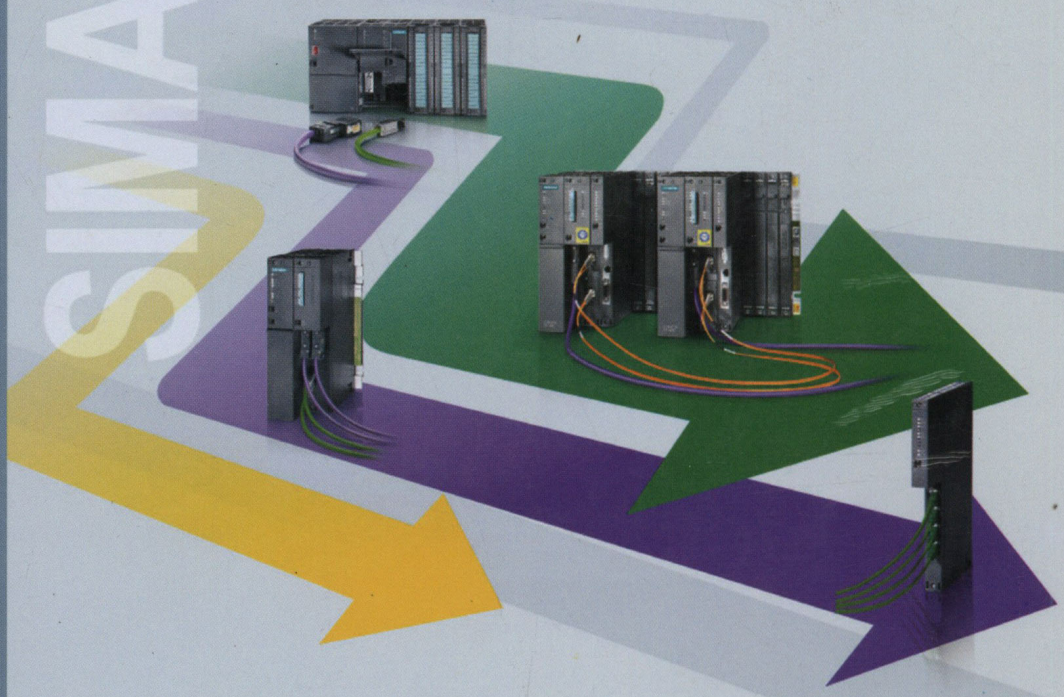
# 西门子S7可编程序控制器
# ——STEP7 编程指南

崔坚　赵欣　任术才　编著

第2版




西门子工业自动化技术丛书

创新工业　知其道　用其妙

SIEMENS

机械工业出版社
CHINA MACHINE PRESS

西门子（中国）有限公司工业业务领域工业自动化与驱动技术集团

西门子工业自动化技术丛书

# 西门子 S7 可编程序控制器
# ——STEP7 编程指南
# 第 2 版

崔　坚　赵　欣　任术才　编著

机 械 工 业 出 版 社

本书介绍了西门子公司的 PLC 系统硬件、CPU 的存储器、数据区、中断和扫描等与用户编程相关的内容。编者结合多年的使用经验就编程语言的使用、项目的创建、调试功能、通信功能及一些典型指令给出了使用示例。

本书的第 2 版中增加了更多 PROFINET 的内容，以示例的方式介绍了 PROFINET IO 设备快速启动功能、设备替换无需存储介质/PG、网络拓扑诊断功能、通过 Web 功能对 CPU 的变量进行监控、浏览 CPU 及模块的诊断信息及整个网络的拓扑结构，以及 PROFINET IRT（等时实时）的 High flexibility。

本书旨在帮助读者由浅入深地学习使用 STEP 7 软件和西门子公司的 PLC，适合做为广大自动化产品工程师快速、深入地掌握西门子公司 PLC 的参考书。

**图书在版编目（CIP）数据**

西门子 S7 可编程序控制器：STEP7 编程指南/崔坚等编著. —2 版. —北京：机械工业出版社，2009.12
（西门子工业自动化技术丛书）
ISBN 978-7-111-28718-6

Ⅰ. 西… Ⅱ. 崔… Ⅲ. 可编程序控制器 Ⅳ. TM571.6

中国版本图书馆 CIP 数据核字（2009）第 235277 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：陈立辉 责任印制：洪汉军
北京瑞德印刷有限公司印刷（三河市胜利装订厂装订）
2010 年 1 月第 2 版第 1 次印刷
184mm×260mm · 26.75 印张 · 1 插页 · 665 千字
0001—3000册
标准书号：ISBN 978-7-111-28718-6
ISBN 978-7-89451-415-8（光盘）
定价：66.00 元（含 1CD）

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 门户网：http：//www.cmpbook.com
销 售 一 部：（010）68326294 教材网：http：//www.cmpedu.com
销 售 二 部：（010）88379649
读者服务部：（010）68993821 **封面无防伪标均为盗版**

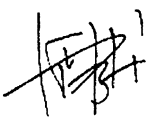
# 序

工业生产率的提高很大程度上取决于工业生产过程中的自动化装置的水平。作为全球自动化领域技术、标准及市场的领导者，西门子公司一直致力于将自动化和驱动产品及系统不断创新，并体现在从传感器、传动设备、可编程序控制器到网络、人机界面、制造执行系统等自控系统的各个层面，着力为用户提供多种创新、可靠、高效、优质的产品、系统、解决方案和服务，在提高用户的行业竞争力的同时，保证用户的最大程度可持续发展，实现长期的投资保障。

西门子公司自动化与驱动集团在生产自动化、过程自动化、楼宇电气安装和电子装配系统领域为中国用户提供全集成自动化（TIA）和全集成能源管理（TIP）解决方案。作为西门子公司全集成自动化和全集成能源管理的控制核心，西门子公司的 S7 系列 PLC 集先进的控制理论、完善的自动化功能与现代通信技术于一体。推出 10 年来，以其灵活的配置、卓越的性能、形式多样、易于扩展的网络通信方式为广大的工业用户所推崇，在中国及全球各个工业领域广泛地成功应用。

《西门子 S7 可编程序控制器——STEP7 编程指南》一书的出版，正是应了广大初学者的要求，由西门子公司中国的资深工程师遵循项目实现的顺序，结合多年技术支持热线的经验，依照入门指南的方式编写而成。在内容上，涵盖了从硬件安装、接线到软件安装、卸载、授权、指令以及网络配置、编程调试等完成项目要涉及的各个方面。对于中国用户在使用产品时，经常遇到的问题给出了解决方案。

希望本书能成为西门子公司产品手册之外的一本学习使用 S7-300/400 PLC 的系统教材和实用工具书，帮助广大学习、使用西门子公司 PLC 的用户快速地理解系统结构，全面掌握 SIMATIC PLC 的应用技能，在项目中充分发挥该产品的功能和性能，从而帮助工业用户提高工业生产率，实现企业最优化运营。



**李永利**

自动化与驱动集团客户支持部总经理

西门子（中国）有限公司

2009 年 9 月

# 前　言

STEP7 是用于 SIMATIC PLC 组态和编程的基本软件包，是西门子公司自动化的入门软件，也是西门子公司自动化各系统的平台软件，可以对 S7-300/400、C7、WinAC 及 ET200 系列智能从站进行编程。STEP7 包含了自动化项目中从项目的启动、实施到测试以及服务的每一阶段所需的全部功能。

STEP7 主要包括以下组件：

- SIMATIC 管理器，用于集中管理所有工具以及自动化项目数据。
- 程序编辑器，用于以 LAD、FBD 和 STL 语言生成用户程序。
- 符号编程器，用于管理全局变量。
- 硬件组态，用于组态和参数化硬件。
- 硬件诊断，用于诊断自动化系统的状态。
- NetPro，用于组态 MPI、PROFIBUS、Ethernet 等网络连接。

除了具有对硬件进行参数化和编程功能外，STEP7 还是 TIA（全集成自动化）的系统平台，不同系统的参数化和编程软件以 STEP7 为平台，集成于其中，使整个控制系统包括 PLC、HMI、驱动等系统具有相同的平台和数据库。

TIA 使整个控制系统具有一致性，体现在以下几个方面：

- 统一的数据管理。
- 统一的编程、组态。
- 统一的通信。
- 诊断功能的一致性。

本书以西门子公司的 PLC 及通信为主，介绍了 PLC 系统硬件以及 CPU 的存储器、数据区、中断和扫描这些与用户编程有关的内容。书中以大量篇幅介绍了编程语言的使用、项目的创建、调试功能、通信功能以及一些指令典型的使用示例，结合编者多年的使用经验以及针对性的应用示例，介绍了复杂的间接寻址的各种使用方法，增强读者阅读程序的能力、扩展编程思路。

随着工业以太网的快速发展，现代控制系统对于系统通信的实时性要求越来越高，为了满足先进控制系统的要求，西门子公司推出实时工业以太网（PROFINET），可以使各层控制都使用同一种网络，实现从上到下的一网到底。因此，我们特别推出了本书的第 2 版。与第 1 版相比，第 2 版中添加了 PROFINET 更多的内容，除了原有的连接分布 IO 设备的 PROFINET IO 通信和 CPU 间通信的 PROFINET CBA 的功能外，还增加了 PROFINET IO 的一些新的功能，以示例的方式介绍了 PROFINET IO 设备快速启动功能、PROFINET IO 设备替换无需存储介质/PG、网络拓扑诊断功能、通过 WEB 功能对 CPU 的变量进行监控、浏览 CPU 及模块的诊断信息及整个网络的拓扑结构，以及 PROFINET IRT（等时实时）的 High flexibility。

本书旨在帮助读者由浅入深地理解和使用 STEP7 软件以及西门子公司的自动化系统，

独立地完成自动化项目的安装、调试任务。希望此书成为广大自动化工程师的良师益友。若有不足之处，敬请指出。

作者

2009 年 9 月

# 目　录


序

前言

**第 1 章　西门子 S7 系列 PLC 系统概述** …… 1

1.1　S7 系列 PLC 介绍 …… 1

1.1.1　S7-200 系列 PLC …… 1

1.1.2　S7-300 系列 PLC …… 1

1.1.3　S7-400 系列 PLC …… 2

1.2　远程分布式 I/O …… 3

1.3　其他控制系统 …… 4

1.3.1　SIMATIC C7 控制器 …… 4

1.3.2　基于 PC 的 SIMATIC WinAC 控制器 …… 4

1.4　STEP7 编程软件 …… 4

1.4.1　编程功能 …… 4

1.4.2　TIA 软件平台 …… 7

**第 2 章　西门子 S7-300/400 系列 PLC 硬件系统** …… 9

2.1　电源模块 …… 9

2.1.1　S7-300 系列 PLC 的 SITOP 电源模块 …… 9

2.1.2　S7-400 系列 PLC 的电源模块 …… 9

2.2　机架 …… 10

2.2.1　S7-300 系列 PLC 机架 …… 10

2.2.2　S7-400 系列 PLC 机架 …… 10

2.3　CPU …… 11

2.3.1　S7-300/400 系列 PLC CPU 简介 …… 11

2.3.2　S7-300/400 系列 PLC CPU 操作模式 …… 12

2.3.3　S7-300/400 系列 PLC CPU 的存储区域 …… 12

2.3.4　S7-CPU 过程映像区的功能 …… 15

2.3.5　S7-CPU 过程映像区的划分 …… 16

2.4　信号模块 …… 16

2.4.1　数字量输入模块 …… 16

2.4.2　数字量输出模块 …… 18

2.4.3　数字量输入/输出模块 …… 20

2.4.4　模拟量输入模块 …… 20

2.4.5　模拟量输出模块 …… 34

2.4.6　模拟量输入/输出模块 …… 37

2.4.7　特殊模块 …… 38

2.5　通信模块 …… 39

2.6　功能模块 …… 40

2.7　接口模块 …… 41

2.7.1　S7-300 系列 PLC 的接口模块 …… 41

2.7.2　S7-400 系列 PLC 的接口模块 …… 42

**第 3 章　西门子 S7-300/400 系列 PLC 系统扩展** …… 43

3.1　S7-300 系列 PLC 的中央扩展 …… 43

3.2　S7-400 系列 PLC 的中央扩展 …… 44

3.3　S7-300/400 系列 PLC 的分布式扩展 …… 45

**第 4 章　S7 系列 PLC 编程软件——STEP7 简介** …… 48

4.1　STEP7 编程软件的订货版本 …… 49

4.2　STEP7 编程软件的安装 …… 49

4.2.1　硬件要求 …… 49

4.2.2　软件要求 …… 50

4.2.3　语言设置 …… 50

4.2.4　安装步骤 …… 51

4.3　STEP7 编程软件的卸载 …… 55

4.4　授权管理功能 …… 55

4.4.1　授权的种类 …… 55

4.4.2　授权管理器 …… 55

4.4.3　使用浮动授权 …… 56

4.5　STEP7 标准软件包 …… 57

4.5.1　SIAMTIC Manager …… 58

4.5.2　硬件配置 …… 59

4.5.3　编程工具 …… 59

4.5.4　符号编辑器 …… 60

4.5.5　硬件诊断 …… 60

4.5.6　NetPro 网络配置 …… 61

4.6　STEP7 扩展软件包 …… 61

4.6.1　工程工具 …… 62
4.6.2　运行版软件 …… 63
4.6.3　人机接口 …… 63
**第5章　数据类型与地址区** …… 64
5.1　S7-300/400 系列 PLC 的数据类型 …… 64
5.1.1　基本数据类型 …… 64
5.1.2　复合数据类型 …… 69
5.1.3　参数类型 …… 72
5.2　S7-300/400 系列 PLC 地址区 …… 72
5.2.1　CPU 地址区的划分及寻址方法 …… 72
5.2.2　全局变量与区域变量 …… 76
5.2.3　地址区数据的排列 …… 76
**第6章　编程指令** …… 77
6.1　指令的处理 …… 77
6.1.1　LAD 指令处理 …… 77
6.1.2　STL 指令处理 …… 79
6.2　位逻辑指令 …… 81
6.2.1　触点指令 …… 81
6.2.2　线圈指令 …… 82
6.2.3　RLO 操作指令 …… 85
6.2.4　立即读与立即写 …… 86
6.3　比较指令 …… 87
6.4　转换指令 …… 89
6.5　计数器指令 …… 92
6.6　数据块操作指令 …… 94
6.7　逻辑控制指令 …… 95
6.7.1　LAD 跳转指令 …… 96
6.7.2　STL 跳转指令 …… 96
6.8　整数运算指令 …… 100
6.9　浮点运算指令 …… 102
6.10　赋值指令 …… 103
6.10.1　LAD 赋值指令 …… 104
6.10.2　STL 装载、传递指令 …… 105
6.11　程序控制指令 …… 107
6.11.1　LAD 程序控制指令 …… 107
6.11.2　STL 程序控制指令 …… 109
6.12　移位和循环指令 …… 112
6.13　状态位指令 …… 115
6.14　定时器指令 …… 119
6.15　字逻辑指令 …… 126
6.16　累加器指令 …… 127
**第7章　程序块** …… 132
7.1　用户程序中的程序块 …… 132
7.1.1　组织块与程序结构 …… 133
7.1.2　用户程序的分层调用 …… 134
7.2　组织块 …… 135
7.2.1　组织块的类型与优先级 …… 135
7.2.2　组织块的区域数据区堆栈 …… 139
7.3　函数 …… 142
7.3.1　函数的接口区 …… 142
7.3.2　无形参函数 …… 144
7.3.3　带有形参的函数 …… 144
7.3.4　函数嵌套调用时，允许参数传递的数据类型 …… 146
7.4　函数块 …… 148
7.4.1　函数块的接口区 …… 149
7.4.2　函数块与背景数据块 …… 150
7.4.3　函数块嵌套调用时，允许参数传递的数据类型 …… 152
7.5　数据块 …… 154
7.5.1　共享数据块 …… 154
7.5.2　背景数据块 …… 156
7.5.3　基于 UDT 的数据块 …… 157
7.6　系统函数与系统函数块 …… 158
7.7　STEP7 集成用于逻辑运算的函数与函数块 …… 167
7.8　用于特殊功能的函数与函数块 …… 168
**第8章　地址寻址** …… 169
8.1　绝对地址寻址与符号地址寻址 …… 169
8.2　间接寻址 …… 170
8.2.1　存储器间接寻址 …… 170
8.2.2　寄存器间接寻址 …… 174
8.3　程序块参数——POINTER 与 ANY 数据类型指针 …… 178
8.3.1　POINTER 数据类型指针 …… 178
8.3.2　ANY 数据类型指针 …… 181
8.4　FB 在多重数据块中的寻址 …… 183
**第9章　使用 STEP7 创建和编辑项目** …… 186
9.1　创建一个项目 …… 186
9.1.1　使用 SIMATIC Manager 向导功能创建一个项目 …… 186
9.1.2　直接创建一个项目 …… 189
9.2　项目基本配置 …… 190

9.2.1 项目属性配置 …… 190
9.2.2 项目用户化设置 …… 191
9.3 硬件配置界面 …… 191
9.4 配置中央机架及扩展机架 …… 192
9.4.1 配置 S7-300 系列 PLC 中央机架 …… 192
9.4.2 配置 S7-300 系列 PLC 扩展机架 …… 194
9.4.3 配置 S7-400 系列 PLC 中央机架 …… 195
9.4.4 配置 S7-400 系列 PLC 扩展机架 …… 196
9.5 CPU 参数配置 …… 198
9.5.1 常规界面 …… 198
9.5.2 启动界面 …… 198
9.5.3 同步循环中断 …… 200
9.5.4 循环/时钟寄存器 …… 201
9.5.5 保持存储区 …… 202
9.5.6 存储区（不适用 S7-300 系列 PLC CPU） …… 203
9.5.7 中断 …… 204
9.5.8 日期中断 …… 205
9.5.9 循环中断 …… 206
9.5.10 诊断/时钟 …… 207
9.5.11 程序保护 …… 208
9.5.12 分配通信资源（不适用 S7-400 系列 PLC CPU） …… 209
9.6 I/O 模块参数配置 …… 210
9.6.1 数字量 I/O 模块参数配置 …… 210
9.6.2 模拟量模块参数配置 …… 213
9.7 更新硬件条目 …… 215
9.8 远程 I/O 扩展 …… 216
9.8.1 配置 PROFIBUS-DP 远程 I/O 站 …… 216
9.8.2 配置 PROFINET IO 远程 I/O 站 …… 217
9.8.3 远程 I/O 站点的诊断 …… 219
9.9 符号地址寻址 …… 221
9.10 生成用户程序 …… 223
9.10.1 生成系统数据 …… 223
9.10.2 生成逻辑程序块 …… 224
9.10.3 地址替换功能 …… 225
9.10.4 块比较 …… 226
9.10.5 生成变量监控表 …… 227
9.10.6 程序块的一致性检查 …… 227
9.11 生成用户库函数 …… 228
9.12 复制其他项目中的程序块 …… 229
9.13 生成源文件 …… 230
9.14 生成地址交叉参考 …… 230
9.14.1 交叉参考表 …… 230
9.14.2 在程序编辑器中快速查询地址的位置 …… 232
**第 10 章 PLC 的通信功能** …… 233
10.1 网络概述 …… 233
10.2 MPI 网络 …… 235
10.2.1 MPI 的种类 …… 235
10.2.2 MPI 网络的通信速率 …… 235
10.2.3 MPI 网络的拓扑结构 …… 235
10.2.4 PLC 通过 MPI 网络的通信方式 …… 236
10.2.5 全局数据包通信方式 …… 236
10.2.6 不需配置连接的通信 …… 237
10.2.7 需要配置连接的通信 …… 239
10.2.8 PLC 通过 MPI 与 HMI 通信 …… 243
10.3 PROFIBUS 网络 …… 243
10.3.1 PROFIBUS 接口的种类 …… 243
10.3.2 PROFIBUS 的访问机制 …… 243
10.3.3 PROFIBUS 网络的通信速率与通信距离 …… 244
10.3.4 PROFIBUS 网络拓扑结构 …… 244
10.3.5 PROFIBUS 支持的通信协议与服务 …… 248
10.3.6 PROFIBUS-DP 通信 …… 249
10.3.7 PROFIBUS-FDL 通信 …… 251
10.3.8 PROFIBUS-S7 通信 …… 254
10.3.9 PLC 通过 PROFIBUS 与 HMI 通信 …… 255
10.4 工业以太网 …… 257
10.4.1 工业以太网接口的种类 …… 257
10.4.2 工业以太网通信介质 …… 257
10.4.3 工业以太网络交换机 …… 258
10.4.4 工业以太网拓扑结构 …… 259
10.4.5 工业以太网支持的通信协议与服务 …… 261
10.4.6 配置 S5 兼容通信 …… 263
10.4.7 配置 S7 通信连接 …… 266

10.4.8　PLC 通过以太网与 HMI 通信 …… 266
10.4.9　使用 DCP 直接配置以太网接口 …… 267
10.5　PROFINET 通信 …… 268
10.5.1　PROFINET IO 通信 …… 268
10.5.2　PROFINET IO 快速启动 …… 269
10.5.3　PROFINET IO 网络拓扑 …… 274
10.5.4　PN IO 设备替换无需存储介质或 PG …… 284
10.5.5　PN IO 网络诊断-Web …… 288
10.5.6　PN IRT-High　flexibility …… 298
10.5.7　PROFINET CBA 通信 …… 302
10.6　串行通信 …… 305
10.6.1　串行通信接口类型及连接方式 …… 305
10.6.2　串行通信支持的通信协议 …… 308
10.6.3　串行通信模块与相应的通信函数 …… 309
10.6.4　通信函数的调用 …… 309
10.6.5　MODBUS RTU 通信协议 …… 311
**第 11 章　功能模块的使用** …… 316
11.1　高速计数器模块 …… 316
11.1.1　高速计数器的应用场合 …… 316
11.1.2　高速计数器的原理 …… 316
11.1.3　高速计数器可以连接的信号 …… 316
11.1.4　脉冲信号的采集方式 …… 318
11.1.5　高速计数器的计数模式 …… 319
11.1.6　高速计数器开始计数的条件 …… 319
11.1.7　高速计数器的其他功能 …… 320
11.1.8　具有高速计数功能的模块 …… 320
11.1.9　FM350-1 高速计数器的使用 …… 321
11.1.10　FM350-2 高速计数器的使用 …… 325
11.1.11　S7-300C 系列 PLC 集成高速计数器的使用 …… 329
11.1.12　ET200S 高速计数器的使用 …… 331
11.2　FM352-5 高速布尔处理器 …… 333
11.2.1　工作方式 …… 334
11.2.2　输入输出端子接线 …… 336
11.2.3　模块的参数化 …… 338
11.2.4　编程 …… 340
11.2.5　FM352-5 的编程资源 …… 345
11.3　定位模块概述 …… 347
11.3.1　双速电动机的定位控制 …… 347
11.3.2　步进电动机的定位控制 …… 348
11.3.3　伺服电动机的定位控制 …… 349
11.4　FM354 伺服电动机定位模块的使用 …… 350
11.4.1　FM354 模块的输入输出接口 …… 350
11.4.2　FM354 模块的操作模式 …… 351
11.4.3　FM354 模块的参数化 …… 352
11.4.4　MD 机械参数 …… 353
11.4.5　SM 增量表的配置 …… 356
11.4.6　WZK 工具补偿参数的配置 …… 357
11.4.7　VP 自动程序的编写 …… 357
11.4.8　将参数化的数据传送到 FM354 中 …… 361
11.4.9　FM354 测试功能 …… 362
11.4.10　FM354 系统数据生成 SDB 文件 …… 365
11.4.11　进入 FM354 模块地址 …… 366
11.4.12　FM354 模块的编程 …… 366
11.5　FM355 PID 控制模块 …… 372
11.5.1　应用概述 …… 372
11.5.2　硬件安装与接线 …… 372
11.5.3　系统配置及参数设置 …… 377
11.5.4　编程控制 FM355 模块 …… 381
11.5.5　监控、调试 …… 388
11.5.6　控制器参数的优化 …… 388
11.5.7　通过操作面板的后援操作 …… 389
**第 12 章　程序调试** …… 391
12.1　建立与 CPU 的连接并进行设置 …… 391
12.1.1　设置 PG/PC 接口 …… 391
12.1.2　建立在线连接 …… 393
12.1.3　显示和改变 CPU 的操作模式 …… 394
12.1.4　显示和改变 CPU 的时钟 …… 394
12.1.5　在线更新硬件固件版本 …… 394
12.2　程序的下载、上传、复位操作 …… 395
12.2.1　程序的下载 …… 395
12.2.2　程序的上传 …… 396
12.2.3　CPU 存储器复位 …… 396
12.2.4　删除 CPU 中的程序块 …… 397
12.3　使用变量表进行调试 …… 397
12.3.1　变量表的创建 …… 397

12.3.2 建立变量表与 CPU 间的通信 …… 398
12.3.3 在变量表中输入变量 …… 398
12.3.4 变量的监控和修改 …… 399
12.3.5 强制变量 …… 400
12.4 使用程序编辑器调试程序 …… 400
12.4.1 调试 LAD/FBD 程序 …… 400
12.4.2 调试 STL 程序 …… 401
12.4.3 使用断点单步调试程序 …… 402
12.4.4 调试数据块 …… 404
12.5 硬件诊断 …… 404
12.5.1 硬件的诊断符号 …… 404
12.5.2 模板诊断信息 …… 405
12.6 使用模拟器 S7 PLCSIM 测试用户程序 …… 408
12.6.1 设置 PLC 模拟器通信接口 …… 408
12.6.2 设置 CPU 的操作模式 …… 408
12.6.3 触发中断 …… 409
12.6.4 回放功能 …… 409
**第 13 章 打印和归档程序** …… 411
13.1 打印项目文档 …… 411
13.2 程序归档 …… 412
**附录 寻求帮助** …… 414
**缩写表** …… 416
**参考文献** …… 418

# 第 1 章　西门子 S7 系列 PLC 系统概述

1979 年西门子公司推出了 S5 系列 PLC（Programmable Logic Controller，可编程序逻辑控制器），经过不同行业多年的应用，系统的稳定性、可靠性及低故障率得到工控界的认可，进入 20 世纪 90 年代，随着现代通信技术和 IT 技术的迅猛发展，S5 系列 PLC 的配置方法、CPU（Central Processing Unit，中央处理器）的处理能力、网络的通信能力越来越不能满足现代化控制的要求，即不能满足对实时性、快速性、大量的网络通信和数据管理、分布式控制、集成现场设备的快速诊断等要求。为了保持西门子公司在工控业的领先地位，1994 年西门子公司推出了 S7 系列 PLC，与 S5 系列 PLC 相比，除了保持原有的控制功能和系统的稳定性外，在 CPU 运算速度、程序执行效率、故障自诊断、网络通信、面向工艺和运动控制功能上有了质的飞跃，并为后续的系统整合打下了良好的基础。根据控制要求和驱动输入、输出点的能力，S7 控制系统划分为 3 个子系列，即 S7-200 系列、S7-300 系列及 S7-400 系列。此外，各种类型的分布式 I/O 系统也被大量地应用。

## 1.1　S7 系列 PLC 介绍

### 1.1.1　S7-200 系列 PLC

S7-200 系列是小型 PLC 系统，其 CPU 如图 1-1 所示，具有串行连接的模块化扩展功能，设计紧凑，CPU 集成输入、输出信号接口功能，输入点集成高速计数器、报警和中断等功能，适合最大输入、输出 100 点左右的控制应用。S7-200 系列 PLC 通过通信模块可以扩展不同的网络接口，如通过 CP243-2 模块扩展 ASI（执行器与传感器接口）网络主站接口；通过 EM277 模块扩展现场总线 PROFIBUS－DP 从站接口；通过 CP243-1 模块扩展以太网接口。此外，S7-200 系列 PLC 还具有远程编程、维护功能。S7-200 系列 PLC 简单的定位功能使控制功能更加完善。S7-200 系列 PLC 使用 STEP7 MICRO WIN 软件进行编程，在本书中不作介绍。

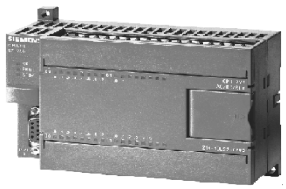


图 1-1　S7-200 系列 PLC CPU

### 1.1.2　S7-300 系列 PLC

S7-300 系列是中型 PLC 系统，具有模块化扩展功能，设计紧凑，适合最大输入、输出 1000 点左右的控制应用。如图 1-2 所示，S7-300 系列 PLC CPU 中集成了各种中断处理能力，如时间中断、报警中断、循环中断等。S7-300 系列 PLC 具有强大的网络通信能力，如通过 CP343-2 模块可以使 S7-300 系列 PLC 作为 ASI（执行器与传感器接口）网络中的主站，在现场总线 PROFIBUS 的应用中完全支持各种通信方式和服务，在主—从通信方式中，S7-300 系列 PLC 既可以作为 PROFIBUS-DP 的主站，也可以作为从

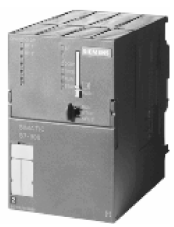


图 1-2　S7-300 系列 PLC CPU

站；在主站间通信方式中，S7-300 系列 PLC 利用不同的通信协议和服务（PROFIBUS-FMS/S7/FDL）可以非常灵活地与通信方进行数据交换；通过工业以太网，S7-300 系列 PLC 之间或与 HMI（Human-Machine Interface，人机接口）可以进行大数据量的通信，利用基于以太网的 PROFINET 总线技术，可以实现数据的实时通信。通过扩展具有独立处理能力的特殊模块，例如功能模块（FM），S7-300 系列 PLC 可以实现高速计数、单轴定位、具有插补功能的 4 轴路径控制，而不会影响 CPU 的处理速度。S7-300 系列 PLC 使用 STEP7 进行编程，是 S7 系列 PLC 的主流产品。

### 1.1.3 S7-400 系列 PLC

S7-400 系列是大型 PLC 系统，如图 1-3 所示，具有模块化扩展功能，可以连接数万点输入、输出信号。与 S7-300 系列 PLC 相比，S7-400 系列 PLC CPU 中集成了强大的中断处理能力，如数量和种类更多的时间中断、报警中断和循环中断等，即使在同一中断类型中还可以选择不同触发事件的中断；完全覆盖 S7-300 系列 PLC 通信服务和通信协议，在现场总线 PROFIBUS-DP 上实现等时数据通信，保证各个从站的输入信号在 CPU 中同时处理，CPU 的输出命令在各个从站中同时响应；此外，S7-400 系列 PLC 还具有连接更多通信设备的能力和更多的通信资源。同样，S7-400 系列 PLC 通过扩展，具有独立处理能力的功能模块（FM），可以实现高速计数、单轴定位等工艺控制而不会影响 CPU 的处理速度。S7-400 系列 PLC 具有更强的实时处理能力，最多可以在一个站上插入 4 个 CPU 完成同一个控制任务，各 CPU 通过背板总线进行非常快的数据交换。S7-400 系列 PLC 使用 STEP7 进行编程，是 S7 系列 PLC 的主流产品。

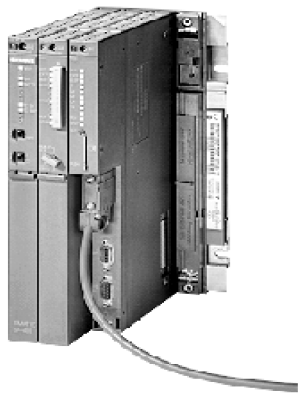


图 1-3 S7-400 系列 PLC CPU

在 S7-400 系列 PLC 中还包括 H（冗余）系统和 F（故障安全）系统，如图 1-4 所示，S7-400H 系统具有冗余的电源、CPU、通信处理器、现场总线、通信接口、输入与输出信号通道等，单一设备故障不会造成整个控制系统的停机，提高了控制系统的可用性；与 S7-400H 冗余系统相比，S7-400F 故障安全系统通过对程序、现场总线、输入与输出信号的再次校验，保证了整个控制系统的可靠性，从而保证了人身的安全以及环境不会遭到破坏。通过一个简单的例子可以区别两个控制系统，例如同一个外部信号分别接入到两个输入模块中，如果其中一路信号连接的模块故障或断线，信号可能在 CPU 中产生差异，对于 S7-400H 系统，将选择预先设定的值继续运行；对于 S7-400F 系统，设备的控制将切换到故障安全模式，通常情况下为停车模式，为了达到设计的安全等级，整个 S7-400F 系统要满足规定的平均无故障时间的要求。

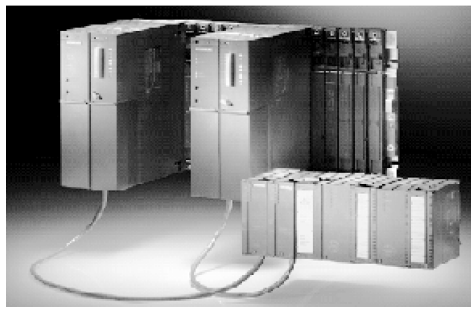


图 1-4 S7-400H CPU

在很多的应用中，将冗余系统和故障安全系统结合使用，如 S7-400H/F，虽然不能提高控制系统的可靠性（F 等级），却可以提高系统的可用性（冗余特点）。STEP7 V5.3 以上版

本已集成冗余系统的配置软件，不需要另外购买，而 F 系统软件需要额外购买。

## 1.2　远程分布式 I/O

在早期的工厂设计中，选择中央扩展模式安装输入、输出模块，即 CPU 通过背板总线快速地访问 I/O 模块，I/O 模块与 CPU 保持非常近的距离（通常为几米，最远可以扩展到 100m），这样所有外围信号接线汇总到一起，造成 PLC 柜内接线复杂、凌乱，如图 1-5a 所示，对于安装、现场调试以及后期的维护都比较困难。如果现场信号比较远，为了减小信号的衰减而必须选择较粗截面的传输电缆，从而增加了工程的费用。现代工厂设计中，如卷烟厂、物流、钢铁厂，使用大量的远程分布式 I/O 替代原有中央扩展 I/O，这样可以将 I/O 模块放在靠近设备的现场，如图 1-5b 所示，每个远程分布式 I/O 站通过简单的一根现场总线（使用屏蔽双绞线的现场总线 PROFIBUS-DP 或实时工业以太网 PROFINET I/O）与 CPU 进行数据交换。使用远程分布式 I/O 有下列好处：

- 减少布线的时间和工程费用；
- 方便设备的调试；
- 简化后期维护。




a)　　b)

图 1-5　中央扩展 I/O 与远程分布式 I/O 接线对比
a）中央扩展 I/O 接线　b）远程分布式 I/O 接线

不同类型的远程分布式 I/O 站可以满足不同现场的需求，例如在 ET200 远程分布式 I/O 系列中，有适合连接大量 I/O 点的 ET200M；有适合紧凑型的 ET200B；有适合小点数、灵活的 ET200S，如图 1-6 所示。在 ET200S 中，可以安装小功率的负载馈电器，从而节省从站电器柜的安装空间；有适合高防护等级 IP67 的 ET200PRO，如图 1-7 所示；有适合安装于防

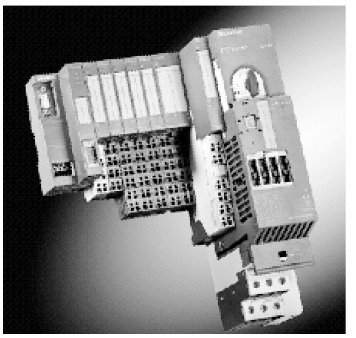


图 1-6 ET200S



图 1-7 ET200PRO

爆区的 ET200ISP，如图 1-8 所示。除了 ET200 系列外，一些驱动装置、仪表等设备也可以作为分布式 I/O 站与主站 CPU 进行通信。

远程分布式 I/O 站也可以带有 CPU，控制本站的I/O 信号，并与主站进行数据交换，这样的远程站称为智能 I/O 站，在网络或主 CPU 故障时可以独立控制设备。各种类型的分布式 I/O 站在满足现场需求的同时，也给设计人员提供了更好的设计思路。



图 1-8 ET200ISP

## 1.3 其他控制系统

### 1.3.1 SIMATIC C7 控制器

将 S7-300 系列 PLC 系统与操作面板（OP/TP）集成一体而形成 SIMATIC C7 控制系统，操作面板与 PLC 合二为一不需要外部电缆连接，整体尺寸更加紧凑，从而节省了机旁操作箱的安装空间，SIMATIC C7 控制系统的编程和操作与 S7-300 系列 PLC 相同。

### 1.3.2 基于 PC 的 SIMATIC WinAC 控制器

在 PC 上安装 PROFIBUS-DP 通信处理器和软件控制器，通过总线连接远程 I/O 而形成 SIMATIC WinAC 控制系统。SIMATIC WinAC 方便集成用户 C/C ++ 程序，插入实时 I/O 卡，可以完成高速响应的控制任务。SIMATIC WinAC 有两种类型：软件 PLC 和插槽型 PLC。

## 1.4 STEP7 编程软件

### 1.4.1 编程功能

使用 STEP7 软件可对 S7-300/400 系列 PLC、SIMATIC C7、SIMATIC WinAC 及 ET200 系列智能从站进行编程。STEP7 包含了自动化项目中从启动、实施到测试及维护的每一阶段所

需的全部功能。

STEP7 是用于 SIMATIC PLC 配置和编程的基本软件包。它包括功能强大、适用于各种自动化项目的工具。

STEP7 主要包括以下组件：

- SIMATIC Manager：用于集中管理所有工具以及自动化项目数据。
- 程序编辑器：用于编辑 LAD（梯形图）、FBD（功能块图）和 ST（结构文本）语言生成用户程序。
- 符号编程器：用于编辑符号表和配置通信及消息。
- 硬件配置：用于配置和参数化硬件。
- 硬件诊断：用于诊断自动化系统的状态。

NetPro：

用于配置网络连接及通信。

STEP7 中集成了三种编程语言，通过安装可选软件 Engineer Tool（工程工具），可以扩展编程语言的种类，工程工具面向特定功能，简化和增强自动化控制任务。下列工具可供编程者选择：

（1）S7-SCL

S7-SCL（结构化控制语言）是基于 PASCAL 的高级语言，符合 DIN EN/IEC 61131-3 中定义的高级文本语言 ST（结构文本）。S7-SCL 尤其适用于复杂算法和算术功能的编程以及数据处理任务。

使用 S7-SCL 可以达到下列目的：

- 通过应用功能强大的编程语句（例如 IF... THEN... ELSE），简便、快速、正确地开发程序。
- 改进程序可读性和结构，更易于理解。
- 使用符号生成程序，保证程序的正确修改和重复使用性。
- 使用 Debugger（调试程序），简化在高级语言中的程序调试。

用户可在很短的时间内，经济地为所有自动化任务提供“公式化”的解决方案。

（2）S7-GRAPH

SIMATIC 软件包 S7-GRAPH 基于 STEP7 编程软件。它适合顺控工艺编程，将控制工艺分成不同的“步”，在每一步中填写触发的事件，适合工艺人员使用配置的方式完成工艺编程要求，在标准化的用户界面中可以对实现工艺的“步”进行直观、快速的配置与编程（符合 IEC 61131-3、DIN EN 61131 标准）。

在每个“步”中定义具体的操作（Action）及其执行控制。跳转（Transition）指令控制下一步执行的条件。每一步的执行都根据定义好的互锁和监控条件进行。与 LAD、FBD、STL 相比，其优点如下：

- 直接按工艺流程图生成图形化的程序。
- LAD、FBD 和 STL 主要用于逻辑控制。对于 S7-GRAPH，控制顺序非常重要。
- 采用顺序链，直观图形化地显示控制过程；易于程序维护和调整。
- 在调试阶段可以选择单“步”、手动“步”、自动“步”传送方式进行调试，方便并节省调试时间。

- 采用集成诊断功能进行故障排查，减少停止时间。

（3）S7-HiGRAPH

S7-HiGRAPH 是一种适合于技术工艺人员、编程人员、调试工程师、操作人员以及维护和维修人员的通用工具。

在使用 S7-HiGRAPH 时，示教将代替编程，并通过在状态图中映射技术功能对象（例如阀门、电动机等）来实现。基本宗旨是将自动化任务分为具体的功能单元。技术功能对象或功能单元的特性以状态图的形式加以描述。

工艺人员应首先勾画出大致结构，并定义功能单元及其特性，然后由编程人员处理具体细节。

典型应用：汽车工业（例如发动机、轴和减速箱的制造）、塑料机械、食品和饮料机械、包装机械、机床、卷取机和专用机械。

S7-HiGRAPH 的优点：

- 配置、上线调试以及维护、保养和维修，所有人员均使用相同的工具。
- 点击按钮，即可以根据状态图生成执行程序。
- 通过易于集成的信号传送和监控功能，可以非常简便地检测故障，降低停机的时间。
- 一旦生成状态图，即可反复地使用。

在编辑状态图时，可以将自动化任务拆分为具体的机械功能单元，每个功能单元的特性都以状态图的形式加以描述。各种操作都以状态（State）的形式触发，例如初始化(Initialize)、拧紧（Tighten）、松开（Loosen）。

由于 S7-HiGRAPH 和 S7-GRAPH 工程工具可提供两种不同的生产过程视图，并能相互最佳实现，也可在一个项目当中组合使用。

（4）CFC

CFC（连续功能图）工程工具是专为那些需要为工厂进行配置和编程的工程师而备的。

使用 CFC，可在参数输入的同时，将工艺技术参数转换为可执行的自动化系统程序。它只需将预置模块链接在一起，然后设置其参数即可，无需高级编程知识。

与 LAD、FBD 和 STL 相比，其优点如下：

- 在工程制图阶段即可优化使用。
- 降低图形链接配置要求。
- 重复使用性。
- 上手快速、简单。
- 快速、直观链接预置功能。
- 使用 S7-SCL，简便生成定制模块。
- 生成整个技术工艺程序。
- 控制结构一目了然。
- 离线测试，缩短调试时间。
- 在线修改，高度透明，工厂可用性高。

CFC 工程工具还可用于生成 SIMATIC S7 和 SIMATIC WinAC 自动化解决方案。任何模块都可以根据工艺规范相互链接，例如开环和闭环控制数据，甚至是配置和归档整个信息流。

配置界面是一种绘图界面，设置有预置模块，并根据工艺条件相互链接。对于工程师来

说，无需任何自动化系统知识，CFC 工程工具可自动处理目标系统资源管理。在标准库中，提供主要的功能块，并可根据需要使用用户生成的块进行扩展。使用 CFC 工程工具对 S7-300、S7-400 系列 PLC 编程时，必须使用 S7-SCL 作为编译语言。

**注意：**

CFC 工程工具适合较大、复杂的应用，但是需要占用大的 CPU 工作内存，建议使用 S7 CPU314 以上的 CPU。

### 1.4.2　TIA 软件平台

在工业控制中，一个大的控制系统由不同的设备组成，如 PLC、HMI、仪表、传动系统等，不同的系统或设备有不同的软件和用户接口，每个独立的系统或设备需要单独配置、编程、调试和启动，使用传统的解决方案则需要大量的时间和一些特有的经验配置系统间的数据通信，这样才能整合不同的系统一起正常地运行。

1997 年，西门子公司自动化与驱动集团提出了“全集成自动化”（Totally Integrated Automation，TIA）的概念。TIA 的概念起源于 PLC 系统的集成，并将所有的功能推广到其他系统中，为生产线提供了一种优秀的解决方案，使控制任务基于一个单一的、完全集成各系统和一致性的操作平台。

对设计人员和用户而言，从最初的规划与设计、工程与实施，到安装与调试、运行与维护，以至系统升级改造，TIA 使企业在整个生命周期中获得了最高的生产力和产品质量，并明显地降低了项目成本。此外，TIA 还能大大地缩短产品上市和系统投入运行的时间。

STEP7 除了具有对硬件进行参数化和编程功能外，还是 TIA 的系统平台，不同系统的参数化和编程软件以 STEP7 为平台并集成于 STEP7 中，使整个控制系统，包括 PLC、HMI、驱动等具有相同的控制平台和数据库。

TIA 使整个控制系统具有一致性，它的优点体现在以下几个方面：

（1）统一的数据管理

以 STEP7 为操作平台，具有相同的数据库，这种统一的数据管理机制，不仅可以减少初始阶段的费用，还可以降低出错率，提高系统诊断的效率，其数据管理功能具体体现在以下几个方面：

- 统一的符号表：无论使用 SIMATIC 家族中的哪一个配置软件，都可以通过全局数据库共享一个统一的符号表。
- 变量名自动映射：STEP7 中定义的变量，通过内部的传送，可以被 SIMATIC HMI（人机接口）软件使用，例如 HMI 软件 WinCC 集成安装于 STEP7 后，可以将 STEP7 符号表中定义的变量上传到 WinCC 中使用。
- 多用户功能：一个大的项目往往需要多人同时编程，这样需要创建一些小的项目，通过多用户功能可以将不同的小项目整合为一个完整的项目。

（2）统一的编程、配置

SIMATIC 工业软件家族具有统一、友好的界面。通过集成安装，用户可以在 SIMATIC Manager 的统一界面下工作，在 STEP7 中直接调用其他软件。这种界面的一致性和集成性大大方便了对整个 TIA 系统的编程和配置。

（3）统一的通信

各个系统经过 STEP7 的整合，系统或设备间的通信更加透明，只需要简单的配置和指定通信接口而不需要特有的通信经验。TIA 中的各种网络可以进行互连，集成的路由功能可以使编程及维护人员在整个网络中任一点实现跨网络的下载、诊断和通信等功能，使整个系统的安装调试更加容易。

（4）诊断功能的一致性

通过 PLC 功能可以将系统或设备诊断、报警信息发送到 HMI 系统中，并可以在 HMI 中观察触发报警信息的条件。在 HMI 中建立的报警信息，如一个位信号，通常是由 HMI 按配置的时间扫描这个位的状态，而由 PLC 的触发的报警信息功能，只有在位信号改变时，才将报警信息发送到 HMI 中，大大减少了网络通信的负载。

# 第 2 章　西门子 S7-300/400 系列 PLC 硬件系统

STEP7 软件编程的接口是 PLC 硬件系统，CPU 采集输入模块信号进行处理，并将逻辑操作结果通过模块输出，编程之前首先需要了解一下 S7-300/400 系列 PLC 的硬件系统。

## 2.1　电源模块

### 2.1.1　S7-300 系列 PLC 的 SITOP 电源模块

S7-300 系列 PLC 的 SITOP 电源模块（Power Supply，PS），如图 2-1 所示。向 CPU 或扩展接口模块提供 24V 电源，CPU 和扩展接口模块将 24V 电源转换为 5V 电源，向背板总线供电，通过背板总线，CPU 监控所有与背板总线连接的 I/O 模块，CPU 和扩展接口模块后面最多可以连接 8 个模块，8 个模块消耗背板总线总的电流（在模块样本手册中，可以查看每个模块所消耗背板总线的电源）不能超过 CPU 和扩展接口模块输出电流（CPU 通常为 1.2A）。SITOP 电源模块按输出容量可分为 2A、5A、10A 和 20A，输入电源可以选择交流 220V 或 380V，SITOP 电源模块具有电源滤波和输出短路保护功能，除向 CPU 供电以外，SITOP 电源模块还可以给 24V 控制回路供电。有些 SITOP 电源模块输出端内部带有反向二极管保护，可以并联连接作为冗余电源输出。SITOP 电源模块与 S7-300 系列 PLC 背板总线之间没有连接，可以与 CPU 机架分离安装，CPU 不能对电源模块进行诊断。可以购买其他品牌的电源模块，但是必须满足 CPU 的供电要求（如输出功率、滤波功能等），否则很容易将 CPU 烧毁。

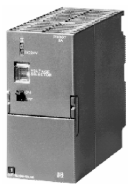


图 2-1　SITOP 2A 电源模块

### 2.1.2　S7-400 系列 PLC 的电源模块

S7-400 系列 PLC 的电源模块如图 2-2 所示，直接向背板总线提供 5V 和 24V 电源（CPU 和以太网模块需要消耗 24V 电源），电源模块按输出容量可分为 4A、10A 和 20A，输入电源可以选择交流 120/230V 或直流 24V，所有安装于一个机架（如果扩展机架需要主机架供电时则为两个机架）上的模块消耗 5V 和 24V 电源总数不能超出电源模块的输出容量（在模块样本手册中，可以查看每个模块所消耗背板总线 5V 和 24V 的电源）。与 S7-300 系列 PLC 的电源模块不同，S7-400 系列 PLC 的电源模块必须安装在机架背板上，所以不能与 CPU 机架分离安装，CPU 可以对电源模块的状态进行诊断。电源模块上可以安装备份电池，即使电源断电，CPU 中的程序和过



图 2-2　S7-400 系列 PLC 电源模块

程数据也不会丢失。S7-400 系列 PLC 电源模块分为标准和冗余两种类型，一个机架上只能安装一块标准类型电源。如果选择冗余类型电源，那么一个机架上可以安装两块电源模块互为备份。

## 2.2 机架

### 2.2.1 S7-300 系列 PLC 机架

S7-300 系列 PLC 使用 DIN 导轨作为主机架和中央扩展机架，机架（rack）中没有背板总线，较长的导轨可以切割成任意尺寸，以便于安装。CPU 与 I/O 模块通过 U 型背板总线连接器（随 I/O 模块一起供货）连接，所以在 S7-300 系列 PLC 主机架和中央扩展机架中不能带电插拔 I/O 模块，通过现场总线 PROFIBUS-DP 连接的远程扩展 ET200M，可以使用“有源”背板总线（在导轨上插入有源的背板总线）实现模块带电插拔功能。每个 S7-300 系列 PLC 机架，除 CPU 和接口模块以外，最多可以插入 8 个模块，每个模块只占用一个槽号而与模块大小无关。

### 2.2.2 S7-400 系列 PLC 机架

S7-400 系列 PLC 的机架带有背板总线，按槽数可划分为 4 槽机架、9 槽机架（见图 2-3）和 18 槽机架。模块的宽度决定占用机架的槽数，如 10A 电源模块需要占用两个槽。S7-400 系列 PLC 机架按功能划分主要有以下几种类型：



图 2-3 S7-400 系列 PLC 9 槽机架

（1）中央机架

用于中央扩展的主机架，机架带有 K 总线（串行通信总线）和 P 总线（并行 I/O 总线），可以插入 FM（功能模块）和 CP（通信处理器）等需要 K 总线通信的模块。

（2）扩展机架

用于中央扩展，扩展机架与中央机架结合使用，一个主机架通过接口模块可以带有多个扩展机架，扩展机架只带有 P 总线，只能插入输入、输出信号模块，不能插入需要 K 总线通信的 FM 和 CP 模块。

（3）通用机架

通用机架带有 K 总线和 P 总线，既可以作为中央机架也可以作为扩展机架使用，这样可以在扩展机架上插入 FM 和 CP 模块。与中央机架和扩展机架相比，模块在通用机架上安装没有槽位的限制，使用灵活，但价格略高。

（4）特殊机架

有一些机架具有特殊功能，如 UR2-H 18 槽机架，在机架中央处物理分开，适合将 S7-400H冗余系统中两个 CPU 插入同一机架上的紧凑型安装，也可以选择通用型机架，将冗余 CPU 分开；CR2 18 槽中央机架，分为两段（10 + 8），可以使两个 S7-400 系列 PLC 站共用一个机架和电源，适合于两个 PLC 站的紧凑型安装。

S7-400 系列 PLC 机架类型比较多，可以根据现场安装条件灵活地选择。

## 2.3　CPU

### 2.3.1　S7-300/400 系列 PLC CPU 简介

CPU 相当于一个 PLC 站的心脏，所有外部信号扫描、计算和逻辑处理、输出等工作都是通过 CPU 完成的。根据 CPU 的处理能力将 S7 PLC 分为 S7-300、S7-400 系列。

S7-300 系列 PLC CPU 从 CPU312 到 CPU319 有不同的型号，CPU 序号由低到高功能逐渐增强，技术指标主要区别于 CPU 的内存空间、计算速度、通信资源和编程资源，如计数器、定时器的个数等。

S7-400 系列 PLC CPU 从 CPU412 到 CPU417 有不同的型号，与 S7-300 系列 PLC 排序相同，序号越大，功能越强，S7-400 系列 PLC CPU 除内存空间、计算速度、通信资源和编程资源优于 S7-300 系列 PLC CPU 外，还具有更多的程序中断处理能力，如过程中断 OB40～OB47等，对 I/O 映像区更新的方式，S7-400 系列 PLC CPU 可以选择将 I/O 映像区分区，从而优化 CPU 的资源。

按功能划分主要有以下几种类型 CPU：

（1）普通型 CPU

实现计算、逻辑处理、定时、通信等 CPU 的基本功能，如 S7-300 系列 PLC CPU314、S7-400 系列 PLC CPU412-2 等。

（2）集成特殊功能的 CPU

CPU 集成高速计数器、简单定位和脉冲输出等功能，如 S7-300 系列 PLC CPU314C、CPU314IFM 等，CPU 上集成实现特殊功能的输入、输出端口，这些紧凑型、功能全的 CPU 适合于中小型设备，如包装机、印刷机等设备的控制，S7-400 系列 PLC CPU 不具有集成的特殊功能。

（3）故障安全型 F-CPU

CPU 经过 TUV（技术监督学会）组织的安全认证，如 S7-300 系列 PLC CPU315F、S7-400 系列 PLC CPU416F 和 414H/F 等。在发生故障时，确保控制系统切换到安全的模式（典型为停止状态）。在 F-CPU 中，对用户程序编码进行可靠性校验，F（故障安全）系统为一独立的控制系统，除 CPU 具有故障安全功能外，输入、输出模块以及 PROFIBUS 通信都要具有故障安全功能。

（4）具有冗余功能的 CPU

在 S7-400 系列 PLC 中，有两种类型 CPU（414H 和 417H）支持 CPU 冗余功能，主 CPU 故障后自动切换到备份 CPU 上执行后续操作。冗余 CPU 间的数据交换采用西门子公司独有的事件同步，保证 CPU 故障切换后系统过程数据和报警记录的完整性。

从 CPU 的型号中也可以看出 CPU 集成通信接口的个数和类型，如 CPU314C-2PTP，表示 CPU314C 集成两个通信接口，一个 MPI（多点接口，默认配置）和一个 PTP（点对点）串行接口；CPU414-2DP，2 表示 CPU414 集成两个通信接口，一个 MPI/PROFIBUS 接口（编程接口，通过软件可以设置为 PROFIBUS 接口）和一个 PROFIBUS-DP 接口。如果是 CPU414-3DP，需要注意的只是在 CPU 上预留了第三个 PROFIBUS-DP 接口的插槽，

PROFIBUS-DP接口需要另行购买；CPU315-2PN/DP 表示 CPU 集成 PROFIBUS-DP 和 PROFINET 两个接口。

S7-300/400 系列 PLC CPU 采用具有精简指令集专用的处理芯片，从而保证工业控制的快速性和实时性，对于 PLC CPU 的处理能力不能简单地以 CPU 芯片的快慢来决定，还要考虑到底层操作系统等因素，综合考虑应该从 CPU 对位、字节、字、浮点等信号运算的处理时间和 CPU 集成的用于程序执行的内存来比较。

### 2.3.2 S7-300/400 系列 PLC CPU 操作模式

操作模式是描述 CPU 在某一时刻的动作，CPU 主要有以下几种操作模式：

1. 停止（Stop）模式

在停止模式下，CPU 检测所有已经配置的模块或设置为默认地址的模块是否存在，并设置模块到预置的初始模式，停止模式下 CPU 不执行用户程序。

2. 启动（Startup）模式

在启动模式下，主要有三种启动类型，不同的启动类型将影响 CPU 数据区中存储的数据。

（1）暖启动（Warm Restart）

执行系统数据和地址区的初始设定（没有设定保持功能的计数器、定时器和标志位数据区被清零），并从用户程序起始位置开始运行。S7 PLC 在 STEP7 编程软件中的默认配置为暖启动模式，CPU 在上电后自动执行暖启动过程，也可以通过 CPU 前面板开关或使用编程器，在 STEP7 编程软件中将 CPU 设置为停止模式后，手动执行一次暖启动过程。暖启动 CPU 执行组织块 OB100 一次。

（2）冷启动（Cold Restart）

CPU 的冷启动将删除通过程序调用（调用 SFC），在工作存储区中创建的数据块，其他保留数据块中的过程值被预置为初始值，过程值丢失。所有过程映像区数据、计数器、定时器和标志位的数据将清零（设定为保持功能的计数器、定时器和标志位数据区也将被清零）。可以在 STEP7 中设定 S7-400 系列 PLC CPU 重新上电后自动执行冷启动模式，S7-300 系列 PLC 不支持 CPU 上电自动冷启动模式，最新的 S7-300 系列以及 S7-400 系列 PLC，通过编程器在 STEP7 中可以手动执行冷启动操作。冷启动 CPU 执行组织块 OB102 一次。

（3）热启动（Hot Restart）

程序在上次中断处开始执行，过程数据保持，只有 S7-400 系列 PLC 支持热启动模式。可以在 STEP7 中设定 CPU 重新上电后，自动执行暖启动或通过编程器在 STEP7 中手动执行暖启动操作。暖启动 CPU 执行 OB101 一次。

3. 运行（Run）模式

CPU 执行用户程序，更新输入、输出信号，响应中断请求，对故障信息进行处理。

4. 驻留（Hold）模式

在驻留模式下，用户程序被中止，只适合按步调试程序，如利用断点调试用户程序。

### 2.3.3 S7-300/400 系列 PLC CPU 的存储区域

S7-CPU 的存储器划分为装载存储器、工作存储器、系统存储器三部分，存储器的分布

如图 2-4 所示。

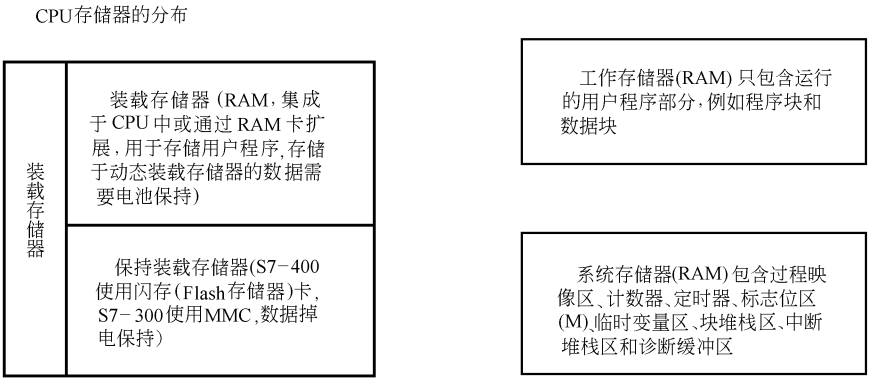



图 2-4　CPU 存储器的分布

1. 装载存储器

装载存储器用于存储用户程序，程序中的符号表及注释只能存储于编程器的硬盘中。装载存储器分为动态装载存储器和保持装载存储器。

（1）动态装载存储器

S7-400 系列 PLC CPU 和早期的 S7-300 系列 PLC CPU（不使用 MMC）中集成动态装载存储器，通过外置的 RAM 卡，可以扩展 S7-400 系列 PLC CPU 的动态装载存储器（在早期的 S7-300 系列 PLC CPU 中集成足够大的动态装载存储器，不需要扩展），新型号的 S7-300 系列 PLC CPU 中没有集成动态装载存储器。在调试 S7-400 系列 PLC 时，用户程序可以直接下传到动态装载存储器中，而不需要额外的操作，比较方便。存储于动态装载存储器中的程序和过程数据需要电池保持。

（2）保持装载存储器

只有 S7-300 系列 PLC CPU312IFM、CPU314IFM 集成保持装载存储器，其他类型的 CPU 都需要外置，S7-400 系列 PLC 和早期的 S7-300 系列 PLC CPU（不使用 MMC）使用 Flash 卡作为保持装载存储器，新的 S7-300 系列 PLC CPU 使用 MMC 作为替代。与 RAM 相比，在调试 S7-400 系列 PLC 时，用户程序不能直接下传到 Flash 卡中，每次修改程序后需要额外的操作，将程序“复制”到卡中，在每次“复制”的过程中。程序完全被覆盖。使用 MMC 的新型号 S7-300 系列 PLC CPU 下传程序时（必须插入 MMC，否则不能下载用户程序），程序可以直接下传到 MMC 中，不需要“复制”过程，比较方便，存储于 Flash 和 MMC 中的程序不需要电池保持。

**注意：**

选择的装载存储器容量应大于 CPU 集成的工作存储器。

2. 工作存储器

工作存储器 RAM 集成在 CPU 中，不能被扩展。工作存储器只包含与运行相关的部分用户程序，用户程序只能在 RAM 和系统存储器中运行。

**注意：**

根据输入、输出的点数及程序的难易程度，选择合适的 CPU，如果程序量超过工作存

储器的空间，只能更换更大存储容量的 CPU。

3. 系统存储器

系统存储器（RAM）集成在 CPU 中，不能被扩展。它包括：

- 标志位、定时器和计数器的地址区
- I/O 的过程映像
- 块堆栈
- 中断堆栈
- 局域数据（一个程序块的本地变量，只有在调用时被激活）

4. 存储器的保持功能

使用动态装载存储器存储用户程序需要电池保持，系统掉电后再上电的用户程序和过程数据将被保持，如果系统在电池故障或没有电池的情况下掉电，则所有存储于动态装载存储器中数据丢失，所以应在系统上电后进行电池的更换。S7-400 系列 PLC CPU 使用 Flash 卡存储用户程序，如果程序有电池保护，系统掉电后过程数据不会丢失；如果没有电池保护，系统掉电再次上电后存储于 Flash 卡中的用户程序被自动复制到动态装载存储器中，过程数据被初始值覆盖。S7-300 系列 PLC CPU 使用 MMC 存储用户程序，不需要电池保护，系统掉电瞬间依靠 CPU 内部电容将过程数据复制到 MMC 中，用户程序和过程数据都不会丢失，所以使用 MMC 的 S7-300 系列 PLC CPU 不再需要备份电池。

5. 用户程序下载到 CPU 存储器

将用户程序下载到 CPU 存储器的过程如图 2-5 所示，完整的用户程序将保留在编程工具的硬盘上，除符号表和程序注释以外，所有的逻辑块和数据块先下载到 CPU 的装载存储器中，为了保证用户程序的快速运行和避免不可扩展的工作存储器不必要的负载，只有与程序执行相关的逻辑块和数据块被放置于工作存储器中运行。

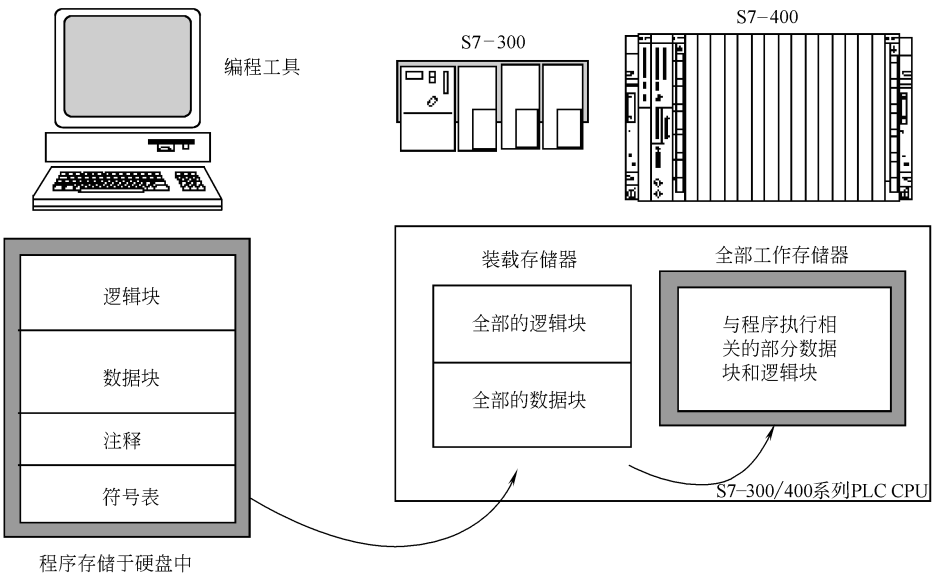



图 2-5　程序下载到 CPU 存储器的过程

**注意：**

通过 RAM 卡、Flash 卡和 MMC 扩展 CPU 的装载存储器，在 CPU 运行过程中不能插拔存储卡，否则容易损坏接口。Flash 卡中装有用户程序，插入 CPU 启动后，用户程序复制到 CPU 中，在后续的操作中，无论 CPU 是否接电都不能将 Flash 卡拔出，否则 CPU 不能启动，这样防止利用 Flash 卡将程序复制到其他的 CPU 中，对程序的安全起到保护作用。

### 2.3.4　S7-CPU 过程映像区的功能

在用户程序中访问输入（I）、输出（Q）地址信息时，通常不直接扫描数字量模块的端口，而是通过位于 CPU 系统存储器的一个存储区域对数字量模块进行访问，这个存储区域就是过程映像区，该过程映像区被分为两部分：过程映像输入区和过程映像输出区。

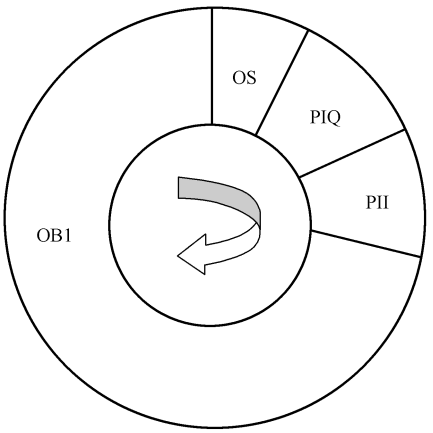


图 2-6　循环程序执行

系统更新过程映像区的过程如图 2-6 所示。

图 2-6 中，OS 为操作系统的一个内部任务，将过程映像输出区（PIQ）的数据输出到数字量输出模块，并将输入模块的信息读到过程映像输入区（PII），过程映像区更新完成后开始执行用户程序的调用，过程映像区被操作系统独立控制，自动更新。

采用过程映像区处理输入、输出信号有两点好处：

1）在 CPU 一个扫描周期中，过程映像区向用户程序提供一个始终一致的过程信号映像，如果在一个扫描周期中，输入模块上的信号状态发生变化，过程映像区中的信号状态保持不变，直到下一个 CPU 扫描周期过程映像区被更新，这样保证 CPU 在执行用户程序过程中数据的一致性。

2）由于过程映像区位于 CPU 的系统存储区中，访问速度比直接访问信号模块显著加快。

S7-300 系列 PLC 的过程映像区是固定的，128 个字节输入和 128 个字节输出；S7-400 系列 PLC 的过程映像区的大小可以在软件中设置。访问过程映像区外的地址，如 S7-300 系列PLC 地址 256，在地址区前加 P。一些信号模块的信号不需要过程映像区的处理，如模拟量模块采样的温度信号，是一个连续变化的数值，在一个程序扫描周期的任意时刻处理都不会影响程序的运算结果，所以模拟量模块默认的地址区分配在过程映像区以外（也可以将地址配置在过程映像区内）。为了减小过程的响应时间，在用户程序中也可以不经过过程映像区直接访问某个 I/O 端口，端口扫描将在指令运行期间执行，由此可获得较快的响应时间，这样只需要在地址区前加 P，例如在程序中可以一次或多次使用 L PIB2 或 T PQW2 指令替代 L IB2 或 T QW2，使用 P 快速读写数字量 I/O 端口也称为立即读、立即写。直接访问 I/O 端口，允许最小的数据类型为字节，如果对一个位信号进行操作，需要使用指令转换。

**注意：**

在用户程序中/过多使用立即读、立即写操作也会影响 CPU 的程序扫描时间。

### 2.3.5 S7-CPU 过程映像区的划分

过程映像区被操作系统自动更新。此外，过程映像区可以最多被划分为 15 个分区（只有 S7-400 系列 PLC CPU 支持此功能），每个过程映像区分区可以在需要的情况下更新，完全独立于操作系统对过程映像区的更新。在信号模块地址分配的过程中，可以选择模块的信号是否在过程映像区中更新（默认设置，系统自动更新）还是需要在过程映像区分区中更新，过程映像区分区的更新需要触发条件，如指定中断响应 OB 或通过程序调用（SFC26/SFC27）更新。过程映像区的划分优化 CPU 的性能。

## 2.4 信号模块

信号模块（Signal Module，SM）是 CPU 与控制设备之间的接口，通过输入模块将输入信号传送到 CPU 进行计算和逻辑处理，然后将逻辑结果和控制命令通过输出模块输出，以达到控制设备的目的。外部的信号主要分为数字量信号和模拟量信号，以阀门的控制为例，如果控制的是电磁阀，阀门只能有开（“1”信号）和关（“0”信号）两种状态，这样的信号是数字量信号，将数字量输出模块的输出点连接到阀门的控制设备（如接触器）上，将阀门的状态反馈信号（可以选择接触器的辅助触点作为反馈信号）接入到数字量输入模块上，这样通过用户程序可以控制阀门的开和关，输出信号为“1”时打开阀门，输出信号为“0”时关闭阀门，同样如果阀门打开，得到的状态反馈信息为“1”，如果阀门关闭，得到的状态反馈信息为“0”，在用户程序中可以监控阀门当前的状态；如果控制对象为可调节阀门，阀门除开和关两种状态外，还可以在中间任意位置上停留，这样就不能通过数字量信号表示模拟量的输入、输出信号控制和表示阀门的开度。例如阀门的控制和反馈信号同为 0～10V 电压信号，若输出 0V 信号，阀门全闭；若输出 10V 信号，阀门全开；若输出 5V 信号时，则阀门开度为 50%，同样通过模拟量输入可以监控阀门的开度，0～10V 与阀门的开度为线性对比关系，通过模拟量控制的设备大部分需要进行 PID 回路调节，例如通过阀门的开度控制蒸汽的流量以达到控制温度的目的，回路的调节为闭环控制。S7-300 系列 PLC 与 S7-400 系列 PLC 信号模块功能相似，书中以 S7-300 系列 PLC 为例介绍了信号模块的特性，有特殊功能的 S7-400 系列 PLC 信号模块将单独列出。

### 2.4.1 数字量输入模块

S7-300 系列 PLC 的数字量输入模块型号以“SM321”开头，3 表示 S7-300 系列，1 表示输入类型；同样 S7-400 系列 PLC 的数字量输入模块型号以“SM421”开头，两者功能相同。以 S7-300 系列 PLC 为例介绍数字量输入模块的类型和功能，模块的类型和技术参数参考表 2-1 和表 2-2。

数字量输入模块的分类：

1. 按外部输入信号电压等级分类

可连接的外部输入信号电压等级分为 DC/AC24～48V、DC48～125V、AC120～230V 等类型，覆盖大部分输入信号的电压等级，如果输入信号电压特殊，则需要使用继电器进行转换后接入到输入模块上。

**表 2-1　S7-300 系列 PLC 数字量输入模块特性（一）**

| 模块 / 特性 | SM321；DI32 × DC24V（－1BL00－） | SM321；DI32 × AC120V（－1EL00－） | SM321；DI16 × DC24V（－1BH02－） | SM321；DI16 × DC24V 高速（－1BH10－） | SM321；DI16 × DC24V 带硬件和诊断中断（－7BH01－） | SM321；DI16 × DC24V 源输入（－1BH50－） |
|---|---|---|---|---|---|---|
| 输入点数 | 32DI，隔离为 2 组 | 32DI，隔离为 4 组 | 16DI，隔离为 1 组 | 16DI，隔离为 1 组 | 16DI，隔离为 1 组 | 16DI，隔离为 1 组 |
| 额定输入电压/V | DC24 | AC120 | DC24 | DC24 | DC24 | DC24 |
| 适用于… | 开关；2/3/4 线接近开关（BERO） | | | | | |
| 支持时钟同步 | × | × | × | √ | √ | × |
| 可编程诊断 | × | × | × | × | √ | × |
| 诊断中断 | × | × | × | × | √ | × |
| 沿触发硬件中断 | × | × | × | × | √ | × |
| 输入延迟可设定 | × | × | × | × | √ | × |
| 备注 | — | — | — | — | 两个短路保护传感器为 8 个通道供电，可用外部冗余电源为传感器供电 | — |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

**表 2-2　S7-300 系列 PLC 数字量输入模块特性（二）**

| 模块 / 特性 | SM321；DI16 × UC 24/48V（－1CH00－） | SM321；DI16 × DC48～125V（－1CH20－） | SM321；DI16 × AC120/230V（－1FH00－） | SM321；DI8 × AC120/230V（－1FF01－） | SM321；DI8 × AC120/230V ISOL（－1FF10－） |
|---|---|---|---|---|---|
| 输入点数 | 16DI，隔离为 16 组 | 16DI，隔离为 2 组 | 16DI，隔离为 4 组 | 8DI，隔离为 4 组 | 8DI，隔离为 8 组 |
| 额定输入电压/V | DC/AC24～48 | DC48～125 | AC120/230 | AC120/230 | AC120/230 |
| 适用于… | 开关；2/3/4 线接近开关（BERO） | | 开关；2/3 线 AC 接近开关 | | |
| 支持时钟同步 | × | × | × | × | × |
| 可编程诊断 | × | × | × | × | × |
| 诊断中断 | × | × | × | × | × |
| 沿触发硬件中断 | × | × | × | × | × |
| 输入延迟可设定 | × | × | × | × | × |
| 备注 | — | — | — | — | — |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

2．按输入点数分类

输入模块的输入点数分为 8、16、32 点，将不同的点数分成组，组和组间电源隔离，如 SM321（－1EL00－）模块，32 个点分成 4 组，模块中每 8 个输入点为一组相互隔离，又如 SM321（－1CH00－）模块，16 个点分成 16 组，那么模块中每个点相互隔离。组隔离的好处在于输入组电源窜入高电压（不能超过隔离电压）烧毁后不会影响到其他组的输入点，整个模块与背板总线也是隔离的，一个模块烧毁不会影响到其他模块（不能超过隔离

电压)。

3. 按输入模块功能分类

有的模块带有中断功能，带有中断功能的模块需要在软件中配置产生中断的事件，如信号从“0”到“1”跳变，中断触发后将停止主程序的运行而执行中断程序，中断程序执行完成后返回到主程序中断点处继续运行。有些输入模块带有断线诊断功能，但是必须在外部输入信号端并联 15～18kΩ 电阻，如图 2-7 所示。

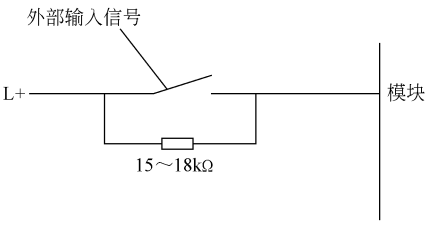



图 2-7 具有断线诊断功能的外部回路

在 STEP7 软件中使能断线检测功能后，模块的输入端将产生小的恒流源形成电流回路，用于检测外部连线的状态。

数字量输入模块的种类比较多，应根据连接外部信号的电压类型、是否需要中断快速响应、诊断功能以及组隔离等要求进行选择。

**注意：**

S7-300/400 系列 PLC 直流输入模块大多为漏型输入（公共端为负）。在 S7-300 系列 PLC 中只有 SM321（－1BH50－）输入模块为源型输入（公共端为正），S7-400 系列 PLC 中则没有源输入模块。选择模块时，注意输入信号与输入模块的匹配性；小的 PLC 系统中，如 S7-200 系列 PLC 可以通过接线的方式满足不同类型的输入信号，在大的项目中不建议使用，因为它容易造成电源的混乱。

## 2.4.2 数字量输出模块

S7-300 系列 PLC 的数字量输出模块型号以“SM322”开头，3 表示 S7-300 系列，2 表示输出类型；同样 S7-400 系列 PLC 的数字量输出模块型号以“SM422”开头，4 表示 S7-400 系列，两者功能相同。以 S7-300 系列 PLC 为例介绍数字量输出模块的类型和功能，模块的类型和技术参数见表 2-3、表 2-4 和表 2-5。

**表 2-3 S7-300 系列 PLC 数字量输出模块特性（一）**

| 模块 / 特性 | SM322；DO32×DC24V/0.5A（－1BL00－） | SM322；DO32×AC120/230V/1A（－1FL00－） | SM322；DO16×DC24V/0.5A（－1BH01－） | SM322；DO16×DC24V/0.5A 高速（－1BH10－） | SM322；DO16×DC24/48VDC（－5GH00－） | SM322；DO16×DC120/230V/1A（－1FH00－） |
|---|---|---|---|---|---|---|
| 输出点数 | 32DO，隔离为 4 组 | 32DO，隔离为 4 组 | 16DO，隔离为 2 组 | 16DO，隔离为 2 组 | 16DO，隔离为 16 组 | 16DO，隔离为 2 组 |
| 输出电流/A | 0.5 | 1 | 0.5 | 0.5 | 1.5 | 0.5 |
| 额定负载电压/V | DC24 | AC120 | DC24 | DC24 | DC/AC 24～48 | AC120/230 |
| 适用于… | 电磁阀、直流触点和指示灯 | | | | | |
| 支持时钟同步 | × | × | × | √ | × | × |
| 可编程诊断 | × | × | × | × | √ | × |
| 诊断中断 | × | × | × | × | √ | × |
| 替换值输出 | × | × | × | × | √ | × |
| 备注 | — | | | | | |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

**表 2-4　S7-300 系列 PLC 数字量输出模块特性（二）**

| 模块 / 特性 | SM322；DO 8 × DC24V/2A（－1BF01－） | SM322；DO 8 × DC24V/0.5A 带诊断中断（－8BF00－） | SM322；DO 8 × DC48～125V/1.5A（－1CF00－） | SM322；DO 8 × AC120/230V/2A（－1FF01－） | SM322；DO 8 × AC120/230V/2A ISOL（－5FF00－） |
|---|---|---|---|---|---|
| 输出点数 | 8DO，隔离为 2 组 | 8DO，隔离为 1 组 | 8DO，隔离并反极性保护，隔离为 2 组 | 8DO，隔离为 2 组 | 8DO，隔离为 8 组 |
| 输出电流/A | 2 | 0.5 | 1.5 | 2 | 2 |
| 额定负载电压/V | DC24 | DC24 | DC48～125 | AC120/230 | AC120/230 |
| 适用于… | 电磁阀、直流触点和指示灯 | | | 交流电磁阀、触点、电动机起动器、电动机和指示灯 | |
| 支持时钟同步 | × | × | × | × | × |
| 可编程诊断 | × | √ | × | × | √ |
| 诊断中断 | × | √ | × | × | √ |
| 替换值输出 | × | √ | × | × | √ |
| 备注 | — | 可以冗余驱动负载（I/O 冗余） | — | 可更换熔丝 | — |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

**表 2-5　S7-300 系列 PLC 数字量继电器输出模块特性**

| 模块 / 特性 | SM322；DO 16 × AC 120V/继电器（－1HH01－） | SM322；DO 8 × AC 230V/继电器（－1HF01－） | SM322；DO 8 × AC 230V/5A 继电器（－5HF00－） | SM322；DO 8 × AC 230V/5A 继电器（－1HF10－） |
|---|---|---|---|---|
| 输出点数 | 16DO，隔离为 2 组 | 8DO，隔离为 4 组 | 8DO，隔离为 8 组 | 8DO，隔离为 8 组 |
| 额定负载电压/V | DC24～120<br>AC48～230 | DC24～120<br>AC48～230 | DC24～120<br>AC48～230 | DC24～120<br>AC24～230 |
| 支持时钟同步 | × | × | × | × |
| 可编程诊断 | × | × | √ | × |
| 诊断中断 | × | × | √ | × |
| 替换值输出 | × | × | √ | × |
| 备注 | — | | | |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

数字量输出模块的分类：

1．按输出信号电压等级分类

根据外部负载电压等级分为 DC24～48V、DC48～125V、AC120～230V 等类型，如果负载控制回路电压特殊，可以通过外接继电器进行转换或使用集成继电器输出模块。继电器输出模块集成继电器，可连接负载电压等级比较宽，但是继电器输出不能具有高的响应频率，使用 24V 晶体管输出，输出切换频率可达 1kHz，每个模块具体的切换频率需要查看模块规范手册。

2．按输出点数分类

输出模块的输出点数分为 8、16、32 点，将不同的点数分成组，组和组间电源隔离，如 SM322（－1BL00－）模块，32 个点分成 4 组，模块中每 8 个输出点为一组相互隔离。又如 SM322（－5GH00－）模块，16 个点分成 16 组，那么模块中每个点相互隔离，组隔离的好

处在于输出组电源窜入高电压（不能超过隔离电压）烧毁后不会影响到其他组的输出点，整个模块与背板总线也是隔离的，一个模块烧毁不会影响到其他模块。

3．按输出模块功能分类

有的模块带有诊断、替换值输出、断线检测和支持 PROFIBUS-DP 等时同步等功能，可以根据控制要求选择合适的输出模块。

**注意：**

输出模块最大的输出电流必须小于负载电流，否则输出模块将被烧毁。输出模块的最大输出电流在模块规范样本中通常以组为单位，如输出模块 SM322（－1BF01－），8 点输出，分为 4 个组，每组两个输出点在 40°C 最大电流输出为 4A（例如：第一个通道输出电流为 2.6A，第二个通道最大输出电流为 1.4A）。

### 2.4.3 数字量输入/输出模块

数字量输入、输出通道集成在同一个模块上，S7-300 系列 PLC 的数字量输出模块型号以“SM323”开头，3 分别表示 S7-300 系列输入、输出类型。S7-400 系列 PLC 中没有数字量输入、输出集成在一起的模块。S7-300 系列 PLC SM323 模块的类型和技术参数见表2-6。

**表 2-6 S7-300 系列 PLC SM323 模块特性**

| 模块 / 特性 | SM323；DI 16/DO 16×DC24V/0.5A（－1BL00－） | SM323；DI 8/DO 8×DC24V/0.5A（－1BH01－） | SM327；DI 8/DX 8×DC24V/0.5A 可设置参数（－1BH00－） |
|---|---|---|---|
| 输入点数 | 16DI，隔离为 1 组 | 8DI，隔离为 1 组 | 8DI 和 8 点可单独配置为 DI 或 DO，隔离为 1 组 |
| 输出点数 | 16DO，隔离为 2 组 | 8DO，隔离为 1 组 | |
| 额定输入电压/V | DC24 | DC24 | DC24 |
| 输出电流/A | 0.5 | 0.5 | 0.5 |
| 额定负载电压/V | DC24 | DC24 | DC24 |
| 输入适用于 | 开关和 2/3/4 线接近开关（BERO） | | |
| 输出适用于 | 电磁阀、DC 接触器和指示灯 | | |
| 支持时钟同步 | × | × | × |
| 可编程诊断 | × | × | × |
| 诊断中断 | × | × | × |
| 沿触发硬件中断 | × | × | × |
| 可调节输入延时 | × | × | × |
| 替换值输出 | × | × | × |
| 备注 | — | | |

注：×表示不支持该项功能；—表示没有。

### 2.4.4 模拟量输入模块

模拟量输入模块将模拟量信号转换为数字信号用于 CPU 的计算，如阀门的开度信号，阀门从关到开输出为 0～10V，通过 A/D（模/数）转换器按线性关系转换为数字信号

0～27648，这样在 CPU 就可以计算出当前阀门的开度，采样的数值可以用于其他计算，也可以发送到人机接口用于阀门的开度显示。

1．模拟量输入模块类型

S7-300 系列 PLC 的模拟量输入模块型号以“SM331”开头，S7-400 系列 PLC 的模拟量输入模块型号以“SM431”开头，两者功能相同，表示方法与数字量模块相同。以 S7-300 系列 PLC 为例介绍模拟量输入模块的类型和功能，类型和技术参数参见表 2-7、表 2-8。

**表 2-7　S7-300 系列 PLC SM331 模块特性（一）**

| 模块 / 特性 | SM331；<br>AI 8×16 位<br>（－7NF00－） | SM331；<br>AI 8×16 位<br>（－7NF10－） | SM331；<br>AI 8×14 位<br>（－7HF0x－） | SM331；<br>AI 8×13 位<br>（－1KF01－） |
|---|---|---|---|---|
| 输入数量 | 8AI<br>4 通道组 | 8AI<br>4 通道组 | 8AI<br>4 通道组 | 8AI<br>8 通道组 |
| 分辨率 | 每个通道组可调：<br>15 位＋符号 | 每个通道组可调：<br>15 位＋符号 | 每个通道组可调：<br>13 位＋符号 | 每个通道组可调：<br>12 位＋符号 |
| 测量方法 | 每个通道组可调：<br>电压<br>电流 | 每个通道组可调：<br>电压<br>电流 | 每个通道组可调：<br>电压<br>电流 | 每个通道组可调：<br>电压<br>电流<br>电阻<br>温度 |
| 测量范围的选择 | 每个通道组任意 | 每个通道组任意 | 每个通道组任意 | 每个通道组任意 |
| 支持时钟同步 | × | × | √ | × |
| 可编程诊断 | √ | √ | √ | √ |
| 诊断中断 | 可设置 | 可设置 | 可设置 | × |
| 极限值监控 | 2 个通道可设置 | 8 个通道可设置 | 2 个通道可设置 | × |
| 由于超过极限造成硬件中断 | 可设置 | 可设置 | 可设置 | × |
| 循环结束时硬件中断 | × | √ | × | × |
| 电位关系 | 光隔离：<br>CPU | 光隔离：<br>CPU | 光隔离：<br>CPU<br>负载电压（不适用于 2－DMU） | 光隔离：<br>CPU |
| 输入之间的允许电位差（ICM）/V | DC50 | DC60 | DC11 | DC2.0 |
| 备注 | — | — | — | |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

模拟量输入模块可以连接不同类型的传感器信号，如电压、电流、电阻等信号，覆盖大多数的应用，如果信号特殊，必须通过变送器进行转换；模块通道的分组，如 8 个输入分 4 个通道，表明 2 个模拟量输入共用一个 A/D 转换器或测量参考点相互隔离；模拟量输入测量方式分为积分和逐次递进两种，前者的采样时间可以设置，分辨率高，但转换时间较长，通常为几十毫秒，后者分辨率低，但转换时间较短，通常为几十微秒。

**表 2-8 S7-300 系列 PLC SM331 模块特性（二）**

| 特点 \ 模块 | SM331；AI 8×12 位（-7KF02-） | SM331；AI 8×RTD（-7PF00-） | SM331；AI 8×TC（-7PF10-） | SM331；AI 2×12 位（-7KB02-） |
|---|---|---|---|---|
| 输入数量 | 8AI<br>4 通道组 | 8AI<br>4 通道组 | 8AI<br>4 通道组 | 2AI<br>1 通道组 |
| 分辨率 | 每个通道可调：<br>9 位 + 符号<br>12 位 + 符号<br>14 位 + 符号 | 每个通道可调：<br>15 位 + 符号 | 每个通道可调：<br>15 位 + 符号 | 每个通道可调：<br>9 位 + 符号<br>12 位 + 符号<br>14 位 + 符号 |
| 测量方法 | 每个通道可调：<br>电压<br>电流<br>电阻<br>温度 | 每个通道可调：<br>电阻<br>温度 | 每个通道可调：<br>温度 | 每个通道可调：<br>电压<br>电流<br>电阻<br>温度 |
| 测量范围的选择 | 每个通道组任意 | 每个通道组任意 | 每个通道组任意 | 每个通道组任意 |
| 可编程诊断 | × | √ | √ | √ |
| 支持时钟同步 | √ | × | × | × |
| 诊断中断 | 可设置 | 可设置 | 可设置 | 可设置 |
| 极限值监控 | 2 个通道可设置 | 8 个通道可设置 | 8 个通道可设置 | 1 个通道可设置 |
| 由于超过极限造成硬件中断 | 可设置 | 可设置 | 可设置 | 可设置 |
| 循环结束时硬件中断 | × | 可设置 | 可设置 | × |
| 电位关系 | 光隔离：<br>CPU<br>负载电压（不适用于 2-DMU） | 光隔离：<br>CPU | 光隔离：<br>CPU | 光隔离：<br>CPU<br>负载电压（不适用于 2-DMU） |
| 输入之间的允许电位差（ICM）/V | DC2.5 | DC75/AC60 | DC75/AC60 | DC2.5 |
| 备注 | — | — | | |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

2．模拟量输入分辨率的表示

CPU 只能以二进制形式处理模拟值。模拟值用一个二进制补码定点数表示，宽度为 16 位，模拟值的符号总是在第 15 位。如果一个模拟量模块的分辨率少于 16 位，则模拟值将左移调整，然后才被保存在模块中。分辨率表示方法参考表 2-9，所有标有“X”的位都置为“0”，例如输入模块的分辨率为 12 位（+符号位），最后 3 位将被置为“0”，那么转换后数值都是以 8 的倍数进行变换。模块的分辨率与模块的误差是不同的概念，每个模块的误差范围需要参考模块规范手册。

**表 2-9 S7-300 系列 PLC SM331 输入模块分辨率表示方法**

| 分辨率［位］（+符号） | 单位 | | 模拟值 | |
|---|---|---|---|---|
| | 十进制 | 十六进制 | 高位字节 | 低位字节 |
| 8 | 128 | $80_H$ | 符号 0000000 | 1×××××× |
| 9 | 64 | $40_H$ | 符号 0000000 | 01×××××× |
| 10 | 32 | $20_H$ | 符号 0000000 | 001××××× |
| 11 | 16 | $10_H$ | 符号 0000000 | 0001×××× |
| 12 | 8 | $8_H$ | 符号 0000000 | 00001××× |
| 13 | 4 | $4_H$ | 符号 0000000 | 000001×× |
| 14 | 2 | $2_H$ | 符号 0000000 | 0000001× |
| 15 | 1 | $1_H$ | 符号 0000000 | 00000001 |

3．模拟量输入模数的对应关系

不同量程范围的模拟量输入信号对应不同的测量值，S7-300 系列 PLC 模块与 S7-400 系列 PLC 模块模数的对应关系相同，以 S7-300 系列 PLC 模拟量输入模块为例，模数的对应关系见表 2-10 ~ 表 2-15。

**表 2-10　电压测量范围为 ±10 ~ ±1V 的模拟值表示方法**

| 系　统 | | 电压测量范围/V | | | | |
|---|---|---|---|---|---|---|
| 十进制 | 十六进制 | ±10 | ±5 | ±2.5 | ±1 | |
| 32767 | 7FFF | 11.851 | 5.926 | 2.963 | 1.185 | 上溢 |
| 32512 | 7F00 | | | | | |
| 32511 | 7EFF | 11.759 | 5.879 | 2.940 | 1.176 | 高于正常 |
| 27649 | 6C01 | | | | | |
| 27648 | 6C00 | 10 | 5 | 2.5 | 1 | 正常范围 |
| 20736 | 5100 | 7.5 | 3.75 | 1.875 | 0.75 | |
| 1 | 1 | $361.7\times10^{-6}$ | $180.8\times10^{-6}$ | $90.4\times10^{-6}$ | $36.17\times10^{-6}$ | |
| 0 | 0 | 0 | 0 | 0 | 0 | |
| -1 | FFFF | | | | | |
| -20736 | AF00 | -7.5 | -3.75 | -1.875 | -0.75 | |
| -27648 | 9400 | -10 | -5 | -2.5 | -1 | |
| -27649 | 93FF | | | | | 低于正常 |
| -32512 | 8100 | -11.759 | -5.879 | -2.940 | -1.176 | |
| -32513 | 80FF | | | | | 下溢 |
| -32768 | 8000 | -11.851 | -5.926 | -2.963 | -1.185 | |

**表 2-11　电压测量范围为 1 ~ 5V 和 0 ~ 10V 的模拟值表示方法**

| 系　统 | | 电压测量范围/V | | |
|---|---|---|---|---|
| 十进制 | 十六进制 | 1 ~ 5 | 0 ~ 10 | |
| 32767 | 7FFF | 5.741 | 11.852 | 上溢 |
| 32512 | 7F00 | | | |
| 32511 | 7EFF | 5.704 | 11.759 | 高于正常 |
| 27649 | 6C01 | | | |
| 27648 | 6C00 | 5 | 10 | 正常范围 |
| 20736 | 5100 | 3.75 | 7.5 | |
| 1 | 1 | $1+144.7\times10^{-6}$ | $0+361.7\times10^{-6}$ | |
| 0 | 0 | 1 | 0 | |
| -1 | FFFF | | 不可能是负值 | 低于正常 |
| -4864 | ED00 | 0.296 | | |
| -4865 | ECFF | | | 下溢 |
| -32768 | 8000 | | | |

**表 2-12 电流测量范围为 0 ~ 20mA 和 4 ~ 20mA 的模拟值表示方法**

| 系统 | | 电流测量范围/mA | | |
|---|---|---|---|---|
| 十进制 | 十六进制 | 0 ~ 20 | 4 ~ 20 | |
| 32767 | 7FFF | 23.70 | 22.96 | 上溢 |
| 32512 | 7F00 | | | |
| 32511 | 7EFF | 23.52 | 22.81 | 高于正常 |
| 27649 | 6C01 | | | |
| 27648 | 6C00 | 20 | 20 | 正常范围 |
| 20736 | 5100 | 15 | 15 | |
| 1 | 1 | $723.4 \times 10^{-6}$ | $4 + 578.7 \times 10^{-6}$ | |
| 0 | 0 | 0 | 4 | |
| -1 | FFFF | | | 低于正常 |
| -4864 | ED00 | -3.52 | 1.185 | |
| -4865 | ECFF | | | 下溢 |
| -32768 | 8000 | | | |

**表 2-13 电流测量范围为 ±20mA、±10mA 和 ±3.2mA 的模拟值表示方法**

| 系统 | | 电流测量范围/mA | | | |
|---|---|---|---|---|---|
| 十进制 | 十六进制 | ±20 | ±10 | ±3.2 | |
| 32767 | 7FFF | 23.70 | 11.85 | 3.79 | 上溢 |
| 32512 | 7F00 | | | | |
| 32511 | 7EFF | 23.52 | 11.76 | 3.76 | 高于正常 |
| 27649 | 6C01 | | | | |
| 27648 | 6C00 | 20 | 10 | 3.2 | 正常范围 |
| 20736 | 5100 | 15 | 7.5 | 2.4 | |
| 1 | 1 | $723.4 \times 10^{-6}$ | $361.7 \times 10^{-6}$ | $115.7 \times 10^{-6}$ | |
| 0 | 0 | 0 | 0 | 0 | |
| -1 | FFFF | | | | |
| -20736 | AF00 | -15 | -7.5 | -2.4 | |
| -27648 | 9400 | -20 | -10 | -3.2 | |
| -27649 | 93FF | | | | 低于正常 |
| -32512 | 8100 | -23.52 | -11.76 | -3.76 | |
| -32513 | 80FF | | | | 下溢 |
| -32768 | 8000 | -23.70 | -11.85 | -3.79 | |

**表 2-14　电阻测量范围为 10kΩ、150Ω、300Ω 和 600Ω 的模拟值表示方法**

| 系　统 | | 电阻测量范围/Ω | | | | |
|---|---|---|---|---|---|---|
| 十进制 | 十六进制 | $10\times10^3$ | 150 | 300 | 600 | |
| 32767 | 7FFF | $11.852\times10^3$ | 177.77 | 355.54 | 711.09 | 上溢 |
| 32512 | 7F00 | | 150.01 | 300.01 | 600.02 | |
| 32511 | 7EFF | $11.759\times10^3$ | 176.38 | 352.77 | 705.53 | 高于正常 |
| 27649 | 6C01 | | | | | |
| 27648 | 6C00 | $10\times10^3$ | 150 | 300 | 600 | 正常范围 |
| 20736 | 5100 | $7.5\times10^3$ | 112.5 | 225 | 450 | |
| 1 | 1 | $361.7\times10^{-3}$ | $5.43\times10^{-3}$ | $10.85\times10^{-3}$ | $21.70\times10^{-3}$ | |
| 0 | 0 | 0 | 0 | 0 | 0 | |
| | | 不可能有负值 | | | | 低于正常 |

电压信号、电流信号及电阻信号的测量值有一个共同的特点，单极性输入信号时，对应的测量范围为 0～27648，双极性输入信号时，对应的测量范围为 -27648～27648，超出测量范围上溢值为 32767、下溢值为 -32768（为了能够表示测量值超限，模拟值用一个二进制补码定点数表示，宽度为 16 位，带有信号位的 16 位分辨率输入信号正常范围为 -27648～27648，而不是 -32768～32767）。

**表 2-15　RTD 温度传感器（Pt100、200、500、1000）的模拟值表示方法**

| Ptx00 标准［°C］（1 个数位 =0.1°C） | 单　位 | | Ptx00 标准［°F］（1 个数位 =0.1°F） | 单　位 | | 范　围 |
|---|---|---|---|---|---|---|
| | 十进制 | 十六进制 | | 十进制 | 十六进制 | |
| >1000.0 | 32767 | $7FFF_H$ | >1832.0 | 32767 | $7FFF_H$ | 上溢 |
| 1000.0<br>:<br>850.1 | 10000<br>:<br>8501 | $2710_H$<br>:<br>$2135_H$ | 1832.0<br>:<br>1562.1 | 18320<br>:<br>15621 | $4790_H$<br>:<br>$3D05_H$ | 高于正常 |
| 850.0<br>:<br>-200.0 | 8500<br>:<br>-2000 | $2134_H$<br>:<br>$F830_H$ | 1562.0<br>:<br>-328.0 | 15620<br>:<br>-3280 | $3D04_H$<br>:<br>$F330_H$ | 正常范围 |
| -200.1<br>:<br>-243.0 | -2001<br>:<br>-2430 | $F82F_H$<br>:<br>$F682_H$ | -328.1<br>:<br>-405.4 | -3281<br>:<br>-4054 | $F32F_H$<br>:<br>$F02A_H$ | 低于正常 |
| <-243.0 | -32768 | $8000_H$ | <-405.4 | -32768 | $8000_H$ | 下溢 |

表 2-15 为热电阻传感器在参数化中选择“Standard”测量方式时的模拟值，1 个数位 = 0.1°C，即测量值为 200 时，表示实际的温度值为 20°C；如果在参数化中选择“Climate”测量方式时，1 个数位 =0.01°C，即测量值为 200 时，表示实际的温度值为 2°C。其他类型的热电阻传感器，如 Nix00 等模拟值的表示方法与实际温度值的表示方法相同。

模拟量输入信号也可以直接连接热电偶采集温度信号，模拟值的表示方法与热电阻（例如 Ptx00）在“Standard”测量方式时的模拟值表示方法相同，1 个数位 =0.1°C，即测

量值为 200 时，表示实际的温度值为 20°C。热电偶与热电阻相比，采样的温度范围更宽，但是需要温度补偿。

4．模拟量输入通道连接不同传感器的设定

通过模拟量模块内部的跳线，同一个模拟量输入模块每个通道组间可以连接不同类型的传感器，设定模块测量方法和测量范围主要使用 STEP7 和量程卡。

配有量程卡的模拟量模块（在供货时已插入模块一侧），如果需要更改量程和连接传感器的类型，必须重新调整量程卡。量程卡可以设定为："A"、"B"、"C" 和 "D" 4 个位置，通过设定量程卡的范围选择连接传感器的类型，"A" 为热电阻、热电偶，测量值通常为毫伏信号；"B" 为电压；"C" 为四线制电流，传感器电源线与信号线分开；"D" 为二线制电流，传感器电源线与信号线共用，传感器电源通过模拟量输入模块供给。

采取以下步骤调整量程卡：

1）使用螺钉旋具，将量程卡从模拟量输入模块中松开，如图 2-8 所示。

2）将量程卡（正确定位（1））插入模拟量输入模块中。所选测量范围为指向模块上标记点（2）的测量范围，如图 2-9 所示。

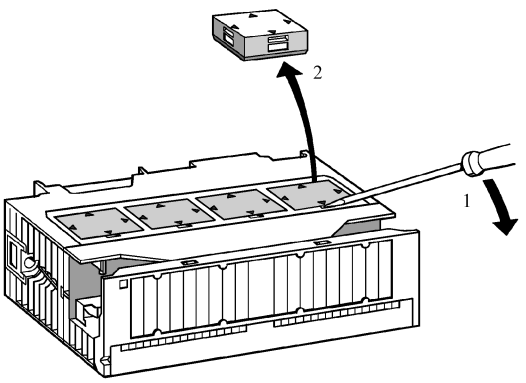



图 2-8　将量程卡从模拟量输入模块中松开

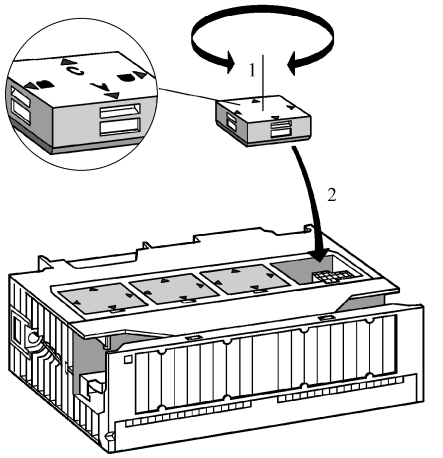



图 2-9　将量程卡插入模拟量输入模块

3）对于其他量程卡采用相同步骤进行。

在 STEP7 中，对模拟量模块进行参数化，选择测量传感器类型必须与模块上量程卡设定的类型相匹配（在 STEP7 中选择传感器类型时，同样提示量程卡应设置的位置），如参数化与实际设定不匹配，模块上的 SF 指示灯指示模块故障。

5．传感器是否隔离与连接方法

根据量程卡的设定，可以将电压、电流、电阻等不同类型的传感器连接到模拟量输入模块。

为了减少电磁干扰，对于模拟信号应使用屏蔽电缆，并且电缆的屏蔽层应该两端接地。如果电缆两端存在电位差，将会在屏蔽层中产生等电位耦合电流，造成模拟信号的干扰。在这种情况下，应该让电缆的屏蔽层一端接地。

对于隔离的模拟量输入模块，在 CPU 的 M 端和测量电路参考点 $M_{ANA}$（一般是模块的 11

号端子）之间没有电气连接。如果参考点 $M_{ANA}$ 和 CPU 的 M 端存在一个电位差 $U_{ISO}$，必须选用隔离模拟输入模块。如果 $U_{ISO}$ 超过允许值（参考每个模块的技术规范手册），必须使用一根等电位连接导线短接 $M_{ANA}$ 端子和 CPU 的 M 端子。

对于不带隔离的模拟量输入模块，在 CPU 的 M 端和测量电路参考点 $M_{ANA}$ 之间，必须建立电气连接。为此，应连接 $M_{ANA}$ 端子与 CPU 或者 IM153 的 M 端子。$M_{ANA}$ 和 CPU 或 IM153 的 M 端子之间的电位差会造成模拟信号的中断。

模拟量输入模块是否隔离可以查看模块技术规范手册或产品手册，绝大多模拟量输入数模块是隔离的，部分模拟量输入、输出混合型的模块是非隔离的，如 SM334。

如果使用的传感器是非隔离传感器，在输入通道的测量线 M－和测量电路参考点 $M_{ANA}$ 之间会发生有限电位差 $U_{CM}$（共模电压）。为了防止超过允许值，在测量点之间必须使用等电位导线连接。

建议的连接图中所用的缩写词和助记符含义如下：

M＋：测量导线（正）；

M－：测量导线（负）；

$M_{ANA}$：模拟测量电路的参考电压；

M：接地端子；

L＋：DC24V 电源端子；

$U_{CM}$：$M_{ANA}$ 与模拟量输入通道之间或模拟量输入通道之间的电位差；

$U_{ISO}$：$M_{ANA}$ 和 CPU 的 M 端子之间的电位差。

传感器信号与模拟量输入模块是否隔离有不同连接方式，可参见图 2-10～图 2-13 的连接示例。

- 连接带隔离的传感器至带隔离的模拟量输入模块

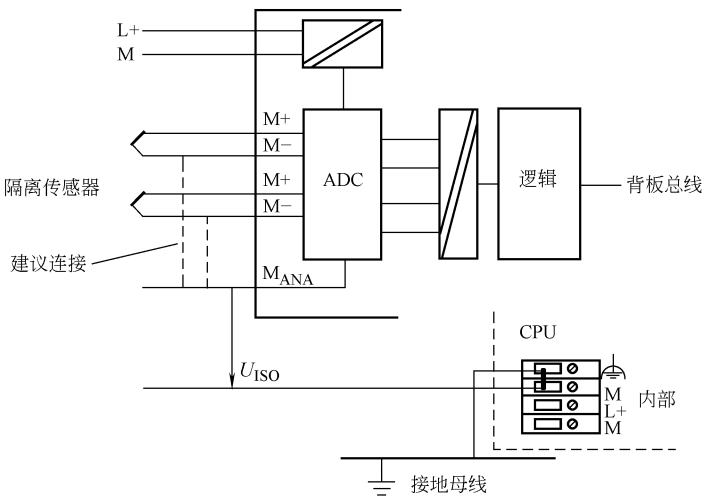



图 2-10　连接带隔离的传感器至带隔离的模拟量输入模块

连接带隔离的传感器与带隔离的模拟量输入模块时，在传感器和模拟量模块端都不需要将信号负端接地，建议信号的负端与输入模块电位参考点（$M_{ANA}$）连接。

- 连接带隔离的传感器至非隔离的模拟量输入模块

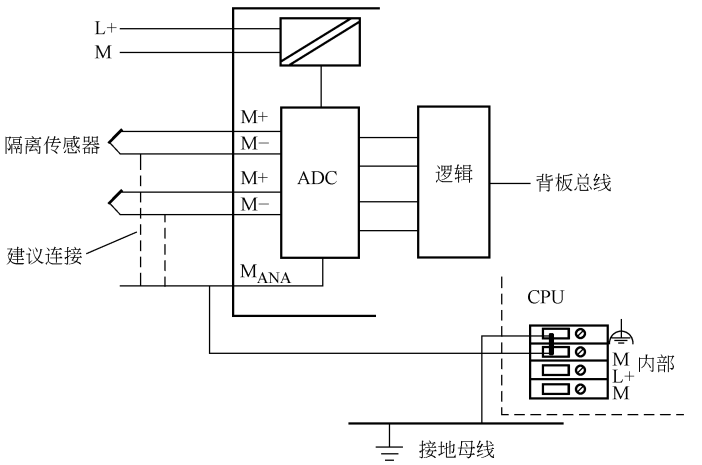



图 2-11 连接带隔离的传感器至非隔离的模拟量输入模块

连接带隔离的传感器与非隔离的模拟量输入模块时，在非隔离的模拟量输入模块一端将模块电位参考点（$M_{ANA}$）与地连接。

- 连接非隔离传感器至带隔离的模拟量输入模块

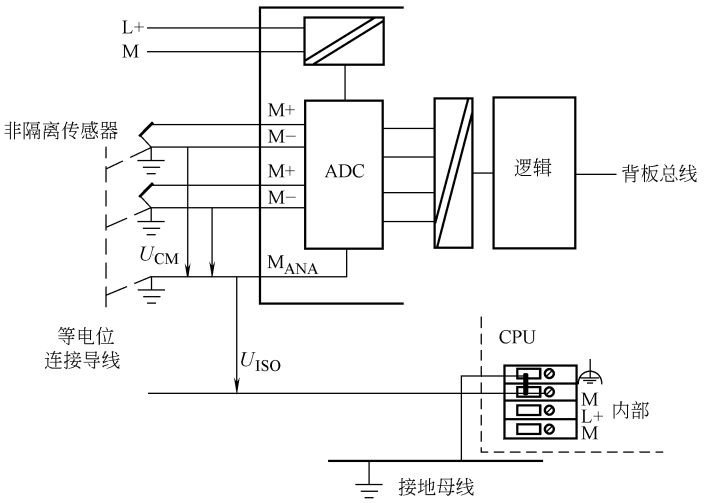



图 2-12 连接非隔离的传感器至带隔离的模拟量输入模块

连接非隔离的传感器与带隔离的模拟量输入模块时，在非隔离的传感器一端将输入信号的负端接地，如果 $U_{CM}$ 超出模块允许的范围（参考模块规范手册），模块的电位参考点（$M_{ANA}$）与传感器的地必须通过等电位导线连接。

- 连接非隔离传感器至非隔离的模拟量输入模块

连接非隔离的传感器与非隔离的模拟量输入模块时，在非隔离的传感器和模拟量输入模块两端分别将输入信号的负端和模块的电位参考点（$M_{ANA}$）接地，此外还需要将 CPU 的地与传感器的地通过等电位导线连接。

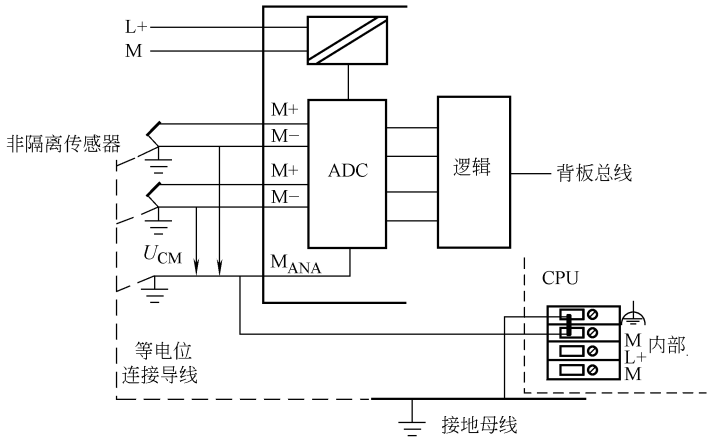



图 2-13　连接非隔离的传感器至非隔离的模拟量输入模块

**注意：**

不能连接非隔离的两线电流传感器和非隔离的电阻传感器至非隔离的模拟量输入模块，通过参考模板规范手册可以判断模拟量输入模块是否隔离。

6. 连接不同类型的传感器

（1）连接电压型传感器（如图2-14所示）

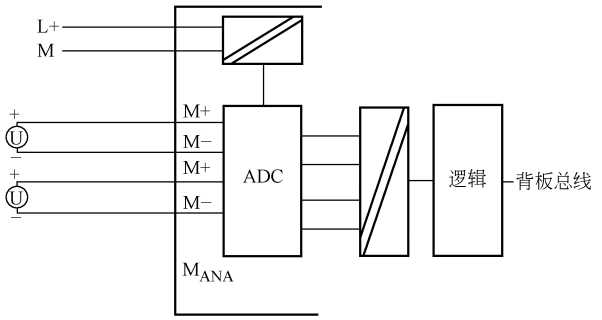



图 2-14　连接电压型传感器至隔离的模拟量输入模块

**注意：**

由于信号的参考电位不同，测量值可能出现超上下限，建议连接信号的负端至模块的电位参考点（$M_{ANA}$）。

（2）连接两线制变送器（如图 2-15 所示）

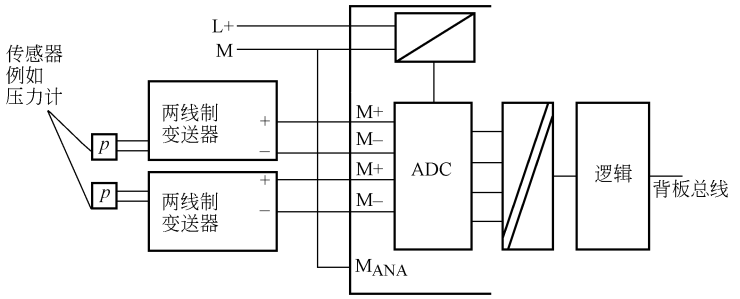



图 2-15　连接两线制变送器至隔离的模拟量输入模块

两线制变送器是指变送器的电源与信号线相同，当模块通道被设置为连接两线制变送器时，模块通过信号线向变送器提供 DC24V 电源，变送器通过信号线输出电流信号。由于两线制变送器电源由模块提供，输出信号只能是 4 ~ 20mA 信号，而不可能是 0 ~ 20mA 信号。

两线制变送器的电源也可以通过外部电源提供，如图 2-16 所示。

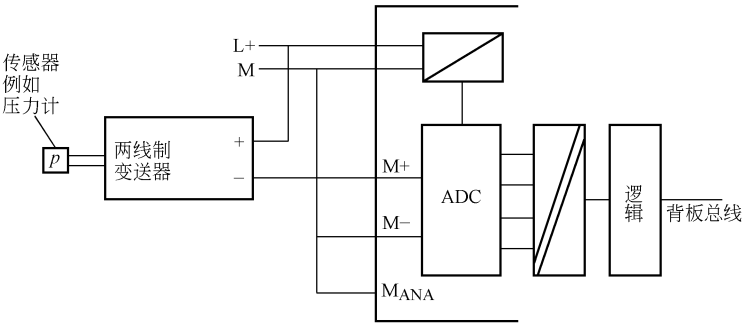



图 2-16　连接从 L + 供电的两线制变送器至隔离的模拟量输入模块

从 L + 提供两线制变送器电源，信号从变送器负端输出到模块的 M + 端，经 M – 流出形成电流回路，这种情况下，模拟量输入模块不提供电源，连接的方式与连接四线制变送器相同，在 STEP7 软件和量程卡的设置中，选择连接四线制变送器方式。

**注意：**

连接两线制变送器时，变送器电源由输入模块提供，在图 2-16 中，实际为四线制变送器连接方式，由于信号的参考电位不同，测量值可能出现超上下限，建议连接电源的负端至模块的电位参考点（$M_{ANA}$）。

（3）连接四线制变送器（如图 2-17 所示）

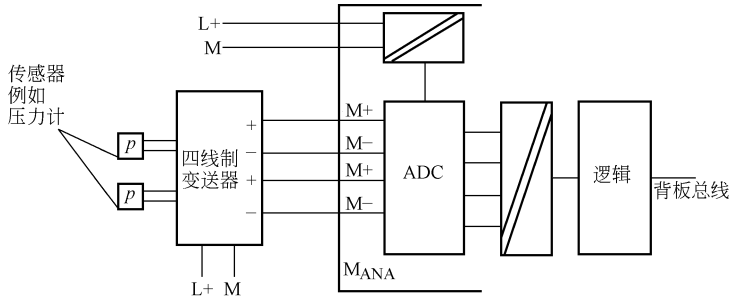



图 2-17　连接四线制变送器至隔离的模拟量输入模块

**注意：**

四线制变送器的电源线与输出的信号线分开，并不是输出四根线连接到输入模块上，与两线制变送器的设置不同，输入模块不向变送器提供电源。如果连接四线制变送器，而在模块上设置连接变送器的类型为两线制，容易将变送器烧坏。由于变送器输出信号的参考电位与输入模块不同，测量值可能出现超上下限，建议连接信号的负端至模块的电位参考点（$M_{ANA}$）。

（4）连接热敏电阻和电阻信号

热敏电阻和普通电阻可以使用两线制、三线制或四线制方法进行接线，通常情况下，连接热敏电阻和电阻信号需要占用两个模拟量输入通道，一个通道通过端子 $I_{C+}$ 和 $I_{C-}$ 向传感器提供恒流源信号，在传感器两端产生电压信号；另一个输入通道作为测量端测量电压信号，这也是测量热敏电阻和电阻信号的原理，有的专用测量热敏电阻和电阻信号的模块，在

每个测量通道都提供 $I_{C+}$ 和 $I_{C-}$ 恒流源输出端子。

考虑到导线电阻对测量阻值的影响，使用四线制接线和三线制接线，可以补偿测量电缆中由于电阻引起的偏差，使测量结果更精确。

热敏电阻和电阻信号四线制接线方式如图 2-18 所示。

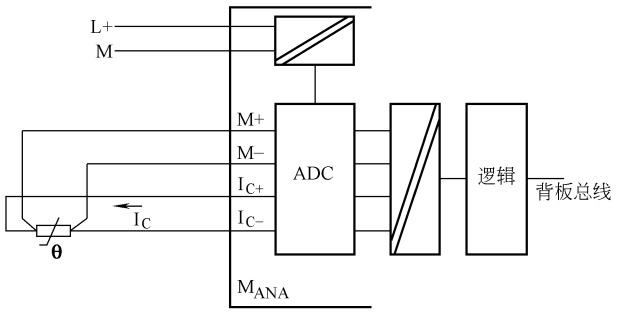



图 2-18　热敏电阻和电阻信号与隔离模拟量输入模块的四线制连接

热敏电阻和电阻信号三线制接线方式如图 2-19 所示，必须短接 M – 和 $I_{C-}$ 端子，应注意使用 SM331，AI 8 × RTD 输入模块时例外，需要短接 M + 和 $I_{C+}$ 端子。

采用热敏电阻和电阻信号两线制接线方式如图 2-20 所示，短接 M + 和 $I_{C+}$ 以及 M – 和 $I_{C-}$ 端子，如果连接输入信号电缆细长，阻值较大，将影响测量精度。

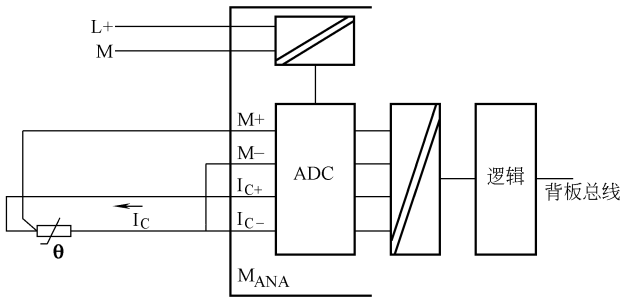



图 2-19　热敏电阻和电阻信号与隔离模拟量输入模块的三线制连接

（5）连接热电耦传感器

热电耦由一对传感器以及所需安装和连接部件组成。热电耦的两根导线可以使用不同金属或金属合金进行焊接。根据所使用材料的成分，可以分为几种热电耦，例如 K 型、J 型和 N 型热电耦。不管其类型如何，所有热电耦的测量原理都是相同的。

1）热电耦的工作原理，如图 2-21 所示。如果测量点的温度与热电耦的自由端（连接点）的温度不同，会在热电耦自由端产生电压——热电动势。所产生的电压大小取决于测量点的温度与自由端的温度差，以及热电耦所使用的材料成分。

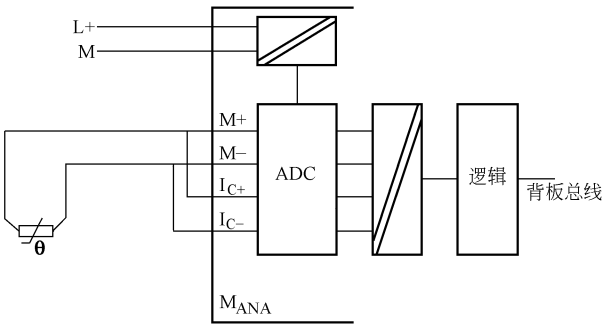



图 2-20　热敏电阻和电阻信号与隔离模拟量输入模块的两线制连接

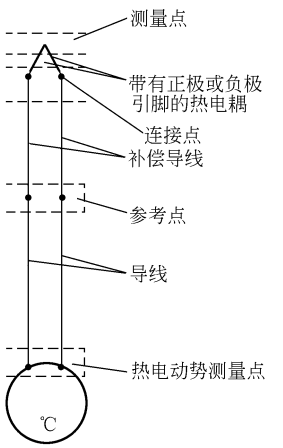



图 2-21　热电耦的结构

由于热电耦测量的总是温度差，必须留有自由端作为参考点的已知温度，以便确定测量结果的温度。

使用补偿导线，热电耦可以从其连接点扩展到参考点。这些补偿导线是由和热电耦导线相同材料制成的。

**注意：**

*应确保这些导线都连接到了正确的极性，否则将会造成明显的测量误差。*

2）参考点的温度补偿。参考点的温度补偿在这里显得特别重要，它在很大程度上会影响热电耦的测量结果。用于采集参考点的温度，以便根据参考点和测量点之间的温度差，获得绝对温度值。根据参考点的位置，可以使用内部补偿或外部补偿方式，补偿参考点处温度的波动造成对测量值的影响。参考点温度补偿方式见表 2-16。

**表 2-16 温度补偿的方式**

| 选　项 | 说　明 |
|---|---|
| 没有补偿 | 如果只想采集测量点和参考点之间的温度差 |
| 内部补偿（参见图 2-22） | 如果采用内部补偿，可以使用模块的内部温度进行比较（热电耦内部比较） |
| 使用热电耦补偿盒进行外部补偿。（参见图 2-23） | 使用补偿盒采集并补偿参考点温度（热电耦外部比较）。不需模块进行进一步处理 |
| 使用热敏电阻进行外部补偿，用于参考点温度的采集。（参见图 2-24），适用于 SM331；AI 8 × TC 和 S7-400 系列 PLC 模块 | 可以使用热敏电阻（铂或镍）采集参考点温度，并由模块进行计算 |
| 使用热敏电阻对具有相同参考点温度的几个热电耦输入模块进行补偿（S7-400 系列 PLC 使用） | 使用热敏电阻测量参考点温度，在 STEP7 程序中调用 SFC55，将参考点温度值分别写到每个模块中。只有在 CPU 运行时设定 |
| 固定参考点温度 | 如果参考点温度是一个已知的常数，可以在 STEP7 中直接定义 |

图 2-22 ~ 图 2-24 所用的缩写词和助记符含义如下：

M +：测量导线（正）；

M −：测量导线（负）；

$I_{C+}$：恒定电流输出的正端；

$I_{C-}$：恒定电流输出的负端；

$COMP_{+}$：补偿端子（正）；

$COMP_{-}$：补偿端子（负）；

$M_{ANA}$：模拟测量电路的参考电压；

M：接地端子；

L +：DC24V 电源端子；

P5V：模块逻辑电源；

$KV_{+}$：冷端（正）；

$KV_{-}$：冷端（负）。

3）使用内部补偿的热电耦的连接。热电耦与模块输入之间可以直接连接，也可以使用补偿导线连接。每一个通道组都可以使用一种类型的热电耦，与其他通道组无关，参见图 2-22 所示。

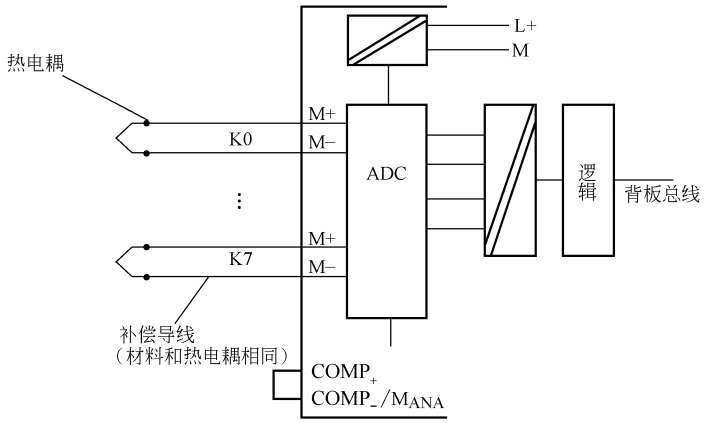



图 2-22　使用内部补偿的热电耦与带隔离的模拟量输入模块之间的连接

内部补偿时，将模块的 $COMP_+$ 端子和 $COMP_-$ 端子短接。

4）热电耦通过补偿盒补偿温度。如果连接到模块输入的所有热电耦都具有相同的参考点，可以使用补偿盒进行温度补偿，连线如图 2-23 所示，将补偿盒侧的输出连接模块的 $COMP_+$ 端子和 $COMP_-$ 端子。补偿盒可以根据现场的要求（连接热电耦的类型）选择不同的订货号。

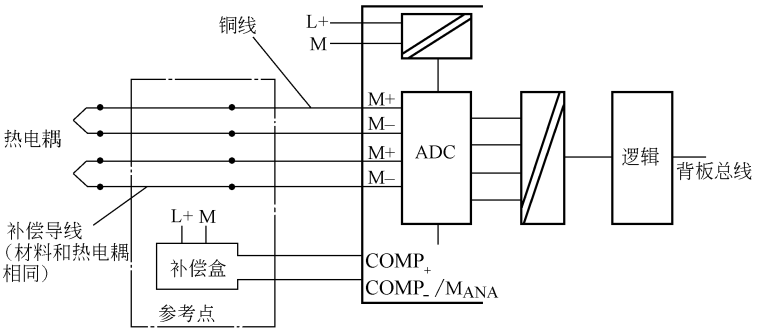



图 2-23　使用补偿盒的热电耦与带隔离的模拟量输入模块之间的连接

5）使用热敏电阻进行温度补偿的热电耦与模块间的连接。用这种补偿类型时，参考点处的温度由电阻传感器决定，温度范围为 –25 ~ 85°C，连线方式如图 2-24 所示。

在 S7-300 系列 PLC 中只有 SM331 AI 8 × TC 模块可以使用热敏电阻进行温度补偿的连接，在 S7-400 系列 PLC 中有许多输入模块（参考模块规范手册）可以使用热敏电阻进行温度补偿，但是在 STEP7 中参数化中，只能选择“0 通道”连接热敏电阻进行温度补偿。

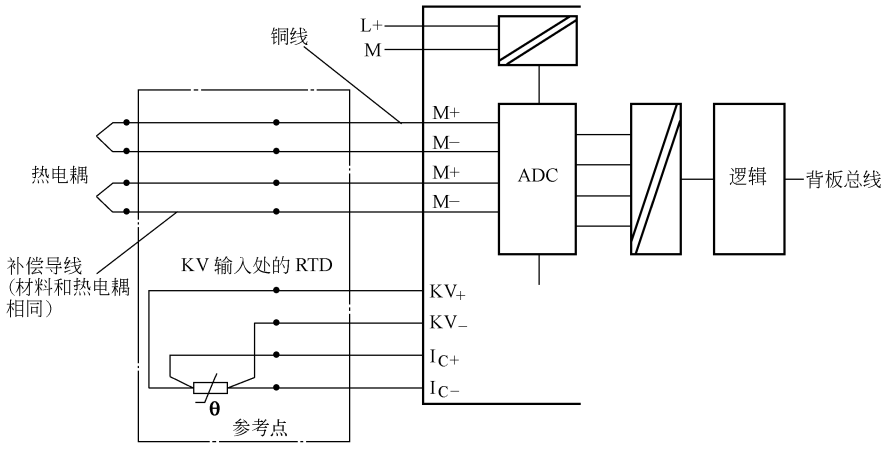



图 2-24 SM331 AI 8 × TC 使用热敏电阻进行温度补偿的同类型热电耦的连接

## 2.4.5 模拟量输出模块

模拟量输出模块将数字量信号转换为模拟量信号，并用于 CPU 输出控制，如控制阀门的开度，0 ~ 10V 为控制阀门从关到开，0 ~ 27648 为通过 D/A（数/模）转换器按线性关系转换为模拟量信号，这样 CPU 输出数值为 13824 时转换为 5V 信号，控制阀门的开度为 50%。模拟量输出模块只有电压和电流信号，以 S7-300 系列 PLC 为例，介绍模拟量输出模块。模块类型见表 2-17。

**表 2-17 模拟量输出模块类型**

| 模块 / 特性 | SM332；AO 8 × 12 位（–5HF00–） | SM332；AO 4 × 16 位（–7ND01–） | SM332；AO 4 × 12 位（–5HD01–） | SM332；AO 2 × 12 位（–5HB01–） |
|---|---|---|---|---|
| 输出数量 | 8AO<br>8 通道组 | 4AO<br>4 通道组 | 4AO<br>4 通道组 | 2AO<br>2 通道组 |
| 分辨率 | 12 位 | 16 位 | 12 位 | 12 位 |
| 输出方式 | 按通道输出：<br>电压<br>电流 | 按通道输出：<br>电压<br>电流 | 按通道输出：<br>电压<br>电流 | 按通道输出：<br>电压<br>电流 |
| 支持时钟同步 | × | √ | × | × |
| 可编程诊断 | √ | √ | √ | √ |
| 诊断中断 | 可设置 | 可设置 | 可设置 | 可设置 |
| 替代值输出 | 不可设置 | 可设置 | 可设置 | 可设置 |
| 电位关系 | 光隔离：<br>CPU<br>负载电压 | 光隔离：<br>CPU 和通道之间<br>通道之间<br>输出和 L + 、M 之间<br>CPU 和 L + 、M 之间 | 光隔离：<br>CPU<br>负载电压 | 光隔离：<br>CPU<br>负载电压 |
| 备注 | — | — | — | — |

注：×表示不支持该项功能；√表示支持该项功能；—表示没有。

1．模拟量输出数/模的对应关系

与模拟量输入信号相反，模拟量输出将数字量信号转换为模拟量信号，数/模对应关系见表 2-18、表 2-19。

**表 2-18　模拟量输出模块数/模对应关系（一）**

| 系　统 | | | 电压输出范围/V | | | |
|---|---|---|---|---|---|---|
| | 十进制 | 十六进制 | ±10 | 0～10 | 1～5 | |
| 118.5149% | 32767 | 7FFF | 0.00 | 0.00 | 0.00 | 上溢、断路和断电 |
| | 32512 | 7F00 | | | | |
| 117.589% | 32511 | 7EFF | 11.76 | 11.76 | 5.70 | 高于正常 |
| | 27649 | 6C01 | | | | |
| 100% | 27648 | 6C00 | 10 | 10 | 5 | 正常范围 |
| 75% | 20736 | 5100 | 7.5 | 7.5 | 3.75 | |
| 0.003617% | 1 | 1 | $361.7\times10^{-6}$ | $361.7\times10^{-6}$ | $1V+144.77\times10^{-6}$ | |
| 0% | 0 | 0 | 0 | 0 | 1 | |
| | -1 | FFFF | $-361.7\times10^{-6}$ | 不可能低于0 | 低于正常 | |
| -25% | -6912 | E500 | | | 0V | |
| | -6913 | E4FF | | | 不可能低于0 | |
| -75% | -20736 | AF00 | -7.5 | | | |
| -100% | -27648 | 9400 | -10 | | | |
| | -27649 | 93FF | | | | 低于正常 |
| -117.593% | -32512 | 8100 | -11.76 | | | |
| | -32513 | 80FF | | | | 下溢、断路和断电 |
| -118.519% | -32768 | 8000 | 0.00 | 0.00 | 0.00 | |

**表 2-19　模拟量输出模块数/模对应关系（二）**

| 系　统 | | | 电流输出范围/mA | | |
|---|---|---|---|---|---|
| | 十进制 | 十六进制 | 0～20 | 4～20 | |
| 118.5149% | 32767 | 7FFF | 0.00 | 0.00 | 上溢、断路和断电 |
| | 32512 | 7F00 | | | |
| 117.589% | 32511 | 7EFF | 23.52 | 22.81 | 高于正常 |
| | 27649 | 6C01 | | | |
| 100% | 27648 | 6C00 | 20 | 20 | 正常范围 |
| 75% | 20736 | 5100 | | | |
| 0.003617% | 1 | 1 | $723.4\times10^{-6}$ | $4+578.7\times10^{-6}$ | |
| 0% | 0 | 0 | 0 | 4 | |
| | -1 | FFFF | 不可能低于0 | | 低于正常 |
| -25% | -6912 | E500 | | 0 | |
| | -6913 | E4FF | | 不可能低于0 | 不可能低于0 |
| -100% | -27648 | 9400 | | | |
| | -27649 | 93FF | | | |
| -117.593% | -32512 | 8100 | | | |
| | -32513 | 80FF | | | 下溢、断路和断电 |
| -118.519% | -32768 | 8000 | 0.00 | 0.00 | |

2．模拟量输出分辨率与设定连接不同类型的负载

模拟量输出模块与输入模块的分辨率表示方法相同。连接不同的负载不需要在硬件上设定量程卡，通过编程软件 STEP7 选择输出的类型，在实际接线时电压信号与电流信号使用不同的端子（通常共用公共端）。

3．连接不同类型的负载

模拟量输出模块连接不同负载时，接线图中出现的缩写和助记符含义如下：

$Q_V$：模拟量输出电压；

$Q_I$：模拟量输出电流；

S +：探测器导线端子（正）；

S －：探测器导线端子（负）；

$M_{ANA}$：模拟电路的参考电压；

$R_L$：负载阻抗；

L +：DC24V 电源端子；

M：接地端子；

$U_{ISO}$：$M_{ANA}$和 CPU 的 M 端子之间的电位差。

（1）隔离模块电压输出与负载的四线制连接

连接电压信号至负载时，连接电缆过长、线径细将导致输出到负载的电压信号产生压降，影响控制精度，采用四线制接线方式可以保证输出的精度，如图 2-25 所示，通过探测器导线端子（S－和 S＋）测量负载两端的压降，对输出电压进行修正补偿，探测器导线端子（S－和 S＋）必须直接连接负载两端，否则影响测量精度。

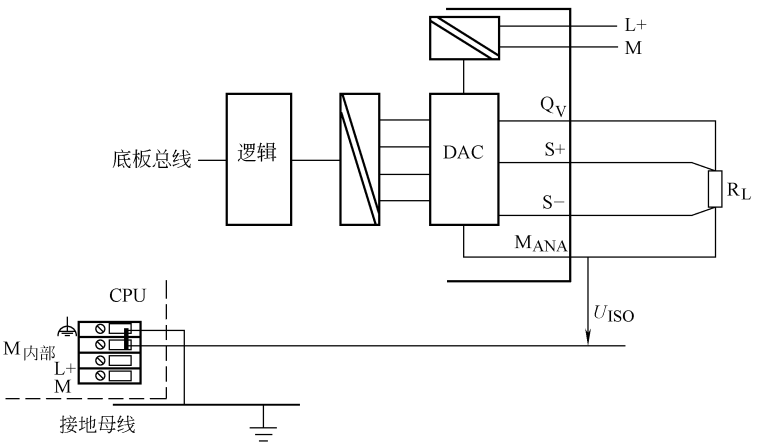



图 2-25　负载与隔离模拟量电压输出之间的四线制连接

干扰或压降会在传感器的接线端子 S－和模拟量输出模块的 $M_{ANA}$之间造成电位差。该电位差不能超过允许值，如果超过允许电位差，将影响模拟信号的精度。

（2）隔离模块电压输出与负载的两线制连接

如果隔离模块与负载间的距离比较近或对输出精度要求不高的情况下，可以采用两线制

方式连接电压型负载，S + 和 S – 端子可以不连线，隔离模块电压输出端子 $Q_V$ 和 $M_{ANA}$ 直接连接到负载，如图 2-26 所示。

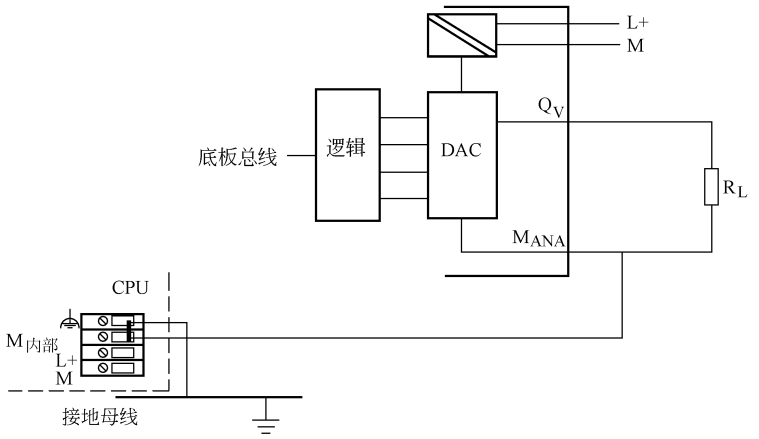



图 2-26　负载与隔离模拟量电压输出之间的两线制连接

（3）隔离模块电流输出与负载的两线制连接

将隔离模块电流输出端子 $Q_I$ 和 $M_{ANA}$ 直接连接到负载，如图 2-27 所示。

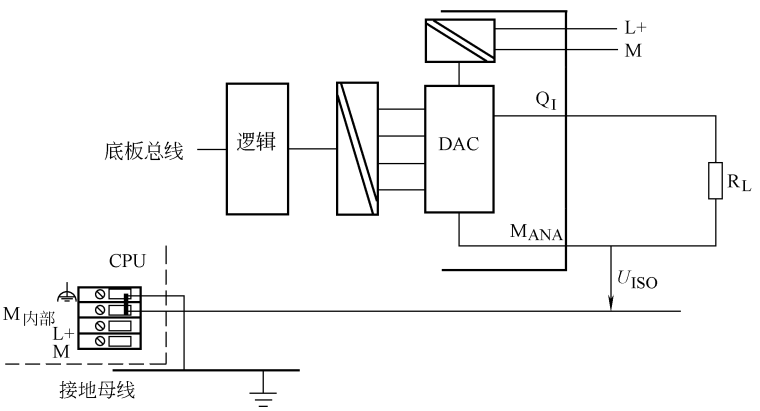



图 2-27　负载与隔离模拟量电流输出之间的两线制连接

**注意：**

使用非隔离的模拟量输出模块，需要将 $M_{ANA}$ 与 CPU 的 M 端子短接。

## 2.4.6　模拟量输入/输出模块

模拟量输入、输出集成在同一模块上，适合紧凑型安装的应用，模拟量输入/输出模块的类型见表 2-20，在 S7-400 系列 PLC 中没有模拟量输入/输出集成的模块。

表 2-20 模拟量输入/输出模块特性

| 模块 / 特点 | SM334；AI 4/AO 2×8/8 位 （-0CE01-） | SM334；AI 4/AO 2×12 位 （-0KE00-） |
|---|---|---|
| 输入数量 | 4AI<br>1 通道组 | 4AI<br>2 通道组 |
| 输出数量 | 2AO<br>1 通道组 | 2AO<br>1 通道组 |
| 分辨率 | 8 位 | 12 位 + 符号 |
| 测量方法 | 每个通道可设置<br>电压<br>电流 | 每个通道可设置<br>电压<br>电阻<br>温度 |
| 输出方式 | 按通道输出：<br>电压<br>电流 | 按通道输出：<br>电压 |
| 可编程诊断 | × | × |
| 诊断中断 | × | × |
| 极限值监控 | × | × |
| 由于超过极限造成硬件中断 | × | × |
| 循环结束时，硬件中断 | × | × |
| 替代值输出 | × | × |
| 电位关系 | 与 CPU 非隔离<br>与负载电压光隔离 | 与 CPU 隔离<br>与负载电压光隔离 |
| 备注 | 不能参数化，测量设置和输出类型与布线方式有关 | — |

注：×表示不支持该项功能；—表示没有。

**注意：**

有些模拟量输入、输出集成的模块没有隔离，连接传感器时，需要注意非隔离连接的限制。

## 2.4.7 特殊模块

在信号模块范围中还有一些特殊应用的模块，如仿真模块、占位模块、测量 SSI 绝对编码器模块等，特殊模块见表 2-21。

表 2-21 特殊信号模块

| 模块 / 特性 | 仿真模块 SM374<br>输入/输出 16 | 占位模块<br>DM370 | 位置编码器模块<br>SM338；POS-INPUT |
|---|---|---|---|
| 输入/输出点数 | 最多 16 个输入或输出 | 为没有参数化的模块保留一个插槽 | 3 个输入连接绝对值编码器（SSI）<br>2 个数字量输入用于保留编码值 |
| 适用于 | 仿真：<br>16 点输入（小开关）<br>16 点输出<br>8 点输入和输出 | 占位用于：<br>接口模块<br>不可编程的信号模块<br>占用两个插槽的模块<br>安全模块、本安型模式与普通模块间的隔离 | 最多检测 3 个绝对值编码器（SSI）<br>编码器类型：带 13 位、21 位或 25 位报文帧长度的编码器（SSI）数据格式：格雷码或二进制码 |
| 支持时钟同步 | × | × | √ |
| 可编程诊断 | × | × | × |
| 诊断中断 | × | × | 可设置 |
| 备注 | 可通过调整拨位开关设定功能 | 当用其他模块更换 DM370 时，其机械结构和地址不变 | 超过 64μs 的绝对值编码器不能用在 SM338 上 |

注：×表示不支持该项功能；√表示支持该项功能。

在 S7-400 系列 PLC 中只有一款仿真模块，必须安装于中央机架上，可以同时模拟数字量信号和模拟量信号，S7-400 系列 PLC 通过远程 I/O 的连接方式，也可以使用上述特殊模块。

## 2.5　通信模块

S7-300/400 系列 PLC 通过通信模块可以使多个相对独立的站点连成网络并建立通信关系。在网络架构中从上到下划分为三层：

1）管理层　管理层包括 ERP（企业资源规划，Enterprise Resource Planning）、MES（制造执行系统，Manufacturing Execution System）和 HMI（人机界面，数据监控）。在 ERP、MES 与 HMI 的数据交换中采用以太网通信。

2）控制层　主要包括各个 PLC 站和 HMI 等主站，HMI 与控制层 PLC 及 PLC 站点间的通信可以根据数据量的大小、速率、实时性的要求，选择工业以太网、PROFINET CBA 和 PROFIBUS/MPI。

3）现场层　保证现场信号采集的实时性，通常情况下选择 PROFINET-IO（基于工业以太网的实时以太网）、PROFIBUS-DP 和 ASI 网络，在实际的网络应用中，由于连接控制设备的多样性和安装调试的方便性，大部分用户选择使用 PROFIBUS-DP 网络连接现场信号，在一些不易安装 PLC 模块和防护等级高的场合，通常选择 ASI 网络。

不同的通信协议和服务需要不同的 PLC 通信模块支持，通信模块的类型及功能见表 2-22。

**表 2-22　通信模块类型及功能**

| 支持的通信网络 / 模块类型 | | PROFINET/工业以太网 | PROFIBUS | MPI | ASI |
|---|---|---|---|---|---|
| S7-300 系列 PLC | S7-300 系列 PLC 集成的通信接口 | S7-300 系列 PLC CPU31X-PN 带有 PROFINET 接口，支持 PROFINET-IO/CBA 及 S7 通信服务 | S7-300 系列 PLC CPU31X-DP 带有 PROFIBUS 接口，支持 DP 主/从、S7 通信服务 | 每个 CPU 都带有 MPI（编程接口） | 没有集成 ASI |
| | CP343－1（6GK7343-1EX20-0XE0） | 支持 PROFINET-IO/CBA、S7、S/R 通信服务 | 不支持 | 不支持 | 不支持 |
| | CP342-5 | 不支持 | 支持 PROFIBUS-DP、FDL、S7 通信服务 | 不支持 | 不支持 |
| | CP343-5 | 不支持 | 支持 PROFIBUS-FMS、FDL、S7 通信服务 | 不支持 | 不支持 |
| | CP343-2 | 不支持 | 不支持 | 不支持 | ASI 主站 |

（续）

| 支持的通信网络 / 模块类型 | | PROFINET/工业以太网 | PROFIBUS | MPI | ASI |
|---|---|---|---|---|---|
| S7-400 系列 PLC | S7-400 系列 PLC 集成的通信接口 | S7-400 系列 PLC CPU41X-PN 带有 PROFINET 接口，支持 PROFINET-IO/CBA 及 S7 通信服务 | S7-400 系列 PLC CPU41X-DP 带有 PROFIBUS 接口，支持 DP 主/从、S7 通信服务 | 每个 CPU 都带有 MPI（编程接口） | 没有集成 ASI |
| | CP443-1 | 支持 S7、S/R 通信服务 | 不支持 | 不支持 | 不支持 |
| | CP443-1ADV | 支持 PROFINET-IO/CBA、S7、S/R 通信服务，带有 4 个网络交换机接口 | 不支持 | 不支持 | 不支持 |
| | CP443-5BASIC | 不支持 | 支持 PROFIBUS-FMS、FDL、S7 通信服务 | 不支持 | 不支持 |
| | CP443-5EXTEND | 不支持 | 支持 PROFIBUS-DP、FDL、S7 通信服务 | 不支持 | 不支持 |

**注意：**

S7-400 系列 PLC 没有 ASI，只能通过 DP/ASI 网关作为 ASI 的主站。在一个 STEP7 项目中，可以有不同的网络，在网络上连接不同的、具有编程功能的站点，使用编程设备（如笔记本电脑）接入网络任意一点都可以对整个网络中已配置的站点进行路由编程，使用路由功能时，必须所有的站点配置在一个 STEP7 项目中。

## 2.6 功能模块

功能模块（FM）通常实现单一、特殊的功能，这些特殊功能 CPU 往往不能实现，例如使用 CPU 内部的计数器，计数的最高频率受到 CPU 扫描周期和输入信号的转换时间限制，假设 CPU 的扫描周期为 50ms，那么 CPU 有可能捕捉不到低于 50ms 的信号变化，信号将丢失，因此 CPU 内部计数器最高计数频率为 100Hz。在有些应用中，使用脉冲编码器测量速度值和位置值时，就不能使用 CPU 中集成的计数功能，需要插入高速计数器模块实现计数功能。功能模块集成处理器，可以独立处理与应用相关的功能。功能模块实现的特殊功能见表 2-23。

**表 2-23 功能模块特性**

| 模块类型 | | 实现的功能描述 |
|---|---|---|
| S7-300 系列 PLC | FM350-1 | 高速计数器模块，1 个计数通道，可以连接 5V/24V 编码器，5V 编码器最高计数频率为 500kHz，24V 编码器最高计数频率为 200kHz |
| | FM350-2 | 高速计数器模块，8 个计数通道，可以连接 24V 编码器，最高计数频率为 20kHz |
| | FM351 | 单轴定位模块，适合使用数据量输出控制电动机起停，定位的速度必须预置，如控制双速电动机时，在模块中预置高低速，位置控制为闭环 |

（续）

| 模块类型 | | 实现的功能描述 |
|---|---|---|
| S7-300 系列 PLC | FM352 | 电子凸轮控制器，根据设定的凸轮轨迹控制模块集成的输出（快速响应）或 PLC 控制的输出点（慢速响应） |
| | FM352-5 | 高速布尔处理器，使用门阵列控制非常快的过程，在模块中只能进行简单的运算（不能使用浮点值），运算一个周期固定为 1μm |
| | FM353 | 单轴步进电动机定位控制模块，位置控制为开环。将位置控制信号以脉冲方式输出到步进电动机控制器，从而实现控制步进电动机 |
| | FM354 | 单轴伺服电动机定位控制模块，位置控制为闭环。将位置控制信号以模拟量电压形式输出到伺服电动机控制器，从而实现位置控制。通过编码器采集位置反馈信号 |
| | FM355 | PID 控制器，集成标准算法和温度控制算法。在 CPU 处于停止模式下可以进行后援操作 |
| | FM355-2 | 温度 PID 控制器，集成的算法适合不同的温度控制 |
| | FM356 | M7 模块，实现快速运算，已淘汰 |
| | FM357-2 | 四轴定位模块，可以控制步进电动机、伺服电动机，具有四轴插补功能，实现路径控制 |
| S7-400 系列 PLC | FM450-1 | 与 FM350-1 功能相同，具有两个通道 |
| | FM451 | 与 FM351 功能相同，具有 3 个通道 |
| | FM452 | 与 FM352 功能相同，具有更多的高速输出 |
| | FM453 | 集成 FM353 和 FM354 功能，三轴定位模块，但是不能进行插补定位 |
| | FM455 | 与 FM355 功能相同，具有更多的控制通道 |
| | FM456 | 与 FM356 功能相同，已淘汰 |
| | FM458 | 属于 D7 系统，实现非常快速的运算功能，CPU 的执行周期可以达到 1ms。可以进行复杂的运算，集成的定位函数也可以用于速度和定位控制 |

**注意：**

每个 CPU 带有功能模块的个数有限制，可以参考 S7CPU 的选型手册。

## 2.7　接口模块

接口模块用于 CPU 机架的扩展，如果在中央机架上安装一个接口模块作为发送器，则在扩展机架上就必须安装一个接口模块作为接收器，发送器和接收器的使用必须匹配。不同类型的接口模块决定扩展机架的个数、最大扩展距离以及扩展机架上安装模块的限制。

### 2.7.1　S7-300 系列 PLC 的接口模块

S7-300 系列 PLC 的接口模块有两种，IM360/IM361 和 IM365，接口模块特性见表 2-24。

一个 IM360 最多可以连接 3 个 IM361，IM360 与 IM361、IM361 与 IM361 间可以传送 P（I/O）总线和 K（通信）总线，所以在扩展机架上模块安装没有限制，接口模块间不能传送电源，每个接口模块需要单独供电。IM365 为一对接口模块，最多扩展一个机架，IM365 接口模块间只能传送 P（I/O）总线，不能传送 K（通信）总线，所以在扩展机架上只能安装信号模块而不能安装需要 K 总线的模块，如 FM 和 CP 模块，接口模块间传送电源，在扩展机架上 IM365 不需要单独供电。

表 2-24 S7-300 系列 PLC 接口模块的特性

| 模块 / 特点 | 接口模块 IM360 | 接口模块 IM361 | 接口模块 IM365 |
|---|---|---|---|
| 适合安装的机架号 | 0（中央机架） | 1～3（扩展机架） | 0 和 1 |
| 数据传送 | 通过连接电缆 386，数据从 IM360 传送到 IM361 | 通过连接电缆 386，数据从 IM360 传送到 IM361 或数据从 IM361 传送到 IM361 | 通过连接电缆 386，数据从 IM365 传送到 IM365 |
| 扩展距离/m | 最长 10 | 最长 10 | 固定长度 1 |
| 特殊标注 | — | — | • 预装配一对 IM365 模块（包括发送器和接收器）<br>• 在机架 1（扩展机架）只能安装信号模块<br>• IM365 接口模块不具有通信总线功能 |

## 2.7.2 S7-400 系列 PLC 的接口模块

与 S7-300 系列 PLC 的接口模块相比，S7-400 系列 PLC 的接口模块具有类型多、扩展功能强、扩展距离远的特点，S7-400 系列 PLC 接口模块特性见表 2-25。

表 2-25 S7-400 系列 PLC 接口模块的特性

| | 短距离扩展 | | 长距离扩展 | |
|---|---|---|---|---|
| IM 发送器 | IM460-0 | IM460-1 | IM460-3 | IM460-4 |
| IM 接收器 | IM461-0 | IM461-1 | IM461-3 | IM461-4 |
| 每个扩展链最大的扩展数 | 4 | 1 | 4 | 4 |
| 最长扩展距离/m | 5 | 1.5 | 102.25 | 605 |
| 5V 传送 | 不可以 | 可以 | 不可以 | 不可以 |
| 每个接口最大的传送电流/A | — | 5 | — | — |
| 是否可以传送通信总线 | 可以 | 不可以 | 可以 | 不可以 |

总共有 4 对接口模块可以选择，IM460-0/IM461-0 带有通信总线，但是没有电源传送，在每个扩展机架上需要安装电源模块，扩展机架上没有模块安装的限制；IM460-1/IM461-1 不带有通信总线，但是具有电源传送功能，在扩展模块上不需要安装电源模块，在扩展机架上只能安装信号模块而不能安装需要 K 总线的模块，如 FM 和 CP 模块；IM460-3/IM461-3 带有通信总线，但是没有电源传送，在每个扩展机架上需要安装电源模块，扩展机架上没有模块安装的限制；IM460-4/IM461-4 不带有通信总线和电源传送，在每个扩展机架上需要安装电源模块，在扩展机架上只能安装信号模块而不能安装需要 K 总线的模块，如 FM 和 CP 模块。

**注意：**

与 S7-300 系列 PLC 的扩展相比，有的 S7-400 系列 PLC 的扩展需要在扩展链的终端添加终端电阻，如果没有终端电阻，CPU 不能识别扩展机架。终端电阻的类型参考 S7-400 系列 PLC 样本手册。

# 第 3 章　西门子 S7-300/400 系列 PLC 系统扩展

## 3.1　S7-300 系列 PLC 的中央扩展

S7-300 系列 PLC 的一个机架上最多可以安装 8 个模块，如果使用更多的模块，则需要通过接口模块（IM36X）扩展机架。使用接口模块 IM365 只能扩展一个机架，如图 3-1 所示。

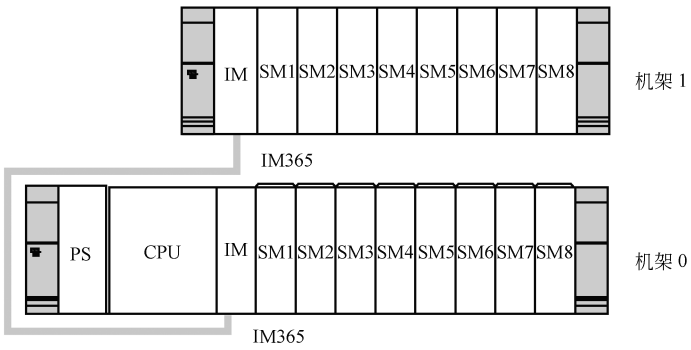



图 3-1　使用 IM365 的中央扩展

通过扩展电缆，IM365 接口模块间只能传送 P（I/O）总线，在机架 1 上只能安装信号模块，而不能安装通信模块（CP）和功能模块（FM），接口模块间传送电源，扩展机架 1 上的 IM365 不需要单独供电，两个机架中所有模块消耗背板总线电流总数不能超过 CPU 的输出电流。

如果需要扩展多个机架，则需要使用 IM360/IM361 进行扩展，如图 3-2 所示。

一个 IM360 最多可以连接 3 个 IM361，接口模块间不传送电源，每个接口模块需要 24V 电源供电，扩展的接口模块向背板总线输出电流，每个机架中所有模块消耗背板总线的电流不能超过 CPU 或接口模块的输出电流。使用 IM360/IM361 与使用 IM365 进行中央扩展相比，每个机架上安装的模块没有限制，向背板总线输出的功率大，驱动能力强，扩展最长可以达到 10m，完全覆盖 IM365 的扩展功能，价格略高。

**注意：**

S7-300 系列 PLC 的一个机架上，除了电源、接口模块和 CPU 以外，有 8 个槽位可以安装模块，所有模块包括功能模块（FM）等双槽宽的模块在实际的配置中只占用一个槽位，所以一个机架上最多可以安装 8 个模块，没有槽号的限制。电源模块（PS）向 CPU 或接口模块（IM）提供 24V 电源，CPU 和接口模块向底板总线提供 5V 电源，一个机架上安装模块所消耗的电流总数不能超过提供的最大电流值，一个机架上模块的安装数量受到槽号和电流消耗的限制（每个模块消耗背板总线电流可以参考 S7-300/400 系列 PLC 选型手册）。

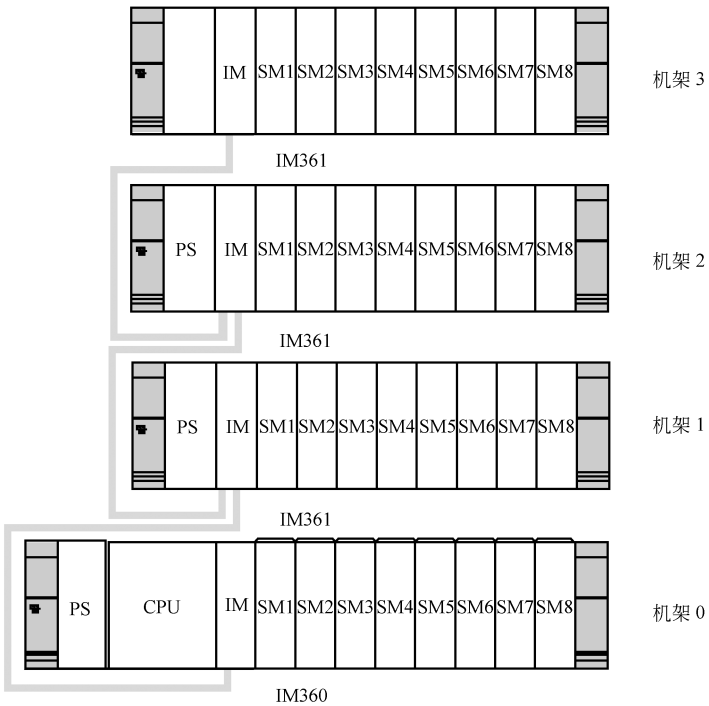



图 3-2 使用 IM360/IM361 的中央扩展

## 3.2 S7-400 系列 PLC 的中央扩展

S7-400 系列 PLC 系统使用接口模块进行中央扩展结构，如图 3-3 所示。

S7-400 系列 PLC 中央扩展的规则如下：

1）在中央机架（CR）上最多可以安装 6 块 IM 发送器，但是最多只能安装 2 块具有电源发送功能的 IM 发送器。

2）一个中央机架最多可以连接 21 个扩展机架（ER）。

3）每个扩展机架编码开关可以设置为 1 ~ 21，编码序号必须惟一。一个 IM 发送器接口最多可以连接 4 个无电源传送的扩展机架（ER）或一个带有电源传送的扩展机架（ER）。有些 IM 发送器具有两个接口，如一个 IM460-0 具有两个发送器接口，最大可以扩展 8 个机架。

4）扩展机架的通信总线功能被限制到 7 号机架，也就是说 1 ~ 6 号扩展机架具有通信总线功能。

5）不能超过最大的扩展距离。

6）中央机架和扩展机架都可以使用通用机架（UR）。

**注意：**

S7-400 系列 PLC 一个机架最多有 18 个槽，电源（PS）、CPU 都要占用槽位，有的 CPU

和电源占用双槽位，一个机架上最多可以安装模块的个数与槽位有关；此外，电源模块向背板提供5V 和24V 电源，机架上所有的模块消耗背板总线电流不能超过电源模块的输出。

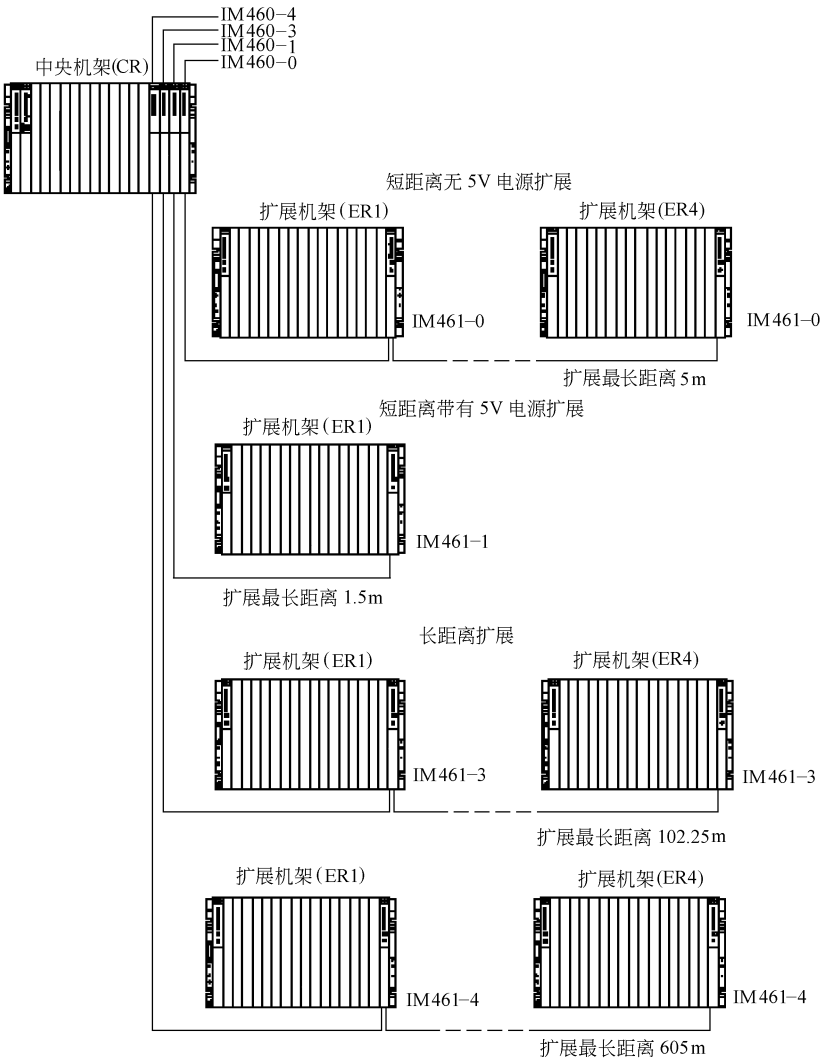



图 3-3　S7-400 系列 PLC 的中央扩展

## 3.3　S7-300、400 系列 PLC 的分布式扩展

通过现场总线，以通信的方式对远程 I/O 站点信号进行读写操作，这种分布式扩展方式渐渐替代了原有的中央扩展方式，使用分布式扩展方式有以下几点好处：

1）使用分布式扩展，可以大大地扩展 CPU 与远程 I/O 站的距离，最远可以达到几十千米。

2）接线方便，远程 I/O 站通常放置于传感器附近，通过一根总线就可以将 I/O 站的信号采集到 CPU 中，而不用将所有的信号线连接到主控室。

3）节省布线的费用，使用远程 I/O 站，大大降低了电缆的使用量，从而降低了布线费用。

4）远程 I/O 站的种类多，适合各种应用的场合。

5）便于调试，复杂的接线往往增加调试的时间，但使用分布式扩展的方式，在 I/O 站即可校正传感器的信号。

6）节省工程时间。

7）便于设备的维护。

8）由于通信方式统一，第三方的设备很容易集成到系统中。

9）为了增加系统的可靠性，远程 I/O 站也可以成为智能从站，独立控制远程设备。

10）在一些不便布线的场合，如旋转台、玻璃隔断等，可以使用红外、无线以太网、滑触线等方式通信。

11）通过现场总线，可以对总线上所有智能的站点进行编程。

通过现场总线 PROFIBUS-DP 实现的分布式扩展，如图 3-4 所示（STEP7 配置界面）。

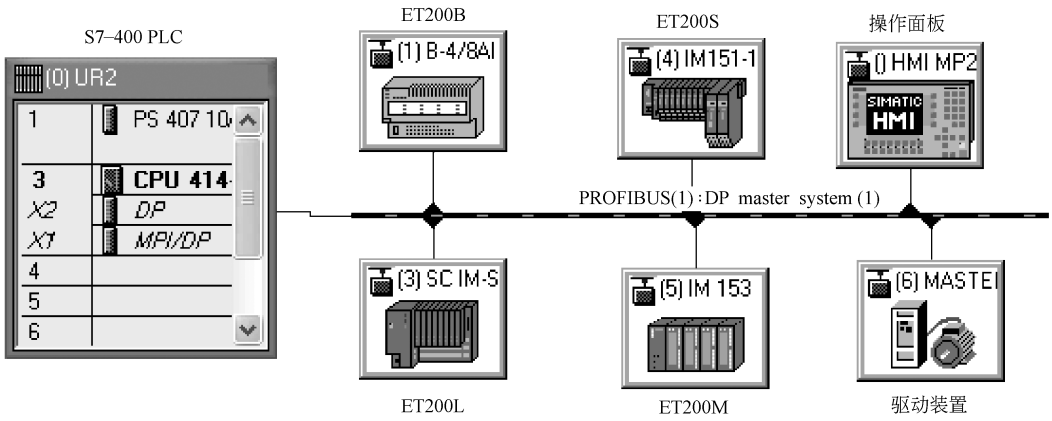



图 3-4　通过 PROFIBUS-DP 实现的分布式扩展

PROFIBUS-DP 现场总线的通信介质为屏蔽双绞线或光纤，一条现场总线带有远程 I/O 站的个数与主站的类型有关，通常 S7-300 系列 PLC 集成的 DP 接口可以连接 64 个远程从站，S7-400 系列 PLC 可以连接 123 个远程从站。PROFIBUS-DP 主站循环访问从站数据，具有独立的处理功能，不占用 CPU 的运行时间，从站轮询时间典型值为 10ms 左右，满足实时性的要求。

随着以太网的快速发展，以太网的实时性得到很大的提高，基于实时以太网的 PROFINET-IO 满足现场设备实时性的控制要求，也可以实现分布式扩展，如图 3-5 所示（STEP7 配置界面）。

PROFINET-IO 与 PROFIBUS-DP 相比，连接的 I/O 站个数更多，如果使用等实时（IRT）技术，可以满足非常高的实时性要求，循环时间可以为 1ms，循环间隔误差小于 1μs。

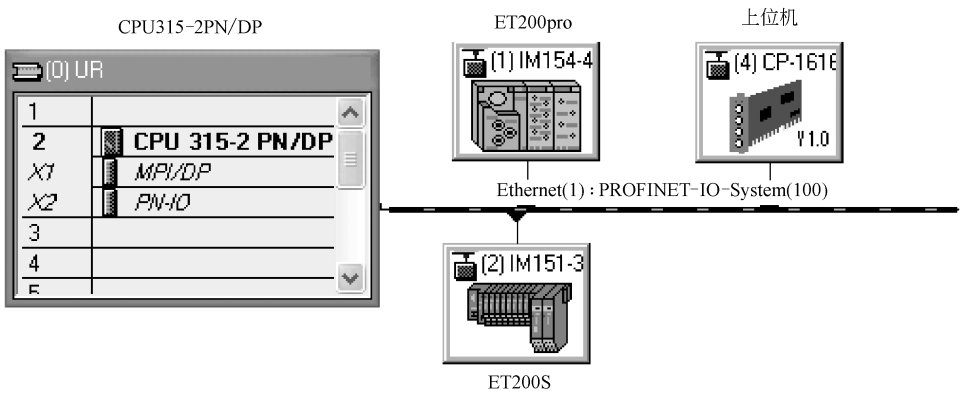



图 3-5　通过 PROFINET-IO 实现的分布式扩展

# 第 4 章　S7 系列 PLC 编程软件——STEP7 简介

在上面的章节中主要介绍了 S7-300、S7-400 系列 PLC 的硬件组成，通过软件编程控制硬件执行的动作及顺序从而达到控制外围设备的目的，STEP7 就是实现编程任务的工具软件。一个简单完整的编程过程如图 4-1 所示。

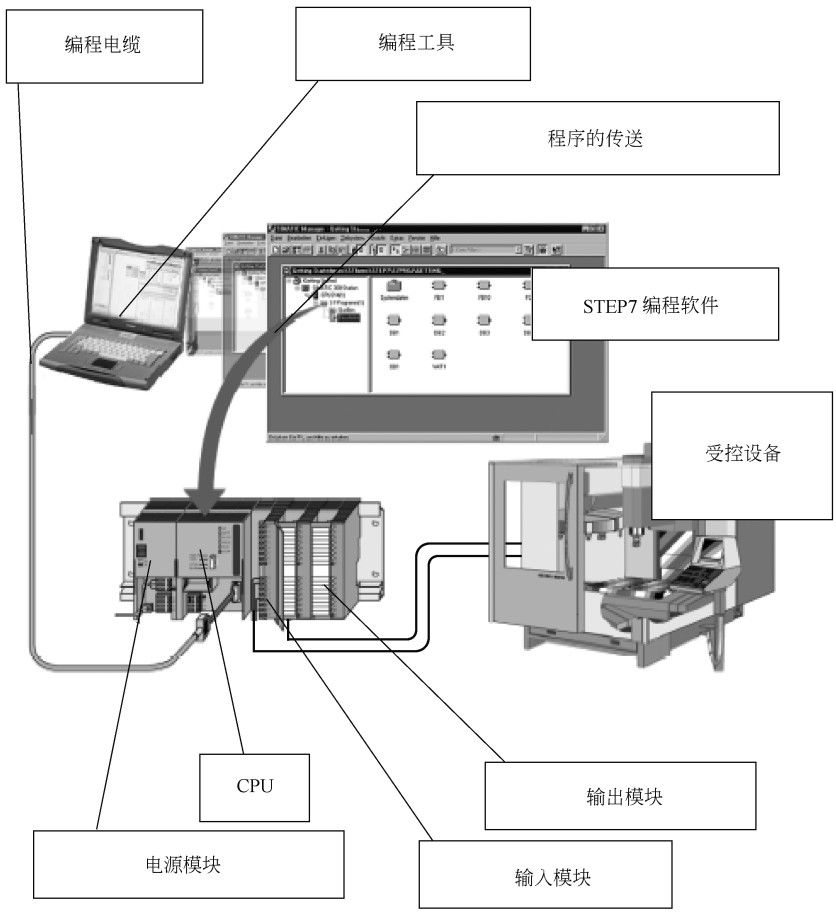



图 4-1　简单的编程应用

根据设备的控制，选择 PLC 的硬件系统，包括电源模块、CPU 的类型、输入和输出点的数量。在 STEP7 中编写用户程序，通过编程工具（西门子公司的工业编程器 PG 或商用 PC）将程序下载到 CPU 中，同样 CPU 中的程序也可以上传到编程工具中，程序执行后可以通过在线程序和变量表监控设备的运行状况，完成一个简单的自动化控制任务。

## 4.1　STEP7编程软件的订货版本

根据不同的应用，STEP7软件有以下几个订货版本：

（1）STEP7 Lite

适用于S7-300、C7系列PLC以及带有CPU的ET200X和ET200S系列分布式I/O的编程、硬件配置的软件包，不能作为一个软件平台集成其他软件，如WinCC、PROTOOL等，也不能安装可选软件，如CFC、SCL等编程语言，不能配置CP和FM模块，只适合简单的单站控制，价格便宜，相当于STEP7简装版。

（2）STEP7 Basis

适用于S7-300/S7-400、M7-300/M7-400和C7系列的编程、硬件配置的标准软件包，与STEP7 Lite相比没有上述的限制，可以作为一个软件平台集成其他软件，可以实现STEP7软件所有功能。

（3）STEP7 Professional

除包含了STEP7 Basis版本中标准组件外，还扩展了STEP7的编程语言和调试工具，如SCL、GRAPH、PLCSIM，相当于软件的打包销售，比软件的拆分购买要便宜。

（4）STEP7升级版

可以将多个不同低版本的STEP7升级为最新的版本（如有V2、V3、V4三套版本的软件，可以同时升级为三套最新版本的软件，对相同版本的软件只能升级一次），满足新的硬件和软件功能的需求。

**注意：**

STEP7软件不能对S7-200系列PLC进行编程，S7-200系列PLC的编程软件为STEP7 MICRO-WIN。

## 4.2　STEP7编程软件的安装

STEP7作为SIMATIC产品软件平台，为了确保软件运行的稳定性，对安装的环境有明确的要求。不同版本的STEP7软件对于安装的软件和硬件环境有不同要求，以STEP7 V5.4为例介绍对软硬件的要求。

### 4.2.1　硬件要求

操作系统为“MS Windows 2000 Professional”或“MS Windows XP Professional”时，对硬件的要求：

- CPU：主频600MHz以上；
- RAM：512MB内存（推荐1GB）；
- 显示设备：XGA，支持1024×768pix分辨率，16bit以上彩色深度。

操作系统为“MS Windows Server 2003”时，对硬件的要求：

- CPU：主频2.4GHz以上；
- RAM：1GB内存；

- 显示设备：XGA，支持 1024 × 768pix 分辨率，16bit 以上彩色深度。

### 4.2.2 软件要求

- MS Windows 2000 Professional SP4；
- MS Windows XP Professional SP1（或 SP1a）或 SP2；
- MS Windows Server 2003 带有 SP1 或作为工作站不带有 SP1；
- Microsoft Internet Explorer 6.0（或以上）版本。

Microsoft Windows 3.1/95/98/ME/NT 不再支持 STEP7 V5.4 版本，Windows XP Home 版本也不支持 STEP7 V5.4 的安装。

**注意：**

*在安装不同版本的 STEP7 软件时，自述文件会主动提示软、硬件的要求。STEP7 V5.3 中文版需要安装于 MS Windows XP Professional 操作系统上。*

### 4.2.3 语言设置

与其他软件安装相同，在 Windows 2000/XP 操作系统中只有具有管理员（Administrator）权限才能进行 STEP7 的安装。

由于不能识别中文字符，在 STEP7 安装过程中会出现安装中断的情况，为了避免上述情况的发生，在安装前将“区域和语言选项”中的语言设置为英文。首先进入控制面板，点击“区域选项”图标进行语言选择。在操作系统“MS Windows 2000 Professional”和“MS Windows XP Professional”中操作界面稍有不同。

“MS Windows 2000 Professional”操作系统：

在“常规”选项栏中，点击“设置默认值”按钮，在弹出的对话框中选择英语（美国），然后点击“确定”按钮，重新启动计算机后，设置生效。

“MS Windows XP Professional”操作系统：

在“高级”选项栏中，选择匹配的语言为英语（美国），然后点击“确定”按钮，重新启动计算机后，设置生效。设置界面如图 4-2 所示。

在 STEP7 安装完成之后，可以仿照上述步骤，恢复原先的区域设置，不会影响到中文字符的编辑。

**注意：**

*设置完成后自动弹出重新启动*

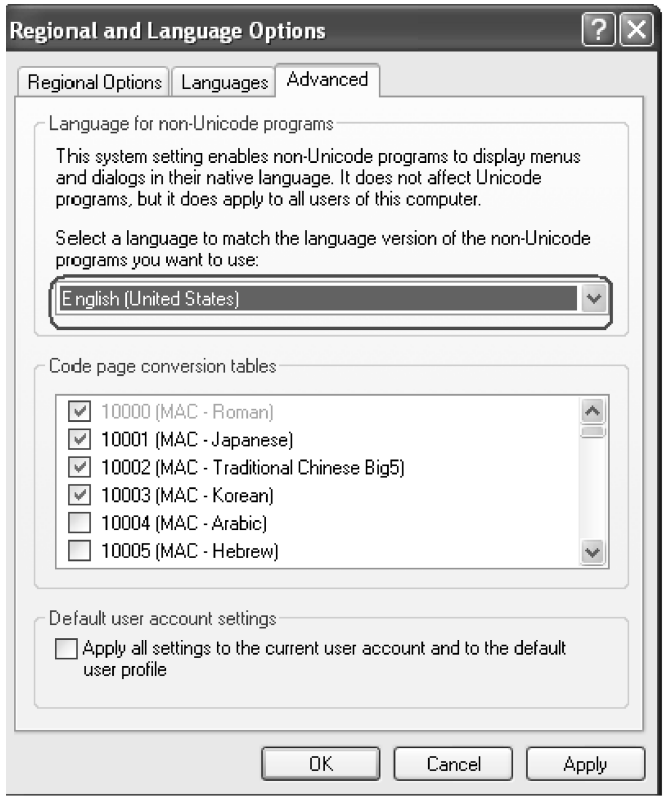



图 4-2 区域选项设置（XP 操作系统）

的提示对话框，只有重新启动后设置才能生效，如果没有弹出提示对话框，可能是设置错误。

### 4.2.4　安装步骤

插入STEP7安装光盘后，安装程序会自动执行，安装程序将指导整个软件安装过程，如果不使用自动安装程序，可以通过根目录中的SETUP.EXE文件执行安装过程。在安装STEP7前，操作系统需要重新启动以激活相应的服务。

1．选择安装软件

在可选安装软件对话框中，可以选择Acrobat Reader、STEP7和授权管理器，如图4-3所示，如果用户的PC上已经安装了该软件，可不必选择。授权管理器是可以对PC上安装的软件授权进行导入和导出，并查看授权的状态（好、故障、正在使用中）。

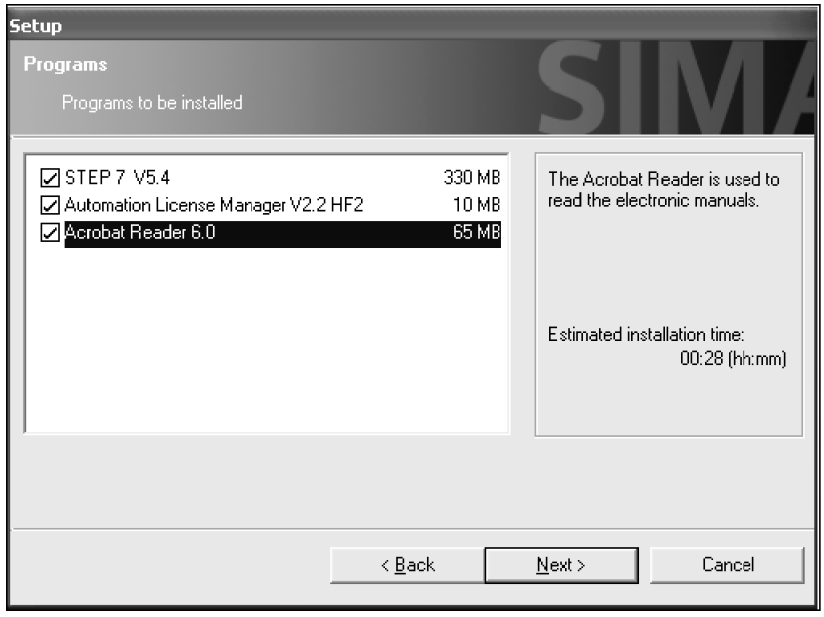



图4-3　选择安装软件（STEP7安装）

点击“Next”按钮，将自动安装所选择的软件，如图4-4所示。




图4-4　安装选择的软件（STEP7安装）

2. 选择安装方式和安装路径

在安装 STEP7 软件时，弹出对话框，选择安装的方式和安装路径，如图 4-5 所示。

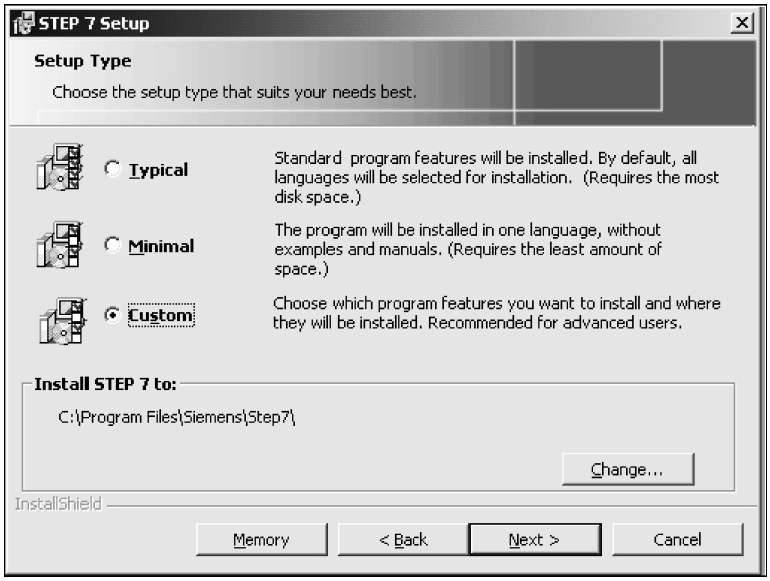



图 4-5 选择安装方式和安装路径（STEP7 安装）

（1）典型安装（Typical）

安装所有的语言、所有的应用程序、项目示例和文档，占用较大的硬盘空间。对于初次使用者选择典型安装不会错过需要的文档和示例程序，但其他语言的文档，如法文、德文等将占用较大的硬盘空间。

（2）最小安装（Minimal）

只安装一种语言和 STEP7 程序，不安装项目示例和文档，占用最小的硬盘空间。

（3）自定义安装（Custom）

用户可以选择希望安装的程序、语言、项目示例和文档。使用自定义安装类型，可以选择需要的示例程序和手册，优化安装空间。

安装路径可以任意选择，但是不能安装在带有特殊字符（ASCII 表中没有列出）的目录下，如果路径下带有中文，可能在安装过程中出错。

如果 PC 上已经安装了 STEP7 先前的版本，在安装 STEP7 V5.4 时不会将原有的版本删除（STEP7 V5.4 以下版本覆盖 PC 原有软件时，需要先删除原有的版本后再进行安装），可以选择直接升级原有版本软件，如图 4-6 所示。

3. 安装过程中导入软件授权

在安装过程中，将执行授权的安装程序，如图 4-7 所示，如果选择“Yes，transfer should take place during the installation”，系统自动将插入软驱的授权盘中的授权导入计算机中，如果授权盘中没有授权或损坏，将会报错提示；如果选择“No，transfer License Keys later”，系统将跳过授权安装程序继续安装，授权将通过授权管理器进行安装。

4. 选择写卡器类型

写卡器可以直接将用户程序写入到 Flash 卡、MMC 卡中，也可以将存储于这些卡中的

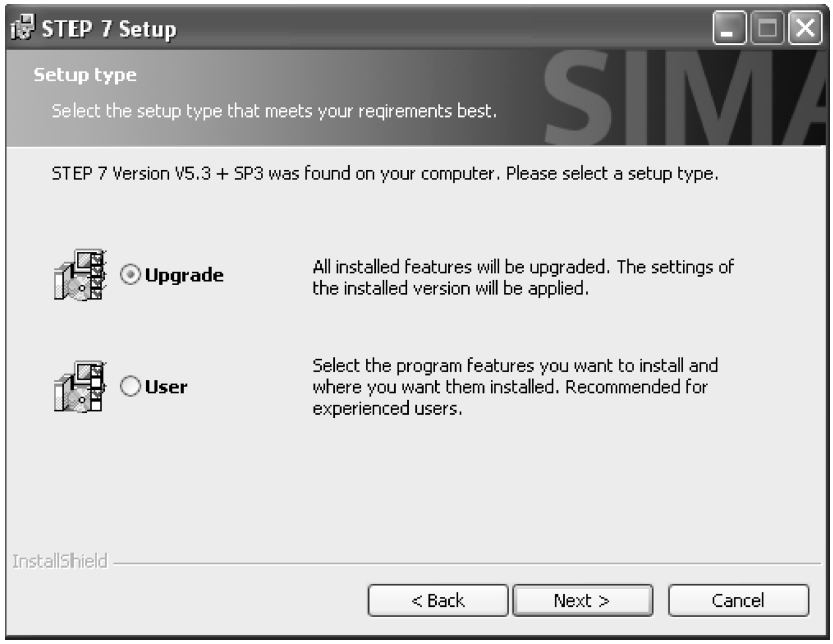



图4-6　覆盖原有版本的STEP7（STEP7安装）

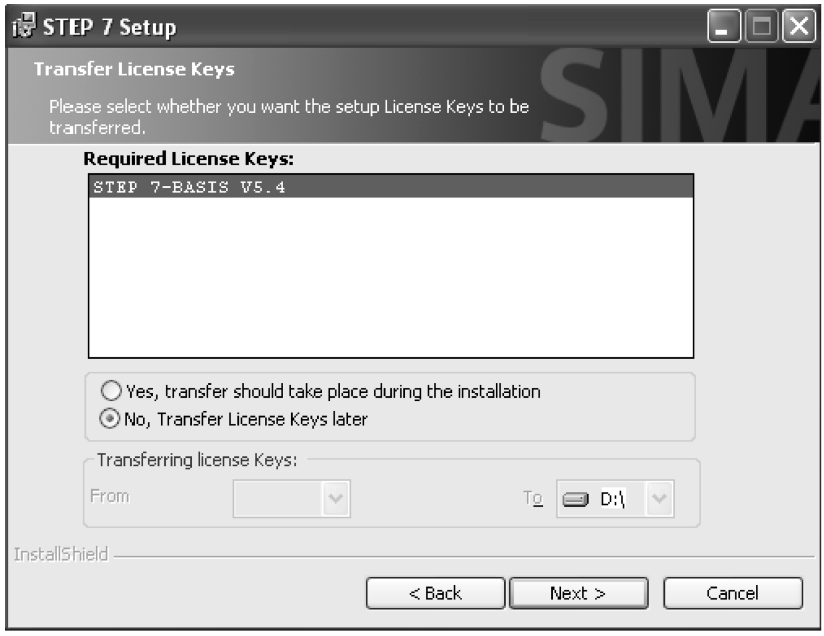



图4-7　选择软件授权（STEP7安装）

程序上传到编程工具中，写卡器也是CPU升级的必要工具。STEP7在安装过程中，系统将自动检查计算机是否带有写卡器，并弹出对话框对写卡器参数进行配置，如图4-8所示。

如果没有安装写卡器，选择“None”选项；如果使用西门子公司的工业编程器PG，已内置写卡器，选择“Internal programming device interface”；如果使用商业PC，安装了外置写

卡器，选择“External prommer”。如果对旧的 M7 系统存储卡进行格式化（必须预先安装写卡器），则需要激活“Flash file for M7 on device”选项。

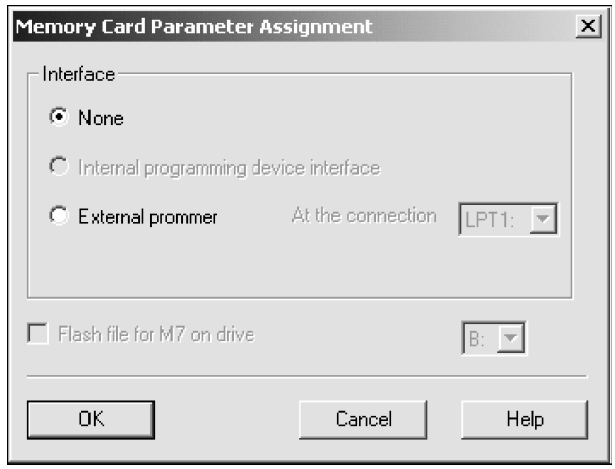



图 4-8 配置写卡器参数（STEP7 安装）

5. 选择编程接口

软件安装后期，弹出编程接口配置对话框如图 4-9 所示，在对话框中可以选择 MPI、PROFIBUS、以太网网卡（STEP7 V5.3 版本以上集成以太网卡作为编程接口的驱动）中任一个网卡或适配器作为编程接口。编程接口也可以在软件安装完成后，通过控制面板中的“PG/PC Interface”设定。

**注意：**

PC 自动识别通信处理器 CP，如 CP5512、CP5613 等不需要安装。如果在 PC 上插入通信处理器而在编写接口选型中没有出现，则需要检查硬件安装，重新识别外围硬件设备。PC 适配器和 TS 适配器需要在编程接口选型中安装。

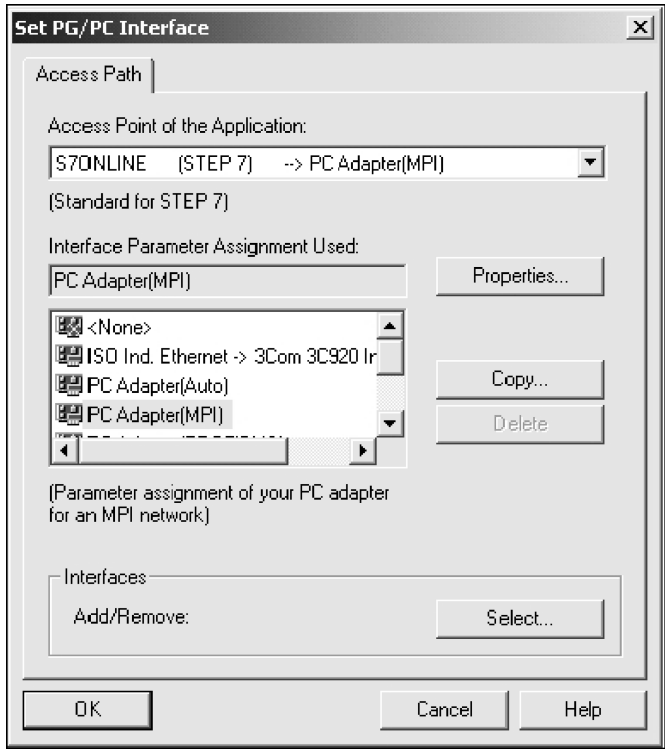



图 4-9 设置编程接口（STEP7 安装）

软件安装完成后，重新启动计算机，STEP7 安装完毕。

## 4.3　STEP7编程软件的卸载

使用视窗标准的方法卸载STEP7软件：

1）在控制面板中，双击“Add/Remote Program”图标，打开视窗软件安装对话框。

2）在已经安装软件条目中，选择STEP7软件，点击“Add/Remote”按钮，卸载软件。

3）弹出“Remove Shared File”的对话框，如果不需要处理，选择“No”按钮。

如果不能完全卸载STEP7软件或卸载软件时出现故障，则需要在注册表中删除，具体步骤如下：

1）点击“开始”→“运行”，输入“REGEDIT”。

2）将“HKEY_CURRENT_USER”→“SOFTWARE”→“SIEMENS”→目录下的“STEP7”文件删除。

3）将“HKEY_LOCAL_MACHINE”→“SOFTWARE”→“SIEMENS”→目录下的“STEP7”文件删除。

## 4.4　授权管理功能

使用STEP7及其他SIMATIC软件时需要授权，一个授权授予使用者合法的权利使用相应的产品。每一个软件产品都相对应有一个授权，如上位监控软件WinCC有WinCC的授权，使用CFC编程时需要相对应的CFC授权，如果同时使用多个软件则需要在一台编程器中安装多个授权。授权可以存放在软盘上、本地硬盘和网络服务器上，例如在购买STEP7软件时，除带有数据光盘，还带有一张装有授权的3.5in软盘，经过特殊制作、不能复制的授权盘（软盘）作为授权的载体，通过授权管理软件可以将授权传递到任意一台PC上，使这台PC带有使用权限。

### 4.4.1　授权的种类

对于西门子公司的软件产品，有不同类型的授权，见表4-1。

**表4-1　授权类型**

| 授权类型 | 描　述 |
|---|---|
| 单一授权 | 软件可以在任意一台PC（使用本地硬盘中的授权）上使用，没有时间的限制 |
| 浮动授权 | 授权可以安装在网络服务器中，同时只允许一台客户机使用授权，没有时间的限制 |
| 临时授权 | 从开始使用，只有14天有效期 |
| 租用类型授权 | 使用周期最高为50天，总共使用1200小时受限 |
| 升级类型授权 | 将旧版本的授权升级为最新版本的授权，一个升级授权可以将不同旧版本授权升级为多个最新版本授权，如可以将V2.X、V3.X、V4.X 3个旧版本授权升级为3个最新版本授权，相同版本只能升级一次 |

### 4.4.2　授权管理器

在安装STEP7软件时，必须选择安装授权管理器，授权管理器可以传递、检测、删除

授权，操作界面如图 4-10 所示。

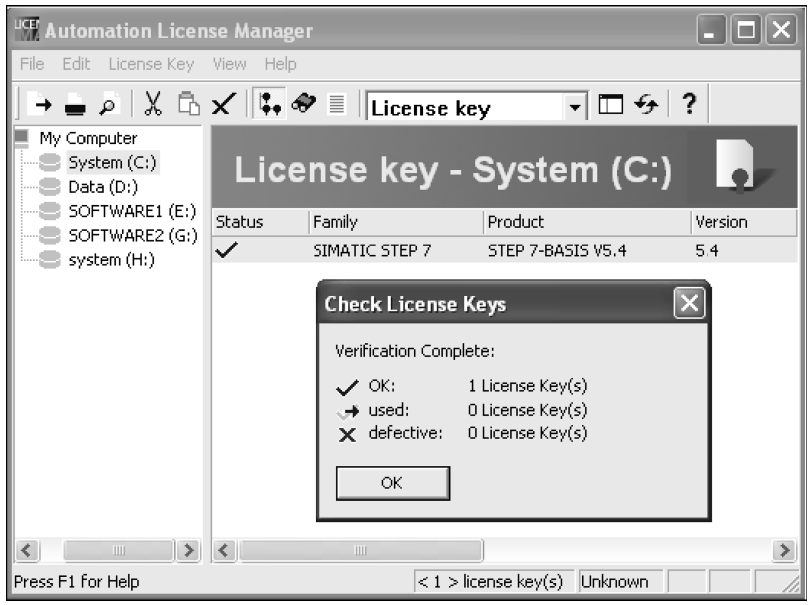



图 4-10　授权管理器操作界面

在初始状态，点击软盘中包含的授权，拖放到 PC 任意的硬盘分区中即可（有些软件要求授权安装在系统分区上），这样授权就安装在这台 PC 上，可以使用带有授权的 PC 操作软件，例如使用 STEP7 进行编程、调试和监控。如果将授权移到其他的 PC 上，那么这台 PC 将没有授权使用相关软件，得到授权的 PC 将可以进行操作。点击授权，可以查看授权的状态，也可以传送到其他分区中。

### 4.4.3　使用浮动授权

浮动授权可以安装在网络上任意的 PC 上，其他 PC 可以共享这台 PC 上的授权，一个浮动授权只能被网络上的一台 PC 使用（包括本机），如果授权再次空闲，才可以被网络上其他一个 PC 使用。使用浮动授权共享方式时，必须进行设置。

带有浮动授权 PC 上的设置：在授权管理器中，点击“File”→“Customize”，弹出对话框如图 4-11 所示，点击“General”栏，将本地授权

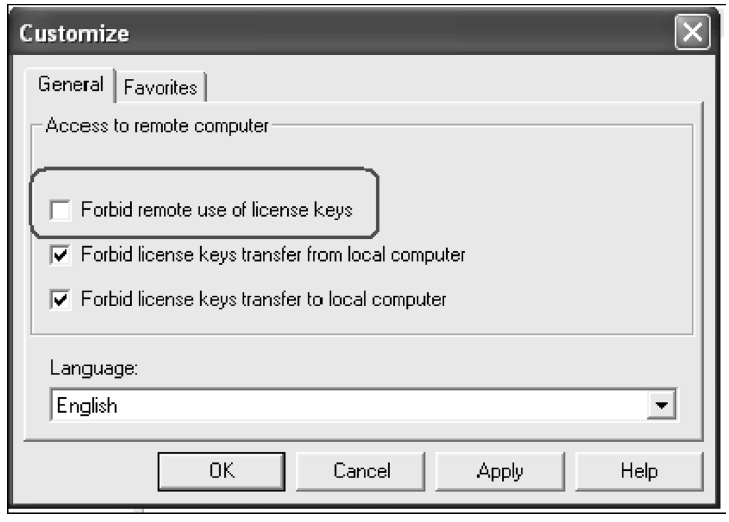



图 4-11　共享浮动授权 PC 上的设置界面

共享。

远程PC上的设置：在授权管理器中，点击“File”→“Customize”，弹出对话框如图4-12所示，点击“Favorites”栏，选择使用哪一个共享PC上的授权。

授权除可以在本机硬盘间相互传递外，还可以通过网络传递。在图4-11界面中必须将禁止传递功能去除，点击菜单命令“Edit”→“Connect Computer”后连接到远程PC上，将授权拖放到远程PC的硬盘分区上。

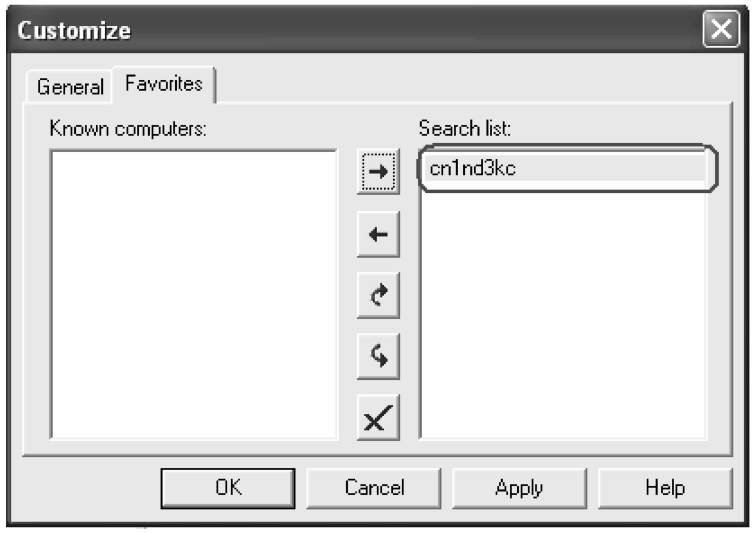



图4-12　远程PC上使用浮动授权的设置界面

## 4.5　STEP7标准软件包

STEP7标准软件包集成的SIAMTIC编程语言符合EN61131-3标准，图形化与面向对象的视窗操作理念相匹配，STEP7标准软件包对生成一个自动化控制任务的不同阶段提供强有力的支持，例如：

- 设置和管理项目；
- 配置硬件参数和通信连接；
- 符号表的管理；
- 用户程序的上传和下载；
- 调试；
- 诊断。

STEP7标准软件包提供不同的应用工具，如图4-13所示。

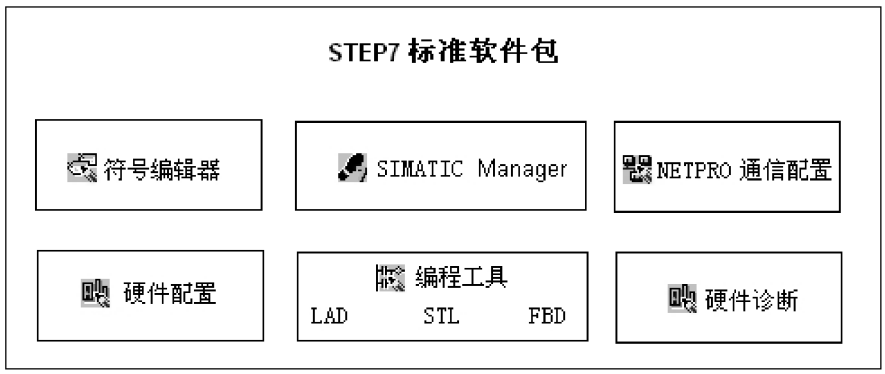



图4-13　STEP7标准软件包集成的应用工具

### 4.5.1 SIMATIC Manager

STEP7 是 TIA 的软件平台，不同设备的应用软件可以集成于 STEP7 中，SIMATIC Manager 管理一个项目下所有不同设备的数据，例如驱动、人机界面、C7 等设备及控制器的数据。一个项目下的数据都是透明的，例如可以实现将 PLC 项目中的变量上传到上位监控软件 WinCC 中，自动生成通过 PROFIBUS-DP 与驱动设备连接的通信数据等功能。在视窗桌面上点击图标或通过菜单路径“开始菜单”→“Simatic 程序组”→“SIMATIC Manager”进入，SIMATIC Manager 窗口，如图 4-14 所示。

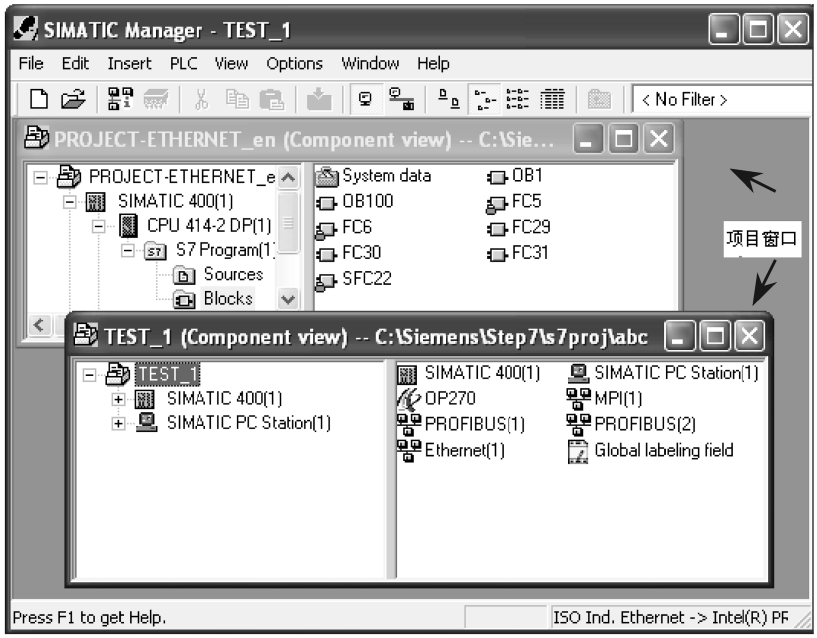



图 4-14 SIMATIC Manager 界面

在 SIMATIC Manager 中可以创建多个项目，每一个项目都有一个独立的窗口，在项目的窗口中分为左右两部分：左边部分为项目结构窗口，在项目结构窗口中包括项目中所有的站点，如 S7-300 站、HMI、S7-400 站等，在每个单站中包括硬件信息、程序、符号表等条目；在右边部分的窗口中显示左边窗口文件夹中的对象和子文件夹，例如在左边窗口中点击项目名，在右边窗口显示项目下所有的站点；在左边窗口中点击程序块“Blocks”文件夹，在右边窗口显示文件夹中所有的程序块。

通过 SIMATIC Manager 菜单栏中的菜单选项可以完成不同的设置功能，不同菜单完成的功能如下：

- “File”：实现对项目文件的编辑、组织、存储、删除及打印功能，除此之外还可以对存储卡中的项目进行管理。
- “Edit”：实现对选择的对象进行编辑的功能，如复制、删除、查看属性、生成 PROFINET 组件等。
- “Insert”：插入一个程序组件，如网络、程序块等。

- “PLC”：实现程序的上传、下载、监控、查询以太网站点等功能。
- “View”：实现对项目组件的查看功能，可以切换项目的离线、在线状态。
- “Options”：项目设置选项，完成变量交叉参考表、配置网络、设定编程及监控格式等功能。
- “Window”：完成对窗口的设定。
- “Help”：在线帮助功能，点击对象，按“F1”键将弹出该对象的帮助文件。

SIMATIC Manager中的设置命令采用保护功能，如果没有选择正确的设置对象，菜单中的按钮将不能操作。在菜单下面的快捷键可以完成菜单中部分频繁使用的操作，点击操作对象，使用鼠标右键弹出的下拉菜单同样可以实现快速操作的功能。

### 4.5.2　硬件配置

在SIMATIC Manager中插入一个站点时，首先需要对站点的硬件进行配置，包括配置网络站点、分配输入及输出地址、设定模拟量连接传感器的类型等，这些信息将下载到CPU中，CPU识别和监控配置信息，如果配置信息与实际的硬件不匹配，有可能配置错误或硬件故障，CPU将报错，并将诊断信息写到诊断缓存区中，只有完成硬件配置后才能生成相应的CPU程序块文件夹。点击“Hardware”图标，进入硬件配置界面，如图4-15所示。

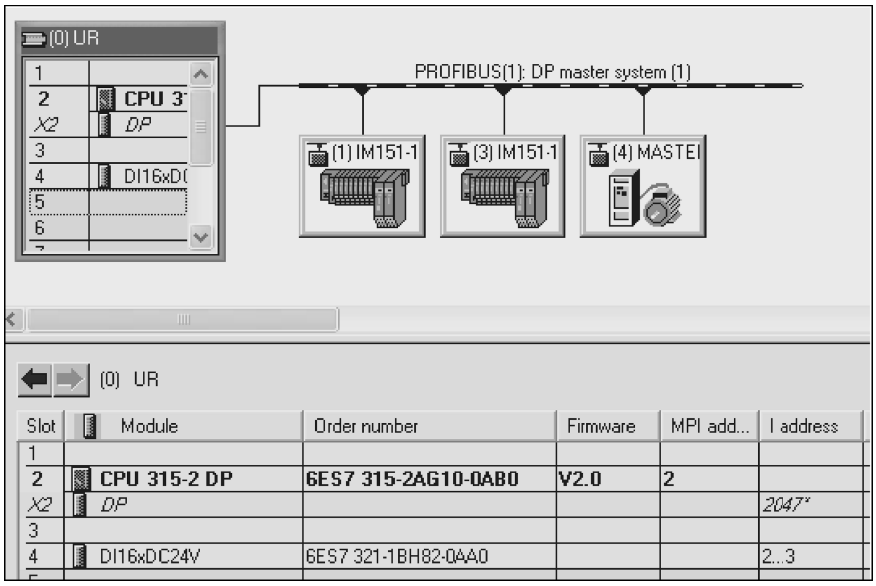



图4-15　硬件配置界面

### 4.5.3　编程工具

点击站点中的程序块进入编程界面，在STEP7中集成三种编程语言，即梯形图（LAD）、语句表（STL）和功能块图（FBD）。大部分梯形图程序和功能块图程序可以相互转换，对于初学者来说，使用梯形图或功能块图编程，可以从指令库中将指令直接拖放到程序中使用，调试也比较直观，但是占用CPU存储器的空间比较大，存储器的大小决定CPU

的价格，同时指令的多少也影响 CPU 的扫描时间；使用语句表编程，没有梯形图和功能块图的程序中用于显示而与控制无关的指令，对于复杂的程序可以使用地址指针的方式间接寻址，占用 CPU 存储器的空间比较小，减少 CPU 的扫描时间，在一些进口项目中大多使用语句表编程方式，但是语句表编程需要牢记指令，不能使用拖放的方法将指令添加到程序中，调试不直观，复杂程序需要使用断点调试。使用三种语言编写相同控制功能的程序，例如将变量 MW10 加上变量 MW12 后除以变量 MW16，将运算结果存储于变量 MW18 中，表示方法如图 4-16 所示。

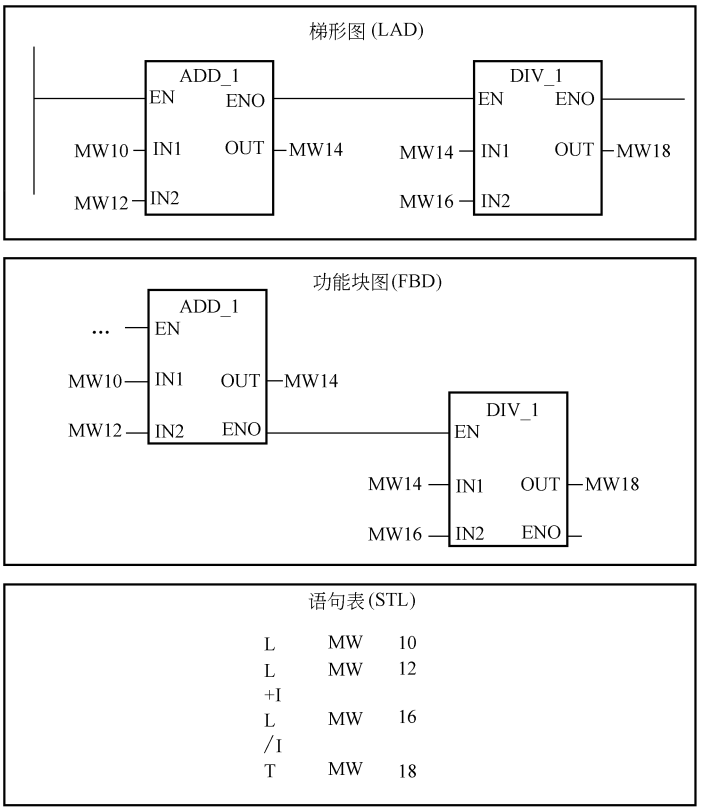



图 4-16　三种编程语言的表示方法

### 4.5.4　符号编辑器

在每个程序下都包含符号表编辑器，该工具为输入/输出信号（Input/Output）、位存储（Bit Memory）和块（Block）设定符号名和注释，对程序的编写和阅读具有很好的助记功能，符号表是全局有效的，可供其他工具使用，例如将符号表中定义的变量传送到上位监控软件 WinCC，也可以在符号表中定义 PROFIBUS-FMS 通信变量等。

### 4.5.5　硬件诊断

如果 CPU 或者模块上的故障显示灯点亮，可以通过硬件诊断功能对 PLC 站的各硬件模块进行在线状态诊断，显示相关故障的信息和状态信息，便于修改和更正。

### 4.5.6　NetPro网络配置

在网络配置界面中配置站点间的通信关系，例如通过PROFIBUS网络或者工业以太网建立主站间的通信连接，根据使用的硬件和通信要求选择FDL、FMS、S7、TCP、ISO_ON_TCP等通信连接，PROFIBUS-DP主从通信在硬件配置界面直接配置，不需要在NetPro中配置连接属性。在网络配置界面中加入PG/PC，可以在整个通信网络中实现跨网络的编程功能，可以对网络上任意一个站点进行编程和调试。

## 4.6　STEP7扩展软件包

除STEP7标准软件包，还可通过可选软件包进行扩展，以适合不同场合的需要，例如可以安装图形化的编程软件GRAPH和HI-GRAPH，以及适合计算功能的SCL编程语言等。同时STEP7也是一个平台软件，几乎所有与PLC相关的软件都可以以STEP7为平台进行安装，包括驱动调试软件、人机接口软件等，通过软件集成安装，实现全集成自动化的理念，使一个大的项目中所有的设备，在一个平台上进行操作，使站点间的通信更透明、报警信息全局化。这些可选软件包按照功能可以分为三类：

1）工程工具（Engineering Tool）：较高层次的编程语言和面向工艺的软件。

2）运行版软件（Run-Time Software）：生产过程中一些常用的预编程的解决方案。

3）人机接口（Human Machine Interface，HMI）：用于操作人员对设备进行监控的软件。

表4-2列出了STEP7可选软件包的应用条件。需要注意的是，这些软件并不包括在STEP7 V5.4标准软件包的安装光盘中，需单独订购，并安装相应的授权。

**表4-2　STEP7扩展软件包**（不包括驱动和仪表软件）

| | S7-300<br>S7-400 | C7 |
|---|---|---|
| Engineering Tool | | |
| · Borland C/C ++ | | |
| · CFC | + ① | + ② |
| · DOCPRO | + | + |
| · HARDPRO | + | |
| · M7 ProC/C ++ | | |
| · S7 GRAPH | + ① | + ② |
| · S7 HiGraph | + | + |
| · S7 PDIAG | + | |
| · S7 PLCSIM | + | + |
| · S7 SCL | + | + |
| · Teleservice | + | + |
| Run-Time Software | | |
| · Fuzzy Control | + | + |
| · M7-DDE Server | | |

（续）

| | S7-300<br>S7-400 | C7 |
|---|---|---|
| · M7-SYS RT | | |
| · Modular PID Control | + | + |
| · PC-DDE Server | + | |
| · PRODAVE MPI | + | |
| · Standard PID Control | + | + |
| Human Machine Interface | | |
| · ProAgent | | |
| · SIMATIC ProTool | | |
| · SIMATIC ProTool/Lite | | O |
| · SIMATIC WinCC | | |

注：O 表示必须的；+表示可选的。
① 建议从 S7-400 系列 PLC 以上使用。
② 对于 C7 不推荐。

### 4.6.1 工程工具

工程工具有三种类型，即高级编程语言、图形化的编程语言、辅助软件。

（1）高级编程语言

S7-SCL 是基于文本符合 IEC1131-3 标准的高级编程语言，SCL 的语言结构类似于 C 语言和 PASCAL 语言，适合于熟悉高级编程语言的用户，应用于复杂或频繁再次调用的程序。

（2）图形化的编程语言

1）S7-GRAPH 用于顺控（步和步间传递）的编程语言，在 GRAPH 编程语言中过程顺序被分成“步”，在每个“步”中包括动作和控制输出，当条件满足时传递到下一个“步”的控制，在程序调试中可以进行单“步”调试。

2）S7-HiGRAPH 是将控制过程以图形状态的形式描述为异步、非顺序控制的编程语言，将整个项目分成独立的功能单元，每个功能单元能够表示不同的状态，这些功能单元通过信息的交换进行同步操作。

3）S7-CFC 适合将复杂的控制和控制回路转换为简单的逻辑操作，在库中有大量的实现各种功能的块，将块复制到图表中，通过连接块的输入、输出引脚完成复杂的控制任务。CFC 只是编程语言，不能单独使用，需要使用编译器进行编译才能下载到控制器中，S7-300/400系列 PLC 的 CPU 使用 SCL 作为编译器，M7 使用 Borland C ++ 作为编译器，TDC、FM 458 控制器使用 D7-SYS 作为编译器。

（3）辅助软件

1）Borland C ++ 只适合 M7 编程（M7 系统已淘汰）。

2）DOCPRO 软件将 STEP7 的配置数据转换为配线手册。

3）HARDPRO 适合复杂控制的 S7-300 系列 PLC 硬件配置。

4）M7 ProC/C ++ 只适合 M7 系统（M7 系统已淘汰）。

5）S7-PLCSIM只适合S7系统，实现对用户程序进行离线模拟调试功能。

6）S7-PDIAG只适合S7系统，适合将自定义诊断上传到监控软件（WinCC、PROTOOL）中，在上位监控软件中可以同时监控与故障相关的程序。

7）TeleService软件提供远程编程的解决方案，通过电话线（模拟或数字）连接CPU的MPI、PROFIBUS接口进行远程编程和监控。

### 4.6.2　运行版软件

- 标准PID控制软件集成闭环控制器，适合数字量输出的PID脉冲控制器和模拟量输出控制器，通过参数赋值工具，可以在很短的时间内完成控制器的优化任务。
- 模块化PID控制软件集成更多的PID算法，适合复杂的PID调节。
- 模糊控制软件（FUZZY CONTROL）适合非线性的控制，可以同时控制多个模拟量输出实现复杂的控制任务，根据控制经验对隶属度参数（IF…THEN）进行赋值。
- PRODAVE MPI软件提供MPI的通信函数库，用户可以通过VB、VC编写上位监控界面。

### 4.6.3　人机接口

- 开放的过程监控系统SIMAATIC WinCC是一个标准的操作接口，适合于对控制对象的操作、监控及数据归档功能。
- SIMATIC ProTool软件适合对控制面板进行编程组态，将被新的组态软件WinCC FLEXIBLE替代。
- ProAgent安装于WinCC或ProTool，用于查询控制器的故障信息，与S7-PDIAG配合使用。

# 第 5 章　数据类型与地址区

## 5.1　S7-300/400 系列 PLC 的数据类型

用户程序中所有的数据必须通过数据类型来识别，只有相同数据类型的变量才能进行计算，数据类型主要分为三类：

- STEP7 中提供的基本数据类型；
- 复合数据类型，编程人员可以将基本数据类型组合为复合的数据类型；
- 参数类型，适合函数或函数块中形参的数据类型。

梯形图、语句表和功能块图的指令系统也是与数据类型相对应的，位逻辑指令只能对位信号进行操作，语句表中的装载（L）与传送（T）指令与梯形图、功能块图中的移动（MOVE）指令只能对字节、字和双字进行操作。一个位是一个二进制的数字，通过“0”或“1”表示，一个字节由 8 个位组成，一个字由 16 个位组成，一个双字由 32 个位组成。数学运算指令对字节、字和双字进行操作，这些字节、字和双字经过不同的编码可以转换为整数和浮点数。

### 5.1.1　基本数据类型

基本数据类型共包含 12 种，每一个数据类型都具备关键字、数据长度、取值范围和常数表达格式等属性。以字符型数据为例，该类型的关键字是 Char，数据长度 8bit，取值范围是 ASCII 字符集，常数表达格式为两个单引号包含的字符，如‘A’。STEP7 基本数据类型的关键字、长度、取值范围和以常数为例子的表示方法见表 5-1。

**表 5-1　STEP7 的基本数据类型**

| 数据类型及关键字 | 长度 | 取 值 范 围 | 常数表示方法举例 |
|---|---|---|---|
| BOOL（位） | 1bit | True 或 False | TRUE |
| BYTE（字节） | 8bit | 十六进制表达：<br>B#16#0 ~ B#16#FF | B#16#10 |
| WORD（字） | 16bit | 二进制表达：<br>2#0 ~<br>2#1111_1111_1111_1111<br><br>十六进制表达：<br>W#16#0 ~ W#16#FFFF<br><br>无符号十进制表达：<br>B#（0，0）~B（255，255）<br><br>BCD（二进制编码的十进制数）表达：<br>C#0 ~ C#999 | L2#0001<br><br>W#16#10<br><br>B#（10，20）<br><br>C#998 |

（续）

| 数据类型及关键字 | 长度 | 取 值 范 围 | 常数表示方法举例 |
|---|---|---|---|
| DWORD<br>（双字） | 32bit | 二进制表达：<br>2#0 ~ 2#1111_1111_1111_1111<br>1111_1111_1111_1111<br><br>十六进制表达：<br>DW#16#0 ~ DW#16#FFFF_FFFF<br><br>无符号十进制表达：<br>B#（0，0，0，0）~ B#（255，255，255，255） | 2#1000_0001<br>0001_1000<br>1011_1011<br>0111_1111<br><br>DW#16#10<br><br>B#（1，10，10，20） |
| CHAR<br>（字符） | 8bit | ASCII 字符集<br>‘A’、‘b’ 等 | ‘A’ |
| INT<br>（整数） | 16bit | －32768 ~ 32767 | 12 |
| DINT<br>（双整数，32 位） | 32bit | －L#2147483648 ~ L#2147483647 | L#12 |
| REAL<br>（浮点数） | 32bit | －3.402823E＋38 ~ －1.175495E－38，<br>0，<br>＋1.175495E－38 ~ ＋3.402823E＋38 | 12.3 |
| S5Time<br>（SIMATIC 时间） | 16bit | S5T#0H_0M_0S_10MS ~<br>S5T#2H_46M_30S_0MS | S5T#10S |
| TIME<br>（IEC 时间） | 32bit | IEC 时间格式（带符号），分辨率为 1ms：<br>－T#24D_20H_31M_23S_648MS ~<br>T#24D_20H_31M_23S_648MS | T#0D_1H<br>_1M_0S_0MS |
| DATE<br>（IEC 日期） | 16bit | EC 日期格式，分辨率 1 天：<br>D#1990－1－1 ~ D#2168－12－31 | DATE#1996－3－15 |
| Time_OF_DAY | 32bit | 24 小时时间格式，分辨率 1ms<br>TOD#0:0:0.0 ~ TOD#23:59:59.999 | TIME_OF_<br>DAY#1:10:3.3 |

下面简单介绍不同数据类型数据的表示方法：

（1）WORD（字）

一个 WORD 包含 16 个位，以二进制编码表示一个数值时，将 16 个位分为 4 个组，每个组 4 个位，组合表示数值中的一个数字或符号位，例如以十六进制表示数值 W#16#1234 的方法如图 5-1 所示，使用十六进制表示数值时没有符号位，所以十六进制表示的数值不可能有负值。

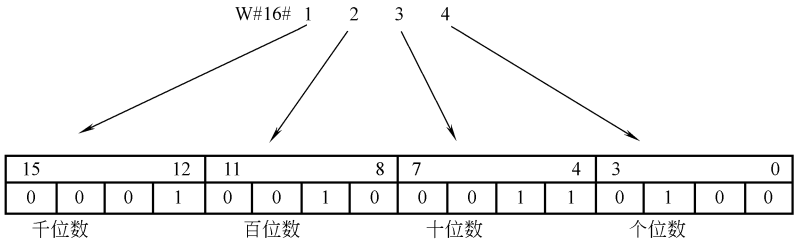



图 5-1　一个 WORD 数据类型的十六进制表示方法

以 BCD 码表示 +123 方法如图 5-2 所示，BCD 码通常表示时间格式数值，与十六进制表示方法相比较，BCD 码带有符号位，数值中不能含有 A、B、C、D、E、F 等十六进制数字。计数器 C 同样使用 BCD 码表示，但是不识别符号位，例如 +123 和 –123 表示计数器的值相同，都是 C#123。

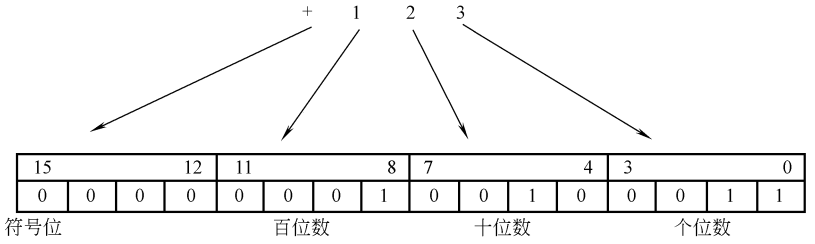



图 5-2 一个 WORD 数据类型的 BCD 码表示方法

DWORD 与 WORD 的表示方法相同。

（2）INT（整数）

一个 INT 包含 16 个位，在存储器中占用一个字的空间。第 16 位为符号位，可以表示数值的正负。以二进制编码表示一个数值时，除符号位以外将每一位信号的数值相加即可以表示一个整数，例如以整数方式表示 +34 的位图排列如图 5-3 所示。

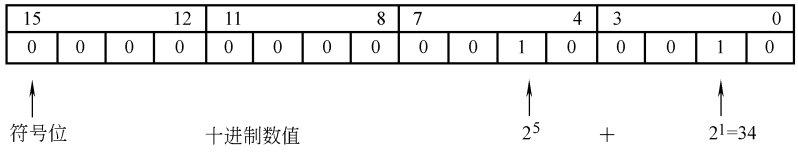



图 5-3 一个 INT 数据类型的正整数表示方法

一个负数的位表示方法则是在正数的基础上将所有位信号取反后加 1，如 –34 的表示方法如图 5-4 所示。

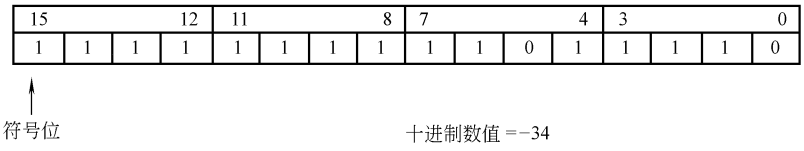



图 5-4 一个 INT 数据类型的负整数表示方法

DINT 与 INT 的表示方法相同。

（3）REAL（浮点数）

一个 REAL 包含 32 个位，在存储器中占用两个字的空间。STEP7 中的 REAL 数据类型符合“IEEE standard 754 – 1985”标准，一个 REAL 数值包括符号位 S、指数 e 和尾数 m，分别占用的位数如图 5-5 所示。

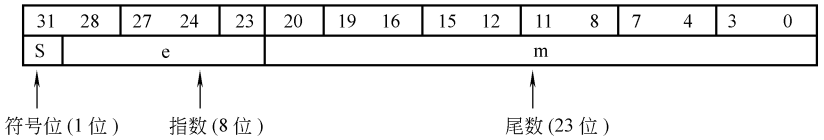



图5-5　一个REAL数据类型的表示方法

指数e和尾数m的取值见表5-2。

**表5-2　指数e和尾数m的取值**

| 浮点值的组成部分 | 位　　号 | 值 |
|---|---|---|
| 符号位S | 31 | |
| 指数e | 30 | $2^7$ |
| ⋮ | ⋮ | ⋮ |
| 指数e | 24 | $2^1$ |
| 指数e | 23 | $2^0$ |
| 尾数m | 22 | $2^{-1}$ |
| ⋮ | ⋮ | ⋮ |
| 尾数m | 1 | $2^{-22}$ |
| 尾数m | 0 | $2^{-23}$ |

REAL数据类型的值等于$1.\ m\times2^{(e-bias)}$，其中

e：$1\leqslant e\leqslant254$

bias：bias = 127

S：S = 0值为正，S = 1值为负。

例如浮点值12.25的表示方法为：

符号位S = 0；

指数$e=2^7+2^1=128+2=130$；

尾数$m=2^{-1}+2^{-5}=0.5+0.03125=0.53125$

浮点数值 = $(1+0.53125)\times2^{(130-127)}=1.53125\times8=12.25$

西门子公司的PLC规定浮点值的小数部分最多为6位数，如果相差大于等于$10^7$的两个浮点数进行运算，可能导致不正确的结果，例如100000000.0 + 1.0 = 100000000.0，因为值1.0在前者中无法表示（最小数值分辨率）。为了增加浮点运算的准确性，在程序中避免相差大于$10^7$的两个浮点值进行加减运算。

（4）S5Time（SIMATIC时间）

S7系列PLC中的定时器使用S5 TIME的数据类型，格式为S5T#XH_XM_XS_XMS，其中H表示为小时；M表示为分钟；S表示为秒；MS表示为毫秒。时间数据以BCD码二进制编码的格式存储于16个位中，例如时基为1s（时间最小变化率为1s），时间值为127s的位图表示方法如图5-6所示。

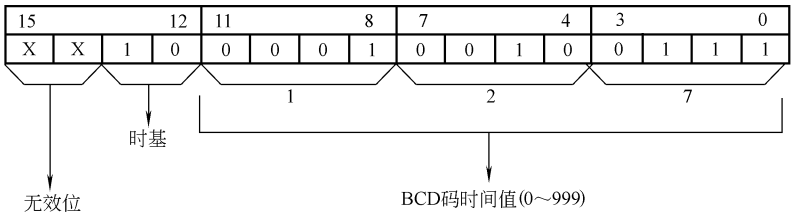



图 5-6 一个 S5 TIME 数据类型的表示方法

时基为时间的最小变化率，时基的几种方式见表 5-3。

**表 5-3 S5 TIME 数据格式的时基**

| 时 基 类 型 | 二进制编码值 |
|---|---|
| 10ms | 00 |
| 100ms | 01 |
| 1s | 10 |
| 10s | 11 |

同样一个定时器，BCD 码的时间值最大为 999，通过选择不同的时基可以改变定时器的定时长度，10ms 时基的最大定时长度为 9990ms，100ms 时基的最大定时长度为 99900ms，1s 时基的最大定时长度为 999s，10s 时基的最大定时长度为 9990s，所以定时器最大的定时长度为 9990s（2H_46M_30S），但是最小分辨率将变为 10s。在编写用户程序时可以直接装载设定的时间值，CPU 根据时间值大小自动选择时基值，例如在程序中设定时间值为 S5T#2M_30S，等于 150s，大于 100ms 时基最大的定时长度而小于 1s 时基最大的定时长度，时基自动选择为 1s；如果选择时间值为一个变量，则需要对时基值进行赋值，例如使用不支持 SIMATIC S5 TIME 数据格式的第三方 HMI 监控软件设定时间值时，需要设定时基值（西门子公司的 WinCC 软件支持 SIMATIC S5 TIME 数据格式，不需要选择）。

（5）TIME（IEC 时间）

TIME（IEC 时间）采用 IEC 标准的时间格式，占用 32 个位，格式为：T#XD_XH_XM_XS_XMS，其中 D 表示为天；H 表示为小时；M 表示为分钟；S 表示为秒；MS 表示为毫秒。在规定的取值范围内，TIME（IEC 时间）类型数据可以与 DINT 类型的数据相互转换（T#0ms 对应 L#0），DINT 数据每增加 1，时间值增加 1ms。与 SIMATIC S5 TIME 相比，没有时基，定时时间更长，但是每一个 IEC 时间定时器需要占用 CPU 的存储区。

（6）DATE（IEC 日期）

DATE（IEC 日期）采用 IEC 标准的日期格式，占用 16 个位，例如 2006 年 8 月 12 日的表示格式为：D#2006 - 08 - 12，按年 - 月 - 日排序。在规定的取值范围内，DATE（IEC 日期）类型数据可以与 INT 类型的数据相互转换（D#1990 - 01 - 01 对应 0），INT 数据每增加 1，日期值增加 1 天。

（7）Time_OF_DAY

Time_OF_DAY（时间），占用 32 个位，例如 10h11min58s312ms 的表示格式为：TOD#10:11:58.312，按时:分:秒:毫秒排序。在规定的取值范围内，Time_OF_DAY（时间）类型

数据可以与 DINT 类型的数据相互转换（TOD#00:00:00.000 对应 0），DINT 数据每增加 1，时间值增加 1ms。

**注意：**

上述介绍的为 S7 CPU 的数据类型，如果与第三方数据通信，需要注意数据类型的构成，如一些串口设备自定义浮点数据类型，与 S7 CPU 的数据类型不匹配，需要用户编程转换。

### 5.1.2　复合数据类型

复合数据类型中的数据是由基本数据类型的数据组合而成或者长度超过 32 位的数据类型。STEP7 中可以有 DATE_AND_TIME、STRING、ARRAY、STRUCT、UDT（User-Defined data Types）、FB 和 SFB 等复合数据类型。

（1）DATE_AND_TIME（时钟）

DATE_AND_TIME 数据类型表示时钟信号，数据长度为 8 个字节（64 位），分别以 BCD 码的格式表示相应的时间值，如时钟信号为 1993 年 12 月 25 日 8 点 12 分 34 秒 567 毫秒存储于 8 个字节中，每个字节代表的含义见表 5-4。

**表 5-4　DATE_AND_TIME 数据类型中每个字节的含义**

| 字　节　数 | 含义及取值范围 | 示例（BCD 码） |
|---|---|---|
| 0 | 年（1990～2089） | B#16#93 |
| 1 | 月（1～12） | B#16#12 |
| 2 | 日（1～31） | B#16#25 |
| 3 | 时（00～23） | B#16#8 |
| 4 | 分（00～59） | B#16#12 |
| 5 | 秒（00～59） | B#16#34 |
| 6 | 毫秒中前两个有效数字（0～99） | B#16#56 |
| 7（高 4 位） | 毫秒中第三个有效数字（0～9） | B#16#7 |
| 7（低 4 位） | 星期：（1～7）<br>1 = 星期日<br>2 = 星期一<br>3 = 星期二<br>4 = 星期三<br>5 = 星期四<br>6 = 星期五<br>7 = 星期六 | B#16#5 |

通过调用 IEC 标准函数块，可以将 DATE_AND_TIME 时间类型的数据与基本数据类型的数据相转换，如：

1）转换 DATE 和 TIME OF DAY 值到 DATE_AND_TIME 值需要调用 FC3：D_TOD_DT；

2）从 DATE_AND_TIME 数据中选取 DATE 值需要调用 FC6：DT_DATE；

3）从 DATE_AND_TIME 数据中选取 TIME OF DAY 值需要调用 FC8：DT_TOD。

（2）STRING（字符串）

STRING 字符串最大长度为 256 个字节，前两个字节存储字符串长度信息，所以最多包

含 254 个字符，其常数表达形式为由两个单引号包括的字符串，例如‘SIAMTIC S7’。STRING 字符串第一个字节表示字符串中定义的最大字符长度，第二个字节表示当前字符串中有效字符的个数，从第三个字节开始为字符串中第一个有效字符（数据类型为“CHAR”），例如定义为最大 4 个字符的字符串 STRING［4］中只包含两个字符‘AB’，实际占用 6 个字节，字节排列如图 5-7 所示。

| 字节0 | 字节1 | 字节2 | 字节3 | 字节4 | 字节5 |
|---|---|---|---|---|---|
| 4 | 2 | ‘A’ | ‘B’ | | |

图 5-7　STRING 字符串数据类型数据排列

（3）ARRAY（数组）

由相同数据类型的数据组成数组。数组的维数最大可以到 6 维；数组中的元素可以是基本数据类型或者复合数据类型中任意数据类型（Array 类型除外，即数组类型不可以嵌套）；例如：Array［1..3，1..5，1..6］INT，定义了一个元素为整数，大小为 3×5×6 的三维数组，可以使用索引（［2，2］）访问数组中的数据，数组中每一维的索引取值范围是 -32768 ~32767，但是索引的下限必须小于上限，例如 1..2、-5..-1 都是合法的定义，索引值按偶数占用 CPU 存储区空间，例如一个单元为字节的数组 ARRAY［1..21］，数组中只有 21 个字节，实际占用 CPU 22 个字节。定义一个数组时，需要指明数组的元素类型、维数和每一维的索引范围，可以用符号名加上索引来引用数组中的某一个元素，例如 a［1，2，3］，也可以通过地址直接访问。

与其他高级语言相比，索引必须为常数，不能作为变量间接寻址，只能通过使用指针间接寻址，这样必须清楚数组元素的排列顺序及绝对地址，例如一个元素为字节的两维数组 ARRAY［1..2，1..3］在数据块中的排列顺序及绝对地址如图 5-8 所示。

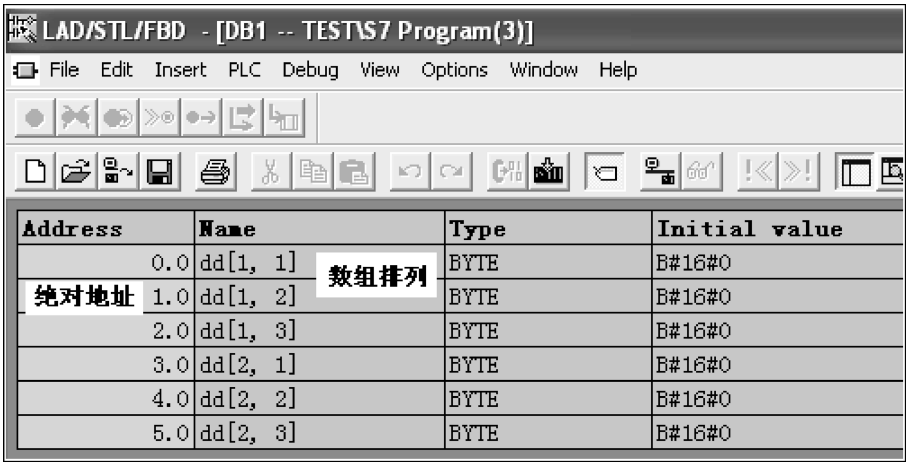
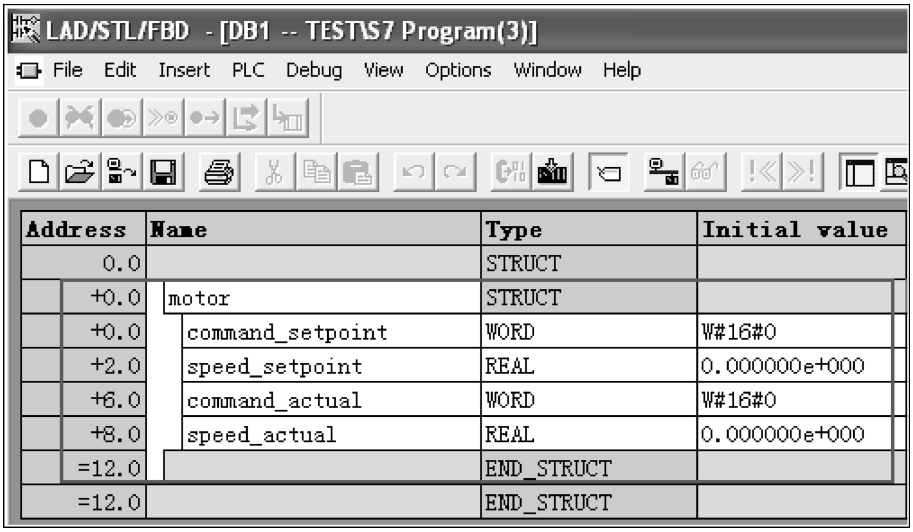



图 5-8　数组单元的排列顺序和绝对地址

（4）STRUCT（结构体）

结构体是由不同数据类型组成的复合型数据，通常用来定义一组相关的数据，例如在数据块 DB1 中定义电动机的一组数据，如图 5-9 所示（框中为结构体中的元素及绝对地址）。



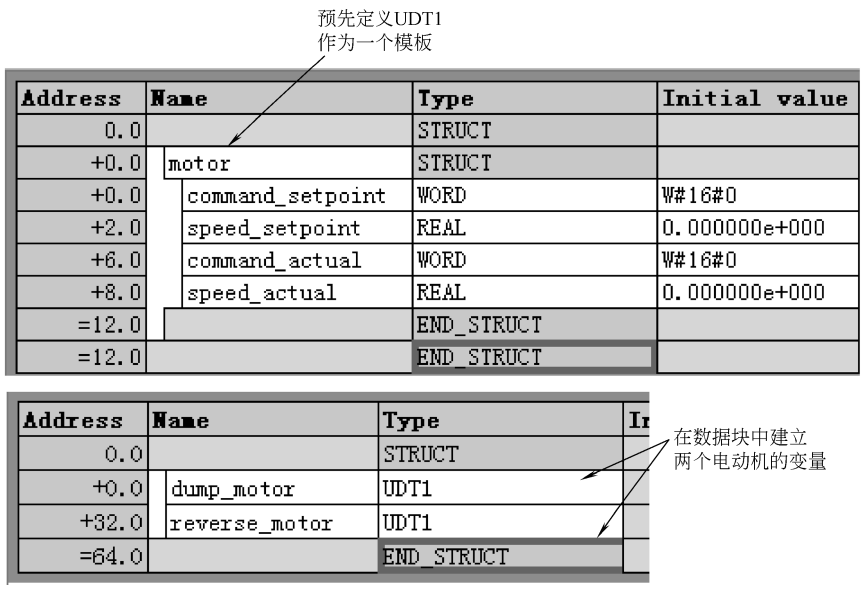

| Address | Name | Type | Initial value |
|---|---|---|---|
| 0.0 | | STRUCT | |
| +0.0 | motor | STRUCT | |
| +0.0 | command_setpoint | WORD | W#16#0 |
| +2.0 | speed_setpoint | REAL | 0.000000e+000 |
| +6.0 | command_actual | WORD | W#16#0 |
| +8.0 | speed_actual | REAL | 0.000000e+000 |
| =12.0 | | END_STRUCT | |
| =12.0 | | END_STRUCT | |

图 5-9　结构体变量的定义和绝对地址

如果引用整个结构体变量，可以直接填写符号地址，例如“DB1. motor”，如果引用结构体变量中的一个单元例如“command_setpoint”，可以直接访问绝对地址如：DB1. DBW0，也可以使用符号名访问如：“DB1. motor. command_setpoint”。

（5）UDT（用户自定义数据类型）

UDT 与 STRUCT 数据类型的定义相同，可以由不同的数据类型组成，如基本数据类型和复合数据类型，与 STRUCT 不同的是，UDT 是一个用户自定义数据类型模板，作为一个整体的变量可以多次调用。例如，在 S7 程序的 Blocks 目录下点击右键，在弹出的菜单中，选择“Insert New Object”→“Data Type”新建一个 UDT 块，在 UDT 中定义图 5-9 中的 motor数据结构作为一个模板，然后在数据块或程序块的形参中插入多个已经定义的 UDT，可以定义不同电动机的变量，如图 5-10 所示。




| Address | Name | Type | Initial value |
|---|---|---|---|
| 0.0 | | STRUCT | |
| +0.0 | motor | STRUCT | |
| +0.0 | command_setpoint | WORD | W#16#0 |
| +2.0 | speed_setpoint | REAL | 0.000000e+000 |
| +6.0 | command_actual | WORD | W#16#0 |
| +8.0 | speed_actual | REAL | 0.000000e+000 |
| =12.0 | | END_STRUCT | |
| =12.0 | | END_STRUCT | |

| Address | Name | Type | Ir |
|---|---|---|---|
| 0.0 | | STRUCT | |
| +0.0 | dump_motor | UDT1 | |
| +32.0 | reverse_motor | UDT1 | |
| =64.0 | | END_STRUCT | |

图 5-10　UDT 的使用

**注意：**

上述复合数据类型只能在数据块（DB）和本地数据区（L）中建立，对于超出 4 个字节的数据，必须以符号名的方式引用，例如一个数组类型数据包含 10 个元素，每个元素为字节，首先在 DB1 中按要求建立一个数组类型数据，10 个元素，每个元素为字节，符号名称为“ARR”，赋值时必须填写 DB1. ARR，数组中的数据可以按单字节变量处理。DATE_AND_TIME、STRING、STRUCT 数据类型变量与数组类型变量处理方式相同（通过指令只能对基本数据类型的数据进行操作）。

（6）FB 和 SFB（函数块和系统函数块）

这种数据类型仅可以在 FB 的静态变量区作定义，可以将多次调用 FB 而生成的多个背景数据传送到一个背景数据块中（多背景 DB）。

### 5.1.3 参数类型

参数数据类型是专用于 FC（函数）或者 FB（函数块）的接口参数的数据类型。可以包括以下几种接口参数的数据类型：

（1）Timer，Counter（定时器和计数器类型）

在 FC、FB 中定义定时器和计数器只有程序块调用时才执行，如果将定时器和计数器定义为形参，对应的实参必须为定时器（T）和计数器（C），数据类型的表示方法与基本数据类型中的定时器（T）和计数器（C）相同。

（2）BLOCK_FB，BLOCK_FC，BLOCK_DB，BLOCK_SDB（块类型）

将定义的程序块作为输入输出接口，参数的声明决定程序块的类型，如 FB、FC、DB 等，如果将块类型作为形参，赋实参时必须为相应的程序块，如 FC101（也可以使用符号地址）。

（3）Pointer（6 字节指针类型）

一个指针只包含地址而不是实际值，将指针数据类型作为形参时，赋实参时必须定义一个地址，实参可以是一个简单的地址，如 M50. 0，也可以是指针格式指向地址的开始，如 P#M50. 0。

（4）Any（10 字节指针类型）

如果实参是未知的数据类型或任意的数据类型时，可以选择“ANY”类型。

使用这些参数类型，可以把定时器、计数器、程序块、数据块，甚至不确定类型和长度的数据通过参数传递给 FC 和 FB。参数类型为程序提供了很高的灵活性，可以实现更通用的控制功能。

参数传递时允许的数据类型将在有关 FC、FB 的章节中做详细介绍，“Pointer”与“Any”的使用将在指针的章节中做详细介绍。

## 5.2 S7-300/400 系列 PLC 地址区

### 5.2.1 CPU 地址区的划分及寻址方法

S7 CPU 的存储器中划分为不同的地址区，在程序中通过指令，可以直接访问存储于地址区的数据。地址区包括过程映像输入区（I）、过程映像输出区（Q）、标志位存储区

（M）、计数器（C）、定时器（T）、数据块（DB）、本地数据区（L）、外设地址输入区（PI）和外设地址输出（PQ），地址区可访问的单位及表示方法见表5-5。

**表5-5　S7-300/400系列PLC地址区**

| 地址区域 | 可以访问的地址单位 | S7符号及表示方法（IEC） |
|---|---|---|
| 过程映像输入区 | 输入（位） | I |
| | 输入（字节） | IB |
| | 输入（字） | IW |
| | 输入（双字） | ID |
| 过程映像输出区 | 输出（位） | Q |
| | 输出（字节） | QB |
| | 输出（字） | QW |
| | 输出（双字） | QD |
| 标志位存储区 | 存储器（位） | M |
| | 存储器（字节） | MB |
| | 存储器（字） | MW |
| | 存储器（双字） | MD |
| 定时器 | 定时器（T） | T |
| 计数器 | 计数器（C） | C |
| 数据块 | 数据块，用“OPN DB”打开 | DB |
| | 数据位 | DBX |
| | 数据字节 | DBB |
| | 数据字 | DBW |
| | 数据双字 | DBD |
| | 数据块，用“OPN DI”打开 | DI |
| | 数据位 | DIX |
| | 数据字节 | DIB |
| | 数据字 | DIW |
| | 数据双字 | DID |
| 本地数据区 | 局部数据位 | L |
| | 局部数据字节 | LB |
| | 局部数据字 | LW |
| | 局部数据双字 | LD |
| 外设地址（I/O）输入 | 外设输入字节 | PIB |
| | 外设输入字 | PIW |
| | 外设输入双字 | PID |
| 外设地址（I/O）输出 | 外设输出字节 | PQB |
| | 外设输出字 | PQW |
| | 外设输出双字 | PQD |

（1）过程映像输入区（I）

过程映像输入区位于 CPU 的系统存储器，在循环执行用户程序之前，CPU 首先扫描输入模块的信息，并将这些信息记录到过程映像输入区中，与输入模块的逻辑地址相匹配。使用过程映像输入区的好处是在一个程序执行周期中保持数据的完整性。使用地址标识符“I”（不分大小写）访问过程映像输入区，如果在程序中访问输入模块中一个输入点，在程序中表示方法如图 5-11 所示。

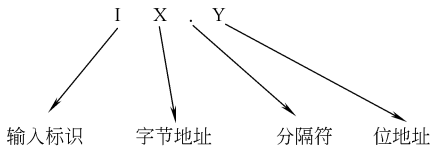



图 5-11 输入模块地址表示方法

一个字节包含 8 个位，所以位地址的取值范围为 0～7，一个输入点即为一个位信号，如果一个 32 点的输入模块设定的逻辑地址为 8，那么第 1 个点的表示方法为 I8.0；第 10 个点的表示方法为 I9.1；第 32 个点的表示方法为 I11.7。按字节访问地址表示方法为 IB8、IB9、IB10、IB11（B 为字节 BYTE 的首字母）；按字访问表示方法为 IW8、IW10（W 为字 WORD 的首字母）；按双字访问表示方法为 ID8（D 为双字 DOUBLE WORD 的首字母）。

（2）过程映像输出区（Q）

过程映像输出区位于 CPU 的系统存储器，在循环执行用户程序中，CPU 将程序逻辑输出的值存放在过程映像输出区，在一个程序执行周期时，开始更新过程映像输出区并将所有的输出值发送到输出模块，保证输出模块输出的一致性。在 S7-300 系列 PLC 中，过程映像输出区固定为 128 个字节，在 S7-400 系列 PLC 中，过程映像输出区的大小可以设置，过程映像输出区与输出模块的逻辑地址应相匹配。

使用地址标识符“Q”（不分大小写）访问过程映像输出区，在程序中表示方法与输入信号相同，输入模块与输出模块分别属于两个地址区，模块逻辑地址可以相同。

（3）标志位存储区（M）

标志位存储区位于 CPU 的系统存储器，地址标识符为“M”，数据区的大小与 CPU 的类型有关，早期的 S7-300 系列 PLC 的标志位存储区只有 256 个字节，新的 S7-CPU315-2DP（2AG10）的标志位存储区为 2048 个字节，CPU412-2DP 的标志位存储区为 4096 个字节。在程序中，访问标志位存储区的表示方法与输入信号相同。M 区中，掉电保持的数据区大小可以在 CPU 中设置。

（4）定时器（T）

定时器存储区位于 CPU 的系统存储器，地址标识符为“T”，定时器的数量与 CPU 的类型有关，定时器的表示方法为 T X，T 表示定时器标识符，X 表示第几个定时器，每一个定时器占用一个字的存储空间。存储区中掉电保持的定时器个数可以在 CPU 中设置。

（5）计数器（C）

计数器存储区位于 CPU 的系统存储器，地址标识符为“C”，定时器的数量与 CPU 的类型有关，计数器的表示方法为 C X，C 表示计数器的标识符，X 表示第几个计数器，每一个计数器占用一个字的存储空间。存储区中掉电保持的计数器个数可以在 CPU 中设置。

**注意：**

如果在程序中访问 M 区、定时器、计数器地址超出 CPU 规定地址区范围，则 CPU 将报错，并提示程序不能下载。

（6）数据块存储区（DB）

数据块可以存储于装载存储器、工作存储器以及系统存储器中（块堆栈），共享数据块地址标识符为“DB”，函数块FB、SFB的背景数据块地址标识符为“DI”。以标识符DB为例，按位访问DB区的表示方法为DB1. DBX20. 0（第一个数据块中，第21个字节的第一位，X表示位信号）；按字节访问DB区的表示方法为DB1. DBB20（第一个数据块中，第21个字节，B为字节BYTE的首字母）；按字访问DB区的表示方法为DB1. DBW8（W为字WORD的首字母）；按双字访问DB区的表示方法为DB1. DBD8（D为双字DOUBLE WORD的首字母）。数据块的个数与数据块存储区的大小与CPU类型有关，每一个数据块的大小与使用S7-300CPU或S7-400CPU有关，通常S7-300系列PLC数据块的容量为32K，S7-400系列PLC数据块的容量为64K。数据块中的数据默认设置为掉电保持，不需要额外设置。

**注意：**

*在语句表编程中，通过“DB”或“DI”区分两个打开的数据块，在其他应用中，函数块FB、SFB的背景数据块也可以使用“DB”表示。*

（7）本地数据区（L）

本地数据区位于CPU的系统数据区，地址标识符为“L”，包括函数、函数块的临时变量、组织块中的开始信息、参数传送信息及梯形图编程的内部逻辑结果。在程序中访问本地数据区的表示方法与输入信号相同。

在S7-300系列PLC中，每一类优先级组织块及其程序调用相关函数所占用最大的本地数据区固定为256个字节，例如在主程序OB1中调用函数FC1和FC2，三者使用总的临时变量不能超过256个字节；在S7-400系列PLC中，每一类优先级组织块及其程序调用相关函数所占用的本地数据区大小可以设置，CPU可以使用的本地数据区大小与CPU的类型有关，如CPU414-3的本地数据区为16K。

**注意：**

*如果使用本地数据区超出CPU的限制，则CPU将停机。*

（8）外设地址输入区（PI）

外设地址输入区位于CPU的系统数据区，地址标识符为“PI”，与过程映像区功能相反，不经过过程映像区的扫描，程序访问外设地址区时，直接将输入模块当前的信息读入并作为逻辑运算的条件，例如在程序中直接读出模拟量输入的信息等，同样一个模块，逻辑地址的设定决定该模块数据信息的读取方式，例如S7-300系列PLC过程映像输入区为128个字节，如果设定模块逻辑地址大于128个字节，那么该模块只能通过外设输入区读取。访问外设输入地址区最小单位为字节，例如访问一个字节表示方法为PIB X（B为字节BYTE的首字母，X为外设地址区），访问一个字表示方法为PIW X（W为字WORD的首字母，X为外设地址区），访问一个双字表示方法为PID X（D为双字DOUBLE WORD的首字母，X为外设地址区）。

（9）外设地址输出区（PQ）

外设地址输出区位于CPU的系统数据区，地址标识符为“PQ”，与外设地址输入区的访问方式相同，访问字节、字和双字的表示方法为PQB X、PQW X和PQD X（X为外设地址区）。

**注意：**

*在过程映像区内，I/O地址的数据也可以直接使用，立即读（PI）或立即写（PQ）的*

方式为直接访问，访问最小单位为字节。

### 5.2.2 全局变量与区域变量

（1）全局变量可以在 CPU 范围内被所有的程序块调用，例如在 OB（组织块）、FC（函数）、FB（函数块）中使用，在某一个程序块中赋值后，在其他的程序中可以读出，没有使用限制。全局变量包括 I、Q、M、定时器（T）、计数器（C）、数据块（DB）、PI、PQ 等数据区。

（2）区域变量只能在所属的程序块（OB、FC、FB）范围内调用，在程序块调用时有效，程序块调用完成后被释放，所以不能被其他程序块使用，本地数据区（L）中的变量为区域变量，例如每个程序块中的临时变量都属于区域变量。

### 5.2.3 地址区数据的排列

一个双字包括两个字，一个字又包括两个字节，一个字节包括 8 个位，它们之间的排列关系如图 5-12 所示（以 MD20 为例）。




图 5-12 位寻址、字节寻址、字寻址和双字寻址之间的关系

注意数据的排列次序，序号低的字节或字为高字节或高字，例如将 123 存放于 MD20 中，数据先存放于 MB23 中，如果数据超过一个字节的上限 255 后，将数据高位存放在 MB22 中，依此类推。如果在程序中使用字变量，如 MW20、MW21，或双字变量，如 MD20、MD22，从图 5-12 中可以看出每两个变量中都有相互重叠的部分，这样将在程序中造成数据冲突，影响程序的执行，为避免数据区使用冲突与重叠，使用字节时，建议地址按 1 的倍数增加；使用字时，建议地址为偶数并按 2 的倍数增加；使用双字时，建议地址为偶数并按 4 的倍数增加。

S7 系列 PLC 与其他设备进行数据通信时，应注意字节的排列顺序，例如在 PLC 中，一个 WORD 的值为 W#16#1234，在通信方读出的数据可能为 W#16#3412，这是由于 WORD 数据中高低字节颠倒所造成的，需要在通信方作处理或在本地 PLC 中编程转换。

# 第6章　编程指令

STEP7 集成梯形图（Ladder Logic Programming Language，LAD）、语句表（Statement List Programming Language，STL）和功能块图（Function Block Diagram Programming Language，FBD）三种编程语言，不同的编程语言可以为具有不同知识背景的编程人员提供多种选择：

（1）LAD

梯形图和继电器原理图类似，采用诸如触点和线圈等符号。这种编程语言适合于对继电器控制电路比较熟悉的技术人员，各个厂商 PLC 都具有梯形图语言，上手快，编程指令可以直接从指令集窗口中拖放到程序中使用。LAD 程序可以转换成 STL 语言。

（2）STL

语句表包含丰富的 STEP7 指令，采用文本编程的方式，适合熟悉高级编程语言的人员使用，简化编程量，但是没有助记功能，编程指令必须事先了解或从在线帮助中查询。由于 STL 编程简练，绝大多数程序不能转换成 LAD 程序（如果按照 LAD 格式编写，可以转换）。

（3）FBD

功能块图使用不同的功能“盒”相互搭接成一段程序，逻辑采用“与”、“或”、“非”进行判断。与梯形图相似，编程指令也可以直接从指令集窗口中拖放出来使用，大部分程序可以与梯形图程序相互转换。

任何一种编程语言都有相应的指令集，指令集是最基本的编程元素，通过指令集可以调用使用基本指令编写的函数和函数块、系统函数和函数块以及 STEP7 提供的库函数。三种编程语言指令集的对比如图 6-1 所示。

LAD 和 STL 在编程中比较常见，本书将着重介绍这两种编程语言的指令集。

## 6.1　指令的处理

### 6.1.1　LAD 指令处理

LAD 程序逻辑处理中以能流的方式从左向右进行传递，如图 6-2 所示，位信号 M1.1 和 M1.2 相“与”，结果和位信号 M1.3 相“或”，并将逻辑执行结果传递到输出线圈 M1.4，图 6-2 中位信号 M1.1 和 M1.2 信号为 1，处于导通状态，将能流传递给 M1.4，触发线圈的输出。

LAD 程序中逻辑运算、比较等指令也都是由位信号触发的，在这些指令中，左边输入端为“EN”使能信号，如果使能信号为 1，指令执行，如果条件满足触发输出信号“ENO”，如图 6-3 所示，位信号 M1.1 为 1，触发执行“CMP = I”比较指令，变量 MW2 不等于 MW4，没有将能流传递到输出线圈 M1.4。

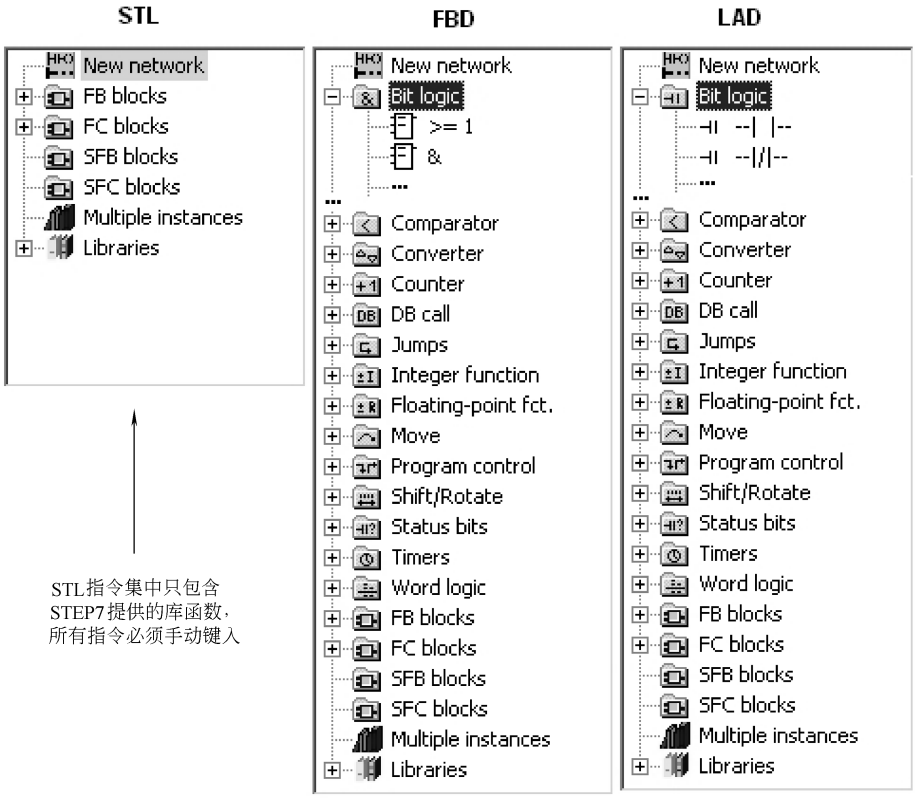



图 6-1　STL、FBD、LAD 指令系统

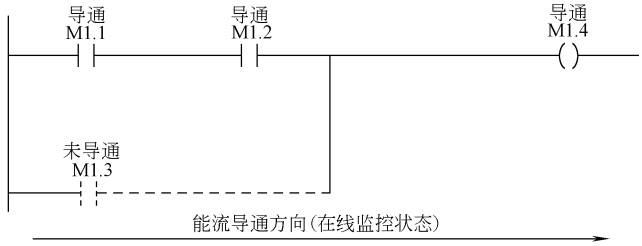



图 6-2　LAD 逻辑处理能量流向

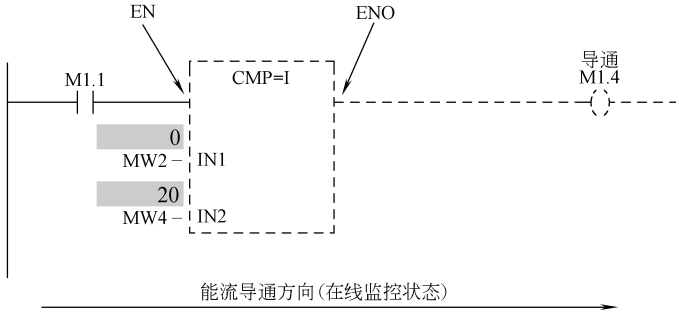
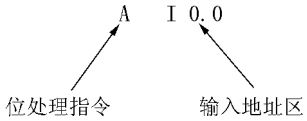



图 6-3　LAD 运算处理能量流向

## 6.1.2　STL 指令处理

1. 状态字的使用

与 LAD 指令相比，STL 使用指令直接对地址区进行操作，例如位处理的“与”指令：




操作指令在前地址区在后，通过指令直接对地址操作，STL 指令的监控通过 CPU 内部寄存器中的状态字实现，状态字中的前 9 位分别表示地址当前状态、逻辑处理结果、数据溢出等操作状态，状态字的结构如图 6-4 所示。

状态字:

| 第八位 | 第七位 | 第六位 | 第五位 | 第四位 | 第三位 | 第二位 | 第一位 | 第零位 |
|---|---|---|---|---|---|---|---|---|
| BR | CC 1 | CC 0 | OV | OS | OR | STA | RLO | /FC |

图 6-4　CPU 寄存器中的状态字

状态字中每一个位的作用及含义如下：

（1）/FC

首次检测位，控制逻辑操作的开始。每一个逻辑操作都需要查询/FC 位的状态和被操作的逻辑地址状态，如果/FC 位为 1，与检测逻辑地址状态相结合，并将结果存储于 RLO 位；如果/FC 位为 0，开始新的逻辑操作，例如赋值指令（S，R，=）或基于 RLO 的跳转指令，都会结束逻辑的操作，并将/FC 位复位。

（2）RLO

存储逻辑操作或比较指令的结果，RLO 在程序调试中很重要，赋值指令根据 RLO 的状态判断线圈是否输出。

（3）STA

存储一个位地址的值。

状态位/FC、RLO、STA 的关系可以参考图 6-5 的示例程序。

在上面的示例程序中，根据状态字可以判断每个信号的状态，M1.1 = 0；M1.2 = 1；M1.3 = 0；输出结果为 0，赋值指令复位/FC 位，由于 M1.1 的 RLO 位为 0，即使 M1.2 的 STA 位为 1，与上面指令的逻辑结果 RLO 仍然为 0，如果 M1.1 为 1，三条“与”指令的逻辑结果 RLO 为 1，赋值指令根据 RLO 输出。

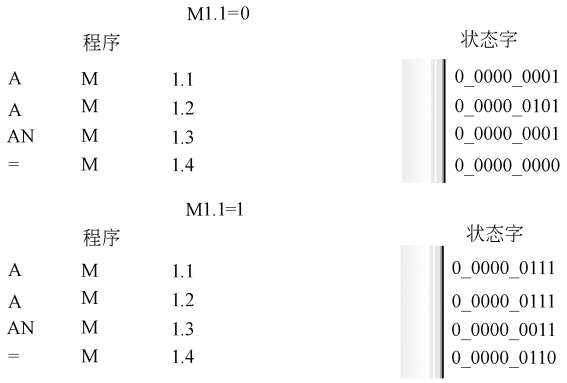



图 6-5　状态位/FC、RLO、STA 的关系

（4）OR

使用“0”指令结合“AND”指令，如图 6-6 所示，将 M1.1、M1.2 相“与”的结果再

与 M1.3“或”，如果“0”指令前 M1.1、M1.2 相“与”的逻辑结果为 1，OR 位为 1。

（5）OS

如果浮点运算出现问题，触发 OV 位并置位 OS 位，故障消除，OS 位保持，所以 OS 位可以记录 OV 位是否出过故障。JOS（OS 为 1 跳转）、块调用、块结束指令自动复位 OS 位。

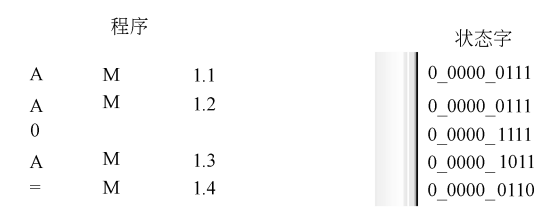



图 6-6 状态位 OR

（6）OV

溢出位，浮点运算故障（溢出、非法操作、无序比较），OV 位置位，故障消除，OV 位复位。

（7）CC1，CC0

条件代码，指示位逻辑操作、比较指令、算数指令、移位指令等操作状态，也表示累加器 1、2 的关系，例如累加器 2 中的值大于累加器 1 中的值，通过 CC0、CC1 的状态可以触发程序跳转指令等。

（8）BR

二进制结果位，解释字逻辑的结果，例如两个字相“与”的结果。另一方面，在编写函数或函数块时，如果没有将 RLO 位存储到 BR，使用 LAD 调用时，不能激活函数输出“ENO”（不导通状态），程序如图 6-7 所示，调用用户编写的函数 FC1 时，函数执行，但是 ENO 没有输出，能流没有导通。在函数的结尾，简单使用 SAVE 指令可以显示 ENO 的状态。

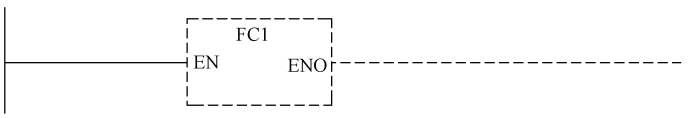



图 6-7 状态位 BR

状态字中的位信号，可以在程序中使用指令进行操作，如调用 SET、CLR 指令直接对 RLO 位进行操作。在调试阶段，可以根据状态位的状态调试程序，如使用 STA、RLO 位的信号可以判断操作地址的状态和指令处理结果，状态字是调试 STL 程序不可缺少的工具。

2. 累加器的使用

对于运算指令，STL 使用累加器作为数据的缓存区，累加器属于 CPU 内部寄存器，S7-300 CPU 中有两个累加器（ACCU1 和 ACCU2），S7-400 CPU 中有 4 个累加器（ACCU1、ACCU2、ACCU3、ACCU4），每个累加器占用 32 位地址空间，可以将 4 个字节的变量放置在累加器中进行运算，累加器的使用参考下面的示例程序：

```
L     MD    10      //装载变量 MD10 进入累加器 1
L     12.1          //装载浮点值 12.1 进入累加器 1，MD10 进入累加器 2
+R                  //累加器 1 与累加器 2 中的值相加，将结果传送到累加器
                      1 中
T     MD    14      //将累加器 1 中存储的运算结果传送到变量 MD14
```

累加器 1 和 2 中的数据通过“L”指令自动堆栈。累加器的使用可以节省大量中间计算数据的存储空间，LAD 没有累加器的概念，中间计算数据必须占用数据区进行转存。

两个累加器往往不能满足复杂的运算，S7-400 系列 PLC 提供 4 个累加器作为数据的缓存区可以满足复杂计算的要求，累加器 ACCU3 和 ACCU4 必须使用特殊指令进行操作。

## 6.2　位逻辑指令

### 6.2.1　触点指令

STEP7 中提供的触点指令见表 6-1，在触点指令中分为标准触点指令、取反指令和沿检测指令。

**表 6-1　触点指令**

| | LAD | 说　明 | STL | 说　明 |
|---|---|---|---|---|
| 触点指令 | ---\| \|--- | 常开触点（地址） | A | “与”操作 |
| | ---\|/\|--- | 常闭触点（地址） | A（ | “与”操作嵌套开始 |
| | ---\|NOT\|--- | 信号流反向 | AN | 与非操作 |
| | ---（N）--- | RLO 下降沿检测 | AN（ | 与非操作嵌套开始 |
| | ---（P）--- | RLO 上升沿检测 | O | 或操作 |
| | NEG | 地址下降沿检测 | O（ | 或操作嵌套开始 |
| | POS | 地址上升沿检测 | ON | 或非操作 |
| | | | ON（ | 或非操作嵌套开始 |
| | | | X | 异或操作 |
| | | | X（ | 异或操作嵌套开始 |
| | | | XN | 异或非 |
| | | | XN（ | 异或非嵌套开始 |
| | | | ） | 嵌套闭合 |
| | | | NOT | 非操作（RLO 取反） |
| | | | FN | 下降沿 |
| | | | FP | 上升沿 |

1. 标准触点指令

触点表示一个位信号的状态，地址可以选择 I、Q、M、DB、L 数据区，触点可以是输入信号、程序处理的中间点及与其他站点通信的数据位信号，在 LAD 中常开触点指令为“---| |---”，常闭触点为“---|/|---”，当值为 1 时，常开触点闭合；当值为 0 时，常闭触点闭合；在 LAD 编程时，标准触点间的“与”、“或”、“异或”操作没有相应指令，需要通过标准指令搭接出来；使用 STL 编程，对常开触点使用 A（与）、O（或）、X（异或）指令，对常闭触点使用 AN（与非）、ON（或非）、XN（异或非）指令，两段程序间的逻辑操作，需要使用嵌套符号“（）”。

2. 取反指令

取反指令（---| NOT |---、NOT）改变能流输入的状态，将当前值由 0 变 1，或由 1 变 0。

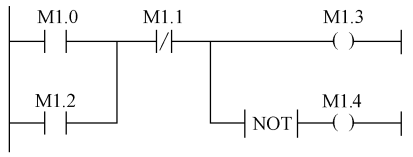
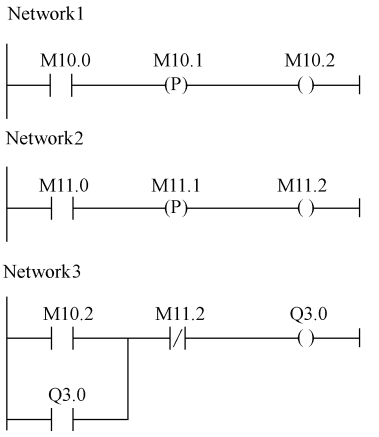
3. 沿检测指令

信号沿的检测分为上升沿检测(---（P）---、FP) 和下降沿检测(---（N）---、FN)，沿信号在程序中比较常见，如电动机起动、停止信号，故障信号的扑捉等都是通过沿信号实现的。上升沿检测指令检测每一次 0 到 1 的正跳变，让能流接通一个扫描周期，下降

沿检测指令检测每一次 1 到 0 的负跳变，让能流接通一个扫描周期。---（ P ）---与 POS、---（ N ）---与 NEG 功能相同，前者检测指令前面 RLO 信号的跳变，后者检测一个位地址的跳变。

使用触点指令的示例程序见 6-2。

**表 6-2 触点指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| 标准触点和取反指令的使用 | | |
|  | A ( <br>O M 1.0<br>O M 1.2<br>)<br>AN M 1.1<br>= M 1.3<br>NOT<br>= M 1.4 | 常开触点 M1.0 与 M1.2 相“或”的结果再与常闭 M1.1 相“与”，逻辑结果赋值给线圈 M1.3，逻辑结果取反赋值给线圈 M1.4，M1.3 和 M1.4 的状态相反。在 STL 程序中程序段间的关系使用嵌套符号“ ()” |
| 沿检测指令的使用 | | |
|  | Network 1<br>A M 10.0<br>FP M 10.1<br>= M 10.2<br>Network 2<br>A M 11.0<br>FP M 11.1<br>= M 11.2<br>Network 3<br>A ( <br>O M 10.2<br>O Q 3.0<br>)<br>AN M 11.2<br>= Q 3.0 | 程序实现最简单的电动机自锁功能，程序段 1、2 中分别检测触点 M10.0、M11.0 的上升沿信号，将检测结果赋值到线圈 M10.2、M11.2。当 M10.0 从 0 到 1 跳变，M10.2 产生脉冲信号使线圈 Q3.0 输出，然后通过自己的触点信号保持输出状态，当 M11.0 从 0 到 1 跳变，M11.2 产生脉冲信号使线圈 Q3.0 断开 |

所有的触点指令不能对外设输入输出区进行操作。例如 A PI0.0 指令为非法。在程序中能流不能反向，“或”操作不能短路，如图 6-8 所示，在这些情况下，STEP7 会自动检查，程序不能进行有效的连接。

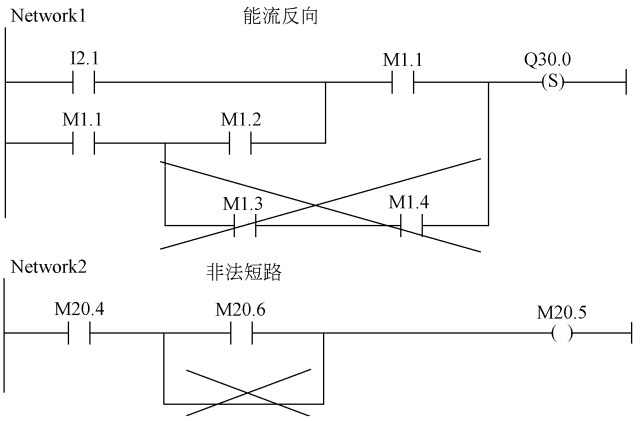



图 6-8 触点编程错误

## 6.2.2 线圈指令

STEP7 中提供的线圈指令见表 6-3，在线圈指令中分为输出指令和置位/复位指令。

**表 6-3　线圈指令**

| | LAD | 说　明 | STL | 说　明 |
|---|---|---|---|---|
| 线圈指令 | ---（ ） | 结果输出/赋值 | = | 赋值 |
| | ---（#）--- | 中间输出 | | |
| | ---（ R ） | 复位 | R | 复位 |
| | ---（ S ） | 置位 | S | 置位 |
| | RS | 复位置位触发器 | | |
| | SR | 置位复位触发器 | | |

1. 线圈输出指令

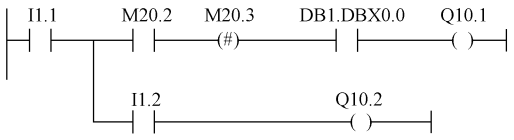
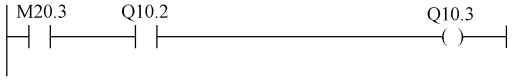
线圈指令对一个位信号进行赋值，地址可以选择 Q、M、DB、L 数据区，线圈可以是输出信号、程序处理的中间点，当触发条件满足（RLO = 1），线圈被赋值 1，当条件再次不满足时（RLO = 0），线圈被赋值 0。在程序处理中，每个线圈可以带有若干个触点（没有限制），线圈的值决定常开触点、常闭触点的状态。在 LAD 中线圈输出指令为“-- ( )”，总是在一个程序段的最右边，如果需要得到逻辑处理的中间状态，可以使用中间输出指令“-（#）-”查询，中间输出指令不能在一个程序段的两段使用；STL 编程中只有赋值指令“ = ”，中间输出指令可以通过编程实现。

2. 置位/复位指令

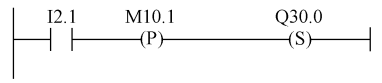
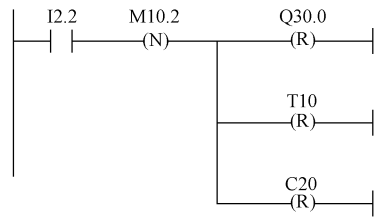
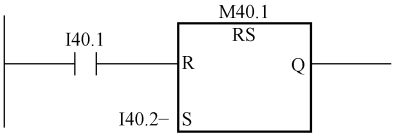
当触发条件满足（RLO = 1），置位指令将一个线圈置 1，当条件再次不满足（RLO = 0），线圈值保持不变，只有触发复位指令才能将线圈值复位为 0。单独的复位指令也可以对定时器、计数器的值进行清零。LAD 编程指令中 RS、SR 触发器带有触发优先级，置位、复位信号同时为 1 时，优先级高的指令触发，如 RS 触发器，S（置位在后）优先级高。STL 编程中没有 RS、SR 触发器，置位、复位的优先级与指令在程序中的位置有关，通常指令在后的优先级高。

使用线圈指令的示例程序见表 6-4。

**表 6-4　线圈指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| 线圈输出指令 | | |
| Network1<br>  <br><br>Network2<br>  <br> | **Network 1**<br>A　I　1. 1<br>A　M　20. 2<br>=　M　20. 3<br><br>A　M　20. 3<br>A　DB1. DBX　0. 0<br>=　Q　10. 1<br><br>A　I　1. 1<br>A　I　1. 2<br>=　Q　10. 2<br>**Network 2**<br>A　M　20. 3<br>A　Q　10. 2<br>=　Q　10. 3 | 在程序段 1 中将地址 I1. 1 和 M20. 2 的“与”结果存储于中间输出 M20. 3 中，再和变量 DB1. DBX0. 0“与”，将结果输出到输出线圈 Q10. 1，同样将地址 I1. 1 和 I1. 2 的“与”结果输出到输出线圈 Q10. 2<br>在程序段 2 中，将线圈 M20. 3 和 Q10. 2 的触点信号相“与”，结果输出到输出线圈 Q10. 3 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 置位、复位指令 | | |
| Network1<br>  <br>Network2<br>  <br>Network3<br>  | **Network 1**<br>A I 2.1<br>FP M 10.1<br>S Q 30.0<br>**Network 2**<br>A I 2.2<br>FN M 10.2<br>R Q 30.0<br>R T 10<br>R C 20<br>**Network 3**<br>A I 40.1<br>R M 40.1<br>A I 40.2<br>S M 40.1 | 在程序段1中，I2.1上升沿置位Q30.0，在程序段2中，只有I2.2产生下降沿才能复位Q30.0。同时I2.2产生的下降沿信号复位定时器T10和计数器C20中的值<br>在程序段3，I40.1和I40.2同时为1时，将置位M40.1 |

所有的线圈指令不能对外设输入、输出区进行操作。例如 = PQ0.0 指令为非法。只有触发条件才能触发输出或置位、复位，例如在 STL 程序中，触发条件为状态字的 RLO 位，为 1 时触发。参考下面的示例程序：

```
Network 1 :
L       1
=       Q       10.2
```

在程序段 1 中，Q10.2 不能被赋值，因为 L 1（装载 1 到累加器 1 中）不能使 RLO 位变为 1，同样参考下面的示例程序：

```
Network 2:
S       Q       10.3
```

在程序段 2，Q10.3 能否被置位，需要参考 RLO 位的状态，如果需要程序无条件置位 Q10.3，可以在置位语句前加上条件，示例程序如下：

```
Network 3:
AN      M       20.0
S       Q       10.3
```

可以将 M20.0 作为一个未用的触点，语句 AN M 20.0 默认的逻辑结果 RLO 为 1，也可

以直接使用 SET 指令将当前的 RLO 置位，示例程序如下：

```
Network 4 :
SET
S          Q       10.3
```

### 6.2.3　RLO 操作指令

在位操作中，还有一些指令可以直接对状态字中的逻辑结果位 – RLO 进行操作，指令见表 6-5。

**表 6-5　RLO 操作指令**

| | LAD | 说　明 | STL | 说　明 |
|---|---|---|---|---|
| RLO 操作指令 | ---（SAVE） | 将 RLO 存入 BR 寄存器 | SAVE | 把 RLO 存入 BR 寄存器 |
| | | | CLR | RLO 清零（=0） |
| | | | SET | RLO 置位（=1） |

–（SAVE）/ SAVE 指令将 RLO 状态存入 BR 寄存器，在前面已经介绍，如果没有存储 BR 位信号，编写的函数或函数块使用 LAD 语言直接调用时，函数的输出 ENO 不使能，函数显示为虚线，如图 6-9 所示，FC1 的 ENO 不输出，M100.1 不能为 1。

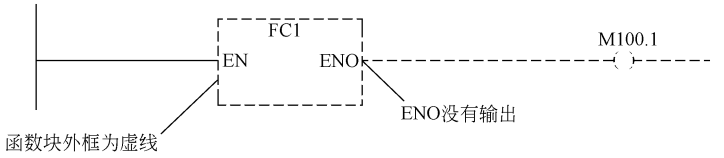



图 6-9　没有将 RLO 位存入 BR 位调用 FC1 显示状态

从程序显示上看，FC1 似乎没有调用，实际已经调用，只是显示问题，在调用 FC1 前加入条件（常开或常闭触点）或在 FC1 的程序结尾使用 SAVE 指令，可以改变 FC1 调用的显示状态，例如在 FC1 结尾加入语句：

```
Network X:
SET
SAVE
```

在程序结尾使用 SAVE 指令后，主程序中调用 FC1 的在线监控状态如图 6-10 所示，FC1 的显示发生变化。

SET 和 CLR 指令只有在 STL 编程中使用，可以将上面的操作结果 RLO 置位或复位，影响线圈指令的输出，示例程序如下：

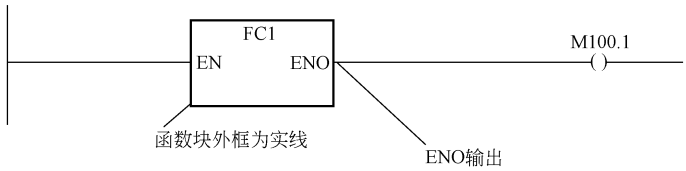



图 6-10　将 RLO 位存入 BR 寄存器后调用 FC1 的显示状态

```
SET
=       M       10.1
=       M       10.2
CLR
=       M       10.3
=       M       10.4
```

语句执行后，M10.1、M10.2 输出为 1，M10.3、M10.4 输出为 0。

### 6.2.4 立即读与立即写

1. 立即读

立即读可以不经过过程映像区的处理，直接读出外设输入地址的信息，例如 16 点的输入模块设定的地址为 10，地址位于过程映像输入区，通常情况下使用输入地址标识符“I”查询输入模块信息，如果 CPU 的扫描时间为 40ms，输入信号的状态需要 40ms 更新一次，使用立即读的方法，不依赖 CPU 的扫描时间，当程序执行到该地址区（使用外设地址地址区 PI 替代 I）时，立即更新输入点信号进行逻辑处理。立即读不考虑输入信号的一致性，着重于输入信号的立即采集，适合有严格时间要求的应用，在程序中可以多次使用立即读访问同一地址区，这样在一个程序执行周期中（一个 CPU 扫描）可以多次更新一个输入模块的状态（使用过程映像区，一个扫描周期只更新一次）。立即读有固定的编程格式，示例程序如图 6-11 所示。

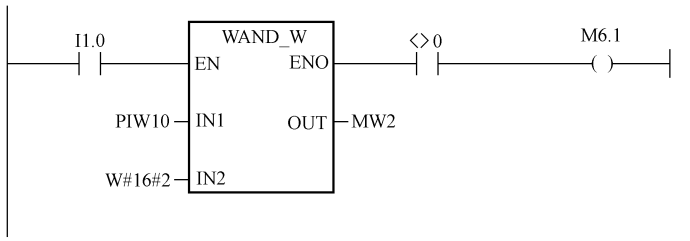



图 6-11 立即读的编程模式

当程序执行 PIW10 时，将输入地址为 10 的 16 点输入模块的信号状态立即读出（外设输入区只能使用字节、字、双字读出），通过 WAND_W（两个字相“与”）指令过滤其他位信号，指令处理如下，

```
PIW10           0000000000101010
W#16#2          0000000000000010
MW2             0000000000000010
```

只对 PIW10 中第二个位信号进行处理，如果 I1.0、第二个位信号为 1，字相“与”的结果不为 0，<>0 导通，赋值 M6.1 为 1。例子为 LAD 程序，可以转换为 STL 程序，在 STL 程序中使用 BR 位判断字逻辑结果。

2. 立即写

立即写与立即读功能相同，可以不经过过程映像区的处理，直接将逻辑结果写到输出地址区。使用立即写不依赖 CPU 的扫描时间，当程序执行到该地址区（使用外设地址地址区 PQ 替代 Q）时，立即更新输出点状态。在程序中可以多次使用立即写功能访问同一地址区，这样在一个程序执行周期中，可以多次更新一个输出模块的状态。立即写的示例程序如图 6-12 所示。

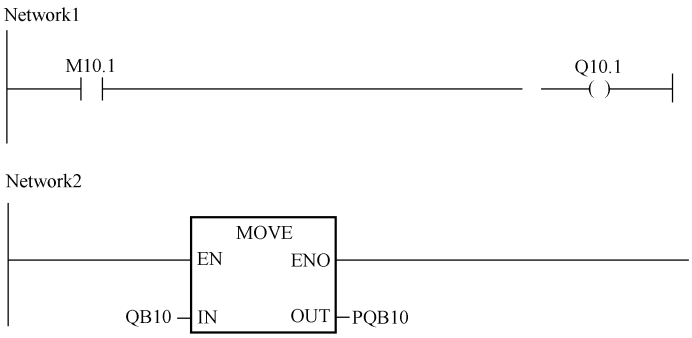



图 6-12　立即写编程

在程序段 1 中，M10.1 为 1 时，只有经过输出过程映像区更新时才能触发 Q10.1 输出（等待一个扫描周期），在程序段 2 中，将 QB10 传送到 PQB10 中，当程序扫描到 PQB10 时立即输出，更新输出模块的状态。

## 6.3　比较指令

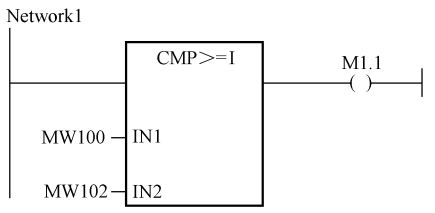
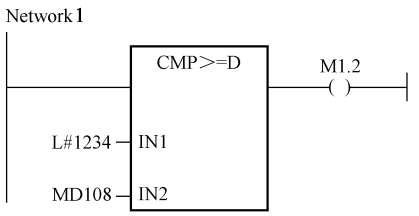
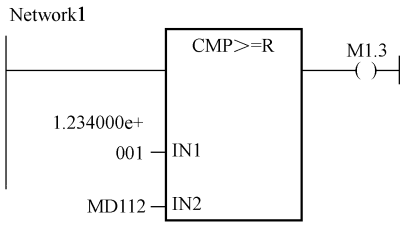
LAD 的比较指令对两个输入参数 IN1 和 IN2 的值进行比较，比较的内容可以是相等、不等、大于、小于、大于等于和小于等于。如果比较结果为真，则逻辑结果为“1”。比较指令有三类，分别用于整数、双整数和浮点；STL 分别将两个值装载到累加器 1 和 2 中，然后将两个累加器进行比较，比较的内容和指令类别与 LAD 相同，但是 STL 编程更灵活，可以将字节间、字节与字、字与双字相比较，使用 LAD 编程时，参数 IN1 和 IN2 的数据类型必须相同。比较指令见表 6-6。

**表 6-6　比较指令**

| | LAD | 说　明 | STL | 说　明 |
|---|---|---|---|---|
| 比较指令 | CMP >= D | 双整数比较<br>==：等于<br><>：不等于<br>>：大于<br><：小于<br>>=：大于等于<br><=：小于等于 | >= D | 双整数比较（32 位）<br>==：等于<br><>：不等于<br>>：大于<br><：小于<br>>=：大于等于<br><=：小于等于 |
| | CMP >= I | 整数比较（==，<>，>，<，>=，<=） | >= I | 整数比较（16 位），>，<，>=，<=，==，<> |
| | CMP >= R | 浮点比较（==，<>，>，<，>=，<=） | >= R | 浮点比较 >，<，>=，<=，==，<> |

使用比较指令的示例程序见表 6-7。

**表 6-7 比较指令的示例程序**

| LAD | STL | 程 序 说 明 |
|---|---|---|
| 整数类型比较指令（以大于等于指令为例） | | |
|  | **Network 1**<br>L MW 100<br>L MW 102<br>>=I<br>= M 1.1 | 在程序段 1 中，如果变量 MW100 大于等于 MW102 时，则输出 M1.1 为 1，IN1 和 IN2 中也可以使用常数，例如在 IN2 中，输入 123；使用 STL 编程，先将 MW100 装入累加器 1 中，再将 MW102 装入累加器 1 中，原先累加器 1 中的变量 MW100 堆栈进入累加器 2 中，如果累加器 2 大于等于累加器 1 中的值，则输出 M1.1 为 1，也可以将常数载入累加器中 |
| 双整数类型比较指令（以大于等于指令为例） | | |
|  | **Network 1**<br>L L#1234<br>L MD 108<br>>=D<br>= M 1.2 | 与程序段 1 中的程序相同，比较的数据类型变为双整数类型，如果常数 1234 大于等于变量 MD108 时，则输出 M1.2 为 1 |
| 浮点类型比较指令（以大于等于指令为例） | | |
|  | **Network 1**<br>L 1.234000e+001<br>L MD 112<br>>=R<br>= M 1.3 | 比较的数据类型为浮点类型，在程序段 1 中，如果常数 12.34 大于等于变量 MD112 时，则输出 M1.3 为 1 |

使用 LAD 编程时，在输入参数 IN1 和 IN2 输入的变量必须完全符合数据类型的要求，例如 CMP >=I 比较指令，输入参数必须为整数类型，如果输入变量 MW100 和 MW102 在符号表中定义数据类型为“WORD”，则在输入变量时，报错不能输入，输入变量变为警示颜色—红色，如图 6-13 所示。

使用 STL 编程，程序相同但是不会提示故障信息，程序如下：

```
L     MW    100
L     MW    102
>=I
=     M     1.1
```

实际上数据已经自动转换为整数类型（带有符号位），例如 MW100 的值为 W#16#8001，

MW102 的值为 W#16#0001，但是不能输出 M1.1，因为 W#16#8001 转换为整数类型后变为 -32767，W#16#0001 转换为整数类型后变为 1，MW100 小于 MW102 不能触发 M1.1 输出，其他数据类型的比较也会自动转换为指定的数据类型。

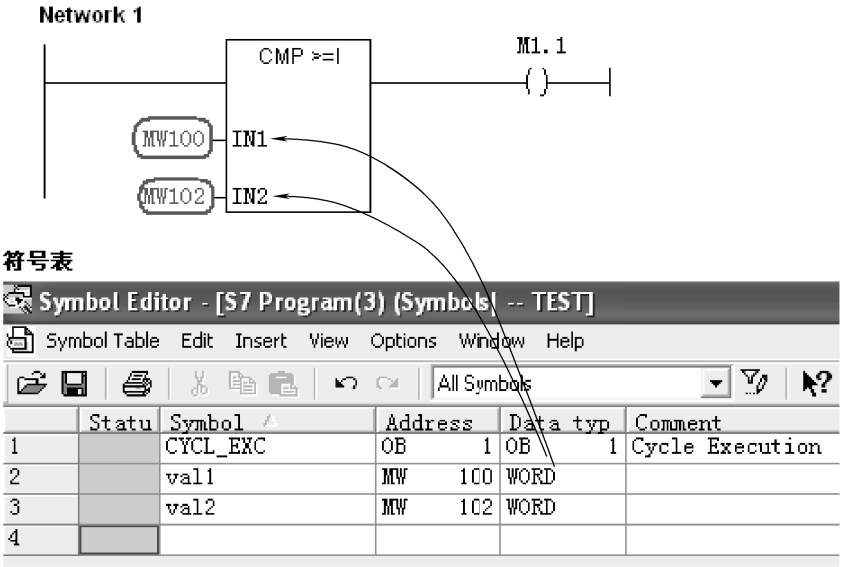



图 6-13　输入数据类型不符

使用 STL 编程时，不同数据类型的变量也可以相比较，参考下面的示例程序：

```
L     MB    100
L     MD    102
>=I
=     M     1.1
```

将 MB100 与 MD102 相比，指定比较的数据类型为整数，实际上将存储于变量 MB100 中的整数值与 MW104（MD102 的低字）中的整数值相比较。

**注意：**

在实际编程中，最好使用相同类型的数据进行比较。

## 6.4　转换指令

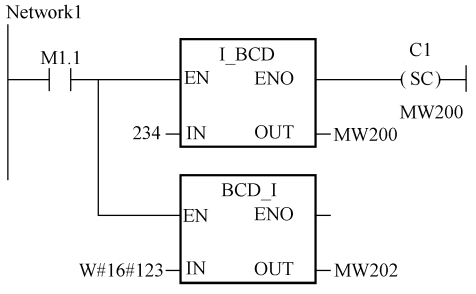
转换指令可以将一个输入参数的数据类型转换为一个需要的数据类型，在大多数的逻辑运算时，数据类型有可能不相同，例如数据类型可能为整数、双整数、浮点等，这样需要转换为统一的数据类型进行运算（字与整数类型不需要转换，在符号表或在数据块中可以定义变量的数据类型，如果没有定义变量的数据类型，例如 MW100，在编程时，既可以作为一个字类型也可以作为一个整数类型，数据类型根据指令自动转换）。转换指令见表 6-8。

使用转换指令的示例程序见表 6-9。

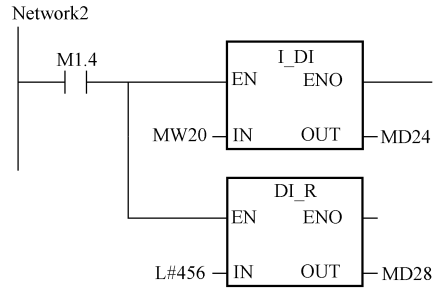
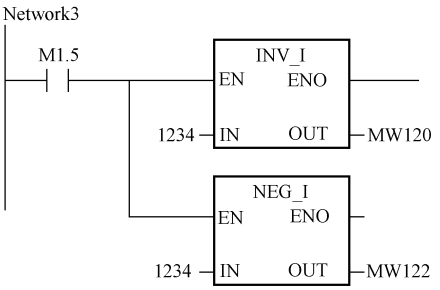
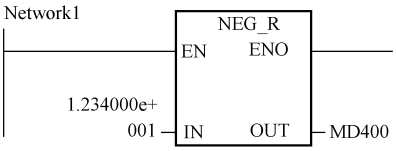
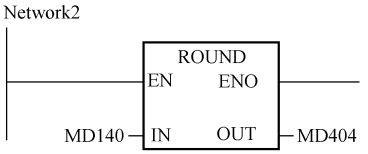
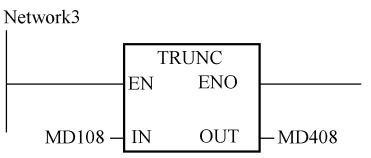
**表 6-8 转换指令**

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 转换指令 | BCD_I | BCD 码转换为整数 | BTI | BCD 转成单字整数（16 位） |
| | I_BCD | 整数转换为 BCD 码 | ITB | 16 位整数转换为 BCD 数 |
| | I_DI | 整数转换为双整数 | ITD | 单字（16 位）转换为双字整数（32 位） |
| | BCD_DI | BCD 码转换为双整数 | BTD | BCD 转成双字整数（32 位） |
| | DI_BCD | 双整数转换为 BCD 码 | DTB | 双字整数（32 位）转换为 BCD 数 |
| | DI_R | 双整数转换为实数 | DTR | 双字整数（32 位）转换为浮点（32 位 IEEE 浮点数） |
| | INV_I | 整数的二进制反码 | INVI | 单字整数反码（16 位） |
| | INV_DI | 双整数的二进制反码 | INVD | 双字整数反码（32 位） |
| | NEG_DI | 双整数的二进制补码 | NEGD | 双字整数补码（32 位） |
| | NEG_I | 整数的二进制补码 | NEGI | 单字整数补码（16 位） |
| | NEG_R | 浮点数求反 | NEGR | 浮点求反（32 位 IEEE FP） |
| | ROUND | 取整 | RND | 取整 |
| | TRUNC | 舍去小数，取整为双整数 | TRUNC | 截尾取整 |
| | CEIL | 上取整 | RND + | 取整为较大的双字整数 |
| | FLOOR | 下取整 | RND – | 取整为较小的双字整数 |
| | | | CAD | 改变 ACCU1 字节的次序（32 位） |
| | | | CAW | 改变 ACCU1 字中字节的次序（16 位） |

**表 6-9 转换指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| BCD 码与整数的转换指令 | | |
|  | **Network 1**<br>A M 1.1<br>JNB _001<br>L W#16#123<br>BTI<br>T MW 202<br>L 234<br>ITB<br>T MW 200<br>AN OV<br>SAVE<br>CLR<br>_001: A BR<br>L MW 200<br>S C 1 | 在程序段 1 中，M1.1 为 1 时，将整数 234 转换为 BCD 码 W#16#234 存入变量 MW200 中，将 BCD 码 W#16#123 转换为整数 123，存入变量 MW202 中，同时设置计数器 C1 的预置值为 MW200（计数器的值必须为 BCD 码）；使用 STL 编程时，使用 OV 状态位判断整数转换 BCD 码是否超限，如果转换错误不会将故障值放入计数器 C1 中（执行 JNB 跳转指令，将 RLO 信息复制到 BR 位，只有 M1.1 从 0 到 1 跳变时，才能将 MW200 中存储的 BCD 码数值载入计数器 C1 中） |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 整数转换指令 | | |
|  | **Network 2**<br>A M 1.4<br>JNB _007<br>L MW 20<br>ITD<br>T MD 24<br>L L#456<br>DTR<br>T MD 28<br>_007：NOP 0 | 在程序段2中，M1.4为1时，将整数变量MW20转换为双整数变量MD24（使用MOVE指令将MW20传送到MW26中也可以实现同样的功能），将双整数常数L#456转换为浮点值456.0存入变量MD28中 |
| 码制转换指令 | | |
|  | **Network 3**<br>A M 1.5<br>JNB _003<br>L 1234<br>INVI<br>T MW 120<br>L 1234<br>NEGI<br>T MW 122<br>_003：NOP 0 | 在程序段3中，M1.5为1时，执行INV_I指令，将输入参数1234与字W#16#FFFF的位进行“异或”操作（实际上将输入参数中所有的位信号取反）并将结果存入MW120中，执行NEG_I指令，将输入参数1234的值取反转换为-1234并将结果存入MW122中 |
| 浮点转换指令 | | |
|  | **Network 1**<br>L 1.234000e+001<br>NEGR<br>T MD 400 | 在程序段1中，将浮点值12.34取反转换为-12.34，并将结果存入MD400中 |
|  | **Network 2**<br>L MD 140<br>RND<br>T MD 404 | 在程序段2中将存储于变量MD140的浮点值转换为双整型数值，并存储于MD404中，转换采用最接近的原则，例如浮点值12.5转换双整型数值为L#12，浮点值12.51转换双整型数值为L#13 |
|  | **Network 3**<br>L MD 108<br>TRUNC<br>T MD 408 | 在程序段3中将存储于变量MD108的浮点值转换为双整型数值，并存储于MD408中，转换采用去小数部分的原则，例如大于12.0小于13.0的浮点值转换双整型数值为L#12 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 浮点转换指令 | | |
| Network4<br>CEIL<br>EN ENO<br>MD152 IN OUT MD412 | **Network 4**<br>L MD 152<br>RND +<br>T MD 412 | 在程序段 4 中将存储于变量 MD152 的浮点值转换为双整型数值，并存储于 MD412 中，转换采用接近最大值（正方向）的原则，例如大于 12.0、小于 13.0 的浮点值转换双整型数值为 L#13 |
| Network5<br>FLOOR<br>EN ENO<br>MD156 IN OUT MD416 | **Network 5**<br>L MD 156<br>RND –<br>T MD 416 | 在程序段 5 中将存储于变量 MD156 的浮点值转换为双整型数值，并存储于 MD416 中，转换采用接近最大值（负方向）的原则，例如大于 –13.0 小于 –12.0的浮点值转换双整型数值为 L# –13 |

## 6.5 计数器指令

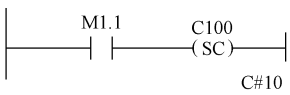
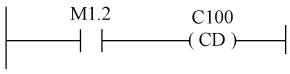
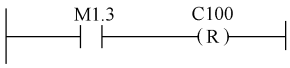
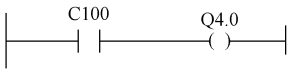
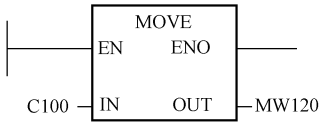
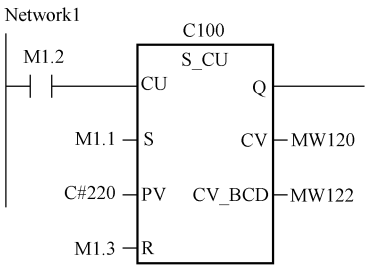
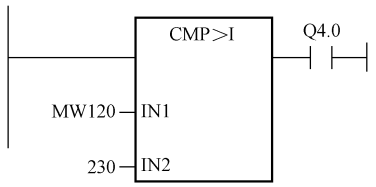
在 CPU 的系统存储器中，留有计数器存储区。该存储区为每一个计数器地址保留一个 16 位字。而能够使用计数器的个数由具体的 CPU 类型决定。计数器指令见表 6-10。

**表 6-10 计数器指令**

| | LAD | 说明 | STL | 说明 |
|---|---|---|---|---|
| 计数器指令 | ---（CD） | 减计数器线圈 | CD | 降计数器 |
| | ---（CU） | 加计数器线圈 | CU | 升计数器 |
| | ---（SC） | 预置计数器值 | S | 计数值置初值，例如：S C15 |
| | S_CD | 减计数器 | R | 复位计数器，例如：R C15 |
| | S_CU | 加计数器 | FR | 重新启动计数器，例如：FR C10 |
| | S_CUD | 加-减计数器 | L | 以整数形式将当前的计数器值写入 ACCU1，例如：L C15 |
| | | | LC | 把当前的计数器值以 BCD 码形式装入 ACCU1，例如 LC C15 |

使用 LAD 编程，计数器指令分为两种：1）加减计数器线圈如 –（CD）、–（CU），使用计数器线圈时必须与预置计数器值指令 –（SC）、计数器复位指令结合使用；2）加减计数器，计数器中包含计数器复位、预置等功能。使用 STL 编程，计数器指令只有升计数器 CU 和降计数器 CD 两个指令，S、R 指令为位操作指令，可以对计数器进行预置初值和复位操作，FR 指令可以重新启动计数器，例如设定计数器初值需要一个沿触发信号，如果触发信号常为 1，不能再次触发设定指令，使用 FR 指令，将清除计数器的沿存储器，常 1 的触发信号可以再次产生沿信号并重新设定计数器初值，FR 指令在实际编程中很少使用。使用计数器指令的示例程序见表 6-11。

**表 6-11　计数器指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| 计数器线圈的使用（以减计算指令为例） | | |
| Network1<br>  <br><br>Network2<br>  <br><br>Network3<br>  <br><br>Network4<br>  <br><br>Network5<br>  <br> | **Network 1**<br>A　M　1.1<br>L　C#10<br>S　C　100<br><br>**Network 2**<br>A　M　1.2<br>CD　C　100<br><br>**Network 3**<br>A　M　1.3<br>R　C　100<br><br>**Network 4**<br>A　C　100<br>=　Q　4.0<br><br>**Network 5**<br>L　C　100<br>T　MW　120<br>LC　C　100<br>T　MW　122 | 在程序段 1 中，M1.1 为 1 时，设置计数器 C100 的初值为 C#10，计数值为 16 位 BCD 码 10<br><br>在程序段 2 中，位信号 M1.2 每一个上升沿触发计数器 C100 减 1（初始值为 10）<br><br>在程序段 3 中，位信号 M1.3 的上升沿将复位计数器 C100 存储的计数值，指令执行完成，C100 计数值为 0<br><br>在程序段 4 中，计数器 C100 存储的计数值减到 0 或被复位，计数器 C100 的位信号输出，赋值 Q4.0 为 1<br><br>在程序段 5 中，当前的计数值可以传送到其他变量中，使用 STL 编程，MW120 中的计数值为整数，MW122 中的计数值为 BCD 码 |
| 计数器指令的使用（以减计算指令为例） | | |
| Network1<br>  <br><br>Network2<br>  <br> | **Network 1**<br>A　M　1.2<br>CU　C　100<br><br>A　M　1.1<br>L　C#220<br>S　C　100<br><br>A　M　1.3<br>R　C　100<br><br>L　C　100<br>T　MW　120<br><br>LC　C　100<br>T　MW　122<br>**Network 2**<br>L　MW　120<br>L　230<br>>I<br>=　Q　4.0 | 在程序段 1 中，M1.1 为 1 时，设置计数器 C100 的初值为 C#220，计数值为 16 位 BCD 码 220；位信号 M1.2 每一个上升沿触发计数器 C100 加 1（初始值为 220）；位信号 M1.3 的上升沿将复位计数器 C100 存储的计数值，指令执行完成，C100 计数值为 0；计数器 C100 存储的计数值最大为 999，如果达到最大值后再有新的脉冲信号，计数器不变。当前的计数值可以传送到其他变量中，MW120 中的计数值为整数，MW122 中的计数值为 BCD 码<br><br>在程序段 2 中，如果计数值大于 230，触发输出位 Q4.0 |

**注意：**

在计算计数器采样的最大频率时，需要考虑 CPU 的扫描时间、输入信号 0→1 的跳变时间和 1→0 的跳变时间，如果输入频率过高，计数器可能会丢失采样脉冲，建议采用高速计数器模块。计数器的最大值为 999，如果需要更大的计数范围，可以使用两个计数器叠加或调用 IEC 计数器 SFB0～2。

## 6.6 数据块操作指令

数据块占用 CPU 的工作存储区和装载存储区，数据块的个数及每个数据块的大小，用户可以自由定义（数据块的个数和大小不能超出 CPU 的最大限制），数据块中包含用户定义的变量，访问数据块中的变量首先需要将数据块打开。数据块的打开指令是一个数据块的无条件调用。数据块打开后，可以通过 CPU 内的数据块寄存器 DB 或 DI 直接访问数据块的内容。数据块的操作指令见表 6-12。

**表 6-12 数据块操作指令**

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 数据块指令 | ---（OPN） | 打开数据块 | OPN | 打开数据块 |
| | | | CDB | 交换 DB 与 DI 寄存器 |
| | | | L DBLG | 把共享数据块的长度写入 ACCU1 |
| | | | L DBNO | 把共享数据块的号写入 ACCU1 |
| | | | L DILG | 把背景数据块的长度写入 ACCU1 |
| | | | L DINO | 把背景数据块的号写入 ACCU1 |

使用数据块指令的示例程序见表 6-13。

**表 6-13 数据块指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| 数据块中数据的访问 | | |
| Network1<br>DB2<br>(OPN) | **Network 1**<br>OPN DB 2 | 在程序段 1 中，打开 DB2<br>在程序段 2 中，如果 DB2 中的变量 DBW2 大于 DBW4，则输出 DBX20.0（所有变量均为 DB2 中数据） |
| Network2<br>CMP>I<br>DBW2 IN1<br>DBW4 IN2<br>DBX20.0 | **Network 2**<br>L DBW 2<br>L DBW 4<br>>I<br>= DBX 20.0 | 在程序段 3 中，直接调用 DB4 中的数据 DB4.DBW12，相当于关闭 DB2，打开 DB4，如果小于 DB4.DBW18，则输出 DBX21.0（由于打开 DB4，网络 3 中所有变量均为 DB4 中数据） |
| | **Network 3**<br>L DB4.DBW 12<br>L DB4.DBW 18<br><I<br>= DBX 21.0 | 在编程应用中，每次直接调用数据块数据时都会先打开数据块，然后调用数据块数据，在一个数据块中进行数据处理，直接调用数据块数据，将产生多余的打开数据块指令，增加指令处理时间 |
| Network3<br>CMP<I<br>DB4.DBW12 IN1<br>DB4.DBW18 IN2<br>DBX21.0 | **Network 4**<br>OPN DB 2<br>OPN DI 4<br>L DBW 50<br>T DIW 60<br>CDB<br>L DBW 70<br>T DIW 80 | 在程序段 4 中，同时只能打开两个数据块，使用 DB 和 DI 便于区别，例如 OPN DB2、OPN DI4，程序中将 DB2.DBW50 中的数据传送到 DB4.DBW60 中；CDB 将 DB2 和 DB4 两个数据块的“号”相互交换，下面的程序中将 DB4.DBW70 中的数据传送到 DB2.DBW80 中 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 数据块处理指令 | | |
| | **Network 1**<br>OPN DB 2<br>OPN DI 4<br>L DBLG<br>T MD 20<br>L DILG<br>T MD 24 | 在程序段1中，分别打开DB2和DB4，将数据块DB2的长度（数据块容量，包含的字节个数）传送到MD20中，将数据块DB4的长度传送到MD24中 |
| | **Network 2**<br>L DBNO<br>T MD 40<br>L DINO<br>T MD 44 | 在程序段2中，将打开的DB的“号”传送到MD40中，例如DB2，将2传送到MD40中；将打开的DI块的“号”传送到MD44中 |

## 6.7 逻辑控制指令

逻辑控制包含各种跳转指令，通过跳转指令及程序跳转标签（LABEL）控制程序的跳转。逻辑控制指令见表6-14。

**表6-14 逻辑控制指令**

| | LAD | 说明 | STL | 说明 |
|---|---|---|---|---|
| 逻辑控制指令 | ---（JMP） | 跳转 | JC | 如果RLO=1，则跳转 |
| | ---（JMPN） | 若非则跳转 | JCN | 如果RLO=0，则跳转 |
| | LABEL | 标号 | JCB | 如果RLO=1，则跳转，并把RLO的值存于状态字的BR位中 |
| | | | JB1 | 如果BR=1，则跳转 |
| | | | JL | 跳转到表格（多路多支跳转） |
| | | | JM | 如果为负，则跳转 |
| | | | JMZ | 如果小于等于0，则跳转 |
| | | | JN | 如果非0，则跳转 |
| | | | JNB | 如果RLO=0，则跳转，并把RLO的值存于状态字的BR位中 |
| | | | JNBI | 如果BR=0，则跳转 |
| | | | JO | 如果OV=1，则跳转 |
| | | | JOS | 如果OS=1，则跳转 |
| | | | JP | 如果大于0，则跳转 |
| | | | JPZ | 如果大于等于0，则跳转 |
| | | | JU | 无条件跳转 |
| | | | JUO | 若无效数，则跳转 |
| | | | JZ | 如果为0，则跳转 |
| | | | LOOP | 循环 |

使用LAD编程，程序跳转指令少，使用比较简单；使用STL编程时，可以根据状态位的状态进行程序跳转，跳转指令比较灵活。下面分别介绍两种编程语言的跳转指令。

### 6.7.1 LAD 跳转指令

LAD 跳转指令有 –（JMP）（为 1 跳转）和 –（JMPN）（为 0 跳转）两种，根据前面的条件，跳转到 LABEL（标号，自己定义，最多 4 个字符，第一字符必须是字母）程序段。典型的跳转指令如图 6-14 所示。

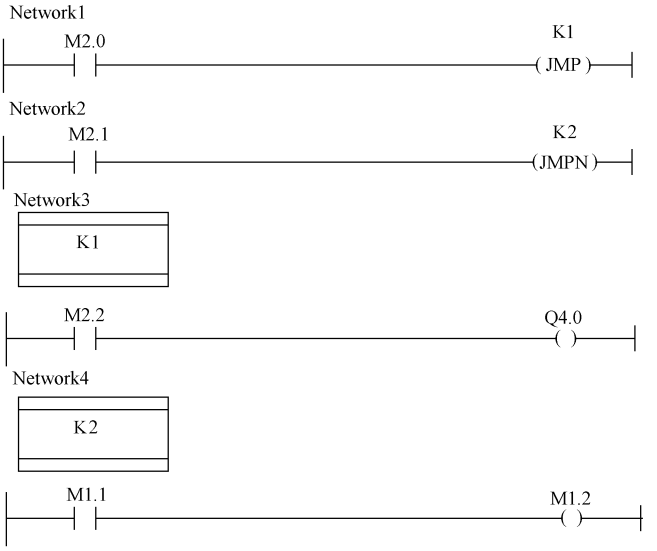



图 6-14 LAD 跳转指令

使用 –（JMP）指令，如果前面没有触发条件，程序执行后无条件跳转到指定标号的程序段；使用 –（JMPN）指令，前面必须有触发条件，否则指令违法，跳转指令中指定的标号必须与程序段标号一致，可以调用多条跳转指令跳转到同一标号的程序段中。在示例程序中，如果触点信号 M2.0 为 1，则程序跳转到标号为 K1 的程序段（程序段 3）；如果触点信号 M2.1 为 0，则程序跳转到标号为 K2 的程序段（程序段 4），CPU 不扫描程序段 3 程序。

### 6.7.2 STL 跳转指令

STL 与 LAD 编程跳转的方式相同，所有跳转指令必须定义跳转的标号，典型格式如下：

```
      JC      M2
      *                    //合法指令
      *
      *
M2：  *
      *
      *
```

JC 为跳转指令，M2 为用户定义的标号，在跳转的程序标号后面，必须加上符号“：”，根据跳转的条件，STL 编程语言可划分下面几种跳转指令。

1. 无条件跳转指令 JU 、JL

JU 指令的使用参考下面的示例程序：

```
        A     I     1.2
        JC    DELE        //如果 I1.2 为 1，则跳转到标号“ DELE”
        L     MB10
        INC   1
        T     MB10
        JU    FORW        //不需要触发条件，程序无条件跳转到标号“FORW”
DELE:   L0
        T     MB10
FORW:   A     I     2.1   //程序跳转到标号“FORW”后，程序从这里继续执行
```

JL 指令根据累加器 1 的数值跳转到相应的标号程序段，指令的使用参考下面的示例程序：

```
        L     MB0         //装载 MB0 到累加器 1
        JL    LSTX        //累加器 1 的值大于 3 时，跳转到标号 LSTX 程序段
        JU    SEG0        //累加器 1 的值等于 0 时，跳转到标号 SEG0 程序段
        JU    SEG1        //累加器 1 的值等于 1 时，跳转到标号 SEG1 程序段
        JU    SEG2        //累加器 1 的值等于 2 时，跳转到标号 SEG2 程序段
        JU    SEG3        //累加器 1 的值等于 3 时，跳转到标号 SEG3 程序段
LSTX:   JU    COMM
SEG0:*                    //合法指令
     *
        JU    COMM
SEG1:*                    //合法指令
     *
        JU    COMM
SEG2:*                    //合法指令
     *
        JU    COMM
SEG3:*                    //合法指令
     *
        JU    COMM
COMM:*                    //合法指令
     *
```

上面为 JL 指令固定的编程格式，最多允许有 255 个跳转条目，例子程序中有 4 个跳转条目，每个条目有一个标号，条目的序号从 0 开始（JU 的个数），如果累加器 1 的值大于所罗列的条目数，则跳转到 JL 指令指定的标号。

2. 基于逻辑结果的跳转指令 JC、JCN、JCB、JNB

如果RLO = 1，JC 指令执行，指令的使用参考下面的示例程序：

```
        A     I 1.2
        JC    JOVR        //如果 RLO = 1 （I1.2 为 1），则跳转到标号“JOVR”
        L     IW8         //如果跳转没有执行，程序从这里继续扫描
```

```
        T    MW22
JOVR:A    I 2.1        //程序跳转到标号“JOVR”后，程序从这里继续执行
```

与 JC 指令相反，如果 RLO = 0，JCN 指令执行，示例程序相同，如果 I1.2 为 0 时，触发 JCN 指令执行：

```
        A    I 1.2
        JCN  JOVR       //如果 RLO = 0（I1.2 为 0），则跳转到标号“JOVR”
        L    IW8        //如果跳转没有执行，程序从这里继续扫描
        T    MW22
JOVR:A    I 2.1        //程序跳转到标号“JOVR”后，程序从这里继续执行
```

**注意：**

当指令 JC 和 JCN 执行时，如果跳转不执行，将 RLO 位置 1 并从下一个指令开始执行。

如果 RLO = 1，JCB 指令执行，并将 RLO 位的状态复制到 BR 位，指令的使用参考下面的示例程序：

```
        A    I 1.2
        JCB  JOVR       //如果 RLO = 1（I1.2 为 1），则跳转到标号“JOVR”，
                          将 RLO 位的状态复制到 BR 位
        L    IW8        //如果跳转没有执行，程序从这里继续扫描
        T    MW22
JOVR:A    I 2.1        //程序跳转到标号“JOVR”后，程序从这里继续执行
```

如果 RLO = 0，JNB 指令执行，并将 RLO 位的状态复制到 BR 位，指令的使用参考下面的示例程序：

```
        A    I 1.2
        JNB  JOVR       //如果 RLO = 0（I1.2 为 0），则跳转到标号“JOVR”，
                          将 RLO 位的状态复制到 BR 位
        L    IW8        //如果跳转没有执行，程序从这里继续扫描
        T    MW22
JOVR:A    I 2.1        //程序跳转到标号“JOVR”后，程序从这里继续执行
```

3. 基于状态字中位状态（除 RLO 位）的跳转指令 JBI、JNBI、JO、JOS

JBI 与 JNBI 指令根据 BR 位的状态跳转，如果 BR 位为 1，则执行 JBI 指令，如果 BR 位为 0，则执行 JNBI 指令。以 JBI 指令为例，指令的使用参考下面的示例程序：

```
        CALL   SFC 14    // 调用系统函数 SFC14
        LADDR  := MW2
        RET_VAL := MW4
        RECORD := MW6
        JBI    M1        //如果调用出错，BR 位为 0，则函数执行跳转到标号“M1”
        L      2
        T      MW  12
        BE               //程序结束
M1:     L      3
```

```
        T     MW  12      //程序跳转到标号“M1”后，程序从这里继续执行
```

如果运算结果溢出，状态字的溢出位 OV =1，则 JO 指令通过判断 OV 位进行程序跳转，指令的使用参考下面的示例程序：

```
        L     MW10
        L     30
        *I                //MW10 * 30
        JO    OVER        //如果超上限（OV =1），则程序跳转
        T     MW10        //如果跳转没有执行，程序从这里继续扫描
        A     M 4.0
        R     M 4.0
        JU    NEXT
OVER:   AN    M 4.0       //程序跳转到标号“OVER”后，程序从这里继续执行
        S     M 4.0
NEXT:   NOP   0           //程序跳转到标号“NEXT”后，程序从这里继续执行
```

示例程序中，从位信号 M4.0 中，可以查询运算结果是否超上限（OV =1），若 M4.0 为 1，指示运算结果超上限；若 M4.0 为 0，指示运算结果正常。

如果运算结果溢出，状态字的溢出位 OV =1，则运算结果正常；若溢出位 OV =0，但是状态字的溢出保持位 OS 仍然为 1，记录运算结果溢出是否出现。通过程序块调用或调用 JOS 复位 OS 状态位，JOS 指令通过判断 OS 位进行程序跳转，指令的使用参考下面的示例程序：

```
        L     IW10
        L     MW12
        *I
        L     DBW25
        +I
        L     MW14
        -I
        JOS   OVER      //如果上面三条运算指令中任何一条运算结果超上限
                          （OV =1），则程序跳转到标号“OVER”
        T     MW16      //如果跳转没有执行，程序从这里继续扫描
        A     M 4.0
        R     M 4.0
        JU    NEXT
OVER:   AN    M 4.0     //程序跳转到标号“OVER”后，程序从这里继续执行
        S     M 4.0
NEXT:   NOP   0         //程序跳转到标号“NEXT”后，程序从这里继续执行
```

与 JO 跳转指令程序相同，示例程序中，若 M4.0 为 1，指示运算结果超上限；若 M4.0 为 0，指示运算结果正常。

4. 基于运算结果的跳转指令 JZ、JN、JP、JM、JPZ、JMZ、JUO

这些跳转指令基于运算结果 CC0 和 CC1 的状态，不同的状态触发不同的跳转指令，

CC0、CC1 位的状态与条件跳转指令的关系见表 6-15。

以 JP 为例，介绍跳转指令的使用，参考下面程序：

**表 6-15 CC0、CC1 位的状态与条件跳转指令的关系**

| CC1 信号状态 | CC0 信号状态 | 运算结果 | 触发的跳转指令 |
|---|---|---|---|
| 0 | 0 | =0 | JZ |
| 1<br>或<br>0 | 0<br>或<br>1 | <>0 | JN |
| 1 | 0 | >0 | JP |
| 0 | 1 | <0 | JM |
| 0<br>或<br>1 | 0<br>或<br>0 | >=0 | JPZ |
| 0<br>或<br>1 | 0<br>或<br>0 | <=0 | JMZ |
| 1 | 1 | =1 | JUO |

```
      L     IW8
      L     MW12
      -I          //IW8 的值与 MW12 的值相减
      JP    POS   //如果计算结果 大于0（ ACCU 1 > 0），则跳转到程序标号
                    “POS”
      AN    M 4.0 //如果跳转没有执行，程序从这里继续扫描
      S     M 4.0
      JU    NEXT
POS:  AN    M 4.1 //程序跳转到标号“POS”后，程序从这里继续执行
      S     M 4.1
NEXT: NOP   0     //程序跳转到标号“NEXT”后，程序从这里继续执行
```

在程序中，如果变量 IW8 大于变量 MW12 中存储的值，则置位 M4.1，否则置位 M4.0。

## 6.8 整数运算指令

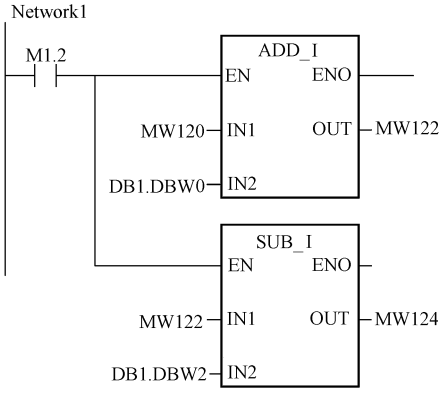
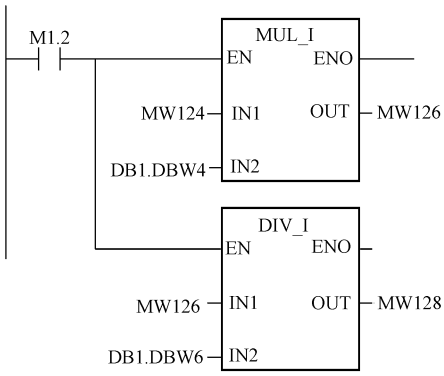
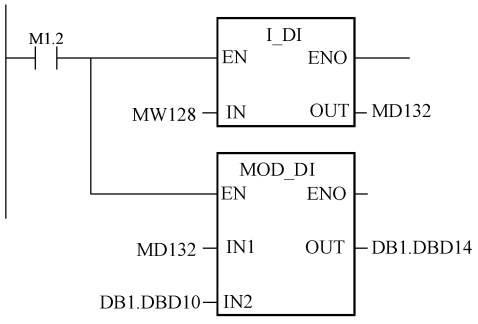
实现 16 位整数或者 32 位双整数之间的加、减、乘、除、取余等算术运算。整数指令见表 6-16。

**表 6-16 整数运算指令**

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 整数运算指令 | ADD_DI | 双整数加法 | +D | ACCU1 和 ACCU2 双字整数相加（32 位） |
| | ADD_I | 整数加法 | +I | ACCU1 和 ACCU2 整数相加（16 位） |
| | | | + | 整数常数加法（16 位，32 位） |
| | SUB_DI | 双整数减法 | -D | 从 ACCU2 减去 ACCU1 双整数（32 位） |
| | SUB_I | 整数减法 | -I | 从 ACCU2 减去 ACCU1 整数（16 位） |
| | MUL_DI | 双整数乘法 | *D | ACCU1 和 ACCU2 双字整数相乘（32 位） |
| | MUL_I | 整数乘法 | *I | ACCU1 和 ACCU2 整数相乘（16 位） |
| | DIV_DI | 双整数除法 | /D | ACCU2 除以 ACCU1 双整数（32 位） |
| | DIV_I | 整数除法 | /I | ACCU2 除以 ACCU1 整数（16 位） |
| | MOD_DI | 双整数取余数 | MOD | 双字整数形式的除法取余数（32 位） |

使用整数运算指令的示例程序见表6-17。

**表6-17　整数运算指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| Network1<br>  <br>Network2<br>  <br>Network3<br>  | OPN　DB　1<br>A　M　1.2<br>JCN　m1<br>L　MW　120<br>L　DBW　0<br>+I<br>L　DBW　2<br>-I<br>L　DBW　4<br>*I<br>L　DBW　6<br>/I<br>L　DBD　10<br>MOD<br>T　DBD　14<br>m1：NOP　0 | M1.2为开始计算的使能信号，实现等式（MW120 + DB1.DBW0 - DB1.DBW2 ）* DB1.DBW4 / DB1.DB W6 / DB1.DBD10的运算，将余数传送到DB1.DBD14中。在程序段1中，将变量MW120与DB1.DBW0相加，结果传送到MW122中<br>将变量MW122与DB1.DBW2相减，结果传送到MW124中<br><br>在程序段2中，将变量MW124与DB1.DBW4相乘，结果传送到MW126中<br>将变量MW126与DB1.DBW6相除，结果传送到MW128中<br><br>在程序段3中，将变量MW128转换为双整数MD132后除以DB1.DBD10，将余数传送到DB1.DBD14中<br>使用STL编程可以省去多余的中间变量，编程语句比较简练 |

**注意：**

在LAD编程时，在输入参数IN1和IN2时，输入的变量必须为整数数据类型，如果输入变量在符号表中定义数据类型为“WORD”或“DWORD”，则在输入变量时，报错不能输入。

## 6.9 浮点运算指令

实现对 32 位浮点的算术运算，与整数运算相比，浮点运算结果可以有小数，所以多出一些适合浮点运算的指令，例如平方、平方根、正余弦运算等，浮点运算指令见表 6-18。

**表 6-18 浮点运算指令**

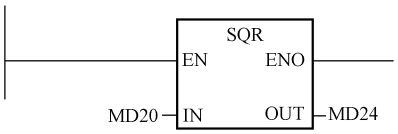
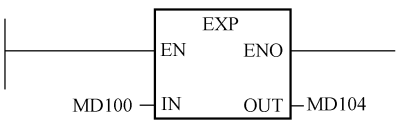
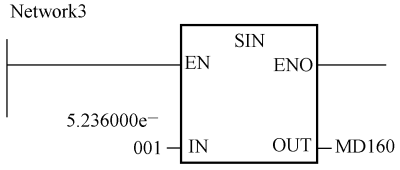
| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 浮点运算指令 | ADD_R | 浮点加法 | + R | ACCU1、ACCU2 相加（32 位 IEEE 浮点数） |
| | SUB_R | 浮点减法 | - R | 从 ACCU2 减去 ACCU1 浮点（32 位 IEEE 浮点数） |
| | MUL_R | 浮点乘法 | * R | ACCU1、ACCU2 相乘（32 位 IEEE 浮点数） |
| | DIV_R | 浮点除法 | /R | ACCU2 除以 ACCU1（32 位 IEEE 浮点数） |
| | ABS | 浮点数绝对值运算 | ABS | 绝对值（32 位 IEEE 浮点数） |
| | SQR | 浮点数平方 | SQR | 求平方（32 位 IEEE 浮点数） |
| | SQRT | 浮点数平方根 | SQRT | 求平方根（32 位 IEEE 浮点数） |
| | EXP | 浮点数指数运算 | EXP | 求指数（32 位 IEEE 浮点数） |
| | LN | 浮点数自然对数运算 | LN | 求自然对数（32 位 IEEE 浮点数） |
| | COS | 浮点数余弦运算 | COS | 余弦（32 位 IEEE 浮点数） |
| | SIN | 浮点数正弦运算 | SIN | 正弦（32 位 IEEE 浮点数） |
| | TAN | 浮点数正切运算 | TAN | 正切（32 位 IEEE 浮点数） |
| | ACOS | 浮点数反余弦运算 | ACOS | 反余弦（32 位 IEEE 浮点数） |
| | ASIN | 浮点数反正弦运算 | ASIN | 反正弦（32 位 IEEE 浮点数） |
| | ATAN | 浮点数反正切运算 | ATAN | 反正切（32 位 IEEE 浮点数） |

使用浮点运算指令的示例程序见表 6-19。

**表 6-19 浮点运算指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| 浮点运算基本指令（加、减、乘、除、绝对值，以加、除指令为例） | | |
| Network1<br>M1.3<br>ADD_R<br>EN ENO<br>MD200 IN1 OUT MD204<br>DB1.DBD20 IN2<br>DIV_R<br>EN ENO<br>MD204 IN1 OUT DB1.DBD28<br>DB1.DBD24 IN2 | **Network 1**<br>OPN DB 1<br>A M 1.3<br>JCN m2<br>L MD 200<br>L DBD 20<br>+ R<br>L DBD 24<br>/R<br>T DBD 28<br>m2： NOP 0 | 以 M1.3 为开始计算的使能信号，实现等式（MD200 + DB1.DBD20）/ DB1.DBD24 的运算，将运算结果传送到 DB1.DBD28 中 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 浮点运算扩展指令（求平方、求平方根指令，以求平方指令为例） | | |
| Network1<br>  | **Network 1**<br>L MD 20<br>SQR<br>AN OV<br>JC OK<br>BEU<br>OK: T MD 24 | 在程序段1中，将变量MD20的平方值传送到MD24中。在STL程序示例中加入溢出判断 |
| 浮点运算扩展指令（求指数、求自然对数指令，以求指数指令为例） | | |
| Network2<br>  | **Network 2**<br>L MD 100<br>EXP<br>T MD 104 | 在程序段2中，变量MD100为e的指数，并将运算结果传送到MD104中 |
| 浮点运算扩展指令（余弦、正弦、正切、反余弦、反正弦、反正切指令，以正弦指令为例） | | |
| Network3<br>  | **Network 3**<br>L 5.236000e-001<br>SIN<br>T MD 160 | 在程序段3中，将弧度为0.5236的正弦值传送到MD160中。所有角度指令中，使用弧度替代角度值 |

## 6.10 赋值指令

LAD编程语言MOVE（赋值）指令将输入端IN指定地址中的值或常数复制到输出端OUT指定的地址中。MOVE最多可以复制4B的变量，用户定义的数据类型（例如数组或结构）必须使用系统功能“BLKMOVE”（SFC20）进行复制。在STL编程语言中，使用装载和传递指令实现相同功能，装载功能实现将一个最大4B的常数、变量或地址寄存器传送到累加器，传递功能实现将累加器中的值传送到变量，除此之外，装载和传递指令中还包含对地址寄存器操作的指令。

CPU内部寄存器中有两个地址寄存器，分别以AR1、AR2表示，每个地址寄存器占有32位地址空间，地址寄存器存储区域内部和区域交叉地址指针，用于地址的间接寻址，地

址寄存器及指针的使用，在地址指针章节中将详细介绍。赋值指令见表 6-20。

**表 6-20 赋值指令**

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 赋值指令 | MOVE | 赋值 | L | 把数据装载入 ACCU1 |
| | | | L STW | 把状态字写入 ACCU1 |
| | | | LAR1 | 将 ACCU1 存储的地址指针写入 AR1 |
| | | | LAR1 <D> | 将指明的地址指针写入 AR1 |
| | | | LAR1 AR2 | 将 AR2 的内容写入 AR1 |
| | | | LAR2 | 将 ACCU1 存储的地址指针写入 AR2 |
| | | | LAR2 <D> | 将指明的地址指针写入 AR2 |
| | | | T | 把 ACCU1 的内容传到目标单元 |
| | | | T STW | 把 ACCU1 的内容传输给状态字 |
| | | | TAR1 | 将 AR1 存储的地址指针传输 ACCU1 |
| | | | TAR1 <D> | 将 AR1 存储的地址指针传输给指明的变量中 |
| | | | TAR1 AR2 | 将 AR1 存储的地址指针传输 ACCU2 |
| | | | TAR2 | 将 AR2 存储的地址指针传输 ACCU1 |
| | | | TAR2 <D> | 将 AR2 存储的地址指针传输给指明的变量中 |
| | | | CAR | 交换 AR1 和 AR2 的内容 |

从指令表中可以看到，使用 LAD 编程语言只有赋值指令，使用 STL 编程语言指令分为装载和传递指令，其中包含地址寄存器的处理指令。

## 6.10.1 LAD 赋值指令

在前面的示例程序中已经多次使用赋值指令，它的作用是将一个常数或者变量的值传送到另一个变量中（输出端必须是变量）。因为赋值指令是数值的传递，所有指令的输入、输出端的数据类型没有限制，例如输入端是一个字节，输出端可以是一个字节、字、双字，也可以是一个整数、双整数，数值的类型可以自动转换（已经定义的数据类型不能转换）。赋值指令的使用如图 6-15 中示例程序。

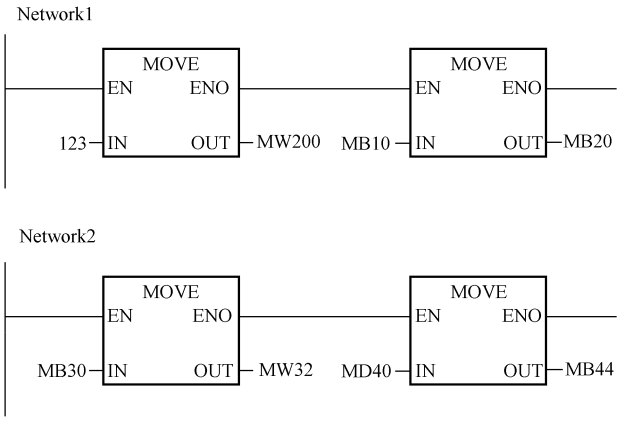



图 6-15 赋值指令

示例程序中，在程序段 1 将常数 123 传送到变量 MW200 中，将 MB10 中的数值传送到变量 MB20 中，在程序段 2 将变量 MB30 中的数值传送到变量 MW32 中，字节传送到字中，

数值不会溢出，如果将变量 MD40 传送到变量 MB44 中，则传送的数值大于 255 出现溢出（实际将 MB43 传送到变量 MB44，如果 MD40 的值为 DW#16#1234，指令执行后 MB44 的值为 B#16#34），运算时会出现错误，编程时应注意。

### 6.10.2 STL 装载、传递指令

STL 的赋值指令分为装载和传递指令，指令介绍如下。

1. L（装载）指令与 T（传送）指令

L 指令将数值传送到累加器 1，T 指令将累加器 1 中的数值传送到变量，指令使用的示例程序如下：

```
L     IB 10          //将 IB10 装载到累加器 1 中
T     QB1            //将累加器 1 中的值（IB10）传送到 QB1

L     MB 120         //将 MB120 装载到累加器 1 中
T     DBB100         //将累加器 1 中的值（MB120）传送到 DBB100

L     DIW 16         //将 DIW16 装载到累加器 1 中
T     DIW80          //将累加器 1 中的值（DIW16）传送到 DIW80

L     LD 252         //将临时变量 LD 252 装载到累加器 1 中
T     MD 40          //将累加器 1 中的值（LD 252）传送到 MD40

L     P# I 8.7       //将指针 P# I8.7 装载到累加器 1 中
T     MD80           //将累加器 1 中的值（指针 P# I 8.7）传送到 MD80
```

装载指令 L 与传送指令 T 配合使用，装载指令也可以将累加器 1 中的值堆栈到累加器 2 中，例如程序：

```
L     MB 10          //将 MB10 装载到累加器 1 中
L     MB 11          //将 MB11 装载到累加器 1 中，MB10 自动进入累加器 2
T     DBB100         //将累加器 1 中的值（MB11）传送到 DBB100
```

传送指令 T 只能将累加器 1 中值传送到变量中。

2. L STW 与 T STW 指令

L STW 指令装载状态字到累加器 1 中，但是不能将 S7-300 CPU 的/FC、STA、OR 状态位装载到累加器；T STW 指令将累加器 1 中的值传送到状态字中，指令使用的示例程序如下：

```
L     STW            //将当前的状态字装载到累加器 1 中
T     MW 140         //将累加器 1 中的值（状态字）传送到 MW140 进行分析判断
L     2#111111111    //将 2#111111111 装载到累加器 1 中
T     STW            //将状态字中所有状态位置 1
```

L STW 指令与 T STW 指令在程序中对状态字进行监控，在实际的编程应用中很少使用。

3. LAR1 与 TAR1 指令

LAR1 指令将累加器 1 中的值装载到地址寄存器 1 中；TAR1 指令将地址寄存器 1 中的

值传送到累加器 1 中，指令使用的示例程序如下：

```
L     P# I20.0      //将指针 P# I20.0 装载到累加器 1 中
LAR1                //将累加器 1 中的值（指针 P# I20.0）装载到地址寄存器 1
TAR1                //将地址寄存器 1 中的地址（指针 P# I20.0）传送到累加器 1
                      中
T     MD80          //将累加器 1 中的值（指针 P# I20.0）传送 MD80
```

上面的示例程序实现对地址寄存器 1 的读写操作，程序实际将指针 P#I 20.0 传送到变量 MD80 中。

4. LAR2 与 TAR2 指令

与指令 LAR1、TAR1 使用方式相同，实现对地址寄存器 2 的读写操作。

5. LAR1 <D> 与 TAR1 <D> 指令

与 LAR1 相比，LAR1 <D> 指令直接将地址指针装载到地址寄存器 1 中，同样 TAR1 <D> 直接将地址寄存器 1 中的地址指针传送到变量中。指令中的 <D> 表示存储地址指针的双整型变量或指针常数，指令使用的示例程序如下：

```
LAR1   DBD 24      //将数据块变量 DBD 24 存储的地址指针直接装载到地址寄存
                     器 AR1 中
LAR1   LD 100      //将区域变量 LD 100 存储的地址指针直接装载到地址寄存器
                     AR1中
LAR1   MD 40       //将变量 MD 40 存储的地址指针直接装载到地址寄存器AR1中
LAR1   P# M100.0   //将地址指针常数 P# M100.0 直接装载到地址寄存器 AR1 中

TAR1   DBD 20      //将地址寄存器 AR1 中的值，直接传送到变量 DBD 20 中
TAR1   DID 30      //将地址寄存器 AR1 中的值，直接传送到变量 DID 30 中
TAR1   LD 180      //将地址寄存器 AR1 中的值，直接传送到区域变量 LD 180 中
TAR1   MD 24       //将地址寄存器 AR1 中的值，直接传送到变量 MD 24 中
```

6. LAR2 <D> 与 TAR2 <D> 指令

与指令 LAR1 <D> 、TAR1 <D> 使用方式相同，实现对地址寄存器 2 的直接读写操作。

7. LAR1 AR2 与 TAR1 AR2 指令

LAR1 AR2 指令将地址寄存器 AR2 中的值，直接装载到地址寄存器 AR1 中；TAR1 AR2 指令将地址寄存器 AR1 中的值，直接传送到地址寄存器 AR2 中，指令使用的示例程序如下：

```
LAR1   P#10.0      //将地址指针常数 P# 10.0，直接装载到地址寄存器 AR1 中
TAR1   AR2         //将地址寄存器 AR1 中的值，直接传送到地址寄存器 AR2 中
LAR1   AR2         //将地址寄存器AR2 存储的地址直接装载到地址寄存器AR1中
TAR1   MD  100     //将地址寄存器 AR1 中的值，直接传送到变量 MD 100 中

A      I [MD 100]//如果 I10.0 为 1，Q1.1 输出为 1
=      Q     1.1
```

8. CAR 指令

CAR 指令将地址寄存器 1 与地址寄存器 2 中存储的地址指针相互交换，指令使用的示

例程序如下：

```
LAR1    P#10.0      //将地址指针常数 P# 10.0 直接装载到地址寄存器 AR1 中
LAR2    P#11.0      //将地址指针常数 P# 11.0 直接装载到地址寄存器 AR2 中
CAR                 //AR1 与 AR2 地址指针交换，AR1 中装载地址指针 P# 11.0，AR2 中装载地址指针 P# 10.0
TAR1    MD  100     //将地址寄存器 AR1 中的值直接传送到变量 MD100 中，MD100 中存储地址指针 P# 11.0
CAR                 //AR1 与 AR2 地址指针交换，AR1 中装载地址指针 P# 10.0，AR2中装载地址指针 P# 11.0
TAR1    MD  104     //将地址寄存器 AR1 中的值直接传送到变量 MD104 中，MD 104 中存储地址指针 P# 10.0
A       M[MD 100]//如果 M11.0 为 1，M10.0 输出为 1
=       M[MD 104]
```

## 6.11 程序控制指令

程序控制指令实现函数、函数块的调用以及通过主控传递方式（Master Control Relay）实现程序段使能的控制，程序控制指令见表 6-21。

**表 6-21 程序控制指令**

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 程序控制指令 | ---（MCR <） | 主控传递接通 | MCR（ | 把 RLO 存入 MCR 堆栈，开始 MCR |
| | ---（MCR >） | 主控传递断开 | ）MCR | 把 RLO 从 MCR 堆栈中弹出，结束 MCR |
| | ---（MCRA） | 主控传递启动 | MCRA | 激活 MCR 区域 |
| | ---（MCRD） | 主控传递停止 | MCRD | 去活 MCR 区域 |
| | ---（CALL） | 调用 FC/SFC（无参数） | CALL | 块调用 |
| | CALL_FB | 调用 FB | CC | 条件调用 |
| | CALL_FC | 调用 FC | UC | 无条件调用 |
| | CALL_SFB | 调用 SFB | BE | 块结束 |
| | CALL_SFC | 调用 SFC | BEC | 条件块结束 |
| | ---（ RET ） | 返回 | BEU | 无条件块结束 |

### 6.11.1 LAD 程序控制指令

1. 主控传递指令

主控传递指令可以将程序段分区、嵌套控制，在控制启动指令 -（MCRA）和控制停止指令 -（MCRD）间，通过主控传递接通指令 -（MCR <）和主控传递断开指令 -（MCR >），可以最多将一段程序分成 8 个区，只有打开第一个区，才能打开第二个区，以此类推，每打开一个区，才能执行本区的程序。-（MCRA）、-（MCRD）及 -（MCR >）指令前不能加入触发条件，-（MCR <）指令前必须加入触发条件。主控传递指令使用的示例程序如图 6-16 所示。

在示例程序中，如果 M1.1 为 1，打开 MCR 程序分区 1，分区 1 中的程序可以运行，例如 I1.2 为 1，将置位 Q2.1；如果 M1.2 为 1，打开 MCR 程序分区 2，分区 2 中的程序可以运

行，例如 I1.1 为 1，触发 Q2.0 输出，如果 M1.1 为 0，分区 1 关闭，即使 I1.1 为 1 也不能触发 Q2.0 输出，程序分区相互嵌套。

2. 程序调用指令

集成于 STEP7 函数库“Libraries”或用户编写的函数及函数块（FB Blocks、FC Blocks 目录）必须在主程序中调用才能运行，使用指令 CALL_FB、CALL_FC、CALL_SFB、CALL_SFC 可以对不同函数、函数块进行调用，指令的使用如图 6-17 所示的示例程序，程序段 1 中调用无形参函数 FC1，如果函数带有形参必须赋值，否则报错，如果已经编写 FC1，在“FC Blocks”库中可以找到，可以将 FC1 直接拖放到程序段中；程序段 2 为系统函数的调用，必须对形参赋值，否则报错，在“System Function Blocks”系统函数库中，将需要调用的函数直接拖放到程序段中；程序段 3、4 为函数块和系统函数块的调用，共同的特点是都需要背景数据块，从函数库中直接拖放到程序段中，在函数块的上方写入未使用的数据块作为背景数据块，点击“确认”自动生成，每次调用函数块或系统函数块时，必须分配不同的数据块号。

使用这些程序调用指令，都是将已经存在的函数或函数块拖放到 LAD 程序段的程序线中，形成一个类似盒子形状的程序框图，如果调用的函数带有形参，在程序框图的左边为输入端，在程序框图的右边为输出端。

使用 -(CALL) 指令只能调用函数 FC 和系统函数 SFC，并且函数不能带有形参，否则不能赋实参，

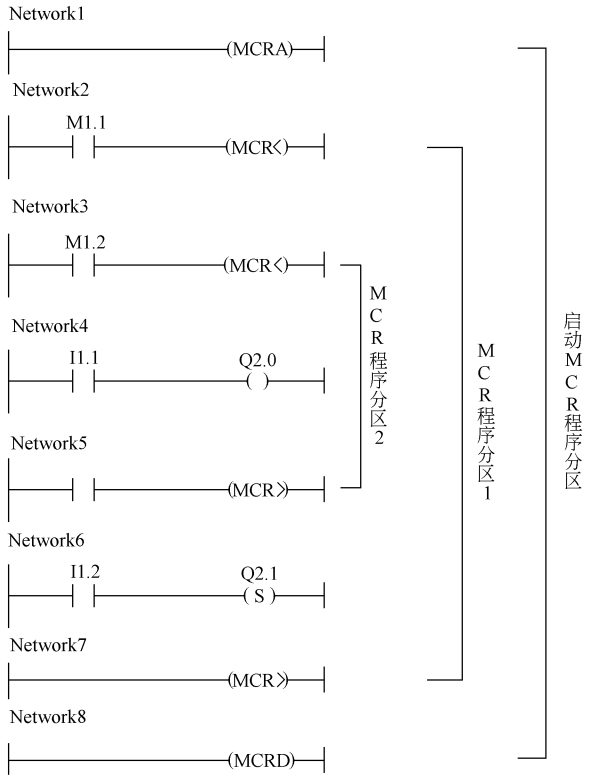



图 6-16 主控分程传递指令

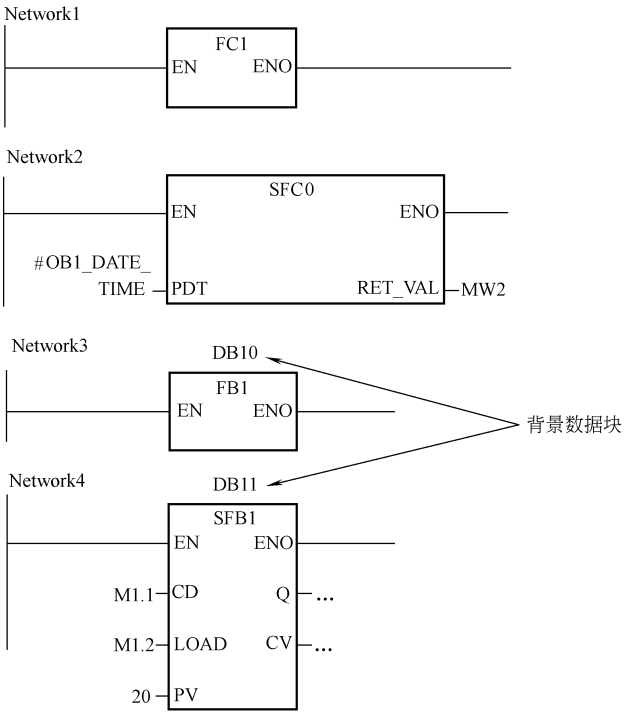



图 6-17 程序块调用指令

指令的使用如图 6-18 所示。




图 6-18 -(CALL) 指令

图 6-18 的程序中，如果 M1.1 为 1，调用函数 FC1，FC1 必须用手动输入。

**注意：**

如果没有预先创建函数或函数块，在调用时报错。

3. -( RET ) 返回指令

如果在主程序中执行返回指令，程序扫描重新开始，如果在子程序中执行返回指令，程序扫描返回子程序调用处，共同的特点是返回指令后面的程序不执行，返回指令的使用如图 6-19 所示示例程序。

示例程序中，如果 M1.2 为 1，执行返回指令，CPU 不扫描程序段 3 的程序。

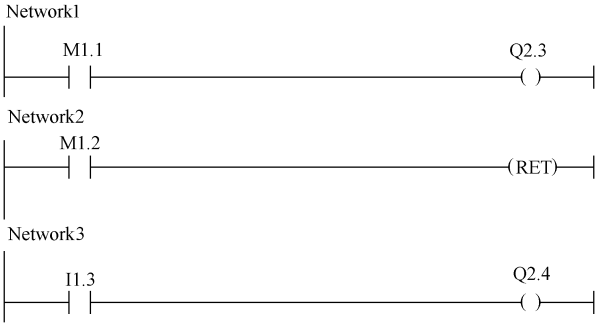



图 6-19 -（RET）返回指令

## 6.11.2 STL 程序控制指令

1. 主控传递指令

与 LAD 编程语言中使用主控传递指令的方法相同，使用 STL 编程示例程序如下：

```
MCRA                    //启动 MCR 分区
A       I 10.0
MCR (                   //I10.0 为 1，激活 MCR 分区 1，I10.0 为 0，关闭 MCR 分区 1
A       I 10.2
MCR (                   //I10.2 为 1，激活 MCR 分区 2，I10.2 为 0，关闭 MCR 分区 2
A       I 40.0
=       Q 80.0          //如果 MCR 分区 2 关闭，Q80.0 被复位为 0，与 I40.0 的值无关
L       MW   20
T       QW   100        //如果 MCR 分区 2 关闭，0 将传送到 QW100 中
) MCR                   //MCR 分区 2 结束
```

```
A       I10.4
=       Q80.2       //如果 MCR 分区 1 关闭，Q80.2 被复位为 0，与 I10.4. 的
                      值无关
) MCR               //MCR 分区 1 结束
MCRD                //MCR 分区结束
A       I10.1       //下面的指令与 MCR 分区无关
=       Q80.1
```

2. 程序调用指令

STL 编程语言中包括“CALL”、“CC”和“UC”指令，用于程序的调用，CC（有条件调用）与 UC（无条件调用）指令只能调用无形参的函数、函数块，与 LAD 中（CALL）指令的使用相同。“CALL”指令相当于 LAD 编程语言中的 CALL_FB、CALL_FC、CALL_SFB、CALL_SFC 指令，“CALL”指令的使用参考下面的示例程序。

（1）函数的调用

固定格式　CALL FCX。X 为函数号。

例如函数 FC6 的调用，FC6 带有形参，符号“:”左边为形参，右边为赋的实参，如果形参不赋值，程序调用报错。

```
CALL   FC6
形参              实参
NO OF TOOL  :  = MW100
TIME OUT    :  = S5T#12 S
FOUND       :  = Q 0.1
ERROR       :  = Q 100.0
```

（2）系统函数的调用

固定格式　CALL SFCX。X 为系统函数号。

例如系统函数 SFC43 的调用，不带有形参。

```
CALL    SFC43       // SFC43 实现重新触发看门狗定时器功能
```

系统函数如果带有形参，与函数的调用相同，必须赋值，否则程序调用报错。

（3）函数块的调用

固定格式　CALL FBX，DBY。X 为函数块号，Y 为背景数据块号，函数块与背景数据块使用符号“,”隔离。

例如函数块 FB99 的调用，背景数据块为 DB1，带有形参，符号“:”左边为形参，右边为赋的实参，由于调用函数块带有背景数据块，形参可以直接赋值，也可以稍后对背景数据块中的变量赋值。多次调用函数块时，必须分配不同的数据块作为背景数据块。

```
CALL    FB99, DB1
形参              实参
MAX_RPM:    = #RPM1_MAX
MIN_RPM :   = MW2
MAX_POWER:  = MW4
MAX_TEMP:   = #TEMP1
```

如果函数块 A 作为函数块 B 的形参，在函数块 B 调用函数块 A 时，不分配背景数据块，例如函数块 FB_A 的调用：

```
CALL   #FB_A
IN_1      : =
IN_2      : =
OUT_1     : =
OUT_2     : =
```

调用函数块 B 时，分配的背景数据块中包括所有函数 A 和 B 的背景参数，如果在函数块 B 中插入多个函数块作为形参，程序调用时只使用一个数据块作为背景数据块，节省数据块的资源（不能节省 CPU 的存储区），这样函数块具有多重背景数据块的能力，在函数块创建时可以选择。

（4）系统函数块的调用

固定格式　CALL SFBX，DBY。X 为函数块号，Y 为背景数据块号，系统函数块与背景数据块使用符号“,”隔离。

例如函数块 SFB4 的调用，背景数据块为 DB4，带有形参，符号“:”左边为形参，右边为赋的实参，由于调用系统函数块带有背景数据块，形参可以直接赋值，也可以稍后对背景数据块中的变量赋值。多次调用系统函数块时，必须分配不同的数据块作为背景数据块。

```
CALL   SFB4，DB4
形参              实参
IN          : = I0.1
PT          : = T#20s
Q           : = M0.0
ET          : = MW10
```

3. 程序结束指令

BE（程序结束）与 BEU（程序无条件结束）指令使用方法相同，如果程序执行上述指令，CPU 终止当前程序块的扫描，跳回程序块调用处继续扫描其他程序，如果程序结束指令被跳转指令跳过，程序扫描不结束，从跳转的目标点继续扫描。指令的使用参考下面的示例程序：

```
A       I 1.0
JC      NEXT        //如果 I1.0 为 1，程序跳转到 NEXT
L       IW4         //如果没有跳转，程序从这里连续扫描
T       IW10
A       I 6.0
A       I 6.1
S       M 12.0
BE                  //程序结束
NEXT：NOP 0         //跳转执行，程序从这里连续扫描
```

BEC 为有条件程序结束，在 BEC 指令前，必须加入条件触发，示例程序如下：

```
A   M     1.1
BEC
=   M     1.2
```

如果 M1.1 为 1，程序结束；如果 M1.1 为 0，程序继续运行，与 BE、BEU 指令不同，BEC 指令触发条件没有满足，置 RLO 位为 1，所以 M1.1 为 0 时，M1.2 为 1。

## 6.12 移位和循环指令

LAD 移位指令可以将输入参数 IN 中的内容向左或向右逐位移动；循环指令可以将输入参数 IN 中的全部内容循环地逐位左移或右移，空出的位用输入 IN 移出位的信号状态填充。

STL 移位指令将累加器 1 的低字或全部内容向左或向右逐位移动；循环指令将累加器 1 的全部内容循环地逐位左移或右移，空出的位用累加器 1 移出位的信号状态填充。移位和循环指令参考表 6-22。

**表 6-22 移位和循环指令**

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 移位和循环指令 | ROL_DW | 双字左循环 | RLD | 双字循环左移操作（32 位） |
| | | | RLDA | 带 CC1 位的 ACCU1 循环左移（32 位） |
| | ROL_DW | 双字右循环 | RRD | 双字循环右移（32 位） |
| | | | RRDA | 带 CC1 位的 ACCU1 循环右移（32 位） |
| | SHL_DW | 双字左移 | SLD | 双字左移（32 位） |
| | SHL_W | 字左移 | SLW | 单字左移（16 位） |
| | SHR_DI | 双整数右移 | SSD | 移位有符号双字整数（32 位） |
| | SHR_DW | 双字右移 | SRD | 双字右移（32 位） |
| | SHR_I | 整数右移 | SSI | 移位有符号单字整数（16 位） |
| | SHR_W | 字右移 | SRW | 单字右移（16 位） |

字移位指令移位的范围为 0～15，双字移位指令移位的范围为 0～31，对于字、双字移位指令，移出的位信号丢失，移空的位使用 0 补足，例如将一个字左移 6 位，移位前后位排列次序如图 6-20 所示。

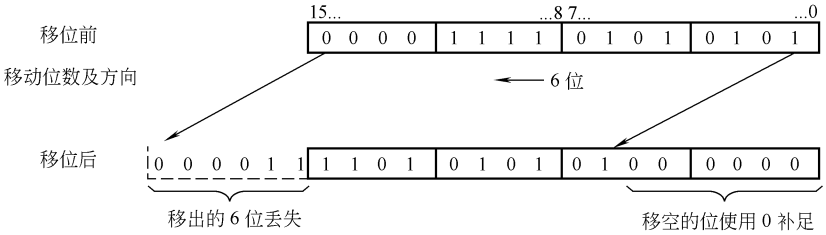



图 6-20 左移 6 位示意图

带有符合位的整数移位范围为0～15，双整数移位范围为0～31，移位方向只能向右移，移出的位信号丢失，移空的位使用符号位补足，整数为负值，符号位为1，整数为正值，符号位为0。例如将一个整数右移4位，移位前后位排列次序如图6-21所示。

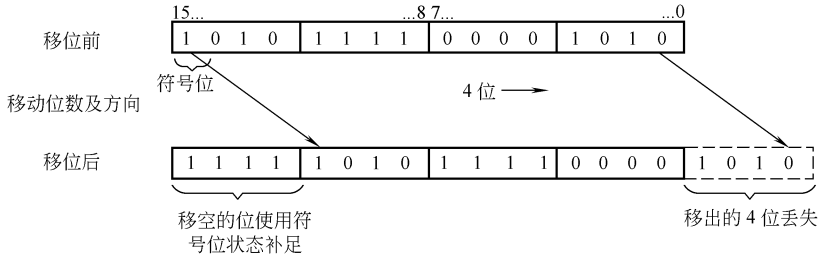



图6-21　整数右移4位示意图

使用STL编程时，注意固定的格式，例如一个字左移5位的程序：

```
L 5              //移动的位数
L MW120          //移位的变量
SLW
T MW122          //移位结果
```

执行移位指令时，将累加器2中值作为移动的位数，对累加器1中的值进行移位操作。

循环移位指令只能对双字进行操作，移位范围为0～31，如果移位大于32，实际移位为（（N－1）modulo 32）＋1，高位移出的位信号插入到低位移空的位中，例如将一个双字循环左移3位，移位前后位排列次序如图6-22所示。

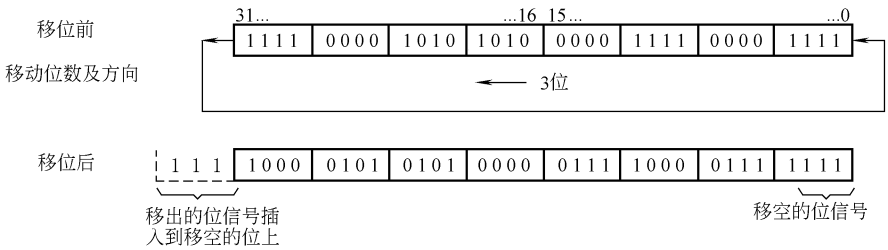
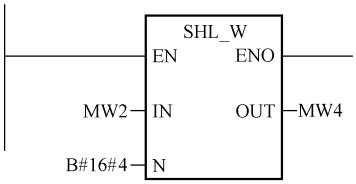
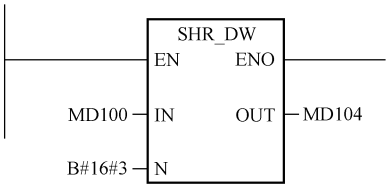
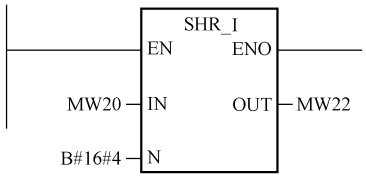
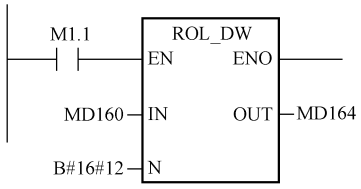



图6-22　循环移位示意图

STL编程语言中，RLDA与RRDA指令对双字进行循环移位操作，每次触发时循环左移、右移一位，将状态字中CC1的信号插入移空的位上。如果移出的位信号为1，置位状态字中CC1位，可以触发JP程序跳转指令进行逻辑判断。

使用移位和循环指令的示例程序见表6-23。

**表 6-23 移位和循环指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| 移位指令 | | |
| Network1<br>  | **Network 1**<br>L B#16#4<br>L MW 2<br>SLW<br>T MW 4 | 程序段 1 中将 MW2 左移 4 位，将结果传送到 MW4 中。假如移位指令执行前 MW2 中位的排列次序为：<br>1100 0101 1111 0011<br>指令执行完后的排列次序为：<br>0101 1111 0011 0000 高位 1100 被移出 |
| Network2<br>  | **Network 2**<br>L B#16#3<br>L MD 100<br>SRD<br>T MD 104 | 程序段 2 中，将 MD100 右移 3 位，将结果传送到 MD104 中 |
| Network3<br>  | **Network 3**<br>L B#16#4<br>L MW 20<br>SSI<br>T MW 22 | 程序段 3 中，将 MW20 右移 4 位，将结果传送到 MW22 中。假如移位指令执行前 MW2 中位的排列次序为：<br>1100 0101 1111 0011<br>指令执行完后的排列次序为：<br>1111 1100 0101 1111 |
| 循环指令 | | |
| Network4<br>  | **Network 4**<br>A M 1.1<br>JCN p2<br>L B#16#12<br>L MD 160<br>RLD<br>T MD 164<br>p2：NOP 0 | 程序段 4 中，如果 M1.1 为 1，将 MD160 中的值左移 18 位，将结果传送到 MD164 中 |
| | **Network 5**<br>L 100<br>L MW 140<br>–I<br>T MW 142<br>A M 12.1<br>FP M 12.2<br>JCN m1<br>L MD 2<br>RLDA<br>T MD 2<br>JP next<br>L 2<br>T MW 10<br>BE<br>next：L 3<br>T MW 10<br>m1：NOP 0 | 程序段 5 中，每次 M12.1 产生上升沿信号，执行 RLDA 指令执行，MD2 向左循环移一位，将 CC1 信号（如果 MW140 小于 100，状态字中的 CC1 位为 1，大于 100，CC1 位为 0）插入到移出空位上（MD2 的最后一位），如果移出的位信号为 0，将 2 传送到 MW10 中，如果移出的位信号为 1，置位 CC1 位，执行 JP 跳转指令将 3 传送到 MW10 中 |

## 6.13 状态位指令

前面已经介绍状态字是 CPU 中存储区中的一个寄存器，用于指示 CPU 运算结果的状态。状态位指令是位逻辑指令，针对状态字的各个位进行操作。通过状态位可以判断 CPU 运算中溢出、异常、进位、比较结果等状态。由于编程方法的原因，状态位指令只能在 LAD 中使用，STL 编程语言中对状态位的信息，有的可以直接使用，有的可以通过跳转指令完成，同样可以实现状态位指令在 LAD 中实现的功能，在指令示例程序中，介绍使用 STL 编程语言实现的方法。状态位指令见表 6-24。

**表 6-24 状态位指令**

| | LAD | 说 明 |
|---|---|---|
| 状态位指令 | ==0 ---\|\|--- | 结果位等于“0” |
| | >0 ---\|\|--- | 结果位大于“0” |
| | >=0 ---\|\|--- | 结果位大于等于“0” |
| | <=0 ---\|\|--- | 结果位小于等于“0” |
| | <0 ---\|\|--- | 结果位小于“0” |
| | <>0 ---\|\|--- | 结果位不等于“0” |
| | BR ---\|\|--- | 异常位二进制结果 |
| | OS ---\|\|--- | 存储溢出异常位 |
| | OV ---\|\|--- | 溢出异常位 |
| | UO ---\|\|--- | 无序异常位 |
| | ==0 ---\|/\|--- | 结果位取反等于“0” |
| | >0 ---\|/\|--- | 结果位取反大于“0” |
| | >=0 ---\|/\|--- | 结果位取反大于等于“0” |
| | <=0 ---\|/\| --- | 结果位取反小于等于“0” |
| | <0 ---\|/\|--- | 结果位取反小于“0” |
| | <>0 ---\|/\|--- | 结果位取反不等于“0” |
| | BR ---\|/\|--- | 异常位二进制结果取反 |
| | OS ---\|/\|--- | 存储溢出异常位取反 |
| | OV ---\|/\|--- | 溢出异常位取反 |
| | UO ---\|/\|--- | 无序异常位取反 |

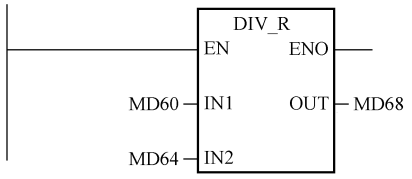
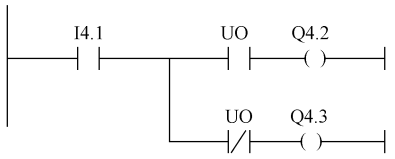
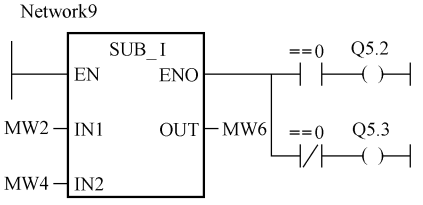
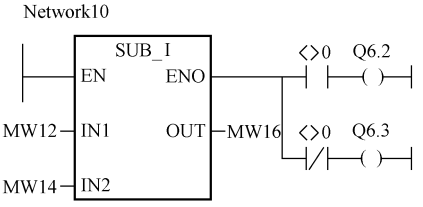
状态字中每个位的含义在前面已经做过介绍，这里不再描述，使用状态位指令的示例程序见表 6-25。

从下面的示例程序中，可以看到 LAD 程序和指令可以自由转换为 STL 程序，LAD 编程语言使用的状态位指令实现的功能在 STL 中也可以实现。

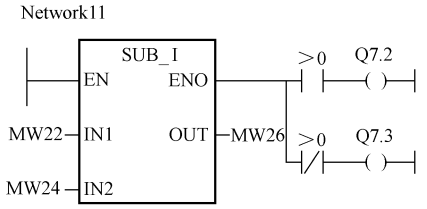
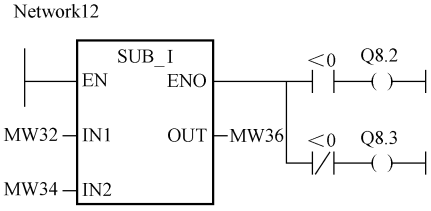
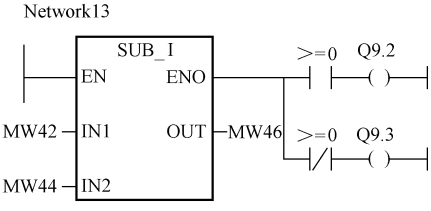
**表 6-25 状态位指令的示例程序**

| LAD | STL | 程 序 说 明 |
|---|---|---|
| 基于 BR 位指令 | | |
| Network1<br>M1.2 —— (SAVE)<br>Network 2<br>I1.1 —— BR —— Q1.1 ( )<br>I1.2 —— BR —— /<br>(BR 与 Q1.1 之间为并联) | **Network 1**<br>A M 1.2<br>SAVE<br>**Network 2**<br>A I 1.1<br>A BR<br>O<br>A I 1.2<br>AN BR<br>= Q 1.1 | 使用 LAD 编程时，BR 位用于指示函数调用及指令块输出 ENO 的状态，调用 SAVE 指令或其他字操作指令，如 MOVE、运算指令可以赋值 BR 位为 1。使用 BR 指令可以检测 BR 位的状态<br>在程序段 1 中，如果 M1.2 为 1，则 BR 位为 1<br>在程序段 2 中，如果 I1.1 与 BR 位同时为 1 或 I1.2 为 1，BR 位为 0 时，都将赋值 Q1.1 为 1 |
| 基于 OV 位指令 | | |
| Network3<br>MUL_I<br>EN ENO<br>MW20 — IN1 OUT — MW24<br>MW22 — IN2<br>Network4<br>I2.1 —— OV —— Q2.2 ( )<br>OV —— Q2.3 ( ) | **Network 3**<br>L MW 20<br>L MW 22<br>* I<br>T MW 24<br>**Network 4**<br>A I 2.1<br>A OV<br>= Q 2.2<br>A I 2.1<br>AN OV<br>= Q 2.3 | OV 位判断最近的运算结果是否溢出，如果溢出赋值 OV 位为 1，溢出消除，赋值 OV 位为 0<br>在程序段 3 中，如果 MW20 乘以 MW22 溢出，则赋值 OV 位为 1<br>在程序段 4 中，如果 I2.1 与 OV 同时为 1，则赋值 Q2.2 为 1，如果 I2.1 为 1，OV 位为 0，则赋值 Q2.3 为 1 |
| 基于 OS 位指令 | | |
| Network5<br>M1.2<br>ADD_I<br>EN ENO<br>MW30 — IN1 OUT — MW34<br>MW32 — IN2<br>ADD_I<br>EN ENO<br>MW36 — IN1 OUT — MW40<br>MW38 — IN2<br>Network6<br>I3.1 —— OS —— Q3.2 ( )<br>OS —— Q3.3 ( ) | **Network 5**<br>A M 1.2<br>JCN p001<br>L MW 30<br>L MW 32<br>+ I<br>T MW 34<br>L MW 36<br>L MW 38<br>+ I<br>T MW 40<br>p001: NOP 0<br>**Network 6**<br>A I 3.1<br>= Q 3.2<br>A I 3.1<br>AN OS<br>= Q 3.3 | OS 位判断上面的运算结果是否溢出，如果其中一个运算结果溢出，则赋值 OS 位为 1；所有溢出消除，则赋值 OS 位为 0<br>在程序段 5 中，两个运算，如果有一个运算结果溢出，则赋值 OS 为 1<br>在程序段 6 中，如果 I3.1 与 OS 同时为 1，则赋值 Q3.2 为 1；如果 I3.1 为 1，OS 位为 0，则赋值 Q3.3 为 1 |

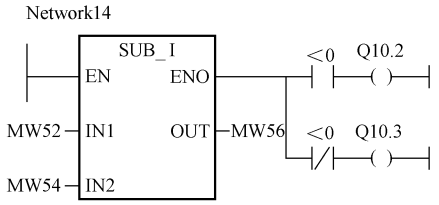
（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 基于 UO 位指令（CC1、CC0 位判断） | | |
| Network7<br>  <br>Network8<br>  | **Network 7**<br>L MD 60<br>L MD 64<br>/R<br>T MD 68<br><br>**Network 8**<br>A I 4.1<br>A UO<br>= Q 4.2<br><br>A I 4.1<br>AN UO<br>= Q 4.3 | UO 位判断最近的浮点运算是否有效，例如无效的浮点值，或浮点指令中除以 0 值，如果浮点值无效，则赋值 UO 位为 1，错误消除，赋值 UO 位为 0<br>在程序段 7 中，如果 MD64 为 0，则赋值 UO 为 1<br><br>在程序段 8 中，如果 I4.1 与 UO 同时为 1，则赋值 Q4.2 为 1，如果 I4.1 为 1，UO 位为 0，则赋值 Q4.3 为 1 |
| 基于运算结果等于 0 的判断指令（CC1 = 0、CC0 = 0） | | |
| Network9<br>  | **Network 9**<br>L MW 2<br>L MW 4<br>-I<br>T MW 6<br>AN OV<br>SAVE<br>CLR<br>A BR<br>A ==0<br>= Q 5.2<br><br>A BR<br>AN ==0<br>= Q 5.3 | 如果算术运算结果等于 0（CC1 = 0，CC0 = 0），则触点 ==0 - \| \| - 为 1；如果运算结果不等于 0，则触点 = =0 - \|/\| - 为 1<br><br>在程序段 9 中，如果 MW6 等于 0，则 Q5.2 为 1，反之 Q5.3 为 1 |
| 基于运算结果不等于 0 的判断指令（CC1 <> CC0） | | |
| Network10<br>  | **Network 10**<br>L MW 12<br>L MW 14<br>-I<br>T MW 16<br>AN OV<br>SAVE<br>CLR<br>A BR<br>A <>0<br>= Q 6.2<br><br>A BR<br>AN <>0<br>= Q 6.3 | 如果算术运算结果不等于 0（CC1 <> CC0），则触点 <>0 - \| \| - 为 1；如果运算结果等于 0，则触点 <>0 - \|/\| - 为 1<br><br>在程序段 10 中，如果 MW16 不等于 0，则 Q6.2 为 1，反之 Q6.3 为 1 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 基于运算结果大于 0 的判断指令（CC1 > CC0） | | |
|  | **Network 11**<br>L MW 22<br>L MW 24<br>-I<br>T MW 26<br>AN OV<br>SAVE<br>CLR<br>A BR<br>A >0<br>= Q 7.2<br><br>A BR<br>AN >0<br>= Q 7.3 | 如果算术运算结果大于 0（CC1 > CC0），则触点 >0 - \| \| - 为 1，否则触点 >0 - \|/\| - 为 1<br><br>在程序段 11 中，如果 MW26 大于 0，则 Q7.2 为 1，反之 Q7.3 为 1 |
| 基于运算结果小于 0 的判断指令（CC1 < CC0） | | |
|  | **Network 12**<br>L MW 32<br>L MW 34<br>-I<br>T MW 36<br>AN OV<br>SAVE<br>CLR<br>A BR<br>A <0<br>= Q 8.2<br><br>A BR<br>AN <0<br>= Q 8.3 | 如果算术运算结果小于 0（CC1 < CC0），则触点 <0 - \| \| - 为 1，否则触点 <0 - \|/\| - 为 1<br><br>在程序段 12 中，如果 MW36 小于 0，则 Q8.2 为 1，反之 Q8.3 为 1 |
| 基于运算结果大于等于 0 的判断指令（CC1 >= CC0） | | |
|  | **Network 13**<br>L MW 42<br>L MW 44<br>-I<br>T MW 46<br>AN OV<br>SAVE<br>CLR<br>A BR<br>A >=0<br>= Q 9.2<br>A BR<br>AN >=0<br>= Q 9.3 | 如果算术运算结果大于等于 0（CC1 >= CC0），则触点 > =0 - \| \| - 为 1，否则触点 >=0 - \|/\| - 为 1<br><br>在程序段 13 中，如果 MW46 大于等于 0，则 Q9.2 为 1，反之 Q9.3 为 1 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| 基于运算结果小于等于0的判断指令（CC1 <= CC0） | | |
|  | **Network 14**<br>L　MW　52<br>L　MW　54<br>-I<br>T　MW　56<br>AN　OV<br>SAVE<br>CLR<br>A　BR<br>A　<0<br>=　Q　10.2<br><br>A　BR<br>AN　<0<br>=　Q　10.3 | 如果算术运算结果小于等于0（CC1 <= CC0），则触点 <=0 -\|\|- 为1，否则触点 <=0 -\|/\|- 为1<br><br>在程序段14中，如果MW56小于等于0，则Q10.2为1，反之Q10.3为1 |

## 6.14　定时器指令

在CPU的系统存储器中，为定时器保留有存储区，每一定时器占用一个16位的字。具体能够使用定时器的个数由具体的CPU类型决定。定时器指令见表6-26。

**表6-26　定时器指令**

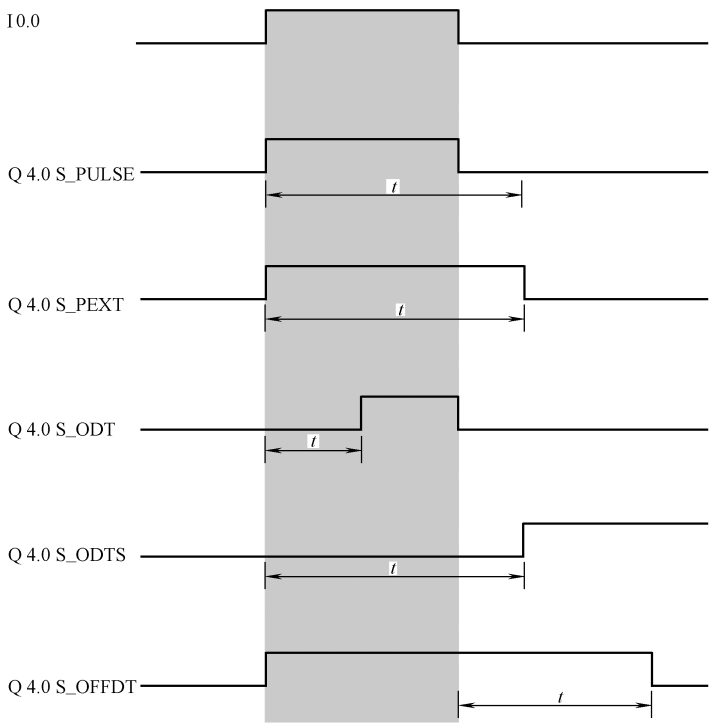
| | LAD | 说　明 | STL | 说　明 |
|---|---|---|---|---|
| 定时器指令 | S_PULSE | 脉冲S5定时器 | SP | 脉冲定时器 |
| | S_PEXT | 扩展脉冲S5定时器 | SE | 扩展脉冲定时器 |
| | S_ODT | 接通延时S5定时器 | SD | 接通延时定时器 |
| | S_ODTS | 保持型接通延时S5定时器 | SS | 带保持的接通延时定时器 |
| | S_OFFDT | 断电延时S5定时器 | SF | 断电延时定时器 |
| | ---[ SP ] | 脉冲定时器输出 | | |
| | ---[ SE ] | 扩展脉冲定时器输出 | | |
| | ---[ SD] | 接通延时定时器输出 | | |
| | ---[ SS ] | 保持型接通延时定时器输出 | | |
| | ---[ SF ] | 断开延时定时器输出 | | |
| | | | FR | 定时器允许（例如FR T0） |
| | | | L | 以整数形式把当前的定时器值写入ACCU1（例如：L T32） |
| | | | LC | 把当前的定时器值以BCD码形式装入ACCU1（例如LC T32） |
| | | | R | 复位定时器（例如：R T32） |

在LAD编程语言中，对定时器的操作指令分为定时器指令，如S_PULSE（脉冲定时器）和定时器线圈指令，如-(SP)（脉冲定时器输出），定时器指令为一个指令块，包含触发条件、定时器复位、预置值等与定时器所有相关的条件参数，定时器线圈指令将与定时器相关的条件参数分开使用，可以在不同的程序段中对定时器参数进行赋值和读取；使用STL

编程语言，定时器指令与 LAD 中的定时器线圈指令使用方式相同，除此之外，FR 指令可以重新启动定时器，例如设定定时器初值需要一个沿触发信号，如果触发信号常为 1，不能再次触发设定指令，使用 FR 指令，可以清除定时器的沿存储器，常 1 的触发信号可以再次产生沿信号并触发定时器重新开始定时，FR 指令在实际编程中很少使用；L 指令以整数的格式将定时器的定时剩余值写入到累加器 1 中，LC 指令以 BCD 码的格式将定时器的定时剩余值和时基一同写入到累加器 1 中；使用普通复位指令 R 可以将定时器复位（禁止启动）。

在前面的章节中已经介绍了定时器的数据类型，定时器使用的时间值为 BCD 码，给定时器赋值可以带有时基格式，例如 W#16#TXYZ，T 为时基值，XYZ 为时间值（BCD 码），总的定时时间为 T * XYZ，一个字的 12 位、13 位（T 的最低两位）组合选择时基值，00 表示时基为 10ms，01 表示时基为 100ms，10 表示时基为 1s，11 表示时基为 10s，如 W#16#1234转换时间值为 100 * 234ms = 23s400ms；定时器赋值也可以直接输入时间常数，格式为 S5T#aH_bM_cS_dMS ，a 为小时值，b 为分钟值，c 为秒值，d 为毫秒值，如 S5T#23s400ms，时基根据输入的时间长短自动选择，例如 10ms 到 9s_990ms 的分辨率为 10ms（时基为 10ms，时间的最小变化为 10ms），1s 到 16m_39s 的分辨率为 1s（时基为 1s，时间的最小变化为 1s）。

定时器指令中包括 5 种类型的定时器，对于定时应用必须选择合适的类型，不同类型的定时器实现的功能如图 6-23 所示。




注：
I0.0 为输入信号，Q4.0 定时器输出，*t* 为定时时间。

图 6-23　定时器的类型

1．S_PULSE

脉冲定时器，输入信号变为1时，触发定时器开始定时，并输出为1，输出信号保持为1的时间为设定的定时时间 $t$，如果输入信号在设定的定时时间内变为0，则定时器输出为0，与定时时间长短无关。S_PULSE 定时器的时序如图6-24所示。

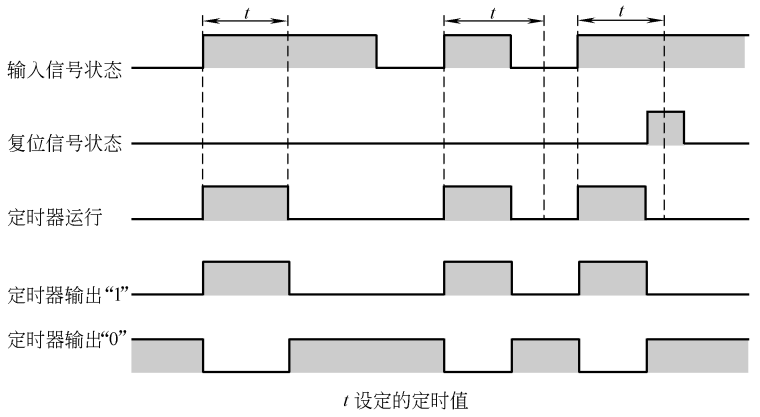



图6-24　S_PULSE 定时器时序图

2．S_PEXT

扩展脉冲定时器，输入信号变为1时，触发定时器开始定时并输出为1，输出信号保持为1的时间为设定的定时时间 $t$，与输入信号为1的时间长短无关。定时器定时期间，输入信号从0变1将再次触发定时器重新开始定时，定时器输出保持为1直到定时器定时停止。S_PEXT 定时器的时序如图6-25所示。

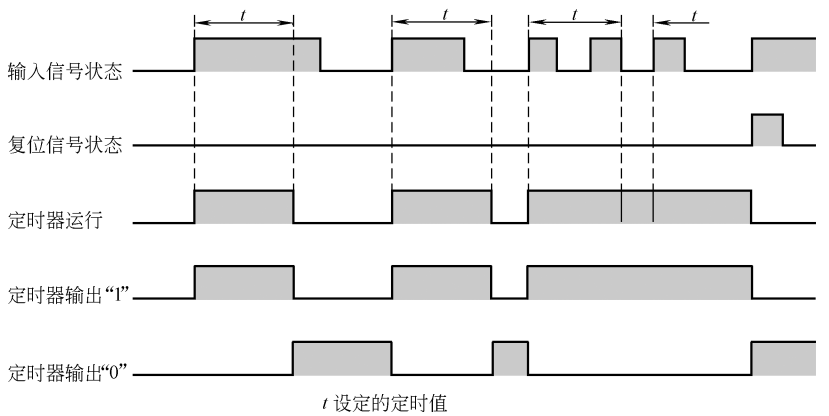



图6-25　S_PEXT 定时器时序图

3．S_ODT

接通延时定时器，输入信号变为1时，触发定时器开始定时，只有在设定的延时时间以后，并且输入信号仍然为1时，才能触发定时器输出为1。S_ODT 定时器的时序如图6-26

所示。

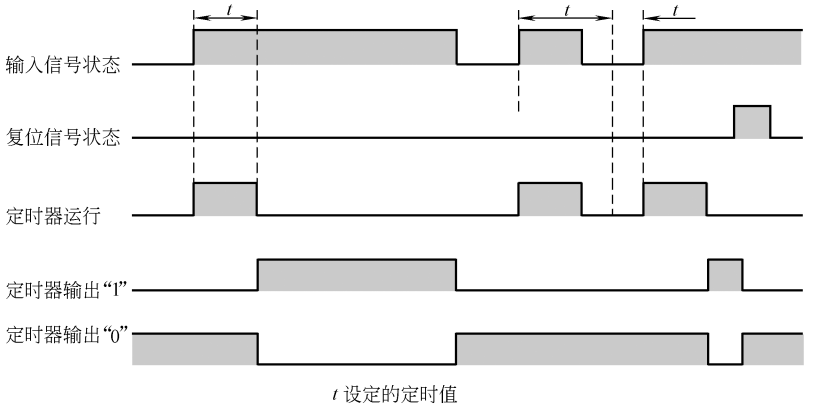



图 6-26　S_ODT 定时器时序图

4．S_ODTS

保持型接通延时定时器，输入信号为 1 时，触发定时器开始定时，在设定的延时时间以后触发定时器输出为 1，与输入信号为 1 的时间长短无关。定时器输出只有复位以后，才能再次触发定时功能。S_ODTS 定时器的时序如图 6-27 所示。

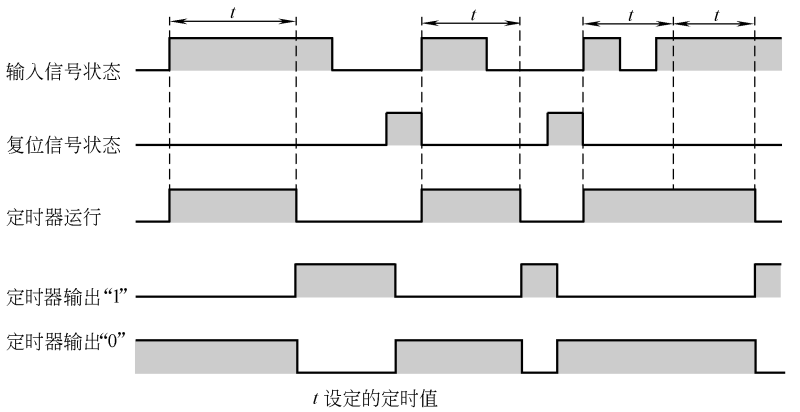



图 6-27　S_ODTS 定时器时序图

5．S_OFFDT

断电延时定时器，输入信号为 1 时，定时器输出为 1，输入信号从 1 变为 0，触发定时器开始定时，在设定的延时时间以后，赋值定时器输出为 0。定时器定时期间，输入信号从 0 变为 1 时将复位定时器，只有输入信号再次从 1 变为 0 时才能触发定时器开始定时，定时器输出在输入信号为 1 或定时器定时没有完成时，保持为 1。S_OFFDT 定时器的时序如图 6-28所示。

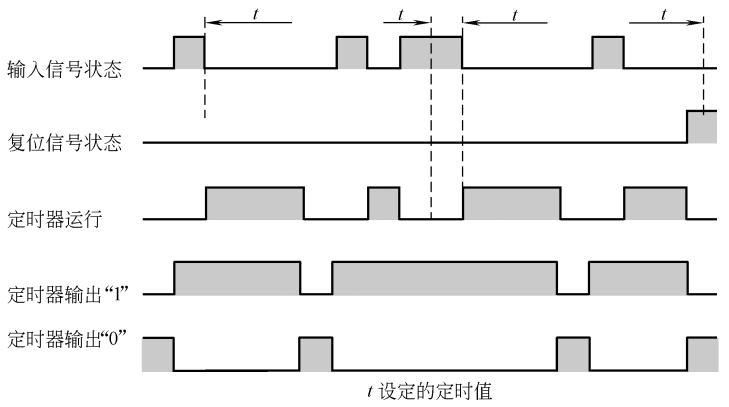


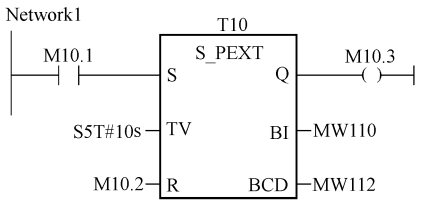
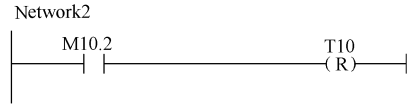
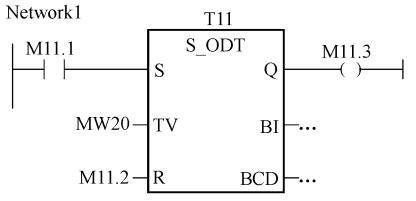

图 6-28 S_OFFDT 定时器时序图

使用定时器指令的示例程序见表 6-27。

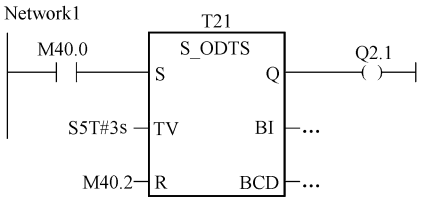
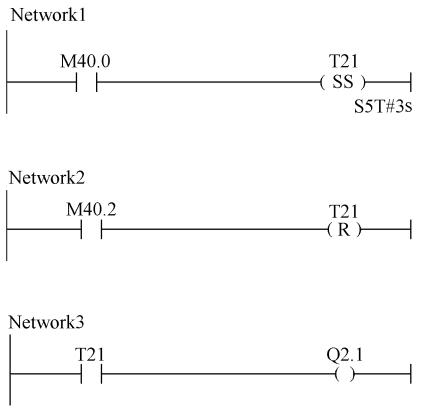
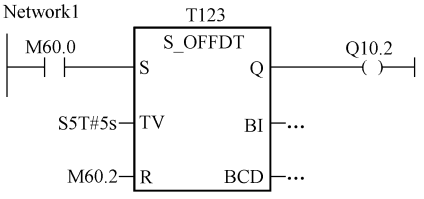
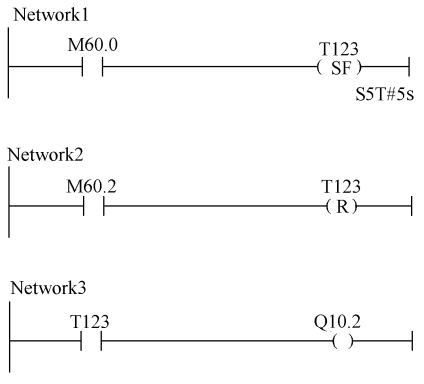
**表 6-27 定时器指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| S_PULSE 定时器指令 | | |
| Network1<br>T1<br>M1.1 — S_PULSE — M1.3<br>S — Q — ( )<br>S5T#10s — TV — BI — MW102<br>M1.2 — R — BCD — MW104 | **Network 1**<br>A M 1.1<br>L S5T#10s<br>SP T 1<br>A M 1.2<br>R T 1<br>L T 1<br>T MW 102<br>LC T 1<br>T MW 104<br>A T 1<br>= M 1.3 | 在程序段 1 中，设定定时器 T1 的定时时间为 10s，如果 M1.1 为 1（大于 10s），启动定时器，M1.3 产生一个 10s 脉冲信号；如果 M1.2 为 1，复位 T1 和 M1.3，定时器将不能启动，定时器运行时，将定时器定时剩余值以整数形式传送到 MW102 中，以 BCD 码的格式传送到 MW104 中 |
| S_PULSE 定时器线圈指令（与 S_PULSE 定时器指令完成相同的功能） | | |
| Network1<br>M1.1 — T1 ( SP )<br>S5T#10s | **Network 1**<br>A M 1.1<br>L S5T#10s<br>SP T 1 | 在程序段 1 中，M1.1 为 1 时，设定定时器 T1 的定时时间为 10s，并启动定时器 |
| Network2<br>M1.2 — T1 ( R ) | **Network 2**<br>A M 1.2<br>R T 1 | 在程序段 2 中，如果 M1.2 为 1，则复位 T1 输出 |
| Network3<br>T1 — M1.3 ( ) | **Network 3**<br>A T 1<br>= M 1.3 | 在程序段 3 中，将定时 T1 输出，并赋值到 M1.3 中 |
| Network4<br>MOVE<br>EN — ENO<br>T1 — IN — OUT — MW102 | **Network 4**<br>L T 1<br>T MW 102 | 在程序段 4 中，将定时器定时剩余值以整数形式传送到 MW102 中，LAD 不能实行 BCD 码的传送，STL 使用 LC 指令可以传送 BCD 码的时间格式 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| S_PEXT 定时器指令 | | |
|  | **Network 1**<br>A M 10.1<br>L S5T#10s<br>SE T 10<br>A M 10.2<br>R T 10<br>L T 10<br>T MW 110<br>LC T 10<br>T MW 112<br>A T 10<br>= M 10.3 | 在程序段 1 中，设定定时器 T10 的定时时间为 10s，如果 M10.1 为 1，启动定时器，M10.3 产生一个 10s 脉冲信号（与 M10.1 为 1 的时间长短无关）；如果 M10.2 为 1，则复位 T10 和 M10.3，定时器将不能启动，定时器运行时，将定时器定时剩余值以整数形式传送到 MW110 中，以 BCD 码的格式传送到 MW112 中 |
| S_PEXT 定时器线圈指令（与 S_PEXT 定时器指令完成相同的功能） | | |
|  | **Network 1**<br>A M 10.1<br>L S5T#10s<br>SE T 10 | 在程序段 1 中，M10.1 为 1 时，设定定时器 T10 的定时时间为 10s，并启动定时器，定时器输出 |
|  | **Network 2**<br>A M 10.2<br>R T 10 | 在程序段 2 中，如果 M10.2 为 1，则复位 T10 输出 |
|  | **Network 3**<br>A T 10<br>= M 10.3 | 在程序段 3 中，将定时器 T10 输出，并赋值到 M10.3 中 |
| S_ODT 定时器指令 | | |
|  | **Network 1**<br>A M 11.1<br>L MW 20<br>SD T 11<br>A M 11.2<br>R T 11<br>A T 11<br>= M 11.3 | 在程序段 1 中，定时器 T11 的定时时间为变量存储于 MW20 中，如果 M11.1 为 1，启动定时器，延时时间（存储于 MW20 的时间值）到并且 M11.1 仍然为 1，M11.3 接通；如果 M11.2 为 1，复位 T11 和 M11.3，定时器将不能启动 |
| S_ODT 定时器线圈指令（与 S_ODT 定时器指令完成相同的功能） | | |
|  | **Network 1**<br>A M 11.1<br>L MW 20<br>SE T 11 | 在程序段 1 中，M11.1 为 1 时，启动定时器 T11，定时器 T11 延时（存储于 MW20 的时间值）输出 |
|  | **Network 2**<br>A M 11.2<br>R T 11 | 在程序段 2 中，如果 M11.2 为 1，则复位 T11 输出 |
|  | **Network 3**<br>A T 11<br>= M 11.3 | 在程序段 3 中，将定时器 T11 输出，并赋值到 M11.3 中 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| S_ODTS 定时器指令 | | |
|  | **Network 1**<br>A M 40.0<br>L S5T#3s<br>SS T 21<br>A M 40.2<br>R T 21<br>A T 21<br>= Q 2.1 | 在程序段1中，如果M40.0为1时，启动定时器T21并开始定时，Q2.1延时3s接通而与M40.0的状态无关，如果M40.2为1，则复位T21和Q2.1，定时器将不能启动 |
| S_ODTS 定时器线圈指令（与S_ODTS定时器指令完成相同的功能） | | |
|  | **Network 1**<br>A M 40.0<br>L S5T#3s<br>SS T 21<br><br>**Network 2**<br>A M 40.2<br>R T 21<br><br>**Network 3**<br>A T 21<br>= Q 2.1 | 在程序段1中，如果M40.0为1时，启动定时器T21并开始定时，定时器T21延时3s接通而与M40.0为1的状态无关<br><br>在程序段2中，如果M40.2为1，则复位T21和Q2.1，定时器将不能启动<br><br>在程序段3中，将定时器T21输出，并赋值到Q2.1中 |
| S_OFFDT 定时器指令 | | |
|  | **Network 1**<br>A M 60.0<br>L S5T#5s<br>SF T 123<br>A M 60.2<br>R T 123<br>A T 123<br>= Q 10.2 | 在程序段1中，如果M60.0为1时，则定时器T123输出，Q10.2为1；如果M60.0从1变为0，启动定时器开始定时，Q10.2延时5s后断开为0。如果M60.2为1，则复位T123和Q10.2，定时器将不能启动 |
| S_OFFDT 定时器线圈指令（与S_OFFDT定时器指令完成相同的功能） | | |
|  | **Network 1**<br>A M 60.0<br>L S5T#5s<br>SF T 123<br><br>**Network 2**<br>A M 60.2<br>R T 123<br><br>**Network 3**<br>A T 123<br>= Q 10.2 | 在程序段1中，如果M60.0为1，则定时器T123输出为1；如果M60.0从1变为0，启动定时器开始定时，定时器T123延时5s后断开为0<br><br>如果M60.2为1，则复位T123定时器将不能启动<br><br>在程序段3中，将定时器T123输出，并赋值到Q10.2中 |

**注意：**

一个定时器不能在同一时刻多次调用、运行，如果程序中多处使用同一个定时器，注意定时器启动的时序。

## 6.15 字逻辑指令

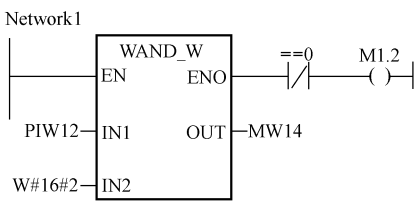
字逻辑指令是对多个位信号进行操作，将两个 WORD（字）或 DWORD（双字）逐位进行“与”、“或”、“异或”逻辑运算。字逻辑指令见表 6-28。

**表 6-28 字逻辑指令**

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 字逻辑指令 | WAND_DW | 双字和双字相“与” | AD | 双字“与”（32 位） |
| | WAND_W | 字和字相“与” | AW | 字“与”操作（16 位） |
| | WOR_DW | 双字和双字相“或” | OD | 双字或操作（32 位） |
| | WOR_W | 字和字相“或” | OW | 单字或操作（16 位） |
| | WXOR_DW | 双字和双字相“异或” | XOD | 双字异或操作（32 位） |
| | WXOR_W | 字和字相“异或” | XOW | 单字异或操作（16 位） |

“与”操作可以判断两个字或双字在相同的位数上有多少位为 1，通常用于变量的过滤，例如一个字变量与常数 W#16#00FF 相“与”，则可以将字变量中的高字节过滤为 0；“或”操作可以判断两个字或双字中为 1 位的个数；“异或”操作可以判断两个字或双字有多少位不相同。字逻辑指令影响状态字的 CC1 位，如果字逻辑结果等于 0，CC1 位为 0；如果字逻辑结果不等于 0，CC1 位为 1，字逻辑指令可以与状态位指令结合使用，例如前面介绍的立即读的方法就是利用“与”指令与状态位指令结合使用判断一个位的信号。使用字逻辑指令的示例见表 6-29。

**表 6-29 字逻辑指令的示例程序**

| LAD | STL | 程序说明 |
|---|---|---|
| “与”操作 | | |
|  | **Network 1**<br>A (<br>L PIW 12<br>L W#16#2<br>AW<br>T MW 14<br>SET<br>SAVE<br>CLR<br>A BR<br>)<br>AN ==0<br>= M 1.2 | 在程序段 1 中，实现立即读操作，不能单独访问外设地址的位信号，将 PIW12 与参数 W#16#2 相“与”，状态位 CC1 对逻辑结果进行判断，如果 PIW12 的第 2 位（过程映像区地址为 I13.1）为 1（CC1 = 1），则赋值 M1.2 为 1 |

（续）

| LAD | STL | 程序说明 |
|---|---|---|
| “或”操作 | | |
| Network2<br>WOR_W<br>EN ENO<br>MW4 IN1 OUT MW8<br>MW6 IN2 | **Network 2**<br>L MW 4<br>L MW 6<br>OW<br>T MW 8<br><br><br><br>**Network 3**<br>L IW 20<br>OW W#16#FFF<br>JP NEXT | 在程序段2中将两个字变量MW4与MW6相“或”，结果传送到MW8中<br>如果MW4二进制的值为2#0000 1010 1101 0011，MW6二进制的值为2#1001 0000 0001 1001，存储于MW8中的操作结果为2#1001 1010 1101 1011<br>在程序段3为STL指令，如果IW20与W#16#FFF相“或”逻辑结果不等于0（CC1 = 1），跳转到NEXT标签指定的程序段 |
| “异或”操作 | | |
| Network4<br>WXOR_W<br>EN ENO<br>MW12 IN1 OUT MW16<br>MW14 IN2 | **Network 4**<br>L MW 12<br>L MW 14<br>XOW<br>T MW 16<br><br><br><br>**Network 5**<br>L IW 30<br>XOW W#16#46<br>JP NEXT | 在程序段4中，将两个字变量MW12与MW14进行“异或”操作，结果传送到MW16中。如果MW12二进制的值为2#0000 1010 1101 0011，MW14二进制的值为2#1001 0000 0001 1001，存储于MW16中的操作结果为2#1001 1010 1100 1010<br>在程序段5为STL指令，如果IW30与W#16#46进行“异或”操作，逻辑结果不等于0（CC1 = 1），跳转到NEXT标签指定的程序段 |

## 6.16 累加器指令

累加器指令对累加器1（ACCU1）、累加器2（ACCU2）、累加器3（ACCU3）、累加器4（ACCU4）进行操作，并且只适合STL编程语言。S7-300 CPU只有累加器1和累加器2，S7-400 CPU具有4个累加器，累计器3、4的使用减少了中间运算变量的使用。对累加器1、2进行数据的装载和传送，使用L、T就可以完成；对累加器3、4进行操作，必须使用累加器指令。累加器指令见表6-30。

在前面已经介绍了累加器的使用，一个累加器占用32位，如果装载一个16位的字或整数数据，只占用累加器的低16位或低“字”，如果装载一个32位的双字、双整数、浮点数据，则将累加器占满，在下面示例程序说明中，ACCU N（1～4）表示累加器N，ACCU N—L表示累计器N的低16位。

1. TAK指令

TAK指令交换累加器1、2的内容，指令的使用参考下面的示例程序，示例程序中将两个值进行比较，然后使用值大的变量减值小的变量。

表 6-30 累加器指令

| | LAD | 说 明 | STL | 说 明 |
|---|---|---|---|---|
| 累加器指令 | | | TAK | 交换 ACCU1 和 ACCU2 的内容 |
| | | | PUSH | ACCU3→ACCU4，ACCU2→ACCU3，ACCU1→ACCU2（S7-400 CPU） |
| | | | PUSH | ACCU 1→ACCU 2（S7-300 CPU） |
| | | | POP | ACCU 1←ACCU 2，ACCU2←ACCU3，ACCU3←ACCU4（S7-400 CPU） |
| | | | POP | ACCU 1←ACCU 2（S7-300 CPU） |
| | | | ENT | ACCU3→ACCU4，ACCU2→ACCU3 |
| | | | LEAVE | ACCU3→ACCU2，ACCU4→ACCU3 |
| | | | INC | ACCU1 加 1 |
| | | | DEC | ACCU1 减 1 |
| | | | + AR1 | ACCU1 与 AR1 相加转载地址值到 AR1 |
| | | | + AR2 | ACCU1 与 AR2 相加转载地址值到 AR2 |
| | | | BLD | 程序显示指令 |
| | | | NOP 0 | 空操作 0 |
| | | | NOP 1 | 空操作 1 |
| | | | CAW | 改变 ACCU1 字中字节的次序（16 位） |
| | | | CAD | 改变 ACCU1 字节的次序（32 位） |

```
L     MW10     //装载 MW10 的内容到 ACCU 1-L
L     MW12     //装载 ACCU 1-L 的内容到 ACCU 2-L，装载 MW10 的内容到
                 ACCU 1-L
>I             //检测 ACCU 2-L（MW10）是否大于 ACCU 1-L（MW12）
JP    NEXT     //如果 ACCU 2（MW10）大于 ACCU 1（MW12），跳转到程序
                 标号 NEXT
TAK            //如果小于，交换 ACCU 1 和 ACCU 2 的内容
NEXT：-I       // 执行 ACCU 2-L 减 ACCU 1-L 操作
T     MW14     //将结果传送到 MW14
```

2. PUSH 指令

PUSH 指令在 S7-300 系列 PLC CPU 中使用时，将累加器 1 的值复制到累加器 2 中，累加器 1 中的值不变，指令在 S7-400 系列 PLC CPU 中使用时，将累加器 3 的值复制到累加器 4 中，将累加器 2 的值复制到累加器 3 中，将累加器 1 的值复制到累加器 2 中，累加器 1 中的值不变。

PUSH 指令执行前后累加器中值的变化如下：

| 内容 | ACCU1 | ACCU 2 | ACCU 3 | ACCU 4 |
|---|---|---|---|---|
| 执行 PUSH 指令前 | 值 A | 值 B | 值 C | 值 D |
| 执行 PUSH 指令后 | 值 A | 值 A | 值 B | 值 C |

指令的使用参考下面的示例程序：

```
L     MW10     //装载 MW10 的内容到 ACCU 1-L
PUSH           //将 ACCU 3 的内容复制到 ACCU 4，将 ACCU 2 的内容复制
```

到 ACCU 3，将 ACCU 1 的内容复制到 ACCU 2，ACCU 1 的值未变为 MW10

3. POP 指令

POP 指令与 PUSH 指令复制的方向相反，在 S7-300 系列 PLC CPU 中使用时，将累加器 2 的值复制到累加器 1 中，累加器 2 中的值不变；指令在 S7-400 系列 PLC CPU 中使用时，将累加器 2 的值复制到累加器 1 中，将累加器 3 的值复制到累加器 2 中，将累加器 4 的值复制到累加器 3 中，累加器 4 中的值不变。POP 指令执行前后累加器中值的变化如下：

| 内容 | ACCU 1 | ACCU 2 | ACCU 3 | ACCU 4 |
|---|---|---|---|---|
| 执行 POP 指令前 | 值 A | 值 B | 值 C | 值 D |
| 执行 POP 指令后 | 值 B | 值 C | 值 D | 值 D |

指令的使用参考下面的示例程序：

```
        TMD10          //将 ACCU 1 中的值（值 A）传送到 MD10 中
POP                    //复制 ACCU 2 中的值到 ACCU 1 中
        TMD14          //将 ACCU 1 中的值（值 B）传送到 MD14 中
```

4. ENT 指令

ENT 指令将累加器 3 的值复制到累加器 4 中，将累加器 2 的值复制到累加器 3 中，如果直接在 L 指令前使用，将运算的中间结果存储于 ACCU3 中。指令的使用参考下面的示例程序：

```
L     DBD0     //装载 DBD0 到 ACCU 1（浮点格式）
L     DBD4     //复制 ACCU 1 中的值到 ACCU 2，装载 DBD4 到 ACCU 1
               （浮点格式）
+R             //将 ACCU 1 和 ACCU 2 中的值相加，结果存储于 ACCU1
               中
L     DBD8     //复制 ACCU 1 中的值到 ACCU 2，装载 DBD8 到 ACCU 1
ENT            //复制 ACCU 3 中的值到 ACCU 4，复制 ACCU 2 中的值
               //（中间结果 DBD0 + DBD4）到 ACCU 3
L     DBD12    //复制 ACCU 1 中的值到 ACCU 2，装载 DBD12 到 ACCU
               1
-R             //ACCU2 减去 ACCU1 的值并存储于 ACCU1 中，复制
               ACCU 3中的值到ACCU 2，复制 ACCU 4 中的值到 ACCU 3
/R             //ACCU 2 中的值（DBD0 + DBD4）除以 ACCU 1 的值
               （DBD8 - DBD12）并将结果存储于 ACCU 1 中
T     DBD16    //将运算结果传送到 DBD16 中
```

5. LEAVE 指令

与 ENT 指令复制的方向相反，将累加器 3 的值复制到累加器 2 中，将累加器 4 的值复制到累加器 3 中，累加器 1、4 中的值不变。

6. INC 与 DEC 指令

INC 指令将累加器 1 低 8 位（ACCU 1-L-L）中存储的值加 1（8 位的整数值），DEC 指令将累加器 1 低 8 位（ACCU 1-L-L）中存储的值减 1（8 位的整数值），累加器 1 中其他位

保持不变，由于指令只对累加器 1 低 8 位进行操作，最大增减值为 255。指令的使用参考下面的示例程序：

```
L     MB22          //装载 MB22 到 ACCU 1
INC 15              //ACCU 1（MB22）增加 15，将结果存储于 ACCU 1-L-L
T     MB22          //将 ACCU 1-L-L 的值传送到 MB22 中
L     MB42          //装载 MB42 到 ACCU 1
DNC   20            // ACCU 1（MB42）减少 20，将结果存储于 ACCU 1-L-L
T     MB42          //将 ACCU 1-L-L 的值传送到 MB42 中
```

7. +AR1 与 +AR2 指令

+AR1 指令将累加器 1 中的值装载到地址寄存器 1 中，+AR2 指令将累加器 1 中的值装载到地址寄存器 2 中，指令后面可以直接定义地址指针，例如 +AR1 P#10.0，将 P#10.0 装载到地址寄存器 1 中，指令的使用参考下面的示例程序：

```
L     P#200.0           //装载地址指针 P#200.0 到 ACCU 1
+AR1                    //将 ACCU1 中的值（P#200.0）装载到地址寄存
                          器 1 中
+AR2  P#300.0           //将 P#300.0 装载到地址寄存器 2 中
L     MW [AR1, P#0.0]   //装载 MW200 到 ACCU 1
L     MW [AR2, P#0.0]   //ACCU1 的值复制到 ACCU2 中，装载 MW300 到
                          ACCU 1
-I                      //ACCU2-ACCU1
T     MW    4           //运算结果传送到 MW4 中
```

8. CAW 与 CAD 指令

CAW 指令将累加器 1 低字中包含的两个字节相互转换，CAD 指令将累加器 1 中包含的 4 个字节相互转换。

CAW 指令执行前后累加器 1 中值的变化如下：

| 内容 | ACCU 1-H-H | ACCU 1-H-L | ACCU 1-L-H | ACCU 1-L-L |
|---|---|---|---|---|
| 执行 CAW 指令前 | 值 A | 值 B | 值 C | 值 D |
| 执行 CAW 指令后 | 值 A | 值 B | 值 D | 值 C |

CAD 指令执行前后累加器 1 中值的变化如下：

| 内容 | ACCU 1-H-H | ACCU 1-H-L | ACCU 1-L-H | ACCU 1-L-L |
|---|---|---|---|---|
| 执行 CAD 指令前 | 值 A | 值 B | 值 C | 值 D |
| 执行 CAD 指令后 | 值 D | 值 C | 值 B | 值 A |

指令的使用参考下面的示例程序：

```
L     W#16#1234         //装载参数 W#16#1234 到 ACCU 1
CAW                     //转换 ACCU1-L 中两个字节存储值的次序
T     MW   10           //运算结果传送到 MW10 中。MW10 中的值为
                          W#16#3412
L     DW#16#12345678    //装载参数 W#16#12345678 到 ACCU 1
CAD                     //转换 ACCU1 中四个字节存储值的次序
```

```
T       MD   20          //运算结果传送到 MD20 中。MD20 中的值为
                        //DW#16#78563412
```

9. NOP 0、NOP1 与 BLD 指令

NOP 0、NOP 1、BLD 指令用于 LAD、FBD 编程语言的显示，当 LAD、FBD 编程语言转换为 STL 编程语言时自动产生空指令，无实际意义。

# 第 7 章　程　序　块

在同一个 CPU 中运行着两种程序，即操作系统程序和用户程序。

操作系统是固化在 CPU 中的程序，组织 CPU 内部所有的功能和与用户控制任务无关的运行次序。操作系统主要完成以下工作：

- 处理启动（暖启动和热启动）；
- 更新输入、输出过程映像区；
- 调用用户程序；
- 检测中断，并调用中断程序；
- 检测并处理错误；
- 管理存储区域；
- 与编程设备和其他通信设备的通信。

下传到 CPU 中的用户程序是为了完成特定的自动化任务，由用户编写。用户程序需要完成以下工作：

- 暖启动和热启动时的初始化工作；
- 处理过程数据（数字信号、模拟信号）；
- 对中断的响应；
- 对异常和错误的处理。

## 7.1　用户程序中的程序块

在用户程序中包含不同的程序块，不同类型的程序块实现的功能不同，STEP7 中包含程序块的类型见表 7-1。

**表 7-1　程序块类型**

| 程　序　块 | 功能简要描述 |
| --- | --- |
| 组织块（OB） | OB 决定用户程序的结构 |
| 系统函数块（SFB）与系统函数（SFC） | SFB、SFC 集成于 S7 CPU 中，通过调用可以访问一些重要的系统功能，例如 CPU 的时钟、CPU 的诊断缓存区 |
| 函数块（FB） | FB 可以使用户编写的函数带有“存储区” |
| 函数（FC） | FC 可以作为子程序，也可以作为经常调用的函数使用 |
| 背景数据块（Instance DB） | 背景 DB 与 SFB、FB 调用相关，在调用时自动生成，存储 SFB、FB 中的所有数据并作为它们的“存储区” |
| 数据块（DB） | DB 存储用户数据，与背景 DB 相比，DB 格式由用户定义，背景 DB 格式与 FB、SFB 相关 |

程序部分包含的 OB、FB、SFB、FC、SFC 作为逻辑程序程序块，每种类型的程序块允许的数量及每个程序块最大的容量与 CPU 的类型有关。

### 7.1.1 组织块与程序结构

OB 是 CPU 操作系统与用户程序的接口，被操作系统自动调用，CPU 通过组织块循环或者事件驱动控制用户程序的执行，此外 CPU 的启动及故障处理都要调用不同的组织块，在这些组织块中编写用户程序可以判断 CPU 及外部设备的状态。

PLC CPU 循环执行操作系统程序，操作系统程序在每一个循环中调用主程序 OB1 一次，因此在 OB1 中编写的用户程序循环执行，操作系统与主程序执行过程如图 7-1 所示。

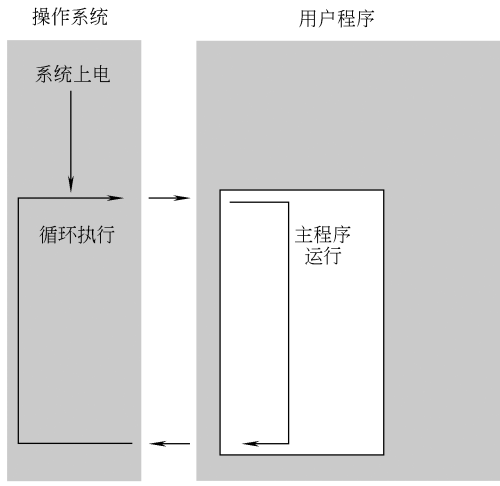



图 7-1 操作系统与主程序关系

循环执行的程序可以被高优先级的中断事件中断，如果中断事件出现，当前执行的程序在指令执行完成后（两个指令边界处）中断而执行相应中断程序，中断程序执行完成后跳回到中断处继续执行。不同的中断事件由操作系统触发不同的 OB，中断程序编写在相应的 OB 中，这样中断事件出现，执行中断 OB 中程序一次，然后回到中断点继续执行主程序，事件中断的过程如图 7-2 所示。

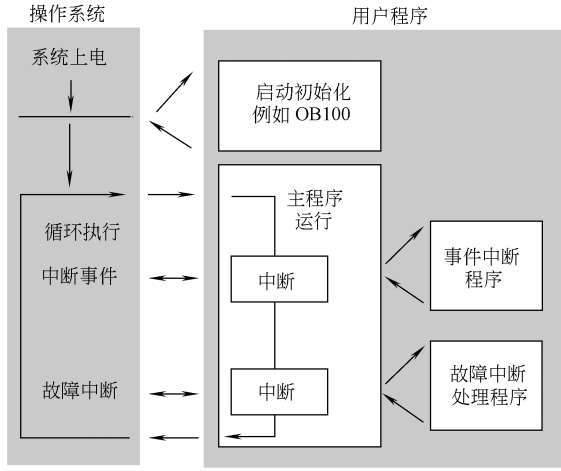



图 7-2 中断程序的执行

早先的程序设计中，通常使用线性化编程方式，即将所有的程序指令都写在主程序中以实现一个自动化控制任务，这样的编程方式不利于程序的查看、修改和调试，在与线性化编程方式相对应的结构化编程方式中，将整个控制任务划分为相对独立的控制任务，每个相对独立的控制任务可以对应结构化程序中的一个程序段或子程序（FC 或者 FB），OB1 通过调用这些程序块来完成整个自动化任务，两种编程方式及程序结构的对比如图 7-3 所示。

结构化的编程方式是由不同的程序块构成的，具有下列优点：

- 程序一目了然；
- 独立的程序段，可以标准化；
- 程序修改简单；
- 控制任务分开，设备调试方便。

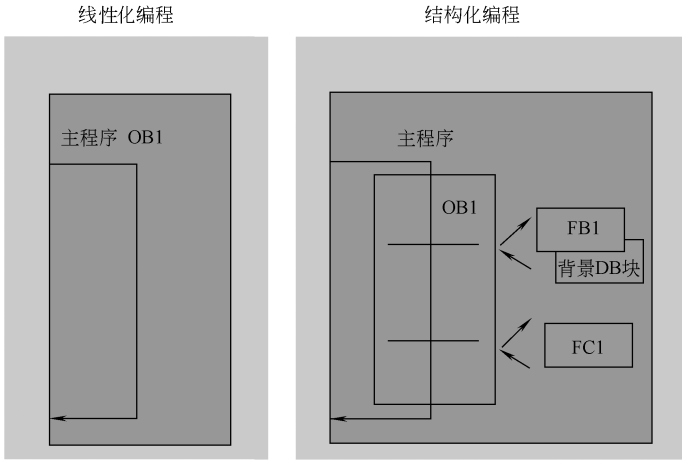



图 7-3 线性化编程与结构化编程

## 7.1.2 用户程序的分层调用

用户编写的函数或程序块必须在 OB 中调用才能执行，在一个程序块中可以使用指令调用其他程序块，被调用的程序块执行完成后返回原程序中断处继续运行，程序块的调用过程如图 7-4 所示。

调用的程序块类型只能是 OB、FB、FC，被调用的程序块可以是 FB、FC、SFB、SFC，OB 不能被调用。

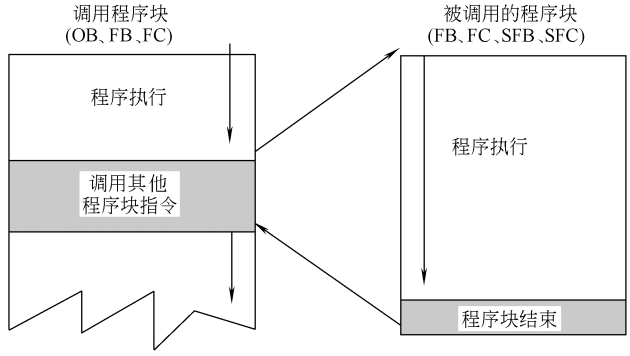



图 7-4 程序块的调用

在控制任务中，可以将工厂级控制任务划分为几个车间级控制任务，将车间级控制任务再划分为几组生产线的控制任务，将生产线的控制任务划分为几个电动机的控制。这样从上到下将控制任务分层划分，同样也可以将控制程序根据控制任务分层划分，每一层控制程序作为上一层控制程序的子程序，同时调用下一层的控制程序作为子程序，形成程序块的嵌套调用。用户程序的分层调用就是将整个程序按照控制工艺划分为小的子程序，按次序分层嵌套调用（嵌套深度参考 CPU 样本手册，通常不能超过 8 层），例如将一个控制任务划分为 3 个独立的子任务，在每个子任务下划分小的控制任务，程序的分层调用如图 7-5 所示。

3 个独立的子程序分别为 FB10、FB2 和 FC3，在 FB2 中又划分两层控制程序，分别为 FB1 和 SFC1，这样控制任务的分层通过程序块或子程序的嵌套调用实现，数据块 DB10 的使用与嵌套无关。用户程序执行次序为 OB1→FB10 + 背景 DB7→FC1→FB2 + 背景 DB2→FB1 + 背景 DB1→SFC1→FC3→OB1。用户程序的分层调用是结构化编程方式的延伸。

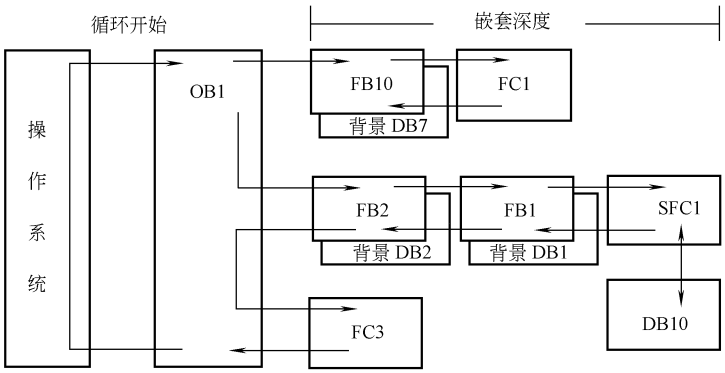



图 7-5 用户程序的分层调用

## 7.2 组织块

OB 由操作系统自动调用，同时执行编写在组织块中的用户程序，用户程序中没有组织块不可能执行用户程序，所以组织块最基本的功能就是调用用户程序。组织块代表 CPU 的系统功能，不同类型的组织块完成不同的系统功能，一个 CPU 功能越强大，相应的组织块个数及类型越多，例如 S7-300 系列 PLC CPU 硬件中断只能触发 OB40，在 S7-400 系列 PLC CPU 中硬件中断可以选择触发的 OB（OB40 ~ OB47）。通常情况下，在 S7-300 系列 PLC 一个组织块中可以编写最大的程序容量为 16KB，在 S7-400 系列 PLC 一个组织块中可以编写 64KB 的用户程序。

### 7.2.1 组织块的类型与优先级

系统为每个组织块分配相应的优先级，如果系统同时触发调用多个组织块，通过优先级可以决定组织块之间的调用次序，同时也决定每个组织块中用户程序的执行次序，通常情况下组织块号码越大，优先级越高，例如 CPU 硬件中断为 OB40，循环执行为 OB1，硬件中断事件可以中断 OB1 的执行并触发 OB40，OB40 执行一个周期后返回 OB1 中断处，再次循环执行用户程序。除主程序循环 OB1 外，其他的组织块都是由事件触发的中断，组织块的中断类型及优先级分类见表 7-2。

**表 7-2 OB 类型及优先级**

| 类 型 | OB | 优先级（默认） |
|---|---|---|
| 主程序循环 | OB1 | 1 |
| 日期中断 | OB10 ~ OB17 | 2 |
| 时间延迟中断 | OB20<br>OB21<br>OB22<br>OB23 | 3<br>4<br>5<br>6 |

（续）

| 类　　型 | OB | 优先级（默认） |
|---|---|---|
| 循环中断 | OB30<br>OB31<br>OB32<br>OB33<br>OB34<br>OB35<br>OB36<br>OB37<br>OB38 | 7<br>8<br>9<br>10<br>11<br>12<br>13<br>14<br>15 |
| 硬件中断 | OB40<br>OB41<br>OB42<br>OB43<br>OB44<br>OB45<br>OB46<br>OB47 | 16<br>17<br>18<br>19<br>20<br>21<br>22<br>23 |
| DPV1 中断 | OB55<br>OB56<br>OB57 | 2<br>2<br>2 |
| 多处理器中断 | OB60 多处理器 | 25 |
| PROFIBUS 时钟同步中断 | OB61<br>OB62<br>OB63<br>OB64 | 25 |
| 工艺同步处理中断 | OB65 | 25 |
| 冗余错误中断 | OB70 I/O 冗余错误（只适用于 H 系统）<br>OB72 CPU 冗余错误（只适用于 H 系统） | 25<br>28 |
| 异步故障中断 | OB80 时间错误<br>OB81 电源故障<br>OB82 诊断中断<br>OB83 插入/拔出模板中断<br>OB84 CPU 硬件故障<br>OB85 程序循环错误<br>OB86 机架故障<br>OB87 通信错误 | 26<br>（或者如果异步错误 OB 存在于启动程序中则为 28） |
| 处理中断 | OB88 | 28 |
| 背景循环中断 | OB90 | 29 * |
| 启动中断 | OB100 暖启动<br>OB101 热启动<br>OB102 冷启动 | 27<br>27<br>27 |
| 同步错误中断 | OB121 编程错误<br>OB122 访问错误 | 引起错误 OB 的优先级 |

注：* 优先级别 29 相应于优先级 0.29；背景循环的优先级低于自由循环（OB1）。

1. 自由循环组织块 OB1

S7 CPU 启动完成后，操作系统循环执行 OB1，OB1 执行完成后，操作系统再次启动 OB1。在 OB1 中可以调用 FB、SFB、FC、SFC 等用户程序使之循环执行。除 OB90 以外，OB1 优先级最低，可以被其他 OB 中断。OB1 默认扫描监控时间为 150ms（可设置），扫描超时，CPU 自动调用 OB80 报错，如果程序中没有建立 OB80，CPU 进入停止模式。

2. 日期中断组织块 OB10 ~ OB17

在 CPU 属性中，可以设置日期中断组织块 OB10 ~ OB17 触发的日期、执行模式（到达设定的触发日期后，OB 只执行一次或按每分、每小时、每周、每月周期执行）等参数，当 CPU 的日期值大于设定的日期值时，触发相应的 OB 并按设定的模式执行。在用户程序中也可以通过调用 SFC28 系统函数设定 CPU 日期中断的参数，调用 SFC30 激活日期中断投入运行，与在 CPU 属性中的设置相比，通过用户程序，可以在 CPU 运行时灵活地修改设定的参数，两种方式可以任意选择，也可以同时对一个 OB 进行设置。

3. 时间延迟中断组织块 OB20 ~ OB23

时间延迟中断组织块 OB20 ~ OB23 的优先级及更新过程映像区的参数需要在 CPU 属性中设置，通过调用系统函数 SFC32 触发执行，OB 号及延迟时间在 SFC32 参数中设定，延迟时间为 1 ~ 60000ms，延迟精度为 1ms，大大优于定时器精度。

4. 循环中断组织块 OB30 ~ OB38

循环中断组织块 OB30 ~ OB38 按设定的时间间隔循环执行，循环中断的间隔时间在 CPU 属性中设定，每一个 OB 默认的时间间隔不同，例如 OB35 默认的时间间隔为 100ms，在 OB35 中的用户程序将每隔 100ms 调用一次，时间间隔可以自由设定，最小时间间隔不能小于 5ms。OB 中的用户程序执行时间必须小于设定的时间间隔，如果间隔时间较短，由于循环中断 OB 没有完成程序扫描而再次被调用，从而造成 CPU 故障，触发 OB80 报错，如果程序中没有创建 OB80，CPU 进入停止模式。通过调用 SFC39 ~ SFC42 系统函数可以禁用、延迟、使能循环中断的调用。循环中断组织块通常处理需要固定扫描周期的用户程序，例如 PID 函数块通常需要在循环中断中调用以处理积分时间的计算。

5. 硬件中断组织块 OB40 ~ OB47

硬件中断也称过程中断，由外部设备产生，例如功能模块 FM、通信处理器 CP 及数字量输入、输出模块等。通常使用具有硬件中断的数字量输入模块触发中断响应，然后为每一个模块配置相应的中断 OB（一个模块只能触发一个中断 OB，S7-300 系列 PLC CPU 只能触发硬件中断 OB40），在模块配置中可以选择输入点的上升沿、下降沿或全部作为触发中断 OB 的事件。配置的中断事件出现，中断主程序，执行中断 OB 中的用户程序一个周期，然后跳回中断处继续运行主程序。使用中断与普通输入信号相比，没有主程序扫描和过程映像区更新时间，适合需要快速响应的应用。

如果输入模块中的一个通道触发硬件中断，操作系统将识别模块的槽号及触发相应的 OB，中断 OB 执行之后发送与通道相关的确认。在识别和确认过程中，该通道再次触发的中断事件将丢失；如果模块其他通道触发中断事件，中断不会丢失，在当前正在运行的中断确认之后触发；如果是不同的模块触发的中断事件，中断请求被记录，中断 OB 在空闲（没有模块其他通道的中断请求）时触发。通过调用 SFC39 ~ SFC42 系统函数可以禁用、延迟、使能硬件中断的调用。

6. DPV1 中断组织块 OB55 ~ OB57

CPU 响应 PROFIBUS-DP V1 从站触发的中断信息。

7. 多处理器中断组织块 OB60

用于 S7-400 系列 PLC 多 CPU（一个机架中最多插入 4 个 CPU 完成同一个复杂任务）处理功能，通过调用 SFC35，可以触发 OB60 在多个 CPU 中同时执行。

8. PROFIBUS 时钟同步中断组织块 OB61 ~ OB64

用于处理 PROFIBUS-DP V1 等时同步，从采集各个从站的输入到逻辑结果输出，需要经过从站输入信号采样循环（信号转换）、从站背板总线循环（转换的信号从模块传递到从站接口）、PROFIBUS-DP 总线循环（信号从站传递到主站）、程序执行循环（信号的程序处理）、PROFIBUS-DP 总线循环（信号从主站传递到从站）、从站背板总线循环（信号从站接口传递到输出模块）及模块输出循环（信号转换）7 个循环，时钟同步中断将 7 个循环同步，优化数据的传递并保证 PROFIBUS-DP 各个从站数据处理的同步性。PROFIBUS 时钟同步中断只能用于 S7-400 系列 PLC CPU（具有 DP V2 功能）。

9. 工艺同步处理中断组织块 OB65

用于 T-CPU（具有运动控制功能的 CPU）工艺块与开始程序的同步处理。

10. 冗余故障中断组织块 OB70、OB72

用于 S7-400H 冗余系统，当 I/O 冗余故障，例如冗余的 PROFIBUS-DP 从站故障时，触发 OB70 的调用，当 CPU 冗余故障，如 CPU 切换、同步故障时，触发 OB72 的调用。如果 I/O冗余，或者 CPU 冗余故障而在 CPU 中没有创建 OB70、OB72，CPU 不会进入停止模式。

11. 异步故障中断组织块 OB80 ~ OB87

异步故障中断用于处理各种故障事件。

OB80：处理时间故障、CIR（Configuration In Run）后的重新运行等功能，例如 OB1 或 OB35 运行超时，CPU 自动调用 OB80 报错，如果程序中没有创建 OB80，CPU 进入停止模式。

OB81：处理与电源相关的各种故障信息（S7-400 系列 PLC CPU 只有电池故障时调用），出现故障，CPU 自动调用 OB81 报错，如果程序中没有创建 OB81，CPU 不会进入停止模式。

OB82：诊断中断，如果使能一个具有诊断中断模块的诊断功能（例如断线、传感器电源丢失），出现故障时调用 OB82，如果程序中没有创建 OB82，CPU 进入停止模式。

诊断中断还对 CPU 所有内、外部故障，包括模块前连接器拔出、硬件中断丢失等作出响应。

OB83：用于模块插拔事件的中断处理，事件出现，CPU 自动调用 OB83 报警，如果程序中没有创建 OB83，CPU 进入停止模式。

OB84：用于处理存储器、冗余系统中两个 CPU 的冗余连接性能降低等事件。

OB85：用于处理操作系统访问模块故障、更新过程映像区时 I/O 访问故障、事件触发但是相应的 OB 没有下载到 CPU 等事件，事件出现，CPU 自动调用 OB85 报错，如果程序中没有创建 OB85，CPU 进入停止模式。

OB86：用于处理扩展机架（不适用 S7-300 系列）、PROFIBUS-DP 主站、PROFIBUS-DP 或 PROFINET I/O 分布 I/O 系统中站点故障等事件，事件出现，CPU 自动调用 OB86 报错，如果程序中没有创建 OB86，CPU 进入停止模式。

OB87：用于处理 MPI GD 通信及时钟同步故障，事件出现，CPU 自动调用 OB87 报错，

如果程序中没有创建 OB87，CPU 不会进入停止模式。

12. 处理中断组织块 OB88

用于处理程序嵌套、区域数据分配故障，故障出现，CPU 自动调用 OB88 报错，如果程序中没有创建 OB88，CPU 进入停止模式。

13. 背景循环中断组织块 OB90

优先级最低，保证 CPU 最短的扫描时间，避免过程映像区更新过于平凡。程序的下载和 CPU 中程序的删除触发 OB90 的调用。只能用于 S7-400 系列 PLC CPU。

14. 启动中断组织块 OB100～OB102

用于处理 CPU 启动事件，暖启动 CPU 调用 OB100，热启动 CPU 调用 OB101（不适合 S7-300 系列 PLC 和 S7-400H），冷启动 CPU 调用 OB102，温度越低，CPU 启动时清除存储器中数据区的类型越多。

15. 同步错误中断组织块 OB121、OB122

OB121 处理与编程故障有关的事件，例如调用的函数没有下传到 CPU 中、BCD 码出错等，OB122 处理与 I/O 地址访问故障有关的事件，例如访问一个 I/O 模块时，出现读故障等。如果上述故障出现，在程序中没有创建 OB121、OB122，CPU 进入停止模式。

**注意：**

不是所有的 OB 都可以在 S7 CPU 中使用，例如 S7-300 系列 PLC CPU 中只有暖启动 OB100，操作系统不能调用 OB101、OB102，CPU 中可以使用的 OB 参考 CPU 的选型手册。

S7-300 系列 PLC 中组织块的优先级是固定的，不能修改，在 S7-400 系列 PLC 中下列组织块的优先级可以进行修改：

OB10～OB47：优先级修改范围 2～23。

OB70～OB72：优先级修改范围 2～38。

OB81～OB87：优先级修改范围 2～26，优先级 24～26 确保异步故障中断不被其他的事件中断。

几个组织块可以具有相同的优先级，当事件同时出现时，组织块按事件出现的顺序（进入系统的顺序）触发，如果超过 12 个相同优先级的 OB 同时触发，中断可能丢失。

### 7.2.2 组织块的区域数据区堆栈

CPU 为每个相同优先级类别的组织块分配了区域数据 L 作为堆栈（L 堆栈），堆栈包括下列数据：

- 程序块中的临时变量；
- 组织块的开始信息（组织块与操作系统的接口区）；
- FC、FB 的参数接口；
- LAD 程序中的中间结果。

如果一个程序块中使用临时变量，被调用时将占用组织块的 L 堆栈，程序块嵌套调用越深占用 L 堆栈空间越大，例如在 OB1 中调用 FC1，在 FC1 中调用 FC10、FC11，在 FC11 中调用 FC12、FC13，占用 L 堆栈大小的计算方式见表 7-3。

S7-300 系列 PLC 每个优先级别的组织块（按优先级别划分，一些具有 S7-400 系列 PLC 功能的 S7-300 系列 PLC CPU 如 CPU318 除外）分配固定大小的 L 堆栈，每款 CPU 的 L 堆栈

表 7-3 组织块的 L 堆栈

| 优 先 级 | L 堆栈中的字节数/B |
|---|---|
| OB1（带有 20B 的开始信息和 10B 区域变量）的调用 | 30 |
| 调用 FC1（带有 30B 的区域变量）<br>30B（OB1）+ 30B（FC1） | 60 |
| 调用 FC10（带有 20B 的区域变量）<br>60B（OB1 + FC1）+ 20B（FC10） | 80 |
| 调用 FC11（带有 20B 的区域变量）<br>60B（OB1 + FC1）+ 20B（ FC11） | 80 |
| 调用 FC12（带有 30B 的区域变量）<br>80B（OB1 + FC1 + FC11）+ 30B（FC12） | 110 |
| 调用 FC13（带有 40B 的区域变量）<br>80B（OB1 + FC1 + FC11）+ 40B（FC13） | 120 |

大小可能不同，具体参数参考 CPU 样本手册。

S7-400 系列 PLC 为每个优先级别组织块（按优先级别划分）分配 L 堆栈的大小可以在 CPU 属性进行配置，可以将没有使用的组织块 L 堆栈分配给其他需要使用的组织块，优化 L 堆栈的分配（S7-400 系列 PLC L 堆栈占用 CPU 的工作存储区）。L 堆栈的大小决定程序中区域变量使用的数量。

**注意：**

如果使用的区域变量超过 L 堆栈规定的限制，CPU 将停机报错。

每个组织块临时变量区前 20B 被称为开始信息，用于操作系统与组织块间的信息传递，开始信息中的变量类型为只读，可以查询组织块每次调用的信息。例如主循环 OB1 的开始信息如图 7-6 所示。

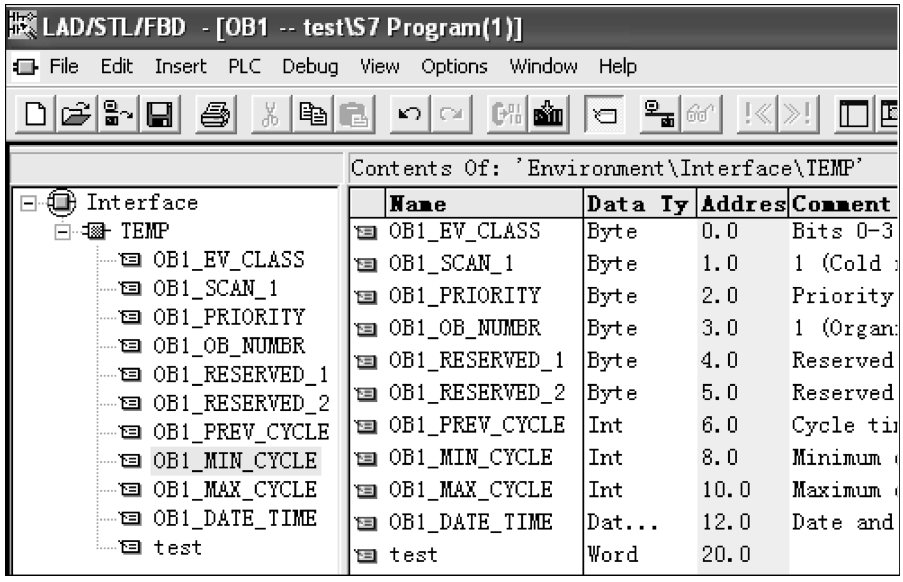



图 7-6 OB1 开始信息

开始信息数据区不能被修改，如果用户需要自定义临时变量，必须在开始信息后创建，如临时变量“test”，开始地址为 L 20.0。在 OB 中定义的临时变量数据类型可以是基本数据类型和复合数据类型，参数类型中只能选择“ANY”指针类型。

当操作系统调用组织块时，将触发的事件类型、OB 的优先级、时间及操作系统状态传送到组织块的开始信息中，组织块执行完成后，存储于开始信息中的数据被释放，可以将开始信息传送到全局变量中以保留上次 OB 调用时的状态信息，例如将 OB1 开始信息中 CPU 上次循环扫描周期、最短扫描周期、最长扫描周期读出，并将结果传送到全局变量中，示例程序如下：

```
L     #OB1_PREV_CYCLE      //上次循环扫描周期（ms），数据类型为整数，
                             使用符号名编程，必须与 OB1 开始信息定义
                             相同，也可以使用绝对地址，如 L LW6
T     MW  20
L     #OB1_MIN_CYCLE       //最短扫描周期（ms），数据类型为整数，可
                             以使用绝对地址，如 L LW8
T     MW  22
L     #OB1_MAX_CYCLE       //最长扫描周期（ms），数据类型为整数，可
                             以使用绝对地址，如 L LW10
T     MW  24
```

这样 CPU 扫描程序的状态信息存储于 MW20、MW22、MW24 中，可以用于监控或逻辑判断，每次 OB1 调用更新变量中存储的状态信息。

大部分组织块是由事件触发，例如 PROFIBUS-DP 主站、从站通信，如果从站故障，CPU 调用 OB86 一个周期，将事件信息写入 OB86 的开始信息数据区中，如果从站故障排除，再次调用 OB86 一个周期并将事件信息写入 OB86 的开始信息数据区中（事件触发 CPU 调用 OB86，如果没有下传，CPU 停机）。通过 OB86 可以将故障的从站站点读出，用于显示、归档及逻辑判断，例如对 3 号从站进行状态判断，示例程序如下：

```
A (
   L        #OB86_EV_CLASS
   L        B#16#39          //判断事件类别，如果等于 B#16#39，表示故障
   ==I                         触发 OB86 的调用
   )
A (
   L        #OB86_FLT_ID     //判断故障类别，如果等于 B#16#C4，表示
   L        B#16#C4            PROFIBUS 站点故障
   ==I
   )
A (
   L        LB  11           //判断触发OB86 调用的源，如果等于3，表示
   L        3                  由 3 号站触发
   ==I
```

```
      )
      S       M   100.0        //M100.0 为 1 表示 3 号站点故障
A (
      L       #OB86_EV_CLASS
      L       B#16#38          //判断事件类别，如果等于 B#16#38，表示从站
      ==I                        故障排除触发 OB86 的调用
      )
A (
      L       #OB86_FLT_ID     //故障类别
      L       B#16#C4
      ==I
      )
A (
      L       LB   11
      L       3                //判断触发 OB86 调用的源
      ==I
      )
      R       M   100.0        // M100.0 为 0 表示 3 号站点故障排除
```

上面的示例程序针对触发 OB86 的事件进行的判断，如果 M100.0 为 1，表示 3 号从站故障，如果 M100.0 为 0，表示 3 号从站无故障或故障消除。STEP7 中已经将不同的事件分类放在不同组织块的开始信息数据区中，所有组织块的开始信息都可以读出，用于程序的判断。

## 7.3 函数

函数（FC）有两个作用：

1）作为子程序使用，将整个程序进行结构化划分，将相互独立的控制设备分成不同的 FC 编写，统一由 OB 调用，便于程序调试及修改，使整个程序条理性强、易读。

2）作为函数使用，对相类似的设备统一编程，函数中通常带有形参，通过多次调用，对形参赋值不同的实参实现对类似设备的控制。在通常情况下 S7-300 系列 PLC 中一个函数最大程序容量为 16KB，S7-400 系列 PLC 中一个函数最大程序容量为 64KB，CPU 中可创建 FC 的数量与 CPU 的类型相关，可参考 CPU 样本手册。

### 7.3.1 函数的接口区

每个函数都带有形参接口区，参数类型分为输入参数、输出参数、输入/输出参数、临时数据区，每种形参类型可以定义多个变量，个数及容量与 OBL 堆栈的大小有关，形参接口用于与调用程序进行数据传递，如图 7-7 所示。

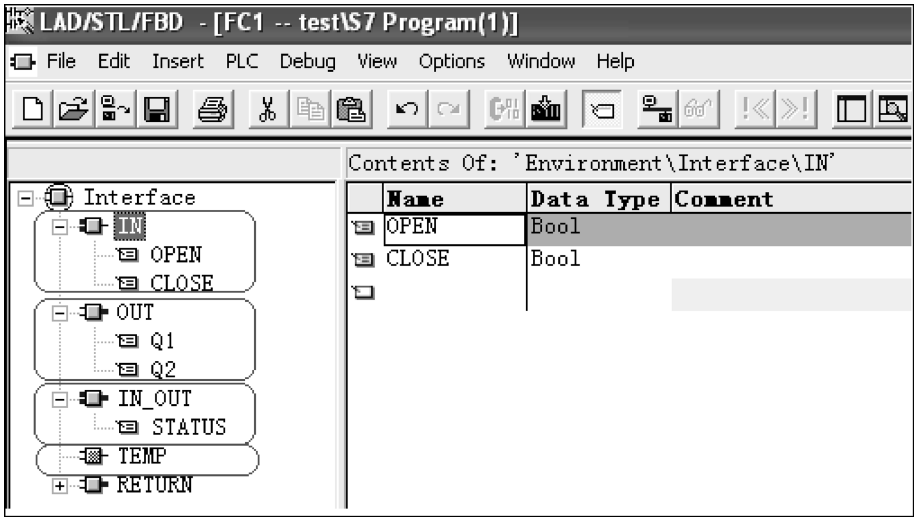



图 7-7 函数形参接口区

IN（输入）：只读参数，函数调用时将用户程序数据传递到函数中。实参可以为常数。

OUT（输出）：只写参数，函数调用时将函数执行结果传递到用户程序中，实参不能为常数。

IN_OUT（输入_输出）：读写参数，接收数据后进行运算，然后将结果返回，例如带灯按钮的控制，输入信号为 1，起动电动机，电动机起动后将参数值置 1，反之置 0，实参不能为常数。

TEMP（临时变量）：只能用于函数内部中间变量（区域数据区 L），不参与数据的传递，临时变量在函数调用时生效，函数执行完成后，临时变量区的数据释放，所以临时变量不能存储中间数据。

函数（FC）接口区允许的数据类型见表 7-4。

**表 7-4 函数（FC）接口区的数据类型**

| 声明的数据类型 | 基本数据类型 | 复合数据类型 | 参数类型 TIME | 参数类型 COUNTER | 参数类型 BLOCK | 参数类型 POINTER | 参数类型 ANY |
|---|---|---|---|---|---|---|---|
| Input | × | ×（2） | × | × | × | × | × |
| Output | × | ×（2） | — | — | — | × | × |
| In/Out | × | ×（2） | — | — | — | × | × |
| Temp | ×（1） | ×（1） | — | — | — | — | ×（1） |

注：×表示可以。

— 表示限制。

（1）位于 FC 的 L 堆栈。

（2）STRINGS 只能定义为默认的长度。

### 7.3.2 无形参函数

在函数的接口数据区中可以不定义形参变量，调用程序与函数之间没有数据交换，只是运行函数中的程序，这样的函数作为子程序调用。子程序将整个控制程序进行结构化划分，清晰明了，便于设备的调试及维护，例如控制 3 个相互独立的设备，将程序分别编写在 3 个子程序中，在主程序中分别调用子程序，实现设备的控制，程序结构如图 7-8 所示。

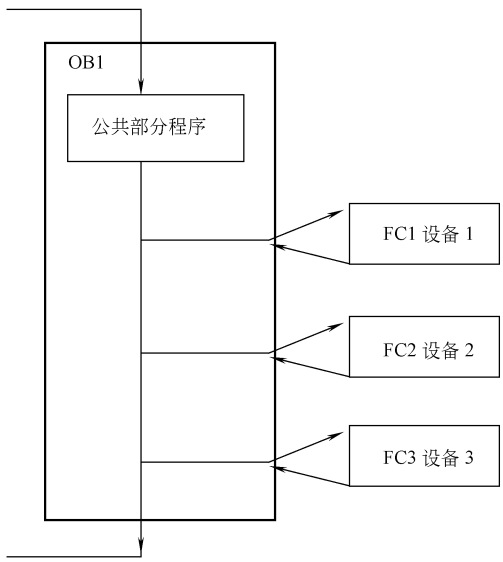



图 7-8 无形参函数（FC）的调用

**注意：**

子程序中也可以带有形参，是否带有形参应根据实际应用而定。在子程序中还可以再次分层调用 FC 作为子程序，子程序的调用是结构化编程重要的组成部分，在实际编程中建议使用图 7-8 中的程序结构，在 OB1 中只进行子程序的调用，控制程序编写在不同的子程序中。

### 7.3.3 带有形参的函数

在应用中常常遇到对许多相似功能的设备进行编程，例如控制 6 组电动机，每个电动机的运行参数相同，如果分别对每一个电动机编程，除输入输出地址不同外，每个电动机控制程序基本相同，重复的工作量比较大。使用函数可以将一个电动机的控制程序作为模板，在程序中多次调用，并赋值参数即可实现对多个设备的控制，例如控制 3 个功能相同的阀门，在函数 FC1 中定义的形参见表 7-5。

**表 7-5 示例阀门形参**

| 地 址 | 声 明 | 阀门形参 | 数据类型 | 初 始 值 |
|---|---|---|---|---|
| 0.0 | IN | OPEN | BOOL | FALSE |
| 0.1 | IN | CLOSE | BOOL | FALSE |
| 2.0 | OUT | OPEN_COM | BOOL | FALSE |
| 2.1 | OUT | CLOSE_COM | BOOL | FALSE |
| 4.0 | IN_OUT | VALVE | BOOL | FALSE |

函数 FC1 程序如下：

```
A (
   O     #OPEN
```

```
O     #VALVE
)
AN    #CLOSE
=     #VALVE

A     #VALVE
=     #OPEN_COM

AN    #VALVE
=     #CLOSE_COM
```

OPEN 与 CLOSE 为开关阀门的命令（通常使用脉冲信号），如果阀门开（VALVE 为1），OPEN_COM 输出为 1，如果阀门关（VALVE 为 0），CLOSE_COM 输出为 1。

在 OB1 中调用函数 FC1 时，形参自动显示，对形参赋值实参，实参通过函数的接口区传递到函数程序中，示例程序如下：

```
CALL   FC    1                    //控制阀门 1
       OPEN        : = M1. 1
       CLOSE       : = M1. 2
       OPEN_COM    : = Q1. 1
       CLOSE_COM   : = Q1. 2
       VALVE       : = M1. 3

CALL   FC    1                    //控制阀门 2
       OPEN        : = M2. 1
       CLOSE       : = M2. 2
       OPEN_COM    : = Q2. 1
       CLOSE_COM   : = Q2. 2
       VALVE       : = M2. 3

CALL   FC    1                    //控制阀门 3
       OPEN        : = M3. 1
       CLOSE       : = M3. 2
       OPEN_COM    : = Q3. 1
       CLOSE_COM   : = Q3. 2
       VALVE       : = M3. 3
```

这样通过函数 FC1 的多次调用，实现 3 个阀门的控制，每个阀门的程序相同而输入、输出接口不同。函数的调用减少重复的工作量，但是不能节省 CPU 的程序容量。

**注意：**

1）在编写函数的输出参数时，应避免没有直接输出，例如在函数中定义输出参数“OUT1”，数据类型为 INT 格式，示例程序如下：

```
      A       #OPEN_COM
      JCN     M1
      L       20
      T       #OUT1
M1:   NOP     0
```

如果 OPEN_COM 为 1，将 20 传递到输出参数 OUT1 中，如果 OPEN_COM 为 0，没有数据传递到输出参数 OUT1 中，OUT1 可能输出一个随机的数值，影响程序的判断，为了避免输出随机值，可以在函数的开始时，将所有输出参数初始化，例如在上述示例函数的开始添加初始化程序，示例如下：

```
      L       0
      T       #OUT1
```

这样在条件 OPEN_COM 为 0 时，#OUT1 输出为 0。如果输出参数是位信号，在函数的开始进行复位操作。

2）函数在逻辑程序中调用后再次修改函数的接口参数（增加或减少形参），在调用程序中，赋值实参的时间早于函数形参再次生成的时间，打开调用程序后，会出现时间标签冲突的提示，调用函数变成红色，如图 7-9 所示。

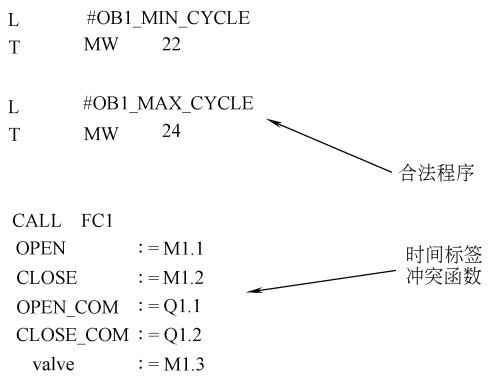



图 7-9 时间标签冲突

带有时间标签冲突的程序下传到 CPU 中将导致停机，所以修改函数的形参后必须检查是否与调用程序时间标签冲突，可以点击程序块“Blocks”，右键选择“Check block Consistency”菜单检测所有程序块，检测完成后标示所有出现问题的程序块，分别打开有问题的程序块，选择时间标签冲突的函数，点击菜单“Edit”→“block call”→“update”更新接口参数。

3）函数的输入为只读参数，输出为只写参数，输入、输出的流向不能反向，否则影响实参的数值。

### 7.3.4 函数嵌套调用时，允许参数传递的数据类型

在主程序中，调用带有形参的函数，可以直接对形参赋值实参，对于函数使用形参的数据类型没有限制（符合表 7-4 要求），在带有形参的函数或函数块中嵌套调用带有形参的函数，可以使用调用函数或函数块中的形参对被调用函数的形参赋值，但是对于调用函数形参的数据类型有限制，下面分别介绍函数间及函数块调用函数允许参数传递的数据类型。

1. 函数间调用参数的传递

函数间嵌套调用时，可以使用调用函数的形参作为实参对被调用函数的形参进行赋值，例如带有形参函数 FC10 调用带有形参的函数 FC12，参数传递如图 7-10 所示。

函数间调用允许参数传递的数据类型见表 7-6。

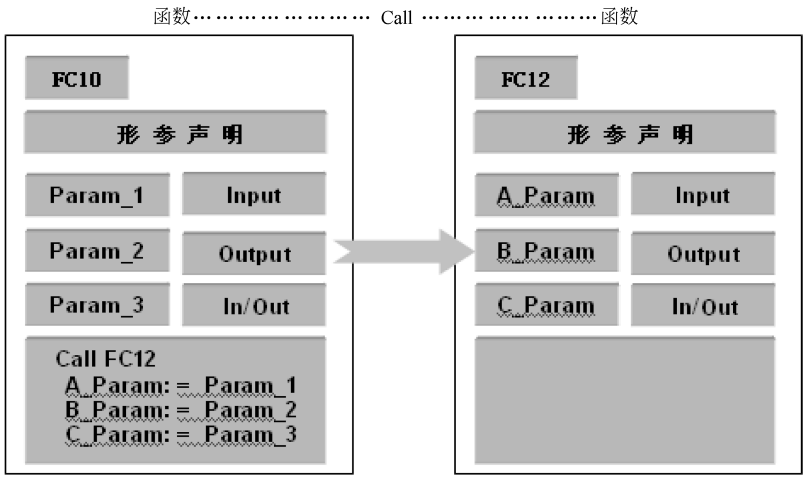



图 7-10 函数调用函数时参数的传递

**表 7-6 函数调用函数时，允许参数传递的数据类型**

| 声明的数据类型 | 基本数据类型 | 复合数据类型 | 参数类型 TIME | 参数类型 COUNTER | 参数类型 BLOCK | 参数类型 POINTER | 参数类型 ANY |
|---|---|---|---|---|---|---|---|
| Input –> Input | × | — | — | — | — | — | — |
| Input –> Output | — | — | — | — | — | — | — |
| Input –> In/Out | — | — | — | — | — | — | — |
| Output –> Input | — | — | — | — | — | — | — |
| Output –> Output | × | — | — | — | — | — | — |
| Output –> In/Out | — | — | — | — | — | — | — |
| In/Out –> Input | × | — | — | — | — | — | — |
| In/Out –> Output | × | — | — | — | — | — | — |
| In/Out –> In/Out | × | — | — | — | — | — | — |

注：× 表示可以。
—表示限制。

2. 函数块调用函数参数的传递

FB 嵌套调用 FC 时，使用 FB 的形参作为实参对 FC 的形参进行赋值，例如带有形参函数块 FB10 调用带有形参的函数 FC12，参数传递如图 7-11 所示。

函数块调用函数时，允许参数传递的数据类型见表 7-7。

这就是为什么有些函数的形参在主程序中调用可以直接赋值实参，而在其他函数中嵌套调用时不能赋值的原因，如果对不符合数据类型要求的形参进行赋值，实参变为红色报警，不能被确认。

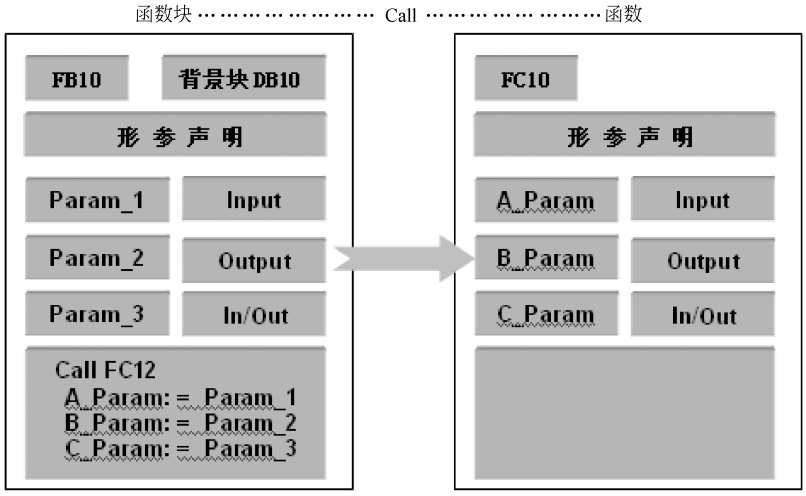



图 7-11 函数块调用函数时参数的传递

**表 7-7 函数块调用函数时，允许参数传递的数据类型**

| 声明的数据类型 | 基本数据类型 | 复合数据类型 | 参数类型 TIME | 参数类型 COUNTER | 参数类型 BLOCK | 参数类型 POINTER | 参数类型 ANY |
|---|---|---|---|---|---|---|---|
| Input –> Input | × | × | — | — | — | — | — |
| Input –> Output | — | — | — | — | — | — | — |
| Input –> In/Out | — | — | — | — | — | — | — |
| Output –> Input | — | — | — | — | — | — | — |
| Output –> Output | × | × | — | — | — | — | — |
| Output –> In/Out | — | — | — | — | — | — | — |
| In/Out –> Input | × | — | — | — | — | — | — |
| In/Out –> Output | × | — | — | — | — | — | — |
| In/Out –> In/Out | × | — | — | — | — | — | — |

注：× 表示可以。
—表示限制。

## 7.4 函数块

FB 与 FC 相比，每次调用函数块都必须为之分配背景数据块，一个数据块可以作为一个函数块的背景数据块，也可以作为多个函数块的背景数据块（多重背景数据块），背景数据块作为函数块的存储器，可以将接口数据区（TEMP 类型除外）以及函数块运算的中间数据存储于背景数据块中，其他逻辑程序可以直接使用背景数据块存储的数据。对于 FC，中

间逻辑结果必须使用函数的输入、输出接口区存储。

函数块很少作为子程序使用，通常将函数块作为具有存储功能的函数使用，每调用一次分配一个背景数据块，将运算结果传递到背景数据块中存储，例如 STEP7 中提供的 PID 函数块，调用时为每个控制回路分配一个背景数据块，在背景数据块中存储控制回路所有的参数。一些特殊编程应用可以在函数块中指定接口数据区存储于多重背景数据块的开始位置，使用更灵活。通常情况下，S7-300 系列 PLC 中一个函数块最大程序容量为 16KB，S7-400 系列 PLC 中一个函数块最大程序容量为 64KB，CPU 中可创建 FB 的数量与 CPU 的类型相关，可参考样本手册。

### 7.4.1 函数块的接口区

与 FC 相同，FB 也带有形参接口区，参数类型除输入参数、输出参数、输入/输出参数、临时数据区外，还带有存储中间变量的静态数据区，参数接口如图7-12所示。

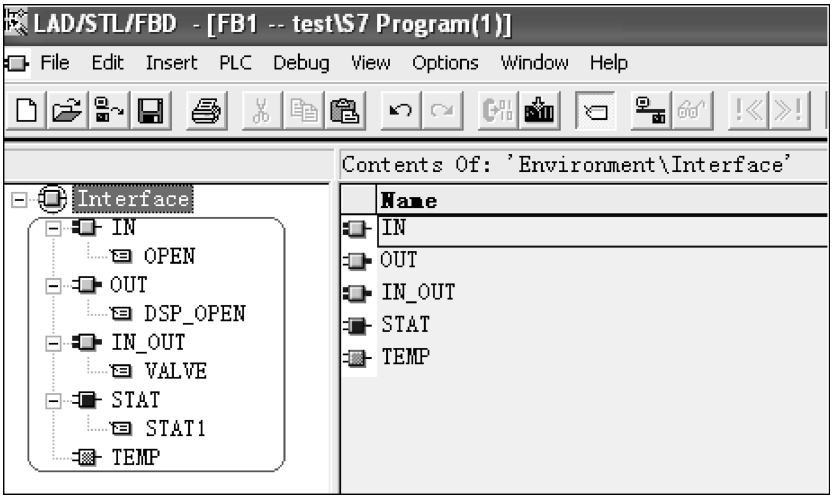



图 7-12 函数块形参接口区

IN（输入）：只读参数，函数块调用时将用户程序数据传递到函数块中。赋值的实参可以为常数。

OUT（输出）：只写参数，函数块调用时将函数块执行结果传递到用户程序中，实参不能为常数。

IN_OUT（输入_输出）：读写参数，接收数据后进行运算，然后将结果返回。

STAT：静态变量，不参与参数传递，用于存储中间过程值。

TEMP（临时变量）：只能用于函数内部中间变量（区域数据区 L），不参与数据的传递，临时变量在函数调用时生效，函数执行完成后临时变量区的数据释放，所以临时变量不能存储中间数据。

FB 接口区允许的数据类型见表 7-8 所示。

表 7-8 FB 接口区的数据类型

| 声明的数据类型 | 基本数据类型 | 复合数据类型 | 参数类型 TIME | 参数类型 COUNTER | 参数类型 BLOCK | 参数类型 POINTER | 参数类型 ANY |
|---|---|---|---|---|---|---|---|
| Input | × | × | × | × | × | × | × |
| Output | × | × | — | — | — | — | — |
| In/Out | × | ×（1）（3） | — | — | — | × | × |
| Static | × | × | — | — | — | — | — |
| Temp | ×（2） | ×（2） | — | — | — | — | ×（2） |

注：×表示可以。
— 表示限制。
（1）在背景数据块中以作为 48 位指针形式存储。
（2）位于 FB 的 L 堆栈。
（3）STRINGS 只能定义为默认的长度。

## 7.4.2 函数块与背景数据块

在逻辑程序中，每次调用 FB 时，都必须分配一个数据块作为 FB 的背景数据块存储数据，背景数据块不能相同，否则控制设备输入、输出信号冲突。FB 与背景数据块的关系如图 7-13 所示。

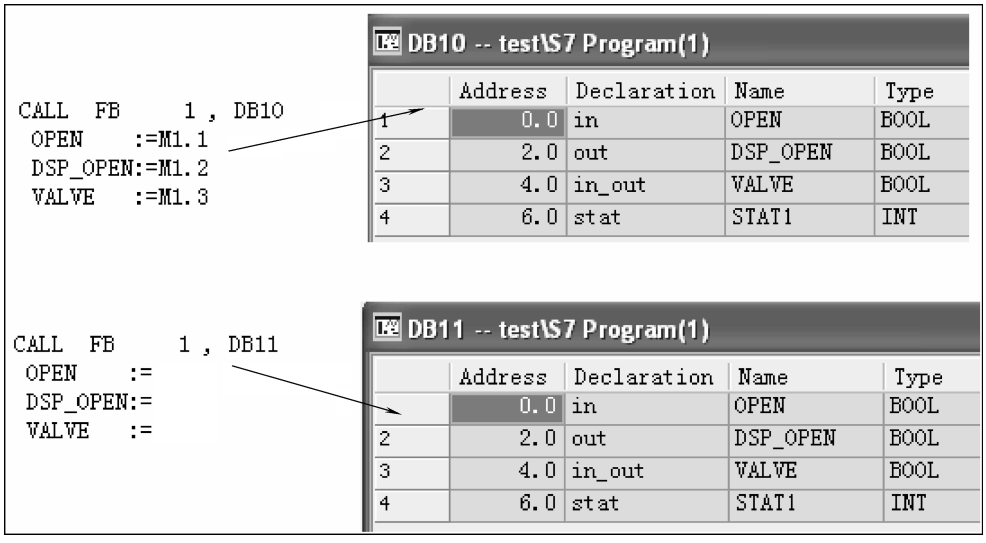



图 7-13 FB 与背景数据块的关系

在图 7-13 中 DB10、DB11 分别存储函数块 FB1 的接口数据区（TEMP 除外），输入数据流向为：赋值的实参→背景数据块→函数块接口输入数据区，输出数据流向为：函数块接口输出数据区→背景数据块→赋值的实参。所以函数块调用时，形参可以不用赋值，通过背景数据块可以直接赋值或读出函数块输出数值。

每次调用 FB 需要分配一个背景数据块，这将影响 DB 的使用资源，如果将多个 FB 作为

一个主 FB 的形参调用，最后主 FB 在 OB 中调用时就会生成一个总的背景数据块，这个背景数据块称为多重背景数据块，多重背景数据块存储所有相关 FB 的接口数据区。每个 FB 在创建时默认设置为具有多重数据块能力，如图 7-14 所示。

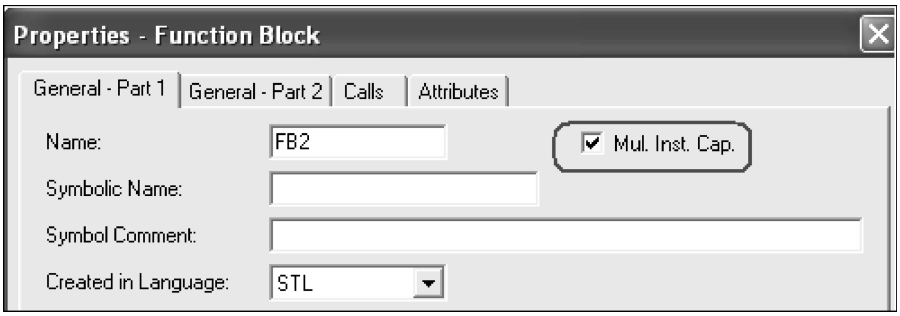



图 7-14 FB 多重背景数据块选项

下面以例子的方式介绍多重数据块的创建，例如在主函数块 FB2 中插入 FB1，作为形参并在 FB2 中调用，如图 7-15 所示。

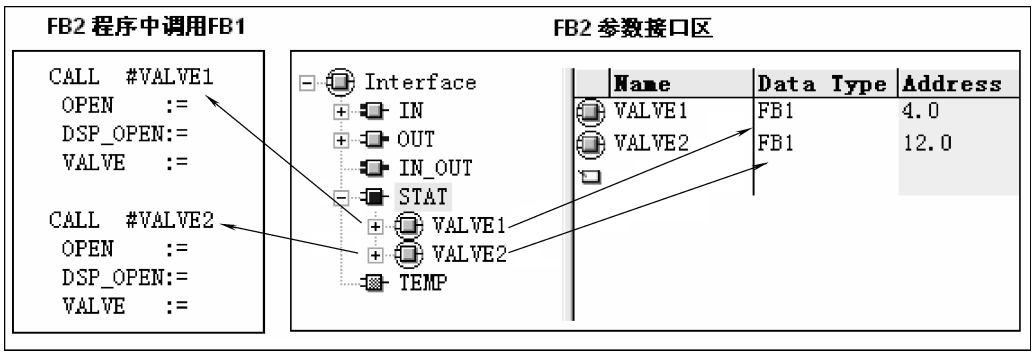



图 7-15 调用作为形参的函数块

在图 7-15 中，函数块 FB1 必须在静态数据区 STAT 中定义并且分配符号名，例如“VALVE1”、“VALVE2”，如果控制多个设备，必须插入多个 FB1，否则控制设备输入、输出接口区数据会发生冲突。在程序中以符号名方式调用作为形参的 FB，调用时不能分配背景数据块，在示例程序中调用 FB1 两次分别控制两个阀门。

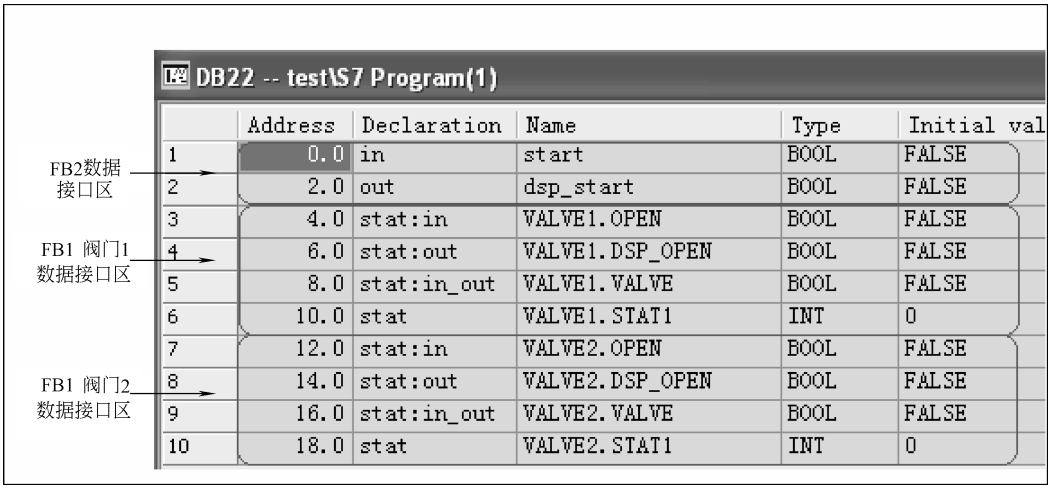
FB2 在 OB 中调用生成多重背景数据块，例如 DB22，DB22 同时作为 FB2 和两个 FB1 的背景数据块，如图 7-16 所示。

**注意：**

1）与 FC 不同，由于 FB 带有存储区——背景数据块，输出参数不会输出随机值，所以不用在 FB 中编写初始化程序。

2）与 FC 相同，避免函数块在逻辑程序中调用后再次修改函数块的接口参数（增加或减少形参）时产生的时间标签冲突。

3）输入、输出的流向不能反向，否则影响实参的数值。




| | Address | Declaration | Name | Type | Initial val |
|---|---|---|---|---|---|
| 1 | 0.0 | in | start | BOOL | FALSE |
| 2 | 2.0 | out | dsp_start | BOOL | FALSE |
| 3 | 4.0 | stat:in | VALVE1.OPEN | BOOL | FALSE |
| 4 | 6.0 | stat:out | VALVE1.DSP_OPEN | BOOL | FALSE |
| 5 | 8.0 | stat:in_out | VALVE1.VALVE | BOOL | FALSE |
| 6 | 10.0 | stat | VALVE1.STAT1 | INT | 0 |
| 7 | 12.0 | stat:in | VALVE2.OPEN | BOOL | FALSE |
| 8 | 14.0 | stat:out | VALVE2.DSP_OPEN | BOOL | FALSE |
| 9 | 16.0 | stat:in_out | VALVE2.VALVE | BOOL | FALSE |
| 10 | 18.0 | stat | VALVE2.STAT1 | INT | 0 |

图 7-16 生成多重背景数据块

### 7.4.3 函数块嵌套调用时，允许参数传递的数据类型

与函数相同，如果主程序调用函数块，对函数块的形参可以直接赋值实参，函数块使用形参的数据类型没有限制（符合表 7-8 要求），但是带有形参的函数或函数块中嵌套调用带有形参的函数块时，使用调用函数或函数块的形参作为实参赋值被调用函数块的形参是有限制的。

1. 函数调用函数块参数的传递

函数嵌套调用函数块时，可以使用调用函数的形参作为实参赋值被调用函数块的形参，例如带有形参函数 FC10 调用带有形参的函数块 FB12，参数传递如图 7-17 所示。

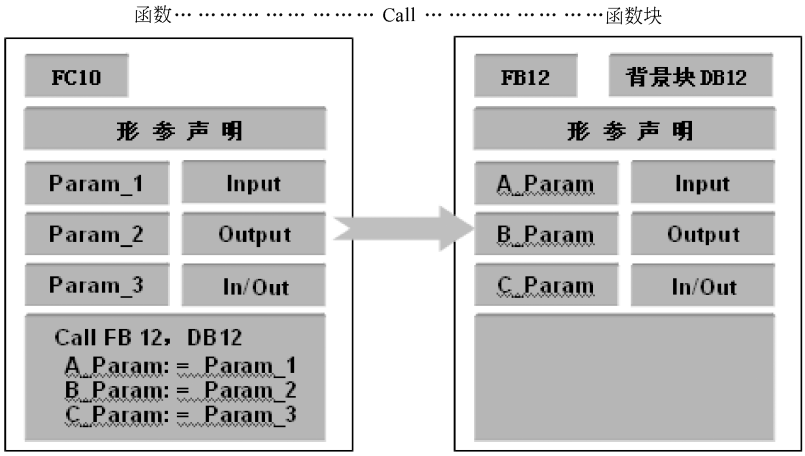



图 7-17 函数调用函数块参数的传递

函数调用函数块时，允许参数传递的数据类型见表 7-9。

表 7-9 函数调用函数块时，允许参数传递的数据类型

| 声明的数据类型 | 基本数据类型 | 复合数据类型 | 参数类型 TIME | 参数类型 COUNTER | 参数类型 BLOCK | 参数类型 POINTER | 参数类型 ANY |
|---|---|---|---|---|---|---|---|
| Input –> Input | × | — | × | × | × | — | — |
| Input –> Output | — | — | — | — | — | — | — |
| Input –> In/Out | — | — | — | — | — | — | — |
| Output –> Input | — | — | — | — | — | — | — |
| Output –> Output | × | — | — | — | — | — | — |
| Output –> In/Out | — | — | — | — | — | — | — |
| In/Out –> Input | × | — | — | — | — | — | — |
| In/Out –> Output | × | — | — | — | — | — | — |
| In/Out –> In/Out | × | — | — | — | — | — | — |

注：× 表示可以。
—表示限制。

2. 函数块间调用参数的传递

函数块嵌套调用函数块时，可以使用调用函数块的形参作为实参对被调用函数块的形参进行赋值，例如带有形参函数块 FB10 调用带有形参的函数块 FB12，参数传递如图 7-18 所示。

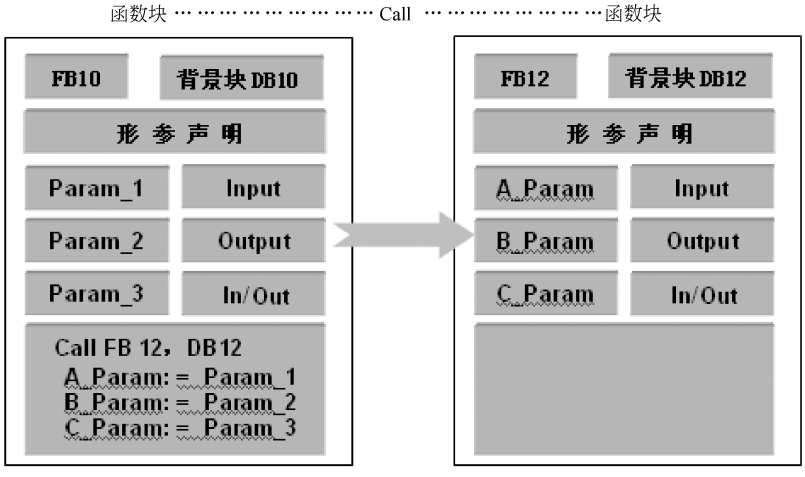



图 7-18 函数块间调用的参数传递

函数块嵌套调用时，允许参数传递的数据类型见表 7-10 所示。

表 7-10 函数块间调用时，允许参数传递的数据类型

| 声明的数据类型 | 基本数据类型 | 复合数据类型 | 参数类型 TIME | 参数类型 COUNTER | 参数类型 BLOCK | 参数类型 POINTER | 参数类型 ANY |
|---|---|---|---|---|---|---|---|
| Input –> Input | × | × | × | × | × | — | — |
| Input –> Output | — | — | — | — | — | — | — |
| Input –> In/Out | — | — | — | — | — | — | — |
| Output –> Input | — | — | — | — | — | — | — |
| Output –> Output | × | × | — | — | — | — | — |
| Output –> In/Out | — | — | — | — | — | — | — |
| In/Out –> Input | × | — | — | — | — | — | — |
| In/Out –> Output | × | — | — | — | — | — | — |
| In/Out –> In/Out | × | — | — | — | — | — | — |

注：×表示可以。
—表示限制。

**小窍门：**

从表 7-9、表 7-10 中可以得出，参数的传递方向不能从“Output”到“Input”，如果编程需要，可以先将类型为“Output”的数据传递到类型为“Static”的数据中，然后再将类型为“Static”的数据传递到类型为“Input”的数据中，参数传递的方向为“Output”→“Static”→“Input”。

## 7.5 数据块

数据块（DB）用于存储用户数据及程序的中间变量，默认状态下，数据块中存储的数值掉电保持，数据块占用 CPU 的装载存储区和工作存储区，与标志存储区（M）相比，使用功能相同，都是全局变量，但是 M 数据区的大小在 CPU 技术规范中已经定义，不可扩展，而数据块存储区由用户定义，最大不能超过工作存储区或装载存储区（只存储于装载存储区）。通常情况下 S7-300 系列 PLC 中一个数据块最大数据空间为 32KB，S7-400 系列 PLC 中一个数据块最大数据空间为 64KB，CPU 中可创建数据块的数量与 CPU 的类型相关，可参考样本手册。按功能划分，DB 可以作为共享数据块（Share DB）、背景数据块（Instance DB）和基于 UDT（用户定义数据类型）的数据块（DB of Type）。下面分别介绍三种类型数据块。

### 7.5.1 共享数据块

共享数据块（Share DB）可以作为所有程序使用的全局变量，在 CPU 允许的条件下，一个程序中可以自由创建多个数据块，每个数据块最高可以存储 64KB 数据。共享数据块必须事先定义才可以在程序中使用，在 SIMATIC Manager 界面下点击“S7 Program”→“Blocks”插入一个数据块，选择数据块为“Share DB”（默认设置），如图 7-19 所示。

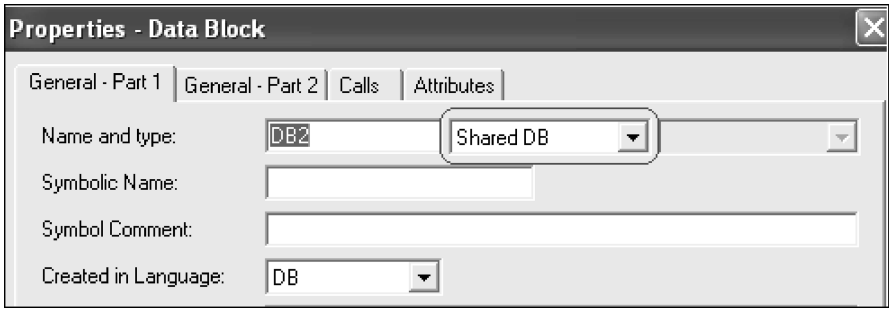



图 7-19 创建共享 DB 块

打开数据块在“STRUCT”与“END_STRUCT”中间定义数据块的空间（通过数组变量可以简单定义 DB 的空间），例如定义数据块的空间为 4KB，如图 7-20 所示。

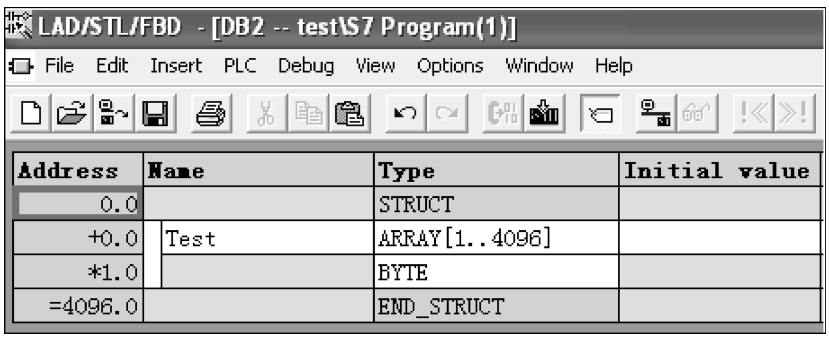



图 7-20 定义共享 DB 块空间

图 7-20 中使用数组定义 DB 的空间，数组的元素为字节，可以以任何数据类型使用，例如 DB2. DBW2 可以在整数相加指令中作为一个整数使用（使用 M 数据区，如果在符号表中定义数据类型为字，则在整数相加指令中不能作为一个整数使用）。用户也可以根据应用分别插入不同数据类型的变量。

**注意：**

如果访问 DB 的地址超出 DB 定义的空间，CPU 调用 OB121（编程错误），如果没有下传 OB121，CPU 停机。

默认条件下，共享数据块存储的数据掉电保持，下载到 CPU 中既占用 CPU 的装载存储区又占用工作存储区（便于数据的快速访问），通过 DB 的属性可以修改这些特性，点击某一个 DB，右键选择“Open Properties”，打开 DB 的属性界面如图 7-21 所示。

如果选择“Non Retain”选项，这个 DB 存储的数据掉电后丢失，如果选择“Unlinked”选项，DB 下载后只存储于 CPU 的装载存储区，如果程序需要访问 DB 的数据，通过调用 SFC20（S7-400 系列 PLC）或 SFC83、SFC84（以 MMC 作为装载存储区的 CPU）将装载存储区的数据复制到工作存储区中。打开 DB，点击“File”→“store read only”可以将 DB 作为只读属性存储，“block Read-only”属性在 DB 的属性界面中只能显示而不能进行设置。

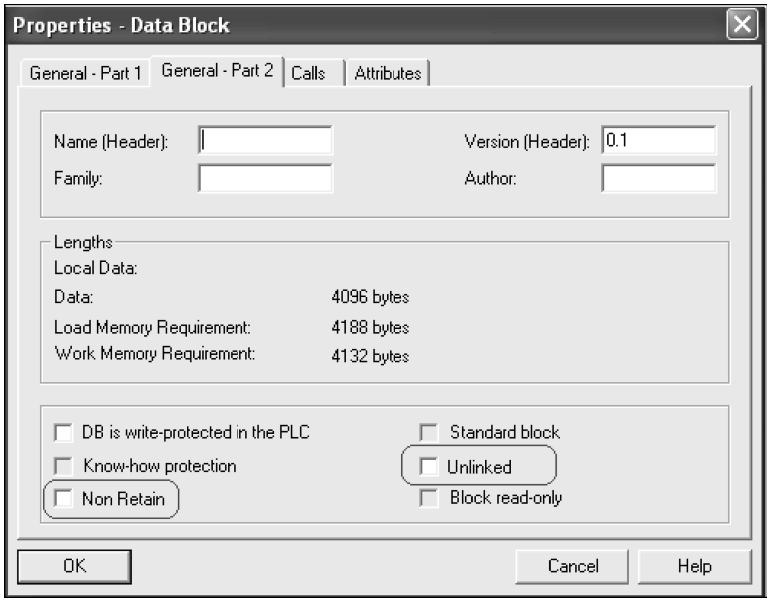



图 7-21 DB 属性界面

## 7.5.2 背景数据块

背景数据块（Instance DB）与 FB、SFB 相关联，在创建背景数据块时，必须指定它所属的 FB，而且该 FB 必须已经存在，如图 7-22 所示。

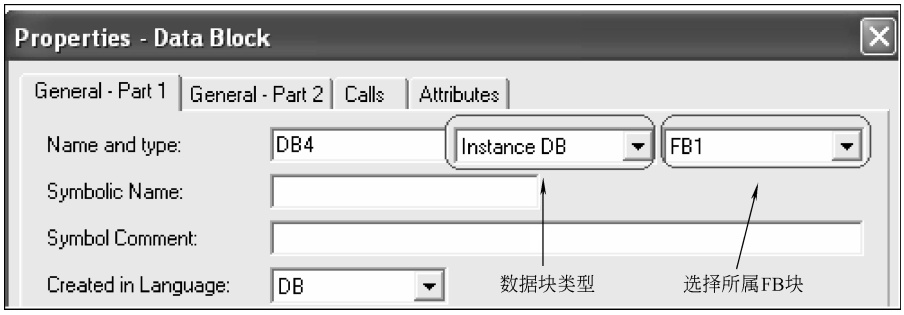



图 7-22 创建背景数据块

在调用一个 FB 时，分配一个已经创建的背景数据块，也可以直接定义一个没有使用的 DB，自动生成背景数据块。背景数据块与共享 DB 相比，背景数据块只存储与 FB 接口数据区（TEMP 变量除外）相关的数据，数据块格式随接口数据区的变化而变化，不能插入用户自定义的数据区，如图 7-23 所示。

在属性设置界面中，不能选择“Unlinked”选项，否则背景数据块只存储于装载存储区中，CPU 由于未发现 FB 的背景数据块而出错。通常背景数据块的属性保持为默认设置。

DB3 -- test\S7 Program(1)

| | Address | Declaration | Name | Type | Initial value | Actual value |
|---|---|---|---|---|---|---|
| 1 | 0.0 | in | OPEN | BOOL | FALSE | FALSE |
| 2 | 2.0 | out | DSP_OPEN | BOOL | FALSE | FALSE |
| 3 | 4.0 | in_out | VALVE | BOOL | FALSE | FALSE |
| 4 | 6.0 | stat | STAT1 | INT | 0 | 0 |

图 7-23 格式固定的背景数据块

背景数据块与共享数据块都是全局变量，访问方式相同，STL 编程指令“OPN DB”与“OPN DI”可以在程序中同时打开两个 DB，与 DB 的类型无关。

### 7.5.3 基于 UDT 的数据块

在数据类型中已经介绍了 UDT 是一个用户自定义的数据类型模板，可以由不同的数据类型组成，作为一个整体的变量多次使用，UDT 另外一个功能是提供一个固定格式的数据结构，便于用户使用，例如在使用功能模块 FM 时，FM 模块带有独立处理功能，CPU 通过调用特殊函数与 FM 模块进行数据通信，通信格式固定，数据量大，如与 FM354 模块的通信数据区超过 500B，用户不方便创建，功能模块的软件包提供一个 UDT 数据格式，包括定义模块地址、通道以及所有 CPU 与功能模块通信相关的数据区，通过 UDT，用户可以方便地将 UDT 数据格式存储于一个 DB 中。

创建基于 UDT 的数据块时，必须指定它所属的 UDT，如图 7-24 所示。

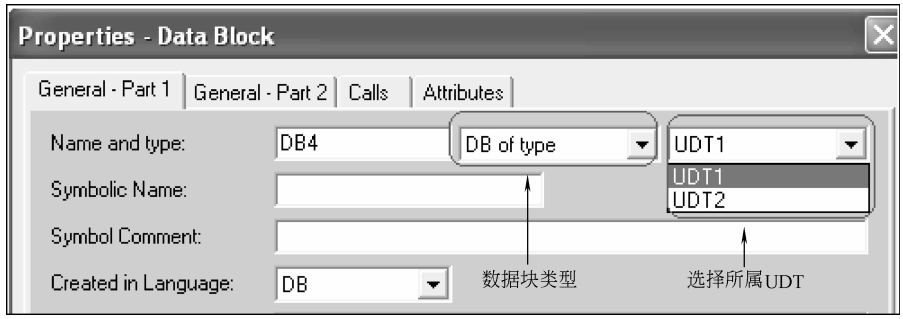
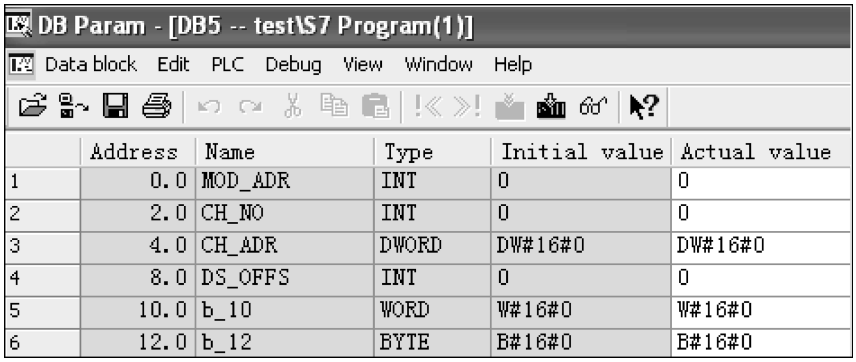



图 7-24 创建基于 UDT 的 DB

如果 UDT 的格式发生变化，点击“Blocks”右键选择“Check block Consistency”更新 DB 的内容或删除 DB 重新生成。与背景数据块相同，基于 UDT 的 DB 只存储与 UDT 相关的数据，不能插入用户自定义的数据区，如图 7-25 所示。

基于 UDT 的 DB 属性可以修改，建议保持默认设置。基于 UDT 的 DB 为全局变量，与共享数据块、背景数据块的访问方式相同。




| | Address | Name | Type | Initial value | Actual value |
|---|---|---|---|---|---|
| 1 | 0.0 | MOD_ADR | INT | 0 | 0 |
| 2 | 2.0 | CH_NO | INT | 0 | 0 |
| 3 | 4.0 | CH_ADR | DWORD | DW#16#0 | DW#16#0 |
| 4 | 8.0 | DS_OFFS | INT | 0 | 0 |
| 5 | 10.0 | b_10 | WORD | W#16#0 | W#16#0 |
| 6 | 12.0 | b_12 | BYTE | B#16#0 | B#16#0 |

图 7-25 基于 UDT 的 DB

## 7.6 系统函数与系统函数块

系统函数（SFC）与系统函数块（SFB）固化在 S7 的 CPU 中，相当于系统提供可供用户程序调用的 FC 和 FB，实现与 CPU 系统相关的一些功能，如读写 CPU 的时钟、CPU 状态指示灯信号等功能。SFC、SFB 与 FC、FB 调用方式相同，调用 SFB 时也需要背景数据块存储相关信息，由于固化在 CPU 中，SFC、SFB 的内容和序号不能修改。

与 OB 一样，可供调用的 SFC、SFB 与 CPU 的型号相关，通过 CPU 的技术手册可以找到该 CPU 支持的 SFC、SFB 的详细信息或在线直接查看 CPU 中包含 SFC、SFB 的内容。如果用户程序中调用的 SFC、SFB 在 CPU 中没有固化，CPU 因不能识别而报错。SFC、SFB 的序号、功能分类见表 7-11。

**表 7-11 系统函数（SFC）与系统函数块（SFB）**

| 系统功能分类 | 相关 SFC、SFB | 简要功能描述 |
|---|---|---|
| 复制与块功能 | SFC20 “BLKMOV” | 复制一段存储区（源存储区）的数据到另一段存储区（目标存储区）中，可复制的存储区包括 M、DB、I、Q。与 MOVE 指令相比，SFC20 传送的数据量更大。可以将装载存储区 UNLINKED 的 DB 复制到工作存储区中，SFC20 执行过程可以被中断 |
| | SFC81 “UBLKMOV” | 复制最大长度为 512B 的存储区（源存储区）的数据到另一段存储区（目标存储区）中，复制过程不能被中断，增加中断执行的响应时间 |
| | SFC21 “FILL” | 将一段存储区（源存储区）的数据填充到另一段存储区（目标存储区）中，如果目标存储区大于源存储区，按源存储区存储数据的次序，一直将目标存储区填满为止 |
| | SFC22 “CREAT_DB” | 通过程序调用，生成 DB |
| | SFC23 “DEL_DB” | 通过程序调用，删除 DB |
| | SFC24 “TEST_DB” | 通过程序调用，测试一个 DB 是否存在、DB 的长度、是否为只读 |
| | SFC25 “COMPRESS” | 由于 DB 的多次删除和再次下载，CPU 的工作存储区和装载存储区程序出现间隔，影响存储区的存储效果，通过程序调用消除存储区的间隔。在外部通过切换 CPU 的选择开关，从 “STOP” 到 “RUN-P” 位置具有相同的作用 |

（续）

| 系统功能分类 | 相关 SFC、SFB | 简要功能描述 |
| --- | --- | --- |
| 复制与块功能 | SFC44 “REPL_VAL” | SFC44 可以将替代值传送到被故障中断 OB 的累加器 1 中，使程序使用替代值连续运行，SFC44 只能在 OB121、OB122 中调用 |
| | SFC82 “CREA_DBL” | 通过程序调用在装载存储区（MMC，只适合 S7-300 系列 PLC CPU）中生成 DB，DB 的属性可以定义 |
| | SFC83 “READ_DBL” | 将存储于装载存储区（MMC，只适合 S7-300 系列 PLC CPU）中 DB 中的值读出并复制到工作存储区中 |
| | SFC84 “WRIT_DBL” | 将工作存储区中的数据存储于装载存储区（MMC，只适合 S7-300 系列 PLC CPU）DB 中。与 SFC83 方向相反 |
| | SFC85 “CREA_DB” | 通过程序调用生成 DB，与 SFC22 相比可以定义 DB 的属性 “RETAIN” |
| 控制程序执行功能 | SFC43 “RE_TRIGR” | 重新触发 “watchdog” 循环监控时间 |
| | SFC46 “STP” | 执行 SFC46，CPU 切换到 “STOP” 模式，通过手动重新启动 CPU 运行 |
| | SFC47 “WAIT” | 执行 SFC47，程序执行最长可以延迟 32767μs |
| | SFC35 “MP_ALM” | 触发多个 CPU 中断，在多个 CPU（一个站中最多 4 个 S7-400系列 PLC CPU 同时运行完成同一个任务，以增加实时性）运行时，触发所有的 CPU，同时执行 OB60 中断 |
| | SFC104 “CiR” | 控制 CIR（Configuration In Run）模式运行 |
| 系统时钟功能 | SFC0 “SET_CLK” | 设置系统时钟 |
| | SFC1 “READ_CLK” | 读出系统时钟 |
| | SFC48 “SNC_RTCB” | 通过 MPI、S7 背板总线、S7-400 系列 PLC K 总线同步所有具有时钟功能的模块，调用 SFC48 进行时钟同步独立于设定的同步间隔 |
| | SFC100 “SET_CLKS” | 设置 CPU 的时钟和时钟状态，如校正时间、夏时制、冬时制等 |
| 处理 CPU 运行时间定时表功能 | CPU 内部集成多个运行时间定时表（有的 CPU 集成 16 位定时表，有的 CPU 集成 32 位定时表，单位为小时），可以测量 CPU、控制设备及连接元件的运行时间 | |
| | SFC101 “RTM” | SFC101 启动、停止、设置、读出 CPU 内部其中一个 32 位定时表功能 |
| | SFC2 “SET_RTM” | 设置 CPU 内部一个 16 位定时表预置值，也可以设置 32 位定时表，但是与 16 位定时表操作相同（定时范围 0～32767h） |
| | SFC3 “CTRL_RTM” | 启动、停止 CPU 内部一个 16 位定时表 |
| | SFC4 “READ_RTM” | 读出 CPU 内部一个 16 位定时表定时值 |
| | SFC64 “TIME_TCK” | 读出 CPU 系统时间，时间范围 0～2147483647ms，循环定时，S7-400 系列 PLC CPU 及 CPU 318 定时精度为 1ms，S7-300系列 PLC CPU 定时精度为 10ms。如果使用定时器数量超过 CPU 规定的限制，调用 SFC64，通过编程作为定时器功能使用 |

（续）

| 系统功能分类 | 相关 SFC、SFB | 简要功能描述 |
| --- | --- | --- |
| 传送数据记录功能 | 一些智能模块带有存储区，存储模块配置、诊断等信息，模块的存储区最多可以划分 0 ~ 240 个小的分区称为数据记录区（不是每个模块都有 241 个数据记录区），每个记录区最大空间为 240B，数据记录区属性分为只读和只写，CPU 通过调用 SFC 或 SFB 读写这些数据记录区（数据记录区的序号与数据格式参考相应模块技术规范手册，有些模块手册中同时说明适合使用那些 SFC、SFB 进行读写操作），对数据记录区的读写属于异步操作，同时调用 SFC 或 SFB 的个数受 CPU 的限制（参考 CPU 订货样本） | | |
| | SFC54“RD_DPARM” | 读出某个模块定义的参数（如模拟量模块设定值等）并复制到指定的数据区（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFC102“RD_DPARA” | 读出某个模块预先定义的参数（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFC55“WR_PARM” | 通过程序修改模块的动态参数，例如 S7-400 系列 PLC 模拟量模块温度补偿值等（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFC56“WR_DPARM” | 通过程序修改模块的默认参数（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFC57“PARM_MOD” | 通过程序分配模块参数（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFC58“WR_REC” | 写数据记录区（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFC59“RD_REC” | 读数据记录区。读数据记录区在程序中使用频繁，例如读写 ASI 模块连接 AB 型传感器信号、SIMOCODE 信息等（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFB81“RD_DPAR” | 读设备的参数（数据记录区的序号与数据格式参考相应模块技术规范手册），与 SFC54 相比可以读取分布式 I/O（PROFIBUS-DP 或 PROFINET IO）中的设备 |
| 符合PNO AK 1131规约用于DP V1功能的SFB | SFB52“RDREC” | 读取分布式 I/O（PROFIBUS-DP 或 PROFINET IO）中模块或子模块数据记录区的数据（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFB53“WRREC” | 将数据传送到分布式 I/O（PROFIBUS-DP 或 PROFINET IO）模块或子模块数据记录区中（数据记录区的序号与数据格式参考相应模块技术规范手册） |
| | SFB54“RALRM” | 接收中央扩展或分布式 I/O（PROFIBUS-DP 或 PROFINET IO）模块或子模块产生的中断信息 |
| | SFB75 SALRM | PROFIBUS-DP 智能从站调用 SFB75 产生中断，额外的中断信息主站需要调用 SFB54 接收。SFB 75 只能在“S7-compatible”模式下使用 |

（续）

| 系统功能分类 | 相关 SFC、SFB | 简要功能描述 |
| --- | --- | --- |
| 处理日期中断功能 | 日期中断 OB10 ~ OB17 触发日期及模式可以在 CPU 硬件属性中配置，也可以通过调用 SFC 实现，两者取其一 | |
| | SFC28“SET_TINT” | 设置日期中断序号、触发模式及开始日期 |
| | SFC29“CAN_TINT” | 取消某一序号的日期中断 OB |
| | SFC30“ACT_TINT” | 激活某一序号的日期中断 OB |
| | SFC31“QRY_TINT” | 查询某一序号日期中断 OB 的状态 |
| 处理时间延迟中断功能 | 通过调用系统函数触发时间延迟中断 OB20 ~ OB23 的执行。延时精度可以达到 1ms。可以使用的时间延迟中断 OB 参考相关 CPU 的技术手册 | |
| | SFC32“SRT_DINT” | 触发某一序号的时间延迟中断 OB 及延迟时间（1 ~ 60000ms） |
| | SFC33“CAN_DINT” | 取消某一序号的时间延迟中断 OB |
| | SFC34“QRY_DINT” | 查询某一序号时间延迟中断 OB 的状态 |
| 处理同步故障功能 | 同步故障为编程和地址访问故障，例如访问不正确的地址区、地址等地址访问故障及 BCD 转换故障、DB 没有下载等编程故障等。通过调用系统函数可以掩饰同步故障，不触发故障中断 OB 及导致选择性的反应 | |
| | SFC36“MSK_FLT” | 掩饰不同类型的同步故障，如果故障出现，不会触发中断 OB，故障信息进入 CPU 故障寄存器中 |
| | SFC37“DMSK_FLT” | 去除掩饰的同步故障，如果故障出现，触发中断 OB |
| | SFC38“READ_ERR” | 读出进入 CPU 故障寄存器中的故障信息 |
| 处理中断及异步故障功能 | 处理中断 OB10 ~ OB72，异步故障 OB80 ~ OB87 | |
| | SFC39“DIS_IRT” | 去除中断功能 |
| | SFC40“EN_IRT” | 使能中断功能 |
| | SFC41“DIS_AIRT” | 延时当前 OB 被高优先级 OB 中断，调用 SFC42 或当前 OB 执行完成后执行高优先级 OB 中断 |
| | SFC42“EN_AIRT” | 使能 SFC41 延时的 OB |
| 诊断功能 | SFC6“RD_SINFO” | 读出最近运行 OB 开始信息 |
| | SFC51“RDSYSST” | 读出系统状态信息 |
| | SFC52“WR_USMSG” | 将用户定义的诊断信息写入 CPU 的诊断缓冲区中 |
| | SFC78“OB_RT” | 确定单独 OB 的运行时间 |
| | SFC87“C_DIAG” | 诊断当前 S7 通信连接的状态，如诊断冗余系统当前使用的通信连接等 |
| | SFC103“DP_TOPOL” | 与带有诊断功能的中继器配合使用，初始化 PROFIBUS-DP网络拓扑结构 |
| 更新过程映像区功能 | SFC26“UPDAT_PI” | 更新 OB1 输入过程映像区。在 S7-400 系列 PLC 中，有些 OB 没有选择更新 OB1 过程映像区功能，通过调用 SFC26 更新 |
| | SFC27“UPDAT_PO” | 更新 OB1 过程映像区输出 |
| | SFC126“SYNC_PI” | 更新输入过程映像分区。在 S7-400 系列 PLC 中，可以将过程映像区划分多个小的分区（分区个数与 CPU 类型相关）。在每个 OB 中可以选择更新整个过程映像区或映像区其中一个分区 |

（续）

| 系统功能分类 | 相关 SFC、SFB | 简要功能描述 |
| --- | --- | --- |
| 更新过程映像区功能 | SFC127“SYNC_PO” | 更新输出过程映像分区 |
| | SFC79“SET” | 将设定范围的输出点置位为 1，与输出地址是否在过程映像区无关 |
| | SFC80“RSET” | 将设定范围的输出点复位为 0，与输出地址是否在过程映像区无关 |
| | SFB32“DRUM” | 按次序执行，最高 16 步，步与步间的切换可以使用事件方式触发（上升沿信号），也可以以时间方式触发 |
| 模块寻址功能 | SFC5“GARD_LGC” | 基于模块通道，由模块物理地址查询逻辑地址 |
| | SFC49“LGC_GADR” | 基于模块通道，由模块逻辑地址查询物理地址 |
| | SFC50“RD_LGADR” | 读出模块所有通道地址 |
| | SFC70“GEO_LOG” | 由中央机架、PROFIBUS-DP 及 PROFINET IO 站点上的模块槽号查询开始地址 |
| | SFC71“LOG_GEO” | 由模块开始地址查询安装的位置（中央机架、PROFIBUS-DP 及 PROFINET IO 站点上）及槽号 |
| 处理分布式 I/O 及 PROFINET IO 功能 | SFC7“DP_PRAL” | 在 PROFIBUS-DP 智能从站中，触发硬件中断，主站调用 OB40 响应 |
| | SFC11“DPSYC_FR” | 将属于一个 PROFIBUS-DP 组的从站进行同步或冻结操作，保证所有从站发送到主站数据及主站发送到所有从站数据的同时性 |
| | SFC12“D_ACT_DP” | 可以将 PROFIBUS-DP 或 PROFINET IO 某个从站激活或禁用，将某个故障从站或在调试阶段没有安装的从站禁用，减少从站轮询时间，CPU 上没有从站丢失故障显示 |
| | SFC13“DPNRM_DG” | 读取某个从站的诊断信息 |
| | SFC14“DPRD_DAT” | 在主站中解包从站连续的数据（数组类型，例如通信数据的连续性为“TOTAL LENGTH”）。SFC14 适合于 PROFIBUS-DP 及 PROFINET IO。有的 CPU 集成有数据打包、解包功能，不需要调用 SFC14、SFC15 |
| | SFC15“DPWR_DAT” | 将数据打包发送到从站上 |
| 处理 PROFINET 功能 | SFC112“ PN_IN” | 更新 CBA 组建输入接口 |
| | SFC113“PN_OUT” | 更新 CBA 组建输出接口 |
| | SFC114“PN_DP” | 更新 PROFINET 与 PROFIBUS 网关通信接口 |
| 处理 MPI 网络 GD 通信功能 | SFC60“GD_SND” | 适用 S7-400 系列 PLC 事件触发发送 GD 数据包 |
| | SFC61“GD_RCV” | 与 SFC60 相对应接收 GD 数据包 |
| 处理基于 MPI 网络 S7 Basic 通信（SFC72/73/74 可用于 PROFIBUS） | SFC65“ X_SEND” | 发送数据包（通信双方必须编程进行数据交换），与 SFC66“X_RCV”相对应 |
| | SFC66“X_RCV” | 与 SFC65“ X_SEND”相对应 |
| | SFC67“X_GET” | 将通信方的数据存储于本地数据区，通过单方编写可以进行数据交换 |

（续）

| 系统功能分类 | 相关 SFC、SFB | 简要功能描述 |
| --- | --- | --- |
| 处理基于 MPI 网络 S7 Basic 通信（SFC72/73/74 可用于 PROFIBUS） | SFC68“X_PUT” | 将本地数据区数据存储于通信方的数据区中，通过单方编写可以进行数据交换 |
| | SFC69“X_ABORT” | 断开一个通信连接 |
| | SFC72“I_GET” | 与 SFC67/68/69 相比，功能相同，不同之处在于 SFC67/68/69处理站点间数据交换，SFC72/73/74 处理站点内数据交换，例如一个站内 PROFIBUS-DP 主从通信 |
| | SFC73“I_PUT” | |
| | SFC74“I_ABORT” | |
| 处理 S7 通信 | S7 通信使用应用层进行数据交换不基于网络，可以使用 MPI、PROFIBUS、工业以太网。通信时，必须先建立 S7 通信连接 | |
| | SFB/FB8“USEND” | 发送数据不带有确认信息。S7-300 系列 PLC MPI 网络不支持，在 PROFIBUS、工业以太网上，必须通过通信处理器 CP 建立通信连接并调用 FB8/FB9 进行通信，S7-400 系列 PLC 没有限制。与 SFB/FB9“URCV”相对应 |
| | SFB/FB9“URCV” | 与 SFB/FB 8“USEND”相对应 |
| | SFB/FB12“BSEND” | 发送数据带有确认信息。S7-300 系列 PLC MPI 网络不支持，在 PROFIBUS、工业以太网上，必须通过通信处理器 CP 建立通信连接并调用 FB12/FB13 进行通信，S7-400 系列 PLC 没有限制。与 SFB/FB13“BRCV”相对应 |
| | SFB/FB13“BRCV” | 与 SFB/FB12“BSEND”相对应 |
| | SFB/FB14“GET” | 将通信方的数据存储于本地数据区，通过单方编写可以进行数据交换。S7-300 系列 PLC 在 MPI 网络只能作为数据服务器（被 S7-400 系列 PLC 进行读写操作），在 PROFIBUS、工业以太网上，必须通过通信处理器 CP 建立通信连接。调用 FB14/FB15 进行通信 |
| | SFB/FB15“PUT” | 将本地数据区数据存储于通信方的数据区中，通过单方编写可以进行数据交换。与 SFB/FB14“GET”方式相同 |
| | SFB16“PRINT” | 发送数据到打印机（CP441） |
| | SFB19“START” | 将通信方暖启动或冷启动进行初始化 |
| | SFB20“STOP” | 将通信方切换到 STOP 模式 |
| | SFB21“RESUME” | 将通信方热启动进行初始化 |
| | SFB22“STATUS” | 查询通信方的状态 |
| | SFB23“USTATUS” | 接收通信方状态改变后发送的状态信息 |
| | SFC/FC62“CONTROL” | SFC 查询 S7-400 系列 PLC（本地站内）连接状态，FC 查询 S7-300 系列 PLC（本地站内）连接状态 |
| 处理通过块调用产生的消息 | 产生的消息上传到 HMI 中，与 HMI 创建的报警信息不同，块调用产生的消息只有在消息出现时才由 CPU 主动发送到 HMI 中，减少通信负荷，HMI 创建的报警信息由 HMI 周期扫描信号位的变化。S7-300 系列 PLC 不支持大部分 SFB 触发的信息 | |
| | SFB36“NOTIFY” | 监控一个信号，发送消息不带有确认 |
| | SFB31“NOTIFY_8P” | SFB36“NOTIFY”功能扩展，同时监控 8 个信号 |
| | SFB33“ALARM” | 监控一个信号，发送消息带有确认 |
| | SFB34“ALARM_8” | 监控八个信号，发送消息带有确认 |

（续）

| 系统功能分类 | 相关 SFC、SFB | 简要功能描述 |
|---|---|---|
| 处理通过块调用产生的消息 | SFB35“ALARM_8P” | 监控八个信号，发送消息时可以附加某个相关的过程值并带有确认 |
| | SFB37“AR_SEND” | 发送归档数据到 HMI 日志中。归档数据最大 65534B |
| | SFC9“EN_MSG” | 使能 CPU 发送消息功能 |
| | SFC10“DIS_MSG” | 禁止 CPU 发送消息功能 |
| | SFC17“ALARM_SQ” | 监控一个信号，发送消息时可以附加某个相关的过程值，接收消息后必须确认。同时调用个数与 CPU 资源有关，查看 CPU 订货手册 |
| | SFC18 ALARM_S | 与 SFC17 功能相同，消息由内部确认 |
| | SFC19“ALARM_SC” | 查询确认状态 |
| | SFC107“ALARM_DQ”/SFC108“ALARM_D” | 替代 SFC17/18 功能，带有资源管理功能 |
| | SFC105“READ_SI” | 读出被 SFC17/18 占用的资源 |
| | SFC106“DEL_SI” | 删除占用资源 |
| IEC 定时器、计数器 | IEC 定时器、计数器与 STEP7 中集成的定时器、计数器相比数据格式不同，定时时间及计数的范围扩大，占用 DB 的个数 | |
| | SFB0“CTU” | 向上计数器 |
| | SFB1“CTD” | 向下计数器 |
| | SFB2“CTUD” | 上下计数器 |
| | SFB3“TP” | 脉冲定时器输出 |
| | SFB4“TON” | 接通延时定时器 |
| | SFB5“TOF” | 断电延时定时器 |
| 处理 S7-300C 特殊功能 | SFB44“ANALOG” | 通过集成的模拟量接口实现简单定位功能 |
| | SFB46“DIGITAL” | 通过集成的数字量接口实现简单定位功能 |
| | SFB47“COUNT” | 控制集成高速计数器功能 |
| | SFB48“FREQUENC” | 控制频率测量功能 |
| | SFB49“PULSE” | 调制脉冲宽度输出 |
| | SFB60“SEND_PTP” | 控制 ASCII 码、3964R 协议发送（串口通信） |
| | SFB61“RCV_PTP” | 控制 ASCII 码、3964R 协议接收（串口通信） |
| | SFB62“RES_RECV” | 删除 ASCII 码、3964R 协议接收缓存区（串口通信） |
| | SFB 63“SEND_RK” | 控制 RK512 协议报文发送（串口通信） |
| | SFB64“FETCH” | 控制 RK512 协议报文“FETCH”功能，从通信方（服务器）得到数据（串口通信） |
| | SFB65“SERVE” | 将本方设置为数据服务器（使用 RK512 协议） |
| 处理冗余系统功能 | SFC90“H_CTRL” | 控制冗余系统同步及后台检测功能 |

SFC 与 SFB 实际上是 PLC 系统与用户程序的接口函数、函数块，通过调用 SFC/SFB，

实现对 PLC 内部信息的读取和功能的控制。下面以例子的方式介绍 SFC/SFB 的使用。

1. 设置 CPU 的时钟

查看表 7-11，通过调用 SFC0，可以设置 CPU 的时钟信息，时钟信息的数据类型为复合数据类型 DT（DATE_AND_TIME），共 8B（按字节排序表示的内容为：年、月、日、时、分、秒、毫秒中前两个有效数字、第 8 个字节高 4 位为毫秒中第三个有效数字，低 4 位为星期值），不能使用普通数据类型（普通数据类型最多 4B）接收时钟信息，复合数据类型 DT 必须在 DB 或 L 堆栈（临时变量）中预先定义，例如在 DB1 中建立一个 DT 类型的变量，变量的符号名称为“Clock”，如图 7-26 所示。

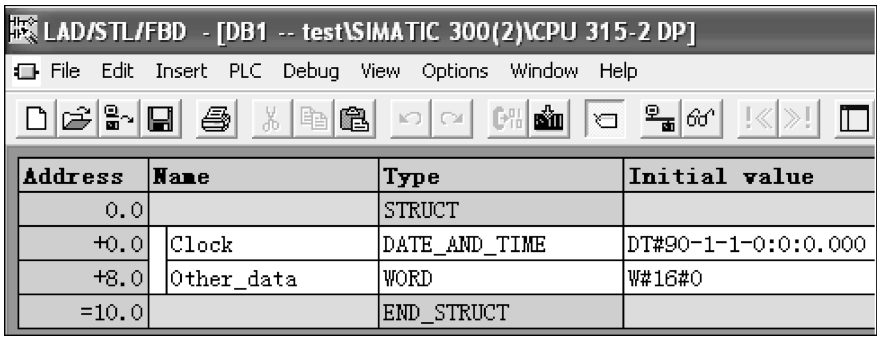



图 7-26 在 DB1 中创建 DT 数据类型变量

在 OB1 中调用 SFC1，程序如下：

```
      L     B#16#11              //设置月信号为 11 月
      T     DB1. DBB    1
      L     B#16#12              //设置日信号为 12 日
      T     DB1. DBB    2
      L     B#16#24              //设置分信号为 24min
      T     DB1. DBB    4
      L     B#16#15              //设置秒信号为 15s
      T     DB1. DBB    5
CALL  SFC     0                  //调用设置时钟系统函数 SFC0
      PDT     := DB1. Clock      //设定的时钟信息
      RET_VAL:= MW2              //块调用返回值
```

系统函数 SFC0 执行后，修改时钟的月、日、分、秒信号，其他保持原有时钟值，调用 SFC0 时，应注意只能采用信号触发的方式调用，否则时钟信号保持不变。

2. 读出带有诊断功能模拟量输入模块的断线诊断信息

模拟量模块连接 4 ~ 20 mA 信号，如果信号低于 3. 6mA 时，模块判断连接传感器的导线断开并报警，通过调用 SFC51 或 SFC59，可以将诊断信息读出。模块的诊断功能必须预先设定，SM331 模块参数化界面如图 7-27 所示。

选择电流测量，可以使能断线诊断功能，如果激活“Diagnostic Interrupt”诊断中断，断

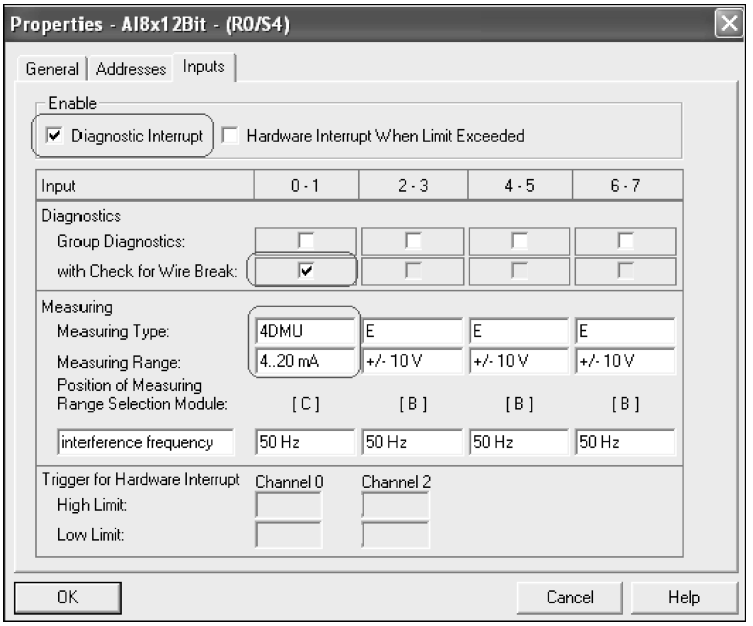



图 7-27　SM331 参数化界面

线故障同时触发 OB82（选择激活“Diagnostic Interrupt”功能，故障出现时，调用 OB82 一次。同时调用 SFC51 或 SFC59 的数量受到 CPU 资源的限制，如果在 OB82 中调用，只有在故障事件发生时，才调用 SFC51、SFC59，节省 CPU 内部通信资源）。

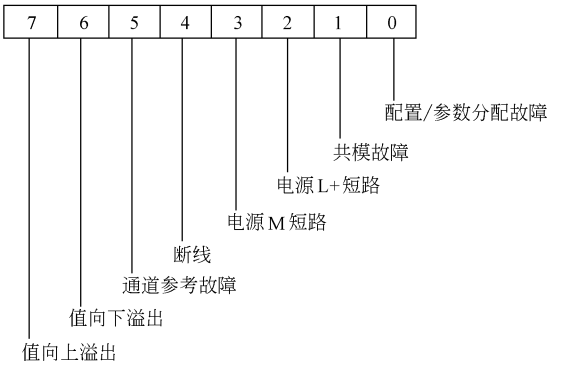



图 7-28　通道诊断故障信息

SM331 模块诊断信息存储于模块数据记录区 1 中，共 16B 长度，基于通道诊断信息存储于 9～16B 中，共 8 个通道，每个通道占用一个字节，每个字节的位信号分别表示故障的类型，如图 7-28 所示。

假设模块的逻辑地址为 256，调用 SFC51 或 SFC59 都可以读出第一个通道的断线诊断信息，在 OB82 中调用 SFC51 读出 SM331 模块诊断信息的示例程序如下：

```
L     #OB82_MDL_ADDR          // 如果触发 OB82 调用的模块地址，不等于 W
                                 #16#100 跳转到 M1
L     W#16#100
 ==I
JCN   M1
CALL  SFC     51              // 如果触发 OB82 地址，等于 W#16#100，调
                                 用 SFC51
 REQ          :=TRUE          // 执行 SFC51
```

```
SZL_ID      := W#16#B3          // SSL_ID 为 W#16#B3，读出模块诊断信息
INDEX       := W#16#100         // 模块的逻辑地址
RET_VAL     := MW22
BUSY        := M1.2
SZL_HEADER := #SZL_HEADER       // 输出结构类型变量，包括两个字元素，第一
                                   个字为指示读出信息的字节长度，第二个字
                                   为读出的数据记录
DR          := P#DB1.DBX0.0 BYTE 16
                                // 读出的信息存储于 DB1.DBB0 ~ DBB15 中
M1:  NOP  0
```

在 OB82 中，调用 SFC59 同样可以读出 SM331 模块的诊断信息，示例程序如下：

```
L     #OB82_MDL_ADDR        // 如果触发 OB82 调用的模块地址不等于
                               W#16#100跳转到 M1
L     W#16#100
 ==I
JCN   M1
CALL  SFC   59              // 如果触发 OB82 地址等于 W#16#100，调用
                               SFC59
 REQ      := TRUE           // 执行 SFC59
 IOID     := B#16#54        // 模块的类型，B#16#54 为输入，B#16#55 为输
                               出
 LADDR    := W#16#100       // 模块的逻辑地址
 RECNUM   := B#16#1         // 数据记录区 1
 RET_VAL  := MW2
 BUSY     := M1.2
 RECORD   := P#DB1.DBX0.0 BYTE 16
                            // 读出的信息存储于 DB1.DBB0 ~ DBB15 中
M1:  NOP 0
```

读出第一通道的诊断信息存储于 DB1.DBB8 中，如果 DB1.DBX8.4 为 1，连接传感器的导线断开。

## 7.7 STEP7 集成用于逻辑运算的函数与函数块

安装 STEP7 后，在函数库中的“Standard Library”中，提供一些用于计算功能、数据转换功能的 FC 及 FB，这些 FC、FB 可以在用户程序中直接调用，如图 7-29 所示。

S5-S7 函数库：用于将 S5 PLC 程序中使用的标准函数转换为适合 S7 PLC 程序使用的函数及函数块，适用于 S5 程序转换为 S7 程序。

IEC 函数库 ：符合 IEC 标准的逻辑及数据转换函数，例如字符串的比较、整数转换为字符串等函数。

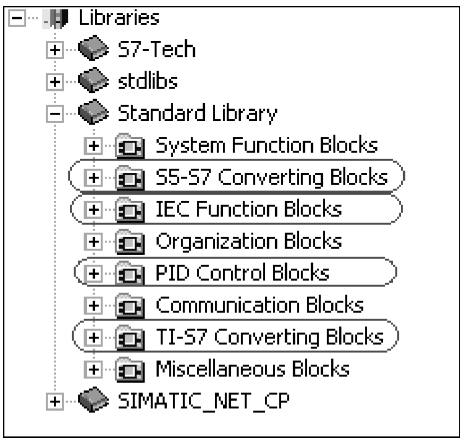



图 7-29 逻辑运算函数库

PID 函数库 ：用于简单的 PID 运算、温度控制的函数块。

TI-S7 函数库：用于在 TI PLC 程序中使用的标准函数转换为适合 S7 PLC 程序使用的函数及函数块，也可以作为普通函数和函数块在 S7 程序中的调用，函数库中提供一些类似北美、日本 PLC 计算功能的函数及函数块，例如数据堆栈、查表功能等。

## 7.8 用于特殊功能的函数与函数块

特殊函数与函数块用于 PLC CPU 与带有处理器模块（通信处理器 CP、功能模块 FM）间进行的数据交换（通过发送命令、接收反馈信息控制模块的运行）、时间标签的处理、冗余 I/O 的处理等功能，例如几乎所有的功能模块都需要安装配置软件，软件中提供相应的函数及函数块，这些函数与函数块都是在内部将多个标准逻辑运算函数、函数块及系统函数、函数块组合而生成一个新的函数或函数块，这些组合的函数、函数块具有新的功能，便于用户的使用。

# 第 8 章　地 址 寻 址

S7-300、400 系列 PLC 地址寻址分为直接地址寻址和间接地址寻址两大类，直接地址寻址通过指令直接对地址进行访问，地址通常是一个常数，不可以改变；间接地址寻址时地址存储于地址指针中，地址是一个变量，程序执行时才能确定实际的地址。在各自寻址的方式下又进行细的划分，不同的寻址方式如图 8-1 所示。

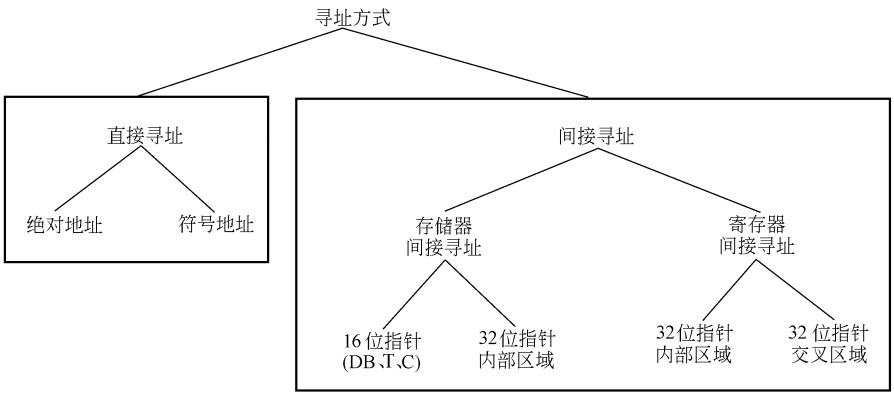



图 8-1　寻址方式

## 8.1　绝对地址寻址与符号地址寻址

绝对地址寻址是指存储单元地址可以直接包含在指令中，指令通过地址标识符号直接对变量进行读写访问，在 STEP7 中，I/O 信号、标志位存储区、定时器、计数器、程序块、数据块等都可以通过绝对地址进行访问，地址区及直接访问举例见表 8-1 所示。

**表 8-1　直接地址寻址类型**

| 地址区符号及访问长度 | 说　　明 | 举　　例 |
|---|---|---|
| I/IB/IW/ID | 过程映像区输入 | I1.0、IB2、IW4、ID12 |
| Q/QB/QW/QD | 过程映像区输出 | Q3.2、QB12、QW20、QD40 |
| PIB/PIW/PID | 外设输入（或立即读） | PIB256、PIW300、PID400 |
| PQB/PQW/PQD | 外设输出（或立即写） | PQB256、PQW288、PQD300 |
| M/MB/MW/MD | 标志位存储区 | M4.0、MB3、MW12、MD42 |
| L/LB/LW/LD | 区域数据 | L2.2、LB1、LW20、LW42 |
| T | 定时器 | T1 |
| C | 计数器 | C1 |

（续）

| 地址区符号及访问长度 | 说　明 | 举　例 |
|---|---|---|
| FC/FB/SFC/SFB | 程序块 | FC1、SFC67 |
| DBX/DBB/DBW/DBD | 数据块（使用 OPN DB）* | DBX12.0、DBB20、DBW40、DBD100 |
| DIX/DIB/DIW/DID | 数据块（使用 OPN DI）* | DIX12.0、DIB20、DIW40、DID100 |

* DB 的访问也可以直接带有 DB 号，例如 DB1.DBX20.0。

I、Q、M、L 有位寻址、字节寻址、字寻址、双字寻址几种方式。PI 类型和 PQ 类型只有字节寻址、字寻址、双字寻址几种方式。

符号地址寻址是指为每个绝对地址分配一个符号名称，便于识别和记忆，增强用户程序的可读性，便于设备的调试，例如用符号“Motor_Start”来代替绝对地址 I1.0。STEP7 中符号寻址分为全局符号（在符号表中定义）和区域符号（在程序块的通信接口及临时变量、静态变量中定义），全局符号在整个用户程序范围内有效，局部符号只能在一个程序块内部使用。

## 8.2 间接寻址

间接寻址分为存储区间接寻址和寄存器间接寻址，前者寻址的地址指针存储于存储器中，如 M、L 区等，后者存储于 CPU 的地址寄存器 AR1 或 AR2 中。

### 8.2.1 存储器间接寻址

存储于存储器中的地址指针分为 16 位地址指针和 32 位地址指针，允许存储地址指针的存储器为 M（标志位）、L（区域数据）及数据块（DB 或 DI），下面分别介绍两种地址指针的使用。

1. 16 位地址指针

16 位地址指针用于定时器、计数器、程序块（DB、FC、FB）的寻址，16 位指针被看作是一个无符号整数（取值范围 0～65535），它指向定时器（T）、计数器（C）、数据块（DB、DI）或程序块（FB、FC）的号，16 位指针的格式如下：

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

$2^{15}$ $2^{7}$ $2^{0}$

$2^5+2^2+2^0=37$（指针指向 37）

所有的定时器、计数器及程序块都可以使用间接寻址访问，访问时需要使用 T、C、DB、DI、FB、FC 等区域标识符，寻址的指针存储于一个 16 位的字中，地址寻址表示格式为：区域标识符［16 位地址指针］，例如打开一个 DB 表示为：

OPN　　DB　　［MW20］

打开指令　区域标识符　地址指针

使用16位地址指针访问一个定时器和计数器的示例程序如下：

```
L     11                    //将11传送到累加器1中
T     MW    20              //将累加器1中的数值传送到MW20中
A     I       2.1           //如果I2.1为1，将预置值10s装载到T11中
L     S5T#10S
SE    T［MW20］
L     MW    20
L     1
+I
T     MW    22              //MW20再加1
A     I       2.2           //如果I2.2为1，C12向上计数一次
CU    C［MW22］
```

从上面的示例程序中，可以看出地址指针存储于MW20中，可以使用普通指令对MW20进行操作，MW20中存储数值的变化直接影响定器、计数器访问的地址。

数据块可以使用DB打开，也可以使用DI打开，如果地址指针为0，CPU不会报错，使用16位地址指针访问数据块的示例程序如下：

```
L     20
T     MW    40
OPN   DB［MW40］            //打开DB20
L     1
+I
T     MW    42
OPN   DI［MW42］            //打开DI21
```

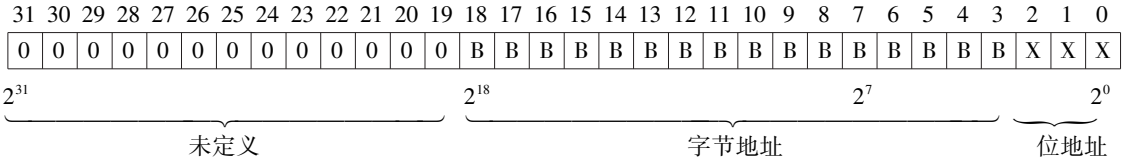
使用16位地址指针调用程序块（FB、FC）时，只能使用UC（无条件调用）或CC(有条件调用)，而不能使用CALL指令，函数块中不能带有任何接口参数或静态变量，使用指针调用函数及函数块的示例程序如下：

```
L     12
T     LW    20
UC    FC［LW20］            //无条件调用FC12
L     13
T     MW    20
A     I     2.3
CC    FB［MW20］            //如果I2.3为1，调用FB13
```

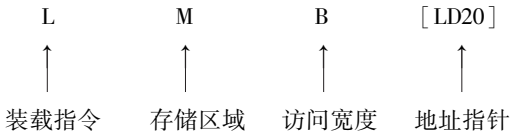
FC12和FB13不能带有形参，这是由CC和UC调用指令决定的。

2. 32位地址指针

32位地址指针用于I、Q、M、L数据块等存储器中位、字节、字及双字的寻址，32位的地址指针可以使用一个双字表示，第0位～第2位作为寻址操作的位地址，第3位～第18位作为寻址操作的字节地址，第19位～第31位没有定义，32位地址指针的格式如下：




访问时需要使用地址存储器标识符及 32 位地址指针，地址寻址表示格式为：
地址存储器标识符［32 位地址指针］，例如指针存储于 LD20 中，装载 M 存储器一个字节表示为：




32 位地址指针也可以使用常数表示，例如装载 32 位指针常数 L P#40.3（P = 指针，字节地址 = 40，位地址 = 3）。32 位地址指针数据与双整数可以相互转换，由于指针指到一个位地址上，地址指针最小变化为一个位，相邻两个位的指针转换为双整数相差 1，例如 P#0.0 转换双整数为 L#0，P#0.1 转换双整数为 L#1，每一个字节地址加 1，相应转换的整数值加 8 的倍数，例如 P#2.0 转换双整数为 L#16，P#3.1 转换双整数为 L#25，在指针寻址时，可以使用指针的格式，也可以使用整数格式进行运算。使用 32 位地址指针寻址参考下面的示例程序：

```
OPN     DB      1       //打开 DB1
OPN     DI      3       //打开 DB3，最多可以同时打开两个 DB
L       4               //装载 4 到累加器 1 中
SLD     3               //累加器 1 中数值左移 3 位
T       MD      20      //将逻辑操作结果传送到 MD20 中，MD20 包含地址
                          指针为 P#4.0
L       P#20.0          //将地址指针 P#20.0 装载到 MD24 中
T       MD      24
L       320             //320 转换指针为 P#40.0 并装载到 MD28 中
T       MD      28
L       DBW［MD20］     //装载 DB1.DBW4
L       DBW［MD24］     //装载 DB1.DBW20
+I                      //相加
L       DIW［MD28］     //装载 DB3.DBW40
-I                      //相减
T       DIW     2       //将运算结果传送到 DB3.DBW2 中
```

使用 32 位地址指针寻址数据块地址时，数据块必须先打开，然后才能寻址，数据块寻址方法参考下面的示例程序，如果直接使用指令，则对完整数据格式地址（例如地址 DB1.DBB［MD100］）进行间接寻址被视为非法。

使用 LOOP 指令和 32 位地址指针可以进行循环操作，假设一个编程应用：一个字变量（MW2）与一个数组（假设存储于 DB1 中，包含 100 个元素为字的数组）存储的值相比较，

如果数值相同，则指出第一个相同数值存储在 DB 中的位置（数组中的位置）。使用通常的编程方法，需要逐字进行比较，因而程序量比较大，如果实际需要与很多个数值比较，一个小型的 CPU 将无法完成控制任务，使用 LOOP 指令和地址指针相结合的方法，可以轻松地解决上述问题，参考下面的示例程序：

```
      L     0                  //初始化 MW100 和 MD4
      T     MW    100
      T     MD    4
      OPN   DB    1            //打开 DB1
      L     100                //循环操作的次数，100 次
next: T     MW    100          //将循环 100 次装载到 MW100 中，固定格式
      L     MW    2            //进行比较的数值存储于 MW2
      L     DBW [MD4]          //与 DB 中存储的值进行比较，开始地址为 DBW0
      =I                       //如果数值相等跳到 m1
      JC    m1
      L     MD    4            //将地址指针加 2（每个相邻的字地址相差 2）
      L     P#2.0
      +D
      T     MD    4
      L     MW    100          //次数减 1，跳回 next，如果 MW100 等于 0，跳出循
                                 环操作 LOOP 指令，LOOP 指令固定格式
      LOOP  next
m1: FP    M     10.0         //如果数值相等，记录 MD4 存储的指针，转换为数
                                 组的位置［（地址值/P#2.0）+1］值，并存储于
                                 MD8 中
      JCN   m2
      L     MD    4
      L     P#2.0
      /D
      +     1
      T     MD    8
m2: NOP   0
```

上述示例程序中编程量减少，但是程序执行时间并没有减少。32 位地址指针指向一个位地址，在编程应用中注意 P#1.0 不等于整数 1，否则 CPU 会因为无法处理而停机，参考下面错误的程序：

```
      OPN   DB    1
      L     20
      T     MD    20           //MD20 装载的地址指针为 P#2.4
      L     11
      T     DBB [MD20]         //指针指向 P#2.4，相当于 L DBB2.4，CPU 因无法
```

识别地址而停机

**编程小技巧：**

如果对字节进行操作，则指针转换为双整数最小变化率必须为 8（指针为 P#1.0）的倍数；如果对字进行操作，则指针转换为双整数值最小变化率必须为 16（指针为 P#2.0）的倍数；如果对双字、浮点变量进行操作，则指针转换为双整数值最小变化率必须为 32（指针为 P#4.0）的倍数，对指向字与双字的地址指针，这样的要求避免数据间的冲突，例如 DBW［MD2］，MD2 为 16 的倍数时，按照 DBW2、DBW4、DBW6 寻址，如果为 8 的倍数，按照 DBW1、DBW2、DBW3 寻址，则造成数据冲突。

## 8.2.2 寄存器间接寻址

与存储器间接寻址不同，寄存器间接寻址使用 CPU 内部集成的两个 32 位地址寄存器 AR1、AR2 存储地址指针。使用地址寄存器指令可以对 AR1、AR2 地址寄存器进行操作，指令如下所示：

LAR1 ：将 ACCU1 存储的地址指针写入 AR1。
LAR1 <D> ：将指明的地址指针写入 AR1，例如 LAR1 P#20.0 或 LAR1 MD20。
LAR1 AR2 ：将 AR2 的内容写入 AR1。
LAR2 ：将 ACCU1 存储的地址指针写入 AR2。
LAR2 <D> ：将指明的地址指针写入 AR2，与 LAR1 <D> 使用方式相同。
TAR1 ：将 AR1 存储的地址指针传输给 ACCU1。
TAR1 <D> ：将 AR1 存储的地址指针传输给指明的变量中。
TAR1 AR2 ：将 AR1 存储的地址指针传输给 AR2。
TAR2 ：将 AR2 存储的地址指针传输给 ACCU1。
TAR2 <D> ：将 AR2 存储的地址指针传输给指明的变量中。
CAR ：交换 AR1 和 AR2 的内容。

寄存器间接寻址分为 32 位内部区域指针和 32 位交叉区域指针。

1．32 位内部区域指针

32 位内部区域指针用于 I、Q、M、L 数据块等存储器中位、字节、字及双字的寻址，与 32 位存储器指针的使用相同，不同之处在于指针存储的位置不同。32 位内部区域地址指针的格式如下：

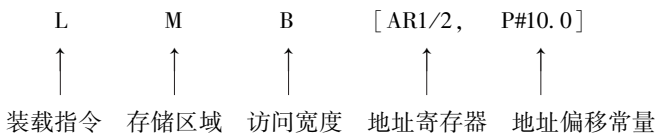
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | X | X | X |

$2^{31}$ $2^{18}$ $2^{7}$ $2^{0}$

未定义 字节地址 位地址

第 0 位～第 2 位作为寻址操作的位地址，第 3 位～第 18 位作为寻址操作的字节地址，第 19 位～第 30 位没有定义，第 31 位为内部区域与交叉区域指针标识，0 表示内部区域指针，1 表示交叉区域指针。

32 位内部区域指针地址寻址表示格式为：地址存储器标识符［地址寄存器，地址偏移常量］，例如装载 M 存储器一个字节表示为：

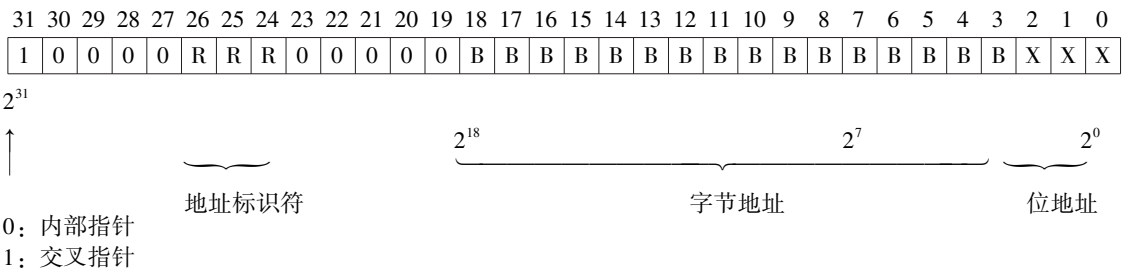



指针指向地址 = 地址寄存器存储地址 + 地址偏移常量，如果 AR1 装载的地址为 P#8.0，实际装载的地址为 MB18。32 位内部区域指针的使用方法参考下面的示例程序：

```
OPN     DB      1               //打开 DB1
LAR1    P#10.0                  //将指针 P#10.0 装载到地址寄存器 1 中
L       DBW [AR1, P#12.0]       //将 DBW22 装载到累加器 1 中
LAR1    MD      20              //将存储于 MD20 中的指针装载到地址寄存器 1
                                  中
L       DBW [AR1, P#0.0]        //将 DBW 装载到累加器 1 中，地址存储于 MD20
                                  中
+I
LAR2    P#40.0                  //将指针 P#40.0 装载到地址寄存器 2 中
T       DBW [AR2, P#0.0]        //运算结果传送到 DBW40 中
```

2．32 位交叉区域指针

32 位交叉区域指针与 32 位内部指针相比，地址指针中带有存储区域标识符，如 I、Q、M 等，32 位交叉区域地址指针的格式如下：




第 0 位～第 2 位作为寻址操作的位地址，第 3 位～第 18 位作为寻址操作的字节地址，第 24 位～第 26 位为地址标识符，表示的地址区域如下：

000 表示没有地址区，例如 P#12.0；

001 表示输入地址区 I，例如 P#I12.0；

010 表示输出地址区 Q，例如 P#Q12.0；

011 表示标志位地址区 M，例如 P#M12.0；

100 表示数据块（DB）中的数据，例如 P#DB1.DBX12.0；

101 表示数据块（DI）中的数据，例如 P#DI1.DIX12.0；

110 表示区域地址区 L，例如 P#L12.0；

111 表示调用程序块的区域地址区 V，例如 P#V12.0。

第 31 位为内部区域与交叉区域指针标识，0 表示内部区域指针，1 表示交叉区域指针。使用交叉区域指针的表示方法（例如装载 M 存储器一个字节）为：

```
LAR1/2  P#M20.0             //装载地址指针 P#M20.0 到 AR1 或 AR2
```

```
L          B        [AR1/2,    P#10.0]
↑          ↑        ↑          ↑
装载指令  访问宽度  地址寄存器  地址偏移常量
```

指针指向地址 = 地址寄存器存储地址 + 地址偏移常量，上例中实际装载的地址为 MB30。如果访问一个位信号，则没有访问宽度。32 位交叉区域指针的使用方法参考下面的示例程序：

```
LAR1    P#M 20.0          //将指针 P#M20.0 装载到地址寄存器 1 中
A       [AR1, P#1.1]      //M21.1 “与” 操作
=       Q       1.2       //如果 M21.1 为 1，则输出 Q1.2 为 1
L       P#I 40.0          //将指针 P#I40.0 装载到累加器 1 中
LAR2                      //将累加器 1 中存储的地址指针装载到地址寄存器 2
                            中
L       W [AR2, P#0.0]    //装载 IW40.0 到累加器 1 中
T       MW      60        //将累加器 1 中存储的数值传送到 MW60 中
```

3. 地址寄存器 AR1、AR2 的监控

存储于存储器的地址指针可以直接通过变量监控表监控，存储于 AR1、AR2 中的地址指针则需要使用 STL 程序 “Monitor” 进行监控，监控之前必须在程序块中的菜单进行设置，点击菜单 “Options” → “Customize” → “STL” 界面中，选择 “Address Register1” 和 “Address Register2” 选项，在程序监控中可以显示 AR1、AR2 的内容，如图 8-2 所示。

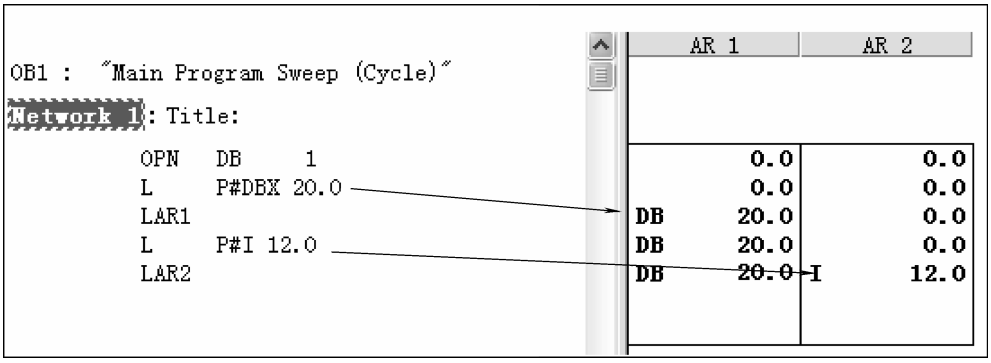



图 8-2 监控 AR1、AR2

4. 地址寄存器 AR1、AR2 的限制

存储区间接寻址和寄存器间接寻址具有相同的功能，在一些特殊条件下，使用地址寄存器 AR1、AR2 会有下列的限制：

1）在形参的传递中，STEP7 使用地址寄存器 AR1 访问函数（FC）接口及函数块（FB）“INOUT” 接口中定义的复合类型参数，如 ARRAY、STRUCT、DATE_AND_TIME 等，AR1 和 DB 寄存器中的内容将被覆盖，例如在 FC1 中 “IN” 接口中定义一个数组变量，在 OB1 中调用，使用 OB1 的 L 区域数据进行赋值，调用关系如图 8-3 所示。

如果在 FC1 中访问数组变量的元素，如 ARR_TEST [1]，地址寄存器 AR1 及 DB 寄存

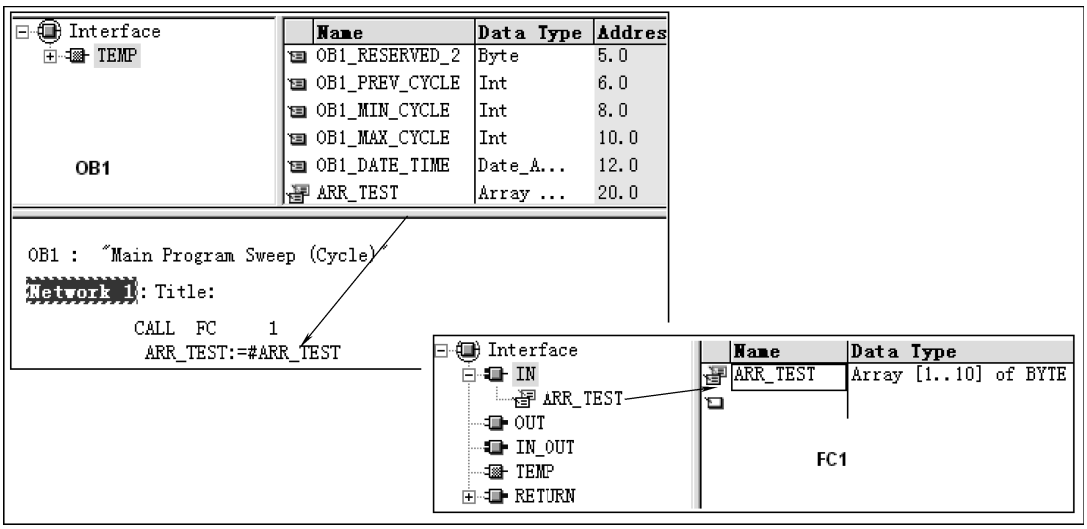



图 8-3　调用关系

器会发生变化，示例程序如图 8-4 所示。

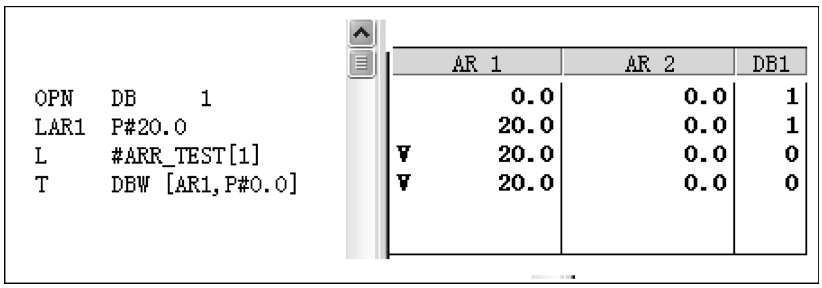



图 8-4　访问 FC 中复合数据类型数据

图 8-4 示例程序中前两条语句中打开 DB1，并将 P#20.0 装载到 AR1 中，在第三条语句访问数组的一个元素后，AR1 存储的地址指针变为 P#V20.0（指向 OB1 中实参 ARR_TEST 的地址，参考图 8-3），DB 寄存器中存储的值为 0，程序下传到 CPU 后将出现故障报警。将程序的执行次序进行修改，CPU 即可运行，修改后的程序如下：

```
L     #ARR_TEST［1］    //装载形参变量 ARR_TEST［1］到累加器 1 中
OPN   DB      1         //打开 DB1
LAR1  P#20.0            //将 P#20.0 装载到地址寄存器 AR1 中
T     DBW［AR1，P#0.0］//将累加器 1 中的值传送到 DB1.DBW20 中
```

执行次序修改后可能会造成不必要的错误，访问复合变量时建议使用存储区间接寻址。访问 FC、FB 区域变量 L，不会覆盖 AR1、DB 寄存器。

2）AR2 和 DI 寄存器分别包含 FB 背景数据块的块号及在背景数据中偏移地址（多重背景数据块），在 FB 中，使用 AR2 和 DI 寄存器将会覆盖系统存储的内容。如果必须在 FB 中使用 AR2 和 DI 寄存器，建议使用下面的方法处理 AR2 和 DI 寄存器，首先保存 AR2 和 DI 寄存器中的数据，程序如下：

```
TAR2    MD      100     //将 AR2 的数据存储于 MD100 中
L       DINO            //将背景数据块 DB 的块号存储于 MW104 中
T       MW      104
```

然后编写用户程序，可以对 AR2 和 DI 寄存器进行操作，但是在程序中不能访问 FB 参数或静态变量。使用完成后恢复 AR2 和 DI 寄存器的系统值，程序如下：

```
LAR2    MD      100     //将 MD100 中存储的地址指针装载到 AR2 中
OPN     DI［MW104］     //打开 DI
```

3）P# < 地址 > 可以直接指向布尔型变量，例如 P#M20.0、P#DBX12.0 等，也可以指向程序块中接口参数区、区域数据区 L 及静态变量，例如 P##PARA（参数）指向 PARA 参数的开始地址。将指向函数 FC 的 IN、OUT、INOUT 类型参数的指针装载到地址寄存器中不可以直接操作，例如：

```
LAR1    P##PARA（参数）  //非法指令
```

必须先装载到累加器中，然后再装载到地址寄存器中，例如：

```
L       P##PARA（参数）  //将地址指针装载到累加器 1
LAR1/2                  //将累加器 1 中存储的地址指针装载到地址寄存器中
```

## 8.3 程序块参数—POINTER 与 ANY 数据类型指针

STEP7 中除提供 16 位、32 位存储器地址指针和 32 位寄存器地址指针外，在程序块 FC、FB 的接口参数中还提供 48 位（“POINTER”数据类型）地址指针和 80 位（“ANY”数据类型）地址指针，16 位和 32 位地址指针可以直接装载到存储器或地址寄存器中，从而可以直接在程序块中使用进行间接寻址，“POINTER”和“ANY”数据类型指针由于大于 32 位而不能装载到存储器或寄存器中，所以不能在程序块中直接使用，必须进行拆分使用。这两种地址指针专用于函数（FC）及函数块（FB）接口参数的传递，例如调用函数赋值形参时，对实参的完全寻址。

### 8.3.1 POINTER 数据类型指针

POINTER 数据类型指针用于向被调用 FC 及 FB 传递复合数据类型（如 ARRAY、STRUCT 及 DT 等）的实参。在被调用的 FC 及 FB 内部可以间接访问实参的存储器。

POINTER 指针占用 48 位地址空间，数据格式如下：

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB 号（或 0） | | | | | | | | | | | | | | | |
| 数据区 | | | | | | | | 0 | 0 | 0 | 0 | 0 | B | B | B |
| B | B | B | B | B | B | B | B | B | B | B | B | B | X | X | X |
| | | | | | | | | B = 字节地址 | | | | | X = 位地址 | | |

POINTER 指针前 16 位的数值表示数据块（DB 或 DI）的块号，如果指针没有指向一个 DB，则数值为 0，POINTER 指针可以指向的数据区见表 8-2。

表 8-2 POINTER 指针数据区表示的地址区

| 16 进制代码 | 数 据 区 | 简 单 描 述 |
|---|---|---|
| B#16#81 | I | 过程映像输入区 |
| B#16#82 | Q | 过程映像输出区 |
| B#16#83 | M | 标志位 |
| B#16#84 | DB | 数据块 |
| B#16#85 | DI | 背景数据块 |
| B#16#86 | L | 区域数据区 |
| B#16#87 | V | 上一级赋值的区域数据 |

与区域交叉指针相比，POINTER 类型指针可以直接指向一个数据块中的变量，例如 P#DB1. DBX0. 0。

调用 FB、FC 时，对 POINTER 指针数据类型的形参进行赋值时，可以选择指针显示方式为直接赋值，例如：

```
P#DB2.  BX12.0          //指向 DB2. DBX12. 0
P#M12.1                 //指向 M12. 1
```

也可以选择使用地址声明或符号名（不使用符号 P#）的方式进行赋值，例如：

```
DB2.  DBX12.0           //指向 DB2. DBX12. 0
M12.1                   //指向 M12. 1
```

在被调用的 FC、FB 中，需要对 POINTER 指针数据类型形参拆分，以便读出实参的地址，下面以示例的方式介绍 POINTER 指针的使用，例如编写一个计算功能的函数（FC3），在输入参数“In_Data”输入首地址，在输入参数“NO”输入变量（浮点格式，以首地址开始，地址连续即每隔 4 个字节为一个浮点变量）的个数，在输出参数“OUT_VAL”输出几个变量的平均值。OB1 中调用函数 FC3 的程序如下：

```
CALL        FC      3       //调用函数 3
In_Data：=P#M100.0          //输入的首地址
NO      ：=4                //变量的个数
OUT_VAL：=MD20              //计算结果
```

完成的计算功能相当于 MD20：=（MD100 + MD104 + MD108 + MD112）/4。在函数（FC3）的接口参数中定义输入、输出变量及临时变量，见表 8-3 所示。

表 8-3 FC3 接口参数

| 数 据 接 口 | 名 称 | 数 据 类 型 | 地 址 |
|---|---|---|---|
| IN | In_Data | POINTER | |
| IN | NO | INT | |
| OUT | OUT_VAL | REAL | |
| TEMP | BLOCK_NO | INT | 0. 0 |
| TEMP | NO_TEMP | INT | 2. 0 |
| TEMP | ADD_TEMP | REAL | 4. 0 |

FC3 中的示例程序如下：

```
      L     0                    //初始化临时变量#ADD_TEMP
      T     #ADD_TEMP
      L     P##In_Data           //指向存储地址指针 P#M100.0 的首地址，并装载到
      LAR1                         地址寄存器 AR1 中
      L     0                    //判断 OB1 中赋值的地址指针是否为数据块（参考
      L     W [AR1, P#0.0]         POINTER 的数据格式）
      =I
      JC    M1
      T     #BLOCK_NO
      OPN   DB [#BLOCK_NO]       //如果是 DB，打开指定的 DB
M1:   L     D [AR1, P#2.0]       //找出需要计算数据区的开始地址，POINTER 数据
                                   中，后 4 个字节包含内部交叉指针，将 P#M100.0
                                   装载到 AR1 中
      LAR1
      L     0
      L     #NO                  //如果输入变量个数为 0，结束 FC3 的执行。如果
      =I                           不等于 0 作为循环执行的次数（NO_TEMP）
      JC    END
NO:   T     #NO_TEMP             //如果不为 0，作为循环执行的次数。循环执行加运
                                   算，本例中循环执行的次数为 4
      L     D [AR1, P#0.0]       //装载 MD100 到累加器 1 中
      L     #ADD_TEMP            //与临时变量#ADD_TEMP 相加后，将计算结果再存
                                   储于#ADD_TEMP 中
      +R
      T     #ADD_TEMP
      +AR1  P#4.0                //地址寄存器加 4，下一次与 MD104 相加
      L     #NO_TEMP             //LOOP 指令固定格式
      LOOP  NO                   //跳回“NO”循环执行，执行完定义在变量#NO_
                                   TEMP 的次数后，自动跳出循环程序
      L     #ADD_TEMP            //求平均值，装载运算结果到累加器 1 中
      L     #NO
      DTR                        //将变量个数转变为浮点值便于运算
      /R
      T     #OUT_VAL             //输出运算结果
END:  NOP   0
```

**注意：**

在 OB1 中调用 FC3 时，如果需要对指针类型的“In_Data”参数赋值为变量（指向的地址区为变量），可以在 OB1 中使用 POINTER 指针变量预先赋值（在临时变量或 DB 中定义

POINTER 数据类型变量），根据 POINTER 指针的数据格式，通过改变 POINTER 指针组成部分的值而改变赋值的地址指针，达到将输入参数“In_Data”作为变量的目的。

## 8.3.2　ANY 数据类型指针

ANY 数据类型指针主要用于为 SFC 及 SFB 分配参数。用户程序中也可以使用 ANY 数据类型指针，作为程序块的接口参数传递数据。

ANY 数据类型指针由数据类型、数据长度、DB 号、存储器数据开始地址组成，占用 80 位地址空间，数据格式如下：

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10H（保留） | | | | | | | | 数据类型 | | | | | | | |
| 数据长度 | | | | | | | | | | | | | | | |
| DB 号（或 0） | | | | | | | | | | | | | | | |
| 数据区 | | | | | | | | 0 | 0 | 0 | 0 | 0 | B | B | B |
| B | B | B | B | B | B | B | B | B | B | B | B | B | X | X | X |

B = 字节地址　　X = 位地址

ANY 指针使用的数据类型见表 8-4。

**表 8-4　ANY 指针使用的数据类型**

| 数据类型代码 | | |
|---|---|---|
| 十六进制代码 | 数 据 类 型 | 简 单 描 述 |
| B#16#00 | NIL | 空 |
| B#16#01 | BOOL | 位 |
| B#16#02 | BYTE | 8 位字节 |
| B#16#03 | CHAR | 8 位字符 |
| B#16#04 | WORD | 16 位字 |
| B#16#05 | INT | 16 位整形 |
| B#16#06 | DWORD | 32 位双字 |
| B#16#07 | DINT | 32 位双整形 |
| B#16#08 | REAL | 32 位浮点 |
| B#16#09 | DATE | IEC 日期 |
| B#16#0A | TIME_OF_DAY（TOD） | 24 小时时间 |
| B#16#0B | TIME | IEC 时间 |
| B#16#0C | S5TIME | SIMATIC 时间 |
| B#16#0E | DATE_AND_TIME（DT） | 时钟 |
| B#16#13 | STRING | 字符串 |
| B#16#17 | BLOCK_FB | FB 号 |
| B#16#18 | BLOCK_FC | FC 号 |
| B#16#19 | BLOCK_DB | DB 号 |
| B#16#1A | BLOCK_SDB | SDB 号 |
| B#16#1C | COUNTER | 计数器 |
| B#16#1D | TIMER | 定时器 |

指针中的数据长度表示指向一个数据区域，例如指向整个数组、结构体等；如果 ANY 指针没有指向一个 DB，DB 号将为 0；ANY 指针的数据区与 POINTER 指针数据区定义相同（参考表 8-2）。

与 POINTER 指针相比，ANY 类型指针可以表示一段数据区域，例如 P#DB1. DBX0. 0 BYTE10，表示指向 DB1. DBB0 ~ DB1. DBB9。调用 FB、FC 时，对 ANY 指针数据类型的形参进行赋值时，可以选择指针显示方式为直接赋值，例如：

```
P#DB2. DBX12. 0  WORD  22        //指向从 DB2. DBW12 开始 22 个字
P#M12. 1  BOOL  10               //指向从 M12. 1 开始 10 个位信号
```

也可以选择使用地址声明或符号名（不使用符号 P#）的方式进行赋值，例如：

```
DB2. DBW12                       //指向 DB2. DBW12 一个字，数据长度为 1
M12. 1                           //指向 M12. 1 一个位信号，数据长度为 1
```

使用地址声明或符号名只能指向一个变量。下面以示例的方式介绍 ANY 指针的使用，实现与 POINTER 指针示例相同的功能。编写一个计算功能的函数（FC13），输入参数“In_Data”为一个数组变量，如果数组元素为浮点数，输出所有元素的平均值“OUT_VAL”，如果数组元素为其他数据类型，不执行计算功能。OB1 中调用函数（FC13）的程序如下：

```
CALL   FC      13                //调用函数 13
In_Data: = P#DB1. DBX0. 0 REAL 8 //输入数据区从 DB1. DBD0 开始 8 个浮点值
OUT_VAL: = MD20                  //计算结果
```

完成的计算功能相当于 MD20: = (DB1. DBD0 + … + … + DB1. DBD28)/8。在函数（FC13）的接口参数中定义输入、输出变量及临时变量见表 8-5 所示。

**表 8-5 FC13 接口参数**

| 数据接口 | 名　称 | 数据类型 | 地　址 |
|---|---|---|---|
| IN | In_Data | ANY | |
| OUT | OUT_VAL | REAL | |
| TEMP | DATA_LEN | INT | 0. 0 |
| TEMP | BLOCK_NO | INT | 2. 0 |
| TEMP | ADD_TEMP | REAL | 4. 0 |
| TEMP | DATA_NO | INT | 8. 0 |

FC13 中的示例程序如下：

```
L     0                  //初始化临时变量#ADD_TEMP
T     #ADD_TEMP

L     P##In_Date         //指向存储地址指针 In_Date 首地址，并装载到
LAR1                     地址寄存器 AR1 中
L     B [AR1, P#1. 0]    //如果数据类型不是 REAL，跳转到 END
L     B#16#8
< >R
JC    END

L     0
```

```
      L     W [AR1, P#4.0]   //判断 OB1 中赋值的地址指针是否为数据块（参考
      =I                       ANY 的数据格式）
      JC    M1
      T     #BLOCK_NO
      OPN   DB [#BLOCK_NO]   //如果是 DB，打开指定的 DB

M1:   L     W [AR1, P#2.0]   //判断 ANY 指针中数据长度，本例中为 REAL 变量的
      T     #DATA_LEN          个数
      L     D [AR1, P#6.0]   //找出需要计算数据区的开始地址，本例中为
                               DB1.DBX0.0
      LAR1
      L     #DATA_LEN
NO:   T     #DATA_NO         //循环执行加运算，本例中循环执行的次数为 8
      L     D [AR1, P#0.0]   //装载 DB1.DBD0 到累加器 1 中
      L     #ADD_TEMP        //与临时变量#ADD_TEMP 相加后，将计算结果再存储
      +R                       #ADD_TEMP 中
      T     #ADD_TEMP
      +AR1  P#4.0            //地址寄存器加 4，地址偏移量
      L     #DATA_NO         //LOOP 指令固定格式
      LOOP  NO               //跳回“NO”循环执行，执行完定义在变量#NO_
                               TEMP 的次数后，自动跳出循环程序
      L     #ADD_TEMP        //求平均值，装载运算结果到累加器 1 中
      L     #DATA_LEN
      DTR                    //将变量个数转变为浮点值
      /R
      T     #OUT_VAL         //输出运算结果
END:  NOP   0
```

**注意：**

在 OB1 中调用 FC13 时，如果需要对指针类型的“In_Data”参数赋值为变量（指向的地址区为变量），可以在 OB1 中使用 ANY 指针变量预先赋值（在临时变量或 DB 中定义 ANY 数据类型变量），根据 ANY 指针的数据格式，通过改变 ANY 指针组成部分的值而改变赋值的地址指针，达到将输入参数“In_Data”作为变量的目的。

## 8.4　FB 在多重数据块中的寻址

在一些编程应用中，一些功能计算或包含部分工艺的函数或函数块往往由开发人员编写，例如编写 FB，通常情况下不会考虑在程序中的调用次序。工程人员在程序中调用这些已经开发完成的 FB，实现整个的控制任务。在一个 FB 中，将实现固定功能的 FB 作为静态变量使用，生成多重背景数据块 DB，每个实现固定功能 FB 的接口参数都会占用多重背景

数据块的空间，如图 8-5 所示。

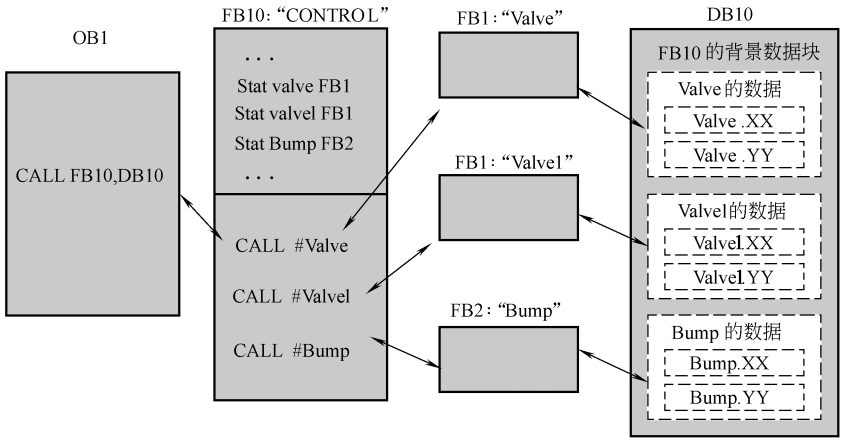



图 8-5 生成多重背景数据块

在 FB10 中，将具有独立控制功能的 FB1、FB2 作为静态变量调用，FB10 在 OB1 中调用生成背景数据块 DB、DB10，在背景数据块中包含 FB10、FB1、FB2 的接口参数，每个 FB 块接口参数在 DB10 中的位置与在 FB10 静态变量中的次序有关。

如果在 FB1、FB2 中使用 POINTER 或 ANY 类型地址指针进行拆分时，如果不考虑在多重背景数据块 DB 中的位置，将会造成错误，例如在 FB1 中定义输入接口参数 FB1_POS，数据类型为 POINTER，FB1 中的程序如下：

```
L     P##FB1_POS          //指向存储地址指针 FB1_POS 首地址
LAR1                      //存储于地址寄存器 1 中
L     D［AR1，P#2.0］     //装载实参赋值的地址指针，并传送到 MD20 中
T     MD      20
```

同样在 FB2 中定义输入接口参数 FB2_POS，数据类型为 POINTER，在 FB2 中的程序如下：

```
L     P##FB2_POS          //指向存储地址指针 FB2_POS 首地址
LAR1                      //存储于地址寄存器 1 中
L     D［AR1，P#2.0］     //装载实参赋值的地址指针，并传送到 MD24 中
T     MD      24
```

在 FB10 中，将 FB1、FB2 作为静态变量使用，见表 8-6 所示。

**表 8-6 FB10 的接口参数**

| 数 据 接 口 | 名　　称 | 数 据 类 型 | 地　　址 |
|---|---|---|---|
| STAT | FB1_POS | FB1 | 0.0 |
| STAT | FB2_POS | FB2 | 6.0 |

FB10 的程序如下：

```
CALL    #FB1_POS          //调用 FB1，赋值地址指针 P#M100.0
FB1_POS：=P#M 100.0
```

```
CALL    #FB2_POS            //调用 FB2，赋值地址指针 P#M120.0
FB2_POS: =P#M 120.0
```

在 OB1 中，调用 FB10，并生成 DB10，程序如下：

```
CALL    FB      10,     DB10        //调用 FB10，生成 DB10
```

希望将 P#100.0 赋值到 MD20 中，将 P#120.0 赋值到 MD24 中，但程序执行后 MD20 和 MD24 存储的地址指针同为 P#M100.0，这是由于 FB1 中的指令 L P##FB1_POS 与 FB2 中的指令 L P##FB2_POS 同时指向多重背景数据块 DB10 中 FB1 接口数据区的首地址 DB10.DBX0.0，以地址 DB10.DBX0.0 开始的 POINTER 指针变量存储的地址指针为 P#M100.0；FB2 接口数据区在 DB10 的首地址为 DB10.DBX6.0，以地址 DB10.DBX6.0 开始的 POINTER 指针变量存储的地址指针为 P#M120.0。在 FB1、FB2 程序中，没有考虑接口数据区在多重背景数据块的偏移地址而造成地址冲突。系统将每一个 FB 接口数据区在多重背景数据块中的偏移地址自动存储于地址寄存器 AR2 中，利用 TAR2 功能可以实现地址偏移。在 FB1 中实现地址自动偏移的示例程序如下：

```
TAR2                        //将偏移地址传送到累加器 1 中
L       DW#16#FFFFFF        //过滤地址区，如将 P#M20.0 变为 P#20.0
AD
L       P##FB1_POS          //将偏移地址与 FB1_POS 首地址相加，并装载到
                              AR1 中
+D
LAR1                        //得到 FB1 在多重背景数据块中的首地址
L       D [AR1, P#2.0]      //将 P#M100.0 装载到 MD20 中
T       MD      20
```

在 FB2 中，按上述的方法编写程序，将 P#M120.0 装载到 MD20 中，这样 FB1、FB2 在 FB10 中作为参数的次序没有限制。

**提示：**

使用间接寻址编程灵活、节省程序空间，但是不容易入门。在实际编程应用中，可选择比较简单、容易上手、易于监控的编程方法。

# 第 9 章　使用 STEP7 创建和编辑项目

一个项目中可以包含多个 PLC 站、HMI、驱动等设备，一个 PLC 站中主要包含系统的硬件配置信息和设备控制的用户程序。硬件配置是对 PLC 硬件系统的参数化过程，使用 SIMATIC Manager 软件，按硬件实际安装次序将 CPU、电源、信号模块等硬件配置到相应的机架上，并对 PLC 硬件模块的参数进行设置和修改。硬件配置对于系统的正常运行非常重要，它的功能如下：

1）配置信息下载到 CPU 中，CPU 功能按配置的参数执行；

2）将 I/O 模块的物理地址映射为逻辑地址，用于程序块调用；

3）CPU 比较模块的配置信息与实际安装的模块是否匹配，例如 I/O 模块的安装位置、模拟量模块选择的连接模式等，如果不匹配，CPU 报警，并将故障信息存储于 CPU 的诊断缓存区中，根据 CPU 提供的故障信息作出相应的修改；

4）CPU 根据配置的信息对模块进行实时监控，如果模块有故障，CPU 报警，并将故障信息存储于 CPU 的诊断缓存区中；

5）一些智能模块的配置信息存储于 CPU 中，例如通信处理器（CP）、功能模块（FM）等，模块故障后直接更换，不需要重新下载配置信息。

本章着重介绍项目中 PLC 站的硬件配置及编程设置。

## 9.1　创建一个项目

SIMATIC Manager 是 STEP7 的主视窗，通过桌面的快捷图标或点击“Start”→“SIMATIC”→“SIMATIC Manager”都可以进入 SIMATIC Manager 配置界面。在 SIMATIC Manager 界面中，创建一个控制项目，并有两种方式：通过向导功能；直接创建。通过向导功能可以创建一个最简单的项目，项目中包含一个 PLC 站，PLC 站包括 CPU、OB 的信息，而直接创建将产生一个空项目，用户需要手动添加项目框架中的各项内容。

### 9.1.1　使用 SIMATIC Manager 向导功能创建一个项目

利用向导创建一个项目的过程如下：打开 SIMATIC Manager，在“File”菜单下单击‘New Project’Wizard...，弹出新建项目向导对话框。用向导创建项目共分为四步：

1）向导的使用说明如图 9-1 所示。在向导创建项目界面的右上角显示向导功能当前的步数，如果选择“Display Wizard On starting the SIMATIC Manager”选项，在每次开启 SIMATIC Manager 时，都会显示向导功能界面。点击“Preview”按钮，可以显示/隐藏对话框下方的预览窗口。在预览窗口中可以立即看到不同选项对将生成的项目产生的影响。点击“Next”按钮进入下一步。

2）选择 CPU 类型如图 9-2 所示，按订货号选择 CPU 型号并配置 MPI 地址，此处配置 CPU 的类型及订货号必须与实际安装的硬件相匹配，选择 CPU 后，在下面的显示框中显示




图 9-1　使用向导创建一个项目第一步

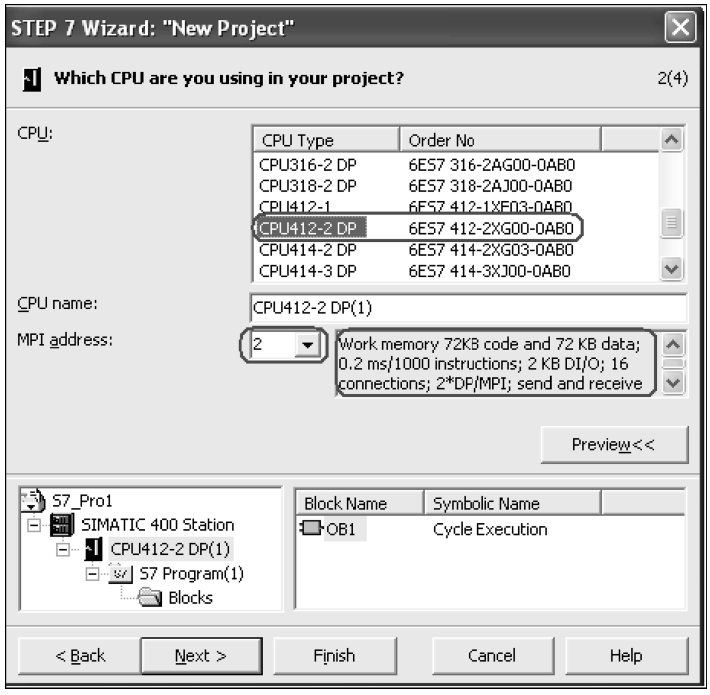



图 9-2　使用向导创建一个项目第二步

CPU 的基本特性。MPI 地址默认值为 2，点击“Next”按钮，进入下一步。

3）添加 OB，如图 9-3 所示。

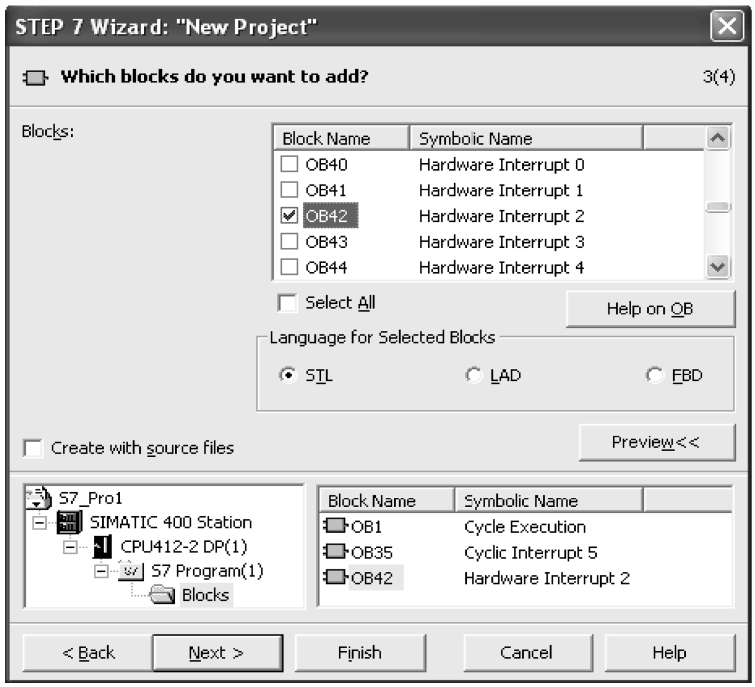



图 9-3 使用向导创建一个项目第三步

用户可以选择需要的 OB 和使用的编程语言，配置的 CPU 必须具有所选择 OB 的功能，否则 OB 不能执行。点击“Next”按钮，进入下一步。

4）定义项目名称如图 9-4 所示。

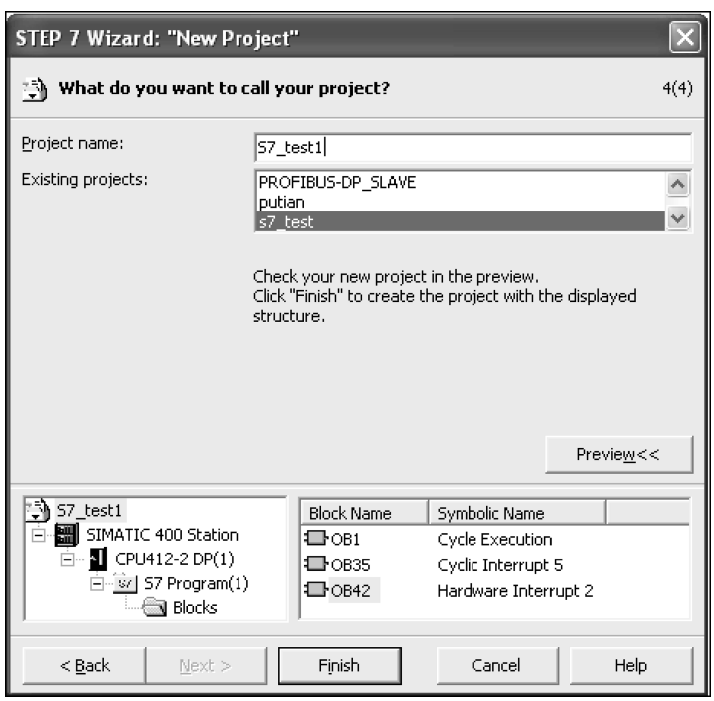



图 9-4 使用向导创建一个项目第四步

在“Project name”一栏中，输入项目名称，点击“Finish”按钮，一个新的项目就创建完成了。使用向导功能创建项目步骤简单，但是设置不灵活，如 CPU 的版本号、项目的存储路径等不能直接定义，必须在生成后进行修改。使用直接创建的方法比较灵活，自由度大。

### 9.1.2　直接创建一个项目

在“File”菜单下，点击“New”命令，或者点击工具栏按钮，可以直接创建一个新项目。在弹出的对话框中输入项目名称，如图 9-5 所示。在“Type”选项中，可以选择创建一个项目、库（程序块不能下传，只能作为库函数被其他项目调用）和多重化项目（一个项目中包含多个站，由多人编程，最后合并为一个项目中）。在“Storage location”中选择项目存储路径。点击“OK”按钮完成新项目的创建。

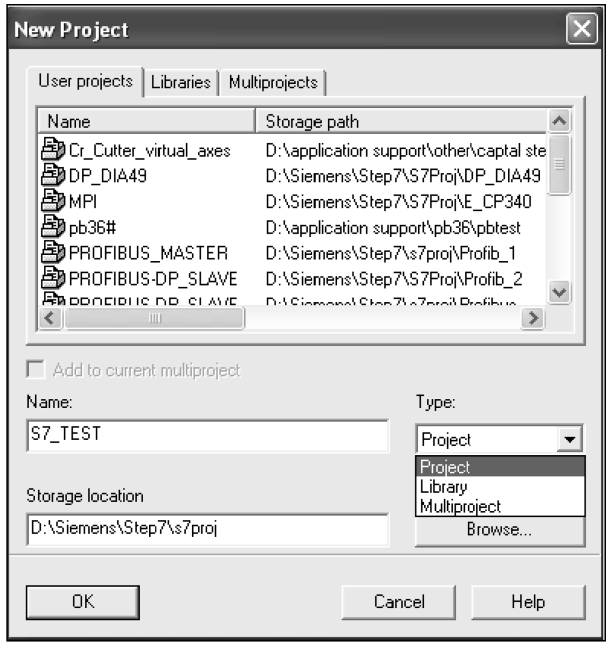



图 9-5　直接创建一个项目

选择创建的项目名，点击鼠标右键或点击菜单“Insert”→“Station”插入一个站或多个站如图 9-6 所示，选择不同型号、用途的 CPU 必须与站的类型相匹配，例如选择使用 S7-400 CPU 必须插入一个“SIMATIC 400 Station”，如果选择其他类型的站，在硬件配置中不能对 S7-400 系列 PLC 进行配置。

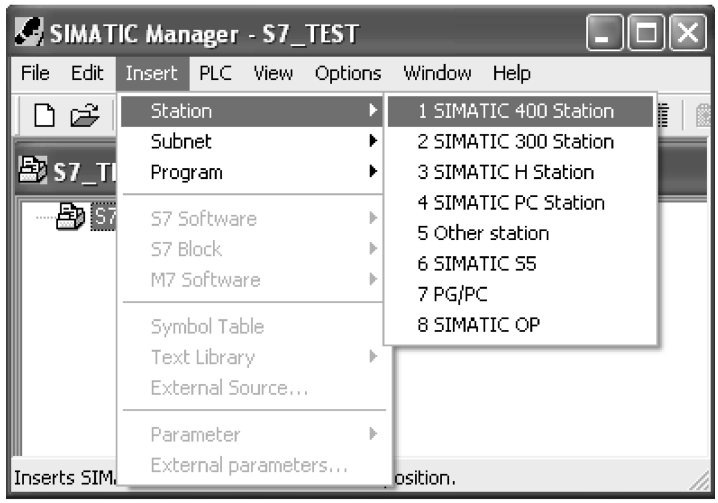



图 9-6　插入一个站

直接创建项目后，需要在后续的工作中手动配置硬件和插入程序块，在下面的章节中将会详细介绍。

## 9.2 项目基本配置

### 9.2.1 项目属性配置

点击项目名，按鼠标右键弹出下拉菜单，选择“Object Properties”，进入项目属性配置界面，如图 9-7 所示。

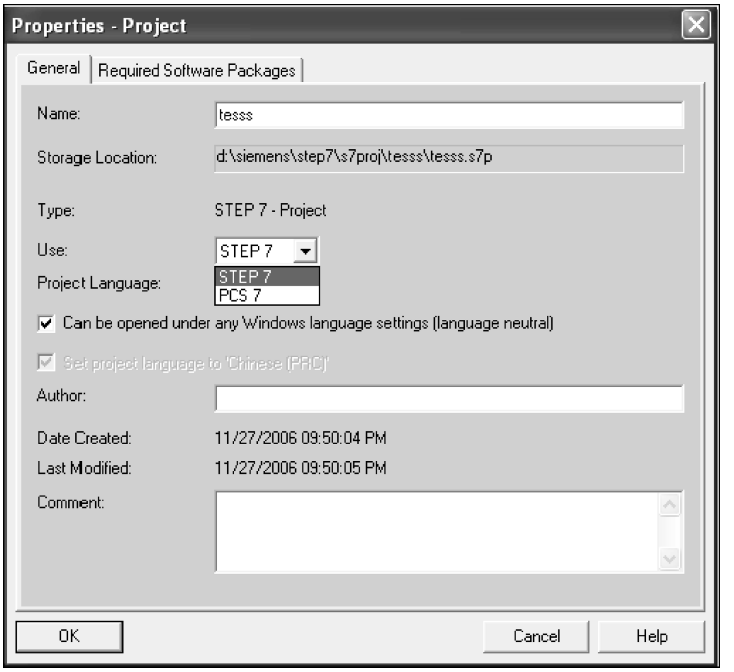



图 9-7 项目属性界面

在项目属性界面中，主要参数的功能描述如下：

1. “Use”项目选项

可以选择“STEP7”项目或“PCS7”项目，如果选择“STEP7”项目，每个程序块可以独立下载到 CPU 中；如果选择“PCS7”项目，为了保证程序的完整性，必须整个站（包括硬件配置和所有程序块）一起下载。

2. “Project Language”设定

显示当前项目使用的语言。使用 STEP7 浏览其他项目程序，如果该项目中使用的语言与编程器设置的语言不同，STEP7 无法打开该项目，在这种情况下，根据打开项目时提示的警告信息，在编程器的语言设置界面中设置与项目相同的语言，这样可以打开项目文件，在项目属性界面中选择“Can be opened under any Windows language settings［language neutral］”选项后，编程器使用任何语言都可以打开该项目。

### 9.2.2　项目用户化设置

开始使用 STEP7 之前，用户可以对 STEP7 的软件使用环境进行设置，打开 SIMATIC Manager，通过菜单命令“Options”→“Customize”打开项目用户化设置菜单，如图 9-8 所示。

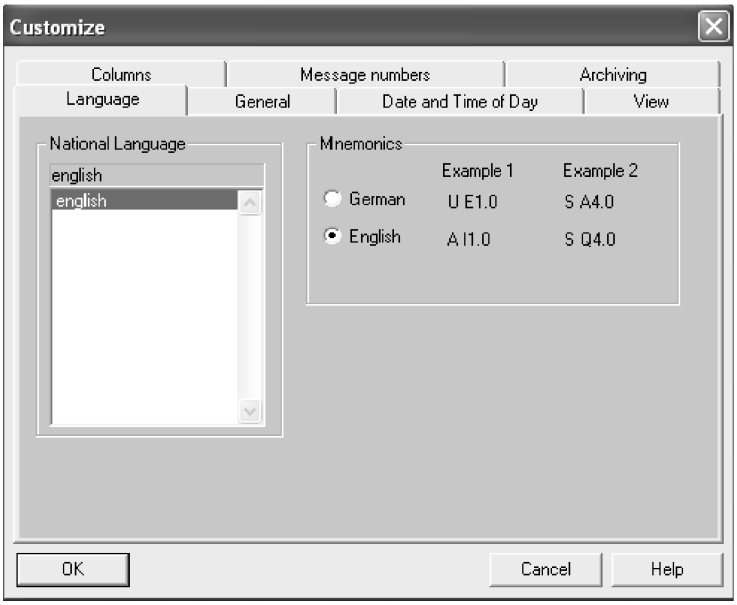



图 9-8　用户化设置界面

在界面中可以选择德语和英语两种编程助记符（Mnemonics）、日期格式、视图显示方式、归档使用的压缩方式以及项目和库函数默认的存储路径等。

## 9.3　硬件配置界面

使用向导功能创建的项目不包含输入、输出以及其他一些模块的信息，必须进行硬件配置。使用直接创建的方式在项目下插入一个站后，没有任何站点的信息，也必须进行硬件配置。选择 PLC 站，双击“Hardware”图标，进入硬件配置界面，如图 9-9 所示。

硬件配置界面主要有三部分：

1）硬件目录窗口，通过工具栏上的目录按钮，可以显示/隐藏硬件目录。硬件目录按分布式 I/O PROFIBUS-DP、PROFINET IO 以及 PLC 站 SIMATIC 300、SIMATIC 400 等子目录划分，硬件目录中包含所有与 PLC 站相关的硬件信息条目，包括订货号及版本号等，如果硬件目录中不包含实际硬件的条目，则需要更新硬件目录或升级 STEP7 版本。选择其中一个硬件条目，在硬件目录的下方显示当前选中条目的属性信息，例如模块的功能、接口特性、对特殊功能的支持等。

2）选择的硬件条目，可以拖放到硬件配置视窗中进行配置，下方的硬件配置信息视图则显示了机架中所插入的模块，以及模块的订货号、版本、地址分配等详细信息。

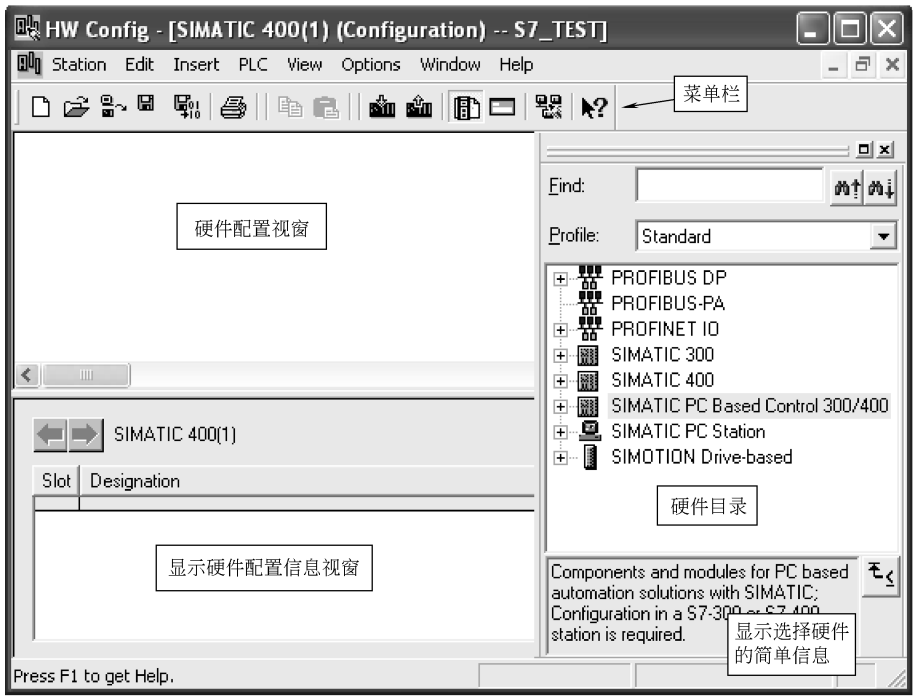



图 9-9 硬件配置界面

3）在菜单栏中可以选择快捷键进行操作，如上传、下载硬件信息等，硬件初始配置时，CPU 中不包含任何配置信息，不能将实际的硬件信息上传到 PC 上，必须手动配置。

## 9.4 配置中央机架及扩展机架

中央机架中带有 CPU 模块，通过接口模块可以进行机架的扩展，扩展机架上不能插入 CPU 模块。根据不同的扩展接口，有的扩展机架上带有通信总线可以插入通信处理器（CP）模块及功能模块（FM）不带有通信总线的扩展机架上只能插入 I/O 模块。

### 9.4.1 配置 S7-300 系列 PLC 中央机架

配置 S7-300 中央机架，必须遵循以下规则：

1）1 号槽只能放置电源模块，由于电源模块不带有背板总线接口，可以不进行硬件配置。

2）2 号槽只能放置 CPU 模块，不能为空。

3）3 号槽只能放置接口模块，如果一个 S7-300 系列 PLC 站只有主机架，而没有扩展机架，则主机架不需要配置接口模块，但是 3 号槽必须预留（实际的硬件排列仍然是连续的）。

4）由于机架不带有背板总线，相邻模块间不能有空槽位。

5）SIM 374 IN/OUT 16 数字仿真模块专用于实验室，不能连接实际的输入、输出设备，

在硬件目录里并不存在。因此，在配置该模块时，应该添加需要仿真的模块，而不是该模块本身。

6）4 ~ 11 号槽可放置最多 8 个信号模块、功能模块或通信处理器，与模块的宽窄无关。如果需要配置更多的模块，则需要进行机架扩展。

使用 SIMATIC Manager 进行硬件配置的过程与硬件实际安装过程相同，在项目中插入一个“SIMATIC 300 Station”，双击“Hardware”图标，进入硬件配置界面。在硬件目录中找到 S7-300 机架，双击或使用拖放的方法添加到硬件配置视图中，配置的机架中带有 11 个槽位，按实际需求及配置规则，将硬件分别插入到相应的槽位中，如图 9-10 所示。

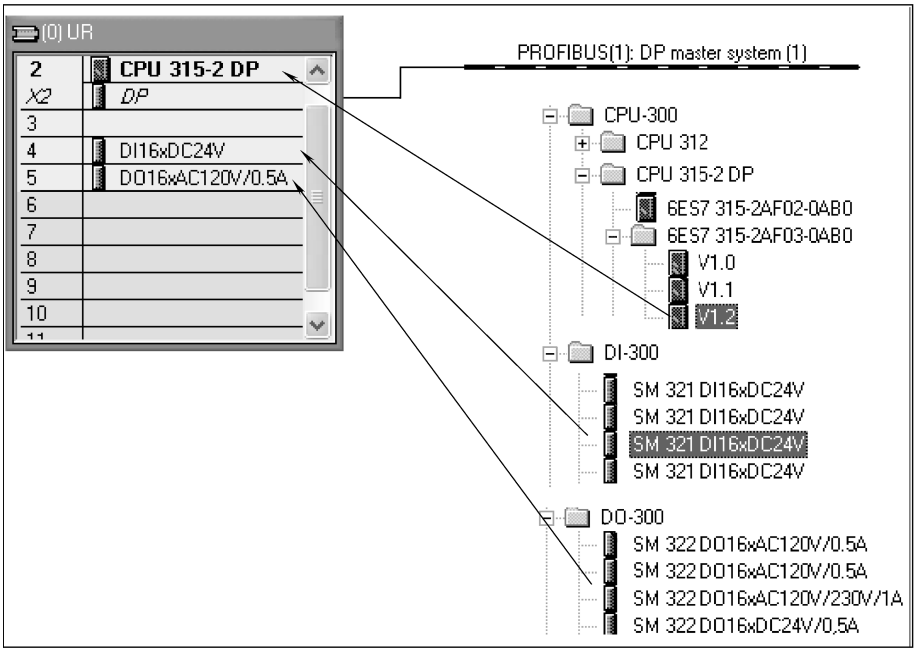



图 9-10　插入 S7-300 系列 PLC 模块

在 2 号槽添加 CPU。先选择需要插入的槽号，再双击选择的 CPU 或使用鼠标将选择的 CPU 拖放到相应的槽位中。硬件目录中的某些 CPU 型号有多种固件（Firmware）版本，在添加 CPU 时，CPU 的型号和固件版本都要与实际硬件一致。使用相同的方法在 4 ~ 11 号槽中可以插入信号模块、功能模块、通信处理器等硬件，上述模块分别在硬件目录中的 SM-300、FM-300 和 CP-300 目录下。例如，在中央机架中添加了一个数字量输入模块和一个数字量输出模块。

在配置过程中，STEP7 可以自动检查配置的正确性。有些模块不能插入 S7-300 系列 PLC 主机架中使用，只能放在分布式 I/O 站上使用。当在硬件目录中选择一个模块时，机架中允许插入该模块的槽位会变成绿色，而不允许该模块插入的槽位颜色无变化。如果使用鼠标拖放的方法将选中的模块拖到允许插入的槽位时，鼠标指针变为，如果将模块拖放到禁止插入的槽位上，鼠标指针变为，同时会提示不能插入该模块的原因，如图 9-11 所示。

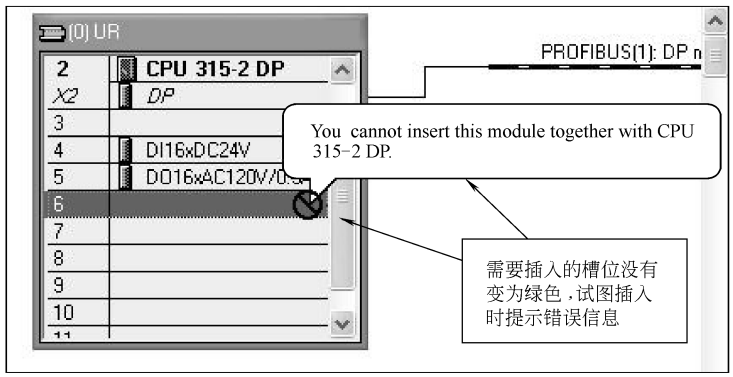



图 9-11 插入模块时的自检测功能

## 9.4.2 配置 S7-300 系列 PLC 扩展机架

一个 S7-300 站最多可以有一个主机架（0 号机架）和 3 个扩展机架（1 ~ 3 号机架）。主机架和扩展机架通过接口模块（IM）连接。S7-300 系列 PLC 接口模块的种类参考 2.7.1 节。

机架扩展有以下两种情况：

1）只有一个扩展机架时，可以使用 IM365 成对接口模块进行扩展，主机架（0）和扩展机架（1）的 3 号槽中分别插入 IM365 成对接口模块。扩展机架不带有通信总线，不能插入 FM 和 CP 模块，由于背板总线电源由 CPU 提供，两个机架上所有模块消耗背板总线的电流不能超过 CPU 的输出。

2）有 1 ~ 3 个扩展机架时，可以使用 IM360、IM361 成对接口模块进行扩展，主机架（0）的 3 号槽位中插入 IM360 接口模块，扩展机架（1 ~ 3 号）3 号槽位中插入 IM361 接口模块。扩展机架带有通信总线，可以插入 FM 和 CP 模块，IM361 需要 24V 电源供电，同时 IM361 向扩展机架的背板总线提供 5V 电源，驱动连接的模块。

在硬件配置中，可以像添加中央机架一样，通过拖放的方法在站窗口中插入扩展机架（与中央机架相同），分别在中央机架和扩展机架中插入相应的接口模块，机架之间的连接自动建立，然后在机架上插入所需的模块，如图 9-12 所示。

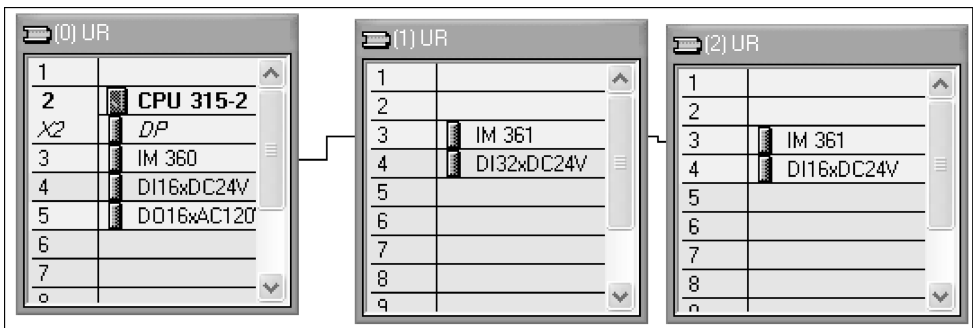



图 9-12 配置 S7-300 系列 PLC 的中央扩展

### 9.4.3　配置 S7-400 系列 PLC 中央机架

与 S7-300 系列 PLC 中央机架的配置相比，S7-400 系列 PLC 的中央机架可选择的型号多，由于有些机架带有背板总线，对模块插入槽位的限制少。

下列机架可以作为中央机架：

1）CR2：18 槽机架，机架内部划分为两段（10 + 8）。

2）CR3：4 槽机架，适用于使用 S7-400 系列 PLC 的 CPU 连接分布式 I/O，通常中央机架不插入 I/O 模块。

3）UR1：通用 18 槽机架，可以作为中央机架和扩展机架。

4）UR2：通用 9 槽机架，可以作为中央机架和扩展机架。

5）UR2ALU：通用 9 槽铝制机架，可以作为中央机架和扩展机架。

S7-400 系列 PLC 中央机架配置需要遵循以下规则：

1）电源模块只能插入机架从第一个槽号开始的槽位上。

2）CPU、I/O 模块、接口模块和 CP 及 FM 可以在机架任意槽位中安装。

3）一个机架插入模块的数量与机架的槽位、通信资源和消耗背板总线电流有关，有些模块可能占用多个槽位，例如 20A 的电源模块需要占用 3 个槽位。

4）机架中带有背板总线，机架中相邻两个模块之间可以有空槽位。

使用 SIMATIC Manager 进行硬件配置与 S7-300 系列 PLC 的配置过程相同，首先在项目中插入一个“SIMATIC 400 Station”，双击“Hardware”图标，进入硬件配置界面。在硬件目录中找到 S7-400 系列 PLC 机架，选择实际使用的机架后双击或使用拖放的方法添加到硬件配置视图中，按实际需求及配置规则，将硬件分别插入到相应的槽位中，如图 9-13 所示。

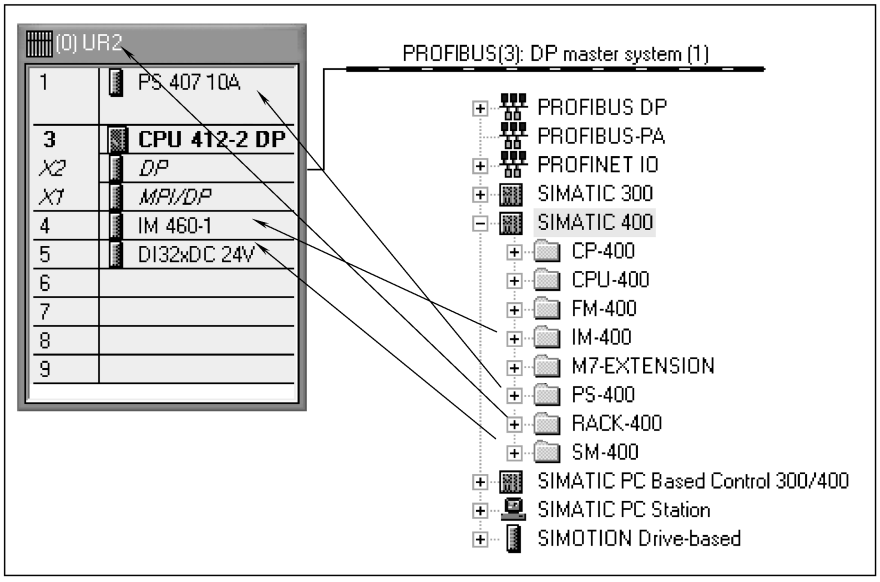



图 9-13　插入 S7-400 系列 PLC 模块

在 1 号槽添加电源（PS），先选择需要插入的槽号，再双击选择的 PS 或使用鼠标将选择的 PS 拖放到相应的槽位中，图 9-13 中 PS 模块占用两个槽位。在其他的槽位中任意插入选择的模块，没有槽位限制。硬件目录中的某些 CPU 型号有多种固件（Firmware）版本，在插入 CPU 时，CPU 的型号和固件版本都要与实际硬件一致。

STEP7 自动检查 S7-400 系列 PLC 配置的正确性，与 S7-300 系列 PLC 检查方式相同，在硬件目录中选择一个模块时，机架中允许插入该模块的槽位会变成绿色，而不允许该模块插入的槽位颜色无变化。如果使用鼠标拖放的方法，将选中的模块拖到允许插入的槽位时，鼠标指针变为  ，如果将模块拖放到禁止插入的槽位上，鼠标指针变为  ，同时会提示不能插入该模块的原因。

### 9.4.4 配置 S7-400 系列 PLC 扩展机架

下列机架可以作为扩展机架：

1）ER1：18 槽扩展机架，不带有通信总线，只能插入 I/O 信号模块（SM）。

2）ER2：9 槽扩展机架，不带有通信总线，只能插入 I/O 信号模块（SM）。

3）UR1：通用 18 槽机架，可以作为中央机架和扩展机架。

4）UR2：通用 9 槽机架，可以作为中央机架和扩展机架。

5）UR2ALU：通用 9 槽铝制机架，可以作为中央机架和扩展机架。

**注意：**

*扩展机架上带有的通信总线可以插入 CP 和 FM，通用机架是否带有通信总线能力与接口的模块有关，S7-400 系列 PLC 接口模块的详细信息参考 2.7.2 节。*

S7-400 系列 PLC 扩展机架配置需要遵循以下规则：

1）接口模块成对使用，发送与接收接口模块必须匹配。

2）配置时，注意发送接口模块的特性，例如 IM460-4，不传输电源和通信总线，那么在扩展机架上必须插入电源模块供电，且不能插入带有通信总线接口的 FM、CP。

3）接收接口模块放置在扩展机架的最后一个槽位上，如果需要插入电源模块，必须放置在扩展机架上第一个槽号开始的槽位上，其他模块可以任意放置。

4）如果 CP443-5 或 IM467 模块需要作为 PROFIBUS-DP 主站连接远程 I/O，必须放置在中央机架中。

**注意：**

*大多数接口模块在扩展时，需要在最后一个扩展机架上插入终端电阻，否则整个扩展机架不能被 CPU 识别。*

配置扩展机架，首先在中央机架上插入发送接口模块，如 IM460-X，在硬件条目视窗中，通过双击需要插入的机架或使用鼠标拖放的方式在站窗口中插入扩展机架。在扩展机架上插入接收接口模块，如 IM461-X 和所需的 I/O 模块，双击发送接口模块进入属性界面，选择扩展机架进行连接，如图 9-14 所示。

通常情况下，发送接口模块带两个扩展接口，例如 C1、C2，可以将扩展机架连接到不同的扩展接口上（如果分别连接到两个扩展接口，有两个扩展网络，则需要两个终端电阻），选择扩展机架，点击“Connect”按钮，连接到选择的扩展接口上，点击“Disconnect”

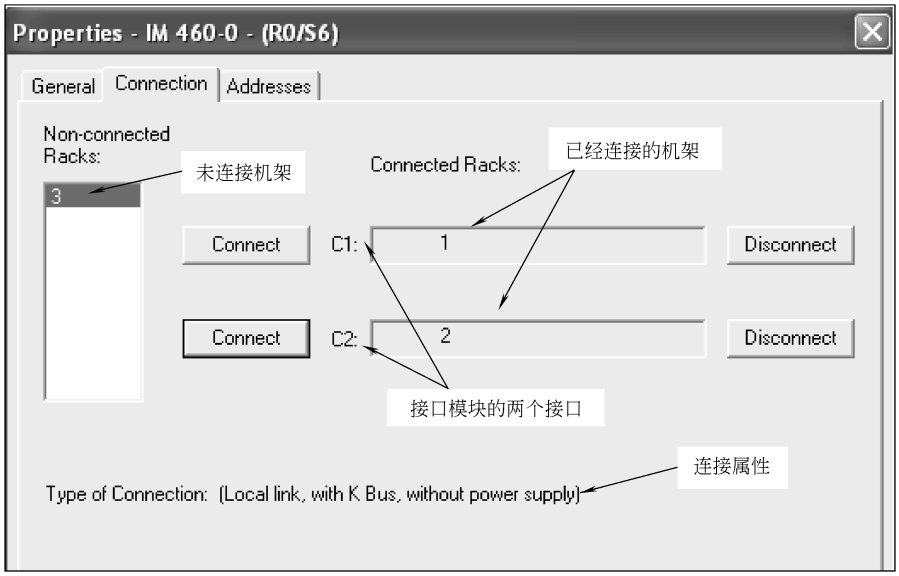



图 9-14　配置 S7-400 系列 PLC 中央扩展的连接路径

按钮可以重新进行连接分配。

**注意：**

扩展连接时，必须与实际连接相匹配。

连接配置完成后，在硬件配置界面中，可以看到中央机架与扩展机架自动连接，如图 9-15 所示。IM460-0 提供两个扩展接口，分别连接扩展机架 1、2、3。

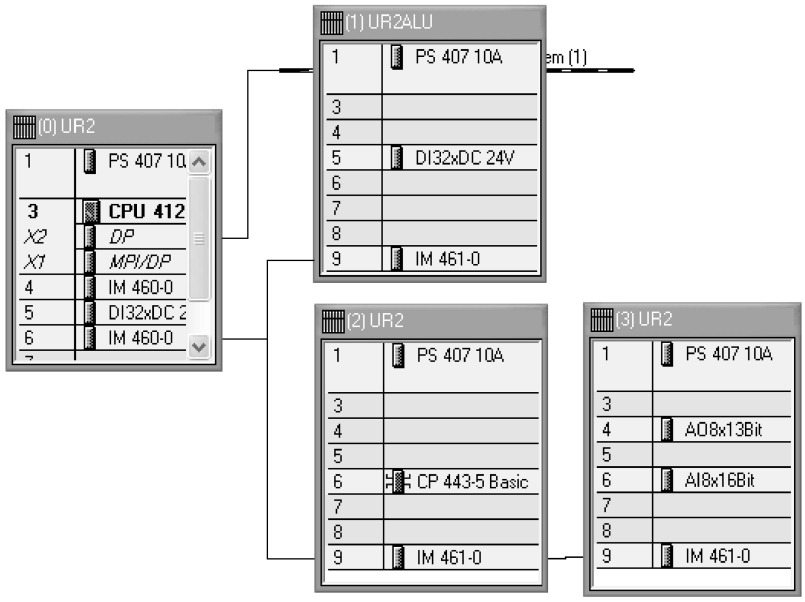



图 9-15　配置 S7-400 系列 PLC 的中央机架扩展机架连接

## 9.5 CPU 参数配置

双击机架中的 CPU，打开 CPU 属性对话框，在这里可以配置 CPU 的各种参数，如 CPU 的启动特性、OB 以及存储区的参数设置等，CPU 属性对话框按照参数类别显示为若干标签页。由于 S7-400 系列 PLC CPU 的功能完全覆盖 S7-300 系列 PLC CPU 的功能，以 S7-400 系列 PLC 为例介绍 CPU 的参数设置。

### 9.5.1 常规界面

点击“General”标签，进入常规界面，如图 9-16 所示。S7-400 系列 PLC 的常规界面中没有参数设置，只有一些 CPU 特性的描述，用户可以在“Name”、“Comment”等空白处作一些提示性的标注。

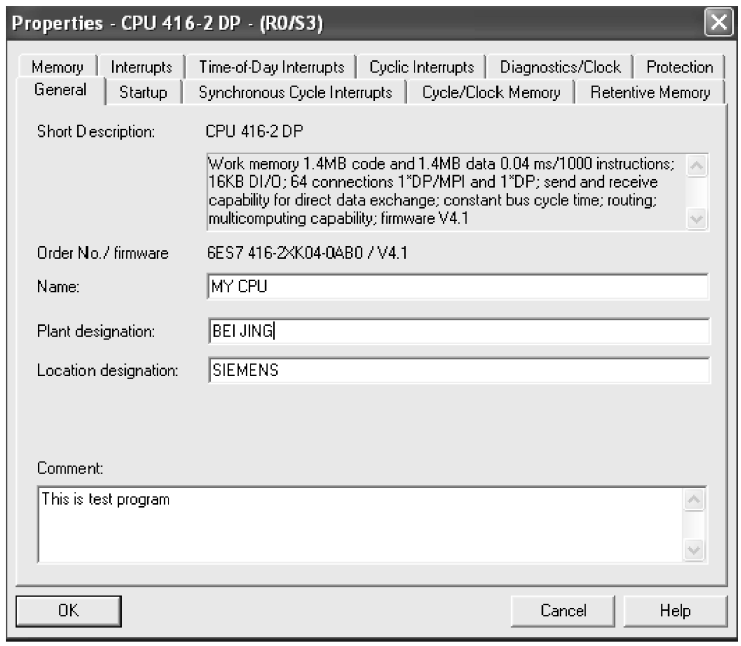



图 9-16 S7-400 系列 PLC CPU 常规界面

有些 S7-300 系列 PLC 的常规界面中可以设置 CPU 的 MPI，例如 CPU315-2DP，S7-400 站点的硬件配置中，MPI 占用一个子槽位。

### 9.5.2 启动界面

点击“Startup”标签，进入启动界面，所有设置的参数与 CPU 的启动特性有关，如图 9-17 所示。

启动界面主要参数及选项的功能描述如下。

“Startup if preset configuration does not match actual configuration”：该选项决定当硬件配置信息与实际硬件不匹配时，CPU 是否启动。如果取消该选项，CPU 检测到硬件配置信息与实际硬件不匹配时，CPU 启动后进入停止模式。

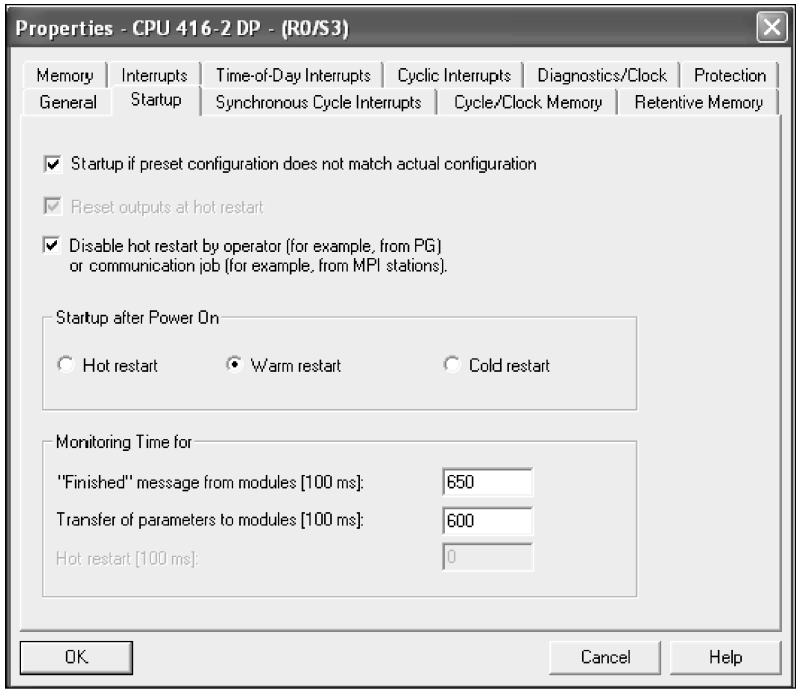



图 9-17　S7-400CPU 启动界面

“Reset outputs at hot restart”（不适用于 S7-300 系列 PLC CPU）：启动模式选择“hot restart”后，该选项被激活，选择热启动后是否复位所有的输出信号（过程映像区输出（PIQ））。

“Disable hot restart by operator. .”（不适用于 S7-300 系列 PLC CPU）：该选项决定是否禁用通过编程器或其他站的通信命令，触发 CPU 热启动功能。

“Startup after Power On”：选择上电后 CPU 的启动特性，大多数 S7-300 系列 PLC CPU 中只有“Warm restart”（有些新的 S7-300 带有“Cold restart”），S7-400 系列 PLC CPU 支持 3 种启动方式，冷、暖、热启动，温度越低，复位 CPU 存储器的数量越多。

“Finished message from modules［100ms］”：这个时间是上电后 CPU 收到各个模块已准备就绪的信号（“Finished”）的最长时间。如果超过这个时间，CPU 还没有收到所有模块准备就绪的信号，就认为实际硬件与配置信息不同。

“Transfer of parameters to modules［100ms］”：这个时间是 CPU 把参数分配到各个模块的最大时间（从收到模块的准备就绪的信息后开始定时），如果超过该时间，仍然没有分配完所有模块的参数，就认为实际硬件与配置信息不同。

“Hot restart［100ms］”：热启动监控时间，如果超出监控时间，S7-400 系列 PLC CPU 不能启动。该选项不适用 S7-300 系列 PLC CPU。

**注意：**

有些情况下，CPU 通过分布式 I/O 站带有智能模块，如 FM，同时上电后，由于 CPU 与智能模块启动的时间不同，智能模块不能被 CPU 识别，需要延长监控时间。

### 9.5.3 同步循环中断

点击“Synchronous Cycle Interrupts”标签，进入同步循环中断界面如图 9-18 所示。

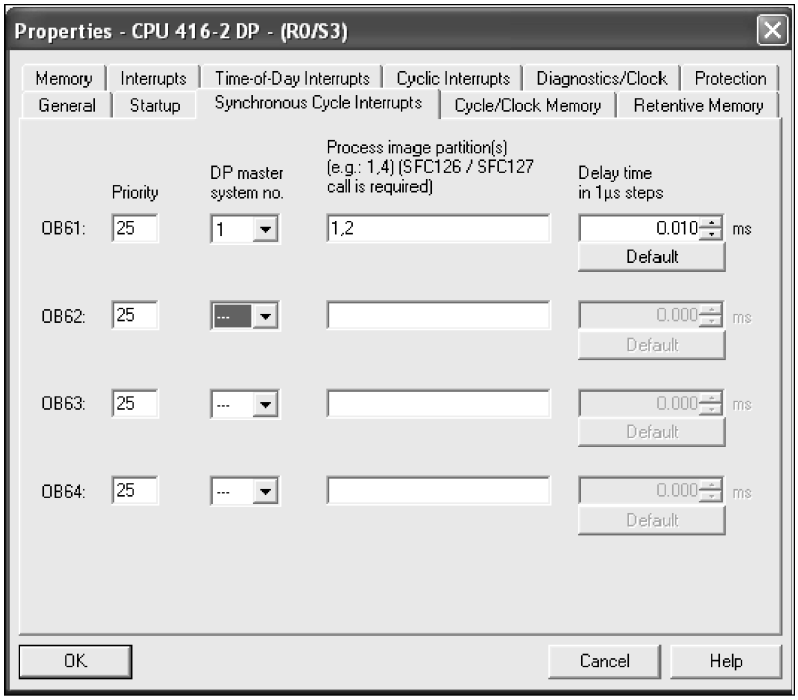



图 9-18 S7-400 系列 PLC CPU PROFIBUS-DP 同步循环中断界面

该界面用于设置 PROFIBUS-DP 等时同步，通常情况下，PROFIBUS-DP 主站轮询多个从站，两个从站同一时刻触发的输入信号进入 CPU 中处理的次序可能不同，同样 CPU 发出的输出命令在各个从站的响应也会有先后，通过 PROFIBUS-DP 等时同步操作，保证 CPU 同时处理不同从站的输入信号，发出的命令在不同从站中同时响应（要求接口模块及 I/O 必须支持等时功能）。S7-300 系列 PLC CPU 没有此功能，CPU31XT-2DP 内置等时功能只能同步西门子驱动设备，不能自由配置其他从站。对等时同步参数化界面主要参数及选项的功能描述如下。

“OB61 ~ OB64”：执行等时同步中断程序，保证从站 I/O 信号处理的快速性、同步性。每一个 OB 被指定处理一个 PROFIBUS-DP 网络的等时同步任务，每次主站轮询从站结束后触发 OB6X 的调用。

“Priority”：定义不同 OB 的优先级，事件同时发生，优先级高的 OB 先触发。

“DP master system no.”：为 OB6X 指定一个 PROFIBUS-DP 网络。

“Process image partition（s）”：过程映像区分区，OB6X 触发时更新过程映像区，可以选择整个过程映像区或过程映像区的一部分，如 1、2 表示更新过程映像区分区 1 和分区 2。

“Delay time”：主站发出同步控制信号到触发 OB6X 运行的延时时间，为了保证等时的快速性，主站读完从站输入信号后，立即触发 OB6X 执行用户程序，点击“Default”键，由系统计算延时时间值。

### 9.5.4　循环/时钟寄存器

点击“Cycle/Clock Memory”标签，进入循环/时钟寄存器界面。在该界面中，设置与 CPU 循环扫描相关的参数及 CPU 集成时钟寄存器，如图 9-19 所示。

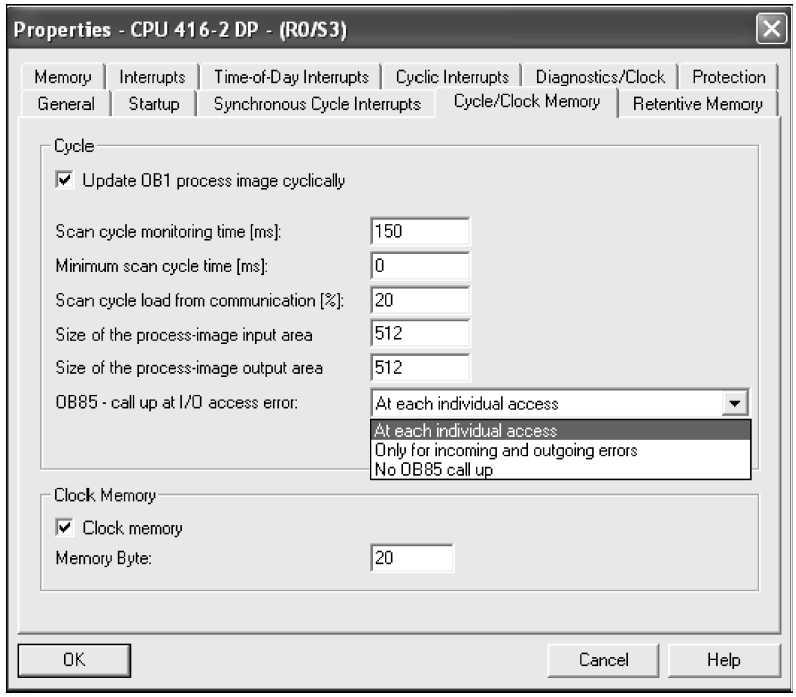



图 9-19　S7-400 系列 PLC CPU 循环/时钟寄存器界面

循环/时钟寄存器参数化界面主要参数及选项的功能描述如下。

“Update OB1 process image cyclically”（不适用 S7-300 系列 PLC CPU）：选择是否激活在 CPU 每一个循环扫描都需要更新 I/O 的过程映像区。S7-400 系列 PLC 的 I/O 信号可以放置在过程映像区分区中，并指定相应的中断 OB，当中断 OB 调用时，更新相应过程映像区分区中的 I/O 信号；通过调用 SFC26 UPDAT_PI、SFC27 UPDAT_PO 系统函数也可以更新选择的过程映像区。过程映像区详细信息可参考 2.3.4 节。

“Scan cycle monitoring time”：设定程序循环扫描的监控时间，如果超过了这个时间，CPU 就会进入停机状态。通信处理、连续调用中断（故障）、CPU 程序故障都会增加 CPU 的扫描时间。在 S7-300 系列 PLC CPU 中，可以在 OB80 中处理超时错误，此时扫描监视时间会加倍。如果此后扫描时间仍然超过了加倍以后的限制，CPU 就会进入停机状态。

“Minimum scan cycle time”（不适用 S7-300 系列 PLC CPU）：在有些应用中，需要设定 CPU 最小的扫描时间，如果实际扫描时间小于设定的最小时间，直到达到最小扫描时间后 CPU 才进行下一个扫描周期，如果用户程序中含有 OB90，在 CPU 等待期间将处理 OB90 中的程序。

“Scan cycle load from communication”：这个参数限制通信在一个循环扫描周期中所占的

比例。例如当循环扫描时间设置为 150ms，通信占用时间比例设置为 20% 时，每个周期分配给通信的时间就是 150ms × 20% = 30ms。因此，在设定循环扫描时间时，除了要考虑 OB1 的执行时间、其他异步事件的处理时间（如中断等），还要考虑通信占用的时间。

“Size of the Process Image”（不适用 S7-300CPU）：设定过程映像区输入、输出的范围。参数为从 0 字节开始的字节数。如果超出设定的范围，使用 P（访问外设）访问 I/O 地址，例如 PIB20。

**注意：**

S7-400 系列 PLC CPU 过程映像区中的每个字节占用 CPU “Code memory” 12 字节，如果设定大的过程映像区将影响 CPU 的存储空间。

“OB85call up at I/O access error”：OB85 用于处理 I/O 访问故障，这里可以设置出现I/O 访问错误时 CPU 不同的响应模式：

“No OB85 call up”：不调用 OB85；

“At each individual access”：选择该项则每一个 I/O 错误都会调用一次 OB85；

“Only for incoming and outgoing errors”：选择该项，则在故障出现和消除时，分别执行 OB85 一次，可以避免 OB85 频繁调用而导致 CPU 扫描时间的增加。

“Clock Memory”：CPU 内部集成的功能，将 8 个固定频率的方波时钟信号输出到一个标志位存储区的字节中，字节中每一位对应的频率和周期见表 9-1。

**表 9-1 时钟寄存器**

| 时钟存储器的位 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 频率/Hz | 0.5 | 0.62 | 1 | 1.25 | 2 | 2.5 | 5 | 10 |
| 周期/s | 2 | 1.6 | 1 | 0.8 | 0.5 | 0.4 | 0.2 | 0.1 |

例如在图 9-19 中，时钟信号存储于 MB20 中，M20.0 即为 100ms 的方波信号。在许多通信程序中，发送块需要脉冲触发，可以简单利用 CPU 集成的时钟寄存器作为脉冲信号。

### 9.5.5 保持存储区

点击“Retentive Memory”标签，进入保持存储区参数化界面。在该界面中，可以设置存储器 M、C、T 掉电保持的范围（S7-400 系列 PLC 需要备份电池保持，S7-300 系列 PLC 掉电后数据存储于 MMC 中），如图 9-20 所示。

保持存储区参数化界面主要参数及选项的功能描述如下。

“Retentivity”：设定 M、C、T 存储区的保持功能，可以分别指定从 MB0、T0 和 C0 开始需要保持的位存储区、定时器和计数器的数目（每个数据区最大保持范围参考订货样本），例如在“Number of memory bytes starting with MB0”中填入 16，则当系统掉电后，或者 CPU 从 STOP 模式转为 RUN 时，从 MB0 到 MB15 这 16 个字节中存储的过程值保持，而没有设置为保持的位存储区将被初始化为 0。

“DB Areas”：带有 MMC 的 S7-300 系列 PLC CPU 和 S7-400 系列 PLC CPU（需要带有备份电池）中所有 DB（除 DB 属性中选择“Non Retain”）都具有掉电保持功能，不需要设置。一些旧版本的 S7-300 系列 PLC CPU 中（对 DB 设置框可以进行设置），仅当 CPU 没有后备电池时，设置的保持 DB 才有效。

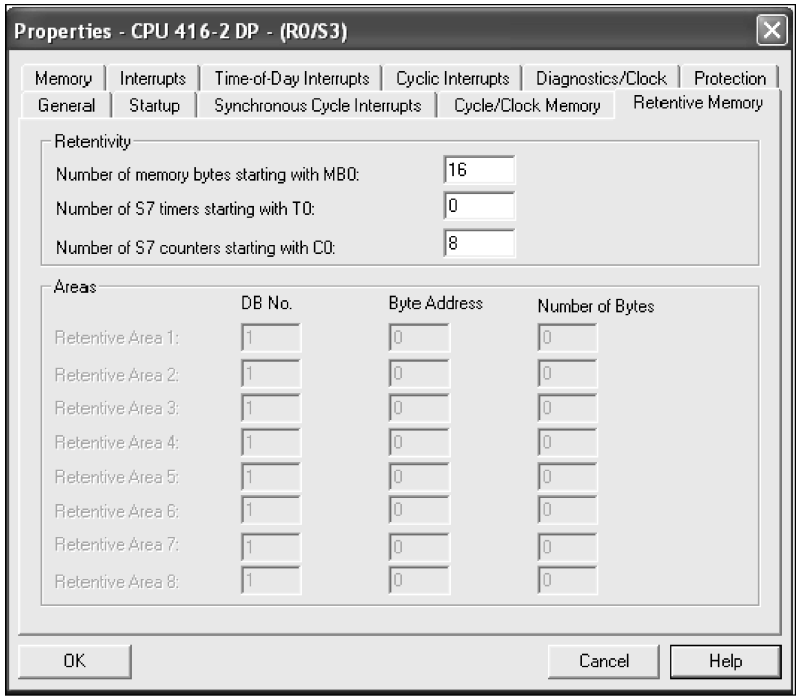



图 9-20　S7-400 CPU 保持存储区界面

### 9.5.6　存储区（不适用 S7-300 系列 PLC CPU）

点击“Memory”标签，进入存储区设置界面。在该界面中，可以设置区域数据区堆栈（L 堆栈）大小及通信资源的数量，这些设置将影响 CPU 存储区的使用，参数化界面如图 9-21

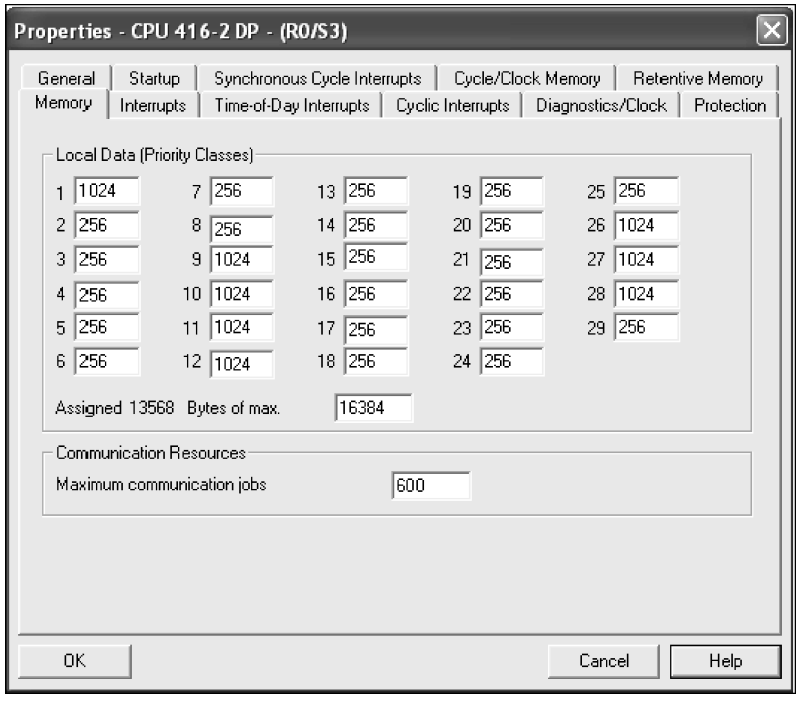



图 9-21　S7-400 CPU 存储区界面

所示。

存储区参数化界面主要参数及选项的功能描述如下：

“Local Data（Priority Classes）”：在 7.2.2 节中已经介绍了 OB 运行及嵌套调用函数、函数块时，需要占用 L 堆栈的大小。每个 OB 都具有优先级，在同一个时刻，优先级相同的 OB 中只能有一个运行，在这里可以为每个优先级相同的 OB 分配 L 堆栈的字节数。分配的字节数必须为偶数，且不能为 2～18，如果某些优先级的 OB 没有使用，可以设置为 0。

“Communication Resources”：分配与“S7connections”相关的最大通信任务，通信函数块包括 STATUS、USEND/URCV、BSEND/BRCV、PUT/GET/PRINT、START/STOP/RESUME、STATUS、USTATUS、ALARM、ALARM_8、ALARM_8P、NOTIFY、AR_SEND 等，每个通信函数块的背景数据块为一个通信任务。

**注意：**

L 堆栈中每个字节占用 CPU “Data memory” 1 字节，每个通信任务占用 CPU “Code memory” 72 字节，设定大的 L 堆栈或多的通信资源将影响 CPU 的存储空间。如果用户程序过大，CPU 内存稍小，在不影响 CPU 工作的情况下，可以减少 L 堆栈的字节数及通信任务数。

### 9.5.7 中断

点击“Interrupts”标签，进入中断设置界面。在该界面中可以设置硬件中断、时间延时中断、异步故障中断等中断优先级以及更新过程映像区分区参数。参数化界面如图 9-22 所示。

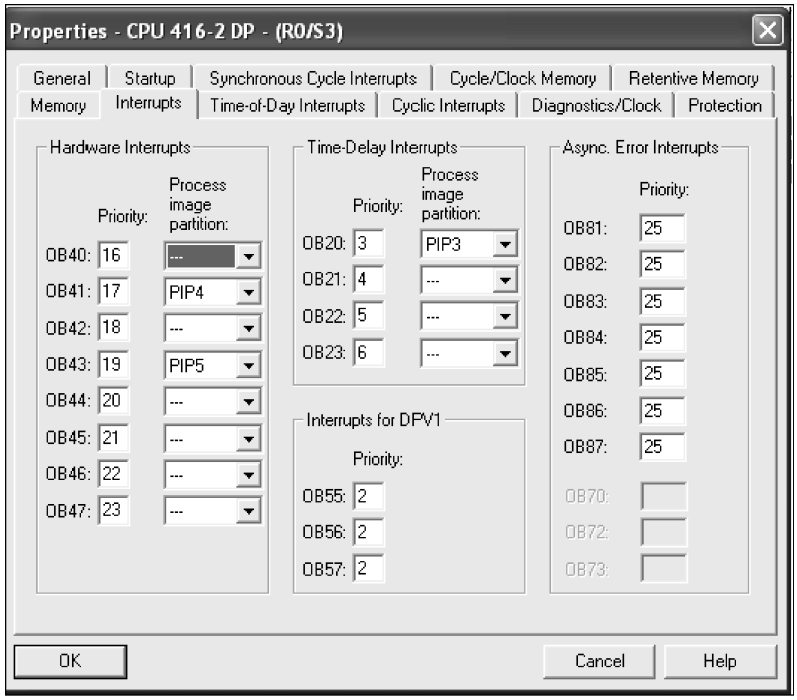



图 9-22 S7-400 系列 PLC CPU 中断界面

中断参数化界面主要参数及选项的功能描述如下：

“Hardware Interrupts”：硬件中断由实际的 I/O 信号及智能模块（如 FM）触发，OB40 ~ OB47 为硬件中断组织块，在 I/O 的配置中，可以选择某一个信号触发哪一个中断组织块，硬件中断在 S7-300 系列 PLC CPU 中只能触发 OB40，S7-400 系列 PLC CPU 可用 OB 与 CPU 的类型有关。

为每个中断分配不同的优先级，如果禁用某个中断块，将优先级选择 0，当中断事件同时出现，优先级高的 OB 先触发。S7-300 系列 PLC CPU 不能修改优先级。

在过程映像区分区参数中，可以选择更新设定的过程映像区分区，这样在 OB 调用时，更新过程映像区分区中的 I/O 信号，通常情况下，更新与 OB 调用相关的 I/O 信号，程序以当前的 I/O 状态执行操作。默认设置为未更新过程映像区分区。

“Time-Delay Interrupts”：当某一事件产生时，延时中断组织块 OB20 ~ OB23 经过设定的延时时间后被执行，同时更新设定的过程映像区分区，延时中断的触发条件由用户程序定义，并必须通过系统函数 SFC32 实现。使用 SFC32 触发 OB21 的示例程序如下：

```
A       M       1.1          //触发条件
FP      M       1.2          //取沿信号
JNB     M1                   //如果信号为 0，跳转到 M1 程序段
CALL    SFC     32           //调用 SFC32
OB_NR   :=21                 //触发 OB21
DTIME   :=T#2ms              //延时 2ms 触发 OB21 执行
SIGN    :=W#16#11            //用户自定义的标识符，可以在 OB21 中识别
RET_VAL :=MW20               触发的信号源
M1：NOP 0
```

与延时中断相关的系统函数还有 SFC33 和 SFC34，用于取消延时中断和查询延时中断的状态。可使用的延时中断 OB 的个数与 CPU 类型有关。

“Asynchronous Error Interrupts”：设定异步故障中断 OB（OB81 ~ OB87）的优先级，S7-300 系列 PLC 不能设置 OB 的优先级。

“Interrupts for DPV1”：设定与 PROFIBUS-DP V1 中断相关 OB（OB55 ~ OB57）的优先级。

### 9.5.8　日期中断

点击“Time-of-Day Interrupts”标签，进入日期中断参数化界面。在该界面中，可以设置日期中断的开始时间及执行方式，参数化界面如图 9-23 所示。

选中激活（Active）选项，可以激活相应的日期中断 OB，为 OB 选择开始触发的日期，日期以 CPU 的内部时钟信号为基准。在设定的日期触发相应的 OB，在后续的执行方式中，可以选择只执行一次（Once）或按特定的间隔如每分钟、每小时、每天、每周或每年执行。

通过调用系统函数 SFC28、SFC29、SFC30、SFC31，在程序中也可以用来设置、取消、激活和查询日期时间中断。手动与程序设定日期中断同时有效。可使用的日期中断 OB 的个数与 CPU 类型有关。

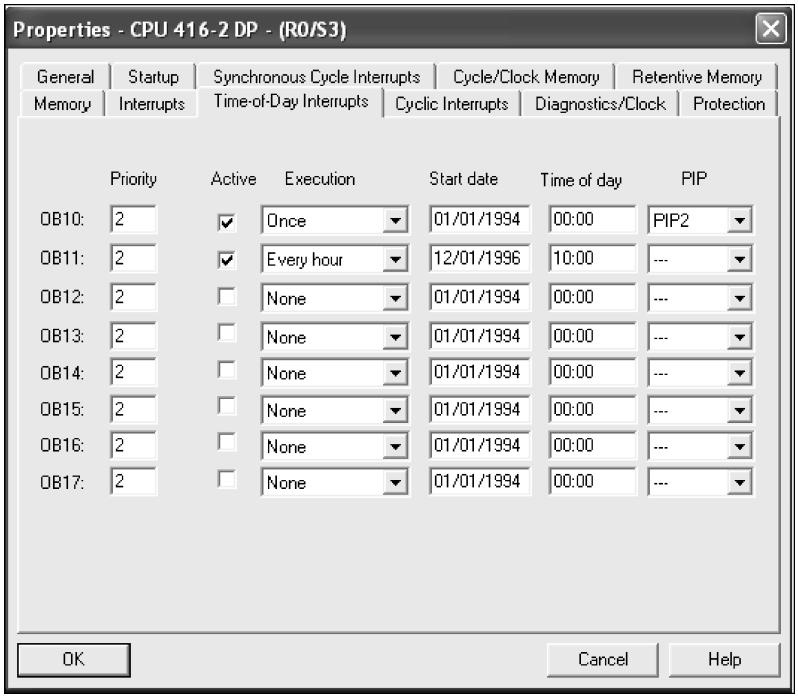



图 9-23 S7-400 CPU 日期中断界面

## 9.5.9 循环中断

点击“Cyclic Interrupts”标签，进入循环中断参数化界面，如图 9-24 所示。在该界面

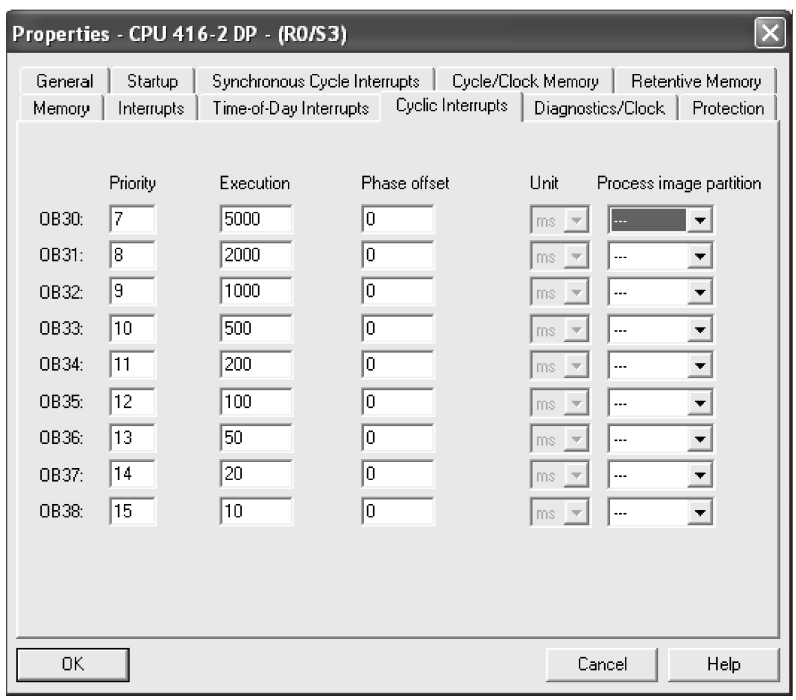



图 9-24 S7-400 CPU 循环中断界面

中，可以设置循环中断的优先级及间隔。循环中断用于在一个固定的时间间隔，执行循环中断组织块。时间间隔可以设置，其范围从 1 ~ 60000ms。设定 OB 的时间间隔必须大于中断程序的执行时间，否则会产生循环中断错误，并调用 OB80。

循环中断参数化界面主要参数及选项的功能描述如下。

“Execution”：OB 执行的时间间隔（ms）。

“Phase Offset”：循环中断执行的延时时间，如果同时使用多个循环中断 OB，在某一时刻，多个循环中断 OB 同时被调用，只能按照它们的优先级高低顺序执行，使用延时时间可以调整循环中断的执行时间。在 S7-300 系列 PLC CPU 中，通常只能使用 OB35，不会有多个循环中断 OB 冲突的情况。

### 9.5.10　诊断/时钟

点击“Diagnostics/Clock”标签，进入诊断/时钟参数化界面。在该界面中，可以设置系统诊断功能及时钟同步功能，如图 9-25 所示。

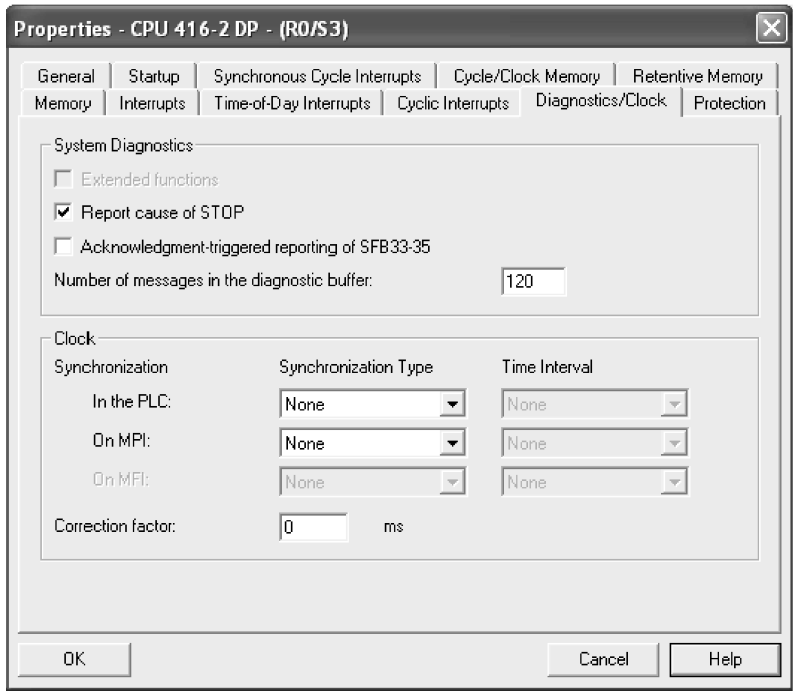



图 9-25　S7-400 CPU 诊断/时钟界面

诊断/时钟参数化界面主要参数及选项的功能描述如下。

“System Diagnostics”：系统诊断功能包括 3 项。

“Report cause of STOP”：如果激活该选项，CPU 将停机的原因上传到设定的 HMI 中，如编程器、OP 等，同时停机信息进入 CPU 的诊断缓存区。

“Acknowledgment-triggered reporting of SFB 33-35”：PCS7 应用时，可以使用 SFB33 ~ SFB35 产生文本消息，如果激活该选项，相对于 HMI 系统，在上次消息被确认后，如果有新的消息产生，SFB33 ~ SFB35 只传送触发的信号状态，这样可以阻止文本信息频繁的产生，也可以避免信息不能被执行。

Number of messages in the diagnostic buffer：设置 CPU 诊断缓存区中事件信息的条码数量，S7-300 系列 PLC CPU 中不能设置。

**注意：**

每条信息占用 CPU “Code memory” 20 或 32 字节（与 CPU 类型有关）的内存空间。

“Clock”：在时钟同步（Synchronization）选项中，可以设置 CPU 在 PLC 站内部或在 MPI 网络上作为时钟主站（Master），还是作为时钟从站（Slave）或者不使用时钟同步（None）功能。如果通过以太网同步其他站点的时钟信息，在主动方“In the PLC”中选择“Master”，并设置同步的间隔，在以太网模块中激活时钟同步选项；在其他被动方“In the PLC”中选择“Slave”，同样激活以太网模块中时钟同步选项，这样时钟信号自动通过主站按设定的间隔同步从站的时钟信号。

校正因子（Correction factor）（ms）用于校正系统时钟的误差。例如：时钟每 24h 快 3s，则应该在此处填入-3000。

### 9.5.11 程序保护

点击“Protection”标签，进入程序保护界面。在该界面中，可以设置不同的保护级别及口令，如图 9-26 所示。

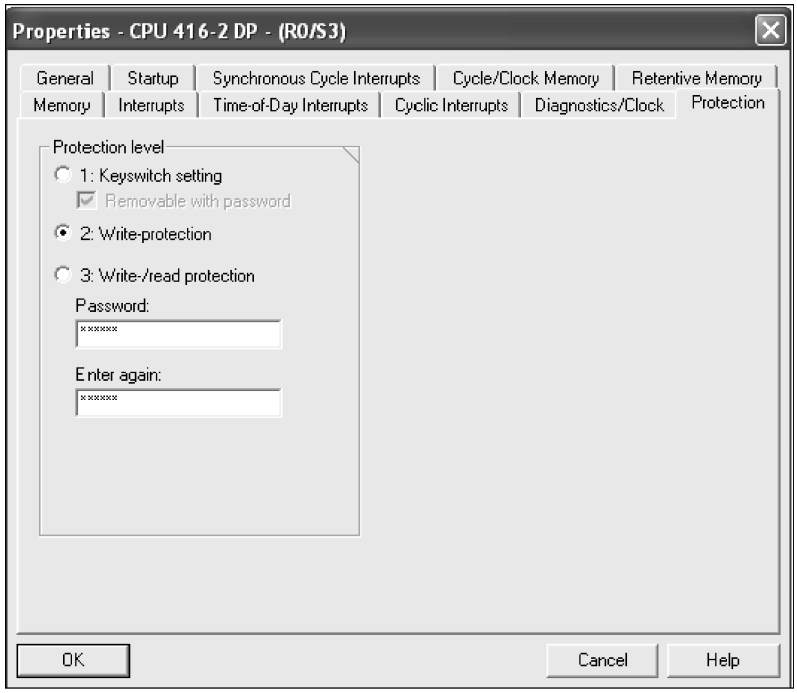



图 9-26 S7-400 CPU 程序保护界面

程序保护参数化界面主要参数及选项的功能描述如下。

“Protection level”：保护级别的设置。

第一级“Keyswitch setting”：默认级别。

根据 CPU 模块上的模式选择开关决定保护方式，在 RUN-P 或者 STOP 模式下没有限制，

在 RUN 模式下只能读而不能写，如果选择“Removable with password”，在 RUN 模式下可以进行写操作，但是需要相应口令保护。使用 MMC 的 S7-300 系列 PLC CPU 只有 RUN-STOP 开关，不能在这里设置口令保护。

第二级“Write-protection”：写保护。

不管模式选择开关在什么位置，只能读。写操作需要设置口令。

第三级“Write-/read protection”：读/写保护。

不管模式选择开关在什么位置，都禁止读写操作。写操作需要输入口令。

“Mode”（不适用 S7-400 系列 PLC CPU）。

“Process Mode”：通常设备在运行阶段时采用这种模式。在该模式下，保证 CPU 的扫描时间，系统的测试功能受到限制。也可以手动设置允许测试功能（例如监视或修改变量）占用 CPU 最长的扫描时间。断点测试和单步执行（CPU 处于 HOLD 模式）都不能实现。

“Test Mode”：设备处于调试阶段时，可以选择这种模式。在该模式下，所有测试功能都不受限制，但是测试会带来 CPU 扫描时间的增加。

### 9.5.12　分配通信资源（不适用 S7-400 系列 PLC CPU）

点击“Communication”标签，进入通信参数化界面。在该界面中，可以设置不同通信方式，占用 CPU 的通信资源，如图 9-27 所示。

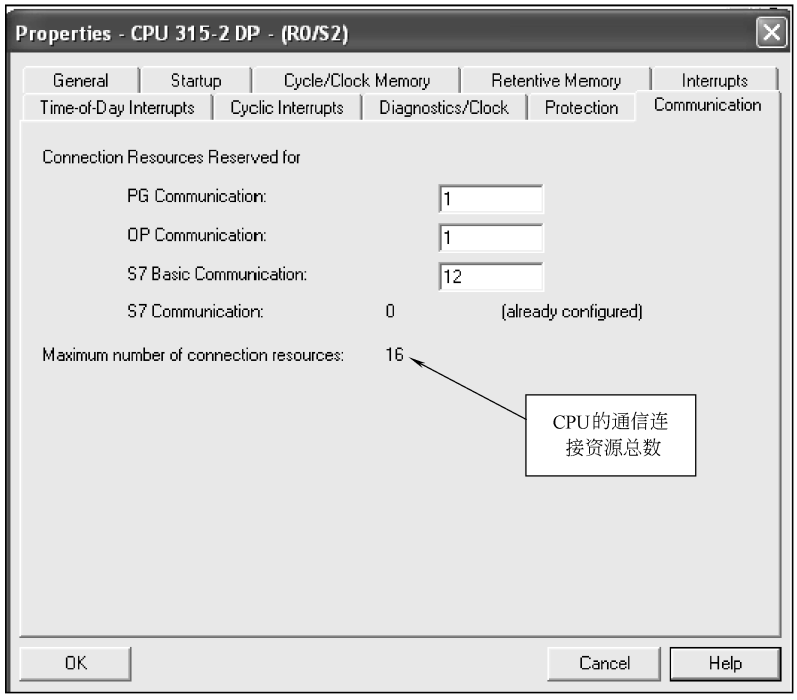



图 9-27　S7-300 系列 PLC CPU 通信资源参数化界面

带有 MMC 的 S7-300 系列 PLC CPU 在属性界面中，可以分配 CPU 的通信连接资源，“Maximum number of connection resources”显示该 CPU 可提供的最大通信连接资源数（总的通信资源数量与 CPU 类型有关），编程器（PG）通信、操作面板（OP）通信、S7 基本通信

（S7Basic Communication）及 S7 通信（S7 Communication）的连接都需要占用 CPU 的通信连接资源，在参数化界面中，可以为这些通信连接预留一定的连接资源。

**注意：**

PG 至少占用一个连接资源。如果为某一个通信方式预留较多的通信资源，则不能在其他的通信方式中建立更多的连接，如果在参数化界面中为每种通信方式预留较少的通信资源，当其中一种通信方式超过预留的通信连接后，将自动占用 CPU 剩余的连接资源，通信资源由 CPU 自动分配，但是连接总数不能超过 CPU 的连接资源。

## 9.6 I/O 模块参数配置

### 9.6.1 数字量 I/O 模块参数配置

1. 更改模块逻辑地址

在机架上插入数据量 I/O 模块时，系统自动为每个模块分配逻辑地址，删除或添加模块，不会导致逻辑地址冲突。有些应用中，用户预先编写程序，在现场进行硬件配置，可能需要调整 I/O 模块的逻辑地址，以匹配控制程序。除一些旧版本的 S7-300 系列 PLC 不能修改数字量 I/O 逻辑地址外，新型号的 CPU 都可以自由修改数字量 I/O 的逻辑地址。如果需要更改模块的逻辑地址，可以双击该模块，在弹出的属性对话框中，选择“Addresses”标签页，如图 9-28 所示。

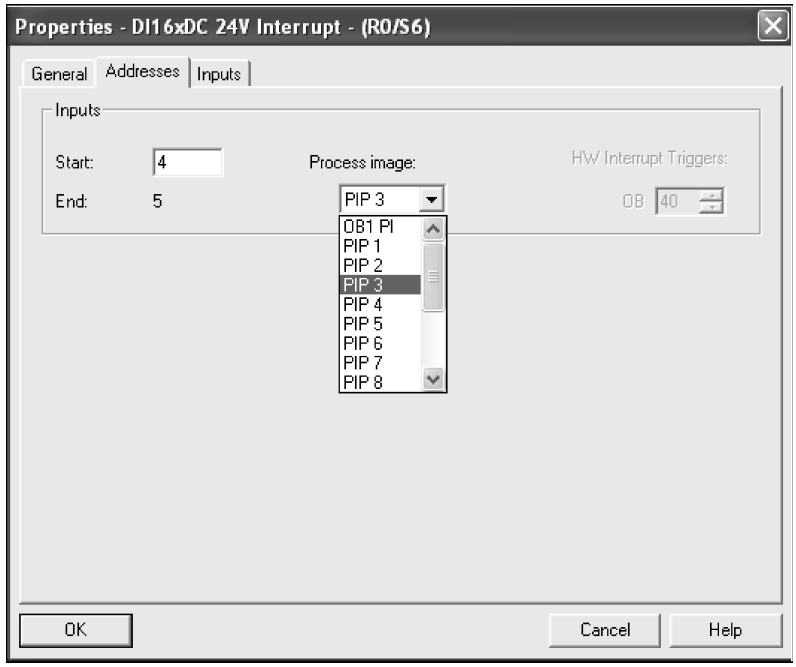



图 9-28 数字量输入模块的地址界面

在“Start”框中，输入新的起始地址，修改后，系统根据模块的 I/O 数量自动计算结束地址。在 S7-300 系列 PLC 中，需要去掉“System selection”选择，才能进行模块开始地址。

如果修改的模块与其他模块地址相冲突，系统自动提示地址冲突信息，修改不能被确认。

在 S7-400 系列 PLC 系统中，如果 I/O 模块的地址在过程映像区内，可以选择模块更新的过程映像区分区，例如选择模块在 PIP2（过程映像区分区 2）中更新，在 OB35 配置中，选择更新 PIP2，该模块只能在 OB35 调用时被更新，将总的过程映像区分区，减少过程映像区更新的时间。

2. 参数化数字量输入模块

高特性的输入模块带有中断和诊断功能，使用这些功能必须进行配置，双击该模块，在弹出的属性对话框中选择“Inputs”标签页，如图 9-29 所示。

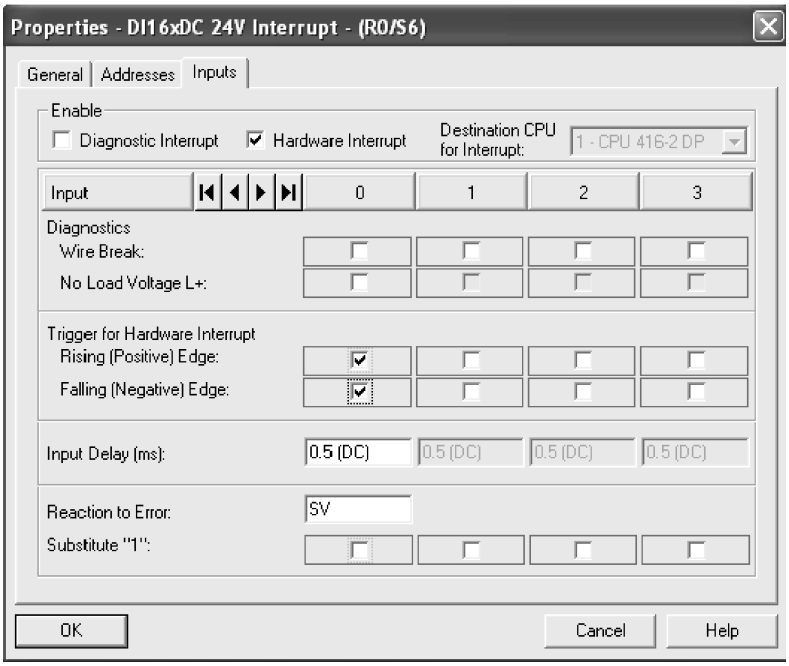



图 9-29　数字量输入模块参数化界面

在输入参数化界面中，主要参数及选项的功能描述如下：

“Enable”：如果使能选项“Diagnostic Interrupt”，在“Diagnostics”栏中，选择触发诊断中断的故障类型，出现监控的故障时，产生诊断中断，并由 CPU 调用 OB82；如果使能选项“Hardware Interrupt”，在“Trigger for Hardware Interrupt”栏中，选择触发硬件中断的事件，使能硬件中断功能后，还需在模块“Address”标签页选择该模块触发硬件中断的 OB，例如 OB40，每次出现硬件中断事件时，触发 CPU 调用 OB40 一次。

“Input”：通过方向箭头，选择模块输入通道。

“Diagnostics”：选择是否激活断线（（Wire Break）参考接线方式）和丢失负载电压（Missing Load Voltage L+）诊断功能。故障信息可以通过 SFC51、SFC59 读出，SFC 的调用参考 7.6 节。如果使能诊断中断，出现监控的故障类型时，触发 CPU 调用 OB82。

“Trigger for Hardware Interrupt”：选择触发硬件中断的信号源，如第几个输入通道，上升沿、下降沿或上升沿、下降沿同时产生硬件中断。

“Input Delay”：选择每个输入通道的输入延时时间，输入延时越长，信号越不容易受到

干扰，但是影响响应速度。

“Reaction to Error”：如果诊断事件出现，模块将按下列设定提供输入信号：

“SV”：CPU 读入的信号选择替代值，如果选择 substitute “1”，替代值为 1，如果不选择，替代值为 0。

“KLV”：CPU 读入信号，选择上次有效的值。

模块故障消除，恢复原先读取的方式（过程映像区输入）。

3. 参数化数字量输出模块

有些数字量输出模块带有诊断功能，可以进行参数化，例如输出模块 6ES7322-8BF00-0AB0 参数化界面，如图 9-30 所示。

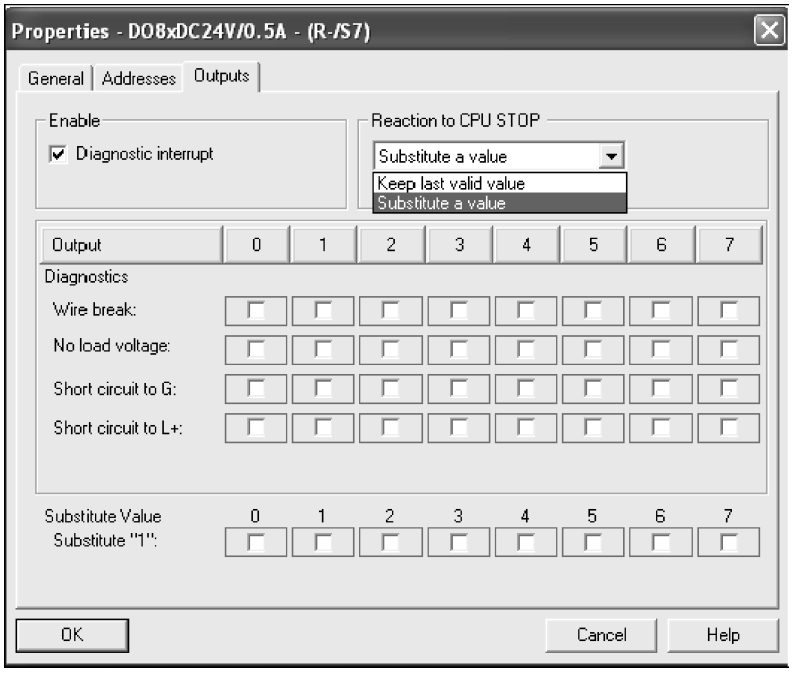



图 9-30 数字量输出模块参数化界面

在输出参数化界面中，主要参数及选项的功能描述如下：

“Enable”：如果使能选选项 “Diagnostic Interrupt”，在 “Diagnostics” 栏中，选择触发诊断中断的故障类型，出现监控的故障时，产生诊断中断，并由 CPU 调用 OB82。

“Output”：通过箭头方向选择模块输出通道。

“Diagnostics”：选择是否激活断线（（Wire break）参考接线方式）、丢失负载电压（No load voltage）、对地短路（Short circuit to Ground）和与 L + 端短路（Short circuit to L + :）诊断功能。故障信息可以通过 SFC51、SFC59 读出，SFC 的调用参考 7.6 节。如果使能诊断中断，出现监控的故障时，触发 CPU 调用 OB82。

“Reaction to CPU STOP”：如果诊断事件出现，模块按下列设定输出：

“KLV”：CPU 停止，模块输出保持上次有效值。

“SV”：CPU 停止，模块输出使用替代值，在“Substitute Value”设置替代值。

## 9.6.2　模拟量模块参数配置

1. 参数化模拟量输入模块

通常一个模拟量输入模块可以连接多种传感器，在模块上需要跳线以匹配不同的传感器，同样在硬件配置中也必须进行参数化，并与实际安装类型及模块设置相匹配。以经常使用的模拟量输入模块 6ES7 331-7KF02-0AB0 为例，介绍模块的参数化。

在“Addresses”标签栏中，设置模块的开始地址，模拟量值不需要过程映像区处理，通常开始地址位于过程映像区外。选择“Inputs”标签栏，进入参数化界面，如图 9-31 所示。

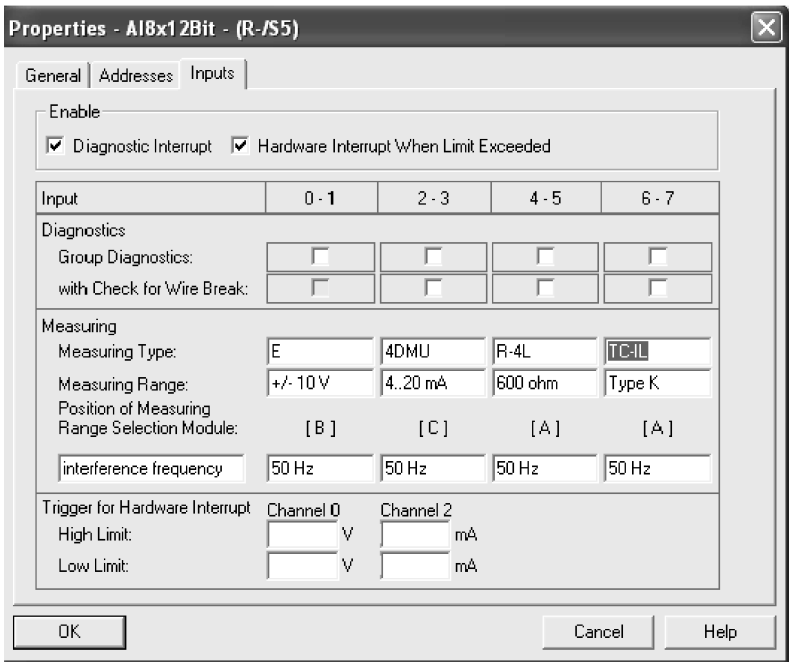



图 9-31　模拟量输入模块参数化界面

在模拟量输入参数化界面中，主要参数及选项的功能描述如下：

“Enable”：如果使能选项“Diagnostic Interrupt”，在“Diagnostics”栏中，选择触发诊断中断的故障类型，出现监控的故障时，产生诊断中断，并由 CPU 调用 OB82；如果使能选项“Hardware Interrupt”，在“Trigger for Hardware Interrupt”栏中，选择触发硬件中断的事件，使能硬件中断功能后，还需在模块“Addresses”标签页中，选择该模块触发硬件中断的 OB，例如 OB40，每次出现硬件中断事件时，触发 CPU 调用 OB40 一次。

“Input”：指示选择模块输入通道。

“Diagnostics”：选择是否激活断线（Wire Break）和丢失负载电压（Missing Load Voltage L+）诊断功能。故障信息可以通过 SFC51、SFC59 读出，SFC 的调用参考 7.6 节。如果使能诊断中断，出现监控的故障时，触发 CPU 调用 OB82。

“Measurement”：

“Measuring Type”：选择连接传感器的类型，如电压、电流、电阻等信号。

“Measuring Range”：选择测量范围，如选择连接电压类型传感器，测量范围可选择 +/-10V、1～5V 等信号。

“Position of the Measuring”：与量程卡跳线位置相匹配。

“Integration time / Interference frequency”：选择输入信号的积分时间，积分时间长，模拟量转换时间长，分辨率高。

“Trigger for Hardware Interrupt”：设置输入信号的上限和下限，当超过这个范围时，会产生一个硬件中断，并由 CPU 调用 OB40。只有通道 0 和通道 2 可以选择。

2. 参数化模拟量输出模块

模拟量输出模块只能连接电压和电流负载，参数化比较简单，如图 9-32 所示。

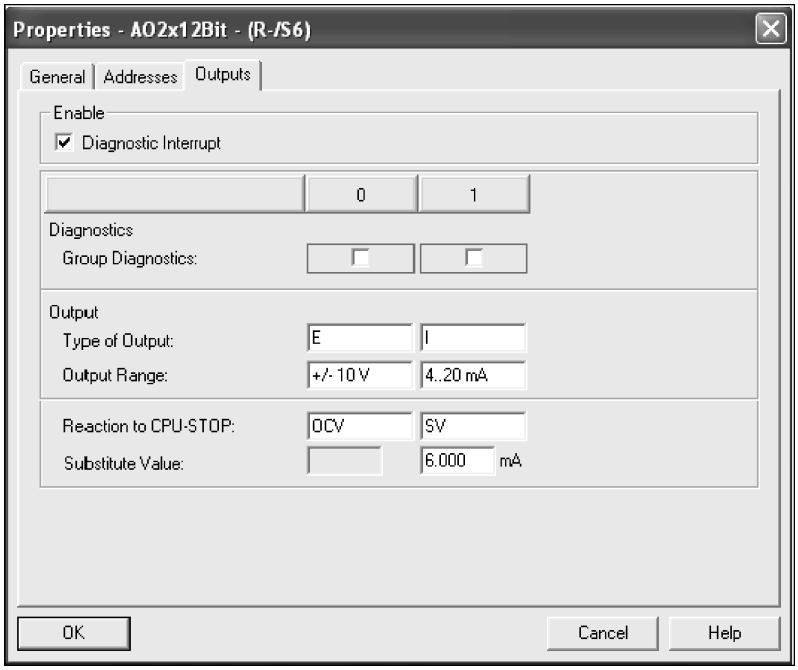



图 9-32 模拟量输出模块参数化界面

在模拟量输出参数化界面中，主要参数及选项的功能描述如下：

“Enable”：选择是否使能诊断中断，模块故障时，触发 CPU 调用 OB82；

“Diagnostics”：选择是否使能组诊断，如果选择组诊断，出现下列故障，例如断线（电流输出）、与地短路（电压输出）及丢失负载电压等，通过 SFC51、SFC59 可以读出，SFC 的调用参考 7.6 节。

“Output”

“Type of Output”：选择输出类型，例如选择电压输出或电流输出；

“Output Range”：选择输出范围，例如电压输出类型，可以选择 +/-10V、0～10V 等输出范围。

“Reaction to CPU-STOP”

“OCV”：-CPU 停止，模块不输出；

“KLV”：-CPU 停止，模块输出保持上次有效值；

“SV”：-CPU 停止，模块输出使用替代值，在“Substitute Value”设置替代值。

## 9.7　更新硬件条目

由于硬件版本的不断更新，使用旧版本 STEP7 进行硬件配置时，可能在硬件目录中不能发现合适的条目与实际硬件相匹配，而不能进行硬件配置。STEP7 V5.2 以上的版本支持硬件目录更新的功能，有两种方法可以更新硬件目录：

1. 在线更新

将装有 STEP7 软件的编程器连接到 internet 上，在硬件配置界面中，使用菜单命令“Options”→“Install HW Updates”，弹出对话框，选择“Download From Internet”，点击“Execute”键，将当前最新的硬件目录下载到编程器硬盘中，如图 9-33 所示。

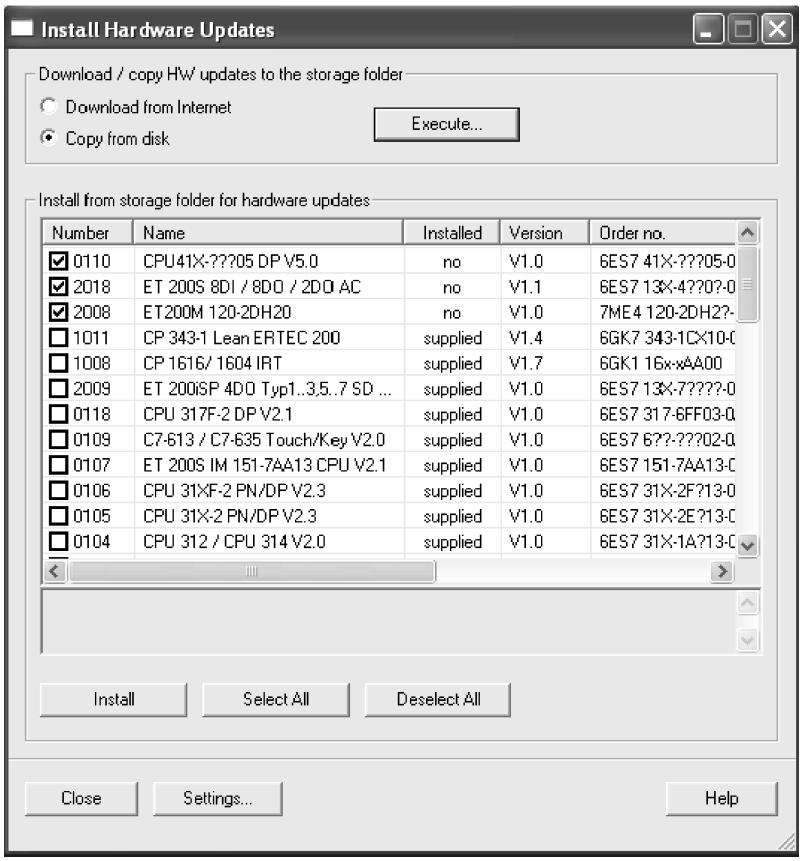



图 9-33　下载更新的硬件目录

在“Installed”一列中，如果显示“no”，则表示该硬件尚未安装，如果显示为“supplied”，则表示当前的 STEP7 中已经包含了该硬件，无需再更新。选中需要更新的硬件，点击“Install”按钮，按照提示即可完成硬件更新。

2. 离线更新

从 Internet 下载的文件，在默认的情况下存储于“C：HWUpdates”文件夹中，从更新过的硬件目录编程器中复制 HWUpdates 文件夹，并存储于本地编程器 C：HWUpdates”文件夹中或通过“Settings. .”按钮，选择任意存储路径，点击“Install”按钮，更新硬件目录。

**注意：**

不是所有的硬件都可以更新，在这种情况下必须升级 STEP7 版本。

## 9.8 远程 I/O 扩展

远程 I/O 扩展可以通过现场总线，将现场 I/O 站以通信的方式连接到主 CPU 上，与中央扩展方式相比较，远程扩展在布线、安装、I/O 类型的选择、防护等级、调试等方面都优于中央扩展，中央扩展方式将逐渐被淘汰。

远程 I/O 扩展通常在中央机架上安装电源、CPU、通信处理器 CP 模块，根据需要在不同的位置上安装 I/O 站点，通过带有通信接口（PROFIBUS-DP 或 PROFINET IO）的 CPU 连接远程 I/O 站点，通信处理器 CP 可以与其他主站或上位机进行通信。

通过 PROFIBUS-DP 可以连接 ET200I/O 系列，例如 ET200S、ET200iS、ET200L、ET200B、ET200M、ET200X、ET200ECO、ET200R、ET200Pro 等适用于各种不同应用场合的远程站，也可以连接智能从站，如 S7-200 系列 PLC、S7-300 系列 PLC、S7-400 系列 PLC 站点、第三方设备、驱动设备等；通过实时以太网 PROFINET IO 可以连接 ET200S、ET200 Pro、西门子公司的交换机、网关等设备。

### 9.8.1 配置 PROFIBUS-DP 远程 I/O 站

S7-300 系列 PLC 中 CPU31X-2DP 和 CP342-5 带有 PROFIBUS-DP 接口，可以连接远程 I/O站点；S7-400 系列 PLC 中所有 CPU 和 CP443-5、IM467 都带有 PROFIBUS-DP 接口，可以连接远程 I/O 站点。

在 STEP7 硬件配置中，添加一个 DP 主站（带有集成 DP 接口的 CPU，或者 Profibus 通信处理器 CP），弹出新建 PROFIBUS 网络的提示对话框，选择新建一条 PROFIBUS 网络并设置主站站号、通信速率及 PROFIBUS 参数行规（选择 DP）后出现一条代表 PROFIBUS-DP 总线的网络，如图 9-34 所示。

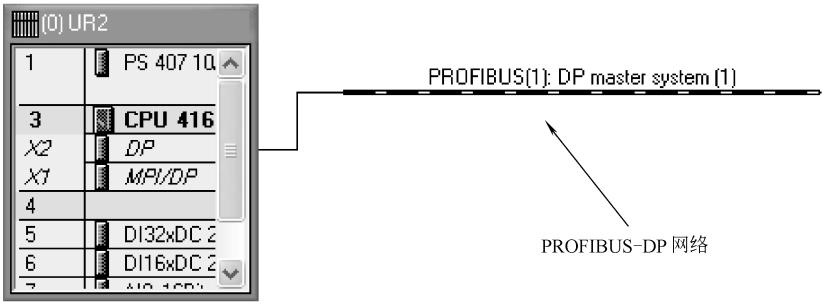



图 9-34 设置 PROFIBUS 网络

在硬件目录中的 PROFIBUS-DP 子目录下列出了各种 DP 从站设备。将所需的设备拖放

到总线上，例如配置 ET200M、ET200S 及驱动设备 MM440，如图 9-35 所示。

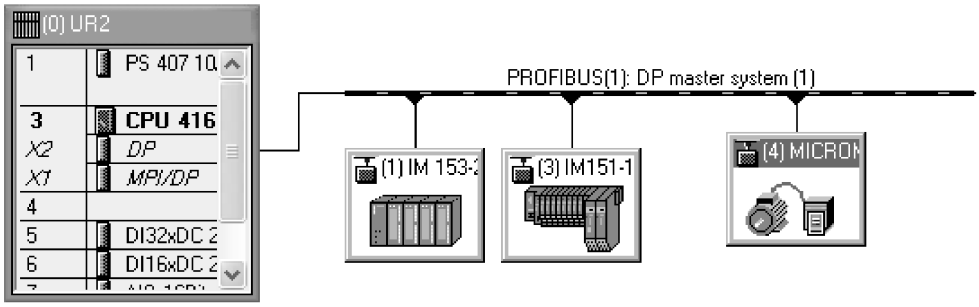



图 9-35　配置 PROFIBUS-DP 从站

有的从站图标带有从站类型标识、PROFIBUS-DP 从站接口模块类型及站号，配置的从站站号必须与现场实际设定的站号相同。点击每个从站，为从站配置相应的 I/O 模块或与主站的通信接口区，通过 CPU 集成的 DP 接口、S7-400 系列 PLC 通信处理器、IM467 连接远程扩展，模块的 I/O 地址占用 I、Q 区，与中央扩展寻址方式相同，如果使用 S7-300 系列 PLC CP342-5 作为主站，则需要调用通信函数 FC1、FC2 与从站建立通信接口区。

**注意：**

PROFIBUS 的通信速率与总线的长度有关，详细信息请参考第 10 章网络通信。

## 9.8.2　配置 PROFINET IO 远程 I/O 站

S7-300 系列 PLC 系列中 CPU31X-2DP/PN、通信处理器 CP343-1PN 及 S7-400 系列 PLC 系列中 CPU41X-2DP/PN、通信处理器 CP443-1ADV 带有 PROFINET IO 接口，可以连接带有 PROFINET IO 接口的远程 I/O 站点。

在 STEP7 硬件配置中，以 CPU315-2DP/PN 为例创建一个站点，插入 CPU 时，弹出新建以太网的提示对话框，选择新建以太网网络并设置 CPU 的 IP 地址（CPU 的前面板标有 MAC 地址，在 SIMATIC Manager 界面下，点击菜单命令“PLC”→“Edit Ethernet Nodes”可以分配 IP 地址或通过 MPI 接口下载配置信息设置 IP 地址）、子网掩码（255.255.255.0）后出现一条代表 PROFINET IO 总线的网络，如图 9-36 所示。

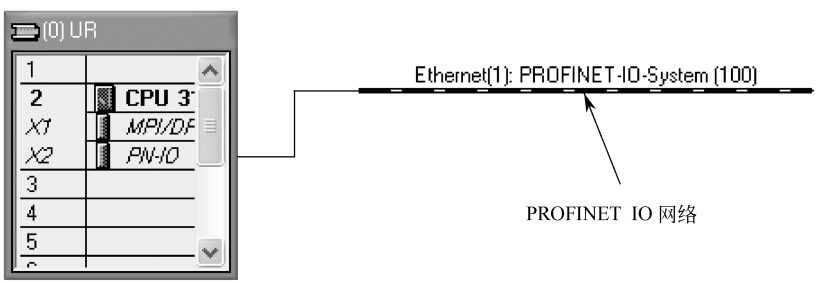



图 9-36　设置 PROFINET IO 网络

在硬件目录中的 PROFINET IO 子目录下，将所需的设备拖放到总线上，例如配置 ET200S、交换机 X208 及网关设备如图 9-37 所示，使用 IE/PB 网关（以太网/PROFIBUS-DP）可以从实时以太网 PROFINET IO 上直接连接 PROFIBUS-DP 从站。

点击每个从站，为从站配置相应的 I/O 模块，模块的 I/O 地址占用 I、Q 区，与中央扩展寻址方式相同。

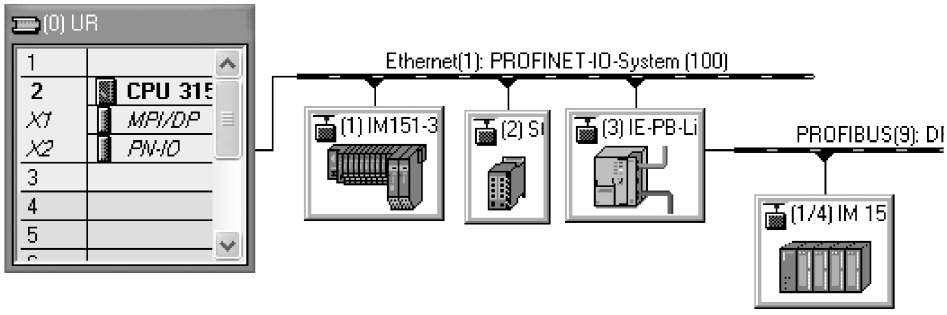



图 9-37　配置 PROFINET IO 远程 I/O

双击远程 I/O 站图标，为每个站点分配地址信息，如图 9-38 所示。在“Device Name”下输入元件名称，每个远程站点的元件名称是惟一的，PROFINET IO 控制器通过元件名称自动分配远程站点 IP 地址；在“Device Number”中定义元件号，每个元件号是惟一的，通过元件号可以对站点进行诊断；点击“Ethernet”为远程 I/O 站分配 IP 地址，IP 地址在每个网络上是惟一的。地址信息分配完成后编译存盘。

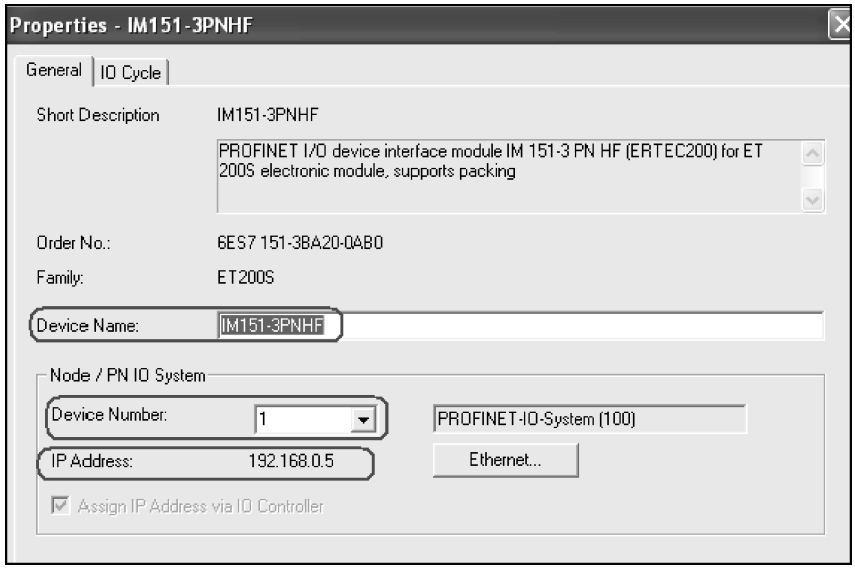



图 9-38　远程 I/O 的配置

将编程器通过以太网网卡连接到实际的 PROFINET IO 网络上，在硬件配置界面上使用菜单命令“PLC”→“Ethernet”→“Assign Device Name”以每个远程 I/O 站的 MAC 地址为

参考分配元件名称。通过菜单命令“PLC”→“Ethernet”→“Verify Device Name”检查元件名称是否分配成功。将配置信息下载到 CPU 中，CPU 在启动过程中自动为每个远程 I/O 站分配 IP 地址。诊断功能通过 IP 地址进行访问，实时通信通过网络的第一、二层实现。

在“IO Cycle”标签中，设置远程 I/O 站数据的更新时间，更新时间与远程 I/O 站点的个数、通信数据量的大小有关，例如 CPU（IO 控制器）连接远程 I/O 站点的数量在 13 个以内可以设置的更新时间为 1ms，使用无线网络、多个交换机级联的情况下应增加更新时间。

### 9.8.3　远程 I/O 站点的诊断

一条网络上连接多个远程 I/O 站点，单一 I/O 站点故障不会影响到其他站点的正常工作。每个远程 I/O 站点的故障信息可以通过编程方式读到 CPU 中，参与逻辑控制和上位机故障归档及显示。

1. PROFIBUS-DP 网络远程 I/O 站点的诊断

- 通过 OB86 的开始信息，读出站点的诊断信息（通过 PROFIBUS 地址），示例程序可以参考 7.2.2 节。
- 通过调用 FC125 或 FB125 读写远程 I/O 站点的信息，FC125 和 FB125 可以从西门子网站上下载。FC125 为 FB125 的简化版本，可以读出故障的站点和丢失的站点；FB125 功能完全覆盖 FC125，可以实现对 I/O 模块每个通道进行诊断、优化诊断的评估方式（可以对故障信息逐一进行评估）、对 HART 通道进行诊断等功能，FB125 功能强大但是参数较多，这里不再介绍。如果只是对从站的站点进行诊断，可以使用 FC125 进行诊断，调用 FC125 的示例程序如下（使用符号地址赋值）：

```
CALL "GLOBAL_DP_DIAG"
CHECK_ACTIV                 : ="CHECK_ACTIV"
EXTERNAL_DP_INTERFACE       : =FALSE
DP_MASTERSYSTEM             : =1
DATA_FIELD                  : ="GLOBAL_DIAG_DB".DATA_FIELD
SUM_SLAVES_DIAG             : ="GLOBAL_DIAG_DB".SUM_SLAVES_DIAG
LIST_SLAVES_NOT_PRESENT     : ="GLOBAL_DIAG_DB".LIST_SLAVES_NOT_
                                PRESENT
LIST_SLAVES_ERROR           : ="GLOBAL_DIAG_DB".LIST_SLAVES_ERROR
RETVAL                      : ="GLOBAL_DIAG_DB".RETVAL
BUSY                        : ="GLOBAL_DIAG_DB".BUSY
```

FC125 参数含义如下：

CHECK_ACTIVE：函数输入，数据类型为 BOOL，为 1 时开始检测从站

EXTERNAL_DP_INTERFACE：函数输入，数据类型为 BOOL，选择 DP 主站的接口，0 表示集成在 CPU 上的 DP 主站，1 表示外部 DP 主站接口，如 CP443-5、IM467 等，不能使用 CP342-5 作为主站

| | |
|---|---|
| DP_MASTERSYSTEM | ：函数输入，数据类型为 INT，PROFIBUS 的 ID 号，在硬件配置中可以查看 PROFIBUS 的 ID 号 |
| DATA_FIELD | ：函数输入，数据类型为 POINTER 指针。50 字节的数据区，仅供 FC125 内部使用，例如 P#DB100. DBX0. 0 表示从 DB100. DBB0 开始的 50 字节 |
| SUM_SLAVES_DIAG | ：函数输出，数据类型为 INT。有故障从站的总数 |
| LIST_SLAVES_NOT_PRESENT | ：函数输出，数据类型为 POINTER 指针。检查到丢失的从站（软件已经配置，但是现场没有连接硬件或从站断电）输出 16 字节的数据区，每一个数据位对应一个从站，置 1 表示一个从站丢失，如果丢失的从站返回或维修好，相对应的位复位 |
| LIST_SLAVES_ERROR | ：函数输出，数据类型为 POINTER 指针。故障的从站（部分模块有故障，其他模块可以连续运行），输出 16 字节的数据区，每一个数据位对应一个从站，置 1 表示一个从站故障，如果从站故障消除，相对应的位复位 |
| RETVAL | ：函数输出，数据类型为 INT。调用 FC125 时的状态返回值 |
| BUSY | ：函数输出，数据类型为 BOOL，为 1 表示当前正在执行 |

从站站号与输出数据区的对应关系，如图 9-39 所示。

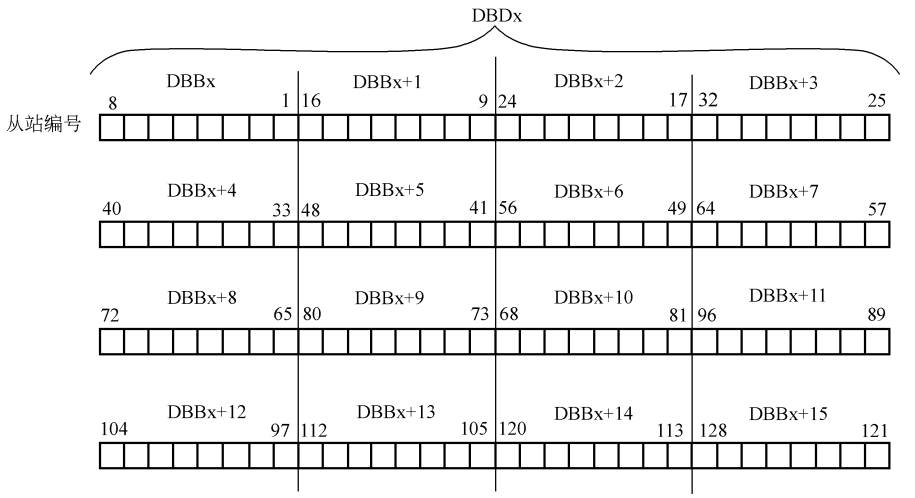



图 9-39　PROFIBUS-DP 从站站号与输出数据区对应关系

FC125 和 FB125 函数存储于光盘“网上下载”文件夹的“FC_FB125”子文件夹中，使用 STEP7 菜单“File”→“Retrieve”将文件“DP_DIA_49. zip”解压缩并复制函数 FC125

到应用项目中即可调用。

**注意：**

使用 S7-300 系列 PLC CP342-5，作为 PROFIBUS-DP 主站不能使用 FC125 和 FB125。

2. PROFINET IO 网络远程 I/O 站点的诊断

通过 OB86 的开始信息，读出站点的诊断信息（通过 PROFINET “Device Number”），示例程序参考 7.2.2 节，与 PROFIBUS 诊断编程相同，不能使用 FC125 和 FB125 对 PROFINET IO 站点进行诊断。

**注意：**

通过添加 GSD 文件，可以在 STEP7 中安装西门子公司或第三方 PROFIBUS-DP 从站和 PROFINET IO 远程站点的设备。GSD 文件描述远程站点设备的通信属性，可以由设备方提供或从相关网站下载，在 STEP7 硬件配置界面中使用菜单命令 “Options” → “Install GSD File” 命令安装 GSD 文件，安装完成后，在硬件目录 “PROFIBUS-DP” → “Additional Field Devices”（PROFIBUS-DP 从站）或 “PROFINET IO” → “Additional Field Devices”（PROFINET IO 设备）下可以发现安装的现场设备，这些现场设备可以作为分布式 I/O 站点与 CPU 进行通信。

## 9.9　符号地址寻址

在 STEP7 程序中，I/O 信号、位存储器、计数器、定时器、数据块和函数块可以使用绝对地址也可以使用符号地址寻址。使用符号地址寻址，增强程序的可读性，地址的符号名分为共享和区域两种。

1. 共享符号名

共享符号名可以在整个项目不同的程序块中使用，共享符号名需要在符号表中定义。在 “SIMATIC Manager” 界面中，点击 PLC 站→CPU→S7 Program，双击 “Symbols” 图标，进入符号表编辑界面，为每个绝对地址编辑所需的符号名，如图 9-40 所示。

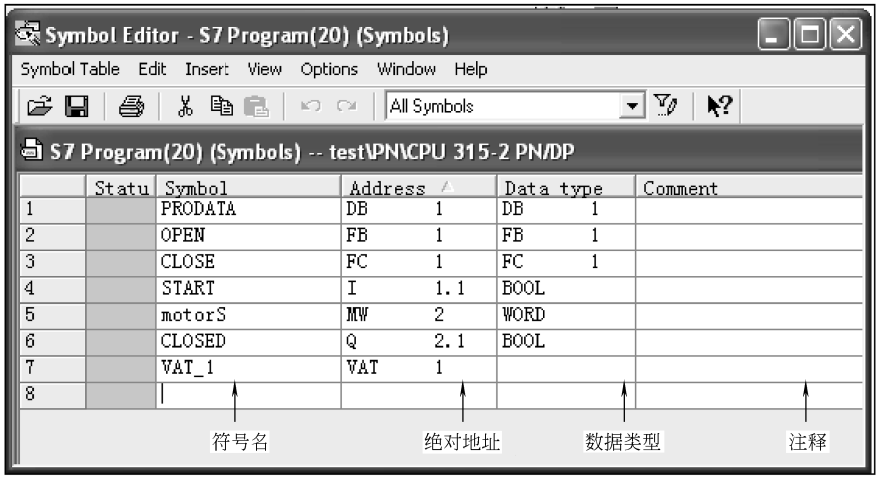



图 9-40　编辑符号表

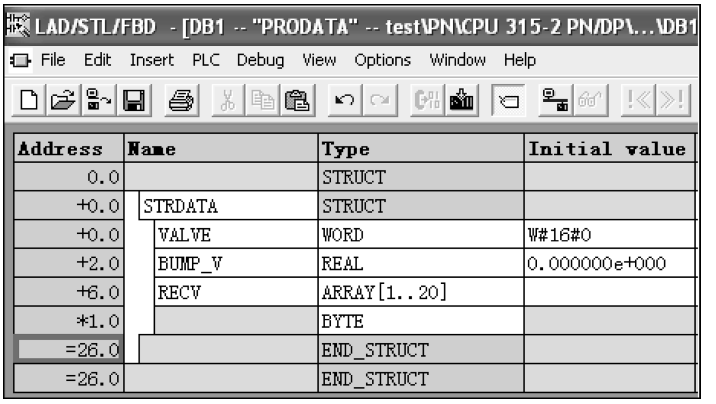
符号表的每一行中，在“Symbol”列中定义符号名称，在“Address”列中指定相应的绝对地址，在“Data type”列中自动显示绝对地址的数据类型。如在“Symbol”列和“Address”列中输入的地址与符号名相同，以红色高亮标出相同的行号。

在符号表中可以为下列对象定义符号名：

- I/O 信号（I、IB、IW、ID、Q、QB、QW、QD）
- 外设输入与输出（PI、PQ）
- 位存储器（M、MB、MW、MD）
- 定时器（T）/计数器（C）
- 逻辑块（FB、FC、SFB、SFC）
- 数据块（DB）
- 用户定义数据类型（UDT）
- 变量表（VAT）

在程序编辑界面中，使用菜单命令“View”→“Display with”→“Symbolic Representation”可以切换符号地址与绝对地址的显示。

在程序中访问数组、结构体、时钟等复合数据类型变量时，必须使用符号名寻址，这些数据必须在数据块（共享符号名）或区域数据区 L 中定义，例如在 DB1（图 9-40 中定义的符号名为 PRODATA）定义一个结构体类型变量，符号名“STRDATA”如图 9-41 所示。




| Address | Name | Type | Initial value |
|---|---|---|---|
| 0.0 | | STRUCT | |
| +0.0 | STRDATA | STRUCT | |
| +0.0 | VALVE | WORD | W#16#0 |
| +2.0 | BUMP_V | REAL | 0.000000e+000 |
| +6.0 | RECV | ARRAY[1..20] | |
| *1.0 | | BYTE | |
| =26.0 | | END_STRUCT | |
| =26.0 | | END_STRUCT | |

图 9-41　复合数据类型的符号地址

只能使用符号名“PRODATA. STRDATA”或“DB1. STRDATA”表示数据块中的结构体（变量大于四个字节不能使用绝对地址寻址），变量的从属关系用符号“.”隔开，例如“PRODATA. STRDATA. VALVE”寻址结构体中的一个变量。

2. 区域符号名

区域符号名只能在一个程序块中使用，可以在下列数据区中定义区域符号名：

- 函数或函数块参数（输入、输出及输入/输出参数）；
- 函数块的静态数据；
- 函数、函数块及 OB 的临时数据区。

使用区域符号名寻址与共享符号名寻址方式相同，例如在 FC1 中使用区域符号名的编程，示例如图 9-42 所示。

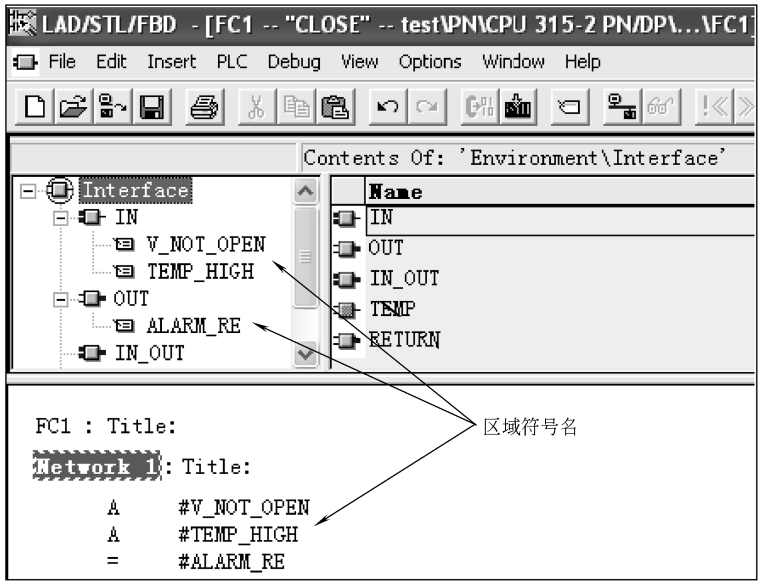



图 9-42　在程序块中使用区域符号名编程

使用区域符号名访问数组、结构体等复合数据类型与共享符号名的表示方式相同，变量的从属关系也是使用符号“.”隔开。

3. 区分共享符号名与区域符号名

如果共享符号名与区域符号名相同，在程序块中不会产生地址冲突，共享符号名被“”符号括起来，区域符号名前自动添加“#”符号，这样可以区分两者之间的符号名。

## 9.10　生成用户程序

一个 PLC 站的所有的程序块存储于“S7 Program”目录下的“Blocks”文件夹中，在“Blocks”文件夹中包括系统数据（System data）、逻辑程序块（OB、FB、FC、DB、UDT）和变量监控表（VAT）。

### 9.10.1　生成系统数据

系统数据包括系统数据块（SDB），这些系统数据块由硬件配置编译而成，如图 9-43 所示，不同序号的 SDB 表示不同类型的硬件配置和应用，例如 SDB1 由硬件配置编译或由 CPU 重新启动后生成，SDB5 表示由 CPU MPI 参数配置编译生成。硬件配置每次新的编译都将覆盖原有的编译信息，SDB 的生成时间表示硬件配置最后的编译时间。系统数据将占用 CPU 的装载存储器空间，这样 CPU 在运行时可以通过系统数据识别外围设备是否与实际配置的数据相匹配。

**注意：**

有些硬件配置编译后生成错误的系统数据，下载后 CPU 不能运行，将在线（CPU 中）和离线（编程器中）的系统数据删除后，重新对硬件配置进行编译，将新生成的系统数据下载到 CPU 中可以解决上述的问题。

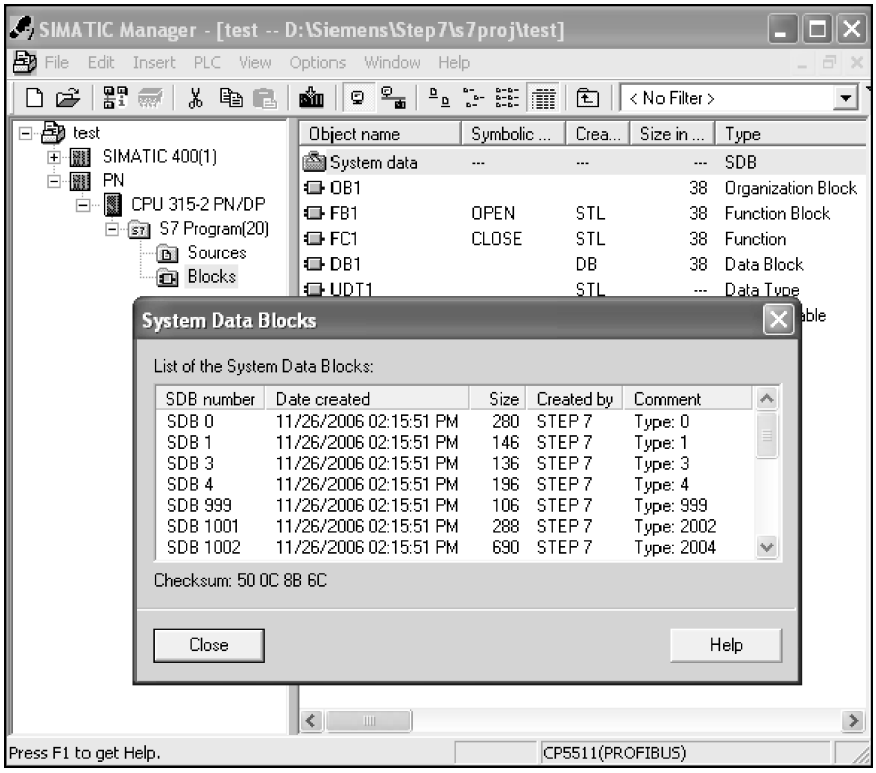



图 9-43 系统数据块

## 9.10.2 生成逻辑程序块

逻辑程序块与实际控制相对应，按需求由用户单独创建。点击“Blocks”图标，按鼠标右键，弹出下拉菜单，点击“Inset New Object”标签，插入所需的程序块，如图 9-44 所示。

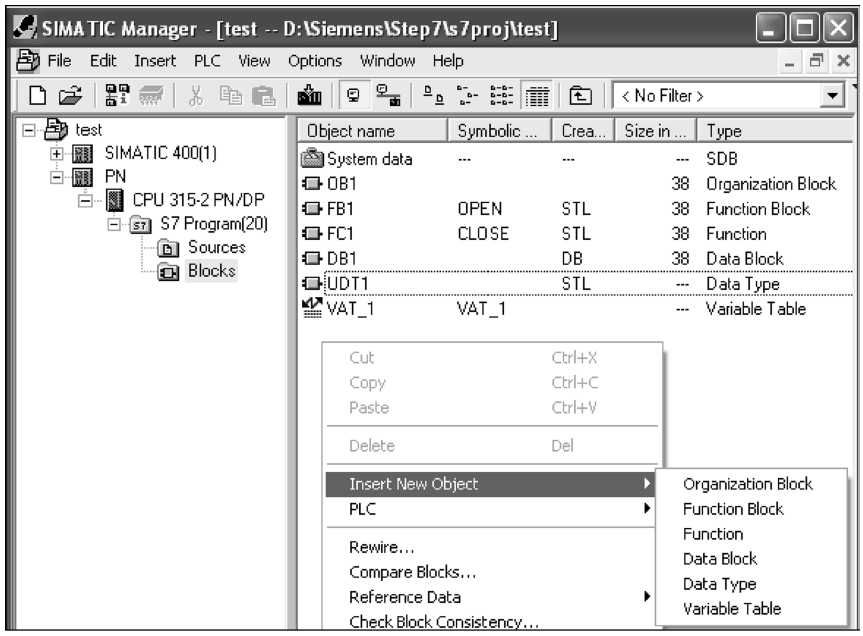



图 9-44 插入用户程序块

双击所要编辑的程序块，如 FC1，进入程序编辑界面，如图 9-45 所示。

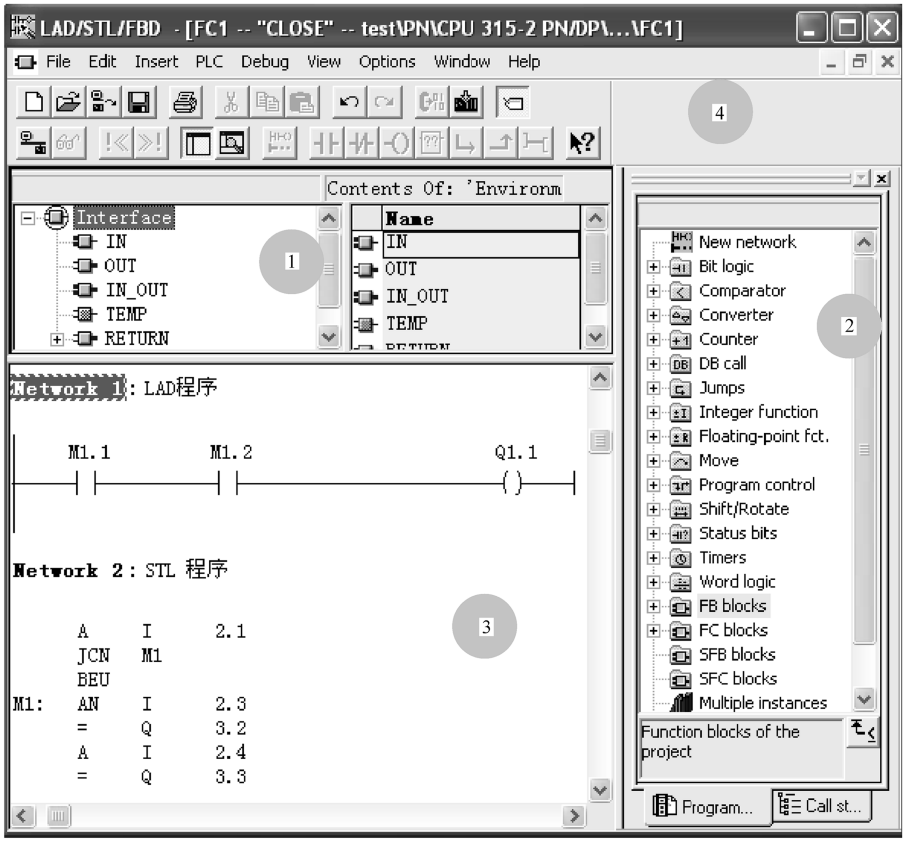



图 9-45　编辑用户程序界面

编程界面划分为四个区：

1）程序块参数传递的接口区。在 FC、FB 中可以声明输入、输出形参、区域数据和静态数据（适用 FB），在 OB 中只能定义区域数据，如果作为无形参的子程序，可以不定义接口区。

2）编程指令集，适用于 LAD、FBD 编程语言，可以使用鼠标将选择的指令拖放到代码区程序段中。STL 指令需要手动输入。编程指令集的函数库（Libraries）提供一些集成计算功能的函数和函数块，也可以创建函数库，用于包含用户编写的函数或函数块。

3）程序代码区，可以使用 STEP7 集成的三种编程语言，如 LAD、FBD 和 STL 进行编程，通过菜单命令“View”可以选择所需显示的编程语言，图 9-45 中以 LAD 编程语言显示程序，在“Network2”中，使用 STL 编写的代码不符合 LAD 编程语法，不能转换为 LAD 显示，所以在一个程序块中可以同时存在多种编程语言的显示。

4）工具栏，可以进行存盘、编译、打印、在线、监控等操作，工具栏中也包含 LAD 编程语言常用的指令。

### 9.10.3　地址替换功能

在编程中变量地址需要经常修改，例如在程序中，多次在不同的程序块中使用 IW10

(可能是位、字节、字信号),连接的信号模块出现故障而将所有输入信号连接到地址为 IW20 的备用模块上,这样需要将程序中所有与 IW10 相关的变量更换为 IW20,通过地址替换功能可以简单地解决上述问题。

下列地址区和程序块可以进行地址替换:

- 输入、输出地址区
- 位存储器、定时器、计数器
- 函数、函数块

实现地址替换功能的步骤如下:

1)在“SIMATIC Manager”中,选择需要进行地址替换的程序块(所有或部分程序块)。

2)选择菜单命令“Options”→“Rewire”。

3)在弹出的表格对话框中,输入旧地址(被替代地址)和新地址(替代值)。

4)选择选项“All addresses within the specified address area(指定地址区域内的所有地址)”,如果旧地址输入 IW10,新地址输入 IW20,没有选择该选项,只有 IW20 替换 IW10,选择该选项,则整个地址范围被替换,包括位、字节和字信号。

5)点击“OK”按钮,完成地址替换工作。

地址替换完成后弹出地址替换提示信息,该信息文件包含“旧地址”和“新地址”的列表和对每个程序块进行地址替换处理的个数。

### 9.10.4 块比较

块比较功能可以对同一个项目、不同项目、离线或在线任意两个程序块或所有程序块进行比较。

在“SIMATIC Manager”界面下进行块比较步骤如下:

1)在“SIMATIC Manager”中,选择需要比较的程序块或单个程序块。

2)选择菜单命令“Options”→“Compare Blocks”。

3)在显示的“Compare Blocks(块比较)”对话框中,选择比较类型(ONLINE/offline or Path1/Path2)。

4)对于“Path1/Path2”的比较:在 SIMATIC Manager 中,选择要比较所有的程序块或单一程序块。这些块将自动地输入到对话框中。

5)如果需要比较 SDB,选择“Including SDB”选项。

6)如果需要比较程序代码,选择“Execute code comparison”选项。在详细比较中,除了比较程序块中与运行相关的部分外(接口和程序代码),区域变量名和参数名的不同也将显示。此外,还可以选择“Including blocks created in different programming languages(包括由不同编程语言生成的块)”选项比较不同编程语言生成的程序块。

7)选择比较的内容后,点击“OK”按钮确认。

比较结果显示在“Compare Blocks-Results(比较块-结果)”的对话框中,如图 9-46 所示。

点击对话框中的“Details”按钮,可以显示详细比较结果(例如上次修改时间等)。点击“Go to”按钮,打开程序块显示差异之处。

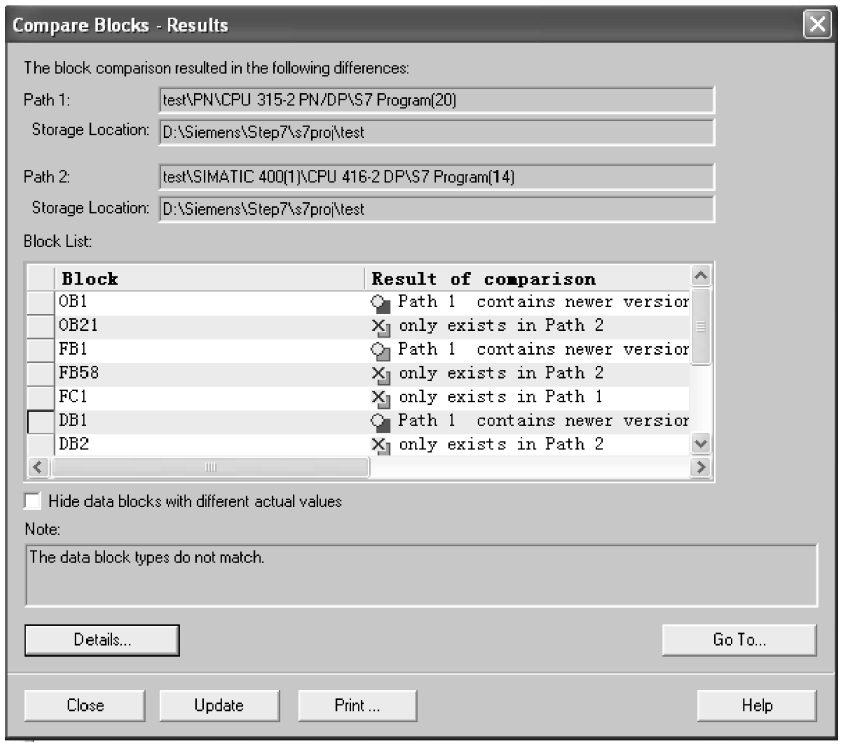



图 9-46　块比较结果

### 9.10.5　生成变量监控表

在“SIMATIC Manager”界面中，点击“Blocks”图标，按鼠标右键弹出下拉菜单，点击“Insert New Object”插入“Variable Table”变量监控表，在变量监控表中输入需要监控的变量，对变量可以进行监视、赋值、强制等操作，变量监控表存储于项目的离线程序块中，可以用于设备调试及后期的运行维护和修改。根据不同的控制对象可以在一个程序块中生成多个变量监控表。

### 9.10.6　程序块的一致性检查

如果一个函数或函数块在调用后再次更改接口参数（增加或减少），打开调用程序块时出现时间标签冲突，被调用的程序块变为红色，调用的程序编译不能通过。

每个程序块在创建和修改后都会生成时间标签，时间标签包含在程序块的属性中。以下几种情况可能产生时间标签冲突：

- 一个被调用程序块比调用程序块的时间标签新。
- 一个程序块比正在使用它的程序块时间标签新。
- 一个 UDT 比使用它的程序块时间标签更新；这些块可以是一个 DB 或其他的 UDT，或者在变量声明表中使用了该 UDT 的 FC、FB、OB。
- 一个 FB 比其相应的背景数据块的时间标签新。

- 一个 FB2 在 FB1 中被定义为静态变量，并且 FB2 的时间标签比 FB1 的更新。

如果程序中产生时间标签冲突，可以按以下步骤执行程序块的一致性检查：

1）在“SIMATIC Manager”界面中，选择所需的块（Blocks），使用菜单命令“Edit”→“Check Block Consistency”执行程序块的一致性检查，如图 9-47 所示，程序中 DB10 出现时间标签冲突。

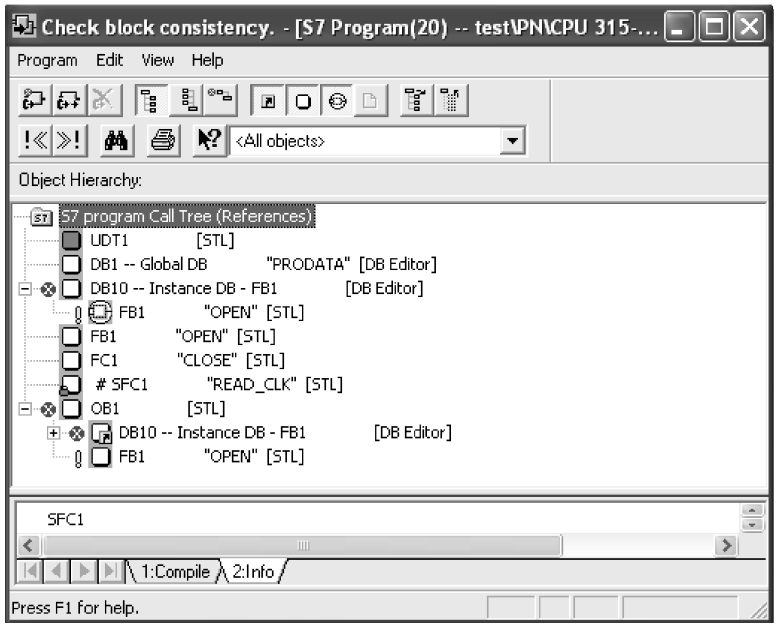



图 9-47　程序块一致性检查界面

2）如果程序块带有时间标签冲突，选择菜单命令“Program”→“Compile All”对程序块进行编译，编译器将自动消除程序块时间标签冲突。

3）如果仍然有些程序块的时间标签冲突不能消除，在界面的“Compile”信息输出窗口中，相应的程序块被标示错误信息，双击错误信息，自动跳转并指向该程序块在调用程序块中的位置，手动进行修改。

4）重新进行编译，直到没有时间标签冲突为止。

## 9.11　生成用户库函数

图 9-45 编程指令集函数库（Libraries）中包含 STEP7 提供的函数、函数块及系统函数、函数块，用户可以将在项目中编写的函数或函数块添加在函数库中，应用于其他项目。

在“SIMATIC Manager”界面中，使用菜单命令“File”→“New”或点击工具栏快捷键，弹出提示对话框，如图 9-48 所示。

选择“Libraries”标签，在“Name”栏中命名函数库，在“Type”栏中选择“Library”，在“Storage location”中可以定义函数块存储路径。点击“OK”按扭确认，函数库生成，将编写的函数或函数块拖放到创建的库中即可在编程界面中调用，如图 9-49 所示。

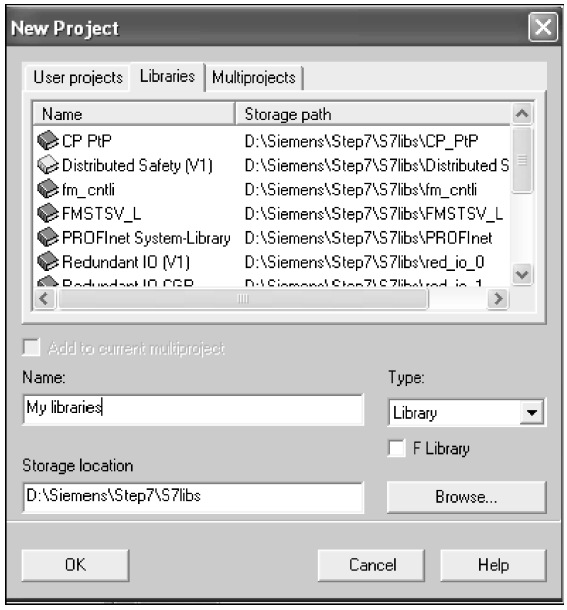



图 9-48　创建用户函数库

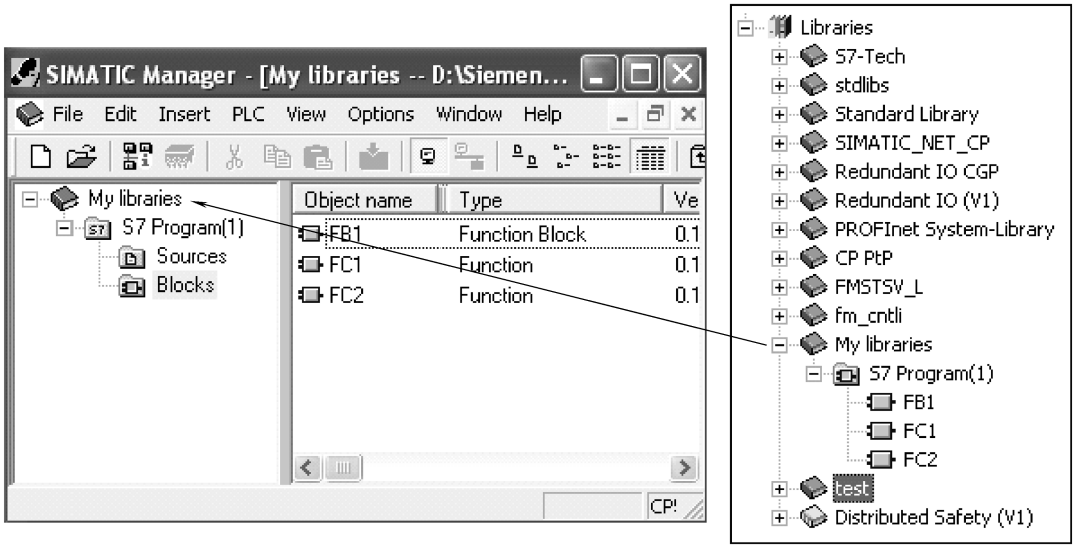



图 9-49　生成用户函数库

## 9.12　复制其他项目中的程序块

如果其他项目中带有一些计算和特殊功能的函数和函数块，可以复制到正在编辑的项目中使用，步骤如下：

1）打开需要复制程序块的项目。

2）用鼠标拖放功能，将所需的程序块复制到应用项目“Blocks”中。

3）如果复制的程序块与正在编辑项目中的程序块号码冲突，弹出对话框，提示是否覆盖当前项目中程序块或重新命名复制程序块的号码，选择更改程序块号码。

4）如果复制的程序块中包含分层调用结构，将调用的其他程序块一同复制，否则将出现故障。

## 9.13 生成源文件

在 PLC 站的“S7Programs”中，除程序块和符号表以外，还包括源文件（Sources）。源文件有以下功能：

1）创建源文件，在源文件中可以使用 STL 文本的编程方式编写程序，一个源文件中可以编写所有程序块，编写的源文件经过编译后生成相应的程序块并存储于项目“Blocks”中。源文件编程在这里不做介绍。

2）可以将所有的程序块以 STL 文本的方式保存。例如在程序编辑界面中，选择菜单命令“File”→“Generate Source”，按照提示创建源文件名称、选择所需生成源文件的程序块，最后生成源文件。点击源文件，按鼠标右键，在弹出的对话框中选择“Export Source”命令，可以将生成的源文件以 *. AWL 文件格式导出。

3）导入已经生成的源文件（*. AWL 文件格式），生成程序块。点击“Sources”，按鼠标右键，在弹出的对话框中选择“Insert New Object”→“External Source ”命令，导入源文件，打开编译后，生成的程序块存储于同一个“S7 Programs”的“Blocks”中。S5 程序转化为 S7 程序就是使用导入源文件的方式生成 S7 的程序块。

4）在源文件中可以更改程序块的属性，例如打开源文件中，在所需程序块的第 4 行添加文本“KNOW_HOW_PROTECT”，编译生成的程序块覆盖原有程序块，相应的程序块加保护锁而不能打开。反之删除源文件第 4 行添加的文本“KNOW_ HOW_ PROTECT”，经过编译后，相应的程序块可以打开。程序块加保护后应先保存源文件，然后删除项目中的源文件，这样在项目中因无源文件而无法进行编译打开加锁的程序块。

## 9.14 生成地址交叉参考

地址的交叉参考可以查询所有数据区的地址在程序中的使用概况，快速推断上一级的逻辑关系，方便程序的阅读，此外可以查询程序中有效地址区的分配，避免造成地址冲突。使用交叉参考表和程序编辑器都可以实现地址的交叉参考。

### 9.14.1 交叉参考表

交叉参考表给出了 S7 用户程序所用地址、程序结构的概览。通过交叉参考表可以显示输入（I）、输出（Q）、位存储器（M），定时器（T）、计数器（C）、函数块（FB）、函数（FC）、系统函数块（SFB）、系统函数（SFC）、I/O（外设地址）和数据块（DB）存储区中被用户程序使用的地址列表及程序块调用结构等。

在“SIMATIC Manager”界面中，使用菜单命令“Options”→“Reference Data”→

“Generate”生成交叉参考表，然后使用菜单命令“Display”显示交叉参考表，如图 9-50 所示。

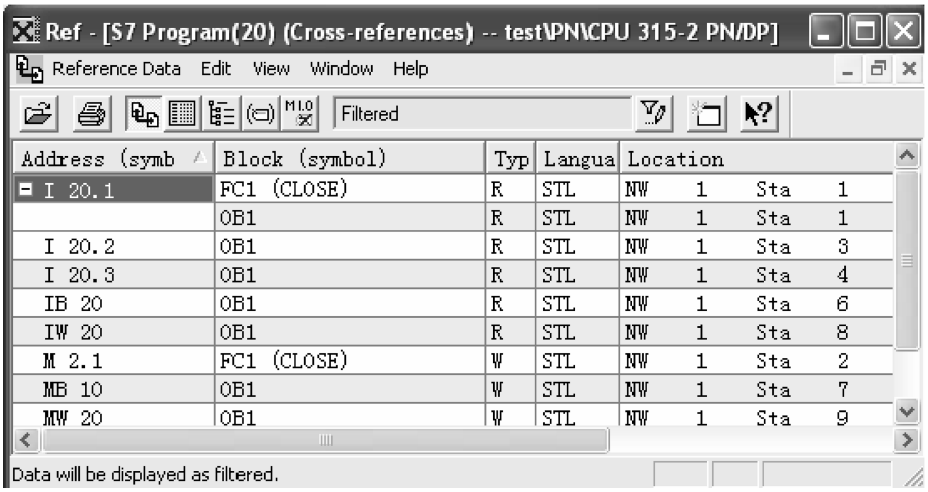



图 9-50　交叉参考表

使用菜单命令“View”或工具栏上的按钮，可以显示交叉参考表以下地址信息：

（1）交叉参考

列出存储区域 I、Q、M、P、T、C 以及 DB、FB、FC、SFB、SFC 程序块在用户程序中使用的概况。如果多次使用同一个变量，将显示在不同程序块中使用的位置，使用鼠标双击该地址栏，自动跳转并指向该变量在程序块中的位置。

（2）输入/输出及位存储器赋值表

显示用户程序已占用的定时器和计数器（T 和 C）以及 I、Q、M 存储区中的地址概况。符号“X”表示某个位被占用，“B”表示占用一个字节，“W”表示占用一个连续的字，“D”表示占用一个连续的双字。没有占用的数据区域可以分配使用，以避免地址冲突。

（3）程序结构

显示用户程序内块的分层调用结构，以及所使用的块及其嵌套层次。

（4）未使用的符号

列出了所有已在符号表中定义但未在用户程序中使用的符号。

（5）无符号的地址

列出了在用户程序中使用了但未在符号表中定义的绝对地址。

使用菜单命令“View”→“Filter”或工具栏上的按钮，可以在交叉参考表中设置交叉参考、输入/输出及位存储器赋值表、程序结构和无符号的地址过滤功能，如图 9-51 所示，在交叉参考数据中，只对位存储区开始地址为 10、20 的变量进行查看。

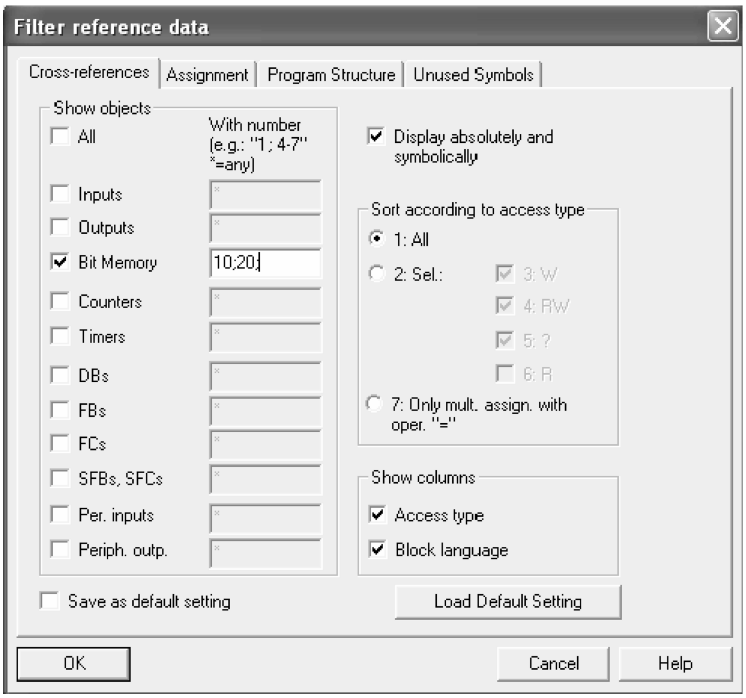



图 9-51 交叉参考表过滤功能

**注意：**

每次修改程序块后，打开交叉参考表时都会提示更新生成的交叉参考表。

## 9.14.2 在程序编辑器中快速查询地址的位置

在阅读程序时，为了理解控制过程的逻辑关系，往往需要查询某一个变量在其他程序块中的使用情况，如果使用交叉参考表查询，设置步骤多而不方便，使用程序编辑器中的交叉参考可以快速查询地址使用的位置。在程序中选择需要查询的地址，按鼠标右键，在弹出的下拉菜单中点击“Go To”→“Location”，弹出的对话框中显示所有与查询地址相关的程序块以及在程序块中具体的位置、读写类型等信息，使用鼠标双击显示的程序块，自动跳转并指向该变量在程序块中的位置。使用菜单命令“Go To”→“Local Application”可以在当前程序块中进行查询。

# 第 10 章　PLC 的通信功能

## 10.1　网络概述

一个大型自动化项目中常常包括若干个控制相对独立的 PLC 站，PLC 站之间通常需要传递一些连锁信号，HMI 系统通过网络控制 PLC 站的运行并采集过程信号归档，这些都需要通过 PLC 的通信功能实现，通信功能在大型项目中尤为重要。西门子工业通信网络统称为 SIMATIC NET，它提供了各种开放的、应用于不同通信要求及安装环境的通信网络。SIMATIC NET 主要定义下列内容：

- 网络传输介质；
- 通信协议和服务；
- PLC 及 PC 连网所需的通信处理器（Communication Processor，CP）。

对于通信数据量及通信实时性的要求，SIMATIC NET 提供了不同的通信网络，如图10-1所示。

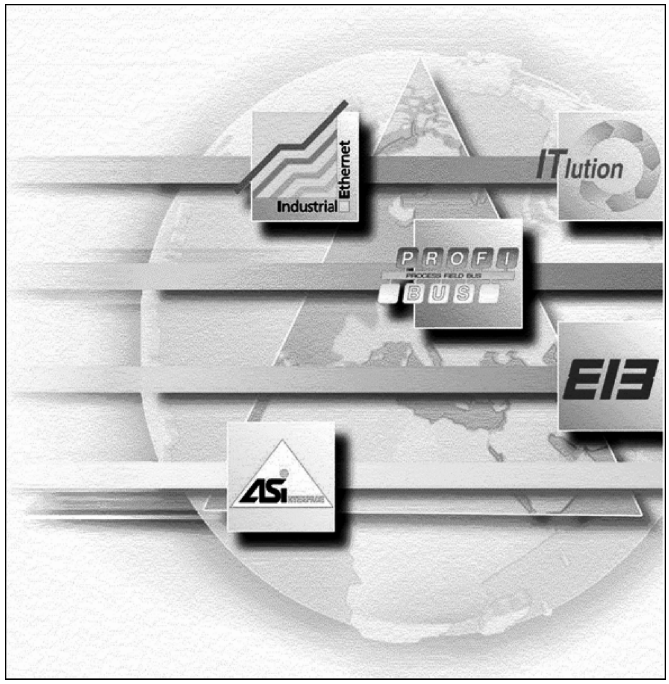



图 10-1　SIMATIC NET 网络结构

从上到下分别为工业以太网、PROFIBUS/MPI、EIB 和 ASI，通信的数据量由大到小、实时性由弱到强。

（1）工业以太网（Industrial Ethernet）

依据 IEEE 802.3 标准建立的单元级和管理级的控制网络，传输数据量大，数据终端传输速率为 100Mbit/s，通过西门子 X400 交换机，主干网络传输速率可达到 1000Mbit/s。

（2）PROFIBUS

Profibus（Process field Bus）作为国际现场总线标准 IEC 61158 的组成部分（TYPE Ⅲ）和国家机械制造业标准 JB/T10308.3-2001，具有标准化的设计和开放的结构，以令牌方式进行主-主或主-从通信。PROFIBUS 传输中等数据量，在通信协议中只有 PROFIBUS-DP（主-从通信）具有实时性。

（3）EIB

Instabus EIB（European Installation Bus）应用于楼宇自动化，可以采集亮度调节、百叶窗控制、温度测量及门控等信号，通过 DP/EIB 网关，可以将数据传送到 PLC 或 HMI 中。

（4）AS-Interface

AS-I（Actuator Sensor-Interface）网络通过 AS-I 总线电缆连接最底层的执行器及传感器，将信号传输至控制器。通信数据量小，适合位信号的传输，每个从站通常最多带有 4 个位信号，主站轮询 31 个从站的时间固定为 5ms，适合实时性的通信控制。

（5）串行通信

与带有串行通信接口（RS-422/485 和 RS-232C）的设备进行通信，通信报文透明，通信双方可以定义报文格式。使用串行通信安全性低，通常需要用户对通信数据作校验。

工业以太网通信速率高、传输数据量大，但是数据传输时间带有不确定性。基于工业以太网的 PROFINET 作为实时以太网完全满足现场实时性的要求，通过 PROFINET 可以实现通信网络的一网到底，即从上到下都可以使用同一种网络，如图 10-2 所示，便于网络的安装、调试和维护。

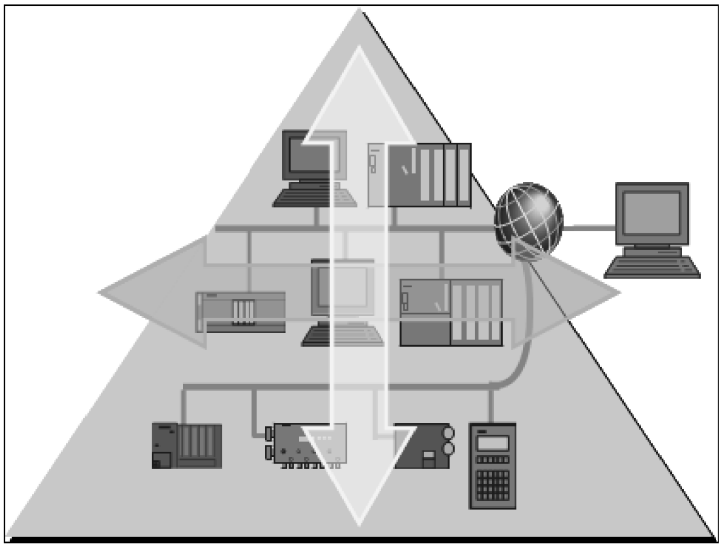


图 10-2　PROFINET 一网到底

下面将着重对 MPI、PROFIBUS 和工业以太网（包括 PROFINET 网络）及串行通信进行

介绍。

## 10.2　MPI 网络

MPI（Multipoint Interface）也是编程接口，集成于每个 S7-300、400 系列 PLC CPU。通过 MPI 可以组成 S7 系列 PLC 最简单的通信网络，是一种对数据传输速率要求不高，通信数据量不大，简单经济的通信方式。

### 10.2.1　MPI 的种类

1）在 PLC 侧，CPU 集成 MPI，不能通过通信处理器扩展。

2）HMI 操作面板集成 MPI 接口，不能通过通信处理器扩展。

3）编程器或上位机上插入通信处理器或使用适配器可以与 PLC 进行通信，可用的种类为：

- PC 适配器：PC Adapter，将串口或 USB 接口转换为 MPI。
- TS 适配器：TS Adapter，将 USB 接口转换为 MPI，TS Adapter II 集成调制解调器，可以直接连接模拟电话网或 ISDN 数字网（也可以连接外置调制解调器），进行远程编程。
- CP5512：带有 PCMCIA 总线接口的通信处理器，适用于笔记本电脑。
- CP5611：带有 PCI 总线接口的通信处理器，适用于台式计算机。
- CP5613：带有 PCI 总线接口的通信处理器，适用于台式计算机，内部集成处理器。

### 10.2.2　MPI 网络的通信速率

MPI 网络的通信速率可以设置为 19.2Kbit/s 和 187.5Kbit/s，如果 MPI 在硬件配置中可以设置为 PROFIBUS 接口，那么该接口支持 19.2Kbit/s 、187.5Kbit/s、1.5Mbit/s、3Mbit/s、6Mbit/s 和 12Mbit/s 等不同的波特率。

### 10.2.3　MPI 网络的拓扑结构

MPI 为 9 针 RS-485 接口，站点间使用 PROFIBUS 连接器和 PROFIBUS 通信电缆（屏蔽双绞线，与 PROFIBUS 电气网络介质相同）连接，MPI 网络不支持光纤连接，一个 MPI 网络最多可以有 32 个站点。MPI 网络支持总线型拓扑结构，通信速率为 187.5Kbit/s 时，通信距离最长为 50m，可以使用中继器进行扩展，扩展的方式有两种：

1）两个中继器间有站点，每个中继器最长可以扩展 50m，例如 3 个 PLC 站通过两个中继器扩展，PLC 站到中继器的最长距离为 50m，3 个 PLC 站通信最长距离可以达到 200m（中继器在中间位置）。

2）两个中继器间没有站点，PLC 站到中继器的最长距离为 50m，中继器间最长可以扩展 1000m，如图 10-3 所示。

最多可以增加 10 个中继器，两个站点之间最长通信距离为 9100m，在总线的两端使用终端电阻。

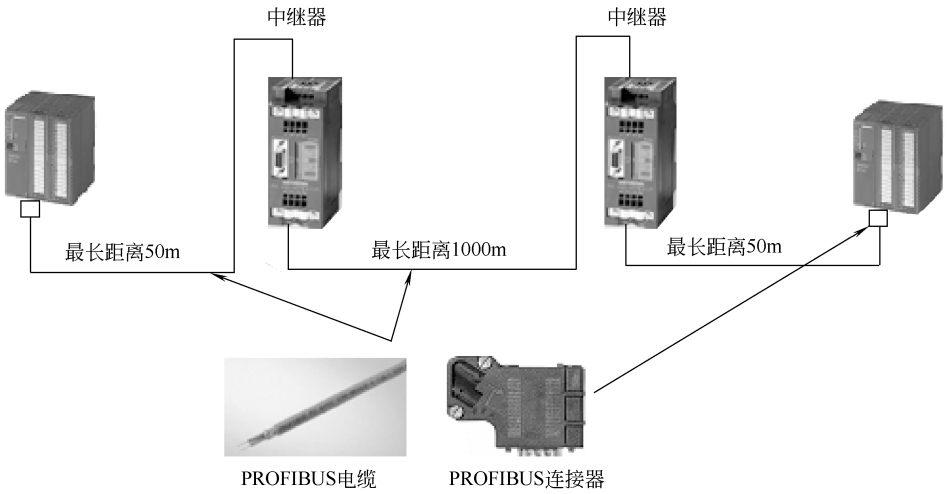



图 10-3 MPI 网络扩展

### 10.2.4 PLC 通过 MPI 网络的通信方式

PLC 之间通过 MPI 网络有三种通信方式：

1）全局数据包（GD）通信方式。

2）不需配置连接的通信（调用 SFC）。

3）需配置连接的通信（调用 SFB）。

### 10.2.5 全局数据包通信方式

全局数据包（GD）的通信方式是在 PLC 硬件配置中，设置通信 PLC 站点间的发送区和接收区，不需要任何程序处理，这种通信方式适合 S7-300、400 系列 PLC 之间的相互通信并且所有通信站点必须在同一项目下。

全局数据包的通信方式配置步骤如下：

1）在硬件配置中将所有通信的 PLC 站点设置为同一 MPI 网络。

2）在“SIMATIC Manager”界面下，点击项目名，选择需要配置的 MPI 网络，使用菜单命令“Options”→“Define Global Date”进入配置界面。

3）双击 GD ID 右边的 CPU 栏，从项目中选择需要通信的 PLC 站 CPU，CPU 栏总共有 15 列，这就意味着最多有 15 个 CPU 能够参与通信。

4）在每个 CPU 列中配置发送和接收数据区，例如键入 DB1.DBW0：10 表示从 DB1.DBW0 开始长度为 10 个字的数据区（S7-300 GD 数据包最大为 22B，S7-400 GD 数据包最大为 54B）如图 10-4 所示，同一行为其他 PLC 站相应的数据接收区和发送区。

5）选择相应数据包，点击图标，将该数据包作为发送数据区，点击图标将该数据包作为接收数据区（默认为接收数据包），GD 通信为广播方式，在每一行中只能有一个站进行发送。

6）如果需要，可在其他列中配置通信数据区。每个 CPU 可使用的 GD 包的个数参考

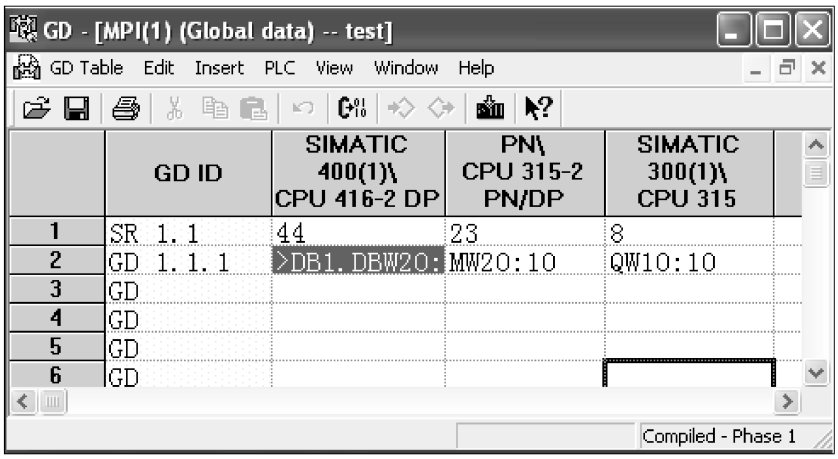



图 10-4　配置 GD 通信区

CPU 订货手册。

7）使用菜单命令“View”→“Scan Rates”，显示发送和接收数据区的扫描时间，根据 CPU 的负荷，在“SR”中配置扫描时间（与 CPU 扫描时间有关）。

8）编译无错后，下载配置信息，通信建立。

通常配置的 GD 包通信方式，在程序中不能干预，S7-400 CPU 支持事件触发的接收和发送，在 SR 中设置扫描频率为 0，在程序中调用 SFC60（GD_SND）和 SFC61（GD_RCV）进行通信控制。

**注意：**

GD 通信不带有通信确认信息，通信状态需要编程校验。

### 10.2.6　不需配置连接的通信

不需要在连接表中建立通信连接，而在程序中调用系统函数，在赋值参数时指定通信方的 MPI 地址。这种通信方式适合 S7-300、400 系列 PLC 之间的相互通信，PLC 站可以不在同一个项目下，发送、接收可以通过程序控制，灵活性强，最大通信数据量为 76B。与 GD 数据包通信方式相比，发送和接收程序带有确认信息。不需配置连接的通信方式占用 CPU S7 标准通信资源，CPU 支持 S7 标准通信资源的数量与 CPU 型号有关，可以参考 CPU 订货手册。

不需配置连接的通信有两种通信方式：双边通信方式和单边通信方式。

1. 双边通信方式

通信双方都需要调用通信函数，一方调用发送块，另一方就要调用接收块来接收数据，发送块是 SFC65（X_SEND），接收块是 SFC66（X_RCV），发送频率过高，将造成通信方没有响应，加重 CPU 的负荷，建议在 OB35 中调用发送块，间隔 100ms 发送一次，在 OB1 中编写接收块。发送的示例程序如下：

```
CALL" X _SEND"
REQ      : = M0. 0
CONT     : = TRUE
```

```
DEST_ID   : = W#16#4
REQ_ID    : = DW#16#1
SD        : = P#DB1. DBX0. 0 BYTE 76
RET_VAL   : = MW10
BUSY      : = M0. 5
```

发送函数 SFC65 的参数含义：

REQ　　：触发发送请求，为 1 时发送。

CONT　　：为 1 表示发送数据完成后仍将占用 S7 标准通信资源，为 0 表示发送数据完成后释放 S7 标准通信资源，需要重新触发发送任务。

DEST_ID　：接收方的 MPI 地址。

REQ_ID　：数据包的标识符，用户定义。

SD　　：发送区，ANY 指针格式，示例程序中的发送区为 DB1 中前 76B。发送区最大为 76B。

RET_VAL　：表示发送任务的状态。

BUSY　　：为 1 表示通信正在执行，只有为 0 时，才能执行下一个发送任务。

示例程序中将 DB1 中前 76B 发送到 MPI 地址为 4 的 CPU 中，在通信方需要调用 SFC66（X_RCV）接收数据，示例程序如下：

```
CALL     "X_RCV"
EN_DT    : = TRUE
RET_VAL  : = MW10
REQ_ID   : = MD12
NDA      : = M0. 1
RD       : = P#DB2. DBX 0. 0 BYTE 76
```

接收函数 SFC66 的参数含义：

EN_DT　：为 1 表示激活接收功能。

RET_VAL：表示接收任务的状态。

REQ_ID　：数据包的标识符，接收在发送函数“REQ_ID”中定义的标识符。多个站向一个站发送数据时，通过标识符识别发送方。

NDA　　：为 1 时表示接收到新的数据包，为 0 则没有，沿信号。

RD　　：接收区，示例程序中将 DB2 中前 76B 定义为接收区。

**小窍门：**

如果通信数据超过 76B，可以利用通信数据区分批次发送。

2. 单边通信方式

单边通信只在一方编写通信程序，类似客户机与服务器的关系，在客户机上调用相应通信函数对服务器中数据进行读写操作。这种通信方式适合 S7-300/400/200 系列 PLC 之间通信，S7-300/400 系列 PLC CPU 可以同时作为客户机和服务器，S7-200 系列 PLC 只能作为数据的服务器（不能调用通信函数）。SFC67（X_GET）用来读服务器中的数据并存储于本地的数据区中，SFC68（X_PUT）用来将本地数据区的数据写到服务器中指定的数据区中。

调用 SFC67（X_GET）示例程序如下：

```
CALL    " X _GET"
REQ          : = M0. 0
CONT         : = TRUE
DEST_ID      : = W#16#4
VAR_ADDR     : = P#DB1. DBX 0. 0 BYTE 76
RET_VAL      : = MW20
BUSY         : = M0. 5
RD           : = P# DB2. DBX 0. 0 BYTE 76
```

通信函数 SFC67 的参数含义：

REQ　　：触发通信任务，为 1 时任务执行。

CONT　　：为 1 表示执行通信任务后仍将占用 S7 标准通信资源，为 0 表示执行完成后释放 S7 标准通信资源，需要重新触发通信任务。

DEST_ID　　：通信方的 MPI 地址。

VAR_ADDR　：通信方的数据区。

RET_VAL　　：任务执行的状态。

BUSY　　：为 1 表示通信正在执行，只有为 0 时，才能执行下一个通信任务。

RD　　：本地数据接收区。

示例程序中将 MPI 地址为 4 的 PLC 站 DB1 中的前 76B 复制到本地数据 DB2 的前 76B 中。

调用 SFC68（X_PUT）示例程序如下：

```
CALL    " X _PUT"
REQ          : = M0. 0
CONT         : = TRUE
DEST_ID      : = W#16#4
VAR_ADDR     : = P#M 100. 0 BYTE 76
SD           : = P#DB2. DBX 0. 0 BYTE 76
RET_VAL      : = MW22
BUSY         : = M0. 2
```

通信函数 SFC68 的参数含义与 SFC67 相似，这里不再介绍，示例程序中将本地数据区 DB2 的前 76B 中的数据写到 MPI 地址为 4 的 PLC 站 MB100 到 MB175 数据区中。

### 10. 2. 7　需要配置连接的通信

需要在连接表中建立通信连接，并在程序中调用 SFB，这种通信方式适合于 S7-300 系列 PLC 与 S7-400 系列 PLC 和 S7-400 系列 PLC 之间的通信，S7-300 系列 PLC 与 400 系列 PLC 通信时，只能进行单边通信，S7-300 系列 PLC 作为一个数据的服务器（不能调用通信函数块），S7-400 系列 PLC 作为客户机调用通信函数块（SFB14、SFB15）对 S7-300 系列 PLC 的数据进行读写操作。S7-400 系列 PLC 之间通信时，S7-400 系列 PLC 既可以作为数据的服务器，同时又可以作为客户机进行单边通信，双边通信方式时，通过调用 SFB12、SFB13 进行数据的发送和接收，最大通信数据量为 160B，通信带有确认。需要配置连接的

通信方式占用 CPU S7 通信资源，CPU 支持 S7 通信资源的数量与 CPU 型号有关，可以参考 CPU 订货手册。

需要配置连接的通信有两种通信方式：双边通信方式和单边通信方式。

（1）双边通信方式

适合 S7-400 系列 PLC 之间的通信，通信连接需要在连接表中建立。实现的通信步骤如下：

1）在“SIMATIC Manager”界面中，使用菜单命令“Options”→“Configuring Network”进入网络配置界面“NETPRO”。

2）点击 S7-400 系列 PLC CPU 后，在下方出现连接表，双击连接表中任意一空白行弹出对话框，选择需要连接的对象（与之通信的 PLC 站），在“Type”中选择“S7 Connection”，点击“Apply”键进行确认，进入连接属性界面，如图 10-5 所示。在连接属性中列出连接 ID（编程时需要使用）、使用的网络、通信双方地址、使用的接口等参数，如果是 S7-400系列 PLC 与 S7-300 系列 PLC 间的单边通信，“One-way”选项自动被选择。

3）在被连接 CPU 的连接表中自动生成一个连接，该连接同样表示本地 CPU 连接的信息。双击打开连接，记录本方的连接 ID，用于编程时使用。

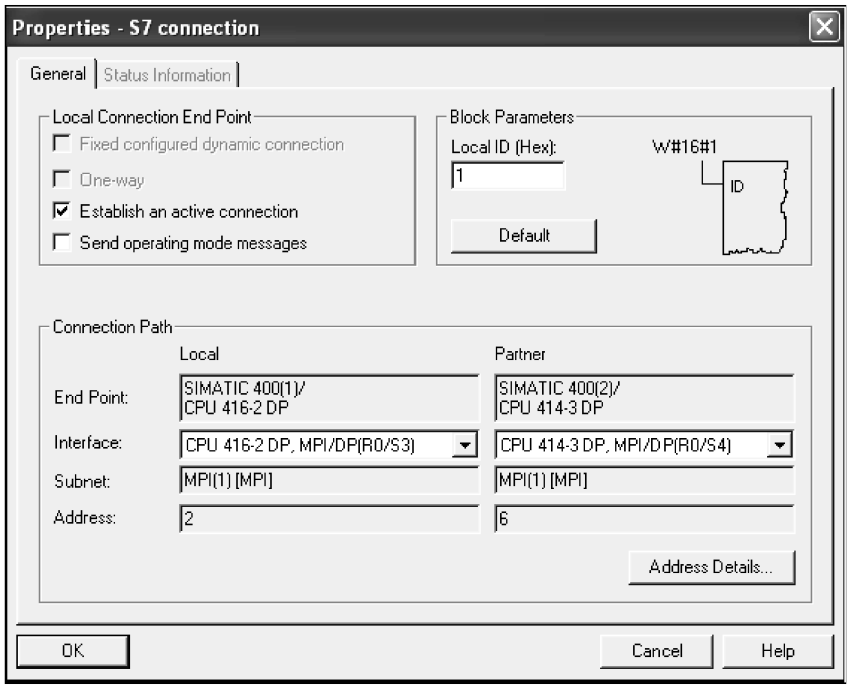



图 10-5　S7 连接属性

4）通信连接建立后，需要调用通信函数发送和接收数据。在本地 OB35（间隔发送）中调用 SFB 12" BSEND" 发送数据，示例程序如下：

```
CALL" BSEND" , DB20
REQ      : = M1. 1
```

```
R        : = M1. 2
ID       : = W#16#1
R_ID     : = DW#16#1
DONE     : = M1. 3
ERROR    : = M1. 4
STATUS   : = MW20
SD_1     : = P#DB1. DBX0. 0 BYTE 60
LEN      : = MW24
L        60
T        MW     24
```

通信函数块 SFB12 的参数含义：

REQ　：沿触发，每个上升沿触发一次数据的发送。

R　：为 1 时，停止通信任务。

ID　：通信连接 ID，指定一个通信连接，包括通信双方的通信参数（参考连接表）。

R_ID　：标识符，发送与接收函数必须一致。

DONE　：每次发送成功，产生一个上升沿。

ERROR　：错误位。

STATUS　：通信状态字，如果错误位为 1，可以查看通信状态信息。

SD_1　：发送区。

LEN　：发送数据（字节）的长度。

示例程序中 S7-400 系列 PLC 发送 DB1 中前 60B。

5）在通信方 PLC OB1 中相应调用 SFB13 "BRCV" 接收数据，示例程序如下：

```
CALL   "BRCV", DB30
EN_R   : = TRUE
ID     : = W#16#1
R_ID   : = DW#16#1
NDR    : = M10. 1
ERROR  : = M10. 2
STATUS : = MW40
RD_1   : = P#DB2. DBX 0. 0 BYTE 60
LEN    : = MW60
```

通信函数块 SFB13 的参数含义：

EN_R　：为 1 时，激活接收功能。

ID　：通信连接 ID，指定一个通信连接，包括通信双方的通信参数（参考连接表）。

R_ID　：标识符，发送与接收函数块标识必须一致。

NDR　：每次接收到新数据，产生一个上升沿。

ERROR：错误位。

STATUS：通信状态字。

RD_1 ：接收区。

LEN ：接收数据（字节）的长度。

示例程序中 S7-400 系列 PLC 将接收到的数据存储于本地数据区 DB2 的前 60B 中。

（2）单边通信方式

适合 S7-300 系列 PLC 与 S7-400 系列 PLC、S7-400 系列 PLC 之间的通信，以 S7-400 系列 PLC 与 S7-300 系列 PLC 通信为例，在 S7-400 系列 PLC 侧调用 SFB14 "GET" 和 SFB15 "PUT"，对 S7-300 系列 PLC 数据进行读写。通信的步骤如下：

1）与上例相同，在“NETPRO”中将需要通信的双方建立连接表（只能在 S7-400 系列 PLC 侧建立）。

2）在 S7-400 系列 PLC CPU 编写 SFB14“GET”，示例程序如下：

```
CALL   SFB  14，DB40
REQ    :=M0.0
ID     :=W#16#1
NDR    :=M1.1
ERROR  :=M1.2
STATUS :=MW100
ADDR_1 :=P#M 200.0 BYTE 100
ADDR_2 :=
ADDR_3 :=
ADDR_4 :=
RD_1   :=P#DB2.DBX 0.0 BYTE 100
RD_2   :=
RD_3   :=
RD_4   :=
```

通信函数 SFB14 的参数含义：

REQ ：读请求。每个上升沿触发读任务。

ID ：通信连接 ID，指定一个通信连接，包括通信双方的通信参数。

NDR ：每次接收到新数据，产生一个上升沿。

ERROR ：错误位。

STATUS ：通信状态字。

ADDR1_4 ：通信方的数据区。

RD1_4 ：本地数据接收区。

示例程序中将通信方数据区 MB200～MB299 复制到本地数据 DB2 的前 100B 中。

3）SFB15"PUT" 写操作的示例程序如下：

```
CALL   "PUT"，DB100
REQ    :=M20.1
ID     :=W#16#1
DONE   :=M20.2
ERROR  :=M20.3
```

STATUS ：= MW22
ADDR_1：= P#DB10. DBX0. 0 BYTE 10
ADDR_2：=
ADDR_3：=
ADDR_4：=
SD_1　　：= P#DB1. DBX0. 0 BYTE 10
SD_2　　：=
SD_3　　：=
SD_4　　：=

通信函数 SFB15 与 SFB14 不同参数的含义：

DONE　：每次发送成功后，产生一个上升沿。
SD1_4　：本地数据发送区。

示例程序中将本地数据区 DB1 前 10 字节存储的数据写到通信方数据区 DB10 的前 10B 中。

### 10. 2. 8　PLC 通过 MPI 与 HMI 通信

与 HMI 操作面板连接，在 PROTOOL 或 WinCC FLEXIBLE 编程软件中，直接定义连接 CPU 的 MPI 地址和机架上的槽号，即可建立通信。

与 HMI WinCC 连接，添加“S7 Protocol Suite”驱动，选择连接的通信驱动，按鼠标右键，在弹出的连接属性对话框中选择上位机使用的通信处理器或适配器。在“MPI”网络建立与 PLC 连接时，选择 CPU 的 MPI 地址和在机架上的槽号即可建立通信。

## 10. 3　PROFIBUS 网络

PROFIBUS（Process Field Bus）适用于中小数据量站点间的通信，PROFIBUS 提供不同的通信协议和服务，既可以应用单元级站点的通信，又可以与现场 I/O 站直接通信，经过 DP/PA 耦合器的转换，还可以连接防爆区的仪表。

### 10. 3. 1　PROFIBUS 接口的种类

S7-300 系列 PLC：CPU31X-2DP 至少带有一个 PROFIBUS 接口（有些 CPU MPI 可以配置为 PROFIBUS 接口）、CP342-5、CP343-5。

S7-400 系列 PLC：所有 CPU 都集成 PROFIBUS 接口（所有 CPU MPI 都可以配置为 PROFIBUS 接口）、CP443-5、IM467。

编程器或上位机：PC 适配器、TS 适配器、CP5512、CP5611、CP5613。

### 10. 3. 2　PROFIBUS 的访问机制

连接在 PROFIBUS 网络上的站点，按照它们的地址顺序组成一个逻辑令牌环，令牌从低站地址到高站地址传递，如果传递到最高站地址 126，则跳回到最低站地址重新开始，获得令牌的主站可以在拥有令牌期间，对属于它的从站进行发送或读取数据的操作，令牌只在主动站点间进行传递，PROFIBUS-DP 从站通常为被动站点，不能得到令牌。如图 10-6 所示，

主站和 HMI 可以传递令牌，从站 1、2、3 不能得到令牌。PROFIBUS-DP 通常为一主多从，主站轮询从站，保证通信的实时性。

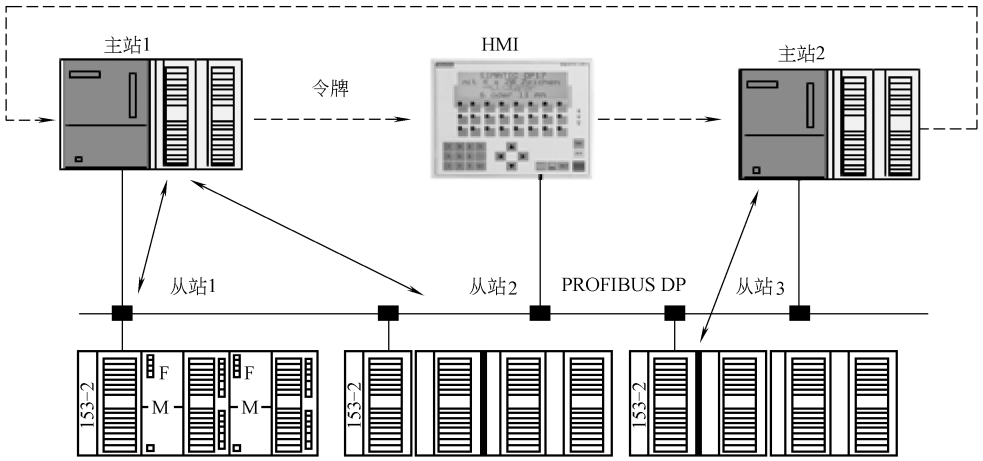



图 10-6 令牌的传递

## 10.3.3 PROFIBUS 网络的通信速率与通信距离

PROFIBUS 网络支持的通信速率与通信距离相对应，见表 10-1。

**表 10-1 PROFIBUS 通信速率与通信距离的对应关系**

| 波特率/（Kbit/s） | 9.6 ~ 187.5 | 500 | 1500 | 3000 ~ 12000 |
|---|---|---|---|---|
| 总线最大长度/m | 1000 | 400 | 200 | 100 |

## 10.3.4 PROFIBUS 网络拓扑结构

PROFIBUS 总线符合 EIA RS-485［8］标准，PROFIBUS RS-485 的传输程序是以半双工、异步、无间隙同步为基础的。传输介质可以是光缆或屏蔽双绞线。

1. 电气传输时网络拓扑

电气传输时为总线型拓扑结构，使用 PROFIBUS 电缆和 PROFIBUS 连接器（参考 10.2 节）连接 PROFIBUS 站点，每一个 RS-485 网段为 32 个站点，包括有源网络元件（RS-485 中继器），在总线的两端必须使用终端电阻，结构如图 10-7 所示。

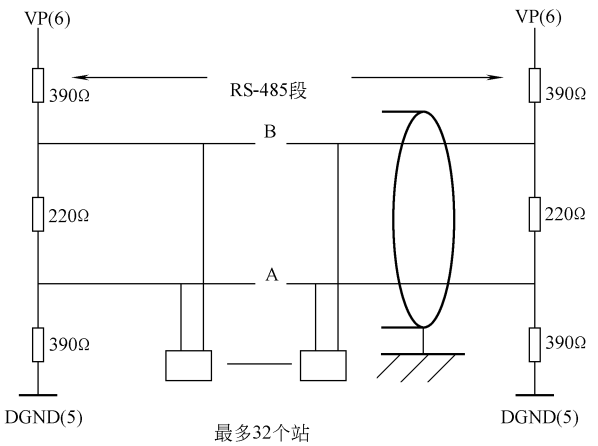



图 10-7 PROFIBUS 电气网络结构

总线的终端电阻集成在连接器及网络部件中（通常不使用连接器直接连接的网络部件都带有终端电阻）。如果需要扩展总线的长度或者 PROFIBUS 站点数大于 32 个时，就要使用 RS-485 中继器（ RS-485 Repeater）进行扩展。例如，PROFIBUS 的长度为 500m，而波特率要求达到 1. 5Mbit/s，对照表 10-1，波特率为 1. 5Mbit/S 时，最大的长度为 200m，要扩展到 500m，就需要加入两个 RS-485 中继器，每个中继器可扩展 200m。西门子公司的 RS-485 中继器具有信号放大和再生功能，在一条 PROFIBUS 总线上最多可以安装 9 个 RS-485 中继器。

使用 RS-485 中继器，PROFIBUS 网络可以进行总线和树形拓扑，如图 10-8 所示，网段 2 是网段 1 的信号放大，在中继器上带有拨码开关，可以设置网段 1、网段 2，隔离以及使能终端电阻。网段 1 可以象网段 2 一样进行扩展，但是在连接最后一个站点时，必使能终端电阻（在中继器上不能使能终端电阻）。

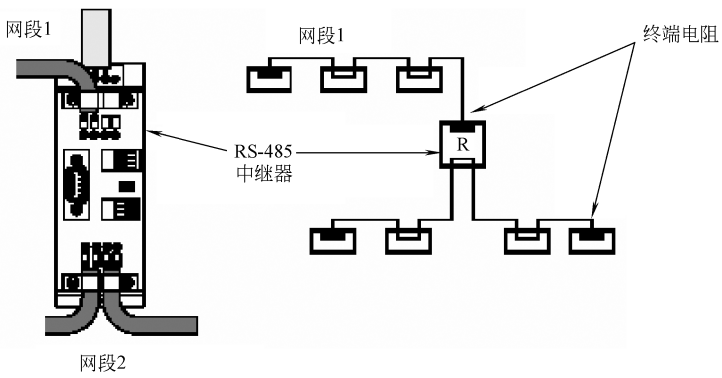



图 10-8　使用 RS-485 中继器的拓扑结构

由于中继器占用站点数，使用多个中继器进行扩展时，应遵守下列规则：

- 一个网段最大有 32 个站点（包括 Repeater 及其他带有 RS-485 驱动的元件）。
- 第一个和最后一个网段最多有 31 个站点 。
- 两个中继器间最多有 30 个站。
- 每一个网段终端必须有终端电阻。

2. 光纤网络拓扑结构

OLM（Optical Link Module）模块转换光电信号，可以使用玻璃光纤、PCF 光纤、塑料光纤组成光纤网络，使用光纤的种类及 OLM 类型、通信距离见表 10-2。

**表 10-2　OLM、光纤类型与通信距离**（OLM 间的通信距离）

| 光纤种类 / OLM类型 | 塑料光纤 980/1000μm | PCF 光纤 200/230μm | 玻璃光纤 62. 5/125μm | 玻璃光纤 50/125μm | 玻璃光纤 10/125μm |
|---|---|---|---|---|---|
| OLM/P | 80m | 400m | — | — | — |
| OLM/G | — | — | 3000m | 3000m | — |
| OLM/G-1300 | — | — | 10km | 10km | 15km |

一条 PROFIBUS 网络可安装 OLM 的个数与总的可连接站点数有关（最大为 125 个站点，OLM 在总线上作为一个站点），整个网络传输速率最大为 12Mbit/s，使用 OLM 的 PROFIBUS 网络可以有三种拓扑结构，即总线型结构；冗余环；星形结构。使用光纤将总线型结构的头尾两段连接，即可构成冗余环，如图 10-9 所示。

组成光纤环网的 OLM 类型必须相同（连接相同类型的光纤），使用星形拓扑结构，可

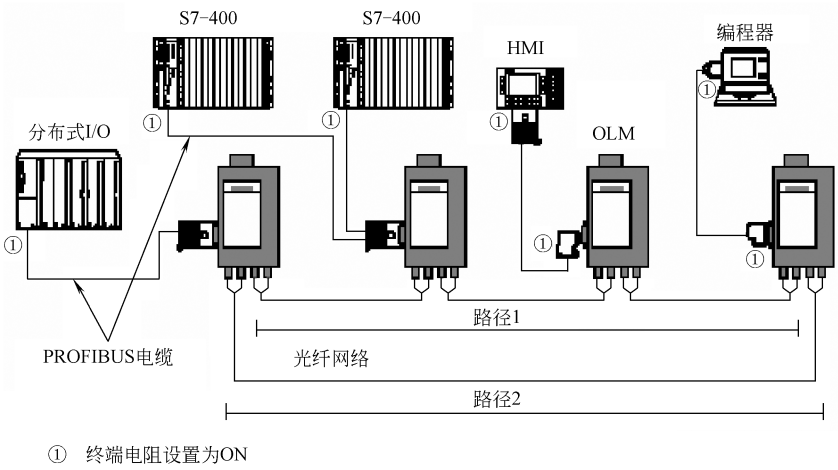



图 10-9 使用 OLM 构成冗余环网

以混合使用不同类型的 OLM，但是两个相连接的 OLM 类型必须相同，星形拓扑结构如图 10-10 所示。

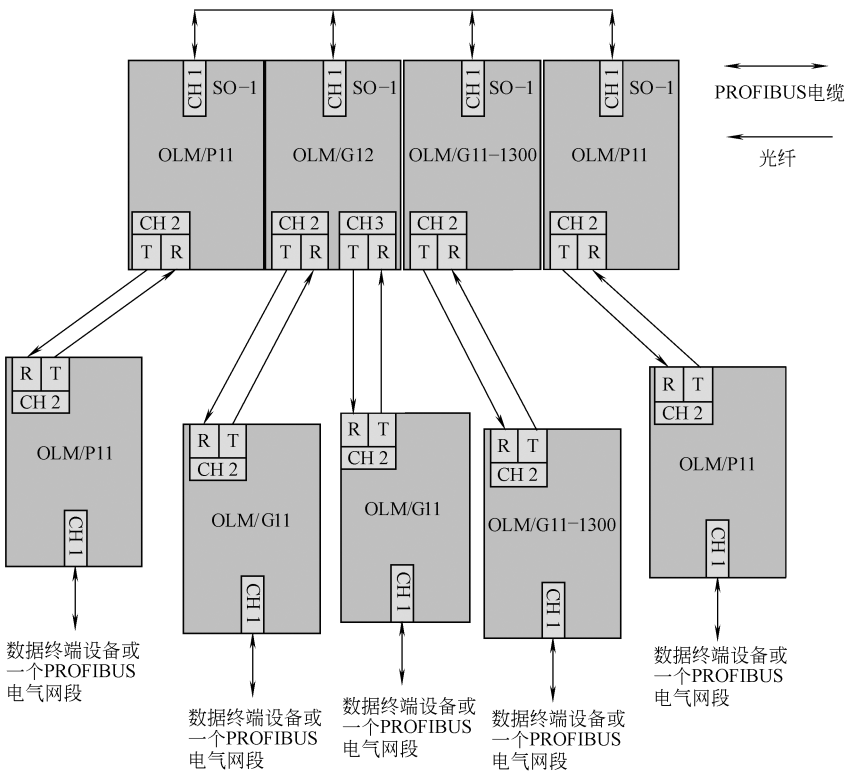



图 10-10 使用 OLM 构成星形拓扑

PROFIBUS 网络除电气和光纤网络外，还支持激光、红外线、轨道滑触线等通信介质。使用 RS-485 中继器和 OLM 作为网络元件不需要在“SIMATIC Manager”中配置，但是信号经过转换或放大后会有时间延时，如果在一条 PROFIBUS 总线上连接多个 OLM 或 RS-485 中继器，延时时间超过 PROFIBUS 在硬件配置中设定的轮询监控时间，通信不能建立，需要在总线配置时加入使用网络元件的个数，以补偿通信的延时时间。在硬件配置界面中，双击 PROFIBUS 网络图标，弹出属性对话框，在“Network Settings”标签下，点击“Options”，在“Cable”标签页中配置网络参数，如图 10-11 所示。

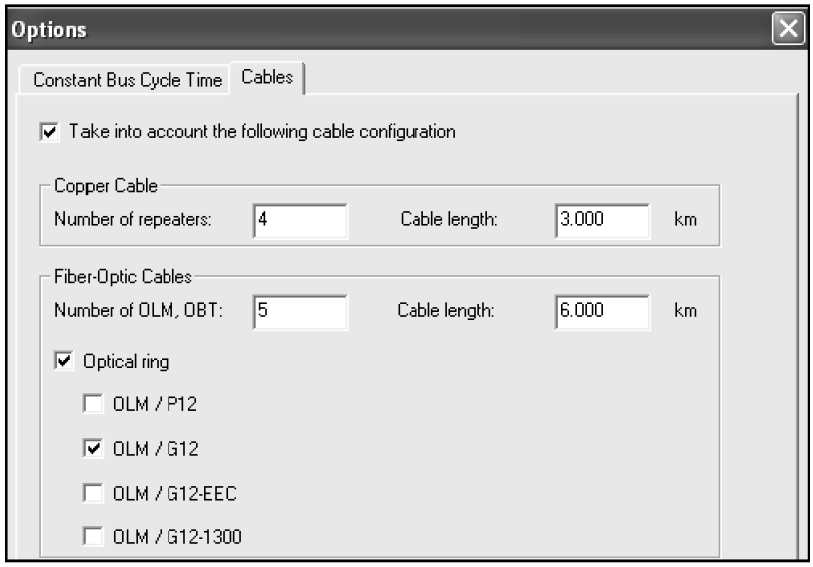



图 10-11　配置 PROFIBUS 参数

输入网络上使用的 OLM 与 RS-485 中继器的个数、总线的长度，如果使用光纤环网，激活“Optical ring”选项，并选择使用 OLM 的类型，硬件编译后软件自动计算 PROFIBUS 网络轮询监控时间。

3. PROFIBUS-PA 拓扑结构

PROFIBUS-PA 通过 DP/PA Coupler 耦合器或通过 DP/PA Link 连接耦合器转换为 PA 总线。使用 DP/PA Coupler 耦合器，整个 PROFIBUS 网络通信速率将降为 45.45Kbit/s（PA 侧为 31.25Kbit/s）；使用 DP/PA Link 最多可以连接五个 DP/PA Coupler 耦合器，DP 侧的通信速率与 PA 侧无关。DP/PA Coupler 耦合器是一个网络元件，不需要在 SIMATIC Manager 中配置，DP/PA Link 则需要配置（连接 DP/PA Coupler 耦合器不需要配置）。PA 总线向仪表供电，每个仪表需要消耗总线电流，这样每个 DP/PA Coupler 耦合器输出的电流、总线末端电压（大于 9V）及 PROFIBUS-DP 网络站点数量的限制决定可以连接仪表的个数，通过 DP/PA Link 可以连接更多的仪表。PA 总线支持总线和树形网络拓扑结构，总线型拓扑结构如图 10-12 所示。

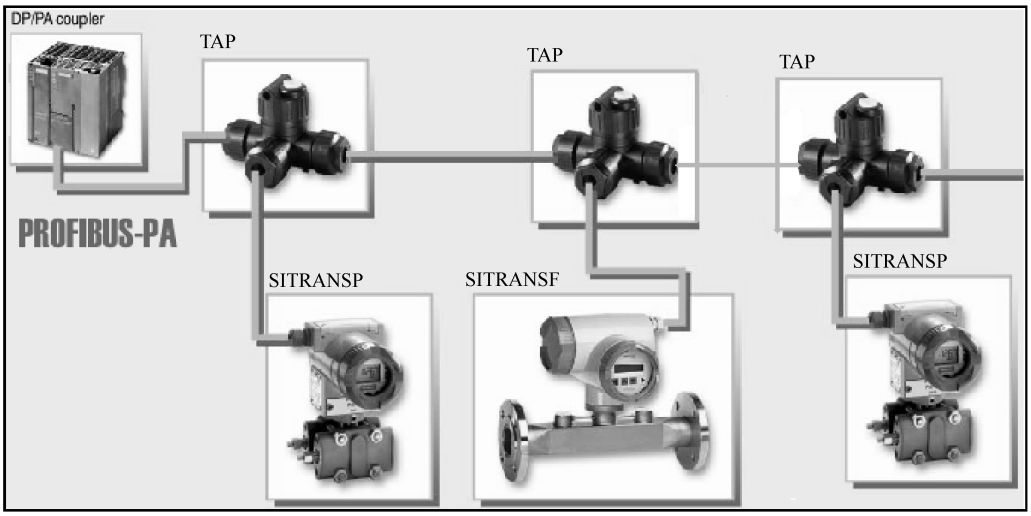



图 10-12 PA 总线型拓扑结构

### 10.3.5 PROFIBUS 支持的通信协议与服务

PROFIBUS 提供了三种标准和开放的通信协议：DP、FMS 和 PA，除此之外还有一些优化的通信服务适合西门子公司 PLC 之间、PLC 与 HMI 之间的通信。

1. PROFIBUS-DP

使用了 ISO/OSI 网络模型的第一层和第二层，这种精简的结构保证了数据的高速传送，具有很高的实时性，特别适合 PLC 与分布式 I/O 设备之间的通信。

2. PROFIBUS-FMS

使用了 ISO/OSI 网络模型第一层、第二层和第七层。应用层（第七层）包括 FMS（Fieldbus Message Specification，现场总线报文规范）和 LLI（低层接口）。FMS 包含应用协议和提供的通信服务，LLI 建立各种类型的通信关系，并给 FMS 提供不依赖于设备的对第二层的访问。

FMS 处理单元级（PLC 和 PC）的数据通信，其功能强大和配置灵活，但是由于配置和编程比较繁琐，目前很少使用，逐渐被淘汰了。

3. PROFIBUS-PA

PROFIBUS-PA 使用扩展的 PROFIBUS-DP 协议进行数据传输，它执行规定现场设备特性的 PA 设备行规。传输技术依据 IEC1158-2［7］标准，确保本质安全和通过总线对现场设备供电。使用 DP/PA 耦合器和 DP/PA Link，很容易将 PA 设备集成到 PROFIBUS-DP 网络之中。

4. PROFIBUS-S7

使用了 ISO/OSI 网络模型第一层、第二层和第七层。特别适用于 S7 PLC 与 HMI（PC）和编程器之间的通信。也可以应用于 S7-300 系列 PLC 与 S7-400 系列 PLC、S7-400 PLC 之间的通信。

5. PROFIBUS-FDL（与 S5 兼容通信）

使用了 ISO/OSI 网络模型第一层、第二层。数据传输快，特别适合 S7-300 系列 PLC、S7-400 系列 PLC 和 S5 PLC 之间的通信。

6. PG/OP 通信服务

编程器和操作面板可以通过 PROFIBUS 对网络上的站点进行访问。

在下面将主要介绍 PROFIBUS-DP、FDL、S7 的通信。

### 10.3.6　PROFIBUS-DP 通信

PROFIBUS-DP 为主从结构，主站可以得到令牌轮询扫描从站。DP 通信方式实时性强，可以连接分布式 I/O。主站与一个从站最多通信数据为 244 字节输入和 244 字节输出。

1. 安装 GSD 文件

PROFIBUS-DP 除了连接西门子公司的分布式 I/O 系列、驱动等从站，还可以连接其他厂商的从站设备。在 SIMATIC Manager 硬件配置界面里 PROFIBUS-DP 设备目录中，可以添加其他厂商硬件设备，这需要由生产厂商提供 GSD 文件（GSD 文件是对设备通用描述，通常以 *.GSD 或 *. GSE 结尾）或从相关网站上下载。安装 GSD 文件的步骤如下：

1）将 GSD 文件复制到编程器的硬盘中。

2）打开 SIMATIC Manager 硬件配置界面，关闭所有应用。

3）使用菜单命令“Options”→“Install GSD File”，在弹出的对话框中，选择需要安装的 GSD 文件。

4）使用菜单命令“Options”→“Update Catalog”更新硬件目录。

5）使用菜单命令“View”→“Catalog”打开硬件目录，在“PROFIBUS-DP”子目录中的“Additional Field Devices ”中，可以发现安装的从站。

6）将从站配置到主站 PROFIBUS-DP 网络上（参考 9.8.1 节远程 I/O 从站的配置过程），从站通信数据区的含义必须参考从站设备手册。

**注意：**

1）打开包含有其他厂商从站设备的项目时（打开硬件配置），如果 SIMATIC Manager 没有安装相应的 GSD 文件，会弹出对话窗口，提示是否需要安装所需的 GSD 文件，点击确认后，项目中包含的 GSD 文件自动导入“Additional Field Devices ”硬件目录中。

2）从 PLC 中上传硬件配置时，如果上传的硬件配置中带有其他厂商的从站设备，而在 SIMATIC Manager 没有安装相应的 GSD 文件，在硬件配置中不能进行编辑下载，否则 PLC 不识别相应的从站设备。

2. 配置智能从站

通过 PROFIBUS-DP，不但可以连接分布式 I/O 从站，也可以将 S7-300 系列 PLC、S7-400 系列 PLC 站点作为智能从站连接到主站上，以 S7-400 系列 PLC CPU 作为主站，以 S7-300 系列 PLC CPU 作为智能从站为例介绍配置的步骤：

先配置 S7-300 系列 PLC CPU 从站：

1）创建 S7-300 系列 PLC 站，在硬件配置中插入 S7-300 系列 PLC CPU，例如 CPU315-2DP。

2）双击集成的“DP”栏（X2 槽位），弹出 DP 接口属性对话框，在“Operating Mode”标签中，选择“DP Slave”作为从站。如果选择“Test，Commissioning，Routing”选项，则

智能从站可以传递令牌，站点作为网关支持网络路由功能并可以与 HMI WinCC 通过 S7 协议建立通信连接。

3）在“Configuration”标签中，输入从站的通信接口，通信区占用 I、Q 地址区，例如 16 字节输入和 16 字节输出，在通信数据区“Consistency”选项中，如果选择“UNIT”，表示通信数据中每个字节都是独立的单元；如果选择“ALL”，表示通信数据是一个数组，数组中的元素是设置的字节。DP 主站与 CPU 间的数据交换可能是分批次进行的，选择“ALL”保证数据处理的完整性（通信数据在同一数据包中，同时处理），例如驱动器 PKW 参数，选项自动设置为“ALL”不能修改。

**注意：**

如果选择“ALL”，在 CPU 中需要调用 SFC14、SFC15 对通信数据进行读写操作。新型号的 S7-300 系列 PLC、400 系列 PLC CPU 间的主从通信，对通信区的读写操作由 CPU 处理，只需要在硬件中配置，不需要调用 SFC14、SFC15。

4）编译存盘，在程序块中创建 OB82、OB86（故障中断），从站工作完成。

配置 S7-400 系列 PLC CPU 主站：

1）创建 S7-400 系列 PLC 站，在硬件配置中插入 S7-400 系列 PLC CPU，例如 CPU416-2DP。

2）在弹出新建 PROFIBUS 网络时，选择与从站使用相同的网络并配置主站站号。

3）在硬件目录中，选择“PROFIBUS-DP”→“ Configured Stations”→“CPU31X”拖放到主站网络中。

4）在弹出 DP 从站属性对话框“Connection”标签中，点击“Connect”键，将主从 CPU 进行连接。

5）在 DP 从站属性对话框“Configuration”标签中，点击“Edit. .”键，将从站通信接口区配置对应主站的通信接口区，如图 10-13 所示。

6）在配置界面中，只需配置主站的通信起始地址和输入、输出类型，配置规则为从站输入数据区对应主站输出数据区；从站输出数据区对应主站输入数据区。其他参数也可以修改，但是通信双方配置必须匹配。

7）编译存盘，在程序块中创建 OB82 ~ OB86、OB122（故障中断），主站工作完成。

**注意：**

如果主站和从站不在一个项目下，从站配置不需要改变（不能选择“Test，Commissioning，Routing”选项），在主站配置中，需要在硬件目录中安装从站的 GSD 文件，在“Additional Field Devices ”目录中，将从站拖放到主站网络上，配置从站的站号及通信区，通信区中的输入、输出先后次序、格式必须与从站中的设置相匹配。

3. PROFIBUS-DP 其他的功能

1）同步与冻结功能：主站访问多个从站的数据会有先后，同步功能可以使主站在某一时刻发送的命令同时传递到从站中；冻结可以使从站某一时刻的输入信号同时传递到主站中。同步、冻结强调的是某一时刻，执行后，从站数据停止更新，如需循环更新从站数据必须去除同步、冻结命令。

2）等时同步功能：等时同步计算所有需要同步模块的输入、输出时间、信号到从站接口模块的传送时间、主站扫描从站的时间、触发 CPU 程序执行时间，通过等时同步功能可

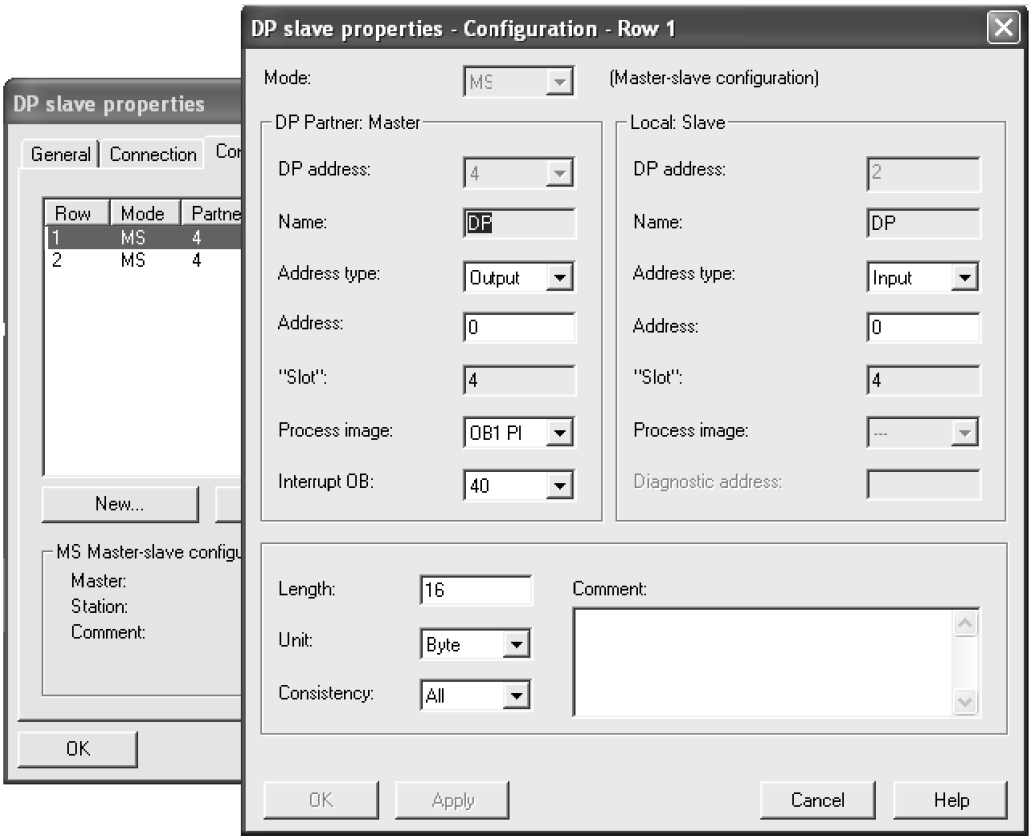



图 10-13　配置 PROFIBUS-DP 主从通信接口（智能从站）

以使几个从站中设定的输入信号同时被 CPU 扫描，CPU 的输出信号同时发送到输出模块上。只有 S7-400 系列 PLC CPU 支持等时同步功能，从站和输入、输出模块也必须支持等时同步功能。

3）DPV1 功能：非周期访问从站接口中存储的数据，减少通信的负荷。在 S7 主站 CPU 中调用 SFC58、SFC59 访问从站接口中的数据。DP 的等时同步、从站时钟同步、从站输入信号的时间标签功能属于 DPV2 功能。

### 10.3.7　PROFIBUS-FDL 通信

FDL（Fieldbus Data Link）主－主通信方式，每个站点具有传送令牌的能力，适合于单元级主站间的数据交换，通信双方使用发送和接收函数进行通信，每个通信数据包最大为 240 个字节，两个站点间可以发送和接收多个数据包。FDL 通信还支持多播和广播等多种通信方式。只有 PROFIBUS 通信处理器 CP 支持 FDL 的数据传输，CP342-5、CP343-5 用于 S7-300 系列 PLC；两种型号的 CP443-5 用于 S7-400 系列 PLC；上位机系统中可以插入 PROFIBUS 网卡，如 CP5512、CP5611、CP5613 实现 FDL 通信。每一个通信处理器可以同时与多个主站建立通信连接，例如 CP342-5 的连接数为 16 个，其他通信处理器的连接个数参考样本手册。

FDL 通信不占用 CPU 的“S7 Connection”通信资源，也是 S7-300 系列 PLC 在 PROFIBUS 上实现主站通信的常用方式。下面以 S7-300 系列 PLC 在相同的项目下为例介绍实现 FDL 通信的步骤：

1）在一个项目下建立两个 S7-300 系列 PLC 站。分别进行硬件配置，插入 CPU 和 CP342-5，创建 PROFIBUS 网络，分配 PROFIBUS 站号，使两个站点在相同的网络上。

2）在“SIMATIC Manager”界面中，使用菜单命令“Options”→“Configuring Network”进入网络配置界面“NETPRO”。

3）点击其中一个 S7-300 CPU，后在下方出现连接表，双击连接表中任意一空白行弹出对话框，选择需要连接的对象（与之通信的 S7-300 系列 PLC 站），在“Type”中选择“FDL”，点击“Apply”键确认并进入 FDL 连接属性界面，如图 10-14 所示。在连接属性中自动列出连接 ID、连接名和块参数（编程时需要使用）。

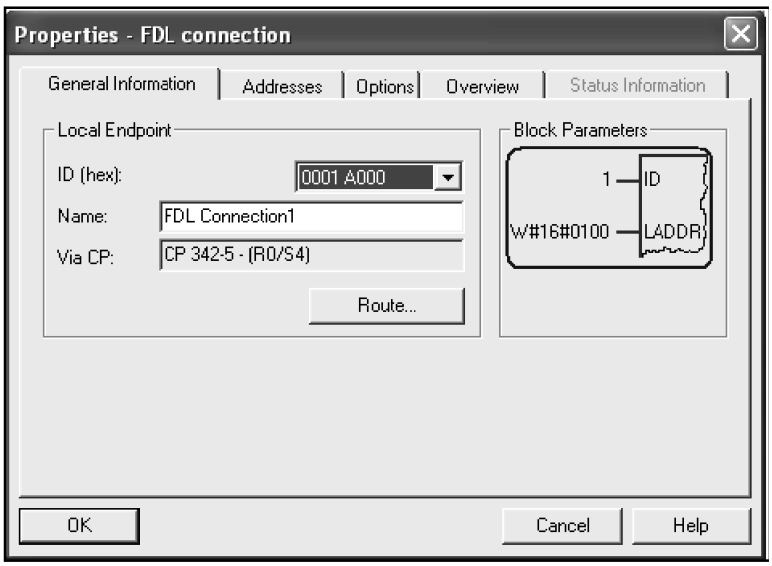



图 10-14　FDL 连接属性界面

4）在被连接 CPU 的连接表中自动生成一个连接，该连接同样表示本地 CPU 的连接信息。双击打开连接，记录本方的块参数，用于编程时使用。

5）在一个 CPU OB35（间隔发送）中调用发送函数 FC5 AG_SEND（“Libraries”→“Standard Library”→“SIMATIC_NET_CP”→“CP300”，如果是 S7-400 系列 PLC CPU，在“CP400”目录下），示例程序如下：

```
CALL" AG _SEND"
     ACT   := M20. 1
     ID    := 1
     LADDR := W#16#100
     SEND  := P#DB1. DBX 0. 0 BYTE 60
     LEN   := 60
```

```
        DONE : = M1. 2
        ERROR: = M1. 3
        STATUS: = MW2
```

通信函数 FC5 的参数含义：

ACT　：沿触发信号。
ID　：参考本地 CPU 连接表中的块参数。
LADDR　：参考本地 CPU 连接表中的块参数。
SEND　：发送区，最大通信数据为 240 字节。
LEN　：实际发送数据长度。
DONE　：每次发送成功，产生一个上升沿。
ERROR　：错误位。
STATUS　：通信状态字，如果错误位为 1，通过状态字查找错误原因。

示例程序中 S7-300 系列 PLC 发送 DB1 中前 60 字节。

6）在通信方 CPU OB1 中调用接收函数 FC6 AG_RECV（“Libraries”→“Standard Library”→“SIMATIC_NET_CP”→“CP300”，如果是 S7-400 CPU，在“CP400”目录下），示例程序如下：

```
CALL " AG _RECV"
     ID    : = 1
     LADDR : = W#16#100
     RECV  : = P#DB2. DBX 0. 0 BYTE 60
     NDR   : = M10. 1
     ERROR : = M10. 2
     STATUS: = MW12
     LEN   : = MW14
```

通信函数 FC6 的参数含义：

ID　：参考本地 CPU 连接表中的块参数。
LADDR　：参考本地 CPU 连接表中的块参数。
RECV　：接收区。接收区应大于等于发送区。
NDR　：每次接收到新数据，产生一个上升沿。
ERROR　：错误位。
STATUS　：通信状态字。
LEN　：实际接收数据长度。

将配置信息和通信程序下载到相应的 CPU 中，通信建立。示例程序中 S7-300 系列 PLC 将接收的数据存储于本地数据区 DB2 的前 60 字节中。两个站点通过一个连接可以同时进行发送和接收任务。

7）如果通信不能建立，双击 CP 模块，在属性对话框中选择“Diagnostics”标签，联机后点击“Run. .”键进入诊断界面，可以诊断连接状态、发送成功的数据包和实际接收的数据包。诊断功能在这里不作详细介绍。

在实际应用中，通信的站点往往不在同一项目下，由于程序保密不可能复制对方程序，

这样需要通信双方相互提交通信长度、PROFIBUS 站地址、LSAP 地址及选择的 PROFIBUS 参数组（通常选择“Standard”）等信息。下面以 S7-300 系列 PLC 在不同的项目下为例介绍实现 FDL 通信的步骤：

1）在项目下建立 S7-300 系列 PLC 站。进行硬件配置，插入 CPU 和 CP342-5，创建 PROFIBUS 网络，按双方协定分配 PROFIBUS 站号、选择 PROFIBUS 参数组。

2）在“SIMATIC Manager”视窗中，使用菜单命令“Options”→“Configuring Network”进入网络配置界面“NETPRO”。

3）点击 S7-300 系列 PLC CPU 后在下方出现连接表，双击连接表中任意一空白行弹出对话框，选择连接的对象为“Unspecified”，在“Type”中选择“FDL”，点击“Apply”键确认，进入 FDL 连接属性界面，如图 10-15 所示。在“General Information”标签中，自动列出连接 ID、连接名和块参数（编程时需要使用）。

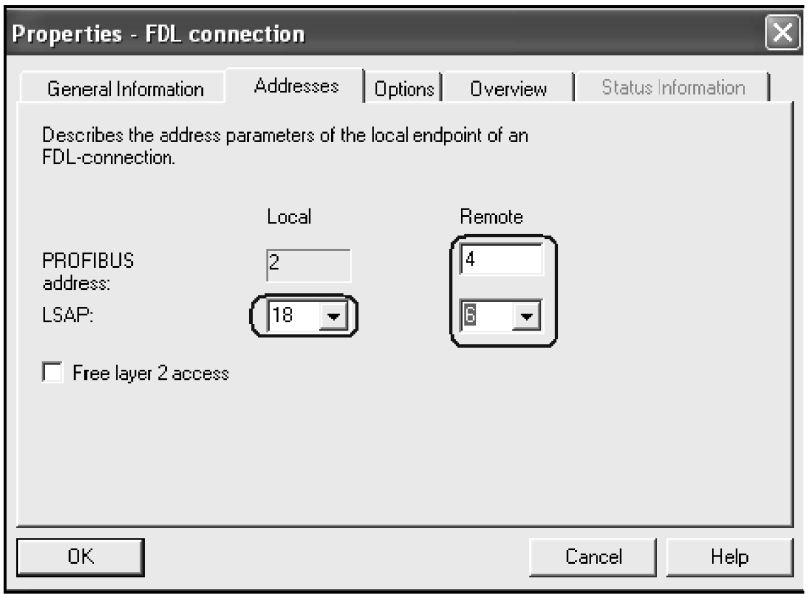



图 10-15　FDL 连接属性“Addresses”标签

4）在“Addresses”标签中，按双方协定设置本方的 LSAP 和对方的 PROFIBUS 地址以及 LSAP 地址，LSAP 地址范围与通信方式有关，在下拉菜单中显示允许的范围。

5）通信方按相同的方法进行配置。

6）编写通信程序（与在相同项目下建立 FDL 连接的通信程序相同），将配置信息和通信程序下载到 CPU 中，通信建立。

### 10.3.8　PROFIBUS-S7 通信

在 PROFIBUS 网络上，也可以使用 S7 通信服务实现 PLC 间的通信。S7 通信服务使用了 ISO/OSI 网络模型的第七层，不依赖于使用的网络，通过 MPI（需要配置连接的通信）、PROFIBUS 和工业以太网都可以实现 S7 通信服务。

S7 通信需要在连接表中建立通信连接，这种通信方式适合于 S7-300 系列 PLC 与 S7-400 系列 PLC 以及 S7-400 系列 PLC 之间的通信。如果 S7-300 系列 PLC 使用集成的 PROFIBUS 接口与 400 通信时，只能进行单边通信，S7-300 系列 PLC 作为一个数据的服务器，S7-400 系列 PLC 作为客户机通过调用通信函数块（SFB14、SFB15）对 S7-300 系列 PLC 的数据进行读写操作。如果 S7-300 系列 PLC 使用通信处理器 CP（例如 CP342-5）与 S7-300 或 S7-400 系列 PLC 进行通信，S7-300 系列 PLC 既可以作为数据的服务器同时又可以作为客户机进行单边通信（“Libraries”→“Standard Library”→“SIMATIC_NET_CP”→“CP300”调用 FB14、FB15），也可以调用发送和接收函数块进行双边通信（“Libraries”→“Standard Library”→“SIMATIC_NET_CP”→“CP300”调用 FB12、FB13）。S7-400 系列 PLC 之间通信没有限制，既可以作为数据的服务器同时又可以作为客户机进行单边通信，也可以发送和接收数据进行双边通信（调用 SFB12、SFB13）。

S7-300 系列 PLC 使用 CP 最大通信数据量为 32KB，S7-400 系列 PLC 最大通信数据量为 64KB，不同的通信函数块决定通信数据量的大小，见表 10-3。

**表 10-3　通信函数块与通信数据量的关系**

| 本方 CPU | 对方 CPU | SFB/FB | 参数 SD_i RD_i ADDR_i（1≥ i≥4）BYTE | | | |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| S7-300 系列 PLC | 任意 | PUT/GET<br>USEND | 160 | — | — | — |
| | | BSEND | 32768 | — | — | — |
| S7-400 系列 PLC | S7-300 系列 PLC | PUT | 212 | 196 | 180 | 164 |
| | | GET | 222 | 218 | 214 | 210 |
| S7-400 系列 PLC | S7-400 系列 PLC | PUT | 452 | 436 | 420 | 404 |
| | | GET | 462 | 458 | 454 | 450 |
| | | USEND | 452 | 448 | 444 | 440 |
| | | BSEND | 65534 | — | — | — |

S7-300 系列 PLC 使用 CP 可以扩展 CPU 内部的 S7 通信连接资源（只限于 PLC 间通信），通过一个 CP 与其他 PLC 间的通信，最多可以建立 16 个“S7 Connection”只占用一个 CPU 内部的 S7 通信连接资源。S7-400 系列 PLC CPU 每建立一个“S7 Connection”都占用 CPU 内部一个 S7 通信连接资源。CPU 中 S7 通信资源的数量与 CPU 型号有关，可以参考 CPU 订货手册。

在 PROFIBUS 网络配置 S7 通信的步骤，参考 MPI 网络小节中“需要配置连接的通信方式”，配置过程和调用的通信函数块相同。

### 10.3.9　PLC 通过 PROFIBUS 与 HMI 通信

（1）与操作面板的连接

与 HMI 操作面板连接，在 PROTOOL 或 WinCC FLEXIBLE 编程软件中，直接定义连接 PLC 站的 PROFIBUS 地址及 CPU 的槽号（即使连接通信处理器 CP）即可建立通信。

（2）与 HMI WinCC 的连接

与 HMI WinCC 的连接，添加“S7 Protocol Suite”驱动，选择连接的通信驱动，按鼠标

右键，在弹出的连接属性对话框中选择上位机使用的通信处理器或适配器。在“PROFIBUS”驱动中，在建立与 PLC 的连接时选择 PLC PROFIBUS 地址及 CPU 在机架上的槽号（即使连接通信处理器 CP），即可建立通信。WinCC 通过 PROFIBUS-DP 通信驱动也可以与 S7-300、400 系列 PLC 建立通信连接，WinCC 作为 DP 主站，PLC 作为从站，需要在 SIMATIC Manager 或 SIMATIC NET 中配置，在这里不作介绍。

（3）建立 OPC 服务器或与其他监控软件的连接

上位机安装 SIMATIC NET 后可以用作 OPC 服务器，其他监控软件可用作 OPC 的客户端，通过 OPC 通道可以与 OPC 服务器建立通信关系，访问与 OPC 服务器连接的 PLC。使用 VB、VC 等软件也可以开发 OPC 客户端的监控画面，读写 PLC 中的变量。

其他厂商的监控软件带有 S7 通信服务的驱动，通常需要安装 SIMATIC NET 驱动软件并配置通信连接。配置通信连接与配置 OPC 服务器的过程相同，步骤如下：

1）在 SIMATIC Manager 或 SIMATIC NET PC（如果上位机只安装 SIMATIC NET）中，打开所要与 PLC 建立连接的项目并创建“SIMATIC PC Station”。

2）双击打开“SIMATIC PC Station”站的“Configuration”，弹出硬件配置界面。

3）在“SIMATIC PC Station”硬件目录中，选择“OPC Server”（配置 OPC 服务器）或“Application”（配置与监控软件的通信连接）及上位机使用的通信处理器（例如 CP5611）插入到 PC 站的机架中（安装软件的版本与配置的版本必须匹配）。双击 CP5611，选择与 PLC 相同的 PROFIBUS 网络，并设置 CP5611 的站地址。如果使用 SIMATIC NET PC，需要复制 PLC 站的项目配置通信连接。

4）在“SIMATIC Manager”界面中，使用菜单命令“Options”→“Configuring Network”进入网络配置界面“NETPRO”。

5）点击 PC 站的“OPC Server”或“Application”，在下方出现连接表，双击连接表中任意一空白行弹出对话框，选择需要连接的对象（与之通信的 PLC 站），在“Type”中选择“S7 Connection”，点击“Apply”键确认，进入“S7 Connection”连接属性界面，如图 10-16 所示，在连接属性中列出连接 ID（连接名）、使用的网络、通信双方地址、使用的接口等参数。如果配置 OPC 服务器，若需要可以在“OPC”标签中更改 OPC 服务器的连接、优化等参数（通常使用默认设置），存盘编译。

6）在控制面板中设置 PG/PC 接口，设置“S7_ONLINE”访问点指向“PC internal”，设置“CP_L2_1”访问点指向“CP5611（PROFIBUS）”。

7）在桌面双击打开“Station Configurator”编辑器，点击“Station Name..”键，进入 PC 站的名称，该名称必须与配置的“SIMATIC PC Station”站名完全相同，否则不能下载配置信息。在“INDEX”栏中，按“SIMATIC PC Station”中配置的序号插入“OPC Server”或“Application”以及通信处理器 CP5611。

8）如果配置 OPC 服务器，在视窗界面点击“Start”→“Simatic”→“Simatic Net”→“Settings”→“Configuration Console”，进入 PC 站配置界面，在“Applications”→“Select OPC protocol”目录下，选择“S7”OPC Server。

9）在“NETPRO”界面中，将配置信息下载到所有与配置相关的站点中。如果在“Station Configurator”编辑器中，“OPC Server”或“Application”及通信处理器 CP5611 有故障，重启上位机，如果故障仍然存在，需要检查通信处理器 CP5611 及软件的安装。

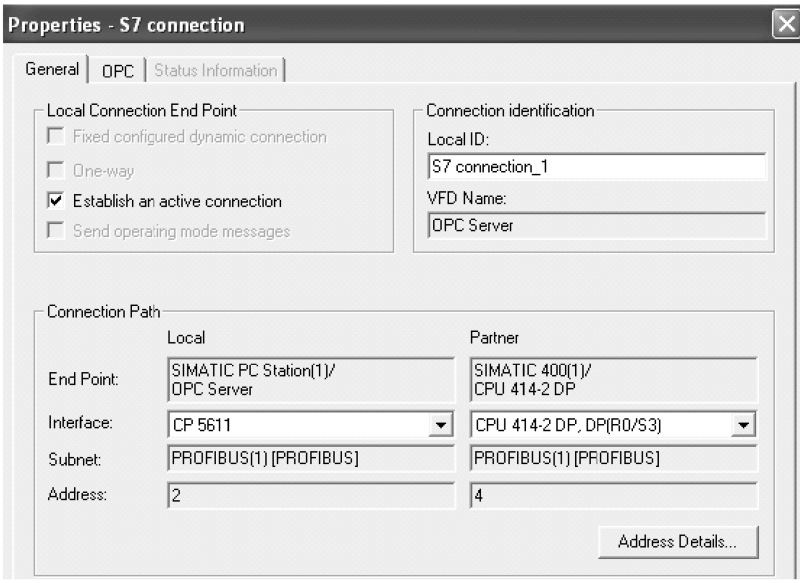



图 10-16　配置 PC 连接属性

10）下载配置信息后，在 OPC 客户端或监控软件创建通信连接时，选择已经配置的连接 ID，图 10-16 中的连接 ID 为“S7 connection_1”。

11）如果配置的是 OPC 服务器，可以通过 SIMATIC NET 自带的 OPC 客户端软件 OPC Scout（“Start”→“Simatic”→“Simatic Net”→“Settings”→“OPC Scout”）进行测试。

## 10.4　工业以太网

工业以太网应用于单元级、管理级的网络，通信数据量大、距离长。原有的工业以太网数据交换不能保证通信的实时性，基于工业以太网的 PROFINET（实时以太网）具有很好的实时性，可以直接连接现场设备（使用 PROFINET IO），使用组件化的设计，PROFINET 支持分布的自动化控制方式（使用 PROFINET CBA，相当于主站间的通信）。

### 10.4.1　工业以太网接口的种类

S7-300 系列 PLC：CPU31X-2DP/PN 集成 PROFINET 接口、CP343-1、CP343-1IT（集成 FTP、E_MAIL 功能）、CP343-1Lean（简化版本）、CP343-1 Advanced（集成 FTP、E_MAIL 功能，覆盖 CP343-1IT 功能）。

S7-400 系列 PLC：CPU41X-2DP/PN 集成 PROFINET 接口、CP443-1、CP443-1 Advanced（集成 FTP、E_MAIL 功能和 4 端口交换机）。

编程器或上位机：CP1613、CP1616/CP1604（支持 PROFINET IO，需要软件开发）、商用以太网卡。

### 10.4.2　工业以太网通信介质

西门子公司工业以太网使用双绞线、光纤和无线进行数据通信。

1. IE FC TP（Industry Fast Connection Twist Pair）

工业快速连接双绞线配合西门子公司的 FC TP RJ45 接头使用，连接如图 10-17 所示。将双绞线按照 TP RJ45 接头标示的颜色插入连接孔中，快捷、方便地将 DTE（数据终端设备）连接到工业以太网上。使用 FC 双绞线从 DTE 到 DTE、DTE 到交换机、交换机之间最长通信距离为 100m（通信速率为 100M，DTE 到 DTE 之间需要交叉连接，DTE 到交换机之间需要直通连接，西门子公司的交换机由于采用了自适应技术，可以自动检测线序，故通过交换机可以采用任意一种方式进行连接），主干网使用 IE FC 4X2 电缆可以达到 1000M。使用普通 RJ45 接头，不带有信号的屏蔽，为了保证数据传输的可靠性，最长通信距离为 10m。

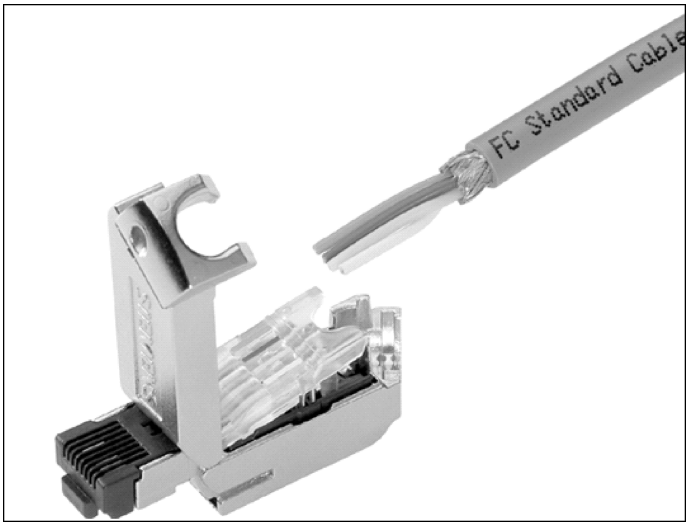



图 10-17　FC TP 电缆与 TP RJ45 接头

2. ITP（工业双绞线，Industry Twist Pair）

ITP 电缆预装配 9/15 针 SUB D 接头，连接通信处理器 CP 的 ITP 接口，适合恶劣的现场环境，ITP 电缆最长为 100m，逐渐被 IE FC TP 连接电缆替代。

3. 光纤

光纤适合于抗干扰、长距离的通信，西门子公司的交换机间或交换机与 DTE 之间可以使用 POF 光纤（最长 50m）、PCF 光纤（最长 100m）连接；使用多模光纤（西门子公司的交换机之间通信速率为 100M 时，最长距离为 3000m，通信速率为 1000M 时，最长距离为 750m）、单模模光纤（西门子公司的交换机之间通信速率为 100M 时，最长距离为 26km，通信速率为 1000M 时，最长距离为 10km）适合西门子公司的交换机之间的连接。

**注意：**

使用光纤连接时，接头必须匹配。

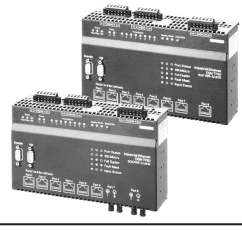
4. 无线以太网

使用无线以太网收发器相互连接，通信距离与通信标准及天线有关。

### 10.4.3　工业以太网络交换机

西门子公司的网络交换机适合不同的应用，型号与功能描述见表 10-4。

**表 10-4　交换机型号与功能描述**

| 交换机类型 | 图例 | 简要描述 |
|---|---|---|
| SCALANCE X 系列 | | |
| X005 |  | 价格便宜的入门级交换机，带有 5 个 TP 接口，面板集成 LED 诊断 |
| X100 子系列 |  | 工业现场应用的经济型交换机，系列中部分交换机带有光纤接口，但是不能组成环网。X100 系列交换机没有集成网络管理功能 |
| X200 子系列 |  | 工业现场功能扩展型交换机，系列中带有光纤接口，可以组成环网并带有冗余管理器功能。X200 系列交换机集成网络管理功能，支持 PROFINET 功能，带有 IRT 型号的交换机支持 PROFINET 等时操作 |
| X400 |  | 模块化高端交换机，可以集成多个连接不同通信介质的接口，支持 1000M 通信速率，支持 VLAN 和 IP Routing |
| SCALANCE W 无线以太网系列 | | |
| SCALANCE W |  | 支持 IEEE 802.11b、g、a 通信标准。W744 为客户端，W788 既可以作为访问点也可以作为客户端。通信距离与通信标准及天线有关 |
| OSM/ESM | | |
| OSM 与 ESM |  | 早先使用的交换机，逐渐被 SCALANCE X 系列替代 |

## 10.4.4　工业以太网拓扑结构

使用西门子公司的工业交换机可以组成总线型、树形、环形网络拓扑结构。环形网络拓扑结构是总线型网络的一个特例，将总线型的头尾两段连接形成环形网络结构。环形网络可

以使用光纤和双绞线构成，在环形网络中必须有一个交换机作为冗余管理器，例如 X400、OSM 或 X204-2IRT 都可以作为冗余管理器，环形网络中的每一个交换机都能够通过冗余检测报文，所以只有 X200 以上交换机支持环形网络。由 X200 与 X400 交换机组成的环形网络如图 10-18 所示。

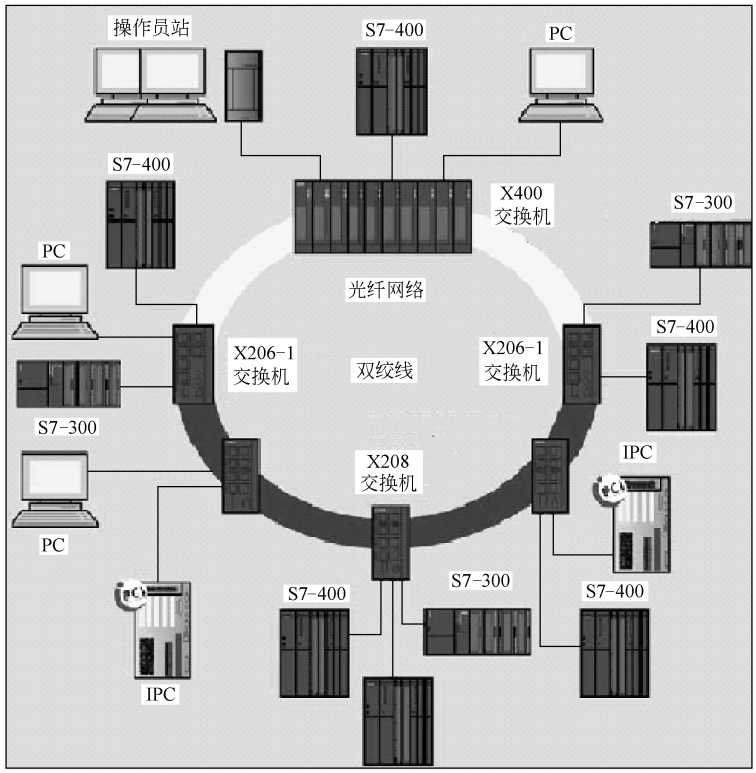



图 10-18　环形拓扑结构

无线以太网上有 AP（访问点）和客户端设备，连接的方式有两种："Ad-hoc"模式和"Infrastructure"模式，在"Ad-hoc"通信模式中，通过移动的客户端设备级联形成网络；在"Infrastructure"通信模式中，所有的客户端通过 AP 访问网络上的设备，如图 10-19 所示。

图 10-19 中每个客户端是一个自动导向小车，小车上的设备 HMI、PC 及 S7-300 系列 PLC 通过客户端模块 W746-1 与 S7-400 系列 PLC 和 SIMATIC HMI 站通信。AP 与客户端形成一个无线网络单元，相互独立。如果多个 AP 通过电气网络连接，设置的无线网络 ID（SSID）相同，则小车可以在所有 AP 覆盖的无线网络中漫游。在实际应用中，AP 之间的连接通常也使用无线连接，这样形成 WDS（Wireless Distribution System）网络，如图10-20 所示，AP W788-2PRO 带有两个无线网络，一个可以连接客户端，另一个可以连接其他的 AP，AP 间点到点连接需要在配置界面进行桥接。

SCALANCE W 系列中绝大多数模块支持 WEB 配置模式，可以直接通过 PC 的浏览器对模块网络 ID、桥接、网络加密等参数进行配置以及对模块进行诊断。

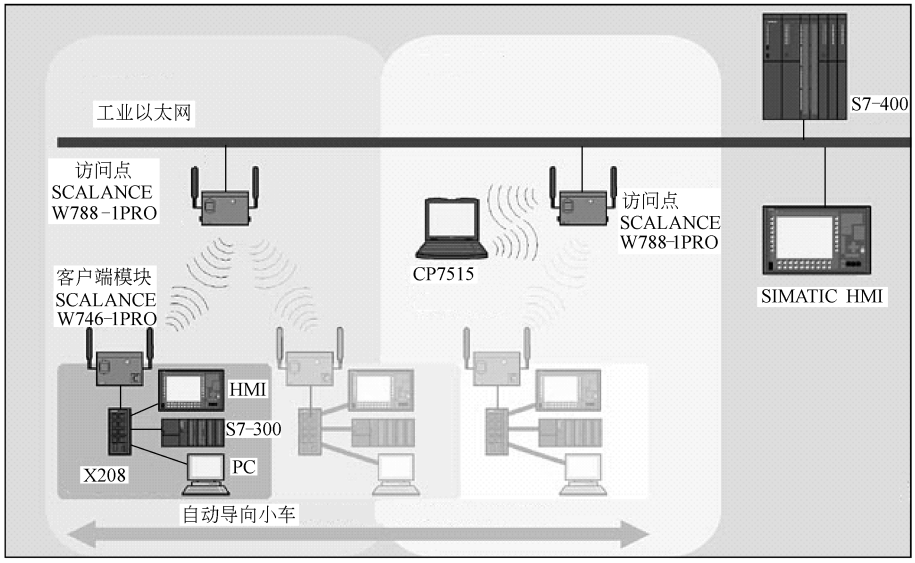



图 10-19　AP 通过电气网络连接

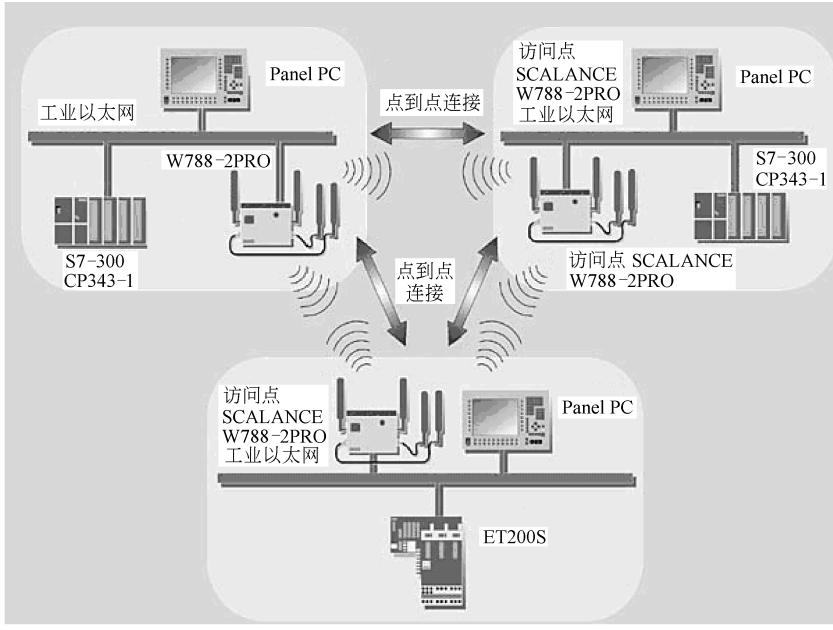



图 10-20　WDS 网络连接

## 10.4.5　工业以太网支持的通信协议与服务

西门子公司的工业以太网支持不同的通信协议和服务适用不同的现场应用：

1．S5 兼容通信（SEND/RECEIVE 通信服务）

开放的 S5 兼容通信适用于 S7（除 S7-200 系列 PLC）PLC 之间通信、S7 PLC 与 S5 PLC

间的通信。S5 兼容通信有下列通信服务：

1）ISO Transport——该通信服务支持第四层（ISO Transport）开放的数据通信 。用于 SIMATIC S7 和 SIMATIC S5 的工业以太网的连接。S7 PLC 间的通信也可以使用 ISO 通信方式，但是有些新的 S7-300 系列 PLC 通信处理器不再支持该通信服务。

2）ISO-on-TCP——由于 ISO 不支持以太网路由，因而西门子公司应用 RFC1006 将 ISO 通信服务映射到 TCP 上，实现网络路由。

3）UDP——该通信服务属于第四层协议，支持简单数据传输，数据无须确认，适用于用户自定义的报文格式。上位机可以通过 VB、VC SOCKET 控件读写 PLC 数据。

4）TCP/IP——该通信服务支持 TCP/IP 开放的数据通信。用于连接 SIMATIC S7 和 PC 以及非西门子公司设备。上位机可以通过 VB、VC SOCKET 控件读写 PLC 数据。

2．S7 通信服务

特别适用于 S7 PLC 与 HMI（PC）和编程器之间的通信。也可以应用于 S7-300 系列 PLC 与 S7-400 系列 PLC、S7-400 系列 PLC 之间通信。S7 服务使用了 ISO/OSI 网络模型第七层通信协议，独立于使用的网络，在 MPI、PROFIBUS 和以太网网络中都可以使用。

3．PG/OP 通信服务

编程器和操作面板可以通过以太网对网络上的站点进行访问。

4．PROFINET 服务

基于工业以太网网络的实时以太网 PROFINET 分为两种通信服务：PROFINET IO 和 PROFINET CBA。

（1）PROFINET IO

PROFINET IO 适合模块化、分布式的应用，现场设备（IO-Devices）可以直接连接到以太网上，通过 SIMATIC Manager，将现场设备分配到一个 IO 控制器（IO-Controller）上。PROFINET IO 提供三种执行水平：

1）非实时（Non-Real-Time，NRT）数据传输：用于项目的监控和非实时要求的数据传输，例如项目诊断等功能，典型通信时间大约为 100ms。

2）实时（Real-Time，RT）通信：用于要求实时通信的过程数据，通过提高报文的优先级和优化数据堆栈（ISO/OSI 模型第一层和第二层）实现实时通信，使用标准网络元件可以执行高性能的数据传输，典型通信时间为 1 ~ 10ms。

3）等时实时（Isochrons-Real-Time，IRT）：等时实时确保数据在相等的时间间隔进行传输，例如多轴同步操作，普通交换机不支持等时实时通信。等时实时典型通信时间为 0.5 ~ 1ms，每次传输的时间偏差小于 1μs。

支持 IRT 的交换机数据通道分为标准通道和 IRT 通道，标准通道用于 NRT 和 RT 的数据通信，IRT 通道专用于 IRT 的数据通信，网络上其他的通信不会影响 IRT 过程数据的通信，PROFINET IO 实时通信的 OSI/ISO 模型如图 10-21 所示。

PROFINET IO 具有下列特点：

1）现场设备（IO-Devices）通过 GSD 文件的方式集成到 SIMATIC Manager 中，与 PROFIBUS-DP 不同的是，PROFINET IO 的 GSD 文件以 XML 格式存在。

2）为了保护原有的投资，PROFINET IO 控制器可以通过 IE/PB LINK 连接 PROFIBUS-DP 从站。

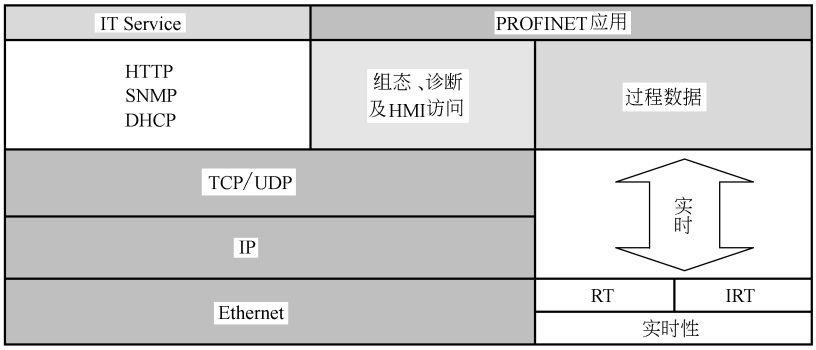



图 10-21　PROFINET 数据访问 OSI/ISO 模型

PROFINET IO 与 PROFIBUS-DP 的通信方式相似，术语的比较见表 10-5。

**表 10-5　PROFINET IO 与 PROFIBUS-DP 术语的比较**

| 数量 | PROFINET | PROFIBUS | 解释 |
|---|---|---|---|
| 1 | IO system | DP master system | 网络系统 |
| 2 | IO controller | DP master | 控制器与 DP 主站 |
| 3 | IO supervisor | PG/PC 2 类主站 | 调试与诊断 |
| 4 | 工业以太网 | PROFIBUS | 网络结构 |
| 5 | HMI | HMI | 监控与操作 |
| 6 | IO device | DP slave | 分布的现场元件分配到 IO controller/ DP master |

（2）PROFINET CBA

PROFINET CBA（Component Based Automation）支持基于机器、设备分布式模块化的自动化解决方案。将分布的自动化设备封装为一个组件，组件中包含设备执行的程序及与其他组件的通信接口。

## 10.4.6　配置 S5 兼容通信

通过调用发送和接收函数进行 PLC 站点间的通信，每个通信数据包最大为 8KB，两个站点间可以发送和接收多个数据包。只有以太网通信处理器 CP 支持 S5 兼容通信（S/R），每一个通信处理器可以同时与多个站点建立通信连接，例如 CP343-1 的连接数为 16 个，其他通信处理器的连接个数参考样本手册。

S5 兼容通信不占用 CPU 的“S7 Connection”通信资源（通信数据大于 240 字节时占用 S7-300 系列 PLC CPU 的连接资源），是 S7-300、400 系列 PLC 常用的通信方式。下面以两个 S7-300 系列 PLC 在相同的项目中为例，介绍实现“TCP”通信的步骤：

1）在一个项目下，建立两个 S7-300 系列 PLC 站。分别进行硬件配置，插入 CPU 和 CP343-1，创建以太网网络，分配 IP 地址，使两个站点在相同的网络上。

2）在 SIMATIC Manager 界面中，使用菜单命令“Options”→“Configuring Network”，

进入网络配置界面“NETPRO”。

3）点击其中一个 S7-300 系列 PLC CPU 后在下方出现连接表，双击连接表中任意一空白行弹出对话框，选择需要连接的对象（与之通信的 S7-300 系列 PLC 站），在“Type”中选择连接类型为“TCP”，点击“Apply”键确认，进入连接属性界面，如图 10-22 所示。在连接属性中自动列出连接 ID、连接名和块参数（编程时需要使用）。

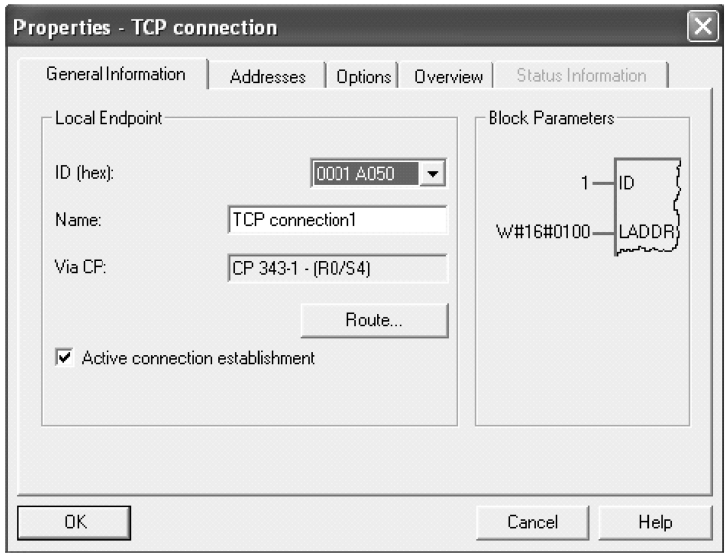



图 10-22 TCP 连接属性界面

4）在被连接 CPU 的连接表中自动生成一个连接，该连接同样表示本地 CPU 连接的信息。双击打开连接，记录本方的块参数，用于编程时使用。

5）在一个 CPU OB35（间隔发送）中调用发送函数 FC5 AG_SEND（“Libraries”→“Standard Library”→“SIMATIC_NET_CP”→“CP300”，如果是 S7-400 系列 PLC CPU，在“CP400”目录下），示例程序如下：

```
CALL" AG _SEND"
     ACT   := TRUE
     ID    := 1
     LADDR := W#16#100
     SEND  := P#DB1. DBX 0. 0 BYTE 60
     LEN   := 60
     DONE  := M1. 2
     ERROR := M1. 3
     STATUS:= MW2
```

通信函数 FC5 的参数含义：

ACT　　：沿触发信号。

ID　　：参考本地 CPU 连接表中的块参数。

LADDR　：参考本地 CPU 连接表中的块参数。
SEND　：发送区，最大通信数据为 8KB。
LEN　：实际发送数据长度。
DONE　：每次发送成功，产生一个上升沿。
ERROR　：错误位。
STATUS　：通信状态字。
示例程序中 S7-300 系列 PLC 发送 DB1 中前 60B。

6）在通信方 CPU OB1 中调用接收函数 FC6 AG_RECV（“Libraries”→“Standard Library”→“SIMATIC_NET_CP”→“CP300”，如果是 S7-400 系列 PLC CPU，在“CP400”目录下），示例程序如下：

```
CALL "AG_RECV"
ID    := 1
LADDR := W#16#100
RECV  := P#DB2.DBX 0.0 BYTE 60
NDR   := M10.1
ERROR := M10.2
STATUS:= MW12
LEN   := MW14
```

通信函数 FC6 的参数含义：
ID　：参考本地 CPU 连接表中的块参数。
LADDR　：参考本地 CPU 连接表中的块参数。
RECV　：接收区。接收区应大于等于发送区。
NDR　：每次接收到新数据，产生一个上升沿。
ERROR　：错误位。
STATUS　：通信状态字。
LEN　：实际接收数据长度。

将配置信息和通信程序下载到相应的 CPU 中，通信建立。示例程序中 S7-300 系列 PLC 将接收的数据存储于本地数据区 DB2 的前 60 字节中。两个站点通过一个连接可以同时进行发送和接收任务。

7）如果通信不能建立，双击 CP 模块，在属性对话框中选择“Diagnostics”标签，联机后点击“Run..”键，进入诊断界面，可以诊断连接状态、发送成功的数据包和实际接收的数据包。诊断功能在这里不作详细介绍。

在实际应用中往往通信的站点不在同一项目下，由于程序保密不可能复制对方程序，这样就需要通信双方相互提交通信长度、IP 地址、端口号等信息。下面以两个 S7-300 系列 PLC 在不同的项目中为例，介绍实现“TCP”通信的步骤：

1）在项目下，建立 S7-300 系列 PLC 站。进行硬件配置，插入 CPU 和 CP343-1，创建以太网网络，按双方协议分配 IP 地址。

2）在“SIMATIC Manager”界面中，使用菜单命令“Options”→“Configuring Network”，进入网络配置界面“NETPRO”。

3）点击 S7-300 系列 PLC CPU 后，在下方出现连接表，双击连接表中任意一空白行弹出对话框，选择连接的对象为“Unspecified”，在“Type”中选择连接类型为“TCP”，点击“Apply”键确认，进入连接属性界面，如图 10-23 所示。在“General Information”标签中，自动列出连接 ID、连接名和块参数（编程时需要使用），两个站在配置中必须且只能有一个站激活“Active connection establishment”选项。

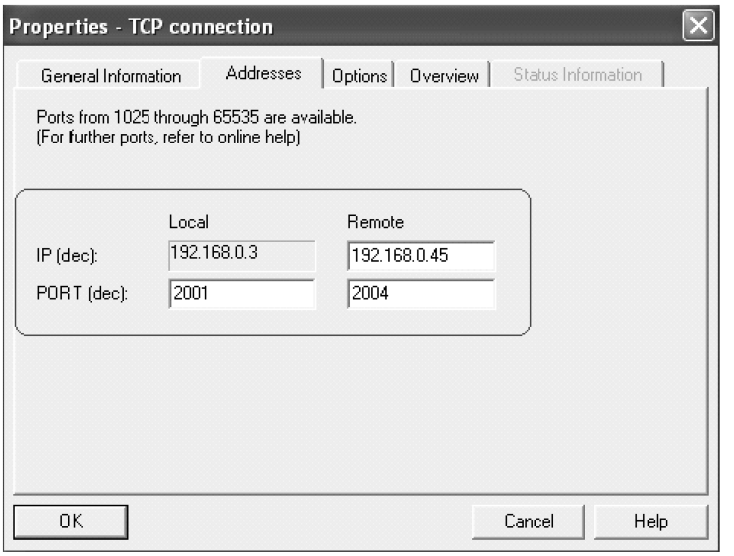



图 10-23 TCP 连接属性“Addresses”标签

4）在“Addresses”标签中，按双方协议设置本方的端口号和对方的 IP 地址和端口号，建议使用 2000 以上的端口号。如果选择“ISO”或“ISO_ON_TCP”通信方式，则需要设置 TASP（自由设置，ASCII 码格式）。

5）按相同的步骤配置通信方。

6）与在相同项目下建立的 TCP 连接的通信程序相同，将配置信息和通信程序下载到 CPU 中，通信建立。

**注意：**

通信函数 FC50（AG_LSEND）和 FC60（AG_LRECV）适合 S7-400 系列 PLC 的接收、发送超过 240 字节的报文。S7-400 系列 PLC 建立的 TCP 连接必须调用 FC50 和 FC60 作为通信函数。

### 10.4.7 配置 S7 通信连接

CPU 集成的以太网接口或通信处理器 CP 都支持 S7 通信连接，CPU 资源的占用、通信配置及调用的通信函数参考 MPI 和 PROFIBUS S7 通信连接，方式和方法相同，只是使用的网络不同。

### 10.4.8 PLC 通过以太网与 HMI 通信

（1）与操作面板的连接

选择支持以太网通信功能的 HMI 操作面板，使用 WinCC FLEXIBLE 编程软件中直接定

义连接 IP 地址及 CPU 的槽号（即使连接通信处理器 CP）即可建立通信。

（2）与 HMI WinCC 的连接

与 HMI WinCC 连接，添加“S7 Protocol Suite”驱动，选择连接的通信驱动，例如“TCP/IP”，按鼠标右键，在弹出的连接属性对话框中选择上位机使用的通信处理器。在“ETHERNET”（使用 MAC 地址）或“TCP/IP”（使用 IP 地址）驱动中建立与 PLC 连接时选择站地址及 CPU 在机架上的槽号（即使连接通信处理器 CP）即可建立通信。

（3）建立 OPC 服务器或与其他监控软件的连接

参考 PROFIBUS 配置过程。

### 10.4.9　使用 DCP 直接配置以太网接口

通过 DCP（Detect Configuration Protocol）可以直接对以太网接口，如 CP343-1、CP443-1 及 CPU 集成的以太网接口分配 IP 地址，这样在项目的初始阶段可以直接使用 PC 的以太网网卡与 PLC 进行通信，而不再需要 MPI 适配器或通信处理器对 PLC 进行初始化（例如分配以太网 IP 地址等）。西门子公司为每一个以太网接口都分配了一个 MAC 地址（出厂缺省设置，可以修改），通过 MAC 地址可以为该接口分配 IP 地址，配置步骤如下：

1）在控制面板“PG/PC”接口中设置访问点指向本机的以太网网卡，网卡选择 ISO 或 TCP（Auto），使以太网网卡作为联机网卡。

2）在 SIMATIC Manage 界面中，使用菜单命令“PLC”→“Assign Ethernet Address”，根据硬件惟一的 MAC 地址浏览网络上所有的站点，如图 10-24 所示。

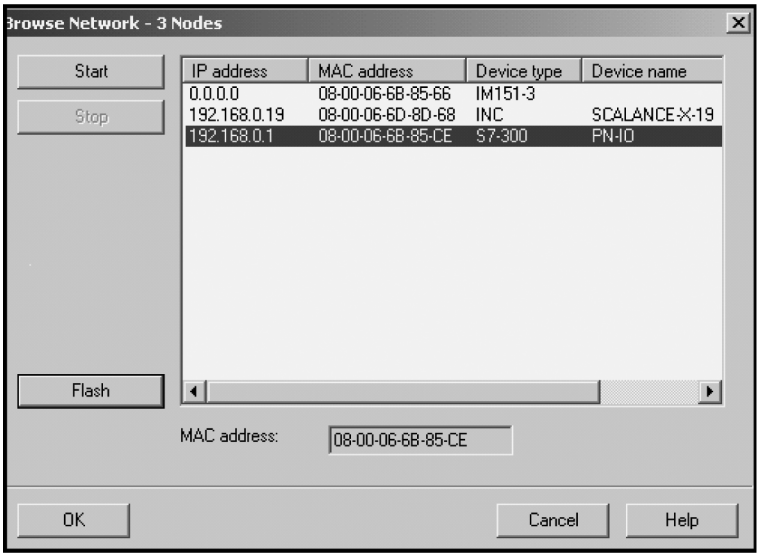



图 10-24　浏览网络站点

点击“Start”按钮，浏览网络上所有西门子公司以太网站点，如 IO 控制器、IE/PB LINK IO、交换机等，如果不能确认 MAC 地址所对应的站点，先点击“MAC”地址栏，然后点击“Flash”按钮，对应的以太网接口指示灯将闪烁，可以方便地识别网络上的站点，避免造成站点混乱（有些接口不支持闪烁功能）。

3）选择其中一个站点分配 IP 地址和子网掩码。

4）如果站点的 MAC 地址被修改，可以点击“Reset”恢复工厂设置。

## 10.5 PROFINET 通信

### 10.5.1 PROFINET IO 通信

S7-300 系列 PLC 中的 CPU 31X-2 PN/DP、CPU319-3PN/DP、集成 PN 接口的通信处理器 CP343-1；S7-400 系列 PLC 中的 CPU41X-3PN/DP、集成 PN 接口通信处理器 CP443-1，可以连接带有 PROFINET IO 接口的远程 I/O 站点，例如 ET200M、ET200S 等。

在 STEP 7 硬件配置中，以 CPU315-2PN/DP 为例创建一个站点，插入 CPU 时弹出新建以太网的提示对话框，选择新建以太网网络并设置 CPU 的 IP 地址（CPU 的前面板标有 MAC 地址，在 SIMATIC Manager 界面下点击菜单命令“PLC”→“Edit Ethernet Nodes”可以分配 IP 地址或通过 MPI 下载配置信息设置 IP 地址）、子网掩码（255.255.255.0），随后出现一条代表 PROFINET IO 总线的网络，如图 10-25 所示。

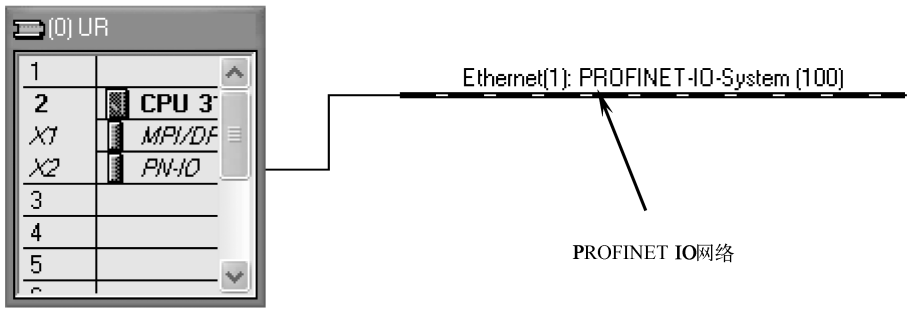



图 10-25 设置 PROFINET IO 网络

在硬件目录中的 PROFINET IO 子目录下，将所需的设备拖放到总线上，例如配置 ET200S、交换机 X208 和网关设备，如图 10-26 所示。使用 IE/PB LINK PN IO 网关（以太网/PROFIBUS-DP）可以从实时以太网 PROFINET IO 上直接连接 PROFIBUS-DP 从站。

点击每个从站，为从站配置相应的 I/O 模块，模块的 I/O 地址占用 I、Q 区，与中央扩展寻址方式相同。

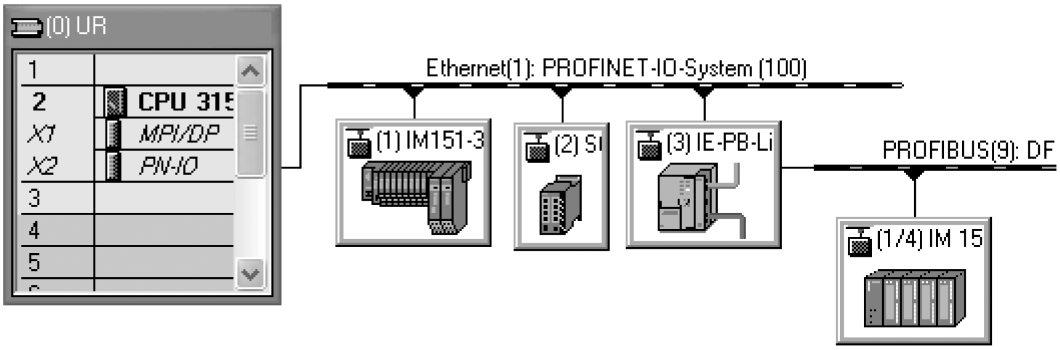



图 10-26 配置 PROFINET IO 远程 I/O

双击远程 I/O 站图标，为每个站点分配地址信息，如图 10-27 所示。在“Device Name”下键入设备名，每个远程站点的设备名是惟一的，PROFINET IO 控制器通过设备名自动分配远程站点 IP 地址。在“Device Number”中定义设备号，每个设备号是惟一的。点击“Ethernet”可为远程 I/O 站分配 IP 地址，IP 地址在每个网络上是惟一的。地址信息分配完成后编译存盘。

将编程器通过以太网网卡连接到实际的 PROFINET IO 网络上，在硬件配置界面使用菜单命令“PLC”→“Ethernet”→“Assign Device Name”，以每个远程 I/O 站的 MAC 地址为参考分配设备名称。通过菜单命令“PLC”→“Ethernet”→“Verify Device Name”检查设备名称是否分配成功。将配置信息下载到 CPU 中，CPU 在启动过程中自动为每个远程I/O站分配 IP 地址。实时通信通过通信堆栈的物理层和应用层实现。

在“IO Cycle”标签中设置远程 I/O 站数据的更新时间，更新时间与远程 I/O 站点的个数、通信数据量的大小有关，例如 CPU（IO 控制器）连接远程 I/O 站点的数量在 64 个以内，可以设置的更新时间为 1ms。使用无线网络或在多个交换机级联的情况下应增加更新时间。

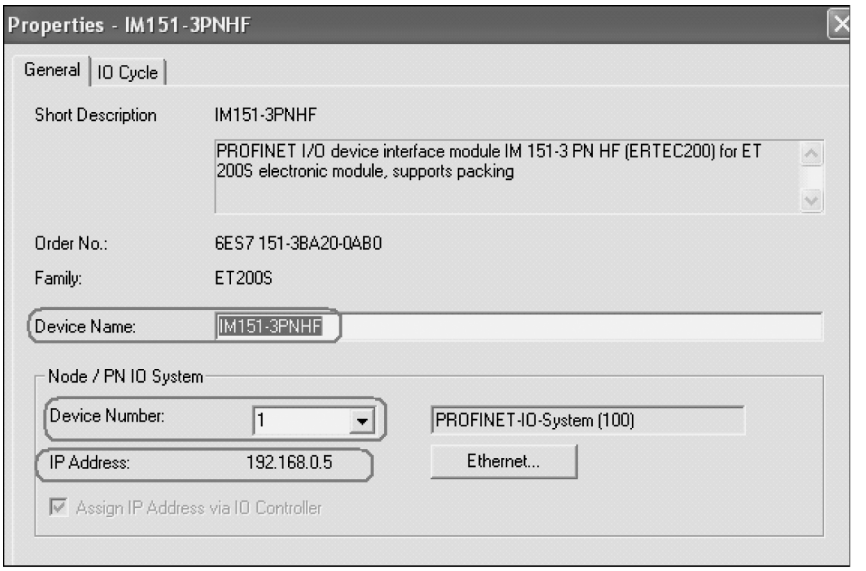



图 10-27　远程 I/O 的配置

## 10.5.2　PROFINET IO 快速启动

PROFINET IO 的“快速启动”功能可实现在 PROFINET IO 应用中，部件或工具及其 IO 设备（分布式 I/O）可随工艺要求进行更换。通过快速启动功能，重新启动和预定进程之间的等待时间（几秒钟）将缩短为最小。这加速了具有可移动 IO 设备（分布式 I/O）的生产进程，并能大幅提高生产能力。参见图 10-28 所示的机器人机械手臂。

快速启动是指在具有 RT 和 IRT 通信的 PROFINET IO 中用于加速 IO 设备（分布式I/O）启动速度的 PROFINET 功能。它缩短了相应组态的 IO 设备（分布式 I/O）所需要的时间，以便实现下列情况中快速的循环用户数据交换：

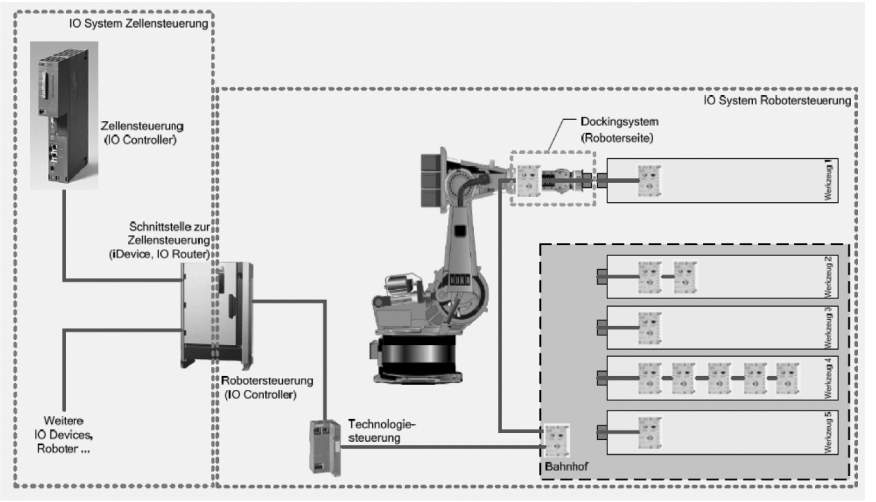



图 10-28 机器人机械手臂

- 设备电源恢复后；
- 该站已经返回后；
- 激活分布式 I/O 设备。

通过 PROFINET IO 快速启动可以缩短分布式 I/O 的通信准备就绪时间，最小到 500ms。并且该功能适用于 RT 或 IRT。PROFINET IO 快速启动的时间长短依赖于以下几点：

- 所使用的分布式 I/O 设备；
- 分布式 I/O 设备的 IO 结构；
- 分布式 I/O 设备所用的模块；
- 所使用的 IO 控制器；
- 所使用的交换机；
- 端口设置；
- 电缆。

快速启动要求禁止端口自协商和自交叉模式，并优化了 IO 的连接建立过程。在 PROFINET IO 系统中实现快速启动功能，首先要求分布式 IO 设备支持快速启动功能，例如 ET200S PN 从 4.0 版本开始即支持该功能。IO 控制器也要支持该功能，CPU 从 Firmware 版本 2.6 开始即支持快速启动。一个 PROFINET IO 系统中最多可以组态 32 个快速启动设备，但是最多只能一次激活 8 个 IO 设备实现快速启动。

对于快速启动的设备以及相邻设备的相连端口，必须禁止自协商和自交叉模式，然后根据所使用的设备选择交叉或平行接的网线。对于连接相同类型的设备端口，应该使用交叉接的网线，例如交换机之间或终端设备之间。参考图 10-29 所示的相同类型设备之间的使用交叉接的网线。而对于不同类型的设备则使用平行接的网线，例如交换机和终端设备之间。参考图 10-30 所示的不同类型设备之间的使用交叉接的网线。新的分布式 IO 设备，例如 ET200S 的 6ES7 151-3BA23-0AB0 以及后续版本，其中第一个端口作为终端设备端口，这样连接不再需要交叉接的电缆而只需要平行接的电缆。参考图 10-31 所示的 ET200S 使用平行接网线串联。

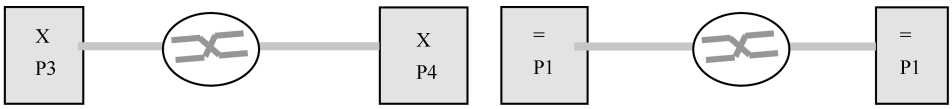



图 10-29　相同类型设备之间的使用交叉接的网线

如果希望实现 500ms 的最短启动时间，必须执行下列操作：

- 在 STEP7 中组态 PROFINET IO 的快速启动功能；
- IO 设备（分布式 I/O）上的端口设置；
- 布线取决于互连的 PROFINET 设备；
- 用户程序中的操作，需要组态 Docking 站、Docking 单元和编写 SFC12。

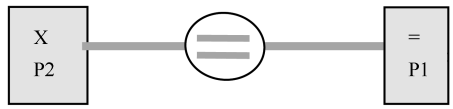



图 10-30　不同类型设备之间的使用交叉接的网线

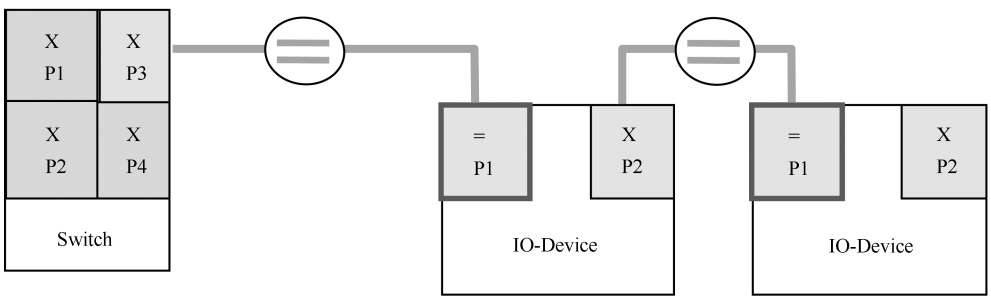



图 10-31　ET200S 使用平行接网线串联

PROFINET IO 系统中，CPU319-3PN/DP 作为 IO 控制器，连接 SCALANCE X400 交换机，机器人手臂连接切换的两个工具都为 ET200S。例子参考图 10-32 所示的快速启动例子。其中通过 SCALANCE X414-3E 的 9.1 端口使用平行接的网线连接两台相互切换 ET200S 的端口 1。

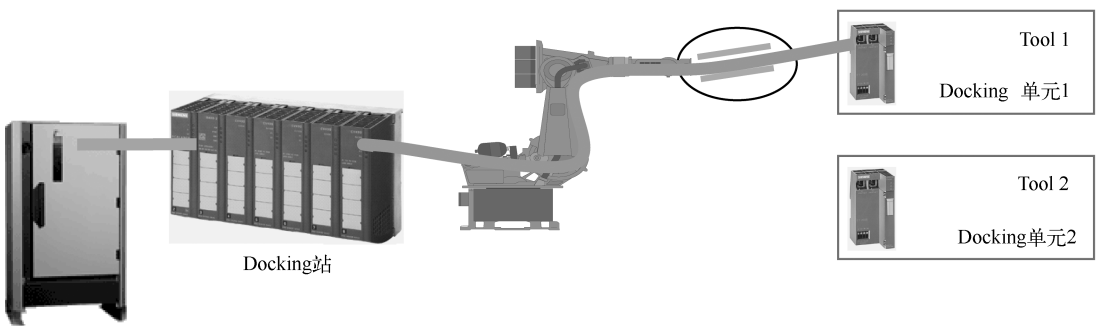



图 10-32　快速启动例子

根据实际网络设备，在 STEP7 中进行硬件组态，如图 10-33 所示。

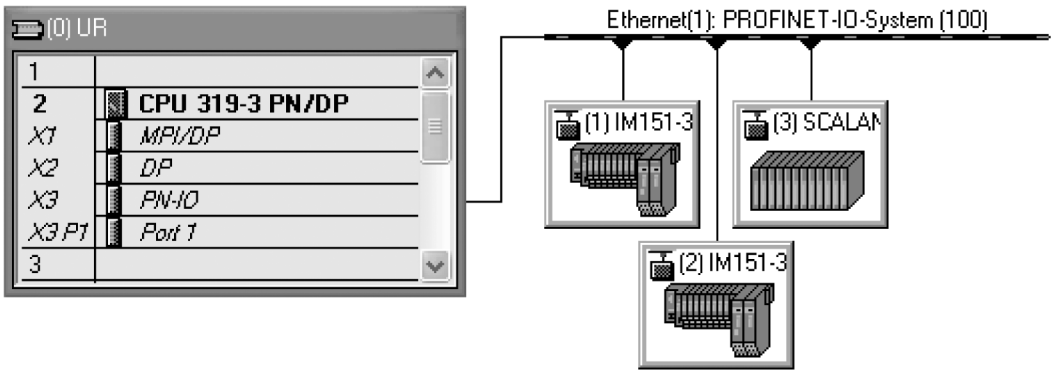



图 10-33　硬件组态

在 HW 组态画面中，点击 PROFINET 1 号站 IM151-3，然后双击“0 号槽的 X1 子槽的 PN-IO”，弹出“PN-IO”的属性对话框。使能“Prioritized startup”。点击“OK”，关闭该对话框。以同样的方式，给 2 号站 ET200S IM151-3 使能快速启动，如图 10-34 所示。

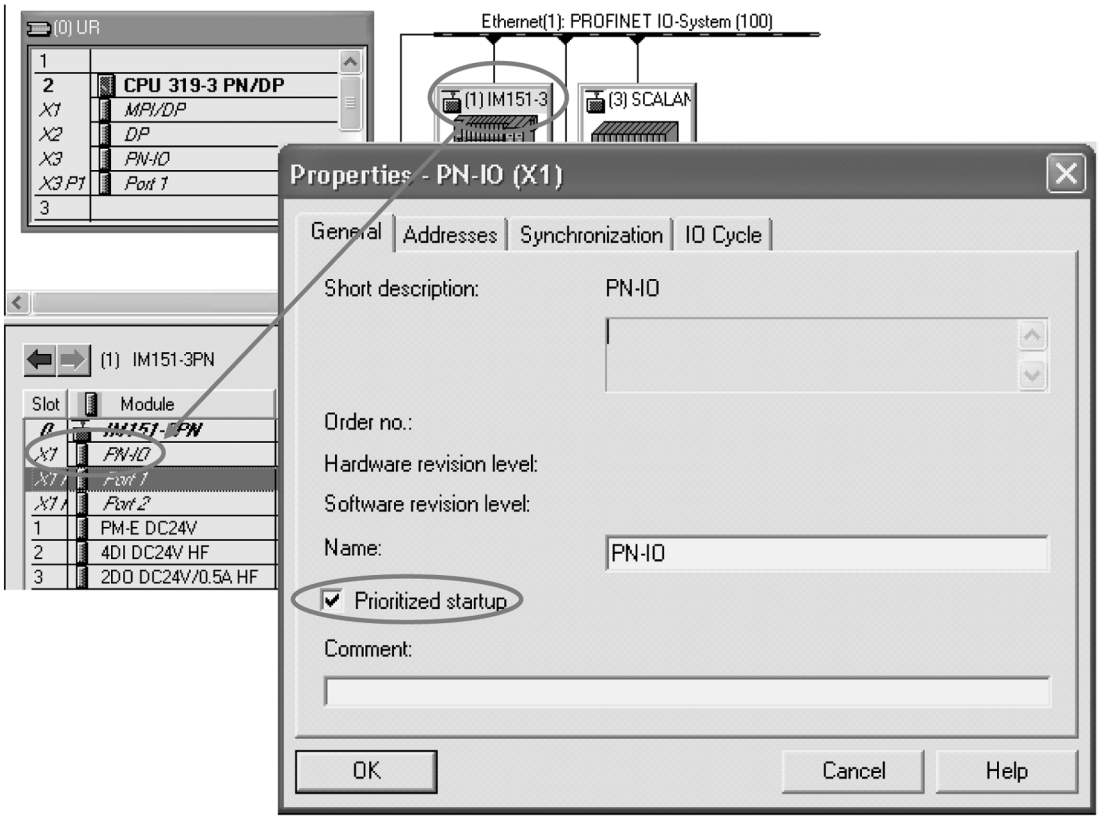



图 10-34　两台 ET200S 使能快速启动

在 HW 组态画面中，点击“PROFINET 1 号站 IM151-3”，然后双击“0 号槽的 X1P1 子槽的 Port1”，弹出“PN-IO-Port1”的属性对话框。选择传输介质和双工模式为“TP/ITP

100Mbps full duplex”，并使能“Disable autonegotiation”，这样就禁止了自协商和自交叉功能。点击“OK”，关闭该对话框。以同样的方式，给 2 号站的 ET200S IM151-3 的端口 1 设置属性，如图 10-35 所示。

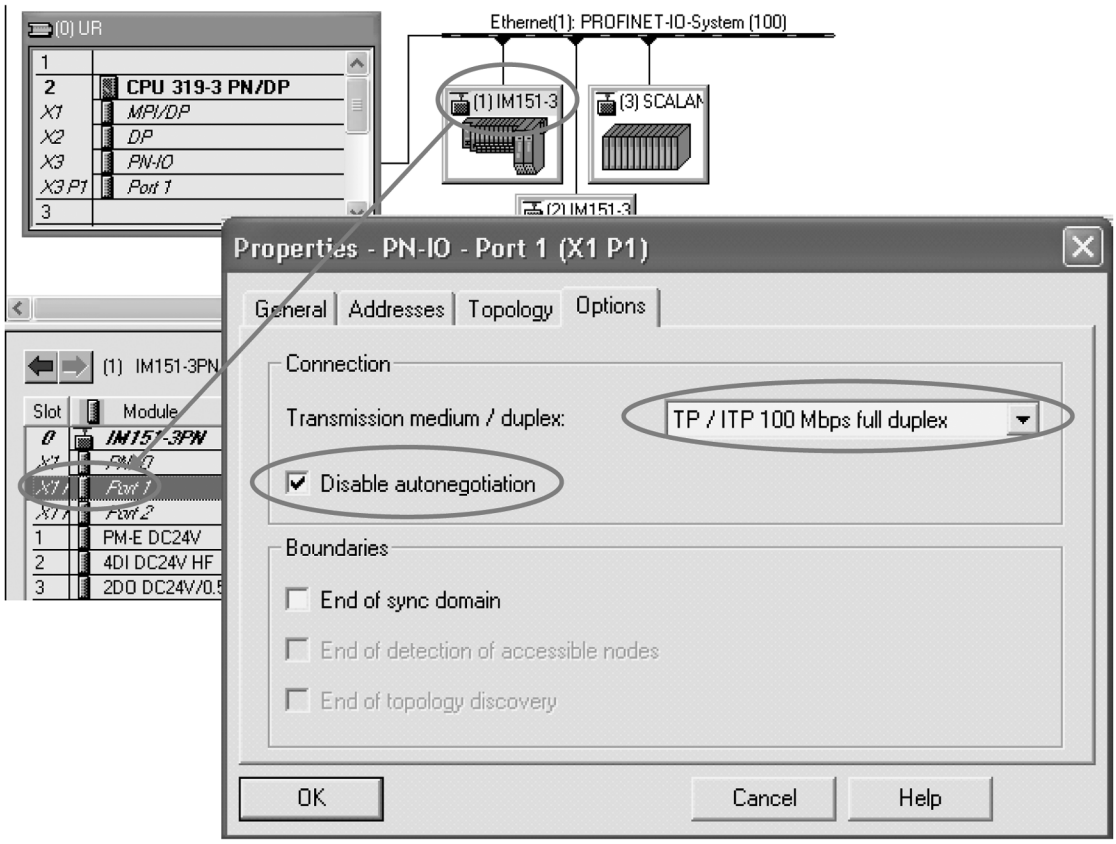



图 10-35　设置两台 ET200S 的端口 1 属性

对 Docking 站进行组态。在 HW 组态画面中，点击“PROFINET 3 号站 SCALANCE X414-3E”，然后双击“9 号槽的 X1P1 子槽的 Port1”，弹出“PN-IO-Port1”的属性对话框。选择传输介质和双工模式为“TP/ITP 100Mbps full duplex”，并使能“Disable autonegotiation”，这样就禁止了自协商和自交叉功能。点击“OK”，关闭该对话框。设置 SCALANCE X414-3E 的端口属性如图 10-36 所示。

然后选择 SCALANCE X414-3E 的端口 9 的 PN-IO-Port1 的属性对话框的 Topology 页面，在 Partner port 选项中，选择“Alternating partner port”，如图 10-37 所示。

然后在该对话框中，通过点击“Add…”按钮，分别加入切换工具的两个 ET200S 的端口 1。如图 10-38 所示。点击“OK”结束，这样就完成了 Docking 站的组态。

查看组态后的拓扑组态信息。使用鼠标右键点击 PROFINET 总线 Ethernet（1）：PROFINET-IO-System（100），选择“PROFINET IO Topology…”，如图 10-39 所示。

弹出拓扑编辑器对话框，选择“Graphic view”可以看见在 PROFINET IO 系统中的 IO 设备的连接关系，如图 10-40 所示。

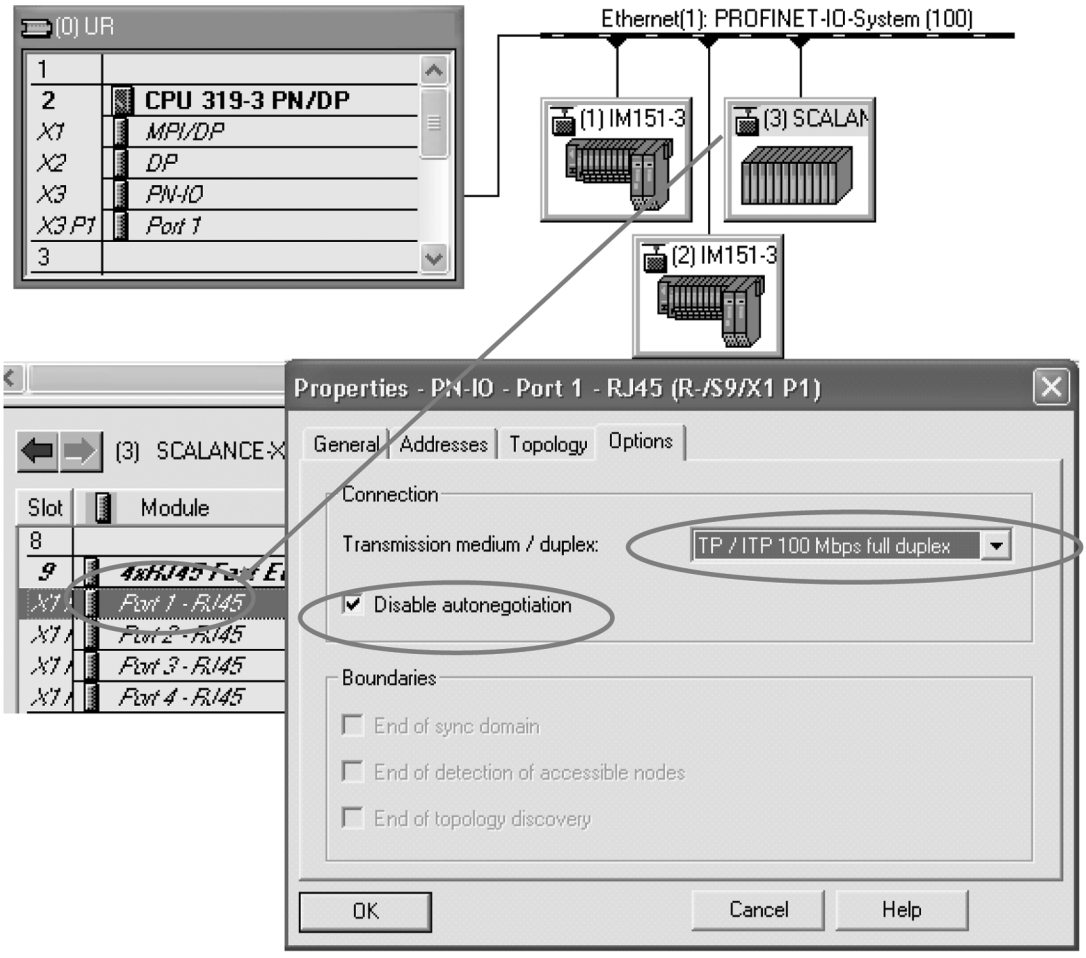



图 10-36 设置 SCALANCE X414-3E 的端口属性

对于 PROFINET IO 组态通信，分配 Device name 等，请参考 10.5.1 节。

打开“OB1”，在 OB1 的起始位置编写 SFC12，用于激活和禁止 IO 设备。其中 LADDER 为十六进制 ET200S 的诊断地址。编写 SFC12 如图 10-41 所示。然后，保存下载 OB1 到 PLC 中。

这样两个工具之间就可以实现切换，当一个设备被禁止时，另外一个设备被激活。需要注意的是，每次执行 SFC12 不同的 Mode 任务时（例如 Mode = 1，激活 IO 设备；mode = 2，禁止 IO 设备），REQ 必须重新置 1。当使用 Mode = 3 或者 4 时，需要在 PLC 中下载 OB86，否则工具在切换过程中 PLC 会停止。当通过 CP343-1 实现快速启动时，不能使用 SFC12。

### 10.5.3 PROFINET IO 网络拓扑

从 STEP7 V5.4 SP2 开始支持网络拓扑组态，可以在拓扑上组态 PROFINET IO 系统。有了拓扑组态，可以使用 PROFINET 的一些功能，例如“在操作期间更改 IO 设备”或“设备

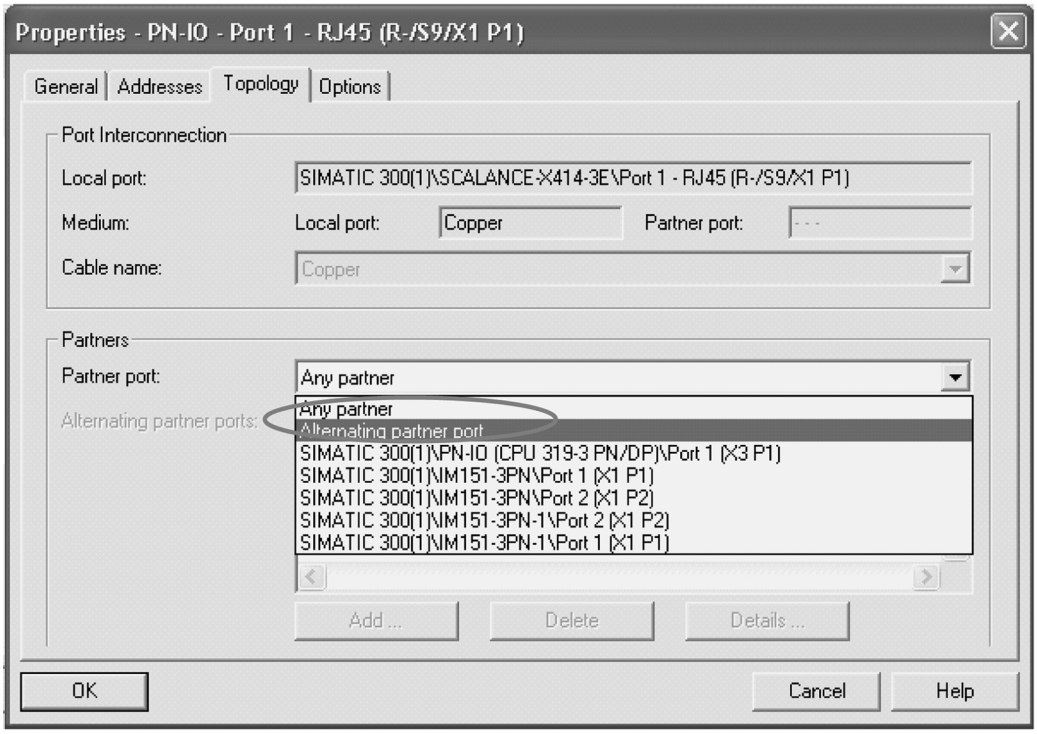



图 10-37　选择“Alternating partner port”

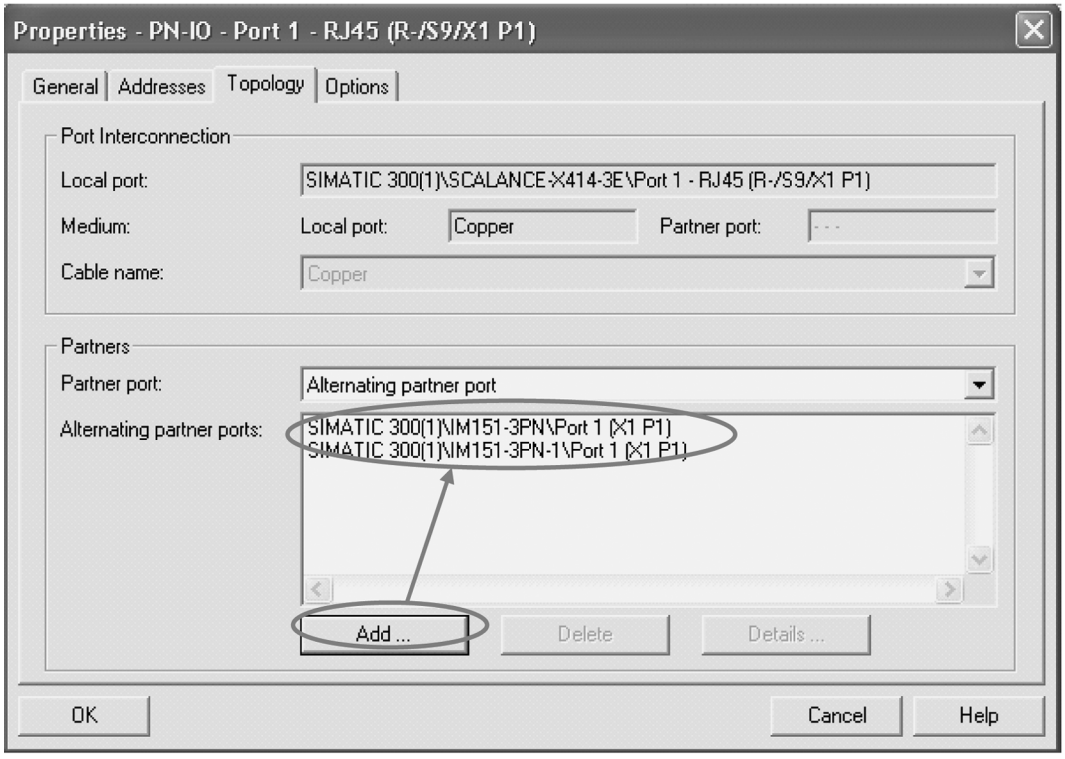



图 10-38　加入替换的 ET200S 端口 1

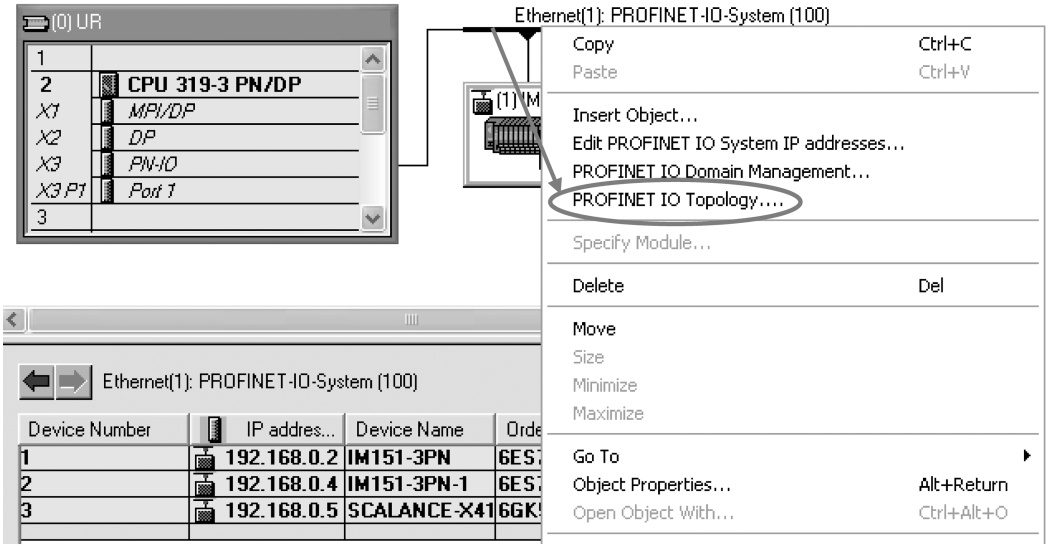



图 10-39　选择 PROFINET IO 拓扑

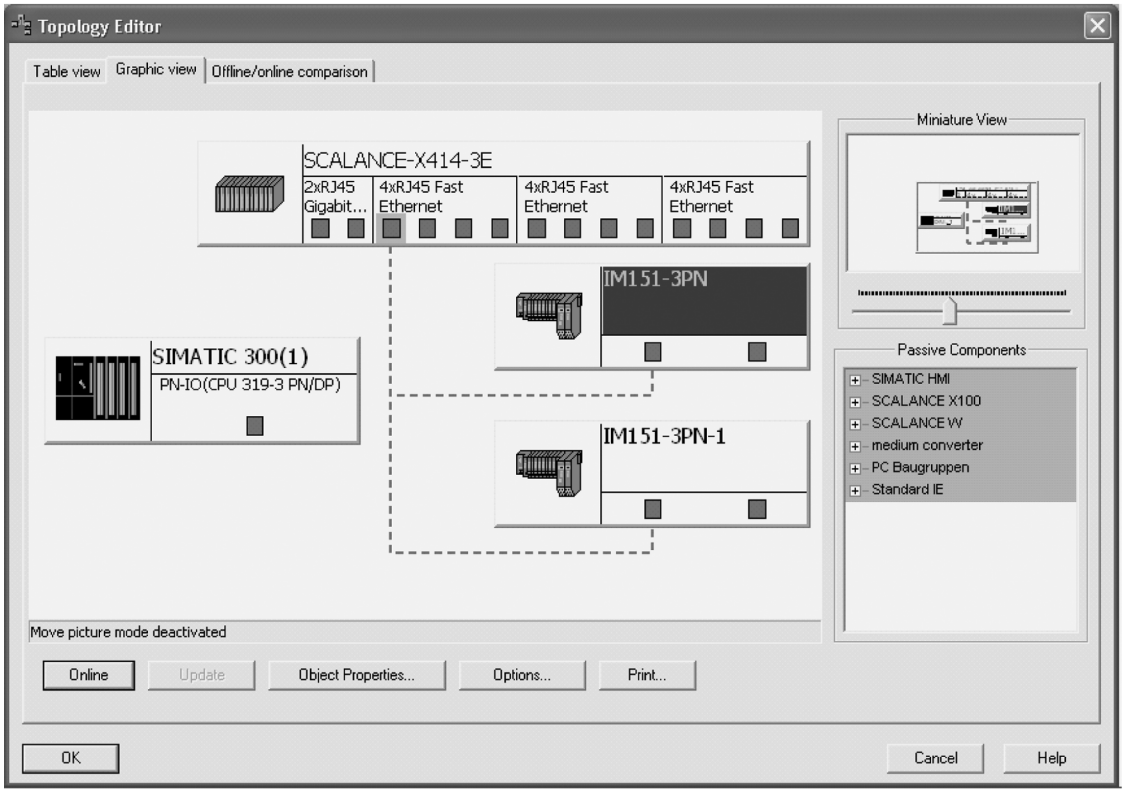



图 10-40　拓扑设备的拓扑连接关系

```
//IP:192.168.0.2  DeviceNo:1

      CALL  "D_ACT_DP"
       REQ     :=M10.2
       MODE    :=MB0
       LADDR   :=W#16#FF7
       RET_VAL:=MW2
       BUSY    :=M10.0

//IP:192.168.0.4 DeviceNo:2

      CALL  "D_ACT_DP"
       REQ     :=M10.4
       MODE    :=MB1
       LADDR   :=W#16#FF6
       RET_VAL:=MW4
       BUSY    :=M10.1
```

| Device... | IP address ... | Device Name | Order number | Fir... | Diagnostic address | initial state |
|---|---|---|---|---|---|---|
| 1 | 192.168.0.2 | IM151-3PN | 6ES7 151-3BA23-0AB0 | V6.0 | 4087* | Deactivated |
| 2 | 192.168.0.4 | IM151-3PN-1 | 6ES7 151-3BA23-0AB0 | V6.0 | 4086* | Deactivated |
| 3 | 192.168.0.5 | SCALANCE-X41 | 6GK5 414-3FC00-2AA2 | V2.1 | 4066* | activated |

图 10-41　编写 SFC12

更换无需存储介质/PG”。

在下列任务中，拓扑编辑器可为其提供支持：

- 获得关于项目中 PROFINET 设备所有的拓扑信息；
- 采用简单的拖拽操作将接口和端口进行互连，然后在 PROFINET 上组态拓扑集，并设置属性。

拓扑编辑器具有一系列功能，可用于设置、互连和诊断包括端口在内的所有 PROFINET 设备的属性。可提供下列功能和信息：

- 项目中所有 PROFINET 设备及其端口的显示；
- 使用计算的信号运行时间，为每个端口组态电缆长度和组态电缆类型；
- 通过本地识别各个 PROFINET 设备来互连数据；
- 每个单独端口的 PROFINET 设备的诊断信息；
- 通过在线/离线比较节点数据，简化默认检测；
- 从图形视图调用诊断（模块信息）；
- 导入网络拓扑。

在 PROFINET IO 系统中，CPU319-3PN/DP 作为 IO 控制器，IO 设备包含 SCALANCE X414-3E、SCALANCE X201-3P IRT、ET200S IM151-3PN、ET200S IM151-3PN FO，IO Supervisor（PG）也连接在网络中。PN IO 系统详细的网络拓扑如图 10-42 所示。

根据图 10-42 网络设备，在 STEP7 中进行硬件组态，如图 10-43 所示。

分配 Device name，然后下载硬件组态到 PLC 中，建立 PN IO 通信。具体方法请参考 10.5.1 节。使用鼠标右键点击“PROFINET 总线 Ethernet（1）：PROFINET-IO-System（100）”，选择“PROFINET IO Topology…”，如图 10-44 所示。

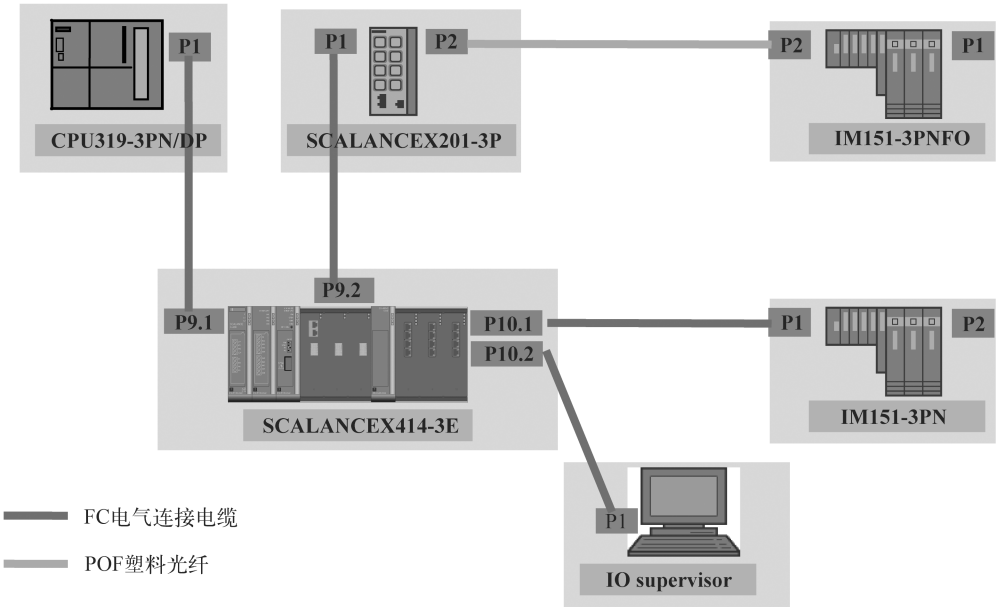



图 10-42　PN IO 系统详细的网络拓扑

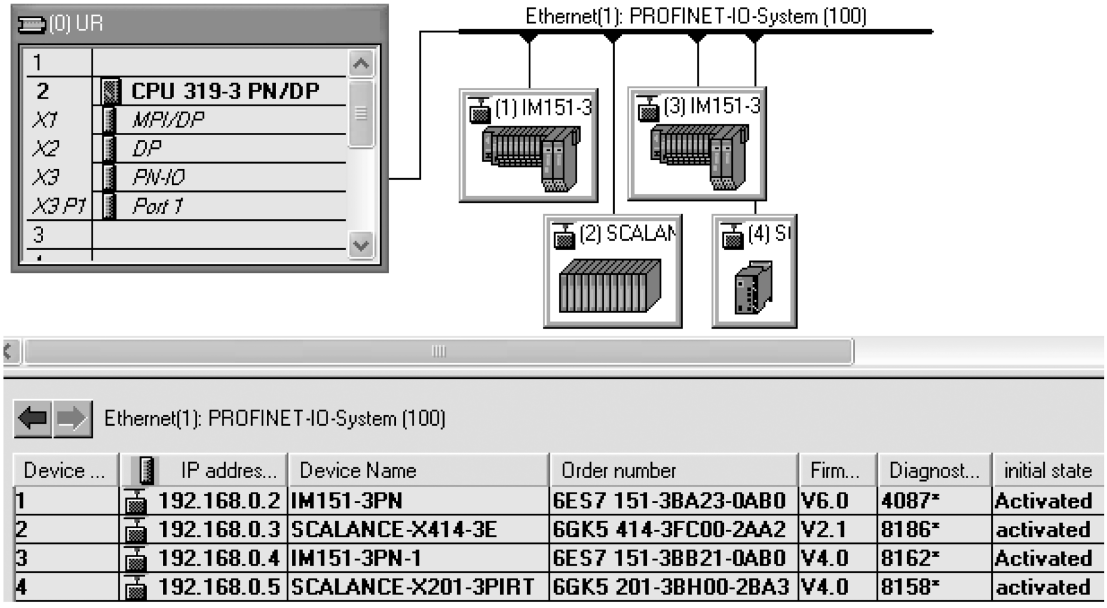



图 10-43　硬件组态

弹出拓扑编辑器对话框，选择“Graphic view”可以看见在 PROFINET IO 系统中的 IO 设备的端口连接关系。拓扑编辑器的设备如图 10-45 所示。目前拓扑中仅存在上述设备，还没有建立它们之间的连接关系。可以按照图 10-42 所示的 PN IO 系统详细的网络拓扑手动进行端口一对一的连接。

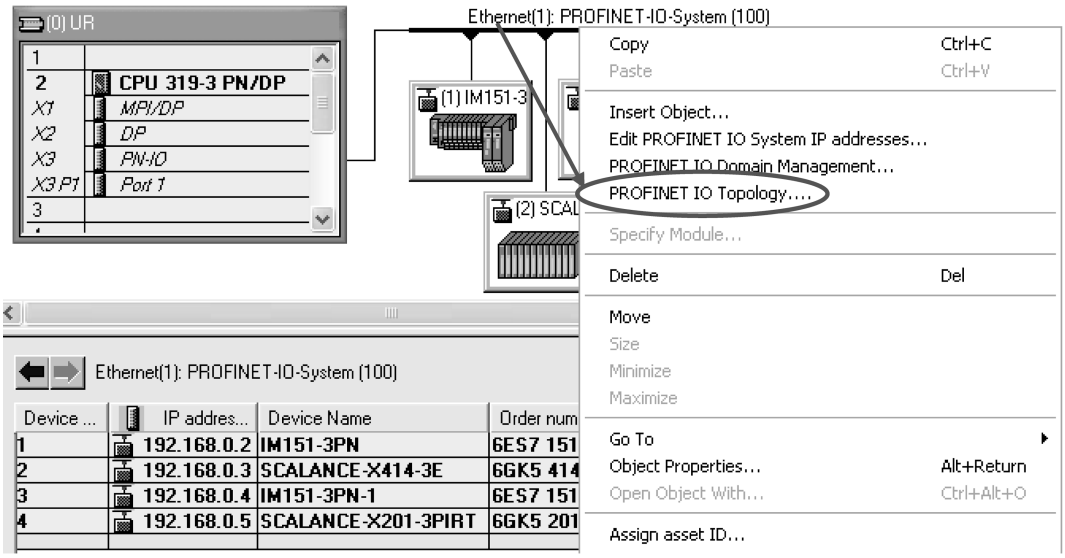



图 10-44　选择 PROFINET IO 拓扑

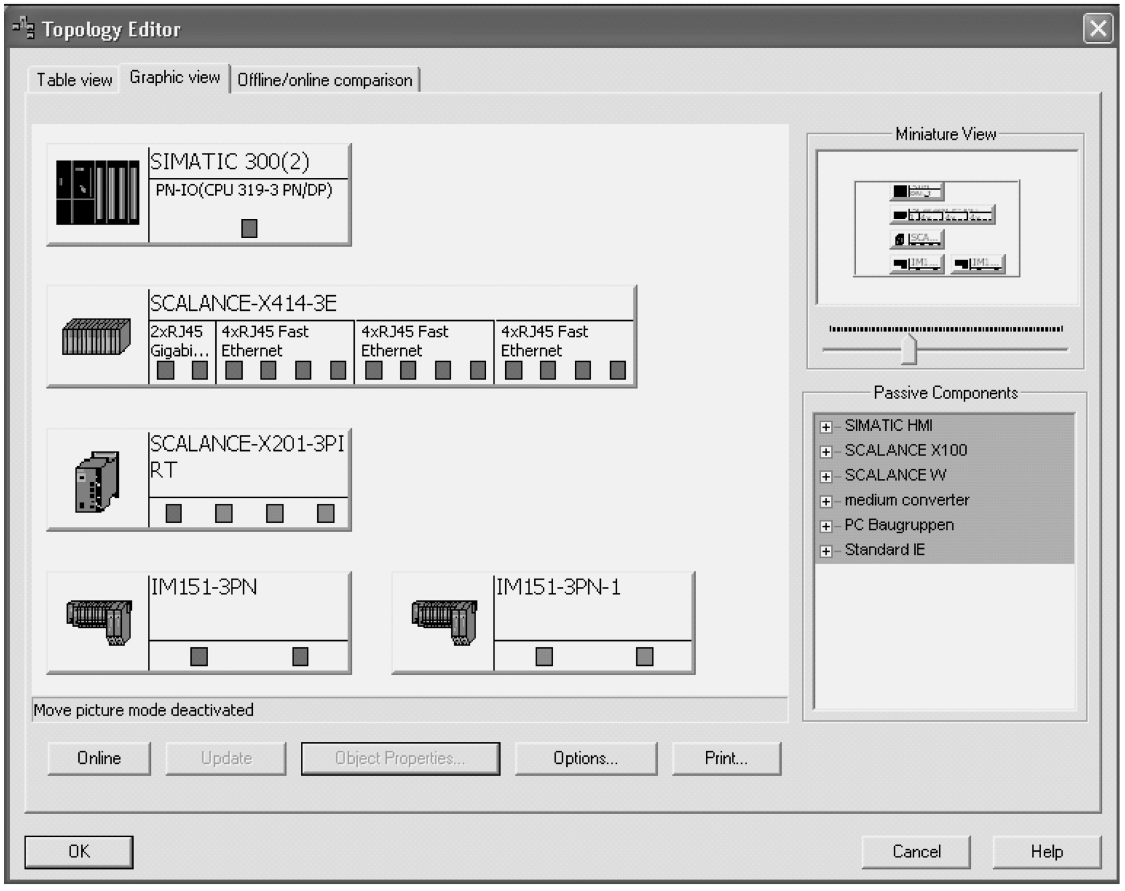



图 10-45　拓扑编辑器的设备

也可以在拓扑编辑器对话框中，选择“Offline/online comparison”，点击右侧的“Start”按钮，检测 PN IO 系统在线的拓扑连接信息。左侧为离线状态，右侧是在线状态，黄色背景表示离线和在线的设备匹配，但端口连接信息不匹配。橘黄色表示不与任何的组态设备相匹配。对于其他颜色的意义，请点击“Help”按钮查看帮助。离线与在线的比较如图 10-46 所示。

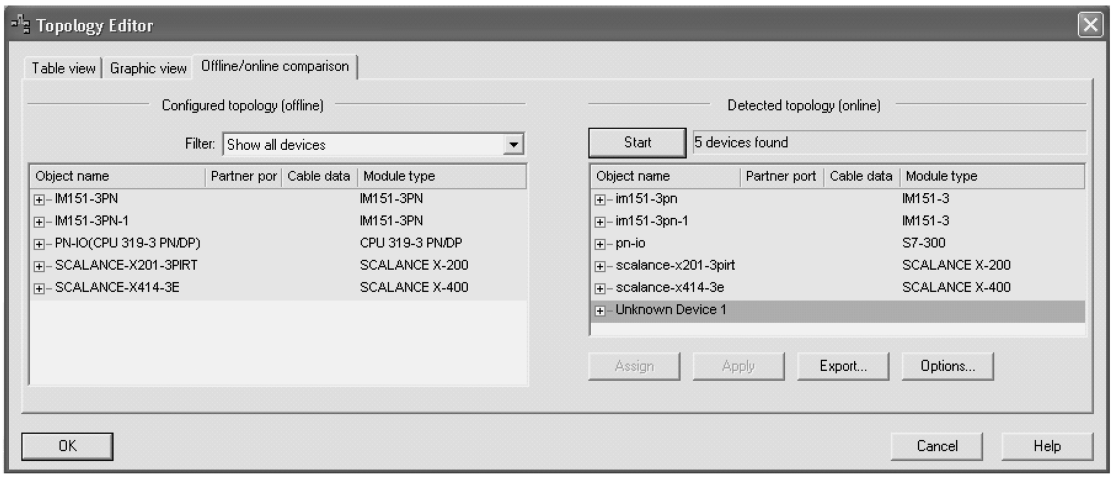



图 10-46　离线与在线的比较

在右侧在线检测的设备栏内，点击“+”号进行展开，鼠标左键点击在“Partner port”含有连接信息的 Port 端口。这时“Apply”按钮，显示为高亮状态，点击该按钮后，在离线侧出现了对应端口连接的信息，如图 10-47 所示。这时，对象背景颜色从黄色变为绿色，表示离线与在线设备全部匹配，包括端口连接信息。

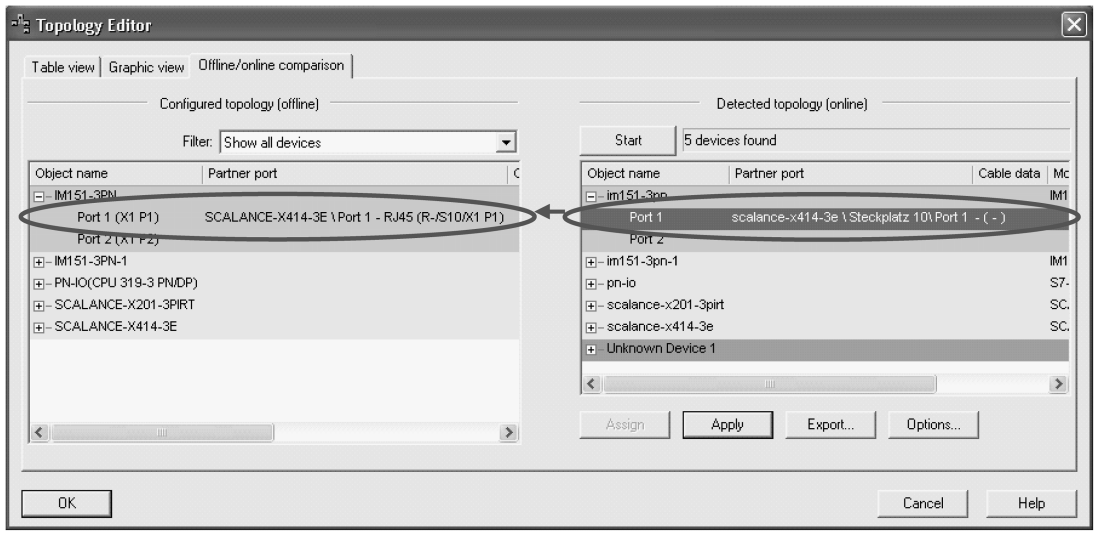



图 10-47　应用连接信息

按照上述同样的方法，使用“Apply”按钮把在线的端口连接信息应用到相对应的离线端口连接信息上。然后回到“Graphic view”页，可看到检测到的各个 IO 设备的端口连接信息。设备的拓扑关系如图 10-48 所示。

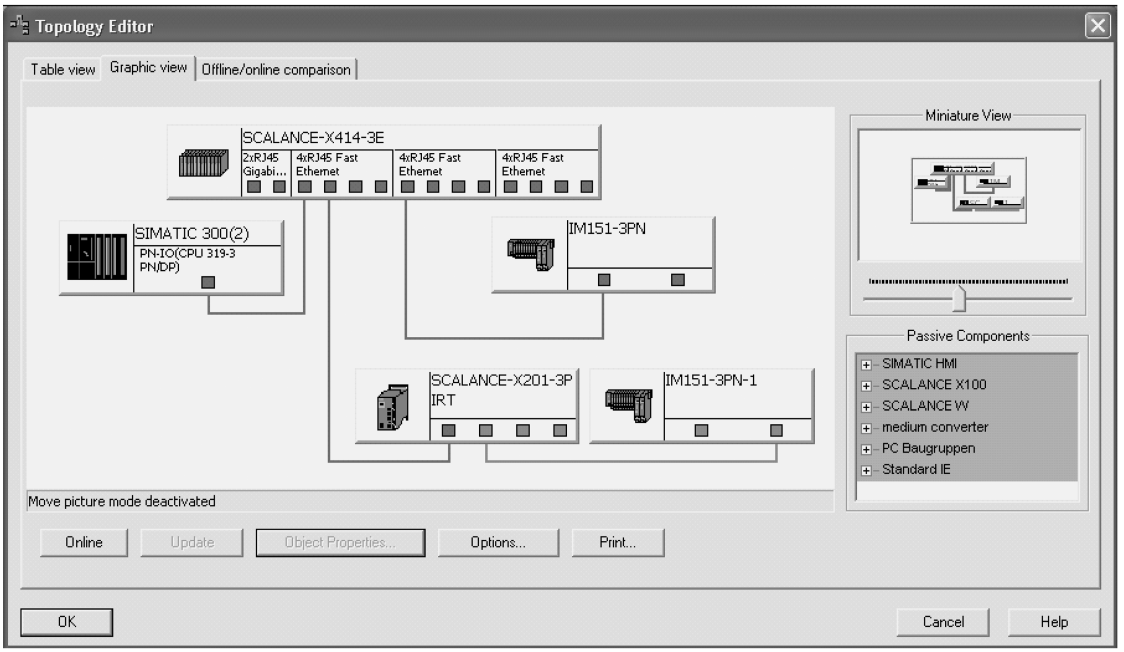



图 10-48　设备的拓扑关系

对于 IO supervisor，由于使用本地以太网网卡，并不支持 PDEV，所以需要手动添加到 PN IO 拓扑网络中。在右侧的 PC Baugruppen 中，使用鼠标把 Desktop PC 拖拽到拓扑中，并连接端口到 SCALANCE X414-3E 的 10.2 端口，如图 10-49 所示。

双击“Desktop PC”，弹出“Passive Components”对话框，输入 IO supervisor 的 IP 地址，这里是 192.168.0.254。设置 IO supervisor 的 IP 地址如图 10-50 所示。

再次切换到“Offline/online comparison”页，点击右侧的“Options”按钮，弹出“Options”对话框的“Search method”页，使能“Use ping（ICMP）”和“Use extended switch information（bridge MIB）”两个选项。再在“Scan range”页，设置搜索范围从 192.168.0.250～192.168.0.255，搜索范围越小，搜索的速度越快。点击“OK”结束。设置“Option”选项如图 10-51 所示。

在“Graphic view”页点击“online”按钮可查看在线的状态。点击“Help”按钮可查看帮助。IO supervisor 的暗绿色连线表示不能诊断。在线浏览拓扑状态如图 10-52 所示。

切换到拓扑编辑器的“Table view”页，点击在线“online”按钮，可以查看设备状态和端口的连接信息，如图 10-53 所示。

关闭拓扑编辑器，再次下载硬件组态，这样网络拓扑信息就下载到 PLC 中，当拓扑发生变化时，模块的 SF 灯会亮，提示拓扑错误。

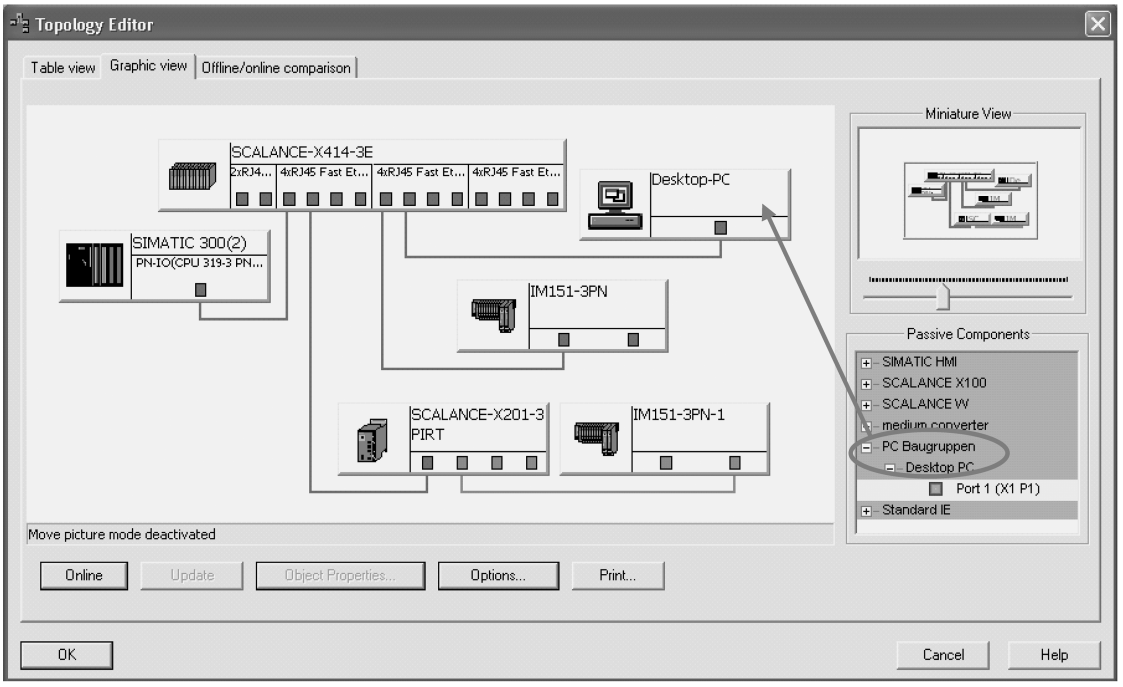



图 10-49　插入 Desktop PC

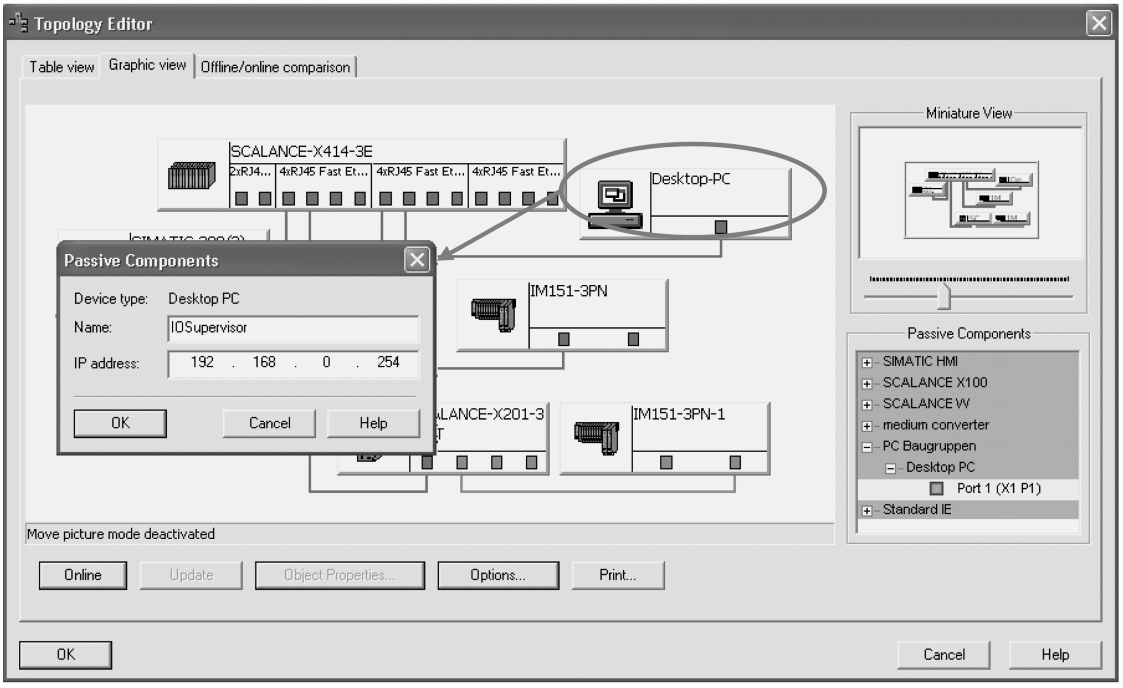



图 10-50　设置 IO supervisor 的 IP 地址

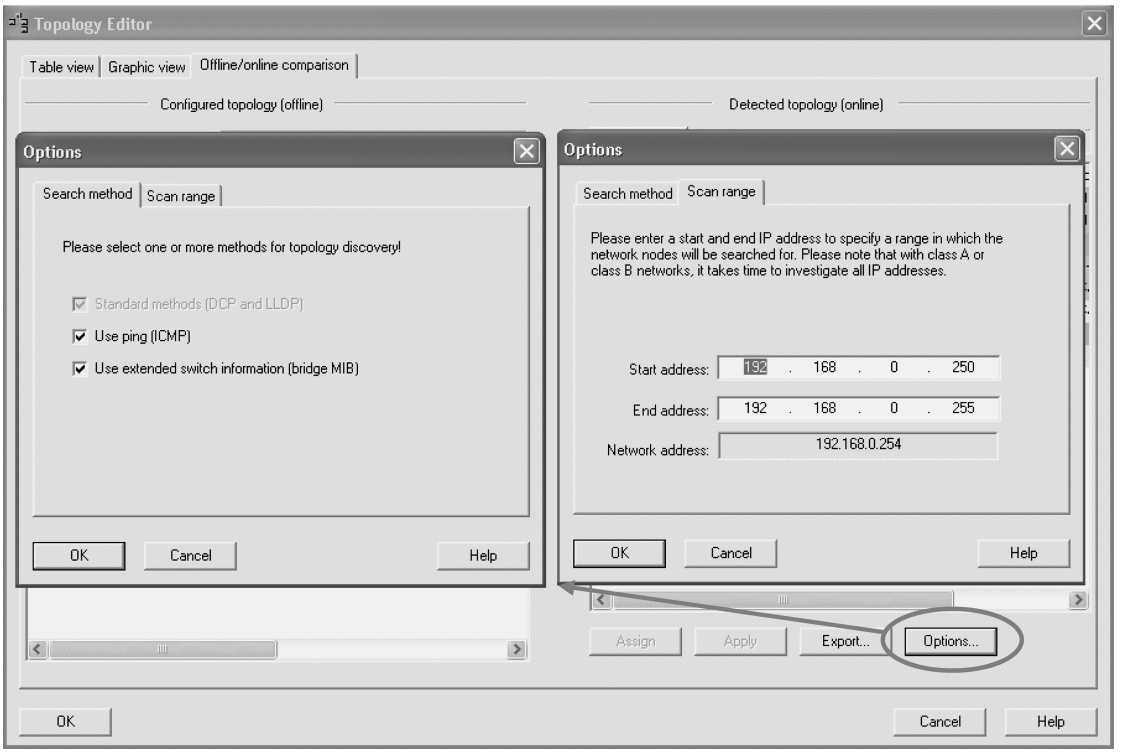



图 10-51　设置“Option”选项

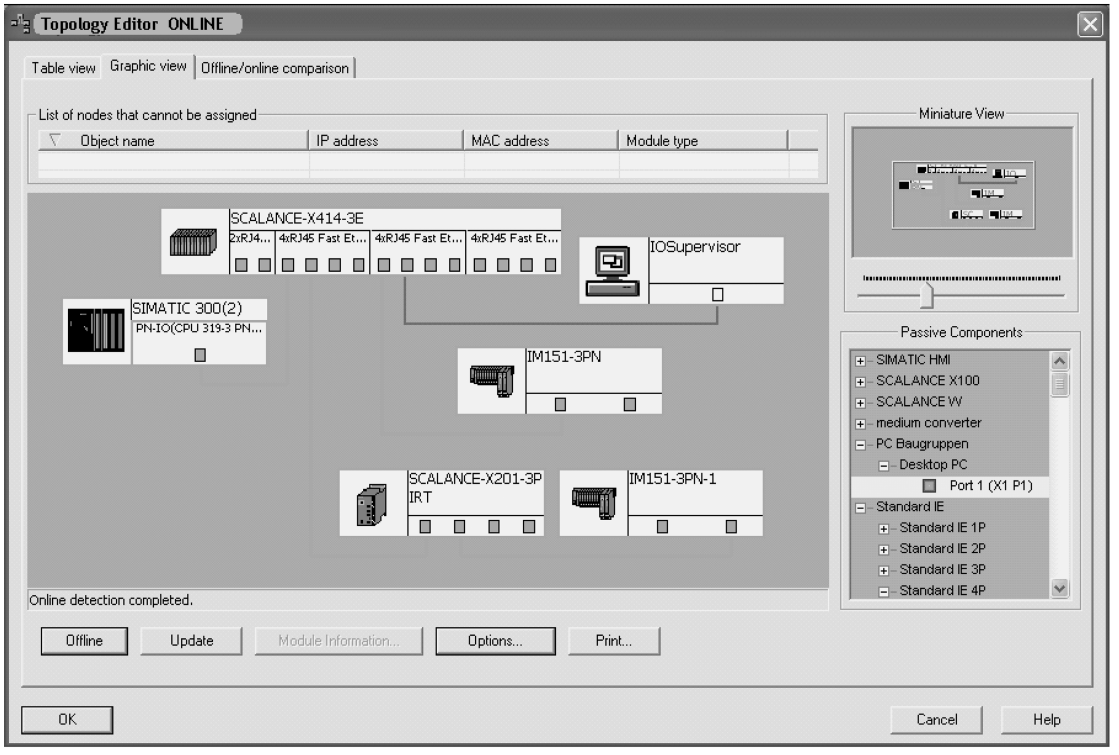



图 10-52　在线浏览拓扑状态

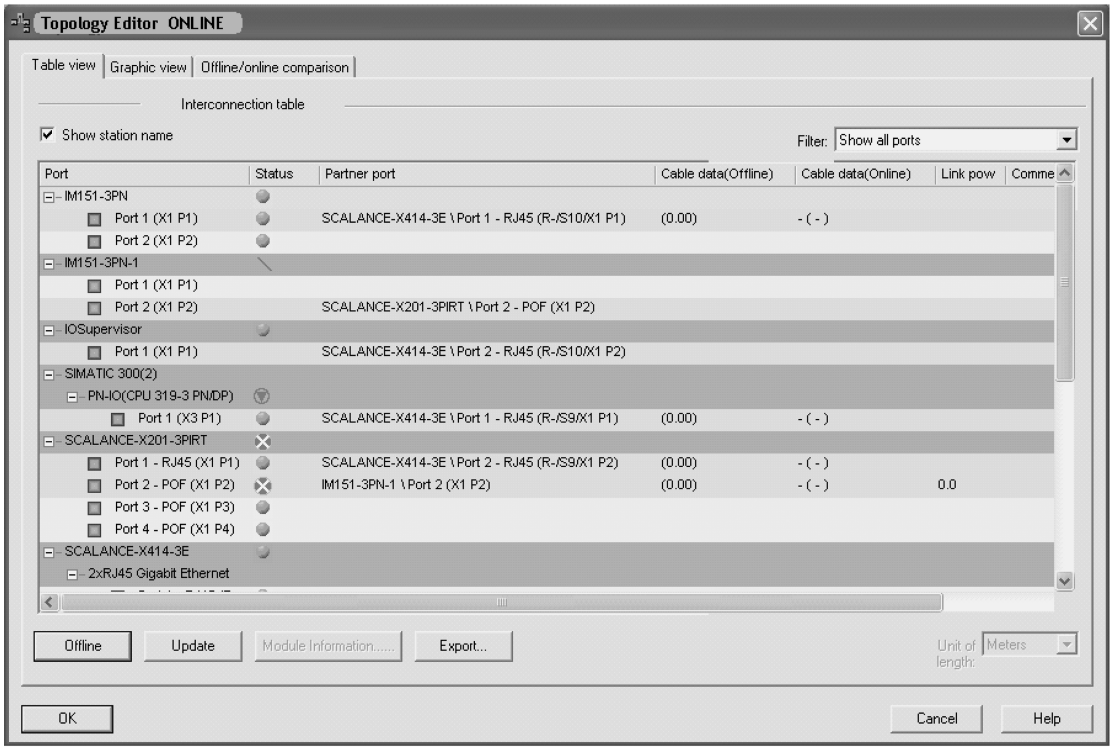



图 10-53 设备和端口的状态信息

## 10.5.4 PN IO 设备替换无需存储介质或 PG

支持 PROFINET 的“设备更换无需可移动介质/PG”功能的 IO 设备在更换过程中无需插入可移动介质（例如 MMC）或无需 PG 为其分配设备名。替换的 IO 设备的设备名由 IO 控制器分配而不是由可移动介质或 PG 分配。为此，更换的 IO 设备的 IO 控制器和邻近的 PROFINET 设备必须支持 PROFINET 的“设备更换无需可移动介质/PG”功能。为分配设备名称，IO 控制器使用组态的拓扑和从 IO 设备建立邻居关系。

如果希望使用 PROFINET 的“设备更换无需可移动介质/PG”功能，必须满足下列条件：

- 必须组态带有相应 IO 设备的 PROFINET IO 系统的拓扑。通过组态拓扑，PROFINET IO 系统或 IO 控制器将会清楚 PROFINET IO 系统中所有 PROFINET 设备的相邻关系。通过比较设定拓扑所规定的相邻关系和实际的 PROFINET 设备所确立的真实相邻关系，IO 控制器可识别没有名称的更换的 IO 设备，并将组态的名称和 IP 地址分配给更换的 IO 设备，然后再与其进行用户数据通信。
- 必须在 STEP7 中组态 PROFINET 的“设备更换无需可移动介质/PG”功能。
- IO 控制器和 IO 设备必须支持 PROFINET 的“设备更换无需可移动介质/PG”功能。
- 连接到相应 IO 设备的 PROFINET 设备必须支持 PROFINET 的“设备更换无需可移动介质/PG”功能。
- 在更换之前，要更换的设备必须能够复位为其出厂设置。

使用 PROFINET 的“设备更换无需可移动介质/PG”功能可享有下列优势：

- 在更换 IO 设备之后，它自动地从 IO 控制器获取其设备名称。
- 对于更换的 IO 设备，可以将其名称保存在本地的存储介质上。
- 可以节约已加载存储卡和 IO 设备中的设备数据的时间。
- 简化具有相同组态和设定拓扑的机械设备的设备名称的分配。

用一个例子来描述设备更换无需移动介质/PG 的原理，如图 10-54 所示。

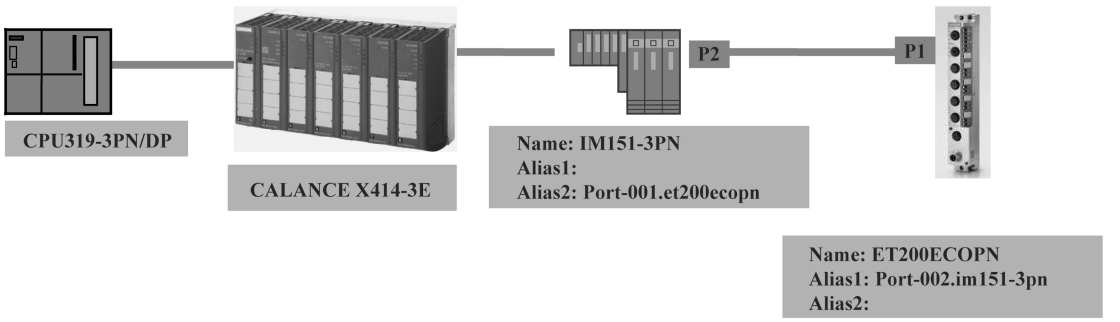



图 10-54　PROFINET IO 系统网络组态

IO 控制器使用 CPU319-3PN/DP 连接 SCALANCE X414-3E 交换机，该交换机再连接 IM151-3PN ET200S，通过该 ET200S 的端口 P2 连接到 ET200ECOPN 的端口 P1 上。这里假设替换设备为 ET200ECOPN，替换的 ET200ECOPN 需要复位到工厂默认值，即该 IO 设备没有设备名。当替换的设备接入到 PROFINET IO 网络中，CPU319 发送 DCP 识别（ET200ECOPN）该设备，由于该设备没有设备名，CPU319 不能收到 DCP 识别的响应。CPU319 然后会发送 DCP 识别别名（Port-002. IM151-3PN），ET200ECOPN 会响应 DCP 别名请求，CPU319 判断该替换设备拓扑连接信息正确，于是把设备名 ET200ECOPN 通过 DCP 设置分配给替换设备，启动过程继续直到通信完成。设备替换无需移动介质/PG 原理如图 10-55所示。

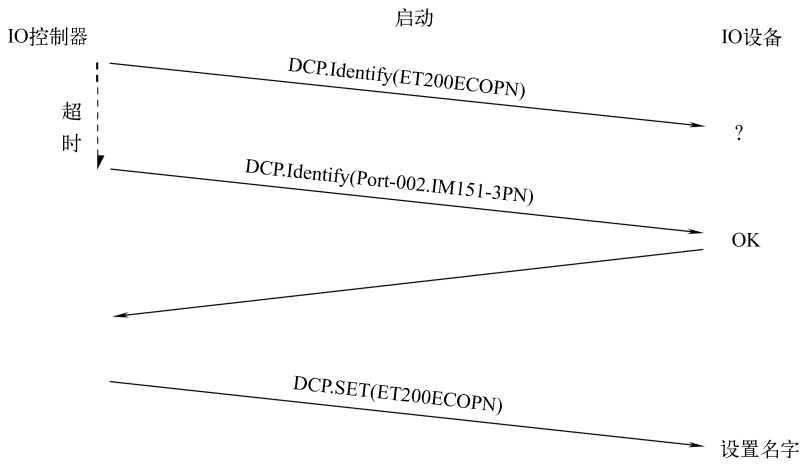



图 10-55　设备替换无需移动介质/PG 原理

新的设备都处于工厂默认状态，如果曾经使用过的设备用作替换设备时，首先需要复位到工厂默认状态。选择 SIMATIC Manager 的菜单“PLC→Edit Ethernet node…”，弹出编辑以太网节点对话框，点击“Browse…”按钮，浏览网络设备，选择要替换的 ET200ECOPN，点击“OK”结束。编辑以太网节点如图 10-56 所示。

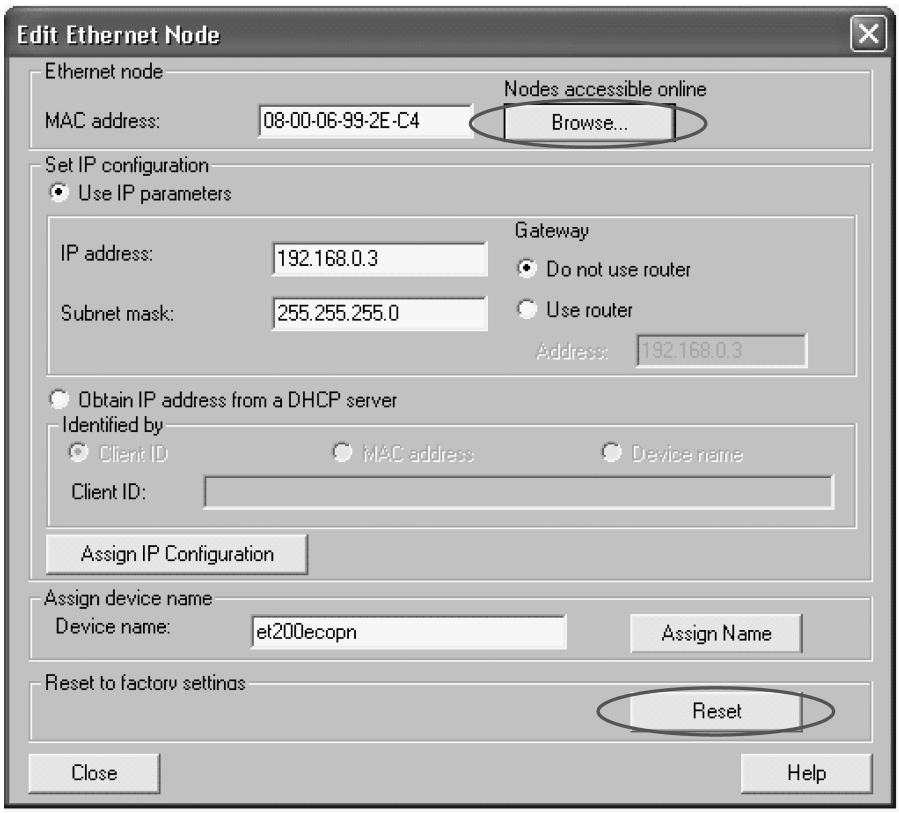



图 10-56 编辑以太网节点

在复位工厂默认值之前，需要断开 PROFINET IO 通信服务，然后点击“Reset”按钮，提示恢复工厂默认值成功，如图 10-57 所示。

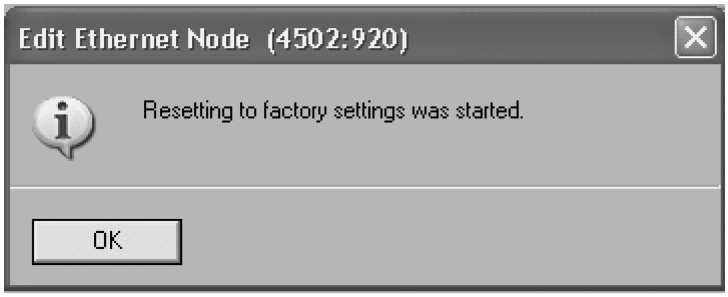



图 10-57 恢复工厂默认值

按照图 10-54 所示进行硬件组态，该系统进行 PROFINET IO 通信，具体组态 PROFINET IO 通信的详细信息请参考 10.5.1 节。硬件组态如图 10-58 所示。

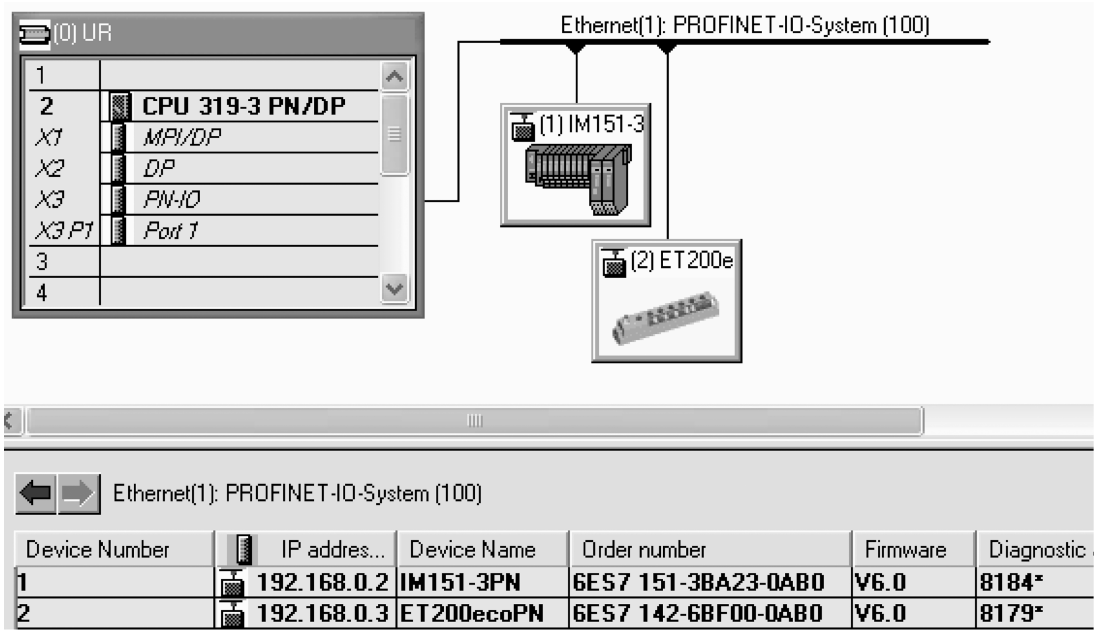



图 10-58　硬件组态

双击 CPU319-3PN/DP 的接口“PN-IO”，弹出接口属性对话框，保持“Support device replacement without exchangeable medium”为默认状态，即使能状态，如图 10-59 所示。

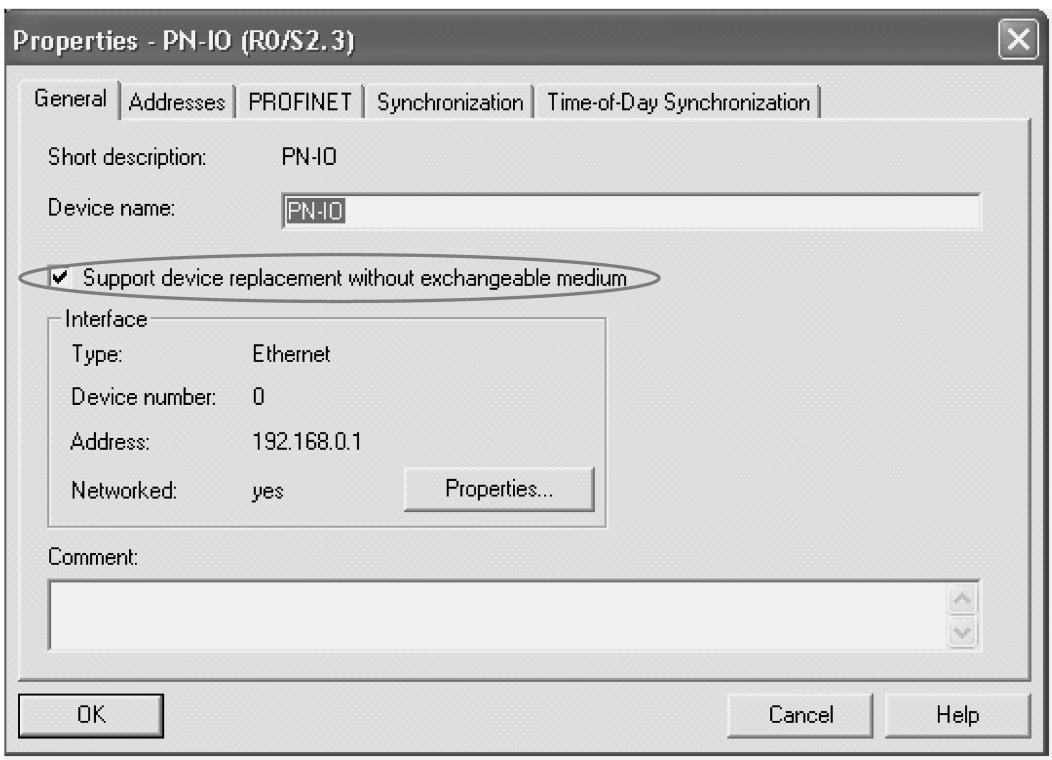



图 10-59　PN-IO 属性对话框

点击总线“Ethernet（1）：PROFINET-IO-System（100）”，右键点击弹出菜单选择“PROFINET IO Topology…”，进入拓扑编辑器中，在“Graphics view”页，使用鼠标连接 ET200S 的端口 P2 到 ET200ECOPN 的端口 P1。保存编译下载到 PLC 中。连接拓扑组态如图 10-60 所示。

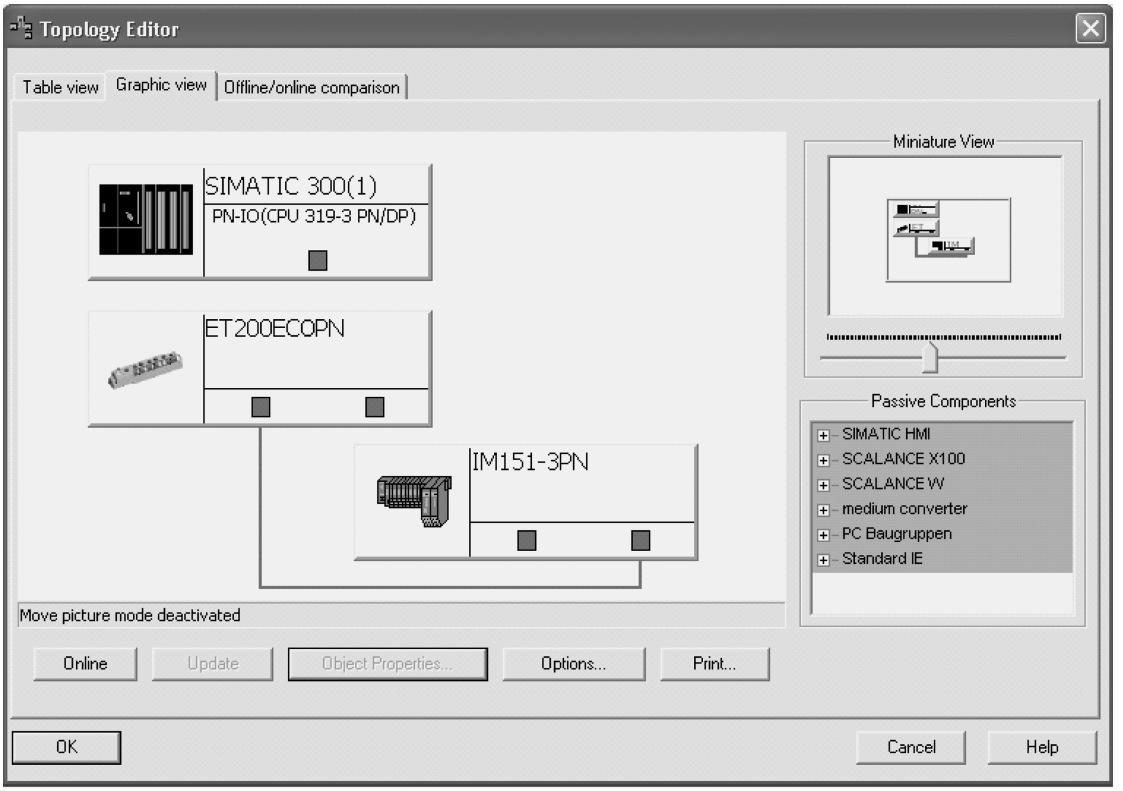



图 10-60　连接拓扑组态

当 ET200ECOPN 发生故障需要更换时，断开该设备的电源，然后把工厂复位后的 ET200ECOPN（或新的设备）的端口 1 连接到 ET200S 的端口 2 上，恢复供电即可实现无需移动介质和 PG 的功能，进而完成 PROFINET IO 数据交换。

### 10.5.5　PN IO 网络诊断-Web

连接 Web 服务器的客户端，例如 PG/PC 到 CPU 的 PROFINET 接口或者网络中的交换机端口上，打开 IE 浏览器输入该 CPU 的 IP 地址（例如：http://192.168.0.1）即可。

S7-400 系列 PN 的 CPU 从 Firmware v5.2 开始支持 Web 服务器，S7-300 系列 PN 的 CPU 从 Firmware v2.6 开始支持 Web 服务器。

Web 服务器可以从 CPU 中读到以下信息：

- 起始页 CPU 基本信息；
- 识别、订货号、版本等信息；
- CPU 的诊断缓冲区；

- 模块的信息；
- 消息（没有应答选项）；
- 关于通信的信息；
- 拓扑信息；
- 变量状态；
- 变量表。

组态上述 Web 服务器的数据存储在 CPU 的存储介质上，例如 MMC 卡。推荐使用的 MMC 卡的容量大于 512k。

另外，与 CP 卡不同，集成的 Web 服务器的 CPU 本身不提供任何的安全机制，那么如果需要防止非授权的访问，可以使用一个防火墙来保护信息安全。

PROFINET IO 系统的硬件组态，如图 10-61 所示。

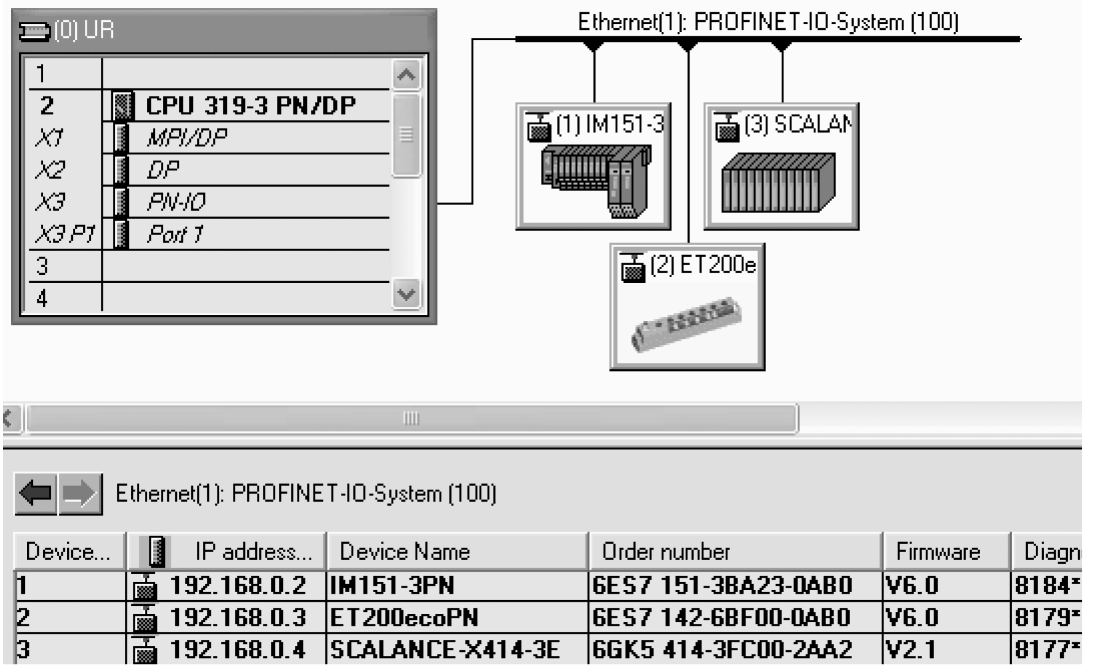



图 10-61　硬件组态

双击“CPU319-3PN/DP”，弹出“CPU”属性对话框，使能“Enable Web server on this module”，选择语言为“English”。这里最多可以选择两种语言，该语言用于显示消息和诊断信息的文本信息。自动刷新使能“Enable”，保持默认的刷新 Web 页面的时间间隔为 10s。这里所组态的刷新时间是最小的刷新时间，当数据量大或者具有多个客户端连接时，刷新时间将增大。保持显示分类为默认状态 0～16，其中的 0～16 勾选的越少，生成的 SDB 越小，从而占用存储介质的容量越小。这些设置显示分类的消息会显示在“Message”的 Web 页面上，没有选择消息分类的消息，一经产生将以十六进制代码显示而没有文本显示。CPU 属性如图 10-62 所示。

所选择的基于语言（English）的文本下载到 PLC 中，同时需要注意的是，这里所选择

的语言（English）需要被安装。可通过在 SIMATIC Manager 下的菜单命令“Options →Language for Display Devices”进行安装。添加语言如图 10-63 所示。

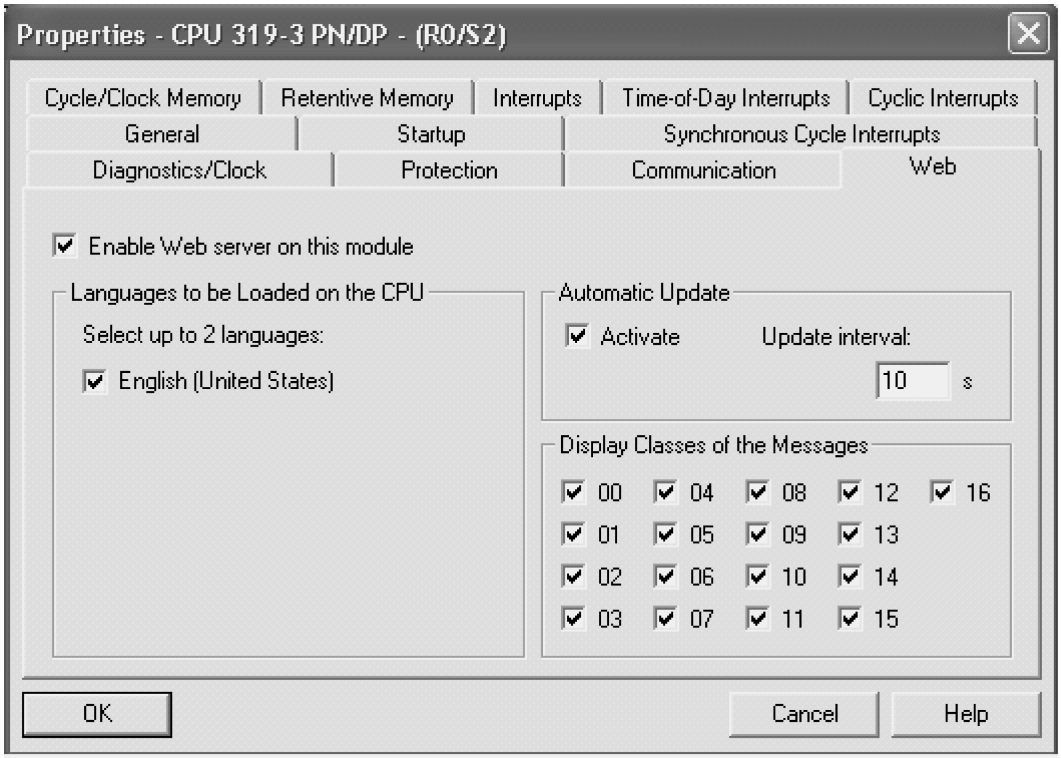



图 10-62　CPU 属性

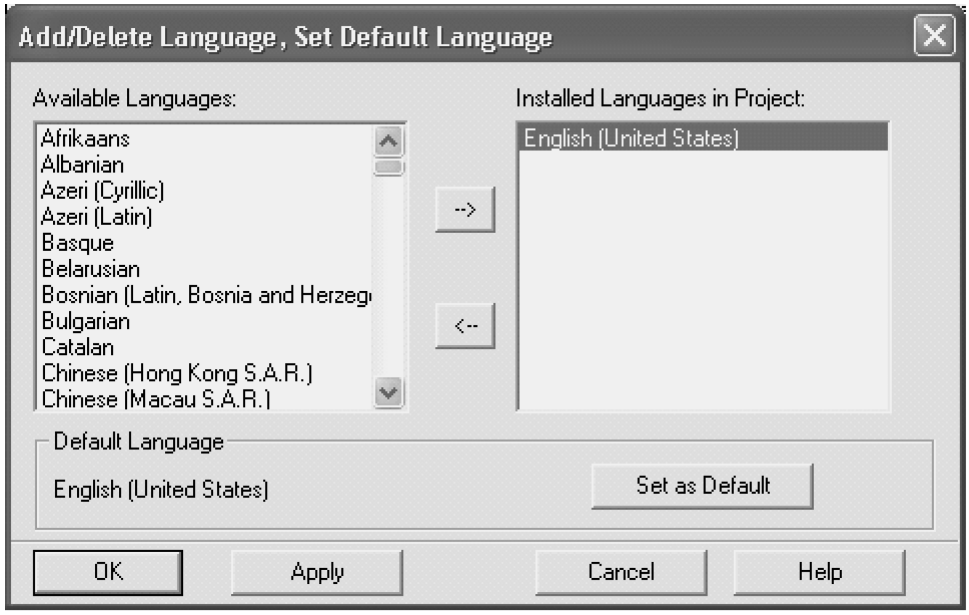



图 10-63　添加语言

组态消息有两种方式：一种是 STEP7 中功能块相关的消息，该种方式手动使用 Alarm 进行编程，手动编写消息文本；另外一种使用 STEP7 提供的“Report system error”功能，在硬件组态中，点选一下 CPU，然后选择菜单“Options→ Report system error”。在 Message 页面可以进行设置消息显示分类，其他设置为默认状态，然后点击“Generate”按钮，自动生成。RSE 组态如图 10-64 所示。

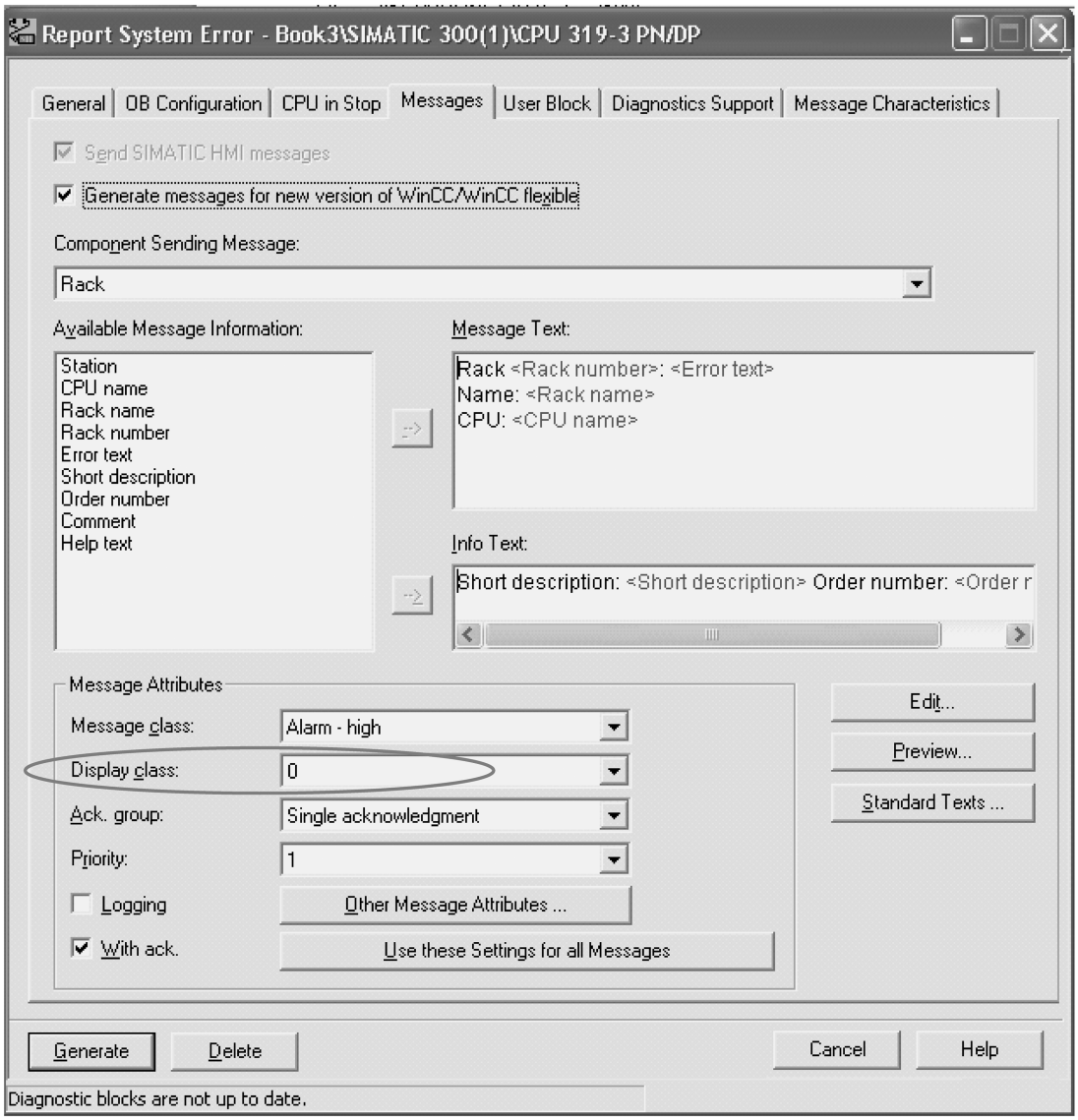



图 10-64　RSE 组态

添加一个变量表 VAT_1 到 Web 服务器中，通过该变量表的属性对话框使能“Web server”。变量表属性如图 10-65 所示。

最后把整个项目保存编译并全部下载到 CPU 中。打开 IE 浏览器，输入 CPU319-3PN/DP 的 IP 地址 192. 168. 0. 1，然后回车，Web 首页如图 10-66 所示，语言选择“English”。

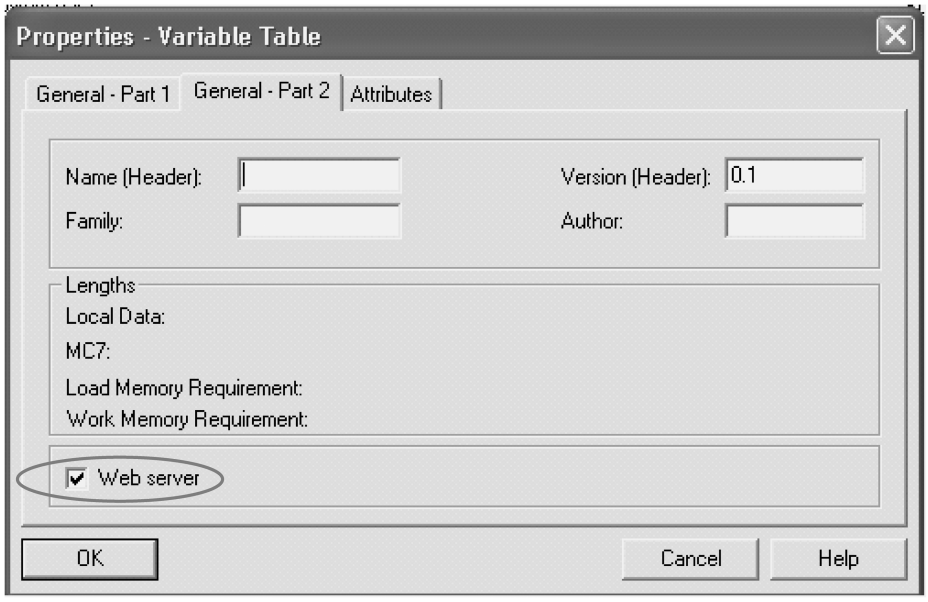



图 10-65　变量表属性

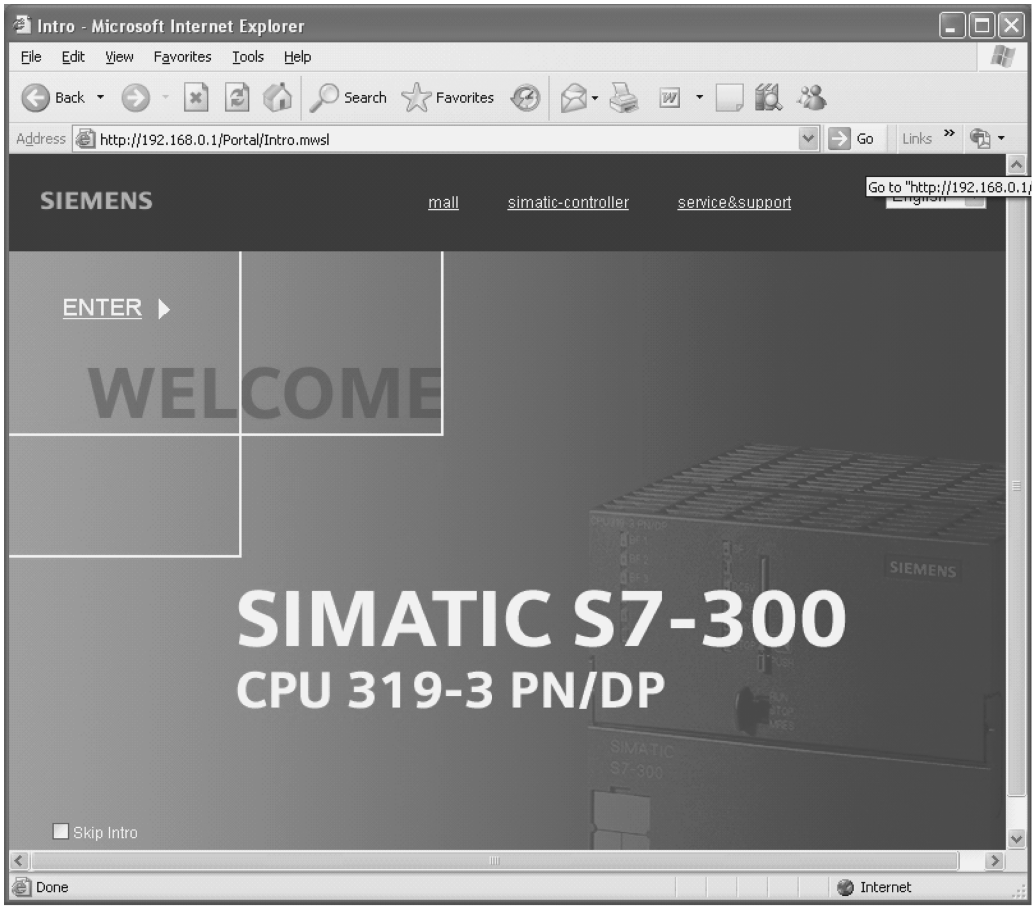



图 10-66　Web 首页

点击“Enter”进入 CPU 的起始画面，可看到 CPU319 的状态，如启动、停止、错误等一般信息。Web 起始页如图 10-67 所示。

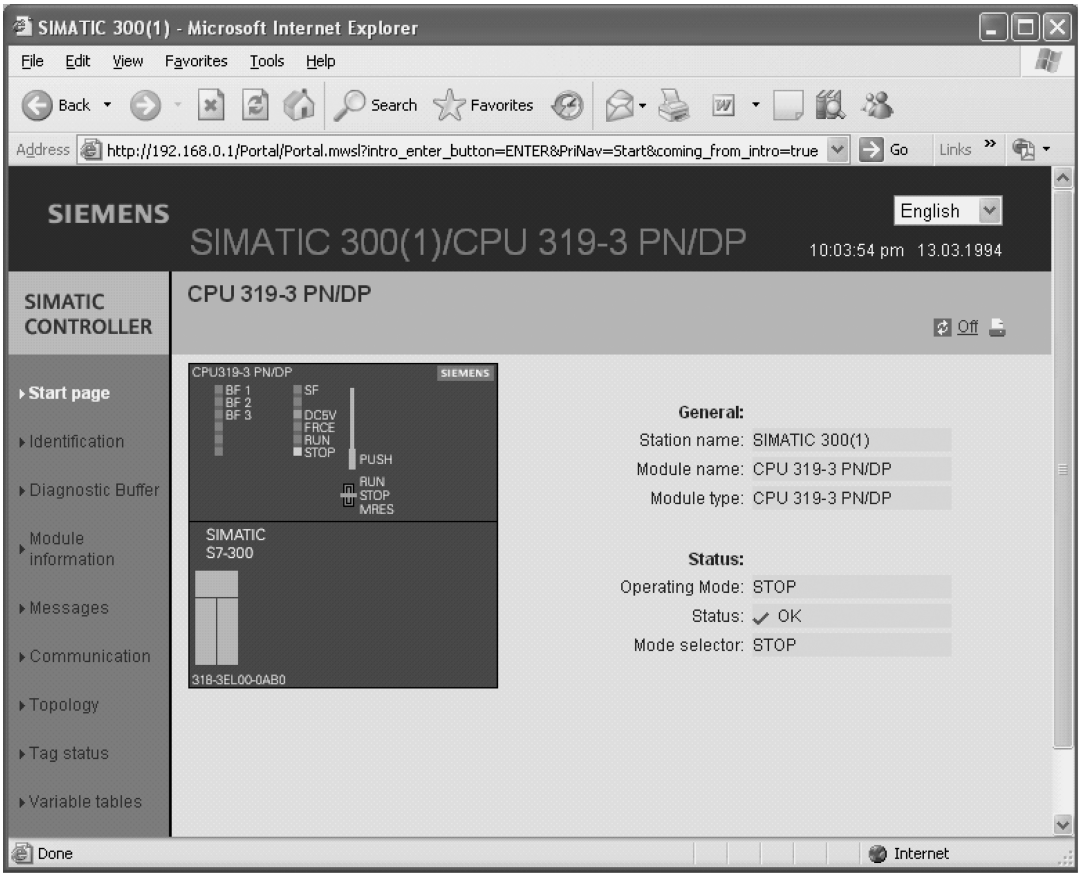



图 10-67　Web 起始页

点击左侧的“Identification”，可以看见 CPU 的订货号、序列号、Firmware 版本等信息。Web 识别画面如图 10-68 所示。

点击 Web 页面的“Diagnostic Buffer”，可以看见与 CPU 的 Buffer 一样的消息。CPU 的诊断缓冲区如图 10-69 所示。

点击 Web 页面的“Module information”，可以看见机架、网络、网络的设备状态，可以逐级点击查看更加详细的信息。模块信息如图 10-70 所示。

点击 Web 页面“Messages”，可以查看报警消息。报警消息如图 10-71 所示。

点击 Web 的“Communication”，可以看见相关通信的各种参数，包括网络连接、接口属性、IP 参数、统计信息等。通信参数如图 10-72 所示。

点击 Web 页面的“Topology”，可以看见整个 PROFINET IO 网络的拓扑信息、设备的状态等。拓扑信息如图 10-73 所示。

点击 Web 页面的“Tag status”，可以输入一些变量进行在线监视，例如输入 M0.0 的 Value 为 1。变量状态如图 10-74 所示。

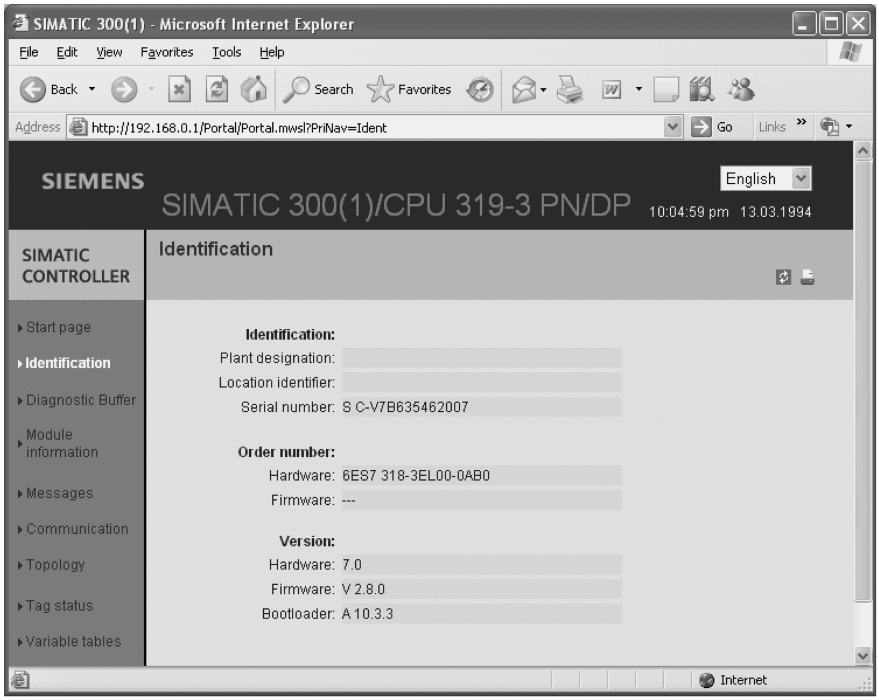



图 10-68　Web 识别画面

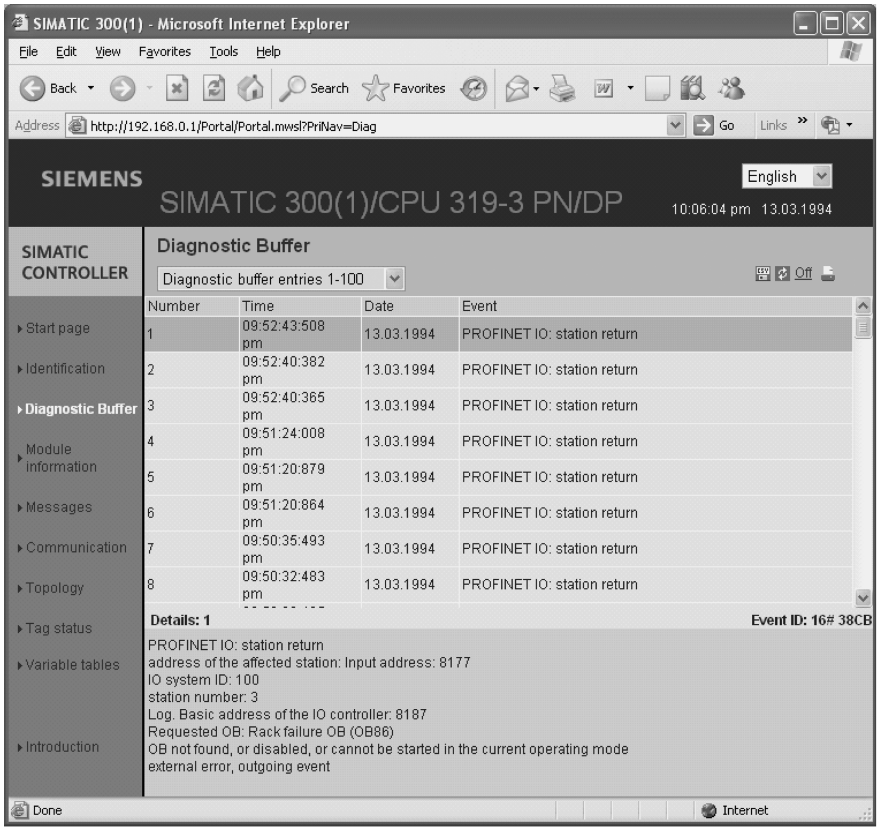



图 10-69　CPU 的诊断缓冲区

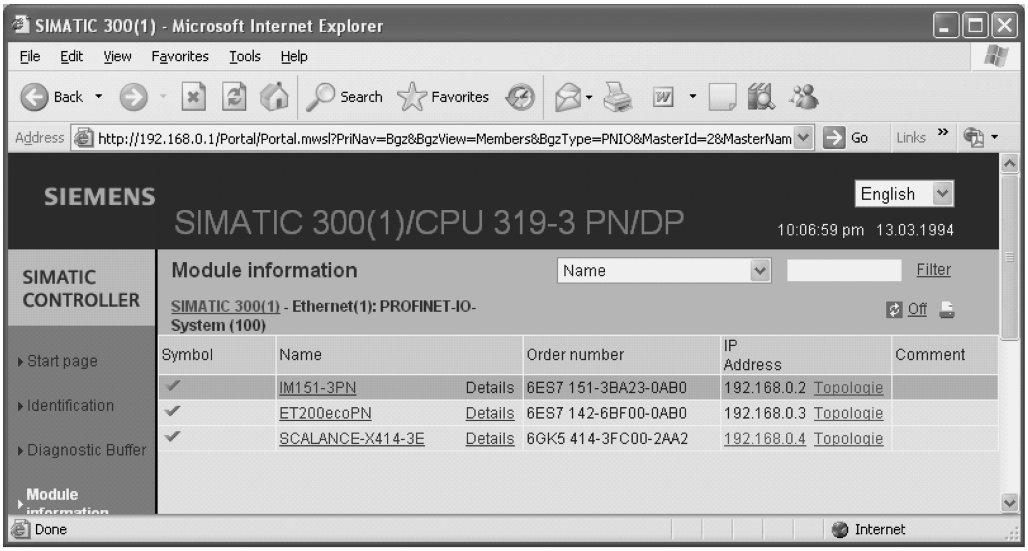



图 10-70　模块信息

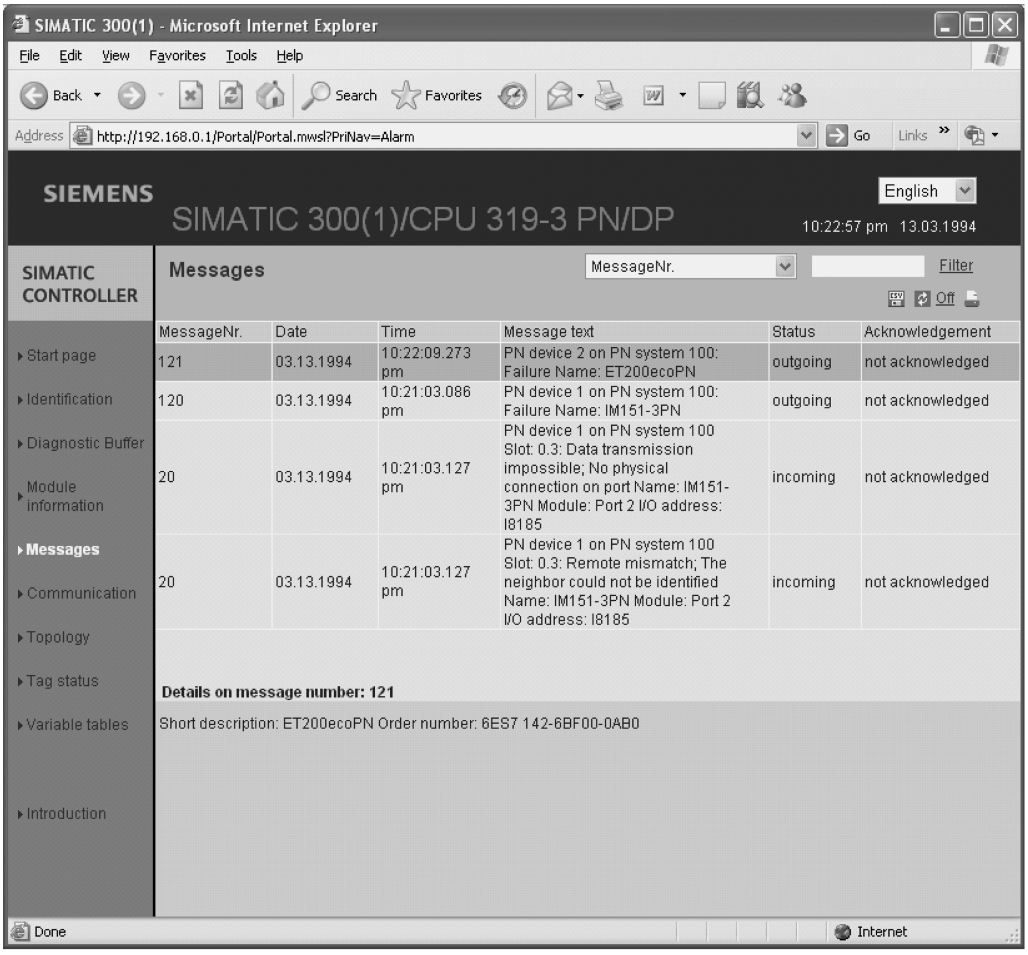



图 10-71　报警消息

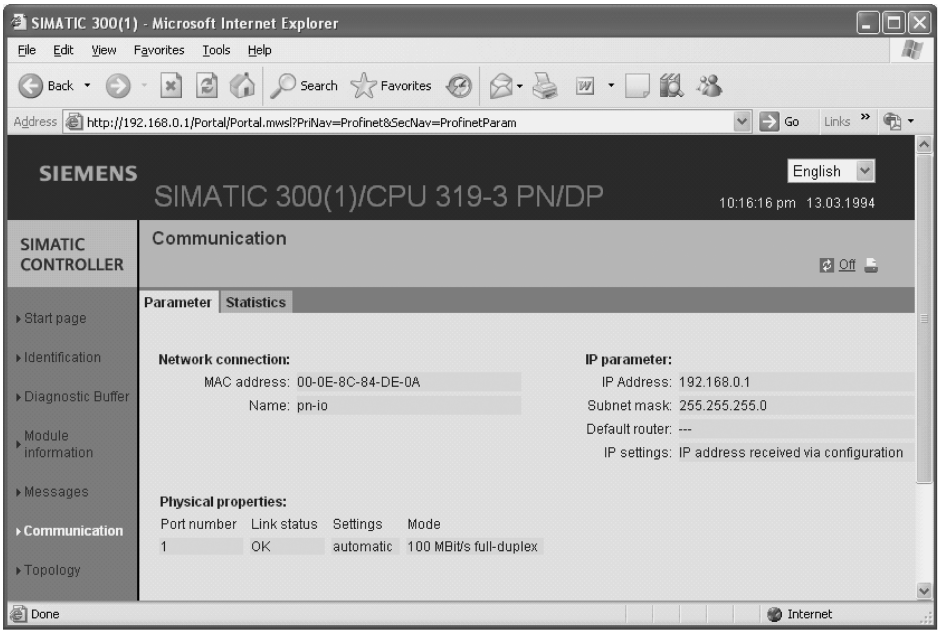



图 10-72　通信参数

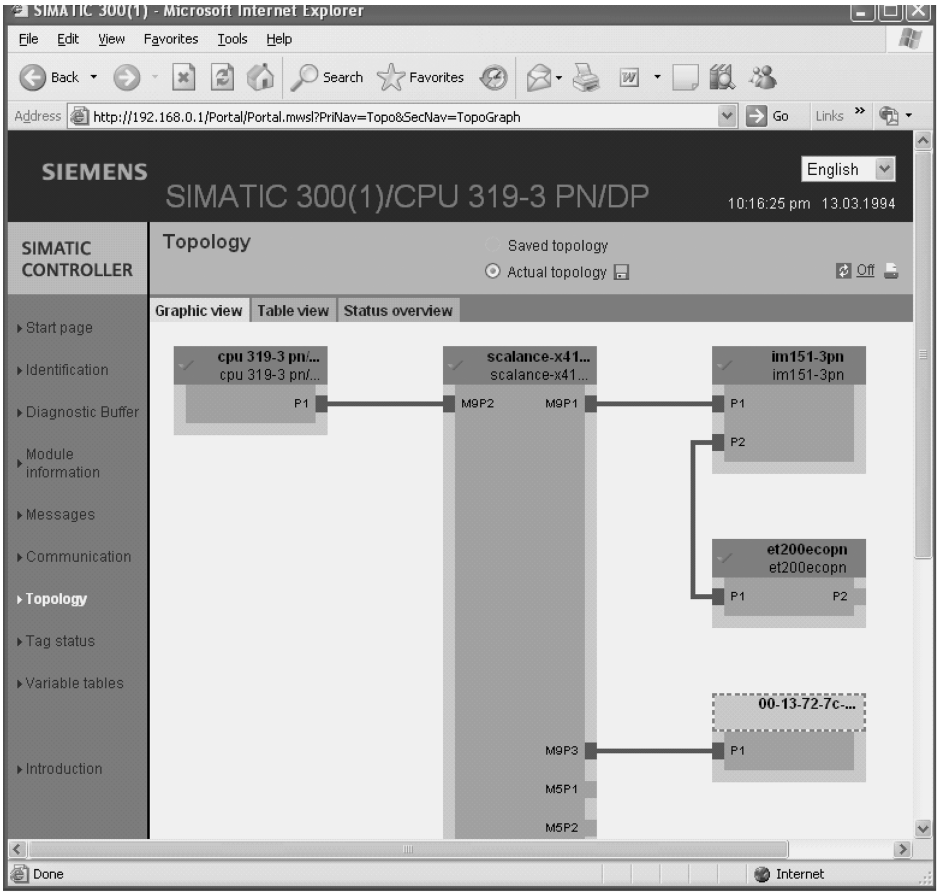



图 10-73　拓扑信息

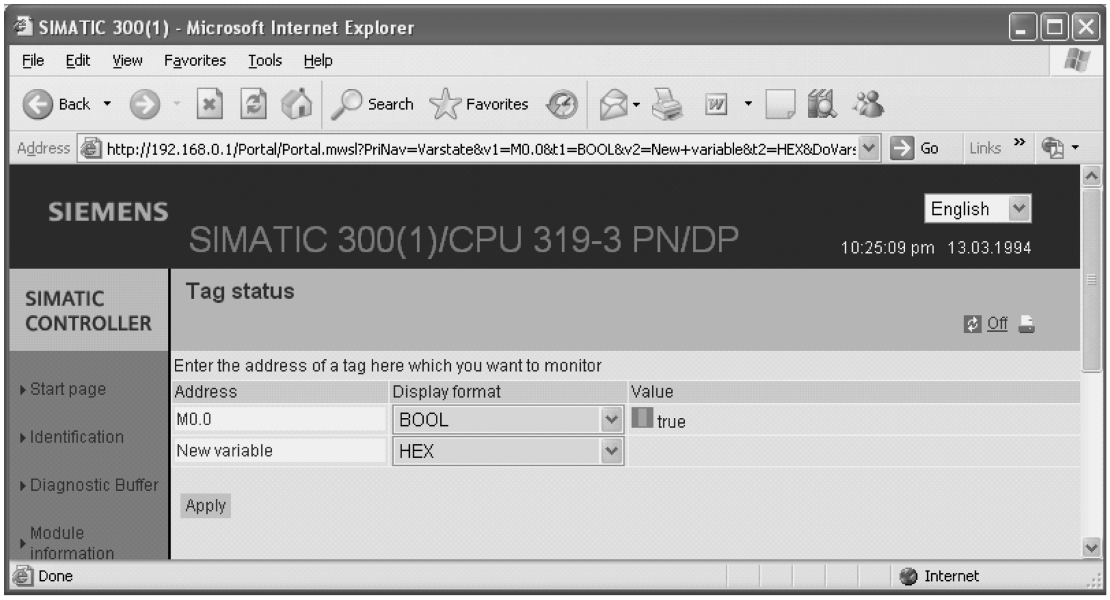



图 10-74　变量状态

点击 Web 页面的“Variable tables”，可以监控 VAT_1 变量表。变量表如图 10-75 所示。

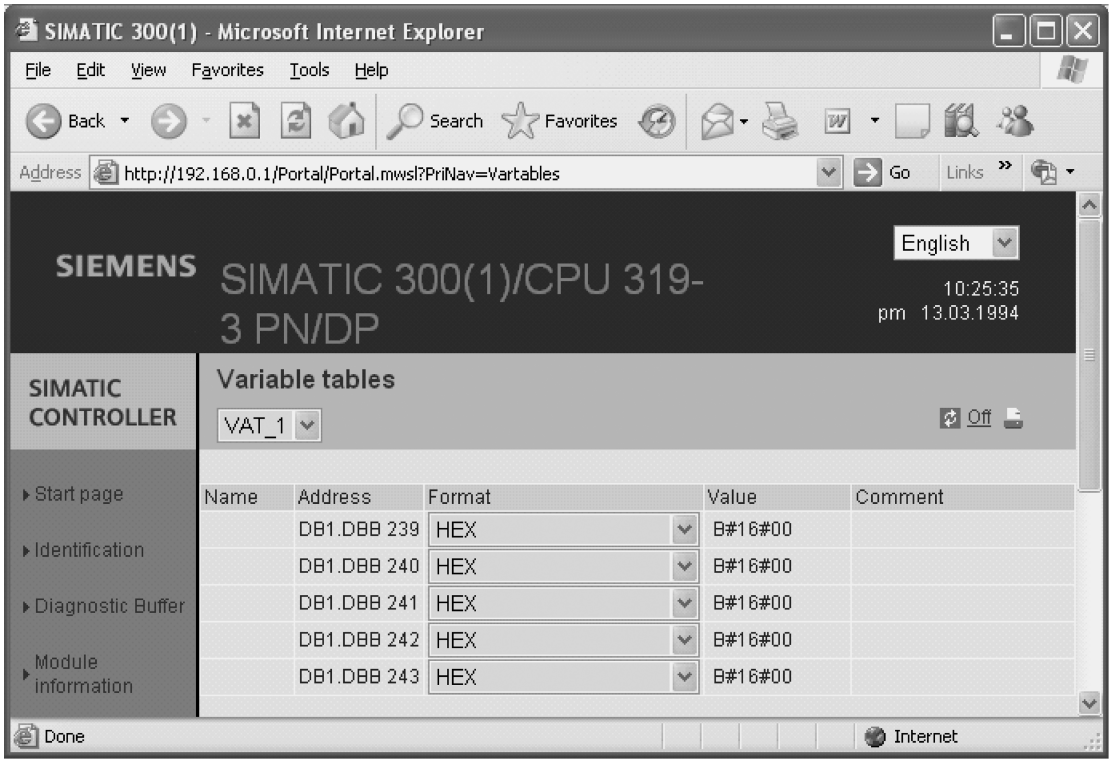



图 10-75　变量表

### 10.5.6 PN IRT-High flexibility

具备 IRT 功能的 PROFINET 是预留时间间隔内同步的通信。IRT 允许控制时间性很强的应用，例如通过 PROFINET 的动态运动控制。

即使标准通信中存在相当大的网络负载，仍具有高度精确的确定性。

通过 IRT 还可获得如下更多优势：

- 对于现有公司网络，可以进行实时应用，可实现简单而灵活的 PROFINET 设备集成；
- 预留传输带宽，可为数据传输到实时控制提供足够空闲的资源；
- 通过相同的传输介质可实现与实时通信并行的安全标准通信；
- 可以继续在同步域之外的 PROFINET IO 系统中使用标准部件；
- 线性总线拓扑中使用 IRT 通信的大量 PROFINET 设备，例如道路运输应用；
- 固定的预留带宽，不会由于延长线路问题而导致 STEP7 计算刷新时间的增加。

IRT-High flexibility 用于 PROFINET 设备之间 IRT 数据循环交换的同步传输过程。发送时钟内的预留带宽可供 IRT IO 数据使用。预留带宽可确保以预留的同步间隔传输 IRT 数据，同时还可保持不受其他更高网络负载（例如 TCP/IP 通信或附加的实时通信）的影响，当设备或网络故障同步功能无法完成时，IRT-High flexibility 的实时等级将降为 RT 通信。“高度灵活性”可实现系统的简单规划和扩展而无需进行网络拓扑组态。

IRT 通信的同步域内要求以太网控制器、交换机、分布式 IO 都需要 ERTEC ASIC（增强的实时以太网控制器）芯片支持，最小的更新时间为 250μs ，且发送时钟的抖动精度小于 1μs。借助特殊的以太网控制器（例如 ERTEC），可在硬件技术中实现预留传输带宽，在这种情况下，对于每个发送时钟，为 IRT 通信确立固定的时间段，剩余的时间段用于 RT 和 TCP/IP 通信。

用一个例子来描述 PROFINET IRT-High flexibility，如图 10-76 所示。

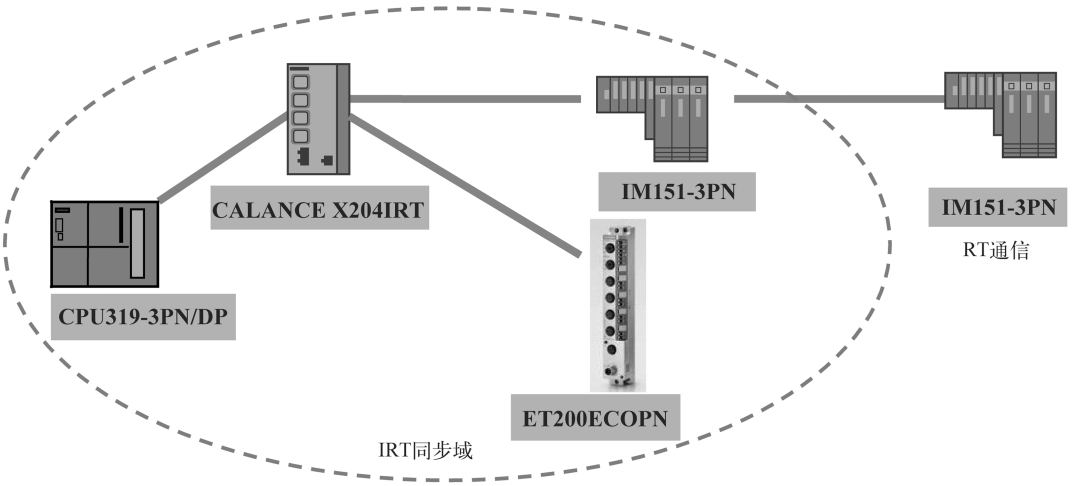



图 10-76 PROFINET IO 系统网络组态

IO 控制器使用 CPU319-3PN/DP v2.8 连接 SCALANCE X204IRT v4 交换机，交换机的两个端口分别连接 ET200S IM151-3PN（6ES7 151-3BA23-0AB0）和 ET200ECOPN（6ES7 142-

6BF00-0AB0），其中一台 ET200S 连接另外一台 ET200S。在同步域内，所有设备必须支持 IRT，即集成 ERTEC 芯片。同步域内的设备采用 IRT High flexibility 的通信方式，同步域外的设备采用 RT 通信方式。

在 STEP7 中对其进行硬件组态，如图 10-77 所示。

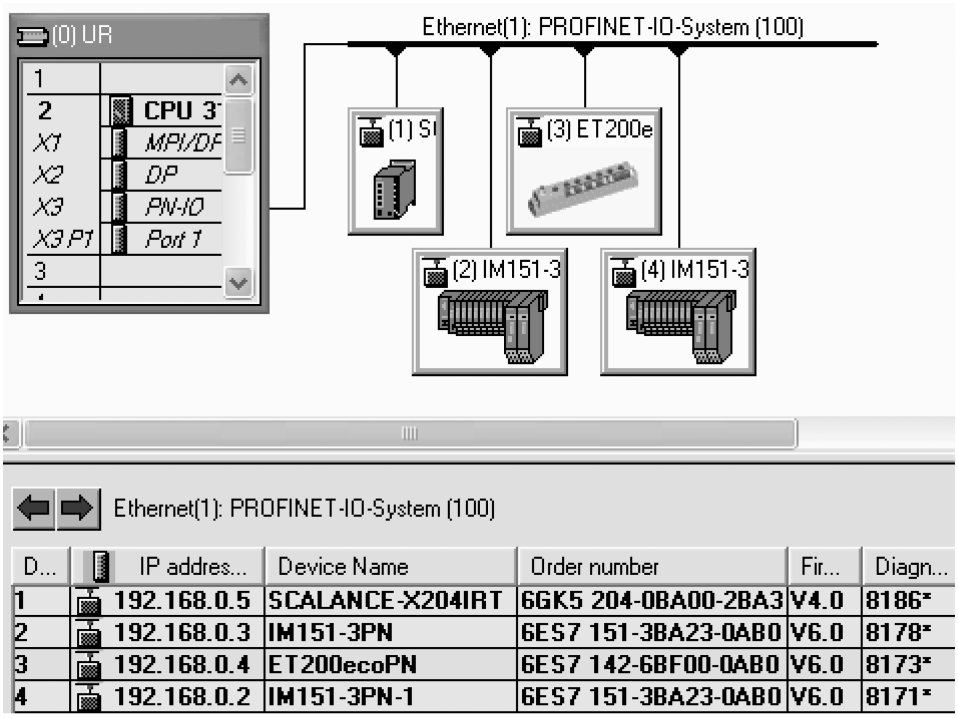



图 10-77　硬件组态

点击总线“Ethernet（1）：PROFINET-IO-System（100）”，右键点击弹出菜单选择“PROFINET IO Domain Management…”，弹出同步域管理对话框，双击“SIMATIC 300（1）/PN-IO”，即 IO 控制器 CPU319-3PN/DP，弹出设备属性对话框，在同步角色中选择“Sync master”，设置为同步时钟主站，如图 10-78 所示。

同样给 ET200S 和 ET200ECOPN 以及 SCALANCE X204IRT 交换机设置时钟同步从站。设置 IM151-3 为时钟同步从站如图 10-79 所示。

设置完毕后的同步域管理对话框如图 10-80 所示。

点击对话框中的“Details…”按钮，弹出同步域详细信息，其中黄色部分为 IRT 带宽预留部分，IRT 数据仅在这部分时间段内进行通信，空白部分是不使用的带宽，任何数据都不会出现在该时间段，亮绿色和绿色为 RT 数据和非 RT 数据（例如 TCP/IP）共享带宽部分。这里保持默认状态。点击“OK”结束设置。同步域详细信息如图 10-81 所示。

双击总线”Ethernet（1）：PROFINET-IO-System（100）”总线，弹出 PROFINET IO 系统属性对话框，在 Update Time 页更新时间为默认状态。PROFINET IO 系统属性如图 10-82 所示。

然后保存编译项目下载到 PLC 中进行 PROFINET IO 通信，具体组态 PROFINET IO 通信的详细信息请参考 10.5.1 节。这样就实现了 IRT 和 RT 同时在一根总线上通信。

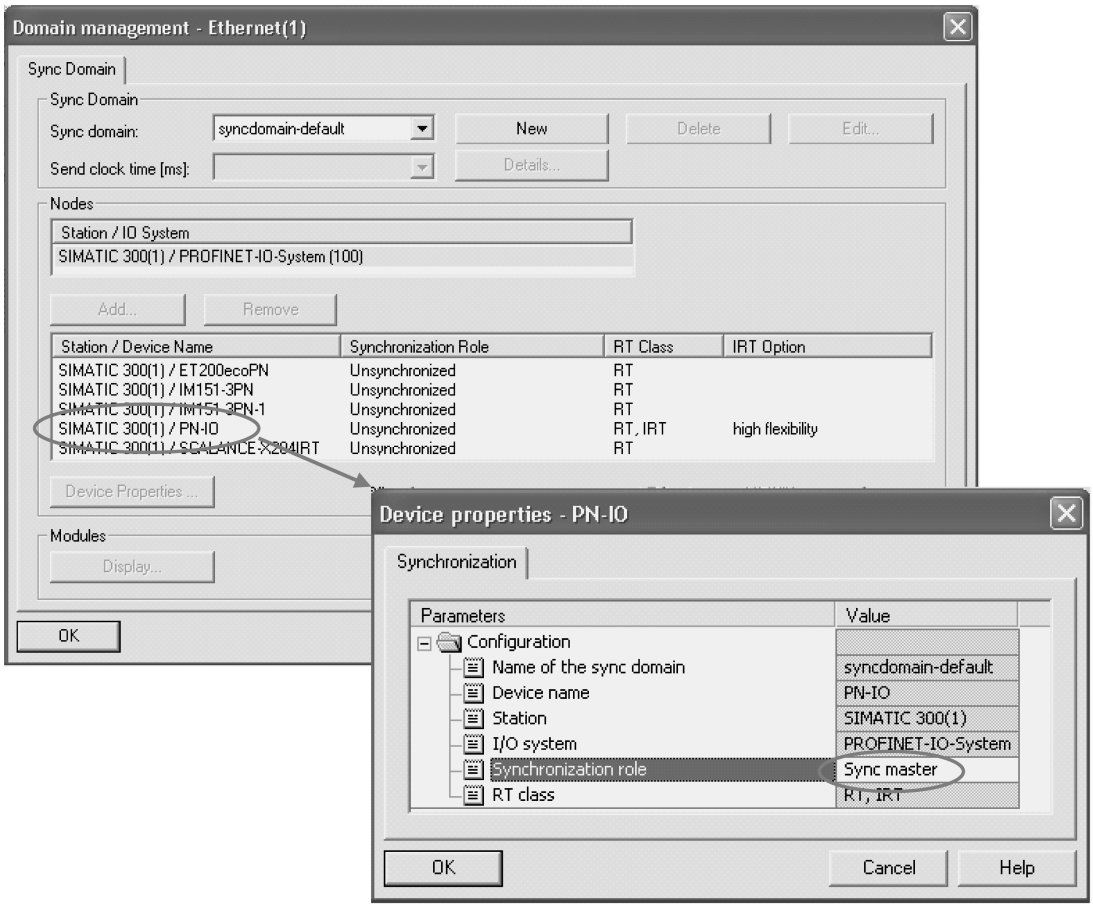



图 10-78　设置时钟同步主站

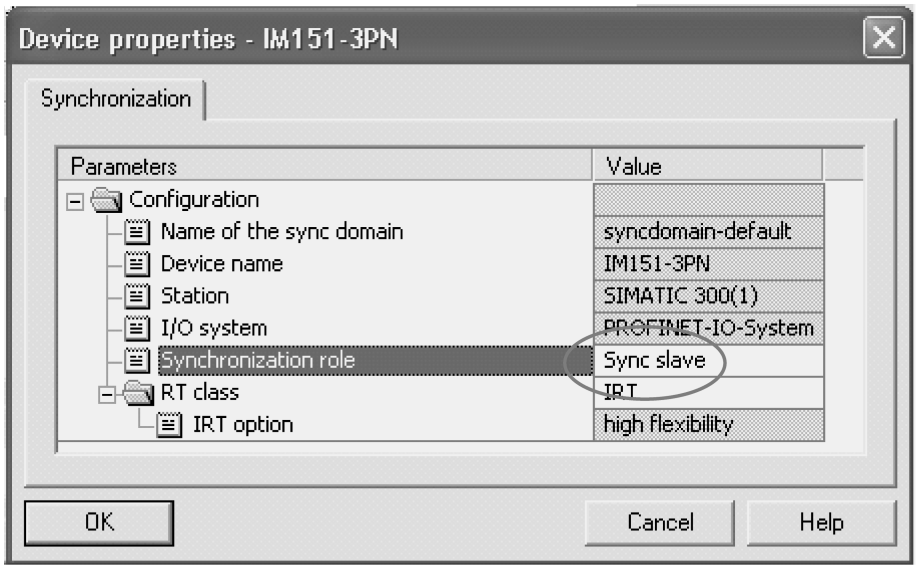



图 10-79　设置 IM151-3 为时钟同步从站

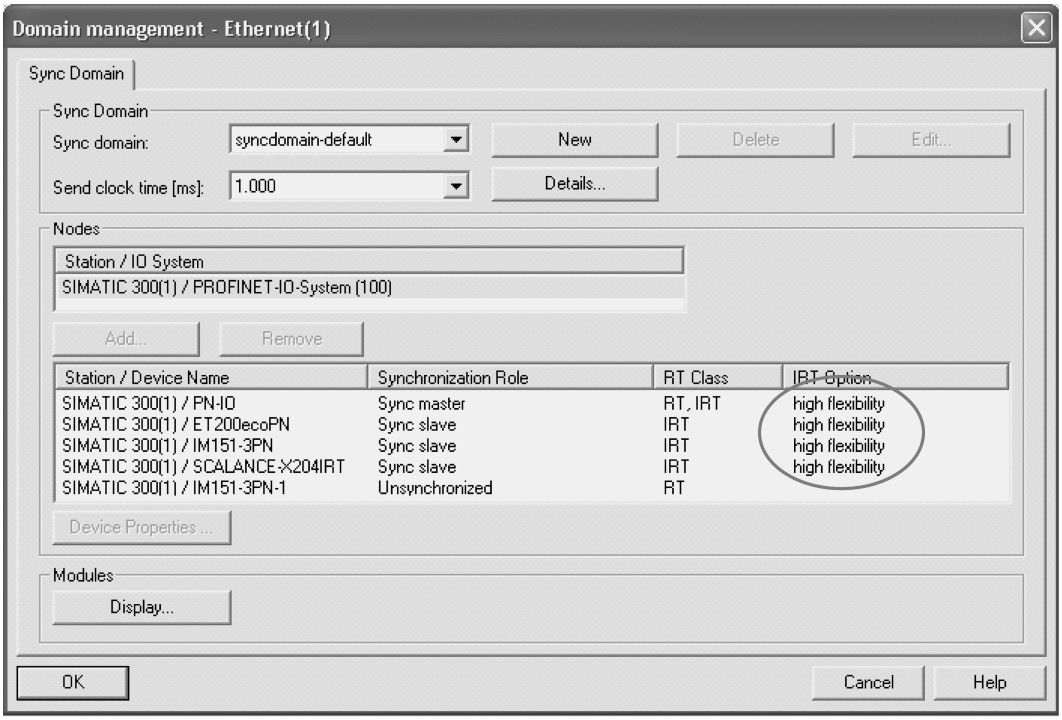



图 10-80　同步域管理对话框

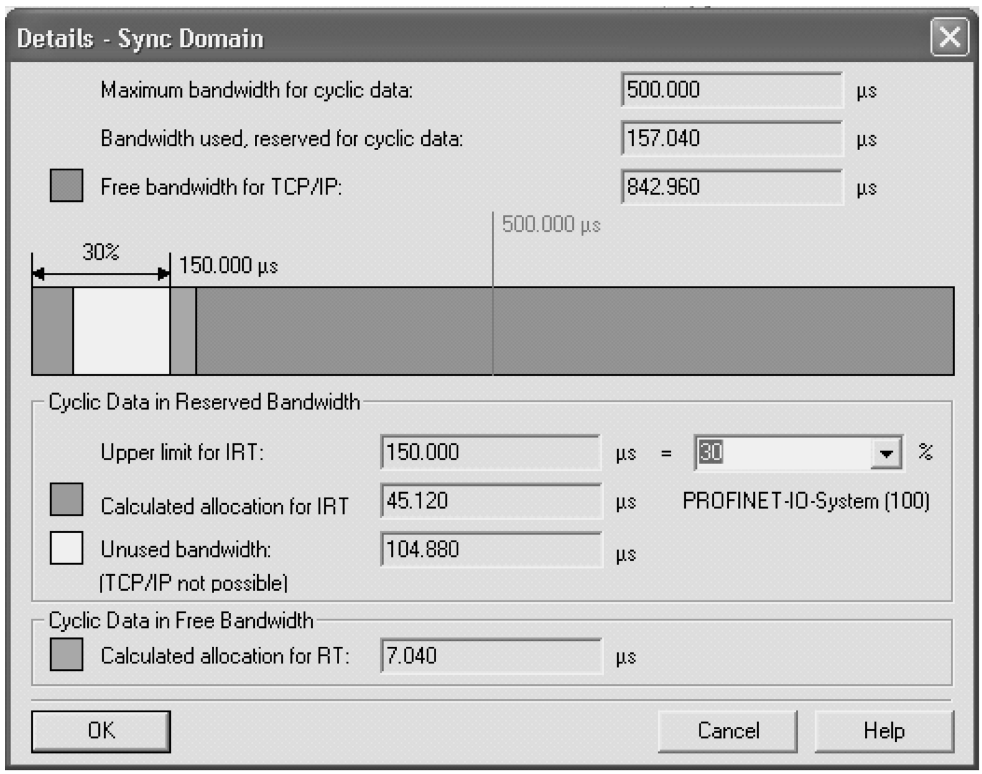



图 10-81　同步域详细信息

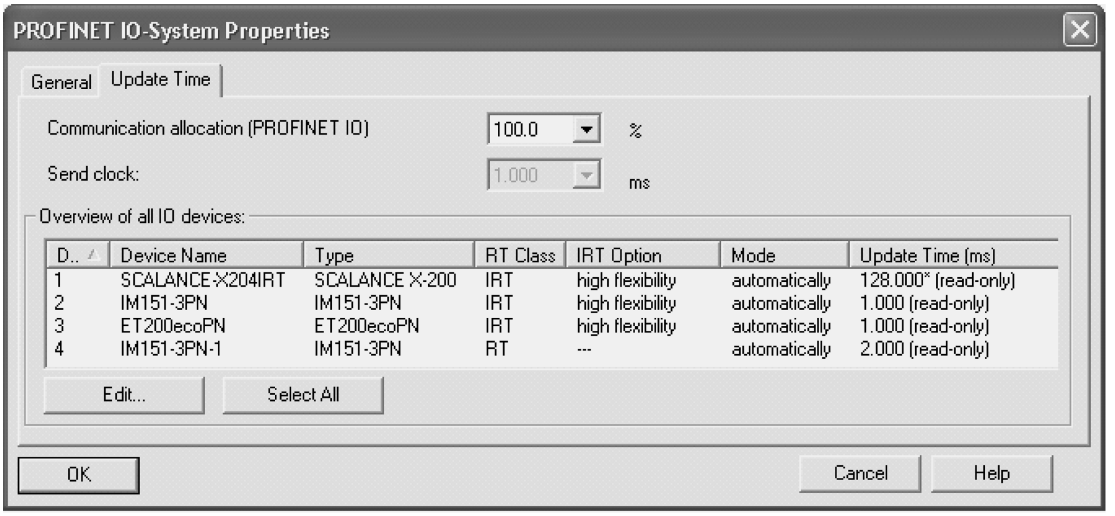



图 10-82 PROFINET IO 系统属性

### 10.5.7 PROFINET CBA 通信

PROFINET CBA 适合分布式智能站点之间的通信应用，能把大的控制系统分成不同功能、分布、智能、相对独立的控制单元，将控制单元封装生成功能组件，利用 iMAP 工具软件，连接各个组件进行通信。S7-300 系列 PLC 站点通过 CPU 集成的 PN 接口（例如 CPU31X-2DP/PN）或通过 CP343-1 可以作为一个组件，S7-400 系列 PLC 站点通过 CPU 集成的 PN 接口或通过 CP443-1ADV 也可以作为一个组件。下面以两个 CPU317-2DP/PN 为例介绍 PROFINET CBA 配置的步骤：

1）插入一个“SIMATIC 300 Station”，配置硬件，按安装次序依次插入机架、电源、CPU317—2DP/PN，在 CPU 属性中（“Cycle/Clock Memory”标签）配置通信的负载为 50%。

2）如果接口不使用 PROFINET IO 连接远程 I/O 站，可以不创建以太网网络。点击 PROFINET 网络栏，配置 CPU317—2DP/PN 的 IP 地址、子网掩码等参数。

3）如果 PN 接口既使用 PROFINET IO 方式又使用 PROFINET CBA 方式，需要插入“PROFINET IO SYSTEM”，出现图形化的 PROFINET 网络，双击 PROFINET 网络，弹出 PROFINET 属性对话框，点击“Update Time”标签配置更新时间及 PROFINET IO 与 PROFINET CBA 通信所占用实时通道的比例，比例的大小会影响数据的更新时间。PROFINET 属性界面如图 10-83 所示。配置完成后存盘编译。

4）配置通信接口区。单击站图标，按鼠标右键选择“Create Profinet Interface”进入编辑界面，通信接口区存储于数据块中。新建一个数据块，如 DB1 作为数据通信区，在通信区中分别建立 20 字节（数组变量）作为组件的输入和输出区，如图 10-84 所示：

“PN-Input”作为生成组件的输入，“PN-Output”作为生成组件的输出，“S7_Variable”作为与上位机通信变量，在组件输入输出接口中不显示。在地址栏显示通信区在数据块 DB1 中占用的地址。

5）生成组件。单击站图标，按鼠标右键选择“Create Profinet component”进入编辑界

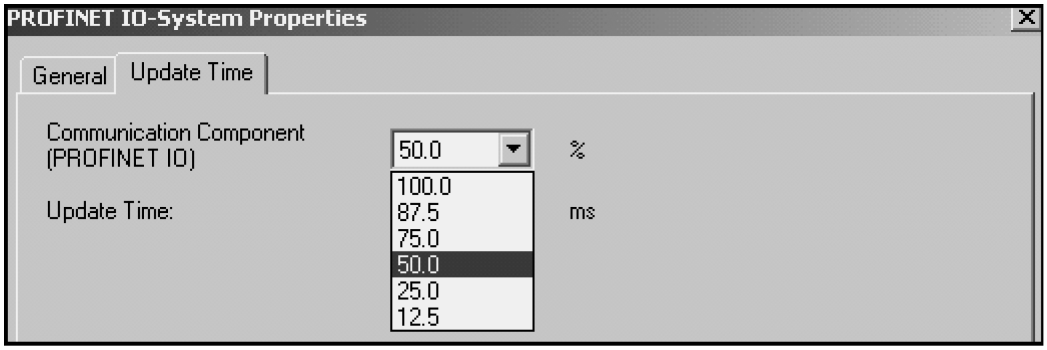



图 10-83　PROFINET 属性界面

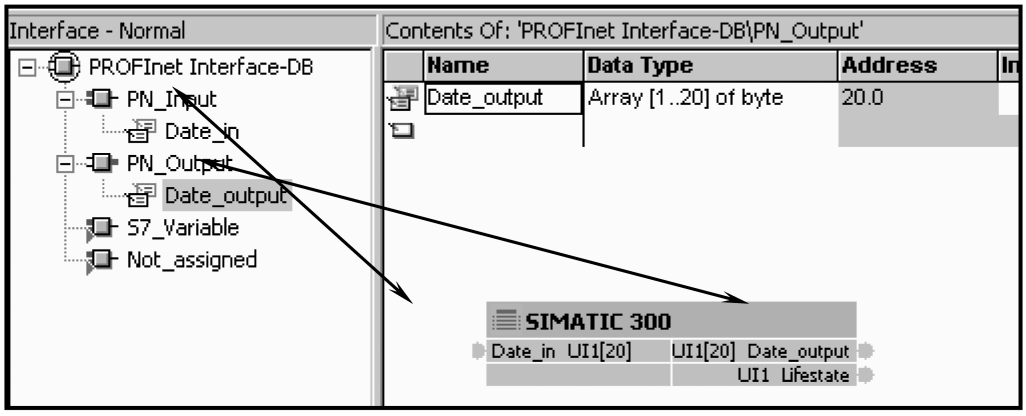



图 10-84　生成组件的通信接口区

面，配置组件的名称和类型及存放目录，如图 10-85 所示。

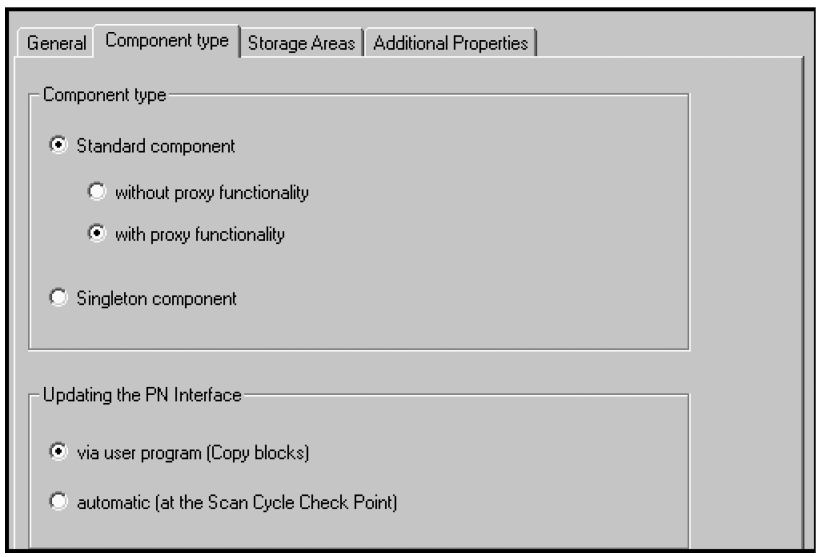



图 10-85　组件类型的选择

CPU317-2DP/PN 组件的类型分为标准组件和独立体组件。标准组件分为带有网络代理功能（可以连接 PROFIBUS 设备组件，并需要在程序中添加 OB82、OB85、OB86）和不带有网络代理功能的组件，如果加入 PROFINET IO 功能只能选择独立体组件。更新 PROFINET 组件通信接口区可以选择自动更新或调用系统函数 SFC112（读通信接口数据）和 SFC113（写输出到通信接口）。

6）以同样的方法生成另一个站的组件。然后将配置信息分别下载到 CPU 中。

7）打开 iMAP 软件，在“Project Library”窗口导入先前生成的组件文件（*.XML），并拖放到工作区。在“Net View”界面分别点击每个组件，在属性中定义 IP 地址，定义的 IP 地址与实际的地址必须匹配。在“Plant View”界面中连接组件的通信区，使用鼠标点击组件的输出，然后再点击另一个组件的输入，两者之间自动连接（数据类型必须匹配）。组件通信区的连接如图 10-86 所示。

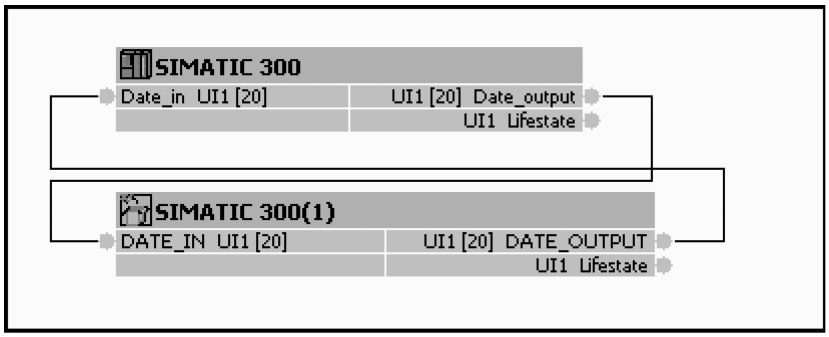



图 10-86 组件通信区的连接

8）在项目范围内设定传输频率。点击菜单“Project”→“Properties”弹出项目属性对话框，在“Interconnections”界面预先设定项目中所有连接的传输频率，传输模式分为周期和非周期，周期模式为实时通信。周期模式频率分快、中、慢三个档次，分别为每个档次的频率设定传输时间。在每个组件连接的属性中只能选择频率的档次而不能修改传输时间。

9）在 PG/PC 接口中设定下载使用的以太网网卡，在 iMAP 菜单栏选择“Online”→“Download All Instances”，下载所有配置信息后通信建立，这样两个 CBA 站点通过双方的 DB1 进行数据交换。

PROFINET CBA 实时通信只适合 PLC 间的通信，与 HMI 之间是非实时的通信，并且只能通过 OPC 的方式实现。组件中，“PN-Input”、“PN-Output”及“S7_Variable”类型的通信变量都可以生成 OPC 的“Item”。配置步骤如下：

1）在 iMAP 中完成组件通信的配置后，使用菜单命令“Options”→“Great OPC symbol file”，生成 OPC 的变量符号表，生成的文件保存在指定的目录下或存放在本项目“OPC-BASE”目录下。

2）在 SIMATIC Manager 配置 PC 站，插入 OPC 服务器和通信网卡，OPC 服务器不需要建立连接。

3）在桌面上通过快捷方式打开“Station Configurator”，相应插入“OPC Server”和以太网网卡，插入的槽号与在 SIMATIC Manage 中 PC 站的配置必须一致。配置完成后，将 PC 站的配置信息下载到“Station Configurator”编辑器中。

4）选择通信服务。使用视窗的菜单命令“Start”→“Simatic”→“SIMATIC NET”→“Settings”→“Configuration console”打开配置界面，在“OPC Server”子目录中选择“S7”和“PROFINET”服务。

5）选择 OPC 符号文件。在“Symbols”符号管理器中选择使用 iMAP 软件生成的 OPC 符号文件。至此，配置工作完成。

6）使用 OPC 客户端进行测试。点击视窗的菜单命令“Start”→“Simatic”→“SIMATIC NET”→“Industry Ethernet”→“PROFInet OPC Server ”→“OPC Scout”打开 OPC 客户端软件可以监控 OPC 服务器中的变量。

## 10.6　串行通信

串行通信主要用于连接调制解调器、扫描仪、条码阅读器等带有串行通信接口的设备，与西门子的传动装置的 USS 协议及 MODBUS RTU 也属于串行通信的范畴。

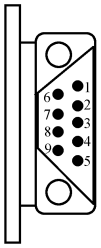
### 10.6.1　串行通信接口类型及连接方式

西门子串行接口有 RS-422/485、RS-232C、TTY 三种类型，TTY 电流环接口很少使用，这里不作介绍。

1. RS-232C（V.24）接口

RS-232C（V.24）的最大通信距离为 15m，只能连接单个设备，转换为 RS-485 接口可以连接多个设备。串行通信处理器 RS-232C（V.24）接口引脚定义如表 10-6 所示。

**表 10-6　RS-232 D9 连接器接口引脚定义**

| CP340 连接头 | 针脚 | 符号 | 输入/输出 | 说明 |
|---|---|---|---|---|
|  | 1 | DCD | 输入 | 数据载波检测 |
| | 2 | RXD | 输入 | 接收数据 |
| | 3 | TXD | 输出 | 发送数据 |
| | 4 | DTR | 输出 | 数据终端准备好 |
| | 5 | GND | — | 信号地 |
| | 6 | DSR | 输入 | 数据装置准备好 |
| | 7 | RTS | 输出 | 请求发送 |
| | 8 | CTS | 输入 | 允许发送 |
| | 9 | RI | 输入 | 振铃指示 |

RS-232C 为标准接口，每个设备接口引脚定义相同。以两个 CP340 RS-232C 接口为例，引脚的连接如图 10-87 所示。

2. RS-422/485（V.27）接口

RS-422/485（V.27）的最大通信距离为 1200m，RS-422/485 是一个 15 针串行接口，根据接线的方式可以选择 RS-422 或者 RS-485 接口，其中只有一个接口有效。RS-422 为 4 线制，全双工模式；RS-485 为两线制，半双工模式。RS-485 串行接口可以连接多个设备。串行通信处理 RS-422/485（V.27）接口引脚定义如表 10-7 所示。

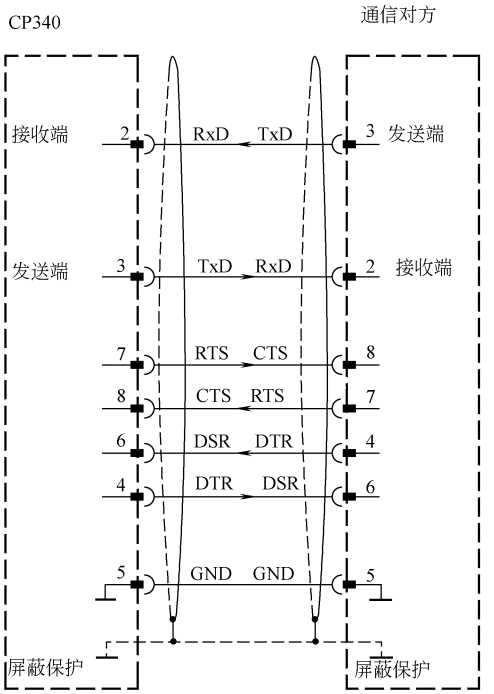
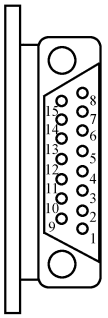



图 10-87　RS-232C 引脚连接

**表 10-7　RS-422/485 接口引脚定义**

| RS-422/485 连接头 | 针脚 | 符号 | 输入/输出 | 说明 |
|---|---|---|---|---|
|  | 1 | — | — | — |
| | 2 | T（A） | 输出 | 发送数据（四线模式） |
| | 3 | — | — | — |
| | 4 | R（A）/T（A） | 输入<br>输入/输出 | 接收数据（四线模式）<br>接收/发送数据（两线模式） |
| | 5 | — | — | — |
| | 6 | — | — | — |
| | 7 | — | — | — |
| | 8 | GND | — | 功能地（隔离） |
| | 9 | T（B） | 输出 | 发送数据（四线模式） |
| | 10 | — | — | — |
| | 11 | R（B）/T（B） | 输入<br>输入/输出 | 接收数据（四线模式）<br>接收/发送数据（两线模式） |
| | 12 | — | — | — |
| | 13 | — | — | — |
| | 14 | — | — | — |
| | 15 | — | — | — |

RS-422/485 为非标准接口（有的使用 9 针接头），每个设备接口引脚定义不同，使用 RS-422 接口引脚的连接如图 10-88 所示。

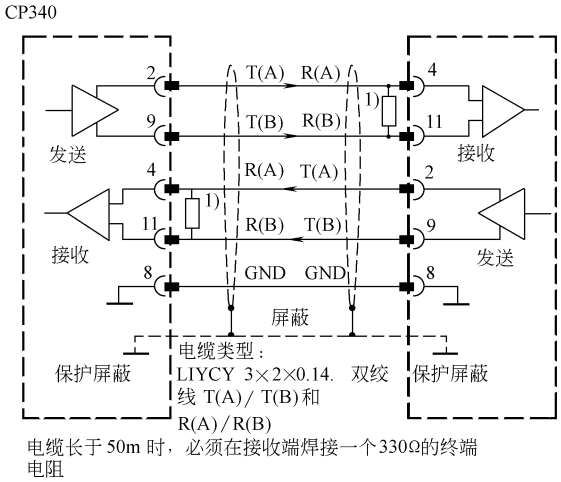



图 10-88　RS-422 接线方式（4 线制）

2、9 为发送端，连接通信方的接收端，即 T(A)-R(A)、T(B)-R(B)；引脚 4、11 为接收端，连接通信方的发送端，即 R(A)-T(A)、R(B)-T(B)。使用 RS-485 接口引脚的连接如图 10-89 所示。

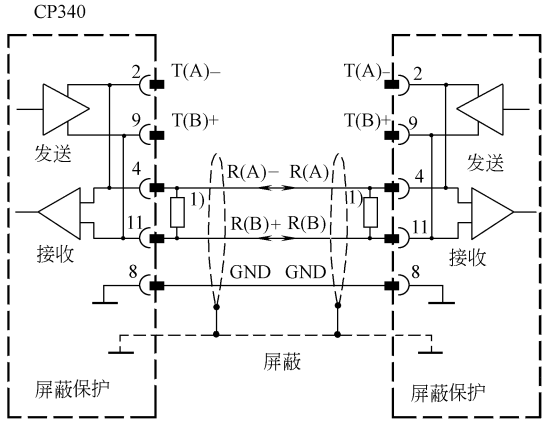



图 10-89　RS-485 接线方式（2 线制）

引脚 2、9 与 4、11 内部短接，不需要外部短接。引脚 4 为 R(A)，引脚 11 为 R(B)，通信双方的连线为 R(A)- R(A)，R(B)- R(B)。在通信过程中，发送和接收工作不可以同时进行，为半双工通信制。

**注意：**

有些厂商在串行通信接口引脚的标注上没有使用 R(A)、T(A)、R(B)、T(B)，而用 R－、T－、R＋、T＋来标注，在这里 R(A)＝R－、R(B)＝R＋、T(A)＝T－、T(B)＝T＋。

### 10.6.2 串行通信支持的通信协议

西门子串行接口通常支持 ASCII 码、3964（R）和 RK512 三种通信协议。ASCII 码较为常用，下面以 CP341 为例着重介绍 ASCII 码的参数化。

安装随模块携带的点到点软件参数化后进行硬件配置。插入 CP341 后双击模块，在弹出的对话框中选择“Parameter”按钮进入参数化界面，在“Protocol”栏中选择 ASCII，双击“Protocol”图标进入 ASCII 码参数化界面，如图 10-90 所示。

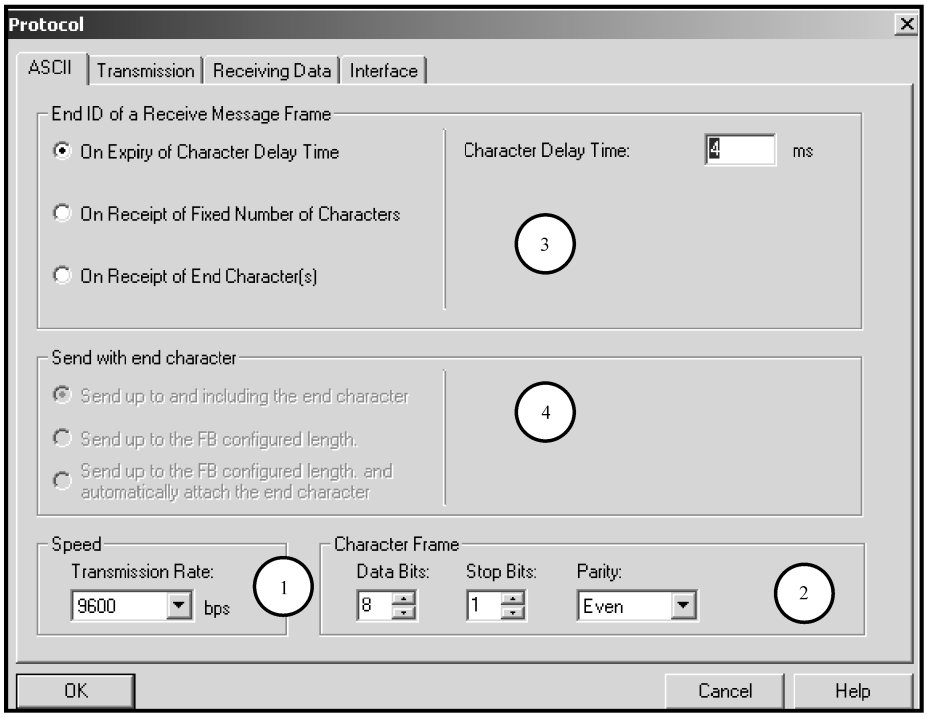



图 10-90 ASCII 参数化界面 ASCII 栏

ASCII 标签中的设置：

1）在区域① 中选择传送速率，通信双方的传送速率必须一致，传送速率要根据通信距离的长短进行选择 ，通常通信距离越长，所选的速率就越低。

2）在区域②中选择字符结构，根据通信方的要求进行参数化。

3）在区域③中选择接收报文的结束符，可以使用三种方法判断接收的报文是否结束：

- “On Expiry of Character Delay Time”

用两个字符的间隔时间来判断报文是否结束。缺省设置为 4ms，用户可以设置延迟时间。如果延迟时间超过所定义的时间，那么下一个字符将是下一帧报文的开始。

- “On Receipt of Fixed Number of Characters”

以固定的报文长度判断报文是否结束，缺省的设置为 240 个字节。当接收的实际字符数等于设置的字节数时判定一帧报文结束。接收报文中，如果两个字符的延迟时间超过定义的时间而接收的字符数没有达到参数化的字符数时，将终止接收并报错。

- “On Receipt of End Character（s）”

以结束符判断报文是否结束，结束符可以是一个也可以是两个。选择“On Receipt of End Character（s）”后，在区域③中出现结束符参数化界面，在区域④出现发送报文带有结束符的参数化界面。

在“Transmission”标签中对流量控制进行设置，在“Receiving Date”标签中对接收数据缓存区进行设置，这里不作介绍。在“Interface”标签中选择接口类型，如 RS-422 或 RS-485。

### 10.6.3　串行通信模块与相应的通信函数

西门子公司的 PLC 中有不同类型的串行通信处理器，每个通信处理器有与之相应的通信函数相匹配，对应关系如表 10-8 所示。

**表 10-8　使用 ASCII 协议通信处理器所需的通信函数**

| 通信功能块 / CP 类型 | 发送 | 接收 | 流量控制及辅助信号控制 | 有效接口 |
|---|---|---|---|---|
| S7-300 系列 | | | | |
| CP340 | FB3 | FB2 | FC5、FC6 | TTY、RS－232C、RS-422/485 |
| CP341 | FB8 | FB7 | FC5、FC6 | TTY、RS－232C、RS-422/485 |
| S7-300C PTP | SFB60 | SFB61 | — | RS-422/485 |
| S7-400 系列 | | | | |
| CP440 | FB10 | FB9 | — | RS-422/485 |
| CP441-1 | SFB12 | SFB13 | FB5、FB6 | TTY、RS－232C、RS-422/485 |
| CP441-2 | SFB12 | SFB13 | FB5、FB6 | TTY、RS－232C、RS-422/485 |
| 分布式 IO | | | | |
| ET200S Serial | FB3 | FB2 | FB4、5、6、7、8 | RS－232C、RS-422/485 |

### 10.6.4　通信函数的调用

以调用 CP340 的通信函数为例介绍串行通信数据的接收和发送。首先按照通信双方的约定配置 CP340 的接口参数，如通信速率、数据格式及接收方式，配置完成之后调用通信函数发送和接收数据。

1. 发送程序

在 OB1 中调用 FB3 P_SEND（“Libraries”→“CP PtP”→“CP340”）发送数据到通信方。

```
CALL    "P_SEND", DB3
  REQ     := M1.1
```

```
R       :=
LADDR   :=256
DB_NO   :=1
DBB_NO  :=0
LEN     :=10
DONE    :=M1.2
ERROR   :=M1.3
STATUS  :=MW2
```

FB3 P_SEND 参数含义：

REQ：　发送请求，每个上升沿发送一帧数据。

LADDR：CP340 的逻辑地址。

DB_NO：指定发送区（需要发送哪一个 DB 中的数据）。

DBB_NO：指定发送区在 DB 中的起始字节。

LEN：　发送字节的长度。

DONE：　发送完成输出一个脉冲。

ERROR：发送失败输出 1。

STATUS：FB3 调用的状态字。

例子程序中，如果 M1.1 产生一个上升沿，发送 DB1.DBB0 ~ DB1.DBB9 中存储的报文格式（根据通信方定义的报文格式发送数据）。

2. 接收程序

在 OB1 中调用 FB2 P_RCV 接收通信方发送的数据。

```
CALL    "P_RCV", DB25
  EN_R  :=TRUE
  R     :=
  LADDR :=256
  DB_NO :=2
  DBB_NO:=0
  NDR   :=M4.1
  ERROR :=M4.2
  LEN   :=MW6
  STATUS:=MW8
```

FB2 P_RCV 参数含义：

EN_R：　接收使能。

LADDR：CP340 的逻辑地址。

DB_NO：指定接收 DB。

DBB_NO：指定接收 DB 中的起始字节。

NDR：　接收新数据时输出一个脉冲。

ERROR：接收失败输出 1。

LEN：　输出实际接收字节的长度。

STATUS：FB2 调用的状态字。

接收到的数据将放到从 DB2. DBB0 开始的数据中。与串口设备通信，在大多数情况下采用由 PLC 根据通信方定义的报文格式发送数据请求，通信方响应并返回请求数据的通信方式。

### 10. 6. 5　MODBUS RTU 通信协议

MODBUS RTU 是基于串口（RS-232C、RS-422/485）的一种通信协议，协议开放，多用于连接现场仪表设备，通信距离与串行通信定义相同。由于报文简单、开发成本比较低，许多现场仪表仍然使用 MODBUS RTU 协议通信。MODBUS RTU 格式通信协议以主从的方式进行数据传输，主站发送数据请求报文到从站，从站返回响应报文。不同数据区的交换是通过功能码（Function Code）来控制的，有些功能码是对位操作的，通信的用户数据是以位（bit）为单位的，例如：

FC01 读输出位的状态；

FC02 读输入位的状态；

FC05 强制单一输出位；

FC15 强制多个输出位。

有些功能码是对 16 位寄存器操作的，通信的用户数据是以字（word）为单位的，例如：

FC03 读输出寄存器；

FC04 读输入寄存器；

FC06 写单一输出寄存器；

FC16 写多个输出寄存器。

这些功能码是对位输入（0XXXX 数据区）、位输出（1XXXX 数据区）、寄存器输入（4XXXX 数据区）、寄存器输出（3XXXX 数据区）进行访问的，功能码与数据区的对应关系如表 10-9 所示。

**表 10-9　功能码访问的数据区**

| 功能码 | 数据 | 数据类型 | | 访问 | 用户级的地址表示法（十进制） |
|---|---|---|---|---|---|
| 01、05、15 | 输出的状态 | 位 | 输出 | 读、写 | 0XXXX |
| 02 | 输入的状态 | 位 | 输入 | 只读 | 1XXXX |
| 03、06、16 | 输出寄存器 | 16 位寄存器 | 输出寄存器 | 读、写 | 4XXXX |
| 04 | 输入寄存器 | 16 位寄存器 | 输入寄存器 | 只读 | 3XXXX |

用户也可以使用西门子公司的串行通信处理器，根据 MODBUS RTU 报文格式编写通信程序，但是程序比较繁琐，这里不作介绍。西门子公司提供的 MODBUS RTU 协议转换卡，可以插入 CP341 和 CP441-2 通信处理器中作为 MODBUS RTU 主站或从站，MODBUS RTU 协议转换卡将报文进行校验（符合 MODBUS RTU CRC 校验），并将用户数据传送到 CPU 中。同样，发送的用户数据经过协议转换卡封装（带有校验码）后符合协议规定的报文格式，使用 MODBUS RTU 协议转换卡可以使通信更为容易。下面分别以 CP341 为例，介绍

MODBUS RTU 主站和从站的使用方法。

1. MODBUS RTU 主站

主站发送请求报文，从站响应报文，报文格式如表 10-10 所示。

**表 10-10 MODBUS RTU 报文格式**

| ADDRESS | FUNCTION | DATA | CRC CHECK |
|---|---|---|---|

报文格式解释如下，

ADDRESS： 从站地址，占用一个字节。

FUNCTION： 功能码，占用一个字节。

DATA： 信息数据，N 个字节，格式根据功能码来定义。

CRC CHECK： CRC 校验，两个字节，被协议转换卡过滤，不需要考虑。

主站支持的功能码有：FC01、FC02、FC03、FC04、FC05、FC06、FC07、FC08、FC11、FC12、FC15、FC16，先前介绍的功能码处理数据的通信，FC07、FC08、FC11、FC12 是对通信状态的处理。例如主站利用功能码 FC04 读出 5 号从站寄存器 30016 ~ 30018 存储的数据，需要发送的报文为（十六进制）：

05H： 从站地址；

04H： 功能码为 04；

00H： 寄存器开始高字节；

10H： 寄存器开始低字节；

00H： 读寄存器的个数高字节；

03H： 读寄存器的个数低字节；

XXH： CRC 校验高字节；

XXH： CRC 校验低字节。

用户编写程序时，从通信处理器 CP 发送的数据为 05 04 00 10 00 03，CRC 校验通过协议转换卡自动加载到报文中。

从站响应的报文为：

05H： 从站地址；

04H： 功能码为 04；

04H： 计数字节；

31H： 寄存器 30016 高字节；

32H： 寄存器 30016 低字节；

33H： 寄存器 30017 高字节；

34H： 寄存器 30017 低字节；

35H： 寄存器 30018 高字节；

36H： 寄存器 30018 低字节；

XXH： CRC 校验高字节；

XXH： CRC 校验低字节。

用户编写接收程序时，从通信处理器 CP 只能读回用户请求的数据 31 32 33 34 35 36，

其他数据被协议转换卡过滤掉。

按上面的例子配置 CP341 MODBUS RTU 主站的过程如下：

1）插入一个“SIMATIC 300 Station”，配置硬件，按安装次序依次插入机架、电源、CPU 和 CP341 模块。

2）双击 CP341 模块，在弹出对话框中点击“Parameter”按钮进入协议选择界面，选择“MODBUS Master”驱动，在联机状态，双击“Load Drives”图标，加载 MODBUS RTU 协议到协议转换卡（Dongle）中。

3）双击“Protocol”图标，弹出协议配置对话框，在“MODBUS Master”标签页中设置通信速率、数据格式（数据位、奇偶校验位、停止位）等参数，设置的参数必须与从站相匹配，其他参数保持默认值。

4）编写程序发送数据请求。在 OB1 中调用 FB8 P_SND_RK（“Libraries”→“CP PtP”→“CP341”）的示例程序如下：

```
L     5
T     DB1. DBB      0    //从站地址
L     4
T     DB1. DBB      1    //FC04 功能码
L     16                 //开始地址 30016
T     DB1. DBB      3    //低字节为 DB1. DBB3
L     3                  //读出的数量为 3
T     DB1. DBB      5    //低字节为 DB1. DBB5
CALL "P_SND_RK" , DB8    //调用 CP341 发送函数块
SF          : =
REQ         : = M1. 1    //上升沿触发
R           : =
LADDR       : = 256      //CP341 逻辑地址
DB_NO       : = 1        //发送 DB1 中前 6B，发送数据中包含数据请求报文格式
DBB_NO      : = 0
LEN         : = 6
R_CPU_NO    : =
R_TYP       : = ‘X’      //必须为大写的‘X’
R_NO        : =
R_OFFSET    : =
R_CF_BYT    : =
R_CF_BIT    : =
DONE        : = M1. 2    //发送成功后产生一个上升沿
ERROR       : = M1. 3    //故障位信号
STATUS      : = MW2      //如果故障位为 1，可从状态字查看故障信息
```

程序调用后，M1. 1 产生一次上升沿，数据请求报文发送一次。

5）编写程序接收从站数据。在 OB1 中调用 FB7 P_RCV_RK（“Libraries”→“CP PtP”

→“CP341”）的示例程序如下：

```
CALL     "P _RCV_RK", DB7
EN_R       : = TRUE          //接收使能
R          : =
LADDR      : = 256           //CP341 逻辑地址
DB_NO      : = 2             //接收数据块
DBB_NO     : = 0             //接收区的开始地址
L_TYP      : =
L_NO       : =
L_OFFSET   : =
L_CF_BYT   : =
L_CF_BIT   : =
NDR        : = M4. 1         //接收到新的数据后，产生一个上升沿
ERROR      : = M4. 2         //故障位信号
LEN        : = MW6           //接收数据长度
STATUS     : = MW8           //如果故障位为 1，可从状态字查看故障信息
```

接收的从站数据 05 04 00 10 00 03 存储于 DB2. DBB0 ~ DB2. DBB5 中。由于主站数据请求和从站的响应是同步操作，如果访问多个从站数据，需要在主站中编写轮询程序。

2. MODBUS RTU 从站

S7 PLC 作为 MODBUS RTU 从站，只需要将 S7 CPU 的数据区地址与 MODBUS 数据区地址相对应，即可建立通信，例如主站需要读出 5 号从站（S7 PLC）寄存器 30016 ~ 30018 存储的数据，在从站 S7 PLC 中的配置过程如下：

1）插入一个“SIMATIC 300 Station”，配置硬件，按安装次序依次插入机架、电源、CPU 和 CP341 模块。

2）双击 CP341 模块，在弹出对话框中点击“Parameter”按钮，进入协议选择界面，选择“MODBUS Slave”驱动，在联机状态时，双击“Load Drives”图标，加载 MODBUS RTU 协议到协议转换卡（Dongle）中。

3）双击“Protocol”图标，弹出协议配置对话框，在“MODBUS Slave”标签页中设置通信速率、数据格式（数据位、奇偶校验位、停止位）及从站站号（站号为 5）等参数，设置的参数必须与主站相匹配。

4）查询表 10-9，从站数据区 3XXXX 对应功能码为 FC04，点击“FC04”标签，设置通信接口区为 DB10，如图 10-91 所示，这样从站数据区 3XXXX 对应 S7 的数据区为 DB10，主站需要读的寄存器 30016 ~ 30018 对应 S7 CPU 的数据区为 DB10. DBW30、DB10. DBW32、DB10. DBW34。

5）调用从站初始化程序。使用菜单命令“File”→“Open”→“Sample projects”，打开 MODBUS 从站配置软件提供的示例程序“MODSL”，在示例程序中将 S7-300 系列 PLC 从站的程序块（不包括“System Data”）复制到应用项目中，打开 OB1，将参数“LADDR”修改为 CP341 实际配置的逻辑地址，如果复制的示例程序与项目中使用的数据区、程序块号冲突，应作相应修改。程序下载后通信建立。

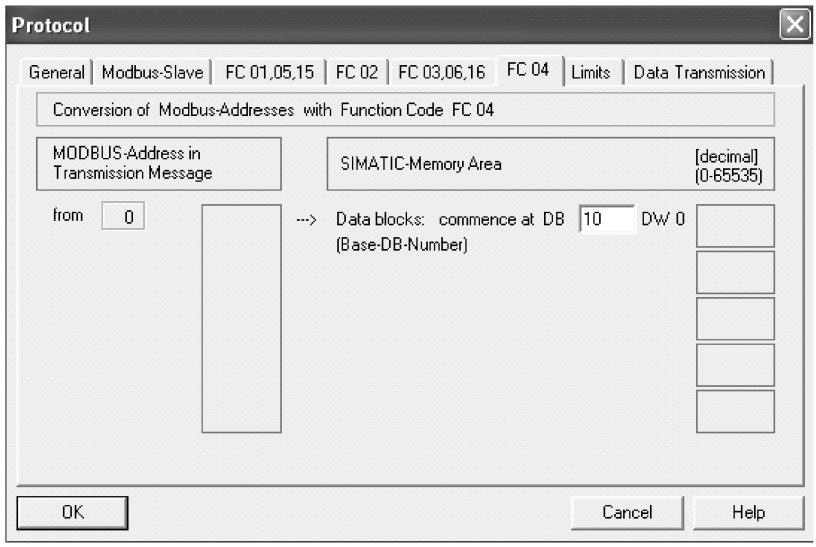



图 10-91　设置 MODBUS 从站通信数据区

# 第 11 章　功能模块的使用

## 11.1　高速计数器模块

### 11.1.1　高速计数器的应用场合

通常情况下，将数字量输入信号或 PLC 执行程序产生的脉冲信号作为 CPU 集成计数器的信号源，如果输入频率较高，则必须计算程序的扫描周期。例如 CPU 的扫描周期为25ms，数字量输入信号从 0 到 1，从 1 到 0，需要 CPU 扫描两次，则计数的响应时间为 50ms，计数频率最高为 20Hz；如果输入频率超出最高计数频率，则部分输入的脉冲信号将丢失，PLC 不能保证计数器的准确性，这时就需要使用高速计数器模块完成计数任务。

### 11.1.2　高速计数器的原理

高速计数器自带处理器，不占用主 CPU 的处理时间，高速计数功能以及一些快速响应功能在高速计数器内部完成，快速响应直接通过高速计数器触发。例如计数值达到设定的比较值时，通过高速计数器模块自身带有的输出点快速响应外部事件，如果将计数值读到 CPU 中处理，再通过输出点控制外围设备，响应时间往往达不到控制的要求，读到 CPU 中的计数值通常只作为参考或控制一些响应慢的设备。CPU 通过调用函数与高速计数器进行通信，例如设定高速计数器的计数模式、比较值等参数，可以读出当前的计数值及计数器内部的状态信息。

### 11.1.3　高速计数器可以连接的信号

高速计数器连接的信号可分为以下几种类型：

1）24V 脉冲信号，没有方向信号，只有一个输入端，连接高速计数器通道的 A 端作为输入信号，如图 11-1 所示。

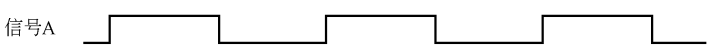



图 11-1　24V 脉冲信号

2）24V 脉冲信号和方向信号，如图 11-2 所示。

信号 A 作为高速脉冲输入信号，信号 B 作为计数的方向信号，为 0 时，向上计数；为 1 时，向下计数。

3）5V 增量脉冲编码器输出信号，输入信号为带有 A、B 相对称脉冲串，A、B 相位相差为 90°，N 为零脉冲，如图 11-3 所示。通常情况下，编码器每转 1 圈输出一个零脉冲信号，计数的方向与 A、B 的相位有关，默认情况下，A 的相位比 B 的相位提前，为向上计数，

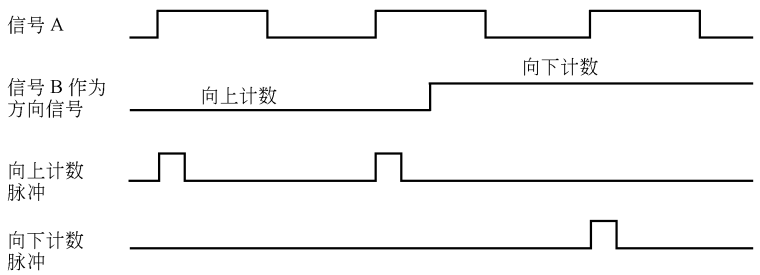



图 11-2　24V 脉冲信号和方向信号

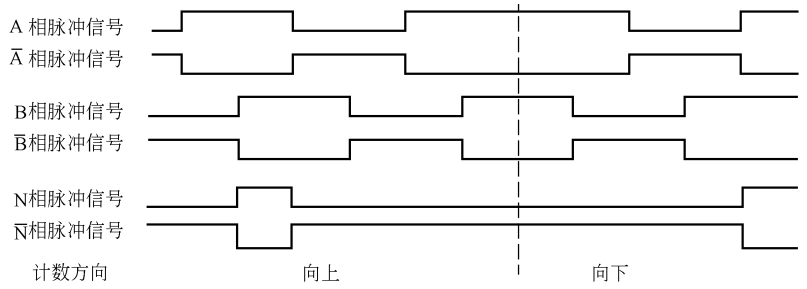



图 11-3　5V 增量脉冲信号

反之为向下计数，计数的方向可以通过参数修改。

4）24V 增量脉冲编码器输出信号，输入信号为 A、B 相脉冲串，A、B 相位相差为 90°，不需要连接零脉冲信号，如图 11-4 所示。

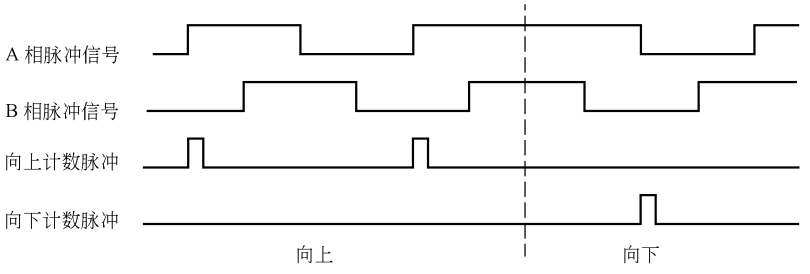



图 11-4　24V 增量脉冲信号

计数方向与 A、B 的相位有关，默认的情况下，A 的相位比 B 的相位提前，为向上计数，反之为向下计数。计数的方向可以通过参数修改。

**注意：**

1）编码器脉冲输出信号的类型（PNP 和 NPN）必须与高速计数器输入通道的类型相匹配。

2）连接 24V 增量编码器，在参数配置中选择脉冲和方向类型信号，在一些情况下，可能会出现偏差错误，如传动带拖动电动机，手压传动带时电动机向前微动，计数器向上计

数，手松开时，电动机回到原位置，而计数器没有向下计数回到原值，造成电动机位置的偏移，在振动的情况下，也有类似的情况发生，将配置修正即可避免上述问题。

### 11.1.4 脉冲信号的采集方式

高速计数器采集脉冲信号时，可以选择下面三种方式。

1. 单倍频信号的采集方式（默认方式）

图 11-5 所示为选择单倍频信号的采集方式，向上计数时，高速计数器捕捉 A 相信号的上升沿（B 相信号电平为 0）进行计数，向下计数时，高速计数器捕捉 A 相信号的下降沿（B 相信号电平为 0）进行计数。

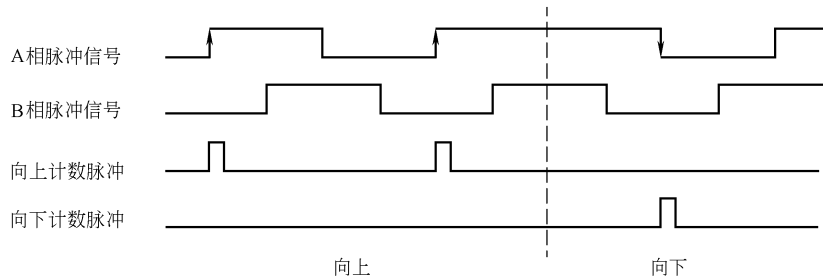



图 11-5 单倍频信号的捕捉方式

2. 双倍频信号的采集方式

图 11-6 所示为选择双倍频信号的采集方式，高速计数器捕捉 A 相信号的上升沿和下降沿，计数方向取决于 A、B 相脉冲的相位。

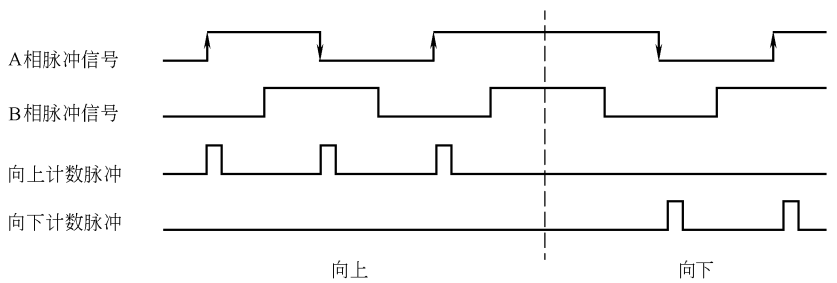



图 11-6 双倍频信号的捕捉方式

3. 四倍频信号的采集方式

图 11-7 所示为选择四倍频信号的采集方式，高速计数器同时捕捉 A、B 相信号的上升沿和下降沿，计数方向取决于 A、B 相脉冲的相位。

选择不同信号的采集方式，将影响高速计数器的计数值，例如脉冲编码器一圈发出 1024 个脉冲，如果选择单倍频信号的采集方式，旋转一圈计数值为 1024；如果选择双倍频信号的采集方式，计数值将为 2048；设定高倍频信号采集方式，可以增加计数器的分辨率。

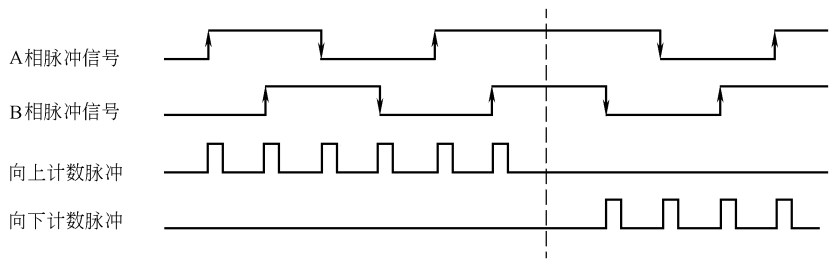



图 11-7　四倍频信号的捕捉方式

### 11.1.5　高速计数器的计数模式

高速计数器有三种计数模式：单次计数、连续计数和周期计数。

1．单次计数

• 如果没有激活上下计数方向时，从装载值开始计数，当达到计数的上下限并接收到新的脉冲信号时，跳回到装载值，高速计数器模块不再计数。开始计数信号（门信号）必须被再次激活后才能进行新的计数任务。

• 激活向上计数方向时，计数从装载值开始计数，当达到计数的上限并接收到新的脉冲信号时，跳回到装载值，高速计数器模块不再计数，如果重新开始计数信号（门信号），则必须被再次激活后才能进行新的计数任务。

• 激活向下计数方向时，计数从装载值开始计数，当达到计数的下限并接收到新的脉冲信号时，跳回到装载值，高速计数器模块不再计数，如果重新开始计数信号（门信号），则必须被再次激活后才能进行新的计数任务。

2．连续计数

计数值达到上限并接收到新的脉冲时，计数值跳到下限重新开始计数；计数值达到下限并接收到新的脉冲时，计数值跳到上限重新开始计数。计数值上下限的跳转不会丢失脉冲信号，计数值的上下限可以设置。

3．周期计数

• 如果没有激活上下计数方向时，计数从装载值开始计数，当达到计数的上下限并接收到新的脉冲信号时，跳回到装载值重新开始计数，这也是周期计数的默认设置。

• 激活向上计数方向时，从装载值开始计数，当达到计数的上限并接收到新的脉冲信号时，跳回到装载值重新开始计数。

• 激活向下计数方向时，从装载值开始计数，当达到计数的下限并接收到新的脉冲信号时，跳回到装载值重新开始计数。上下限跳转不会丢失脉冲信号。

**注意：**

装载值为用户设定的计数开始值，默认状态下装载值为 0。

### 11.1.6　高速计数器开始计数的条件

高速计数器通常有三种可设置的开始计数条件：1）不需要门信号；2）软件门信号；3）硬件门信号。“门信号”是开始计数的条件，可以是一个 PLC 内部的位信号（软件门）或集成在高速计数器上的数字量输入信号（硬件门），只有在门打开时，接收到的脉冲信号

才被记录下来。有的硬件门可设置为高电平控制信号，即信号为 1 时，门打开，如图 11-8 所示。

在门内的 4 个脉冲信号被记录下来，所有的软件门均采用高电平控制信号。有的硬件门可设置为脉冲沿控制信号，如图 11-9 所示。

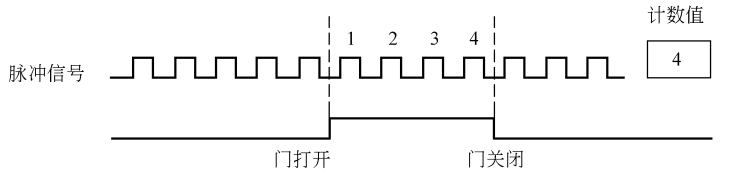



图 11-8 门信号 – 电平信号

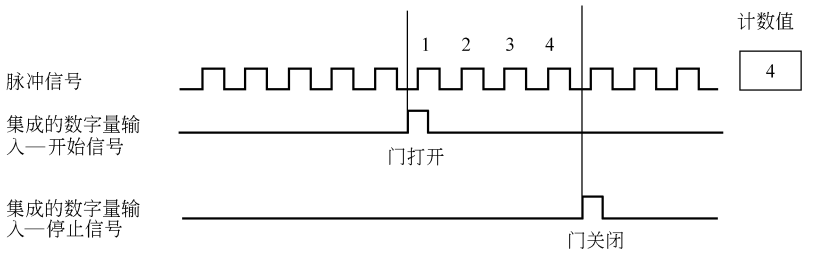



图 11-9 硬件门信号 – 脉冲沿信号

触发硬件门的沿信号分别为计数器模块集成的两个数字量输入信号，通过信号的变化改变门的开关。

1）不需要门信号。在连续计数模式时，可以选择“NO GATE”，只要接收到脉冲信号，计数器开始计数。

2）软件门信号。软件门信号是由 CPU 发出的开始计数命令，通过调用通信块将命令传递到计数器模块中，软件门信号可以是 CPU 中存在的任意一个位信号。软件门信号通常适合对计数开始条件没有快速响应的应用。

3）硬件门信号。开始计数的信号由集成在高速计数器上的数字量输入信号触发，硬件门信号通常适合对计数开始条件需要快速响应的应用，例如测量快速移动物体的长度，如果通过普通的数字输入信号触发，包括输入延时时间、CPU 扫描时间及 CPU 与计数器的通信时间，时间滞后将影响测量的精度，使用硬件门信号可以最大限度地提高测量精度。

### 11.1.7 高速计数器的其他功能

高速计数器利用计数功能，再加上其他的设置可以完成一些辅助功能，例如通过集成的数字输入信号可以实现锁存功能、测量两个输入信号的间隔时间，在计数的基础上设定时间窗口测量频率值等。

### 11.1.8 具有高速计数功能的模块

上面介绍了计数器的计数模式和功能，西门子公司的 PLC 中有许多模块都具有高速计

数功能，见表 11-1。

**表 11-1　具有高速计数功能的模块**

| 功　能 | 产　品 | 解 决 方 案 |
|---|---|---|
| 计数频率最高为 20kHz | S7-200 | 集成计数功能 |
| 计数频率最高为 60kHz | S7-300C、C7 | 集成计数功能 |
| 计数频率最高为 100kHz（24V）、500kHz（5V） | 1COUTER | ET200S 计数模块 |
| 计数频率最高为 200kHz（24V）、500kHz（5V） | FM350-1、FM450-1 | 功能模块 |
| 计数频率最高为 20kHz（24V），8 个通道 | FM350-2 | 功能模块 |
| 计数功能（自由编程） | FM352-5 | 应用模块 |
| 计数功能（自由编程） | FM458-1DP | 应用模块 |
| SSI（同步串行接口）编码器信号 | 1 SSI | ET200S 接口模块 |
| SSI（同步串行接口）编码器信号 | SM338 | 接口模块 |

SSI 编码器以通信的方式将当前的位置信号传送到接口模块中，由于是机械结构，再次上电后计数值不会丢失，增量编码器信号是电压脉冲信号，掉电后计数值不会保留，通过程序可以保留掉电前的计数值，如果掉电后位置发生变化，保留的计数值与实际值将产生偏差。除表 11-1 中列出的高速计数功能的模块外，还有一些模块具有计数功能，如 FM351、FM352、FM354 等定位模块，计数器只作为位置环的反馈信息，在特定的操作模式下，可以将计数值读出，但是比较繁琐。下面介绍几种常见的高速计数器的使用。

**注意：**

不是每一种计数模块都具有上述介绍的功能，选型时请参考样本手册。

## 11.1.9　FM350-1 高速计数器的使用

FM350-1 高速计数器模块可以连接 24V 增量编码器或 5V 增量编码器，一个通道连接的信号源可以是 PNP 型，也可以是 NPN 型。随硬件包装的光盘中，包含模块配置软件包和电子手册。在 STEP7 中配置 FM350-1 模块时，必须预先安装配置软件。以连接 24V 编码器为例，介绍 FM350-1 模块的使用。

1. FM350-1 的接线

FM350-1 的端子接线如图 11-10 所示。将 24V 增量编码器的输出 A、B 端连接到 FM350-1 的 6、8 号端子上，端子 3、5 可以给编码器提供 24V 电源，端子 13、14、15 为模块集成的数字量输入信号，分别为硬件门的启停信号和同步信号，端子 17、18 为模块集成数字量输出信号，用于两个比较器快速响应输出（设定比较值，当计数值接近设定的比较值时，触发输出），连接类型不同的编码器，需要在模块的侧面设定跳线开关，如图 11-11 所示。跳线开关分为

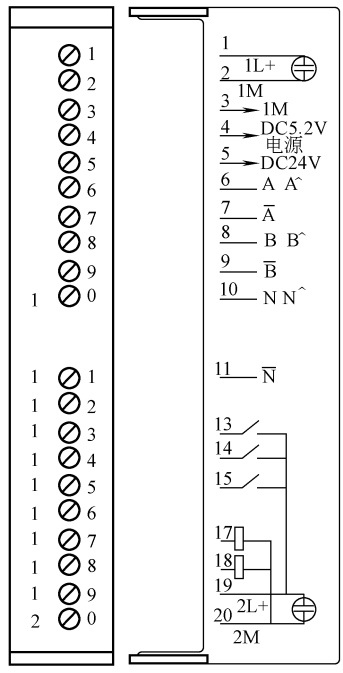



图 11-10　FM350-1 端子接线图

A、D 两个方向，开关指向 A，表示连接 5V 编码器信号；开关指向 D，表示连接 24V 编码器信号，本例中将调节跳线开关指向 D。

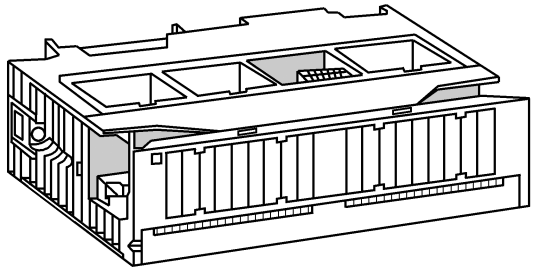


图 11-11　选择连接编码器类型

2. FM350-1 的硬件配置

在 SIMATIC Manager 硬件配置中，双击 FM350-1 模块，进入配置界面，点击“Encoders”图标，选择连接的编码器类型，如图 11-12 所示。选择“24V Incremental”，单倍频计数；在“Sensor inputs”选项中，可以选择连接信号源的类型，如源流入或源输出；在“Max. Count Frequency”选项中，选择输入信号的最大计数频率，如果输入频率低于 20kHz 时，应选择“20kHz”选项，增加信号的滤波时间。

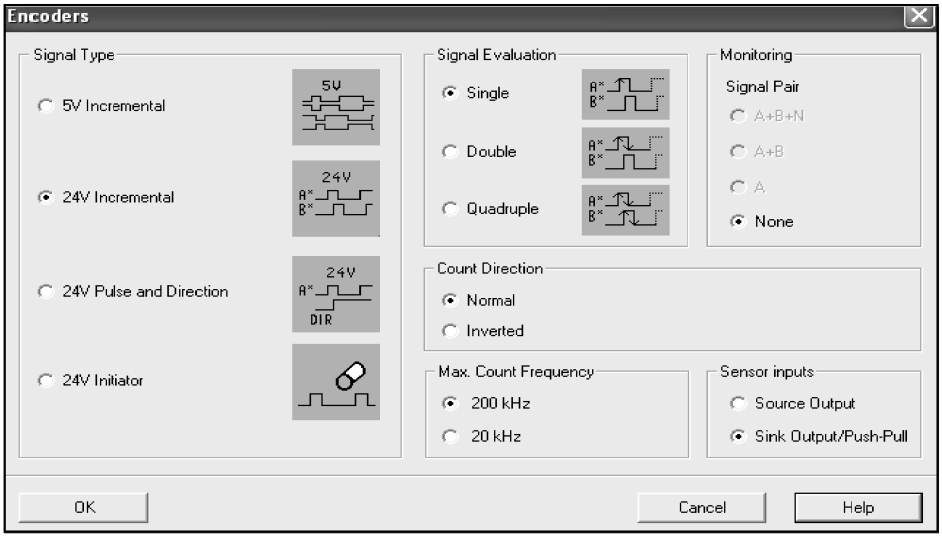



图 11-12　FM350-1 编码器选择界面

在操作模式选择界面中，可以选择计数范围、操作模式及门控制等参数，如图 11-13 所示。

图中，选择连续计数和软件门控制，如果选择硬件门控制，则需要在“Inputs”界面中进行配置，在“Inputs”界面中，还可以设置 DI2（模块集成的第三个数字输入）的同步功能，通过集成的输入信号纠正当前的计数值，当输入信号触发时，将设定值作为当前的计数值，应用于快速移动的物体通过某一开关时，对当前计数值的快速设定，而不会产生大的偏差。在模块的属性中，如果选择产生中断，则“Hardware Interrupt Enable”界面被激活，可以选择产生中断的事件，中断将调用 OB40（硬件中断）。在“Outputs”界面中，可以选择触发集成快速输出的事件，这些事件与设置的比较值有关，如图 11-14 所示。配置完成后编译存盘。

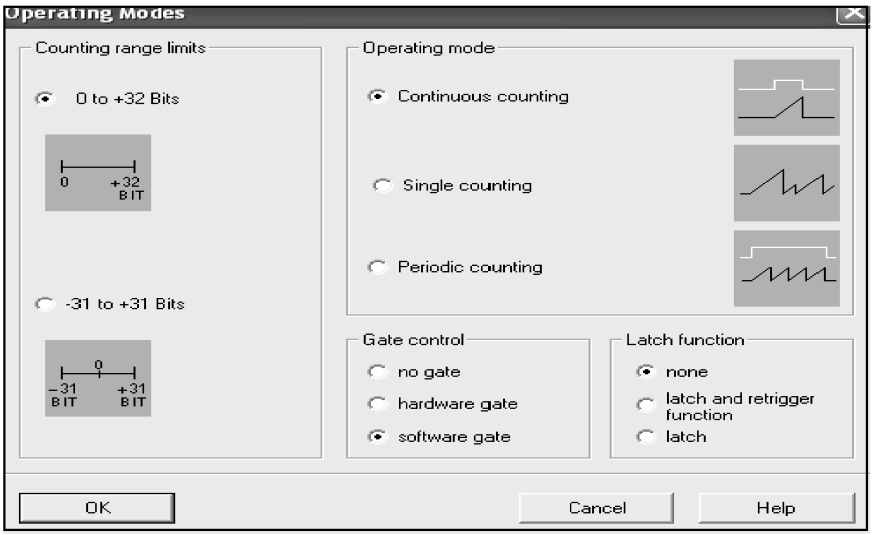



图 11-13　FM350-1 操作模式选择界面

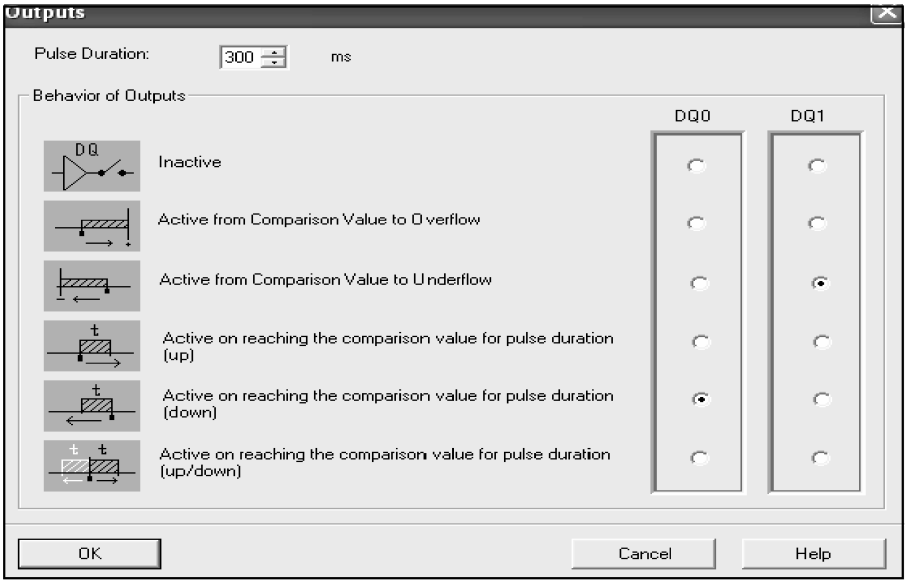



图 11-14　FM350-1 集成输出的配置界面

3．FM350-1 编程

在 CPU 中，编写 FM350-1 控制程序，将控制命令传送到 FM350-1 中，并读出计数值及反馈信息。安装 FM350-1 模块的配置软件后，在 SIMATIC Manager 中的“Libraries”中，将自动创建 FM350-1 的“FMX50LIB”函数库，打开函数库，将所有函数及 UDT 复制到当前的项目中。UDT2 包含 CPU 与高速计数器的通信结构，根据 UDT2 创建数据块，例如 DB1，将通信结构赋值到数据块 DB1 中，CPU 发送命令后，定义在 DB1 中的值将传送到高速计数器中，同时模块的反馈信息存储于 DB1 相应的地址中。在 OB1 调用 FC2 的控制程序如下：

```
L       256
```

```
T       DB1. DBW     6       //FM 350-1/450-1 地址
L       P#256. 0
T       DB1. DBD     8       //FM 通道地址，固定设置
L       16                   //FM 350-1 为 16；FM 450-1 为 32
T       DB1. DBB     12      //用户数据接口长度
CALL    " CNT_CTL1"
DB_NO     : =1
SW_GATE   : =M1. 1
GATE_STP  : =M1. 2
OT_ERR_A  : =M1. 3
SET_DO0   : =M1. 4
SET_DO1   : =M1. 5
OT_ERR    : =M1. 6
L_DIRECT  : =M1. 7
L_PREPAR  : =M2. 1
T_CMP_V1  : =M2. 2
T_CMP_V2  : =M2. 3
C_DOPARA  : =M2. 4
RES_SYNC  : =M2. 5
RES_ZERO  : =M2. 6
```

通过 UDT2 生成的 DB1 是 CPU 与高速计数器模块的通信数据区，在数据块中，包含 FM350-1 的地址信息，例如逻辑地址 256 存放在 DB1. DBW6 中，FM350-1 的通道地址与逻辑地址相同，但是表达形式为指针形式，例如 P#256. 0，FM350-1 接口区长度为 16，这些参数的设定为固定模式，除此之外，DB1 中还包括当前计数值、比较值及比较器状态等参数，对 FM350-1 模块的操作是通过调用 FC2 实现的。

**注意：**

如果在模块属性配置中直接指定 DB，则只需要调用 FC2 控制模块，不需要编程指定模块地址。

对函数 FC2 的参数解释如下：

DB_NO ：指定 CPU 与 FM350-1 模块数据交换的数据块，本例为 DB1。

SW_GATE ：软件门，例如 M1. 1 为 1 时，软件门打开，开始计数，为 0 时，软件门关闭。

GATE_STP ：为 1 时，终止计数功能。

OT_ERR_A ：复位 OT_ERR 操作故障。

SET_DO0 ：如果 CRTL_DO0（DB1. DBX28. 0 使能比较器的快速输出功能）为 1 时，通过该参数可以直接控制 FM350-1 集成的第一个输出点。

SET_DO1 ：如果 CRTL_DO1（DB1. DBX28. 1 使能比较器的快速输出功能）为 1 时，通过该参数可以直接控制 FM350-1 集成的第二个输出点。

OT_ERR ：输出操作故障信息，为 1 表示有故障。

L_DIRECT　：为 1 时，直接将 DB1. DBD14 中的值作为计数值。如果发送命令成功，FM350-1 将此位复 0。

L_PREPAR　：为 1 时，将 DB1. DBD14 中的值作为预备值，出现特殊事件（例如周期计数模式中没有设定计数方向而到达上下限、同步操作、锁存功能触发等）将预备值作为实际的计数值。如果发送命令成功，FM350-1 将此位复 0。

T_CMP_V1　：为 1 时，将 DB1. DBD18 中的值作为比较值 1 存放到模块中，根据模块“Outputs”界面中的配置决定 DO0 的输出。如果发送命令成功，FM350-1 将此位复 0。

T_CMP_V2　：为 1 时，将 DB1. DBD22 中的值作为比较值 2 存放到模块中，根据模块“Outputs”界面中的配置决定 DO1 的输出。如果发送命令成功，FM350-1 将此位复 0。

C_DOPARA　：为 1 时，将存储于 DB1. DBD14 中的数值作为命令源（有关命令格式参考模块手册），动态设定两个快速输出 DO0、DO1 的参数。如果发送命令成功，FM350-1 将此位复 0。

RES_SYNC　：通过 DI2 及 ENSET_UP、ENSET_DN、零脉冲信号进行同步操作，如果同步完成，同步状态位置 1，通过 RES_SYNC 复位同步状态位，从而可以得到新的同步信息。如果发送命令成功，FM350-1 将此位复 0。

RES_ZREO　：复位过零、上下限等状态信息，如果发送命令成功，FM350-1 将此位复 0。

本例中当 M1. 1 为 1 时，软件门打开（其他参数可以保持默认状态），计数开始，计数值可以从 DB1. DBD34 中读出。

**注意：**

在早期的版本中，通过 UDT1 生产数据块，通过 FC0 传递 CPU 与高速计数器模块的数据；在新的版本中，增加了一些新的功能，例如频率、速度的测量功能，通过 FC2，既可以实现计数功能，又可以满足测量的要求。

### 11. 1. 10　FM350-2 高速计数器的使用

FM350-2 高速计数器模块可以连接 24V 增量编码器，8 个通道，可以连接漏型输入信号（电流流入计数器模块）。随硬件包装的光盘中，包含模块配置软件包和电子手册。在 STEP7 中配置 FM350-2 时，必须预先安装配置软件。以连接 24V 编码器为例，介绍 FM350-2 模块的使用。

1. FM350-2 的接线

FM350-2 采用 40 针的前连接器，端子接线如图 11-15 所示。模块需要连接 24V 电源，A0 ~ A7 为 8 个计数器的计数信号端，B0 ~ B7 为 8 个计数器的方向信号端，如果只连接高速脉冲信号，可以连接到 A0 ~ A7 端子上；如果连接脉冲和方向信号，A0 ~ A7 端子连接计数脉冲信号，B0 ~ B7 端子连接计数的方向信号，信号为 0 时，向上计数，为 1 时，向下计数；如果连接 24V 增量编码器，A0 ~ A7、B0 ~ B7 连接编码器的 A、B 相，计数方向与 A、B 的相位有关。模块不能向 24V 编码器提供电源。I0 ~ I7 为 8 个计数通道的硬件门信号，并且只能为电平信号，为 1 时，门打开，为 0 时，门关闭，每个计数通道的门信号分为软件门信号和硬件门信号（使用硬件门时，必须先使能软件门）。Q0 ~ Q7 为 8 个计数通道比较器的快

速输出。除连接 24V 信号外，FM350-2 还可以连接 NAMUR 类型传感器，并提供传感器电源。本例选择连接两个 24V 增量编码器，分别连接到第一（端子 A0、B0）和第五通道（端子 A4、B4）上。

2. FM350-2 的硬件配置

在 SIMATIC Manager 硬件配置中，插入 FM350-2 模块，双击 FM350-2 模块，进入配置界面，在弹出的“Specify Channels”对话框中，可以选择计数通道的用途，例如选择连接“NAMUR”类型传感器或由 4 个通道组成的比例型计数器等，也可以选择利用 FM350-2 的逻辑地址直接读出 4 个计数通道的计数值，但是模块的其他功能受到限制，本例中选择默认设置。点击“Encoders”图标，选择连接的编码器类型，如图 11-16 所示。在例图中，选择旋转编码器单倍频信号采集；如果选择计数模式为单次计数或者周期计数，计数方向栏被激活，可以选择计数的主方向；在“Hysteresis”参数中填写输出滞后值，这个值影响比较器触发的快速输出，例如设定的比较值为 5，没有设置滞后值（等于 0），计数方向向上，计数值大于等于比较值时输出，如果计数值在 4 ~ 6 间振荡时，快速输出被不断触发，可能不符合工艺要求，如果设置输出滞后值，相当于加入一个“死区”，计数值在一定范围中振荡时，不影响输出的变化。

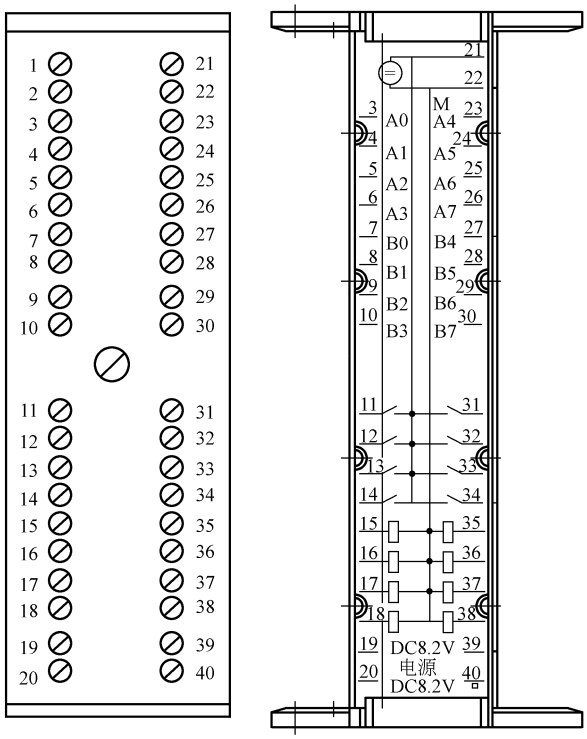



图 11-15　FM350-2 端子接线图

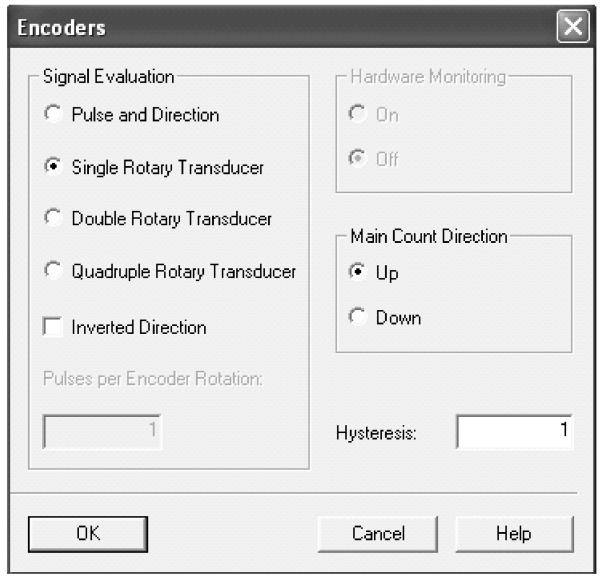



图 11-16　FM350-2 编码器配置图

操作模式的选择如图 11-17 所示。

在操作模式界面中，选择计数模式和测量模式，频率测量模式需要定义采样的时间窗口；在操作模式界面中，还可以选择是否使用硬件门作为计数的触发条件。本例中，选择连续计数模式，使用软件门触发计数。设置第一个计数通道后，以相同的方法设置第五个计数通道，配置完成后编译存盘。

3. FM350-2 编程

安装 FM350-2 模块的配置软件后，在 SIMATIC Manager 中的“Libraries”中，将自动建立 FM350-2 的函数库“fm_cntli”，打开函数库，将所有函数和 UDT 复制到当前的项目中，UDT1 包含 CPU 与 FM350-2 模块的通信结构，根据 UDT1 创建数据块，例如 DB1，将通信结构赋值到数据块 DB1 中，CPU 发送命令后，定义在 DB1 中命令值将传送到 FM350-2 模块中，同时 FM350-2 将状态值和计数值发送到 DB1 相应的地址中。使用多个 FM350-2 模块时，每个模块需要一个独立的数据块，数据块中包含 FM350-2 的地址参数。在硬件配置界面中，用右键点击 FM350-2 模块，在弹出的下拉菜单中选择进入属性界面，在模块属性中，指定已经生成的数据块，FM350-2 的地址参数自动赋值到数据块中，否则地址参数需要在程序中赋值（DB1. DBW12 和 DB1. DBD14）。在 OB1 中编写的程序如下：

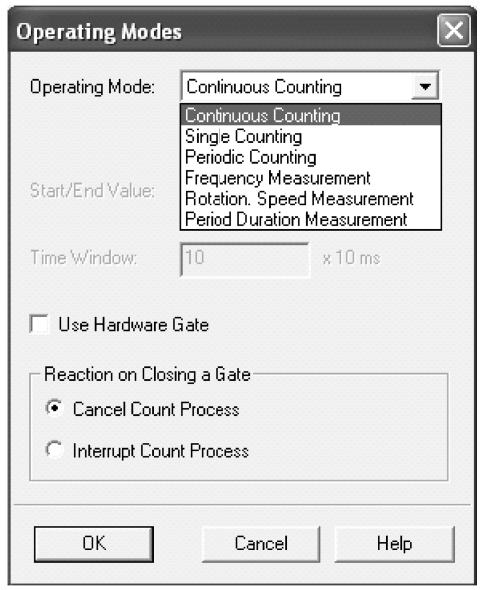



图 11-17　FM350-2 操作模式选择

```
        L       W#16#110            //FM350-2 逻辑地址
        T       DB1. DBW    12
        L       P#110. 0            //FM350-2 的通道地址、逻辑地址的指针形式
        T       DB1. DBD    14      //如果使用模块属性将地址参数赋值到数据块中，
                                      不需编写以上程序
        CALL    " CNT_CTL1"         //调用 FM350-2 控制块 FC2
        DB_NO：= W#16#1             //指定 DB1 为 CPU 与 FM350-2 模块数据交换的数
                                      据块
        SET
        =       DB1. DBX    23. 0   //打开第一个计数器的软件门
        =       DB1. DBX    23. 4   //打开第五个计数器的软件门
        A       M           1. 3    //M1. 3 为 1 时，读前 4 通道计数值
        JCN     m1
        L       100
        T       DB1. DBB    2
m1：    AN      M           1. 3    //M1. 3 为 0 时，读后 4 通道计数值
        JCN     m2
        L       101
        T       DB1. DBB    2
m2：    CALL    " CNT2_RD"          //调用读通信块 FC4
        DB_NO ：= W#16#1            //指定 DB1 为 CPU 与 FM350-2 模块数据交换的数
                                      据块
```

```
RET_VAL  :  = MW2
```

将 FM350-2 的地址参数 W#16#110（十进制为 272）存放到 DB1. DBW12 中，将逻辑地址的指针形式存放到 DB1. DBD14 中；调用 FC2 的作用是为了设定计数通道的软件门和读出状态反馈，8 个计数通道的设定值存放在 DB1. DBB23 中，FM350-2 软件门状态存储于 DB1. DBB43 中，用于软件门的判断。本例中，分别打开第一和第五计数通道的软件门。通过读工作号和调用读操作函数 FC4，读出计数值，读工作号为 100，表示读前 4 个计数通道的计数值；读工作号为 101，表示读后 4 个计数通道的计数值，但不能直接读出 8 个通信计数值，读工作号必须存储于 DB1. DBB2 中。示例程序中，M1. 3 为在 CPU 中设定的时钟信号，脉冲频率为0. 5s，M1. 3 为 1 时，读出前 4 个计数通道的计数值；M1. 3 为 0 时，读出后 4 个计数通道的计数值。这样 8 个计数通道的前后 4 个通道的计数值在 CPU 中的更新间隔为0. 5s（编写的间隔时间必须大于 CPU 与 FM350-2 的通信时间），第一个计数通道的计数值存储于 DB1. DBD148 中，第五个计数通道的计数值存储于 DB1. DBD180 中。

读数据通过读工作号完成，同样 CPU 向 FM350-2 模块写的数据是通过写工作号和调用写操作函数 FC3 完成的，例如通过写工作号 10 设定第一个通道的装载值，示例程序如下：

```
A       M          6. 1
JCN     M3
L       123
T       DB1. DBD   52          //第一计数通道的装载值
L       10                     //写工作号 10
T       DB1. DBB   0
CALL    " CNT_CTL2"            //调用写操作函数 FC3
DB_NO     :=W#16#1
RET_VAL   :=MW4
M3：NOP   0
```

写工作号必须存储于 DB1. DBB0 中，当 M6. 1 为 1 时，调用写操作函数 FC3，执行存放在 DB1. DBB0 中的写任务——写工作号 10（写第一个计数通道的装载值），将存放在 DB1. DBD52 中的值 123 作为第一个计数通道的装载值存储于 FM350-2 中。写任务完成后，应立刻复位写任务操作（如复位 M6. 1），释放通信资源。其他的读写工作号可以参考 FM350-2 手册。

4. FM350-2 的调试功能

FM350-2 模块带有在线调试功能，在编写控制程序之前，可以通过在线调试功能测试设置的参数及连接编码器的类型是否正确，并读出计数值，如果在调试界面不能读出计数值，通过程序同样不能读出计数值。在 FM350-2 的硬件参数化界面中，点击菜单命令“Debug”“Commissioning”，进入调试界面，如图 11-18 所示。

通过调试界面可以对快速输出、软件门、装载值及比较值进行操作，也可以读出计数值及测量值等状态信息。在调试界面所有的测试任务及步骤都可以使用程序完成，可以根据测试的步骤编写用户程序自动执行计数工作。

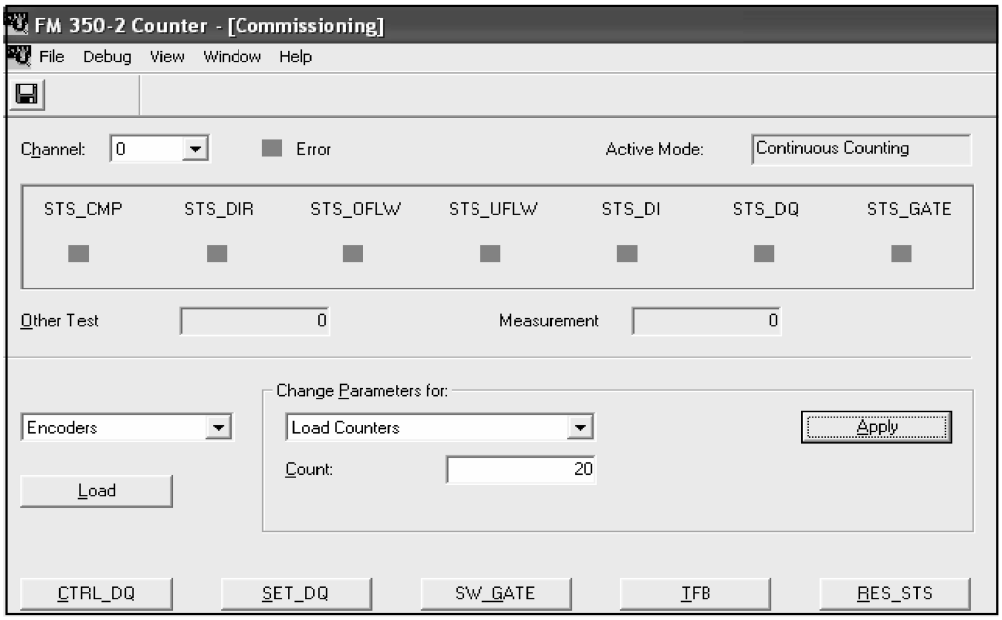



图 11-18　FM350-2 调试界面

## 11.1.11　S7-300C 系列 PLC 集成高速计数器的使用

对于 S7-300C 系列和 C7 系列 PLC 集成高速计数功能，以 CPU314C 为例介绍集成高速计数功能。CPU314C 集成四路完全独立的计数器，最高计数频率为 60kHz，可以连接计数脉冲信号、脉冲和方向信号以及 24V 增量编码器信号，可以连接漏型输入信号（电流流入计数器模块），配置软件集成在 STEP7 中。

1. CPU314C 集成高速计数器的接线

以第一个计数通道为例，介绍计数器的接线，接线端子功能见表 11-2。

**表 11-2　CPU314C 第一个计数通道接线**

| 端子号 | 名称/地址 | 功　　能 |
|---|---|---|
| 2 | DI +0.0 | 通道 0：A 相/脉冲信号 |
| 3 | DI +0.1 | 通道 0：B 相/方向信号 |
| 4 | DI +0.2 | 通道 0：硬件门 |
| 16 | DI +1.4 | 通道 0：锁存功能输入 |
| 22 | DO +0.0 | 通道 0：高速响应输出 |

如果信号源为单脉冲信号，连接到 2 号端子；如果信号源为脉冲和方向信号，将脉冲信号连接到 2 号端子，将方向信号连接到 3 号端子；如果信号源为 24V 增量编码器信号，连接 A、B 相到 2、3 号端子。端子 4 为硬件门信号，端子 16 为计数器的锁存功能输入，22 号端子为计数器 0 通道的高速响应输出。

2. CPU314C 高速计数器的硬件配置

在硬件配置中，插入 CPU314C-2DP CPU，双击“Count”槽位，进入高速计数器配置界面，在“General”标签页中，选择需要配置的计数通道和计数模式；在“Count”标签页，

为每个选择的计数通道进行参数配置，如图 11-19 所示。

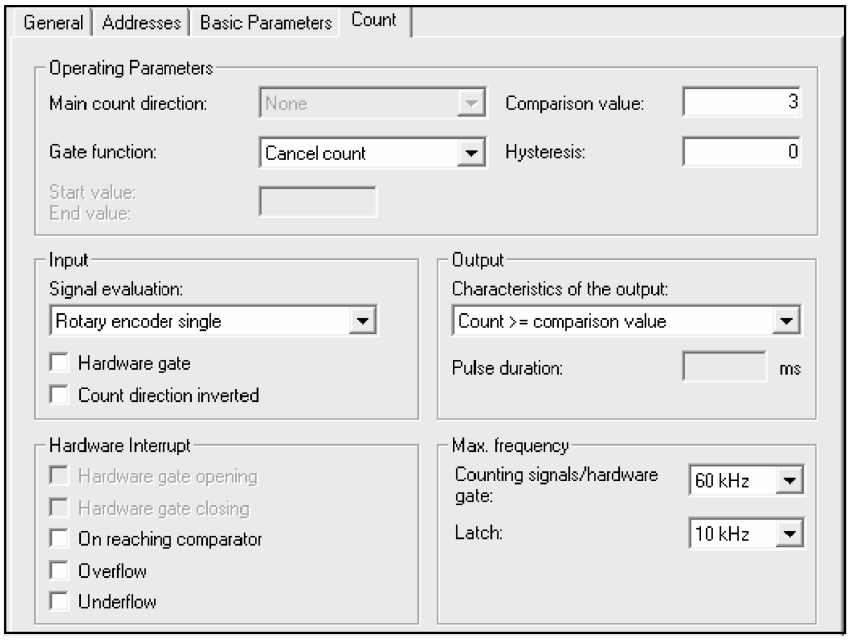



图 11-19　S7-300C 计数器通道的配置

如果选择计数模式为单次计数和周期计数，在“Operating Parameters”栏中可以设置主计数方向、计数值的开始值及结束值（与主计数方向的设置有关）；如果选择连续计数模式，则不需要设定计数方向；设定门功能时，如果选择“Cancel count”，则门关闭后再次打开，从装载值或上次计数值开始计数；如果选择“Stop count”，则门关闭后再次打开，从初始值开始计数；比较值的设定影响高速响应输出，在“Hysteresis”参数中，可设定滞后值，防止计数值在比较值附近振荡而造成高速响应输出频繁动作。

在输入栏中，选择输入信号的类型及是否使用硬件门。例图中，选择旋转编码器单倍频采集，无硬件门。在输出栏中，选择触发高速响应输出的事件。如需要，可在硬件中断栏中选择触发硬件中断的事件，硬件中断事件触发 CPU 调用 OB40。高速计数器配置完成后编译存盘。

3．CPU314C 集成高速计数器的程序控制

在 CPU 中编写控制程序，将控制参数传送到集成的高速计数器中，并读出计数值等反馈信息。在 OB1 中调用专用系统函数块 SFB47，背景数据块选择 DB47。其程序如下：

```
CALL    " COUNT",    DB47
          LADDR      :=W#16#300   //逻辑地址，默认为 768
          CHANNEL    :=0          //通道号
          SW_GATE    :=M1.1       //软件门
          CTRL_DO    :=M1.2       //控制输出
          SET_DO     :=M1.3       //置位输出
          JOB_REQ    :=M1.4
```

```
JOB_ID      := MW2
JOB_VAL     := MD4
STS_GATE    :=
STS_STRT    :=
STS_LTCH    :=
STS_DO      :=
STS_C_DN    :=
STS_C_UP    :=
COUNTVAL    := MD12        //实际计数值
LATCHVAL    :=
JOB_DONE    :=
JOB_ERR     :=
JOB_STAT    :=
```

系统函数块 SFB47 参数的含义如下：

LADDR　　：计数器的逻辑地址，默认为 W#16#300。

CHANNEL　：计数器通道，0 为第一个计数通道。

SW_GATE　：计数通道软件门。

CTRL_DO　：为 1 时，使能高速响应输出。

SET_DO　　：CTRL_DO、SET_DO 为 1，可以通过程序控制高速响应输出。

JOB_REQ、JOB_ID、JOB_VAL：读写工作参数。

STS_X　　：状态反馈值。

COUNTVAL　：实际计数值。

LATCHVAL　：锁存计数值。

JOB_DONE、JOB_ERR、JOB_STAT：读写工作状态值。

上例中，M1.1 为 1 时，第一个计数通道的计数值存放在 MD12 中。如果进行写工作操作，例如写入比较值，从在线帮助中找出写比较值的任务 ID 号，即为 W#16#0004 并传送到 MW2 中，将需要设定的比较值传送到 MD4 中，M1.4 上升沿触发写操作，这样存储于 MD4 中的比较值传送到高速计数器中。同样可以从高速计数器中再次读出写入的比较值，将读比较值的任务 ID 号，即为 W#16#0084 传送到 MW2 中，M1.4 上升沿触发读操作，将读出的比较值存放在 DB47. DBD28 中。

### 11.1.12　ET200S 高速计数器的使用

ET200S 分布式 I/O 计数器模块分为连接 24V 增量编码器和连接 5V 对称脉冲串增量编码器两种。两种模块计数方式相同，以连接 24V 增量编码器为例，介绍 ET200S 高速计数器模块的使用。ET200S 每个计数器模块为单通道计数模块，最高计数频率为 100kHz，可以连接脉冲信号、脉冲和方向信号以及 24V 增量编码器信号，可以连接漏型输入（电流流入计数器模块）和源型输入（电流流出计数器模块信号），配置软件集成在 STEP7 中。

1．ET200S 高速计数器模块的接线

接线端子如图 11-20 所示。

如果信号源为单脉冲信号，可以连接到 5 号端子；如果信号源为脉冲和方向信号，将脉冲信号连接到 5 号端子，将方向信号连接到 1 号端子；如果信号源为 24V 增量编码器信号，则连接 A、B 相到 5、1 号端子。8 号端子为计数器的硬件门或锁存功能输入（在配置中选择），4 号端子为计数器的高速输出。端子 2、3，6、7 为模块电和给编码器提供的电源。

2. ET200S 高速计数器模块的硬件配置

在 SIMATIC Manager 硬件配置中，插入 ET200S 高速计数器模块，双击模块进入配置界面，如图 11-21 所示，在“Signal evaluation A＊，B＊”选项中，选择连接的信号源，本例中选择旋转编码器单倍频采集；在“Filter xx”选

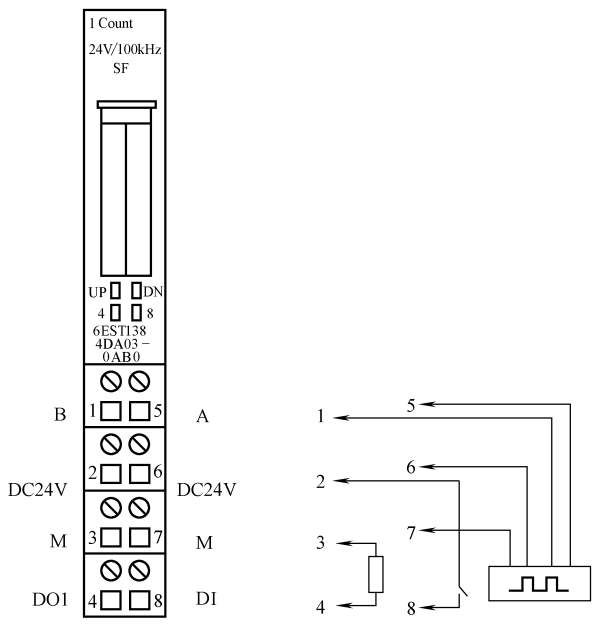



图 11-20 ET200S 高速计数器模块接线图

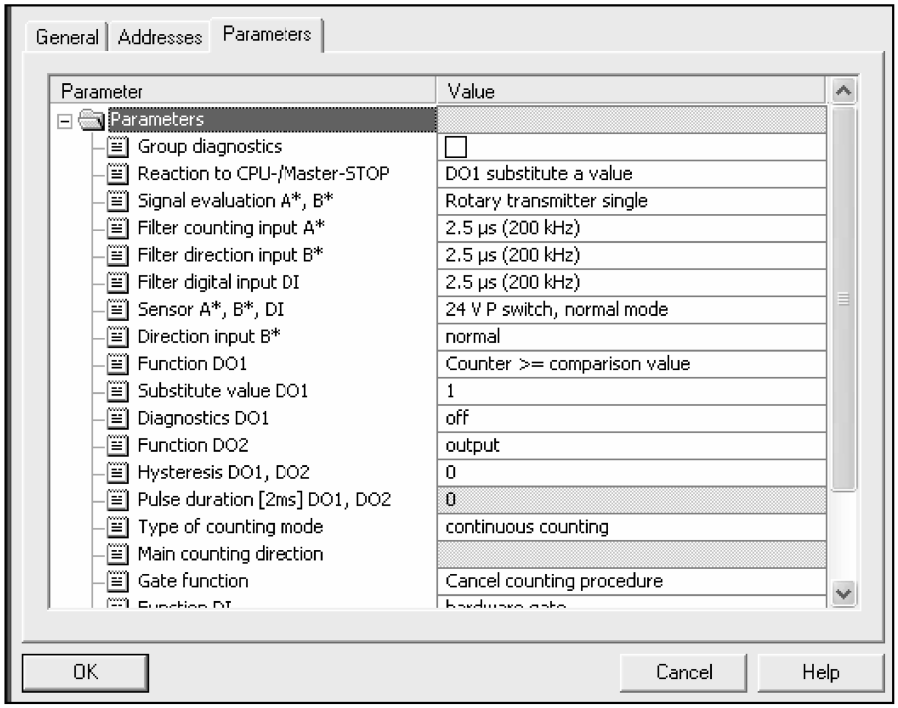



图 11-21 ET200S 计数器模块配置界面

项中，根据输入的频率选择信号的滤波时间，滤波时间长，信号稳定，但输入频率低；反之频率高，信号容易被干扰；在“Sensor A＊，B＊，DI”选项中，选择输入信号的类型－漏

型或源型；在“Function DO1、DO2”选项中，选择触发 DO1、DO2 的事件，例如计数值大于等于比较值时，触发 DO1 输出，比较值通过程序设置，高速计数器模块没有集成 DO2，DO2 功能，只提供信号，可以触发其他输出点，用于响应较慢的应用；在“Hysteresis DO1，DO2”参数中，可设定滞后值，防止计数值在比较值附近振荡，而造成高速响应输出频繁动作；在“Type of counting mode”选项中，可以选择连续计数、单次计数和周期计数等三种模式；在“Gate Function”门功能设定时，如果选择“Interrupt counting procedure”，则门关闭后，再次打开，从上次计数值开始计数；如果选择“Cancel counting procedure”，则门关闭后，再次打开，从初始值开始计数；在“Function DI”选项中，可以选择模块集成的输入为硬件门信号还是为锁存信号。配置完成并编译存盘。

3. 编写 ET200S 高速计数器控制程序

ET200S 高速计数器在硬件配置中占用 CPU 八个字节的输入和八个字节的输出，CPU 通过输入、输出字节读写计数模块信息。读写的示例程序如下：

```
L     PID    272      //模块开始地址，将计数器模块状态值存放到 MB20
                        ~MB27 中
T     MD     20       //当前计数值
L     PID    276
T     MD     24

L     123             //装载比较值
T     MD     30

SET
=     M      34.0     //打开软件门

A     M      10.0     //使能传送比较值功能
=     M      35.2

L     MD     30       //将操作命令传送到计数模块中
T     PQD    272
L     MD     34
T     PQD    276
```

示例程序中 ET200S 高速计数器的输入、输出地址同为 272，输入的前 4 个字节为当前的计数值存放于 MD20 中。在输出格式中，第 5 个字节的第一位为软件门信号，第 6 个字节的第三位触发传送比较值命令，将输出的前 4 个字节中的值作为比较值，传送到计数器中。具体的输入、输出接口请参考 ET200S 计数器手册。

高速计数器的示例程序 High_counter.zip、参数化软件及电子手册参考光盘“功能模块 FM”目录。

## 11.2　FM352-5 高速布尔处理器

FM352-5 模块高速处理数字量输入、输出信号，从读入数字量输入信号、CPU 程序处理

到信号的输出最小周期不到 10μs，适合逻辑简单、要求快速响应的应用，如速度测量、脉宽调制（最小 20μs 可调）输出等。由于 FM352-5 是高速布尔处理器，所以只能对位、字节、字、双字等信号进行处理，而不能处理浮点信号。

### 11.2.1 工作方式

FM352-5 具有独立的处理能力，安装方式灵活，可以安装于 S7-300 系列 PLC 中央机架、分布式 I/O ET200M 中（可以是非西门子主站），也可以独立安装。独立安装时，只能通过集成的输入、输出信号对控制设备进行操作并读出反馈状态，而不能通过人机界面以通信的方式操作。与 S7-300 系列 PLC CPU 结合安装于中央机架，CPU 与 FM352-5 数据交换过程如图 11-22 所示。

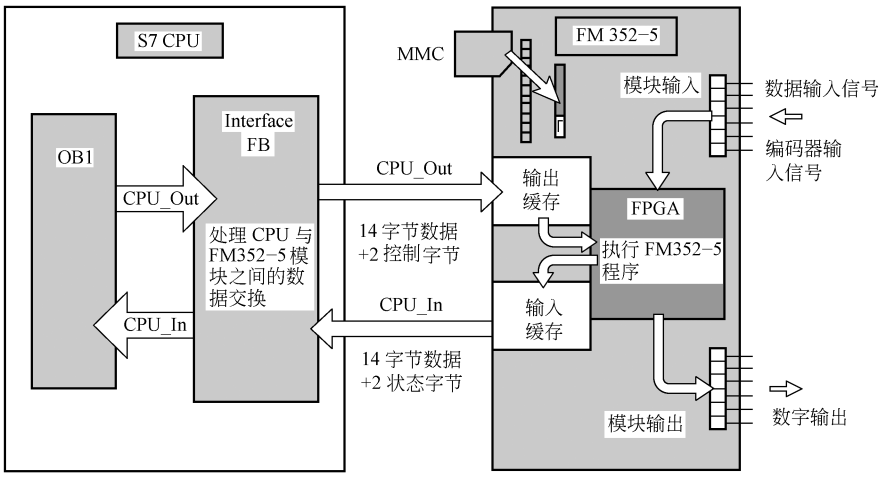



图 11-22 CPU 与 FM352-5 数据交换示意图

FM352-5 模块通过输入端接收编码器信号、数字量输入信号及 CPU 发送的命令信号，经过 FPGA（现场可编程门阵列）处理后，直接通过模块集成的输出点输出，一些数据可以作为状态信号反馈到 CPU。CPU 与 FM352-5 间的通信数据经过接口函数块的处理后，进入 CPU 的用户数据区，FM352-5 与 CPU 之间的数据交换也可以通过FM352-5 的逻辑地址直接通信。

普通 CPU 程序执行是以串行扫描的，FM352-5 程序执行则是以并行扫描的，保证程序执行的快速性，由于并行执行程序，有些信号在不同程序段中的状态不同，将导致逻辑结果错误，FM352-5 使用连接器和多相时钟的技术使相竞争的条件延迟，保证信号的正确性。连接器的使用规则如下：

- 如果连接器的触点在它的输出线圈以前引用，那么触点的值将取决于上个扫描结果。
- 如果连接器的触点在它的输出线圈以后引用，那么触点的值将取决于当前扫描结果。

连接器的使用如图 11-23 所示。

连接器使用的方式相当于中间变量，使用梯形图编写的 FM352-5 程序存储于一个 FB 中，所有的变量使用在 FB 中定义的形参，中间变量使用固定的形参格式#Conn. arrXcon。在图 11-23 程序段 1 中，连接器#Conn. arrXcon［8］和#Conn. arrXcon［9］的触点信号在输出线圈以前引用，所以它们的值将取自于线圈的上个扫描结果。在程序段 4 中，连接器

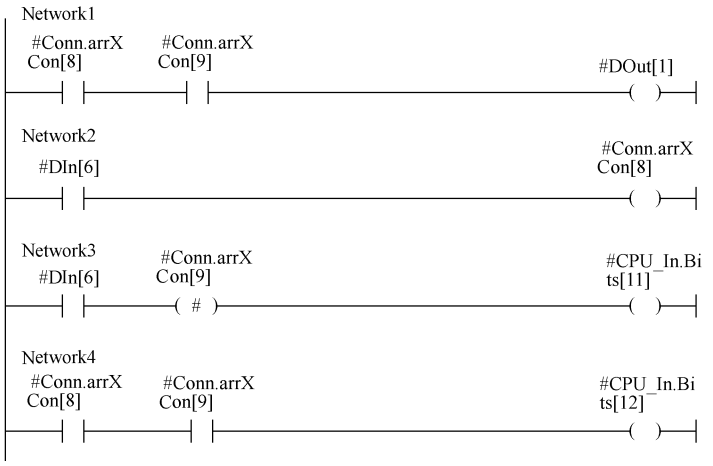



图 11-23　FM352-5 连接的使用

#Conn. arrXcon［8］和#Conn. arrXcon［9］的触点信号在它的输出线圈以后引用，所以它们的值将取自于线圈的当前扫描结果。如果在不同程序段，对于连接器使用具有保持功能的指令块，如计数器、定时器、RS 触发器、沿检测、位移寄存器等时（参考表 11-10），模块通过多相时钟管理这些指令块正确的时序，示例程序如图 11-24 所示。

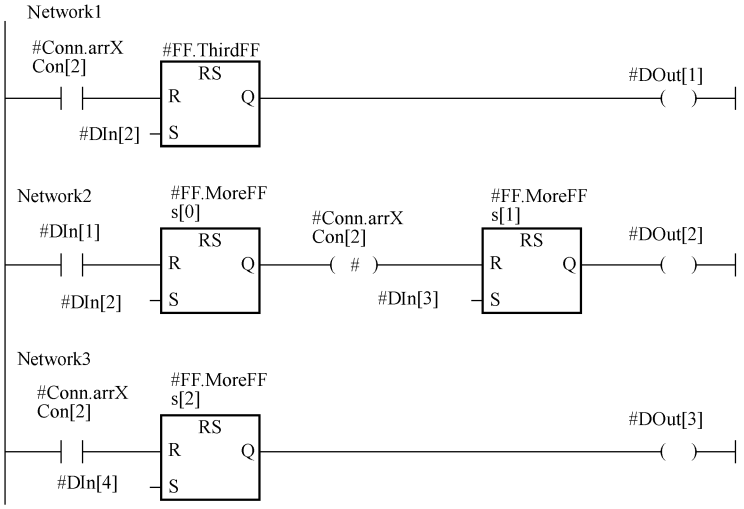



图 11-24　保持功能块的多相定时

在程序段 1 中，连接器#Conn. arrXcon［2］的触点信号在它的输出线圈前引用，它的值将取自于线圈的上个扫描结果，所以 RS 触发器#FF. ThirdFF 被看作相序 1；在程序段 2 中，RS 触发器#FF. MoreFFs［0］的相序为 1，#FF. MoreFFs［1］的相序为 2。连接器线圈#Conn. arrXcon［2］在相序 1 后输出；在程序段 3 中，通过连接器触点#Conn. arrXcon［2］，将 RS 触发器#FF. MoreFFs［2］连接到程序段 2 的相序 1 后，同样被看作为相序 2，程序段 3 相当于程序段 2 的扩展部分，这样程序执行相序图（顺序）如图 11-25 所示。

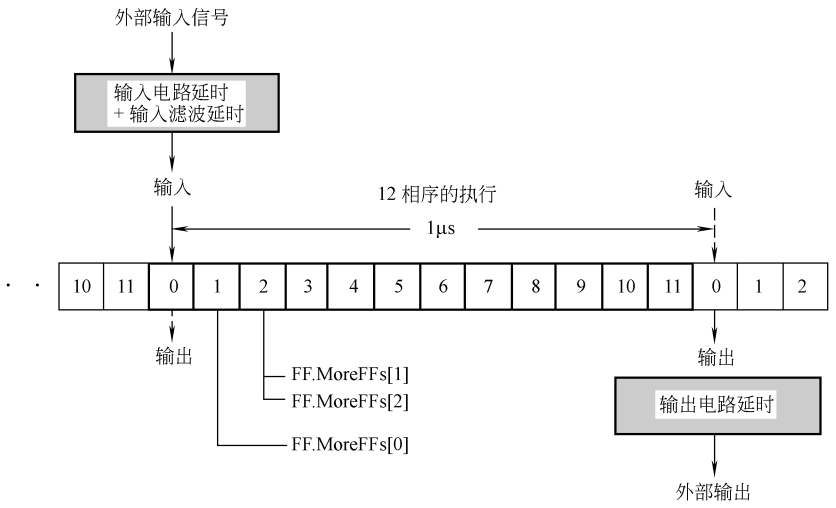



图 11-25　12 相序定时和 I/O 时序图

从图 11-25 中可以看到，#FF. MoreFFs［0］为相序 1 最先执行，#FF. MoreFFs［2］和#FF. MoreFFs［1］同为相序 2 被同时执行。每一个程序段中最多可以串连 11 个（相序）具有保持功能的命令块，第 12 个相序作为输出，如果利用连接器扩展，如图 11-24 中程序段 2 和程序段 3 中最多只能连接 11 个指令块，否则编译时报错。利用 12 相序的处理方式也保证 FM352-5 并行处理的稳定性——程序的执行周期固定 1μs。

## 11. 2. 2　输入输出端子接线

FM352-5 集成最多 15 个数字输入，8 个数字输出和一路编码器输入信号。输入、输出端子的定义见表 11-3 所示。

**表 11-3　FM352-5 输入、输出端子定义**

| 端子号 | 输入/输出 | 名称 | 功　能 | 指示灯 |
|---|---|---|---|---|
| 1 | | 2M | 2 区电源公共端 – 输入输出电路 | — |
| 2 | 输入 | I0 | 输入 | 绿灯 |
| 3 | 输入 | I1 | 输入 | 绿灯 |
| 4 | 输入 | I2 | 输入 | 绿灯 |
| 5 | 输入 | I3 | 输入 | 绿灯 |
| 6 | 输入 | I4 | 输入 | 绿灯 |
| 7 | 输入 | I5 | 输入 | 绿灯 |
| 8 | 输入 | I6 | 输入 | 绿灯 |
| 9 | 输入 | I7 | 输入 | 绿灯 |
| 10 | | 见注 1 | 2 区电源端 – 输入输出电路 | — |
| 11 | 输出 | Q0 | 源输出/源输入型输出（见注 2） | 绿灯 |
| 12 | 输出 | Q1 | 源输出/源输入型输出（见注 2） | 绿灯 |
| 13 | 输出 | Q2 | 源输出/源输入型输出（见注 2） | 绿灯 |
| 14 | 输出 | Q3 | 源输出/源输入型输出（见注 2） | 绿灯 |
| 15 | 输出 | Q4 | 源输出/源输入型输出（见注 2） | 绿灯 |

（续）

| 端子号 | 输入/输出 | 名称 | 功　　能 | 指示灯 |
|---|---|---|---|---|
| 16 | 输出 | Q5 | 源输出/源输入型输出（见注 2） | 绿灯 |
| 17 | 输出 | Q6 | 源输出/源输入型输出（见注 2） | 绿灯 |
| 18 | 输出 | Q7 | 源输出/源输入型输出（见注 2） | 绿灯 |
| 19 | | 2L + | 2 区电源端输出 – 输入输出电路 | — |
| 20 | | 2M | 2 区电源公共端 – 输入输出电路 | — |

注 1：
订货号为 FM352-5AH00-0AE0 模块，引脚 10 为 2M，作为 2 区输入输出电路公共端。
订货号为 FM352-5AH10-0AE0 模块，引脚 10 为 2L + ，作为 2 区输入输出电路电源端。
注 2：
订货号为 FM352-5AH00-0AE0 模块带有漏型（NPN）输出。
订货号为 FM352-5AH10-0AE0 模块带有源型（PNP）输出。

根据现场信号和工艺的要求，选择漏型或者源型输出的 FM352-5 模块，漏型比源型输出的响应频率高。

编码器的端子定义见表 11-4。

**表 11-4　编码器的端子定义**

| 端子号 | 输入/输出 | 名称 | 功　　能 | | | | 指示灯 |
|---|---|---|---|---|---|---|---|
| | | | 5V 编码器 | SSI Master | SSI Listen | 24V 编码器 | |
| 21 | | 3L + | 3 区电源端 – 编码器电路 | | | | — |
| 22 | | 3M | 3 区电源公共端 – 编码器电路 | | | | |
| 23 | | 3M | 3 区电源公共端 – 编码器电路 | | | | |
| 24 | 输出 | 5V 输出 | 5. 2V 编码器电源 | | | | 红灯 |
| 25 | 输出 | 24V 输出 | 24V 编码器电源 | | | | 红灯 |
| 26 | 输入 | 编码器 | A | DATE | DATE | I12 + | |
| 27 | 输入 | 编码器 | /A | /DATE | /DATE | I12- | |
| 28 | 输入 | 编码器 | B | I13 + | CLOCK | I13 + | |
| 29 | 输入 | 编码器 | /B | I13- | /CLOCK | I13- | |
| 30 | 输入 | 编码器 | N | I14 + | I14 + | I14 + | |
| 31 | 输入 | 编码器 | /N | I14- | I14- | I14- | |
| 32 | 输入 | 编码器 | — | CLOCK | — | — | |
| 33 | 输入 | 编码器 | — | /CLOCK | — | — | |
| 34 | — | — | — | | | | — |
| 35 | — | — | — | | | | — |
| 36 | 输入 | I8 | I8 | I8 | I8 | I8 | 绿灯 |
| 37 | 输入 | I9 | I9 | I9 | I9 | A | 绿灯 |
| 38 | 输入 | I10 | I10 | I10 | I10 | B | 绿灯 |
| 39 | 输入 | I11 | I11 | I11 | I11 | N | 绿灯 |
| 40 | | 3M | 3 区电源公共端 – 编码器电路 | | | | — |

FM352-5 模块可以连接下列类型的编码器：

- 5V RS-422 对称脉冲串增量型编码器；
- 24V 非对称脉冲增量型编码器；
- 具有同步串行接口 SSI 的 Master 或 Listen 型绝对值编码器。

通过模块的硬件配置选择连接编码器的类型，同时只能连接一个编码器。使用编码器将占用对应的输入信号，例如连接 24V 增量型编码器时，I9、I10、I11 被占用。如果没有连接编码器，接收编码器信号的端子可以作为数字量输入信号，最多为 15 个数字输入，其中包括 3 个差分输入信号 I12、I13 和 I14。

### 11.2.3 模块的参数化

安装 FM352-5 模块软件包后，可以对模块进行参数化，在“SIMATIC Manager”硬件配置界面中插入 FM352-5 模块，双击 FM352-5 的图标，在弹出的对话框中，点击“Parameters”按钮，进入参数化界面，如图 11-26 所示。

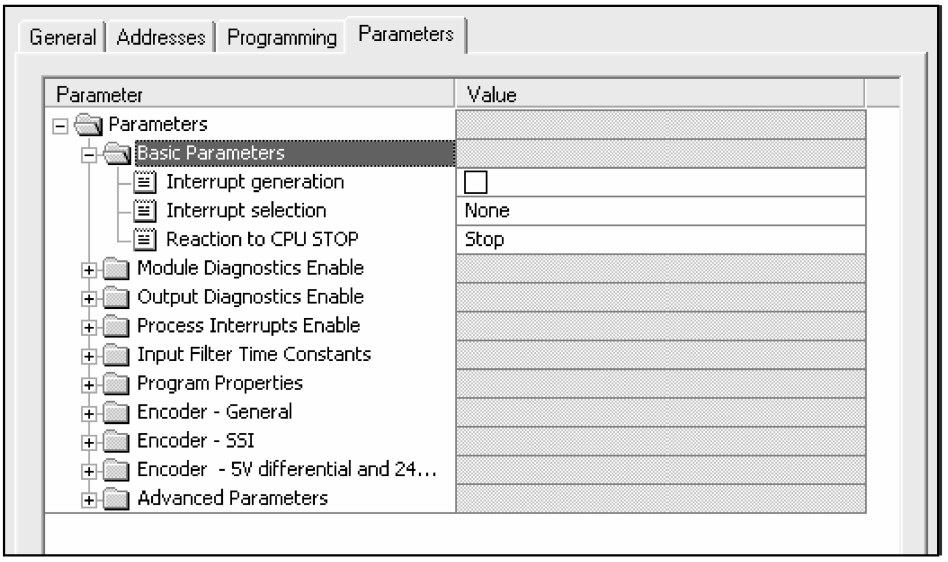



图 11-26 FM352-5 参数化界面

FM352-5 模块可配置的参数如下：

1）“Basic Parameters” 定义是否产生中断，产生的中断分为诊断中断（OB82）和过程中断（OB40）。

2）“Module Diagnostics Enable” 定义产生诊断中断的条件，如电源 1L + 、2L + 掉电等。

3）“Output Diagnostics Enable” 定义输出过载是否产生诊断中断。

4）“Process Interrupts Enable” 定义 FM352-5 生成的 8 个过程中断，这些中断需要通过在 FM352-5 中编程触发（“Intr”形参）。产生的中断在 CPU 中调用相应的 OB，例如 OB40，通过 OB40 形参“OB40_POINT_ADDR”，第一个字节的 8 个位可以判断 FM352-5 中产生中断的事件。“Module Diagnostics Enable”、“Output Diagnostics Enable ”、“Process Interrupts Enable ” 中的参数是动态参数，在 CPU 中可以通过调用 SFC55 写数据记录 1 进行配置和修改，数据记录 1 中包含 8 个字节，数据格式见表 11-5。

表 11-5　数据记录区 1 的诊断数据

| 字节 | 位 7 | 位 6 | 位 5 | 位 4 | 位 3 | 位 2 | 位 1 | 位 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | M1L | M2L | ESSF | M3L | — | — | — | — |
| 1 | SSIF | DBW | — | — | — | — | — | — |
| 2 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| 3 | MMC | — | — | — | — | — | — | — |
| 4 | PAE7 | PAE6 | PAE5 | PAE4 | PAE3 | PAE2 | PAE1 | PAE0 |
| 5 | — | — | — | — | — | — | — | — |
| 6 | — | — | — | — | — | — | — | — |
| 7 | — | — | — | — | — | — | — | — |

表 11-5 中参数含义如下：

M1L　：1L 电源掉电。

M2L　：2L 电源掉电。

ESSF　：编码器电源故障。

M3L　：3L 电源掉电。

SSIF　：SSI 编码器值溢出。

DBW　：差分输入编码器断线。

07 ~ 00：输出过载。

MMC　：MMC 卡故障。

PAE　：8 个过程中断。

表中每一个位表示一个生成中断的信息，值为 0 时不产生中断，值为 1 时激活中断功能。没有使用的位被保留，设置为 0。修改的参数在 CPU 再次启动后，恢复模块中的设置值。

- “Input filter time constants”：定义输入信号的滤波时间，选择滤波时间长，采样不易受到干扰，但响应慢；反之易受到干扰，响应快。
- “Program properties”：选择 FM352-5 是否可以独立运行。选择独立运行时，FM352-5 的程序通过 PG 或写卡器写入 MMC 中。
- “Encoder-General”：选择使用编码器的类型，SSI、5V 和 24V 编码器，或者不连接编码器。
- “Encoder-SSI”：如果连接 SSI 编码器，在本栏中定义 SSI 编码器的参数。
- “Encoder-5V Differential and 24V single-ended encoder”：如果连接 5V 或 24V 编码器，在本栏中定义编码器的参数。
- “Advanced Parameters”：参数 “Module Diagnostics hardware support”、“Output Diagnostics hardware support”、“Process Interrupts hardware support” 是 “Module Diagnostics Enable”、“Output Diagnostics Enable”、“Process Interrupts Enable” 中选择产生中断条件的先决条件，如果在 “Advanced Parameters” 中没有选择，将不会产生任何中断信息。FM352-5 可使用的编程资源有限，所有配置的中断都将占用 FM352-5 的资源，如果去掉没有使用的中断，将节省模块的资源，增加用户程序的空间。

“Consistency check” 检查 CPU 与 FM352-5 的配置匹配性，如果选择 “Module checks for consistency”，修改 FM352-5 参数后，编译后的配置信息必须重新下载到 CPU 中。

### 11.2.4 编程

FM352-5 模块具有独立的执行程序，与普通的 CPU 编程方法类似，但是不能使用语句表编程。由于 FM352-5 是针对位信号进行处理的，在 SIMATIC Manager 中集成的一些指令不能使用，如图 11-27 所示。由于时序的原因，在位逻辑中不能使用单线圈的 R/S 触发器及

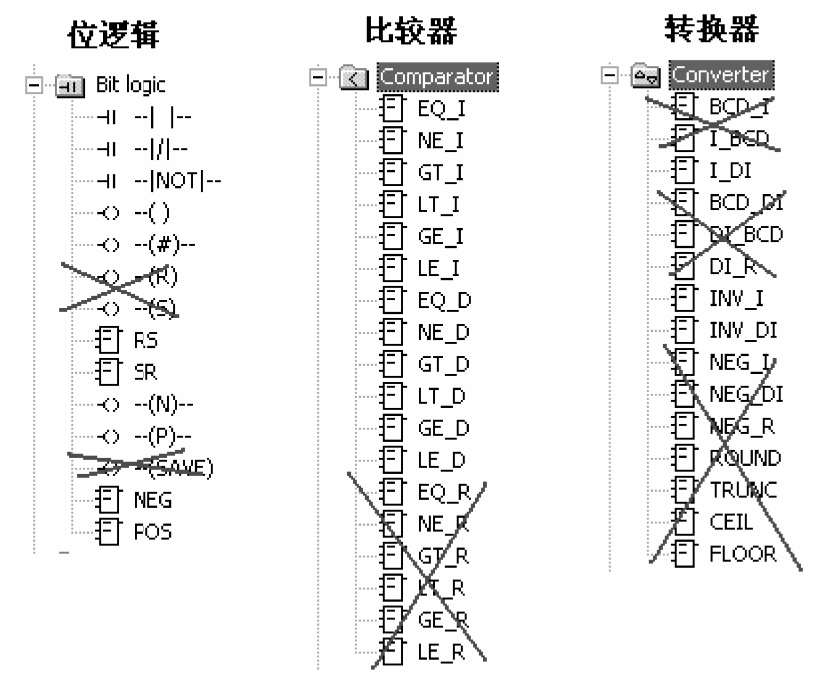



图 11-27 FM352-5 不能使用的指令

SAVE 指令。在比较器和转换器中，不能使用涉及浮点运算的指令，计数器、定时器，则需要使用 FM352-5 专用的函数库。安装了 FM352-5 的参数化软件后，在“SIMATIC Manager”中自动集成 FM352-5 专用的函数库，如图 11-28 所示，函数库中包含计数器、定时器、加减乘除函数块、移位寄存器函数块等，如果在 FM352-5 中使用非法的指令，则在程序编译时会报错。FM352-5 的执行程序只能存储于一个函数块 FB 中，执行程序中不能使用 M、DB、I、Q、L、PIW、PQW 等数据区，也不能使用绝对地址，必须使用 FB 中的形参进行程序的编写，FB 形参是由 IN、OUT 及 STAT 接口区组成，可以是数组、结构及 FB 等数据类型。FM352-5 使用的函数块 FB 形参有固定的

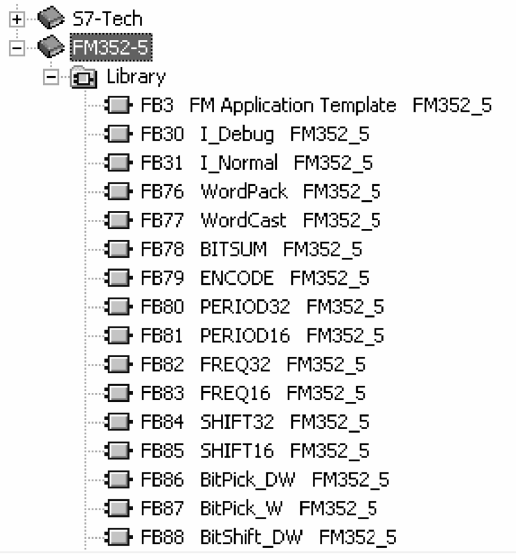



图 11-28 FM352-5 专用函数库

格式，有的形参可以添加修改，有的则不能做任何改动。函数块 FB 输入接口区的形参见表 11-6 所示。

**表 11-6　FB 输入接口区的形参**

| 地址 | 声明 | 名称 | 类型 | 注　　解 |
|---|---|---|---|---|
| 0.0 | in | DIn | ARRAY［0..14］ | FM352-5 集成的数字输入（0～11 = 24V，12～14 = RS422 差分输入信号，地址不能修改，但是数据类型及名称可以修改 |
| *0.1 | in | | BOOL | |
| 2.0 | in | CPU_Out | STRUCT | CPU 传送到 FM352-5 的 14 个字节，不能修改 |
| +0.0 | in | Bits | ARRAY［0..15］ | 名称和数据类型都可以修改，但是通信的字节数不能超过 14 个字节 |
| *0.1 | in | | BOOL | |
| +2.0 | in | T1_PVT | DIN | |
| +6.0 | in | T2_PV | BYTE | |
| +7.0 | in | CmpByte | BYTE | |
| +8.0 | in | C1_PV | INT | |
| +10.0 | in | CP_Period | WORD | |
| +12.0 | in | CMPInt | INT | |
| =14.0 | in | | END_STRUCT | 不能修改 |

输入接口区的形参为集成的数字量输入信号（2B）和 CPU 发送的命令信号（14B），形参使用符号名寻址，例如集成的第一个输入信号地址表示为 Din［0］，而不能使用 IX. X。函数块 FB 输出接口区的形参见表 11-7 所示。

**表 11-7　FB 输出接口区的形参**

| 地址 | 声明 | 名称 | 类型 | 注　　解 |
|---|---|---|---|---|
| 16.0 | out | DOut | ARRAY［0..7］ | FM352-5 集成的 8 个数字量输出，地址不能修改但是数据类型及名称可以修改 |
| *0.1 | out | | BOOL | |
| 18.0 | out | CPU_in | STRUCT | CPU 接收 FM352-5 的信息，14B，不能修改 |
| +0.0 | out | Bits | ARRAY［0..15］ | 名称和数据类型都可以修改，但是通信的字节数不能超过 14B |
| *0.1 | out | | BOOL | |
| +2.0 | out | T2_CVasByte | BYTE | |
| +3.0 | out | C1_CVasByte | BYTE | |
| +4.0 | out | T2_CV | INT | |
| +6.0 | out | T1_CV | DINT | |
| +10.0 | out | Enc_CV1 | DINT | |
| =14.0 | out | | END_STRUCT | 不能修改 |

输出接口区的形参为集成的数字量输出信号（2B）和向 CPU 发送的状态信息（14B）。函数块 FB 静态接口区的形参见表 11-8 所示。

**表 11-8　应用 FB 的静态形参格式**

| 地址 | 声明 | 名称 | 类型 | 注　　解 |
|---|---|---|---|---|
| 32.0 | stat | Intr | ARRAY［0..7］ | FM352-5 触发过程中断的事件源，不能改变 |
| *0.1 | stat | | BOOL | |
| 34.0 | stat | ST | STRUCT | FM352-5 模块的诊断信息，不能修改 |

（续）

| 地址 | 声明 | 名称 | 类型 | 注　解 |
|---|---|---|---|---|
| +0.0 | stat | FIRSTSCAN | BOOL | FM352-5 从 STOP 到 RUN 的第一个扫描，不能修改 |
| +0.1 | stat | M3L | BOOL | 3L 电源断电，不能修改 |
| +0.2 | stat | ESSF | BOOL | 编码器电源过载，不能修改 |
| +0.3 | stat | M2L | BOOL | 2L 电源断电，不能修改 |
| +0.4 | stat | M1L | BOOL | 1L 电源断电，不能修改 |
| +2.0 | stat | OVERLOAD | ARRAY［0..7］ | 模块集成的输出信号电压过载，不能修改 |
| *0.0 | stat | | BOOL | |
| =4.0 | stat | | END_STRUCT | 不能修改 |
| 38.0 | stat | Encoder | STRUCT | 编码器数据，如果使用编码器则不能修改，如果没有连接编码器，编码器的形参可以删除 |
| +0.0 | stat | Direction | BOOL | |
| +0.1 | stat | Home | BOOL | |
| +0.2 | stat | Homed | BOOL | |
| +0.3 | stat | Overflow | BOOL | |
| +0.4 | stat | Underflow | BOOL | |
| +0.5 | stat | SSIFrame | BOOL | |
| +0.6 | stat | SSIDataReady | BOOL | |
| +0.7 | stat | Open_Wire | BOOL | |
| +1.0 | stat | Hold | BOOL | |
| +1.1 | stat | Reset | BOOL | |
| +1.2 | stat | Load | BOOL | |
| +2.0 | stat | Cur_Val | DINT | |
| +6.0 | stat | Load_Val | DINT | |
| +10.0 | stat | | END_STRUCT | |
| 48.0 | stat | UCtrl | “CTU16” | 可以根据用户的需求添加 FM352-5 的函数块，例如 FB121。加入的数量与 FM352-5 的资源有关 |
| 60.0 | stat | FF | STRUCT | 专用于 S/R，R/S 触发器，不能修改 |
| +0.0 | stat | FirstFF | BOOL | 名称可以修改，数量可以根据需要增加 |
| +0.1 | stat | SecondFF | BOOL | |
| +0.2 | stat | ThirdFF | BOOL | |
| +2.0 | stat | MoreFFs | ARRAY［0..15］ | |
| *0.1 | stat | | BOOL | |
| =4.0 | stat | | END_STRUCT | |
| 64.0 | stat | Edge | STRUCT | 专用于信号沿检查，不能修改 |
| +0.0 | stat | FirstEdge | BOOL | 名称可以修改，数量可以根据需要增加 |
| +0.1 | stat | SecondEdge | BOOL | |
| +0.2 | stat | ThirdEdge | BOOL | |
| +2.0 | stat | Edge4to10 | ARRAY［4..10］ | |
| *0.1 | stat | | BOOL | |
| +4.0 | stat | LastEdge | BOOL | |
| =6.0 | stat | | END_STRUCT | |
| 70.0 | stat | Conn | STRUCT | 专用于连接器，相当于普通 CPU 的中间变量，不能修改 |
| +0.0 | stat | XCon | BOOL | 名称可以修改，数量可以根据需要增加 |
| +2.0 | stat | arrXCon | ARRAY［0..31］ | |
| *0.1 | stat | | BOOL | |

（续）

| 地址 | 声明 | 名称 | 类型 | 注　　解 |
|---|---|---|---|---|
| +6.0 | stat | ICon | INT | 名称可以修改，数量可以根据需要增加 |
| +8.0 | stat | arrICon | ARRAY［0..3］ | |
| *2.0 | stat | | INT | |
| +16.0 | stat | DICon | DINT | |
| +20.0 | stat | arrDICon | ARRAY［0..3］ | |
| *4.0 | stat | | DINT | |
| =36.0 | stat | | END_STRUCT | |

形参“Intr”中每一个位都可以产生过程中断，产生的中断在 CPU 中调用相应的 OB 块，例如 OB40，通过 OB40 中的形参“OB40_POINT_ADDR”第一个字节的 8 个位，可以判断产生中断的事件是由“Intr”中哪一个位触发的；“ST”为 FM352-5 模块的诊断信息；“Encoder”为编码器的数据区，可以对编码器进行读写操作；在接下来的数据区可以自由插入 FM352-5 专用的函数块，例如计数器和定时器等，数量与 FM352-5 的资源有关；“FF”专用于 S/R，R/S 触发器的寄存器，不能作为其他数据区使用；“Edge”专用于信号沿检查的寄存器，不能作为其他数据区使用；“Conn”专用于连接器，相当于普通 CPU 的中间变量，不能作为其他数据区使用。

FM352-5 模块只能使用上面介绍的形参，并以符号名寻址方式编程，下面以示例的方式介绍 FM352-5 的编程方法，假设当 FM352-5 的第一个输入点为 1 时，使能 FM352-5 第一个输出点输出 10kHz 脉冲，脉冲频率由 CPU 设定并将频率设定值读回到 CPU 中进行校验。根据控制要求，可以使用 FM352-5 专用函数库中的 FB119 产生脉冲输出，FB119 的输入、输出参数见表 11-9 所示。

**表 11-9　FB119 输入、输出参数**

| 参　　数 | 数 据 类 型 | 输入/输出 | 功　　能 |
|---|---|---|---|
| ENABLE | BOOL | 输入 | 开始输入信号 |
| Q | BOOL | 输出 | 脉冲输出 |
| PERIOD | WORD | 输入 | 脉冲周期，最小 20μs |

参数“PERIOD”等于 50000 除以所需要的频率，例如当“PERIOD”= W#16#C350 时，输出频率为 1Hz；当“PERIOD”= W#16#1 时，输出频率为 50kHz；当“PERIOD”= W#16#5 时，输出频率为 10kHz，所以从 CPU 中将 W#16#5 赋值参数“PERIOD”就能满足控制要求。FB119 的功能如图 11-29 所示。

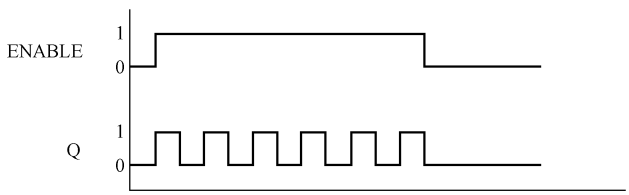



图 11-29　FB119 的功能图

当“ENABLE”为 1 时，Q 输出频率信号。

编程步骤如下：

1）建立 S7-300 系列 PLC 站，在硬件配置中插入 CPU 和 FM352-5，并编译存盘。

2）在“SIMATIC Manager”中，打开库“Library”，将“FM352-5”中函数块复制到应用程序中。其中 FB3 为 FM352-5 示例函数块，形参格式已经按照要求建立。

3）在硬件配置中，双击 FM352-5 图标，点击“Programming”标签，进入编程设置界面，如图 11-30 所示。

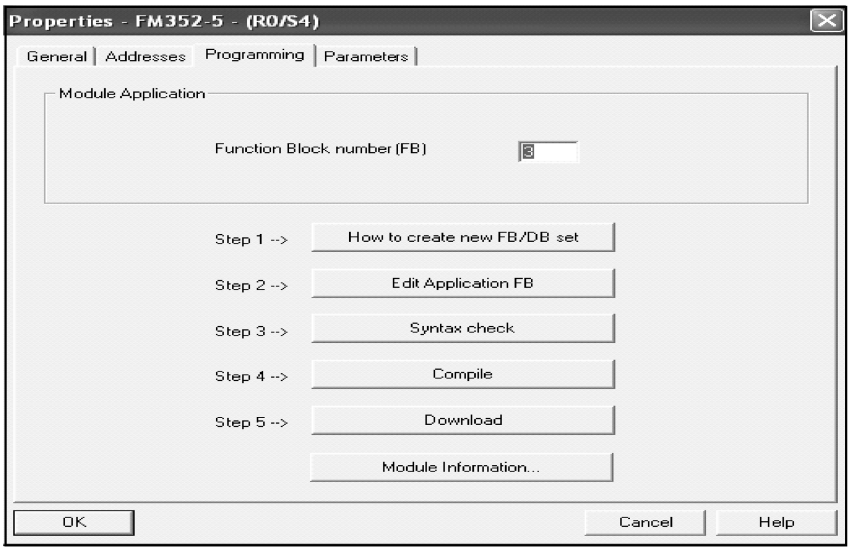



图 11-30　FM352-5“Programming”界面

在“Function Block number（FB）”处输入需要编写的 FB，本例中为 3，点击“Edit Application FB”，编辑 FB3。

4）在 FB3 接口区“OUT”中，加入形参“#CPU_In. Feedback1”（地址 2.0）作为 FM352-5 返回 CPU 的信息；在接口区“IN”中，加入形参“#CPU_Out. CP_Period1”（地址 10.0）作为 CPU 发送的命令；在“STAT”中，加入 FB119，命名为 CP1。

5）在 FB3 中编写程序如图 11-31 所示。

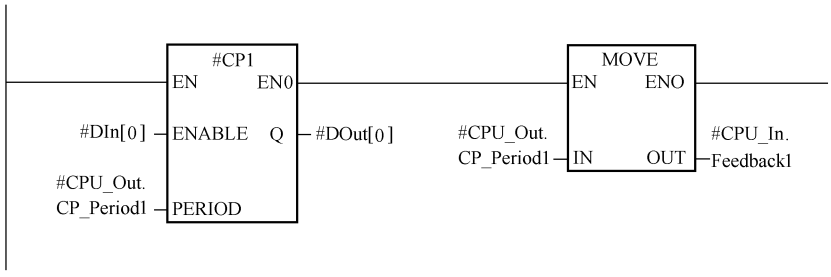



图 11-31　FB3 示例程序

“#DIn［0］”为 FM352-5 集成的第一个输入点，“#DOut［0］”为 FM352-5 集成的第一个输出点，FB119 参数“#CPU_Out. CP_Period1”为 CPU 发送的命令，并将发送的命令再返

回到 CPU 中，存盘后关闭 FB3。

6）分别点击图 11-30 中的第 3、4、5 步进行语法检查、代码编译，并将 FB3 中的程序下载到 FM352-5 中（联机状态），在编译时显示程序占用 FM352-5 资源的百分比。

7）在 CPU 中调用 FB31 功能块，建立与 FM352-5 的通信，示例程序如下：

```
CALL    " I_Normal"  ," NormalDB"
        Run          : = M1. 1          //启动 FM352-5
        OneScan      : = M1. 2          //FM352-5 单次扫描
        LADDRIn      : = L#256          //FM352-5 输入逻辑地址
        LADDROut     : = L#256          //FM352-5 输出逻辑地址
        CPU_Out      : = M100. 0        //CPU 发送的命令，从 MB100 开始的
                                          14B
        CPU_In       : = M120. 0        //CPU 接收 FM352-5 返回的信息，从
                                          MB120 开始的 14B
        Error        : = M1. 3          //通信故障位
        Status       : = MW2            //通信状态字
        L            W#16#5             //将脉冲周期命令发送到 FM352-5
        T            MW 110             //与#CPU_Out. CP_Period1 相对应
```

M1. 1 为 1 时，通信建立，FM352-5 运行，并将 W#16#5 发送到 FM352-5 的数据接收区 #CPU_Out. CP_Period1中，这样当 FM352-5 第一个输入信号为 1 时，第一个输出点输出 10kHz 频率。FM352-5 模块专用函数库中的 FB30 为调试块，当在 CPU 中调用 FB30 时，FM352-5 模块集成的输入、输出点由 CPU 接替控制，适合于调试阶段程序模拟，FB30、FB31 不能同时调用。

FM352-5 的示例程序 FM352-5. zip、参数化软件及电子手册参考光盘“功能模块 FM”目录。

### 11. 2. 5　FM352-5 的编程资源

FM352-5 的编程资源分为 1200 份，其中 436 份已被诊断或硬件配置占用，每个指令或多或少都需要占用编程资源，见表 11-10。

**表 11-10　指令占用 FM352-5 的编程资源**

| 指　令 | 所占的份数 | 指　令 | 所占的份数 | 指　令 | 所占的份数 | 指　令 | 所占的份数 |
|---|---|---|---|---|---|---|---|
| 触发器类型 | | 移位寄存器类型（接上） | | BitCast_W_U | 0 | 逻辑操作类型 | |
| BISCALE * | 2 | SHR_W * | 34 | BitPack_DW * | 17 | AND | 1 |
| CP_GEN * | 29 | SHR_W_U | 34 | BitPack_DW_U | 0 | OR | 1 |
| POS * | 2 | FIFO32 * | 19 | BitPack_W * | 9 | XOR | 1 |
| NEG * | 2 | FIFO16 * | 19 | BitPack_W_U | 0 | 字逻辑操作类型 | |
| SR * | 1 | LIFO32 * | 21 | BitInsert32 * | 33 | WAND_W * | 9 |
| RS * | 1 | LIFO16 * | 21 | BitInsert32_U | 32 | WAND_W_U | 8 |
| 计数器类型 | | BitShift_DW * | 17 | BitInsert16 * | 17 | WAND_DW * | 17 |
| CTD16 * | 36 | BitShift_W * | 19 | BitInsert16_U | 16 | WOR_DW_U | 16 |
| CTU16 * | 31 | 运算指令类型 | | 编码器类型 | | WOR_W * | 9 |

（续）

| 指　令 | 所占的份数 | 指　令 | 所占的份数 | 指　令 | 所占的份数 | 指　令 | 所占的份数 |
|---|---|---|---|---|---|---|---|
| CTUD16 * | 47 | FMABS16 | 18 | Encoder 16 bit | 64 | WOR_DW * | 17 |
| CTUD32 * | 99 | FMABS32 | 37 | Encoder 32 bit | 117 | WOR_W_U | 8 |
| 定时器类型 | | FMADD16 | 9 | SSI master 13 bit | 61 | WOR_DW_U | 16 |
| TOF16 * | 26 | FMADD32 | 17 | SSI master 25 bit | 100 | WXOR_DW * | 17 |
| TOF32 * | 55 | FMDIV16 * | 86 | SSI listen 16 bit | 77 | WXOR_DW_U | 16 |
| TON16 * | 25 | FMDIV32 * | 153 | SSI listen 32 bit | 122 | WXOR_W * | 9 |
| TON32 * | 53 | FMMUL16 * | 62 | None | 0 | WXOR_W_U | 8 |
| TP16 * | 26 | FMMUL32 * | 118 | 比较器类型 | | 其他类型 | |
| TP32 * | 54 | BITSUM * | 21 | CMP16_EQ | 6 | FREQ32 * | 71 |
| 移位寄存器类型 | | BITSUM_U | 21 | CMP16_GE | 8 | FREQ16 * | 51 |
| SHIFT * | 18 | ENCODE * | 19 | CMP16_GT | 8 | PERIOD32 * | 43 |
| SHIFT2 * | 18 | ENCODE_U | 19 | CMP16_LE | 8 | PERIOD16 * | 23 |
| SHIFT4 * | 18 | 数据传送类型 | | CMP16_LT | 8 | = = （INT） | 6 |
| SHIFT8 * | 19 | MOVE（锁存） | 17 | CMP16_NE | 6 | > = （INT） | 8 |
| SHIFT16 * | 21 | MOVE_U（无锁存） | 0 | CMP32_EQ | 11 | > （INT） | 8 |
| SHIFT32 * | 29 | DATSEL16 | 8 | CMP32_GE | 25 | < = （INT） | 8 |
| SHR_I * | 36 | DATSEL32 | 16 | CMP32_GT | 25 | < （INT） | 8 |
| SHR_I_U | 36 | WordPack * | 17 | CMP32_LE | 25 | < > （INT） | 6 |
| SHR_DI * | 88 | WordPack_U | 0 | CMP32_LT | 25 | = = （DINT） | 11 |
| SHR_DI_U | 87 | WordCast * | 17 | CMP32_NE | 11 | > = （DINT） | 25 |
| ROL_DW * | 81 | WordCast_U | 0 | 转换类型 | | > （DINT） | 25 |
| ROL_DW_U | 80 | BitPick_DW * | 10 | I_DI * | 9 | < = （DINT） | 25 |
| SHL_DW * | 81 | BitPick_DW_U | 10 | I_DI_U | 0 | < （DINT） | 25 |
| SHL_DW_U | 80 | BitPick_W * | 5 | INV_DI * | 17 | < > （DINT） | 11 |
| SHL_W * | 35 | BitPick_W_U | 5 | INV_DI_U | 0 | | |
| SHL_W_U | 34 | BitCast_DW * | 17 | INV_I * | 9 | | |
| SHR_DW * | 81 | BitCast_DW_U | 0 | INV_I_U | 0 | | |
| SHR_DW_U | 81 | BitCast_W * | 9 | | | | |

* 号表示指令具有存储功能，需要多项时钟控制；_U 表示没有锁存功能和存储功能。

此外，诊断和中断也需要占用 FM352-5 的资源，见表 11-11。

**表 11-11　诊断和中断占用 FM352-5 的资源**

| 参　数 | 所占的份数 |
|---|---|
| 模块诊断硬件支持 | |
| 1L 电源掉电 | 3 |
| 2L 电源掉电 | 11 |
| 编码器电源故障 | 12 |
| 3L 电源掉电 | 11 |
| SSI 编码器值溢出 | 34 |
| 差分输入编码器断线 | 10 |
| 输出诊断硬件支持 | |

（续）

| 参　　数 | 所占的份数 |
| --- | --- |
| 输出过载 Q0 ~ Q7 | 每个 12 个 |
| 过程中断硬件支持 | |
| 过程中断 0 ~ 7 | 每个 4 个 |

在编写程序时，需要注意使用指令的次数，以避免超出 FM352-5 的限制，在编译用户程序时，会提示所占总资源的百分比，可以先编写部分程序，然后根据容量提示优化程序。

## 11.3　定位模块概述

定位模块通过控制双速电动机、步进电动机、伺服电动机的位置环达到定位的目的。根据工艺要求和成本，选择不同的电动机进行定位控制，不同的电动机定位精度不同，普通异步交流电动机的定位精度大于 5mm，西门子公司的紧凑型异步交流电动机的精度可以达到 0.1mm，如果选择伺服电动机，则控制精度可以达到 0.001mm。不同的电动机需要不同的驱动装置控制，例如西门子公司的 MM4 和 Master Driver 变频器控制普通异步交流电动机和紧凑型异步交流电动机，SIMODRIVE-611U 和 SINAMIC S120 可以控制伺服电动机，通常情况下驱动装置控制电动机的速度环和电流环，定位模块通过编码器作为当前位置和速度的反馈，并将控制命令发送到驱动装置进行位置环的控制。对于不同的控制要求，S7 系列 PLC 提供不同的定位控制模块和具有定位功能的 CPU。

### 11.3.1　双速电动机的定位控制

S7-300 系列 PLC 系列中 FM351 模块、S7-300C 系列 PLC CPU 集成的定位功能、S7-400 系列 PLC 中 FM451 模块以及 ET200S 1POS 模块中都是通过控制双速电动机实现定位功能，控制方式如图 11-32 所示。

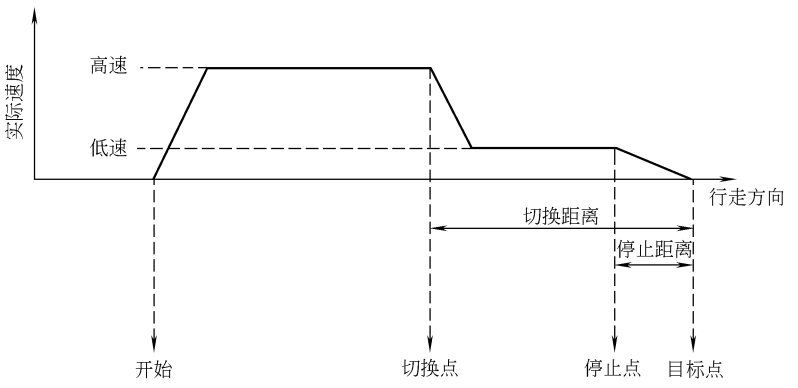



图 11-32　双速电动机控制方式

从图 11-32 中可以看到，电动机的速度只有两档 – 高速和低速。在定位控制时，首先使用高速行走（大部分距离），在行走到切换点位置时，由高速切换到低速，当穿过停止点时，电动机停车，完成定位任务。切换距离、停止距离在模块配置时，根据工艺要求设定，

如果切换距离长，则定位的时间长，下一个定位任务需要等待上一个定位任务完成后才能执行；如果距离太短，则有可能到达目标点位置时停不下来，物体的行走方向需要反向，同样延长了定位时间。切换距离和停止距离可以根据现场电动机负载的情况进行微调，以达到良好的控制效果。除控制双速电动机外，这种控制方式还可以通过控制变频器实现，通过数字量输出控制变频器的两个固定速度，穿过停止点时，电动机抱闸停车；有的模块可以通过模拟量输出控制变频器的速度，但速度并不是任意可调，在模块配置时需要设定两个固定的电压值与变频器的速度相对应，从而控制电动机的两个速度－高速和低速。

### 11.3.2 步进电动机的定位控制

对于伺服电动机而言，步进电动机的功率、转矩较小，转速比较低，控制精度可以达到0.005mm，控制系统整体价格低。西门子公司在不同的 PLC 系列都有相应的步进电动机控制模块，例如 S7-200 系列 PLC 系列中的 EM253 模块；S7-300 系列 PLC 系列中的 FM353 模块（单轴控制）和 FM357 模块（4 轴定位，可以进行路径等复杂控制）；S7-400 系列 PLC 系列中的 FM453 模块（步进电动机和伺服电动机控制）以及分布式 I/OET200S 系列的 1STEP 模块，这些定位模块将脉冲信号和方向信号发送到步进电动机控制器，从而控制步进电动机，定位模块通过输出脉冲的个数控制步进电动机行走的距离，通过输出脉冲的频率控制步进电动机的速度。典型的步进电动机定位控制方式如图 11-33 所示（以 FM353 模块为例）。

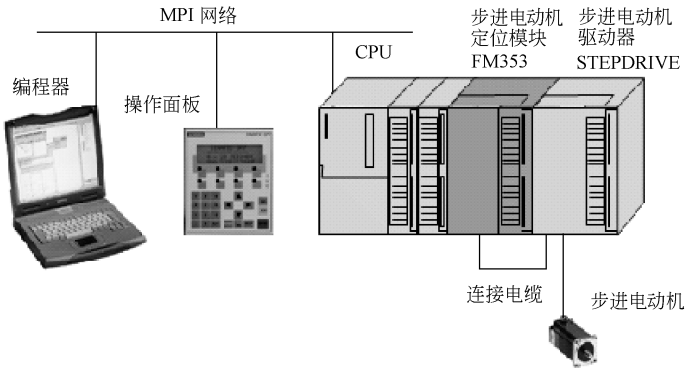



图 11-33 步进电动机控制方式

从图 11-33 中可以看到，通过编程器对 FM353 模块进行参数化，将电动机最大加速度、速度、电动机旋转一周对应的实际行走的距离，以及与 CPU 接口等参数下载到 FM353 模块中，通过操作面板，将位置给定和控制方式等控制命令，直接传送到 FM353 模块中，由 FM353 模块实现步进电动机位置环的控制，FM353 模块将脉冲输出信号发送到步进电动机驱动器，完成对步进电动机的控制。步进电动机定位模块将实际输出的脉冲个数作为位置的反馈，而没有将实际的位置信号通过编码器反馈到位置控制回路中，所以步进电动机的定位是位置开环控制（FM357 可以选择闭环控制）。定位模块中已经固化位置控制算法和控制模式，可以选择使用。FM353/FM357/FM453 在参数化中将脉冲个数转换成为长度单位，以位置值作为参考，方便编程，此外模块还集成了 G 代码的编程方式，并将程序存储于模块中。

### 11.3.3　伺服电动机的定位控制

使用伺服电动机定位可以满足动态响应快、控制精度高的工艺要求，伺服电动机的位置控制精度可以达到 0.001mm，输出功率可以达到 30kW，转矩可以达到 140N·m，最高转速可达 6000r/min 以上，控制系统整体价格较高。西门子公司在不同的 PLC 系列都有相应的伺服电动机控制模块，例如 S7-300 系列 PLC 系列中的 FM354 模块（单轴控制）和 FM357 模块（4 轴定位，可以进行路径等复杂控制）；S7-400 系列 PLC 系列中的 FM453 模块（步进电动机和伺服电动机控制）、FM354 模块、FM357 模块、FM453 模块通过输出电压信号（+/-10V）控制伺服驱动器，从而对伺服电动机进行控制。输出电压信号的正负表示控制的方向，电压绝对值的大小与控制速度成正比，典型的伺服电动机定位控制方式如图 11-34 所示（以 FM354 模块为例）。

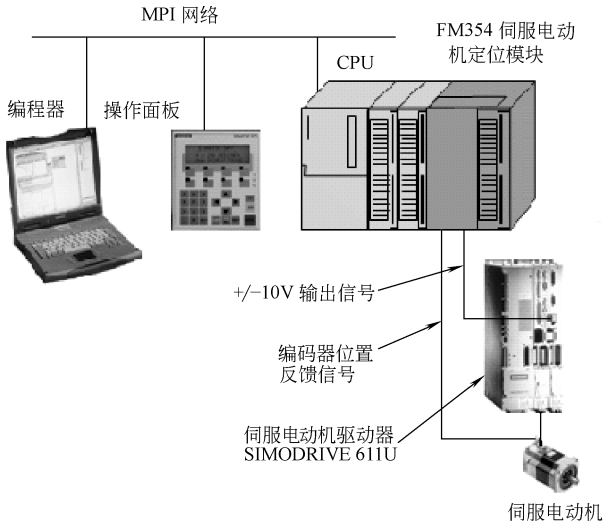



图 11-34　伺服电动机控制方式

通过编程器对 FM354 模块进行参数化，将电动机最大加速度、速度、电动机旋转一周对应的实际行走的距离，以及与 CPU 接口等参数下传到 FM354 模块中。通过操作面板将位置给定和控制方式等控制命令直接传送到 FM354 模块中，FM354 模块将电压输出信号传送到伺服电动机驱动器中，从而对伺服电动机进行控制。与步进电动机定位控制相比，FM354 模块将实际的位置信号通过编码器反馈到位置控制器回路中，所以伺服电动机的定位是位置闭环控制。定位模块中已经固化位置控制算法和控制模式，可以选择使用。FM354、FM357、FM453 模块以位置值作为参考，方便编程，除此之外模块还集成了 G 代码的编程方式，并将程序存储于模块中。

无论控制信号是电压信号，还是脉冲信号，都是模拟输出信号，因此控制信号可能会有漂移。通过 PROFIBUS-DP 网络连接驱动装置，传送的数字信号不会造成位置的漂移，FM357-2 模块、集成伺服定位功能的 T_CPU（CPU315T 和 CPU317T）以及 SIMOTION 定位控制器都可以通过 PROFIBUS-DP 接口连接伺服驱动装置。通过 PROFIBUS-DP 连接伺服驱动装置的典型配置如图 11-35 所示（以 T_CPU 为例）。

在图 11-35 中，CPU 通过 PROFIBUS-DP 等模式技术连接驱动装置，可以将控制命令同时发送到每一个驱动装置，并将位置反馈信号同时传送到 CPU 中，保证有联动关系的轴之间的控制精度（例如 Gear 和 Cam 的控制）；连接轴的个数与 PROFIBUS-DP 扫描的时间有关，连接轴的个数越多，PROFIBUS-DP 扫描的时间越长，这样位置差补计算的间隔长，可能不能满足控制工艺的要求，除 FM357-2（只能连接 4 个轴）外，连接轴的个数参考 CPU 或 SIMOTION 的技术参数。

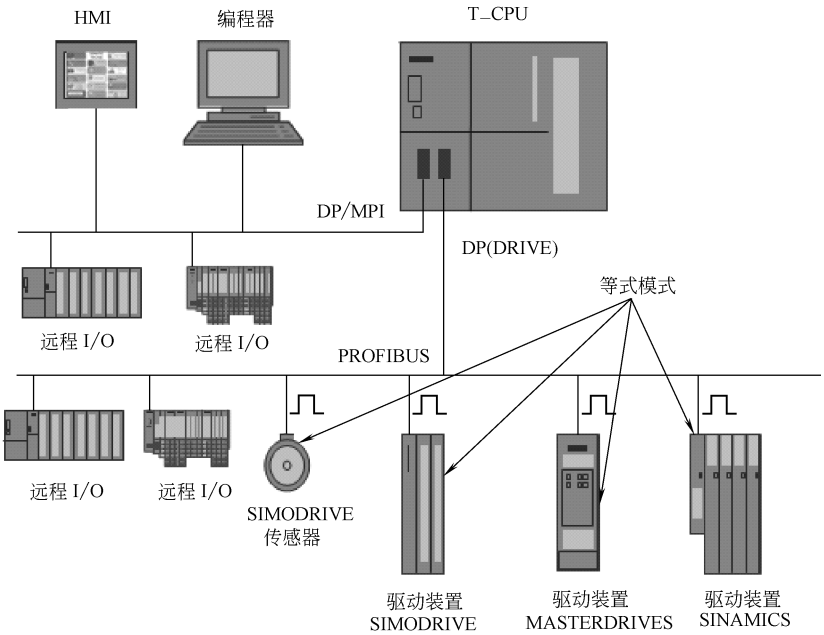



图 11-35 通过 PROFIBUS-DP 连接伺服驱动装置的位置控制

## 11.4 FM354 伺服电动机定位模块的使用

在图 11-34 中已经简单介绍了 FM354 的典型配置，模块可以安装于 S7-300 系列 PLC 中央机架，也可以安装于 ET200M 中，作为分布式 I/O 使用（主站只适合使用 S7-300、400 系列 PLC），CPU 将人机界面发送的命令传送到 FM354 中，并将实际值及状态值反馈到 CPU 和人机界面，FM354 中集成了固定的控制算法，位置和速度的控制将在 FM354 中完成，保证位置差补计算的快速性和稳定性。

### 11.4.1 FM354 模块的输入输出接口

FM354 模块的输入、输出接口如图 11-36 所示。

在 FM354 模块上分别标注 3 个输入、输出接口 X1 ~ X3：

X1 接口：集成的四个数字输入和四个输出点，输入、输出点功能可以在模块中进行配置，如回零的参考点信号、外部电源使能信号、输出快速响应信号及显示控制方向的正反转信号等。

X2 接口：+/－10V 模拟信号，连接控制伺服电动机的驱动器，如西门子公司的 SIMODRIVE-611U，电压的大小与控制电动机的转速成正比，信号的正反控制电动机旋转的方向。在 X2 接口中的数字量输出信号可以作为控制器的使能信号。

X3 接口：测量端口，连接编码器信号，FM354 模块只能连接 5V RS-422 对称脉冲增量型编码器和具有同步串行接口 SSI 的绝对值编码器作为位置反馈信号，而不能使用 24V 增量型编码器。

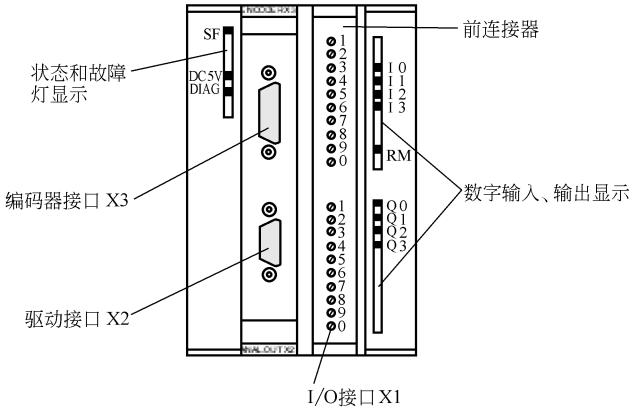



图 11-36　FM354 模块输入、输出接口

通过 FM354 模块的输入、输出接口，知道可以连接驱动器的类型必须是使用电压信号作为速度控制的驱动器，而不能连接使用脉冲信号作为速度控制的驱动器。

### 11.4.2　FM354 模块的操作模式

FM354 中集成固定的控制算法和模式，控制模式分为以下几种：

1）JOG 模式：点动模式，选择定位的速度值后，连接正反转控制信号，信号为 1 时，电动机运行；信号为 0 时，电动机停止。加速度在模块的参数化中设定。点动模式是位置闭环控制，如果启动命令发出后没有位置反馈值，输出的电压值将增加到最大值输出。

2）开环模式：速度控制，直接选择输出电压值，电压与控制速度成正比，连接正反转控制信号，信号为 1 时，按设定的电压值输出到 X2 接口控制电动机速度，信号为 0 时，电动机停止。与点动模式相比，开环模式只是对速度进行控制而不对位置进行控制，对位置控制是开环控制。

3）回零模式：回零的模式也是寻找基准点的过程，物体绝对位置的移动需要有基准点，例如物体绝对正向移动到第 50 个位置单位（可以是度、英寸、毫米等），首先需要确定零点，然后以零点为基准点移动 50 个位置单位，在增量模式、MDI（手动数据输入）模式和自动模式情况下必须先回零，得到与轴同步信号后才允许启动，否则报错。

4）增量模式：在增量表中（1 ~ 100 行）预先定义相对行走的距离后，在操作中通过选择增量表中存储相对位置值的行号，以达到定位的要求。在行数 1 ~ 100 中定义的位置值不能修改，如果需要的位置值是变量，则可以在程序中定义特殊行号 254（不能在增量表中设定）。启动增量模式必须先回零。

5）MDI 模式：MDI（手动数据输入），预先设定定位的方向、速度及相对或绝对的位置值后，通过启动信号达到定位的要求，定位任务不是连续的，上次定位任务完成后必须再次触发启动信号执行下一个定位任务。启动 MDI 模式必须先回零。

6）自动模式：利用 G 代码编写物体移动程序，并预先下载到 FM354 中，最多可以存储 199 个 G 代码定位程序。通过在 CPU 中调用存储于 FM354 模块中的程序，实现定位的任务，定位任务是连续，在一个 G 代码程序中可以编写 100 条位移块，每一个块中包含速度、位

置等指令。启动自动模式必须先回零。

7）自动单块模式：与自动模式相同，G 代码程序中每一个位移块需要单独启动。启动自动单块模式必须先回零。

FM354 中的控制模式及算法是固定的，不可以改变，单独一种控制模式往往不能满足实际的位置控制要求，通过灵活组合不同的控制模式可以完成大多数的定位控制任务。

### 11.4.3 FM354 模块的参数化

FM354 模块附带参数化软件，参数化软件也可以从西门子公司的网站上下载，软件以 STEP7 为安装平台。在硬件配置中插入 FM354 模块，双击模块，弹出属性对话框，在“Addresses”栏中，显示 FM354 模块的逻辑地址，在“Basic parameters”栏中，可以选择 FM354 模块是否产生中断、中断类型以及产生的中断是否影响 CPU 的运行等选项，点击按钮“Parameter”，进入 FM354 的参数化界面（如果没有安装参数化软件，“Parameter”按钮为虚，不能点击；如果没有联机，则会显示连接中断，从而进入离线参数化模式），如图 11-37 所示。

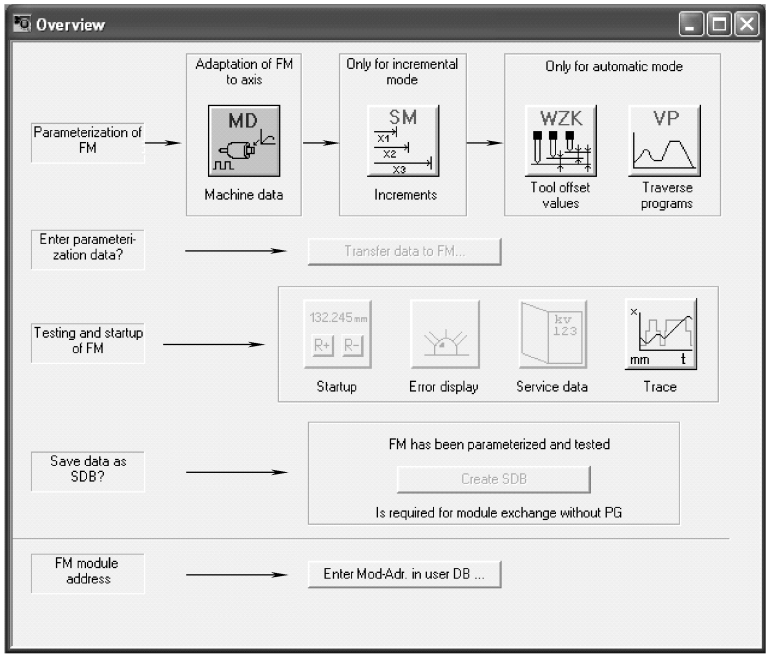



图 11-37 FM354 参数化界面

图 11-37 中的灰色按钮（不可以点击的按钮）只有在联机的情况下才能点击进入，这些按钮打开的操作界面与设备的调试有关，例如配置参数和程序的下载、联机运行、故障显示与确认、显示运行数据等。“Parameterization of FM”指向 4 个参数化界面的启动按钮 – MD、SM、WZK、VP，在这些界面中完成对 FM354 模块的参数化，各个界面实行的功能如下：

MD：机械参数，在该界面中对电动机、FM354 系统参数、连接编码器类型等参数进行设置。

SM：增量表，仅在增量模式下使用。

WZK/VP：工具补偿和 G 代码程序，仅在自动模式下使用。

### 11.4.4　MD 机械参数

首先点击 MD 按钮，进入机械参数配置界面，如图 11-38 所示。

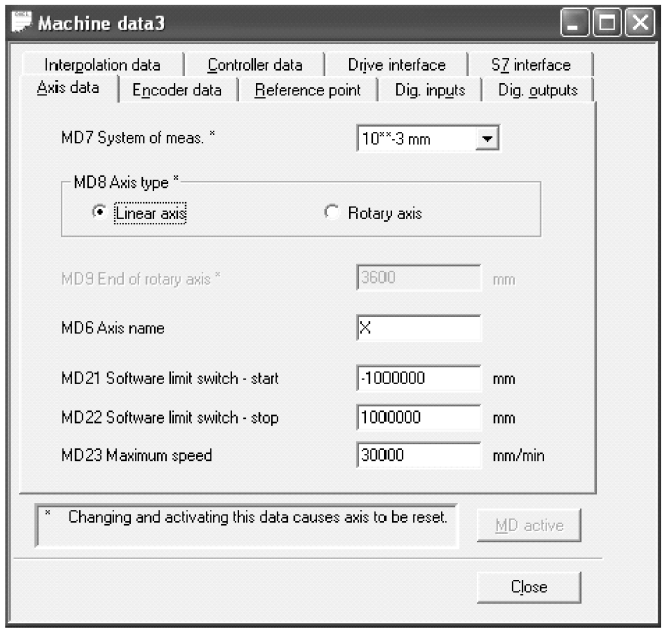



图 11-38　FM354 MD 参数化界面

界面中各个标签栏的功能及涉及的参数如下：

（1）在“Axis data”标签中，对轴参数进行设置

MD7：　选择系统的测量单位，在以后的参数化中以此单位为基准；

MD8：　选择轴的类型，如果选择旋转轴，在 MD9 中定义轴的开始点（结束点），如定义 360°，当轴旋转 360°时，回零重新开始计算旋转角度；

MD21/22：轴的工作范围，当定位超出此范围，电动机停止，在 MD21/22 定义的是软件限位，在实际的应用中可以外加硬件限位，如物体碰到硬件限位时切断电源，保证人员和设备的安全；

MD23：　定义电动机的最大速度。

（2）在“Encoder data”标签中，对编码器参数进行设置

MD10：　选择连接编码器的类型，在前面已经介绍，只有 5V 增量型编码器和 SSI 绝对值编码器可以作为位置反馈信号，由于 SSI 编码器为机械结构，记录的位置值在掉电后不会丢失，所以不需要回零操作，只需要在“Reference point”标签 MD17 中定义即可，使用增量型编码器在重新上电后位置值丢失，需要回零操作；

MD11/12：定义编码器旋转一周对应物体的位移，MD11 为整数部分，MD12 为小数部分；

MD13：　定义编码器旋转一周的脉冲数；

MD19.0： 选择旋转方向是否反向；

MD14/15：SSI 编码器参数；

MD20： 选择是否对编码器信号进行监控，如果编码器故障将产生诊断中断。

（3）在“Reference point”标签中，对回零操作参数进行设置

回零操作中需要安装参考点开关，它的作用是给回零操作作为一个基准点，参考点开关必须连接到 FM354 模块任意一个输入点上。主要有两种回零方式，一种是利用编码器的零脉冲（/N、N 相）寻找参考点，例如回零方式“Direction +，zero pulse right”，回零过程如图 11-39所示。

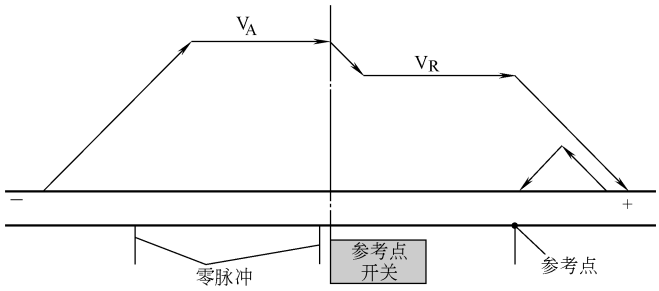



图 11-39 回零操作方式 1

$V_A$ 为设定回零操作的起始速度，在 MD28 中定义；$V_R$ 为碰到参考点开关后的速度，在 MD29 中定义。轴以 $V_A$ 速度正方向（向右）移动，当碰到参考点开关后速度降为 $V_R$，以参考点开关右边的编码器零脉冲点作为参考点。另一种是利用参考点开关寻找参考点，例如回零方式“Direction +，reference point switch center”以参考点开关的中央点作为参考点，回零过程如图 11-40 所示。

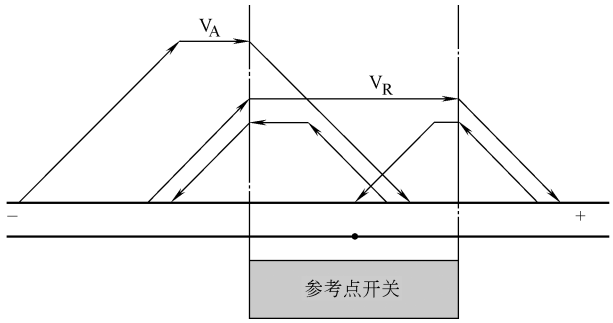



图 11-40 回零操作方式 2

物体以 $V_A$ 速度正方向（向右）移动，当碰到参考点开关后速度降为零，经过反复移动后，计算出参考点开关的长度，并以参考点开关的中心点作为参考点。

**注意：**

*轴经过参考点开关的时间要大于等于 2 倍的模块扫描时间（2ms）。*

与回零操作相关的参数如下：

MD18： FM354 回零有 8 种方式，选择其中一种回零方式；

MD16： 当前参考点的坐标位置；

MD17： 选择 SSI 绝对值编码器时定义的调整值；

MD27： 更换故障编码器时，新编码器与原编码器的零脉冲位置会有偏差，将偏差值填写在 MD27 中进行位置补偿；

MD28： 设定回零操作的起始速度；

MD29： 碰到参考点开关后的减速度。

（4）在“Dig. inputs”标签中，对模块集成的数字量输入点进行设置

在 MD34 中，可以选择每一个输入点的功能，例如在自动模式和 MDI 模式下使用的“External block change”、“Set actual value on-the-fly”功能；“Reference point switch for reference point approach”指定其中一个输入点作为参考点开关；“Reversing switch for reference point approach”指定其中一个输入点作为寻找参考点的反向开关，当物体碰到此开关后，反向寻找参考点开关；只有 4 个输入信号可供选择，在 MD36 中可以单独选择每个输入信号是否取反。

（5）在“Dig. outputs”标签中，对模块集成的数字量输出点进行设置

在 MD35 中，可以选择每一个输出点的功能，例如在自动模式和 MDI 模式下使用的“Change M97”、“ Change M97”功能，直接通过 G 代码程序控制输出而没有用户程序的扫描延迟，保证输出的快速响应；“Position reached、stop”、“ Axis movement forward”、“Axis movement reverse”可以输出到信号灯作为当前状态显示；“Direct output”CPU 可以直接控制输出等功能。只有 4 个输出信号可供选择。

（6）在“Interpolation data”标签中，对轴的加减速度参数进行设置

MD40：　轴位移的加速度；

MD41：　轴位移的减速度；

MD42：　设置速度斜坡拐点是否平滑。

（7）在“Controller data”标签中，对位置控制参数进行设置

MD38：　位置控制的比例参数；

MD37. 16：选择是否自动漂移补偿，在高温的条件下产生位置的漂移，选择该功能在位置闭环控制中自动补偿位置漂移；

MD39：　选择是否对最小跟踪误差进行监控，如果跟踪误差超出设定的范围模块，将报错；

MD24/25：定义定位目标范围和监控时间，如果超出设定的监控时间而实际的位置没有在设定的目标范围内，模块将报错；

MD26：　设定轴的停滞范围，定位完成，轴处于停止状态，如果由于外力而使轴移动并超出设定的范围，模块将报错。

（8）在“Drive interface”标签中，对模块与驱动器间的接口参数进行设置

MD37：　设定驱动的使能信号；

MD19. 1：选择是否更改模拟量输出信号的方向；

MD43：　设定模拟量最大的输出值，输出的最大电压值与 MD23 中设定的最大速度相对应；

MD44：　设定模拟量输出的偏移量补偿，有些情况下，模拟量输出有漂移，在没有定位命令时，电动机也会有小的移动，通过 MD44 补偿模拟量偏移；

MD45：　电压斜坡参数，在位置开环控制或 FM354 故障输出 0V 电压时，电压值的上升或下降斜坡参数；

MD30：　齿轮的间隙补偿；

MD31：　齿轮的间隙补偿方向；

MD52：　齿轮间隙补偿的速度，以最大速度 MD23 的百分比形式设置；

MD53：　齿轮间隙补偿的方式，可以选择在定位前或在定位中进行补偿。

（9）在“S7 interface”标签中，对模块与 CPU 间的接口参数进行设置

M 功能除在位移块（Traversing Block）中的特殊功能外，还可以将当前执行位移块的位置（那一条指令）传送到 PLC CPU 中（用户数据块 DBB26 显示 M 的数值），M 功能的输出可以在执行位移块前、后或者在执行中，也可以是时间控制，输出定义的时间长度。如果 M 功能需要 CPU 确认，则位移块输出 M 功能后等待 CPU 的确认后才能执行下一个位移块（用户数据块 DBX15.4 置位确认）。

MD32： 选择 M 功能的输出类型；

MD33： 如果在 MD32 中，选择 M 功能输出的类型为时间控制，在 MD33 中定义 M 功能输出的时间长度；

MD5： 选择触发过程中断的事件，例如在 S7-300 系列 PLC 中，过程中断调用 OB40；

MD37.15：选择紧急停止后是否连续操作，如果未选择该选项，轴在移动时紧急停止，将终止位移块的执行，恢复后将剩余的位置值删除，执行新的位置设定操作；

MD37.7：选择是否激活时间倍率操作，如果没有选择该选项，为速度倍率操作，例如倍率值从 100% 变为 200% 后，速度值变为 2 倍，加减速度不发生变化，0% ~255% 为速度倍率设定范围；如果选择时间倍率操作，0% ~100% 为时间倍率设定范围，例如倍率值从 100% 变为 50% 后，速度值变为原来的一半，加减速度变为原来的四分之一，定位时间缩短一半。

机械参数配置完成后生成 DB1200，DB1200 为 FM354 的系统数据块而非 CPU 中的数据块。

在线修改 MD 参数并下传到 FM354 模块中，必须点击“MD active”按钮，才能使修改的参数生效。

### 11.4.5 SM 增量表的配置

点击“SM”按钮，进入增量表配置界面，如图 11-41 所示。

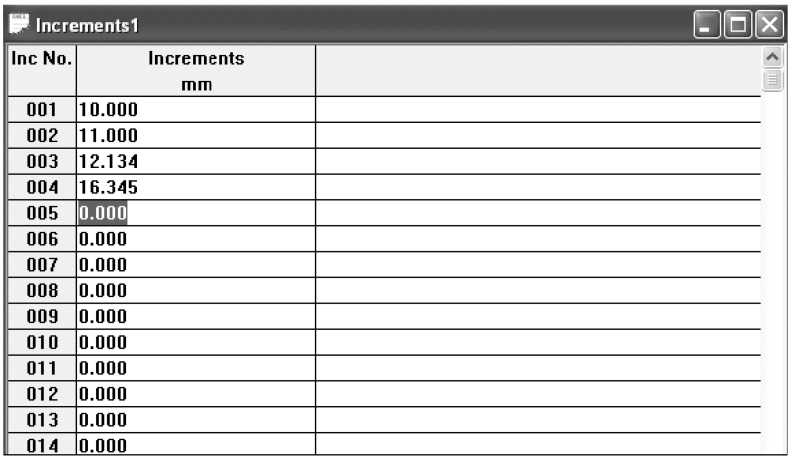



图 11-41 FM354 增量表参数化界面

在增量表中，预先设定相对位移值。在增量操作模式下，选择增量表的行号以确定需要

位移的距离，增量表中最多定义 100 个位移值。增量表配置完成后生成 DB1230，DB1230 为 FM354 的增量表块而非 CPU 中的数据块。

### 11.4.6　WZK 工具补偿参数的配置

点击“WZK”按钮，进入工具补偿配置界面，如图 11-42 所示。

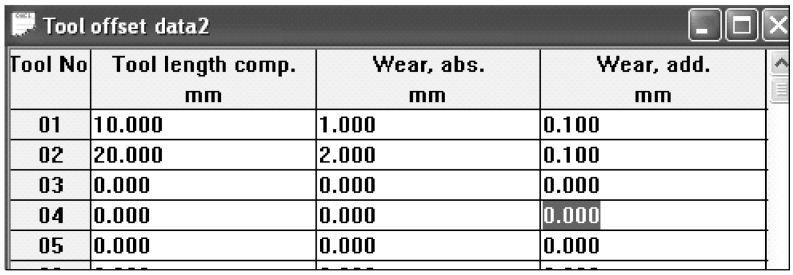



图 11-42　FM354 工具补偿参数化界面

在补偿界面预先设定工具长度补偿值、绝对磨损值和附加磨损值，在自动程序中可以调用工具补偿的行号，以确定需要补偿的长度，在界面中最多可以定义 20 行工具补偿值。工具补偿配置完成后生成 DB1220，DB1220 为 FM354 的工具补偿块而非 CPU 中的数据块。

### 11.4.7　VP 自动程序的编写

点击“VP”按钮，进入 G 代码编程界面，如图 11-43 所示。

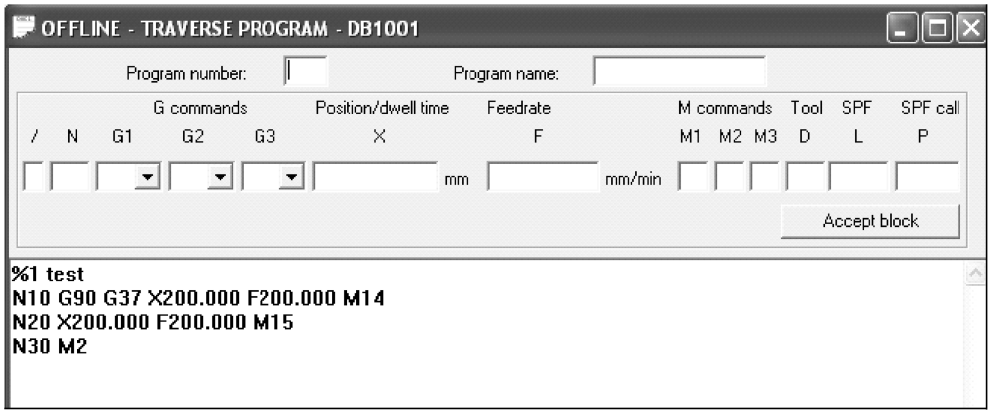



图 11-43　FM354 自动程序编程界面

“Program number”G 代码程序号从 1 ~ 199 可用，每一个 G 代码程序在 FM354 模块中生成一个包含程序的数据块 DB，程序号 1 ~ 199 对应数据块号 DB1001 ~ DB1199。每一个 G 代码程序最多可以编写 100 条位移块。每一个位移块中的指令结构如图 11-44 所示。

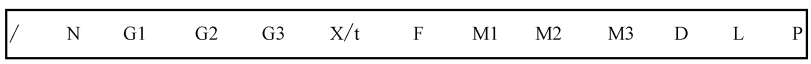



图 11-44　FM354 一条自动程序位移块的指令结构

指令解释如下：

/ ：是否跳过该条位移块，在 CPU 中置位用户数据块 DBX15.6 后，定位程序将跳过有“/”的位移块执行；

N ：位移块的序号，在 FM354 模块中，序号必须按从小到大次序排列，最大序号为 255；

G1 ：G 功能编组 1；

G2 ：G 功能编组 2；

G3 ：G 功能编组 3；

X/t ：位移值或停滞时间，如使用 G04 则为停滞时间，其他则为位移值；

F ：位移的速度；

M1～3：M 功能；

D ：工具补偿，调用 WZK 中预置的补偿值；

L ：调用的子程序号；

P ：子程序位移块的序号。

1．G 代码功能描述

G 代码指令划分为 3 组，功能描述及编组见表 11-12。

**表 11-12 FM354 G 功能代码**

| G 功能编号 | G 功能描述 | G 功能编组 |
|---|---|---|
| 04 * | 停滞功能，X/T 的值为停滞时间值，取值范围为 2～100000ms，带有停滞功能的位移块只能包含时间值和 M 功能 | 1 |
| 87 | 关闭在运行中设定实际值功能 | |
| 88 * | 正方向连续运行中设定实际值 | |
| 89 * | 负方向连续运行中设定实际值 | |
| 90 | 按绝对位置值运行 | |
| 91 | 按相对增量位置值运行 | |
| 30<br>31<br>32<br>⋮<br>39 | 加减速度按 100% 倍率计算<br>加减速度按 10% 倍率计算<br>加减速度按 20% 倍率计算<br>⋮<br>加减速度按 90% 倍率计算 | 2 |
| 43 | 工具补偿（+） | 3 |
| 44 | 工具补偿（-） | |
| 50 * | 外部条件触发位移块改变 | |
| 60 | 位移块改变—精确定位（设定的位置到达，速度停止后再执行下一个位移块） | |
| 64 | 位移块改变—连续路径（相对于 G60） | |

* 表示只在一个位移块内生效，其余的 G 功能代码使用后没有消除，功能保留，例如，在一个位移块中定义按绝对位置运行 G90 功能，在接下的位移块中如果没有重新定义，将按上一个移块中定义的 G90 功能运行，如果在接下的位移块中定义 G91 功能，G90 功能将被替代。

在一个位移块中最多可以包含 3 个 G 功能代码，G1、G2、G3 各一个，FM354 模块 G 功能初始定义为 G90、G30、G64。

（1）G90、G91 功能描述

G90 功能的使用如图 11-45 所示。

轴当前的绝对位置为 50mm，执行位移块 G90 X20 F100 后，轴从 50mm 的位置以 100mm$^{-3}$/min 的速度回到 20mm 的位置上。

G91 功能的使用如图 11-46 所示。

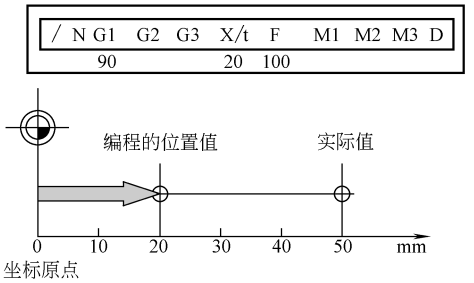



图 11-45　G90 功能

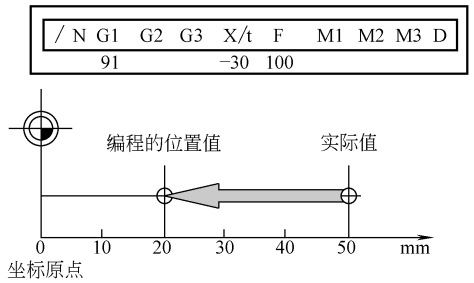



图 11-46　G91 功能

与上个位移块的功能相同，执行位移块 G91 X-30 F100 后，轴从当前 50mm 的位置上增加 −30mm，回到 20mm 的位置上。上面介绍了 G90/91 的功能是在直线轴上实现的，在旋转轴上操作应当注意最短执行距离问题，例如位移块 G90 X315 F100，轴当前位置为 45°，如果按启动（START）按钮执行位移块，轴将按最短距离运行，逆时针运行到 0°再到 315°，如果需要控制运行方向，必须同时使用方向控制信号 R + 或 R−；使用 G91 功能，可以在位置设定时通过正负号定义运行的方向；多圈操作时，只需将设定的位置值大于 360°即可。

（2）G87、G88、G89 连续运行中设定实际值功能

G87、G88、G89 的使用如图 11-47 所示。

| / | N | G1 | G2 | G3 | X/t | F | M1 | M2 | M3 | D |
|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 90 | | | 100 | 400000 | | | | |
| | 15 | 89(88) | | | 50 | 200000 | | | | |
| | 20 | 90 | | | 300 | 400000 | | | | |
| | 25 | 87 | | | 400 | 400000 | | | | |

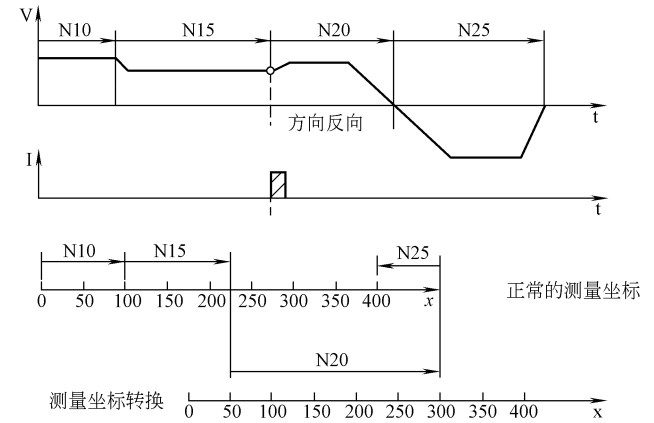



图 11-47　G87、G88、G89 功能

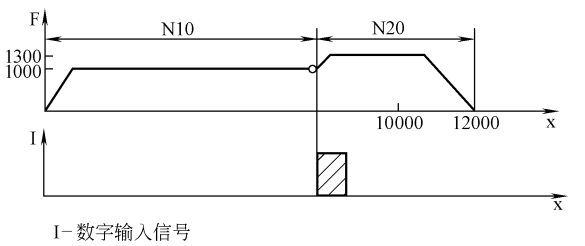
执行位移块 N10 指令，轴以 400000 的速度位移到绝对位置 100 后，跳转到位移块 N15，

在 N15 中使用 G88 或 G89 功能，以 200000 的速度位移，直到触发 FM354 模块集成的输入信号（输入信号在 MD34 中设置为“Set actual value on-the-fly”），将当前位置设置为 50 并跳转到位移块 N20，这样测量坐标转换为新的坐标测量系统，在后续执行的位移块以新坐标为基准，例如执行位移块 N20 指令，轴以 400000 的速度以新坐标为基准位移到绝对位置 300 后跳转到位移块 N25，在位移块 N25 中，G87 取消 G88 或 G89 坐标转换功能，切换回原先的坐标测量系统并位移到 400。X、F 的单位为 MD 机械参数中定义的测量单位，如 $mm^{-3}/min$。

（3）G50 外部块改变功能

G50 的功能是通过外部输入信号执行位移块的跳转，使用示例如图 11-48 所示。

| / | N | G1 | G2 | G3 | X/t | F | M1 | M2 | M3 | D |
|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | | | 50 | 10000 | 1000 | | | | |
| | 20 | | | | 12000 | 1300 | | | | |




I－数字输入信号

图 11-48　G50 功能

假设位移方向已经确定，执行位移块 N10 指令，轴以 1000 的速度位移，直到触发 FM354 模块集成的输入信号（输入信号在 MD34 中设置为“External block change”）才跳转到位移块 N20，并以速度 1300 位移到 12000，X、F 的单位为 MD 机械参数中定义的测量单位，如 $mm^{-3}/min$。如果在 N20 中没有设定位置值，触发输入信号后轴停止；如果执行 N10 位移指令超过设定的位置 10000 而没有触发输入信号，模块将报错。

2. M 功能

M 功能在 FM354 位移块中有两个作用，一是简单位移块控制；另一个功能是用户功能，将当前执行位移块的位置传送到 CPU 中。FM354 模块可用的 M 功能见表 11-13。

**表 11-13　FM354 模块可用的 M 功能**

| M 功能编号 | M 功能描述 | M 功能编组 |
|---|---|---|
| 0 | 如果 M0 功能在位移块中被调用，程序执行完成后停止，并输出 M0，必须有新的 START 命令才能执行后续的位移块 | 1，2，3 |
| 2，30 | 位移程序结束，输出 M2 或 M30，在调用子程序结束返回主程序时不输出 | |
| 1，3～17 | 用户功能 | |
| 18 | LOOP 功能，返回程序的开始，反复执行 | |
| 19～29　31～96 | 用户功能 | |
| 97，98 | 使能 MD35 中定义的快速输出 | |

M 功能分列在 3 个组中，在一个位移块中最多可以包含任意的 3 个 M 功能，输出的次序从 M1 功能组至 M3 功能组（在位移块中 M 功能的先后次序）。M 功能输出的方式在 MD32、MD33 中预先定义，输出的 M 功能在用户数据块中 DBB26 中显示，如果需要确认 M 功能，通过用户数据块中 DBX15.4 位进行确认。下面以示例的方式介绍 M 功能的使用。

（1）简单位移块控制功能

位移程序如图 11-49 所示。

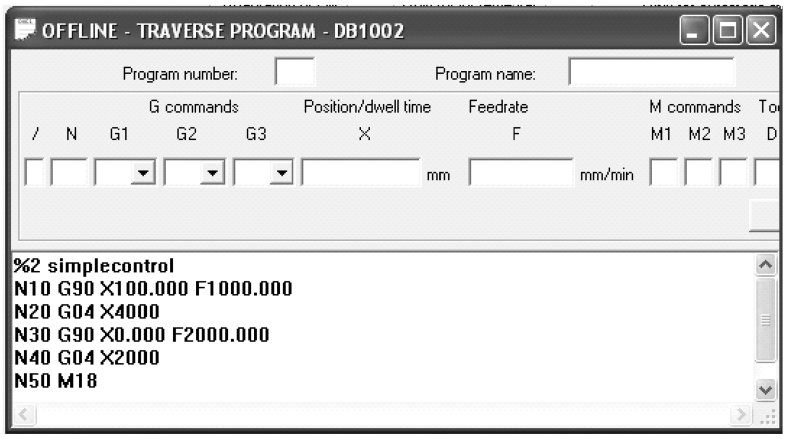



图 11-49　M18 程序控制功能

位移块 N10 使轴以 1000 的速度定位到 100，定位完成后执行 N20，轴停滞 4s 后执行 N30，轴以 2000 的速度回到零点后执行 N40，轴再次停滞 2s 后执行 N50，返回程序的开始点再次执行 N10，周而复始，使用 STOP 命令停止程序的执行。X、F 的单位为 MD 机械参数中定义的测量单位，如$mm^{-3}/min$。

（2）使用 M 的用户功能

位移程序块中 M 用户功能的使用如图 11-50 所示。

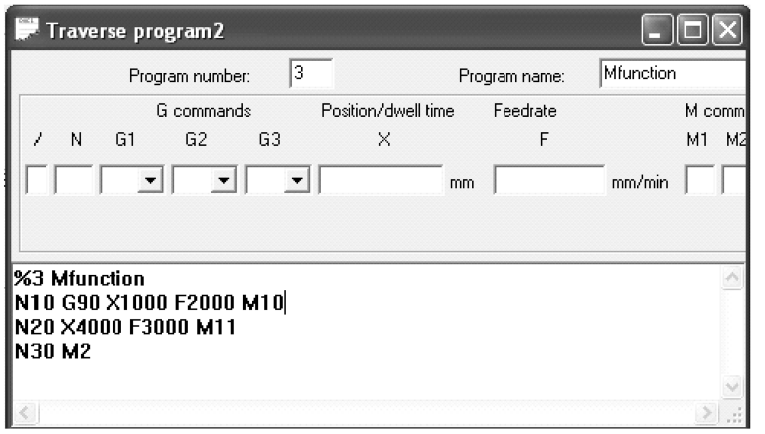



图 11-50　M 的用户功能

位移块 N10 使轴以 2000 的速度定位到 1000，输出 M10（输出的时序在 MD32 中定义），在用户数据块 DBB26 中显示 10，用户可以通过程序判断当前位移程序的位置，如果需要确认，通过置位用户数据块 DBX15.4 位进行确认后执行 N20，以同样的方式输出 M11，执行 N30 后，位移程序结束。

## 11.4.8　将参数化的数据传送到 FM354 中

参数化过程完成后，将配置的数据传送 FM354 模块中，存储于 FM354 中的参数化数据永久保持且不需要电池。必须在联机模式下，将数据传送到 FM354 模块中，有两种方法可

以进入到联机模式。

1）将 MPI 电缆插在 CPU 的 MPI 或使用其他网络编程接口（如 PROFIBUS、以太网接口）保持编程器与 PLC 站处于联机状态，在硬件配置界面中点击 FM354 模块的“Parameter…”按钮，进入参数化界面时首先进入联机模式。

2）如果离线进入 FM354 模块的参数化界面，确认编程器与 PLC 站处于联机状态后，在菜单“Target system”中，选择“communication”命令，进入联机模式。

在联机模式中，可以选择“Transfer date to FM”按钮，将配置的系统数据传送到 FM354 模块中，如图 11-51 所示。

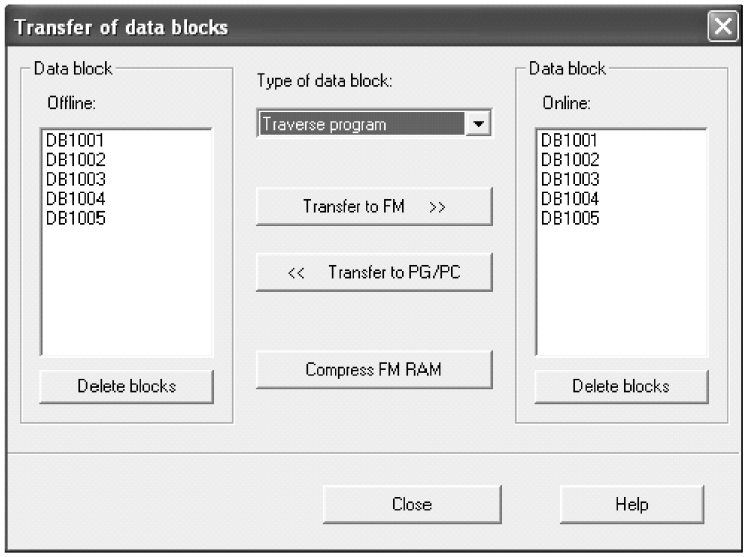



图 11-51 FM354 模块数据传送界面

在“Offline”栏中，为离线配置的数据块，在“Type of date block”下拉菜单中，可以选择需要显示的系统数据类型，如选择显示“Traverse program”，则在数据块中只显示与位移程序块相关的数据块 DB1001 ~ DB1199。选择离线数据块，点击“Transfer to FM”，将配置的参数下载到 FM 模块中，同样也可以将 FM354 模块中存储的数据块上传到编程器中，进行比较和编辑。

### 11.4.9 FM354 测试功能

1. 启动界面

在 CPU 处于“STOP”模式或在用户数据块中置位 DBX14.1，利用 FM354 模块集成的测试功能，可以对各种操作模式进行模拟和联机调试。在测试模式下，FM354 模块不受 CPU 程序控制，所有的操作必须在“Start up”界面中手动完成，如选择操作模式、设定位置值和速度值等参数，在参数化界面中，点击“Start up”按钮，进入启动界面如图 11-52 所示。

在启动界面中首先选择需要的操作模式，如“JOG”、“Open loop”、“MDI”等模式，“JOG”和“Open loop”模式不需要回零操作。选择启动控制的先决条件“Controller enable”、

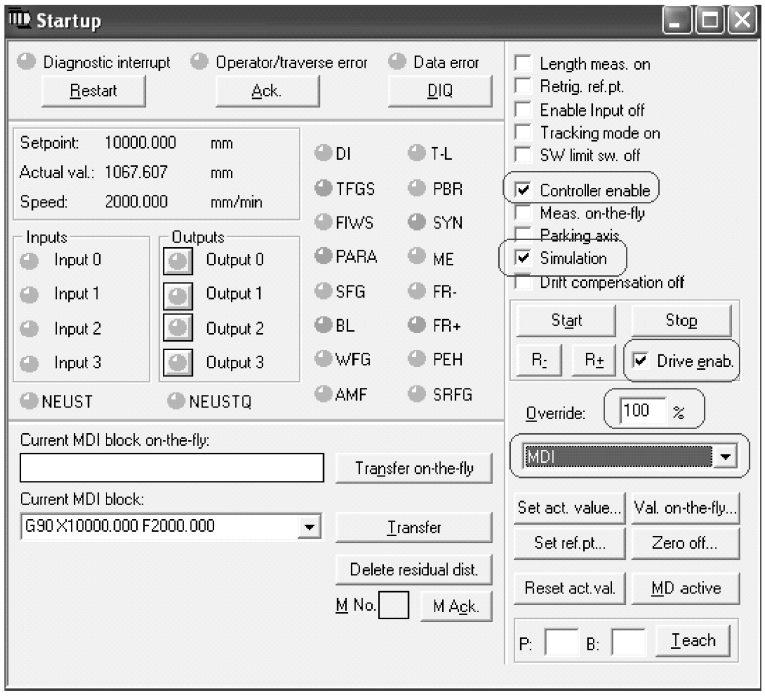



图 11-52　FM354 模块测试界面

“Drive enable”并设置速度倍率“Override”（设定速度的百分比），在左边的选项中选择速度值和电压值，点击“R +”、“R –”键后，必须按空格键才能起动电动机。在测试界面中，可以监控位置设定值、速度值和实际位置值等；也可以监控到其他位移触发的信号，如“FR +”正方向移动、“BL”正在位移信号、“PEH”定位完成信号等；如果在 MD 中对输入、输出信号进行配置，在输入、输出栏中显示相应配置信息的状态。

如果 FM354 模块没有连接外围设备，如驱动器、编码器等同样可以利用模拟的方式对配置的参数进行调试，点击选项“Simulation”进入模拟方式，与联机调试方式相比较，在模拟的方式下，FM354 模块的运算结果不会输出到驱动器接口（FM354 模拟量接口不输出），位置的实际值不是通过采集编码器信号而是通过内部计算得到。

如果需要测试“MDI”、“AUTOMATIC”等控制模式，则需要轴同步信号（SYN 指示灯必须为绿色），否则轴不能运行并报错。有两种方法可以得到轴同步信号：1）选择寻找参考点模式，触发参考点开关得到轴同步信号；2）在“JOG”或其他可以选择设置参考点的模式下，点击“Set ref pt”按钮，将当前位置设置为参考点并得到轴同步信号。如图 11-52 所示，在设置参考点后使能“SYN”信号，然后选择“MDI”模式，在“Current MDI block”中，手动输入位移块“G90 X10000. 000 F2000. 000”，点击“Start”键，启动位移指令，轴将以 2000mm/min 的速度位移到绝对地址 10000mm 处。

在测试界面中，可以对所有控制模式进行测试，除此之外还可以对 M 功能、示教功能（在自动程序中往往不能确定需要行走的位移距离，可以预先设定一个值，在调试过程中得到需要的位置值时，在参数“P”中选择自动程序号，在参数“B”中选择自动程序中位移块号，点击“Teach”键，将当前的位置值赋值到指定的位移块中，替代原有位移值）、测

量功能等进行测试。

在调试过程中，如果需要修改 MD 参数，并修改完成传送到模块中，则需要在测试界面中，通过点击“MD active”键或模块掉电重启动，使新的 MD 参数生效。

2. 故障诊断界面

如果在调试过程或程序执行中，由于操作引起故障，可以点击“Error display”按钮，进入故障诊断界面查看引起故障的原因，如图 11-53 所示。

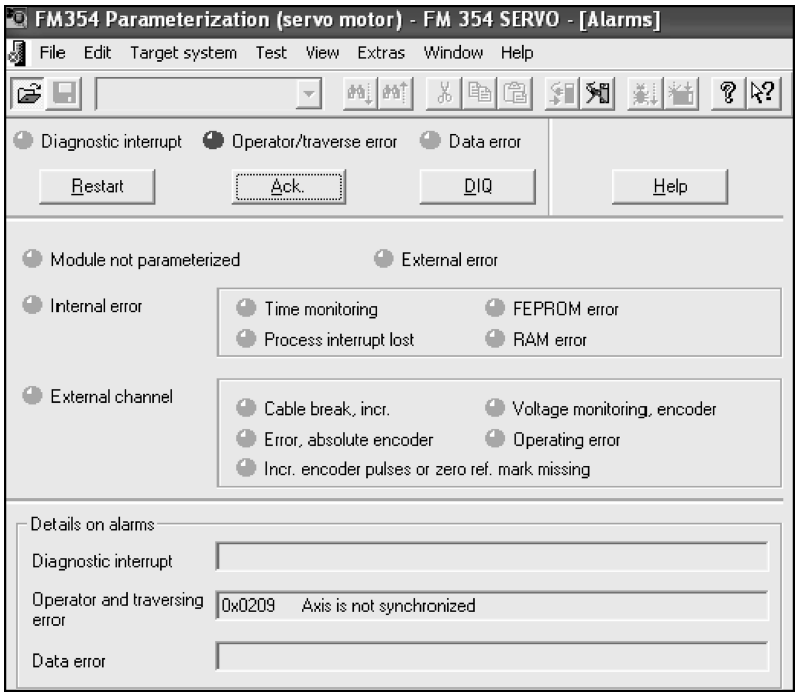



图 11-53 FM354 模块故障诊断界面

将故障划分为 3 个类别，分别为诊断中断、操作/位移故障、数据故障。不同类别的故障需要不同的复位方式，如重新启动、确认等，如图 11-53，由于没有轴同步信号而启动“MDI”操作，从而引起“Axis is not synchronized”故障，指示为操作/位移类型故障，需要通过确认键“Ack”，消除故障报警信号，如果故障类型为诊断中断，则需要重新启动模块，消除故障报警信号。

3. FM354 模块运行数据界面

FM354 模块的运行数据界面如图 11-54 所示。

在运行数据界面中，可以对 DAC 模拟量的输出值、编码器的实际计数值、跟随偏差等参数进行监控，例如通过监控运行时的跟随偏差，适当调整控制器的增益值以满足工艺要求。

4. FM354 模块位置跟踪界面

点击“Trace”按钮，进入 FM354 模块的位置跟踪界面，如图 11-55 所示。

在位置跟踪界面中，可以监控位置的设定值与实际值在运行过程中的偏差及位置控制的效果。通常位置跟踪是对位置控制的监控，速度环的控制则需要通过驱动器的专用软件，如

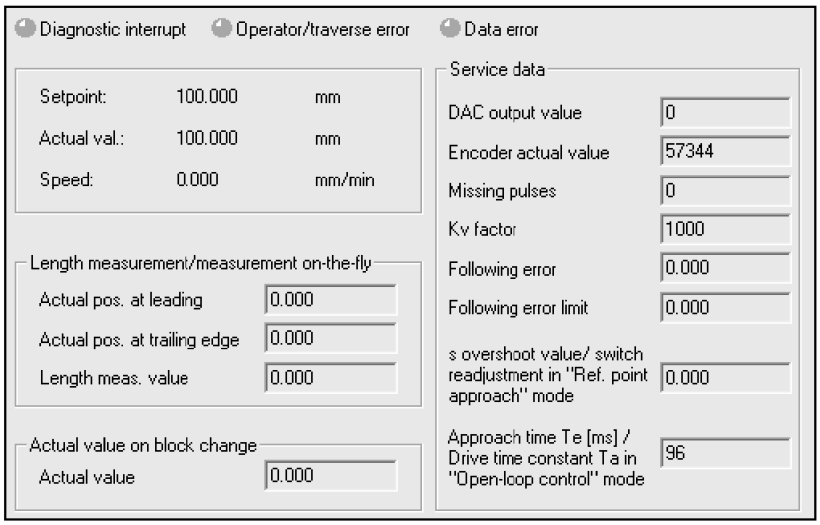



图 11-54　FM354 模块运行数据界面

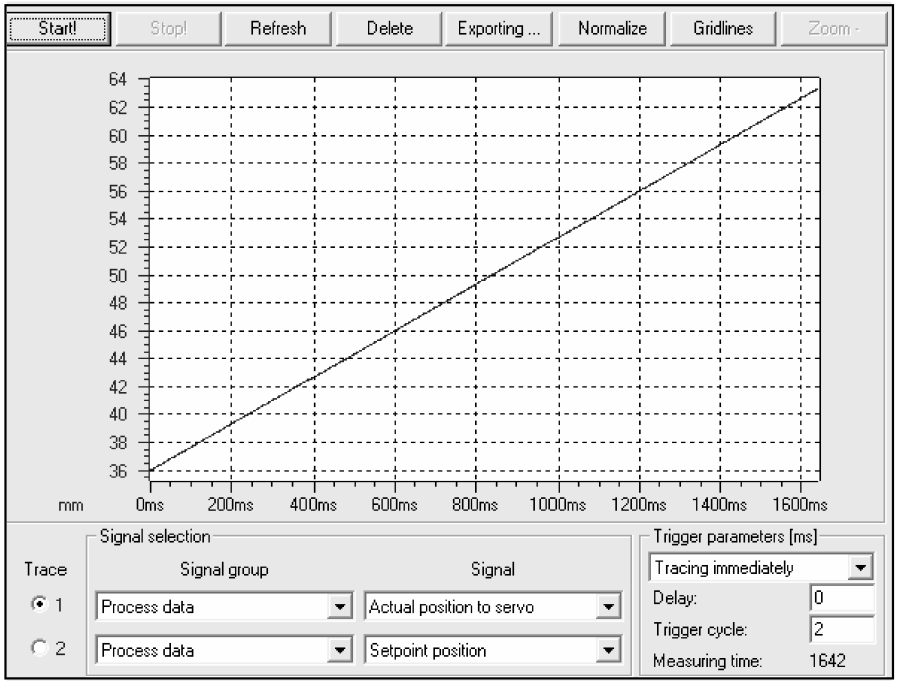



图 11-55　FM354 模块位置跟踪界面

通过 SimoComU 软件对 SIMODRIVE 进行设置，根据工艺的要求设定相应的速度曲线。

### 11.4.10　FM354 系统数据生成 SDB 文件

将生成的系统数据传送到 FM354 模块中，启动模块运行，如果模块故障需要更换，则需要将生成的系统数据块再次下传到新的 FM354 模块中，现场维护人员可能因为没有编程

器而不能进行模块的更换，FM354 模块系统数据生成 SDB 文件的功能将生成的系统数据存储于 CPU 中，更换 FM354 模块后，只需要 CPU 再次接电即可将存储于 CPU 中的系统数据传送到模块中，不需要重新下载。在参数界面中，点击“Create SDB”按钮，将模块系统数据块存储于 CPU 中（在 PLC 站 FM354 模块文件夹中，可以看到用于 FM354 模块的系统数据块），操作必须在联机模式下完成。

**建议：**

在调试中，可能需要多次修改系统参数并下载到 FM354 中，如果没有即时生成 SDB，再次接电后修改的参数可能被存储于 CPU 中的 SDB 覆盖，建议调试完成后一次性生成系统数据块。

### 11.4.11 进入 FM354 模块地址

在参数化界面中，点击“Enter Mod-Adr in user DB”按钮，在选择项目中生成的用户数据块，将 FM354 模块的逻辑地址赋值到数据块中，否则需要在数据块中编程赋值。

### 11.4.12 FM354 模块的编程

1. 初始化程序

在启动界面中，如图 11-52 所示可以手动操作 FM354，实现不同操作模式的控制。如果 CPU 在运行模式并且用户数据块中 DBX14.1 不为 1 时，则 FM354 的操作将被 CPU 接替。与手动测试操作方式相比，惟一不同的是手动操作时，所有操作和赋值是通过鼠标和键盘完成的，例如选择控制器使能、点击启动按钮等；使用 CPU 控制操作时，原先的手动操作通过在用户数据块中的赋值完成，并将不同的控制过程连接起来实现自动控制。下面介绍通过用户程序控制 FM354 模块的方法。

在“SIMATIC Manager”界面中，使用菜单命令“File”→“Open”，打开“Libraries”函数库并选择“FMSTSV_L”目录，将目录下的函数 FC0 ~ FC3、UDT1 复制到当前的项目中，以 UDT1 为数据格式创建用户数据块，如 DB1，如图 11-56 所示。

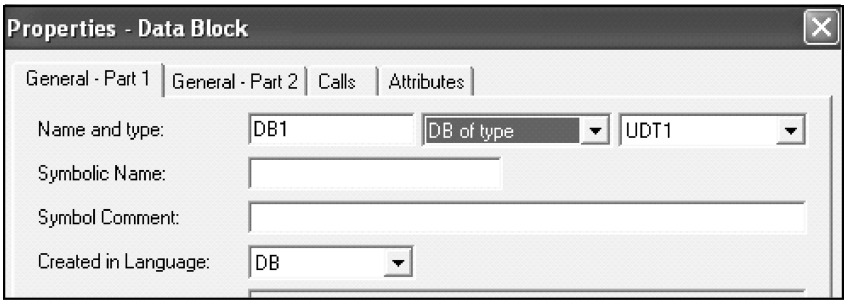



图 11-56 生成用户数据块

用户数据块是用于 CPU 与 FM354 进行数据交换的数据区，CPU 通过用户数据块将命令发送到 FM 模块中，FM 模块的反馈信息同样存储于用户数据块中。用户数据块的内容和通信格式已经在 UDT1 中定义，不需要用户创建。

首先在 OB100 中调用 FC0，指定创建的数据块，如 DB1 作为 CPU 与 FM 模块的通信接

口区，示例程序如下：

```
CALL    "POS_INIT"
        DB_NO:=1
        CH_NO:=1
        LADDR:=256
        RET_VAL:=MW2
```

FC0 参数含义如下：

DB_NO　：指定一个数据块作为通信接口区；

CH_NO　：通道号，FM354 只有一个通信；

LADDR　：FM354 的逻辑地址；

RET_VAL：调用函数的返回值。

在 OB1 中调用 FC1，建立与 FM354 的通信关系，示例程序如下：

```
CALL    "POS_CTRL"
        DB_NO:=1
        RET_VAL:=MW4
```

FC1 参数含义如下：

DB_NO　：指定一个数据块作为通信接口区，与 FC0 指定的数据块必须相同；

RET_VAL：调用函数的返回值。

FC2、FC3 为诊断和测量功能函数，在这里不做介绍。调用 FC1 后，CPU 通过通信接口区将命令发送到 FM 模块，并读出返回值，例如在手动测试界面中，选择启动控制的先决条件“Controller enable”、“Drive enable”和设置的速度倍率“Override”（设定速度的百分比），在 CPU 运行时，可以通过程序实现，示例程序如下：

```
SET
=       DB1.DBX     15.7    //驱动使能信号
=       DB1.DBX     34.0    //伺服使能信号
L       100
T       DB1.DBB     18      //设定倍率为 100%
```

用户数据块中 DB1.DBX15.7 为“Drive enable”信号，DB1.DBX34.0 为“Controller enable”信号，速度倍率通过 DB1.DBB18 设置。设置完成后，选择操作模式进行位置控制。通过编程实现的功能和操作次序与在手动测试界面中实现的功能和操作完全相同，可以借鉴手动操作实现程序控制。

2. 点动模式控制程序

选择点动操作模式的示例程序如下：

```
L       1
T       DB1.DBB     16      //选择模式 JOG
L       1
T       DB1.DBB     17      //选择速度级别 1
L       L#60000
T       DB1.DBD     160     //设定速度级别 1 的速度
```

```
L     L#800
T     DB1. DBD    164    //设定速度级别 2 的速度
A     M           20. 1
S     DB1. DBX    38. 0  //将设定的速度传送到 FM354 中
A     DB1. DBX    48. 0  //模块确认信号
R     M           20. 1
R     DB1. DBX    48. 0
A     M           20. 2
=     DB1. DBX    15. 2  //反向位移命令
A     M           20. 3
=     DB1. DBX    15. 3  //正向位移命令
```

点动操作模式代码为 1，将 1 赋值到 DB1. DBB16 中选择点动操作模式，赋值 DB1. DBB17 选择速度级别（只有两个速度级别可选），通过赋值 DB1. DBD160、DB1. DBD164 设定两个级别的速度值，赋值完成后通过置位 DB1. DBX38. 0（M20. 1）将选择的设定发送到 FM354 模块中，M20. 2 为 1 时电动机反转，为 0 时停止；M20. 3 为 1 时电动机正转，为 0 时停止。

3. 开环模式控制程序

在开环模式中，直接输出模拟量电压值控制轴的速度而与位置控制无关，模拟量输出的大小与电动机速度关系在 MD 参数中设置，例如电动机的最大转速为 3000r/min 与输出电压最大值 8V 相对应。选择开环操作模式的示例程序如下：

```
L     2
T     DB1. DBB    16     //选择开环控制模式
L     1
T     DB1. DBB    17     //选择输出电压级别 1
L     L#4000
T     DB1. DBD    168    //设定电压级别 1 的输出电压值（mV）
L     L#5000
T     DB1. DBD    172    //设定电压级别 2 的输出电压值（mV）
A     M           30. 1
S     DB1. DBX    38. 1  //将设定的命令传送到 FM354 中
A     DB1. DBX    48. 1  //模块确认信号
R     M           30. 1
R     DB1. DBX    48. 1
A     M           30. 2
=     DB1. DBX    15. 2  //轴反转命令
A     M           30. 3
=     DB1. DBX    15. 3  //轴正转命令
```

开环操作模式代码为 2，将 2 赋值到 DB1. DBB16 中选择开环操作模式，赋值 DB1. DBB17 选择电压级别（只有两个电压级别可选），通过赋值 DB1. DBD168、

DB1. DBD172 分别设定两个级别的电压值 4V 和 5V，赋值完成后通过置位 DB1. DBX38. 1（M30. 1）将选择的设定发送到 FM354 模块中，M30. 2 为 1 时电动机反转，为 0 时停止；M30. 3 为 1 时电动机正转，为 0 时停止。

4. 接近参考点（回零）模式控制程序

接近参考点模式是通过外围的输入信号作为参考点开关，在位移的坐标轴上找到位置的基准点，如零点，回零点操作完成后，使能轴同步信号为 DB1. DBX25. 0（SYNC），轴同步信号是增量模式、MDI 和自动模式操作的先决条件，选择接近参考点操作模式的示例程序如下：

```
L     3
T     DB1. DBB    16     //选择回零控制模式
A     M           40. 0  //Start 命令
S     DB1. DBX    15. 0
R     M           40. 0  //复位 Start 命令
A     DB1. DBX    23. 1  //运行信号
FP    M           40. 4
R     DB1. DBX    15. 0
A     M           40. 1  //Stop 命令
=     DB1. DBX    15. 1
```

回零操作模式代码为 3，将 3 赋值到 DB1. DBB16 中，选择回零操作模式，通过 M40. 0 启动回零操作，启动信号为脉冲信号，通过 M40. 1 可以终止回零操作。

5. 增量模式控制程序

启动增量模式前，必须得到轴同步信号 DB1. DBX25. 0（SYNC），有两种方式可以得到轴同步信号。①执行回零操作；②设定当前值作为参考点。增量模式有两种方式：①选择增量表中的值作为相对移动的位置值；②设定任意的位置值作为相对移动的位置值。以速度级别 1 作为增量模式的速度值，选择增量操作模式的示例程序如下：

```
L     4
T     DB1. DBB    16     //选择增量控制模式
L     L#100000
T     DB1. DBD    160    //设定速度级别 1 的速度
L     L#800
T     DB1. DBD    164    //设定速度级别 2 的速度
A     M           50. 5
=     DB1. DBX    38. 0  //将设定的速度传送到 FM354 中
A     DB1. DBX    48. 0  //反馈信息
R     M           50. 5
A     M           50. 6  //选择增量表位置值模式
JCN   M1
L     1                  //选择增量表第一行设置的位置值
T     DB1. DBB    17
M1：A  M          50. 7  //选择随机位置值模式
```

```
      JCN     M2
      L       254                 //选择增量表 254 行
      T       DB1. DBB    17
      L       L#100000
      T       DB1. DBD    156     //随机位置值
      S       DB1. DBX    38. 2   //将随机位置值传送到 FM354 中
      A       DB1. DBX    48. 2   //命令确认信号
      FP      M           50. 4
      R       M           50. 7
      R       DB1. DBX    48. 2
M2：  A       M           50. 2   //反向位置增量命令
      =       DB1. DBX    15. 2
      A       M           50. 3   //正向位置增量命令
      =       DB1. DBX    15. 3
      A       M           50. 1   //Stop 命令
      =       DB1. DBX    15. 1
```

增量操作模式代码为 4，将 4 赋值到 DB1. DBB16 中选择增量操作模式，以速度级别 1 作为增量模式的速度值，速度级别 1 的设定值为 10000（MD 中定义的测量单位），通过 M50. 5 将设定的速度值传送到 FM354 模块中。当 M50. 6 为 1 时，选择在增量表中定义的位置值作为相对位置设定点，在程序中将 1 赋值到 DB1. DBB17 中，指定增量表第一行的数值作为位置值；当 M50. 7 为 1 时，选择随机的位置值作为相对位置设定点，将随机的位置值赋值到 DB1. DBD156 中，通过使能 DB1. DBX38. 2，将随机位置传送到 FM354 模块中，M50. 6 和 M50. 7 不能同时为 1。通过 M50. 2 和 M50. 3（脉冲信号）控制位移的方向。

6. MDI 模式控制程序

与增量模式相同，启动 MDI（手动数据输入）控制模式前必须得到轴同步信号 DB1. DBX25. 0（SYNC）。在 MDI 操作模式中预先设定位移的方式（相对位移和绝对位移）、速度和位置值，点击启动命令执行定位操作。选择 MDI 操作模式的示例程序如下：

```
      L       6
      T       DB1. DBB    16      //选择 MDI 模式
      A       M           60. 2
      S       DB1. DBX    178. 0  //使能 G 功能编组 1
      L       90
      T       DB1. DBB    180     //赋值 G90 功能
      S       DB1. DBX    178. 4  //使能位置设定
      L       L#50000
      T       DB1. DBD    184     //赋值位置值 50000
      S       DB1. DBX    179. 0  //使能速度设定
      L       L#100000
      T       DB1. DBD    188     //赋值速度值 100000
```

```
S       DB1. DBX    38. 3    //使能 MDI 位移块
A       DB1. DBX    48. 3
R       M           60. 2
R       DB1. DBX    48. 3
A       DB1. DBX    22. 4    //是否有故障
JC      END
A       M           60. 0    //启动信号
S       DB1. DBX    15. 0
R       M           60. 0    //复位启动信号
A       DB1. DBX    23. 1
FP      M           60. 4
R       DB1. DBX    15. 0
A       M           60. 1    //停止信号
=       DB1. DBX    15. 1
END：   NOP         0
```

MDI 操作模式代码为 6，将 6 赋值到 DB1. DBB16 中选择 MDI 操作模式，通过 M60. 2 分别使能 G1 功能开关 DB1. DBX178. 0、位置值开关 DB1. DBX178. 4 和速度值开关 DB1. DBX179. 0，然后分别赋值 DB1. DBB180 选择 G1 组 G 功能代码、DB1. DBD184 选择位置值和 DB1. DBD188 选择速度值，程序中实际执行的位移块为 G90 X5000 F100000，通过使能 DB1. DBX38. 3 将位移块传送到 FM354 模块中，模块确认后复位该信号。通过 M60. 0 启动 MDI 定位操作，启动信号为脉冲信号，通过 M60. 1 终止 MDI 操作。

7. 自动模式控制程序

自动位移程序必须已经编辑完成，并且存储于 FM354 模块中，通过 PLC 的程序选择自动位移程序，并执行定位功能，与增量模式相同，启动自动控制模式前必须得到轴同步信号 DB1. DBX25. 0（SYNC）。选择自动操作模式的示例程序如下：

```
L       8
T       DB1. DBB    16       //选择自动模式
A       M           70. 5
=       DB1. DBX    15. 5    //使能读下一条位移块
L       B#16#1               //选择位移块 1
T       DB1. DBB    242      //将选择的位移块赋值到 DB 中
FP      M           70. 6
S       DB1. DBX    39. 5    //使能自动位移程序
A       DB1. DBX    22. 4    //是否有故障
JC      M1
A       M           70. 0    //启动信号
S       DB1. DBX    15. 0
R       M           70. 0    //复位启动信号
A       DB1. DBX    23. 1    //定位执行中信号
```

```
      FP        M         70.4
      R         DB1.DBX   15.0
      A         M         70.1    //停止信号
      =         DB1.DBX   15.1
M1：NOP         0
```

自动操作模式代码为 8，将 8 赋值到 DB1.DBB16 中选择自动操作模式，使能 Read_In 功能（DB1.DBX15.5）可以连续执行自动程序，这个条件是必须的，在 DB1.DBB242 选择存储于 FM354 模块中的程序，程序中选择的自动程序号为 1，通过使能 DB1.DBX39.5 发送程序选择命令，FM354 模块确认后复位该信号，通过 M70.0 启动自动程序执行，启动信号为脉冲信号，通过 M70.1 可以终止自动程序的执行。

**注意：**

示例程序为基本编程操作，在实际的编程中需要读出操作的反馈信号以确认发送的命令是否传送到 FM354 模块中并且没有错误，如模式选择的结果可以在 DB1.DBB24 中读出。

**小窍门：**

CPU 运行时，FM354 模块的测试界面只能监控不能操作，可以在测试界面中查看程序中设定值（包括操作模式选择以及设定的参数）是否正确。

FM353 模块与 FM354 模块的参数化相似（除步进电动机参数以外），控制模式和程序完全相同，FM353 的应用可以参考 FM354 的示例程序。FM354 的示例程序 FM354.zip、参数化软件及电子手册参考光盘“功能模块 FM”目录。

## 11.5 FM355 PID 控制模块

### 11.5.1 应用概述

FM355 PID 闭环控制器适合压力、流量、温度的闭环控制。在西门子公司的软件中有多种 PID 控制器，如集成于 STEP7 的 FB41、FB42、FB43 PID 控制函数；参数过程图形化，实现手动、自动无扰切换功能的标准 PID 控制函数（需要额外购买）；集成更多的 PID 控制算法、将不同的控制算法或 PID 组件搭接为一个完整的 PID 控制回路的模块化 PID 控制函数（需要额外购买）等都是软件 PID 函数。PID 控制算法在 CPU 内进行，输入和输出接口可以直接指定不同的信号模块，对于编程人员来说，有很高的灵活性，但是在控制回路较多的情况下将占用大量的 CPU 循环扫描时间，从而影响 CPU 快速处理能力。与软件 PID 控制函数相比，FM355 模块集成控制算法，所有 PID 控制的计算和信号采样由模块完成，不占用 CPU 的扫描时间，CPU 通过专用函数与 FM355 模块进行数据交换—发送命令并接收反馈信号。除此之外，FM355 模块还集成后援操作模式，即在 CPU 故障停止的情况下，将当前设定切换到在模块中预先定义的设定值进行 PID 闭环控制，如果 CPU 连接操作面板，在后援操作模式下通过操作面板可以直接对 FM355 模块进行操作，避免 CPU 停机造成的过程控制失控。

### 11.5.2 硬件安装与接线

FM355 模块可以安装于 S7-300 系列 PLC 的中央机架，也可以安装于分步式 I/O ET200M

中。在分步式 I/O ET200M 中安装 FM355 模块时，由于 CPU 需要调用系统函数对 FM355 模块进行访问，同时调用系统函数的个数受到 CPU 资源的限制，对于 S7-300 系列 PLC CPU，在 ET200M 中最多可以插入一个模块（4 个 PID 回路）；对于 S7-400 系列 PLC CPU，在 ET200M 中最多可以插入两个模块（8 个 PID 回路）。通常在 S7-300 系列 PLC 中央机架可以连接 8 个模块，具体可使用的数量查看 CPU 相关的技术规范。FM355 模块有两种类型：1）FM355C，反馈信号和控制输出同为模拟量，对过程进行连续控制；2）FM355S，反馈信号为模拟量，控制输出为开关量，适合控制具有开关量输入接口的过程控制器，如使用晶闸管的温度控制器等。FM355C 的输入、输出接口分别使用两个 20 针的前连接器，分为左右两端，端子接线见表 11-14。

**表 11-14　FM355C 端子接线**

| 左端前连接器 | | | | 右端前连接器 | | | |
|---|---|---|---|---|---|---|---|
| 连接 | 模拟输入 | 名称 | 功　能 | 连接 | 模拟输出 | 名称 | 功　能 |
| 1 | — | — | — | 1 | — | L + | DC 24V 电源正 |
| 2 | 1 | $IC_+$ | 恒流源（正极） | 2 | — | I1 | 数字输入 |
| 3 | | $IC_-$ | 恒流源（负极） | 3 | — | I2 | 数字输入 |
| 4 | | M + | 测量端（正极） | 4 | — | I3 | 数字输入 |
| 5 | | M – | 测量端（负极） | 5 | — | I4 | 数字输入 |
| 6 | 2 | $IC_+$ | 恒流源（正极） | 6 | — | I5 | 数字输入 |
| 7 | | $IC_-$ | 恒流源（负极） | 7 | — | I6 | 数字输入 |
| 8 | | M + | 测量端（正极） | 8 | — | I7 | 数字输入 |
| 9 | | M – | 测量端（负极） | 9 | — | I8 | 数字输入 |
| 10 | — | COMP + | 补偿端（正极） | 10 | — | — | — |
| 11 | — | COMP – | 补偿端（负极） | 11 | 1 | Q1 | 模拟量输出 |
| 12 | 3 | $IC_+$ | 恒流源（正极） | 12 | | $M_{ANA}$ | 模拟输出参考点 |
| 13 | | $IC_-$ | 恒流源（负极） | 13 | 2 | Q2 | 模拟量输出 |
| 14 | | M + | 测量端（正极） | 14 | | $M_{ANA}$ | 模拟输出参考点 |
| 15 | | M – | 测量端（负极） | 15 | 3 | Q3 | 模拟量输出 |
| 16 | 4 | $IC_+$ | 恒流源（正极） | 16 | | $M_{ANA}$ | 模拟输出参考点 |
| 17 | | $IC_-$ | 恒流源（负极） | 17 | 4 | Q4 | 模拟量输出 |
| 18 | | M + | 测量端（正极） | 18 | | $M_{ANA}$ | 模拟输出参考点 |
| 19 | | M – | 测量端（负极） | 19 | — | — | — |
| 20 | — | $M_{ANA}$ | 模拟输入参考点 | 20 | — | M | DC 24V 电源负 |

FM355C 最多可以连接 4 个模拟量输入信号，作为实际值反馈、4 个模拟量输出连接执行机构和 8 个用于 FM355C 参数选择的数字量输入信号。FM355S 的输入、输出接口同样使用两个 20 针的前连接器，分为左右两端，端子接线见表 11-15。

**表 11-15　FM355S 端子接线**

| 左端前连接器 | | | | 右端前连接器 | | | |
|---|---|---|---|---|---|---|---|
| 连接 | 模拟输入 | 名称 | 功　　能 | 连接 | 控制通道 | 名称 | 功　　能 |
| 1 | — | — | | 1 | — | L + | DC 24V 电源正 |
| 2 | 1 | $IC_+$ | 恒流源（正极） | 2 | — | I1 | 数字输入 |
| 3 | | $IC_-$ | 恒流源（负极） | 3 | — | I2 | 数字输入 |
| 4 | | M + | 测量端（正极） | 4 | — | I3 | 数字输入 |
| 5 | | M – | 测量端（负极） | 5 | — | I4 | 数字输入 |
| 6 | 2 | $IC_+$ | 恒流源（正极） | 6 | — | I5 | 数字输入 |
| 7 | | $IC_-$ | 恒流源（负极） | 7 | — | I6 | 数字输入 |
| 8 | | M + | 测量端（正极） | 8 | — | I7 | 数字输入 |
| 9 | | M – | 测量端（负极） | 9 | — | I8 | 数字输入 |
| 10 | — | COMP + | 补偿端（正极） | 10 | — | — | — |
| 11 | — | COMP – | 补偿端（负极） | 11 | 1 | Q1 | 数字输出步进控制器：输出上升信号脉冲控制器：输出 A |
| 12 | 3 | $IC_+$ | 恒流源（正极） | 12 | | Q2 | 数字输出步进控制器：输出下降信号脉冲控制器：输出 B |
| 13 | | $IC_-$ | 恒流源（负极） | 13 | 2 | Q3 | 数字输出步进控制器：输出上升信号脉冲控制器：输出 A |
| 14 | | M + | 测量端（正极） | 14 | 3 | Q4 | 数字输出步进控制器：输出下降信号脉冲控制器：输出 B |
| 15 | | M – | 测量端（负极） | 15 | | Q5 | 数字输出步进控制器：输出上升信号脉冲控制器：输出 A |
| 16 | 4 | $IC_+$ | 恒流源（正极） | 16 | | Q6 | 数字输出步进控制器：输出下降信号脉冲控制器：输出 B |
| 17 | | $IC_-$ | 恒流源（负极） | 17 | 4 | Q7 | 数字输出步进控制器：输出上升信号脉冲控制器：输出 A |
| 18 | | M + | 测量端（正极） | 18 | | Q8 | 数字输出步进控制器：输出下降信号脉冲控制器：输出 B |
| 19 | | M – | 测量端（负极） | 19 | — | — | — |
| 20 | — | $M_{ANA}$ | 模拟输入参考点 | 20 | — | M | DC 24V 电源负 |

FM355S 最多可以连接 4 个模拟量输入信号作为实际值反馈，8 个数字量输出连接执行机构（每个回路最多为两个输出，如温度控制中，一个作为加热输出，另一个作为冷却输出）和 8 个数字量输入信号用于 FM355S 的参数选择。

电压、电流、热电耦、热电阻信号可以直接连接到 FM355 模块作为反馈信号，电压信号的接线如图 11-57 所示。

M + 、M – 为每个通道的输入端，信号负端连接第 20 引脚 $M_{ANA}$（接地端），保证信号与

测量端没有电位差。如果连接非隔离传感器，则需要传感器端和 PLC 端同时接地，并保证同一个地。

连接四线电流（传感器电源与信号线分开）传感器的接线方式，如图 11-58 所示。

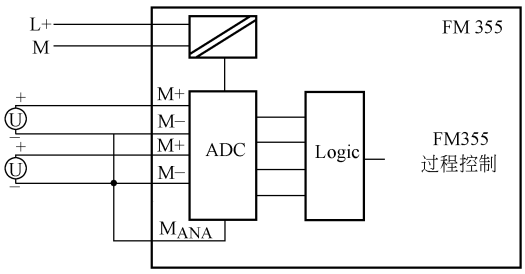



图 11-57　连接电压传感器

M + 、M – 为每个通道的输入端，信号负端连接第 20 引脚 $M_{ANA}$（接地端）保证信号与测量端没有电位差。如果连接非隔离传感器，则需要传感器端和 PLC 端同时接地，并保证同一个地。在输入端需要并联电阻，连接的电阻随模块包装携带。

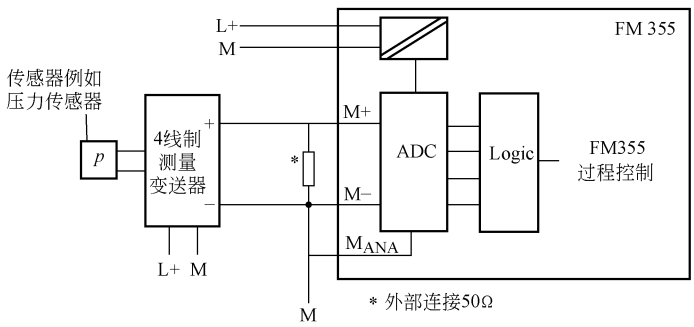



图 11-58　连接四线电流传感器

连接两线电流传感器（传感器电源与信号线共用，传感器没有电源）的接线方式如图 11-59 所示。

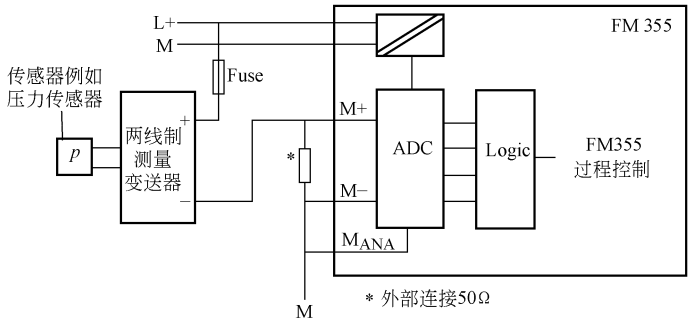



图 11-59　连接两线电流传感器

M + 、M – 为每个通道的输入端，信号负端连接第 20 引脚 $M_{ANA}$（接地端），保证信号与测量端没有电位差。在输入端需要并联电阻，连接的电阻随模块包装携带。不能连接非隔离传感器。

连接热电阻信号的接线方式，如图 11-60 所示。

M + 、M – 为每个通道的输入端，$I_{C+}$、$I_{C-}$为恒流源，FM355 模块发出恒流源信号，将

热电阻产生的电压信号作为输入信号。

连接热电耦信号接线如图 11-61 所示。

M + 、M – 为每个通道的输入端，通常热电耦信号需要冷端补偿，FM355 模块使用热电阻将冷端的温度采集到模块中动态补偿热电耦信号的偏差值。外部补偿利用 FM355 的第三个通道中 $I_{C+}$、$I_{C-}$ 信号作为恒流源，COMP + 、COMP – 作为信号输入端，如果选择为外部补偿，第三个通道不

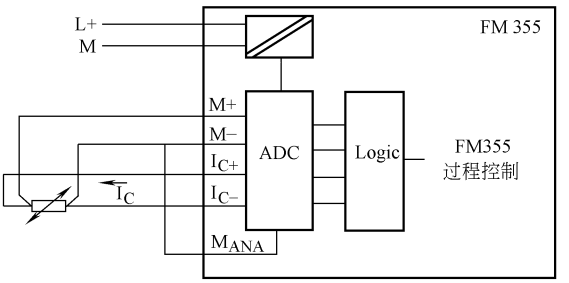



图 11-60　连接热电阻传感器

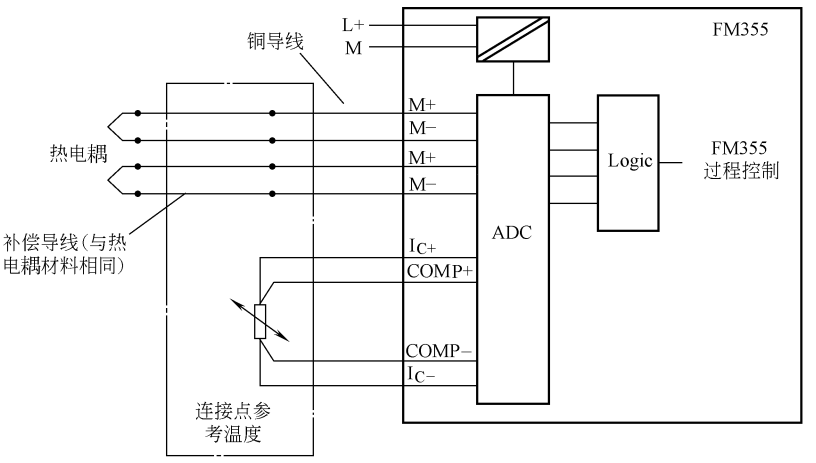



图 11-61　连接热电耦传感器–外部补偿

能再连接热电阻信号，只能连接其他两线输入信号。在模块参数设定中，也可以选择热电耦冷端补偿方式为内部补偿，即补偿值为一固定的温度值，不能变化，内部温度补偿的接线方式如图 11-62 所示。

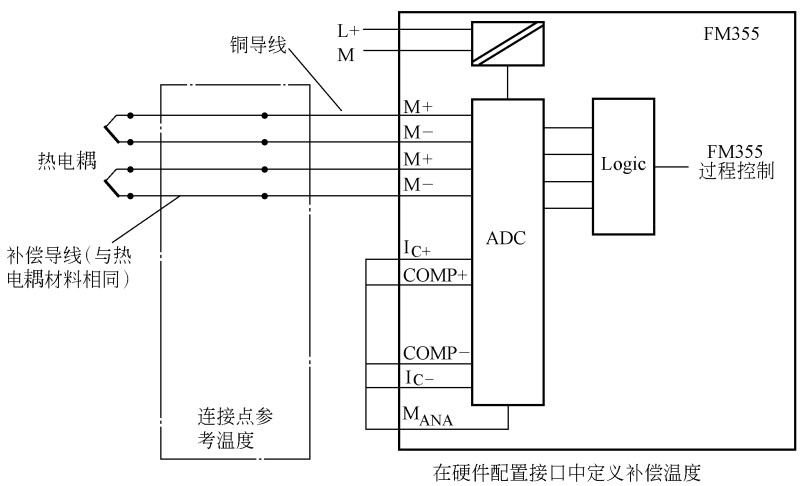



图 11-62　连接热电耦传感器–内部补偿

其他信号的连接这里不作介绍。

### 11.5.3　系统配置及参数设置

FM355 的参数化软件光盘随硬件包装携带，参数化软件以 STEP7 为平台，直接安装后，在“SIMATIC Manager”硬件配置中可以对 FM355 模块进行参数化。以 FM355C 为例，介绍参数的设置。

打开“SIMATIC Manager”硬件配置界面，双击 FM355C 模块，点击“parameter…”按钮，进入参数化界面，如图 11-63 所示。

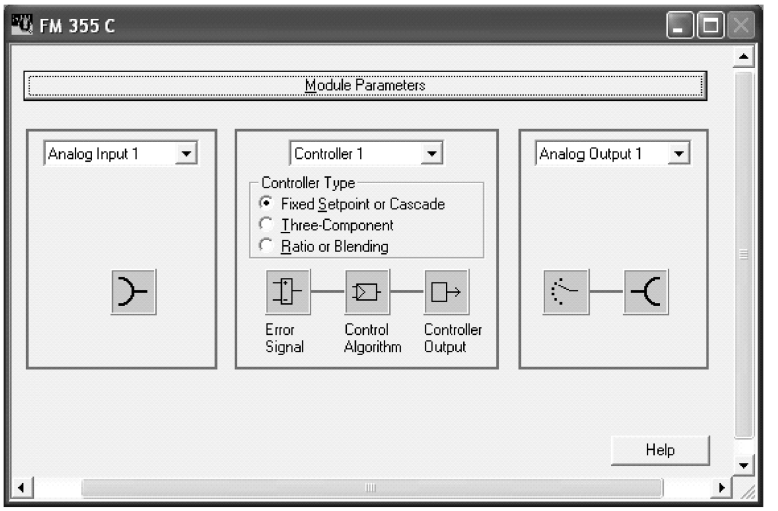



图 11-63　FM355C 模块参数化界面

在“Module Parameters”对话框中，可以对 FM355C 模块集成的数字量输入信号进行参数化，如选择信号为 1 的条件（高电平或低电平）；也可以选择温度的单位、设定热电耦内部补偿的温度值等。从图 11-63 中可以看到一个 PID 控制回路，包括过程值采样回路、控制器回路和输出回路 3 个子回路，数据的流向也是从左向右经过过程值采样，然后与设定值相减算出偏差值，再经过 PID 运算，将计算结果通过输出信号传送到实际的控制器上。默认条件下相同序号的模拟量输入、控制器和模拟量输出为一个 PID 回路。

首先选择需要参数化的模拟量输入通道，然后点击按钮，进入参数化界面如图 11-64 所示。

图 11-64 中参数的功能如下：

“A/D”　：选择模拟量数模转换的分辨率，分辨率高，则采样时间长；

“Sensor”　：选择模拟量输入连接的传感器类型；

“Filter”　：设定模拟量滤波时间，抑制高频信号对输入信号的干扰，默认状态为关；

“Square Root”：对输入信号开平方；

“Polyline”　：对非线性的传感器信号进行线性差补，如在“Sensor”对话框中，选择“Free thermocouple type”类型传感器时，需要进行差补运算。选择确定类型的传感器已经对输入信号的特征曲线进行处理，不需要差补运算；

“Normalize” ：对输入信号进行线性化。

如果使用外部温度补偿，则需要选择“Reference Input”；如果使用内部温度补偿，则需要选择“Parameters”。

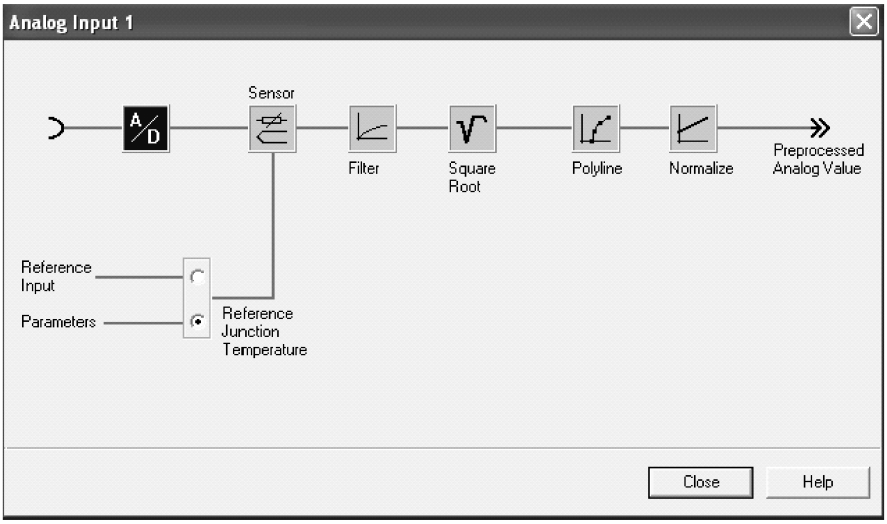



图 11-64 FM355C 模块过程值参数化界面

过程值信号参数化完成后进入控制器回路，选择需要配置的控制器通道和控制器类型，不同类型的控制器将影响设定和反馈的关系，即偏差的计算，如选择“Three-Component”类型控制器，最多可以将 3 个模拟量输入值叠加，作为一个 PID 回路的反馈值，默认值为具有固定设定点和串级功能的控制器，点击偏差按钮，进入参数化界面，如图 11-65 所示。

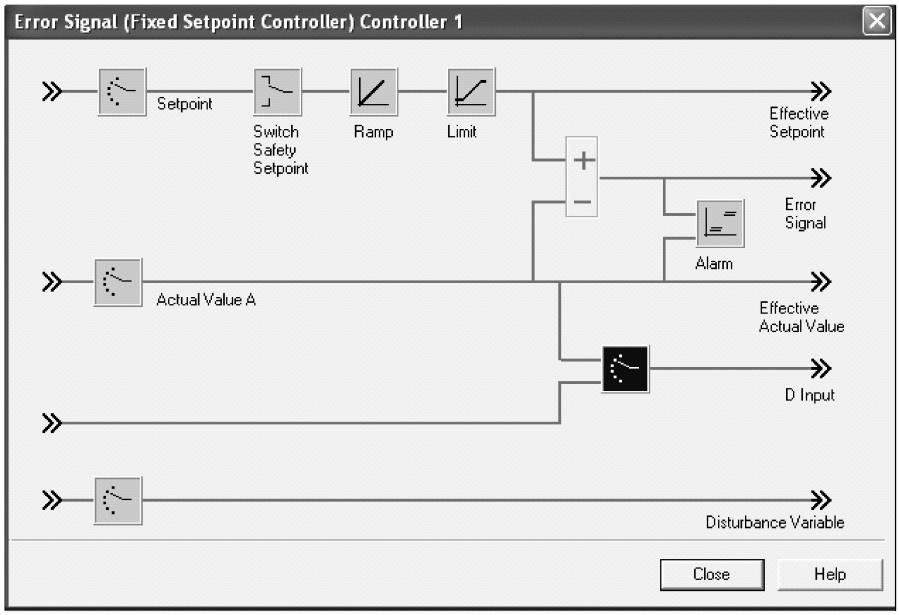



图 11-65 FM355C 模块偏差信号参数化界面

图 11-65 中参数的功能如下：

“Setpoint”　：选择 PID 设定点的来源，3 个选择：1）功能块；2）预处理的模拟量输入；3）控制器输出。设定点来自功能块为默认设置。

“Switch Safety Setpoint”：对后援操作（CPU 处于停止状态）中设定点的设置；

“Ramp”　：对设定值进行斜坡处理，防止设定产生大的阶越，引起输出震荡；

“Limit”　：对设定点进行线性化和限幅；

“Actual Value A”　：选择过程值通道；

“D Input”　：选择微分的方式：1）将偏差值进行微分；2）将过程值取反进行微分；3）将其他模拟量采集值进行微分，通过函数块 FM_PID中 D_EL_SEL 参数（背景数据块地址为 74）也可以在线修改；

“Alarm”　：对实际值或偏差值进行监控，设定报警和警告值的高低限；

“Disturbance Variable”：选择扰动变量的采样通道，即通过第几个模拟量输入通道采集扰动变量。

设定值与实际值的差值为偏差值，在控制算法中对偏差值进行计算，点击“Control Algorithm”按钮，进入控制算法界面，如图 11-66 所示。

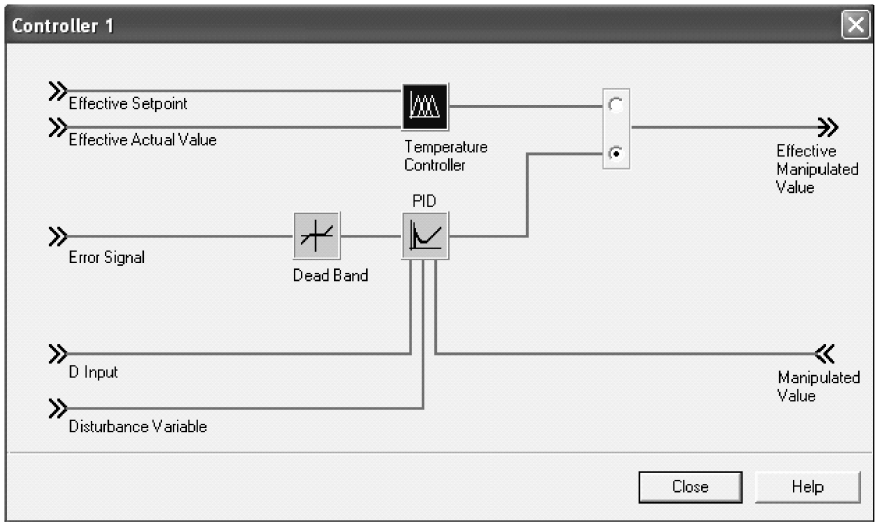



图 11-66　FM355C 模块控制算法参数化界面

在控制算法界面中，可以选择两种算法：1）温度算法；2）PID 算法。在温度算法中，可以选择加热或制冷控制，温度算法中集成优化的温度控制，控制简单，只需在 CPU 中调用函数块对控制过程识别（自整定过程）一次即可使用，不需要温度控制的经验参数，但是温控算法没有开放，在有些情况下（如测温元件安装问题或控制器机械问题）模块往往不能识别控制过程而不能使用；采用 PID 算法，偏差值经过“Dead Band”死区处理后，进入 PID 控制器进行运算，在“PID”中可以选择的控制方式，如 PI、PD、PID 控制，并可以

设定手自动无扰切换等参数。

点击“Controller Output”按钮，进入控制器回路输出界面，如图 11-67 所示。

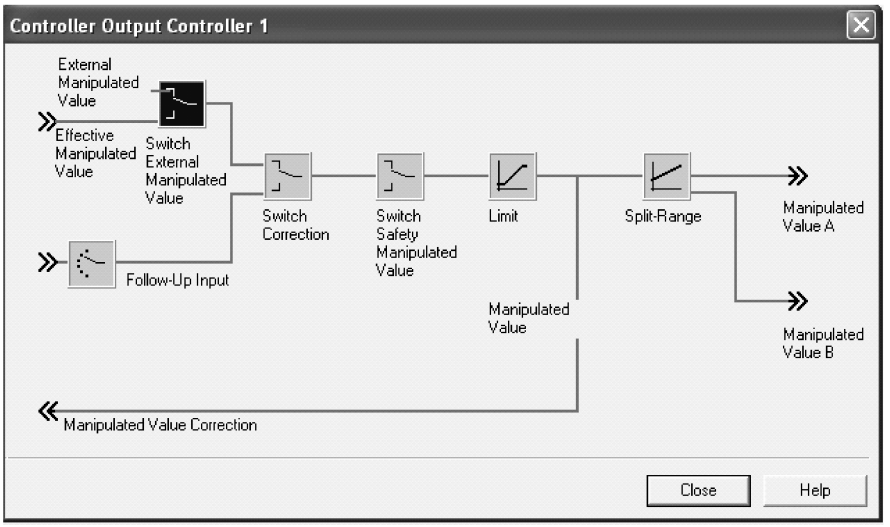



图 11-67　FM355C 模块控制回路输出参数化界面

图 11-67 中参数的功能如下：

“External Manipulated Value”　：选择手–自动操作模式，手动操作时，需要在 CPU 中对输出赋值；

“Follow-Up Input”　：跟随模式输入，选择零值或经过处理的模拟量输入直接作为控制器输出，防止手自动模式切换时输出产生阶越；

“Switch Correction”　：可以通过程序或集成的数字输入信号，选择输出源；

“Switch Safety Manipulated Value”　：可以通过程序或集成的数字输入信号，选择后援操作（CPU 处于停止状态）的输出值；

“Limit”　：对控制器输出值进行限幅；

“Split Range”　：将一个控制器的输出值“LMN”经过“Split Range”功能的转换，按输入、输出范围分成两个输出值“LMN_A”和“LMN_B”，通常情况下“LMN_B”为反向过程控制。在选项中，输入的开始值必须小于结束值，相对应的输出值没有限制。

在输出回路中，为控制器输出选择对应的模拟量输出以及输出的类型。FM355C 模块如果使用“Split Range”功能，可以将一个控制器的两个输出“LMN_A”和“LMN_B”连接到不同的模拟量输出通道（独立的控制回路数量减少）；在 FM355S 中数字量的输出端口是固定的，不需要选择，例如第一个控制器输出固定对应模块集成的第一个和第二个数字量输出，依此类推，第四个控制器输出对应第七个和第八个数字量输出，以温度控制为例，第一个数字量输出控制加热过程，第二个数字量输出控制冷却过程。参数化完成后编译存盘。

### 11.5.4　编程控制 FM355 模块

安装 FM355 参数化软件后，在 SIMATIC Manager 的函数库中自动创建 FM355 的专用函数库“FM_PID”。在 CPU 中调用 FM355 专用函数块，可以对 FM355 的参数进行设定，并读出反馈值，函数库中包括七个函数块，它们的功能如下：

“PID_FM”函数块　：主函数块，将设定点发送到 FM355 模块并读回反馈值，对模块大部分参数可以进行修改。

“FUZ_355”函数块　：读出 FM355 模块，通过温度优化算法的整定结果并保存于数据块中，更换模块后不需要再次对控制过程进行整定，只需要将存储的整定结果下传到模块即可。

“FORCE_355”函数块：模拟模块集成的模拟量和数字量输入信号。

“READ_355”函数块　：将模块集成的模拟量和数字量输入信号读到 CPU 中。

“CH_DIAG”函数块　：读出指定通道的状态值，用于对通道的诊断。

“PID_PAR”函数块　：对“PID_FM”函数块不能操作的参数进行操作，例如更改线条差补“Polyline”对话框中的数值。

“CJ_T_PAR”函数块　：在线更改内部补偿的温度值。

在模块参数化过程中，设定的数值及选项在模块初始化过程中（通过调用 PID_FM 函数，置位 COM_RST，将 FM355 模块的参数读到 CPU 中）将作为 PID 控制回路的初始值，通过调用专用函数，可以在控制过程中修正控制参数以适应控制过程的变化。

以软件自带的示例程序，介绍 FM355 模块的编程。安装 FM355 模块参数化软件后，在 SIMATIC Manager 中的“Sample project”示例程序目录下，可以发现 FM355 的示例程序“FM_PIDEx”。在示例程序中 FB100 和 FC101 模拟控制过程和反馈值，在实际控制中应将 FB100 和 FC101 删除。主程序在 FC100 中编写，在 OB100 中调用一次进行初始化，然后在 OB35 中循环调用与 FM355 模块进行数据交换，程序结构如图 11-68 所示。

OB35中循环调用程序

```
CALL   FC     100
COM_RST:=FALSE
CYCLE:=T#100MS

BE
```

OB100中初始化程序

```
CALL   FC     100
COM_RST:=TRUE
CYCLE:=T#100MS

BE
```

图 11-68　FM355 模块循环调用和初始化程序

在 OB100 中置位“COM_RST”参数，将存储于模块中所有控制参数复制到 CPU 数据块（FB31 的背景数据块）中，完成之后模块自动复位“COM_RST”参数，在 OB35 中不需要再次初始化，并以数据块中存储的参数作为 PID 回路初始控制参数，控制参数可以进行修改。在函数 FC100 中，首先对 PID 回路进行预处理，控制程序如下：

```
A     #COM_RST          //是否需要初始化
S     DB31. DBX  44. 0  //FB31 COM_RES 参数
S     DB31. DBX  56. 4  //切换到手动模式
L     S5T#1S            //延时 1s 跳转
SF    T          1
```

```
      A     T     1
      JCN   M001
      L     0.000000e+000
      T     DB31.DBD   52      //将手动值设置为 0
M001: A     DB31.DBX   40.4    //手动模式反馈信号
      R     DB31.DBX   56.4    //切换到自动模式
      SET
      S     DB31.DBX   44.2    //将 FM355 模块过程变量读到 CPU 中
      S     DB31.DBX   44.1    //将操作变量写到 FM355 模块
```

在回路预处理程序中，初始化命令（COM_RST）将 FM355 模块的参数复制到 CPU 中，并切换到手动模式，输出零值，保证了控制器启动时不会产生输出值，初始化命令执行完成后，COM_RST 和 DB31.DBX44.0 自动复位，并切换到自动控制模式，通过置位 DB31.DBX44.2 和 DB31.DBX44.1 对模块进行读写操作，每次读写完成后模块将读写位复位，所以必须循环置位读写位，读写过程是异步操作，保证了数据读写的快速性，但同时也增加了总线的负荷。如果不通过读写位进行读写操作，CPU 与 FM355 模块的通信则将自动通过 FM355 的逻辑地址完成，通信时间需要 2～3 个 CPU 扫描周期（只适合中央机架安装方式）。在后续的程序中，直接调用函数“FM_PID”（FB31）并生成背景数据块 DB31，选择模块地址和 PID 通道号与 FM355 模块建立通信关系。

每一个 PID 回路，都需要通过 FB31 建立一个独立的背景数据块，背景数据块是 CPU 与 FM355 模块的接口数据区。通过接口数据区，CPU 将设定值发送到模块中，同时模块将反馈的状态信息复制到接口数据区。在背景数据块中的参数有三种数据类型：

1）输入数据类型，数据的流向从 CPU 到 FM355 模块，在数据块中的地址及功能见表 11-16。

**表 11-16　PID_FM 函数背景数据块输入参数**

| 地址 | 参数 | 数据类型 | 描　　述 | 取值范围 | 默认值 | 解　　释 | 参数对话框 |
|---|---|---|---|---|---|---|---|
| 0.0 | MOD_ADDR | INT | FM355/455 模块地址 | | 256 | 模块逻辑地址 | — |
| 2.0 | CHANNEL | INT | 通道号 | 1～4 | 1 | 控制第几个 PID 回路 | — |
| 4.0 | PHASE | INT | PID 自整定控制器的相位 | 没有配置 | 0 | 与 PID 自整定函数（需单独购买）“PHASE”连接 | — |

2）输出数据类型，数据的流向从 FM355 模块到 CPU，在数据块中的地址及功能见表 11-17。

**表 11-17　PID_FM 函数背景数据块输出参数**

| 地址 | 参数 | 数据类型 | 描　　述 | 取值范围 | 属性 | 解　　释 | 参数对话框 |
|---|---|---|---|---|---|---|---|
| 6.0 | RET_VALU | INT | SFC58/59 返回值 | | 0 | 如果模块故障（QMOD_F）可以评估故障信息 | — |
| 8.0 | out_par | WORD | 模块输出参数起始标识 | W#16#3130 | W#16#3130 | 通过 READ_VAR = 1 读出模块输出信息 | — |

（续）

| 地址 | 参数 | 数据类型 | 描　　述 | 取值范围 | 属性 | 解　　释 | 参数对话框 |
|---|---|---|---|---|---|---|---|
| 10.0 | SP | REAL | 设定值 | 工艺参考数值范围 | 0.0 | PID 设定值反馈信号，用于监控 | — |
| 14.0 | PV | REAL | 过程变量 | 工艺参考数值范围 | 0.0 | PID 过程值反馈信号，用于监控 | — |
| 18.0 | ER | REAL | 偏差信号 | 工艺参考数值范围 | 0.0 | 偏差值反馈信号 | — |
| 22.0 | DISV | REAL | 扰动变量 | -100.0 ~ 100.0（%） | 0.0 | 扰动值反馈信号 | — |
| 26.0 | LMN | REAL | 输出变量 | -100.0 ~ 100.0（%） | 0.0 | PID 输出反馈信号 | — |
| 30.0 | LMN_A | REAL | Split range 功能输出 A | -100.0 ~ 100.0（%） | 0.0 | PID 输出 A 反馈信号 | — |
| 34.0 | LMN_B | REAL | Split range 功能输出 B | -100.0 ~ 100.0（%） | 0.0 | PID 输出 B 反馈信号 | — |
| 38.0 | QH_ALM | BOOL | 过程值高限报警 | | FALSE | 过程值超出 H_ALM 参数设定的监控值 | — |
| 38.1 | QH_WRN | BOOL | 过程值高限警告 | | FALSE | 过程值超出 H_WRN 参数设定的监控值 | — |
| 38.2 | QL_WRN | BOOL | 过程值低限警告 | | FALSE | 过程值超出 L_WRN 参数设定的监控值 | — |
| 38.3 | QL_ALM | BOOL | 过程值低限报警 | | FALSE | 过程值超出 L_ALM 参数设定的监控值 | — |
| 38.4 | QLMN_HLM | BOOL | 输出值高限报警 | | FALSE | 输出值超出设定的高限值 | — |
| 38.5 | QLMN_LLM | BOOL | 输出值低限报警 | | FALSE | 输出值超出设定的低限值 | — |
| 38.6 | QPARA_F | BOOL | 模块参数错误 | | FALSE | 模块检查参数的有效性，该位指示参数是否出错，详细信息可以通过调试工具中 PLC→Parameter Assignment Error 查看 | — |
| 38.7 | QCH_F | BOOL | 通道错误 | | FALSE | 指示控制器故障，详细信息通过诊断数据记录区 DS1 读出（调用 SFC59） | — |
| 39.0 | QUPRLM | BOOL | 斜坡功能 | | FALSE | 对设定值设置斜坡功能，该位指示经过斜坡时间后向上达到设定值 | — |
| 39.1 | QDNRLM | BOOL | 斜坡功能 | | FALSE | 对设定值设置斜坡功能，该位指示经过斜坡时间后向下达到设定值 | — |
| 39.2 | QSP_HLM | BOOL | 达到设定值上限 | | FALSE | 设定值的高低限，该位指示设定值超出监控的高限值 | — |
| 39.3 | QSP_LLM | BOOL | 达到设定值下限 | | FALSE | 设定值的高低限，该位指示设定值超出监控的低限值 | — |

（续）

| 地址 | 参数 | 数据类型 | 描　　述 | 取值范围 | 属性 | 解　　释 | 参数对话框 |
|---|---|---|---|---|---|---|---|
| 39.4 | QLMNUP | BOOL | 正向控制信号 | | FALSE | 步进控制器和脉冲控制器正向控制信号 | — |
| 39.5 | QLMNDN | BOOL | 反向控制信号 | | FALSE | 步进控制器和脉冲控制器反向控制信号 | — |
| 39.6 | QID | BOOL | 温度识别过程运行 | | FALSE | 指示识别过程（自整定过程）运行 | — |
| 40.0 | QSPOPON | BOOL | 设定值操作开关显示 | | FALSE | 指示是否使用调试工具输出设定值，如果该位为1，参数 SP_OP 作为设定值输出（使用调试工具和在程序中设定相同） | — |
| 40.1 | QLMNSAFE | BOOL | 后援操作 | | FALSE | 指示后援操作（CPU 处于停止状态）是否输出 | — |
| 40.2 | QLMNPON | BOOL | 输出值操作开关显示 | | FALSE | 指示是否使用调试工具设定输出值，如果该位为1，参数 LMN_OP 作为输出（使用调试工具和在程序中设定相同） | — |
| 40.3 | QLMNTRK | BOOL | 跟踪操作 | | FALSE | 指示输出值是否与设定的模拟量输入相匹配 | — |
| 40.4 | QLMN_RE | BOOL | 手动=1 自动=0 | | FALSE | 指示手自动操作状态 | — |
| 40.5 | QLMNR_HS | BOOL | 步进控制（FM355S） | | FALSE | 指示控制的阀门在高限位停止 | — |
| 40.6 | QLMNR_LS | BOOL | 步进控制（FM355S） | | FALSE | 指示控制的阀门在低限位停止 | — |
| 40.7 | QLMNR_ON | BOOL | 输出指示（FM355S） | | FALSE | 指定步进控制器设定的模式 | — |
| 41.0 | QFUZZY | BOOL | PID 算法=0<br>温度算法=1 | | FALSE | 指示选择的控制算法 | — |
| 41.1 | QSPLEPV | BOOL | 温度控制显示；<br>设定值小于过程变量 | | FALSE | 温度控制算法被选择，当前设定值小于过程值 | — |
| 41.2 | QSPR | BOOL | Split range 操作 | | FALSE | 指示是否使用 Split range 功能 | — |
| 41.4 | QMAN_FC | BOOL | 串级 PID 回路操作 | | FALSE | 主回路受控二级回路手动操作或由于二级回路设定或输出值受限幅控制而暂停主回路积分部分 | — |
| 41.7 | QPARABUB | BOOL | 内部变量 | | FALSE | 指示操作参数是否通过配置工具改变 | — |
| 42.0 | QMOD_F | BOOL | 模块错误 | | FALSE | | — |

3）输入/输出数据类型，命令从 CPU 传送到 FM355 模块，模块将反馈信息传送到与设定相同的地址中，数据的流向是双方向的，在数据块中的地址及功能见表 11-18。

**表 11-18　PID_FM 函数背景数据块输入/输出参数**

| 地址 | 参数 | 数据类型 | 描　　述 | 取值范围 | 属性 | 解　　释 | 参数对话框 |
|---|---|---|---|---|---|---|---|
| 44. 0 | COM_RST | BOOL | 从 FM355/455 读取控制参数 | | FALSE | 如果 COM_RST = TRUE 模块执行一次初始化过程，将模块的参数复制到 CPU 中，完成后模块复位COM_RST参数 | — |
| 44. 1 | LOAD_OP | BOOL | 将操作参数下传到 FM355/455 模块中 | | FALSE | 如果参数被置位，操作参数下传到模块中，完成后模块复位该参数 | — |
| 44. 2 | READ_VAR | BOOL | 从 FM355/455 读取变量 | | FALSE | 如果参数被置位，读取模块输出参数并存储到背景数据块中，完成后模块复位该参数 | — |
| 44. 3 | LOAD_PAR | BOOL | 将控制参数下传到 FM 355/455 模块中 | | FALSE | 如果参数被置位，控制参数下传到模块中，完成后模块复位该参数 | — |
| 46. 0 | OP_PAR | WORD | 操作参数的开始位置 | W#16#3130 | W#16#3130 | 操作参数开始标识符，指示下面的参数为操作参数，使用 LOAD_OP下传操作参数 | — |
| 48. 0 | SP_RE | REAL | 外部设定值 | 工艺参考数值范围 | 0. 0 | PID 的设定值 | — |
| 52. 0 | LMN_RE | REAL | 外部输出变量 | -100. 0 ~ 100. 0（%） | 0. 0 | 手动输出值 | — |
| 56. 0 | SP_OP_ON * | BOOL | 通过调试工具选择设定值开关 | | FALSE | 是否选择调试工具输出设定值，如果该位为 1，参数 SP_OP 作为设定值输出值（使用调试工具和在程序中设定相同） | — |
| 56. 1 | SAFE_ON | BOOL | 安全操作开关 | | FALSE | 指示后援操作值作为操作值输出 | — |
| 56. 2 | LMNOP_ON * | BOOL | 通过调试工具选择输出值开关 | | FALSE | 是否选择调试工具设定输出值，如果该位为 1，参数 LMN_OP 作为输出值（使用调试工具和在程序中设定相同） | — |
| 56. 3 | LMNTRKON | BOOL | 跟踪操作(LMN 取模拟量输入值) | | FALSE | 模块集成模拟量输入作为输出信号 | — |
| 56. 4 | LMN_REON | BOOL | 手动操作模式开关 | | FALSE | 选择手动模式，通过 LMN_RE 直接输出 | — |
| 56. 5 | LMNRHSRE | BOOL | 步进控制（FM355S） | | FALSE | 指示控制的阀门在高限位停止 | — |
| 56. 6 | LMNRLSRE | BOOL | 步进控制（FM355S） | | FALSE | 指示控制的阀门在低限位停止 | — |
| 56. 7 | LMNSOPON * | BOOL | 通过调试工具选择输出值开关（步进控制） | | FALSE | 如果置位，LMNUP_OP 和 LMNDN_OP 可以作为输出信号 | — |
| 57. 0 | LMNUP_OP * | BOOL | 输出信号 | | FALSE | 与 LMNSOPON 相关 | — |

（续）

| 地址 | 参数 | 数据类型 | 描　　述 | 取值范围 | 属性 | 解　　释 | 参数对话框 |
|---|---|---|---|---|---|---|---|
| 57.1 | LMNDN_OP * | BOOL | 输出信号 | | FALSE | 与 LMNSOPON 相关 | — |
| 57.3 | LMNRS_ON | BOOL | 输出仿真 | | FALSE | 模拟控制阀门的位置信号 | — |
| 57.4 | FUZID_ON | BOOL | 温度算法，开始过程识别 | | FALSE | 过程识别（自整定过程）开始，完成后需用户复位 | — |
| 58.0 | SP_OP * | REAL | 设定值操作 | 技术参考数值范围（物理变量） | 0.0 | 如果参数 SP_OP_ON 为 1，该参数作为设定点。在调试工具上与设定值相连 | — |
| 62.0 | LMN_OP * | REAL | 操作变量状态 | -100.0 ~ 100.0（%） | 0.0 | 如果参数 LMN_OP_ON 为 1，该参数作为输出值。在调试工具上与输出值相连 | — |
| 66.0 | LMNRSVAL | REAL | 仿真中输出的初始值 | -100.0 ~ 100.0（%） | 0.0 | 只适合没有位置反馈的步进控制 | — |
| 70.0 | CONT_PAR | WORD | 控制参数的开始位置 | W#16#3 130 | W#16#3 130 | 控制参数开始标识符，指示下面的参数为操作参数，使用 LOAD_PAR 下传控制参数 | — |
| 72.0 | P_SEL | BOOL | 使能比例功能 | | TRUE | PID 算法中，P、I、D 分量可以单独使能 | PID 控制器 |
| 72.1 | PFDB_SEL | BOOL | 比例值作为 PID 算法的反馈 | | FALSE | 如果该参数置位，P 分量可以作为 PID 算法的反馈 | PID 控制器 |
| 72.2 | MONERSEL | BOOL | 监控：<br>过程变量 =0<br>偏差信号 =1 | | FALSE | 选择监控偏差信号或过程值信号的限幅值 | 报警控制器 |
| 74.0 | D_EL_SEL | INT | 控制器 D 分量输入 | 0 ~4 或 17 | 0 | 选择将哪些值进行微分处理，<br>0：偏差信号；<br>1 ~4：模拟量输入1 ~4；<br>17：过程值取反 | 偏差信号（…）控制器 |
| 76.0 | SP_HLM | REAL | 设定值高限 | > SP_LLM（物理变量） | 100.0 | 设定值高限监控值 | 设定值限幅控制器 |
| 80.0 | SP_LLM | REAL | 设定值低限 | < SP_HLM（物理变量） | 0.0 | 设定值低限监控值 | 设定值限幅控制器 |
| 84.0 | H_ALM | REAL | 高限报警 | > H_WRN（物理变量） | 100.0 | 与参数 MONERSEL 有关，设定偏差信号或过程值高限报警值 | 警报控制器 |
| 88.0 | H_WRN | REAL | 高限警告 | H_ALM ~ L_WR N（物理变量） | 90.0 | 与参数 MONERSEL 有关，设定偏差信号或过程值高限警告值 | 警报控制器 |
| 92.0 | L_WRN | REAL | 低限警告 | H_WRN ~ L_AL M（物理变量） | 10.0 | 与参数 MONERSEL 有关，设定偏差信号或过程值低限警告值 | 警报控制器 |
| 96.0 | L_ALM | REAL | 低限报警 | < L_WRN（物理变量） | 0.0 | 与参数 MONERSEL 有关，设定偏差信号或过程值低限报警值 | 警报控制器 |
| 100.0 | HYS | REAL | 时滞 | ≥0.0（物理变量） | 1.0 | 设定时滞值，减少报警信号的频繁触发 | 警报控制器 |

（续）

| 地址 | 参数 | 数据类型 | 描　　述 | 取值范围 | 属性 | 解　　释 | 参数对话框 |
|---|---|---|---|---|---|---|---|
| 104.0 | DEADB_W | REAL | 死区带宽 | ≥0.0（物理变量） | 0.0 | 设定死区带宽，偏差值在死区带宽内，控制器不进行计算 | 死区控制器 |
| 108.0 | GAIN | REAL | 比例增益 | 所有数值（非复数） | 1.0 | 设定比例增益 | PID 控制器 |
| 112.0 | TI | REAL | 积分时间/s | =0.0 或者≥0.5 | 3000.0 | 设定的积分时间 | PID 控制器 |
| 116.0 | TD | REAL | 微分时间/s | =0.0 或者≥1.0 | 0.0 | 设定的微分时间 | PID 控制器 |
| 120.0 | TM_LAG | REAL | 微分作用中的时间滞后 | TM_LAG≥0.5 | 5.0 | 设定的微分滞后时间 | PID 控制器 |
| 124.0 | LMN_SAFE | REAL | 后援操作变量 | -100.0～100.0（%） | 0.0 | 设定的后援操作输出值 | 后援输出值控制器 |
| 128.0 | LMN_HLM | REAL | 输出值高限 | LMN_LLM～100.0（%） | 100.0 | 设定输出值高限 | 输出限幅值控制器 |
| 132.0 | LMN_LLM | REAL | 输出值低限 | -100.0～LMN_HLM（%） | 0.0 | 设定输出值低限 | 输出限幅值控制器 |
| 136.0 | MTR_TM | REAL | 电机输出变量/s | MTR_TM≥0.001 | 60.0 | 阀门开关的运行时间（只适合步进控制器） | 脉冲发生器 |
| 140.0 | PULSE_TM | REAL | 最小脉冲时间/s | ≥0.0 | 0.2 | 防止控制设备的抖动而设定最小脉冲时间（只适合步进控制器或脉冲控制器） | 脉冲发生器分程/脉冲发生器 |
| 144.0 | BREAK_TM | REAL | 最小脉冲间隔时间/s | ≥0.0 | 0.2 | 防止控制设备的抖动而设定最小脉冲间隔时间（只适合步进控制器或脉冲控制器） | 脉冲发生器分程/脉冲发生器 |

*：参数也可以在调试工具中修改。

在输入/输出类型中，从参数“op_par”（地址 46.0）到参数“cont_par”（地址 70.0）为操作参数，如 PID 回路的设定点、手动输出参数等，参数“cont_par”以后为控制参数，如比例参数、积分参数等。CPU 与 FM355 模块数据的传递通过控制位完成，控制位功能如下：

“COM_RST”（地址 44.0）：在 CPU 初始化过程中，将存储于 FM355 模块的控制参数值复制到背景数据块中，完成后模块将该位复位；

“LOAD_OP”（地址 44.1）：为 1 时，将操作参数传送到 FM355 模块中，完成后模块将该位复位；

“READ_VAR”（地址 44.2）：为 1 时，将输出类型参数复制到背景数据块中，完成后模块将该位复位；

“LOAD_PAR”（地址 44.3）：为 1 时，将控制参数传送到 FM355 模块中，完成后模块将该位复位。

在实际编程中，自动模式下背景数据块参数“SP_RE”（地址 48.0）作为 PID 回路的设定点，“PV”（地址 14.0）作为实际反馈值用于监控，参数“LMN”（地址 26.0）以百分比的形式作为输出值反馈用于监控，输出信号直接通过集成的模拟量或数字量端口输出。通过参数“LMN_REON”（地址 56.4）或通过模块集成的数字输入点（在硬件配置中设定）作

为手动模式的选择开关，参数“LMN_RE”（地址 52.0）作为手动值输出。

### 11.5.5 监控、调试

FM355 模块带有参数化调试工具，在程序中每一个 PID 回路需要占用一个 FB31 的背景数据块，通过调试工具打开 PID 回路相对应的数据块，可以对 PID 回路进行调试，调试中测试的数值保存在相应的背景数据块中。在 FM355 模块参数化界面中，点击菜单“Test”→“Loop Monitor”→“Open Instance DB”，选择 PID 回路的背景数据块，进入回路监控界面，如图 11-69 所示。在回路监控界面中，可以监控 PID 的参数及 PID 控制回路的设定、反馈和输出的值，也可以修改设定值、切换到手动模式的输出。

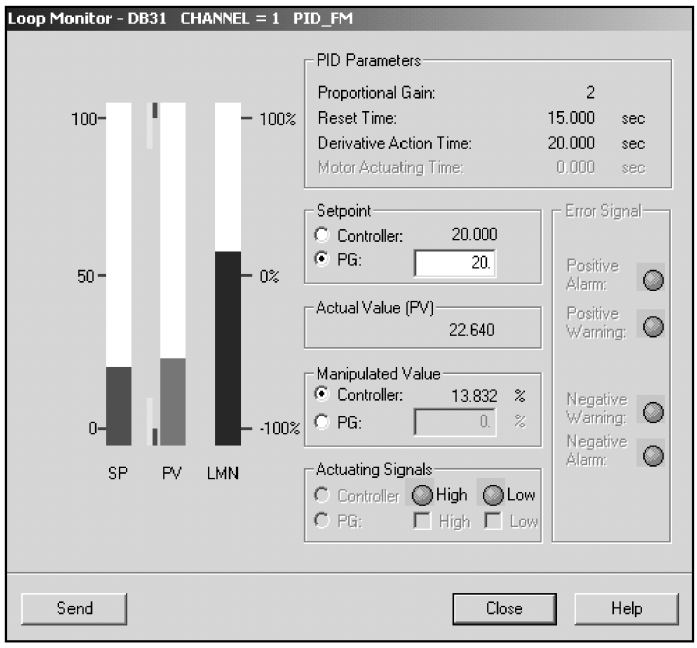



图 11-69 FM355 回路监控界面

点击菜单“Test”→“Curve Recorder”→“Open Instance DB”，选择 PID 回路的背景数据块，参考 PID 回路操作参数曲线，图 11-70 所示为 PID 回路设定、反馈和输出值曲线。通过柱状图和曲线图可以对 PID 回路进行监控，如果需要在线修改 PID 的控制参数，可以在参数化界面修改后，点击下载键，直接下传到 FM355 模块中。

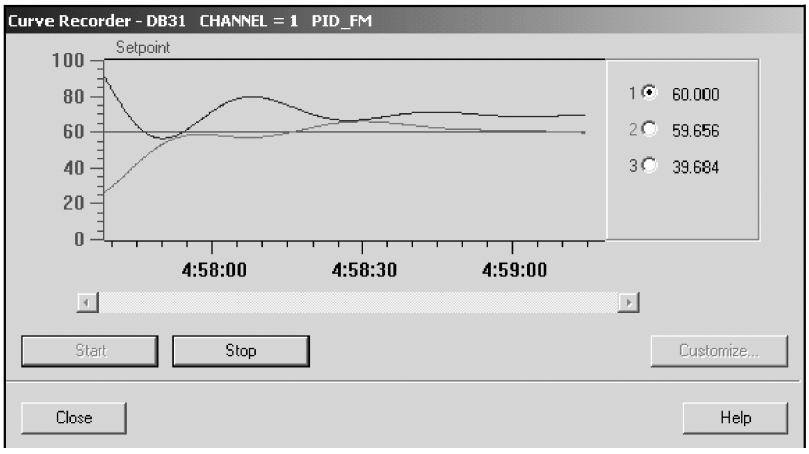



图 11-70 FM355 参数曲线界面

### 11.5.6 控制器参数的优化

FM355 模块有两个控制器 – 专用温度控制器和 PID 控制器，通过程序控制方式可以

对温度控制器参数进行优化（自整定），置位 FB31 背景数据块中的参数“FUZID_ON”（地址 57.4），给设置值一个阶越，阶越值必须大于整个控制范围的 12% 并且大于 5 度，模块将自动识别控制过程，并可以通过函数块“FUZ_355”监控控制器的优化过程，优化过程完成后必须复位参数“FUZID_ON”，否则将进行新的过程识别。通过调试工具，可以手动对温度控制器和 PID 控制器进行优化，点击菜单“Test”→“Controller Optimization”，进入控制器优化界面，根据界面中的提示，按步骤进行过程优化，例如进入数据采集界面如图 11-71 所示。

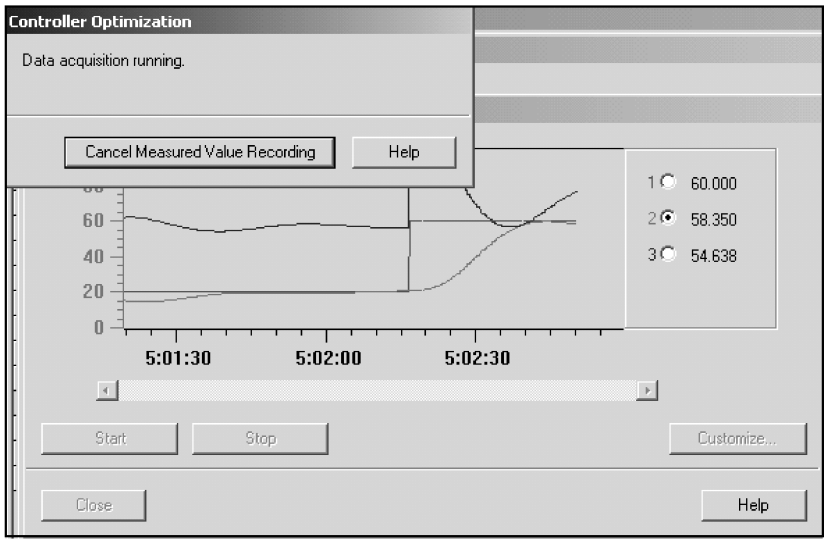



图 11-71　FM355 控制器参数的优化 – 数据采集界面

优化过程完成后计算出新的 PID 控制参数，如图 11-72 所示。

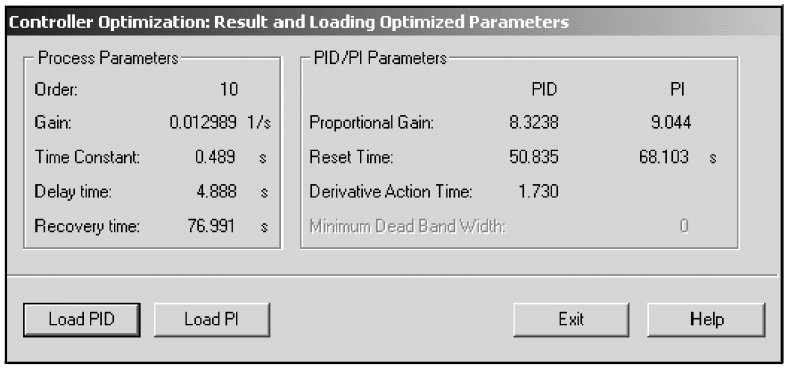



图 11-72　FM355 控制器参数优化结果界面

选择合适的控制方式，如 PI 或 PID 控制，点击“Load PID”或“Load PI”按钮，使用优化的参数替代原设定参数。

### 11.5.7　通过操作面板的后援操作

CPU 故障停机后（CPU 不能掉电），FM355 模块进入后援操作模式，在模块中设定的安

全值将作为 PID 回路后援操作的设定值或直接输出值。如果连接操作面板，例如 MP270 连接到 PLC 站的 MPI，一种方式是通过 CPU 间接控制 FM355 模块，另一种方式是直接控制 FM355 模块，在后面的连接方式中，即使 CPU 处于停止的模式下也可以控制 FM355 模块的参数，例如 MP270 连接到 CPU 的 MPI 上，在选择控制器时可以直接选择连接到 FM355 模块，如图 11-73 所示。

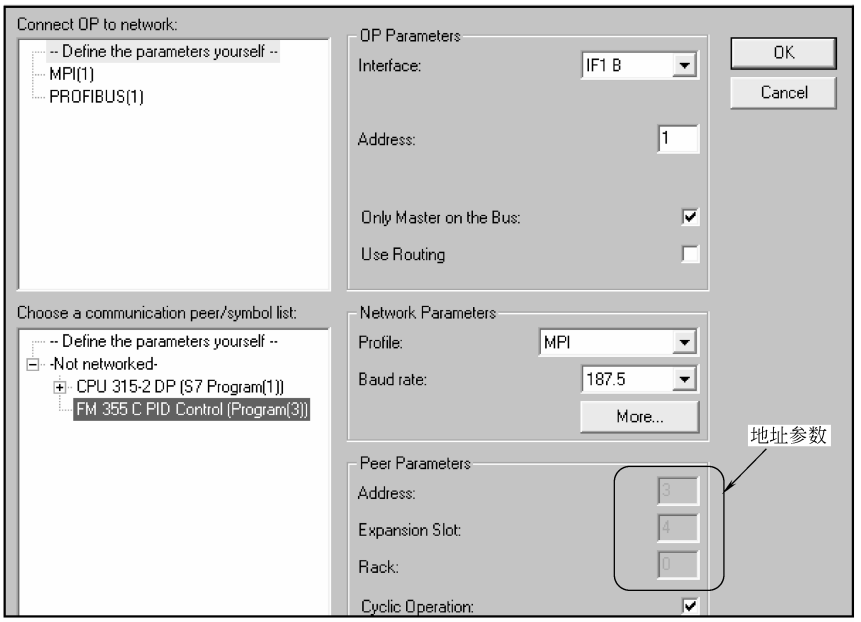



图 11-73　操作面板直接连接 FM355

在地址参数中，选择 FM355 模块的 MPI 地址及槽号，如果通过其他网络连接到 CPU，可以通过路由的方式连接 FM355 模块。数据块 DB101 ~ DB104 分别代表一个 FM355 模块的四个通道，数据块中参数的格式可以参考 FM355 手册，这里不再介绍。通过操作面板直接连接 FM355 模块时，只有在后援操作模式下才可以对模块进行写操作，对模块的读操作没有限制。

FM355 模块的示例程序 FM355. zip、参数化软件及电子手册参考光盘“功能模块 FM”目录。

# 第 12 章　程 序 调 试

## 12.1　建立与 CPU 的连接并进行设置

编程器与 CPU 之间建立在线连接，用于下载 S7 用户程序、从 CPU 上传程序到编程器和其他操作：

- 调试用户程序；
- 显示和改变 CPU 操作模式；
- 为 CPU 设置时间和日期；
- 显示模板信息；
- 比较在线和离线的程序块；
- 诊断硬件。

### 12.1.1　设置 PG/PC 接口

在“SIMATIC Manager”界面下，使用菜单命令“Options”→“Set PG/PC Interface”或在视窗中点击菜单“Start”→“Settins”→“Control Panel”→“Set PG/PC Interface”，都可以打开编程器接口设置程序，使用控制面板中的设置程序还可以选择相关的访问点。

通过 MPI、PROFIBUS 和工业以太网可以与 CPU 建立在线连接，在编程器上需要指定适配器、通信处理器或以太网网卡用于连接不同的网络：

（1）MPI 网络

首先选择访问点“S7_ONLINE”，然后在接口栏中选择使用的通信接口，如“S7_ONLINE→PC Adapter（MPI）”或“S7 _ ONLINE → CP5512/CP5611/CP5613（MPI）”。点击“Properties”按钮，设置接口在 MPI 网络的地址及通信速率。CP5512/CP5611/CP5613 通信处理器带有诊断功能，选择接口后可以点击“Diagnostics. .”按钮，对连接的站点进行检测，如图 12-1 所示，0 号

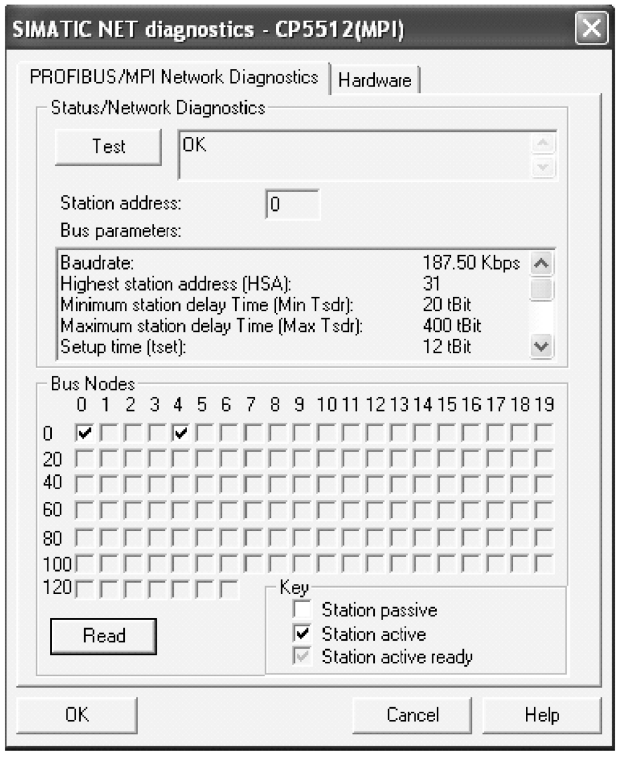



图 12-1　CP5512 的诊断功能

地址为 CP5512 自身的地址，4 号地址为连接通信方的地址。如果没有读出通信方的地址，不可能与 CPU 建立通信。

**注意：**

1）接口与通信方的通信速率必须匹配。

2）网络上站地址必须惟一。

3）PC Adapter（MPI）为串行接口转换 MPI 的适配器，没有网络诊断功能。串口侧的设置必须与连接 PC 的串口相匹配，MPI 侧的设置必须与通信方 CPU 的 MPI 相匹配。

（2）PROFIBUS 网络

首先选择访问点“S7_ONLINE”，然后在接口栏中，选择使用的通信接口，如“S7_ONLINE→PC Adapter（PROFIBUS）”或“S7_ONLINE→CP5512/CP5611/CP5613（PROFIBUS）”，参数和诊断功能与 MPI 设置相同。在 PLC 侧连接的接口可以选择 CPU 集成的 DP 接口或通信处理器，如 CP342-5 等。

**注意：**

1）接口与通信方的通信速率必须匹配。

2）网络上站地址必须惟一。

3）CPU 集成的 DP 接口及 CP 的 PROFIBUS 站地址必须进行初始化（程序下载）后，才能使用。

4）“PC Adapter”需要由通信方的接口提供电源（如主站接口），大部分的从站接口（如 ET200M）不能提供电源，所以不能插入从站的编程接口对网络上的主站进行编程。CP5512/CP5611/CP5613 由 PC 供电，可以插入 PROFIBIUS 网络中任意站点上使用。

（3）工业以太网网络

首先选择访问点“S7_ONLINE”，然后在接口栏中选择使用的通信接口，如“S7_ ONLINE →TCP/IP→Intel（R）PRO/1000 MT M...”或“S7_ONLINE → ISO Ind. Ethernet→Intel（R）PRO/1000 MT M...”，选择 TCP/IP（使用 IP 地址）或 ISO（使用 MAC 地址）协议必须与通信方匹配。

**注意：**

1）网络上地址必须惟一。

2）以太网接口必须进行初始化（使用默认的 MAC 地址或通过“SIMATIC Manager”菜单“PLC”→“Editing Ethernet Nodes”分配 IP 地址）后，才能使用。

3）编程器以太网网卡的 IP 地址必须与 PLC 以太网接口在相同的网段。

4）PROFINET 接口只支持 TCP/IP 的通信服务，编程接口应选择“S7_ONLINE→TCP/IP→网卡名”。

（4）使用网络路由功能

如果在一个项目中使用多种网络（MPI、PROFIBUS、以太网）或站点间通过网络相互连接，这样使用编程器连接网络的任意一点，即可对整个网络上的设备进行访问。如图 12-2 所示，编程器可以对网络上 3 个站点进行编程访问，访问 ET200S 站点的网络路由路径为：编程器 MPI CP5611→CPU414-3 MPI→ CP443-5→CP343-5→CPU315-2DP→ET200S CPU。在大多数的应用中，编程器（工程师站）位于主控室，通过以太网路由到其他网络。实现网

络路由的编程功能具体步骤如下：

1）所有 PLC 站必须在同一项目下。

2）在“SIMATIC Manager”中，点击“Options”→“Configure Network”菜单，或者双击当前项目下的网络，如“Profibus”，进入 NETPRO 界面。

3）在硬件目录“Stations”中，将“PG/PC”拖放到网络配置界面中。

4）双击网络界面中的“PG/PC”，在“Interfaces”对话框中添加编程器所在的网络，在图 12-2 中为 MPI 网络，所以在这里添加 MPI 网络并使之与 S7-400 系列 PLC 站在同一 MPI 网络中，定义编程器中使用 MPI 的站地址，站地址与“PG/PC Interface”中设置的地址要一致。

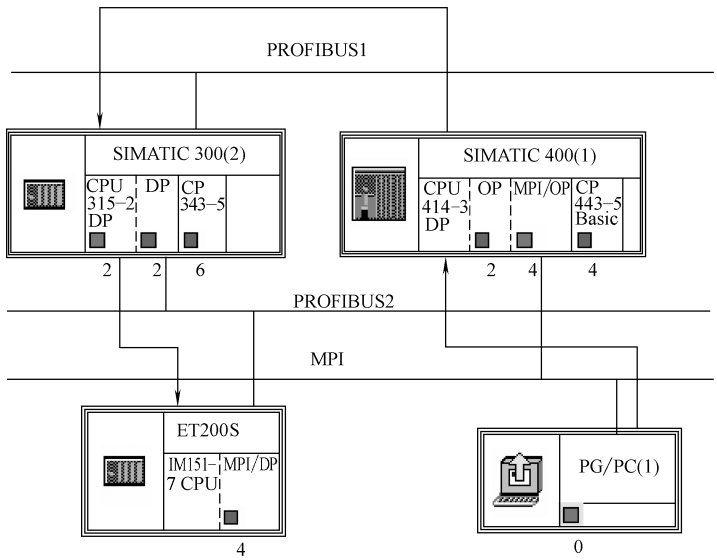



图 12-2　网络路由功能示例

5）在“Assignment”对话框中，选择使用的 MPI，如 PC 适配器、CP5611 等，点击“Assign”键，将选择的适配器或通信处理器作为编程接口。

6）配置完的编程器会出现黄色线条连接到 MPI 网络上，同时“PG/PC interface”中参数也被设定，如果修改“PG/PC interface”接口参数，需要在网络配置界面中重新设置编程接口参数。

## 12.1.2　建立在线连接

1. 通过“Accessible Nodes”建立在线连接

在“SIMATIC Manager”中使用菜单命令“PLC”→“Display Accessible Nodes”，打开“Accessible Nodes”窗口，在“Accessible Nodes”窗口中显示所有可访问的可编程序控制器站点及其地址。网络上编程器或操作面板站点也可以显示出来。

显示的站点带有下列信息：

- (direct)：该节点直接连接到编程器上。
- (passive)：该站点不能通过 PROFIBUS-DP 进行编程、状态监控和修改。
- (waiting)：由于该节点与网络中其他的设置不匹配，所以不能与该节点通信。

这种访问方式可以快速访问可编程序控制器，例如用于测试的目的。在编程器中没有可编程控制器的硬件配置数据时可选择该种方式，打开站点后可以查看、修改和删除 CPU 在线的程序块。

2. 通过在线功能建立连接

在“SIMATIC Manager”中，使用菜单命令“View”→“Online”打开一个在线窗口。该窗口显示 PLC 中的项目数据（与离线窗口相比较，离线窗口显示编程器中的项目数据）。

该连接方式可以显示 PLC 站的硬件信息（离线项目中必须带有硬件配置信息），如 CPU 的诊断信息、运行模式及模块的状态。

### 12.1.3 显示和改变 CPU 的操作模式

在联机的情况下，选择 PLC 站中的 CPU，使用“SIMATIC Manager”菜单，命令“PLC”→“Diagnostics/Settings”→“Operating Mode”打开 CPU 操作模式界面，在界面中通过相应的按钮，可以将 CPU 的操作模式切换为“STOP”、“Warm Restart”、“Cold Restart”和“Hot Restart”。

**注意：**

1）CPU 面板上的开关只有在“RUN”的位置上，才可以使用软件进行操作。

2）如果 CPU 不支持某些操作模式，相应的按钮将变为灰色，不可操作。

### 12.1.4 显示和改变 CPU 的时钟

CPU 时钟信号的应用都是与消息相关的，例如 CPU 内部的诊断缓存区可以记录所有信息的触发和结束时间；可以记录信号（例如快速输入信号）上升沿和下降沿的触发时间，并将消息上传到 PC 归档；PLC、HMI 站间的时钟信号可以相对（以某一个站的时钟信号作为主时钟）或绝对（GPS）同步。使用“SIMATIC Manager”菜单命令“PLC”→“Diagnostics/Settings”→“Set Time of Day”，打开 CPU 时钟界面，时钟界面显示当前编程器和 CPU 的时钟信息，如图 12-3 所示。

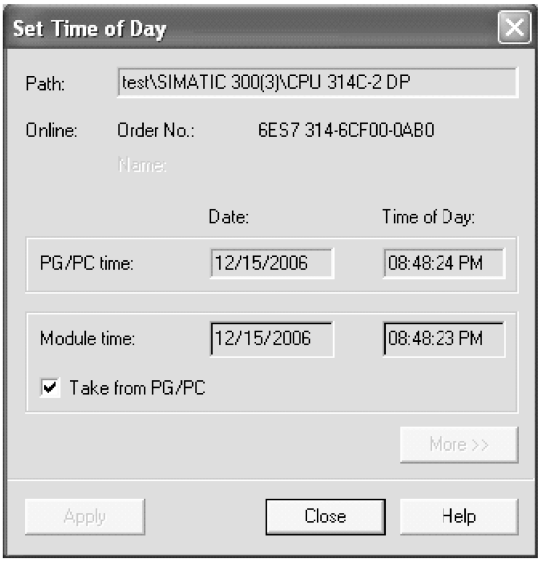



图 12-3 CPU 时钟界面

选择“Take from PG/PC”选项，点击“Apply”按钮，可以将编程器的时钟信号赋值到 CPU 中，不选择“Take from PG/PC”选项，CPU 的时钟信号可以手动修改。

在程序中可以通过调用 SFC0、SFC1，对 CPU 时钟信号进行读写操作。

### 12.1.5 在线更新硬件固件版本

固件（Firmware）相当于智能模块（例如 CPU、CP、IM）的操作系统，智能模块功能的更新以及一些小故障的更正，可以通过固件版本的升级实现。通过在线联机的方式可以更新硬件的固件版本。具体的步骤如下：

1）从西门子公司的网站上下载相关的更新文件。

2）在 PLC 硬件配置界面中，选择所要更新的模块，使用菜单命令“PLC”→“Update Firmware”打开相应的操作界面，选择更新的文件，点击“Run”按钮，进行固件版本的更新操作。

3）固件版本更新后，可以使用模块新的功能。

**注意：**

1）在更新操作中，模块不能掉电或拔插通信电缆。

2）有些模块，如 CPU314C 不支持在线更新功能，菜单命令不能激活。这些模块必须使用 MMC 或 FLASH 卡及带有 MMC 或 FLASH 读写装置的编程器进行更新。

3）在下载固件版本文件时，提示更新文件适用的硬件订货号、版本号、升级的方法及具体步骤，从中可以知道该固件版本是否支持在线更新功能。

## 12.2 程序的下载、上传、复位操作

### 12.2.1 程序的下载

程序编写完成后，通过 SIMATIC Manager 的操作，可以将程序下载到 CPU 中，程序首先下载到 CPU 的装载存储器中（EPROM：程序保持不需要电池有电池时可以保持过程数据；RAM：程序保持需要电池）。与执行相关的部分程序存储于工作存储器中，例如函数 FC1，在程序中没有调用或当前没有调用，FC1 存储于装载存储器中，如果条件满足主程序调用 FC1，则 FC1 同时存储于装载存储器和工作存储器中。下载的过程如图 12-4 所示。

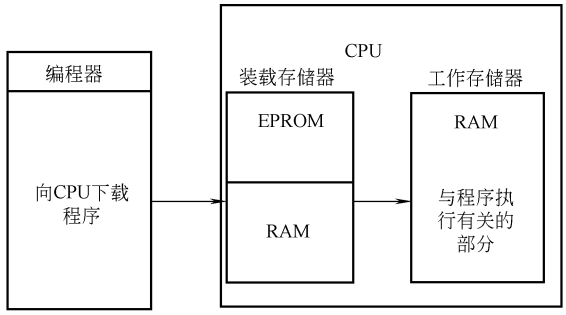



图 12-4 程序的下载

有下面几种方式可以将用户程序下载到 CPU 的 RAM 中：

1）在 SIMATIC Manager 中，选择需要下载的对象，点击下载按钮，就可以将该对象的内容下载到 CPU 中。选中一个块时，下载该块；选中“Blocks”目录时，下载整个目录中所有的块；选中 PLC 站，则下载该站内的所有用户程序块和硬件配置信息。（“Blocks”目录只可能包含一个 CPU 的所有用户程序和硬件配置信息，PLC 站在多个 CPU 处理时包含多个 CPU 的用户程序和硬件配置信息）

2）在程序编辑器中，点击下载按钮，下载当前正在编辑的程序块或数据块。

3）在硬件配置界面中，点击下载按钮，下载当前正在编辑的硬件配置信息。

有下面几种方式可以将用户程序下载到 CPU 的 EPROM 中：

1）使用下载功能可以直接将程序存储于 S7-300 系列 PLC CPU 的 MMC 中。

2）使用西门子公司的 PG 或带有外置 EPROM 读写器的 PC 离线下载。下载前必须在视窗控制面板中的“Memory Card Parameter Assignment”对话框中设置使用 EPROM 读写器的

类型，在“SIMATIC Manager”中使用菜单命令“File”→“S7 Memory Card”→“Open”将程序复制到存储卡中。MMC 卡也可以使用相同的方式进行复制。

3）将 EPROM 插入 CPU 的插槽中，在 SIMATIC Manager 中选择需要下载的“Blocks”目录，使用菜单命令“PLC”→“Download User Program to Memory Card”，将程序复制到存储卡中，该操作必须在 CPU 停止模式下进行。

4）使用菜单命令“PLC”→“Copy RAM to ROM”，将程序复制到 CPU 集成的 ROM 中（适用 CPU312IFM、CPU314IFM 及 C7 相应的 CPU，这些 CPU 内部集成 ROM，没有存储器扩展插槽）。

**注意：**

1）项目的属性为“PCS7”（点击项目名，按鼠标右键可以查看项目属性）时，只能将整个 PLC 站同时下载。

2）除 S7-400 系列 PLC 的 CIR（Configuration In Run）功能及冗余系统外，所有与硬件配置相关的下载都要求 CPU 处于 STOP 模式。

3）CPU 通过程序语法和 CPU 资源（例如超出定时器的数量）对用户程序进行双层校验，如果错误，CPU 进行 STOP 模式。

### 12.2.2 程序的上传

上传与下载的过程相反，将存储于 CPU 装载存储器中程序复制到编程器的硬盘中。有下面几种方式可以实现程序的上传：

1）在 SIMATIC Manager 中，使用菜单命令“PLC”→“Upload Station to PG”命令可以将 PLC 站中的程序上传到编程器中，上传的程序包括硬件配置信息和用户程序。

2）在 SIMATIC Manager 中，点击工具栏中“Accessible Nodes”按钮，或选择菜单命令“PLC”→“Display Accessible Nodes”，将站点中的程序块复制到编程器中。

3）在 SIMATIC Manager 中，使用菜单命令“View”→“Online”打开项目的在线窗口，将窗口中的程序块复制到编程器中。

4）在硬件配置界面中，可以通过工具栏上的上载按钮上载 PLC 站。以这种方式上传会在项目中自动插入一个站，只包含硬件配置信息而不包含用户程序。

如果 CPU 中包含硬件配置信息，可以将站中硬件信息上传。如果 CPU 中不包含硬件配置信息，例如新安装的 PLC 站点，对于 S7-300 系列 PLC 上传的硬件配置包括扩展机架，但是不能上传分布式 I/O（DP）部分；对 S7-400 系列 PLC，只能上传主机架配置，不能上传扩展机架和分布式 I/O，S7-300、400 系列 PLC 上传的硬件信息不完整，必须重新定义，如分配模块的订货号和参数等。

上传的配置信息和用户程序与源程序相比并不完整，例如：

- 程序块不包含任何参数、变量的符号名称。
- 程序块不包含任何注释。
- 不能更进一步地处理全局数据通信（GD）以及与符号相关的消息。

### 12.2.3 CPU 存储器复位

通过 SIMATIC Manager 软件的操作和 CPU 前面板的开关，可以将 CPU 中的存储器复位，

不同类型 CPU 的复位方法不同：

1. 不带有 MMC 的 S7-300 系列 PLC CPU 及 S7-400 系列 PLC CPU 存储器的复位

• 在“SIMATIC Manager”界面中，选择 PLC 站的 CPU，使用菜单命令“PLC”→“Diagnostics/Settings”→“Clear/Reset”清除 CPU 存储器中的程序。

• 通过 CPU 前面板开关复位，将开关切换到“STOP”位置，按开关至“MRES”位置，停止灯慢闪，如果停止灯常亮，抬起开关至“STOP”位置后，快速按至“MRES”位置，停止灯快闪，抬起开关至“STOP”位置，存储器复位。如果没有成功，重复尝试。

• 如果 CPU 插入 FLASH（闪存）存储器卡，复位操作不能删除 FLASH 中的程序，再次接电后，FLASH 中的程序将复制到工作存储器中。FLASH 存储器卡中的程序只能通过西门子公司的 PG 或带有外置 EPROM 读写器的 PC 删除。将 FLASH 卡插入 CPU 的插槽中，在 SIMATIC Manager 中选择空的“Blocks”目录，使用菜单命令“Download User Program to Memory Card”，可以将存储卡中的程序覆盖到达清除的目的。

• 使用菜单命令“PLC”→“Copy RAM to ROM”覆盖 CPU 集成 ROM 中的程序。

2. 带有 MMC 的 S7-300 系列 PLC CPU 复位

• 离线删除：将 MMC 卡插入到西门子公司的 PG 或带有外置 EPROM 读写器的插槽中，使用菜单命令“File”→“S7 Memory Card”→“Delete”，删除 MMC 中的程序。

• 在线删除：使用菜单命令“View”→“online”打开在线窗口，删除选择的程序块。

### 12.2.4 删除 CPU 中的程序块

1）在“SIMATIC Manager”中，使用菜单命令“View”→“Online”打开在线窗口，在“Blocks”目录中，可以将所有或某些程序块删除。

2）在“SIMATIC Manager”中，使用菜单命令“PLC”→“Display Accessible Nodes”打开可访问站点窗口，在“Blocks”目录中，可以将所有或某些程序块删除。

## 12.3 使用变量表进行调试

使用变量表，可以保存各种测试环境，有效地进行测试和监控。项目中保存变量表的数量没有限制。变量监控表有下列作用：

• 监视变量

该功能可以在编程器上显示用户程序，或 CPU 中每个变量的当前值。

• 修改变量

可以对用户程序或 CPU 中的每个变量进行修改。用户程序可以将变量值覆盖。

• 外设输出

允许 CPU 在停止模式下将固定值赋予每个输出。

• 强制变量

可以对用户程序或 CPU 中每个变量赋予一个固定值，这个值是不能被用户程序覆盖的。

### 12.3.1 变量表的创建

通常有两种方法创建变量表：

1）在“SIMATIC Manager”中，选择“Blocks”目录，使用菜单命令“Insert”→“S7 Block”→“Variable Table”插入变量表，在弹出的对话框中，可以命名变量表，通过双击打开变量表。在项目中插入变量表的数量没有限制。

2）在程序编辑窗口中，使用菜单命令“PLC”→“Monitor/Modify Variables”生成一个没有名称的变量表，可以将变量表存储于与项目相关的目录中。

### 12.3.2 建立变量表与 CPU 间的通信

在项目“Blocks”目录中，创建的变量表自动与 CPU 建立通信连接，如果使用冗余 CPU 或其他项目中建立的变量表监控变量时，有时与 CPU 不能建立通信连接，必须为变量表分配连接路径。在变量表中使用菜单命令“PLC”→“Connect to”连接相应的 CPU，变量表连接 CPU 有三种方式：

1）“Configured CPU”：PLC 站“Blocks”目录中下创建的变量表，连接路径自动建立，该种方式同样适合网络路由功能同样可以进行监控，也是变量表默认的连接路径。

2）“Direct CPU”：与编程电缆直接连接的 PLC 建立通信连接，如使用 MPI 电缆连接 CPU MPI。

3）“Accessible CPU”：可以选择与编程器中所有项目中的 CPU 建立连接。项目中自动带有路径信息。

### 12.3.3 在变量表中输入变量

打开变量表输入相应监控的变量，如图 12-5 所示。

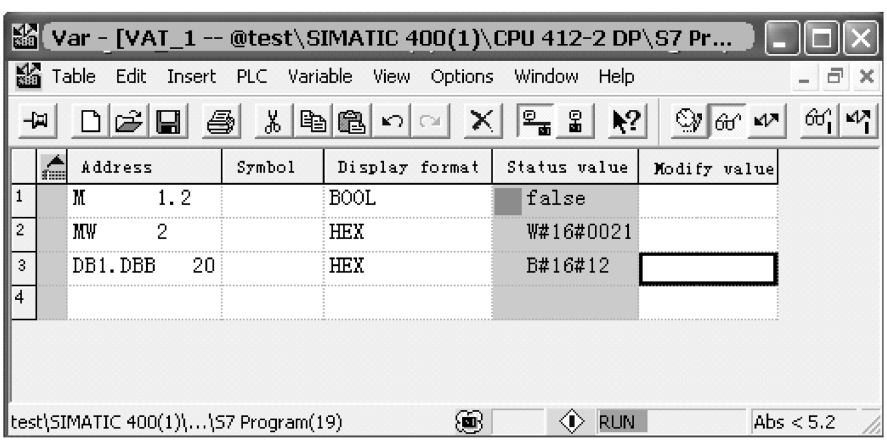



图 12-5 变量表

在“Address”栏中，输入需要监控的变量地址，如 I、Q、M 等地址区和数据类型，如果需要监控一个连续的地址范围，使用菜单命令“Insert”→“Range of Variables”打开对话框，输入变量起始地址、个数及显示的数据类型，确认后在“Address”栏中插入多个监控变量；在“Display format”栏中，按鼠标右键选择显示的类型，如十进制、十六进制、字符、浮点等格式，选择显示的格式与监控变量的数据类型有关。

**注意：**

过程映像区输出变量不能显示，例如PQW256，但是可以赋值。

### 12.3.4　变量的监控和修改

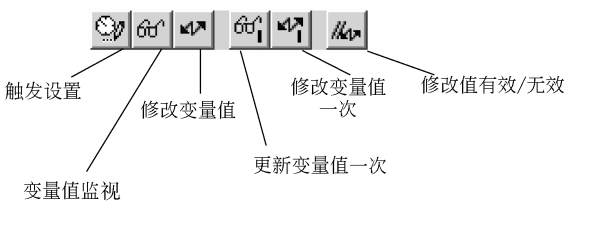



图12-6　变量表快捷键

通过工具栏中的快捷键，可以对变量表中的变量进行监视和修改，快捷键符号和命令含义如图12-6所示。

使用菜单命令“Variable”→“Trigger”或点击快捷键，打开监控与修改触发设置界面，如图12-7所示，监控和修改的触发点有三种选择：在CPU循环开始触发、在循环结束时触发或者在CPU由RUN转换为STOP状态时触发。触发条件可以选择只触发一次或者每个循环触发。触发条件的设置与快捷键和有关。

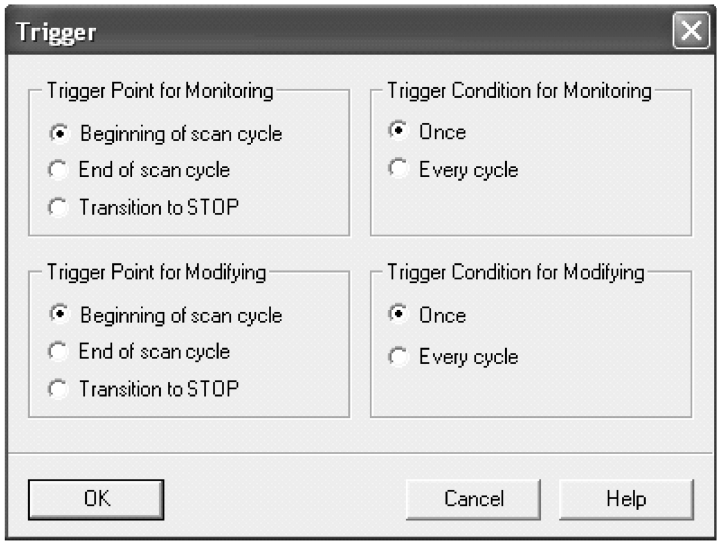



图12-7　监控与修改触发界面

（1）变量值监视

根据触发条件的设置，该按钮可以将每个CPU周期更新或只更新一次变量值。如果设置为每个CPU周期更新，按下该按钮后监控任务保持，再次按下该按钮则取消监视；如果设置为更新一次变量值，按下该按钮后立刻弹起，与按钮功能相同。被监视的变量值在变量表的Status一列中显示。

（2）更新变量值一次

按下一次该按钮，更新当前的变量值一次。

（3）修改变量值

按下该按钮，根据触发点的设置，每个CPU周期或只有一次将“Modify Value”一列中的值赋给变量。如果设置为每个CPU周期修改，按下该按钮后任务保持，再次按下该按钮则取消修改。如果设置为修改一次变量值，按下该按钮后立刻弹起，与按钮功能相同。

（4）修改变量值一次

按下一次该按钮，只将修改值赋给变量一次。

（5）取消修改值

当选中变量表中的一列时，按下这个按钮，会使该变量的修改值暂时失效，此时在 Modify Value 一列中，该变量值会以“//”符号方式作为注释，按下按钮或，带有“//”符号的值不赋值给变量，再次按下该按钮，可以使修改值重新生效。

**注意：**

只有在当前窗口中显示的变量才能被修改，如果有些变量不在视窗的显示范围之内，则修改动作对这些变量不起作用。变量值前面会出现图标，表示该变量的值并未修改。

### 12.3.5 强制变量

在程序调试过程中，由于一些输入信号不满足而不能对某个控制过程进行调试，强制（Force）功能可以让某些 I/O 保持用户指定的值，与修改变量不同，一旦强制了 I/O 的值，这些 I/O 将不受程序影响，始终保持该值，直到用户取消强制功能。

在程序编辑界面中使用菜单命令“PLC”→“Display Force Values”或者在变量表中通过菜单命令“Variable”→“Display Force Values”，可以打开强制变量窗口。一个 CPU 只能打开一个强制变量窗口。强制变量窗口与变量表界面类似，输入需要强制的变量地址和强制值，使用菜单命令“Variable”→“Force”启动强制命令。

强制功能由 CPU 提供，不具备强制功能的 CPU 无法在变量表中使用强制功能。使能强制功能后，CPU 前面板上强制指示灯（FRCE）变为黄色，提示强制功能可能导致的危险。

关闭强制窗口或退出“Monitoring and Modifying Variables”应用程序，并不能删除强制任务，强制任务只能使用菜单命令“Variable”→“Stop Forcing”删除或终止。

## 12.4 使用程序编辑器调试程序

### 12.4.1 调试 LAD/FBD 程序

LAD 或 FBD 程序以能流的方式传递信号状态，通过程序中线条、指令元素及参数的颜色和状态判断程序的运行结果，如图 12-8 所示。

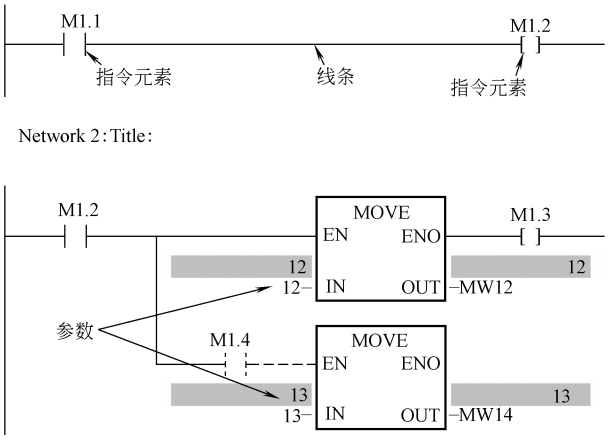



图 12-8 LAD 监控界面

线条颜色默认设置为：

- 状态满足：绿色实线
- 状态不满足：蓝色点线
- 状态未知：黑色实线

通过菜单命令“Options”→“Customize”打开用户定义对话框，在“LAD/FBD”标签栏中，可以改

变线条及指令元素的颜色。判断线条、指令元素及参数状态的规则如下：

1. 程序中线条的状态

- 线条的状态如果未知或没有完全运行，则是黑色的。
- 在能流开始处线条的状态总是满足的（“1”）。
- 并行分支开始处线条的状态总是满足的（“1”）。
- 如果一个指令元素和它前面的线条的状态都满足，则该元素后面的线条状态满足。
- 如果 NOT 指令前面的线条状态不满足（相反），则 NOT 指令后面的线条状态满足。
- 在下列情况下，线条交叉点后面的线条状态如果满足：
  ◇之前至少有一个线条的状态满足；
  ◇分支前的线条的状态满足。

2. 指令元素的状态

- 触点的状态
  ◇如果该地址为“1”，值则满足；
  ◇如果该地址为“0”，值则不满足；
  ◇如果该地址的值不知道，则为未知。
- 输出 Q 的元素状态对应于该触点状态。
- 如果 BR 位在调用后被置位，则调用（CALL）的状态满足。
- 如果跳转被执行，则跳转指令的状态满足，即意味着跳转条件满足。
- 带有使能输出（ENO）的元素，如果使能输出未被连接，则该元素显示为黑色。

3. 参数的状态

- 黑体类型的参数值是当前值。
- 细体类型的参数值来自前一个扫描，表明该程序区在当前扫描循环中未被处理。

在程序编辑界面中，点击工具栏按钮，即可进入监视状态，如图 12-8 所示，使用鼠标点击变量，按右键可以直接修改变量的值；同样按右键选择“Representation”可以切换显示的数据格式。

### 12.4.2 调试 STL 程序

STL 程序通过状态字及其他显示信息判断程序的运行结果，点击工具栏按钮，即可进入监视状态，如图 12-9 所示。

在 STL 监控界面右边的状态域中显示程序执行的状态及结果，可显示的信息包括：

- Status Bit（STA）—状态位；
- Result of Logical Operation（RLO）—逻辑操作结果位；
- Default Status（STANDARD）—标准状态，即累加器 1；
- Address Register 1（AR 1）—地址寄存器 1；
- Address Register 2（AR 2）—地址寄存器 2；
- Accumulator 2（ACCU 2）—累加器 2；
- DB Register 1（DB 1）—数据块寄存器 1；
- DB Register 2（DB 2）—数据块寄存器 2；
- Indirect —间接寻址存储器；

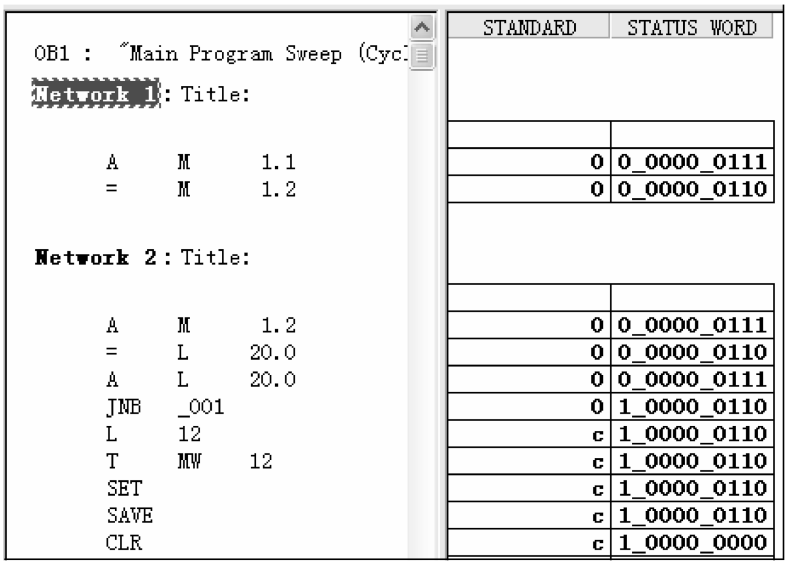



图 12-9 STL 监控界面

- Status Word —状态字。

选择显示的信息有两种方法：

1）通过菜单命令“Options”→“Customize”打开用户定义对话框，在“STL”标签栏中，选择需要在状态域中显示的信息。

2）点击状态域标题栏按右键，在弹出的对话框中使用“Hide”和“Show”命令选择状态域中需要显示的信息。点击“STANDARD”标题按右键，可以选择显示的数据格式。

### 12.4.3 使用断点单步调试程序

借助断点调试功能，可以一个指令一个指令地单步调试程序。是否支持断点功能以及支持断点的数目参考 CPU 订货手册。

实现单步调试程序必须满足下列条件：

1）使用菜单命令“Debug”→“Operation”，在弹出的对话框中，选择“Test operation”测试模式。

2）只有 STL 编程语言才能使用单步调试模式，使用菜单命令“View”→“STL”将 LAD 或 FBD 生成的块转换为 STL 编程语言。

3）程序块必须在线打开，在“SIMATIC Manager”界面下使用菜单命令“View”→“online”打开在线窗口及需要单步调试的程序块。

4）已打开的程序块不能在编辑器中进行编辑。

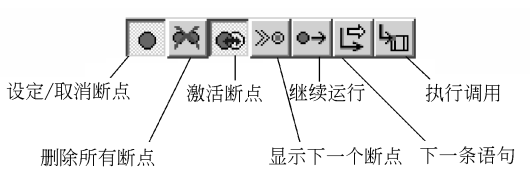



图 12-10 断点调试工具栏

通过菜单“View”→“Breakpoint Bar”显示断点调试工具栏，如图 12-10 所示。

工具栏的功能描述如下：

- 设定/取消断点　　—点击该按钮，在当前语句上设置或者取消断点；
- 删除所有断点　　—删除程序中所有断点；
- 激活断点　　—通过该按钮，可以让断点生效/失效；
- 显示下一个断点　—光标跳到下一个断点处；
- 继续运行　　—程序从当前断点处继续运行，直到遇到下一个断点；
- 下一条语句　　—该按钮可以实现单步运行，不进入块调用的内部；
- 执行调用　　—遇到块调用时，点击该按钮，可以进入块调用的内部执行。

在程序中设置断点，并点击激活断点功能，进入单步调试模式，“PLC Register contents”对话框自动显示，显示 PLC 内部寄存器的当前值。可以通过菜单命令“View”→“PLC Register”显示或关闭对话框。断点调试如图 12-11 所示，点击按钮或，以单步方式执行程序，黄色箭头指示当前指令执行的位置，同时在“PLC Register contents”对话框中显示指令执行后 CPU 寄存器的状态。断点调试与程序状态监控相比，可以更精确地观察每一条指令的执行情况。

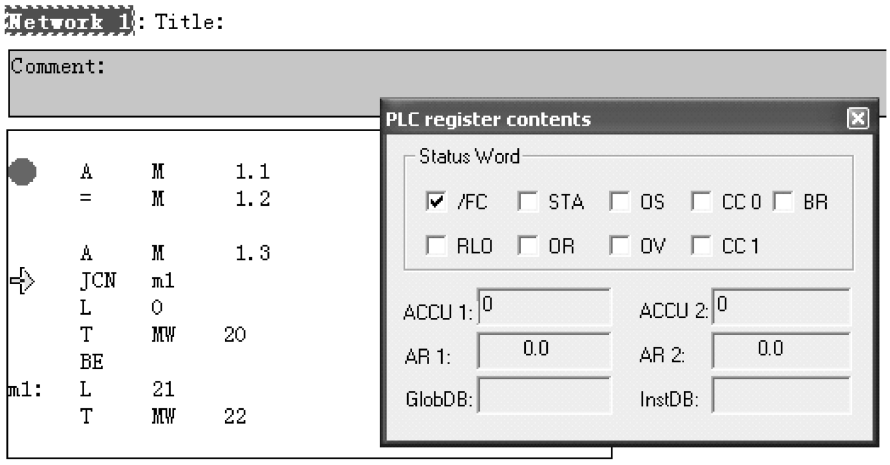



图 12-11　断点调试

激活断点调试功能后，CPU 进入“HOLD”模式（CPU 的前面板“RUN”灯闪烁，“STOP”亮）。在“HOLD”模式下 CPU 具有下列特性：

- 在 HOLD 模式下不处理 S7 指令码，没有优先级被进一步处理。
- 所有定时器被冻结：

◇不处理任何定时器；

◇所有监控时间停止；

◇时间控制的基本时钟被停止。

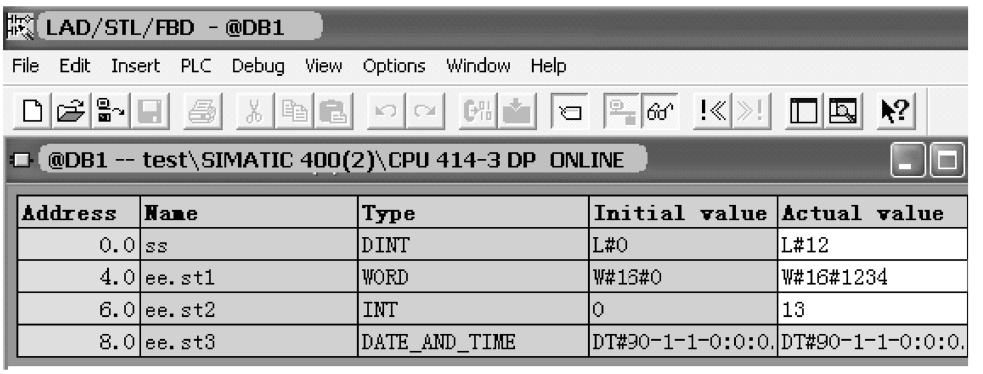
- 实时时钟继续运行。
- 为安全原因，在 HOLD 模式下输出被禁止。

点击删除所有断点键，并使激活断点键功能失效，点击继续运行键恢复 CPU

正常运行模式。

### 12.4.4 调试数据块

共享数据块和背景数据块中的数值可以在联机的模式下直接监控，使用菜单命令“View”→“data view”切换到数据视窗中，点击按钮，数值显示在“Actual Value”栏中，如图 12-12 所示。




| Address | Name | Type | Initial value | Actual value |
|---|---|---|---|---|
| 0.0 | ss | DINT | L#0 | L#12 |
| 4.0 | ee.st1 | WORD | W#16#0 | W#16#1234 |
| 6.0 | ee.st2 | INT | 0 | 13 |
| 8.0 | ee.st3 | DATE_AND_TIME | DT#90-1-1-0:0:0. | DT#90-1-1-0:0:0. |

图 12-12 数据块监控界面

在监控过程中，“data view”和“Declaration”视窗不能切换，数据块状态栏中将显示一个绿框。数值分别以各自的数据类型显示，其格式不能修改。结束监控状态后，“Actual Value”栏将显示数据块之前的有效内容，不能将更新的在线数值上传至离线数据块中。

如果需要修改数据块中的数值，结束监控状态，在“Actual Value”栏中键入修改的数值，点击键下载整个数据块。

数据块中有些数据类型不能监控和更新，在“Actual Value”栏中，不能更新的数据区域将显示为灰色背景：

- 复合数据类型 DATE_AND_TIME 和 STRING 不能更新，如图 12-12。
- 在复合数据类型 ARRAY、STRUCT、UDT、FB 和 SFB 中，只有基本数据类型元素才能被更新。
- 在背景数据块的 IN、OUT 声明中，复合数据类型以指针数据类型显示，不显示自身数据类型元素，指针数据类型不能更新。
- 参数类型不能更新。

在变量表中输入数据块内变量的绝对地址，同样可以对数据块内的数据进行监视和修改。

## 12.5 硬件诊断

### 12.5.1 硬件的诊断符号

在硬件配置窗口，使用菜单命令“Station”→“Open Online”，打开硬件配置在线窗

口。通过在线窗口可以快速查看配置的硬件模块是否有故障，如果模块没有故障，则在模板图标上不显示诊断符号；如果模板有诊断信息，则会在模板的图标上增加一个诊断符号或显示的模板符号对比度降低。模块的诊断符号与含义见表12-1。

**表12-1　硬件诊断符号**

| 诊断符号 | 含　义 |
|---|---|
|  | 硬件配置与实际不匹配、配置的模板不存在或者插入了不同类型的模板。如果是PROFIBUS－DP从站，表示从站不存在 |
|  | 故障：模板出现故障<br>可能的原因：诊断中断，I/O访问错误或检查到故障LED<br>如果是PROFIBUS-DP从站，表示从站中某个模块故障 |
|  | 不能进行诊断。原因：无在线连接或该模块不支持模板诊断功能（如：电源或子模板） |

CPU的信息也可以通过诊断符号快速诊断，诊断符号见表12-2。

**表12-2　CPU诊断符号**

| 诊断符号 | 含　义 |
|---|---|
|  | 起动（STARTUP）模式 |
|  | 停机（STOP）模式 |
|  | 停机（STOP）模式<br>在多CPU操作模式下由其他CPU触发的停机 |
|  | 运行（RUN）模式 |
|  | 保持（HOLD）模式 |
|  | 有变量被强制，即在用户程序中有变量被赋予一个固定值，该数据值不能被程序改变<br>强制符号可以与其他符号组合显示（这里是与运行（RUN）模式符号一起显示） |

## 12.5.2　模板诊断信息

在硬件配置在线窗口中，双击具有诊断功能的模块，如CPU、模拟量模块、具有诊断功能的数字量输入、PROFIBUS-DP从站接口模块等弹出模块的在线诊断信息，不同模块显示的诊断信息不同，例如CPU显示的模板信息，如图12-13所示。

在模板信息中，不同标签栏对应不同的诊断信息和用途，模板信息包含的内容见表12-3。

**表12-3　模板信息包含的内容**

| 功能/标签 | 信　息 | 用　途 |
|---|---|---|
| 概述 | 所选模板的标识数据，例如订货号、版本号、状态、机架中的插槽 | 可将所插模板的在线信息与配置数据进行比较 |

（续）

| 功能/标签 | 信　息 | 用　途 |
| --- | --- | --- |
| 诊断缓存区 | 诊断缓存区中事件总览以及所选中事件的详细信息 | 查找引起 CPU 进入 STOP 模式的原因，使用诊断缓存区可以对系统错误进行分析，查找停机原因并对出现的每个诊断事件进行追踪和分类 |
| 诊断中断 | 所选模板的诊断数据 | 评估模板故障的原因 |
| DP 从站诊断 | 所选 DP 从站（参照 EN50170）的诊断数据 | 评估 DP 从站中的故障原因 |
| 存储器 | 存储能力。所选 CPU 工作存储器和装载存储器以及具有保持功能的存储器当前使用的情况 | 检查 CPU/功能模板的装载存储器中是否有足够的空间，或者需要压缩存储器内容 |
| 扫描循环时间 | 所选 CPU 最长、最短和最近一次的循环扫描时间 | 用于检查配置的最小循环时间、最大循环时间和当前循环时间 |
| 时间系统 | 当前时间、操作小时数以及同步时钟的信息 | 显示模板的时间和日期，检查时间同步信息 |
| 性能数据 | 所选模板（CPU/FM）的地址区和可使用的程序块 | 在生成用户程序之前或之中，检查 CPU 是否满足执行用户程序的要求。例如，装载存储区的大小或过程映像的大小 |
| 可用程序块（“Performance Data”标签中） | 显示所选 CPU 中所有可用的程序块类型，包括 OB、SFB、SFC 列表，可将这些程序块用于该 CPU | 检查用户程序中包含或调用的 OB、SFB、SFC 是否可以在所选的 CPU 上运行 |
| 通信 | 传送速率，通信连接概述，通信负载以及所选模板在通信总线上最大的信息容量 | CPU 或 FM 的连接数量和种类，以及正在使用的数量 |
| 堆栈 | 堆栈标签：只能在 STOP 模式或 HOLD 模式下调用<br>显示所选模板的 B（块）堆栈、I（中断）堆栈、L（局域）堆栈以及嵌套堆栈。可以跳转到中断块的故障点 | 可判明引起停机的原因，并对程序块进行修改 |

如果模板出现故障，模板信息对于用户来说至关重要，提供的诊断信息可以帮助用户快速定位故障原因，并找到解决方法，例如 CP341 MODBUS 通信，如果通信指示灯闪烁，但是没有接收到数据，CP341 的模板信息会显示出错的原因和错误代码，在手册中可以查找错误的原因和解决方法。

对于程序的调试、维护，CPU 的诊断缓存区非常重要，在诊断缓冲区中列出所有与 CPU 故障相关的触发事件，例如模板拔出的时间（或故障时间）和插入时间（更换时间）、程序执行故障点、CPU 模式切换记录等。利用诊断缓存区可以跟踪程序执行的故障信息，如图 12-14 所示。

选择缓存区中的故障信息，点击“Open Block”键，直接跳转到程序故障点。在程序中使用 M、C、T 数据区超出 CPU 系统存储器规定的范围，程序下载后，CPU 报错，使用相同的方法可以查出非法 M、C、T 数据区在程序中使用的位置。

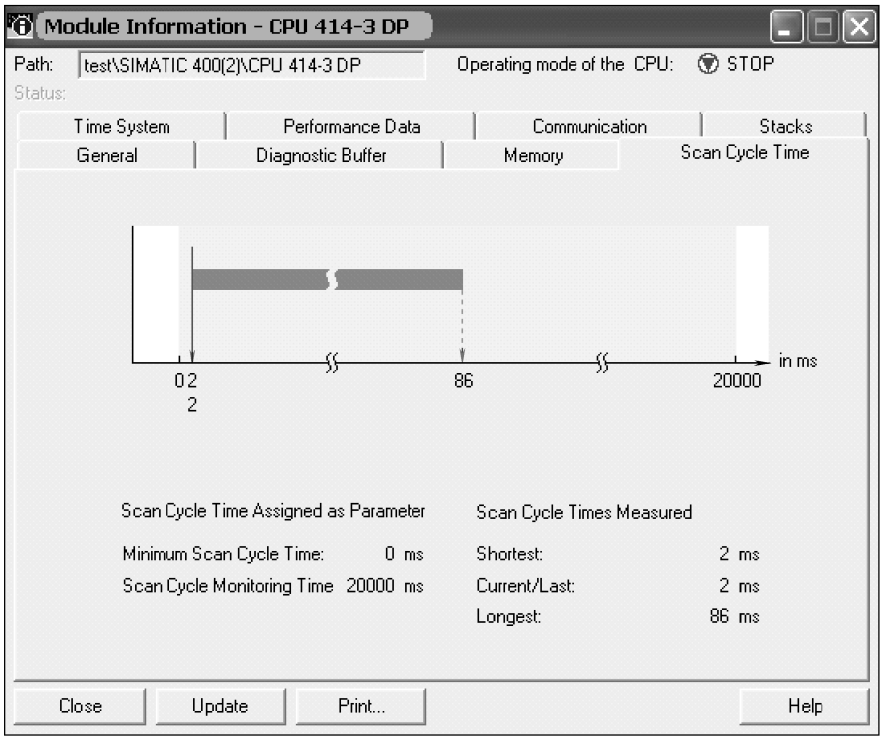



图 12-13　CPU 的模板信息

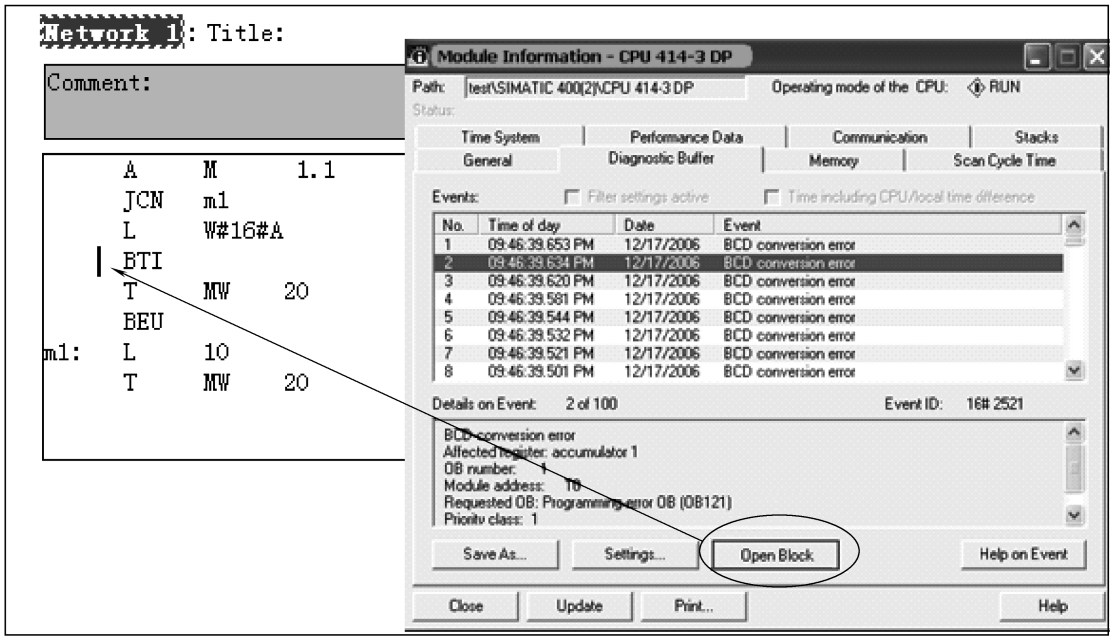



图 12-14　CPU 的诊断缓存区

## 12.6 使用模拟器 S7 PLCSIM 测试用户程序

安装可选软件包 PLC Simulation（PLC 模拟器），在编程器上可以直接模拟 S7-300、S7-400 系列 PLC 的运行，并测试程序。PLC 模拟器完全由软件实现，不需要任何 S7 硬件，例如 CPU 或信号模板。

PLC 模拟器只能对用户程序进行测试或由用户程序引起的故障进行诊断，由于没有连接硬件，不能模拟硬件的诊断、串口模块通信程序、FM 模块的调试操作、冗余系统的操作及 PLC 站点间的通信。

PLC 模拟器为监视和修改用户程序提供了一个简单的用户界面，例如切换输入开关接通和断开等功能，也可以使用 STEP7 中各种调试工具，例如断点调试、程序监控及使用变量表监视和修改变量。

软件安装后，在“SIMATIC Manager”界面中，点击工具栏上的按钮，或使用菜单命令“Options”→“Simulate Modules”，进入 PLCSIM 操作界面，如图 12-15 所示。

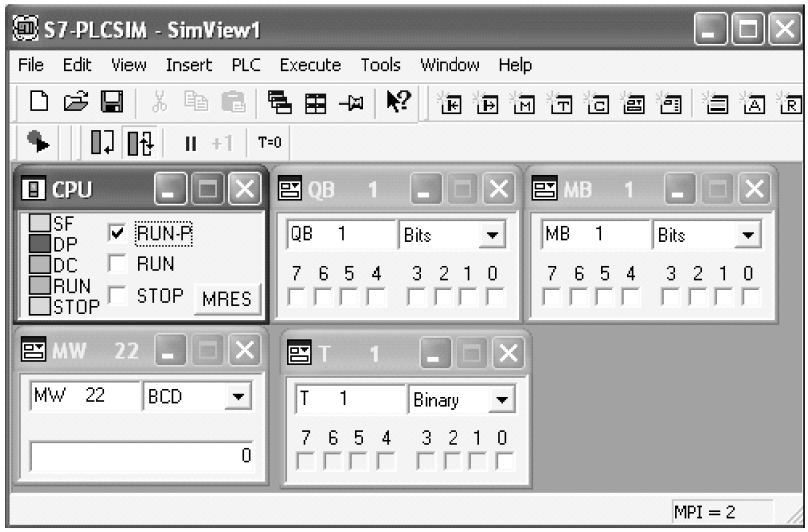



图 12-15 PLCSIM 界面

### 12.6.1 设置 PLC 模拟器通信接口

在“PLCSIM”界面中，点击菜单“PLC”→“MPI Address”设定 PLC 模拟器的 MPI 地址（旧版本只支持 MPI 网络，V5.3 以上版本支持其他网络），设定的 MPI 地址必须与硬件配置中 CPU 的 MPI 地址相同（默认地址为 2）；在“Set PG/PC Interface”中，选择“S7ONLINE（STEP7）”指向 MPI 适配器或网卡，例如 PC Adapter（MPI），设置完成后可以将用户程序下载到 PLC 模拟器中进行测试。

### 12.6.2 设置 CPU 的操作模式

在 CPU 栏中，可以点击“RUN－P”、“RUN”、“STOP”及“MRES”选项或通过菜单

命令“Execute”→“Key Switch Position”模拟CPU的运行模式；使用菜单命令“Execute”→“Startup Switch Position”可以模拟CPU启动模式，如冷启动、暖启动等特性；使用菜单命令“Execute”→“Scan Mode”，可以选择CPU执行单次扫描还是循环扫描，扫描模式也可以通过工具栏选择，工具栏按钮及含义如图12-16所示。

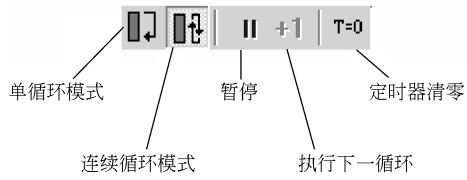



图12-16　CPU的扫描模式

在单循环模式下，可以点击按钮 +1 进行下一次循环。在任意模式下，都可以通过 II 暂停程序的执行。

### 12.6.3　触发中断

通过触发相应的OB，对中断程序进行测试，使用菜单命令“Execute”→“Trigger Error OB”选择相应的中断OB，不同OB中可选择的触发事件不同，例如可以选择触发诊断中断OB82的事件类型，如图12-17所示。

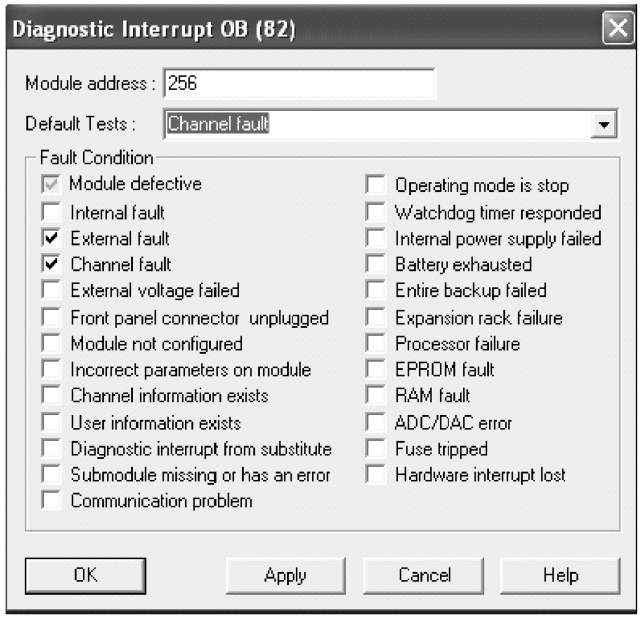



图12-17　可触发OB82的事件

选择引起中断触发的模块地址、故障类型（可以选择软件定义的故障类型，如外部电压故障等，每个故障类型中可能包含多个触发事件），点击“Apply”键，触发中断，执行中断程序一次，用于中断程序的测试。

### 12.6.4　回放功能

PLCSIM具有回放功能，可以记录信号的连锁反应，通过回放记录，可以查看连锁反应的正确性，通过菜单命令“Tools”→“Record/Playback”，或点击回放按钮，弹出回放对

话框，如图 12-18 所示。

点击●键记录程序的连锁响应，记录过程中可以手动模拟输入、输出及 M 等信号状态，完成后可以将记录过程保存为一个文本文件。点击▶键回放记录的事件，通过■▾键可以调节回放的速度。

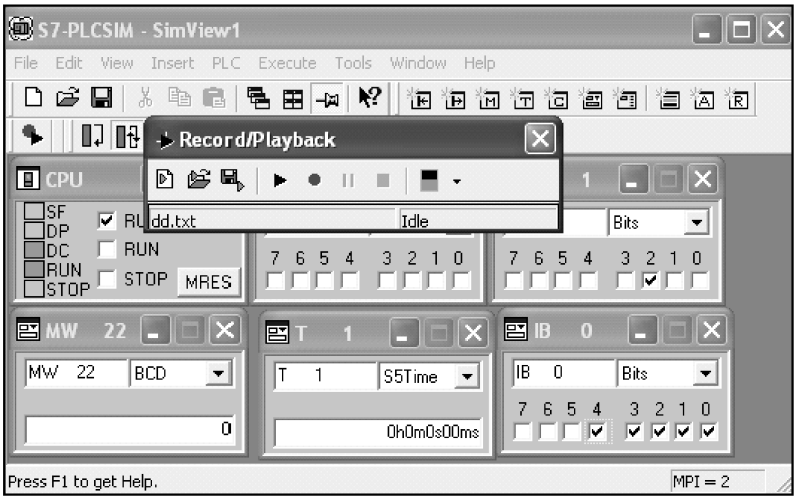



图 12-18　PLCSIM 回放功能

# 第 13 章　打印和归档程序

## 13.1　打印项目文档

程序编辑和调试完成之后，需要打印成文档保存。STEP7 中集成的打印功能可以将所有重要数据打印出来，作为项目的参考文档。

可以从 SIMATIC Manager 中直接打印选定的内容，也可以打开相关的对象进入视图中打印选定的内容。

在 SIMATIC Manager 中，可以打印下列内容：

- 选择对象的树型图（项目或库的结构）；
- 选择目录下的对象清单；
- 选择对象的内容；
- 消息。

在打开相关的对象中，可以打印下列内容：

- 以梯形图、语句表或功能图编写的程序块，也可以是其他语言（选件）编写的程序块；
- 带有符号名称和绝对地址的符号表；
- 硬件配置信息，包括 PLC 中模块的排列及模块参数；
- 诊断缓冲器区的内容；
- 变量表，包括监控的数据格式及监控值和更改值；
- 交叉参考表，例如：交叉参考、赋值表、程序结构、未使用地址表、无符号地址表；
- 全局数据（GD 通信）；
- 带有模块状态的模块信息；
- 操作员相关文本（用户文本和文本库）；
- 可选软件包的文档，例如其他编程语言。

可选软件包 DOCPRO，能够生成、编辑和打印标准化的接线手册。该软件包可生成符合 DIN 和 ANSI 标准的设备文档。

在打印对象界面中，使用菜单命令“File”（根据相应的打印对象，在符号表中为“Symbol Table”）→“Print”进入“Print”对话框，如果需要可以修改打印选项（打印机，打印范围，份数等），然后进行打印。

有些打印对象可以进行其他功能的打印设置，见表 13-1。

**表 13-1　不同对象的打印功能**

| 打印对象 | 菜单命令 | 打印功能 | | |
|---|---|---|---|---|
| | | 打印预览 | 页面设定<br>“纸张大小” | 打印设定<br>“页眉和页脚” |
| 程序块、STL 源文件 | File > * | • | • | • |

（续）

| 打印对象 | 菜单命令 | 打印功能 | | |
|---|---|---|---|---|
| 模板信息 | | – | • | • |
| 全局数据表 | GD Table > * | • | • | • |
| 硬件配置信息 | Station > * | • | • | • |
| 对象、对象文件夹 | File > * | – | • | • |
| 参考数据 | Reference Data > * | • | • | • |
| 符号表 | Symbol Table > * | • | • | • |
| 变量表 | Table > * | – | • | • |
| 连接表 | Network > * | • | • | • |
| 操作员相关文本（用户文本，文本库） | Texts > * | • | • | • |

*：通配符，表示菜单命令中的相关功能（例如打印预览或页面设定）。

•：提供该项功能。

–：不能使用该项功能。

## 13.2 程序归档

在 SIMATIC Manager 中，可以对选择的程序进行归档。归档应用程序 ARJ 和 PKZIP 4.0 作为 SIMATIC Manager 的组成部分已包含在软件包内，在…\ Step7 \ S7bin \ 路径下，可以找到这些程序及其使用说明。

SIMATIC Manager 支持下列版本的归档程序：

- PKZip Command line V4.0（包含在 STEP7 中）；
- WinZip 版本 6.0 以上；
- JAR 版本 1.02 以上；
- ARJ V2.4.1a（只适合于可恢复的归档，包含在 STEP7 中）；
- ARJ32 V3.x（只适合于可恢复的归档）；
- LHArc 版本 2.13 以上（只适合于可恢复的归档）。

如果使用没有包含在 STEP7 中的归档程序，必须在其中进行设置，在“SIMATIC Manager”中，使用菜单命令“Options”→“Customize”弹出设置对话框，在“Archiving”标签中，点击“Configuring”为选择的归档程序设置程序路径。

有三种方法可以对程序进行归档：

1）在“SIMATIC Manager”中使用菜单命令“File”→“Save As”，将当前的程序另存为其他路径下。归档的项目可以使用菜单命令“File”→“Open”直接打开。

2）查看项目存储路径（点击“Options”→“Customize”对话框中的“General”标签），将项目复制到其他路径下。复制过程中，在“SIMATIC Manager”中必须关闭该项目所有的窗口，否则项目复制不完整。归档的项目可以使用菜单命令“File”→“Open”直接打开。

3）在“SIMATIC Manager”中使用菜单命令“File”→“Archive”，将选择的项目压缩

归档于其他路径下，在压缩过程中，在“SIMATIC Manager”中必须关闭该项目所有的窗口，否则压缩归档的项目文件不完整。与前两种归档方式相比，压缩归档占用存储空间小，在归档方式中建议使用压缩归档存储项目。使用菜单命令“File”→“Retrieve”，可以将压缩归档的项目解压缩，根据弹出的提示对话框，可以将解压缩项目存储于指定的路径下，并在当前界面中打开。

# 附录　寻求帮助

如果在编程、调试以及设备维护过程中遇到有关硬件及编程问题时，可以通过在线帮助文档、手册、拨打热线电话和网站支持的方式寻求帮助。

## 1. 在线帮助系统

在线帮助系统提供给用户有效快速的信息，无需查阅手册，在线帮助具有如下信息方式：

- 显示帮助信息的号码。
- 首先用鼠标选中或在对话框或窗口选择某一对象，然后使用 F1 键得到相应的帮助信息。
- 对某种功能的使用、主要特性及功能范围做一个简要说明。
- 某些功能的快速入门。
- 在线帮助中对查找特殊信息的方法提供描述。
- 提供有关当前版本的信息。

可以使用下列方法访问在线帮助系统：

- 在不同界面中，选择“Help”菜单栏，从中选择相关文档。
- 在某些对话框中，单击“Help”键，则可以显示该对话框的帮助信息。
- 使用鼠标选择希望得到帮助的窗口或对话框，而后按 F1 键，弹出该窗口或对话框的帮助信息。
- 使用鼠标点击工具栏中的键，然后再点击希望得到帮助的窗口或对话框得到相应的帮助信息。

## 2. 相关手册和示例程序

所有安装的软件都包含相关内容的 PDF 格式手册和示例程序。使用菜单命令“start”→“SIMATIC”→“Documentation”→“English”，可以打开手册存储的路径，查看相关的文档；在“SIMATIC Manager”中使用菜单命令“File”→“Open”或工具栏快捷键打开项目存储对话框，在“Sample projects”标签中，列出所有的示例程序。

## 3. 热线服务系统

如果通过在线帮助和相关文档不能解决遇到的问题，可以拨打西门子公司的技术支持与服务热线 + 86 – 800 – 810 – 4288（如果不支持 800 电话，可以拨打电话 + 86 – 10 – 64719990）或发送传真 + 86 – 010 – 64719991 与西门子公司联系，也可以通过电子邮件：

8008104288. cn@ siemens. com，将现场照片以及故障记录信息发送过来。热线服务内容，包括低压电气与 PLC 系统的技术支持、产品咨询与售后服务。

软件授权维修服务可拨打亚太服务热线 + 86 - 10 - 64757575 或发送传真 + 86 - 10 - 64747474 与西门子公司联系，相关文本信息可以通过电子邮件：support. asia. automation@ siemens. com 发送。

## 4. 网站支持

登录西门子公司技术支持与服务主页 www. ad. siemens. com. cn/service，可以下载相关产品手册、驱动软件以及 FAQ（常见问题解答）。网上课堂栏目中，包括热线工程师根据多年的技术支持经验，撰写的产品入门指导课程数百篇！覆盖 PLC、通信/网络、冗余、人机界面、驱动、传动、低压、数控、过程自动化等众多产品线。快速入门指导是产品手册很好的补充材料，可以自由地下载。附带光盘“网上下载”目录中摘录了一些与 PLC 编程应用相关的资料。在技术论坛栏目中，可以留言、分享、讨论、交流经验，是专家级用户的在线交流圈，产品初学者的充电线。通过西门子公司的网站可以时时得到无极限的支持，了解西门子公司产品最新的动态信息及相关的活动。

# 缩 写 表

| 缩写英文全称 | 中文注释 |
| --- | --- |
| ASI Actuator – Sensor Interface | 执行器 – 传感器接口。用于执行器 – 传感器分散于机器或工厂内的场合。符合标准 EN 50295 |
| CPU Central Processing Unit | 中央处理单元 |
| CFC Continuous Function Chart | 连续功能图 |
| CP Communication Processor | 通信处理器 |
| CIR Configuration In Run | 在运行中对硬件进行配置 |
| CBA Component Based Automation | 基于组件的自动化 |
| DIN | 德国标准 |
| DCP Detect Configuration Protocol | 侦测配置协议 |
| ERP Enterprise Resource Planning | 企业资源计划系统 |
| EIB European Installation Bus | 楼宇自动化标准（EN 50090，ANSI EIA 776）。在楼宇自动化系统中应用总线技术，只用一根通用的电缆就能控制、监视和报告所有的运行功能和状态 |
| FM Function Module | 功能模块 |
| FBD Function Block Diagram | 功能块图编程语言 |
| FLASH Flash File System | 程序存储卡，具有掉电保持功能一种数据存储器，主要用于 S7 – 400 系列 PLC |
| FDL Fieldbus Data Link | 现场总线数据链路——PROFIBUS 协议第 2 层，也是 ISO 参考模型的第 2 层。现场总线数据链路由现场总线链路控制（FCL）和存取介质（MAC）组成 |
| HMI Human Machine Interface | 人–机接口 |
| ITP Industry Twist Pair | 工业双绞线 |
| IE FC TP Industry Ethernet Fast Connection Twist Pair | 工业 以太网快速连接双绞线 |
| ISO Transport | 使用 ISO 标准的通信协议 |
| ISO – on – TCP | 使用 ISO – on – TCP 标准的通信协议，具有网络路由功能 |
| LAD Ladder Logic | 梯形图编程语言 |
| MES Manufacture Execute System | 制造执行系统 |
| MDI Manual Data Input | 手动数据输入 |
| MMC Micro Memory Card | 程序存储卡，具有掉电保持功能，主要用于 S7-300 系列 PLC |
| OB Organization Block | 组织块 |
| OLM Optical Link Module | 光链路模块 |
| PLC Programmable Logic Controller | 可编程序控制器 |
| PROFIBUS PROcess Field BUS | 过程现场总线。符合现场总线国际标准和欧洲过程现场总线系统标准（IEC 61158/EN50170 V. 2），可提供功能强大的过程和现场通信，适合于自动化工厂中单元级和现场级符合 PROFIBUS 标准的自动化系统和现场设备的数据通信网络。PROFIBUS 可以使用通信协议 FMS、 |

| | |
|---|---|
| | DP、PA 进行通信 |
| PROFINET | 由 PROFIBUS 国际组织（PROFIBUS International PI）推出，是新一代基于工业以太网技术的自动化总线标准 |
| PS Power Supply | 电源 |
| PIQ Process Image Qutput | 过程映像输出 |
| PII Process Image Input | 过程映像输入 |
| PID Proportional Integral Derivative | 比例、积分、微分控制 |
| POF Polymer Optical Fiber | 塑料光纤 |
| PCF Polymer Cladded Fiber | 塑料包层光纤 |
| PG/OP Programming Gevice/Operation Panel | 编程器/操作面板 |
| RACK Rack | 机架 |
| RAM Random Access Memory | 数据存储器 |
| STL StaTement List | 语句表编程语言 |
| SM Signal Module | 信号模块 |
| SSI Synchronous Serial Interface | 同步串行接口，这里指绝对值编码器 |
| SFC System Function | 系统函数 |
| SFB System Function Block | 系统函数块 |
| TIA Totally Integrated Automation | 全集成自动化 |
| T $\overline{\text{U}}$V | （德语）技术监督协会 |
| TCP/IP Transmission Control Protocol/Internet Protocol | 用于网络的一组标准通信协议 |
| UR Universal Rack | 通用机架 |
| UDT User – Defined data Types | 用户定义数据类型 |
| UDT User – Defined data Types | 用户定义数据类型 |
| UDP User Datagram Protocol | 用户数据报协议 |
| WDS Wireless Distribution System | 无线分布系统 |

# 参 考 文 献


1. 西门子公司《STEP 7 – Programming with STEP 7》
2. 西门子公司《STEP 7 – System and Standard Functions for S7 – 300 and S7 – 400》
3. 西门子公司《FM 350 – 1 – Counting Function Module》
4. 西门子公司《FM352 – 5 – Manual》
5. 西门子公司《Manual FM354》
6. 西门子公司《FM 355 – PID Control》
7. 西门子公司《Technological Functions ET 200S Manual》
8. 崔坚，李佳. 西门子工业网络通信指南［M］. 北京：机械工业出版社，2004.
9. 西门子公司《CPU 31xC Technological Functions Manual》

| 书　名 | 书号 | 定价 |
| --- | --- | --- |
| 西门子工业网络通信指南（上册） | 15177 | 49.00 |
| 西门子工业网络通信指南（下册） | 16508 | 58.00 |
| 西门子S7可编程序控制器—STEP7编程指南（第2版） | 28718 | 49.00 |

上架指导：工业技术／自动化技术

地址：北京市百万庄大街22号　邮政编码：100037
电话服务　网络服务
社服务中心：(010)88361066　门户网：http://www.cmpbook.com
销售一部：(010)68326294　教材网：http://www.cmpedu.com
销售二部：(010)88379649
读者服务部：(010)68993821
封面无防伪标均为盗版

ISBN 978-7-111-28718-6
ISBN 978-7-89451-415-8（光盘）
策划：林春泉／封面设计：鞠杨
定价：66.00元（含1CD）

