

机器自动化控制器 原理与应用

JIQI ZIDONGHUA KONGZHIQI
YUANLI YU YINGYONG

SYSTMAC
always in control

徐世许 等编著



机器人自动化控制器 原理与应用

徐世许 等编著



机械工业出版社

本书以欧姆龙公司最先进的 NJ 系列机器人自动化控制器为背景机,系统地介绍了 NJ 的工作原理、硬件单元与系统配置、程序组织与编程技术、指令系统、自动化平台软件 Sysmac Studio 的使用,以及 NJ 的 EtherCAT 网络、运动控制功能与运动控制指令,还有 NJ 的 EtherNet/IP 网络。

本书系统性强,内容丰富而新颖,阐述清楚,理论联系实际。本书可作为大专院校自动化、电气技术、机电一体化等专业的本科和研究生教材,也可以作为工程技术人员的培训用书。

图书在版编目 (CIP) 数据

机器人自动化控制器原理与应用/徐世许等编著. —北京:
机械工业出版社, 2013. 8

ISBN 978 - 7 - 111 - 44136 - 6

I. ①机… II. ①徐… III. ①自动控制设备 IV. ①TP23

中国版本图书馆 CIP 数据核字 (2013) 第 223516 号

机械工业出版社 (北京市百万庄大街 22 号 邮政编码 100037)

策划编辑: 林春泉 责任编辑: 赵 任

封面设计: 鞠 杨 责任校对: 陈秀丽 李锦莉

责任印制: 张 楠

北京京丰印刷厂印刷

2013 年 10 月第 1 版 · 第 1 次印刷

184mm × 260mm · 27 印张 · 668 千字

0 001—4 000 册

标准书号: ISBN 978 - 7 - 111 - 44136 - 6

定价: 69.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

电话服务

网络服务

社服务中心: (010) 88361066

教材网: <http://www.cmpedu.com>

销 售 一 部: (010) 68326294

机工官网: <http://www.cmpbook.com>

销 售 二 部: (010) 88379649

机工官博: <http://weibo.com/cmp1952>

读者购书热线: (010) 88379203

封面无防伪标均为盗版

前 言

OMRON 公司是世界上生产可编程序控制器 (Programmable Logic Controller, PLC) 的著名厂家之一, 其 PLC 产品多次更新换代, 早期为 P、C200H、C1000H, 后又推出 CPM1A、CPM2A、C200H α 、CQM1、CQM1H、CV, 现在主推 CP1、CJ1/CJ2、CS1 等系列机型。OMRON 的大、中、小、微型机各具特色, 各有所长, 在中国市场上的占有率位居前列, 在国内用户中享有较高的声誉。

众所周知, PLC 以高可靠性著称, 其硬件芯片的选用相对保守。基于系统可靠性考虑, PLC 并不像个人计算机那样总是采用最先进的芯片, 如 PLC 的 CPU 与计算机的相比, 处在相当落后的地步, 很多仍停留在单片机的水平, 它的工作速度等各方面的性能偏低, 限制了 PLC 自身的能力。随着机器自动化水平的提高, 对运动控制的要求越来越高, PLC 在面对一些精密、高速、复杂的运动控制时, 往往力不从心以至于无法胜任。

针对目前 PLC 存在的技术“瓶颈”, OMRON 公司从硬件体系和软件体系进行全新的架构设计, 2011 年推出了具有划时代意义的新产品: NJ 系列的机器自动化控制器。NJ 从外观上看, 是一台小巧玲珑的 PLC, 但从内部的硬件结构上看, 其配置相当于一台个人计算机, 例如, CPU 采用时下先进的微处理器, 工作频率达到 1.66GHz, 全新的设计为 NJ 整体性能跃升打下坚实的基础。

NJ 凝聚了机器自动化所需要的各种高端功能, 它的主要特点如下: 继续保持了 PLC 高可靠性的优良传统; 指令系统与国际标准 IEC61131-3 完全接轨; 除具有 PLC 常规的功能外, 还能够实现高速、复杂、精密的运动控制; 具有丰富的通信与网络功能; 提供功能强大的自动化平台软件 Sysmac Studio。特别值得称道的是 NJ 在运动控制方面具有卓越的能力, 鉴于此, NJ 被称为机器自动化控制器。

本书编者长期跟踪 OMRON PLC 技术的发展, 2011 年下半年开始接触 NJ, 便被 NJ 强大的功能所折服, 感到很有必要把这一当今世界上先进的机器自动化控制器介绍给广大读者, 使之能迅速地应用到我国的工业自动化领域。

本书共分 8 章。首先介绍了 NJ 的工作原理、NJ 的硬件单元与系统配置。接着介绍了 NJ 的程序组织与编程技术, 以较大篇幅详尽叙述了 IEC61131-3 标准编程语言, 内容有任务、程序组织单元 (程序、功能块和功能)、数据类型、变量、编程语言 (梯形图、结构文本) 等。介绍了 NJ 的指令系统, 主要讲述了常用指令的功能和使用方法。介绍了自动化平台软件 Sysmac Studio 的使用方法, 如应用 Sysmac Studio 进行系统基本配置、EtherCAT 网络配置等, 进行控制器、运动控制、凸轮数据、任务和数据跟踪等设置; 应用 Sysmac Studio 进行编程和调试。重点介绍了 NJ 的 EtherCAT 通信、运动控制原理、运动控制指令和运动控制功能, 详细地讲述了单轴、轴组运动控制的实现方法, 如单轴速度、定位、齿轮、凸轮等控制, 轴组的直线插补、圆弧插补控制, 同时提供完整的编程实例。最后简要介绍了 NJ 的 EtherNet/IP 网络功能。

OMRON 公司的松井先生、铃木先生、金颖女士对本书的编写给予极大的支持, 提供了

大量的技术资料。崔玉兰女士、周岩先生也提供很大帮助。在此表示诚挚的谢意。

特别感谢 OMRON 公司的张杰先生、王伟先生。在编写过程中参考了张杰先生主编的 NJ 培训教材。编者多次就疑难问题与王伟先生交流，每次都得到了满意的解答。

本书由徐世许任主编，程敏、元志超、刘磊、陈大程、房福胜、徐世伟、吴光强、张德芹、邢迟、徐筱青、季文伟、刘明参加了编写工作。

由于编者水平有限，书中难免有错误和不妥之处，希望得到读者的批评指正。

编者

2013 年 6 月

目 录

前言

第 1 章 概述 1

1.1 PLC 的产生与发展 1

1.1.1 什么是 PLC 1

1.1.2 PLC 的产生与发展 1

1.1.3 PLC 的发展趋势 2

1.2 PLC 的特点与应用领域 3

1.2.1 PLC 的特点 3

1.2.2 PLC 的应用领域 4

1.3 PLC 的基本组成与各部分的作用 5

1.3.1 PLC 的基本组成 5

1.3.2 PLC 各部分的作用 5

1.4 PLC 的工作原理 10

1.4.1 PLC 的循环扫描工作过程 10

1.4.2 PLC 的 I/O 滞后现象 12

1.5 PLC 的编程语言 13

1.5.1 梯形图 14

1.5.2 语句表 15

1.5.3 功能块 15

1.5.4 结构文本 16

1.6 OMRON PLC 发展概况 17

1.7 OMRON 可编程自动化控制器 19

1.7.1 NJ 的特点 20

1.7.2 使用 NJ 的操作步骤 23

第 2 章 NJ 的硬件与配置 24

2.1 NJ 的硬件 24

2.1.1 NJ 的硬件单元 24

2.1.2 CPU/扩展机架 29

2.2 Sysmac Studio 自动化软件的安装 与使用 31

2.2.1 Sysmac Studio 的规格 31

2.2.2 Sysmac Studio 的安装与卸载 33

2.2.3 创建工程 35

2.2.4 操作视窗 40

2.2.5 命令菜单 44

2.2.6 Sysmac Studio 与控制器的 连接 47

2.3 NJ 的配置和设置 48

2.3.1 EtherCAT 的配置和设置 49

2.3.2 CPU/扩展机架的配置和 设置 51

2.3.3 I/O 映射 58

2.3.4 控制器设置 63

2.3.5 运动控制设置 64

2.3.6 Cam 数据设置 68

2.3.7 任务设置 68

2.3.8 数据跟踪设置 69

第 3 章 NJ 的编程基础 70

3.1 任务 70

3.1.1 任务类型 70

3.1.2 任务的基本操作 71

3.1.3 保证任务间变量取值的一 致性 75

3.1.4 从控制器外部访问变量与 任务执行的同步 77

3.1.5 任务设置 78

3.1.6 与任务相关的指令 81

3.1.7 与任务相关的系统定义变量 81

3.1.8 与任务相关的错误 83

3.1.9 监视任务执行状态和任务 执行时间 84

3.2 程序组织单元 85

3.2.1 程序 85

3.2.2 功能块 86

3.2.3 功能 93

3.3 变量 99

3.3.1 变量的类型 99

3.3.2 应用于 POU 的用户定义 变量的类型 100

3.3.3 变量的属性 100

3.4 数据类型 101

3.4.1 基本数据类型 101

3.4.2 导出数据类型 109

3.4.3 数组 114

3.5 编程语言	116	5.4 离线调试	210
3.5.1 梯形图	116	5.4.1 程序模拟调试	210
3.5.2 结构文本	121	5.4.2 离线调试顺序控制和运动 控制程序	213
第4章 NJ 的指令系统	133	5.4.3 NS 系列 PT 的离线模拟调试	213
4.1 概述	133	5.5 在线调试	213
4.2 梯形图指令	137	5.5.1 同步 (自动检验后上载/ 下载)	214
4.3 顺序输入指令	140	5.5.2 同步、传送和检验 EtherCAT 配置	216
4.4 顺序输出指令	142	5.5.3 同步、传送和 CPU/扩展机架 组件比较	216
4.5 顺序控制指令	145	5.5.4 测试轴运行 (MC 试 运行)	216
4.6 定时器指令	150	5.5.5 复位控制器	216
4.7 计数器指令	154	5.6 故障分析	216
4.8 比较指令	159	第6章 NJ 的 EtherCAT 网络	218
4.9 运算指令	163	6.1 概述	218
4.10 BCD 转换指令	167	6.1.1 EtherCAT 特点	218
4.11 数据类型转换指令	169	6.1.2 NJ 的 EtherCAT 网络配置	219
4.12 位串处理指令	171	6.1.3 内置 EtherCAT 端口的规范	221
4.13 数据传送指令	174	6.1.4 EtherCAT 主站、从站的 通信端口	222
4.14 移位指令	178	6.1.5 EtherCAT 网络拓扑结构	223
第5章 Sysmac Studio 在编程与 调试中的作用	182	6.1.6 EtherCAT 网络的连接	225
5.1 编程与调试步骤	182	6.2 EtherCAT 通信机制	226
5.2 编程操作	183	6.2.1 EtherCAT 通信类型和设置	226
5.2.1 添加梯形图或 ST	183	6.2.2 EtherCAT 通信的实现	227
5.2.2 梯形图编程	184	6.2.3 EtherCAT 通信的状态转换	229
5.2.3 ST 编程	193	6.3 EtherCAT 配置与设置	230
5.2.4 查找和替换	196	6.3.1 EtherCAT 网络操作步骤	230
5.2.5 程序检查	196	6.3.2 创建 EtherCAT 网络配置	232
5.2.6 编译和重编译	197	6.3.3 设置 EtherCAT 从站变量 和轴	234
5.3 调试操作	198	6.3.4 EtherCAT 主站和从站参 数设置	239
5.3.1 模拟器与控制器	198	6.3.5 EtherCAT 网络配置比较 和合并	243
5.3.2 监视	199	6.3.6 从 Sysmac Studio 下载网 络配置信息	249
5.3.3 改变当前值、设置/重置、 强制刷新	199	6.3.7 完成 EtherCAT 配置和设置 后确认通信正常	250
5.3.4 交叉索引窗口	201	6.4 PDO 通信和 SDO 通信	250
5.3.5 在线编辑	202		
5.3.6 改变操作模式	203		
5.3.7 监视控制器状态	203		
5.3.8 任务执行状态监视	204		
5.3.9 任务执行时间监视	204		
5.3.10 轴状态监视 (MC 监 视表)	206		
5.3.11 数据跟踪	206		

6.4.1	PDO 通信	250	7.5.8	相对定位	301
6.4.2	SDO 通信	254	7.6	运动控制编程基础	301
6.4.3	EtherCAT 通信使用的指令	257	7.6.1	运动控制指令	302
6.5	EtherCAT 故障诊断	257	7.6.2	状态转换	302
6.5.1	查找错误的方法	257	7.6.3	运动控制指令的执行时序	304
6.5.2	与 EtherCAT 主站功能模块 相关的错误	258	7.6.4	运动控制指令重新执行时 序图	307
6.5.3	复位错误	261	7.6.5	运动控制指令多次执行时 序图	307
6.5.4	诊断和统计信息	261	7.6.6	运动控制的系统定义变量	308
6.6	EtherCAT 其他功能	262	7.6.7	凸轮表和凸轮数据变量	309
6.6.1	监视包	262	7.7	单轴运动控制指令	311
6.6.2	从站的使能与禁用	264	7.7.1	概述	311
6.6.3	从站的断开连接与重新 连接	265	7.7.2	伺服上电指令 MC_Power	314
第 7 章	NJ 的运动控制	267	7.7.3	点动指令 MC_MoveJog	315
7.1	概述	267	7.7.4	原点返回指令 MC_Home	316
7.2	运动控制配置与原理	269	7.7.5	运动指令 MC_Move	318
7.2.1	CPU 单元的内部配置	269	7.7.6	伺服停止指令 MC_Stop	318
7.2.2	运动控制配置	270	7.7.7	速度控制指令 MC_MoveVelocity	319
7.2.3	运动控制原理	270	7.7.8	启动凸轮操作指令 MC_CamIn	324
7.2.4	EtherCAT 通信和运动控制	273	7.7.9	结束凸轮操作指令 MC_CamOut	341
7.3	轴	275	7.7.10	启动齿轮操作指令 MC_GearIn	342
7.3.1	轴简介	275	7.7.11	结束齿轮操作指令 MC_GearOut	346
7.3.2	轴参数	276	7.8	轴组运动控制指令	347
7.3.3	轴参数设置举例	284	7.8.1	概述	347
7.3.4	轴变量	285	7.8.2	轴组使能指令 MC_GroupEnable	348
7.3.5	从 Sysmac Studio 进行轴参数 设置	287	7.8.3	轴组禁用指令 MC_GroupDisable	348
7.4	轴组	287	7.8.4	轴组停止指令 MC_GroupStop	349
7.4.1	轴组简介	287	7.8.5	轴组直线插补指令 MC_MoveLinear	351
7.4.2	轴组参数	288	7.8.6	轴组圆弧插补指令 MC_MoveCircular2D	360
7.4.3	轴组变量	290	7.9	运动控制功能	369
7.4.4	从 Sysmac Studio 进行轴组 参数设置	292	7.9.1	单轴定位控制	369
7.5	MC 试运行	296	7.9.2	单轴同步控制	372
7.5.1	MC 试运行功能	296			
7.5.2	启动 MC 试运行	297			
7.5.3	监视传感器信号	299			
7.5.4	检查伺服电动机 ON/OFF 操作	300			
7.5.5	点动	300			
7.5.6	原点返回	300			
7.5.7	绝对定位	301			

7.9.3 单轴速度控制	373	8.1.6 NJ 系列 CPU 单元内置 EIP 端口的特性	399
7.9.4 单轴扭矩控制	373	8.2 标签数据链接	400
7.9.5 单轴控制的通用功能	374	8.2.1 标签数据链接规格	400
7.9.6 单轴运动控制指令的重新 执行	381	8.2.2 标签数据链接的设置	402
7.9.7 单轴运动控制指令的多次 执行	383	8.2.3 启动和停止数据链接	404
7.9.8 多轴协调控制	388	8.2.4 控制器状态	404
7.9.9 多轴协调控制的通用功能	389	8.2.5 标签数据链接数据的并发	404
7.9.10 多轴协调控制运动控制 指令的重新执行	390	8.3 CIP 信息通信	406
7.9.11 多轴协调控制运动控制 指令的多次执行	390	8.3.1 概述	406
7.9.12 多轴协调控制运动控制 指令的过渡模式	391	8.3.2 CIP 通信指令	407
第 8 章 NJ 的 EtherNet/IP 网络	394	8.3.3 CIP 通信指令的使用	408
8.1 概述	394	8.3.4 路由路径	409
8.1.1 内置 EtherNet/IP 端口的 地址	396	8.4 其他通信服务	412
8.1.2 内置 EtherNet/IP 端口的 指示灯	396	8.4.1 Socket 服务	412
8.1.3 EtherNet/IP 网络的连接	397	8.4.2 FTP 服务	413
8.1.4 与内置 EIP 端口相关的 系统定义变量	398	8.4.3 自动时钟调整 (NTP 服务)	415
8.1.5 内置 EtherNet/IP 端口的 设置	399	8.4.4 SNMP 服务	416
		8.4.5 BOOTP 客户端	418
		8.4.6 IP 路由表	418
		8.4.7 HOST 名指定	419
		8.4.8 FINS 通信	419
		8.4.9 NJ 控制器与 NS 触摸屏通信	420
		8.4.10 NJ 控制器与 CJ PLC 的 EtherNet/IP 通信	421
		参考文献	422

第 1 章 概 述

1.1 PLC 的产生与发展

1.1.1 什么是 PLC

PLC 是一种以计算机 (微处理器) 为核心的通用工业控制装置, 目前已被广泛地应用于工业生产的各个领域。早期的 PLC 只能进行开关量的逻辑控制, 被称为可编程序逻辑控制器。现代 PLC 采用微处理器 (Microprocessor) 作为中央处理单元, 其功能大大增强, 它不仅具有逻辑控制功能, 还具有过程控制、运动控制和通信联网等功能, PLC 这一名称已不能准确地反映它的特性, 于是, 人们将其称为可编程序控制器 Programmable Controller, 简称 PC。但近年来个人计算机 (Personal Computer) 也简称 PC, 为了避免混淆, 可编程序控制器常被称为 PLC。

1987 年, 国际电工委员会 (IEC) 在其颁布的 PLC 标准草案第三稿中, 对 PLC 定义如下: PLC 是一种数字运算操作的电子系统, 专为工业环境下应用而设计。它采用可编程序的存储器, 用来在其内部存储执行逻辑运算、顺序控制、定时、计数和算术运算等操作的指令, 并通过数字式、模拟式的输入和输出, 控制各种机械或生产过程。PLC 及其有关设备, 都应按易于使工业控制系统形成一个整体, 易于扩充其功能的原则设计。

现代 PLC 的功能已远远超出上述定义的范围。近年来, PLC 的发展异常迅猛, 产品更新换代的速度明显加快, 功能日益增强, 应用领域愈加广泛。PLC 已成为实现工业自动化的一种强有力工具。

1.1.2 PLC 的产生与发展

在 PLC 出现之前, 机械控制及工业生产控制是用工业继电器实现的。在一个复杂的控制系统中, 可能要使用成百上千个各式各样的继电器, 接线、安装的工作量很大。如果控制工艺及要求发生变化, 控制柜内的元件和接线也需要做相应的改动, 但是这种改动往往费用高、工期长, 以至于有的用户宁愿扔掉旧的控制柜, 去制作一台新的控制柜。在一个复杂的继电器控制系统中, 如果有一个继电器损坏, 甚至某一个继电器的某一对触点接触不良, 都会导致整个系统工作不正常, 由于元件多、线路复杂, 查找和排除故障往往很困难。继电器控制的这些固有缺点, 给日新月异的工业生产带来了不可逾越的障碍。由此, 人们产生了一种寻求新型控制装置的想法。

1968 年, 美国最大的汽车制造厂家通用汽车公司 (GM 公司) 为了适应汽车型号不断翻新的要求, 提出如下设想: 能否把计算机功能完备、灵活、通用等优点和继电器控制系统的简单易懂、操作方便、价格便宜等优点结合起来, 做成一种通用控制装置, 并把计算机的编程方法和程序输入方式加以简化, 用面向过程、面向问题的“自然语言”编程, 使不熟悉

计算机的人也能方便地使用。这样,使用人员不必在编程上花费大量的精力,而是集中力量去考虑如何发挥该装置的功能和作用。这一设想提出后,美国数字设备公司(DEC公司)首先响应,于1969年研制出了世界上第一台PLC,型号为PDP-14。用它代替传统的继电器控制系统,在美国GM公司的汽车自动装配线上试用获得了成功。

此后,这项新技术就迅速发展起来。1971年日本从美国引进了这项新技术,很快就研制出了日本第一台PLC(DSC-8)。1973~1974年,原西德和法国也研制出自己的PLC。我国从1974年开始研制,1977年研制成功了以一位微处理器MC14500为核心的PLC,并开始工业应用。

PLC自产生以来,随着大规模集成电路和微处理器技术的发展,一直在不断地更新换代。现代PLC全面使用16位、32位的微处理器芯片,位片式微处理器、精简指令系统微处理器(RISC)等高性能、高速度的CPU,极大地提高了PLC的工作性能、速度和可靠性;同时由于大量含有微处理器的智能模块的出现,使现代PLC不仅具有逻辑控制,还同时具有过程控制、运动控制、数据处理、通信联网等诸多功能,真正成为名副其实的多功能控制器。

1.1.3 PLC的发展趋势

随着微处理技术的发展,PLC也得到了迅速发展,其技术和产品日趋完善。它不仅以其良好的性能满足了工业生产的广泛需要,而且将通信技术和信息处理技术融为一体,使其功能更加完备。目前,为了适应大中小型企业不同需要,进一步扩大PLC在工业自动化领域的应用范围,PLC正朝着以下3个方向发展:其一是小型PLC向体积缩小、功能增强、速度加快、价格低廉的方向发展,使之能更加广泛地取代继电器控制;其二是大中型PLC向大容量、高可靠性、高速度、多功能、网络化的方向发展,使之能对大规模、复杂系统进行综合性的自动控制;其三是各种专业化的PLC不断涌现,如安全型PLC、运动控制PLC、过程控制PLC、软PLC等。总的趋势是:

1. CPU处理速度进一步加快

PLC的CPU使用64bit RISC芯片,多CPU并行处理或分时处理或分任务处理,各种模块智能化,部分系统程序用门阵列电路固化,以加快处理速度。近年来出现采用计算机先进微处理器的新型控制器,其性能和功能较之传统的PLC全面跃升,这种新型控制器称为可编程自动化控制器(Programmable Automation Controller, PAC)。

2. 控制系统将分散化

根据分散控制、集中管理的原则,PLC控制系统的I/O模块将直接安装在控制现场,通过通信电缆或光缆与主CPU进行数据通信。这样使控制更有效,系统更可靠。

3. 可靠性进一步提高

随着PLC进入过程控制领域,对可靠性的要求进一步提高。硬件冗余的容错技术将进一步应用。不仅会有CPU单元冗余、通信单元冗余、电源单元冗余、I/O单元冗余,甚至会有整个系统冗余。

4. 控制与管理功能一体化

为了满足现代化大生产的控制与管理的需要。PLC将广泛采用计算机信息处理技术、网络通信技术和图形显示技术,使PLC系统的生产控制功能和信息管理功能融为一体。

1.2 PLC 的特点与应用领域

1.2.1 PLC 的特点

PLC 的诞生给工业控制带来了一次革命性的飞跃,与继电器、微机控制相比,PLC 有它独特之处,下面来看一下 PLC 的特点。

1. 可靠性高、抗干扰能力强

对工业控制器件来讲,可靠性是一个非常重要的指标,如何能在各种恶劣的工业环境和条件(如电磁干扰、低温潮湿、灰尘、超高温等)下,平稳、可靠地工作,将故障率降至最低,是研制每一种控制器件必须考虑的问题。

PLC 采用的是微电子技术,大量的开关动作是由无触点的半导体电路来完成的,因此不会出现继电器控制系统中的接线老化、脱焊、触点电弧等现象,提高了可靠性。另外,PLC 还在硬件和软件两方面采取了许多有力措施来提高其可靠性。

PLC 具有很高的可靠性和抗干扰能力,因此被称为“专为适应恶劣工业环境而设计的计算机”。

2. 灵活、通用

在继电器控制系统中,使用的控制器件是大量的继电器,整个系统是根据设计好的电气控制图,由人工通过布线、焊接、固定等手段组装完成的,其过程费时、费力。如果因为工艺上的稍许变化,需要改变电气控制系统的话,那么原先的整个电气控制系统将被全部拆除,而重新进行布线、焊接、固定等工作,耗费了大量的人力、物力和时间。而 PLC 是通过存储在存储器中的程序实现控制功能的,如果控制功能需要改变的话,只需要修改程序以及改动极少量的接线即可。而且,同一台 PLC 还可以用于不同的控制对象,改变软件就可以实现不同的控制要求,因此具有很大的灵活性、通用性。PLC 有多种功能模块,可以根据需要灵活组合成各种不同功能的控制装置,实现各种特殊的控制要求。

3. 编程简单、使用方便

用微机实现控制,使用的是汇编语言,难于掌握,要求使用者具有一定水平的计算机硬件和软件知识。而 PLC 采用面向控制过程、面向问题的“自然语言”编程,容易掌握。例如目前大多数 PLC 采用的梯形图语言编程方式,既继承了继电器控制线路的清晰直观感,又考虑到大多数电气技术人员的读图习惯及应用微机的水平,很容易被电气技术人员所接受。PLC 易于编程,程序改变时也容易修改,灵活方便。这种面向控制过程、面向问题的编程方式,与目前微机控制常用的汇编语言相比,虽然在 PLC 内部增加了解释程序,增加了程序执行时间,但对大多数的机电控制设备来说,PLC 的控制速度是足够快的。

用微机控制,还要在输入、输出接口上做大量工作,才能与控制现场连接起来,调试也比较繁琐。而 PLC 的输入、输出接口已经做好,可直接与控制现场的用户设备连接。输入接口可以与各种开关和传感器连接,输出接口具有较强的驱动能力,可以直接与继电器、接触器、电磁阀等连接,使用很方便。

4. 接线简单

PLC 的接线只需将输入设备(如按钮、开关等)与 PLC 输入端子连接,将输出设备(如

接触器、电磁阀等) 与 PLC 输出端子连接。接线工具仅为螺钉旋具, 接线工作极其简单、工作量极少。

5. 功能强

现代 PLC 不仅具有条件控制、计时、计数、步进等控制功能, 而且还能完成 A-D、D-A 转换、数字运算和数据处理以及通信联网、生产过程监控等。因此, 它既可对开关量进行控制, 又可对模拟量进行控制; 既可控制一台单机、一条生产线, 又可控制一个机群, 多条生产线; 既可现场控制, 又可远距离控制; 既可控制简单系统, 又可控制复杂系统。

6. 体积小、重量轻、易于实现机电一体化

由于 PLC 采用半导体集成电路, 因此具有体积小、重量轻、功耗低的特点。且由于 PLC 是专为工业控制而设计的专用计算机, 其结构紧凑、坚固耐用、体积小巧, 并由于具备很强的可靠性和抗干扰能力, 使之易于装入机械设备内部, 因而成为实现机电一体化十分理想的控制设备。

1.2.2 PLC 的应用领域

1. 开关量的逻辑控制

开关量的逻辑控制是 PLC 最基本的控制功能。所控制的逻辑可以是各种各样的: 时序的、组合的、延时的、计数的、不计数的等。控制的输入、输出点数可以不受限制。少则十点、几十点, 多则成千上万点, 并可通过联网来实现控制。

用 PLC 进行开关量控制的实例很多, 冶金、机械、纺织、轻工、化工等, 几乎所有工业行业都需用到它。目前, PLC 首用的目标, 就是用于开关量的控制。

2. 模拟量的闭环控制

PLC 具有 A-D、D-A 转换及算术运算等功能, 因此可以实现模拟量控制。有的 PLC 还具有 PID 控制或模糊控制的功能, 可用于闭环的位置控制、速度控制和过程控制。目前, 除大型机、中型机具有此功能外, 小型机也普遍具有这种功能。

3. 数字量的智能控制

利用 PLC 能接收和输出高速脉冲的功能, 再配备相应的传感器 (如旋转编码器) 或脉冲伺服装置 (如环型分配器、功放、步进电动机) 就能实现数字量的智能控制。较高级的还专门开发了位控单元模块、运动单元模块等, 实现曲线插补。新开发的运动单元, 还认识数控技术的编程语言, 为 PLC 进行数字量控制提供了方便。

4. 数据采集与监控

PLC 在控制现场实行控制, 把现场的数据实时显示出来或采集保存下来, 供进一步分析研究是很重要的。较普遍使用的是 PLC 加上触摸屏, 可随时观察采集下来的数据及统计分析结果。有的 PLC 本身就具有数据记录单元, 可使用一般的便携式计算机的存储卡, 插入该单元中即可保存采集到的数据。

利用 PLC 自检信号多的特点实现自诊断式的监控, 减少系统的故障, 提高累计平均无故障运行时间, 同时可减少故障修复时间, 提高系统的可靠性。

5. 通信、联网及集散控制

PLC 的通信联网能力很强。除了 PLC 与 PLC 之间通信和联网以外, PLC 还可以与计算机进行通信和联网, 由计算机来实现对其编程和管理。PLC 也能与智能仪表、智能执行装置

(如变频器) 进行通信和联网, 互相交换数据并实施对其的控制。

利用 PLC 强大的通信联网功能, 把 PLC 分布到控制现场, 并实现各站间的通信, 上、下层间的通信, 达到分散控制、集中管理, 即构成了现在的 PCS 系统。

1.3 PLC 的基本组成与各部分的作用

1.3.1 PLC 的基本组成

PLC 是一种通用的工业控制装置, 其组成与一般的微机系统基本相同。按结构形式的不同, PLC 可分为整体式和组合式两类。

整体式 PLC 是将中央处理单元 (CPU)、存储器、输入单元、输出单元、电源、通信接口等组装成一体, 构成主机。另外, 还有独立的 I/O 扩展单元与主机配合使用。在主机中, CPU 是 PLC 的核心, I/O 单元是连接 CPU 与现场设备之间的接口电路, 通信接口用于 PLC 与编程器和上位机等外部设备的连接。整体式 PLC 的特点是结构紧凑、体积小、重量轻、价格低、输入/输出点数固定、实现的功能和控制规模固定, 灵活性较低。小型 PLC 常采用这种结构, 适用于工业生产中的单机控制。

组合式 (模块式) PLC 为总线结构, 其总线做成总线板, 上面有若干个总线槽, 每个总线槽上可安装一个 PLC 模块, 不同的模块实现不同的功能。PLC 的 CPU、存储器做成一个模块 (有的把电源也做在上面), 该模块在总线上的安装位置一般来说是固定的。其它的模块可根据 PLC 的控制规模、实现的功能选用, 安装在总线板的其它任一总线槽上。组合式 PLC 安装完成后, 需进行登记, 使 PLC 对安装在总线上的各模块进行地址确认。组合式 PLC 的总线板又称底板。装有 CPU 单元的底板称为 CPU 底板, 其它称为扩展底板。CPU 底板与扩展底板之间通过电缆连接, 距离一般不超过 10m。组合式 PLC 的特点是: 系统配置灵活, 可构成具有不同控制规模 and 功能的 PLC, 但它的价格较高。一般大、中型 PLC 采用这种结构。

无论哪种结构类型的 PLC, 都可根据需要进行配置与组合。例如 OMRON CPM1A 型 PLC 为整体式结构, 通过主机连接 I/O 扩展单元, I/O 点数可在 10 ~ 160 点的范围内进行配置。组合式 PLC 则在 I/O 配置上更方便、更灵活。

图 1-1 所示为整体式 PLC 的组成示意图, 图 1-2 所示为组合式 PLC 的组成示意图。

1.3.2 PLC 各部分的作用

1. 中央处理单元 (CPU)

CPU 在 PLC 中的作用类似于人体的神经中枢, 它是 PLC 的运算、控制中心。它按照系统程序所赋予的功能, 完成以下任务:

- 1) 接收并存储从编程器输入的用户程序和数据。
- 2) 诊断电源、PLC 内部电路的工作状态和编程的语法错误。
- 3) 用扫描的方式接收输入信号, 送入 PLC 的数据寄存器保存起来。

4) PLC 进入运行状态后, 根据存放的先后顺序逐条读取用户程序, 进行解释和执行, 完成用户程序中规定的各种操作。

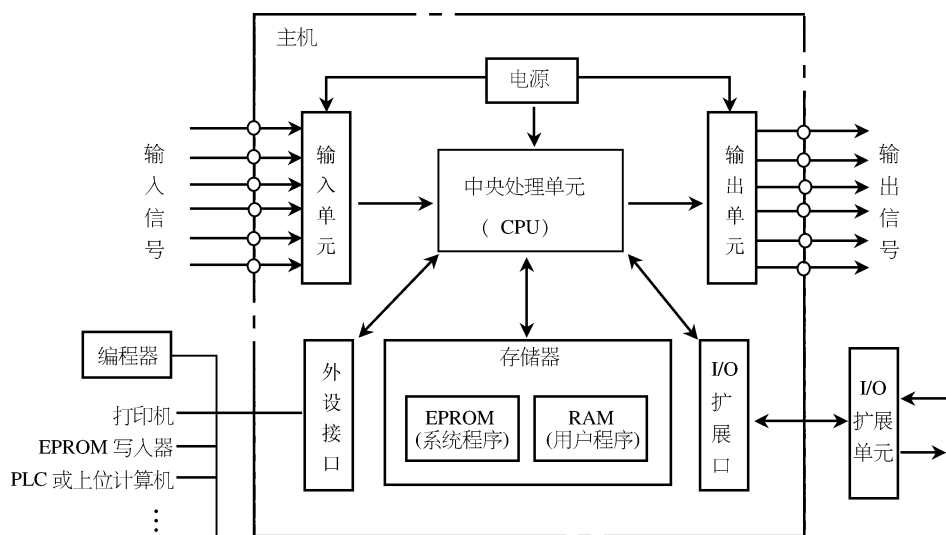


图 1-1 整体式 PLC 的组成示意图

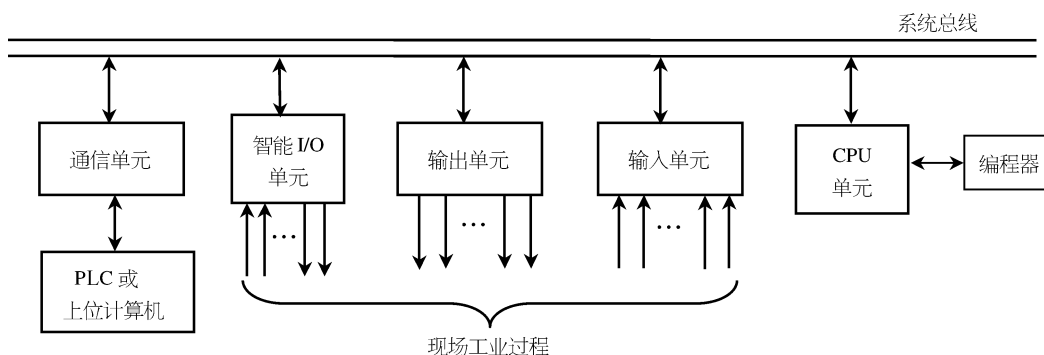


图 1-2 组合式 PLC 的组成示意图

5) 将用户程序的执行结果送至输出端。

现代 PLC 使用的 CPU 主要有以下几种：

1) 通用微处理器，如 8080、6800、Z80A、8086 等。通用微处理器的价格便宜，通用性强，还可以借用微机成熟的实时操作系统、丰富的软硬件资源。

2) 单片机，如 8051 等。由于单片机集成度高、体积小、价格低和可扩充性好，很适合在小型 PLC 上使用，也广泛用于 PLC 的智能 I/O 模块。

3) 位片式微处理器，如 AMD2900 系列等。位片式微处理器是独立于微型机的另一分支。它主要追求运算速度快，它以 4 位为一片，用几个位片级联，可以组成任意字长的微处理器。改变微程序存储器的内容，可以改变计算机的指令系统。位片式结构可以使用多个微处理器，将控制任务划分为若干个可以并行处理的部分，几个微处理器同时进行处理。这种高运算速度与可以适应用户需要的指令系统相结合，很适用于以顺序扫描方式工作的 PLC 使用。

2. 存储器

根据存储器在系统中的作用，可以把它分为以下三种：

1) 系统程序存储器: 和各种计算机一样, PLC 也有其固定的监控程序、解释程序, 它们决定了 PLC 的功能, 称为系统程序, 系统程序存储器就是用来存放这部分程序的。系统程序是不能由用户更改的, 故所使用的存储器为只读存储器 ROM 或 EPROM。

2) 用户程序存储器: 用户根据控制功能要求而编制的应用程序称为用户程序, 用户程序存放在用户程序存储器中。由于用户程序需要经常改动、调试, 故用户程序存储器多为可随时读写的 RAM。由于 RAM 掉电会丢失数据, 因此使用 RAM 作用户程序存储器的 PLC, 都有后备电池 (锂电池) 保护 RAM, 以免电源掉电时, 丢失用户程序。当用户程序调试修改完毕, 不希望被随意改动时, 可将用户程序写入 EPROM。目前较先进的 PLC (如欧姆龙公司的 CPM1A 型 PLC) 采用快闪存储器作用户程序存储器, 快闪存储器可随时读写, 掉电时数据不会丢失, 不需后备电池保护。

3) 工作数据存储器: 工作数据是经常变化、经常存取的一些数据。这部分数据存储在 RAM 中, 以适应随机存取的要求。在 PLC 的工作数据存储器区, 开辟有元件映像寄存器和数据表。

元件映像寄存器用来存储 PLC 的开关量输入/输出和定时器、计数器、辅助继电器等内部继电器的 ON/OFF 状态。

数据表用来存放各种数据, 它的标准格式是每一个数据占一个字。它存储用户程序执行时的某些可变参数值, 如定时器和计数器的当前值和设定值。它还用来存放 A-D 转换得到的数字和数学运算的结果等。

根据需要, 部分数据在停电时用后备电池维持其当前值, 在停电时可以保持数据的存储器区域称为数据保持区。

3. I/O 单元

I/O 单元也称为 I/O 模块。PLC 通过 I/O 单元与工业生产过程现场相联系。输入单元接收操作指令和现场的状态信息, 如控制按钮、操作开关和限位开关、光电管、继电器触点、行程开关、接近开关等信号, 并通过输入电路的滤波、光电隔离和电平转换等将这些信号转换成 CPU 能够接收和处理的信号。输出单元将 CPU 送出的弱电控制信号通过输出电路的光电隔离和功率放大等转换成现场需要的强电信号输出, 以驱动接触器、电磁阀、电磁铁等执行元件。

I/O 单元有多种类型, 但各种 I/O 单元的原理基本相同。下面介绍几种常用的 I/O 单元, 并说明其工作原理。

(1) 开关量输入单元

PLC 的开关量输入单元按照输入端电源类型的不同, 分为直流输入单元和交流输入单元。

图 1-3 所示为直流输入单元。直流输入单元外接直流电源, 电路如图 1-3 所示, 虚线框内为 PLC 内部输入电路, 框外左侧为外部用户接线, 图中只画出对应于一个输入点的输入电路, 各个输入点所对应的输入电路均相同。图中, T 为光耦合器, 它是将发光二极管与光电晶体管封装在一个管壳中, 当二极管中有电流流过时, 二极管发光, 使光电晶体管导通; 当二极管中无电流流过时, 晶体管不导通。发光二极管 LED 指示该点输入状态。R₁ 为限流电阻, R₂ 和 C 构成滤波电路, 可滤除输入信号中的高频抖动。输入单元的外接直流电源的极性任意。当 S 闭合时, 光耦合器导通, A 点为高电平, 经滤波器、输入选择器后, 送到 I/O 总线, CPU 访问该路信号时, 将该点对应的输入映像寄存器状态置 1; 发光二极管 LED 点

亮, 指示输入开关处于接通状态。当 S 断开时, 光耦合器不导通, A 点为低电平, 该电平经滤波电路、输入选择器后, 送到 I/O 总线, CPU 访问该路信号时, 将该点对应的输入映像寄存器状态置 0; 发光二极管 LED 不亮, 指示输入开关处于断开状态。

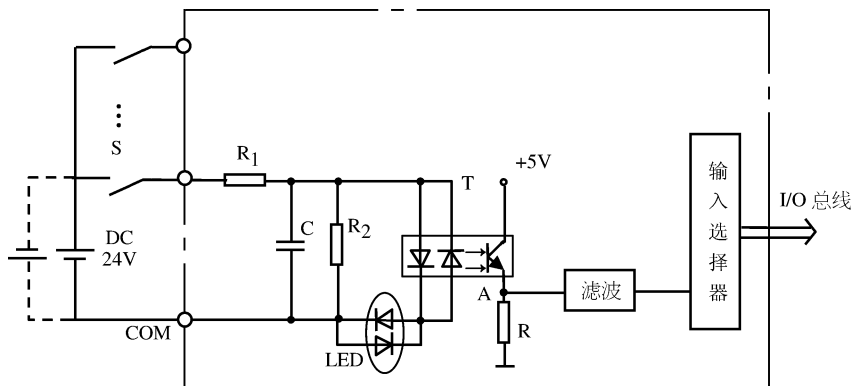


图 1-3 直流输入单元

有的直流输入单元不需要外接电源, 称为无源式输入单元。无源式输入单元的内部提供 24V 直流电源, 用户只需将开关接在输入端子和公共端子之间即可, 不用再外接电源, 简化了输入端的接线。

在图 1-3 所示的输入电路中, 输入端子有一个公共端子 COM, 即有一个公共汇集点, 因此称为汇点式输入方式。除此之外, 输入单元还有分组式和分隔式。分组式输入单元的输入端子分为若干组, 每组共用一个公共端子和一个电源。分隔式输入单元的输入端子互相隔离, 互不影响, 各自使用独立的电源。

(2) 开关量输出单元

PLC 的开关量输出单元按输出电路所用开关器件的不同可分为晶体管输出单元、晶闸管输出单元、继电器输出单元。

图 1-4 所示为晶体管输出单元。在晶体管输出单元中, 输出电路采用晶体管作为开关器件, 虚线框内为 PLC 内部输出电路, 框外右侧为外部用户接线。图中只画出对应于一个输出点的输出电路, 各个输出点所对应的输出电路均相同。图中, LED 为输出指示灯, VT 为输出晶体管, VD_2 为负载续流二极管, VD_1 为保护二极管, FU 为熔断器。工作原理如下: 当对应于 VT 的内部继电器的状态为 1 时, CPU 通过 I/O 总线使输出锁存器对应的位为高电平, LED 点亮, 指示该输出点开关量为 1 状态; 光耦合器导通, 晶体管 VT 饱和导通, 负载得电。反之, 当对应于 VT 的内部继电器的状态为 0 时, CPU 通过 I/O 总线使输出锁存器对应的位为低电平, LED 不亮, 指示该输出点开关量为 0 状态; 光耦合器截止, 晶体管 VT 截止, 负载失电, 并通过续流二极管 VD_2 释放能量。

晶体管开关量输出单元为无触点输出, 使用寿命长, 响应速度快。

4. 智能单元

PLC 除开关量 I/O 基本单元外, 还有多种智能单元。智能单元本身是一个独立的计算机系统, 它有自己的 CPU、系统程序、存储器, 以及与外界过程相连的接口。另外, 根据单元的功能还有一些特殊组成部分。智能单元是 PLC 系统的一个模块, 和 CPU 单元通过系统

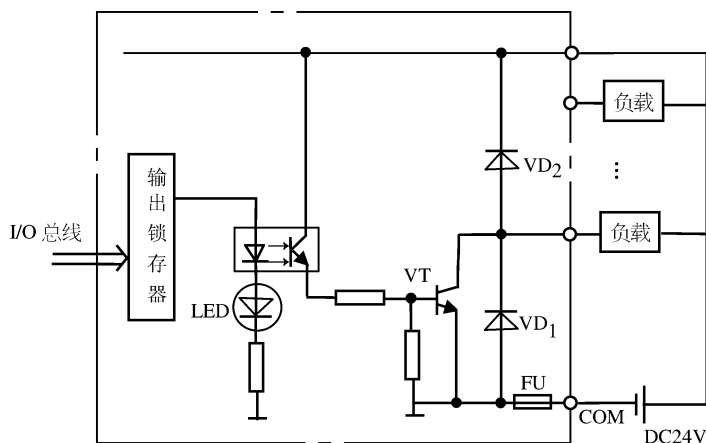


图 1-4 晶体管输出单元

总线相连接，进行数据交换，并在 CPU 单元的协调管理下独立地进行工作。这里所说的独立是指智能单元的工作不参加循环扫描过程，而是按照它自己的规律参与系统工作，即多数情况下的运算功能都由其本身的 CPU 完成。

目前，已开发的常用的智能单元有：A-D 单元、D-A 单元、高速计数单元、位置控制单元、PID 控制单元、温度控制单元和各种通信单元等。

通信单元用来完成 PLC 与其它 PLC、上位计算机或智能控制设备之间的通信，构成各种网络。例如，利用 OMRON 的 DeviceNet 单元可以搭建 DeviceNet 现场总线，利用 Ethernet 单元搭建 Ethernet 网络。

5. 电源

现代 PLC 一般配有开关式稳压电源，供内部电路使用。与普通电源相比，开关电源的输入电压范围宽，体积小，重量轻，效率高，抗干扰性能好。有的 PLC 还向外提供 DC 24V 的直流电源，给开关量输入单元连接的现场无源开关使用，或给外部传感器供电。

6. 扩展口

扩展口是 PLC 的总线接口。当用户所需的 I/O 点数或类型超出主机上的点数或类型时，可以通过加接 I/O 扩展单元来解决。主机与 I/O 扩展单元通过扩展口连接。

7. 编程工具

编程工具的主要作用是用来编辑程序、调试程序和监控程序的执行，还可以在线测试 PLC 的内部状态和参数，与 PLC 进行人机对话。因此，编程工具是开发、应用、监控运行和检查维护 PLC 不可缺少的设备。编程工具可以是简易编程器，也可以是配有专用编程软件的通用计算机。目前，简易编程器用的很少，主要为计算机辅助编程。

世界上各 PLC 生产厂家都提供在个人计算机上运行的辅助编程软件，如欧姆龙公司的 CX-P 编程软件。应用不同型号和厂家的 PLC，只需要更换相应的软件就可以了。编程软件的功能很强，它可以编辑、修改用户的程序，监控系统运行，打印文件，采集和分析数据，在屏幕上显示系统运行状况，对工业现场和系统仿真，将程序备份保存，实现计算机和 PLC 之间的程序传送等。

1.4 PLC 的工作原理

1.4.1 PLC 的循环扫描工作过程

各种 PLC 都采用扫描工作方式，具体工作过程大同小异。下面以欧姆龙公司的小型机 CP1 为例，介绍 PLC 的工作过程。如图 1-5a 所示，PLC 上电后，首先进行初始化，然后进入循环工作过程。一次循环过程可归纳为 5 个工作阶段，如图 1-5b 所示。一次循环所用的时间称为一个工作周期 (或扫描周期)，其长短与用户程序的长短以及 PLC 本身的性能有关，其数量级为 ms 级，典型值为几十 ms。

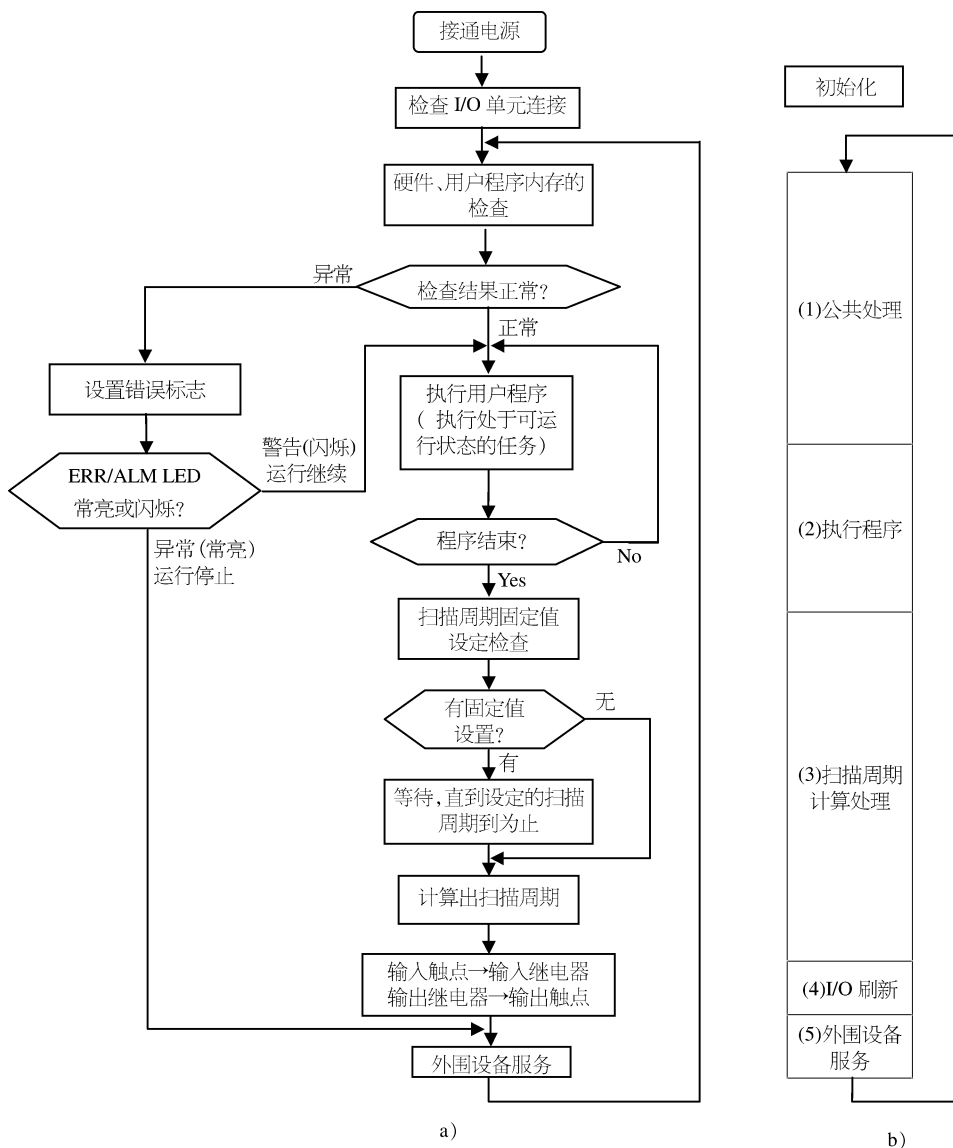


图 1-5 PLC 工作流程图

各阶段完成的任务如下：

1) 公共处理：进行硬件检查、用户程序内存检查和电池异常检查等。检查正常后，方可进行下面的操作。如果有异常情况，则根据错误的严重程度发出报警或停止 PLC 运行。

2) 程序执行：在程序执行阶段，CPU 按先左后右，先上后下的顺序对每条指令进行解释、执行，CPU 从输入映像寄存器（每个输入继电器对应一个输入映像寄存器，其通/断状态对应 1/0）和元件映像寄存器（即与各种内部继电器、输出继电器对应的寄存器）中读出各继电器的状态，根据用户程序给出的逻辑关系进行逻辑运算，运算结果再写入元件映像寄存器中。

3) 扫描周期计算处理：若设定扫描周期为固定值（可由用户在 PLC 设置中设定），则进入等待循环，直到该固定值到，再往下进行。若设定扫描周期为不定的（即决定于用户程序的长短等，为不定值），不等待，接着进行扫描周期的计算。

4) I/O 刷新：在此阶段，进行 I/O 刷新。输入刷新时，CPU 从输入电路中读出各输入点状态，并将此状态写入输入映像寄存器中；输出刷新时，将输出继电器的元件映像寄存器的状态（1/0）传送到输出锁存电路，再经输出电路隔离和功率放大，驱动外部负载。

5) 外围设备服务：包括 USB 端口通信、串行端口通信、存储盒访问，以及通信单元的链接服务等。

根据上述 PLC 的工作过程，可以得出从输入端子到输出端子的信号传递过程如图 1-6 所示。若输入端子外接开关状态有变化，则此变化反映到输入电路的输出端；在 I/O 刷新阶段，CPU 从输入电路的输出端读出各路状态，并将其写入输入映像寄存器；在程序执行阶段，CPU 从输入映像寄存器和元件映像寄存器中读出各继电器的状态，并根据此状态执行用户程序，执行结果再写入元件映像寄存器中；在紧接着的下一个 I/O 刷新阶段，将输出映像寄存器的状态写入输出锁存电路，再经输出电路传递到输出端子，从而控制外接器件动作。

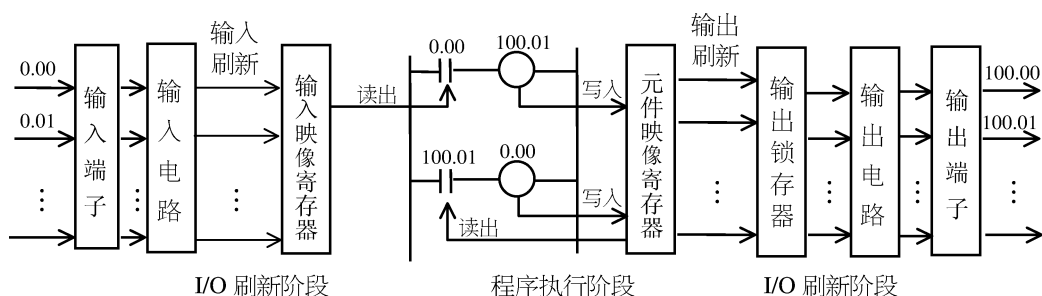


图 1-6 从输入到输出的信号传递过程

PLC 的循环扫描工作方式也为 PLC 提供了一条死循环自诊断功能。PLC 内部设置了一个监视定时器 WDT，其定时时间可由用户设置为大于用户程序的扫描周期，PLC 在每个扫描周期的公共处理阶段将监视定时器复位。在正常情况下，监视定时器不会动作，如果由于

CPU 内部故障使程序执行进入死循环, 那么扫描周期将超过监视定时器的定时时间, 这时监视定时器动作, 运行停止, 以示用户。

在循环扫描过程中, 有时需要立即对某个信息进行处理, 以加快响应速度, 为此, PLC 中引入中断功能, 设有内置输入中断、定时中断、高速计数器中断、外部中断等, PLC 在循环扫描的各个阶段都可以响应中断信号。

1.4.2 PLC 的 I/O 滞后现象

PLC 有很多优越特性, 但也有不足之处, 其中之一就是 PLC 的 I/O 有响应滞后现象。产生 I/O 滞后现象的原因主要有 3 个方面:

1) 输入滤波器有时间常数。输入电路中的滤波器对输入信号有延迟作用, 时间常数越大, 延迟作用越大。

2) 输出继电器有机械滞后。从输出继电器的线圈通电到其触点闭合有一段时间, 这是输出电路的硬件参数, 如 CP1 输出继电器的滞后时间为 15ms。

3) PLC 循环操作时, 进行公共处理、I/O 刷新和执行用户程序等产生扫描周期。

在图 1-7a 给出的梯形图中, 从输入触点闭合到输出触点闭合有一段延迟时间, 称为 I/O 响应时间。图 1-7b 为最小 I/O 响应时间, 在 I/O 刷新阶段, 输入信号已经在输入电路的输出端反映出来, CPU 将其写入输入映像寄存器, 经过程序执行后, 结果在随后扫描周期的 I/O 刷新阶段被输出, 在这种情况下, I/O 响应时间最小, 为输入 ON 时间、一个扫描周期和输出 ON 时间之和。图 1-7c 为最大 I/O 响应时间, 在 I/O 刷新阶段刚结束, 输入信号恰好在输入电路的输出端反映出来, 由于错过了 I/O 刷新阶段, CPU 不能读取, 而要等到第一个扫描周期的 I/O 刷新阶段, 才能被 CPU 写入输入映像寄存器, 经过程序执行后, 结果在第二个扫描周期的 I/O 刷新阶段被输出, 在这种情况下, I/O 响应时间最大, 为输入 ON 时间、两个扫描周期和输出 ON 时间之和。

以 CP1 为例, I/O 响应时间计算如下:

输入 ON 延时: 8ms (默认设置)

公共处理 + I/O 刷新时间: 2ms

程序执行时间: 14ms

输出 ON 延时: 15ms

最小 I/O 响应时间 = $8 + (2 + 14) + 15\text{ms} = 39\text{ms}$

最大 I/O 响应时间 = $8 + (2 + 14) \times 2 + 15\text{ms} = 55\text{ms}$

由于 PLC 是在循环扫描的 I/O 刷新阶段才对 I/O 更新的, PLC 这种集中输入、集中输出的刷新方式是造成输出滞后输入的原因之一。对一般工业控制设备来说, 这些滞后现象是完全允许的。但某些场合下, 需要输出对输入做出快速响应, PLC 提供以下措施:

1) 执行 I/O 刷新指令。CP1 设置了 I/O 刷新指令 IORF (97), 在程序中执行该指令, 可随时对指定的 CPU 单元内置的或扩展单元的 I/O 通道进行刷新, 以缩短输出滞后输入的时间, 提高 I/O 响应速度。

2) 执行立即刷新指令。常用的输入指令是 LD/LD NOT、AND-AND NOT 和 OR/OR NOT, 常用的输出指令是 OUT/OUT NOT, CP1 为这些指令设置了立即刷新功能, 执行时立即对指令所访问的 CPU 单元内置的 I/O 通道 (字) 进行刷新, 刷新是以通道 (字) 为单位操

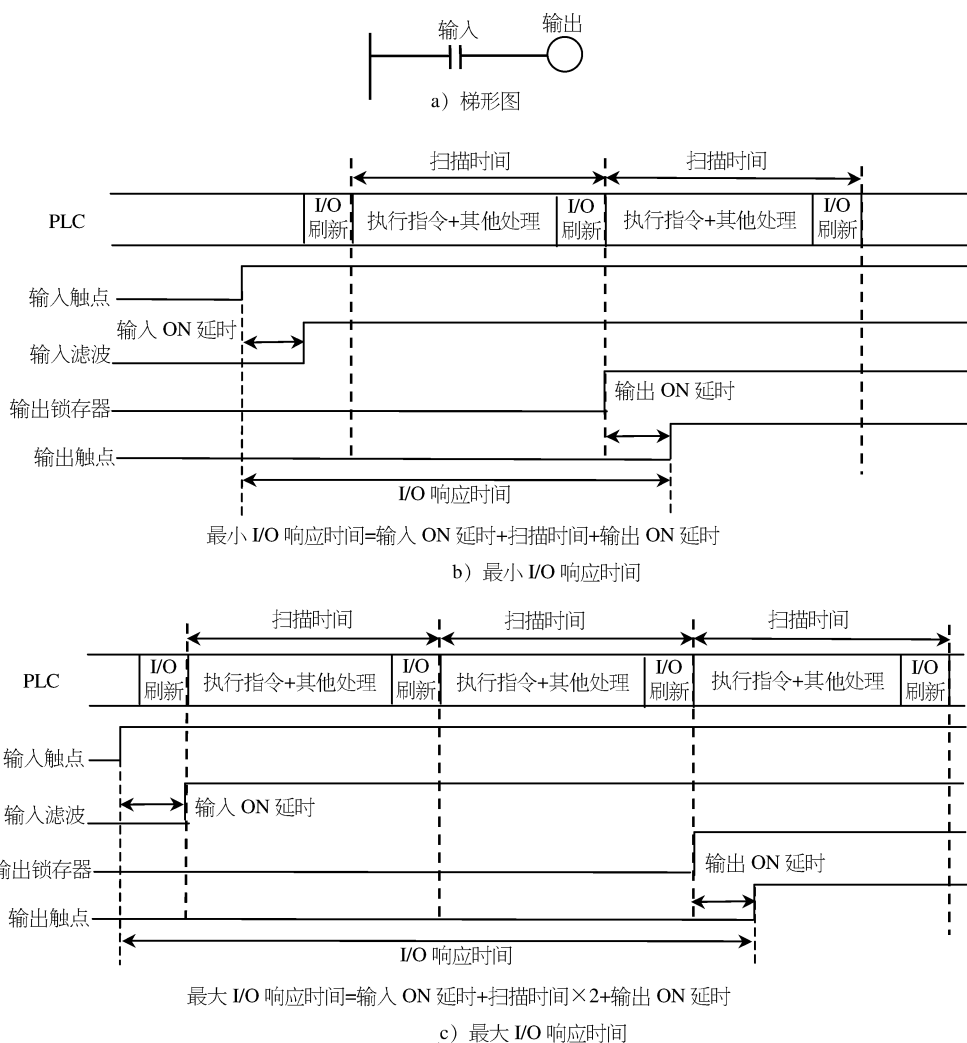


图 1-7 I/O 响应时间

作的，即刷新指定位时，对含该位的通道内的其他位也同时进行刷新。立即刷新指令如图 1-8 所示，包含!LD 和!OUT，图中第一行执行时，先对 PLC 内置输入通道 2 所有的 16 位立即读入，然后进行逻辑运算；第二行执行时，在已更新的输入通道 2 的数据基础上，进行逻辑运算，将结果立即输出，这时内置输出通道 100 所有的 16 位一起被更新。

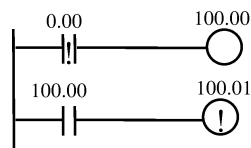


图 1-8 立即刷新指令

1.5 PLC 的编程语言

PLC 是专为工业控制而开发的装置，其主要使用者是工厂广大电气技术人员，为了适应他们的传统习惯和掌握能力，通常 PLC 不采用微机的编程语言，而常常采用面向控制过程、面向问题的“自然语言”编程，这些编程语言有梯形图、语句表、功能块、结构文本。

1.5.1 梯形图

梯形图 (LAD) 表达式是在原电器控制系统中常用的接触器、继电器梯形图基础上演变而来的。它与电气控制原理图相呼应, 形象、直观和实用, 广大电气技术人员很容易掌握, 是 PLC 的主要编程语言。图 1-9 所示为两种梯形图。

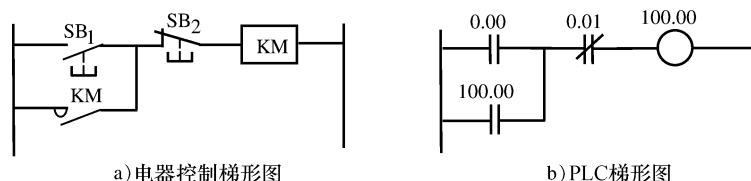


图 1-9 两种梯形图

由图 1-9 可以看出, PLC 梯形图在形式上类似于继电器控制梯形图。它是用图形符号——、、、、等连接而成, 这些符号依次为常开触点、常闭触点、并联连接、串联连接、继电器线圈。梯形图按自上而下、从左到右的顺序排列。一般每个继电器线圈对应一个逻辑行。梯形图的最左边是起始母线, 每一逻辑行必须从起始母线开始画起, 然后是触点的各种连接, 最后终于继电器线圈。梯形图的最右边是结束母线, 有时可以省去不画。

在梯形图中的每个编程元件应按一定的规则加注字母和数字串, 不同的编程元件常用不同的字母符号和一定的数字串来表示。

PLC 梯形图具有以下特点:

1) 梯形图中的继电器不是物理继电器, 每个继电器实际上是映像寄存器中的一位, 因此称为“软继电器”。相应位的状态为 1, 表示该继电器线圈通电, 其常开触点闭合, 常闭触点断开; 相应位的状态为 0, 表示该继电器线圈失电, 其常开触点断开, 常闭触点闭合。

梯形图中继电器线圈是广义的, 除了输出继电器、辅助继电器线圈外, 还包括定时器、计数器、移位寄存器以及各种算术运算等。

2) 每个继电器对应映像寄存器中的一位, 其状态可以反复读取, 因此可以认为继电器有无限多个常开触点和常闭触点, 在程序中可以反复引用。

3) 梯形图是 PLC 形象化的编程手段, 梯形图两端是没有任何电源可接的。梯形图中并没有真实的物理电流流动, 而仅是“概念”电流, 是用户程序解算中满足输出执行条件的形象表示方式。“概念”电流只能从左向右流动。

4) 输入继电器供 PLC 接收外部输入信号, 而不是由内部其它继电器的触点驱动, 因此梯形图中只出现输入继电器的触点, 而不出现输入继电器的线圈。输入继电器的触点表示相应的输入信号。

5) 输出继电器供 PLC 作输出控制用。它通过开关量输出模块对应的输出开关 (晶体管、双向晶闸管或继电器触点) 去驱动外部负载。因此, 当梯形图中输出继电器线圈满足接通条件时, 就表示在对应的输出点有输出信号。

6) 当 PLC 处于运行状态时, PLC 就开始按照梯形图符号排列的先后顺序 (从上到下、从左到右) 逐一处理, 也就是说, PLC 对梯形图是按扫描方式顺序执行程序。

7) 用户程序解算时, 输入触点和输出线圈状态是从 I/O 映像寄存器中读取的, 不是解算时现场开关的实际状态。梯形图中前面程序解算的结果马上可以被后面程序的解算所利用。

1.5.2 语句表

语句表 (STL) 是一种与汇编语言类似的助记符编程表达式。在许多小型 PLC 的编程器中没有 CRT 屏幕显示, 或没有较大的液晶屏幕显示, 就用一系列 PLC 操作命令组成的语句表将梯形图控制逻辑描述出来, 并通过编程器输入到机器中去。

不同的 PLC, 语句表使用的助记符不相同。以 OMRON 的 PLC 为例, 对应图 1-9b 的语句表为:

LD	0.00	(表示逻辑操作开始, 常开触点与母线连接)
OR	100.00	(表示常开触点并联)
AND NOT	0.01	(表示常闭触点串联)
OUT	100.00	(表示输出)

语句是用户程序的基础单元, 每个控制功能由一条或多条语句组成的用户程序来完成。每条语句是规定 CPU 如何动作的指令, 它的作用和微机的指令一样, PLC 的语句也由操作码和操作数组成, 故其表达式也和微机指令类似。

PLC 的语句: 操作码 + 操作数

操作码用来指定要执行的功能, 告诉 CPU 该进行什么操作; 操作数内包含执行该操作所必需的信息, 告诉 CPU 用什么地方的数据来执行此操作。

操作数应该给 CPU 指明为执行某一操作所需要信息的所在地, 所以操作数的分配原则是:

- 1) 为了让 CPU 区别不同的编程元件, 每个独立的元件应指定一个互不重复的地址。
- 2) 所指定的地址必须在该型机器允许的范围之内。超出机器允许的操作参数, PLC 不予响应, 并以出错处理。

语句表编程有键入方便、编程灵活的优点, 但不如梯形图形象、直观。

1.5.3 功能块

功能块 (FB) 是一个将处理功能标准化的基本程序单元。功能块由 PLC 生产厂家以库文件形式提供, 或由用户自行定义。

功能块定义时包括两部分: 算法和变量定义。

1. 算法

功能块的算法可由梯形图或结构文本编写, OMRON 由编程软件 CX-P (5.0 及以上版本) 实现。

2. 变量定义

变量定义要求列出每个变量的类型和属性。

(1) 变量类型

内部变量: 只能在功能块内部使用的变量。内部变量不能与功能块的 I/O 参数直接建立联系, 例如, 从功能块的输入参数直接获得数据, 或将数据直接从功能块输出。

输入变量：输入变量从功能块外面指定的输入参数中获得数据。每个功能块都有一个默认输入变量 EN，EN 为使能变量，功能块创建时自动生成，不需要用户定义，当 EN 处于 ON 时，功能块执行。

输出变量：输出变量可以将数据赋给功能块外面指定的输出参数。每个功能块都有一个默认输出变量 ENO，ENO 为使能输出变量，功能块创建时自动生成，不需要用户定义，功能块执行时，ENO 处于 ON 状态。

外部变量：外部变量是由 CX-P 定义保存的系统变量（条件标志和一些辅助区域位），或用户定义的全局符号（可用于每个实例中）。

(2) 属性

属性是为变量指定名称、数据类型、初始值等。

编程时，功能块嵌入到程序中，称作“例程”或“功能块例程”，每个例程要有一个名称。一个功能块可以创建多个例程嵌入到程序中，如果这些例程具有相同的名称，则它们使用相同的内部变量；如果这些例程具有不同的名称，则它们使用的内部变量不同。

在程序中每创建一个例程，就需要为功能块的输入变量、输出变量设置相应的参数：I/O 内存地址或者常数。

使用功能块编程便于程序结构化，具有更好的设计质量和更短的设计时间。一旦为功能块创建了标准程序并将其保存为文件，便可将功能块嵌入程序中并设置功能块的 I/O 参数，功能块可在程序中反复使用。当创建/调试程序时，反复使用现有功能块将节省大量的时间并减少编码错误，而且程序更容易理解。

OMRON 为用户提供了功能强大的功能块库文件，可安装在 CX-P (5.0 及以上版本) 中，使用非常方便。OMRON 新型号的 PLC (如 CS1、CJ1/CJ2、CP1 等) 支持功能块编程。

1.5.4 结构文本

随着 PLC 技术的发展，PLC 的高级功能越来越多，如果仍然用梯形图来实现，会很不方便。为了方便用户使用 PLC 的数学运算、数据处理、图形显示、打印报表等功能，许多大中型或先进的 PLC 都配备了结构文本 (ST) 编程语言。

结构文本是一种类似于 Pascal 的高级编程语言，使用诸如选择语句和迭代语句的语言结构。

结构文本使用语句来编写结构化的文本程序，语句用变量和方程式来配置。

方程式是包含运算符和操作数（变量或常数）的表达式。运算符包括算术运算符、比较运算符和逻辑运算符等。

语句既可以是赋值语句，也可以是控制语句。赋值语句将方程式计算的结果保存于变量中。控制语句包括选择语句和迭代语句等。

OMRON 的 CS1、CJ1/CJ2、CP1 等新型号的 PLC 均可使用结构文本编程，一般使用结构文本来创建功能块。OMRON 编程软件 CX-P (5.0 及以上版本) 支持结构文本编程。

实际上，熟悉计算机编程的人很容易使用结构文本，采用结构文本编程来实现一些更为复杂的数学运算并生成功能块不但极大地节省编程时间，而且使源程序清晰、易读，降低了出错率。

与梯形图相比，结构文本有两个很大优点，其一是能实现复杂的数学运算，其二是非常简洁和紧凑。

1.6 OMRON PLC 发展概况

PLC 的种类很多,其实现的功能、内存容量、控制规模、外形等方面均存在较大差异。PLC 的分类没有一个严格的统一标准,一般是按照结构形式、控制规模、实现的功能进行大致地分类。

1) 按结构分类:PLC 按照硬件的结构形式可以分为整体式和组合式。整体式 PLC 外观上是一个长方形箱体,又称箱体式 PLC。组合式 PLC 在硬件构成上具有较高的灵活性,其模块可以像拼积木似的进行组合,构成具有不同控制规模和功能的 PLC,因此这种 PLC 又称为模块式或积木式 PLC。

2) 按控制规模分类:控制规模主要是指控制开关量的输入、输出点数及模拟量的输入、输出路数。但主要以开关量计数,模拟量的路数可折合成开关量的点数,一般一路相当于 8~16 点。根据 I/O 点数的不同,PLC 大致可分为微型机、小型机、中型机及大型机、超大型机。

3) 按生产厂家分类:目前世界上能生产 PLC 的厂家较多,在中国市场占有较大份额、较有影响的公司有:德国西门子公司、日本 OMRON 公司、日本三菱公司、美国 ROCKWELL 公司、法国施耐德公司、日本松下电工公司等。

对于 PLC,一般应从基本性能、特殊功能及通信联网 3 个方面考察其性能。基本性能包括指令系统、工作速度、控制规模、程序容量、PLC 内部器件、数据存储器容量等。特殊功能指中断、A-D、D-A、温度控制等,模块式 PLC 的特殊功能是由智能单元完成的。通信联网是指 PLC 与各种外设通信及 PLC 组成各种网络,这一功能通常由专用通信板或通信单元完成。

日本 OMRON 公司是世界上生产 PLC 的著名厂家之一,OMRON 的大、中、小、微型机各具特色、各有所长,在中国市场上的占有率位居前列,在国内用户中享有较高声誉。

OMRON 公司从 20 世纪 80 年代至今,产品多次更新换代,下面依时间顺序对其发展情况做一简单回顾。

80 年代初期,OMRON 的大、中、小型机分别为 C 系列的 C2000、C1000、C500、C120、C20 等。这些型号的 PLC 指令少,而且指令执行时间长,内存也小,内部器件有限,PLC 体积大。例如,C20 仅 20 条指令,基本指令执行时间为 4~80 μ s。

随后小型机换代出现 P 型机,替代了 C20。P 型机 I/O 点数最多可达 148 点。指令增加到 37 条,指令执行的速度加快了,基本指令执行时间为 4 μ s,体积也明显缩小。P 型机具有较高的性能价格比,且易于掌握和使用,因而具有较强的竞争力,在当时的小型机市场上独占鳌头。

80 年代后期,OMRON 开发出 H 型机,大、中、小型对应有 C2000H/C1000H、C200H、C60H/C40H/C28H/C20H。大、中型机为模块式结构,小型机为整体式结构。H 型机的指令增加较多,有 100 多种,特别出现了指令的微分执行,一条指令可顶多条指令使用,为编程提供了方便。H 型机指令的执行速度又加快了,大型 H 型机基本指令执行时间为 0.4 μ s,而 C200H 机也只有 0.7 μ s。H 型机的通信功能增强了,甚至小型 H 型机也配有 RS-232C 口,与计算机可以直接通信。大型机 C2000H 的 CPU 可进行热备配置,其一般的 I/O 单元还可在线插拔。中型机 C200H 的特殊功能模块很丰富,结构合理,功能齐全,为当时中型机中较优

秀的机型, 获得了非常广泛的应用。C200H 曾用于太空实验站, 开创了业界先例。

90 年代初期, OMRON 推出无底板模块式结构的小型机 CQM1。CQM1 控制 I/O 点数最多可达 256 点。CQM1 的指令已超过 100 种, 它的速度较快, 基本指令执行时间为 $0.5\mu\text{s}$, 比中型机 C200H 还要快。CQM1 共有 7 种 CPU 单元, 每种 CPU 单元都带有 16 个输入点 (称为内置输入点), 有输入中断功能, 都可接增量式旋转编码器进行高速计数, 计数频率单相为 5kHz、两相为 2.5kHz。CQM1 还有高速脉冲输出功能, 标准脉冲输出可达 1kHz。此外, CPU42 带有模拟量设定功能, CPU43 有高速脉冲 I/O 端口, CPU44 有绝对式旋转编码器端口, CPU45 有 A-D、D-A 端口。CQM1 采用模块式结构, 有 A-D、D-A、温控等特殊功能单元和各种通信单元。CQM1 的 CPU 单元除 CPU11 外都自带 RS-232C 通信口。

在 CQM1 推出之前, OMRON 推出大型机 CV 系列, 其性能比 C 系列大型 H 型机有显著的提高, 增强了 OMRON 在大型机方面的竞争实力。1998 年底, OMRON 推出了 CVM1D 双机热备系统, 它具有双 CPU 单元和双电源单元, 不仅 CPU 可热备, 而且电源也可热备。CVM1D 继承了 CV 系列的各种功能, 可以使用 CV 的 I/O 单元、特殊功能单元和通信单元。CVM1D 的 I/O 单元可在线插拔。

值得注意的是进入 90 年代后, OMRON 更新换代的速度明显加快, OMRON 在中型机和小型机上又有不少技术更新。

中型机从 C200H 发展到 C200HS。C200HS 于 1996 年进入中国市场, 到了 1997 年全新的中型机 C200HX/HG/HE (统称为 α 机) 又来了。它的性能比 C200HS 又有显著提高。除基本性能比 C200HS 提高外, α 机的突出特点是它的通信组网能力强。例如, CPU 单元除自带的 RS-232C 口外, 还可插上通信板, 板上配有 RS-232C、RS-422/RS-485 口, α 机使用协议宏功能指令, 通过上述各种串行通信口与外围设备进行数据通信。 α 机可加入 OMRON 的高层信息网 Ethernet 网, 还可加入中层控制网 Controller Link 网, 而 C200H、C200HS 不可以。

1999 年 OMRON 在中国市场上又推出 α 机的后续机型, 即功能更加完美的 CS1 机, CS1 兼容了 α 机的功能, 性能上做了很大的改进。CS1 属大中型机, 分为 CS1G、CS1H、CS1D 3 个子系列。

OMRON 在小型机方面也取得了长足的进步。1997 年, OMRON 在推出 α 机的同时, 就推出 P 型机的升级产品, 即小型机 CPM1A。与 P 型机相比, CPM1A 体积很小, 只及同样 I/O 点数 P 型机的 1/2, 但是它的性能改进很大, 例如, 它的指令有 93 种、153 条, 基本指令执行时间为 $0.72\mu\text{s}$, 程序容量达 2048 字, 单相高速计数达 5kHz (P 型机为 2kHz)、两相为 2.5kHz (P 型机无此功能), 有脉冲输出、中断、模拟量设定、子程序调用、宏指令功能等。通信功能也增强了, 可实现 PLC 与 PLC 链接、PLC 与上位机通信、PLC 与 PT 链接。

1999 年前后, OMRON 在推出 CS1 系列的同时, 在小型机方面相继推出 CPM2A/CPM2C/CPM2AE、CQM1H 等机型。

CPM2A 是 CPM1A 之后的另一系列机型。CPM2A 的功能比 CPM1A 有新的提升, 例如, CPM2A 指令的条数增加、功能增强、执行速度加快, 可扩展的 I/O 点数、PLC 内部器件的数目、程序容量、数据存储器容量等也都增加了; 所有 CPM2A 的 CPU 单元都自带 RS-232C 口, 在通信联网方面比 CPM1A 改进不少。

CPM2C 具有独特的超薄、模块化设计。它有 CPU 单元和 I/O 扩展单元, 也有模拟量 I/O、温度传感和 CompoBus/S I/O 链接等特殊功能单元。CPM2C 的 I/O 采用 I/O 端子台或 I/O

连接器形式。每种单元的体积都极小，仅有 90mm × 65mm × 33mm。CPU 单元使用 DC 电源，共有 10 种型号，输出是继电器或晶体管形式，有的 CPU 单元带时钟功能。CPM2C 的 I/O 扩展单元共有 10 种型号，输出有继电器或晶体管形式。CPM2C 最多可扩展到 140 点，单元之间通过侧面的连接器相连。CPU 单元有 RS-232C 口。CPM2C 使用专用的通信接口单元 CPM2C-CIF01/CIF02，可把外设端口转换为 RS-232C 口或 RS-422/RS-485 口。CPM2C CPU 单元的基本性能、特殊功能和通信联网的功能与 CPM2A 相一致。

CPM2AE 是 OMRON 公司专为中国市场开发的，该机型仅有 60 点继电器输出的 CPU 单元，是 CPM2A-60CDR-A 的简化机型。CPM2AE 删除 CPM2A 的一些功能以减少成本，降低售价。被删除的功能主要有：后备电池（可选）、RS-232C 端口、CTBL 指令（寄存器比较指令）等。其它功能则与 CPM2A 完全相同。

CQM1H 是小型机 CQM1 的升级换代产品。在延续原先 CQM1 所有优点的基础上，提升并充实了 CQM1 的多种功能。CQM1H 对 CQM1 有很好的兼容性，对原先使用 CQM1 的老用户来说，升级换代十分方便。CQM1H 在三大性能方面做了较大的提升和充实：I/O 控制点数、程序容量和数据容量均比 CQM1 翻一番；提供多种先进的内装板，能胜任更加复杂和柔性的控制任务；CQM1H 可以加入 Controller Link 网，还支持协议宏通信功能。

OMRON 小型机的发展极为迅速，从 80 年代的 C120、C20、C60P 等，到 90 年代的 CPM1、CPM1A、CQM1 等，再到 2000 年的 CPM2A/2AE、CPM2C、CQM1H 等系列。在当今越来越激烈的竞争环境下，为进一步降低成本和改进质量，创造更大的附加值，OMRON 开发了面向 21 世纪的新机型 CJ1/CJ2 系列，它具有更快的处理能力、更小的尺寸、更迅速的数据传送等特点，满足高速、小型、无缝的要求，能在先进的机器控制领域发挥小巨人的作用，而且在设计和开发方面采用了以任务为单位的编程方式，大大提高了设计效率。

2005 年之后，OMRON 公司为适应市场的需要，又推出了新一代的 CP1 系列小型机。CP1 包括 CP1H、CP1L 和 CP1E 3 个子系列。CP1 具有“高度扩展性”的多功能一体 PLC，它简易、灵活、多功能一体、开放和紧凑的特点不仅迎合了现代小型 PLC 的发展趋势，而且满足了用户对高功能、高度集成、高维护性能、低成本追求。它具有和 OMRON CJ1/CS1 系列兼容的体系结构和指令系统；尺寸与以往小型机 CPM* 系列相同，但处理速度提高了 10 倍。通过 CP1 实现高速处理、多种功能的组合，能方便地解决用户的各种问题。

目前，OMRON 主推的 PLC 机型：大中型机 CS1，小型机 CJ1/CJ2、CP1。

2011 年，OMRON 推出新一代 NJ 系列控制器，NJ 从硬件到软件全面升级，它保持了 PLC 高可靠性的特点，功能大幅跃进。NJ 实际为可编程自动化控制器，鉴于其在运动控制方面的卓越能力，故称 NJ 为机器自动化控制器。

1.7 OMRON 可编程自动化控制器

众所周知，PLC 以高可靠性著称，其硬件芯片的选用相对保守。基于系统可靠性考虑，PLC 并不像计算机那样总是采用最先进的芯片，而是选用那些经过考验证明为非常成熟、十分可靠的芯片。例如，目前应用于个人计算机的 Intel Core i7 系列的 CPU，其工作频率达到 3.2GHz，而 PLC 的 CPU 与计算机的相比，处在相当落后的地步，很多仍停留在单机的水平，它的工作频率低，一般不超过 0.1GHz，这样低的工作速度限制了 PLC 自身的能力，在

面对运算量大、实时性要求高的控制任务时, PLC 往往力不从心。随着机器自动化水平的提高, 对运动控制的要求越来越高, 而 PLC 解决复杂的运动控制任务时, 需配置运动控制单元或位置控制单元, CPU 单元要管理和协调这些高功能单元, 由于 PLC 采用循环扫描工作方式, 受扫描周期时间的限制, CPU 单元与高功能单元交换信息时无法达到理想的高速度, 而且控制程序编写较为麻烦, 有一些精密、高速、复杂的运动控制尚无法完成。到目前为止, PLC 拘泥于称为 DM、CIO 等硬件实际物理地址的编程, 程序的重用性比较差; 而且, 运动编程使用与高功能单元机号相关特定的一段 DM、CIO 区域, 一旦高功能单元机号发生改变, 程序就无法再用了, 必须修改。

针对目前 PLC 存在的问题, OMRON 公司从硬件体系和软件体系进行全新的架构设计, 2011 年推出具有划时代意义的新产品: NJ 系列的机器自动化控制器。

NJ 从外观上看, 是一台小巧玲珑的 PLC, 但从内部的硬件结构上看, 其配置相当于一台个人计算机, 例如, CPU 采用时下先进的微处理器, 工作频率达到 1.66GHz, 比较单片机的 0.1GHz 的工作频率, 两者显然不在一个档次上。全新的设计为 NJ 整体性能跃升打下坚实的基础。

1.7.1 NJ 的特点

NJ 凝聚了机器自动化所需要的各种高端功能, 它的主要特点如下:

1. 继续保持 PLC 高可靠性的优良传统

OMRON 与世界上著名的芯片生产厂商 Intel 进行战略合作, 由对方提供最先进的 CPU 等核心器件, 集几十年生产 PLC 的专业经验, 开发推出的 NJ 具有与 CJ 系列 PLC 相同的质量水平和系统鲁棒性。

2. 指令系统与国际标准 IEC61131-3 完全接轨

为了提高编程的生产性, 实现了以变量为基础的编程, 开发了与之相对应的指令体系。指令体系不再是欧姆龙独立封闭的, 而是遵循全球编程语言标准 IEC61131-3 和运动控制标准 PLCOpen 的。

NJ 指令的执行速度远远高于目前的 PLC, 是 CJ2H 的 4 倍, 例如, LD 指令的执行时间: NJ501 为 1.9ns, CJ2H 为 16ns; LREAL 指令的执行时间: NJ501 为 26ns, CJ2H 为 6000ns。

3. 实现高速、复杂、精密的运动控制

传统 PLC 的运动控制功能是通过高功能单元实现的, PLC 引擎和运动引擎从硬件上是分开的, 而 NJ 是将两者集成到同一个硬件上, 如图 1-10 所示。NJ 的顺序控制和运动控制在同一个任务中按同一时序执行, 消除了复杂握手和数据交换处理花费时间所引起的损失。由于 PLC 和运动控制的数据在同一时序里处理, 可以改善数据跟踪的精度, 也能实现数据跟踪的仿真。NJ 取消了运动控制单元, 通过在控制器里搭载的运动控制功能, 达到比运动控制单元更好的性能。

NJ 借助高速 EtherCAT 通信进行运动控制, 最多可扩展到 64 轴伺服驱动。EtherCAT 通信与 I/O 刷新同步进行, 各从站节点间通过分布时钟功能达到 1 μ s 内同步, EtherCAT 具有 100Mbit/s 超高速通信, 可实现 1ms 周期的运动控制。

4. 丰富的通信与网络功能

利用 NJ 控制器自身内置的 EtherCAT 端口和 EtherNet/IP 端口, 可以方便地搭建 Ether-

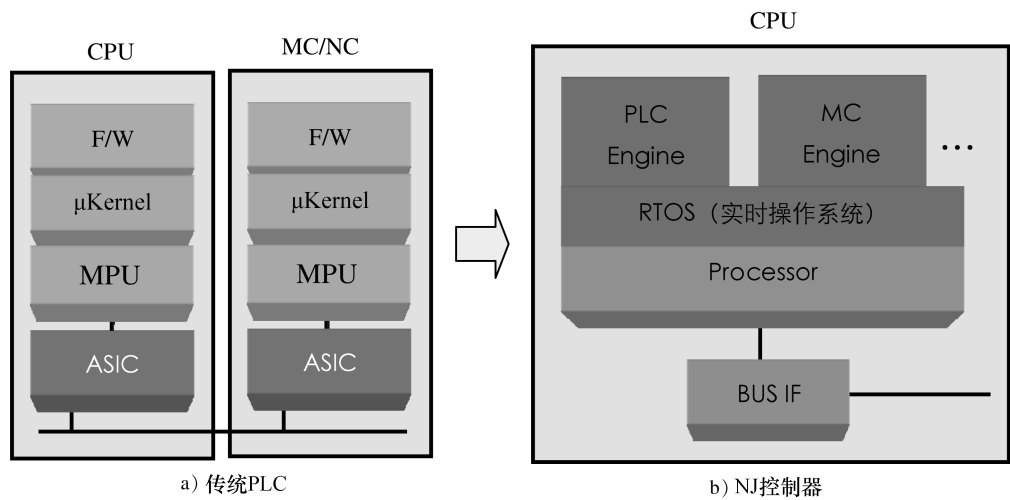


图 1-10 运动控制比较

CAT 网和 EtherNet/IP 网。利用 EtherCAT 通信，除了完成快速、高效的运动控制外，还可以实现远程 I/O 控制，起到现场总线的作用。上位计算机、人机界面和 NJ 控制器可以组成标准的 EtherNet/IP 网络。此外，利用 CJ 系列的通信、网络高功能单元，可以实现串行通信及搭建 DeviceNet 现场总线。

5. 功能强大的自动化平台软件 Sysmac Studio

全新开发的自动化平台软件 Sysmac Studio 集成了机器自动化需要的各种功能，如设置、编程、调试和维护等。

NJ 系列的性能指标见表 1-1。

表 1-1 NJ 系列的性能指标

项 目		NJ501-1300	NJ501-1400	NJ501-1500
编程	程序容量	20MB (执行对象和变量表 (包括变量名程))		
	变量的内存容量	具有保持属性的变量 (不包括 CJ 系列单元的保持、DM 和 EM 区内存)		
		2MB		
	CJ 系列单元内存 (变量可以用 AT 规格指定)	非保持属性的变量 (不包括 CJ 系列单元的 CIO 和工作区内存)		
		4MB		
		CIO 区	6144 字 (CIO 0 ~ CIO 6143)	
		工作区	512 字 (W0 ~ W511)	
		保持区	1536 字 (H0 ~ H1535)	
		DM 区	32768 字 (D0 ~ D32767)	
		EM 区	32768 字 × 25 存储体 (E0_00000 ~ E18_32767)	

(续)

项 目		NJ501-1300	NJ501-1400	NJ501-1500
单元配置	连接单元的最大数	每个 CPU 机架或扩展机架最多：10 个单元 整个控制器：40 个单元		
	扩展机架数	最多 3		
	I/O 容量	最多 2560 点，加上 EtherCAT 从站 I/O 容量		
	CPU 机架和扩展机架的供电单元	NJ-P□3001		
运动控制	可控制的伺服驱动器	内置 EtherCAT 通信的 OMRON G5 系列伺服驱动器 推荐单元版本：版本 2.1 或更高		
	可控制的编码器输入端子	OMRON GX 系列 GX-EC0211/EC0241 EtherCAT 远程 I/O 端子 推荐单元版本：版本 1.1 或更高		
	控制方法	使用 EtherCAT 通信的控制命令		
	控制模式	位置控制 (周期同步位置控制模式) 速度控制 (周期同步速度控制模式) 扭矩控制 (周期同步扭矩控制模式)		
	控制轴的数量	控制轴的最大数量	16 轴	32 轴
		单轴控制	最多 16 轴	最多 32 轴
		线性插补控制	每个轴组最多 4 轴	
		圆弧插补控制	每个轴组最多 2 轴	
	轴组数量		最多 32 个轴组	
	凸轮	凸轮数据点数量	每个凸轮表最多 65535 点 所有凸轮表总计最多 1048560 点	
		凸轮表数量	最多 640 个表	
通信	外设 USB 端口	支持的服务	Sysmac Studio 连接	
		物理层	符合 USB 2.0 标准的 B 型连接器	
		传输距离	最长 5m	
	内置 EtherNet/IP 端口	通信协议	TCP/IP、UDP/IP 和 BOOTP 客户端	
	内置 EtherNet/IP 端口	支持的服务	Sysmac Studio 连接，标签数据链接，CIP 信息通信，Socket 服务，FTP 服务器，自动时钟调整 (NTP 客户端)，SNMP 代理和 DNS 客户端	
		物理层	10Base-T 或 100Base-TX	
		介质访问方法	CSMA/CD	
		调制	基带	
		拓扑	星形	
		波特率	100Mbit/s (100Base-TX)	
		传输介质	Ethernet 5 类或更高的 STP (屏蔽双绞线)	
		传输距离	最长 100m (集线器与节点之间的距离)	
		级联连接数	如果使用交换集线器，没有任何限制	
	CIP 服务：标签数据链接 (周期通信)		—	

(续)

项 目				NJ501-1300	NJ501-1400	NJ501-1500
	内置 EtherNet/IP 端口		连接数	32		
			允许通信频段	1000pps (包括 heartbeat)		
			标签组数量	32		
	内置 EtherCAT 端口	通信协议		特定的 EtherCAT 协议		
		支持的服务		CoE (PDO 通信和 SDO 通信)		
		同步通信		DC (分布式时钟)		
		物理层		100Base-TX		
		调制		基频		
		波特率		100Mbit/s (100Base-TX)		
		双工模式		自动		
		拓扑		线形, 菊花链和分支		
		传输媒体		使用 Ethernet 5 类 (100Base-TX) 或更高的屏蔽双绞线 (用铝带双层屏蔽和编织)		
		传输距离		节点之间的距离: 最长 100m		
		最大从站数		192		

1.7.2 使用 NJ 的操作步骤

使用 NJ 时, 操作步骤比使用 PLC 要复杂一些, 以下列出主要步骤, 更详细的内容可参见相关手册。

1. 软件设计

设计总体的系统配置、任务配置、程序和变量。

- (1) 设计 I/O 和处理
- (2) 设计任务
- (3) 设计程序

2. 软件设置和编程

在支持软件上, 创建系统配置并分配变量。创建任务和程序, 利用模拟仿真功能离线调试它们。

- (1) 从站和单元配置
- (2) 控制器设置
- (3) 编程
- (4) 离线调试

3. 安装和设置硬件

安装硬件单元并做必要的硬件设置。

4. 接线

进行网络配线和 I/O 接线。

5. 确认运行正常和启动实际系统运行

将支持软件和实际系统相连并下载工程。先初步检查实际系统的运行, 确认正常后, 启动实际系统运行。

第 2 章 NJ 的硬件与配置

2.1 NJ 的硬件

2.1.1 NJ 的硬件单元

1. CPU 单元

NJ 系列的 CPU 单元规格见表 2-1。

表 2-1 NJ 系列的 CPU 单元的规格

型号	I/O 容量/配置单元的最大数量 (扩展机架的最大数量)	程序容量	变量的内存容量	运动轴的数量
NJ501-I300	2560 点/40 单元 (3 个扩展机架)	20MB	2MB: 断电保持 4MB: 断电不保持	16
NJ501-I400				32
NJ501-I500				64

NJ 系列的 CPU 单元如图 2-1 所示。

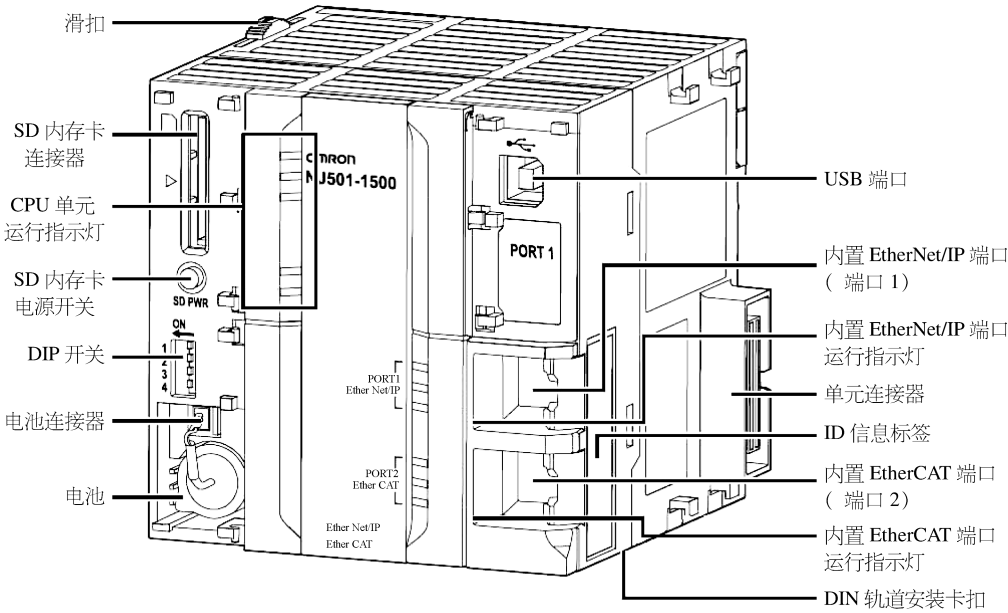


图 2-1 NJ 系列的 CPU 单元

CPU 单元运行指示灯的含义见表 2-2。

表 2-2 CPU 单元运行指示灯的含义

指示灯	颜色	状 态	含 义
RUN	绿色	亮	CPU 单元在 RUN 模式下正常运行 (用户程序正在执行)
		闪烁 (3s 间隔)	出现不正确供电单元连接错误 (CJ 系列供电单元连接在 CPU 机架上) CPU 单元不运行并且错误指示灯亮
		闪烁 (1s 间隔)	CPU 单元启动 (启动时进入 RUN 模式或 PROGRAM 模式)
		不亮	停止在 PROGRAM 模式下, 当 CPU 单元处于复位过程中, 或当出现下面任何的错误时: ● 重大故障级的控制器错误 ● CPU 错误 (WDT 错误)
ERROR	红色	亮	自诊断发现下列错误之一: ● 重大故障级的控制器错误 ● CPU 错误 (WDT 错误) CPU 单元运行将停止, 所有输出都会置 OFF
		闪烁 (1s 间隔)	自诊断发现下列错误之一: ● 局部故障级的控制器错误 (功能模块中所有控制被禁用) ● 轻微故障级的控制器错误 (功能模块中部分控制被禁用) 出现其中错误之一时, CPU 单元将继续运行。 出现局部故障级控制器错误的功能模块, 其运行将停止。 出现轻微故障级控制器错误的功能模块, 其运行将继续
		不亮	CPU 单元处于正常运行状态, CPU 复位处于进行过程中, 或出现监视信息 (CPU 单元运行将继续)
BUSY	黄色	闪烁	内置非易失性存储器访问正在进行
		不亮	其他情况
SD PWR	绿色	亮	正在为 SD 存储卡供电, SD 存储卡可以使用
		闪烁	……
		不亮	停止为 SD 存储卡供电, SD 存储卡没有安装, 或安装的 SD 存储卡的文件格式不支持
SD BUSY	黄色	亮	正在访问 SD 存储卡
		不亮	没有访问 SD 存储卡

当 BUSY 指示灯闪烁时, 不要断掉控制器的电源。CPU 单元的用户程序和设置正在内置的非易失性存储器中后备。如果停电, 数据不能后备, 下一次控制器启动时, 会出现重大故障级的控制器错误, 运行停止。

正在访问 SD 存储卡时 (也就是 SD BUSY 指示灯闪烁时), 不要断电或移走 SD 存储卡。否则, 数据可能会损坏, 控制器使用损坏的数据会导致不正确的运行。在 CPU 单元供电时, 如果要从 CPU 单元上移走 SD 存储卡, 需按下 SD 存储卡供电电源开关, 等到 SD PWR 指示灯灭, 然后再移走 SD 存储卡。

通过电源单元和 CPU 单元前面板上的 PWR、RUN 和 ERROR 指示灯, 可以检查 CPU 单元的运行状态。如判断用户程序执行状态; 与 Sysmac Studio 或 PT 的通信状态; 出现的错

误是致命性的还是非致命性的，进一步确定是致命错误中的哪一类或是非致命错误中的哪一类。

内置 EtherNet/IP 端口 (端口 1) 指示灯的含义见 8.1 节。

内置 EtherCAT 端口 (端口 2) 指示灯的含义见 6.1 节。

DIP 开关不能用来对 NJ 控制器的用户程序进行写保护。需要时，从 Sysmac Studio 的 CPU 单元的“保护设置/启动写保护设置”设置写保护。通常情况下，CPU 单元的 DIP 开关置为 OFF。将 DIP 开关的引脚 4 置为 ON 时，如图 2-2 所示，控制器处于安全模式。

电池用于断电保护。在供电电源断电后，NJ 中具有保持属性的变量的值，CJ 系列单元 H、DM 和 EN 区的字，时钟信息，某些系统定义变量以及事件日志，会保持断电前的值。电池型号为 CJ1W-BAT01，使用寿命为 5 年。

ID 信息标签贴在 CPU 单元的右侧面，上面有 CPU 单元型号、CPU 单元版本、CPU 单元的批号和序列号、CPU 内置端口的 MAC 地址。

2. SD 存储卡

推荐使用 OMRON 存储卡，如图 2-3 所示，型号为 HMC-SD291，容量为 2GB，格式为 FAT16，写入次数为 100000 次。

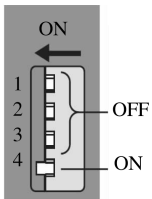


图 2-2 安全模式 DIP 设置



图 2-3 存储卡单元

当存储卡上的写保护开关置于 LOCK 位置时，不能对其进行写操作，这样可以避免对内容不正确的覆盖。

用户程序中使用指令，可以读写 SD 存储卡中的文件。在 EtherNet/IP 网络，FTP 客户端可以读写 SD 存储卡中的文件。

3. 电源单元

有两种电源单元类型：一种是交流供电的，型号为 NJ-PA3001，供电电压为 AC 100 ~ 240V；另一种是直流供电的，型号为 NJ-PD3001，供电电压为 DC 24V。

NJ 系列的 CPU 机架或扩展机架只能使用 NJ 系列的电源单元。NJ 系列的 CPU 单元不能使用 CJ 系列的电源单元，同样 CJ 系列的 CPU 单元也不能使用 NJ 系列的电源单元。

两种电源单元为 CPU 机架或扩展机架提供 DC 5V，还提供 DC 24V，它们的输出功率都是 30W，系统配置时，注意电源单元对外提供的电流、功率不要超过其额定的容量。

4. CJ 系列基本 I/O 单元

NJ CPU 单元可以使用 CJ 系列的基本 I/O 单元和高功能单元。高功能单元是指特殊 I/O 单元和 CPU 总线单元。可用的特殊 I/O 单元包括 CJ 系列的 A-D、D-A、温度控制、ID 传感器和高速计数等单元，可用的 CPU 总线单元包括 CJ 系列的串行通信单元和 DeviceNet 单元等。

限于篇幅，以下仅介绍 CJ 系列基本 I/O 单元。

(1) CJ 系列基本输入单元

CJ 系列基本输入单元的规格见表 2-3。

表 2-3 CJ 系列基本输入单元的规格

名 称	型 号	规 格
DC 输入单元	CJ1W-ID201*	8 点, 端子排, DC 12 ~ 24V
	CJ1W-ID211	16 点, 端子排, DC 24V
	CJ1W-ID212	
	CJ1W-ID231	32 点, 富士通连接器, DC 24V
	CJ1W-ID232	32 点, MIL 连接器, DC 24V
	CJ1W-ID233	
	CJ1W-ID261	64 点, 富士通连接器, DC 24V
	CJ1W-ID262	64 点, MIL 连接器, DC 24V
AC 输入单元	CJ1W-IA201*	8 点, 端子排, AC 200 ~ 240V
	CJ1W-IA111	16 点, 端子排, AC 100 ~ 120V
中断输入单元	CJ1W-INT01	16 点, 端子排, DC 24V
快速响应输入单元	CJ1W-IDP01	16 点, 端子排, DC 24V
B7A 接口单元	CJ1W-B7A14	64 点

* 尽管分配 16 个 I/O 位, 其中只有 8 个可用于外部 I/O。在单元配置时, 这些单元被视为 16 点 I/O 单元。

(2) CJ 系列基本输出单元

CJ 系列基本输出单元的规格见表 2-4。

表 2-4 CJ1 系列基本输出单元的规格

名 称	型 号	规 格
继电器输出单元	CJ1W-OC201*	8 点, 独立接点; 端子排, AC 250V/DC 24V, 2A
	CJ1W-OC211	16 点; 端子排, AC 250V/DC 24V, 2A
晶闸管输出单元	CJ1W-OA201*	8 点; 端子排, AC 250V, 0.6A
晶体管输出单元 漏型	CJ1W-OD201*	8 点; 端子排, DC 12 ~ 24V, 2A
	CJ1W-OD203*	8 点; 端子排, DC 12 ~ 24V, 0.5A
	CJ1W-OD211	16 点; 端子排, DC 12 ~ 24V, 0.5A
	CJ1W-OD213	16 点; 端子排, DC 24V, 0.5A
	CJ1W-OD231	32 点; 富士通连接器, DC 12 ~ 24V, 0.5A
	CJ1W-OD233	32 点; MIL 连接器, DC 12 ~ 24V, 0.5A
	CJ1W-OD234	32 点; MIL 连接器, DC 24V, 0.5A
	CJ1W-OD261	64 点; 富士通连接器, DC 12 ~ 24V, 0.3A
	CJ1W-OD263	64 点; MIL 连接器, DC 12 ~ 24V, 0.3A
晶体管输出单元 源型	CJ1W-OD202*	8 点, 负载短路保护, 断线检测; 端子排, DC 24V, 2A
	CJ1W-OD204*	8 点, 负载短路保护; 端子排, DC 24V, 0.5A
	CJ1W-OD212	16 点, 负载短路保护; 端子排, DC 24V, 0.5A
	CJ1W-OD232	32 点, 负载短路保护; MIL 连接器, DC 24V, 0.5A
	CJ1W-OD262	64 点; MIL 连接器, DC 12 ~ 24V, 0.3A
B7A 接口单元	CJ1W-B7A04	64 点

* 尽管分配 16 个 I/O 位, 其中只有 8 个可用于外部 I/O。在单元配置时, 这些单元被视为 16 点 I/O 单元。

(3) CJ 系列基本混合 I/O 单元

CJ 系列基本混合 I/O 单元见表 2-5。

表 2-5 CJ 系列基本混合 I/O 单元

名 称		型 号	规 格
DC 24V 输入/晶体管输出单元	漏型	CJ1W-MD231	富士通连接器 输入：16 点，DC 24V 输出：16 点，DC 12 ~ 24V，0.5A
		CJ1W-MD261	富士通连接器 输入：32 点，DC 24V 输出：32 点，DC 12 ~ 24V，0.3A
		CJ1W-MD233	MIL 连接器 输入：16 点，DC 24V 输出：16 点，DC 12 ~ 24V，0.5A
		CJ1W-MD263	MIL 连接器 输入：32 点，DC 24V 输出：32 点，DC 12 ~ 24V，0.3A
	源型	CJ1W-MD232	MIL 连接器 输入：16 点，DC 24V 输出：16 点，负载短路保护；DC 24V，0.5A
TTL I/O 单元		CJ1W-MD563	MIL 连接器 输入：TTL (DC 5V)，32 点 输出：TTL (DC 5V，35mA)，32 点
B7A 接口单元		CJ1W-B7A22	输入：32 点 输出：32 点

5. B7A 接口单元

如图 2-4 所示，B7A 为 1:1 传输路径，不需要主站。使用 3 芯 VCTF 电缆 (最大长度为 500m)，可以传送总计 16 个信号。B7A 接口单元型号为 CJ1W-B7A□□，为 CJ 系列基本 I/O 单元，使用 B7A 传输路径，可以与 B7A 链接端子交换多达 64 点的 I/O 数据。

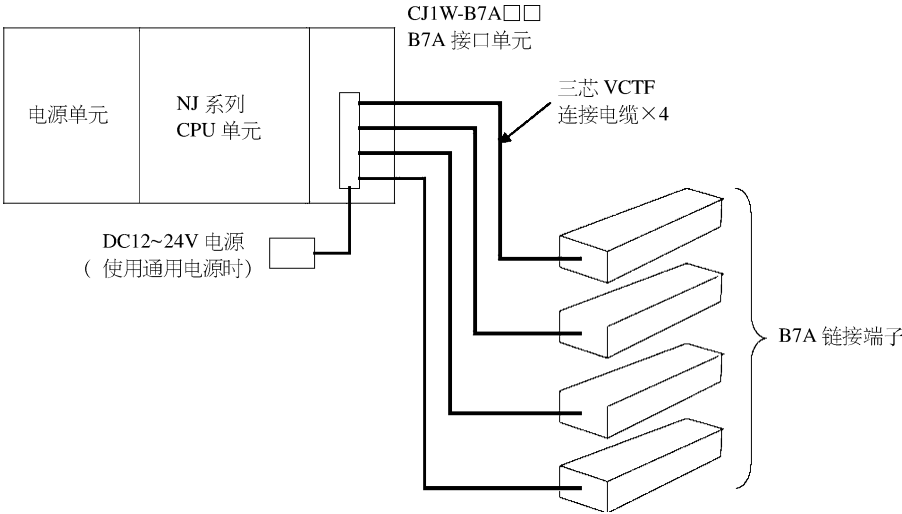


图 2-4 B7A 接口单元

B7A 接口单元和 B7A 链接端子可以像标准的基本 I/O 单元和 I/O 端子一样使用，不需要担心通信问题。当使用多个远程传感器或执行器时，这种用法显著减少了配线。

B7A 接口单元规格见表 2-6。

表 2-6 B7A 接口单元规格

B7A 接口单元	规 格	分配给单元的 I/O 字	可连接的 B7A 链接端子
CJ1W-B7A14	64 点输入 (4 个 B7A 端口)	4 个输入字	输入：4 个 16 点输入端子，2 个 32 点输入端子，或 2 个 16 点的输入端子和 1 个 32 点输入端子
CJ1W-B7A04	64 点输出 (4 个 B7A 端口)	4 个输出字	输出：4 个 16 点输出端子或 2 个 32 点输出端子
CJ1W-B7A22	32 点输入，32 点输出 (4 个 B7A 端口)	2 个输入字 2 个输出字	输入：2 个 16 点输入端子或 1 个 32 点输入端子 输出：2 个 16 点输出端子或 1 个 32 点输出端子或 2 个混合 I/O 端子 (16 输入/16 输出)

6. CJ 系列的 I/O 控制单元和 I/O 接口单元

CJ 系列的 I/O 控制单元和 I/O 接口单元用于 CPU 机架与扩展机架的连接，进行系统扩展。I/O 控制单元如图 2-5 所示，型号为 CJ1W-IC101。I/O 接口单元如图 2-6 所示，型号为 CJ1W-II1011。

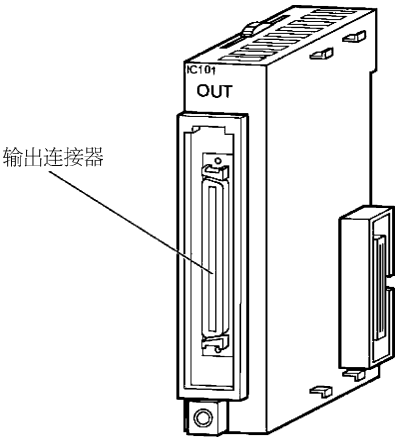


图 2-5 CJ1W-IC101 I/O 控制单元

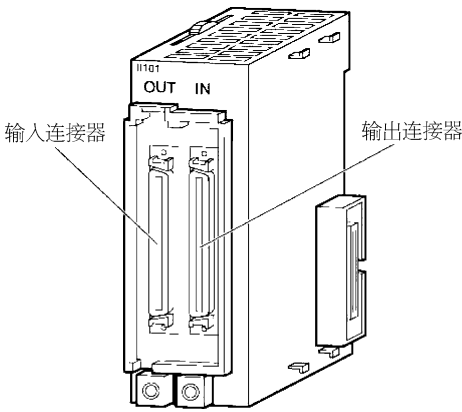


图 2-6 CJ1W-II101 I/O 接口单元

I/O 控制单元安装在 CPU 机架上，紧靠在 CPU 单元的右侧。每一个 CJ 扩展机架都需要一个接口单元。

2.1.2 CPU/扩展机架

NJ 的 CPU 机架如图 2-7 所示，由 NJ 系列电源单元、NJ 系列 CPU 单元、CJ 系列配置单元、I/O 控制单元和端盖组成。CJ 系列配置单元包括 CJ 系列基本 I/O 单元、特殊 I/O 单元和 CPU 总线单元。I/O 控制单元是在连接扩展机架时安装的，紧靠在 CPU 单元的右侧。

NJ 的扩展配置如图 2-8 所示，NJ 在 CPU 机架上安装一个 I/O 控制单元，在每个扩展机架上安装一个 I/O 接口单元，最多允许安装 3 个扩展机架，每个扩展机架最多安装 10 个 I/O

单元。这样组成一个包含多达 40 个单元的系统，即 10 个单元的 CPU 机架和 30 个单元 (10 × 3) 的扩展机架。使用专用的 CS/CJ 系列的 I/O 连接电缆，连接 CPU 机架上的 I/O 控制单元和扩展机架上的 I/O 接口单元，或连接 2 个扩展机架上的 I/O 接口单元。注意，所有 I/O 连接电缆的总长度不要超过 12m。每个扩展机架也需要一个端盖。

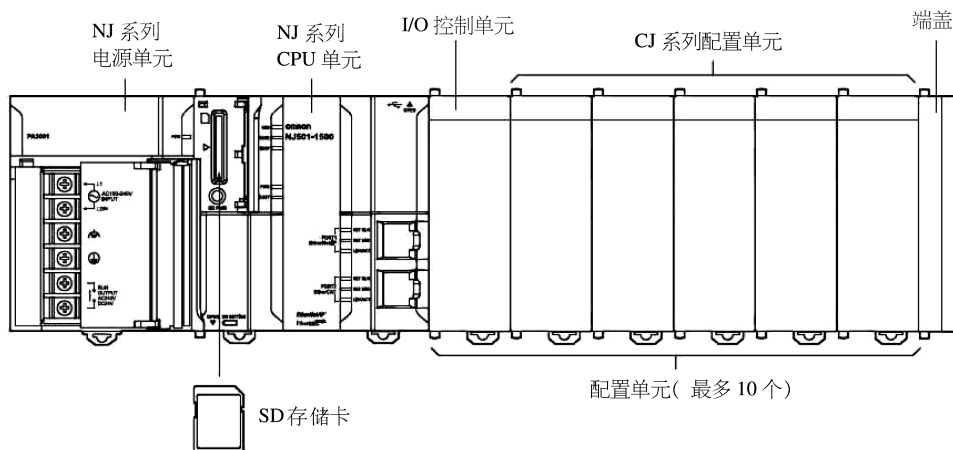


图 2-7 NJ 的 CPU 机架

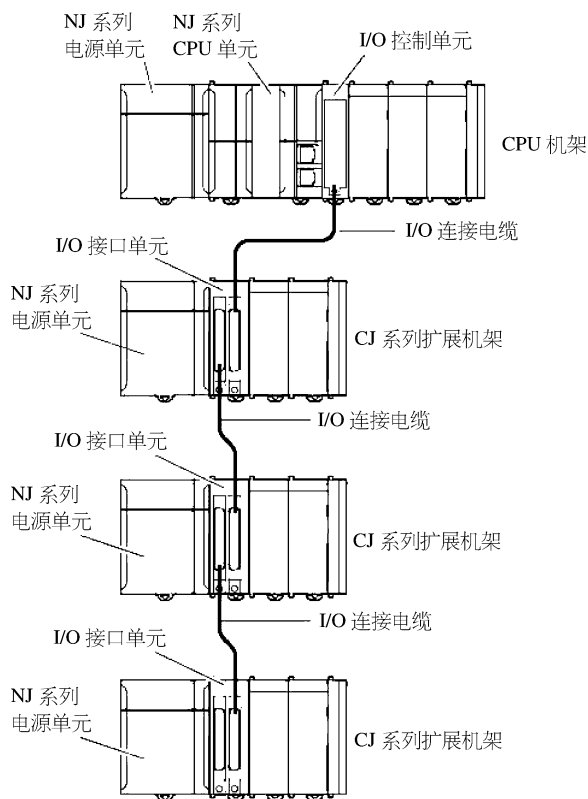


图 2-8 NJ 的扩展配置

2.2 Sysmac Studio 自动化软件的安装与使用

Sysmac Studio 自动化软件提供一个集成的开发环境，用于设置、编程、调试、维护 NJ 系列控制器和其他机器自动化控制器以及 EtherCAT 从站。

Sysmac Studio 具有以下特点。

1. 更加支持 IEC 61131-3 的编程语言

Sysmac Studio 基于国际标准 IEC 61131-3，它提供了一种基于梯形图和结构文本编程语言、基于 POU (包含程序、功能和功能块) 的先进的编程环境。

2. 便捷的操作

Sysmac Studio 在设计流程上尽可能设置少的限制，以使用户可以从系统的任何部分开始设计工作。设计理念是提供自然、灵活、易于理解、用户友好的操作，配以适当的指南，在设置和流程出现错误时可以立即纠正，或者只要它们不会导致严重问题可以留到工程完成时解决。在机器自动化控制器 (例如 NJ 系列控制器) 与 Sysmac 兼容的 EtherCAT 从站及其他 Sysmac 设备的结合方面，Sysmac Studio 可以达到最佳的功能和简便的操作。

3. 完善的调试手段

Sysmac Studio 提供了完善的调试顺序控制功能，如在线改变当前值和改变程序。也提供运动控制模拟仿真的调试功能，如以 2D 或 3D 形式显示跟踪结果和显示虚拟设备的轨迹。这些功能使得虚拟调试更接近实际设备。另外，支持对视觉传感器进行视觉系统仿真调试。使用仿真视觉版，可以对单独视觉传感器进行模拟。使用标准版，还可以对控制器进行集成仿真。

4. 强大的诊断维护功能

Sysmac Studio 允许用户在状态列表里检查控制器的状态。故障诊断功能可以让用户轻松地检查错误细节并校正控制器错误。也可以用分配控制器错误同样的方式分配用户定义错误。

2.2.1 Sysmac Studio 的规格

1. 产品型号

首次购买 Sysmac Studio 时，要同时购买一个 DVD 安装盘和单用户或多用户的使用许可。也可以单独购买 DVD，但 DVD 不包含许可。Sysmac Studio 的产品型号见表 2-7。

表 2-7 Sysmac Studio 的产品型号

产 品	许 可	媒 介	型 号
Sysmac Studio Standard Edition version 1. □□	没有 (只 DVD)	DVD* 1	SYSMAC-SE200D
	1 用户	...	SYSMAC-SE201L
	3 用户	...	SYSMAC-SE203L
	10 用户	...	SYSMAC-SE210L
	30 用户	...	SYSMAC-SE230L
	50 用户	...	SYSMAC-SE250L

(续)

产 品	许 可	媒 介	型 号
Sysmac Studio Vision Edition* 2 Version 1. □□	1 用户	...	SYSMAC-VE001L

* 1 标准版和视觉版在同一 DVD 上。

* 2 对于视觉版, 只能使用 FQ-M 系列视觉传感器的设置功能。

2. 从 Sysmac Studio DVD 安装的支持软件和封装数据

从 Sysmac Studio DVD 安装的支持软件和封装数据见表 2-8。

表 2-8 从 Sysmac Studio DVD 安装的支持软件和封装数据

可安装的软件	版 本	分 类	封装数据
Sysmac Studio	Ver. 1. □	自动化软件	• CPS • Manuals (PDF 文件)
CX-Integrator*	Ver. 2. □	支持软件	
CX-Designer*	Ver. 3. □	支持软件	
CX-Protocol*	Ver. 1. □	支持软件	
Network Configurator*	Ver. 3. □	支持软件	
CX-Server	Ver. 5. □	通信中间件	
Communications Middleware	Ver. 1. □	通信中间件	

* 该软件不能从 Sysmac Studio 视觉版安装。

3. 支持的语言

Sysmac Studio 支持以下语言: 日语、英语、德语、法语、意大利语、西班牙语、繁体中文、简体中文和韩语。

4. 适用机型

Sysmac Studio 可以使用的机型为控制器和视觉传感器。其中, 控制器型号可以选择 NJ501-1300/1400/1500、NJ301-1100/1200, 视觉传感器为 FQ-M 系列。

5. 许可的版本

许可为一系列的数字, Sysmac Studio 安装到计算机时, 安装的台数不要超过许可允许的用户数量, 否则, 以后的软件升级会遇到麻烦甚至许可作废。

(1) 标准版

该版本允许使用所有支持的产品。

(2) 视觉版

该版本仅允许使用 FQ-M 系列视觉传感器, 不能使用 NJ 系列控制器。

(3) 试用版 (无许可)

安装后可以在 30 天内使用所有标准版的功能。

可以同时注册一个以上的许可, 功能的优先级: 标准版 > 视觉版 > 试用版。

6. 适用的计算机

Sysmac Studio 在 Microsoft Windows 环境下运行, 它可以运行在以下版本的 Microsoft Windows 环境中。

- Windows XP with SP3 (64 位除外)
- Windows Vista (64 位除外)

- Windows 7 (32 位或 64 位)

应用软件 .NET Framework 3.5 必须安装, 如果尚未安装, 在 Sysmac Studio 安装时会自动安装到计算机上。

Sysmac Studio 对计算机硬件要求见表 2-9。

表 2-9 Sysmac Studio 对计算机硬件要求

	CPU	RAM	显示器	可用硬盘空间	光驱	通信端口
最低	IBM AT 或兼容机, 赛扬 540 (1.8 GHz) 处理器	2GB	XGA 1024 × 768, 1600 万色	最少 1.6GB	DVD-ROM	Ethernet 或 USB(USB 2.0)
推荐	IBM AT 或兼容机, 酷睿 i5 M520 (2.4 GHz) 或相当的处理器	2GB	WXGA 1280 × 800, 1600 万色			

当使用 3D 运动跟踪功能时, 推荐选用 NVIDIA GeForce 200 系列或更高、ATI Radeon HD5000 系列或更高的显卡, 显存最小 512 MB。

2.2.2 Sysmac Studio 的安装与卸载

Sysmac Studio 与 CX-One 可安装在同一台计算机上使用, 需注意以下几点。

1) CX-One 的版本必须为 4.0 或更高。

2) 如果 CX-One 已预先安装, 若 CX-One 的版本低于 4.0, 则先要升级至版本 4.0 以上, 再安装 Sysmac Studio, 否则 Sysmac Studio 的安装进行不下去。安装 Sysmac Studio 时, 支持软件会随之安装到 CX-One 的安装文件夹中, 由于 CX-One 已安装, 因此这些支持软件实际上覆盖了原有的自身软件。

3) 若 CX-One 的版本为 4.00 ~ 4.21 时, 必须先安装 CX-One, 后安装 Sysmac Studio。卸载时, 先卸载 Sysmac Studio, 后卸载 CX-One。

4) 若 CX-One 的版本为 4.22 或更高, CX-One 和 Sysmac Studio 的安装顺序没有限制。

1. 安装 Sysmac Studio

将 Sysmac Studio 安装盘放入 DVD-ROM 光驱, 安装程序自动启动。安装过程中每次出现的对话框操作完成后, 根据提示, 点击下方按钮, 转换到相应的其他对话框。如点击“下一步 >”按钮, 则进入下一步的安装, 等等。

1) 选择使用的语言为“简体中文”, 点击“是”按钮, Sysmac Studio 安装向导被启动。

如果计算机上没有安装 .NET Framework 3.5 SP1, 会出现要求安装的对话框, 根据提示将其安装, 重启计算机, 安装向导会自动继续后面的安装步骤。

2) 在“许可协议”对话框中, 选择“接受许可协议的条款”。

3) 在“用户信息”对话框中, 输入用户、公司及许可序列号, 如图 2-9 所示。

如果没有输入许可序列号, 则默认安装为试用版, 有 30 天的使用时间。

如果 CX-One (版本 4.0 以上) 已预先安装, 则跳到步骤 7)。

4) 在“区域信息”对话框, 输入购买 Sysmac Studio 的区域。

5) 在“选择目标位置”对话框, 选择安装的文件夹, 默认为“C:\ Program Files \ OMRON”。



图 2-9 用户信息对话框

6) 在“安装类型”对话框，选择安装的类型，如图 2-10 所示，有两种安装类型：

- ①完全：Sysmac Studio 所有的应用组件都安装。
- ②自定义：选择用户需要的应用组件安装。



图 2-10 安装类型对话框

如果选中①，则跳到步骤 8)。

如果选中②，则跳到步骤 7)。

7) 在“选择功能”对话框，如图 2-11 所示，选中“CX Common Tools”，另外两项“Sysmac Studio”和“Communications Middleware”默认为必须安装，不能更改。

8) 在“准备安装程序”对话框，点击“安装”按钮，启动“Sysmac Studio”的安装。

9) 在安装过程中，会出现 OMRON USB 驱动安装向导，点击“下一步 >”按钮，根据提示，完成对 OMRON USB 驱动的安装。



图 2-11 选择功能对话框

10) 在安装过程中，会出现安装 Communications Middleware 的 WinPCap 组件的提示，点击“是”按钮，显示 WinPCap 安装向导，点击“下一步 >”按钮，根据提示，完成对 WinPCap 的安装。

11) 在安装过程中，会出现“直接 Ethernet 连接：网卡选择”对话框，选择运行 Sysmac Studio 计算机上的网卡，此网卡与控制器通过 Ethernet 电缆连接。

12) 在安装过程中，会出现 OMRON SYSMAC PLC/NS USB 驱动安装向导，点击“下一步 >”按钮，根据提示，完成对 SYSMAC PLC 和 NS PT 的 USB 设备驱动的安装。

13) 安装最后会出现“安装向导完成”对话框，选择“是，我想马上启动我的计算机”，接着点击“完成”按钮，计算机重启，至此完成了对 Sysmac Studio 的安装。

2. 卸载 Sysmac Studio

卸载 Sysmac Studio 时，要同时卸载 Communications Middleware、CX-Server Driver Manager 和 CX-Server。由于卸载的方法基本相同，这里仅介绍卸载 Sysmac Studio 的操作步骤。

按以下步骤卸载 Sysmac Studio。

1) 从 Windows 启动菜单打开控制面板，选中“添加或删除程序”图标，出现“添加或删除程序”对话框。

2) 选中“Sysmac Studio”，点击“卸载”按钮，显示确认删除“Sysmac Studio”的对话框。

3) 点击“是”按钮，显示确认删除 CX Common Tools 的对话框，点击“是”按钮，完成卸载 CX Common Tools。Sysmac Studio 的卸载继续。

4) 卸载的最后出现“卸载完成”对话框，点击“完成”按钮，至此完成了对 Sysmac Studio 的卸载。

2.2.3 创建工程

下面介绍如何启动和退出 Sysmac Studio、创建和保存工程以及其他基本操作。

1. Sysmac Studio 的启动和退出

双击桌面上的 Sysmac Studio 图标，或从 Windows 开始菜单选择“所有程序 (P)” → “OMRON” → “Sysmac Studio”，即可启动 Sysmac Studio。

Sysmac Studio 启动后，点击标题栏右上角的“退出”按钮，或从“文件”菜单中选择“退出”，即可关闭 Sysmac Studio。

2. 创建新工程文件

1) Sysmac Studio 启动后，如图 2-12 所示，在工程视图中点击“新建工程”按钮，出现图 2-13 所示的工程属性对话框。

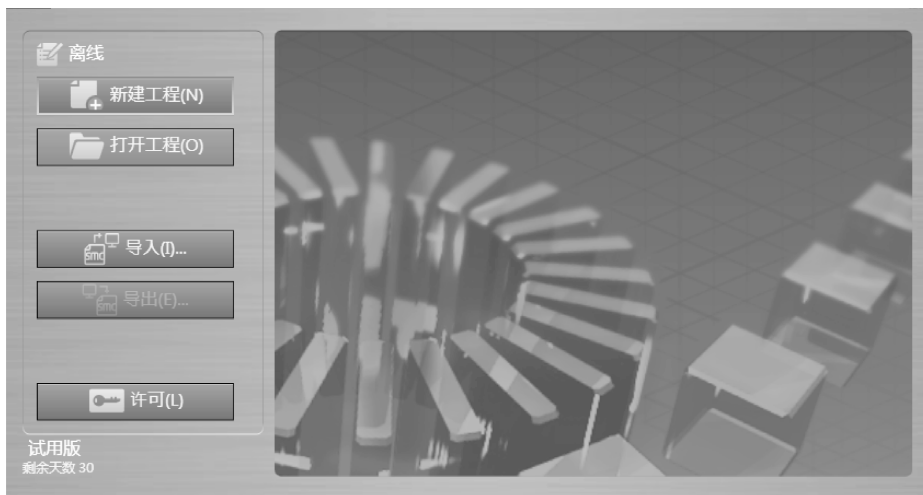


图 2-12 工程视图



图 2-13 工程属性对话框

2) 在工程属性对话框中，输入工程名称、作者、注释及选择工程类型，选择设备类型、

设备型号和设备版本号,图中,工程的名称为“New Project”,设备为“NJ501-1300”,版本号为“1.00”,点击“创建”按钮,一个新工程被创建,如图 2-14 所示,新工程下的设备为控制器,名称为“new_Controller_0”。

3) 在一个工程下面添加多个设备。右击设备图标,从菜单中选择“添加设备”,如图 2-15 所示;或直接从“插入”菜单中选择“设备”,显示出“添加设备”对话框,选择设备并点击“是”按钮,设备便添加到该工程中。

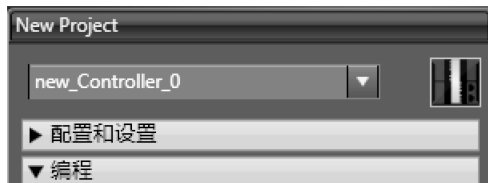


图 2-14 新工程



图 2-15 添加设备

除了添加设备外,还可以对工程下的设备进行变更、重命名和删除操作。

为设备选择控制器时,要设置其版本。可为正在使用的控制器或先前使用的单元版本,设置单元版本。控制器的单元版本不同,支持的功能也有差别。在控制器的单元版本所支持的范围内,可以编程和设置。如果试图使用所设置单元版本不支持的功能,则无法使用或出现错误。

例如,如果控制器单元版本设置为 1.01,可以在版本 1.01 支持的范围内使用控制器。这种情况下,可以将创建的用户程序传递到单元版本 1.02 的控制器,但不能将用户程序传递到单元版本 1.00 的控制器。

添加控制器后,可以改变其版本。改变单元版本后,新单元版本不支持的功能设置会失掉。

在工程文件创建后,可以对其属性做修改,也就是重新编辑工程文件的属性。方法如下:在工程窗口选中工程文件,点击出现在选中的工程文件左侧的“编辑属性”按钮,出现“编辑属性”对话框,编辑完成后,点击“保存”按钮。

3. 关闭工程文件并返回工程视图/保存工程文件/保存工程文件为另一名称

关闭工程文件并返回工程视图、保存工程文件和保存工程文件为另一名称,以上三种操作都是在“文件”菜单中完成的,下面仅介绍关闭工程文件并返回工程视图的操作。

1) 从“文件”菜单中选择“关闭”,如图 2-16 所示。

出现询问是否需要保存工程的对话框,如图 2-17 所示。

2) 在图 2-17 中,若点击“是”按钮,则当前工程文件保存为先前登记的名称,并返回到工程视图。如点击“否”按钮,则当前工程文件不保存,并返回到工程视图。

4. 打开工程文件

对于已存在的工程文件,点击工程视图中的“打开工程”按钮,会显示出工程对话框,从工程名称的列表选择一个工程,点击“打开”按钮。

工程对话框中显示一个工程的信息有：工程名称、作者、创建的日期和时间、最后修改的时间、注释。因此，两个工程即使同名，但其他项目不同。例如，创建的日期和时间不一样，也是不同的两个工程。



图 2-16 “文件”菜单



图 2-17 保存对话框

5. 导出工程文件

可以使用下列两种方法导出工程文件。

- 1) 从工程视图导出，点击“导出”按钮。
- 2) 从“文件”菜单导出，选中菜单中的“导出”。

6. 导入工程文件

可以使用下列两种方法导入工程文件。

- 1) 从工程视图导入，点击“导入”按钮。
- 2) 从“文件”菜单导入，选中菜单中的“导入”。

7. 工程文件的密码保护

可以为工程文件设置密码保护，一旦设置，下列操作都需要输入密码。

- 1) 打开工程文件。
- 2) 改变工程文件的属性。
- 3) 导入工程文件的内容。

按以下步骤设置和清除工程文件的密码保护。

1) 在工程视图选择工程并点击“编辑属性”按钮，显示出工程属性对话框，如图 2-18 所示。

2) 在工程属性对话框，选择“应用工程文件的密码保护”，输入密码，点击“保存”按钮。

清除密码保护时，去掉“应用工程文件的密码保护”，点击“保存”按钮。

8. 工程历史文件的管理

通过为工程分配顺序号，可以管理不同时刻的工程历史文件。如图 2-19 所示，工程顺序号按层次在工程视图中显示。可以在不同顺序号工程之间离线比较。



图 2-18 设置密码保护

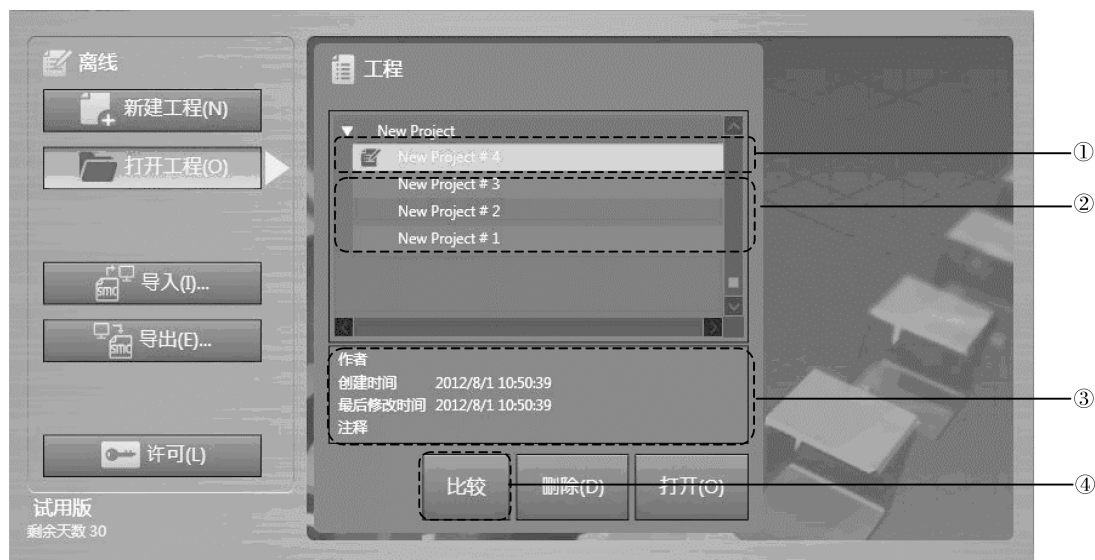


图 2-19 历史工程文件

①最新工程 ②不同序列号的工程 ③属性 ④比较

按下列步骤用新的工程顺序号存储工程。

- 1) 如图 2-20 所示，从“文件”菜单选择“另存为新号码...”，显示工程属性对话框。
- 2) 在工程属性对话框编辑属性，然后点击“存储”按钮，当前工程以最新的顺序号存储。顺序号自动创建，由小到大增加，用户不能改变。存储的历史工程根据顺序号分层次显示，最新的（也是顺序号最大的）在顶部，最早的（也是顺序号最小的）在底部，如图 2-19 所示。对于每一个工程顺序号，都要设置工程属性。

打开一个指定顺序号的工程时，从工程分层中选择具有期望顺序号的工程，点击“打开”按钮。需要注意的是，打开一个指定顺序号的工程后，不能对其进行覆盖操作。工程变化后，以不同的名称保存或以最新的顺序号保存。

9. 显示和注册许可

使用下列步骤显示和注册许可序列号。

1) 在工程视窗点击“许可”按钮，显示出当前注册的许可产品。在产品名称栏点击“许可”，在产品详细信息栏显示详细信息。

试用版剩余的天数显示在“许可”按钮下方。

2) 注册一个新的许可，点击“注册许可”按钮，显示出注册许可对话框，如图 2-21 所示。



图 2-20 “文件”菜单选择

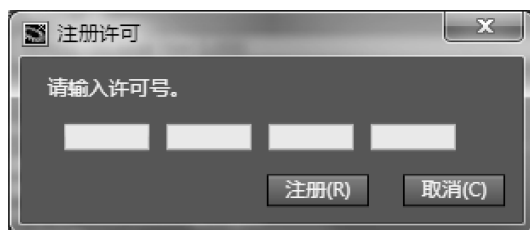


图 2-21 注册许可对话框

3) 输入许可序列号，点击“注册”按钮。

如果许可正常登记，会显示出重启软件的对话框，点击“是”按钮，重启 Sysmac Studio，完成注册。

2.2.4 操作视窗

图 2-22 所示为 Sysmac Studio 的操作视窗。

下面介绍 Sysmac Studio 操作视窗中各个视图的作用。

1. 多视图浏览器

图 2-23 所示为多视图浏览器。

1) 通过这个面板可以访问所有 Sysmac Studio 数据。视图分为配置和设置、编程两部分。每一部分以树状结构显示。

2) 点击树中每个项目前面的图标▶或▼，可以展开或收起树状结构。

3) 如果右击可以设置的任何数据项目，会弹出一个菜单。

4) 点击窗口左边的隐藏/显示条，可以隐藏或显示多视图浏览器。即使面板已隐藏，把鼠标放到隐藏/显示条后，面板会暂时显示出来。当把鼠标从隐藏/显示条移走后，面板又隐藏起来。

5) 对多视图浏览器中的目录和成员，可以用 5 种颜色标记区分。

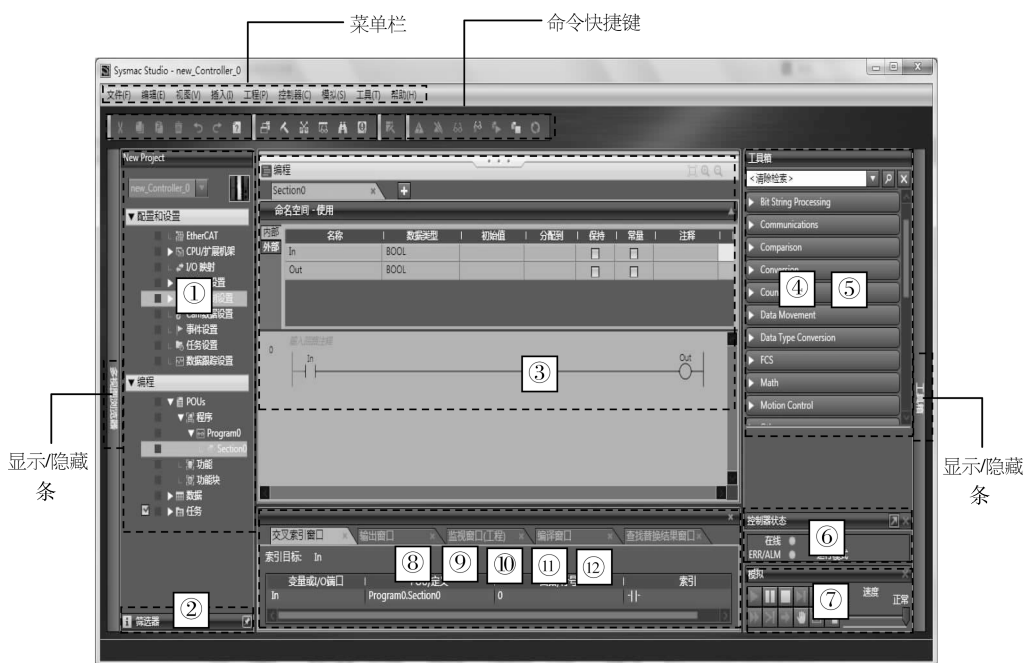


图 2-22 Sysmac Studio 的操作视窗

- ①多视图浏览器 ②筛选器面板 ③编辑面板 ④工具箱 ⑤查找和替换面板 ⑥状态监视面板 ⑦模拟面板
⑧交叉索引窗口 ⑨输出窗口 ⑩监视窗口 ⑪编译窗口 ⑫查找和替换结果窗口

6) 多视图浏览器中出现红色错误图标时, 表示编辑的数据有错误。点击错误图标, 会在编辑面板中显示有错误的项目。如果显示错误图标, 数据不能下载到 NJ 系列 CPU 单元中。

2. 筛选器面板

图 2-24 所示为筛选器面板。

筛选器面板用于搜索颜色编码和错误图标的项目, 结果列表显示。点击筛选器面板栏显示筛选器面板。如果鼠标不放在上面的时间超过 5s, 筛选器面板自动隐藏。

点击筛选器面板右上角的针状图标 , 消除筛选器面板自动隐藏功能。

3. 编辑面板

编辑面板用来显示和编辑各个项目的数据。配置和设置以及编程都是在编辑面板上完成的。双击多视图浏览器中的项目, 可切换编辑面板和显示选中项目的细节。图 2-25 所示的编辑面板为梯形图编程界面。

4. 工具箱

图 2-26 所示为工具箱。工具箱展示在编辑面板中编辑数据时用到的对象。通过点击窗口右侧的隐藏/显示



图 2-23 多视图浏览器

条，可以隐藏或显示工具箱。即使隐藏了工具箱，把鼠标放在隐藏/显示条上，可以暂时显示工具箱。从隐藏/显示条上移走鼠标，工具箱会隐藏起来。

5. 查找和替换面板

图 2-27 所示为查找和替换面板，可以查找和替换编程中的字符串。

6. 状态监视面板

图 2-28 所示为状态监视面板，显示控制器的当前运行状态。只有当 Sysmac Studio 与控制器在线连接时，状态监视面板才显示。当 Sysmac Studio 离线时，状态监视面板不显示。



图 2-24 筛选器面板

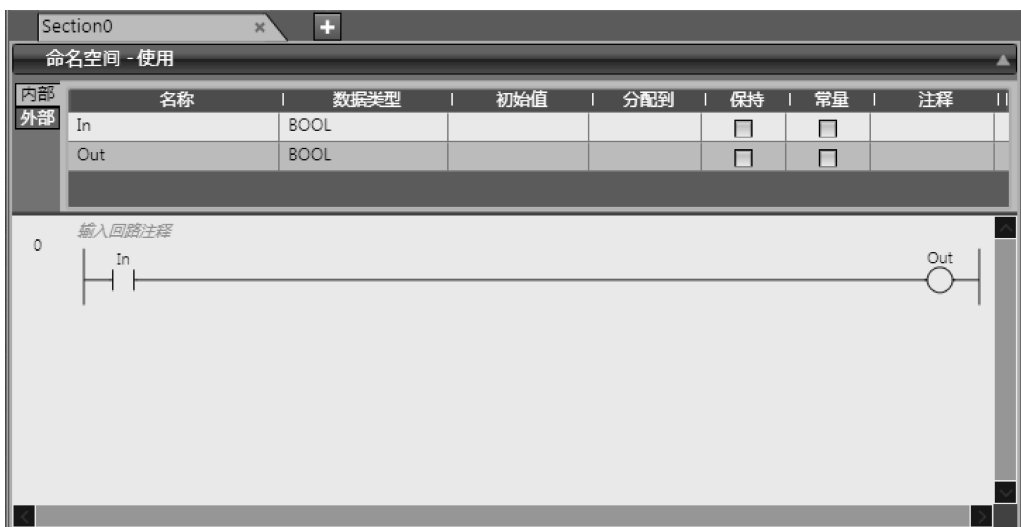


图 2-25 编辑面板

7. 模拟面板

图 2-29 所示为模拟面板，用来设置、启动和停止对控制器的模拟仿真。当 Sysmac Studio 与控制器在线连接时，模拟面板不显示。

8. 交叉索引窗口

图 2-30 所示为交叉索引窗口，显示变量、数据类型、I/O 端口、功能和功能块在 Sysmac Studio 中使用位置的列表。在梯形图编辑器或变量表选择变量或其他项目，显示这些项目使用位置的列表。点击列表中的任何一项，跳转到使用这个项目的位置。

9. 输出窗口

图 2-31 所示为输出窗口，展示编译结果。

10. 监视窗口

图 2-32 所示为监视窗口，展示仿真器或在线控制器的监视结果。通过直接输入变量、从变量表或程序回路中拖放变量，可以监视变量的当前值。



图 2-26 工具箱



图 2-27 查找和替换面板



图 2-28 状态监视面板



图 2-29 模拟面板

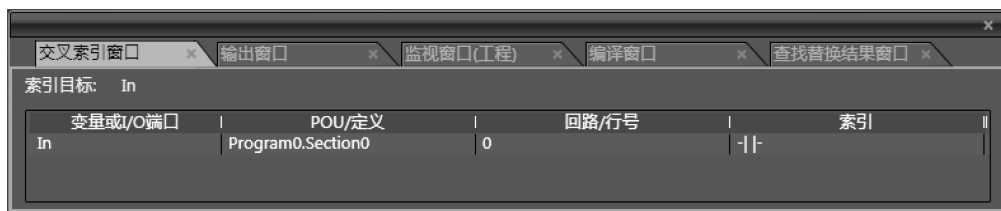


图 2-30 交叉索引窗口

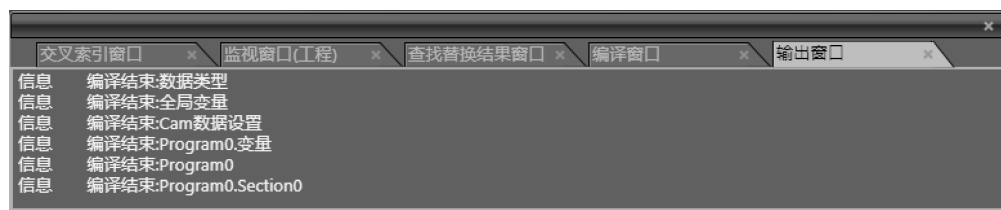


图 2-31 输出窗口



图 2-32 监视窗口

11. 编译窗口

图 2-33 所示为编译窗口，展示程序检查和编译的结果。双击结果中的任何一项，在编辑面板中显示程序检查错误、程序检查警告或编译错误的位置，这样可以进行编辑。



图 2-33 编译窗口

12. 查找替换结果窗口

图 2-34 所示为查找替换结果窗口，展示执行“全部搜索”或“全部替换”时的结果。双击搜索/替换结果中的项，显示在编辑面板中的位置，这样可以进行编辑。

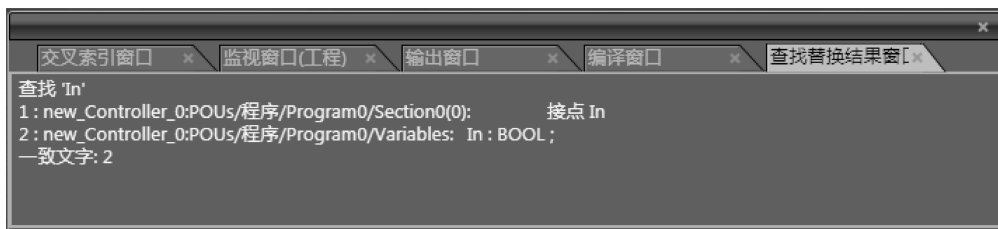


图 2-34 查找替换结果窗口

2.2.5 命令菜单

Sysmac Studio 菜单命令结构见表 2-10。

表 2-10 Sysmac Studio 菜单命令结构

菜 单	子菜单/命令	
文件	关闭	...
	保存	
	另存为	
	另存为新号码	
	导入	

(续)

菜 单	子菜单/命令	
文件	导出	...
	离线比较	
	页面设置	
	打印	
	退出	
编辑	撤销	...
	重做	
	剪切	
	复制	
	粘贴	
	删除	
	全选	
	查找和替换	
视图	跳转	...
	输出窗口	
	监视窗口	
	交叉索引窗口	
	编译窗口	
	查找和替换结果窗口	
	模拟窗口	
	缩放	放大
		缩小
		缩放到合适大小
		重置缩放
插入	回路	线
		常开接点
		常闭接点
		常开或接点
		常闭或接点
		线圈
		非线圈
		功能
		功能块
	控制器	NJ301
		NJ501
	视觉传感器	FQ-M
	程序	梯形图
		ST
		段
	功能	梯形图
		ST
	功能块	梯形图
		ST
	调试程序	梯形图
		ST
		段

(续)

菜 单	子菜单/命令	
插入	轴设置	...
	轴组设置	
	CAM 数据设置	
	数据跟踪	
工程	检查所有程序	...
	检查选择的程序	
	编译控制器	
	重编译控制器	
	中止编译	
	内存使用	
	在线编辑	开始
		传送
		取消
	库	显示参考
		库设置
		创建库
控制器	通信设置	...
	变更设备	
	在线	
	离线	
	同步	...
	模式	运行模式
		编程模式
	监视	...
	停止监视	
	设置/重置	设置
		重置
	强制刷新	TRUE
		FALSE
		取消
		全部取消
	MC 试运行	开始
		停止
	MC 监测表	...
	SD 存储卡	
	控制器时钟	
	释放访问权限	
	更新 CPU 单元名称	
	安全性	设置操作权限
		用户程序执行用 ID
		CPU 单元写保护
		设置/解除数据保护
		临时更改禁止的数据保护
		完成临时更改禁止的数据保护

(续)

菜 单	子菜单/命令	
控制器	清除所有内存	...
	重置控制器	
模拟	运行	...
	暂停	
	停止	
	单步执行	
	单步进入	
	单步跳出	
	连续单步执行	
	执行单次扫描	
	跳转到当前位置	
	断点窗口	
	设置/删除断点	
	取消全部断点	
	校准	
	执行预测模式	
	传送所有变量当前值	
	启动 NS 统合模拟	
工具	故障分析	...
	备份变量和内存	备份
		还原
	导出全局变量	NetworkConfigurator
		CX-Designer
	选项	...
帮助	帮助内容	...
	命令索引	
	系统定义变量索引	
	键盘映射索引	
	在线注册	
	关于 Sysmac Studio	

2.2.6 Sysmac Studio 与控制器的连接

Sysmac Studio 与控制器硬件连接有 4 种方式。

1. 通过 USB 端口的直接连接

计算机上的 USB 端口直接连接到控制器上的外设 USB 端口，如图 2-35 所示，这是最常用的一种连接方式。

2. 通过 Ethernet 端口的直接连接

计算机上的 Ethernet 端口直接连接到控制器上的内置 EtherNet/IP 端口，如图 2-36 所示。



图 2-35 USB-直接连接



图 2-36 Ethernet-直接连接

3. 通过 USB 端口的远程连接

计算机上的 USB 端口直接连接到控制器上的外设 USB 端口，通过 Ethernet 网，建立起与另一台控制器的连接，如图 2-37 所示。

4. 通过 hub 的 Ethernet 连接

计算机上的 Ethernet 端口和控制器上内置的 EtherNet/IP 端口都连接到 hub 上，通过 Ethernet 网，计算机与控制器建立连接，如图 2-38 所示。

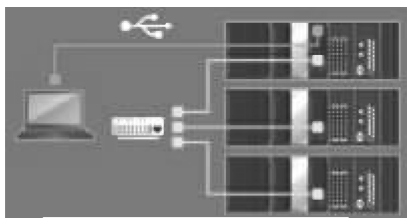


图 2-37 USB-远程连接

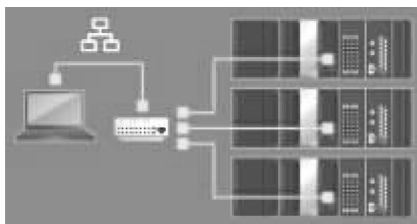


图 2-38 Ethernet-hub 连接

从“控制器”菜单，选择“通信设置”，出现通信设置对话框。设置通信方法、远程 IP 地址。对话框中设有 USB 通信测试按钮和 Ethernet 通信测试按钮，点击后检测 Sysmac Studio 与远程控制器的通信是否建立起来。还可以对选项“在线时确认序列 ID”、“离线时检查强制刷新”做出选择，以及设置响应检测时间。

2.3 NJ 的配置和设置

使用 NJ 前，需要对 NJ 进行配置和设置，完成后随工程下载到 NJ 中。NJ 配置和设置的主要内容有：

- 1) 如果使用 EtherCAT，则创建 EtherCAT 从站配置，进行 EtherCAT 主站设置。
- 2) 如果使用 CJ 系列单元，则创建 CPU/扩展机架配置，进行特殊单元设置。
- 3) 为 I/O 端口分配设备变量。
- 4) 如果使用运动控制，则创建轴并分配它们给伺服驱动器和编码器输入从站，进行运动控制参数设置，如轴参数、轴组参数和凸轮数据。
- 5) 如果使用 EtherNet/IP，则进行内置 EtherNet/IP 端口设置。
- 6) 控制器基本设置，如启动模式、写保护、系统服务监视时间等。
- 7) 任务设置，如设置任务的执行周期、为任务分配程序、任务中变量的独占控制设置等。

对 NJ 进行配置和设置的操作在 Sysmac Studio 多视图浏览器中完成。

操作时，选中多视图浏览器中的项目，双击或右击，则弹出相应操作界面，根据提示进行下一步操作。其中，右击时，弹出一个菜单，显示的各个命令项表示能够进行的操作，用户从中选择，使用起来很方便。除此之外，还可以通过 Sysmac Studio 的菜单栏完成一些操作，选中项目后，到菜单栏单击相应的选项，在弹出的下拉子菜单中选择各种命令。

2.3.1 EtherCAT 的配置和设置

1. EtherCAT 配置

EtherCAT 配置如图 2-39 所示，用 Sysmac Studio 中的 EtherCAT 编辑器对从站进行配置，完成后下载到 NJ 主站中，NJ 的 CPU 单元将此当作“正确”的 EtherCAT 配置。

配置时，自动为从站登记 I/O 端口。用户为 I/O 端口分配设备变量后，就可以在程序中通过指定设备变量去访问从站。

EtherCAT 从站配置有两种方法。

方法 1：离线创建从站配置

(1) 打开 EtherCAT 编辑器标签页

在多视图浏览器中，双击“配置和设置”下的“EtherCAT”；或右击“配置和设置”下的“EtherCAT”，从菜单中选择“编辑”，EtherCAT 编辑器标签页显示出来。

(2) 登记从站

1) 将从站从工具箱中拖放到拓扑显示中的主站，从站便添加到主站之下。

2) 同步骤 1) 一样，将从站从工具箱中拖放到拓扑显示中已连接的从站，从站便添加到先前从站之下。

图 2-40 所示为使用 EtherCAT 编辑器建立的 EtherCAT 从站配置，共有 3 个从站，第一个为远程 I/O 从站 GX-MD1612，第二、三个都是 G5 系列伺服 R88D-KN01H-ECT。

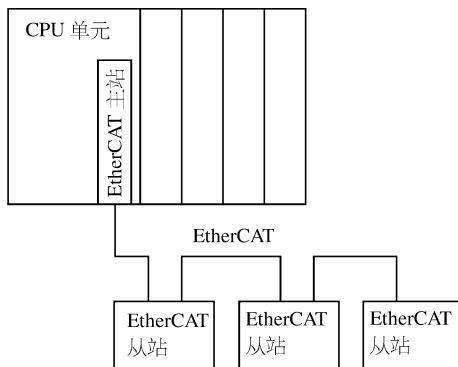


图 2-39 EtherCAT 配置



图 2-40 EtherCAT 从站配置

拓扑显示中的从站可以被删除、复制和粘贴。

方法 2：在线读取实际从站的配置

如果实际的 EtherCAT 网络已搭建，将 Sysmac Studio 与其在线连接，读取从站配置。

注意，当 EtherCAT 从站登记为伺服驱动器或编码器输入从站后，用户在创建轴时，轴变量会自动生成。

2. EtherCAT 主站设置

双击“配置和设置”下的“EtherCAT”，然后选中主站，显示出 EtherCAT 主站设置标签页，如图 2-41 所示。



图 2-41 EtherCAT 主站设置

EtherCAT 主站设置中包括以下各项。

- (1) 设备名称
- (2) 型号
- (3) 产品名称
- (4) 从站数目
- (5) 电缆总长度
- (6) PDO 通信周期
- (7) 故障弱化操作设置
- (8) 从站启动等待时间
- (9) PDO 通信超时检测次数
- (10) 版本检测方法
- (11) 串口号检测方法

EtherCAT 主站设置的详细内容参见 6.3 节。

3. EtherCAT 从站设置

双击“配置和设置”下的“EtherCAT”，然后选中从站，显示出 EtherCAT 从站设置标签页，如图 2-42 所示。

EtherCAT 从站设置中包括以下各项。



图 2-42 EtherCAT 从站设置

- (1) 设备名称
- (2) 机型
- (3) 产品名称
- (4) 版本
- (5) 节点地址
- (6) 有效/无效设置
- (7) 串口号
- (8) PDO 映射设置
- (9) 分布式时钟有效
- (10) 参考时钟
- (11) 初始化参数设置
- (12) 备份参数设置

EtherCAT 从站设置的详细内容参见 6.3 节。

2.3.2 CPU/扩展机架的配置和设置

1. CPU/扩展机架配置

图 2-43 所示为 NJ 的机架配置。使用 Sysmac Studio “CPU/扩展机架配置” 单元编辑器创建单元配置，并下载到 CPU 单元中，CPU 单元将此当作 “正确” 的单元配置。

当电源 ON 时, CPU 单元会自动检查并确认“正确”单元配置是否与实际的单元配置相符。单元配置时会自动登记指定单元的 I/O 端口, 用户为 I/O 端口分配设备变量后, 程序中通过设备变量就可以访问这些配置的单元。

NJ 控制器装配了实际单元后, 即使 Sysmac Studio 没有为控制器创建单元配置并传送, 控制器仍然可以启动。但是这种情况下, I/O 端口和设备变量不会自动创建。这样, 不能从程序中访问这些单元。

(1) 创建单元配置

有两种创建单元配置的方法。

方法 1: 离线创建单元配置并传送

如图 2-44 所示, 在 Sysmac Studio 的单元编辑器中, 创建所谓“正确”的单元配置。与用户程序一起, 下载到 CPU 单元中。

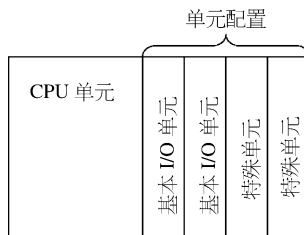


图 2-43 机架配置

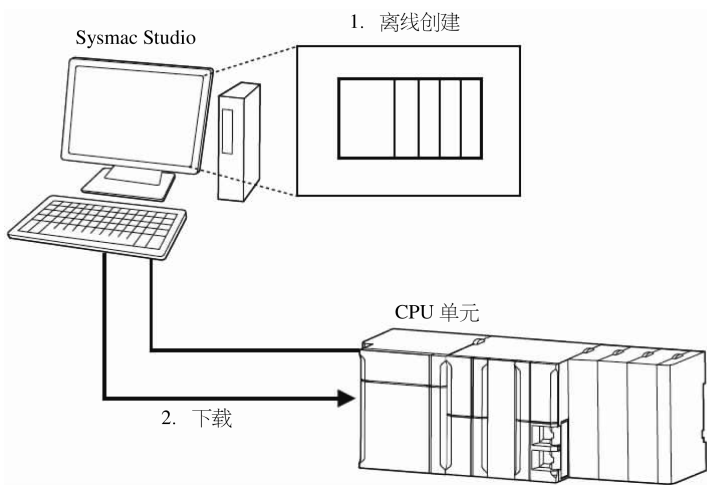


图 2-44 离线创建单元配置并下载

在多视图浏览器中, 双击“配置和设置”下的“CPU/扩展机架”; 或右击“配置和设置”下的“CPU/扩展机架”, 从菜单中选择“编辑”, 单元编辑器显示出来。

1) 登记单元

①改变电源单元型号: 右击电源单元, 从菜单中选择“改变型号”, 改变型号对话框会显示出来; 选择电源单元, 然后点击“是”按钮。

②插入单元: 从型号选择窗口里选中单元, 拖放到单元编辑器中, 单元即被插入。

③改变单元型号: 右击单元, 从菜单中选择“改变型号”, 改变型号对话框会显示出来; 选择单元, 然后点击“是”按钮, 单元即改变为选中的型号。

④删除单元: 右击单元, 从菜单中选择“删除”, 单元即被删除。

⑤复制和粘贴单元: 右击要复制的单元, 从菜单中选择“复制”; 右击要插入单元的位置, 从菜单中选择“粘贴”, 单元即被粘贴。

图 2-45 所示为创建的 CPU 机架配置。

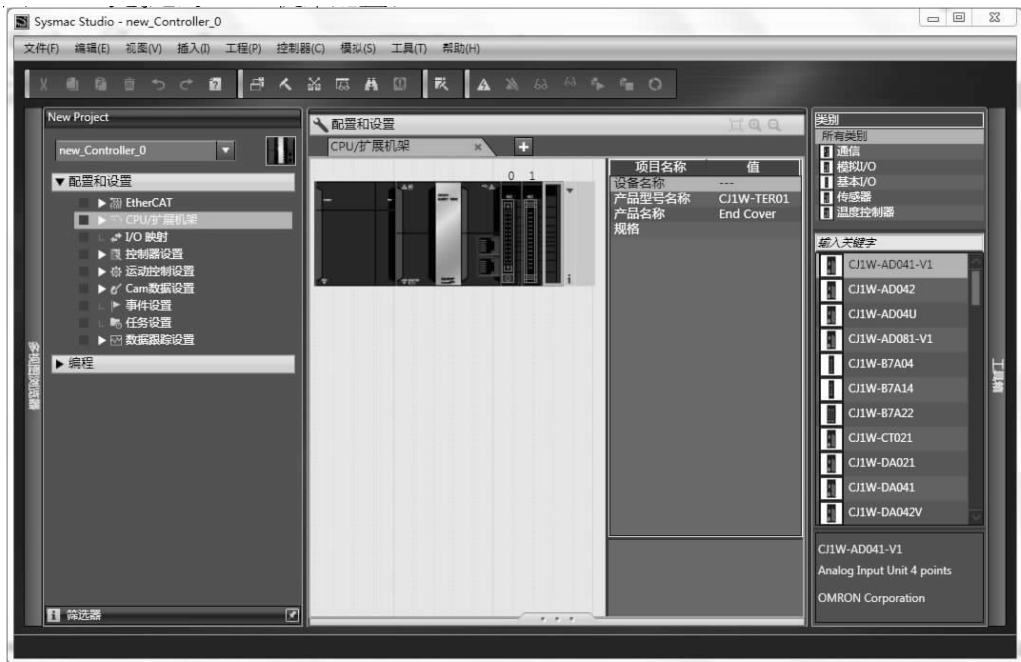


图 2-45 CPU 机架配置

2) 创建扩展机架

①添加扩展机架：在没有单元的任何位置，右击，从菜单中选择“添加机架”，机架即被添加。

②删除扩展机架：选中要删除机架的一个单元；在没有单元的任何位置，右击，从菜单中选择“删除机架”，机架即被删除。

③删除所有机架和单元：在没有单元的任何位置，右击，从菜单中选择“全部清除”，所有的机架和单元即被删除。

方法 2：从实际安装的系统在线读取单元配置并下载

如图 2-46 所示，这种方法将当前的实际单元配置作为“正确”配置。Sysmac Studio 在线连接实际单元，读取单元配置。随后创建用户程序，然后，将单元配置和用户程序一同下载。

1) 在多视图浏览器中，双击“配置和设置”下的“CPU/扩展机架”；或右击“配置和设置”下的“CPU/扩展机架”，从菜单中选择“编辑”，单元编辑器显示出来。

2) 在单元编辑器没有单元的任何位置，右击，从菜单中选择“与实际单元配置比较和合并”，如图 2-47 所示。

读出实际单元配置并与 Sysmac Studio 中的单元配置比较，结果显示在“与实际单元配置比较和合并”窗口，如图 2-48 所示。

3) 点击“实际单元配置应用”按钮，与实际配置同步，如图 2-49 所示。

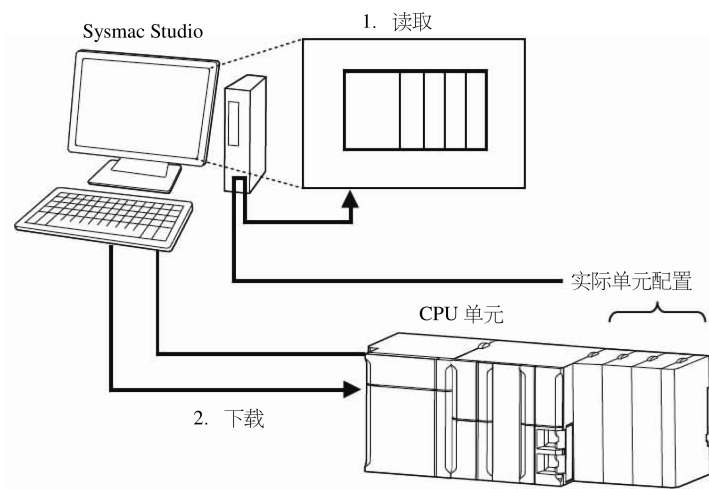


图 2-46 在线读取实际的单元配置并下载



图 2-47 与实际单元配置比较和合并



图 2-48 单元配置比较结果

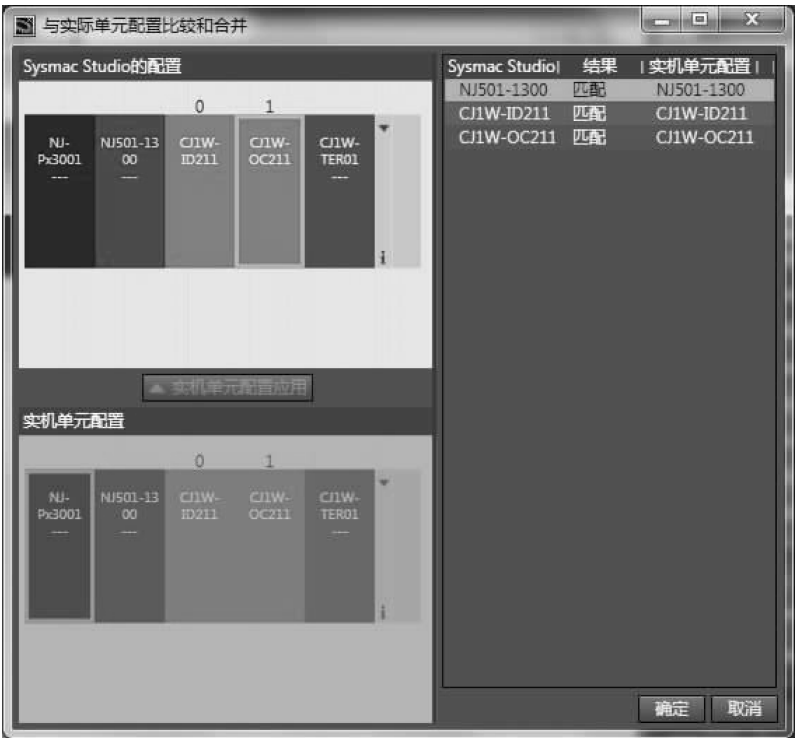


图 2-49 单元配置合并

4) 点击“确定”按钮。

(2) 校验单元配置

用 Sysmac Studio 完成单元配置比较。

1) 在 Sysmac Studio (计算机) 的单元配置和实际的单元配置之间比较。

如图 2-50 所示, 首次从 Sysmac Studio 下载单元配置到 CPU 单元时, 可以将 Sysmac Studio 中的系统配置与实际的单元配置相比较, 以确认是否一致。

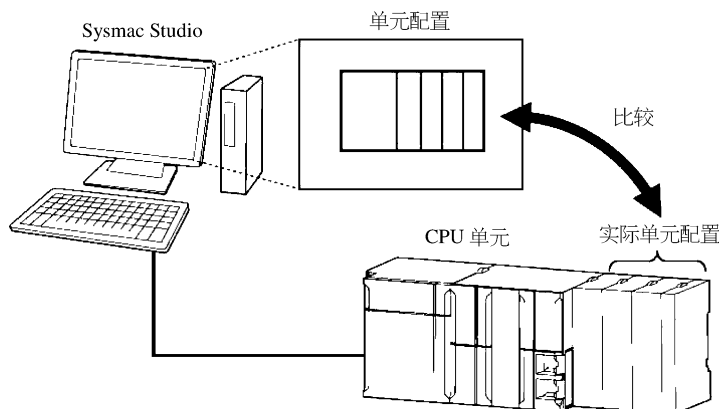


图 2-50 单元配置比较 1

2) 在 Sysmac Studio (计算机) 的单元配置和实际 CPU 单元中的单元配置之间比较。

如图 2-51 所示, 从 Sysmac Studio 下载单元配置至 CPU 单元时, 可以将 Sysmac Studio 中的系统配置与存储在 CPU 单元中的单元配置信息相比较, 以确认是否一致。

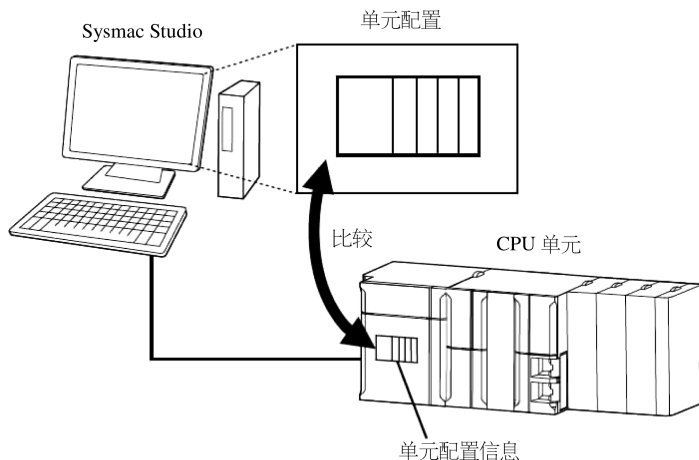


图 2-51 单元配置比较 2

2. 单元设置

这里的单元是指 CPU 机架和扩展机架上的基本 I/O 单元、高功能单元, 这里的高功能单元是指 CPU 总线单元和特殊 I/O 单元。在单元编辑器中, 为配置的单元做设置, 包括:

设备名称、输入响应时间和单元号。设备名称用于构造访问配置单元的设备变量，输入响应时间是为每个机架插槽中的基本 I/O 单元设置的，单元号是为安装的 CPU 总线单元和特殊 I/O 单元设置的。

设备名称是在单元编辑器添加单元时自动生成的，默认名称为“J”加上从“01”开始的序列数。为方便使用，用户可以修改设备名称。

(1) 基本 I/O 单元设置

基本 I/O 单元可设置参数为输入响应时间，或称为输入滤波器时间常数，根据需要设置，可减少振动和外部杂波干扰造成的不可靠性。在 0ms/0.5ms/1ms/2ms/4ms/8ms/16ms/32ms 范围内设置，默认为 8ms。

(2) 高性能单元设置

高性能单元要设置单元号，应该使其与单元面板上旋转开关设置的单元号相一致。如果不一致，以面板上旋转开关设置的单元号为准。

高性能单元的设置有以下四种方法。

方法 1：在 Sysmac Studio 的单元设置界面中设置

- 1) 在单元配置与设置中选中单元。
- 2) 在单元设置标签页进行设置，图 2-52 所示为 CJ1W-AD041-V1 特殊单元设置界面。



图 2-52 A-D 特殊单元设置

- 3) 在线连接 CPU 单元并将设置传送到 CPU 单元。

方法 2：使用 Sysmac Studio 在 I/O 映射中为 I/O 端口设定初始值

1) 使用 Sysmac Studio 的 I/O 映射，为 I/O 端口设定初始值。

2) 重启单元，复位控制器，或控制器断电再上电。

方法 3：使用 Sysmac Studio 为 CJ 系列单元的设备变量设定初始值

1) 使用 Sysmac Studio 为 CJ 系列单元的设备变量设定初始值。

2) 从 Sysmac Studio 下载变量表到 CPU 单元，下载时，在复选框中选择“初始化具有保持属性的变量的当前值”。

3) 重启单元，复位控制器，或控制器断电再上电。

方法 4：使用指令为 CJ 系列单元设备变量赋值

如图 2-53 所示，在操作开始时，从用户程序中使用指令，例如 MOVE 指令，为 CJ 系列单元的设备变量赋值，然后重启单元。

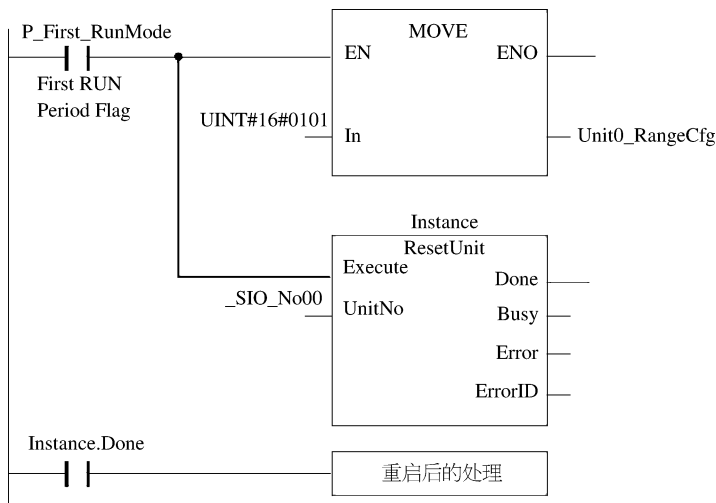


图 2-53 指令改变设置

当特殊单元改变设置后，重启前确认与设备相连目标的安全。

2.3.3 I/O 映射

1. I/O 端口

如图 2-54 所示，I/O 端口是一个逻辑接口，CPU 单元使用它与外部设备（从站和单元）交换数据。在 Sysmac Studio 创建从站和单元配置时，I/O 端口自动创建。为 I/O 端口分配设备变量后，用户程序通过设备变量访问从站和单元。

在 Sysmac Studio 中创建 EtherCAT 从站或单元配置时，或从 Sysmac Studio 读取控制器实际的 EtherCAT 从站或单元配置时，I/O 端口自动登记在 I/O 映射中。如图 2-55 所示，可以检查登记在 I/O 映射中的 I/O 端口。

2. I/O 端口名称

(1) EtherCAT 从站

1) 远程 I/O 终端的 I/O 端口名称。例如，
一个 16 点远程输入终端：In_Bit00 ~ In_Bit15，

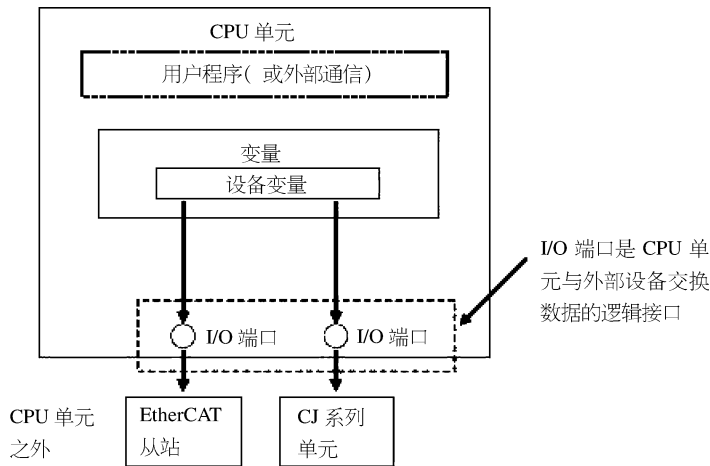


图 2-54 I/O 端口

位置	端口	说明	R/W	数据类型	变量	变量注释	变量类型
CPU 扩展机架							
CPU 扩展机架 0							
[00]	CJ1W-ID211 (DC Input Unit)						
	Ch1_In	Input CH1	R	WORD	J01_Ch1_In		全局变量
	Ch1_In00	Input CH1 bit 00	R	BOOL	J01_Ch1_In00		全局变量
	Ch1_In01	Input CH1 bit 01	R	BOOL	J01_Ch1_In01		全局变量
	Ch1_In02	Input CH1 bit 02	R	BOOL	J01_Ch1_In02		全局变量
	Ch1_In03	Input CH1 bit 03	R	BOOL	J01_Ch1_In03		全局变量
	Ch1_In04	Input CH1 bit 04	R	BOOL	J01_Ch1_In04		全局变量
	Ch1_In05	Input CH1 bit 05	R	BOOL	J01_Ch1_In05		全局变量
	Ch1_In06	Input CH1 bit 06	R	BOOL	J01_Ch1_In06		全局变量
	Ch1_In07	Input CH1 bit 07	R	BOOL	J01_Ch1_In07		全局变量
	Ch1_In08	Input CH1 bit 08	R	BOOL	J01_Ch1_In08		全局变量
	Ch1_In09	Input CH1 bit 09	R	BOOL	J01_Ch1_In09		全局变量
	Ch1_In10	Input CH1 bit 10	R	BOOL	J01_Ch1_In10		全局变量
	Ch1_In11	Input CH1 bit 11	R	BOOL	J01_Ch1_In11		全局变量
	Ch1_In12	Input CH1 bit 12	R	BOOL	J01_Ch1_In12		全局变量
	Ch1_In13	Input CH1 bit 13	R	BOOL	J01_Ch1_In13		全局变量
	Ch1_In14	Input CH1 bit 14	R	BOOL	J01_Ch1_In14		全局变量
	Ch1_In15	Input CH1 bit 15	R	BOOL	J01_Ch1_In15		全局变量
[01]	CJ1W-OC211 (Relay Output Unit)						
	Ch1_Out	Output CH1	RW	WORD			
	Ch1_Out00	Output CH1 bit 00	RW	BOOL			
	Ch1_Out01	Output CH1 bit 01	RW	BOOL			

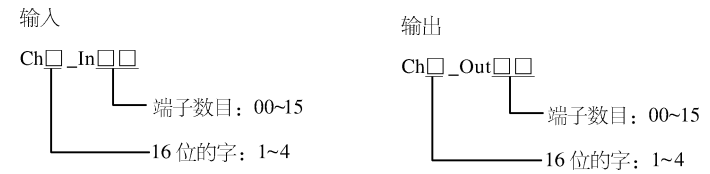
图 2-55 I/O 映射

一个 16 点远程输出终端：Out_Bit00 ~ Out_Bit15。

2) 对其他从站，使用在 EtherCAT 对象字典中定义的对象名称的全部或部分。例如，模拟量输入单元：CH1_Input_16bit，R88D-KN50H-ECT：Position_actual_value，Digital_inputs。

(2) CJ 系列基本 I/O 单元

根据以下规则创建 I/O 端口名称。



CJ 系列基本 I/O 单元的 I/O 端口名称见表 2-11。

表 2-11 CJ 系列基本 I/O 单元的 I/O 端口名称

输入点数/ 输出点数	I/O 端口名称			
	输入	数据类型	输出	数据类型
8 点	Ch1_In	WORD	Ch1_Out	WORD
	Ch1_In00 ~ Ch1_In08	BOOL	Ch1_Out00 ~ Ch1_Out08	BOOL
16 点	Ch1_In	WORD	Ch1_Out	WORD
	Ch1_In00 ~ Ch1_In15	BOOL	Ch1_Out00 ~ Ch1_Out15	BOOL
32 点	Ch1_In	WORD	Ch1_Out	WORD
	Ch1_In00 ~ Ch1_In15	BOOL	Ch1_Out00 ~ Ch1_Out15	BOOL
	Ch2_In	WORD	Ch2_Out	WORD
	Ch2_In00 ~ Ch2_In15	BOOL	Ch2_Out00 ~ Ch2_Out15	BOOL
64 点	Ch1_In	WORD	Ch1_Out	WORD
	Ch1_In00 ~ Ch1_In15	BOOL	Ch1_Out00 ~ Ch1_Out15	BOOL
	Ch2_In	WORD	Ch2_Out	WORD
	Ch2_In00 ~ Ch2_In15	BOOL	Ch2_Out00 ~ Ch2_Out15	BOOL
	Ch3_In	WORD	Ch3_Out	WORD
	Ch3_In00 ~ Ch3_In15	BOOL	Ch3_Out00 ~ Ch3_Out15	BOOL
	Ch4_In	WORD	Ch4_Out	WORD
	Ch4_In00 ~ Ch4_In15	BOOL	Ch4_Out00 ~ Ch4_Out15	BOOL

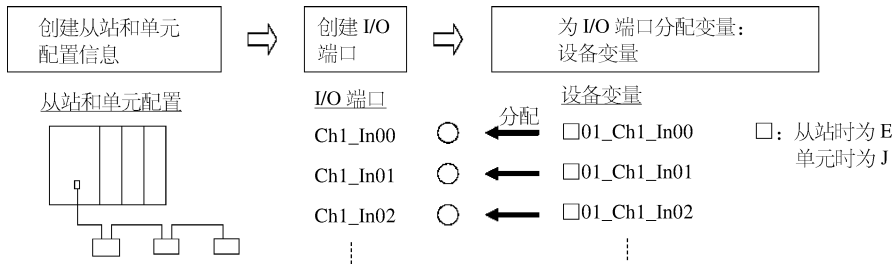
(3) CJ 系列特殊单元

由单元型号和功能决定 I/O 端口名称。例如，CJ1W-AD041-V1 模拟量输入单元：Ch1_PkHdCmd, Ch1_AveCfg 等。

3. 设备变量

NJ 控制器在 CPU 单元里并不为外部设备 (从站和单元) 分配指定的内存地址，而是为 I/O 端口分配变量，这些变量称为设备变量。

使用 Sysmac Studio 分配设备变量如下：



设备变量属性见表 2-12。

可以在用户程序或外部通信中指定设备变量访问设备 (从站或单元)。

4. 登记设备变量

在 Sysmac Studio 的 I/O 映射中,为 I/O 端口分配设备变量,设备变量会登记在变量表中。

表 2-12 设备变量属性

属性	设 置	改变设置
变量名称	自动生成的变量：[Device_name] + [I/O_port_name] 默认的设备变量名称如下： 对于 EtherCAT 从站，E 开头后面跟着从 001 开始的序列号 对于 CJ 系列单元，J 开头后面跟着从 01 开始的序列号 如果手动输入，变量名称是输入的字符串	允许
数据类型	根据 I/O 端口的数据类型	允许
AT 规格	EtherCAT 从站的设备变量： ECAT ://node#[node_number]/[I/O_port_name] CJ 系列单元的设备变量 IOBus ://rack#[rack_number]/slot#[slot_number]/[I/O_port_name]	不允许
保持	EtherCAT 从站的设备变量：不保持 CJ 系列单元分配给操作区 (CIO) 的设备变量：不保持 CJ 系列单元分配给设置区 (DM) 的设备变量：保持	不允许
初始值	无	允许
常量	无	允许
网络公开	不公开	允许
边沿	无	不允许

有下面三种方式分配设备变量。

(1) 手动输入设备变量名称

用这种方法分配设备变量，可为从站 I/O 端子或基本 I/O 单元分配用户自己指定的名称。

步骤如下：

- 1) 创建从站配置信息或单元配置信息。
- 2) 在 I/O 映射中选择 I/O 端口，在变量栏输入一个变量名称，如图 2-56 所示。

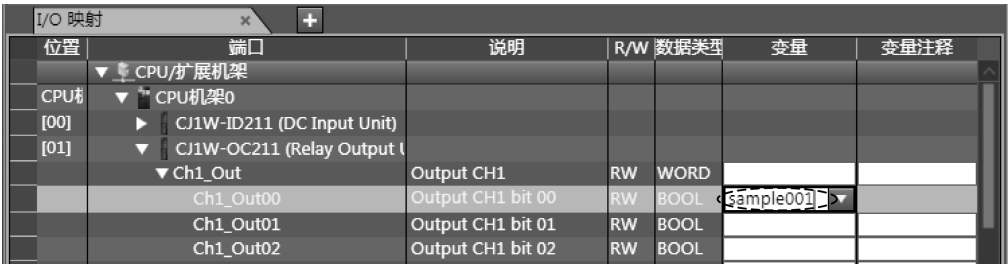


图 2-56 手动输入设备变量名称

每一个从站或单元的 I/O 端口自动分配了设备变量。这些设备变量也是自动登记在变量类型列设定的变量表中。

(2) 自动创建新的设备变量名称

从 I/O 端口名称和设备名称的组合中，设备变量自动命名。在下列情况下，使用这种方

法分配设备变量。

当不想花费时间手动输入设备变量名称。

自动创建设备变量名称，对特殊单元访问运行的数据和设置数据。

步骤如下：

1) 创建从站配置信息或单元配置信息。

2) 在 EtherCAT 编辑器或单元编辑器中设置设备名称，如图 2-57 所示。

默认设备名称如下：

对于从站，设备名称以 E 开头，跟着从

001 开始的顺序号。

对于单元，设备名称以 J 开始，跟着从 01 开始的顺序号。

3) 在 I/O 映射中，右击从站、单元、或一个或多个 I/O 端口，然后从菜单中选择“创建设备变量”，如图 2-58 所示。

项目名称	值
设备名称	J02
产品型号名...	CJ1W-OC211
产品名称	Relay Output Unit
规格	250V AC/24V DC max., 2A, 16 output...
机架号	0
槽号	1

图 2-57 输入设备名称

I/O 映射					
位置	端口	说明	R/W	数据类型	变量
▼ CPU/扩展机架					
CPU机架0					
▼ CPU机架0					
[00]	▶ CJ1W-ID211 (DC Input Unit)				
[01]	▼ CJ1W-OC211 (Relay Output Un				
	▼ Ch1_Out	Output CH1	RW	WORD	J02_Ch1_Out ▼
	Ch1_Out00	Output CH1 bit 00	RW	BOOL	J02_Ch1_Out00
	Ch1_Out01	Output CH1 bit 01	RW	BOOL	J02_Ch1_Out01
	Ch1_Out02	Output CH1 bit 02	RW	BOOL	J02_Ch1_Out02
	Ch1_Out03	Output CH1 bit 03	RW	BOOL	J02_Ch1_Out03
	Ch1_Out04	Output CH1 bit 04	RW	BOOL	J02_Ch1_Out04
	Ch1_Out05	Output CH1 bit 05	RW	BOOL	J02_Ch1_Out05
	Ch1_Out06	Output CH1 bit 06	RW	BOOL	J02_Ch1_Out06
	Ch1_Out07	Output CH1 bit 07	RW	BOOL	J02_Ch1_Out07
	Ch1_Out08	Output CH1 bit 08	RW	BOOL	J02_Ch1_Out08
	Ch1_Out09	Output CH1 bit 09	RW	BOOL	J02_Ch1_Out09
	Ch1_Out10	Output CH1 bit 10	RW	BOOL	J02_Ch1_Out10
	Ch1_Out11	Output CH1 bit 11	RW	BOOL	J02_Ch1_Out11
	Ch1_Out12	Output CH1 bit 12	RW	BOOL	J02_Ch1_Out12
	Ch1_Out13	Output CH1 bit 13	RW	BOOL	J02_Ch1_Out13
	Ch1_Out14	Output CH1 bit 14	RW	BOOL	J02_Ch1_Out14
	Ch1_Out15	Output CH1 bit 15	RW	BOOL	J02_Ch1_Out15

自动生成设备变量

图 2-58 自动创建设备变量名称

对于 I/O 端口或从站，自动分配设备变量。这些设备变量也自动登记在“变量”栏指定的变量表中。

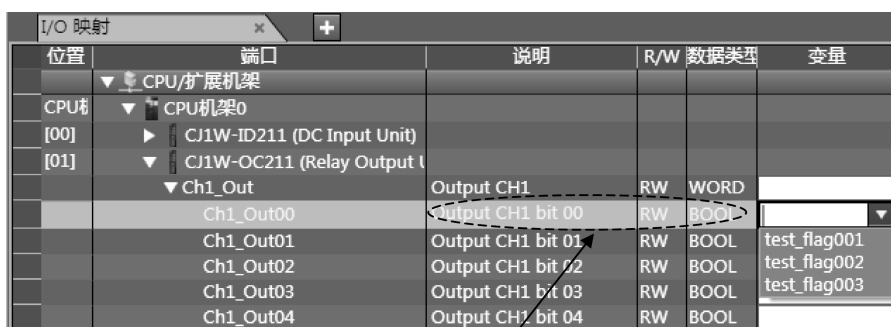
(3) 从已登记的变量中选择

在 I/O 映射中选择一个已登记的变量。如果先开始编程，已生成了变量，而后创建从站配置信息或单元配置信息；或移植使用其他工程的程序。以下两种情况下可用这种方法分配设备变量。

1) 输入程序。

2) 创建从站配置信息或单元配置信息。

3) 在 I/O 映射中, 从程序已创建的变量列表选择一个变量, 把它分配给一个 I/O 端口, 如图 2-59 所示。



选择一个已经在全局变量表中登记的用户定义变量

图 2-59 从已登记的变量中选择

如果要去掉分配给 I/O 端口的变量, 操作如下: 右击变量, 从菜单中选择“删除变量”, 设备变量被删除, 同时变量从登记的变量表中被删除; 或右击变量, 从菜单中选择“重置分配”, 设备变量被去掉, 然而重置分配并未将变量从登记的变量表中删除。

2.3.4 控制器设置

1. 操作设置

双击多视图浏览器中“配置和设置-控制器设置”下的“操作设置”, 显示出操作设置标签页, 如图 2-60 所示, 视图中 2 行连续显示 2 个图标, 代表 2 个项目: 基本设置和 FINS 设置, 点击某个图标后会显示出对应项目的设置内容。



图 2-60 操作设置

基本设置是指设置 CPU 单元支持的功能, 例如, 上电时运行模式的定义。基本设置的内容见表 2-13。

表 2-13 基本设置的内容

参数	设置组	说 明	设定值	默认	改变时刻	RUN 模式 下改变
操作 设置	启动模式	设置 CPU 单元启动时的操作模式	RUN 或 PRO-GRAM 模式	RUN 模式	当下载到 CPU 单元时	不允许
SD 存储 卡设置	启动时存储卡 诊断	设置电源 ON 时是否对插入的 SD 存储卡执行自诊断(文件系统检查和恢复)	不检查 检查	不检查	当下载到 CPU 单元时	不允许
系统服 务监视 设置	系统服务执行 时间 [ms]	设置系统服务执行的时间	10ms ~ 1s	10ms	当下载到 CPU 单元时	不允许
	系统服务执行 时间比 [%]	设置系统服务执行在 CPU 单元整个进程中所占的时间比	5% ~ 50%	10%	当下载到 CPU 单元时	不允许
安全 设置	启动时 写保护	当控制器的电源 ON 时, 自动使能写保护	不使用 使用	不使用	当电源 ON 时	支持

2. 内置 EtherNet/IP 端口设置

双击多视图浏览器中“配置和设置-控制器设置”下的“内置 EtherNet/IP 端口设置”，显示出内置 EtherNet/IP 端口设置标签页，视图中 7 行连续显示 7 个图标，代表 7 个项目，点击某个图标后会显示出对应项目的设置内容。

内置 EtherNet/IP 端口设置的 7 个项目依次为：

- (1) TCP/IP 设置
- (2) 链接设置
- (3) FTP 设置
- (4) NTP 设置
- (5) SNMP 设置
- (6) SNMP 陷阱设置
- (7) FINS 设置

2.3.5 运动控制设置

1. 轴

使用运动控制功能对 EtherCAT 伺服驱动或编码器输入从站操作时，要在 Sysmac Studio 中创建轴并且将它们定义为 EtherCAT 伺服轴或编码器轴。这样，轴变量自动生成为系统定义的变量，如图 2-61 所示。

在用户程序的运动控制指令中，可以指定轴变量，这样可以很容易地对伺服驱动和编码器输入从站访问和操作。

2. 轴变量和轴组变量

轴变量和轴组变量都是系统定义的变量，当使用 Sysmac Studio 创建轴或轴组时，轴变量或轴组变量将自动生成。轴变量用于单轴控制，轴组变量用于多轴协调控制。轴变量和轴

组变量说明见表 2-14。

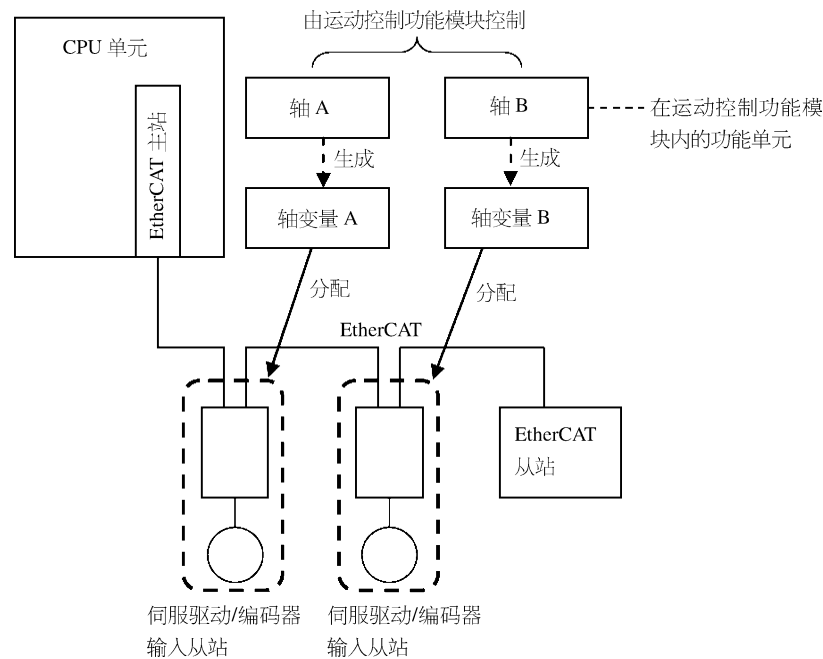


图 2-61 创建轴

表 2-14 轴变量和轴组变量说明

类型	名 称	
	轴变量	轴组变量
用 Sysmac Studio 生成的变量名称	MC_Axis* * * “* * *” 是顺序号，按照变量生成的顺序从 000 开始按升序分配。需要时可以改变名称	MC_Group* * * “* * *” 是顺序号，按照变量生成的顺序从 000 开始按升序分配。需要时可以改变名称
系统定义变量名称	_MC_AX [0..63] (数组元素序号按照变量生成的顺序从 00 开始按升序分配)	_MC_GRP [0..31] (数组元素序号按照变量生成的顺序从 00 开始按升序分配)

- 在两种情况下使用轴变量和轴组变量。
- 1) 在运动控制指令中指定轴和轴组变量：如果运动控制指令为一个 I/O 变量指定一个轴或轴组，可以对 OMRON 伺服驱动或编码器输入从站进行操作。
 - 2) 监视轴变量成员：使用指令监视伺服驱动和编码器输入从站的实际位置、错误信息或其他信息
- 轴变量的使用如图 2-62 所示。
- 1) 假设在 Sysmac Studio 创建轴，轴的名称为 A。基于系统定义变量自动创建轴变量，名称为 A。轴变量由轴基本设置、单位换算设置、I/O、操作状态、当前值、错误状态和警告状态组成。

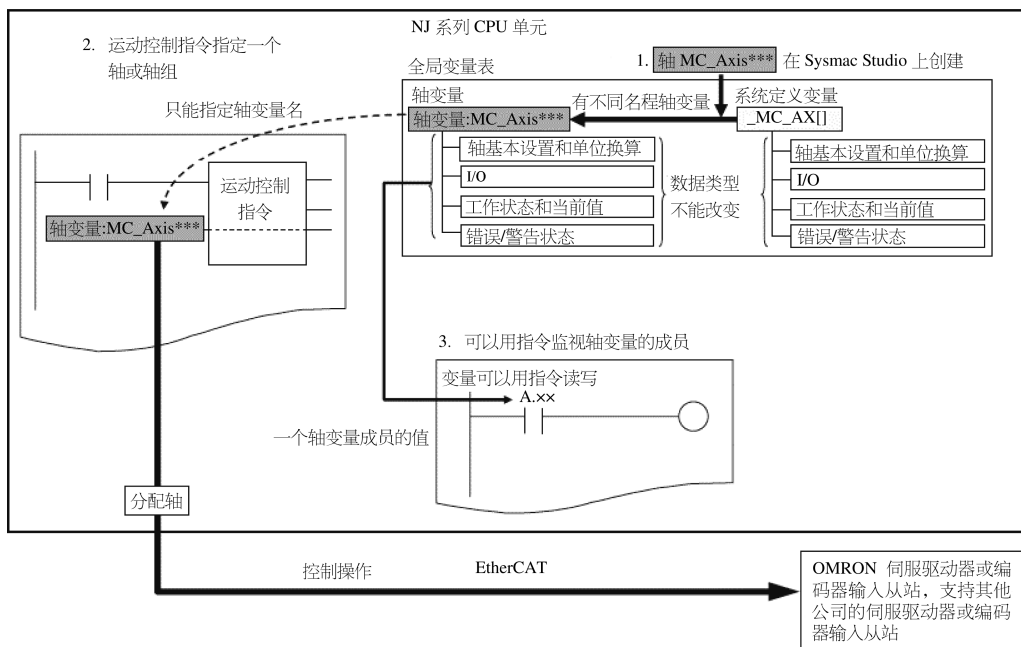


图 2-62 轴变量的使用

2) 为一个运动控制指令的输入-输出变量指定轴变量，名称为 A。用轴变量名称可以访问 OMRON 的伺服驱动或编码器输入从站，或其他公司支持的伺服驱动或编码器输入从站，并对其操作。

3) 需要时在指令中指定轴变量，监视伺服驱动或编码器输入从站，如实际位置、错误信息或其他信息。

3. 创建和使用轴和轴变量

按下列步骤创建和使用轴和轴变量。

1) 右击多视图浏览器中的“配置和设置-运动控制设置”下的“轴设置”，在弹出菜单中选中“添加-轴设置”。

如果需要，可以改变轴变量默认名称 MC_Axis***。“***”是序号，按照轴变量创建的顺序从 000 开始递增。

2) 从 Sysmac Studio 进行 EtherCAT 从站配置时，将创建的轴分配给伺服驱动器或编码器输入从站。

3) 从 Sysmac Studio 设置轴基本配置，包括：轴号、轴使用、轴类型、输入设备/输出设备。

4) 使用 Sysmac Studio 指定 MC 试运行需要的设置（单位换算、计数模式、限制等）和实际系统运行需要的设置，然后和工程一起传送到 CPU 单元。

5) 在用户程序的运动控制指令里，为输入-输出变量 Axis 指定轴名称。指定的轴名称（轴变量名称）是在运动控制设置中指定的，或是系统定义变量_MC_AX[0..63]。

对于一个已分配了轴的伺服驱动或编码器输入从站执行运动控制，如图 2-63 所示，轴变量名称为 MyAxis1。

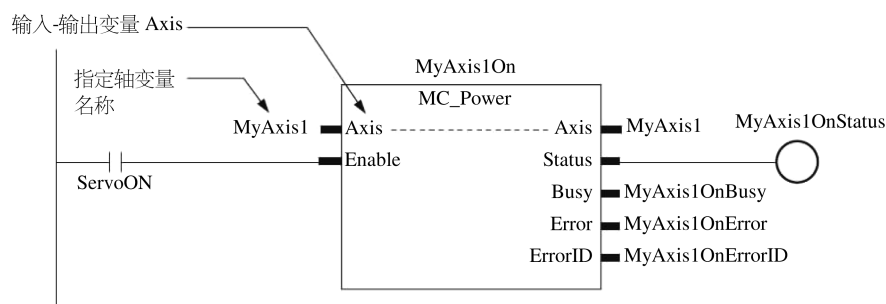


图 2-63 指令中使用轴变量

4. 运动控制参数设置

设置运动控制参数，包括下列参数。

- 1) 轴参数：单轴操作的运动控制参数。
- 2) 轴组参数：多轴协调操作的运动控制参数。

图 2-64 所示为运动控制参数设置。

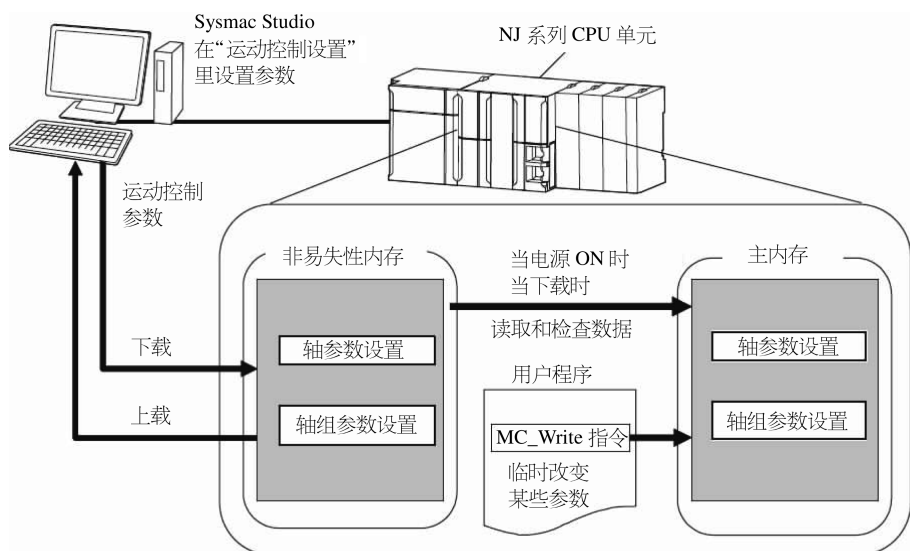


图 2-64 运动控制参数设置

可以用下面任何一种方法设置运动控制参数。

方法 1：在 Sysmac Studio 中设置运动控制参数

右击多视图浏览器中“配置和设置-运动控制设置”下的“轴设置”，显示出轴设置表，如图 2-65 所示。



图 2-65 轴参数设置

将运动控制参数下载到 CPU 单元，存储到 CPU 单元的非易失性内存中。当电源 ON 时，下载完成后，下载的设置生效。

方法 2：用 MC_Write 指令设置

可以用 MC_Write 指令临时覆盖某些运动控制参数。

运动控制参数设置的详细内容及操作参见第 7 章。

2.3.6 Cam 数据设置

Cam 数据设置用来创建电子 Cam 数据。Cam 数据是指 Cam 表中主轴相位和对应从轴位移之间的关系。

右击多视图浏览器中“配置与设置”下的“Cam 数据设置”，从菜单里选择“添加-CamProfile”，CamProfile 添加到多视图浏览器中，启动凸轮编辑器，设置凸轮数据。

2.3.7 任务设置

用来增加任务和设置任务。

双击“配置和设置”下的“任务设置”，显示出任务设置标签页，视图中 6 行连续显示 6 个图标，代表 6 个项目，点击某个图标后会显示出对应项目的设置内容。

任务设置的 6 个项目依次为：

- (1) 任务设置
- (2) I/O 控制任务设置
- (3) 程序分配设置
- (4) 任务中变量独占控制设置
- (5) 任务执行状态监视
- (6) 任务执行时间监视

任务设置的详细内容及操作参见 5.3 节。

2.3.8 数据跟踪设置

可以对指定的变量进行数据跟踪，无需任何额外的编程。右击多视图浏览器中“配置与设置”下的“运动跟踪设置”，从菜单里选择“增加-数据跟踪”，“数据跟踪”便添加到多视图浏览器中，启动数据跟踪编辑器，设置以下项目：跟踪类型、采样间隔、后触发数据比率、触发条件、跟踪的变量等。

数据跟踪设置的详细内容及操作参见 5.3 节。

第 3 章 NJ 的编程基础

NJ 编程包括以下内容。

1. 确认 I/O 和处理过程

- (1) 外部 I/O 设备和单元配置
- (2) 外部设备的刷新周期
- (3) 处理内容

2. 设计任务

- (1) 任务配置
- (2) 任务和程序之间的关系
- (3) 任务周期
- (4) 从站和单元的刷新时间
- (5) 任务之间变量的排他控制方法

3. 设计程序

- (1) 设计 POU

确定算法编程语言；设计程序、功能块和功能。

- (2) 设计变量

- 1) 定义在多个 POU 中使用的变量和在指定 POU 中使用的变量。
- 2) 定义用来访问从站和单元的设备变量名称。
- 3) 定义变量的属性，例如名称和保持属性等。
- 4) 定义变量的数据类型。

4. 编程操作

- (1) 登记变量

- 1) 用 Sysmac Studio 将在多个 POU 中使用的变量定义为全局变量。
- 2) 为每一个程序登记本地变量。
- 3) 为每一个功能块和功能登记本地变量。

- (2) 编写 POU

在 Sysmac Studio 的程序编辑器中,用选定的编程语言编写 POU(程序、功能块和功能)。

- (3) 任务设置

在 Sysmac Studio 的配置与设置下，对任务进行设置。

3.1 任务

3.1.1 任务类型

在 CPU 单元中，任务用来为一系列的过程，例如 I/O 刷新和用户程序，分配执行的条

件和执行的顺序。I/O 刷新是指 CPU 与 CJ 系列基本 I/O 单元、CJ 系列特殊单元和 EtherCAT 从站交换 I/O 数据。任务总是周期性地循环执行。根据执行条件和执行优先级，任务分为两种类型：主周期任务和周期任务。CPU 单元执行主周期任务或周期任务的时间称为任务周期。

1. 主周期任务

具有最高的优先级。每次循环执行时，主周期任务总要执行一次。总是在设定的周期时间内执行。运动控制和 EtherCAT 通信在主周期任务里执行。

主周期任务的任务周期可以设定为 500 μ s、1ms、2ms 或 4ms。

主周期任务只有一个，其执行优先级最高，固定为 4。

2. 周期任务

执行优先级比主周期任务低。周期任务是在主周期任务执行的间隔之间空余的时间内执行。周期任务周期性执行，其任务周期设定为主周期任务的整数倍。

周期任务有 3 个，优先级分别为 16、17 或 18。

一个任务可以分配 1~128 个程序。程序按照分配的顺序执行。

主周期任务、周期任务执行时，处理的内容是有区别的，如下所示：

主周期任务

I/O 刷新	用户程序执行	运动控制
--------	--------	------

优先级 16 的周期任务

I/O 刷新	用户程序执行
--------	--------

优先级 17 或 18 的周期任务

用户程序执行

I/O 刷新可分配给主周期任务或优先级 16 的周期任务，缺省情况下，I/O 刷新分配给主周期任务。运动控制总是分配给主周期任务。优先级 16 的周期任务不能执行运动控制。优先级 17 或 18 的周期任务只能执行用户程序，不能进行 I/O 刷新和运动控制。

3.1.2 任务的基本操作

1. 任务执行的优先级

CPU 单元首先执行优先级最高的任务。主周期任务具有最高的执行优先级，CPU 总是首先执行它。当一个任务正在执行时，另一个优先级高的任务的执行条件满足，则这个优先级高的任务被赋予了执行优先权。周期任务有 3 个执行优先级，如图 3-1 所示。

2. 任务执行的时序

主周期任务的执行周期称为主周期。任务将主周期作为标准周期来运行。

1) 控制器在整数倍主周期的时间内执行一次周期任务。例如，如果主周期任务的周期设为 1ms，优先级 16 的周期任务的周期设为 4ms，则在主周期任务执行 4 次的时间内，优先级 16 的周期任务执行一次。

2) 无论在运行模式还是编程模式，任

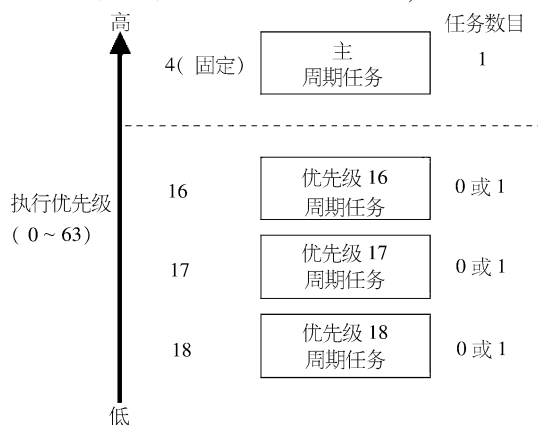


图 3-1 周期任务执行优先级

务都会被处理执行。但用户程序不能在编程模式下执行，只在运行模式下执行。

3) I/O 刷新根据任务周期被执行。

4) CPU 单元在任务执行时未用的时间内执行系统服务，例如通信处理。

图 3-2 所示为周期任务执行的时序，其中，图 3-2a 仅有主周期任务，图 3-2b 有主周期任务、优先级 16 的周期任务和优先级 17 的周期任务。

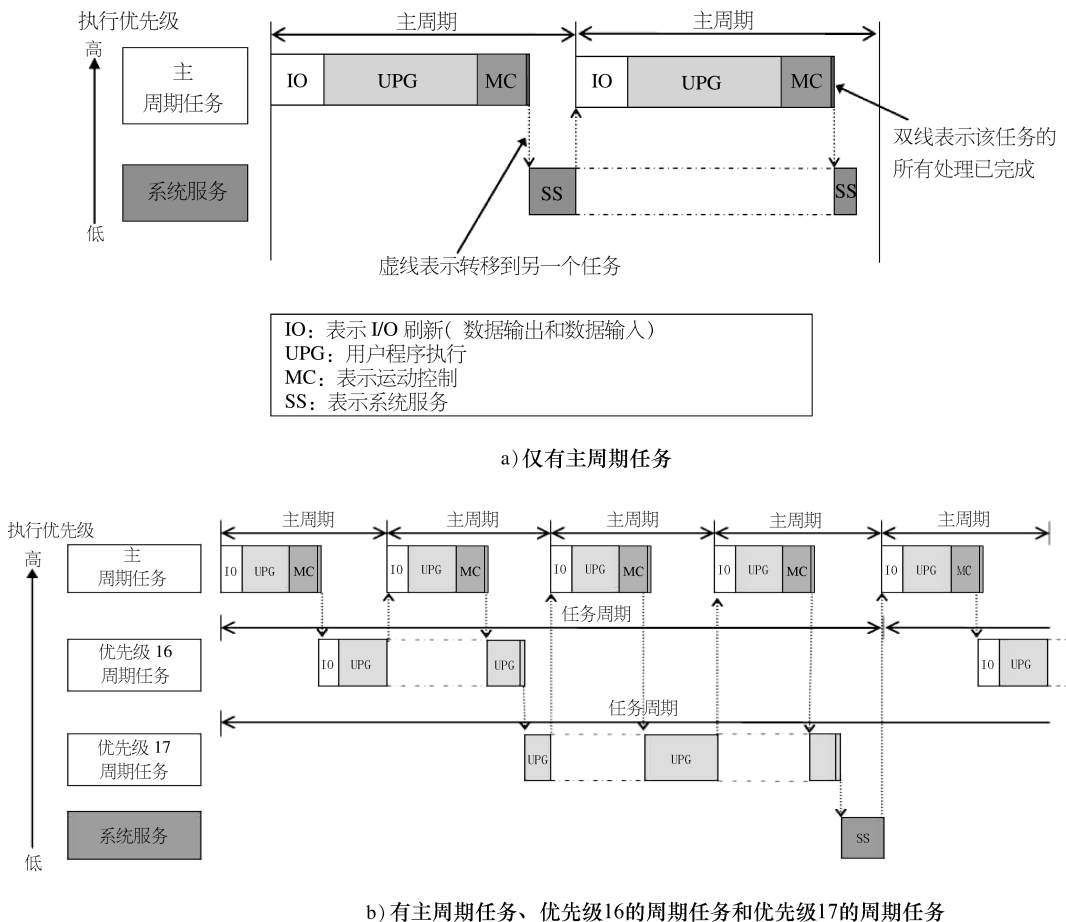


图 3-2 周期任务执行的时序

任务执行的顺序不仅依赖于执行优先级。低优先级任务有时在高优先级任务执行期间也能被执行。主周期任务不会被中断去执行其他任务。当主周期任务的执行完成后，优先级 16 的周期任务被启动执行。

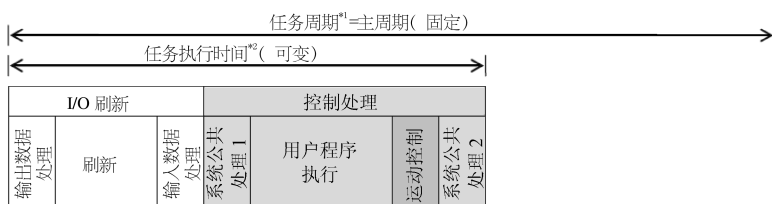
3. 任务和系统服务的处理

(1) 主周期任务

主周期任务有最高的优先级，执行处理时具有高速度、高精度。

如图 3-3 所示，在每一个执行周期，主周期要完成系统公共处理、I/O 刷新、用户程序执行和运动控制。与普通的周期任务不同，主周期任务要完成运动控制处理。

主周期任务处理的内容见表 3-1。



*1: 任务周期: 这是一个预置固定的时间, CPU 单元在此固定周期内执行任务

*2: 任务执行时间: 这是从执行条件满足开始到执行完成所花费的实际时间

图 3-3 主周期任务的处理

表 3-1 主周期任务处理的内容

处 理		处 理 内 容
I/O 刷新	输出数据处理	● 对需要 I/O 刷新的输出单元生成输出刷新数据 ● 如果设置强制刷新, 强制刷新值反映在输出刷新数据中
	刷新	● 与 I/O 交换数据
	输入数据处理	● 从 I/O 刷新的输入单元装载输入刷新数据 ● 如果设置强制刷新, 强制刷新值反映在读取的输入刷新数据中
系统公共处理 1		● 处理任务 (当访问任务设置) 中的变量的排他控制 ● 完成运动输入处理 ● 完成数据跟踪处理 (采样和触发检查)
用户程序执行		● 根据程序分配的顺序, 执行分配给任务中的程序
运动控制		● 执行用户程序中的运动控制指令 ● 在下一个主周期任务处理运动输出的 I/O 刷新
系统公共处理 2		● 处理任务 (当访问任务设置) 中变量的排他控制 ● 完成从控制器外部访问变量的处理, 以维持与任务执行 (在任务设置已设置变量访问时间) 的一致性 注意: 如果有 EtherNet/IP 标签数据链接的通信处理, 对标签 (也就是具有网络公开属性的变量) 设置了刷新任务, 变量访问处理在系统公共处理 2 完成

当主周期任务的用户程序中有运动控制指令时, 在随后的运动控制处理中, CPU 单元马上执行来自这些指令的结果。在下一个主周期任务的 I/O 刷新期间, CPU 单元将结果输出给伺服驱动器, 如图 3-4 所示。

(2) 周期任务

周期任务在每一个任务周期内执行自己的程序, 它的任务周期是主周期的整数倍。可以使用 0~3 个周期任务。

优先级 16 的周期任务也可以刷新 I/O。

不控制 I/O 周期任务的处理与控制 I/O 周期任务的处理是不同的。

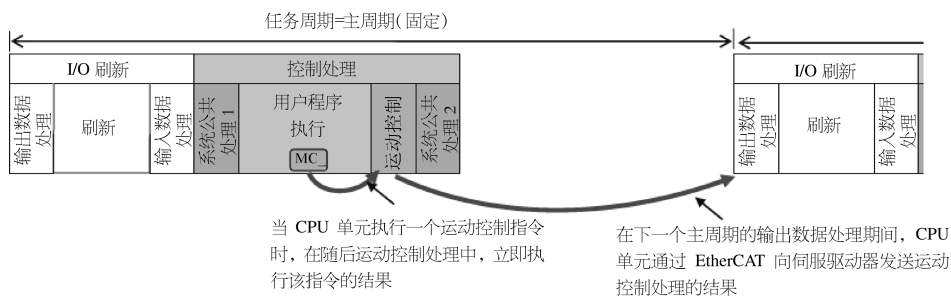


图 3-4 主周期任务对运动控制处理

1) 无 I/O 的周期任务。

如图 3-5 所示,无 I/O 周期任务处理的内容有:系统公共处理 1、用户程序执行、系统公共处理 2。

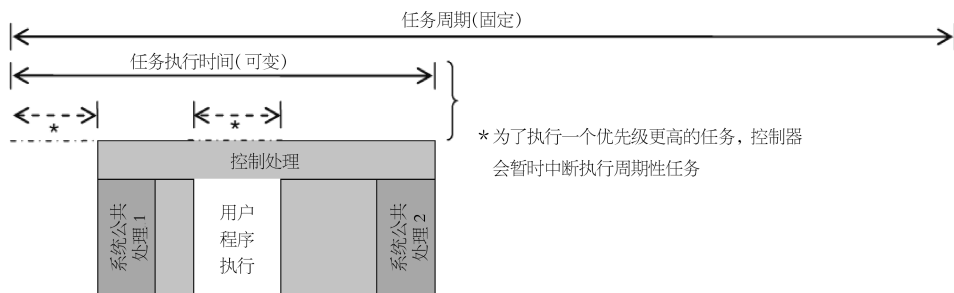


图 3-5 无 I/O 的周期任务

2) 有 I/O 的周期任务。

如图 3-6 所示,有 I/O 周期任务处理的内容有:有 I/O 刷新、系统公共处理 1、用户程序执行、系统公共处理 2。

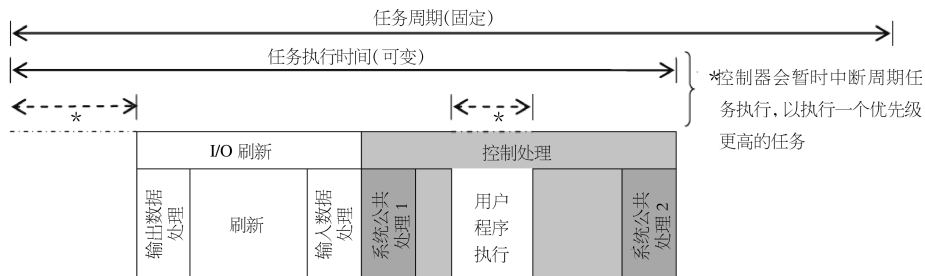


图 3-6 有 I/O 的周期任务

(3) 系统服务

系统服务在任务执行未用的时间被执行,如图 3-7 所示。

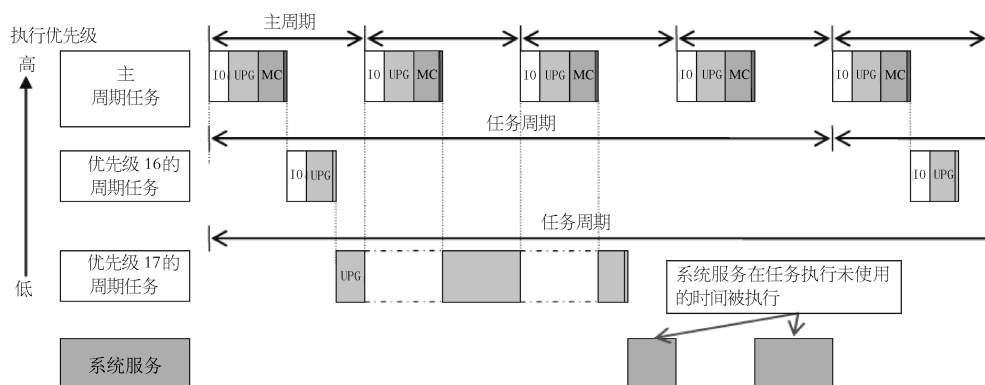


图 3-7 系统服务的执行

系统服务的内容见表 3-2。

表 3-2 系统服务的内容

系统服务	说 明
USB 端口服务	● 处理来自 Sysmac Studio 或 HMI 的服务请求，例如 CIP 命令
内置 EtherNet/IP 端口服务	● 处理信息服务请求，例如 CIP 命令，来自 Sysmac Studio、HMI、上位计算机或其他控制器 ● EtherNet/IP 标签数据链接通信处理 注意：如果有 EtherNet/IP 标签数据链接的通信处理，并对标签（也就是具有网络公开属性的变量）设置了刷新任务，则变量访问处理在系统公共处理 2 完成，不属于系统服务的内容
CJ 特殊单元服务	● CJ 系列特殊单元的事件服务 ● 通信指令（CIP）的执行 注意：在 I/O 刷新期间，CPU 单元在 CJ 系列特殊单元和它们分配的内存字之间交换数据
SD 存储卡服务 自诊断	● 从 FTP 服务器访问 ● 从 Sysmac Studio 对 SD 存储卡操作 ● SD 存储卡指令的执行
	● 硬件错误检测

3.1.3 保证任务间变量取值的一致性

如果多个任务读写同一个全局变量，可以使用下面两种方法中的任一种来保证任务之间变量取值的一致性。

1. 为任务中的变量设置排他控制功能

如图 3-8 所示，在任务中设置变量的排他控制功能，即可将任务设定为刷新一个全局变量或访问这个全局变量。变量只能由一个任务“写”，其他任务只能“读”，这就保证了全局变量取值的一致性。

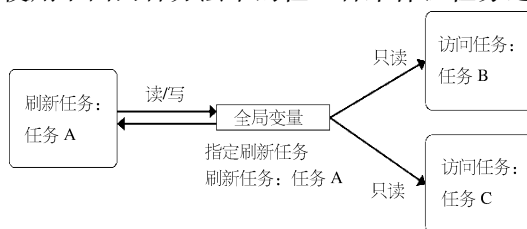


图 3-8 设置变量的排他控制

对于一个指定的全局变量，一个任务设定为其“写”值，这个任务称为刷新任务。如果设定了一个刷新任务，其他任务就不能“写”这个全局变量的值。

对于一个全局变量，如果没有设定刷新周期，则变量值可以被任何任务在任何时刻写，因此它的取值会改变，这取决于何时去读它。

如图 3-9 所示，刷新任务可以读写全局变量 GVar1，访问任务只能读 GVar1，保证了全局变量 GVar1 在任务间取值的一致性。

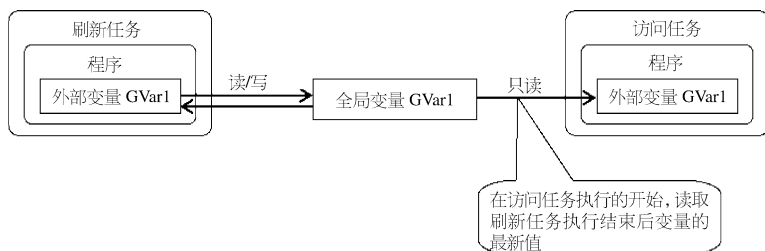


图 3-9 变量的排他控制举例

如图 3-10 所示，如果为一个全局变量设置了一个刷新任务，访问任务则在访问任务执行开始时，读取由刷新任务执行完成后写入的变量最新的值，这就可以在任务内维持全局变量值的一致性，而且不需要特殊的编程。

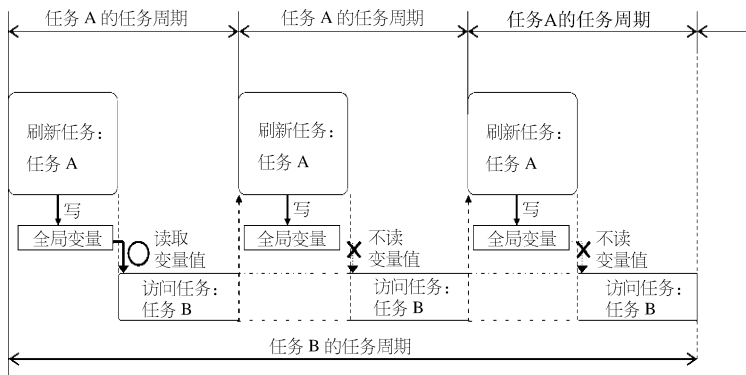


图 3-10 对设置排他控制的变量读写操作

如果在访问任务中用指令“写”一个全局变量的值，用 Sysmac Studio 检查程序时会出现错误。

对结构体或共用体变量设置刷新任务时，对于整个结构体或共用体变量只能设置一个刷新任务。不能对结构体或共用体不同的成员设置不同的刷新任务。

对数组变量设置刷新任务，对于整个数组变量只能设置一个刷新任务。不能对不同的数组元素设置不同的刷新任务。

对每一个全局变量只能设置一个刷新任务。如果需要在多个任务里写一个全局变量，使用下面介绍的任务排他控制指令来保证一致性。

2. 使用任务排他控制指令

当需要从多个任务里对同一个变量“写”值时，使用任务排他控制指令，即 LOCK 和

UNLOCK 指令。

LOCK 和 UNLOCK 指令阻止不同任务间的 LOCK 和 UNLOCK 程序区域的执行。

如图 3-11 所示，即使任务 B 比任务 A 的优先级高，如果任务 A 中的区域 1 正在执行，则任务 B 中的区域 1 不能执行，直到任务 A 中的区域 1 执行完毕才能执行。这里，任务 A 中的区域 1 的执行有特定的优先级。

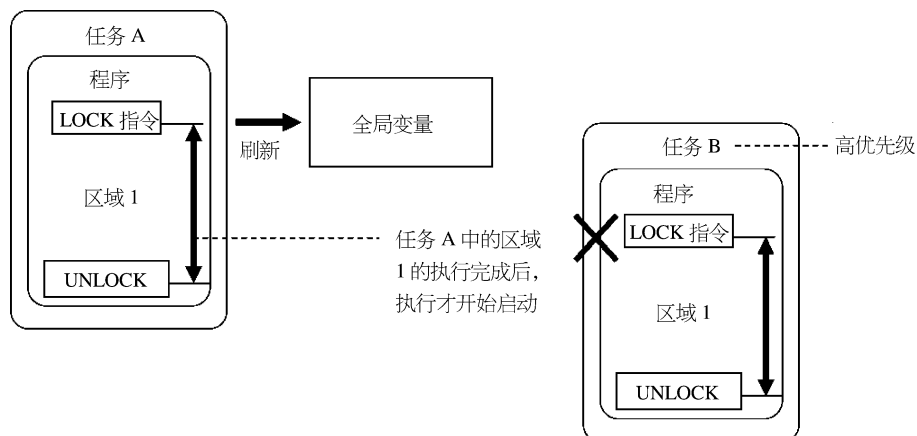


图 3-11 使用任务排他控制指令

如图 3-12 所示，当任务 A 中的区域 1 执行完成，任务 B 中的区域 1 才被执行。

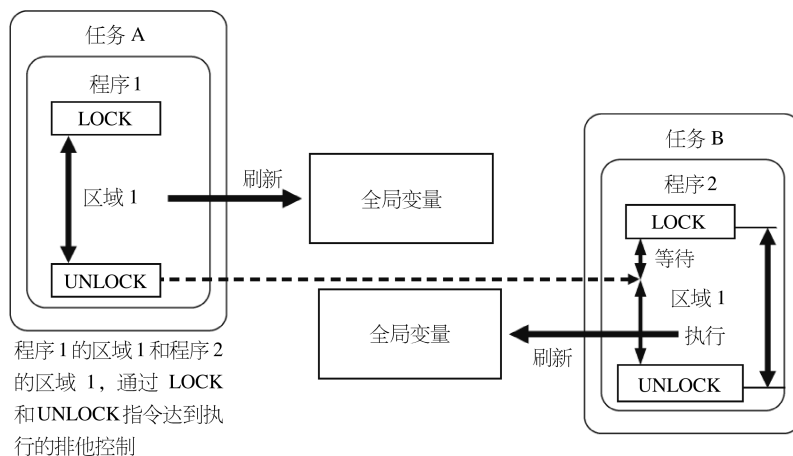


图 3-12 任务的排他控制

注意，不要使锁住的区域太长，否则会出现任务执行周期超出错误。

3.1.4 从控制器外部访问变量与任务执行的同步

如图 3-13 所示，为了达到从控制器外部访问变量与任务执行的同步，在任务设置里为任务中变量的排他控制做设置。同样，特别当使用标签数据链接时，在 Sysmac Studio 的任务设置里，设置变量访问时间。

使用下列方法从控制器外部访问全局变量。

1) EtherNet/IP 标签数据链接。

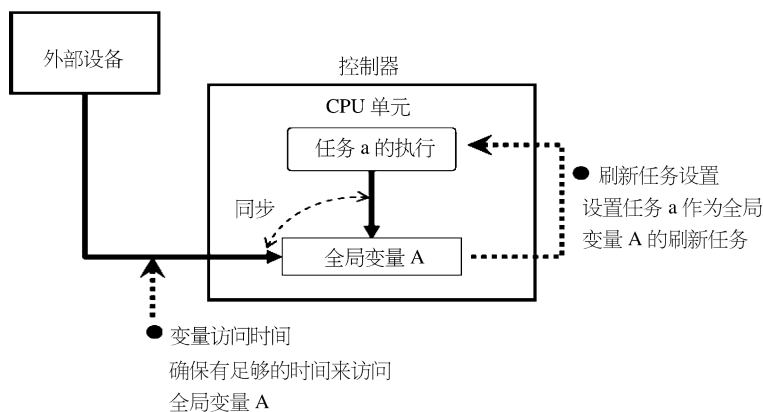


图 3-13 从控制器外部访问变量

2) 从 NS 系列 PT 访问变量。

3) 从 Sysmac Studio 访问变量（只对写操作实现与任务执行的同步）。

4) 从上位计算机用 CIP 通信访问变量。

当从控制器外部访问全局变量时，为这些全局变量设置特定的任务作为刷新任务。特别当使用标签数据链接时，对在同一标签组（有网络公开属性的变量）的标签，一定要设置同一任务为刷新任务，以保证标签组中标签间的一致性。

不能对下列已分配的全局变量设置刷新任务，下面给出的任务自动设置为刷新任务。

1) EtherCAT 从站的设备变量：设置为 I/O 控制任务的任务。

2) CJ 系列 I/O 单元的设备变量：设置为 I/O 控制任务的任务。

3) CJ 系列特殊单元的设备变量：主周期任务。

4) 用于 CJ 系列单元内存具有 AT 规格的变量：主周期任务。

3.1.5 任务设置

1. 任务设置

任务设置用来增加和建立任务，如图 3-14 所示。



图 3-14 任务设置

主周期任务和周期任务的参数见表 3-3。

表 3-3 主周期任务和周期任务的参数

参数	说明	设置值	默认	更新时间	在 RUN 模式下更改
任务类型	设置任务类型	优先级 4 的主周期任务 优先级 16 的周期任务 优先级 17 的周期任务 优先级 18 的周期任务	优先级 4 的主周期任务	下载到 CPU 单元时	不允许
	设置任务执行优先级	根据任务类型自动设置	主周期任务：4	下载到 CPU 单元时	不允许
任务名称	设置任务名称	字符串	主周期任务	下载到 CPU 单元时	不允许
周期/执行条件	设置任务周期	主周期任务：500 μ s、1ms、2ms、或 4ms 周期任务：1ms、2ms、3ms、4ms、5ms、8ms、10ms、15ms、20ms、25ms、30ms、40ms、50ms、60ms、75ms、或 100ms	主周期任务：1ms 周期任务：10ms 注意，周期任务的周期总是主周期任务周期的整数倍	下载到 CPU 单元时	不允许
任务周期超出检测	设置当任务周期超出时是否检测错误	● 检测 (轻微故障级别的控制错误) ● 不检测 (存储观察级别的日志记录)	检测	下载到 CPU 单元时	不允许
任务超时检测时间	设置任务执行超时时间。当超过超时时间时，将会产生一个任务执行超时错误	主周期任务和周期任务： 任务周期 $\times 1 \sim$ 任务周期 $\times 5$	主周期任务和周期任务： 5 个周期	下载到 CPU 单元时	不允许
变量访问时间 [%]	设置从控制器外部访问变量的时间占任务周期的百分比	1% ~ 50%	3%	下载到 CPU 单元时	不允许

2. I/O 控制任务设置

如图 3-15 所示，I/O 控制任务设置用来为输入和输出刷新执行设置时序。



图 3-15 I/O 控制任务设置

为任务分配 I/O 刷新见表 3-4。

表 3-4 为任务分配 I/O 刷新

分 类	分配单元	I/O 刷新目标	任务可能的分配
CJ 系列单元	单元	基本 I/O 单元	主周期任务和优先级 16 的周期任务
		特殊 I/O 单元	
		CPU 总线单元	
EtherCAT 从站	从站	分配给轴的从站	主周期任务
		其他从站	主周期任务和优先级 16 的周期任务

3. 程序分配设置

如图 3-16 所示, 程序分配设置用来为任务分配程序, 并设置程序执行的顺序。可以为一个任务分配多达 128 个程序, 不能将一个程序分配一个以上的任务。



图 3-16 程序分配设置

在一个任务内, 程序按照分配的顺序从上到下执行。

4. 任务中变量的排他控制设置

任务中变量的排他控制设置如图 3-17 所示, 用来设置刷新全局变量的任务和访问全局变量的任务。



图 3-17 任务中变量的排他控制设置

任务中变量的排他控制设置参数见表 3-5。

表 3-5 任务中变量的排他控制设置参数

参 数	说 明	设置值	默认	更新时间	在运行模式下更改
刷新的变量	设置在主周期任务或者周期任务中刷新的变量		无	当下载到 CPU 单元时	不允许
数据类型	设置变量的数据类型	无			
变量注释	给变量设置注释	无			
访问任务	设置访问变量的任务				

3.1.6 与任务相关的指令

与任务相关的指令见表 3-6，这些指令支持读取当前任务的状态，确定其他任务是否在执行中，完成为保证任务间区域一致性的排他控制。

表 3-6 与任务相关的指令

指令	指令名称	说 明	
GetMyTaskStatus	读当前任务状态	读取当前任务的下列状态： 最近一次的任务执行时间，最大的任务执行时间，最小的任务执行时间，任务执行次数，任务周期超出标志和任务周期超出次数	
Task_IsActive	确定任务状态	确定指定的任务是否正在执行	
Lock	任务加锁	启动任务间加锁	任何有相同加锁号加锁区的其他任务的执行被禁止
Unlock	任务解锁	停止任务间加锁	

3.1.7 与任务相关的系统定义变量

与任务相关的系统定义变量见表 3-7，可以显示每个任务的状态。例如，名称为

TaskName 的任务执行周期超出标志为_TaskName_Exceeded。不要在用户程序中使用这些变量，因为在更新它们的时候会有延迟，与功能模块的错误状态相关的一致性也有问题。这些变量只用在从 Sysmac Studio 进行数据跟踪时，采样任务状态。

用户程序中也可以使用指令 GetMyTaskStatus 和 Task_IsActive 来读取任务状态。不能直接通过系统定义变量来访问这些变量。

表 3-7 与任务相关的系统定义变量

指 令	含 义	说 明	数据类型	R/W
_TaskName_Active	任务活动标志	任务执行时为 TRUE 任务不执行时为 FALSE	BOOL	R
_TaskName_LastExecTime	最近一次任务执行时间	给出任务最近一次的执行时间	TIME	R
_TaskName_MaxExecTime	最大的任务执行时间	给出任务执行时间的最大值	TIME	R
_TaskName_MinExecTime	最小的任务执行时间	给出任务执行时间的最小值	TIME	R
_TaskName_ExecCount	任务执行次数	包含任务执行的次数 如果当前值超过数据类型的最大值， 当前值返回 0，计数继续	UDINT	R
_TaskName_Exceeded	任务周期超出标志	如果任务周期超出了为 TRUE 如果任务执行在任务周期内完成则 为 FALSE	BOOL	R
_TaskName_ExceedCount	任务周期超出次数	包含任务执行周期超出的次数 如果当前值超过数据类型的最大值， 当前值返回 0，计数继续	UDINT	R

图 3-18 所示为任务活动标志_TaskName_Active 的操作。

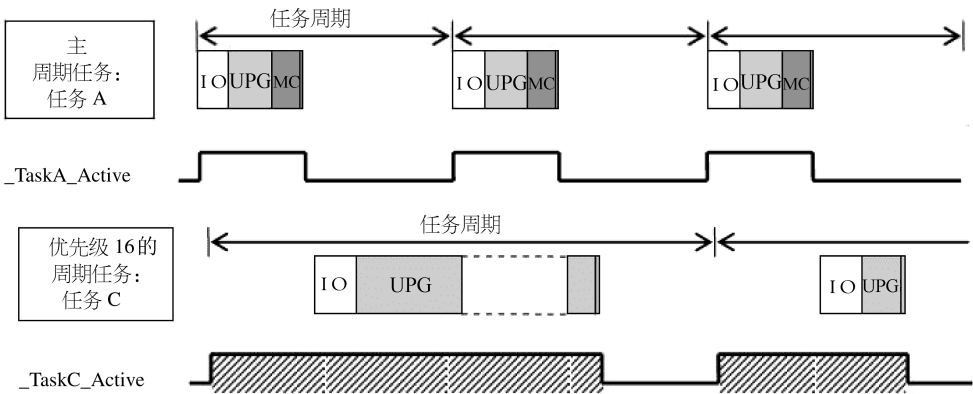


图 3-18 任务活动标志操作

图 3-19 所示为任务周期超出标志_TaskName_Exceeded 的操作。

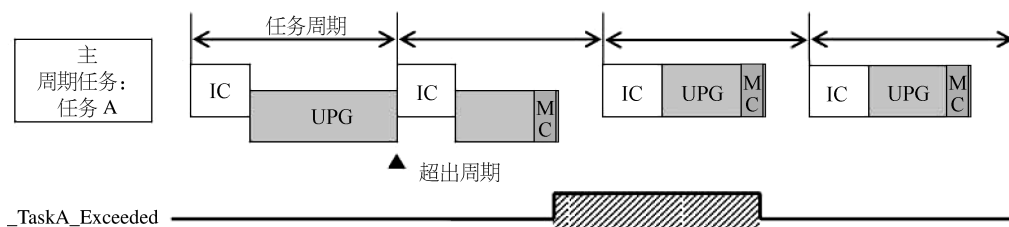


图 3-19 任务周期超出标志的操作

3.1.8 与任务相关的错误

与任务相关的错误如下：

1. 任务周期超出错误

如图 3-20 所示，如果任务执行时间超出指定的任务周期，就会出现一个任务周期超出错误。这是一个轻微故障级的控制器错误，即使发生此错误，运行仍将继续。它可以发生在主周期任务和周期任务中。

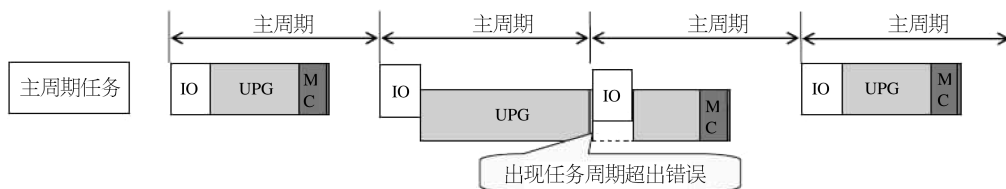


图 3-20 任务周期超出错误

可以在设置中将任务周期超出错误检测设为禁用。在 Sysmac Studio 的任务设置中，可设置使用任务周期超出检测，而默认设置为检测。

即使禁用检测任务周期超出错误，如果任务处理在指定周期内未完成，信息仍将输出到下面的标志：任务周期超出标志（_TaskName_Exceeded）、任务周期超出计数（_TaskName_ExceedCount）、控制器错误状态（_ErrSta）和事件日志。

如果在任务周期内，任务处理未完成，I/O 刷新如下操作。

输出：输出上一个周期的值。

输入：用户程序中不反映输入的刷新值。

如果出现任务周期超出错误，则缩短任务中的程序以适应任务周期，或增加任务周期的设定值。

2. 运动控制周期超出错误

如果在主周期（即运动控制周期）内，连续两次没有完成运动控制处理（MC），则出现运动控制周期超出错误。在运动控制功能模块会出现局部故障级控制器错误。同时，出现任务周期超出错误。

出现运动控制周期超出错误时，减少程序中处理的数目，或在不影响正常运行的前提下，增加控制周期。

3. 任务执行超时错误

如图 3-21 所示,如果在指定的任务执行超时时间内,未完成任务处理,则出现任务执行超时错误。

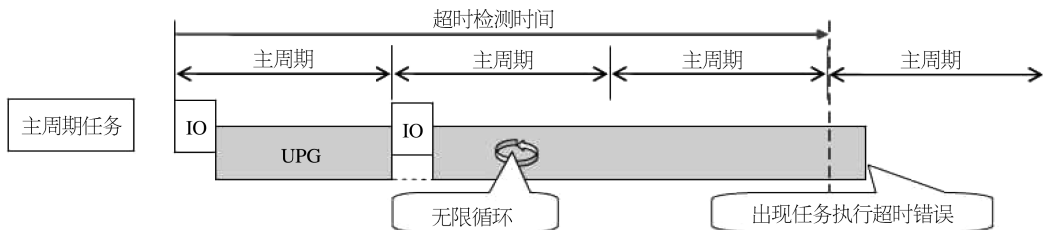


图 3-21 任务执行超时错误

这是一个重大故障级的控制器错误,出现时,用户程序执行将停止。

由于程序逻辑错误,如无限循环,正常的任务操作无法进行时,会发生此错误。

出现任务执行超时错误时,检查任务设置,下载工程,电源需断电再重新上电或 CPU 单元复位。

4. I/O 刷新超时错误

如图 3-22 所示,如果 I/O 刷新在指定的周期内连续两次未完成,则出现 I/O 刷新超时错误。

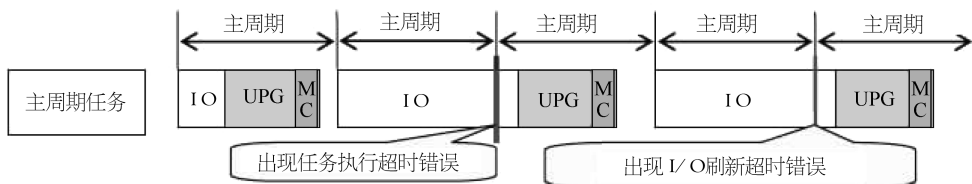


图 3-22 I/O 刷新超时错误

这是一个重大故障级的控制器错误,出现时,用户程序执行将停止。

这个错误只发生在主周期任务和设置了 I/O 刷新优先级 16 的周期任务。

出现 I/O 刷新超时错误时,检查任务设置,下载工程,电源需断电再重新上电或 CPU 单元复位。

5. 系统服务时间不足错误

如果不能获得为系统服务监视设置而指定的时间,会出现系统服务时间不足的错误。

这是一个重大故障级的控制器错误,出现时,用户程序执行将停止。

出现系统服务时间不足错误时,检查任务设置和系统服务监视设置,下载工程,电源需断电再重新上电或 CPU 单元复位。

3.1.9 监视任务执行状态和任务执行时间

可以从 Sysmac Studio 监视所有任务中的程序的执行状态和任务执行时间。

在多视图浏览器中,双击“配置和设置”下的“任务设置”,在显示的编辑面板点击“任务执行状态监视”按钮,可以观察到所有正在执行的任务/程序的名称和执行状态;点击

“任务执行时间监视”按钮，可以观察到所有正在执行的任务的执行时间。

3.2 程序组织单元

程序组织单元 (Program Organization Unit, POU) 是 IEC 61131-3 用户程序执行模型里定义的一个单元。一个 POU 包括一个本地变量表和一个算法 (也就是一系列编码或逻辑)，它是构成用户程序的基本单元。NJ 系列 CPU 单元里运行的用户程序是由 POU 组合而成的。

POU 按功能分为程序、功能块和功能。POU 的标准部分 (如标准功能块、标准功能等) 由控制器制造商提供。用户根据 POU 的定义设计用户程序的 POU，对 POU 调用和执行。

一个程序对应一个主路径，这是形成算法的 POU 的主要类型。可以将任何指令、功能、或功能块放置到程序算法中。

功能块可以输出不同的值，即使有相同的输入。功能块可以被程序或另一个功能块调用。

在相同的输入下，功能总是输出相同的值。功能可以被程序、另一个功能或功能块调用。

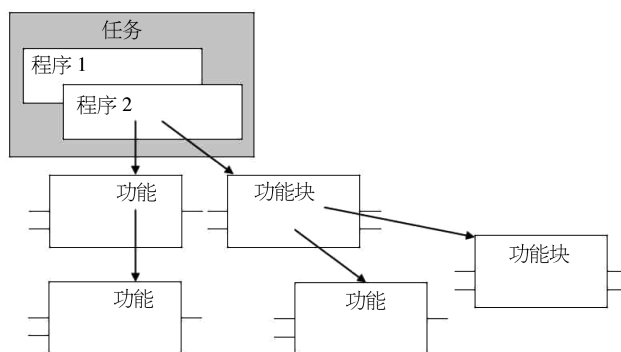


图 3-23 层次关系

POU 由这三种单元组合而成，可以生成许多 POU。程序要分配给任务才能执行。

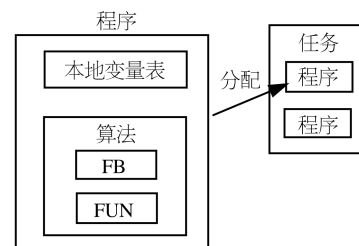
程序、功能块、功能与任务的层次关系如图 3-23 所示。

3.2.1 程序

程序由本地变量表和算法组成。在程序的算法里，可以使用任何功能块和功能。不能从其他 POU 中调用程序。当程序所在的任务执行时，程序才能执行。一个任务可以分配 128 个程序。

图 3-24 所示为程序结构。

在任务里，为所有的程序设置执行顺序。在 Sysmac Studio “任务设置”下的“程序分配设置”指定这个顺序。



不能调用其他 POU 中的程序

图 3-24 程序结构

所有的程序在本地变量 (也就是内部变量) 中有下面 3 个系统定义的变量。

P_First_RunMode: 第一次运行周期标志, BOOL 型, 只读。从 PROGRAM 模式转为 RUN 模式时, 当 CPU 单元开始运行, 只在第一次运行周期为 TRUE, 其他运行周期为 FALSE, 通常使用这个标志完成初始化处理。

P_PRGER: 指令错误标志, BOOL 型, 可读可写。当出现指令错误时, 变为 TRUE 并保持。当这个标志变为 TRUE 时, 一直保持直到程序将它变为 FALSE。

P_CY: 进位标志, BOOL 型, 只读。某些指令可以修改这个标志。

3.2.2 功能块

功能块作为一种程序组织单元,需要设置名称,而且不同的功能块要设置不同的名称。功能块具有静态变量(即具有记忆功能),要占用内存区。当需要对一些状态进行记忆,比如定时器、计数器指令,则使用功能块。功能块的输出与内部变量和外部变量的状态有关,因此,即使相同的输入也能产生不同的输出结果。

功能块分为标准功能块和用户定义功能块。标准功能块由系统预先定义,可由其他功能块和程序调用。用户定义功能块则由用户根据需要自行创建,可以像标准功能块一样使用。

1. 功能块结构

功能块可以用梯形图或 ST 来表示。在梯形图中,功能块用一个方框来表示,图 3-25 所示为梯形图表示的一个功能块。

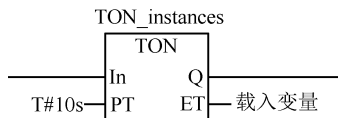


图 3-25 梯形图表示的一个功能块

功能块结构如图 3-26 所示。

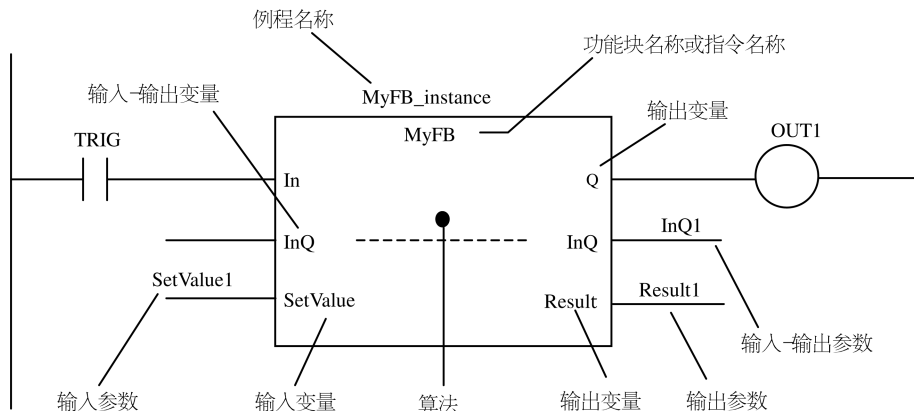


图 3-26 功能块结构

功能块定义由本地变量表和算法组成,如图 3-27 所示。

功能块各组成部分说明如下:

(1) 功能块名称或指令名称

生成功能块时,在功能块定义时,分配的功能块名称或指令名称

(2) 例程名称

给程序中的功能块例程一个例程名称,以便能够管理它。

(3) 算法

可以用梯形图或 ST 编写一个算法。在算法中,可以使用任何指令、用户定义的功能,或用户定义的功能块。

(4) 本地变量表

本地变量表用来定义输入变量、输出变量、输入-输出变量、内部变量和外部变量。

(5) 参数

输入变量的输入参数:当功能块执行开始时,输入参数把值传给功能块中的输入变量。输入参数可以是一个变量或常数。

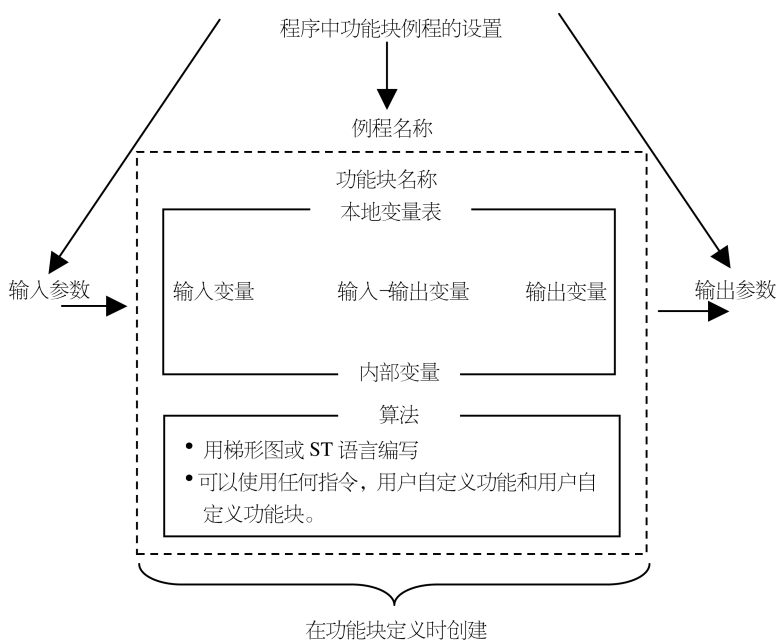


图 3-27 功能块定义

输出变量的输出参数：当功能块执行完毕后，输出参数接收功能块中输出变量的值。

输入-输出变量共享的输入-输出参数：在功能块中改变输入-输出参数的值，同一参数既可以用作输入也可以用作输出。

2. 功能块变量的指定

功能块变量的指定如图 3-28 所示。

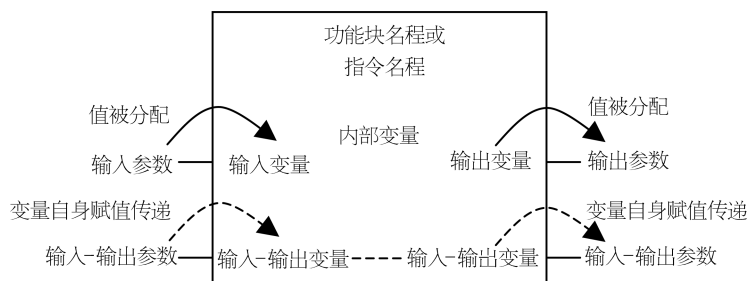


图 3-28 功能块变量的指定

功能块变量的说明见表 3-8。

表 3-8 功能块变量的说明

变量	数量	规 范
输入变量	1 ~ 64	输入变量用作功能块内部的输入参数。它们不能在功能块内部被改变
		- 当功能块执行时，输入变量被置为输入的参数值 - 可以将输入参数指定为常量或者变量 - 省略输入参数

(续)

变量	数量	规 范
输入变量	1 ~ 64	● 至少需要一个 BOOL 输入变量 ● 可以指定检测什么时候变量更改为 TRUE 或 FALSE ● 可以从功能块外部访问输入变量的值。使用以下格式访问这些值: InstanceName. InputVariable_Name。但是, 不能直接给输入变量赋值
输出变量	1 ~ 64	输出变量用作功能块的输出参数 ● 输出参数在功能块执行之后设置为输出变量的值 ● 不可以为输出参数指定一个常量, 必须指定一个变量 ● 至少需要一个 BOOL 输出变量 (包括 ENO) ● 可以省略输出参数的连接: 如果省略一个输出参数, 这个输出变量的值不分配给任何参数 ● 可以从功能块外部访问输出变量的值。使用以下格式访问这些值: InstanceName. OutputVariable_Name。但是, 不能直接给输出变量赋值
输入-输出变量	0 ~ 64	输入-输出变量用作功能块的输入和输出。它们可以在功能块内部被改变 ● 一个输入-输出参数的值传递给一个输入-输出变量, 然后该输入-输出变量的值传递给输入-输出参数 ● 不可以为输入-输出参数指定一个常量, 必须指定一个变量 ● 如果改变一个功能块的输入-输出变量的值, 那么输入-输出参数的值也随之改变 ● 不能省略输入-输出参数
内部变量	无限制	内部变量在各功能块内用作暂时存储 ● 无论功能块是否执行, 内部变量的值都会被保持 ● 内部变量可以具有保持属性 ● 不能从功能块外部访问内部变量的值
外部变量	无限制	外部变量用于访问全局变量
EN	0	EN 变量不能在功能块中使用 (这适用于用户定义功能块和 FB 指令)
ENO	0 或 1	通常, 这是 BOOL 输出变量, 正常结束时设置为 TRUE, 出错时设置为 FALSE ● 在一些功能块指令中可以省略它

3. 功能块调用

功能块既可用梯形图, 也可用 ST 调用。

从 ST 调用功能块的方法如下

```
instance_name(input_variable_1:= input_parameter_1,...,
              input_variable_L:= input_parameter_L,
              input_output_variable_1:= input_output_parameter_1,...,
              input_output_variable_M:= input_output_parameter_M,
              output_variable_1:=> output_parameter_1,...,
              output_variable_N:=> output_parameter_N);
```

输入赋值使用 “:=”; 输出赋值使用 “=>”; 输入-输出赋值时两种都可以用, 一般用 “:=”。

图 3-29 所示为从 POU 调用功能块的梯形图。

ABC: 功能块例程名称。

A 和 B: 功能块的输入或输入-输出变量名称。

x1 和 x2: 调用功能块的 POU 的输入或输入-输出参数。

C: 功能块的输出变量。

y1: 调用功能块的 POU 的输出参数。

用 ST 调用功能块的方法有两种。

1) 给出功能块的变量名程和调用功能块的 POU 的参数名程

ABC(A:=x1,B:=x2,C=>y1);

- 括号内参数和返回值的顺序任意。

- 可以省略输入变量的名称和输入参数名称。如果省略这些名称, 上一次调用时分配给输入变量的值又被分配给输入变量。如果第一次调用功能块, 输入变量置为它们的初始值。

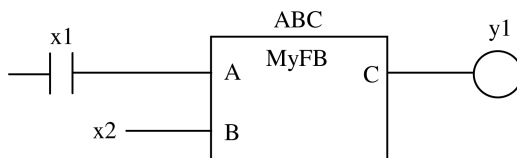


图 3-29 从 POU 调用功能块的梯形图

- 可以省略输出变量的名称和输出参数名称。如果它们被省略, 输出变量的值不被分配。

2) 省略功能块的变量名称, 给出调用功能块的 POU 的参数名称

ABC(x1,x2,y1);

参数的顺序基于功能块定义。顺序与功能块本地变量定义相同, 从顶部到底部。

ST 调用功能块的各种方式见表 3-9。

表 3-9 ST 调用功能块的各种方式

方法	表示方式		注释
方法 1	包括的参数	举例	可用
	给出所有的参数	instance (A:= x1, B:= x2, C:=> y1);	
	给出多个参数	instance (A:= x1, B:= x2);	
	给出一个参数	instance (B:= x2);	
	不给出任何参数	instance (A:=);	
方法 2	给出所有的参数	instance (x1, x2, y1);	可用
	所有的参数都不给出	instance ();	
	只给出第一个参数	instance (x1);	不可用
	给出一个参数	instance (x2, ,);	
	给出多个参数	instance (x1, x2);	

图 3-30 所示为调用功能块的梯形图。

用 ST 调用此功能块, 两种编写方法如下。

方法 1:

```
ChangeFixToFloat(Execute:= Exe, Data32_1:= FixPointData1, Data32_2:= FixPointData2,
NoOfDigit_1:= FixPointPos1, NoOfDigit_2:= FixPointPos2,
OK=> Done, Double_1=> RealData1, Double_2=> RealData2);
```


方法 2:

```
ChangeFixToFloat(Exe, FixPointData1, FixPointData2, FixPointPos1, FixPointPos2,
Done, RealData1, RealData2);
```

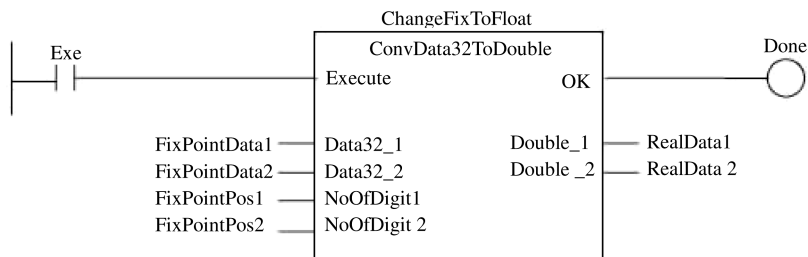


图 3-30 调用功能块的梯形图

4. 功能块的创建

除了系统提供的标准功能块外，用户可以创建自定义的功能块，由程序或其他功能块来调用。

下面创建一个时钟脉冲发生器的功能块，输入参数为接通时间和断开时间，根据输入参数输出不同占空比的脉冲。

操作步骤如下：

1) 如图 3-31 所示，右击多视图浏览器中“编程-POUs-程序”下的“功能块”，从菜单中选择“添加-梯形图”，会默认生成一个叫做功能块 0 的功能块，如图 3-32 所示。功能块可以用梯形图或 ST 来编写，这里选用梯形图。



图 3-31 添加功能块



图 3-32 生成功能块 0

2) 双击功能块 0，进入编程界面，上半部为变量表，有“内部”、“输入/输出”“外部”3 个变量表，如图 3-33 所示。

在内部变量表添加两个 BOOL 变量 a 和 b，如图 3-34 所示。在输入/输出变量表添加 timeon、timeoff、pulse、in 4 个变量，如图 3-35 所示。

3) 在功能块 0 的编程界面输入梯形图，如图 3-36 所示。

BOOL 变量 in 为控制输入，控制时钟脉冲的启动/停止。

这样就创建了一个自定义功能块。



图 3-33 功能块 0 编程界面



图 3-34 内部变量表



图 3-35 输入/输出变量表

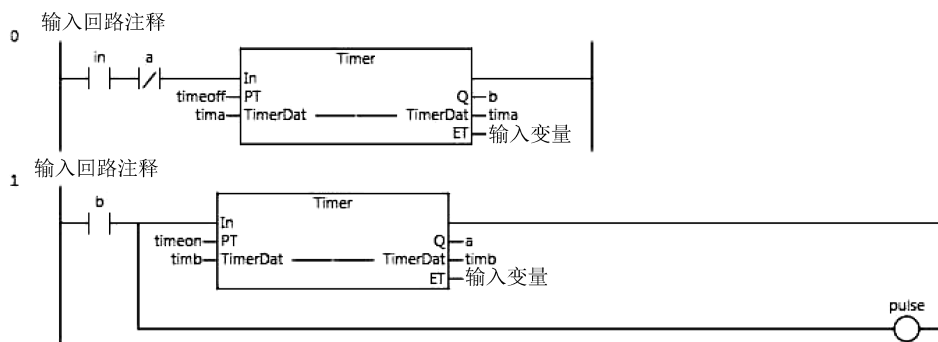


图 3-36 功能块 0 的梯形图

创建自定义功能块后，就可以在编程时调用它。现在回到主程序下，对内部变量进行定义，如图 3-37 所示。



图 3-37 程序中内部变量定义

输入调用功能块的梯形图程序，输入到功能块时，键入“功”，就发现刚才定义的功能块 0 已经存在了，如图 3-38 所示。

完整的调用功能块梯形图如图 3-39 所示。

timeondata、timeoffdata 和 start 为功能块的输入参数，testp 为输出参数。其中，timeondata 为脉冲 ON 的时间，timeoffdata 为脉冲 OFF 的时间，start 控制脉冲发生器的启动或停止，testp 为脉冲发生器的输出。

timeondata、timeoffdata 的初值设为 10，这是个振荡周期 2s 的时钟脉冲，开 1s 关 1s，在线仿真调试后，如图 3-40 所示。

可以通过更改 timeondata、timeoffdata 的值，改变脉冲的占空比。在监视选项卡中修改，如图 3-41 所示，以上两个参数改为 30、10 后，就变成一个 4s 周期的脉冲信号，其中导通 3s、关断 1s。

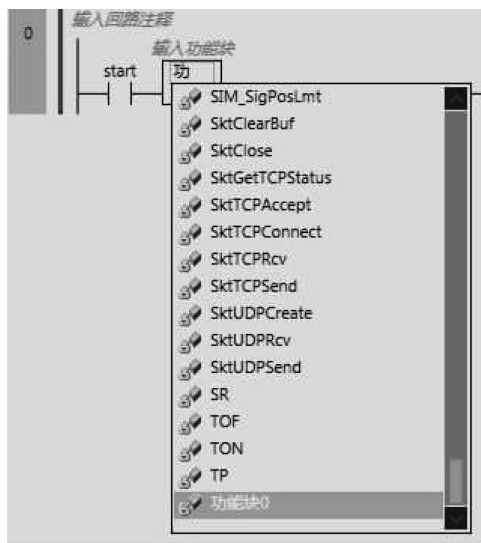


图 3-38 输入功能块 0



图 3-39 调用功能块梯形图

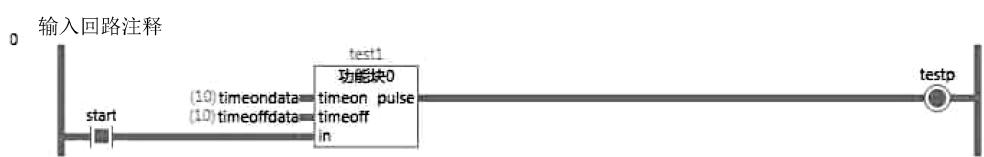


图 3-40 仿真调试

名称	在线值	修改	数据类型	分配到	显示格式
Program0.timeondata	30	30	UINT		Decimal
Program0.timeoffdata	10		UINT		Decimal
输入名称..					

图 3-41 结果显示

3.2.3 功能

功能 (FUN) 也是一种程序组织单元, 但与功能块不同, 功能不需要设置名称, 因此输入简单。功能没有静态变量, 即没有记忆, 不占用内存, 当只需要简单的操作运算不需要对状态进行记忆的时候使用功能。功能一般只要输入相同的值就会输出一样的结果。

功能分为标准功能和用户定义功能。标准功能由系统预先定义, 可由其他功能、功能块和程序调用。用户定义功能由用户根据需要进行自行创建, 可以像标准功能一样使用。

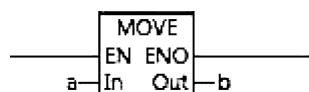


图 3-42 梯形图表示的一个功能

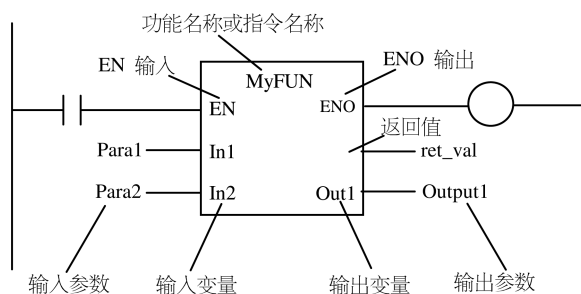


图 3-43 功能结构

1. 功能结构

功能可以用梯形图或 ST 来表示。在梯形图中, 功能表示为一个方框, 图 3-42 所示为梯形图表示的一个功能。

功能结构如图 3-43 所示。

功能定义由本地变量表和算法组成, 如图 3-44 所示。

功能各组成部分说明如下:

(1) 功能名称或指令名称

生成功能时, 在功能定义中分配功能名称或指令名称。

(2) 例程名称

功能没有例称名称。

(3) 算法

可以使用梯形图或 ST 为算法

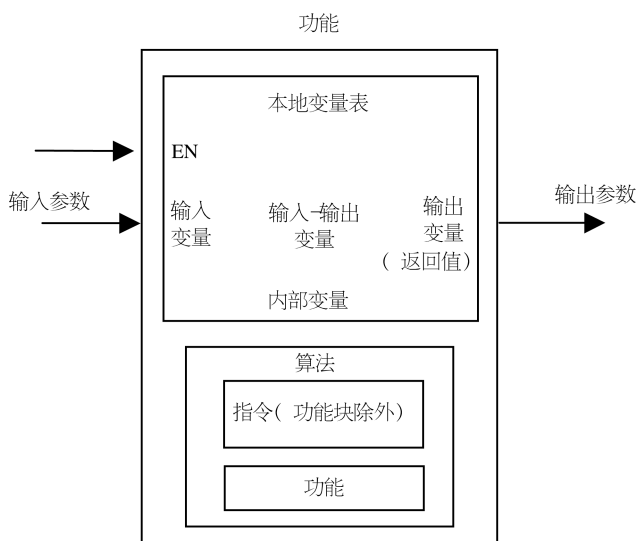


图 3-44 功能定义

编程。在功能的算法里，可以使用任何指令、功能或用户定义的功能。不能使用任何 FB 指令或用户定义的功能块。也不能使用微分指令，例如，R_TRIG 或 UP。

(4) 本地变量表

本地变量表用来定义输入变量、输出变量、输入-输出变量、内部变量和外部变量。

(5) 参数

输入变量的输入参数：当功能执行开始时，输入参数把值传给功能中的输入变量，输入参数可以是一个变量或常数。

输出变量的输出参数：当功能执行完毕后，输出参数接收功能中输出变量的值。

输入-输出变量共享的输入-输出参数：输入-输出参数的值在功能内改变，同一变量既用在输入也用在输出。

2. 功能变量的指定

功能变量的指定如图 3-45 所示。

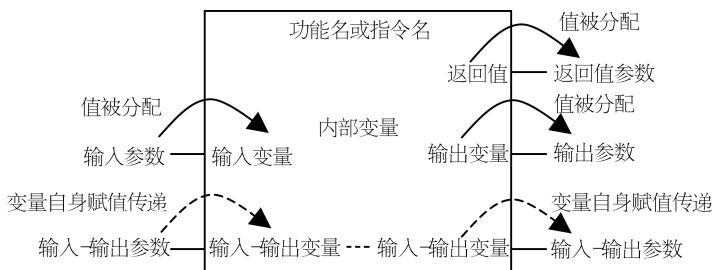


图 3-45 功能变量的指定

功能变量的说明见表 3-10。

表 3-10 功能变量的说明

变量	数量	规 范
输入变量	1 ~ 64	输入变量用作功能内部的输入参数。它们不能在功能内部被改变
		- 当功能执行时，输入变量被设置为输入的参数值 - 可以将输入参数指定为常量或者变量 - 省略输入参数 - 至少需要一个 BOOL 输入变量 - 与功能块不同，不可以指定检测变量更改为 TRUE 或 FALSE - 不能从功能外部访问输入变量的值 - 某些由 OMRON 提供的指令，它们的输入变量数目可变，但是用户创建的功能其输入变量数目不可变
输出变量	1 ~ 64	输出变量用作功能的输出参数
		- 输出参数在功能执行之后设置为输出变量的值 - 不可以为输出参数指定一个常量，必须指定一个变量 - 至少需要一个 BOOL 输出变量（包括 ENO 和返回值） - 可以省略输出参数的连接：如果省略一个输出参数，这个输出变量的值不分配给任何参数 - 不能从功能外部访问输出变量的值

(续)

变量	数量	规 范
输入-输出变量	0 ~ 64	输入-输出变量用来向功能输入和从功能输出。它们可以在功能内部被改变
		● 一个输入-输出参数的值传递给一个输入-输出变量，然后该输入-输出变量的值传递给输入-输出参数 ● 不可以为输入-输出参数指定一个常量，必须指定一个变量 ● 如果改变一个功能块的输入-输出变量的值，那么输入-输出参数的值也随之改变 ● 不能省略输入-输出参数 ● 不能从功能外部访问输入-输出变量的值
内部变量	无限制	内部变量在各功能内用作暂时存储
		● 功能执行完成后，内部变量的值不保持 ● 不能从功能外部访问内部变量的值
外部变量	无限制	外部变量用于访问全局变量
EN	1	这是用来执行功能的 BOOL 变量
		● 当 EN 为 TRUE 时，执行功能 ● 必须有一个 EN 变量 (这适用于用户定义功能和 FUN 指令)
ENO	0 或 1	通常，这是 BOOL 输出变量，正常结束时设置为 TRUE，出错时设置为 FALSE
		● 在一些功能指令中可以省略它
返回值	1	返回值是从功能返回到调用它的 POU 时的值。代表功能中的算法执行后的处理结果
		● 每一个功能必须有一个返回值 ● 返回值可以为各种基本数据类型，但不能是数组、结构体或共用体

3. 功能调用

功能既可用梯形图，也可用 ST 调用。

从 ST 调用功能的方法如下

```

return_value := function_name(input_variable_1:= input_parameter_1, ...,
                               input_variable_L:= input_parameter_L,
                               input_output_variable_1:= input_output_parameter_1, ...,
                               input_output_variable_M:= input_output_parameter_M,
                               output_variable_1:=> output_parameter_1, ...,
                               output_variable_N:=> output_parameter_N);
  
```

图 3-46 所示为用梯形图调用功能。

MyFUN：功能名称。

x0：设定是否调用功能。

A 和 B：功能的输入变量名称。

x1 和 x2：功能的输入参数。

C：功能的输出变量。

y1：功能返回值的存储位置。

y2：功能的输出参数。

用 ST 调用功能的方法有两种。

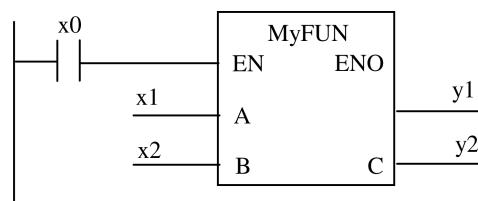


图 3-46 用梯形图调用功能

1) 给出功能的变量名程和调用功能的 POU 的参数名程

```
IF (x0 = TRUE) THEN
```

```
    y1 := MyFUN(A:= x1,B:= X2,C => y2);
```

```
END_IF
```

- 参数的顺序任意。

- 可以省略输入变量的名称和输入参数名称。如果省略这些名称，输入变量置为它们的初始值。

- 也可以省略 EN，如果省略，EN 分配的值为 TRUE。

2) 省略功能的变量名称，给出调用功能的 POU 的参数名称

```
IF (x0 = TRUE) THEN
```

```
    Y1:= MyFUN(x1, x2, y2);
```

```
END_IF
```

参数的顺序基于功能块定义。顺序与功能块本地变量定义相同，从顶部到底部。

ST 调用功能的各种方式见表 3-11。

表 3-11 ST 调用功能的各种方式

方法	表示方式		注释
方法 1	包括的参数	举例	可用
	给出所有的参数	FUN (A:= x1, B:= x2, C:=> y2);	
	给出多个参数	FUN (A:= x1, B:= x2);	
	给出一个参数	FUN (B:= x2);	
	不给出任何参数	FUN (A=);	不可用
方法 2	给出所有的参数	FUN (x1, x2, y2);	可用
	所有的参数都不给出	FUN ();	
	只给出第一个参数	FUN (x1);	不可用
	给出一个参数	FUN (x1, ,);	
	给出多个参数	FUN (x1, x2);	

图 3-47 所示为梯形图调用功能。

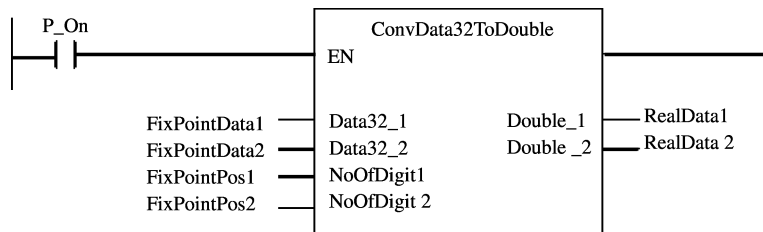


图 3-47 梯形图调用功能

用 ST 调用此功能，两种编写方法如下。

方法 1:

```
ConvData32ToDouble(Data32_1:= FixPointData1, Data32_2:= FixPointData2,
```



```
NoOfDigit_1:=FixPointPos1, NoOfDigit_2:=FixPointPos2,
Double_1=>RealData1, Double_2=>RealData2);
```

方法 2:

```
ConvData32ToDouble(FixPointData1, FixPointData2, FixPointPos1, FixPointPos2,
RealData1, RealData2);
```

4. 功能的创建

除了系统提供的标准功能外，用户可以创建自定义的功能，用户定义的功能同样可由程序、功能块、功能来调用。

下面创建一个用户定义功能，将角度转换为弧度，输入参数为角度数，输出参数为弧度数。

操作步骤如下：

1) 如图 3-48 所示，右击多视图浏览器中“编程-POUs”下的“功能”，从菜单中选择“添加-梯形图”，会默认生成一个叫功能 0 的功能，如图 3-49 所示。功能可以用梯形图或 ST 来编写，这里选用梯形图。

2) 双击功能 0，进入编程界面，如图 3-50 所示，上半部为变量表，有“内部”、“输入/输出”、“外部”和“返回值”4 个变量表。

在内部变量表添加一个 LREAL 变量 internals1，如图 3-51 所示。在输入/输出变量表添加两个 LREAL 变量 deg 和 rag，deg 为输入变量，rag 为输出变量，如图 3-52 所示。



图 3-48 添加功能



图 3-49 生成功能 0



图 3-50 功能 0 编程界面

内部	名称	数据类型	初始值	分配到	保持	常量	注释
输入/输出	internals1	LREAL			☐	☐	

图 3-51 内部变量表

内部	名称	输入/输出	数据类型	边沿	初始值	保持	常量
输入/输出	EN	输入	BOOL	无边沿		☐	☐
外部	deg	输入	LREAL	无边沿		☐	☐
返回值	rag	输出	LREAL	无边沿		☐	☐

图 3-52 输入/输出变量表

3) 在功能 0 的编程界面输入梯形图，如图 3-53 所示。

这样就创建了一个用户定义功能。

创建自定义功能后，就可以在编程时调用它。现在回到主程序下，对内部变量进行定义，如图 3-54 所示。

编写调用功能的梯形图程序，当输入到功能时，键入“功”，就发现刚才定义的功能 0 已经存在了，如图 3-55 所示。

完整的调用功能的梯形图如图 3-56 所示。

degtest 为角度，是功能的输入参数。radtest 为弧度，是功能的输出参数。

开始仿真测试，对 a 选择设置，如图 3-57 所示。

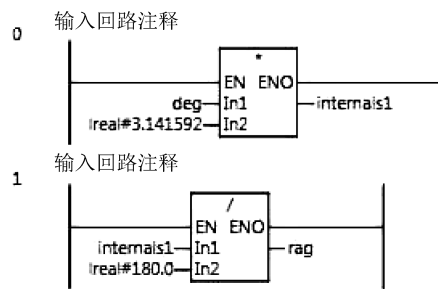


图 3-53 功能 0 编程

内部	名称	数据类型	初始值	分配到	保持	常量	注释
外部	a	BOOL			☐	☐	
	degtest	LREAL			☐	☐	
	radtest	LREAL			☐	☐	

图 3-54 程序中内部变量定义

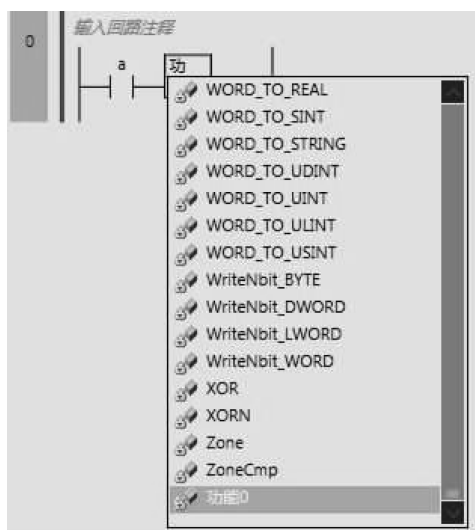


图 3-55 输入功能 0



图 3-56 调用功能的梯形图



图 3-57 仿真测试

如图 3-58 所示，对 degtest 输入 180，则 radtest 为 3.141592，结果正确。

输出窗口					
编译窗口					
监视选项卡1					
名称	在线值	修改	数据类型	分配到	显示格式
Program0.degtest	180	180	LREAL		Real
Program0.radtest	3.141592		LREAL		Real
输入名称...					

图 3-58 显示结果

3.3 变量

在 NJ 系列控制器中，变量用来与外部设备交换 I/O 信息，完成数据计算和其他过程。

变量存储与外部设备交换的 I/O 数据，或存储内部 POU 处理时的临时数据。换句话说，变量是一个数据的容器，有名字、数据类型和其他属性。

不需要为变量分配一个内存地址。然而，需要时可以分配一个特定的内存地址，例如，NJ 系列 CPU 单元自动为变量在内存区分配地址。

3.3.1 变量类型

变量一般分为下面三种类型。

1. 用户定义变量

用户定义一个用户变量的所有的属性。

2. 半用户定义变量

这些变量用来访问特定的设备和数据。有两种类型的半用户定义的变量：设备变量和凸轮数据变量。

3. 系统定义变量

在 NJ 系列 CPU 单元里，系统定义的变量预先提供。名称和所有的属性都由系统定义，它们有特定的功能。每一种功能模块都要使用系统定义的变量。

3.3.2 应用于 POU 的用户定义变量的类型

在 POU 中应用的用户定义的变量类型如下。

1. 本地变量

本地变量定义在一个 POU (程序、功能或功能块) 中，它只能在该 POU 中被读或写。如果 POU 是一个程序，本地变量就是内部变量。如果 POU 是一个功能块或功能，本地变量就是内部变量、输入变量、输出变量、输入-输出变量和外部变量的统称。

(1) 内部变量

内部变量只用在 POU 中，一个内部变量在 POU 的本地变量表中声明，不能从 POU 外访问内部变量的值。在不同的 POU 中可以声明同一名称的内部变量，这些变量的每一个都分配不同的内存地址。

(2) 输入变量

当 POU 被调用时，从调用的 POU 为输入变量分配输入参数的值。输入变量在 POU 的本地变量表中声明。

(3) 输出变量

在一个 POU 的处理完成之前，将返回到调用的 POU 中的输出参数分配给输出变量。输出变量在 POU 的本地变量表中声明。

(4) 输入-输出变量

当 POU 被调用时，从调用的 POU 为输入-输出变量分配输入-输出参数的值。如果改变一个 POU 中输入-输出变量的值，输入-输出参数的值随之改变。输入-输出变量在 POU 的本地变量表中声明。

2. 外部变量

外部变量用来访问 POU 外部的数据。可以从 POU 中访问全局变量。

3. 全局变量

全局变量在全局变量表中声明。

从系统配置和从单元配置中自动产生的设备变量，以及从轴设置表中产生的轴/轴组变量，自动登记为全局变量。

3.3.3 变量的属性

变量的属性见表 3-12。

表 3-12 变量的属性

属性	描 述	规 格	缺 省
变量名称	变量的名称用于识别变量		
数据类型	数据类型定义存储在变量中的数据格式		INT
AT 规格	如果想将 CJ 系列单元的一个特定地址作为一个变量处理，指定这个地址分配给该变量	● 未指定 ● 指定	未指定
保持	在下面的情况下指定是否要保持变量的值： ● 当电源断电后再上电 ● 当 CPU 单元切换到运行模式时 ● 当操作出现重大故障级的控制器错误时	● 保持：如果有电池，在左边的指定值被保持 ● 非保持：更改为初始值	非保持：复位为初始值
初始值	可以选择设置或不设置初始值 初始值设置：在下列情况下为变量指定值，不指定保持属性： ● 当电源接通时 ● 当操作模式改变 ● 当出现重大故障级控制器错误 如果初始值不设置，值将不保持	初始值 ● 有 ● 没有	取决于数据类型
常量	如果设置了常量的属性，可以在变量下载时设置它的初始值，但此后不能覆盖它的值	指定是一个常量或非常量	
网络公开	这个属性允许使用 CIP 通信和数据链接，从控制器外部读写变量	● 不公开 ● 公开 ● 输入 ● 输出	不公开
边缘	边缘属性允许对一个功能块的输入参数检测何时改变为 TRUE 或 FALSE。仅适用于 BOOL 输入变量	● 无 ● 更改为 TRUE ● 更改为 FALSE	无

注意，并非所有的变量都支持表 3-12 中的变量属性，这取决于变量的类型，例如，程序中的内部变量不支持网络公开属性和边缘属性，仅全局变量支持网络公开属性，边缘属性仅用于功能块的输入变量。

3.4 数据类型

IEC 61131-3 的数据类型分为基本数据类型和导出数据类型两类。数据类型与它在数据存储单元中所占用的数据长度有关。

定义数据类型可防止因对数据类型的不同设置而发生出错。数据类型的标准化是编程语言开放性的重要标志。

3.4.1 基本数据类型

基本数据类型是预先定义的标准化数据类型，有布尔型、位串、整数、实数、持续的时

间、一天中的时间、日期、日期和时刻、字符串等类型，总计 20 种，基本数据类型见表 3-13。

表 3-13 基本数据类型

分类	数据类型	数据长度	字节数	取值范围	标记
布尔型	BOOL	1 位		FALSE 或 TRUE	BOOL#1, BOOL#0, TRUE 或 FALSE
位串	BYTE	8 位	1 字节	BYTE#16#00 ~ FF	BYTE#2#01011010 BYTE#2#0101_1010 BYTE#16#5A 使用“_”作为分隔符
	WORD	16 位	2 字节	WORD#16#0000 ~ FFFF	
	DWORD	32 位	4 字节	DWORD#16#00000000 ~ FFFFFFFF	
	LWORD	64 位	8 字节	LWORD#16#0000000000000000 ~ FFFFFFFFFFFFFFFF	
整数	SINT	8 位	1 字节	SINT# -128 ~ +127	100 INT#2#00000000_01100100 INT#8#144 INT#10#100 INT#16#64 -100
	INT	16 位	2 字节	INT# -32768 ~ +32767	
	DINT	32 位	4 字节	DINT# -2147483648 ~ +2147483647	
	LINT	64 位	8 字节	LINT# -9223372036854775808 ~ +9223372036854775807	
	USINT	8 位	1 字节	USINT#0 ~ +255	
	UINT	16 位	2 字节	UINT#0 ~ +65535	
	UDINT	32 位	4 字节	UDINT#0 ~ +4294967295	
	ULINT	64 位	8 字节	ULINT#0 ~ +18446744073709551615	
实数	REAL	32 位	4 字节	REAL# -3.402823e+38 ~ -1.175494e-38 0 -1.175494e-38 ~ 3.402823e+38 +∞ / -∞	REAL#3.14 LREAL#3.14 3.14 -3.14 1.0E+6 1.234e4
	LREAL	64 位	8 字节	LREAL# -1.79769313486231e+308 ~ -2.22507385850720e-308 0 2.22507385850720e-308 ~ 1.79769313486231e+308 +∞ / -∞	
持续时间	TIME	64 位	8 字节	T# -9223372036854.775808ms (T# -106751d_23h_47m_16s_854.775808ms) ~ T# +9223372036854.775807ms (T# +106751d_23h_47m_16s_854.775807ms)	T#12d3h3s T#3s56ms TIME#6d_10m TIME#16d_5h_3m_4s T#12d3.5h T#10.12s T#61m5s (相当于 T#1h1m5s) TIME#25h_3m

(续)

分类	数据类型	数据长度	字节数	取值范围	标记
一天中的时间	TIME_OF_DAY	64 位	8 字节	TOD#00:00:00.000000000 ~ TOD#23:59:59.999999999 (00:00:00.000000000 ~ 23:59:59.999999999)	在字符串的开始加“TIME_OF_DAY#”,“time_of_date#”,“TOD#”,或“tod#”,用 hh-mm-ss 格式表示一天中的时间 例如: Tod#12:16:28.12
日期	DATE	64 位	8 字节	D#1970-01-01~D#2106-02-06 (January 1, 1970 ~ February 6,2106)	在字符串的开始加“DATE#”,“date#”,“D#”或“d#”并用 yyyy-mm-dd 格式表示日期。 例如: d#1994-09-23
日期和时刻	DATE_AND_TIME	64 位	8 字节	DT#1970-01-01-00:00:00.000000000 ~ DT#2106-02-06-23:59:59.999999999 (January 1, 1970 00:00:00.000000000 ~ July 21, Ferbruary 6, 2106, 23:59.999999999 seconds.)	在字符串的开始加“DT#”或“dt#”,用 yyyy-mm-dd-hh:mm:ss 格式表示日期和时刻。 例如: dt#1994-09-23-12:16:28:12
字符串	STRING	(单字节字符数目+1)×8 位	1 字节	字符代码是 UTF-8。0~1986 字节(0~1985 单字节字母数字型字符)。 一个 NULL 字符加到字符串的末尾,因此,字节数为单字节字符数目+1。 在 Sysmac Studio 中设置字符串变量的字节数目,默认值为 256	用单字节的单引号(')将字符串括起来。 例如: 'OMRON','PLC'

1. 布尔型 (BOOL)

布尔型为位数据,取值为 TRUE 或 FALSE,也可表示为 BOOL#1 或 BOOL#0。布尔型数据表达为 {data_type} # {numeric_value}。

例如,布尔型变量赋值时

正确表达: BOOL_variable:=TRUE;或 BOOL_variable:= BOOL#1;

错误表达: BOOL_variable:=1;

2. 位串 (bit string)

位串的数据类型为 BYTE、WORD、DWORD 和 LWORD,基可选 2、8、16。位串数据的表达为 {data_type}#base#{numeric_value}。

例如,位串变量赋值时

正确表达: bit_string_variable:= WORD#16#3;

错误表达: bit_string_variable:=3;

不能直接比较两个位串数据的大小,可以用数据转换指令将它们转换为整数类型的数

据, 再进行比较。

3. 整数 (Integer)

整数的数据类型有 SINT、INT、DINT、LINT、USINT、UINT、UDINT、ULINT。基可选 2、8、10、16, 即一个整数可定义为二进制、八进制、十进制、十六进制数。整数数据的表达为 {data_type} #base# {numeric_value}。十进制数, 为表示负数, 可在数值前添加“-”, 如 -15。若省略“基”, 写为 {data_type} # {numeric_value}, 则默认基为 10 (十进制数)。数据类型不允许省略。

例如, 为单字整数变量赋值 -1 时

正确表达: int_variable:=INT#2#10000000_00000001;

int_variable:=INT#8#100001;

int_variable:=INT#10#-1; 或 int_variable:=INT#-1;

int_variable:=INT#16#8001;

错误表达: int_variable:= -1;

在数值中单一的下划线字符“_”是没有意义的, 只增加可读性, 例如, 2#1111_1111 表示二进制 11111111。

不能对整数型的数据直接进行逻辑运算, 可以用数据转换指令转换为位串型数据, 再进行逻辑处理。

4. 实数 (Real number)

实数的数据类型有 REAL、LREAL。基为 10, 表示十进制数。实数数据的表达为 {data_type} #base# {numeric_value}, 可省略“基”, 写为 {data_type} # {numeric_value}。若省略数据类型, 写为 {numeric_value}, 则默认数据类型为 LREAL。

例如, 实数变量赋值 3.14 时

正确表达: real_variable:=REAL#3.14;

错误表达: int_variable:=3.14;

实数也称为浮点数, 在计算机中用二进制数来表示, 通常为 32 位或 64 位, 分别叫作 REAL 或 LREAL。

在数据的位数有限的情况下, 用定点数表示的数的范围小。当用浮点数表示一个实数时, 小数点的位置可变, 随着小数点位置的改变, 指数的数值也随之改变。因此, 在有效数字的位数有限的情况下, 不仅浮点数的范围可以比定点数的范围大得多, 而且也可以保持数据的有效精度。

对浮点数的表示法, 美国电气与电子工程师协会 (IEEE) 制定的标准已被众多计算机制造商采用。下面以浮点数的 IEEE754 格式为例, 介绍浮点数的表示法。

(1) 浮点数的 IEEE754 格式

浮点数的 IEEE754 标准格式可表示为:

REAL 数值 = $(-1)^s 2^{e-127} (1.f)$

LREAL 数值 = $(-1)^s 2^{e-1023} (1.f)$

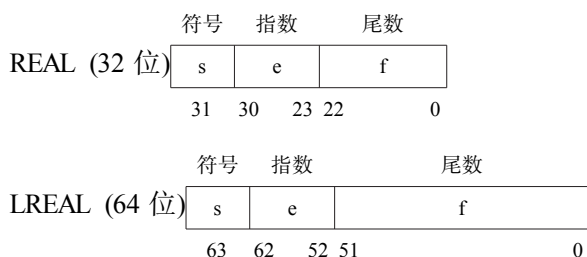
式中各符号的意义是:

s: 符号

e: 指数

f: 尾数 (小数点后面的有效数字)

浮点数的格式如下:



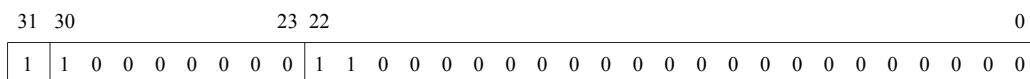
以 REAL 数为例, 介绍各部分的含义:

符号 s 占 1 位, 取值为 0 表示正数, 1 表示负数。

指数 e 占 8 位, 取值为 0 ~ 255, 实际指数是 e 减去 127 的差, 结果为 -127 ~ 128。

尾数 f 占 23 位, 二进制浮点数的尾数部分满足 $2.0 > 1.f \geq 1.0$ 。

例如, 某 REAL 数表示如下:



符号 s 为 1, 表示该数据是负数;

指数 e 为 $2^7 = 128$, 实际指数为 $128 - 127 = 1$;

尾数 f 为 $(2^{22} + 2^{21}) \times 2^{-23} = 0.75$, 实际尾数为 $1 + 0.75 = 1.75$;

该 REAL 数据值为 $-2^1 \times 1.75 = -3.5$ 。

(2) 用浮点数表示的数据范围

REAL 和 LREAL 数据的取值范围见表 3-14, 非常接近于 0 的值不能表示。

表 3-14 REAL 和 LREAL 数据的取值范围

数据类型	$-\infty$	负数	0	正数	$+\infty$
REAL	$-\infty$	$-3.402823\text{e}+38 \sim -1.175494\text{e}-38$	0	$+1.175494\text{e}-38 \sim +3.402823\text{e}+38$	$+\infty$
LREAL	$-\infty$	$-1.79769313486231\text{e}+308 \sim -2.22507385850720\text{e}-308$	0	$+2.22507385850720\text{e}-308 \sim +1.79769313486231\text{e}+308$	$+\infty$

浮点数的数据范围如图 3-59 所示。

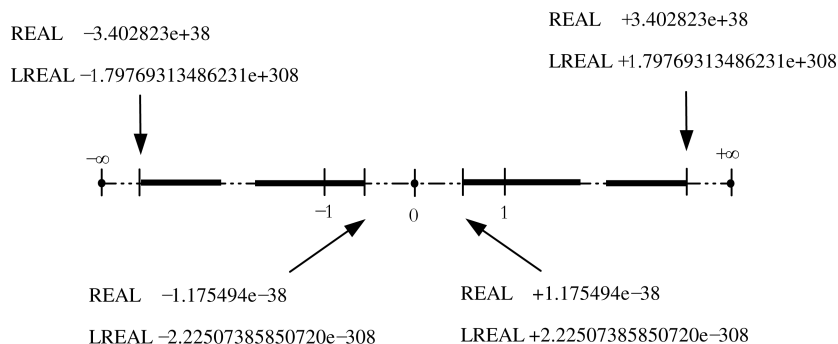


图 3-59 浮点数的数据范围

浮点数表示的特殊数据见表 3-15。

表 3-15 浮点数表示的特殊数据

数据类型名称	特殊值	符号 s	指数 e	尾数 f
REAL	$+\infty$	0	255	0
	$-\infty$	1	255	0
	+0	0	0	0
	-0	1	0	0
	NaN	...	255	不为 0
LREAL	$+\infty$	0	2047	0
	$-\infty$	1	2047	0
	+0	0	0	0
	-0	1	0	0
	NaN	...	2047	不为 0

非标准数据的浮点数表示为:

$$\text{LREAL} \quad (-1)^s \times 2^{-1022} \times (\text{尾数} \times 2^{-52})$$

以 REAL 数为例, 介绍各部分的含义:

- 符号 s 为 0 表示正数, 1 表示负数。
- 指数 e 为 0, 实际指数为 -126 。
- 尾数 f 用 $1 \sim (2^{23} - 1)$ 表示, 在实际尾数 $0.f$ 中, 设定位 23 为 0 (这也是一种规定), 二进制数的小数点要紧跟在它的后面。

例如，某非标准的 REAL 数如下：

[illegible]

尾数 f 为 $(2^{22} + 2^{21}) \times 2^{-23} = 0.75$, 实际尾数为 $0 + 0.75 = 0.75$;

该 REAL 数据值为 $-2^{-126} \times 0.75$, 是一个绝对值极小的数值。

(5) 数据处理

浮点数表达的只是一个近似值。因此，浮点数和它的真值有差别。浮点数有效数字的位数是有限的。因此，当完成浮点数运算时要采取以下措施。

1) 四舍五入

如果实际值超过尾数的有效数字，按照下列规则四舍五入处理：

- 结果是浮点数所能表示的最接近的值。
- 当浮点数运算结果中的数字个数超过内部处理表达式中的有效数字时，若结果近似于两个内部浮点表达式中的一个，则使用较接近的值。若结果介于两个内部浮点表达式的中间，可四舍五入，以便使尾数的最后一个数字为 0。

2) 上溢出、下溢出

上溢出将根据结果的符号以 $+\infty$ 或 $-\infty$ 输出。下溢出将根据结果的符号以 $+0$ 或 -0 输出。如果浮点数转换为整数时出现上溢出，那么结果值可能不正确。

非法计算将导致出现 NaN (不是一个数)。

当实际值的绝对值大于浮点数所能表示的最大值时，发生上溢。当实际值的绝对值小于浮点数所能表示的最小值时，发生下溢。

- 如果发生上溢且实际值为正数，则计算的结果为 $+\infty$ 。如果发生下溢且实际值为负数，则计算的结果为 $-\infty$ 。

- 如果发生下溢且实际值为正数，则计算的结果为 $+0$ 。如果发生下溢且实际值为负数，则计算的结果为 -0 。

(6) 特殊数据计算

特殊数据包括 0、 ∞ 、NaN，处理时注意以下事项：

按照下面的原则完成涉及特殊数据 (也就是， $+\infty$ ， $-\infty$ ， $+0$ ， -0 和 NaN) 的计算

- $+\infty$ 和 $-\infty$ 相加的结果为 NaN。
- 同样符号的两个 ∞ 值相减的结果为 NaN。
- $+0$ 或 -0 与 ∞ 相乘的结果为 NaN。
- $+0/+0$ 、 $-0/-0$ 或 ∞/∞ 的结果为 NaN。
- $+0$ 和 -0 的和为 $+0$ 。
- $+0$ 减 $+0$ 或 -0 减 -0 的结果为 $+0$ 。相同符号的 0 之差为 $+0$ 。
- 任何涉及 NaN 数据的四则运算的结果为 NaN。如果有一个操作数为 NaN，结果将为 NaN。
- 比较指令(例如 Cmp 指令)认为 $+0$ 和 -0 一样大。 $+0$ 和 -0 进行比较时按相等处理。
- 如果将 NaN 与其他数据比较，结果永远不相等。

5. 持续时间 (Duration)

持续时间用于表达一个事件经过的时间。

持续时间用 TIME#{day}d{hour}h{minutes}m{seconds}s{milliseconds}ms 表示，式中 TIME 可简化为 T。

持续时间前半部分为标识符 TIME (或 T) 及分界符#，后半部分是用天、小时、分、秒和毫秒或其任意组合表示的持续时间，如 TIME#3d4h5m6s。

持续时间标识符 TIME 或 T，因字母的大小写具有相同意义，因此，持续时间标识符也可表示为 time 或 t。其中，TIME 或 time 称为长前缀，T 或 t 称为短前缀。

表示时间单位天、小时、分、秒和毫秒等字母的大小写具有相同意义。因此，T#12.3S 与 T#12.3s 等价。

由于下划线没有意义，因此持续时间数值可用下划线分隔。例如，T#1d_2h_3m 表示持续时间为 1 天 2 小时 3 分，它与 T#1d2h3m 具有相同的含义。

持续时间的单位有天 (d)、小时 (h)、分 (m)、秒 (s)、毫秒 (ms)。可以用小数表示持续时间，如 T#12.3s、T#4.5d。可以任意指定时间表示的顺序，如 T#1h_2d 也是合法的。

持续时间的数值允许有负值，如 T#-23.4h14.5m，但不允许表示为 -T#23.4h14.5m。

持续时间具有超载属性,即允许数据超过有效的范围。例如,T#25h80m 中 25 小时已超过 1 天 24 小时的有效时间范围。同样,80 分钟也超出了分钟的有效时间范围,但这样的表示是允许的,编程系统会自动将它转换为不超过有效时间范围的数据,即它与 T#1d2h20m 是等价的。

6. 一天中的时间 (Time of Day)

一天中的时间用于表示在一天中的时间,也称为时刻,例如,当前时间等。

一天中的时间用时刻标识符、分界符#和时刻表示。表达式为 TIME_OF_DAY#{hour}:{minutes}:{seconds},或 TOD#{hour}:{minutes}:{seconds}。

TIME_OF_DAY 或 time_of_day 是时刻标识符的长前缀,TOD 或 tod 是时刻标识符的短前缀。例如,TIME_OF_DAY#12:34:56 和 TOD#12:34:56 是等效的。

时刻文字中,时刻的分界符用冒号,不用下划线。例如,TOD#12:34:56.25 表示在 12 时 34 分 56.25 秒。TOD#12_34_56.25 是错误的时刻表示。

时刻单位是时、分、秒,没有毫秒单位,但可用小数表示时刻。例如 TOD#12:34:56.25 中 56.25 表示 56.25 秒。由于时刻是一天中的时间,因此,不需要天的时间单位。

时、分和秒的时间数据不能缺省,当最小时间单位没有时,可省略,例如,TOD#12 表示 12 时。

7. 日期 (Date)

日期用于表示当天是某年某月某日。日期用日期标识符、分界符#和日期表示,表达式为 DATE#{year} - {mouth} - {day}或 D#{year} - {mouth} - {day}。

年、月、日的开始可以加一个或多个 0,例如 2005 年 5 月 10 日,可表示为 D#2005 - 05 - 10 或 D#2005 - 5 - 10。

日期标识符的长前缀用 DATE 或 date 表示,短前缀用 D 或 d 表示。因此,D#2008 - 08 - 08 与 DATE#2008 - 08 - 08 等价。

日期文字中,日期的分界用连字符“-”,不用冒号“:”或下划线“_”。此外,由于表示日期,因此,年、月和日的数据不能缺省。例如,不能表示为 D#08 - 08。

8. 日期和时刻 (Date and Time)

日期和时刻用于表示某年某月某日某时某分某秒某毫秒的时间。表达式为 DATE_AND_TIME#{year} - {mouth} - {day} - {hour}:{minutes}:{seconds},也可以简写为 DT#{year} - {mouth} - {day} - {hour}:{minutes}:{seconds}。

由于它既有日期,又有时刻,因此,需注意时间之间分隔符的不同。

1) 年、月和日的分界符,与日期分界符相同,用连字号“-”分界。

2) 时、分、秒和毫秒的分界符,与时刻分界符相同,用冒号“:”分界。

3) 日期和时刻之间用连字号“-”分界。例如,DT#2008-05-12-14:28:04 表示 2008 年 5 月 12 日 14 时 28 分 04 秒。

需要指出的是,以上介绍的 4 种时间类型的数据表达式中的标识符,其包含的字母的大小写意义相同。例如,Dt#2005-05-19-21:05:8.25 表示日期和时刻,toD#15:36:22.25 表示一天中的时间等。

9. 字符串 (Text Strings)

字符串 (Text Strings) 由单字节字符组成。

单字节字符串由一系列通用的字符表示,或由 '\$ '、'\$ "、'\$ 与十六进制数字等组成。它们用单引号在其前后标识,如'ABC'、"、'\$ "、'\$ "'、'\$ C4'等。当美元符号与两个十六进制数组成3字节字符串时,应将该字符串的两个十六进制数组成的数对应解释为8位字符码的十六进制数。例如,'\$ 0A'表示 LF 字符的长度为1的串,它与字符'\$ L'等效。

用美元符号和单或双引号等组合形成一些特殊字符,具有特定含义,见表3-16。

表 3-16 特殊字符

组 合	名 称	含 义
\$ \$	单字节美元符号	单字节美元符号(\$;字符代码 0x24)
\$ '	单字节单引号	单字节单引号(' ;字符代码 0x27)
\$ L 或 \$ l	换行 LF	将光标移到下一行 LF 控制字符(换行;字符代码 0x0A)
\$ N 或 \$ n	纵向制表	将光标移到下一行 NL 控制字符(纵向制表;字符代码 0x0B)
\$ P 或 \$ p	换页 FF	将光标移到下一页 FF 控制字符(换页;字符代码 0x0C)
\$ R 或 \$ r	回车 CR	将光标移到下一行的开始 CR 控制字符(回车;字符代码 0x0D)
\$ T 或 \$ t	水平制表 Tab	表示一个 Tab Tab 字符(字符代码 0x09)
\$ "	单字节双引号	输出单字节双引号(字符代码 0x22)
\$ (2 位十六进制数)	直接字符代码规格	指定字符代码为圆括号内的两位十六进制数,取值范围 00 ~ FF。例如,'\$ 35'与'5 '等价。

字符串可以是空串,如",单字节字符串用单引号开始,可表示为 STRING#'YES'或'YES'。

10. 数据类型转换

程序根据数据类型进行适当的操作。例如,赋值表达式的左边和右边通常使用相同的数据类型。然而,在某些情况下,根据需要为变量赋值不同数据类型的数据,例如:

```
Var3 :=Var1;
```

Var1 是数据类型 INT 的变量

Var3 是数据类型 REAL 的变量

为了将 Var1 的值赋给 Var3,数据必须转换。这种转换的类型称为数据类型转换,或简称为“类型转换”。

数据类型转换通过执行专用的数据类型转换指令来完成,或在赋值和指令中自动转换。

3.4.2 导出数据类型

NJ 系列控制器使用的导出数据类型有:结构体、联合体、枚举体。

1. 结构体 (Structure)

结构体是一种导出数据类型,它将相同或不同类型的数据组合成一个有机的整体,以便于引用。这些组合在一个整体中的数据是互相联系的。例如,描述一个盒子大小的指标有三

项：长度 (Length)、宽度 (Width)、高度 (Height)，如果将 Length、Width、Height 分别定义为互相独立的简单变量，是难以反映它们之间的内在联系的。应当将它们组织成一个组合项，在这个组合项中包含若干个相同或不同类型的数据项。根据以上描述，定义一个“BOX”的结构体，它有 Length、Width、Height 3 个成员。设定一个结构体变量 Box1，如图 3-60 所示，可以这样访问 Box1 的各个成员：

Box1. Length

Box1. Width

Box1. Height

生成结构体的步骤如下。

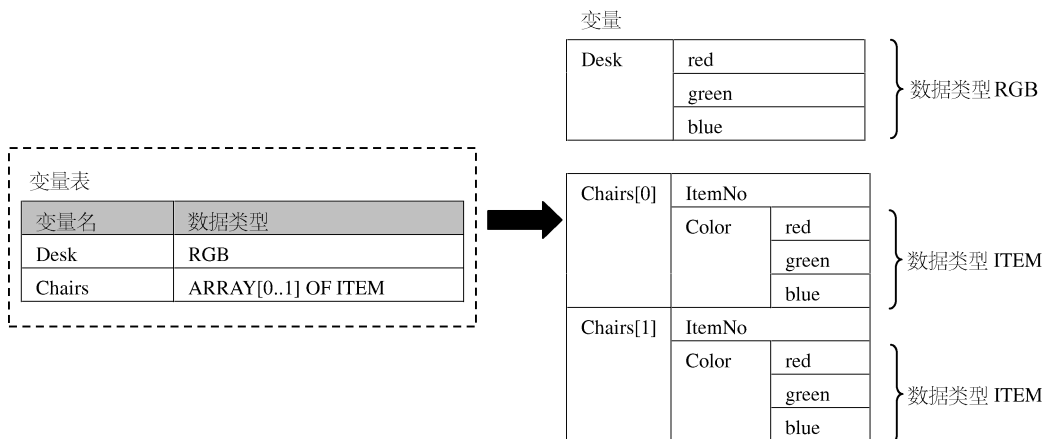
1) 在数据类型表中生成结构体数据类型

指定数据类型名称、成员和数据类型。

数据类型表		
名称	成员	数据类型
RGB	red	INT
	green	INT
	blue	INT
ITEM	ItemNo	INT
	Color	RGB

可以看出定义两个结构体：RGB 和 ITEM，ITEM 中的成员 Color 是一个结构体变量，类型为已定义的结构体 RGB。

2) 指定变量的名称和结构体数据类型，在变量表中登记变量



以上登记了变量 Desk、数组 ARRAY[0..1], Desk 的数据类型为 RGB, ARRAY[0..1] 的数据类型为 ITEM。

如果成员本身又属一个结构体类型，则要用若干个成员运算符，一级一级地找到最低的一级成员。只能对最低的成员进行赋值或存取及运算。例如，对于上面定义的结构体数组元素 Chairs[1]，这样访问其成员

Chairs[1]. Color. red

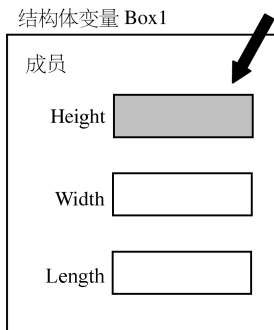


图 3-60 结构体变量 Box1

Chairs[1].Color.green

Chairs[1].Color.blue

结构体规格见表 3-17。

表 3-17 结构体规格

项 目	规 格		
结构体名	名称不区分大小写。禁用的字符和字符长度的限制与变量名相同		
成员数据类型	分类	数据类型	用法
	基本数据类型	Boolean, bit string, integer, real, duration, date, time of day, date and time, 或 text string 数据	支持
		Boolean, bit string, integer, real, duration, date, time of day, date and time, 或 text string 数据的数组	支持
	导出数据类型	结构体, 共用体和枚举体 注意: 不允许递归和循环 (当程序运行时会发生错误检查)	支持
		结构体, 共用体和枚举体的数组	支持
	POU 例程		不支持
成员属性	成员名称注释		
成员数量	1 ~ 2048		
结构体嵌套深度	最多 8 级 (但成员名必须是 511 字节或更少, 其中包括变量名)		
一个结构变量的最大尺寸	没有限制		

可以设置数组, 其元素为结构体。也可以设置一个结构体, 其成员为数组。

某些指令可以将结构体变量作为其参数。例如, 结构体变量的成员和结构体变量都可以作为 MOVE 指令的参数。

以下是未使用结构体和使用结构体处理数据的对比:

不用结构体

变量表	
变量名	数据类型
Color1_red	UINT
Color1_green	UINT
Color1_blue	UINT
Color2_red	UINT
Color2_green	UINT
Color2_blue	UINT

```
Color1_green:=255;
Color1_red:=128;
Color2_red:= Color1_red;
Color2_green:= Color1_green;
Color2_blue:= Color1_blue;
```

使用结构体

数据类型表		
名称	成员	数据类型
RGB	red	UINT
	green	UINT
	blue	UINT
变量表		
变量名	数据类型	
Color1	RGB	
Color2	RGB	

```
Color1.green:=255;
Color1.red:=128;
Color2:= Color1
```

成员变量可以像普通变量一样进行各种运算，根据其类型决定可以进行的运算。

一个结构体变量可以直接赋值给另一个具有相同结构的结构体变量。

2. 共用体 (Unions)

共用体规格见表 3-18。

表 3-18 共用体规格

项 目	规 格		
成员指定的数据类型	分类	数据类型	用法
	基本数据类型	BOOL 和位串	支持
		BOOL 和位串的数组	支持
	导出数据类型	结构体、共用体和枚举体的数组	不支持
	POU 例程		不支持
成员数量	最多 4 个		
设置初始值	不支持，一直为 0		

共用体使用时注意：共用体的初始值总是为 0；不能传送共用体；不能对 POU 指定共用体作为参数。

3. 枚举体 (Enumerations)

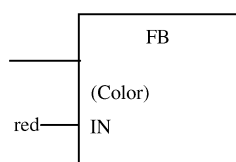
如果一个变量只有几种可能的值，可以定义为枚举类型。所谓“枚举”是指将变量的值一一列举出来。变量的值只限于列举出来的值的范围内。

生成枚举体 (ENUM) 的步骤如下。

1) 在枚举体表中生成枚举体数据类型。

为枚举体设置枚举元素和它们的值。

2) 指定变量的名称和枚举体数据类型，在变量表中登记变量。



枚举体表

数据类型	
Color	ENUM
枚举元素	值
red	0
yellow	1
green	2

变量表

变量名程	数据类型
DiscColor	Color

这里，Color 定义为枚举体数据类型。设置 3 个枚举元素：red、yellow 和 green。与这些枚举元素相关联的数值为：red = 0、yellow = 1、green = 2。变量 DiscColor 登记为 Color 类型。现在，对变量 DiscColor 可以在三种不同的元素 red、yellow 和 green 中选择，相应地又可以为变量 DiscColor 写出对应的值 0、1、2。

枚举体使用时注意以下问题：

1) 枚举体元素名称由单字节字母数字型字符组成，不区分大小写。禁用字符的规定与

变量名相同。如果不止一次指定相同的枚举体元素，会发生编译错误。如果指定的枚举体元素与用户程序中的变量重名，或者指定的枚举体元素已存在于另一个枚举体，也会出现编译错误。

2) 枚举体元素取值的有效范围：-2147483648 ~ 2147483647 之间的整数。值不必连续。如果指定相同的值超过一次，编译会出错。不能比较枚举体变量长度的大小，只能测试判断枚举体元素是否相同。

3) 枚举体元素数目：1 ~ 2048 个。

3.4.3 数组

数组是按照顺序排列的一组数据的集合。数组中的每一个元素都属于同一个数据类型，可以为整型、字符型、实型等基本数据类型，也可以为结构体、共用体、枚举体等导出数据类型。

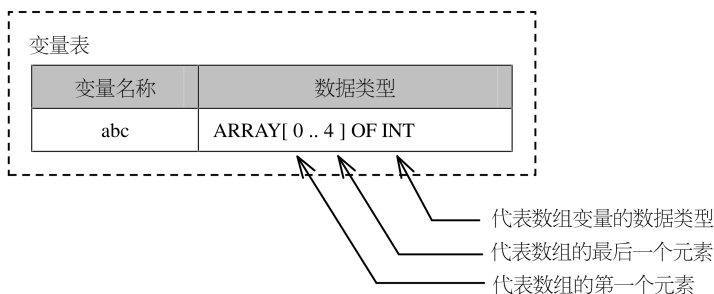
数组名定名的规则和变量名相同，遵循标识符定名规则。

数组中的元素用一个统一的数组名和下标来唯一地确定。下标放在数组名后，使用单字节方括号括起来，可以为整数常量或变量，在 ST 中可用整数表达式表示。

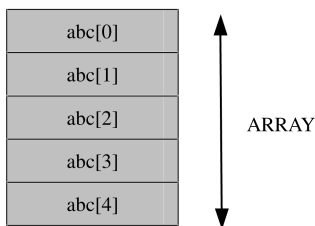
生成数组的步骤如下。

1) 在变量表的数据类型列输入“A”，从可选的数据类型名称列表中选择 ARRAY[?..?] OF ?。

2) 在“[?..?]”部分，对应左边的问号，输入数组的第一个元素的序号，对应右边的问号，输入数组的最后一个元素的序号。接下来，在“OF ?”部分，对应问号输入数据类型，并登记变量。



这里，定义 INT 数组变量 abc，共有 5 个元素，如下所示。



数组变量规格见表 3-19。

除了一维数组，还可以以同样的方式定义二维、三维数组，数组元素的表示用与维数相同的同样数目的下标。数组最多三维。

表 3-19 数组变量规格

项 目	规 格			
数组变量元素的最大数目	65535			
元素序号	0 ~ 65535 数组的第一个元素的序号不一定非要是 0			
下标	常量：整型值介于 0 ~ 65535 之间 变量：			
	分 类	数据类型		
	基本数据类型	整型	SINT, INT, DINT, USINT, UINT 或 UDINT	支持
			LINT 或 ULINT	不支持
			Boolean, bit string, real, duration, date, time of day, date and time 或 text string data	
	导出数据类型	Structures, unions 和 enumerations		不支持
	POU 例程			不支持
	算术表达式：算术表达式只在 ST 中指定 例如：y:= x[a + b]；			

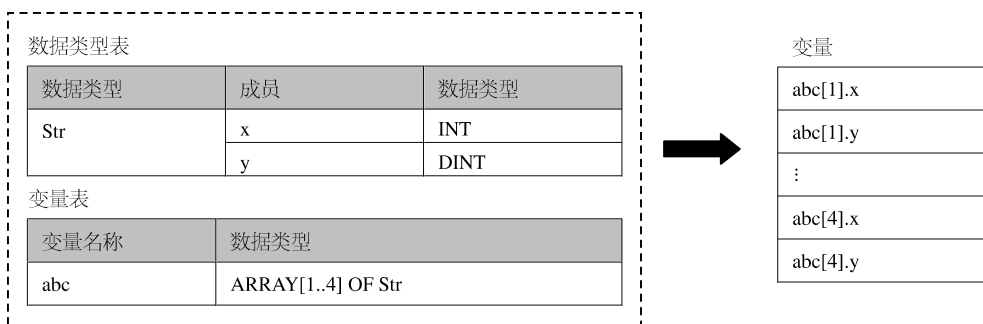
二维数组如下所示：

变量表	
变量名称	数据类型
abc	ARRAY[0..5, 0..3] OF INT

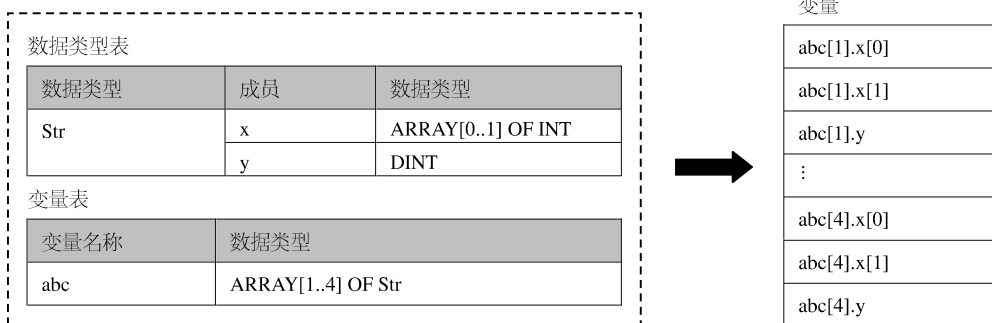
三维数组如下所示：

变量表	
变量名称	数据类型
ITEM	ARRAY[0..1, 0..2, 0..3] OF INT

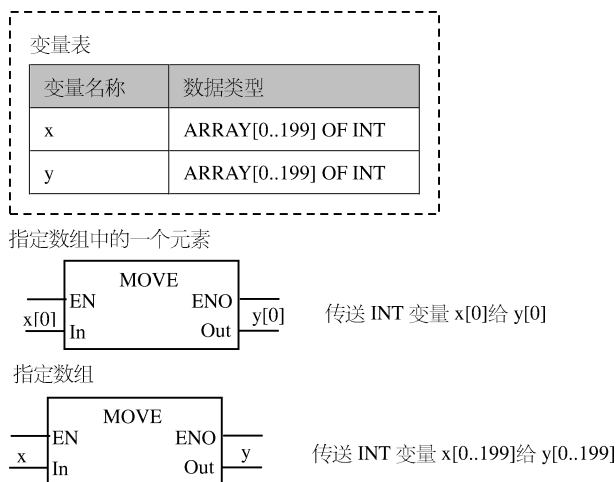
可以设置数组，其元素为结构体。也可以设置结构体，其成员为数组。
元素为结构体的数组如下：



成员为数组的结构体如下：



有些指令将数组变量作为参数传递，为此，仅指定数组变量的名称作为输入参数。例如，可用 MOVE 指令传送单个数组元素，也可传送一个数组。



3.5 编程语言

编程语言用来表达 POU (程序、功能或功能块) 中的算法。NJ 系列控制器使用两种不同的编程语言：梯形图 (LD) 和 ST (结构文本)。

3.5.1 梯形图

1. 梯形图结构

梯形图是一种图形编程语言，如图 3-62 所示，一个梯形图由左母线、右母线、连接线、梯形图的结构元素 (例如，输入和输出)、功能和功能块等组成。算法由连接在一起的多个回路实现，回路也称为梯级。一个回路是其所有配置在左母线和右母线之间的元素的一个连接。一个程序回路由逻辑块组成，逻辑块开始通常为 LD/ LD NOT 指令，表示逻辑的开始。

1) 母线：一个梯形图的左侧和右侧的垂直线称为母线。左母线是起始母线，每一逻辑行必须从起始母线开始画起，结束于右母线。

2) 连接线：连接母线和配置元素的水平直线称为连接线。

3) 输入：输入放在连接线上，并进行相应操作，其结果作为后续执行的依据。

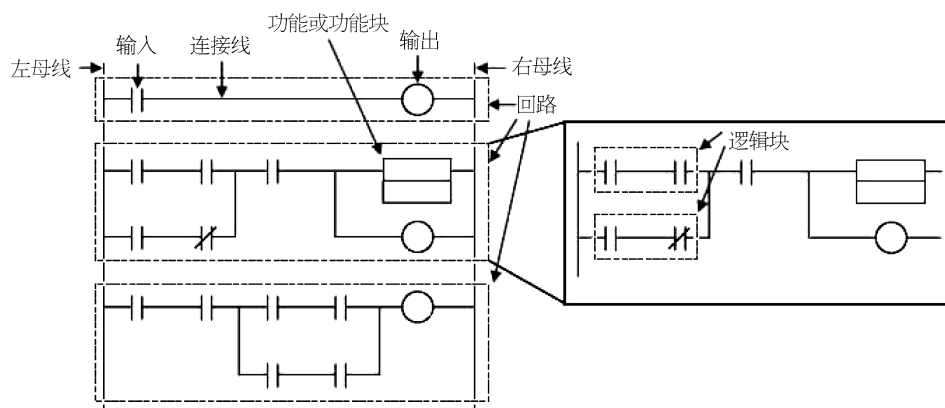


图 3-62 梯形图

4) 输出：输出放置在连接线上，根据输出之前的逻辑结果，给一个变量写入 TRUE 或 FALSE 值。

5) 功能和功能块：功能和功能块放置在连接线上，进行相应操作。

梯形图按照从上到下、从左到右的顺序执行，碰到 END 或 RETURN 指令，执行过程结束。

2. 在梯形图中连接功能和功能块

(1) 基本连接

图 3-63 所示为梯形图中功能和功能块的基本连接方法。BOOL 型输入变量 IN1 通过 x1 连接到左母线，BOOL 型输出变量 OUT1 通过 y1 连接到右母线。其他输入变量 IN2 不连接到左母线，x2 为其指定的参数输入；其他输出变量 OUT2 不连接到右母线，y2 为其指定的参数输出。

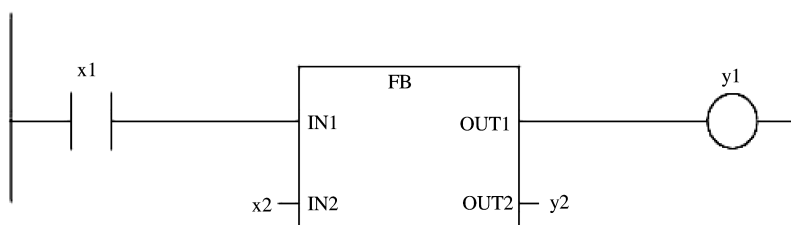


图 3-63 功能和功能块的基本连接

(2) BOOL 变量的数目

功能或功能块的输入和输出至少需要一个 BOOL 变量，例如 EN 和 ENO，如图 3-64 所示。

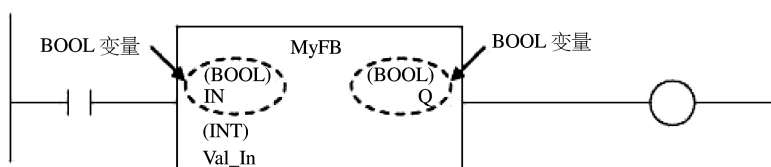


图 3-64 至少需要一个 BOOL 变量

(3) 基于 BOOL 变量位置的连接

顶端的 BOOL 变量连接到左母线、右母线，如图 3-65 所示。

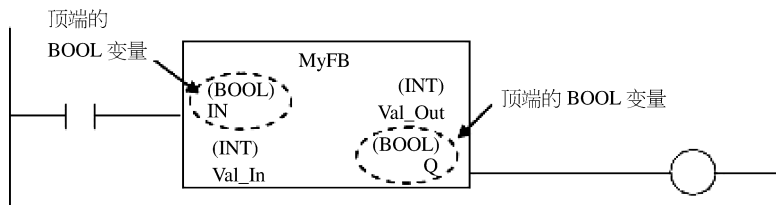


图 3-65 顶端的 BOOL 变量连接

不在顶端的 BOOL 变量指定 BOOL 参数输入或参数输出，如图 3-66 所示。

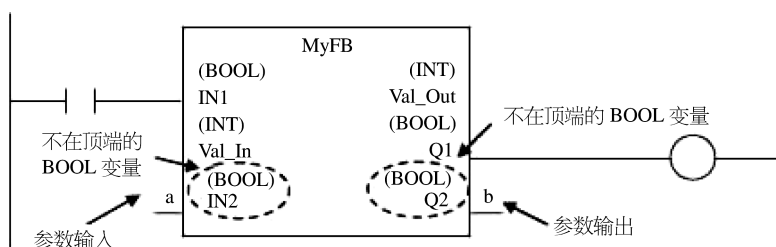


图 3-66 不在顶端的 BOOL 变量指定参数

左母线或右母线不能连接多个 BOOL 变量，如图 3-67 所示。

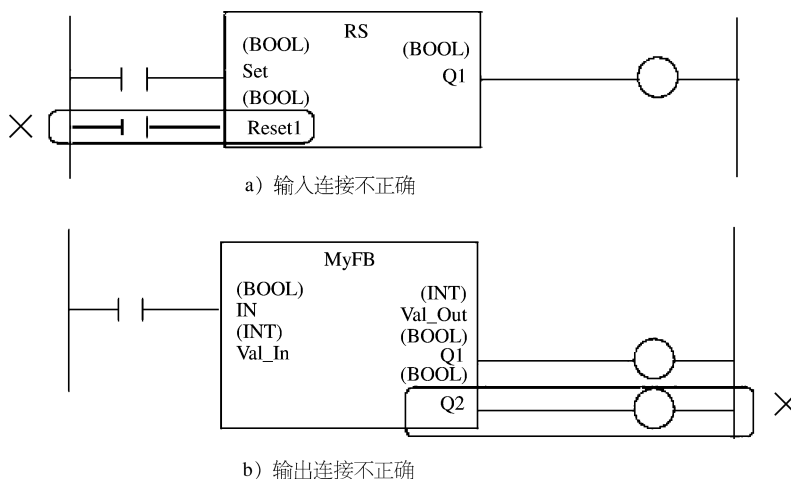


图 3-67 错误连接

可以将功能或功能块直接连接到右母线，这里，OUT 指令不是必需的，如图 3-68 所示。

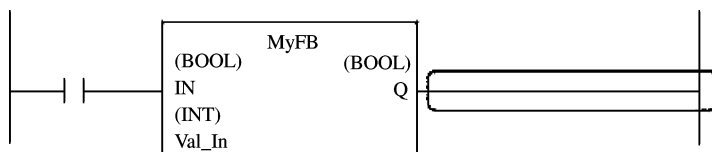


图 3-68 与右母线直接连接

可以将功能或功能块直接连接到左母线，这里，LD 指令不是必需的，如图 3-69 所示。

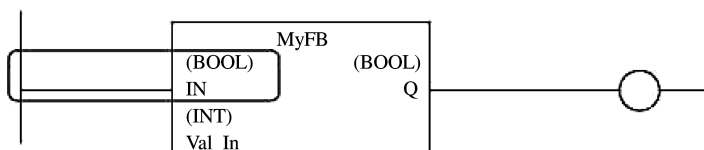


图 3-69 与左母线直接连接

(4) 级联

级联如图 3-70 所示，功能或功能块顶端的 BOOL 输出连接到后续的功能或功能块顶端的 BOOL 输入。

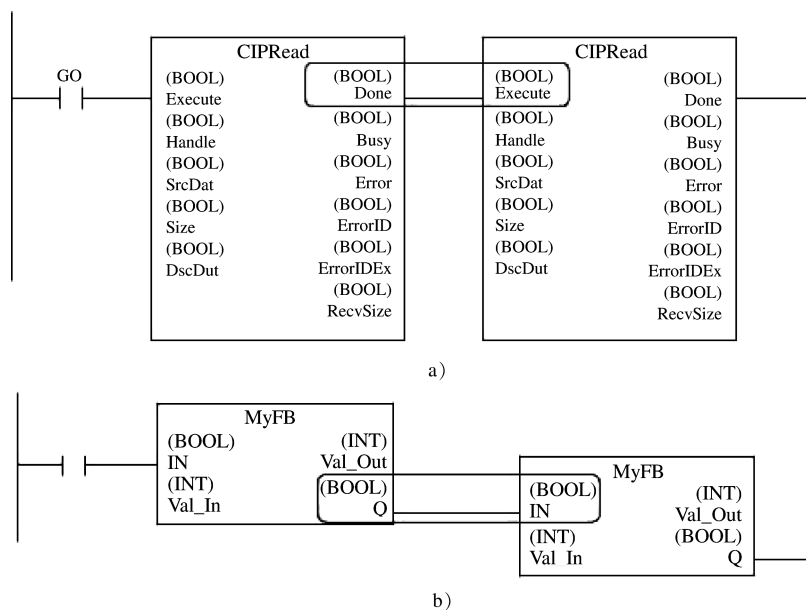


图 3-70 级联

(5) 分支连接

图 3-71 所示为分支连接。

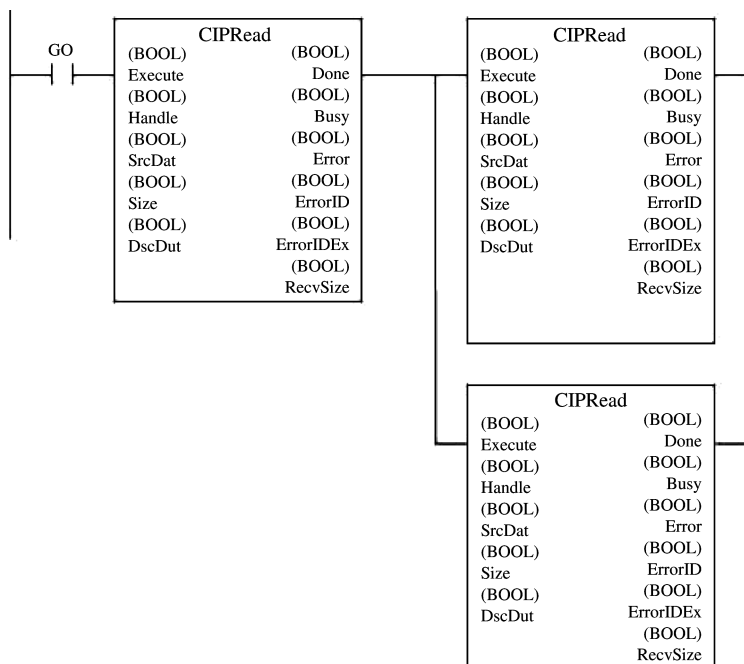


图 3-71 分支连接

(6) 输入取反

图 3-72 所示为输入取反。

(7) 内嵌 ST

如图 3-73 所示，梯形图编程时可以嵌入 ST，称为内嵌 ST。有了内嵌 ST，可以在梯形图中轻松地进行数据处理和字符串处理等。ST 语言代码写在内嵌 ST 框内，内嵌 ST 框的连接线是它的执行条件。

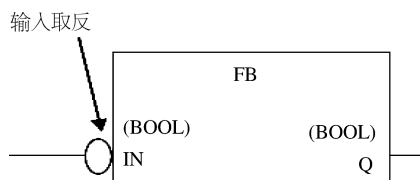


图 3-72 输入取反

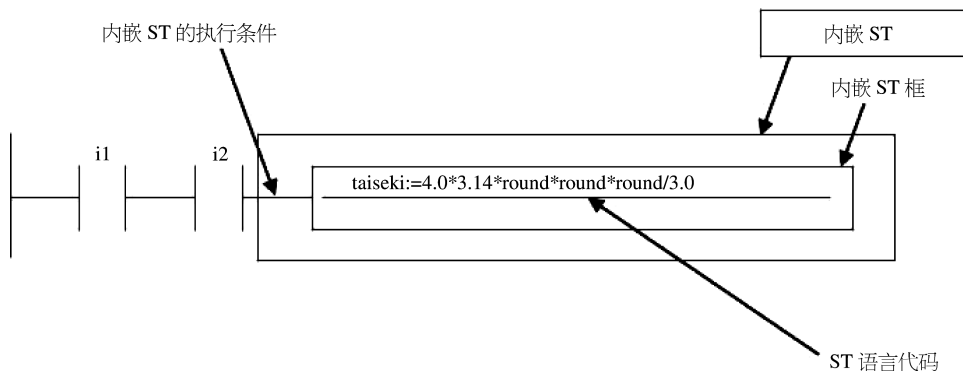


图 3-73 内嵌 ST

3.5.2 结构文本

1. 结构文本的结构

结构文本 (ST) 语言是一种由 IEC 61131-3 标准定义的用于工业控制 (主要是 PLC) 的高级编程语言代码。标准控制语句、运算符和功能使 ST 语言特别适用于复杂数学处理, 而用梯形图则难于实现。

以下为一段 ST 程序:

```
//Program Overview
IF NOT fault THEN
    Ready:=TRUE; //Green Light in ON state
    AlarmSignal:=FALSE;
    (* Turn OFF the red light
       AND enable machine FOR operation* )
    IF NOT HomingDone THEN
        ExecuteHoming:=TRUE;//Make homing FOR 1st TIME;
        OpenGrip:=TRUE;//OpenGrip grip FOR Enable grasping
    END_IF;
END_IF;
```

下面对 ST 做一些说明:

1) 使用 ST 可以像高级语言 C 一样编程, 程序中插入注释而易于阅读。使用“(* ...*)”可以插入单行或多行注释。也有简单方法插入注释, 使用“//”插入单行注释。

2) 语句末必须是单字节的分号“;”。

3) 对于一个较长或复杂的表达式, 允许分成多行, 如:

```
SafetyChain := (Sensor_1 AND Sensor_2 AND NOT Sensor3)
               AND (PowerEnable AND NOT StandStill OR MotorDisabled OR Compres-
               sor)
               OR (MainSupply AND PowerSupply OR SecondarySupply)
               AND NOT (SafetyRelays OR SecurityLatch OR SafetyStop) ;
```

4) 运算符、关键字和变量名称不区分字符的大小写, 但必须是单字节的字符, 使用双字节字符会出现语法错误。

2. 语句关键字和运算符

(1) 语句关键字

语句关键字见表 3-20。

表 3-20 语句关键字

关键字	含 义	关键字	含 义
:=	赋值	REPEAT	重复
RETURN	Return	EXIT	跳出循环
IF	If	;	语句结束或空语句
CASE	Case	(* Text*)	多行注释
FOR	For	//Text	单行注释
WHILE	While		

(2) 运算符

运算符见表 3-21。

表 3-21 运算符

算子	操作		优先级 1:最高级 11:最低级
()	括号	Value:=(1+2)*(3+4)//Value is 21 执行顺序:(), * / , + -	1
	功能/功能块调用		2
**	指数	Value:=2**8;//Value is 256	3
NOT	非	Value:=NOT TRUE;//Value is FALSE	4
* / MOD	乘法	Value:=8*100;//Value is 800	5
	除法	Value:=200/25;//Value is 8	
	求余	Value:=10 MOD 6;//Value is 4	
+	加法	Value:=200+25;//Value is 225	6
-	减法	Value:=200-25;//Value is 175	
<,>,<=,>=	比较	Value:=60>10;//Value is TRUE	7
=	匹配	Value:=8=7;//Value is FALSE	8
<>	不匹配	Value:=8<>7;// Value is TRUE	
AND,&	逻辑与	Value:=2#1001 AND 2#1100;//Value is 2#1000	9
XOR	逻辑异或	Value:=2#1001 XOR 2#1100;//Value is 2#0101	10
OR	逻辑或	Value:=2#1001 OR 2#1100;//Value is 2#1101	11

如果不同优先级的运算符混合,则优先级最高的运算符首先执行。

3. ST 语句

(1) 赋值语句

赋值语句结构为

<variable>:=<expression>;

<variable>:=<variable>;

把右边表达式的值赋给左边的变量。

例 1: 把表达式 X+1 的值赋给变量 A。

A:=X+1;

例 2: 把变量 B 的值赋给变量 A。

A:=B;

例 3: 把常数 10 赋给变量 A。

A:=10;

注意:

1) 源数据类型必须与目标数据类型相匹配, 或者数据类型的组合必须允许隐性转换。否则, 会出现编译错误。

2) 如果要赋的值是 STRING 型数据, 目标 STRING 型变量的长度要大于源 STRING。否则, 会出现错误。

3) 对于 STRING 型变量, 如果左边变量的长度大于右边变量存储的字符串, 赋值是允许的。

例 4: 下面的赋值是允许的。

变量表

变量名	数据类型	长度
Var1	STRING	10
Var2	STRING	20

用户程序

Var2:= 'ABC';

Var1:= Var2

4) 不能为结构体变量赋值, 可对结构体的各个成员赋值。

(2) RETURN

表示当前程序、功能或功能块结束。

(3) 单条件 IF

单条件 IF 的一般结构为

IF <condition_expression> THEN

<statement_1>;

ELSE

<statement_2>;

END_IF;

当条件满足时, 执行一个语句; 当条件不满足时, 执行另一个语句。

IF 的过程流程如图 3-74 所示。

注意:

1) IF 必须与 END_IF 一起使用。

2) 条件表达式是值为 TRUE 或 FALSE 的语句或 BOOL 变量。

3) IF 可以嵌套使用。

例如

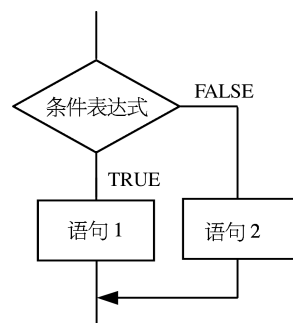


图 3-74 IF 的过程流程

```

IF <condition_expression_1> THEN
  IF <condition_expression_2> THEN
    <statement_1>;
  ELSE
    <statement_2>;
  END_IF;
END_IF;

```

图 3-75 所示为 IF 嵌套的过程流程。

4) 无论 <statement_1> 还是 <statement_2> 都可以写多个语句, 语句之间用分号(;)隔开。

5) 可以省略 ELSE 语句, 如果省略了, 当 <condition_expression> 为 FALSE 时, 不做任何执行。省略 ELSE 的 IF 过程流程如图 3-76 所示。

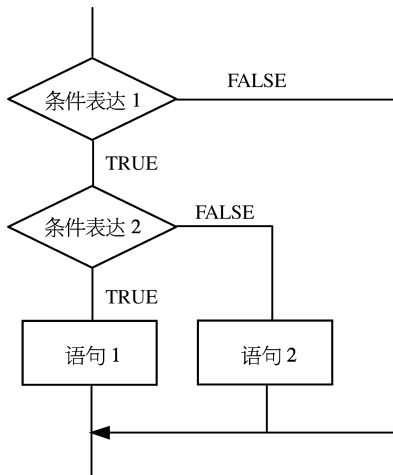


图 3-75 IF 嵌套的过程流程

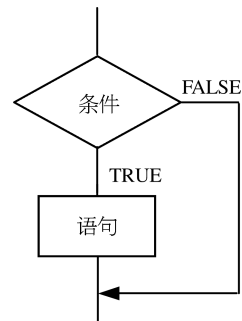


图 3-76 省略 ELSE 的 IF 过程流程

例 1: 当语句 $A > 0$ 为 TRUE 时, 给变量 X 赋值 10; 当语句 $A > 0$ 为 FALSE 时, 给变量 X 赋值 0。

```

IF A > 0 THEN
  X:=10;
ELSE
  X:=0;
END_IF;

```

例 2: 当语句 $A > 0$ 和 $B > 1$ 都为 TRUE 时, 给变量 X 赋值 10、变量 Y 赋值 20; 当语句 $A > 0$ 和 $B > 1$ 只要一个为 FALSE 时, 给变量 X 赋值 0、变量 Y 赋值 0。

```

IF A > 0 AND B > 1 THEN
  X:=10;Y:=20;
ELSE

```

```

X:=0;Y:=0;
END_IF;

```

例3：当 BOOL 变量 A 为 TRUE 时，给变量 X 赋值 10；当变量 A 为 FALSE 时，给变量 X 赋值 0。

```

IF A THEN X:=10;
ELSE X:=0;
END_IF;

```

(4) 多条件 IF

多条件 IF 的一般结构为

```

IF <condition_expression_1> THEN <statement_1>;
ELSIF <condition_expression_2> THEN <statement_2>;
ELSIF <condition_expression_3> THEN <statement_3>;
:
ELSIF <condition_expression_n> THEN <statement_n>;
ELSE <statement_m>;
END_IF;

```

当条件满足时，执行一个语句。当条件不满足时但另一个条件满足时，执行另一个语句。如果条件都不满足，则执行另一个语句。

图 3-77 所示为多条件 IF 的过程流程。

注意：

1) IF 必须与 END_IF 一起使用。
2) 条件表达式是值为 TRUE 或 FALSE 的语句（例如，IF A > 0）或 BOOL 变量。

3) <statement_1> 和 <statement_2> 等可写多个语句，语句之间用分号(;)隔开。

4) 可以省略 ELSE 语句，如果省略，任何条件都不满足，则不执行任何语句。

例 1：

当语句 A > 0 为 TRUE 时，变量 X 赋值为 10。

当语句 A > 0 为 FALSE 并且 B = 1 为 TRUE 时，变量 X 赋值为 1。

当语句 A > 0 为 FALSE 并且 B = 2 为 TRUE 时，变量 X 赋值为 2。

如果没有一个条件为 TRUE 时，变量 X 赋值为 0。

```

IF A > 0 THEN X:=10;
ELSIF B:=1 THEN X:=1;

```

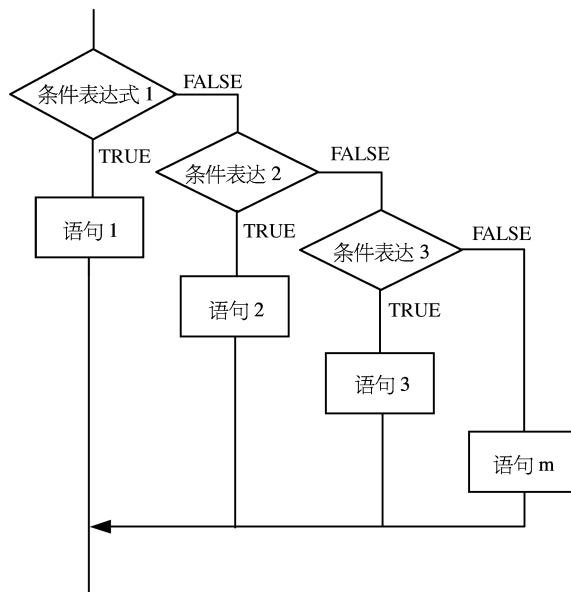


图 3-77 多条件 IF 的过程流程


```
ELSIF B:=2 THEN X:=2;  
ELSE X:=0;  
END_IF;
```

(5) CASE

CASE 是多分支选择语句，它的一般结构为

```
CASE < integer_expression > OF  
    < integer_expression_value_1 > : < statement_1 >;  
    < integer_expression_value_2 > : < statement_2 >;  
    ⋮  
    < integer_expression_value_n > : < statement_n >;  
ELSE < statement_m >;  
END_CASE;
```

计算 CASE 的判断表达式，并按照计算结果依次寻找与之相等的标号值，如果找到就将流程转至该标号处，执行其后的语句；如果找不到与之相等的标号，就只能执行 ELSE 之后的语句。ELSE 对于 CASE 结构而言不是必需的。当不存在 ELSE 且没有相符的标号时，该 CASE 结构就不被执行。

CASE 的过程流程如图 3-78 所示。

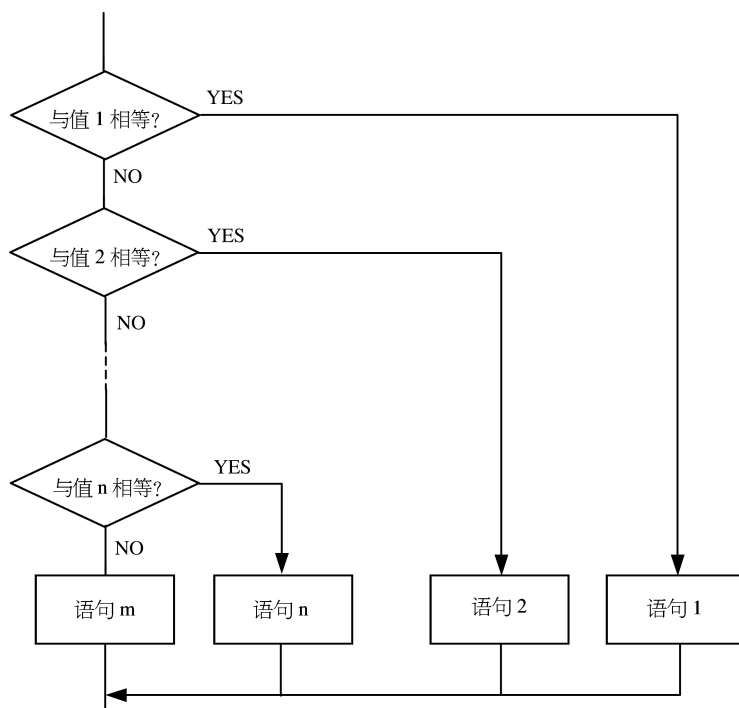


图 3-78 CASE 的过程流程

注意:

1) CASE 必须与 END_CASE 一起使用。

2) <integer_expression> 可以为:

一个整型或枚举体型变量, 例如, abc;

一个整型表达式, 例如, abc + def;

一个返回整型值的功能, 例如, xyz ()。

3) 可以指定一个连续整数范围, 开始整数和结束整数之间放两个句号 (.)。

例 1: 判断表达式为整型变量 A。

```
CASE A OF
  1: X:=1;          -----当变量 A 为 1 时, 变量 X 赋值为 1
  2: X:=2;          -----当变量 A 为 2 时, 变量 X 赋值为 2
  3: X:=3;          -----当变量 A 为 3 时, 变量 X 赋值为 3
ELSE
  X:=0;            -----当没有值匹配时, 变量 X 赋值为 0
END_CASE;
```

例 2: 判断表达式为整型变量 A, 标号值中指定连续整数范围。

```
CASE A OF
  1: X:=1;          -----当变量 A 为 1 时, 变量 X 赋值为 1
  2,5: X:=2;        -----当变量 A 为 2 或 5 时, 变量 X 赋值为 2
  6..10: X:=3;       -----当变量 A 为 6~10 之间时, 变量 X 赋值为 3
  11,12,15..20: X:=4;-----当变量 A 为 11、12 或 15~20 之间时, 变量 X 赋值
                        为 4
ELSE
  X:=0;            -----当 A 没有值匹配时, 变量 X 赋值为 0
END_CASE;
```

例 3: 判断表达式为枚举体变量 ColorVar。

```
CASE ColorVar OF
  RED:
    X := 0;
  BLUE:
    X := 1;
  ELSE
    X := 2;
END_CASE;
```

例 4: 判断表达式为 (a1 + a2), 其值为整型。

```
CASE (a1 + a2) OF
  0:
    X := 0;
  1:
```

```

    X := 1;
ELSE
    X := 2;
END_CASE;

```

例 5：判断表达式为功能 FUN()，其返回值为整数。

```

CASE FUN( ) OF
0:-----根据 FUN( )的返回值分支
    X := 10;
1:
    X := 11;
ELSE
    X := 12;
END_CASE;

```

(6) FOR

FOR 结构的一般形式为

```

FOR <FOR_variable> := <initial_value> TO <end_value> BY <increment/decrement>
DO
    <statement>;
END_FOR;

```

这个结构重复执行同一语句，直到变量（称为 FOR_variable）从一个值变为另一个值。

FOR 的过程流程如图 3-79 所示，Increment/decrement 可以为正数，也可以为负数。

FOR 的执行过程如下。

1) 首先为变量 FOR_variable 赋初值，注意在整个循环中只执行一次。

2) 重复下面的过程：计算表达式 FOR_variable > END_variable，若其值为 FALSE，就执行一次循环体语句，然后将 Increment/decrement 加到 FOR_variable 上；再计算表达式 FOR_variable > END_variable，判断其值是否为“FALSE”，直到 FOR_variable > END_variable 的值为 TRUE，就不再执行循环体了。

如果 Increment/decrement 为负数，表达式 FOR_variable > END_variable 中的比较符号由“>”改为“<”。

注意：

- 1) 如果 FOR_variable 是有符号的，Increment/decrement 可为负数。
- 2) FOR 总是与 END_FOR 一起使用。
- 3) END_variable 和 Increment/decrement 这两项，不能使用加法或其他算术表达式。

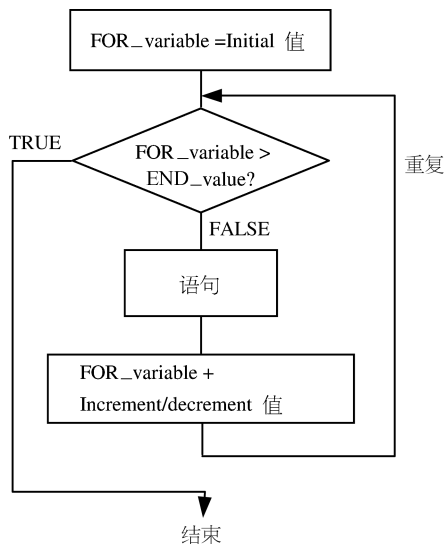


图 3-79 FOR 的过程流程

4) FOR 循环执行完毕时, FOR_variable 的值变为 END_variable 加上 Increment/decrement。

例 1: 当 FOR 循环结束时, i 的值是 101。

```
FOR i:=0 TO 100 BY 1 D
  X[i]:=0;
END_FOR;
//Here, i is 101
```

5) 不要在循环体内直接修改 FOR_variable, 否则, 会出现意想不到的结果。

例 2:

```
FOR i:=0 TO 100 BY 1 DO
  X[i]:=0;
  i:=i+INT#5;
END_FOR;
```

6) 循环体内可以编写多条语句, 语句之间用分号 (;) 隔开。

7) 可以省略 BY <Increment/decrement>, 如果省略了则默认 Increment 的值为 1。

8) 可以为 <FOR_variable>、<initial_value>、<Increment/decrement> 指定任意的表达式, 这个表达式的值为整数变量或整数值。也可以指定一个功能, 其返回值为整数。

例 3: 将数值 100 赋值给数组变量元素 SP[n], FOR_variable 是 n, 初始值为 0, 结束值为 50, 增量为 5。

```
FOR n:= 0 TO 50 BY 5 DO
  SP[n] := 100;
END FOR;
```

例 4: 计算数组元素 DATA[1] ~ DATA[50] 的和, 将结果放到变量 SUM 中。

```
SUM:=0;
FOR n:= 0 TO 50 BY 1 DO
  SUM:= SUM + DATA[n] ;
END_FOR;
```

例 5: 找到数组元素 DATA[1] ~ DATA[50] 的最大值和最小值。最大值放到 MAX 中, 最小值放到 MIN 中。DATA [n] 的取值范围是 0 ~ 1000。

```
MAX:=0;
MIN:=1000;
FOR n:=1 TO 50 BY 1 DO
  IF DATA[n] > MAX THEN
    MAX:=DATA[n];
  END_IF;
  IF DATA[n] < MIN THEN
    MIN:=DATA[n];
  END_IF;
END_FOR;
```

9) 如果 FOR 循环总的执行时间超过周期时间, 会出现任务周期超出错误。

例 6: FOR_variable 变量在逻辑上无法达到结束值。

```
FOR i:= 0 TO 100 BY 1 DO
    intArray[i] := i;
    i:= INT#50;-----无限循环,出现任务周期超出错误
END_FOR;
FOR i:= 0 TO 100 BY 0 DO-----无限循环,出现任务周期超出错误
;
END_FOR;
```

例 7: 当 FOR_variable 变量超出结束值时, 会出现上溢或下溢。

```
FOR i:= 0 TO 254 BY 2 DO
    INTArray[i] := i;
END_FOR;
```

<FOR_variable>、<initial_value>、<Increment/decrement> 这三项的数据类型可以为整型或枚举型。必须为同一数据类型, 否则, 在 Sysmac Studio 建立程序时会出错。

(7) WHILE

WHILE 的结构为

```
WHILE <condition_expression> DO
    <statement>;
END_WHILE;
```

WHILE 是一种循环结构, 只要条件为 TRUE, 则重复执行指定的语句。

WHILE 的过程流程如图 3-80 所示。

循环体在执行 WHILE 语句时, 先对条件表达式 Condition_expression 进行计算, 若其值为 TRUE, 则执行循环体语句 <statement>; 否则不执行 <statement>。每执行一次循环体语句后, 都对条件表达式进行一次计算和判断。当发现条件表达式的值为 FALSE 时, 便立即退出循环。

注意:

- 1) WHILE 总是与 END_WHILE 一起使用。
- 2) 在循环体语句执行前, 如果条件表达式的值是 FALSE, 则 WHILE 循环终止, 循环体语句不再执行。
- 3) 循环体中可以有多条语句, 语句之间用分号 (;) 隔开。

- 4) 条件表达式可以指定为 BOOL 型变量, 包括返回值为 BOOL 值的功能。

例 1: 计算出大于 1000 且能被 7 整除的第一个整数, 并将其赋值给变量 A。

```
A := 0;
WHILE A <= 1000 DO
    A := A + INT#7;
END_WHILE;
```

例 2: 如果变量 X 小于 3000, 则 X 的值加倍, 并将其赋给数组变量元素 DATA[1]。接

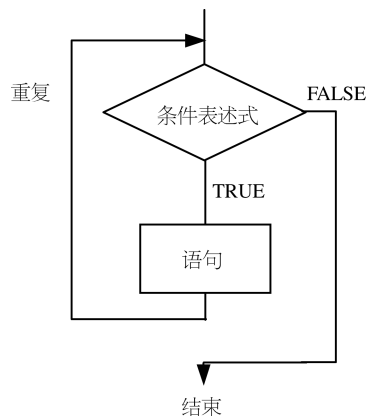


图 3-80 WHILE 的过程流程

着, X 的值再加倍, 赋给数组变量元素 DATA[2]。这个过程重复。

```
n := 1;
X := 1;
WHILE X < 3000 DO
  X:= X* INT#10#2;
  DATA[n]:= X;
  n := n + INT#1;
END_WHILE;
```

5) 如果条件表达式不正确, 程序执行时间增加, 会引起任务周期超出错误。

例 3:

```
boolVar := TRUE;
WHILE boolVar DO
  intVar := intVar + INT#1;
END_WHILE;
```

(8) REPEAT

REPEAT 的结构为

```
REPEAT
  <statement>;
UNTIL <condition_expression>
END_REPEAT;
```

REPEAT 是一种“直到型”循环结构, 它的过程流程如图 3-81 所示。

当流程到达关键字 REPEAT 时, 立即执行循环体一次, 然后才对判断表达式进行计算和测试。若表达式的值为 FALSE, 则重复执行循环体一次; 否则退出。与 WHILE 结构相比, REPEAT 结构至少要执行循环体一次。这样的结构应用于需要事先执行一次的循环体的程序中。

例 1: 将 1~10 之间的整数加起来, 并把和赋给变量 TOTAL。

```
A := 1;
TOTAL:= 0;
REPEAT
  TOTAL:= TOTAL + A;
  A:= A + INT#1;
UNTIL A > 10
END_REPEAT;
```

例 2: 如果不能正确编写条件表达式, 程序执行时间增加, 会引起任务周期超出错误。

```
intVar := INT#1;
REPEAT
```

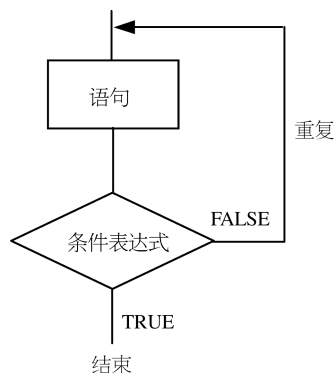


图 3-81 REPEAT 的过程流程

```
intVar := intVar + INT#1;  
UNTIL intVar := INT#0  
END_REPEAT;
```

(9) EXIT

只用在重复结构 (FOR 结构、WHILE 结构或 REPEAT 结构) 体内，退出循环。
循环体中，在 IF 结构内使用这个语句，当条件满足时，从循环体内退出。

```
FOR (WHILE, REPEAT) <statement >  
  ⋮  
  IF <condition_expression > THEN  
    EXIT;  
  END_IF;  
  ⋮  
END_FOR (WHILE, REPEAT);
```

第 4 章 NJ 的指令系统

4.1 概述

NJ 系列控制器提供了丰富的指令系统。根据功能这些指令分为：梯形图指令、ST 语句指令、顺序输入指令、顺序输出指令、顺序控制指令、比较指令、定时器指令、计数器指令、数学指令、BCD 转换指令、数据类型转换指令、位串处理指令、选择指令、数据传送指令、移位指令、转换指令、堆栈和表指令、FCS 指令、文本串指令、时间和一天中的时间指令、系统控制指令、通信指令、SD 存储卡指令及一些特殊指令。

1. 指令的表示

(1) 指令的梯形图表示

NJ 的绝大多数指令用功能 (FUN) 或功能块 (FB) 表示。功能可从程序、功能块和功能调用。功能块只能从程序和功能块调用。

图 4-1 所示为指令的梯形图表示。

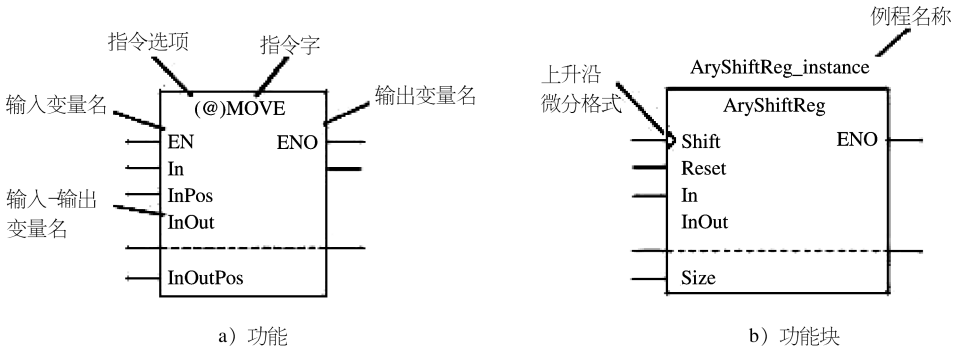


图 4-1 指令的梯形图表示

1) 功能的指令选项

图 4-1a 中，指令选项 “@” 表示上升沿微分，如果指令字前放置 “%” 表示下降沿微分。如果指定为上升沿微分时，输入变量 EN 在上一个任务周期时为 FALSE，而在当前任务周期时变为 TRUE，则指令将执行一次。同样，如果指定为下降沿微分时，输入变量 EN 在上一个任务周期时为 TRUE，而在当前任务周期时变为 FALSE，则指令将执行一次。

2) 功能块的上升沿微分格式

图 4-1b 中，功能块的输入端的箭头表示指令在 Shift 的上升沿执行。

3) 功能块的例程名称

在 FB 指令的上方有一个例程名称, 例如 “AryShiftReg_instance”, 必须为 FB 指令的任何例程分配一个例程名称。

(2) 指令的 ST 表示

指令用 ST 表示时, 有两种方法。

1) 在参数和输入、输出、输入-输出变量之间直接指明对应关系

例如, MoveBit(In:=abc, InPos:=def, InOut:=ghi, InOutPos:=jkl);

2) 只指明参数, 忽略输入、输出和输入-输出变量

例如, MoveBit (In, InPos, InOut, InOutPos);

AryShiftReg_instance (Shift, Reset, In, InOut, Size);

2. 公用变量

指令所用的变量分为输入、输出和输入-输出变量。其中, EN、ENO、Execute、Done、Busy、Error、ErrorID 和 ErrorIDEx 这 8 个变量为许多指令所使用, 称为公用变量。

(1) EN

EN 是 FUN 指令的一个输入变量, 产生指令的执行条件。

EN 为 BOOL 型, 当 EN 为 TRUE 时, 指令执行; 当 EN 为 FALSE 时, 指令不执行。其默认值为 TRUE。

如果 “@” 设为指令选项, 则 EN 由 FALSE→TRUE 时, 指令执行一次。如果 “%” 设为指令选项, 则 EN 由 TRUE→FALSE 时, 指令执行一次。

FB 指令没有 EN 输入变量。

用 ST 调用 FUN 指令时, 忽略 EN 输入变量。因为指令的执行条件由操作顺序确定, 不需要 EN 输入变量。

(2) ENO

ENO 为输出使能变量, 在梯形图中为下一条指令传递一个执行条件。通常, 当指令执行完成后, ENO 值变为 TRUE, 下一条指令便启动。

ENO 为 BOOL 型, 当 ENO 为 TRUE 时表示指令执行正常结束; ENO 为 FALSE 时, 表示错误结束, 或程序正在执行, 或执行条件不满足。

大多数 FUN 指令和 FB 指令都有 ENO 输出变量, 但是, 有些指令没有 ENO 输出变量。

在 ST 中省略 ENO 输出变量。ENO 输出变量在 ST 中并不是必须有的, 因为下一条指令的执行条件是由操作顺序确定的。

(3) Execute、Done 和 Busy

Execute、Done 和 Busy 变量见表 4-1。

Execute 是某些 FB 指令执行条件的输入变量。当 Execute 变为 TRUE 时, 指令开始执行。即使 Execute 的值变为 FALSE 或指令的执行时间超过任务周期, 指令的执行仍将继续直到处理完成。

Done 是某些 FB 指令用来表示指令执行完成的输出变量。

Busy 是某些 FB 指令用来表示指令正在执行的输出变量。

表 4-1 Execute、Done 和 Busy 变量

名称	含义	I/O	说 明	数据类型	有效范围	初始值
Execute	执行	输入	TRUE: 指令被执行* 1 FALSE: 指令没被执行* 2	BOOL	TEUE 或 FALSE	FALSE
Done	完成	输出	TRUE: 正常结束* 3* 4 FALSE: 错误结束, 执行过程中或执行条件不满足	BOOL	TEUE 或 FALSE	...
Busy	正在执行		TRUE: 正在执行 FALSE: 不在执行			

* 1 当控制器运行开始时, 如果 Execute 的值已经是 TRUE, 则指令不被执行。这种情况下, 如果要执行指令, 首先要把 Execute 的值改为 FALSE。

* 2 在执行过程中即使 Execute 变为 FALSE, 指令的执行依然继续直到结束。

* 3 正常结束后, 当执行条件不再满足时, Done 的值变为 FALSE。

* 4 正常结束时, 如果执行条件不再满足时, Done 的值为 TRUE 保持一个任务周期, 然后变为 FALSE。

(4) Error、ErrorID 和 ErrorIDEx

Error、ErrorID 和 ErrorIDEx 变量见表 4-2, 它们是输出变量, 某些 FB 指令用它们来表示执行时出现的错误。

表 4-2 Error、ErrorID 和 ErrorIDEx 变量

名称	含义	I/O	说 明	数据类型	有效取值范围	初始值
Error	错误	输出	TRUE: 错误结束* 1* 2 FALSE: 正常结束、执行进行中或执行条件不满足	BOOL	TEUE 或 FALSE	...
ErrorID	错误代码		这是出现错误时的错误代码 正常结束时的值为 WORD#16#0	WORD	取决于指令	
ErrorIDEX	错误扩展代码		扩展单元硬件错误的错误代码 正常结束时的值为 DWORD#16#0	DWORD		

* 1 错误结束后, 当执行条件不再满足时, Error 的值变为 FALSE。

* 2 错误结束时, 当执行条件不再满足时, Error 的值为 TRUE 保持一个任务周期, 然后变为 FALSE。

3. Execute、Done、Busy 和 Error、ErrorID、ErrorIDEx 的时序图

图 4-2 所示为正常结束时的时序。

图 4-3 所示为出现错误时的时序。

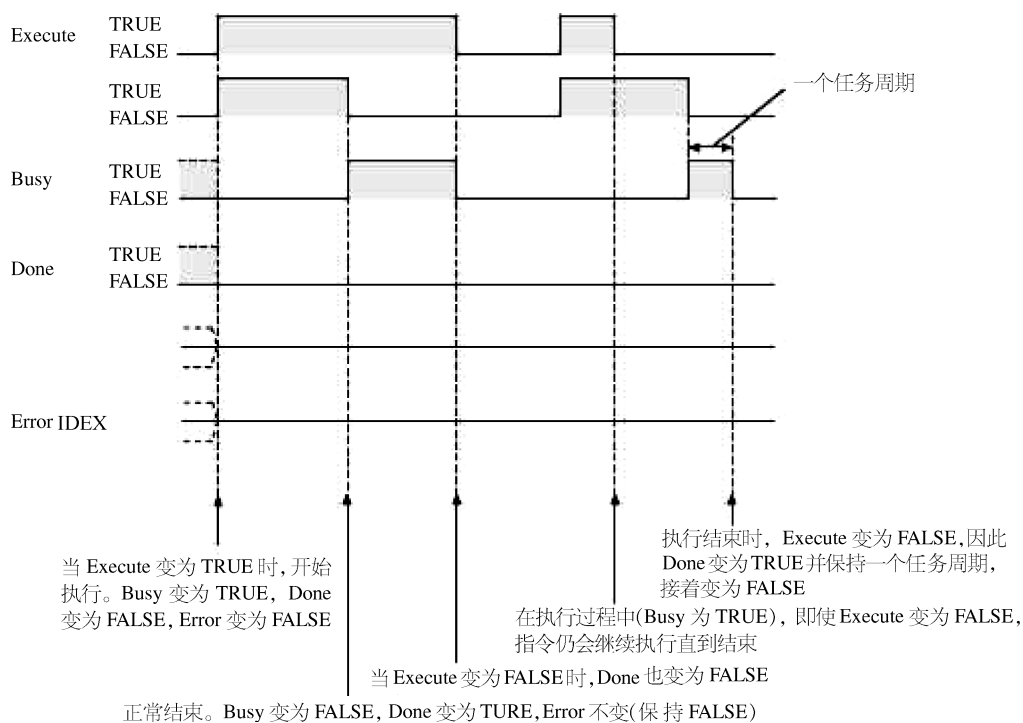


图 4-2 正常结束时的时序

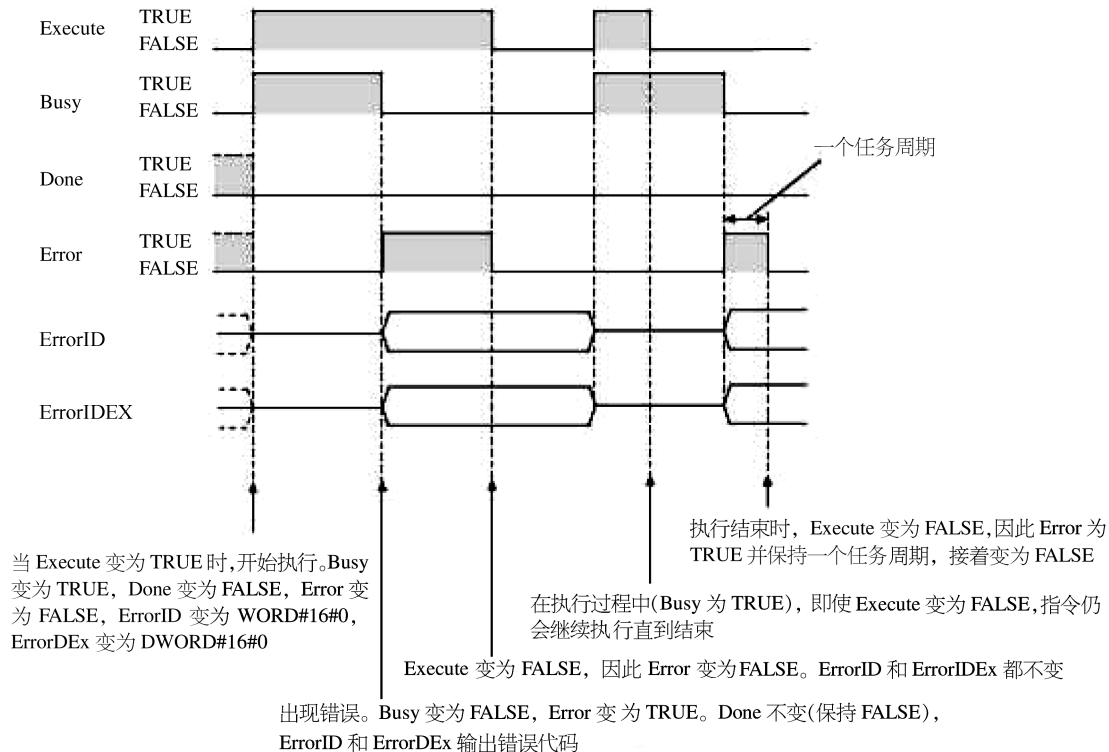


图 4-3 出现错误时的时序

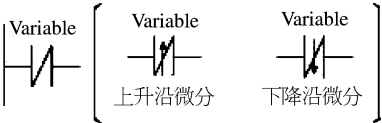
4.2 梯形图指令

NJ 提供梯形图指令 LD/LDN、AND/ANDN、OR/ORN 和 OUT/OUTN。

1. LD/LDN 指令

LD/LDN 指令见表 4-3。

表 4-3 LD/LDN 指令

指令	名称	FB/FUN	梯 形 图	ST
LD	装载	...		无
LDN	装载非	...		无

功能：

LD 指令读取一个指定 BOOL 变量的值，并将其输出给下一条指令。如果指定变量的值是 TRUE，则输出为 TRUE；如果值是 FALSE，则输出为 FALSE。LD 指令表示与左侧母线连接的第一个常开触点或者逻辑块的第一个常开触点。

LDN 指令读取一个指定 BOOL 变量值的非，并将其输出给下一条指令。如果指定变量的值是 TRUE，则输出为 FALSE；如果值是 FALSE，则输出为 TRUE。LDN 指令表示与左侧母线连接的第一个常闭触点或者逻辑块的第一个常闭触点。

LD/LDN 指令还有上升沿微分和下降沿微分的形式，见表 4-4。

表 4-4 LD/LDN 指令的上升沿微分和下降沿微分

名称	微分	变量上一次执行时的值和当前值	输出值
LD	上升沿微分	上一次执行时为 FALSE→当前为 TRUE	TRUE
		其他	FALSE
	下降沿微分	上一次执行时为 TRUE→当前为 FALSE	TRUE
		其他	FALSE
LDN	上升沿微分	上一次执行时为 FALSE→当前为 TRUE	FALSE
		其他	TRUE
	下降沿微分	上一次执行时为 TRUE→当前为 FALSE	FALSE
		其他	TRUE

图 4-4 所示为 LD/LDN 指令的应用。

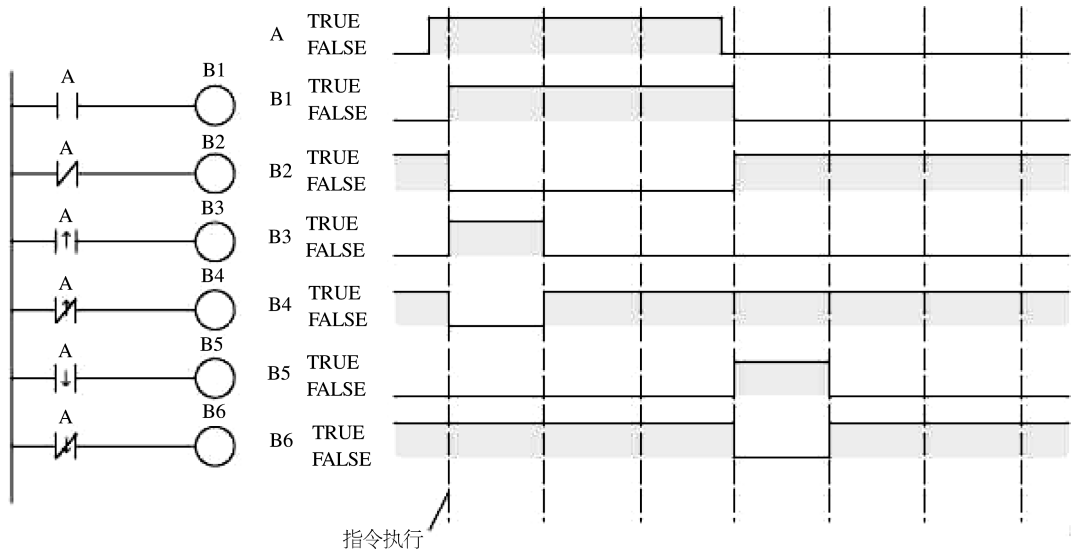


图 4-4 LD/LDN 指令的应用

2. AND/ANDN 指令

AND/ANDN 指令见表 4-5，除了 AND/ANDN 基本形式之外，还有上升沿微分和下降沿微分两种形式。

表 4-5 AND/ANDN 指令

指令	名称	FB/FUN	梯 形 图	ST
AND	与	...	Variable Variable Variable 上升沿微分 下降沿微分	Result: = vBool1 AND vBool2; Result: = vBool1 & vBool2;
ANDN	与非	...	Variable Variable Variable Variable 上升沿微分 下降沿微分	Result: = vBool1 AND NOT vBool2;

3. OR/ORN 指令

OR/ORN 指令见表 4-6，除了 OR/ORN 基本形式之外，还有上升沿微分和下降沿微分两种形式。

4. OUT/OUTN 指令

OUT/OUTN 指令见表 4-7，除了 OUT/OUTN 基本形式之外，还有上升沿微分和下降沿微分两种形式。

表 4-6 OR/ORN 指令

指令	名称	FB/FUN	梯形图	ST
OR	或	...		Result: = vBool1 OR vBool2;
ORN	或非	...		Result: = vBool1 OR NOT vBool2;

表 4-7 OUT/OUTN 指令

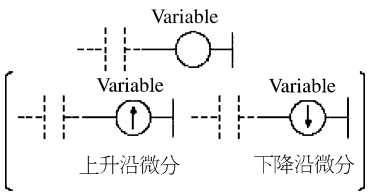
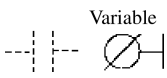
指令	名称	FB/FUN	梯形图	ST
Out	输出	...		Variable: = (到前一条指令为止的逻辑表达式);
OutNot	输出非	...		Variable: = NOT(到前一条指令为止的逻辑表达式);

图 4-5 所示为 OUT/OUTN 指令的应用。

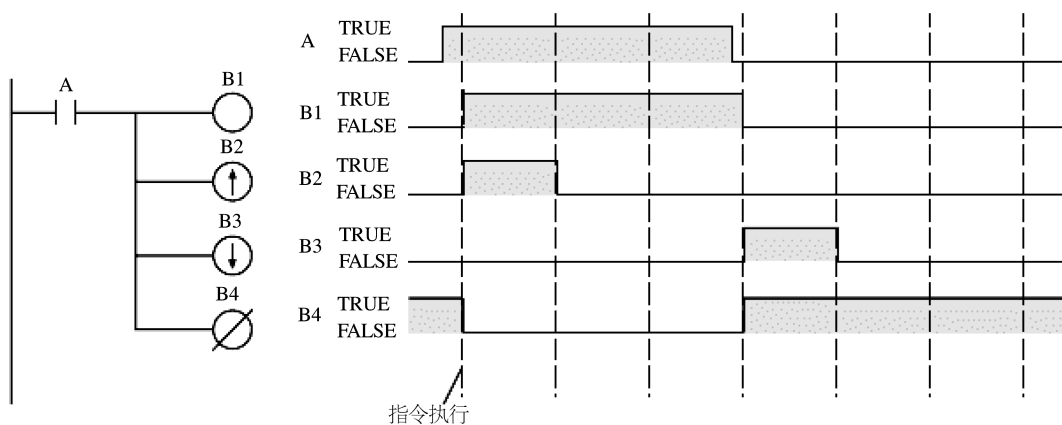


图 4-5 OUT/OUTN 指令的应用

图 4-6 所示为梯形图正确连接。

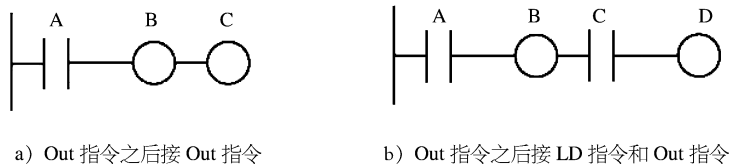


图 4-6 梯形图正确连接

图 4-7 所示为梯形图不正确连接。

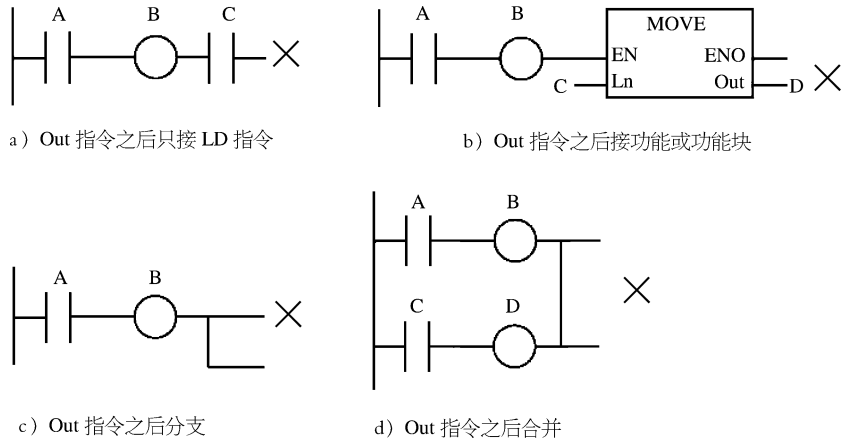


图 4-7 梯形图不正确连接

4.3 顺序输入指令

NJ 提供如下顺序输入指令。

R_TRIG (Up) 和 F_TRIG (Down): 上升沿触发和下降沿触发。

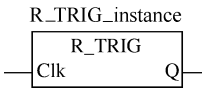
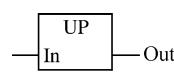
TestABit 和 TestABitN: 位测试和位测试非。

下面介绍 R_TRIG (Up) 和 F_TRIG (Down) 指令。

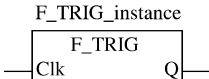
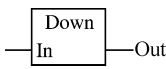
1. 指令的梯形图及 ST

R_TRIG (Up) 和 F_TRIG (Down) 指令见表 4-8。

表 4-8 R_TRIG (Up) 和 F_TRIG (Down) 指令

指令	名称	FB/FUN	梯 形 图	ST
R_TRIG	上升沿触发	FB		R_TRIG_instance(Clk,Q);
Up		FUN		无

(续)

指令	名称	FB/FUN	梯 形 图	ST
F_TRIG	下降沿触发	FB		F_TRIG_instance(Clk,Q);
Down		FUN		无

2. 变量

Clk、In：输入变量，BOOL 型。

Q、Out：输出变量，BOOL 型。

3. 功能

1) R_TRIG 指令：当输入信号 CLK 由 OFF 变为 TRUE 时，使输出 Q 为 TRUE 并保持一个任务周期，其他情况下，Q 是 FALSE。在执行这条指令的第一个任务周期内，无论 CLK 是什么值，Q 的值是 FALSE。当电源上电时，如果 CLK 的值是 TRUE，Q 的值一直为 FALSE，直到 CLK 变为 FALSE 再变为 TRUE 时，Q 的值才为 TRUE 并保持一个任务周期。

2) UP 指令：与 R_TRIG 指令的功能完全一样。UP 指令的 In 变量对应 R_TRIG 指令的 CLK 变量，Out 变量对应 Q 变量。

3) F_TRIG 指令：当输入信号 CLK 由 TRUE 变为 OFF 时，使输出 Q 为 TRUE 一个任务周期，其他情况下，Q 是 FALSE。在执行这条指令的第一个任务周期内，无论 CLK 是什么值，Q 的值是 FALSE。当电源上电时，如果 CLK 的值是 FALSE，Q 的值一直为 FALSE，直到 CLK 变为 TRUE 时再变为 FALSE，Q 的值才为 TRUE 并保持一个任务周期。

4) Down 指令：与 F_TRIG 指令的功能完全一样。Down 指令的 In 变量对应 F_TRIG 指令的 CLK 量，Out 变量对应 Q 变量。

图 4-8 所示为 R_TRIG/Up 指令的应用。

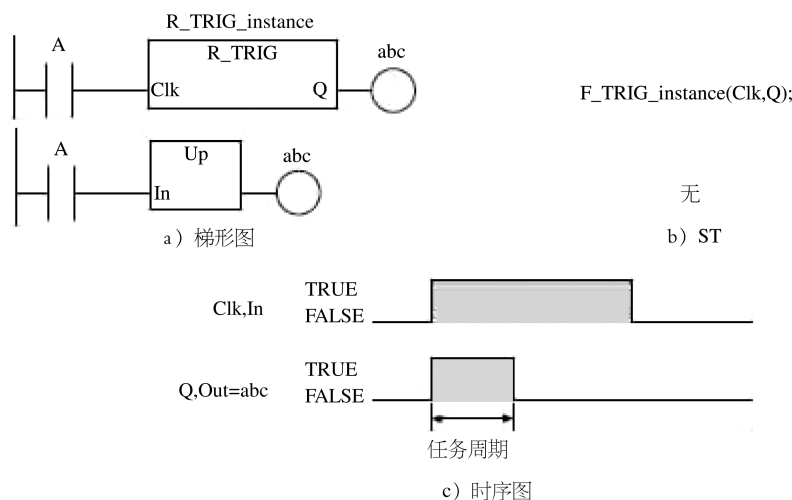


图 4-8 R_TRIG/UP 指令的应用

图 4-9 所示为 F_TRIG/Down 指令的应用。

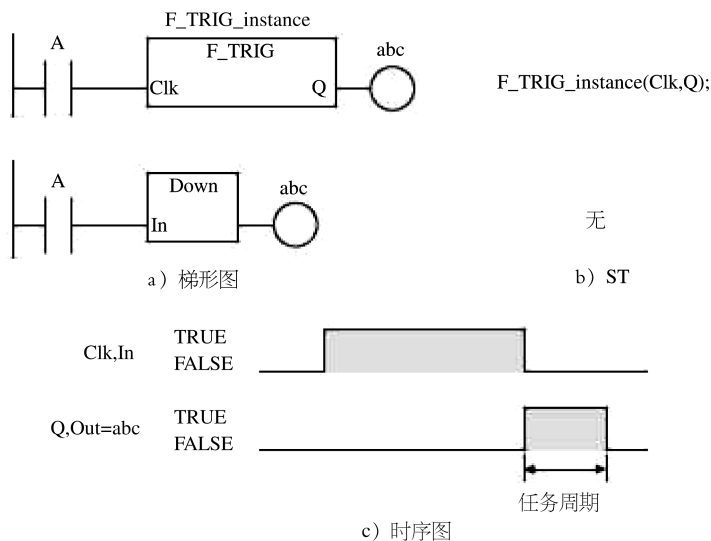


图 4-9 R_TRIG/Down 指令的应用

4.4 顺序输出指令

顺序输出指令见表 4-9。

1. 复位优先保持 RS 指令

(1) 指令的梯形图及 ST

RS 指令的梯形图及 ST 如图 4-10 所示。

表 4-9 顺序输出指令

指令	FB/FUN	名 称
RS	FB	复位优先保持
SR	FB	置位优先保持
Set/Reset	...	置位/复位
SetBits/ResetBit	FUN	多位置位/多位复位
SetABit/ResetABit	FUN	1 位置位/1 位复位
OutABit	FUN	输出 1 位

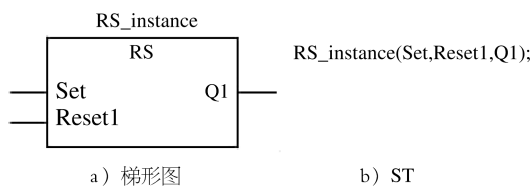


图 4-10 RS 指令的梯形图及 ST

(2) 变量

Set: 置位输入, 输入变量, BOOL 型, 默认值 0。

Reset1: 复位输入, 输入变量, BOOL 型, 默认值 0。

Q1: 保持输出, 输出变量, BOOL 型。

(3) 功能

RS 指令输出具有复位优先的自保持功能, 见表 4-10。

表 4-10 RS 指令功能

Set	Reset1	Q1	Set	Reset1	Q1
TRUE	TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE	不变

图 4-11 所示为 RS 指令的应用。

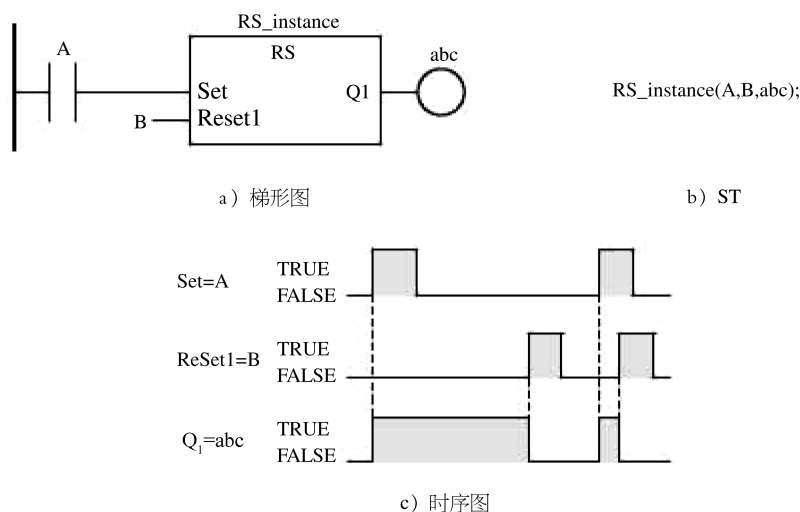


图 4-11 RS 指令的应用

RS 指令的功能和启保停电路一样，如图 4-12 所示。

2. 置位优先 SR 指令

(1) 指令的梯形图及 ST

SR 指令的梯形图及 ST 如图 4-13 所示。

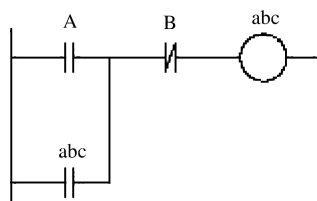


图 4-12 启保停电路

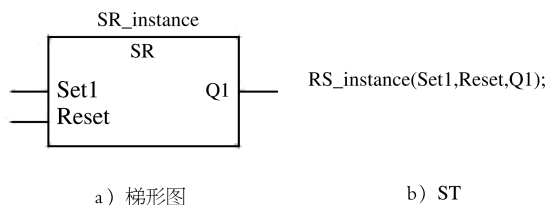


图 4-13 SR 指令的梯形图及 ST

(2) 变量

Set1: 置位输入，输入变量，BOOL 型，默认值 0。

Reset: 复位输入，输入变量，BOOL 型，默认值 0。

Q1: 保持输出，输出变量，BOOL 型。

(3) 功能

SR 指令输出具有置位优先的自保持功能，见表 4-11。

表 4-11 SR 指令功能

Set1	Reset	Q1	Set1	Reset	Q1
TRUE	TRUE	TRUE	FALSE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE	不变

图 4-14 所示为 SR 指令的应用。

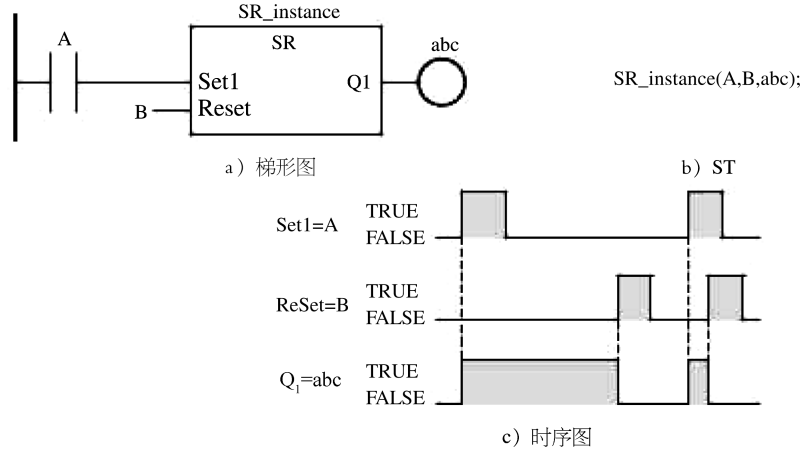


图 4-14 SR 指令的应用

SR 指令的功能和图 4-15 所示的自保电路一样。

3. Set/Reset 指令

Set/Reset 指令见表 4-12。

功能：

当 Set 指令的输入 In 为 TRUE 时，使 Out 置位为 TRUE；当 In 为 FALSE 时，Set 指令不改变 Out 的状态。使用 Reset 指令将 Out 变为 FALSE。

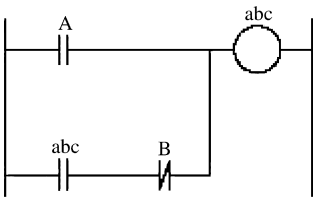


图 4-15 自保电路

表 4-12 Set/Reset 指令

指令	名称	梯形图	ST
Set	置位		无
Reset	复位		无

当 Reset 指令的输入 In 为 TRUE 时, 使 Out 复位为 FALSE; 当 In 为 FALSE 时, Reset 指令不改变 Out 的状态。使用 Set 指令将 Out 变为 TRUE。

Set/Reset 指令还有上升沿微分和下降沿微分两种形式。

图 4-16 所示为 Set/Reset 指令的应用。

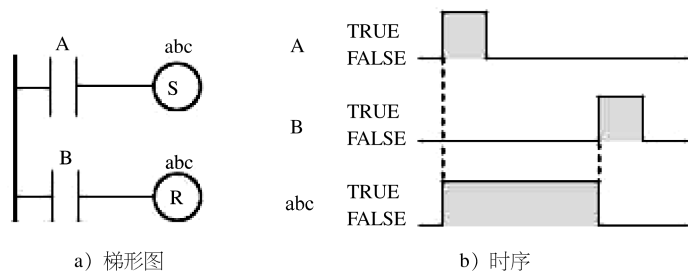


图 4-16 Set/Reset 指令的应用

图 4-17 所示为 Set/Reset 指令的上升沿微分形式。

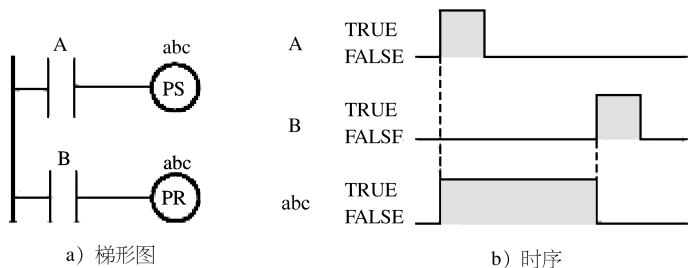


图 4-17 Set/Reset 指令的上升沿微分形式

图 4-18 所示为 Set/Reset 指令的下降沿微分形式。

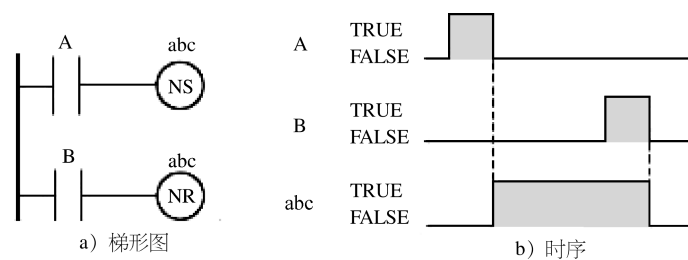


图 4-18 Set/Reset 指令的下降沿微分形式

4.5 顺序控制指令

顺序控制指令见表 4-13。

1. 结束指令 End

功能: END 指令结束当前任务周期内程序的执行。

图 4-19 所示为 END 指令的应用, 当 END 指令执行后, 跟在后面的 SR 指令不再执行。

表 4-13 顺序控制指令

指令	FB/FUN	名 称
END	FUN	结束
RETURN	FUN	返回
MC/MCR	...	主控启动/主控结束
JMP	FUN	跳转
FOR/NEXT	FUN	循环开始/循环结束
BREAK	FUN	中断循环

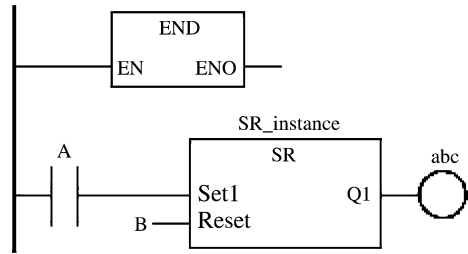


图 4-19 END 指令的应用

2. 返回指令 RETURN

RETURN 指令结束一个功能或功能块，并返回到调用指令中。

图 4-20 所示为 RETURN 指令的应用，当 RETURN 指令执行后，跟在后面的 SR 指令不再执行。

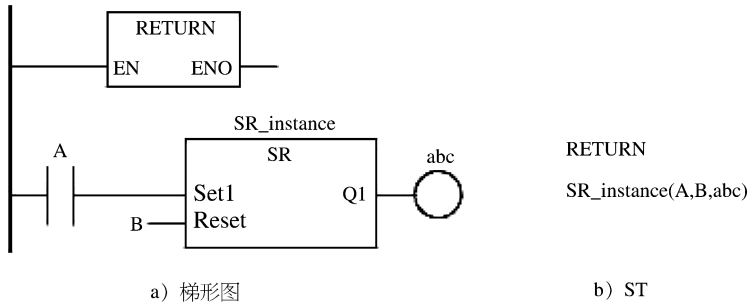


图 4-20 RETURN 指令的应用

3. 主控启动/主控结束指令 MC/MCR

(1) 指令的梯形图及 ST

MC/MCR 指令见表 4-14。

表 4-14 MC/MCR 指令

指令	名称	梯形图	ST
MC	主控启动		无
MCR	主控结束		无

(2) 变量

In (仅 MC 指令): 主控输入，输入变量，BOOL 型，当 In 取值为 FALSE 时，复位主控区。

MCNo: 主控号, 输入变量, UINT 型, 取值范围为 0 ~ 14, 默认值 1。

(3) 功能

MC 总是和 MCR 一起使用。MC 标记一个主控区的起始点, MC 可复位主控区; MCR 标记一个主控区的结束点。使用 MC 可方便地控制对一个比较长的程序段处理的执行条件。

MC/MCR 指令的应用如图 4-21 所示, MC 和 MCR 之间的区域称为主控区。如果 MC 的输入条件为 TRUE, 则主控区的程序正常执行, 如同没有 MC 和 MCR 一样。如果 MC 的输入条件为 FALSE, 则主控区内连接到左母线上所有 LD 指令输出值为 FALSE, 这称为主控复位。

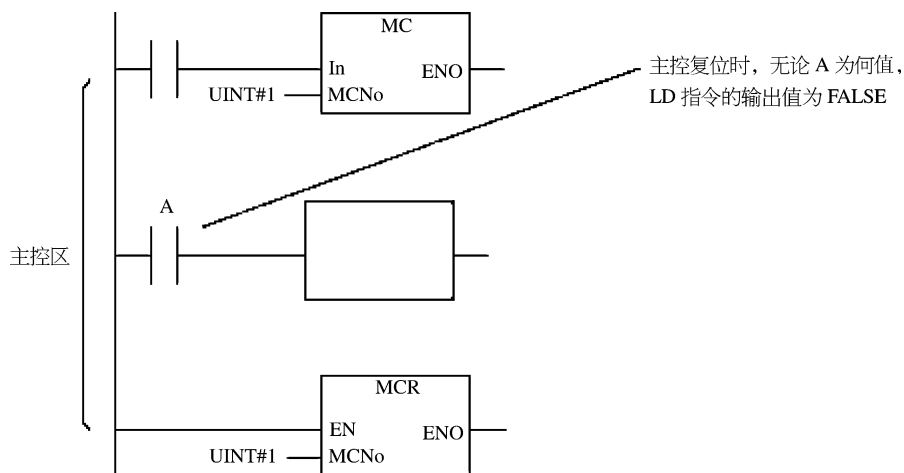


图 4-21 MC/MCR 指令的应用

主控复位时, 主控区指令输出结果如下。

Out 和 OutABit: 输出为 FALSE;

OutNot: 输出为 FALSE;

Set 和 Reset: 保持以前状态;

TON、TOF、TP 和 Timer: 复位;

Accumulation Timer: 停止;

CTU、CTD 和 CTUD: 保持以前状态;

JMP: 不执行;

FOR/NEXT 指令: 不执行;

BREAK: 不执行;

多个任务周期执行的功能块: 如果主控复位之前, 指令正在执行, 则继续执行直至结束。

4. 跳转指令 JMP

当执行条件为 TRUE 时, JMP 指令将处理转移到跳转目的地, 在梯形图中为跳转目的地指定一个标签, 标签可以是任意文本字符串。

图 4-22 所示为 JMP 指令的应用, 标签使用文本字符串 STEP1。JMP 指令执行时, 跳转到标记为 STEP1 的位置, JMP 指令和标签之间的 Out 指令不执行, 变量 B 的值保持。

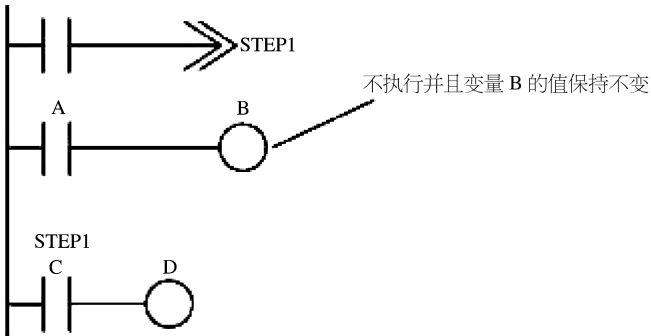


图 4-22 JMP 指令的应用

5. 循环开始/循环结束指令 FOR/NEXT

(1) 指令的梯形图及 ST

FOR/NEXT 指令见表 4-15。

表 4-15 FOR/NEXT 指令

指令	名称	梯形图	ST
FOR	循环开始		FOR Index:= InitVal TO EndVal BY StepVal DO expression END_FOR* ; * 在 ST,不用 NEXT 而用 END_FOR 表示循环的结束位置
NEXT	循环结束		

(2) 变量

InitVal: 初始值, 输入变量, 与 InitVal 相同的数据类型, 循环启动时 Index 的值。

EndVal: 结束值, 输入变量, 与 InitVal 相同的数据类型, 循环结束时 Index 的值。

StepVal: 步长, 输入变量, 与 InitVal 相同的数据类型, 每次循环时加到 Index 上的值。

Index: 控制变量, 输出变量, 与 InitVal 相同的数据类型, 为循环索引。

(3) 功能

FOR 和 NEXT 指令循环执行它们之间的程序段, FOR-NEXT 循环的处理步骤如下。

1) 将 InitVal 的值赋给控制变量 Index。

2) 检查 Index 的值, 看它是否等于或大于 InitVal 并且等于或小于 EndVal。如果是, 则处理转步骤 3; 如果不是, 处理过程结束。如果不是的话, 循环处理结束后, 移动到 NEXT 指令 (ST 为 END_FOR 指令) 之后的下一条指令。

3) FOR 指令和 NEXT 指令 (ST 为 END_FOR 指令) 之间的处理指令执行一次。

4) Index 的值增加 StepVal。

5) 处理返回到步骤 2。

图 4-23 所示为 FOR/NEXT 指令的应用, InitVal 为 INT#0, EndVal 为 INT#9, StepVal 为 INT#1, MOVE 指令执行 10 次, 将 INT#0 赋值给数组变量 AryOut [0] ~ AryOut [9]。

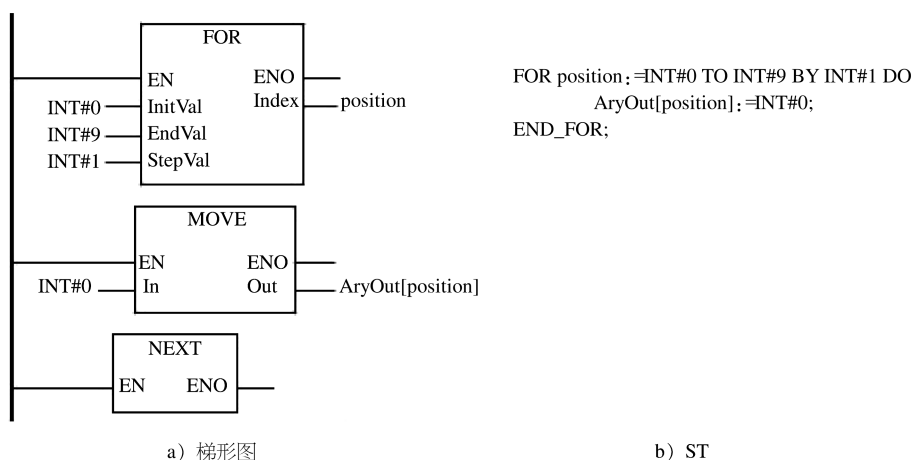


图 4-23 FOR/NEXT 指令的应用

6. 中断循环指令 BREAK

BREAK 指令的梯形图及 ST 如图 4-24 所示。

BREAK (EXIT) 指令中断从 FOR 指令到 NEXT 指令 (ST 为 END_FOR 指令) 的循环处理, 将处理转移到 NEXT 指令 (ST 为 END_FOR 指令) 之后的下一条指令, 而 BREAK 指令和 NEXT 指令之间的处理将不再执行。

图 4-25 所示为 BREAK 指令的应用。当 FOR 循环执行时, 每次检查变量 A 的值, 如果 A 的值是 TRUE, 循环处理立即结束。BREAK 指令后面的 Out 指令不执行, 变量 C 的值保持。在 ST 中, 使用 EXIT 指令, 取代 BREAK 指令。

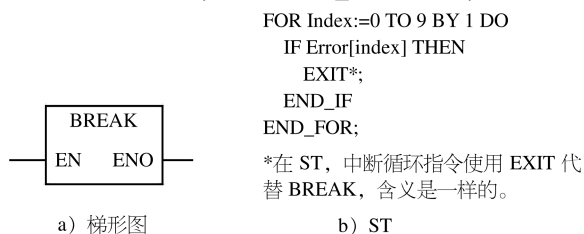


图 4-24 BREAK 指令的梯形图及 ST

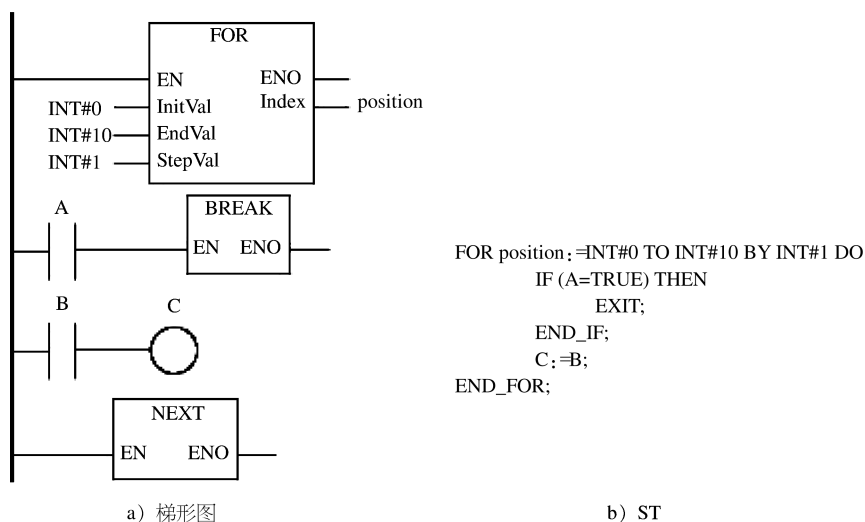


图 4-25 FOR/NEXT 指令的应用

4.6 定时器指令

定时器指令见表 4-16。

下面介绍常用的定时器指令。

1. 通电延时定时器指令 TON

(1) 指令的梯形图及 ST

TON 指令的梯形图及 ST 如图 4-26 所示。

表 4-16 定时器指令

指令	FB/FUN	名称
TON	FB	通电延时定时器
TOF	FB	断电延时定时器
TP	FB	定时器脉冲
Accumulation Timer	FB	累加定时器
Timer	FUN	100ms 定时器

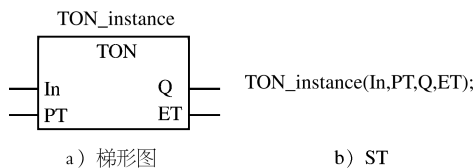


图 4-26 TON 指令的梯形图及 ST

(2) 变量

In: 定时器输入, 输入变量, BOOL 型。取值为 TRUE 时, 定时器启动; 取值为 FALSE 时, 定时器复位。

PT: 设定时间, 输入变量, TIME 型。指定定时器启动到 Q 变为 TRUE 的时间, 单位 ms。

Q: 定时器输出, 输出变量, BOOL 型。定时器输出 ON 时, 取值为 TRUE; 定时器输出 OFF 时, 取值为 FALSE。

ET: 过去时间, 输出变量, TIME 型。指定定时器启动后过去的时间, 单位 ms。

PT、ET 的取值范围都为 T#0ms ~ T#106751d_23h_47m_16s_854.775807ms。

(3) 功能

当定时器启动到达设定时间时, TON 指令输出 TRUE。以纳秒 (ns) 为单位设定时间, 定时精度为 100ns。当定时器输入 In 为 TRUE 时, 定时器启动。随着时间的推移, 过去时间 ET 增加。当 ET 到达设定时间 PT 时, 输出 Q 变为 TRUE, 之后 ET 不再增加。当 IN 变为 FALSE 时, 定时器复位, 即 ET 变为 0、Q 变为 FALSE。

定时器启动后, 在 ET 到达 PT 之前, 如果 IN 变为 FALSE, 则定时器复位。

图 4-27 所示为 TON 指令的应用, PT 设为 T#10ms, 在变量 A 变为 TRUE 10ms 后, 变量 abc 变为 TRUE。

2. 断电延时定时器指令 TOF

(1) 指令的梯形图及 ST

TOF 指令的梯形图及 ST 如图 4-28 所示。

(2) 变量

In: 定时器输入, 输入变量, BOOL 型。取值为 TRUE 时, 定时器复位; 取值为 FALSE 时, 定时器启动。

PT: 设定时间, 输入变量, TIME 型。指定定时器启动到 Q 变为 TRUE 的时间, 单位 ms。

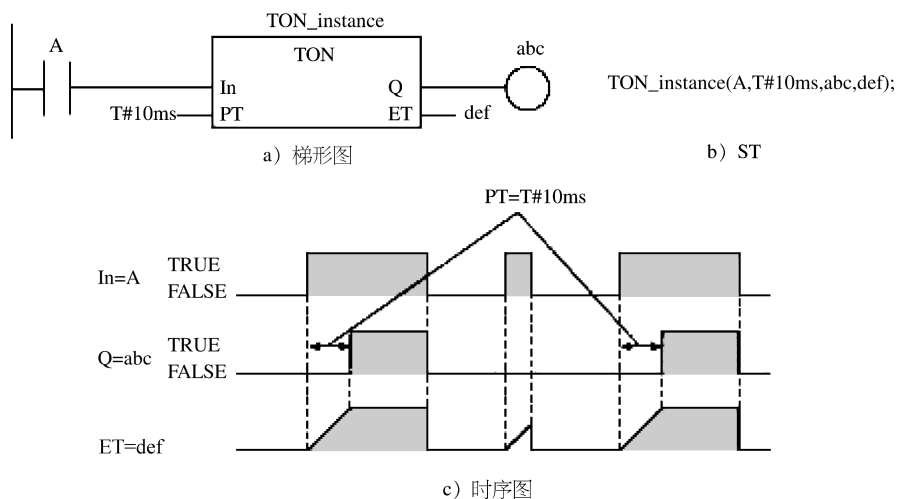


图 4-27 TON 指令的应用

Q: 定时器输出, 输出变量, BOOL 型。定时器输出 ON 时, 取值为 TRUE; 定时器输出 OFF 时, 取值为 FALSE。

ET: 过去时间, 输出变量, TIME 型。指定时器启动后过去的时间, 单位 ms。

PT、ET 的取值范围都为 T#0ms ~ T#106751d_23h_47m_16s_854.775807ms。

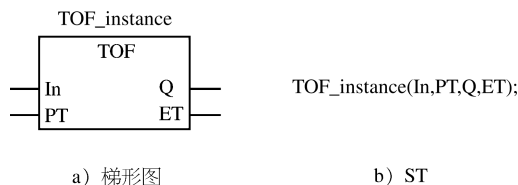


图 4-28 TOF 指令的梯形图及 ST

(3) 功能

当定时器启动到达设定时间时, TOF 指令输出 FALSE。以纳秒 (ns) 为单位设定时间, 定时精度为 100ns。当定时器输入 In 为 FALSE 时, 定时器启动。随着时间的推移, 过去时间 ET 增加。当 ET 到达设定时间 PT 时, 输出 Q 变为 FALSE, 之后 ET 不再增加。当 IN 变为 TRUE 时, 定时器复位, 则 ET 变为 0、Q 变为 TRUE。

定时器启动后, 在 ET 到达 PT 之前, 如果 IN 变为 TRUE, 则定时器复位。

图 4-29 所示为 TOF 指令的应用, PT 设为 T#10ms, 在变量 A 变为 FALSE 10ms 后, 变量 abc 变为 FALSE。

3. 定时器脉冲指令 TP

(1) 指令的梯形图及 ST

TP 指令的梯形图及 ST 如图 4-30 所示。

(2) 变量

In: 定时器输入, 输入变量, BOOL 型。取值为 TRUE 时, 定时器启动; 取值为 FALSE 时, 定时器复位。

PT: 设定时间, 输入变量, TIME 型。指 Q 保持 TRUE 的时间, 单位 ms。

Q: 定时器输出, 输出变量, BOOL 型。定时器输出 ON 时, 取值为 TRUE; 定时器输出 OFF 时, 取值为 FALSE。

ET: 过去时间, 输出变量, TIME 型。指定时器启动后过去的时间, 单位 ms。

PT、ET 的取值范围都为 T#0ms ~ T#106751d_23h_47m_16s_854.775807ms。

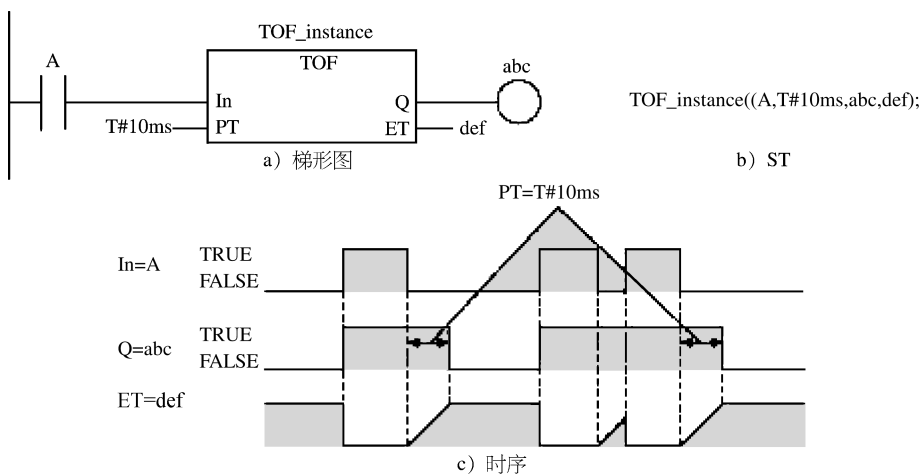


图 4-29 TOF 指令的应用

(3) 功能

当定时器启动到达设定时间之前，TP 指令输出 TRUE。以纳秒 (ns) 为单位设定时间，定时精度为 100ns。当定时器输入 In 为 TRUE 时，定时器启动，定时器输出 Q 变为 TRUE。随着时间的推移，过去时间 ET 增加。当 ET 到达设定时间 PT 时，输出 Q 变为 FALSE，之后 ET 不再增加。当 IN 变为 FALSE 时，定时器复位，则 ET 变为 0。定时器一旦启动，在 ET 到达 PT 之前，即使 IN 变为 FALSE，定时器也不复位。

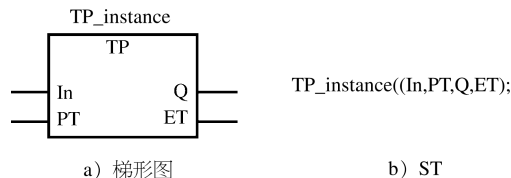


图 4-30 TP 指令的梯形图及 ST

图 4-31 所示为 TP 指令的应用，PT 设为 T#10ms，只要变量 A 变为 TRUE，变量 abc 就会变为 TRUE。过 10ms 后，变量 abc 变为 FALSE。

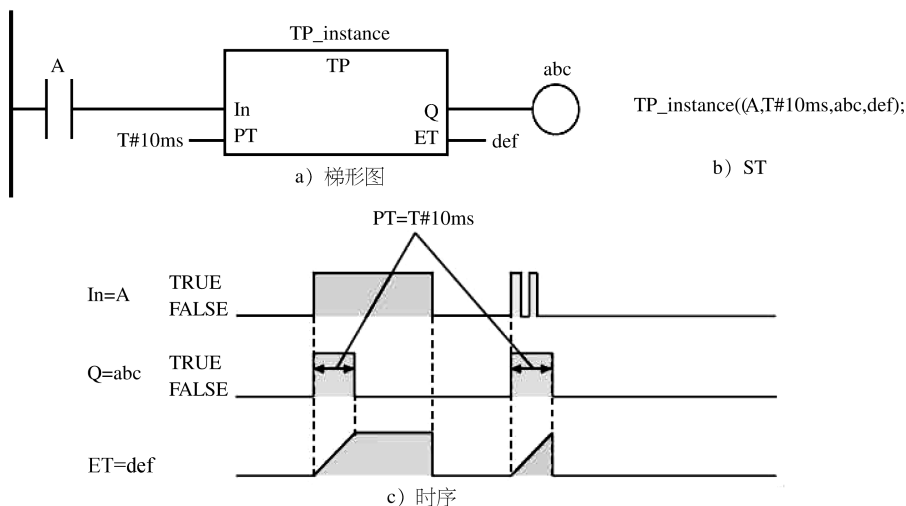


图 4-31 TP 指令的应用

4. 定时器指令 Timer

(1) 指令的梯形图及 ST

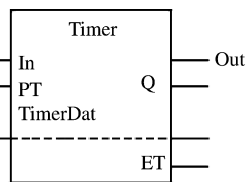
Timer 指令的梯形图及 ST 如图 4-32 所示。

(2) 变量

In: 定时器输入, 输入变量, BOOL 型。取值为 TRUE 时, 定时器启动; 取值为 FALSE 时, 定时器复位。

PT: 设定时间, 输入变量, UINT 型。指定定时器启动到 Q 变为 TRUE 的时间, 单位 ms。

TimerDat: 定时器状态, 输入-输出变量, Structure_sTimer 型, 表示定时器当前状态。



a) 梯形图

Out:=Timer(In,PT,TimerDat,Q,ET);

b) ST

图 4-32 Timer 指令的梯形图及 ST

Out: 返回值, 输出变量, BOOL 型。定时器输出 ON 时, 取值为 TRUE; 定时器输出 OFF 时, 取值为 FALSE。

Q: 定时器输出, 输出变量。其意义与 Out 相同。

ET: 剩余的时间, UINT 型, 单位 ms。

(3) 功能

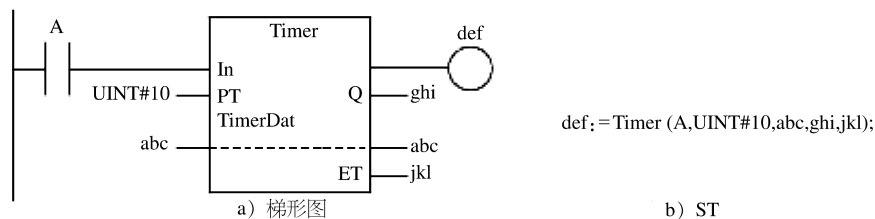
当定时器启动到达设定时间时, Timer 指令输出为 TRUE。设定时间以 100ms 为单位, 定时单位为 100ms。

当定时器输入 In 变为 FALSE 时, 定时器复位。此时, 剩余时间 ET 被设置为设定时间 PT, 定时器输出 Q 变为 FALSE。

当 In 变为 TRUE 时, 定时器启动。ET 的值随时间而减少, 当 ET 的值到达 0 时, 定时器输出 Q 变为 TRUE, 之后 ET 值不再随时间而减少。

定时器启动后, 在 ET 到达 0 之前, 如果 In 变为 FALSE, 则定时器复位。

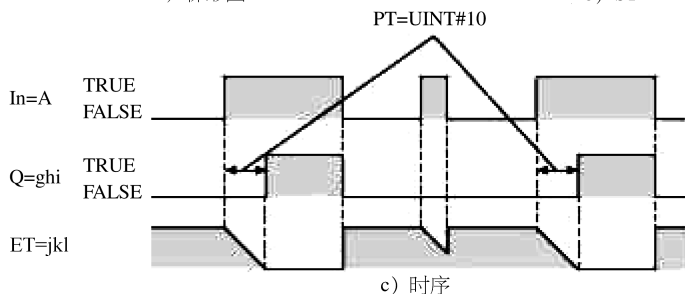
图 4-33 所示为 Timer 指令的应用, PT 设为 UNIT#10, 在变量 A 变为 TRUE 1000ms (1s) 后, 变量 ghi 会变为 FALSE。



a) 梯形图

def:=Timer (A,UINT#10,abc,ghi,jkl);

b) ST



c) 时序

图 4-33 Timer 指令的应用

4.7 计数器指令

计数器指令见表 4-17。

表 4-17 计数器指令

指令	FB/FUN	名 称	指令	FB/FUN	名 称
CTD	FB	递减计数器	CTU_*	FB	递增计数器类
CTD_*	FB	递减计数器类	CTUD	FB	递增-递减计数器
CTU	FB	递增计数器	CTUD_*	FB	递增-递减计数器类

下面介绍常用的计数器指令。

1. 递减计数器指令 CTD

(1) 指令的梯形图及 ST

CTD 指令的梯形图及 ST 如图 4-34 所示。

(2) 变量

CD: 计数器输入, 输入变量, BOOL 型, 默认值为 FALSE。

Load: 装载信号, 输入变量, BOOL 型, 默认值为 FALSE。取值为 TRUE 时, 将 PV 赋给 CV。

PV: 预设值, 输入变量, INT 型。计数器预设值, 取值范围为 0 ~ 32767, 默认值为 0。

Q: 计数器输出, 输出变量, BOOL 型。计数器输出 ON 时, 取值为 TRUE; 计数器输出 OFF 时, 取值为 FALSE。

CV: 计数器值, 输出变量, INT 型。计数器当前值, 取值范围为 0 ~ 32767。

(3) 功能

CTD 指令创建一个递减计数器。预设值和计数器值必须都是 INT 型的数据类型。

当装载信号 Load 变为 TRUE 时, 计数器值 CV 赋值为预设值 PV 并且计数器输出 Q 变为 FALSE。当计数器输入信号 CD 变为 TRUE 时, CV 递减。当 CV 值达到 0 或更小, Q 的值变为 TRUE。

当 CV 值达到 0 或以下, 即使 CD 变为 TRUE, CV 值也不变。

当 Load 为 TRUE 时, CD 被忽略, CV 不递减。

图 4-35 所示为 CTD 指令的应用, PV 设为 INT#5。

2. 递增计数器指令 CTU

(1) 指令的梯形图及 ST

CTU 指令的梯形图及 ST 如图 4-36 所示。

(2) 变量

CU: 计数器输入, 输入变量, BOOL 型, 默认值为 FALSE。

Reset: 复位信号, 输入变量, BOOL 型, 默认值为 FALSE。取值为 TRUE 时, CV 赋值为 0。

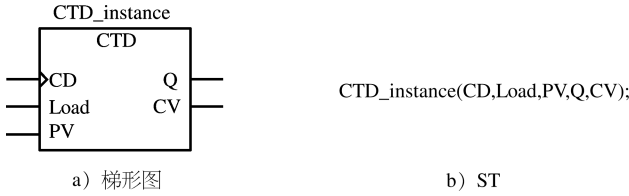


图 4-34 CTD 指令的梯形图及 ST

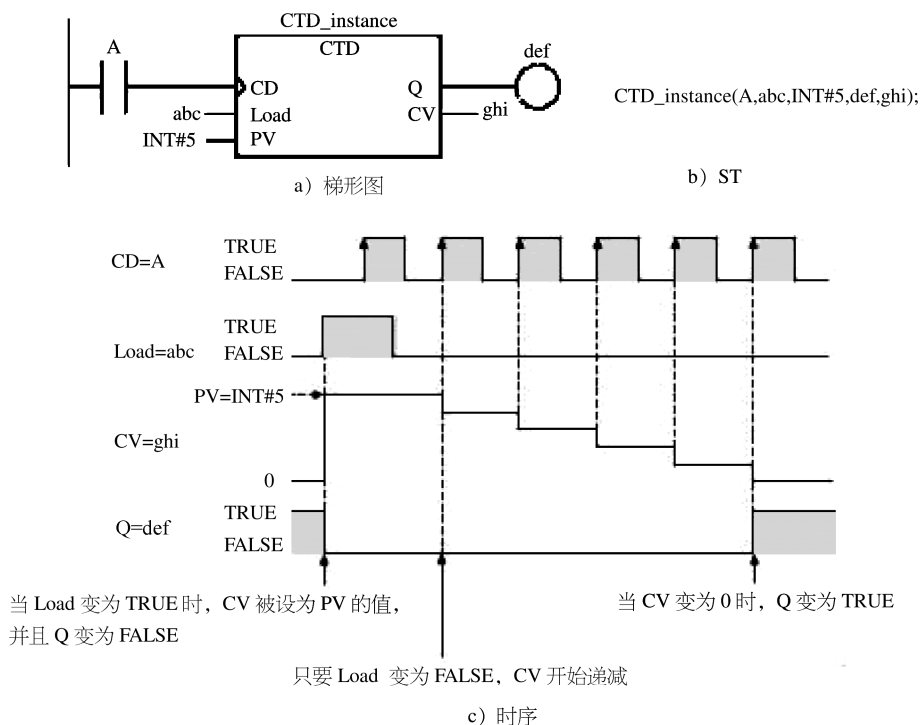


图 4-35 CTD 指令的应用

PV: 预设值, 输入变量, INT 型。计数器预设值, 取值范围为 0 ~ 32767, 默认值为 0。

Q: 计数器输出, 输出变量, BOOL 型。计数器输出 ON 时, 取值为 TRUE; 计数器输出 OFF 时, 取值为 FALSE。

CV: 计数器值, 输出变量, INT 型。计数器当前值, 取值范围为 0 ~ 32767。

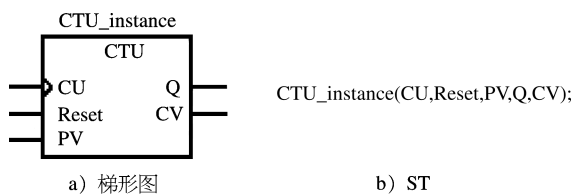


图 4-36 CTU 指令的梯形图及 ST

(3) 功能

CTU 指令创建一个递增计数器。计数器值和预设值必须都是 INT 型的数据类型。

当复位信号 Reset 变为 TRUE 时, 计数器值 CV 赋值为 0 并且计数器输出 Q 变为 FALSE。当计数器输入信号 CU 变为 TRUE 时, CV 递增。当 CV 值达到 PV 值或更大时, Q 的值变为 TRUE。

当 CV 值达到 PV 值或更大时, 即使 CU 变为 TRUE, CV 值也不变。

当 Reset 为 TRUE 时, CU 被忽略, CV 不递增。

图 4-37 所示为 CTU 指令的应用, PV 设为 INT#5。

3. 递增-递减计数器指令 CTUD

(1) 指令的梯形图及 ST

CTUD 指令的梯形图及 ST 如图 4-38 所示。

(2) 变量

CU: 递增计数器输入, 输入变量, BOOL 型, 默认值为 FALSE。

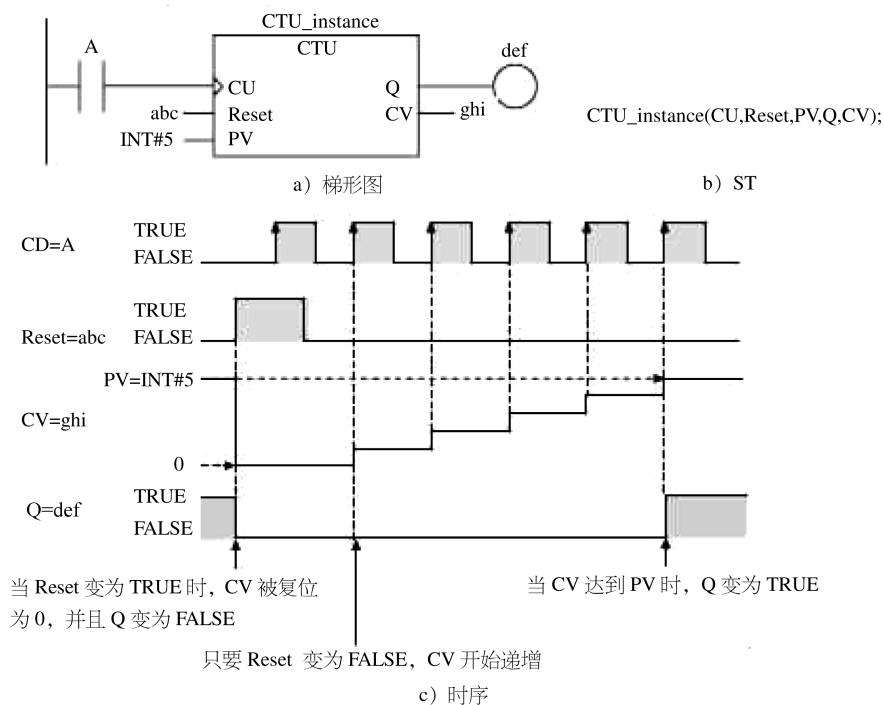


图 4-37 CTU 指令的应用

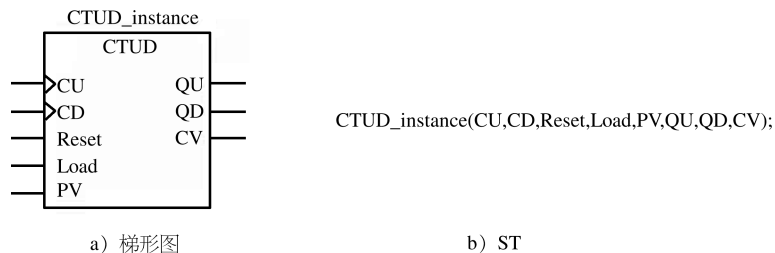


图 4-38 CTUD 指令的梯形图及 ST

CD: 递减计数器输入, 输入变量, BOOL 型, 默认值为 FALSE。

Reset: 复位信号, 输入变量, BOOL 型, 默认值为 FALSE。取值为 TRUE 时, CV 赋值为 0。

Load: 装载信号, 输入变量, BOOL 型, 默认值为 FALSE。取值为 TRUE 时, 将 PV 赋给 CV。

PV: 预设值, 输入变量, INT 型。作为递增计数器时, PV 为计数器最终值; 作为递减计数器时, PV 为计数器初始值。取值范围为 0 ~ 32767, 默认值为 0。

QU: 递增计数器输出, 输出变量, BOOL 型。递增计数器输出 ON 时, 取值为 TRUE; 递增计数器输出 OFF 时, 取值为 FALSE。

QD: 递减计数器输出, 输出变量, BOOL 型。递减计数器输出 ON 时, 取值为 TRUE; 递减计数器输出 OFF 时, 取值为 FALSE。

CV: 计数器值, 输出变量, INT 型。计数器当前值, 取值范围为 0 ~ 32767。

(3) 功能

CTUD 指令创建一个递增 - 递减计数器，其操作依据递增计数器输入信号和递减计数器输入信号。既有递增计数器功能也有递减计数器功能。预设值和计数器值必须同属 INT 数据类型。

1) 作为递增计数器的操作

当复位信号 Reset 变为 TRUE 时，计数器值 CV 变为 0，并且递增计数器输出 QU 变为 FALSE。当递增计数器输入信号 CU 变为 TRUE 时，CV 递增。当 CV 的值达到 PV 的值或者更高，QU 的值变为 TRUE。CV 的值达到 PV 的值或者更大之后，CV 的值不再改变即使 CU 的值变为 TRUE。

2) 作为递减计数器的操作

当装载信号 Load 变为 TRUE 时，计数器值 CV 变为设定值 PV 的值，并且递减计数器输出 QD 变为 FALSE。当递减计数器输入信号 CD 变为 TRUE 时，CV 递减。当 CV 的值达到 0 或者更低，QD 的值变为 TRUE。当 CV 的值达到 0 或者更小之后，CV 的值不再改变，即使 CD 的值变为 TRUE。

3) 递增计数器和递减计数器的公共操作

当 Load 和 Reset 信号为 TRUE 时，CU 和 CD 被忽略。CV 不递增或递减。如果 CU 和 CD 同时都变为 TRUE，CV 将不变。如果 Reset 和 Load 都为 TRUE，Reset 具有优先权并且 CV 值变为 0。如果 Reset 变为 TRUE，CV 变为 0，并且因此 QD 变为 TRUE。如果 Load 变为 TRUE，CV 变为 PV，并且因此 QU 变为 TRUE。

Reset、Load、CV、QU 和 QD 对应的操作见表 4-18，假定 PV 的值大于 0。

表 4-18 Reset、Load、CV、QU 和 QD 对应的操作

Reset	Load	CV	QU	QD	操作
FALSE	FALSE	0 或更小	FALSE	TRUE	只能完成递增计数器的操作 ● 当 CU 变为 TRUE 时，CV 递增。当 CD 变为 TRUE 时，不递减
		0 到 PV 之间	FALSE	FALSE	递增和递减计数器都能进行操作 ● 当 CU 变为 TRUE 时，CV 递增；当 CD 变为 TRUE 时，CV 递减
		PV 或更大	TRUE	FALSE	只能完成递减计数器的操作 ● 当 CD 变为 TRUE 时，CV 递减；当 CU 变为 TRUE 时，CV 不递增
TRUE	FALSE	0	FALSE	TRUE	递增计数器复位 ● CV 的值被设为 0
FALSE	TRUE	PV	TRUE	FALSE	递减计数器复位 ● CV 的值被设为 PV
TRUE	TRUE	0	FALSE	TRUE	递增计数器复位，Reset 优先于 Load ● CV 的值被设为 0

即使将 PV 设置为负值，当 Load 值变为 TRUE 时，CV 被设置为 PV 的值。CV 的值将是 0 或更小，因此 QD 立即变为 TRUE。此后，即使 CD 的值变化，CV 值不递减。

PV 设置为负值时，当 Reset 值变为 TRUE 时，CV 的值变为 0。CV 的值比 PV 的值大，

因此 QU 立即变为 TRUE。此后，即使 CU 的值变化，CV 值不递增。

在指令的执行过程中，可以更改 PV 值。如果新的 PV 值小于当前 CV 的值，QU 的值立即变为 TRUE。

如果 CU 或 CD 的值是 FALSE，并且电源中断或操作模式改为编程模式，在指令重新执行时，CU 或 CD 的值若为 TRUE，则 CV 值递增或递减一次。

图 4-39 所示为 CTUD 指令的应用。

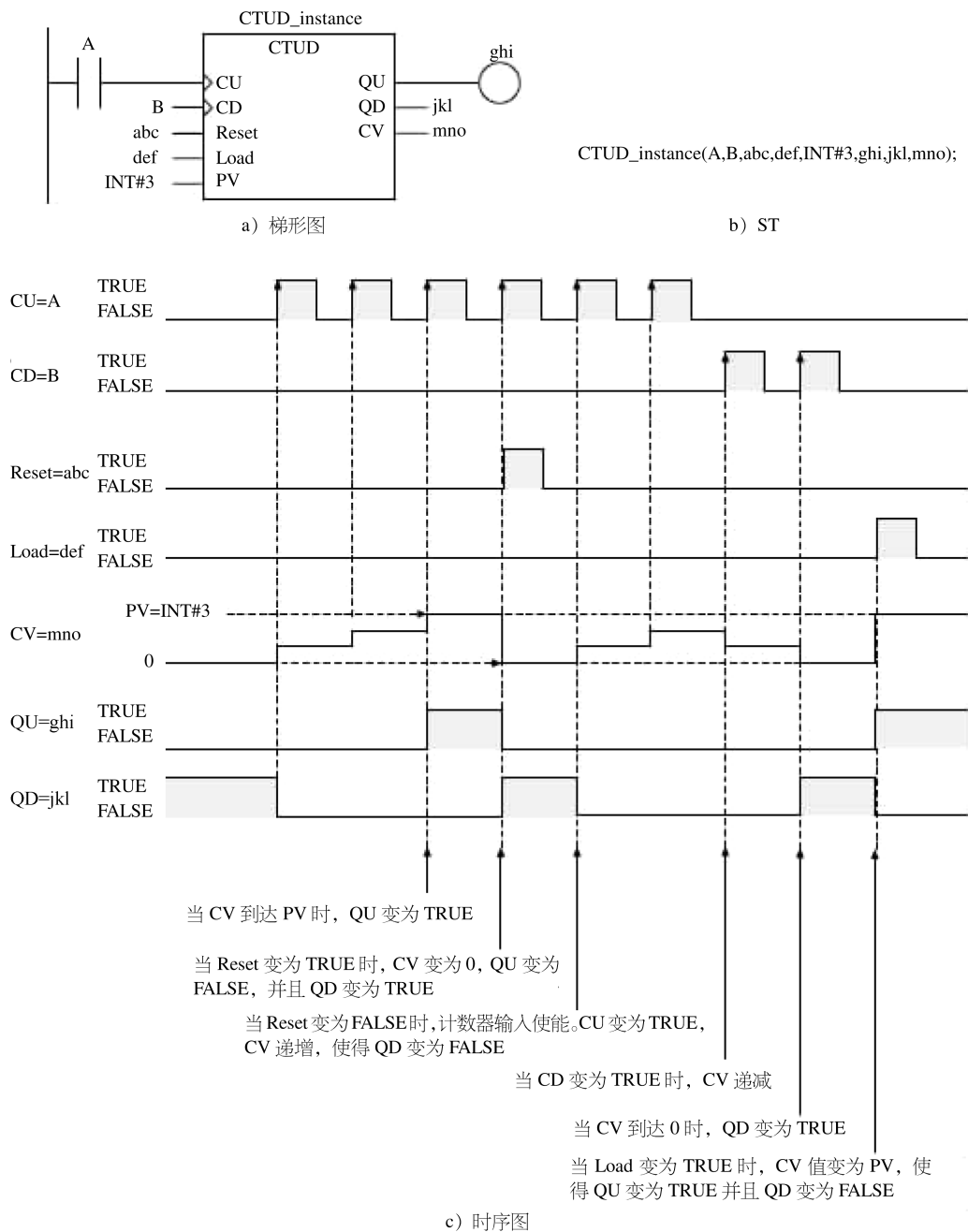


图 4-39 CTUD 指令的应用

4.8 比较指令

比较指令见表 4-19。

表 4-19 比较指令

指 令	FB/FUN	名 称
EQ(=)	FUN	相等
NE(< >)	FUN	不相等
LT(<), LE(<=), GT(>)和 GE(>=)	FUN	小于/小于等于/大于/大于等于
EQascii	FUN	字符串相等比较
NEascii	FUN	字符串不相等比较
LTascii, LEascii, GTascii 和 GEascii	FUN	字符串小于/小于等于/大于/大于等于比较
Cmp	FUN	比较
ZoneCmp	FUN	区域比较
TableCmp	FUN	表比较
AryCmpEQ 和 AryCmpNE	FUN	数组相等/不相等比较
AryCmpLT, AryCmpLE, AryCmpGT 和 AryCmpGE	FUN	数组小于/小于等于/大于/大于等于比较
AryCmpEQV 和 AryCmpNEV	FUN	数组值相等/不相等比较
AryCmpLTV, AryCmpLEV, AryCmpGTV 和 AryCmpGEV	FUN	数组值小于/小于等于/大于/大于等于比较

下面介绍常用的数据比较指令。

1. 相等比较指令 EQ (=)

(1) 指令的梯形图及 ST

EQ (=) 指令的梯形图及 ST 如图 4-40 所示。

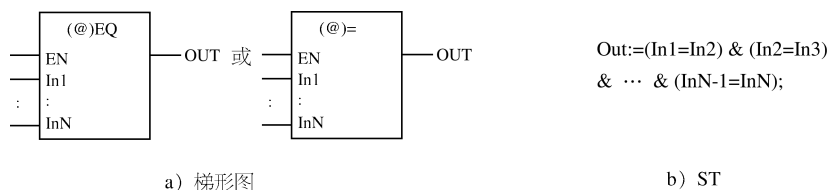


图 4-40 EQ (=) 指令的梯形图及 ST

(2) 变量

In1 ~ InN: 比较数据, 输入变量, 比较变量个数 $N=2 \sim 5$ 。输入变量可为 BOOL、位串、整数、实数、枚举等数据类型。

Out: 比较结果, 输出变量, BOOL 型。

(3) 功能

EQ (=) 指令确定 In1 ~ InN 这 2 ~ 5 个变量的内容是否相等。只有当所有的值相等时, 比较结果 Out 为 TRUE, 否则, Out 为 FALSE。

图 4-41 所示为 EQ (=) 指令的应用, In1 为 INT#3, In2 为 INT#5, In3 为 INT#10, 变量

abc 的值为 FALSE。

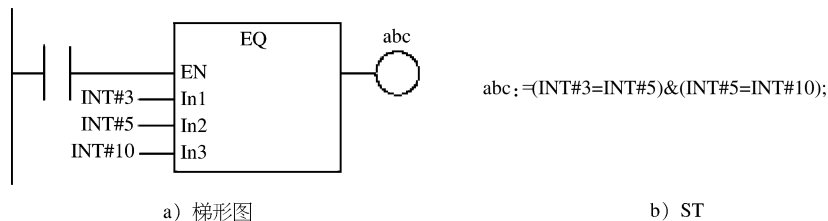


图 4-41 EQ (=) 指令的应用

2. 不相等比较指令 NE (<>)

(1) 指令的梯形图及 ST

NE (<>) 指令的梯形图及 ST 如图 4-42 所示。

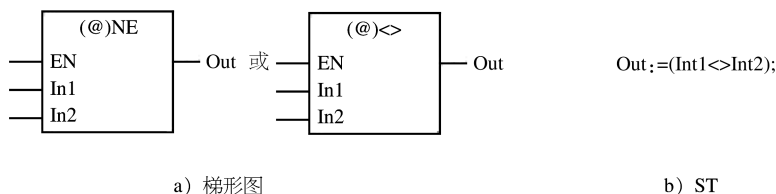


图 4-42 NE (<>) 指令的梯形图及 ST

(2) 变量

In1 和 In2: 比较数据, 输入变量, 可为 BOOL、位串、整数、实数、枚举等数据类型。

Out: 比较结果, 输出变量, BOOL 型。

(3) 功能

NE (<>) 指令确定两个变量 In1 和 In2 的内容是否相等。如果它们不相等时, 比较结果 Out 为 TRUE; 如果它们相等, Out 就是 FALSE。

图 4-43 所示为 NE 指令的应用, In1 等于 In2, 都是 INT#5, 变量 abc 的值为 FALSE。

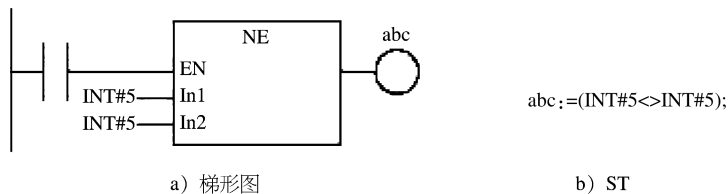


图 4-43 NE 指令的应用

3. LT (<)、LE (<=)、GT (>) 和 GE (>=) 指令

LT (<)、LE (<=)、GT (>) 和 GE (>=) 指令见表 4-20。

(1) 变量

In1 ~ InN: 比较数据, 输入变量, 比较变量个数 N=2~5, 输入变量为整数、实数, 默认值为 0。

Out: 返回值, 输出变量, BOOL 型。

表 4-20 LT (<)、LE (<=)、GT (>) 和 GE (>=) 指令

指令	名称	梯形图	ST
LT(<)	小于		$\text{Out} := (\text{In1} < \text{In2}) \& (\text{In2} < \text{In3}) \\ \& \dots \& (\text{InN} - 1 < \text{InN});$
LE(<=)	小于或等于		$\text{Out} := (\text{In1} <= \text{In2}) \& \\ (\text{In2} <= \text{In3}) \& \dots \& \\ (\text{InN} - 1 <= \text{InN});$
GT(>)	大于		$\text{Out} := (\text{In1} > \text{In2}) \& (\text{In2} > \text{In3}) \\ \& \dots \& (\text{InN} - 1 > \text{InN});$
GE(>=)	大于或等于		$\text{Out} := (\text{In1} >= \text{In2}) \& \\ (\text{In2} >= \text{In3}) \& \dots \& \\ (\text{InN} - 1 >= \text{InN});$

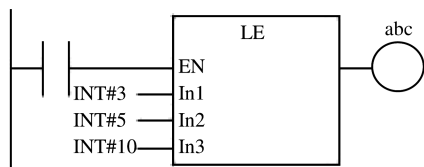
(2) 功能

这些指令执行时, 对 In1 ~ InN (N=2~5) 的值进行比较, 比较结果 Out 见表 4-21。

表 4-21 比较结果 Out

指令	Out 值
LT (<)	如果 $\text{In1} < \text{In2} < \dots < \text{InN}$, Out 为 TRUE; 否则, Out 为 FALSE
LE (<=)	如果 $\text{In1} <= \text{In2} <= \dots <= \text{InN}$, Out 为 TRUE; 否则, Out 为 FALSE
GT (>)	如果 $\text{In1} > \text{In2} > \dots > \text{InN}$, Out 为 TRUE; 否则, Out 为 FALSE
GE (>=)	如果 $\text{In1} >= \text{In2} >= \dots >= \text{InN}$, Out 为 TRUE; 否则, Out 为 FALSE

图 4-44 为 LE 指令的应用, In1 为 INT#3, In2 为 INT#5, In3 为 INT#10, 变量 abc 的值为 TRUE。



a) 梯形图

$$\text{abc} := (\text{INT}\#3 <= \text{INT}\#5) \& (\text{INT}\#5 <= \text{INT}\#10);$$

b) ST

图 4-44 LE 指令的应用

3. 比较指令 Cmp

(1) 指令的梯形图及 ST

Cmp 指令的梯形图及 ST 如图 4-45 所示。

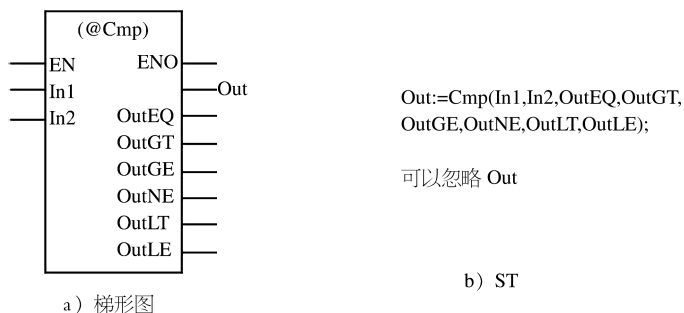


图 4-45 CMP 指令的梯形图及 ST

(2) 变量

In1 和 In2: 比较数据, 输入变量, 为整数或实数。

Out: 返回值, 输出变量, 总为 TRUE。

OutEQ: 等于标志, 输出变量, BOOL 型。

OutGT: 大于标志, 输出变量, BOOL 型。

OutGE: 大于等于标志, 输出变量, BOOL 型。

OutNE: 不等于标志, 输出变量, BOOL 型。

OutLT: 小于标志, 输出变量, BOOL 型。

OutLE: 小于等于标志, 输出变量, BOOL 型。

(3) 功能

Cmp 指令比较 In1 和 In2 的值, 输出各个标志的值。

标志的值见表 4-22。

表 4-22 标志值

标志	值
OutEQ	如果 In1 等于 In2, 为 TRUE, 否则为 FALSE
OutGT	如果 In1 大于 In2, 为 TRUE, 否则为 FALSE
OutGE	如果 In1 大于等于 In2, 为 TRUE, 否则为 FALSE
OutNE	如果 In1 不等于 In2, 为 TRUE, 否则为 FALSE
OutLT	如果 In1 小于 In2, 为 TRUE, 否则为 FALSE
OutLE	如果 In1 小于等于 In2, 为 TRUE, 否则为 FALSE

图 4-46 所示为 Cmp 指令的应用, In1 为 INT#10, In2 为 INT#20, 变量 def、ghi 和 jkl 的值都为 FALSE, abc、mno、pdr 和 stu 的值都为 TRUE。

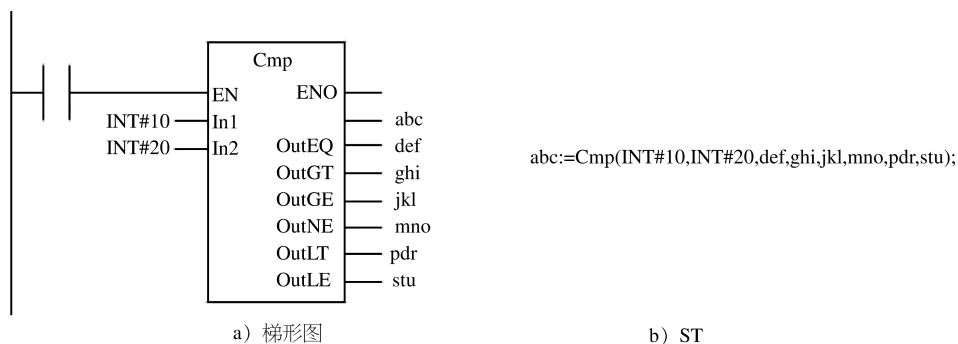


图 4-46 Cmp 指令的应用

4.9 运算指令

运算指令见表 4-23 所示。

表 4-23 运算指令

指 令	名 称	指 令	名 称
ADD(+)	加法	LN 和 LOG	自然对数/以 10 为底的对数
AddOU(+ OU)	带上溢或下溢检测的加法	EXP	自然指数运算
SUB(-)	减法	EXPT(* *)	求幂
SubOU(- OU)	带上溢或下溢检测的减法	Inc 和 Dec	递增/递减
MUL(*)	乘法	Rand	随机数
MulOU(* OU)	带上溢或下溢检测的乘法	AryAdd	数组加法
DIV(/)	除法	AryAddV	数组值加法
MOD	模除	ArySub	数组减法
ABS	绝对值	ArySubV	数组值减法
RadToDeg 和 DegToRad	弧度转换成角度/角度转换成弧度	AryMean	数组平均值
SIN、COS 和 TAN	弧度的正弦/余弦/正切	ArySD	数组元素标准方差
ASIN、ACOS 和 ATAN	反正弦/反余弦/反正切	ModReal	实数模除
SQRT	平方根	Fraction	实数小数部分
		CheckReal	实数检测

运算指令只有随机数 Rand 为功能块，其余指令都属功能。下面介绍常用的运算指令。

1. 加法指令 ADD (+)

(1) 指令的梯形图及 ST

ADD (+) 指令的梯形图及 ST 如图 4-47 所示，ADD 指令和 + 指令的功能相同。

(2) 变量

In1 ~ InN: 加数，输入变量，N 为加数个数 (N = 2 ~ 5)，加数为整数、实数或字符串，默认值为 0。

Out: 加法结果，输出变量，为整数、实数或字符串。

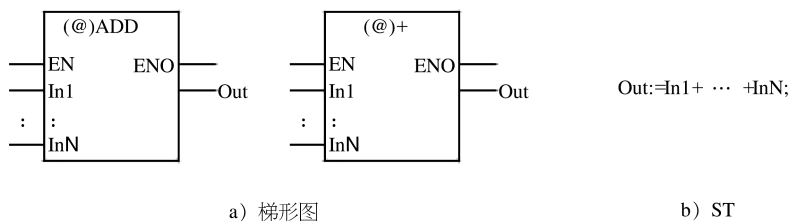


图 4-47 ADD (+) 指令的梯形图及 ST

(3) 功能

ADD (+) 指令将整数、实数相加，或将字符串连接，加数 2 ~ 5 个。加数 In1 ~ InN 和 Out 可以有不同的数据类型。

图 4-48 所示为 ADD 指令的应用，In1、In2、In3 分别是 INT#10、INT#20、INT#30，变量 abc 的值是 INT#60。

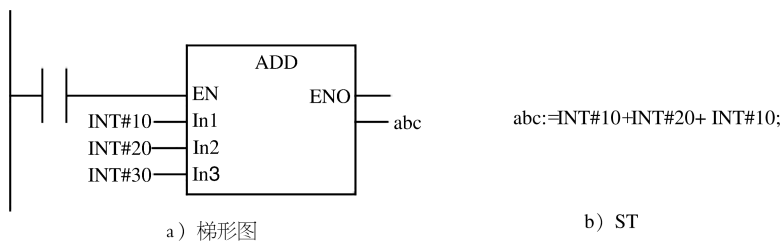


图 4-48 ADD 指令的应用

如果 In1 ~ InN 为字符串数据，则实现字符串相连。图 4-49 所示为字符串相加，In1、In2、In3 分别是 'UV'、'WX'、'YZ'，则 abc 的值是 'UVWXYZ'。

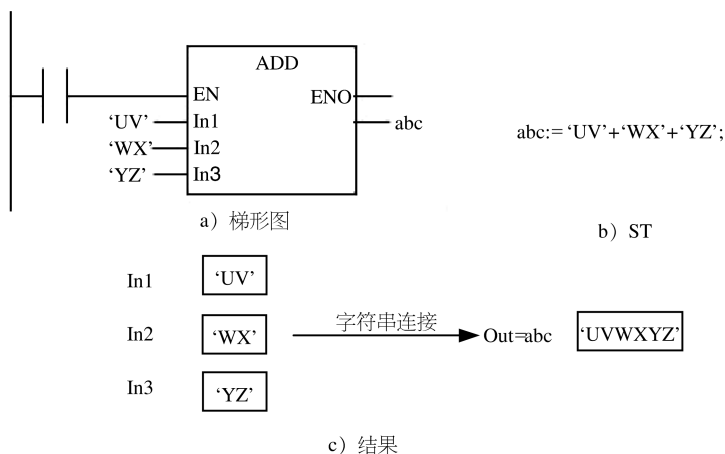


图 4-49 文本串相加

2. 减法指令 SUB (-)

(1) 指令的梯形图及 ST

SUB (-) 指令的梯形图及 ST 如图 4-50 所示，SUB 指令和 - 指令的功能相同。

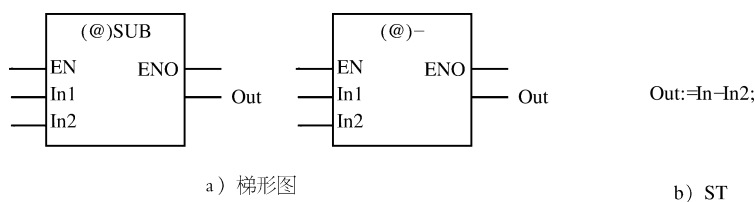


图 4-50 SUB (-) 指令的梯形图及 ST

(2) 变量

In1: 被减数, 输入变量, 默认值为 0。

In2: 减数, 输入变量, 默认值为 0。

Out: 减法结果, 输出变量。

In1、In2、Out 为整数或实数。

(3) 功能

SUB (-) 指令是从被减数 In1 减去减数 In2, 并输出该结果。In1、In2 的数据类型和减法结果 Out 可以有不同的数据类型。

图 4-51 所示为 SUB 指令的应用, In1、In2 分别为 INT#50、INT#10, 变量 abc 的值为 INT #40。

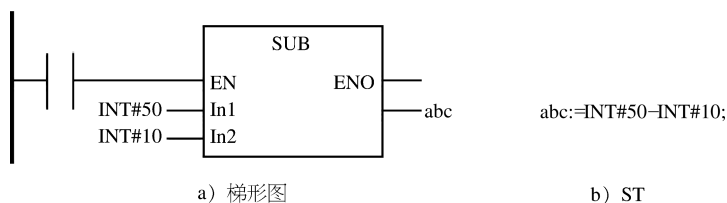


图 4-51 SUB 指令的应用

3. 乘法指令 MUL (*)

(1) 指令的梯形图及 ST

MUL (*) 指令的梯形图及 ST 如图 4-52 所示, MUL 指令和 * 指令的功能相同。

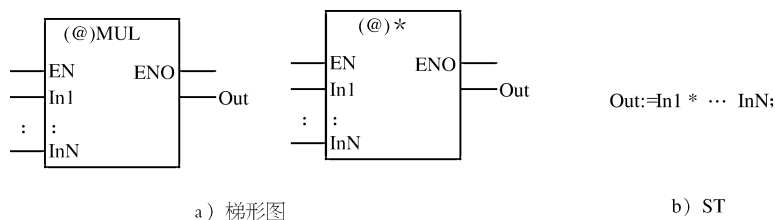


图 4-52 MUL (*) 指令的梯形图及 ST

(2) 变量

In1 ~ InN: 乘数, 输入变量, N 为乘数个数 (N = 2 ~ 5), 乘数为整数或实数, 默认值为 1。

Out: 乘法结果, 输出变量, 为整数或实数。

(3) 功能

MUL (*) 指令将 2~5 个整数和实数相乘, 输出结果。乘数 In1 ~ InN 和乘法结果 Out 可以有不同的数据类型。

图 4-53 所示为 MUL 指令的应用, In1、In2、In3 分别为 INT#10、INT#20、INT#30, 变量 abc 的值为 INT#6000。

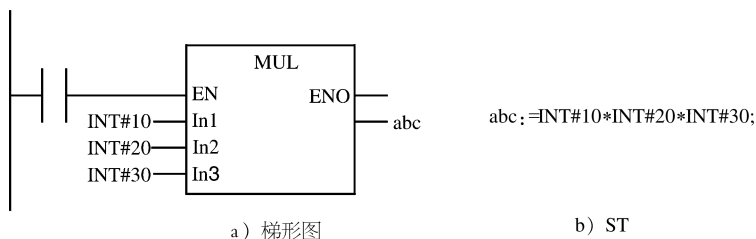


图 4-53 MUL 指令的应用

4. 除法指令 DIV (/)

(1) 指令的梯形图及 ST

DIV (/) 指令的梯形图及 ST 如图 4-54 所示, DIV 指令和/指令的功能相同。

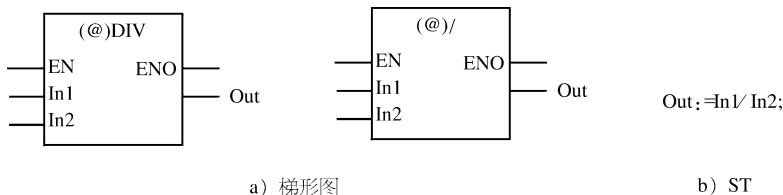


图 4-54 DIV (/) 指令的梯形图及 ST

(2) 变量

In1: 被除数, 输入变量。

In2: 除数, 输入变量。

Out: 除法结果, 输出变量。

In1、In2、Out 为整数或实数。

(3) 功能

DIV (/) 指令将被除数 In1 除以除数 In2, 并输出结果。In1、In2 和商 Out 可以有不同的数据类型。如果 In1、In2 和 Out 是整数, 则有余数, 余数被去掉。

图 4-55 所示为 DIV 指令的应用, In1、In2 分别为 INT#100、INT#5, 变量 abc 的值为 INT#20。

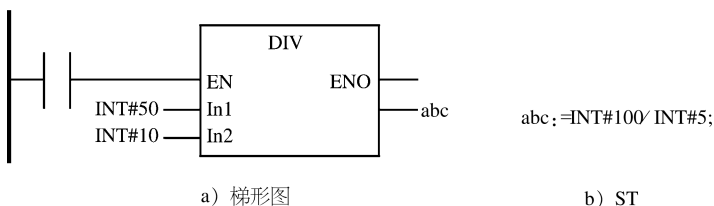


图 4-55 DIV 指令的应用

4.10 BCD 转换指令

BCD 转换指令见表 4-24。

表 4-24 BCD 转换指令

指令	FB/FUN	名称
_BCD_TO_*	FUN	BCD→无符号整数转换类
_TO_BCD_*	FUN	无符号整数→BCD 转换类
BCD_TO_**	FUN	BCD 数据类型→无符号整数转换类
BCDsToBin	FUN	有符号 BCD→有符号整数转换
BinToBCDs_**	FUN	有符号整数→有符号 BCD 转换类
AryToBCD	FUN	数组→BCD 转换
AryToBin	FUN	数组→无符号整数转换

下面介绍常用的数据比较指令。

1. BCD→无符号整数转换类指令 **_BCD_TO_

(1) 指令的梯形图及 ST

**_BCD_TO_指令的梯形图及 ST 如图 4-56 所示。

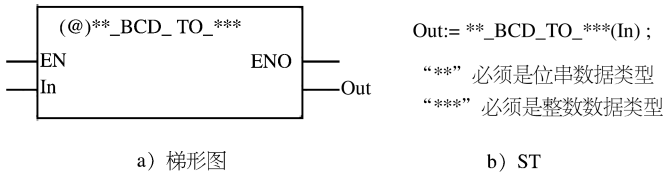


图 4-56 **_BCD_TO_*** 指令的梯形图及 ST

(2) 变量

In：被转换的数据，输入变量，位串型，默认值为 0。

Out：转换结果，输出变量，整数型。

(3) 功能

这些指令执行时，将数据 In (必须是 BCD 位串) 转换成一个无符号整数。指令的名称由 In 和转换结果 Out 的数据类型决定。例如，如果 In 是 WORD 数据、Out 是 UINT 数据，指令的名称是 WORD_BCD_TO_UINT。

图 4-57 所示为 WORD_BCD_TO_UINT 指令的应用，In 是 WORD#16#3452。

2. 无符号整数→BCD 转换类指令 **_TO_BCD_***

(1) 指令的梯形图及 ST

_TO_BCD_* 指令的梯形图及 ST 如图 4-58 所示。

(2) 变量

In：被转换的数据，输入变量，整数型，默认值为 0。

Out：转换结果，输出变量，位串型。

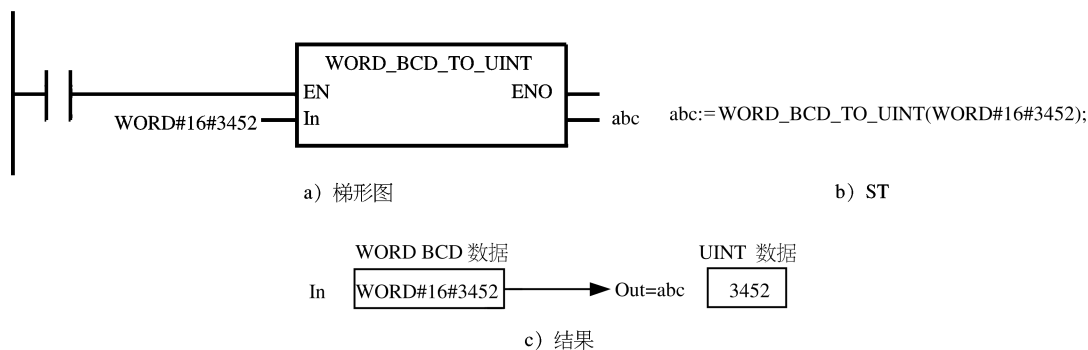


图 4-57 WORD_BCD_TO_UINT 指令的应用

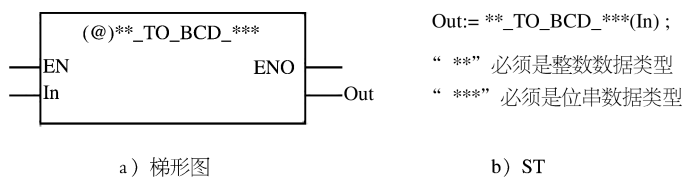


图 4-58 **_TO_BCD_*** 指令的梯形图及 ST

(3) 功能

这些指令执行时，将数据 In (必须是无符号整数) 转换成一个 BCD 位串。指令的名称由 In 和转换结果 Out 的数据类型决定。例如，如果 In 是 UINT 数据、Out 是 WORD 数据，则指令的名称是 UINT_BCD_TO_WORD。

图 4-59 所示为 UINT_BCD_TO_WORD 指令的应用，In 是 WORD#16#3452。

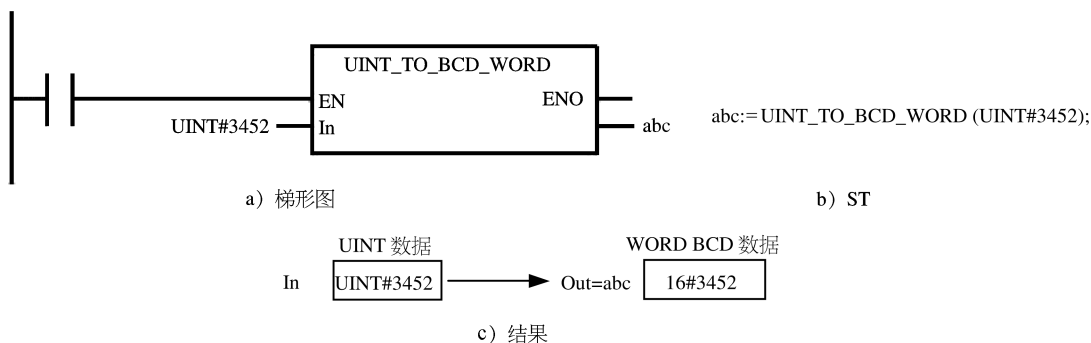


图 4-59 UINT_TO_BCD_WORD 指令的应用

3. BCD 数据类型→无符号整数转换类指令 BCD_TO_**

(1) 指令的梯形图及 ST

BCD_TO_** 指令的梯形图及 ST 如图 4-60 所示。

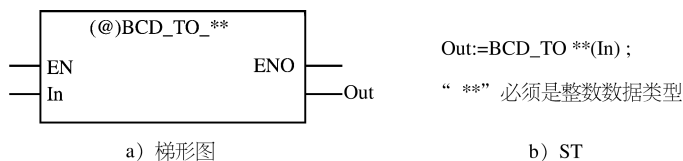


图 4-60 BCD_TO_** 指令的梯形图及 ST

(2) 变量

In: 被转换的数据, 输入变量, 位串型。

Out: 转换结果, 输出变量, 整数型。

(3) 功能

这些指令执行时, 将数据 In (必须是 BCD 位串) 转换成一个无符号整数。指令的名称由转换结果 Out 的数据类型决定。例如, 如果 Out 是 UINT 数据类型, 则指令的名称是 BCD_TO_UINT。

图 4-61 所示为 BCD_TO_UINT 指令的应用, In 是 WORD#16#3452。

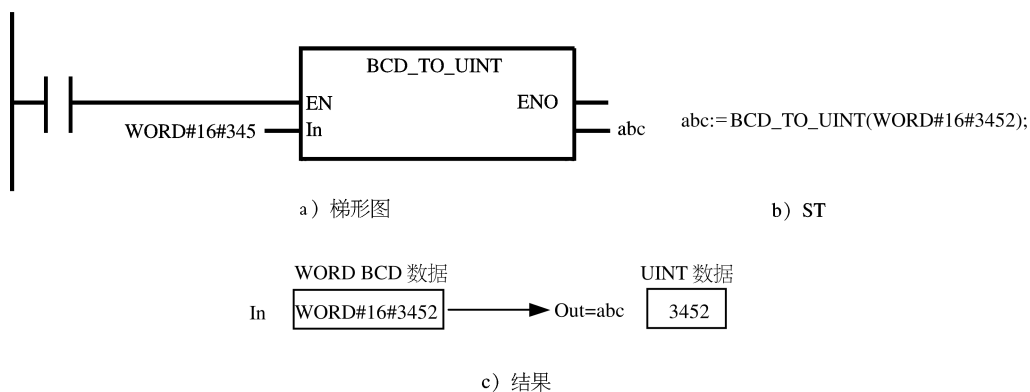


图 4-61 BCD_TO_UINT 指令的应用

4.11 数据类型转换指令

数据类型转换指令见表 4-25。

表 4-25 数据类型转换指令

指 令	名 称
TO* (Integer-to-Integer Conversion Group)	整数→整数转换类
TO* (Integer-to-Bit String Conversion Group)	整数→位串转换类
TO* (Integer-to-Real Number Conversion Group)	整数→实数转换类
TO* (Bit String-to-Integer Conversion Group)	位串→整数转换类
TO* (Bit String-to-Bit String Conversion Group)	位串→位串转换类
TO* (Bit String-to-Real Number Conversion Group)	位串→实数转换类
TO* (Real Number-to-Integer Conversion Group)	实数→整数转换类
TO* (Real Number-to-Bit String Conversion Group)	实数→位串转换类
TO* (Real Number-to-Real Number Conversion Group)	实数→实数转换类
**_TO_STRING(Integer-to-Text String Conversion Group)	整数→字符串转换类
**_TO_STRING(Bit String-to-Text String Conversion Group)	位串→字符串转换类
**_TO_STRING(Real Number-to-Text String Conversion Group)	实数→字符串转换类
RealToFormatString	实数→格式字符串
LrealToFormatString	长实型→格式字符串

(续)

指 令	名 称
STRING_TO_** (TextString-to-Integer Conversion Group)	字符串→整数转换类
STRING_TO_** (TextString-to-Bit String Conversion Group)	字符串→位串转换类
STRING_TO_** (TextString-to-Real Number Conversion Group)	字符串→实数转换类
TO_** (Integer Conversion Group)	整数转换类
TO_** (Bit String Conversion Group)	位串转换类
TO_** (Real Number Conversion Group)	实数转换类
TRUNC、Round 和 RoundUp	舍去小数/小数四舍五入/小数进位

下面介绍常用的数据比较指令。

1. 整数→实数转换类指令**_TO_*** (Integer-to-Real Number Conversion Group)

(1) 指令的梯形图及 ST

TO* (Integer-to-Real Number Conversion Group) 指令的梯形图及 ST 如图 4-62 所示。

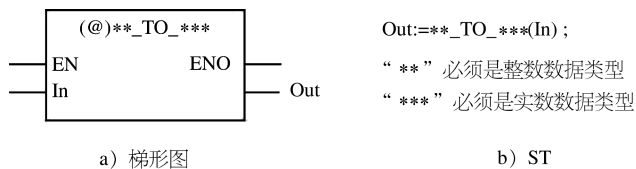


图 4-62 **_TO_*** (Integer-to-Real Number Conversion Group) 指令的梯形图及 ST

(2) 变量

In: 被转换的数据, 输入变量, 整数型, 默认值为 0。

Out: 转换结果, 输出变量, 实数型。

(3) 功能

这些指令执行时, 将整数 In 转换为一个实数。指令的名称由 In 和转换结果 Out 的数据类型决定。例如, 如果 In 是 INT 数据、Out 是 REAL 数据, 则指令的名称是 INT_TO_REAL。

图 4-63 所示为 INT_TO_REAL 指令的应用, In 是 INT#1234。

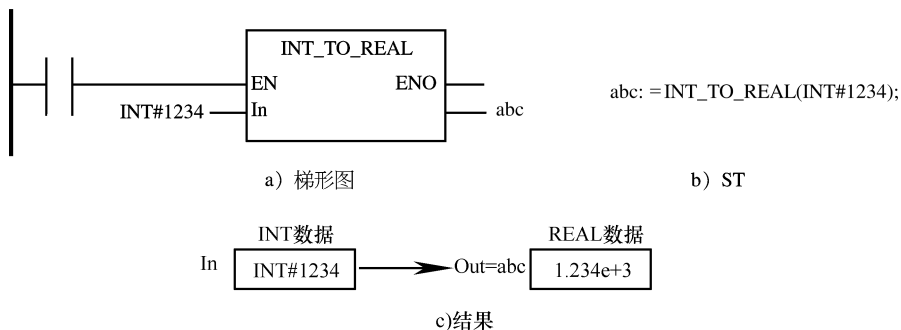


图 4-63 INT_TO_REAL 指令的应用

2. 实数→整数转换类指令* *_TO_*_*_* (Real Number-to-Integer Conversion Group)

(1) 指令的梯形图及 ST

* *_TO_*_*_* (Real Number-to-Integer Conversion Group) 指令的梯形图及 ST 如图 4-64 所示。

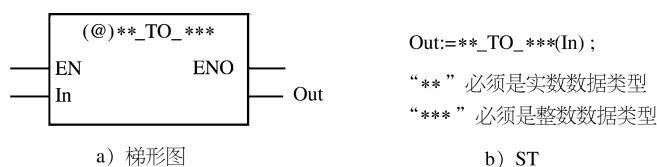


图 4-64 * *_TO_*_*_* (Real Number-to-Integer Conversion Group) 指令的梯形图及 ST

(2) 变量

In: 被转换的数据, 输入变量, 实数型, 默认值为 0。

Out: 转换结果, 输出变量, 整数型。

(3) 功能

这些指令执行时, 将实数 In 转换为一个整数。指令的名称由 In 和转换结果 Out 的数据类型决定。例如, 如果 In 是 LREAL 数据、Out 是 LINT 数据, 则指令的名称是 LREAL_TO_LINT。

图 4-65 所示为 LREAL_TO_LINT 指令的应用, In 是 LREAL#1.0e+10。

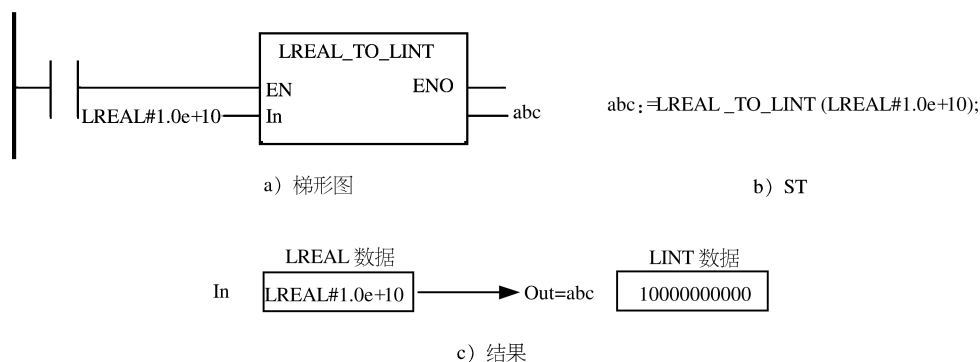


图 4-65 LREAL_TO_LINT 指令的应用

4.12 位串处理指令

位串处理指令见表 4-26。

表 4-26 位串处理指令

指 令	FB/FUN	名 称
AND (&)、OR 和 XOR	FUN	逻辑与/逻辑或/逻辑异或
XORN	FUN	逻辑异或非
NOT	FUN	按位取反
AryAnd、AryOr、AryXor 和 AryXorN	FUN	数组逻辑与/数组逻辑或/数组逻辑异或/数组逻辑异或非

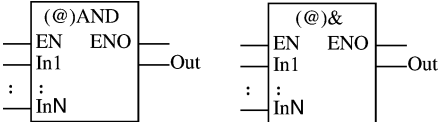
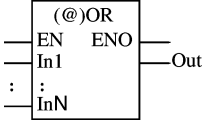
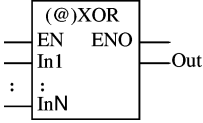
下面介绍常用的数据比较指令。

1. 逻辑与 AND (&)、逻辑或 OR、逻辑异或 XOR 指令

(1) 指令的梯形图及 ST

AND (&)、OR 和 XOR 指令见表 4-27。

表 4-27 AND (&)、OR 和 XOR 指令

指令	名称	FB/FUN	梯 形 图	ST 表达式
AND(&)	与	FUN		Out: = In1 AND ... AND InN; Out: = In1 & ... & InN;
OR	或	FUN		Out: = In1 OR ... OR InN;
XOR	异或	FUN		Out: = In1 XOR ... XOR InN;

(2) 变量

In1 ~ InN: 被处理的数据, 输入变量, BOOL 型或位串型, 数据的个数 N = 2 ~ 5。

Out: 处理结果, 输出变量, 必须与 In1 ~ InN 相同的数据类型。

(3) 功能

这些指令对 BOOL 变量或位串中相应的位进行处理。待处理的数据是 In1 ~ InN, 它们必须是相同的数据类型。

AND (&): 如果两个位都是 TRUE, 则处理结果为 TRUE, 否则为 FALSE。

OR: 如果两个位都为 FALSE, 则处理结果为 FALSE, 否则为 TRUE。

XOR: 如果两个位是相同的, 则处理结果为 FALSE; 如果一个位是 TRUE, 另一个是 FALSE, 则结果为 TRUE。

图 4-66 所示为 AND 指令的应用。

2. 逻辑异或非指令 XORN

(1) 指令的梯形图及 ST

XORN 指令的梯形图及 ST 如图 4-67 所示。

(2) 变量

In1 ~ InN: 被处理的数据, 输入变量, N = 2 ~ 5 为数据个数, 数据为 BOOL 或位串型。

Out: 处理结果, 输出变量, 与 In1 ~ InN 相同的数据类型。

(3) 功能

这些指令对 BOOL 变量或位串中相对应的位进行处理, 待处理的数据 In1 ~ InN 必须具有相同的数据类型。

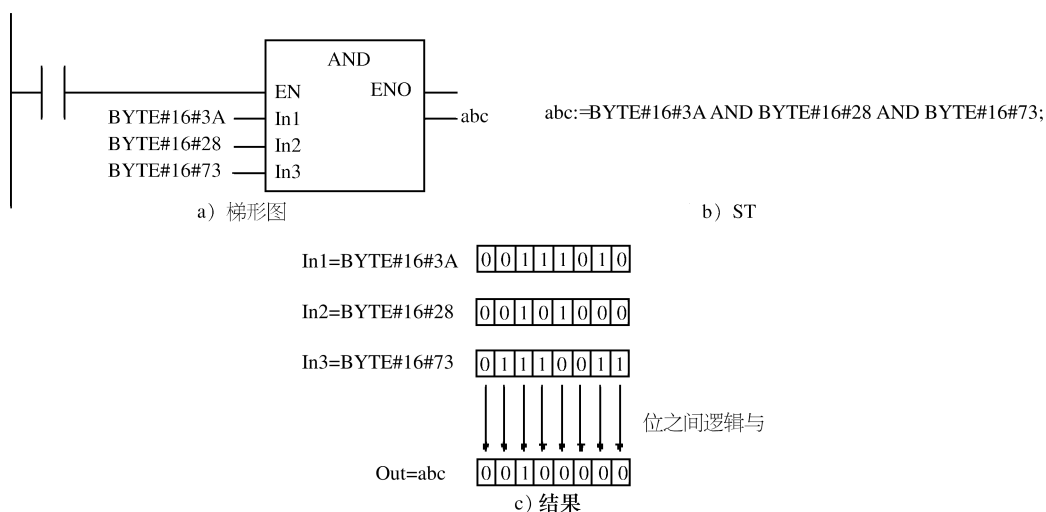


图 4-66 AND 指令的应用

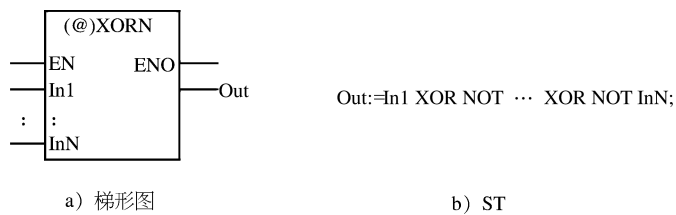


图 4-67 XORN 指令的梯形图及 ST

XORN: 如果两个位是相同的, 则位处理结果为 TRUE, 否则为 FALSE。
图 4-68 所示为 XORN 指令的应用。

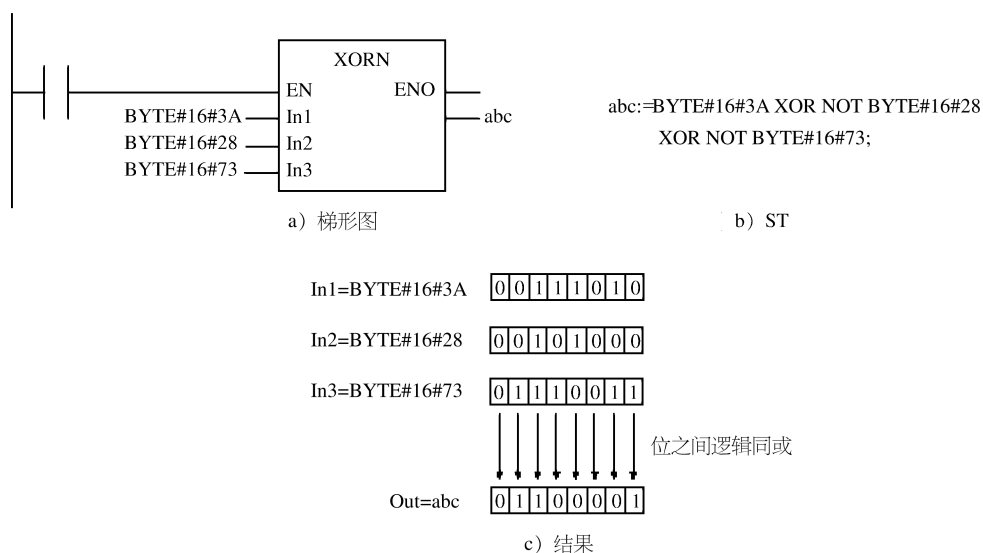


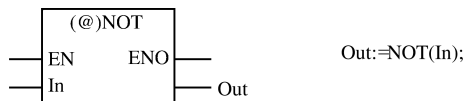
图 4-68 XORN 指令的应用

(1) 指令的梯形图及 ST

(2) 变量

Out: 处理结果, 输出变量, 与 In 相同的数

据类型。



(3) 功能

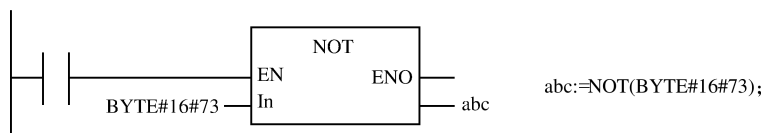
NOT 指令对 BOOL 变量或位串中各个位取反, 处理的数据是 In。In 和处理结果 Out 必须有相同的位数。

a) 梯形图

b) ST

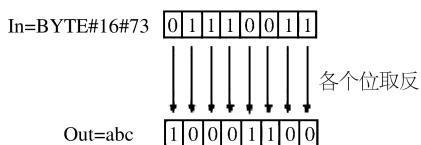
图 4-69 NOT 指令的梯形图及 ST

图 4-70 所示为 NOT 指令的应用, In 是 BYTE#16#73。



a) 梯形图

b) ST



c) 结果

图 4-70 NOT 指令的应用

4.13 数据传送指令

数据传送指令见表 4-28。

表 4-28 数据传送指令

指 令	FB/FUN	名 称
MOVE	FUN	传送
MoveBit	FUN	位传送
MoveDigit	FUN	数字传送
TransBits	FUN	多位传送
MemCopy	FUN	内存复制
SetBlock	FUN	块设置
Exchange	FUN	数据交换
AryExchange	FUN	数组数据交换
AryMove	FUN	数组传送

(续)

指 令	FB/FUN	名 称
Clear	FUN	初始化
Copy* * ToNum(Bit String to Signed Integer)	FUN	位模式复制(位串→有符号整数)类
Copy* * To* * * (Bit String to Real Number)	FUN	位模式复制(位串→实数)类
CopyNumTo* * (Signed Integer to Bit String)	FUN	位模式复制(有符号整数→位串)类
CopyNumTo* * (Signed Integer to Real Number)	FUN	位模式复制(有符号整数→实数)类
Copy* * To* * * (Real Number to Bit String)	FUN	位模式复制(实数→位串)类
Copy* * ToNum(Real Number to Signed Integer)	FUN	位模式复制(实数→有符号整数)类

下面介绍常用的数据传送指令。

1. 传送指令 MOVE

(1) 指令的梯形图及 ST

MOVE 指令的梯形图及 ST 如图 4-71 所示。

(2) 变量

In：源数据，输入变量，数据类型为 BOOL、位串、整数、实数、时间、日期、一天中的时间、日期和时间、字符串。

Out：目的通道，输出变量，与 In 相同的数据类型。

(3) 功能

MOVE 指令将源 In 中的值传送到目标 Out。传递给 In 的输入参数可以是一个变量或常量。可以为 In 指定一个枚举体、数组、数组元素、结构体或结构体成员。

图 4-72 所示为 MOVE 指令的应用，变量 abc 的内容传送到变量 def 中。

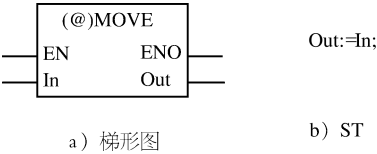


图 4-71 MOVE 指令的梯形图及 ST

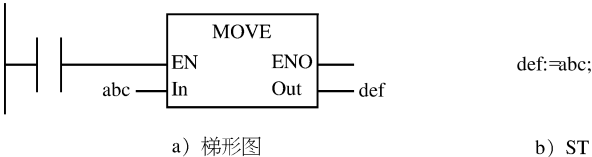


图 4-72 MOVE 指令的应用

当传送一个数组，既可以传送一个元素，也可以传送这个数组里的所有元素。仅传送一个元素时，为数组变量名加下标。传送整个数组时，数组变量名不加下标。

图 4-73 所示为传送一个数组元素。

图 4-74 所示为传送所有数组元素。

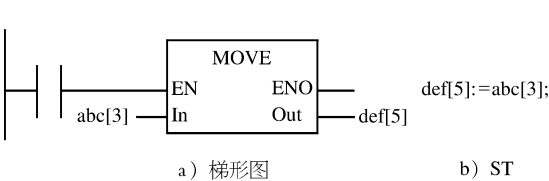


图 4-73 传送一个数组元素

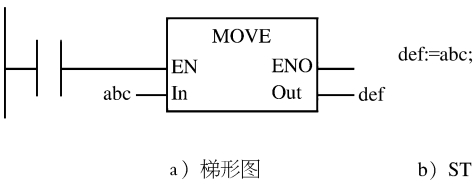


图 4-74 传送所有数组元素

当传送一个结构体，可以仅传送一个成员，或传送结构体中所有成员。仅传送一个成员时，指定该成员。传送整个结构体时，仅给出结构体的名称。

图 4-75 所示为传送一个结构体成员。

图 4-76 所示为传送整个结构体。

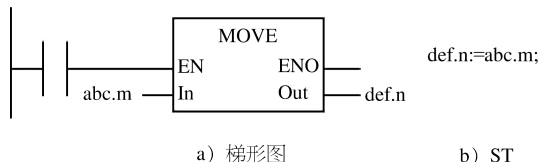


图 4-75 传送一个结构体成员

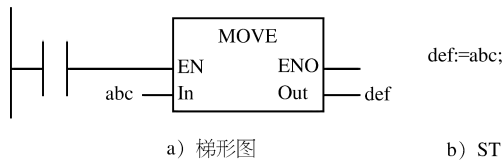


图 4-76 传送整个结构体

使用 MemCopy 指令可以比 MOVE 指令更快地传送整个数组。

2. 位传送指令 MoveBit

(1) 指令的梯形图及 ST

MoveBit 指令的梯形图及 ST 如图 4-77 所示。

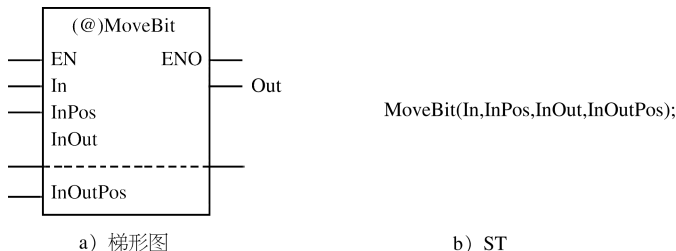


图 4-77 MoveBit 指令的梯形图及 ST

(2) 变量

In: 传送的源数据，输入变量，位串型数据。

InPos: 源数据中要传送位的位置，输入变量，USINT 型，取值范围为 0 至 In 位数 - 1，默认值为 0。

InOutPos: 目标中接收的位的位置，输入变量，USINT 型，取值范围为 0 至 InOut 位数 - 1，默认值为 0。

InOut: 传送的目标，输入-输出变量，位串型数据。

Out: 返回值，输出变量，总为 TRUE。

(3) 功能

MoveBit 指令将源 In 的第 InPos 位传送到目标 InOut 的第 InOutPos 位上。

图 4-78 所示为 MoveBit 指令的应用，InPos 是 USINT#3，InOutPos 是 USINT#5。

3. 数字传送指令 MoveDigit

(1) 指令的梯形图及 ST

MoveDigit 指令的梯形图及 ST 如图 4-79 所示。

(2) 变量

In: 传送的源数据，输入变量，位串型数据。

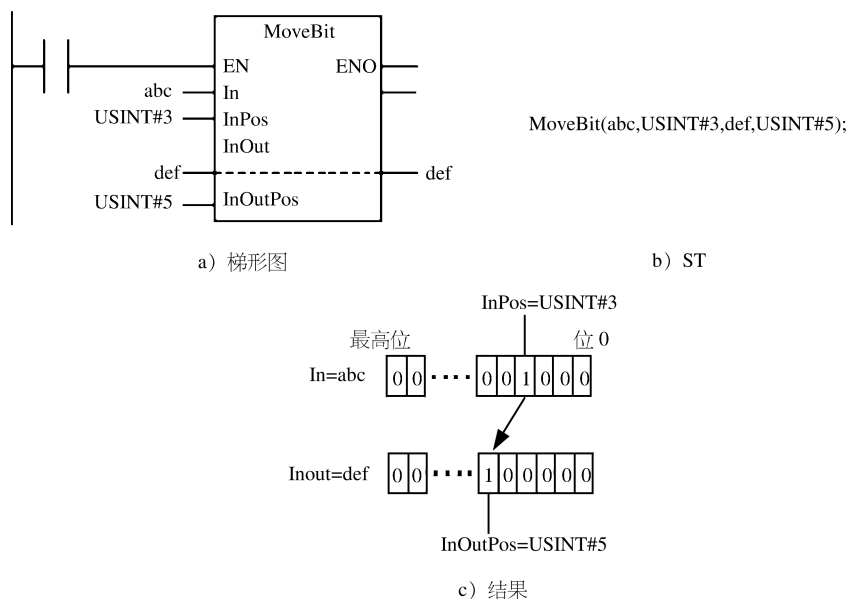


图 4-78 MoveBit 指令的应用

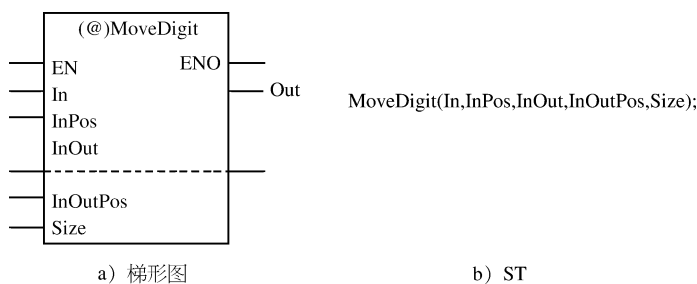


图 4-79 MoveDigit 指令的梯形图及 ST

InPos: 源数据中要传送数字的位置, 输入变量, USINT 型, 取值范围为 0 至 In 位数除以 4 减去 1, 默认值为 0。

InOutPos: 目标中接收的位的位置, 输入变量, USINT 型, 取值范围为 0 至 InOut 位数除以 4 减去 1, 默认值为 0。

Size: 要移动的数字数目, 输入变量, USINT 型, 取值范围为 0 至 In 位数除以 4, 默认值为 1。

InOut: 传送的目标, 输入-输出变量, 位串型数据。

Out: 返回值, 输出变量, 总为 TRUE。

(3) 功能

MoveDigit 指令将源 In 从第 InPos 数字开始的 Size 个数字, 移动到目标 InOut 从第 InOutPos 数字开始的位置上。一个数字是四位。

图 4-80 所示为 MoveDigit 指令的应用, InPos 是 USINT#1, InOutPos 是 USINT#2, Size 是 USINT#2。

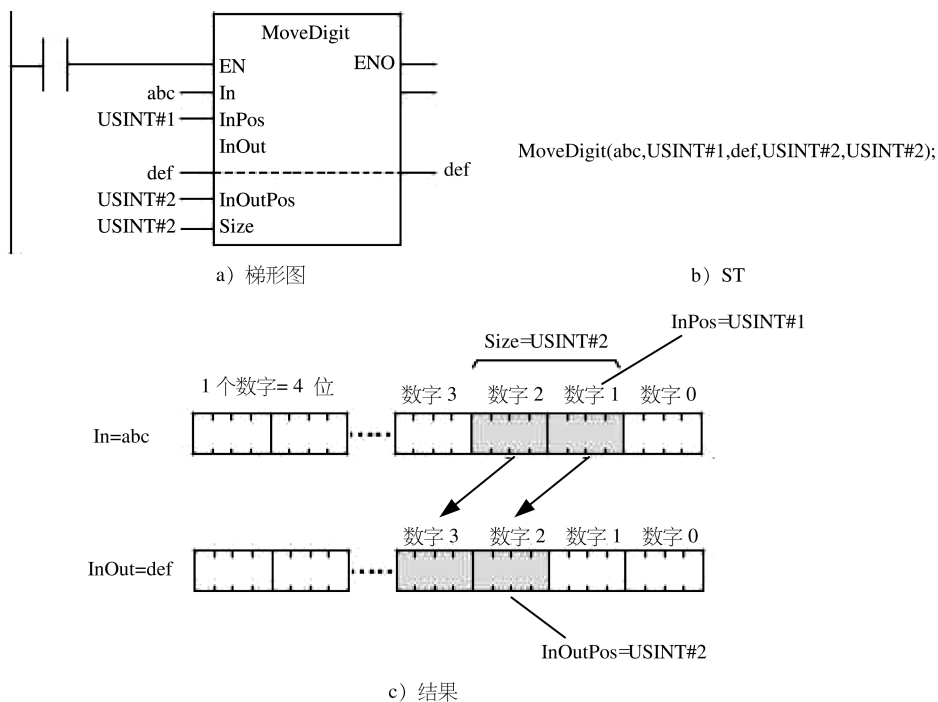


图 4-80 MoveDigit 指令的应用

4.14 移位指令

移位指令见表 4-29。

表 4-29 移位指令

指 令	FB/FUN	名 称
AryShiftReg	FB	移位寄存器
AryShiftRegLR	FB	反向移位寄存器
ArySHL 和 ArySHR	FUN	数组 N 个元素左移位/数组 N 个元素右移位
SHL 和 SHR	FUN	N 位左移位/N 位右移位
NSHLC 和 NSHRC	FUN	带 CY 的 N 位左移位/带 CY 的 N 位右移位
ROL 和 ROR	FUN	循环 N 位左移位/循环 N 位右移位

下面介绍常用的移位指令。

1. SHL 和 SHR 指令

(1) 指令的梯形图及 ST

SHL 和 SHR 指令的梯形图及 ST 如图 4-81 所示。

(2) 变量

In: 移位数据, 输入变量, 位串型。

Num: 移位位数, 输入变量, USINT 型, 取值范围 0 至 In 的位数, 默认值为 1。

Out: 处理结果, 输出变量, 与 In 具有相同的数据类型。

(3) 功能

指令执行一次, 将 In (位串数据) 移动 Num 位。移出该寄存器的位将被舍弃, 而寄存器的另一端则移入 0。

SHL 指令从右向左移位 (从最低位到最高位), SHR 指令从左向右移位 (从最高位到最低位)。

图 4-82 所示为 SHL 指令的应用, In 是 BYTE#16#89, Num 是 USINT#2。

图 4-83 所示为 SHR 指令的应用, In 是 BYTE#16#89, Num 是 USINT#2。

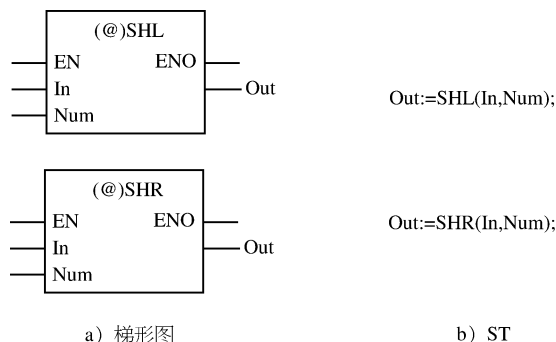


图 4-81 SHL 和 SHR 指令的梯形图及 ST

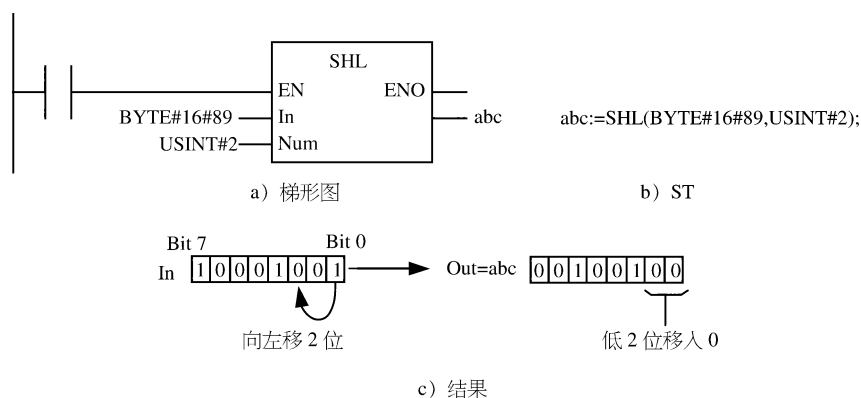


图 4-82 SHL 指令的应用

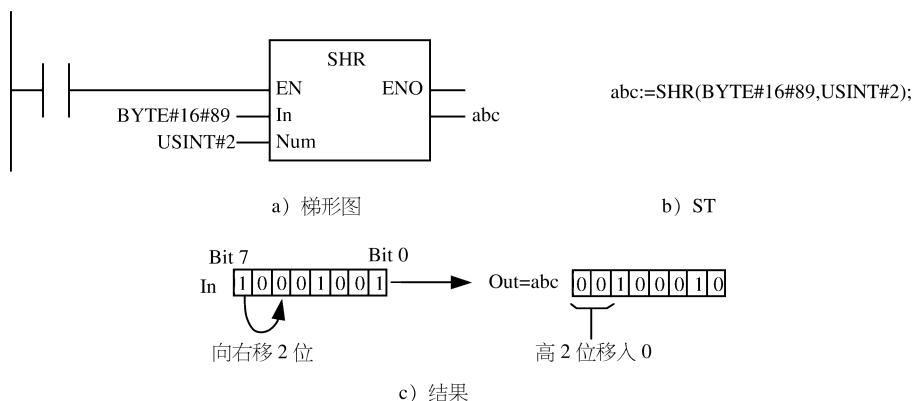


图 4-83 SHR 指令的应用

2. NSHL 和 NSHRC 指令

(1) 指令的梯形图及 ST

NSHL 和 NSHRC 指令的梯形图及 ST 如图 4-84 所示。

(2) 变量

Size: 移位寄存器的位数, 输入变量, USINT 型, 默认值为 1。

Num: 移位位数, 输入变量, USINT 型, 默认值为 1。

InOut [] (数组): 移位寄存器数组, 输入-输出变量, 位串型数据, 为移位的位串数组。

Out: 返回值, 输出变量, 总为 TRUE。

(3) 功能

指令执行一次, 将移位寄存器数组 InOut [] 从 InOut [0] 开始的数组元素的 Size 个位, 移动 Num 位。移出寄存器的最后一位输出到 CY 位, 其另一端的位则移入零。

NSHLC 指令从数组的低位元素到高位元素、从数组元素的低位到高位移位。NSHRC 指令从数组的高位元素到低位元素、从数组元素的高位到低位移位。

图 4-85 所示为 NSHLC 指令的应用, InOut [] 是 BYTE 型数组, Size 是 USINT#80, Num 是 USINT#3。

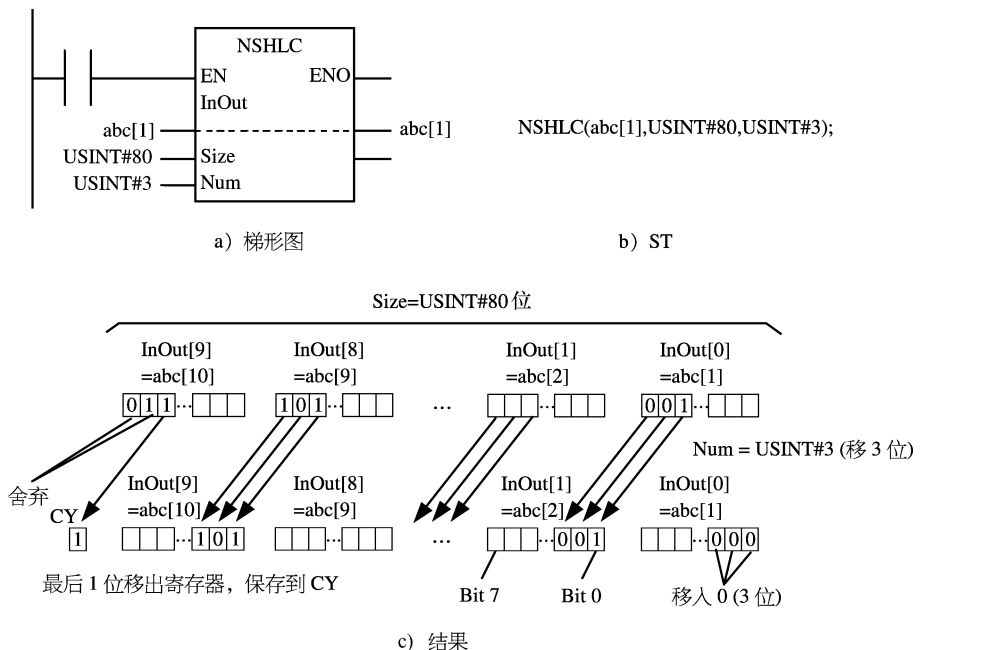


图 4-85 NSHLC 指令的应用

3. ROL 和 ROR 指令

(1) 指令的梯形图及 ST

ROL 和 ROR 指令的梯形图及 ST 如图 4-86 所示。

(2) 变量

In: 循环移位数据, 输入变量, 位串型数据。

Num: 移位位数, 输入变量, USINT 型, 取值范围 0 至 In 的位数, 默认值为 1。

Out: 处理结果, 输出变量, 与 In 具有相同的数据类型。

(3) 功能

指令执行一次, In (位串型数据) 循环移动 Num 位, 移出寄存器的位移入到寄存器的另一端。

ROL 指令从右向左循环移位, 即从最低位到最高位。ROR 指令从左向右循环移位, 即从最高位到最低位。

图 4-87 所示为 ROL 指令的应用, In 是 BYTE#16#89, Num 是 USINT#2。

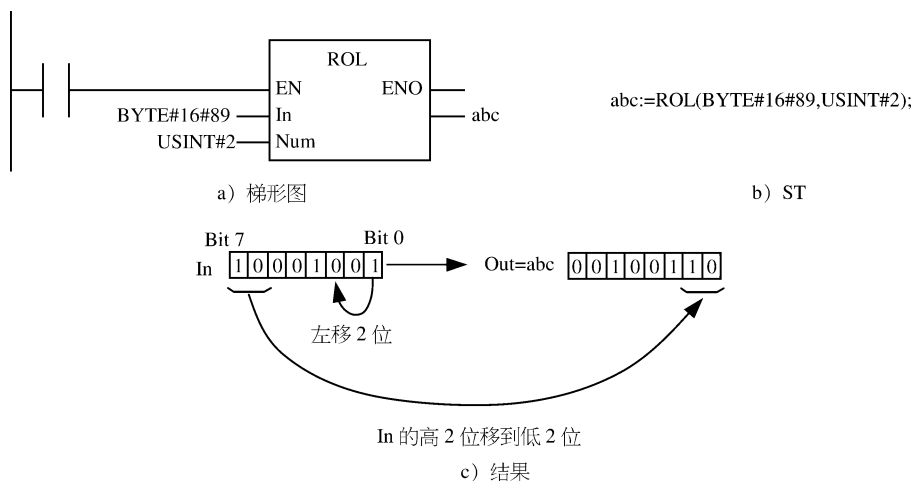


图 4-87 ROL 指令的应用

图 4-88 所示为 ROR 指令的应用, In 是 BYTE#16#89, Num 是 USINT#2。

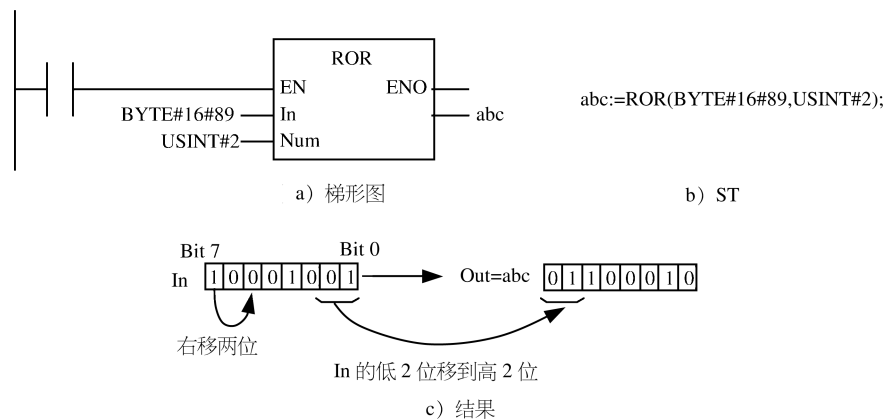


图 4-88 ROR 指令的应用

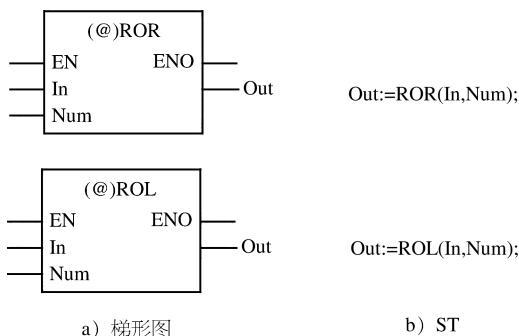


图 4-86 ROL 和 ROR 指令的梯形图及 ST

第 5 章 Sysmac Studio 在编程与调试中的应用

第 2 章介绍了 Sysmac Studio 的组成、安装、工程创建和硬件组态，第 3 章介绍了应用 Sysmac Studio 登记数据类型、登记变量、登记任务等。本章将介绍 Sysmac Studio 在编程与调试中的应用。

学好 NJ 的关键是大量实践，由于价格等原因，普通读者很难得到 NJ 的硬件进行实际操作。好在 Sysmac Studio 集成了强大的仿真功能，能够模拟 NJ 的绝大部分功能，如顺序控制、运动控制和视觉传感器等。在没有实际 NJ 的情况下，完全可以利用 Sysmac Studio 进行学习，熟悉各种操作，甚至完成 NJ 控制系统的预开发。

5.1 编程与调试步骤

在使用 Sysmac Studio 创建工程并完成控制器的配置和设置后，接下来的工作便是编写和调试用户程序，涉及如下操作。

1. 设计用户程序

登记变量和创建 POU 的。

(1) 登记全局变量

将在多个程序中使用的用户定义变量登记为全局变量。

(2) 登记 POU 的

登记程序、功能和功能块。

(3) 登记本地变量

将只在一个 POU 中使用的用户定义变量登记为这个 POU 的本地变量。

(4) 登记轴变量

(5) 登记轴组变量

(6) 登记凸轮数据变量

(7) 创建 POU 算法

2. 设计任务

(1) 登记任务

(2) 设计任务

(3) 编辑任务

将程序分配给任务。

(4) 编辑任务

设计由任务控制的 I/O 和任务间共享的变量。

3. 离线调试程序

使用模拟器调试程序。

4. 在线调试准备

为在线调试，需做以下设置。

- (1) 在“轴设置”中分配变量和轴标识
- (2) 在“I/O 映射”中为实际 I/O 分配变量

5. 在线调试程序

在实际系统上完成调试。

- (1) 将数据传送到控制器
传送用户程序、系统配置和变量信息到控制器。
- (2) 测试轴操作
使用 MC 试运行检查接线。

- (3) 检查 I/O

使用 I/O 映射在内部 I/O 设备和外部 I/O 设备间检查 I/O 状态、与外部 I/O 设备的接线。

- (4) 检查分配

使用查看窗口检查用户定义变量和 I/O 设备之间的分配。

6. 测试操作

使用测试操作完成最后的调试。

用数据跟踪检查用户程序的运行，调试程序。

以下几节主要介绍与用户程序编写和调试相关的操作，如梯形图编程、ST 编程、程序调试、离线调试和在线调试。

5.2 编程操作

5.2.1 添加梯形图或 ST

POU 中的程序、功能或功能块既可以使用梯形图编写，也可以使用 ST 编写。究竟使用哪一种编程语言，在创建程序、功能或功能块时，由用户选择决定。

图 5-1 所示为添加程序操作，右击多视图浏览器中的“编程-POUs”下的“程序”，从菜单中选择“添加-梯形图”，则创建梯形图程序，如图 5-2 所示，添加了梯形图程序“Program1”；若选择“添加-ST”，则创建 ST 程序，如图 5-3 所示，添加了 ST 程序“程序 0”。注意，图中的“Program0”是工程创建时自动创建的，默认为梯形图程序。



图 5-1 添加程序操作



图 5-2 添加梯形图程序

功能和功能块添加时也可以选择梯形图或 ST，操作过程和添加程序一样。

5.2.2 梯形图编程

梯形图编程是用连接线把电路部分连接起来构建算法。在梯形图编辑器中输入电路和连接线。

1. 打开梯形图编辑器

(1) 程序

在多视图浏览器中的“编程-POUs-程序”下，双击梯形图程序的一个段；或右击段，从菜单中选择“编辑”，如图 5-4 所示。



图 5-3 添加 ST 程序



图 5-4 打开梯形图编辑器

梯形图编辑器如图 5-5 所示，下半部显示出一个空的梯形图。



图 5-5 梯形图编辑器

(2) 功能和功能块

参照图 5-1，双击“编程-POUs-功能”；或右击“功能”，从菜单中选择“编辑”，打开功能的梯形图编辑器。同样可打开功能块的梯形图编辑器。

2. 梯形图程序段的操作

为了便于管理，梯形图程序分成更小的单位，称为段。一个梯形图程序可以有多个段。

图 5-6 所示为添加段的操作界面，操作步骤如下：

1) 选中多视图浏览器中“编程-POUs-程序”下的“Program0”。

2) 右击“Program0”，从菜单中选择“添加-段”，“Section1”便添加到“Program0”下面。



图 5-6 添加段

对于“段”的操作有：编辑、剪切、复制、粘贴、删除、重命名、上移、下移。

3. 梯形图输入方法

梯形图输入界面如图 5-7 所示。图中右边的视图为工具箱，里面放置 NJ 所有的梯形图指令；左上半部为全局变量表或本地变量表；左下半部为输入的梯形图程序。梯形图左侧数字为回路编号，从回路 0 开始，每个回路开头可输入回路注释。输入时，当前选择的回路左总线处显示一条粗线，粗线为红色高亮时表示编程出错，蓝色表示输入正确。此外，梯形图中如果出现错误，则元素的文本为红色。

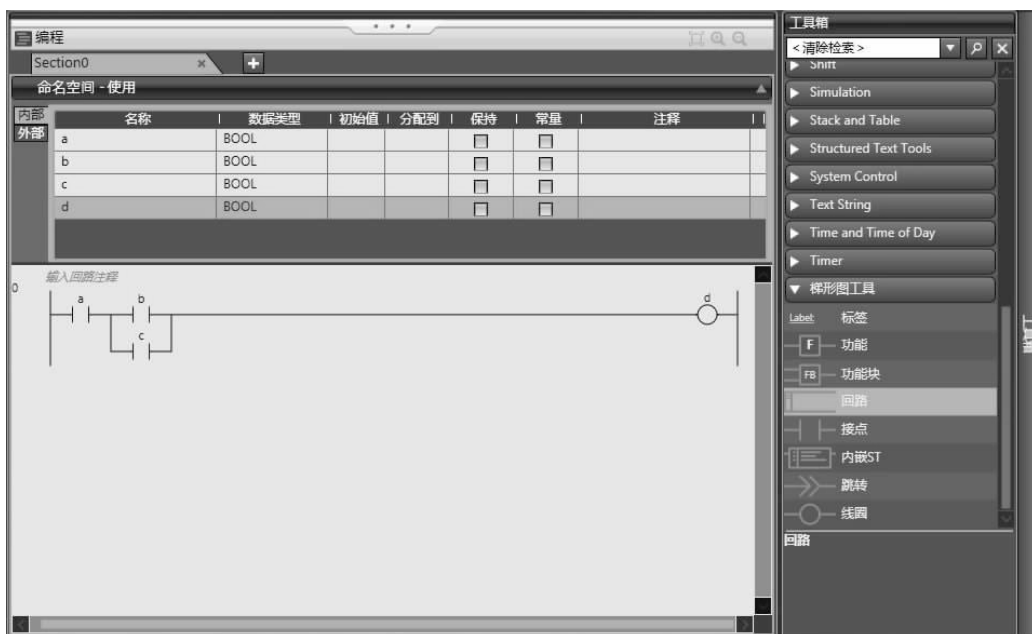


图 5-7 梯形图输入界面

梯形图由连接线、电路元件组成，连接线将电路元件连接在一起。

(1) 插入电路元件

通常有 4 种方法插入电路元件：

- 1) 从工具箱拖放电路元件。
- 2) 右击连接线，从弹出的菜单中选择插入的电路元件。

- 3) 选择连接线，按快捷键插入电路部分。
- 4) 选择连接线，从主菜单中选择电路元件。

(2) 输入变量

变量分为全局变量或本地变量。电路元件输入后，要接着输入变量。

输入全局变量时，从变量列表中选择登记在全局变量表中的变量。

输入本地变量有 3 种方法：

- 1) 在梯形图编辑器，从变量列表中选择登记在本地变量表中的变量。
 - 2) 从 POU 的本地变量表中拖放一个变量到梯形图编辑器。
 - 3) 如果变量没有提前登记到本地变量表，可以直接在梯形图编辑器输入一个变量。
- 注意，使用 ST 编辑器时，变量需提前登记。

4. 梯形图基本电路输入

图 5-7 中的梯形图，电路元件类型有接点、输出，连接既有串联也有并联。梯形图输入过程如下。

- 1) 右击连接线，从弹出的菜单中选择“插入接点”，如图 5-8 所示。
输入接点的变量名称 a。
- 2) 按同样方法输入串联接点 b，如图 5-9 所示。

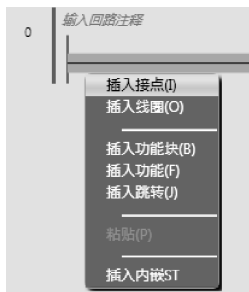


图 5-8 插入接点



图 5-9 输入串联的接点 b

- 3) 右击接点 b，从菜单中选择“插入接点 (向下)”，输入并联的接点，如图 5-10 所示。

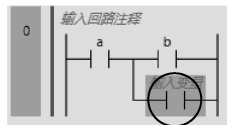
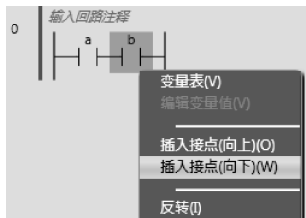


图 5-10 输入并联的接点

输入并联接点的变量名称 c。

- 4) 右击连接线，从菜单中选择“插入线圈”，输入线圈，如图 5-11 所示。

输入线圈的变量名称 d。

至此，完成了梯形图输入。



图 5-11 输入线圈

对接点和输出进行取反、上升沿微分或下降沿微分时，选中接点或输出，右击后弹出菜单，从中选择，图 5-12 所示为接点取反操作。

梯形图编辑时，除了插入，还可进行修改、剪切、复制、粘贴、移动、删除、撤销、恢复、查找、替换等操作。

5. 插入回路

使用以下方法插入回路。

1) 从工具箱中的梯形图工具选择回路，拖放到欲插入回路位置的上一个回路的起始位置。



图 5-12 接点取反操作

2) 右击回路，并选择“插入回路(向上)”或“插入回路(向下)”，一条空的回路插在选中回路的上方或下方，如图 5-13 所示。



图 5-13 插入回路

6. 插入和删除功能块指令和用户定义功能块

(1) 插入功能块指令

以输入通信指令 SendCmd 为例。

1) 从工具箱的通信里选择 SendCmd，把它拖放到插入的位置，如图 5-14 所示。
2) 在功能块顶部输入功能块的例程名称。如图 5-15 所示，如果本地变量表中登记功能块例程，可以从中选择。

(2) 插入用户定义的功能块和功能块指令

插入用户定义功能块和功能块指令有 3 种常用方法。

1) 从工具箱里拖放功能块

①从工具箱的梯形图工具里选择功能块，把它拖放到插入的位置，如图 5-16 所示，添加一个空的功能块。

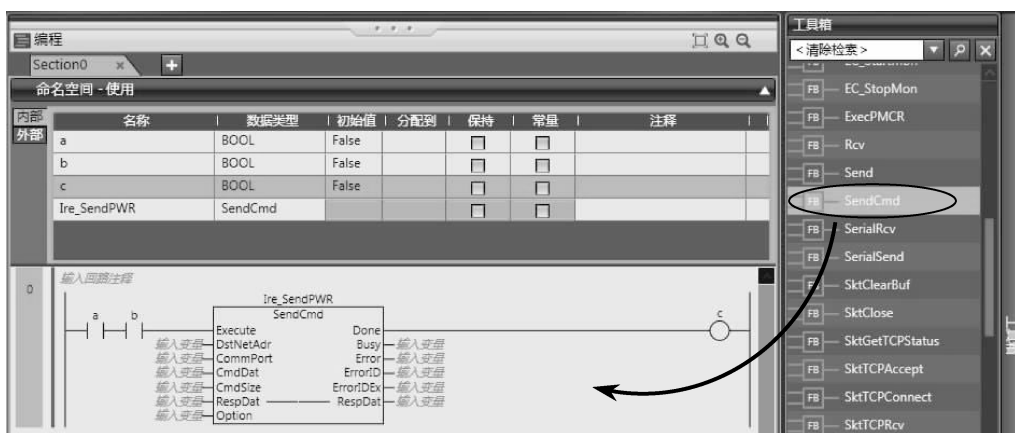


图 5-14 插入功能块指令

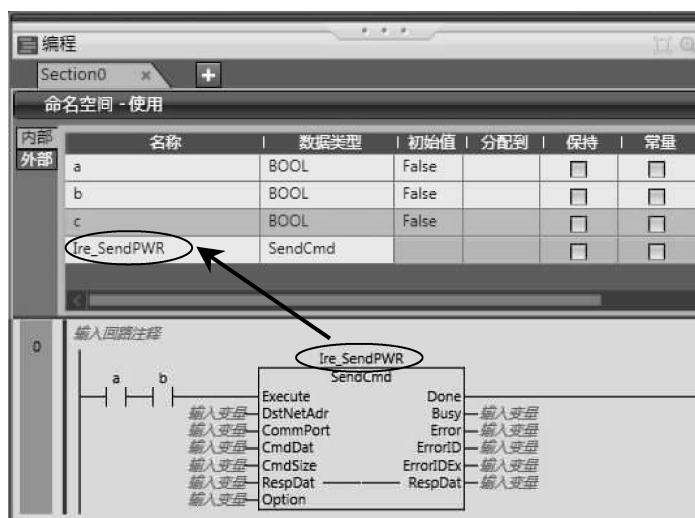


图 5-15 输入功能块例程名称



图 5-16 插入功能块

②向插入的功能块输入功能块的类型名称 (数据类型), 可以从显示的列表中选择一个功能块或功能块指令, 如图 5-17 所示。

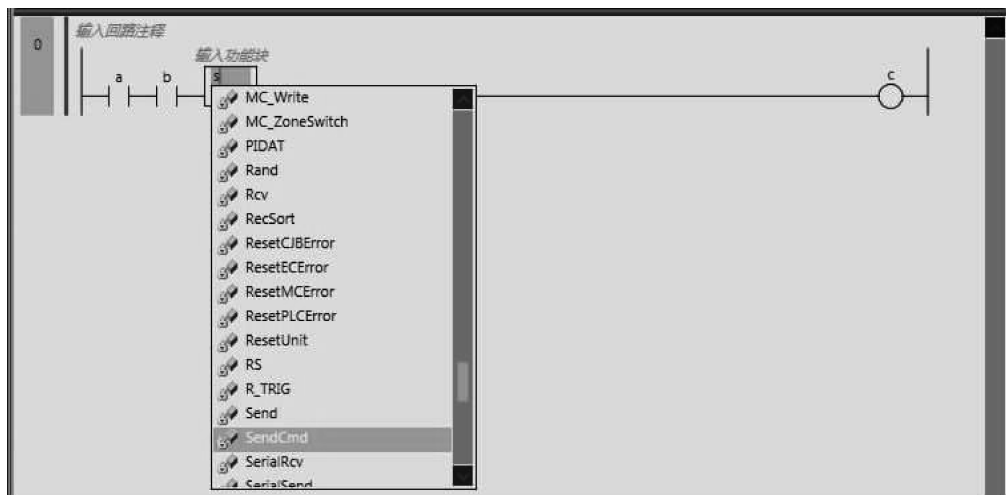


图 5-17 输入功能块的类型名称

③在插入功能块顶部输入功能块的例程名称。如果本地变量表中登记功能块例程, 可以从中选择。

至此, 完成功能块的插入。如果输入一个新的例程名称, 会登记到本地变量表中。

2) 右击要插入的位置, 从菜单中选择“插入功能块”; 向插入的功能块输入功能块的类型名称, 并在功能块顶部输入功能块的例程名称。

3) 将光标移到插入位置, 按 F 键; 向插入的功能块输入功能块的类型名称, 并在功能块顶部输入功能块的例程名称。

(3) 删除功能块例程

右击功能块例程并从菜单中选择“删除”。或者, 选中功能块例程并按 delete 键。

注意, 功能块被删除, 然而, 登记在变量表中的功能块例程和用作参数的变量没有删掉。

(4) 输入功能块变量

功能块变量的输入方法与电路元件变量的输入方法相同。

7. 插入和删除功能指令和用户定义功能

插入功能指令时, 从工具箱里拖放。

插入用户定义功能时, 操作过程和插入用户定义的功能块相似, 有 3 种方法。

1) 从工具箱里拖放功能。

2) 右击插入的位置, 从菜单中选择“插入功能块”。

3) 将光标移到插入位置, 按“F”键。

删除功能时, 右击功能并从菜单中选择“删除”; 或者, 选中功能后按 delete 键。选中的功能虽然被删除, 但是, 功能中使用的变量并没有删除。

功能的变量输入方法与电路元件变量的输入方法相同。

8. 插入和删除内嵌 ST

通过内嵌 ST，可以将 ST 程序嵌入到梯形图中，实现更加灵活的编程。

插入内嵌 ST 时，在准备插入 ST 位置的连接线处右击，从菜单中选择“插入内嵌 ST”；或者，在工具箱中的梯形图工具拖放“内嵌 ST”，如图 5-18 所示。

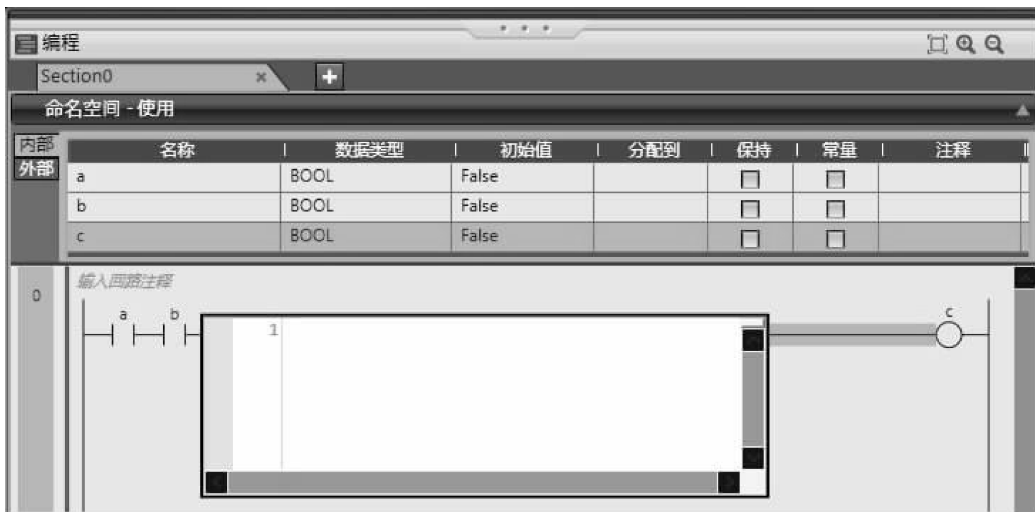


图 5-18 插入内嵌 ST 方框

删除内嵌 ST 时，右击要删除的内嵌 ST 方框，点击之，从弹出的菜单中选择“删除”；或选中后，按 delete 键。

也可以复制和粘贴内嵌 ST。

9. 插入和删除跳转标签和跳转

图 5-19 所示为跳转标签/跳转梯形图，在跳转的起始回路中指定 Jump1 标签，跳转的目标回路中插入了 Jump1 标签。

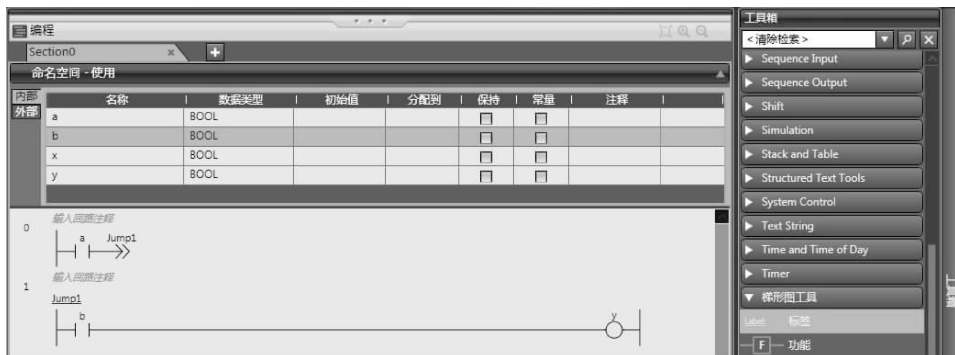


图 5-19 跳转标签/跳转梯形图

(1) 插入跳转标签

1) 从工具箱的梯形图工具拖放“标签”至目标回路的起始，如图 5-20 所示；也可右击目标回路的起始，从菜单中选择“插入跳转标签”。



图 5-20 插入跳转标签

2) 输入跳转标签名称，如图 5-21 所示，目标回路 1 中的跳转标签名称为“Jump1”。

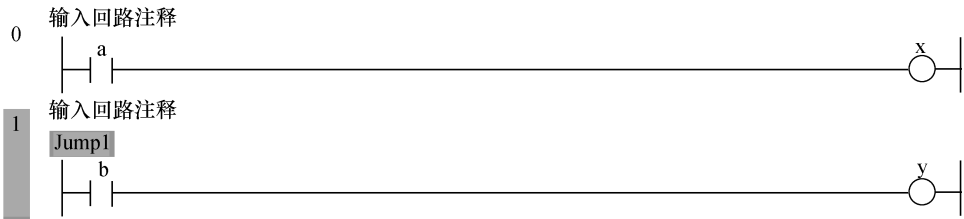


图 5-21 输入跳转标签名称

(2) 插入跳转

1) 从工具箱的梯形图工具拖放“跳转”至起始回路的连接线，如图 5-22 所示；或右击跳转起始回路的连接线，从菜单中选择“插入跳转”。插入跳转的右边所有电路都被删除，出现跳转电路部分。

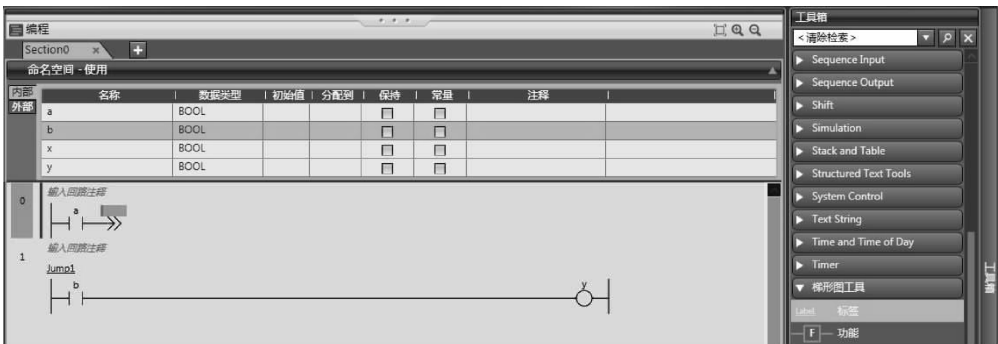


图 5-22 插入跳转

2) 点击跳转电路部分的跳转标签，输入目标标签的名称“Jump1”。

(3) 删除跳转标签和跳转

右击跳转标签或跳转，从弹出的菜单中选择“删除”；或选中后，按 delete 键。

10. 插入和清除书签

可以在回路的开始位置添加一个书签。右击回路的开始处，从菜单中选择“书签-切换书签”，如图 5-23 所示。书签会显示在回路的开始处，如图 5-24 所示。



图 5-23 输入书签

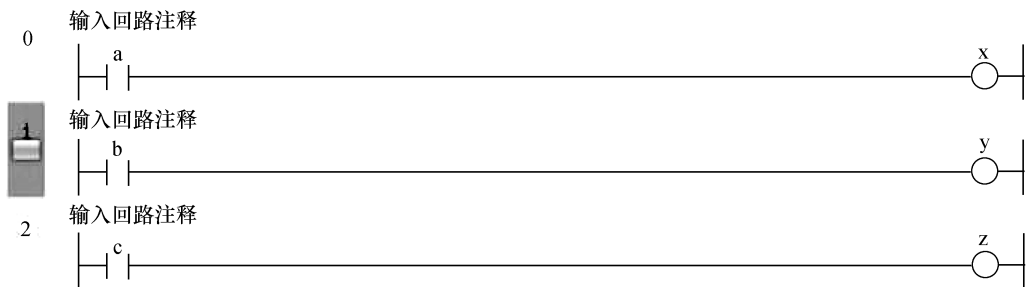


图 5-24 书签显示界面

清除回路的书签时，右击该回路的开始处，从菜单中选择“书签-切换书签”，书签就会被清除。若要清除全部书签，从菜单中选择“书签-清除全部书签”。

11. 输入回路注释

点击回路中的“输入回路注释”，输入注释，如图 5-25 所示。

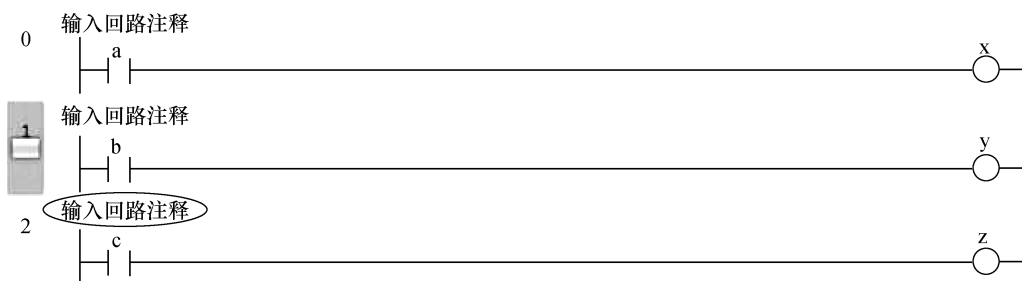


图 5-25 输入回路注释

5.2.3 ST 编程

1. 启动 ST 编辑器

在多视图浏览器中的“编程-POUs-程序”下，双击 ST 程序；或右击 ST 程序，从菜单中选择“编辑”，ST 编辑器如图 5-26 所示。

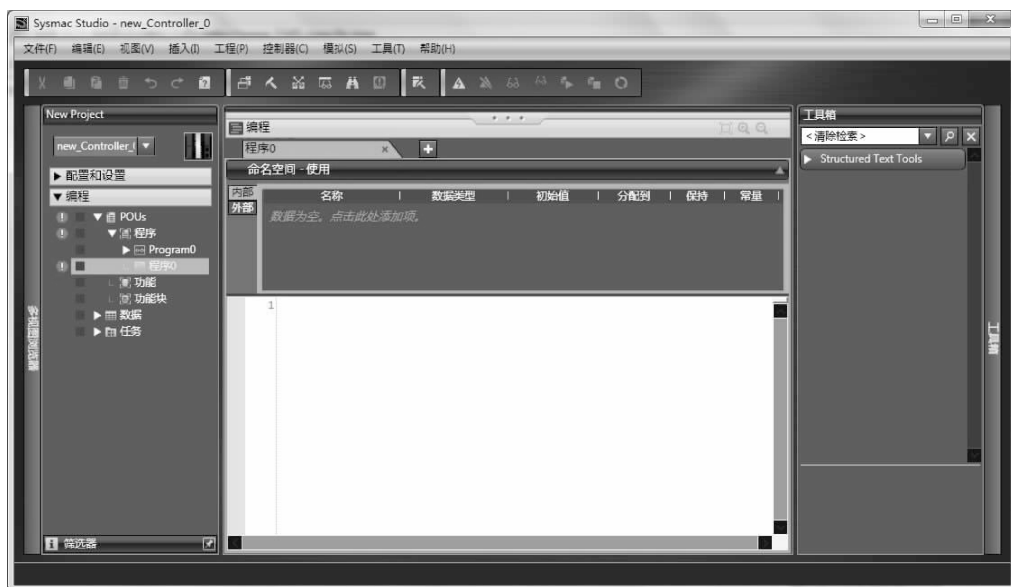


图 5-26 ST 编辑器

在多视图浏览器中，双击“编程-POUs-功能”下的 ST 功能；或右击 ST 功能，从菜单中选择“编辑”，可以编辑 ST 功能。按同样方法编辑 ST 功能块。

2. ST 输入

在 ST 编辑器中输入 ST 语句构造算法。

ST 语句中的元素（如变量、控制语句）输入时，输入元素第一个字母后，会出现所有候选元素的列表，可方便地从中选择。

输入 ST 语句时，用到的变量需预先创建并登记。

赋值语句、常数的输入相对简单，这里不再介绍。下面介绍控制语句输入、调用功能或功能块输入的操作步骤。

(1) 控制语句输入

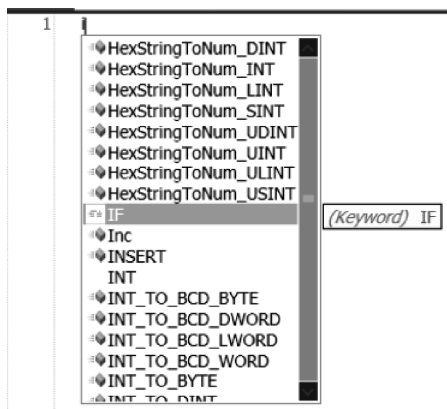
假设输入 IF 结构

IF aaa = bbb THEN

ccc: = aaa;

END_IF

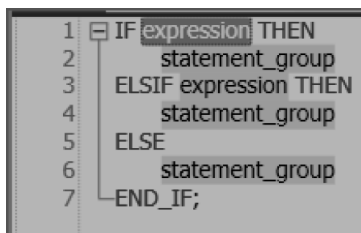
1) 在 ST 编辑器输入 IF 语句的第一个字母 i，出现一个首字母为 i 的字符串列表。



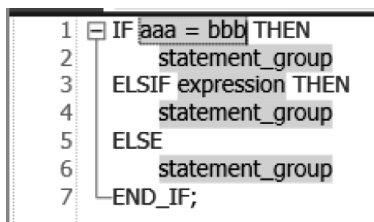
2) 选择 IF 关键字，按回车键，“IF” 便被输入。



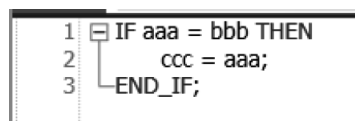
3) 按 TAB 键，组成 IF 结构的其他关键字会自动输入。



4) 输入语句 aaa = bbb。



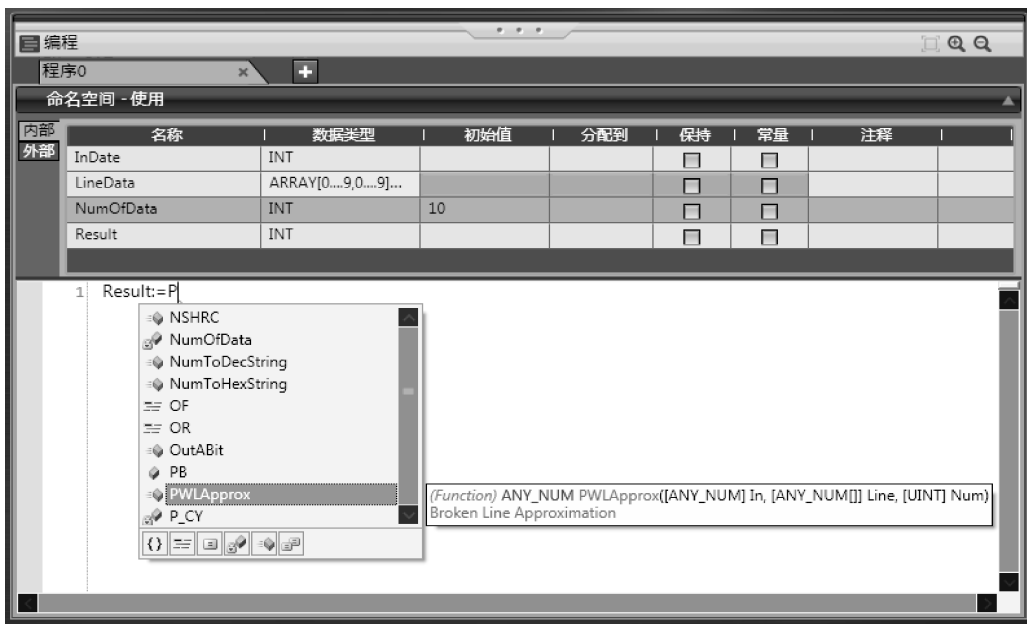
5) 删除 ELSIF、THEN 和 ELSE，然后输入 ccc: = aaa;。



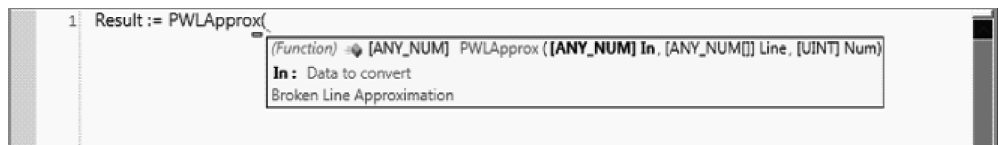
至此，完成 IF 结构的输入。

(2) 调用功能或功能块的输入

- 1) 在 ST 编辑器中输入功能或功能块例程名称的第一个字母。
- 2) 从列表中选择功能名称或功能块例程名称，按下回车键。



显示选中的功能或功能块例程，输入左括号，显示参数描述。



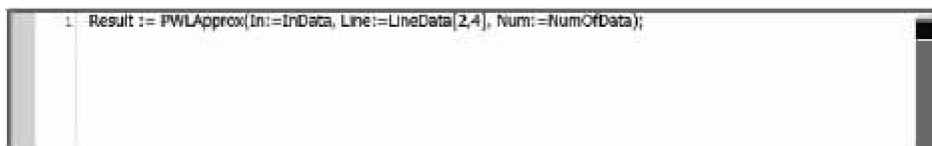
- 3) 点击功能或功能块例程中显示的参数标记，或标记正在显示时按下 Tab 键。



参数随之添加。



- 4) 输入需要的参数，完成语句。



ST 编辑器中的注释以 “(* ” 开头，以 “ *)” 结束。如果只需要注释一行，在行的开始输入双斜线 (/)。

为了便于阅读，可以缩进嵌套语句。

ST 编辑器中也可以做书签标记。

5.2.4 查找和替换

可以查找和替换工程数据里的字符串。

1. 查找和替换的项目

操作的范围：变量表、梯形图和 ST，查找和替换的项目如下。

变量表：变量名称、变量注释。

梯形图：回路注释、变量名称、功能名称、功能块定义名称、功能块例程名称。

ST：文本串、变量名称、功能名称、功能块例程名称。

2. 查找和替换面板

从“编辑”菜单中选择“查找和替换”，在工具箱的位置显示出“查找和替换”面板，如图 5-27 所示。

查找和替换面板分为三部分，最上面部分为“查找和替换”，设置“查找内容”、“替换为”、“查找范围”、“查找对象”和“使用”；中间部分为“查找选项”，选择“区分大小写”、“全字匹配”、“向上查找”、“反转常开/常闭接点”；最下面部分为操作按钮，可点击“查找下一个”、“全部查找”、“替换”、“全部替换”、“取消”其中之一进入下一步操作。

在设置“使用”时，如果选择“通配符”，则在“查找内容”中可以输入以下 5 种通配符。

*：一个任意字符或包含更多字符的任意字符串。

?：任意字符。

#：任意一个数字。

[]：指定范围内任意一个字符。

[!]：指定范围外任意一个字符。

选中“反转常开/常闭接点”时，对所有在梯形图中出现的 BOOL 变量，常开接点用常闭接点替换，常闭接点用常开接点替换。注意，输出线圈不受影响。



图 5-27 查找和替换面板

5.2.5 程序检查

使用程序检查可以检测创建的 POU（程序、功能或功能块）中的错误。有两种程序检查类型。

1. 检查所有的程序

从“工程”菜单里选择“检查所有的程序”，程序检查的结果会显示在编译窗口，如图 5-28 所示。



图 5-28 显示检查所有程序的结果

2. 检查选择的程序

- 1) 在多视图浏览器中，选中要检查的 POU 或程序段。
- 2) 从“工程”菜单里选择“检查选择的程序”，程序检查的结果会显示在编译窗口，如图 5-29 所示。



图 5-29 显示检查指定程序的结果

在编译窗口的显示中，给出了错误或警告数量、错误或警告说明、错误或警告位置。双击错误所在的行会跳转到错误的位置，错误的位置显示在编程界面中。在多视图浏览器中，包含错误的 POU 用惊叹号！标记。

5.2.6 编译和重编译

编译是在 CPU 单元里将工程程序转换成一种可执行文件格式的过程。在此过程中会检查程序和变量数据。如果存在任何错误，编译将不能完成并且错误会显示在编译窗口。第一次编译后，只有改变的程序才被重新编译。

1. 自动编译

当用户改变程序时，程序会自动编译。也可以手动编译。

更改数据类型、全局变量或 POU 包括本地变量和算法之后，如果 5s 内不进行任何操作，自动启动编译。

程序的编译和重编译的进程会在窗口右下角出现的进度条上显示出来。

2. 手动编译

在编译和重编译程序的进程中，不能检查全部或部分程序。可以终止自动编译，然后启用手动编译。

(1) 退出编译

退出编译时，从“工程”菜单中选择“退出编译”，编译操作随即终止，在输出窗口中会显示退出编译的信息。

(2) 手动编译

1) 从“工程”菜单中选择“编译控制器”，编译开始，编译过程中的状态会显示在输出窗口中，如图 5-30 所示。



图 5-30 输出窗口

2) 点击“编译窗口”以显示编译窗口，如果有错误，错误列表会显示出来。

3) 双击任意错误行会显示错误的位置，然后纠正错误。

(3) 重编译

重编译是用来编译已经编译过的工程程序。所有的程序都将被重新编译。

从“工程”菜单中选择“重编译控制器”，编译开始，编译过程的状态会显示在输出窗口里。

5.3 调试操作

5.3.1 模拟器与控制器

设计好 NJ 的用户程序后，需要对程序进行调试，一般用 NJ 的硬件来调试程序。但有时会遇到以下情况：学习人员或开发人员手头上没有实际的控制器；开发人员需要对程序进行修改和调试，但控制设备不在本地或者对实际系统进行某些调试有一定的风险。以上情况下，可以使用 Sysmac Studio 提供的模拟器进行仿真调试。

模拟器可以在计算机上模拟 NJ 的功能，提供一个与实际 NJ 等价的开发环境，用它代替 NJ 硬件来调试用户程序，可以在开发阶段有效地发现和排除故障，提高用户程序的质量和降低现场调试的费用。模拟器也是学习 NJ 控制器的有力工具。

模拟器可以模拟 NJ 控制器的绝大部分功能，但也有少量的功能不能实现，例如，内存全清除操作、监视控制器信息、复位控制器、故障诊断、时钟信息设置和发布访问权限，另外，网络通信也不能模拟。

但模拟器拥有实际控制器所不具备的多种专用的调试功能，例如，可以估算任务执行的时间，设置断点，设置单步执行、单周期执行或连续执行，改变模拟速度等，在 Sysmac Studio 上使用这些功能，可以对 NJ 的用户程序进行全面、细致的调试。

离线调试是在不连接控制器的情况下调试程序。将工程传送到实际控制器之前，可以在模拟器上仿真调试，检查控制程序的逻辑。

在线调试是将 Sysmac Studio 与实际的 NJ 控制器建立在线连接，检查和调试用户程序的运行。

以下为 Sysmac Studio 在线调试和离线调试都可以使用的操作。

5.3.2 监视

在梯形图编辑器、ST 编辑器、监视窗口或 I/O 映射中，可以监视程序输入和输出的 TRUE/FALSE 状态和控制器中变量的当前值。根据变量的类型选择数据显示的格式。

下面以监视窗口为例，介绍监视操作过程。

在监视窗口中检查一个或多个变量的当前值。

有两种类型的监视窗口，包含不同的信息。

(1) 监视窗口 (工程)

这个监视窗口包含登记在工程中的控制器信息。可以同时检查同一工程下多个在线控制器的当前值。

(2) 监视窗口 (控制器)

这个监视窗口包含当前正在显示的控制器的信息。

对于监视窗口 (工程)，登记变量时必须指定控制器，除此之外，两种监视窗口的操作是相同的。

从“视图”菜单选择“监视窗口”，出现监视窗口 (工程) 和监视窗口 (控制器)，如图 5-31 所示。点击窗口的关闭按钮，则关闭监视窗口。



图 5-31 监视窗口

监视窗口中显示变量的内容：控制器名称、变量名称、在线值、修改、数据类型、AT、显示格式，其中控制器名称只在监视窗口 (工程) 有，而监视窗口 (控制器) 没有。

监视窗口登记变量有两种方法。

(1) 在监视窗口的名称栏输入变量名

在监视窗口的底部点击“输入名称”，如果使用监视窗口 (工程)，要指定控制器名称。输入变量名称的字符时，会显示候选变量名称的列表，从中选择变量名称。

(2) 从编辑器或变量表中拖放变量至监视窗口

从梯形图编辑器、ST 编辑器、全局变量表和本地变量表中，选择变量，将其拖放至监视窗口底部的“输入变量名”处。

5.3.3 改变当前值、设置/重置、强制刷新

在梯形图编辑器、ST 编辑器、监视窗口和 I/O 映射中，可以改变用户程序中使用的变量的当前值，使用设置/重置命令将程序输入和输出改为 TRUE 或 FALSE，使用强制刷新将外部

输入和输出强制为 TRUE 或 FALSE。

注意，当 Sysmac Studio 与 CPU 单元在线连接，进行改变变量当前值、设置/重置和强制刷新操作之前，一定要确认 NJ 所连接的输出单元、从站的安全性，避免引起所连设备的不正确运行，以至造成伤害。

1. 控制 BOOL 变量

在梯形图编辑器、监视窗口和 I/O 映射中均可控制 BOOL 变量，在 ST 编辑器通过改变变量当前值操作控制 BOOL 变量。下面介绍在梯形图编辑器中的操作。

1) 双击多视图浏览器中“编程”下的梯形图程序、梯形图功能或梯形图功能块，显示梯形图编辑器。

2) 如图 5-32 所示，右击输入或输出，若选择“设置/重置-设置”，输入或输出改为 ON；若选择“设置/重置-重置”，输入或输出改为 OFF。

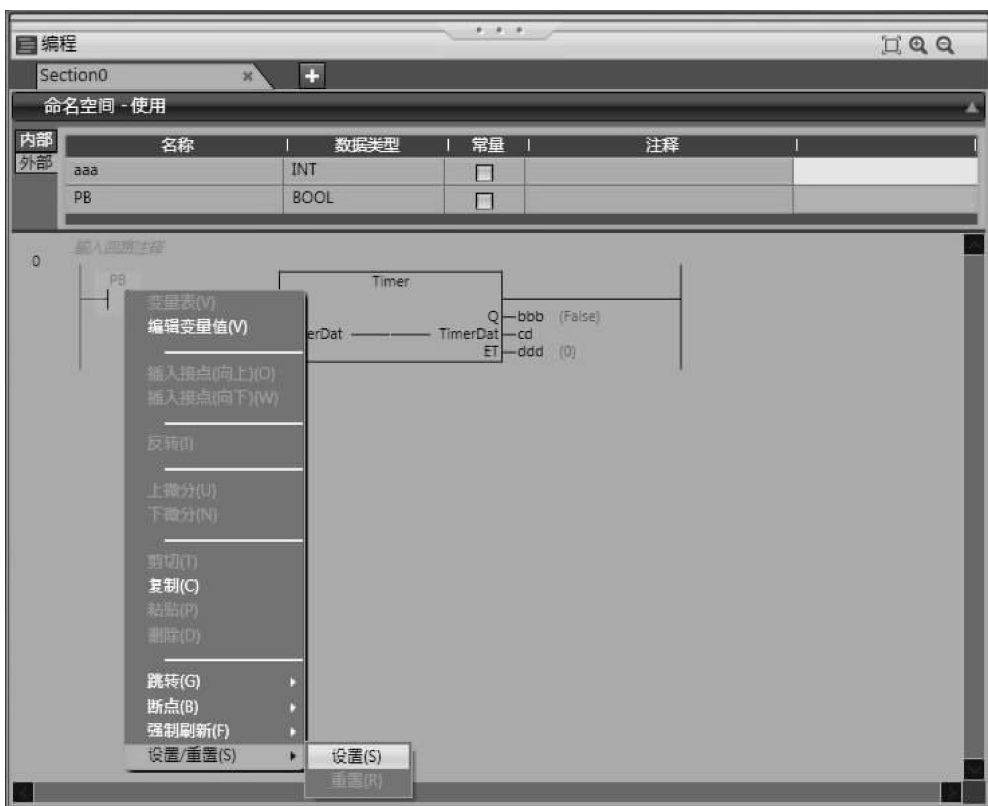


图 5-32 在梯形图编辑器中的设置/重置操作

2. 改变变量的当前值

需要时，可以在梯形图编辑器、ST 编辑器、监视窗口或 I/O 映射，改变用户定义变量、系统定义变量和设备变量的当前值。

在梯形图编辑器改变变量当前值的步骤如下。

1) 如图 5-33 所示，在梯形图编辑器，右击变量，选择“编辑变量值”，或右击变量的当前值。

2) 如图 5-34 所示, 输入新的值, 按回车键。

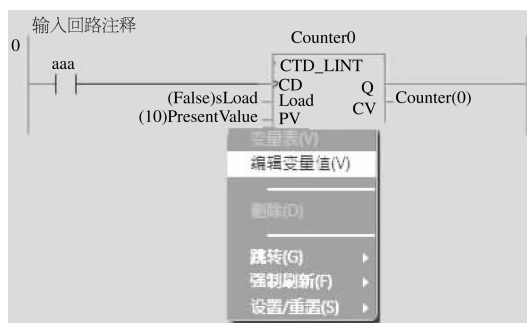


图 5-33 改变变量当前值

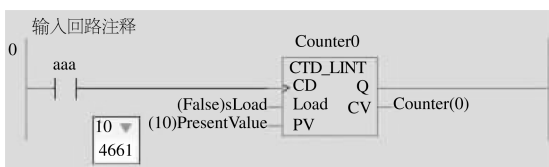


图 5-34 编辑变量值

3) 改变后的当前值如图 5-35 所示。

3. 强制刷新

强制刷新允许从 Sysmac Studio 用指定的值刷新外部输入和输出来调试程序。强制刷新不是对指定的设备变量执行的, 而是对 I/O 端口分配的设备变量。用强制刷新设定的值会一直保持, 直到从 Sysmac Studio 取消强制刷新。

在梯形图编辑器中, 强制刷新强制程序输入和输出为 TRUE 或 FALSE。解除强制刷新是取消输入和输出的强制 TRUE 或 FALSE 值。

在梯形图编辑器中, 强制刷新的步骤如下。

1) 在多视图浏览器中的“编程”下, 双击梯形图程序或梯形图功能块, 出现梯形图编辑器。

2) 如图 5-36 所示, 右击程序输入或输出, 若选择“强制刷新-TRUE”, 输入或输出强制为 ON; 若选择“强制刷新-FALSE”, 输入或输出强制为 OFF。

同样, 在 ST 编辑器、监视窗口、I/O 映射等视图中均可进行强制刷新或解除强制刷新。

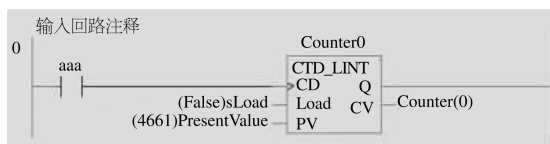


图 5-35 改变后的当前值

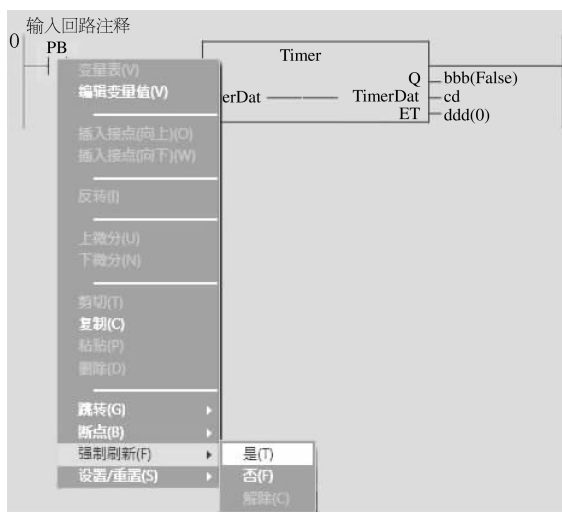


图 5-36 在梯形图编辑器中的强制刷新

5.3.4 交叉索引窗口

使用交叉索引可以观察到程序元素所在的程序及其位置。从这个列表中可以观察到元素所有出现的位置。元素可以为: 变量、数据类型、I/O 端口、功能、功能块。

从“视图”菜单中选择“交叉索引窗口”，会看到关于选中元素的列表，如图 5-37 所示，交叉索引窗口中为变量 PB 的列表，细节内容有：“变量或 I/O 端口”、“POU/定义”、“回路/行号”、“索引”。

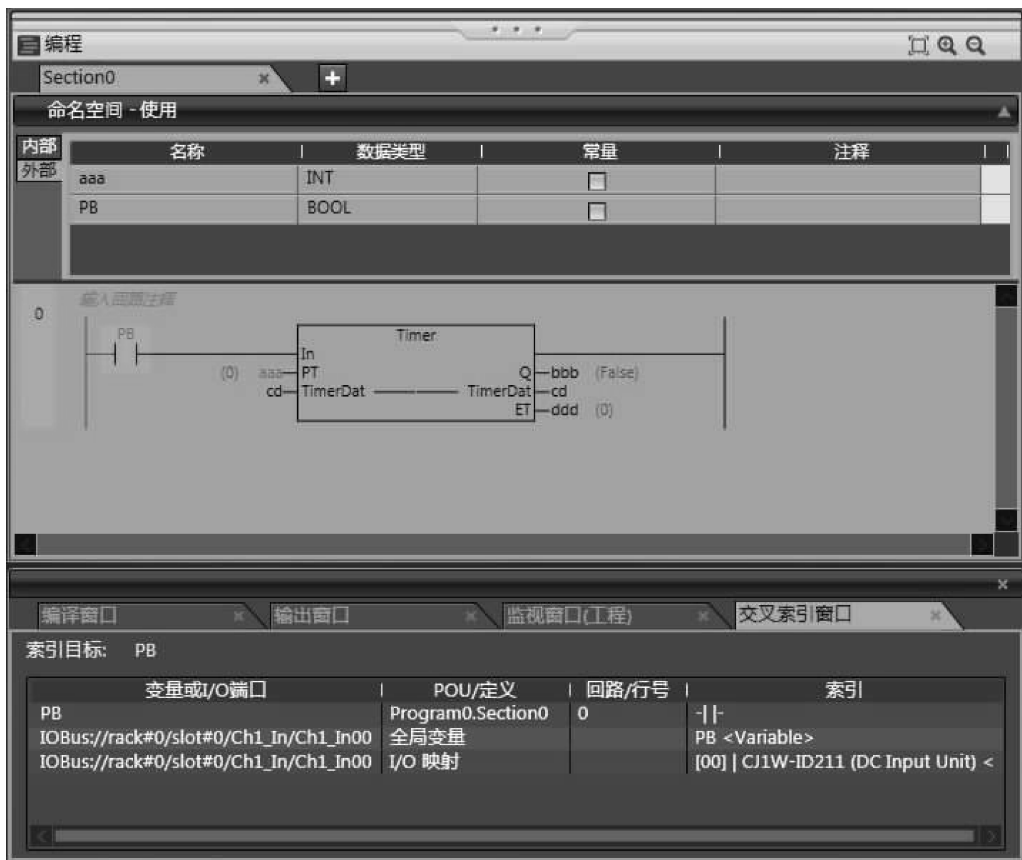


图 5-37 交叉索引窗口

可以从交叉索引窗口中查看使用的元素。根据要查看的项目，在列表中选择对应的行，该项目会出现在多视图浏览器中，在编辑面板中的位置将显示出来。

5.3.5 在线编辑

Sysmac Studio 使用在线编辑功能可以直接增加或修改 CPU 单元的程序部分。

梯形图的段、ST 编写的 POU (程序、功能或功能块) 和全局变量可以在线编辑。在线编辑状态下，可以改变定时器和计数器的设定值。

从“工程”菜单选择“在线编辑-启动”，启动在线编辑。修改完成后，从“工程”菜单选择“在线编辑-传送”，将在线编辑的结果传送到控制器或仿真器。

在线编辑期间，如果从“工程”菜单选择“在线编辑-取消”，所有修改全部取消。

多个用户可以同时在线编辑，但实际操作时有一些限制条件。

在线编辑时，不能从变量表中删除变量；可以修改变量的初始值、注释，但不能修改变量的其他属性。

需要提醒用户注意，有了在线编辑可以边修改边运行，虽然带来一些方便，但要慎重使用。特别是现场调试时，因为控制器有输出，一定要确认不会出现意外。如果没有把握，最好不用。

5.3.6 改变操作模式

NJ 系列控制器有运行和编程两种操作模式。两种操作模式可以相互转换。改变 NJ 操作模式之前，要确认系统的安全性，避免出现意外。

改变操作模式时，除了属性设置为“保持”的变量，其他所有 I/O 设备的值都被清零。

即使在编程模式，I/O 设备仍然被刷新。当改变操作模式时，要注意对 I/O 设备运行的影响。

从“控制器”菜单中选择“模式-运行模式”，则控制器改为运行模式，如图 5-38 所示；若选择“模式-编程模式”，则控制器改为编程模式，如图 5-39 所示。也可以通过命令快捷键进行上述操作。



图 5-38 改为运行模式



图 5-39 改为编程模式

5.3.7 监视控制器状态

可以监视连接的控制器，并将结果显示在状态监视面板。只有与控制器在线或与模拟器连接后，才能观察状态监视面板。

状态监视面板显示在工具箱的下方，与控制器在线连接后的显示如图 5-40 所示，与模拟器连接后的显示如图 5-41 所示。



图 5-40 与控制器连接



图 5-41 与模拟器连接

使用状态监视面板右上角的按钮 (🔍 🖨️)，视图可以在基本和详细之间转换，图 5-42 所示为与控制器连接的详细视图，图 5-43 所示为与模拟器连接的详细视图。



图 5-42 与控制器连接的详细视图



图 5-43 与模拟器连接的详细视图

5.3.8 任务执行状态监视

可以监视 NJ 系列控制器或模拟器中正在运行的任务的执行状态。

在多视图浏览器中，双击“配置和设置”下的“任务设置”，在显示的编辑面板点击“任务执行状态监视”按钮，可以观察到所有正在执行任务/程序的名称和执行状态，如图 5-44 所示。

5.3.9 任务执行时间监视

当用户程序在 NJ 控制器或模拟器上运行时，可以监视每一个任务的执行时间。与模拟器连接时，可以监视任务的实际处理时间，进行控制器性能测试。

如图 5-45 所示，与控制器在线连接后，双击多视图浏览器中“配置和设置”下的“任务设置”，在显示的编辑面板点击“任务执行时间监视”按钮，可以观察到所有正在执行任务的下列信息：最小值、平均值、最大值、设置周期、超出周期、任务执行次数和超出任务

周期次数。注意，超出任务周期次数是任务执行的时间超出了任务周期的次数，这和发生超出任务周期错误的次数是不一样的。



图 5-44 任务执行状态监视



图 5-45 任务执行时间监视

与模拟器连接时，任务执行从开始到结束需要的时间可以通过模拟估算出来。

- 1) 可以显示估算的平均执行时间，最大的任务执行时间，每个周期任务执行和系统服务时的 CPU 使用情况。
- 2) 可以将估算的时间作为确定任务执行周期的指南。
- 3) 可以在执行时间估计模式下运行模拟器。

5.3.10 轴状态监视 (MC 监测表)

轴状态监视是用来监视轴的错误状态、输入信号状态、命令值和实际值。为了监视轴状态，轴必须登记并传送到控制器。

与控制器在线连接后，右击多视图浏览器中的“配置与设置-运动控制设置”下的“轴设置”，从菜单里选择“MC 监测表”，即显示轴状态监视，如图 5-46 所示。

轴名称	1 MC_Axis000(0)	2 MC_Axis001(1)	3 MC_Axis002(2)	4 MC_Axis003(3)
▼ Cfg				
AxNo	0	1	2	3
AxEnable	使用的轴	使用的轴	使用的轴	使用的轴
AxType	伺服轴	伺服轴	虚拟伺服轴	虚拟伺服轴
NodeAddress	2	3		
▼ Status				
Ready	0	0	1	1
Disabled	1	1	0	0
Standstill	0	0	1	1
Discrete	0	0	0	0
Continuous	0	0	0	0
Synchronized	0	0	0	0
Homing	0	0	0	0
Stopping	0	0	0	0
ErrorStop	0	0	0	0
Coordinated	0	0	0	0
▼ Details				
Idle	1	1	1	1
InPosWaiting	0	0	0	0
Homed	0	0	0	0
InHome	0	0	0	0
VelLimit	0	0	0	0
▼ Dir				
Posi	1	1	1	1
Nega	0	0	0	0
▼ DrvStatus				

图 5-46 MC 监测表

可以通过操作改变监视的轴。打开轴状态监视，在面板的任何位置右击，从菜单中选择“选择轴显示”，从随后出现的对话框中选择要监视的轴，点击“确定”按钮。

5.3.11 数据跟踪

可以对指定的变量进行数据跟踪，无需任何额外的编程。可以在两种连续跟踪方法中选择：触发跟踪或连续跟踪，跟踪时采集保存的数据总量为 10000 个。触发跟踪设置触发条件并且符合条件之前和之后的数据被保存，之前和之后的数据各占多少由后触发数据比率设置决定；连续跟踪执行连续采样没有任何触发条件，结果存储在计算机的文件里。当然，即使使用触发跟踪，仍然可以显示 Sysmac Studio 保存的数据并把这些数据存储到一个文件里。同样的功能也适用于模拟器。

1. 数据跟踪的操作

(1) 执行数据跟踪

1) 右击多视图浏览器中的“配置与设置”下的“运动跟踪设置”，从菜单里选择“添加-数据跟踪”，DataTrace0 添加到多视图浏览器中。

2) 双击 DataTrace0，双击新的数据跟踪设置，或右击设置，从菜单中选择“编辑”，显示数据跟踪视图，如图 5-47 所示。

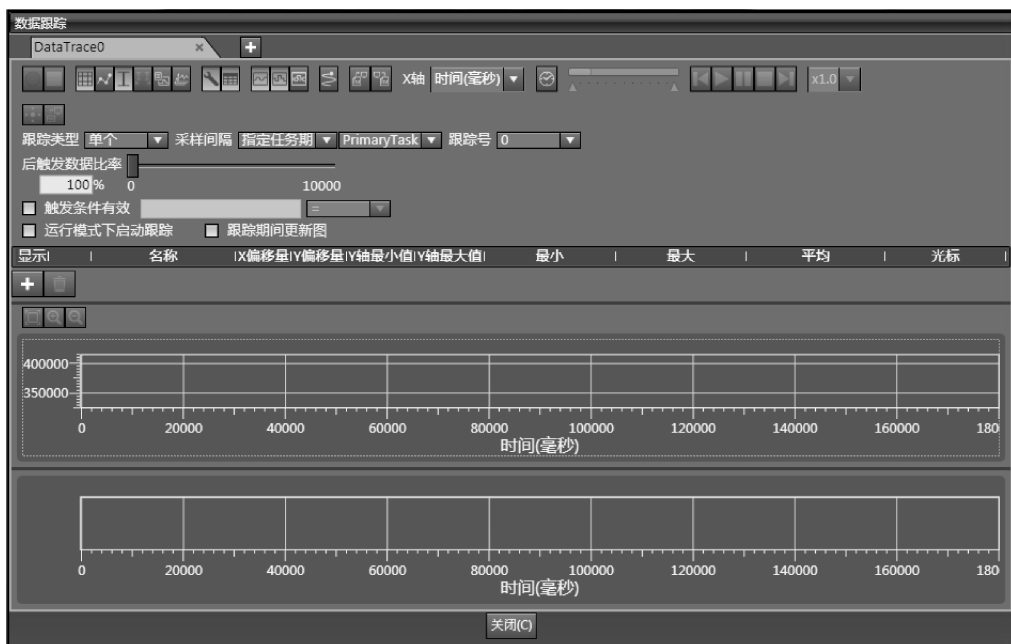


图 5-47 数据跟踪视图

(2) 数据跟踪设置

需要设置以下项目：跟踪类型、采样间隔、后触发数据比率、触发条件、跟踪的变量等。

(3) 启动和停止跟踪

将数据跟踪设置传送到控制器，开始跟踪。如果选择触发 (单个) 作为跟踪类型，跟踪会等待触发信号到来才开始采样。如果选择连续作为跟踪类型，采样立即开始并且所有跟踪数据传送到计算机，被收集并保存到一个文件中。

与控制器在线连接后，数据跟踪视图顶部工具栏如图 5-48 所示，点击“执行”或“停止”按钮，即可启动或停止数据跟踪。



图 5-48 数据跟踪工具栏

(4) 显示跟踪结果

在图或三维运动跟踪显示模式下，可以查看跟踪数据的结果。采样开始后，采样数据立即传送并绘制在图上。跟踪目标变量表显示每一个变量的最大、最小和平均值。

可以在三种不同类型的图上进行切换：BOOL 数据类型的数字图、其他类型数据的模拟图和观察轴运行的 3D 运动跟踪图。图 5-49 所示为模拟图，图 5-50 所示为数字图。

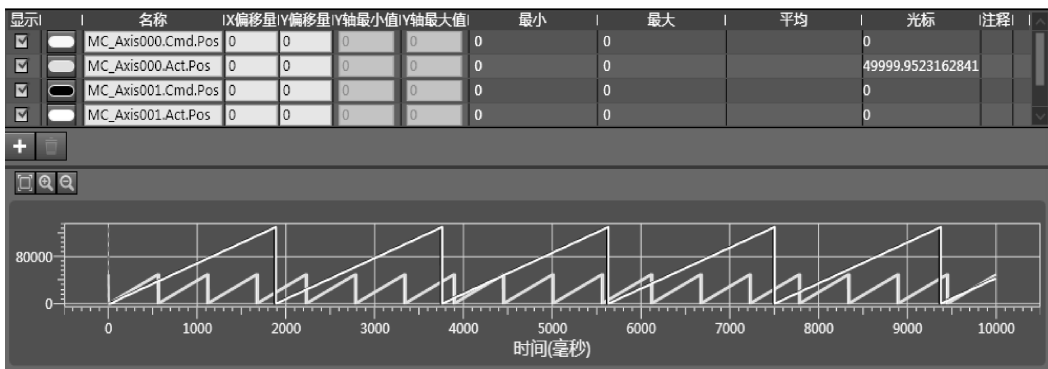


图 5-49 模拟图

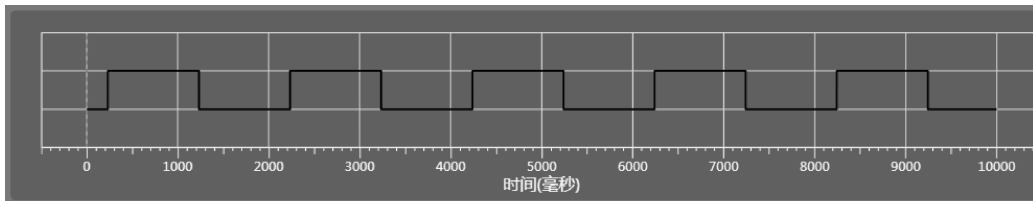


图 5-50 数字图

(5) 导出跟踪结果

在 Sysmac Studio 上保存工程时，跟踪结果会自动保存在工程里。如果想把数据单独存为一个文件，可以将数据导出到一个 CSV 文件。

(6) 导入跟踪结果

如果想导入跟踪结果或者设置，可以导入一个包含跟踪结果数据的 CSV 文件。

(7) 打印跟踪结果

可以打印出数据跟踪设置以及数字和模拟图。

2. 3D 运动跟踪显示模式

为 3D 设备模型中的每个元素设置轴变量，然后根据这些轴运动设置 3D 设备的运动。这个运动显示到数据跟踪时间曲线图上，也可以同时在 3D 设备上显示标记的路径，还可以为每个 3D 显示的投影显示一个二维的路径标记。

显示 3D 运动跟踪模式的步骤如下：

- 1) 在多视图浏览器添加轴，然后设置轴变量参数。
- 2) 创建一个操作设备的程序。
- 3) 创建数据跟踪的 3D 设备模型。

- 4) 登记触发设置和跟踪变量。
- 5) 与控制器在线连接, 将操作模式改为“RUN”模式, 执行用户程序。
- 6) 启动数据跟踪, 采样数据。
- 7) 在数据跟踪窗口检查跟踪结果。
- 8) 在数据跟踪窗口点击“开始跟踪”按钮, 运动开始。

点击“录像”按钮, 将3D设备模型运行存储为AVI文件。点击“停止”按钮, 停止录像。

除了3D显示外, 还有2D显示。在3D运动跟踪显示的切换框中选择X-Y、Y-Z或X-Z, 即可进行3D显示投影的2D标记路径的显示。

3. 创建3D设备模型

创建3D设备模型的步骤如下:

- 1) 启动数据跟踪, 点击显示3D运动跟踪按钮.
- 2) 点击3D设备模型设置按钮, 从菜单中选择“添加”, 出现3D设备模型列表, 如图5-51所示, 从中选择。

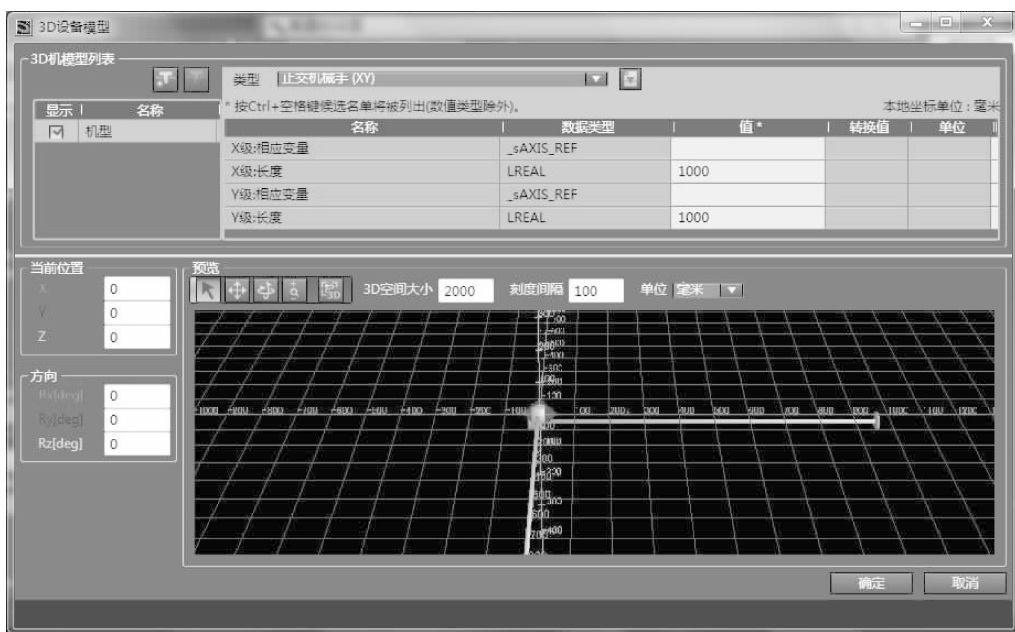


图 5-51 3D 设备模型列表

3) 在3D设备模型显示底部的预览区点击“全显示3D设备模型”按钮, 这样可以从预览区上空观察整个设备模型, 如图5-52所示。

4) 设置3D设备模型参数, 点击“确定”按钮, 添加了3D设备模型。

图5-53所示为3D运动跟踪的轨迹图。

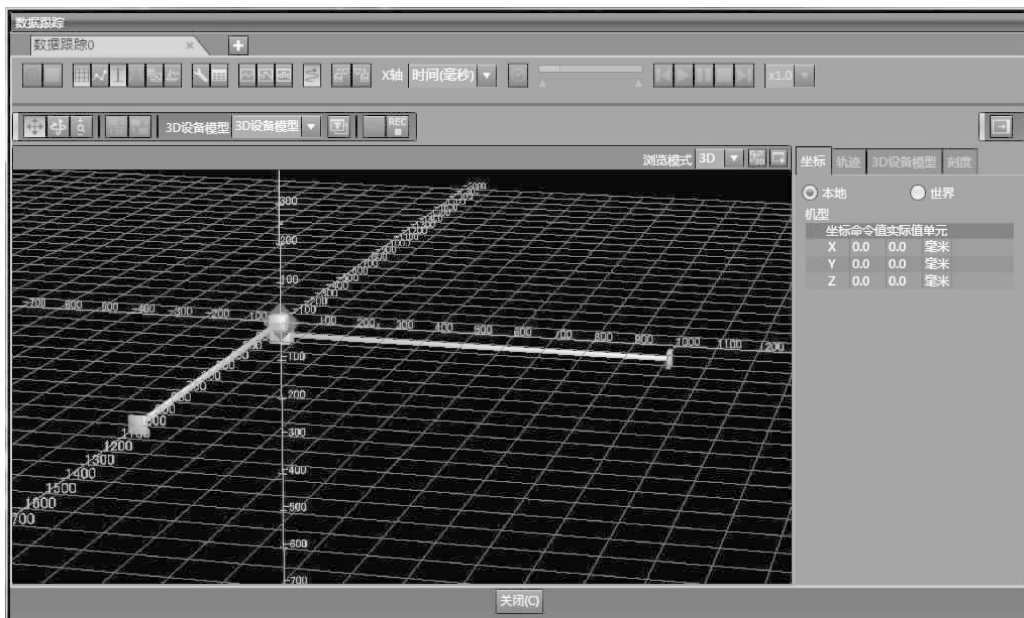


图 5-52 创建 3D 设备模型

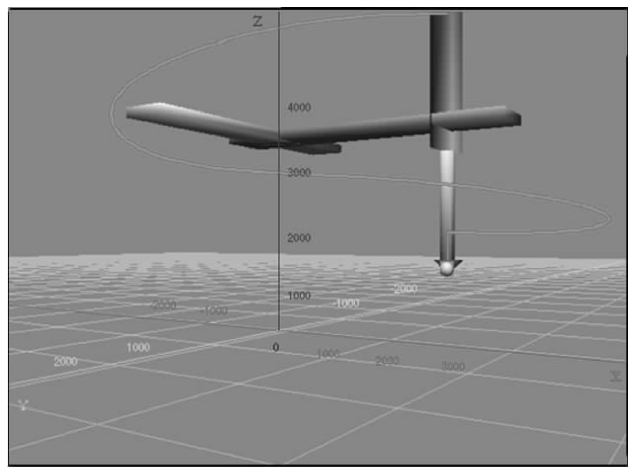


图 5-53 3D 运动跟踪的轨迹图

5.4 离线调试

离线调试在模拟器上进行，Syamac Studio 不连接控制器。将工程传送到实际的控制器之前，可以在模拟器上检查控制程序的逻辑是否正确。

5.4.1 程序模拟调试

要调试控制程序，最好先是在计算机上仿真控制程序，检查运行逻辑和参数设置，这个过程称为模拟仿真调试。Syamac Studio 配备了一个模拟器，可以模拟 CPU 单元几乎所有的

功能。

1. 仿真步骤

1) 启动 Sysmac Studio, 创建工程。

2) 如图 5-54 所示, 从“视图”菜单选择“模拟画面”, 模拟面板显示在窗口右侧的工具箱面板之下, 如图 5-55 所示。

3) 使用 Sysmac Studio 进行控制器配置和设置, 创建程序。

4) 从“工程”菜单选择“编译”, 编译用户程序。只要程序改变, 就从“工程”菜单选择“编译”再次编译程序。若程序没有编译, 不能连接模拟器。

5) 如果要执行指定的任务或仅一个段或程序的部分, 在多视图浏览器的“编程-任务”下, 选择要模拟的项目。



图 5-54 选择模拟画面

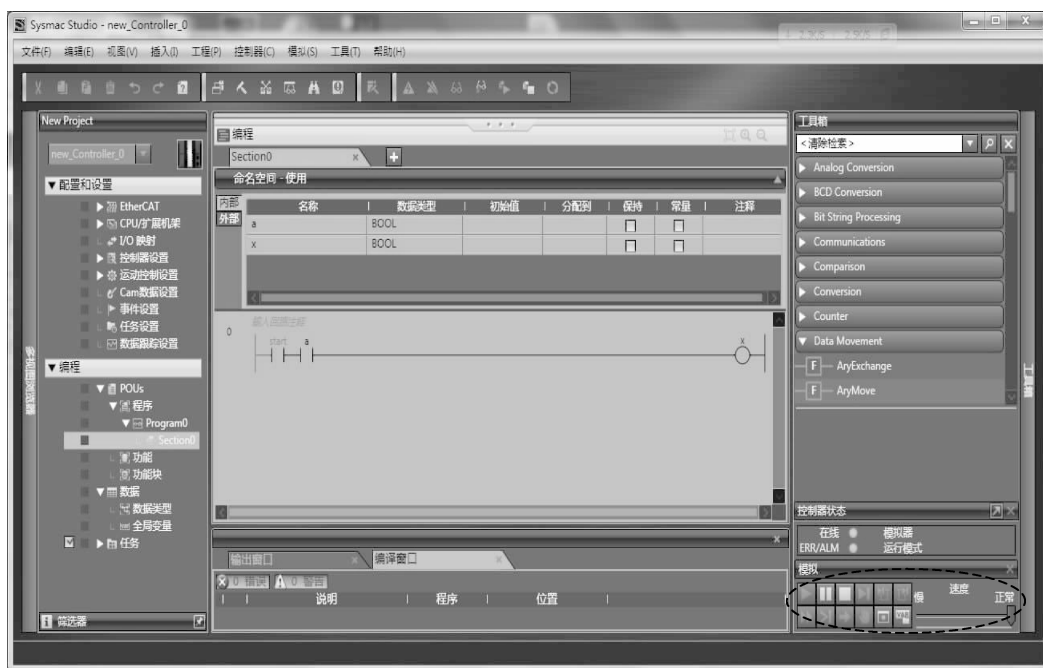


图 5-55 模拟器

6) 如图 5-56 所示, 点击模拟面板的“运行”按钮, 从“模拟”菜单中选择“执行”。

7) 模拟器启动和连接后, 状态监视在线指示灯点亮呈绿色, 出现“模拟器”显示。在这种状态下, 该工程全部传送到模拟器中, 以“RUN”模式运行。

8) 停止模拟时, 点击模拟面板的“停止”按钮。与模拟器的连接中断, 退出模拟。

9) 要结束模拟, 点击模拟面板上的关闭按钮 (×)。

2. 设置模拟程序

设置要模拟的任务或程序。可以选择用户程序中的部分或全部程序进行模拟。

从“视图”菜单选择“模拟画面”后，会显示模拟面板，同时，在多视图浏览器中的“编程-任务”的左侧，出现了一个复选框，供用户选择要模拟仿真的任务、程序或段。

3. 设置调试程序

调试程序是专用于离线调试的程序代码。它包含对接收到的控制器外部输入进行虚拟输入处理和进行强置用户定义错误的指令，还包含完成其他调试任务的指令。可以为每个程序设置调试属性，设置为调试属性的程序只能在模拟器上执行，不能在控制器上执行。如果要在控制器上执行，需改变其属性为正常程序。调试程序和正常程序一样，将它们分配给任务执行。使用调试程序可以提高估计任务执行时间的准确性。

4. 估计任务执行时间

任务执行从开始到结束所需要的时间可以近似从模拟仿真得到。估计的值与实际控制器的任务执行时间会有所不同。估计的时间可以作为确定任务周期的一个指南。

先从“模拟”菜单选择“校准”，完成后回到“模拟”菜单，选择“执行时间估计模式”，再进入“任务设置”，选择“任务执行时间监视”，就可以看到任务执行时间的估计值。

5. 模拟操作

模拟面板如图 5-56 所示。

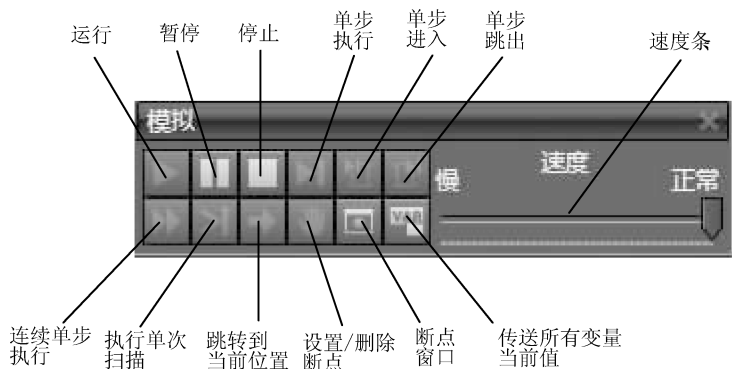


图 5-56 模拟面板

在模拟面板上做如下操作。

(1) 改变模拟的速度

调试时，如果要想模拟器的运行比正常速度慢，拖动模拟面板中的模拟速度条，更改模拟速度，从 0.1% ~ 100% 可调。在模拟进行或停止时，都可以更改模拟速度。

(2) 设置断点

可以设置断点停止模拟的执行，例如，观察一段特定程序执行后的状态。

(3) 单步执行

在 ST 编辑器里，一次执行一行程序。或在梯形图编辑器中，一次执行一条指令。

(4) 连续单步执行

当程序停在断点、暂停或单步执行时停止，点击连续单步执行按钮，在固定时间 (1s)

内，光标移动到下一步（在 ST 编辑器）或下一条指令（在梯形图编辑器）。移动到下一步需要的时间取决于程序。

(5) 暂停

暂停程序的执行，但是模拟器并没有关闭。

(6) 单步进入

在功能或功能块内部，完成源代码的单步执行。

(7) 单步跳出

执行时，离开功能和功能块当前的步。

(8) 单周期执行

执行当前的任务一个周期，在下一个周期的程序开始处暂停执行。

(9) 设置条件断点

条件断点指在程序中设置的断点处设置条件表达式。

(10) 变量当前值批传递

模拟时，可以将变量在特定时间的值保存到文件中，或者将存储在文件中变量的值写回到模拟器中。这允许启动模拟前写入变量的初始值。进行模拟的程序中保持和非保持类型的变量都可以保存或写入。

5.4.2 离线调试顺序控制和运动控制程序

在数据跟踪窗口，可以形象观察顺序控制和运动控制程序的运行。连接到仿真器后，利用数据跟踪窗口离线调试。在数据跟踪窗口可以使用以下三种格式。

(1) 每个变量的时间曲线图

(2) 每个轴的路径曲线

(3) 3D 图形显示的 3D 机器运行

为了将运动控制的轴变量和 3D 设备模型的机器模型联系起来，必须进行轴设置。

5.4.3 NS 系列 PT 的离线模拟调试

可以模拟 NJ 顺序程序和 NS 系列 PT 链接的运行，离线调试顺序程序和画面数据。NS 系列 PT 的画面数据由 CX-Designer 的支持软件创建。

操作开始时，从“模拟”菜单，选择“启动 NS 统合模拟”，如图 5-57 所示。



图 5-57 启动 NS 统合模拟

5.5 在线调试

将 Sysmac Studio 与 NJ 控制器建立在线连接后，就可以检查和调整用户程序的运行了。

在线调试时，控制器有输出或通信，所连的外部设备会有动作，因此，在线调试之前，一定要对系统安全性进行确认，避免出现意外。

5.5.1 同步（自动检验后上载/下载）

同步是指计算机中 Sysmac Studio 的数据与 NJ 控制器中的数据自动比较，并按用户指定的方向传递数据。同步有两种操作。

- 1) 下载：从“控制器”菜单选择“同步”，点击“传递到控制器”按钮。
- 2) 上载：从“控制器”菜单选择“同步”，点击“从控制器传递”按钮。

1. 自动检验步骤

1) 将计算机与实际控制器连接，从“控制器”菜单选择“在线”，如图 5-58 所示，或点击命令快捷键 **A**。

2) 从“控制器”菜单选择“同步”，如图 5-59 所示，或点击命令快捷键 **Q**。



图 5-58 在线



图 5-59 同步

在工具箱面板的同样位置显示同步面板，如图 5-60 所示，Sysmac Studio 与控制器之间的用户程序及参数设置的检验启动。可以点击“取消”按钮取消该操作。



图 5-60 同步比较

2. 检验结果

检验结果显示在图 5-61 中。

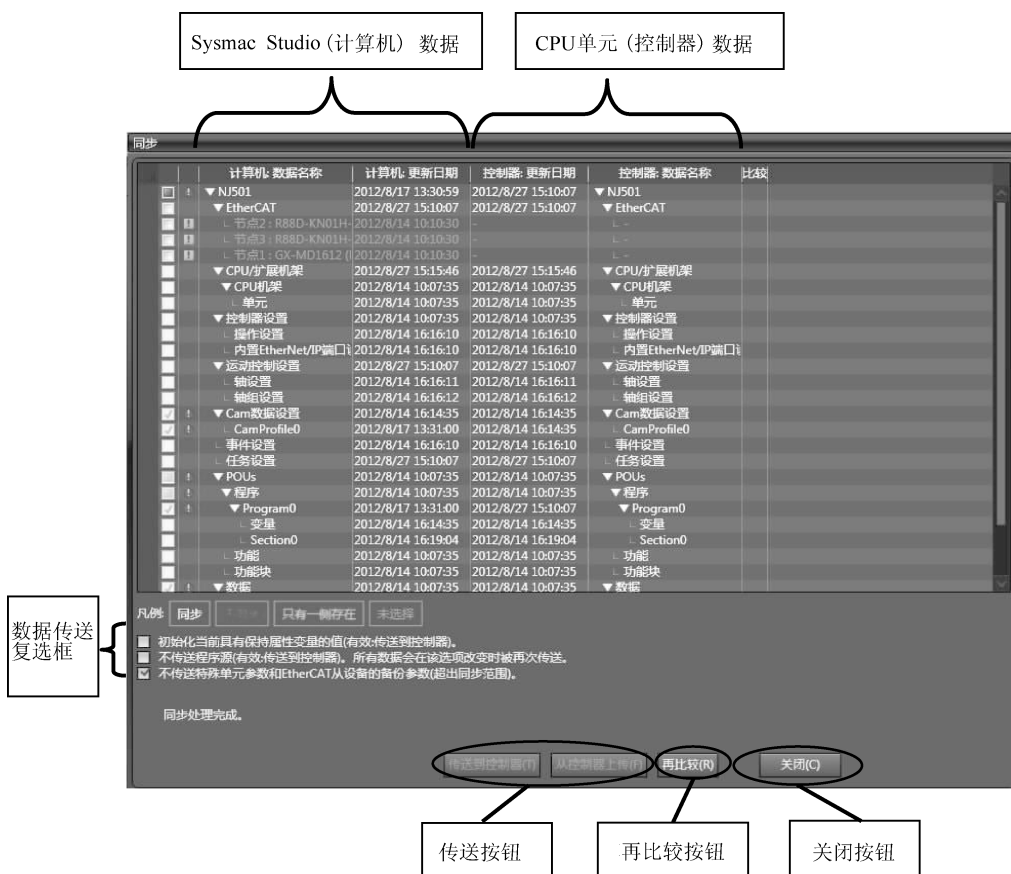


图 5-61 检验结果

图 5-61 上半部显示的主要内容有三部分。

1) Sysmac Studio (计算机) 数据有两列: “计算机: 数据名称” 和 “计算机: 更新日期”。其中, “计算机: 数据名称” 是 Sysmac Studio 工程中各个项目的名称, “计算机: 更新日期” 是 Sysmac Studio 工程中各个项目最后更新的时间。

2) CPU 单元 (控制器) 数据有两列: “控制器: 更新日期” 和 “控制器: 数据名称”。其中, “控制器: 更新日期” 是 CPU 单元工程中各个项目最后更新的时间, “控制器: 数据名称” 是 CPU 单元工程中各个项目的名称。

3) 比较: 经同步后, Sysmac Studio 和 CPU 单元中不相同的 POU: (程序、功能和功能块) 项目, 会在 “比较” 栏对应的行出现比较按钮 , 点击后显示详细比较结果。

图 5-61 的左侧有一栏出现不同颜色的! 图标, 表示比较结果。其中, 白色为 “同步”, 表示 Sysmac Studio 和 CPU 单元对应的项目完全相同; 红色为 “不同步”, 两方的项目不相同, 有差异; 绿色为 “只有一方存在”, 表示项目只存在于其中一方, 另一方没有; 灰色为 “不检查”, 表示工程创建未完成或者是非同步数据。

图 5-61 中的数据传送复选框有三项。

- 1) 清除当前具有保持属性的变量值 (传送到控制器时有效)。
 - 2) 不传送程序源 (传送到控制器时有效)。所有数据会在该选项改变时被再次传送。
 - 3) 不传送特殊单元参数和 EtherCAT 从站的备份参数 (超出同步范围)。
- 前两项默认为“不选中”，最后一项默认为“选中”。

3. 上载和下载数据

如果计算机的数据是正确的，点击“传送到控制器”按钮，完成下载。

如果控制器的数据是正确的，点击“从控制器上传”按钮，完成上载。

传送完成后，Sysmac Studio 的数据和 NJ 控制器的数据将取得一致。

5.5.2 同步、传送和检验 EtherCAT 配置

这些功能允许用户通过各种方法将 Sysmac Studio 数据和 NJ 控制器数据同步。可以将 NJ 控制器的实际网络配置传送到 Sysmac Studio，将从站的后备参数设置在 Sysmac Studio 和 NJ 控制器之间传送。

同步、传送和检验 EtherCAT 配置的详细操作见 6.3 节。

5.5.3 同步、传送和 CPU/扩展机架组件比较

这些功能允许用户通过各种方法将 Sysmac Studio 数据和 NJ 控制器数据同步。可以将 NJ 控制器的实际单元配置传送到 Sysmac Studio，将特殊单元设置在 Sysmac Studio 和 NJ 控制器之间传送。

5.5.4 测试轴运行 (MC 试运行)

不需要任何梯形图程序，直接从控制器发送命令，操作电动机，以检查接线，或对已进行轴分配的 OMRON 伺服驱动完成简单的运行测试。

MC 试运行期间，只有 Sysmac Studio 能够控制运行，而运动控制指令的命令将被忽略。

5.5.5 复位控制器

复位控制器模拟控制器断电再上电时的操作和状态。只能在编程模式下完成。在运行模式下，不能复位控制器。与模拟器连接时，也不能复位控制器。

复位时从“控制器”菜单选择“复位控制器”，复位操作完成后，Sysmac Studio 离线。

5.6 故障分析

Sysmac Studio 提供非常强大的故障分析功能。故障分析用来检查控制器出现的错误，显示修正错误方法和清除错误。与 NJ 控制器在线连接后，控制器当前的错误和事件日志中的内容显示在故障诊断和事件日志对话框中。

从“工具”菜单中选择“故障分析”，或点击工具条中的“故障分析”按钮。显示故障分析对话框，如图 5-62 所示。

点击图 5-62 中顶部的 4 个选项卡，可分别显示“控制器错误”、“控制器事件日志”、“用户定义错误”、“用户定义事件日志”窗口，详细情况见有关手册。



图 5-62 故障分析对话框

第 6 章 NJ 的 EtherCAT 网络

6.1 概述

EtherCAT (Ethernet Control Automation Technology) 是一种基于 Ethernet 的高性能工业网络系统，具有更快捷、更高效的通信功能。

6.1.1 EtherCAT 特点

1. EtherCAT 的特点

1) 具有 100Mbit/s 超高速通信功能。网络中每个节点都可以高速传送 Ethernet 帧，从而缩短通信周期。

2) 与 Ethernet 有极高的兼容性。EtherCAT 是一种独立的通信协议，但其在物理层采用标准 Ethernet 技术，因此可以使用普通 Ethernet 电缆。EtherCAT 无论应用于需要高速处理速度和系统集成的大型控制系统，还是应用于小型和中型控制系统，都能充分体现出有效性。

2. EtherCAT 通信机制

图 6-1 所示为 EtherCAT 通信机制。

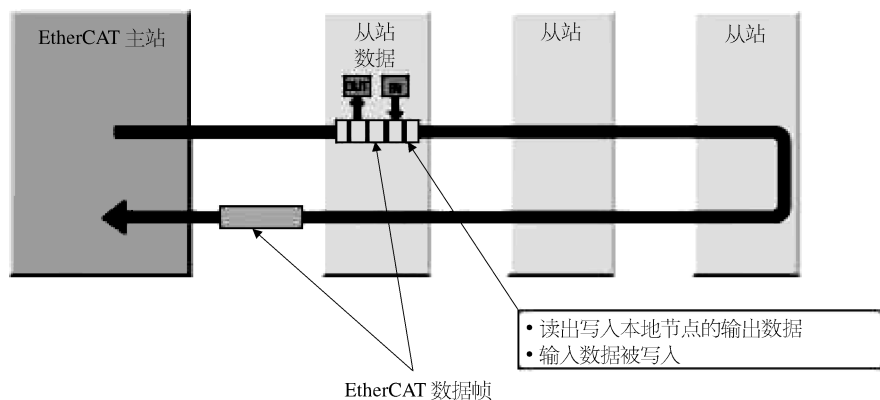


图 6-1 EtherCAT 通信机制

EtherCAT 网络中，主站并不是把数据发送到单独的节点，而是将 Ethernet 帧传递到所有的从站节点。

当帧传递到一个从站节点时，该从站会在帧分配给它的特定区域里读取和写入数据，这个过程的时间只要几纳秒。

Ethernet 帧由 EtherCAT 主站向所有的 EtherCAT 从站发送，中途不停顿。每当到达最后一个从站时，帧就会被返回，再次穿过所有的从站并最终回到 EtherCAT 主站。

以上通信机制保证了高速和实时的数据传送。

3. NJ 系列 CPU 单元 EtherCAT 的特点

(1) CPU 单元处理周期和 Ethernet 通信周期同步

CPU 单元中的顺序处理和运动处理的周期与 EtherCAT 的过程数据通信周期相匹配。这就为高精度的顺序控制和运动控制提供一个稳定的固定周期。

(2) 通过设备变量进行数据访问，无需考虑地址问题

使用设备变量访问 EtherCAT 从站，方式和访问 NJ 系列 CPU 机架和扩展机架上的单元一样。用预先准备的结构体类型轴变量访问伺服驱动和编码器输入从站各种类型的数据。这样访问 EtherCAT 网络中的从站时，不需要考虑地址。

(3) 基于统一的规格实现最佳的功能和方便的操作

NJ 系列机器自动化控制器与 Sysmac 设备以及 Sysmac Studio 自动化软件一起使用，可以达到最佳的功能和方便的操作。

6.1.2 NJ 的 EtherCAT 网络配置

NJ 的 EtherCAT 网络配置如图 6-2 所示。

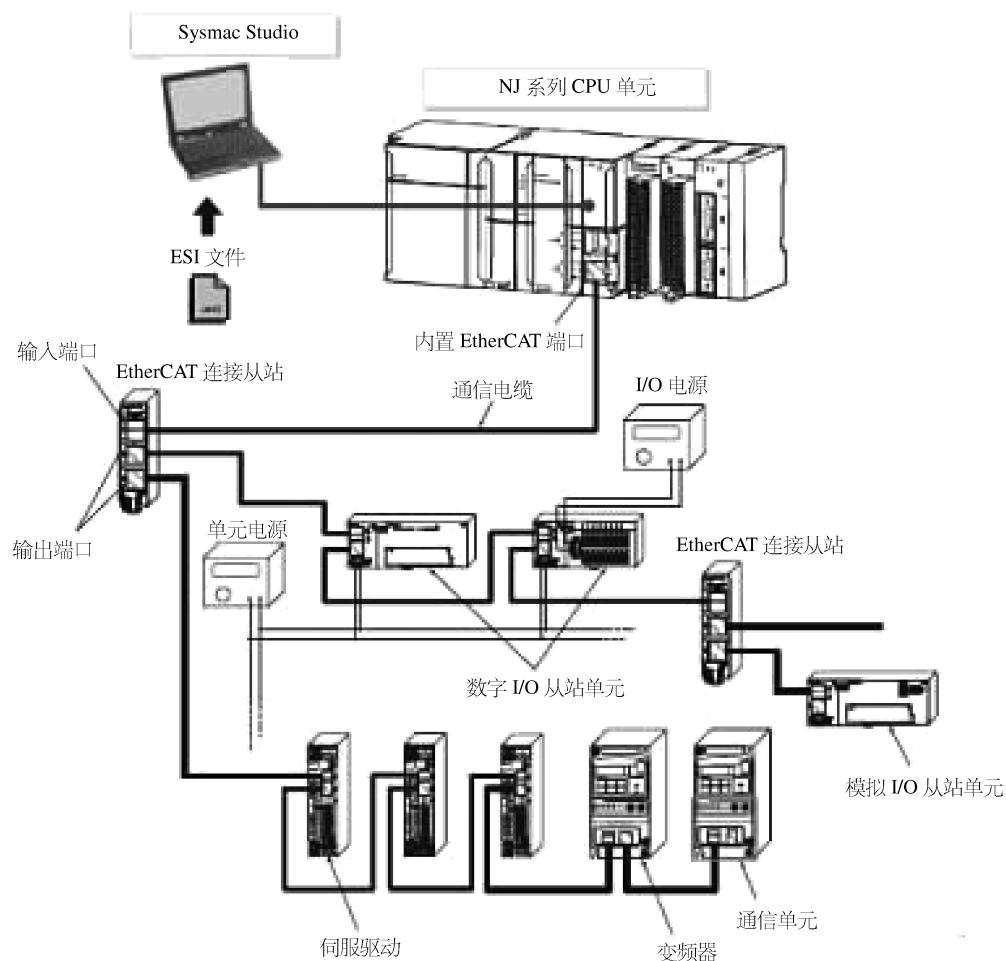


图 6-2 NJ 的 EtherCAT 网络配置

网络中的各种设备说明如下。

1. EtherCAT 主站

EtherCAT 主站管理网络，监视从站状态，与从站交换 I/O 数据。它有一个输出端口。

2. 输出端口

输出端口把 EtherCAT 的通信数据传送到其他设备。用输出端口连接另一台设备时，总是将它连接到另一台设备的输入端口。如果连接到另一台设备的输出端口，则不能正常通信。

3. 输入端口

这个端口用来输入 EtherCAT 通信数据。总是将它连接到另一台设备的输出端口。如果连接到另一台设备的输入端口，则不能正常通信。

4. EtherCAT 从站

每个 EtherCAT 从站输出从 EtherCAT 网络中接收到的 EtherCAT 主站输出的数据。它也通过 EtherCAT 网络发送输入数据给 EtherCAT 主站。在 EtherCAT 网络中，可以使用总共 192 个从站节点。从站可以实现同步，即使网络中有多种非同步从站和连接从站。从站有一个输入端口和至少一个输出端口。从站的节点地址在 1 ~ 192 之间，可以在这个范围内任意分配，无需考虑从站的类型，但是每个节点地址只能用于一个从站。每个从站都要设置节点地址，而连接从站要设置多个节点地址。

(1) 非同步从站

这些从站完成顺序控制，包括数字从站、模拟从站等，它们之间不需要同步。每个非同步从站有一个输入端口和一个输出端口。

(2) 同步从站

这些从站的运行与分布时钟 (DC) 同步。包括伺服驱动和编码器输入从站，每个同步从站有一个输入端口和一个输出端口。

欧姆龙的 EtherCAT 从站见表 6-1。

表 6-1 欧姆龙的 EtherCAT 从站

名称	型 号	同步/非同步从站	分配轴
数字 I/O 从站	GX-□D16□1/OC1601	非同步从站	不可以
	GX-ID16□2/OD16□2/MD16□2	非同步从站	不可以
	GX-□D16□8/□32□8	非同步从站	不可以
模拟 I/O 从站	GX-AD0471/DA0271	非同步从站	不可以
多功能、紧凑型变频器	配有 EtherCAT 通信单元的 3G3MX2 3G3AX-MX2-ECT	非同步从站	不可以
AC 伺服驱动	R8MM-K/R88D-KN□-ECT	同步从站	可以
编码器输入从站	GX-EC0211/EC0241	同步从站	可以

(3) EtherCAT 连接从站

这是一个用来给 EtherCAT 网络布线分支的特殊单元。EtherCAT 连接从站要设置节点地址。

每个连接从站有一个输入端口和多个输出端口。每个连接从站的输出端口都能连接到另一个连接从站或 EtherCAT 从站。

OMRON 的 EtherCAT 连接从站现有 GX-JC03、GX-JC06 两个型号，分别有 3 个、6 个端口。GX-JC06 要设置 2 个节点地址。

5. Sysmac Studio

Sysmac Studio 在个人计算机上运行，用来配置 EtherCAT 网络和从站，以及用来编程、监视和调试控制器。

6. 通信电缆

使用 Ethernet 5 类 (100Base-TX) 或更高标准的屏蔽双绞线电缆。

7. ESI (EtherCAT Slave Information) 文件

ESI 文件包含 XML 格式的 EtherCAT 从站特定信息。将 ESI 文件导入 Sysmac Studio 后，就能方便地为从站分配过程数据和进行其他设置。

ESI 文件定义了 EtherCAT 从站单元的设置信息。各种 EtherCAT 通信设置是基于相连从站单元的 ESI 定义。OMRON 从站单元的 ESI 文件已预先导入 Sysmac Studio 中，如果 EtherCAT 主站要与其他厂商的从站单元连接，需其他厂商提供各自从站单元的 ESI 文件。可以从 Sysmac Studio 导入其他厂商的 ESI 文件，并利用它们编辑 EtherCAT 配置，创建新的网络配置信息，然后下载到 EtherCAT 主站中，配置 EtherCAT 网络。

8. 单元电源

为从站通信和内部操作供电。

9. I/O 电源

为从站通信和内部操作供电。

6.1.3 内置 EtherCAT 端口的规范

内置 EtherCAT 端口的规范见表 6-2。

表 6-2 内置 EtherCAT 端口的规范

项 目	规 范	项 目	规 范
通信协议	EtherCAT 协议	从站最大数目	192
支持的服务	CoE (过程数据通信和 SDO 通信)	最大过程数据长度	输入：5736 字节 输出：5736 字节 但数据不能超过 4 帧 (过程数据帧最大数目)
同步	DC (分布时钟)		
物理层	100Base-TX		
调制	基带	每个从站的最大数据长度	输入：1434 字节 输出：1434 字节
波特率	100Mbit/s (100Base-TX)		
全双工模式	自动	最大信息长度	2048 字节
拓扑结构	菊花链式，分支连接	通信周期	500 μ s, 1000 μ s, 2000 μ s, 或 4000 μ s
传输介质	5 类双绞线电缆或者更高	最小通信周期	500 μ s
传输距离	节点间的距离：100m 或者更少 (IEEE802.3)	最大通信周期	4000 μ s
		从站间的同步跳动	最大 1 μ s

内置 EtherCAT 端口支持的功能见表 6-3。

表 6-3 内置 EtherCAT 端口支持的功能

功能名称	说 明	功能名称	说 明
过程数据通信	使用 CoE 的 PDO 映射	RAS 功能	启动网络时进行从站配置检查
	从站通信错误时故障弱化运行		读取错误信息 (紧急错误历史, 预订信息 etc)
	从站通信错误时停止运行		数据包监视
DC (分布时钟)	时钟同步		故障诊断信息
SDO 通信	CoE	出现错误时的运行	当错误出现 (停止通信或者改为故障弱化运行)
	● 紧急信息服务 (从从站接收)		清除错误后恢复通信 (转到运行状态)
	● SDO 请求和响应		
配置	● 从站间的通信		
	自动设置设备的变量名, 等等	从站信息	从站数目: 192
	使用从站的硬件开关设置节点地址		使能/禁用从站
	使用 Sysmac Studio 设置节点地址		断开/重新连接从站
	网络扫描 (收集连接到网络上的设备信息)		
	显示网络配置信息 (显示支持的拓扑结构)		

6.1.4 EtherCAT 主站、从站的通信端口

1. EtherCAT 主站通信端口

EtherCAT 主站有一个输出通信端口, 位于 NJ 系列 CPU 单元的右下方, 如图 6-3 所示。EtherCAT 通信端口有 3 个指示灯: NET RUN、NET ERR 和 LINK/ACT, 观察指示灯可以判断 EtherCAT 通信的状态, 指示灯对应的 EtherCAT 通信状态见表 6-4。

2. EtherCAT 从站的通信端口

EtherCAT 从站有一个输入端口和至少一个输出端口。OMRON GX 系列的远程 I/O 端子可用作从站, 如图 6-4 所示, 从站的节点地址用硬件开关设置, 设置的范围是 00、01 ~ 99, 十位数用左边旋转开关设置, 个位数用右边旋转开关设置。当设置为 00 时, 表示地址超过硬件开关能够设置的范围, 即大于 99, 此时地址用 Sysmac Studio 软件设置。如果从站单元没有硬件开关, 地址也要用 Sysmac Studio 软件设置。

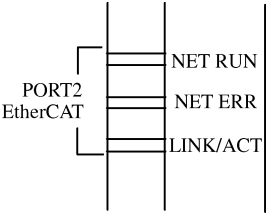


图 6-3 内置 EtherCAT 端口的指示灯

表 6-4 指示灯对应的 EtherCAT 通信状态

标签	名称	颜色	状态	含 义
EtherCAT NET RUN	运行	绿色	亮	EtherCAT 通信进行中 ● I/O 数据正在输入和输出

(续)

标签	名称	颜色	状态	含 义
EtherCAT NET RUN	运行	绿色	闪烁	EtherCAT 通信已建立 通信处在下列状态之一： - 只有信息通信在运行：预操作状态 - 只有信息通信和 I/O 数据的输入操作在运行：安全操作状态
			不亮	EtherCAT 通信停止 - 电源 OFF 或者单元被复位 - 出现 MAC 地址错误、通信控制器错误或者其他错误
EtherCAT NET ERR	错误	红色	亮	有一个不可恢复的错误，如硬件错误或异常
			闪烁	有一个可恢复的错误
			不亮	没有错误
EtherCAT LINK/ACT	链接/活动	黄色	亮	链接已建立
			闪烁 (周期 1s)	链接已建立并且数据正在被发送和接收 每当数据发送和接收时指示灯闪烁
			不亮	链接没有建立

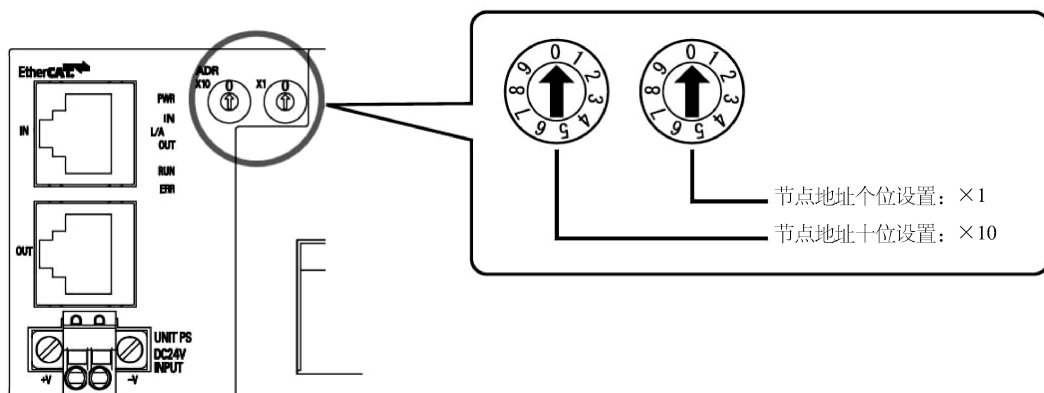


图 6-4 EtherCAT 从站地址设置

6.1.5 EtherCAT 网络拓扑结构

NJ 系列的 EtherCAT 支持两种拓扑结构。

1. 菊花链总线连接

图 6-5 所示为菊花链总线连接，这种结构没有分支。

2. 分支连接

图 6-6 所示为分支连接，这种结构使用连接从站，实现分支。

EtherCAT 不支持环形拓扑。

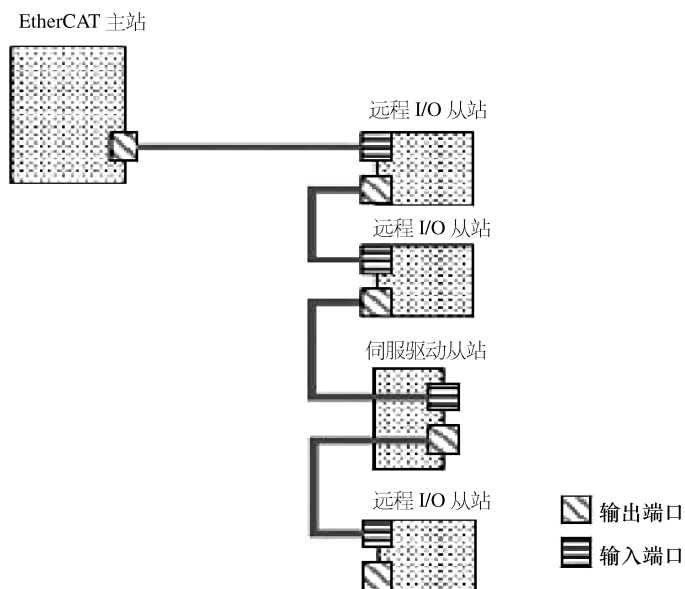


图 6-5 菊花链总线连接

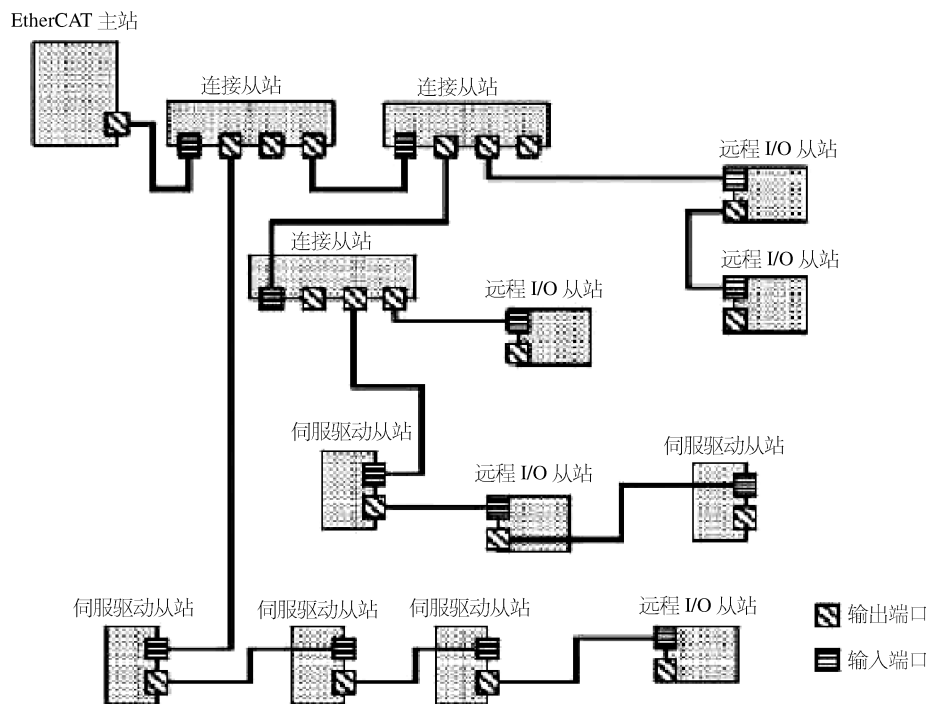


图 6-6 分支连接

6.1.6 EtherCAT 网络的连接

EtherCAT 通信电缆为 Ethernet 5 类 (100Base-TX) 或更高标准的屏蔽双绞线，电缆两端通过 RJ45 插头与 EtherCAT 通信端口连接。图 6-7 所示为 EtherCAT 端子接线。

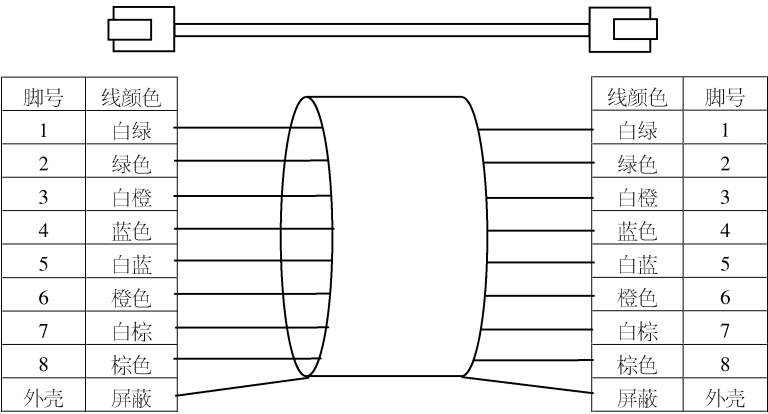


图 6-7 EtherCAT 端子接线

EtherCAT 通信口引脚的功能分配表见表 6-5。

表 6-5 EtherCAT 通信口引脚的功能分配表

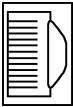
外形	脚号	信号名称	缩写	信号方向
	1	发送数据 +	TD +	输出
	2	发送数据 -	TD -	输出
	3	接收数据 +	RD +	输入
	4	不使用	...	...
	5	不使用	...	...
	6	接收数据 -	RD -	输入
	7	不使用	...	...
	8	不使用	...	...
	外壳	骨架接地	FG	...

图 6-8 所示为 EtherCAT 网络接线图。EtherCAT 主站接到第一个从站的输入端口，第一个从站的输出端口接到下一个从站的输入端口，依次类推。最后一个从站的输出端口不要做任何连接。布线时注意，任何两个节点 (L1, L2, ..., Ln) 之间电缆长度不能超过 100m。电缆总长度取决于网络的拓扑结构，有了电缆总的长度，用 Sysmac Studio 可以确定网络是否可以通信。

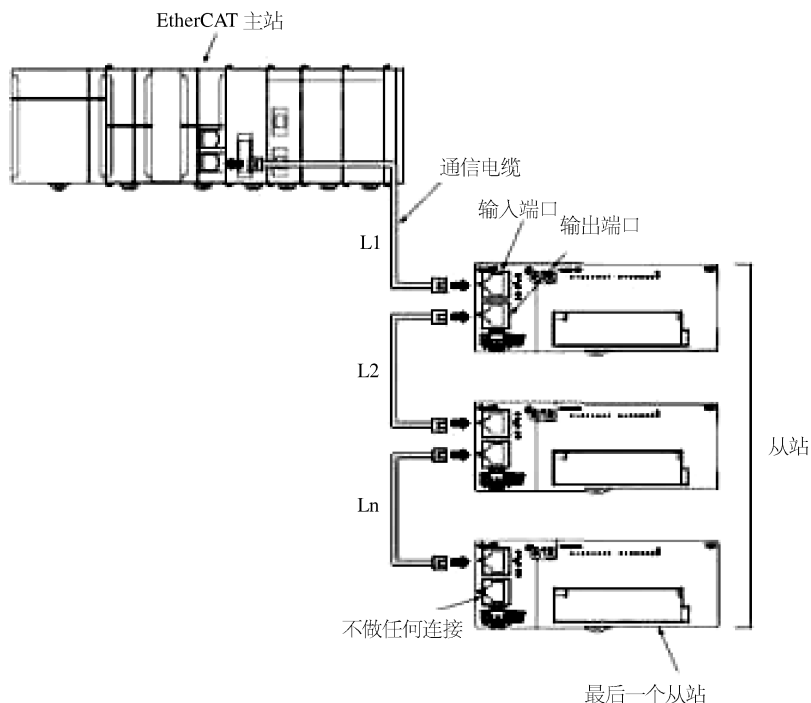


图 6-8 EtherCAT 网络接线图

6.2 EtherCAT 通信机制

6.2.1 EtherCAT 通信类型和设置

1. CoE (CAN Application Protocol over EtherCAT)

NJ 系列 CPU 内置 EtherCAT 端口，使用 CoE 协议通过 EtherCAT 网络与从站交换信息。CoE 协议在 EtherCAT 网络中实现 CANopen 通信，CANopen 是一种由 CiA 开发的工业通信协议。在 CoE 协议里，从站的参数和控制信息被定义成一种数据规范，这种规范叫做对象字典 (object dictionary)。

2. 过程数据通信和 SDO 通信

如图 6-9 所示，在主站和从站之间有两种类型的通信，用以交换数据。

- 1) 过程数据通信：使用过程数据对象 (PDOs) 的通信，固定周期内实时交换数据。
- 2) SDO 通信：需要时使用服务数据对象 (SDOs) 的通信。

NJ 系列内置的 EtherCAT 端口使用过程数据通信，在固定控制周期内刷新 I/O 数据，包括 EtherCAT 从站的 I/O 数据和伺服电动机的位置控制数据。使用 SDO 通信发送命令读写数据，如在指定时间传送参数。

每一个控制周期都要执行过程数据通信，不停地刷新数据，SDO 通信在 PDO 通信之间的间隙内执行。

3. 对象字典

对象字典的作用如图 6-10 所示。对象字典是一种设备数据表，是应用和通信的接口，

把两者联系起来。对象字典描述了设备处理的信息，每一条信息称为对象，每个对象分配 4 位十六进制数为索引。每个对象划分为一个数据类型区、一个 CoE 通信区和一个制造商专用区。

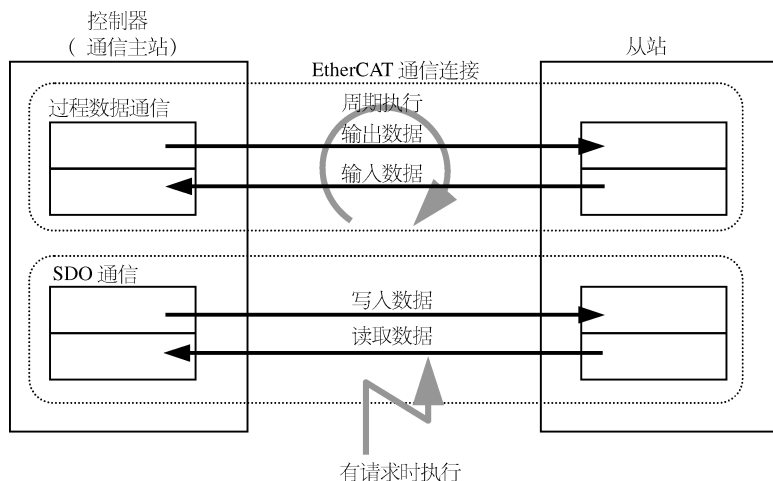


图 6-9 EtherCAT 主站与从站的通信方式

过程数据通信用来读和写这些对象里 CoE 通信区的过程数据。

SDO 通信用来读和写这些对象里制造商专用区的参数。

4. 分布时钟 (DC)

分布时钟如图 6-11 所示，这是 EtherCAT 独有的特性，能够实现精确的同步定时。基于 DC 时钟同步，EtherCAT 主站和从站实现时间（也就是 EtherCAT 系统时间）共享。

所有支持分布时钟的从站（同步从站）与参考时钟同步。在这个 EtherCAT 系统里，网络上所有从站中与 EtherCAT 主站最近的从站提供参考时钟。EtherCAT 主站和从站自动访问这个参考时钟。

6.2.2 EtherCAT 通信的实现

1. 实现方法

图 6-12 所示为 EtherCAT 两种通信实现的方法。

(1) 过程数据通信 (PDO 通信)

PDO 通信是在主站和从站之间固定的数据交换，称为过程数据通信。预先分配的 PDO 数据，也就是映射到 PDOs 的 I/O 数据，在每一个 EtherCAT 过程数据通信周期（也就是主周期任务的周期）内定时输入和输出。

NJ 系列 CPU 单元访问从站有两种方式：

1) 通过对应 EtherCAT 从站 I/O 的设备变量。

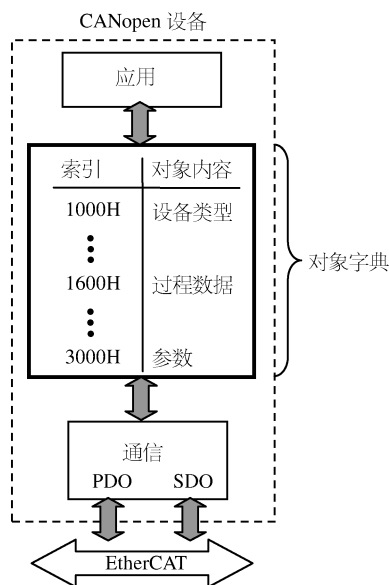


图 6-10 对象字典的作用

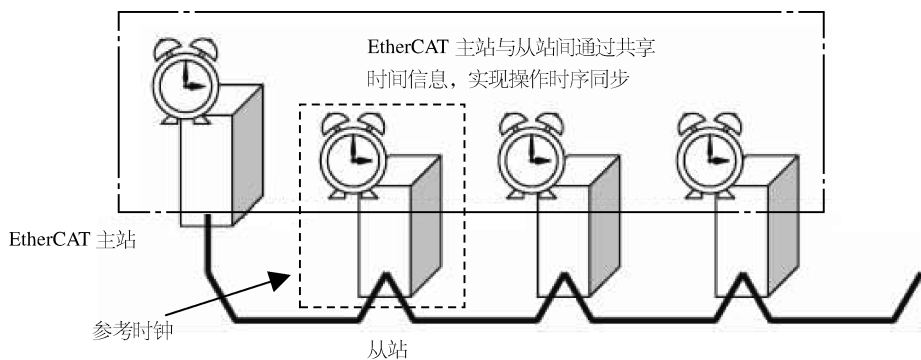


图 6-11 分布时钟

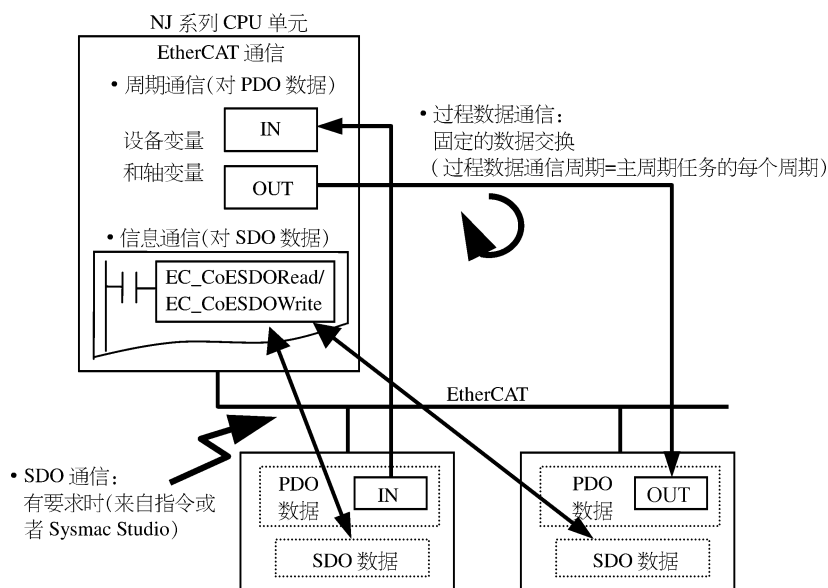


图 6-12 EtherCAT 两种通信实现的方法

2) 通过伺服驱动和编码器输入从站（已分配轴）的轴变量。

(2) SDO 通信

SDO 通信是在需要时，主站读取指定从站的数据，或向其写入数据。

使用 EC_CoESDORead（读 CoE SDO）指令或 EC_CoESDOWrite（写 CoE SDO）指令读/写指定的从站数据，如参数、错误信息等。

2. EtherCAT 变量的类型

EtherCAT 变量有三种类型。

(1) 设备变量

这是 EtherCAT 从站通用的输入-输出变量，可以改变它们的名称。

(2) 轴变量

这是表示伺服驱动和编码器输入从站中 I/O、参数的结构体变量，可以改变它们的名称。

(3) EtherCAT 主站的系统定义变量

这是由系统为通信参数、通信状态和其他功能定义的变量，不能改变它们的名称。

3. EtherCAT 通信设置

EtherCAT 通信时，必须用 Sysmac Studio 设置以下信息并下载到 CPU 单元。

- 1) 网络配置：主站和从站配置。
- 2) EtherCAT 主站设置：参数设置，例如过程数据通信周期。
- 3) 过程数据信息：为从站 PDO 数据分配的信息。

4. EtherCAT 通信编程

NJ 系列 CPU 单元的用户程序可以读写 EtherCAT 从站的数据，完成对伺服驱动和编码器输入从站的运动控制。

EtherCAT 通信编程的要点见表 6-6。

表 6-6 EtherCAT 通信编程的要点

从站类型	数据类型	时序	指令	变量
不能分配轴的从单元	过程数据	定时读/写	读/写指令，如 LD、OUT 和 MOV	设备变量
	SDO 数据	需要时读/写	EtherCAT 指令 (EC_CoESDORed 或 EC_CoESDOWrite)	用户定义变量
分配轴的从单元	过程数据	定时读/写	运动控制指令，或读/写指令如 LD、OUT 和 MOV	轴变量或设备变量
	SDO 数据	需要时读/写	EtherCAT 指令 (EC_CoESDORed 或 EC_CoESDOWrite)	用户定义变量

6.2.3 EtherCAT 通信的状态转换

1. 启动时自诊断

当电源 ON 时，EtherCAT 主站执行下列自诊断功能，错误分别用系统定义变量表示。

(1) 网络配置信息诊断

网络配置信息的错误用变量_EC_NetCfgErr 表示。

(2) 通信端口诊断

MAC 地址错误用变量_EC_MacAdrErr 表示，通信控制器错误用变量_EC_LanHwErr 表示。

2. EtherCAT 通信的控制状态

EtherCAT 通信有 4 个控制状态，通信会在这 4 个状态之间转换，如图 6-13 所示。

当电源上电后，主站和从站会依次从初始状态到预操作状态，再到安全操作状态，最后到操作状态，然后开始 EtherCAT 通信。另外，根据出现的错误和其他条件，状态会自动改变，此时，EtherCAT 通信照常进行。

观察前面板的 RUN 指示灯可以确定当前控制状态。例如，RUN 不亮时，处于初始状态；常亮时，处于操作状态；闪烁时，处在预操作状态或安全操作状态。

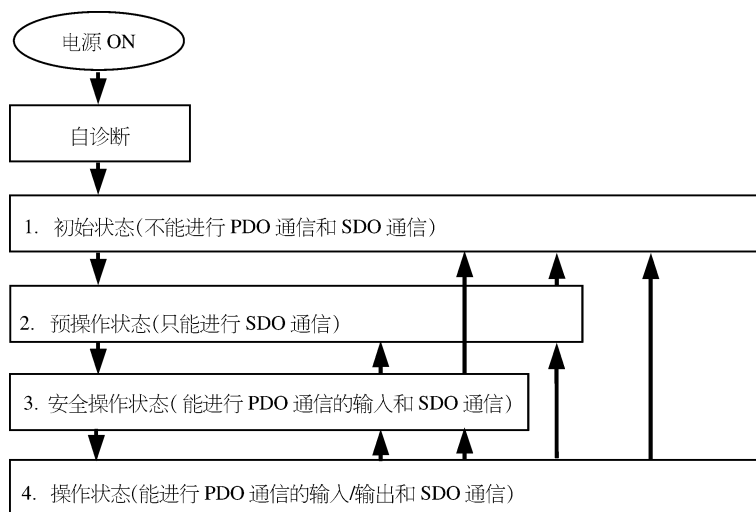


图 6-13 EtherCAT 通信的状态转换

6.3 EtherCAT 配置与设置

执行 EtherCAT 通信时，必须用 Sysmac Studio 创建以下三种类型的数据并下载到 CPU 单元。

- 1) EtherCAT 网络配置。
- 2) 过程数据信息。这是为从站 PDO 数据分配的信息，也称为 PDO 映射数据。
- 3) EtherCAT 主站和从站参数设置。例如，过程数据通信周期和等待所有从站启动时间。

当下载网络配置信息到 CPU 单元时，EtherCAT 主站初始化从站并完成基于网络配置信息的过程数据通信。

6.3.1 EtherCAT 网络操作步骤

图 6-14 所示为 EtherCAT 网络配置，共有 3 个从站，节点#1 为数字 I/O 从站，节点#2、节点#3 均为内置 EtherCAT 通信的 OMRON G5 系列伺服驱动器。

EtherCAT 网络操作步骤如下。

1. 接线和设置

(1) 接线

- 1) 安装 NJ 系列 CPU 单元和从站。
- 2) 用通信电缆连接 EtherCAT 主站和从站。
- 3) 连接电源。

(2) 设置

对每一个从站单元设置节点地址。

2. 设置 EtherCAT 网络配置

启动 Sysmac Studio 并做下列设置：

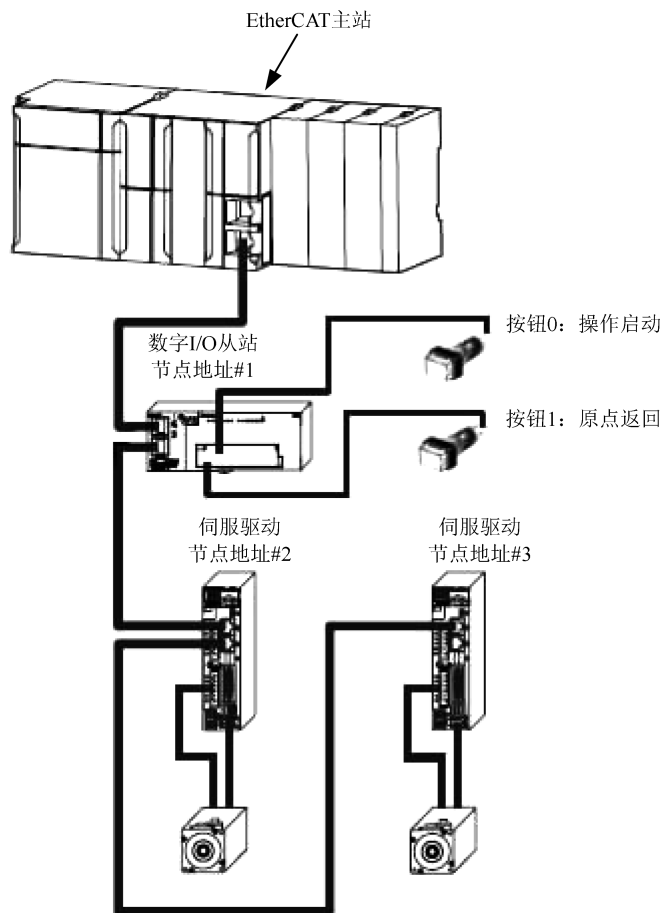


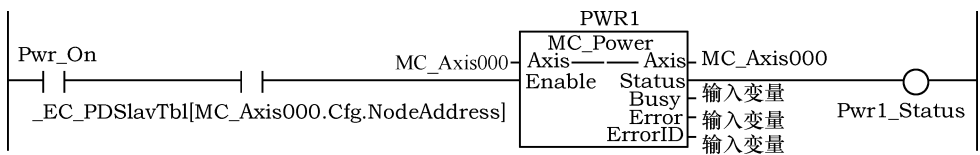
图 6-14 EtherCAT 网络配置

- 1) 创建 EtherCAT 网络配置。
- 2) 为数字 I/O 从站 (节点地址#1) 分配设备变量，BOOL 型变量 Pwr_On、Hm1_On 分别对应按钮 0、按钮 1。
- 3) 伺服驱动 (节点地址#2 和#3) 的轴设置。
- 4) 设置 EtherCAT 主站参数。
- 5) 设置从站参数。

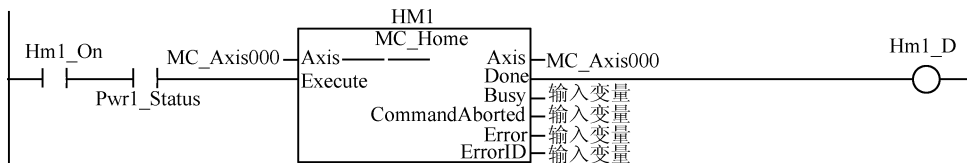
3. 编程

(1) 编程

当操作启动按钮被按下，Pwr_On 变量变为 TRUE，轴控制被使能。



当原点返回按钮被按下，Hm1_On 变量变为 TRUE，原点返回被执行。



(2) 程序分配

程序分配用于给任务分配程序和设置程序执行顺序。

4. 离线调试

可以在离线调试状态下用模拟器来检查程序和任务执行时间。

5. 电源上电

接通以下电源。

- (1) 从站单元电源 (电源接通时，从站上的 PWR 指示灯会亮)
- (2) 从站 I/O 电源
- (3) NJ 系列控制器电源

6. 在线连接

将 Sysmac Studio 上设置的网络配置和实际的网络配置进行比较并合并。

7. 下载网络配置信息和用户程序

将 Sysmac Studio 上创建的网络配置信息和用户程序下载至 NJ 系列控制器。

8. 确认通信开始

检查以确认所有登记的从站都加入了网络并且通信开始。

1) 确认主站指示灯状态：NET RUN 指示灯亮、NET ERROR 指示灯不亮、LINK/ACT (物理层链接) 指示灯闪烁。

2) 确认从站指示灯状态：PWR 指示灯和 RUN 指示灯亮、ERR 指示灯不亮、L/A IN (物理层链接输入) 闪烁、L/A OUT (物理层链接输出) 闪烁 (最后一个从站上不亮)。

6.3.2 创建 EtherCAT 网络配置

创建 EtherCAT 网络配置的步骤如下。

- 1) 离线启动 Sysmac Studio。
- 2) 如图 6-15 所示，双击多视图浏览器中“配置和设置”下的“EtherCAT”；或右击“EtherCAT”，从菜单中选择“编辑”。

EtherCAT 主站显示在编辑面板中，如图 6-16 所示。

3) 在工具箱选择一个从站，将其拖放到编辑面板中主站的下方，从站便添加到网络中。从工具箱选择另一个从站，将其拖放到已添加的从站之下，另一个从站便添加到网络中。最后的 EtherCAT 配置如图 6-17 所示。



图 6-15 EtherCAT 网络



图 6-16 EtherCAT 主站

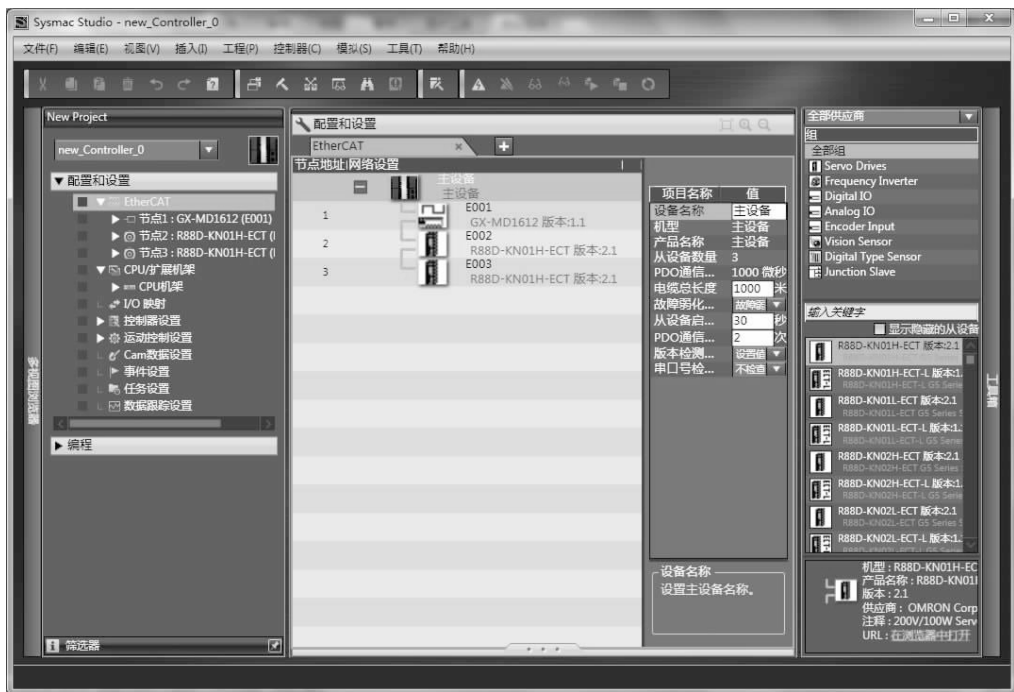


图 6-17 EtherCAT 配置

6.3.3 设置 EtherCAT 从站变量和轴

设备变量用来读写 EtherCAT 从站的过程数据。轴变量用来从运动控制功能模块操作分配了轴的从站。

1. 为所有的 EtherCAT 从站登记设备变量

在 Sysmac Studio 使用 I/O 映射为 I/O 端口分配设备变量，创建的设备变量登记在全局变量表中。创建的方法如下。

(1) 从已登记在变量表中的变量选择

在 I/O 映射的菜单中选择已登记的变量，步骤如下。

- 1) 在全局变量表或一个 POU 的本地变量表中预先登记变量。
- 2) 用这些变量编程。
- 3) 创建从站配置信息。
- 4) 在 I/O 映射的下拉列表中选择变量，并将它们分配给 I/O 端口，如图 6-18 所示。

(2) 输入新的设备变量名称

输入需要的设备变量名称。例如，从站的 I/O 分配了设备变量，给变量一个合适的名称。

1) 双击多视图浏览器中“配置和设置”下的“I/O 映射”；或右击“I/O 映射”，从菜单中选择“编辑”。

2) 在 I/O 映射窗口选择 I/O 端口，在变量栏输入新的变量名称，如图 6-19 所示。



图 6-18 选择设备变量



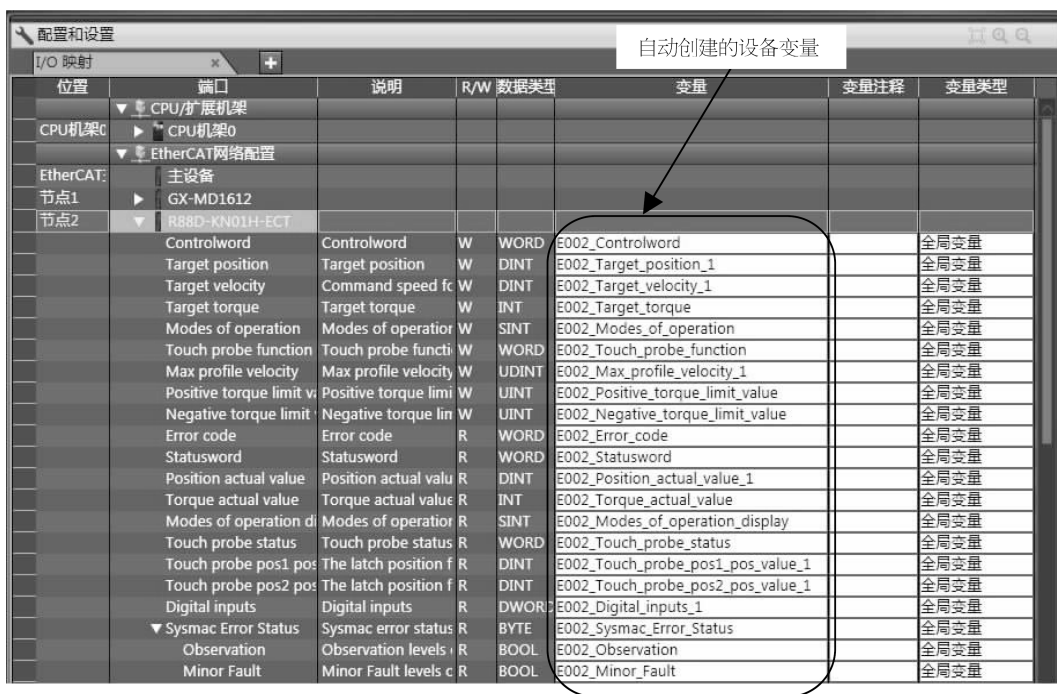
图 6-19 输入新的变量名称

(3) 自动创建设备变量名称

结合设备名称和 I/O 端口名称创建设备变量名称。设备名称在从站参数中设置。默认设备名称是“E”后面跟着一个从“001”开始的顺序号。

1) 双击多视图浏览器中“配置和设置”下的“I/O 映射”；或右击“I/O 映射”，从菜单中选择“编辑”，I/O 映射窗口显示出来。

2) 在 I/O 映射窗口选择一个从站或 I/O 端口，右击，从菜单中选择“生成设备变量”，设备变量自动分配给从站的 I/O 端口，并且登记在变量表中，如图 6-20 所示。



位置	端口	说明	R/W	数据类型	变量	变量注释	变量类型
▼ CPU/扩展机架							
CPU机架0	▼ CPU机架0						
▼ EtherCAT网络配置							
EtherCAT主设备							
节点1	▼ GX-MD1612						
节点2	▼ R88D-KN01H-ECT						
	Controlword	Controlword	W	WORD	E002_Controlword		全局变量
	Target position	Target position	W	DINT	E002_Target_position_1		全局变量
	Target velocity	Command speed f	W	DINT	E002_Target_velocity_1		全局变量
	Target torque	Target torque	W	INT	E002_Target_torque		全局变量
	Modes of operation	Modes of operation	W	SINT	E002_Modes_of_operation		全局变量
	Touch probe function	Touch probe functi	W	WORD	E002_Touch_probe_function		全局变量
	Max profile velocity	Max profile velocity	W	UDINT	E002_Max_profile_velocity_1		全局变量
	Positive torque limit v	Positive torque limi	W	UINT	E002_Positive_torque_limit_value		全局变量
	Negative torque limit	Negative torque lin	W	UINT	E002_Negative_torque_limit_value		全局变量
	Error code	Error code	R	WORD	E002_Error_code		全局变量
	Statusword	Statusword	R	WORD	E002_Statusword		全局变量
	Position actual value	Position actual valu	R	DINT	E002_Position_actual_value_1		全局变量
	Torque actual value	Torque actual value	R	INT	E002_Torque_actual_value		全局变量
	Modes of operation d	Modes of operator	R	SINT	E002_Modes_of_operation_display		全局变量
	Touch probe status	Touch probe status	R	WORD	E002_Touch_probe_status		全局变量
	Touch probe pos1 pos	The latch position f	R	DINT	E002_Touch_probe_pos1_pos_value_1		全局变量
	Touch probe pos2 pos	The latch position f	R	DINT	E002_Touch_probe_pos2_pos_value_1		全局变量
	Digital inputs	Digital inputs	R	DWORD	E002_Digital_inputs_1		全局变量
	▼ Sysmac Error Status	Sysmac error status	R	BYTE	E002_Sysmac_Error_Status		全局变量
	Observation	Observation levels	R	BOOL	E002_Observation		全局变量
	Minor Fault	Minor Fault levels	R	BOOL	E002_Minor_Fault		全局变量

图 6-20 自动创建设备变量名称

2. 伺服驱动和编码器输入从站的轴设置

(1) 添加轴

1) 右击多视图浏览器中的“配置和设置-运动控制设置”下的“轴设置”，从菜单中选择“添加-轴设置”，如图 6-21 所示。

轴变量 MC_Axis000 添加在“轴设置”里，如图 6-22 所示。

2) 如图 6-23 所示，双击多视图浏览器中的“编程-数据”下的“全局变量”。

可以确认 MC_Axis000 轴变量已经自动被添加，如图 6-24 所示。

轴变量可以改名称、删除或复制。

(2) 将轴登记为从站

1) 双击已登记的轴变量 MC_Axis000，轴参数设置界面会在编辑面板中显示。轴基本设置窗口将会首先显示，如图 6-25 所示。



图 6-21 添加轴



图 6-22 轴变量被添加



图 6-23 全局变量



图 6-24 显示 MC_Axis000 轴变量

2) 为了使用伺服轴，设置“轴使用”为“使用的轴”，“轴类型”为“伺服器轴”。

3) 为输入设备选择从站，该从站分配已登记的伺服轴，如图 6-26 所示。

(3) 设置轴参数

如图 6-25 所示，点击轴参数设置窗口的每一个图标，每个图标对应的轴参数见表 6-7。可以将所有的轴参数显示在一个轴设置表格，在这个表格编辑所有的轴参数。



图 6-25 轴基本设置窗口



图 6-26 为输入设备选择从站

表 6-7 轴 参 数

图标	名称	说 明
	轴基本设置	设置轴的使能/禁用、轴类型、节点地址和通道
	单位换算设置	设置每个电动机旋转时使用脉冲的电子齿轮的齿轮比、移动的距离

(续)

图标	名称	说 明
	操作设置	设置速度、加速度、减速度、扭矩报警值和其他监视参数
	其他操作设置	设置伺服驱动器的 I/O
	限位设置	设置软件限位及位置偏移的限位
	原点返回设置	设置原点返回操作
	位置计数设置	设置控制器的计数模式
	伺服驱动设置	设置伺服驱动器的参数

6.3.4 EtherCAT 主站和从站参数设置

1. EtherCAT 主站设置

- 1) 双击多视图浏览器中的“配置和设置”下的“EtherCAT”；或右击“EtherCAT”，从菜单中选择“编辑”。
- 2) 在编辑面板中选择显示的 EtherCAT 主站，如图 6-27 所示。



图 6-27 选择 EtherCAT 主站

EtherCAT 主站设置见表 6-8。

表 6-8 EtherCAT 主站设置

名称	编辑	说 明
设备名称	允许	显示 EtherCAT 主站名称 默认设置：主站
机型	不允许	总是为“主站”
产品名称	不允许	总是为“主站”
从站数量	不允许	从站的数目根据拓扑结构自动计算，能够达到 192 个从站 显示范围：0 ~ 192
PDO 通信周期	不允许	根据主周期任务的任务周期自动计算 设置范围：500 ~ 4000 μ s
电缆总长度	允许	这是从站间电缆的总长度 设置范围：1 ~ 4294967295 默认设置：1000m
故障弱化操作设置	允许	为所有网络中的从站选择“停止”或“故障弱化操作”，当主站检测到通信错误（电缆断开、从站错误等）时 默认设置：故障弱化操作
从站启动等待时间	允许	设置从检测 EtherCAT 端口链接开始直到所有从站加入网络的等待时间 设置范围：3 ~ 200s 默认设置：30s 如果所有从站都在等待时间内出现，则过程数据通信启动。如果有的从站在等待时间过去后没有出现，则发生一个网络配置检验错误。当使用的从站需要一段时间来启动，设置较长的等待时间来防止错误发生
PDO 通信超时检测次数	允许	如果过程数据（PDO）通信超时不断出现，超过指定的次数，则发生一个过程数据接收超时错误。如果过程数据通信超时频繁发生，例如在高噪声的环境下，则要增加设置。把从站需要替换的值设置为 2 或者更高 设置范围：1 ~ 8 次错误 默认设置：2 次错误
版本检测方法	允许	指定通信开始时版本的检测方法，将储存在网络配置信息（设置）中的版本号与从站（实际设备）的实际版本号进行比较验证。验证是必须的，如果没有对从站验证，通信将不会启动，并且出现一个网络配置检测错误 设置值：设置值 $\leq$ 实际设备、设置值 = 实际设备或者不检测 默认设置：设置值 $\leq$ 实际设备
序列号检测方法	允许	指定在初始化通信时是否检验从站序列号，将储存在网络配置中从站的序列号与实际从站中设置的序列号进行比较验证。如果没有对从站验证，通信将不会启动，并且出现一个网络配置检测错误 设置值：设置值 = 实际设备或不检查 默认值：不检查

1) 版本号检测见表 6-9。

表 6-9 版本号检测

设置值	版本号检测结果	与从站通信
设置值 ≤ 实际设备	设置值 > 实际设备	不可以
	设置值 < 实际设备	可以
	设置值 = 实际设备	可以
设置值 = 实际设备	设置值 > 实际设备	不可以
	设置值 < 实际设备	不可以
	设置值 = 实际设备	可以
不检查	设置值 > 实际设备	可以
	设置值 < 实际设备	可以
	设置值 = 实际设备	可以

2) 当设置值是“设置值 = 实际设备”时，如果一个从站被替换，那么网络中的从站通信不会启动。如果设置值是“设置值 = 实际设备”时需要替换从站，必须纠正网络配置信息并且再次传送到 EtherCAT 主站。如果要求严格管理设备配置时，这个参数设置为“设置值 = 实际设备”。

3) 如果故障弱化操作设置参数被设置为“停止”，当在一个从站中检测到一个 EtherCAT 通信错误时，过程数据通信将会停止。由于这个原因，如果伺服驱动已经连接，所有轴的伺服驱动将会变为“OFF”状态。故障弱化操作参数设置可以保证当设备故障发生时处于安全操作状态。

4) 如果验证的版本是无效的，参数也会将不同的 EtherCAT 版本传送到从站。如果一个版本不兼容的从站被连接，不正确的参数可能被设置，操作也会不正确。如果不使用版本检测，确保在传输参数之前只有兼容的从站被连接。

5) 网络配置检验。当 EtherCAT 主站启动时，网络配置信息与实际网络的信息将比较验证。在验证过程中如果发现不一致，将在为网络配置验证错误提供的系统定义变量中反映出来。此设置适用于整个系统而不是单个从站。

校验时，连接的从站数量、厂商 ID、产品编码、版本和序列号，前三项是必选的，后两项是可选的。序列号验证针对 EtherCAT 拓扑结构中所有的从站，不能只验证单个从站。默认设置是“不检查”。

2. 设置 EtherCAT 从站

1) 双击多视图浏览器中的“配置和设置”下的“EtherCAT”；或右击“EtherCAT”，从菜单中选择“编辑”。

2) 选择在编辑面板中显示的 EtherCAT 从站，如图 6-28 所示。

EtherCAT 从站设置见表 6-10。

● 即使禁用的从站在 EtherCAT 网络中存在，也不会出现网络错误。可以使用它们为未来的系统扩展做准备。

当实际添加从站到网络中时，主站和新的从站之间的通信将不会被初始化，直到 Sysmac Studio 中禁用从站变为使能并且网络配置信息再次下载到 NJ 系列 CPU 单元的内置 EtherCAT

主站中。一个禁用的从站即使它实际上存在于网络中，也被当做不存在。确保实际添加从站时下载网络配置信息。



图 6-28 选择 EtherCAT 从站

表 6-10 EtherCAT 从站设置

名称	编辑	说 明
设备名称	允许	显示从站的名称 默认设置: E* * * (* * * 是从 001 开始的顺序号)。默认值根据节点地址自动产生 设置范围: 1 ~ 192
机型	不允许	自动显示从站机型
产品名称	不允许	显示从站产品名称
版本	不允许	显示从站的版本号
节点地址	允许	设置节点地址 当添加从站时, 自动设置默认值 设置范围: 1 ~ 192
使能/禁用设置	允许	使能/禁用作为通信目标的从站 ●使能: 从站运行 ●禁用: 从站不运行 将还没有在 EtherCAT 网络中安装但计划稍后添加的从站登记为禁用的从站。即使对于禁用的从站, 过程数据分配也能设置, 并且在用户程序里作为系统定义变量和设备变量使用 默认设置: 使能

(续)

名称	编辑	说 明
序列号	不允许	显示从站序列号 默认: 0x00000000 (离线时) 从菜单中为主站选择“获取从站序列号”时, 会更新为实际从站的序列号 当序列号检查方法在主站设置为“设置值 = 实际设备”时, 如果 Sysmac Studio 的序列号与实际从站的序列号不相符, 会出现网络配置检验错误
PDO 映射设置	允许	从站的默认过程数据分配由 Sysmac Studio 提供 创建网络配置时, 设备变量在初始过程数据分配中被自动创建。作为设备变量, 可以从用户程序中读取和写入这些过程数据分配 当有效的 PDOs 存在时, 它们的列表将会被显示。当有效的 PDOs 不存在时, 显示 “---” 如果要求编辑过程数据分配, 单击列表下的“编辑 PDO 映射设置”按钮
分布时钟有效	允许	如果从站提供分布时钟, 显示“有效”。可以使分布时钟无效 如果从站提供分布时钟并且可以使它无效, 则显示“有效”或“无效” 如果从站没有提供分布时钟, 显示 “---”
参考时钟	不允许	显示是否从站提供参考时钟。如果在系统中有两个或多个从站的分布时钟有效时, 拥有参考时钟的从站必须放在第一个分支从站前。如果从站提供一个参考时钟显示“存在”, 如果从站不提供参考时钟显示“不存在”
参数设置	允许	如果从站有初始参数设置功能, 显示“设置” 如果从站没有初始化参数设置功能, 显示 “---” 点击“编辑参数设置”按钮, 编辑设置
备份参数设置	允许	如果从站有备份参数设置功能, 显示“设置” 如果从站没有备份参数设置功能, 显示 “---” 点击“编辑备份参数设置”按钮, 编辑设置

当从站在实际网络配置中出现在禁用的从站位置时, 操作见表 6-11。

表 6-11 禁用从站的操作

	在实际网络配置中的从站		对禁用从站的操作	
	节点地址	厂商 ID/产品编码	过程数据通信	故障
1	匹配	匹配	无	正常操作 (无错误)
2	匹配	不同	无	网络配置检验错误
3	不同	匹配	无	网络配置检验错误
4	不同	不同	无	网络配置检验错误

6.3.5 EtherCAT 网络配置比较和合并

使用 Sysmac Studio 来比较 Sysmac Studio 中的网络配置信息设置和连接的实际网络配置, 包括分支线和分支位置。

比较的项目如下。

- 节点地址

- 厂商 ID
- 产品编码
- 版本
- 连接端口

如果 Sysmac Studio 的设置和实际网络配置完全匹配, 则当错误出现时可以确定错误的位置。如果比较结果显示有差别, 从 Sysmac Studio 合并网络配置。

使用下面的过程来比较和合并 Sysmac Studio 的网络配置设置和实际的网络配置。

1) 启动 Sysmac Studio 并且在线连接控制器。

2) 双击多视图浏览器中“配置和设置”下的“EtherCAT”, 或者右击“EtherCAT”, 从菜单中选择“编辑”。

3) 右击显示在编辑面板的 EtherCAT 主站并且选择“与物理网络配置比较和合并”, 如图 6-29 所示。

显示“与物理网络配置比较和合并”的对话框。Sysmac Studio 的设置与实际网络配置的比较结果显示在比较结果栏中, 如图 6-30 所示。

4) 如果有从站仅存在于实际网络中。“添加”会显示在比较结果栏中。将实际网络配置的从站拖到 Sysmac Studio 的网络配置中, 如图 6-31 所示。



图 6-29 选择“与物理网络配置比较和合并”



图 6-30 比较结果

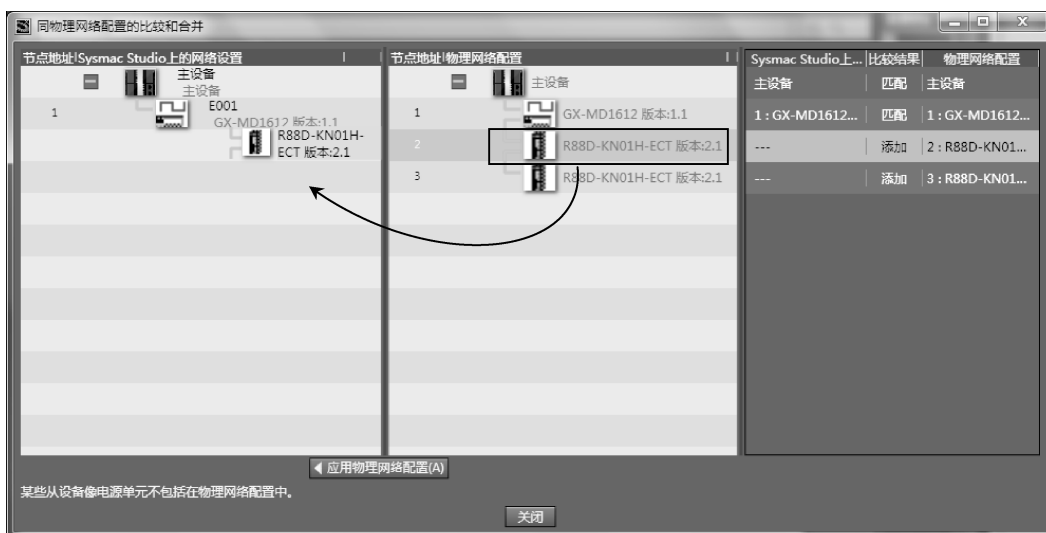


图 6-31 添加从站

在把所有缺失的从站拖过去并且把它们添加到 Sysmac Studio 中的网络配置时，“比较结果栏”将显示配置相同的所有项目，如图 6-32 所示。



图 6-32 完全匹配显示

5) 点击“关闭”按钮。

Sysmac Studio 返回到编辑面板。这就完成了与物理网络配置比较和合并的操作。

当一个从站只存在于 Sysmac Studio 的设置中时，比较结果栏会显示“删除”。图 6-33 所示的例子显示一个节点地址在 Sysmac Studio 中和实际网络配置不一样，这种情况下，比较结果栏显示“添加”和“删除”。

在这里，无论是 Sysmac Studio 网络配置中正确的节点地址还是实际网络配置中的节点地址，都要反复进行比较和合并操作，以确保配置是相同的。



图 6-33 比较结果栏显示“添加”和“删除”

注意：

1) 在执行比较和合并操作之前要确保主站和从站之间的电缆是正确连接的。如果连接不正确的话，“与物理网络配置比较和合并”的对话框是不会显示的。

2) 如果在实际网络配置中的从站节点没有被设定或者相同的地址被设置了多个从站，“与物理网络配置比较和合并”的对话框是不会显示的。在执行比较和合并之前要确保为实际网络配置的从站正确设置节点地址。

3) 当执行比较和合并时，Sysmac Studio 和控制器之间不能保持同步。在对从站执行任何在线操作之前，要执行 Sysmac Studio 和控制器的同步。

自动创建网络配置

为了取代离线手动设置网络配置，也能根据实际网络配置在 Sysmac Studio 中自动创建网络配置。使用以下步骤自动地在 Sysmac Studio 中复制实际网络配置。

1) 启动 Sysmac Studio 并且在线连接控制器。

2) 双击多视图浏览器中“配置和设置”下的“EtherCAT”，或者右击“EtherCAT”，从菜单中选择“编辑”。

3) 右击显示在编辑面板中的 EtherCAT 主站并且选择“与物理网络配置比较和合并”，如图 6-34 所示。

显示“与物理网络配置比较和合并”对话框，如图 6-35 所示。



图 6-34 EtherCAT 网络



图 6-35 与物理网络配置比较和合并

4) 点击“应用物理网络配置”按钮，如图 6-36 所示，实际的网络配置复制到 Sysmac Studio 中的网络配置中，比较结果栏显示配置的各个方面完全匹配。

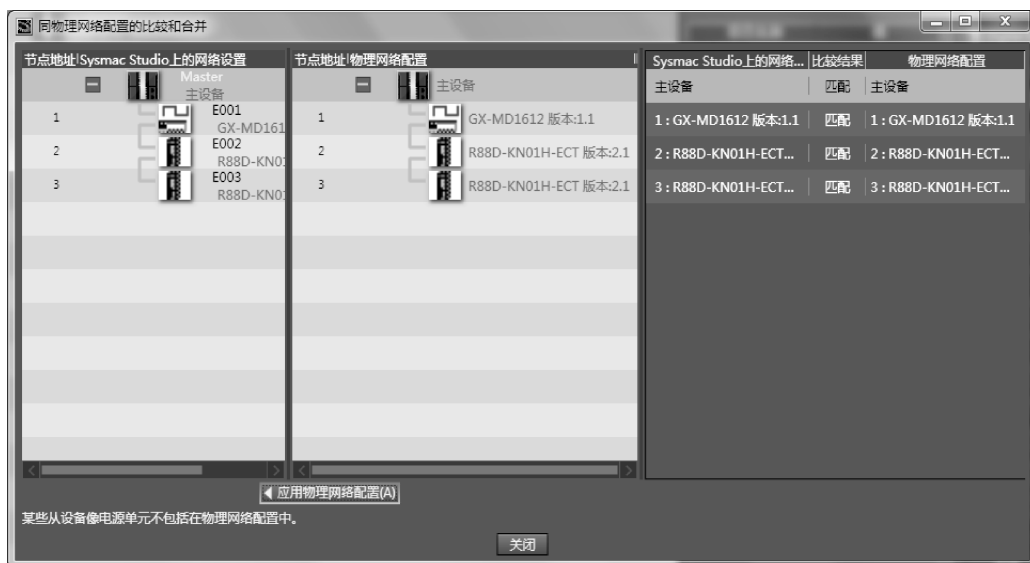


图 6-36 点击“应用物理网络配置”按钮

5) 单击“关闭”按钮。

6) 离线，并且分配设备变量，设置轴参数，建立任务，设置主站和从站参数。

在 Sysmac Studio 中自动创建网络配置时，Sysmac Studio 之前配置和设置的信息将会被删除。为了在 Sysmac Studio 中没有丢失当前设置的情况下合并物理网络配置信息，使用“与

物理网络配置比较和合并”操作来创建网络配置。

3. 使用 Sysmac Studio 从实际的网络配置获得序列号

如果在 EtherCAT 主站设置序列号检测方法为“设置值 = 实际设备”，必须把设置了从站序列号的网络配置信息下载到 CPU 单元。使用以下步骤获取实际从站的序列号并且把它们应用到 Sysmac Studio 中的从站序列号设置中。

- 1) 启动 Sysmac Studio 并且在线连接控制器。
- 2) 双击多视图浏览器中的“配置和设置”下的“EtherCAT”；或者右击“EtherCAT”，从菜单选择“编辑”。
- 3) 右击显示在编辑面板中的 EtherCAT 主站，从菜单中选择“取得从站序列号”，如图 6-37 所示。



图 6-37 取得从站序列号

实际从站的序列号作为 Sysmac Studio 中从站设置的序列号来保存。

注意：

- 1) 在执行操作之前要确保主站和从站之间的电缆是正确连接的。在连接不正确的情况下不能获取从站的序列号。
- 2) 如果在实际网络配置中的从站节点没有被设定或者相同的地址被设置了多个从站，则不能获取序列号。在执行操作之前要确保为实际网络配置的从站正确设置节点地址。
- 3) 除非“与物理网络配置比较和合并”对话框显示所有的从站配置相同，否则不能取得从站的序列号。在操作之前确保“与物理网络配置比较和合并”对话框显示所有的从站配置相同。但是如果在 Sysmac Studio 中有从站设置为禁用，即使所有的从站配置不相同也

可以获得序列号。

4) 当在实际网络配置中不存在禁用的从站时, 如果获得了从站的序列号, 那么禁用的从站将在 Sysmac Studio 设置中被清除为 0x00000000。如果禁用的从站存在于实际的网络配置中, 那么实际网络配置中的从站序列号将保存于 Sysmac Studio 的设置中。

5) 当从实际网络配置获取序列号时, 在 Sysmac Studio 设置中任何为从站设置的序列号都要被重写。

6) 当从实际网络配置获取序列号时, Sysmac Studio 和控制器之间不能保持同步。在对从站执行任何在线操作之前要同步 Sysmac Studio 和控制器。

6.3.6 从 Sysmac Studio 下载网络配置信息

必须从 Sysmac Studio 把工程中的网络配置信息下载到 NJ 系列 CPU 单元。使用同步操作下载网络配置信息。

1) 在线并且从“控制器”菜单中选择“同步”, 如图 6-38 所示。

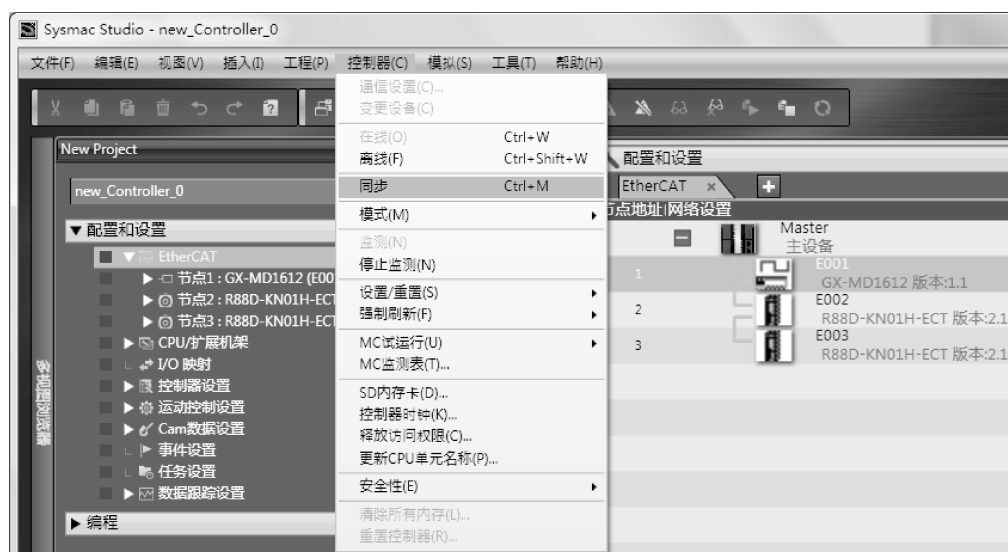


图 6-38 同步操作

显示同步对话框, 如图 6-39 所示。

2) 在同步对话框中选择 EtherCAT 复选框并且点击“传送到控制器”按钮。

注意:

1) 在传送用户程序后, CPU 单元被重启。EtherCAT 从站的通信被中断 45s。在此期间, 从站的输出取决于从站设置。在传送用户程序之前, 要确保控制系统的安全性。

2) 使用 Sysmac Studio 的同步菜单上载和下载工程。

3) EtherCAT 从站的备份参数超出了同步的范围, 并且默认设置为不下载到从站中。为了传送备份参数, 在同步面板的复选框中清除“不传送特殊单元参数和 EtherCAT 从站的备份参数 (超出同步范围)”的选择。

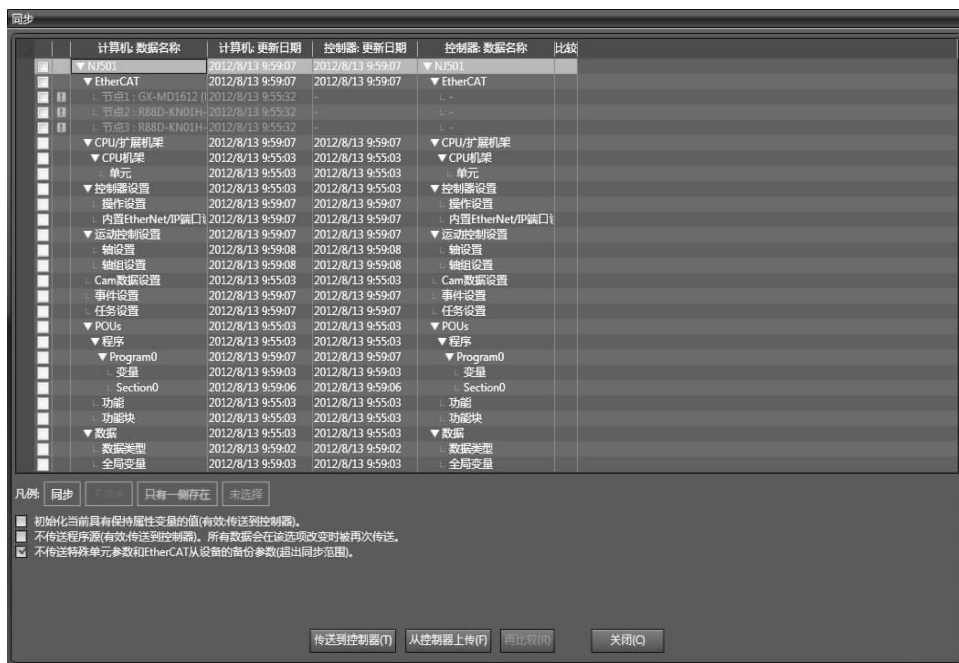


图 6-39 同步对话框

6.3.7 完成 EtherCAT 配置和设置后确认通信正常

如果正常的网络配置信息下载到 CPU 单元,无论 CPU 单元处于何种操作模式,EtherCAT 通信会自动启动。

如果 NJ 系列 CPU 单元面板上的 EtherCAT NET RUN 指示灯常亮,EtherCAT 的通信处于进行状态(正常通信状态)。

如果网络配置信息没有下载到 CPU 单元,EtherCAT 主站不会执行过程数据通信。正因为如此,错误通知将不会出现。EtherCAT 通信将处在初始状态,无论是过程数据通信还是 SDO 通信都将被禁用。

从用户程序可以检查系统定义的变量_EC_PDSlavTbl(过程数据通信从站表),确定 I/O 刷新是否正常。

EtherCAT 的通信并不总是电源通电后立即建立。在控制设备变量和 Axis 变量之前,在用户程序中使用系统定义的变量确认通信已经建立。

当更改任何 EtherCAT 从站或特殊单位设置,仔细检查控制系统的安全性,然后再重新启动单元。

改变 EtherCAT 从站配置和重新配置轴分配设置后,必须重新下载网络配置信息。

6.4 PDO 通信和 SDO 通信

6.4.1 PDO 通信

PDO 通信是指在过程通信周期(也就是主周期任务的任务周期),主站和从站之间周期

性地交换数据。

在 NJ 系列 CPU 单元的用户程序里, 通过预先分配的变量访问从站数据。

1. 过程数据通信分配的变量

分配的变量取决于从站的类型, 例如, EtherCAT 从站在“I/O 映射”中分配设备变量, 伺服驱动和编码器输入从站则在“轴设置”中分配轴变量 (结构体)。

(1) 所有的 EtherCAT 从站

如图 6-40 所示, 通过读写设备变量, NJ 系列 CPU 单元的用户程序可以读写 EtherCAT 从站过程数据。从站的过程数据是每个周期内主站和从站交换的 I/O 数据。

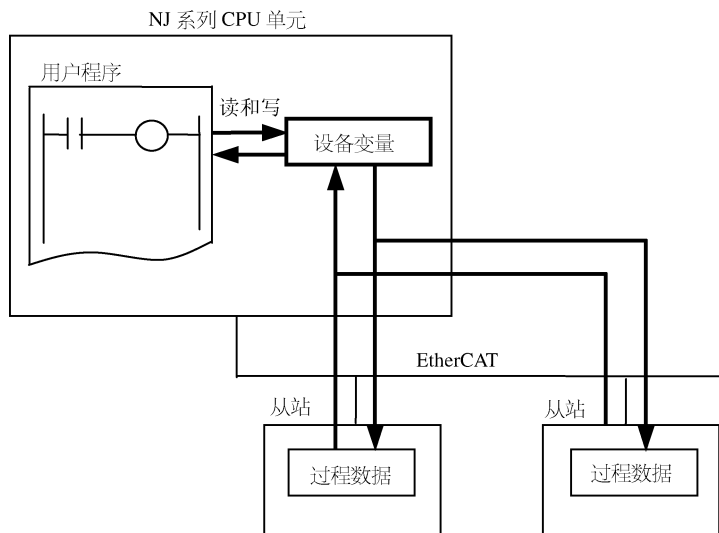


图 6-40 通过设备变量访问

(2) 仅伺服驱动和编码器输入从站

如图 6-41 所示, 将指定轴变量 (结构体) 作为运动控制指令的输入-输出变量的参数, 控制 EtherCAT 伺服驱动和编码器输入从站。

轴变量 (结构体) 由下列数据组成。

- 1) 参数: 最大加速度、软件限位等。
- 2) I/O: 原点接近输入、伺服 ON 等。
- 3) 运行状态: 加速、等待到位等。
- 4) 错误和警告状态: 过度跟踪错误、加速错误等。

2. PDO 通信编程案例

(1) 单独联锁

本案例为单独联锁, 用于检查来自每一个从站的过程数据的有效性。系统定义变量 `_EC_PDSlaveTbl [1..192]` (过程数据通信从站表格) 用来检查过程数据输入和输出对相关从站 (从站#1) 是否有效。

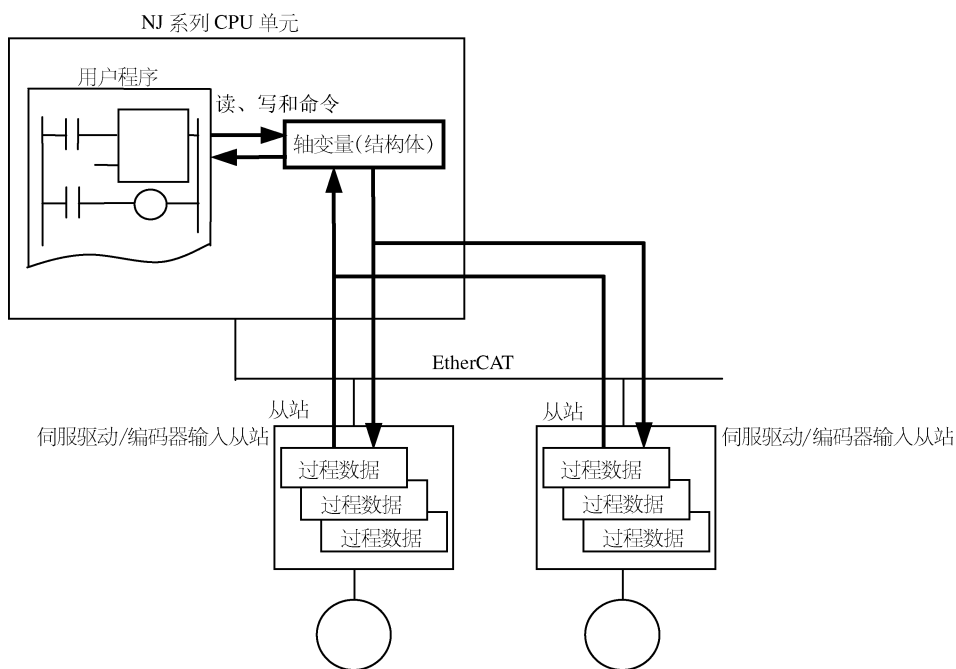
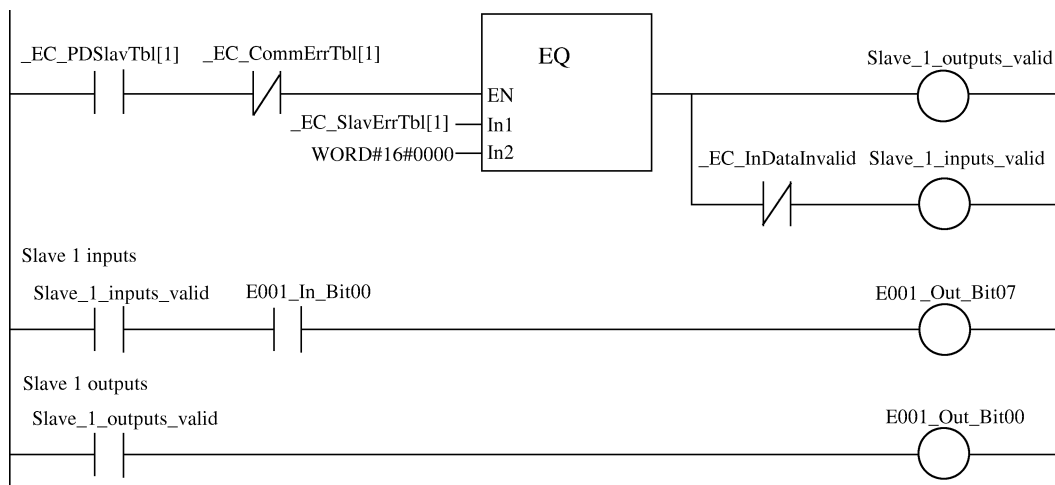


图 6-41 通过轴变量访问

1) 梯形图



2) ST

```

IF _EC_PDSlavTbl[ 1 ] = TRUE AND _EC_CommErrTbl[ 1 ] = FALSE AND
_EC_SlavErrTbl[ 1 ] = WORD#16#0 THEN
    A: = TRUE;
ELSE
    A: = FALSE;
END_IF;
(* Output valid condition *)
IF A = TRUE THEN

```

```

    Slav_Out := TRUE;
ELSE
    Slav_Out := FALSE;
END_IF;
(* Input valid condition * )
IF A = TRUE AND InDataInvalid = FALSE THEN
    Slav_In := TRUE;
ELSE
    Slav_In := FALSE;
END_IF;
(* Output data valid * )
IF Slav_Out = TRUE THEN
    E001_Out_Bit00 := TRUE;
ELSE
    E001_Out_Bit00 := FALSE;
END_IF;
(* Input data valid * )
IF Slav_In = TRUE AND E001_In_Bit00 = TRUE THEN
    E001_Out_Bit07 := TRUE;
ELSE
    E001_Out_Bit07 := FALSE;
END_IF;

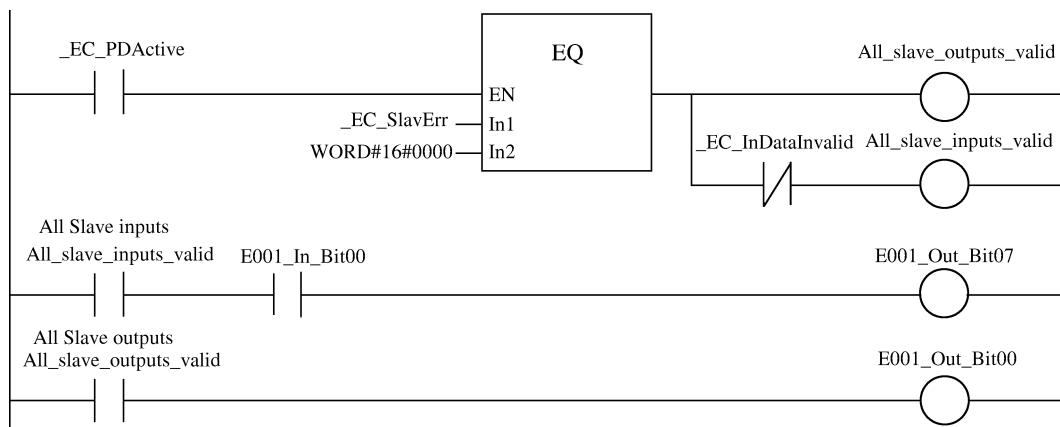
```

可以从用户程序读取系统定义变量 `_EC_PD_SlavTbl [1..192]` 的状态，确定 I/O 刷新是否正常。

(2) 总体联锁

总体联锁用来检查来自连接到网络上的所有的从站过程数据的有效性。系统定义变量 `_EC_PDActive` (过程数据通信状态) 用来检查所有从站过程数据输入和输出是否有效。

1) 梯形图




```

2) ST
IF_EC_PDActive = TRUE AND_EC_SlavErr = WORD#16#0 THEN
    A: = TRUE;
ELSE
    A: = FALSE;
END_IF;
(* Output valid condition * )
IF A = TRUE THEN
    AllSlav_Out: = TRUE;
ELSE
    AllSlav_Out: = FALSE;
END_IF;
(* Input valid condition * )
IF A = TRUE AND InDataInvalid = FALSE THEN
    AllSlav_In: = TRUE;
ELSE
    AllSlav_In: = FALSE;
END_IF;
(* Output data valid * )
IF ALLSlav_Out = TRUE THEN
    E002_Out_Bit00: = TRUE;
ELSE
    E002_Out_Bit00: = FALSE;
END_IF;
(* Input data valid * )
IF ALLSlav_In = TRUE AND E002_In_Bit00 = TRUE THEN
    E002_Out_Bit07: = TRUE;
ELSE
    E002_Out_Bit07: = FALSE;
END_IF;

```

6.4.2 SDO 通信

SDO 通信是在需要时通过使用 EtherCAT 指令访问从站的 SDO 数据。EC_CoESDORed 指令读取从站的对象字典的数据，EC_CoESDOWrite 指令则向从站的对象字典写入数据。

图 6-42 中，使用 EC_CoESDORed 指令读取欧姆龙 V2.1 R88D-KN01H-ECT 伺服驱动的软件版本，从站的节点地址是 1。

软件版本的对象索引是 16#100A，子索引是 0。读的值存储在字符串变量 VersionInfo。

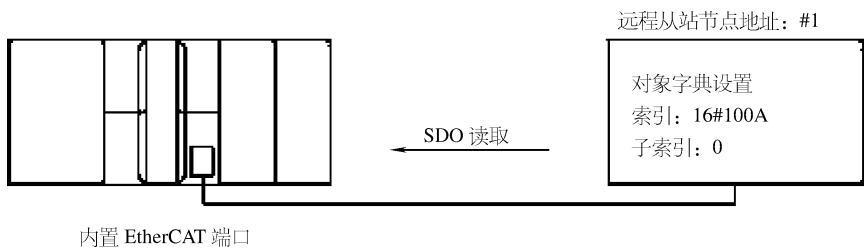
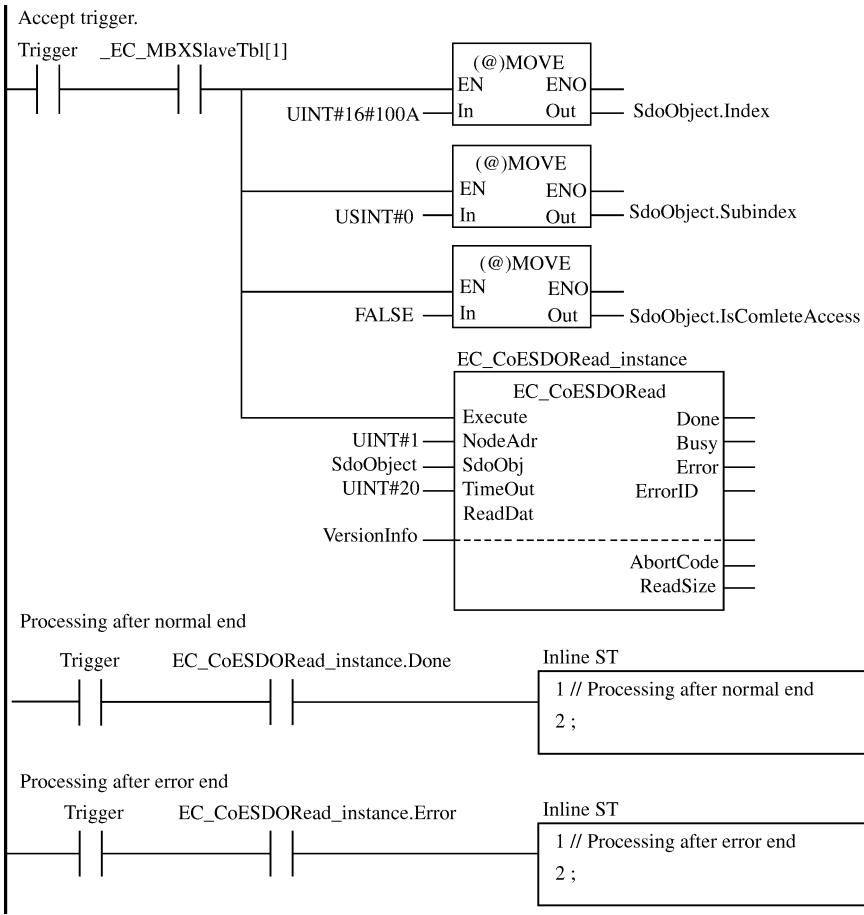


图 6-42 SDO 通信

1. 梯形图

内部变量	变量	数据类型	初始值	注释
	Trigger	BOOL	False	执行条件
	SdoObject	_sSDO_ACCESS	(Index:=0,Subindex:=0,IsCompleteAccess:=False)	SDO 参数
	VersionInfo	STRING[256]	''	读取数据
	EC_CoESDORead_instance	EC_CoESDORead		

外部变量	变量	数据类型	恒定	注释
	_EC_MBXSlaVtBl	ARRAY[1..192] OF BOOL	☒	信息通信使能从站表



2. ST

内部变量	变量	数据类型	初始值	注释
	Trigger	BOOL	False	执行条件
	SdoObject	_sSDO_ACCESS	(Index:=0,Subindex:=0,IsCompleteAccess:=False)	SDO 参数
	DoSdoRead	BOOL	False	进程
	VersionInfo	STRING[256]	''	读取数据
	NormalEnd	UINT	0	正常结束
	ErrorEnd	UINT	0	错误结束
	EC_CoESDORRead_instance	EC_CoESDORRead		

外部变量	变量	数据类型	恒定	注释
	_EC_MBXSlavTbl	ARRAY[1..192] OF BOOL	☒	信息通信使能从站表

//Detect when Trigger changes to TRUE.

IF((Trigger = TRUE)AND(DoSdoRead = FALSE)AND(_EC_MBXSlavTbl[1] = TRUE))THEN

 DoSdoRead := TRUE;

 SdoObject. Index := UINT#16#100A;

 SdoObject. Subindex := USINT#0;

 SdoObject. IsCompleteAccess:= FALSE;

 EC_CoESDORRead_instance(

 Execute:= FALSE, //Initialize instance

 ReadDat:= VersionInfo); //Dummy

END_IF;

//Execute EC_CoESDORRead instruction.

IF(DoSdoRead = TRUE) THEN

 EC_CoESDORRead_instance(

 Execute := TRUE,

 NodeAdr := UINT#1, // Node address 1

 SdoObj := SdoObject, // SDO parameter

 TimeOut := UINT#20, // Timeout time:2.0 s

 ReadDat := Versioninfo; // Read data

 IF(EC_CoESDORRead_instance. Done = TRUE) THEN

 //Processing after normal end

 NormalEnd:= NormalEnd + UINT#1;

 ELSIF (EC_CoESDORRead_instance. Error = TRUE) THEN

 //Processing after error end

 ErrorEnd:= ErrorEnd + UINT#1;

 END_IF;

END_IF;

6.4.3 EtherCAT 通信使用的指令

EtherCAT 通信使用的指令见表 6-12。

表 6-12 EtherCAT 通信使用的指令

功能	指令	功 能
启动 EtherCAT 包监控	EC_StartMon	启动 NJ 系列 CPU 单元内置 EtherCAT 主站的包监视
停止 EtherCAT 包监控	EC_StopMon	停止 NJ 系列 CPU 单元内置 EtherCAT 主站的包监视
保存包数据文件	EC_SaveMon	将捕获的数据保存在 NJ 系列 CPU 单元系统的一个文件里。包监视是 NJ 系列 CPU 单元内置 EtherCAT 主站的一个功能。捕获在 NJ 系列 CPU 单元系统内的文件在断电时不会保留
复制包数据文件到 SD 存储卡	EC_CopyMon	将捕获的保存在 NJ 系列 CPU 单元系统内的数据，复制到 SD 存储卡的一个文件中。捕获的数据先前从 NJ 系列 CPU 单元内置 EtherCAT 主站保存到 NJ 系列 CPU 单元系统。保存到 SD 存储卡的文件在断电后仍然保留
断开 EtherCAT 从站	EC_DisconnectSlave	暂时断开 EtherCAT 网络的一个从站以进行维护，如更换从站
连接 EtherCAT 从站	EC_ConnectSlave	进行维护后重新连接暂时断开的 EtherCAT 网络从站，如更换从站
获取 EtherCAT 错误状态	GetECError	获取出现在 EtherCAT 主站中控制器错误（局部故障或轻微故障）的状态和当前错误最高级别的事件代码
复位 EtherCAT 控制器错误	ResetECError	复位 EtherCAT 主站的控制器错误（只有当消除了产生错误的原因后才能执行此命令）

6.5 EtherCAT 故障诊断

NJ 系列控制器将所有的错误作为事件来管理。可以观察到出现什么错误并找到改正的方法。使用 Sysmac Studio 的故障诊断功能和 NS 系列 PT 的故障诊断器可快速查找出现的错误并找到解决的方法。

6.5.1 查找错误的方法

1. 检查指示灯状态

(1) CPU 单元运行状态

使用电源单元的 PWR 指示灯和 CPU 单元的 RUN、ERROR 指示灯，可以确定错误的事件级别。PWR 有两个状态：不亮或亮，RUN、ERROR 各有 3 个状态：不亮、闪烁或亮，这 3 个指示灯状态的不同组合，代表 CPU 单元不同的运行状态，由此可以判断出现错误时的类型及级别。

(2) EtherCAT 主站功能模块的错误状态

除了 PWR、RUN 和 ERROR 指示灯外，EtherCAT NET ERR 指示灯也用来表示错误，显示 EtherCAT 主站功能模块的错误状态。当 EtherCAT NET ERR 闪烁或常亮时，出现轻微级别

或更高级别的错误。常亮时,表示无法通过用户的操作恢复正常的状态,可能要更换 CPU 单元。闪烁时,表示通过用户的操作可以恢复正常的状态。不亮时,表示没有轻微级别或更高级别的错误。

2. 使用 Sysmac Studio 的故障诊断功能

当出现错误时,将 Sysmac Studio 和控制器在线连接,就可以检查当前的控制器错误和已发生过的控制器错误记录。也可以查找错误原因和校正的记录。

3. 使用 NS 系列 PT 的故障诊断器

当出现错误时,通过 NS 系列 PT 和控制器建立通信连接,就可以检查当前的控制器错误和已发生过的控制器错误记录。也可以查找错误原因和校正的记录。

4. 使用读错误状态指令

可在用户程序里使用指令来检查每个功能模块的错误状态。使用 GetECError 指令可以获得 EtherCAT 主站功能模块当前控制器错误的最高级别状态(局部故障或轻微故障)和最高级别事件代码。

5. 检查系统定义的变量

可以使用系统定义的变量来检查 EtherCAT 主站功能模块的错误。

(1) 错误状态变量

使用错误状态变量来检查 NJ 系列控制器每个功能模块的错误。表 6-13 列出的变量用来反映 EtherCAT 主站功能模块的错误状态。

表 6-13 与 EtherCAT 主站功能模块错误相关的系统定义变量

变量名称	数据类型	含义	功 能
_EC_ErrSta	WORD	内置 EtherCAT 错误	给出 EtherCAT 主站功能模块所有错误状态的集中错误状态
_EC_PortErr	WORD	通信端口错误	给出 EtherCAT 通信端口所有错误状态的集中错误状态
_EC_MstrErr	WORD	主站错误	给出 EtherCAT 主站错误状态的集中错误状态和 EtherCAT 主站检测出的从站错误状态
_EC_SlavErr	WORD	从站错误	给出所有 EtherCAT 从站的所有错误状态的集中错误状态
_EC_SlavErrTbl	ARRAY[1..192] OF WORD	从站错误表格	给出每个 EtherCAT 从站的错误状态

表 6-13 中错误状态变量的各个位有特定的含义,位取值 TRUE 或 FALSE,TRUE 时代表检测到相应的错误,详见有关手册。

(2) 其他与错误相关的系统定义变量

对于 EtherCAT 主站功能模块,除了错误状态变量外,还有其他系统定义变量,也反映错误状态,详见有关手册。

6.5.2 与 EtherCAT 主站功能模块相关的错误

1. EtherCAT 主站功能模块错误的分布

如图 6-43 所示, EtherCAT 主站功能模块内部出现错误,或内置 EtherCAT 端口、Ether-

CAT 从站出现错误，会在 NJ 系列 CPU 单元里形成事件记录。

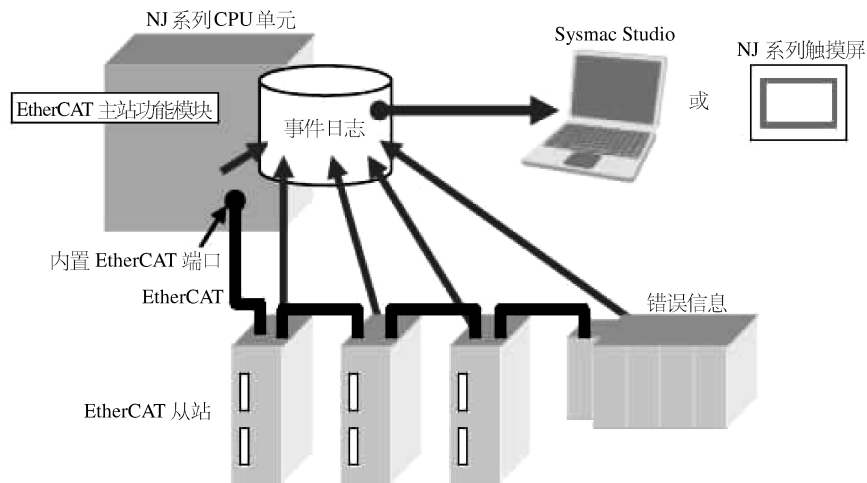


图 6-43 EtherCAT 事件记录

如果多个从站在同一时间发生以下任一错误，只有最接近主站的从站错误被记录在事件日志中，距离主站较远的从站的同样错误不记录在事件日志中。

- 1) 网络配置检验错误。
- 2) 过程数据通信错误 (断开连接电缆引起的)。
- 3) 从站节点地址重复。
- 4) 从站初始化错误。

2. 错误来源

在 EtherCAT 主站功能模块中有以下三类错误来源：

1) 通信端口错误。如果在整体 EtherCAT 通信中检测到一个错误，通信端口错误状态变量相应的位显示错误。

2) EtherCAT 主站错误。如果 EtherCAT 主站检测到其本身的设置或处理中的错误，主站错误状态变量的相应位显示错误。如果 EtherCAT 主站检测到从站的一个错误，主站错误状态变量的相应位显示错误。

3) EtherCAT 从站错误。如果 EtherCAT 主站检测一个从站错误，从站错误状态会显示主站检测到一个错误。

EtherCAT 主站周期性地读取从站错误状态信息，同时将系统定义的变量作为 I/O 变量更新。EtherCAT 主站会在从站错误表里将不报告错误状态的 EtherCAT 从站位置为 FALSE。

3. 事件级别

错误的事件级别及产生的操作见表 6-14。

表 6-14 错误的事件级别及产生的操作

错误的事件级别	操 作
重大故障	该事件级别下的错误使所有 NJ 系列控制器的控制操作停止
局部故障	该事件级别下的错误使 NJ 系列控制器某功能模块的所有控制操作停止。如果 EtherCAT 主站功能模块发生局部故障错误，EtherCAT 主站功能模块所有功能停止

(续)

错误的事件级别	操 作
轻微故障	该事件级别下的错误使 NJ 系列控制器某功能模块的一些控制操作停止。如果 EtherCAT 主站功能模块发生轻微故障错误, EtherCAT 通信能进行, 但控制可能会受影响。需要用户进行设置操作
观察	观察级别的错误不会影响 NJ 系列控制器的控制操作。报告观察是防止它们发展成轻微故障或更高级别故障
信息	归类为信息的事件提供非错误的信息

三种来源产生的事件类型及级别见表 6-15。

表 6-15 事件类型及级别

来源	级别	通信端口	EtherCAT 主站	EtherCAT 从站
重大故障		● 用户程序/控制器配置和设置传送错误 (没有来源分类)		
局部故障		● 通信控制器错误 ● MAC 地址错误	● EtherCAT 处理错误	● 无
轻微故障		● 链接 OFF 错误	● 网络配置信息错误 ● 网络配置校验错误 ● 网络配置错误 ● 过程数据接收超时错误 ● 过程数据发送超时错误 ● 从站节点地址重复 ● 从站初始化错误	● 网络配置校验错误 ● 过程数据通信错误 ● 从站节点地址重复 ● 从站初始化错误 ● 从站应用错误
观察		● 无	● EtherCAT 信息错误	● 检测到紧急消息
信息		● 无	● 错误复位	● 从站断开连接 ● 从站连接

当 EtherCAT 网络系统发生一个轻微错误时, 可以设置所有从站停止通信, 或者和出错从站之前的从站继续通信, 如图 6-44 所示, 选择“故障弱化操作”。

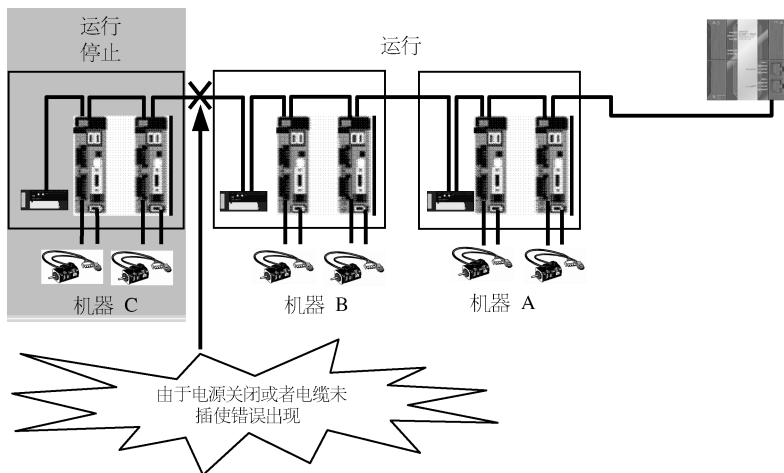


图 6-44 故障弱化操作

在菊花链连接中，与位于出错从站下方（左侧）所有从站的通信将停止，从站上的机器随之停止运行。

如果使用分支单元独立连接机器和从站，在机器运行的同时也能维护或者纠正出现的错误。

6.5.3 复位错误

有三种复位错误方法。

- 1) 从 Sysmac Studio。
- 2) 从 NS 系列 PT。
- 3) 在用户程序中执行复位 EtherCAT 系统错误的指令 (ResetECError)。

6.5.4 诊断和统计信息

EtherCAT 主站提供诊断和统计信息。参考这些内容，可以分离出错误的原因，并获得网络状态。按下列步骤从 Sysmac Studio 访问这些信息。

- 1) 启动 Sysmac Studio 与控制器在线连接。
- 2) 双击多视图浏览器中的“配置和设置”下的“EtherCAT”；或者右击“EtherCAT”，从菜单选择“编辑”。
- 3) 右击显示在编辑面板中的 EtherCAT 主站，从菜单中选择“显示诊断/统计信息”，如图 6-45 所示，则弹出“诊断/统计信息”对话框，如图 6-46 所示。



图 6-45 选择“显示诊断/统计信息”

对话框下方有 3 个按钮：“输出文件”将信息输出到一个 CSV 文件，“更新”从控制器

获得诊断/统计信息并修改显示,“清空”将控制器的诊断/统计信息清零。



图 6-46 “诊断/统计信息”对话框

6.6 EtherCAT 其他功能

6.6.1 监视包

如图 6-47 所示, 监视包存储由 EtherCAT 主站最近发送和接收的与时间相关的一定数目的数据包。利用 EtherCAT 通信指令或 Sysmac Studio, 将捕捉到的包数据存储在 CPU 单元的

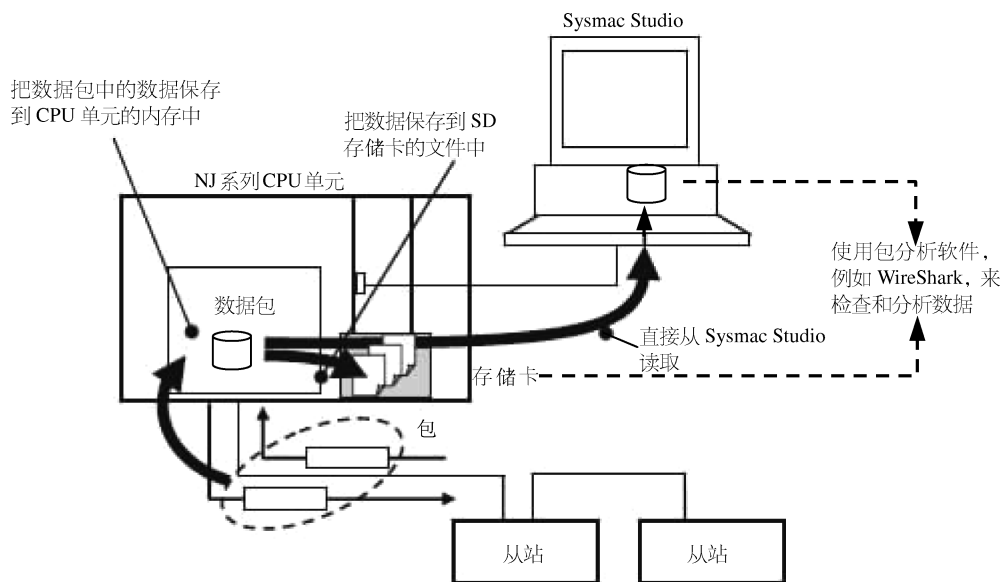


图 6-47 监视包

包数据文件中。要得到存储在 CPU 单元内存中的包数据，可以使用下列两种方法之一。

- 1) 直接从 Sysmac Studio 读取。
- 2) 存储到插在 CPU 单元的 SD 存储卡上。

使用包分析器软件，例如 WireShark，可观察捕捉到的包数据。也可将这些数据用于分析研究，例如错误分析和数据挖掘。

1. 启动和停止包监视

在用户程序中使用指令或使用 Sysmac Studio 相关操作，可启动和停止包监视。

启动 EtherCAT 包监视的指令为 EC_StartMon，停止 EtherCAT 包监视的指令为 EC_StopMon。

下面介绍使用 Sysmac Studio 的操作过程。

- 1) 启动 Sysmac Studio 与控制器在线连接。
- 2) 双击多视图浏览器中的“配置和设置”下的“EtherCAT”；或者右击“EtherCAT”，从菜单选择“编辑”。
- 3) 右击显示在编辑面板中的 EtherCAT 主站，如图 6-48 所示，从菜单中选择“显示包监视”。



图 6-48 启动包监视

- 4) 在弹出的包监视对话框，点击“开始”按钮，则开始监视包，如图 6-49 所示。如果点击“停止”，则停止监视包。

NJ 上电时，包监视会被启动。

2. 保存包数据

保存包数据有两种方式。

- 1) 从 Sysmac Studio 读取包数据并存储到文件中。如图 6-50 所示，停止监视包数据，点

击“保存包数据”，显示出保存对话框，输入一个文件名字，然后点击“保存”按钮，将包数据文件保存为 TCPDump 格式，扩展名为 .cap。



图 6-49 启动监视



图 6-50 保存包数据

2) 从 CPU 单元的内存中保存到 SD 存储卡中。停止监视包，执行指令 EC_SaveMon (保存 EtherCAT 包)，将收集到的一定数目的包数据保存到 CPU 单元内存中。然后，执行指令 EC_CopyMon (转移 EtherCAT 包)，将保存在 CPU 单元内存中的包数据，以文件形式存储到插在 CPU 单元的 SD 存储卡上。可以指定文件名称，存储文件的数目受 SD 存储卡可用的存储空间限制。监视包指标见表 6-16。

表 6-16 监视包指标

项 目	指 标
单个包数据文件最大数据长度	12MB
包最大数目	3904 个
包数据文件格式	TCPDump 格式 (.cap)
时间信息	当包监视启动时，记录过去的时间 (单位: μ s)
包数据文件存储位置	CPU 单元系统: 1 个文件 插入 CPU 单元的存储卡: 多个文件 (取决于 SD 存储卡的容量)

6.6.2 从站的使能与禁用

通过 Sysmac Studio 可以使能或禁用一个从站。

使能从站意味着从站能操作。

禁用从站意味着从站不能操作。

当现在有一个从站没有连接进实际的 EtherCAT 网络，但是如果将来要拓展，可以先把它登记为一个禁用从站。

注意，即使禁用从站，一样可以设置变量，一样可以在用户程序中使用这个禁用从站默认变量或设备变量等。

可以通过传送设置给 CPU 单元来改变从站使能与禁用 (CPU 必须在 PROGRAM 模式下)。

6.6.3 从站的断开连接与重新连接

EtherCAT 网络可以暂时停止或重新启动与指定的从站及其后续站点的通信，而不停止整个网络的通信。当从站出现故障需要维修或替换时，将这台设备从网络中断开，会导致所有其他连接到该设备之下的从站都停止通信。进行替换操作时，首先将指定的从站从网络中断开并移走，再将新换上的从站重新连接，如图 6-51 所示。

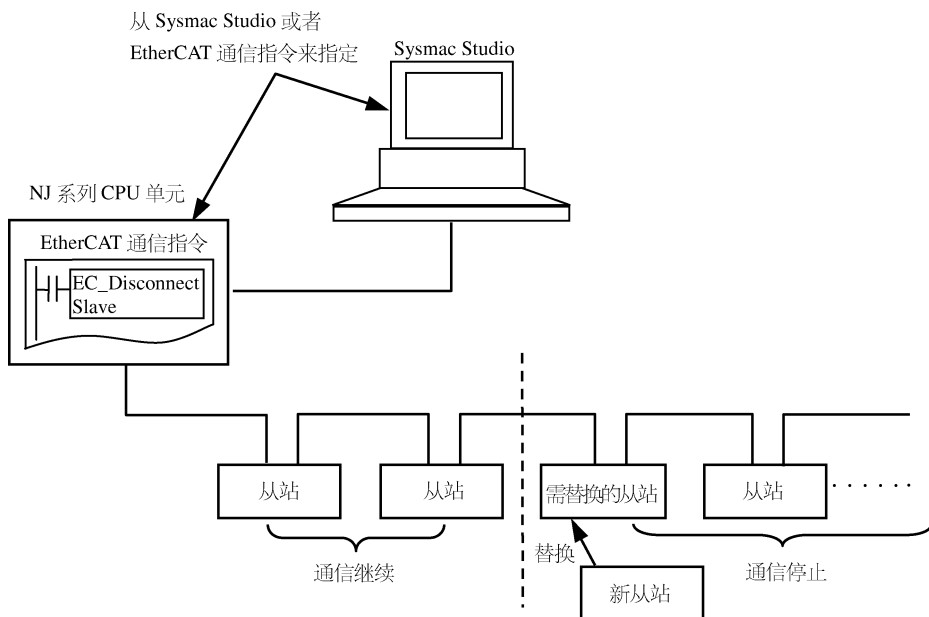
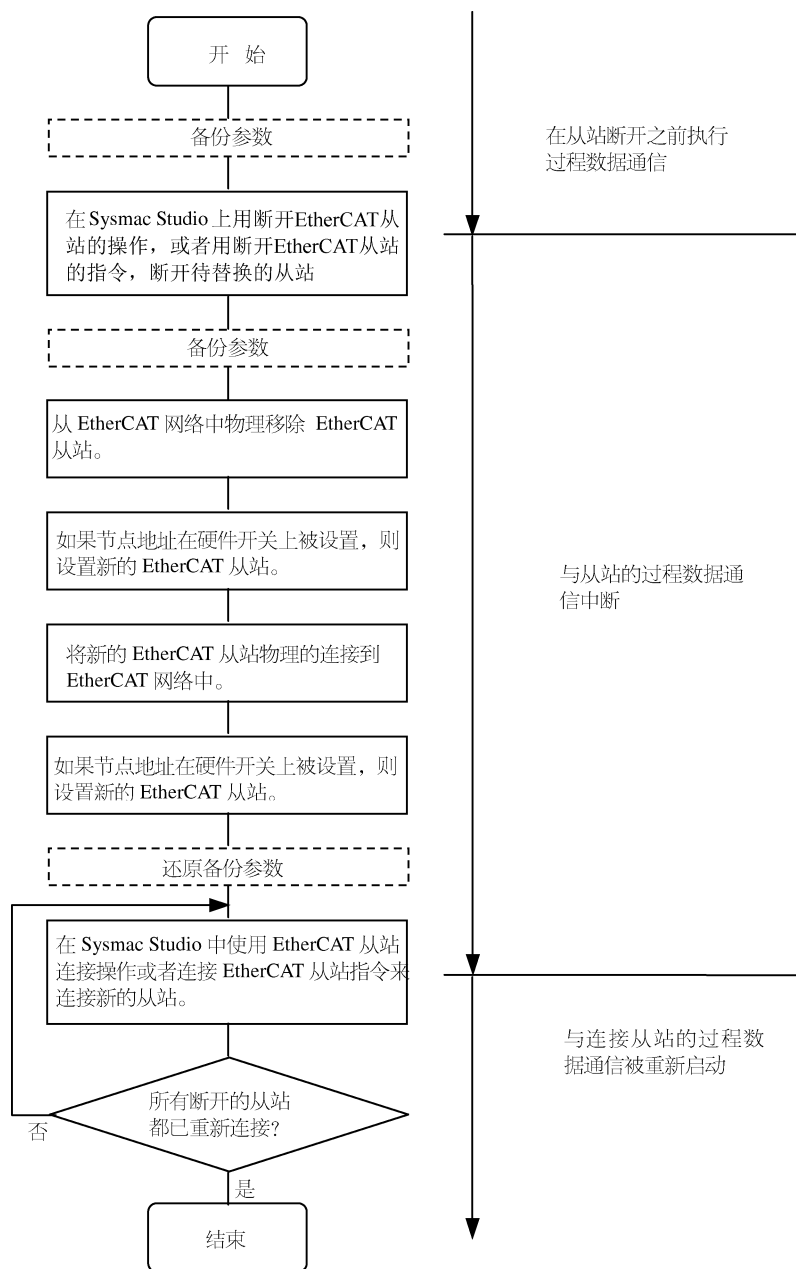


图 6-51 EtherCAT 从站替换

将从站从网络中断开或重新连接有两种方法。

- 1) 从 Sysmac Studio 断开或重新连接指定从站。
- 2) 执行断开 EtherCAT 从站指令 EC_DisconnectSlave 和连接 EtherCAT 从站指令 EC_ConnectSlave。

图 6-52 所示为 EtherCAT 从站断开和重新连接的操作步骤。



注：虚线框所示的步骤取决于系统。

图 6-52 EtherCAT 从站断开和重新连接的操作步骤

第 7 章 NJ 的运动控制

7.1 概述

NJ 的 CPU 单元内置了一种软件模块，称为运动控制功能模块 (Motion Control Function Module)，简称 MC (Motion Control) 功能模块。通过内置在 CPU 单元上的 EtherCAT 端口，MC 功能模块可完成多达 64 轴的运动控制。MC 功能模块进行运动控制时，通过 EtherCAT 端口与连接到该端口的伺服驱动器进行周期通信，从而实现高速度、高精度的伺服驱动控制，是进行大规模复杂运动控制的一种理想选择。

1. NJ 运动控制系统的结构

MC 功能模块从设备和控制面板接收传感器信号状态，从用户程序中执行的运动控制指令接收命令，使用以上两项对伺服驱动和编码器输入终端进行运动控制。

NJ 运动控制系统的结构如图 7-1 所示。Sysmac Studio 通过 USB 电缆或 Ethernet 电缆与 CPU 连接，使用 Ethernet 电缆时，与 CPU 上内置的 EtherNet/IP 端口连接。NJ 的运动控制通过 EtherCAT 网络实现，因此，进行 EtherCAT 网络配置是必要的。

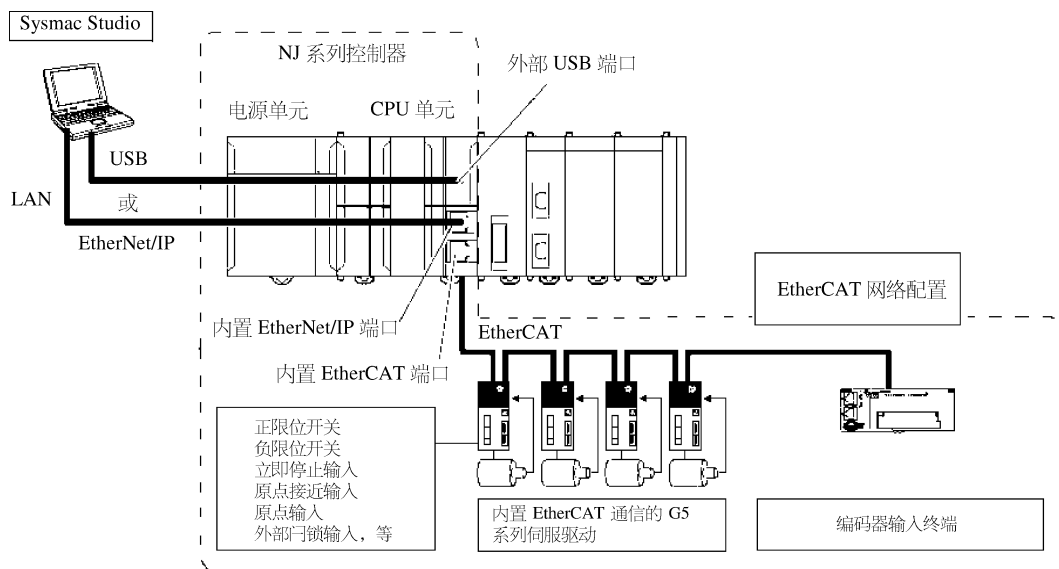


图 7-1 NJ 运动控制系统的结构

2. NJ 运动控制的特点

(1) 基于 PLCopen 的运动控制指令

MC 功能模块的运动控制指令基于由 PLCopen 协会标准化的运动控制块。这些指令允许用户编程进行单轴 PTP (Point to Point) 定位、插补控制、同步控制 (例如，电子凸轮)、速

度控制和扭矩控制。根据实际应用,在运动控制指令执行时,可以设定速度、加速度、减速度和加加速度,达到灵活、柔性控制效果。

加加速度是加速度或减速度的变化率。如果设定加加速度恒定,则加速或减速时,速度曲线会呈 S 形。

(2) 使用 EtherCAT 通信的数据传送

MC 功能模块通过内置 EtherCAT 通信,与 OMRON 的 G5 系列伺服驱动相结合,以高速的数据通信交换所有的控制信息。各种控制命令均通过数据通信传送。这意味着伺服电动机的运行性能被最大化,不受接口规格限制,例如,编码器反馈脉冲的响应频率。在上位机上,可以使用伺服驱动的各种控制参数和监控数据,集中统一管理信息。

3. NJ 运动控制的性能指标

NJ 运动控制的性能指标见表 7-1。

表 7-1 NJ 运动控制的性能指标

项 目		规 格		
		NJ501-1300	NJ501-1400	NJ501-1500
可控制的伺服驱动		带内置 EtherCAT 通信的欧姆龙 G5 系列伺服驱动		
可控制的编码器输入终端		欧姆龙 GX 系列 GX-EC0211/EC0241 编码器 I/O 终端		
控制方法		使用 EtherCAT 通信的控制命令		
控制模式		位置控制(周期同步位置控制模式) 速度控制(周期同步速度控制模式) 扭矩控制(周期同步扭矩控制模式)		
被控轴的数量	被控轴的最大数量	16 轴	32 轴	64 轴
	单轴控制	最多 16 轴	最多 32 轴	最多 64 轴
	直线插补控制	每个轴组最多 4 轴		
	圆弧插补控制	每个轴组最多 2 轴		
轴组数目		最多 32 个轴组		
单位转换	显示单位	脉冲、毫米、微米、纳米、度或英寸		
	电子齿轮比	电动机每转的脉冲数/电动机每转一圈移动的距离		
可以监视的位置		命令位置 and 实际位置		
位置命令值		正或负的长实数(LREAL)或 0(命令单位)		
速度命令值		正或负的长实数(LREAL)或 0(命令单位/s)		
加速度命令值和减速度命令值		正的长实数(LREAL)或 0(命令单位/s ²)		
加加速度命令值		正的长实数(LREAL)或 0(命令单位/s ³)		
倍率因子		0.00% 或 0.01% ~ 500.00%		
轴类型		伺服轴、虚拟伺服轴、编码器轴或虚拟编码器轴		
运动控制周期		与 EtherCAT 通信的过程数据通信周期相同		
凸轮	凸轮数据点	每个凸轮表最多 65535 点 所有凸轮表总计最多 1048560 点		
	凸轮表数量	最多 640 个表		
	凸轮曲线	由凸轮编辑器创建 可以指定位置(主轴相位和从轴位移)改变每一段的相节距		
	覆盖凸轮数据	凸轮数据可以被用户程序覆盖		

4. 运动控制的操作流程

使用 NJ 进行运动控制时，通常要完成以下操作：

(1) 设置

- 1) 创建工程；
- 2) 生成 EtherCAT 网络配置；
- 3) 添加轴；
- 4) 分配轴；
- 5) 设置轴参数；
- 6) 对控制器做设置。

(2) 传送

将工程传送到控制器。

(3) 检查连接

在 MC 试运行标签页操作，通过监视输入信号检查接线，进行点动、原点返回、绝对定位和相对定位等操作。

(4) 编写手动操作的简单程序

编写点动程序、通过用户程序对轴执行点动。

(5) 原点返回

编写原点返回程序，通过用户程序对轴执行原点返回。

(6) 编写运动控制程序

(7) 调试

如果有错误，读取错误代码，消除错误原因并清除错误记录。

(8) 运行

控制器运行，操作机器。

(9) 维护

完成定期维护。

7.2 运动控制配置与原理

7.2.1 CPU 单元的内部配置

NJ 系列 CPU 单元内部软件配置如图 7-2 所示，PLC 功能模块在 OS 之上运行，其他功能模块在 PLC 功能模块之上运行，MC 功能模块是完成运动控制的一个软件模块。



图 7-2 NJ 系列 CPU 单元的内部软件配置

各个软件模块的作用如下。

PLC 功能模块：管理整体调度，执行用户程序，将命令发送到 MC 功能模块，并提供到 USB 和 SD 存储卡的接口。

运动控制 (MC) 功能模块：根据在用户程序中执行运动控制指令时产生的命令来完成运动控制，将数据发送到 EtherCAT 主站功能模块。

EtherCAT 主站功能模块：作为 EtherCAT 主站与 EtherCAT 从站通信。

7.2.2 运动控制配置

运动控制配置如图 7-3 所示。

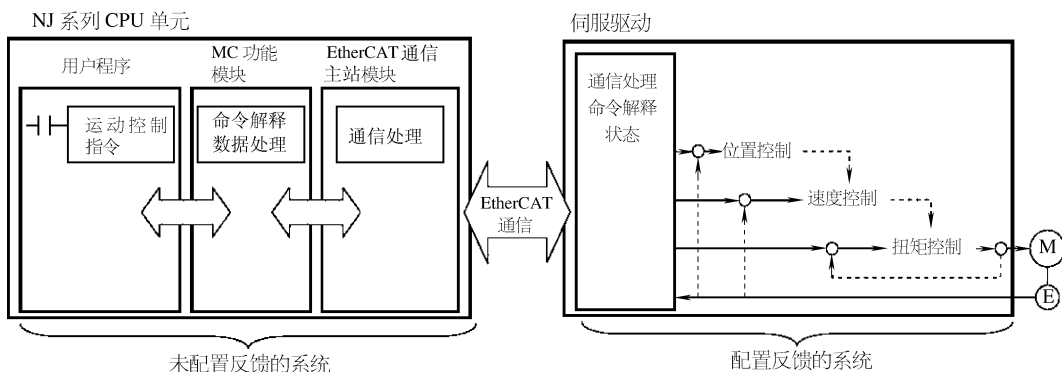


图 7-3 运动控制配置

一个带有伺服驱动的系统一般采用半闭环方式控制电动机运行。半闭环使用一个连接在电动机上的编码器来检测电动机响应命令值完成的旋转量。这可以作为机器移动距离的反馈。计算出命令值与电动机实际旋转之间的跟踪误差，并进行控制，可使跟踪误差为 0。

在使用 MC 功能模块的机器配置里，不能为 CPU 单元里用户程序的命令提供反馈信息。反馈系统内置在伺服驱动里。

当运动控制指令在用户程序里执行时，MC 功能模块解释生成的命令。

然后，MC 功能模块根据命令解释的结果，在一个固定周期内完成运动控制处理。它产生的命令值发送到伺服驱动，产生下列命令值：目标位置、目标速度和目标扭矩。

在 EtherCAT 通信的每个过程数据通信周期内，命令值通过 PDO 通信发送。

基于 EtherCAT 通信每个过程数据通信周期内接收到的命令值，伺服驱动完成位置环控制、速度环控制以及扭矩环控制。

在 EtherCAT 通信的每个过程数据通信周期内，编码器的当前值和伺服驱动的状态被发送到 CPU 单元。

7.2.3 运动控制原理

运动控制指令可以在主周期任务或在一个优先级 16 的周期性任务里执行。

1. 在主周期任务中使用运动控制指令

如果需要高速运动控制，将运动控制指令 (FB) 放在主周期任务里。

主周期任务执行运动控制指令如图 7-4 所示。

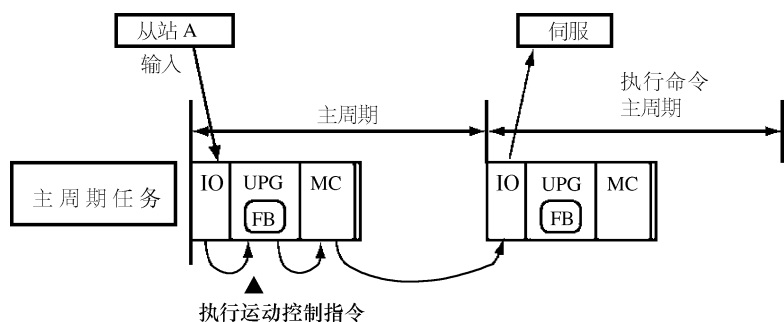


图 7-4 主周期任务执行运动控制指令

(1) 装载数据

在 I/O 刷新 (IO) 过程中，装载 EtherCAT 从站 (从站 A) 的输入数据。

(2) 指令执行

在用户程序执行 (UPG) 过程中，根据装载的数据，执行运动控制指令 (FB)。同时，刷新运动控制指令的输出变量。

(3) 命令产生

主周期任务的用户程序执行之后紧跟的运动控制 (MC) 过程中，完成执行的运动控制指令 (FB) 对应的运动处理，生成给伺服驱动的执行命令。

(4) 发送命令

生成的执行命令，在下一个周期的 I/O 刷新过程中发送到伺服驱动。

2. 在优先级 16 周期任务中使用运动控制指令

如果不需要高速的运动控制，或用户程序太长，可把运动控制指令 (FB) 放在优先级 16 的周期任务里，如图 7-5 所示。

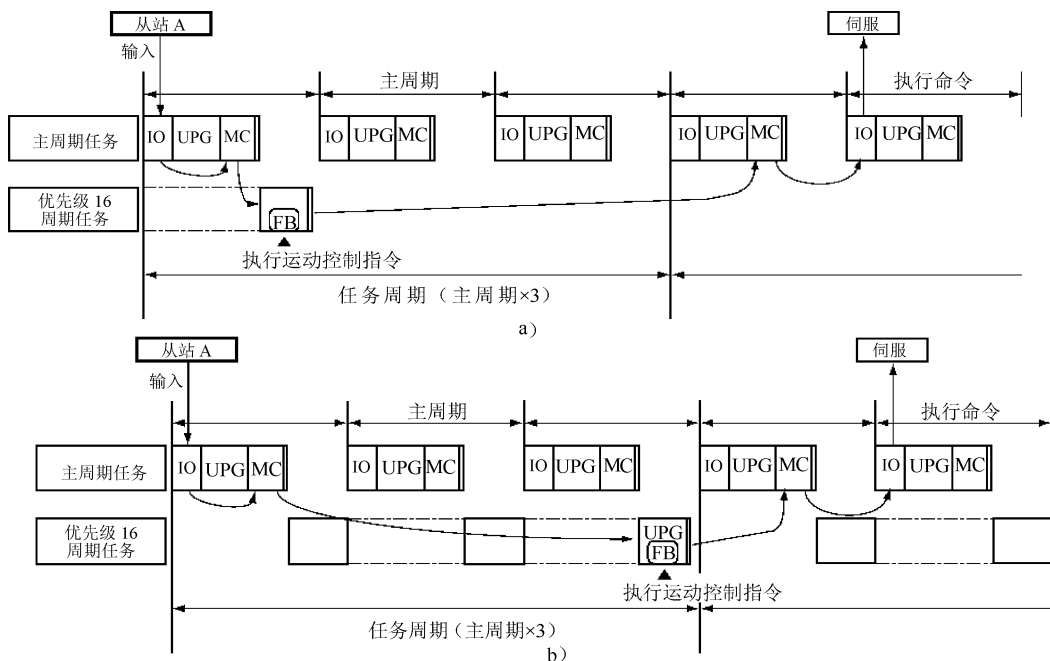


图 7-5 在优先级 16 周期任务中使用运动控制指令

(1) 处理时序

在优先级 16 周期任务里执行的运动控制指令 (FB)，其运动控制处理与任务一起在相同时间内被执行，因此，对多轴的处理可以同时执行或停止。

1) 装载数据：在 I/O 刷新 (IO) 过程中，装载 EtherCAT 从站 (从站 A) 的输入数据。

2) 指令执行：在用户程序执行 (UPG) 过程中，根据装载的数据，执行运动控制指令 (FB)。同时，刷新运动控制指令的输出变量。

3) 命令产生：在优先级 16 周期任务的任务周期内，执行运动控制指令 (FB)。在周期任务的下一个主周期任务的运动控制处理时，完成对应的运动控制处理，生成给伺服驱动的执行命令。

4) 发送命令：生成的执行命令，在下一个周期的 I/O 刷新过程中发送到伺服驱动。

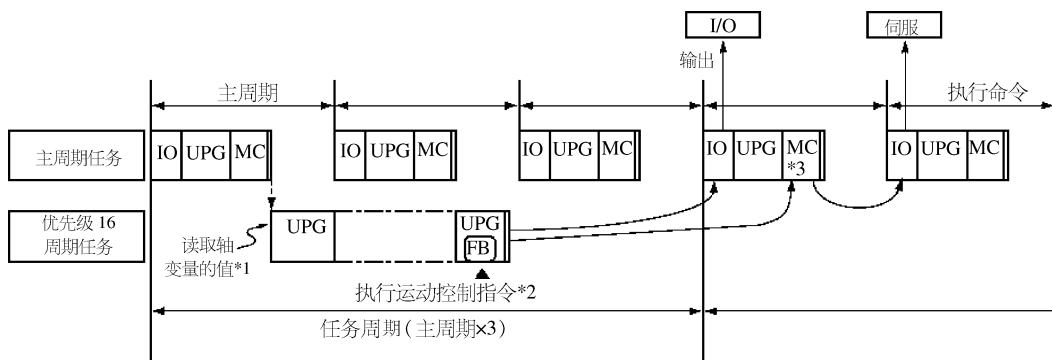
(2) 轴变量更新时序

轴变量是为一些轴参数和监视信息设置的系统定义变量，例如，由 MC 功能模块控制的轴的实际位置和错误信息。

如果在一个优先级 16 周期任务中访问一个轴变量，应使用在任务周期开始时读出的变量值。

此外，轴变量的值不会在运动控制指令 (FB) 执行时被写入。会在下一个周期任务开始时写入运动控制处理 (MC)。

轴变量更新时序如图 7-6 所示。



* 1 轴变量的值在周期任务的用户程序执行开始时读取。

* 2 轴变量的值不会在周期任务中的运动控制指令 (FB) 执行时被写入。

* 3 在运动控制处理中 (MC)，值被写入。

图 7-6 轴变量更新时序

当运动控制指令放在一个周期性任务，如果周期任务的任务周期延长，那么伺服驱动的响应时间会增加。

确保所有轴在紧急停止时可以安全停下来，包括来自外部设备的紧急停止命令。

优先级 16 周期任务的运动控制指令执行时序和 I/O 控制的执行时序是不同的，设计用户程序要考虑这一点。

3. 在两个不同类型的任务中使用运动控制指令

对于同一轴，如果有需要高速运动控制的处理和不需要高速运动控制的处理，可以把运

动控制指令 (FB) 同时放在主周期任务和优先级 16 周期任务中。

在优先级 16 周期任务的周期内, 如果运动控制指令在两个任务中都执行, MC 功能模块会首先完成主周期任务中指令的运动处理。

如图 7-7 所示, MC_MoveAbsolute 指令在优先级 16 周期任务中执行, MC_MoveRelative 在主周期任务里对相同的轴操作。MC 功能模块首先执行 MC_MoveRelative, MC_MoveAbsolute 会以指令多次执行的方式执行。

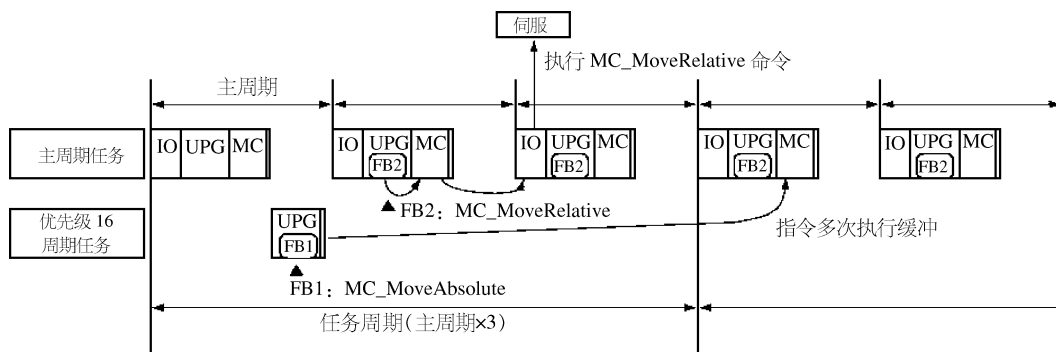


图 7-7 在两个不同类型的任务中使用运动控制指令

一个运动控制指令输出变量的值和运动控制的系统定义变量的值, 在执行该指令的任务的 I/O 刷新时会改变。因此, 在不同的任务中用运动控制指令控制相同的轴时, 可能观察到任务不同会有不同的情况。在开发用户程序之前有必要全面了解每个任务的过程。

如果在主周期任务和优先级 16 周期任务里包含同一个轴的运动控制指令, 当开发用户程序时, 务必搞清楚以下几点: 运动控制指令的执行顺序、运动控制系统定义变量的更新时序和命令值的输出时序。

如果在多任务中对同一个轴的运动控制使用系统定义变量, 开发用户程序时, 务必搞清楚运动控制中系统定义变量更新时序的差别。

7.2.4 EtherCAT 通信和运动控制

通过 CPU 单元中 EtherCAT 主站功能模块的 PDO 通信, MC 功能模块控制伺服驱动和编码器输入从站。

1. CoE (CAN Application Protocol over EtherCAT)

MC 功能模块使用 CAN 协议通过 EtherCAT 网络与 EtherCAT 的从站交换数据。使用 CoE, 根据对象字典 (object dictionary) 的数据规范, 设定从站的参数和控制信息。在控制器 (通信主站) 和从站之间有两种类型的通信:

- 1) 过程数据通信: 固定周期内实时交换数据。
- 2) SDO 通信: 需要时交换数据。

MC 功能模块将 PDO 通信用于在固定控制周期内刷新 I/O 数据的命令, 例如, 伺服电动机定位控制。将 SDO 通信用于在指定时间读写数据的命令, 例如, 参数传送。

2. EtherCAT 主站功能模块和 MC 功能模块的关系

NJ 系列 CPU 单元可以对连接的 EtherCAT 从站执行顺序控制和运动控制，如图 7-8 所示。

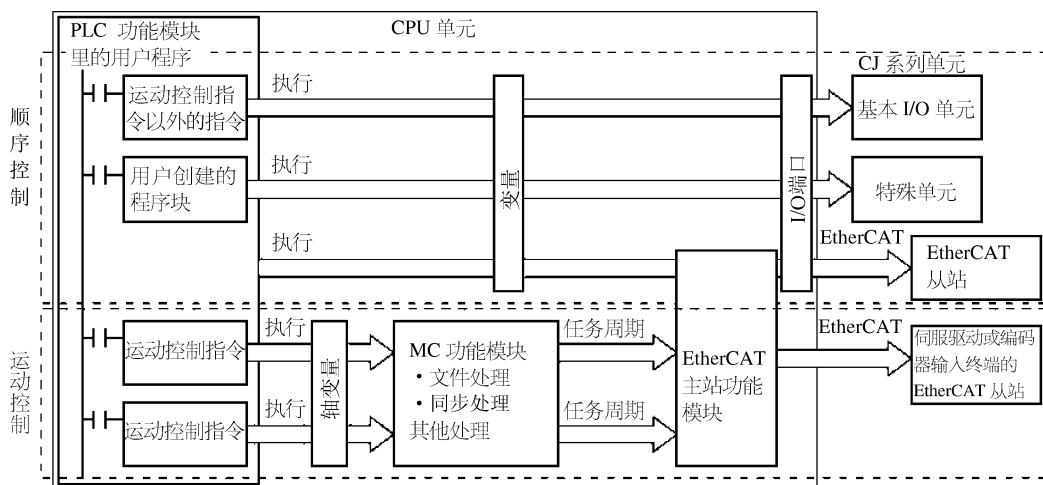


图 7-8 顺序控制和运动控制

(1) 顺序控制

在 EtherCAT 编辑选项卡里创建 EtherCAT 配置时，自动创建配置从站的 I/O 端口。通过运动控制指令以外的指令执行顺序控制。

(2) 运动控制

在 EtherCAT 编辑选项卡里创建 EtherCAT 配置时，自动创建配置从站的 I/O 端口。在运动控制设置视图中创建轴变量并且为执行运动控制的轴分配 EtherCAT 从站。通过运动控制指令来执行运动控制。

有两种类型的 EtherCAT 从站可以分配给轴变量：伺服驱动从站和编码器输入端子从站。

- 对于运动控制指令以外的指令，命令不是直接通过 PDO 通信发送给分配了轴变量的 EtherCAT 从站的。但是，可以通过轴变量间接访问 EtherCAT 从站的状态。

- 可以使用 SDO 通信，读写分配了轴变量的 EtherCAT 从站的对象。但是，不要通过 SDO 通信去写映射到 PDO 通信的对象。如果这样做，从站的操作将取决于从站规范。对于欧姆龙从站，SDO 通信会出错。

- 如果伺服驱动和编码器输入终端的 EtherCAT 从站没有分配轴变量，必须用对一般 EtherCAT 从站同样的方式对它们执行顺序控制。

3. 过程数据通信周期和运动控制周期之间的关系

如图 7-9 所示，当运动控制指令在用户程序中执行时，PLC 功能模块向 MC 功能模块发送运动控制命令。这样，MC 功能模块完成基于这些命令的运动控制处理，并且把处理的结果作为命令发送给 EtherCAT 中的伺服驱动器。如下关系成立：

主周期 = 运动控制周期 = EtherCAT 通信的过程数据通信周期

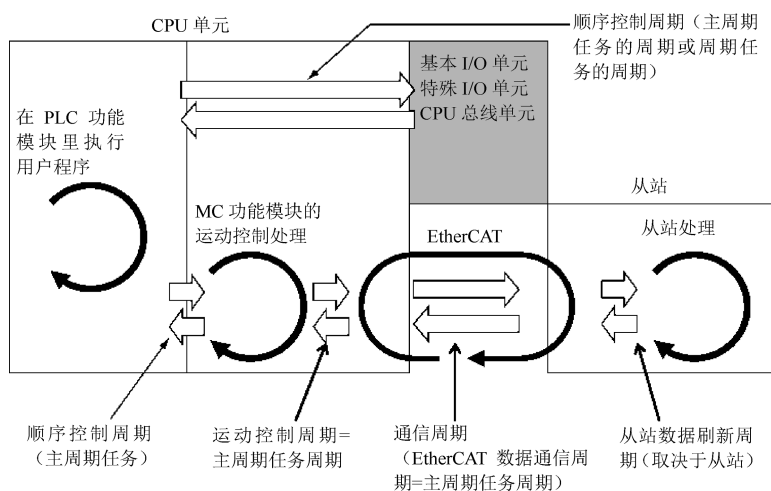


图 7-9 过程数据通信周期和运动控制周期之间的关系

7.3 轴

7.3.1 轴简介

在运动控制系统中，运动控制的目标称为轴。轴可以是一个通过 EtherCAT 连接的实际的伺服驱动或编码器，或者是在 MC 功能模块中虚拟的伺服驱动或编码器。

MC 功能模块支持的轴类型见表 7-2。

表 7-2 轴 类 型

轴 类 型	说 明
伺服轴	EtherCAT 从站伺服驱动使用的轴。它们被分配给实际的伺服驱动。一个伺服电动机使用一个轴
虚拟伺服轴	虚拟轴只存在于 MC 功能模块内,实际的伺服驱动不使用它们。例如,用它们作主轴来进行同步控制
编码器轴	编码器轴使用 EtherCAT 从站编码器输入终端。编码器轴分配给实际的编码器输入终端。如果一个编码器输入终端包含两个编码器输入,每个编码器输入起一个轴的作用
虚拟编码器轴	这些轴用作虚拟编码器操作。在没有实际编码器时,虚拟编码器轴可临时顶替编码器轴

可用的伺服驱动是 OMRON 的 G5 系列伺服驱动，内置了 EtherCAT 通信。

可用的编码器输入终端是 OMRON 的 GX 系列 GX-EC0211/EC0241 编码器 I/O 端子。

虚拟编码器轴用在运动控制指令中，更新虚拟编码器轴的实际位置。计数不能用在不支持虚拟编码器轴的指令中。

与轴相关的元素见表 7-3。每个轴都有这些元素，NJ501-1300 有 16 轴的轴参数，NJ501-1400 有 32 轴的轴参数，NJ501-1500 有 64 轴的轴参数。

表 7-3 轴 元 素

配置 元素	说 明
轴参数	轴参数是为 MC 功能模块控制的轴操作设置的最大速度、点动、原点返回以及其他选项,使用 Sysmac Studio 来设置轴参数

(续)

配置元素	说 明
轴变量	轴变量是系统定义变量,用来表示 MC 功能模块控制的轴的实际位置、错误信息及其他轴的信息。当从 Sysmac Studio 的多视图浏览器中添加一个轴时,轴变量就会被创建。轴变量的名称可在这里设置
在用户程序中指定轴	在用户程序中,运动控制由运动控制指令来实现。完成单轴控制的运动控制指令用来生成轴命令。为通过轴命令控制一个轴,对指令的输入-输出变量 Axis,指定一个系统定义变量的轴变量名称或者是用 Sysmac Studio 设置的轴变量名称

7.3.2 轴参数

轴参数见表 7-4。

表 7-4 轴 参 数

分 类	参 数 名 称	分 类	参 数 名 称
轴基本设置	轴号	限位设置	软件限位
	轴使用		正软件限位
	轴类型		负软件限位
	节点(输入设备和输出设备)		位置偏移超出值
单位换算设置	显示单位		位置偏移警告值
	电动机转一圈的指令脉冲数	位置计数设置	计数模式
	电动机转一圈的工作行程		模最大位置设置值
操作设置	最大速度		模最小位置设置值
	最大点动速度		编码器类型
	最大加速度	伺服驱动设置	模最大位置设置值
	最大减速度		模最小位置设置值
	加速度/减速度超出	原点返回设置	原点返回方法
	换向操作选择		原点输入信号
	速度警告值		原点返回开始方向
	加速度警告值		原点输入检测方向
	减速度警告值		正限位输入时操作选项
	正扭矩警告值		负限位输入时操作选项
	负扭矩警告值		原点返回速度
	实际速度滤波器的时间常数		原点返回接近速度
	定位范围		原点返回加速度
	定位检查时间		原点返回减速度
	零位置范围		原点返回加加速度
其他操作设置	立即停止输入的停止方法		原点输入掩码距离
	限位输入的停止方法		原点偏移量
	驱动器错误复位监测时间		原点返回保持时间
	正扭矩限位最大值		原点返回补偿值
	负扭矩限位最大值		原点返回补偿速度

轴参数分类说明如下。

1. 轴基本设置

轴基本设置用来设置是否使用轴。如果使用轴，则设置轴的类型和 EtherCAT 从站设备的节点地址。

轴基本设置见表 7-5。

表 7-5 轴基本设置

参 数	功 能	设 置 范 围	默 认 值
轴号	设置轴的逻辑序号。访问_sAXIS_REF 获取这个序号,以识别轴	—	—
轴使用*	选择是否使能或禁用轴。如果对一个未定义的轴执行运动控制指令,会出现错误。但是对于一个未使用的轴,不会出现错误 0:未定义的轴 1:未使用的轴 2:使用的轴	0 ~ 2	0
轴类型	设置轴的类型,虚拟轴不需要 I/O 接线 0:伺服轴 1:编码器轴 2:虚拟伺服轴 3:虚拟编码器轴	0 ~ 3	0
节点地址 (输入设备和输出设备)	为分配给轴的 EtherCAT 从站指定节点地址。如果轴类型的参数设定为虚拟轴,不能选择节点参数	0 ~ 192	0

* 对于未定义或未使用的轴，如果执行运动控制指令，Busy 变为 TRUE。当 Execute 或 Enable 变为 FALSE，则 Busy 变为 FALSE。可以将轴设为未用轴，这样，不同的轴配置使用相同的用户程序，不需要删掉程序。

2. 单位换算设置

单位换算设置见表 7-6，这些参数设置位置单位。

表 7-6 单位换算设置

参 数 名 称	功 能	设 置 范 围	默 认 值
显示单位	设置命令位置的单位 pulse mm μm nm degree inch	0 ~ 5	0
电动机转一圈的命令脉冲数* 1	根据编码器的分辨率* 2,为命令位置设置电动机转一圈的脉冲数。根据电子齿轮比,命令值转换为脉冲数	1 ~ 4294967295	10000
电动机转一圈的工作行程* 3	为命令位置设置电动机转一圈的工作行程	0. 000000001 ~ 4294967295	10000

* 1 这是电子齿轮比的分子。

* 2 例如，如果编码器分辨率是 10000 脉冲/转，设置为 10000。

* 3 这是电子齿轮比的分母。

MC 功能模块和伺服驱动或编码器输入终端之间，定位通常以脉冲给出。对运动控制指令，使用 mm 或 degree 为显示单位，容易理解所进行的操作。

使用电子齿轮设置显示单元和 MC 功能模块中脉冲单元之间的联系。使用 Sysmac Studio 并设置电子齿轮比。

电子齿轮比的作用如图 7-10 所示。

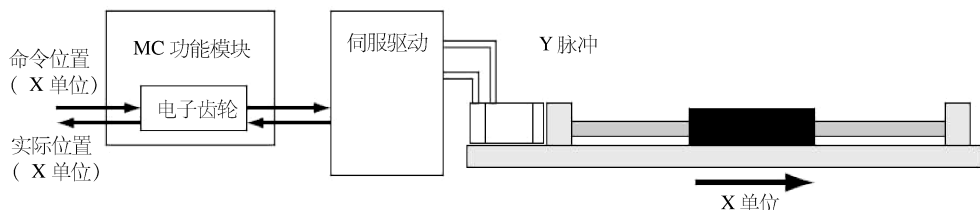


图 7-10 电子齿轮比的作用

命令位置值(脉冲) = 命令位置(X 单位) × 电子齿轮比

电子齿轮比 = $\frac{\text{电动机转一圈的命令脉冲数(Y 脉冲)}}{\text{电动机转一圈的工作移动距离(X 单位)}}$

对于编码器轴，电子齿轮比的分子是编码器转一圈发出的脉冲数，分母是编码器转一圈时轴移动的距离。

如图 7-11 所示，使用了一个带有 17 位绝对式编码器的 OMRON G5 系列伺服电动机。减速器的减速比为 1/5，并且滚珠丝杠螺距为 10mm。显示单位设置为毫米。电动机转一圈的命令脉冲数设置为伺服电动机上编码器的分辨率。

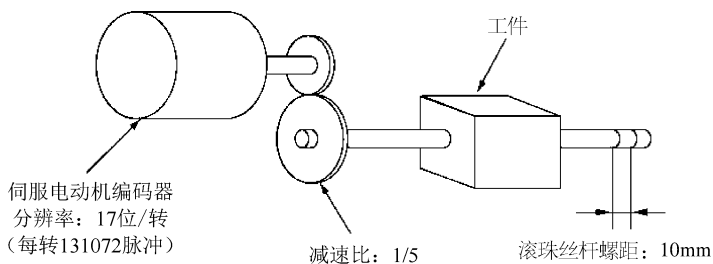


图 7-11 工作示意图

减速比为 1/5，伺服电动机旋转一圈滚珠丝杠旋转 1/5 圈，工件移动 2mm(10mm × 1/5)，所以电动机每转一圈的工件移动距离设置为 2。

参数设置如下：

显示单位：mm

电动机每转一圈的命令脉冲数：131072

电动机每转一圈的工作行程：2

通过这些设置，用户程序里用于定位的命令单位为 1mm。例如，移动一个 100.5mm 的绝对位置，MC_MoveAbsolute (绝对定位) 指令的 Position (目标位置) 输入变量设置为 100.5。

3. 操作设置

操作设置见表 7-7，这些参数设置了最大速度、最大加速度/减速度等用于轴操作的选项。

表 7-7 操 作 设 置

参 数	功 能	设 置 范 围	默 认 值
最大速度	设置每个轴的最大速度* ¹ (单位:命令单位/s)	正长实数	40000000
最大点动速度	设置每个轴的最大点动速度* ² (单位:命令单位/s)	正长实数	1000000
最大加速度	设置每个轴操作命令的最大加速度。如果设置为 0 将对加速度没有限制(单位/s ²)	非负长实数	0
最大减速度	设置为每个轴操作命令的最大减速度。如果设置为 0 将对减速度没有限制(单位/s ²)	非负长实数	0
加速度/减速度超出	轴加速/减速控制时,因为停在目标位置的优先级是特定的,导致过度加速/减速,会出现超出最大加速度/减速度,在上述情况下,设置如下操作: 0:使用快速加速/减速(混合变为缓冲) 1:使用快速加速/减速 2:轻微故障停止	0 ~ 2	0
换向操作选择	为多次执行指令、重新执行指令以及中断指令,设置换向旋转操作: 0:减速停止 1:立即停止	0 ~ 1	0
速度警告值	在输出轴的速度警告时设置最大速度的百分比。如果设置为 0 将没有速度警告输出(单位:%)	0 ~ 100	0
加速度警告值	在输出轴的加速度警告时设置最大加速度的百分比。如果设置为 0 将没有加速度警告输出(单位:%)	0 ~ 100	0
减速度警告值	在输出正扭矩警告时设置最大减速度的百分比。如果设置为 0 将没有减速度警告输出(单位:%)	0 ~ 100	0
正扭矩警告值	在输出正扭矩警告时设置扭矩的设定值。如果设置为 0 将没有正扭矩警告输出(单位:%)	0 ~ 1000	0
负扭矩警告值	在输出负扭矩警告时设置扭矩的设定值。如果设置为 0 将没有负扭矩警告输出(单位:%)	0 ~ 1000	0
实际速度滤波器的时间常数	设置时间周期来计算毫秒内实际速度的平均行程。如果设置为 0,这个平均行程将不会被计算(单位:ms) 用这个来减少当轴速度降低时所引起的实际当前速度的变化	0 ~ 100	0
定位范围	设置定位的宽度(单位:命令单位)	非负长实数	10
定位检查时间	设置在毫秒内的定位检查时间。设置 0 来检查位置的终点,这种情况只能发生在原点返回时定义了原点的位置并且在其他时间不检查位置(单位:ms)	0 ~ 1000	0
零位置范围	设置原点定位检测的宽度(单位:命令单位)	非负长实数	10

* 1 如果指定的命令速度超过最大速度,则使用最大速度。这个参数也用于插补控制操作。

* 2 如果指定的命令速度超过最大点动速度,则使用最大点动速度。

4. 其他操作设置

其他操作设置见表 7-8, 用于输入信号使能时, 设置停止方法和转矩限位。

表 7-8 其他操作设置

参 数	功 能	设 置 范 围	默 认 值
立即停止输入的停止方法	当立即停止输入使能时, 在运动控制模块设置停止方式: 0: 立即停止 2: 立即停止并且错误复位 3: 立即停止并且伺服器 OFF	02 或者 3	0
限位输入的停止方法	当正限位或者负限位使能时, 在运动控制模块设置停止方式: 0: 立即停止 1: 减速停止 2: 立即停止并且错误复位 3: 立即停止并且伺服器 OFF	0 ~ 3	0
驱动错误复位监测时间	为驱动错误复位设置监测时间(单位: ms) 当监测时间过去后, 监测过程将会终止即使驱动器错误没有复位	1 ~ 1000	200
正扭矩限位最大值*	设置正扭矩最大限位(单位: %)	0.0 ~ 1000.0	300.0
负扭矩限位最大值*	设置负扭矩最大限位(单位: %)	0.0 ~ 1000.0	300.0

* 如果正扭矩限位 (60E0H) 和负扭矩限位 (60E1H) 已建立 PDO 映射, 则这些参数的设置值用 EtherCAT 过程数据通信发送。如果 MC_SetTorqueLimit 指令的扭矩限位被使能, 为指令输入变量指定的值将被发送。

5. 限位设置

限位设置见表 7-9。

表 7-9 限 位 设 置

参 数	功 能	设 置 范 围	默 认 值
软件限位*	选择软件限位功能 0: 无效 1: 命令位置的减速停止 2: 命令位置的立即停止 3: 实际位置的减速停止 4: 实际位置的立即停止	0 ~ 4	0
正软件限位	设置正方向的软件限位(单位: 命令单位)	长实数	2147483647
负软件限位	设置负方向的软件限位(单位: 命令单位)	长实数	- 2147483647
位置偏移超出值	设置超出位置偏移的检查值。设置 0 使位置偏移值检查无效(单位: 命令单位)	非负长实数	0
位置偏移警告值	设置位置偏移警告的检查值。设置 0 使位置偏移值警告检查无效(单位: 命令单位)	非负长实数并且小于等于位置偏移超出值	0

* 只有当计数模式设置为线性模式并且已定义原点, 这个功能才能被使能。

6. 位置计数设置

位置计数设置见表 7-10。

表 7-10 位置计数设置

参 数	功 能	设 置 范 围	默 认 值
计数模式	设置位置计数模式： 0:线性模式(有限长度) 1:环形模式(无限长度)	0、1	0
模最大位置设置值	当计数模式设置为环形模式时,设置模的最大位置(单位:命令单位)	长实数	2147483647
模最小位置设置值	当计数模式设置为环形模式时,设置模的最小位置(单位:命令单位)	长实数	-2147483647
编码器类型	设置编码器类型： 0:增量编码器(INC) 1:绝对编码器(ABS)	0、1	0

(1) 计数模式

计数模式是轴的进给模式。为每个轴的命令位置选择计数模式。有两种计数模式：线性模式和环形模式，线性模式有一个有限的轴进给范围，环形模式有一个无限的轴进给范围。

1) 线性模式 (有限长度的轴)：线性模式以 0 为中心。这种模式用于设备有限机械范围的运动，例如 XY 平台。

当值转换为脉冲时，设定范围 40 位，为有符号整数，取值范围为 8000000000H ~ 7FFFFFFFHH。

不能指定超出该范围的绝对或相对定位的目标位置。

没有目标位置的操作，如速度控制、原点返回或扭矩控制，超出该范围，会发生命令位置的上溢或下溢观察。此时，命令位置将继续输出，但实际位置并不更新，被固定在上限或下限。

当实际位置的值固定后，可以执行命令，并且用任何指向线性范围方向上没有目标位置的操作停止轴。任何指定远离该范围方向的命令在执行时都会出现错误。

上溢或下溢状态被清除后，实际位置才更新。

2) 环形模式 (无限长度的轴)。

如图 7-12 所示，为环形模式。

该模式使用一个环形计数器在设定范围内进行无限数量的重复计数。旋转表或卷绕轴使用该模式。

使用 Sysmac Studio 设置模的最大位置和模的最小位置，定义环形计数器的范围。

当值转换为脉冲时，设定范围 40 位，为有符号整数，取值范围为 8000000000H ~ 7FFFFFFFHH。

(2) 模最大位置和模最小位置的设置值

当计数模式设置为环形模式时，这些参数的设置使能。设置环形计数器的上限和下限。

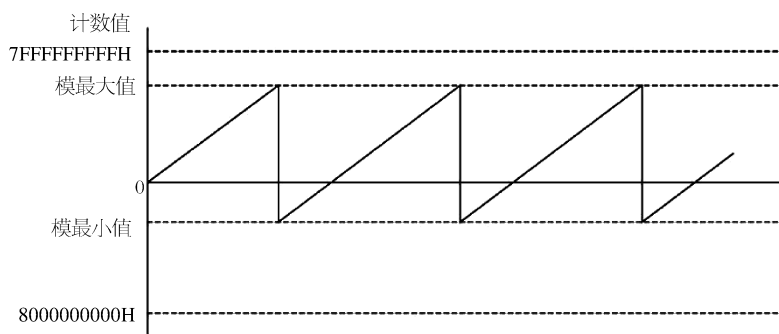


图 7-12 环形模式

如果设置计数模式为环形模式，确保环形计数器一个周期的值可转换为整数个脉冲。如果环形计数器一个周期的脉冲数不为整数，因为小数部分被截断，位置发生偏移。该命令的当前位置也将无法正确显示。

如果 0 不包括在环形计数器的上限和下限之间，MC_MoveZeroPosition (高速原点返回) 指令执行时会发生错误。

当使用 MC_MoveAbsolute 或 MC_Move 指令执行绝对定位时，确保目标位置在环形计数器的范围之内。如果目标位置不在环形计数器的范围之内，则将发生错误。如果指令的 Direction 输入变量设置为“无指定方向”，可以设置不在环形计数器范围之内的目标位置。如果这样，则使用目标位置和命令当前位置的差作为目标距离进行相对定位。

(3) 编码器类型

对于反馈输入设置编码器类型，虚轴禁用这个参数的设置。

7. 伺服驱动设置

伺服驱动设置见表 7-11。

表 7-11 伺服驱动设置

参 数	功 能	设置范围	默 认 值
模最大位置设置值	在伺服驱动器上设置模最大位置	$-2^{63} \sim 2^{63} - 1$	2147483647
模最小位置设置值	在伺服驱动器上设置模最小位置	$-2^{63} \sim 2^{63} - 1$	-2147483648

8. 原点返回设置

原点返回参数见表 7-12，用于伺服电动机确定原点的操作。

表 7-12 原点返回参数

参 数 名 称	说 明	设 定 值	默 认 值
原点返回方法	设置原点返回操作： 0:接近反转/原点接近输入 OFF 1:接近反转/原点接近输入 ON 4:原点接近输入 OFF 5:原点接近输入 ON 8:限位输入 OFF 9:接近反转/原点输入掩码距离 11:仅限位输入 12:接近反转/保持时间 13:无原点接近输入/保持原点输入 14:零位置预设	0,1,4,5,8,9, 或 11~14	14

(续)

参 数 名 称	说 明	设 定 值	默 认 值
原点输入信号	选择输入作为原点输入信号: 0:使用 Z 相输入作为原点 1:使用外部原点输入* ¹	0 或 1	0
原点返回启动方向	设置原点返回启动时的启动方向: 0:正方向 2:负方向	0 或 2	0
原点输入检测方向	设置原点返回后的原点输入检测方向: 0:正方向 2:负方向	0 或 2	0
正限位输入时操作选项	原点返回过程中,正向限位输入变为 ON 时,设置停止方式: 0:无反转/轻微故障停止 1:反转/立即停止 2:反转/减速停止	0 ~ 2	1
负限位输入时操作选项	原点返回过程中,负向限位输入变为 ON 时,设置停止方式: 0:无反转/轻微故障停止 1:反转/立即停止 2:反转/减速停止	0 ~ 2	1
原点返回速度	设置原点返回速度(单位:命令单位/s)	正长实数	10000
原点返回接近速度	设置原点接近输入变为 ON 后的速度(单位:命令单位/s)	正长实数	1000
原点返回加速度	设置原点返回加速度。无加速时设为 0(单位:命令单位/s ²)	非负长实数	0
原点返回减速度	设置原点返回减速度。无减速时设为 0(单位:命令单位/s ²)	非负长实数	0
原点返回加加速度	设置原点返回加加速度。无加加速时设为 0(单位:命令单位/s ³)	非负长实数	0
原点输入掩码距离	当设置原点操作模式为接近反转和原点输入掩码距离时,设置原点输入掩码距离(单位:命令单位)	非负长实数	10000
原点偏移量	为原点返回后的设置值预置实际位置	长实数	0
原点返回保持时间	当设置原点操作模式时为接近反向和保持时间时,设定保持时间	0 ~ 10000	100
原点返回补偿值	设置原点定义后的原点返回补偿值(单位:命令单位)	长实数	0
原点返回补偿速度	设置原点返回的补偿速度(单位:命令单位/s)	正长实数	1000

* 1 此设置用于内置 EtherCAT 通信的欧姆龙 G5 系列伺服驱动器。在 OMRON G5 系列伺服驱动器的默认设置中,外部原点输入被分配给闩锁 1。闩锁 1 的分配可以用伺服驱动里的伺服参数对象改变。

7.3.3 轴参数设置举例

1. 单轴定位

如图 7-13 所示，两个轴各自独立完成单轴定位。

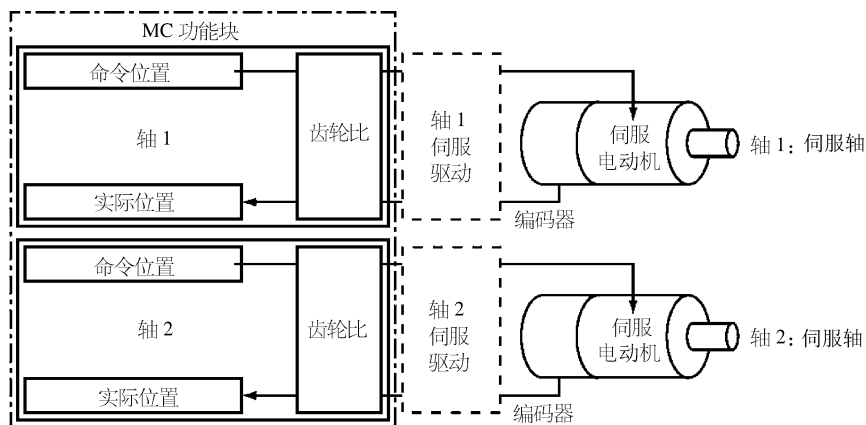


图 7-13 单轴定位

单轴定位参数设置见表 7-13。

表 7-13 单轴定位参数设置

参数名称	设 置		参数名称	设 置	
	轴 1	轴 2		轴 1	轴 2
轴变量名称	轴 1	轴 2	最大速度	$500,000^{*2}$	$500,000^{*2}$
轴号	1	2	最大点动速度	$50,000^{*3}$	$50,000^{*3}$
轴使能	使用	使用	最大加速度	$5,000,000^{*4}$	$5,000,000^{*4}$
轴类型	伺服轴	伺服轴	最大减速度	$5,000,000^{*4}$	$5,000,000^{*4}$
节点地址	1	2	软件限位	命令位置立即停止	命令位置立即停止
显示单位	μm^{*1}	μm	正软件限位	$500,000^{*5}$	500,000
每转的命令脉冲数	1,048,576	1,048,576	负软件限位	0^{*5}	0
每转工件移动距离	10,000	10,000	计数模式	线性模式	线性模式

注意：

- 1) 位置命令单位为 $1\mu\text{m}$ 。
- 2) 最大速度为 $3000\text{r/min} = 30\text{m/min} = 0.5\text{m/s} = 500000\mu\text{m/s}$ 。
- 3) 最大点动速度是最大速度的 10%，即 $0.05\text{m/s} = 50000\mu\text{m/s}$ 。
- 4) 最大加速度和最大减速度是 5m/s^2 。加速到最大速度 (3000r/min) 的时间是 0.1s。
- 5) 在设备可移动范围内设置定位，正软件限位设置为 $50\text{cm} = 500000\mu\text{m}$ 。

2. 编码器作主轴的同步控制

编码器作主轴的同步控制如图 7-14 所示。

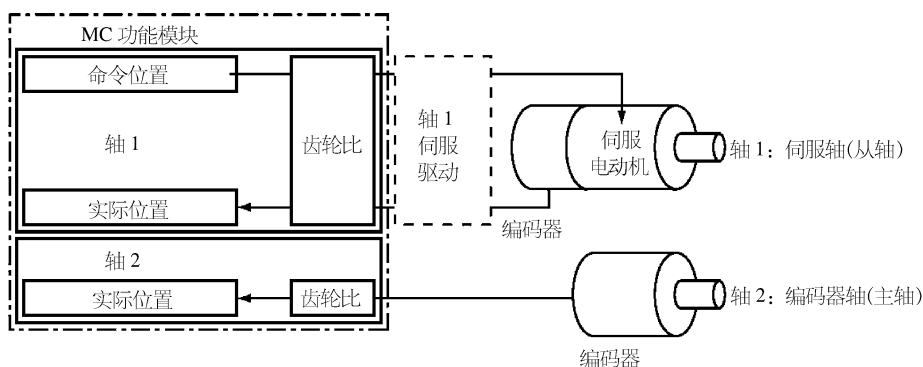


图 7-14 编码器作主轴的同步控制

同步控制参数设置见表 7-14。

表 7-14 同步控制参数设置

参数名称	设 置		参数名称	设 置	
	轴 1	轴 2		轴 1	轴 2
轴变量名称	轴 1	轴 2	最大点动速度	500,000 ^{* 3}	—
轴号	1	2	最大加速度	5,000,000 ^{* 4}	—
轴使能	使用	使用	最大减速度	5,000,000 ^{* 4}	—
轴类型	伺服轴	编码器轴	软件限位	命令位置立即停止	禁用
节点地址	1	2	正软件限位	5,000,000 ^{* 5}	—
显示单位	$\mu\text{m}^* 1$	$\mu\text{m}^* 1$	负软件限位	0 ^{* 5}	—
每转的命令脉冲数	1,048,576	1,048,576	计数模式	线性模式	环形模式
每转工件移动距离	10,000	10,000	模最大位置	—	1,000,000 ^{* 6}
最大速度	500,000 ^{* 2}	—	模最小位置	—	0 ^{* 6}

注意：

- 1) 位置命令单位为 $1\mu\text{m}$ 。
- 2) 最大速度为 $3000\text{r}/\text{min} = 30\text{m}/\text{min} = 0.5\text{m}/\text{s} = 500000\mu\text{m}/\text{s}$ 。
- 3) 最大点动速度是最大速度的 10%，即 $0.05\text{m}/\text{s} = 50000\mu\text{m}/\text{s}$ 。
- 4) 最大加速度和最大减速度是 $5\text{m}/\text{s}^2$ 。加速到最大速度 (3000r/min) 的时间是 0.1s。
- 5) 在设备可移动范围内设置定位，正软件限位设置为 $50\text{cm} = 500000\mu\text{m}$ 。
- 6) 位置的周期范围是 $0 \sim 1\text{m}$ ($1000000\mu\text{m}$)。

7.3.4 轴变量

MC 功能模块控制轴，轴变量是系统定义变量，用于表示轴参数和监视信息，例如，实际位置和错误信息。用 Sysmac Studio 创建轴时，轴变量按照轴生成的顺序登记在变量表中。轴变量是结构体，数据类型为 `_sAXIS_REF`。

1. 轴变量名称

如表 7-15 所示，MC 功能模块中的每一个轴变量都有两个变量名：系统定义变量的轴变

量名称和 Sysmac Studio 添加轴时分配的名称。

系统定义变量的轴变量名称为_MC_AX[0]~_MC_AX[63]。

在 Sysmac Studio 中添加轴时, MC_Axis000 ~ MC_Axis063 默认分配给_MC_AX[0]~_MC_AX[63], 序号按照轴添加的顺序分配。需要时, 可以在 Sysmac Studio 中改变这些轴的名称。

表 7-15 轴 变 量

系统定义变量的轴变量名称	Sysmac Studio 添加轴时默认的轴变量名称	轴 号 例 子
_MC_AX[0]	MC_Axis000	轴 0
_MC_AX[1]	MC_Axis001	轴 1
⋮	⋮	⋮
_MC_AX[63]	MC_Axis063	轴 63

在用户程序中指定轴变量时, 可以使用系统定义变量的轴变量名称或 Sysmac Studio 添加轴时分配的名称。

2. 轴变量结构

轴变量结构如下所示。

_MC_AX[0]	轴变量
_MC_A[0].Status	轴状态层级
_MC_AX[0].Status.Ready	轴准备执行
_MC_AX[0].Status.Disabled	轴禁用
⋮	
_MC_AX[0].Details	轴控制状态层级
_MC_AX[0].Details.Idle	轴空闲状态
_MC_AX[0].Details.InPosWaiting	定位等待
⋮	
_MC_AX[0].Cmd	轴命令层级
_MC_AX[0].Cmd.Pos	命令当前位置
_MC_AX[0].Cmd.Vel	命令当前速度
_MC_AX[0].Cmd.AccDec	命令当前加速度/减速度
⋮	
_MC_AX[0].Act	轴当前值层级
_MC_AX[0].Act.Pos	当前实际位置
_MC_AX[0].Act.Vel	当前实际速度
_MC_AX[0].Act.Trq	当前实际扭矩
⋮	
_MC_AX[0].Cfg	轴的基本设置层级
_MC_AX[0].Cfg.AxNo	轴号
_MC_AX[0].Cfg.AxEnable	轴使能
_MC_AX[0].Cfg.AxType	轴类型
⋮	

<code>_MC_AX[0].Scale</code>	显示单元层级
<code>_MC_AX[0].Scale.Units</code>	显示单元
<code>_MC_AX[1]</code>	轴变量
<code>⋮</code>	

如果 `_MC_Axis000` 改为 `MyAxis1`，那么，`MyAxis1.Act.Pos` 或者 `_MC_AX[0].Act.Pos` 都可以用作表示实际当前位置的变量。

3. 在用户程序中指定轴

如图 7-15 所示，在用户程序中的运动控制指令，为输入-输出变量 `Axis` 指定一个轴变量名。在 Sysmac Studio 中，把系统定义的轴变量名称 `_MC_AX[0]` 改为 `MyAxis1`。

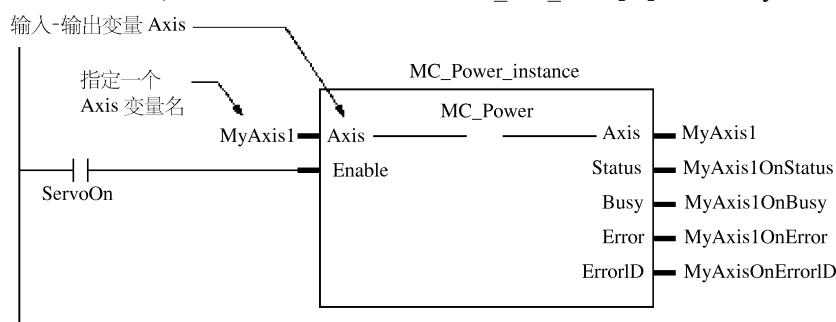


图 7-15 指定轴

在 `MyAxis1` 位置，也可以用系统定义变量 `_MC_AX[0]`。

7.3.5 从 Sysmac Studio 进行轴参数设置

用 Sysmac Studio 进行轴参数设置的步骤如下。

- 1) 创建一个工程；
- 2) 生成 EtherCAT 网络配置；
- 3) 添加轴；
- 4) 分配轴；
- 5) 设置轴参数；
- 6) 在线并同步；
- 7) 将工程传送到控制器。

轴设置的详细操作过程参见 6.3 节。

7.4 轴组

7.4.1 轴组简介

轴组由多个轴组成，用来完成多轴的复杂操作，例如直线插补或圆弧插补。使用 Sysmac Studio 设置轴组变量，来使能轴组运动控制指令的执行，或使能访问轴组状态。MC 功能模块可以处理 32 个轴组，每个轴组最多 4 个轴。

使用轴组参数设置与轴组相关的操作，这些操作由 MC 功能模块控制，例如，轴组配置、最大插补速度、轴组停止方法。NJ501-1300、NJ501-1400 或 NJ501-1500 都有 32 组轴组参数。相同的参数设置提供给每个轴组。

MC 功能模块的轴组元素见表 7-16。

表 7-16 MC 功能模块的轴组元素

配置元素	说 明
轴组参数	轴组参数设置最大插补速度、最大插补加速度/减速度以及 MC 功能模块控制的轴组其他操作的设置。使用 Sysmac Studio 来设置轴组参数
轴组变量	轴组变量是系统定义变量,它包括一部分轴组参数,以及命令插补速度、错误信息和 MC 功能模块控制的轴组的其他监视信息。当从 Sysmac Studio 的多视图浏览器中添加一个轴组时,轴组变量就会被创建。轴组变量的名称在这里设置
用户程序中指定轴组	在用户程序中,运动控制由运动控制指令来实现。多轴协调控制的运动控制指令用来创建轴组命令。用轴组命令控制一个轴组时,对指令的 AxesGroup 轴组输入-输出变量,使用系统定义的变量或者 Sysmac Studio 设置的轴组变量名称

7.4.2 轴组参数

轴组参数见表 7-17。

表 7-17 轴 组 参 数

分 类	参 数 名 称	临时改变* 1	读变量* 2	分 类	参 数 名 称	临时改变* 1	读变量* 2
轴组基本 设置	轴组号	—	OK	轴组操作 设置	插补加速度/减速度 超限	—	—
	轴组使用	—	OK		插补速度警告值	OK	—
	构成	—	OK		插补加速度警告值	OK	—
	构成轴	—	OK		插补减速度警告值	OK	—
轴组操作 设置	最大插补速度	—	—		轴组停止方法	—	—
	最大插补加速度	—	—		修正允许比率	—	—
	最大插补减速度	—	—				

* 1 表示是否能用 MC_Write 指令临时改变参数。
* 2 该列表示是否能在用户程序里用变量访问这些参数。

轴组参数分类说明如下。

1. 轴组基本设置

轴组基本设置见表 7-18，设置是否使用轴组。如果要使用，设置要使用的轴和轴组配置。

表 7-18 轴组基本设置

参数名称	功 能	设置范围	默 认 值
轴组使用	设置是否使能或者禁用轴组,如果对一个未定义的或者未使用的轴组执行运动控制指令,会出现错误; 0:未定义的轴组 1:未使用的轴组 2:使用的轴组	0~2	0

(续)

参 数 名 称	功 能	设 置 范 围	默 认 值
构成	设置轴组的轴构成： 0:2 轴 1:3 轴 2:4 轴	0 ~ 2	0
构成轴	设置分配给轴组的轴序号。从 Sysmac Studio 为 A0 ~ A3 轴设置轴变量的名称	2 ~ 4	0

(1) 构成

表 7-19 列出了 MC 功能模块可以使用的轴构成，默认轴组禁用。根据实际设备，使用 Sysmac Studio 设置轴构成。

表 7-19 构 成

构 成	说 明
2 轴	使用两轴的配置。例如,一个机器使用两轴 Cartesian 坐标系
3 轴	使用三轴的配置。例如,一个机器使用三轴 Cartesian 坐标系
4 轴	使用四轴的配置。例如,一个机器使用三轴 Cartesian 坐标系,末端工具使用一个旋转轴

(2) 构成轴

轴组中的轴称为构成轴。对于轴组指令，为了更容易重复使用插补指令的编程，用逻辑轴 (轴 A0 ~ 轴 A3) 代替轴号 (轴 0 ~ 轴 63)。对于构成轴的参数，设置轴组中的各个轴的轴号和逻辑轴号。伺服轴或虚拟伺服轴可以选作逻辑轴。

构成轴设置见表 7-20，使用 Sysmac Studio，从逻辑轴 A0 开始，为轴组里选择的轴号分配逻辑轴。

如果创建了不少一个轴组，从逻辑轴 A0 开始为各个轴组设置轴号。在多个轴组可以设置相同的轴号。

表 7-20 构成轴设置

构 成 轴	构成轴参数设置
2 轴	为轴 A0 和轴 A1 设置轴变量名称(轴号)
3 轴	为轴 A0、轴 A1 和轴 A2 设置轴变量名称(轴号)
4 轴	为轴 A0、轴 A1、轴 A2 和轴 A3 设置轴变量名称(轴号)

构成轴设置举例 I 见表 7-21，轴组分配 4 个轴，轴号为 1、2、5 和 8。

表 7-21 构成轴设置举例 I

逻 辑 轴	轴 号	逻 辑 轴	轴 号
轴 A0	轴 1	轴 A2	轴 5
轴 A1	轴 2	轴 A3	轴 8

构成轴设置举例 II 见表 7-22，轴组分配 3 个轴，轴号为 1、8 和 2。

表 7-22 构成轴设置举例 II

逻辑轴	轴号	逻辑轴	轴号
轴 A0	轴 1	轴 A2	轴 2
轴 A1	轴 8	轴 A3	None

2. 轴组操作设置

轴组操作设置见表 7-23，设置最大插补速度和轴组停止方法等轴组操作时用的选项。

表 7-23 轴组操作设置

参数名称	功能	设置范围	默认值
最大插补速度	为路径设置最大插补速度,设置 0 表示没有插补速度限制。如果目标速度超出了为轴组操作指令指定的最大插补速度,轴将以最大插补速度运行(单位:命令单位/s)	非负长实数	800000000
最大插补加速度	为路径设置最大插补加速度,设置 0 表示没有插补加速度限制(单位:命令单位/s ²)	非负长实数	0
最大插补减速度	为路径设置最大插补减速度,设置 0 表示没有插补减速度限制(单位:命令单位/s ²)	非负长实数	0
插补加速度/减速度超限	当停止在目标位置的优先权给出后,轴组的加速/减速运动因为过度加速/减速,可能超出最大的加速度/减速度,设置操作方法: 0:使用快速加速/减速(Blending 变为 Buffered) 1:使用快速加速/减速 2:轻微故障停止	0 ~ 2	0
插补速度警告值	设置最大插补速度的百分比,作为输出插补速度警告值。如果设置为 0 将没有插补速度警告输出(单位:%)	0 ~ 100	0
插补加速度警告值	设置最大插补加速度的百分比,作为输出插补加速度警告值。如果设置为 0 将没有插补加速度警告输出(单位:%)	0 ~ 100	0
插补减速度警告值	设置最大插补减速度的百分比,作为输出插补减速度警告值。如果设置为 0 将没有插补减速度警告输出(单位:%)	0 ~ 100	0
轴组停止方法	处于多轴协调运动中的轴出现错误迫使其立即停止时,为没有 出现错误的构成轴设置停止方法: 0:立即停止 1:各轴以最大减速度减速停止 3:立即停止并且伺服 OFF	0、1 或 3	0
修正允许比率	圆弧插补指令使用中心指定方法时用这个参数。当起始点与中心点的距离不等于终点与中心点的距离时,它能够补偿距离。修正值作为半径的百分比,要设置合理的范围。设置百分比为 0.1% 或者更大。如果设置为 0,不进行错误检查	0 ~ 100 之间单精度的浮点数	0

7.4.3 轴组变量

轴组变量是系统定义变量,用于对 MC 功能模块控制的轴组设置信息和监视信息,例如,实际位置和错误信息。用 Sysmac Studio 生成轴组时,轴组变量按照轴组生成的顺序登记在变量表中。轴组变量是结构体,数据类型为_sGROUP_REF。

1. 轴组变量名称

MC 功能模块中的每一个轴组变量都有两个变量名：系统定义变量的轴组变量名称和 Sysmac Studio 增加轴组时分配的名称。

系统定义变量的轴组变量名称为 `_MC_GRP[0] ~ _MC_GRP[31]`。

在 Sysmac Studio 中增加轴组时，`MC_Group000 ~ MC_Group031` 默认分配给 `_MC_GRP[0] ~ _MC_GRP[31]`，序号按照轴增加的顺序分配。需要时，可以在 Sysmac Studio 中改变这些轴组的名称。

在用户程序中指定轴组变量时，可以使用系统定义变量中的轴组变量名称或 Sysmac Studio 添加轴组时分配的名称。

轴组变量的名称见表 7-24。

表 7-24 轴组变量的名称

系统定义变量的轴组变量名称	Sysmac Studio 增加轴组时分配的名称	轴组号例子
<code>_MC_GRP[0]</code>	<code>MC_Group000</code>	轴组 0
<code>_MC_GRP[1]</code>	<code>MC_Group000</code>	轴组 1
<code>⋮</code>	<code>⋮</code>	<code>⋮</code>
<code>_MC_GRP[31]</code>	<code>MC_Group031</code>	轴组 31

2. 轴组变量的结构

轴组变量结构如下所示。

<code>_MC_GRP[0]</code>	轴组变量
<code>_MC_GRP[0].Satus</code>	轴组状态层级
<code>⋮</code>	
<code>_MC_GRP[0].Cmd</code>	轴组命令值层级
<code>_MC_GRP[0].Cmd.Vel</code>	命令插补速度
<code>_MC_GRP[0].Cmd.AccDec</code>	命令插补加速度/减速度
<code>⋮</code>	
<code>_MC_GRP[0].Cfg</code>	轴组基本设置层级
<code>_MC_GRP[0].Cfg.GrNo</code>	轴组号
<code>_MC_GRP[0].Cfg.GrEnable</code>	轴组是否使能
<code>_MC_GRP[0].Kinematics</code>	运动学转换设置层级
<code>_MC_GRP[0].Kinematics.GrType</code>	轴构成
<code>_MC_GRP[0].Kinematics.Axis[0]</code>	轴 A0 对应的构成轴
<code>⋮</code>	
<code>_MC_GRP[0].Kinematics.Axis[3]</code>	轴 A3 对应的构成轴
<code>_MC_GRP[1]</code>	轴组变量
<code>⋮</code>	

如果 `_MC_GRP[0]` 改为 `MyGroup1`，那么，`MyGroup1.Cmd.Vel` 或者 `_MC_GRP[0].Cmd.Vel` 都可以用作表示命令插补速度的变量。

3. 在用户程序中指定轴组

如图 7-16 所示，在用户程序的运动控制指令中，为输入-输出变量 `AxesGroup` 指定一个

轴组变量名。在 Sysmac Studio 中, 把系统定义的轴组变量名称 `_MC_GRP[0]` 改为 `MyGroup1`。

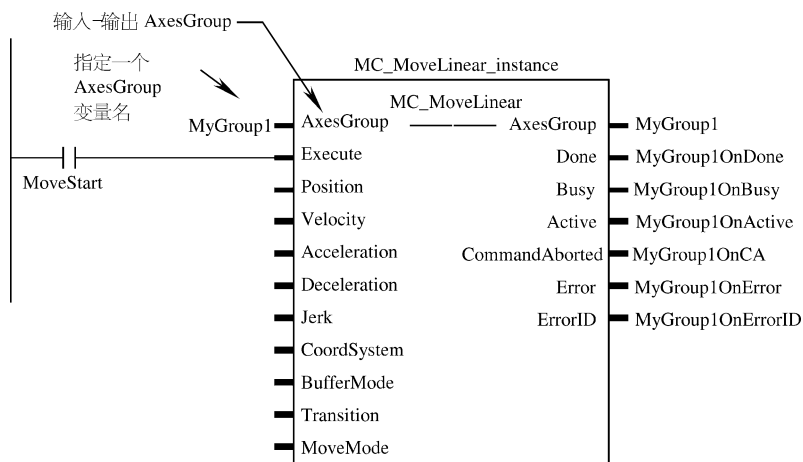


图 7-16 指定轴组变量

在 `MyGroup1` 位置, 也可以用系统定义变量 `_MC_GRP[0]`。

7.4.4 从 Sysmac Studio 进行轴组参数设置

用 Sysmac Studio 进行轴组参数设置的步骤如下。

1. 启动 Sysmac Studio 并打开一个工程
2. 添加一个轴组

如图 7-17 所示, 右击多视图浏览器的“配置和设置-轴组设置”, 从“添加”菜单中选择“轴组设置”, 添加一个轴组, 新的轴组默认名称为 `MC_Group000`, 如图 7-18 所示。

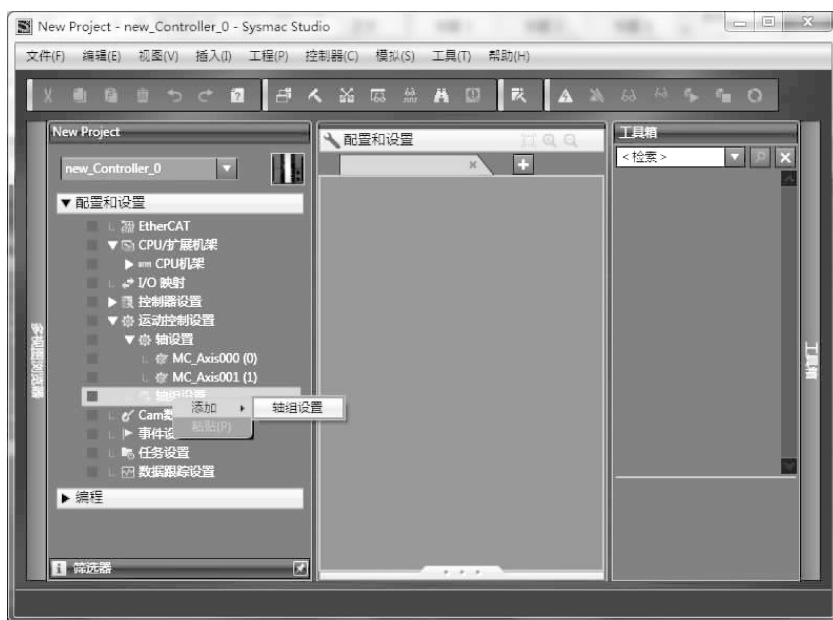


图 7-17 添加轴组



图 7-18 添加了轴组 MC_Group0000

也可以通过复制来生成轴组。

3. 设置轴组参数

1) 右击多视图浏览器中的一个轴组，从菜单中选择“编辑”。
轴组参数设置标签页显示轴组基本参数设置，如图 7-19 所示。



图 7-19 轴组基本参数设置

2) 在“轴组使用”对话框选择“使用轴组”，如图 7-20 所示。



图 7-20 使用轴组

3) 在“构成”对话框选择轴组的构成，图 7-21 中为 2 轴的构成。

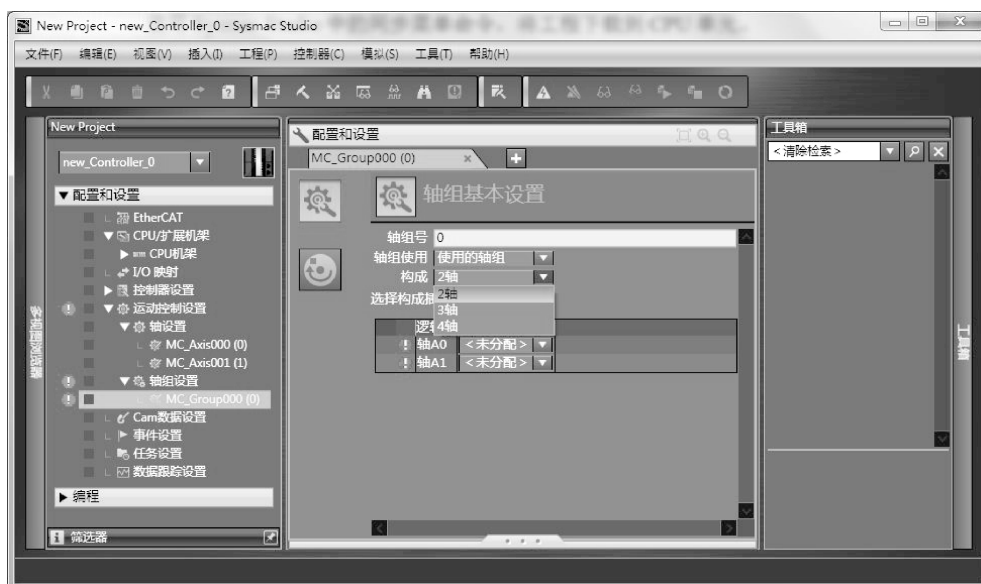


图 7-21 2 轴的构成

4) 在“逻辑轴”对话框中分配使用的轴，如图 7-22 所示。

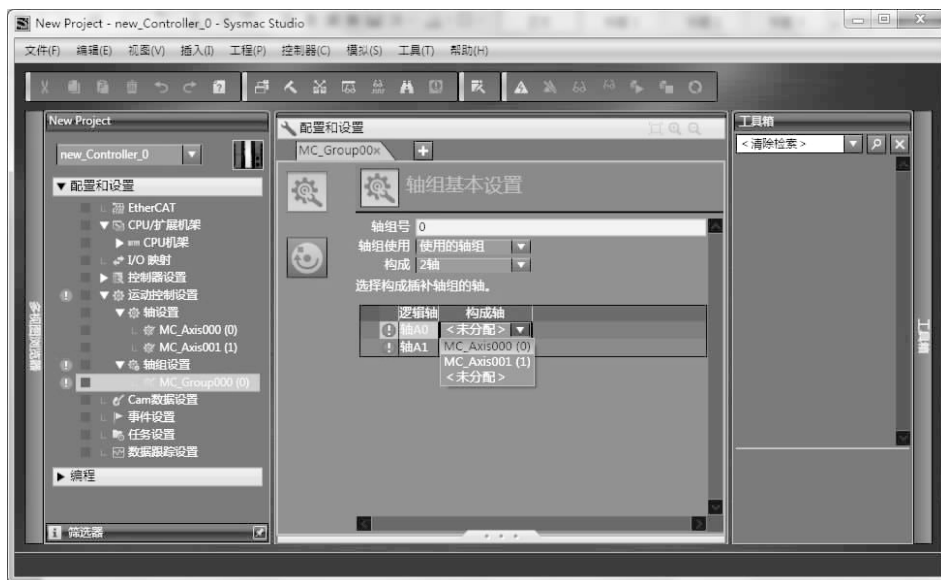


图 7-22 分配使用的轴

5) 点击底部的图标，显示轴组操作设置标签页，如图 7-23 所示，基于设备操作条件设置合适的值。



图 7-23 轴组操作设置

改变轴组变量名称时，既要在变量声明的变量表中改变名称，也要在用户程序中改变名称，即两个位置同时改变名称。只在变量表中改变了轴组变量名称，用户程序中的轴组变量名称不会自动改变。如果使用的变量名称在变量表中没有声明，用户程序会出现错误。

4. 下载到 CPU 单元

轴组设置完毕，使用 Sysmac Studio 中的“控制器”菜单命令，将工程下载到 CPU 单元。

- 1) 从控制器菜单选择“在线”，Sysmac Studio 与控制器在线连接。
- 2) 从控制器菜单选择“同步”，然后点击“传送到控制器”按钮。

7.5 MC 试运行

可以从 Sysmac Studio 使用 MC 试运行功能来检查系统连接，如监视传感器信号，检查伺服电动机接线等，这些操作不需要编程。

MC 试运行功能用于内置 EtherCAT 通信的 OMRON G5 系列伺服驱动，不要用于其他制造商的伺服驱动。

7.5.1 MC 试运行功能

MC 试运行功能见表 7-25。

表 7-25 MC 试运行功能

分 类	功 能	说 明	设置/监视选项
轴操作	减速停止	在 MC 试运行期间执行减速停止	—
	伺服 ON/OFF	使伺服变为 ON 或 OFF	—
	复位错误	复位 MC 功能模块中的错误	—
	点动	在正或负方向执行点动	目标速度 加速度/减速度
	绝对定位	执行绝对定位*	目标位置 目标速度 加速度/减速度 加加速度
	相对定位	执行相对定位	移动距离 目标速度 加速度/减速度 加加速度
	原点返回	使用原点返回参数设置,执行原点返回	原点返回参数
监视	错误列表	监视 MC 功能模块中的错误	MC 常见错误 轴错误 轴组错误
	轴状态	监视轴状态	轴准备执行 待机 离散运行 连续运行 原点返回 停止 原点定义 在原点位置
	实际位置监视	监视实际位置	命令和实际当前位置
	实际速度监视	监视实际速度	命令和实际当前速度

(续)

分 类	功 能	说 明	设置/监视选项
监视	伺服驱动状态	监视伺服驱动的状态	伺服 ON/OFF 伺服准备就绪 主电源
	输入信号	监视输入信号的状态	正限位输入 负限位输入 立即停止输入 原点接近输入 原点输入 外部门锁输入 1~2

* 当轴的计数模式设置为旋转模式时，沿正方向向目标位置定位。

7.5.2 启动 MC 试运行

单轴设备如图 7-24 所示。

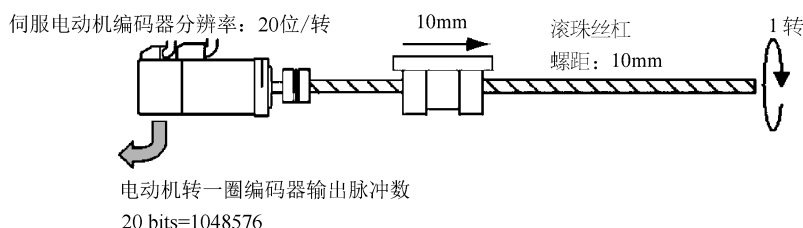


图 7-24 单轴设备

轴参数设置见表 7-26。

表 7-26 轴参数设置

参 数 名 称	设 置	参 数 名 称	设 置
轴变量名称	Axis1* ¹	最大速度	500000* ⁵
轴号	1* ²	最大点动速度	50000* ⁶
轴使用	使用	最大加速度	5000000* ⁷
轴类型	伺服轴	最大减速度	5000000* ⁷
节点地址(输入设备)	1* ³	软件限位	立即停止的命令位置
显示单位	μm	正软件限位	500000* ⁸
每转的命令脉冲数	1048576* ⁴	负软件限位	0* ⁸
每转工作行程	10000* ⁴	计数模式	线性模式

注意：

- 1) 如果有多个轴，每个轴要设置不同的变量名。
- 2) 如果有多个轴，每个轴要设置不同的轴号。
- 3) 为伺服驱动设置同样的节点地址。如果有多个轴，每个轴要设置不同的地址。
- 4) 位置命令单位是 1μm。
- 5) 最大速度是 $3000\text{r/min} = 0.5\text{m/s} = 500000\mu\text{m/s}$ 。
- 6) 最大的点动速度是最大速度的 10%，也就是， $0.05\text{m/s} = 50000\mu\text{m/s}$ 。
- 7) 最大加速度和最大减速度是 5m/s^2 。到达最大速度 (3000r/min) 的加速时间为 0.1s。

8) 设置的正软件限位、负软件限位值,应在设备的可移动范围之内。正软件限位设置为 $50\text{cm} = 500000\mu\text{m}$ 。

操作步骤如下:

- 1) 启动 Sysmac Studio, 打开一个工程, 在工程里完成轴设置。
- 2) 从“控制器”菜单选择“在线”, Sysmac Studio 与控制器在线连接。
- 3) 如图 7-25 所示, 从“控制器”菜单选择“MC 试运行”。



图 7-25 MC 试运行

当图 7-26 所示的注意事项对话框出现后,仔细阅读,确认安全后,点击“确定”按钮,出现图 7-27 所示的 MC 试运行标签页。



图 7-26 注意事项

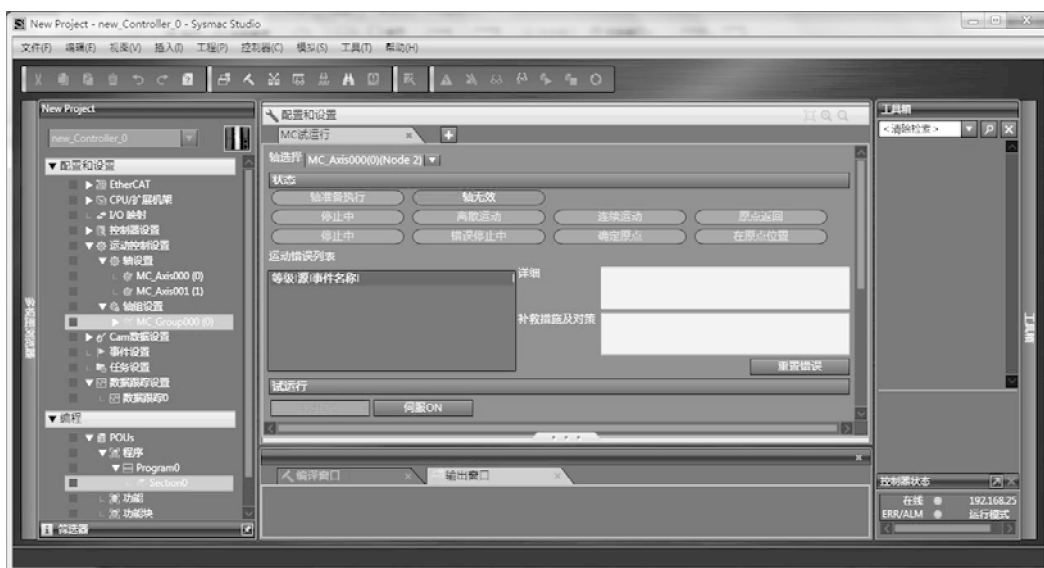


图 7-27 MC 试运行标签页

MC 试运行标签页的中部显示轴状态，如图 7-28 所示。



图 7-28 轴状态

7.5.3 监视传感器信号

可以使用输入信号显示检查传感器接线，如图 7-29 所示。

- 1) 在 MC 试运行标签页选择要检查的轴。
- 2) 通过将连接到输入信号的传感器变 ON 或 OFF，在监视屏幕，检查确认信号是否正确 ON 或 OFF。

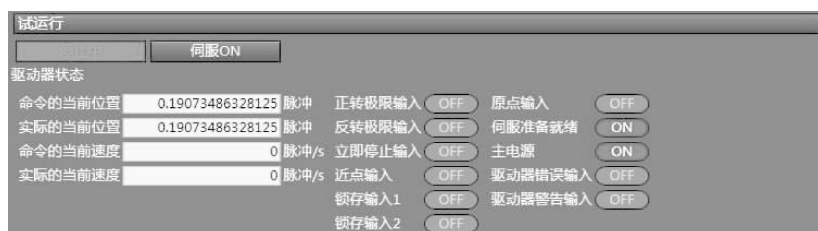


图 7-29 监视传感器信号

7.5.4 检查伺服电动机 ON/OFF 操作

使用 MC 试运行功能检查电动机 ON/OFF 操作。

- 1) 选择使伺服 ON 的轴。
- 2) 点击“伺服 ON”按钮使伺服 ON。
- 3) 点击“伺服 OFF”按钮使伺服 OFF。

正确使用注意事项：

1) 当以下一个来自 Sysmac Studio 的命令的操作执行，伺服电动机将以设定的速度运行：伺服 ON、点动、绝对定位、相对定位或原点返回。在执行这些操作之前务必确定伺服电动机的操作是安全的。

2) 当从 Sysmac Studio 操作控制器，应安装外部应急电路确保必要时伺服电动机可以安全停止，Sysmac Studio 在某些情况下可能无法发送命令，例如：如果计算机发生错误。

3) 在尝试从 Sysmac Studio 执行操作之前，设置 EtherCAT 通信并建立通信。

7.5.5 点动

- 1) 在 MC 试运行标签页的点动选项卡，选择进行点动的轴。
- 2) 点击“伺服 ON”按钮使伺服 ON。
- 3) 如图 7-30 所示，输入目标速度、加速度和减速度，然后按“应用”按钮。
- 4) 点击或按钮，点击其中一个按钮后，伺服电动机会在正方向或负方向运行。检查电动机是否在设定的方向运行。



图 7-30 点动

7.5.6 原点返回

- 1) 在轴参数设置标签页的原点返回设置中，设置原点返回的参数。
- 2) 在 MC 试运行标签页，点击“原点返回”选项。
- 3) 选择原点返回的轴。
- 4) 点击“伺服 ON”按钮使伺服 ON。
- 5) 点击“应用原点返回参数”按钮，如图 7-31 所示。

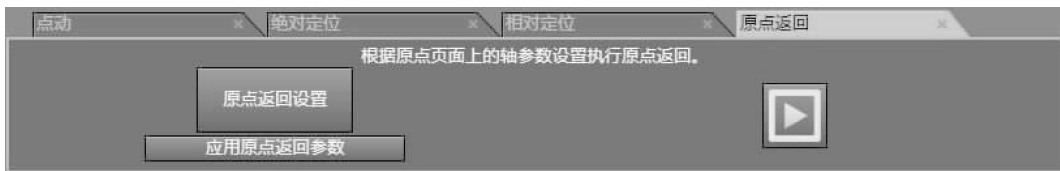


图 7-31 原点返回

6) 点击  按钮。

检查原点返回操作是否与设置匹配。

7.5.7 绝对定位

1) 在 MC 试运行标签页，点击“绝对定位”选项。

2) 选择绝对定位的轴。

3) 点击“伺服 ON”按钮使伺服 ON。

4) 如图 7-32 所示，输入目标位置、目标速度、加速度、减速度和加加速度，然后按“应用”按钮。

5) 单击  按钮，开始绝对定位。

检查定位与设置是否匹配。



图 7-32 绝对定位

7.5.8 相对定位

1) 在 MC 试运行标签页，点击“相对定位”选项。

2) 选择相对定位的轴。

3) 点击“伺服 ON”按钮，使伺服 ON。

4) 如图 7-33 所示，输入目标位置、目标速度、加速度、减速度和加加速度，然后按“应用”按钮。

5) 单击  按钮，开始相对定位。

检查定位与设置是否匹配。



图 7-33 相对定位

7.6 运动控制编程基础

NJ 系列 CPU 单元既可进行顺序控制也可进行运动控制，用户程序中的运动控制指令完成对 EtherCAT 从站的伺服驱动和其他设备的运动控制。

通常, 要为 EtherCAT 从站的伺服驱动分配轴变量。如果不分配轴变量, 要像对一般的从站一样分配设备变量。运动控制指令用在主周期任务和优先级 16 的周期任务中。

7.6.1 运动控制指令

运动控制指令定义为功能块 (FB)。MC 功能块的运动控制指令基于 PLCopen 运动控制功能块的技术规范。有两种类型的运动控制指令: PLCopen 定义的指令和 MC 功能块独家指令。

1. PLCopen 运动控制的功能块

PLCopen 将运动控制的功能块标准化, 用 IEC61131-3 规定的编程语言定义程序接口。单轴定位、电子齿轮、多轴协调控制与执行指令的基本步骤一起定义。

通过使用 PLCopen 运动控制的功能块, 用户程序可以更加容易被重复使用, 而与硬件无关。也减少了培训和支持的费用。

2. MC 功能块的运动控制指令

有三种类型的运动控制指令, 见表 7-27。

表 7-27 运动控制指令类型

类 型	概 要
通用指令	MC 功能模块的通用指令
轴命令	用于执行单轴控制的 MC 功能模块指令
轴组命令	用于执行多轴协调控制的 MC 功能模块指令

7.6.2 状态转换

由指令执行引起的轴和轴组状态以及状态转换, 是基于 PLCopen 运动控制功能块的技术规范。

1. MC 功能模块的状态

MC 功能模块的状态如下:

(1) MC 运行

运动控制指令使能, 用户程序中的运动控制指令被解释, 运动控制被执行。可以不考虑 CPU 单元的运行模式, 而设置 MC 运行模式。

这个状态可以用 MC 公共变量 `_MC_COM.Status.RunMode` 监视。

(2) MC 试运行

这种状态下, 可以从 Sysmac Studio 执行试运行。

这个状态可以用 MC 公共变量 `_MC_COM.Status.TestMode` 监视。

(3) 保存凸轮表文件

指系统对凸轮表进行保存或等待处理时的状态。

这个状态可以用 MC 公共变量 `_MC_COM.Status.CamTableBusy` 监视。

2. 轴状态

运动控制指令顺序执行时, 轴进入图 7-34 所示状态中的某一种。

轴状态转换说明如下。

1) 除了协调控制状态, 其他任何一种状态下, 出现轴错误就会转换到这种状态。

2) 没有轴错误并且 MC_Power 指令的 Status 输出为 FALSE 时 (伺服 OFF), 转换到这种状态。

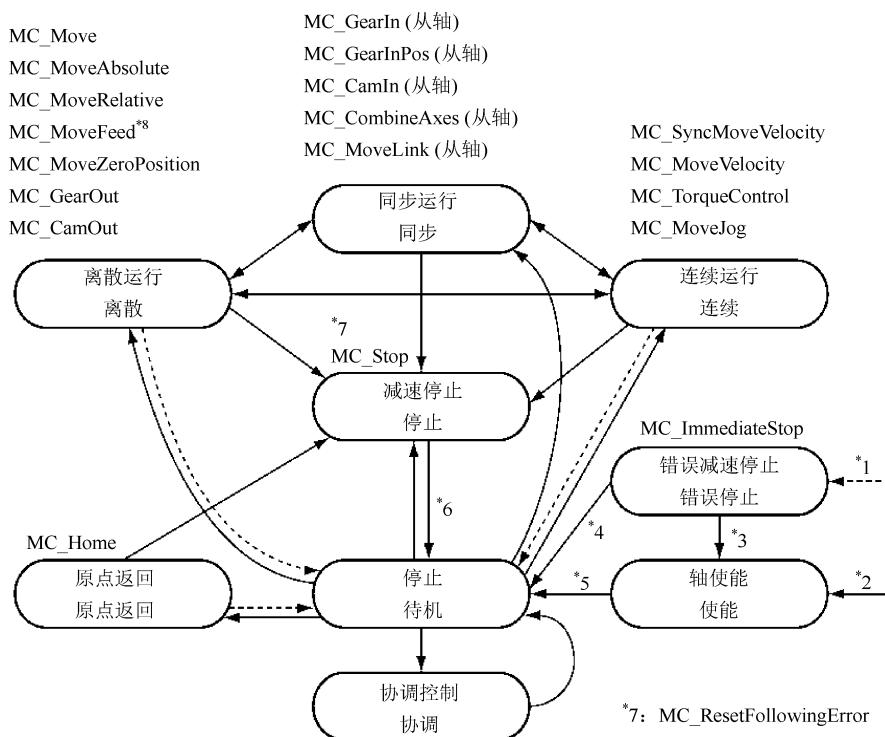


图 7-34 轴状态转换

- 3) 当伺服 OFF 时, 错误用 MC_Reset 或 ResetMCErr 指令复位, 转换到这种状态。
- 4) 当伺服 ON 时, 错误用 MC_Reset 或 ResetMCErr 指令复位时, 转换到这种状态。
- 5) 当 MC_Power 指令的 Enable 输入变为 TRUE, 并且 MC_Power 指令的 Status 输出 (伺服 ON) 变为 TRUE 时, 转换到这种状态 (伺服为 ON)。
- 6) 当 MC_STOP 指令的 Done 输出为 TRUE, 并且 MC_STOP 指令的 Execute 输入变为 FALSE 时, 转换到这种状态。
- 7) 当 MC_ResetFollowingError 指令执行时, 转换到减速停止状态。
- 8) 从 MC_MoveFeed 指令的 MoveMode 输入变量设置为速度控制, 一直到检测到触发输入, 连续运行状态一直存在。

3. 轴组状态

图 7-35 所示为轴组运动控制指令执行时, 轴组状态转换。

轴组状态转换说明如下。

- 1) 任何轴组运动控制指令执行时，进入运行状态。
- 2) 可以从其他任何状态进入错误停止状态，甚至在轴组禁用时发生错误也进入。
- 3) 对一个使能的轴组执行 MC_GroupReset 或 ResetMCErrror 指令时，进入待机状态。
- 4) 对一个禁用的轴组执行 MC_GroupReset 或 ResetMCErrror 指令时，进入禁用状态。
- 5) 在错误停止状态下，执行 MC_GroupDisable 指令，将返回到相同的状态。

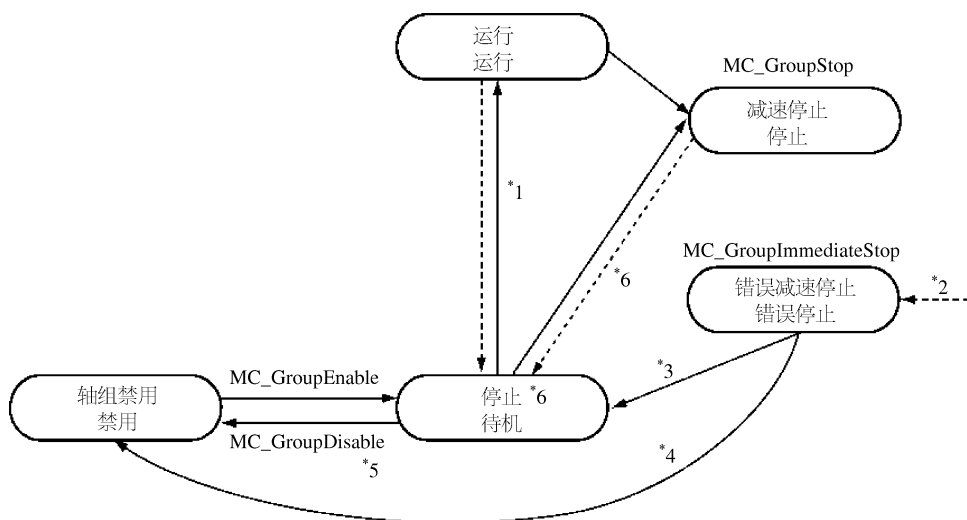


图 7-35 轴组状态转换

6) 当 MC_GroupStop 指令的 Done 输出变量为 TRUE，并且该指令的 Execute 输入变量是 FALSE 时，进入停止状态。

7) 在待机状态下，满足下列所有条件，就进入准备就绪状态。

- ①所有构成轴的伺服都为 ON。
- ②所有构成轴都没有执行 MC_Stop 指令。
- ③所有构成轴的原点都被定义。

7.6.3 运动控制指令的执行时序

MC 功能模块的运动控制指令是有条件执行的功能块，分为两类：

(1) Execute 型

当指令的输入变量 Execute 变为 TRUE 时，这些运动控制指令执行。这些指令会继续执行，直到出现下列情况之一。

- 1) 指定的操作被完成。
- 2) 另一个运动控制指令被执行并且中断操作。
- 3) 当 Execute 再次从 FALSE 变为 TRUE 时，指令重新执行。

当 Execute 变为 TRUE 时，其他输入变量的值被更新。

(2) Enable 型

当指令的输入变量 Enable 变为 TRUE 时，这些运动控制指令每个周期都执行。只要 Enable 为 TRUE 时，其他输入变量每个周期也会刷新。但是，MC_MoveJog 指令的输入变量 Velocity、Acceleration 和 Deceleration 除外。当 PositiveEnable 或 NegativeEnable 变为 TRUE 时，这些输入变量的值更新。

1. Execute 型指令的时序

Execute 型指令的梯形图如图 7-36 所示。

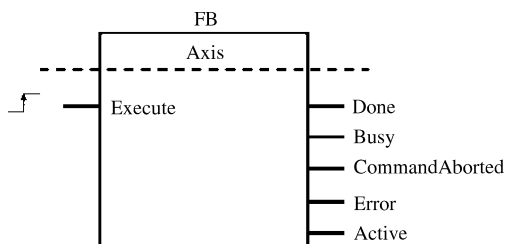


图 7-36 Execute 型指令的梯形图

Execute 型指令的输出变量见表 7-28。

表 7-28 输出变量

Done	当指令操作正常结束或当命令条件达到时变为 TRUE 在一些 FB 中,Done 输出功能由 InGear (齿轮比到达)或 InSync(同步到达)代替
CommandAborted	当另一条操作指令中断当前命令操作时,输出变量 CommandAborted 变为 TRUE
Error	指令例程执行时,如果输入变量在有效范围之外,或指令处理出错,将输出一个错误
Busy	当指令例程正在执行时,输出变量 Busy 为 TRUE。当输入变量 Execute 变为 TRUE,Busy 将变为 TRUE。当输出变量 Done、CommandAborted 或 Error 变为 TRUE,BUSY 将变为 FALSE
Active	当指令例程获得控制轴的许可时,输出变量 Active 变为 TRUE。在非缓冲模式下,输出变量 Active 可能比 Busy 变量变化的慢。但在缓冲模式下,变量可能取不同的值

图 7-37 所示为正常操作,输入变量 Execute 变为 TRUE 后,指令正常执行,完成操作。当 Execute 变为 FALSE 时,没有错误发生。

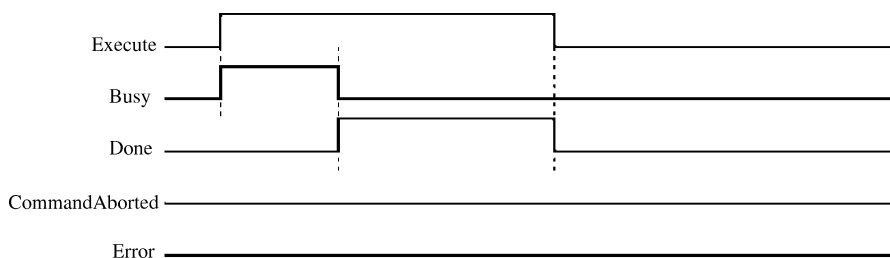


图 7-37 正常操作

图 7-38 所示为出现错误,输入变量 Execute 变为 TRUE 后,指令执行过程中出现错误。操作完成后,输出变量 Error 会保持 TRUE,即使 Execute 变为 FALSE。

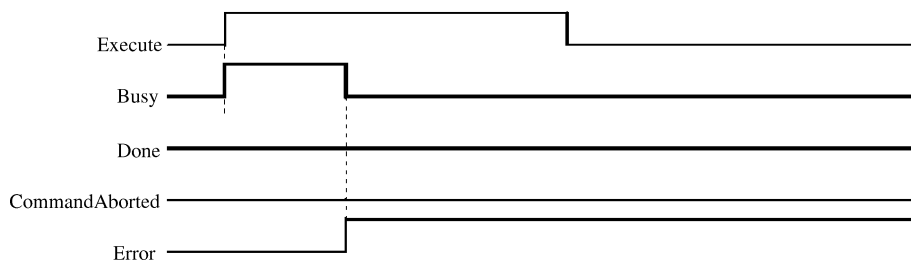


图 7-38 出现错误

图 7-39 所示为指令被中止,输入变量 Execute 为 TRUE,指令执行过程中被中止。

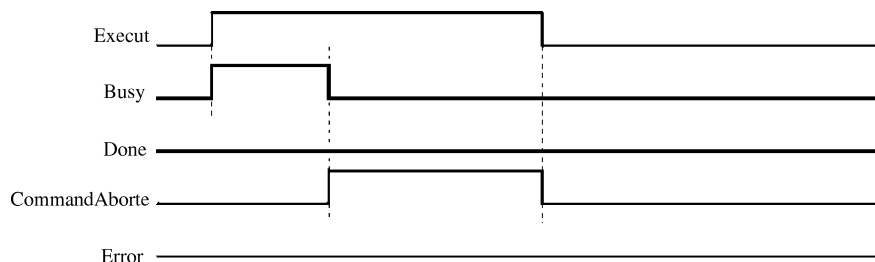


图 7-39 指令被中止

如图 7-40 所示, 输入变量 **Execute** 只保持 TRUE 一个周期, 指令执行过程中没有错误发生。输出变量 **Done** 只在指令操作完成后的一个周期内变为 TRUE。

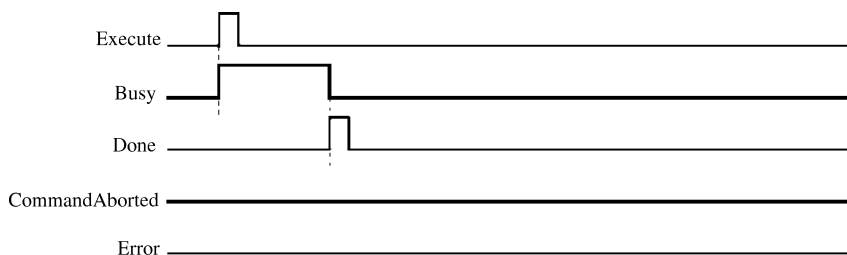


图 7-40 **Execute** 为 TRUE 只保持一个周期

如图 7-41 所示, 输入变量 **Execute** 只保持 TRUE 一个周期, 指令执行过程中有错误发生, 输出变量 **Error** 会输出 TRUE 并保持。

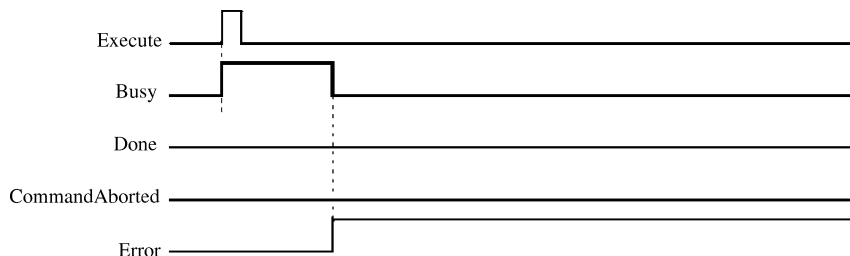


图 7-41 **Execute** 为 TRUE 只保持一个周期且出现错误

2. Enable 型指令的时序

Enable 型指令的梯形图如图 7-42 所示。

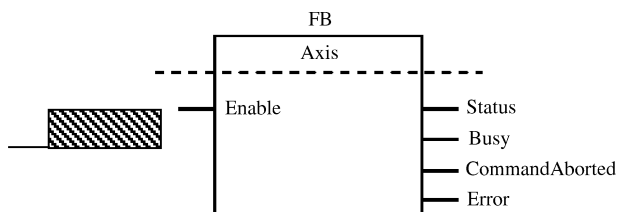


图 7-42 Enable 型指令的梯形图

Enable 型指令的输出变量见表 7-29。

表 7-29 Enable 型指令的输出变量

Status	输入变量 Enable 为 TRUE, 设备处于运行准备状态, 则输出变量 Status 变为 TRUE。当 Enable 变为 FALSE 时, 设备的运行准备状态消除, Status 变为 FALSE
CommandAborted	当另一条操作指令中断当前命令操作时, 输出变量 CommandAborted 变为 TRUE
Error	指令例程执行时, 如果输入变量在有效范围之外或指令处理出错, 将输出一个错误
Busy	当指令例程正在执行时, 输出变量 Busy 为 TRUE。当输入变量 Enable 变为 TRUE, Busy 将变为 TRUE。当输出变量 CommandAborted 、 Error 变为 TRUE, Busy 将变为 FALSE

如图 7-43 所示, 输入变量 Enable 变为 TRUE, 指令执行过程中没有出现错误。

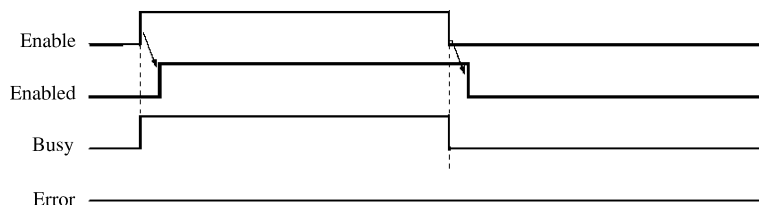


图 7-43 正常操作

如图 7-44 所示, 输入变量 Enable 变为 TRUE, 指令执行过程中有错误发生。

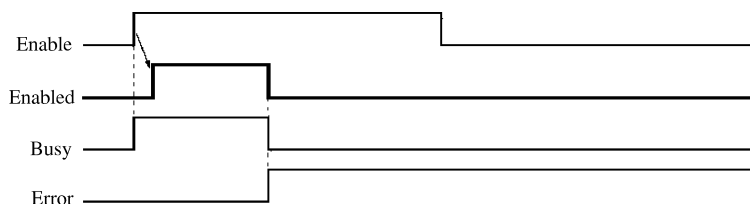


图 7-44 出现错误

7.6.4 运动控制指令重新执行时序图

在运动控制指令执行时, 如果同一例程的输入变量的值改变, 当 Execute 为 TRUE 后变为 FALSE, 然后再次变为 TRUE, 指令会按输入变量新的值操作。

图 7-45 所示为 MC_MoveAbsolute (绝对定位) 指令重新执行改变速度时的时序图。

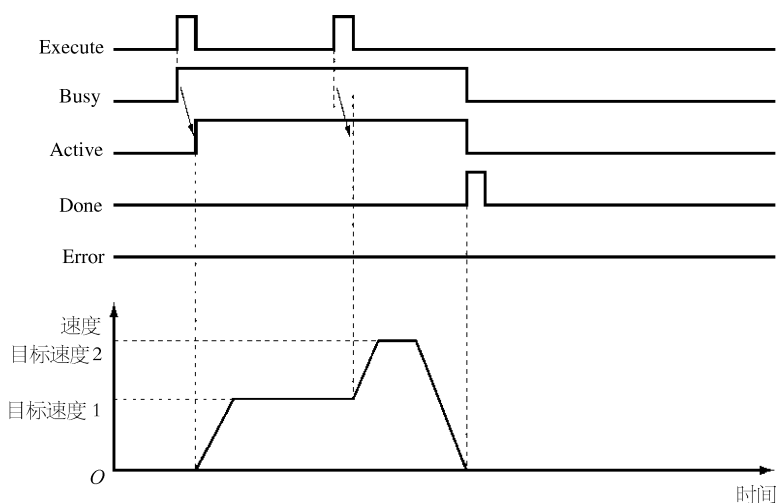


图 7-45 MC_MoveAbsolute 指令重新执行改变速度时的时序图

7.6.5 运动控制指令多次执行时序图

轴运动过程中, 可以执行轴的另一个例程。此时, 需要对运动控制指令输入变量 BufferMode 设置何时启动操作。

如图 7-46 所示, MC_MoveAbsolute (绝对定位) 指令多次执行, FB1 和 FB2 是指令的例程名称, FB1 的 BufferMode 设置为 Aborting。

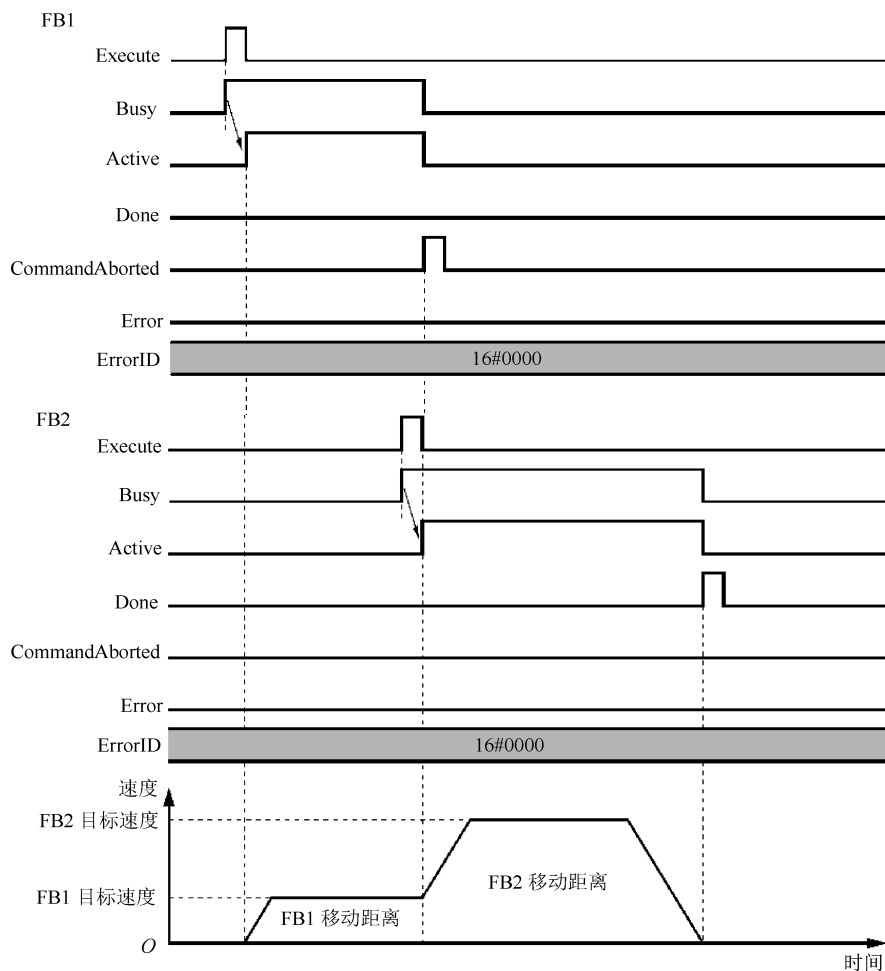


图 7-46 指令多次执行时序图

7.6.6 运动控制的系统定义变量

NJ 系列控制器与 IEC61131-3 标准相兼容。在 NJ 系列控制器的用户程序中, 参数设置、状态信息和其他数据被当作变量处理。其中, 属于 MC 功能模块的系统定义变量称为运动控制的系统定义变量。

运动控制的系统定义变量有以下三类:

1. MC 公共变量

变量名为 `_MC_COM`, 数据类型为结构体 `_sCOMMON_REF`。使用公共变量, 可以监视 MC 功能模块的总体状态。

`_MC_COM` 的结构如下。

<code>_MC_COM</code>	MC 公共变量
<code>_MC_COM.Satus</code>	MC 公共状态层级
<code>_MC_COM.Satus.RunMode</code>	MC 运行
<code>_MC_COM.Satus.TestMode</code>	MC 试运行
<code>_MC_COM.Satus.CamTableBusy</code>	凸轮表忙
<code>_MC_COM.PFaultLvl</code>	MC 公共局部故障层级
<code>_MC_COM.PFaultLvl.Active</code>	MC 公共局部故障出现
<code>_MC_COM.PFaultLvl.Active.Code</code>	MC 公共局部故障代码
<code>_MC_COM.MFaultLvl</code>	MC 公共轻微故障层级
<code>_MC_COM.MFaultLvl.Active</code>	MC 公共轻微故障出现
<code>_MC_COM.MFaultLvl.Active.Code</code>	MC 公共轻微故障代码
<code>_MC_COM.Obsr</code>	MC 公共观察层级
<code>_MC_COM.Obsr.Active</code>	MC 公共观察出现
<code>_MC_COM.Obsr.Active.Code</code>	MC 公共观察代码

2. 轴变量

使用轴变量处理 EtherCAT 从站、伺服驱动、编码器输入终端、虚拟伺服驱动和虚拟编码器输入终端。

在用户程序中，使用系统定义变量或 Sysmac Studio 设置的变量指定轴变量。

从 Sysmac Studio 可以改变任何一个轴变量。

系统定义变量轴变量名称：`_MC_AX[0] ~ _MC_AX[31]`。

在 Sysmac Studio 创建轴时产生的默认轴变量名称：`MC_Axis000 ~ MC_Axis063`。

关于轴变量的详细介绍可参见 7.3 节。

3. 轴组变量

使用轴组变量处理一个轴组中的多个轴。

在用户程序中，使用系统定义变量或 Sysmac Studio 设置的变量指定轴组变量。

从 Sysmac Studio 可以改变任何一个轴组变量。

系统定义变量的轴组变量名称：`_MC_GRP[0] ~ _MC_GRP[31]`。

在 Sysmac Studio 创建轴组时产生的默认轴组变量名称：`MC_Group[0] ~ MC_Group[31]`。

关于轴组变量的详细介绍可参见 7.4 节。

7.6.7 凸轮表和凸轮数据变量

用 Sysmac Studio 的凸轮编辑器生成凸轮 (表) 曲线，MC 功能模块将凸轮曲线当作变量使用。在 NJ 控制器的用户程序中，凸轮表数据是作为凸轮数据变量处理的。

1. 凸轮表的生成和保存

凸轮表的生成和保存如图 7-47 所示。

Sysmac Studio 创建的凸轮曲线下载到 CPU 单元，凸轮曲线作为凸轮表保存到 CPU 单元的非易失性内存中。

在凸轮表被下载后或上电时，保存的凸轮表在主内存中被当作凸轮数据变量。

可以使用用户程序在主内存中编辑凸轮数据变量。

应用运动控制指令 MC_SaveCamTable 可将主内存中凸轮数据变量保存到非易失性内存。

不管 CPU 单元处于何种操作模式或 MC 功能模块处于何种状态，都可以上载和下载凸轮表。进行凸轮表保存操作时，不能上载凸轮数据、下载凸轮数据、启动在线操作、进行在线编辑或启动数据跟踪。在线编辑时，不能执行 MC_SaveCamTable 指令。

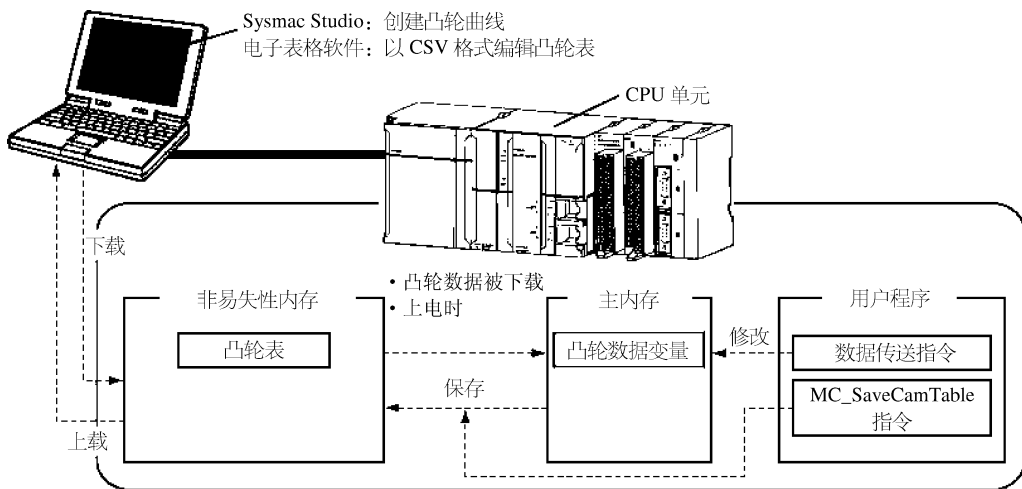


图 7-47 凸轮表的生成和保存

当启动下载过程后，所有运行的轴会以最大减速度减速并且伺服变为 OFF。

如果在用户程序中改变凸轮数据，当重新上电或从 Sysmac Studio 下载凸轮数据时，这些更改会丢失并且非易失性内存中的凸轮表恢复。如果需要保存在用户程序中修改的凸轮数据到非易失性内存，或者需要把它上载到 Sysmac Studio，先执行 MC_SaveCamTable 指令。

2. 在计算机上编辑凸轮变量

如果在用户程序中编辑凸轮数据变量，然后使用 MC_SaveCamTable 指令将凸轮表保存到非易失性内存中，不能使用 Sysmac Studio 的凸轮编辑器编辑凸轮数据。可使用电子表格软件对数据进行编辑，然后把它当作一个凸轮表使用。

具体操作时，使用 Sysmac Studio 从非易失性内存将凸轮表保存为 CSV 文件，使用电子表格软件或其他的 CSV 兼容软件编辑 CSV 文件，然后将 CSV 文件传到 CPU 单元，要使该凸轮数据有效，需复位控制器或控制器断电后重新上电。

3. 凸轮曲线名称

在 Sysmac Studio 生成凸轮曲线时，CamProfile0 是默认名称。当生成另一凸轮曲线时，名称末尾的序号递增。需要时，可以从 Sysmac Studio 修改凸轮曲线的名称。

4. 在用户程序中指定凸轮表

如图 7-48 所示，在用户程序的运动控制指令中，为输入-输出变量 CamTable 指定凸轮变量名 CamProfile0。

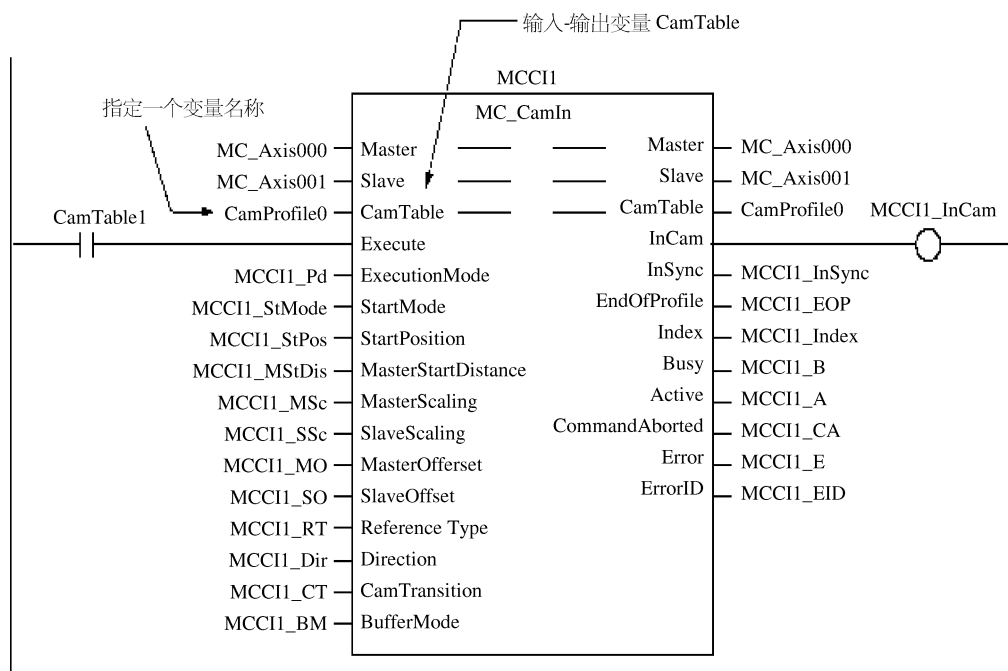


图 7-48 指定凸轮表

7.7 单轴运动控制指令

7.7.1 概述

单轴运动控制指令见表 7-30。

表 7-30 单轴运动控制指令

类 别	指 令	名 称	功 能
单轴位置控制	MC_MoveAbsolute	绝对定位	指定绝对目标位置,进行定位
	MC_MoveRelative	相对定位	指定离开命令当前位置移动的距离,进行定位
	MC_MoveFeed	中断进给	指定从外部设备触发中断输入的位置开始移动的距离,进行定位
单轴速度控制	MC_MoveVelocity	速度控制	速度控制在定位控制模式下实现
	MC_SyncMoveVelocity	周期同步速度控制	速度控制模式下,速度指令在每个控制周期都输出
单轴同步控制	MC_TorqueControl	扭矩控制	对电动机的扭矩进行控制
	MC_CamIn	启动凸轮操作	启动指定凸轮表的凸轮操作
	MC_CamOut	结束凸轮操作	结束指定了输入参数的轴的凸轮操作
	MC_GearIn	启动齿轮操作	指定主轴和从轴之间的齿轮比,启动齿轮操作
	MC_GearInPos	定位齿轮操作	指定主轴与从轴之间的齿轮比和同步位置,启动齿轮操作
	MC_GearOut	结束齿轮操作	停止当前的齿轮操作或定位齿轮操作

(续)

类 别	指 令	名 称	功 能
单轴同步控制	MC_MoveLink	同步定位	与指定主轴同步的定位
	MC_Phasing	主轴相移	当前同步控制中的主轴相位被移动
单轴手动操作	MC_CombineAxes	组合轴	输出两个轴命令位置的和或差作为命令位置
	MC_Power	伺服上电	伺服驱动上电使能
单轴控制的辅助功能	MC_MoveJog	点动	根据指定的目标速度进行点动
	MC_Reset	复位轴错误	清除轴错误
	MC_Home	原点返回	使用限位信号、原点接近信号和原点信号,操纵电动机返回原点
	MC_MoveZeroPosition	高速原点返回	将绝对位置 0 作为目标位置,轴返回原点
	MC_Stop	停止	轴被停止
	MC_ImmediateStop	立即停止	轴被立即停止
	MC_SetOverride	设置倍率因子	改变轴的目标速度
	MC_SetPosition	改变当前位置	需要时改变轴的命令当前位置或实际当前位置
	MC_TouchProbe	使能外部门锁	出现触发时,记录轴的位置
	MC_AbortTrigger	禁用外部门锁	禁用当前的门锁
	MC_ZoneSwitch	区域监视	系统确定轴的命令位置或实际当前位置是否在指定区域内
	MC_AxesObserve	轴跟踪误差监视	监视两个指定轴的命令位置或实际位置的跟踪误差,以确定是否超过了允许值
	MC_ResetFollowingError	跟踪误差计数器复位	命令当前位置和实际当前位置的误差置为 0
	MC_SetTorqueLimit	扭矩限制	通过设置扭矩的限值,使能和禁用伺服驱动转矩限位功能,限制输出转矩

以下为单轴指令常用的变量,这里统一说明,后面单独介绍指令时不再说明。

1. 输入变量

单轴指令常用的输入变量见表 7-31。

表 7-31 单轴指令常用的输入变量

名 称	含 义	数 据 类 型	取 值 范 围	默认值	说 明
Execute	执行	BOOL	TRUE 或 FALSE	FALSE	变为 TRUE 时,指令执行
Enable	使能	BOOL	TRUE 或 FALSE	FALSE	为 TRUE 时,使设备准备运行
Position	目标位置	LREAL	负实数、正实数或 0	0	指定运动模式为绝对定位时,设置绝对坐标系下的绝对位置;指定运动模式为相对定位时,设置相对坐标系下的相对位置。单位是命令单位

(续)

名 称	含 义	数 据 类 型	取 值 范 围	默认值	说 明
Velocity	目标速度	LREAL	非负实数	0	指定目标速度,单位是命令单位/s
Acceleration	加速度	LREAL	非负实数	0	指定加速度,单位是命令单位/s ²
Deceleration	减速度	LREAL	非负实数	0	指定减速度,单位是命令单位/s ²
Jerk	加加速度	LREAL	非负实数	0	指定加加速度,单位是命令单位/s ³
Direction*	方向	_eMC_DIRECTION	0:_mcPositiveDirection 1:_mcShortestWay 2:_mcNegativeDirection 3:_mcCurrentDirection 4:_mcNoDirection	0	当 MoveMode 指定为绝对定位并且计数模式为环形模式时,指定旋转方向: 0:正方向 1:最短路程 2:负方向 3:当前方向 4:未指定方向
BufferMode*	缓冲模式选择	_eMC_BUFFER_MODE	0:_mcAborting 1:_mcBuffered 2:_mcBlendingLow 3:_mcBlendingPrevious 4:_mcBlendingNext 5:_mcBlendingHigh	0	指定执行一个以上的运动控制指令时的操作: 0:Aborting 1:Buffered 2:Blend with Low 3:Blend with Previous 4:Blend with Next 5:Blend with High
MoveMode*	运动模式	_eMC_MOVE_MODE	0:_mcAbsolute 1:_mcRelative	0	选择运动模式: 0:绝对定位 1:相对定位

* 对枚举体变量,实际指定时用枚举体的元素,不用它对应的数值。

2. 输出变量

单轴指令常用的输出变量见表 7-32。

表 7-32 单轴指令常用的输出变量

名 称	含 义	数 据 类 型	取 值 范 围	说 明
Done	完成标志	BOOL	TRUE 或 FALSE	为 TRUE 时,指令完成
Busy	执行标志	BOOL	TRUE 或 FALSE	为 TRUE 时,指令得到认可
Active	受控标志	BOOL	TRUE 或 FALSE	为 TRUE 时,轴正在被控制
CommandAborted	指令中止	BOOL	TRUE 或 FALSE	为 TRUE 时,指令中止

(续)

名 称	含 义	数 据 类 型	取 值 范 围	说 明
Error	错误标志	BOOL	TRUE 或 FALSE	为 TRUE 时, 出现错误
ErrorID	错误代码	WORD		出现错误时, 包含错误代码。16#0000 表示正常执行

3. 输入-输出变量

Axis: 轴, 结构体_sAXIS_REF 型, 指定操作的轴。

7.7.2 伺服上电指令 MC_Power

1. 指令的梯形图

MC_Power 指令的梯形图如图 7-49 所示。

输出变量 Status: 伺服 ON, BOOL 型, 取值 TRUE 或 FALSE。为 TRUE 时, 轴操作准备就绪。

2. 功能

当 Enable 变为 TRUE 后, 由 Axis 指定的轴就会变为准备就绪。在轴处于准备就绪后, 可以对该轴实行控制。

当 Enable 转为 FALSE 后, 指定轴的准备状态立刻消除。当准备状态被消除后, 无法对该轴进行控制, 这是因为在此状态下, 轴不会承认相关的操作命令。此时, 执行运动命令会出现错误。在轴处于未准备状态下, 只能执行 MC_Power 和 MC_Reset 指令。

当轴处于运动状态下, 可以用这条指令禁止轴的操作。这种情况下, Commandaborted 变为 TRUE, 操作指令的输出会停止并且轴不再处于准备状态。

该指令适用于伺服轴和虚拟伺服轴, 如果用于编码器轴或者虚拟编码器轴会产生错误。如果有多条 MC_Power 指令对同一对象轴进行操作, 最后的指令有效。

图 7-50 为 MC_Power 指令的时序图。

当 Enable 变为 TRUE 时, Busy 变为 TRUE, 表明指令得到了认可。

当轴变为准备就绪状态, Status 变为 TRUE。

当 Enable 转为 FALSE, Busy 变为 FALSE, 当轴的准备状态被清除后, Status 变为 FALSE。

Status 这个输出变量反映轴的准备状态, 不管 Enable 为 TRUE 还是 FALSE。

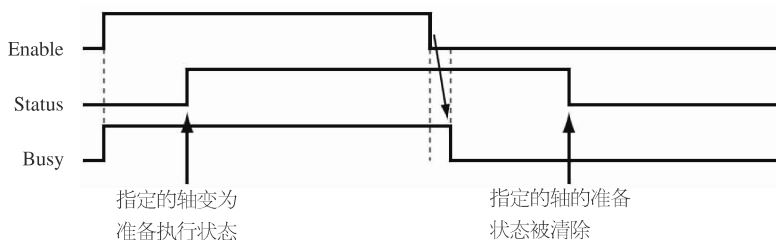


图 7-50 MC_Power 指令的时序图

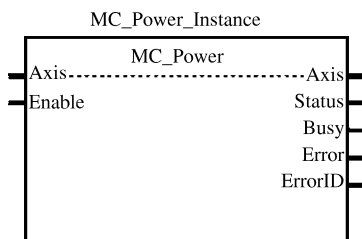


图 7-49 MC_Power 指令的梯形图

7.7.3 点动指令 MC_MoveJog

1. 指令的梯形图

MC_MoveJog 指令的梯形图如图 7-51 所示。

输入变量 PositiveEnable：正方向使能，BOOL 型，取值 TRUE 或 FALSE，默认 FALSE。变为 TRUE 时，轴正向启动；变为 FALSE 时，轴停止运行。

输入变量 NegativeEnable：负方向使能，BOOL 型，取值 TRUE 或 FALSE，默认 FALSE。变为 TRUE 时，轴负向启动；变为 FALSE 时，轴停止运行。

2. 功能

根据指定的速度（目标速度），MC_MoveJog 指令执行点动。

在正方向点动，将 PositiveEnable 变为 TRUE。在负方向点动，将 NegativeEnable 变为 TRUE。

如果 PositiveEnable 和 NegativeEnable 同时变为 TRUE，PositiveEnable 优先，结果，轴将正向点动。

如果 MC_MoveJog 指令的命令速度超过设置在轴参数中的最大点动速度，使用最大点动速度。

编写梯形图时，必须将 PositiveEnable 输入变量连接到左母线，同时为 NegativeEnable 输入变量指定一个变量，如图 7-52 所示。

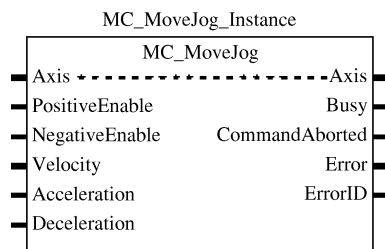


图 7-51 MC_MoveJog 指令的梯形图

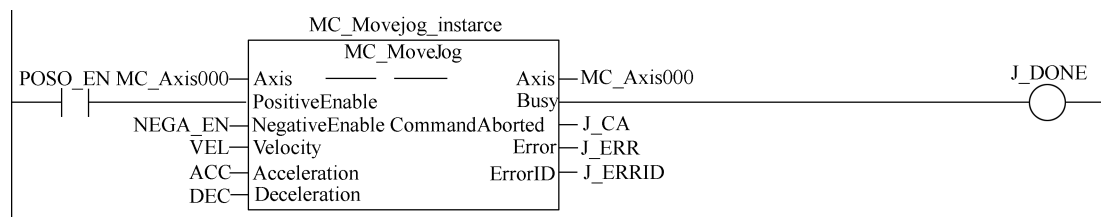


图 7-52 MC_MoveJog 指令的使用

MC_MoveJog 指令的时序图如图 7-53 所示。

只要 PositiveEnable 或 NegativeEnable 变为 TRUE，Busy 就变为 TRUE。

只要 PositiveEnable 或 NegativeEnable 变为 FALSE，轴开始减速。当轴完全停止时，Busy 就变为 FALSE。

如果另一条指令中止这条指令，CommandAborted 变为 TRUE，Busy 就变为 FALSE。

可以指定 Velocity、Acceleration 和 Deceleration 作为输入变量。在运动中，只有当 PositiveEnable 或 NegativeEnable 从 FALSE 变为 TRUE 时，输入变量 Velocity、Acceleration 和 Deceleration 才更新。因此，当 PositiveEnable 或 NegativeEnable 保持为 TRUE 时，即使 Velocity 变化，轴速度仍然不会改变。

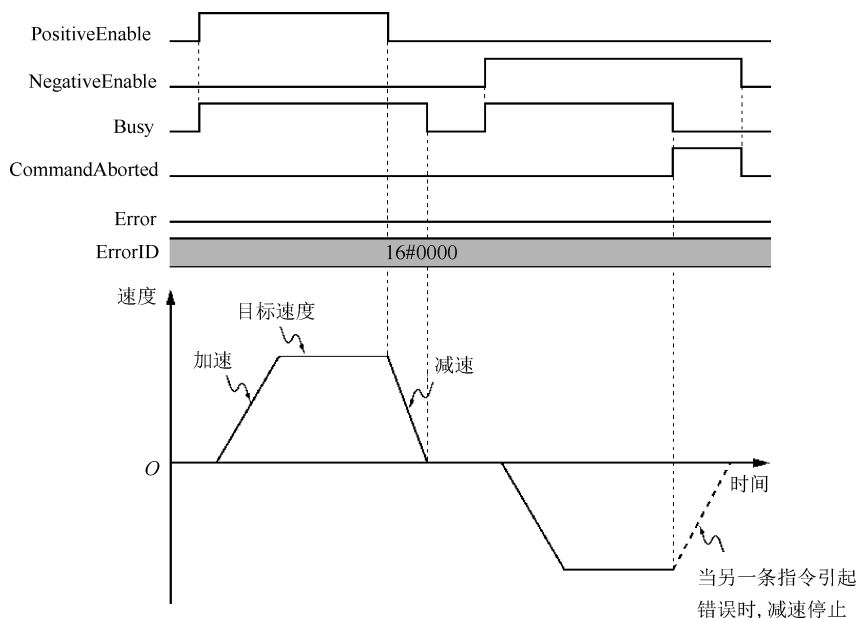


图 7-53 MC_MoveJog 指令的时序图

当 Velocity 为 0 时，启动轴的点动，轴将进入连续操作但没有运动。当 Acceleration 或 Deceleration 是 0，启动轴的点动，轴将不经过加速或减速过程直接达到目标速度。

可在同一方向上重新执行 MC_MoveJog 指令，例如，当 PositiveEnable 或 NegativeEnable 为 FALSE、轴处于减速状态，将 PositiveEnable 或 NegativeEnable 变为 TRUE，轴将开始朝着目标速度加速。这时，如果 Velocity、Acceleration 和 Deceleration 的值已改变，使用这些输入变量新的值。

可在相反方向上重新执行 MC_MoveJog 指令，例如，当 PositiveEnable 为 TRUE、轴在正方向上点动时，将 NegativeEnable 从 FALSE 变为 TRUE，轴将改变方向，在负方向上启动点动。这时，仍然使用 Velocity、Acceleration 和 Deceleration 这些输入变量的值。

7.7.4 原点返回指令 MC_Home

1. 指令的梯形图

MC_Home 指令的梯形图如图 7-54 所示。

2. 功能

当 Execute 变为 TRUE 时，对指定的轴启动原点返回操作。

MC_Home 指令有 10 种原点返回方法，在 Sysmac Studio 的原点返回设置中，设置要求的方法，如图 7-55 所示，原点返回方法设置为“零位置预设”，即把当前的位置设为零点。

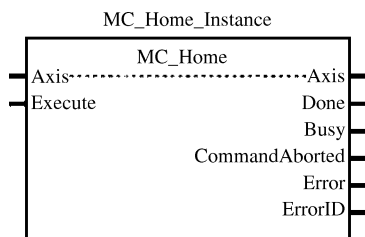


图 7-54 MC_Home 指令的梯形图



图 7-55 原点返回设置为“零位置预设”

也在 Sysmac Studio 的原点返回设置中, 设置 MC_Home 指令所用的参数, 如图 7-56 为“零位置预设”的参数设置, 这种原点返回方法只能设置原点偏移量, 其他不能设置。

使用 MC_Home 指令时, 在 Sysmac Studio 中的轴基本设置显示中的详细设置区建立下列映射:

- 探针功能 (60B8H)
- 探针状态 (60B9H)
- 探针 pos1 位置值 (60BAH)

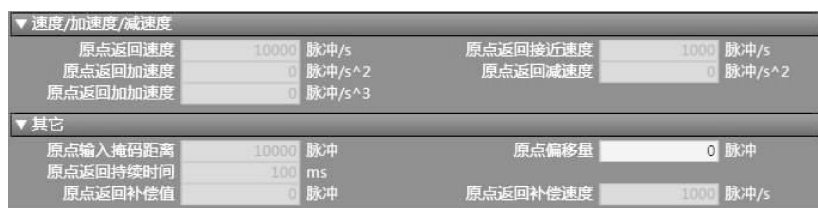


图 7-56 MC_Home 指令的参数设置

模式 11、12 和 14 的原点返回方式, 不需要以上设置。

如果一个需要的对象没有设置, 一个过程数据对象目标设置丢失错误 (错误代码: 3461H) 会出现。

对 OMRON C5 系列伺服驱动使用原点返回指令时, 要设置输入信号, 例如原点接近信号、原点信号。

图 7-57 为零位置预设 (无原点补偿) 的原点返回时序图。

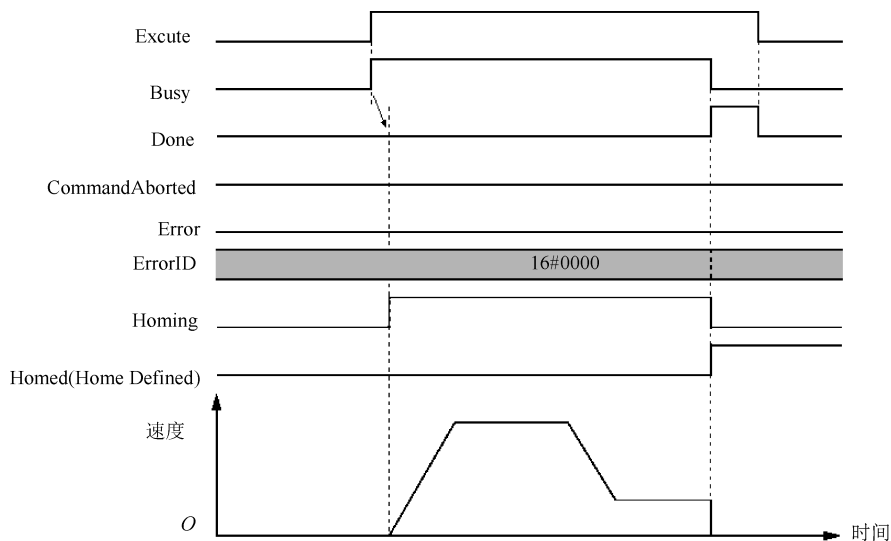


图 7-57 零位置预设 (无原点补偿) 的原点返回时序图

7.7.5 运动指令 MC_Move

1. 指令的梯形图

MC_Move 指令的梯形图如图 7-58 所示。

2. 功能

可以使用 MC_Move 指令完成绝对定位或相对定位。

如果将 MoveMode 指定为绝对定位，其操作与 MC_MoveAbsolute 指令是一样的。如果将 MoveMode 指定为相对定位，其操作与 MC_MoveRelative 指令是一样的。

如果使用相对定位，输入变量 Direction 是不用的。

进行绝对定位时，设定目标位置，不要超过轴参数中模最大位置设置值和模最小位置设置值。如果超出了，会出现目标位置设置范围超出错误 (错误代码 5478H)。

MC_Move 指令可以重新执行，也可以多次执行。

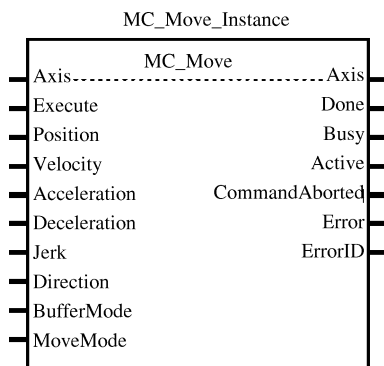


图 7-58 MC_Move 指令的梯形图

7.7.6 伺服停止指令 MC_Stop

1. 指令的梯形图

MC_Stop 指令的梯形图如图 7-59 所示。

2. 功能

Execute 变为 TRUE 后，启动减速停止操作，将轴的速度从当前值减速到 0。

当 MC_Stop 执行时，正在运行指令的 CommandAborted 变为 TRUE。

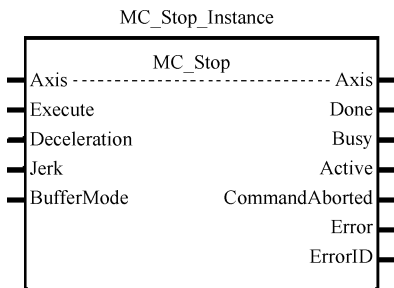


图 7-59 MC_Stop 指令的梯形图

如果指定减速度为 0，执行立即停止，即没有减速立即将轴停止。

当 Jerk 为 0 时，速度以指定的减速度直线下降，如图 7-60 所示。

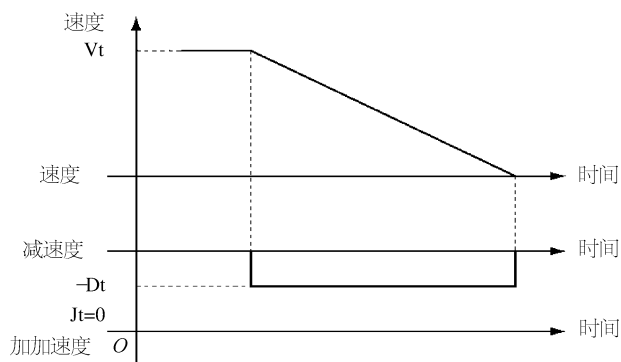


图 7-60 速度以指定的减速度直线下降 (Jerk = 0)

当 Jerk $\neq 0$ 时，减速度呈梯形，减速度达到其限值，速度以 S 曲线下降，如图 7-61 所示。

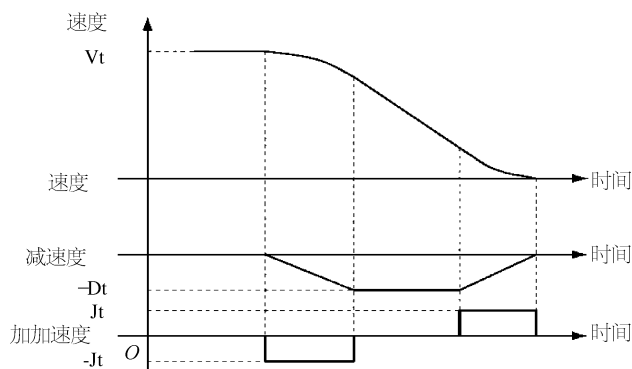


图 7-61 速度以 S 曲线下降 (Jerk $\neq 0$)

图 7-62 为 MC_Stop 指令的时序图。

当 Execute 变为 TRUE 后，Busy 同时变为 TRUE。Active 在下一个周期变为 TRUE。

当 Velocity 到达 0 时，Done 变为 TRUE。

如果另一条指令中断这条指令，CommandAborted 变为 TRUE，Busy、Active 变为 FALSE。

MC_Stop 指令可以重新执行，也可以多次执行。

7.7.7 速度控制指令 MC_MoveVelocity

1. 指令的梯形图

MC_MoveVelocity 指令的梯形图如图 7-63 所示。

输入变量 Continuous (保留): 连续模式选择，BOOL 型，取值范围 TRUE 或 FALSE。

输出变量 InVelocity: 目标速度到达，BOOL 型，取值范围 TRUE 或 FALSE。为 TRUE 时，表示目标速度到达。

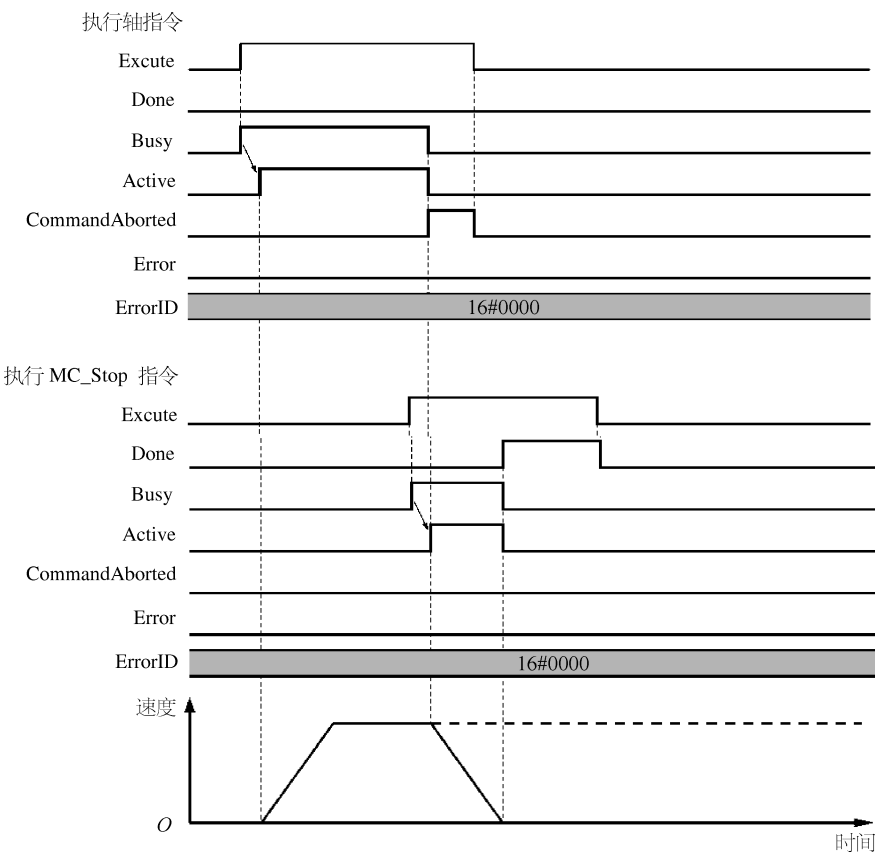


图 7-62 MC_Stop 指令的时序图

2. 功能

当 Execute 变为 TRUE 后，启动速度控制操作。

当 Execute 变为 TRUE 后，Busy 同时变为 TRUE。

Active 在下一个周期变为 TRUE。

图 7-64 为 MC_MoveVelocity 指令的时序图。

当 Velocity 到达设定值时，InVelocity 变为 TRUE。

如果另一条指令中断这条指令，CommandAborted 变为 TRUE，Busy、Active 和 InVelocity 变为 FALSE。

输出变量 InVelocity 表示这条指令和重新执行的指令达到同样的速度，因此，当 InVelocity 变为 TRUE 后，即使速度因为倍率因子而改变，InVelocity 不会变为 FALSE。如果在 InVelocity 变为 TRUE 之前倍率因子改变，当新的目标速度达到时，InVelocity 变为 TRUE。

MC_MoveVelocity 指令可以重新执行，也可以多次执行。

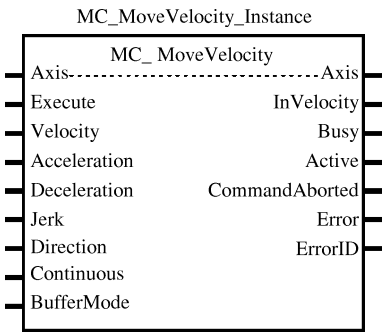


图 7-63 MC_MoveVelocity 指令的梯形图

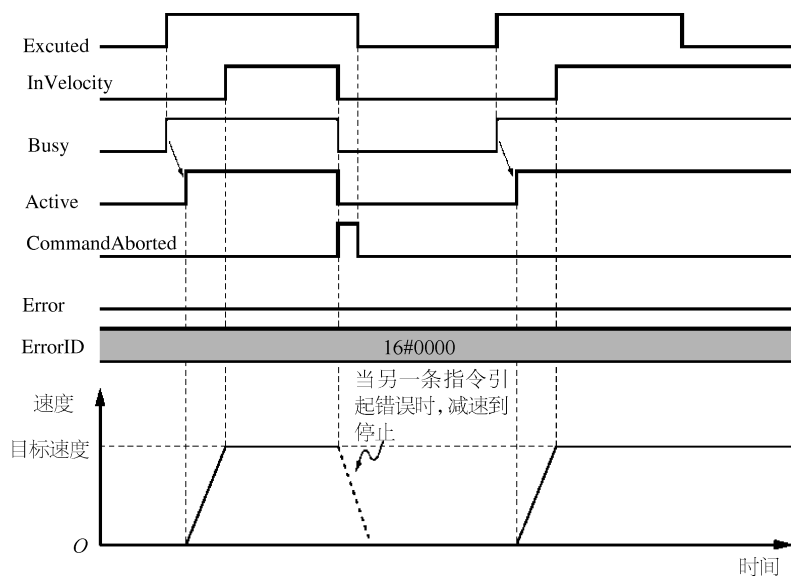


图 7-64 MC_MoveVelocity 指令的时序图

例 7.1 设备的速度控制如图 7-65 所示, 设备可以是纺纱机、离心机等。

(1) 参数设置

轴类型

轴 1: 伺服轴

(2) 操作要求

图 7-66 为速度控制时的操作要求。

操作过程如下:

1) 启动速度控制

传感器 1 检测液体化学品的渗入, 当它变为 ON 时, 启动轴 1 的速度控制。

2) 变为第二速度

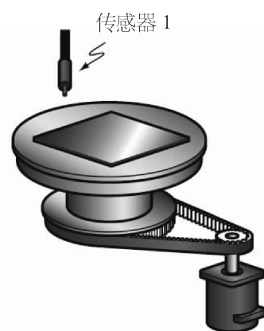


图 7-65 设备的速度控制

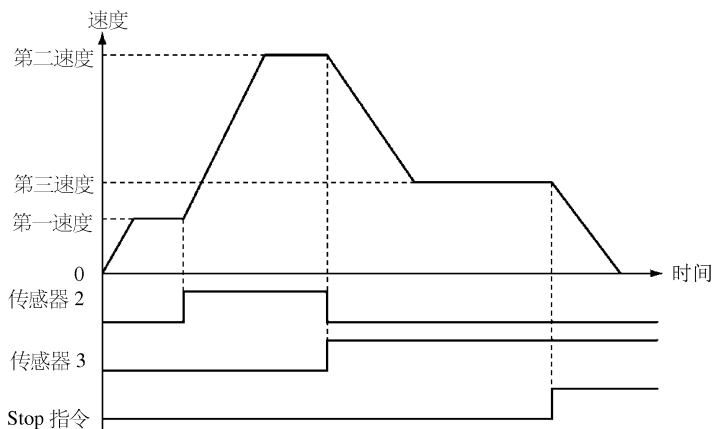


图 7-66 操作要求

当传感器 2 变为 TRUE 时，倍率因子设置为 500%，速度被改变。

3) 变为第三速度

当传感器 3 变为 TRUE，倍率因子设置为 200%，速度被改变。如果传感器 2 和 3 同时变为 TRUE，倍率因子设置为 200%。

4) 停止速度控制

当停止命令 (StopTrig) 变为 TRUE 时，轴减速到停止。

(3) 主要变量

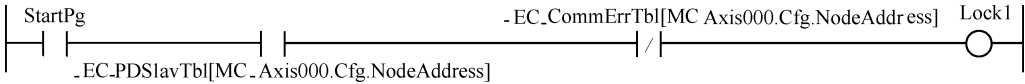
主要变量见表 7-33。

表 7-33 主要变量

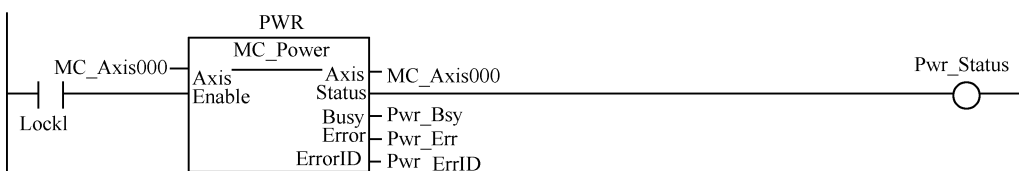
名 称	数 据 类 型	默 认 值	注 释
MC_Axis000	_sAXIS_REF	—	轴 1 的轴变量
MC_Axis000. MFaultLvl. Active	BOOL	FALSE	如果轴 1 有一个轻微故障级的错误,则为 TRUE
MC_Axis001. Details. Homed	BOOL	FALSE	轴 1 的原点定义之后是 TRUE
Pwr_Status	BOOL	FALSE	该变量被分配给 MC_Power 指令的 PWR 实例中的 Status 输出变量。当伺服为 ON 时,该变量为 TRUE
StartPg	BOOL	FALSE	如果该变量为 TRUE 并且建立了 EtherCAT 数据通信, 伺服为 ON
Vel_Act	BOOL	FALSE	该变量被分配给 MC_MoveVelocity 指令的 VEL 实例中的 Active 输出变量。VEL 实例的速度控制中,该变量为 TRUE
Set_Ov_Velfct	LREAL	0	这是速度倍率因子
StopTrig	BOOL	FALSE	如果该变量为 TRUE,MC_Stop 指令被执行
Sensor1	BOOL	FALSE	检测到化学溶液时,变为 TRUE 如果伺服轴 1 为 ON 时,MC_MoveVelocity 指令执行
Sensor2	BOOL	FALSE	如果这个变量为 TRUE,倍率因子设定为 500%。这个变量变为 TRUE 后,一直保持 TRUE,直到传感器 3 变为 TRUE,此时它变为 FALSE
Sensor3	BOOL	FALSE	如果这个变量为 TRUE,倍率因子设定为 200%。这个变量变为 TRUE 后,一直保持 TRUE

(4) 梯形图

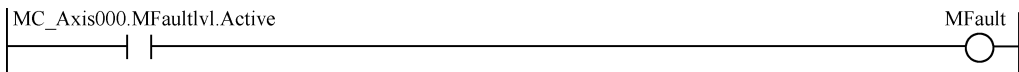
如果 StartPg 为 TRUE，检查 EtherCAT 通信，确认过程数据通信是否正常。



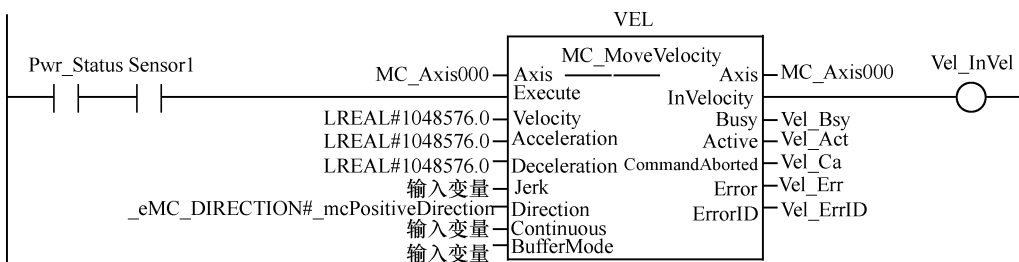
如果过程数据通信激活, Servo 变为 ON。



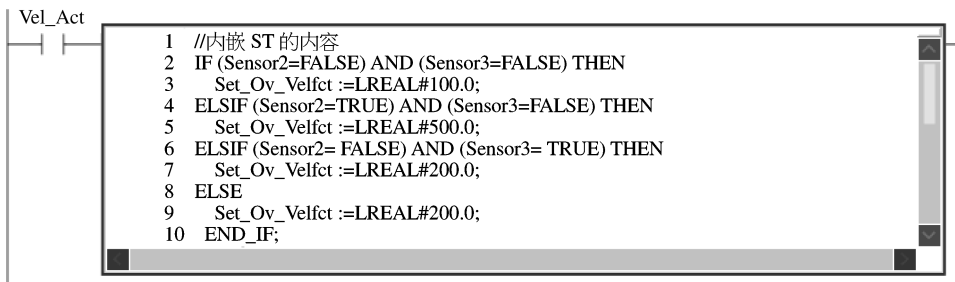
如果轴 1 有一个轻微故障级错误, 变量 MFault 为 ON。



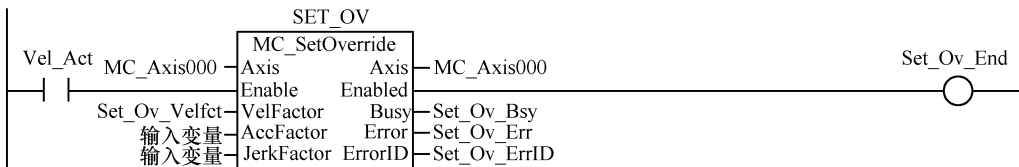
当 Sensor1 变为 TRUE 时, 执行 MC_MoveVelocity 指令。



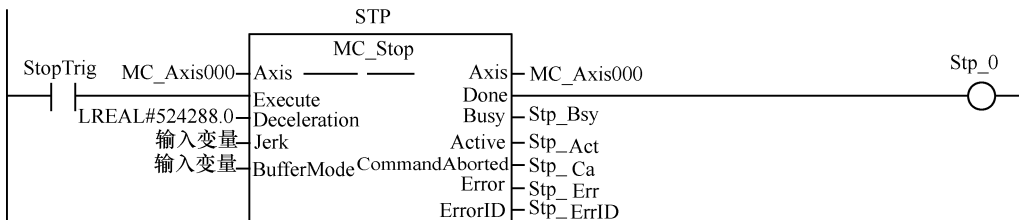
在执行 MC_MoveVelocity 指令期间, 根据 Sensor2 和 Sensor3 位的状态改变倍率因子。



在执行 MC_MoveVelocity 指令期间, 执行 MC_SetOverride 指令。



当停止输入 (StopTrig) 变为 TRUE 时, 执行 MC_Stop 指令。



(5) 时序图

图 7-67 为速度控制的时序图。

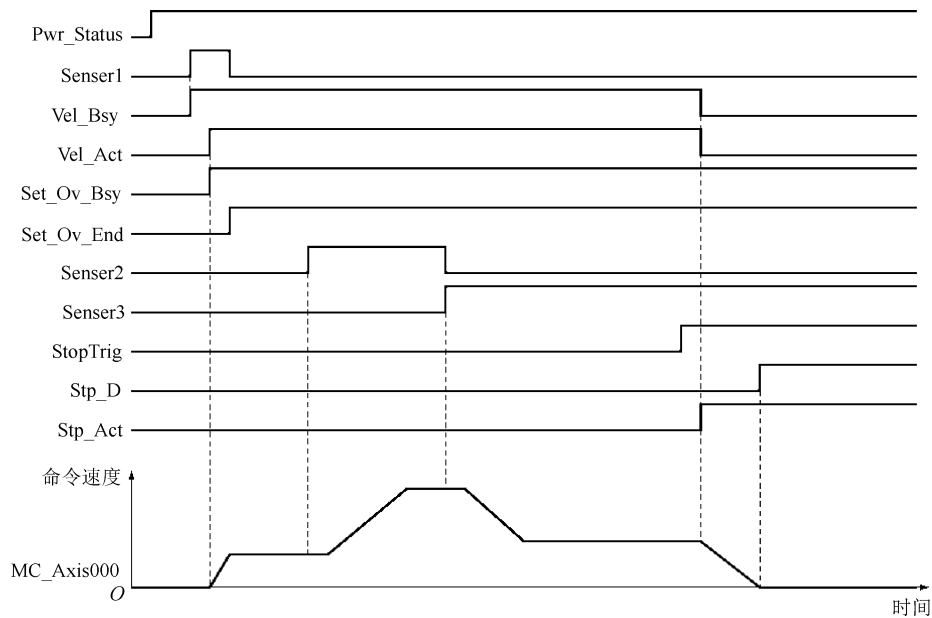


图 7-67 速度控制的时序图

7.7.8 启动凸轮操作指令 MC_CamIn

1. 指令的梯形图

MC_CamIn 指令的梯形图如图 7-68 所示。

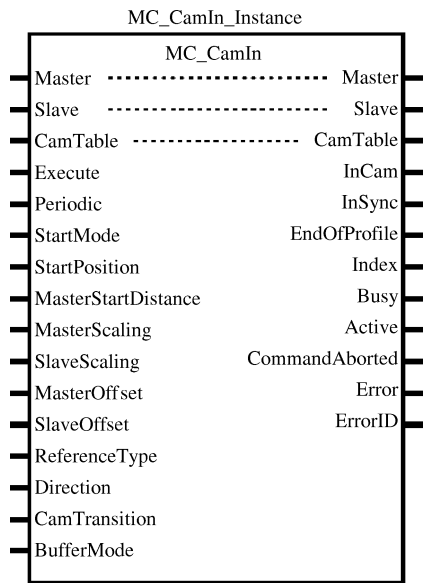


图 7-68 MC_CamIn 指令的梯形图

MC_CamIn 指令的输入变量见表 7-34。

表 7-34 MC_CamIn 指令的输入变量

名 称	含 义	数 据 类 型	取 值 范 围	默认值	说 明
Execute	执行	BOOL	TRUE 或 FALSE	FALSE	当 Execute 变为 TRUE 时,指令被执行
Periodic	周期模式	BOOL	TRUE 或 FALSE	FALSE	指定是否周期性或一次性执行指定的凸轮表: FALSE:非周期 TRUE:周期
StartMode	启动模式	_eMC_START_MODE	0: _mcAbsolutePosition 1: _mcRelativePosition	0	指 定 MasterStartDistance 使用的坐标: 0:绝对位置 1:相对位置
StartPosition	凸轮表 起始位置	LREAL	负数、正数或 0	0	指定凸轮表的起始位置(0 相位),作为主轴的绝对位置。单位是命令单位
MasterStart Distance	主轴跟随 距离	LREAL	负数、正数或 0	0	指定跟随轴开始凸轮运动时主轴的位置。当用 StartMode 指定绝对位置时,指定主轴的绝对位置。指定相对位置时,指定一个相对 StartPosition (Cam 表的起始位置)的距离。单位是命令单位
MasterScaling	主轴系数	LREAL	正值(>0.0)	1.0	使用指定的尺度扩展或收缩主轴的相位
SlaveScaling	从轴系数	LREAL	正值(>0.0)	1.0	使用指定的尺度扩展或收缩从轴的位移
MasterOffset	主轴偏移	LREAL	负数、正数或 0	0	使用指定的偏移值偏移主轴的相位
SlaveOffset	从轴偏移	LREAL	负数、正数或 0	0	使用指定的偏移值偏移从轴的位移
ReferenceType	位置类型 选择	_eMC_REFERENCE_TYPE	0: _mcCommand 1: _mcFeedback 2: _mcLatestCommand	0	指定主轴的位置类型: 0:命令位置(值在之前的主周期计算) 1:实际位置(值在同一主周期获得) 2:命令位置(值在同一主周期计算)

(续)

名 称	含 义	数 据 类 型	取 值 范 围	默认值	说 明
Direction	方向	_eMC_DIRECTION	0: _mcPositiveDirection 2: _mcNegativeDirection 4: _mcNoDirection	4	当主轴只在指定的方向运动时,从轴凸轮运行。如果主轴移动方向与指定的方向相反时,从轴凸轮停止: 0:正方向 2:负方向 4:无指定方向
CamTransition (保留)	凸轮过渡选择	_eMC_CAM_TRANSITION	0: _mcCTNone	0	(保留)
BufferMode	缓冲模式选择	_eMC_BUFFER_MODE	0: _mcAborting 1: _mcBuffered	0	指定执行一个以上的运动指令时的操作: 0:中止 1:缓冲

MC_CamIn 指令的部分输出变量见表 7-35。

表 7-35 MC_CamIn 指令的部分输出变量

名 称	含 义	数据类型	取 值 范 围	单位	说 明
InCam	凸轮运行	BOOL	TRUE 或 FALSE	—	当凸轮表起始点执行时为 TRUE
InSync	同步	BOOL	TRUE 或 FALSE	—	凸轮处在操作过程中为 TRUE
EndOfProfile	凸轮周期结束	BOOL	TRUE 或 FALSE	—	当凸轮表结束点执行时为 TRUE
Index	索引	UNIT	非负值	—	包含凸轮数据的索引号

MC_CamIn 指令的输入-输出变量见表 7-36。

表 7-36 MC_CamIn 指令的输入-输出变量

名 称	含 义	数 据 类 型	取值范围	说 明
Master	主轴	_sAXIS_REF	—	指定主轴
Slave	从轴	_sAXIS_REF	—	指定从轴
CamTable	凸轮表	ARRAY[0 . N] OF _sMC_CAM_REF	—	指定凸轮数据结构_sMC_CAM_REF 数组变量作为凸轮表

2. 功能

MC_CamIn 指令根据凸轮表执行一个主轴相位和从轴位移同步的凸轮运动。

必须使用凸轮编辑器为该指令创建凸轮表，并预先下载到控制器中。

当 Execute 变为 TRUE 时，执行指令。

(1) 凸轮操作

凸轮操作如图 7-69 所示，按照凸轮表使得从轴与主轴位置同步。用 MC_CamIn 指令启

动凸轮操作，用 MC_CamOut 指令或 MC_Stop 指令结束凸轮操作。在 Sysmac Studio 中使用凸轮编辑器生成凸轮表并下载到 CPU 单元中。从用户程序，可以修改凸轮表的凸轮数据。

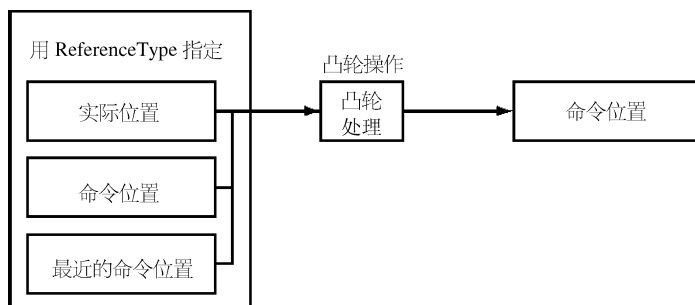


图 7-69 凸轮操作

(2) 凸轮表

凸轮与凸轮表如图 7-70 所示。凸轮操作用到凸轮表。在凸轮表中，指定主轴相位和从轴对应位移为一个凸轮数据，从起始点 0-0 开始。

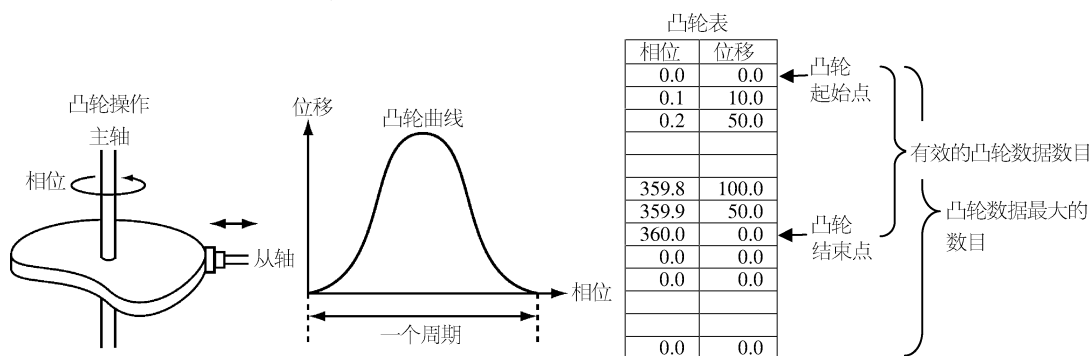


图 7-70 凸轮与凸轮表

凸轮数据如图 7-71 所示。两个凸轮数据之间，凸轮运行的命令位置是线性插值。如果只有少量的凸轮数据，相位间隔大，凸轮操作不会很精确。如果有许多凸轮数据，相位间隔小，凸轮操作将很精确。凸轮表中的数据越多，凸轮曲线的轨迹越精确和平滑。

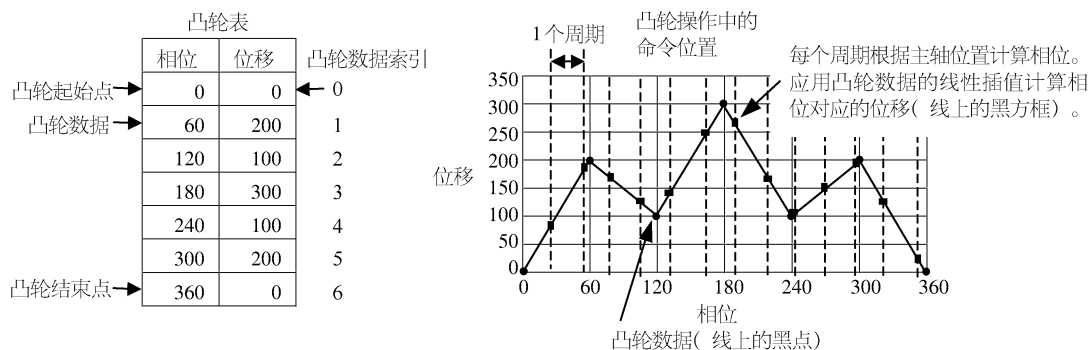


图 7-71 凸轮数据

执行时,指令检查相位是否按升序排列,如果不是升序,会出现错误,凸轮运动会停止。

改变凸轮数据时,执行 MC_SetCamTableProperty (设置凸轮表属性) 指令以确保相位按升序排列。

凸轮数据变量是全局变量。因此,可以从多个任务访问或改变凸轮数据变量的值。如果从多个任务改变凸轮数据变量的值,编程时注意,不要存在竞争关系。

对于一个凸轮数据变量,如果在任务间使用全局变量的排他控制,不控制这个变量的任务中的运动控制指令,不要使用凸轮数据变量。否则,一个不正确的凸轮表规范错误 (错误代码: 5439H) 会发生。

(3) 凸轮表指标

凸轮表指标见表 7-37。

表 7-37 凸轮表指标

项 目	说 明
每个凸轮表凸轮数据的最大数量	65535 点
所有凸轮数据的最大值	1048560 点
凸轮表的最大数量	640 个表
切换凸轮操作	可以通过执行运动控制指令切换到不同凸轮操作
覆盖凸轮数据	凸轮数据可以通过用户程序覆盖
保存凸轮数据	凸轮数据可以通过使用保存凸轮表指令,将其保存到非易失性内存中
与凸轮数据相关的信息	可以下载或上载信息,显示在凸轮编辑器
加载凸轮数据到主内存的时序	● 当数据从 Sysmac Studio 下载 ● 电源上电

如果每一个凸轮表有 65535 个点,最多 16 个凸轮表。如果分辨率为 0.1° ,则每个凸轮最多 3600 个点时,最多 291 个凸轮。凸轮总的容量为 10MB。

(4) 凸轮表的数据类型

凸轮表声明为凸轮数据结构体的数组,凸轮数据结构体的类型声明如下:

```

TYPE
(* Cam data structure* )
_sMC_CAM_REF:
STRUCT
    Phase      : REAL;    (* Phase* )
    Distance   : REAL;    (* Displacement* )
END_STRUCT;
END_TYPE

```

必须用 Sysmac Studio 的凸轮编辑器生成凸轮数据,然后指定凸轮表的名称和凸轮数据的数目 (也就是,数组的长度),例如,用下面的声明,生成名称为 MyCam1 的凸轮表,有

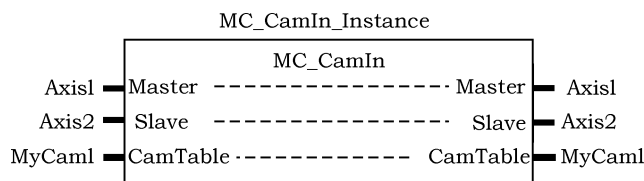
1000 个点。

```

VAR
    (* Cam table* )
    MyCam1      :   ARRAY[0..999] OF_sMC_CAM_REF;
END_VAR

```

下列标记用来为 Cam 操作指令指定 MyCam1，本例中，主轴是 Axis1，从轴是 Axis2。



如果指定的凸轮表在控制器中不存在，会出现错误。可以为多个凸轮指定同一个凸轮表。

(5) 凸轮装载/存储凸轮数据和存储凸轮表

像其他变量一样，从用户程序可以装载和存储凸轮数据。例如，在名称为 MyCam1 的凸轮表的首个数组元素，使用 MyCam1[0].Phase 指定相位和 MyCam1[0].Distance 指定位移。通过执行 MC_SaveCamTable 指令，从用户程序覆盖的凸轮数据作为凸轮表存储到 CPU 单元非易失性内存中。

3. 指令详情

(1) 指令执行条件

进行定位控制、速度控制或同步控制的主轴，在停止后，执行这条指令。

(2) 软件限位

凸轮操作期间，如果从轴超出软件限位，出现错误。

(3) 凸轮数据变量

凸轮数据变量为凸轮数据结构体的数组。

(4) 启动凸轮操作

指令启动后，主轴必须到达 StartPosition（凸轮表起始位置）。当主轴到达 StartPosition、移动 MasterStartDistance 指定的距离后，从轴凸轮操作开始。

MasterStartDistance 被指定为一个从 StartPosition 离开的绝对位置或相对距离。

用 StartMode 设置使用绝对位置或相对位置。

图 7-72 所示的 StartMode 设置为相对位置，凸轮表的起始位置为 20°，主轴启动距离为 40°，并且执行模式设置为周期性的。

图 7-73 为 StartMode 设置为绝对位置，凸轮表的起始位置为 20°，主轴启动距离到 60°，并且执行模式设置为周期性的。

(5) Periodic (周期模式)

如果将 Periodic 指定为 TRUE (周期)，凸轮将从凸轮表的起始点到终点重复运动。

如果将 Periodic 指定为 FALSE (非周期)，凸轮运动将在凸轮表的最后一点执行后停止。

如果从轴的行程位置与凸轮表的起始点和终点相同，在 Periodic 设置为 TRUE 时，凸轮

作为往复凸轮工作。如果从轴的行程位置与凸轮表的起始点和终点不同，凸轮作为进给凸轮工作。

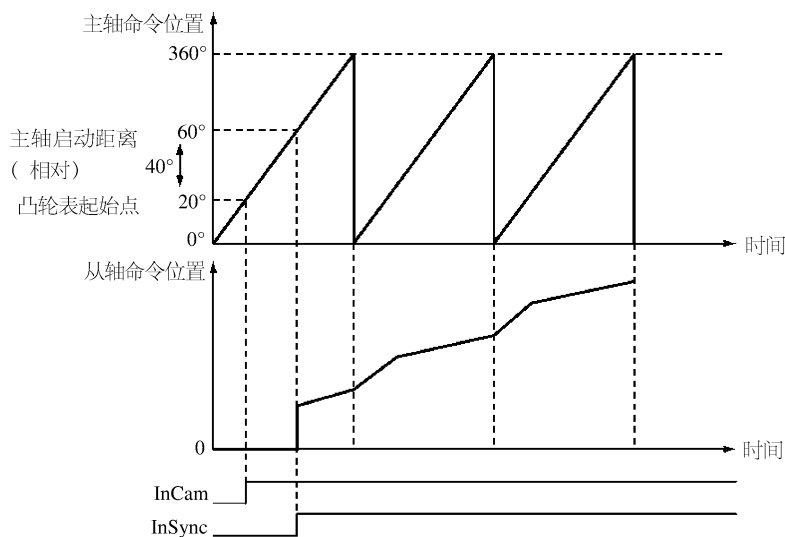


图 7-72 StartMode 设置为相对位置

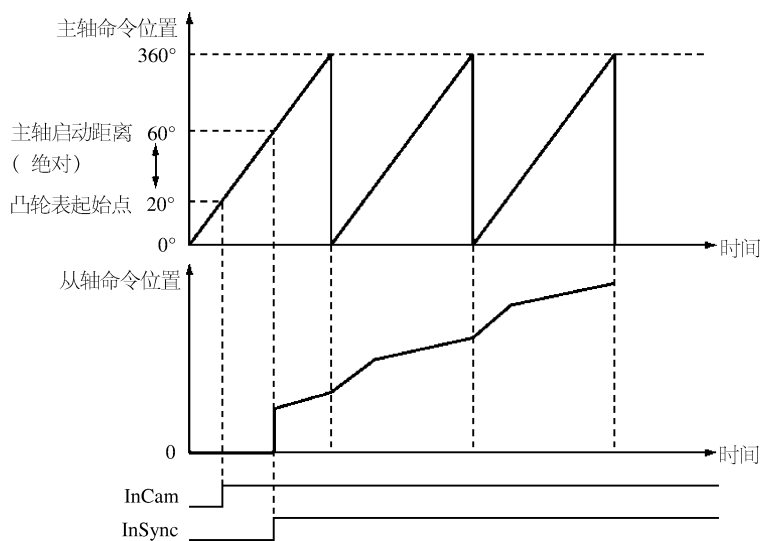


图 7-73 StartMode 设置为绝对位置

(6) EndOfProfile (凸轮周期结束)

在凸轮表终点指定的相位和位移的凸轮运动的命令值输出后，EndOfProfile 为 TRUE 一个周期。为 StartPosition 设置主轴的绝对位置，则凸轮表相对于那个位置。EndOfProfile 的输出表示凸轮表结束。

(7) 结束凸轮操作

在凸轮操作完成之前，可使用 MC_CamOut (结束凸轮操作) 指令或 MC_Stop 指令停止。

(8) Scaling Factor (缩放因子)

可以指定一个缩放因子，以扩大或缩小指定的凸轮表的主轴相位和从轴的位移。主轴和从轴可单独设置缩放因子。

(9) Offset (偏移)

可以对指定凸轮表的相位和位移进行偏置移动。主轴相位和从轴位移的偏移可单独设置。

(10) ReferenceType (位置类型选择)

可以为主轴选择以下位置类型，使从轴与其同步。

1) _mcCommand: 命令位置 (值在上一个主周期中计算)

当前周期使用上一个主周期计算出的主轴命令位置。该命令值用于当前周期计算命令位置。

2) _mcFeedback: 值在同一主周期中得到

使用在同一主周期中得到的主轴实际位置。

3) _mcLatestCommand: 命令位置 (值在同一主周期中计算)

使用在同一主周期中计算得到的主轴命令位置。这样就可以使用比 _mcCommand 更新的信息。然而，主轴的轴号必须设置低于从轴的轴号。如果从轴的轴号低于主轴的轴号，Error 会变为 TRUE。一个主轴/从轴号非升序排列错误 (错误代码: 5438H)，将输出到 ErrorID。

(11) 轴类型和位置类型之间的关系

轴类型和位置类型之间的关系见表 7-38。

表 7-38 轴类型和位置类型之间的关系

轴 类 型	ReferenceType	
	_mcCommand 或 _mcLatestCommand	_mcFeedback
伺服轴	OK	OK
编码器轴	No*	OK
虚拟伺服轴	OK	OK
虚拟编码器轴	No*	OK

* 当指令执行时，一个位置类型选择超出范围错误 (错误代码: 5438H) 会发生。

(12) Direction (方向)

只有当主轴的移动方向与 Direction 的设置相同时，才能对从轴启动凸轮操作。只有当 InSync 为 TRUE 时，Direction 才是有效的。

Direction 有 3 个选择: _mcNoDirection、_mcPositiveDirection 和 _mcNegativeDirection。

(13) Index (索引)

在两个用来寻找主轴和从轴命令位置的凸轮数据中，具有较小的凸轮数据索引号的输出给 Index 输出变量。该值用于凸轮编辑器或用户程序微调凸轮数据。

(14) 定位检查

MC_CamIn 指令不做定位检查。

(15) 倍率因子

不能用 MC_SetOverride 指令为这条指令设置倍率因子。

(16) 时序图

1) 非周期操作

图 7-74 显示了当 MC_CamIn 指令的 Periodic 是 FALSE 时的非周期操作。

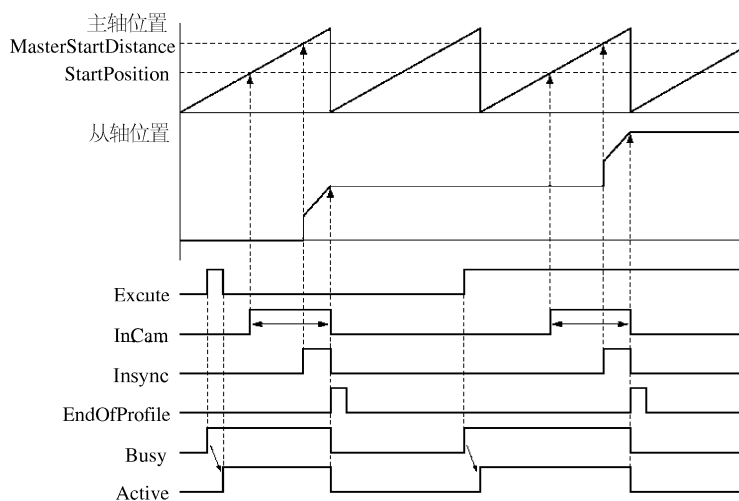


图 7-74 非周期操作时序图

2) 周期操作

图 7-75 显示了当 MC_CamIn 指令的 Periodic 是 TRUE 时的周期操作，凸轮运动重复执行。当 MC_CamOut (结束凸轮操作) 指令结束凸轮操作，从轴减速到 0。主轴的操作不受影响。

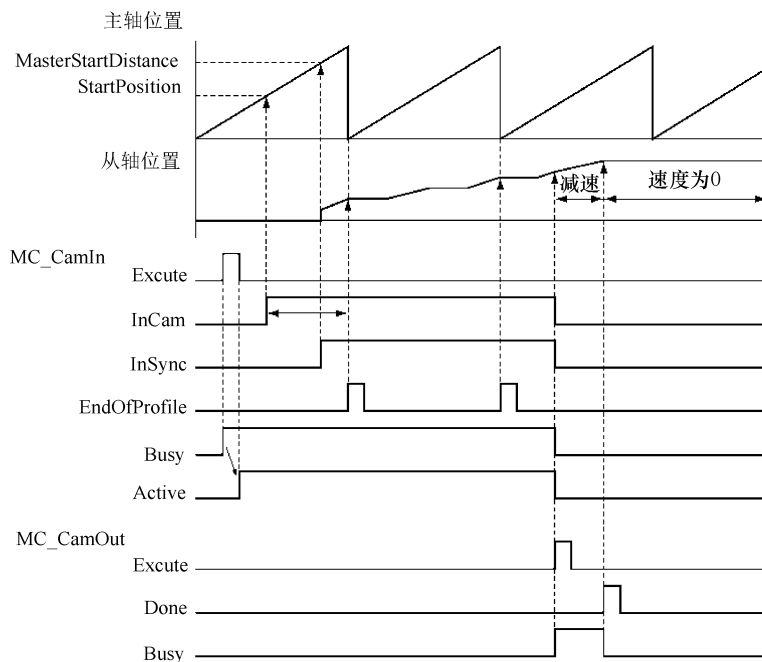


图 7-75 周期操作时序图

(17) 指令的重新执行

如图 7-76 所示，在凸轮操作期间，可以切换凸轮表重新执行这条指令。切换后，对于重新执行的指令，用指定的凸轮数据完成凸轮操作。根据新的凸轮表，从 MC_CamIn 指令输

出 EndOfProfile 和 Index。

使用之前，要切换的凸轮表必须存储在非易失性内存中。

在凸轮运行期间，切换凸轮表会引起不连续的速度。调整切换凸轮表的时序，避免过度的速度不连续。

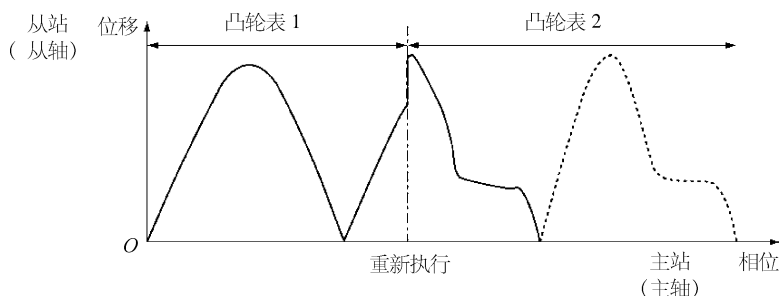


图 7-76 重新执行

(18) 指令的多次执行

MC_CamIn 指令可以多次执行。

4. 从 Sysmac Studio 创建凸轮表

(1) 登记凸轮数据设置

如图 7-77 所示，在多视图浏览器中的“配置与设置”下，右击“Cam 数据设置”，从菜单中选择“添加-CamProfile”，则添加了凸轮表 CamProfile0，如图 7-78 所示。



图 7-77 时序图



图 7-78 添加了凸轮表 CamProfile0

(2) 编辑凸轮数据设置

1) 启动凸轮编辑器

如图 7-79 所示，右击要编辑的凸轮数据 CamProfile0，从菜单中选择“编辑”，显示凸轮编辑器，如图 7-80 所示。



图 7-79 编辑

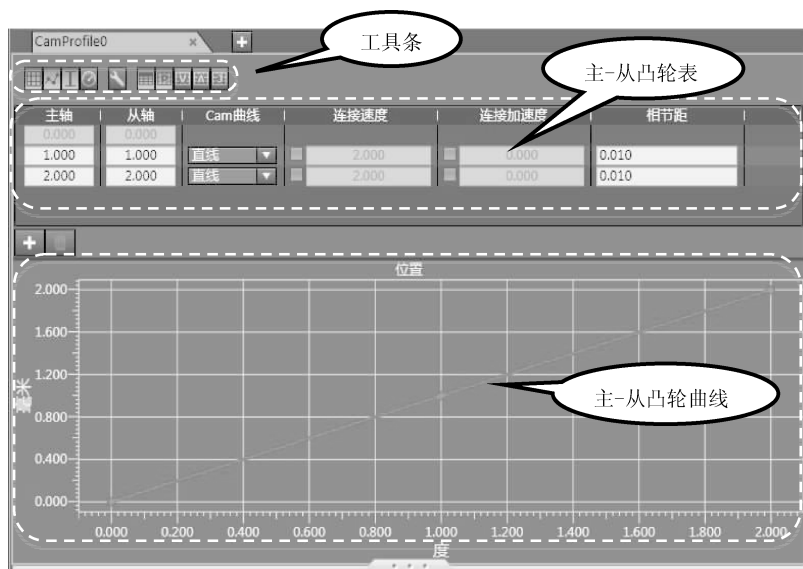


图7-80 凸轮编辑

凸轮编辑器的顶部工具条如图 7-81 所示。

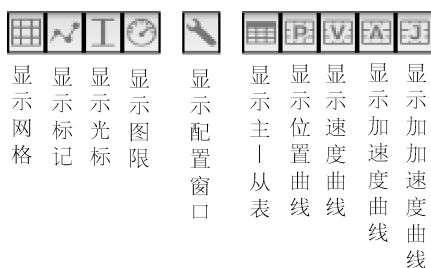


图 7-81 工具条

点击工具条中的图标，显示参数配置窗口，如图 7-82 所示。

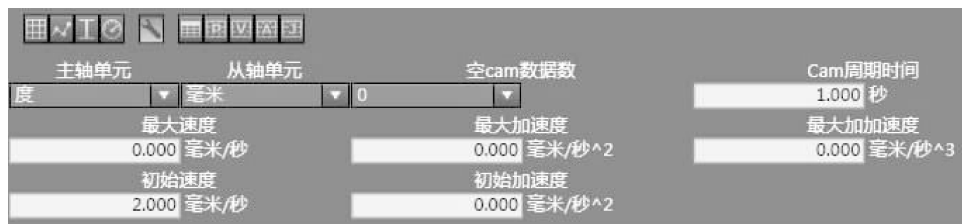


图 7-82 参数配置窗口

凸轮参数配置见表 7-39。

表 7-39 凸轮参数配置

参 数	范围/单位	描 述
主轴单元单位	Deg、inch、mm、pulse、 μm 或 nm	设置用作主轴的单位

(续)

参 数	范围/单位	描 述
从轴单元单位	Deg、inch、mm、pulse、 μm 或 nm	设置用作从轴的单位
空 Cam 数据数	0、10、50、100、500、1000、5000 或 10000	在凸轮表的终点之后,插入主轴相位、从轴位移数据,数据都设为 0,设置插入数据的数量
Cam 周期时间	—	为一个凸轮循环设置时间
最大速度	—	设置从轴的最大速度。设置值在速度曲线图上以蓝色点划线显示
最大加速度	—	设置从轴的最大加速度。设置值在加速度曲线图上以蓝色点划线显示
最大加加速度	—	设置从轴的最大加加速度。设置值在加加速度曲线图上以蓝色点划线显示
初始速度	—	设置当从轴的相位是 0 时的速度。如果相位 0 后跟着一条多项式 3、多项式 5 或自由曲线,可以指定初始速度
初始加速度	—	设置当从轴的相位是 0 时的加速度。当相位 0 后跟着一条多项式 5 曲线,可以指定初始加速度

Cam 曲线设置如图 7-83 所示。

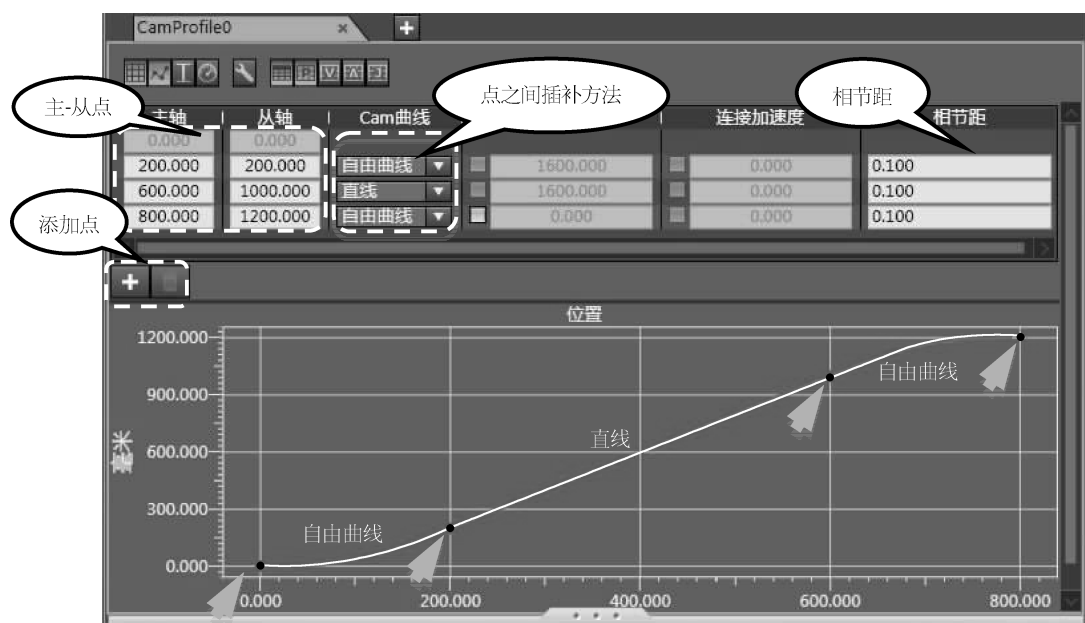


图 7-83 Cam 曲线设置

一旦设置了主-从曲线,可以观察得到的速度、加速度和加加速度曲线,检查各种物理量的极限值。

点击工具条中的图标,显示速度曲线,如图 7-84 所示。

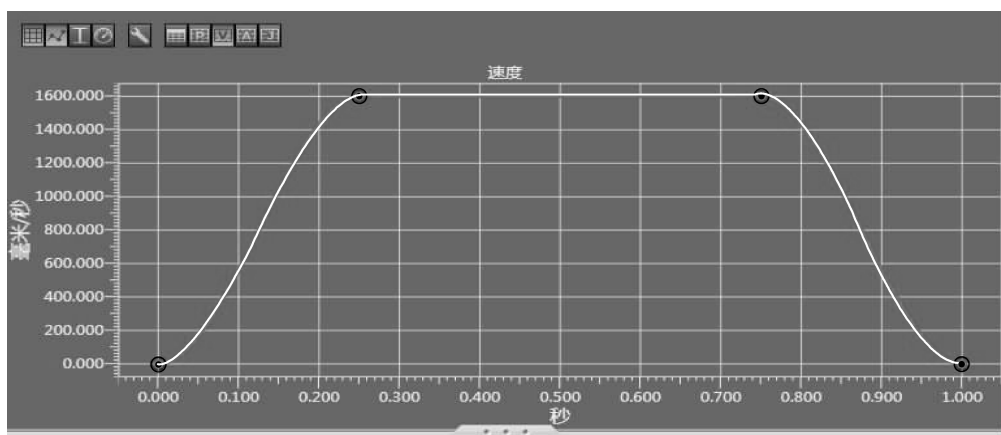


图 7-84 速度曲线

点击工具条中的图标, 显示加速度曲线, 如图 7-85 所示。

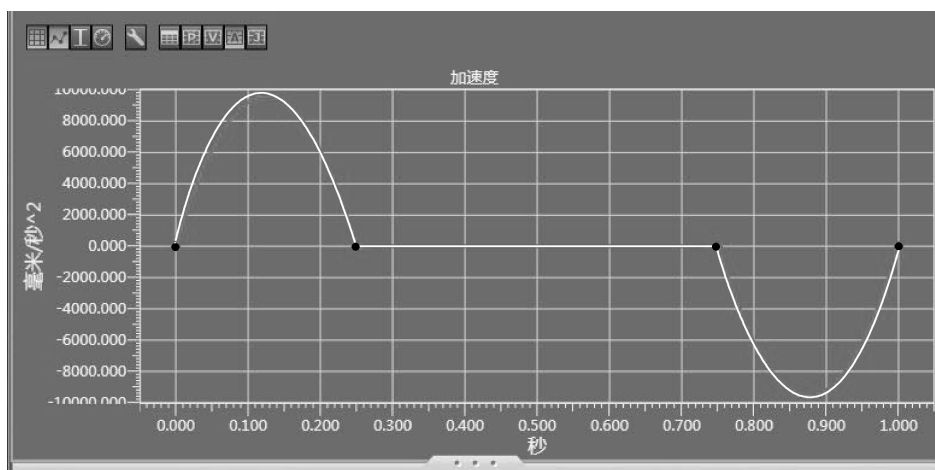


图 7-85 加速度曲线

点击工具条中的图标, 显示加加速度曲线, 如图 7-86 所示。

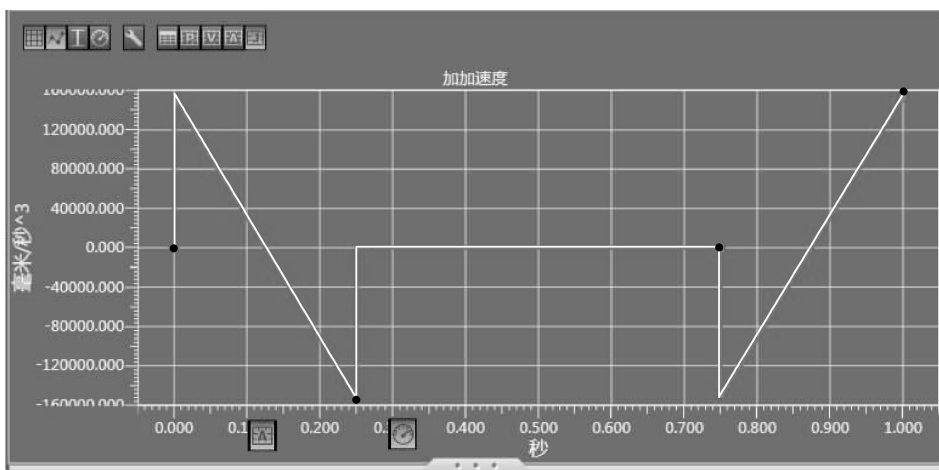


图 7-86 加加速度曲线

点击工具条中的图标，再点击，检查加速度极限值，如图 7-87 所示，

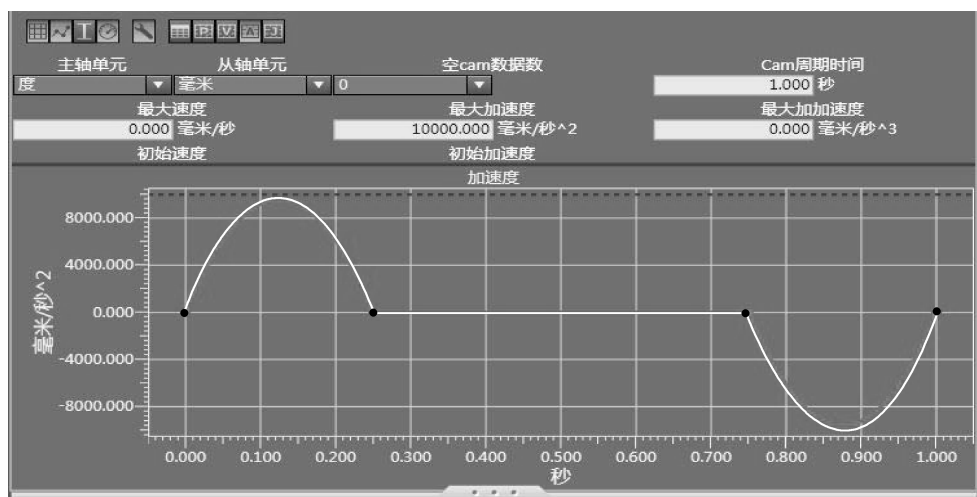


图 7-87 检查加速度极限值

同样可以检查速度、加加速度的极限值。

例 7.2 进行凸轮周期性操作，主轴为轴 1，从轴为轴 2。

(1) 设置轴参数

1) 轴类型：

轴 1：伺服轴（主轴）；

轴 2：伺服轴（从轴）。

2) 计数模式：

轴 1：环形模式；

轴 2：线性模式。

3) 环形计数器（模最大位置、模最小位置）：

轴 1：360、0。

4) 显示单位：

轴 1：degree；

轴 2：mm。

(2) 操作要求

如图 7-88 所示，进行周期性凸轮操作。

操作过程如下。

1) 启动凸轮操作：如果凸轮表的起始位置（零相位位置）为 20° ，当主轴达到一个和该点相对角度 40° 的位置时，从轴启动操作。Periodic 为 TRUE，凸轮操作周期性进行。

2) 结束凸轮操作：如果从轴 MC_Axis001.Act.Pos 实际位置超出 1000.0，凸轮操作结束，并且从轴以减速度 DecRate2 停止。

(3) 主要变量

主要变量见表 7-40。

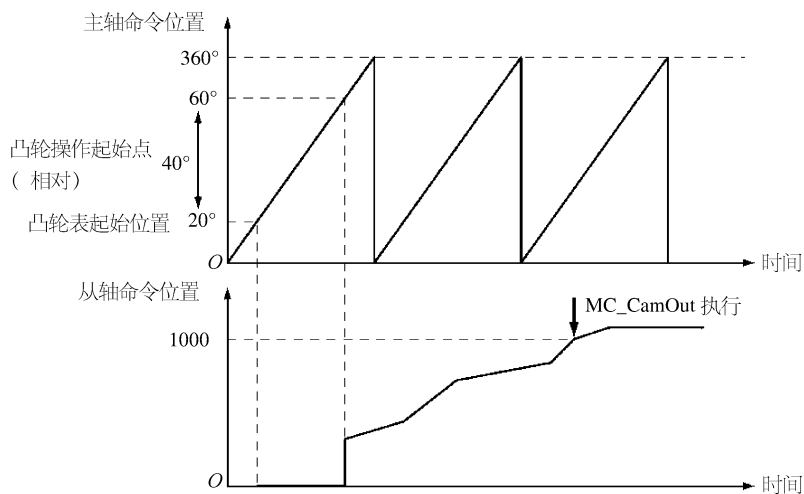


图 7-88 操作要求

表 7-40 主要变量

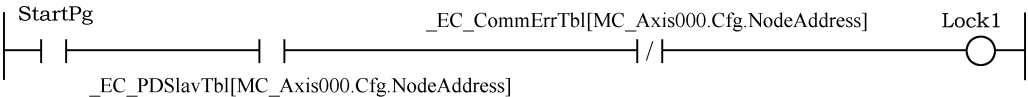
名 称	数 据 类 型	默认值	注 释
MC_Axis000	_sAXIS_REF	—	主轴的轴变量,轴 1
MC_Axis000. MFaultLvl. Active	BOOL	FALSE	当轴 1 出现轻微故障级错误时为 TRUE
MC_Axis000. Details. Homed	BOOL	FALSE	当轴 1 原点定义时为 TRUE
MC_Axis001	_sAXIS_REF	—	从轴的轴变量,轴 2
MC_Axis001. MFaultLvl. Active	BOOL	FALSE	当轴 2 出现轻微故障级错误时为 TRUE
MC_Axis001. Details. Homed	BOOL	FALSE	当轴 2 原点定义时为 TRUE
Pwr1_S	BOOL	FALSE	为指令 MC_Power 的例程 PWR1 分配 Statue 输出变量。当伺服为 ON 时,该变量为 TRUE
Pwr2_S	BOOL	FALSE	为指令 MC_Power 的例程 PWR2 分配 Statue 输出变量。当伺服为 ON 时,该变量为 TRUE
CamProfile0	ARRAY[0..360] OF _sMC_CAM_REF	—	凸轮数据变量。用凸轮编辑器设定数组元素 ARRAY[0..N]。在此示例中,应用 0~360,但是数组元素的个数取决于凸轮编辑器的设定
DecRate2	LREAL	10000.0	该变量设定 MC_CamOut 执行时的减速度
Vel_InVel	BOOL	FALSE	为 MC_MoveVelocity 指令的例程 VEL 分配 InVelocity 输出变量。当到达目标速度时为 TRUE
Camin_InCam0	BOOL	FALSE	为指令 MC_CamIn 的例程 CAMIN 分配 InCam 输出变量。在凸轮执行期间为 TRUE

(续)

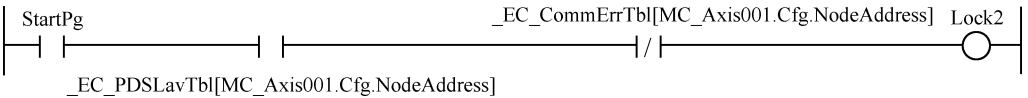
名 称	数 据 类 型	默认值	注 释
Camout_Ex	BOOL	FALSE	当该变量为 TRUE 时, 执行 MC_CamOut 指令的 CAMOUT 例程
StartPg	BOOL	FALSE	如果该变量为 TRUE 并且建立了 EtherCAT 数据通信, 伺服为 ON

(4) 梯形图

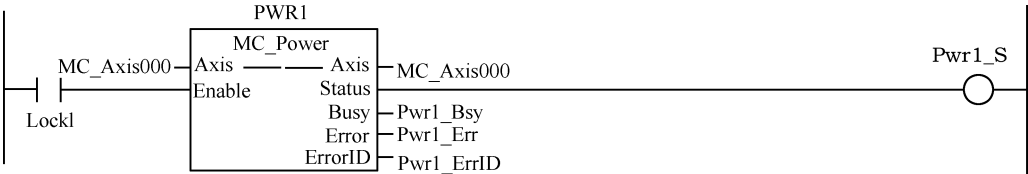
如果 StartPg 为 TRUE, 检查轴 1 的 EtherCAT 通信, 确认过程数据通信是否正常。



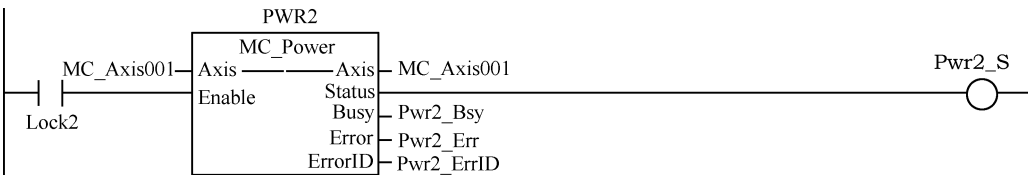
如果 StartPg 为 TRUE, 检查轴 2 的 EtherCAT 通信, 确认过程数据通信是否正常。



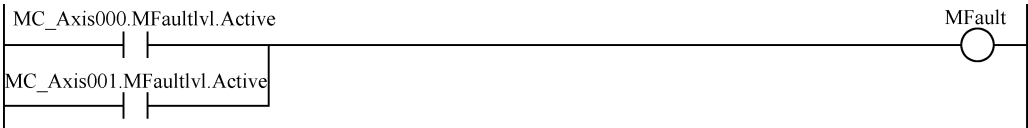
如果轴 1 的过程数据通信激活, Servo 变为 ON。



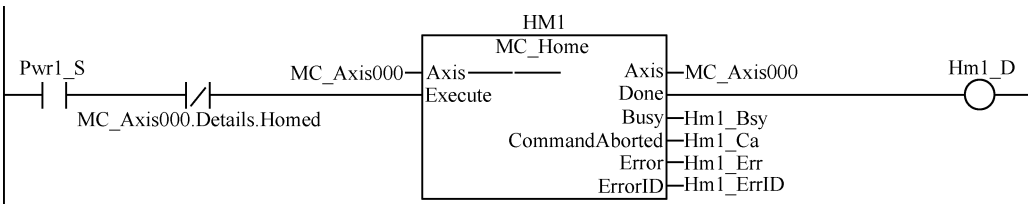
如果轴 2 的过程数据通信激活, Servo 变为 ON。



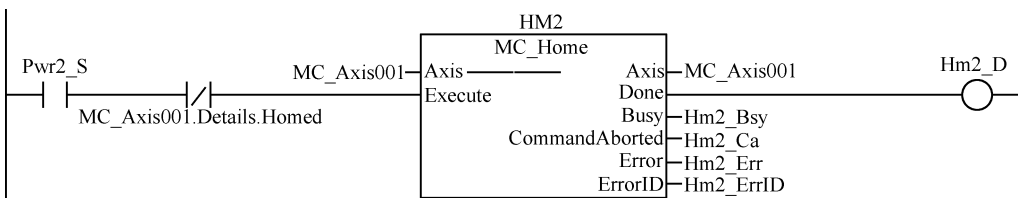
如果组合轴有一个轻微故障级错误, 变量 MFault 为 ON。



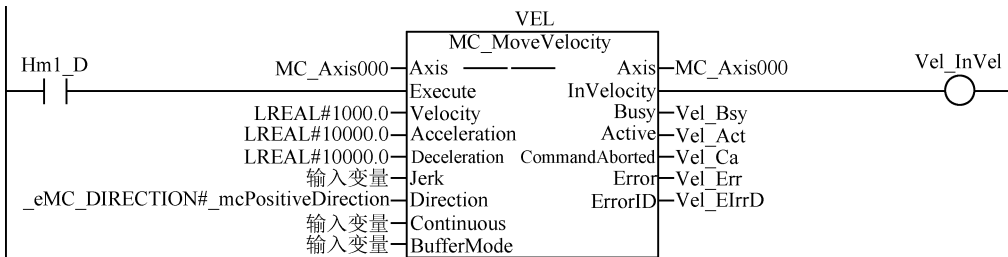
如果轴 1 的 Servo 为 ON 但原点没有定义, 执行 MC_Home 指令以定义原点。



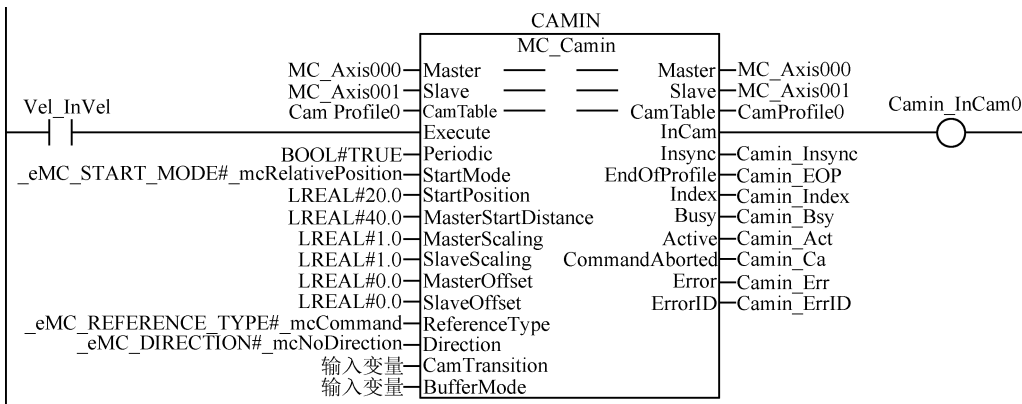
如果轴 2 的 Sevro 为 ON 但原点没有定义, 执行 MC_Home 指令以定义原点。



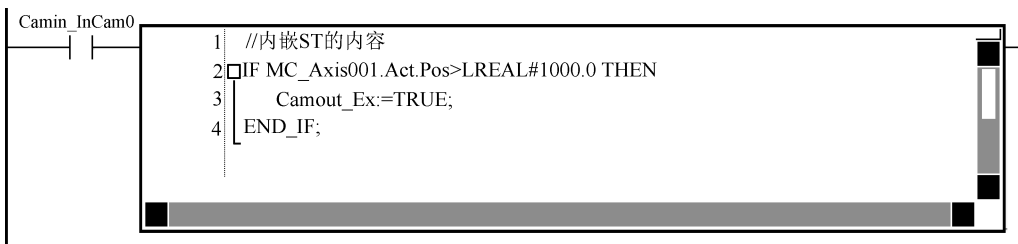
如果轴 1 的原点返回执行完毕, 执行 MC_MoveVelocity 指令。



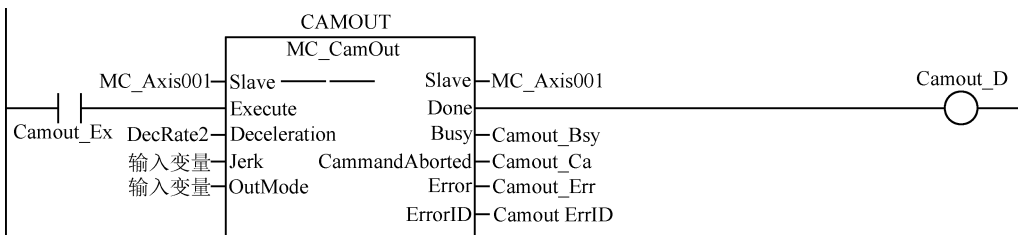
如果 MC_MoveVolecity 指令的 Vel_InVel 变为 TRUE, 则执行 MC_CamIn 指令。



如果 Camin_InCam0 为 TRUE 和 MC_Axis001. Act. Pos 超过 1000, CamOut_Ex 变为 TRUE。



当 CamOut_Ex 变为 TRUE 时, 执行 MC_CamOut 指令。



(5) 时序图

图 7-89 为凸轮操作的时序图。

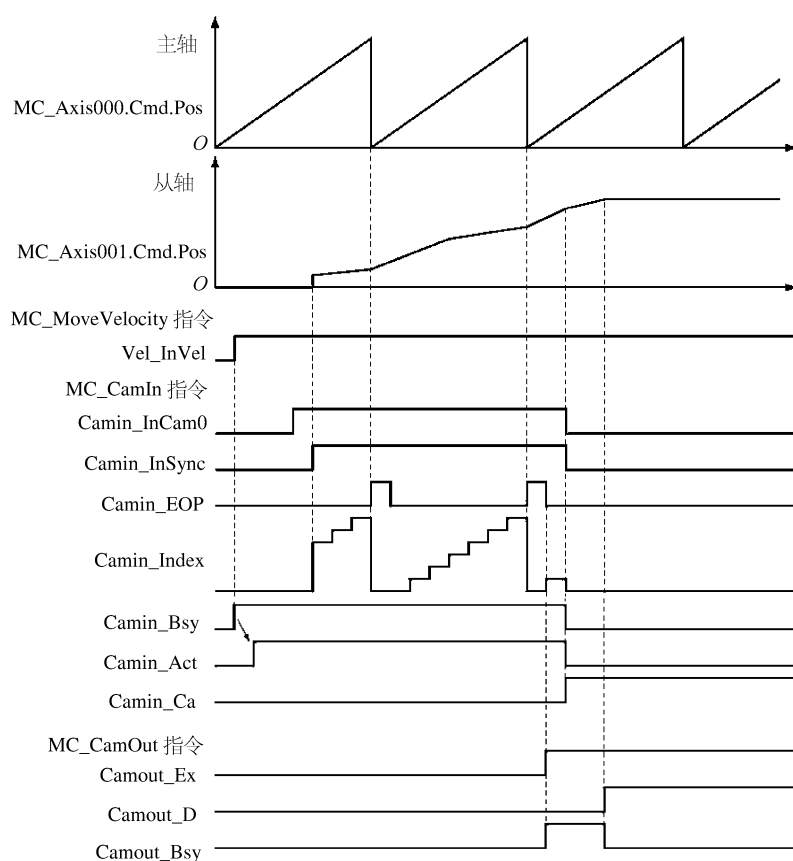


图 7-89 凸轮操作的时序图

7.7.9 结束凸轮操作指令 MC_CamOut

1. 指令的梯形图

MC_CamOut 指令的梯形图如图 7-90 所示。

输入变量 Jerk (保留): 加加速度, LREAL 型, 取值为 0。

输入变量 OutMode (保留): 同步结束模式选择, 枚举体_eMC_OUT_MODE 型, 取值为 _mcStop。

2. 功能

MC_CamOut 指令禁用从轴的凸轮操作。

当 Execute 变为 TRUE 时, 以 Deceleration 指定的减速度, 轴减速至 0。

当命令速度到达 0 时, 指令执行结束。

如果对不进行凸轮操作的轴执行这个指令, 会发生错误。

该指令不能重新执行, 但可以多次执行。

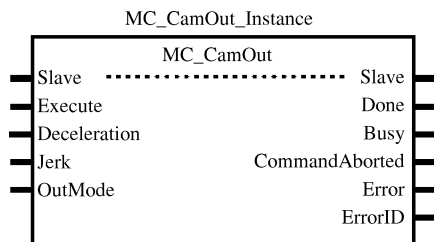


图 7-90 MC_CamOut 指令的梯形图

7.7.10 启动齿轮操作指令 MC_GearIn

1. 指令的梯形图

MC_GearIn 指令的梯形图如图 7-91 所示。

输入变量 RatioNumerator: 齿轮比分子, UINT 型变量, 正整数, 默认值为 10000。指定主轴和从轴之间电子齿轮的分子。

输入变量 RatioDenominator: 齿轮比分母, UINT 型变量, 正整数, 默认值为 10000。指定主轴和从轴之间电子齿轮的分母。

输入变量 ReferenceType: 位置类型选择, 枚举体_eMC_REFERENCE_TYPE 型变量, 有效取值范围:

- 0: _mcCommand
- 1: _mcFeedback
- 2: _mcLatestCommand

默认值为 0。

输入变量 Jerk (保留): 加加速度, LREAL 型变量, 取值为 0。

输入变量 BufferMode: 缓冲模式选择, 枚举体_eMC_BUFFER_MODE 型变量, 有效取值范围:

- 0: _mcAborting
- 1: _mcBuffered

默认值为 0。

输出变量 InGear: 齿轮比到达, BOOL 型变量, 取值有效范围为 TRUE 或 FALSE。为 TRUE 时, 表示从轴到达目标速度。

2. 功能

MC_GearIn 指令完成指定从轴的齿轮操作。下列参数需要指定: RatioNumerator、RatioDenominator、ReferenceType、Acceleration 和 Deceleration。

齿轮操作如图 7-92 所示, 对于主轴, 可以指定命令位置、实际位置和上一次命令位置。

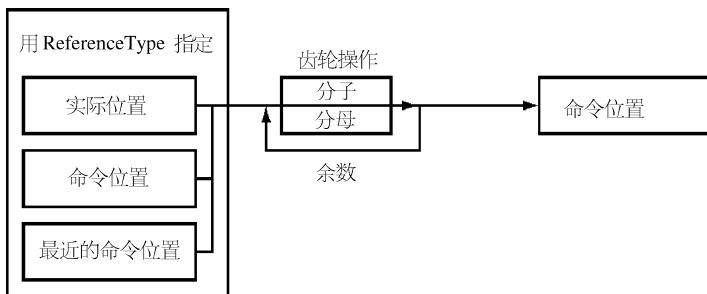


图 7-92 齿轮操作

齿轮操作过程如图 7-93 所示, 运行启动后, 从轴使用主轴速度乘以齿轮比为它的目标

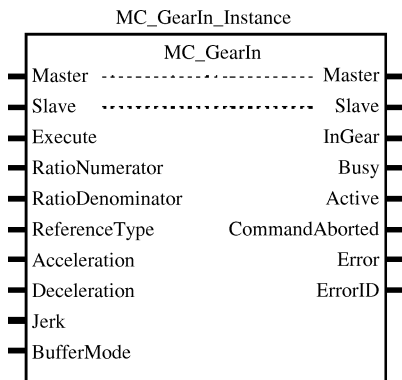


图 7-91 MC_GearIn 指令的梯形图

速度，相应地进行加速/减速。从轴处于追赶阶段，直到到达从轴同步启动位置。当到达从轴同步启动位置时，从轴进入 InGear (齿轮同步) 阶段。此后，从轴位置与主轴同步。

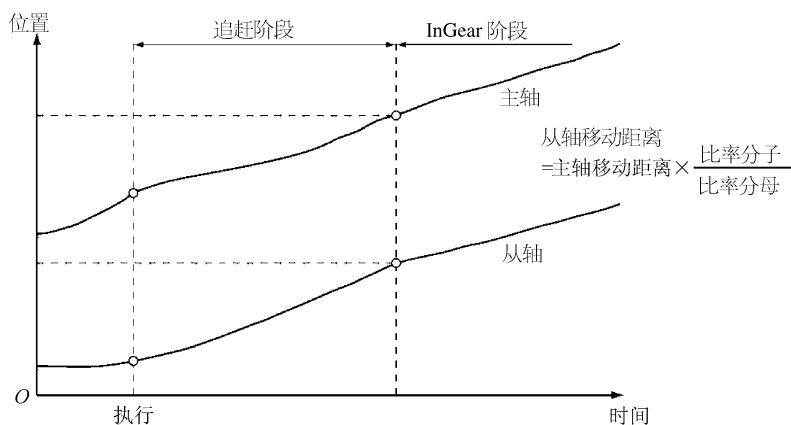


图 7-93 齿轮操作过程

时序图如图 7-94 所示。

当 Execute 变为 TRUE，Busy 同时变为 TRUE。Active 下一个周期变为 TRUE。

当目标速度到达时，InGear 变为 TRUE。

如果另一条指令中止这条指令，CommandAborted 变为 TRUE，Busy、Active 和 InGear 就变为 FALSE。

使用 MC_GearOut 或 MC_Stop 指令停止电子齿轮操作。

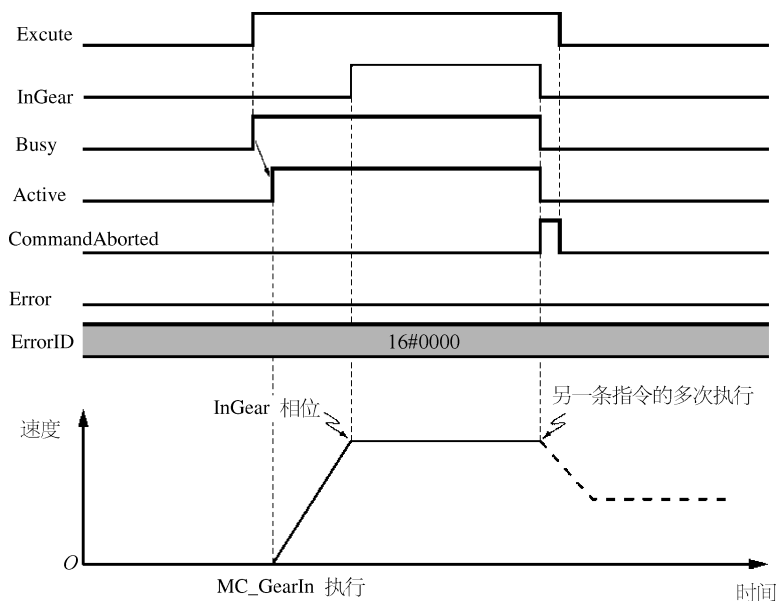


图 7-94 MC_GearIn 指令的时序图

MC_GearIn 指令可以重新执行，也可以多次执行。

例 7.3 如图 7-95 所示, 轴 1 为主轴, 轴 2 为从轴, 进行齿轮同步控制。

(1) 参数设置

1) 轴类型:

轴 1: 伺服轴 (主轴);

轴 2: 伺服轴 (从轴)。

2) 计数模式:

轴 1: 环形模式;

轴 2: 环形模式。

3) 环形计数器 (模最大位置、模最小位置):

轴 1: 3600、0;

轴 2: 3600、0。

4) 显示单位:

轴 1: 毫米;

轴 2: 毫米。

(2) 操作过程

1) 启动主轴: 主轴 (轴 1) 是一个实际伺服轴并且执行速度控制。

2) 执行从轴: 当主轴的实际速度达到目标速度, 执行齿轮操作, 使轴 2 (从轴) 与主轴的实际位置的齿轮比为 1:2。

3) 停止从轴: 如果主轴实际位置 MC_Axis000.Act.Pos 超出 1000.0, 轴 2 的齿轮操作结束, 以减速度 DecRate 减速停止。

(3) 主要变量

主要变量见表 7-41。

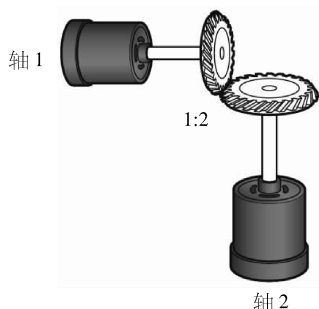


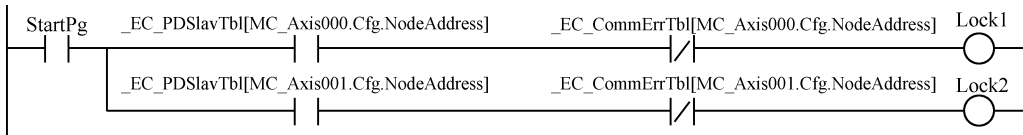
图 7-95 齿轮操作

表 7-41 主要变量

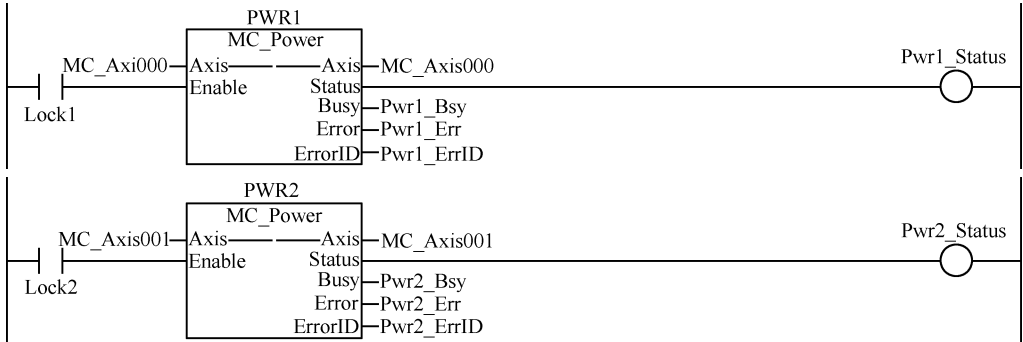
名 称	数 据 类 型	默 认 值	注 释
MC_Axis000	_sAXIS_REF	—	主轴的轴变量, 轴 1
MC_Axis000.Act.Pos	LREAL	—	该变量给出轴 1 的实际当前位置
MC_Axis001	_sAXIS_REF	—	从轴的轴变量, 轴 2
Pwr1_Status	BOOL	FALSE	为指令 MC_Power 的例程 PWR1 分配 Status 输出变量。当伺服为 ON 时, 这个变量为 TRUE
Pwr2_Status	BOOL	FALSE	为指令 MC_Power 的例程 PWR2 分配 Status 输出变量。当伺服为 ON 时, 这个变量为 TRUE
StartPg	BOOL	FALSE	如果该变量为 TRUE 并且建立了 EtherCAT 数据通信, 伺服为 ON
Vel_InVel	BOOL	FALSE	为 MC_MoveVelocity 指令的例程 VEL 分配 InVelocity 输入变量。当到达目标速度时为 TRUE
Gearin1_Act	BOOL	FALSE	为 MC_GearIn 指令的例程 GEARIN1 分配 Active 输出变量。在 GEARIN1 执行期间为 TRUE
Gearout_Ex	BOOL	FALSE	当该变量为 TRUE 时, 执行 MC_GearOut 指令的 GEAROUT 例程

(4) 梯形图

如果 StartPg 为 TRUE, 检查 EtherCAT 通信, 确认过程数据通信是否正常。



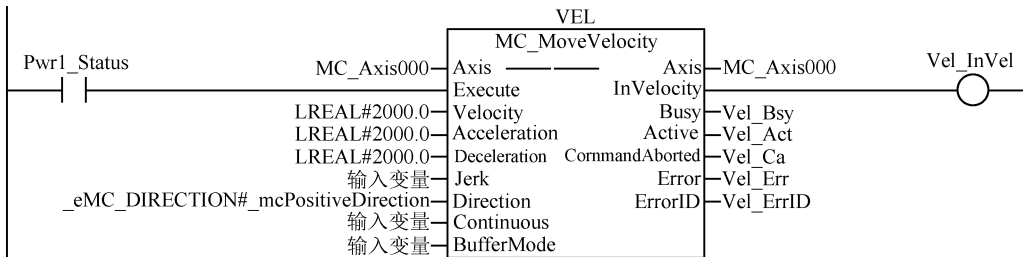
如果过程数据通信激活, 对应轴的 Servo 变为 ON。



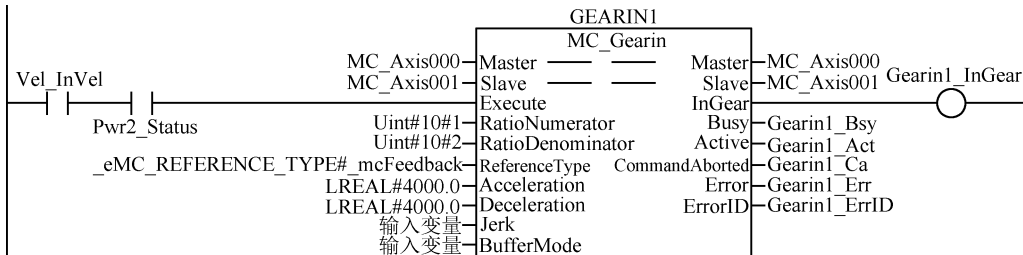
如果组合轴有一个轻微故障级错误, 变量 MFault 为 ON。



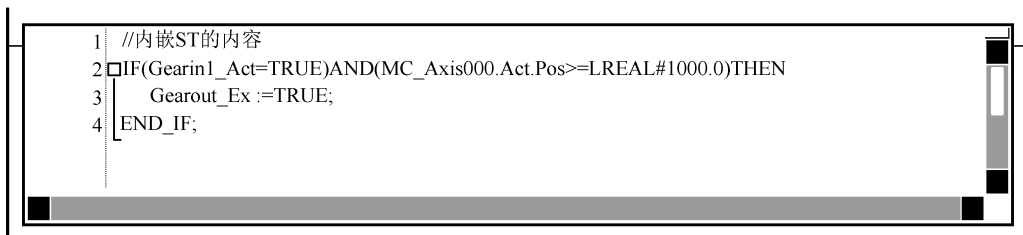
主轴 (轴 1) 的伺服变为 ON 后, 执行 MC_MoveVelocity 指令



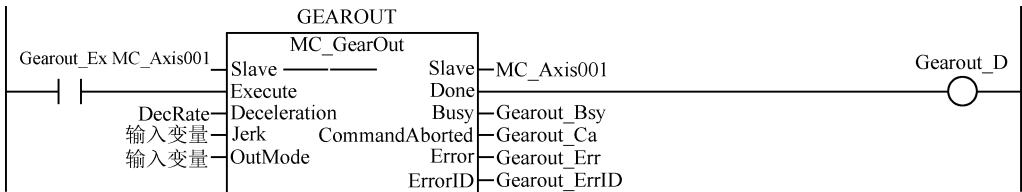
当主轴 (轴 1) 的实际速度达到目标速度时, 执行齿轮操作, 使轴 2 (从轴) 的齿轮比是 1:2。



轴 2 (从轴) 齿轮操作期间, 当主轴的实际位置是 1000.0 或更高时, Gearout_Ex 变为 TRUE。



当 Gearout_Ex 变为 TRUE 时，轴 2 (从轴) 的齿轮操作停止。轴减速到停止。



(5) 时序图

图 7-96 为时序图。

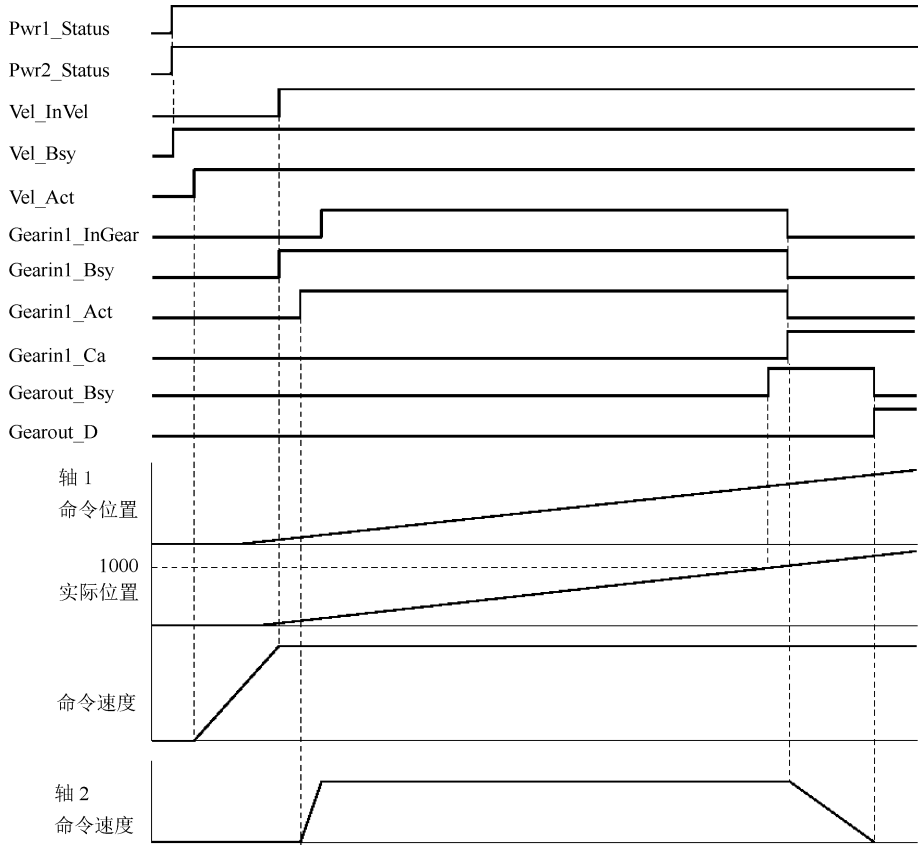


图 7-96 时序图

7.7.11 结束齿轮操作指令 MC_GearOut

1. 指令的梯形图

MC_GearOut 指令的梯形图如图 7-97 所示。

输入变量 Jerk (保留): 加加速度, LREAL 型变量, 取值为 0。

输入变量 OutMode (保留): 同步结束模式选择, 枚举体变量_eMC_OUT_MODE, 取值为_mcStop。

2. 功能

MC_GearOut 指令对于 Slave 指定的轴, 以指定的

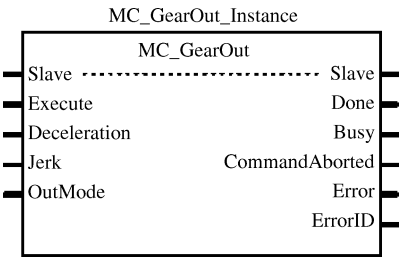


图 7-97 MC_GearOut 指令的梯形图

减速度 Deceleration，停止 MC_GearIn 指令或 MC_GearInPos 指令的操作。

这条指令不影响 MC_GearIn 指令或 MC_GearInPos 指令的主轴操作。

这条指令不能重新执行，但可以多次执行。

7.8 轴组运动控制指令

7.8.1 概述

轴组运动控制指令见表 7-42。

表 7-42 轴组运动控制指令

分 类	指 令	名 称	功 能
多轴协调控制	MC_MoveLinear	直线插补	执行直线插补
	MC_MoveLinearAbsolute	绝对直线插补	指定绝对位置,执行直线插补
	MC_MoveLinearRelative	相对直线插补	指定相对位置,执行直线插补
	MC_MoveGroup2D	2D 圆弧插补	执行 2 轴的圆弧插补
多轴协调控制的辅助功能	MC_GroupReset	复位轴组错误	清除轴组和轴的错误
	MC_GroupEnable	轴组使能	使能轴组
	MC_GroupDisable	轴组禁用	禁用轴组
	MC_GroupStop	轴组停止	所有插补运行的轴减速,直到停止
	MC_GroupImmediateStop	轴组立即停止	所有插补运行的轴立即停止
	MC_GroupSetOverride	设置轴组倍率因子	改变插补运行过程中的目标速度

轴组指令使用的变量很多与单轴指令的相同，介绍单轴指令时已对变量做了统一说明，因此，下面介绍轴组指令的变量时，不再重复在单轴指令中已做了说明的变量。

1. 输入变量

常用的输入变量有：Execute、Position、Velocity、Acceleration、Deceleration、Jerk、CoordSystem、BufferMode、TransitionMode、MoveMode 等。

轴组指令部分常用的输入变量见表 7-43。

表 7-43 轴组指令部分常用的输入变量

名 称	含义	数 据 类 型	取 值 范 围	默认值	说 明
Execute	执行	BOOL	TRUE 或 FALSE	FALSE	Execute 变为 TRUE 时,指令执行
CoordSystem*	坐标系	_eMC_ COORD_ SYSTEM	0:_mcACS	0	指定坐标系: 0:轴坐标系(ACS)
BufferMode*	Buffer 模式选择	_eMC_ BUFFER_ MODE	0:_mcAborting 1:_mcBuffered 2:_mcBlendingLow 3:_mcBlendingPrevious 4:_mcBlendingNext 5:_mcBlendingHigh	0	当执行一条以上的运动指令时, 指定操作: 0:Aborting 1:Buffered 2:Blend with low 3:Blend with previous 4:Blend with next 5:Blend with high

(续)

名 称	含义	数 据 类 型	取 值 范 围	默认值	说 明
Transition-Mode*	过渡模式	_eMC_TRANSITION_MODE	0:_mcTMNone 10:_mcTMComerSuperimpose	0	指定运行的路径: 0:过渡禁用 10:叠加拐弯
MoveMode*	移动模式	_eMC_MOVE_MODE	0:_mcAbsolute 1:_mcRelative	0	指定移动方法 0:绝对定位 1:相对定位

* 对枚举体变量, 实际指定时用枚举体的元素, 不用它对应的数值。

2. 输出变量

常用输出变量有: Done、Busy、Active、CommandAborted、Error 和 ErrorID。

3. 输入-输出变量

AxesGroup: 轴组, 结构体_sGROUP_REF 型变量, 指定操作的轴组。

7.8.2 轴组使能指令 MC_GroupEnable

1. 指令的梯形图

MC_GroupEnable 指令的梯形图如图 7-98 所示。

2. 功能

MC_GroupEnable 指令使指定的轴组处于使能状态。

当轴组处于使能状态时, 可以对轴组执行任何多轴协调控制指令。

在轴组中只能设置伺服轴和虚拟伺服轴。如果包括其他类型的轴, 会出现错误。

使能一个轴组时, 属于同一个轴组所有的轴必须处在停止状态。如果轴变量中的 Status.Disabled 或 Status.Standstill 是 TRUE, 则轴处于停止状态。

如果有轴已经属于另一个轴组, 而其他轴组已使能, 则不能执行 MC_GroupEnable 指令。若试图执行, 则会发生错误。

当一个轴组使能, 轴组中的轴处于协调运动状态, 轴变量中的 Status.Coordinated 变为 TRUE。

如果 MC_GroupDisable 指令被执行, 或者操作模式改为 PROGRAM 后运行停止, 或者启动 MC 试运行, 则轴组被禁用。

图 7-99 为 MC_GroupEnable 指令的时序图。

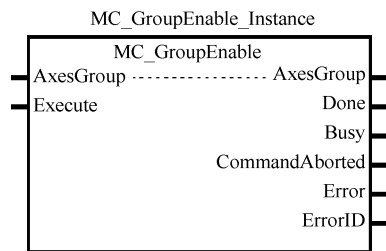


图 7-98 MC_GroupEnable 指令的梯形图

7.8.3 轴组禁用指令 MC_GroupDisable

1. 指令的梯形图

MC_GroupDisable 指令的梯形图如图 7-100 所示。

2. 功能

MC_GroupDisable 指令禁用一个轴组。

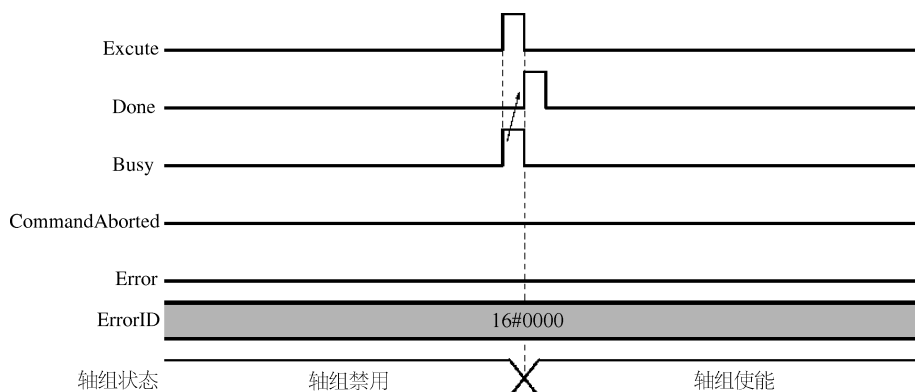


图 7-99 MC_GroupEnable 指令的时序图

MC_GroupDisable 指令使指定的轴组处于禁用状态，当轴组处于禁用状态，轴组停止回应多轴协调控制指令。

对于指定的轴组，当其状态变为禁用时，会清除任何缓冲指令。

即使当操作停止时，轴组也是禁用的。

当轴组禁用时，轴组中轴的状态 Status.Coordinated 从 TRUE 变回各自轴的状态。使用轴变量中的状态（轴状态）来确定每个轴的状态。

图 7-101 为 MC_GroupDisable 指令的时序图。

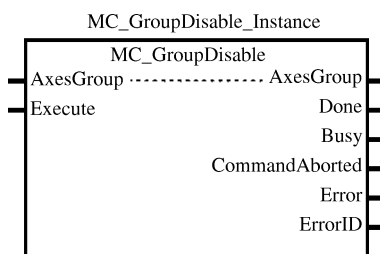


图 7-100 MC_GroupDisable 指令的梯形图

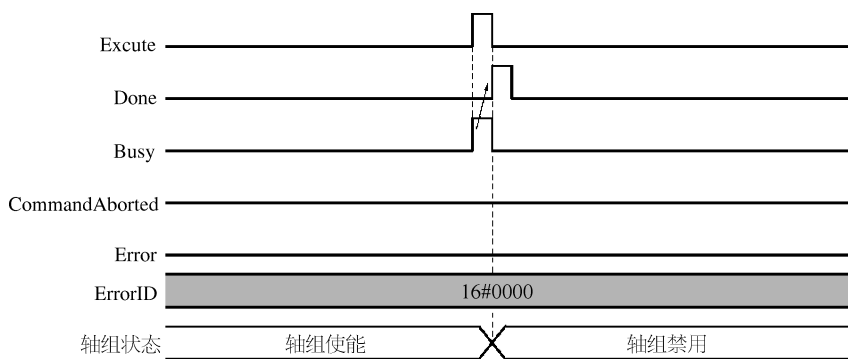


图 7-101 MC_GroupDisable 指令的时序图

处于多轴协调控制下的轴组，执行 MC_GroupDisable 指令时，多轴协调控制指令的 CommandAborted 将变为 TRUE，任何缓冲的多轴协调控制指令的 CommandAborted 也将变为 TRUE。如果轴在运动，每个轴将以最大减速度减速停止。

7.8.4 轴组停止指令 MC_GroupStop

1. 指令的梯形图

MC_GroupStop 指令的梯形图如图 7-102 所示。

2. 功能

这个指令将所有插补指令中运动的轴，从实际的速度减速到 0，然后禁用轴组命令。它可以用于以下插补指令：MC_MoveLinearAbsolute、MC_MoveLinearRelative、MC_MoveCircular2D。

执行这条指令时，正处在操作状态的插补指令的 CommandAborted 将变为 TRUE。

如果执行这个指令而插补指令正在执行，轴将沿着直线插补或圆弧插补路径减速停止。

当 Execute 变为 TRUE 时，减速停止操作将启动。

指令详情如下：

(1) Deceleration 和 Jerk

设置输入变量 Deceleration 和 Jerk，以指定减速停止时使用的减速度和加加速度。

当 Jerk 设置为 0 时，插补速度以指定的减速度直线下降。

当 Jerk 设置为非 0 值时，减速度呈梯形，插补速度沿 S 型曲线下降。

(2) BufferMode (缓冲模式选择)

BufferMode 指定插补指令、先前的插补指令如何加入到轴运行中。这里，BufferMode 只有一种选择：Aborting，即中断正在执行的指令，转去执行这条指令。

(3) 定位检查

这条指令执行结束后，不做定位检查。

(4) 时序图

图 7-103 为 MC_GroupStop 指令的时序图，先前运行的是线性插补指令。

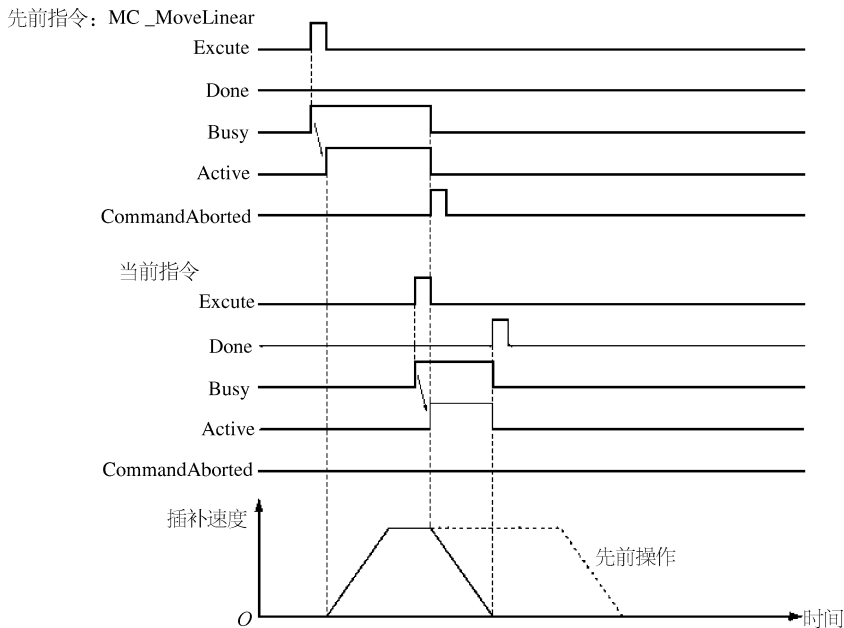


图 7-103 MC_GroupStop 指令的时序图

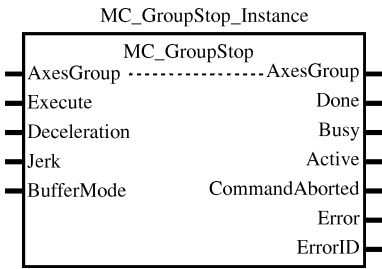


图 7-102 MC_GroupStop 指令的梯形图

当 Execute 变为 TRUE 时, Busy 同时变为 TRUE。Active 在下一个周期变为 TRUE。

当 Velocity 到达 0 时, Done 变为 TRUE。

如果另一条指令中断这条指令, 则 CommandAborted 变为 TRUE, 而 Busy、Active 变为 FALSE。

(5) 运动控制指令的重新执行

这条指令正在执行时, 如果 Execute 再次变为 TRUE 时, 减速度将改变。

(6) 运动控制指令的多次执行

MC_GroupStop 指令可以多次执行。

7.8.5 轴组直线插补指令 MC_MoveLinear

1. 指令的梯形图

MC_MoveLinear 指令的梯形图如图 7-104 所示。

2. 功能

MC_MoveLinear 完成 2~4 轴的直线插补。

对于轴组中的任何构成轴, 如果未定义原点, 则未定义原点的指令执行错误 (错误代码: 5466H) 会出现。

属于轴组的任何逻辑轴, 如果一个轴的限位输入为 ON, 则不能执行直线插补指令。

3. 指令详情

(1) 直线插补的步骤

直线插补的步骤如下:

1) 登记进行插补的轴组

①选择进行插补的轴组, 轴组用 “_MC_GRP[***]” 表示。

②指定轴组的构成, 可以指定 2~4 个轴。

③为轴组指定执行插补的轴的组合。

④为轴分配逻辑轴 (轴 A0~A3), 不是轴号 (轴 0~63)。

⑤按逻辑轴顺序从最低号开始, 为逻辑轴 A0~A3 指定轴号 0~63。

⑥在 Sysmac Studio 的轴组基本设置, 为逻辑轴选择构成轴, 为逻辑轴分配轴号。

如果一个编码器轴或虚拟编码器轴包含在轴组中, 会发生轴类型错误并且操作停止。确保选择的只有伺服轴或虚拟伺服轴。

2) 使能轴组

①执行 MC_GroupEnable 指令来使能登记的轴组。

②使构成轴的伺服 ON, 将 MC_Stop 指令的 Execute 变为 FALSE。

这样, 使用的直线插补指令即被使能。

(2) Position (目标位置)

1) 用 Position 设置轴组中所有构成轴的位置。

2) 必须在 Sysmac Studio 里创建一个 1×4 的数组变量分配给 Position (A0, A1, A2 和 A3), 可以使用任何变量名称。

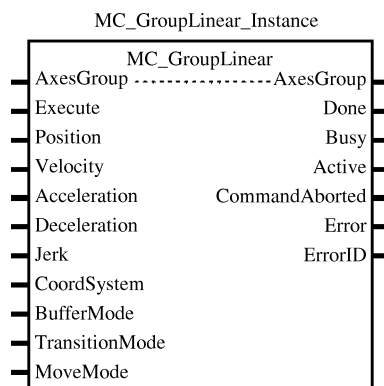


图 7-104 MC_MoveLinear 指令的梯形图

把轴的目标位置分配给该数组里的元素。一般创建一个 1×4 的数组，即使轴组少于 4 个轴。

3) 计数模式设置为环形模式的轴，如果选定为插补的轴，必须指定绝对位置，目标值和 Direction 设定为与“无指定方向”时的一样。

(3) Velocity、Acceleration、Deceleration 和 Jerk

设置 Velocity、Acceleration、Deceleration 和 Jerk，以指定直线插补的速度、加速度、减速度和加加速度。直线插补将插补运动分解为各个轴的运动，图 7-105 是从 A 点到 B 点的两轴直线插补。

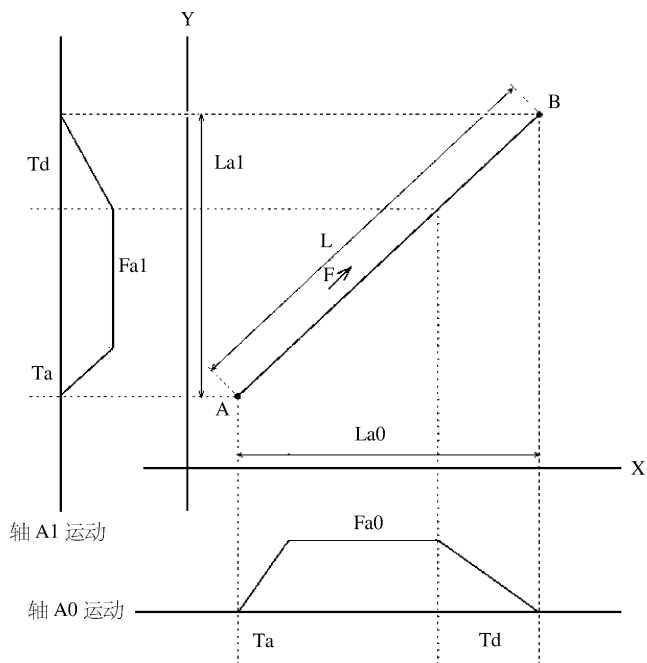


图 7-105 直线插补

对于 4 轴的直线插补，每个轴的插补速度和运行距离决定了目标速度，如下所示。

F: 指定插补进给速度。

Fa0: 轴 A0 基于 F 分解的插补进给速度。

Fa1: 轴 A1 基于 F 分解的插补进给速度。

Fa2: 轴 A2 基于 F 分解的插补进给速度。

Fa3: 轴 A3 基于 F 分解的插补进给速度。

Ta: 插补加速时间。

Td: 插补减速时间。

L: 在指定路径上的移动距离。

La0、La1、La2 和 La3: 轴 A0、轴 A1、轴 A2 和轴 A3 的移动距离。

L、Fa0、Fa1、Fa2、Fa3 可用下面的公式计算:

$$Fa0 = F \times La0 / L$$

$$Fa1 = F \times La1 / L$$

$$Fa2 = F \times La2/L$$

$$Fa3 = F \times La3/L$$

$$L = \sqrt{La0^2 + La1^2 + La2^2 + La3^2}$$

1) Velocity (目标速度)

如果 Velocity 设置为 0，一个插补速度规范错误会发生，轴组中所有的轴将停止。

如果根据 Velocity 计算出的目标速度 $Fa0 \sim Fa3$ 超出了最大速度，Velocity 会自动调整，使其中一个轴工作在最大速度。

2) Jerk

当 Jerk 设置为 0 时，指定的插补速度 Vt 和指定的加速度 At 、指定的减速度 Dt 之间的关系如图 7-106 所示，加速和减速过程中，速度呈直线。图 7-106b 中，由于移动的距离短，插补速度达不到指定的 Vt 。

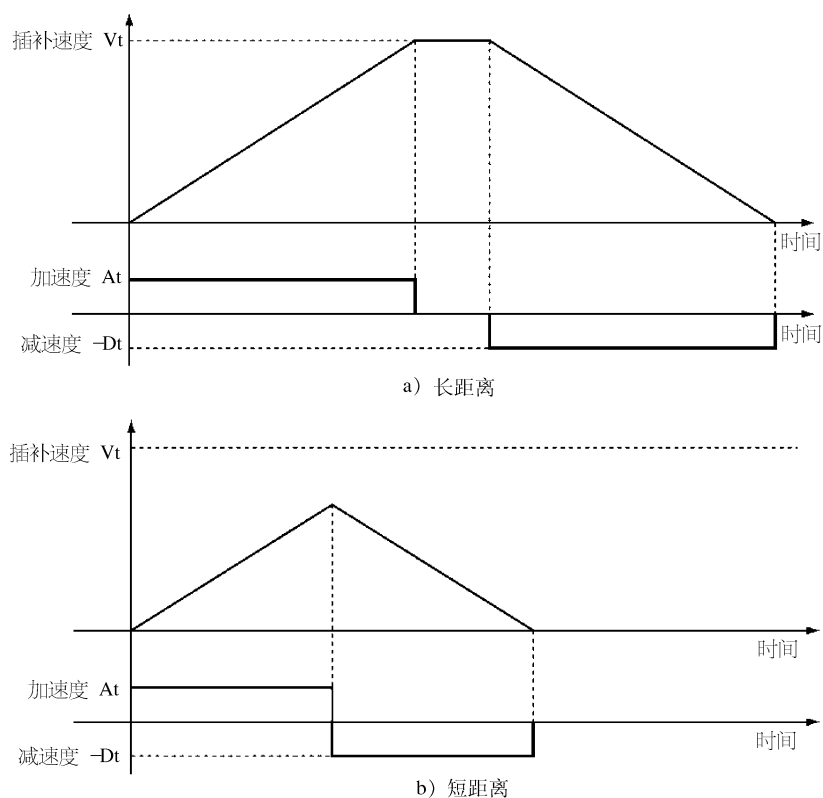


图 7-106 Vt 和 At 、 Dt 之间的关系 ($Jerk = 0$)

当 Jerk 设置为非 0 值时，指定的插补速度 Vt 和指定的加速度 At 、指定的减速度 Dt 之间的关系如图 7-107 所示。加速度和减速度呈梯形，加速度或减速度达到其限位值。加速和减速过程中，速度呈 S 形曲线。图 7-107b 中，由于移动的距离短，插补速度达不到指定的 Vt 。

(4) CoordSystem (坐标系统)

CoordSystem 指定用于直线插补的坐标系统，只支持两轴或多轴组成的轴坐标系统 (ACS)。

(5) BufferMode (缓冲模式选择)

BufferMode 指定插补指令、先前的插补指令如何加入到轴运行中，与单轴一样，也有 6 种模式。

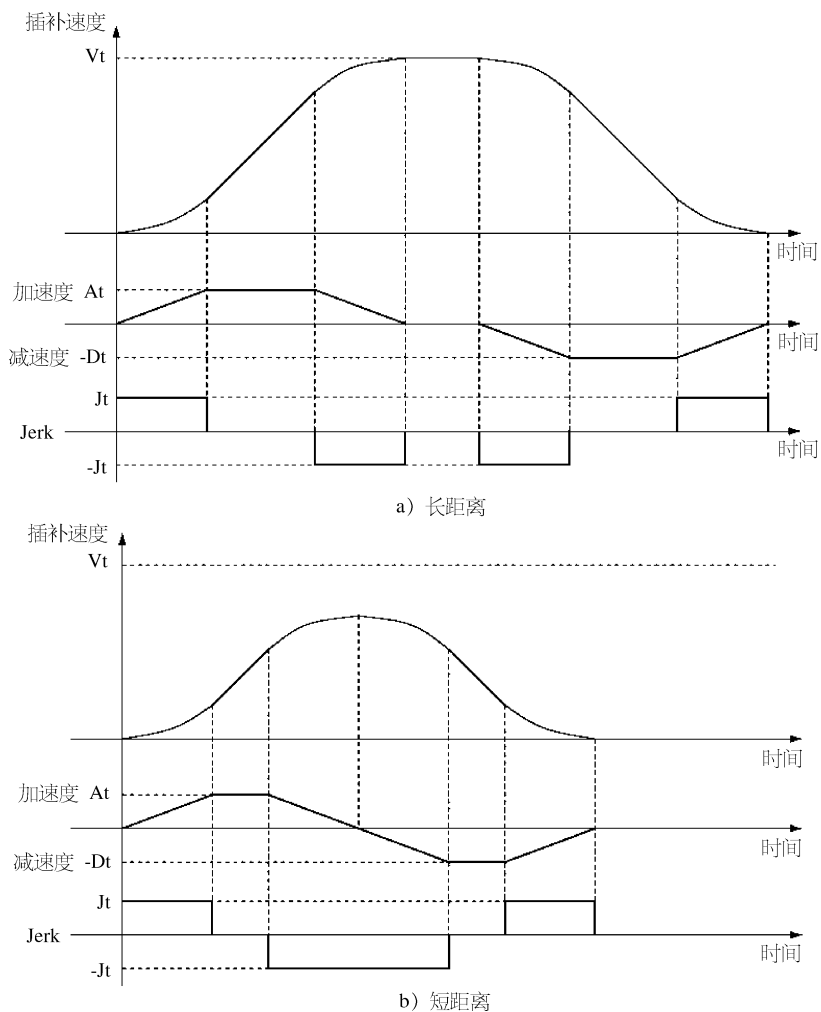


图 7-107 V_t 和 A_t 、 D_t 之间的关系 ($Jerk \neq 0$)

(6) TransitionMode (过渡模式)

TransitionMode 指定如何将先前插补操作和后面插补操作建立的路径结合起来。

TransitionMode 设置为 `_mcTMNone` (过渡禁用) 或者 `_mcTMCornerSuperimposed` (重叠拐弯)。

当 BufferMode 指定为混合时，TransitionMode 被使能。

当 BufferMode 未指定为混合时，如果 TransitionMode 不设置为 `_mcTMNone` (过渡禁用)，则会发生错误。

1) Transition Disabled (过渡禁用)

当创建速度命令值时，由于路径优先给出，所以当从一个操作切换到下一个操作时，轴

的速度命令值可以迅速改变。

如图 7-108 所示, 从 P1 过渡到 P2、P2 过渡到 P3 时, Velocity、BufferMode 和 TransitionMode 如下所示。

①从 P1 到 P2 的运动: Velocity = F, BufferMode = Aborting, TransitionMode = _mcTMNone。

②从 P2 到 P3 的运动: Velocity = F, BufferMode = Blend with next, TransitionMode = _mcTMNone。

③从位置 P1 开始运动, 经过位置 P2, 直线插补执行到位置 P3。

④在经过位置 P2 时, 直线插补速度 F 是保持的。正因为如此, 图 7-108 中显示的在位置 P2 的速度是不连续的。

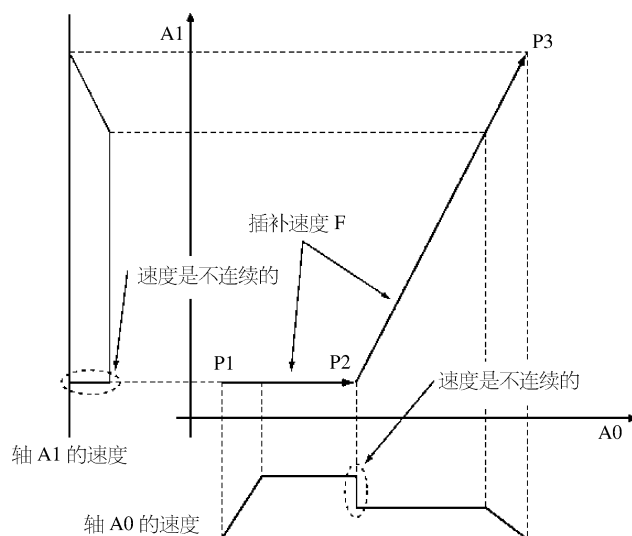


图 7-108 TransitionMode 为 Transition Disabled

2) Superimpose Corner (叠加拐角)

如果想使轴命令速度连续, 使用 Superimpose Corner 规范。

如图 7-109 所示, 从 P1 过渡到 P2、P2 过渡到 P3 时, Velocity、BufferMode 和 TransitionMode 如下所示。

①从 P1 到 P2 的运动: Velocity = F, BufferMode = Aborting, TransitionMode = _mcTMNone。

②从 P2 到 P3 的运动: Velocity = F, BufferMode = Blend with next, TransitionMode = _mcTMCornerSuperimposed。

③从位置 P1 开始运动, 从位置 P2 附近经过, 直线插补执行到位置 P3。

④为了使轴命令速度连续, 先前运动的减速范围和当前运动的加速范围相结合来产生命令速度。出于这个原因, 当前运动的加速时间和先前运动的减速时间是相同的。

组合路径从 P2 附近经过。

当插补速度较快或之前指令的减速度较小时, 从 P2 到路径的距离会比较长。当插补速度较慢或之前指令的减速度较大, 会比较短。

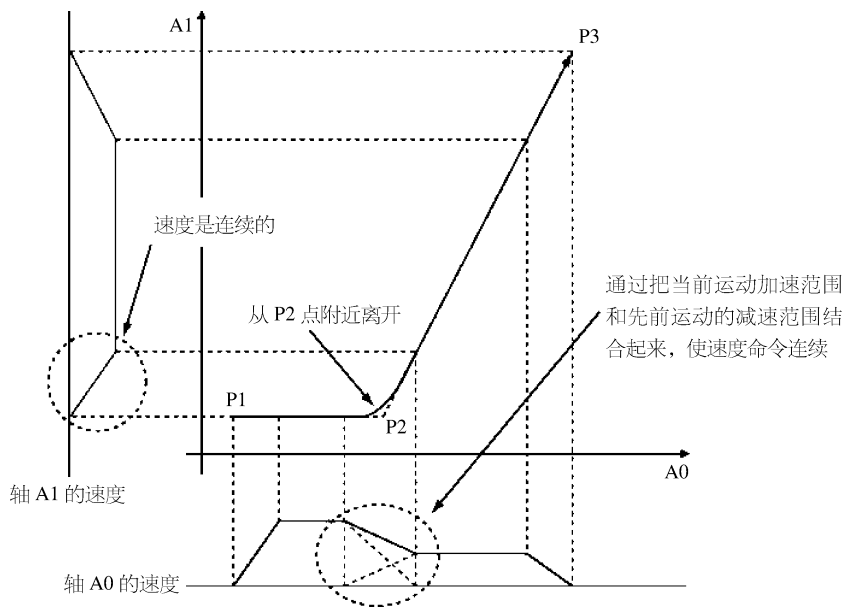


图 7-109 TransitionMode 为 Superimpose Corner

(7) 运动控制指令的重新执行

MC_MoveLinear 指令不能重新执行。

(8) 运动控制指令的多次执行

MC_MoveLinear 指令可以多次执行。

例 7.4 通过 MC_MoveLinear 指令的多次执行完成两轴直线插补过程, 如图 7-110 所示, 从原点出发经过直线①、②、③, 最后返回到原点并停止。(轴 1, 轴 2) 的过程顺序: (0.00mm, 0.00mm)→(50.00mm, 5.00mm)→(50.00mm, 0.00mm)→(0.00mm, 0.00mm)。

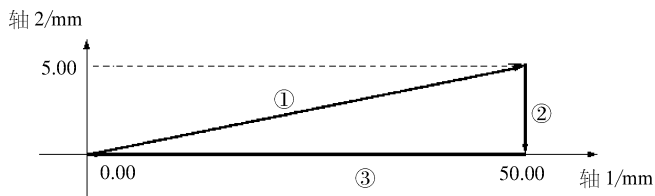


图 7-110 直线插补过程

1) 参数设置

设置轴参数

①轴类型:

轴 1: 伺服轴;

轴 2: 伺服轴。

②计数模式:

轴 1: 线性模式;

轴 2: 线性模式。

③显示单位:

轴 1: mm;

轴 2: mm。

轴组参数设置

①轴构成:

设置两个轴。

②轴选择:

设置轴 1 和轴 2。

2) 主要变量

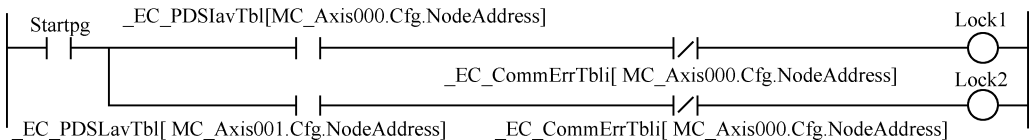
主要变量见表 7-44。

表 7-44 主要变量

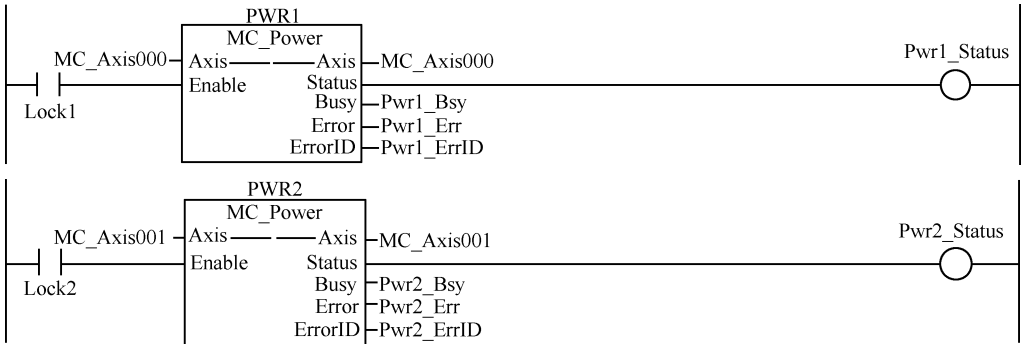
变 量	数 据 类 型	默认值	注 释
MC_Group000	_sGROUP_REF	—	这是轴组 0 的轴组变量
MC_Group000. MFaultLvl Active	BOOL	—	如果轴组 0 有一个轻微故障级别的错误,则为 TRUE
MC_Group000. Status. Disabled	BOOL	FALSE	轴组 0 禁用时值是 TRUE
MC_Axis000	_sAXIS_REF	—	这是轴 1 的轴变量
MC_Axis000. Details. Homed	BOOL	FALSE	当轴 1 原点定义时为 TRUE
MC_Axis000. MFaultLvl Active	BOOL	FALSE	当轴 1 出现轻微故障级错误时为 TRUE
MC_Axis001	_sAXIS_REF	FALSE	这是轴 2 的轴变量
MC_Axis001. Details. Homed	BOOL	FALSE	当轴 2 原点定义时为 TRUE
MC_Axis001. MFaultLvl Active	BOOL	FALSE	当轴 2 出现轻微故障级错误时为 TRUE
Pwr1_Status	BOOL	FALSE	该变量被分配给 MC_POWER 指令的 PWR1 例程的 Status 输出变量。伺服 ON 时,该变量变为 TRUE
Pwr2_Status	BOOL	FALSE	该变量被分配给 MC_POWER 指令的 PWR2 例程的 Status 输出变量。伺服 ON 时,该变量变为 TRUE
StartPg	BOOL	FALSE	如果该变量为 TRUE 并且建立了 EtherCAT 数据通信,伺服为 ON
InitFlag	BOOL	FALSE	该变量表示是否有必要设置输入参数。当这个变量为 FALSE 时设置输入参数,当输入参数设置完成后,该变量变为 TRUE

3) 梯形图

如果 StartPg 为 TRUE, 检查 EtherCAT 通信, 确认过程数据通信是否正常。



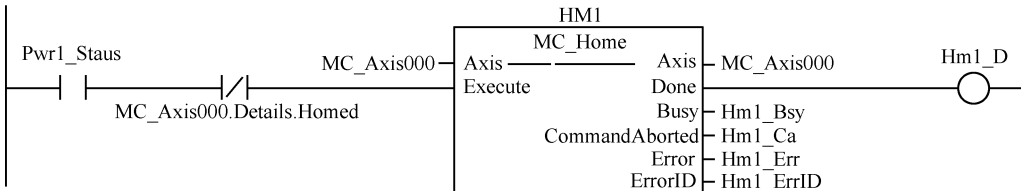
如果过程数据通信激活，Servo 变为 ON。



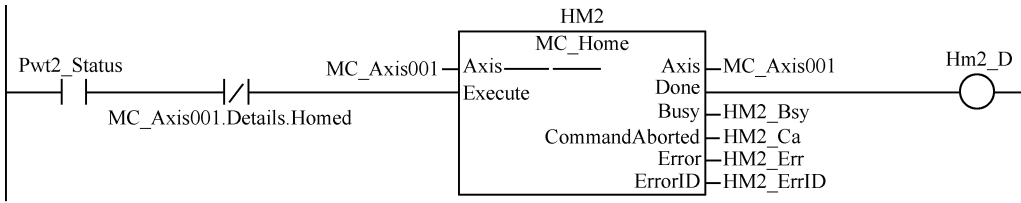
如果构成轴有一个轻微故障级错误，变量 MFault 为 ON。



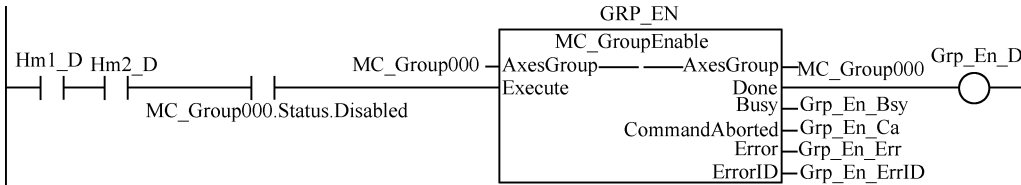
如果轴 1 的 Servo 为 ON 但原点没有定义，执行 MC_Home 指令以定义原点。



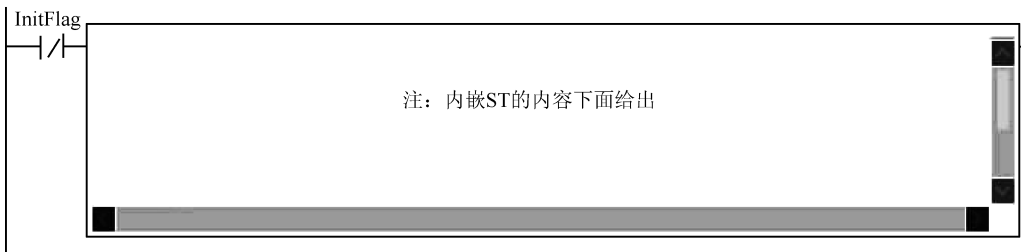
如果轴 2 的 Servo 为 ON 但原点没有定义，执行 MC_Home 指令以定义原点。



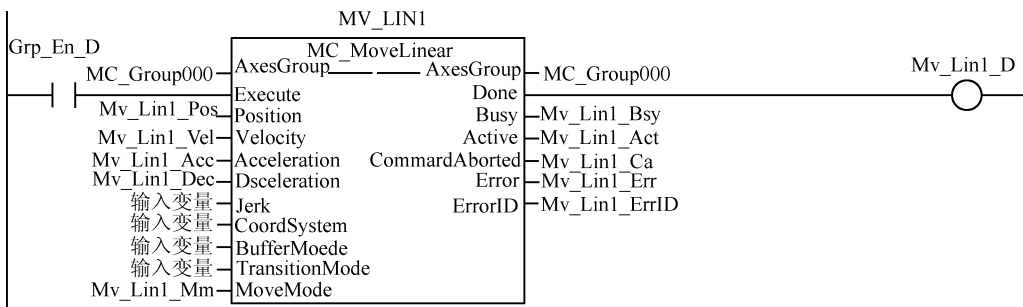
在轴 1 和轴 2 完成原点返回后，轴组使能。



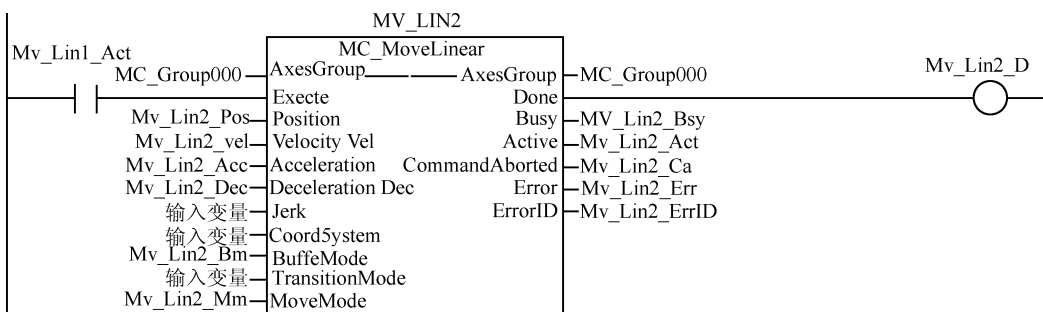
设置直线插补的参数。



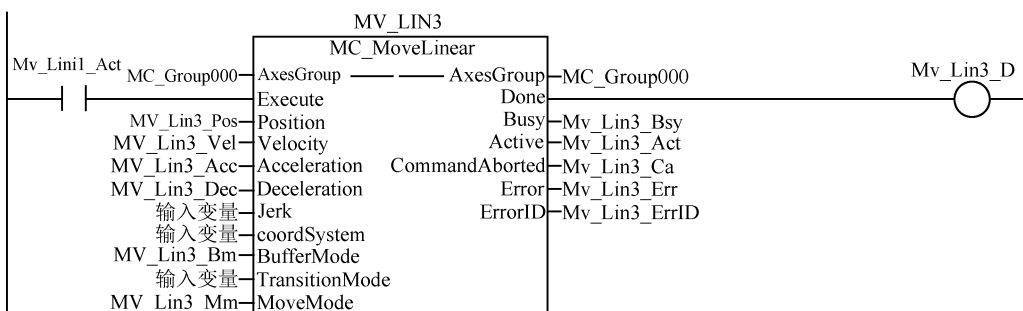
如果轴组使能, 执行直线插补①



直线插补①启动后, 用指令多次执行方式执行直线插补②。



直线插补①启动后, 用指令多次执行方式执行直线插补③



内嵌 ST 的内容:

```
//MV_LIN1 parameters
```

```
Mv_Lin1_Pos[0] := LREAL#10#50.0;
```

```
Mv_Lin1_Pos[1] := LREAL#10#5.0;
```

```
Mv_Lin1_Vel := LREAL#10#100.0;
```

```
Mv_Lin1_Acc := LREAL#10#100.0;
```

```
Mv_Lin1_Dec := LREAL#10#100.0;
```

```
Mv_Lin1_Mm := _eMC_MOVE_MODE#_mcAbsolute;
```

```
//MV_LIN2 parameters
```

```
Mv_Lin2_Pos[0] := LREAL#10#50.0;
```

```
Mv_Lin2_Pos[1] := LREAL#10#0.0;
```

```
Mv_Lin2_Vel := LREAL#10#100.0;
```

```
Mv_Lin2_Acc := LREAL#10#100.0;
```

```

Mv_Lin2_Dec: = LREAL#10#100. 0;
Mv_Lin2_Bm: = _eMC_BUFFER_MODE#_mcBuffered;
Mv_Lin2_Mm: = _eMC_MOVE_MODE#_mcAbsolute;

```

```
//MV_LIN3 parameters
```

```

Mv_Lin3_Pos[0]: = LREAL#10#0. 0;
Mv_Lin3_Pos[1]: = LREAL#10#0. 0;
Mv_Lin3_Vel: = LREAL#10#100. 0;
Mv_Lin3_Acc: = LREAL#10#100. 0;
Mv_Lin3_Dec: = LREAL#10#100. 0;
Mv_Lin3_Bm: = _eMC_BUFFER_MODE#_mcBuffered;
Mv_Lin3_Mm: = _eMC_MOVE_MODE#_mcAbsolute;

```

```
//输入参数设置后,InitFlag 变为 TRUE
```

```
InitFlag: = TRUE;
```

7.8.6 轴组圆弧插补指令 MC_MoveCircular2D

1. 指令的梯形图

MC_MoveCircular2D 指令的梯形图如图 7-111 所示。

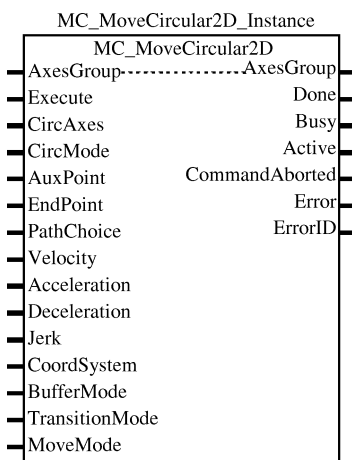


图 7-111 MC_MoveCircular2D 指令的梯形图

MC_MoveCircular2D 指令的主要输入变量见表 7-45。

表 7-45 MC_MoveCircular2D 指令的主要输入变量

名 称	含义	数 据 类 型	取 值 范 围	默认值	说 明
CircAxes	圆弧轴	ARRAY[0,1] OF UINT	0 ~ 3	0	指定圆弧插补的轴: 0:轴 A0 1:轴 A1 2:轴 A2 3:轴 A3

(续)

名 称	含义	数 据 类 型	取 值 范 围	默认值	说 明
CircMode*	圆弧插补模式	_eMC_CIRC_MODE	0: _mcBorder 1: _mcCenter 2: _mcRadius	0	指定圆弧插补的方法: 0:边界点 1:中心 2:半径
AuxPoint	辅助点	ARRAY[0,1] OF LREAL	负数,正数或 0	0	指定边界点,中心或半径。 单位是命令单位
EndPoint	终点	ARRAY[0,1] OF LREAL	负数,正数或 0	0	指定目标位置,单位是命令单位
PathChoice*	路径选择	_eMC_CIRC_PATHCHOICE	0: _mcCW 1: _mcCCW	0	指定路径方向: 0: CW 1: CCW
CoordSystem*	坐标系	_eMC_COORD_SYSTEM	0: _mcACS	0	指定坐标系: 0:轴坐标系(ACS)
BufferMode*	Buffer 模式选择	_eMC_BUFFER_MODE	0: _mcAborting 1: _mcBuffered 2: _mcBlendingLow 3: _mcBlendingPrevious 4: _mcBlendingNext 5: _mcBlendingHigh	0	当执行一条以上的运动指令时,指定操作: 0: Aborting 1: Buffered 2: Blend with low 3: Blend with previous 4: Blend with next 5: Blend with high
Transition-Mode*	过渡模式	_eMC_TRANSITION_MODE	0: _mcTMNone 10: _mcTMComerSuperimpose	0	指定运动路径: 0:过渡禁用 10:叠加拐弯
MoveMode*	移动模式	_eMC_MOVE_MODE	0: _mcAbsolute 1: _mcRelative	0	指定移动方法 0:绝对定位 1:相对定位

* 对枚举体变量,实际指定时用枚举体元素,不用它对应的数值。

2. 功能

MC_MoveCircular2D 指令完成两轴的 2D 圆弧插补。

3. 指令详情

(1) 圆弧插补的步骤

圆弧插补的步骤如下。

1) 登记进行插补的轴组

①选择进行插补的轴组,轴组用“_MC_GRP[***]”表示。

②指定轴组的构成,指定 2 个轴。

③为轴组指定执行插补的轴的组合。

④为轴分配逻辑轴 (轴 A0~A3), 不是轴号 (轴 0~63)。

⑤按逻辑轴顺序从最低号开始, 为逻辑轴 A0~A3 指定轴号 0~63。

⑥在 Sysmac Studio 的轴组基本设置, 为逻辑轴选择构成轴, 为逻辑轴分配轴号。

圆弧插补时只用两个轴, 所以可只用 A0、A1 为它们分配构成轴的轴号。

如果一个编码器轴或虚拟编码器轴包含在轴组中, 会发生轴类型错误并且操作停止。确保选择的只有伺服轴或虚拟伺服轴。

2) 使能轴组

①执行 MC_GroupEnable 指令来使能登记的轴组。

②使构成轴的伺服 ON, 将 MC_Stop 指令的 Execute 变为 FALSE。

这样, 使用的圆弧插补指令即被使能。

(2) CircAxes (圆弧轴)

圆弧插补使用 X 轴和 Y 轴, 如图 7-112 所示。

用 CircAxes 指定用作 X 轴和 Y 轴的轴, 轴使用逻辑轴 (轴 A0~A3), 而不是轴号 (轴 0~63)。

指定用作 X 轴和 Y 轴的轴的计数模式为线性模式。如果指定为环形模式, 在执行时计数模式会引起圆弧插补执行错误 (错误代码: 544AH)。

(3) CirMode (圆弧插补模式)

有三种圆弧插补方法: 边界点、中心和半径, 可以用 CirMode 指定这些方法中的一种。这些方法可以用于绝对定位和相对定位。用 MoveMode 指定绝对定位或相对定位。

如图 7-113 所示, 绝对定位时, 在轴坐标系统中, 指定符合边界点规范的边界点, 或符合中心规范的中心点和终点, 从原点进行绝对定位。

如图 7-114 所示, 相对定位时, 指定符合边界点规范的边界点, 或符合中心规范的中心点和终点, 从起始点进行相对定位。

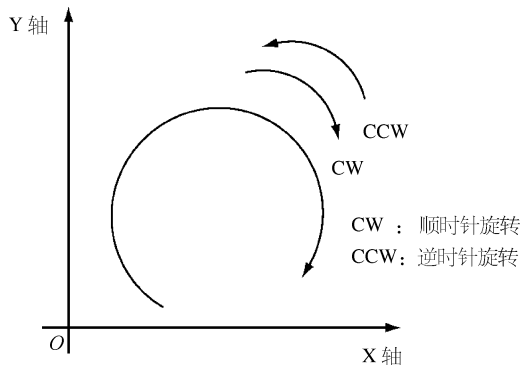


图 7-112 圆弧轴

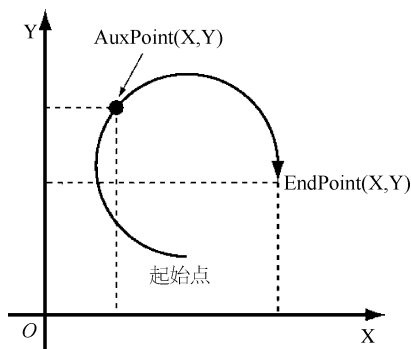


图 7-113 绝对定位

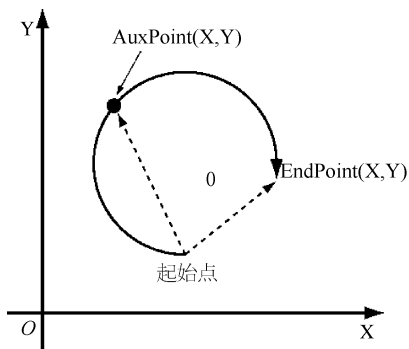


图 7-114 相对定位

假设 MoveMode 指定为绝对定位，以下为三种圆弧插补方法。

1) 边界点

如图 7-115 所示，当前位置是起始点，圆弧插补经过边界点 AuxPoint(X,Y)，到终点 EndPoint(X,Y) 结束。

如图 7-116 所示，如果起始点、边界点和终点是在同一条直线上，如果边界点和终点是同一点，或者，如果起始点和边界点是同一点，则从起始点到终点进行直线插补。

起始点和终点之间连一条直线，无论是在 X 轴还是 Y 轴方向上，如果某点到这条直线的距离都小于一个脉冲，则认为这个点与起始点、终点在一条直线上。

如果起始点、边界点和终点是相同点，会出现错误。

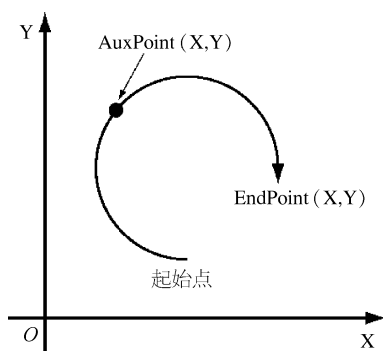


图 7-115 指定边界点的圆弧插补

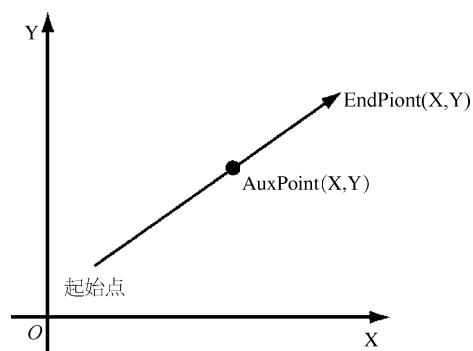


图 7-116 直线插补

如图 7-117 所示，如果起始点和终点是相同的点，用起始点和边界点作为直径绘制一个完整的圆。PathChoice 指定圆弧插补方向。

2) 中心

当前位置是起始点，对中心点 AuxPoint (X,Y) 和终点 EndPoint (X,Y) 指定的圆，执行圆弧插补，图 7-118 所示。PathChoice 指定圆弧插补方向。当起始点和终点都在同一点上时绘制出一个完整的圆。如果从中心到起始点的半径，与到终点的半径不同，两个半径的平均值用于执行圆弧插补。这种情况下，中心用和计算半径相同的方式计算出来。圆弧插补时使用计算出来的半径和中心。

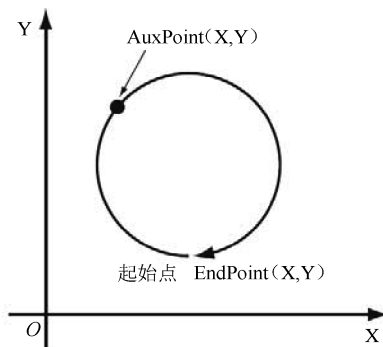


图 7-117 起始点和终点重叠

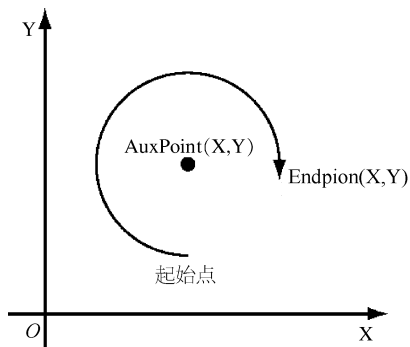


图 7-118 指定中心的圆弧插补

3) 半径

当前位置是起始点, 对半径 AuxPoint (X,Y) 和终点 EndPoint (X,Y) 指定的圆, 执行圆弧插补, 如图 7-119 所示。半径被指定为 AuxPoint (X,Y) 的第一个元素, 第二个元素不使用。例如, 半径 100 设置 AuxPoint (X,Y) 为 AuxPoint (100,0)。如果半径的符号是负的, 则绘制一个长的圆弧。如果半径的符号是正的, 则绘制一个短的圆弧。PathChoice 指定圆弧插补方向。

如果起点和终点是一样的, 一个圆弧插补起始点和终点相同的错误会发生, 并且该组的所有轴的操作会停止。

如果指定的半径小于起始点和终点之间的距离的一半, 画不出圆来, 会发生错误。

(4) Velocity、Acceleration、Deceleration 和 Jerk

设置 Velocity、Acceleration、Deceleration 和 Jerk, 以指定圆弧插补的速度、加速度、减速度和加加速度。如果设置圆弧插补的速度为 0, 一个速度规范错误会发生并且该组的所有轴的操作会停止。如果指定的插补速度超过了轴的最大速度, 以下操作会被执行。

1) 如果只有一个轴超过最大速度: 该轴按此最大速度运行并且圆弧插补速度会进行相应调整。

2) 如果所有轴超过最大速度: 插补速度自动调整以适应所有最大速度中速度较小的一个轴。

(5) CoordSystem (坐标系)

CoordSystem 指定用于圆弧插补的坐标系统。只支持两轴或多轴组成的轴坐标系统 ACS。

(6) BufferMode (缓冲模式选择)

BufferMode 指定插补指令、先前的插补指令如何加入到轴运行中, 与单轴一样, 也有 6 种模式。

(7) TransitionMode (过渡模式)

TransitionMode 指定插补指令、先前的插补指令如何加入到运行中。

如果 BufferMode 设置为 Blending, TransitionMode 则为使能。

不使用 Blending 时, 如果不将 TransitionMode 设置为 mcTMNone (过渡禁用), 会出现错误。

(8) 运动控制指令的重新执行

MC_MoveCircular2D 指令不能重执行。

(9) 运动控制指令的多次执行

MC_MoveCircular2D 指令可以多次执行。

例 7.5 图 7-120 为两轴协调控制, 其中的轴通过直线插补或圆弧插补, 从原点开始最终运动到目标位置 (20.00mm,20.00mm)。

1) 参数设置

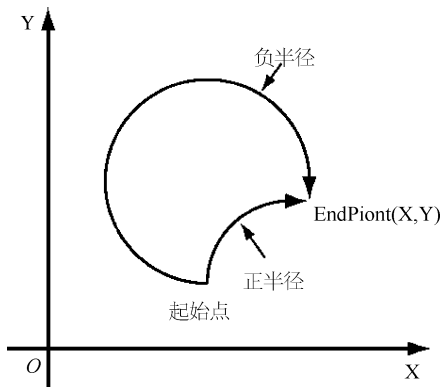


图 7-119 指定半径的圆弧插补

轴参数

①轴类型:

轴 1: 伺服轴;

轴 2: 伺服轴。

②计数模式:

轴 1: 线性模式;

轴 2: 线性模式。

③显示单位:

轴 1: mm;

轴 2: mm。

轴组参数

①轴构成:

设置两个轴。

②轴选择:

设置轴 1 和轴 2。

2) 操作过程

①执行: 在将操作启动开关置 ON, 通过直线插补, 轴移动到点 (10.00mm, 5.00mm)。

②连续运行: 通过圆弧插补, 轴继续移动到点 (15.00mm, 10.00mm); 通过直线插补, 到点 (15.00mm, 15.00mm); 通过圆弧插补到点 (20.00mm, 20.00mm)。这里的速度为 10.00mm/s。

3) 主要变量

主要变量见表 7-46。

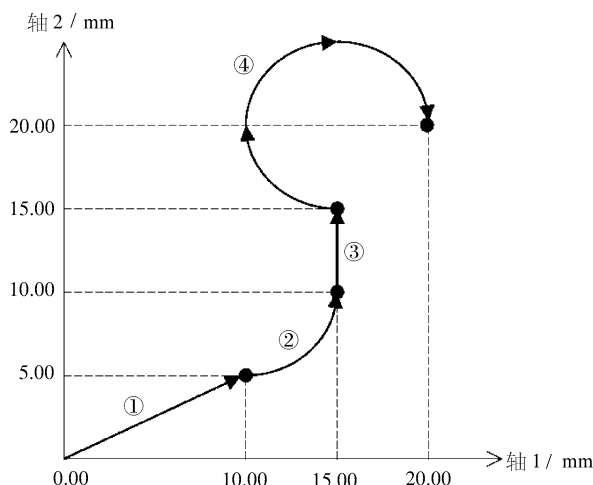


图 7-20 为两轴协调控制

表 7-46 主要变量

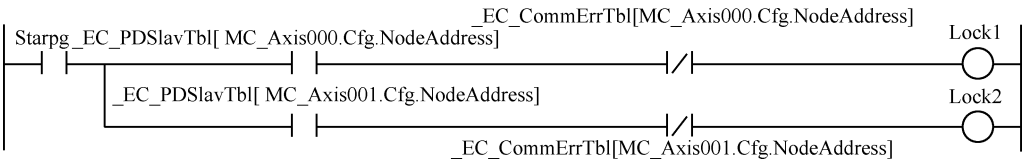
变 量	数 据 类 型	默认值	注 释
MC_Group000	_sGROUP _REF	—	这是轴组 0 的轴组变量
MC_Group000. MFaultLvl. Active	BOOL	FALSE	如果轴组 0 有一个轻微故障级的错误则为 TRUE
MC_Group000. Status. Disabled	BOOL	FALSE	轴组 0 禁用时值为 TRUE
MC_Axis000	_sAXIS_REF	—	这是轴 1 的轴变量
MC_Axis000. Details. Homed	BOOL	FALSE	当轴 1 的原点定义时为 TRUE
MC_Axis000. MFaultLvl. Active	BOOL	FALSE	如果轴 1 有一个轻微故障级的错误则为 TRUE
MC_Axis001	_sAXIS_REF	FALSE	这是轴 2 的轴变量
MC_Axis001. Details. Homed	BOOL	FALSE	当轴 2 的原点定义时为 TRUE
MC_Axis001. MFaultLvl. Active	BOOL	FALSE	如果轴 2 有一个轻微故障级的错误则为 TRUE
Pwr1_Status	BOOL	FALSE	该变量被分配给 MC POWER 指令的 PWR1 例程的 Status 输出变量。伺服 ON 时, 该变量变为 TRUE

(续)

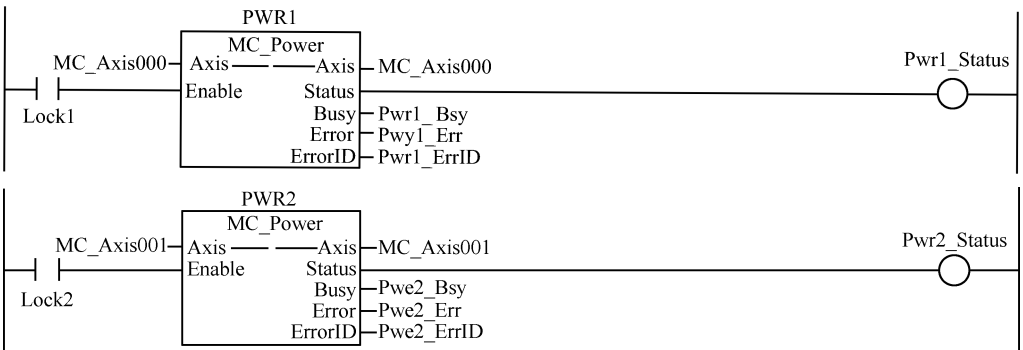
变 量	数 据 类 型	默认值	注 释
Pwr2_Status	BOOL	FALSE	该变量被分配给 MC_POWER 指令的 PWR2 例程的 Status 输出变量。伺服 ON 时,该变量变为 TRUE
StartPg	BOOL	FALSE	如果该变量为 TRUE 并且建立了 EtherCAT 数据通信, 伺服为 ON
InitFlag	BOOL	FALSE	此变量指示是否有必要设置输入参数。该变量为 FALSE 时设置输入参数,当输入参数设置完成后,该变量变为 TRUE

4) 梯形图

如果 StartPg 为 TRUE, 检查 EtherCAT 通信, 确认过程数据通信是否正常。



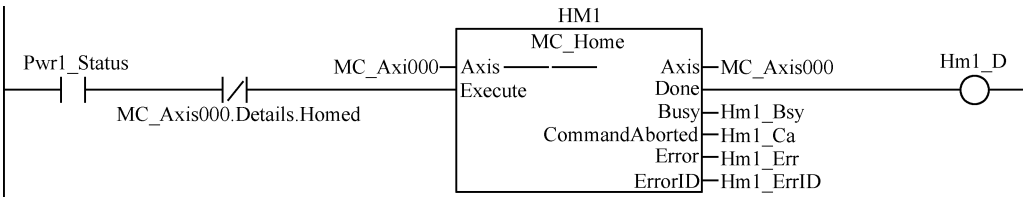
如果过程数据通信激活, Servo 变为 ON。



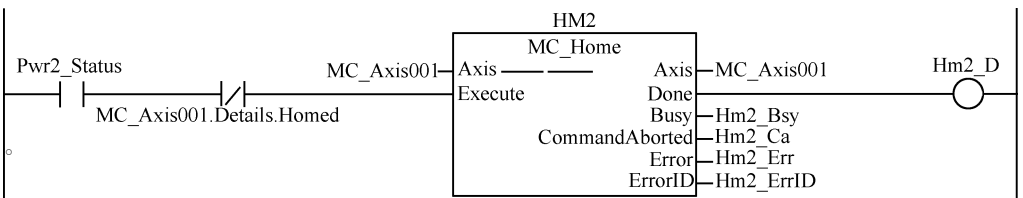
如果构成轴有一个轻微故障级错误, 变量 MFault 为 ON。



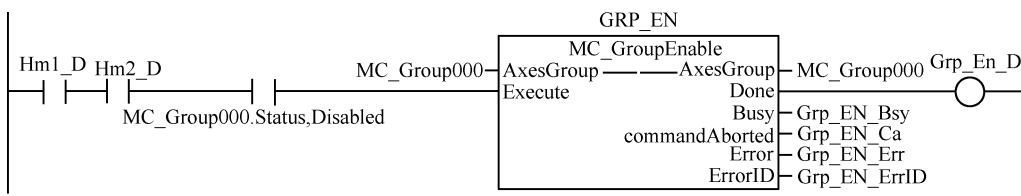
如果轴 1 的 Servo 为 ON 但原点没有定义, 执行 MC_Home 指令以定义原点。



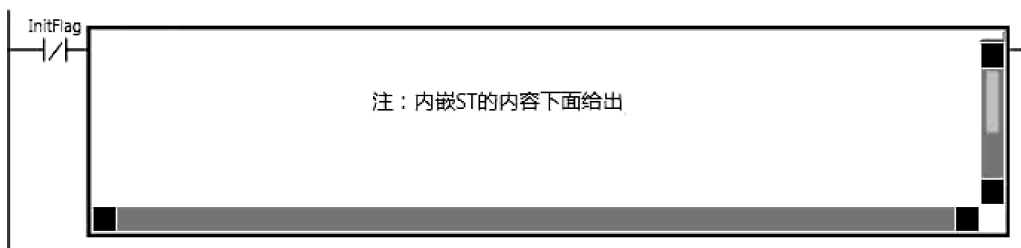
如果轴 2 的 Servo 为 ON 但原点没有定义, 执行 MC_Home 指令以定义原点。



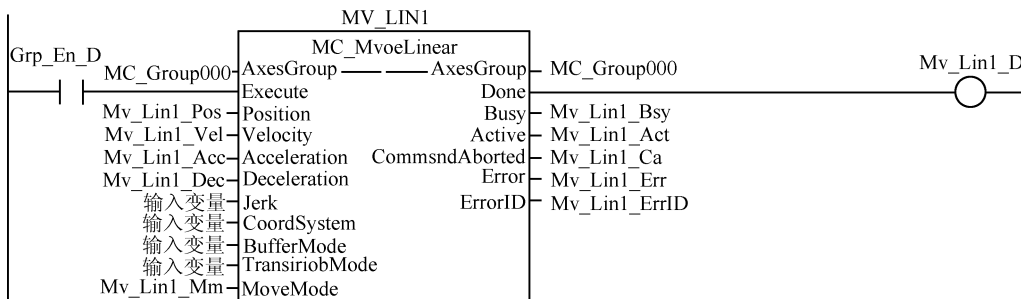
在轴 1 和轴 2 完成原点返回后，轴组使能



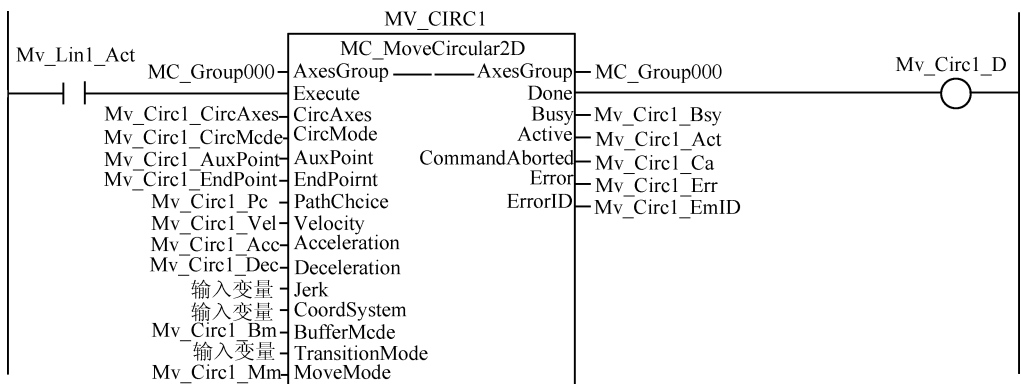
设置直线插补和圆弧插补的参数。



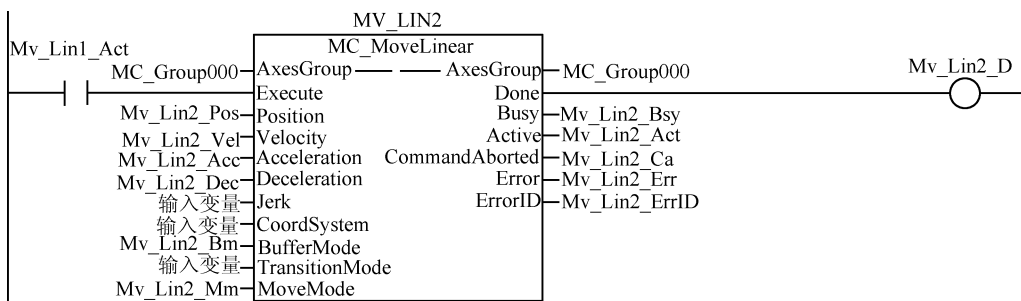
如果轴组使能，执行直线插补①。



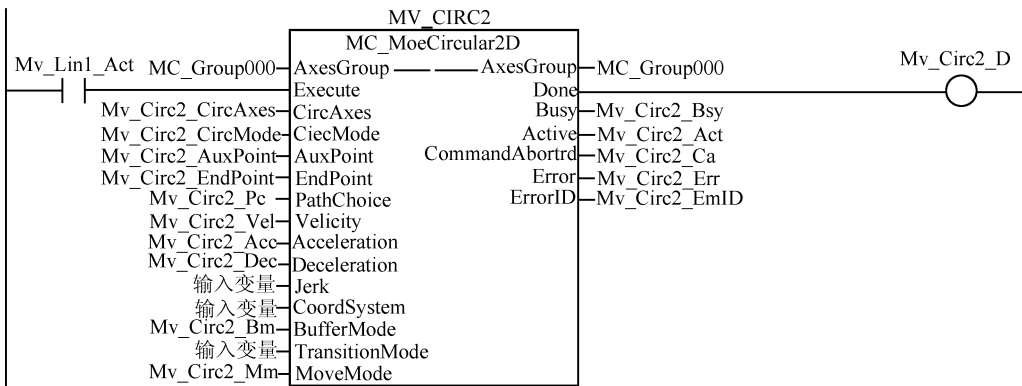
直线插补①启动后，用指令多次执行方式执行圆弧插补②。



直线插补①启动后，用指令多次执行方式执行直线插补③。



直线插补①启动后，用指令多次执行方式执行圆弧插补④。



内嵌 ST 的内容

//输入参数没有输入时的处理

IF InitFlag = FALSE THEN

 //MV_CIRC1 参数

```

Mv_Circ1_CircAxes[0] := UINT#10#0;
Mv_Circ1_CircAxes[1] := UINT#10#1;
Mv_Circ1_CircMode := _eMC_CIRC_MODE#_mcRadius;
Mv_Circ1_AuxPoint[0] := LREAL#5.0;
Mv_Circ1_AuxPoint[1] := LREAL#0.0;
Mv_Circ1_EndPoint[0] := LREAL#15.0;
Mv_Circ1_EndPoint[1] := LREAL#10.0;
Mv_Circ1_Pc := _eMC_CIRC_PATHCHOICE#_mcCCW;
Mv_Circ1_Vel := LREAL#100.0;
Mv_Circ1_Acc := LREAL#20.0;
Mv_Circ1_Dec := LREAL#20.0;
Mv_Circ1_Bm := _eMC_BUFFER_MODE#_mcBuffered;
Mv_Circ1_Mm := _eMC_MOVE_MODE#_mcAbsolute;

```

 //MV_CIRC2 参数

```

Mv_Circ2_CircAxes[0] := UINT#10#0;
Mv_Circ2_CircAxes[1] := UINT#10#1;
Mv_Circ2_CircMode := _eMC_CIRC_MODE#_mcCenter;
Mv_Circ2_AuxPoint[0] := LREAL#15.0;
Mv_Circ2_AuxPoint[1] := LREAL#20.0;
Mv_Circ2_EndPoint[0] := LREAL#20.0;
Mv_Circ2_EndPoint[1] := LREAL#20.0;
Mv_Circ2_Pc := _eMC_CIRC_PATHCHOICE#_mcCW;
Mv_Circ2_Vel := LREAL#100.0;
Mv_Circ2_Acc := LREAL#20.0;

```

```

Mv_Circ2_Dec:=LREAL#20.0;
Mv_Circ2_Bm:=_eMC_BUFFER_MODE#_mcBuffered;
Mv_Circ2_Mm:=_eMC_MOVE_MODE#_mcAbsolute;

//MV_LIN1 参数
Mv_Lin1_Pos[0]:=LREAL#10#10.0;
Mv_Lin1_Pos[1]:=LREAL#10#5.0;
Mv_Lin1_Vel:=LREAL#10#100.0;
Mv_Lin1_Acc:=LREAL#10#20.0;
Mv_Lin1_Dec:=LREAL#10#20.0;
Mv_Lin1_Mm:=_eMC_MOVE_MODE#_mcAbsolute;

//MV_LIN2 参数
Mv_Lin2_Pos[0]:=LREAL#10#15.0;
Mv_Lin2_Pos[1]:=LREAL#10#15.0;
Mv_Lin2_Vel:=LREAL#10#100.0;
Mv_Lin2_Acc:=LREAL#10#20.0;
Mv_Lin2_Dec:=LREAL#10#20.0;
Mv_Lin2_Bm:=_eMC_BUFFER_MODE#_mcBuffered;
Mv_Lin2_Mm:=_eMC_MOVE_MODE#_mcAbsolute;

//输入参数设置完毕,将 InitFlag 变为 TRUE
InitFlag:=TRUE;
END_IF;

```

7.9 运动控制功能

本章的第7、8两节分别介绍了NJ的单轴运动控制指令和轴组运动控制指令，在此基础上，本节进一步介绍应用NJ运动控制指令所能实现的各种功能。

7.9.1 单轴定位控制

1. 绝对定位

绝对定位时，指定目标位置相对原点的绝对坐标。例如，通过设置计数模式为环形模式和指定操作方向，在一个转盘上，完成最短路径定位。

图7-121为绝对定位。

绝对定位时，使用指令 MC_MoveAbsolute、MC_Move指令。

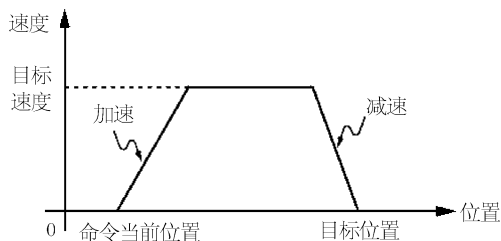


图 7-121 绝对定位

2. 相对定位

相对定位时,指定离开实际位置的距离。计数模式设置为环形模式时,可以指定一个超过了环形计数器范围的移动距离。

图 7-122 为相对定位。

相对定位时,使用指令 MC_MoveRelative、MC_Move 指令。

3. 中断进给

中断进给是当触发信号产生时,轴从实际位置以指定的速度进给指定的距离。当指定绝对或相对进给时,在指定的进给距离内,如果触发信号不出现,可以选择输出错误。进给不受后面出现的错误影响。当触发信号出现时,通过使用伺服驱动的闩锁功能,确定实际位置。可以使用视窗功能禁止在指定的位置范围之外出现触发信号。对于包装机械等应用,当在包装膜上打印商标的触发信号出现时,才使能进给,这样可消除其他影响。

如图 7-123 所示,在运动方向上进给指定的距离。

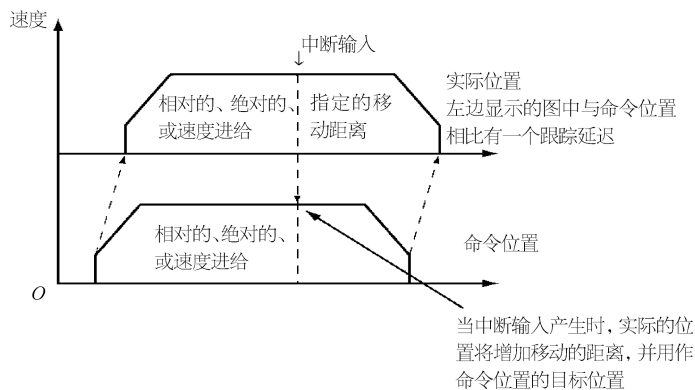


图 7-123 在运动方向上进给指定的距离

如图 7-124 所示,在运动反方向上进给指定的距离。

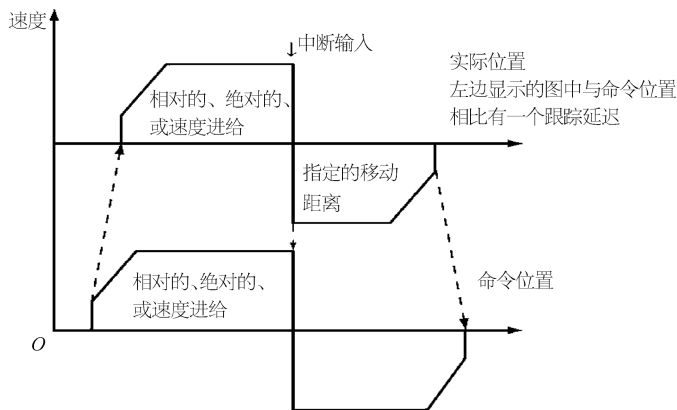


图 7-124 在运动反方向上进给指定的距离

如果操作选择指定反向后减速停止，反向的轴参数中使用加速/减速曲线。

中断进给时，使用指令 MC_MoveFeed 指令。

4. 停止

伺服操作停止功能包括：连接到伺服驱动的立即停止输入信号和限位信号引起的停止、用户程序中运动控制指令的停止功能、由错误引起的停止。

(1) 伺服输入信号引起的停止

连接到伺服驱动的立即停止输入信号或限位输入信号，停止轴运行。可以在 Sysmac Studio 选择停止方法。

1) 立即停止输入。

2) 限位输入 (正限位输入或负限位输入)。

(2) 用运动控制指令停止

用 MC_Stop 或 MC_ImmediateStop 指令停止单轴操作。

1) MC_Stop 指令

单轴控制和同步控制时，减速停止时可以指定减速度和加加速度。指定减速度为 0 时，发出命令立即停止伺服驱动。当这条指令减速停止同时输入变量 Execute 为 TRUE 时，其他操作命令不会得到应答。

2) MC_ImmediateStop 指令

单轴控制和同步控制时，使用这条指令可以立即停止。也可以用于在轴组中已被使能的轴。

(3) 由于出错或其他问题停止

1) 单轴操作时因出错停止；

2) 软件限位停止；

3) 由于运行控制周期过长停止；

4) 引起伺服变 OFF 的错误；

5) 由于 MC 试运行启动引起的停止；

6) 由于 MC 试运行结束引起的停止；

7) 由于 CPU 单元运行模式改变引起的停止。

(4) 停止方法

1) 减速停止

轴以指令指定的减速度，或以最大减速度停止。

2) 立即停止

命令不再更新。轴只对伺服器驱动剩余脉冲保持运动，然后停止。当立即停止的原因出现时，停止的位置就是命令的位置。

3) 立即停止且错误复位

立即停止原因发生时，实际位置用作命令位置。惯性会使轴转过这个位置，但当立即停止原因发生时它会返回实际位置并停在那里。

4) 立即停止且伺服 OFF

命令不再更新。当伺服变为 OFF 时，按照设置在伺服驱动中禁止操作的选项代码 (对象 605CH) 指定的方法，轴停止。

(5) 停止优先级

如表 7-47 所示, 列出每一种停止的优先级。停止时如果高优先级的停止方法出现, 停止方法转到高优先级方法上。

表 7-47 停止方法的优先级

停 止 方 法	优先级(数值越大优先级越高)	停 止 方 法	优先级(数值越大优先级越高)
立即停止且伺服 OFF	4	立即停止	2
立即停止且错误复位	3	减速停止	1

如图 7-125 所示, 当限位输入出现, 进入减速停止过程中, 当立即停止输入变为 ON 时, 则转入立即停止。

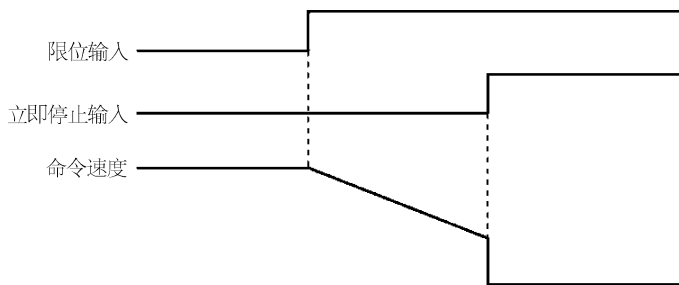


图 7-125 停止方法转换

5. 倍率因子

倍率因子设置为目标速度的百分数, 设置范围为 0% ~ 500%。如果目标速度的倍率因子为 0%, 运行状态继续, 轴停止在 0 速度上。只要倍率因子使能, 倍率因子则被读出。如果倍率因子禁用, 倍率因子返回 100%。当倍率因子改变时, 如果超出最大速度, 使用最大速度。

使用 MC_SetOverride 指令为当前正在运行的轴设置倍率因子。

7.9.2 单轴同步控制

所谓单轴同步控制是将从轴的位置与主轴的位置相同步。任何从轴的命令位置 and 实际位置都可以根据主轴指定。如果从轴的命令速度超过其轴参数设置的最大速度, 则命令以轴的最大速度执行。

1. 齿轮操作

这个功能指定主轴和从轴之间的齿轮比, 并启动运行。用 MC_GearIn 指令启动齿轮操作, 用 MC_GearOut 指令或 MC_Stop 指令结束同步。

2. 定位齿轮操作

这个功能指定主轴和从轴之间的齿轮比, 并启动运行。定位齿轮操作这个功能允许设置主轴和从轴的位置, 在这个位置上启动同步。用 MC_GearInPos 指令启动定位齿轮操作, 用 MC_GearOut 指令或 MC_Stop 指令结束同步。

操作启动后, 从轴使用主轴速度乘以齿轮比作为它的目标速度, 相应地进行加速/减速。从轴处于追赶阶段, 直到到达从轴同步启动位置。当到达从轴同步启动位置时, 从轴进入

InGear 阶段。此后，从轴位置与主轴同步。

3. 凸轮操作

凸轮操作使从轴位置按照凸轮表与主轴位置相同步。用 MC_CamIn 指令启动凸轮操作，用 MC_CamOut 指令或 MC_Stop 指令结束凸轮操作。在 Sysmac Studio 中使用凸轮编辑器生成凸轮表并下载到 CPU 单元中。

4. 单轴同步定位

当指定的从轴与指定的主轴进行同步时，该功能使用梯形曲线完成定位。这是电子凸轮的一种类型，但它不使用凸轮编辑器里创建的凸轮表。当执行 MC_MoveLink (同步定位) 指令时启动操作，使用 MC_STOP 指令停止轴的运行。从站 (从轴) 操作执行时进行如下设置：主站 (主动轴)、MasterDistance (主轴距离)、MasterDistanceInACC (主轴加速距离)、MasterDistanceInDEC (主轴减速距离)、SlaveDistance (从轴移动距离) 和 MasterStartDistance (主轴跟随距离)。主轴的命令位置或实际位置可以指定。可以指定下列条件之一作为同步操作的启动条件：指令启动、检测到触发信号或者主轴到达主轴的跟随距离。

5. 组合轴

两个命令位置的和或差作为从轴的命令位置。用 MC_CombineAxes 指令启动操作，用 MC_Stop 指令停止轴的运行。

6. 主轴相移

对当前的指令，主轴的相位可以参照从轴的相位移动。参照从轴移动的数量是一个相对值。在同步过程中，从轴与主轴相对距离相同步。对于同步控制指令，可以执行 MC_Phasing (主轴相位移动) 指令移动相位。

MC_Phasing 指令可以指定相移量、目标速度、加速度、减速度和加加速度。

7.9.3 单轴速度控制

1. 速度控制

速度控制用来使轴以指定的速度恒定运行。可以指定加速度、减速度和加加速度。使用 MC_Stop 指令或执行另一条运动指令，停止轴。如果指定目标速度为 0，轴不会运行但轴状态会显示正在运行。

MC 功能模块使用伺服驱动的位置控制模式，发送目标位置指令，以达到指定的目标速度。

速度控制时，使用 MC_MoveVelocity 指令。

2. 周期同步速度控制

伺服驱动的控制模式设置为速度控制模式，每一个控制周期输出一个命令速度。要停止一个轴，使用 MC_Stop 指令或执行另一条运动控制指令。如果指定目标速度为 0，轴不会运行但轴状态会显示轴在运行。

周期同步速度控制时，使用 MC_SyncMoveVelocity 指令。

7.9.4 单轴扭矩控制

如图 7-126 所示，扭矩控制是指连续输出指定数量的扭矩。可以使用 TorqueRamp 指定扭矩的变化速度，直到 Torque (目标扭矩) 达到。为停止轴，使用 MC_Stop 指令或执行另一

条运动指令。如果指定一个扭矩 0 (目标扭矩), 轴不会运行但轴状态会显示轴在运行。

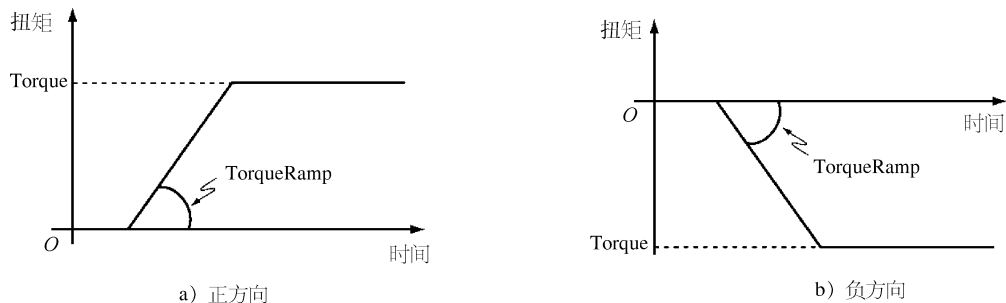


图 7-126 扭矩控制

MC 功能模块使用伺服驱动的扭矩控制模式。伺服驱动在扭矩控制环接收从 MC 功能模块来的扭矩命令值, 控制扭矩。可以在运动控制指令的 Velocity 输入变量, 为伺服驱动指定速度限值。在扭矩控制模式, 当电动机上的负载降低时, 使用它限制电动机的高速度。

为安全起见, 一定为扭矩控制设置速度限值。

7.9.5 单轴控制的通用功能

1. 位置

(1) 位置类型

MC 功能模块使用下列两种位置类型, 见表 7-48。

表 7-48 位置类型

位置类型	定义
命令位置	这是 MC 功能模块的输出, 控制一个轴的位置
实际位置	实际位置是伺服驱动输入或编码器的输入

图 7-127 所示为命令位置 and 实际位置的关系。

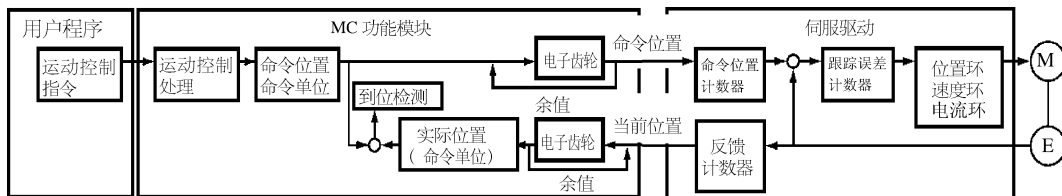


图 7-127 命令位置 and 实际位置的关系

命令位置 and 实际位置都有表 7-49 所列的项目。

表 7-49 命令位置 and 实际位置的项目

项 目	命 令 位 置	实 际 位 置
计数模式	可以设置线性模式或环形模式	和命令位置相同的计数模式
位置增量	可以设置其中一个: mm、μm、nm、inch、degree 或 pulse	和命令位置的单位相同
软件限位	可以设定软件的操作范围	和命令位置的范围相同

(续)

项 目	命 令 位 置	实 际 位 置
改变当前位置	可以改变当前位置到任何希望的位置	可设定为和命令位置相同的位置*
定义原点	原点可以是定义的或未定义的	原点的状态和命令位置相同

* 如果改变前有跟踪误差，那么跟踪误差的值在实际位置保持。

(2) 与位置相关的轴参数

与位置相关的轴参数见表 7-50。

表 7-50 与位置相关的轴参数

参 数 名 称	功 能	设 定 范 围	默 认 值
定位范围	设置定位的范围(单位:命令单位)	非负长实数	10
定位检查时间	以毫秒为单位设置定位检查时间。只有当原点返回时定义了原点位置,才设为 0 以检查定位结束,其他时间不检查定位(单位:ms)	0 ~ 10000	0
软件限位	选择软件限位功能: 0:禁用 1:命令位置的减速停止 2:命令位置的立即停止 3:实际位置的减速停止 4:实际位置的立即停止	0 ~ 4	0
正软件限位	设置正方向上的软件限位(单位:命令单位)	长实数	2147483647
负软件限位	设置负方向上的软件限位(单位:命令单位)	长实数	-2147483647
跟踪误差超出值	设置跟踪误差超出检查值。设为 0 禁用跟踪误差超出检查(单位:命令单位)	非负长实数	0
跟踪误差警告值	设置跟踪误差警告检查值。设为 0 禁用跟踪误差警告检查(单位:命令单位)	小于或等于跟踪误差超出值的非负长实数	0

(3) 为轴操作指定目标位置

定位运动的实际位置或距离，用运动控制指令的 Position (目标位置) 和 Distance (移动距离) 输入变量指定。

(4) 监视位置

可在用户程序中读轴位置变量，监视位置，见表 7-51。

表 7-51 轴位置变量

变 量 名 称	数据类型	含 义	功 能
_MC_AX[0-63]. Cmd. Pos	LREAL	命令当前位置	这是命令位置的当前值。当伺服 OFF 并且不在位置控制模式时,实际当前位置是输出
_MC_AX[0-63]. Act. Pos	LREAL	实际当前位置	这是实际当前位置

2. 速度

(1) 速度类型

在 MC 功能模块中,使用两种类型的轴速度,见表 7-52。

表 7-52 速度类型

速度类型	定 义
命令速度	这是 MC 功能模块输出的速度,控制一个轴
实际速度	这是 MC 功能模块基于来自伺服驱动的输入或编码器输入的实际位置,计算出来的速度

(2) 速度单位

速度单位为命令单位/s。

(3) 与速度相关的轴参数

与速度相关的轴参数见表 7-53。

表 7-53 与速度相关的轴参数

参 数 名 称	功 能	设定值	默认值
最大速度	为轴指定最大速度。如果目标速度超过了轴运动指令指定的最大速度,轴将以最大速度运行	正长实数	40000000
最大点动速度	为轴指定最大点动速度。如果目标速度超过了一个轴的点动指令指定的最大点动速度,轴将以最大点动速度运行	正长实数	1000000
速度警告值	设定最大速度的百分比,达到这个速度时,轴输出一个速度警告。如果设为 0,则无警告输出(单位:%)	0~100	0
实际速度滤波器时间常数	以毫秒为单位设置时间段,计算实际运行速度的平均值。如果设为 0,则不计算平均速度(单位:ms) 当轴的速度缓慢时,以此来减少实际当前速度的变化	0~100	0

(4) 为轴操作指定目标速度

用于实际定位运动的速度,由运动控制指令的 Velocity 输入变量指定。

(5) 监视速度

在用户程序中读轴变量,监视速度,见表 7-54。

表 7-54 监视速度的轴变量

变 量 名 称	数据类型	含 义	功 能
_MC_AX[0-63]. Cmd. Vel	LREAL	命令当前速度	这是命令速度的当前值。运行在正方向时加一个加号,运行在负方向时加一个减号
_MC_AX[0-63]. Act. Vel	LREAL	实际当前速度	这是当前实际速度。运行在正方向时加一个加号,运行在负方向时加一个减号

3. 加速度和减速度

(1) 加速度和减速度的单位

加速度和减速度的单位为命令单位/s²。

(2) 与加速和减速相关的轴参数

与加速和减速相关的轴参数见表 7-55。

表 7-55 与加速和减速相关的轴参数

参数名称	功能	设定范围	默认值
最大加速度	为轴的操作指令设置最大加速度。设为 0 时,最大加速度没有限制(单位:命令单位/s ²)	非负长实数	0
最大减速度	为轴的操作指令设置最大减速度。设为 0 时,最大减速度没有限制(单位:命令单位/s ²)	非负长实数	0
加速度/减速度超出	轴加速/减速控制时,因为停在目标位置的优先级是特定的,导致过度加速/减速,会出现超出最大加速度/减速度,在上述情况下,设置操作: 0:使用快速加速/减速(混合变为缓冲) 1:使用快速加速/减速 2:轻微故障停止	0~2	0
加速度警告值	设置最大加速度的百分比,达到这个值时,轴输出加速度警告。设为 0 时,不输出加速度警告(单位:%)	0~100	0
减速度警告值	设置最大减速度的百分比,达到这个值时,轴输出减速度警告。设为 0 时,不输出减速度警告(单位:%)	0~100	0

(3) 为轴操作指定加速度和减速度

用于实际定位运动的加速度和减速度,由运动控制指令的 Acceleration 输入变量和 Deceleration 输入变量指定。

(4) 监视加速度和减速度

在用户程序中读轴变量,监视加速度和减速度。变量为_MC_AX[0-63].Cmd.AccDec, LREAL 型,这是命令的当前加速度/减速度。加速时加一个加号,减速时加一个减号。

4. Jerk (加加速度)

Jerk 指定加速度或减速度的变化率。如果 Jerk 已指定,加速时的波形为 S 形曲线,可以减少机器的振动。

(1) Jerk 单位

Jerk 单位为命令单位/s³。

(2) 为轴运动指定 Jerk

用于实际定位运动中的 Jerk,由运动控制指令的 Jerk 输入变量指定。它的值用于加速或减速,除同步控制操作外,可以为其他所有的操作设置。

使用下列公式计算 Jerk 的设定值:

$$\text{Jerk} = \text{加速度} \div (\text{加速时间} \times \text{加速时应用 Jerk 的时间比率}/2)$$

Jerk 在两个时间段使用:启动加速和停止加速。

在 Jerk 指定的区域内,加速以恒定的速率改变,命令速度会形成一条平滑的 S 形曲线。Jerk 为 0 的加速区域,加速度固定不变。命令速度形成一条直线。

如图 7-128 所示，加速度为 25000mm/s^2 ，加速时间为 0.1s ，Jerk 应用率为 50% 。
 $\text{Jerk} = 25000 / (0.1 \times 0.5/2) = 1000000(\text{mm/s}^3)$ 。

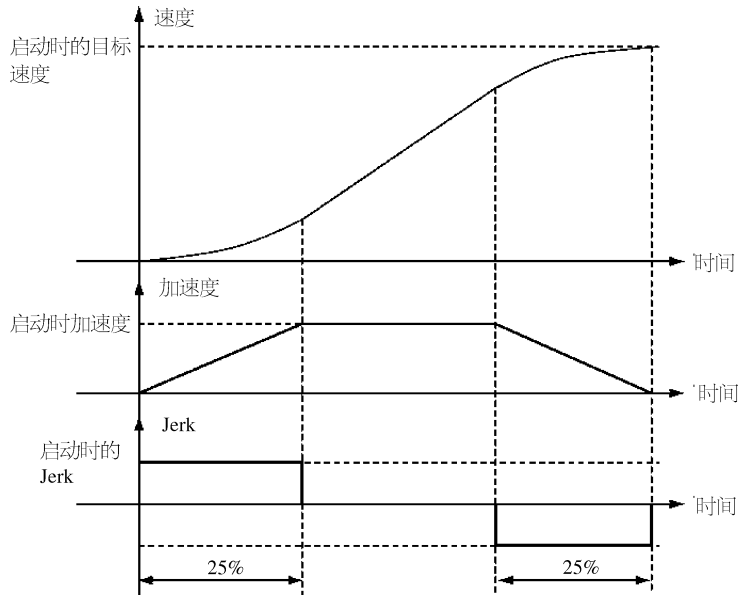


图 7-128 速度曲线

(3) 监视 Jerk

在用户程序中可以读轴变量，监视命令当前的 Jerk 值。变量为 `_MC_AX[0 ~ 63]`.
`Cmd. Jerk`，`LREAL` 型。

5. 指定操作方向

如果使用索引表指定旋转方向，例如最短路径，则要为旋转模式设置计数模式。接着，对绝对定位的运动控制指令，用 `Direction` 输入变量设置操作方向。在一个环计数周期或者一个环计数器的上、下限位内，设置任意目标位置。如果计数模式设定为线性模式，则忽略 `Direction` 输入变量，定位向着目标位置执行。

表 7-56 列出在 MC 功能模块中可以指定的不同方向。

表 7-56 MC 功能模块中的方向

方 向	操 作
最短路径	沿命令的当前位置和目标位置彼此最为接近的方向启动运行
正方向	沿正方向启动运行
负方向	沿负方向启动运行
当前方向	沿和先前的操作相同的方向启动运行
未指定方向	沿着不经过环形计数器的上限和下限的方向启动运行。有了这个方向指标,可以指定一个超过环形计数器上限或下限的目标位置。出现这种情况时,使用目标位置和命令当前位置之间的差别作为目标距离来进行相对定位。这样可以利用环形计数器进行多圈定位

图 7-129 为最短路径，命令当前位置为 50，定位的目标位置为 -20。

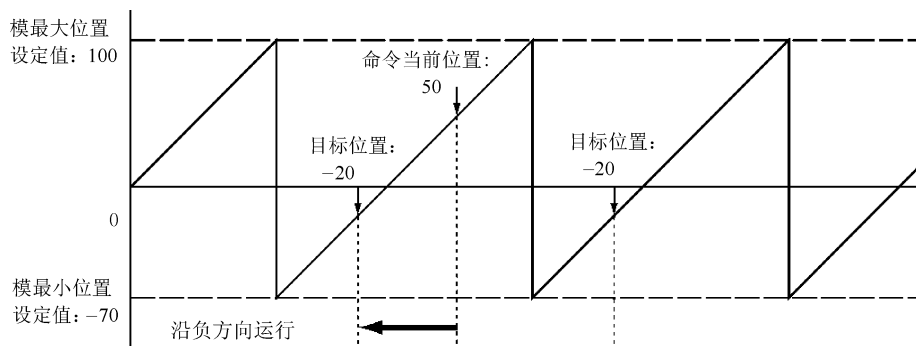


图 7-129 最短路径

图 7-130 为正方向，命令当前位置为 50，定位的目标位置为 -20。

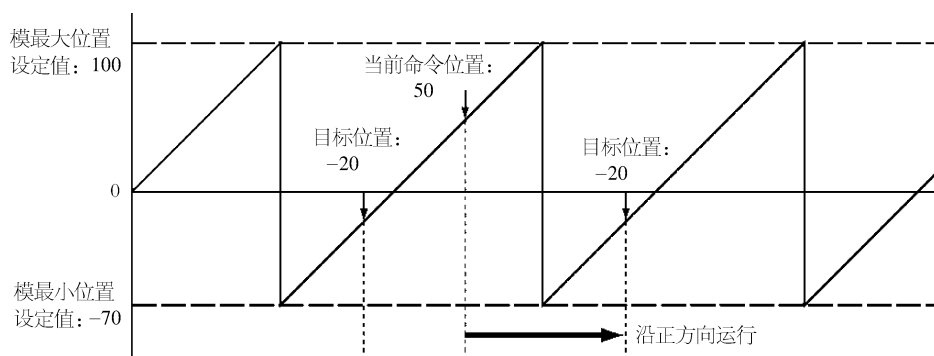


图 7-130 正方向

图 7-131 为负方向，命令当前位置为 50，定位的目标位置为 -20。

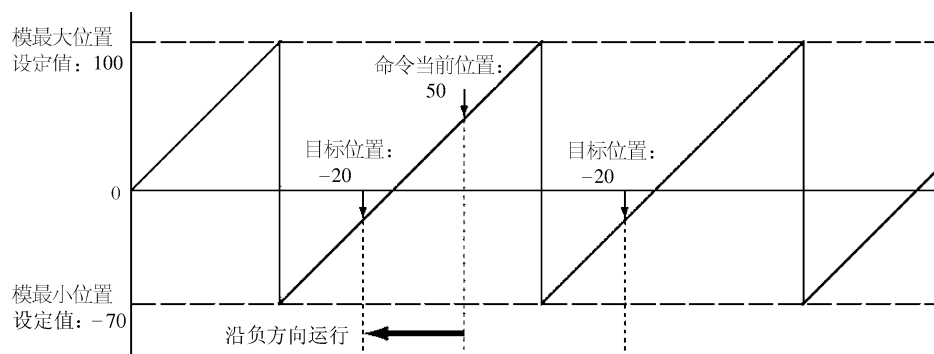


图 7-131 负方向

图 7-132 为当前方向，命令当前位置为 50，定位的目标位置为 -20。

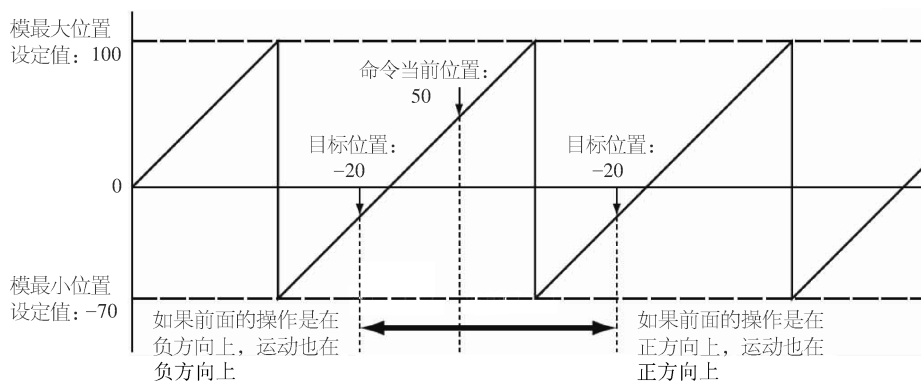


图 7-132 当前方向

图 7-133 为无方向 I 示意图, 命令当前位置为 50, 定位的目标位置为 -20。

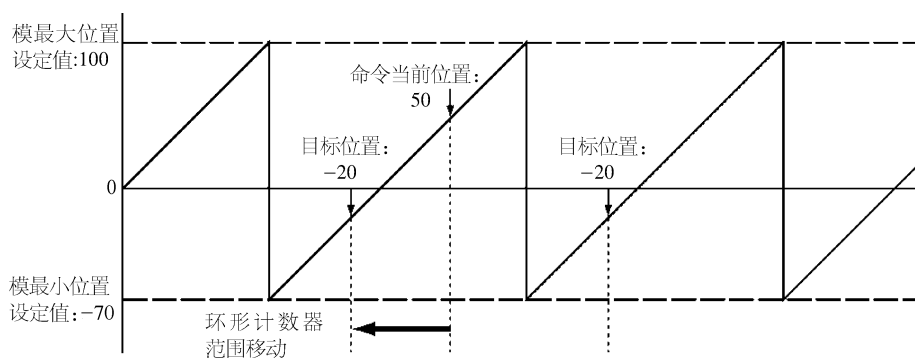


图 7-133 无方向 I

同样, 当环形计数器的上限为 100, 下限为 -70, 命令当前位置为 -20, 定位向着目标位置 290 执行, 如图 7-134 所示。

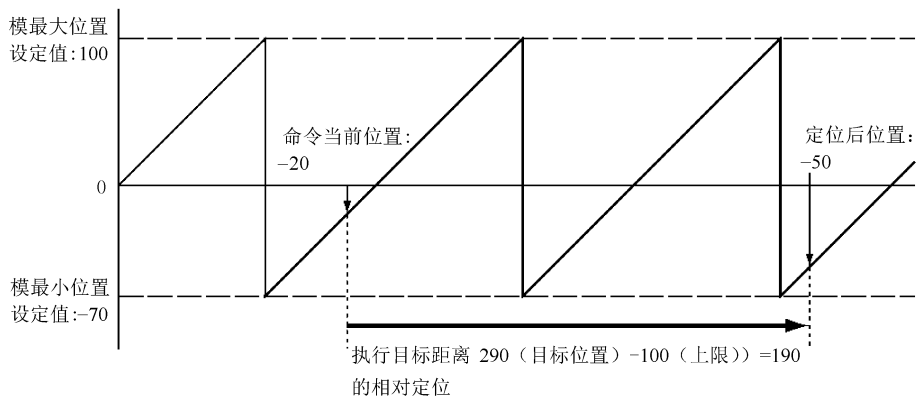


图 7-134 无方向 II

7.9.6 单轴运动控制指令的重新执行

单轴运动控制指令的 Re-execution (重新执行), 是指单轴运行期间, 修改运动控制指令的同一例程的输入变量, 重新执行该指令。重新执行时, 可修改的输入变量有 Position、Distance、Velocity、Acceleration、Deceleration 和 Torque, 有时其他输入变量也可以修改。如果修改一个不能修改的输入变量, 并尝试重新执行该指令, 则会出现指令错误。如果一条指令由于多次执行已经缓冲, 重新执行该指令, 缓冲区中指令的输入变量会改变。

1. 改变目标位置

如果重新执行时改变目标位置, 根据改变的时序和新的目标位置, 操作会改变。如果由于目标位置的改变, 运动方向相反, 改变方向后, 可以选择减速停止, 或转向后立即停止, 根据反向轴参数的操作选择。

(1) 新的命令值不改变方向

图 7-135 为恒速运行时重新执行命令。

图 7-136 为加速或减速期间重新执行指令。

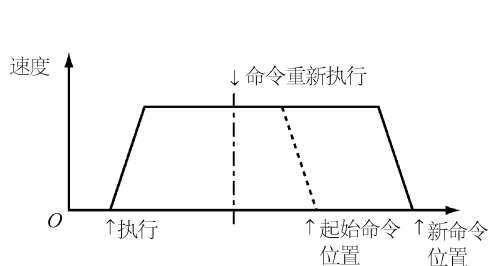
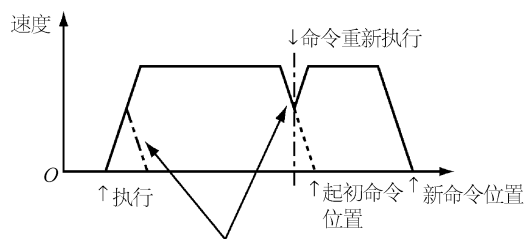


图 7-135 恒速运行时重新执行命令



如果在三角控制或者减速过程中重执行指令, 会再次加速到目标速度, 可能导致一个恒定速度。

图 7-136 加速或减速期间重新执行命令

(2) 新的命令值在反方向上出现

图 7-137 为减速停止后反转。

图 7-138 为反转后立即停止。

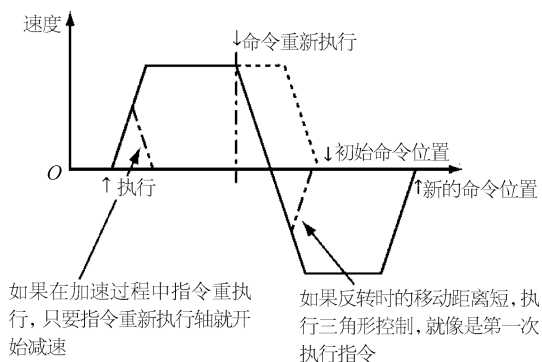


图 7-137 减速停止后反转

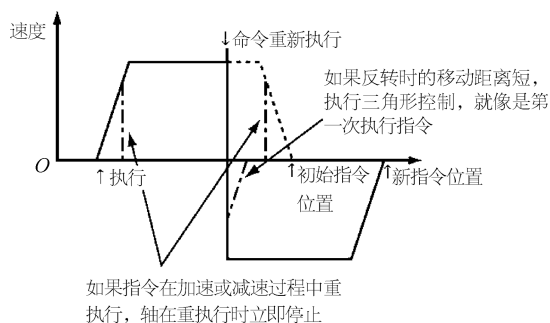


图 7-138 反转后立即停止

(3) 三角形控制模式

图 7-139 为三角形控制, 如果目标位置改变导致移动距离很短, 则产生三角形控制。

2. 改变移动距离

即使改变了移动距离和重新执行 MC_MoveRelative (相对定位) 指令, 参照运动首次启动时的位置, 完成新的移动距离。然而, 恰好在定位完成前再次执行指令, 作为一个新的指令而不是重新执行同样的指令。

图 7-140 为运动过程中重新执行指令。

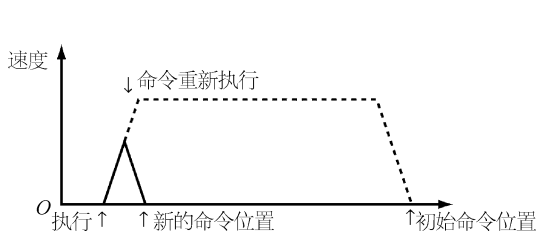


图 7-139 三角形控制

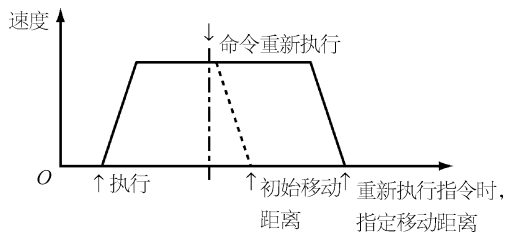
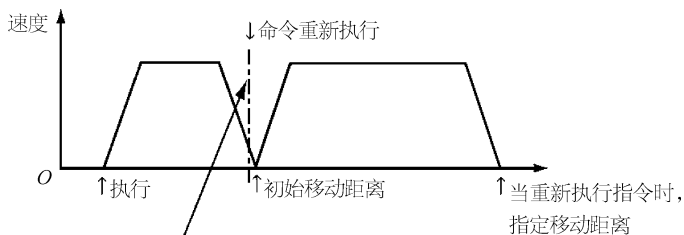


图 7-140 运动过程中重新执行指令

图 7-141 为定位刚结束重新执行指令。



如果指令刚好在定位结束时执行, 对于指令重新执行时指定的移动距离的定位, 有时是基于完成初始移动距离后轴所对应的目标位置

图 7-141 定位刚结束重新执行指令

3. 改变目标速度

只在加速 (包括三角形控制加速) 和恒速运动时改变操作。当轴减速时改变也可以接受, 但是运行不受影响。

4. 改变加速度

只在加速和三角形控制的加速时改变操作。当以恒定速度运行时, 如果改变, 改变的速率会覆盖加速度。当轴减速时改变也可以接受, 但是运行不受影响。

5. 改变减速度

只在加速、恒速运行、减速、三角形控制或过减速控制时, 减速度才改变。如果新的减速度引起轴超过目标位置, 停止在目标位置是最高的优先级。因此, 这种情况下, 实际的减速度会超过指定的减速度。

6. 改变扭矩命令

重新执行运动控制指令时, 基于扭矩斜率指标改变扭矩命令值。

图 7-142 为 MC_MoveAbsolute 指令的重新执行。绝对定位时, 目标位置从 1000 改为 2000, 变量 Axis1Pos 作为目标位置的输入参数。用 MOV 指令将目标位置指定为 1000, Axis1Execute 变为 TRUE, 开始定位。在操作过程中, 用 MOV 指令将目标位置修改为 2000, 并再次改变 Axis1Execute 为 TRUE, 切换到目标位置为 2000 的新定位操作。

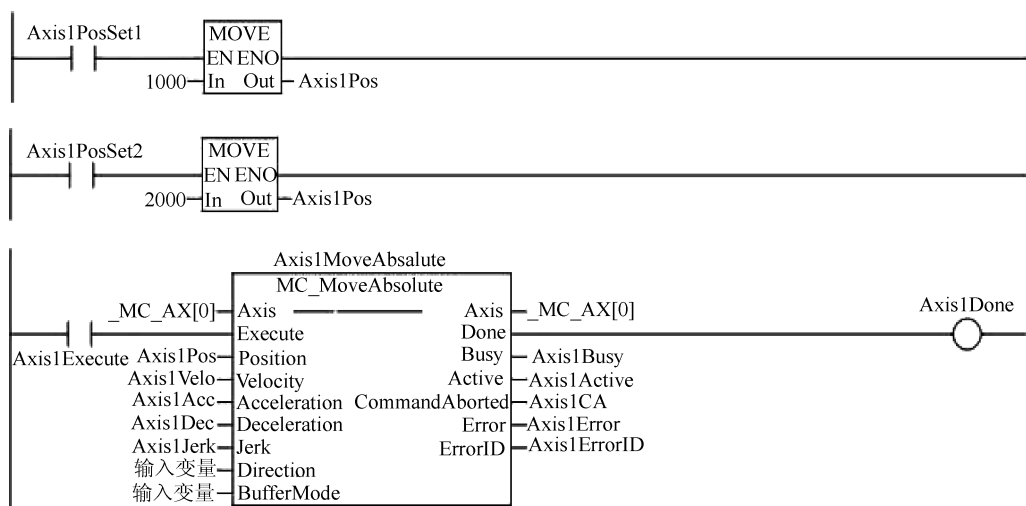


图 7-142 MC_MoveAbsolute 指令的重新执行

7.9.7 单轴运动控制指令的多次执行

当轴运行时，可以执行同一条运动控制指令的多个例程。在 PLCopen 技术规范中，这个功能被定义为 Buffer Mode (缓冲模式)，在 MC 功能模块中有时称之为指令的 Multi-execution (多次执行)。可以使用指令的多次执行功能，按顺序去多次执行运动控制指令，而无需停止整个运行。

可以为运动控制指令设置 BufferMode (缓冲模式选择) 输入变量，选择表 7-57 所列 6 种缓冲模式之一。这些模式的主要区别是缓冲指令执行的时序和过渡速度。

表 7-57 缓冲模式

缓冲模式	操作说明
Aborting(中止)	当前指令中止,多次执行的指令被执行
Buffered(缓冲)	当前指令的操作正常结束后,缓冲指令才被执行
Blending(混合)	当前指令的目标位置达到后,缓冲指令才被执行。该模式下,当前指令和缓冲指令之间的执行不停止。当前指令达到目标位置时,可以选择以下的过渡速度
Blending Low(Low Velocity)	过渡速度设置为当前指令和缓冲指令的目标速度中最低的一个
Blending Previous(Previous Velocity)	当前指令的目标速度用作过渡速度
Blending Next(Next Velocity)	缓冲指令的目标速度用作过渡速度
Blending High(High Velocity)	过渡速度设置为当前指令和缓冲指令目标速度中最高的一个

在 MC 功能模块中，多次执行的指令是缓冲的。无论 Buffered (缓冲) 还是 Blending (混合) 模式，都以指定 BufferMode 的时序和过渡速度执行。每一个轴都有缓冲，如果设置为 Aborting (中止)，后面执行的指令立即执行，不再有缓冲。

每一个轴只能有一条多次执行指令被缓冲。如果对两条或多条指令多次执行，会出现指令错误。

作为单轴运行的多轴，不能进行多轴协调控制指令 (轴组指令) 的多次执行。同样，不

能将单轴控制指令的多次执行用到处于多轴协调控制下的各个轴。如果违反这些规则，会出现指令错误。

1. Aborting (中止)

这是默认模式，这种模式下，没有缓冲。当前命令被中断，执行新的指令。无论单轴控制还是同步控制，中断模式都可用到指令的多次执行。

当前命令的 BufferMode 为 Aborting，多次执行的命令中断当前命令的执行，以下为当前命令执行转换为缓冲命令执行的各种情况。

(1) 多次执行指令的命令位置不改变方向

图 7-143 为运行处于恒速过程中，多次执行指令。

图 7-144 为运动处于加速/减速过程中，多次执行指令。

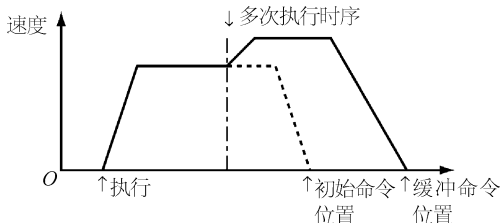


图 7-143 恒速过程中的多次执行

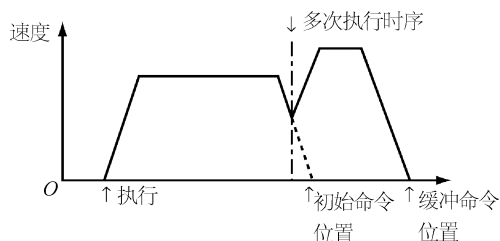


图 7-144 加速/减速过程中的多次执行

(2) 多次执行指令的命令位置在反方向上

如图 7-145 所示，反转后减速停止。

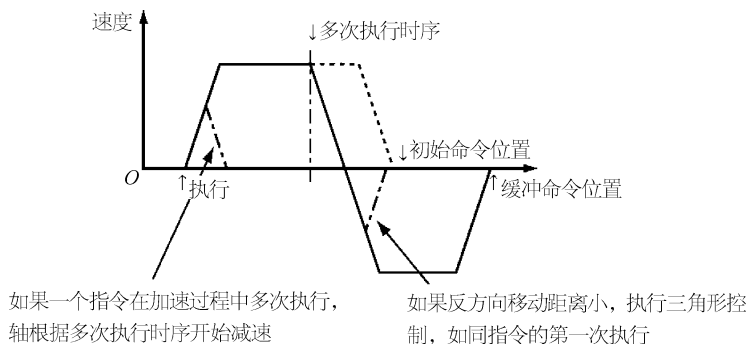


图 7-145 反转后减速停止

如图 7-146 所示，反转后立即停止。

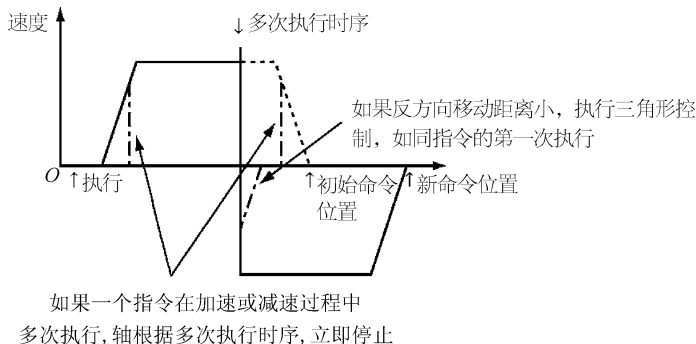


图 7-146 反转后立即停止

2. Buffered (缓冲)

缓冲指令在缓冲区中保留，直到当前指令的操作完成。当前指令正常结束时，缓冲指令才执行，如图 7-147 所示。

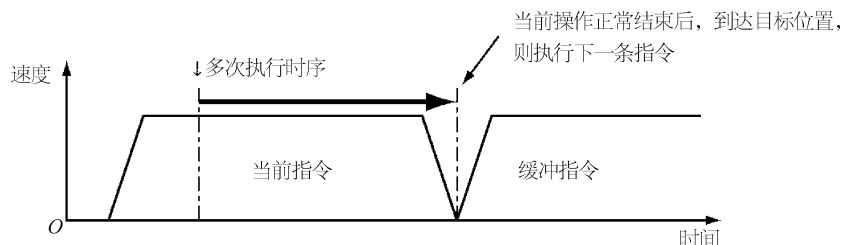


图 7-147 缓冲指令执行

如图 7-148 所示，MC_MoveRelative 指令多次执行，例程 Move_Axis_1、Move_Axis_2、Move_Axis_3 的 BufferMode 分别为 Aborting、Buffered、Buffered。例程 Move_Axis_1 执行时，例程 Move_Axis_3 不能执行。当 Go_Move_1 变为 ON 时，例程 Move_Axis_1 执行。此时，如果 Go_Move_2 变为 ON 时，因为例程 Move_Axis_1 的 BufferMode 为 Aborting，则 Move_Axis_1 的执行中止，例程 Move_Axis_2 开始执行。如果 Go_Move_3 变为 ON，因为例程 Move_Axis_2 的 BufferMode 为 Buffered，则例程 Move_Axis_3 必须等到例程 Move_Axis_2 执行完毕才能执行。

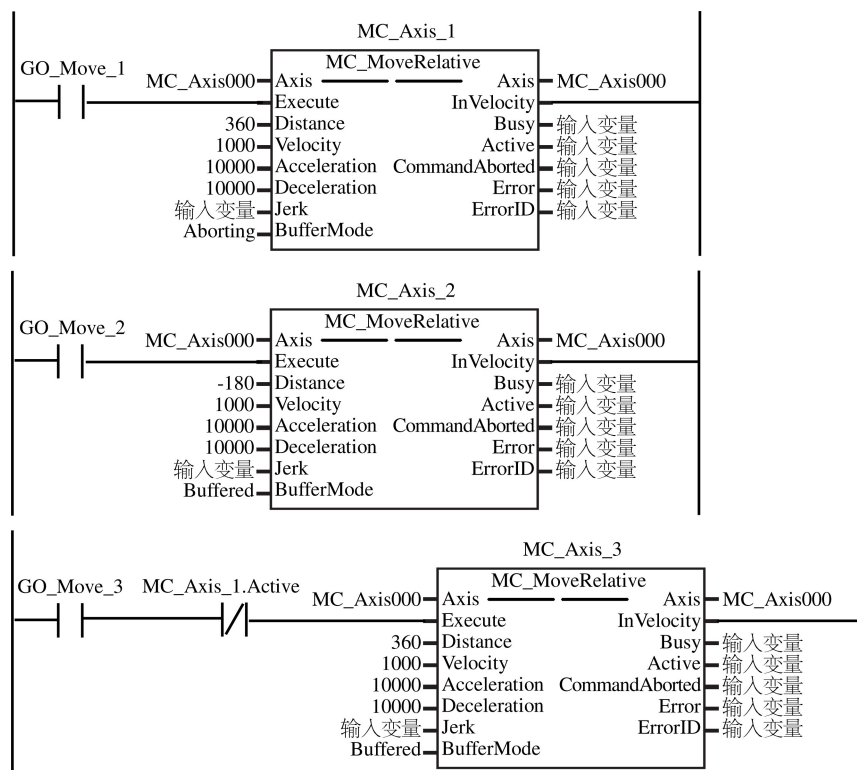


图 7-148 MC_MoveRelative 指令的多次执行

3. Blending (混合)

当前指令处于执行状态, 缓冲指令在缓冲区中保留, 当前指令的目标位置达到后, 缓冲指令才执行。然而, 这时运动不停止, 以 BufferMode 输入变量指定的速度, 操作过渡到下一条指令。对于相对移动, 最终的位置是两条指令总的数值。对于绝对移动, 最终的位置是第二次多次执行指令的目标位置。当以轴参数设置的最大加速度和最大减速度操作, 超过目标位置时, 轴参数中的加速/减速选择下列之一:

- 使用快速加速/减速 (Blending 变为 Buffered);
- 使用快速加速/减速;
- 轻微故障停止。

在 Blending 模式下, 不能将单轴和同步控制结合起来。

(1) Blending Low (Low Velocity)

操作使用当前指令的目标位置、当前指令和缓冲指令中目标速度最小的速度。

(2) Blending Previous (Previous Velocity)

以当前指令的目标速度, 达到当前指令的目标位置。一旦当前指令的目标位置达到, 经过加速/减速, 缓冲指令的目标速度达到。

图 7-149 为不改变方向的 Blending Previous, 其中图 7-149a 为加速情况, 图 7-149b 为减速情况。

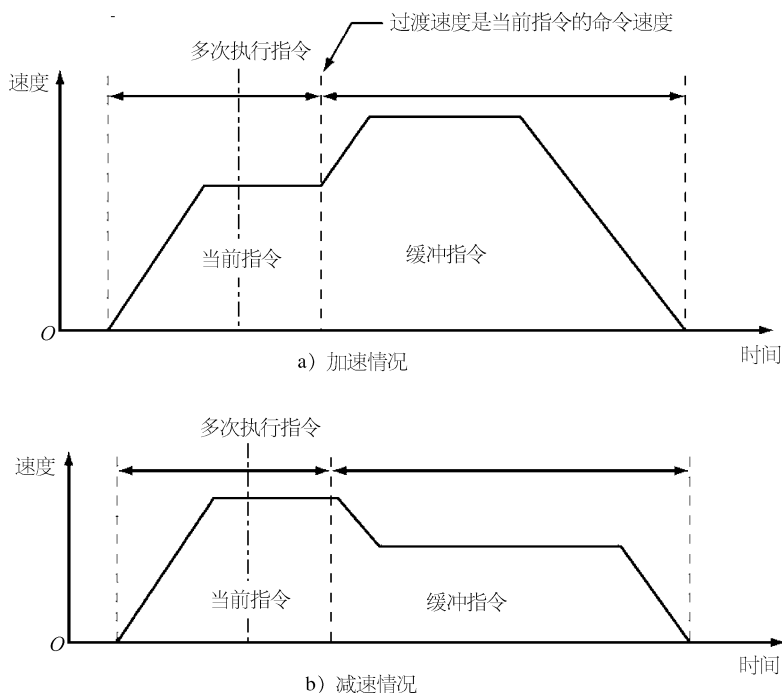


图 7-149 不改变方向的 Blending Previous

图 7-150 为改变方向的 Blending Previous。

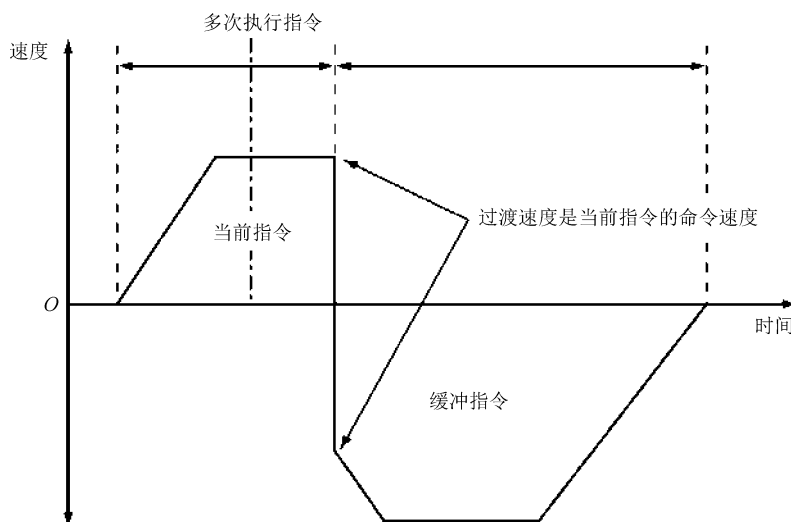


图 7-150 改变方向的 Blending Previous

(3) Blending Next (Next Velocity)

图 7-151 使用当前指令的目标位置和缓冲指令的目标速度，其中，图 7-151a 为缓冲指令的目标速度大于当前指令的，因此要进行加速；图 7-151b 为缓冲指令的目标速度小于当前指令的，因此要进行减速。

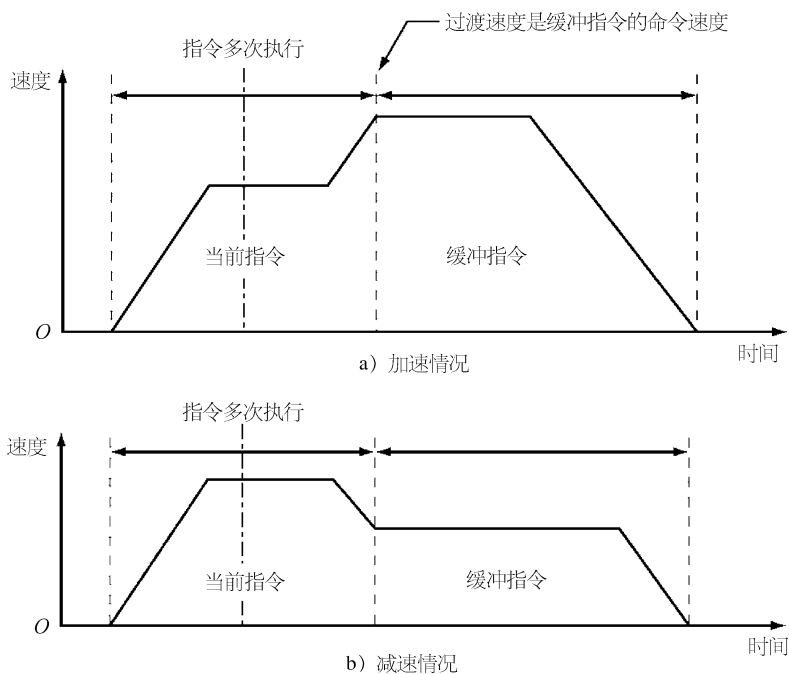


图 7-151 Blending Next

(4) Blending High (High Velocity)

操作使用当前指令的目标位置、当前指令与缓冲指令中最高速度作为目标速度。

7.9.8 多轴协调控制

多轴协调控制时,从 Sysmac Studio 预先设置轴组,通过 MC 功能模块完成多轴插补控制,轴组操作过程如图 7-152 所示。

1. 使能和禁用轴组

为了使能轴组,用 MC_GroupEnable 指令指定轴组。当执行轴组指令时,如果轴组仍然未使能,会出现指令错误。为了禁用轴组,用 MC_GroupDisable 指令指定轴组。当禁用处于运行状态的轴组时,轴组中的所有的轴以轴参数中指定的最大减速度减速停止。

2. 复位轴组错误

如果在轴组中出现错误,一旦消除原因,可以使用 MC_GroupReset 指令消除错误。

3. 直线插补

直线插补是在起始点和结束点之间,在一条直线上有 2~4 条逻辑轴 A0~A4 移动。既可以绝对定位也可以相对定位。可以指定插补速度、插补加速度、插补减速度和加加速度等。

MC 功能模块使用下列三种直线插补指令。

- MC_MoveLinear (直线插补)
- MC_MoveLinearAbsolute (直线绝对插补)
- MC_MoveLinearRelative (直线相对插补)

4. 圆弧插补

圆弧插补是在 2D 平面内,沿着圆弧移动 A0~A3 中的两轴。可以对两轴指定圆弧插补模式、路径方向、插补速度、插补加速度、插补减速度和组合加加速度。

MC 功能模块使用圆弧插补指令 MC_MoveCircular2D,可以用输入变量 Circular 指定圆弧三种插补方法:边界点、中心和半径。

5. 轴协调控制下的停止

(1) 用运动控制指令停止

用 MC_GroupStop 或 MC_GroupImmediateStop 指令停止轴组运行。

1) MC_GroupStop 指令:对于轴组中完成的直线插补和圆弧插补,沿着控制路径减速停止。可以指定减速度和 Jerk。指定减速度为 0 时,发送命令立即停止伺服驱动。指定 0 减速度发送出一个立即停止伺服驱动的命令。当这条指令减速停止同时输入变量 Execute 为 TRUE 时,其他操作命令不会得到应答。

2) MC_GroupImmediateStop 指令:对于轴组中所有的轴实施立即停止。MC_GroupImmediateStop 也可以对正在执行 MC_GroupStop 减速停止的轴组实施立即停止。

(2) 由于出错或其他问题停止

1) 轴组运行因出错停止。

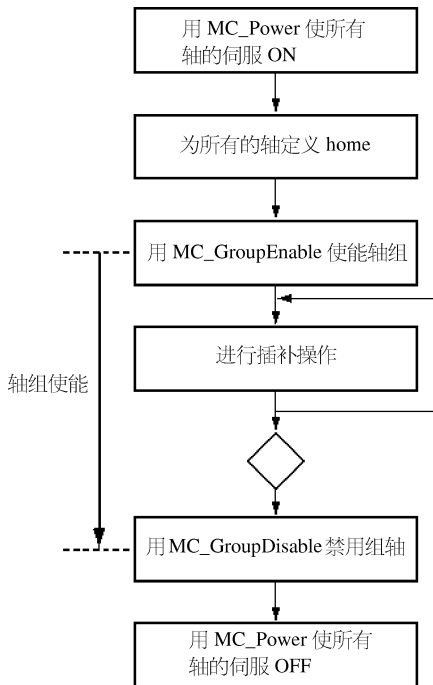


图 7-152 轴组操作过程

- 2) 由于运行控制周期过长停止。
- 3) 由于启动 MC 试运行引起的停止。
- 4) 由于 CPU 单元运行模式改变引起的停止。

6. 多轴协调控制的倍率

使用 MC_GroupSetOverride (设置轴组倍率) 指令为当前正在插补运行的多轴协调控制轴组设置倍率因子。倍率因子设置为插补目标速度的百分数。设置范围为 0% ~ 500%，如果插补目标速度的倍率因子为 0%，运行状态继续，轴停止在 0 速度上。只要倍率因子使能，则倍率因子就能被读出。如果倍率因子禁用，倍率因子返回 100%。当倍率因子改变时，如果超出最大插补速度，使用轴组最大插补速度。

7.9.9 多轴协调控制的通用功能

1. 多轴协调控制的速度

指定多轴协调控制的速度，指定路径上的插补速度。单位与单轴的相同，都是命令单位/s。

(1) 速度类型

MC 功能模块支持的轴组插补速度只有一种类型，即命令插补速度的实际值。

(2) 与速度相关的轴参数

与速度相关的轴参数见表 7-58。

表 7-58 与速度相关的轴参数

参数名称	功 能	设 定 值	默 认 值
最大插补速度	为路径设置最大插补速度。无插补速度限制设置为 0。如果目标速度超出了为一个轴组操作指令指定的最大插补速度，轴将以最大插补速度运行	非负 LREAL	800000000
插补速度警告值	设置最大插补速度的百分比以输出一个插补速度警告。设置为 0 时无插补速度警告输出(单位:%)	0 ~ 100	0

(3) 指定轴操作的目标速度

在实际定位运动中使用的插补速度由运动控制指令的 Velocity 输入变量指定。

(4) 监视速度

在用户程序中读轴组变量监视插补速度，命令插补速度的当前值由变量_MC_GPR[0 - 63].Cmd.Vel 反映，沿正方向运动时为正值，沿负方向运动时为负值。

2. 多轴协调控制下的加速和减速

(1) 与插补加速和插补减速相关的轴参数

与插补加速和插补减速相关的轴参数见表 7-59。

表 7-59 与插补加速和插补减速相关的轴参数

参数名称	功 能	设定范围	默 认 值
最大插补加速度	为轴的操作指令设置最大加速率, 设为 0 时最大加速率不会有限制(单位: 命令单位/s ²)	非负长实数	0

(续)

参 数 名 称	功 能	设定范围	默 认 值
最大插补减速度	为轴的操作指令设置最大减速率, 设为 0 时最大减速率不会有限制(单位: 命令单位/s ²)	非负长实数	0
插补加速度/减速度超出	轴组加速/减速控制时, 因为停在目标位置的优先级是特定的, 导致过度加速/减速, 会出现超出最大加速度/减速度, 在上述情况下, 设置操作: 0: 使用快速加速/减速(混合变为缓冲) 1: 使用快速加速/减速 2: 轻微故障停车	0 ~ 2	0
插补加速度警告值	设置最大加速度的百分比, 为轴输出加速度的警告值, 设为 0 时没有输出加速度警告(单位: %)	0 ~ 100	0
插补减速度警告值	设置最大减速度的百分比, 为轴输出减速度的警告值, 设为 0 时没有输出减速度警告(单位: %)	0 ~ 100	0

(2) 为轴组指定一个插补加速度和插补减速度

用于实际定位运动的插补加速度和插补减速度, 由运动控制指令的 Acceleration 输入变量和 Deceleration 输入变量指定。

(3) 监视插补加速度和插补减速度

在用户程序中读轴组变量可以监视插补加速度和插补减速度。变量为_MC_GPR[0 ~ 31].Cmd.AccDec, LREAL 型, 这是命令插补加速度/减速度的当前值。加速时加一个加号, 减速时加一个减号。

3. 多轴协调控制的 Jerk

对于多轴协调控制, 加加速度用来减少机械的冲击, 沿着 S 形曲线插补路径平滑插补加速度/减速度。多轴协调控制的 Jerk, 其作用与单轴的 Jerk 相似, 这里不再介绍。

7.9.10 多轴协调控制运动控制指令的重新执行

如果重新执行直线插补或圆弧插值指令, 会出现指令错误。

如果重新执行 MC_GroupStop 指令, 可以改变减速度, 但不能用这种方式改变 Jerk。

如果重新执行 MC_GroupReset 指令, 重新执行的命令会被忽略, 但错误复位过程继续。

7.9.11 多轴协调控制运动控制指令的多次执行

如图 7-153 所示, 在轴组的多轴协调控制时, 可以像轴操作一样完成多次执行。如果使用缓冲模式, 对于多轴协调控制, 可以完成多条连续的直线、圆弧路径控制。

可以为运动控制指令设置 BufferMode 输入变量, 选择与单轴操作支持的相同的缓冲模式。对于轴组总计有 8 个指令缓冲, 每个轴组有对当前正在运行指令的一个缓冲和对多次执

行指令的 7 个缓冲。

对于单个轴组，同一时间，7 个指令可以被缓冲。如果 8 个或更多多次指令被执行，会出现指令错误。

作为单轴正处于操作状态的轴，不能用于多轴协调控制指令（轴组指令）的多次执行。同样，已用于多轴协调控制正处于操作状态的轴，不能进行单轴控制指令的多次执行。否则，会出现指令错误。

和单轴指令一样，轴组指令也有如下几种缓冲模式：

- (1) Aborting
- (2) Buffered
- (3) Blending
 - 1) Blending Low (Low Velocity)
 - 2) Blending Previous (Previous Velocity)
 - 3) Blending Next (Next Velocity)
 - 4) Blending High (High Velocity)

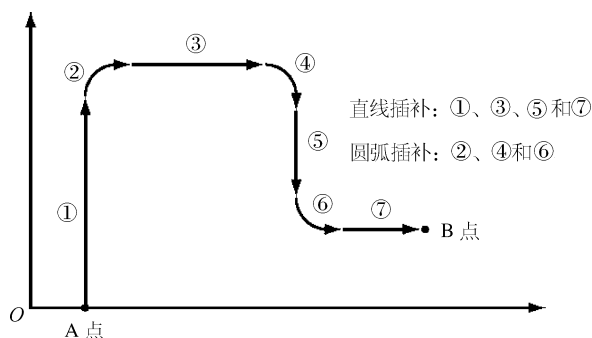


图 7-153 轴组指令的多次执行

7.9.12 多轴协调控制运动控制指令的过渡模式

由于插补路径的方向改变，轴组指令的多次执行可能会对设备、工件产生一些冲击。对于运动控制指令，可以通过指定 TransitionMode（过渡模式）输入变量，来选择指令间使用的过渡方法，以减少这种冲击。在 MC 的功能模块，可以从下面的过渡模式中选择 TransitionMode。

1. Transition Disabled (0: TMNone)

Transition Disabled（过渡禁用），即过渡时不做任何处理，这是默认的方式，不能减少冲击，但是减少了运行时间。

选用这种模式，两个位置之间的连接不做任何处理。

(1) TransitionMode = TMNone and BufferMode = Buffered

如图 7-154 所示，轴运动到位置 End1，停止，接着运动到位置 End2。

(2) TransitionMode = TMNone and BufferMode = Blending

如图 7-155 所示，轴运动到位置 End1，接着运动到位置 End2。

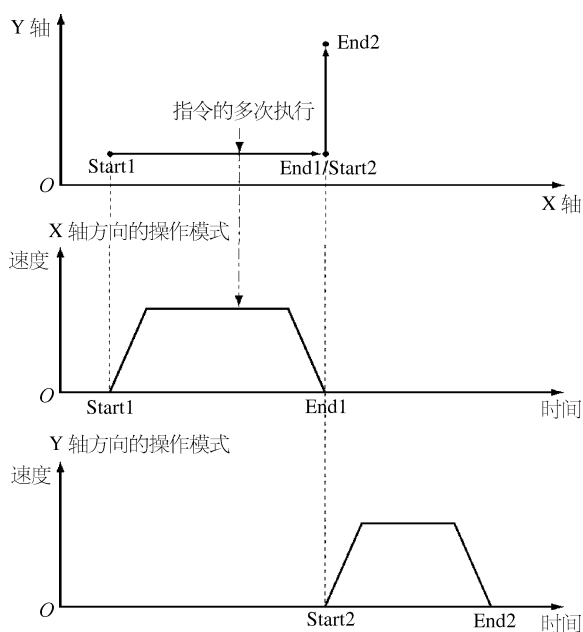


图 7-154 过渡模式为 TMNone、缓冲模式为 Buffered

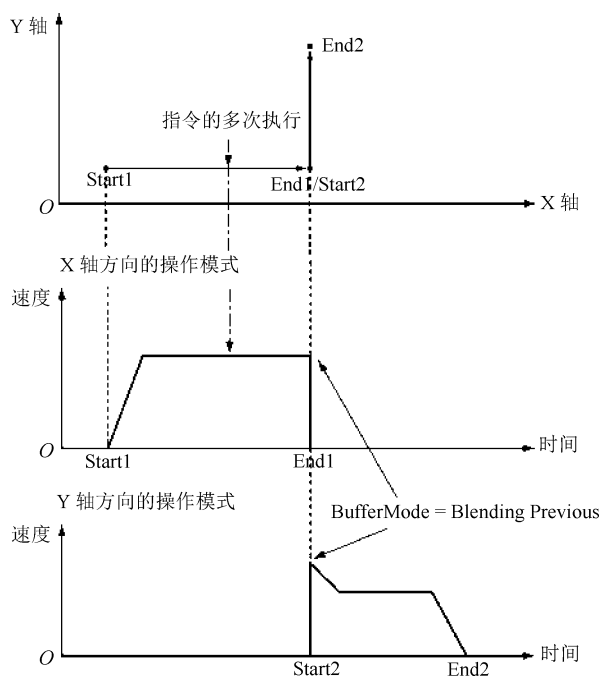


图 7-155 过渡模式为 TMNone、缓冲模式为 Blending

(3) TransitionMode = TMNone and BufferMode = Aborting

如图 7-156 所示, 轴从 End1' (指令的多次执行) 运动到 End2。

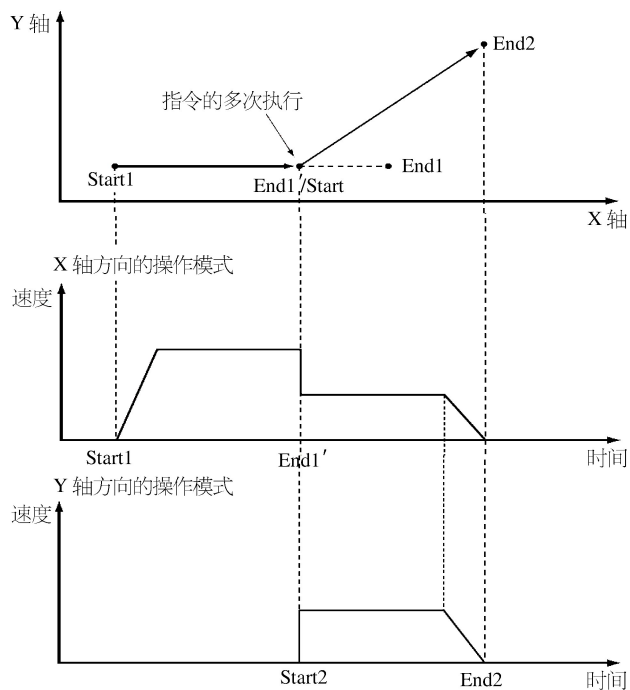


图 7-156 过渡模式为 TMNone、缓冲模式为 Aborting

2. Superimpose Corners (10: TMCornerSuperimpose)

Superimpose Corners (叠加拐角) 如图 7-157 所示, 当前指令的减速叠加到缓冲指令的加速上, 对于缓冲指令, 不管指定什么样的加速, 操作是在当前指令减速的同样时间内执行。即使设定了 Jerk, 叠加区域里也不使用 Jerk。

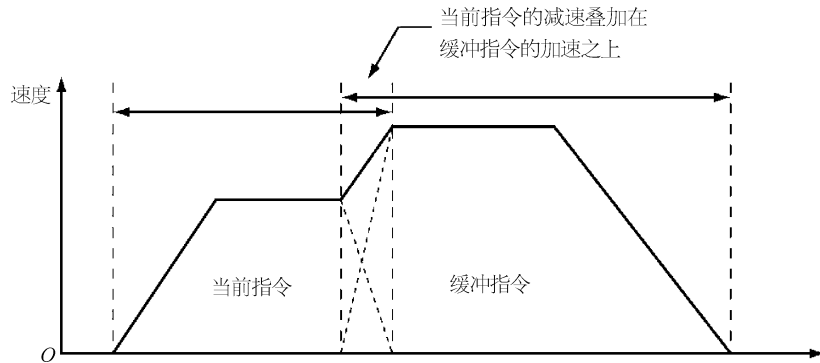


图 7-157 叠加拐角

对于叠加拐角, 表示运动指令执行结束的输出变量 Done, 在叠加区域执行完后, 会变为 TRUE。

第 8 章 NJ 的 EtherNet/IP 网络

8.1 概述

EtherNet/IP (EIP) 是一种多厂商工业以太网。EtherNet/IP 规范与 DeviceNet 一样，也是由 ODVA (Open DeviceNet Vendor Association) 管理的开放标准。EtherNet/IP 不仅是一种连接控制器的网络，也能当做一种现场总线来使用。因为 EtherNet/IP 使用标准的 EtherNet 技术，各种通用的 EtherNet 设备都可以应用在网络中。

EtherNet/IP 网络中的节点除了内置 EIP 端口的 NJ 系列 CPU 单元外，可另外选择内置 EIP 端口的 PLC CPU 单元 CJ2H-CPU□□-EIP 或 CJ2M-CPU3□□-EIP，以及内置 EIP 端口的 CJ2 CPU 单元 (有的 CJ2 CPU 无 EIP 端口)，还有专用的 EIP 单元如 CJ 系列的 CJ1W-EIP21、CS 系列的 CS1-EIP21。EtherNet/IP 传输介质为屏蔽双绞线。图 8-1 为 NJ 系列控制器组成的简单 EtherNet/IP 网络。

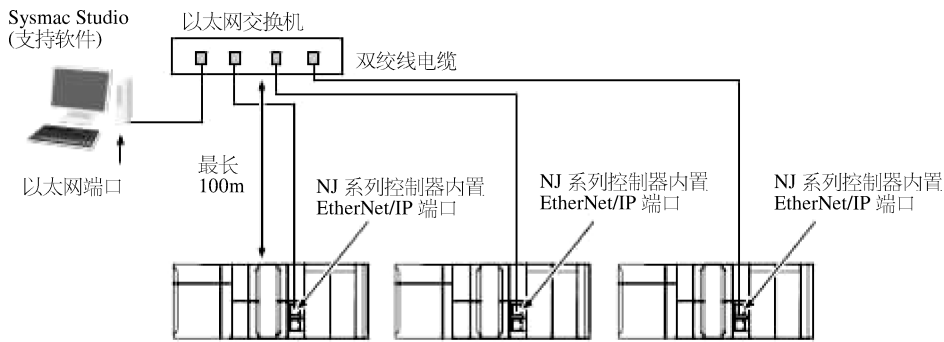


图 8-1 NJ 系列控制器组成的简单 EtherNet/IP 网络

内置 EtherNet/IP 端口的性能指标见表 8-1。

表 8-1 内置 EtherNet/IP 端口的性能指标

项 目		规 格
通信协议		TCP/IP 或者 UDP/IP
支持的服务		Sysmac Studio 连接, 标签数据链接, CIP 信息通信, Socket 服务, FTP 服务器, 自动时钟调整(NTP 客户端), SNMP 代理, DNS 客户端和 BOOTP 客户端
物理层		100Base-TX 或者 10Base-T(推荐使用 100Base-TX)
传输规格	介质访问方法	CSMA/CD
	调制	基带
	传输路径	星形

(续)

项 目		规 格
传输规格	波特率	100Mbit/s(100Base-TX)
	传输介质	屏蔽双绞线(STP)电缆,类 5,5e 或更高
	传输距离	最长 100m(集线器与节点之间的距离)
	级联连接数目	没有限制(如果用以太网交换机)
CIP 服务:标签 数据链接(循环通 信)	连接数	32
	数据包间隔(刷新周期)	10 ~ 1000ms,以 1ms 为增量 可为每个连接独立设置数据包的时间间隔(网络中数据在 预设的时间间隔内刷新,不依赖于节点的数目)
	允许的每单元通信带宽	1000pps (包括 heartbeat)
	可登记标签的数量	256
	标签类型	网络变量 CIO, Work, Holding, DM 或 EM 区
	每个链接(即一个标签组)的标签数	8(当标签组包含控制器状态时为 7)
	每个节点链接数据的最大容量	19200 字节
	每个链接的最大数据容量	600 字节
	可登记标签组的数量	32(1 个连接 = 1 个标签组)
	一个标签组的最大容量	300 字(当标签组包含控制器状态时,控制器状态占用 1 个 字)
	当控制器处于运行模式时,改变标签 数据链接参数	支持
	组播数据包滤波器	支持
CIP 信息服务: 显式信息	3 类(连接)	连接数:32(客户 + 服务器)
	UCMM(无连接)	一次可以通信的客户数目:最多 32 一次可以通信的服务器数目:最多 32
SNMP	代理	SNMPv1 或者 SNMPv2c
	MIB	MIB-II
EtherNet/IP 一致性测试		符合 A8
Ethernet 接口		10Base-T 或 100Base-TX 自动协商或固定的设置

注意:

1) 使用数据链接时,物理层选择 100Base-TX。

2) pps 即“每秒包”,表示每秒钟可以处理的数据包的数目。

3) 一个链接的数据容量为 505 字节或更大时,系统必须支持一个大的正向开放(可选的 CIP 规格)。CS/CJ 系列单元支持大的正向开放,但是连接其他公司的节点之前,确认这些设备也支持。

4) 如果内置 EtherNet/IP 端口的参数改变, 端口需重新启动。当其他节点与受影响的节点通信时, 通信会暂时超时, 随后自动恢复。

5) 因为内置 EtherNet/IP 端口装备了 IGMP 客户端 (版本 2), 通过支持 IGMP 监听的 Ethernet 交换机, 不需要的多播数据包被滤掉。

8.1.1 内置 EtherNet/IP 端口的地址

内置 EtherNet/IP 端口的地址为 12 位十六进制数, 在 CPU 单元的两个位置上有显示, 如图 8-2 所示。

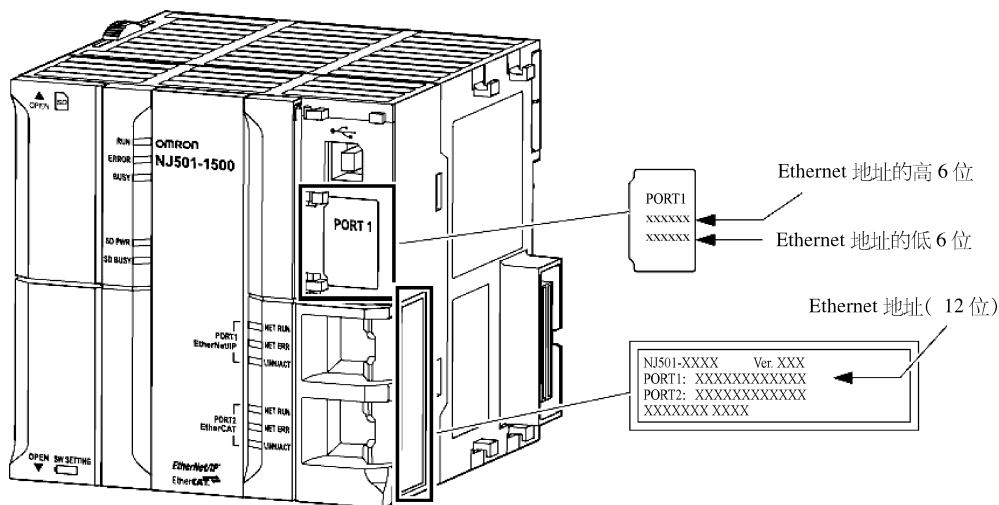


图 8-2 内置 EtherNet/IP 端口的地址

8.1.2 内置 EtherNet/IP 端口的指示灯

内置 EtherNet/IP 端口的指示灯如图 8-3 所示。

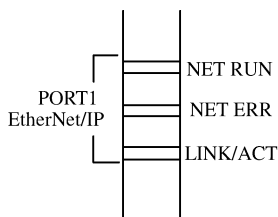


图 8-3 内置 EtherNet/IP 端口的指示灯

内置 EtherNet/IP 端口指示灯含义见表 8-2。

表 8-2 内置 EtherNet/IP 端口指示灯含义

指示灯	颜色	状态	运行状态
NET RUN	...	不亮	以太网通信不能进行 ● 电源 OFF 或者控制器正在复位 ● MAC 地址错误或通信控制器发生错误

(续)

指示灯	颜色	状态	运行状态
NET RUN	绿色	闪烁	以太网通信正在进行 ● 标签数据链接的连接正在建立(源操作) ● 正在通过 BOOTP 获得一个 IP 地址
		亮	正常运行 如果目标只是设为数据链接,不管来自源的连接是否建立,这个指示灯亮,即使数据链接停止,仍保持亮
NET ERR	红色	不亮	不存在以太网通信错误 ● 电源 OFF 或者控制器正在复位
		闪烁	出现用户可恢复的错误 ● 在 TCP/IP 通信或者 CIP 通信中出现错误 ● FTP 服务器设置错误,NTP 服务器设置错误等 ● 数据链接设置错误,标签数据链接校验错误等
		亮	用户发生不可恢复的错误 ● MAC 地址错误或者通信控制器错误
LINK/ACT	黄色	不亮	链接没有建立 ● 电缆未连接 ● 电源 OFF 或者控制器正在复位
		闪烁	链接建立之后,数据通信在进行中
		亮	链接已建立

8.1.3 EtherNet/IP 网络的连接

如图 8-4 所示, EtherNet/IP 网络用屏蔽双绞线电缆连接, 不要将电缆外层的屏蔽线接到内置 EtherNet/IP 端口的连接器上, 如果用电线连接两个 Ethernet 交换机, 只在其中一端接屏蔽线。

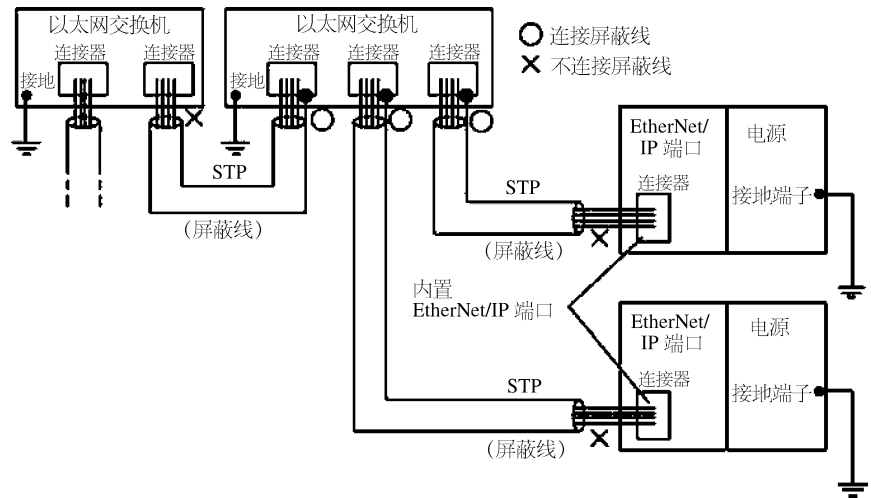


图 8-4 EtherNet/IP 的屏蔽双绞线电缆连接

图 8-5 为 Ethernet 交换机的连接, MDI 端口和 MDI-X 端口用直通电缆, 两个 MDI 端口或两个 MDI-X 端口用交叉电缆。

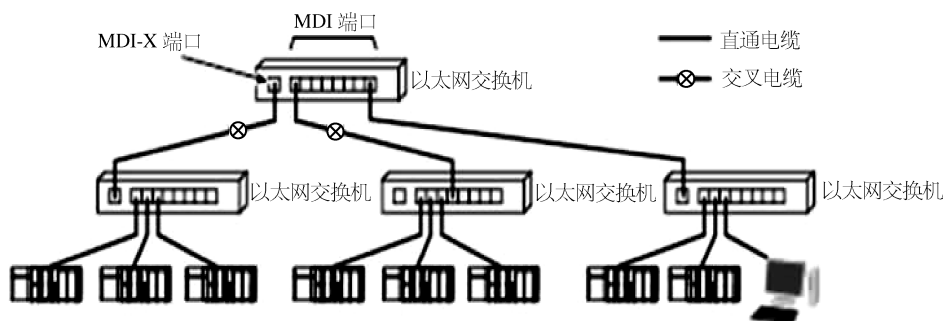


图 8-5 Ethernet 交换机的连接

内置 EtherNet/IP 端口的连接器为 RJ45, 接线方法见图 6-7, 各引脚的含义见表 6-5。

8.1.4 与内置 EIP 端口相关的系统定义变量

与内置 EIP 端口相关的系统定义变量共有三类: EIP 通信错误、EIP 通信状态和 EIP 通信开关。

反映 EIP 通信错误的系统定义变量如图 8-6 所示, 在用户程序中可以使用它们检查 EIP 端口的错误, 如 Sysmac Studio 设置错误、Network Configurator 设置错误、TCP/IP 应用错误 (如 FTP 或 NTP) 等。

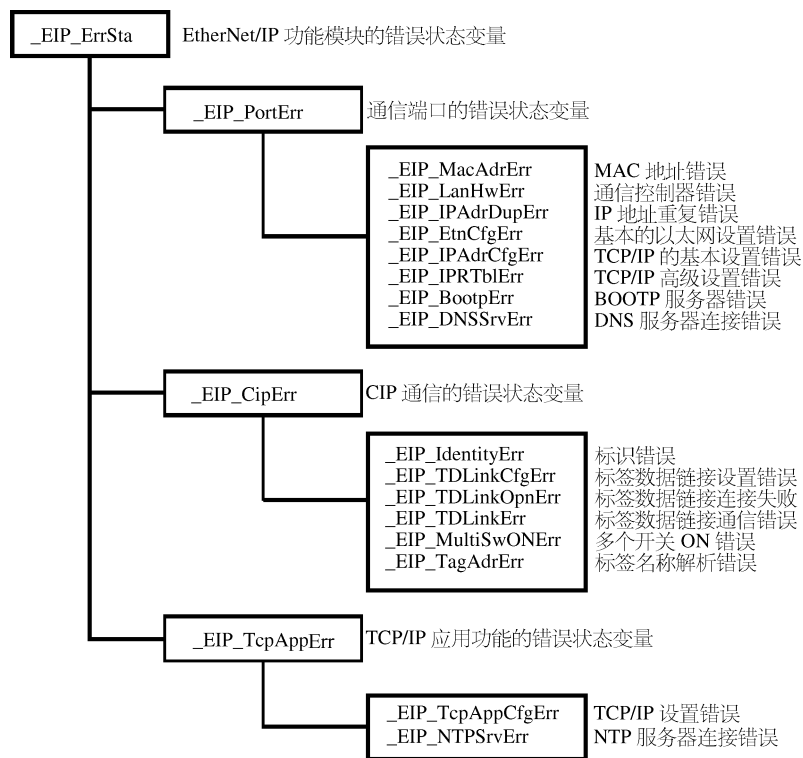


图 8-6 反映 EIP 通信错误的系统定义变量

8.1.5 内置 EtherNet/IP 端口的设置

使用内置 EtherNet/IP 端口时，需要用 Sysmac Studio 进行设置，图 8-7 为 EtherNet/IP 端口设置的开始界面，设置的项目有：TCP/IP、Link、FTP、NTP、SNMP、SNMP Trap 和 FINS，这些项目的详细内容和设置步骤见相关手册，此处不做进一步介绍。

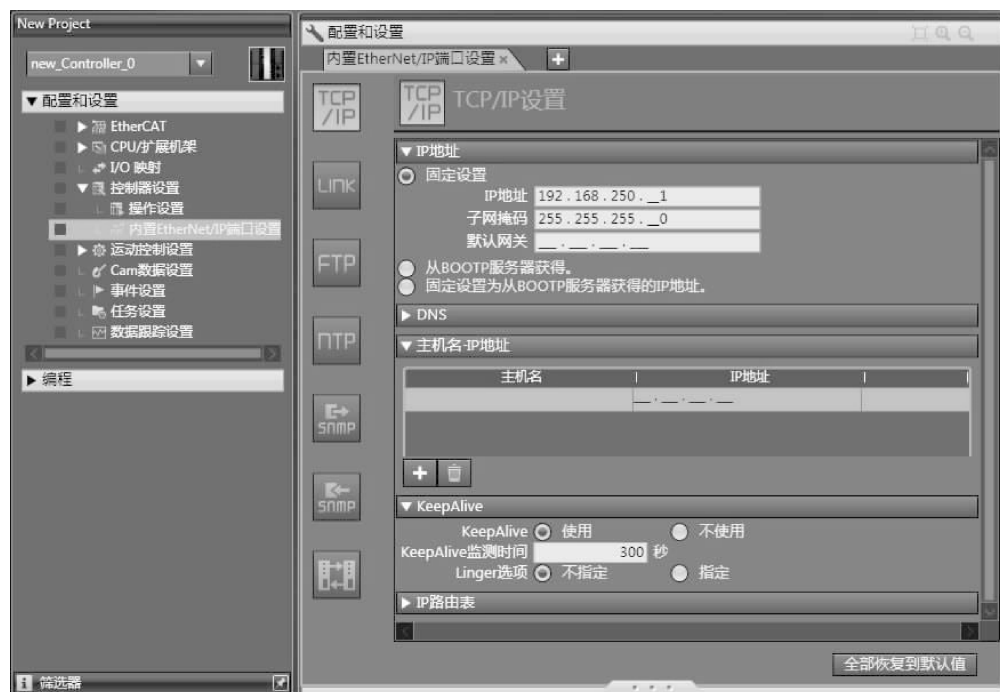


图 8-7 EtherNet/IP 端口设置的开始界面

8.1.6 NJ 系列 CPU 单元内置 EIP 端口的特性

1. 标签数据链接

通过数据链接实现高速、高容量数据交换。EtherNet/IP 协议支持隐式通信，允许 EIP 设备周期性地交换数据，称为标签数据链接。在控制器和设备之间，可以交换多达 9600 字数据。

不管节点的数目多少，每个应用的标签数据链接（循环通信）在指定的循环周期内完成。在每个链接指定的刷新周期内，网络上的数据交换完成。即使节点数增加，通信的刷新周期也不会增加，也就是链接数据的并发性得到保持。因为可以为每一个链接设置刷新周期，每一个应用都可以在它的理想刷新周期内通信。例如，进程间互锁以高速传送，而生产命令和状态监视信息以低速传送。

2. CIP 信息通信

CIP (Common Industrial Protocol) 通用工业协议，是 OSI 应用层一个共享的工业协议，CIP 应用于 EtherNet/IP、CompoNet 和 DeviceNet 网络中。通过执行程序中 CIP 通信指令，向网络上的设备发送 CIP 命令，从而实现数据的发送和接收。

3. BOOTP 客户

如果 NJ 内置的 EtherNet/IP 端口进行了 BOOTP 设置，控制器电源上电时，BOOTP 客户

端运行，从 BOOTP 服务器获得 IP 地址。可以同时设置多个内置 EtherNet/IP 端口的 IP 地址。

4. 与上位计算机传递文件的内置 FTP 服务器

FTP 服务器内置控制器中。作为 FTP 客户端的工作站和计算机可以使用它，读写控制器中的数据文件。FTP 服务器允许从客户端传送大量的数据，无需任何额外的梯形图编程。

5. 自动的控制器时钟调整

连接到 Ethernet 网的控制器，其内置的时钟可自动调整为 NTP 服务器中时钟的时间。系统中所有的时钟自动调整到相同的时间。为自动调整控制器时钟，需要一个独立的 NTP 服务器。

6. Socket 服务

Socket 服务可以用来在普通的应用程序和控制器之间发送/接收数据。可以使用这些通信服务向/从远程节点发送/接收数据，例如，上位机和控制器、控制器和控制器之间。有两种 Socket 服务：UDP Socket 服务和 TCP Socket 服务。

7. HOST 名称

对于 NTP 服务器、SNMP 管理器或 Socket 指令和 CIP 通信指令的目标 (DNS 客户端或 HOST 设置)，可以直接指定 IP 地址，但也可以使用 HOST 名称代替 IP 地址。这是很有用的，例如，使用 HOST 名称后，当因为系统修改引起服务器的 IP 地址改变时，IP 地址会被自动发现。DNS 客户端使用 HOST 名称时需要一个独立的 DNS 服务器。DNS 服务器直接由它的 IP 地址指定。

8. SNMP 管理器

SNMP Agent 的内部状态信息通过内置的 EtherNet/IP 端口传给 SNMP 管理器。网络管理时需要一个独立的 SNMP 管理器。

9. 完整的故障诊断功能

提供多种功能，以快速辨别和处理错误。例如，启动时的自我诊断、记录错误发生时间和错误细节的事件日志。

8.2 标签数据链接

8.2.1 标签数据链接规格

标签数据链接能够使 EtherNet/IP 网络的控制器之间或者控制器与其他设备之间进行标签数据的交换。将变量分配至标签，也可将 I/O 内存地址分配至标签，如图 8-8 所示。标签数据链接的设置是通过 Network Configurator 软件实现的。

一个标签数据链接中，某个节点请求与其他节点建立连接，交换数据。请求连接的这个节点称为源，接收请求的节点称为目标。

控制器之间通信时，在接收 (消费) 数据 (即源) 控制器的内置 EtherNet/IP 端口设置链接信息。

用作数据交换的每个节点的输出字和输入字都在链接信息中设置。这些字是输出标签组和输入标签组。一个标签组必须指定至少一个标签。进行数据交换的数据大小是包含在标签组中总的标签大小。输出标签组的大小与输入标签组的大小必须一致。

一个标签就是用标签数据链接交换数据的一个单元。数据交换是在标签指定的本地网络变量和远程网络变量之间，或在指定的 I/O 内存区之间进行。

当一个链接已经建立，一个标签组可以包含 1~8 个标签（包括控制器状态）。每一个标签组代表一个标签数据链接的链接数据。标签数据链接通过一个标签组与另一个标签组建立连接。必须为每一个标签组设置一个标签组名。标签组的数目最多为 32。

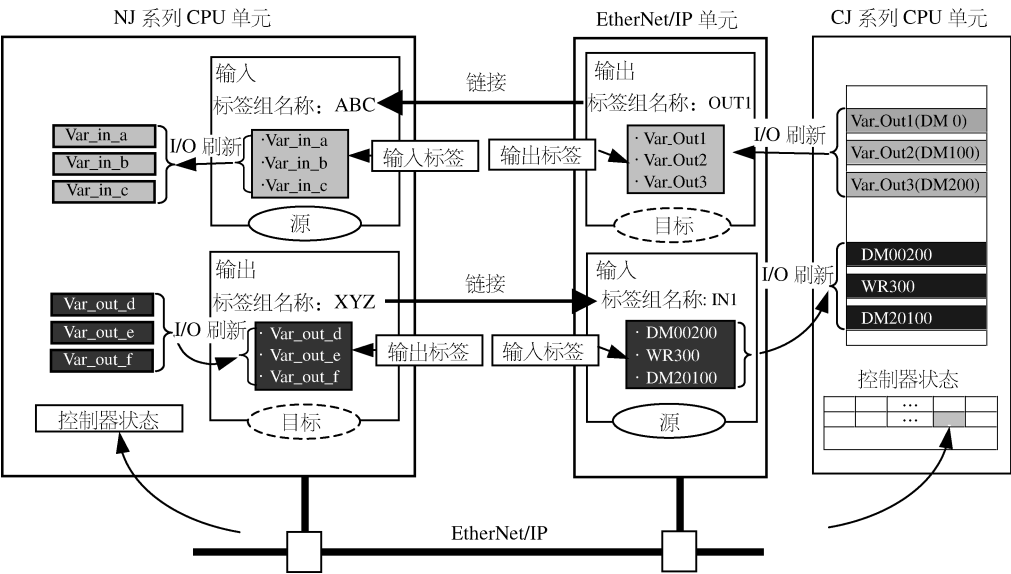


图 8-8 标签数据链接

标签数据链接规格见表 8-3。

表 8-3 标签数据链接规格

项 目		规 格
通信类型		标准 EtherNet/IP 隐式通信(连接类型循环通信)
设置方法		用 Network Configurator 软件设置了标签、标签组和连接之后,必须将标签数据链接的参数下载至 EtherNet/IP 网络中所有的设备 可以将 Sysmac Studio 中生成的网络变量导出为 CSV 文件。然后,将文件导入至 Network Configurator 并为标签分配网络变量 参数下载之后,要启动标签数据链接需重新启动 EtherNet/IP 单元
标签	支持的变量类型	指定以下网络变量作为标签 全局变量
	每个标签的最多字数	300 字(600 字节)
	最多标签数	256
标签组	每个标签组最多标签数	8(包括控制器状态为 7)
	每个标签组最多字数	300 字(600 字节)
	标签组最多数目	32
连接		每个单元的最大连接数;32
连接类型		每个连接可以设置成 1:1(单播)或者 1:N(多播)通信
数据包间隔(RPI)		10~10000ms,以 1ms 为单位 每一个连接的数据包间隔可以单独设置

注意:

1) 可为标签指定一个特定的 I/O 内存地址。在 Sysmac Studio 中使用 I/O 内存地址的 AT 规格, 创建一个变量, 然后为标签指定 AT 规格的变量。

2) 可以向 Network Configurator 导入在 Sysmac Studio 中生成的网络变量, 作为标签。具有网络公开属性的变量, 如果变量名程与 I/O 内存地址标记相同, 例如“0000”和“H0000”, 不能导出至 CSV 文件。

3) 可以作为标签的变量见表 8-4。

表 8-4 可以作为标签的变量

类 型		举 例	规 格
基本数据类型		aaa	支持
数组变量	数组	bbb	支持
	元素	bbb[2]	支持
结构体变量	结构体	ddd	支持
	成员	ddd. xxx	支持

8.2.2 标签数据链接的设置

用 Sysmac Studio 和 Network Configurator 设置标签数据链接参数。

1. 用 Sysmac Studio 设置网络变量

首先, 在 Sysmac Studio 将用作标签数据链接的变量创建为网络变量。

1) 在全局变量表中, 将用作标签数据链接 (即标签) 的输入或输出变量设置为网络公开属性。

2) 为了保持同一标签组内标签数据的并发性, 为同一个标签组内所有的标签 (也就是具有网络公开属性的变量) 设置一个刷新任务。使用 AT 规格的变量用作标签时, 不需要设置刷新任务, 因为其在主周期任务刷新。

2. 用 Network Configurator 设置和下载标签数据链接参数

用 Network Configurator 生成标签数据链接参数 (例如, 连接信息), 然后将参数下载到所有 EtherNet/IP 网络的源设备。当内置的 EtherNet/IP 端口使用标签数据链接时, 用 Network Configurator 进行以下设置。

(1) 生成配置信息

为生成定义数据链接的连接, 登记 EtherNet/IP 端口和 EtherNet/IP 单元。

(2) 设置标签

为输入 (消费) 标签和输出 (生产) 标签创建 CPU 单元的变量。对 NJ 内置的 EtherNet/IP 端口可以最多创建 256 个标签。每个标签的数据长度最大为 600 字节 (300 字), 可以导入和导出网络变量, 它们由 Sysmac Studio 生成 CSV 文件。可以用 Network Configurator 把它们登记为标签。输出标签可以被定义为: 当 CPU 单元发生致命错误时, 清除输出数据为 0, 或保持输出数据为发生错误之前的数据。

(3) 设置标签组

可以生成输出标签组和输入标签组, 并把标签分配给它们。可以生成总计 8 个 I/O 标签组。对 NJ 内置的 EtherNet/IP 端口可以生成多达 32 个标签组。每个标签的数据长度最多为 600 字节 (300 字), 在标签组中可以指定控制器状态表示 CPU 单元的运行状态 (运行信息和

错误信息)。

(4) 设置连接

目标设备输出标签组和源设备输入标签组通过连接关联在一起。对于一个 EtherNet/IP 端口可以打开总计 32 个连接。

1) 连接数：连接数是接收数据的输入标签组数目和发送数据的输出标签组数目的总和。无论连接是多点广播连接还是单点广播连接，每个连接设置用于一个连接。

2) 设置数据包间隔 (RPI)：RPI 是在以太网线路上完成标签数据链接时的数据 I/O 刷新周期。RPI 可以为每个连接分别设定。内置 EtherNet/IP 端口可以设置在 10 ~ 10000ms (1ms 为单位) 之间，默认设置为 50ms。在 EtherNet/IP 的通信线路上，不管节点的数目是多少，数据在每个连接设置的数据包时间间隔内交换。

3) 设置多点广播和单点广播通信：在标签数据链接的设置中，可以选择多点广播连接或单点广播连接作为连接类型。

多点广播连接，也称为多播连接，可以在一个数据包中发送一个输出标签组到多个节点，并分配到输入标签组，如图 8-9 所示。如果一个输出标签组发送到多个节点，多播连接可以减少通信负载。因为有的节点是不需要接收数据包的，为了减少通信负载，使用一个具有多播滤波功能的以太网交换机。使用多播连接并发送一个数据包里的一个输出标签组到多个节点，接收节点的下列设置必须和发送节点的设置相同：连接类型 (多播)、连接 I/O 类型、数据包间隔 (RPI) 和超时值。

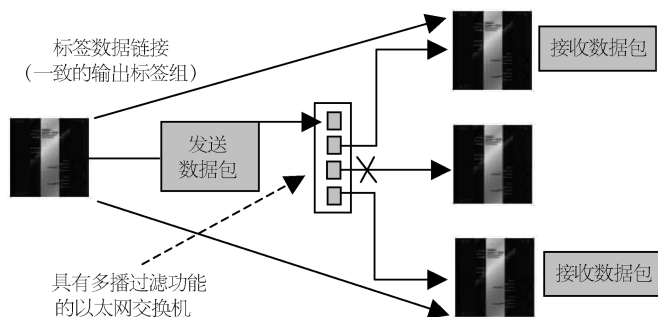


图 8-9 多播连接

点对点广播连接方式，也称为单播连接，如图 8-10 所示，为每个节点单独发送一个输出标签，发送的数据包的数量与输入标签的数量相同。这种通信方式增加了通信负担，因为同样的一个数据包如果要被不同的节点收到现在得重复发送多次。

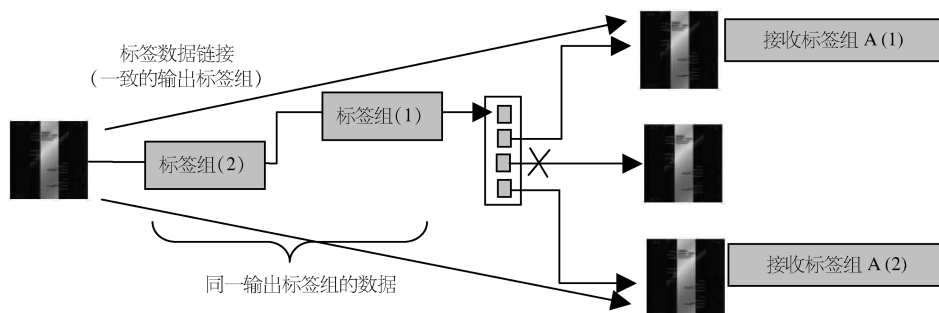


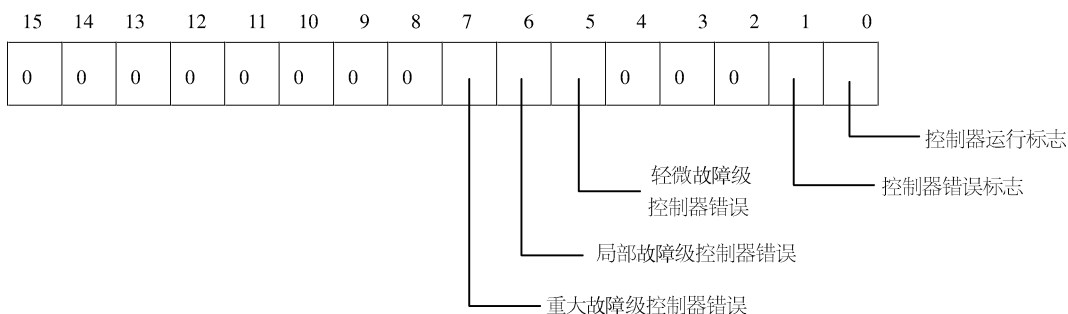
图 8-10 单播连接

8.2.3 启动和停止数据链接

当数据链接参数从 Network Configurator 中下载, NJ 系列控制器的电源上电后, 标签数据链接会自动启动。此后, 可以从网络配置中启动和停止整个网络或单独几个设备的标签数据链接。启动和停止单独几个设备的标签数据链接必须从源完成。此外, 可以使用系统定义变量来启动和停止整个网络。

8.2.4 控制器状态

在发送和接收的数据中, 可以将控制器状态作为标签组中的一个成员。控制器状态是表示 CPU 单元运行状态的一组标志 (运行信息、错误信息、控制器错误级别)。如果控制器状态被指定为一个输出 (发送) 标签, 则控制器状态以下面格式加到标签组的开始。



当一个标签数据链接启动时, 控制器状态的内容被存储在下面给出的系统变量中。

- _EIP_TargetPLCModeSta (目标 PLC 运行模式);
- _EIP_TargetPLCErr (目标 PLC 错误信息)。

8.2.5 标签数据链接数据的并发

为了保持标签数据链接数据的并发, 必须为分配给标签的每一个网络变量设置一个刷新任务。

1. 保持标签组标签数据的并发

当需要几个标签的数据保持并发, 必须满足以下条件。

- 1) 标签必须分配在同一标签组 (连接)。
- 2) 分配给标签的网络变量必须设置刷新任务, 所有标签的刷新任务必须为同一个。如果为网络变量设置了刷新任务, 必须设置足够的时间保证从控制器外部访问网络变量。
- 3) 使用 AT 规格的标签不能包含在同一个标签组。
- 4) 每个任务的变量访问时间设置时, 必须设置一个比传输标签数据需要的时间更大的值。

为标签 (网络变量) 设置刷新任务, 标签组里的标签保持并发, 如图 8-11 所示。不需要为使用 AT 规格的变量 (标签) 设置刷新任务, 因为它们总是在主周期任务里更新的。

如果不为标签 (网络变量) 设置刷新任务, 该标签组里的标签不能保持并发, 如图 8-12 所示。

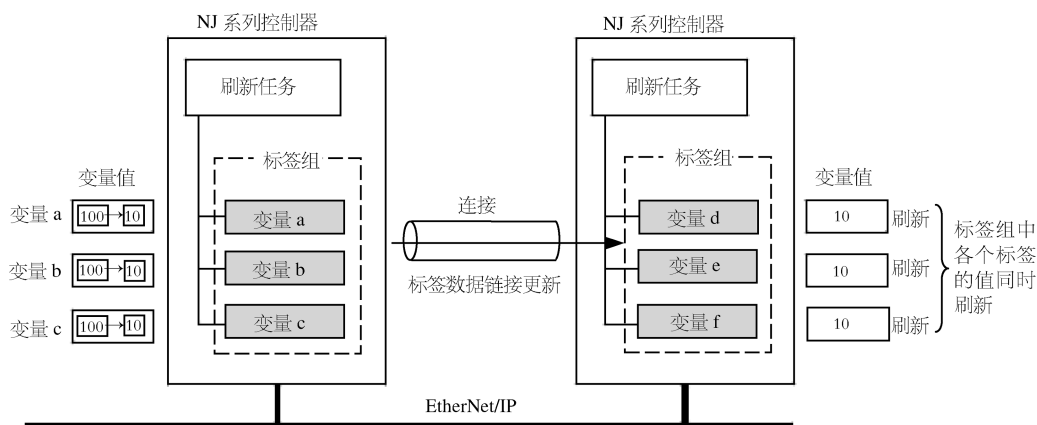


图 8-11 保持标签组标签的并发

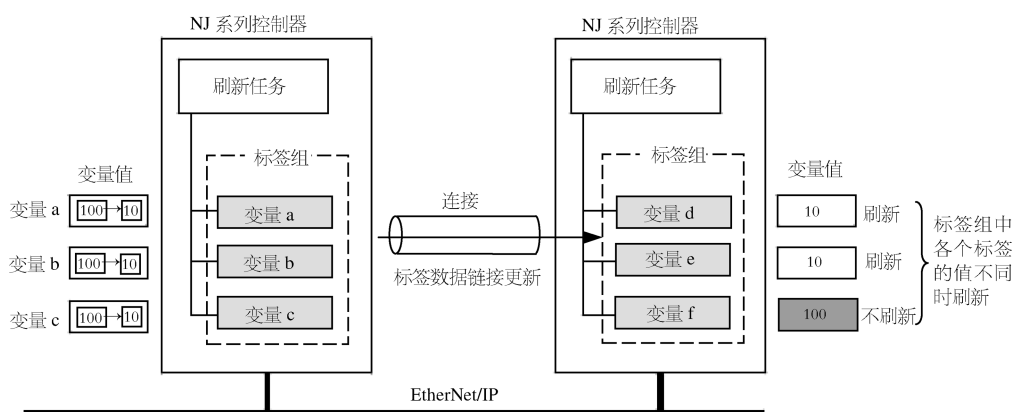


图 8-12 标签组标签不保持并发

2. 更新网络变量（分配给标签）的时序和访问该网络变量程序的执行周期同步

分配了刷新任务的标签和没有分配刷新任务的标签的操作不同。为标签（网络变量）设置刷新任务，如果该刷新任务包含访问它们的程序，这些标签在程序执行的同时被刷新。没有设置刷新任务的标签（网络变量）的刷新是由最低执行优先级的系统服务处理的，因此它和程序的执行不同步。

图 8-13 为系统服务时序。主周期任务的刷新是在任务进程中的系统公共进程 2 期间完成。如果一个程序需要访问一个具有 AT 规格的网络变量，把此程序放到主周期任务中以便于匹配使用 AT 规格网络变量的刷新时序。

刷新任务和标签数据链接中数据并发之间的关系如图 8-14 所示，如果在标签数据链接中不为全局变量指定同一个刷新任务，可能会发生以下情况。

- 1) 当对输出标签组发送数据时，另一个任务可能已经在那个数据发送前写入了不同的值，这取决于任务的时序。
- 2) 当输入标签组接收数据时，另一个任务可能在那个数据接收后写入不同的值，这取决于任务的时序。

因此，为了保持标签数据链接中数据的并发，无论是输出 CPU 单元上还是输入 CPU 单元上，对输出标签和输入标签必须指定同一刷新任务。

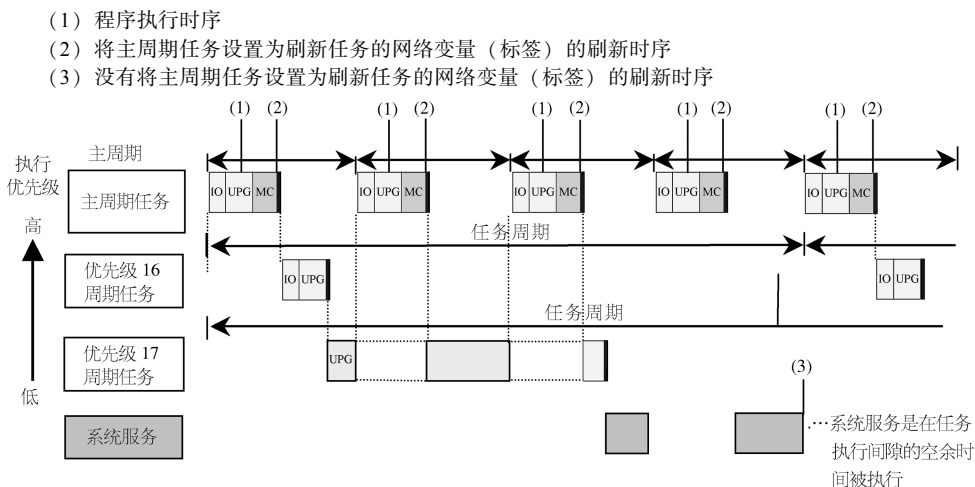


图 8-13 系统服务时序

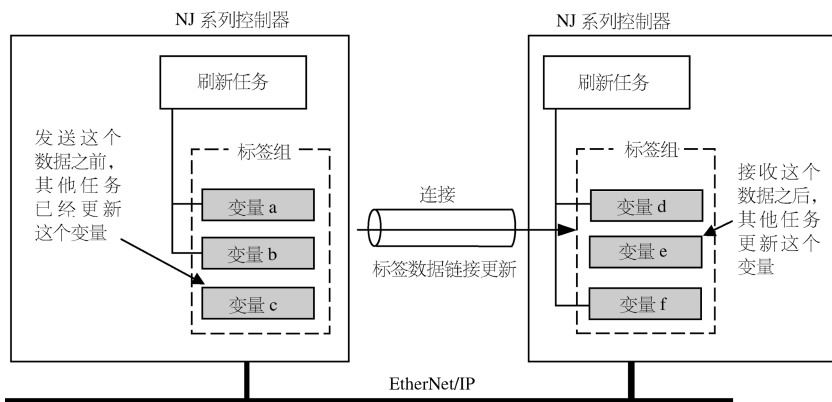


图 8-14 刷新任务和标签数据链接中数据并发之间的关系

3. 保持并发性需要的处理时间

为保持数据链接数据的并发性，为标签（网络变量）设置一个刷新任务，该指定任务需要的处理时间增加。由于任务处理时间的增加，在为每个连接设置的数据包间隔（RPI）期间，标签数据链接数据的刷新可能不会发生。因此，需要调整数据包间隔（RPI）设置以匹配指定的刷新任务的周期。

4. 任务设置步骤

- 1) 设置全局变量并为其指定刷新任务，在 Sysmac Studio 的“任务设置”→“任务变量的排他控制设置”里设置刷新任务和访问任务。
- 2) 为每个刷新任务设置变量的访问时间。

8.3 CIP 信息通信

8.3.1 概述

CIP 命令能在需要的任何时刻发送到 EIP 网络中的其他设备。如图 8-15 所示，CIP 通信

基于客户-服务器模式，从客户端应用 CIP 命令读写服务器的内存，而作为服务器的控制器不用添加任何特殊的程序。

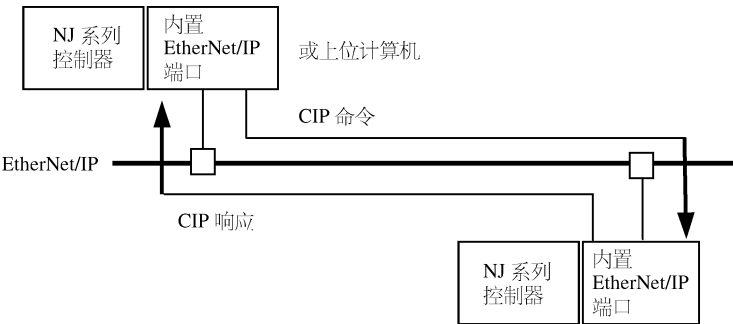


图 8-15 CIP 信息通信

CIP 通信规格见表 8-5。

表 8-5 CIP 通信规格

项 目	规 格
信息类型	可以选择以下任何一个： CIP UCMM 无连接通信； CIP 3 类连接通信
执行方法	CIPSend(发送显式信息 3 类)指令或 CIPUCMM(读变量 UCMM 显式)指令
数据内容	发送 CIP 命令及接收响应
通信参数	信息类型,超时值以及根路径指标

8.3.2 CIP 通信指令

如图 8-16 所示，CIP 通信是在 NJ CPU 单元程序中执行 CIP_SEND 命令发送 CIP 通信指令，例如，读写 EIP 网络中 NJ CPU 单元或 CJ2 CPU 单元的变量。

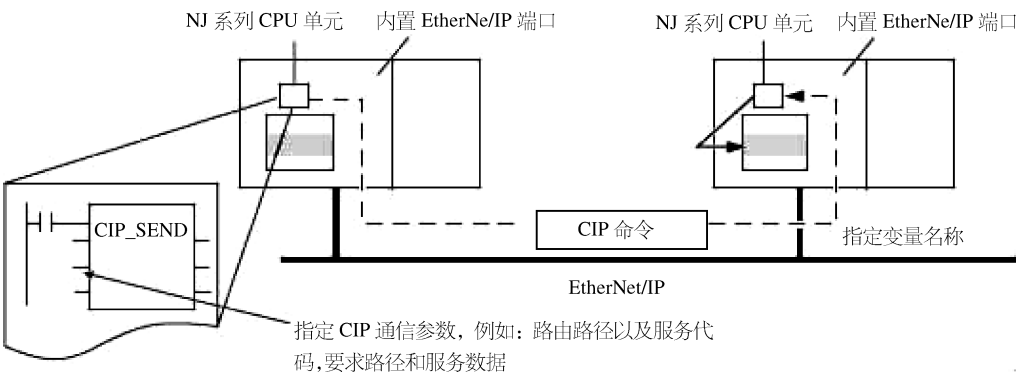


图 8-16 CIP 通信指令的执行

CIP 通信指令见表 8-6。

表 8-6 CIP 通信指令

指 令	名 称	说 明	通 信 方 法
CIPUCMMRead	读变量(UCMM 显式)	从 CIP 网络指定的远程控制器中读取网络公开属性变量的值,并存储到本地控制器的变量中	CIP UCMM 无连接信息
CIPUCMMWrite	写变量(UCMM 显式)	将本地控制器变量的值,写到 CIP 网络指定的远程控制器中网络公开属性的变量	
CIPUCMMSend	发送显式信息(UCMM)	向 CIP 网络指定的远程控制器发送指定的 CIP 指令	
CIPOpen	打开 CIP 3 类连接	该指令打开与指定远程节点的一个 CIP 3 类连接	CIP 3 类连接信息
CIPRead	读变量(3 类显式)	从 CIP 网络指定的远程控制器中读取网络公开属性变量的值,并存储到本地控制器的变量中	
CIPWrite	写变量(3 类显式)	将本地控制器变量的值,写到 CIP 网络指定远程控制器中网络公开属性的变量	
CIPSend	发送显式信息(3 类)	向 CIP 网络指定的远程控制器发送一条指定的 CIP 3 类指令	
CIPClose	关闭 CIP 3 类连接	该指令关闭由句柄指定的 CIP 3 类连接	

8.3.3 CIP 通信指令的使用

CIP 信息通信包括的过程见表 8-7。如果使用 CIP 3 类连接,则数据发送之前要求打开过程,数据接收之后要求关闭过程。

表 8-7 CIP 信息通信包括的过程

过 程	说 明	指 令
打开过程(仅适用于 CIP 3 类连接)	使用 CIP 信息之前执行这个过程,打开过程一直持续到连接建立	CIPOpen
发送和接收变量数据	这个过程用来读和写网络公开属性的指定变量	CIPUCMMRead CIPUCMMWrite CIPRead CIPWrite
发送 CIP 命令	设置要求的 CIP 命令	CIPUCMMSend CIPSend
关闭过程(仅适用于 CIP 3 类连接)	这个过程关闭连接	CIPClose

可以同时执行最多 32 条 CIP 通信指令。在用户程序中使用独占控制,保证同时不会有超过 32 条指令执行。

CJ 系列单元的内存地址(例如 D00000)不能直接指定。为了访问 CJ 系列单元的内存,访问 AT 规格的变量,如图 8-17 所示。

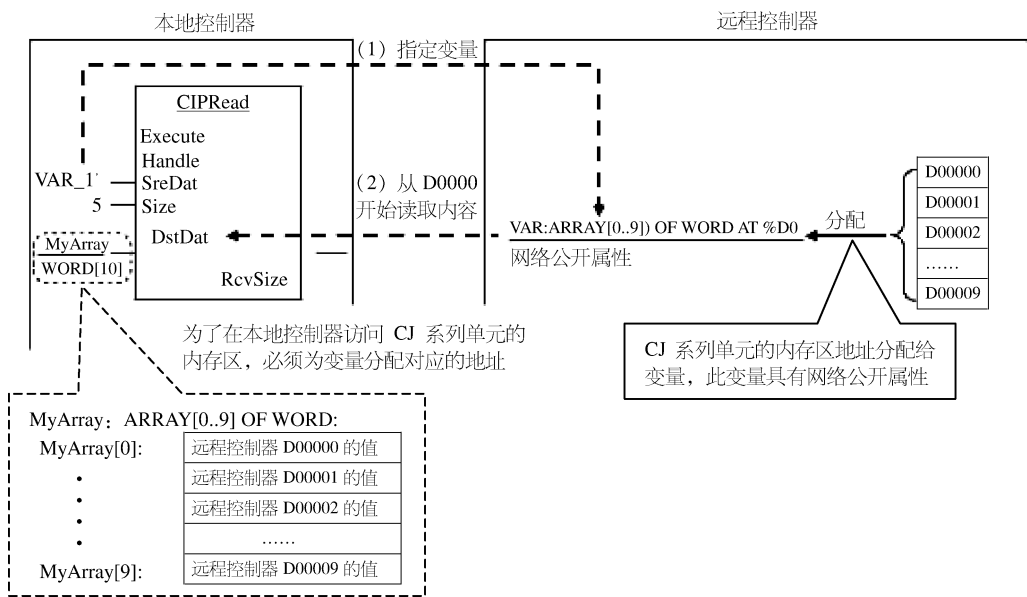


图 8-17 访问 CJ 单元的内存

8.3.4 路由路径

路由路径是在网络中从本地 CPU 单元到远程控制器的路径。基于路由路径形成 CIP 通信时的路径。

路由路径由 EPATH 数据类型给出，基本的格式为

Network_type_number\Remote_address

网络类型号和远程地址则根据路由类型而定，路由类型是 CPU 机架上的单元或通信单元上的通信端口，表 8-8 列出网络类型号和远程地址。

表 8-8 网络类型号和远程地址

路 由	网络类型号(十六进制)	远程地址(十六进制)
输出至 CPU 机架上的单元	#01(底板端口)	远程单元地址
从通信单元的通信端口输出	#02(内置 EIP 端口)	IP 地址

图 8-18 为输出的路由。

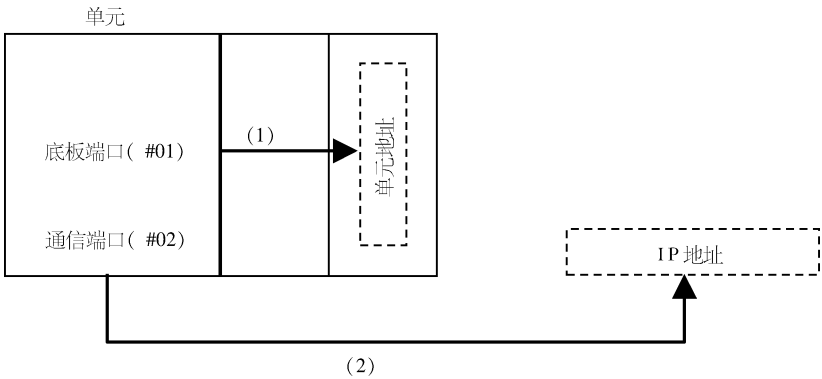


图 8-18 输出的路由

根据输出目的地确定路由，有两种情况。

1. 输出到 CPU 机架上的一个单元

输出命令到 CPU 机架上底板的网络端口时，指定单元地址作为远程单元地址。单元地址用来区别连接到网络中同一个节点的多种器件，单元地址如下设置。

CPU 单元:	00H
CPU 总线单元 (EIP 单元):	单元号 + 10H
特殊 I/O 单元:	单元号 + 20H
内板:	E1H
计算机:	01H
连接至网络的单元:	FEH

2. 输出到一个通信单元的通信端口上

输出命令到 EIP 端口。指定 IP 地址作为远程节点的地址。

内置 EIP 端口的通信与 EIP 单元通信的路由路径的标记不同。下面提供路由路径的例子。

(1) 内置 EIP 端口之间的通信

CPU 单元 1 与 CPU 单元 2 的内置 EIP 端口之间的通信如图 8-19 所示。

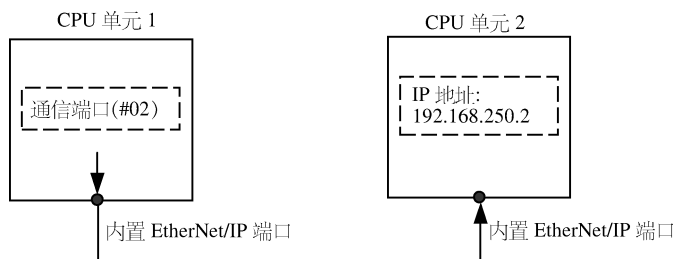


图 8-19 内置 EIP 端口之间的通信

路由路径构造为

- 网络类型号: #02 (通过 EtherNet/IP 端口输出命令);
- 远程地址: 指定远程 IP 地址;
- 路由路径: \02\192.168.250.2。

(2) 从内置 EIP 端口至 EIP 单元的通信

从 CPU 单元 1 内置 EIP 端口至 CPU 单元 2 通信端口如图 8-20 所示，CPU 单元 2 上安装 EIP 单元。

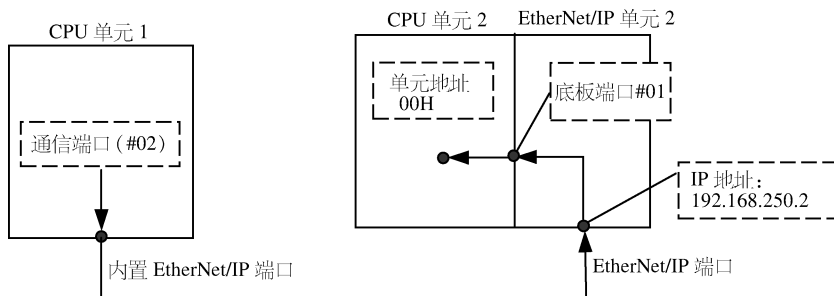


图 8-20 从内置 EIP 端口至 EIP 单元的通信

路由路径构造为

1) CPU 单元 1 至 EIP 单元 2:

- 网络类型号: #02 (通过 EIP 端口输出命令);
- 远程地址: 指定远程 IP 地址。

2) EIP 单元 2 至 CPU 单元 2:

- 网络类型号: #01 (通过内部底板端口输出命令);
- 远程地址: #00 (CPU 单元的单元地址)。

路由路径: \02\192.168.250.2\01\#00

①

②

①远程 EIP 单元的 IP 地址;

②远程 CPU 单元的单元地址。

(3) 通过中继节点访问

图 8-21 为 CPU 单元 1 通过中继节点 CPU 单元 2, 访问 CPU 单元 3。

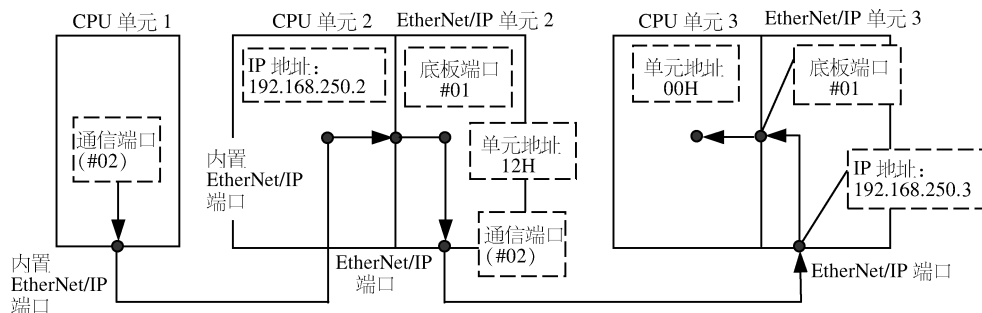


图 8-21 通过中继节点访问

路由路径构造为

1) CPU 单元 1 至 CPU 单元 2:

- 网络类型号: #02 (输出命令通过 EIP 端口);
- 远程地址: 指定远程 IP 地址。

2) EIP 单元 2 至 CPU 单元 2:

- 网络类型号: #01 (输出命令通过内部底板端口);
- 远程地址: #12 (EIP 单元的单元地址: $2 + 10H = 12H$, 单元号为 2)。

3) EIP 单元 2 至 EIP 单元 3:

- 网络类型号: #02 (输出命令通过 EIP 端口);
- 远程地址: 指定远程 IP 地址。

4) EIP 单元 3 至 CPU 单元 3:

- 网络类型号: #01 (输出命令通过内部底板端口);
- 远程地址: #00 (CPU 单元的单元地址)。

路由路径: \02\192.168.250.2\01\#12\02\192.168.250.3\01\#00

①

②

③

④

①远程内置 EIP 端口的 IP 地址;

②远程 EIP 单元 2 的单元地址 (2 + 10H, 单元号为 2);

③远程 EIP 单元 3 的 IP 地址;

④远程 CPU 单元 3 的单元地址。

8.4 其他通信服务

8.4.1 Socket 服务

NJ 内置的 EIP 端口支持 Socket 服务。Socket 是一种通信接口, 它允许用户程序直接使用 TCP 和 UDP 协议。在上位计算机中, Socket 被写成 C 语言的接口库函数, 在用户程序中可以调用。在一台 UNIX 计算机上, Socket 接口以系统调用的方式提供。对于内置的 EIP 端口, 通过在用户程序里执行指令使用 Socket。可以使用这些通信服务, 与远程节点交换数据, 例如, 计算机与控制器之间, 控制器与控制器之间。内置的 EIP 端口提供 UDP Socket 服务和 TCP Socket 服务。

图 8-22 为上位机和控制器之间的 Socket 操作, Socket 服务用于控制器和不支持 CIP 信息通信的普通应用间的数据交换。上位机和 PLC 通过 Socket 使用 TCP 和 UDP 进行数据交换。

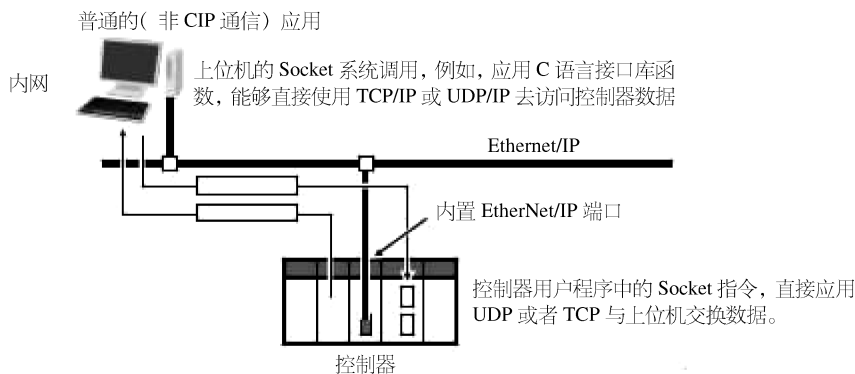


图 8-22 Socket 操作

内置 EIP 端口有 16 个 UDP Socket、16 个 TCP Socket, 可以使用总计 32 个 UDP 和 TCP Socket。

在一个 UNIX 工作站上, 端口号 0 ~ 1023 只能由超级用户使用。另外, 端口 0 ~ 255 是为已知端口保留的。因此, 建议 Socket 服务使用 1024 ~ 65535 或者更高的端口号。不能为内置 EIP 端口指定端口号 0。另外, 表 8-9 中的端口号仅由内置 EIP 端口使用, 不能用作其他目的。

表 8-9 内置 EIP 端口使用的端口号

应 用	UDP	TCP	应 用	UDP	TCP
FTP 服务	—	20,21*	SNMP 陷阱	162*	—
BOOTP 客户	68	—	FINS/UDP	9600	—
HTTP 服务	—	80	FINS/TCP	—	9600*
NTP 服务	123*	—	由系统使用	2222,2223 和 2224	9610
SNMP 代理	161*	—	CIP 信息	—	44818

* 这是默认的端口号，可以在设置中改变。

Socket 支持客户-服务器模式。TCP Socket 启动数据传输之前，要在两个节点之间建立连接，称为“虚电路”。建立连接时，源节点的客户端执行一条建立 TCP 连接的命令，向目标节点的服务器发出请求；服务器执行一条接受 TCP 连接的指令，向客户端发出同意的响应。连接建立起来后，客户端向服务器发送数据。TCP 提供高可靠性服务，通信时要求目标节点应答，加以确认。数据分包传送，每个数据包最大长度为 1024 字节。

UDP 提供高效率服务，通信时不要求目标节点应答，通信可靠性要由用户程序解决。数据分包传送，每个数据包最大长度为 1472 字节。

可以将 UDP 服务的目标 IP 地址设为广播地址，向在同一 EtherNet/IP 网络上的上位机和其他节点发送一个广播数据。如果网络上有一个路由器，数据包不能发送到路由器之后。可以广播发送最多 1472 字节的数据，超过 1472 字节的数据不能广播。

Socket 服务指令见表 8-10，只在 ST 编程时使用。可以同时执行最多 32 个 Socket 指令。在用户程序里通过独占控制使得同时执行的 Socket 指令最多不超过 32 个。

表 8-10 Socket 服务指令

UDP/TCP	指 令	Socket 服务	UDP/TCP	指 令	Socket 服务
UDP Socket	SktUDPCreate	创建 UDP Socket 指令	TCP Socket	SktTCPRcv	TCP Sockets 接收指令
	SktUDPRcv	UDP Socke 接收指令		SktTCPSEND	TCP Sockets 发送指令
	SktUDPSend	UDP Socke 发送指令		SktGetTCPStatus	读 TCP Sockets 状态指令
TCP Socket	SktTCPAccept	接受 TCP Sockets 指令	UDP 和 TCP	SktClose	关闭 UDP/TCP Socket 指令
	SktTCPConnect	连接 TCP Sockets 指令	Socket 服务	SktClearBuf	清除 UDP/TCP Socket 缓冲指令

使用 Socket 服务时，需要在 Sysmac Studio 多视图浏览器“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“TCP/IP”的对话框中设置：IP 地址、子网掩码、IP 路由表、Keep alive 和 Linger Option。

8.4.2 FTP 服务

内置 EIP 端口包含 FTP (文件传输协议) 服务器功能。FTP 基于客户-服务器模式，如图 8-23 所示，以太网中的计算机作为客户端，内置 EIP 端口作为服务器，客户端向服务器发出 FTP 命令，可以上载或下载控制器 SD 存储卡中的大文件。SD 存储卡插在 CPU 单元上。内置 EIP 端口不支持 FTP 客户端功能，仅可作为 FTP 服务器。

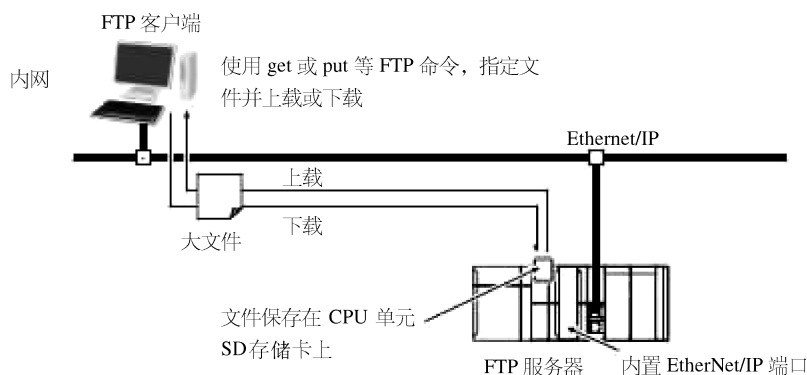
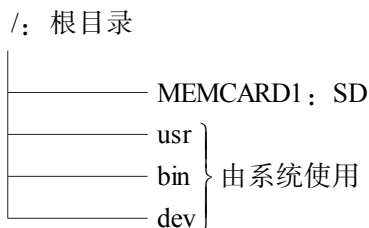


图 8-23 FTP 客户-服务器模型

在 FTP 服务器模式下，插在 CPU 单元上的 SD 存储卡的树状目录如下：



文件名加上扩展名最多 62 个字符，扩展名 3 个字符。

FTP 有两种文件传送模式：ASC II 模式和二进制模式，使用 type 命令选择传送模式。

上位机使用 UNIX 操作系统时，FTP 由一组 Shell 命令实现。上位机调用 open 命令与 FTP 服务器建立连接，一旦成功，双方进入交互式会话状态。然后，上位机调用其他 FTP 命令，与 FTP 服务器传输数据（如文件复制等），用户键入 close 和 quit（或 bye）命令，退出 FTP 会话。FTP 命令集见表 8-11。

表 8-11 FTP 命令集

命令	功 能	命令	功 能
open	连接指定的 FTP 服务器	get	将指定文件从 SD 存储卡中传到本地客户端
user	为远程 FTP 服务器指定用户名	mget	将多个指定文件从 SD 存储卡中传到本地客户端
ls	列出远程服务器上的文件名	put	将指定的本地文件传到远程服务器
dir	列出远程服务器上的文件名	mput	将指定多个本地文件传到远程服务器
mdir	列出远程服务器上的文件名及细节	delete	从远程服务器上删除 SD 存储卡中指定文件
rename	改变文件名	mdelete	从远程服务器上删除 SD 存储卡中多个文件
mkdir	在远程服务器上的工作目录创建一个子目录	append	使用由 type 命令指定的文件数据类型,将本地文件添加到远程服务器
rmdir	从远程服务器上的工作目录删除一个子目录	close	断开 FTP 服务器
cd	将远程服务器的工作目录改变到指定目录	bye	关闭 FTP 客户端
pwd	列出远程服务器的工作目录	quit	关闭 FTP 客户端
type	指定要传输文件的数据类型		

使用 FTP 服务功能时，需要在 Sysmac Studio 多视图浏览器“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“FTP 设置”的对话框中设置：FTP 服务器、端口号、FTP 登录名及口令。FTP 服务器设为“使用”。端口号默认为 21，使用其他号需设置。登录名、口令分别是 1~12 个、8~32 个字符组成的字符串，这些字符是字母、数字和“-”或“_”等，都是单字节的 ASCⅡ 字符，区分字母的大小写。

在使用 FTP 服务器功能之前，上位机必须与 FTP 服务器建立连接，从客户端输入 FTP 登录名及口令。

8.4.3 自动时钟调整（NTP 服务）

自动时钟调整由 NTP (Network Time Protocol) 服务完成。如图 8-24 所示，在 CPU 单元上电后的指定时间或时间段内，通过内置的 EtherNet/IP 端口，时钟信息从 NTP 服务器读取，CPU 单元内部时钟的时间更新为读取的时间。NTP 服务器用来控制局域网上的时间。

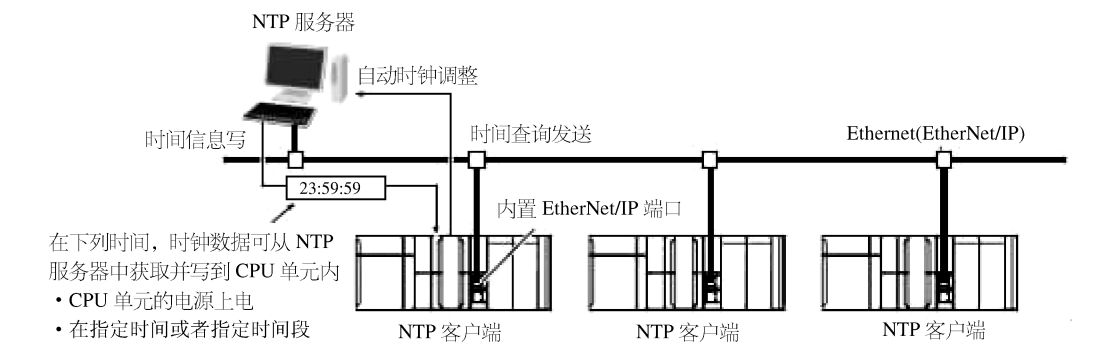


图 8-24 NTP 服务

NTP 服务规格见表 8-12。

表 8-12 NTP 服务规格

项 目	规 格	
协议	NTP	
端口号	123(UDP) 也可以通过 Sysmac Studio 在“内置 EtherNet/IP 端口设置”改变端口号	
访问 NTP 服务器	将时钟信息从 NTP 服务器写至本地 CPU 单元	从设置在网络上的 NTP 服务器获取时钟信息， 将获取的此信息应用到本地 CPU 单元
NTP 操作时序	如果应用 NTP 功能,时钟信息将在以下时间自动更新： • 当 CPU 单元的电源上电时连接建立； • 指定的时间或者指定的时间段(根据 NTP 操作时序的选项)	

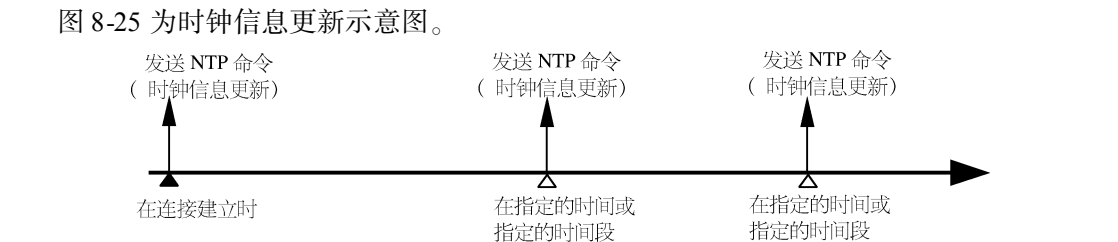


图 8-25 时钟信息更新示意图

使用 NTP 服务时，需要在 Sysmac Studio 多视图浏览器“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“NTP 设置”的对话框中设置：NTP 服务器时钟信息、端口号、服务器指定方法、NTP 操作时序。

8.4.4 SNMP 服务

SNMP (Simple Network Management Protocol) 是一种网络管理协议。可以使用 SNMP 管理由支持 SNMP 的设备组成的任何网络。管理该网络的服务器称为 SNMP 管理器。被管理的网络设备称为 SNMP Agent。

图 8-26 为 SNMP 服务示意图。

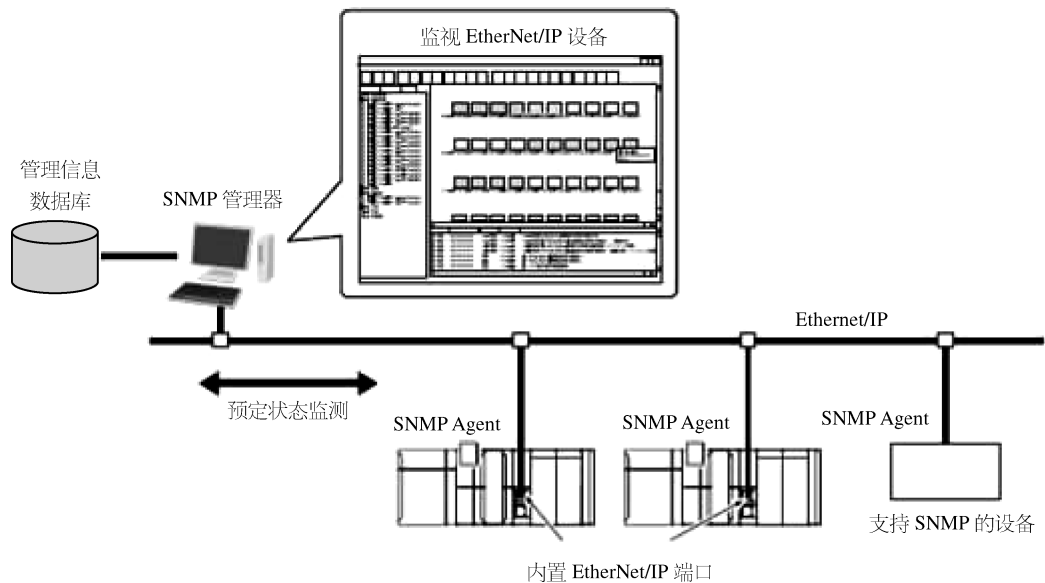


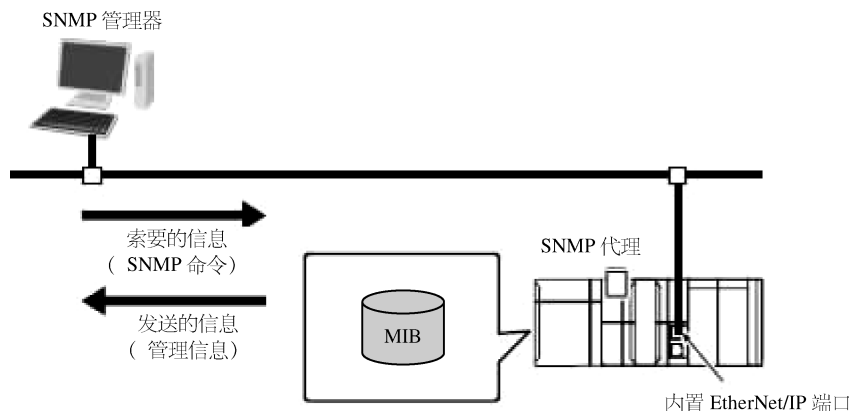
图 8-26 SNMP 服务示意图

SNMP 服务规格见表 8-13。

表 8-13 SNMP 服务规格

项 目	规 格
协议	SNMP
Agent	SNMPv1,SNMPv2c
MIB	MIB-II
端口号	SNMP Agent: 161(UDP) SNMP Trap:162 (UDP) 这些可以在 Sysmac Studio 的“内置 EtherNet/IP 端口设置”中改变
SNMP Trap 操作时序	状态报告在以下时间发送给管理器： ● 控制器上电； ● 连接建立； ● SNMP Agent 授权失败
支持的 MIB 命令	GetRequest/GetNextRequest

SNMP Agent 如图 8-27 所示。内置 EIP 端口都有自己的管理信息称为 MIB (管理信息基础)。该信息可以提供给 SNMP 管理器。SNMP 管理器是收集和处理 SNMP 网络设备的信息并将该信息提供给网络管理员的软件。SNMP 管理器可用来监视内置 EIP 端口。

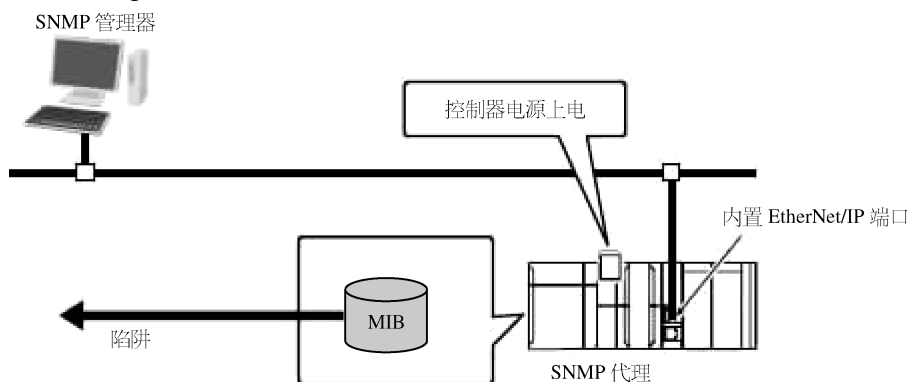


SNMP 管理器有一条索要 MIB 信息的 SNMP 命令。内置 EIP 端口 SNMP Agent 功能支持 SNMPv1 (RFC1157) 和 SNMPv2c (RFC1901)。SNMP 管理器使用 SNMPv1 或 SNMPv2c 协议管理内置的 EIP 端口，也可以同时使用 SNMPv1 或 SNMPv2c 协议。

使用 SNMP Agent 时，需要在 Sysmac Studio 多视图浏览器“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“SNMP 设置”的对话框中进行必要的设置。

SNMP Trap 如图 8-28 所示。SNMP Trap 就是 SNMP 陷阱报告，当控制器出现故障或其他一些特定问题时，形成一个称为陷阱的状态通知报告，将被发送至 SNMP 管理器。这样，即使管理器没有定期监视内置 EIP 端口，仍然可以掌握各个控制器的状态变化。下列情况下会发送陷阱。

- 1) 控制器电源上电时；
- 2) 连接建立时；
- 3) 对 SNMP Agent 授权失败时。



使用 SNMP Trap 时，需要在 Sysmac Studio 多视图浏览器“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“SNMP Trap 设置”的对话框中进行必要的设置。

8.4.5 BOOTP 客户端

设置内置 EIP 端口的 IP 地址有下列方法。

1. 固定设置

如果需要改变内置 EIP 端口默认的 IP 地址，设置一个用户指定的 IP 地址。

2. 从 BOOTP 服务器获得

每次电源上电时从 BOOTP 服务器获得 IP 地址。

3. 固定设置为从 BOOTP 服务器获得 IP 地址

一旦电源上电从 BOOTP 服务器获得 IP 地址后，将一直保持不变。

后面两种方法都是从 BOOTP 服务器自动获得 IP 地址，BOOTP 客户端如图 8-29 所示，内置 EIP 端口为客户端，通过 BOOTP 服务器，可以给所有的内置 EIP 端口设置 IP 地址。使用 BOOTP 设置软件，根据每个 CPU 单元内置 EIP 端口的 MAC 地址设置 IP 地址。

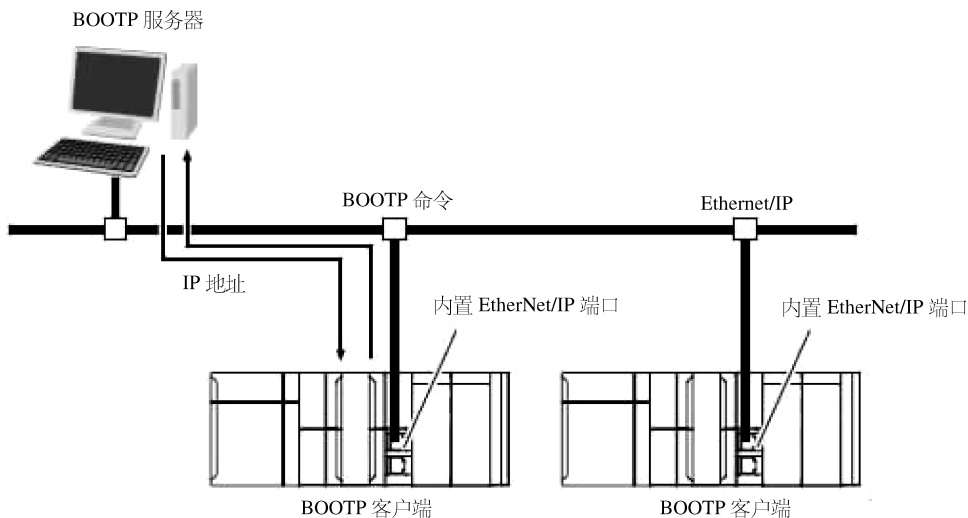


图 8-29 BOOTP 客户端

使用 BOOTP 客户端功能时，需要在 Sysmac Studio 多视图浏览器“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“TCP/IP 设置”的对话框中进行设置。

内置 EIP 端口的 IP 地址可以进行测试：选择“控制器”菜单的“通信设置”，在出现的视图输入 IP 地址，点击“Ethernet 通信测试”按钮。或者与控制器在线后，观察右下角的控制器状态显示面板，会出现实际的 IP 地址。

8.4.6 IP 路由表

两个以太网网段通过路由器连接。NJ 与其他网段的节点通信时，要设置路由表，表中包括：目标 IP 地址、目标 IP 地址的子网掩码和网关地址。在 Sysmac Studio 多视图浏览器的“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“TCP/IP 设置”对话框中进行设置。

当路由表设置发生错误时，_EIP_IPRTblErr 变量会变为 TRUE。

8.4.7 HOST 名指定

HOST 名指定功能是将一个 HOST 名赋予给一个指定的 IP 地址。之后在 NTP 服务、SNMP 管理器、Socket 服务、CIP 通信时都可以使用它。该功能会根据 PLC 内的 HOST 名 - 地址表去寻找, 如果无法找到则去询问 DNS 服务器, 如图 8-30 所示。

HOST 名指定操作是在多视图浏览器“配置与设置”→“控制器设置”→“内置 EtherNet/IP 端口设置”→“TCP/IP 设置”的对话框中完成的。

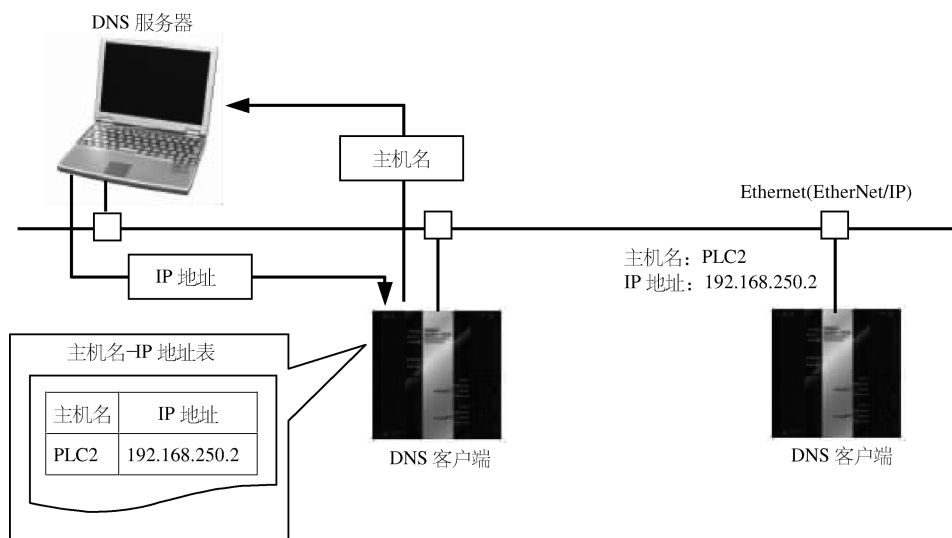


图 8-30 HOST 名指定

8.4.8 FINS 通信

NJ 系列内置的 EIP 端口支持 FINS 通信, 如图 8-31 所示。由于 FINS 指令不支持变量, 所以程序里涉及交换的数据部分必须要用 CJ 系列单元的内存。

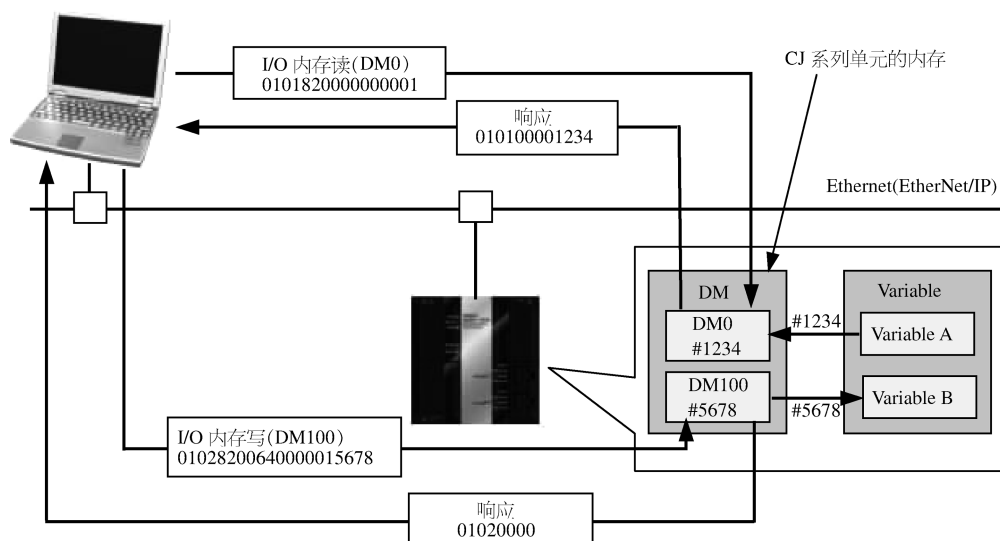


图 8-31 FINS 服务

可以使用大部分 FINS 指令，但由于 NJ 与 CJ 存在差异，所以有些指令使用时会有所限制，例如，不能对 NJ 指定 EM 区操作，所以和 EM 区有关的一些 FINS 指令就无效了。

FINS 指令见表 8-14。

表 8-14 FINS 指令

	MR	SR	命令	功 能	注 释
I/O 内存区访问	01	01	内存区读	读取连续的 I/O 内存区字的内容	仅 CJ 系列单元的存区支持
	01	02	内存区写	写连续的 I/O 内存区字的内容	
	01	03	内存区填	写同样的数据到 I/O 内存区字的特定范围	
	01	04	多内存区读	读特定的不连续 I/O 内存区字的内容	
	01	05	内存区传送	复制连续的 I/O 内存区字的内容到另一 I/O 内存区字	
参数访问	02	01	参数区读	读连续的参数区字的内容	不能访问 PLC 系统的设置区
	02	02	参数区写	写连续的参数区字的内容	
	02	03	参数区清	写同样的数据至特定范围的参数区字	
运行方式控制	04	01	运行	改变 CPU 单元的运行模式至运行, 监视, 调试	只适用于运行和编程模式
文件内存	22	01	文件名读	读文件内存区的数据	EM 文件不能被指定
	22	02	单文件读	从一个特定位置的单独文件读一个特定长度的文件数据	
	22	03	单文件写	从一个特定位置的单独文件写一个特定长度的文件数据	
	22	04	文件内存格式化	格式化(初始化)文件内存区	
	22	05	文件删除	在文件存储区中的特定文件	
	22	07	文件复制	从一个文件内存区中复制文件到另一个同样的系统	
	22	08	文件更名	更改文件名称	

使用 FINS 通信时，需要在 Sysmac Studio 的多视图浏览器“配置与设置”→“控制器设置”→“运行设置”→“FINS 设置”的对话框中进行设置。

8.4.9 NJ 控制器与 NS 触摸屏通信

欧姆龙 NS 系列新型号的触摸屏内置了 EIP 端口，可以与 CS/CJ 系列的 EtherNet/IP 单元、内置 EIP 端口的 CJ2H□□-CPU-EIP、CJ2M-CPU3□-EIP 或 NJ501-1□00 等 CPU 单元建立连接，进行 EIP 通信。具体实现时需要注意以下问题：

- 1) NS 触摸屏必须是 V2 或更高的版本。
- 2) 触摸屏编程软件 CX-Designer 必须是 V3.3 或更高的版本。
- 3) 用 CX-Designer 对触摸屏新建工程时，系统版本必须选为 8.5 或更高。

NS 与主机建立 EIP 通信后，通过地址（称为通信地址）访问主机内存。有两种类型的

通信地址：直接地址和标签。直接地址使用区域名称和数字地址指定主机内存中的一个特定单元，如 D00000。标签指定在 PLC 中设置的网络变量。

NJ 系列 CPU 中有变量，没有直接地址，NJ 有两类变量可以被 NS 触摸屏访问。

- 1) 属性设置成网络公开的全局变量。
- 2) 属性为网络公开的系统定义变量 (可通过软件操作手册确认)。

需要指出的是，NS 触摸屏内没有结构体这类变量，在触摸屏登记变量时，直接登记到结构体成员，例如，登记 NS_struct. bool，表示一个 BOOL 型变量。

8.4.10 NJ 控制器与 CJ PLC 的 EtherNet/IP 通信

NJ 系列 CPU 单元可以和 CJ1/CJ2 PLC 实现标签数据链接以及 CIP message 通信。但是要注意以下几点。

1) NJ 系列 CPU 可以和 CJ 系列所具有的各种数据类型的变量进行数据交换，这些数据类型包括 TIMER、COUNTER 以及 CHANNEL。

2) NJ 系列无法与 CJ 系列进行那些 CJ 不支持的数据类型变量的数据交换。例如 BYTE 和 SINT 数据类型。

3) 当需要使用结构体来进行通信的时候注意会发生一些变化。

①当结构体里含有 BOOL 数据类型变量时不能通信。

②当结构体里含有多种数据类型时会有限制或发生变化。例如，NJ 结构体中无论 INT 还是 DINT 的数据类型，实际都会占用 32 个位即 2 个通道，而在 CJ2 中 INT 只占用 1 个通道，DINT 是 2 个通道。

参 考 文 献

- [1] 徐世许. 可编程序控制器原理·应用·网络 [M]. 2 版. 合肥: 中国科学技术大学出版社, 2000.
- [2] 徐世许, 宫淑贞, 彭涛. 可编程序控制器应用指南 编程·通信·联网 [M]. 北京: 电子工业出版社, 2007.
- [3] 彭瑜, 何衍庆. IEC 61131-3 编程语言及应用基础 [M]. 北京: 机械工业出版社, 2009.
- [4] OMRON NJ-series CPU Unit Hardware User's Manual, 2011.
- [5] OMRON NJ-series CPU Unit Software User's Manual, 2011.
- [6] OMRON NJ-series Instructions Reference Manual, 2011.
- [7] OMRON Automation Software Sysmac Studio Version 1, 2011.
- [8] OMRON NJ-series CPU Unit Built-in EtherCAT Port User's Manual, 2011.
- [9] OMRON NJ-series CPU Unit Motion Control User's Manual, 2011.
- [10] OMRON NJ-series Motion Control Instructions Reference Manual, 2011.
- [11] OMRON NJ-series CPU Unit Built-in EtherNet/IP Port User's Manual, 2011.
- [12] OMRON NJ-series Troubleshooting Manual, 2011.
- [13] 张杰. Sysmac NJ501 Training Sysmac Studio I ~ II 篇, 2012.
- [14] 张杰. Sysmac NJ501 Training EtherCAT & Motion 篇, 2012.
- [15] 张杰. Sysmac NJ501 Training EtherNet-IP 篇, 2012.
- [16] <http://www.fa.omron.com.cn>.

机器自动化控制器 原理与应用

JIQI ZIDONGHUA KONGZHIQI

YUANLI YU YINGYONG

地址:北京市百万庄大街22号

邮政编码:100037

电话服务

社服务中心:010-88361066

销售一部:010-68326294

销售二部:010-88379649

读者购书热线:010-88379203

网络服务

教材网:<http://www.cmpedu.com>机工

官网:<http://www.cmpbook.com>机工官

博:<http://weibo.com/cmp1952>封面无防伪

标均为盗版

机械工业出版社 工业技术 / 自动化

ISBN 978-7-111-44136-6

策划编辑◎林春泉 / 封面设计◎鞠杨

ISBN 978-7-111-44136-6



9 787111 441366 >

定价: 69.00元