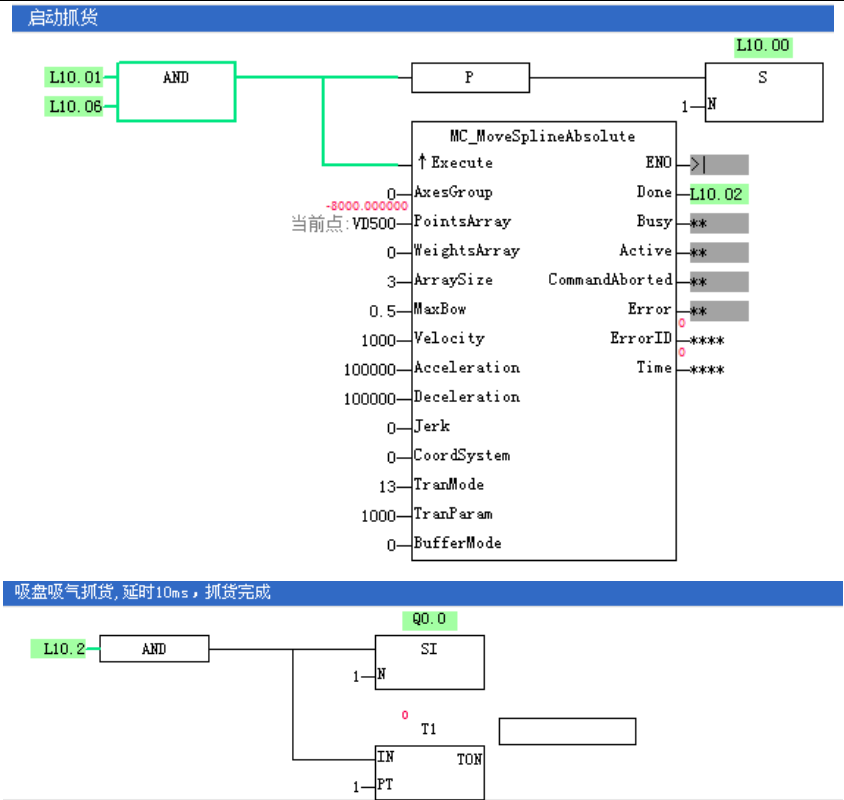


图 8.7.4 启动抓货

➤ 启动放货

启动放货如图 8.7.5 所示。

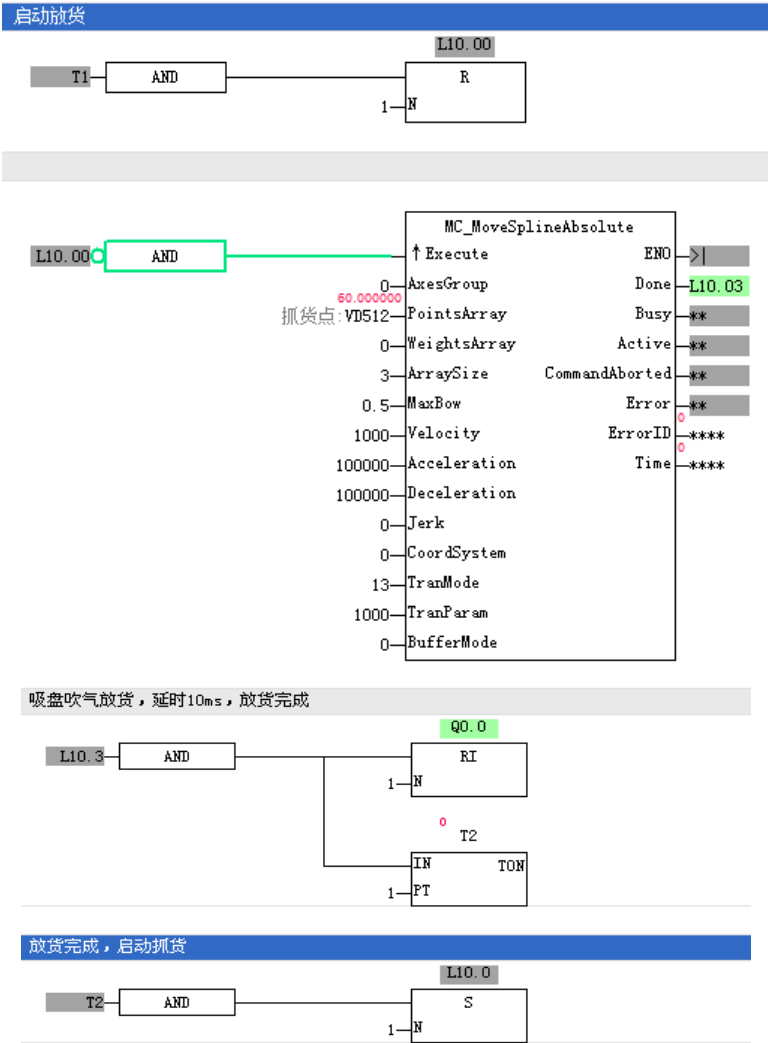


图 8.7.5 启动放货

附录 A 参数设置和说明

表 A-1 产品参数列表和说明

名称	说明	操作
产品序号	产品生产序号，随产品不同	只读
模块地址	外部设备可访问地址，模块地址=硬地址+软地址	只读
软地址	软地址，可由用户进行修改	读写
硬地址	拨码开关对应的地址（部分可选）	只读
IP 地址	以太网访问连接地址	读写
以太网通信端口号	以太网 UDP 方式访问的端口号，建议不要修改	读写
设备名称	设备的名称标识	只读
DI 防抖时间	数字量输入滤波时间，范围为 0~20ms，分辨率为 0.1ms，对所有数字量输入有效	读写
温度补偿	模拟量输入为热电偶时的补偿选择，热电偶时为补偿，其余为不补偿	只读
从设备自恢复扫描控制字	0~15 位分别对应设备编号为 0~15 的从设备，相应位为 1 则此设备下线后对其进行自动恢复扫描	读写
从设备自恢复扫描时间间隔	从设备下线后对其进行扫描的时间间隔，单位为秒	读写
串行通信	主从设置： 选择“主”时表示此串口可以主动发送命令到串行通信总线，用来扩展从设备；选择“从”时表示此串口可以被动的接收通信总线上的命令，用来和人机界面相连； 通信协议：此串口支持的通信协议，可以选择 ModbusRTU 和 ModbusASCII 等	读写
	通信基本设置： 校验选择为无校验、奇校验和偶校验； 数据长度选择 8 位数据位（如果此时有奇偶校验，则第 9 位为奇偶校验位）和 7 位数据位（若 7 位数据位还要带奇偶校验位，则最高位为奇偶校验位）； 1 个停止位和 2 个停止位（2 个停止位时，奇偶校验位无效）；	
	通信波特率： 支持 1200、2400、4800、9600、19200、38400、57600、115200、500K、1M；	
	报文传送间隔时间： 不同波特率下串行通信每个报文传送的间隔时间，一般按照默认值即可。可根据与不同设备通信进行修改，建议不要修改。	
	从设备通信最大允许错误次数： 主口时与从设备连续通信错误次数，超过此值则认为从设备掉线	
	从设备通信时间间隔：主口时与从设备每次通信的时间间隔，单位为 ms	
	从设备通信最大等待时间：主口时与从设备每次通信的最大等待时间，单位为 ms	
AI 通道参数	通道使能：可以选择每个通道的模拟量输入 AI 是否允许转换，默认为全部使能	读写
	工程校准值：每路模拟量输入通道的转换数值加上此数值为实际变量中用户使用的数值	
	传感器类型：每路的模拟量输入传感器类型，只读	
	滤波选择：用于选择模拟量输入是否使能滤波	
	采样速率：用于选择模拟量输入的转换时间	
AQ 输出类型	用于查看模拟量输出的类型，电压、电流或开出（数字量输出）	只读
实时时钟	模块内的实时时钟，年、月、日、时、分、秒、星期均可以设置	读写

表 A-2 可编程控制器、MAC 系列主控器和运动控制器默认参数

名称	默认数值
模块地址	1
软地址	0
硬地址	1
IP 地址	192.168.1.181(DIO、AIO、CIO 及 PMC 系列产品无用)
以太网通信端口号	11000
设备名称	只读，显示设备名称
DI 防抖时间	1ms

从设备自恢复扫描控制字		65535
从设备自恢复扫描时间间隔		10
串行通信 1 号口	主从设置	主
	通信协议	Modbus RTU (MAC 系列默认为 IO 协议)
	通信波特率	115200 (PEC 系列控制器, MAC 系列主控制器) 9600 (其他控制器)
	通信基本设置	N, 8, 1 (无校验, 8 个数据位, 1 个停止位)
	报文传送时间间隔	4
	从设备通信最大允许 错误次数	10
	从设备通信时间间隔	100
	从设备通信最大等待 时间	1000
串行通信 2 号口	主从设置	从
	通信协议	Modbus RTU
	通信波特率	9600
	通信基本设置	N, 8, 1 (无校验, 8 个数据位, 1 个停止位)
	报文传送时间间隔	4
	从设备通信最大允许 错误次数	10
	从设备通信时间间隔	100
	从设备通信最大等待 时间	1000
串行通信 3 号口	主从设置	从
	通信协议	Modbus RTU
	通信波特率	9600
	通信基本设置	N, 8, 1 (无校验, 8 个数据位, 1 个停止位)
	报文传送时间间隔	4
	从设备通信最大允许 错误次数	10
	从设备通信时间间隔	100
	从设备通信最大等待 时间	1000
AI 通道参数	滤波选择	不滤波
	采样速率	15Hz
实时时钟		和当前时间一致
其余参数		随出厂选择不同而不同

附录 B 特殊功能变量区 (SM)

- (1) 特殊变量区提供了状态和控制功能，可以起到帮助用户程序和控制器之间交互信息的作用。
- (2) 标有*号标识的变量用户可在程序中利用指令（例如 MOV_W）进行设置来配合相关功能使用。
- (3) 保留部分的变量区为产品内部使用，请不要使用。

(1) 厂家参数

SMW15: 控制器的 DEVICEID

SM 区	SM 位				
SMW15	设备的 DEVICEID，具体如下				
控制器	DEVICEID	控制器	DEVICEID	控制器	DEVICEID
PEC6000	79	PEC3000	90	DIO3000	91
PEC8000	92	PEC3400	94	PEC5000	95
AIO5000	96	DIO3400	97	PEC9000	101
EDC8000	105	PEC7000	116	EDC3200	120
PEC2000	127	PEC4000	129	PEC6300	132
PEC6800	133	PEC6600	134	PEC6680	135
PEC6900	136	CIO100	145	CIO200	146
MAC1000	158	MAC300	159	MAC1600	160
MAC1800	161	MCD1000	162	MAC1200	163
MAC1100	164	MAC310	165	MAC1610	166
MAC1110	167	MAC1120	168	CIO110	180
EDC3000	190	EDC5000(AO)	191	PMC2000	192
EDC5000(DO)	195	AIO4000	196	--	--

SMW70~SMW74: 设备序号

SM 区	描述
SMD70	无符号四字节，设备生产序号
SMW72	生产日期的年
SMW73	生产日期的月
SMW74	生产日期的日

SMW80: 版本号

SM 区	描述
SMW80	下位机程序内部版本号

SMW81: 功能选择

SM 区	描述
SMW81	0~15 位分别代表产品扩展功能标志位： 0 位为 1 代表具备 ZigBee 功能 1 位为 1 代表具备冗余功能 2 位为 1 代表具备支持 Beckhoff 的 EtherCat 功能 3 位为 1 代表 Profibus 功能 4 位为 1 代表 MPI 功能 5 位为 1 代表液体仪表功能 其余位保留

SMW124: 设备程序版本

SM 区	描述
SMW124	设备程序内部版本号

(2) 设备参数

SMW0: 状态位

SM 位	描述

SM0.0	该位始终为 1，可用于编程时指令使能使用
SM0.1	该位在首次扫描时 1，可用于编程时初始化部分指令或数据使用
SM0.8	指令加密位，1 为用户程序为加密状态，0 为用户程序未加密

SMW1、SMW6、SMW7、SMD167：时基状态位

SM 区	描述
SMW1	16 位无符号整型数，该字每 100ms 自动加一，每一位可以作为一个 100 毫秒的时钟分频标准，溢出后自动复位清零
SMW6	16 位无符号整型数，秒定时器，每秒加 1，溢出后自动清零
SM7.0	秒脉冲，0.5 秒为 1，0.5 秒为 0，周期为 1 秒钟
SM7.1	分钟脉冲，30 秒为 1，30 秒为 0，周期为 1 分钟
SMD167	32 位无符号整型数，毫秒(ms)定时器，每 1ms 加 1，溢出后自动复位清零

SMW52~SMW58：实时时钟状态位

SM 区	描述
SMW52	实时时钟的年
SMW53	实时时钟的月
SMW54	实时时钟的日
SMW55	实时时钟的时
SMW56	实时时钟的分
SMW57	实时时钟的秒
SMW58	实时时钟的星期，1~7 对应星期一到星期日

SMW91：设备内部电池电压监视

SM 区	描述
SMW91	电池电压指示(单位 mv)，每分钟刷新一次

(3) 功能参数

SMW0：状态位

SM 位	描述
SM0.4	该位指示用户程序运行状态，为 1 全速运行，为 0 停止运行
SM0.6	该位为扫描时钟，本次扫描时置 1，下次扫描置零。可用作扫描计数器的输入

SMW2：地址状态位

SM 区	描述
SMW2	模块硬件地址（拨码开关数值）

SMD8、SM10、SMD12：用户程序扫描时间指示

SM 区	描述
SMD8	无符号四字节，记录上一次用户程序扫描执行时间，记录时间单位为 ms
SMD10	无符号四字节，记录用户程序扫描执行的最短时间，记录时间单位为 ms
SMD12	无符号四字节，记录用户程序扫描执行的最长时间，记录时间单位为 ms

SMW14：系统参数读取及功能块下载状态指示

SM 位	描述
SM14.0	设备参数读取错误
SM14.2	控制参数读取错误
SM14.3	描述参数读取错误
SM14.4	系统参数未初始化错误
SM14.8	功能块指令下载第一个指令不是输入指令错误
SM14.9	功能块指令下载最后一个指令不是输出指令、子程序返回指令或中断程序返回指令错误

SM14.10	功能块指令下载出现未定义指令错误
SM14.11	功能块指令下载写入 Flash 错误
SM14.13	功能块指令下载出现控制器不支持指令错误

SMW47: 通信实时与分时处理参数

SM 区	描述
SMW47	通信分时处理时指令运行时间百分比，默认为 50，即指令运行时间占 50%，通信处理时间占 50%

SMW59: 定时器标志位

SM 位	描述
*SM59.0	定时器兼容标志，默认为 0 (1) 为 1: T80~T119 时基为 10ms T1008~T1023 时基为 1ms T0~T79 时基为 100ms T120~T751 时基为 100ms (2) 为 0: T0~T1007 时基为 100ms T1008~T1023 时基为 1ms

SMW68: PEC8000 温度补偿控制标志位

SM 位	描述
*SM68.0	PEC8000/PMC2000 温度补偿控制，为 0 时三路补偿，为 1 时两路补偿

SMW79: 模块地址

SM 区	描述
SMW79	模块软地址

SMW93~SMW100: 模拟量输入工程校准值

SM 区	描述
*SMW93~SMW100	模拟量输入的实际码值加上工程校准值为用户实际使用的数值，例如通道 0 的实际输入值为 234，工程校准值为 10，用户实际使用的 AIW0 变量数值为 244。SMW93 对应通道 0，对应变数 AIW0，SMW94 对应通道 1，对应变数 AIW1，以此类推，SMW100 对应通道 7，对应变数 AIW7

SMW148: ADC 每轮转换时间

SM 区	描述
SMW148	ADC 每一轮（多个通道）的转换时间，单位 ms，PID 使用

SMW149: 冗余系统可配置的系统变量区

SM 区	描述
*SMW149	时钟同步报文发送间隔。用于控制在以太网冗余系统运行过程中，活动设备与备份设备之间的时钟同步报文的发送间隔。默认为 2000 ms

SM164.1: 电池电压报警标示位

SM 位	描述
*SM164.1	电池电压报警标示位，默认为 1: 1 表示电池电压报警功能开启 0 表示电池电压报警功能关闭

SMW164.2: 工作模式切换标志位

SM 位	描述
*SM164.2	工作模式切换标志位，默认为 0:

	1 表示设备工作在安全模式下, 在该模式下, 对内存中各系统参数增加 CRC 校验功能, 增加所有 EPA 报文的 CRC 校验功能, 写系统参数前增加口令写入功能 0 表示设备工作在正常模式下, 兼容了原有状态

SMW198: 功能块指令总数统计

SM 区	描述
SMW198	当前程序中的功能块指令总数

SMW212: 程序延迟运行时间

SM 区	描述
*SMW212	程序延迟运行时间, 单位为秒, 默认为 10 秒

SMW253: 冗余系统可配置的系统变量区

SM 区	描述
*SMW253	功能块同步间隔参数。该参数区负责控制功能块同步的时机, 指定每隔多少轮同步一次功能块的运行。默认值为每轮同步一次功能块运行时间。默认为 0

SMW347~SMW511: 永久保存区

SM 区	描述
*SMW347~SMW511	此段区域作为用户永久保存区使用, 用户可将永久保存的数据放入此段区域, 数据将在 Flash 中保存, 掉电后不丢失 (变化后立即写入 Flash, 数据变化频繁会造成设备 Flash 损坏)

(4) 通信参数

SMW46: 通信实时与分时处理参数

SM 区	描述
SMW46	通信实时处理与分时处理标识, 0 为实时处理通信, 1 为按照时间比处理通信, 此时把指令运行和通信分开处理, 指令运行时间的比例参看 SMW47, 默认为 0

SMW16~SMW17: 从设备监视的保护标识

SM 区	描述
SMW16	从设备不通信时, 串口 0 错误处理次数(只读)
SMW17	从设备不通信时, 串口 1 错误处理次数(只读)

SMW40: 主从通信写服务延时参数

SM 区	描述
SMW40	由于主从通信时写服务有时涉及到写入 Flash 参数, 处理所需时间较长, 所以自动加入最大等待时间延时, 此数值加上设备参数中“从设备通信最大等待时间”为真正的写服务通信最大等待时间, 默认为 400, 单位为 ms。SMW40 为 0 号串口

SMW67: 串口通信错误标示

SM 位	描述
SM67.0	该位为 1 表示串口通信模式错误, 为 0 则无错误
SM67.1	该位为 1 表示 Modbus 内部寻址功能号溢出, 为 0 则无错误
SM67.2	该位为 1 表示 Modbus 读命令通道数量溢出, 为 0 则无错误
SM67.3	该位为 1 表示 Modbus 写单个变量命令地址溢出, 为 0 则无错误
SM67.4	该位为 1 表示 Modbus 写多个变量命令通道数量溢出, 为 0 则无错误
SM67.6	该位为 1 表示 Modbus 扫描从设备命令串行口号溢出, 为 0 则无错误
SM67.10	该位为 1 表示 Modbus 转发错误, 为 0 则无错误
SM67.12	该位为 1 表示 ASCII 到 RTU 转换错误, 为 0 则无错误

SMW89~SMW90: 主口时串口通信错误次数

SM 区	描述
SMW89	1 号串口作为主口时出错次数
SMW90	2 号串口作为主口时出错次数

SMW104、SMW165：以太网连接状态位

SM 位	描述
SM104.0	0 表示自适应状态下以太网口 1 为半双工，1 表示自适应状态下为全双工
SM104.1	0 表示自适应状态下以太网口 1 为 10M，1 表示自适应状态下为 100M
SM104.2	0 表示自适应状态下以太网口 2 为半双工，1 表示自适应状态下为全双工
SM104.3	0 表示自适应状态下以太网口 2 为 10M，1 表示自适应状态下为 100M
*SM165.0~ SM165.1	0 表示为 10M 模式，1 表示为 100M 模式、2 表示自适应模式。默认为自适应模式
SM165.2	1 表示全双工，0 表示半双工，当自适应模式的时候该位无意义

SMW158~SMW159：串行通信收发切换延时

SM 区	描述
*SMW158	串口 0 通信收发切换延时，默认为 3000，最小值为 400。在某些情况下，串口通信配置为高于 115200bps 以上速率时，有可能由于通信的收发切换过快导致没有及时切换到串口的发送状态，引起发送数据包的第一个字节丢失，适当增大此数值可避免此类问题，但同时也会增加串行通信的时间
*SMW159	串口 1 通信收发切换延时，默认为 3000，最小值为 300

SMW179：串行通信指令兼容标志

SM 区	描述
*SMW179	0 表示为串行通信写指令兼容老模式，写之前和写之后不读取所写变量； 1 表示为串行通信写指令新模式，写之前和写之后均读取所写变量；

SMW181：以太网错误间隔时间

SM 区	描述
*SMW181	以太网错误间隔时间(PEC8000)

SMW71、SMW246~SMW248：串口通信参数配置

SM 区	描述
SMW246	串口 2 从设备通信最大错误次数，默认为 10 次
SMW247	串口 2 从设备通信时间间隔，单位为 ms，默认为 100ms
SMW248	串口 2 从设备通信最大等待时间、单位为 ms、默认为 1000ms
SMW71	串口 1 从设备通信最大错误次数

SMW254：设备转发控制寄存器

SM 区	描述
*SMW254	0 设备转发功能关闭； 1 设备处于透明转发状态； 2 设备处于 Modbus 协议转发状态； 3 设备处于 Modbus 地址映射转发状态。

SMW255：转发内部状态寄存器

SM 位	描述
SM255.0	表示转发类型以及转发接口的协议不符（如系统处在透明转发状态下，但是转发表中存在 Modbus 协议转发的表项）
SM255.1	TCP 链接数量超过限制
SM255.2	UDP 链接数量超过限制
SM255.3	表示统一 IP 地址，在表中出现了两次链接（如转发接口配置中，出现了 IP 相同，端口不同的表项）

(5) IO 配置参数

SMW108、SMW194~SMW196：DQ 和 AQ 编程屏蔽状态位

SM 区	描述
*SMW108	DQ、AQ 屏蔽功能选择，1 为屏蔽功能使能，0 为不使能屏蔽功能，默认为屏蔽功能使能
SMW194 SMW195	0~15 位分别代表 Q0.0~Q0.15 在指令输出中是否编程 0~15 位分别代表 Q1.0~Q1.15 在指令输出中是否编程 1 表示在指令输出中有该变量

	0 表示在指令输出中无该变量
SMW196	0~15 位分别代表 AQW0~AQW15 在指令输出中是否编程 1 表示在指令输出中有该变量 0 表示在指令输出中无该变量

SMW125、SMW169~SMW178：输出冻结选择

SM 区	描述
*SMW169	输出冻结是否使能选择、0 为不使能，程序停止运行时按照 SMW170~SMW178 和 SMW125 输出，1 为使能，程序停止运行时保持当前的输出值不变，默认为 0
*SMW170	非冻结模式，停止运行时 QW0 的输出值，默认为 0
*SMW171	非冻结模式，停止运行时 QW1 的输出值，默认为 0
*SMW172	非冻结模式，停止运行时 AQW0 的输出值，默认为 0
*SMW173	非冻结模式，停止运行时 AQW1 的输出值，默认为 0
*SMW174	非冻结模式，停止运行时 AQW2 的输出值，默认为 0
*SMW175	非冻结模式，停止运行时 AQW3 的输出值，默认为 0
*SMW176	非冻结模式，停止运行时 AQW4 的输出值，默认为 0
*SMW177	非冻结模式，停止运行时 AQW5 的输出值，默认为 0
*SMW178	非冻结模式，停止运行时 AQW6 的输出值，默认为 0
*SMW125	非冻结模式，停止运行时 AQW7 的输出值，默认为 0

SMW131~SMW147：模拟量输入通道报警

SM 区	描述
*SMW131/SMW132	有符号 16 位整型数，AI0 通道报警上限/下限值
*SMW133/SMW134	有符号 16 位整型数，AI1 通道报警上限/下限值
*SMW135/SMW136	有符号 16 位整型数，AI2 通道报警上限/下限值
*SMW137/SMW138	有符号 16 位整型数，AI3 通道报警上限/下限值
*SMW139/SMW140	有符号 16 位整型数，AI4 通道报警上限/下限值
*SMW141/SMW142	有符号 16 位整型数，AI5 通道报警上限/下限值
*SMW143/SMW144	有符号 16 位整型数，AI6 通道报警上限/下限值
*SMW145/SMW146	有符号 16 位整型数，AI7 通道报警上限/下限值
SMW147	AI 通道报警状态，1 为报警状态，0 为正常状态， 0~7 位对应 AI0~AI7 通道，其余位保留

SMW214~SMW245：从设备 DQ 和 AQ 配置读取时间间隔

SM 区	描述
*SMD214	双字型，设备编号为 0 的从设备配置为其他设备时，AQ 和 DQ 参数中读取命令时间间隔，单位为秒，默认为 5 秒
*SMD216~SMD244	同上，对应设备编号为 1~15 的从设备，以此类推

SM164.0：PEC8000 的 AQ 输出兼容标志

SM 位	描述
*SM164.0	PEC8000 的 AQ 输出兼容标志位，默认为 0： 1 表示兼容 PEC7000 的 AQ 输出标准，0~4000 对应 0~5V 0 表示不兼容 PEC7000 的 AQ 输出标准，0~5000 对应 0~5V

SMW75~SMW76 高速测周法采样次数

SM 区	描述
SMW75	高速 DI 测周法采样次数，只有 PEC8000/PMC2000/PEC3000/DIO3000/PEC3400 支持
SMW76	普通 DI 测周法采样次数、只有 PEC3000/DIO3000/PEC3400 支持

SMW213：DI 滤波时间

SM 区	描述
*SMW213	防抖时间，单位为 0.1ms，0~200 对应 0~20ms，默认为 1ms

(6) 从设备控制参数

SMW0：状态位

SM 位	描述
SM0.9	从设备全部扫描完毕则该位为 1，否则为 0

SMW14: 系统参数读取及功能块下载状态指示

SM 位	描述
SM14.1	设备参数读取错误

SMW19: 从设备状态位

SM 区	描述
SMW19	0~15 位分别表示设备编号 0~15 的 16 个从设备的状态, 1 为在线, 0 为不在线

SMW42: 从设备通信模式选择

SM 区	描述
SMW42	0 为改进的通信模式, 1 为兼容以前的通信模式, 默认为 0

SMW126: 从设备在线灯板指示灯映射区

SM 区	描述
SMW126	该位为 1 时, 对应的从设备在线灯点亮(拷机时用)

SMW129: 从设备自恢复扫描控制字

SM 区	描述
*SMW129	从设备自动恢复扫描控制位, 0~15 位对应设备编号为 0~15 的从设备, 为 1 则配置后可自动恢复扫描, 为 0 则配置后不能自动恢复扫描

SMW130: 从设备自恢复扫描时间间隔

SM 区	描述
SMW130	从设备自恢复扫描时间间隔、单位为秒

SMW161~SMW163: 从设备转发命令配置

SM 区	描述
SMW161	从设备转发命令保留时间, 单位为秒
SMW162	从设备转发命令刷新时间间隔, 单位为秒
SMW163.0	从设备转发命令是否使能, 1 为使能, 0 为禁能

SMW246~SMW248: 串口 1 通信参数配置

SM 区	描述
SMW247	串口 1 从设备通信时间间隔, 单位为 ms, 默认为 100ms

(7) 指令状态参数

SMW20~SMW38: 高速输出方式 3 状态位

SM 区	描述
第一路高速输出的 SM 区说明 Q0.7(PEC8000)/Q0.0(PEC3000)	
SMD20	本轮中已输出的脉冲个数(只读)
SMW22	S 曲线加/减速时间 Tup/Tdn, 单位为毫秒(只读)
SMW23	S 曲线最大加速度 Amax(只读)
*SMW24	S 曲线加减速段的起始频率, 默认为 2Hz(读写)
*SM25.0	为 1 时, 将 SMD26 的值写入到第一路当前坐标变量中
*SM25.1	为 1 时, 将 SMD36 的值写入到第二路当前坐标变量中
*SMD26	定位指令坐标值
SMW28	PLSE 指令执行结束阈值
第二路高速输出的 SM 区说明 Q0.6(PEC8000)/Q0.1(PEC3000)	
SMD30	本轮中已输出的脉冲个数(只读)
SMW32	S 曲线加/减速时间 Tup/Tdn, 单位为毫秒(只读)
SMW33	S 曲线最大加速度 Amax(只读)
*SMW34	S 曲线加减速段的起始频率, 默认为 2Hz(读写)
*SM35.0	为 1 时, 将 SMD26 的值写入到第一路当前坐标变量中
*SM35.1	为 1 时, 将 SMD36 的值写入到第二路当前坐标变量中
*SMD36	定位指令坐标值

SMW38	PLSE 指令执行结束阈值

SMW63、SMW187～SMW189：高速计数器状态位

SM 区	描述
SM63.0	高速计数器 0 增计数模式大于等于设定值置位，减计数模式小于等于设定值时置位。高速计数器复位时，该位自动清零
SM63.1	高速计数器 0 复位标志位，当该位为 1，HDEF 指令复位，所有相关状态寄存器清零，当该位为 0，则持续计数
SM63.2	高速计数器 0 计数等于预定值则该位置 1，不等于时该位清零
SM63.3	高速计数器 0 计数大于预定值则该位置 1，小于等于时该位清零
SM63.4	高速计数器 1 增计数模式大于等于设定值置位，减计数模式小于等于设定值时置位。高速计数器复位时，该位自动清零
SM63.5	高速计数器 1 复位标志位，当该位为 1，HDEF 指令复位，所有相关状态寄存器清零，当该位为 0，则持续计数
SM63.6	高速计数器 1 计数等于预定值则该位置 1，不等于时该位清零
SM63.7	高速计数器 1 计数大于预定值则该位置 1，小于等于时该位清零
SM63.10	高速计数器 2 计数大于预定值则该位置 1，小于等于时该位清零，复位标志置位时不清零
SM63.11	高速计数器 2 复位标志位，当该位为 1，HDEF 指令复位，计数值恢复为初始值，频率清零，当该位为 0，则持续计数
SM63.14	高速计数器 3 计数大于预定值则该位置 1，小于等于时该位清零，复位标志置位时不清零
SM63.15	高速计数器 3 复位标志位，当该位为 1，HDEF 指令复位，计数值恢复为初始值，频率清零，当该位为 0，则持续计数
*SM63.8	AB 相计数时方向控制位，默认为 0，置位 1 后，A 相和 B 相位置互换，对所有的 AB 相模式同时有效
*SMW187	高速计数器测周法和测频法分割点，默认为 1Hz
*SMW188	高速计数器 0 滤波次数，默认为 10 次滤波，最大为 50 次
*SMW189	高速计数器 1 滤波次数，默认为 10 次滤波，最大为 50 次

注意：PEC8000 对应 DI4、DI5、DI6 和 DI7，PEC3000 对应 DI20、DI21、DI22 和 DI23

SMW43～SMW45、SMW78：WPC 指令控制参数

SM 区	描述
SMW43	第一个 WPC 指令生产线上的工件长度
SMW44	第二个 WPC 指令生产线上的工件长度
SMW45	第一个 WPC 指令计算的生产线总距离
SMW78	第二个 WPC 指令计算的生产线总距离

SMW48～SMW50、SMD65：WPC 指令距离判断与时间判断模式控制

SM 区	描述
SMW48	WPC 指令使用距离判断模式标识，第一位对应第一个 WPC 指令，第二位对应第二个 WPC 指令，其余位保留，相应位为 1 启用距离判断模式，为 0 使用时间判断模式，默认为 0
SMD49	浮点数，第一个 WPC 指令使用距离判断模式时的速度值，单位和 INTVL 一致
SMD65	浮点数，第二个 WPC 指令使用距离判断模式时的速度值，单位和 INTVL 一致

SMW185：WPC 指令使用指示标识

SM 区	描述
*SMW185	WPC 指令第二个指令的使用指示标识，12～15 位表示使用数据区，4～11 位表示使用此数据区的字索引，0～3 位表示使用字的位索引，默认为 0x30F9，即 L15.9。此标志是为了保持和以前兼容使用，强烈建议用户不要修改此数值

SMW186：PS 指令使用指示标识

SM 区	描述
*SMW186	PS 指令第二个指令的使用指示标识，12～15 位表示使用数据区，4～11 位表示使

	用此数据区的字索引，0~3 位表示使用字的位索引，默认为 0x30F1，即 L15.1。 此标志是为了保持和以前兼容使用，强烈建议用户不要修改此数值

SMW180：子程序调用深度越界标志

SM 位	描述
SM180.0	该位为 1：表示子程序 CALL 调用深度超过 16 层

SMW190~SMW193：定时中断

SM 区	描述
SMW190	定时 0 中断定时时间，单位为 10ms
SMW191	定时 1 中断定时时间，单位为 10ms
SMW192	定时 2 中断定时时间，单位为 10ms
SMW193	定时 3 中断定时时间，单位为 10ms

SMW60：RS 触发器状态位

SM 位	描述
SM60.0	该位为 1，SR 触发器工作在不定状态
SM60.1	该位为 1，RS 触发器工作在不定状态

SMW199~SMW211：自由通信指令

SM 区	描述
SM199.0	1 表示 0 端口发送完毕信号
SM199.1	1 表示 1 端口发送完毕信号
SM200.1	1 表示接受信息功能中止——超出最大字符数
SM200.5	1 表示收到结束字符
SM200.8~ SM200.15	8~15 位与 0~7 位意义相同 0~7 位代表端口 0，8~15 位代表端口 1
*SM201.1	是否启用中断信息检测起始接收
*SM201.4	1 表示使用 SMW204 或 SMW205 中的值检测空闲状态
*SM201.5	1 表示使用 SMW203 值检测结束字符，低字节对应端口 0，高字节对应端口 1
*SM201.6	1 表示使用 SMW202 值检测起始字符，低字节对应端口 0，高字节对应端口 1
*SM201.8~ SM201.15	8~15 位与 0~7 位意义相同 0~7 位代表端口 0，8~15 位代表端口 1
*SMW202	0~7 表示端口 0 信息开始字符 8~15 表示端口 1 信息开始字符
*SMW203	0~7 表示端口 0 信息结束字符 8~15 表示端口 1 信息结束字符
*SMW208	端口 0 允许接受的最大字符数（1 到 1000 字节）
*SMW209	端口 1 允许接受的最大字符数（1 到 1000 字节）
*SM210.6	端口 0 使用自由通信协议
*SM210.7	端口 1 使用自由通信协议
*SM210.8	1 表示 0 口启动自由通信被动接收模式
*SM210.9	1 表示 1 口启动自由通信被动接收模式
*SM211.0	1 表示端口 0 的 Recv 功能块 TBL 没有进行正确的配置
*SM211.2	1 表示端口 0 没有使能自用通信格式
*SM211.3	1 表示端口 0 不是主口
*SM211.4	1 表示端口 0 启动模式错误
*SM211.5	1 表示端口 0 终止模式错误
*SM211.8	1 表示端口 1 的 Recv 功能块 TBL 没有进行正确的配置
*SM211.10	1 表示端口 1 没有使能自用通信格式
*SM211.11	1 表示端口 1 不是主口
*SM211.12	1 表示端口 1 启动模式错误
*SM211.13	1 表示端口 1 终止模式错误

SMW61~SMW62: 网络读、写指令错误标示

SM 位	描述
SM61.0	该位为 1 表示调用网络读指令数量溢出, (以太网读指令最大个数为 64 个)
SM61.2	该位为 1 表示网络读指令目标设备 IP 与本设备 IP 冲突
SM61.3	该位为 1 表示网络读指令发送数据错误或失败
SM61.4	该位为 1 表示网络读指令响应数据错误
SM62.0	该位为 1 表示调用网络写指令数量溢出, (以太网读指令最大个数为 64 个)
SM62.2	该位为 1 表示网络写指令目的 IP 与本设备 IP 冲突
SM62.3	该位为 1 表示网络写指令发送数据错误或失败
SM62.4	该位为 1 表示网络写指令响应数据错误

SMW166、SMW182~SMW184: 以太网通信指令控制字

SM 区	描述
*SM166.0~SM166.1	以太网写指令模式选择, 0 为直接写入模式, 1 为写入前和写入后读取验证模式, 2 为兼容 PEC7000 模式, 其余为保留
*SMW182	以太网读指令通信最小间隔时间、单位为 10ms, 默认为 200ms
*SMW183	以太网读指令最大超时时间, 单位为 10ms, 默认为 400ms
*SMW184	以太网写指令通信最小间隔时间, 单位为 10ms, 默认为 1 秒

SMW103、SMW105、SMW106~SMW107、SMW109~SMW120、SMW150~SMW155: PID 状态位

SM 区	描述
*SMW103	飞升曲线法 PID 整定完后控温的变速积分 B 值, 默认为 1, 量纲和设定值相同
*SMW105	整数, 0~7 位对应无过冲自整定初始输出是否为全部输出, 对应位为 0 则从相应的 SMW339~SMW346 去取相应的百分比作为初始输出, 对应位为 1 则初始为全部输出。
*SMW106	0 表示 PID 自整定时整定 P、I、D 共 3 个参数, 1 表示 PID 自整定时整定 P、I 共 2 个参数, D 自动赋值为 0, 默认为使用 3 个参数
*SMW107	0 表示 PID 的 PWM 输出时间时 DQ 最小单位为 100ms, 1 表示 PWM 输出时间时 DQ 最小单位为 10ms、默认为 100ms 输出
*SMW109	自整定时滤波使能位, 1 为使能, 0 为禁止。默认为使能
*SMW110	自整定时滤波次数, 可通过它来消除自整定计算时过零点的抖动, 默认为 3 次
*SMW111	兼容 TAC6000 方式的 PID 指令 DQ 输出 PWM 周期自定义模式
*SMD112	实数, 自整定时 Kp 系数, 默认为 0.6, 建议不要修改
*SMD114	实数, 自整定时 Ti 系数, 默认为 0.5, 建议不要修改
*SMD116	实数, 自整定时 Td 系数, 默认为 0.1, 建议不要修改
*SMD118	实数, 自整定时 TD 值, 默认为 0.5, 建议不要修改
*SMW120	P_PID 公式选择, 0 为可选择使用 SMW152~SMW155 之间的公式选择, 1 为使用正常公式, 默认为 0
*SMD150	实数, 不完全微分公式中的微分系数, 默认为 0.125, 建议不要修改
*SMW152~SMW155	当 SMW120 为 0 时的 P_PID 公式算法选择字, 16 个 PID 中每个 PID 对应 4 位作为算法选择, 其中 SMW152.0~SMW152.3 对应 0 号 PID、SMW152.4~SMW152.7 对应 1 号 PID, 以此类推, SMW155.12~SMW155.15 对应 15 号 PID。目前算法选择为 0 是对应不完全微分公式, 1 为正常公式, 2 为通用公式, 3~15 目前保留。默认均为 0

SMW29、SMW39、SMW51、SMW64、SMW69: PID 控制、显示参数

SM 区	描述
*SMW29	PID 飞升曲线法自整定控制位
*SMW39	0~15 位对应 0~15 号 PID 的 AREA 起控范围参数是否使能, 相应位为 1 则禁止对应编号 PID 的起控范围功能, 为 0 则使能对应编号 PID 的起控范围功能, 默认为 0
SMW51	PID 自整定错误标识
SM64.1	该位为 1 表示 PID 回路选择越界
SM64.2	该位为 1 表示 PID 功能块执行 PWM 调解方式, 该位为 0 表示 PID 功能块当前执

	行 PID 模拟量输出调节方式
SM64.3	该位为 1 表示 PID 的微分项设置出错，微分项设置小于 0
SM64.4	该位为 1 表示 PID 的积分参数设置出错，积分参数设置小于等于 0
SM64.5	该位为 1 表示当前执行回路号的 PID 正处于采样阶段，为 0 表示当前执行的回路号 PID 正处于采样等待阶段
SM64.6	该位为 1 表示 PID 的比例度设置出错，比例度设置为 0
SM64.7	该位为 1 表示 PID 的采样时间设置出错，采样参数设置小于等于 0
SMW69	0~15 对应 0~15 共 16 个 PID 的第一次清除输出值使能标志位，1 为未清除输出，0 为已经清除输出

SMW256~SMW346：PID 自整定参数

SM 区	描述
SMD256	实数，PID0 自整定参数的 PB 值
SMD258	实数，PID0 自整定参数的 TI 值
SMD260	实数，PID0 自整定参数的 TD 值
SMD262	实数，PID0 自整定参数的采样时间值
SMD264~SMD318	实数，PID1~PID7 的自整定参数值，以上述 PID0 类推，每个 PID 占用 4 个实数变量
*SMW320	0~7 位分别代表 0~7 路 PID 自整定的使能位，1 为使能此路 PID 自整定，0 为不使能此路 PID 自整定
*SMW321	0~7 位分别代表 0~7 路 PID 是否使用自整定参数，1 为使用此路 PID 自整定参数，0 为不使用此路 PID 自整定参数
*SMW322	0~7 位分别代表 0~7 路 PID 是否使用飞升曲线法自整定参数，1 为使用此路 PID 飞升曲线法自整定参数，0 为不使用此路 PID 飞升曲线法自整定参数
SMW323~SMW338	SMD323 为 PID0 的飞升曲线法整定出的斜率，以秒为单位，量纲和设定值相同，浮点数存储，SMD325 为 PID1 的飞升曲线法整定出的斜率……以此类推，其中 SMD337 为 PID7 的飞升曲线法整定出的斜率。
SMW339~SMW346	整型数，无过冲自整定初始输出百分比和整定过程中百分比记录值，在 SMW105 的 0~7 位对应位为 0 时取 SMW339~SMW346 相应的数值作为初始输出百分比，0~100 对应 0%~100%，在自整定完毕后根据 Loop 为 0~7 将整定过程中最后实际用到的百分比数值写入对应的变量中以备查看。

- (1) 特殊寄存器区提供了状态和控制功能，可以起到帮助用户程序和控制器之间交互信息的作用。
- (2) 标有*号标识的寄存器用户可在程序中利用指令（例如 MOV_W）进行设置来配合相关功能使用。
- (3) 保留部分的寄存器区为产品内部使用，请不要使用。

SMW0：状态位

SM 区	描述
SM0.0	该位始终为 1，可用于编程时指令使能使用
SM0.1	该位在首次扫描时 1，可用于编程时初始化部分指令或数据使用
SM0.4	该位指示用户程序运行状态，为 1 全速运行，为 0 停止运行
SM0.6	该位为扫描时钟，本次扫描时置 1，下次扫描置零。可用作扫描计数器的输入
SM0.7	移位寄存器指令溢出标志位
SM0.8	指令加密位，1 为用户程序为加密状态，0 为用户程序未加密
SM0.9	从设备全部扫描完毕则该位为 1，否则为 0
SM0.11	开方数为负数或产生错误数学运算的时候，该位置 1
SM0.12	当进行 BCD 码转换的时候，该位置 1

SMW1、SMW6、SMW7、SMD167：时基状态位

SM 区	描述
SMW1	16 位无符号整型数，该字每 100ms 自动加一，每一位可以作为一个 100 毫秒的时钟分频标准，溢出后自动复位清零
SMW6	16 位无符号整型数，代表秒定时器，每秒加 1，溢出后自动清零
SM7.0	秒脉冲，0.5 秒为 1，0.5 秒为 0，周期为 1 秒钟

SM7.1	分钟脉冲, 30 秒为 1, 30 秒为 0, 周期为 1 分钟
SMD167	32 位无符号整型数, 毫秒(ms)定时器, 每 1ms 加 1, 溢出后自动复位清零

SMW5、SMW156、SMW157: 高速输出方式 1 状态位

SM 区	描述
SM5.0	频率值低于下限频率 (NRF<10)
SM5.1	频率和时间设置不正确
SM5.2	脉冲个数设定不正确
*SMW156	高速十级变速输出 Q0.6(PEC8000)/Q0.0(PEC3000)起始频率
*SMW157	高速十级变速输出 Q0.7(PEC8000)/Q0.1(PEC3000)起始频率

SMW14: 系统参数读取及功能块下载状态指示

SM 位	描述
SM14.0	设备参数读取错误
SM14.1	从设备参数读取错误
SM14.2	控制参数读取错误
SM14.3	描述参数读取错误
SM14.4	系统参数未初始化错误
SM14.8	功能块指令下载第一个指令不是输入指令错误
SM14.9	功能块指令下载最后一个指令不是输出指令、子程序返回指令或中断程序返回指令错误
SM14.10	功能块指令下载出现未定义指令错误
SM14.11	功能块指令下载写入 Flash 错误
SM14.13	功能块指令下载出现控制器不支持指令错误
SM14.14	非法指令下载报警错误标志位
SM14.15	P 区非法使用报警标志位

SMW19: 从设备状态位

SM 区	描述
SMW19	0~15 位分别表示设备编号 0~15 的 16 个从设备的状态, 1 为在线, 0 为不在线

SMW42: 从设备通讯模式选择

SM 区	描述
SMW42	0 为改进的通讯模式, 1 为兼容以前的通讯模式, 默认为 0

SMW82~SMW88: PID 相位偏移设置值

SM 区	描述
SMW82~SMW88	SMW82 对应第 0 路的偏移、以此类推.....SMW88 对应第 6 路的偏移, 单位为 10ms

SMW126: 从设备在线灯板指示灯映射区

SM 区	描述
SMW126	该位为 1 时, 对应的从设备在线灯点亮(拷机时用)

SMW129: 从设备自恢复扫描控制字

SM 区	描述
*SMW129	从设备自动恢复扫描控制位, 0~15 位对应设备编号为 0~15 的从设备, 为 1 则配置后可自动恢复扫描, 为 0 则配置后不能自动恢复扫描

SMW130: 从设备自恢复扫描时间间隔

SM 区	描述
SMW130	从设备自恢复扫描时间间隔、单位为秒

SMW161~SMW163: 从设备转发命令配置

SM 位	描述
SMW161	从设备转发命令保留时间, 单位为秒

SMW162	从设备转发命令刷新时间间隔，单位为秒
SM163.0	从设备转发命令是否使能，1 为使能，0 为禁止
SM163.2	P 区写 FLASH 标志位，为 1 写入 FLASH 失败，为 0 写入 FLASH 成功

SM164.1: 电池电压报警标示位

SM 位	描述
*SM164.1	电池电压报警标示位，默认为 1： 1 表示电池电压报警功能开启，此时电池电压若低于 2V，RUN 灯闪烁 0 表示电池电压报警功能关闭
*SM163.1	电池电压电压低的标志为，为 1 表示电池电量低；为 0 表示电压正常

SMW164.2: 工作模式切换标志位

SM 位	描述
*SM164.2	工作模式切换标志位，默认为 0： 1 表示设备工作在安全模式下，在该模式下，对内存中各系统参数增加 CRC 校验功能，增加所有 EPA 报文的 CRC 校验功能，写系统参数前增加口令写入功能 0 表示设备工作在正常模式下，兼容了原有状态

SM164.15、SM163.2: P 区写入标志位

SM 位	描述
*SM164.15	该标志位为 1，P 区的数据保存到 FLASH 中。该标志位为 0，P 区的数据不保存到 FLASH 中
*SM163.2	P 区保存状态标志位，为 1 表示保存失败，为 0 表示保存成功

SMW246~SMW248: 串口 1 通讯参数配置

SM 区	描述
SMW246	串口 1 从设备通讯最大错误次数，默认为 10 次
SMW247	串口 1 从设备通讯时间间隔，单位为 ms，默认为 100ms
SMW248	串口 1 从设备通讯最大等待时间、单位为 ms、默认为 1000ms

关键字：SM 区，特殊寄存器区；

(8) 运动控制参数

SM 区	描述
SMW20	插补周期。默认设置为 3，插补周期为 2ms (1) SMW20=1，插补周期 500us； (2) SMW20=2，插补周期 1ms； (3) SMW20=3，插补周期 2ms； (4) SMW20=4，插补周期 3ms； (5) SMW20=5，插补周期 4ms； (6) SMW20=6，插补周期 5ms； (7) SMW20=7，插补周期 6ms；
SMW21	换向延时时间，单位为 40ns，默认值为 125
SMW22	轴 0~轴 7 高速 DQ 输出模式选择寄存器，默认值为 0。 每 2 位控制一路高速 DQ 的脉冲输出模式（如 SM22.0 和 SM22.1 控制轴 0 高速 DQ 输出模式，SM22.2 和 SM22.3 控制轴 1 高速 DQ 输出模式） (1) 0：脉冲+方向输出 (2) 1：AB 相输出 (3) 2：正反向脉冲输出
SMW23	轴 8~轴 15 高速 DQ 输出模式选择寄存器，默认值为 0，使用方法同 SMW22
SMW3	FPGA 程序版本号

附录 C 从设备资源和特性

- DCCE 网络化可编程控制器的从设备种类包括 DIO1000、DIO2000、DIO3000、TAC3000、TAC6000、DIO3400、AIO5000、PEC3000、PEC3400、PEC5000、PEC8000 和运动控制器等
- DIO1000、DIO2000、TAC3000、TAC6000 目前只支持简单模式映射资源，其余设备支持简单模式和完全模式映射。

(1) DIO1000 和 DIO2000

简单模式下映射的资源如下：

从设备上的 I/O	从设备映射到主设备的变量区
DI0~DI15	XI0.0~XI0.15
DI16~DI23	XI1.0~XI1.7
DQ0~DQ11	XQ0.0~XQ0.11
V0.0~V0.15	XQ2.0~XQ2.15 和 XI2.0~XI2.15

简单模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值	项目	数值
读功能号	3	命令数量	0
写功能号	无	通道参数	无
通道数量	0	命令参数	无

(2) AQ 配置参数

项目	数值	项目	数值
读功能号	3	命令数量	0
写功能号	16	通道参数	无
通道数量	0	命令参数	无

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 0，返回寄存器个数为 24 命令 1 起始变量地址为 32，返回寄存器个数为 16

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	2
命令参数	命令 0 起始变量地址为 24，返回变量个数为 12 命令 1 起始变量地址为 56，返回变量个数为 16

(2) TAC60 简单模式下映射的资源如下

从设备上的 I/O	从设备映射到主设备的变量区
AI0~AI7	PAIW0~PAIW7, XAID0~XAID14
DI0~DI1	XI0.0~XI0.1
DI2	XI0.3
AI0~AI7 的上限报警	XI1.0~XI1.7
AI0~AI7 的下限报警	XI1.8~XI1.15
AI0~AI7 控温使能标志位*	XI2.0~XI2.7
EC8*	XI2.8

总控温使能*	XI2.9
DQ0~DQ8	XQ0.0~XQ0.8
AI0~AI6 控温使能标志位	XQ0.9~XQ0.15
AI7 控温使能标志位	XQ1.0
EC8	XQ1.1
总控温使能	XQ1.2

为保持和以前兼容，TAC6000 的 AI0~AI7 控温使能标志位、EC8 和总控温使能标志位同时映射到 XI 区和 XQ 区。

简单模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	9
命令数量	1
通道参数	通道 0~通道 8 的地址对应 0~8 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 0，返回变量个数为 9

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	8
命令数量	1
通道参数	通道 0~通道 7 的地址对应 512~519 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 512，返回变量个数为 8

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 32，返回变量个数为 4 命令 1 起始变量地址为 16，返回变量个数为 16

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	3
命令参数	命令 0 起始变量地址为 0，返回变量个数为 9 命令 1 起始变量地址为 36，返回变量个数为 9 命令 2 起始变量地址为 48，返回变量个数为 1

(3) TAC3000

简单模式下映射的资源如下：

从设备上的 I/O	从设备映射到主设备的变量区
AI0~AI7	PAIW0~PAIW7, XAID0~XAID14
DI0~DI3	XI0.0~XI0.3
AI0~AI7 的上限报警	XI1.0~XI1.7
AI0~AI7 的下限报警	XI1.8~XI1.15
DQ0~DQ8	XQ0.0~XQ0.8
总控温使能	XQ1.2

简单模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	9
命令数量	1
通道参数	通道 0~通道 8 的地址对应 0~8 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 9

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	0
命令数量	0
通道参数	无
命令参数	无

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 32, 返回变量个数为 4 命令 1 起始变量地址为 16, 返回变量个数为 16

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	3
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 9 命令 1 起始变量地址为 36, 返回变量个数为 9 命令 2 起始变量地址为 48, 返回变量个数为 1

(4) DIO3000

简单模式

映射的资源如下：

从设备上的 I/O	从设备映射到主设备的变量区
DI0~DI15	XI0.0~XI0.15
DI16~DI23	XI1.0~XI1.7
从设备上的 I/O	从设备映射到主设备的变量区
DQ0~DQ15	XQ0.0~XQ0.15

AQ0~AQ1*

PAQW0 和 PAQW1, XAQD0 和 XAQD2

如果 AQ0 配置成 DQ 输出，则映射到 XQ1.0

如果 AQ1 配置成 DQ 输出，则映射到 XQ1.1

简单模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值	项目	数值
读功能号	3	命令数量	0
写功能号	无	通道参数	无
通道数量	0	命令参数	无

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	2
命令数量	1
通道参数	通道 0~通道 1 的地址对应 2064~2065 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064，返回变量个数为 2

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	1
命令参数	命令 0 起始变量地址为 0，返回变量个数为 32

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	1
命令参数	命令 0 起始变量地址为 16704，返回变量个数为 32

● 完全模式

映射的资源如下：

从设备上的 I/O	从设备映射到主设备的变量区
MW0~MW15	PAIW0~PAIW15
DI0~DI15	XI0.0~XI0.15
DI16~DI31	XI1.0~XI1.15
M16.0~M29.15	XI2.0~XI15.15
DQ0~DQ15	XQ0.0~XQ0.15
DQ16~DQ31	XQ1.0~XQ1.15
V14.0~V27.15	XQ2.0~XQ15.15
AQ0~AQ1*	PAQW0 和 PAQW1, XAQD0 和 XAQD2
VW0~VW13	PAQW2~PAQW15, XAQD4~XAQD30

如果 AQ0 配置成 DQ 输出，则映射到 XQ1.0

如果 AQ1 配置成 DQ 输出，则映射到 XQ1.1

完全模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	16
命令数量	1
通道参数	通道 0~通道 15 的地址对应 2848~2863 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2848, 返回变量个数为 16

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	16
命令数量	2
通道参数	通道 0~通道 1 的地址对应 2064~2065 通道 2~通道 15 的地址对应为 2336~2349 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064, 返回变量个数为 2 命令 0 起始变量地址为 2336, 返回变量个数为 14

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 32 命令 0 起始变量地址为 25216, 返回变量个数为 224

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	2
命令参数	命令 0 起始变量地址为 16704, 返回变量个数为 32 命令 0 起始变量地址为 16992, 返回变量个数为 224

(5) DIO3400

● 简单模式

映射的资源如下:

从设备上的 I/O	从设备映射到主设备的变量区
DI0~DI15	XI0.0~XI0.15
DI16~DI23	XI1.0~XI1.7
DQ0~DQ15	XQ0.0~XQ0.15
从设备上的 I/O	从设备映射到主设备的变量区
DQ15~DQ31	XQ1.0~XQ1.15
DQ32~DQ47	XQ2.0~XQ2.15
AQ0~AQ1*	PAQW0 和 PAQW1, XAQD0 和 XAQD2

如果 AQ0 配置成 DQ 输出, 则映射到 XQ2.0

如果 AQ1 配置成 DQ 输出，则映射到 XQ2.1

简单模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值	项目	数值
读功能号	3	命令数量	0
写功能号	无	通道参数	无
通道数量	0	命令参数	无

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	2
命令数量	1
通道参数	通道 0~通道 1 的地址对应 2064~2065 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064，返回变量个数为 2

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	1
命令参数	命令 0 起始变量地址为 0，返回变量个数为 32

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	1
命令参数	命令 0 起始变量地址为 16704，返回变量个数为 48

● 完全模式

映射的资源如下：

从设备上的 I/O	从设备映射到主设备的变量区
MW0~MW15	PAIW0~PAIW15
DI0~DI15	XI0.0~XI0.15
DI16~DI31	XI1.0~XI1.15
M16.0~M29.15	XI2.0~XI15.15
DQ0~DQ15	XQ0.0~XQ0.15
DQ16~DQ31	XQ1.0 和 XQ1.15
DQ32~DQ47	XQ2.0~XQ2.15
V14.0~V27.15	XQ3.0~XQ15.15
AQ0~AQ1*	PAQW0 和 PAQW1, XAQD0 和 XAQD2
VW0~VW13	PAQW2~PAQW15, XAQD4~XAQD30

如果 AQ0 配置成 DQ 输出，则映射到 XQ2.0

如果 AQ1 配置成 DQ 输出，则映射到 XQ2.1

完全模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值

读功能号	3
写功能号	无
通道数量	16
命令数量	1
通道参数	通道 0~通道 15 的地址对应 2848~2863 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2848, 返回变量个数为 16

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	16
命令数量	2
通道参数	通道 0~通道 1 的地址对应 2064~2065 通道 2~通道 15 的地址对应为 2336~2349 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064, 返回变量个数为 2 命令 0 起始变量地址为 2336, 返回变量个数为 14

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 32 命令 0 起始变量地址为 25216, 返回变量个数为 224

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	2
命令参数	命令 0 起始变量地址为 16704, 返回变量个数为 48 命令 0 起始变量地址为 16992, 返回变量个数为 208

(6) PEC3000 和 PEC3400 和运动控制系列产品

● 简单模式

映射的资源如下:

从设备上的 I/O	从设备映射到主设备的变量区
DI0~DI15	XI0.0~XI0.15
DI16~DI31	XI1.0~XI1.15
DQ0~DQ15	XQ0.0~XQ0.15
DQ16~DQ31	XQ1.0~XQ1.15

简单模式下自动配置的从设备参数如下:

(1) AI 配置参数

项目	数值	项目	数值
读功能号	3	命令数量	0

写功能号	无	通道参数	无
通道数量	0	命令参数	无

(2) AQ 配置参数

项目	数值	项目	数值
读功能号	3	命令数量	0
写功能号	16	通道参数	无
通道数量	0	命令参数	无

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	1
命令参数	命令 0 起始变量地址为 0，返回变量个数为 32

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	1
命令参数	命令 0 起始寄存器地址为 16704，返回寄存器个数为 32

● 完全模式

映射的资源如下：

从设备上的 I/O	从设备映射到主设备的变量区
MW0~MW15	PAIW0~PAIW15, XAID0~XAID14
DI0~DI15	XI0.0~XI0.15
DI16~DI31	XI1.0~XI1.15
MW16~MW29	XI2.0~XI15.15
DQ0~DQ15	XQ0.0~XQ0.15
DQ16 和 DQ31	XQ1.0 和 XQ1.15
V16.0~V29.15	XQ2.0~XQ15.15
VW0~VW15	PAQW0~PAQW15, XAQD0~XAQD30

完全模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	16
命令数量	1
通道参数	通道 0~通道 15 的地址对应 2848~2863 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2848，返回变量个数为 16

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	16
命令数量	1
通道参数	通道 0~通道 15 的地址对应 2336~2351 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000

	输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2336, 返回变量个数为 16

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 32 命令 1 起始变量地址为 25216, 返回变量个数为 224

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	2
命令参数	命令 0 起始变量地址为 16704, 返回变量个数为 32 命令 1 起始变量地址为 17024, 返回变量个数为 224

(7) PEC5000 和 AIO5000 和 AIO4000 和 PEC9000

● 简单模式

映射的资源如下:

从设备上的 I/O	从设备映射到主设备的变量区
AI0~AI7	PAIW0~PAIW7, XAID0~XAID14
AQ0~AQ7*	PAQW0~PAQW7, XAQD0~XAQD14
DQ0~DQ15	XQ0.0~XQ0.15
DI0~DI15*	XI0.0~XI0.15

PEC5000、AIO5000 和 AIO4000 的 DQ0 对应物理 DQ, AQ 若配置为 DQ, 则对应 DQ1~DQ8, 故需要映射 DQ0~DQ15 到 XQ, 其中 XQ0.9~XQ0.15 映射为变量。DI0 对应物理 DI, 其余 DI 只映射变量。

简单模式下自动配置的从设备参数如下:

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	8
命令数量	1
通道参数	通道 0~通道 7 的地址对应 0~7 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 8

AIO4000 的物理 AI 只有 4 个通道, 故其余 AI 只映射对应的变量

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	8
命令数量	1
通道参数	通道 0~通道 7 的地址对应 2064~2071 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000

	输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064，返回变量个数为 8

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	1
命令参数	命令 0 起始变量地址为 0，返回变量个数为 16

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	1
命令参数	命令 0 起始变量地址为 16704，返回变量个数为 16

● 完全模式

映射的资源如下：

从设备上的 I/O	从设备映射到主设备的变量区
AI0~AI7	PAIW0~PAIW7, XAID0~XAID14
MW0~MW7	PAIW8~PAIW15, XAID16~XAID30
DI0~DI15	XI0.0~XI0.15
M8.0~M22.15	XI1.0~XI15.15
DQ0~DQ15	XQ0.0~XQ0.15
V8.0~V22.15	XQ1.0~XQ15.15
AQ0~AQ7*	PAQW0~PAQW7、XAQD0 和 XAQD14
VW0~VW7	PAQW8~PAQW15、XAQD16~XAQD30

PEC5000、AIO5000 和 AIO4000 的 AQ 若配置为 DQ，则对应 DQ1~DQ8

完全模式下自动配置的从设备参数如下：

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	16
命令数量	2
通道参数	通道 0~通道 7 的地址对应 2064~2071 通道 8~通道 15 的地址对应 2848~2855 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064，返回变量个数为 8 命令 1 起始变量地址为 2848，返回变量个数为 8

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	16
命令数量	2
通道参数	通道 0~通道 7 的地址对应 2064~2071 通道 8~通道 15 的地址对应 2336~2043

	数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064, 返回变量个数为 8 命令 0 起始变量地址为 2336, 返回变量个数为 8

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 16 命令 1 起始变量地址为 25088, 返回变量个数为 240

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	2
命令参数	命令 0 起始变量地址为 16704, 返回变量个数为 16 命令 1 起始变量地址为 16896, 返回变量个数为 240

(8) PEC8000 和 PMC2000

● 简单模式

映射的资源如下:

从设备上的 I/O	从设备映射到主设备的变量区
AI0~AI7	PAIW0~PAIW7, XAID0~XAID14
DI0~DI7	XI0.0~XI0.7
DQ0~DQ7	XQ0.0~XQ0.7
AQ0~AQ1*	PAQW0 和 PAQW1, XAQD0 和 XAQD2

如果 AQ0 配置成 DQ 输出, 则映射到 XQ0.8, 如果 AQ1 配置成 DQ 输出, 则映射到 XQ0.9

简单模式下自动配置的从设备参数如下:

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	8
命令数量	1
通道参数	通道 0~通道 7 的地址对应 0~7 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 8

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	2
命令数量	1
通道参数	通道 0~通道 1 的地址对应 2064~2065

	数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064, 返回变量个数为 2

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	1
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 16

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	1
命令参数	命令 0 起始变量地址为 16704, 返回变量个数为 16

● 完全模式

映射的资源如下:

从设备 I/O	主设备映射变量区
AI0~AI7	PAIW0~PAIW7, XAID0~XAID14
MW0~MW7	PAIW8~PAIW15, XAID16~XAID30
DI0~DI7	XI0.0~XI0.7
M8.0~M22.15	XI1.0~XI15.15
DQ0~DQ7	XQ0.0~XQ0.7
V14.0~V28.15	XQ1.0~XQ15.15
AQ0~AQ1*	PAQW0 和 PAQW1, XAQD0 和 XAQD2
VW0~VW13	PAQW2~PAQW15, XAQD4~XAQD30

如果 AQ0 配置成 DQ 输出, 则映射到 XQ0.8, 如果 AQ1 配置成 DQ 输出, 则映射到 XQ0.9

完全模式下自动配置的从设备参数如下:

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	16
命令数量	2
通道参数	通道 0~通道 7 的地址对应 0~7 通道 8~通道 15 的地址对应 2848~2855 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 8 命令 1 起始变量地址为 2848, 返回变量个数为 8

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	16
命令数量	2

通道参数	通道 0~通道 1 的地址对应 2064~2065 通道 2~通道 15 的地址对应为 2336~2349 数据类型均为无符号双字节 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 2064, 返回变量个数为 2 命令 0 起始变量地址为 2336, 返回变量个数为 14

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	2
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 16 命令 0 起始变量地址为 25088, 返回变量个数为 240

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	2
命令参数	命令 0 起始变量地址为 16704, 返回变量个数为 16 命令 0 起始变量地址为 16992, 返回变量个数为 240

(9) NCC2000

简单模式配置, 自动进行资源映射, 映射的资源如下:

从设备上的 I/O	从设备映射到主设备的变量区
0~11 共 12 路计时变量	XAID0~XAID22
切丝模式	XAQD0
通道使能	XAQD2
延时时间	XAQD4
定时时间	XAQD6

(1) AI 配置参数

项目	数值
读功能号	3
写功能号	无
通道数量	12
命令数量	1
通道参数	通道 0~通道 11 的地址对应 0, 2, 4...22 数据类型均为浮点数 输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 0, 返回变量个数为 12

(2) AQ 配置参数

项目	数值
读功能号	3
写功能号	16
通道数量	4
命令数量	1
通道参数	通道 0~通道 4 的地址对应 24~27 数据类型均为浮点数

	输入零点和输入满度均为默认-2000 和+8000 输出零点和输出满度均为默认-200 和+800
命令参数	命令 0 起始变量地址为 24，返回变量个数为 4

(3) DI 配置参数

项目	数值
读功能号	1
写功能号	无
命令数量	0
命令参数	无

(4) DQ 配置参数

项目	数值
读功能号	1
写功能号	15
命令数量	0
命令参数	无

附录 D 指令码堆栈描述表

指令码名称	助记符	图示	说明														
装载	LD	LD NewBit <table><tr><th>执行前</th><th>执行后</th></tr><tr><td>Bit0</td><td>N wBit</td></tr><tr><td>Bit1</td><td>Bit0</td></tr><tr><td>.....</td><td>Bit1</td></tr><tr><td>Bit30</td><td>.....</td></tr><tr><td>Bit31</td><td>Bit30</td></tr></table>	执行前	执行后	Bit0	N wBit	Bit1	Bit0		Bit1	Bit30		Bit31	Bit30	将寄存器数值装载入栈顶。 由 LD 衍生出三种指令： LDN 是将寄存器值取反后装入栈顶； LDI 是当输入为物理寄存器区时将输入点值立即装入栈顶； LDNI 是将物理输入点值取反后立即装入栈顶。		
执行前	执行后																
Bit0	N wBit																
Bit1	Bit0																
.....	Bit1																
Bit30																	
Bit31	Bit30																
栈顶压栈	LPS	LPS <table><tr><th>执行前</th><th>执行后</th></tr><tr><td>Bit0</td><td>Bit0</td></tr><tr><td>Bit1</td><td>Bit0</td></tr><tr><td>...</td><td>Bit1</td></tr><tr><td>Bit30</td><td>.....</td></tr><tr><td>Bit31</td><td>Bit30</td></tr></table>	执行前	执行后	Bit0	Bit0	Bit1	Bit0	...	Bit1	Bit30		Bit31	Bit30	堆栈顶位复制，将复制结果压入堆栈		
执行前	执行后																
Bit0	Bit0																
Bit1	Bit0																
...	Bit1																
Bit30																	
Bit31	Bit30																
出栈	LPP	LPP <table><tr><th>执行前</th><th>执行后</th></tr><tr><td>Bit0</td><td>Bit1</td></tr><tr><td>Bit1</td><td>Bit2</td></tr><tr><td>.....</td><td>.....</td></tr><tr><td>Bit3</td><td>Bit31</td></tr><tr><td>Bit31</td><td>0</td></tr></table>	执行前	执行后	Bit0	Bit1	Bit1	Bit2			Bit3	Bit31	Bit31	0	将栈顶位弹出，其余位依次上移		
执行前	执行后																
Bit0	Bit1																
Bit1	Bit2																
.....																	
Bit3	Bit31																
Bit31	0																
次栈顶复制	LRD	LRD <table><tr><th>执行前</th><th>执行后</th></tr><tr><td>Bit0</td><td>Bit1</td></tr><tr><td>Bit1</td><td>Bit1</td></tr><tr><td>.....</td><td>.....</td></tr><tr><td>Bit30</td><td>Bit30</td></tr><tr><td>Bit31</td><td>Bit31</td></tr></table>	执行前	执行后	Bit0	Bit1	Bit1	Bit1			Bit30	Bit30	Bit31	Bit31	堆栈次栈顶位复制，将复制结果放入到栈顶，堆栈各位位置不变		
执行前	执行后																
Bit0	Bit1																
Bit1	Bit1																
.....																	
Bit30	Bit30																
Bit31	Bit31																
栈内位装载	LDS	LDS <table><tr><th>执行前</th><th>执行后</th></tr><tr><td>Bit0</td><td>Bit2</td></tr><tr><td>Bit1</td><td>Bit0</td></tr><tr><td>Bit2</td><td>Bit1</td></tr><tr><td>.....</td><td>Bit2</td></tr><tr><td>Bit30</td><td>.....</td></tr><tr><td>Bit31</td><td>Bit30</td></tr></table>	执行前	执行后	Bit0	Bit2	Bit1	Bit0	Bit2	Bit1		Bit2	Bit30		Bit31	Bit30	堆栈内任意位复制，将复制结果放入到栈顶，其余位依次下移
执行前	执行后																
Bit0	Bit2																
Bit1	Bit0																
Bit2	Bit1																
.....	Bit2																
Bit30																	
Bit31	Bit30																

栈顶装载	LDH	LDH NewBit t <table><tr><td>执行前</td><td>执行后</td></tr><tr><td>Bit0</td><td>NewBit</td></tr><tr><td>Bit1</td><td>Bit2</td></tr><tr><td>.....</td><td>Bit3</td></tr><tr><td>Bit30</td><td>.....</td></tr><tr><td>Bit31</td><td>Bit1</td></tr></table>	执行前	执行后	Bit0	NewBit	Bit1	Bit2		Bit3	Bit30		Bit31	Bit1	将寄存器数值装载入栈顶，将栈顶值替换，栈内其余位位置不变
执行前	执行后														
Bit0	NewBit														
Bit1	Bit2														
.....	Bit3														
Bit30															
Bit31	Bit1														
堆栈与	ALD	ALD <table><tr><td>执行前</td><td>执行后</td></tr><tr><td>Bit0</td><td>Bit0&Bit1</td></tr><tr><td>Bit1</td><td>Bit2</td></tr><tr><td>.....</td><td>.....</td></tr><tr><td>Bit30</td><td>Bit31</td></tr><tr><td>Bit31</td><td>0</td></tr></table>	执行前	执行后	Bit0	Bit0&Bit1	Bit1	Bit2			Bit30	Bit31	Bit31	0	堆栈顶前两位相与操作，将结果放入到栈顶，其余位依次上移
执行前	执行后														
Bit0	Bit0&Bit1														
Bit1	Bit2														
.....															
Bit30	Bit31														
Bit31	0														
堆栈或	OLD	OLD <table><tr><td>执行前</td><td>执行后</td></tr><tr><td>Bit0</td><td>Bit0 Bit1</td></tr><tr><td>Bit1</td><td>Bit2</td></tr><tr><td>.....</td><td>.....</td></tr><tr><td>Bit30</td><td>Bit31</td></tr><tr><td>Bit31</td><td>0</td></tr></table>	执行前	执行后	Bit0	Bit0 Bit1	Bit1	Bit2			Bit30	Bit31	Bit31	0	堆栈顶前两位相或操作，将结果放入到栈顶，其余位依次上移
执行前	执行后														
Bit0	Bit0 Bit1														
Bit1	Bit2														
.....															
Bit30	Bit31														
Bit31	0														
与	A	A NewBit <table><tr><td>执行前</td><td>执行后</td></tr><tr><td>Bit0</td><td>NewBit&Bit0</td></tr><tr><td>Bit1</td><td>Bit1</td></tr><tr><td>...</td><td>.....</td></tr><tr><td>Bit30</td><td>Bit30</td></tr><tr><td>Bit31</td><td>Bit31</td></tr></table>	执行前	执行后	Bit0	NewBit&Bit0	Bit1	Bit1	...		Bit30	Bit30	Bit31	Bit31	将输入位和栈顶寄存器位进行与操作，结果放入到栈顶
执行前	执行后														
Bit0	NewBit&Bit0														
Bit1	Bit1														
...															
Bit30	Bit30														
Bit31	Bit31														
或	O	O NewBit <table><tr><td>执行前</td><td>执行后</td></tr><tr><td>Bit0</td><td>NewBit Bit0</td></tr><tr><td>Bit1</td><td>Bit1</td></tr><tr><td>.....</td><td>.....</td></tr><tr><td>Bit30</td><td>Bit30</td></tr><tr><td>Bit31</td><td>Bit1</td></tr></table>	执行前	执行后	Bit0	NewBit Bit0	Bit1	Bit1			Bit30	Bit30	Bit31	Bit1	将输入位和栈顶寄存器位进行或操作，结果放入到栈顶
执行前	执行后														
Bit0	NewBit Bit0														
Bit1	Bit1														
.....															
Bit30	Bit30														
Bit31	Bit1														

取反	NOT	NOT		将输入位取反，结果放入到栈顶
		执行前	执行后	
		Bit0	!Bit0	
		Bit1	Bit1	
				
		Bit30	Bit30	
		Bit31	Bit31	
赋值	S=			将栈顶的值复制一份到设定寄存器
		执行前	执行后	
		Bit0	NewBit Bit0	
		Bit1	Bit1	
				
		Bit30	Bit30	
		Bit31	Bit31	

附录 E 运动控制错误码

运动控制器在运动控制指令配置及运行过程中，如果出现异常，系统会停止，并输出错误码。运动控制指令提供两个输出参数 Error，ErrorID 对指令的错误信息进行反馈，用户可以根据异常对照表，查找异常。

Error：布尔型，表示在执行过程中是否出现了异常，“1”异常，“0”正常。

ErrorID：整型，发生异常时，对应的错误号，用户可以根据该 ID 号，查表定位到异常位置，当有多个异常发生时，显示最严重的错误信息对应的 ID。

错误类型	错误 ID	错误描述	解决办法
轴错误	4867 (0x1303)	设定速度超过控制器允许的最大值	减小目标速度或者增大软件设置的限位速度，使目标速度小于等于软件设置的限位速度
	4868 (0x1304)	设定加速度超过控制器允许的最大值	减小目标加速度或者增大软件设置的最大加速度，使目标加速度小于等于软件设置的最大加速度
	4870 (0x1306)	设定加加速度超过控制器允许的最大值	减小目标加加速度，使目标加加速度小于等于软件设置的最大加加速度
	4874 (0x130A)	设定减速度超过控制器允许的最大值	减小目标减速度或者增大软件设置的最大减速度，使目标减速度小于等于软件设置的最大减速度
	4875 (0x130B)	轴处于非停止状态不允许执行 MC_WriteParameter、MC_WriteBoolParameter、MC_WriteActualPosition	待轴停止运动后，再执行 MC_WriteParameter、MC_WriteBoolParameter、MC_WriteActualPosition
	4876 (0x130C)	轴未使能	运动控制参数配置中将轴使能
	4877 (0x130D)	设定目标位置超过控制器允许的最大正向位置软件限制或小于负向位置软件限制	修改运动控制参数配置中软件设定的位置限制，使目标位置在正负位置软件限制内
	4878 (0x130E)	执行 MC_GearIn 同步运行时，两轴的方向不一致	修改主轴与从轴的运动方向
	4879 (0x130F)	MC_Phasing 进行相位偏移时主轴 ID 设置错误	正确选择与配置主轴
	4880 (0x1310)	轴状态为禁止	对轴进行使能
	4881 (0x1311)	轴在轴组中，轴组还没停止	待轴组运动完成后，对单轴或者多轴操作
	4882 (0x1312)	轴组使用编码器轴	重新设置轴类型
	4883 (0x1313)	轴运行错误	轴运动到正负限位位置，可以通过 MC_Reset 复位
	4885 (0x1315)	轴运行中限位急停错误	使用 MC_Reset 指令复位轴错误
PLC 参数错误	5376 (0x1500)	轴 ID 不存在或不允许访问	重新设置轴 ID
	5377 (0x1501)	位置设置错误	修改软件设定的位置限制，使目标位置小于等于软件设定的位置限制
	5378 (0x1502)	加速度设置错误	减小目标加速度或者增大软件设置的最大加速度，使目标加速度小于等于软件设置的最大加速度
	5379 (0x1503)	减速度设置错误	减小目标减速度或者增大软件设置的最大减速度，使目标减速度小于等于软件设置的最大减速度
	5380 (0x1504)	加加速度设置错误	减小目标加加速度或者增大软件设置的最大加加速度，使目标加加速度小于等于软件设置的最大加加速度
	5381 (0x1505)	速度设置错误	减小目标减速度或者增大软件设置的最大减速度，使目标减速度小于等于软件设置的最大减速度
	5382 (0x1506)	参数设置错误	修改读或者写参数
	5383 (0x1507)	轴错误（运行中错误）	对轴进行复位

	5388 (0x150C)	重复调用同一个指令	修改指令
参数错误 参数错误	5888 (0x1700)	输入参数错误	修改输入参数
	5889 (0x1701)	相位偏移时参数设置错误	修改输入参数
	5892 (0x1704)	设置样条参数失败	修改样条型值点参数
	5893 (0x1705)	设置直线参数失败	修改轴组直线输入参数
	5894 (0x1706)	2D 平面圆弧, 设置参数错误	修改 2D 平面圆弧输入参数
	5895 (0x1707)	3D 空间圆弧, 设置参数错误	修改 3D 平面圆弧输入参数
	5896 (0x1708)	变量区越界	修改数据长度, 避免变量区越界
轴组参数错误	6400 (0x1900)	轴组 ID 不存在或不允许访问	修改轴组 ID
	6402 (0x1902)	轴组已空	向轴组中添加轴
	6403 (0x1903)	轴组禁能	使能轴组
	6404 (0x1904)	轴组运行报错	对轴组进行复位
	6405 (0x1905)	轴组中轴非停止状态报错	待轴停止运动
	6406 (0x1906)	轴组输入参数错误	修改轴组输出参数
模型设置错误	8192 (0x2000)	模型不存在或未设定	修改轴组模型
	8193 (0x2001)	模型参数错误	修改轴组模型参数
多轴参数错误	6144 (0x1800)	表 ID 设置错误	修改表 ID
	6145 (0x1801)	凸轮表创建错误	修改凸轮表
	6146 (0x1802)	参数错误	修改参数
	6147 (0x1803)	凸轮表行错误	修改凸轮表行参数
	6148 (0x1804)	旋切长度设置错误	修改旋切长度
	6149 (0x1805)	同步区起始位置设置错误	修改同步区位置
	6150 (0x1806)	循环轴超出最大位置值	修改循环轴位置值

附录 F 运动控制参数监控寄存器

运动控制器参数监控寄存器，主要针对轴和轴组的运动参数、运行状态和异常信息进行监控。单轴运动的监控参数如下表。

描述	数据类型	寄存器			
		轴 0	轴 1	轴 2	轴 3
非使能状态位, TRUE 表示非使能状态, FALSE 表示使能状态	BOOL	AS0.0	AS32.0	AS64.0	AS96.0
停止状态位, TRUE 表示停止状态, FALSE 表示非停止状态	BOOL	AS0.1	AS32.1	AS64.1	AS96.1
离散运动状态位, TRUE 表示离散运动状态, FALSE 表示非离散运动状态	BOOL	AS0.2	AS32.2	AS64.2	AS96.2
连续运动状态位, TRUE 表示连续运动状态, FALSE 表示非连续运动状态	BOOL	AS0.3	AS32.3	AS64.3	AS96.3
同步运动状态位, TRUE 表示同步运动状态, FALSE 表示非同步运动状态	BOOL	AS0.4	AS32.4	AS64.4	AS96.4
回原点状态位, TRUE 表示回原点状态, FALSE 表示未进行回原点状态	BOOL	AS0.5	AS32.5	AS64.5	AS96.5
停止状态位, TRUE 表示停止状态, FALSE 表示非停止状态	BOOL	AS0.6	AS32.6	AS64.6	AS96.6
异常停止状态位, TRUE 表示异常停止状态, FALSE 表示非异常停止状态	BOOL	AS0.7	AS32.7	AS64.7	AS96.7
准备就绪状态位, TRUE 表示准备就绪状态, FALSE 表示非准备就绪状态	BOOL	AS0.8	AS32.8	AS64.8	AS96.8
轴处于轴组中状态位, TRUE 表示轴处于轴组中状态, FALSE 表示轴没有处于轴组中状态	BOOL	AS0.9	AS32.9	AS64.9	AS96.9
原点确定状态位, TRUE 表示原点已经确定状态, FALSE 表示原点未确定状态	BOOL	AS0.10	AS32.10	AS64.10	AS96.10
回原点中状态位, TRUE 表示回原点中状态, FALSE 表示非回原点中状态	BOOL	AS0.11	AS32.11	AS64.11	AS96.11
同步运动时轴速度达到设定的最大速度状态位, TRUE 表示达到最大速度状态, FALSE 表示未达到	BOOL	AS0.12	AS32.12	AS64.12	AS96.12
正向运动状态位, TRUE 表示正向运动状态, FALSE 表示负向运动状态	BOOL	AS0.13	AS32.13	AS64.13	AS96.13
负向运动状态位, TRUE 表示负向运动状态, FALSE 表示正向运动状态	BOOL	AS0.14	AS32.14	AS64.14	AS96.14
设定的目标位置	REAL	ASD2	ASD34	ASD66	ASD98
设定的目标速度	REAL	ASD4	ASD36	ASD68	ASD100
设定的目标加速度	REAL	ASD6	ASD38	ASD70	ASD102
设定的目标减速度	REAL	ASD8	ASD40	ASD72	ASD104
设定的目标加加速度	REAL	ASD10	ASD42	ASD74	ASD106
运动中的实际位置	REAL	ASD12	ASD44	ASD76	ASD108
运动中的实际速度	REAL	ASD14	ASD46	ASD78	ASD110
异常监视开启状态位, TRUE 表示监视开启状态, FALSE 表示监视关闭状态	BOOL	A16.0	AS48.0	AS80.0	AS112.0
异常发生时的异常编号	WORD	ASW17	ASW49	ASW81	ASW113
驱动器是否上线的状态位, TRUE 代表上线状态, FALSE 表示没有上线	BOOL	AS18.0	AS50.0	AS82.0	AS114.0
驱动器的实际脉冲个数	WORD	ASW19	ASW51	ASW83	ASW115

轴组运动的监控参数如下表。

描述	数据类型	寄存器			
		轴组 0	轴组 1	轴组 2	轴组 3

非使能状态位, TRUE 为非使能状态, FALSE 为使能状态	BOOL	AS2048.0	AS2080.0	AS2112.0	AS2144.0
停止状态位, TRUE 表示停止状态, FALSE 表示非停止状态	BOOL	AS2048.1	AS2080.1	AS2112.1	AS2144.1
目标位置运动状态, TRUE 表示向目标位置运动, FALSE 表示没有向目标位置运动	BOOL	AS2048.2	AS2080.2	AS2112.2	AS2144.2
停止运行中状态位, TRUE 表示停止运行中状态, FALSE 表示非停止运行中状态	BOOL	AS2048.3	AS2080.3	AS2112.3	AS2144.3
异常停止状态位, TRUE 表示异常停止状态, FALSE 表示非异常停止状态	BOOL	AS2048.4	AS2080.4	AS2112.4	AS2144.4
准备就绪状态位, TRUE 表示准备就绪状态, FALSE 表示非准备就绪状态	BOOL	AS2048.5	AS2080.5	AS2112.5	AS2144.5
设定的目标速度	REAL	ASD2050	ASD2082	ASD2114	ASD2146
设定的目标加速度	REAL	ASD2052	ASD2084	ASD2116	ASD2148
设定的目标减速度	REAL	ASD2054	ASD2086	ASD2118	ASD2150
异常监视开启状态位, TRUE 表示监视开启状态, FALSE 表示监视关闭状态	BOOL	AS2056.0	AS2088.0	AS2120.0	AS2152.0
异常发生时的异常编号	WORD	ASW2057	ASW2089	ASW2121	ASW2153
动力学模型选择字, 选择当前轴组使用的动力学模型	WORD	ASW2058	ASW2090	ASW2122	ASW2154
低 8 位表示轴组中 0 号位置的轴编号; 高 8 位表示轴组中 1 号位置的轴编号	WORD	ASW2059	ASW2091	ASW2123	ASW2155
低 8 位表示轴组中 2 号位置的轴编号; 高 8 位表示轴组中 3 号位置的轴编号	WORD	ASW2060	ASW2092	ASW2124	ASW2156
低 8 位表示轴组中 4 号位置的轴编号; 高 8 位表示轴组中 5 号位置的轴编号	WORD	ASW2061	ASW2093	ASW2125	ASW2157
低 8 位表示轴组中 6 号位置的轴编号; 高 8 位表示轴组中 7 号位置的轴编号	WORD	ASW2062	ASW2094	ASW2126	ASW2158

附录 G 运动控制轴参数定义

运动控制器轴参数定义如下表所示

参数 ID	说明	类型	数据范围	读写类型
1	指令位置	REAL	实数	R
2	正向位置软件限制	REAL	正实数	R/W
3	负向位置软件限制	REAL	负实数	R/W
4	正向位置软件限制使能	BOOL	0 或 1	R/W
5	负向位置软件限制使能	BOOL	0 或 1	R/W
6	保留	--	--	--
7	保留	--	--	--
8	软件设置最大速度	REAL	正实数	R/W
9	保留	--	--	--
10	实际运行速度	REAL	0 或正实数	R
11	指令速度	REAL	0 或正实数	R
12	最大加速度	REAL	0 或正实数	R/W
13	保留	--	--	--
14	最大减速度	REAL	0 或正实数	R/W
15	保留	--	--	--
16	最大加加速度	REAL	0 或正实数	R/W
17	伺服单周位移	REAL	正实数	R
18	伺服单周脉冲	REAL	正实数	R

附录 H 常见问题

问题一：为什么我电脑上的 PLC_Config 不能自动发现设备？该怎么解决？

答：

(1) 检查设备的以太网指示灯是否闪烁。如果没闪烁首先是硬件连接有问题，请检查网线是否安装正确，交换机是否正确启动。

(2) 确定用户的电脑 IP 与设备 IP 在同一个网段内。例如：设备 IP 是 192.168.1.181，那么用户电脑的 IP 也要调整为 192.168.1.* (*不能为 255)。要修改电脑 IP，用户需要进入本地连接的属性对话框，选 TCP_IP 协议进行 IP 修改。子网掩码填：255.255.255.0。

(3) 确定用户的电脑网卡选项设置的是 100M 以太网或自动适应方式。修改方式为：“我的电脑->属性->硬件->设备管理->网络适配器->属性->高级->连接速度和双工->选择自动检测”。

(4) 确定用户的电脑已经将网络防火墙关闭了。

(5) 确定用户没有关闭系统，通常 PLC_Config 能够自动启动通信连接，但是也可以手动关闭系统的通信连接。如果用户手动关闭了通信连接，则 PLC_Config 不再接收任何数据了。

问题二：从设备的通信线连接到控制器的哪里才会跟控制器进行串行通信呢？如果用触摸屏监控控制器，又该怎么连接呢？

答：

控制器一般有两个串行通信口，分别是作为主设备和从设备来用的。作为主设备时的串行口我们把它叫做主口，作为从设备时的串口我们把它叫做从口。每个口分别对应 485 通信线的 T+和 T-。如果想把从设备挂在控制器下面，只要配置为主口的 T+和 T-连接从设备 485 串行通信口的 T+和 T-即可。如果想要用触摸屏或其它串口监控设备监控控制器变量变化情况，需要把 PLC 的从口与监控设备的 485 通信口相连，连接好后就可以进行串行通信了。

问题三：为什么我已经把线连接好了，控制器还是扫描不到从设备呢？

答：

(1) 点设备管理器，察看控制器的设备参数配置，确保连从设备的串口被配置成主口，且确保另一路串口必须被配置成从口。

(2) 检查从设备的串行通信设置。是否控制器串行通信口设置了相同的串行通信格式（通信波特率、数据位的个数、停止位的个数）等。

(3) 检查从设备配置参数。是否配置了正确的从设备模块地址、扫描命令号、扫描周期数、通信协议等。

(4) 配置好后点击手动扫描，启动 PLC 扫描从设备，如果从设备全部都是 DCCE 系列产品，用户也可以通过自动发现设备来实现设备上线。

(5) 当用户用新的控制器设备替换下原有的设备后，在上位机原有工程由于会保留以前的从设备配置信息，因此需要重新下载从设备配置信息。

问题四：为什么从设备上线了，但是 PLC 不与从设备进行通信呢？

答：

PLC 没有配置从设备的 AI、AQ、DI、DQ 通信参数，或者这些配置不正确，这样虽然 PLC 能够发现从设备，也会因为没有通信控制参数而不与从设备进行通信。

问题五：为什么 PID 的参数已经配置好了，但是 PID 调节不正常呢？

答：

(1) PID 三个参数配置的比例有问题，TI 可能被配置成 0 了，这样 PID 不会进行运算

(2) 如果 PID 用作 PWM 控制, PWM_O 只能是 Q 区的变量, 如果填写了非 Q 区的变量, 则会运行不正常

(3) PID 分为两种工作模式, 即常规工作模式和自整定工作模式。当 SMW320 和 SMW32 被配置后, PID 会自动按照自整定工作方式工作, 这样在常规控制方式下可能会感觉 PID 调节不正常。请查看着两个特殊寄存器, 把它们设成 0, 然后按照常规方式进行 PID 控制。

问题六: PLC_Config 网络复杂时, 为什么有时候不能移动指令?

答:

由于功能块不能跨越连接到其他功能块, 所以当用户要将功能块移动到的位置有如上条件约束时就会造成不能移动功能块。

问题七: 定时器 T 指令和 JMP 指令配合使用时, 为何定时器不计时?

答:

定时器 T 指令在第一次运行时会载入预设值, 并初始化内部变量, 此时不进行定时处理, 只有第二次运行到此 T 指令时才开始定时, 而 T 指令到计时到达标志位也需要运行 T 指令查询处理, 因此使用 T 指令时需要每轮用户程序均运行才能达到真正的编程效果。建议用户不要将 T 指令放入 JMP 会跳转过的程序段当中, 而是放入整个用户程序每次都能执行到的位置。

问题八: 为什么从设备上线了而且控制器可以读取从设备的 DI 和 AI 输入, 但是无法写入 AQ 或 DQ 呢?

答:

- (1) 如果控制器内部有指令程序, 但是没有运行, 需要运行指令程序才能刷新从设备的 DQ 和 AQ。
- (2) PLC 没有配置从设备的 AQ、DQ 通信参数, 或者这些配置不正确。

附录 K 版本信息

版本	修改内容	修改日期
V1.1	增加运动控制指令集部分，详见第 7 章	2015.3.18
V1.0	创建文档	2013-2-10