

电气信息工程丛书

西门子(中国)有限公司重点推荐图书

SIEMENS

# S7-1200 PLC 编程及应用

廖常初 主编



- 国内第一本全面介绍西门子新一代小型 PLC 的图书
- 根据中文版软件全面改写



赠送超值 DVD 光盘：

- 西门子(中国)有限公司授权的编程软件 STEP 7 Basic 中文版和用户手册，与正文配套的例程和视频教程



ISBN 978-7-111-29128-2

◎ 策划 时静  
◎ 封面设计 旭洲企划 刘吉维

## 电气信息工程丛书

- ◇ PLC 编程及应用（第3版）
- ◇ S7-300/400 PLC 应用技术（第2版）
- ◆ S7-1200 PLC 编程及应用
- ◇ 西门子 S7-200 PLC 编程及应用案例精选
- ◇ 西门子人机界面(触摸屏)组态与应用技术
- ◇ 西门子工业通信网络组态编程与故障诊断
- ◇ CS/CJ 系列 PLC 应用基础及案例
- ◇ 欧姆龙 CP1H PLC 应用基础与编程实践
- ◇ IEC 61131-3 编程语言及应用基础
- ◇ Protel 99 SE 原理图与 PCB 设计及电路仿真
- ◇ Protel 99SE 实战详解与技巧
- ◇ Protel 2004 电路原理图及 PCB 设计
- ◇ 基于 Altium Designer 的原理图与 PCB 设计
- ◇ 基于 Allegro 的 PCB 设计与开发
- ◇ VHDL 数字电路及系统设计
- ◇ OrCAD 电路原理图设计与应用
- ◇ 单片机应用及 C51 程序设计
- ◇ 51 单片机快速上手
- ◇ 数字信号处理器原理、结构及应用基础——TMS320F28x
- ◇ Freescale 9S12 十六位单片机原理及嵌入式开发技术
- ◇ 基于 EDK 的 FPGA 嵌入式系统开发
- ◇ 嵌入式可配置实时操作系统 eCos 开发与应用（第2版）
- ◇ Linux PowerPC 详解——核心篇
- ◇ 典型数控系统应用技术（FANUC 篇）

上架建议：工业技术 / 电气

地址：北京市百万庄大街22号

电话服务

社服务中心：(010)83361066

销售一部：(010)83326284

销售二部：(010)83379649

读者服务部：(010)68993821

邮购编码：100037

网路服务

门户网：<http://www.cmpbook.com>

教科网：<http://www.cmpedu.com>

封面无防伪标均为盗版

ISBN 978-7-111-29128-2



9 787111 291282 >

定价：36.00元（含1DVD）

电气信息工程丛书

S7-1200 PLC 编程及应用

廖當初 主編



机械工业出版社

本书介绍了西门子公司新一代小型 PLC S7-1200 的硬件组成、硬件和网络组态的方法、指令系统、用户程序结构、高速输入/高速输出、通信功能、精简系列面板的组态与应用、PID 闭环控制的组态与调试、故障诊断与提高 PLC 控制系统可靠性的措施，还介绍了一整套数字量控制系统梯形图的先进完整的设计方法，这些方法易学易用，可以节约大量的设计时间。

本书具有很强的可操作性，通过大量的例程，深入浅出地介绍了 S7-1200 指令的应用、程序结构和编程方法，详细介绍了用 STEP 7 Basic 完成各种任务的操作过程。读者一边看书一边进行操作，可以较快地掌握 STEP 7 Basic、S7-1200 和精简系列面板的使用方法。

随书光盘附有 S7-1200 的编程软件 STEP 7 Basic、S7-1200 与精简系列面板的用户手册和产品样本，以及作者编写的与正文配套的例程。

本书可作为高校机电类各专业学生学习 S7-1200 的教材，也可供广大工程技术人员参考。

### 图书在版编目 (CIP) 数据

S7-1200 PLC 编程及应用/廖常初主编. —北京：机械工业出版社，2009.12  
(电气信息工程丛书)  
ISBN 978-7-111-29128-2

I . S… II . 廖… III . 可编程序控制器—程序设计 IV . TM571.6

中国版本图书馆 CIP 数据核字 (2009) 第 216609 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：时 静

责任编辑：李馨馨

责任印制：李 妍

北京振兴源印务有限公司印刷

2010 年 1 月第 1 版 · 第 1 次印刷

184mm × 260mm · 15.5 印张 · 385 千字

0001—3500 册

标准书号：ISBN 978-7-111-29128-2

ISBN 978-7-89451-309-0 (光盘)

定价：36.00 元 (含 1DVD)

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

封面无防伪标均为盗版

读者服务部：(010) 68993821

# 前　　言

S7-1200 是西门子公司最新推出的小型 PLC，代表了下一代 PLC 的发展方向。集成的以太网接口用于与计算机、人机界面和其他 PLC 的通信。CPU 集成了很强的工艺功能，可以用插入 CPU 的信号板增加 I/O 点。编程软件 STEP 7 Basic 集成了用于精简系列面板组态的 WinCC Basic。STEP 7 Basic 采用多窗口的界面，硬件和网络的组态、编程和监控均采用图形化的方式。软件具有较高的智能，使用方便，容易入门。S7-1200 可以用多种硬件和软件的方法来诊断和显示故障。

本书具有很强的可操作性，通过大量的例程，深入浅出地介绍了 S7-1200 指令的应用、程序结构和编程方法。详细地介绍了用 STEP 7 Basic 完成各种任务的操作过程，读者一边看书一边进行操作，可以较快地掌握 STEP 7 Basic 和 S7-1200 的使用方法。

本书的前两章介绍了 S7-1200 的硬件组成，STEP 7 Basic 的安装、硬件和网络组态的方法。第 3 章详细介绍了程序编辑器的使用方法，程序的生成、下载和监控的方法，还介绍了用于数字量控制的基本指令。第 4 章介绍了设计数字量控制梯形图的一整套先进完整的方法，这些方法易学易用，可以节约大量的设计时间。第 5 章介绍了 S7-1200 的指令和高速输入/高速输出的使用方法。第 6 章介绍了 S7-1200 的用户程序结构和获取程序信息的方法。第 7 章介绍了 S7-1200 的通信功能、通信程序的设计方法和故障诊断的方法。第 8 章介绍了精简系列面板的组态与应用。第 9 章介绍了 PLC 控制系统的设计与调试步骤，PLC 控制系统的可靠性措施，以及 PID 闭环控制的组态和调试的方法。

随书光盘附有 S7-1200 的编程软件 STEP 7 Basic、S7-1200 与精简系列面板的用户手册和产品样本，以及作者编写的与正文配套的例程。

本书的编写得到了西门子公司的硬件和软件的支持，西门子(中国)有限公司的李冰冰先生、许艳婷女士对本书的编写提供了很大的帮助，谨在此表示衷心的感谢。

本书由廖常初主编，范占华、关朝旺、余秋霞、陈曾汉、陈晓东、王云杰、李远树、廖亮、孙明渝、左源洁、万莉、郑群英、孙剑、唐世友参加了编写工作。

因作者水平有限，书中难免有错漏之处，恳请读者批评指正。

作者 E-mail：liaosun@cqu.edu.cn。

重庆大学 廖常初

2009 年 10 月

# 目 录

前言	
第 1 章 概述	1
1.1 PLC 的基本概念	1
1.1.1 PLC 的基本结构	1
1.1.2 PLC 的特点	3
1.1.3 PLC 的应用领域	4
1.1.4 怎样下载西门子 PLC 的资料和软件	4
1.2 S7-1200 的程序结构与工作原理	5
1.2.1 逻辑运算	5
1.2.2 S7-1200 用户程序结构简介	6
1.2.3 PLC 的工作原理	9
第 2 章 PLC 的硬件与硬件组态	12
2.1 S7-1200 的硬件	12
2.1.1 CPU 模块	12
2.1.2 信号板与信号模块	15
2.1.3 集成的通信接口与通信模块	17
2.2 STEP 7 Basic 编程软件	18
2.2.1 STEP 7 Basic 的特点	18
2.2.2 安装 STEP 7 Basic	19
2.2.3 创建一个项目	22
2.3 硬件组态	24
2.3.1 硬件组态概述	24
2.3.2 项目视图的结构	25
2.3.3 硬件组态的操作	27
2.3.4 信号模块与信号板的参数设置	28
2.3.5 将模拟量输入模块的输出值转换为实际的物理量	31
2.3.6 CPU 模块的参数设置	32
第 3 章 S7-1200 程序设计基础	37
3.1 S7-1200 的编程语言及国际标准	37
3.1.1 PLC 编程语言的国际标准	37
3.1.2 S7-1200 的编程语言	37
3.2 数据类型与系统存储区	39
3.2.1 物理存储器	39
3.2.2 数制与数据类型	40
3.2.3 系统存储区	44

3.3 用 STEP 7 Basic 生成用户程序 .....	46
3.3.1 设置 STEP 7 Basic 的参数 .....	46
3.3.2 编写用户程序 .....	47
3.3.3 使用 PLC 变量表 .....	50
3.4 下载用户程序 .....	52
3.5 用 STEP 7 Basic 调试程序 .....	56
3.5.1 用程序状态功能调试程序 .....	56
3.5.2 用监视表监视与修改变量 .....	57
3.5.3 用监视表强制变量 .....	62
3.6 位逻辑指令 .....	63
3.6.1 触点指令与线圈指令 .....	63
3.6.2 其他位逻辑指令 .....	65
3.7 定时器与计数器指令 .....	68
3.7.1 定时器指令 .....	68
3.7.2 计数器指令 .....	72
<b>第 4 章 数字量控制系统梯形图程序设计方法 .....</b>	<b>75</b>
4.1 梯形图中的基本电路与经验设计法 .....	75
4.1.1 梯形图中的基本电路 .....	75
4.1.2 梯形图的经验设计法 .....	77
4.2 顺序控制设计法与顺序功能图 .....	79
4.2.1 步与动作 .....	79
4.2.2 有向连线与转换条件 .....	81
4.2.3 顺序功能图的基本结构 .....	82
4.2.4 顺序功能图中转换实现的基本规则 .....	84
4.3 使用置位复位指令的顺序控制梯形图设计方法 .....	86
4.3.1 设计顺序控制梯形图的一些基本问题 .....	86
4.3.2 单序列的编程方法 .....	87
4.3.3 选择序列与并行序列的编程方法 .....	88
4.3.4 应用举例 .....	90
4.4 具有多种工作方式的系统的顺序控制梯形图设计方法 .....	92
4.4.1 系统的硬件结构与工作方式 .....	92
4.4.2 公用程序与手动程序 .....	94
4.4.3 自动程序 .....	95
4.4.4 自动回原点程序 .....	97
<b>第 5 章 S7-1200 的指令 .....</b>	<b>99</b>
5.1 数据处理指令 .....	99
5.1.1 比较指令 .....	99
5.1.2 使能输入与使能输出 .....	100
5.1.3 数据转换指令 .....	102

5.1.4	数据传送指令	104
5.1.5	移位与循环移位指令	107
5.2	数学运算指令与逻辑运算指令	109
5.2.1	数学运算指令	109
5.2.2	逻辑运算指令	112
5.3	程序控制指令	114
5.3.1	指令列表中的程序控制指令	114
5.3.2	扩展指令列表中的程序控制指令	114
5.4	字符串转换指令与字符串指令	117
5.4.1	字符串转换指令	117
5.4.2	字符串指令	121
5.5	高速脉冲输出与高速计数器	123
5.5.1	高速脉冲输出	123
5.5.2	编码器	125
5.5.3	高速计数器	126
5.5.4	高速脉冲输出与高速计数器的计数实验	129
5.5.5	用高速计数器测量频率的实验	132
5.6	其他指令	133
5.6.1	实时时钟指令	133
5.6.2	运动控制指令简介	135
<b>第6章</b>	<b>S7-1200的用户程序结构</b>	<b>136</b>
6.1	功能与功能块	136
6.1.1	生成与调用功能	136
6.1.2	生成与调用功能块	138
6.1.3	功能块的多重背景数据块	141
6.2	全局数据块与数据类型	142
6.2.1	全局数据块	142
6.2.2	数据类型	144
6.2.3	数据类型的转换	146
6.3	中断事件与中断指令	148
6.3.1	事件与组织块	148
6.3.2	组织块的实验	149
6.3.3	时间错误中断	156
6.3.4	硬件中断	158
6.3.5	中断连接与中断分离指令	159
6.4	交叉参考表与程序信息	161
6.4.1	交叉参考表	161
6.4.2	分配表	163
6.4.3	调用结构	166

6.4.4	附属结构与资源	167
<b>第 7 章</b>	<b>PLC 的通信与自动化通信网络</b>	<b>170</b>
7.1	计算机通信的国际标准	170
7.1.1	开放系统互连模型	170
7.1.2	IEEE 802 通信标准	171
7.1.3	现场总线及其标准	172
7.2	西门子的工业自动化通信网络	173
7.2.1	工业以太网与 PROFINET	173
7.2.2	现场总线 PROFIBUS 与 AS-i	176
7.3	S7-1200 的以太网通信	177
7.3.1	以太网设备的地址	177
7.3.2	S7-1200 与编程计算机的通信	178
7.4	故障诊断	182
7.4.1	在线功能与故障诊断	182
7.4.2	使用状态 LED 诊断故障	186
7.5	PLC 之间的开放式用户通信	187
7.5.1	开放式用户通信的编程	187
7.5.2	开放式用户通信的组态	190
7.6	串行通信	191
7.6.1	串行通信的基本概念	191
7.6.2	串行通信模块与串行通信指令	193
7.6.3	使用 Modbus 协议与 USS 协议的串行通信	194
<b>第 8 章</b>	<b>精简系列面板的组态与应用</b>	<b>197</b>
8.1	人机界面	197
8.1.1	人机界面与触摸屏	197
8.1.2	SIMATIC HMI 精简系列面板	198
8.2	精简系列面板的画面组态	199
8.2.1	使用 HMI 设备向导生成 HMI 设备	199
8.2.2	组态指示灯	203
8.2.3	组态按钮	206
8.2.4	组态文本域与 IO 域	209
8.2.5	棒图的组态与画面切换	211
8.3	精简系列面板的运行与模拟	212
8.3.1	用运行仿真器模拟 HMI	212
8.3.2	用 HMI 的控制面板设置 HMI 的参数	214
8.3.3	HMI 组态信息的下载与运行	216
<b>第 9 章</b>	<b>PLC 应用中的其他问题</b>	<b>219</b>
9.1	PLC 控制系统的设计与调试步骤	219
9.1.1	系统设计	219

9.1.2	PLC 硬件的选型	220
9.1.3	硬件软件设计与调试	220
9.2	PLC 控制系统的可靠性措施	222
9.3	PLC 在模拟量闭环控制中的应用	224
9.3.1	模拟量闭环控制系统与 PID 控制器	224
9.3.2	PID_Compact 指令与 PID 技术对象的组态	227
9.3.3	用调试窗口整定 PID 控制器	232
9.3.4	PID 参数的手动整定方法	236
<b>附录</b>	<b>随书光盘内容简介</b>	<b>239</b>
<b>参考文献</b>		<b>240</b>

1	PLC 基本概念与控制系统的组成	1
2	PLC 的硬件组成及接线	10
3	PLC 的梯形图语言	15
4	PLC 的功能块语言	18
5	PLC 的语句表语言	21
6	PLC 的 C 语言	24
7	PLC 的高级语言	27
8	PLC 的梯形图设计与编程	30
9	PLC 的功能块设计与编程	33
10	PLC 的语句表设计与编程	36
11	PLC 的 C 语言设计与编程	39
12	PLC 的高级语言设计与编程	42
13	PLC 的软元件与软继电器	45
14	PLC 的寻址方式	48
15	PLC 的数据类型	51
16	PLC 的存储器	54
17	PLC 的时钟与计数器	57
18	PLC 的定时器	60
19	PLC 的脉冲输出与脉冲输入	63
20	PLC 的模拟量输入与输出	66
21	PLC 的串行通信	69
22	PLC 的并行通信	72
23	PLC 的网络通信	75
24	PLC 的 I/O 口与本机总线	78
25	PLC 的电源与接地	81
26	PLC 的驱动与控制	84
27	PLC 的安装与维护	87
28	PLC 的故障诊断与排除	90
29	PLC 的设计与应用	93
30	PLC 的设计与应用	96
31	PLC 的设计与应用	99
32	PLC 的设计与应用	102
33	PLC 的设计与应用	105
34	PLC 的设计与应用	108
35	PLC 的设计与应用	111
36	PLC 的设计与应用	114
37	PLC 的设计与应用	117
38	PLC 的设计与应用	120
39	PLC 的设计与应用	123
40	PLC 的设计与应用	126
41	PLC 的设计与应用	129
42	PLC 的设计与应用	132
43	PLC 的设计与应用	135
44	PLC 的设计与应用	138
45	PLC 的设计与应用	141
46	PLC 的设计与应用	144
47	PLC 的设计与应用	147
48	PLC 的设计与应用	150
49	PLC 的设计与应用	153
50	PLC 的设计与应用	156
51	PLC 的设计与应用	159
52	PLC 的设计与应用	162
53	PLC 的设计与应用	165
54	PLC 的设计与应用	168
55	PLC 的设计与应用	171
56	PLC 的设计与应用	174
57	PLC 的设计与应用	177
58	PLC 的设计与应用	180
59	PLC 的设计与应用	183
60	PLC 的设计与应用	186
61	PLC 的设计与应用	189
62	PLC 的设计与应用	192
63	PLC 的设计与应用	195
64	PLC 的设计与应用	198
65	PLC 的设计与应用	201
66	PLC 的设计与应用	204
67	PLC 的设计与应用	207
68	PLC 的设计与应用	210
69	PLC 的设计与应用	213
70	PLC 的设计与应用	216
71	PLC 的设计与应用	219
72	PLC 的设计与应用	222
73	PLC 的设计与应用	225
74	PLC 的设计与应用	228
75	PLC 的设计与应用	231
76	PLC 的设计与应用	234
77	PLC 的设计与应用	237
78	PLC 的设计与应用	240

# 第1章 概述

## 1.1 PLC 的基本概念

随着微处理器、计算机和数字通信技术的飞速发展，计算机控制已经广泛地应用在几乎所有的工业领域。现代社会要求制造业对市场需求作出迅速的反应，生产出小批量、多品种、多规格、低成本和高质量的产品。为了满足这一要求，生产设备和自动生产线的控制系统必须具有极高的可靠性和灵活性，可编程序控制器（Programmable Logic Controller，PLC）正是顺应这一要求出现的，它是以微处理器为基础的通用工业控制装置。

PLC 的应用面广、功能强大、使用方便，已经成为当代工业自动化的主要支柱之一，在工业生产的几乎所有领域都得到了广泛的使用。PLC 在其他领域，例如在民用和家庭自动化中的应用也得到了迅速的发展。

### 1.1.1 PLC 的基本结构

本书以西门子公司新一代的模块化小型 PLC S7-1200 为主要讲授对象。西门子的 PLC 以其极高的性能价格比，在国际国内占有很大的市场份额，在我国各行各业得到了广泛的应用。

S7-1200 的结构紧凑、功能全面、扩展方便，其 CPU 集成有工业以太网通信接口和多种工艺功能，可以作为一个组件集成在完整的综合自动化系统中。

S7-1200 主要由 CPU 模块（简称为 CPU）、信号板、信号模块、通信模块和编程软件组成，各种模块安装在标准导轨上。通过 CPU 模块或通信模块上的通信接口，PLC 被连接到通信网络上，可以与计算机、其他 PLC 或其他设备通信。

#### 1. CPU 模块

CPU 模块主要由微处理器（CPU 芯片）和存储器组成。在 PLC 控制系统中，CPU 模块相当于人的大脑和心脏，它不断地采集输入信号，执行用户程序，刷新系统的输出；而存储器则用来储存程序和数据。S7-1200 将 CPU 模块简称为 CPU。

集成的 PROFINET 以太网接口用于与编程计算机、HMI（人机界面）、其他 PLC 的通信。此外它还通过开放的以太网协议支持与第三方设备的通信。

S7-1200 CPU（见图 1-1）集成有 6 个高速计数器。其中 3 个的最高输入频率为 100 kHz，另外 3 个为 30 kHz，还集成了两个 100 kHz 的高速脉冲输出，可以输出脉冲宽度调制（PWM）信号。

S7-1200 集成了 50KB 的工作存储器、最多 2MB 的装载存储器和 2 KB 的掉电保持存储器。使用 SIMATIC 存储卡最多可以扩展 24MB 装载存储器。

#### 2. 信号板

每块 CPU 内可以安装一块信号板（见图 1-2），安装后不会改变 CPU 的外形和体积。

有两种型号的信号板，一种有两点数字量输入和两点数字量输出，另一种有一点模拟量输出。

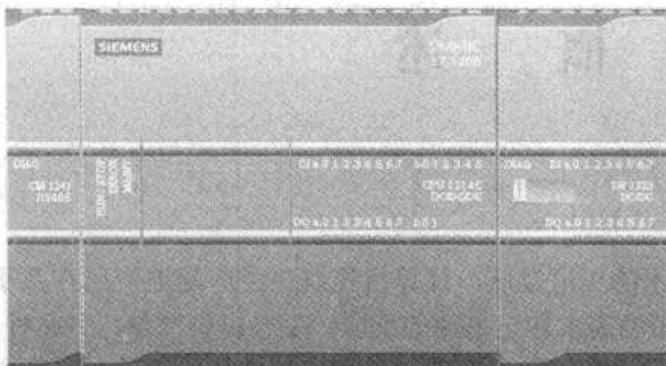


图 1-1 S7-1200 PLC

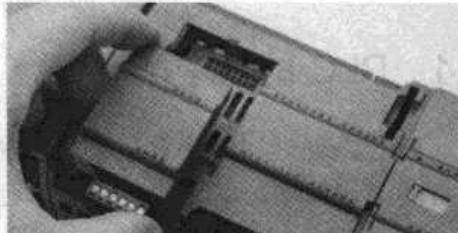


图 1-2 安装信号板

### 3. 信号模块

信号模块安装在 CPU 模块的右边，扩展能力最强的 CPU 可以扩展 8 个信号模块，以增加数字量和模拟量输入/输出点。

输入（Input）模块和输出（Output）模块简称为 I/O 模块，数字量（又称为开关量）输入模块和数字量输出模块简称为 DI 模块和 DO 模块，模拟量输入模块和模拟量输出模块简称为 AI 模块和 AO 模块，它们统称为信号模块，简称为 SM。

信号模块是系统的眼、耳、手、脚，是联系外部现场设备和 CPU 的桥梁。输入模块用来接收和采集输入信号，数字量输入模块用来接收从按钮、选择开关、数字拨码开关、限位开关、接近开关、光电开关、压力继电器等来的数字量输入信号。模拟量输入模块用来接收电位器、测速发电机和各种变送器提供的连续变化的模拟量电流、电压信号，或者直接接收热电阻、热电偶提供的温度信号。

数字量输出模块用来控制接触器、电磁阀、电磁铁、指示灯、数字显示装置和报警装置等输出设备，模拟量输出模块用来控制电动调节阀、变频器等执行器。

CPU 模块内部的工作电压一般是 DC 5V，而 PLC 的外部输入/输出信号电压一般较高，例如 DC 24V 或 AC 220V。从外部引入的尖峰电压和干扰噪声可能损坏 CPU 中的元器件，或使 PLC 不能正常工作。在信号模块中，用光耦合器、光敏晶闸管、小型继电器等器件来隔离 PLC 的内部电路和外部的输入、输出电路。信号模块除了传递信号外，还有电平转换与隔离的作用。

### 4. 通信模块

S7-1200 CPU 最多可以添加三个 RS-485 或 RS-232 串行通信模块，可以使用 ASCII 通信协议、USS 驱动协议、Modbus RTU 主站和从站协议，对通信的组态和编程采用扩展指令或库功能。

### 5. SIMATIC HMI 精简系列面板

全新的 SIMATIC HMI 精简系列面板（又称为基本面板）的触摸屏操作直观，有 4in、6in、10in 和 15in 4 种规格。其防护等级为 IP 65，可以在恶劣的工业环境中使用。

SIMATIC HMI 精简系列面板与 SIMATIC S7-1200 无缝兼容，为紧凑型自动化应用提供

了一种简单的可视化控制解决方案。

## 6. 编程软件

SIMATIC STEP 7 Basic 是西门子公司新一代的 PLC 编程软件, 它的操作直观、上手容易、使用简单。其智能功能可以提高工程组态的效率。STEP 7 Basic 集成了 SIMATIC WinCC Basic, 使用户能对 HMI 进行快速简单的组态和仿真。

由于 SIMATIC STEP 7 Basic 具有通用的项目视图、用于图形化工程组态的最新用户接口技术、智能的拖放功能以及共享的数据处理等特点, 因此有效地保证了项目的质量。

### 1.1.2 PLC 的特点

#### 1. 编程方法简单易学

梯形图是使用得最多的 PLC 的编程语言, 其电路符号和表达方式与继电器电路原理图相似, 梯形图语言形象直观, 易学易懂, 熟悉继电器电路图的电气技术人员只需花几天时间就可以熟悉梯形图语言, 并能用来编制数字量控制系统的用户程序。

#### 2. 功能强, 性能价格比高

一台小型 PLC 内有成百上千个可供用户使用的编程元件, 可以实现非常复杂的控制功能。与相同功能的继电器系统相比, 具有很高的性能价格比。PLC 可以通过通信连网, 实现分散控制、集中管理。

#### 3. 硬件配套齐全, 用户使用方便, 适应性强

PLC 产品已经标准化、系列化、模块化, 配备有品种齐全的各种硬件装置供用户选用, 用户能灵活方便地进行系统配置, 组成不同功能、不同规模的系统。PLC 的安装接线也很方便, 一般用接线端子连接外部接线。PLC 有较强的带负载能力, 可以直接驱动大多数电磁阀和中小型交流接触器。

硬件配置确定后, 通过修改用户程序, 就可以方便快速地适应工艺条件的变化。

#### 4. 可靠性高, 抗干扰能力强

传统的继电器控制系统使用了大量的中间继电器、时间继电器。由于触点接触不良, 容易出现故障。PLC 用软件代替中间继电器和时间继电器, 仅剩下与输入和输出有关的少量硬件元件。与继电器控制系统相比, 可以减少大量的硬件触点和接线, 大大减少了因触点接触不良造成的故障。

PLC 使用了一系列硬件和软件抗干扰措施, 具有很强的抗干扰能力, 平均无故障时间达到数万小时以上, 可以直接用于有强烈干扰的工业生产现场, PLC 被广大用户公认为最可靠的工业控制设备之一。

#### 5. 系统的设计、安装、调试工作量少

PLC 用软件功能取代了继电器控制系统中大量的中间继电器、时间继电器、计数器等器件, 使控制柜的设计、安装、接线工作量大大减少。

PLC 的梯形图程序可以用顺序控制设计法来设计。这种设计方法很有规律, 很容易掌握。对于复杂的控制系统, 用这种方法设计程序的时间比设计继电器系统电路图的时间要少得多。

## 6. 维修工作量小，维修方便

PLC 的故障率很低，并且有完善的故障诊断功能。PLC 或外部的输入装置和执行机构发生故障时，可以根据信号模块上的发光二极管或编程软件提供的信息，方便快速地查明故障的原因，用更换模块的方法可以迅速地排除故障。

## 7. 体积小，能耗低

复杂的控制系统使用 PLC 后，可以减少大量的中间继电器和时间继电器，小型 PLC 的体积仅相当于几个继电器的大小，因此可以将开关柜的体积缩小到原来的  $1/2 \sim 1/10$ 。

PLC 控制系统与继电器控制系统相比，减少了大量的接线，节省了控制柜内安装接线的工作量，加上开关柜体积的缩小，因此可以节省大量的费用。

### 1.1.3 PLC 的应用领域

PLC 已经广泛地应用在很多工业部门，随着其性能价格比的不断提高，应用范围不断扩大。PLC 的应用领域主要有以下几个方面：

#### 1. 开关量逻辑控制

PLC 具有“与”、“或”、“非”等逻辑指令，可以实现梯形图中触点和电路的串、并联，代替继电器进行组合逻辑控制、定时控制与顺序逻辑控制。开关量逻辑控制可以用于单台设备，也可以用于自动生产线，其应用领域已经遍及各行各业，甚至深入到民用和家庭。

#### 2. 运动控制

PLC 使用专用的指令或运动控制模块，对直线运动或圆周运动的位置、速度和加速度进行控制，可以实现单轴、双轴、3 轴和多轴联动的位置控制，使运动控制与顺序控制功能有机地结合在一起。PLC 的运动控制功能广泛地用于各种机械，例如金属切削机床、金属成形机械、装配机械、机器人、电梯等。

#### 3. 闭环过程控制

闭环过程控制是指对温度、压力、流量等连续变化的模拟量的闭环控制。PLC 通过模拟量 I/O 模块，实现模拟量（Analog）和数字量（Digital）之间的 A/D 转换与 D/A 转换，并对模拟量实行闭环 PID 控制。其闭环控制功能广泛地应用于塑料挤压成形机、加热炉、热处理炉、锅炉等设备，以及轻工、化工、机械、冶金、电力、建材等行业。

#### 4. 数据处理

现代的 PLC 具有整数四则运算、矩阵运算、函数运算、字逻辑运算、求反、循环、移位、浮点数运算等运算功能，和数据传送、转换、排序、查表、位操作等功能，可以完成数据的采集、分析和处理。这些数据可以与存储在存储器中的参考值比较，也可以用通信功能传送到别的智能装置，或者将它们打印制表。

#### 5. 通信连网

PLC 的通信包括 PLC 与远程 I/O 之间的通信、多台 PLC 之间的通信、PLC 与其他智能控制设备（例如计算机、变频器、数控装置）之间的通信。PLC 与其他智能控制设备一起，可以组成“集中管理、分散控制”的分布式控制系统。

### 1.1.4 怎样下载西门子 PLC 的资料和软件

可以在西门子（中国）有限公司工业业务领域工业自动化与驱动技术集团的中文网站（[www.ad.siemens.com.cn](http://www.ad.siemens.com.cn)）下载西门子的 PLC 资料。单击该网站主页的“支持中心”后，在界面右侧“技术资源库”中点击“下载中心”（见图 1-3），使用搜索功能，可以下载感兴趣的工控产品的中英文使用手册、产品样本、常问问题和软件。



图 1-3 技术资源库

点击图 1-3 中的“全球技术资源”，将会打开西门子自动化的支持中心网站。在该中心的主页可以设置语言为中文或英语。在左边的导航窗口选择感兴趣的产品，在右边的窗口可以下载软件、手册，查看常见问题解答和技术参数等。

为了阅读 PDF 格式的手册，需要在计算机上安装 Adobe 阅读器，可以在互联网上搜索和下载该阅读器的最新版本。

## 1.2 S7-1200 的程序结构与工作原理

### 1.2.1 逻辑运算

在数字量（或称开关量）控制系统中，变量仅有两种相反的工作状态，例如高电平和低电平、继电器线圈的通电和断电，可以分别用逻辑代数中的 1 和 0 来表示这些状态，在波形图中，用高电平表示 1 状态，用低电平表示 0 状态。

使用数字电路、数字电路或 PLC 的梯形图都可以实现数字量逻辑运算。图 1-4 的上面是 PLC 的梯形图，下面是对应的数字门电路。

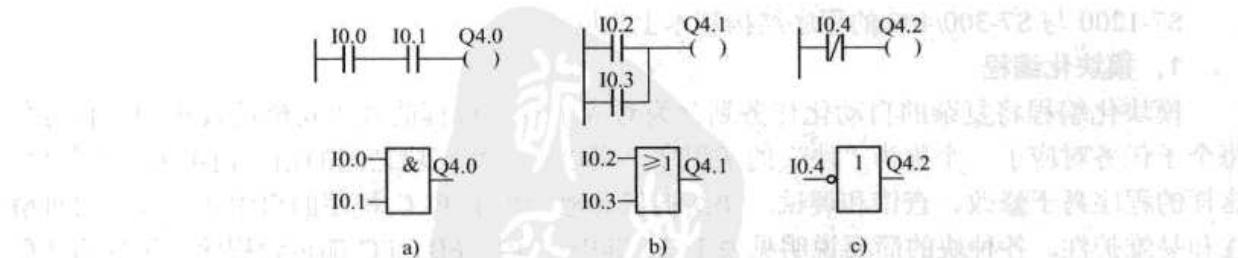


图 1-4 基本逻辑运算

图 1-4 中的 I0.0~I0.4 为数字量输入变量, Q4.0~Q4.2 为数字量输出变量, 它们之间的“与”、“或”、“非”逻辑运算关系如表 1-1 所示。表中的 0 和 1 分别表示输入点的常开触点断开和接通, 或表示线圈断电和线圈通电。

表 1-1 逻辑运算关系表

与			或			非	
$Q4.0 = I0.0 \cdot I0.1$			$Q4.1 = I0.2 + I0.3$			$Q4.2 = \overline{I0.4}$	
I0.0	I0.1	Q4.0	I0.2	I0.3	Q4.1	I0.4	Q4.2
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1	-	-
1	1	1	1	1	1	-	-

用继电器电路或梯形图可以实现基本的逻辑运算, 触点的串联可以实现“与”运算, 触点的并联可以实现“或”运算, 用常闭触点控制线圈可以实现“非”运算。多个触点的串、并联电路可以实现复杂的逻辑运算, 例如图 1-5 中的继电器电路实现的逻辑运算可以用逻辑代数表达式表示为

$$KM = (SB1 + KM) \cdot \overline{SB2} \cdot FR$$

式中的加号表示逻辑或, 乘号(或星号)表示逻辑与, 变量上面的横线表示“非”运算。

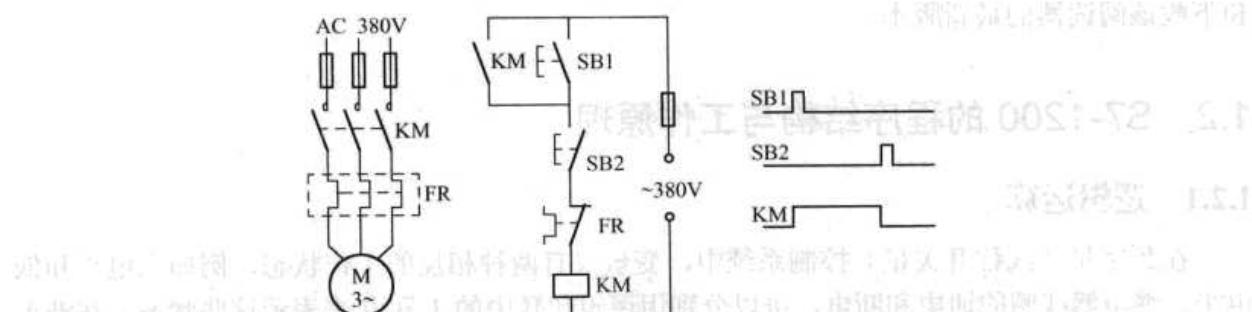


图 1-5 继电器控制电路

## 1.2.2 S7-1200 用户程序结构简介

S7-1200 与 S7-300/400 的程序结构基本上相同。

### 1. 模块化编程

模块化编程将复杂的自动化任务划分为对应于生产过程的技术功能的较小的子任务, 每个子任务对应于一个称为“块”的子程序, 可以通过块与块之间的相互调用来组织程序。这样的程序易于修改、查错和调试。块结构显著地增加了 PLC 程序的组织透明性、可理解性和易维护性。各种块的简要说明见表 1-2, 其中, OB、FB、FC 都包含程序, 统称为代码(Code)块。

表 1-2 用户程序中的块

块	简要描述
组织块 (OB)	操作系统与用户程序的接口，决定用户程序的结构
功能块 (FB)	用户编写的包含经常使用的功能的子程序，有专用的背景数据块
功能 (FC)	用户编写的包含经常使用的功能的子程序，没有专用的背景数据块
背景数据块 (DB)	用于保存 FB 的输入变量、输出变量和静态变量，其数据在编译时自动生成
全局数据块 (DB)	存储用户数据的数据区域，供所有的代码块共享

被调用的代码块又可以调用别的代码块，这种调用称为嵌套调用。CPU 模块的手册给出了允许嵌套调用的层数，即嵌套深度。代码块的个数没有限制，但是受到存储器容量的限制。

在块调用中，调用者可以是各种代码块，被调用的块是 OB 之外的代码块。调用功能块时需要为它指定一个背景数据块。

在图 1-6 中，OB1 调用 FB1，FB1 调用 FC1，应按下面的顺序创建块：FC1→FB1 及其背景数据块→OB1，即编程时被调用的块应该是已经存在的。

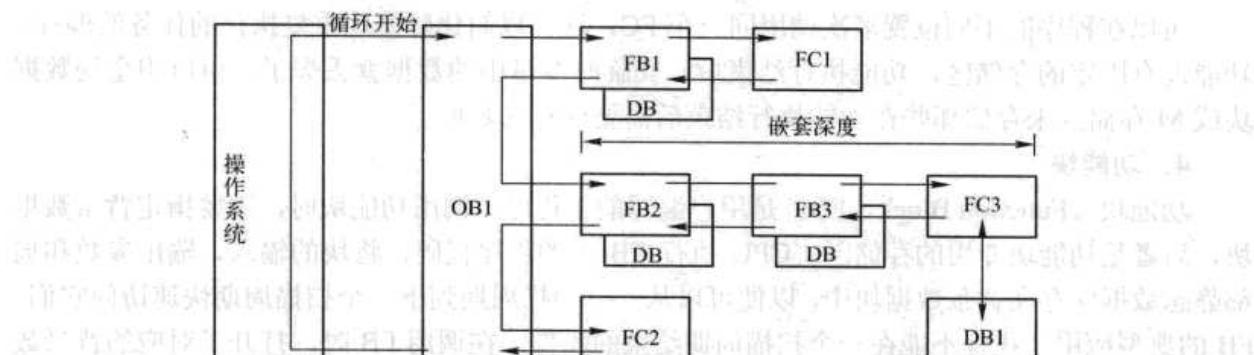


图 1-6 块调用的分层结构

## 2. 组织块

组织块 (Organization Block, OB) 是操作系统与用户程序的接口，由操作系统调用，用于控制扫描循环和中断程序的执行、PLC 的启动和错误处理等。组织块的程序是用户编写的。

每个组织块必须有一个唯一的 OB 编号，200 之前的某些编号是保留的，其他 OB 的编号应大于等于 200。CPU 中特定的事件触发组织块的执行，OB 不能相互调用，也不能被 FC 和 FB 调用。只有启动事件（例如诊断中断或周期性中断事件）可以启动 OB 的执行。

### (1) 程序循环组织块

OB1 是用户程序中的主程序，CPU 循环执行操作系统程序，在每一次循环中，操作系统程序调用一次 OB1。因此 OB1 中的程序也是循环执行的。允许有多个程序循环 OB，默认的是 OB1，其他程序循环 OB 的编号应大于等于 200。

### (2) 启动组织块

当 CPU 的操作模式从 STOP 切换到 RUN 时，执行一次启动 (Startup) 组织块，来初始化程序循环 OB 中的某些变量。执行完启动 OB 后，开始执行程序循环 OB。可以有多个启动 OB，默认的为 OB100，其他启动 OB 的编号应大于等于 200。

### (3) 中断组织块

中断处理用来实现对特殊内部事件或外部事件的快速响应。如果没有中断事件出现，CPU 循环执行组织块 OB1。如果出现中断事件，例如诊断中断和时间延迟中断等，因为 OB1 的中断优先级最低，操作系统在执行完当前程序的当前指令（即断点处）后，立即响应中断。CPU 暂停正在执行的程序块，自动调用一个分配给该事件的组织块（即中断程序）来处理中断事件。执行完中断组织块后，返回被中断的程序的断点处继续执行原来的程序。

这意味着部分用户程序不必在每次循环中处理，而是在需要时才被及时地处理。处理中断事件的程序放在该事件驱动的 OB 中。

6.3 节详细介绍了各种中断组织块和中断事件的处理方法。

### 3. 功能

功能（Function, FC）是用户编写的子程序，它包含完成特定任务的代码和参数。FC 和 FB 有与调用它的块共享的输入/输出参数。执行完 FC 和 FB 后，返回调用它的代码块。

功能是快速执行的代码块，用于执行下列任务：

- 1) 完成标准的和可重复使用的操作，例如算术运算。
- 2) 完成技术功能，例如使用位逻辑运算的控制。

可以在程序的不同位置多次调用同一个 FC，这可以简化频繁地重复执行的任务的编程。功能没有固定的存储区，功能执行结束后，其临时变量中的数据就丢失了。可以用全局数据块或 M 存储区来存储那些在功能执行结束后需要保存的数据。

### 4. 功能块

功能块（Function Block, FB）是用户编写的子程序。调用功能块时，需要指定背景数据块，后者是功能块专用的存储区。CPU 执行 FB 中的程序代码，将块的输入、输出参数和局部静态数据保存在背景数据块中，以便可以从一个扫描周期到下一个扫描周期快速访问它们。FB 的典型应用是执行不能在一个扫描周期结束的操作。在调用 FB 时，打开了对应的背景数据块，后者的变量可以供其他代码块使用。

调用同一个功能块时使用不同的背景数据块，可以控制不同的设备。图 1-7 中的 FB22 用来控制水泵和阀门，使用包含特定的操作参数的不同的背景数据块 DB201~DB203，可以控制不同的水泵和阀门。

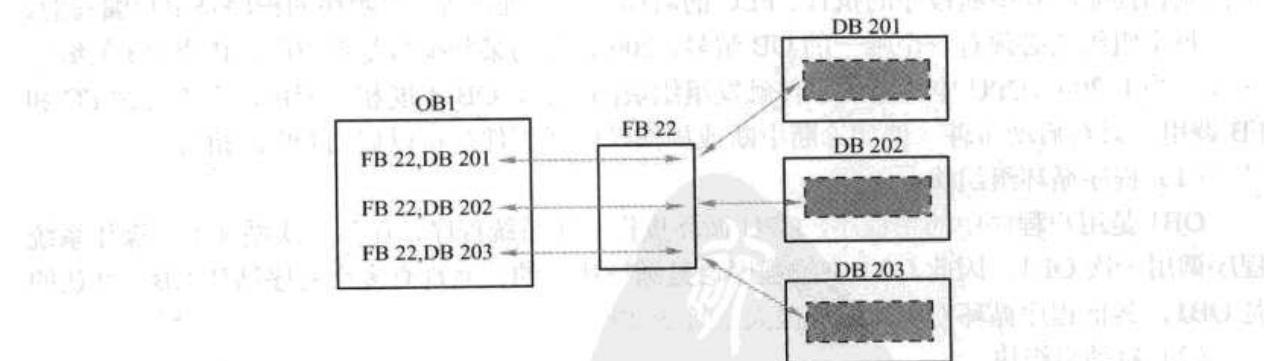


图 1-7 调用功能块

S7-1200 的部分指令（例如 IEC 标准的定时器和计数器指令）实际上是功能块，在调用

它们时需要指定配套的背景数据块。

## 5. 数据块

数据块（Data block, DB）是用于存放执行代码块时所需的数据的数据区，有两种类型的数据块：

- 1) 全局（Global）数据块。存储供所有的代码块使用的数据，所有的 OB、FB 和 FC 都可以访问它们。
- 2) 背景数据块。存储供特定的 FB 使用的数据。

## 1.2.3 PLC 的工作原理

### 1. 操作系统与用户程序

CPU 的操作系统用来组织与具体的控制任务无关的所有的 CPU 功能。操作系统的任务包括处理暖启动，刷新输入/输出过程映像，调用用户程序，检测中断事件和调用中断组织块，检测和处理错误，管理存储器，以及处理通信任务等。

用户程序包含处理具体的自动化任务必须的所有功能。用户程序由用户编写并下载到 CPU，用户程序的任务包括：

- 1) 检查是否满足暖启动需要的条件，例如限位开关是否在正确的位置，安全继电器是否处于正常的工作状态。
- 2) 处理过程数据，例如用读取的数字量输入信号来控制数字量输出信号，读取和处理模拟量输入信号，输出模拟量值。
- 3) 用 OB（组织块）中的程序对中断事件作出反应，例如在诊断错误中断 OB82 中发出报警信号。
- 4) 在程序执行中处理错误。

### 2. CPU 的操作模式

CPU 有 3 种操作模式：RUN（运行）、STOP（停机）与 STARTUP（启动）。CPU 面板上的状态 LED（发光二极管）用来指示当前的操作模式，可以用编程软件的命令改变 CPU 的操作模式。

#### （1）STOP 模式

STOP 模式不执行用户程序，所有的输出被禁止或按组态时的设置提供替代值或保持最后的输出值，以保证系统处于安全状态。CPU 不执行用户程序和自动刷新过程映像，仅处理通信请求和进行自诊断，可以下载项目。

#### （2）STARTUP 模式

上电后 CPU 进入 STARTUP（启动）模式，进行上电诊断和系统初始化，检查到某些错误时，将禁止 CPU 进入 RUN 模式，保持在 STOP 模式。

在 CPU 内部的存储器中，设置了一片区域来存放输入信号和输出信号的状态，它们被称为过程映像输入区和过程映像输出区。

CPU 的操作模式从 STOP 切换到 RUN 时，进入启动模式，CPU 执行下列操作（见图 1-8 中各阶段的符号）：

阶段 A 复位过程映像输入区（I 存储区）。

阶段 B 用上一次 RUN 模式最后的值或替代值初始化输出。

阶段 C 执行启动 OB。如果启动 OB 不止一个，首先执行 OB100，然后按递增的编号执

行其他启动 OB。在启动 OB 中用指令对过程映像输入的读访问均为 0，可以用立即读指令读取物理输入点的当前状态。

阶段 D 将物理输入状态复制到 I 存储区。

阶段 E（整个启动阶段）将中断事件保存到队列，以便在 RUN 模式进行处理。

阶段 F 将过程映像输出区（Q 区）的值写到物理输出。

在启动阶段不进行循环时间监控。

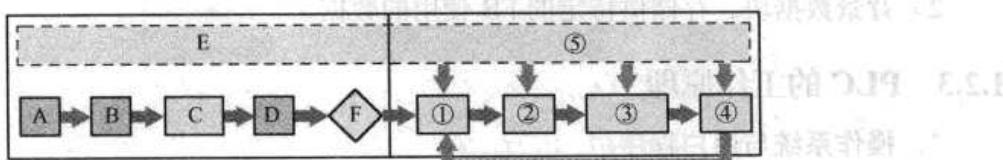


图 1-8 启动与运行过程示意图

### (3) RUN 模式

启动阶段结束后，进入 RUN 模式。为了使 PLC 的输出及时地响应各种输入信号，CPU 反复不停地分阶段处理各种不同的任务（见图 1-8）：

阶段①将过程映像输出区的值写到物理输出。

阶段②将物理输入状态复制到过程映像输入区。

阶段③执行一个或多个程序循环 OB，首先执行主程序 OB1。

阶段④处理通信请求和进行自诊断。

上述任务是按顺序执行的。这种周而复始的循环工作方式称为扫描循环。

在扫描循环的任意阶段（阶段⑤）出现中断事件时，执行中断程序。

可以用编程软件来设置上电时的启动方法（见图 2-33）。CPU 模块上没有模式选择开关，只能用编程软件的命令来切换 RUN、STOP 模式，或者在用户程序中用 STP 指令使 CPU 进入 STOP 模式。

下载了用户程序的块和硬件组态后，下一次切换到 RUN 模式时，CPU 执行冷启动，复位输入，初始化输出，清除非保持存储器和保持存储器。冷启动之后，在下一次下载之前的 STOP 到 RUN 模式的切换均为暖启动，所有非保持的系统数据和用户数据被初始化，不会清除保持存储器。

可以用编程软件的“CPU 操作面板”上的“MRES”按钮（见图 7-12）来复位存储器，即清除所有的工作存储器，清除保持和非保持存储区，复制装载存储器的内容给工作存储器。存储器复位并不清除诊断缓冲区和 IP 地址。

RUN 模式不能下载项目，S7-1200 CPU 之间通过开放式用户通信进行的数据交换只能在 RUN 模式进行。

S7-1200 不支持在运行时插入或拔出信号板、信号模块和通信模块，但是可以插入或拔出 SIMATIC 存储卡。

## 3. RUN 模式 CPU 的操作

下面是 RUN 模式各阶段任务的详细介绍。

### (1) 写物理输出

扫描循环开始时，将过程映像区中的数字量和模拟量输出的当前值写入物理输出，包括

被组态为默认的同步刷新的 CPU 集成的物理输出、信号板和信号模块的物理输出。

CPU 执行完用户程序后，将过程映像输出区的 I/O 状态（或称为 ON/OFF 状态）传送到物理输出模块并锁存起来。梯形图中某一输出位的线圈“通电”时，对应的输出过程映像寄存器为 1 状态。信号经输出模块隔离和功率放大后，继电器型输出模块中对应的硬件继电器的线圈通电，其常开触点闭合，使外部负载通电工作。若梯形图中某输出点的线圈“断电”，则对应的输出过程映像寄存器中的二进制数为 0。将它送到继电器型输出模块，对应的硬件继电器的线圈断电，其常开触点断开，外部负载断电，停止工作。

可以用指令立即改写物理输出点的值，同时刷新了过程映像输出。

#### (2) 读物理输入

在扫描循环的第二阶段，从 CPU 集成的物理输入、信号板和信号模块的物理输入读取被组态为默认的同步刷新的数字量和模拟量输入的当前值，然后将它们写入输入过程映像。

在读取输入阶段，PLC 把所有外部数字量输入电路的 I/O 状态读入输入过程映像寄存器。外接的输入电路闭合时，对应的输入过程映像为 1 状态，梯形图中对应的输入点的常开触点接通，常闭触点断开。外接的输入电路断开时，对应的输入过程映像为 0 状态，梯形图中对应的输入点的常开触点断开，常闭触点接通。

可以用指令立即读取数字量或模拟量的物理输入点的值，但是不会刷新过程映像输入。

#### (3) 执行用户程序

PLC 的用户程序由若干条指令组成，指令在存储器中按顺序排列。读取输入后，从第一条指令开始，逐条顺序执行用户程序，直到最后一条指令，包括程序循环 OB 调用的 FC 和 FB。首先执行 OB1，然后按递增的编号执行其他程序循环 OB。

在执行指令时，从输入/输出过程映像或别的位元件的存储单元读出其 0/1 状态，并根据指令的要求执行相应的逻辑运算，运算的结果写入到相应的输出过程映像和其他存储单元，它们的内容随着程序的执行而变化。

程序执行过程中，各输出点的值被保存到输出过程映像，而不是立即写给物理输出。

在程序执行阶段，即使外部输入信号的状态发生了变化，输入过程映像的状态也不会随之而变，输入信号变化了的状态只能在下一个扫描周期的读取输入阶段被读入。执行程序时，对输入/输出的存取通常是通过过程映像，而不是实际的 I/O 点，这样做有以下好处：

——在整个程序执行阶段，过程映像中各输入点的状态是固定不变的，程序执行完后再用输出过程映像的值更新输出点，使系统的运行稳定，各物理输出点的值在一个扫描周期内不会多次变化。用户程序读写 I/O 过程映像比读写 I/O 点的速度快，这样可以提高程序的执行速度。

#### (4) 通信处理与自诊断

在扫描循环的通信处理和自诊断阶段，处理接收到的报文，在适当的时候将报文发送给通信的请求方。周期性地检查固件、用户程序和 I/O 模块的状态。

#### (5) 中断处理

事件驱动的中断可以在扫描循环的任意阶段发生。有事件出现时，CPU 中断扫描循环，调用组态给该事件的 OB。OB 处理完事件后，CPU 在中断点恢复用户程序的执行。中断功能可以提高 PLC 对某些事件的响应速度。

## 第2章 PLC的硬件与硬件组态

### 2.1 S7-1200 的硬件

S7-1200 是西门子公司的新一代小型 PLC，它具有集成的 PROFINET 接口、强大的集成工艺功能和灵活的可扩展性等特点，为各种工艺任务提供了简单的通信和有效的解决方案，能满足完全不同的自动化需求。

S7-1200 系列 PLC 具有高度的灵活性，用户可根据自身需求确定 PLC 的结构，系统扩展也十分方便。

#### 2.1.1 CPU 模块

S7-1200 现在有 3 种型号的 CPU 模块（简称为 CPU，见表 2-1）。此外还有计划中的 CPU 1215C 和 CPU 1217C。

表 2-1 S7-1200 CPU 技术规范

特性	CPU 1211C	CPU 1212C	CPU 1214C
本机数字量 I/O 点数	6 入/4 出 2	8 入/6 出 2	14 入/10 出 2
脉冲捕获输入点数	6	8	14
扩展模块个数	—	2	8
上升沿/下降沿中断点数	6/6	8/8	12/12
集成/可扩展的工作存储器	25KB/不可扩展 1MB/24MB	25KB/不可扩展 1MB/24MB	50KB/不可扩展 2MB/24MB
集成/可扩展的装载存储器			
高速计数器点数/最高频率	3 点/100kHz 1 点/30kHz	3 点/100kHz 1 点/30kHz	3 点/100kHz 3 点/30kHz
高速脉冲输出点数/最高频率	2 点/100 kHz (DC/DC/DC 型)		
操作员监控功能	无	有	有
传感器电源输出电流/mA	300	300	400
外形尺寸/mm	90×100×75	90×100×75	110×100×75

图 2-1 中的①是集成 I/O(输入/输出)的状态 LED (发光二极管)。②是 3 个指示 CPU 运行状态的 LED，③是 PROFINET 以太网接口的 RJ-45 连接器，④是存储卡插槽 (在盖板下面)，⑤是可拆卸的接线端子板。

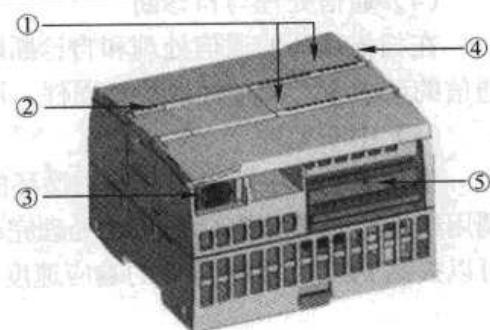


图 2-1 CPU 模块

#### 1. CPU 的共性

1) 集成的 24V 传感器/负载电源可供传感器和编码器使用，也可以用来作输入回路的电源。

2) 2 点集成的模拟量输入 (0~10V)，输入电阻

100kΩ，10位分辨率。

- 3) 2点脉冲列输出(PTO)或脉宽调制(PWM)输出,最高频率100kHz。
- 4) 每条位运算、字运算和浮点数数学运算指令的执行时间为0.1μs、12μs和18μs。
- 5) 最多可以设置2048B有掉电保持功能的数据区(包括位存储器、功能块的接口变量和全局数据块中的变量)。

通过可选的SIMATIC存储卡,可以方便地将程序传输到其他CPU。存储卡还可以用来存储各种文件或更新PLC系统的固件。

- 6) 过程映像输入、输出各1024B。

数字量输入电路的电压额定值为DC 24V,输入电流为4mA。1状态允许的最小电压/电流为DC 15V/2.5mA,0状态允许的最大电压/电流为DC 5V/1mA。可组态输入延迟时间(0.2~12.8ms)和脉冲捕获功能。在过程输入信号的上升沿或下降沿可以产生快速响应的中断输入。

继电器输出的电压范围为DC 5~30V或AC 5~250V。最大电流2A,白炽灯负载为DC 30W或AC 200W。

DC/DC/DC型MOSFET的1状态最小输出电压为DC 20V,输出电流0.5A。0状态最大输出电压为DC 0.1V。最大白炽灯负载为5W。

7) 可以扩展3块通信模块和一块信号板,CPU可以用信号板扩展1路模拟量输出或数字量输入/输出(2DI/2DO)。

- 8) 4个时间延迟与循环中断,分辨率为1ms。

- 9) 硬件实时时钟的缓存时间典型值为10天,最小值6天,25℃时的最大误差为60s/月。

10) 集成的带隔离的PROFINET以太网接口,可使用TCP/IP和ISO-on-TCP协议。支持S7通信,可以作服务器和客户机,传输速率为10Mbit/s/100 Mbit/s,可建立最多16个连接。自动检测传输速率,RJ-45连接器有自协商和自动交叉网线(Auto-Cross-Over)功能。后者是指用一条直通网线或者交叉网线都可以连接CPU和其他以太网设备或交换机。

- 11) 可以使用梯形图和功能块图这两种编程语言。

- 12) 可以用SIMATIC存储卡扩展存储器。

- 13) 有16个参数自整定的PID控制器。

- 14) 可选的仿真器(小开关板)为数字量输入点提供输入信号来测试用户程序。

## 2. CPU的技术规范

每种CPU有3种具有不同电源电压和输入、输出电压的版本(见表2-2)。

图2-2是CPU1214C AC/DC/Relay(继电器)型的外部接线图。输入回路一般使用CPU内置的DC 24V电源,此时需要去除图2-2中的外接DC电源,将输入回路的1M端子与24V电源的M端子连接起来,将24V电源的L+端子接到外接触点的公共端。

表2-2 S7-1200 CPU的3种版本

版本	电源电压	DI输入电压	DO输出电压	DO输出电流
DC/DC/DC	DC 24V	DC 24V	DC 24V	0.5 A, MOSFET
DC/DC/Relay	DC 24V	DC 24V	DC 5~30V, AC 5~250V	2A, DC 30W / AC 200W
AC/DC/Relay	AC 85~264V	DC 24V	DC 5~30V, AC 5~250V	2A, DC 30W / AC 200W

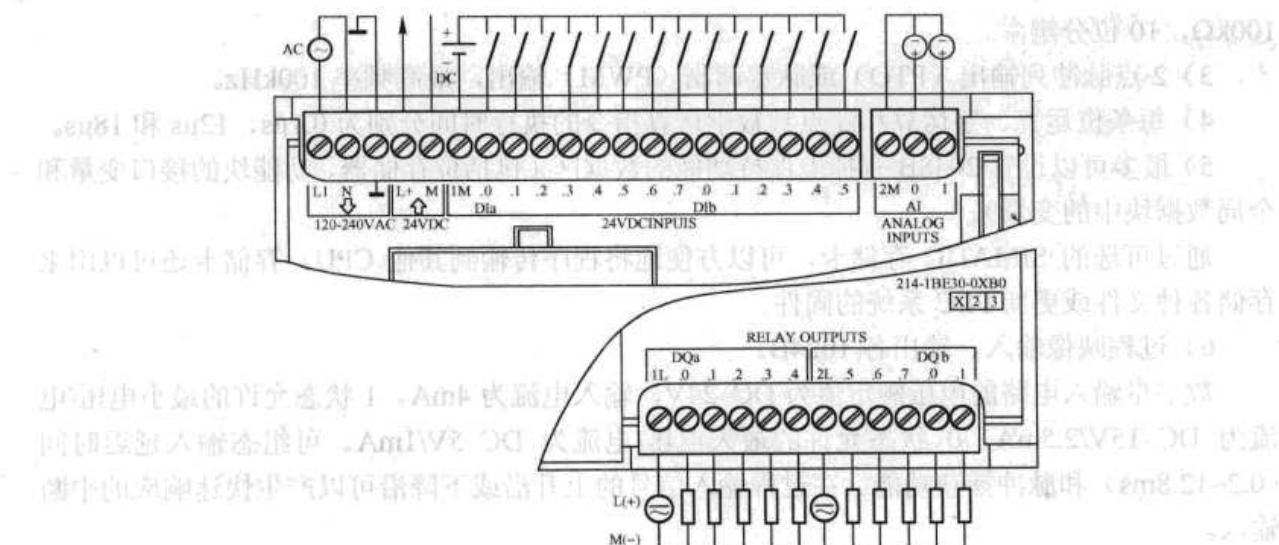


图 2-2 CPU 1214C AC/DC/Relay 的外部接线图

CPU 1214C DC/DC/Relay 的接线图与图 2-2 的区别在于前者的电源电压为 DC 24V。

CPU 1214C DC/DC/DC 的接线图见图 2-3, 其电源电压、输入回路电压和输出回路电压均为 DC 24V。输入回路也可以使用内置的 DC 24V 电源。

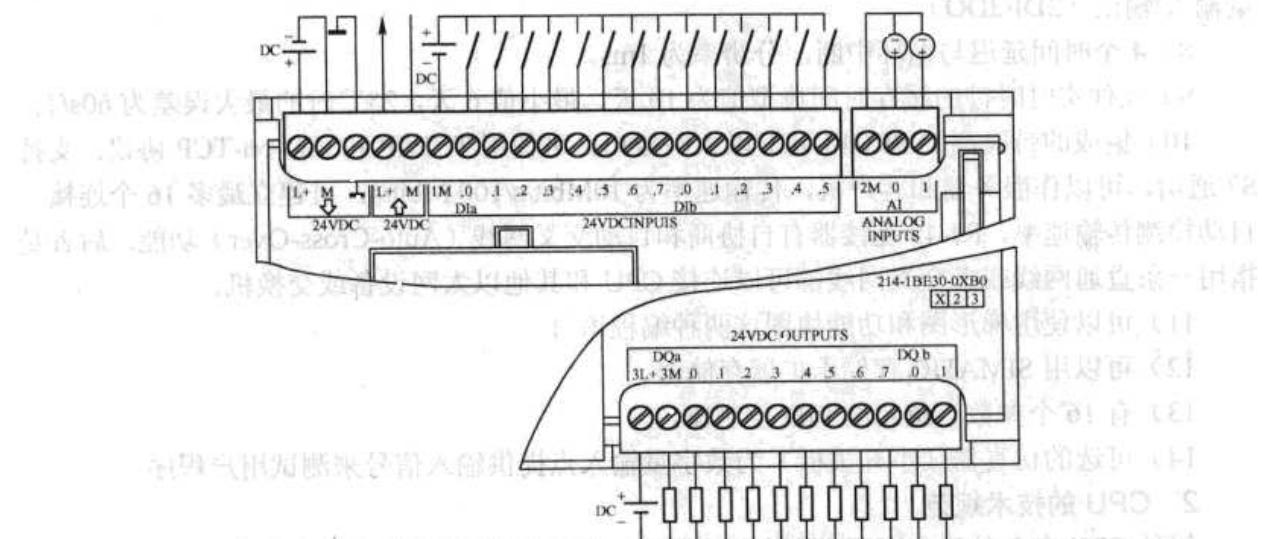


图 2-3 CPU 1214C DC/DC/DC 的外部接线图

### 3. CPU 集成的工艺功能

S7-1200 集成了高速计数与频率测量、高速脉冲输出、PWM 控制、运动控制和 PID 控制功能。

#### (1) 高速计数器

S7-1200 的 CPU 最多有 6 个高速计数器，用于对来自增量式编码器和其他设备的频率信号计数，或对过程事件进行高速计数。3 点集成的高速计数器的最高频率为 100 kHz（单相）或 80 kHz（互差 90°的 AB 相信号）。其余各点的最高频率为 30 kHz（单相）或 20 kHz（互差 90°的 AB 相信号）。

## ② 高速输出

S7-1200 集成了两个 100 kHz 的高速脉冲输出，组态为 PTO 时，它们提供最高频率为 100 kHz 的 50% 占空比的高速脉冲输出，可以对步进电动机或伺服驱动器进行开环速度控制和定位控制，通过两个高速计数器对高速脉冲输出进行内部反馈。

组态为 PWM 输出时，将生成一个具有可变占空比、周期固定的输出信号，来控制电动机速度、阀门位置或加热元件的占空比。

## ③ PLCopen 运动功能块

S7-1200 支持使用步进电动机和伺服驱动器进行开环速度控制和位置控制。通过一个轴工艺对象和 STEP 7 Basic 中通用的 PLCopen 运动功能块，就可以实现对该功能的组态。除了返回原点和点动功能以外，还支持绝对位置控制、相对位置控制和速度控制。

STEP 7 Basic 中的驱动调试控制面板简化了步进电动机和伺服驱动器的启动和调试过程。它为单个运动轴提供了自动和手动控制，以及在线诊断信息。

## ④ 用于闭环控制的 PID 功能

S7-1200 支持多达 16 个用于闭环过程控制的 PID 控制回路（S7-200 只支持 8 个）。

这些控制回路可以通过一个 PID 控制器工艺对象和 STEP 7 Basic 中的编辑器轻松地进行组态。除此之外，S7-1200 还支持 PID 参数自调整功能，可以自动计算增益、积分时间和微分时间的最佳调节值。

STEP 7 Basic 中的 PID 调试控制面板简化了控制回路的调节过程，可以快速精确地调节 PID 控制回路。它除了提供自动调节和手动控制方式之外，还提供调节过程的趋势图。

## 2.1.2 信号板与信号模块

S7-1200 的 CPU 可以根据系统的需要进行扩展。各种 CPU 的正面都可以增加一块信号板，以扩展数字量或模拟量 I/O。信号模块连接到 CPU 的右侧，以扩展其数字量或模拟量 I/O 的点数。CPU 1212C 只能连接两个信号模块，CPU 1214C 可以连接 8 个信号模块。所有的 S7-1200 CPU 都可以在 CPU 的左侧安装最多 3 个通信模块。

S7-1200 所有的模块都具有内置的安装夹，能方便地安装在一个标准的 35mm DIN 导轨上。S7-1200 的硬件可以竖直安装或水平安装。

所有的 S7-1200 硬件都配备了可拆卸的端子板，不用重新接线，就能迅速地更换组件。

### 1. 信号板

信号板可以用于控件有限或只需要少量附加 I/O 的情况。所有的 S7-1200 CPU 模块都可以安装一块信号板，并且不会增加安装的空间。在某些情况下使用信号板，可以提高控制系统的性能价格比。只需要添加一块信号板，就可以根据需要增加 CPU 的数字量或模拟量 I/O 点。

安装时将信号板直接插入 S7-1200 CPU 正面的槽内（见图 1-2）。信号板有可拆卸的端子，因此可以很容易地更换信号板。

有两种信号板：

1) SB 1223 数字量输入/输出信号板如图 2-4 所示。它的两点 DC 24V 输入有上升沿、下降沿中断和脉冲捕获功能。输入参数与 CPU 集成的输入点的基本上相同。用来作高速计数器的时钟输入时最高输入频率为 30 kHz。

两个 DC 24V MOSFET 输出点的最大输出电流为 0.5 A，最大白炽灯负载为 DC 5W，可以输出最高 20 kHz 的脉冲列。

2) SB 1232 模拟量输出信号板如图 2-5 所示。其输出分辨率为 12 位的 -10~+10V 电压，负载阻抗大于等于  $1000\Omega$ ；或输出分辨率为 11 位的 0~20mA 电流信号，负载阻抗小于等于  $600\Omega$ ，不需要附加的放大器。25°C 满量程的最大误差为  $\pm 0.5\%$ ，0~55°C 满量程的最大误差为  $\pm 1.0\%$ 。有超上限/超下限、电压模式对地短路和电流模式断线的诊断功能。

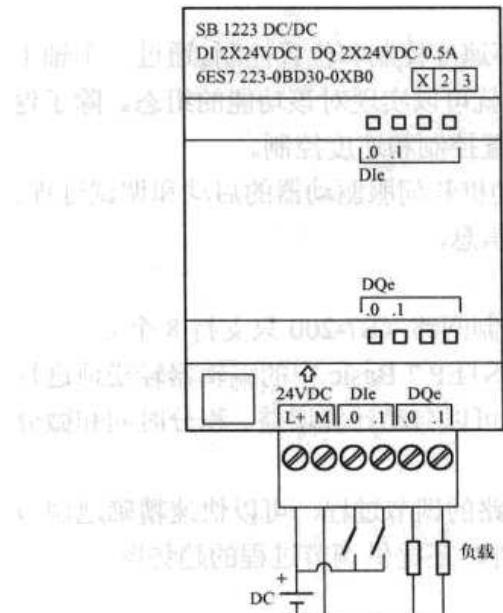


图 2-4 2DI/2DO 信号板

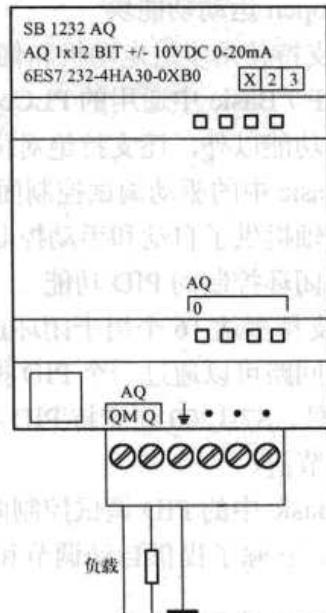


图 2-5 1AO 信号板

## 2. 数字量 I/O 模块

数字量输入/输出（DI/DO）模块和模拟量输入/输出（AI/AO）模块统称为信号模块。可以选用 8 点、16 点和 32 点的数字量输入/输出模块（见表 2-3），来满足不同的控制需要。

表 2-3 数字量输入/输出模块

型 号	各组输入点数	各组输出点数
SM 1221, 8 输入 DC 24V	4, 4	
SM 1221, 16 输入 DC 24V	4, 4, 4, 4	
SM 1222, 8 继电器输出, 2A		3, 5
SM 1222, 16 继电器输出, 2A		4, 4, 2, 6
SM 1222, 8 输出 DC 24V, 0.5A		4, 4
SM 1222, 16 输出 DC 24V, 0.5A		4, 4, 4, 4
SM 1223, 8 输入 DC 24V/8 继电器输出, 2A	4, 4	4, 4
SM 1223, 16 输入 DC 24V/16 继电器输出, 2A	8, 8	4, 4, 4, 4
SM 1223, 8 输入 DC 24V/8 输出 DC 24V, 0.5A	4, 4	4, 4
SM 1223, 16 输入 DC 24V/16 输出 24V DC, 0.5A	8, 8	8, 8

## 3. PLC 对模拟量的处理

在工业控制中，某些输入量（例如压力、温度、流量、转速等）是模拟量，某些执行机

构（例如电动调节阀和变频器等）要求 PLC 输出模拟量信号，而 PLC 的 CPU 只能处理数字量。模拟量首先被传感器和变送器转换为标准量程的电流或电压，例如  $4\sim20mA$ ,  $1\sim5V$ ,  $0\sim10V$ ，PLC 用模拟量输入模块的 A/D 转换器将它们转换成数字量。带正负号的电流或电压在 A/D 转换后用二进制补码来表示。

模拟量输出模块的 D/A 转换器将 PLC 中的数字量转换为模拟量电压或电流，再去控制执行机构。模拟量 I/O 模块的主要任务就是实现 A/D 转换（模拟量输入）和 D/A 转换（模拟量输出）。

A/D 转换器和 D/A 转换器的二进制位数反映了它们的分辨率，位数越多，分辨率越高。模拟量输入/输出模块的另一个重要指标是转换时间。

#### 4. 模拟量模块

S7-1200 现在有 3 种模拟量模块，此外还有计划中的热电阻/热电偶模块。

##### (1) 4 通道模拟量输入模块 SM 1231 AI $4\times13bit$

该模块的模拟量输入可选  $\pm10V$ 、 $\pm5V$  和  $\pm2.5V$  电压，或  $0\sim20mA$  电流。分辨率为 12 位+符号位，电压输入的输入电阻  $\geqslant 9M\Omega$ ，电流输入的输入电阻为  $250\Omega$ 。模块有中断和诊断功能，可监视电源电压和断线故障。所有通道的最大循环时间为  $625\mu s$ 。额定范围的电压转换后对应的数字为  $-27648\sim27648$ 。 $25^{\circ}C$  或  $0\sim55^{\circ}C$  满量程的最大误差为  $\pm 0.1\%$  或  $\pm 0.2\%$ 。

可按弱、中、强 3 个级别对模拟量信号作平滑（滤波）处理，也可以选择不作平滑处理。模拟量模块的电源电压均为 DC 24V。

##### (2) 2 通道模拟量输出模块 SM 1232 AQ $2\times14bit$

该模块的输入电压为  $-10\sim+10V$  时，分辨率为 14 位，最小负载阻抗  $1000\Omega$ 。输出电流为  $0\sim20mA$  时，分辨率为 13 位，最大负载阻抗  $600\Omega$ 。有中断和诊断功能，可监视电源电压、短路和断线故障。数字  $-27648\sim27648$  被转换为  $-10\sim+10V$  的电压，数字  $0\sim27648$  被转换为  $0\sim20mA$  的电流。

电压输出负载为电阻时转换时间为  $300\mu s$ ，负载为  $1\mu F$  电容时转换时间为  $750\mu s$ 。

电流输出负载为  $1mH$  电感时转换时间为  $600\mu s$ ，负载为  $10mH$  电感时为  $2ms$ 。

##### (3) 4 通道模拟量输入/2 通道模拟量输出模块

模块 SM 1234 的模拟量输入和模拟量输出通道的性能指标分别与 SM 1231 AI  $4\times13bit$  和 SM 1232 AQ  $2\times14bit$  的相同，相当于这两种模块的组合。

### 2.1.3 集成的通信接口与通信模块

#### 1. 集成的 PROFINET 接口

实时工业以太网是现场总线发展的趋势，现场总线的国际标准 IEC 61158 第 4 版的 20 种现场总线中，基于实时以太网的现场总线占了一半。PROFINET 是基于工业以太网的现场总线（IEC 61158 现场总线标准的类型 10），是开放式的工业以太网标准，它使工业以太网的应用扩展到了控制网络最底层的现场设备。

通过 TCP/IP 标准，S7-1200 提供的集成 PROFINET 接口可用于与编程软件 STEP 7 Basic 通信（见图 2-6），以及与 SIMATIC HMI 精简系列面板通信，或与其他 PLC 通信（见图 2-7）。此外它还通过开放的以太网协议支持与第三方设备的通信。该接口的 RJ-45 连接器具有自动

交叉网线（Auto-Cross-Over）功能，数据传输速率为 10Mbit/s/100 Mbit/s，支持最多 16 个以太网连接。该接口能实现快速、简单、灵活的工业通信。

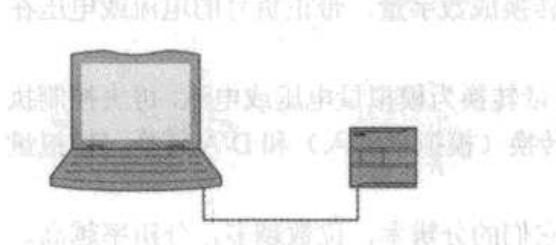


图 2-6 S7-1200 与计算机的通信

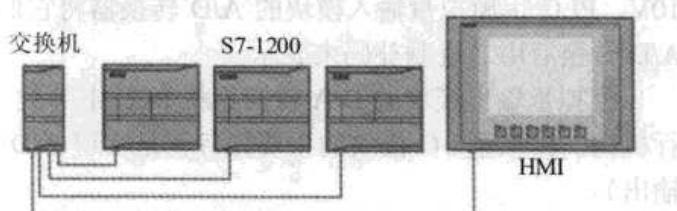


图 2-7 S7-1200 与 HMI 的通信

将来还可以通过 PROFINET 接口将分布式现场设备连接到 S7-1200，或将 S7-1200 作为一个 PROFINET IO 设备，连接到作为 PROFINET IO 主控制器的 PLC。它将为 S7-1200 系统提供从现场级到控制级的统一通信，以满足当前工业自动化的通信需求。

STEP 7 Basic 工程组态系统的网络视图使用户能够轻松地对网络进行可视化组态。

为了使布线最少并提供最大的组网灵活性，可以将紧凑型交换机模块 CSM 1277 和 S7-1200 一起使用，以便组建成一个具有线形、树形或星形拓扑结构的网络。

CSM 1277 是一个 4 端口的紧凑型交换机，用户可以通过它将 S7-1200 连接到最多 3 个附加设备。除此之外，如果将 S7-1200 和 SIMATIC NET 工业无线局域网组件一起使用，还可以构建一个全新的网络。

S7-1200 可以通过成熟的 S7 通信协议连接到多个 S7 控制器和 HMI 设备。S7-1200 上的集成接口不仅可以与其他厂商的设备进行无缝集成，还可以通过开放式以太网协议 TCP/IP 和 ISO-on-TCP 与多个第三方设备进行连接和通信。STEP 7 Basic 为 S7-1200 提供了 TSEND/TRCV 指令来实现上述的通信。

## 2. 通信模块

S7-1200 最多可以增加 3 个通信模块，它们安装在 CPU 模块的左边。

RS-485 和 RS-232 通信模块为点对点（PtP）的串行通信提供连接（见图 2-8）。STEP 7 Basic 工程组态系统提供了扩展指令或库功能、USS 驱动协议、Modbus RTU 主站协议和 RTU 从站协议，用于串行通信的组态和编程。

此外还有计划中的 PROFINET（控制器/IO 设备）模块和 PROFIBUS（主站/从站）模块。

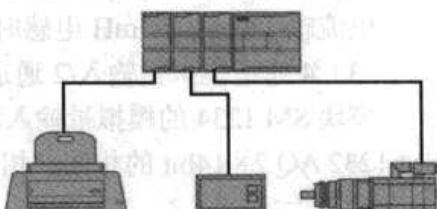


图 2-8 使用通信模块的串行通信

## 2.2 STEP 7 Basic 编程软件

### 2.2.1 STEP 7 Basic 的特点

SIMATIC STEP 7 Basic 是西门子公司开发的高集成度工程组态系统，包括面向任务的 HMI 智能组态软件 SIMATIC WinCC Basic。上述两个软件集成在一起，也被称为 TIA (Totally Integrated Automation, 全集成自动化) Portal，它提供了直观易用的编辑器，用于对 S7-1200 和精简系列面板进行高效组态。除了支持编程以外，STEP 7 Basic 还为硬件和网络组态、诊

断等提供通用的工程组态框架。

STEP 7 Basic 的操作直观、上手容易、使用简单，使用户能够对项目进行快速而简单的组态。由于具有通用的项目视图、用于图形化工程组态的最新用户接口技术、智能的拖放功能以及共享的数据处理等，有效地保证了项目的质量。

由于 STEP 7 Basic（包括 SIMATIC WinCC Basic）具有面向任务的智能编辑器，界面十分直观，因此它可以作为一个通用的工程组态软件框架，对 S7-1200 控制器进行编程和调试。功能强大的 HMI 软件 WinCC Basic 用于对精简系列面板进行高效的组态。

用户可以在两种不同的视图中选择一种最适合的视图：

1) 在入口（Portal）视图中，可以概览自动化项目的所有任务。初学者可以借助面向任务的用户指南，以及最适合其自动化任务的编辑器来进行工程组态。

2) 在项目视图中，整个项目（包括 PLC 和 HMI 设备）按多层结构显示在项目树中。本书主要使用项目视图。

可以使用拖放功能为硬件分配图标，组态连接设备的通信网络。用户可以在同一个工程组态软件框架下同时使用 HMI 和 PLC 编辑器，大大提高了效率。

图形编辑器保证了对设备和网络快速直观地进行组态，使用线条连接单个设备就可以完成对通信连接的组态。在线模式可以提供故障诊断信息。

该软件采用了面向任务的理念，所有的编辑器都嵌入到一个通用框架中。用户可以同时打开多个编辑器。只需轻点鼠标，便可以在编辑器之间切换。

软件能自动保持数据的一致性，可确保项目的高质量。经修改的应用数据在整个项目中自动更新。交叉引用的设计保证了变量在项目的各个部分以及在各种设备中的一致性，因此可以统一进行更新。系统自动生成图标并分配给对应的 I/O。数据只需输入一次，无需进行额外的地址和数据操作，从而降低了发生错误的风险。

通过本地库和全局库，用户可以保存各种工程组态的元素，例如块、变量、报警、HMI 的画面、各个模块和整个站。这些元素可以在同一个项目或在不同的项目中重复使用。借助全局库，可以在单独组态的系统之间进行数据交换。

常用的命令可以保存在一个收藏列表中，所有的工程组态模块可以复制并添加到其他 S7-1200 项目。

## 2.2.2 安装 STEP 7 Basic

### 1. STEP 7 Basic 对计算机的要求

SIMATIC STEP 7 Basic 和 SIMATIC WinCC Basic 是 TIA Portal 的组件。

安装 SIMATIC STEP 7 Basic 对计算机软硬件的最低要求如下：

- Pentium 4, 1.7 GHz 的 CPU 或性能相近的处理器。
- 内存容量：Windows XP 为 1GB，Windows Vista 为 2GB。
- 自由的硬盘空间：2GB。
- 操作系统：Windows XP (Home SP3 或 Professional SP3)，或 Windows Vista (Home Premium SP1, Business SP1, Ultimate SP1)。
- 显卡：32 MB RAM, 32 位彩色。
- 屏幕分辨率：1024×768。

- 网络：10Mbit/s/100M bit/s 以太网卡。

在安装过程中自动安装自动化许可证。卸载 TIA Portal 时，自动化许可证也被自动卸载。

## 2. 安装软件

可以将随书光盘中的文件夹“STEP 7 Basic V10.5”复制到硬盘后安装，保存软件的文件夹不要使用中文名称。

安装时应关闭所有打开的软件。双击文件夹中的 Start 图标（见图 2-9），开始安装软件。如果安装程序发现计算机没有安装微软公司的软件“.NET Framework 2.0 SP2”，将会出现图 2-9 中的对话框，点击“是”按钮开始安装该软件，点击“否”按钮将取消安装。

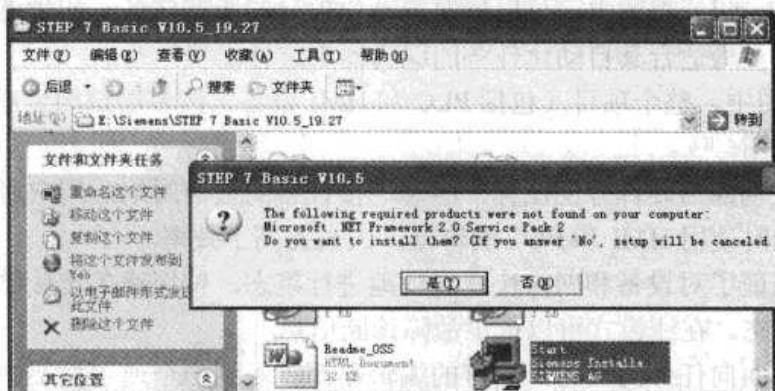


图 2-9 安装 STEP7 Basic

最初出现的视窗下面的“Initialization, this can take several minites...”，意思是通知用户安装程序正在作初始化操作，可能需要几分钟。

在图 2-10 所示的对话框中选择安装语言，默认的安装语言为英语，设置完成后，点击“Next”按钮，进入下一对话框。

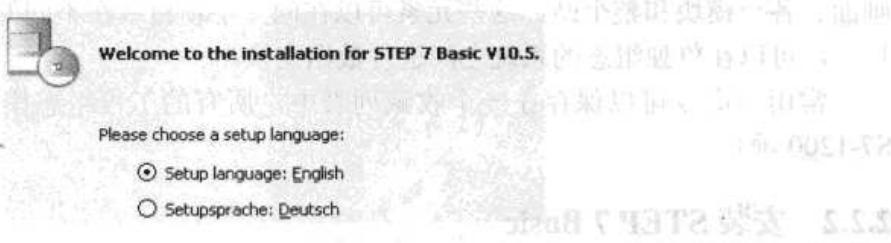


图 2-10 选择安装语言

在图 2-11 所示的对话框中选择产品（软件）使用的语言，默认的设置是只安装英语。点击按钮“*Yes, I would like to read Product Information*”，可以阅读产品信息文件。

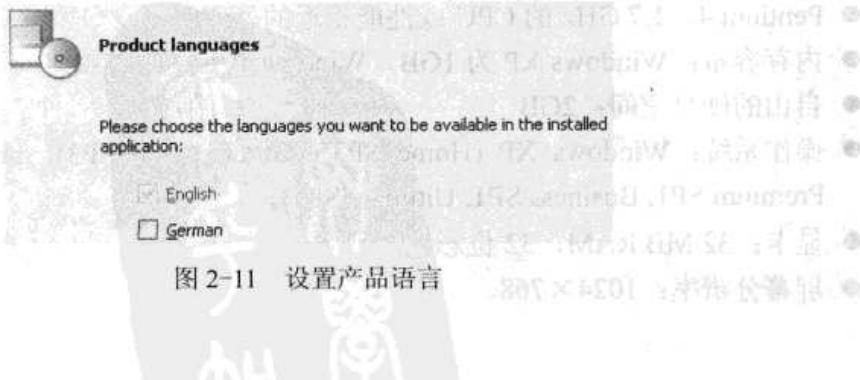


图 2-11 设置产品语言

图 2-12 右边中间的窗口显示出各硬盘分区总的存储空间和可供使用的存储空间，下面的文本框中是 C 盘中默认的安装路径。点击“Browse”（浏览）按钮，可以修改安装路径。点击“Back”（返回）按钮，返回上一对话框。

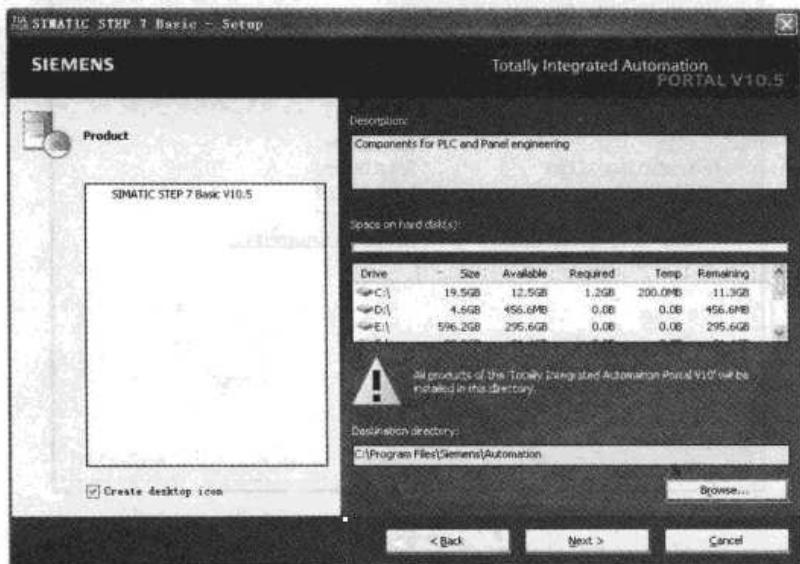


图 2-12 设置安装的文件夹

点击“Next”按钮，出现 Overview（总览）窗口，给出了前面设置的产品语言、安装的路径和将要安装的软件。

必须用鼠标点击总览窗口下面的正方形复选框“*I accept the conditions of the displayed license agreement(s)*”，使方框中出现“√”，接受显示的许可证协议的条款。点击“Install”（安装）按钮，开始安装。

图 2-13 是安装过程中的对话框，正在安装的软件用加粗的字体显示，右边有几个小圆图标。

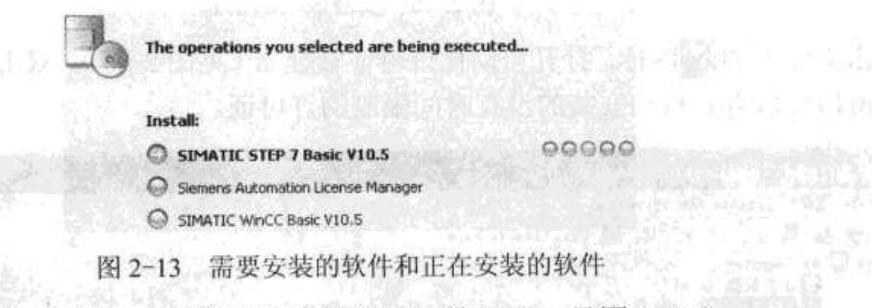


图 2-13 需要安装的软件和正在安装的软件

安装完 STEP 7 Basic 后，自动安装自动化许可证管理器（见图 2-14）。

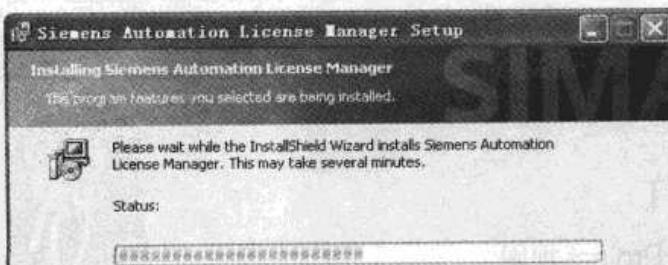


图 2-14 安装自动化许可证管理器

图 2-15 显示正在安装微软公司的 SQL 数据库服务器，最后安装 WinCC Basic。

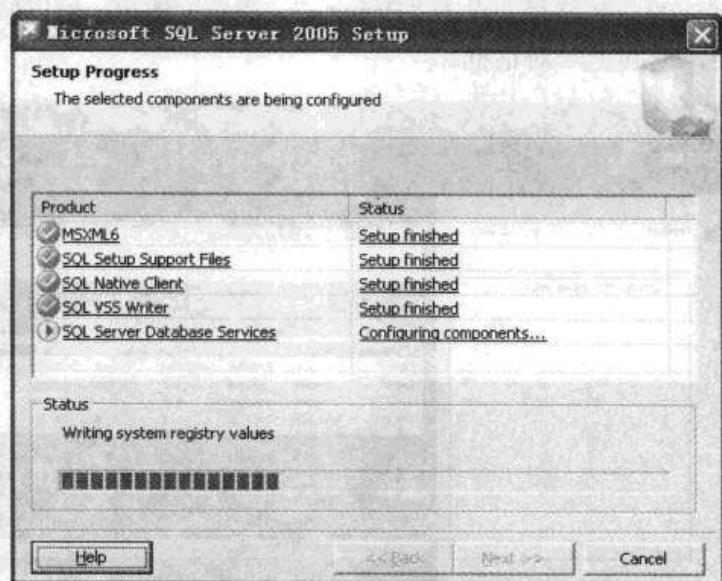


图 2-15 安装 SQL 服务器

图 2-16 是安装成功后显示的信息，默认的设置是立即重新启动计算机。点击“Restart”按钮，计算机被重新启动。

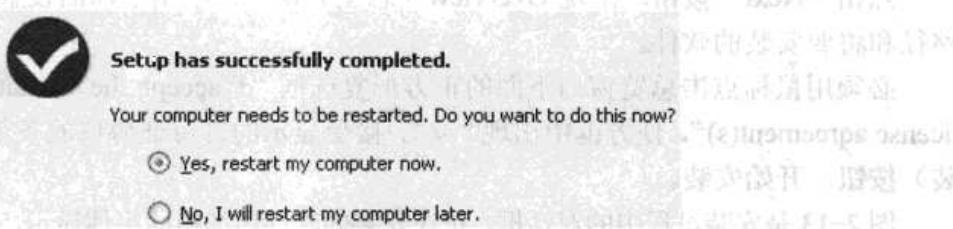


图 2-16 安装成功的信息

双击桌面上的图标，打开自动化许可证管理器（见图 2-17），双击左边窗口中的 C 盘，在右边窗口可以看到自动安装的没有时间限制的许可证。

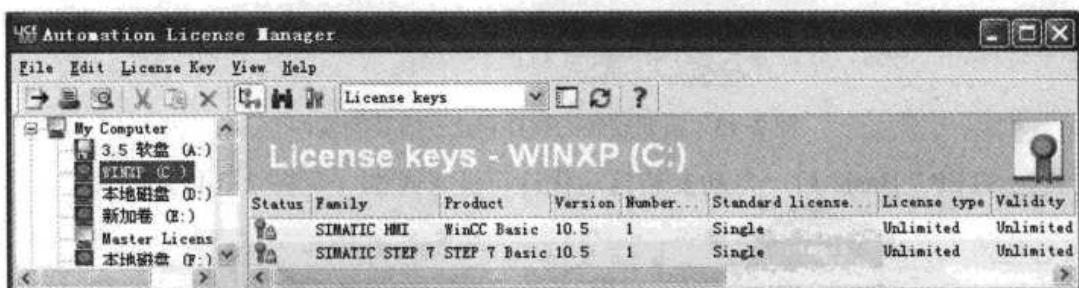


图 2-17 自动化许可证管理器

### 2.2.3 创建一个项目

#### 1. Portal 视图与 Project 视图

TIA Portal 提供两种不同的工具视图：基于项目的项目视图（Project view）和基于任

务的入口视图（Portal view），项目视图可以访问项目中所有的组件。本书主要使用项目视图。

双击桌面上的TIA PORTAL图标，打开STEP 7 Basic的启动画面，图2-18是启动画面的主要部分。点击视图左下角的“Project view”，切换到项目视图（见图2-19）。

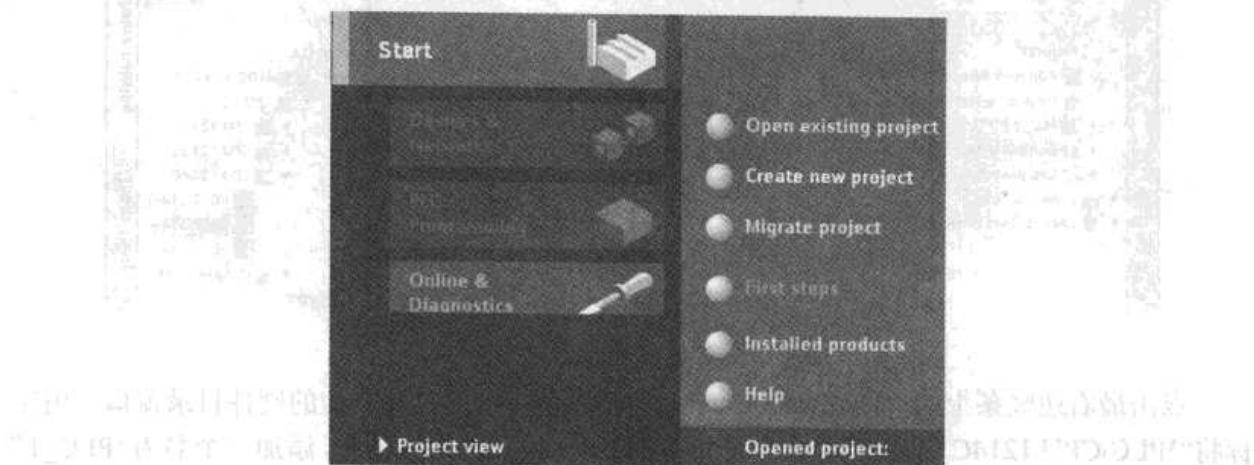


图 2-18 启动画面

## 2. 生成一个项目

执行菜单命令“Project”→“New”，在出现的生成新项目对话框中（见图2-19中的小图），可以修改项目的名称（Project name），或者使用系统指定的名称。点击“Path”（路径）输入框右边的“...”按钮，可以修改保存项目的路径。点击“Create”按钮，开始生成项目。

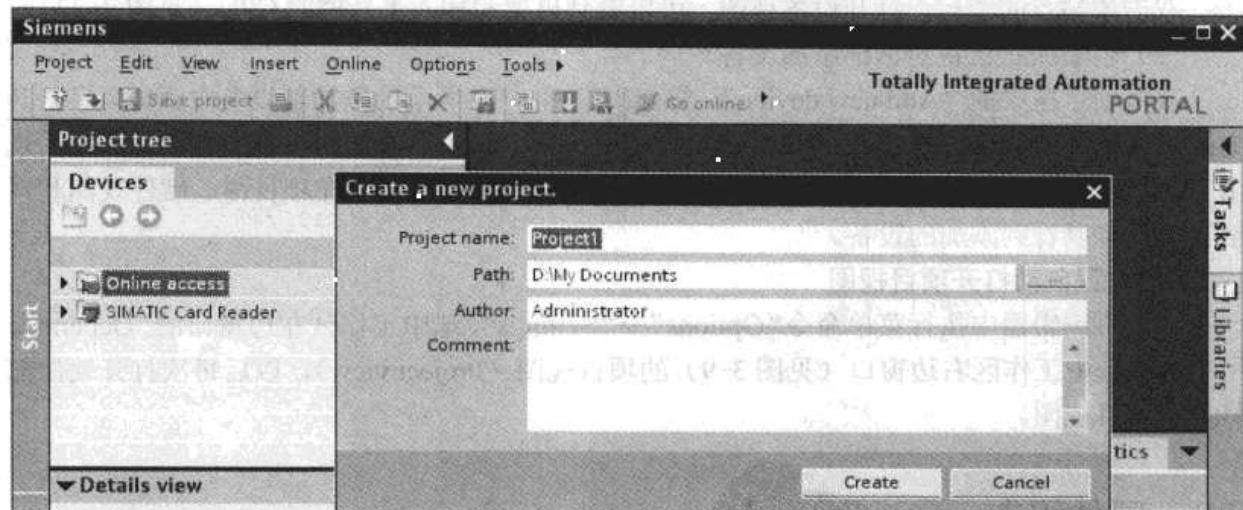


图 2-19 生成新项目

## 3. 添加 PLC 设备

### (1) 在网络视图中添加设备

生成项目后，双击图2-20左边窗口的项目树中的“Device & Network”，中间的工作区是打开的网络视图（Network view）。

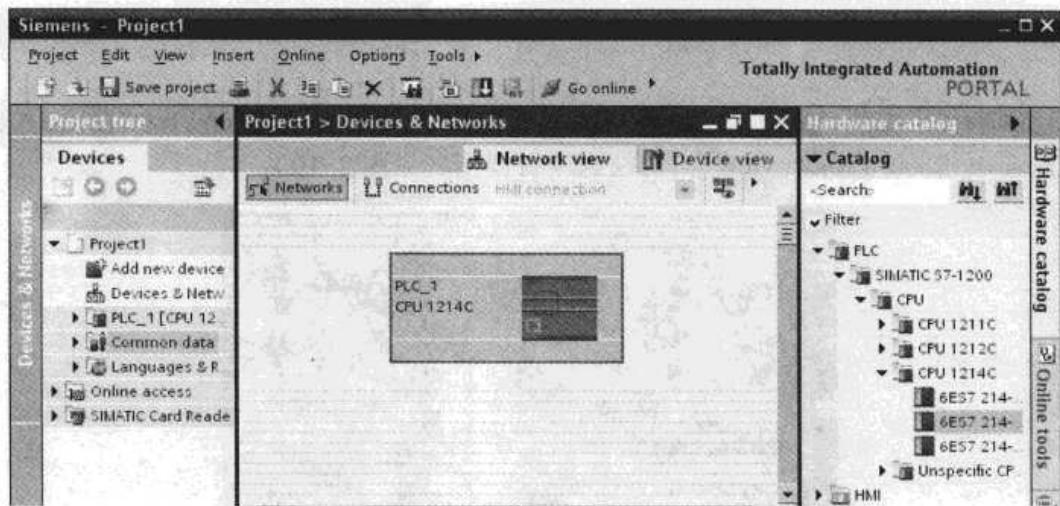


图 2-20 在网络视图中添加设备

点击最右边竖条上的“Hardware Catalog”(硬件目录)，打开右边的硬件目录窗口。用鼠标将“\PLC\CPU 1214C”文件夹中某个订货号的 CPU 拖到网络视图中，添加一个名为“PLC\_1”的设备。

没有移动到允许放置该模块的工作区时，光标的形状为 $\textcircled{\times}$ (禁止放置)。反之光标的形状变为 $\textcircled{+}$ (允许放置)。此时松开鼠标左键，被拖动的 CPU 模块被放置到工作区。

如果激活了硬件目录的过滤器功能(选中图 2-22 中硬件目录窗口上面的“Filter”复选框)，硬件目录只显示与工作区有关的硬件。例如用设备视图打开 PLC 的组态画面时，如果选中了过滤器，则硬件目录窗口不显示 HMI，只显示 PLC 的模块。

双击该站点的图标，打开设备视图，可以看到机架和插入 1 号槽的 CPU (见图 2-22)。

#### (2) 用添加新设备对话框添加设备

双击项目树中的“Add new device”，将会出现图 3-10 所示的添加新设备对话框。点击其中的“SIMATIC PLC”按钮或“SIMATIC HMI”按钮，选中要添加的设备的订货号，然后点击“OK”按钮，可以添加一个 S7-1200 PLC 或精简系列面板设备。在项目树、硬件视图和网络视图中可以看到添加的设备。

### 4. 设置自动打开项目视图

在项目编辑器中执行菜单命令“Options”→“Settings”，选中工作区左边窗口的“General”，用单选框选中工作区右边窗口(见图 3-9)的项目视图(Project view)，以后每次打开软件都将显示项目视图。

## 2.3 硬件组态

### 2.3.1 硬件组态概述

#### 1. 打开已有的项目

双击桌面上的 图标，打开 STEP 7 Basic 的项目视图。点击工具栏上的 按钮，双击打开的对话框中列出的最近打开的某个项目(见图 2-21)，打开该项目。或者点击“Browse”(浏览)按钮，用打开的对话框找到某个项目，双击项目文件夹中带有 的文件，打开该项目。

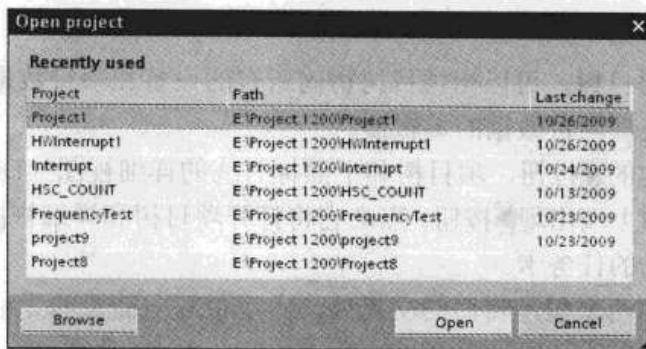


图 2-21 打开项目对话框

## 2. 设备组态的任务

英语单词“Configuring”（配置、设置）一般被翻译为“组态”。设备组态的任务就是在设备与组态编辑器中生成一个与实际的硬件系统完全相同的虚拟系统，包括系统中的设备（PLC 和 HMI），PLC 各模块的型号、订货号和版本，模块的安装位置和设备之间的通信连接，都应与实际的系统完全相同。

此外还应设置模块的参数，即给参数赋值，或称为参数化。

自动化系统启动时，CPU 比较组态时生成的虚拟系统和系统实际的硬件系统，如果两个系统不一致，则采取相应的措施。

双击项目视图的项目树中的“Device & Network”，打开设备与网络编辑器。

### 2.3.2 项目视图的结构

下面介绍图 2-22 的项目视图各组成部分的功能。

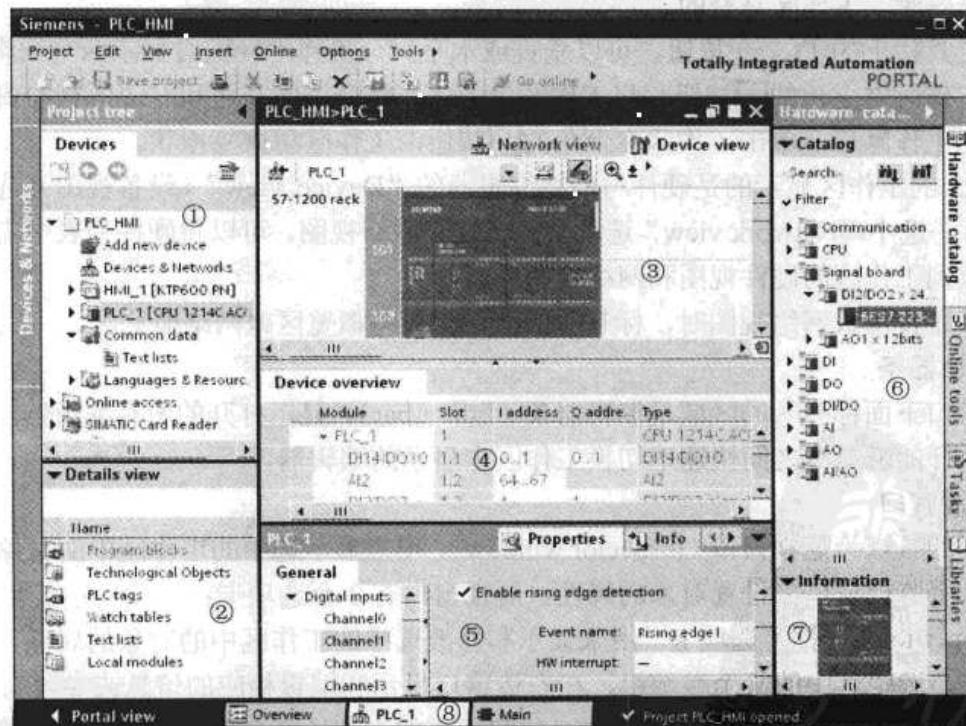


图 2-22 在项目视图中组态硬件

## 1. 项目树

标有①的区域为项目树，可以用项目树访问所有的设备和项目数据，添加新的设备，编辑已有的设备，打开处理项目数据的编辑器。

点击项目树右上角的■按钮，项目树和下面标有②的详细视图（Details view）消失，同时在最左边的垂直条的上端出现▶按钮。点击它将打开项目树和详细视图。可以用类似的方法隐藏和显示右边标有⑥的任务卡。

将鼠标的光标放到两个显示窗口的交界处，出现带双向箭头的光标时，按住鼠标的左键移动鼠标，可以移动分界线，以调节分界线两边的窗口大小。

## 2. 详细视图

项目树窗口下面标有②的区域是详细视图，详细视图显示项目树被选中的对象的内容（文本或变量列表）。图 2-22 中的详细视图显示的是项目树的“PLC\_1”文件夹中的内容。使用拖放功能，可以将详细视图中的对象拖放到需要它的位置，例如将变量拖放到梯形图中。

详细视图窗口不能显示文件夹中的内容，例如不能显示“PLC tags”中的内容。只能用项目树或概览窗口显示文件夹中的内容。

点击详细视图左上角的■按钮，详细视图被关闭，只剩下紧靠“Portal view”的标题，标题左边的按钮变为▶。点击该按钮，将重新显示详细视图。可以用类似的方法显示和隐藏标有⑤的监视窗口和标有⑦的信息窗口。

## 3. 工作区

标有③的区域为工作区，工作区显示一个当前打开的编辑器。可以同时打开几个编辑器，但是同时只能在工作区显示一个编辑器。其他编辑器在最下面标有⑧的编辑器条中显示。没有打开编辑器时，工作区是空的。

单击工具栏上的■、□按钮，可以垂直或水平拆分工作区，同时显示两个编辑器。

单击工作区右上角的□按钮，将工作区最大化，将会关闭左边和右边所有的窗口。

最大化工作区后，单击工作区右上角的□按钮，工作区将恢复原状。

图 2-22 的工作区显示的是硬件与网络编辑器的“Device view”（设备视图）选项卡，可以组态硬件。选中“Network view”选项卡，将打开网络视图。可以将硬件列表中需要的设备或模块拖放到工作区的硬件视图和网络视图中。

显示设备视图或网络视图时，标有④的区域为设备概览区或网络概览区。

## 4. 编辑器条

监视窗口下面标有⑧的区域是编辑器条（Editor bar），显示打开的所有编辑器，可以用编辑器条在打开的编辑器之间快速地切换工作区中显示的编辑器。

## 5. 监视窗口

标有⑤的区域为监视窗口（Inspector window），用来显示选中的工作区中的对象附加的信息，还可以用监视窗口来设置对象的属性。监视窗口有 3 个选项卡：

1) Properties（属性）选项卡。用来显示和修改选中的工作区中的对象的属性。左边窗口是浏览窗口，选中其中的某个参数组，在右边窗口显示和编辑相应的信息或参数。

2) Info（信息）选项卡。显示所选对象和操作的详细信息，以及编译的报警信息。

3) Diagnostics（诊断）选项卡。显示系统诊断事件和组态的报警事件。

## 6. 任务卡

标有⑥的区域为任务卡 (Task card)，任务卡的功能与编辑器有关。可以通过任务卡进行进一步的或附加的操作。例如从库或硬件目录中选择对象，搜索与替代项目中的对象，将预定义的对象拖放到工作区。

可以用最右边的竖条上的按钮来切换任务卡显示的内容。图 2-22 中的任务卡显示的是硬件目录 (Hardware catalog)，任务卡的下面标有⑦的区域是选中的硬件对象的信息 (Information) 窗口，包括对象的图形、名称、版本号、订货号和简要的描述。

### 2.3.3 硬件组态的操作

#### 1. 在设备视图中添加模块

双击项目树中的 PLC 设备对象，打开设备视图后，可以看到 1 号插槽中的 CPU 模块。

在硬件组态时需要将 I/O 模块或通信模块放置到工作区的机架上的插槽内，有两种放置硬件对象的方法。

##### (1) 用“拖放”的方法放置硬件对象

用鼠标打开硬件目录中的文件夹，选中订货号为 6ES7 221-1BH30-0XB0 的 8 点 DI 模块，其背景变为深色。所有可以插入该模块的插槽四周出现深蓝色的方框，只能将该模块插入这些插槽。用鼠标左键按住该模块不放，移动鼠标，将选中的模块“拖”到机架 CPU 右边的 2 号插槽，该模块浅色的图标和订货号随着光标一起移动。

##### (2) 用双击的方法放置硬件对象

放置模块还有另一个简便的方法，首先用鼠标左键点击机架中需要放置模块的插槽，使它的四周出现深蓝色的边框。用鼠标左键双击硬件目录中要放置的模块，该模块便出现在选中的插槽。

放置通信模块和信号板的方法与放置信号模块的方法相同，信号板安装在 CPU 模块内，通信模块安装在 CPU 左侧的 101~103 号槽。

可以将信号模块插入已经组态的两个模块中间。插入点右边的模块将向右移动一个插槽的位置，新的模块被插入到空出来的插槽。

#### 2. 删除硬件组件

可以删除设备视图或网络视图中的硬件组件。被删除的组件的地址可供其他组件使用。不能单独删除 CPU 和机架，只能在网络视图或项目树中删除整个 PLC 站。

用鼠标右键点击要删除的硬件组件，执行出现的快捷菜单中的“Delete”命令，选中的硬件被删除。删除硬件组件后，可能在项目中产生矛盾，即违反插槽规则。选中指令树中的“PLC\_1”，点击工具栏上的  按钮，对硬件组态进行编译。编译时进行一致性检查，如果有错误将会显示错误信息，应改正错误后重新进行编译。

#### 3. 复制与粘贴硬件组件

可以在项目树、网络视图或硬件视图中复制硬件组件，然后将保存在剪贴板上的组件粘贴到其他地方。可以在网络视图中复制和粘贴站点，在硬件视图中复制和粘贴模块。

用鼠标右键点击要复制的硬件组件，执行出现的快捷菜单中的“Copy”命令，选中的硬件被复制到剪贴板。用右键点击要粘贴的剪贴板上的组件，执行出现的快捷菜单中的“Paste”命令，组件被粘贴在选中的位置。可以将剪贴板上的模块粘贴到网络视图中的其他站点。模

块将被插入第一个允许的插槽。

可以用拖放的方法或通过剪贴板在硬件设备视图或网络视图中移动硬件组件，但是不能移动 CPU，因为它必须在 1 号槽。

#### 4. 改变设备的型号

用鼠标右键点击要更改型号的 CPU，执行出现的快捷菜单中的“Change device type”命令，出现“Change device type”对话框。选中设备列表中用来替换的设备的订货号，点击“OK”按钮，设备型号被更改。

#### 5. 建立设备之间的通信连接

首先生成图 2-23 中的 PLC 和 HMI 设备，双击项目树的 PLC\_1 文件夹中的“Device Configuration”，打开工作区中的“Network view”（网络视图）选项卡，选中 CPU 左下角表示以太网接口的绿色小方框。按住鼠标左键不放，将它“拖”到 HMI 图形中表示以太网接口的绿色小方框上，将会出现图中所示的绿色的以太网线，和名称为“PN/IE\_1”的连接。

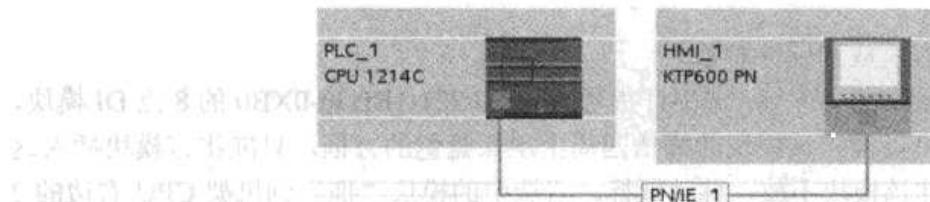


图 2-23 建立连接

### 2.3.4 信号模块与信号板的参数设置

#### 1. 信号模块与信号板的地址分配

双击项目树的某个 PLC 文件夹中的“Device configuration”，打开该 PLC 的设备视图。添加了 CPU、信号板或信号模块后，它们的 I、Q 地址是自动分配的。选中工作区中的 CPU，在工作区下面的“Device overview”（设备概览）区，可以看到 CPU 集成的 I/O 模块和信号模块的字节地址（见图 2-24）。例如 CPU 1214C 集成的 14 点数字量输入的字节地址为 0 和 1 (I0.0~I0.7 和 I1.0~I1.5)，10 点数字量输出的字节地址为 0 和 1 (Q0.0~Q0.7 和 Q1.0~Q1.1)。CPU 的模拟量输入地址为 IW64 和 IW66（每个通道占一个字或两个字节）。DI2/DO2 信号板的地址为 I4.0~I4.1 和 Q4.0~Q4.1。

Device overview					
Module	Slot	I address	Q address	Type	
PLC_1	1			CPU 1214C ADDCINV	
DI14/DO10	1.1	0..1	0..1	DI14/DO10	
AI2	1.2	64..67		AI2	
DI2/DO2 x 24VDC	1.3	4	4	DI2/DO2 signal board	

图 2-24 CPU 集成 I/O 的字节地址

DI、DO 的地址以字节为单位分配，如果没有用完分配给它的某个字节中所有的位，剩余的位也不能再作它用。

模拟量输入、输出的地址以组为单位分配，每一组有两个输入/输出点。

从图 2-25 的设备概览区可以看出，2 号槽的 8 点 DI 模块的地址为 I8.0~I8.7（字节地址为 8），3 号槽的 8 点 DO 模块的地址为 Q12.0~Q12.7，4 号槽的 4 点 AI 模块的地址为 IW128~IW134，5 号槽的 2 点 AO 模块的地址为 QW160 和 QW162。

选中设备概览中 2 号槽的 8 点 DI 模块，再选中下面监视窗口左边的“IO addresses”（IO 地址），可以修改右边窗口的“Start address”（模块的起始地址）。也可以用设备视图中的设备概览表或网络视图中的“Network overview”（网络概览）表来修改自动分配的 I、Q 地址。建议使用自动分配的地址，不要修改它们。

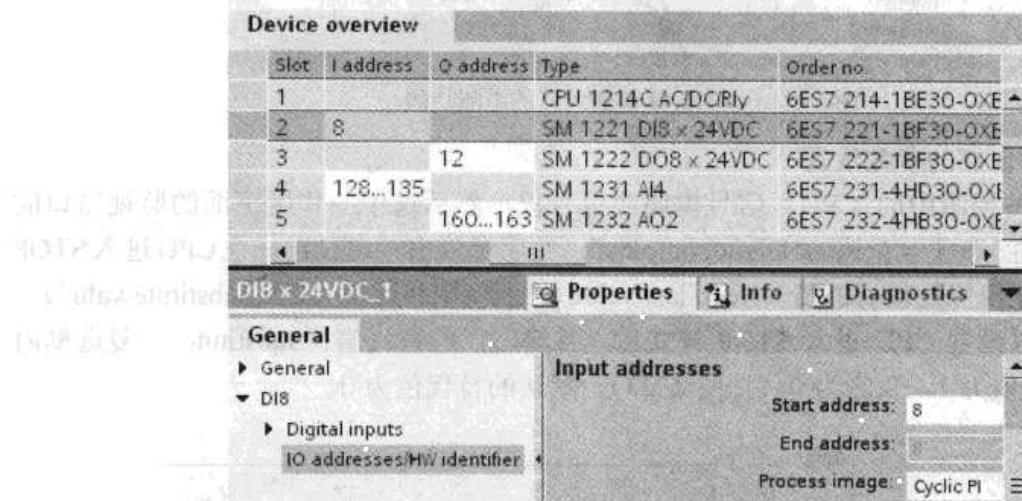


图 2-25 信号模块的地址

## 2. 数字量输入点的参数设置

首先选中设备视图中的 CPU、信号模块或信号板，然后选中工作区下面的监视窗口的“Properties”（属性）选项卡左边的“Digital inputs”（见图 2-26），可以用选择框分组设置输入点的滤波器的时间常数（0.2~12.8ms）。

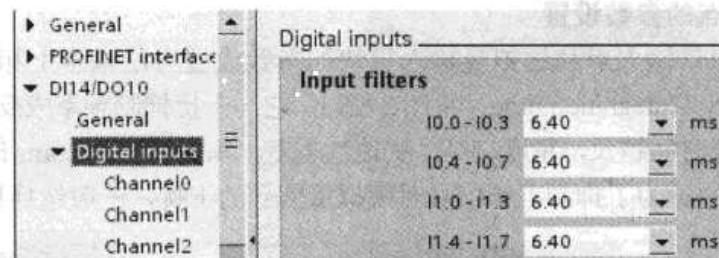


图 2-26 组态数字量输入点的滤波器

选中 CPU 和信号板的某个输入点后（见图 2-27），可以激活 CPU 和信号板各输入点的上升沿（Rising Edge）中断和下降沿（Falling Edge）中断功能，以及设置产生中断事件时调用的硬件中断 OB（HW interrupt）。

选中 CPU 和信号板的某个输入点后（见图 2-27），可以激活该输入点的脉冲捕获（Pulse Catch）功能，即暂时保持窄脉冲的 ON 状态，直到下一次刷新输入过程映像。

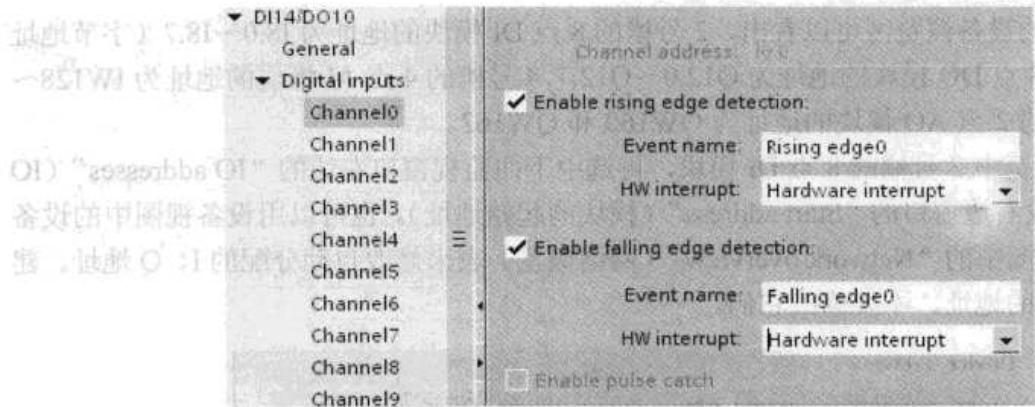


图 2-27 组态数字量输入点的中断功能

### 3. 数字量输出点的参数设置

首先选中设备视图中的 CPU、信号模块或信号板，然后选中工作区下面的监视窗口的“Properties”（属性）选项卡左边的“Digital outputs”（数字量输出），可以选择在CPU进入STOP模式时，数字量输出保持最后的值（Keep last value），或使用替代值（Use substitute value）。选中后者时，可以设置CPU进入STOP模式后，各输出点的替代值（Substitute）。复选框内有“√”表示替代值为1，反之为0（见图 2-28）。默认的替代值为0。

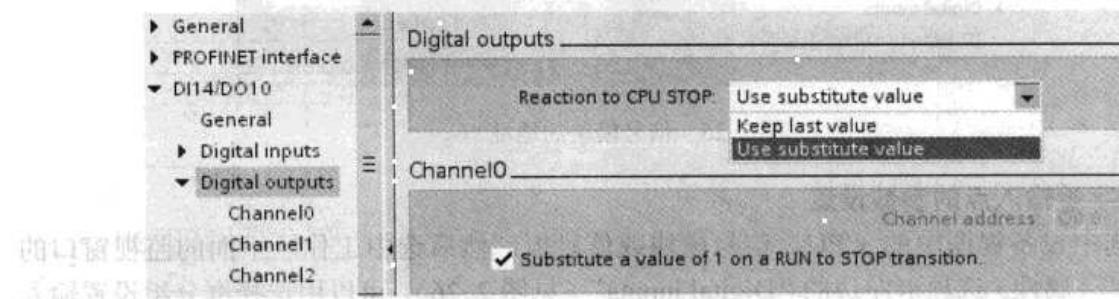


图 2-28 组态数字量输出点

### 4. 模拟量输入点的参数设置

CPU 集成的模拟量输入点和模拟量输入模块的参数设置方法基本上相同。

1) 设置积分时间（Integration time，见图 2-29），它与干扰抑制频率成反比。后者可选 400、60、50 和 10Hz。积分时间越长，精度越高，快速性越差。积分时间为 20ms 时，对 50Hz 的干扰噪声有很强的抑制作用。为了抑制工频信号对模拟量信号的干扰，一般选择积分时间为 20ms。

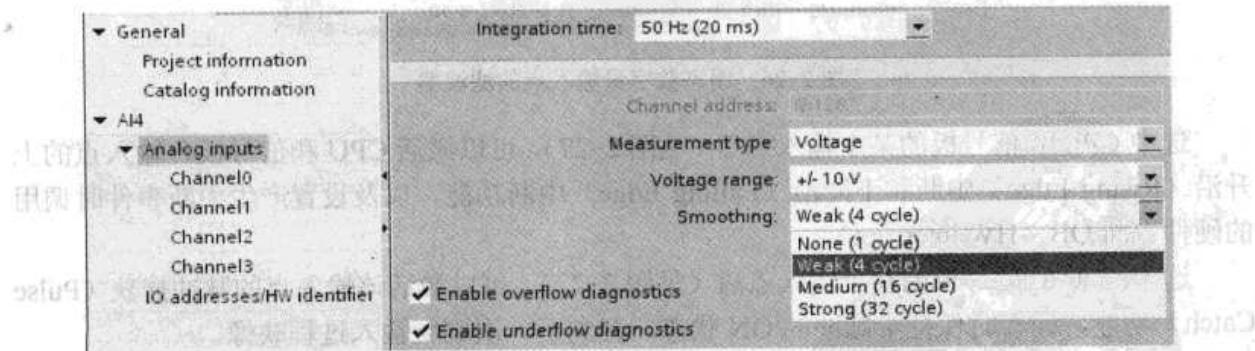


图 2-29 组态模拟量输入点

- 2) 设置测量种类 (Measurement type) 和测量范围 (例如电压范围 Voltage range)。
- 3) 设置 A/D 转换得到的模拟值的平滑 (Smoothing) 等级。

模拟值的滤波处理可以减轻干扰的影响，这对缓慢变化的模拟量信号（例如温度测量信号）是很有意义的。滤波处理用平均值数字滤波来实现，即根据系统规定的转换次数来计算转换后的模拟值的平均值。用户可以在滤波的 4 个等级 None (无)、Weak (弱)、Medium (中)、Strong (强) 中进行选择。这 4 个等级对应的计算平均值的模拟量采样值的个数分别为 1、4、16 和 32。所选的滤波等级越高，滤波后的模拟值越稳定，但是测量的快速性越差。

- 4) 设置诊断功能，可以激活超出上限值 (Overflow) 或低于下限值 (Underflow) 时的诊断功能。

## 5. 模拟量输出点的参数设置

信号板的模拟量输出点和模拟量输出模块的参数设置方法基本上相同（见图 2-30）。

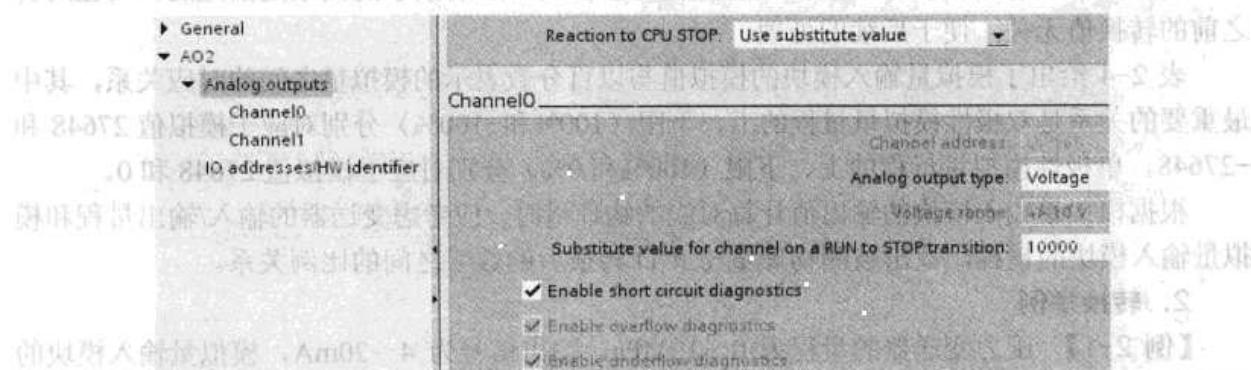


图 2-30 组态模拟量输出点

与数字量输出相同，可以设置 CPU 进入 STOP 模式后，各输出点保持最后的值 (Keep last value)，或使用替代值 (Use substitute value)。选中后者时，可以设置各点的替代值。

可以设置各输出点的输出类型（电压或电流）和输出范围。

可以激活电压输出的短路诊断功能，以及超出上限值 32511 (Overflow，见表 2-4) 或低于下限值 -32512 (Underflow) 的诊断功能。

### 2.3.5 将模拟量输入模块的输出值转换为实际的物理量

#### 1. 模拟量输入转换后的模拟值表示方法

模拟量输入/输出模块中模拟量对应的数字称为模拟值，模拟值用 16 位二进制补码（整数）来表示。最高位（第 15 位）为符号位，正数的符号位为 0，负数的符号位为 1。

表 2-4 模拟量输入模块的模拟值

范围	双极性				单极性			
	十进制	十六进制	百分比	±10V	十进制	十六进制	百分比	0~20 mA
上溢出，断电	32767	7FFFH	118.515%	11.851 V	32767	7FFFH	118.515%	23.70 mA
超出范围	32511	7EFFH	117.589%	11.759 V	32511	7EFFH	117.589%	23.52 mA
正常范围	27648	6C00H	100.000%	10 V	27648	6C00H	100.000%	20 mA
	0	0H	0 %	0 V	0	0H	0 %	0 mA

范围	双极性				单极性			
	十进制	十六进制	百分比	±10V	十进制	十六进制	百分比	0~20 mA
正常范围	-27648	9400H	-100.000%	-10 V	-11.759 V			
低于范围	-32512	8100H	-117.593%					
下溢出，断电	-32768	8000H	-118.519%	-11.851 V				

模拟量经 A/D 转换后得到的数值的位数（包括符号位）如果小于 16 位（包括符号位），则转换值被自动左移，使其最高位（符号位）在 16 位字的最高位，模拟值左移后未使用的低位则填入“0”，这种处理方法称为“左对齐”。设模拟值的精度为 12 位加符号位，左移 3 位后未使用的低位（第 0~2 位）为 0，相当于实际的模拟值被乘以 8。

这种处理方法的优点在于模拟量的量程与移位处理后的数字的关系是固定的，与左对齐之前的转换值无关，便于后续的处理。

表 2-4 给出了模拟量输入模块的模拟值与以百分数表示的模拟量之间的对应关系，其中最重要的关系是双极性模拟量量程的上、下限（100% 和 -100%）分别对应于模拟值 27648 和 -27648。单极性模拟量量程的上、下限（100% 和 0%）分别对应于模拟值 27648 和 0。

根据模拟量输入模块的输出值计算对应的物理量时，应考虑变送器的输入/输出量程和模拟量输入模块的量程，找出被测物理量与 A/D 转换后的数字之间的比例关系。

## 2. 转换举例

**【例 2-1】** 压力变送器的量程为 0~10MPa，输出信号为 4~20mA，模拟量输入模块的量程为 4~20mA，转换后的数字量为 0~27648，设转换后得到的数字为 N，试求以 kPa 为单位的压力值。

解：0~10Mpa(0~10 000kPa) 对应于转换后的数字 0~27648，转换公式为

$$P = 10000 \times N / 27648 \text{ kPa}$$

注意在运算时一定要先乘后除，否则会损失原始数据的精度。

**【例 2-2】** 某温度变送器的量程为 -100℃~500℃，输出信号为 4~20mA，某模拟量输入模块将 0~20mA 的电流信号转换为数字 0~27648，设转换后得到的数字为 N，求以 0.1℃ 为单位的温度值。

单位为 0.1℃ 的温度值 -1000~5000 对应于数字量 5530~27648，根据比例关系（见图 2-31），得出温度 T 的计算公式为

$$\begin{aligned} \frac{T - (-1000)}{N - 5530} &= \frac{5000 - (-1000)}{27648 - 5530} \\ T &= \frac{6000 \times (N - 5530)}{22118} - 1000 \quad (0.1^\circ\text{C}) \end{aligned}$$

## 2.3.6 CPU 模块的参数设置

双击项目树某个 PLC 文件夹中的“Device configuration”，打开该 PLC 的设备视图。选中

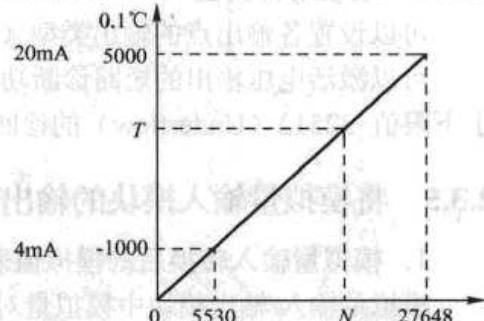


图 2-31 模拟量与转换值的关系

CPU 后，再选中下面的监视窗口左边的某个组对象，可以在右边的窗口设置有关的参数。

CPU 集成的 I/O 和信号板的参数设置方法已在 2.3.4 节介绍过了，集成的 PROFINET 接口、HSC（高速计数器）和脉冲发生器（Pulse generators）的参数设置方法将在有关的章节介绍。本节主要介绍 CPU 其他参数的设置方法。

### 1. 设置系统存储器字节与时钟存储器字节

选中图 2-32 所示的监视窗口左边的“System and clock memory”，点击右边窗口的复选框“Enable the use of system memory byte”（允许使用系统存储器字节），采用默认的 MB1 作系统存储器字节。可以修改系统存储器字节的地址。

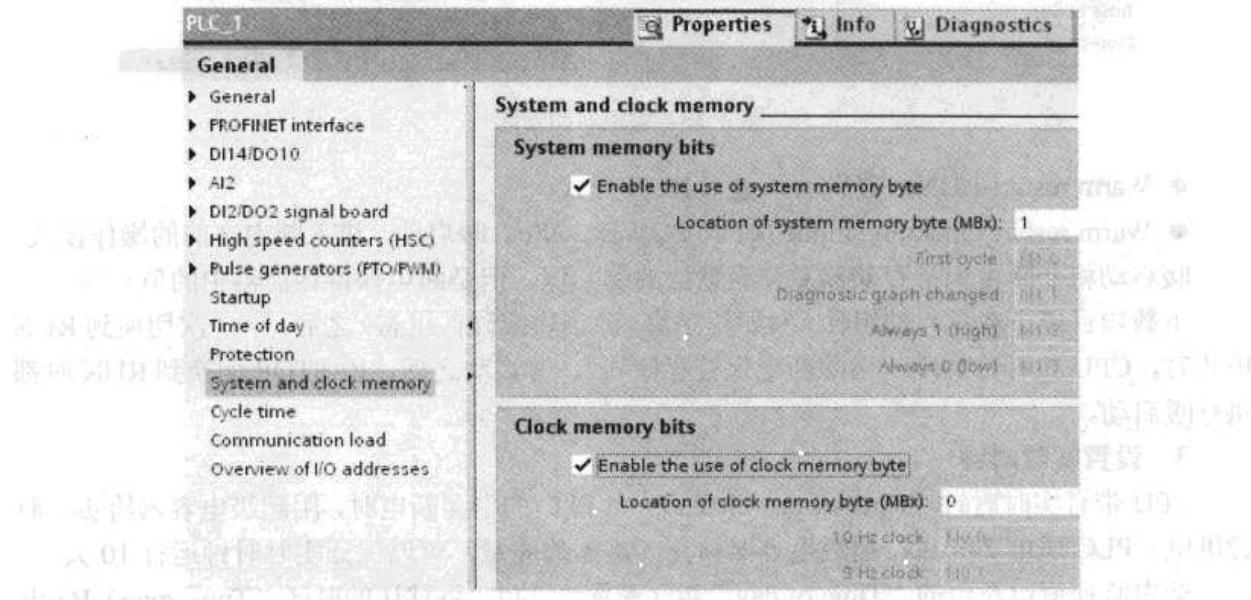


图 2-32 组态系统存储器字节与时钟存储器字节

将 MB1 设置为系统存储器字节后，该字节的 M1.0~M1.3 的意义如下：

- M1.0(First cycle): 仅在进入 RUN 模式的首次扫描时为 1 状态，以后为 0 状态。
- M1.1(Diagnostic graph changed): CPU 登录了诊断事件时，在一个扫描周期内为 1 状态。
- M1.2(Always 1): 总是为 1 状态，其常开触点总是闭合。
- M1.3(Always 0): 总是为 0 状态，其常闭触点总是闭合。

选中图 2-32 右边窗口的复选框“Enable the use of clock memory byte”（允许使用时钟存储器字节），设置用默认的 MB0 作时钟存储器字节。可以修改时钟存储器字节的地址。

时钟脉冲是一个周期内 0 状态和 1 状态所占的时间各为 50% 的方波信号，时钟存储器字节每一位对应的时钟脉冲的周期或频率见表 2-5。CPU 在扫描循环开始时初始化这些位。

以 M0.5 为例，其时钟脉冲的周期为 1s，如果用它的触点来控制接在某输出点的指示灯，指示灯将以 1Hz 的频率闪动，亮 0.5s，熄灭 0.5s。

表 2-5 时钟存储器各位对应的时钟脉冲的周期与频率

位	7	6	5	4	3	2	1	0
周期/s	2	1.6	1	0.8	0.5	0.4	0.2	0.1
频率/Hz	0.5	0.625	1	1.25	2	2.5	5	10

指定了系统存储器和时钟存储器字节后，这些字节不能再作它用，否则将会使用户程序运行出错，甚至造成设备损坏或人身伤害。

因为系统存储器和时钟存储器不是保留的存储器，用户程序或通信可能改写这些存储单元，破坏其中的数据。应避免改写这两个 M 字节，保证它们的功能正常运行。

## 2. 设置 PLC 上电后的启动方式

选中监视窗口左边的 Startup 组（见图 2-33），可以组态上电后 CPU 的 3 种启动方式：

- No restart (stay in STOP mode): 不启动，保持在 STOP 模式。

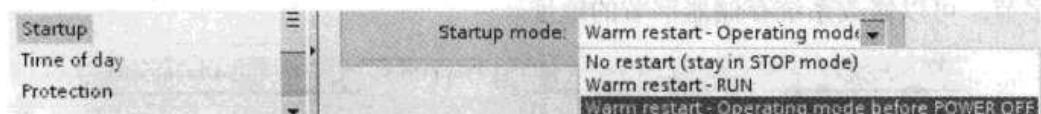


图 2-33 设置启动方式

- Warm restart-RUN: 暖启动，进入 RUN 模式。
- Warm restart-Operating mode before POWER OFF: 暖启动，进入断电之前的操作模式。

暖启动将非断电保持存储器复位为默认的起始值，但是断电保持存储器中的值不变。

下载项目或下载项目的组件（例如程序块、数据块或硬件组态）之后，下一次切换到 RUN 模式时，CPU 执行冷启动（清除断电保持存储器）。冷启动之后，由 STOP 切换到 RUN 时都执行暖启动。

## 3. 设置实时时钟

CPU 带有实时时钟（Time-of-day clock）。在 PLC 的电源断电时，用超级电容器给实时时钟供电。PLC 通电 24h 后，超级电容器被充了足够的能量，可以保证实时时钟运行 10 天。

选中监视窗口左边的“Time of day”组（见图 2-34），将默认的时区（Time zone）Berlin（柏林）改为 Beijing。我国目前没有使用夏时制（Daylight saving time）。

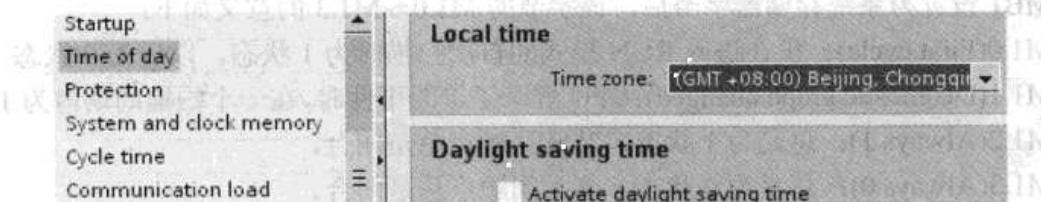


图 2-34 设置实时时钟的时区

在线模式时，双击项目树中某个 PLC 文件夹内的“Online and diagnostics”，选中左边窗口的“Set time of day”，可以设置 CPU 的实时时钟的时间（见图 7-13）。

## 4. 设置读写保护和密码

选中监视窗口左边的“Protection”（保护，见图 2-35），可以选择右边窗口的 3 个保护级别：

- 1) No protection 是默认的级别，没有设置口令保护。
- 2) Write protection 为写保护。输入正确的口令后才能修改 CPU 中的数据，并改变 CPU 的运行模式。
- 3) Read/write protection 为读/写保护。

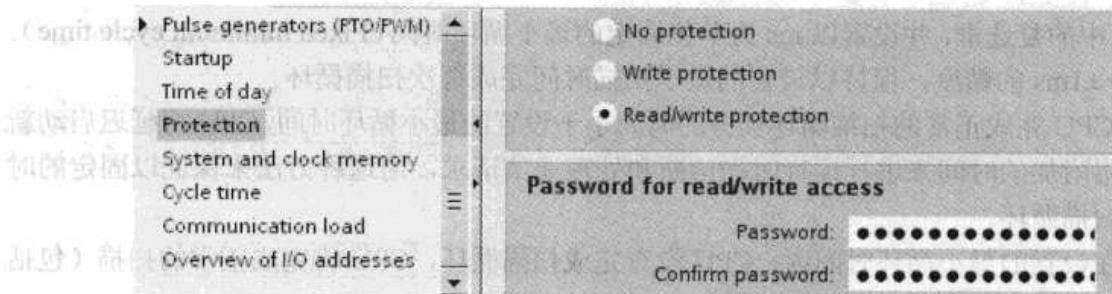


图 2-35 设置读写保护与口令

被授权（知道口令）的用户可以进行读/写访问。不知道口令的人员，只能读有写保护的 CPU，不能访问有读/写保护的 CPU。口令区分大小写。

为了限制对 CPU 的访问，应选中读保护或读/写保护，并输入口令（Password）和输入确认的口令（Confirm password）。

使用通信指令的 PLC 之间的通信和 HMI 的功能不受 CPU 的保护级别的限制。

## 5. 设置循环时间和通信负载

循环时间是操作系统刷新过程映像和执行程序循环 OB 的时间，包括所有中断此循环的程序的执行时间。每次循环的时间并不相等。

CPU 提供两个参数来监视循环时间：最大扫描循环时间和固定的最小扫描循环时间。启动阶段结束后，开始扫描循环监视。在组态 CPU 的属性时选中左边窗口的“Cycle time”（见图 2-36），可以组态这两个参数。

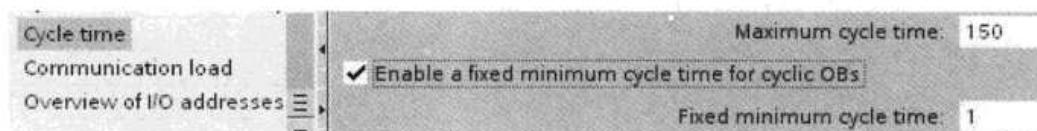


图 2-36 设置循环时间

如果循环时间超过最大循环时间，CPU 将调用 OB80。如果没有下载 OB80，将忽略第一次超过循环时间的事件。

如果循环时间超过最大循环时间的两倍，并且没有执行 RE\_TRIGR 指令来复位监控定时器，不管是否有 OB80，CPU 将立即进入 STOP 模式。

不能结束的循环指令和非常长的扫描时间可能会导致反复调用 RE\_TRIGR 指令，虽然 CPU 不会进入 STOP 模式，但是会造成在一个扫描周期内 CPU 被“锁死”。为了防止出现这种情况，每 100ms 插入一个通信时间片（Time Slice）。选中图 2-36 中的“Communication load”（通信负载），可以改变这一时间片的大小（见图 2-37）。

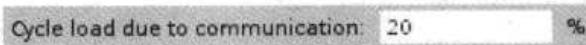


图 2-37 组态通信负载

这一机制提供了恢复 CPU 控制的机会，如果需要，可以命令 CPU 进入 STOP 模式。

通常 CPU 尽可能快地执行扫描循环。与用户程序和通信任务有关，每次扫描循环的时间间隔是变化的。为了使扫描循环时间尽可能一致，可以设置固定的扫描循环时间。为此应选

中图 2-36 中的复选框，并设置以 ms 为单位固定的最小循环时间 (Fixed minimum cycle time)。CPU 将以  $\pm 1\text{ms}$  的精度，保持以设置的最小扫描时间完成每次扫描循环。

如果 CPU 完成正常的扫描循环任务的时间小于设置的最小循环时间，CPU 将延迟启动新的循环，用附加的时间来进行运行时间诊断和处理通信请求，用这种方法来保证以固定的时间来完成扫描循环。

如果在设置的最小循环时间内，CPU 没有完成扫描循环，CPU 将完成正常的扫描（包括通信处理），并且不会产生超出最小循环时间的系统响应。

最大扫描循环时间总是起作用的，固定的最小循环时间是可选的，作为默认的设置，它被禁止。表 2-6 给出了循环时间监视功能的时间范围和默认值。

表 2-6 扫描循环时间

循环时间	范围	默认值
最大扫描循环时间/ms	1~6000	150
固定的最小扫描循环时间	1~最大扫描循环时间	禁止

# 第3章 S7-1200 程序设计基础

## 3.1 S7-1200 的编程语言及国际标准

### 3.1.1 PLC 编程语言的国际标准

IEC（国际电工委员会）是为电子技术的所有领域制定全球标准的国际组织。IEC 61131 是 PLC 的国际标准，它由以下 5 部分组成：通用信息、设备与测试要求、编程语言、用户指南和通信。其中的第三部分（IEC 61131-3）是 PLC 的编程语言标准。IEC 61131-3 是世界上第一个，也是至今为止唯一的工业控制系统的编程语言标准。

目前已有越来越多的生产 PLC 的厂家提供符合 IEC 61131-3 标准的产品，IEC 61131-3 已经成为 DCS（集散控制系统）、IPC（工业控制计算机）、FCS（现场总线控制系统）、SCADA（数据采集与监视控制）和运动控制系统事实上的软件标准。有的厂家推出的在个人计算机上运行的“软件 PLC”软件包也是按 IEC 61131-3 标准设计的。

IEC 61131-3 详细地说明了句法、语义和下述 5 种编程语言（见图 3-1）：

- 1) 指令表 IL(Instruction List)。
- 2) 结构文本 ST(Structured Text)。
- 3) 梯形图 LD(Ladder Diagram): 西门子公司简称为 LAD。
- 4) 功能块图 FBD (Function Block Diagram)。
- 5) 顺序功能图 SFC(Sequential Function Chart)。

### 3.1.2 S7-1200 的编程语言

STEP 7 Basic 是 S7-1200 系列 PLC 的编程软件，S7-1200 只有梯形图和功能块图这两种编程语言。

#### 1. 梯形图

梯形图（LAD）是使用得最多的 PLC 图形编程语言。梯形图与继电器电路图很相似，具有直观易懂的优点，很容易被工厂熟悉继电器控制的电气人员掌握，特别适合于数字量逻辑控制。有时把梯形图称为电路或程序。

梯形图由触点、线圈和用方框表示的指令框组成。触点代表逻辑输入条件，例如外部的开关、按钮和内部条件等。线圈通常代表逻辑运算的结果，常用来控制外部的负载和内部的标志位等。指令框用来表示定时器、计数器或者数学运算等指令。

使用编程软件可以直接生成和编辑梯形图，并将它下载到 PLC。

触点和线圈等组成的独立电路称为网络（Network），STEP 7 Basic 自动地为网络编号。

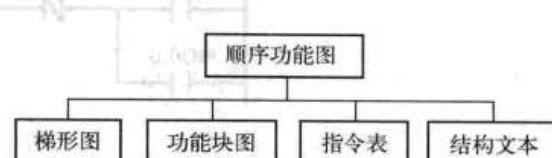


图 3-1 PLC 的编程语言

可以在网络编号的右边加上网络的标题，在网络编号的下面为网络加上注释（见图 3-2）。点击工具栏上的 按钮，可以显示或关闭网络的注释。

本书为了节约篇幅，插图中的梯形图一般没有标出网络编号。

在分析梯形图的逻辑关系时，为了借用继电器电路图的分析方法，可以想象在梯形图的左右两侧垂直“电源线”之间有一个左正右负的直流电源电压，当图 3-2 中 I0.0 与 I0.1 的触点同时接通，或 Q0.0 与 I0.1 的触点同时接通时，有一个假想的“能流”（Power Flow）流过 Q0.0 的线圈。利用能流这一概念，可以借用继电器电路的术语和分析方法，帮助我们更好地理解和分析梯形图。能流只能从左往右流动。

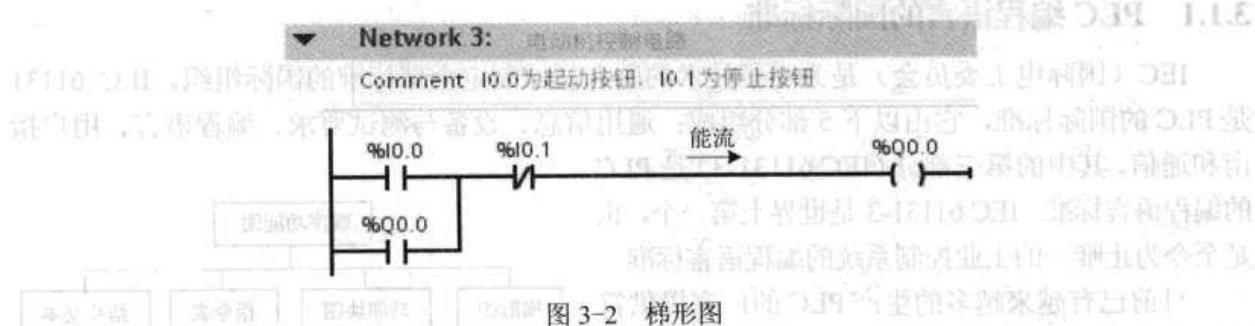


图 3-2 梯形图

网络内的逻辑运算按从左往右的方向执行，与能流的方向一致。如果没有跳转指令，网络之间按从上到下的顺序执行，执行完所有的网络后，下一次扫描循环返回最上面的网络 1，重新开始执行。



图 3-3 功能块图

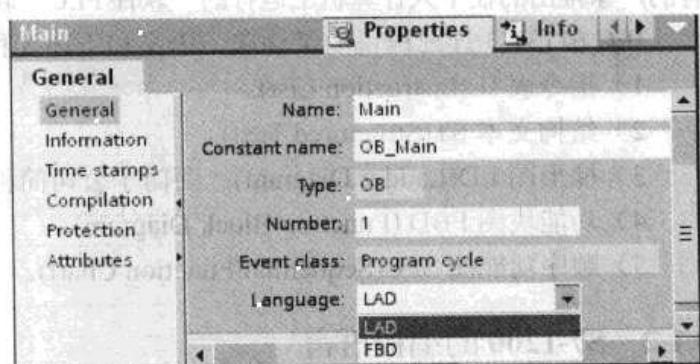


图 3-4 设置块的编程语言

## 2. 功能块图

功能块图（FBD）使用类似于数字电路的图形逻辑符号来表示控制逻辑，有数字电路基础的人很容易掌握。国内很少有人使用功能块图语言。

在功能块图中，用类似于与门（带有符号“&”）、或门（带有符号“ $\geq 1$ ”）的方框来表示逻辑运算关系，方框的左边为输入变量，右边为逻辑运算的输出变量，输入、输出端的小圆圈表示“非”运算，方框被“导线”连接在一起，信号自左向右流动。指令框用来表示一些复杂的功能，例如数学运算等。图 3-3 是图 3-2 中的梯形图对应的功能块图，图 3-3 同时显示绝对地址和符号地址。

## 3. 编程语言的切换

打开项目树中 PLC 的“Program block”（程序块）文件夹，双击其中的某个代码块，打

开程序编辑器，在工作区下面的监视窗口的“Properties”选项卡中（见图 3-4），可以用“Language”下拉式列表改变块使用的编程语言。

## 3.2 数据类型与系统存储区

### 3.2.1 物理存储器

PLC 的操作系统使 PLC 具有基本的智能，能够完成 PLC 设计者规定的各种工作。用户程序由用户设计，它使 PLC 能完成用户要求的特定功能。

#### 1. PLC 使用的物理存储器

##### (1) 随机存取存储器

CPU 可以读出随机存取存储器 (RAM) 中的数据，也可以将数据写入 RAM。它是易失性的存储器，电源中断后，存储的信息将会丢失。

RAM 的工作速度高，价格便宜，改写方便。在关断 PLC 的外部电源后，可以用锂电池保存 RAM 中的用户程序和某些数据。

##### (2) 只读存储器

只读存储器 (ROM) 的内容只能读出，不能写入。它是非易失的，电源消失后，仍能保存存储的内容，ROM 一般用来存放 PLC 的操作系统。

##### (3) 快闪存储器和可电擦除可编程只读存储器

快闪存储器 (Flash EPROM) 简称为 FEPROM，可电擦除可编程的只读存储器简称为 EEPROM。它们是非易失性的，可以用编程装置对它们编程，兼有 ROM 的非易失性和 RAM 的随机存取优点，但是将信息写入它们所需的时间比 RAM 长得多。它们用来存放用户程序和断电时需要保存的重要数据。

#### 2. 微存储卡

SIMATIC 微存储卡基于 FEPROM，用于在断电时保存用户程序和某些数据。微存储卡用来作装载存储器 (Load Memory) 或作便携式媒体。

#### 3. 装载存储器与工作存储器

##### (1) 装载存储器

装载存储器是非易失性的存储器，用于保存用户程序、数据和组态信息。所有的 CPU 都有内部的装载存储器，CPU 插入存储卡后，用存储卡作装载存储器。项目下载到 CPU 时，保留在装载存储器。

##### (2) 工作存储器

工作存储器是集成在 CPU 中的高速存取的 RAM，为了提高运行速度，CPU 将用户程序中与程序执行有关的部分，例如组织块、功能块、功能和数据块从装载存储器复制到工作存储器。

装载存储器类似于计算机的硬盘，工作存储器类似于计算机的内存条。CPU 断电时，工作存储器中的内容将会丢失。

#### 4. 断电保持存储器

断电保持存储器 (Retentive Memory) 用来防止在电源关闭时丢失数据，暖启动后断电保

持存储区中的数据保持不变。冷启动时断电保持存储器的值被清除。

CPU 提供了 2048B 的保持存储器，可以指定在断电时，将工作存储器的数据（例如数据块或位存储器 M）的值永久保存在保持存储器中。断电时 CPU 有足够的时间来保存数量有限的指定的存储单元的值。

断电时选择的工作存储器的值被复制到保持存储器，首先进行求和运算，保存校验和之后，被保持的值写入非易失存储器。校验和与 CPU 的操作系统需要保持的其他值不会占用给用户使用的 2048B 保持存储器。

电源恢复后，系统将保持存储器保存的断电之前工作存储器的数据，恢复到原来的存储单元。需要保存的数据如果超过 2048B，将被拒绝。

在暖启动时，所有非保持的位存储器被删除，非保持的数据块的内容被复位为装载存储器中的初始值。保持存储器和有保持功能的数据块的内容被保持。

可以用下列方法设置变量的断电保持属性：

1) 位存储器中的变量：可以在分配表（见 6.4.2 节）或 PLC 变量表（见 3.3.3 节）中，定义从 MB0 开始的有断电保持功能的位存储器的地址范围。

2) FB 的局部变量：如果激活了“Symbolic access only”（仅符号访问）属性，可以在 FB 的接口区定义单个变量是否有保持功能。如果没有激活 FB 的该属性，只能在指定的背景数据块中定义所有的变量是否有断电保持属性。

3) 全局数据块中的变量：如果激活了“Symbolic access only”属性，则可以对每个变量单独设置断电保持属性。如果禁止了 DB 的该属性，则只能设置 DB 中所有的变量是否有断电保持属性。

在线时用 CPU 操作面板上的“MRES”按钮复位存储器（见图 7-12）。只能在 STOP 模式复位存储器。存储器复位使 CPU 进入所谓的“初始状态”，清除所有的工作存储器，包括保持和非保持的存储区，将装载存储器的内容复制给工作存储器，数据块的当前值被初始值替代。编程设备与 CPU 的在线连接被中断，诊断缓冲区、时间、IP 地址、硬件组态和激活的强制任务保持不变。

如果在 CPU 断电时更换了存储卡，CPU 上电时将复位存储器。

## 5. 查看存储器的使用情况

用鼠标右键点击项目树中的某个 PLC，执行出现的快捷菜单中的“Resources”（资源）命令，可以查看当前项目的存储器使用情况。

双击项目树中某个 PLC 文件夹内的“Online and diagnostics”，打开工作区左边窗口的“Diagnostics”（诊断）文件夹，选中“Memory”（存储器），可以查看 PLC 运行时存储器的使用情况。

### 3.2.2 数制与数据类型

#### 1. 数制

##### (1) 二进制数

二进制数的 1 位 (bit) 只能取 0 和 1 这两个不同的值，可以用来表示开关量（或称数字量）的两种不同的状态，例如触点的断开和接通、线圈的通电和断电等。如果该位为 1，则表示梯形图中对应的位编程元件（例如位存储器 M 和过程映像输出位 Q）的线圈“通电”，其

常开触点接通，常闭触点断开，以后称该编程元件为 1 状态，或称该编程元件 ON（接通）。如果该位为 0，则对应的编程元件的线圈和触点的状态与上述的相反，称该编程元件为 0 状态，或称该编程元件 OFF（断开）。在编程软件中，位编程元件的 1 状态和 0 状态用 TRUE 和 FALSE 来表示。

### （2）多位二进制数

计算机和 PLC 用多位二进制数来表示数字，二进制数遵循逢二进一的运算规则，从右往左的第  $n$  位（最低位为第 0 位）的权值为  $2^n$ 。二进制常数以 2#开始，用下式计算 2#1100 对应的十进制数：

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 8 + 4 = 12$$

表 3-1 给出了不同进制数的表示方法，BCD 码将在 5.1.2 节介绍。

表 3-1 不同进制的数的表示方法

十进制数	十六进制数	二进制数	BCD 码	十进制数	十六进制数	二进制数	BCD 码
0	0	00000	0000 0000	9	9	01001	0000 1001
1	1	00001	0000 0001	10	A	01010	0001 0000
2	2	00010	0000 0010	11	B	01011	0001 0001
3	3	00011	0000 0011	12	C	01100	0001 0010
4	4	00100	0000 0100	13	D	01101	0001 0011
5	5	00101	0000 0101	14	E	01110	0001 0100
6	6	00110	0000 0110	15	F	01111	0001 0101
7	7	00111	0000 0111	16	10	10000	0001 0110
8	8	01000	0000 1000	17	11	10001	0001 0111

### （3）十六进制数

多位二进制数的书写和阅读很不方便。为了解决这一问题，可以用十六进制数来取代二进制数，每个十六进制数对应于 4 位二进制数。十六进制数的 16 个数字是 0~9 和 A~F（对应于十进制数 10~15）。B#16#、W#16#和 DW#16#分别用来表示十六进制字节、字和双字常数，例如 W#16#13AF。在数字后面加“H”也可以表示十六进制数，例如 16#13AF 可以表示为 13AFH。

## 2. 数据类型

数据类型用来描述数据的长度（即二进制的位数）和属性。本节介绍基本数据类型，其他数据类型主要在 6.2.2 节介绍。

很多指令和代码块的参数支持多种数据类型。将鼠标的光标放在用户程序中某条指令的某个未输入地址或常数的参数上，过一会儿在出现的黄色背景的小方框中，可以看到该参数支持的数据类型。

### 3. 数据对象的长度

不同的任务使用不同长度的数据对象，例如位指令使用位数据，传送指令使用字节、字和双字。字节、字和双字分别由 8 位、16 位和 32 位二进制数组成。

### 4. 基本数据类型

表 3-2 给出了基本数据类型的属性。

表 3-2 基本数据类型

变量类型	符号	位数	取值范围	常数举例
位	Bool	1	1、0	TRUE、FALSE 或 1、0
字节	Byte	8	16#00~16#FF	16#12, 16#AB
字	Word	16	16#0000~16#FFFF	16#ABCD, 16#0001
双字	DWord	32	16#00000000~16#FFFFFF	16#02468ACE
字符	Char	8	16#00~16#FF	'A', 't', '@'
有符号字节	SInt	8	-128~127	123, -123
整数	Int	16	-32768~32767	123, -123
双整数	DInt	32	-2147483648~2147483647	123, -123
无符号字节	USInt	8	0~255	123
无符号整数	UInt	16	0~65535	123
无符号双整数	UDInt	32	0~4294967295	123
浮点数(实数)	Real	32	$\pm 1.175495 \times 10^{-38} \sim \pm 3.402823 \times 10^{38}$	12.45, -3.4, -1.2E+12, 3.4E-3
双精度浮点数	LReal	64	$\pm 2.2250738585072020 \times 10^{-308} \sim \pm 1.7976931348623157 \times 10^{308}$	12345.123456789, -1.2E+40
时间	Time	32	T#-24d20h31m23s648ms~T#+24d20h31m23s647ms	T#1d_2h_15m_30s_45ms

数据类型的符号有下列特点：

- 1) 字节、字和双字均为十六进制数，字符又称为 ASCII 码。
- 2) 包含 Int 无 U 的数据类型为有符号整数，包含 Int 和 U 的数据类型为无符号整数。
- 3) 包含 SInt 的数据类型为 8 位整数，包含 Int 且无 D 和 S 的数据类型为 16 位整数，包含 DInt 的数据类型为 32 位双整数。

S7-1200 的新数据类型有下列优点：

- 1) 使用短整型数据类型，可以节约内存资源。
- 2) 无符号数据类型可以扩大正数的数值范围。
- 3) 64 位双精度浮点数可用于高精度的数学函数运算。

## 5. 位

位数据的数据类型为 Bool (布尔) 型，在编程软件中，Bool 变量的值 1 和 0 用英语单词 TRUE (真) 和 FALSE (假) 来表示。

位存储单元的地址由字节地址和位地址组成，例如 I3.2 中的区域标识符 “I” 表示输入 (Input)，字节地址为 3，位地址为 2 (见图 3-5)。这种存取方式称为“字节.位”寻址方式。

## 6. 字节

8 位二进制数组成 1 个字节 (Byte，见图 3-5)，例如 I3.0~I3.7 组成了输入字节 IB3 (B 是 Byte 的缩写)。数据类型 Byte 是十六进制数，Char 为单个 ASCII 字符，SInt 为有符号字节，USInt 为无符号字节。

## 7. 字

相邻的两个字节组成一个字，例如字 MW100 由字节 MB100 和 MB101 组成 (见图 3-6b)。MW100 中的 M 为区域标识符，W 表示字。需要注意以下两点：

- 1) 用组成字的编号最小的字节 MB100 的编号作为字 MW100 的编号。

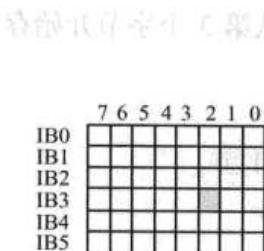


图 3-5 字节与位



图 3-6 字节、字和双字

2) 组成字的编号最小的字节 MB100 为字的高位字节, 编号最大的字节 MB101 为字的低位字节。双字也有类似的特点。

数据类型 Word 是十六进制的字, Int 为有符号的字(整数), UInt 为无符号的字。

整数和双整数的最高位为符号位, 最高位为 0 时为正数, 为 1 时为负数。整数用补码来表示, 正数的补码就是它的本身, 将一个正数对应的二进制数的各位取反后加 1, 得到绝对值与它相同的负数的补码。

## 8. 双字

两个字(或 4 个字节)组成 1 个双字, 双字 MD100 由字节 MB100~MB103 组成(见图 3-6c), D 表示双字, 100 为组成双字的起始字节 MB100 的编号。MB100 是双字 MD100 中的最高位字节。

数据类型 DWord 为十六进制的双字, DInt 为有符号双字(双整数), UDInt 为无符号双字。

## 9. 浮点数

32 位的浮点数又称为实数(Real), 最高位(第 31 位)为浮点数的符号位(见图 3-7), 正数时为 0, 负数时为 1。规定尾数的整数部分总是为 1, 第 0~22 位为尾数的小数部分。8 位指数加上偏移量 127 后(1~255), 占第 23~30 位。

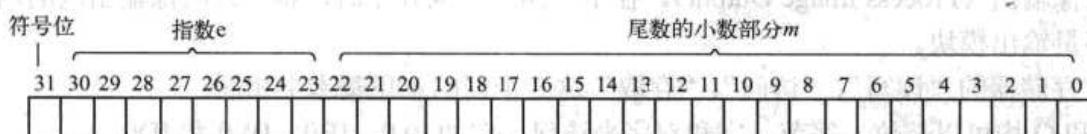


图 3-7 浮点数的结构

浮点数的优点是用很小的存储空间(4B)可以表示非常大和非常小的数。PLC 输入和输出的数值大多是整数, 例如模拟量输入值和模拟量输出值, 用浮点数来处理这些数据需要进行整数和浮点数之间的相互转换, 浮点数的运算速度比整数的运算速度慢一些。

在编程软件中, 用十进制小数来输入或显示浮点数, 例如 50 是整数, 而 50.0 为浮点数。

LReal 为 64 位的双精度浮点数, 它只能在设置了仅使用符号寻址的块中使用。LReal 的最高位(第 63 位)为浮点数的符号位, 11 位指数占第 52~62 位。尾数的整数部分总是为 1, 第 0~51 位为尾数的小数部分。

## 10. 复杂数据类型简介

1) 数组(ARRAY)由相同数据类型的元素组合而成, 5.1.4 节介绍了在数据块中生成数组的方法。

2) 字符串(String)是由字符组成的一维数组, 每个字节存放 1 个字符。第 1 个字节是字

字符串的最大字符长度，第 2 个字节是字符串当前有效字符的个数，字符从第 3 个字节开始存放，一个字符串最多有 254 个字符。

用单引号表示字符串常数，例如'ABC'是有 3 个字符的字符串常数。

3) DTL 用来表示日期时间值，它由 12B 组成，其详细的结构见 5.6.1 节。

4) 结构 (STRUCT) 可以由不同数据类型的元素组成 (见 6.2.1 节)。

### 3.2.3 系统存储区

#### 1. 过程映像输入/输出

在每次扫描循环开始时，CPU 读取数字量输入模块的外部输入电路的状态，并将它们存入过程映像输入 (Process Image Input，见表 3-3)。

表 3-3 系统存储区

存储区	描述	强制	保持
过程映像输入 (I)	在扫描循环开始时，从物理输入复制的输入值	Yes	No
物理输入 (I:P)	通过该区域立即读取物理输入	No	No
过程映像输出 (Q)	在扫描循环开始时，将输出值写入物理输出	Yes	No
物理输出 (Q:P)	通过该区域立即写物理输出	No	No
位存储器 (M)	用于存储用户程序的中间运算结果或标志位	No	Yes
临时局部存储器 (L)	块的临时局部数据，只能供块内部使用	No	No
数据块 (DB)	数据存储器与 FB 的参数存储器	No	Yes

用户程序访问 PLC 的输入 (I) 和输出 (Q) 地址区时，不是去读、写数字量模块中信号的状态，而是访问 CPU 的过程映像区。在扫描循环中，用户程序计算输出值，并将它们存入过程映像输出 (Process Image Output)。在下一循环扫描开始时，将过程映像输出区的内容写到数字量输出模块。

对存储器的“读写”、“访问”、“存取”这 3 个词的意思基本上相同。

I 和 Q 均可以按位、字节、字和双字来访问，例如 I0.0、IB0、IW0 和 ID0。

与直接访问输入模块相比，访问过程映像输入可以保证在整个扫描循环周期内，过程映像输入的状态的一致性。即使在本次循环的程序执行过程中，接在输入模块的外部电路的状态发生了变化，过程映像输入的状态仍然保持不变，直到下一个循环被刷新。由于过程映像保存在 CPU 的系统存储器中，访问速度比直接访问信号模块快得多。

过程映像输入在用户程序中的标识符为 I，它是 PLC 接收外部输入的数字量信号的窗口。输入端可以外接常开触点或常闭触点，也可以接多个触点组成的串并联电路。循环周期开始时，PLC 将外部电路的通/断状态读入并存储在过程映像输入位中，外部输入电路接通时，对应的过程映像输入位为 ON (1 状态)，反之为 OFF (0 状态)。在梯形图中，可以多次使用过程映像输入位的常开触点和常闭触点。

过程映像输出位在用户程序中的标识符为 Q，循环周期开始时，CPU 将过程映像输出位的数据传送给输出模块，再由后者驱动外部负载。如果梯形图中 Q0.0 的线圈“通电”，继电器型输出模块对应的硬件继电器的常开触点闭合，使接在 Q0.0 对应的输出端子的外部负载通电工作。输出模块的每一个硬件继电器仅有一对常开触点，但是在梯形图中，可以多次使用

同一个输出位的常开触点和常闭触点。

## 2. 物理输入

在 I/O 点的地址或符号地址的后边附加 “:P”，可以立即访问物理输入或物理输出。通过给输入点的地址附加 “:P”，例如 I0.3:P 或 “Stop:P”，可以立即读取 CPU、信号板和信号模块的数字量输入和模拟量输入。访问时使用 I:\_P 取代 I 的区别在于前者的数字直接来自被访问的输入点，而不是来自过程映像输入。因为数据从信号源被立即读取，而不是从最后一次被刷新的过程映像输入中复制，这种访问被称为“立即读”访问。

由于物理输入点从直接连接在该点的现场设备接收数据值，因此写物理输入点是被禁止的，即 I:\_P 访问是只读的。

I:\_P 访问还受到硬件支持的输入长度的限制。以被组态为从 I4.0 开始的 2 DI / 2 DQ 信号板的输入点为例，可以访问 I4.0:P、I4.1:P 或 IB4:P，但是不能访问 I4.2:P~I4.7:P，因为没有使用这些输入点。也不能访问 IW4:P 和 ID4:P，因为它们超过了信号板使用的字节范围。

用 I:\_P 访问物理输入不会影响存储在过程映像输入 (I) 中的对应值。

## 3. 物理输出

在输出点的地址后边附加 “:P”（例如 Q0.3:P），可以立即写 CPU、信号板和信号模块的数字量和模拟量输出。访问时使用 Q:\_P 取代 Q 的区别在于前者的数字直接写给被访问的物理输出点，同时写给过程映像输出。这种访问被称为“立即写”，因为数据被立即写给目标点，不用等到下一次刷新时将过程映像输出中的数据传送给目标点。

由于物理输入点直接控制与该点连接的现场设备，因此读物理输入点是被禁止的，即 Q:\_P 访问是只写的。与此相反，可以读写 Q 区的数据。

Q:\_P 访问还受到硬件支持的输出长度的限制。以被组态为从 Q4.0 开始的 2 DI / 2 DQ 信号板的输出点为例，可以访问 Q4.0:P、Q4.1:P 或 QB4:P，但是不能访问 Q4.2:P~Q4.7:P，因为没有使用这些输出点。也不能访问 QW4:P 和 QD4:P，因为它们超过了信号板使用的字节范围。

用 Q:\_P 访问物理输出同时影响物理输出点和存储在过程映像输出 (Q) 中的对应值。

## 4. 位存储器区

位存储器区 (M 存储器) 用来存储运算的中间操作状态或其他控制信息。可以用位、字节、字或双字读/写位存储器区。

## 5. 数据块

数据块 (Data Block) 简称为 DB，用来存储代码块使用的各种类型的数据，包括中间操作状态、其他控制信息，以及某些指令（例如定时器、计数器指令）需要的数据结构。可以设置数据块有写保护功能。

数据块关闭后，或有关的代码块的执行开始或结束后，数据块中存放的数据不会丢失。有两种类型的数据块：

- 1) 全局 (Global) 数据块：存储的数据可以被所有的代码块访问（见图 3-8）。
- 2) 背景 (Instance) 数据块：存储的数据供指定的功能块 (FB) 使用，其结构取决于 FB 的接口区的参数。

## 6. 临时存储器

临时存储器 (Temporary Memory) 用于存储代码块被处理时使用的临时数据。

PLC 为 3 个 OB 的优先级组（见表 6-3）分别提供临时存储器：

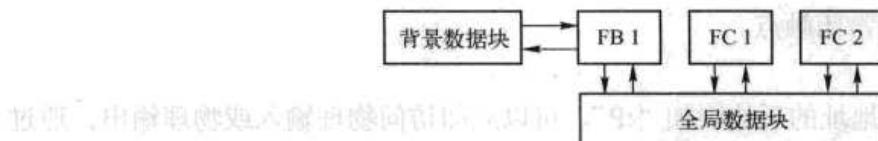


图 3-8 全局数据块与背景数据块

- 1) 启动和程序循环（包括有关的 FB 和 FC）16KB。
- 2) 标准的中断事件（包括有关的 FB 和 FC）4KB。
- 3) 错误中断事件（包括有关的 FB 和 FC）4KB。

临时存储器类似于 M 存储器，二者的主要区别在于 M 存储器是全局的，而临时存储器是局部的：

1) 所有的 OB、FC 和 FB 都可以访问 M 存储器中的数据，即这些数据可以供用户程序中所有的代码块全局性地使用。

2) 在 OB、FC 和 FB 的接口 (Interface) 区生成临时变量 (Temp)。它们具有局部性，只能在生成它们的代码块内使用，不能与其他代码块共享。即使 OB 调用 FC，FC 也不能访问调用它的 OB 的临时存储器。

CPU 按照按需访问的策略分配临时存储器。CPU 在代码块被启动（对于 OB）或被调用（对于 FC 和 FB）时，将临时存储器分配给代码块。

代码块执行结束后，CPU 将它使用的临时存储器区重新分配给其他要执行的代码块使用。CPU 不对在分配时可能包含数值的临时存储单元初始化。只能通过符号地址访问临时存储器。

### 3.3 用 STEP 7 Basic 生成用户程序

#### 3.3.1 设置 STEP 7 Basic 的参数

在项目编辑器中执行菜单命令“Options”→“Settings”，选中工作区左边窗口的“General”，在工作区的右边窗口（见图 3-9），用户接口语言只能选英语，助记符 (Mnemonic) 应选英语的助记符 (International)。

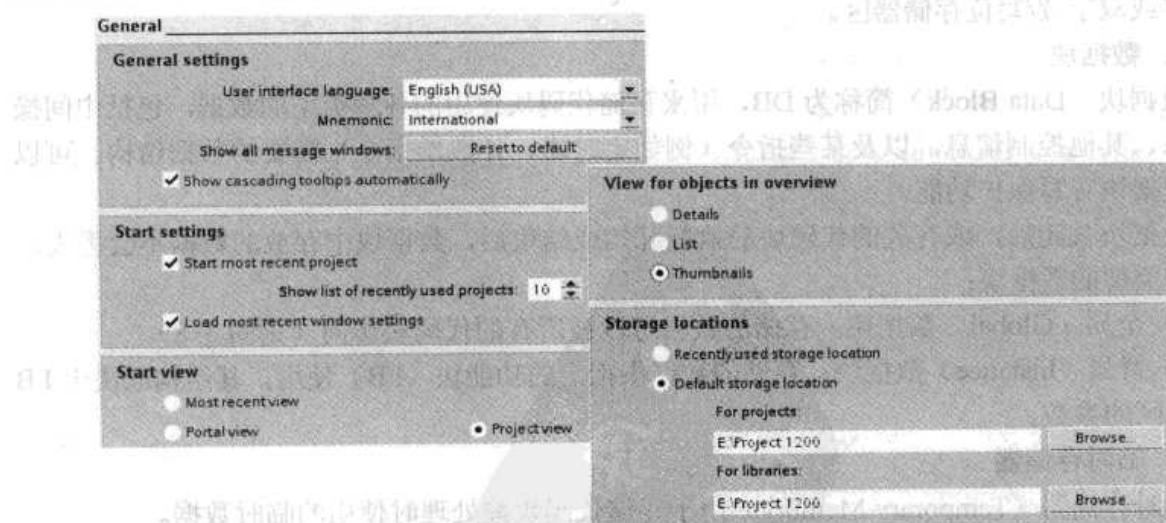


图 3-9 设置 STEP 7 Basic 的参数

在启动设置(Start settings)区，可以用复选框选择启动最近打开的项目(Start most recent project)，设置显示最近使用的项目列表的项目个数>Show list of recently used projects)，默认值为10个。用复选框可以设置是否装载最后的窗口设置(Load most recent window settings)。

在“Start view”区，用单选框选择启动后显示上一次关闭软件时最后显示的视图(Most recent view)、入口视图(Portal view)或项目视图(Project view)。建议选择项目视图，这样每次打开软件后显示的都是项目视图。

在项目存储位置(Storage locations)区，可以选择最近使用的存储位置(Recently used storage location)或默认的存储位置(Default storage location)，选中后者时，可以用“Browse”按钮设置保存项目(Projects)和库(Libraries)的路径。

### 3.3.2 编写用户程序

#### 1. 在项目视图中生成项目

按图3-9设置项目后，启动STEP 7 Basic，将自动打开项目视图(见图3-12)，并打开上一次关闭软件之前打开的项目。

执行菜单命令“Project”→“New”，生成一个新的项目，项目名称为“Bit\_Logic”。

#### 2. 添加新设备

双击项目树中的“Add new device”，添加一个新设备。点击打开的对话框中的“SIMATIC PLC”按钮(见图3-10)，选中右边窗口的“CPU 1214C”文件夹中的某个订货号，点击“OK”按钮，生成名为“PLC\_1”的新PLC。该设备只有CPU模块。

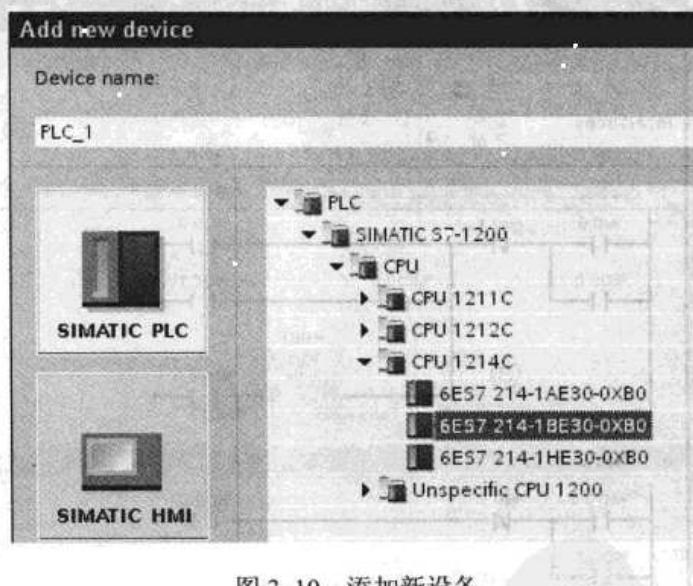


图3-10 添加新设备

### 3. 用户程序简介

图3-11是异步电动机星形-三角形降压起动的主电路和PLC的外部接线图。主电路中的接触器KM1和KM2动作时，异步电动机在星形接线方式运行；KM1和KM3动作时，在三角形接线方式运行。

停车按钮和过载保护器的常开触点并联后接在I0.1对应的输入端，可以节约一个输入点。

输入回路使用 CPU 模块内置的 DC 24V 电源, 其负极 M 点与内部输入电路的公共点 1M 连接, L+是 24V 电源的正极。

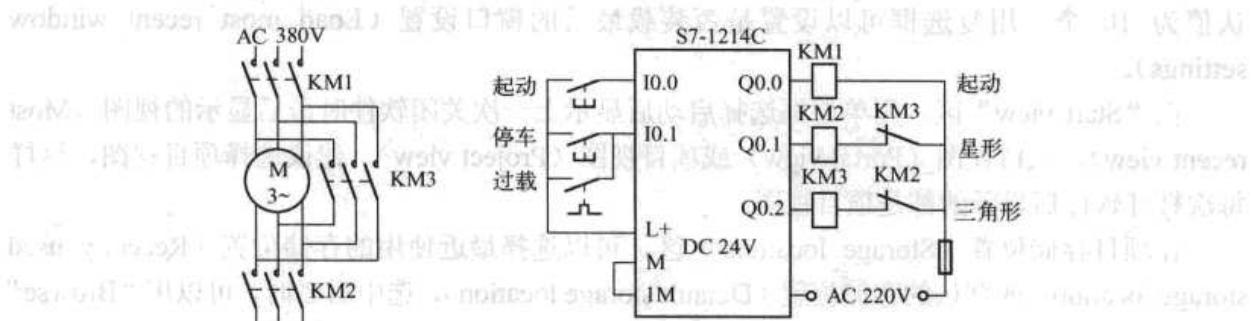


图 3-11 电动机主电路与 PLC 外部接线图

按下起动按钮 I0.0, Q0.0 和 Q0.1 同时变为 1 状态 (见图 3-12 中的梯形图), 使 KM1 和 KM2 同时动作, 电动机按星形接线方式运行, 定时器的 IN 输入端为 1。8s 后受定时器控制的 M10.0 的常闭触点断开, 通过 Q0.1 使 KM2 的线圈断电。M10.0 的常开触点闭合, 通过 Q0.2 使 KM3 的线圈通电, 电动机改为三角形接线方式运行。按下停车按钮, 梯形图中 I0.1 的常闭触点断开, 使 KM1 和 KM3 的线圈断电, 电动机停止运行。过载时 I0.1 的常开触点也会断开, 使电动机停机。

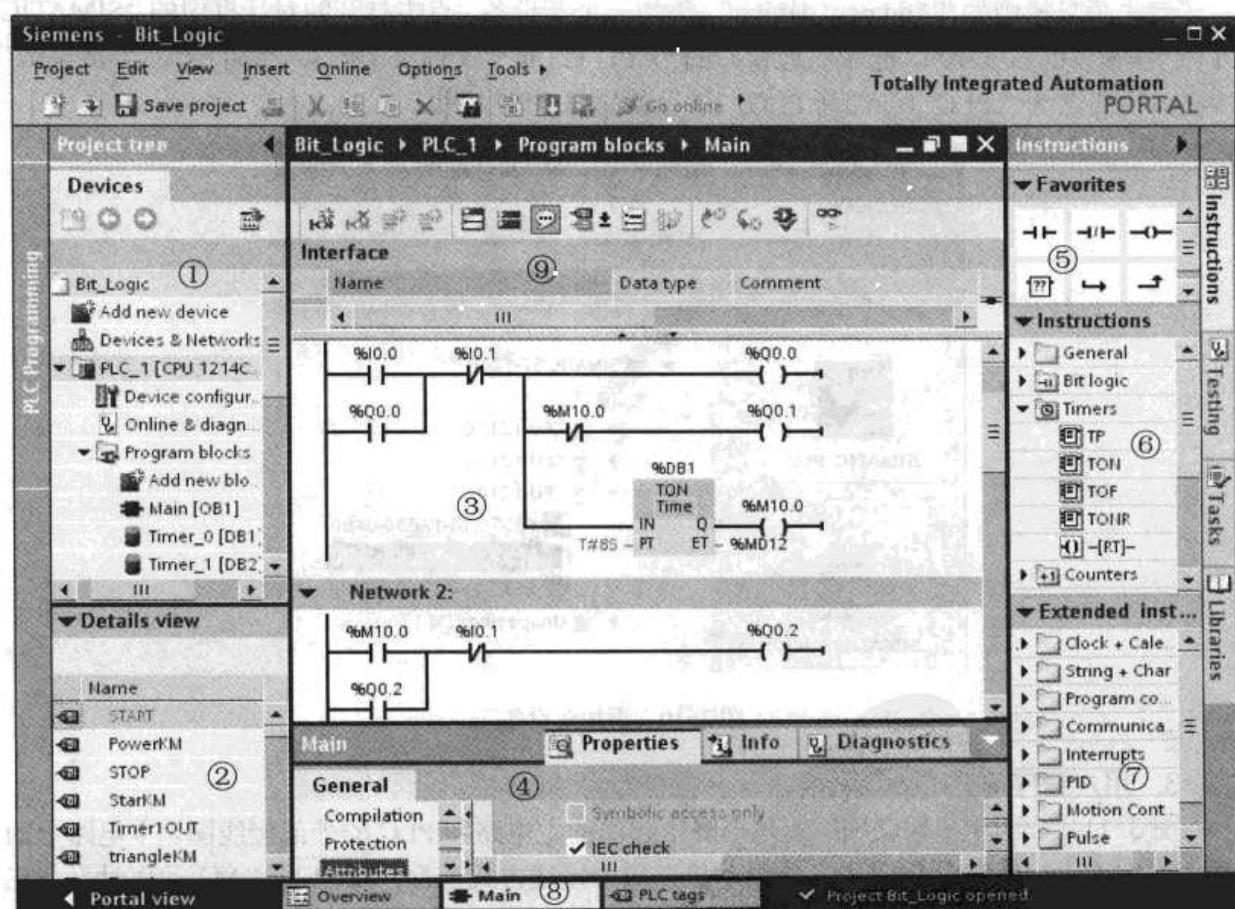


图 3-12 项目视图中的程序编辑器

#### 4. 程序编辑器简介

双击项目树的文件夹“\PLC\_1\Program blocks”中的 OB1，打开主程序（见图 3-12）。选中项目树中的 PLC tags（PLC 变量表）后，标有②的详细视图（Details view）显示变量表中的变量，可以将其中的变量直接拖放到梯形图中使用。

将鼠标的光标放在 OB1 的程序区最上面的分隔条上，按住鼠标左键，往下拉动分隔条，分隔条上面是代码块的接口（Interface）区（见图 3-12 中标有⑨的区域），下面是程序区。将水平分隔条拉至程序编辑器视窗的顶部，不再显示接口区，但是它仍然存在。

程序区的下面标有④的区域是打开的程序块的监视窗口。标有⑥、⑦的区域分别是任务卡中的指令列表和扩展的指令列表。

标有⑤的区域是指令的收藏栏（Favorites），用于快速访问常用的指令。如果关闭了指令列表，点击程序编辑器工具栏上的按钮，可以在程序编辑器中显示收藏栏。

点击最右边的垂直条上的“Testing”、“Tasks”和“Libraries”按钮，可以分别在任务卡中打开测试、任务和库窗口。

标有⑧的编辑器条中的按钮对应于已经打开的编辑器。点击编辑器条上的某个按钮，可以在工作区显示点击的按钮对应的编辑器。

#### 5. 程序编辑器工具栏上的按钮

：在选中的网络的下面插入一个新的网络。

：删除选中的网络。

、：打开、关闭所有的网络。

、：设置变量的显示方式，可选显示绝对地址、符号地址或同时显示两种地址。

：关闭或打开网络的注释。

：在编辑器中显示或隐藏收藏栏。

、：跳转到前一个或下一个语法错误。

：更新不一致的块调用。

：打开或关闭程序状态监视。

#### 6. 生成用户程序

选中 Network1（网络 1）中的水平线，依次双击标有⑤的收藏栏中的、和按钮，水平线上出现从左到右的常开触点、常闭触点和线圈，元件上面红色的按钮用来输入元件的地址。选中最左边的垂直“电源线”，依次双击按钮、和，生成一个与上面的常开触点并联的 Q0.0 的常开触点。

选中图 3-12 中 I0.1 的常闭触点之后的水平线，依次双击、和按钮，出现图中 Q0.1 线圈所在的支路。

输入触点和线圈的绝对地址后，自动生成名为“tag\_x”（x 为数字）的符号地址，可以在 PLC tags 中查看和修改它们。绝对地址前面的字符%是编程软件自动添加的。

S7-1200 使用的 IEC 定时器和计数器属于功能块（FB），在调用它们时，需要生成对应的背景数据块。

选中图 3-12 中 M10.0 的常闭触点左边的水平线，双击按钮，然后打开标有⑥的指令列表中的文件夹“Timer”（定时器），双击其中的接通延时定时器 TON 的图标，出现图 3-13 所示的对话框，点击“OK”按钮，生成指令 TON 的背景数据块。S7-1200 的定时器和计数器

没有编号，可以在图 3-13 中设置定时器的背景数据块的名称（Name），例如 T0，用它来作定时器的标识符。

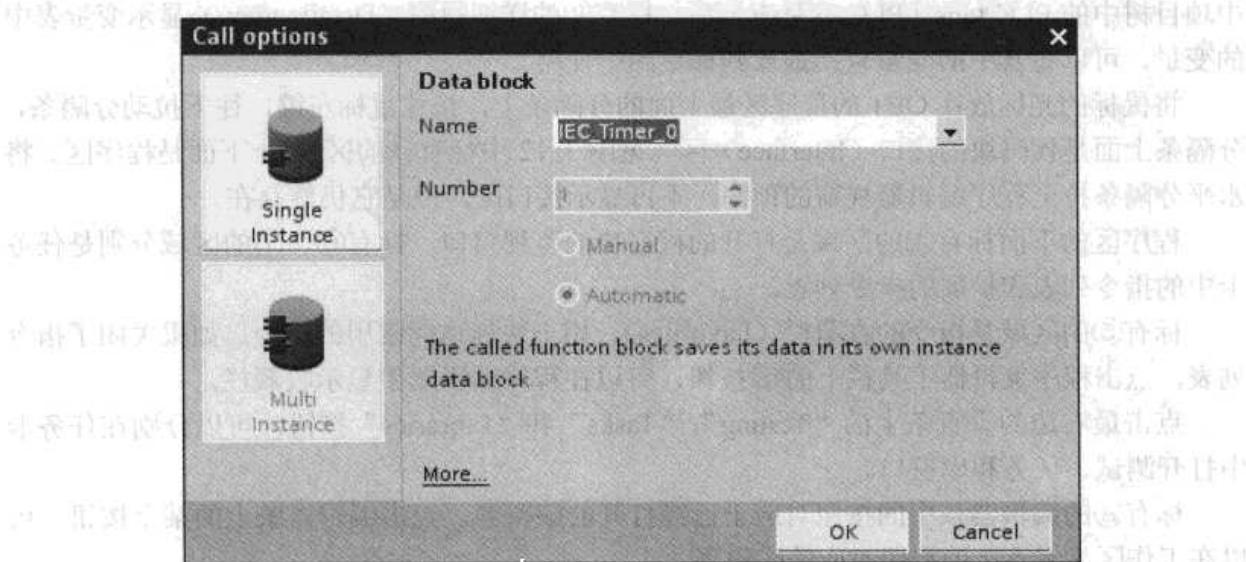


图 3-13 生成定时器的背景数据块

在 TON 的 PT 输入端输入定时器的设定值，在 TON 的 Q 输出端添加一个线圈。点击各元件上面的<???.?>，输入元件的地址。

生成定时器时，也可以将收藏栏的 图标拖放到指定的位置，点击出现的图标中的红色问号，再点击图标中出现的 按钮，输入“TON”。可以用这个方法输入任意的指令。

即使程序块没有输入完整，或者有错误，也可以保存项目。

### 3.3.3 使用 PLC 变量表

#### 1. 生成和修改变量

打开项目树的文件夹“PLC tags”（PLC 变量表），双击其中的“PLC tags”，打开变量编辑器。选项卡“PLC tags”用来定义 PLC 的变量，选项卡“Constants”中是系统自动生成的与 PLC 的硬件和中断事件有关的常数符号（见图 6-20）。

在选项卡“PLC tags”最下面的空白行的“Name”列输入变量的名称（不能使用汉字），点击“Data type”列右侧隐藏的按钮，设置变量的数据类型（只能使用基本数据类型）。在“Address”列输入变量的绝对地址。

符号地址使程序易于阅读和理解。可以首先用 PLC 变量表定义变量的符号地址，然后在用户程序中使用它们。在程序中输入绝对地址后，将会自动生成名为“tag\_x”的符号地址（x 为数字编号）。可以在变量表中修改自动生成的符号地址的名称。找到需要修改符号名称的某个变量后，将默认的名称修改为指定的名称。

图 3-14 是修改变量名称后的 PLC 变量表以及同时显示符号地址和绝对地址的网络 1 中的梯形图。

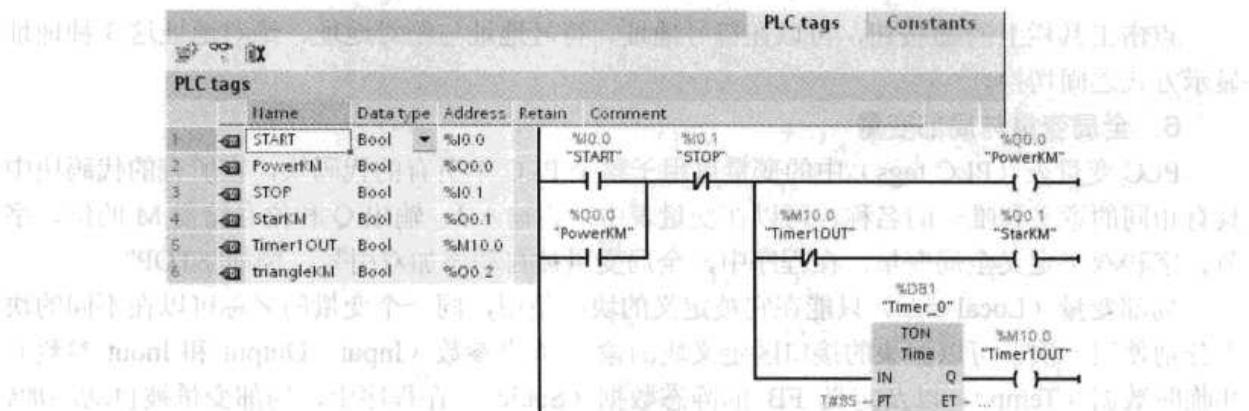


图 3-14 PLC 变量表与梯形图

## 2. 变量表中变量的排序

点击变量表表头中的“Address”，该单元出现向上的三角形，各变量按地址的第一个字母（I、Q 和 M 等）升序排列（从 A 到 Z）。再点击一次该单元，各变量按地址的第一个字母降序排列。可以用同样的方法，根据变量的名称和数据类型等来排列变量。

## 3. 快速生成变量

用鼠标右键点击变量“PowerKM”，执行出现的快捷菜单中的命令“Insert row”，在该变量上面出现一个空白行（见图 3-15 的左图）。选中变量 STOP 左边的标签 ，用鼠标按住右下角的蓝色小正方形不放，向下拖动鼠标，在空白行生成新的变量 Tag\_2（见图 3-15 的右图），它继承了上一行的变量 STOP 的数据类型和地址，其名称是自动生成的。如果选中最下面一行的变量向下拖动，可以快速生成多个同类型的变量。

	Name	Data type	Address	▲ Retain
1	START	Bool	%I0.0	
2	STOP	Bool	%I0.1	
4	PowerKM	Bool	%Q0.0	

	Name	Data type	Address	▲ Retain
1	START	Bool	%I0.0	
2	STOP	Bool	%I0.1	
3	Tag_2	Bool	%I0.2	
4	PowerKM	Bool	%Q0.0	

图 3-15 快速生成变量

## 4. 设置变量的断电保持功能

点击工具栏上的 按钮，可以用打开的对话框（见图 6-52）设置 M 区从 MB0 开始的具有断电保持功能的字节数。

## 5. 设置程序中地址的显示方式

点击工具栏上的 按钮，可以用下拉式菜单选择只显示绝对地址（Symbolic，见图 3-16）、只显示符号地址（Absolute，见图 3-17），或同时显示两种地址（Symbolic and absolute，见图 3-18）。将鼠标的光标放到地址上，图中触点下面出现的黄色小方框中是另一种格式的地址和变量的数据类型（见图 3-16～图 3-18）。

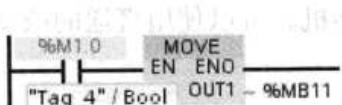


图 3-16 只显示绝对地址



图 3-17 只显示符号地址

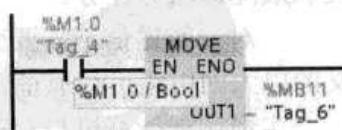


图 3-18 同时显示两种地址

点击工具栏上的 $\text{A}$ 按钮，可以在符号地址、符号地址与绝对地址、绝对地址这3种地址显示方式之间切换。

## 6. 全局变量与局部变量

PLC 变量表（PLC tags）中的变量可用于整个 PLC 中所有的代码块，在所有的代码块中具有相同的意义和唯一的名称，可以在变量表中，为输入 I、输出 Q 和位存储器 M 的位、字节、字和双字定义全局变量。在程序中，全局变量被自动添加双引号，例如"STOP"。

局部变量（Local tags）只能在它被定义的块中使用，同一个变量的名称可以在不同的块中分别使用一次。可以在块的接口区定义块的输入/输出参数（Input、Output 和 Inout 参数）和临时数据（Temp），以及定义 FB 的静态数据（Static）。在程序中，局部变量被自动添加#号，例如#Start。

## 7. 设置块的变量只能用符号访问

生成块时，如果用复选框选中了“Symbolic access only”（只能用符号访问，见图 5-16），则在全局数据块、FB 和 FC 的接口区声明的变量只有符号名，在块内没有固定的地址。在编译时变量的绝对地址被动态地传送，并且不会在全局数据块内或在 FB、FC 的接口区显示出来。变量以优化的方式保存，可以提高存储区的利用率。只能用符号地址的方式访问声明的变量。例如用>Data.Level2 访问数据块 Data 中的变量 Level2。

执行“Options”菜单中的“Settings”命令，选中出现的“Settings”对话框左边窗口的“PLC Programming”组（见图 3-19），用复选框选中右边窗口中的“Symbolic access only”，所有新生成的块默认的设置为其变量仅使用符号访问。

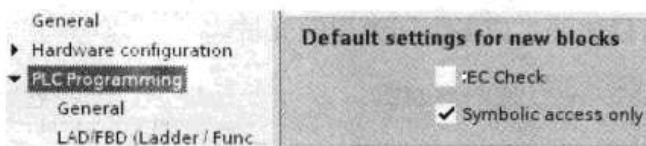


图 3-19 设置新的块的默认属性

## 8. 使用详细窗口

打开项目树下面的详细窗口（见图 3-12 中标有②的区域），选中项目树中的“PLC tags”，详细窗口显示出变量表中的符号。可以将详细窗口中的符号地址或代码块的接口区中定义的局部变量拖放到程序中需要设置地址的 $<??>$ 处。拖放到已设置的地址上时，原来的地址将会被替换。

## 3.4 下载用户程序

通过 CPU 与运行 STEP 7 Basic 的计算机的以太网通信，可以执行项目的下载、上载、监控和故障诊断等任务。

一对一的通信不需要交换机，两台以上的设备通信则需要交换机。可以使用直通的或交叉的以太网电缆进行通信。

实现编程计算机与 CPU 通信的步骤如下：

- 1) 设置计算机网卡的 IP 地址和子网掩码。

- 2) 设置 CPU 的以太网接口的 IP 地址和子网掩码。
- 3) 用以太网电缆连接计算机和 CPU 的以太网接口。

### 1. 设置计算机网卡的 IP 地址

IP 地址与子网掩码的定义见 7.3.1 节。计算机的网卡与 CPU 的以太网接口的 IP 地址应在同一个子网内，即它们的 IP 地址中前 3 个字节的子网地址应完全相同。此外它们还应使用相同的子网掩码。子网的地址一般采用默认的 192.168.0，第 4 个字节是子网内设备的地址，可以取任意值，但是不能与网络中其他设备的 IP 地址重叠。计算机和 CPU 的子网掩码一般采用默认的 255.255.255.0。

用鼠标右键点击计算机桌面上的“网络邻居”，执行出现的快捷菜单中的“属性”命令，打开“网络连接”对话框（见图 3-20），用鼠标右键点击“本地连接”图标，执行出现的快捷菜单中的“属性”命令，打开“本地连接 属性”对话框。选中“此连接使用下列项目”列表框中的“Internet 协议 (TCP/IP)”，点击“属性”按钮，打开“Internet 协议 (TCP/IP) 属性”对话框。用单选框选中“使用下面的 IP 地址”，然后按上述的原则设置网卡的 IP 地址，第 4 个字节只要不与网络中其他设备的 IP 地址相同就可以了。点击子网掩码框，将会自动生成默认的子网掩码 255.255.255.0。设置结束后，点击各级对话框中的“确定”按钮，最后关闭“网络连接”对话框。

使用宽带上互联网时，一般只需要用单选框选中图 3-20 中的“自动获得 IP 地址”。

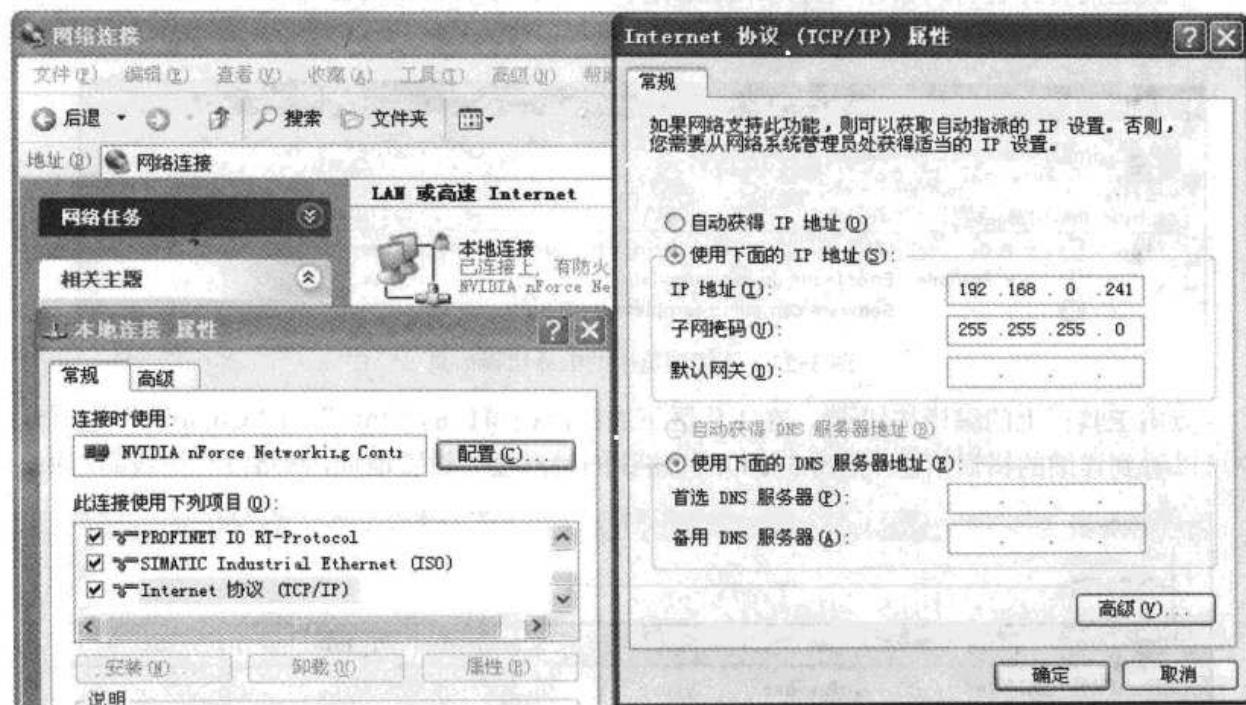


图 3-20 设置计算机网卡的 IP 地址

### 2. 组态 CPU 的 PROFINET 接口

双击项目树中 PLC 文件夹内的“Device configuration”，打开该 PLC 的设备视图。选中 CPU 后，再选中下面的监视窗口左边的“Ethernet addresses”组，可以采用右边窗口默认的 IP 地址和子网掩码（见图 3-21）。设置的地址在下载后才起作用。

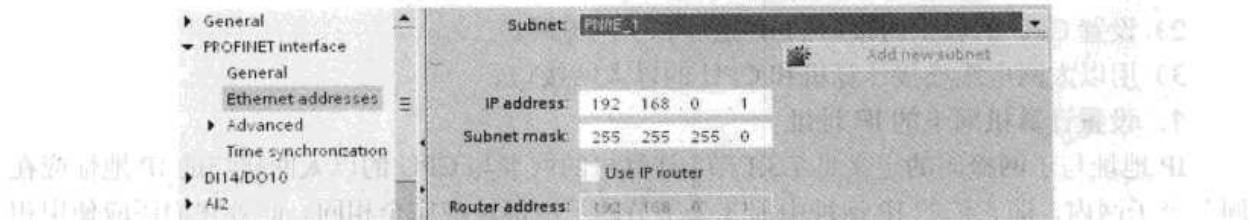


图 3-21 设置 CPU 集成的以太网接口的 IP 地址

### 3. 下载项目

选中项目树的 CPU 设备后, 点击工具栏上的下载按钮 , 如果 PLC 处于 RUN 模式, 将会出现图 3-22 中的对话框。询问是否将选中的 CPU 切换到 STOP 模式, 点击 “OK” 按钮确认。

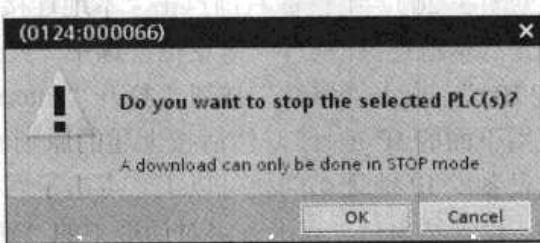


图 3-22 询问是否切换到 STOP 模式的对话框

下载之前首先会自动编译项目, 如果编译时发现错误, 将会出现警告对话框(见图 3-23)。点击 “Cancel” (取消) 按钮关闭对话框。

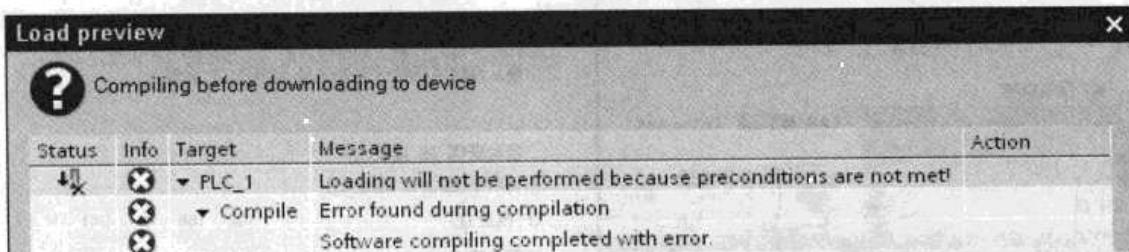


图 3-23 下载预览中的编译错误信息

点击工具栏上的编译按钮 , 在工作区下面的监视窗口的 “Info” 的 “Compile” 选项卡中可以看到详细的错误信息(见图 3-24): 网络 3 的触点没有设置地址, 网络没有用线圈结束。

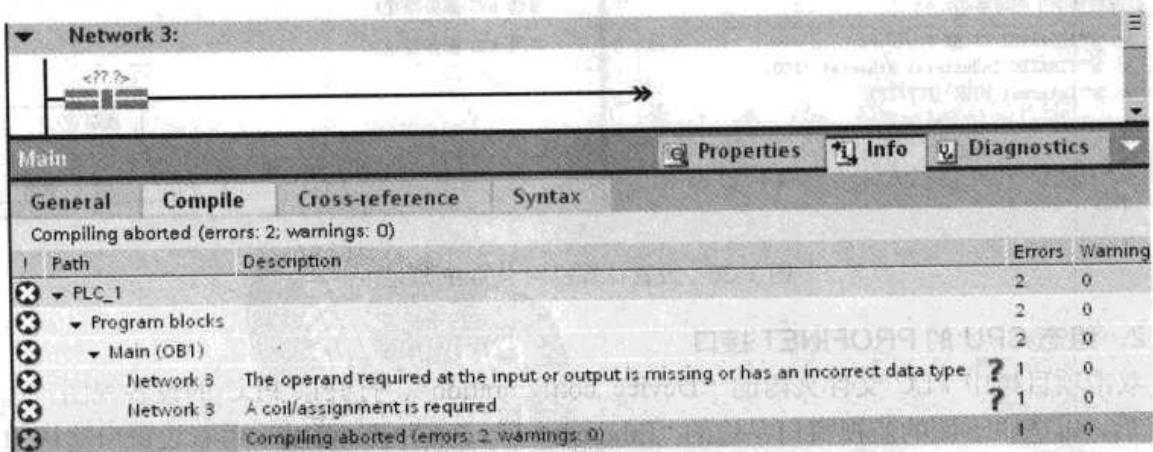


图 3-24 监视窗口中详细的编译错误信息

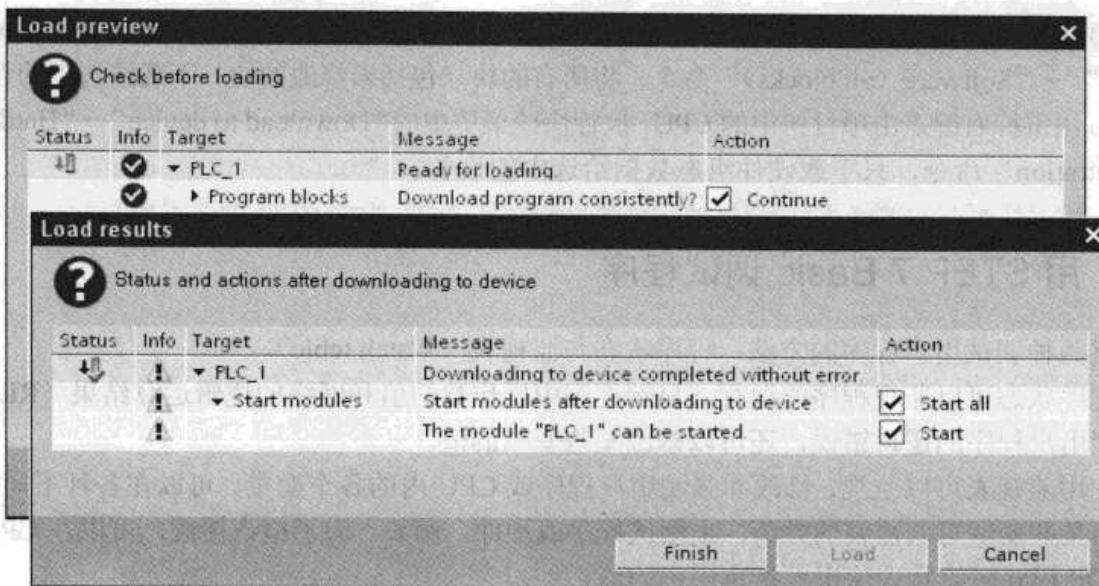


图 3-25 下载预览与下载结果对话框

删除网络 3 的触点后，错误被消除。再次点击下载按钮，出现“Load preview”（下载预览）对话框（见图 3-25），显示准备好下载的信息。选中复选框“Continue”，点击“Load”按钮，出现“Load results”对话框，用复选框选中“Start all”和“Start”（下载后启动 PLC）。点击“Finish”按钮，完成下载。下载后 CPU 进入下载之前的模式。

下载结束后，在工作区下面的监视窗口的 Info 选项卡中，可以看到下载成功的信息（见图 3-26）。

1	Start downloading to device.	8/18/2009	1:18:32 PM
1	FLC_1	8/18/2009	1:18:32 PM
✓	'IEC_Timer_0' was loaded successfully.	8/18/2009	1:21:46 PM
✓	'Main' was loaded successfully.	8/18/2009	1:21:46 PM
✓	PLC_1 started.	8/18/2009	1:26:38 PM
✓	Loading completed (errors: 0, warnings: 0)	8/18/2009	1:26:39 PM

图 3-26 监视窗口中下载成功的信息

设置第一次使用的 CPU 的 IP 地址的方法将在 7.3.2 节介绍。

#### 4. 使用菜单命令下载

1) 使用菜单命令“Online”→“Download to device”，将已编译的硬件组态数据和软件项目数据下载给选中的设备。

2) 使用菜单命令“Online”→“Extended Download to device”，设置与选中的设备的在线连接，将已编译的硬件组态数据和软件项目数据下载给选中的设备。

#### 5. 使用快捷菜单下载部分内容的方法

1) 用鼠标右键点击项目树中的 CPU，执行快捷菜单中的“Download to device”→“All”命令，将已编译的硬件组态数据和软件项目数据下载给选中的设备。

2) 用鼠标右键点击项目树中的 CPU 或“Program blocks”，执行快捷菜单中的“Download to device”→“Software”命令，只将修改过的块下载给选中的设备。

3) 用鼠标右键点击项目树中的 CPU 或“Program blocks”，执行快捷菜单中的“Download to device”→“Software (all blocks)”命令，将所有的块（包括未修改的块）下载给选中的设备。

4) 用鼠标右键点击项目树中的 CPU，执行快捷菜单中的“Download to device”→“Hardware configuration”命令，只下载硬件组态数据给选中的设备。

## 3.5 用 STEP 7 Basic 调试程序

有两种调试用户程序的方法：程序状态与监视表（Watch table）。

程序状态可以监视程序的运行，显示程序中操作数的值和网络的逻辑运算结果（RLO），查找到用户程序的逻辑错误，还可以修改某些变量的值。

使用监视表可以监视、修改和强制用户程序或 CPU 内的各个变量。可以在各种不同的情况下向某些变量写入需要的数值，来测试程序或硬件。例如，为了检查接线，可以在 CPU 处于 STOP 模式时给物理输出点指定固定的值。

### 3.5.1 用程序状态功能调试程序

#### 1. 启动程序状态监视

打开需要监视的代码块，点击程序编辑器工具栏上的 按钮，启动程序状态监视。如果在线（PLC 中的）程序与离线（计算机中的）程序不一致，将会出现图 3-27 所示的对话框。需要重新下载项目，在线、离线的项目一致后，才能启动程序状态功能。进入在线模式后，程序编辑器最上面的标题栏变为桔红色。

如果在运行时测试程序出现功能错误或程序错误，可能会对人员或财产造成严重损害，应确保不会出现这样的危险情况。

#### 2. 程序状态的显示

程序状态从选中的网络开始显示。梯形图用绿色连续线来表示状态满足，即有“能流”流过，见图 3-28 左边较浅的线。用蓝色点状线表示状态不满足，没有能流流过。用灰色连续线表示状态未知或程序没有执行，黑色表示没有连接。

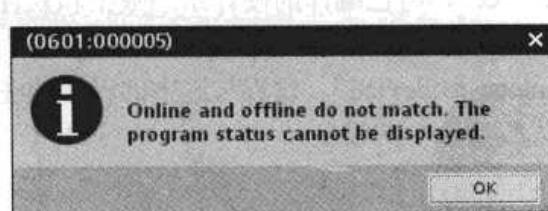


图 3-27 提示信息

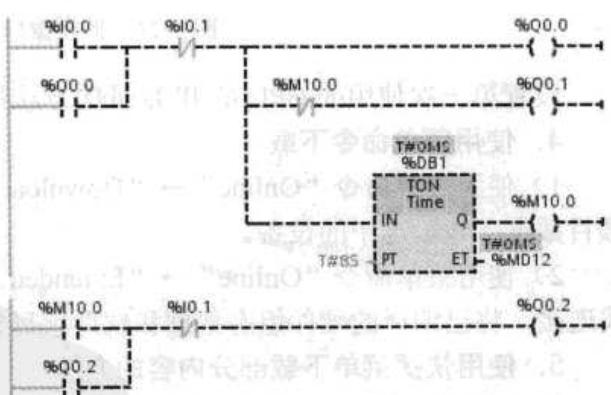


图 3-28 程序状态监视

Bool 变量为 0 状态和 1 状态时，它们的常开触点和线圈分别用蓝色点状线和绿色连续线来表示，常闭触点的显示与变量状态的关系则反之。

进入程序状态之前，梯形图中的线和元件因为状态未知，全部为黑色。启动程序状态监视后，梯形图左侧垂直的“电源”线和与它连接的水平线均为连续的绿线，表示有能流从“电源”线流出。有能流流过的处于闭合状态的触点、方框指令、线圈和“导线”均用连续的绿色线表示。

### 3. 程序状态监视应用举例

接通连接在 PLC 的输入端 I0.0 的小开关后马上断开它（模拟外接的起动按钮的操作），梯形图中 I0.0 的常开触点接通，使 Q0.0 和 Q0.1 的线圈通电并自保持（见图 3-29）。TON 的 IN 输入端有能流流入，开始定时。TON 的当前时间值（Elapsed Time, ET）从 0 开始增大，达到 PT 预置的时间 8s 时，定时器的位输出 Q 变为 1 状态，M10.0 的线圈通电，其常开触点使 Q0.2 的线圈通电（见图 3-30），其常闭触点断开，使 Q0.1 的线圈断电。电动机由星形接法切换到三角形接法运行。

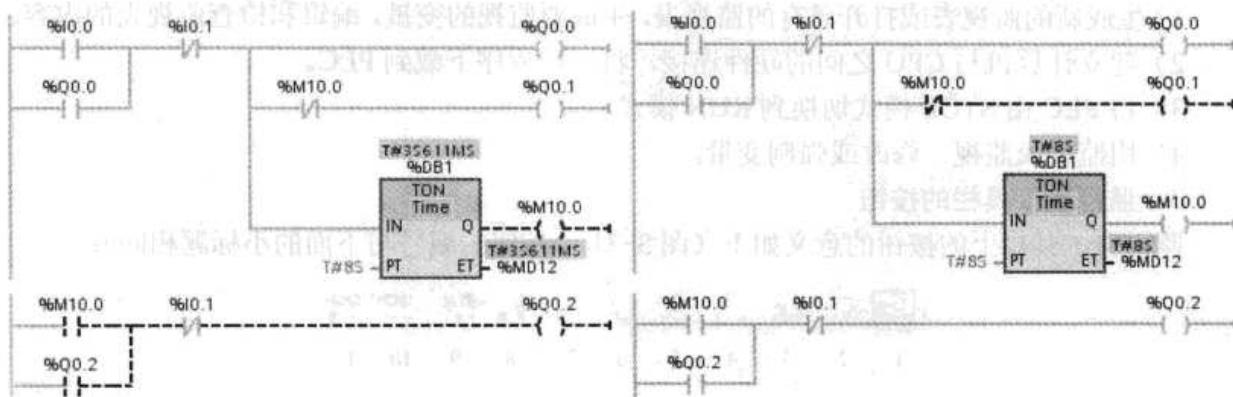


图 3-29 程序状态监视

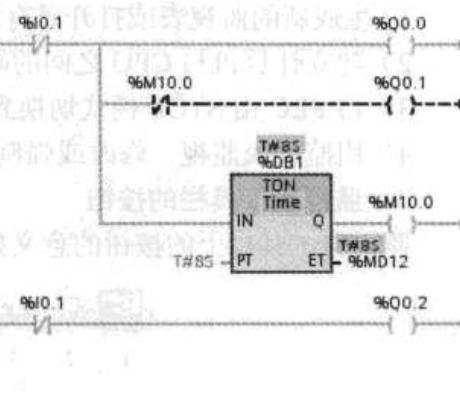


图 3-30 程序状态监视

接通 I0.1 对应的小开关后马上断开它（模拟外接的停车按钮的操作），梯形图中 I0.1 的常闭触点断开，使 Q0.0 和 Q0.2 的线圈断电，TON 被复位，其输出位 Q 变为 0 状态，M10.0 的线圈断电（见图 3-28）。

### 4. 在程序状态修改变量的值

用鼠标右键点击程序状态中的某个变量，执行出现的快捷菜单中的某个命令，可以修改该变量的值：对于 Bool 变量，执行命令“Modify”→“Modify to 1”或“Modify”→“Modify to 0”；对于其他数据类型的变量，执行命令“Modify”→“Modify value”。不能修改连接外部硬件输入电路的输入过程映像（I）的值。如果被修改的变量同时受到程序的控制（例如受线圈控制的 Bool 变量），则程序控制的作用优先。

## 3.5.2 用监视表监视与修改变量

使用程序状态功能，可以在程序编辑器中形象直观地监视梯形图程序的执行情况，触点和线圈的状态一目了然。但是程序状态功能只能在屏幕上显示一小块程序，调试较大的程序时，往往不能同时看到与某一程序功能有关的全部变量的状态。

监视表（Watch Table）可以有效地解决上述问题。使用监视表可以在工作区同时监视、修改和强制用户感兴趣的全部变量。一个项目可以生成多个监视表，以满足不同的调试要求。

监视表可以赋值或显示的变量包括过程映像(I 和 Q)、物理输入(I\_P)和物理输出(Q\_P)、位存储器 M 和数据块 DB 内的存储单元。

## 1. 监视表的功能

- 1) 监视变量：显示用户程序或 CPU 中变量的当前值。
- 2) 修改变量：将固定值赋给用户程序或 CPU 中的变量，这一功能可能会影响到程序运行结果。
- 3) 对物理输出赋值：允许在停机状态下将固定值赋给 CPU 的每一个输出点 Q，可用于硬件调试时检查接线。
- 4) 强制变量：给物理输入/物理输出点赋予一个固定值，用户程序的执行不会影响被强制的变量的值。
- 5) 可以选择在扫描循环周期开始、结束或切换到 STOP 时读写变量的值。

## 2. 用监视表监视和修改变量的基本步骤

- 1) 生成新的监视表或打开已有的监视表，生成要监视的变量，编辑和检查监视表的内容。
- 2) 建立计算机与 CPU 之间的硬件连接，将用户程序下载到 PLC。
- 3) 将 PLC 由 STOP 模式切换到 RUN 模式。
- 4) 用监视表监视、修改或强制变量。

## 3. 监视表工具栏的按钮

监视表工具栏上的按钮的意义如下（图 3-31 中按钮的编号与下面的小标题相同）：



图 3-31 监视表工具栏的按钮

- 1) 显示或隐藏 Modify value (修改值) 列。
- 2) 显示或隐藏所有的扩展模式列。在扩展模式，将增加“Monitor with trigger”(使用触发器的监视) 和“Modify with trigger”(使用触发器的修改) 列。
- 3) 显示或隐藏所有的强制列。
- 4) 立即修改一次所有选中的变量，与定义的用户程序的触发点无关。
- 5) 修改所有选中的变量，与定义的用户程序的触发点有关。
- 6) 使能或禁止物理输出，必须在 STOP 模式修改物理输出。
- 7) 启动对选中的变量的强制，或改变强制值。
- 8) 停止对变量的强制。
- 9) 显示 CPU 中被强制的所有地址。
- 10) 启动或禁止对变量的监视 (Monitor all)。启动后监视表中的变量值将跟随 PLC 中的变量值动态变化。
- 11) 立即读取一次 PLC 中的变量值 (Monitor now)，并用监视表显示它们。

## 4. 生成监视表

打开项目树中 PLC 的“Watch tables”文件夹，双击其中的“Add new watch table”，生成一个新的监视表，并在工作区自动打开它。根据需要，可以为一台 PLC 生成多个监视表。应将有关联的变量放在同一个监视表内。

## 5. 在监视表中输入变量

在监视表的“Name”列输入“PLC Tags”(变量表)中定义过的变量的符号地址，“Address”

(地址)列将会自动出现该变量的地址。在地址列输入变量表中定义过的地址，“Name”列将会自动地出现它的名称。

如果输入了错误的变量名称或地址，将在出错的单元下面出现红色背景的错误提示方框。

可以使用监视表的“Display format”列默认的显示格式，也可以用鼠标右键点击该列的某个单元，选中弹出的快捷菜单中需要的显示格式。图 3-32 的监视表用二进制格式(Bin)显示 MW20，可以同时显示和分别修改 M20.0~M21.7 这 16 个位变量。这一方法用于 I、Q 和 M，可以用字节(8 位)、字(16 位)或双字(32 位)来监视和修改位变量。

	Name	Address	Display format	Monitor value	Modify value
1	"START"	%I0.0	Bool	<input type="checkbox"/> FALSE	
2	"STOP"	%I0.1	Bool	<input type="checkbox"/> FALSE	
3	"PowerKM"	%Q0.0	Bool	<input checked="" type="checkbox"/> TRUE	<input type="checkbox"/> FALSE
4	"StartKM"	%Q0.1	Bool	<input checked="" type="checkbox"/> TRUE	<input checked="" type="checkbox"/>
5	"triangleKM"	%Q0.2	Bool	<input type="checkbox"/> FALSE	
6	"Timer1OUT"	%M1.0	Bool	<input type="checkbox"/> FALSE	
7	"Tag_1"	%MW20	Bin	0011_0101_1010_1110	
8	"Time1"	%MD12	Time	3s_997ms	
9					

图 3-32 在线的监视表

复制 PLC 变量表中的变量名称，然后将它粘贴到监视表的“Name”列，可以快速生成监视表中的变量。具体方法如下：

双击打开项目树中的“PLC tags”，用鼠标点击某个变量最左边的序号单元，该变量被选中，整个行的背景色加深。按住计算机的〈Ctrl〉键，用同样的方法同时选中其他变量。用鼠标右键点击选中的变量，执行出现的快捷菜单中的“Copy”命令，将选中的变量复制到剪贴板。

双击打开项目树中的监视表，用右键点击空白行，执行出现的快捷菜单中的“Paste”命令，将复制的变量粘贴到监视表。

## 6. 监视变量

可以用监视表上面的工具栏上的按钮来执行各种功能。与 CPU 建立在线连接后，点击工具栏上的 按钮，启动监视功能，将在“Monitor value”(监视值)列显示变量的实际值。

如果触发条件设为“Monitor all”，再次点击该按钮，将关闭监视功能。点击工具栏上的 按钮，可以对所选变量的数值作一次立即更新，该功能主要用于 STOP 模式下的监视和修改。

位变量为 TRUE(1 状态)时，监视值列的方形指示灯为绿色。位变量为 FALSE(0 状态)时，指示灯为灰色。

图 3-33 中的 D12 是定时器的当前时间值，在定时器的定时过程中，MD12 的值不断增大。

## 7. 修改变量

首先在要修改的变量的“Modify value”列输入变量新的值。输入 Bool 变量的修改值 0 或 1 后，点击监视表其他地方，它们将自动变为“False”(假)或“True”(真)。

点击工具栏上的 按钮，或用鼠标右键点击变量，执行出现的快捷菜单中的“Modify now”(见图 3-33)命令，将修改值立即送入 CPU。

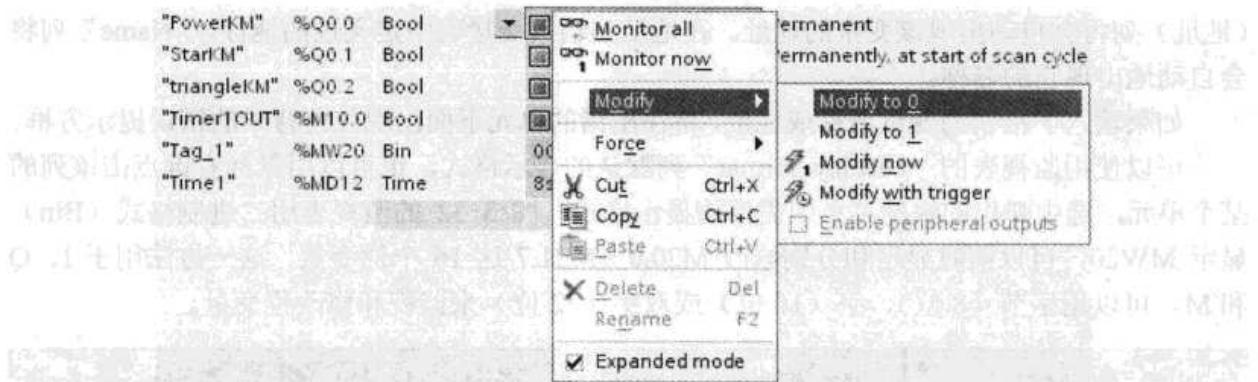


图 3-33 用监视表修改变量

用鼠标右键点击某个位变量，执行出现的快捷菜单中的“Modify to 0”或“Modify to 1”命令，可以将选中的变量修改为 0 或 1。

点击工具栏上的 $\text{F}_\text{6}$ 按钮，或执行出现的快捷菜单中的“Modify with trigger”命令，在定义的用户程序的触发点，修改所有选中的变量。

如果没有启动监视功能，执行快捷菜单中的“Modify now”命令，将读取一次监视值。

在 RUN 模式修改变量时，各变量同时又受到用户程序的控制。假设用户程序运行的结果使 Q0.0 的线圈断电，用监视表不可能将 Q0.0 修改为 1 状态。在 RUN 模式不能改变 I 区的数字量输入点的状态，因为它们的状态取决于外部输入电路的通/断状态。

在程序运行时如果修改变量值出错，可能导致人身或财产的损害。执行修改功能之前，应确认不会有危险情况出现。

## 8. 在 STOP 模式改变物理输出的状态

在调试设备时，这一功能可以用来检查输出点连接的过程设备的接线是否正确。以 Q0.0 为例，操作的步骤如下：

### 1) 在监视表中输入物理输出点 Q0.0:P (见图 3-34)。

Name	Address	Display format	Monitor value	Monitor with trigger	Modify with trigger	Modify value	$\text{F}_\text{6}$
1 "Tag_1":P %I0.0:P	Bool	<input type="checkbox"/> FALSE	Permanent	Permanent	<input type="checkbox"/>		$\text{F}_\text{6}$
2 "Tag_2":P %Q0.0:P	Bool	<input type="checkbox"/>	Permanent	<input type="checkbox"/> Permanent	<input type="checkbox"/>	TRUE	$\text{F}_\text{6}$
3			It is not possible to monitor peripheral outputs.				

图 3-34 在 STOP 模式改变物理输出的状态

- 2) 将 CPU 切换到 STOP 模式。
- 3) 点击监视表工具栏上的 $\text{F}_\text{6}$ 按钮，切换到扩展模式，出现与“trigger”有关的两列。
- 4) 点击工具栏上的 $\text{F}_\text{1}$ 按钮，启动监视功能。
- 5) 点击工具栏上的 $\text{F}_\text{2}$ 按钮，出现“Enable peripheral outputs”(使能物理输出)对话框，点击“Yes”按钮确认。
- 6) 用鼠标右键点击 Q0.0:P 所在的第 2 行，执行出现的快捷菜单中的“Modify”→“Modify to 1”或“Modify”→“Modify to 0”命令(见图 3-33)，CPU 上 Q0.0 对应的 LED(发光二极管)亮或熄灭，监视表中“Modify value”列的值随之而变，表示命令被送给物理输出点。

CPU 切换到 RUN 模式后，工具栏上的 按钮变为灰色，“Enable peripheral outputs” 功能被禁止，Q0.0 受到用户程序的控制。

如果有输入点或输出点被强制，则不能使用这一功能。为了在 STOP 模式下允许物理输出，应取消强制功能。

因为 CPU 只能改写，不能读取物理输出变量 Q0.0:P 的值，符号 表示该变量被禁止监视（不能读取）。将光标放到图 3-34 第 2 行的“Monitor Value”（监视值）列时，将会出现图中黄色背景的帮助信号，提示不能监视物理输出。

### 9. 定义监视表的触发器

“trigger”（触发器）用来设置在扫描循环的哪一点来监视或修改选中的变量。可以选择在扫描循环开始、扫描循环结束或从 RUN 切换到 STOP 时监视或修改某个变量。

点击监视表工具栏上的 按钮，切换到扩展模式，出现与“trigger”有关的两列。

点击这两列的某个单元右边的 按钮（见图 3-35），用出现的下拉式列表设置监视或修改该行变量的触发方式。

Name	Address	Display format	Monitor value	Monitor with trigger	Modify with trigger	Modify value	
"REQ1"	%M1.0	Bool		Permanent		Permanent	!
"Tag_2"	%M1.1	Bool		Permanent			
	%I0.0	Bool		Permanently, at start of scan cycle			
"Tag_8"	%MW6	Hex		Once only, at start of scan cycle		1345	!
				Permanently, at end of scan cycle			
				Once only, at end of scan cycle			
				Permanently, at transition to STOP			
				Once only, at transition to STOP			

图 3-35 定义监视表的触发方式

触发方式（见图 3-35 和表 3-4）可以选择触发一次（Once only）或每个循环触发一次（Permanent，固定的）。如果设置为触发一次，点击一次工具栏上监视变量或修改变量的按钮，执行一次相应的操作。

表 3-4 监视表的触发方式

触发器类型	描述
固定的（permanent）	连续采集数据
在扫描循环开始时	固定的（permanent）：在扫描循环开始时，在 CPU 读取输入之后连续采集数据 仅一次（once only）：在扫描循环开始时，在 CPU 读取输入之后采集一次数据
在扫描循环结束时	固定的：在扫描循环结束时，在 CPU 写输出之前连续采集数据 仅一次：在扫描循环结束时，在 CPU 写输出之前采集一次数据
切换到 STOP 时	固定的：在 CPU 切换到 STOP 时连续采集数据 仅一次：在 CPU 切换到 STOP 时采集一次数据

1) 修改输出的最佳触发事件是在扫描循环结束、CPU 马上要写输出模块之前。在扫描循环开始时监视输出点的值，以决定给物理输出写入了什么值。同样地，为了检查程序逻辑，并与实际的 I/O 特征比较，在 CPU 将数值写给物理输出点之前监视输出点。

2) 修改一个输入点值的最佳触发事件是在扫描循环开始、CPU 读输入之后，用户程序使用输入值之前。

如果在扫描循环开始时修改了输入，应监视在扫描循环结束时的输入值，以保证从扫描循环开始到扫描循环结束，输入值没有变化。如果二者的值有区别，用户程序可能被写入错误的输入。

3) 为了诊断 CPU 为什么进入 STOP 模式，可以使用“Transition to STOP”（转换到 STOP）触发器，以捕获最后的过程值。

### 3.5.3 用监视表强制变量

#### 1. 强制 CPU 中的变量值

可以用监视表给用户程序中的单个变量指定固定的值，这一功能被称为强制（Force）。强制应在与 CPU 建立了在线连接时进行。使用强制功能时，不正确的操作可能会危及人员的生命或健康，造成设备或整个工厂的损失。

S7-1200 系列 PLC 只能强制物理 I/O 点，例如强制 I0.0:P 和 Q0.0:P 等。不能强制组态时指定给 HSC（高速计数器）、PWM（脉冲宽度调制）和 PTO（脉冲列输出）的 I/O 点。在测试用户程序时，可以通过强制 I/O 点来模拟物理条件，例如用来模拟输入信号的变化。

在执行用户程序之前，强制值被用于输入过程映像，在处理程序时，使用的是输入点的强制值。

在写物理输出点时，强制值被送给输出过程映像，输出值被强制值覆盖。强制值在物理输出点出现，并且被用于过程。

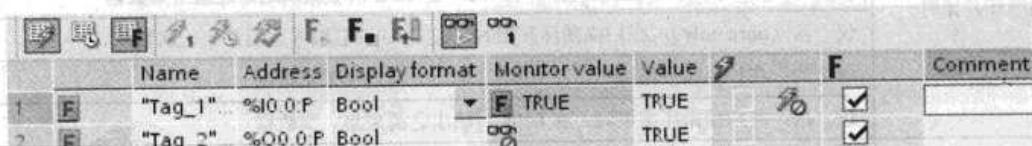
变量被强制的值不会因为用户程序的执行而改变。被强制的变量只能读取，不能用写访问来改变其强制值。

输入、输出点被强制后，即使编程软件被关闭，或编程计算机与 CPU 的在线连接断开，或 CPU 断电，强制值都被保持在 CPU 中，直到在线时用编程软件停止强制功能。

用存储卡将带有强制点的程序装载到别的 CPU 时，将继续程序中的强制功能。

#### 2. 强制的操作步骤

- 1) 在监视表中输入物理输入点 I0.0:P 和物理输出点 Q0.0:P（见图 3-36）。
- 2) 将 CPU 切换到 RUN 模式。
- 3) 点击工具栏上的  按钮，启动监视功能。
- 4) 点击工具栏上的  按钮（见图 3-36），监视表出现标有 “F”的强制列。



	Name	Address	Display format	Monitor value	Value	F	Comment
1		"Tag_1"	%I0.0:P	Bool	<input checked="" type="checkbox"/> TRUE	TRUE	
2		"Tag_2"	%Q0.0:P	Bool	<input checked="" type="checkbox"/> 0	TRUE	

图 3-36 用监视表强制 I/O 变量

- 5) 在 I0.0:P 的 “Value” 列输入 1，点击其他地方，1 变为 TRUE。
- 6) 用 F 列的复选框选中该变量（复选框内打勾），复选框的后面出现中间有感叹号的黄色三角形，表示需要强制该变量。工具栏上的  按钮变为深色，表示可以强制变量。
- 7) 点击工具栏上的  按钮，或者用鼠标右键点某个变量，执行出现的快捷菜单中的“Force all”（强制所有）命令，启动所有在 F 列激活了强制功能的变量的强制。

第一次强制某个变量时，出现“Force all”对话框（见图 3-37 的左图），以后修改变量的强制值时，点击 **F** 按钮，出现“Replace forcing”（改变强制值，见图 3-37 的右图）信息，点击“**Yes**”按钮确认。强制成功后监视表中该行“F”列黄色的三角形符号消失，被强制的变量所在的行最左边和输入点的“Monitor value”列出现红色的标有“F”的小方框（见图 3-36），表示该变量被强制。

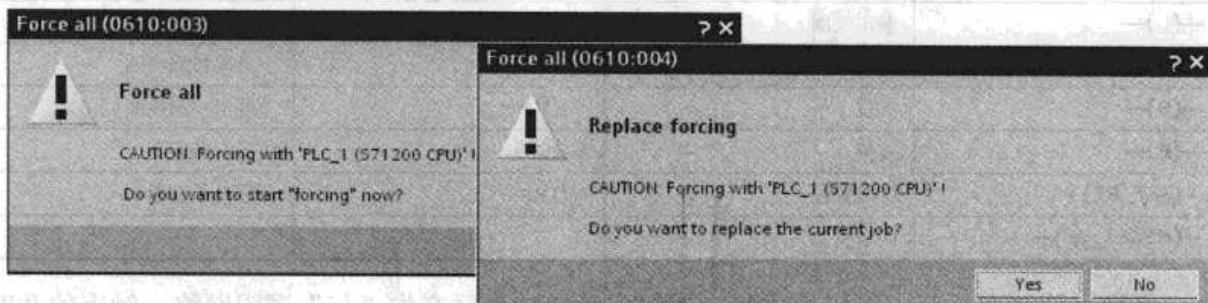


图 3-37 强制数值对话框

I0.0 被强制为 1 状态时，它对应的发光二极管不会亮，但是被强制的值在程序中起作用。

用同样的方法强制 Q0.0:P（见图 3-36），CPU 上 Q0.0 对应的 LED 亮。

也可以用鼠标右键点击要强制的位变量，执行出现的快捷菜单中的“Force to 0”或“Force to 1”命令，点击出现的对话框中的“**Yes**”按钮确认后，将选中的 I/O 变量的值强制为 0 或 1。

### 3. 停止强制

点击工具栏上的 **F** 按钮，或执行快捷菜单中的“Stop forcing”命令，停止对所有地址的强制。被强制的变量最左边和输入点的“Monitor value”列红色的标有“F”的小方框消失，表示强制被停止。复选框后面的黄色的三角形符号重新出现，表示该地址被选择强制，但是 CPU 中的变量没有被强制。

为了停止对单个变量的强制，可以清除该变量的强制列的复选框，然后重新启动强制。

### 4. 显示 CPU 所有被强制的变量

在调试结束，程序正式运行之前，必须停止对所有被强制的变量的强制，否则会影响程序的正常运行，甚至造成事故。

上述停止强制的操作只能停止当前打开的监视表中被强制的变量。如果监视表不止一个，在别的监视表中也有变量被强制，监视表的表头最左边有符号 **F** 闪动。点击工具栏上的 **F** 按钮，或执行快捷菜单中的“Show force values”命令，将在当前监视表显示所有的监视表中被强制了的地址，此时可以用当前的监视表停止全部被强制的变量。

如果被强制的全部变量在同一个监视表内，不能使用“Show force values”命令。

## 3.6 位逻辑指令

### 3.6.1 触点指令与线圈指令

#### 1. 常开触点与常闭触点

常开触点（见表 3-5）在指定的位为 1 状态（ON）时闭合，为 0 状态（OFF）时断开。常闭触点在指定的位为 1 状态时断开，为 0 状态时闭合。

表 3-5 位逻辑指令

指令	描述	指令	描述
—+—	常开触点	RS 触发器	置位优先位锁存器
—N—	常闭触点	SR 触发器	复位优先位锁存器
—NOT—	取反触点	—P—	上升沿检测触点
—( )—	输出线圈	—N—	下降沿检测触点
—(/)—	取反输出线圈	—(P)—	上升沿检测线圈
—(S)—	置位	—(N)—	下降沿检测线圈
—(R)—	复位	P_TRIG	上升沿触发器
—(SET_BF)—	区域置位	N_TRIG	下降沿触发器
—(RESET_BF)—	区域复位		

将触点连接在一起，可以生成组合逻辑。触点的串联产生“与”逻辑网络，触点的并联产生“或”逻辑网络。

位地址 I 或 Q 的值从过程映像寄存器中读取。CPU 周期性地扫描连接在输入端的外接电路提供的物理输入信号，并将它们保存在过程映像输入寄存器中。

在 I 地址之后添加“:P”（例如“%I0.2:P”），可以立即直接读取物理输入。立即读取不会刷新过程映像输入。

## 2. NOT 取反触点

NOT 触点用来转换能流输入的逻辑状态。如果没有能流流入 NOT 触点，则有能流流出（见图 3-38 的左图）。如果有能流流入 NOT 触点，则没有能流流出（见图 3-38 的右图）。



图 3-38 NOT 触点

## 3. 输出线圈

线圈输出指令将线圈的状态写入指定的地址，线圈通电时写入 1，断电时写入 0。如果是 Q 区的地址，CPU 将输出的值传送给过程映像输出寄存器。在 RUN 模式，CPU 不停地扫描输入信号，根据用户程序的逻辑处理输入状态，通过向过程映像输出寄存器写入新的输出状态值来作出响应。在写输出阶段，CPU 将存储在过程映像寄存器中的新的输出状态传送给对应的输出电路。

可以用 Q0.4:P 的线圈将位数据值立即写入过程映像输出 Q0.4，同时直接写给对应的物理输出点。

反相输出线圈中间有“/”符号，如果有能流流过 M4.1 的反相输出线圈（见图 3-39 的左图），则 M4.1 的输出位为 0 状态，其常开触点断开（见图 3-39 的右图），反之 M4.1 的输出位为 1 状态，其常开触点闭合。



图 3-39 反相输出线圈

### 3.6.2 其他位逻辑指令

#### 1. 置位复位指令

S (Set, 置位或置 1) 指令将指定的地址位置位（变为 1 状态并保持）。

R (Reset, 复位或置 0) 指令将指定的地址位复位（变为 0 状态并保持）。

置位指令与复位指令最主要的特点是有记忆和保持功能。如果图 3-40 中 I0.4 的常开触点闭合，Q0.5 变为 1 状态并保持该状态。即使 I0.4 的常开触点断开，Q0.5 也仍然保持 1 状态。在程序状态中，用 Q0.5 的 S 和 R 线圈连续的绿色圆弧和绿色的字母表示 1 状态，用间断的蓝色圆弧和蓝色的字母表示 0 状态。

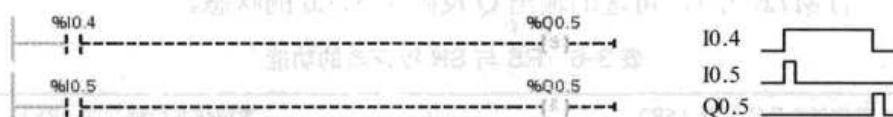


图 3-40 置位与复位指令

I0.5 的常开触点闭合时，Q0.5 变为 0 状态并保持该状态，即使 I0.5 的常开触点断开，Q0.5 也仍然保持 0 状态。

这种记忆功能也可以用图 3-41 中的起停保电路来实现。按下起动按钮，图中 I0.4 的常开触点接通，如果这时未按停止按钮，则 I0.5 的常闭触点接通，Q0.5 的线圈“通电”，它的常开触点同时接通。放开起动按钮，I0.4 的常开触点断开，“能流”经 Q0.5 的常开触点和 I0.5 的常闭触点流过 Q0.5 的线圈，Q0.5 仍为 ON，这就是所谓的“自锁”或“自保持”功能。按下停止按钮，I0.5 的常闭触点断开，使 Q0.5 的线圈“断电”，其常开触点断开，以后即使放开停止按钮，I0.5 的常闭触点恢复接通状态，Q0.5 的线圈仍然“断电”。



图 3-41 起停保电路

在实际电路中，起动信号和停止信号可以由多个触点组成的串、并联电路提供。

#### 2. 多点置位复位指令

SET\_BF (Set bit field, 多点置位) 将指定的地址开始的连续的若干个位地址置位（变为 1 状态并保持）。在图 3-42 的 I0.6 的上升沿（从 0 状态变为 1 状态），从 M5.0 开始的 4 个连续的位被置位为 1 并保持 1 状态。

RESET\_BF (Reset bit field, 多点复位) 将指定的地址开始的连续的若干个位地址复位（变为 0 状态并保持）。在图 3-42 的 M4.4 的下降沿（从 1 状态变为 0 状态），从 M5.4 开始的 3 个连续的位被复位为 0 并保持 0 状态。

与 S7-200 和 S7-300/400 不同，S7-1200 的梯形图允许在一个网络内输入多个独立电路（见图 3-42）。

#### 3. 置位优先锁存器与复位优先锁存器

图 3-43 中的 SR 是复位优先锁存器，其输入/输出关系见表 3-6，两种触发器的区别仅在于表的最下面一行。在置位（S）和复位（R1）信号同时为 1 时，方框上的输出位 M7.2 被复

位为 0。可选的输出 Q 反映了 M7.2 的状态。

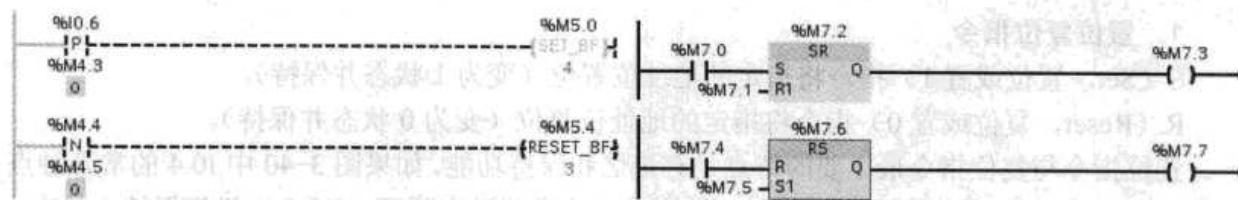


图 3-42 边沿检测触点与多位置位复位

图 3-43 SR 锁存器与 RS 锁存器

RS 是置位优先锁存器，其输入/输出关系见表 3-6。在置位 (S1) 和复位 (R) 信号同时为 1 时，方框上的 M7.6 为 1。可选的输出 Q 反映了 M7.6 的状态。

表 3-6 RS 与 SR 锁存器的功能

复位优先位锁存器 (SR)			置位优先位锁存器 (RS)		
S	R1	输出位	S1	R	输出位
0	0	保持前一状态	0	0	保持前一状态
0	1	0	0	1	0
1	0	1	1	0	1
1	1	0	1	1	1

#### 4. 边沿检测触点指令

图 3-42 中间有 P 的触点是上升沿检测触点，如果输入信号 I0.6 由 0 状态变为 1 状态（即输入信号 I0.6 的上升沿），则该触点接通一个扫描周期。边沿检测触点不能放在电路结束处。

P 触点下面的 M4.3 为边沿存储位，用来存储上一次扫描循环时 I0.6 的状态。通过比较输入信号的当前状态和上一次循环的状态，来检测信号的边沿。如果输入信号的状态发生变化，则通过向输出位写入 TRUE 来报告边沿。反之，将 FALSE 写入输出位。边沿存储位的地址只能在程序中使用一次，它的状态不能在其他地方被改写。

只能使用 M、全局 DB 和静态局部变量 (Static) 来作边沿存储位，不能使用临时局部变量或 I/O 变量来作边沿存储位。

中间有 N 的触点是下降沿检测触点，如果图 3-42 中的输入信号 M4.4 由 1 状态变为 0 状态（即 M4.4 的下降沿），RESET\_BF 的线圈“通电”一个扫描循环周期。N 触点下面的 M4.5 为边沿存储位。

#### 5. 边沿检测线圈指令

中间有 P 的线圈（见图 3-44）是上升沿检测线圈，仅在流进该线圈的能流的上升沿（线圈由断电变为通电），输出位 M6.1 为 1 状态。M6.2 为边沿存储位。

中间有 N 的线圈是下降沿检测线圈，仅在流进该线圈的能流的下降沿（线圈由通电变为断电），输出位 M6.3 为 1 状态。M6.4 为边沿存储位。

边沿检测线圈不会影响逻辑运算结果 RLO，它对能流是畅通无阻的，其输入端的逻辑运算结果被立即送给线圈的输出端。边沿检测线圈可以放置在网络的中间或网络的最右边。

在运行时用外接的小开关使 I0.7 变为 1 状态，I0.7 的常开触点闭合，能流经 P 线圈和 N 线圈流过 M6.5 的线圈。在 I0.7 的上升沿，M6.1 的常开触点闭合一个扫描周期，使 M6.6 置位。在 I0.7 的下降沿，M6.3 的常开触点闭合一个扫描周期，使 M6.6 复位。

## 6. P\_TRIG 指令与 N\_TRIG 指令

在流进 P\_TRIG 指令的 CLK 输入端（见图 3-45）的能流的上升沿（能流刚出现），Q 端输出脉冲宽度为一个扫描周期的能流，使 M8.1 置位。方框下面的 M8.0 是脉冲存储器位。

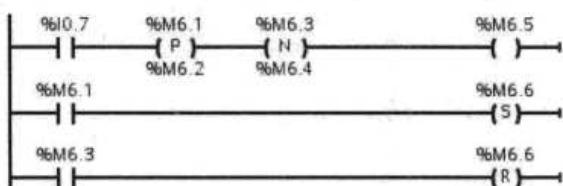


图 3-44 边沿检测线圈指令

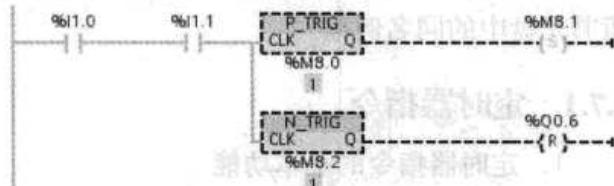


图 3-45 P\_TRIG 指令与 N\_TRIG 指令

在流进 N\_TRIG 指令的 CLK 输入端的能流的下降沿（能流刚消失），Q 端输出脉冲宽度为一个扫描周期的能流，使 Q0.6 复位。指令方框下面的 M8.2 是脉冲存储器位。

P\_TRIG 指令与 N\_TRIG 指令不能放在电路的开始处和结束处。

在设计程序时应考虑输入和存储位的初始状态，是允许还是应避免首次扫描的边沿检测。

## 7. 边沿检测指令的比较

下面比较 3 种边沿检测指令的功能（以上升沿检测为例）：

在 **P** 触点上面的地址的上升沿，该触点接通一个扫描周期。因此 P 触点用于检测触点上面的地址的上升沿，并且直接输出上升沿脉冲。

在流过 **P** 线圈的能流的上升沿，线圈上面的地址在一个扫描周期为 1 状态。因此 P 线圈用于检测能流的上升沿，并用线圈上面的地址来输出上升沿脉冲。

在流入 P\_TRIG 指令的 CLK 端的能流的上升沿，Q 端输出一个扫描周期的能流。因此 P\_TRIG 指令用于检测能流的上升沿，并且直接输出上升沿脉冲。

如果 P\_TRIG 指令左边只有 I1.0 的常开触点，可以用 I1.0 的 P 触点来代替它们。

## 8. 故障显示电路

**【例 3-1】** 设计故障信息显示电路，故障信号 I0.0 为 1 状态，Q0.7 控制的指示灯以 1Hz 的频率闪烁。操作人员按复位按钮 I0.1 后，如果故障已经消失，则指示灯熄灭。如果没有消失，则指示灯转为常亮，直至故障消失。

故障信息显示电路和信号波形图如图 3-46 所示。在设置 CPU 的属性时，令 MB0 为时钟存储器字节（见图 2-32），其中的 M0.5 提供周期为 1s 的时钟脉冲。出现故障时，将 I0.0 提供的故障信号用 M2.1 锁存起来，M2.1 和 M0.5 的常开触点组成的串联电路使 Q0.7 控制的指示灯以 1Hz 的频率闪烁。按下复位按钮 I0.1，故障锁存信号 M2.1 被复位为 0 状态。如果这时故障已经消失，则指示灯熄灭。如果没有消失，则 M2.1 的常闭触点与 I0.0 的常开触点组成的串联电路使指示灯转为常亮，直至故障消失，I0.0 变为 0 状态。

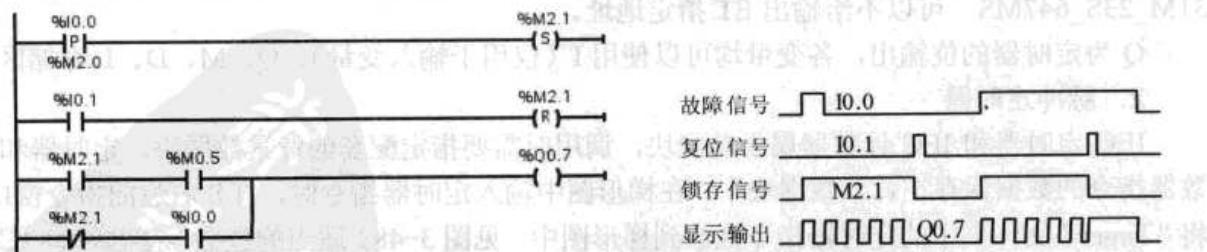


图 3-46 故障显示电路与波形图

## 3.7 定时器与计数器指令

S7-1200 采用 IEC 标准的定时器和计数器指令。本节的程序在项目“TimerCounter”中（见随书光盘中的同名例程）。

### 3.7.1 定时器指令

#### 1. 定时器指令的基本功能

S7-1200 有 4 种定时器，图 3-47 给出了它们的基本功能。

1) 脉冲定时器 TP：在输入信号 IN 的上升沿产生一个预置宽度的脉冲，图中的  $t$  为定时器的预置值。

2) 接通延时定时器 TON：输入 IN 变为 1 状态后，经过预置的延迟时间，定时器的输出 Q 变为 1 状态。输入 IN 变为 0 状态时，输出 Q 变为 0 状态。

3) 断开延时定时器 TOF：输入 IN 为 1 状态时，输出 Q 为 1 状态。输入 IN 变为 0 状态后，经过预置的延迟时间，输出 Q 变为 0 状态。

4) 保持型接通延时定时器 TONR：输入 IN 变为 1 状态后，经过预置的延迟时间，定时器的输出 Q 变为 1 状态。输入 IN 的脉冲宽度可以小于时间预置值。

定时器的输入 IN（见图 3-48）为启动定时的使能输入端，IN 从 0 状态变为 1 状态时，启动 TP、TON 和 TONR 开始定时。IN 从 1 状态变为 0 状态时，启动 TOF 开始定时。

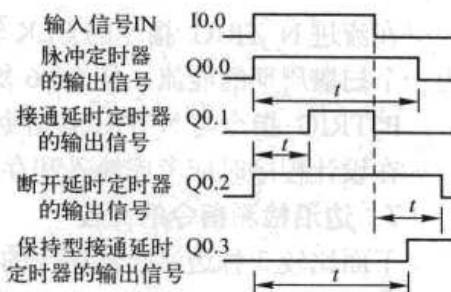


图 3-47 定时器的基本功能

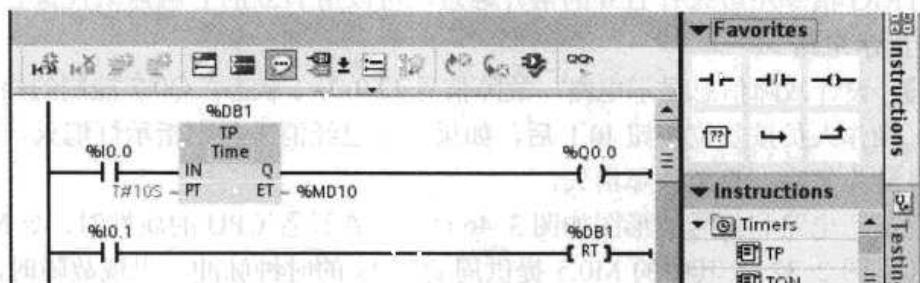


图 3-48 脉冲定时器

PT（Preset Time）为时间预置值，ET（Elapsed Time）为定时开始后经过的时间，或称为当前时间值，它们的数据类型为 32 位的 Time，单位为 ms，最大定时时间长达 T#24D\_20H\_31M\_23S\_647MS。可以不给输出 ET 指定地址。

Q 为定时器的位输出，各变量均可以使用 I（仅用于输入变量）、Q、M、D、L 存储区。

#### 2. 脉冲定时器

IEC 定时器和 IEC 计数器属于功能块，调用时需要指定配套的背景数据块，定时器和计数器指令的数据保存在背景数据块中。在梯形图中输入定时器指令时，打开右边的指令窗口，将“Timer”文件夹中的定时器指令拖放到梯形图中（见图 3-48）适当的位置。在出现的“Call options”对话框中（见图 3-13），可以修改将要生成的背景数据块的名称，或采用默认的名称。

点击“OK”按钮，自动生成数据块。

脉冲定时器类似于数字电路中上升沿触发的单稳态电路。在 IN 输入信号的上升沿，Q 输出变为 1 状态，开始输出脉冲。达到 PT 预置的时间时，Q 输出变为 0 状态（见图 3-50 的波形 A、B、E）。IN 输入的脉冲宽度可以小于 Q 端输出的脉冲宽度。在脉冲输出期间，即使 IN 输入又出现上升沿（见波形 B），也不会影响脉冲的输出。

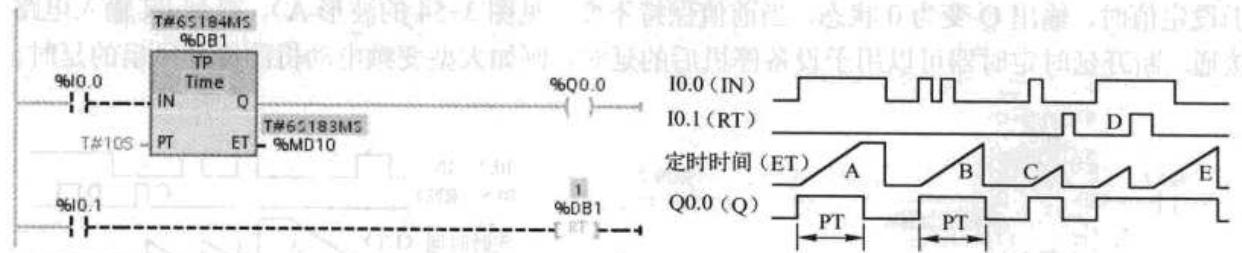


图 3-49 脉冲定时器的程序状态

图 3-50 脉冲定时器的波形图

用程序状态功能可以观察定时时间的变化情况（见图 3-49）。定时开始后，定时时间从 0ms 开始不断增大，达到 PT 预置的时间时，如果 IN 为 1 状态，则当前时间值保持不变（见波形 A）。如果 IN 为 0 状态，则定时时间为 0s（见波形 B）。

定时器指令可以放在网络的中间或结束处。IEC 定时器没有编号，在使用对定时器复位的 RT 指令时，可以用背景数据块的编号或符号名来指定需要复位的定时器。如果没有必要，不用对定时器使用 RT 指令。

图 3-49 中的 I0.1 为 1 时，定时器复位线圈 RT 通电，定时器被复位。如果此时正在定时，且 IN 输入为 0 状态，将使定时时间清零，Q 输出也变为 0 状态（见波形 C）。如果此时正在定时，且 IN 输入为 1 状态，将使定时时间清零，但是 Q 输出保持 1 状态（见波形 D）。复位信号 I0.1 变为 0 状态时，如果 IN 输入为 1 状态，将重新开始定时（见波形 E）。

### 3. 接通延时定时器

接通延时定时器（TON）的使能输入端（IN）的输入电路由断开变为接通时开始定时。定时时间大于等于预置时间 PT 指定的设定值时，输出 Q 变为 1 状态，当前时间值 ET 保持不变（见图 3-52 中的波形 A）。

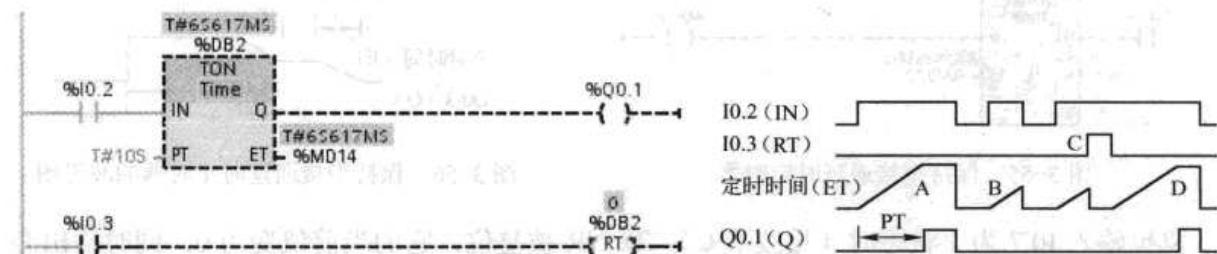


图 3-51 接通延时定时器

图 3-52 接通延时定时器的波形图

IN 输入端的电路断开时，定时器被复位，定时时间被清零，输出 Q 变为 0 状态。CPU 第一次扫描时，定时器输出 Q 被清零。如果输入 IN 在未达到 PT 设定的时间时变为 0 状态（见波形 B），输出 Q 保持 0 状态不变。

图 3-51 中的 I0.3 为 1 状态时，定时器复位线圈 RT 通电（见波形 C），定时器被复位，

定时时间被清零, Q 输出端变为 0 状态。I0.3 变为 0 状态时, 如果 IN 输入为 1 状态, 将开始重新定时(见波形 D)。

#### 4. 断开延时定时器指令

断开延时定时器(TOF)的 IN 输入电路接通时, 输出 Q 为 1 状态, 当前值被清零。输入电路由接通变为断开时(IN 输入的下降沿)开始定时, 当前值从 0 逐渐增大。当前值大于等于设定值时, 输出 Q 变为 0 状态, 当前值保持不变(见图 3-54 的波形 A), 直到 IN 输入电路接通。断开延时定时器可以用于设备停机后的延时, 例如大型变频电动机的冷却风扇的延时。

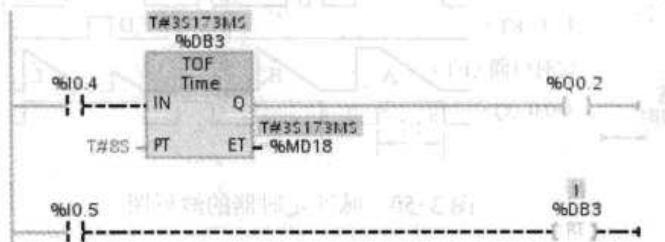


图 3-53 断开延时定时器

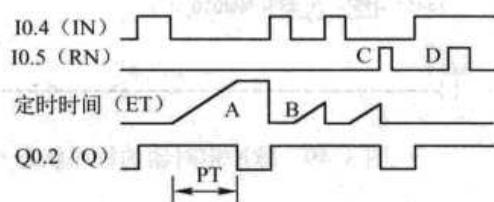


图 3-54 断开延时定时器的波形图

如果定时时间未达到 PT 设定的值, IN 输入就变为 1 状态, 输出 Q 将保持 1 状态不变(见波形 B)。

图 3-53 的 I0.5 为 1 时, 定时器复位线圈 RT 通电。如果 IN 输入为 0 状态, 则定时器被复位, 定时时间被清零, 输出 Q 变为 0 状态(见波形 C)。如果复位时 IN 输入为 1 状态, 则复位信号不起作用(见波形 D)。

#### 5. 保持型接通延时定时器

保持型接通延时定时器(TONR, 见图 3-55)的 IN 输入电路接通时开始定时(见图 3-56 中的波形 A 和 B)。当前值大于等于 PT 指定的设定值时, 输出 Q 变为 1 状态。

输入电路断开时, 当前值保持不变。可以用 TONR 来累计输入电路接通的若干个时间间隔。图 3-56 中的时间间隔  $t_1+t_2 \geq 9s$  时, 定时器的输出 Q 变为 1 状态(见波形 D)。

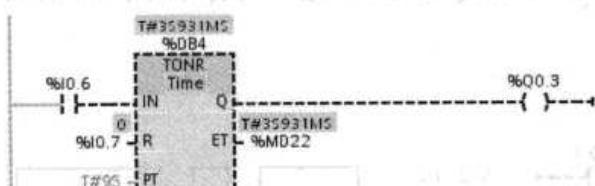


图 3-55 保持型接通延时定时器

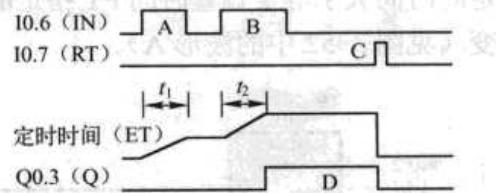


图 3-56 保持型接通延时定时器的波形图

复位输入 I0.7 为 1 状态时(见波形 C), TONR 被复位, 它的当前值变为 0, 同时输出 Q 变为 0 状态。

**【例 3-2】** 用接通延时定时器设计周期和占空比可调的振荡电路。

图 3-57 中 I1.1 的常开触点接通后, 左边的定时器的 IN 输入为 1 状态, 开始定时(见图 3-57 上面的图)。

2s 后定时时间到, 它的 Q 输出端的能流流入右边的定时器的 IN 输入端(见图 3-57 下面的图), 使右边的定时器开始定时, 同时 Q0.7 的线圈通电。

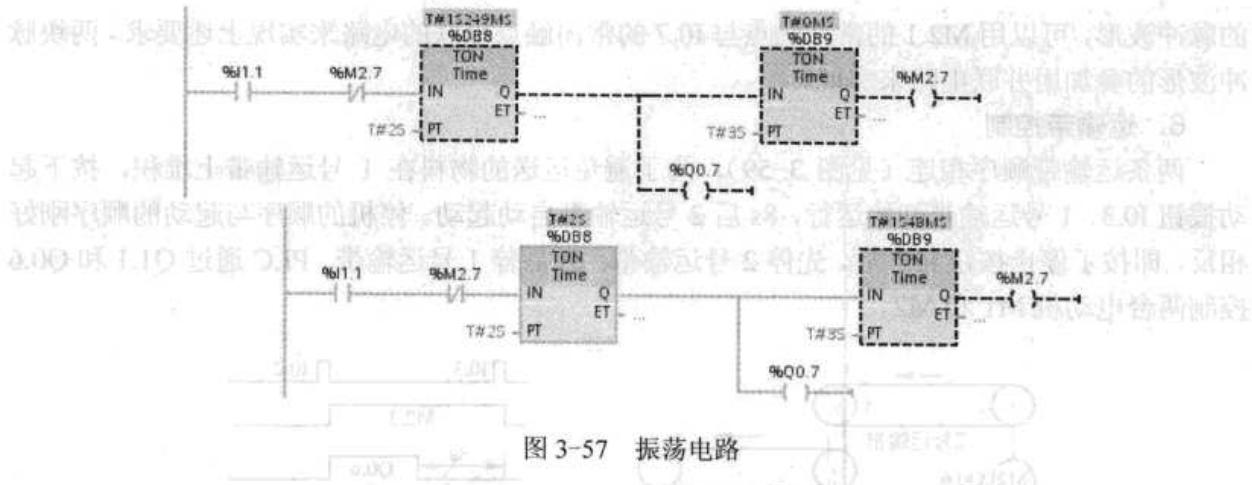


图 3-57 振荡电路

3s 后右边的定时器的定时时间到，它的输出 Q 变为 1 状态，使 M2.7 的常闭触点断开，左边的定时器的 IN 输入电路断开，其 Q 输出变为 0 状态，使 Q0.7 和右边的定时器的 Q 输出也变为 0 状态。下一个扫描周期因为 M2.7 的常闭触点接通，左边的定时器又从预置值开始定时，以后 Q0.7 的线圈将这样周期性地通电和断电，直到 I1.1 变为 0 状态。Q0.7 线圈通电和断电的时间分别等于右边和左边的定时器的预置值。振荡电路实际上是一个有正反馈的电路，两个定时器的输出 Q 分别控制对方的输入 IN，形成了正反馈。

CPU 的时钟存储器字节（见图 2-32）的各位提供周期为 0.1~2s 的时钟脉冲，它们输出高电平和低电平时间相等的方波信号，可以用它们的触点来控制需要闪烁的指示灯。

### 【例 3-3】用 3 种定时器设计卫生间冲水控制电路。

图 3-58 是卫生间冲水控制电路及其波形图。I0.7 是光电开关检测到的有使用者的信号，用 Q1.0 控制冲水电磁阀，图 3-58 的右边是 I0.7 和 Q1.0 的波形图。

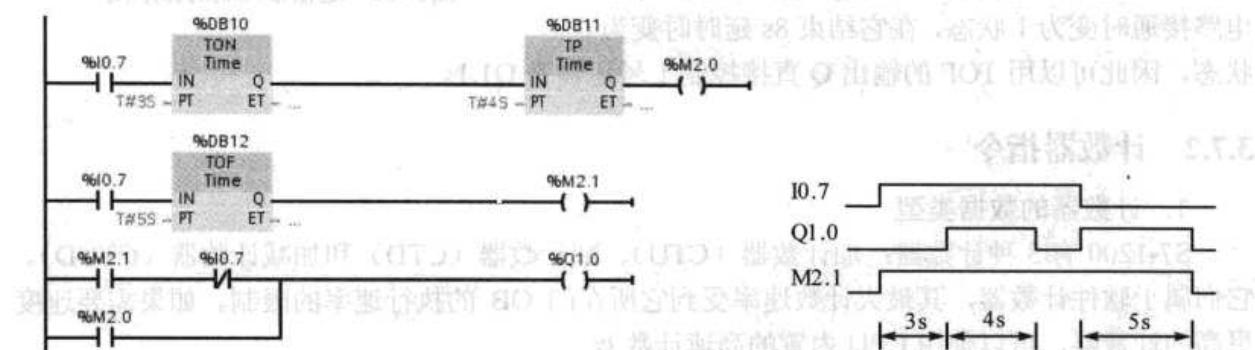


图 3-58 卫生间冲水控制电路与波形图

从 I0.7 的上升沿（有人使用）开始，用接通延时定时器 TON 延时 3s，3s 后 TON 的输出 Q 变为 1 状态，使脉冲定时器 TP 的 IN 输入变为 1 状态，TP 的 Q 输出端通过 M2.0 输出一个宽度为 4s 的脉冲。

从 I0.7 的上升沿开始，断开延时定时器 TOF 的 Q 输出控制的 M2.1 变为 1 状态。使用者离开时（在 I0.7 的下降沿），TOF 开始定时，5s 后 M2.1 变为 0 状态。

由波形图可知，控制冲水电磁阀的 Q1.0 输出的高电平脉冲波形由两块组成，4s 的脉冲波形由 TP 的 Q 输出控制的 M2.0 提供。TOF 控制的 M2.1 的波形减去 I0.7 的波形得到宽度为 5s

的脉冲波形，可以用 M2.1 的常开触点与 I0.7 的常闭触点串联的电路来实现上述要求。两块脉冲波形的叠加用并联电路来实现。

## 6. 运输带控制

两条运输带顺序相连（见图 3-59），为了避免运送的物料在 1 号运输带上堆积，按下起动按钮 I0.3，1 号运输带开始运行，8s 后 2 号运输带自动起动。停机的顺序与起动的顺序刚好相反，即按了停止按钮 I0.2 后，先停 2 号运输带，8s 后停 1 号运输带。PLC 通过 Q1.1 和 Q0.6 控制两台电动机 M1 和 M2。

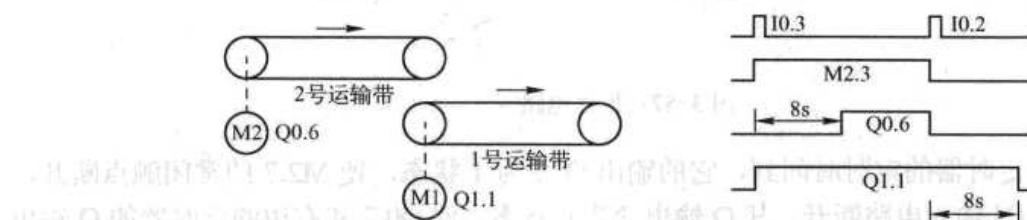


图 3-59 运输带示意图与波形图

梯形图程序如图 3-60 所示，程序中设置了一个用起动按钮和停止按钮控制的辅助元件 M2.3，用它来控制接通延时定时器 TON 和断开延时定时器 TOF 的 IN 输入端。TON 的 Q 输出端控制的 Q0.6 在 I0.3 的上升沿之后 8s 变为 1 状态，在 M2.3 的线圈断电（M2.3 的下降沿）时变为 0 状态。综上所述，可以用 TON 的 Q 输出端直接控制 2 号运输带 Q0.6。

断开延时定时器 TOF 的输出 Q 在它的 IN 输入电路接通时变为 1 状态，在它结束 8s 延时时变为 0 状态，因此可以用 TOF 的输出 Q 直接控制 1 号运输带 Q1.1。

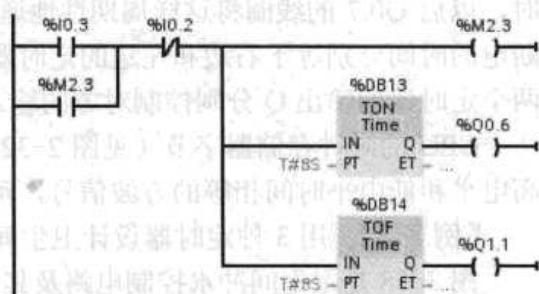


图 3-60 运输带控制的梯形图

## 3.7.2 计数器指令

### 1. 计数器的数据类型

S7-1200 有 3 种计数器：加计数器（CTU）、减计数器（CTD）和加减计数器（CTUD）。它们属于软件计数器，其最大计数速率受到它所在的 OB 的执行速率的限制。如果需要速度更高的计数器，可以使用 CPU 内置的高速计数器。

调用计数器指令时，需要生成保存计数器数据的背景数据块。

CU 和 CD 分别是加计数输入和减计数输入，在 CU 或 CD 由 0 状态变为 1 状态（信号的上升沿），计数当前值 CV 被加 1 或减 1。

复位输入 R 为 1 状态时，计数器被复位，计数当前值 CV 被清 0，计数器的输出 Q 变为 0 状态。CU、CD、R 和 Q 均为 Bool 变量。

将指令列表的“Counter”文件夹中的 CTU 指令拖放到工作区后，点击方框中 CTU 下面的 3 个问号（见图 3-61 左图），再点击问号右边出现的 ▾ 按钮，用下拉式列表

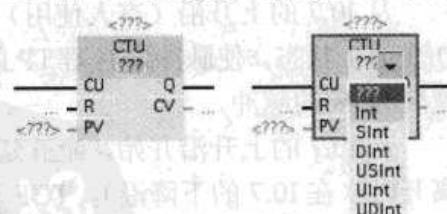


图 3-61 设置计数的数据类型

选择 PV 和 CV 的数据类型。

PV 为预置的计数值，CV 为实际的计数值，它们可以使用的数据类型见图 3-61。各变量均可以使用 I（仅用于输入变量）、Q、M、D 和 L 存储区。

## 2. 加计数器

当接在 R 输入端的复位输入 I1.1 为 0 状态（见图 3-62），接在 CU 输入端的加计数脉冲输入电路由断开变为接通时（即在 CU 信号的上升沿），计数当前值 CV 加 1，直到 CV 达到指定的数据类型的上限值。达到上限值后，CU 输入的状态变化不再起作用，CV 的值不再增加。

当前值大于等于设定值 PV 时，输出 Q 为 1 状态，反之为 0 状态。第一次执行指令时，当前值 CV 被清零。

当各类计数器的复位输入 R 为 1 状态时，计数器被复位，输出 Q 变为 0 状态，当前值被清零。图 3-63 是加计数器的波形图。

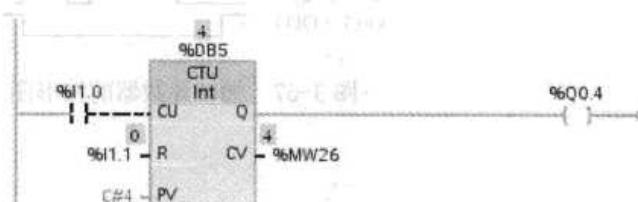


图 3-62 加计数器

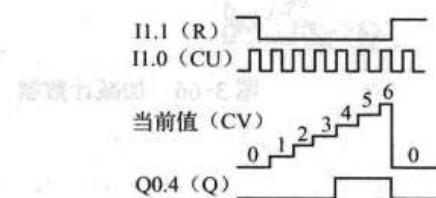


图 3-63 加计数器的波形图

## 3. 减计数器

装载输入 LOAD 为 1 状态时，输出 Q 被复位为 0，并把设定值 PV 的值装入当前值 CV。在减计数脉冲 CD 的上升沿（从 OFF 到 ON），计数当前值 CV 减 1，直到 CV 达到指定的数据类型的下限值。达到下限值时，CD 输入的状态变化不再起作用，CV 的值不再减少。

当前值 CV 小于等于 0 时，输出 Q 为 1 状态（见图 3-64），反之 Q 为 0 状态。第一次执行指令时，当前值 CV 被清零。图 3-65 是减计数器的波形图。

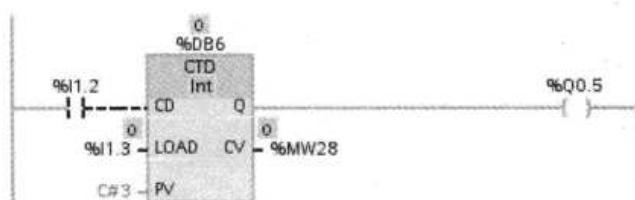


图 3-64 减计数器

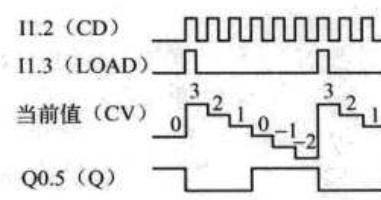


图 3-65 减计数器的波形图

## 4. 加减计数器

在加计数脉冲 CU 的上升沿，计数器的当前值 CV 加 1，直到 CV 达到指定的数据类型的上限值。达到上限值时，CV 的值不再增加。

在减计数脉冲 CD 的上升沿，计数器的当前值 CV 减 1，直到 CV 达到指定的数据类型的下限值。达到下限值时，CV 的值不再减小。

如果同时出现计数脉冲 CU 和 CD 的上升沿，当前值 CV 保持不变。

当前值 CV 大于等于设定值 PV 时，输出 QU 为 1（见图 3-66），反之为 0。CV 小于等于

0时，输出QD为1，反之为0。

装载输入LOAD为1状态时，PV设置的值被装入当前值CV，输出QU变为1状态，QD被复位为0状态。

复位输入R为1状态时，计数器被复位。当前值CV被清零，输出QU变为0状态，QD变为1状态。

R为1状态时，CU、CD和LOAD不再起作用。图3-67是加减计数器的波形图。

图3-66展示了加减计数器的梯形图逻辑，图3-67展示了其对应的波形图。

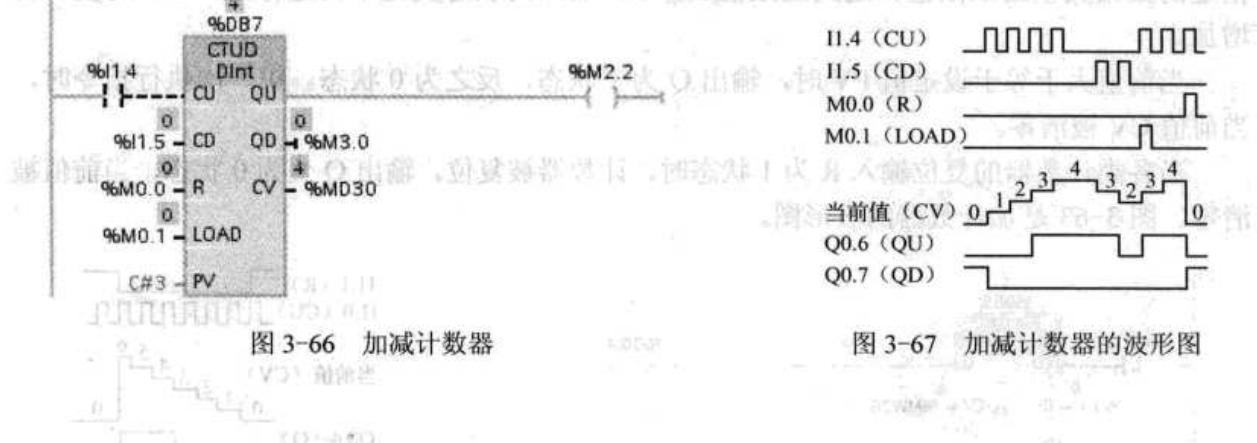


图3-66 加减计数器

图3-67 加减计数器的波形图

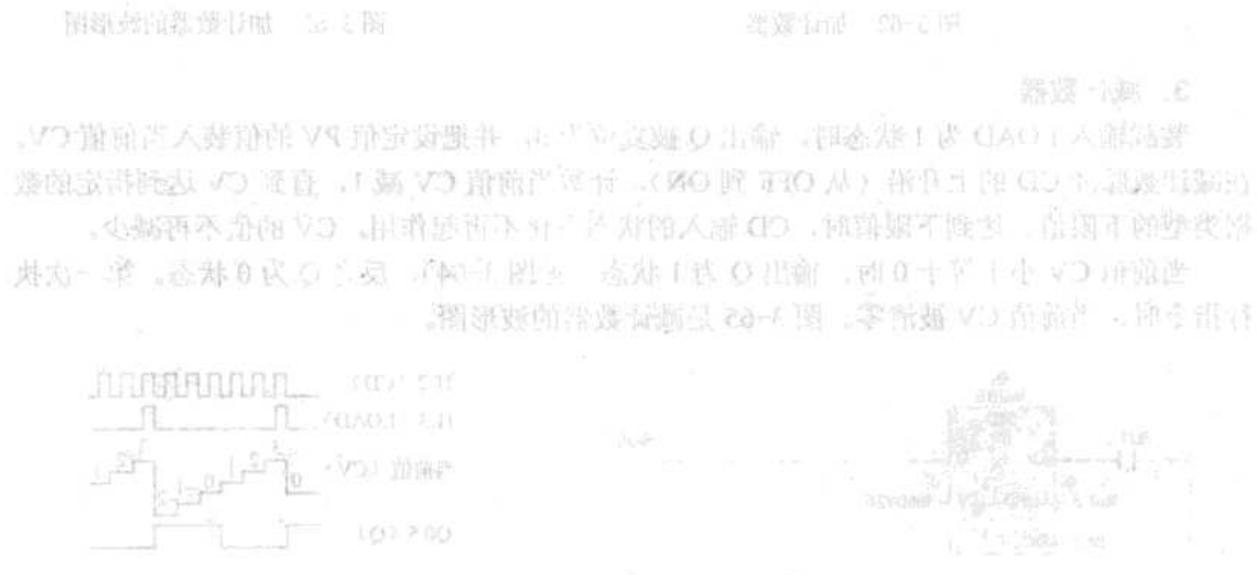


图3-67 加减计数器的波形图

图3-68展示了加减计数器的梯形图逻辑，图3-69展示了其对应的波形图。

## 第4章 数字量控制系统梯形图程序设计方法

### 4.1 梯形图中的基本电路与经验设计法

开关量控制系统（例如继电器控制系统）又称为数字量控制系统。下面首先介绍数字量控制系统的经验设计法常用的一些基本电路。

#### 4.1.1 梯形图中的基本电路

##### 1. 起保停电路与置位复位电路

第3章已经介绍过起动-保持-停止电路（简称为起保停电路），由于该电路在梯形图中的应用很广，现在将它重画在图4-1中。左图中的起动信号I0.0和停止信号I0.1（例如起动按钮和停止按钮提供的信号）持续为1状态的时间一般都很短。起保停电路最主要的特点是具有“记忆”功能，按下起动按钮，I0.0的常开触点接通，Q0.0的线圈“通电”，它的常开触点同时接通。放开起动按钮，I0.0的常开触点断开，“能流”经Q0.0的常开触点和I0.1的常闭触点流过Q0.0的线圈，Q0.0仍为1状态，这就是所谓的“自锁”或“自保持”功能。按下停止按钮，I0.1的常闭触点断开，使Q0.0的线圈“断电”，其常开触点断开。以后即使放开停止按钮，I0.1的常闭触点恢复接通状态，Q0.0的线圈仍然“断电”。这种记忆功能也可以用图4-1中的S指令和R指令来实现。起保停电路与置位复位电路是后面要重点介绍的顺序控制设计法的基本电路。

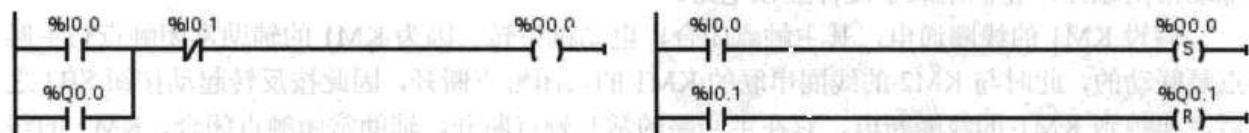


图4-1 起保停电路与置位复位电路

在实际电路中，起动信号和停止信号可能由多个触点组成的串、并联电路提供。

##### 2. 三相异步电动机的正反转控制电路

图4-2是三相异步电动机正反转控制的主电路和继电器控制电路图，KM1和KM2分别是控制正转运行和反转运行的交流接触器。用KM1和KM2的主触点改变进入电动机的三相电源的相序，就可以改变电动机的旋转方向。图中的FR是热继电器，在电动机过载时，它的常闭触点断开，使KM1或KM2的线圈断电，电动机停转。

图4-2中的控制电路由两个起保停电路组成，为了节省触点，FR和SB1的常闭触点供两个起保停电路公用。

按下正转起动按钮SB2，KM1的线圈通电并自保持，电动机正转运行。按下反转起动按钮SB3，KM2的线圈通电并自保持，电动机反转运行。按下停止按钮SB1，KM1或KM2的线圈断电，电动机停止运行。

为了方便操作和保证 KM1 和 KM2 不会同时动作，在图 4-2 中设置了“按钮联锁”，将正转起动按钮 SB2 的常闭触点与控制反转的 KM2 的线圈串联，将反转起动按钮 SB3 的常闭触点与控制正转的 KM1 的线圈串联。设 KM1 的线圈通电，电动机正转，这时如果想改为反转，可以不按停止按钮 SB1，直接按反转起动按钮 SB3，它的常闭触点断开，使 KM1 的线圈断电，同时 SB3 的常开触点接通，使 KM2 的线圈得电，电动机由正转变为反转。

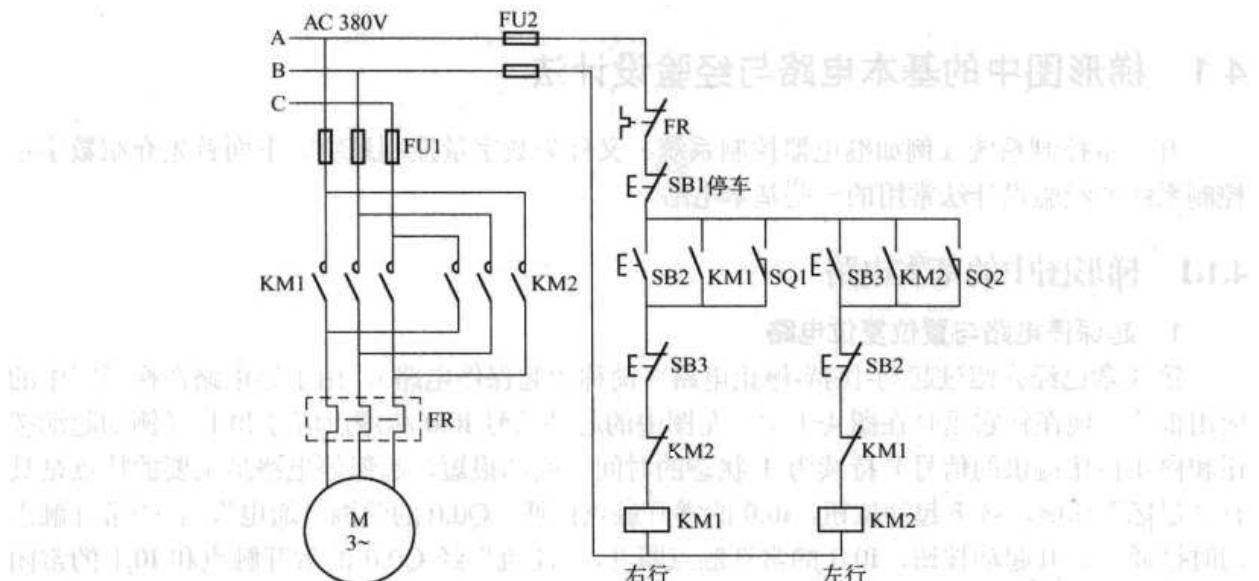


图 4-2 异步电动机正反转继电器控制电路

由主回路可知，如果 KM1 和 KM2 的主触点同时闭合，将会造成三相电源相间短路的故障。在二次回路中，KM1 的线圈串联了 KM2 的辅助常闭触点，KM2 的线圈串联了 KM1 的辅助常闭触点，它们组成了硬件互锁电路。

假设 KM1 的线圈通电，其主触点闭合，电动机正转。因为 KM1 的辅助常闭触点与主触点是联动的，此时与 KM2 的线圈串联的 KM1 的常闭触点断开，因此按反转起动按钮 SB3 之后，要等到 KM1 的线圈断电，它在主回路的常开触点断开，辅助常闭触点闭合，KM2 的线圈才会通电，因此这种互锁电路可以有效地防止电源短路故障。

图 4-3 和图 4-4 是实现上述功能的 PLC 的外部接线图和梯形图。将继电器电路图转换为梯形图时，首先应确定 PLC 的输入信号和输出信号。3 个按钮提供操作人员发出的指令信号，按钮信号必须输入到 PLC 中去，热继电器的常开触点提供了 PLC 的另一个输入信号。显然，两个交流接触器的线圈是 PLC 输出端的负载。

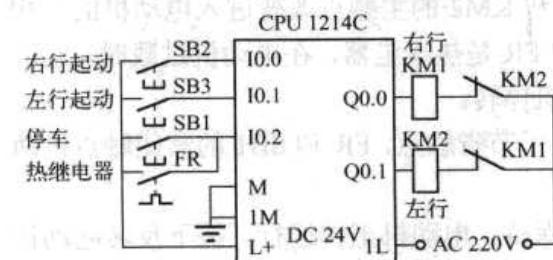


图 4-3 PLC 的外部接线图

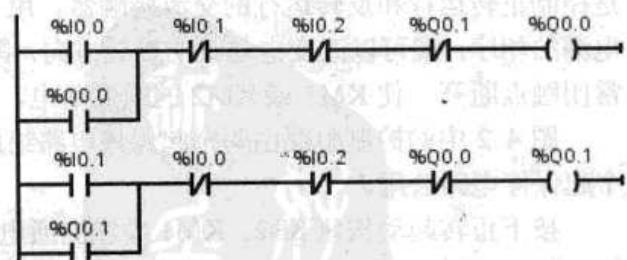


图 4-4 梯形图

画出 PLC 的外部接线图后，同时也确定了外部输入/输出信号与 PLC 内的过程映像输入/输出位的地址之间的关系。可以将继电器电路图“翻译”为梯形图，即采用与图 4-2 中的继电器电路完全相同的结构来画梯形图。各触点的常开、常闭的性质不变，根据 PLC 外部接线图中给出的关系，来确定梯形图中各触点的地址。图 4-2 中 FB1 和 FR 的常闭触点串联电路对应于图 4-4 中 I0.2 的常闭触点。

图 4-4 中的梯形图将控制 Q0.0 和 Q0.1 的两个起停电路分离开来，电路的逻辑关系比较清晰。虽然多用了一个 I0.2 的常闭触点，但是并不会增加硬件成本。

图 4-4 使用了 Q0.0 和 Q0.1 的常闭触点组成的软件互锁电路。如果没有图 4-3 的硬件互锁电路，从正转马上切换到反转时，由于切换过程中电感的延时作用，可能会出现原来接通的接触器的主触点还未断弧，另一个接触器的主触点已经合上的现象，从而造成交流电源瞬间短路的故障。

此外，如果没有硬件互锁电路，且因为主电路电流过大或接触器质量不好，某一接触器的主触点被断电时产生的电弧熔焊而被粘结，其线圈断电后主触点仍然是接通的，这时如果另一个接触器的线圈通电，也会造成三相电源短路事故。为了防止出现这种情况，应在 PLC 外部设置由 KM1 和 KM2 的辅助常闭触点组成的硬件互锁电路（见图 4-3）。这种互锁与图 4-2 的继电器电路的互锁原理相同，假设 KM1 的主触点被电弧熔焊，这时它与 KM2 线圈串联的辅助常闭触点处于断开状态，因此 KM2 的线圈不可能得电。

## 4.1.2 梯形图的经验设计法

### 1. 小车自动往返控制程序的设计

可以用设计继电器电路图的方法来设计比较简单的数字量控制系统的梯形图，即在一些典型电路的基础上，根据被控对象对控制系统的具体要求，不断地修改和完善梯形图。有时需要多次反复地调试和修改梯形图，增加一些中间编程元件和触点，最后才能得到一个较为满意的结果。

这种方法没有普遍的规律可以遵循，具有很大的试探性和随意性，最后的结果不是唯一的，设计所用的时间、设计的质量与设计者的经验有很大的关系，所以有人把这种设计方法叫做经验设计法，它可以用于较简单的梯形图（例如手动程序）的设计。

异步电动机的主回路与图 4-2 中的相同。在图 4-3 的基础上，增加了接在 I0.3 和 I0.4 输入端子的左、右限位开关的常开触点（见图 4-5）。

按下右行起动按钮 SB2 或左行起动按钮 SB3 后，要求小车在左限位开关 SQ1 和右限位开关 SQ2 之间不停地循环往返，按下停止按钮 SB1 后，电动机断电，小车停止运动。

可以在三相异步电动机正反转继电器控制电路的基础上，设计出满足要求的梯形图（见图 4-6）。

为了使小车的运动在极限位置自动停止，将右限位开关 I0.4 的常闭触点与控制右行的 Q0.0 的线圈串联，将左限位开关 I0.3 的常闭触点与控制左行的 Q0.1 的线圈串联。为了使小车自动改变运动方向，将左限位开关 I0.3 的常开触点与手动起动右行的 I0.0 的常开触点并联，将右限位开关 I0.4 的常开触点与手动起动左行的 I0.1 的常开触点并联。

假设按下左行起动按钮 I0.1，Q0.1 变为 1 状态，小车开始左行，碰到左限位开关时，I0.3 的常闭触点断开，使 Q0.1 的线圈“断电”，小车停止左行。I0.3 的常开触点接通，使 Q0.0 的

线圈“通电”，开始右行。碰到右限位开关时，I0.4 的常闭触点断开，使 Q0.0 的线圈“断电”，小车停止右行。I0.4 的常开触点接通，使 Q0.1 的线圈“通电”，又开始左行。以后将这样不断地往返运动下去，直到按下停止按钮 I0.2。

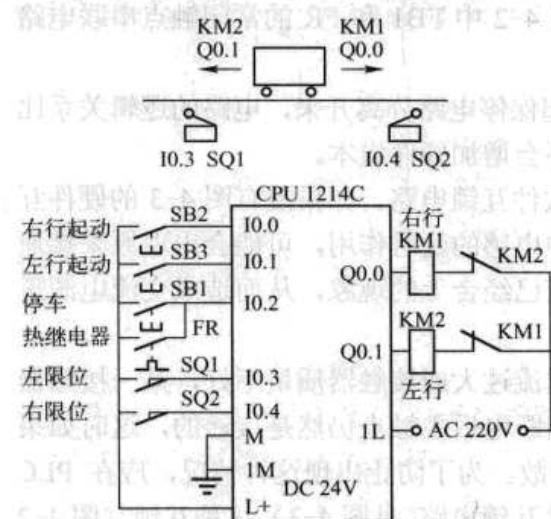


图 4-5 外部接线图

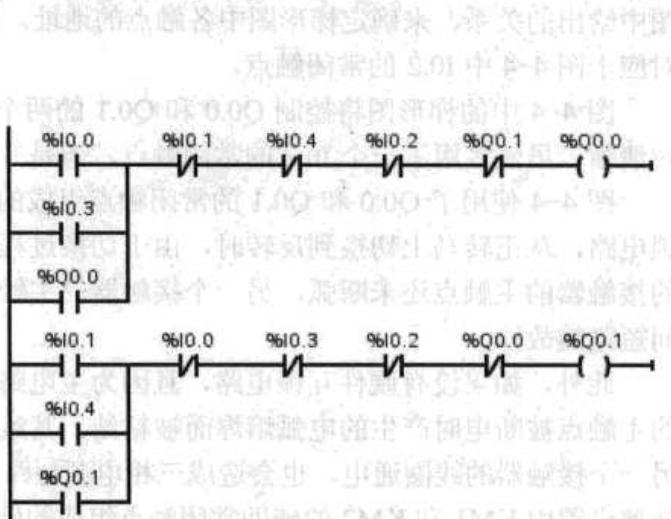


图 4-6 小车自动往返的梯形图

这种控制方法适用于小容量的异步电动机，且往返不能太频繁，否则电动机将会过热。

## 2. 较复杂的小车自动运行控制程序的设计

PLC 的外部接线图与图 4-5 相同。小车开始时停在左边，左限位开关 SQ1 的常开触点闭合。要求按下列顺序控制小车：

- 1) 按下右行起动按钮，小车开始右行。
- 2) 走到右限位开关处，小车停止运动，延时 8s 后开始左行。
- 3) 回到左限位开关处，小车停止运动。

在异步电动机正反转控制电路的基础上设计的满足上述要求的梯形图如图 4-7 所示。

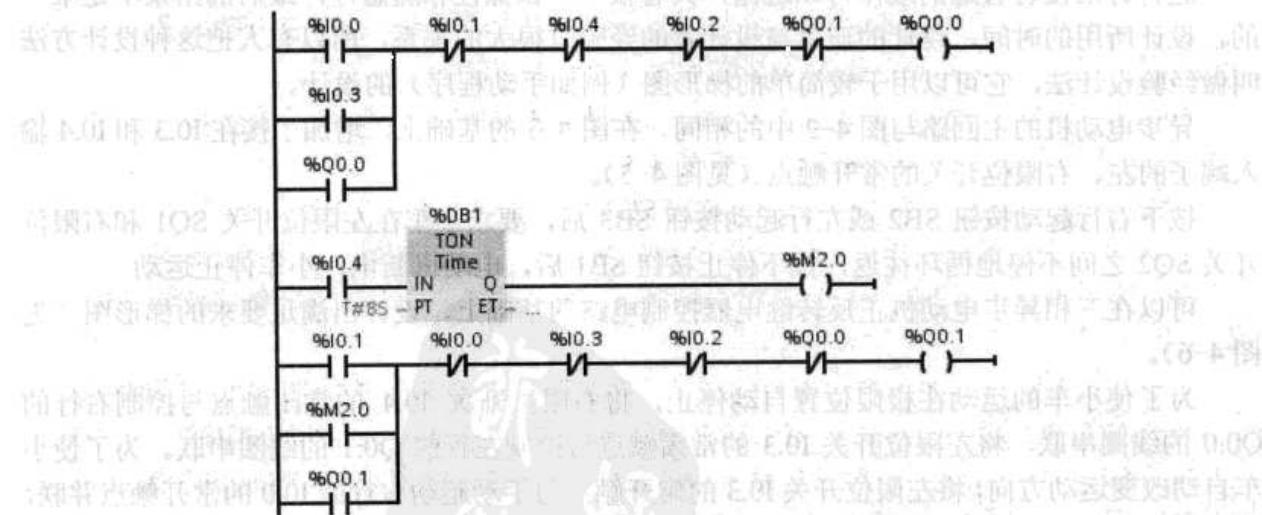


图 4-7 梯形图

在控制右行的 Q0.0 的线圈回路中串联了 I0.4 的常闭触点，小车走到右限位开关 SQ2 处

时, I0.4 的常闭触点断开, 使 Q0.0 的线圈断电, 小车停止右行。同时 I0.4 的常开触点闭合, 定时器 TON 的 IN 输入为 1 状态, 开始定时。8s 后定时时间到, 用 T0 的 Q 输出端控制的 M2.0 的常开触点闭合, 使 Q0.1 的线圈通电并自保持, 小车开始左行。离开限位开关 SQ2 后, I0.4 的常开触点断开, 定时器 TON 因为其 IN 输入变为 0 状态而被复位。小车运行到左边的起始点时, 左限位开关 SQ1 的常开触点闭合, I0.3 的常闭触点断开, 使 Q0.1 的线圈断电, 小车停止运动。

在梯形图中, 保留了左行起动按钮 I0.1 和停止按钮 I0.2 的触点, 使系统有手动操作的功能。串联在起停电路中的左限位开关 I0.3 和右限位开关 I0.4 的常闭触点在手动时可以防止小车的运动超限。

## 4.2 顺序控制设计法与顺序功能图

用经验设计法设计梯形图时, 没有一套固定的方法和步骤可以遵循, 具有很大的试探性和随意性, 对于不同的控制系统, 没有一种通用的容易掌握的设计方法。在设计复杂系统的梯形图时, 用大量的中间单元来完成记忆和互锁等功能, 由于需要考虑的因素很多, 它们往往又交织在一起, 分析起来非常困难, 并且很容易遗漏一些应该考虑的问题。修改某一局部电路时, 很可能会“牵一发而动全身”, 对系统的其他部分产生意想不到的影响, 因此梯形图的修改也很麻烦, 往往花了很长的时间还得不到一个满意的结果。用经验法设计出的复杂的梯形图很难阅读, 给系统的维修和改进带来了很大的困难。

所谓顺序控制, 就是按照生产工艺预先规定的顺序, 在各个输入信号的作用下, 根据内部状态和时间的顺序, 在生产过程中各个执行机构自动地有秩序地进行操作。

顺序功能图 (Sequential Function Chart, SFC) 是描述控制系统的控制过程、功能和特性的一种图形, 也是设计 PLC 的顺序控制程序的有力工具。

顺序功能图并不涉及所描述的控制功能的具体技术, 它是一种通用的技术语言, 可以供进一步设计和不同专业的人员之间进行技术交流之用。

顺序功能图是 IEC 61131-3 位居首位的编程语言, 有的 PLC 为用户提供了顺序功能图语言, 例如 S7-300/400 的 S7 Graph 语言, 在编程软件中生成顺序功能图后便完成了编程工作。

现在还有相当多的 PLC (包括 S7-1200) 没有配备顺序功能图语言。但是可以用顺序功能图来描述系统的功能, 根据它来设计梯形图程序。

顺序控制设计法是一种先进的设计方法, 很容易被初学者接受, 对于有经验的工程师, 也会提高设计的效率, 程序的调试、修改和阅读也很方便。

### 4.2.1 步与动作

#### 1. 步的基本概念

顺序控制设计法最基本的思想是将系统的一个工作周期划分为若干个顺序相连的阶段, 这些阶段称为步 (Step), 并用编程元件 (例如位存储器 M) 来代表各步。步是根据输出量的状态变化来划分的, 在任何一步之内, 各输出量的 ON/OFF 状态不变, 但是相邻两步输出量总的状态是不同的, 步的这种划分方法使代表各步的编程元件的状态与各输出量的状态之间

有着极为简单的逻辑关系。

顺序控制设计法用转换条件控制代表各步的编程元件，让它们的状态按一定的顺序变化，然后用代表各步的编程元件去控制PLC的各输出位。

图4-8中的小车开始时停在最左边，限位开关I0.2为1状态。按下起动按钮，Q0.0变为1状态，小车右行。碰到右限位开关I0.1时，Q0.0变为0状态，Q0.1变为1状态，小车改为左行。返回起始位置时，Q0.1变为0状态，小车停止运行，同时Q0.2变为1状态，使制动电磁铁线圈通电，接通延时定时器开始定时。定时时间到，制动电磁铁线圈断电，系统返回初始状态。

根据Q0.0~Q0.2的ON/OFF状态的变化，显然可以将上述工作过程分为3步，分别用M4.1~M4.3来代表这3步，另外还设置了一个等待起动的初始步。图4-9是描述该系统的顺序功能图，图中用矩形方框表示步，方框中可以用数字表示该步的编号，也可以用代表该步的编程元件的地址作为步的编号，例如M4.0等。

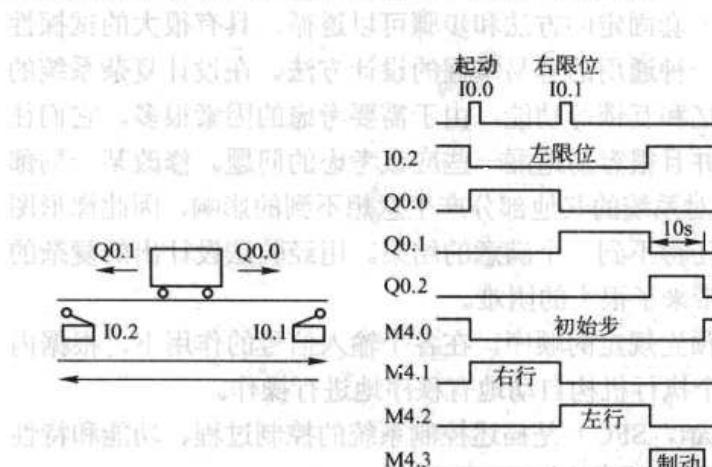


图4-8 系统示意图与波形图

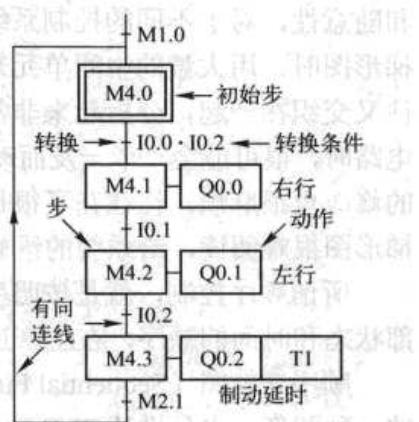


图4-9 顺序功能图

为了便于将顺序功能图转换为梯形图，用代表各步的编程元件的地址作为步的代号，并用编程元件的地址来标注转换条件和各步的动作或命令。

## 2. 初始步

与系统的初始状态相对应的步称为初始步，初始状态一般是系统等待起动命令的相对静止的状态。初始步用双线方框表示，每一个顺序功能图至少应该有一个初始步。

## 3. 活动步

当系统正处于某一步所在的阶段时，该步处于活动状态，称该步为“活动步”。步处于活动状态时，执行相应的非存储型动作；处于不活动状态时，则停止执行。

## 4. 与步对应的动作或命令

可以将一个控制系统划分为被控系统和施控系统，例如在数控车床系统中，数控装置是施控系统，而车床是被控系统。对于被控系统，在某一步中要完成某些“动作”(Action)，对于施控系统，在某一步中则要向被控系统发出某些“命令”(Command)。为了叙述方便，下面将命令或动作统称为动作，并用矩形框中的文字或变量表示动作，该矩形框应与它所在的步对应的方框相连。

如果某一步有几个动作，可以用图 4-10 中的两种画法来表示，但是并不隐含这些动作之间的任何顺序。应清楚地表明动作是存储型的还是非存储型的。图 4-9 中的 Q0.0~Q0.2 均为非存储型动作，例如在步 M4.1 为活动步时，动作 Q0.0 为 1 状态，步 M4.1 为不活动步时，动作 Q0.0 为 0 状态。步与它的非存储性动作是完全一致的。

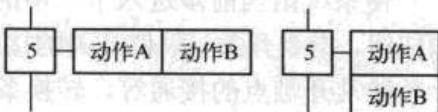


图 4-10 动作

某些动作在连续的若干步都应为 1 状态，可以在顺序功能图中，用动作的修饰词“S”（见表 4-1）将它在应为 1 状态的第一步置位，用动作的修饰词“R”将它在应为 1 状态的最后一步的下一步复位为 0 状态。这种动作是存储性动作，在程序中用置位、复位指令来实现。在图 4-9 中，定时器 T1 的 IN 输入在步 M4.3 为活动步时为 1 状态，步 M4.3 为不活动步时为 0 状态，从这个意义上来说，T1 的 IN 输入相当于步 M4.3 的一个非存储型动作，所以将 T1 放在步 M4.3 的动作框内。

使用动作的修饰词（见表 4-1），可以在一步中完成不同的动作。修饰词允许在不增加逻辑的情况下控制动作。例如，可以使用修饰词 L 来限制配料阀打开的时间。

表 4-1 动作的修饰词

N	非存储型	当步变为不活动步时动作终止
S	置位（存储）	当步变为不活动步时动作继续，直到动作被复位
R	复位	被修饰词 S、SD、SL 或 DS 起动的动作被终止
L	时间限制	步变为活动步时动作被起动，直到步变为不活动步或设定时间到
D	时间延迟	步变为活动步时延迟定时器被起动，如果延迟之后步仍然是活动的，动作被起动和继续，直到步变为不活动步
P	脉冲	当步变为活动步，动作被起动并且只执行一次
SD	存储与时间延迟	在时间延迟之后动作被起动，一直到动作被复位
DS	延迟与存储	在延迟之后如果步仍然是活动的，动作被起动直到被复位
SL	存储与时间限制	步变为活动步时动作被起动，一直到设定的时间到或动作被复位

## 4.2.2 有向连线与转换条件

### 1. 有向连线

在顺序功能图中，随着时间的推移和转换条件的实现，将会发生步的活动状态的进展，这种进展按有向连线规定的路线和方向进行。在画顺序功能图时，将代表各步的方框按它们成为活动步的先后次序顺序排列，并用有向连线将它们连接起来。步的活动状态习惯的进展方向是从上到下或从左至右，在这两个方向有向连线上的箭头可以省略。如果不是上述的方向，则应在有向连线上用箭头注明进展方向。为了更易于理解，在可以省略箭头的有向连线上也可以加箭头。

如果在画图时有向连线必须中断（例如在复杂的图中，或用几个图来表示一个顺序功能图时），应在有向连线中断之处标明下一步的标号和所在的页数，例如“步 83、12 页”。

### 2. 转换

转换用有向连线上与有向连线垂直的短划线来表示，转换将相邻两步分隔开。步的活动状态的进展是由转换的实现来完成的，并与控制过程的发展相对应。

### 3. 转换条件

使系统由当前步进入下一步的信号称为转换条件，转换条件可以是外部的输入信号，例如按钮、指令开关、限位开关的接通或断开等；也可以是 PLC 内部产生的信号，例如定时器、计数器常开触点的接通等，转换条件还可以是若干个信号的与、或、非逻辑组合。

转换条件可以用文字语言、布尔代数表达式或图形符号标注在表示转换的短线旁，使用得最多的是布尔代数表达式（见图 4-11）。

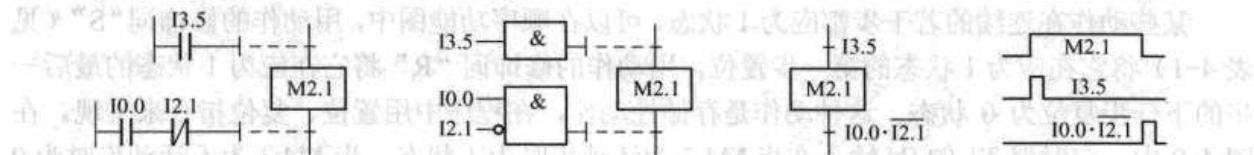


图 4-11 转换与转换条件

转换条件  $I_{10.0}$  和  $\overline{I_{10.0}}$  分别表示当输入信号  $I_{10.0}$  为 1 状态和 0 状态时转换实现。符号  $\uparrow I_{10.0}$  和  $\downarrow I_{10.0}$  分别表示当  $I_{10.0}$  从 0 状态到 1 状态和从 1 状态到 0 状态时转换实现。实际上即使不加符号“ $\uparrow$ ”，转换一般也是在  $I_{10.0}$  的上升沿实现的，因此一般不加“ $\uparrow$ ”。

图 4-11 用高电平表示步  $M_{2.1}$  为活动步，反之则用低电平表示。转换条件  $I_{10.0} \cdot \overline{I_{12.1}}$  表示  $I_{10.0}$  的常开触点与  $I_{12.1}$  的常闭触点同时闭合，在梯形图中则用两个触点的串联来表示这样一个“与”逻辑关系。

图 4-9 中步  $M_{4.3}$  下面的转换条件  $M_{2.1}$  对应于定时器  $T_1$  的 Q 输出信号， $T_1$  的定时时间到时，转换条件满足。

在顺序功能图中，只有当某一步的前级步是活动步时，该步才有可能变成活动步。如果用没有断电保持功能的编程元件来代表各步，进入 RUN 工作方式时，它们均处于 0 状态。

在对 CPU 组态时如果设置默认的 MB1 为系统存储器字节（见图 2-32），则必须用开机时接通一个扫描周期的  $M_{1.0}$  的常开触点作为转换条件，将初始步预置为活动步（见图 4-9），否则因为顺序功能图中没有活动步，系统将无法工作。如果系统有自动、手动两种工作方式，顺序功能图是用来描述自动工作过程的，这时还应在

系统由手动工作方式进入自动工作方式时，用一个适当的信号将初始步置为活动步（见 4.4 节）。

### 4.2.3 顺序功能图的基本结构

#### 1. 单序列

单序列由一系列相继激活的步组成，每一步的后面仅有一个转换，每一个转换的后面只有一个步（见图 4-12a），单序列的特点是没有下述的分支与合并。

#### 2. 选择序列

选择序列的开始称为分支（见图 4-12b），转换符号只能标在水平连线之下。如果步 5 是活动步，并且转换条件  $h$  为 1 状态，则发生由步 5 → 步 8 的进展。如果步 5 是活动步，并且  $k$  为 1 状态，则发生由步 5 → 步 10 的进展。如果将选择条件  $k \cdot \bar{h}$ ，

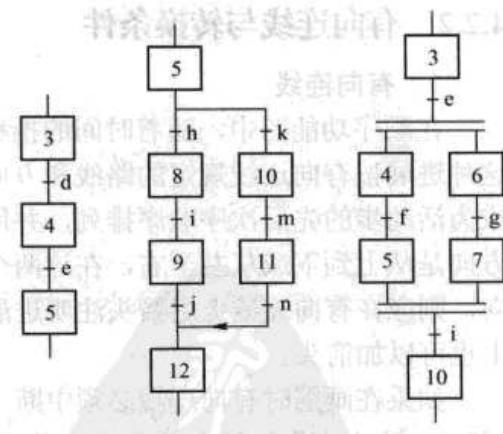


图 4-12 单序列、选择序列与并行序列

则当  $k$  和  $h$  同时为 1 状态时, 将优先选择  $h$  对应的序列, 一般只允许同时选择一个序列。选择序列的结束称为合并 (见图 4-12b), 几个选择序列合并到一个公共序列时, 用需要重新组合的序列相同数量的转换符号和水平连线来表示, 转换符号只允许标在水平连线之上。如果步 9 是活动步, 并且转换条件  $j$  为 1 状态, 则发生由步 9→步 12 的进展。如果步 11 是活动步, 并且  $n$  为 1 状态, 则发生由步 11→步 12 的进展。

### 3. 并行序列

并行序列的开始称为分支 (见图 4-12c), 当转换的实现导致几个序列同时激活时, 这些序列称为并行序列。当步 3 是活动的, 并且转换条件  $e$  为 1 状态, 步 4 和步 6 同时变为活动步, 同时步 3 变为不活动步。为了强调转换的同步实现, 水平连线用双线表示。步 4 和步 6 被同时激活后, 每个序列中活动步的进展将是独立的。在表示同步的水平双线之上, 只允许有一个转换符号。并行序列用来表示系统的几个同时工作的独立部分的工作情况。

并行序列的结束称为合并 (见图 4-12c), 在表示同步的水平双线之下, 只允许有一个转换符号。当直接连在双线上的所有前级步 (步 5 和步 7) 都处于活动状态, 并且转换条件  $i$  为 1 状态时, 才会发生步 5 和步 7 到步 10 的进展, 即步 5 和步 7 同时变为不活动步, 而步 10 变为活动步。

### 4. 复杂的顺序功能图举例

某专用钻床用来加工圆盘状零件上均匀分布的 6 个孔 (见图 4-13), 上面是侧视图, 下面是工件的俯视图。在进入自动运行之前, 两个钻头应在最上面, 上限位开关 I0.3 和 I0.5 为 1 状态, 系统处于初始步, 加计数器 C0 被复位, 计数当前值 CV 被清零。用存储器位 M 来代表各步, 顺序功能图中包含了选择序列和并行序列。操作人员放好工件后, 按下起动按钮 I0.0, 转换条件  $I0.0 \cdot I0.3 \cdot I0.5$  满足, 由初始步转换到步 M4.1, Q0.0 变为 1 状态, 工件被夹紧。夹紧后压力继电器 I0.1 为 1 状态, 由步 M4.1 转换到步 M4.2 和 M4.5, Q0.1 和 Q0.3 使两只钻头

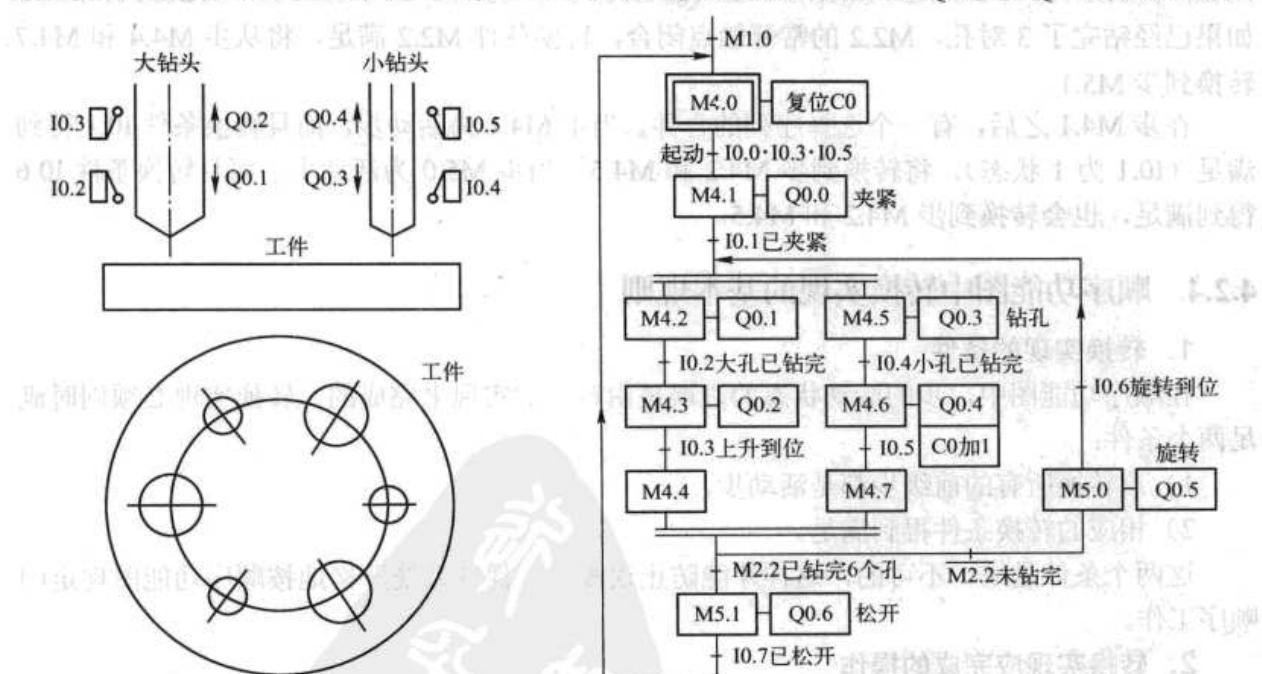


图 4-13 专用钻床控制系统的顺序功能图

同时开始向下钻孔。大钻头钻到由限位开关 I0.2 设定的深度时，进入步 M4.3，Q0.2 使大钻头上升，升到由限位开关 I0.3 设定的起始位置时停止上升，进入等待步 M4.4。小钻头钻到由限位开关 I0.4 设定的深度时，进入步 M4.6，Q0.4 使小钻头上升，设定值为 3 的加计数器 C0 的当前值加 1。升到由限位开关 I0.5 设定的起始位置时停止上升，进入等待步 M4.7。

C0 加 1 后的计数当前值为 1，C0 的 Q 输出端控制的 M2.2 的常闭触点闭合，转换条件 M2.2 满足。两个钻头都上升到位后，将转换到步 M5.0。Q0.5 使工件旋转  $120^\circ$ ，旋转到位时 I0.6 为 1 状态，又返回步 M4.2 和 M4.5，开始钻第二对孔。3 对孔都钻完后，计数器的当前值变为 3，其 Q 输出端控制的 M2.2 变为 1 状态，转换到步 M5.1，Q0.6 使工件松开。松开到位时，限位开关 I0.7 为 1 状态，系统返回初始步 M4.0。

因为要求两个钻头向下钻孔和钻头提升的过程同时进行，故采用并行序列来描述上述的过程。由 M4.2~M4.4 和 M4.5~M4.7 组成的两个单序列分别用来描述大钻头和小钻头的工作过程。在步 M4.1 之后，有一个并行序列的分支。当 M4.1 为活动步，并且转换条件 I0.1 得到满足（I0.1 为 1 状态），并行序列的两个单序列中的第 1 步（步 M4.2 和 M4.5）同时变为活动步。此后两个单序列内部各步的活动状态的转换是相互独立的，例如大孔或小孔钻完时的转换一般不是同步的。

两个单序列的最后 1 步（步 M4.4 和 M4.7）应同时变为不活动步。但是两个钻头一般不会同时上升到位，不可能同时结束运动，所以设置了等待步 M4.4 和 M4.7，它们用来同时结束两个并行序列。当两个钻头均上升到位，限位开关 I0.3 和 I0.5 分别为 1 状态，大、小钻头两个子系统分别进入两个等待步，并行序列将会立即结束。

在步 M4.4 和 M4.7 之后，有一个选择序列的分支。没有钻完 3 对孔时，M2.2 的常闭触点闭合，转换条件 M2.2 满足，如果两个钻头都上升到位，将从步 M4.4 和 M4.7 转换到步 M5.0。如果已经钻完了 3 对孔，M2.2 的常开触点闭合，转换条件 M2.2 满足，将从步 M4.4 和 M4.7 转换到步 M5.1。

在步 M4.1 之后，有一个选择序列的合并。当步 M4.1 为活动步，而且转换条件 I0.1 得到满足（I0.1 为 1 状态），将转换到步 M4.2 和 M4.5。当步 M5.0 为活动步，而且转换条件 I0.6 得到满足，也会转换到步 M4.2 和 M4.5。

#### 4.2.4 顺序功能图中转换实现的基本规则

##### 1. 转换实现的条件

在顺序功能图中，步的活动状态的进展是由转换的实现来完成的。转换实现必须同时满足两个条件：

- 1) 该转换所有的前级步都是活动步。
- 2) 相应的转换条件得到满足。

这两个条件是缺一不可的，这样才能防止误操作，保证系统严格地按顺序功能图规定的顺序工作。

##### 2. 转换实现应完成的操作

转换实现时应完成以下两个操作：

1) 使所有由有向连线与相应转换符号相连的后续步都变为活动步。

2) 使所有由有向连线与相应转换符号相连的前级步都变为不活动步。

以上规则可以用于任意结构中的转换，其区别如下：在单序列中，一个转换仅有一个前级步和一个后续步。在并行序列的分支处，转换有几个后续步（见图 4-12c），在转换实现时应同时将它们对应的编程元件置位。在并行序列的合并处，转换有几个前级步，它们均为活动步时才有可能实现转换，在转换实现时应将它们对应的编程元件全部复位。在选择序列的分支与合并处（见图 4-12b），一个转换实际上只有一个前级步和一个后续步，但是一个步可能有多个前级步或多个后续步。

转换实现的基本规则是根据顺序功能图设计梯形图的基础，它适用于顺序功能图中的各种基本结构和下一章中将要介绍的顺序控制梯形图的编程方法。

如果转换的前级步或后续步不止一个，转换的实现称为同步实现（见图 4-14）。为了强调同步实现，有向连线的水平部分用双线表示。

在梯形图中，用编程元件（例如 M）代表步，当某步为活动步时，该步对应的编程元件为 1 状态。当该步之后的转换条件满足时，转换条件对应的触点或电路接通，因此可以将该触点或电路与代表所有前级步的编程元件的常开触点串联，作为与转换实现的两个条件同时满足对应的电路。

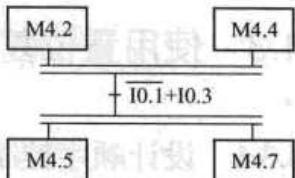


图 4-14 转换的同步实现

以图 4-14 为例，转换条件的布尔代数表达式为  $\overline{I0.1} + I0.3$ ，它的两个前级步对应于 M4.2 和 M4.4，应将 M4.2、M4.4 的常开触点组成的串联电路与 I0.3 的常开触点和 I0.1 的常闭触点组成的并联电路串联，作为转换实现的两个条件同时满足对应的电路。在梯形图中，该电路接通时，应使代表前级步的 M4.2 和 M4.4 复位（变为 0 状态并保持），同时使代表后续步的 M4.5 和 M4.7 置位（变为 1 状态并保持），完成以上任务的电路将在下一节中介绍。

### 3. 绘制顺序功能图时的注意事项

下面是针对绘制顺序功能图时常见的错误提出的注意事项：

- 1) 两个步绝对不能直接相连，必须用一个转换将它们分隔开。
- 2) 两个转换也不能直接相连，必须用一个步将它们分隔开。第 1 条和第 2 条可以作为检查顺序功能图是否正确的判据。

3) 顺序功能图中的初始步一般对应于系统等待起动的初始状态，这一步可能没有什么输出处于 ON 状态，因此有的初学者在画顺序功能图时很容易遗漏这一步。初始步是必不可少的，一方面因为该步与它的相邻步相比，从总体上说输出变量的状态各不相同；另一方面如果没有该步，无法表示初始状态，系统也无法返回等待起动的停止状态。

4) 自动控制系统应能多次重复执行同一工艺过程，因此在顺序功能图中一般应有由步和有向连线组成的闭环，即在完成一次工艺过程的全部操作之后，应从最后一步返回初始步，系统停留在初始状态（单周期操作，见图 4-9），在连续循环工作方式时，应从最后一步返回下一工作周期开始运行的第一步（见图 4-13）。

### 4. 顺序控制设计法的本质

经验设计法实际上是试图用输入信号 I 直接控制输出信号 Q（见图 4-15a），如果无法直

接控制，或者为了实现记忆和互锁等功能，只好被动地增加一些辅助元件和辅助触点。由于不同的系统的输出量 Q 与输入量 I 之间的关系各不相同，以及它们对联锁、互锁的要求千变万化，不可能找出一种简单通用的设计方法。



图 4-15 信号关系图

顺序控制设计法则是用输入量 I 控制代表各步的编程元件（例如内部位存储器 M），再用它们控制输出量 Q（见图 4-15b）。步是根据输出量 Q 的状态划分的，M 与 Q 之间具有很简单的“或”或者相等的逻辑关系，输出电路的设计极为简单。任何复杂系统的代表步的存储器位 M 的控制电路，其设计方法都是通用的，并且很容易掌握，所以顺序控制设计法具有简单、规范、通用的优点。由于 M 是依次变为 ON/OFF 状态的，实际上已经基本上解决了经验设计法中的记忆和联锁等问题。

### 4.3 使用置位复位指令的顺序控制梯形图设计方法

#### 4.3.1 设计顺序控制梯形图的一些基本问题

本节介绍根据顺序功能图设计梯形图的方法，4.4 节介绍具有多种工作方式的控制系统的办法。

本节介绍的编程方法很容易掌握，用它们可以迅速地、得心应手地设计出复杂的数字量控制系统的梯形图。

控制系统的梯形图一般采用图 4-16 所示的典型结构，程序中的汉字是作者添加的。在对 CPU 组态时，设置 MB1 为系统存储器字节（见图 2-32），M1.2 的常开触点一直闭合，每次扫描都会执行公用程序。系统有自动和手动两种工作方式，自动方式和手动方式都需要执行的操作放在公用程序 FC1 中，公用程序还用于自动程序和手动程序相互切换的处理。M6.0 是自动/手动切换开关，当它为 1 状态时调用手动程序 FC2，为 0 状态时调用自动程序 FC3。开始执行自动程序时，要求系统处于与自动程序的顺序功能图的初始步对应的初始状态。如果开机时系统没有处于初始状态，则应进入手动工作方式，用手动操作使系统进入初始状态后，再切换到自动工作方式，也可以设置使系统自动进入初始状态的工作方式（见 4.4 节）。

在本节中，假设刚开始执行用户程序时，系统的机械部分已经处于要求的初始状态。在 OB1 中用仅在首次扫描循环时为 1 状态的 M1.0 将初始步对应的编程元件（例如图 4-17 中的 M4.0）置为 1 状态，其余各步的编程元件（例如图中的 M4.1~M5.7）置为 0 状态，为转换的实现作好准备。

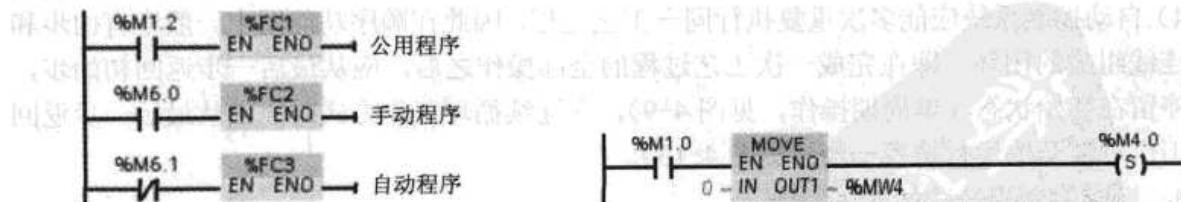


图 4-16 OB1 中的程序

图 4-17 OB1 中的初始化电路

### 4.3.2 单序列的编程方法

#### 1. 设计控制置位复位的电路的方法

在顺序功能图中,如果某一转换所有的前级步都是活动步,并且满足相应的转换条件,则转换实现。即该转换所有的后续步都变为活动步,该转换所有的前级步都变为不活动步。用该转换所有前级步对应的存储器位的常开触点与转换对应的触点或电路串联,来使所有后续步对应的存储器位置位,和使所有前级步对应的存储器位复位。置位和复位操作分别使用置位指令和复位指令。在任何情况下,代表步的存储器位的控制电路都可以用这一原则来设计,每一个转换对应一个这样的控制置位和复位的电路块,有多少个转换就有多少个这样的电路块。这种设计方法特别有规律,梯形图与转换实现的基本规则之间有着严格的对应关系,在设计复杂的顺序功能图的梯形图时既容易掌握,又不容易出错。

#### 2. 编程方法应用举例

在STEP 7 Basic 的项目视图中生成一个名为“Dolly”的新项目(见随书光盘中的同名例程),CPU 的型号为 CPU 1214C。

将图 4-9 的小车控制系统的顺序功能图重新画在图 4-18 中。实现图中 I0.1 对应的转换需要同时满足两个条件,即该转换的前级步是活动步(M4.1 为 1 状态)和转换条件满足(I0.1 为 1 状态)。在梯形图中,用 M4.1 和 I0.1 的常开触点组成的串联电路来表示上述条件。该电路接通时,两个条件同时满足。此时应将该转换的后续步变为活动步,即用置位指令(S 指令)将 M4.2 置位。还应将该转换的前级步变为不活动步,即用复位指令(R 指令)将 M4.1 复位。

用上述的方法编写控制代表步的 M4.0~M4.3 的电路,每一个转换对应一个这样的电路(见图 4-19)。

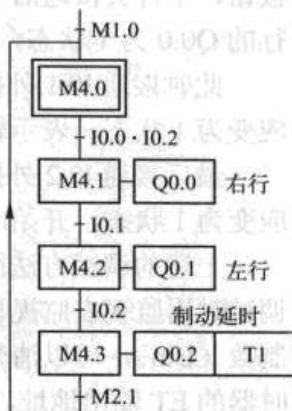


图 4-18 顺序功能图

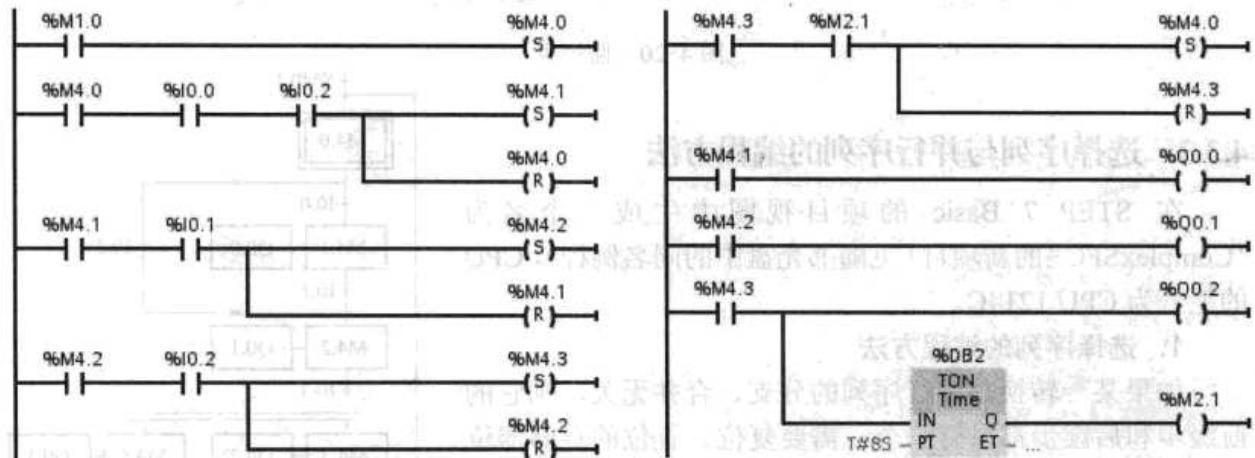


图 4-19 OB1 中的梯形图

#### 3. 输出电路的处理

使用这种编程方法时,不能将输出位的线圈与置位指令和复位指令并联,这是因为图 4-19 中控制置位、复位的串联电路接通的时间是相当短的,只有一个扫描周期。转换条件 I0.1 满足后,前级步 M4.1 被复位,下一个扫描循环周期 M4.1 和 I0.1 的常开触点组成的串联电路断

开，而输出位 Q 的线圈至少应该在某一步对应的全部时间内被接通。所以应根据顺序功能图，用代表步的存储器位的常开触点或它们的并联电路来驱动输出位的线圈。

在制动延时步，M4.3 为 1 状态，它的常开触点接通，使 TON 定时器开始定时。定时时间到时，定时器的 Q 输出端控制的 M2.1 变为 1 状态，转换条件满足，将转换到初始步 M4.0。

#### 4. 程序的调试

顺序功能图是用来描述控制系统的外部性能的，因此应根据顺序功能图而不是梯形图来调试顺序控制程序。

将图 4-19 中的用户程序下载到 CPU 后，将 CPU 切换到 RUN 模式。此时初始步 M4.0 应为活动步。接通 I0.2 对应的小开关，模拟左限位开关动作。用 I0.0 外接的小开关模拟起动按钮，小开关接通后马上断开。CPU 上 Q0.0 对应的 LED（发光二极管）应亮，说明控制右行的 Q0.0 为 1 状态，转换到了步 M4.1。

此时接通 I0.1 外接的小开关后马上断开，模拟右限位开关动作。Q0.0 应变为 0 状态，Q0.1 应变为 1 状态，表示转换到了步 M4.2，小车左行。

最后接通 I0.2 外接的小开关，模拟左限位开关动作。Q0.1 应变为 0 状态，停止左行。Q0.2 应变为 1 状态，开始制动。经过设定的延时后，Q0.2 应变为 0 状态。

上述的调试方法简单方便，但是看不到 CPU 内部代表步的 M4.0~M4.3 的状态变化。在调试时用监视表监视与顺序控制有关的 MB0、QB1 和 IB0（见图 4-20），显示模式均为二进制数（Bin），可以清楚地看到步的活动状态的转换情况。如果需要，可以在梯形图中设置定时器的 ET 输出地址，在监视表中监视该地址，可以看到定期间定时器的当前时间值。

Name	Address	Display format	Monitor value	Modify value
1	%MB4	Bin	0000_0100	
2	%QB0	Bin	0000_0010	
3	%IB0	Bin	0000_0000	

图 4-20 监视表

### 4.3.3 选择序列与并行序列的编程方法

在 STEP 7 Basic 的项目视图中生成一个名为“ComplexSFC”的新项目（见随书光盘中的同名例程），CPU 的型号为 CPU 1214C。

#### 1. 选择序列的编程方法

如果某一转换与并行序列的分支、合并无关，则它的前级步和后续步都只有一个，需要复位、置位的存储器位也只有一个，因此选择序列的分支与合并的编程方法实际上与单序列的编程方法完全相同。

图 4-21 所示的顺序功能图中，除了 I0.3 与 I0.6 对应的转换以外，其余的转换均与并行序列的分支、合并无关，I0.0~I0.2 对应的转换与选择序列的分支、合并有关，它们

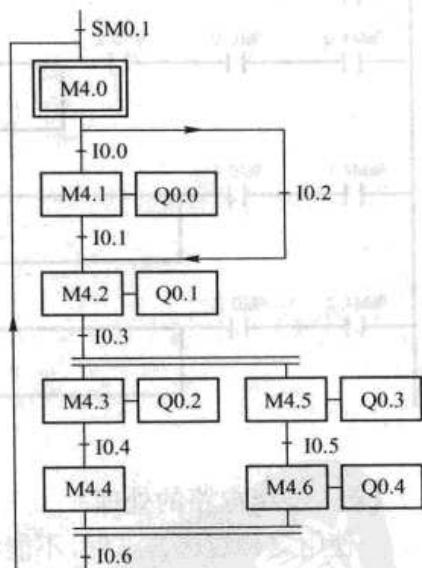


图 4-21 选择序列与并行序列

都只有一个前级步和一个后续步。与并行序列的分支、合并无关的转换对应的梯形图是非常标准的，每一个控制置位、复位的电路块都由前级步对应的一个存储器位的常开触点和转换条件对应的触点组成的串联电路、一条置位指令和一条复位指令组成。

## 2. 并行序列的编程方法

图 4-21 中步 M4.2 之后有一个并行序列的分支，当 M4.2 是活动步，并且转换条件 I0.3 满足时，步 M4.3 与步 M4.5 应同时变为活动步，这是用 M4.2 和 I0.3 的常开触点组成的串联电路使 M4.3 和 M4.5 同时置位来实现的。与此同时，步 M4.2 应变为不活动步，这是用复位指令来实现的。

I0.6 对应的转换之前有一个并行序列的合并，该转换实现的条件是所有的前级步（即步 M4.4 和 M4.6）都是活动步，和转换条件 I0.6 满足。由此可知，应将 M4.4、M4.6 和 I0.6 的常开触点串联，作为控制后续步 M4.0 置位和前级步 M4.4、M4.6 复位的电路。

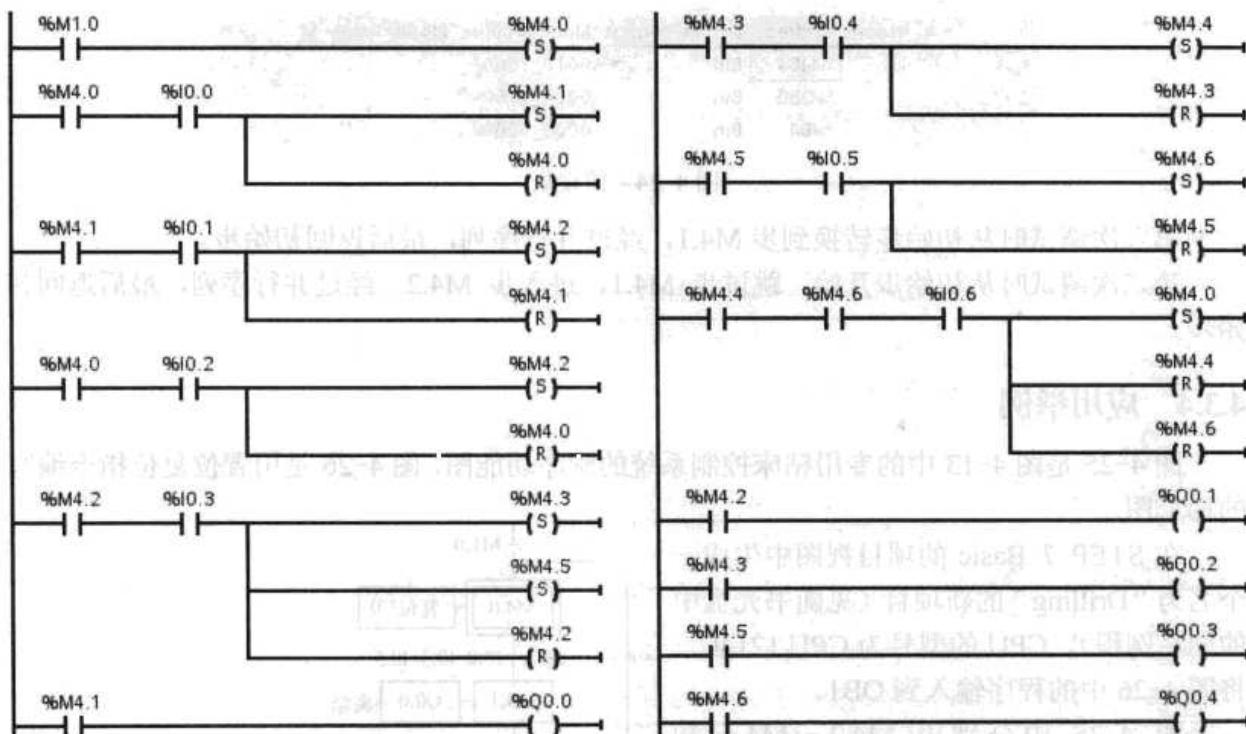


图 4-22 选择序列与并行序列的梯形图

图 4-23 的顺序功能图中，转换的上面是并行序列的合并，转换的下面是并行序列的分支，该转换实现的条件是所有的前级步（即步 M4.2 和 M4.4）都是活动步和转换条件  $\overline{I0.1+I0.3}$  满足。因此应将 M4.2、M4.4、I0.3 的常开触点与 I0.1 的常闭触点组成的串并联电路，作为使 M4.5、M4.7 置位和使 M4.2、M4.4 复位的条件。

## 3. 复杂的顺序功能图的调试方法

调试复杂的顺序功能图时，应充分考虑各种可能的情况，对系统的各种工作方式、顺序功能图中的每一条支路、各种可能的进展路线，都应逐一检查，不能遗漏。特别要注意并行序列中各子序列的第一步（图 4-21 中的步 M4.3 和步 M4.5）是否同时变为活动步，最后一步（步 M4.4 和步 M4.6）是否同时变为不活动步。

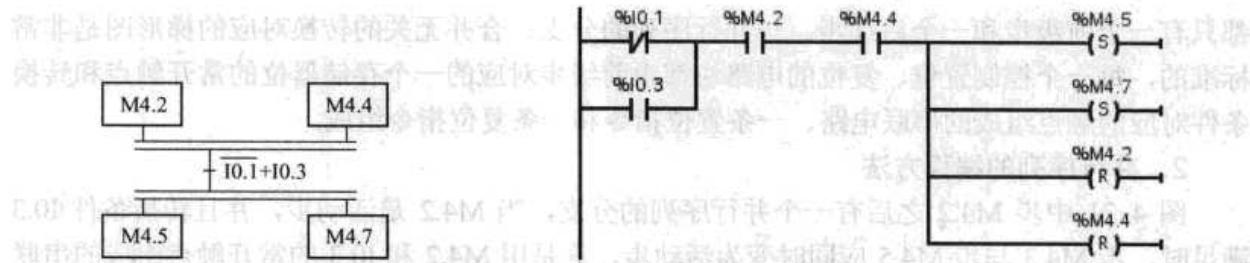


图 4-23 转换的同步实现

发现问题后应及时修改程序，直到每一条进展路线上步的活动状态的顺序变化和输出点的变化都符合顺序功能图的规定。

将图 4-22 中的程序输入到 OB1。调试时可以在监视表中用二进制格式监视 MB0、QB0 和 IB0（见图 4-24），在 RUN 模式时点击状态表工具栏上的 按钮，起动监控功能。

	Name	Address	Display format	Monitor value	Modify value
1	%MB4	Bin		0010_1000	
2	%QB0	Bin		0000_1100	
3	%IB4	Bin		0000_0000	

图 4-24 监视表

第一次调试时从初始步转换到步 M4.1，经过并行序列，最后返回初始步。

第二次调试时从初始步开始，跳过步 M4.1，进入步 M4.2。经过并行序列，最后返回初始步。

#### 4.3.4 应用举例

图 4-25 是图 4-13 中的专用钻床控制系统的顺序功能图，图 4-26 是用置位复位指令编写的梯形图。

在 STEP 7 Basic 的项目视图中生成一个名为“Drilling”的新项目（见随书光盘中的同名例程），CPU 的型号为 CPU 1214C。将图 4-26 中的程序输入到 OB1。

图 4-25 中分别由 M4.2～M4.4 和 M4.5～M4.7 组成的两个单序列是并行工作的，设计梯形图时应保证这两个序列同时开始工作和同时结束，即两个序列的第一步 M4.2 和 M4.5 应同时变为活动步，两个序列的最后一步 M4.4 和 M4.7 应同时变为不活动步。

并行序列的分支的处理是很简单的，在图 4-25 中，当步 M4.1 是活动步，并且转换条件 I0.1 为 1 状态时，步 M4.2 和 M4.5 同时变为活动步，两个序列开始同时工作。在梯形图中，用 M4.1 和 I0.1 的常开触点组成

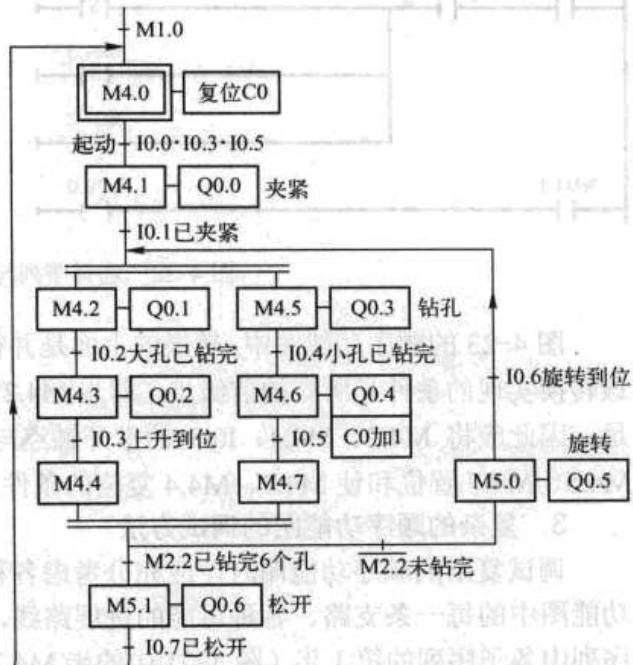


图 4-25 组合钻床控制系统的顺序功能图

的串联电路来控制对 M4.2 和 M4.5 的同时置位，和对前级步 M4.1 的复位。

另一种情况是当步 M5.0 为活动步，并且转换条件 I0.6 为 1 状态时，步 M4.2 和 M4.5 也应同时变为活动步，两个序列开始同时工作。在梯形图中，用 M5.0 和 I0.6 的常开触点组成的串联电路，来控制对 M4.2 和 M4.5 的同时置位，和对前级步 M5.0 的复位。

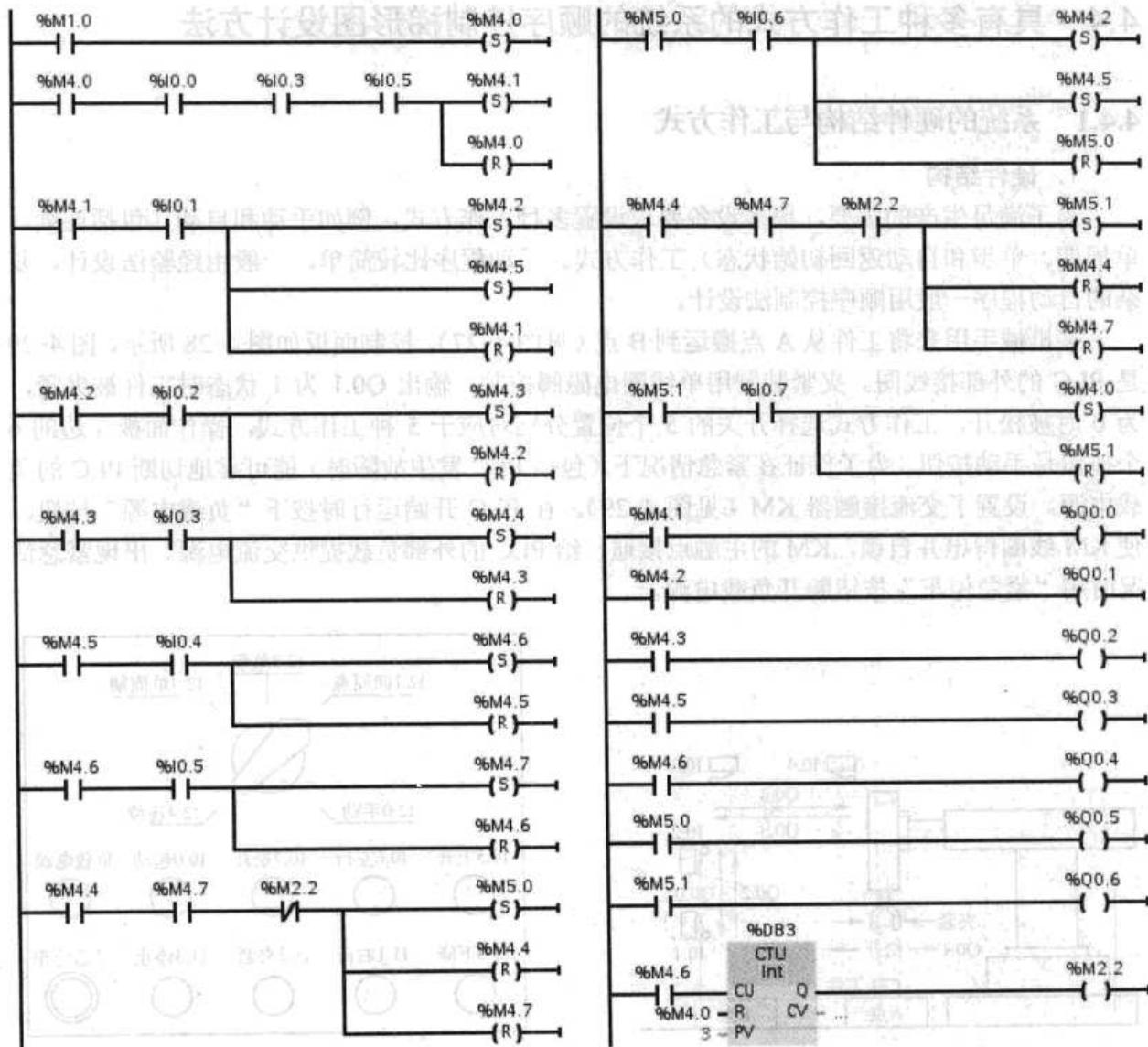


图 4-26 组合钻床控制系统的梯形图

图 4-25 的并行序列合并处的转换有两个前级步 M4.4 和 M4.7，根据转换实现的基本规则，当它们均为活动步并且转换条件满足时，将实现并行序列的合并。未钻完 3 对孔时，加计数器 C0 的当前值小于设定值 3，其常闭触点闭合，转换条件 M2.2 满足，将转换到步 M5.0。在梯形图中，用 M4.4、M4.7 的常开触点和 M2.2 的常闭触点组成的串联电路将 M5.0 置位，使后续步 M5.0 变为活动步；同时用 R 指令将 M4.4 和 M4.7 复位，使前级步 M4.4 和 M4.7 变为不活动步。

钻完 3 对孔时, C0 的当前值等于设定值 3, 其常开触点闭合, 转换条件 M2.2 满足, 将转换到步 M5.1。在梯形图中, 用 M4.4、M4.7 和 M2.2 的常开触点组成的串联电路将 M5.1 置位, 使后续步 M5.1 变为活动步; 同时用 R 指令将 M4.4 和 M4.7 复位, 使前级步 M4.4 和 M4.7 变为不活动步。

## 4.4 具有多种工作方式的系统的顺序控制梯形图设计方法

### 4.4.1 系统的硬件结构与工作方式

#### 1. 硬件结构

为了满足生产的需要, 很多设备要求设置多种工作方式, 例如手动和自动(包括连续、单周期、单步和自动返回初始状态)工作方式。手动程序比较简单, 一般用经验法设计, 复杂的自动程序一般用顺序控制法设计。

某机械手用来将工件从 A 点搬运到 B 点(见图 4-27), 控制面板如图 4-28 所示, 图 4-29 是 PLC 的外部接线图。夹紧装置用单线圈电磁阀控制, 输出 Q0.1 为 1 状态时工件被夹紧, 为 0 时被松开。工作方式选择开关的 5 个位置分别对应于 5 种工作方式, 操作面板左边的 6 个按钮是手动按钮。为了保证在紧急情况下(包括 PLC 发生故障时)能可靠地切断 PLC 的负载电源, 设置了交流接触器 KM(见图 4-29)。在 PLC 开始运行时按下“负载电源”按钮, 使 KM 线圈得电并自锁, KM 的主触点接通, 给 PLC 的外部负载提供交流电源。出现紧急情况时用“紧急停车”按钮断开负载电源。

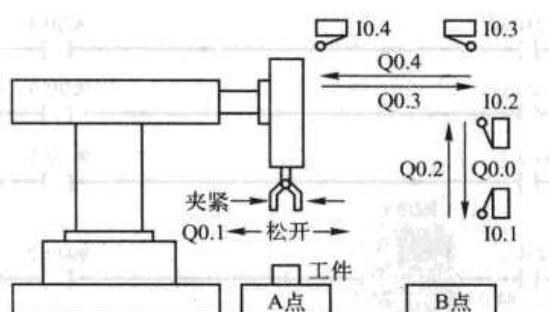


图 4-27 机械手示意图

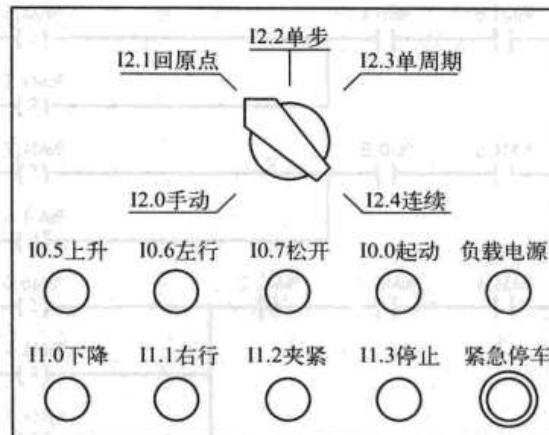


图 4-28 操作面板

#### 2. 工作方式

系统设有手动、单周期、单步、连续和回原点 5 种工作方式。

在手动工作方式, 用 I0.5~I1.2 对应的 6 个按钮分别独立控制机械手的升、降、左行、右行和夹紧、松开。

机械手从初始状态开始, 将工件从 A 点搬运到 B 点, 最后返回初始状态的过程, 称为一个工作周期。

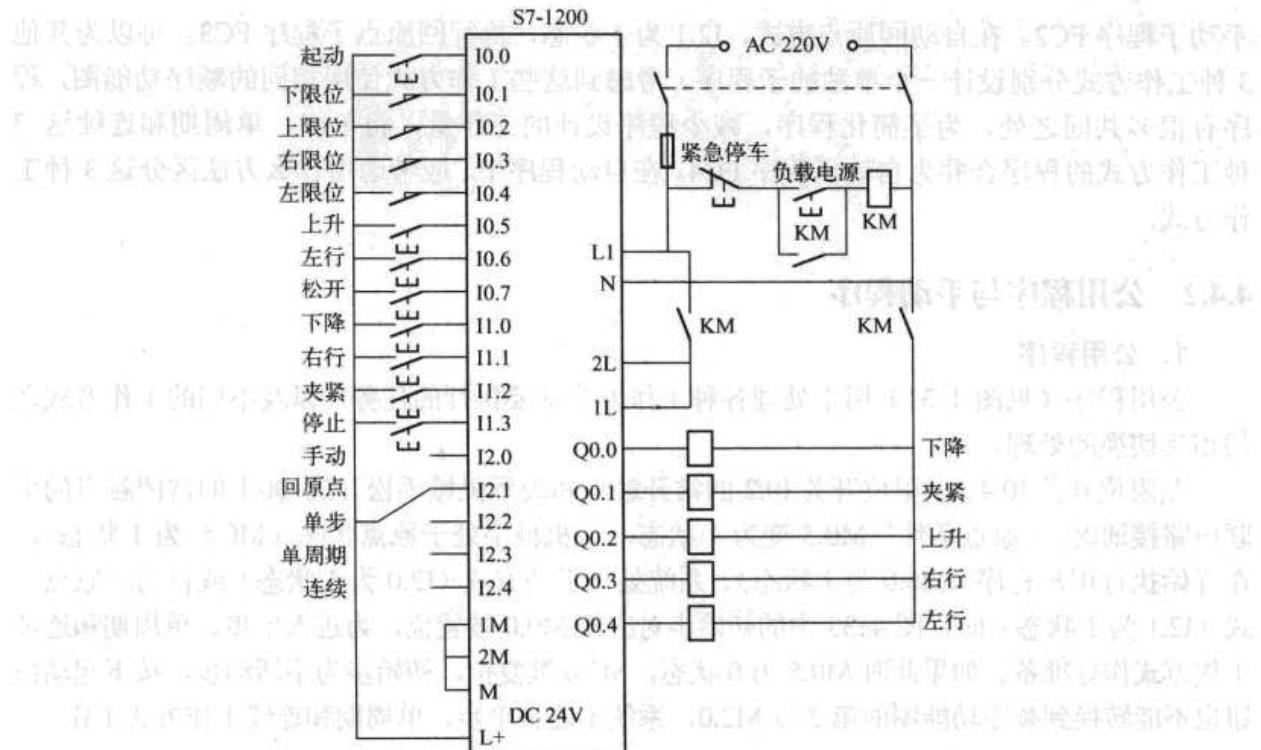


图 4-29 外部接线图

在单周期工作方式，在初始状态按下起动按钮 I0.0，从初始步 M0.0 开始，机械手按顺序功能图（见图 4-33）的规定完成一个周期的工作后，返回并停留在初始步。

在连续工作方式，在初始状态按下起动按钮，机械手从初始步开始，工作一个周期后又开始搬运下一个工件，反复连续地工作。按下停止按钮，并不马上停止工作，完成最后一个周期的工作后，系统才返回并停留在初始步。

在单步工作方式，从初始步开始，按一下起动按钮，系统转换到下一步，完成该步的任务后，自动停止工作并停留在该步，再按一下起动按钮，才开始执行下一步的操作。单步工作方式常用于系统的调试。

机械手在最上面和最左边且夹紧装置松开时，称为系统处于原点状态（或称初始状态）。在进入单周期、连续和单步工作方式之前，系统应处于原点状态。如果不满足这一条件，可以选择回原点工作方式，然后按起动按钮 I0.0，使系统自动返回原点状态。

### 3. 程序的总体结构

在 STEP 7 Basic 的项目视图中生成一个名为“Manipulator”的新项目（见随书光盘中的同名例程），CPU 的型号为 CPU 1214C。本例程需要添加一块 8 点的 DI 模块，在硬件组态时将自动分配的模块字节地址改为 2。

图 4-30 是 OB1 中的程序。在 CPU 组态时设置系统存储器字节（见图 2-32）为 MB6，M6.2 的常开触点一直闭合，公用程序 FC1 是无条件执行的。在手动方式，I2.0 为 1 状态，执行

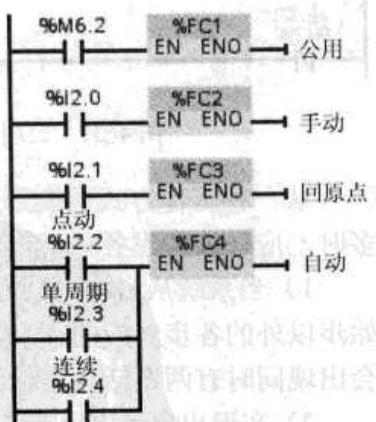


图 4-30 OB1 的程序

手动子程序 FC2。在自动回原点方式, I2.1 为 1 状态, 执行回原点子程序 FC3。可以为其他 3 种工作方式分别设计一个单独的子程序。考虑到这些工作方式使用相同的顺序功能图, 程序有很多共同之处, 为了简化程序, 减少程序设计的工作量, 将单步、单周期和连续这 3 种工作方式的程序合并为自动子程序 FC4。在自动程序中, 应考虑用什么方法区分这 3 种工作方式。

#### 4.4.2 公用程序与手动程序

##### 1. 公用程序

公用程序(见图 4-31)用于处理各种工作方式都要执行的任务, 以及不同的工作方式之间相互切换的处理。

左限位开关 I0.4、上限位开关 I0.2 的常开触点和表示机械手松开的 Q0.1 的常闭触点的串联电路接通时, “原点条件” M0.5 变为 1 状态。当机械手处于原点状态(M0.5 为 1 状态), 在开始执行用户程序(M6.0 为 1 状态)、系统处于手动方式(I2.0 为 1 状态)或自动回原点方式(I2.1 为 1 状态)时, 图 4-33 中的初始步对应的 M0.0 被置位, 为进入单步、单周期和连续工作方式作好准备。如果此时 M0.5 为 0 状态, M0.0 被复位, 初始步为不活动步, 按下起动按钮也不能转换到顺序功能图的第 2 步 M2.0, 系统不能在单步、单周期和连续工作方式工作。

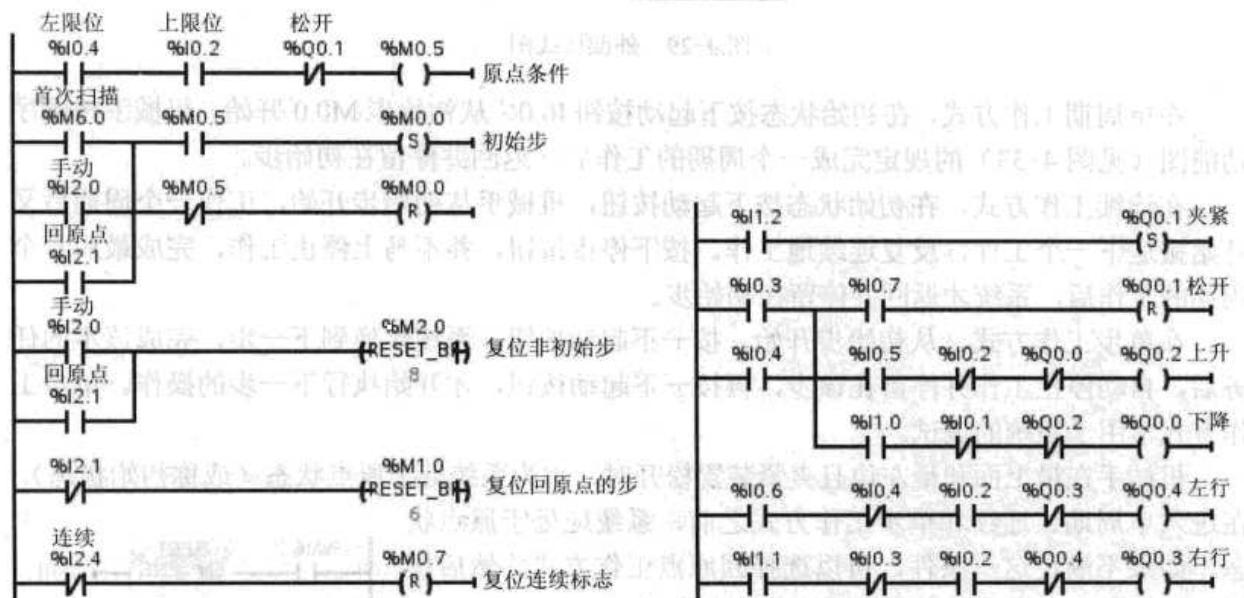


图 4-31 公用程序

图 4-32 手动程序

从一种工作方式切换到另一种工作方式时, 应将有存储功能的位元件复位。工作方式较多时, 应仔细考虑各种可能的情况, 分别进行处理。在切换工作方式时应执行下列操作:

- 1) 当系统从自动工作方式切换到手动或自动回原点工作方式时, 必须将图 4-33 中除初始步以外的各步对应的存储器位(M2.0~M2.7)复位, 否则以后返回自动工作方式时, 可能会出现同时有两个活动步的异常情况, 引起错误的动作。
- 2) 在退出自动回原点工作方式时, I2.1 的常闭触点闭合。此时应将自动回原点的顺序功能图(见图 4-37)中所有的步对应的存储器位(M1.0~M1.5)复位, 以防止下次进入自动回

原点方式时，可能会出现同时有两个活动步的异常情况。

3) 非连续工作方式时 I2.4 的常闭触点闭合，将表示连续工作状态的标志 M0.7 复位。

## 2. 手动程序

图 4-32 是手动程序，为了保证系统的安全运行，在手动程序中设置了一些必要的联锁：

1) 设置上升与下降之间、左行与右行之间的互锁，以防止功能相反的两个输出同时为 1 状态。

2) 用限位开关 I0.1~I0.4 的常闭触点限制机械手移动的范围。

3) 上限位开关 I0.2 的常开触点与控制左、右行的 Q0.4 和 Q0.3 的线圈串联，机械手升到最高位置才能左右移动，以防止机械手在较低位置运行时与别的物体碰撞。

4) 只允许机械手在最左边或最右边时 (I0.3 或 I0.4 的常开触点闭合) 上升、下降和松开工件。

## 4.4.3 自动程序

图 4-33 是处理单周期、连续和单步工作方式的顺序功能图，最上面的转换条件与公用程序有关。图 4-34 是用置位复位指令设计的程序。

单周期、连续和单步这 3 种工作方式主要是用“连续”标志 M0.7 和“转换允许”标志 M0.6 来区分的。

### 1. 单步与非单步的区分

M0.6 的常开触点串接在每一个控制置位复位操作的串联电路中，它们断开时禁止步的活动状态的转换。

系统工作在连续和单周期 (非单步) 工作方式时，I2.2 的常闭触点接通，使 M0.6 为 1 状态，控制置位复位的电路中的 M0.6 的常开触点接通，允许步与步之间的正常转换。

M0.6 对单步方式步的活动状态的转换过程的控制见后面对单步工作方式的介绍。

### 2. 单周期与连续的区分

PLC 上电后如果原点条件不满足，应首先进入单步或回原点方式，通过相应的操作使原点条件满足，公用程序使初始步 M0.0 为 1 状态，然后切换到自动方式。

在单周期和连续工作方式，I2.2 (单步方式) 的常闭触点闭合，M0.6 的线圈 “通电” (见图 4-34)，允许转换。

连续工作方式时 I2.4 为 1 状态，在初始步为活动步时按下起动按钮 I0.0，控制连续工作的 M0.7 的线圈 “通电” 并自保持。图 4-34 左边第 5 行的 M0.0、I0.0、M0.5 (原点条件) 和 M0.6 的常开触点均接通，将 M2.0 置位，系统进入下降步，Q0.0 的线圈 “通电”，机械手下降。

机械手碰到下限位开关 I0.1 时，转换到夹紧步 M2.1，Q0.1 被置位，夹紧电磁阀的线圈通电并保持。同时接通延时定时器 T0 开始定时，2s 后定时时间到，工件被夹紧，定时器 Q 输出端控制

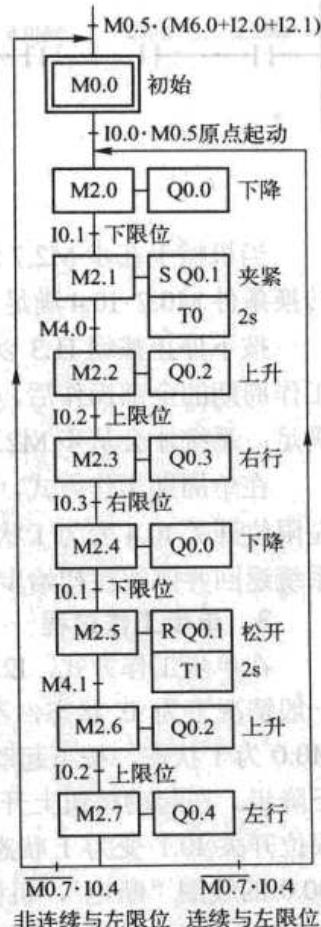


图 4-33 顺序功能图

的 M4.0 变为 1 状态，转换条件 M4.0 满足，转换到步 M2.2。以后系统将这样一步一步地工作下去。

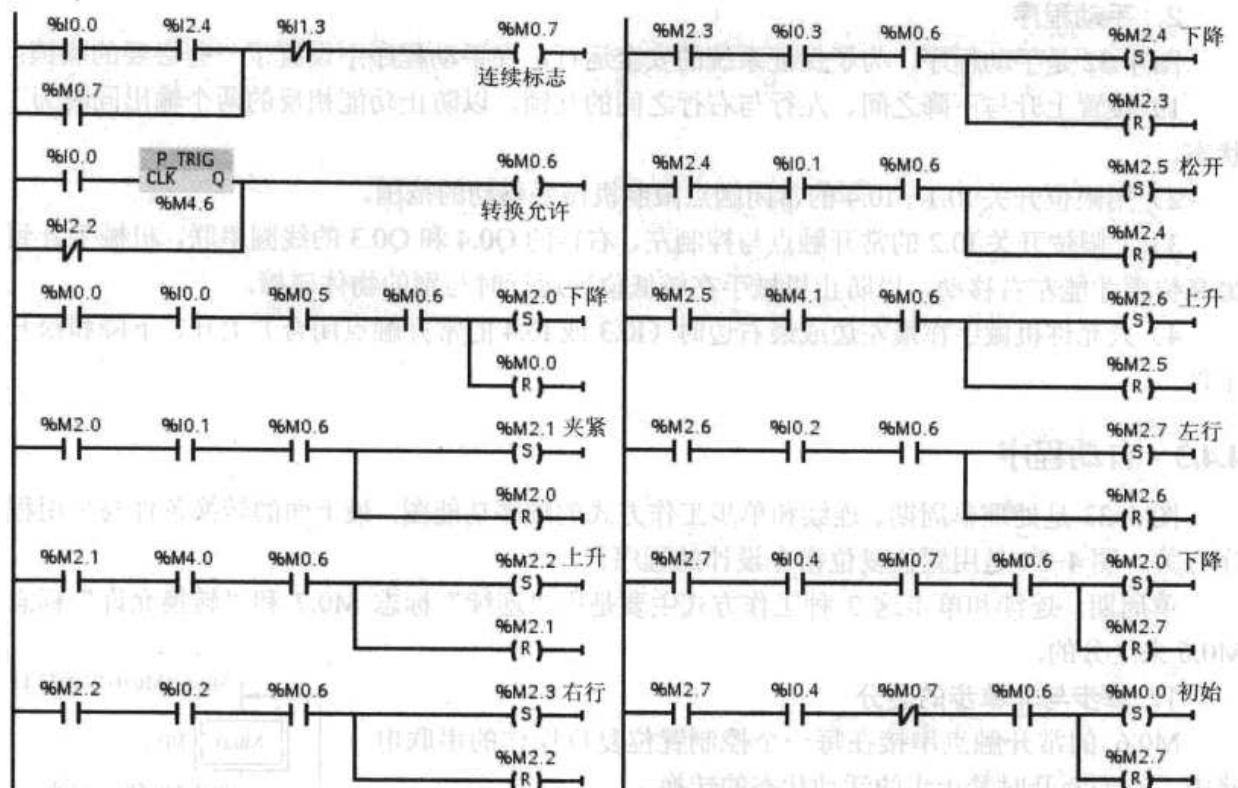


图 4-34 FC4 中的梯形图

当机械手在步 M2.7 返回最左边时，I0.4 为 1 状态，因为“连续”标志位 M0.7 为 1 状态，转换条件 M0.7·I0.4 满足，系统将返回步 M2.0，反复连续地工作下去。

按下停止按钮 I1.3 以后，M0.7 变为 0 状态，但是机械手不会立即停止工作，在完成当前工作周期的全部操作后，机械手返回最左边，左限位开关 I0.4 为 1 状态，转换条件 M0.7·I0.4 满足，系统才会从步 M2.7 返回并停留在初始步 M0.0。

在单周期工作方式，M0.7 一直处于 0 状态。当机械手在最后一步 M2.7 返回最左边时，左限位开关 I0.4 变为 1 状态，因为连续工作标志 M0.7 为 0 状态，转换条件 M0.7·I0.4 满足，系统返回并停留在初始步 M0.0，机械手停止运动。按一次起动按钮，系统只工作一个周期。

### 3. 单步工作过程

在单步工作方式，I2.2 为 1 状态，它的常闭触点断开，“转换允许”辅助继电器 M0.6 在一般情况下为 0 状态，不允许步与步之间的转换。设初始步时系统处于原点状态，M0.5 和 M0.0 为 1 状态，按下起动按钮 I0.0，M0.6 变为 1 状态，使 M2.0 的起动电路接通，系统进入下降步。在起动按钮上升沿之后，M0.6 变为 0 状态。在下降步，Q0.0 的线圈“通电”，当下限位开关 I0.1 变为 1 状态时，与 Q0.0 的线圈串联的 I0.1 的常闭触点断开（见图 4-35），使 Q0.0 的线圈“断电”，机械手停止下降。I0.1 的常开触点闭合后，如果没有按起动按钮，I0.0 和 M0.6 处于 0 状态，不会转换到下一步。一直要等到按下起动按钮，I0.0 和 M0.6 变为 1 状态，M0.6 的常开触点接通，才能使转换条件 I0.1 起作用，M2.1 被置位，系统才能由步 M2.0

进入步 M2.1。以后在完成某一步的操作后，都必须按一次起动按钮，系统才能转换到下一步。

图 4-34 左边第 3 行 I0.0 的常开触点与指令 P\_TRIG 的串联电路，可以用 I0.0 的上升沿检测触点来代替。

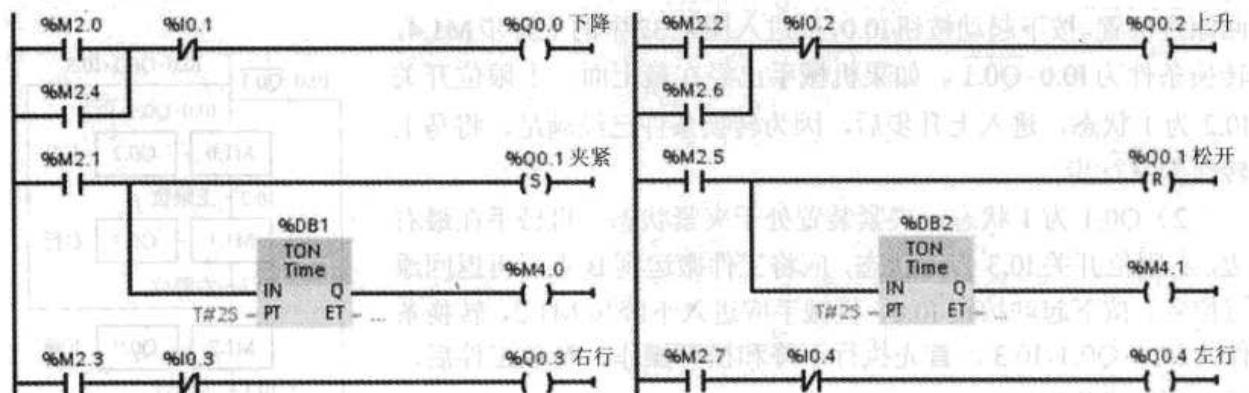


图 4-35 FC4 中的输出电路

#### 4. 输出电路

输出电路（见图 4-35）是自动程序 FC4 的一部分，输出电路中 I0.1~I0.4 的常闭触点是为单步工作方式设置的。以下降为例，当小车碰到下限位开关 I0.1 后，与下降步对应的存储器位 M2.0 或 M2.4 不会马上变为 0 状态。如果 Q0.0 的线圈不与 I0.1 的常闭触点串联，机械手不能停在下限位开关 I0.1 处，还会继续下降。对于某些设备，可能造成事故。

#### 5. 程序的调试

在调试时用监视表（见图 4-36）监视与顺序控制有关的 MB0、MB1、MB2 和 QB0，显示模式均为二进制数（Bin）。

如果仅仅是作为编程练习，可以不增加 DI 模块，在调试程序时，用监视表将 I2.0~I2.4 中的某一位置 1，其他位清零，用这样的方法来设置工作方式。

	Name	Address	Display format	Monitor value	Modify value	
1		%MB0	Bin	0100_0000		
2		%MB2	Bin	0000_1000		
3		%QB0	Bin	0000_1010		
4		%MB1	Bin	0000_0000		
5	"Tag_12"	%I2.0	Bool	<input type="checkbox"/> FALSE	<input checked="" type="checkbox"/> FALSE	<input checked="" type="checkbox"/>
6	"Tag_13"	%I2.1	Bool	<input type="checkbox"/> FALSE	<input checked="" type="checkbox"/> FALSE	<input checked="" type="checkbox"/>
7	"Tag_14"	%I2.2	Bool	<input type="checkbox"/> FALSE	<input checked="" type="checkbox"/> FALSE	<input checked="" type="checkbox"/>
8	"Tag_15"	%I2.3	Bool	<input type="checkbox"/> TRUE	<input checked="" type="checkbox"/> TRUE	<input checked="" type="checkbox"/>
9	"Tag_22"	%I2.4	Bool	<input type="checkbox"/> FALSE	<input checked="" type="checkbox"/> FALSE	<input checked="" type="checkbox"/>

图 4-36 监视表

#### 4.4.4 自动回原点程序

图 4-37 是自动回原点程序的顺序功能图，图 4-38 是用置位复位电路设计的梯形图。在回原点工作方式，I2.1 为 1 状态。按下起动按钮 I0.0 时，机械手可能处于任意

状态，根据机械手当时所处的位置和夹紧装置的状态，可以分为 3 种情况，采用不同的处理方法：

1) Q0.1 为 0 状态，表示夹紧装置松开，机械手没有夹持工件，应上升和左行，直接返回原点位置。按下起动按钮 I0.0，应进入图 4-37 中的上升步 M1.4，转换条件为 I0.0·Q0.1。如果机械手已经在最上面，上限位开关 I0.2 为 1 状态，进入上升步后，因为转换条件已经满足，将马上转换到左行步。

2) Q0.1 为 1 状态，夹紧装置处于夹紧状态。机械手在最右边，右限位开关 I0.3 为 1 状态，应将工件搬运到 B 点后再返回原点位置。按下起动按钮 I0.0，机械手应进入下降步 M1.2，转换条件为 I0.0·Q0.1·I0.3，首先执行下降和松开操作，释放工件后，再返回原点位置。

3) Q0.1 为 1 状态，夹紧装置处于夹紧状态。机械手不在最右边，I0.3 为 0 状态。按下起动按钮 I0.0，应进入步 M1.0，转换条件为 I0.0·Q0.1·I0.3，首先上升、右行、下降和松开工件，将工件搬运到 B 点后再返回原点位置。

机械手返回原点位置后，原点条件满足，公用程序中的原点条件标志 M0.5 为 1 状态，因为此时 I2.1 为 1 状态，图 4-33 的顺序功能图中的初始步 M0.0 在公用程序中被置位，为进入单周期、连续和单步工作方式作好了准备，因此可以认为自动程序的顺序功能图的初始步 M0.0 是步 M1.5 的后续步。

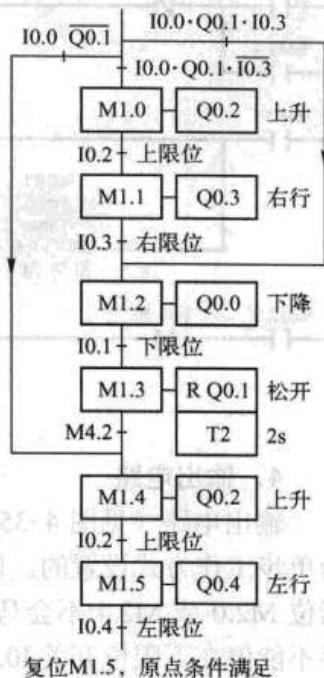


图 4-37 回原点顺序功能图

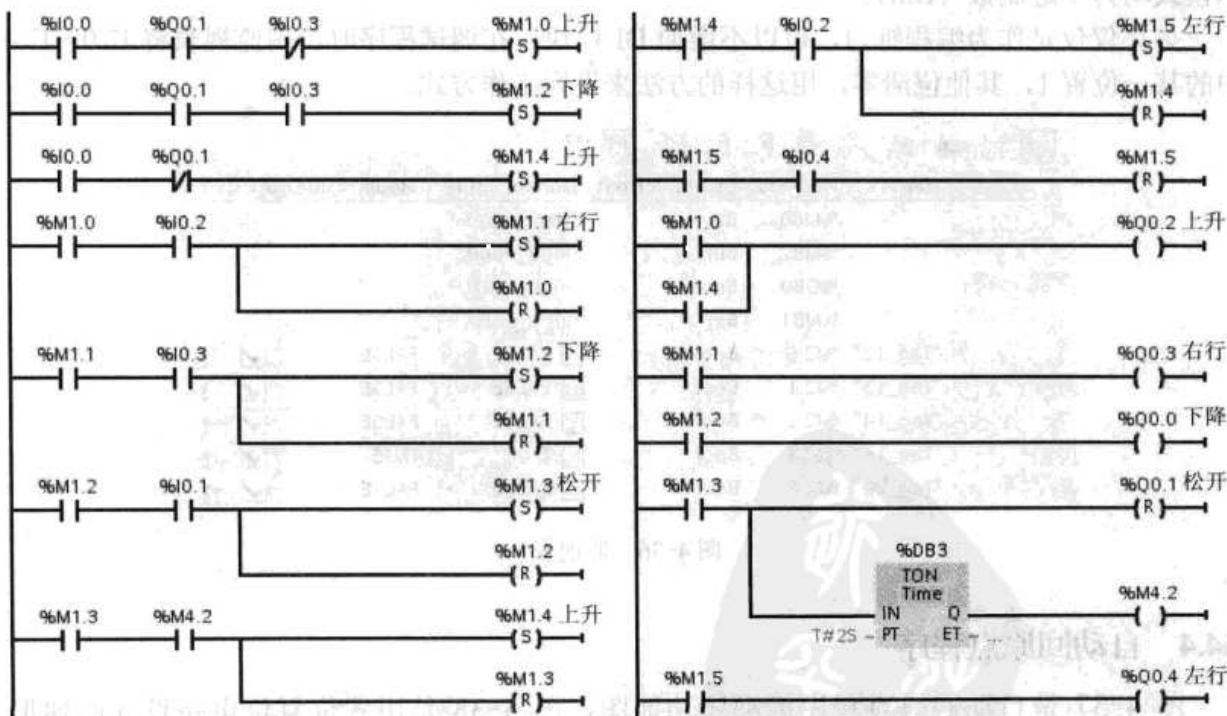


图 4-38 自动回原点的梯形图

# 第5章 S7-1200的指令

## 5.1 数据处理指令

### 5.1.1 比较指令

#### 1. 比较指令

比较指令用来比较数据类型相同的两个数 IN1 与 IN2 的大小（见图 5-1），IN1 和 IN2 分别在触点的上面和下面。它们的数据类型（见图 5-1 中的下拉式列表）应相同。操作数可以是 I、Q、M、L、D 存储区中的变量或常数。比较两个字符串时，实际上比较的是它们各对应字符的 ASCII 码的大小，第一个不相同的字符决定了比较的结果。

可以将比较指令视为一个等效的触点，比较符号可以是“==”（等于）、“<>”（不等于）、“>”、“>=”、“<”和“<=”。满足比较关系式给出的条件时，等效触点接通。例如当 MW8 的值等于-24732 时，图 5-1 第一行左边的比较触点接通。

生成比较指令后，双击触点中间比较符号下面的问号（见图 5-1 右边未输入参数的比较触点），点击出现的 ▾ 按钮，用下拉式列表设置要比较的数的数据类型。

实际上比较指令的比较符号也可以修改，双击比较符号，点击出现的 ▾ 按钮，可以用下拉式列表修改比较符号。

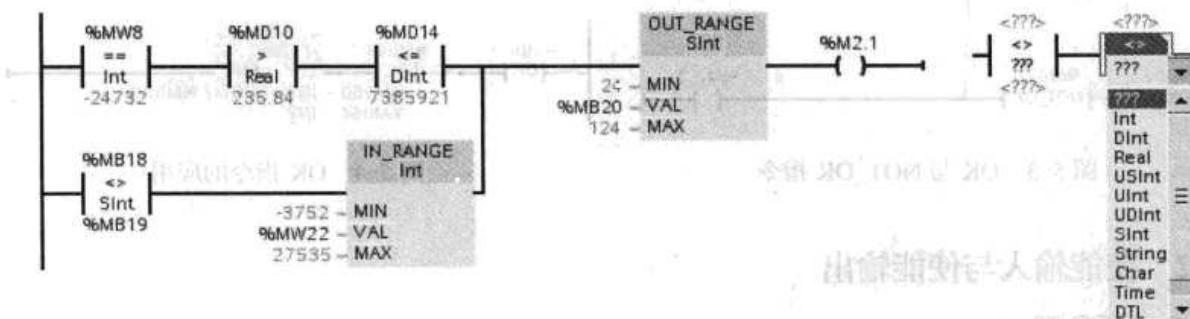


图 5-1 比较指令

#### 2. 范围内与范围外比较指令

范围内比较指令 IN\_RANGE 与范围外比较指令 OUT\_RANGE 可以等效为一个触点。如果有能流流入指令方框，执行比较。图 5-1 中的 IN\_RANGE 指令的  $\text{MIN} \leq \text{VAL} \leq \text{MAX}$  ( $-3752 \leq \text{MW22} \leq 27535$ )，或 OUT\_RANGE 指令的  $\text{VAL} < \text{MIN}$  或  $\text{VAL} > \text{MAX}$  ( $\text{MB20} < 24$  或  $\text{MB20} > 124$ ) 时，等效触点闭合，有能流流出指令框的输出端。如果不满足比较条件，没有能流输出。如果没有能流输入指令框，不执行比较，没有能流输出。

指令的 MIN、MAX 和 VAL 的数据类型必须相同，可选 SInt、Int、DInt、USInt、UInt、

UDInt、Real，可以是 I、Q、M、L、D 存储区中的变量或常数。双击指令名称下面的问号，点击出现的 ▾ 按钮，用下拉式列表框设置要比较的数据的数据类型。

**【例 5-1】** 用接通延时定时器和比较指令组成占空比可调的脉冲发生器。

M2.0 和接通延时定时器 TON 组成一个脉冲发生器，使 MD4 中 TON 的时间当前值按图 5-2 所示的波形变化。比较指令用来产生脉冲宽度可调的方波，Q0.0 为 0 的时间取决于比较触点下面的操作数的值。

MD4 用于保存定时器 TON 的时间当前值 ET，其数据类型为 Time。输入比较指令上面的操作数 MD4 后，指令中 “ $\geq$ ” 符号下面的数据类型自动变为 “Time”。输入 IN2 的值 1000 后，自动变为 T#1000ms。

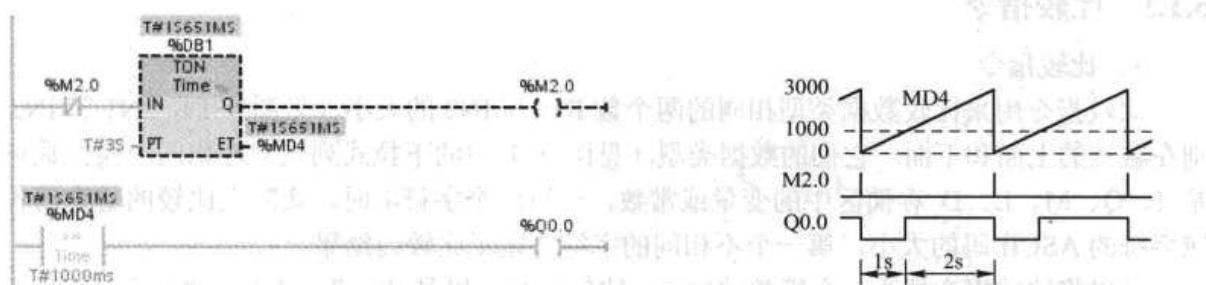


图 5-2 自复位接通延时定时器

### 3. OK 与 NOT\_OK 指令

OK 和 NOT\_OK 指令（见图 5-3）用来检测输入数据是否是实数（即浮点数）。如果是实数，OK 触点接通，反之 NOT\_OK 触点接通。触点上面的变量的数据类型为 Real。

执行图 5-4 之前，首先用 OK 指令检查乘法指令 MUL 的两个操作数是否是实数，如果不是，OK 触点断开，没有能流流入 MUL 指令的使能输入端 EN，不会执行乘法指令。

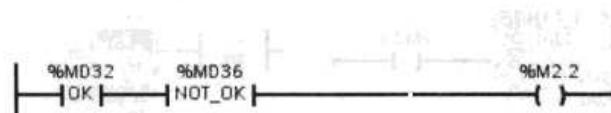


图 5-3 OK 与 NOT\_OK 指令

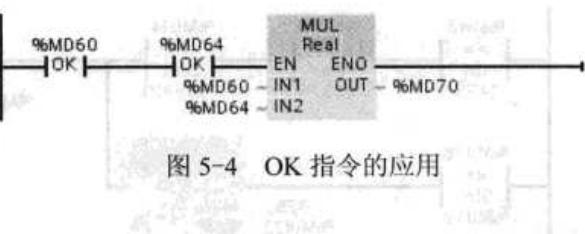


图 5-4 OK 指令的应用

## 5.1.2 使能输入与使能输出

### 1. BCD 码

BCD (Binary-coded Decimal) 是二进制编码的十进制数的缩写，BCD 码用 4 位二进制数表示一位十进制数（见表 3-1），每一位 BCD 码允许的数值范围为 2#0000~2#1001，对应于十进制数 0~9。4 位二进制数共有 16 种组合，有 6 种组合（2#1010~2#1111）没有在 BCD 码中使用。

BCD 码的最高位二进制数用来表示符号，负数为 1，正数为 0。一般令负数和正数的最高 4 位二进制数分别为 1111 和 0000（见图 5-5）。16 位 BCD 码的范围为 -999~+999，32 位 BCD 码的范围为 -9999999~+9999999（见图 5-6）。BCD 码各位之间的关系是逢十进一，图 5-5 中的 BCD 码为 -862。

15	1111	1000	0110	0010	0
符号位	百位	十位	个位		

图 5-5 3 位 BCD 码的格式

31	SXXX			16	15	0
符号位	百万位	十万位	万位	千位	百位	十位

图 5-6 7 位 BCD 码的格式

拨码开关（见图 5-7）内的圆盘圆周面上有 0~9 这 10 个数字，用按钮来增、减各位要输入的数字。它用内部硬件将 10 个十进制数转换为 4 位二进制数。PLC 用输入点读取的多位拨码开关的输出值就是 BCD 码，需要用数据转换指令 CONV 将它转换为二进制整数或双整数。

用 PLC 的 4 个输出点给译码驱动芯片 4547 提供输入信号，可以用 LED 七段显示器显示一位十进制数（见图 5-8）。需要使用数据转换指令 CONV，将 PLC 中的二进制整数或双整数转换为 BCD 码，然后分别送给各个译码驱动芯片。



图 5-7 拨码开关

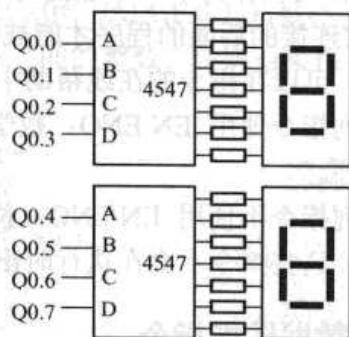


图 5-8 LED 七段显示器电路

## 2. EN 与 ENO

在梯形图中，用方框表示某些指令、功能（FC）和功能块（FB），输入信号均在方框的左边，输出信号均在方框的右边。梯形图中有一条提供“能流”的左侧垂直母线，图 5-9 中 I0.0 的常开触点接通时，能流流到方框指令 CONV 的数字量输入 EN(Enable input, 使能输入) 端，“使能”有允许的意思。使能输入端有能流时，方框指令才能执行。

如果方框指令的 EN 端有能流流入，而且执行时无错误，则使能输出 ENO(Enable Output) 端将能流传递给下一个元件（见图 5-9a）。如果执行过程中有错误，能流在出现错误的方框指令终止（见图 5-9b）。

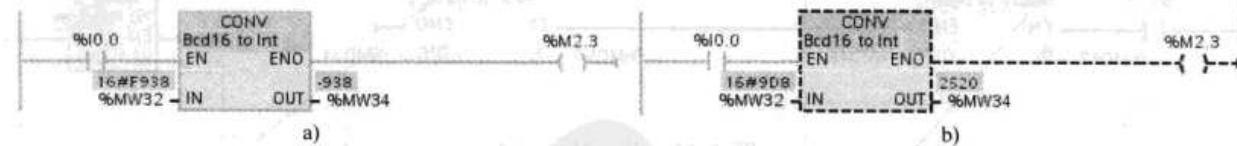


图 5-9 EN 与 ENO

图 5-9 中的方框指令 CONV 是数据转换指令。将指令列表中的 CONV 指令拖放到梯形图中时，CONV 下面的“to”两边分别有 3 个红色的问号（见图 5-10b 的指令），用来设置转换前后的数据的数据类型。点击“to”前面或后面的问号，再点击问号右边出现的 ▾ 按钮，用下拉式列表设置转换前的数据的数据类型为 16 位 BCD 码 (Bcd16)，转换后的数据的数据类型为 Int (有符号整数)。

在 RUN 模式用程序状态监控功能监视程序的运行情况。用监视表设置转换前 MW32 的值为 16#F938（见图 5-9a），最高位的“F”对应于 2#1111，表示负数。转换以后的十进制数为 -938，因为程序执行成功，有能流从 ENO 输出端流出。指令框和 ENO 输出线均为绿色的连续线。如果 ENO 端未接后续元件，该指令框为黑色，ENO 输出线为绿色。

也可以用鼠标右键点击图 5-9 中的 MW32，执行出现的快捷菜单中的“Modify”→“Modify value”命令，在出现的“Modify”对话框中设置变量的值。点击“OK”按钮确认。

设置转换前的数值为 16#9D8（见图 5-9b），BCD 码每一位的有效数字为 0~9，16#D 是非法的数字，因此程序执行出错，没有能流从 ENO 流出，指令框和 ENO 输出线均为蓝色的虚线。如果 ENO 端未接后续元件，指令框和 ENO 输出线均为灰色的实线。

ENO 可以作为下一个方框的 EN 输入，即几个方框可以串联，只有前一个方框被正确执行，与它连接的后面的程序才能被执行。EN 和 ENO 的操作数均为能流，数据类型为 Bool（布尔）型。可以在指令的在线帮助中找到使 ENO 为 0 状态的原因。

下列指令使用 EN/ENO：数学运算指令、传送与转换指令、移位与循环指令、字逻辑运算指令等。

下列指令不使用 EN/ENO：位逻辑指令、比较指令、计数器指令、定时器指令和程序控制指令。这些指令不会在执行时出现需要程序中止的错误，因此不需要使用 EN/ENO。

### 5.1.3 数据转换指令

#### 1. CONV 指令

CONV 指令的参数 IN、OUT 的数据类型可以是 Byte、Word、DWord、SInt、Int、DInt、USInt、UInt、UDInt、BCD16、BCD32 和 Real，IN 还可以是常数。

EN 输入端有能流流入时，CONV 指令将输入 IN 指定的数据转换为指定的数据类型。Bcd16 只能转换为数据类型 Int，Bcd32 只能转换为数据类型 DInt。

图 5-10 中的 I0.3 的常开触点接通时，执行 CONV 指令，将 MD42 中的 32 位 BCD 码转换为双整数后送 MD46。如果执行时没有出错，有能流从左边的 CONV 指令的 ENO 端流出。ROUND 指令将 MD50 中的实数四舍五入转换为双整数后保存在 MD54。

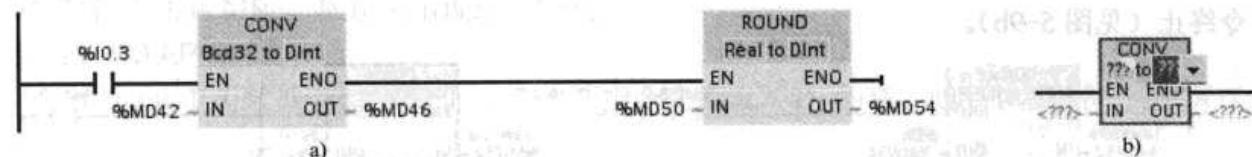


图 5-10 数据转换指令

如果输入 IN 为 INF（无穷大）或 NaN（无效的数学运算结果），或转换结果超出了 OUT 的数据类型允许的范围，ENO 为 0 状态。

#### 2. 浮点数转换为双整数的指令

浮点数转换为双整数有 4 条指令，它们将 IN 输入的浮点数转换为 32 位双整数。其中用得最多的是四舍五入的 ROUND 指令，CEIL 和 FLOOR 指令用得很少。因为转换规则不同，得到的结果也不相同，表 5-1 给出了不同的取整格式的例子。

表 5-1 不同的取整格式举例

指令	取整前	取整后	说明
ROUND	+100.6 -100.6	+101 -101	将浮点数转换为四舍五入的双整数
CEIL	+100.2 -100.6	+101 -100	将浮点数转换为大于或等于它的最小双整数
FLOOR	+100.6 -100.2	+100 -101	将浮点数转换为小于或等于它的最大双整数
TRUNC	+100.7 -100.7	+100 -100	将浮点数转换为截位取整的双整数

因为浮点数的数值范围远远大于 32 位整数，有的浮点数不能成功地转换为 32 位整数。如果被转换的浮点数超出了 32 位整数的表示范围，得不到有效的结果，ENO 为 0 状态。

### 3. SCALE\_X 指令

图 5-11 中的 SCALE\_X 指令的浮点数输入值 VALUE ( $0.0 \leqslant \text{VALUE} \leqslant 1.0$ ) 被线性转换(映射)为参数 MIN (下限) 和 MAX (上限) 定义的数值范围之间的整数。转换结果保存在 OUT 指定的地址。

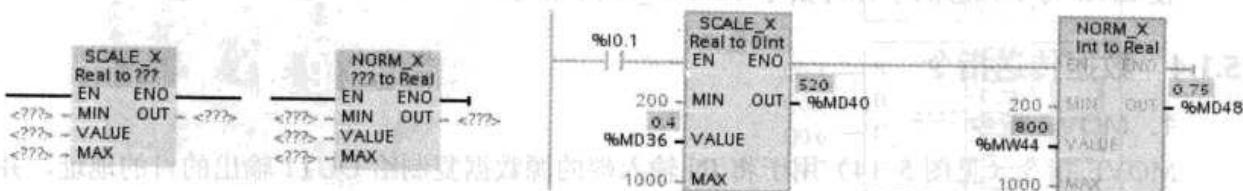


图 5-11 SCALE\_X 与 NORM\_X 指令

点击方框内指令名称下面的问号，用下拉式列表设置变量的数据类型。参数 MIN、MAX 和 OUT 的数据类型应相同，可以是 SInt、Int、DInt、USInt、UInt、UDInt 和 Real，MIN 和 MAX 可以是常数。

各变量之间的线性关系如下（见图 5-12）：

$$\text{OUT} = \text{VALUE} \times (\text{MAX} - \text{MIN}) + \text{MIN} = 0.4 \times (1000 - 200) + 200 = 520$$

如果参数 VALUE 小于 0.0 或大于 1.0，可以生成小于 MIN 或大于 MAX 的 OUT，此时 ENO 为 1。例如 VALUE 为 1.2 时，OUT 为 1160。

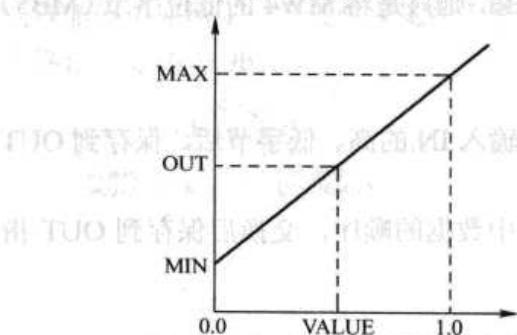


图 5-12 SCALE\_X 指令的线性关系

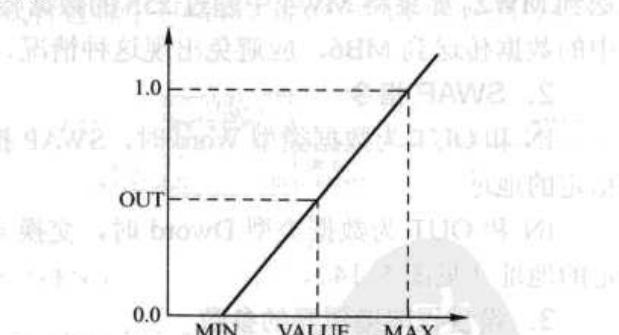


图 5-13 NORM\_X 指令的线性关系

满足下列条件之一时 ENO 为 0 状态：

- 1) EN 输入为 0 状态。
- 2) MIN 的值大于等于 MAX 的值。

- 3) 实数值超出 IEEE-754 规定的范围。
- 4) 有溢出。
- 5) 输入 VALUE 为 NaN (无效的算术运算结果)。

#### 4. NORM\_X 指令

图 5-11 中的 NORM\_X 指令的整数输入值 VALUE ( $\text{MIN} \leq \text{VALUE} \leq \text{MAX}$ ) 被线性转换 (规格化) 为 0.0~1.0 之间的浮点数, 转换结果保存在 OUT 指定的地址。

NORM\_X 的输出 OUT 的数据类型为 Real (实数), 点击方框内指令名称下面的问号, 用下拉式列表设置输入 VALUE 的数据类型。输入参数 MIN、MAX 和 VALUE 的数据类型应相同, 可以是 SInt、Int、DInt、USInt、UInt、UDInt、Real, 也可以是常数。

各变量之间的线性关系如下 (见图 5-13):

$$\text{OUT} = (\text{VALUE} - \text{MIN}) / (\text{MAX} - \text{MIN}) = (800 - 200) / (1000 - 200) = 0.75$$

如果参数 VALUE 小于 MIN 或大于 MAX, 可以生成小于 0.0 或大于 1.0 的 OUT, 此时 ENO 为 1。例如图 5-13 中的 VALUE 为 0 时, OUT 为 -0.25。

使 ENO 为 0 状态的条件与指令 SCALE\_X 的相同。

### 5.1.4 数据传送指令

#### 1. MOVE 指令

MOVE 指令 (见图 5-14) 用于将 IN 输入端的源数据复制给 OUT1 输出的目的地址, 并且转换为 OUT1 指定的数据类型, 源数据保持不变。IN 和 OUT1 可以是 Bool 之外的所有基本数据类型, 和数据类型 DTL、Struct 和 Array。IN 还可以是常数。

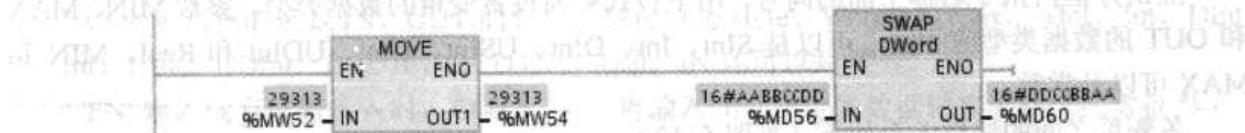


图 5-14 MOVE 与 SWAP 指令

同一条指令的输入参数和输出参数的数据类型可以不相同, 例如可以将 MB0 中的数据传送到 MW2。如果将 MW4 中超过 255 的数据传送到 MB6, 则只是将 MW4 的低位字节 (MB5) 中的数据传送到 MB6, 应避免出现这种情况。

#### 2. SWAP 指令

IN 和 OUT 为数据类型 Word 时, SWAP 指令交换输入 IN 的高、低字节后, 保存到 OUT 指定的地址。

IN 和 OUT 为数据类型 Dword 时, 交换 4 个字节中数据的顺序, 交换后保存到 OUT 指定的地址 (见图 5-14)。

#### 3. 设置程序编辑器的参数

用菜单命令 “Option” → “Settings” 打开 “Settings” 编辑器 (见图 5-15), 选中工作区左边窗口中的 “PLC Programming” 文件夹的 “General” 组, 如果选中了右边窗口的复选框 “IEC Check”, 启动 IEC 检查后, MOVE 指令的输入 IN 与输出 OUT1 的数据类型必须相同。

“Mnemonic” 选择框用来选择使用英文助记符 (International) 或是德文的助记符。

选中“Settings”对话框左边窗口的“LAD/FBD”组，右边窗口的“Size”选择框（见图 5-15 的右图）用来设置程序编辑器中的字体的大小。

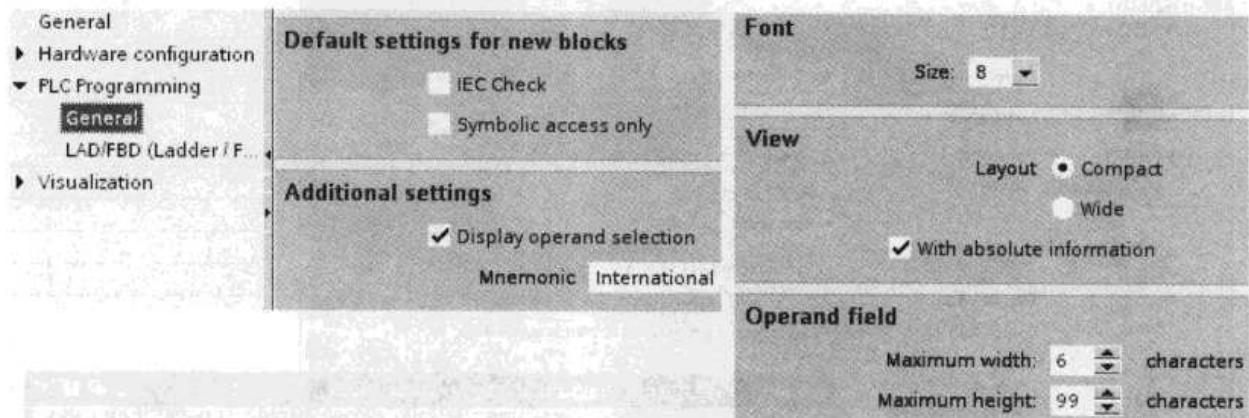


图 5-15 程序编辑器的参数设置

“Compact”（紧凑）和“Wide”（宽）用来设置操作数和其他对象（例如操作数与触点）之间的垂直间距。需要重新打开块，修改的设置才起作用。

“Operand field”（操作数域）的“Maximum width”（最大宽度）和“Maximum height”（最大高度），分别是操作数域水平方向和垂直方向可以输入的最大字符数。

如果操作数域的最大宽度设置过小，有的方框指令内部的空间不够用，该方框指令将自动将方框的宽度加倍。

#### 4. 全局数据块与数组

块传送指令用于传送数据块中的数组的多个元素。为此首先应生成全局数据块和数组。数组由相同数据类型的多个元素组成，数组元素的数据类型可以是所有的基本数据类型。

点击项目树中某个 PLC 的“Program block”文件夹中的“Add new block”，添加一个新的块。在打开的对话框中（见图 5-16 中的大图），点击“Data block”按钮，生成一个数据块，可以修改其名称或采用默认的名称，其类型为默认的“Global DB”（全局数据块），生成方式为默认的“Automatic”（自动）。点击“OK”按钮后自动生成数据块。

如果用单选框选中“Manual”（手动），可以修改块的编号。选中复选框“Symbolic accesses only”，只能用符号地址访问生成的块中的变量，不能使用绝对地址。这种访问方式可以提高存储器的利用率。

选中下面的复选框“Add new and open”，生成新的块之后，将会自动打开它。

在数据块的第 2 行的“Name”列（见图 5-16 中的小图），输入数组（Array）的名称“Source”，点击“Datatype”列中的按钮，选中下拉式列表中的数据类型“Array[lo..hi] of type”。其中的“lo (low)”和“hi (high)”分别是数组元素的编号（下标）的上限值和下限值，最大范围为[-32768..32767]，下限值应小于等于上限值。

将“Array[lo..hi] of type”改为“Array[0..99] of Int”（见图 5-17），其元素的数据类型为 Int，元素的编号为 0~99。S7-1200 只能生成一维数组。

用同样的方法，生成数据块 DB4，在 DB4 中生成有 100 个 Int 元素的数组 Distin。

在用户程序中，可以用符号地址“DB\_1”.Source[2]或绝对地址 DB3.DBW4 访问数组中下标为 2 的元素。

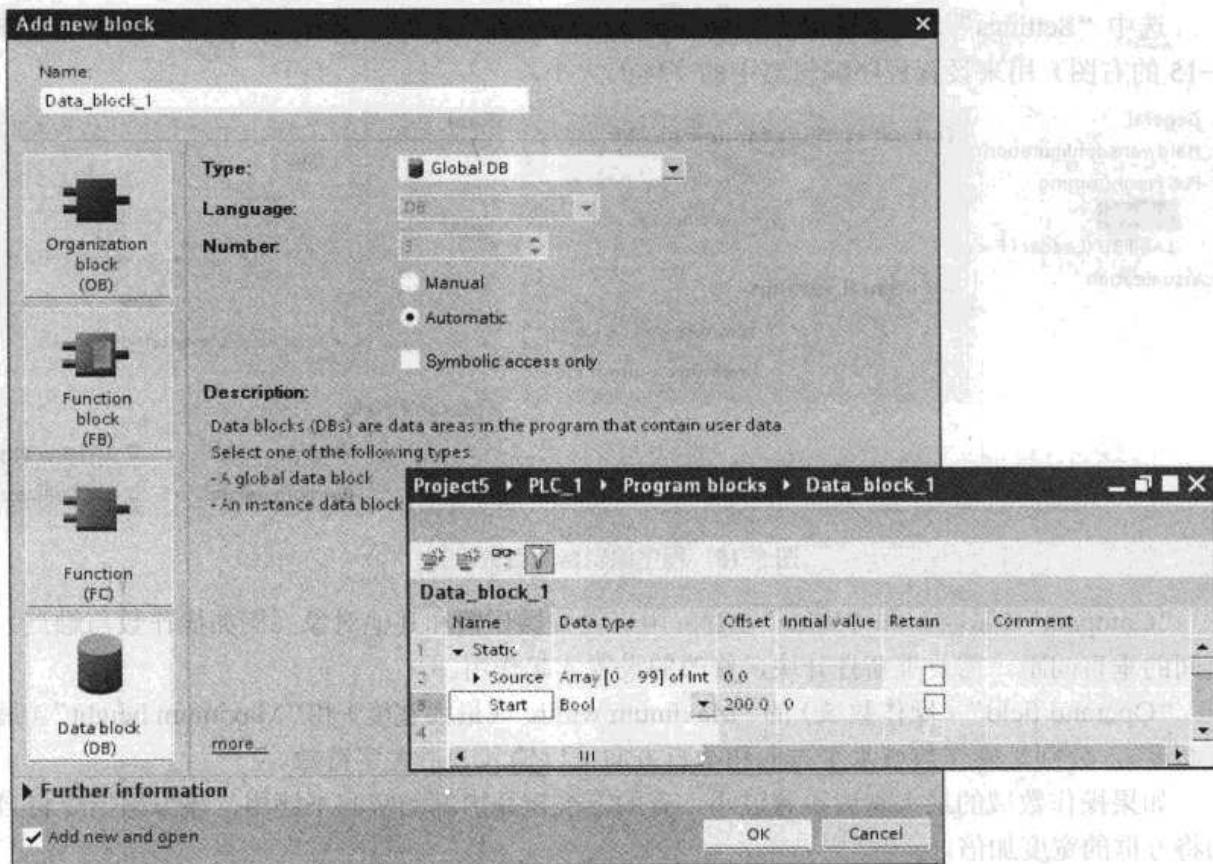


图 5-16 添加数据块与数据块中的数组

Name	Data type	Offset	Default value	Initial value	Retain
1 Static					<input type="checkbox"/>
2 Source	Array [0..99] of Int	0			<input type="checkbox"/>
3 Source[0]	Int			0	
4 Source[1]	Int			0	
5 Source[2]	Int			0	
6 Source[3]	Int			0	

图 5-17 数组的扩展模式

## 5. FILL\_BLK 与 UFILL\_BLK 指令

FILL\_BLK 指令将输入参数 IN 设置的值填充到输出参数 OUT 指定起始地址的目标数据区（见图 5-18），IN 和 OUT 必须是 D、L（数据块或局部数据区）中的数组元素，IN 还可以是常数。COUNT 为填充的数组元素的个数，数据类型为 DInt 或常数。图 5-18 中 I0.4 的常开触点接通时，常数 3527 被填充到 DB3 的 DBW0 开始的 20 个字节中。

FILL\_BLK 与 UFILL\_BLK 指令的功能基本上相同，其区别在于后者的填充操作不会被其他操作系统的任务打断。执行该指令时，CPU 的报警响应时间将会增大。

值得注意的是指令 UFILL\_BLK 的起始地址 DB3.DBW40 中的 40 是数据块中字节的编号，而输入参数 COUNT 是以字为单位的数组元素的个数。指令 FILL\_BLK 已占用了 40B（即 20 个字）的数据，因此 UFILL\_BLK 指令的输出 OUT 指定的地址区从 DBW40 开始。

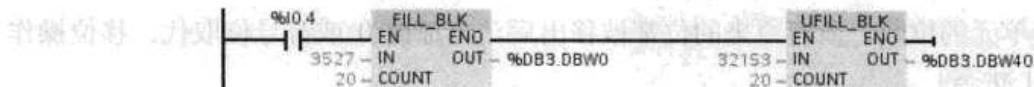


图 5-18 数据填充指令

## 6. MOVE\_BLK 与 UMOVE\_BLK 指令

图 5-19 中的 MOVE\_BLK 指令用于将数据块 DB3 中的数组 Source 的 0 号元素开始的 20 个 Int 元素的值，复制给数据块 DB4 的数组 Distin 的 0 号元素开始的 20 个元素。COUNT 为要传送的数组元素的个数，复制操作按地址增大的方向进行。

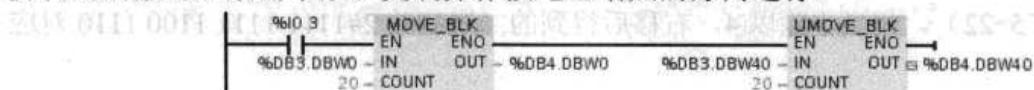


图 5-19 数据块传送指令

除了 IN 不能取常数外，指令 MOVE\_BLK 和 FILL\_BLK 的参数的数据类型和存储区基本上相同。

指令 UMOVE\_BLK 与 MOVE\_BLK 的功能基本上相同（见图 5-19），其区别在于前者的复制操作不会被其他操作系统的任务打断。执行该指令时，CPU 的报警响应时间将会增大。

## 7. 块填充与块传送指令的实验

将图 5-18 和图 5-19 中的程序下载到 CPU，切换到 RUN 模式后，双击打开指令树中的 DB3 和 DB4。单击工具栏上的 按钮，启动扩展模式，显示各数组元素。点击 按钮，启动监视，Monitor value 列是 CPU 中的变量值。

因为没有设置保持（Retain）功能，数组元素的初始值均为 0，此时 DB3 和 DB4 的各数组元素的值均为 0。

接通 I0.4 的常开触点，FILL\_BLK 与 UFILL\_BLK 指令被执行，DB3 中的数组元素 Source[0]~Source[19] 被填充数据 3527，Source[20]~Source[39] 被填充数据 32153。

接通 I0.3 的常开触点，FILL\_BLK 与 UFILL\_BLK 指令被执行，DB3 中的数组 Source 的前 40 个元素被传送给 DB4 中的数组 Distin 的前 40 个元素。图 5-20 是传送给符号名为 DB\_2 的 DB4 的部分数据。

DB_2						
Name	Data type	Offset	Default value	Initial value	Monitor value	Retain
21 Distin[18]	Int		0	3527		
22 Distin[19]	Int		0	3527		
23 Distin[20]	Int		0	32153		
24 Distin[21]	Int		0	32153		

图 5-20 数据块中的部分数据

## 5.1.5 移位与循环移位指令

### 1. 移位指令

移位指令 SHR 和 SHL 将输入参数 IN 指定的存储单元的整个内容逐位右移或左移若干位，移位的位数用输入参数 N 来定义，移位的结果保存在输出参数 OUT 指定的地址。

无符号数移位和有符号数左移后空出来的位用 0 填充。有符号数右移后空出来的位用符号位（原来的最高位）填充，正数的符号位为 0，负数的符号位为 1。

N 为 0 时不会移位，但是 IN 指定的输入值被复制给 OUT 指定的地址。如果移位位数 N

大于被移位存储单元的位数，所有原来的位都被移出后，全部被 0 或符号位取代。移位操作的 ENO 总是为 1 状态。

将指令列表中的移位指令拖放到梯形图后，点击方框内指令名称下面的问号，用下拉式列表设置变量的数据类型。

如果移位后的数据要送回原地址，应将图 5-21 中 I0.5 的常开触点改为 I0.5 的上升沿检测触点（P 触点），否则在 I0.5 为 1 的每个扫描周期都要移位一次。

右移  $n$  位相当于除以  $2^n$ ，例如将十进制数 -200 对应的二进制数 2#1111 1111 0011 1000 右移 2 位（见图 5-22），相当于除以 4，右移后得到的二进制数 2#1111 1111 1100 1110 对应于十进制数 -50。

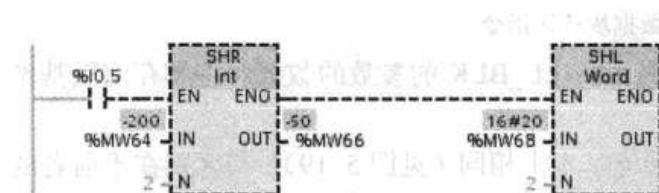


图 5-21 移位与循环指令

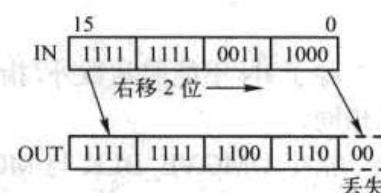


图 5-22 数据的右移

左移  $n$  位相当于乘以  $2^n$ ，例如将 16#20 左移 2 位，相当于乘以 4，左移后得到的十六进制数为 16#80（见图 5-21）。

## 2. 循环移位指令

循环移位指令 ROR 和 ROL 将输入参数 IN 指定的存储单元的整个内容逐位循环右移或循环左移若干位，即移出来的位又送回存储单元另一端空出来的位，原始的位不会丢失。移位的位数用输入参数  $N$  来定义，移位的结果保存在输出参数 OUT 指定的地址。 $N$  为 0 时不会移位，但是 IN 指定的输入值复制给 OUT 指定的地址。移位位数  $N$  可以大于被移位存储单元的位数，执行指令后，ENO 总是为 1 状态。

## 3. 使用循环移位指令的彩灯控制器

在图 5-23 的 8 位循环移位彩灯控制程序中，QB0 是否移位用 I0.6 来控制，移位的方向用 I0.7 来控制。为了获得移位用的时钟脉冲和首次扫描脉冲，双击 PLC 文件夹中的“Device configuration”，选中 CPU 后，打开下面的监视窗口的“Properties”选项卡，选中左边的“System and clock memory”，激活右边窗口的复选框（见图 2-32），系统存储器字节地址和时钟脉冲地址分别是默认的 MB1 和 MB0。时钟脉冲位 M0.5 的频率为 1Hz。

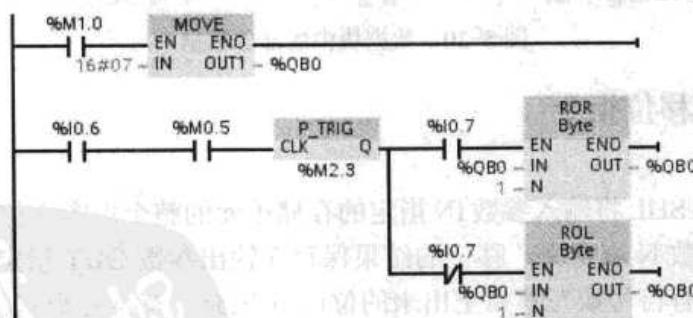


图 5-23 使用循环移位指令的彩灯控制器

PLC 首次扫描时 M1.0 的常开触点接通, MOVE 指令给 QB0 (Q0.0~Q0.7) 置初值 7, 其低 3 位被置为 1。

输入、下载和运行彩灯控制程序, 通过观察 CPU 模块上与 Q0.0~Q0.7 对应的 LED (发光二极管), 观察彩灯的运行效果。

I0.6 为 1 状态时, 在时钟脉冲 M0.5 的上升沿, 指令 P\_TRIG 输出一个扫描周期的脉冲。如果此时 I0.7 为 1 状态, 执行一次 ROR 指令, QB0 的值循环右移 1 位。如果 I0.7 为 0 状态, 执行一次 ROL 指令, QB0 的值循环左移 1 位。表 5-2 是 QB0 循环移位前后的数据。因为 QB0 循环移位后的值又送回 QB0, 循环移位指令的前面必须使用 P\_TRIG 指令, 否则每个扫描循环周期都要执行一次循环移位指令, 而不是每秒钟移位一次。

表 5-2 QB0 循环移位前后的数据

内 容	循 环 左 移	循 环 右 移
移位前	0000 0111	0000 0111
第 1 次移位后	0000 1110	1000 0011
第 2 次移位后	0001 1100	1100 0001
第 3 次移位后	0011 1000	1110 0000

## 5.2 数学运算指令与逻辑运算指令

### 5.2.1 数学运算指令

数学运算指令包括数学运算指令 (见表 5-3)、浮点数函数运算指令和逻辑运算指令。

表 5-3 数学运算指令

梯 形 图	描 述	梯 形 图	描 述
ADD	$IN1 + IN2 = OUT$	INC	将参数 IN/OUT 的值加 1
SUB	$IN1 - IN2 = OUT$	DEC	将参数 IN/OUT 的值减 1
MUL	$IN1 * IN2 = OUT$	ABS	求有符号整数和实数的绝对值
DIV	$IN1 / IN2 = OUT$	MIN	求两个输入中较小的数
MOD	求双整数除法的余数	MAX	求两个输入中较大的数
NEG	将输入值的符号取反	LIMIT	将输入 IN 的值限制在指定的范围内

#### 1. 四则运算指令

数学运算指令中的 ADD、SUB、MUL 和 DIV 分别是加、减、乘、除。

加减法指令:  $IN1 + IN2 = OUT$ ,  $IN1 - IN2 = OUT$ 。

乘除法指令:  $IN1 * IN2 = OUT$ ,  $IN1 / IN2 = OUT$ 。

数据类型可选 SInt、Int、DInt、USInt、UInt、UDInt、Real 和 LReal, IN1 和 IN2 可以是常数。IN1、IN2 和 OUT 的数据类型应该相同。

整数除法指令将得到的商截位取整后, 作为整数格式的输出 OUT。

有下列情况之一时 ENO 为 0 状态:

1) 运算结果超出了选择的数据类型允许的范围。

- 2) 除法运算的除数 (IN2) 为 0。
- 3) 实数运算时一个输入值为 NaN (不是数字) 或运算结果为 INF (无穷大), 返回 NaN。
- 4) 实数的加法运算的两个输入值分别为 -INF 和 +INF, 属于非法的运算, 返回 NaN。
- 5) 实数的减法运算的两个输入值为符号相同的 INF, 属于非法的运算, 返回 NaN。
- 6) 实数的乘法运算的两个输入值分别为 0 和 INF, 属于非法的运算, 返回 NaN。
- 7) 实数的除法运算的两个输入值均为 0 或均为 INF, 属于非法的运算, 返回 NaN。

**【例 5-2】** 压力变送器的量程为 0~10 MPa, 输出信号为 0~10V, 被 CPU 集成的模拟量输入的通道 0 (地址为 IW64) 转换为 0~27648 的数字。假设转换后的数字为  $N$ , 试求以 kPa 为单位的压力值。

解: 0~10MPa (0~10000kPa) 对应于转换后的数字 0~27648, 转换公式为

$$P = (10000 \times N) / 27648 \text{ kPa} \quad (5-1)$$

值得注意的是, 在运算时一定要先乘后除, 否则会损失原始数据的精度。

公式中乘法运算的结果可能会大于一个字能表示的最大值, 因此应使用数据类型为双整数的乘法和除法 (见图 5-24)。为此首先使用 CONV 指令, 将 IW64 转换为双整数 (DInt)。

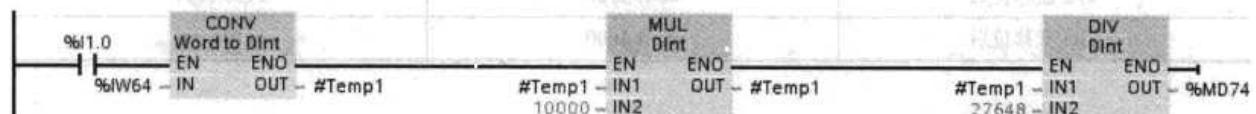


图 5-24 压力测量值计算程序

将指令列表中的 MUL 和 DIV 指令拖放到梯形图中后, 点击指令方框内指令名称下面的问号, 再点击出现的 **▼** 按钮, 用下拉式列表框设置操作数的数据类型为双整数 DInt。

在 OB1 的接口区定义数据类型为 DInt 的局部变量 Temp1, 用来保存运算的中间结果。

双字除法指令 DIV 的运算结果为双字, 但是由式 (5-1) 可知运算结果实际上不会超过 16 位正整数的最大值 32767, 所以双字 MD74 的高位字 MW74 为 0, 运算结果的有效部分在 MD74 的低位字 MW76 中。

**【例 5-3】** 使用浮点数运算计算上例以 kPa 为单位的压力值。将式 (5-1) 改写为式 (5-2):

$$P = (10000 \times N) / 27648 = 0.361690 \times N \text{ kPa} \quad (5-2)$$

在 OB1 的接口区定义数据类型为 Real 的局部变量 Temp2, 用来保存运算的中间结果。

首先用 CONV 指令将 IW64 转换为实数 (Real), 再用实数乘法指令完成式 (5-2) 的运算 (见图 5-25)。最后使用四舍五入的 ROUND 指令, 将运算结果转换为整数。

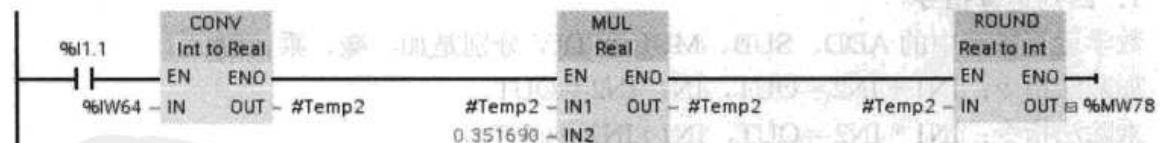


图 5-25 使用浮点数运算指令的压力测量值计算程序

## 2. 其他整数数学运算指令

### (1) MOD 指令

除法指令只能得到商, 余数被丢掉。可以用 MOD 指令来求除法的余数 (见图 5-26)。输出 OUT 中的运算结果为除法运算 IN1 / IN2 的余数。

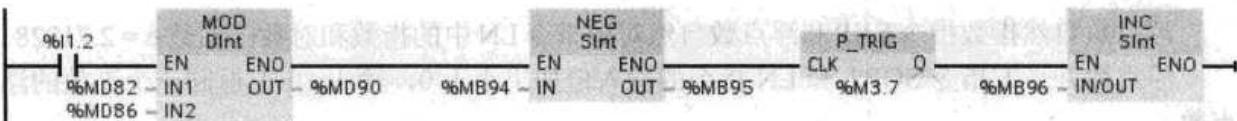


图 5-26 整数运算指令

### (2) NEG 指令

NEG (negation) 将输入 IN 的值的符号取反后，保存在输出 OUT 中。IN 和 OUT 的数据类型可以是 SInt、Int、DInt 和 Real，输入 IN 还可以是常数。

### (3) INC 与 DEC 指令

执行指令 INC 与 DEC 时，参数 IN/OUT 的值分别被加 1 和减 1。IN/OUT 的数据类型可选 SInt、USInt、Int、UInt、DInt 和 UDInt (有符号或无符号的整数)。

如果图 5-26 中的 INC 指令用来计 I1.2 动作的次数，应在 INC 指令之前添加检测能流的上升沿的 P\_TRIG 指令。否则在 I1.2 为 1 状态的每个扫描循环周期，MB96 都要加 1。

### (4) 绝对值指令 ABS

ABS 指令 (见图 5-27) 用来求输入 IN 中的有符号整数 (SInt、Int、DInt) 和实数 (Real) 的绝对值，将结果保存在输出 OUT 中。IN 和 OUT 的数据类型应相同。

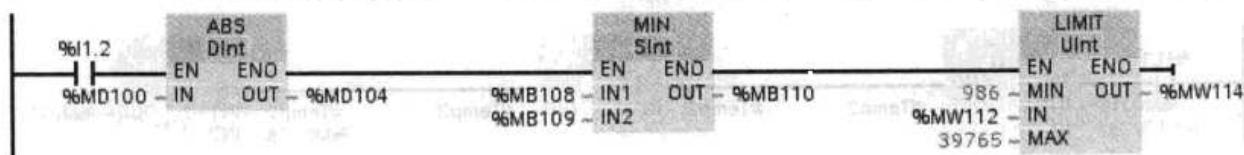


图 5-27 整数运算指令

### (5) MIN 与 MAX 指令

MIN (minimum) 指令比较输入 IN1 和 IN2 的值，将其中较小的值送给输出 OUT。

MAX (maximum) 指令比较输入 IN1 和 IN2 的值，将其中较大的值送给输出 OUT。

IN1 和 IN2 的数据类型相同才能执行指定的操作。

### (6) LIMIT 指令

LIMIT 指令检查输入 IN 的值是否在参数 MIN 和 MAX 指定的范围内，如果 IN 的值没有超出该范围，将它直接保存在 OUT 指定的地址中。如果 IN 的值小于 MIN 的值或大于 MAX 的值，将后者送给输出 OUT。

## 3. 浮点数函数运算指令

浮点数 (实数) 数学运算指令 (见表 5-4) 的操作数 IN 和 OUT 的数据类型为 Real。

表 5-4 浮点数函数运算指令

梯形图	描述	表达式	梯形图	描述	表达式
SQR	求浮点数的平方	$IN^2 = OUT$	TAN	求浮点数的正切函数	$\tan(IN) = OUT$
SQRT	求浮点数的平方根	$\sqrt{IN} = OUT$	ASIN	求浮点数的反正弦函数	$\text{arc sin}(IN) = OUT$
LN	求浮点数的自然对数	$\text{LN}(IN) = OUT$	ACOS	求浮点数的反余弦函数	$\text{arc cos}(IN) = OUT$
EXP	求浮点数的自然指数	$e^{IN} = OUT$	ATAN	求浮点数的反正切函数	$\text{arc tan}(IN) = OUT$
SIN	求浮点数的正弦函数	$\sin(IN) = OUT$	FRAC	求浮点数的小数部分	—
COS	求浮点数的余弦函数	$\cos(IN) = OUT$	EXPT	求浮点数的普通对数	$IN1^{IN2} = OUT$

浮点数自然指数指令 EXP 和浮点数自然对数指令 LN 中的指数和对数的底数  $e = 2.71828$ 。浮点数开平方指令 SQRT 和 LN 指令的输入值如果小于 0，输出 OUT 返回一个无效的浮点数。

浮点数三角函数指令和反三角函数指令中的角度均为以弧度为单位的浮点数。如果输入值是以度为单位的浮点数，使用三角函数指令之前应先将角度值乘以  $\pi/180.0$ ，转换为弧度值。

浮点数反正弦函数指令 ASIN 和浮点数反余弦函数指令 ACOS 的输入值的允许范围为  $-1 \sim 1$ ，ASIN 和 ATAN 的运算结果的取值范围为  $-\pi/2 \sim +\pi/2$  弧度，ACOS 的运算结果的取值范围为  $0 \sim \pi$  弧度。

求以 10 为底的对数时，需要将自然对数值除以 2.302585（10 的自然对数值）。例如  $\lg 100 = \ln 100 / 2.302585 = 4.605170 / 2.302585 = 2$ 。

EXPT 的底数 IN1 和 OUT 的数据类型为 Real，指数 IN2 可以选 7 种数据类型，点击方框内指令名称下面的问号，用下拉式列表设置 IN1 和 IN2 的数据类型。

**【例 5-4】** 测量远处物体的高度时，已知被测物体到测量点的距离  $L$  和以度为单位的夹角  $\theta$ ，求被测物体的高度  $H$ ， $H = L \tan \theta$ ，角度的单位为度。假设以度为单位的实数角度值在 MD116，乘以  $\pi/180 = 0.0174533$  得角度的弧度值（见图 5-28），运算的中间结果保存在数据类型为 Real 的局部变量 Temp2 中。 $L$  的实数值在 MD128，运算结果在 MD132。

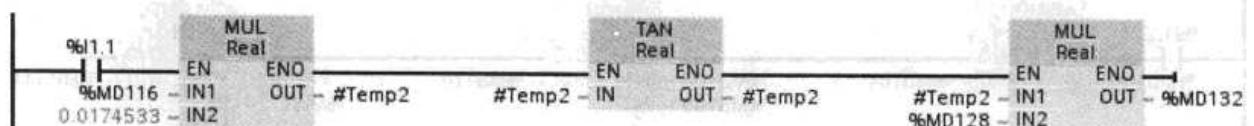


图 5-28 函数运算程序

## 5.2.2 逻辑运算指令

### 1. 逻辑运算指令

逻辑运算指令对两个输入 IN1 和 IN2 逐位进行逻辑运算。逻辑运算的结果存放在输出 OUT 指定的地址（见图 5-29）。

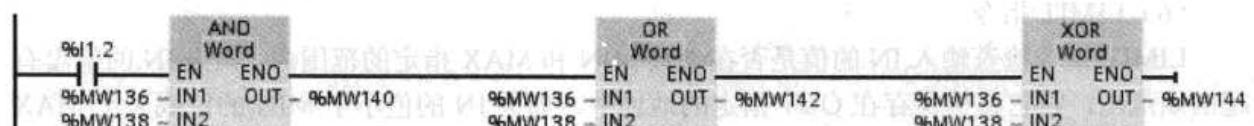


图 5-29 逻辑运算指令

“与”(AND) 运算时两个操作数的同一位如果均为 1，运算结果的对应位为 1，否则为 0（见表 5-5）。

“或”(OR) 运算时两个操作数的同一位如果均为 0，运算结果的对应位为 0，否则为 1。

“异或”(XOR) 运算时两个操作数的同一位如果不相同，运算结果的对应位为 1，否则为 0。以上指令的操作数 IN1、IN2 和 OUT 的数据类型为十六进制的 Byte、Word 和 DWord。

表 5-5 字逻辑运算的结果

参数	数值
IN1	0101 1001 0011 1011
IN2 或 INV 指令的 IN	1101 0100 1011 0101
AND 指令的 OUT	0101 0000 0011 0001
OR 指令的 OUT	1101 1101 1011 1111
XOR 指令的 OUT	1000 1101 1000 1110
INV 指令的 OUT	0010 1011 0100 1010

取反指令 INV (见图 5-30) 将输入 IN 中的二进制整数逐位取反，即各位的二进制数由 0 变 1，由 1 变 0，运算结果存放在输出 OUT 指定的地址。

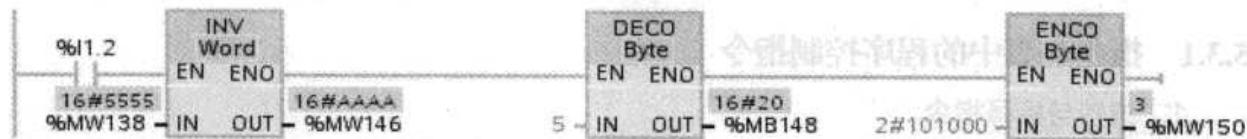


图 5-30 逻辑运算指令

## 2. 解码与编码指令

假设输入参数 IN 的值为  $n$ ，解码（译码）指令 DECO (Decode) 将输出参数 OUT 的第  $n$  位置位为 1，其余各位置 0，相当于数字电路中译码电路的功能。利用解码指令，可以用输入 IN 的值来控制 OUT 中某一位的状态。

如果输入 IN 的值大于 31，将 IN 的值除以 32 以后，用余数来进行解码操作。

IN 的值为 0~7 (3 位二进制数) 时，输出 OUT 的数据类型为 8 位的字节。

IN 的值为 0~15 (4 位二进制数) 时，输出 OUT 的数据类型为 16 位的字。

IN 的值为 0~31 (5 位二进制数) 时，输出 OUT 的数据类型为 32 位的双字。

IN 的值为 5 时 (见图 5-30)，OUT 为 2#0010 0000 (16#20)，仅第 5 位为 1。

编码指令 ENCO(Encode)与解码指令相反，将 IN 中为 1 的最低位的位数送给输出参数 OUT 指定的地址，IN 的数据类型可选 Byte、Word 和 DWord，OUT 的数据类型为 Int。

如果 IN 为 2#0010 1000 (见图 5-30)，OUT 指定的 MW150 中的编码结果为 3。如果 IN 为 1 或 0，MW150 的值为 0。如果 IN 为 0，ENO 为 0 状态。

## 3. SEL 与 MUX 指令

指令 SEL (Select) 的 Bool 输入参数 G 为 0 时选中 IN0 (见图 5-31)，G 为 1 时选中 IN1，并将它们保存到输出参数 OUT 指定的地址。

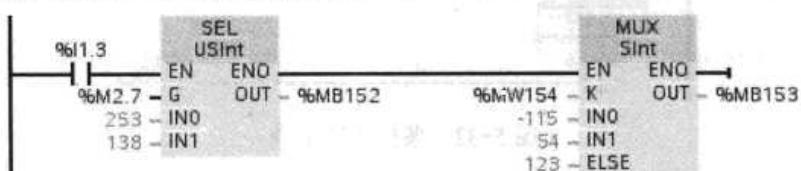


图 5-31 SEL 与 MUX 指令

指令 MUX (Multiplex，多路开关选择器) 根据输入参数  $K$  的值，选中某个输入数据，并将它传送到输出参数 OUT 指定的地址。 $K = m$  时，将选中输入参数 INm。如果  $K$  的值超过允许的范围，将选中输入参数 ELSE。

将 MUX 指令拖放到程序编辑器时，它只有 IN0、IN1 和 ELSE。用鼠标右键点击该指令，执行出现的快捷菜单中的指令“Insert input”(插入输入)，可以增加一个输入。反复使用这一方法，可以增加多个输入。增添输入后，用右键点击某个输入 INn 从方框伸出的水平短线，执行出现的快捷菜单中的指令“Delete”(删除)，可以删除选中的输入。删除后自动调整剩下的输入 INn 的编号。

参数  $K$  的数据类型为 UInt，INn、ELSE 和 OUT 可以取 12 种数据类型，它们的数据类型应相同。

## 5.3 程序控制指令

### 5.3.1 指令列表中的程序控制指令

#### 1. 跳转与标号指令

没有执行跳转指令和循环指令时，各个网络按从上到下的先后顺序执行，这种执行方式称为线性扫描。跳转指令中止程序的线性扫描，跳转到指令中的地址标号所在的目的地址。跳转时不执行跳转指令与标号之间的程序，跳到目的地址后，程序继续按线性扫描的方式顺序执行。跳转指令可以往前跳，也可以往后跳。

只能在同一个代码块内跳转，即跳转指令与对应的跳转目的地址应在同一个代码块内。在一个块内，同一个跳转目的地址只能出现一次。

如果跳转条件满足（图 5-32 中 M2.5 的常开触点闭合），跳转指令 JMP（Jump）的线圈通电（跳转线圈为绿色），跳转被执行，将跳转到指令给出的标号（Label）W1234 处，执行标号之后的第一条指令。被跳过的程序段的指令没有被执行，这些程序段的梯形图为灰色。标号在网络的开始处，标号的第一个字符必须是字母，其余的可以是字母、数字和下划线。如果跳转条件不满足，将继续执行下一个网络的程序。

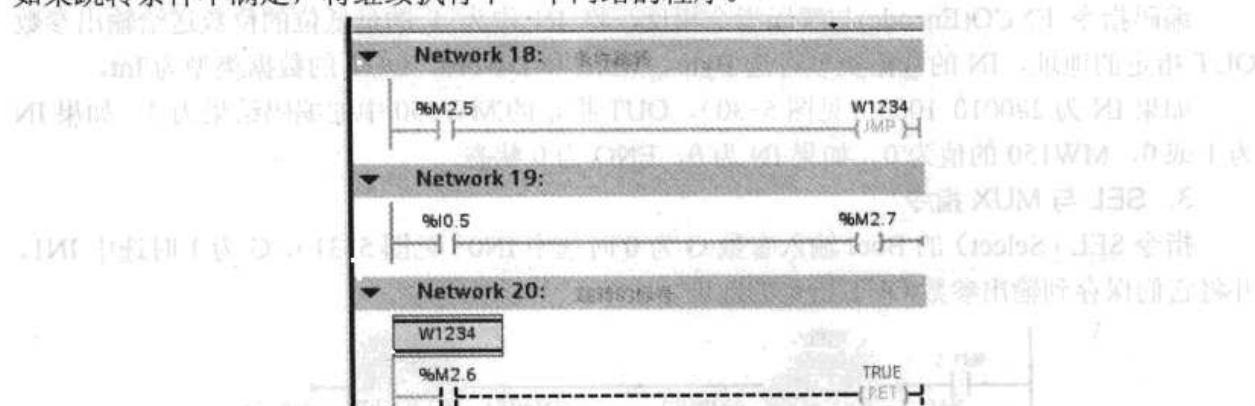


图 5-32 条件跳转指令

JMP 指令直接与右边的垂直电源线相连时，为无条件跳转指令，执行该指令后马上跳转到指令给出的标号处。

JMPN 指令的线圈断电时，将跳转到指令给出的标号处，执行标号之后的第一条指令。

#### 2. 返回指令 RET

RET 指令的线圈通电时，停止执行当前的块，不再执行该指令后面的指令，返回调用它的块后，执行调用指令之后的指令。RET 指令的线圈断电时，继续执行它下面的指令。

RET 线圈的上面是块的返回值，数据类型为 Bool。如果当前的块是 OB，返回值被忽略。如果当前的块是 FC 或 FB，返回值作为 FC 或 FB 的 ENO 的值传送给调用它的块。

一般情况并不需要在块结束时使用 RET 指令来结束块，操作系统将会自动地完成这一任务。RET 指令用来有条件地结束块，一个块可以使用多条 RET 指令。

### 5.3.2 扩展指令列表中的程序控制指令

本小节中的指令在右边的任务卡的“Extended instructions”（扩展指令）窗口的文件夹

“Program control” 中。

### 1. RE\_TRIGR 指令

监控定时器又称看门狗 (Watchdog)，每次扫描循环它都被自动复位一次，正常工作时最大扫描循环时间小于监控定时器的时间设定值，它不会起作用。

以下情况扫描循环时间可能大于监控定时器的设定时间，监控定时器将会起作用：

- 1) 用户程序很长。
- 2) 一个扫描循环内执行中断程序的时间很长。
- 3) 循环指令执行的时间太长。

可以在程序中的任意位置使用指令 RE\_TRIGR (重新触发循环时间监视)，来复位监控定时器 (见图 5-33)。该指令仅在优先级为 1 的程序循环 OB 和它调用的块中起作用；该指令在 OB80 中将被忽略。如果在优先级较高的块中 (例如硬件中断、诊断中断和循环中断 OB) 调用该指令，使能输出 ENO 被置为 0，不执行该指令。

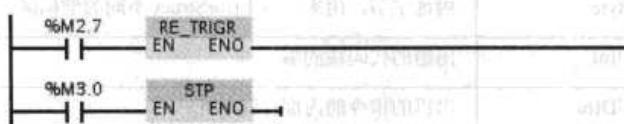


图 5-33 RE\_TRIGR 与 STP 指令

在组态 CPU 时，可以用参数 “Cycle time” 设置最大扫描循环时间，默认值为 150ms。

### 2. STP 指令

STP 指令的 EN 输入为 1 状态时，使 PLC 进入 STOP 模式。STP 指令使 CPU 集成的输出、信号板和信号模块的数字量输出或模拟量输出进入组态时设置的安全状态。可以使输出冻结在最后的状态，或用替代值设置为安全状态。默认的数字量输出状态为 FALSE，默认的模拟量输出值为 0。

### 3. GET\_ERROR 与 GET\_ERR\_ID 指令

GET\_ERROR 指令用来提供有关程序块执行错误的信息。在使用 GET\_ERROR 与 GET\_ERR\_ID 指令之前，应使用鼠标右键点击代码块，执行出现的快捷菜单中的“Properties”(属性)命令，选中打开的对话框左边的 Attributes (属性)，激活右边的复选框“Handle errors within block” (处理块中的错误，见图 5-34)。

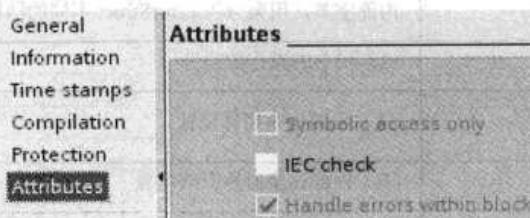


图 5-34 设置代码块的属性

GET\_ERROR 指令用输出参数 ERROR (错误) 显示发生的程序块执行错误 (见图 5-35)，并且将详细的错误信息填入预定义的 ErrorStruct (错误结构) 数据类型 (见表 5-6)。可以用程序来分析错误信息，并作出适当的响应。第一个错误消失时，指令输出下一个错误的信息。

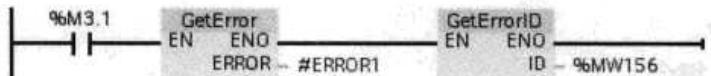


图 5-35 读取错误信息的指令

表 5-6 数据类型 ErrorStruct 的结构

结构元素	数据类型	描述					
ERROR_ID	Word	错误 ID					
FLAGS	Byte	16#01: 块调用时出错, 16#00: 块调用时没有出错					
REACTION	Byte	默认的反应: 0 为写错误, 忽略; 1 为读错误, 继续使用替代值 0; 2 为系统错误, 跳过指令					
BLOCK_TYPE	Byte	出现错误的块的类型: 1 为 OB, 2 为 FC, 3 为 FB					
PAD_0	Byte	内部字节, 用来分隔 ErrorStruct 不同的结构区, 其内容无关紧要					
CODE_BLOCK_NUMBER	UInt	出错的代码块的编号					
ADDRESS	UDInt	出错的指令的内部存储单元					
MODE	Byte	访问模式: 取决于访问的类型, 可能输出下面的信息					
		模式	(A)	(B)	(C)	(D)	(E)
		0					
		1					偏移量
		2			区域		
		3	位置	范围		DB 编号	
		4			区域		偏移量
		5			区域	DB 编号	偏移量
		6	指针编号/Acc		区域	DB 编号	偏移量
		7	指针编号/Acc	槽编号/范围	区域	DB 编号	偏移量
PAD_1	Byte	内部字节, 用来分隔 ErrorStruct 不同的结构区, 其内容无关紧要					
OPERAND_NUMBER	UInt	内部指令的操作数编号					
POINTER_NUMBER_LOCATION	UInt	(A) 内部指令指针位置					
SLOT_NUMBER_SCOPE	UInt	(B) 内部存储器的存储位置					
AREA	Byte	(C) 出现错误的存储区。L: 16#40~4E、86、87、8E、8F、C0~CE; I: 16#81; Q: 16#82; M: 16#83; DB: 16#84、85、8A、8B					
PAD_2	Byte	内部字节, 用来分隔 ErrorStruct 不同的结构区, 其内容无关紧要					
DB_NUMBER	UInt	(D) 出现错误时的数据块编号, 未用数据块时为 0					
OFFSET	UDInt	(E) 出现错误时的位偏移量, 例如 12 为字节 1 的第 4 位。					

在块的接口区定义一个名为 ERROR1 的变量 (见图 5-36) 来作参数 ERROR 的实参, 用

下拉式列表设置其数据类型为 ErrorStruct。也可以在数据块中定义 ERROR 的实参。

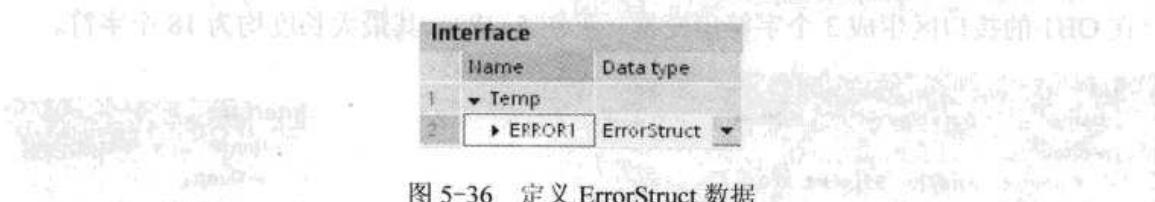


图 5-36 定义 ErrorStruct 数据

GET\_ERR\_ID 指令用来报告错误的 ID（标识符）。如果块执行时出现错误，且指令的 EN 输入为 1 状态，出现的第一个错误的 ID 保存在指令的输出参数“ID”中，ID 的数据类型为 Word。第一个错误消失时，指令输出下一个错误的 ID。

作为默认的设置，PLC 对程序块执行出现错误的响应方式是将错误记录在诊断缓冲区，并使 CPU 切换到 STOP 模式。

如果在代码块中调用 GET\_ERROR 与 GET\_ERR\_ID 指令，出现错误时 PLC 不再作出上述的响应，详细的错误信息将由 GET\_ERROR 指令的输出参数 ERROR 来提供，错误的标识符（ID）在 GET\_ERR\_ID 指令的输出参数 ID 指定的地址中。通常第一条错误是最重要的，后面的错误由第一条错误引起。

如果 GET\_ERROR 或 GET\_ERR\_ID 指令的 ENO 为 1 状态，表示出现了代码块执行错误，有错误数据可用。如果 ENO 为 0 状态，表示没有代码块执行错误。

可以用 GET\_ERROR 和 GET\_ERR\_ID 的 ENO 来连接处理错误的程序。

GET\_ERROR 和 GET\_ERR\_ID 可以用于从当前执行的块（被调用的块）发送错误信息给调用它的块。将它们放在被调用块的最后一个网络，以报告被调用块的最后执行状态。

#### 4. ErrorStruct 数据类型的结构

ErrorStruct 数据类型的结构见表 5-6。

## 5.4 字符串转换指令与字符串指令

### 5.4.1 字符串转换指令

#### 1. 字符串的结构

STRING（字符串）数据有 2B 的头部，后面是最多 254B 的 ASCII 字符代码。字符串的首字节是字符串的最大长度，第 2 个字节是当前长度，即当前实际使用的字符数。当前长度必须小于等于最大长度。字符串占用的字节数为最大长度加 2。

字符串默认的最大长度为 254 个字符，定义字符串的最大长度可以减少它占用的存储空间。例如定义了字符串“MyString[10]”之后，字符串 MyString 的最大长度为 10 个字符。

#### 2. 定义字符串

执行字符串指令之前，首先应定义字符串。不能在变量表中定义字符串，只能在代码块的接口区或全局数据块中定义它。

生成全局数据块 DB3（符号名为 DB\_1），在 DB\_1 中生成字符串变量 string1～string4（见图 5-37）。字符串的数据类型 String[18]中的 “[18]” 表示其最大长度为 18 个字符，加上两个头部字节，共 20B，因此 string1 的起始地址（偏移量 Offset）为 DBB200，String2 的 Offset

为 DBB220。如果字符串的数据类型为 String（没有方括号），每个字符串变量将占用 256B。

在 OB1 的接口区生成 3 个字符串变量（见图 5-38），其最大长度均为 18 个字符。

DB_1					
	Name	Data type	Offset	Initial value	Retain
1	▼ Static				
2	► Source	Array [0..99] of Int	0.0		<input type="checkbox"/>
3	String1	String[18]	200.0	" "	<input type="checkbox"/>
4	String2	String[18]	220.0	" "	<input type="checkbox"/>
5	String3	String[18]	240.0	" "	<input type="checkbox"/>
6	String4	String[18]	260.0	" "	<input type="checkbox"/>

图 5-37 在数据块中生成字符串变量

Interface		
	Name	Data type
1	▼ Temp	
2	► ERROR1	ErrorStruct
3	String1	String[18]
4	String2	String[18]
5	String3	String[18]

图 5-38 在接口区生成字符串变量

### 3. 使用 S\_CONV 指令将字符串转换为数值

S\_CONV 指令用于将输入的字符串转换为对应的数值，或将数值转换为对应的字符串。该指令没有输出格式选项，因此需要设置的参数很少，但是没有指令 STRG\_VAL 和 VAL\_STRG 那样灵活。首先需要在指令方框中设置转换前后的操作数 IN 和 OUT 的数据类型。

使用 S\_CONV 将字符串转换为数值时，输入参数 IN 的数据类型为 String，输出参数 OUT 的数据类型可以是 SInt、Int、DInt、USInt、UInt、UDInt 和 Real。

允许转换的字符包括 0~9、加减号和小数点对应的字符。字符串 IN 的转换从第一个字符开始，直到最后一个字符。如果遇到允许的字符之外的字符，转换停止，ENO 被设置为 0。转换后的数值用参数 OUT 指定的地址保存。如果输出的数值超出 OUT 的数据类型允许的范围，OUT 为 0，ENO 被置为 0 状态。反之，OUT 内为有效的值，ENO 被置为 1 状态。

输入字符串的格式规则如下：

- 1) 如果字符串 IN 使用了十进制数的小数点，应使用字符'.'。
- 2) 允许使用分隔每 3 位十进制数的分号字符';'，转换时忽略它。
- 3) 忽略字符前面的空格。
- 4) 只支持定点表示法，不会将字符 e 和 E 视为指数计数法。

用右键点击图 5-39 中的 M2.0，执行出现的快捷菜单中的“Modify”→“Modify to 1”命令，M2.0 的常开触点闭合，左边的 S\_CONV 指令将字符串'1345.6'转换为双整数 1345，小数部分被截位取整。

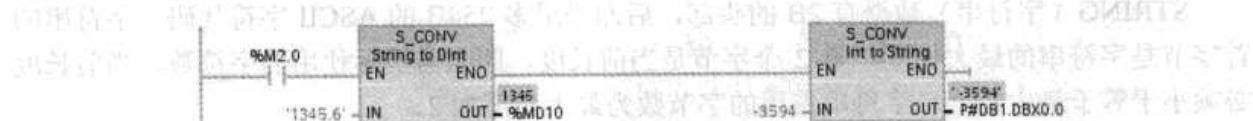


图 5-39 字符串转换指令

### 4. 使用 S\_CONV 指令将数值转换为字符串

可以用指令 S\_CONV 将参数 IN 指定的整数、无符号整数或浮点数转换为输出 OUT 对应的字符串。转换执行之前，应定义参数 OUT 指定的字符串。输入数据值的数据类型可以是 SInt、Int、DInt、USInt、UInt、UDInt 和 Real。

图 5-39 中 M2.0 的常开触点闭合时，右边的 S\_CONV 指令将-3594 转换为字符串'-3594'，替换了 DB1 中定义的字符串 String1 原有的前 5 个字符。

转换后的字符串的长度取决于输入 IN 的数据类型和数值。字符串 OUT 首字节的最大字符串长度应大于等于转换后的字符可能的最大个数。表 5-7 给出了各种数据类型需要的最大的字符串长度。

表 5-7 各种数据类型需要的最大的字符串长度

输入数据类型	输出字符串的最大字符数	例 子	包括头部的总字符串长度
USInt	3	255	5
SInt	4	-128	6
UInt	5	65535	7
Int	6	-32767	8
UDInt	10	4294967295	12
DInt	11	-2147483648	13

输出字符串的格式规则如下：

1) 输出的正数没有符号位，字符串 IN 的十进制数小数点使用字符'!'。

2) 只支持定点表示法，不使用指数计数法。

3) 参数 IN 为数据类型 Real 时，使用英文的句号作十进制数的小数点。

## 5. 复制字符串

如果 S\_CONV 指令输入、输出的数据类型均为 String，输入 IN 指定的字符串将复制到输出 OUT 指定的地址。如果字符串 IN 的实际长度超过了字符串 OUT 的最大长度，只复制 OUT 允许的部分，ENO 将被设置为 0 状态。

## 6. 使用 STRG\_VAL 指令将字符串转换为数值

STRG\_VAL 指令将数值字符串转换为对应的整数或浮点数。从参数 IN 指定的字符串的第 P 个字符开始转换（见图 5-40），直到字符串结束。允许的字符包括数字 0~9、加减号、英语的逗号或小数点、字符 e 和 E。遇到非法的字符时将停止转换，ENO 被设置为 0。

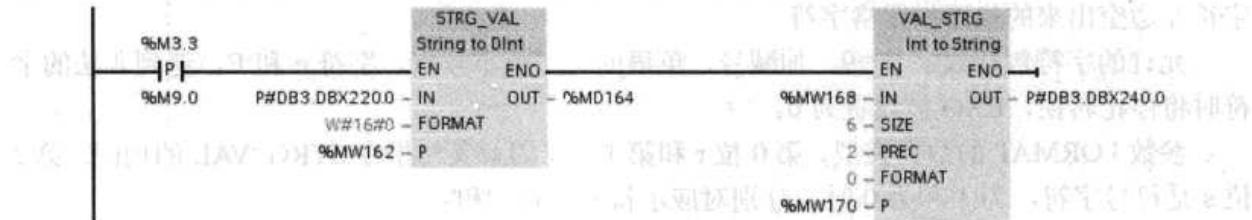


图 5-40 字符串转换指令

转换后的数值保存在参数 OUT 指定的存储单元。如果输出的数值超过 OUT 的数据类型允许的范围，OUT 为 0，ENO 被设置为 FALSE。反之，OUT 内为有效的值，ENO 被设置为 TRUE。

参数 P 是要转换的第一个字符的编号，其参数类型为 IN\_OUT，数据类型为 UInt，P 为 1 时，从字符串的第一个字符开始转换。作为输出参数，是转换结束后字符串中下一个字符的编号。图 5-40 中被转换的字符串从 DB3 的 DBB220 开始存放，其中的字符串为'12345'。

进入 RUN 模式后，图 5-40 中参数 P (MW162) 的值为 0，不能进行转换。在转换之前，用图 5-41 的监视表将 P 的输入值 2 送给 MW162，转换后 P 的输出值为 6。图 5-40 中最初使

用的是 M3.3 的常开触点。调试程序时发现，如果在下一个扫描循环周期以 6 作为 P 的输入值进行转换将会出错，不能进行转换，输出 OUT 为 0。为此将 M3.3 的常开触点改为上升沿检测触点，只是在 M3.3 由 0 变为 1 状态时进行转换，解决了这一问题。

参数 FORMAT 是输出格式选项，数据类型为 Word，第 0 位 r 为 1 和 0 时分别为指数表示法和定点数表示法。第 1 位 f 为 1 和 0 时，分别用英语的逗号和句号作十进制数的小数点，高位为 0。STRG\_VAL 指令的转换规则如下：

1) 如果使用英语的句号“.”作十进制数的小数点，允许使用英语的逗号“,”作分隔每 3 位十进制数的字符，转换时忽略它。

2) 忽略字符前面的空格。

用监视表调试程序时（见图 5-41），首先给 M3.3 写入 0 (FALSE)，然后写入 1 (TRUE)。在 M3.3 的上升沿，MW162 (参数 P) 被写入 2，表示从字符串'12345'的第 2 个字符开始转换。转换结束后 MW162 内是 IN 输入字符串的下一个字符的编号 6。MD164 (OUT) 中的转换结果为 IN 字符串'12345'从第 2 个字符开始的字符对应的数字 2345。

Name	Address	Display format	Monitor value	Modify value
"Tag_119"	%MW162	DEC_unsigned	6	<input checked="" type="checkbox"/> !
"Tag_120"	%MD164	DEC_signed	2345	<input type="checkbox"/> !
"Tag_123"	%M3.3	Bool	TRUE	<input checked="" type="checkbox"/> !
"DB_1".String2	F#DB3.DBX220.0	String	'12345'	<input checked="" type="checkbox"/> !

图 5-41 监视表

## 7. 使用 VAL\_STRG 指令将数值转换为字符串

VAL\_STRG 将输入参数 IN 中的整数、无符号整数或浮点数转换为输出参数 OUT 中对应的字符串。

被转换的字符串将取代 OUT 字符串从参数 P 提供的字符偏移量开始、到参数 SIZE 指定的字符数结束的字符。如果参数 P 和 SIZE 指定的替代字符超过 OUT 允许的最大字符数的范围，ENO 为 0 状态。如果输出值小于指定的长度，写入字符串的字符采用“右对齐”的方式，字符左边空出来的位添加空格字符。

允许的字符包括数字 0~9、加减号、英语的逗号或小数点、字符 e 和 E。遇到非法的字符时将停止转换，ENO 被设置为 0。

参数 FORMAT 的数据类型、第 0 位 r 和第 1 位 f 的意义与指令 STRG\_VAL 的相同。第 2 位 s 是符号字符，为 1 和为 0 时，分别对应于符号字符 '+' 和 '-'。

参数 PREC 用来设置精度或字符串的小数部分的位数。如果参数 IN 的值为整数，PREC 指定小数点的位置。例如数据值为 12345 和 PREC 为 2 时，转换结果为字符串'125.45'。Real 数据类型支持最高精度为 7 位有效数字。

该指令可以用于在文本字符串中嵌入动态变化的数字字符。例如将数字 125.45 嵌入字符串'Ia = A'后，得到字符串'Ia = 125.45A'。

参数 OUT 字符串的格式规则见 S7-1200 的系统手册。  
调试程序时，将初值'Ia = A'写入数据块 DB\_1 中的字符串 string3 (见图 5-42)，用上述的方法产生 M3.3 的上升沿，将初值 5 写入参数 P (MW170)，将参数 IN (MW168) 中的整数 12345 转换为字符串，小数部分为 2 位 (参数 PREC 为 2)，即转换为字符串 '125.45'，转换后的字符串从字符串 string3 的第 5 个字符开始存放，其长度为参数 SIZE 定义的 6 个字

符。指令执行后输出参数 OUT (字符串 string3) 为 'Ia = 123.45 A'，输入 IN 中的数字被成功地嵌入初始字符串，可以用人机界面动态地显示字符串 string3。

Name	Address	Display format	Monitor value	Modify value
"Tag_123"	%M3.3	Bool	TRUE	TRUE
"Tag_121"	%MW168	DEC_signed	12345	12345
"Tag_122"	%MW170	DEC_unsigned	11	5
"DB_1".String3	P#DB3.DBX240.0	String	'Ia=123.45'	'Ia = A'

图 5-42 监视表

## 5.4.2 字符串指令

### 1. LEN 指令 (求字符串长度)

指令 LEN 用输出参数 OUT (整数) 提供输入参数 IN 指定的字符串的当前长度，空字符串(' ')的长度为 0。执行图 5-43 中的 LEN 指令后，MW172 中是输入的字符串的长度 (7 个字符)。

### 2. CONCAT 指令 (合并字符串)

指令 CONCAT 将输入参数 IN1 和 IN2 指定的两个字符串连接在一起，然后用参数 OUT 输出连接后的字符串 (见图 5-43)。合并后字符串 IN1 和 IN2 分别是连接后的字符串的左部分和右部分。如果连接后的字符串的长度大于允许的最大长度，则将它限制在最大长度，并将 ENO 设置为 0。

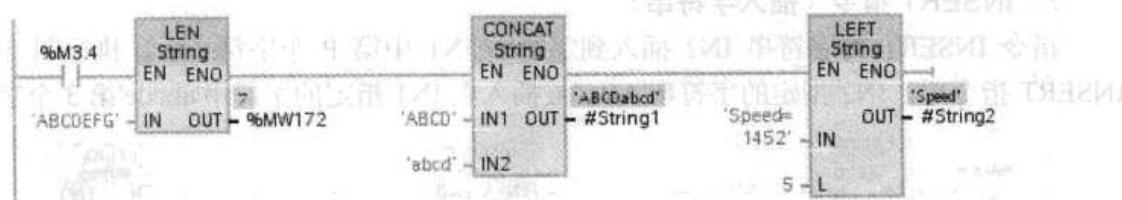


图 5-43 字符串指令

### 3. LEFT 指令 (左子字符串)

指令 LEFT 用参数 OUT 指定的字符串来输出参数 IN 指定的字符串的前 L 个字符，L 的数据类型为 Int。执行图 5-43 中的 LEFT 指令后，输出 OUT 中是 IN 输入的字符串左边的 5 个字符。

如果 L 大于 IN 字符串的当前长度，则 OUT 返回输入的字符串。如果 L 为负数或零，或者输入的是空字符串，则返回空字符串，并将 ENO 置为 0。

如果执行字符串指令时出错，并且可以写输出参数 OUT，将输出一个空的字符串。

### 4. RIGHT 指令 (右子字符串)

指令 RIGHT 用参数 OUT 指定的字符串输出字符串 IN 的最后 L 个字符，L 的数据类型为 Int。对异常情况的处理与指令 LEFT 相同。

执行图 5-44 中的 RIGHT 指令后，输出 OUT 中是字符串 IN 右边的 4 个字符。

### 5. MID 指令 (提供字符串的中间部分)

指令 MID 用参数 OUT 指定的字符串输出字符串 IN 从第 P 个字符开始的 L 个字符。执行图 5-44 中的 MID 指令后，输出 OUT 中是 IN 输入的字符串从第 2 个字符开始的中间 4 个字符。

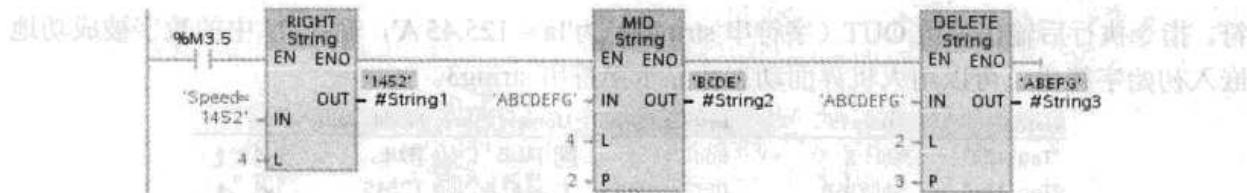


图 5-44 字符串指令

如果 L 加 P 之和大于字符串 IN 的当前长度，则返回的字符串为从字符串 IN 的第 P 个字符到结束字符。如果 P 超出字符串 IN 的当前长度，或者 P 或 L 等于零或为负数，OUT 将返回空字符串，并将 ENO 置为 0。

如果执行字符串指令时出错，并且可以写输出参数 OUT，将输出一个空的字符串。

#### 6. DELETE 指令（删除子字符串）

指令 DELETE 从字符串 IN 中第 P 个字符开始，删除 L 个字符。参数 OUT 输出剩余的子字符串。执行图 5-44 中的 DELETE 指令后，IN 输入的字符串被删除从第 3 个字符开始的 2 个字符后，然后输出到 OUT 指定的字符串。

如果参数 L 或 P 等于零，或者 P 大于字符串 IN 的当前长度，则 OUT 返回输入字符串 IN，并将 ENO 置为 0。如果 L 加 P 之和大于字符串 IN 的当前长度，则一直删除到字符串 IN 的末尾。如果 L 为负数，或者 P 为负数或 0，则 OUT 返回空字符串，并将 ENO 置为 0。

#### 7. INSERT 指令（插入字符串）

指令 INSERT 将字符串 IN2 插入到字符串 IN1 中第 P 个字符之后。执行图 5-45 中的 INSERT 指令后，IN2 指定的字符串'ABC'被插入到 IN1 指定的字符串'abcde'第 3 个字符之后。

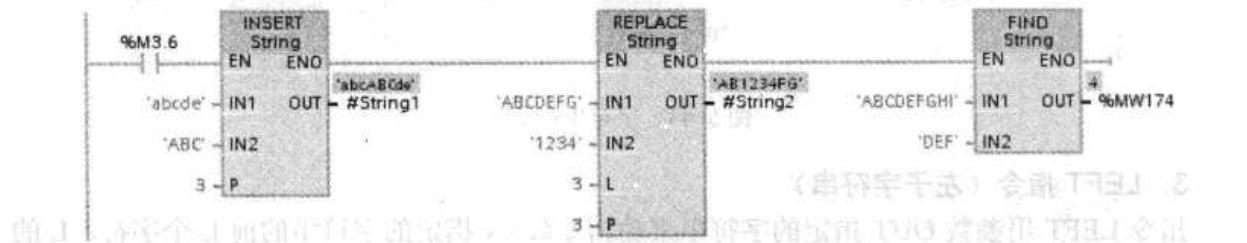


图 5-45 字符串指令

如果 P 大于字符串 IN1 的当前长度，则将字符串 IN2 附加到第一个字符串，并将 ENO 置为 0。如果 P 为负数或 0，则 OUT 输出空字符串，并将 ENO 置为 0。如果插入后的新字符串的长度大于字符串 OUT 允许的最大值，将结果字符串限制在允许的最大长度，并将 ENO 置为 0。

#### 8. REPLACE 指令（子字符串替换）

指令 REPLACE 用字符串 IN2 替换字符串 IN1 中从字符位置 P 开始的 L 个字符，替换的结果用参数 OUT 输出。执行图 5-45 中的 REPLACE 指令后，字符串 IN1 中从第 3 个字符开始的 3 个字符 ('CDE') 被 IN2 指定的字符串'1234'代替。

如果 L 等于零，则 OUT 输出 IN1 指定的字符串。如果 P 等于 1，则字符串 IN1 的前 L 个字符被字符串 IN2 替换。如果 P 大于字符串 IN1 的长度，则将字符串 IN2 附加到字符串 IN1，

并将 ENO 置为 0。如果 L 为负数，或者 P 为负数或 0，则 OUT 返回空字符串，并将 ENO 置为 0。如果替换后新的字符串的长度大于字符串 OUT 允许的最大值，将结果字符串限制在允许的最大长度，并将 ENO 置为 0。

## 9. FIND 指令（查找子字符串）

指令 FIND 提供字符串 IN2 在字符串 IN1 中的位置。查找从左侧开始，输出参数 OUT（整数）返回第一次出现字符串 IN2 的位置。如果在字符串 IN1 中未找到字符串 IN2，则返回零。

执行图 5-45 中的 FIND 指令后，查找到 IN2 指定的字符串 'DEF' 从 IN1 指定的字符串 'ABCDEFGH' 的第 4 个字符开始。

# 5.5 高速脉冲输出与高速计数器

## 5.5.1 高速脉冲输出

### 1. 高速脉冲输出

脉冲宽度与脉冲周期之比称为占空比，脉冲列输出（PTO）功能提供占空比为 50% 的方波脉冲列输出。脉冲宽度调制（PWM）功能提供连续的、脉冲宽度可以用程序控制的脉冲列输出。

每个 CPU 有两个 PTO/PWM 发生器，分别通过 CPU 集成的 Q0.0~Q0.3 或信号板上的 Q4.0~Q4.3 输出 PTO 或 PWM 脉冲（见表 5-8）。

表 5-8 PTO/PWM 的输出点

PTO1		PWM1		PTO2		PWM2	
脉冲	方向	脉冲	方向	脉冲	方向	脉冲	方向
Q0.0 或 Q4.0	Q0.1 或 Q4.1	Q0.0 或 Q4.0	—	Q0.2 或 Q4.2	Q0.3 或 Q4.3	Q0.2 或 Q4.2	—

### 2. PWM 的组态

PWM 功能提供可变占空比的脉冲输出，时间基准可以设置为  $\mu\text{s}$  或 ms。

脉冲宽度为 0 时占空比为 0%，没有脉冲输出，输出一直为 0 状态。脉冲宽度等于脉冲周期时，占空比为 100%，没有脉冲输出，输出一直为 1 状态。

PWM 的高频输出波形经滤波后得到与占空比成正比的模拟量输出电压，可以用来控制变频器的转速和阀门的开度等物理量。

在 STEP 7 Basic 中生成项目“HSC\_COUNT”（见随书光盘中的同名例程），CPU 为 CPU 1214C。使用 PWM 之前，首先应对脉冲发生器组态，具体步骤如下：

- 1) 打开 PLC 的设备视图，选中其中的 CPU。
- 2) 打开下面的监视窗口的“Properties”选项卡，选中左边的“PTO1/PWM1 (Pulse\_1)”中的“General”参数组，用复选框选中右边窗口的复选框“Enable this Pulse generators for use”，激活该脉冲发生器。
- 3) 选中左边窗口的“Parameter assignment”参数组（见图 5-46），在右边的窗口可以设置下列参数：  
使用“Pulse generator used as”下拉式列表，可选脉冲发生器为 PWM 或 PTO。

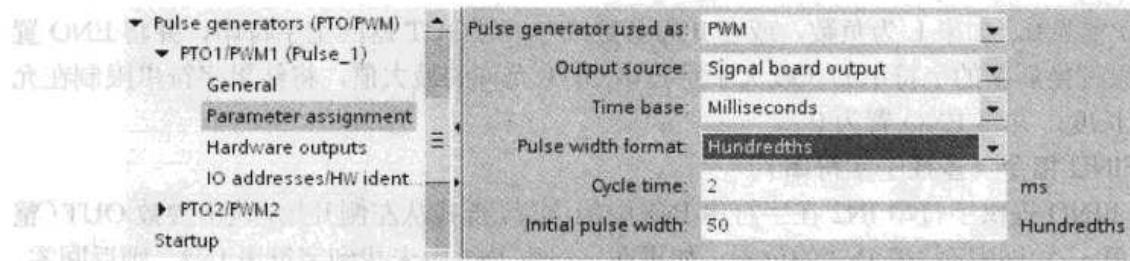


图 5-46 设置脉冲发生器的参数

使用“Output source”（输出源）下拉式列表，可选“Onboard CPU output”（CPU 集成的输出点）或“Signal board output”（信号板的输出点）。

使用“Time base”（时间基准）下拉式列表，可选 Milliseconds (ms) 或 Microseconds ( $\mu$ s)。

使用“Pulse width format”（脉冲宽度格式）下拉式列表，可选下列 4 种脉冲宽度格式：

- Hundredths(百分数，0~100)。
- Thousandths(千分之一，0~1000)。
- Ten-thousandths(万分之一，0~10000)。
- S7 analog format (S7 模拟量格式，0~27648)。

用输入域“Cycle time”设置脉冲的周期值，采用 Time base 选择的时间单位。

用输入域“Initial pulse width”设置初始脉冲宽度，采用 Pulse width format 选择的单位，图 5-46 中为百分数，脉冲的占空比为 50%，脉冲周期为 2ms，脉冲宽度为 1ms。

4) 选中左边窗口的“I/O addresses/HW identifier”参数组，在右边的窗口（见图 5-47）可以看到 PWM1 的起始、结束地址和 HW ID（硬件标识符）。可以修改其起始地址，在运行时可以用这个地址来修改脉冲宽度。

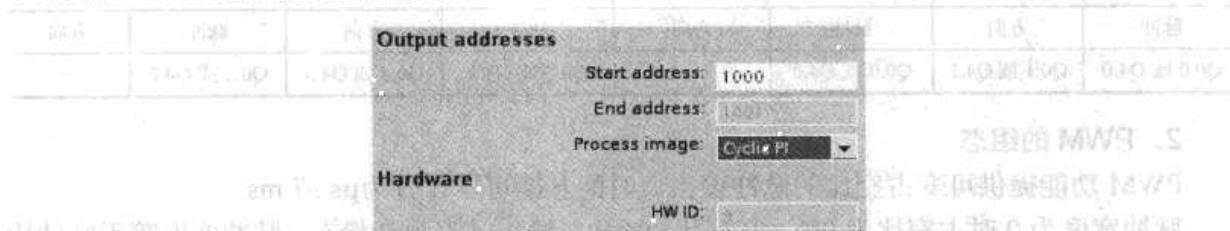


图 5-47 PWM 的输出地址

### 3. PWM 的编程

将右边的扩展指令窗口（Extended instructions）的文件夹“Pulse”中的 CTRL\_PWM 指令拖放到 OB1（见图 5-48），点击出现的“Call options”对话框中的“OK”按钮，生成该指令的背景数据块。

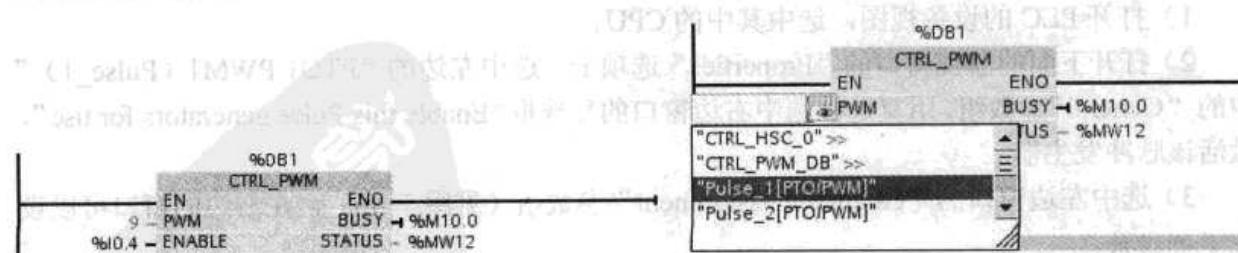


图 5-48 PWM 指令

点击参数 PWM 左边的问号，再点击出现的 按钮，用下拉式列表选中“Pulse\_1”，其硬件标识符（HW ID）为 9。

EN 输入信号为 1 状态时，用参数 ENABLE (I0.4) 来启动或停止脉冲发生器，用 PWM 的输出地址（见图 5-47）来修改脉冲宽度。因为在执行指令 CTRL\_PWM 时 S7-1200 激活了脉冲发生器，输出 BUSY 总是 0 状态。参数 STATUS 是状态代码。

## 5.5.2 编码器

高速计数器一般与增量式编码器一起使用，后者每圈发出一定数量的计数脉冲和一个复位脉冲，作为高速计数器的输入。编码器有以下几种类型：

### 1. 增量式编码器

光电增量式编码器的码盘上有均匀刻制的光栅。码盘旋转时，输出与转角的增量成正比的脉冲，需要用计数器来计脉冲数。有 3 种增量式编码器：

- 1) 单通道增量式编码器内部只有 1 对光耦合器，只能产生一个脉冲列。
- 2) 双通道增量式编码器又称为 A/B 相型编码器，内部有两对光耦合器，输出相位差为 90° 的两组独立脉冲列。正转和反转时两路脉冲的超前、滞后关系相反（见图 5-49），如果使用 A、B 相型编码器，PLC 可以识别出转轴旋转的方向。

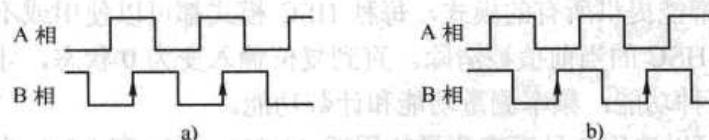


图 5-49 A、B 相型编码器的输出波形

a) 正转 b) 反转

A/B 相正交计数器可以选择 1 倍速模式（见图 5-50）和 4 倍速模式（见图 5-51），1 倍速模式在时钟脉冲的每一个周期计 1 次数，4 倍速模式在时钟脉冲的每一个周期计 4 次数。

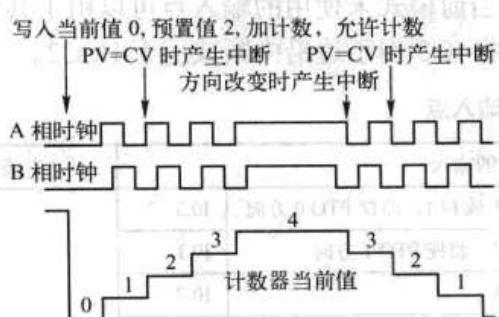


图 5-50 1 倍速正交模式操作举例

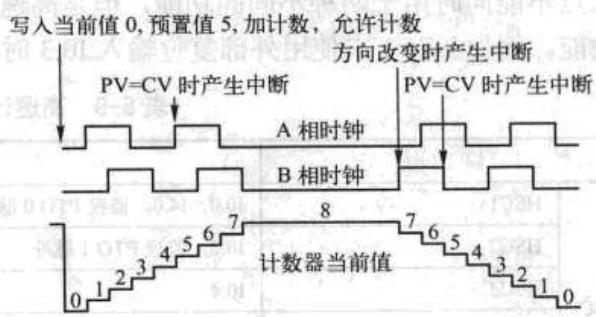


图 5-51 4 倍速正交模式操作举例

3) 三通道增量式编码器内部除了有双通道增量式编码器的两对光耦合器外，在脉冲码盘的另外一个通道内还有一个透光段，每转 1 圈输出一个脉冲，该脉冲称为 Z 相零位脉冲，用于系统清零信号，或作为坐标的原点，以减少测量的积累误差。

### 2. 绝对式编码器

N 位绝对式编码器有 N 个码道，最外层的码道对应于编码的最低位。每一码道有一个光

耦合器，用来读取该码道的 0、1 数据。绝对式编码器输出的  $N$  位二进制数反映了运动物体所处的绝对位置，根据位置的变化情况，可以判别出旋转的方向。

### 5.5.3 高速计数器

PLC 的普通计数器的计数过程与扫描工作方式有关，CPU 通过每一个扫描周期读取一次被测信号的方法来捕捉被测信号的上升沿，被测信号的频率较高时，会丢失计数脉冲，因此普通计数器的最高工作频率一般仅有几十赫兹。高速计数器可以对普通计数器无能为力的高速事件进行计数。

#### 1. 高速计数器的功能

S7-1200 PLC 集成有 6 个高速计数器 (HSC)。HSC1~HSC3 的最高计数频率为 100 kHz。CPU 1211C 可以使用 HSC1~HSC3，CPU 1212C 可以使用 HSC1~HSC4，使用信号板 DI2/DO2 后，它们还可以使用 HSC5。CPU 1214C 可以使用 HSC1~HSC6。

在用户程序使用 HSC 之前，应为 HSC 组态，设置 HSC 的计数模式。大多数 HSC 的参数只能在项目的设备组态中设置，某些 HSC 的参数在设备组态中初始化，以后可以用程序来修改。

HSC 有 4 种工作模式：内部方向控制的单相计数器，外部方向控制的单相计数器，两路计数脉冲输入的计数器和 A/B 相计数器。

并非每个 HSC 都能提供所有的模式，每种 HSC 模式都可以使用或不使用复位输入。复位输入为 1 状态时，HSC 的当前值被清除。直到复位输入变为 0 状态，才能启动计数功能。

高速计数器有两种功能：频率测量功能和计数功能。

某些 HSC 模式可以选用 3 种频率测量的周期 (0.01s、0.1s 和 1.0s) 来测量频率值。频率测量周期决定了多长时间计算和报告一次新的频率值。得到的是根据信号脉冲的计数值和测量周期计算出的频率平均值，频率的单位为 Hz (每秒的脉冲数)。

#### 2. 高速计数器使用的输入点

表 5-9 给出了用于高速计数器的计数脉冲、方向控制和复位的输入点的地址。同一个输入点不能同时用于两种不同的功能，但是高速计数器当前模式未使用的输入点可以用于其他功能。例如 HSC1 未使用外部复位输入 I0.3 时，可以将 I0.3 用于边沿中断或用于 HSC2。

表 5-9 高速计数器的输入点

描    述		默认的输入			功    能
HSC	HSC1	I0.0, I4.0, 监视 PTO 0 脉冲	I0.1 或 I4.1, 监视 PTO 0 方向	I0.3	
	HSC2	I0.2, 监视 PTO 1 脉冲	I0.3, 监视 PTO 1 方向	I0.1	
	HSC3	I0.4	I0.5	I0.7	
	HSC4	I0.6	I0.7	I0.5	
	HSC5	I1.0 或 I4.0	I1.1 或 I4.1	I1.2	
	HSC6	I1.3	I1.4	I1.5	
模式	内部方向控制的单相计数器	计数脉冲		计数复位	计数或测频
	外部方向控制的单相计数器	计数脉冲	方向	计数复位	计数或测频
	两路计数脉冲输入的计数器	加计数脉冲	减计数脉冲	计数复位	计数或测频
	A/B 相正交计数器	A 相脉冲	B 相脉冲	Z 相脉冲	计数或测频
	监视脉冲列输出 (PTO)	计数脉冲	方向		计数

HSC1 和 HSC2 可以分别用来监视脉冲列输出 PTO1 和 PTO2。I4.0 和 I4.1 是 2DI/2DO 信号板的输入点，I0.0~I1.5 是 CPU 集成的输入点，复位信号和 Z 相脉冲仅用于计数模式。

数字量 I/O 点指定给 HSC、PWM（脉冲宽度调制）和 PTO（脉冲列输出）后，不能用监视表的强制功能来修改这些 I/O 点。

HSC1~HSC6 的当前值的数据类型为 DInt，默认的地址为 ID1000~ID1020（见图 5-56），可以在组态时修改地址。

由于 CPU 1211C 和 CPU 1212C 集成的输入点较少，它们不支持某些 HSC，具体的情况见 S7-1200 的系统手册。

### 3. 高速计数器的组态步骤

- 1) 打开 PLC 的设备视图，选中其中的 CPU。
- 2) 选中监视窗口的属性选项卡左边的高速计数器 HSC1 的“General”参数组，用复选框选中“Enable this high-speed counter for use”，激活该 HSC。

如果激活了脉冲发生器 PTO1 或 PTO2，它们分别使用 HSC1 和 HSC2 的“Motion axis”计数模式，来监视硬件输出。如果组态 HSC1 或 HSC2 用于其他任务，它们不能被脉冲发生器 PTO1 或 PTO2 使用。

- 3) 选中左边的“Function”（功能）参数组（见图 5-52），在右边的窗口可以设置下列参数：

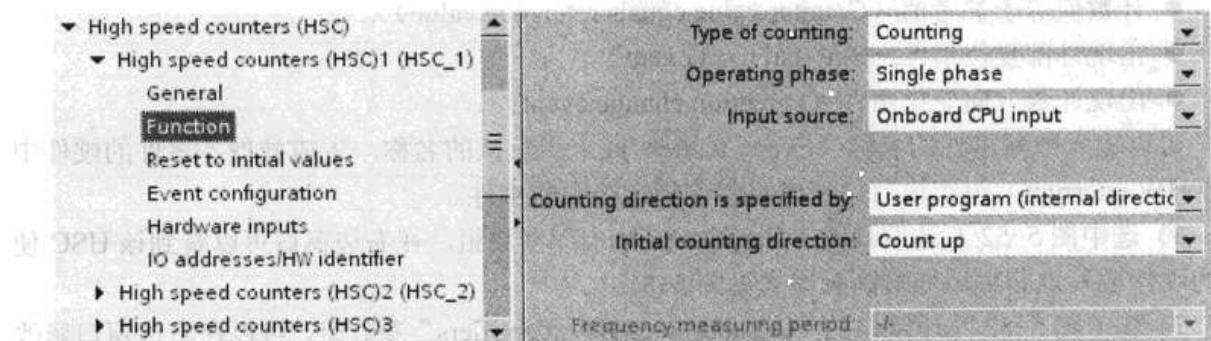


图 5-52 高速计数器的功能设置

- 使用“Type of counting”（计数类型）下拉式列表，可选“Counting”（计数）、“Frequency”（测频）或“Axis of motion”（运动轴）。
- 使用“Operating phase”（运行相）下拉式列表，可选“Single phase”（单相）、“Two phase”（双相）、“AB Quadrature 1×”（1 倍速正交 AB 相）和“AB Quadrature 4×”（4 倍速正交 AB 相）。
- 使用“Input source”（输入源）下拉式列表，可选“Onboard CPU input”（CPU 集成的输入点）或“Signal board input”（信号板上的输入点）。
- 使用“Counting direction is specified by”（用…指定计数方向）下拉式列表，可选“User program (internal direction control)”（用户程序（内部方向控制））或“Input point (external direction control)”（输入点（外部方向控制））。
- 如果选择了“User program (internal direction control)”，可用“Initial counting direction”（初始化计数方向）下拉式列表，选择“Count up”（加计数）或“Count down”（减计数）。

计数)。

- 如果设置为频率测量模式, 使用“Frequency measuring period”(频率测量周期)下拉式列表, 可以选择0.01s、0.1s和1.0s。

4) 选中图5-52左边窗口的“Reset to initial values”(复位到初始值)参数组, 可以设置“Initial counter value”(HSC的计数初始值, 见图5-53)和“Initial reference value”(计数参考值的初始值)。还可以用复选框设置是否使用外部复位输入(external reset input), 用下拉式列表选择“Reset signal level”(复位信号电平)是“Active high”(高电平)或是“Active low”(低电平)。

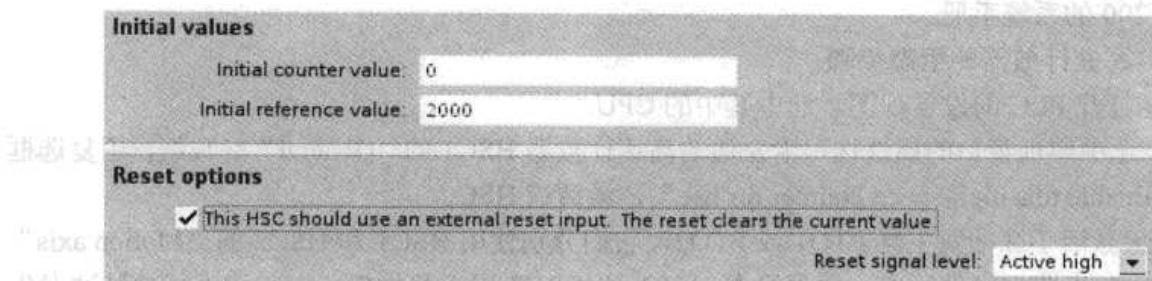


图5-53 设置高速计数器的初始值与复位信号

5) 选中图5-52左边窗口的“Event configuration”(事件组态)参数组, 可以用右边窗口的复选框激活下列事件出现时是否触发中断(见图5-54)：

- 计数值等于参考值(Counter value equals reference value)。
- 出现外部复位事件(External reset event)。
- 出现计数方向变化事件(Direction change event)。

可以输入中断事件的名称(Event name)或采用默认的名称。生成处理各事件的硬件中断(HW interrupt)组织块后, 可以将它们指定给中断事件。

6) 选中图5-52左边窗口的“Hardware inputs”参数组, 在右边窗口可以看到该HSC使用的硬件输入点和可用的最高频率(见图5-55)。

7) 选中图5-52左边窗口的“I/O addresses/HW identifiers”参数组, 可以在右边窗口修改HSC的起始地址(见图5-56)。

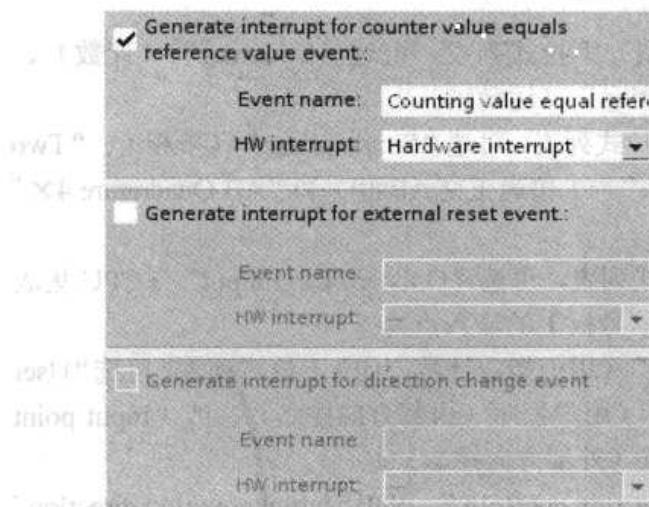


图5-54 高速计数器的事件组态



图5-56 地址与HW ID

## 5.5.4 高速脉冲输出与高速计数器的计数实验

### 1. 实验的基本要求

用高速脉冲输出功能产生周期为 2ms，占空比为 50% 的 PWM 脉冲列，送给高速计数器 HSC1 计数。通过设置不同的参考值，在计数值分别为 2000、3000 和 1500 时产生中断（见图 5-57）。在中断程序中修改计数值、参考值和计数方向，同时改变 Q0.4~Q0.6 的状态。

因为中断事件产生的速率远远低于高速计数器计数脉冲的速率，用高速计数器可以实现高速运动的精确控制，并且与 PLC 的扫描周期关系不大。

### 2. 硬件接线

作者做实验使用的是继电器输出的 CPU 1214C，为了输出高频脉冲，使用了一块 2DI/2DO 信号板。用信号板的输出点 Q4.0 发出 PWM 脉冲，送给 HSC1 的高速脉冲输入点 I0.0 计数。

图 5-58 是硬件接线图，CPU 的 L+ 和 M 端子之间是内置的 DC 24V 电源。将它的参考点 M 与数字量输入的内部电路的公共点 1M 相连，用内置的电源作输入回路的电源。内置的电源同时又作为 2DI/2DO 信号板的电源。电流从 DC 24V 电源的正极 L+ 流出，流入信号板的 L+ 端子，经过信号板内部的 MOSFET（场效应管）开关，从 Q4.0 输出端子流出，流入 I0.0 的输入端，经内部的输入电路，从 1M 端子流出，最后回到 DC 24V 电源的负极 M 点。

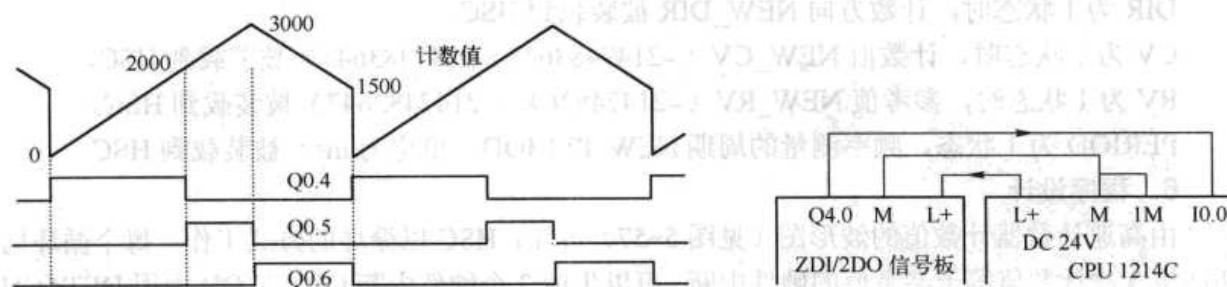


图 5-57 高速计数器的计数当前值波形图

图 5-58 硬件接线图

可以用外部的脉冲信号发生器或增量式编码器为高速计数器提供外部脉冲信号。使用 CPU 的脉冲发生器的优点是简单方便，做频率测量实验时易于验证测量的结果。

### 3. PWM 的组态与编程

组态 PTO1/PWM1 产生 PWM 脉冲（见图 5-46），输出源为信号板上的输出点，时间单位为 ms，脉冲宽度的格式为百分数，脉冲的周期为 2ms，初始脉冲宽度为 50%。

在 OB1 中调用 CTRL\_PWM 指令（见图 5-48），用 I0.4 启动脉冲发生器。

### 4. 高速计数器的组态

组态时设置 HSC1 的工作方式为单相脉冲计数（见图 5-52），使用 CPU 集成的输入点 I0.0，通过用户程序改变计数的方向。设置 HSC 的初始状态为加计数，初始计数值为 0，初始计数参考值为 2000（见图 5-53）。出现计数值等于参考值的事件时，调用硬件中断组织块 OB200（见图 5-54）。HSC 默认的地址为 ID1000（见图 5-56），在运行时可以用该地址监视 HSC 的计数值。

### 5. 高速计数器控制指令

图 5-60 中的 CTRL\_HSC 是高速计数器控制指令。指令各参数的意义见表 5-10。

表 5-10 CTRL\_HSC 指令的参数

参 数	数 据 类 型	存 储 区	描 述
HSC	HW_HSC	I, D 或常数	高速计数器的硬件标识符 (HW ID)
DIR	Bool	I, Q, M, L, D	I 状态时请求设置新的计数方向 (见 NEW_DIR)
CV	Bool	I, Q, M, L, D	I 状态时请求设置新的计数值 (见 NEW_CV)
RV	Bool	I, Q, M, L, D	I 状态时请求设置新的计数参考值 (见 NEW_RV)
PERIOD	Bool	I, Q, M, L, D	I 状态时请求设置新的频率测量周期值 (见 NEW_PERIOD)
NEW_DIR	Int	I, Q, M, L, D	DIR 为 I 状态时装载的新的计数方向, 1 为加计数, -1 为减计数
NEW_CV	DInt	I, Q, M, L, D	CV 为 I 状态时装载的新的计数值
NEW_RV	DInt	I, Q, M, L, D	RV 为 I 状态时装载的新的计数参考值
NEW_PERIOD	Int	I, Q, M, L, D	PERIOD 为 I 状态时装载的新的频率测量周期值 (0.01s、0.1s、1s)
BUSY	Bool	I, Q, M, L, D	正在处理
STATUS	Word	I, Q, M, L, D	执行的状态代码

如果请求修改参数的 DIR、CV、RV 等为 0 状态，相应的输入值被忽略。

只有在组态时设置计数方向由用户程序控制，参数 DIR 才有效。BUSY 参数总是为 0。

DIR 为 1 状态时，计数方向 NEW\_DIR 被装载到 HSC。

CV 为 1 状态时，计数值 NEW\_CV (-2147483648~2147483647) 被装载到 HSC。

RV 为 1 状态时，参考值 NEW\_RV (-2147483648~2147483647) 被装载到 HSC。

PERIOD 为 1 状态，频率测量的周期 NEW\_PERIOD (单位为 ms) 被装载到 HSC。

## 6. 程序设计

由高速计数器计数值的波形图 (见图 5-57) 可知，HSC 以循环的方式工作。每个循环周期产生 3 次计数值等于参考值的硬件中断。可以生成 3 个硬件中断 OB，在 OB 中用 DETACH 指令断开硬件中断事件与原来的中断 OB 的连接，用中断连接指令 ATTACH 将下一个中断 OB 指定给中断事件 (见 6.3.5 节)。

本节的程序采用另一种处理方法，设置 MB11 为标志字节，其取值范围为 0、1、2，其初始值为 0。HSC1 的计数值等于参考值时，调用 OB200。根据 MB11 的值，用比较指令来判断是图 5-57 中的哪一次中断，以调用不同的 CTRL\_HSC 指令，来设置下一阶段的计数方向、计数值的初始值和参考值，同时对输出点进行置位和复位处理。处理完后，将 MB11 的值加 1，运算结果如果为 3，将 MB11 清 0 (见图 5-63)。

组态 CPU 时，采用默认的 MB1 作系统存储器字节 (见图 2-32)。CPU 进入 RUN 模式后，M1.0 仅在首次扫描时为 1 状态。在 OB1 中，用 M1.0 的常开触点将标志字节 MB11 清 0 (见图 5-59)，将输出点 Q0.4 置位为 1。

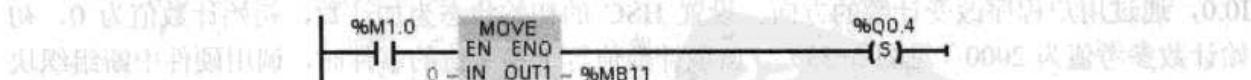


图 5-59 OB1 中的初始化程序

将组态数据、OB1 和硬件中断 OB200 下载到 CPU 后运行程序。用外接的小开关使 I0.4 为 1 状态，信号板的 Q4.0 开始输出 PWM 脉冲，送给 I0.0 计数。因为传送的是占空比为 0.5 的脉冲，Q4.0 和 I0.0 的 LED 的亮度比其他输入点的 LED 稍微暗一点。

开始运行时使用组态的初始值，计数值小于参考值 2000 时（见图 5-57），输出 Q0.4 为 1 状态。计数值等于参考值时产生中断，调用硬件中断组织块 OB200。此时标志字节 MB11 的值为 0，OB200 的 Network 1 中的比较触点接通（见图 5-60），调用第一条 CTRL\_HSC 指令，CV 为 0，HSC1 的计数当前值保持不变。将新的参考值 3000 送给 HSC1，并复位 Q0.4，置位下一阶段的输出 Q0.5。在 Network 4 将 MB11 的值加 1（见图 5-63）。

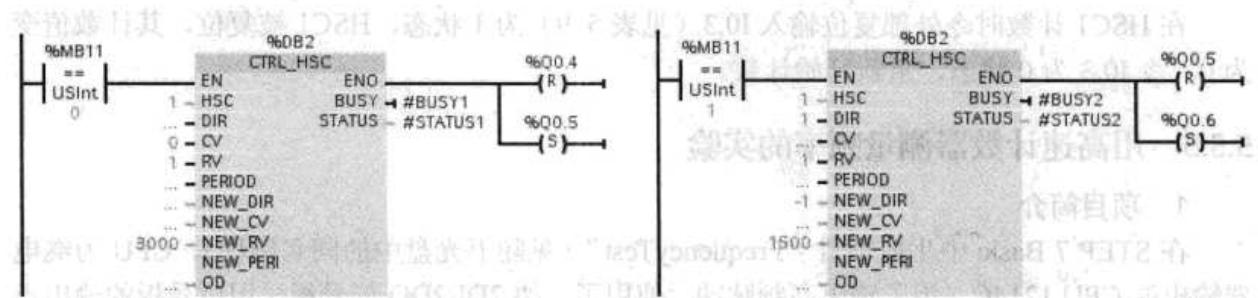


图 5-60 OB200 的 Network 1 的程序

图 5-61 OB200 的 Network 2 的程序

当计数值等于参考值 3000 时产生中断，第 2 次调用硬件中断组织块 OB200。此时标志字节 MB11 的值为 1，OB200 的 Network 2 中的比较触点接通（见图 5-61），调用第 2 条 CTRL\_HSC 指令，虽然未设置 CV，它仍然为 0，HSC1 的计数当前值保持不变。装载新的参考值 1500，将计数方向改为减计数，并复位 Q0.5，置位下一阶段的输出 Q0.6。在 Network 4 将 MB11 的值加 1。

当计数值等于参考值 1500 时产生中断，第 3 次调用硬件中断组织块 OB200。此时标志字节 MB11 的值为 2，OB200 的 Network 3 中的比较触点接通（见图 5-62），调用第 3 条 CTRL\_HSC 指令，装载新的参考值 2000。CV 为 1，用参数 NEW\_CV 将计数当前值复位为 0。计数方向改为加计数，并复位 Q0.6，置位下一阶段的输出 Q0.4。

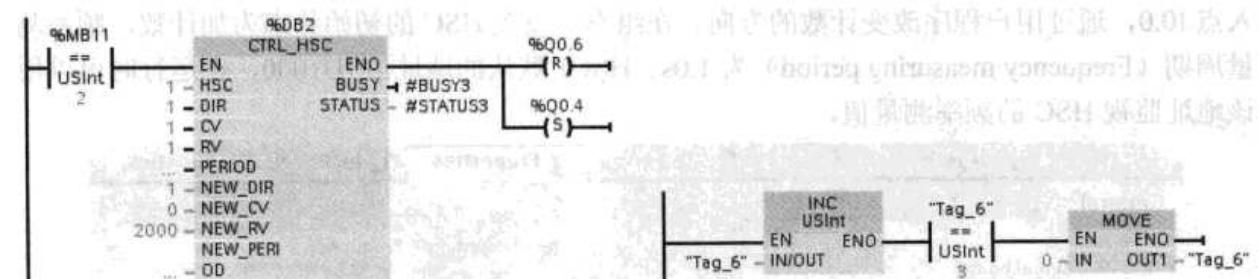


图 5-62 OB200 的 Network 3 的程序

图 5-63 OB200 的 Network 4 的程序

在 Network 4 将 MB11 加 1 后，其值为 3（见图 5-63），比较触点接通，MOVE 指令将 MB11 复位为 0。

以后将重复上述的 3 个阶段的运行，直到 I0.4 变为 0 状态，脉冲发生器停止发出脉冲为止。Q0.4~Q0.6 依次为 1 状态的时间分别为 4s、2s 和 3s，分别与 3 个阶段的计数值 2000、1000 和 1500 对应。

用监视表监视 ID1000，可以看到 HSC1 的计数值的变化情况。图 5-64 同时监视了标志字节 MB11 的值。

	Name	Address	Display format	Monitor value
1		%ID1000	DEC_signed	2254
2	"Tag_6"	%MB11	DEC_signed	1

图 5-64 监视表

在 HSC1 计数时令外部复位输入 I0.3 (见表 5-9) 为 1 状态, HSC1 被复位, 其计数值变为 0。令 I0.3 为 0 状态, 重新开始计数。

### 5.5.5 用高速计数器测量频率的实验

#### 1. 项目简介

在 STEP 7 Basic 中生成项目 “FrequencyTest” (见随书光盘中的同名例程), CPU 为继电器输出的 CPU 1214C。为了输出高频脉冲, 使用了一块 2DI/2DO 信号板。用信号板的输出点 Q4.0 发出 PWM 脉冲, 送给 HSC1 的高速脉冲输入点 I0.0 测量频率, 硬件接线见图 5-58。

#### 2. PWM 的组态与编程

打开 PLC 的设备视图, 选中其中的 CPU。打开下面的监视窗口的 Properties 选项卡, 选中左边的 PTO1/PWM1 (Pulse\_1) 文件夹中的 “General” 参数组, 选中右边窗口的复选框 “Enable this Pulse generators for use”, 激活该脉冲发生器。

选中左边窗口的 “Parameter assignment” 参数组 (见图 5-46), 组态 PTO1/PWM1 产生 PWM 脉冲, 信号源为信号板上的输出点, 时间单位为 ms, 脉冲宽度的格式为百分数, 脉冲的周期为 2ms, 初始脉冲宽度为 50%。

在 OB1 中调用 CTRL\_PWM 指令 (见图 5-48), 用 I0.4 启动脉冲发生器。

#### 3. 高速计数器的组态

设置 HSC1 的工作方式为 “Frequency” (频率测量, 见图 5-65), 使用 CPU 集成的输入点 I0.0, 通过用户程序改变计数的方向。在组态时设置 HSC 的初始状态为加计数, 频率测量周期 (Frequency measuring period) 为 1.0s。HSC1 默认的地址为 ID1000, 在运行时可以用该地址监视 HSC 的频率测量值。

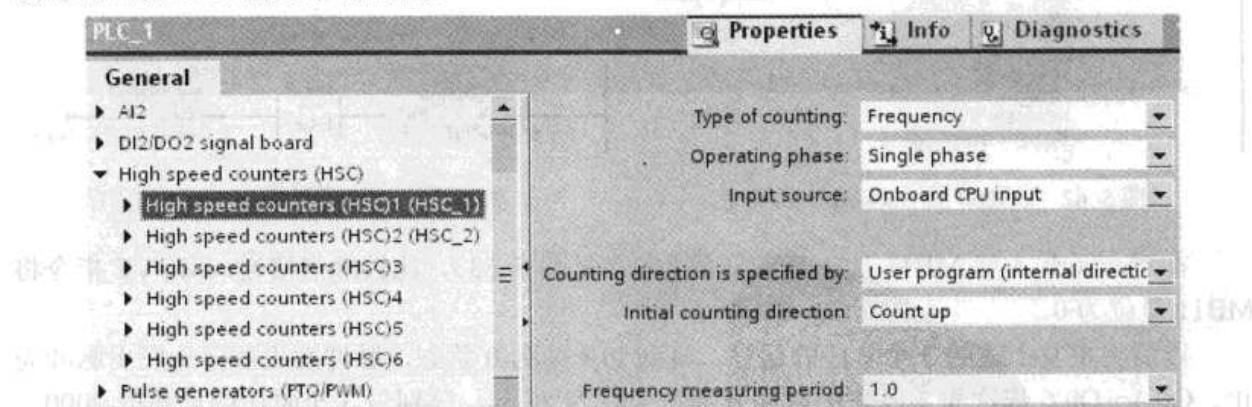


图 5-65 组态 HSC 测频

#### 4. 实验情况

将组态数据和用户程序下载到 CPU 后运行程序。用外接的小开关使 I0.4 为 1 状态, 信号

板的 Q4.0 开始输出 PWM 脉冲，送给 I0.0 测频。PWM 脉冲使 Q4.0 和 I0.0 的 LED 点亮。因为传送的是占空比为 0.5 的脉冲，Q4.0 和 I0.0 的 LED 的亮度比其他输入点的 LED 稍微暗一些。如果脉冲的频率较低，Q4.0 和 I0.0 的 LED 将会闪动。

在监视表中输入 HSC1 的地址 ID1000（见图 5-66），点击工具栏上的  按钮，“Monitor value”列显示测量得到的频率值为 500Hz。

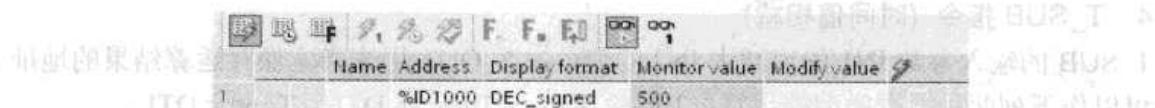


图 5-66 监视表

在设置脉冲发生器的图 5-46 中，修改 PWM 脉冲的周期。在图 5-65 中，修改频率测量的周期。脉冲周期在  $10\mu\text{s} \sim 100\text{ms}$  之间变化时，都能得到准确的频率测量值。信号频率较低时，应选用较大的测量周期。信号频率较高时，频率测量周期为 0.01s 时也能得到准确的测量值。

## 5.6 其他指令

### 5.6.1 实时时钟指令

CPU 的实时时钟（Time-of-day Clock）在 CPU 断电时由超级电容提供的能量保证时钟的运行。CPU 上电至少 24h 后，超级电容充的能量可供时钟运行 10 天。打开在线与诊断视图，在 CPU 的“Clock”属性中设置实时时钟（见图 7-13）。也可以用时钟指令来读、写实时时钟。

#### 1. 日期时间的数据类型

1) 数据类型 Time 的长度为 4B，取值范围为 T#-24d\_20h\_31m\_23s\_648ms ~ T#24d\_20h\_31m\_23s\_647ms (-2 147 483 648 ms ~ 2 147 483 647 ms)。

2) 数据结构 DTL (日期时间) 如表 5-11 所示。可以在全局数据块或块的接口区中定义 DTL 变量。

表 5-11 数据结构 DTL

数 据	字 节 数	取 值 范 围	数 �据	字 节 数	取 值 范 围
年	2	1970~2554	小时	1	0~23
月	1	1~12	分钟	1	0~59
日	1	1~31	秒	1	0~59
星期	1	1~7 (星期日~星期六)	纳秒	4	0~999 999 999

#### 2. T\_CONV 指令(时间转换)

T\_CONV (见图 5-67) 用于将数据类型 Time 转换为 DInt，或者作反向的转换。IN 和输出参数 OUT 均可以取数据类型 Time 和 DInt。

输入程序时，需要用指令名称下面的下拉式列表来选择某些实时时钟指令输入、输出参数的数据类型。

### 3. T\_ADD 指令 (时间值相加)

T\_ADD 和 T\_SUB 的输入参数 IN1 和输出参数 OUT 的数据类型可选 DTL 或 Time，它们的数据类型应相同。IN2 的数据类型为 Time。

T\_ADD 的输入参数 IN1 的值与 IN2 的值相加，参数 OUT 用来指定保存运算结果的地址。

可以作下列两种数据类型的运算：Time + Time = Time 或 DTL + Time = DTL。

### 4. T\_SUB 指令 (时间值相减)

T\_SUB 的输入参数 IN1 的值减去 IN2 的值，参数 OUT 用来指定保存运算结果的地址。

可以作下列两种数据类型的运算：Time - Time = Time 或 DTL - Time = DTL。

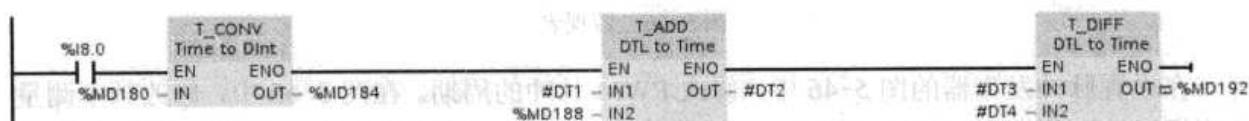


图 5-67 实时时钟指令

### 5. T\_DIFF 指令 (时间差)

T\_DIFF 的输入 IN1 的 DTL 值减去 IN2 的 DTL 值，参数 OUT 提供数据类型为 Time 的差值。即 DTL - DTL = Time。如果 DTL 或 Time 值无效，ENO 为 0，参数 OUT 为 0。

如果 IN2 指定的时间大于 IN1 指定的时间，OUT 输出的结果为负值。如果运算结果超出允许的范围，运算结果被限幅，ENO 被置为 0 状态。

## 6. 时钟指令

时钟指令用来设置和读取系统时钟（见图 5-68）。数据类型 DTL 用于提供日期和时间值。

1) 指令 WR\_SYS\_T（写系统时间）将输入 IN 的 DTL 值写入 PLC 的实时时钟。这个时间值不包括对本地时区和夏令时的补偿。输出 RET\_VAL 是返回的指令执行的状态信息，数据类型为 Int。

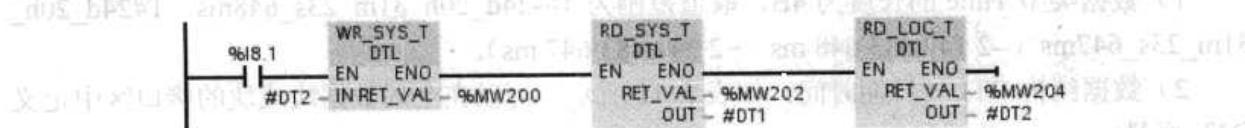


图 5-68 读写实时时钟指令

2) RD\_SYS\_T（读系统时间）将读取的 PLC 当前系统时间保存在输出 OUT 中，数据类型为 DTL。这个时间值不包括对本地时区和夏令时的补偿。输出 RET\_VAL 是返回的指令执行的状态信息。

3) RD\_LOC\_T（读本地时间）的输出 OUT 提供数据类型为 DTL 的 PLC 中的当前本地时间。为了保证读取到正确的时间，在组态 CPU 的属性时，应设置实时时钟（Time of day）的时区（Time zone）为 Beijing（见图 2-34），不设置夏令时。在读取实时时间时，应调用 RD\_LOC\_T 指令。

【例 6-5】用实时时钟指令控制路灯的定时接通和断开，20: 00 开灯，06: 00 关灯，图 5-69 是梯形图程序。首先用 RD\_LOC\_T 读取实时时间，保存在数据类型为 DTL 的局部变量 DT5 中，其中的 HOUR 是小时值，其变量名称为 DT5.HOUR。用 Q0.0 来控制路灯，20:00~0:00 时，上面的比较触点接通；0:00~6:00 时，下面的比较触点接通。

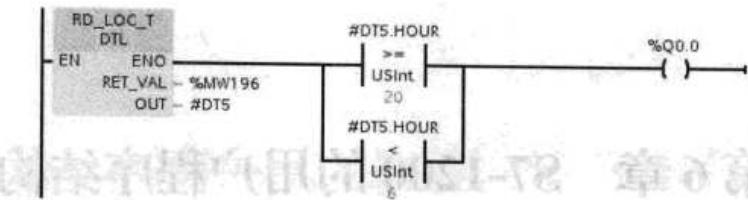


图 5-69 路灯控制电路

## 5.6.2 运动控制指令简介

运动控制指令使用有关的技术数据块和 CPU 的 PTO (脉冲列输出) 来控制转轴的运动。有关运动控制指令的详细信息见它们的在线帮助。

**MC\_Power** 指令 (见图 5-70) 用来激活或禁止运动控制轴。

**MC\_Reset** 指令复位所有的运动控制错误，所有可以确认的运动控制错误被确认。

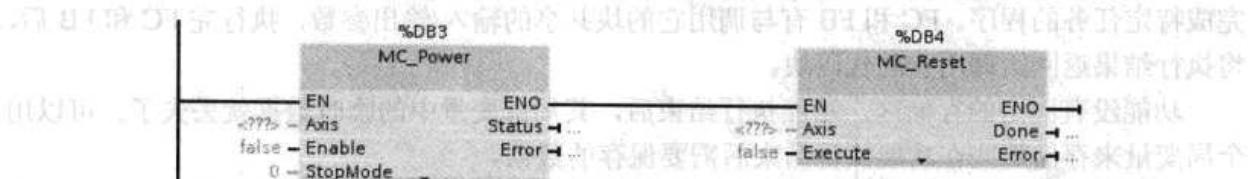


图 5-70 运动控制指令

**MC\_Home** 指令 (见图 5-71) 用来建立轴控制程序和轴的机械定位系统之间的关系。

**MC\_Halt** 指令取消所有的运动处理，使轴的运动停止，但是没有定义停止的位置。

**MC\_MoveAbsolute** 指令启动到一个绝对位置的运动，达到目标位置时任务终止。

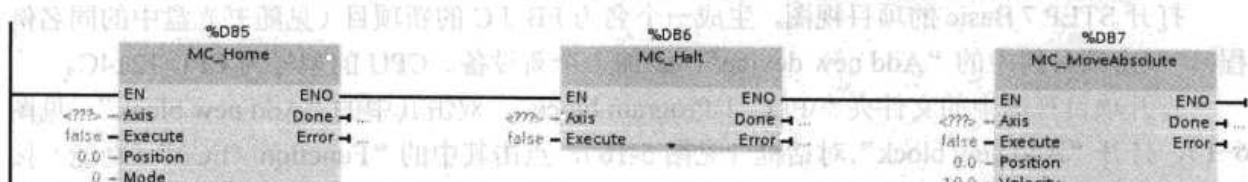


图 5-71 运动控制指令

**MC\_MoveRelative** 指令 (见图 5-72) 启动相对于起始位置的定位运动。

**MC\_MoveVelocity** 指令使轴按设置的转速运动。

**MC\_MoveJog** 指令用于在测试和启动时执行点动 (Jog) 操作。

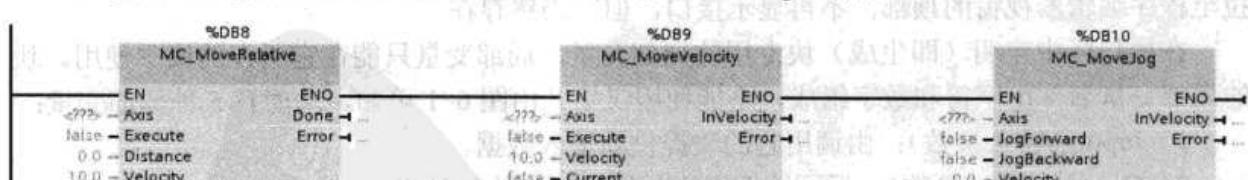


图 5-72 运动控制指令

# 第6章 S7-1200的用户程序结构

## 6.1 功能与功能块

### 6.1.1 生成与调用功能

#### 1. 功能的特点

功能 (Function, FC) 和功能块 (Function block, FB) 是用户编写的子程序, 它们包含完成特定任务的程序。FC 和 FB 有与调用它的块共享的输入/输出参数, 执行完 FC 和 FB 后, 将执行结果返回给调用它的代码块。

功能没有固定的存储区, 功能执行结束后, 其局部变量中的临时数据就丢失了。可以用全局变量来存储那些在功能执行结束后需要保存的数据。

设压力变送器量程的下限为 0 MPa, 上限为 *High* MPa, 经 A/D 转换后得到 0~27648 的整数。下式是转换后的数字 *N* 和压力 *P* 之间的计算公式:

$$P = (\text{High} \times N) / 27648 \quad \text{MPa} \quad (6-1)$$

用功能 FC1 实现上述运算, 在 OB1 中调用 FC1。

#### 2. 生成功能

打开 STEP 7 Basic 的项目视图, 生成一个名为 FB\_FC 的新项目 (见随书光盘中的同名例程)。双击项目树中的 “Add new device”, 添加一个新设备。CPU 的型号为 CPU 1214C。

打开项目视图中的文件夹 “\PLC\_1\Program block”, 双击其中的 “Add new block” (见图 6-1), 打开 “Add new block” 对话框 (见图 5-16), 点击其中的 “Function (FC)” (功能) 按钮, FC 默认的编号为 1, 语言为 LAD (梯形图)。设置功能的名称为 “Pressure”。点击 “OK” 按钮, 自动生成 FC1, 可以在项目树的文件夹 “\PLC\_1\Program block” 中看到新生成的 FC1。

#### 3. 生成功能的局部数据

将鼠标的光标放在 FC1 的程序区最上面的分隔条上, 按住鼠标的左键, 往下拉动分隔条, 分隔条上面是功能的接口 (Interface) 区 (见图 6-1 的右边), 下面是程序区。将水平分隔条拉至程序编辑器视窗的顶部, 不再显示接口, 但它仍然存在。

在接口区中声明 (即生成) 块专用的局部变量, 局部变量只能在它所在的块中使用。块的局部变量名称由字符和数字组成, 不能使用汉字。由图 6-1 可知, 功能有 5 种局部变量:

- 1) Input (输入参数): 由调用它的块提供的输入数据。
- 2) Output (输出参数): 返回给调用它的块的程序执行结果。
- 3) InOut (输入\_输出参数): 初值由调用它的块提供, 块执行后返回给调用它的块。
- 4) Temp (临时数据): 暂时保存在局部数据堆栈中的数据。只是在执行块时使用临时数据, 执行完后, 不再保存临时数据的数值, 它可能被别的块的临时数据覆盖。
- 5) Return 中的 Ret\_Val (返回值), 属于输出参数。

The screenshot shows the SIMATIC Manager software interface. On the left, there's a tree view of program blocks under 'Program blocks': 'Main [OB1]', 'Pressure [FC1]', 'Motor [FB1]', 'ValveCotrl [FB2]', 'GlobalDB1 [DB5]', 'MotorDB1 [DB1]', 'MotorDB2 [DB2]', 'ValveDB1 [DB3]', and 'ValveDB2 [DB4]'. On the right, a table titled 'Interface' lists local variables:

	Name	Data type	Comment
1	In_Data	Int	
2	High	Real	
3	Pressure	Real	
4	Temp	Real	
5	Temp1	Real	
6	Ret_Val	Void	

图 6-1 FC1 的局部变量

在 Input 下面的 Name 列输入参数 In\_Data，点击 Data type 列的 ▾ 按钮，用下拉式列表设置其数据类型为 Int (16 位整数)。用同样的方法生成输入参数 High、输出参数 Pressure 和临时变量 Temp1，它们的数据类型均为 Real。

在变量声明表中赋值时，不需要指定存储器地址；根据各变量的数据类型，程序编辑器自动地为所有局部变量指定存储器地址。

图 6-1 中的返回值 Ret\_Val 属于输出参数，默认的数据类型为 Void，该数据类型不保存数值，用于功能不需要返回值的情况。在调用 FC1 时，看不到 Ret\_Val。如果将它设置为 Void 之外的数据类型，在 FC1 内部编程时可以使用该变量，在调用 FC1 时可以在方框的右边看到作为输出参数的 Ret\_Val。

#### 4. FC1 的程序设计

首先用 CONV 指令将参数 In\_data 接收的 A/D 转换后的整数值 (0~27648) 转换为实数 (Real)，再用实数乘法指令和除法指令完成式 (6-1) 的运算 (见图 6-2)。运算的中间结果用临时局部变量 Temp1 保存。STEP 7 Basic 自动地在局部变量的前面添加#号，例如 “#Pressure”。



图 6-2 FC1 中的压力测量值计算程序

#### 5. 在 OB1 中调用 FC1

在变量表中生成调用 FC1 时需要的 3 个变量 (见图 6-3)，IW64 是 CPU 集成的 AI 点的通道 0 的地址。将项目树中的 FC1 拖放到右边的程序区的水平“导线”上 (见图 6-4)。FC1 的方框中左边的 In\_Data 等是在 FC1 的变量声明表中定义的输入参数，右边的 Pressure 是输出参数。它们被称为 FC 的形式参数，简称为形参。形参在 FC 内部的程序中使用，在别的逻辑块调用 FC 时，需要为每个形参指定实际的参数，简称为实参。实参与它对应的形参应具有相同的数据类型。

指定实参时，可以使用变量表和全局数据块中定义的符号地址或绝对地址，也可以是调用 FC1 的块 (例如 OB1) 的局部变量。

13	<input checked="" type="checkbox"/>	PressIn	Int	%IW64
14	<input checked="" type="checkbox"/>	PressOut	Real	%MD18
15	<input checked="" type="checkbox"/>	PressEN	Bool	%IO.6

图 6-3 PLC 变量表

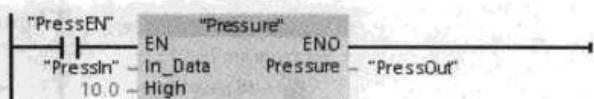


图 6-4 OB1 调用 FC1 的程序

块的 Output (输出) 和 InOut (输入/输出) 参数不能用常数来作实参，因为它们用来保存变量值，例如计算结果，因此应设置为地址。只有 Input (输入参数) 的实参能设置为常数。

## 6. 调用功能的实验

选中项目树中的 PLC 1，将组态数据和用户程序下载到 CPU，将 CPU 切换到 RUN 模式。

在 CPU 集成的模拟量输入的通道 0 的输入端输入一个 DC 0~10V 的电压，用程序状态功能监视 FC1 或 OB1 中的程序。调节该通道的输入电压，观察 MD18 中的压力计算值是否与理论计算值相同。

如果输入模拟量电压不太方便，可以将 OB1 调用 FC1 的程序（见图 6-4）中的输入参数 In\_Data 的实参（IW64）临时改为一个其他存储区中的字，例如 MW14。打开项目树中 PLC 的“Watch tables”文件夹，双击其中的“Add new watch table”，生成一个新的监视表，并在工作区自动打开它。在监视表中生成需要监视的 FC1 的输入、输出参数 MW14 和 MD18（见图 6-5），点击工具栏上的“”按钮，启动监视功能，“Monitor value”（监视值）列显示的是 CPU 中变量的实际值。在 MW14 的“Modify value”列输入修改值后，点击工具栏上的“”按钮，将修改值送入 CPU。接通 I0.6 对应的外接的小开关，使 FC1 的 EN 输入变为 1 状态，开始执行 FC1。分别将 27648 和 0 写入 MW14，MD18 输出的压力计算值应为 10.0 和 0.0 MPa，将某个中间值写入 MW14，FC1 通过 MD18 输出的压力计算值应与计算器计算出的理论值相同。

	Name	Address	Display format	Monitor value	Modify value	
1	"PressOut"	%MD18	Floating-point number	10.0		<input type="checkbox"/>
2	"Tag_1"	%MW14	DEC signed	27648	27648	<input checked="" type="checkbox"/>

图 6-5 用监视表测试程序

## 7. 为块提供密码保护

选中生成的 FC1，执行菜单命令“Edit”→“Know-how protection”→“Enable know-how protection”（使能知识秘诀保护），在打开的对话框中输入密码和密码的确认值。点击“OK”按钮后，项目树中 FC1 的图标上出现一把锁的符号，表示 FC1 受到保护。双击打开 FC1，可以看到接口区的变量，但是看不到程序区的程序。

选中生成的 FC1，执行菜单命令“Edit”→“Know-how protection”→“Change password”，在出现的对话框中输入密码后，可以修改密码。

选中生成的 FC1，执行菜单命令“Edit”→“Know-how protection”→“Disables know-how protection”（取消知识秘诀保护），在出现的对话框中输入密码，点击“OK”按钮后，项目树中 FC1 的图标上的锁消失，表示 FC1 的保护被取消。双击打开 FC1，又可以看到程序区中的程序。

### 6.1.2 生成与调用功能块

## 1. 功能块

功能块(FB)是用户编写的有自己的存储区(背景数据块)的块,FB的典型应用是执行

不能在一个扫描周期结束的操作。每次调用功能块时，都需要指定一个背景数据块。后者随功能块的调用而打开，在调用结束时自动关闭。功能块的输入、输出参数和静态变量（Static）用指定的背景数据块保存，但是不会保存临时局部变量中的数据。功能块执行完后，背景数据块中的数据不会丢失。

## 2. 生成功能块

打开项目树中的文件夹“\PLC\_1\Program block”，双击其中的“Add new block”，点击打开的对话框中的“Function block (FB)”按钮（见图 5-16），FB 默认的编号为 1，语言为 LAD（梯形图）。设置功能的名称为“Motor”。点击“OK”按钮，自动生成 FB1，可以在项目树的文件夹“\PLC\_1\Program block”中看到新生成的 FB1（见图 6-1）。

## 3. 生成功能块的局部变量

将鼠标的光标放在 FB1 的程序区最上面的分隔条上，按住鼠标的左键，往下拉动分隔条，分隔条上面是功能块的接口区（见图 6-6）。与功能相同，功能块的局部变量中也有 Input（输入）、Output（输出）、InOut（输入输出）和 Temp（临时）变量。

功能块执行完后，下一次重新调用它时，其 Static（静态）变量的值保持不变。

背景数据块中的变量就是其功能块的局部变量中的 Input、Output、InOut 和 Static 变量（见图 6-6 和图 6-7）。功能块的数据永久性地保存在它的背景数据块中，在功能块执行完后也不会丢失，以供下次执行时使用。其他代码块可以访问背景数据块中的变量。不能直接删除和修改背景数据块中的变量，只能在它的功能块的接口区中删除和修改这些变量。

Interface			
Name	Data type	Default value	
1 ▾ Input			
2 Start	Bool	false	
3 Stop	Bool	false	
4 TimePre	Time	T#0ms	
5 ▾ Output			
6 Motor	Bool	false	
7 Brake	Bool	false	
8 ▾ InOut			
9 ▾ Static			
10 ► TimerDB IEC_Timer			
11 ▾ Temp			

图 6-6 FB1 的接口区

MotorDB1					
	Name	Data type	Offset	Initial value	Retain
1 ▾ Input					
2 Start	Bool	0.0	false	<input type="checkbox"/>	
3 Stop	Bool	0.1	false	<input type="checkbox"/>	
4 TimePre	Time	2.0	T#0ms	<input type="checkbox"/>	
5 ▾ Output					
6 Motor	Bool	6.0	false	<input type="checkbox"/>	
7 Brake	Bool	6.1	false	<input type="checkbox"/>	
8 ▾ InOut					
9 ▾ Static					
10 ► TimerDB IEC_Timer		8.0		<input type="checkbox"/>	

图 6-7 FB1 的背景数据块

生成功能块的输入、输出参数和静态变量时，它们被自动指定一个默认值（Default value），可以修改这些默认值。变量的默认值被传送给 FB 的背景数据块，作为同一个变量的初始值（Initial value）。可以在背景数据块中修改变量的初始值。调用 FB 时没有指定实参的形参使用背景数据块中的初始值。

## 4. 编写 FB1 的程序

FB1 的控制要求如下：用输入参数 Start（起动按钮）和 Stop（停止按钮）控制输出参数 Motor（电动机）。按下停止按钮，断电延时定时器 TOF 开始定时，输出变量 Brake（制动）为 1 状态，经过输入参数 TimePre 设置的时间预置值后，停止制动。图 6-8 是 FB1 中的程序。TOF 的参数用静态变量 TimerDB 来保存，其数据类型为 IEC\_Timer。图 6-9 是 FB1 的接

口区中静态变量 TimerDB 内部的数据。

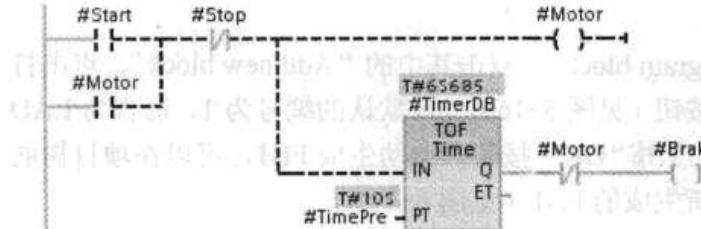


图 6-8 FB1 中的程序

Static		
TimerDB IEC_Timer		
START	Time	T#0ms
PRESET	Time	T#0ms
ELAPSED	Time	T#0ms
RUNNING	Bool	false
IN	Bool	false
Q	Bool	false
PAD	Byte	B#16#00
PAD_1	Byte	B#16#00
PAD_2	Byte	B#16#00
Temp		

图 6-9 定时器的数据结构

### 5. 在 OB1 中调用 FB1

在 PLC 变量表中生成两次调用 FB1 使用的符号地址（见图 6-10）。将项目树中的 FB1 拖放到程序区的水平“导线”上（见图 6-11）。在出现的“Call options”（调用选项）对话框中，输入背景数据块的名称。点击“OK”按钮，自动生成 FB1 的背景数据块。为各形参指定实参时，可以使用变量表中定义的符号地址。也可以使用绝对地址，然后在变量表中修改自动生成的符号的名称。

PLC tags				
	Name	Data type	Address	Retain
1	Start1	Bool	%I0.0	
2	Stop1	Bool	%I0.1	
3	Device1	Bool	%Q0.0	
4	Brake1	Bool	%Q0.1	
5	Start2	Bool	%I0.2	
6	Stop2	Bool	%I0.3	
7	Device2	Bool	%Q0.2	
8	Brake2	Bool	%Q0.3	

图 6-10 PLC 变量表

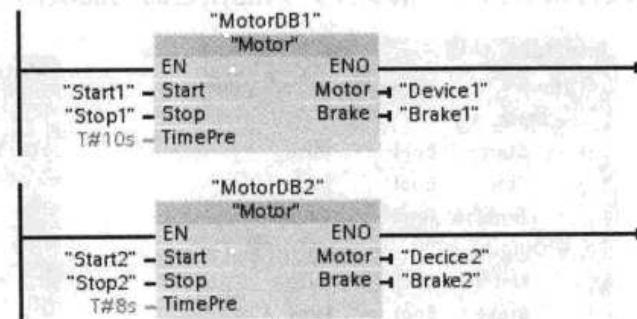


图 6-11 OB1 调用 FB1 的程序

### 6. 处理调用错误

FB1 最初没有输入参数 TimePro。在 OB1 中调用符号名为“Motor”的 FB1 之后，在 FB1 的接口区增加了输入参数 TimePro，OB1 中被调用的 FB1 的方框和字符变为红色（见图 6-12 中的左图）。点击程序编辑器的工具栏上的 按钮，或用鼠标右键点击出问题的块，执行出现的快捷菜单中的命令“Update block call”（更新块调用），出现图 6-12 所示的“Interface update”（接口更新）对话框，显示出原有的块接口和新的接口。点击“OK”按钮，OB1 中被调用的 FB1 被修改为新的接口（见图 6-12 中的右图），FB1 中的红色错误标记消失。

### 7. 调用功能块的实验

将程序块下载到 CPU 后，切换到 RUN 模式。拨动外接的小开关，模拟按钮的操作。分别用两台设备的起动按钮起动设备，然后用停止按钮使设备停车，可以看到两台设备的输出参数 Motor 和 Brake 按程序的要求变化，Brake 为 1 的时间与输入参数 TimePre 设置的相同。可以令两台设备分时工作，也可以令它们同时工作。在运行时可以用 OB1 的程序状态功能监

视被调用的 FB1 的输入、输出参数的状态，也可以在线监视 FB1 内部的程序的执行情况。

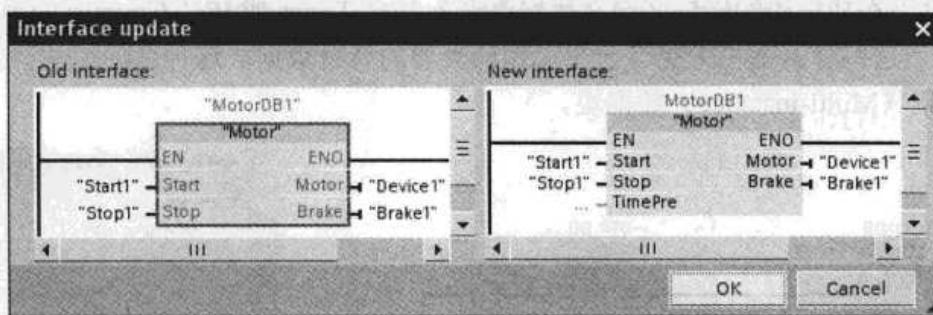


图 6-12 更新块的接口

## 8. 功能与功能块的区别

FB 和 FC 均为用户编写的子程序，接口区中均有 Input、Output、InOut 和 Temp 变量。FC 的返回值 Ret\_Val 实际上属于输出变量。下面是 FC 和 FB 的区别：

- 1) 功能块有背景数据块，功能没有背景数据块。
- 2) 只能在功能内部访问它的局部变量。其他代码块或 HMI（人机界面）可以访问功能块的背景数据块中的变量。
- 3) 功能没有静态变量，功能块有保存在背景数据块中的静态变量（Static）。功能如果有执行完后需要保存的数据，只能存放在全局变量（例如全局数据块和 M 区）中，但是这样会影响功能的可移植性。如果功能或功能块的内部不使用全局变量，只使用局部变量，不需要作任何修改，就可以将它们移植到其他项目。如果内部使用了全局变量，在移植时需要考虑块使用的全局变量是否会与别的块产生地址冲突。
- 4) 可以给功能块的局部变量（不包括 TEMP）设置初始值，不能给功能的局部变量设置初始值。在调用功能块时如果没有设置某些输入、输出参数的实参，将使用背景数据块中保存的数值。调用功能时应给所有的形参指定实参。

## 9. 组织块与 FB 和 FC 的区别

- 1) 对应的事件发生时，由操作系统调用组织块，FB 和 FC 是用户程序在代码块中调用的。
- 2) 组织块没有输入、输出变量和静态变量，只有临时局部变量。有的组织块自动生成的临时局部变量包含了与启动组织块的事件有关的信息，它们是由操作系统提供的。

### 6.1.3 功能块的多重背景数据块

打开项目 FB\_FC 的指令树中的文件夹 “\PLC\_1\Program block”，双击其中的“Add new block”，点击打开的对话框中的“Function block (FB)”按钮（见图 5-16），FB 默认的编号为 2，语言为 LAD（梯形图）。设置功能的名称为“ValveContrl”，点击“OK”按钮，自动生成 FB2。

FB2 用来控制卫生间的冲水阀，图 6-13 中的程序结构与图 3-58 中的相同。将鼠标的光标放在 FB2 的程序区最上面的分隔条上，按住鼠标的左键，往下拉动分隔条，在上面的接口区生成 FB2 的局部变量（见图 6-14）。Using 为有人使用卫生间的信号，Valve 是要控制的电磁阀。

每次调用定时器和计数器指令时，都需要指定一个背景数据块（见图 3-13）。如果这类指

令很多，将会生成大量的数据块“碎片”。为了解决这个问题，在功能块中使用定时器、计数器指令时，可以在功能块的接口区定义数据类型为 IEC\_Timer 或 IEC\_Counter 的静态变量（见图 6-14），用这些静态变量来提供定时器和计数器的背景数据。这种功能块的背景数据块被称为多重背景（Multi-instance）数据块。

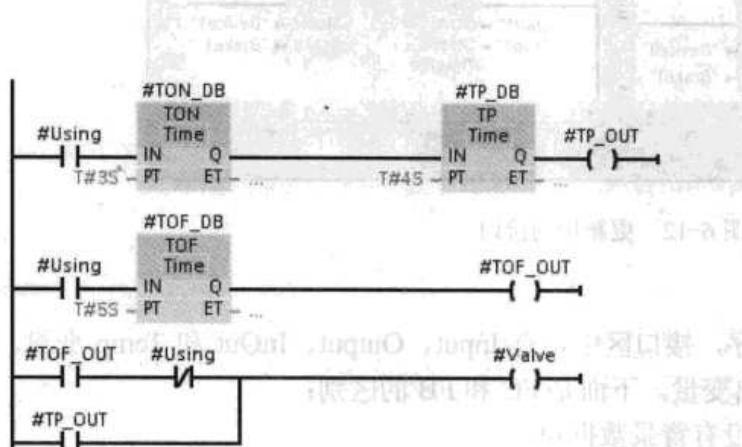


图 6-13 FB2 的程序

Name	Data type	Default value
1 ▼ Input		
2 Using	Bool	false
3 ▼ Output		
4 Valve	Bool	false
5 ▼ InOut		
6		
7 ▼ Static		
8 ▶ TON_DB	IEC_Timer	
9 ▶ TP_DB	IEC_Timer	
10 ▶ TOF_DB	IEC_Timer	
11 ▼ Temp		
12 TP_OUT	Bool	
13 TOF_OUT	Bool	

图 6-14 FB2 的接口区

这样多个定时器或计数器的背景数据被包含在一个背景数据块中，而不需要为每个定时器或计数器设置一个单独的背景数据块。因此减少了处理数据的时间，能更合理地利用存储空间。在共享的多重背景数据块中，定时器、计数器的数据结构之间不会产生相互作用。

在 PLC 变量表（见图 6-15）中定义调用 FB2 需要的变量，在 OB1 中两次调用 FB2（见图 6-16）。

9	Using1	Bool	%I0.4
10	Using2	Bool	%I0.5
11	Valve1	Bool	%Q0.4
12	Valve2	Bool	%Q0.5

图 6-15 PLC 变量表

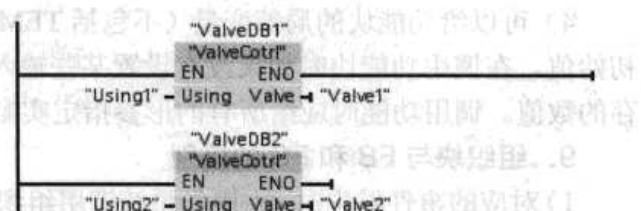


图 6-16 OB1 调用 FB2 的程序

将用户程序下载到 CPU，将 CPU 切换到 RUN 模式。拨动外接的小开关，模拟有人使用卫生间的信号 Using。可以看到输出参数 Valve 的状态按程序的要求变化，各段定时时间与 FB2 中设置的相同。可以令两次调用 FB2 的输入信号 Using1 和 Using2 几乎同时为 1 状态。在运行时可以用主程序 OB1 的程序状态功能监视被调用的 FB2 的输入、输出参数的状态，也可以在线监视 FB2 内部的程序的执行情况。

## 6.2 全局数据块与数据类型

### 6.2.1 全局数据块

数据块(Data blocks, DB)是用于存放执行代码块时所需的数据的数据区。与代码块不同，

数据块没有指令, STEP 7 Basic 按数据生成的顺序自动地为数据块中的变量分配地址。

有两种类型的数据块:

1) 全局 (Global) 数据块存储供所有的代码块使用的数据, 所有的 OB、FB 和 FC 都可以访问它们。

2) 背景数据块存储的数据供特定的 FB 使用。背景数据块中保存的是对应的 FB 的 Input (输入)、Output (输出)、InOut (输入输出) 和 Static (静态) 变量。FB 的临时数据没有用背景数据块保存。

5.1.4 节给出了生成全局数据块和在数据块中生成数组的方法。在项目 FB\_FC 中生成一个名为 GlobalDB1 的全局数据块 DB5 (见图 6-17), 在第 2 行生成一个名为 INC100ms 的无符号整数变量。选中设备视图中的 PLC\_1, 再选中监视窗口左边的 “System and clock memory” (见图 2-32), 用右边窗口的复选框设置默认的 MB0 作时钟存储器。在 OB1 中用 M0.0 产生的 10Hz 的时钟脉冲使绝对地址为 DB5.DBW0 的变量 INC100ms 加 1。

在 DB5 中还生成了一个名为 Generator 有 4 个元素的结构, 和名为 Power 的数组 (见图 6-17 和图 6-18)。结构和数组的 “Offset” 列是它们在数据块中的起始地址。结构的元素的 “Offset” 列是它们在结构中的地址偏移量。可以看出图 6-17 中的结构 Generator 占 8B。

Name	Data type	Offset	Initial value	Retain	Comment
1 ▾ Static					
2	INC100ms UInt	0.0	0	<input checked="" type="checkbox"/>	
3	▶ Generator Struct	2.0		<input checked="" type="checkbox"/>	
4	Current Int	0.0	0	<input checked="" type="checkbox"/>	
5	Voltage Int	2.0	0	<input checked="" type="checkbox"/>	
6	Speed Int	4.0	0	<input checked="" type="checkbox"/>	
7	Switch Bool	6.0	false	<input checked="" type="checkbox"/>	
8	▶ Power Array [0..23] of Int	10.0		<input checked="" type="checkbox"/>	

图 6-17 全局数据块

点击数据块窗口的工具栏上的 按钮, 在选中的变量的下面增加一个空白行, 点击工具栏上的 按钮, 在选中的变量的上面增加一个空白行。点击 按钮, 切换到扩展模式, 将显示或隐藏 “Default value” 列, 同时自动显示或隐藏结构和数组的元素。

选中项目树中的 PLC\_1, 将 PLC 的组态数据和块下载到 CPU, 将 CPU 切换到 RUN 模式。打开 DB5 后, 点击工具栏上的 按钮, 启动监视功能, 出现 “Monitor value” 列 (见图 6-18), 可以看到变量 INC100ms 的值在不断地增大。

Name	Data type	Offset	Default value	Initial value	Monitor value	Retain
1 ▾ Static						
2	INC100ms UInt	0.0	0	0	2565	<input checked="" type="checkbox"/>
3	▶ Generator Struct	2.0				<input checked="" type="checkbox"/>
4	▶ Power Array [0..23] of Int	10.0				<input checked="" type="checkbox"/>

图 6-18 在线的全局数据块

用鼠标右键点击项目树中的 DB5，执行出现的快捷菜单中的“Properties”命令，选中打开的对话框左边的 Attributes（属性）组（见图 6-19），再选中右边的复选框“Data block write-protected in the device”，可以使数据块具有写保护（只读）功能。

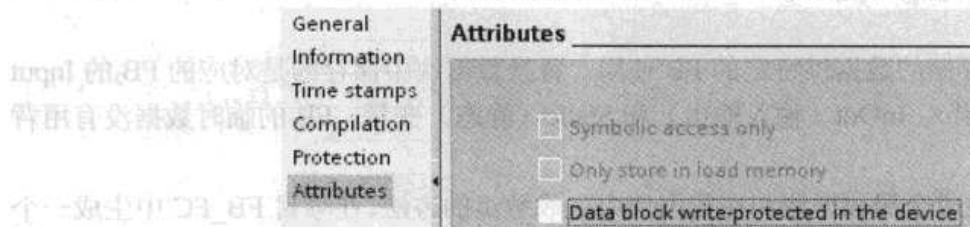


图 6-19 设置数据块的属性

数据块可以按位（例如 DBX3.5）、字节（DBB）、字（DBW）和双字（DBD）来访问。在访问数据块中的数据时，应指明数据块的名称，例如 DB1.DBW20。

## 6.2.2 数据类型

### 1. 数据类型的分类

必须为用户程序中使用的所有数据设置数据类型。可以使用下列的数据类型：

- 1) 基本数据类型：见 3.2.2 节。
- 2) 复杂数据类型：由基本数据类型组合而成。
- 3) 参数类型：用于定义传送到功能和功能块的参数。
- 4) 系统数据类型：由系统生成，可供用户使用，具有预定义的不能修改的结构。用于某些特定的指令，例如用于定时器的 IEC\_TIMER 数据类型。
- 5) 硬件数据类型：用于识别硬件元件、事件和中断 OB 等与硬件有关的对象。用户程序使用与模块有关的指令时，用硬件数据类型的常数来作指令的参数。

用户程序中的操作与特定长度的数据对象有关，例如位逻辑指令使用位（bit）数据，Move 指令使用字节、字和双字数据。

### 2. 复杂数据类型

复杂数据类型由其他数据类型组合而成，有下列 4 种复杂数据类型：

- 1) DTL：用于表示用日期和时间定义的时刻，包括年、月、日、星期、小时、分、秒和纳秒，其长度为 12B。5.6.1 节介绍了 DTL 的详细结构。
- 2) String：最多由 254 个字符组成的字符串，5.4.1 节介绍了 String 的结构。
- 3) Array：由固定个数的相同数据类型的元素组成的数组。5.1.4 节介绍了数组的生成和使用的方法。
- 4) Struct：由固定个数的元素组成的结构，其元素可以具有不同的数据类型。

PLC 变量表只能定义基本数据类型的变量，不能定义复杂数据类型的变量。可以在代码块的接口区或全局数据块中定义复杂数据类型的变量。

### 3. 结构的生成与结构元素的使用

在数据块 DB5 的第 3 行生成一个名为 Generator 的结构（见图 6-17），数据类型为 Struct。未生成结构的元素时，Struct 所在的单元的背景色为表示出错的粉红色。生成一个结构的元素后，其背景色变为正常的白色。输入完结构 Generator 的 4 个元素后，点击“Generator”左边

的▼按钮，它变为►（见图 6-18），同时结构的元素被隐藏起来。

在结构 Generator 的下面一行生成一个名为 Power 的数组（见图 6-18）。

下面是用符号地址表示结构中的元素的例子："GlobalDB1".Generator.Current。在指令中输入结构元素的地址时，首先点击表示被输入地址的红色<??>，再点击出现的▼按钮，双击出现的列表中的"GlobalDB1"，在它的后面输入一个英语的小数点后，双击出现的"GlobalDB1".Generator。在它的后面输入一个英语的小数点后，双击出现的"GlobalDB1".Generator.Current。也可以直接输入它的绝对地址 DB5.DBW2。

#### 4. 参数类型

在 FB 和 FC 中定义代码块之间传送数据的形式参数时，可以使用基本数据类型、复杂数据类型、系统数据类型和硬件数据类型，此外还可以使用参数类型。有两个参数数据类型：Variant 和 Void。

Variant 数据类型的参数是指向可变的变量或参数类型的指针。Variant 可以识别结构并指向它们，还可以指向结构变量的单个元件。在存储区中 Variant 参数类型变量不占用任何空间。

下面是使用符号地址的 Variant 数据类型的例子：MyDB.StructTag.FirstComponent，MyDB、StructTag 和 FirstComponent 分别是数据块、结构和结构的元素的符号地址，用小数点分隔它们。

下面是使用绝对地址的 Variant 数据类型的例子：P#DB10.DBX10.0 INT 12 和%MW10，前者相当于 S7-300/400 的数据类型 ANY，用来表示一个地址区，其起始地址为 DB10.DBW10，一共 12 个连续的 Int（整数）变量。

数据类型 Void 不保存数值，它用于功能不需要返回值的情况（见图 6-1 中 Ret\_Val 的数据类型）。

#### 5. 系统数据类型

系统数据类型由固定个数的元素组成，它们具有不能更改的不同的数据结构。系统数据类型只能用于某些特定的指令，表 6-1 给出了可以使用的系统数据类型和它们的用途。

表 6-1 系统数据类型

系统数据类型	字 节 数	描 述
IEC_Timer	16	用于定时器指令的定时器结构
IEC_SCounter	3	用于数据类型为 SInt 的计数器指令的计数器结构
IEC_USCounter	3	用于数据类型为 USInt 的计数器指令的计数器结构
IEC_UCounter	6	用于数据类型为 UInt 的计数器指令的计数器结构
IEC_Counter	6	用于数据类型为 Int 的计数器指令的计数器结构
IEC_DCounter	12	用于数据类型为 DInt 的计数器指令的计数器结构
IEC_UDCounter	12	用于数据类型为 UDInt 的计数器指令的计数器结构
ErrorStruct	28	编程或 I/O 访问错误的错误信息结构，用于 GET_ERROR 指令
CONDITIONS	52	定义启动和结束数据接收的条件，用于 RCV_GFG 指令
TCON_Param	64	用于指定存放 PROFINET 开放通信连接描述的数据块的结构
Void	—	该数据类型没有数值，用于输出不需要返回值的场合，例如可以用于没有错误信息时的 STATUS 输出

## 6. 硬件数据类型

硬件数据类型的个数与 CPU 的型号有关。指定的硬件数据类型常数与硬件组态时模块的设置有关。在用户程序中插入控制或激活模块的指令时，将使用硬件数据类型常数来作指令的参数。表 6-2 给出了可以使用的硬件数据类型和它们的用途。

表 6-2 硬件数据类型

数据类型	基本数据类型	描述
HW_ANY	Word	用于识别任意的硬件部件，例如模块
HW_IO	HW_ANY	用于识别 I/O 部件
HW_SUBMODULE	HW_IO	用于识别重要的硬件部件
HW_INTERFACE	HW_SUBMODULE	用于识别接口部件
HW_HSC	HW_SUBMODULE	用于识别高速计数器，例如用于 CTRL_HSC 指令
HW_PWM	HW_SUBMODULE	用于识别脉冲宽度调制，例如用于 CTRL_PWM 指令
HW_PTO	HW_SUBMODULE	用于在运动控制中识别脉冲传感器
AOM_IDENT	DWord	用于识别 AS 运行系统中的对象
EVENT_ANY	AOM_IDENT	用于识别任意的事件
EVENT_ATT	EVENT_ANY	用于识别可以动态地指定给一个 OB 的事件，例如用于 ATTACH 和 DETACH 指令
EVENT_HWINT	EVENT_ATT	用于识别硬件中断事件
OB_ANY	Int	用于识别任意的 OB
OB_DELAY	OB_ANY	出现时间延迟中断时，用于识别 OB 调用，例如用于 SRT_DINT 和 CAN_DINT 指令
OB_CYCLIC	OB_ANY	出现循环中断时，用于识别 OB 调用
OB_ATT	OB_ANY	用于识别可以动态地指定给事件的 OB，例如用于 ATTACH 和 DETACH 指令
OB_CYCLE	OB_ANY	用于识别可以指定给循环事件级别的事件的 OB
OB_HWINT	OB_ANY	出现硬件中断时，用于识别 OB 调用
OB_DIAG	OB_ANY	出现诊断错误中断时，用于识别 OB 调用
OB_TIMEERROR	OB_ANY	出现时间错误时，用于识别 OB 调用
OB_STARTUP	OB_ANY	出现启动事件时，用于识别 OB 调用
PORT	UInt	用于识别通信接口，用于点对点通信
CONN_ANY	Word	用于识别任意的连接
CONN_OUC	CONN_ANY	用于识别 PROFINET 开放通信的连接

变量表 (PLC tags) 的 Constants (常数) 选项卡列出了项目中的硬件数据类型的值 (见图 6-20)。其中的变量与项目中组态的硬件结构和组件的型号有关。

### 6.2.3 数据类型的转换

#### 1. 数据类型的转换方式

指令有关的操作数的数据类型应是协调一致的，这一要求也适用于块调用时的参数设置。如果操作数具有不同的数据类型，应对它们进行转换。有两种不同的转换方式。

1) 隐式转换：执行指令时自动地进行转换。

2) 显式转换：在执行指令之前使用转换指令进行转换。

PLC tags		PLC tags	Constants
	Name	Data type	Value
1	OB_Main	OB_FC/CLE	1
2	PROFINET_interface[PN]	Hw_Interface	64
3	HSC_1[HSC]	Hw_Hsc	1
4	HSC_2[HSC]	Hw_Hsc	2
5	HSC_3[HSC]	Hw_Hsc	3
6	HSC_4[HSC]	Hw_Hsc	4
7	HSC_5[HSC]	Hw_Hsc	5
8	HSC_6[HSC]	Hw_Hsc	6
9	AI2[AI]	Hw_SubModule	7
10	DI14/DI10[DI/DO]	Hw_SubModule	8
11	Pulse_1[PTO/PWM]	Hw_Pwm	9
12	Pulse_2[PTO/PWM]	Hw_Pwm	10
13	DI2/DO2_x_24VDC_1[DI/DO]	Hw_SubModule	11
14	Counting value equal reference value0	Event_HwInt	16#C0000001

图 6-20 PLC 变量表中的硬件数据类型

## 2. 隐式转换

如果操作数的数据类型兼容，将自动执行隐式转换。兼容性测试可以使用不同的标准：

- 1) 使用 IEC 检查（见图 6-21），采用严格的兼容性规则，指令有关的操作数必须具有相同的数据类型。

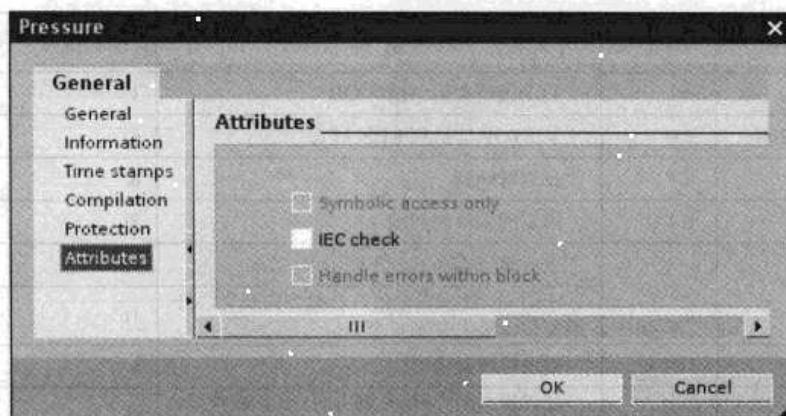


图 6-21 设置块的属性

- 2) 不使用 IEC 检查，兼容性测试采用不太严格的标准。不要求指令有关的操作数具有相同的数据类型，但是必须具有相同的数据位数，例如 16 位的数据类型 Int、UInt 和 Word。

Real 和 Time 之间的转换是例外，不允许这样的隐式转换。

## 3. 显式转换

操作数不兼容时，不能执行隐式转换，可以使用显式转换指令。转换指令在指令列表的“Math”、“String + Char”和“Convert”文件夹中。

显式转换的优点是可以检查出所有不符合标准的问题，并用 ENO 的状态指示出来。  
图 5-25 给出了两个数据类型转换的例子。

## 4. 设置 IEC 检查功能

如果激活了“IEC Check”，在执行指令时，将会采用严格的数据类型兼容性标准。

### (1) 设置对项目中所有新的块进行 IEC 检查

执行“Options”菜单中的“Settings”命令，选中出现的“Settings”对话框左边窗口的“PLC Programming”中的“General”组（见图 3-19），用复选框选中右边窗口“Default settings for new blocks”区中的“IEC Check”，新生成的块默认的设置将使用 IEC 检查。

### (2) 设置单独的块进行 IEC 检查

如果没有设置对项目中所有新的块进行 IEC 检查，可以设置对单独的块进行 IEC 检查。用右键点击项目树中的某个代码块，执行快捷菜单中的“Properties”命令，选中打开的对话框左边窗口的“Attributes”组（见图 6-21），用右边窗口中的“IEC Check”复选框激活这个块的 IEC 检查功能。保存项目时才保存这个设置。

## 6.3 中断事件与中断指令

### 6.3.1 事件与组织块

#### 1. 启动组织块的事件

组织块（OB）是操作系统与用户程序的接口，出现启动组织块的事件时，由操作系统调用对应的组织块。启动组织块的事件的属性见表 6-3。

表 6-3 启动 OB 的事件

事件类型	OB 编号	OB 个数	启动事件	队列深度	OB 的优先级	优先级组
程序循环	1 或 $\geq 200$	$\geq 1$	启动或结束前一循环 OB	1	1	1
启动	100 或 $\geq 200$	$\geq 0$	从 STOP 切换到 RUN 模式	1	1	
时间延迟	$\geq 200$	$\leq 4$	延迟时间到	8	3	
循环中断	$\geq 200$	$\leq 4$	固定的循环时间到	8	4	
硬件中断	$\geq 200$	$\leq 50$	上升沿 ( $\leq 16$ )、下降沿 ( $\leq 16$ )	32	5	2
			HSC 计数值=设定值，计数方向变化，外部复位，最大分别 6 个	16	6	
诊断错误	82	0 或 1	模块检测到错误	8	9	
时间错误	80	0 或 1	超过最大循环时间，调用的 OB 正在执行，队列溢出，因为中断负荷过高丢失中断	8	26	3

启动事件与程序循环事件不会同时发生，在启动期间，只有诊断错误事件能中断启动事件，其他事件将进入中断队列，在启动事件结束后处理它们。

#### 2. 不会启动 OB 的事件

表 6-4 不会启动 OB 的事件

事件级别	事件	事件优先级	系统反应
插入/拔出	插入/拔出模块	21	STOP
访问错误	刷新过程映像的 I/O 访问错误	22	忽略
编程错误	块内的编程错误	23	STOP
I/O 访问错误	块内的 I/O 访问错误	24	STOP
超过最大循环时间的两倍	超过最大循环时间的两倍	27	STOP

### 3. 事件执行的优先级与中断队列

优先级、优先级组和队列用来决定事件服务程序的处理顺序。

每个 CPU 事件都有它的优先级，不同优先级的事件分为 3 个优先级组。表 6-3 给出了各类事件的优先级、优先级组和队列深度。优先级的编号越大，优先级越高。时间错误中断具有最高的优先级 26 和 27。

事件一般按优先级的高低来处理，先处理高优先级的事件。优先级相同的事件按“先来先服务”的原则来处理。

高优先级组的事件可以中断低优先级组的事件的 OB 的执行，例如第 2 优先级组所有的事件都可以中断程序循环 OB 的执行，第 3 优先级组的时间错误 OB 可以中断所有其他的 OB。

一个 OB 正在执行时，如果出现了另一个具有相同或较低优先级组的事件，后者不会中断正在处理的 OB，将根据它的优先级添加到对应的中断队列排队等待。当前的 OB 被处理完后，再处理排队的事件。

当前的 OB 执行完后，CPU 将执行队列中最高优先级的事件的 OB，优先级相同的事件按出现的先后次序处理。如果高优先级组中没有排队的事件了，CPU 将返回较低的优先级组被中断的 OB，从被中断的地方开始继续处理。

不同的事件（或不同的 OB）均有它自己的中断队列和不同的队列深度（见表 6-3）。对于特定的事件类型，如果队列中的事件个数达到上限，下一个事件将使队列溢出，新的中断事件被丢弃，同时产生时间错误中断事件。

有的 OB 用它的临时局部变量提供触发它的启动事件的详细信息，可以在 OB 中编程，作出相应的反应，例如触发报警。

### 4. 中断的响应时间

中断的响应时间是指从 CPU 得到中断事件出现的通知，到 CPU 开始执行该事件的 OB 的第一条指令之间的时间。如果在事件出现时只是在执行程序循环 OB，中断响应时间小于  $175\mu s$ 。

## 6.3.2 组织块的实验

### 1. 循环执行组织块

需要连续执行的程序应放在主程序 OB1 中，CPU 在 RUN 模式时循环执行 OB1，可以在 OB1 中调用 FC 和 FB。如果用户程序生成了其他程序循环 OB，CPU 按 OB 编号的顺序执行它们，首先执行主循环程序 OB1，然后执行编号大于等于 200 的程序循环 OB。一般只需要一个程序循环组织块。

打开 STEP 7 Basic 的项目视图，生成一个名为“Interrupt”的新项目。双击项目树中的“Add new device”，添加一个新设备，CPU 的型号为 CPU 1214C。然后插入 AO 信号板。

打开项目视图中的文件夹“\PLC\_1\Program block”，双击其中的“Add new block”，点击打开的对话框中的“Organization block”（组织块）按钮（见图 6-22），选中“Program cycle”，生成一个程序循环组织块，OB 默认的编号为 200，语言为 LAD（梯形图）。块的名称为默认的 Main\_1。点击“OK”按钮，OB 块被自动生成，可以在项目树的文件夹“\PLC\_1\Program block”中看到新生成的 OB200。

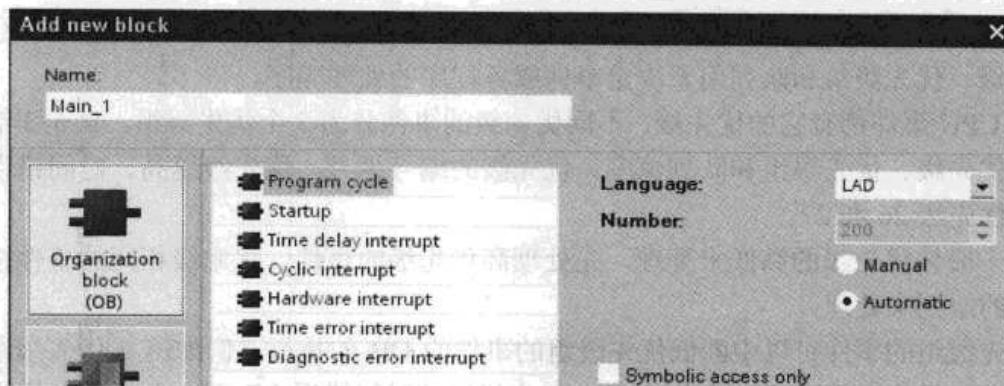


图 6-22 生成程序循环组织块

分别在 OB1 和 OB200 中输入简单的程序（见图 6-23 和图 6-24），将它们下载到 CPU，将 CPU 切换到 RUN 模式后，可以用 I0.4 和 I0.5 分别控制 Q1.0 和 Q1.1，说明 OB1 和 OB200 均被循环执行。

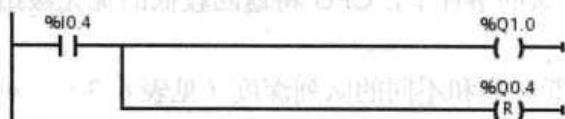


图 6-23 OB1 的程序

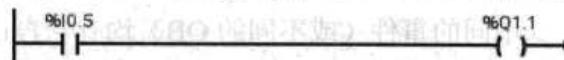


图 6-24 OB200 的程序

## 2. 启动组织块

启动组织块用于系统初始化，CPU 从 STOP 切换到 RUN 时，执行一次启动 OB。执行完后，开始执行程序循环 OB1。允许生成多个启动 OB，默认的是 OB100，其他启动 OB 的编号应大于等于 200。一般只需要使用一个启动组织块。

在项目 Interrupt 中，用上述方法生成启动（Startup）组织块 OB100 和 OB201。

分别在启动组织块 OB100 和 OB201 中生成初始化程序（见图 6-25 和图 6-26）。将它们下载到 CPU，将 CPU 切换到 RUN 模式后，可以看到 QB0 的值被 OB100 初始化为 7，其最低 3 位为 1。

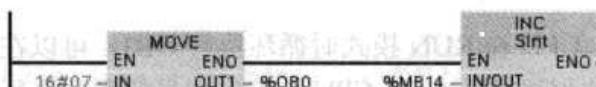


图 6-25 OB100 的程序

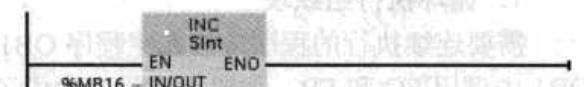


图 6-26 OB201 的程序

该项目的 M 区没有设置保持功能，暖启动时 M 区的存储单元的值均为 0。启动时分别调用了一次 OB100 和 OB201，INC 指令使 MB14 和 MB16 的值加 1。打开监视表，看到 MB14 和 MB16 的值均为 1。

## 3. 循环中断组织块

在设定的时间间隔，循环中断（Syclic interrupt）组织块被周期性地执行。最多可以组态 4 个循环中断事件，循环中断 OB 的编号大于等于 200。

在项目 Interrupt 中，用上述方法生成循环中断组织块 OB202（见图 6-27）。在 OB 的监视窗口的“Properties”选项卡中，循环中断的时间间隔（Scan time）的默认值为 100ms，将

它修改为 1000ms。相位偏移 (Phase shift, 默认值为 0) 用于错开不同时间间隔的几个循环中断 OB，使它们不会被同时执行，以减少连续执行循环中断 OB 的时间。

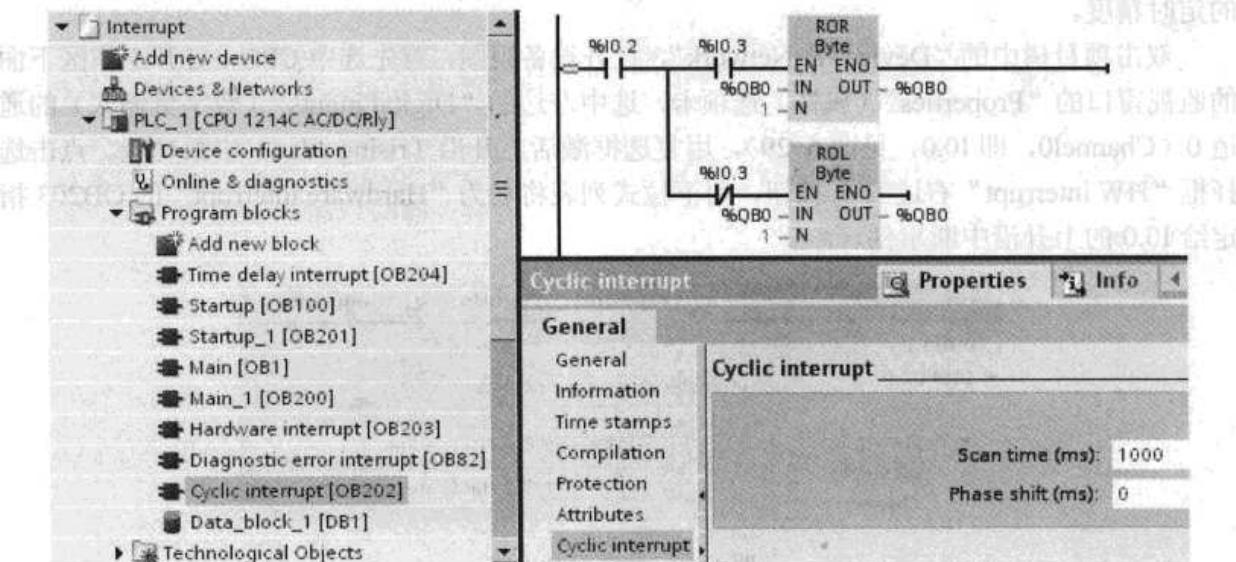


图 6-27 循环中断组织块 OB202

图 6-27 中的程序用于控制 8 位彩灯循环移位，I0.2 控制彩灯是否移位，I0.3 控制移位的方向。I0.3 为 0 状态时彩灯左移，为 1 状态时彩灯右移。

将代码块下载到 CPU，将 CPU 切换到 RUN 模式。通过 CPU 模块上输出点的 LED，可以观察到用 I0.2 和 I0.3 控制彩灯循环移位的情况。

#### 4. SRT\_DINT 与 CAN\_DINT 指令

定时器指令的定时误差较大，如果需要高精度的延时，可以使用时间延迟中断。在过程事件出现后，延时一定的时间再执行时间延迟 (Time delay) OB。在指令 SRT\_DINT 的 EN 使能输入的上升沿，启动延时过程。用该指令的参数 DTIME (1~60000ms) 来设置延时时间（见图 6-28）。在时间延迟中断 OB 中使用计数器，可以得到比 60s 更长的延迟时间。用参数 OB\_RN 来指定延迟时间到时调用的 OB 的编号，S7-1200 未使用参数 SIGN，可以设置任意的值。RET\_VAL 是指令执行的状态代码。

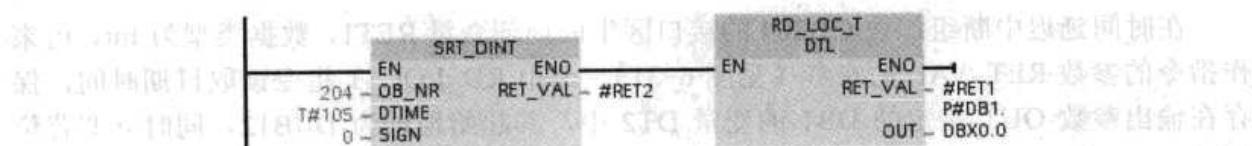


图 6-28 上升沿中断组织块 OB203 的程序

在 SRT\_DINT 启动的延时过程中，如果在 OB1 中调用指令 CAN\_DINT（见图 6-32）取消延迟，不会产生延迟中断事件和执行时间延迟 OB。

#### 5. 时间延迟组织块

可以组态最多 4 个延时中断事件，时间延迟 OB 的编号应大于等于 200。在项目 Interrupt 中生成硬件中断组织块 OB203、时间延迟组织块 OB204 和数据块 DB1。

在 I0.0 上升沿调用的 OB203 中启动时间延迟（见图 6-28），同时读取 PLC 的实时时间。

定时时间到时调用时间延迟组织块 OB204，再次读取实时时间（见图 6-31）。两次读取的实时时间的差值与时间延迟中断的输入参数 DTIME（定时时间）比较，可以得到时间延迟中断的定时精度。

双击项目树中的“Devices & Network”，打开设备视图，首先选中 CPU，打开工作区下面的监视窗口的“Properties”（属性）选项卡，选中左边的“Digital inputs”（数字量输入）的通道 0（Channel0，即 I0.0，见图 6-29），用复选框激活上升沿（rising edge）中断功能。点击选择框“HW interrupt”右边的▼按钮，用下拉式列表将名为“Hardware interrupt”的 OB203 指定给 I0.0 的上升沿中断事件。

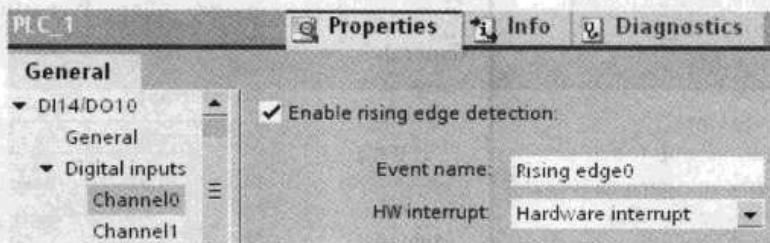


图 6-29 组态硬件中断

为了保存读取的日期时间值，在 DB1 中生成数据类型为 DTL 的变量 DT1 和 DT2（见图 6-30）。

在 I0.0 的上升沿中断组织块 OB203 的接口区生成局部变量 RET1 和 RET2，数据类型为 Int，用来作指令的输出参数 RET\_VAL（返回值）的实参（见图 6-28）。在梯形图中调用 SRT\_DINT，启动时间延迟中断的定时，同时调用指令 RD\_LOC\_T，读取 PLC 的实时时钟的日期时间，保存在输入参数 OUT 指定的 DB1 的变量 DT1 中，其起始地址为 DBB0。

Name	Data type	Offset	Initial value	Monitor value
1 ▼ Static				
2 ► DT1	DTL	0.0	DTL#1970-1...	2009-9-13-12:20 18.685575000
3 ► DT2	DTL	12.0	DTL#1970-1...	2009-9-13-13:20 28.685741000

图 6-30 数据块中的日期时间值

在时间延迟中断组织块 OB204 的接口区生成局部变量 RET1，数据类型为 Int，用来作指令的参数 RET\_VAL 的实参（见图 6-31）。调用 RD\_LOC\_T 指令读取日期时间，保存在输出参数 OUT 指定的 DB1 的变量 DT2 中，其起始地址为 DBB12。同时立即置位 Q0.4。

在 I0.1 为 1 时执行指令 CAN\_DINT（见图 6-32），取消时间延迟中断的定时。在 OB1 的接口区生成局部变量 RET1，用来作指令的参数 RET\_VAL 的实参。

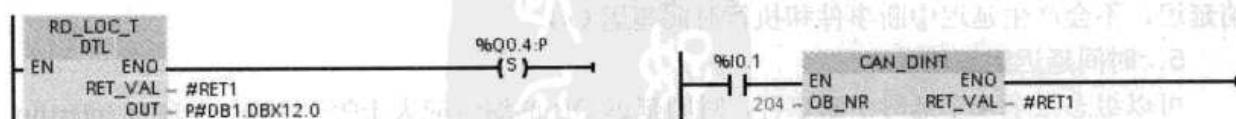


图 6-31 时间延迟中断组织块 OB204 的程序

图 6-32 OB1 中取消时间延迟的程序

将程序块和组态信息下载到 CPU，将 CPU 切换到 RUN 模式。用外接的小开关使 I0.0 变为 1 状态，CPU 调用硬件中断组织块 OB203。10s 后 SRT\_DINT 启动的定时时间到，CPU 调用时间延迟中断组织块 OB204，Q0.4 被立即置位。双击打开项目树中的 DB1，点击工具栏上的 按钮，启动在线监视功能。可以看到指令 SRT\_DINT 启动定时和定时时间到两次读取的实时时间的差值为 10.00017s（见图 6-30），定时精度是相当高的，其差值主要与变化的中断响应时间有关。

用 I0.4 外接的小开关产生一个脉冲信号，将 Q0.4 复位（见图 6-23）。用外接的小开关使 I0.0 变为 1 状态，CPU 调用硬件中断组织块 OB203，再次启动时间延迟中断的定时。在定时期间，用外接的小开关使 I0.1 变为 1 状态，调用指令 CAN\_DINT（见图 6-32），时间延迟中断被取消，不会调用 OB204，10s 的延迟时间到了后，Q0.4 不会变为 1 状态。

## 6. 诊断错误中断组织块 OB82

某些设备有检测和报告诊断错误的功能。如果激活了模块的诊断功能，在检测到诊断错误事件时，CPU 将会调用 OB82。只有 OB82 支持诊断错误事件。

下列情况将调用 OB82：有诊断功能的模块没有用户电源、输入信号超过模拟量模块的测量范围（上溢出和下溢出）、AO 模块输出电路断线和短路故障。

将 AO 信号板安装到 CPU 上，选中设备视图中 CPU 上的 AO 信号板，打开监视窗口的“Properties”（属性）选项卡（见图 6-33），选中左边的“Analog outputs”（模拟量输出）的通道 0（Channel0），设置该通道输出 4~20mA 的电流。用复选框激活断线（wire break）诊断功能。此外还有默认的上溢出和下溢出诊断被激活（不可更改）。

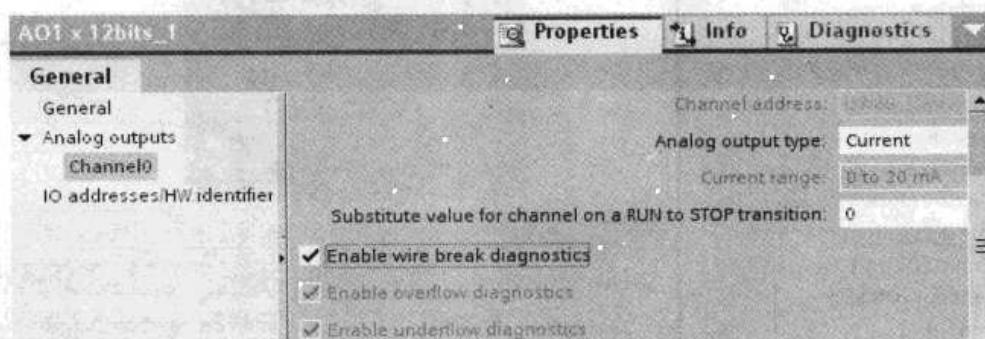


图 6-33 组态 AO 信号板的诊断功能

将程序块和组态信息下载到 CPU，CPU 切换到 RUN 模式。用监视表监视 AO 信号板的输出 QW80（见图 6-34），点击 按钮，启动监控功能。在它的 Modify value 列输入一个较大的值，点击 按钮，将它写入 CPU。断开 AO 的输出电路（见图 2-5），CPU 模块的 ERROR LED 和 AO 信号板的 0 号通道的红色 LED 闪动。接通 AO 模块的输出电路，错误 LED 熄灭。

	Name	Address	Display format	Monitor value	Modify value	
1	"Tag_14"	%MB12	Hex	03		
2		%QW80	Hex	1400	1400	<input checked="" type="checkbox"/> !

图 6-34 监视表

输出值较小时，断开输出电路时 PLC 不会出现故障。

## 7. 诊断缓冲区

双击 PLC\_1 文件夹中的“Online & diagnostics”（见图 6-36），在工作区打开在线与诊断视图。点击工具栏上的按钮 Go online，进入在线模式。选中图 6-35 左边窗口的“Diagnostics buffer”，右边窗口是诊断缓冲区。选中“Wire break”（断线）诊断事件，下面是事件的详细信息，“Incoming event”（进入的事件）表示事件刚发生。“Outgoing event”（离开的事件）表示事件刚结束。

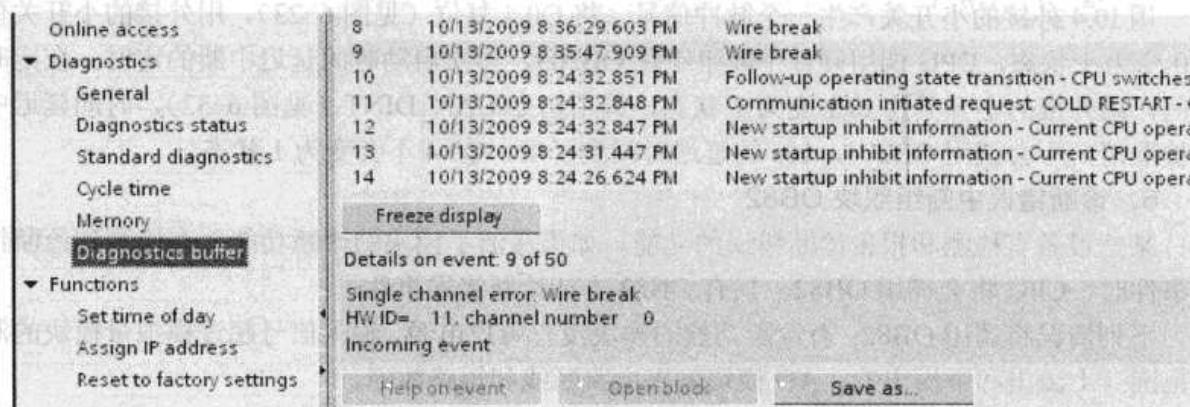


图 6-35 诊断缓冲区

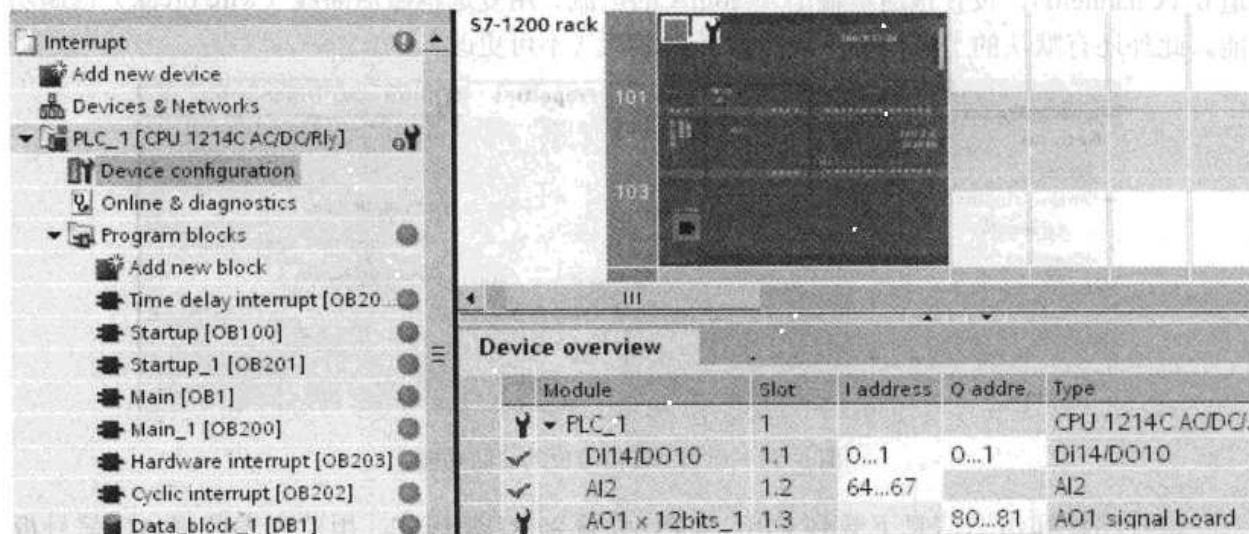


图 6-36 用在线的设备视图诊断故障

缓冲区中的条目包含事件被记录时的日期和时间，以及对事件的描述。条目按事件出现的顺序排列，最上面的是最后发生的事件。PLC 通电时诊断缓冲区最多保留 50 个条目，缓冲区装满后，新的条目将取代最老的条目。PLC 断电后，只保留 10 个最后出现的事件的条目。将 CPU 复位到工厂设置将删除缓冲区中的条目。

选中某个事件，诊断缓冲区下面是该事件的详细信息。

系统出现错误时，诊断事件可能非常快地连续不断地出现，将会使诊断缓冲区的显示以非常快的速率更新。点击“Freeze display”按钮，将会冻结显示，可以更好地查阅诊断事件

详细的信息。点击后该按钮上的字符变为“Cancel freeze”，点击它可以解除冻结。

点击“Help on event”按钮，将更详细地解释选择的事件和解决问题的方法。

点击“Open block”按钮，引起错误的指令所在的离线的块被打开，可以检查和修改块中的程序，然后将它下载到CPU。

点击“Save as”按钮，诊断缓冲区的内容被保存为文本文件，可以修改文件默认的名称。

## 8. 用在线的设备视图诊断故障

在线时可以看到项目树中项目 Interrupt 和 PLC\_1 站点右边的故障符号（红色的扳手，见图 6-36），各程序块右边的绿色小圆表示离线与在线的程序块一致。双击 PLC\_1 站点中的“Device configuration”，打开 PLC\_1 的设备视图，CPU 上也有故障符号。还可以看到下面的列表中 PLC\_1 和 AO 模块左边的故障符号。

如果将 AO 模块的输出类型设为±10V 的电压，可以设置短路诊断功能。

OB82 的启动信息提供了产生诊断错误事件的信息。可以在 OB82 中编程来读取启动信息，并采取相应的处理措施。表 6-5 中是 OB82 的启动信息。

表 6-5 OB82 的启动信息

输入	数据类型	描述
IO_state	Word	有诊断功能的模块的 I/O 状态
Laddr	HW_ANY	硬件组件的标识符（Word）
Channel	UInt	通道号（从 0 号通道开始）
Multi_error	Bool	如果错误不止一个时为 1 状态

可以在 OB82 中编写程序，保存 OB82 的局部变量，供进一步分析时使用。

在 OB82 中执行 INC 指令，将某个存储单元加 1。用监视表监视该存储单元，可以看到故障出现时调用一次 OB82，故障消失时又调用一次 OB82。

## 9. 用 DIS\_AIRT 与 EN\_AIRT 指令禁止与激活报警中断

使用 DIS\_AIRT（禁止报警中断，见图 6-37），可以延迟处理优先级高于当前 OB 的中断 OB。可以在一个 OB 中多次调用 DIS\_AIRT，DIS\_AIRT 的调用将由操作系统进行计数。这些调用中的每一个都将保持有效，直到调用 EN\_AIRT 指令取消延迟，或直到处理完当前 OB 为止。

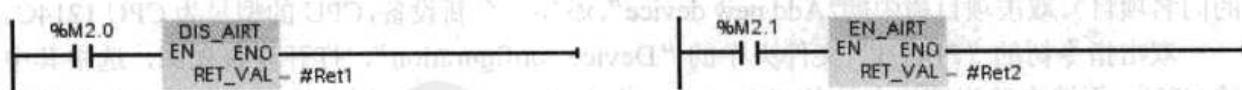


图 6-37 DIS\_AIRT 与 EN\_AIRT 指令

DIS\_AIRT 的参数 RET\_VAL 是延迟的次数，该参数为 0 表示没有延迟。

通过调用 EN\_AIRT（激活报警中断），可以启用以前调用 DIS\_AIRT 禁用的更高优先级的中断处理。为了取消所有的延迟，执行 EN\_AIRT 的次数应等于调用 DIS\_AIRT 的次数。

EN\_AIRT 的参数 RET\_VAL 是执行 EN\_AIRT 后，未取消的延迟次数，该参数为 0 表示所有的延迟被取消。

### 6.3.3 时间错误中断

#### 1. 时间错误中断组织块 OB80

循环时间是 CPU 的操作系统在 RUN 模式执行一次循环任务所需的时间，可以用 CPU 的监视窗口的属性选项卡设置最大循环时间（见图 2-36）。在图 7-12 的在线诊断视图右边的“Online tools”窗口中，可以看到最长、最短和当前的循环时间。

如果发生以下事件，系统调用时间错误中断组织块：

- 1) 实际的扫描循环时间超过设置的最大循环时间。
- 2) 请求执行循环中断或时间延迟中断，但是被请求的 OB 已经在执行。
- 3) 中断事件出现的速度比处理它们的速度还要快，对应的中断队列已满，导致中断队列溢出。
- 4) 因为中断负荷过高而丢失中断。

可以在优先级为 1 的程序循环 OB 和它调用的块中，用指令 RE\_TRIGR（重新触发循环时间监视，见 5.3.2 节）来重新启动监控定时器。

检测到时间错误时，CPU 将调用时间错误中断组织块 OB80。如果循环时间超过最大循环时间，并且下载了 OB80，CPU 将调用 OB80。如果没有下载 OB80，将忽略第一次超过循环时间的事件。

如果循环时间超过最大循环时间的两倍，并且没有执行 RE\_TRIGR 指令，不管是否有 OB80，CPU 将立即进入 STOP 模式。

OB80 的启动信息（见表 6-6）提供了产生时间错误事件的信息。可以在 OB80 中编程来检查启动信息，并采取相应的处理措施。

表 6-6 OB80 的启动信息

输入	数据类型	描述
fault_id	Byte	16#01：超过最大循环时间，16#02：被调用的 OB 正在执行，16#07：中断队列溢出，16#09：因为中断负荷过高丢失了中断
csg_OBnr	OB_ANY	错误发生时正在执行的 OB 的编号
csg_prio	UInt	错误发生时正在执行的 OB 的优先级

#### 2. 用跳转指令产生时间错误

打开 STEP 7 Basic 的项目视图，生成一个名为“HWInterrupt1”的新项目（见随书光盘中的同名项目）。双击项目树中的“Add new device”，添加一个新设备，CPU 的型号为 CPU 1214C。双击指令树的“PLC\_1”文件夹中的“Device configuration”，打开设备视图，选中其中的 CPU。再选中监视窗口左边的“System and clock memory”，用右边窗口的复选框设置默认的 MB1 作系统存储器字节（见图 2-32）。M1.2(Always on)总是为 1 状态，其常开触点总是闭合。

图 6-38 是 OB1 中用来演示 CPU 对时间错误的反应的程序。用外接的小开关使 I1.0 的常开触点闭合后马上断开，定时器输出一个宽度为 200ms 的脉冲，M20.0 的常开触点闭合。在此期间，反复执行 JMP 指令，跳转到标号 M1234 处。上述跳转过程是在一个扫描循环周期内完成的，因此扫描循环时间大于定时器的设定值（200ms），超过了 CPU 默认的循环时间设定值 150ms。

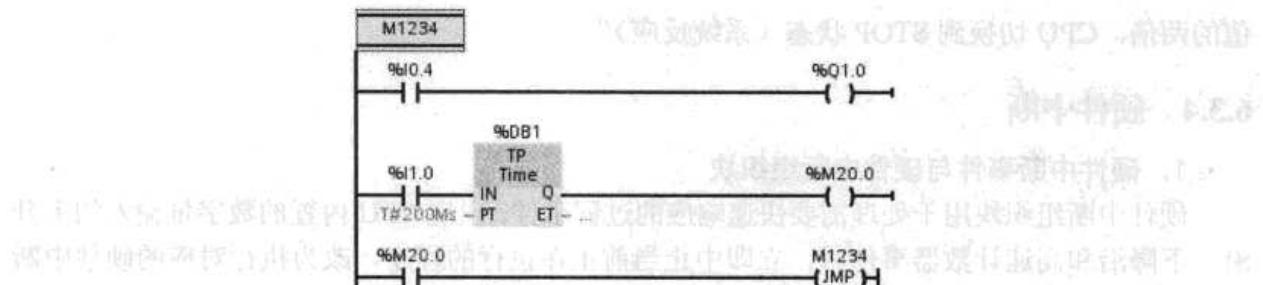


图 6-38 用于产生时间错误事件的电路

### 3. 循环时间超时的实验

将程序和组态信息下载到 CPU 后，切换到 RUN 模式。接通 I1.0 外接的小开关后马上断开它，脉冲定时器 TP 输出一个宽度为 200ms 的脉冲。因为循环时间超过设置的最大值，出现时间错误事件，CPU 的红色 ERROR LED 闪动 6 次后熄灭，仍然处于 RUN 模式。

双击指令树的“Program blocks”文件夹中的“Add new block”，点击出现的对话框中的“Organization block”按钮（见图 6-22），选中“Time error interrupt”，点击“OK”按钮，生成 OB80。在 OB80 中，用系统存储器字节的 M1.2 一直闭合的常开触点将 Q0.7 置位为 1。

将 OB80 下载到 CPU 后，切换到 RUN 模式。接通 I1.0 外接的小开关后马上断开它，出现时间错误事件，CPU 调用 OB80，Q0.7 对应的 LED 亮。CPU 的红色 ERROR LED 闪动 6 次后熄灭。

将图 6-38 中定时器的时间预置值 PT 修改为 400ms。将它下载到 CPU 后，切换到 RUN 模式。接通 I1.0 外接的小开关后马上断开它，出现时间错误事件，因为循环时间超过设置的最大值 150ms 的两倍，CPU 切换到 STOP 模式。

### 4. 诊断缓冲区中的信息

双击 PLC\_1 文件夹中的“Online & diagnostics”，在工作区打开在线与诊断视图。

点击工具栏上的按钮 Go online，进入在线模式。选中工作区左边窗口的“Diagnostics buffer”，右边窗口是诊断缓冲区（见图 6-39），其中的 33 号事件提供的信息为“超过最大循环时间，出现时间错误，请求调用 OB80”。选中该事件，下面是它的详细说明，给出了预置的最大循环时间（150ms），Incoming event 表示事件刚出现。

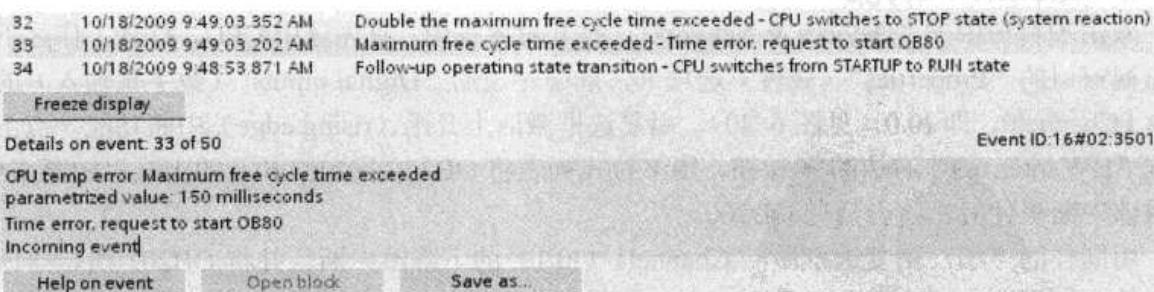


图 6-39 诊断缓冲区中的时间错误事件信息

33 号事件之后 150ms，诊断缓冲区出现 32 号事件。其信息为“循环时间超过设置的最大

值的两倍，CPU 切换到 STOP 状态（系统反应）”。

### 6.3.4 硬件中断

#### 1. 硬件中断事件与硬件中断组织块

硬件中断组织块用于处理需要快速响应的过程事件。出现 CPU 内置的数字量输入的上升沿、下降沿和高速计数器事件时，立即中止当前正在执行的程序，改为执行对应的硬件中断 OB。硬件中断组织块没有启动信息。

最多可以生成 50 个硬件中断 OB，在硬件组态时定义中断事件，硬件中断 OB 的编号应大于等于 200。S7-1200 支持下列硬件中断事件：

1) 上升沿事件，CPU 内置的数字量输入和 2 点信号板的数字量输入由 OFF 变为 ON 时，产生上升沿事件。

2) 下降沿事件，CPU 内置的数字量输入和 2 点信号板的数字量输入由 ON 变为 OFF 时，产生下降沿事件。

3) 高速计数器 HSC 1~6 的当前值等于设定值 (CV = RV)。

4) HSC 1~6 的方向改变，计数值由增大变为减小，或由减小变为增大。

5) HSC 1~6 的外部复位，某些 HSC 的数字量外部复位输入从 OFF 变为 ON 时，将计数值复位为 0。

#### 2. 硬件中断事件的处理方法

1) 给一个事件指定一个硬件中断 OB，这种方法最为简单方便，应优先采用。

2) 多个硬件中断 OB 分时处理一个硬件中断事件，需要用 DETACH 指令取消原有的 OB 与事件的连接，用 ATTACH 指令将一个新的硬件中断 OB 分配给硬件中断事件。

#### 3. 生成硬件中断组织块

打开项目视图中的文件夹 “\PLC\_1\Program block”，双击其中的“Add new block”，点击打开的对话框中的“Organization block”（组织块）按钮（见图 6-22），选中“Hardware interrupt”（硬件中断），生成一个硬件中断组织块，OB 的编号为 200，语言为 LAD（梯形图）。将块的名称设置为 Hardware interrupt1。点击“OK”按钮，OB 块被自动生成和打开，可以在项目树的文件夹 “\PLC\_1\Program block” 中看到新生成的 OB。用同样的方法生成名为 Hardware interrupt2 的 OB201。

#### 4. 组态硬件中断事件

双击项目树中的“Devices & Network”，打开设备视图，首先选中 CPU，打开工作区下面的监视窗口的“Properties”（属性）选项卡，选中左边的“Digital inputs”（数字量输入）的通道 0 (Channel0，即 I0.0，见图 6-40)，用复选框激活上升沿（rising edge）中断功能。点击选择框“HW interrupt”右边的 ▾ 按钮，用下拉式列表将 OB200 指定给 I0.0 的上升沿中断事件。出现该中断事件时，将会调用 OB200。

用同样的方法，用复选框激活 Channel1 (I0.1) 的下降沿中断，并将 OB201 指定给该中断事件。如果选中 OB 列表下面的“—”，没有 OB 连接到 I0.0 的上升沿中断事件。

#### 5. 编写 OB 的程序

在 OB200 和 OB201 中，分别用 M1.2 一直闭合的常开触点将 Q0.0:P 立即置位和立即复位（见图 6-41 和图 6-42）。

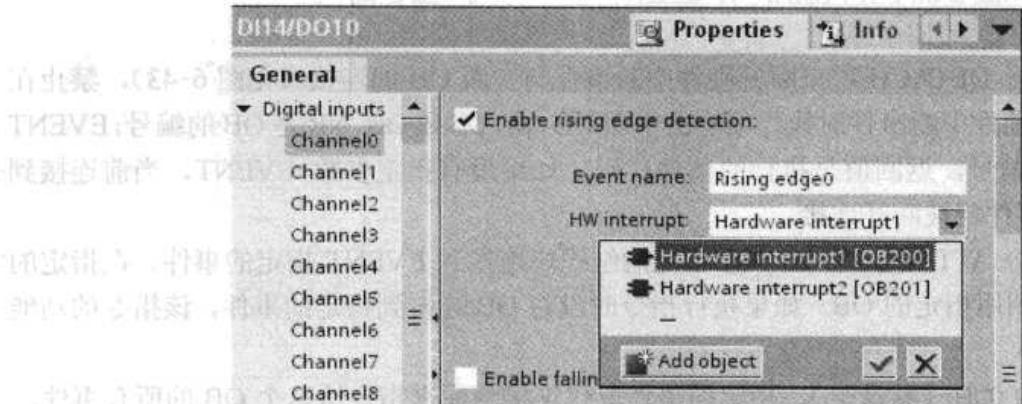


图 6-40 组态硬件中断

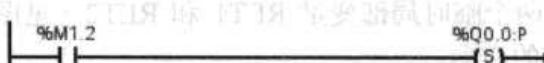


图 6-41 OB200 的程序

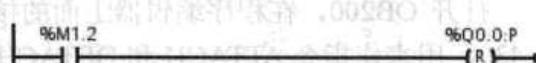


图 6-42 OB201 的程序

## 6. 实验结果

将组态信息和用户程序下载到 CPU，将 CPU 切换到 RUN 模式。用 I0.0 和 I0.1 外接的小开关产生硬件中断，在 I0.0 由断开变为接通（上升沿）时，Q0.0 被置位为 1。在 I0.1 由接通变为断开（下降沿）时，Q0.0 被复位为 0。

### 6.3.5 中断连接与中断分离指令

在下面的项目中，首先将硬件中断组织块 OB200 分配给 I0.0 的上升沿中断事件，该中断事件出现时，调用 OB200。在 OB200 中，用 DETACH 指令断开 I0.0 上升沿事件与 OB200 的连接，用 ATTACH 指令建立 I0.0 上升沿事件与 OB201 的连接。

下一次出现 I0.0 上升沿事件时，调用 OB201。在 OB201 中，用 DETACH 指令断开 I0.0 上升沿事件与 OB201 的连接，用 ATTACH 指令建立 I0.0 上升沿事件与 OB200 的连接。用这样的方法，可以用两个硬件中断 OB 分时处理一个硬件中断事件。

#### 1. 生成硬件中断组织块

打开 STEP 7 Basic 的项目视图，生成一个名为“HWInterrupt2”的新项目（见随书光盘中的同名项目）。双击项目树中的“Add new device”，添加一个新设备，CPU 的型号为 CPU 1214C。

打开项目视图中的文件夹“\PLC\_1\Program block”，双击其中的“Add new block”，点击打开的对话框中的“Organization block”（组织块）按钮（见图 6-22），选中“Hardware interrupt”，将块的名称设置为“Hardware interrupt1”，生成硬件中断组织块 OB200。用同样的方法生成名为 Hardware interrupt2 的 OB201。

#### 2. 组态硬件中断事件

首先选中设备视图中的 CPU，打开监视窗口的“Properties”（属性）选项卡，选中左边的“Digital inputs”（数字量输入）的通道 0（Channel0，即 I0.0，见图 6-40），用复选框激活上升沿中断功能。用下拉式列表将 OB200 指定给 I0.0 的上升沿中断事件。

#### 3. ATTACH 与 DETACH 指令

指令 ATTACH 和 DETACH 分别用于在 PLC 运行时建立和断开硬件中断事件与中断 OB

的连接。

中断分离指令 DETACH 用来断开硬件中断事件与中断 OB 的连接（见图 6-43），禁止在出现指令指定的硬件中断事件时执行指定的中断 OB。输入参数 OB\_NR 是 OB 的编号，EVENT 是指定的事件的编号，返回值是执行的条件代码。如果没有指定参数 EVENT，当前连接到 OB\_NR 的所有事件将被断开连接。

中断连接指令 ATTACH 将 OB\_NR 指定的组织块连接到 EVENT 指定的事件。在指定的事件发生时，将调用指定的 OB。如果执行指令时没有 OB 连接到指定的事件，该指令的功能被忽略。

参数 ADD 为 0 时（默认值），指定的事件取代连接到原来指定给这个 OB 的所有事件。

#### 4. 编写组织块的程序

打开 OB200，在程序编辑器上面的接口区生成两个临时局部变量 RET1 和 RET2（见图 6-43），用来作指令 ATTACH 和 DETACH 的返回值的实参。

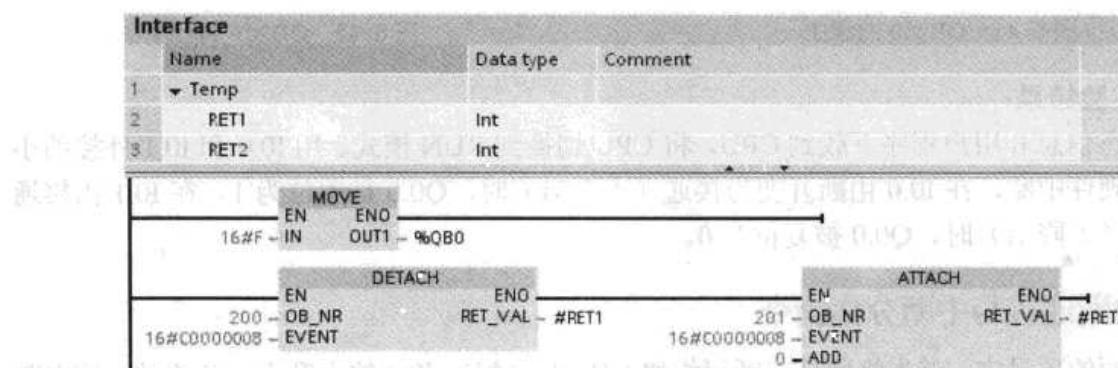


图 6-43 OB200 的程序

打开右边的“Extended Instruction”窗口的“Interrupts”文件夹，将其中的指令 DETACH 拖放到程序编辑器，双击参数 OB\_NR 左边的问号，然后点击出现的按钮（见图 6-44），出现的下拉式列表显示出已有的硬件中断 OB，设置 OB\_NR 的实参为 OB\_Hardware interrupt1（即 OB200）。用同样的方法设置参数 EVENT 的实参为 Rising edge0（代码为 16#C0000008）。DETACH 指令用来断开 I0.0 的上升沿中断事件与 OB200 的连接。

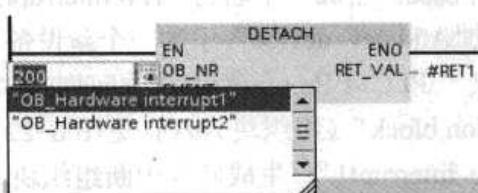


图 6-44 用下拉式列表设置指令的参数

用同样的方法生成指令 ATTACH，和设置它的参数，建立 I0.0 的上升沿中断事件与 OB201 的连接。

图 6-45 是 OB201 中的程序，断开 I0.0 的上升沿中断事件与 OB201 的连接后，建立起该中断事件与 OB200 的连接。

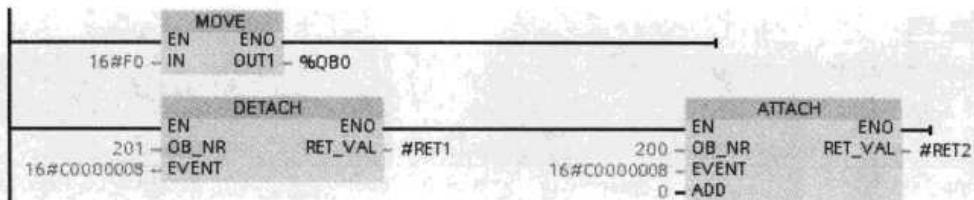


图 6-45 OB201 的程序

由于 OB200 和 OB201 中的 ATTACH 和 DETACH 指令的作用，进入 RUN 模式后，连续扳动 I0.0 外接的小开关。在 I0.0 奇数次的上升沿，QB0 被写入 16#F（低 4 位为 1），在 I0.0 偶数次的上升沿，QB0 被写入 16#F0（高 4 位为 1）。

## 6.4 交叉参考表与程序信息

### 6.4.1 交叉参考表

#### 1. 交叉参考表

交叉参考表（Cross-references list）提供用户程序中操作数和变量使用情况的概览。可以从交叉参考表直接跳转到使用操作数和变量的地方。

在程序测试和查错时，可以从交叉参考表获取下列消息：某个操作数在哪些块的哪个网络使用，用什么指令处理它；某个变量被用于哪个画面中的哪个元件；某个块被哪些块调用。

#### 2. 生成和显示交叉参考表

在项目视图中，可以显示下列对象的交叉参考：PLC 文件夹、程序块（Program blocks）文件夹、单独的块、PLC 变量表（PLC tags）和单独的变量。

可以用下述 3 种方法之一来生成和显示交叉参考表：

- 1) 选中上述某个对象后，点击工具栏上的 按钮。
- 2) 用右键点击项目树中的上述对象，执行快捷菜单中的命令“Cross-references”。
- 3) 选中上述某个对象后，执行 Tools 菜单中的“Cross-references”命令。

打开交叉参考表的“Used by”选项卡（见图 6-46），可以看到对象在什么地方被使用。打开“Uses”选项卡，可以看到显示的对象的使用者。

#### 3. PLC 变量表的交叉参考表

选中项目 PLC\_HMI 的项目树中的“PLC tags”，点击工具栏上的 按钮，生成 PLC 变量表的交叉参考表。选项卡“Used by”（被……使用）中的部分变量如图 6-46 所示。由表可知变量 Start 在 OB1 的网络 1 (NW1) 中被使用一次，在 HMI 的画面“Root screen”上被命名为“Button\_1”的按钮使用了两次。

Location 和 as 列中的字符均为蓝色和有下划线，表示有链接。点击图中的“OB1 NW1”，自动打开 OB1 中的网络 1，光标在变量 Start 处。点击图中的“Button\_1”，自动打开 HMI 的 Root screen 画面，使用变量 Start 的按钮被选中。点击图中“as”列的“Start”，自动打开 PLC tags，其中的变量 Start 被选中。

Cross-references of: PLC tags (5)

Used by	Uses					
Object ▲	Quantity Location as Access Address Type Path					
► Pump				%Q0.0	Bool	PLC_HMI\PLC_1\PLC tags
► Start				%M2.0	Bool	PLC_HMI\PLC_1\PLC tags
► Main	1			OB1	LAD-Organization	PLC_HMI\PLC_1\Program blocks
	OB1 NW1	Read-only				
► Root screen	2				Screen	PLC_HMI\HMI_1\Screens
	Button_1 Start					
	Button_1 Start					
► Stop				%M2.1	Bool	PLC_HMI\PLC_1\PLC tags

图 6-46 PLC 变量表的交叉参考表

点击“Object”(对象)列的表头，出现向上或向下的三角形(见图 6-46)，同时按对象名称的顺序升序或降序排列对象。点击“Address”列的表头，将按地址的顺序升序或降序排列对象。图中的“Quantity”列是对象被使用的次数，“Location”是使用的位置，“as”列是与对象有关的附加信息。“Access”是访问类型，“R”和“W”分别是读访问和写访问，“Read-only”为只读。“Address”是操作数的地址，“Type”是生成对象时使用的数据类型和其他信息，“Path”是对象在项目树中的路径。

工具栏上的 按钮用来更新交叉参考表，按钮 用复选框来分别选择是否显示所有使用的(Referenced)、未使用的(Unreferenced)、定义的(Existing)和未定义的(Not existing)对象。按钮 用来关闭下一层的对象，按钮 用来展开下一层的对象。

#### 4. 在监视窗口显示单个变量的交叉参考信息

选中程序区中的变量 Start，在下面的监视窗口的“Info”的“Cross-reference”选项卡(见图 6-47)中，可以看到选中的变量的交叉参考信息，与图 6-46 中的基本上相同。

The screenshot shows the SIMATIC Manager interface. At the top, there's a ladder logic diagram with three contacts labeled "Start", "Stop", and "Pump". Below the ladder logic is a "Properties" toolbar with buttons for "Start", "Stop", and "Pump". Underneath the toolbar is a menu bar with "File", "Edit", "View", "Search", "Tools", "Help", and "Database". The main area has tabs for "General", "Compile", "Cross-reference", and "Syntax". The "Cross-reference" tab is selected, showing the "Cross-reference information for Main" section. It contains a table with columns: Object, Location, Access, Address, Monitor value, and Type. The table lists the same objects as the table in Figure 6-46, with the "Start" object being the current selection.

Object	Location	Access	Address	Monitor value	Type
► Start			%M2.0		Bool
► Main			OB1		LAD-Orga.
► Root screen			OB1 NW1	Read-only	Screen
Button_1					
Button_1					

图 6-47 变量的监视窗口中的交叉参考

#### 5. 程序块的交叉参考表

选中另一个项目的项目树中的“Program blocks”，点击工具栏上的 按钮，生成程序块的交叉参考表。选项卡“Used by”(被……使用)如图 6-48 所示。由交叉参考表可知功能块

FB1 在 OB1 的网络 1 (NW1) 中被调用一次。全局数据块 DB1 中的变量 Actual 在 OB1 的网络 2 中被使用一次。打开 FB1 的背景数据块 DB2，可以看到其中的变量。

Cross-references of: Program blocks						
Used by	Uses	Object	Quan.	Location	as	Access
		Block_1				FB1
		Main	1			OB1
				OB1 NW1	Call	
		GlobeDB				DB1
		GlobeDB.Actual				Int
		Main	1			OB1
				OB1 NW2	Write	
		GlobeDB.Preset				Int
		Inst_DB				DB2
		Inst_DB				Instance DB of Block_1
		Inst_DB.MOTOR				Bool
		Inst_DB.START				Bool
		Inst_DB.STOP				Bool

图 6-48 程序块交叉参考表的 Used by 选项卡

程序块的交叉参考表的“Uses”（使用）选项卡如图 6-49 所示，可以看到各个代码块的内部使用变量的情况。例如在功能块 FB1 的 NW1（网络 1）中两次使用了局部变量 Motor。还可以看到主程序 OB1 中调用块和使用变量的情况。打开数据块，可以看到里面的变量。双击“Location”列中的链接“FB1 NW1”，自动打开 FB1 的 NW1。

Cross-references of: Program blocks						
Used by	Uses	Object	Quantity	Location	as	Access
		Block_1				FB1
		#MOTOR	2			Bool
				FB1 NW1	Write	
				FB1 NW1	Read-only	
		#STOP	1			Bool
		#START	1			Bool
		Inst_DB	1			DB2
		GlobeDB				DB1
		Inst_DB				Instance DB of Block_1
		Main				OB1
		Block_1	1			FB1
		GlobeDB.Actual	1			Int

图 6-49 程序块交叉参考表的 Uses 选项卡

## 6.4.2 分配表

用户的程序信息包括分配表、调用结构、附属结构和资源。

### 1. 显示分配表

分配表 (Assignment list) 提供 I、Q、M 存储区的地址位的概览，显示地址是否分配给

S7 程序（被程序访问），或者地址是否被分配给 S7 模块。

选中项目树中的“Program blocks”（程序块）文件夹，或选中其中的某个块，执行菜单命令“Tools”→“Assignment list”，将显示选中的程序的分配表（见图 6-50）。

## 2. 分配表中的图形符号

分配表的每一行对应于一个字节，每个字节由 0~7 位组成。B、W 和 DW 列的竖条用来表示程序使用了对应的字节、字和双字来访问地址，组成它们的位用灰色的正方形表示。例如 QB0 的 B 列的竖条表示程序不仅使用了 QB0 中的 0~5 位，还使用了 QB0, IB64~IB65 的 W 列的竖条表示使用了字 IW64, MB18~MB21 的 DW 列的竖条表示使用了双字 MD18。

图 6-50 的 M30.0 中的符号表示指针（Variant）的起始地址，在程序中使用了指针 P#M30.0 BYTE 10。

The screenshot shows the 'Assignment list' dialog in SIMATIC Manager. It has tabs for 'Call structure', 'Dependency structure', 'Assignment list' (which is selected), and 'Resources'. The main area displays two tables: 'Input/Output' and 'Memory'. The 'Input/Output' table shows assignments for addresses IB0, IB64, IB65, and QB0. The 'Memory' table shows assignments for MB2 through MB30. Below the tables is a 'Cross-reference' section for the 'Start1' block, listing OB1 and Main1 with their respective access types (OB1 is Read-only).

Object	Quantity	Location	as	Access	Address	Type	Path
Start1					%I0.0	Bool	ProgramInfo\PLC_1\PLC tags
Main1					OB1	LAD-Orga	ProgramInfo\PLC_1\Program blocks

图 6-50 分配表

点击表格上面的 按钮，将显示分配表中的图形符号列表（见图 6-51）。图中的“Bit access”为位访问，“Byte, Word, DWord access”为字节、字、双字访问，“Pointer access”为指针访问，“Bit and pointer access to the same bit”为对同一位的位访问和指针访问，“No hardware configured”为没有硬件组态，“Bit within byte, word, dword access”为字节、字、双字中的位。

## 3. 显示和设置 M 区的保持功能

点击工具栏上的 按钮，可以用打开的对话框（见图 6-52）设置 M 区从 MB0 开始的具有断电保持功能的字节数。点击工具栏上的按钮 ，可以激活或禁止显示 M 区地址的保持功能。有保持功能的 M 区的地址用地址列的符号 表示（见图 6-50）。将程序块下载到 CPU 后，M 区的保持功能起作用。

## 4. 分配表的附加功能

1) 选中分配表中的某个地址，例如图 6-50 选中了 I0.0，在下面的监视窗口的“Info>Cross-reference”选项卡中显示出选中的地址的交叉参考表。

2) 用鼠标右键点击分配表中的某个地址(包括位地址), 执行快捷菜单中的“Open editor”(打开编辑器)命令, 将会打开 PLC tags (PLC 变量表), 可以编辑变量的属性。

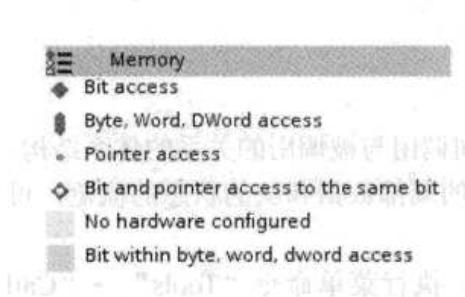


图 6-51 分配表中的图形符号列表

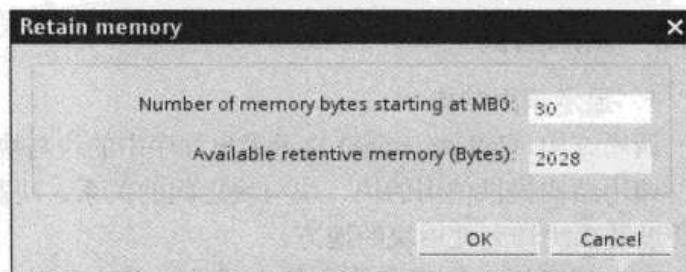


图 6-52 设置保持存储器

3) 点击工具栏上的按钮 , 出现的下拉式列表中有两个复选框:

复选框“Used addresses”用于激活或禁止显示已使用的地址。

复选框“Free hardware addresses”用于激活或禁止显示未使用的硬件地址, 建议禁止此选项。

## 5. 过滤器

可以使用预定义的过滤器(Filter)或生成自己的过滤器来“过滤”分配表显示的内容。

点击工具栏上的 按钮, 打开如图 6-53 所示的过滤器对话框, 用它来生成自己的过滤器。

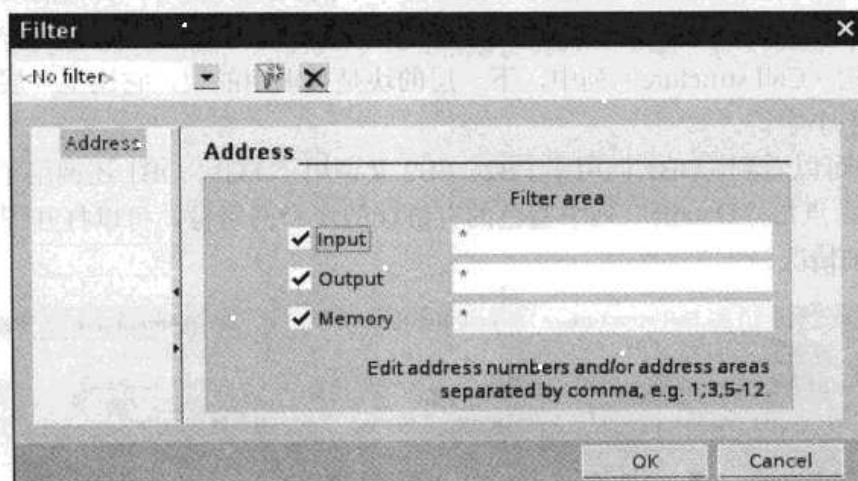


图 6-53 分配表的过滤器

点击 Filter 对话框工具栏上的 按钮, 生成一个新的过滤器。点击 按钮, 将删除当前的过滤器。点击工具栏上选择框右边的 按钮, 可在出现的下拉式列表中选择一个已有的过滤器。可以用选择框修改选中的过滤器的名称。

如果未选中图中的某个复选框, 分配表不显示对应的地址区。

可以在文本框中输入要显示的唯一的地址或部分地址, 例如在 M 区的文本框中输入 12 表示只显示 MB12; 输入“0;12;18”表示只显示 MB0、MB12 和 MB18; 输入“10-19”表示只显示 MB10~MB19 范围内已分配的地址; 输入“\*”表示显示该地址区所有已分配的地址。注意上述表达式应使用英语的标点符号。最后点击“OK”按钮, 确认对过滤器的编辑。

可以生成和编辑几个不同用途的过滤器，点击图 6-50 的工具栏上选择框右边的 ▾ 按钮，选中出现的下拉式列表中的某个过滤器，分配表按选中的过滤器的要求显示被过滤后的地址。

### 6.4.3 调用结构

#### 1. 显示调用结构

调用结构(Call structure)显示用户程序中的块与块之间调用与被调用的关系的体系结构。

调用结构提供使用的块、块与块之间的关系、块需要的局部数据和块的状态的概览，可以通过链接跳转到使用块的地方。

选中项目树中的程序块文件夹或选中其中的某个块，执行菜单命令“Tools”→“Call structure”，将显示选中的程序的调用结构。

#### 2. 调用结构的显示内容

在图 6-54 的调用结构表中，标有“!”的列用于显示调用的类型。Address 列显示块的绝对地址（即块的编号），功能块还包含对应的背景数据块的绝对地址。“Call frequency”列是调用同一个块的次数。点击 Detail 列中蓝色的有链接的块和网络的编号，可以打开程序编辑器，看到调用选中的块的详细情况。

只能用符号寻址的块需要较多的局部数据，因为符号寻址的信息保存在块里面。Local data (in the path)列显示完整的路径需要的局部变量。Local data (for blocks)列显示块需要的局部数据。

调用结构的第一层是组织块，它们不会被程序中的其他块调用。其他的块只是在没有被组织块直接或间接调用时才在第一层显示。

在调用结构(Call structure)列中，下一层的块是被调用的块，它比上一层的块（调用它的块）后退若干个字符。

从图 6-54 可以看到，OB1 调用了 FB2，FB2 又调用了 FC1。OB1 还调用了两次 FB1，调用了两次 FC2。点击“Details”列中蓝色的有链接的块和网络号，可以打开程序编辑器，看到调用块的详细情况。

Call structure		Dependency structure		Assignment list		Resources
<b>Call structure of PLC_1</b>						
Call structure	I	Address	Call frequency	Details	Local data (in path)	Local data (for blocks)
1 Main		OB1			0	0
2 Motor, MotorDB2		FB1, DB2	1	OB1 NM2 (0)	0	0
3 Motor, MotorDB1		FB1, DB1	1	OB1 NM1 (0)	0	0
4 ValveCotrl, ValveDB1		FB2, DB3	1	OB1 NM3 (0)	2	2
5 Pressure		FC1	1	FB2 NM2 (0)	6	4
6 GlobalDB1		DB5	1	OB1 NM4 (0)	0	0
7 Block_1		FC2	2	OB1 NM6 (0)	0	0
8 Cyclic interrupt		OB200			0	0
9 Block_2		FC3	1	OB200 NM1 (0)	0	0

Pressure						Properties	Info	Diagnostics					
General			Compile		Cross-reference								
Cross-references information for Pressure													
Object	Quantity	Location	as	Access	Address	Type	Path						
Pressure					FC1	LAD-Function	ProgramInfo\PLC_1\Program blocks						
ValveCotrl					FB2	LAD-Function block	ProgramInfo\PLC_1\Program blocks						

图 6-54 调用结构

选中调用结构中的 FC1，在下面的监视窗口的“Info”的“Cross-references”选项卡中可以看到 FC1 的交叉参考（见图 6-54），它被 FB2 调用。

用鼠标右键点击调用结构中的某个块，执行快捷菜单中的“Open editor”（打开编辑器）命令，将会打开程序编辑器，可以编辑选中的块。

### 3. 工具栏上按钮的功能

工具栏上的 按钮用来更新调用结构的显示，按钮 用来关闭所有的调用结构显示，点击某个块左边的 按钮，显示它调用的块。按钮 用来打开所有的块的调用结构。

点击 按钮，出现的下拉式列表中有两个复选框：

1) 如果激活了复选框“Display conflicts only”，仅显示调用中有冲突的块，例如调用了有时间标记冲突的块，使用修改了地址或数据类型的变量的块，调用被修改了接口的块，和没有直接、间接被 OB 调用的块。

2) 如果激活了复选框“Combine multiply calls”，几次块调用被组合到一行显示。块被调用的次数在“Call frequency”列显示（见图 6-54 中的第 7 行）。如果没有选中该复选框，将用两行来分别显示 OB1 中两次调用 FC2 的网络号。

工具栏上的 按钮用于检查块的一致性。如果 FB2 调用 FC1 之后，修改了 FC1 的一个输出参数的名称，在检查一致性之后，调用结构中 FC1 上面两层的 OB1 和 FB2 的图标均用红色显示。

点击图 6-54 第 5 行的 Details 列的 FB2 NW2，打开 FB2 的网络 2，用右键点击被调用的红色的 FC1，执行出现的“Update clock call”（更新块调用）命令，消除块调用的不一致性，FC1 上表示出错的红色消失。

改正错误后，重新编译程序。返回调用结构，再次点击 按钮，调用结构中 OB1 和 FB2 的颜色恢复正常。

## 6.4.4 附属结构与资源

### 1. 显示附属结构

附属结构（Dependency structure）是块在用户程序中被使用的情况的列表。块在第一级显示，调用或使用它的块在它的下面向右后退若干个字符。与调用结构相比，背景数据块被单独列出。附属结构是对象的交叉参考表的扩展。

选中程序块文件夹或选中其中的某个块，执行菜单命令“Tools”→“Dependency structure”，将显示选中的程序的附属结构。

在图 6-55 中，FC1 被 FB2 调用，FB2 的背景数据块为 DB3，FB2 又被 OB1 调用。点击“Details”列中蓝色有下划线（表示有链接）的块和网络号，可以打开程序编辑器，看到调用块的详细情况。标有“!”的列用于显示调用的类型。

### 2. 附属结构工具栏上的按钮

工具栏上的 按钮用来更新附属结构的显示，按钮 用来关闭所有的块的附属结构显示，点击某个块左边的 按钮，显示调用它的块。按钮 用来打开所有的块的附属结构。

工具栏上的 按钮和 按钮的功能与调用结构中的相同。

Call structure		Dependency structure		Assignment list		Resources	
<b>Dependency structure of PLC_1</b>							
1	► MotorDB1 -- Instance DB - Motor		DB1				
2	► MotorDB2 -- Instance DB - Motor		DB2				
3	► ValveDB1 -- Instance DB - ValveCotrl		DB3				
4	► GlobalDB1 -- Global DB		DB5				
5	▼ Pressure		FC1				
6	▼ ValveCotrl		FB2	1	FB2 MW2 (0)		
7	► ValveDB1 -- Instance DB - ValveCotrl	④	DB3				
8	► Main		OB1	1	OB1 MW3 (0)		
9	► Block_1		FC2				
10	► Block_2		FC3				
11	▼ Motor		FB1				
12	▼ MotorDB1 -- Instance DB - Motor	④	DB1				
13	► Main		OB1	1	OB1 MW1 (0)		
14	▼ MotorDB2 -- Instance DB - Motor	④	DB2				
15	► Main		OB1	1	OB1 MW2 (0)		
16	► Main		OB1	2	OB1 MW2 (0) ▾		
17	► ValveCotrl		FB2				
18	► Main		OB1				
19	► Cyclic interrupt		OB200				

图 6-55 附属结构

### 3. 附属结构的附加功能

用右键点击图 6-55 中的 FB1，执行快捷菜单中的“Display Usage”（显示应用）命令，在下面的监视窗口中可以看到块的交叉参考。

用鼠标右键点击附属结构中的某个块，执行出现的快捷菜单中的“Open editor”（打开编辑器）按钮，将打开选中的块。

### 4. 资源

资源（Resources）用于显示 CPU 的硬件资源，CPU 的存储区大小，组态的 I/O 和已使用的 I/O，以及 OB、FC、FB、DB、PLC tags 和用户定义的数据结构占用的存储器的详细情况。

选中程序块文件夹或选中其中的某个块，执行菜单“Tools”中的“Resources”命令，将显示 CPU 各存储区的资源（见图 6-56）。

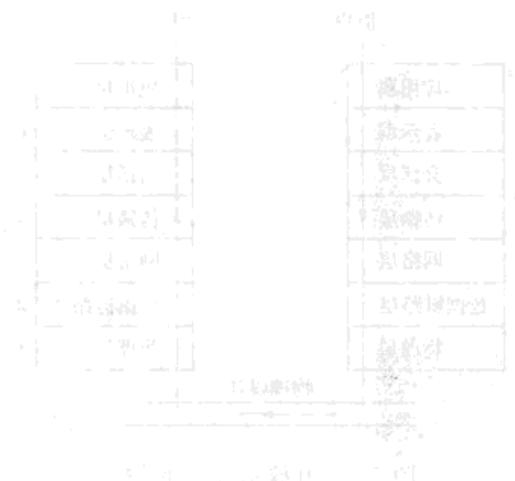
Call structure		Dependency structure		Assignment list		Resources			
Objects	Load memory	Work memory	Retentive memory	I/O	DI	DO	AI	AO	
1	0.45%	0.49%	4.3%		7.5%	7.14%	50%	7%	
2									
<b>Resources of PLC_1</b>									
3	Total:	2 MB	51200	2048	Configured:	80	28	2	0
4	Used:	9430	253	88	Used:	6	2	1	0
5	Details								
6	► OB	?	?						
7	► FC	1623	32						
8	► FB	2773	47						
9	► DB	5034	174	58					
10	PLC tags			30					

图 6-56 资源表

资源表中的“Load memory”是装载存储器，“Work memory”是工作存储器，“Retentive memory”是保持存储器。“Total”和“Used”分别是上述存储器的总字节数和已使用的字节数。“Configured”行和“Used”行分别是已组态和已使用的I/O。资源表还给出了使用的存储器和I/O区占总数的百分数。

第5行下面的“Details”区给出了OB、FC、FB和DB使用的存储器字节数和PLC的变量个数。未编译的块的大小用问号显示（见图6-56）。选中项目树中的程序块，点击工具栏上的 $\text{B}$ 按钮，程序块被成功地编译后，点击资源表工具栏上的 $\text{C}$ 按钮，图中OB行的问号被OB占用的字节数代替。

点击“Total”行、“Load memory”列对应的单元，可以用出现的下拉式列表设置装载存储器的字节数，可选1MB、2MB和24MB。



# 第7章 PLC的通信与自动化通信网络

## 7.1 计算机通信的国际标准

### 7.1.1 开放系统互连模型

如果没有一套通用的计算机网络通信标准，要实现不同厂家生产的智能设备之间的通信，将会付出昂贵的代价。

国际标准化组织（ISO）提出了开放系统互连模型 OSI，作为通信网络国际标准化的参考模型，它详细描述了软件功能的 7 个层次（见图 7-1）。

#### 1. 物理层

物理层的下面是物理媒体，例如双绞线、同轴电缆等。物理层为用户提供建立、保持和断开物理连接的功能，RS-232C、RS-422 和 RS-485 等就是物理层标准的例子。

#### 2. 数据链路层

数据以帧为单位传送，每一帧包含一定数量的数据和必要的控制信息，例如同步信息、地址信息、差错控制和流量控制信息。数据链路层负责在两个相邻节点间的链路上，实现差错控制、数据成帧、同步控制等。

#### 3. 网络层

网络层的主要功能是报文包的分段、报文包阻塞的处理和通信子网中路径的选择。

#### 4. 传输层

传输层的信息传递单位是报文（Message），它的主要功能是流量控制、差错控制、连接支持，传输层向上一层提供一个可靠的端到端（End-To-End）的数据传输服务。

#### 5. 会话层

会话层的功能是支持通信管理和实现最终用户应用进程之间的同步，按正确的顺序收发数据，进行各种对话。

#### 6. 表示层

表示层用于应用层信息内容的形式变换，例如数据加密/解密、信息压缩/解压和数据兼容，把应用层提供的信息变成能够共同理解的形式。

#### 7. 应用层

应用层作为 OSI 的最高层，为用户的应用服务提供信息交换，为应用接口提供操作标准。

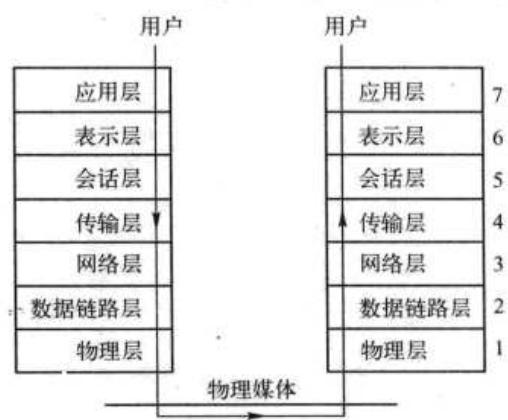


图 7-1 开放系统互连模型

## 7.1.2 IEEE 802 通信标准

### 1. 通信网络中的总线争用问题

在通信网络中，允许一个站发送，多个站接收，这种通信方式称为广播方式。

计算机以方波的形式将数据发送到通信线上，如果两个站或多个站同时向通信线发送数据，由于各个站发送的数据并不同步，通信线上出现的将会是乱七八糟的波形。因此数据通信的首要问题就是要避免两个站或多个站同时发送数据。

如果通信网络中的站点比较少，对通信的快速性要求不是太高，可以采用主从通信方式。网络中设置一个主站（例如上位计算机），其它的站（例如 PLC）均为从站。只有主站才有权主动发送请求报文（或称为请求帧），从站收到后返回响应报文。如果有多个从站，主站轮流向从站发出请求报文，这种通信方式称为轮询。

如果网络中的站点很多，轮询一遍需要的时间较长，某一从站遇到了需要紧急上传的事件，也要等到接收到主站的请求报文才能上传。

IEEE（国际电工与电子工程师学会）的 802 委员会于 1982 年颁布了一系列计算机局域网分层通信协议标准草案，总称为 IEEE 802 标准。它把 OSI 参考模型的底部两层分解为逻辑链路控制层（LLC）、媒体访问控制层（MAC）和物理传输层。前两层对应于 OSI 模型中的数据链路层，数据链路层是一条链路（Link）两端的两台设备进行通信时所共同遵守的规则和约定。

IEEE 802 的媒体访问控制层对应于三种已建立的标准，即带冲突检测的载波侦听多路访问（CSMA/CD）协议、令牌总线（Token Bus）和令牌环（Token Ring）。

### 2. CSMA/CD

CSMA/CD 通信协议的基础是 XEROX 等公司研制的以太网（Ethernet），各站共享一条广播式的传输总线，每个站都是平等的，采用竞争方式发送信息到传输线上，也就是说，任何一个站都可以随时广播报文，并为其他各站接收。当某个站识别到报文上的接收站名与本站的站名相同时，便将报文接收下来。由于没有专门的控制站，两个或多个站可能因为同时发送信息而产生冲突，造成报文作废，因此必须采取措施来防止冲突。

发送站在发送报文之前，先监听一下总线是否空闲，如果空闲，则发送报文到总线上，称之为“先听后讲”。但是这样做仍然有发生冲突的可能，因为从组织报文到报文在总线上上传输需要一段时间，在这一段时间中，另一个站通过监听也可能会认为总线空闲并发送报文到总线上，这样就会因为两个站同时发送而产生冲突。

为了防止冲突，在发送报文开始的一段时间，仍然监听总线，采用边发送边接收的办法，把接收到的信息和自己发送的信息相比较，若相同则继续发送，称之为“边听边讲”；若不相同则产生冲突，立即停止发送报文，并发送一段简短的冲突标志（阻塞码序列），来通知总线上的其他站点。为了避免冲突的站同时重发它们的帧，采用专门的算法来计算重发的延迟时间。通常把这种“先听后讲”和“边听边讲”相结合的方法称为 CSMA/CD（带冲突检测的载波侦听多路访问技术），其控制策略是竞争发送、广播式传送、载体监听、冲突检测、冲突后退和再试发送。

在以太网发展的初期，通信速率较低。如果网络中的设备较多，信息交换比较频繁，可能会经常出现竞争和冲突，影响信息传输的实时性。随着以太网传输速率的提高（100Mbit/s～

1000M bit/s)，这一问题已经基本解决。由于采取了一系列措施，工业以太网较好地解决了实时性问题。以太网在工业控制中得到了广泛的应用，大型工业控制系统最上层的网络几乎全部采用以太网。以太网将会越来越多地用于工业控制网络中的底层网络。

### 3. 令牌总线

IEEE 802 标准中的工厂媒质访问技术是令牌总线，其编号为 802.4。在令牌总线中，媒体访问控制是通过传递一种称为令牌的特殊标志来实现的。按照逻辑顺序，令牌从一个装置传递到另一个装置，传递到最后一个装置后，再传递给第一个装置，如此周而复始，形成一个逻辑环。令牌有“空”、“忙”两个状态，令牌网开始运行时，由指定站产生一个空令牌沿逻辑环传送。任何一个要发送信息的站都要等到令牌传给自己，判断为空令牌时才发送信息。发送站首先把令牌置成“忙”，并写入要传送的信息、发送站名和接收站名，然后将载有信息的令牌送入逻辑环传输。令牌沿逻辑环循环一周后返回发送站时，信息已被接收站复制，发送站将令牌置为“空”，送入逻辑环继续传送，以供其他站使用。

如果在传送过程中令牌丢失，由监控站向网内注入一个新的令牌。

令牌传递式总线能在很重的负荷下提供实时同步操作，传送效率高，适于频繁、较短的数据传输，因此它最适合于需要进行实时通信的工业控制网络系统。

### 4. 令牌环

令牌环媒体访问方案是 IBM 开发的，它在 IEEE 802 标准中的编号为 802.5，它有些类似于令牌总线。在令牌环上，最多只能有一个令牌绕环运动，不允许两个站同时发送数据。令牌环从本质上讲是一种集中控制式的环，环上必须有一个中心控制站负责网的工作状态的检测和管理。

## 7.1.3 现场总线及其标准

### 1. 现场总线的定义

IEC 对现场总线（Fieldbus）的定义是“安装在制造和过程区域的现场装置与控制室内的自动控制装置之间的数字式、串行、多点通信的数据总线称为现场总线”。它是当前工业自动化的热点之一。现场总线以开放的、独立的、全数字化的双向多变量通信代替 4~20mA 现场电动仪表信号。现场总线 I/O 集检测、数据处理、通信为一体，可以代替变送器、调节器、记录仪等模拟仪表，它不需要框架、机柜，可以直接安装在现场导轨槽上。现场总线 I/O 的接线极为简单，只需一根电缆，从主机开始，沿数据链从一个现场总线 I/O 连接到下一个现场总线 I/O。

使用现场总线后，可以减少自控系统的配线、安装、调试等方面的费用。操作员可以在中央控制室实现远程监控，对现场设备进行参数调整，还可以通过现场设备的自诊断功能预测故障和寻找故障点。

### 2. IEC 61158

由于历史的原因，现在有多种现场总线标准并存，IEC 的现场总线国际标准（IEC 61158）在 1999 年底获得通过，2000 年又补充了两种类型。

近年来，为了满足实时性应用的需要，各大公司和标准化组织纷纷提出了各种提升工业以太网实时性的解决方案，从而产生了实时以太网（Real Time Ethernet, RTE）。2007 年 7 月出版的 IEC 61158 第 4 版采纳了经过市场考验的 20 种现场总线（见表 7-1）。

表 7-1 IEC 61158 第 4 版的现场总线类型

类型	技术名称	类型	技术名称
类型 1	TS61158 现场总线, 原 IEC 技术报告	类型 11	TC net 实时以太网
类型 2	CIP 现场总线 (美国 Rockwell 公司支持)	类型 12	EtherCAT 实时以太网
类型 3	PROFIBUS 现场总线 (西门子公司支持)	类型 13	Ethernet Powerlink 实时以太网
类型 4	P-Net 现场总线 (丹麦 Process Data 公司支持)	类型 14	EPA 实时以太网
类型 5	FF HSE 高速以太网 (美国 Rosemount 公司支持)	类型 15	Modbus RTPS 实时以太网
类型 6	SwiftNet (波音公司支持, 已被撤消)	类型 16	SERCOS I、II 现场总线
类型 7	WorldFIP 现场总线 (法国 Alstom 公司支持)	类型 17	VNET/IP 实时以太网
类型 8	Interbus 现场总线 (德国 Phoenix contact 公司支持)	类型 18	CC-Link 现场总线
类型 9	FF H1 现场总线 (美国 Rosemount 公司支持)	类型 19	SERCOS III 实时以太网
类型 10	PROFINET 实时以太网 (西门子公司支持)	类型 20	HART 现场总线

其中的类型 1 是原 IEC 61158 第 1 版技术规范的内容, 类型 2 CIP 包括 DeviceNet、ControlNet 和实时以太网 Ethernet/IP。Type6 SwiftNet 因为市场应用很不理想, 已被撤消。

EPA (Ethernet for Plant Automation, 用于工厂自动化的以太网) 是我国拥有自主知识产权的实时以太网通信标准, 已被列入现场总线国际标准 IEC 61158 第 4 版的类型 14。

### 3. IEC 62026

IEC 62026 是供低压开关设备与控制设备使用的控制器电气接口标准, 于 2000 年 6 月通过。它包括:

IEC 62026-1: 一般要求。

IEC 62026-2: 执行器传感器接口 AS-i (Actuator Sensor Interface, 西门子公司支持)。

IEC 62026-3: 设备网络 DN (Device Network, 美国 Rockwell 公司支持)。

IEC 62026-4: Lonworks (Local Operating Networks) 总线的通信协议 LonTalk, 已取消。

IEC 62026-5: 智能分布式系统 SDS (Smart Distributed System, 美国 Honeywell 公司支持)。

IEC 62026-6: 串行多路控制总线 SMCB (Serial Multiplexed Control Bus, 美国 Honeywell 公司支持)。

## 7.2 西门子的工业自动化通信网络

PLC 的通信包括 PLC 之间、PLC 与上位计算机之间, 以及 PLC 与其他智能设备之间的通信。PLC 与计算机可以直接或通过通信处理器、通信链接器相连构成网络, 以实现信息的交换, 可以构成“集中管理、分散控制”的分布式控制系统, 满足工厂自动化系统发展的需要, 各 PLC 或远程 I/O 模块按功能各自放置在生产现场进行分散控制, 然后用网络连接起来, 构成集中管理的分布式网络系统。

### 7.2.1 工业以太网与 PROFINET

#### 1. 工业以太网

SIMATIC NET 的顶层为工业以太网 (见图 7-2), 它是基于国际标准 IEEE 802.3 的开放

式网络。以太网可以实现管理-控制网络的一体化，可以集成到互联网，为全球联网提供了条件。以太网在局域网(LAN)领域的市场占有率达到80%，通过广域网(例如ISDN或Internet)，可以实现全球性的远程通信。网络规模可达1024站，距离可达1.5km(电气网络)或200km(光纤网络)。符合IEEE802.3u标准的100Mbit/s的高速以太网的传输速率高，占用总线的时间极短。

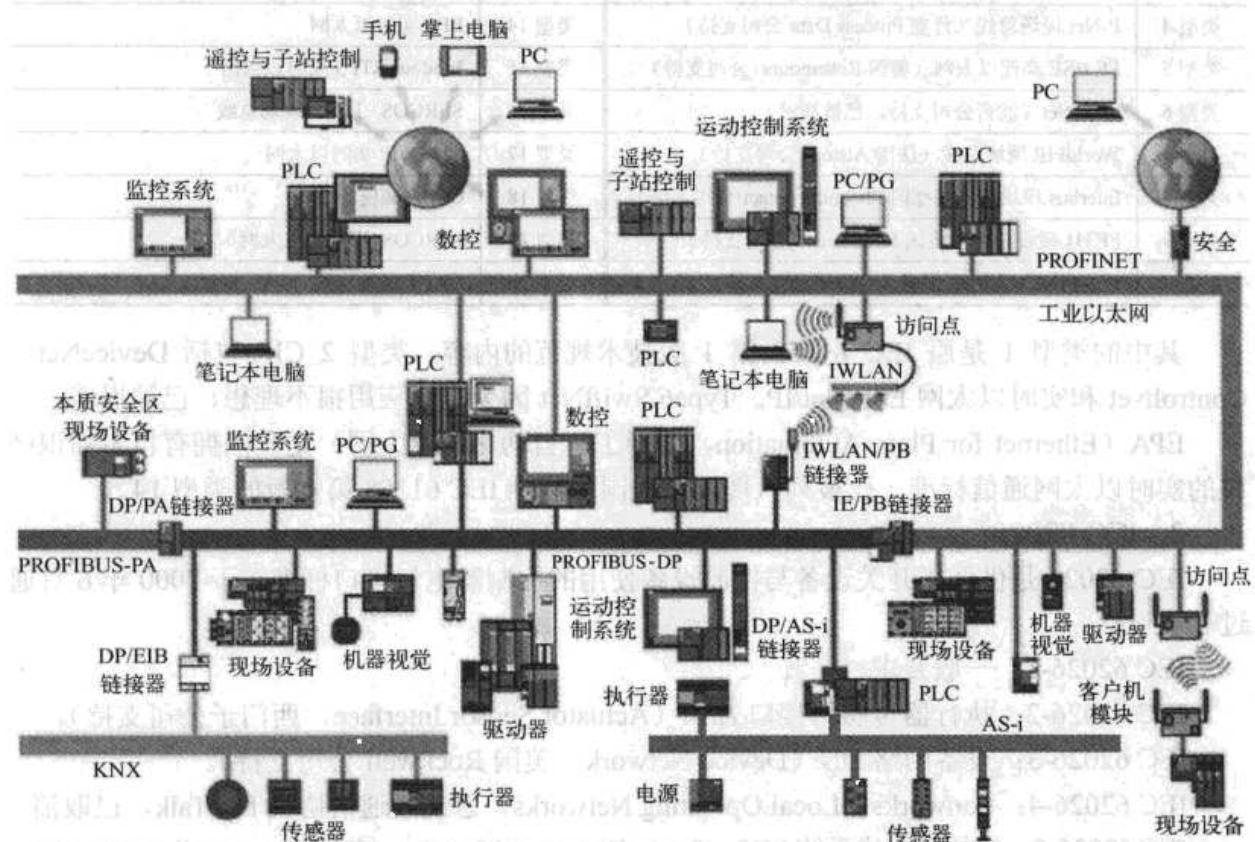


图 7-2 西门子的工业自动化通信网络

S7-1200 的 CPU 集成了一个 PROFINET 以太网接口，可以与编程计算机、人机界面和其他 S7 PLC 通信。

## 2. PROFINET

为了快速应对最新的市场需求，缩短产品面市的时间，需要提供从生产现场到工厂控制层和公司管理层的连续信息流，用于战略决策。现代生产对工厂纵向集成的要求越来越高，工业通信在自动化领域中的地位越来越重要。

PROFINET 是 PROFIBUS 国际组织（PI）推出的基于工业以太网的开放的现场总线标准（IEC 61158 中的类型 10）。使用 PROFINET，可以将分布式 IO 设备直接连接到工业以太网。PROFINET 可以用于对实时性要求更高的自动化解决方案，例如运动控制。

PROFINET 吸纳了多年积累的 PROFIBUS 和工业以太网的技术诀窍，采用开放的 IT 标准，与以太网的 TCP/IP 标准兼容，并提供了实时功能，能满足所有自动化的需求。PROFINET 能与现有的现场总线系统（例如 PROFIBUS）有机地集成（见图 7-3），无需改动现有设备的组态和编程。PROFINET 通过工业以太网，连接从现场层到管理层的设备，可以实现从公

司管理层到现场层的直接、透明的访问，PROFINET 融合了自动化世界和 IT 世界。

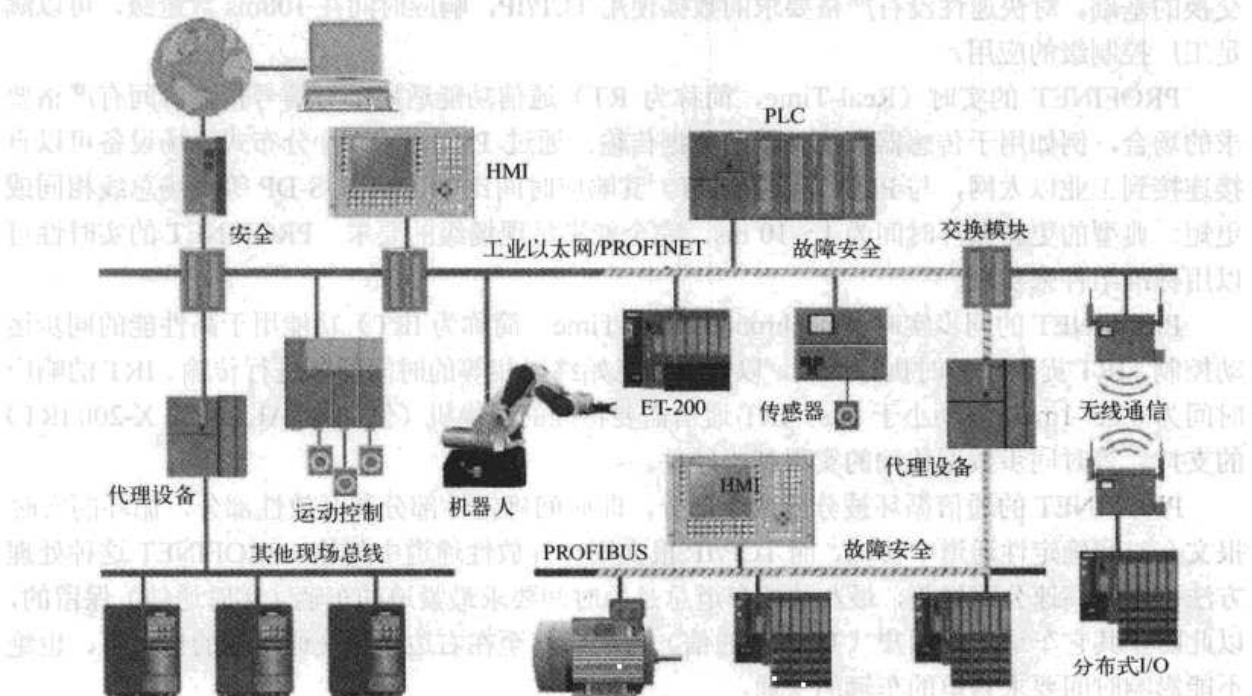


图 7-3 基于工业以太网的 PROFINET

PROFINET 技术的开放性给用户带来更多的选择余地。PROFINET 的设计和推广重点考虑了用户的易用性，用户使用 PROFINET 不需要太深的技术准备和成本。

可以很容易地从现有的 PROFIBUS 方案过渡到 PROFINET 解决方案，很好地整合已有的系统。通过代理服务器（Proxy），PROFINET 可以透明地集成现有的 PROFIBUS 设备，保护对现有系统的投资，实现现场总线系统的无缝集成。

使用 PROFINET IO，使现场设备可以直接连接到以太网，与 PLC 进行高速数据交换。PROFIBUS 各种丰富的设备诊断功能同样也适用于 PROFINET。

使用故障安全通信的标准行规 PROFIsafe，PROFINET 用一个网络可以同时满足标准应用和故障安全方面的应用。PROFINET 支持驱动器配置行规 PROFIdrive，后者为电气驱动装置定义了设备特性和访问驱动器数据的方法，用来实现 PROFINET 上的多驱动器运动控制通信。

PROFINET 已经在诸如汽车工业、食品、饮料以及烟草工业和物流工业等各种行业领域得到了广泛的应用。

PROFINET 为分布式自动化结构开辟了新的前景，基于组件的自动化（Component Based Automation, CBA）可以实现全厂范围项目的彻底模块化。由机械组件、电气/电子组件和应用软件组成智能模块，该模块可以预先测试，交付工厂后可以立即使用。SIMATIC iMap 用图形化方式配置独立的、可以重复使用的模块之间的 PROFINET 和 PROFIBUS 数据交换。界面友好，不需要另外编程。

### 3. PROFINET 在实时控制中的应用

PROFINET 使用以太网和 TCP/UDP/IP 作为通信基础，TCP/UDP/IP 是 IT 领域通信协议

事实上的标准。TCP/UDP/IP 提供了以太网设备通过本地和分布式网络的透明通道中进行数据交换的基础。对快速性没有严格要求的数据使用 TCP/IP，响应时间在 100ms 数量级，可以满足工厂控制级的应用。

PROFINET 的实时（Real-Time，简称为 RT）通信功能适用于对信号传输时间有严格要求的场合，例如用于传感器和执行器的数据传输。通过 PROFINET，分布式现场设备可以直接连接到工业以太网，与 PLC 等设备通信。其响应时间比 PROFIBUS-DP 等现场总线相同或更短，典型的更新循环时间为 1~10 ms，完全能满足现场级的要求。PROFINET 的实时性可以用标准组件来实现。

PROFINET 的同步实时（Isochronous Real-Time，简称为 IRT）功能用于高性能的同步运动控制。IRT 提供了等时执行周期，以确保信息始终以相等的时间间隔进行传输。IRT 的响应时间为 0.25~1ms，波动小于 1μs。IRT 通信需要特殊的交换机（例如 SCALANCE X-200 IRT）的支持。等时同步数据传输的实现基于硬件。

PROFINET 的通信循环被分成两个部分，即时间确定性部分和开放性部分，循环的实时报文在时间确定性通道中传输，而 TCP/IP 报文则在开放性通道中传输。PROFINET 这种处理方法可以与高速公路媲美，最左边的车道总是为时间要求最紧迫的车辆（实时通信）保留的，以此防止其它车道上的用户（TCP/IP 通信）占用。甚至在右边车道交通堵塞的情况下，也绝不能影响时间要求紧迫的车辆的交通。

PROFINET 能同时用一条工业以太网电缆满足三个自动化领域的需求，包括 IT 集成化领域、实时（RT）自动化领域和同步实时（IRT）运动控制领域，它们不会相互影响。

使用铜质电缆最多 126 个节点，网络最长 5km。使用光纤多于 1000 个节点，网络最长 150km。无线网络最多 8 个节点，每个网段最长 1000m。

## 7.2.2 现场总线 PROFIBUS 与 AS-i

西门子通信网络的中间层为工业现场总线 PROFIBUS，它是用于车间级和现场级的国际标准，传输速率最高 12Mbit/s，响应时间的典型值为 1ms，使用屏蔽双绞线电缆（最长 9.6km）或光缆（最长 90km），最多可以接 127 个从站。

PROFIBUS 是开放式的现场总线，已被纳入现场总线的国际标准 IEC 61158，并于 2006 年成为我国首个现场总线国家标准（GB/T 20540-2006）。

到 2007 年底为止，全球安装的 PROFIBUS 节点总数超过 1800 万个。

PROFIBUS 提供了 3 种通信协议：即 PROFIBUS-DP（Decentralized Periphery，分布式外部设备），PROFIBUS-PA（Process Automation，过程自动化）和 PROFIBUS-FMS（Fieldbus Message Specification，现场总线报文规范）。

### 1. PROFIBUS-FMS

PROFIBUS-FMS 主要用于系统级和车间级的不同供应商的自动化系统之间传输数据，处理单元级（PLC 和 PC）的多主站数据通信，FMS 已基本上被以太网取代，现在很少使用。

### 2. PROFIBUS-DP

PROFIBUS-DP 特别适合于 PLC 与现场级分布式 I/O（例如西门子的 ET 200）设备之间的通信。主站之间的通信为令牌方式，主站与从站之间为主从方式，以及这两种方式的组合。

### 3. PROFIBUS-PA

PROFIBUS-PA 用于过程自动化的现场传感器和执行器的低速数据传输。由于传输技术采用 IEC 1158-2 标准，确保了本质安全和通过总线对现场设备供电，可以用于防爆区域的传感器和执行器与中央控制系统的通信。使用 DP/PA 链接器可以将 PROFIBUS-PA 设备很方便地集成到 PROFIBUS-DP 网络中。PROFIIBUS-PA 使用屏蔽双绞线电缆，由总线提供电源。

此外基于 PROFIBUS，还推出了用于运动控制的总线驱动技术 PROFI-drive 和故障安全通信技术 PROFI-safe。

### 4. 现场总线 AS-i

西门子通信网络的底层包括 AS-i 和 EIB，EIB 是楼宇安装总线系统，符合国际标准 CEN TC247。

AS-i 是执行器-传感器接口（Actuator Sensor Interface）的简称，是传感器和执行器通信的国际标准（IEC 62026-2），响应时间小于 5ms，使用未屏蔽的双绞线，由总线提供电源，最长通信距离为 300m，最多 62 个从站。特别适合于连接需要传送开关量的传感器和执行器。

AS-i 属于主从式网络，每个网段只能有一个主站。主站是网络通信的中心，负责网络的初始化，以及设置从站的地址和参数等。AS-i 从站是 AS-i 系统的输入通道和输出通道，它们仅在被 AS-i 主站访问时才被激活。接到命令时，它们触发动作或者将现场信息传送给主站。

## 7.3 S7-1200 的以太网通信

### 7.3.1 以太网设备的地址

#### 1. MAC 地址

在 OSI（开放系统互连）7 层网络协议参考模型中，第 2 层（数据链路层）由 MAC（Media Access Control，媒体访问控制）子层和 LLC（逻辑链路控制）子层组成。

MAC 地址也叫做物理地址、硬件地址或链路地址。MAC 地址是识别 LAN（局域网）节点的标识，即以太网接口设备的物理地址。它通常由设备生产厂家写入 EEPROM 或闪存芯片，在传输数据时，用 MAC 地址标识发送和接收数据的主机的地址。在网络底层的物理传输过程中，通过 MAC 地址来识别主机。MAC 地址是 48 位二进制数，通常分为 6 段（6 个字节），一般用十六进制数表示，例如 00-05-BA-CE-07-0C。其中的前 6 位十六进制数是网络硬件制造商的编号，它由 IEEE（电气与电子工程师协会）分配，后 6 位十六进制数代表该制造商制造的某个网络产品（例如网卡）的系列号。形象的说，MAC 地址就像我们的身份证号码，具有全球唯一性。

在 Windows XP 中，执行菜单命令“开始”→“运行”，在出现的“运行”对话框中输入“CMD”后按〈Enter〉键，在出现的 DOS 窗口中输入命令行“ipconfig /all”后按〈Enter〉键，将显示出计算机网卡的物理地址（即 MAC 地址）、IP 地址和子网掩码等。

MAC 地址是以太网包头的组成部分，以太网交换机根据以太网包头中的 MAC 源地址和 MAC 目的地地址实现包的交换和传递。

#### 2. IP 地址

为了使信息能在以太网上准确快捷地传送到目的地，连接到以太网的每台计算机必须拥

有一个唯一的地址。为每台计算机指定的地址称为 IP 地址。

IP 地址由 32 位二进制数 (4B) 组成, 是 Internet Protocol (网际协议) 地址, 每个 Internet 包必须有 IP 地址, 每个 Internet 服务提供商 (ISP) 必须向有关组织申请一组 IP 地址, 一般是动态分配给其用户, 用户也可以根据接入方式向 ISP 申请一个 IP 地址。

IP 地址通常用十进制数表示, 用小数点分隔, 例如 192.168.0.117。192.168.x.y 是标准的没有路由的专用网络的 IP 地址。CPU 默认的 IP 地址为 192.168.0.1。

同一个 IP 地址可以使用具有不同 MAC 地址的网卡, 更换网卡后可以使用原来的 IP 地址。

### 3. 子网掩码

子网是连接在网络上的设备的逻辑组合。子网掩码 (Subnet Mask) 是一个 32 位地址, 用于将网络划分为一些小的子网。

IP 地址由子网地址 (或称为网络 ID) 和子网内的节点地址组成, 子网掩码用于将这两个地址分开。由子网掩码确定的两个 IP 地址段分别用于寻址子网 IP 和节点 IP。二进制的子网掩码的高位应是连续的 1, 低位应是连续的 0。小型局域网的子网掩码一般为 255.255.255.0, 其高 24 位二进制数为 1, 表示 IP 地址中的网络标识 (类似于长途电话的地区号) 为 24 位; 低 8 位二进制数为 0, 表示子网内节点的标识 (类似于长途电话的电话号) 为 8 位。IP 地址和子网掩码进行“与”逻辑运算, 得到子网地址。IP 地址和子网掩码取反后得到的 0.0.0.255 进行“与”逻辑运算, 得到节点地址。

### 4. 路由器

路由器用于连接子网, 如果 IP 报文发送给别的子网, 首先将它发送给路由器。在组态时子网内所有的节点都应输入路由器的地址。路由器通过 IP 地址发送和接收数据包。路由器的子网地址与子网内的节点的子网地址相同, 其区别仅在于子网内的节点地址不同。

### 5. 传输速率

在串行通信中, 传输速率 (又称波特率) 的单位为 bit/s, 即每秒传送的二进制位数。西门子的工业以太网默认的传输速率为 10Mbit/s/100Mbit/s。

## 7.3.2 S7-1200 与编程计算机的通信

3.4 节介绍了建立编程计算机与 CPU 通信连接的方法。其前提是 CPU 中已经有下载的 IP 地址。新出厂的 CPU 没有 IP 地址, 只有厂家设置的 MAC 地址。CPU 的 MAC 地址在 CPU 的正面左下角以太网接口的上面, 必须打开以太网接口上的盖板才能看到 MAC 地址。PROFINET 通信要求给网络内所有的设备分配唯一的 IP 地址。本节主要介绍新的 CPU 没有 IP 地址时怎样实现计算机与 CPU 的通信。

### 1. 新出厂的 CPU 与计算机的通信

首先用直通的或交叉的以太网电缆连接计算机和 CPU 的 RJ 45 以太网接口。

打开 STEP 7 Basic, 生成一个项目, 在项目中生成一个 PLC 设备, 其 CPU 的型号和订货号应与实际的硬件相同。设置 CPU 的 IP 地址和子网掩码时, 可以采用默认的 IP 地址 192.168.0.1 和默认的子网掩码 255.255.255.0。

用 3.4 节介绍的方法, 设置计算机的网卡的 IP 地址和子网掩码 (见图 3-20), 计算机和 CPU 的子网掩码相同, 它们的 IP 地址的前 3 个字节 (子网地址) 也应相同, 第 4 个字节只要不重叠就可以了。

做好上述的准备工作后，接通 PLC 的电源。选中项目树中的 PLC\_1 后，点击工具栏上的下载按钮 ，打开“Extended download to device”对话框（见图 7-4）。

如果计算机有不止一个以太网卡（例如笔记本电脑一般有一块有线网卡和一块无线网卡），用“PG/PC interface for loading”下拉式列表选择实际使用的网卡。

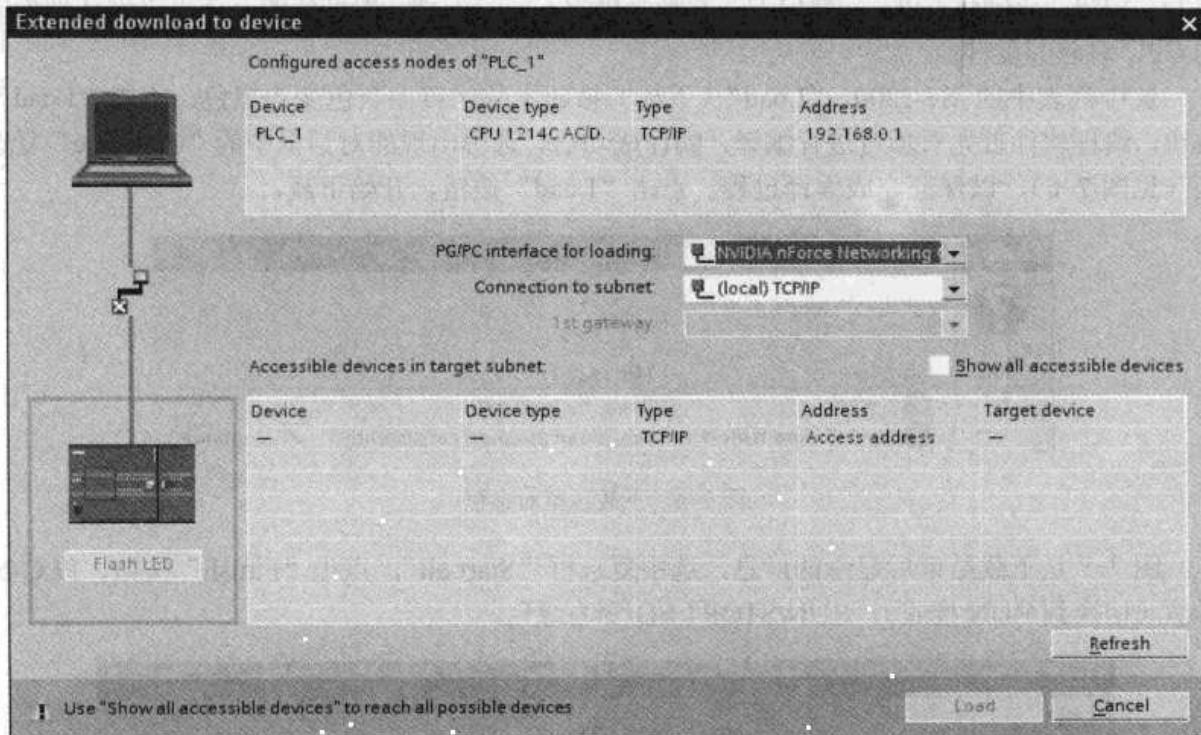


图 7-4 扩展的下载对话框

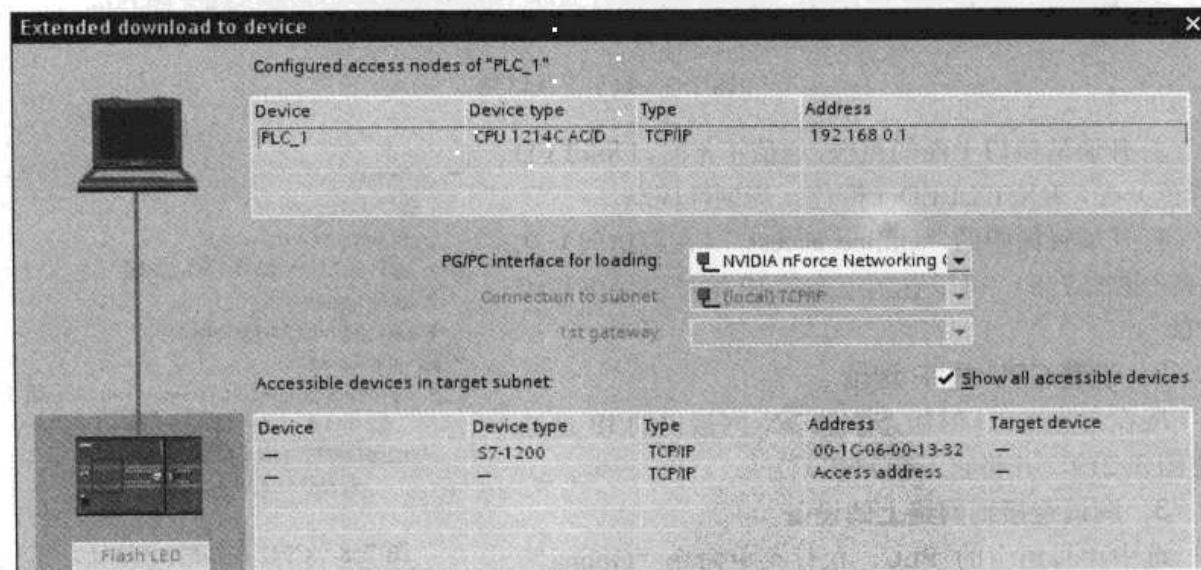


图 7-5 扩展的下载对话框

选中复选框“Show all accessible devices”（显示所有可访问节点），经过一定的时间后，

在下面的“Accessible devices in target subnet”（目标子网中的可访问设备）列表中，出现网络上的 S7-1200 CPU 和它的 MAC 地址（见图 7-5），计算机与 PLC 之间的连线由断开变为接通。CPU 所在方框的背景色变为实心的橙色，表示 CPU 进入在线状态。

如果网络上有多个 CPU，选中设备列表中的某个 CPU，点击左边的 CPU 图标下面的“Flash LED”按钮，该硬件 CPU 上的 LED（发光二极管）将会闪动。用这样的方法可以确认列表中的 CPU 对应的硬件。

选中列表中的 S7-1200，“Load”（下载）按钮上的字符由灰色变为黑色。点击“Load”按钮，编程软件首先对项目进行编译，编译成功后，选中出现的对话框中的“Continue”复选框（见图 7-6），以保证完成编译过程。点击“Load”按钮，开始下载。

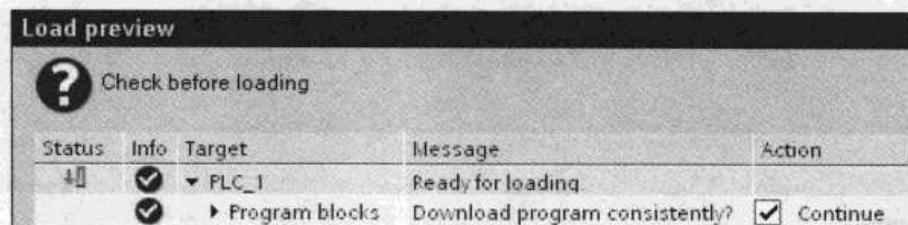


图 7-6 下载预览对话框

图 7-7 是下载结束后显示的信息，选中复选框“Start all”，点击“Finish”按钮，PLC 被启动（切换到 RUN 模式），RUN/STOP LED 变为绿色。

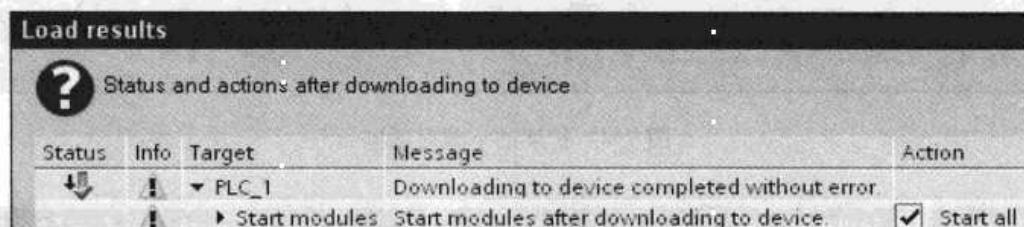


图 7-7 显示下载结果

打开通信接口上面的盖板，通信正常时，Link LED（绿色）亮，RX/TX LED（橙色）周期性闪动。

打开项目树中的“Online access”（在线访问）文件夹（见图 7-8），可以看到组态的 IP 地址已经下载给 CPU。

## 2. 设置 CPU 的 IP 地址

应在组态 CPU 时组态它的以太网接口的 IP 地址（见图 3-21），并将它下载到 PLC。

## 3. 测试连接到网络上的设备

选中项目树中的 PLC，执行菜单命令“Online”

→“Extended download to device”，打开扩展的下载对话框（见图 7-5），计算机与网络建立起连接后，选中该对话框中的复选框“Show all accessible devices”，将显示所有可以访问的网络设备，以及它们的 IP 地址和 MAC 地址。

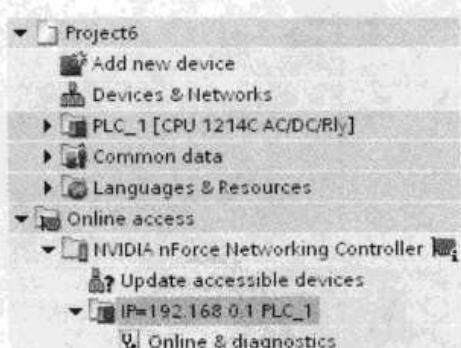


图 7-8 CPU 在线的 IP 地址

如果在可访问设备列表中没有看到某台设备，说明因为某种原因与它的通信中断。应找出设备或网络的硬件或组态的错误。

#### 4. 组态网络时间同步

网络时间协议（Network Time Protocol, NTP）广泛应用于互联网的计算机时钟的时间同步，局域网内的时间同步精度可达 1ms。NTP 采用多重冗余服务器和不同的网络路径来保证时间同步的高精度和高可靠性。

离线组态时，选中 CPU 的以太网接口，打开 PLC\_1 的设备组态视图，首先选中 CPU 的以太网接口，然后选中下面的监视窗口的“Properties”（属性）选项卡左边窗口的“Time synchronization”（时间同步）组，激活实时时钟同步复选框（见图 7-9）。然后设置时钟同步的服务器的 IP 地址和更新的时间间隔（Update interval）。设置的参数下载后起作用。

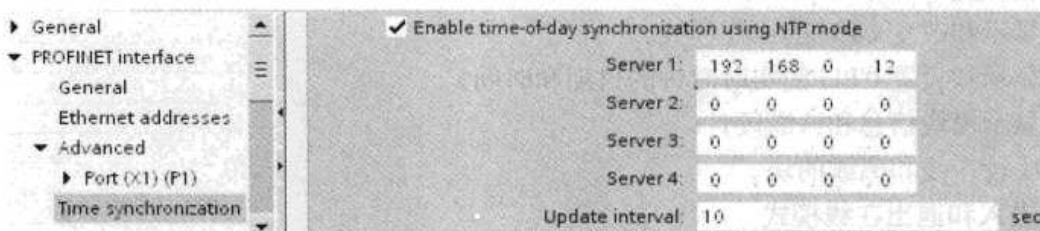


图 7-9 组态网络时间同步

#### 5. 上载程序块

为了上载 PLC 中的程序，首先生成一个新的项目。在项目中生成一个 PLC 设备，其型号和订货号与实际的硬件相同。

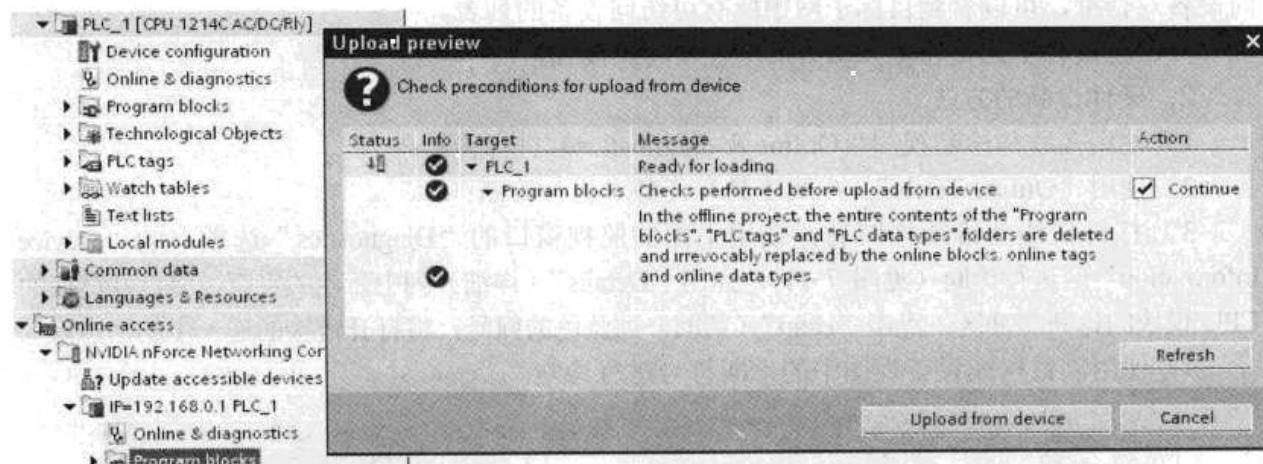


图 7-10 上载程序块

用以太网电缆连接好编程计算机和 CPU 的以太网接口后，打开文件夹“PLC\_1”和“Online access”，将图 7-10 最下面在线的“Program block”（程序块）文件夹拖放到上面离线的“Program block”文件夹中。

松开按住的鼠标左键，出现“Upload preview”（上载预览）对话框，选中“Continue”（继续）复选框，点击“Upload from device”按钮，离线的程序块和 PLC tags（变量表）被在线

的块和在线的 PLC 变量表取代。

与 S7-200 和 S7-300/400 不同，S7-1200 下载了变量表和程序中的注释。因此在上载时可以得到 CPU 中的变量表和程序中的注释，它们对于程序的阅读是非常有用的。

## 7.4 故障诊断

### 7.4.1 在线功能与故障诊断

建立起编程计算机与 PLC 的在线连接后，可以进行下列操作：

- 下载程序和项目组态数据给 CPU。
- 测试用户程序。
- 显示和改变 CPU 的操作模式。
- 显示和设置 CPU 的实时时钟的日期和时间。
- 显示模块信息和诊断硬件。
- 比较在线和离线的块。

#### 1. 进入和退出在线模式

建立在线连接之前，应使用以太网电缆连接编程计算机与 CPU 的硬件连接。打开项目树中的某个 PLC 的文件夹，点击工具栏上的按钮  Go online，进入在线模式。被激活的项目树或工作区的标题栏的背景色为表示在线的橙色，没有激活的窗口的标题栏下沿有橙色的线条。

执行菜单命令“Online”→“Accessible devices”，或点击工具栏上的  按钮，打开可访问设备对话框，可以看到目标子网中所有可访问设备的列表。

点击工具栏上的按钮  Go offline，进入离线模式。

#### 2. 硬件诊断的方法

- 1) 使用在线与诊断视图（Online & diagnostics）进行诊断；
- 2) 使用“Online Tools”（在线工具）任务卡进行诊断；
- 3) 打开在线与诊断视图后，使用下面的监视窗口的“Diagnostics”选项卡的“Device Information”区进行诊断（见图 7-11）。点击“Details”（详细）列中蓝色的字符，将打开链接的 CPU 模块的诊断缓冲区。点击“Help”（帮助）列蓝色的问号，将打开链接的进一步的信息。
- 4) 使用项目树或设备视图中的诊断符号进行诊断。

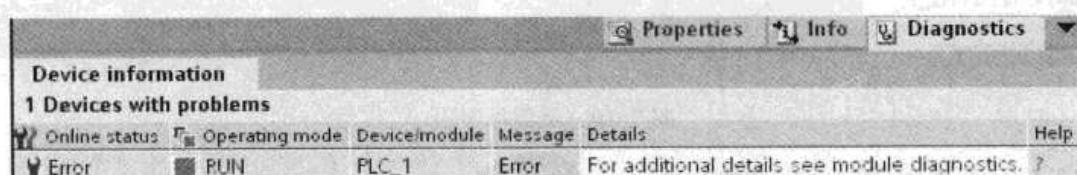


图 7-11 用监视窗口诊断硬件

#### 3. 使用符号显示诊断信息

与 CPU 建立起在线连接，进入在线模式后，用符号显示有关模块的状态和运行模式。

- 1) 设备视图：除了 CPU 上的信号板之外，设备视图中所有的硬件组件将显示诊断符号。

- 2) 网络视图：每个组件都有诊断符号，显示所有硬件组件共同的状态。
- 3) 项目树：硬件组件的右边有诊断符号，有操作模式的模块的右边还显示操作状态符号。
- 4) 项目树的详细视图：选中项目树中的“Local module”（本地模块），在下面的“Details view”（详细视图）中，可以看到 CPU 和其他模块的状态符号。

#### 4. 诊断符号的意义

##### (1) 模块与设备的诊断符号

- ?：目前正在建立与 CPU 的连接。
- ：CPU 不能到达设置的地址。
- ：组态的 CPU 与实际的 CPU 型号不同。
- ✓：没有故障。
- ：需要维护（绿色）。
- ：请求维护（黄色）。
- ：错误（红色）。
- ：模块或设备不能被 CPU 访问。
- ！：因为当前在线的组态数据不同于离线的组态数据，没有可用的诊断数据。
- ：组态的模块或设备与实际的模块或设备不一致。
- ？：连接已经建立，但是不能确定模块的状态（红色）。
- ？：组态的模块不支持诊断状态的显示（灰色）。

##### (2) 比较状态的符号

诊断符号可以与显示在线/离线比较结果的小符号结合。下面是比較状态的符号：

- ：文件夹包含有在线/离线不一致的对象（红色的圆，仅用于项目树）。
- ：对象的在线/离线版本不同（蓝色与红色的半圆）。
- ：仅有在线对象（蓝色的半圆弧和红色的半圆）。
- ：仅有离线对象（蓝色的半圆与红色的半圆弧）。

##### (3) 诊断与比较结合的符号

- ?：没有故障，文件夹包含有在线/离线不一致的对象（仅用于项目树）。
- ：模块或设备不能被 CPU 访问，仅有离线对象。

##### (4) CPU 与通信模块的运行模式符号

- ：RUN（绿色）。
- ：STOP（橙色）。
- ：STARTUP（启动，橙色与绿色）。
- ：HOLD（保持）。
- ：DEFECTIVE（有故障，红色）。
- ：运行状态未知（灰色）。
- ？：组态的模块不支持显示操作模式（灰色）。

#### 5. 在线与诊断视图

进入在线模式后，双击 PLC\_1 文件夹中的“Online & diagnostics”（见图 7-12），在工作区打开在线与诊断视图。

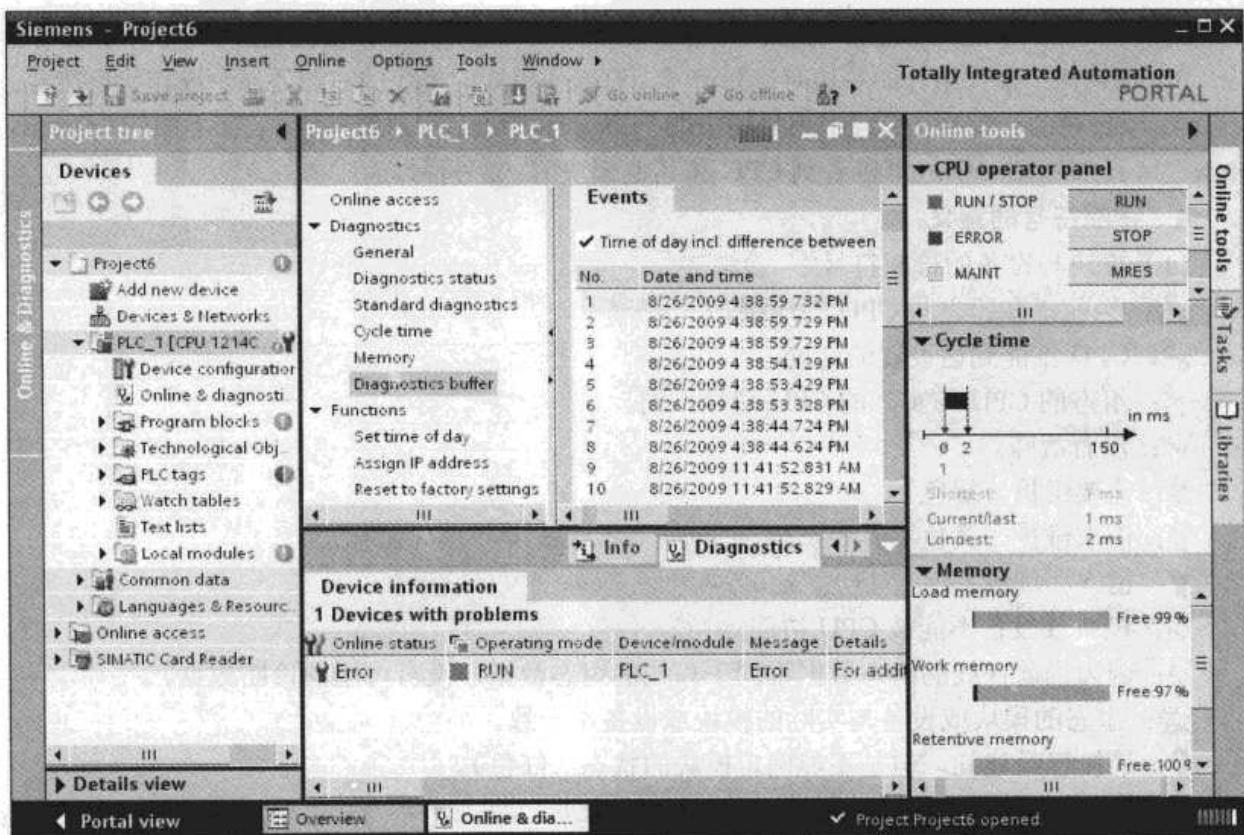


图 7-12 在线与诊断视图

在线与诊断视图左边窗口的属性结构由文件夹和文件夹中的组 (Group) 组成。选中左边窗口中的某个组，右边窗口是有关的详细信息。

选中最上面的“Online access”组，将显示是否建立了在线连接，以及通信接口（网卡）的信息。可以用右边窗口中的“Go offline”按钮断开在线连接。

点击最右边垂直条上的“Online tools”按钮，在右边的任务卡中打开在线工具（Online tools）窗口。

最上面是 CPU 的操作面板（CPU operator panel），可以看到 CPU 上 3 个 LED 的状态。用面板中的“RUN”和“STOP”按钮可以切换 CPU 的操作模式。选中项目树中的某台 PLC 后，点击工具栏上的按钮 或 ，也以使该 PLC 切换到 RUN 或 STOP 模式。

点击 CPU 操作面板上的“MRES”按钮，将会清除工作区所有的保持性和非保持性的存储器，断开现有的通信连接。用户程序、IP 地址、系统时间、诊断缓冲区和存储卡的内容不受影响。

右边中间的“Cycle time”窗口显示了 CPU 最短的、最长的和当前的扫描循环周期。下面的“Memory”窗口显示了使用的装载存储器、工作存储器和保持存储器所占的百分比。选中工作区中的“Cycle time”和“Memory”组，可以获得更多的信息。

## 6. 显示和设置 CPU 的实时时钟的日期和时间

选中工作区左边窗口中的“Set time of day”（见图 7-13），可以在右边窗口设置 PLC 的实时时钟。

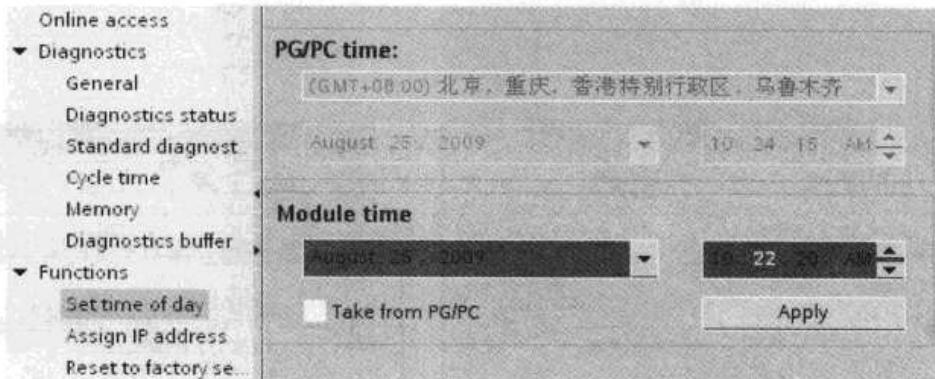


图 7-13 设置实时时钟

选中复选框“Take from PG/PC”，点击“Apply”按钮，PLC 的实时时钟将会与计算机的实时时钟同步。未选中该复选框时，可以在“Module time”区设置 CPU 的日期和时间，例如，点击图中时间的第 2 个数字（分钟，图中为白色的 22），可以用键盘或时间域右边的增、减按钮来设置选中的分钟值。设置好后点击“Apply”按钮确认。

## 7. 用诊断缓冲区显示 CPU 中的诊断事件

诊断缓冲区用于记录诊断事件（例如 CPU 错误和模块错误），和 CPU 的 RUN/STOP 操作模式的变化。选中图 7-13 工作区左边窗口的“Diagnostics buffer”，右边窗口是诊断缓冲区，详细的使用方法见图 6-35 和有关的说明。

## 8. 显示模块信息

双击项目树中要诊断的 PLC 的文件夹中的“Online & diagnostics”，选中工作区左边窗口的“Diagnostics”文件夹中的“Diagnostic status”组（见图 7-13），右边窗口将会显示模块的诊断信息，例如“Module available and OK”（模块存在并且正常）或“Module defective”（模块有故障）。

## 9. 比较在线和离线的块

在离线状态时用鼠标右键点击程序块文件夹或其中的某个块，执行快捷菜单命令“Compare offline/online”，比较离线（编程设备中）和在线（PLC 中）的块中的数据。工作区给出比较的结果（见图 7-14），主程序 Main 的状态（Status）图标为蓝色和橙色的两个半圆，表示在线与离线的 Main 不一致。在项目树中也可以看到在线/离线不一致的块。

Object name	Status	Action	Description
PLC_1	●	●	No action
Program blocks	●		
Main	●		

图 7-14 比较在线和离线的块

双击工作区中的“Main”，工作区出现离线（Offline）和在线（Online）两个窗口（见图 7-15）。工作区下面的监视窗口的“Info”（信息）选项卡中列出了在线和离线有差异的网络（NW）号和信息（Message）。双击某一条信息，上面的离线窗口与在线窗口将会打开对应的不一致的网络，并且用专门的背景色来表示不一致的编程元件或地址。

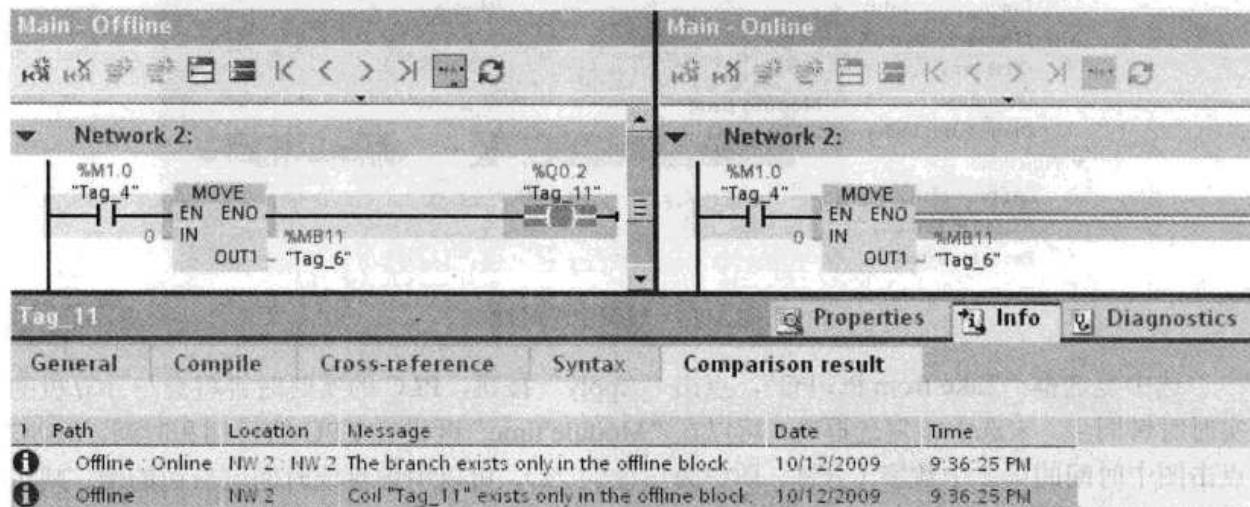


图 7-15 比较在线和离线的块

## 7.4.2 使用状态 LED 诊断故障

### 1. CPU 的 LED

CPU 和 I/O 模块用 LED (发光二极管) 提供运行状态或 I/O 的信息。表 7-2 给出了 CPU LED 的组合意义。

表 7-2 CPU LED 的组合意义

描述	STOP/RUN (橙/绿)	ERR (红)	MAINT (橙)
断电	熄灭	熄灭	熄灭
启动、自检测、固件更新	橙色/绿色交替闪动	—	熄灭
STOP 模式	橙色常亮	—	—
RUN 模式	绿色常亮	—	—
拔出存储卡	橙色常亮	—	闪烁
出错	橙色或绿色常亮	闪烁	—
维护请求	橙色或绿色常亮	—	常亮
硬件故障	橙色常亮	常亮	熄灭
LED 检测或有问题的 CPU 固件	橙色/绿色交替闪动	闪烁	闪烁

#### (1) STOP/RUN LED

常亮的橙色和绿色分别表示 CPU 的 STOP 和 RUN 运行模式。

橙色和绿色交替闪烁表示 CPU 处于启动状态。

#### (2) ERROR LED (红色)

闪烁表示出错，例如 CPU 的内部错误、存储卡错误，或者组态错误（模块不匹配）。

常亮表示硬件有问题。

### (3) MAINT (Maintenance)

插入存储卡时, MAINT (维护) LED 闪烁, CPU 切换到 STOP 模式。执行完成下列功能之一, 初始化存储卡:

- 1) 将 CPU 切换到 RUN 模式。
- 2) 执行存储器复位操作。
- 3) 将 CPU 断电后重新上电。

打开 CPU 顶部的端子板盖板, 可以看到 CPU 用来提供 PROFINET 通信状态的两个 LED: “Link” (绿色) 亮表示连接成功, “Rx/Tx” (黄色) 亮表示数据传输被激活。

#### 2. 信号模块的 LED

CPU 和每块数字量信号模块 (SM) 提供每点数字量输入 (DI)、数字量输出 (DO) 的 I/O 状态 LED。它们点亮和熄灭分别表示对应的输入、输出点为 1 状态和 0 状态。

模拟量信号模块为每个模拟量输入、输出通道提供一个 I/O 通道 LED, 绿色表示通道被组态和激活, 红色表示通道处于错误状态。

此外, 每块数字量信号模块和模拟量信号模块还有一个 DIAG (诊断) LED, 用于显示模块的状态, 绿色表示模块运行正常, 红色表示模块有故障或不可用。信号模块还要检测现场侧的电源是否存在。表 7-3 给出了信号模块 LED 的组合意义。

表 7-3 信号模块 LED 的组合意义

描    述	DIAG (红/绿)	I/O 通道 (红/绿)
现场侧电源消失	红色闪动	红色闪动
没有组态或没有进行更新	绿色闪动	熄灭
模块被正确组态	绿色常亮	绿色常亮
错误的状态	红色闪动	—
I/O 错误 (诊断被激活)	—	红色闪动
I/O 错误 (诊断被禁止)	—	绿色常亮

## 7.5 PLC 之间的开放式用户通信

### 7.5.1 开放式用户通信的编程

#### 1. 开放式用户通信

开放式用户通信 (Open User Communication) 是一种程序控制的通信方式, 可以使用多种通信类型, 其主要特点是它传输的数据结构的灵活性。

这种通信只受用户程序的控制, 可以建立和断开事件驱动的通信连接。在运行期间也可以修改连接。

S7-1200 的开放式用户通信可以使用 TCP 和 ISO-on-TCP 连接。通信伙伴可以是两台 SIMATIC PLC, 或 SIMATIC PLC 和适当的第三方设备。

紧凑型指令 TSEND\_C 和 TRCV\_C 除了分别具有发送或接收功能外, 还可以建立和断开连接。指令 TSEND 和 TRCV 仅有发送或接收功能, 与它们配套的 TCON 和 TDISCON 指令

用于建立和断开连接。

在开放式用户通信中，一台 PLC 调用 TSEND\_C 或 TSEND 发送数据，另一台 PLC 调用 TRCV\_C 或 TRCV 接收数据。

## 2. 组态 CPU 之间的通信连接

生成一个名为 PLC\_PLC 的项目（见随书光盘中的同名例程），点击项目树中的“Add new device”，添加两块 CPU 1214C。

双击项目树中的“Device & Networks”，打开网络视图（图 7-16）。选中左边的 CPU 左下角表示以太网接口的绿色小方框。按住鼠标左键不放，将它“拖”到右边的 CPU 左下角表示以太网接口的绿色小方框上，将会出现图 7-16 所示的绿色的以太网线，和名称为“PN/IE\_1”的连接。

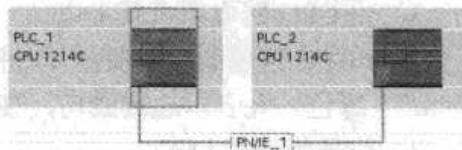


图 7-16 网络组态

## 3. 调用 TSEND\_C 和 TRCV\_C

双击项目树的“Program block”（程序块）文件夹中的主程序 OB1，打开程序编辑器（见图 7-17）。将右边的“Extended instructions”（扩展指令）窗口的“Communication”（通信）文件夹中的 TSEND\_C 拖放到工作区。点击自动出现的“Call options”（调用选项）对话框中的“OK”按钮，除了自动生成被调用的功能块的背景数据块 TSEND\_C\_DB 外，还会自动生成保存连接的组态参数的连接描述数据块 PLC\_1\_Connection\_DB。在项目树的程序块文件夹中，可以看到这些自动生成的数据块。

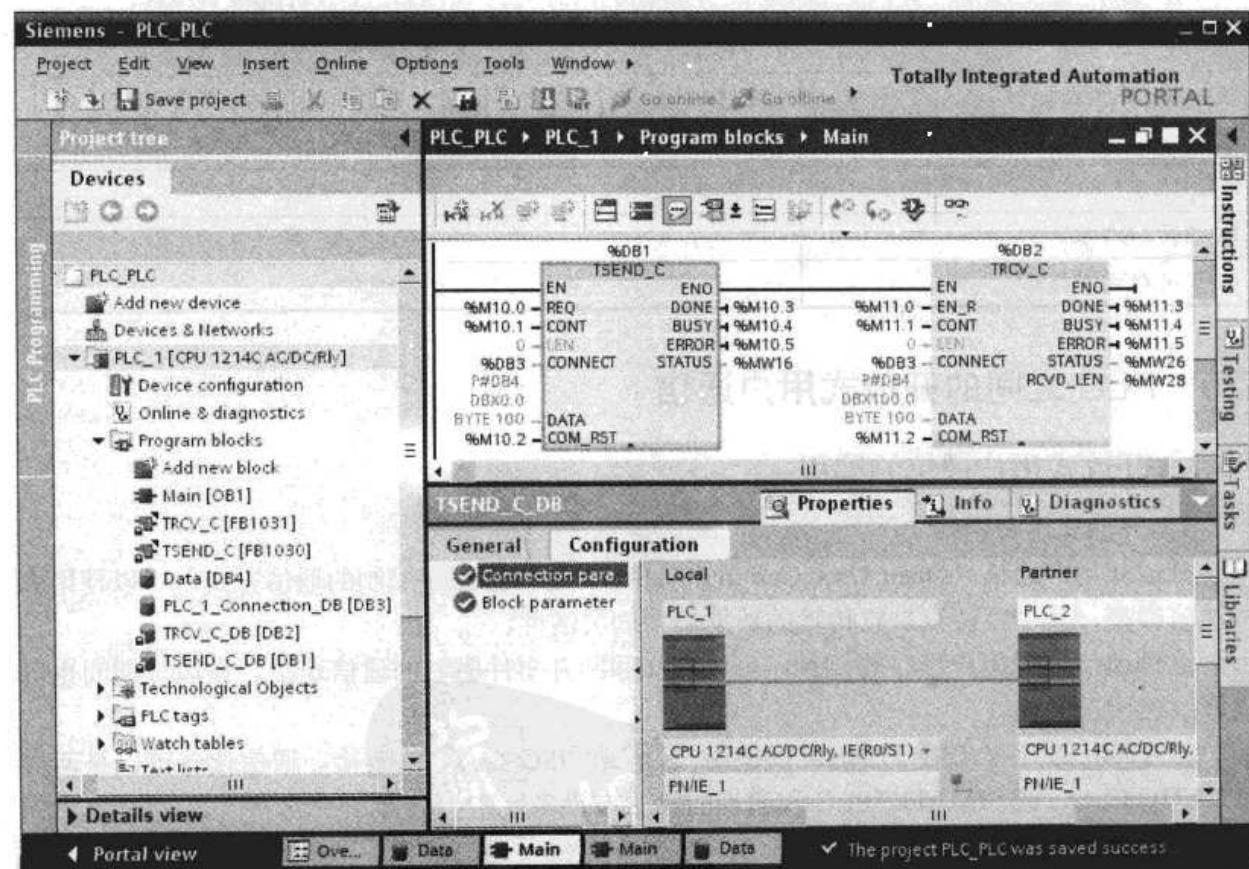


图 7-17 项目视图

用同样的方法调用 TRCV\_C，自动生成背景数据块 TRCV\_C\_DB。

用同样的方法打开 PLC\_2 的 OB1，调用 TSEND\_C 和 TRCV\_C。

#### 4. TSEND\_C 和 TRCV\_C 的参数的意义

TSEND\_C 的参数（见图 7-17）的意义如下：

在请求信号 REQ 的上升沿，根据参数 CONNECT 指定的 DB3 中的连接描述，启动数据发送任务。发送成功后，参数 DONE 在一个扫描周期内为 1。

CONT (Bool) 为 1 时建立和保持连接，为 0 时断开连接，接收缓冲区中的数据将会消失。连接被成功建立时，参数 DONE 在一个扫描周期内为 1。CPU 进入 STOP 模式时，已有的连接被断开。

LEN 是要发送的数据的最大字节数。LEN 为默认值 0 时，发送用 DATA 定义的所有数据，此时不用设置 LEN 的数值。点击指令方框下边沿中点的小三角形符号，可以显示或隐藏默认的参数 LEN。

CONNECT 是连接描述数据的地址指针。

DATA (Variant) 是发送区的地址和数据长度，可以使用除 Bool 和 Bool 数组之外的所有数据类型。

生成名为 Data 的共享数据块 (DB4) 后（见图 7-18），在 DB4 中生成保存发送数据的数据组 To\_PLC2，和保存接收到的数据的数组 From\_PLC2。

图 7-17 中 TSEND\_C 的参数 DATA 的实参 P#DB4.DBX0.0 BYTE 100 是数据块 Data 中的数组 To\_PLC2 的绝对地址。TRCV\_C 的参数 DATA 的实参 P#DB4.DBX100.0 BYTE 100 是数据块 Data 中的数组 From\_PLC2 的绝对地址。

Name	Data type	Offset	Initial value	Retain
1 ▾ Static				
2 ► To_PLC2	Array [1 .. 100] of byte	0.0		<input type="checkbox"/>
3 ► From_PLC2	Array [1 .. 100] of byte	100.0		<input type="checkbox"/>

图 7-18 在数据块中生成发送和接收数据的数组

COM\_RST (Bool) 为 1 时，断开现有的通信连接，新的连接被建立。如果此时数据正在传送，可能导致丢失数据。

DONE (Bool) 为 1 时表示任务执行成功，为 0 时任务未启动或正在运行。

BUSY (Bool) 为 0 时任务完成，为 1 时任务尚未完成，不能触发新的任务。

ERROR (ool) 为 1 时执行任务出错，字变量 STATUS 中是错误的详细信息。

指令 TRCV\_C 的参数的意义如下：

EN\_R (Bool) 为 1 时，准备好接收数据。

参数 CONT 和 EN\_R 均为 1 时，连续地接收数据。

DATA (Variant) 是接收区的起始地址和最大数据长度。

LEN 是接收区的字节长度，RCVD\_LEN 是实际接收的数据的字节数。

其余的参数与 TSEND\_C 的相同。

## 7.5.2 开放式用户通信的组态

### 1. 开放式用户通信的组态

打开 PLC\_1 的 OB1，首先选中指令 TSEND\_C，然后选中下面的监视窗口左边的“Connection parameter”（连接参数）组（见图 7-19）。

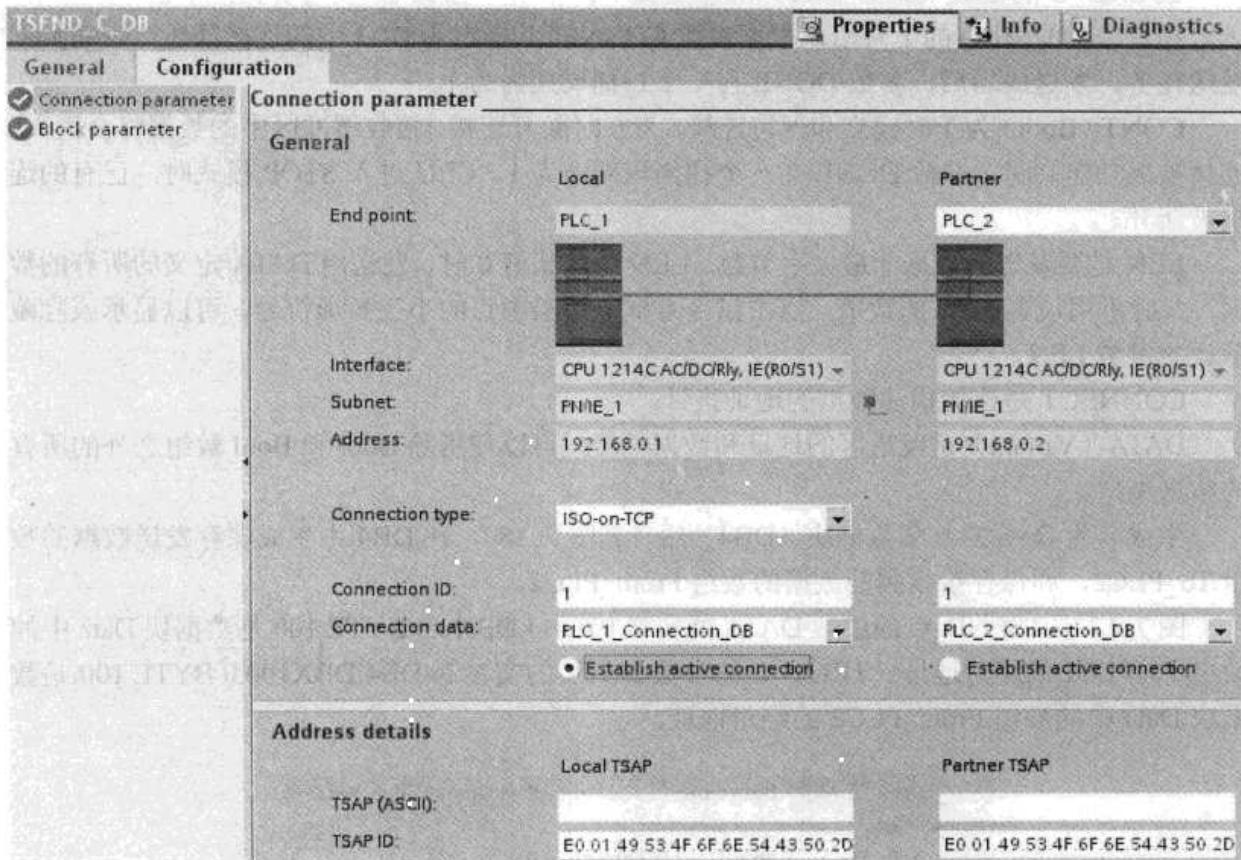


图 7-19 设置 TSEND\_C 的参数

在右边的窗口中，“Local”是选中的名为 PLC\_1 的 CPU。点击“Partner”（伙伴）的“End point”（结束点）选择框右边的▼按钮，用出现的下拉式列表选择通信伙伴为 PLC\_2，两台 PLC 图标之间出现表示“PROFINET”的绿色连线。下拉式列表中的连接伙伴“unspecified”（未定义）是项目中其数据未知的设备。“Connection ID”（连接标识符）的默认值为 1。

可以用“Connection type”选择框选择连接类型为 TCP 或 ISO-on-TCP。如果选择 TCP，“Address details”（地址的详细信息）域伙伴的接口（Port）的默认值为 2000。

点击伙伴的“Connection date”（连接数据）选择框右边的▼按钮，用出现的下拉式列表设置通信伙伴的连接描述数据块。

通信的一方作为主动的伙伴，启动通信连接的建立。另一方作为被动的伙伴，对启动的连接作出响应。图 7-19 用单选框设置 PLC\_1 主动建立连接（Establish active connection）。

需要将程序和组态数据下载到通信伙伴双方的 CPU，才能实现通信。

设置连接参数时，并不检查连接 ID 和 TCP 连接的接口（port）编号和 ISO-on-TCP 连接的 TSAP 是否重叠。应保证这些参数在网络中是唯一的。

## 2. 删除连接

开放式用户连接的组态参数保存在连接描述数据块内。可以通过删除该数据块来删除连接。打开某个通信伙伴的“Program blocks”文件夹，用鼠标右键点击连接描述数据块，执行出现的快捷菜单中的“Delete”命令来删除它。

在删除该数据块的同时，应删除调用时使用它作为输入参数的通信功能块 TSEND\_C、TRCV\_C 及其背景数据块，这样才能保证程序的一致性。

## 7.6 串行通信

### 7.6.1 串行通信的基本概念

#### 1. 并行通信与串行通信

并行数据通信是以字节或字为单位的数据传输方式，除了 8 根或 16 根数据线、一根公共线外，还需要通信双方联络用的控制线。并行通信的传输速度快，但是传输线的根数多，一般用于近距离数据传输。

串行数据通信是以二进制的位（bit）为单位的数据传输方式，每次只传送一位。串行通信需要的信号线少，最少的只需要两根线（双绞线），传输线既作为数据线又作为通信联络控制线，数据按位进行传送。串行通信适用于距离较远的场合，工业控制一般使用串行通信。计算机和 PLC 都有通用的串行通信接口，例如 RS-232C 和 RS-485 接口。

#### 2. 异步通信与同步通信

在串行通信中，接收方和发送方的传输速率应相同，但是实际的发送速率与接收速率之间总是有一些微小的差别，如果不采取措施，在连续传送大量的信息时，将会因为积累误差造成错位，使接收方收到错误的信息。为了解决这一问题，需要使发送过程和接收过程同步。按同步方式的不同，可以将串行通信分为异步通信和同步通信。

图 7-20 是异步通信的字符信息格式，发送的字符由一个起始位、7~8 个数据位、一个奇偶校验位（可以没有）、一个或两个停止位组成。在通信开始之前，通信的双方需要对所采用的信息格式和数据的传输速率作相同的约定。接收方检测到停止位和起始位之间的下降沿后，将它作为接收的起始点，在每一位的中点接收信息。由于一个字符中包含的位数不多，即使发送方和接收方的收发频率略有不同，也不会因两台设备之间的时钟脉冲周期的积累误差而导致收发错位。异步通信传送附加的非有效信息较多，传输效率较低。

同步通信以字节为单位（一个字节由 8 位二进制数组成），每次传送 1~2 个同步字符、若干个数据字节和校验字符。同步字符起联络作用，用它来通知接收方开始接收数据。在同步通信中，发送方和接收方要保持完全的同步，这意味着发送方和接收方应使用同一个时钟脉冲。可以通过调制解调方式在数据流中提取出同步信号，使接收方得到与发送方完全相同的接收时钟信号。

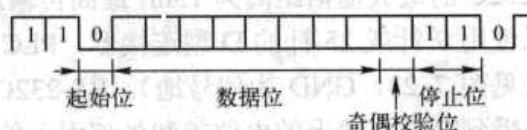


图 7-20 异步通信的字符信息格式

由于同步通信方式不需要在每个数据字符中附加起始位、停止位和奇偶校验位，只需要在数据块（往往很长）之前加一两个同步字符，所以传输效率高，但是对硬件的要求较高，一般用于高速通信。

### 3. 单工与双工通信方式

单工通信方式只能沿单一方向发送或接收数据。双工方式的数据可以沿两个方向传送，每一个站既可以发送数据，也可以接收数据。双工方式又分为全双工和半双工两种方式。

#### (1) 全双工方式

全双工方式数据的发送和接收分别使用两组不同的数据线，通信的双方都能在同一时刻接收和发送信息（见图 7-21）。

#### (2) 半双工方式

半双工方式用同一组线（例如双绞线）接收和发送数据，通信的某一方在同一时刻只能发送数据或接收数据（见图 7-22）。

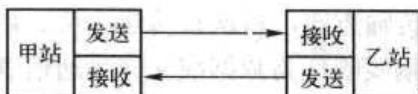


图 7-21 全双工方式

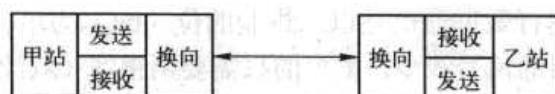


图 7-22 半双工方式

## 4. 串行通信的接口标准

### (1) RS-232C

RS-232C 是美国 EIA (电子工业联合会) 在 1969 年公布的通信协议，至今仍在计算机和 PLC 中使用。

RS-232C 采用负逻辑，用 -15~ -5V 表示逻辑状态 “1”，用 5~15V 表示逻辑状态 “0”。RS-232C 的最大通信距离为 15m，最高传输速率为 20 kbit/s，只能进行一对一的通信。RS-232C 可以使用 9 针或 25 针的 D 型连接器，PLC 一般使用 9 针的连接器，距离较近时只需要 3 根线（见图 7-23，GND 为信号地）。RS-232C 使用单端驱动、单端接收的电路（见图 7-24），容易受到公共地线上的电位差和外部引入的干扰信号的影响。

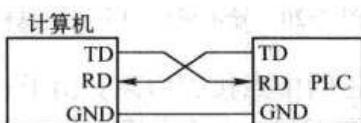


图 7-23 RS-232 的信号线连接



图 7-24 单端驱动单端接收

### (2) RS-422 与 RS-485

RS-422A 采用平衡驱动、差分接收电路（见图 7-25），利用两根导线间的电压差传输信号。这两根导线称为 A (TxD/RxD-) 和 B (TxD/RxD+)。当 B 的电压比 A 高时，认为传输的是逻辑“高”电平信号；当 B 的电压比 A 低时，认为传输的是逻辑“低”电平信号。能够有效工作的差动电压范围十分宽广，从零点几伏到接近十伏。

平衡驱动器有一个输入信号，两个输出信号互为反相信号，图中的小圆圈表示反相。外

部输入的干扰信号主要以共模方式出现，两根传输线上的共模干扰信号相同，因为接收器是差分输入，共模信号可以互相抵消。只要接收器有足够的抗共模干扰能力，就能从干扰信号中识别出驱动器输出的有用信号，从而克服外部干扰的影响。

RS-422 在最大传输速率(10 Mbit/s)时，允许的最大通信距离为 12m。传输速率为 100 kbit/s 时，最大通信距离为 1200m。一台驱动器可以连接 10 台接收器。

RS-422 有 4 根数据线：两根发送线 SDA、SDB 和两根接收线 RDA、RDB，采用全双工模式，两对平衡差分信号线分别用于发送和接收。

将 RS-422 的 SDA、RDA 连接在一起，SDB、RDB 连接在一起，就成了 RS-485 接口（见图 7-26），所以有的 PLC 的串行通信接口称为 RS-422/RS-485 接口。



图 7-25 RS-422 通信接线图

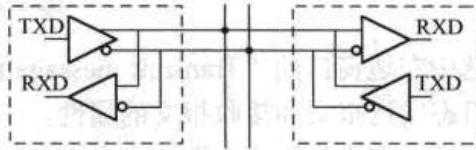


图 7-26 RS-485 网络

可以用 RS-485 通信接口和双绞线组成串行通信网络，构成分布式系统。RS-485 为半双工，只有一对平衡差分信号线，不能同时发送和接收。

## 7.6.2 串行通信模块与串行通信指令

### 1. 串口通信模块

CPU 支持基于字符的点对点（PtP）通信，用户程序可以定义和实现选择的协议。点对点通信具有很大的自由度和灵活性。

PtP 通信可以将信息直接发送给外部设备（例如打印机），接收外部设备（例如条形码阅读器）的信息。

PtP 通信需要使用 RS-232 或 RS-485 通信模块（CM）。STEP 7 Basic 的程序库提供用于 USS 驱动协议、Modbus RTU 主站协议和 Modbus RTU 从站协议的指令。

CM 1241 是 RS-485 模块，CM 1221 是 RS-232 模块。CPU 模块的左边最多可以安装 3 块通信模块。RS-232 或 RS-485 通信模块有下列属性：

- 绝缘的接口。
- 支持点对点协议。
- 通过扩展的指令和库功能编程。
- 用 LED 显示发送和接收被激活，有诊断 LED。
- 电源由 CPU 提供，不需要外接的电源。

### 2. 通信模块的组态

打开设备视图，将右边的硬件目录中的通信模块拖放到 CPU 左边的 101 号槽。选中该模块后，选中下面的监视窗口的属性选项卡左边窗口中的“Port configuration”（接口组态，见图 7-27），可以在右边的窗口设置通信接口的参数，例如传输速率（Baud rate）、奇偶校验（Parity）、数据位（Data bit）的位数、停止位（Stop bits）的位数、流控制（Flow control，仅用于 RS-232）和等待时间（Wait time）等。

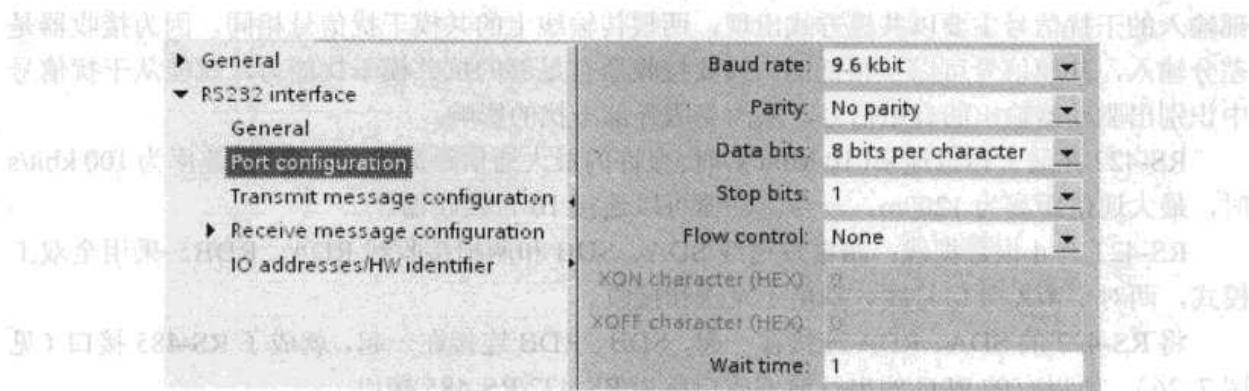


图 7-27 组态通信模块

选中左边窗口的“Transmit message configuration”和“Receive message configuration”，可以组态发送报文和接收报文的属性。

也可以在用户程序中调用指令 PORT\_CFG 来组态接口，但是 PORT\_CFG 指令设置的参数没有断电保持功能。

STEP 7 Basic 的 PtP 通信指令在右边的扩展指令窗口的“Communications”文件夹中。这些指令可以分为用于组态的指令和用于通信的指令。

### 3. 用于组态的串行通信指令

在进行 PtP 通信之前，应组态通信接口和发送数据、接收数据的参数。可以用 STEP 7 Basic 来组态，也可以在用户程序中用下列指令来组态：

PORT\_CFG 用于组态通信接口，SEND\_CFG 用于组态发送数据的属性，RCV\_CFG 用于组态接收数据的属性。

### 4. 用于通信的指令

SEND\_PTP 指令用于发送报文，RCV\_PTP 指令用于接收报文。所有的 PtP 指令的操作是异步的，用户程序可以使用轮询方式确定发送和接收的状态，这两条指令可以同时执行。通信模块发送和接收报文的缓冲区最大为 1024B。

### 5. 其他指令

RCV\_RST 用于清除接收缓冲区，SGN\_GET 用于读取 RS-232 通信信号的当前状态，SGN\_SET 用于设置 RS-232 通信信号的状态。

PtP 指令的详细使用方法见随书光盘中 S7-1200 的系统手册。

## 7.6.3 使用 Modbus 协议与 USS 协议的串行通信

### 1. Modbus 串行链路协议

Modbus 通信协议是 Modicon 公司提出的一种报文传输协议，Modbus 协议在工业控制中得到了广泛的应用，它已经成为一种通用的工业标准。不同厂商生产的控制设备通过 Modbus 协议可以连成通信网络，进行集中监控。许多工控产品，例如 PLC、变频器、人机界面、DCS 和自动化仪表等，都在使用 Modbus 协议。

根据传输网络类型的不同分为串行链路上的 Modbus 和基于 TCP/IP 的 Modbus。Modbus 串行通信可以使用 RS-485，短距离点对点通信时也可以使用 RS-232C 接口。

Modbus 串行链路协议是主-从协议，总线上只有一个主站，最多可以有 247 个子站。主站发出带有从站地址的请求报文，指定的从站接收到后发出响应报文进行应答。子站没有收到来自主站的请求时，不会发送数据，子站之间也不会互相通信。

## 2. Modbus 的报文传输模式

Modbus 协议有 ASCII 和 RTU（远程终端单元）这两种报文传输模式，S7-1200 采用 RTU 模式。报文以字节为单位进行传输，采用循环冗余校验（CRC）进行错误检查，报文最长为 256B。

在设置每个站的串口通信参数（波特率、校验方式等）时，Modbus 网络上所有的站都必须选择相同的传输模式和串口参数。表 7-4 给出了 S7-1200 支持的 Modbus 从站协议功能。

表 7-4 S7-1200 支持的 Modbus 从站协议功能

功能	描述	功能	描述
1	读取过程映像输出位 (Q0.0~Q1023.7)	5	写单个过程映像输出位 (Q0.0~Q1023.7)
2	读取过程映像输入位 (I0.0~I1023.7)	6	写单个保持寄存器字 (数据块中的字)
3	读取多个保持寄存器字 (数据块中的字)	15	写多个过程映像输出位 (Q0.0~Q1023.7)
4	读过程映像输入字 (IW0~IW1022)	16	写多个保持寄存器字 (数据块中的字)

## 3. 用于 MODBUS 通信的指令

指令 MB\_COMM\_LOAD 用于组态使用 Modbus RTU 协议通信的通信接口。

指令 MB\_MASTER 用于 PtP 模块作为 Modbus 主站的通信，使用该指令可以访问一个或多个 Modbus 从站的数据。调用该指令时需要指定背景数据块。

指令 MB\_SLAVE 用于 PtP 模块作为 Modbus 从站的通信，Modbus RTU 主站发出通信请求，用户程序通过指令 MB\_SLAVE 作出响应。调用该指令时需要指定背景数据块。使用该指令可以访问一个或多个 Modbus 从站的数据。

Modbus 协议指令的详细使用方法见随书光盘中 S7-1200 的系统手册。

## 4. 使用 USS 协议库与变频器通信

变频器具有调节范围宽、精度高、工作可靠、效率高、操作方便，便于与其它设备接口和通信等优点。随着技术的发展和价格的降低，变频器在工业控制中的应用越来越广泛。

如果 PLC 通过通信来监控变频器，使用的接线少，传送的信息量大，可以连续地对多台变频器进行监视和控制。还可以通过通信修改变频器的参数，实现多台变频器的联动控制和同步控制。

USS 通信协议用于 S7 PLC 和西门子变频器之间的通信，编程的工作量很小。通信网络由 PLC 和变频器内置的 RS-485 通信接口和双绞线组成，一台 S7-1200 CPU 最多可以监控 16 台变频器。这是一种硬件费用低、使用简便的通信方式。

在通信中，PLC 作为主站，变频器作为从站。主站才有权利发出通信请求报文，报文中的地址字符指定要传输数据的从站。从站只有在接收到主站的请求报文后才能向主站发送数据，从站之间不能直接进行信息交换。

指令 USS\_PORT 处理 CPU 通过 PtP 通信模块与驱动装置的通信。每次调用只处理与一个驱动装置最多一次通信。用户程序调用此功能应足够快，以防止驱动装置引起的通信超时。可以在主程序和中断程序中调用此功能。

指令 USS\_DRV 用于访问 USS 网络上指定的驱动装置。该功能块读取驱动装置的状态，其输出用于控制驱动装置。应为每个驱动装置调用一条 USS\_DRV 指令。

指令 USS\_RPM 和 USS\_WPM 用于读、写远程驱动装置的运行参数。程序可以多次调用这两个功能，但是在任意时刻，对每个驱动装置只能激活一个读写请求。

指令的详细使用方法见随书光盘中 S7-1200 的系统手册。

本章将从驱动装置的连接、驱动装置的参数设置、驱动装置的控制、驱动装置的故障检测

等方面对驱动装置进行详细介绍，帮助读者掌握驱动装置的基本应用。

驱动装置是直接与 CPU 连接的，因此驱动装置的连接方式与前面介绍的 I/O 模块的连接方式相同。

驱动装置的参数设置是指在驱动装置上设置驱动装置的运行参数，如驱动装置的启停、速度、加减速时间等。

驱动装置的控制是指通过编程语言（如梯形图、语句表）对驱动装置进行启停、速度、加减速时间等参数的设置。

驱动装置的故障检测是指通过驱动装置上的故障指示灯或 CPU 上的故障指示灯来检测驱动装置的故障状态。

通过以上对驱动装置的连接、参数设置、控制、故障检测等方面的介绍，读者应该能够掌握驱动装置的基本应用。

驱动装置的连接方式有串行连接和并行连接两种，下面将分别介绍这两种连接方式。

串行连接是指通过串行通信线（如 RS-232C、RS-485 等）将驱动装置与 CPU 连接起来。

并行连接是指通过并行总线（如 PCI、ISA 等）将驱动装置与 CPU 连接起来。

串行连接的优点是成本低、连接简单，但传输速率较低，适用于近距离连接。

并行连接的优点是传输速率高，适用于远距离连接，但成本较高。

驱动装置的参数设置是指在驱动装置上设置驱动装置的运行参数，如驱动装置的启停、速度、加减速时间等。

驱动装置的控制是指通过编程语言（如梯形图、语句表）对驱动装置进行启停、速度、加减速时间等参数的设置。

驱动装置的故障检测是指通过驱动装置上的故障指示灯或 CPU 上的故障指示灯来检测驱动装置的故障状态。

通过以上对驱动装置的连接、参数设置、控制、故障检测等方面的介绍，读者应该能够掌握驱动装置的基本应用。

驱动装置的连接方式有串行连接和并行连接两种，下面将分别介绍这两种连接方式。

串行连接是指通过串行通信线（如 RS-232C、RS-485 等）将驱动装置与 CPU 连接起来。

并行连接是指通过并行总线（如 PCI、ISA 等）将驱动装置与 CPU 连接起来。

串行连接的优点是成本低、连接简单，但传输速率较低，适用于近距离连接。

并行连接的优点是传输速率高，适用于远距离连接，但成本较高。

驱动装置的参数设置是指在驱动装置上设置驱动装置的运行参数，如驱动装置的启停、速度、加减速时间等。

驱动装置的控制是指通过编程语言（如梯形图、语句表）对驱动装置进行启停、速度、加减速时间等参数的设置。

驱动装置的故障检测是指通过驱动装置上的故障指示灯或 CPU 上的故障指示灯来检测驱动装置的故障状态。

通过以上对驱动装置的连接、参数设置、控制、故障检测等方面的介绍，读者应该能够掌握驱动装置的基本应用。

驱动装置的连接方式有串行连接和并行连接两种，下面将分别介绍这两种连接方式。

串行连接是指通过串行通信线（如 RS-232C、RS-485 等）将驱动装置与 CPU 连接起来。

并行连接是指通过并行总线（如 PCI、ISA 等）将驱动装置与 CPU 连接起来。

串行连接的优点是成本低、连接简单，但传输速率较低，适用于近距离连接。

并行连接的优点是传输速率高，适用于远距离连接，但成本较高。

驱动装置的参数设置是指在驱动装置上设置驱动装置的运行参数，如驱动装置的启停、速度、加减速时间等。

驱动装置的控制是指通过编程语言（如梯形图、语句表）对驱动装置进行启停、速度、加减速时间等参数的设置。

驱动装置的故障检测是指通过驱动装置上的故障指示灯或 CPU 上的故障指示灯来检测驱动装置的故障状态。

通过以上对驱动装置的连接、参数设置、控制、故障检测等方面的介绍，读者应该能够掌握驱动装置的基本应用。

# 第8章 精简系列面板的组态与应用

## 8.1 人机界面

### 8.1.1 人机界面与触摸屏

#### 1. 人机界面

人机界面(Human Machine Interface)简称HMI。从广义上说，人机界面泛指计算机(包括PLC)与操作人员交换信息的设备。在控制领域，人机界面一般特指用于操作人员与控制系统之间进行对话和相互作用的专用设备。人机界面可以在恶劣的工业环境中长时间连续运行，是PLC的最佳搭档。

人机界面可以用字符、图形和动画动态显示现场数据和状态，操作人员可以通过人机界面来控制现场的被控对象。此外人机界面还有报警、用户管理、数据记录、趋势图、配方管理、显示和打印报表、通信等功能。

过去应用人机界面的主要的障碍是它的价格较高，随着技术的发展和应用的普及，近年来人机界面的价格已经大幅下降，一个大规模应用人机界面的时代正在到来，人机界面已经成为现代工业控制系统必不可少的设备之一。

#### 2. 触摸屏

触摸屏是人机界面的发展方向，用户可以在触摸屏的屏幕上生成满足自己要求的触摸式按键。触摸屏的面积小，使用直观方便，易于操作。画面上的按钮和指示灯可以取代相应的硬件元件，减少PLC需要的I/O点数，降低系统的成本，提高设备的性能和附加价值。

STN液晶显示器支持的彩色数有限(例如8色或16色)，被称为“伪彩”显示器。STN显示器的图像质量较差，可视角度较小，但是功耗小、价格低，用于要求较低的场合。

TFT液晶显示器又称为“真彩”显示器，每一液晶像素点都用集成在其后的薄膜晶体管来驱动，其色彩逼真、亮度高、对比度和层次感强、反应时间短、可视角度大，但是耗电较多，成本较高，可用于要求较高的场合。

#### 3. 人机界面的工作原理

人机界面最基本的功能是显示现场设备(通常是PLC)中位变量的状态和寄存器中数字变量的值，用监控画面中的按钮向PLC发出各种命令，和修改PLC寄存器中的参数。

##### (1) 对监控画面组态

首先需要用计算机上运行的组态软件对人机界面组态(见图8-1)。“组态”(Configuration)一词有配置和参数设置的意思。使用计算机上运行的组态软件，可以很容易地生成满足用户要求的人机界面的画面，用文字或图形动态地显示PLC中位变量的状态和数字量的数值。用各种输入方式，将操作人员的位变量命令和数字设定值传送到PLC。画面的生成是可视化的，一般不需要用户编程，组态软件的使用简单方便，很容易掌握。

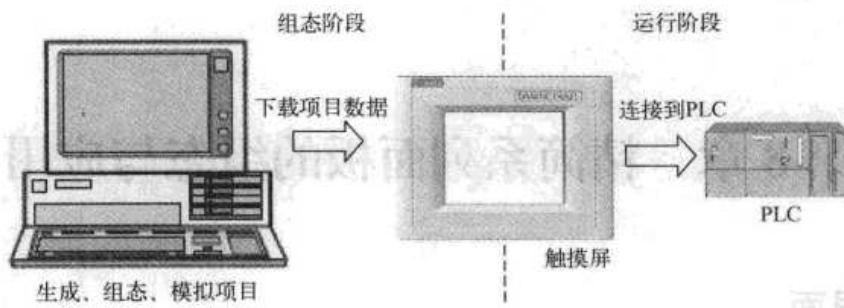


图 8-1 人机界面的工作原理

### (2) 编译和下载项目文件

编译项目文件是指将建立的画面及设置的信息转换成人机界面可以执行的文件。编译成功后，需要将可执行文件下载到人机界面的存储器中。

### (3) 运行阶段

在控制系统运行时，人机界面和 PLC 之间通过通信来交换信息，从而实现人机界面的各种功能。只需要对通信参数进行简单的组态，就可以实现人机界面与 PLC 的通信。将画面中的图形对象与 PLC 的存储器地址联系起来，就可以实现控制系统运行时 PLC 与人机界面之间的自动数据交换。

## 8.1.2 SIMATIC HMI 精简系列面板

### 1. 精简系列面板简介

PLC 的人机接口功能较差，S7-1200 与 SIMATIC HMI 精简系列面板（又称为基本面板，见图 8-2）无缝兼容，极大地增强了控制系统的显示和操作功能，为小型自动化应用提供了一种简单经济的可视化和控制解决方案。SIMATIC HMI 精简系列面板具有基本的功能，操作简单，界面直观，价格较低，具有很高的可靠性和性能价格比。本章介绍用 STEP 7 Basic 集成的 SIMATIC WinCC Basic 来对精简系列面板组态和仿真的方法。精简系列面板也可以用 WinCC flexible 2008 来组态（见参考文献[3]）。

精简系列面板的组态数据多达 32 种语言，可以在全球范围内使用。运行时可以使用多达 5 种语言，并且能在线切换语言。

精简系列面板的组态和使用方法可以参阅随书光盘中的手册《精简系列面板操作说明》。

### 2. 精简系列面板的硬件

精简系列面板配有一个触摸屏，操作直观方便，可以显示矢量图、趋势图、文本、位图和 I/O 域等。4in、6in 或 10in 的面板还有带触摸反馈的可编程按键。如果需要更大的显示尺寸，可以选择 15 英寸的触摸屏。精简系列面板的防护等级为 IP 65，可以在恶劣的工业环境

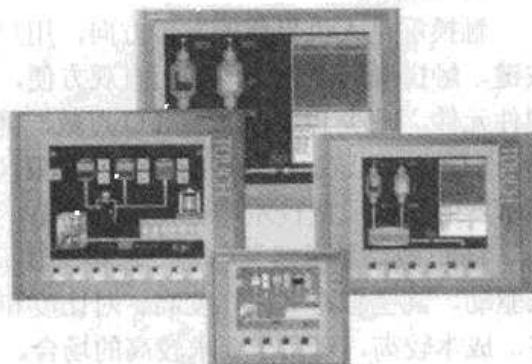


图 8-2 SIMATIC HMI 精简系列面板

中使用。

SIMATIC HMI 精简系列面板采用模拟电阻式触摸屏，它们均有 RJ-45 以太网接口，通信速率为 10/100 Mbit/s。精简系列面板集成的 PROFINET 接口可以用于与组态计算机或 S7-1200 方便地通信。电源电压额定值为 DC 24V，有内部熔断器和内部的实时钟。与 S7-1200 配套的精简系列面板的主要性能指标见表 8-1。

表 8-1 SIMATIC HMI 精简系列面板的主要性能指标

	KTP400 mono PN	KTP600 mono PN	KTP600 PN	KTP1000 PN	TP1500 PN
LCD 显示器	单色 FSTN	单色 FSTN	TFT	TFT	TFT
显示器尺寸	3.8in	5.7in		10.4in	15in
像素点数	320×240			640×480	1024×768
可显示的颜色数	4 级灰度	4 级灰度	256	256	256
功能键个数	4	6		8	—
应用程序存储器	512KB			1024KB	
电流消耗典型值	70 mA	240 mA	350 mA	600 mA	800 mA
最大恒定电流消耗	150 mA	350 mA	550 mA	1000mA	

### 3. WinCC basic 简介

S7-1200 的编程软件 STEP 7 Basic 与精简系列面板的组态软件 WinCC Basic 集成在一起，后者简单、高效，易于上手，功能强大。基于表格的编辑器简化了变量、文本和报警信息等的生成和编辑。通过图形化配置，简化了复杂的组态任务。

S7-1200 与精简系列面板用同一个软件，在同一个项目中组态和编程，它们都采用以太网接口通信，以上特点使精简系列面板成为 S7-1200 的最佳搭档。

精简系列面板具有报警、配方管理、趋势图、用户管理等功能。工程组态系统提供了具有大量图形和组态对象的库，可以用于画面组态。

WinCC Basic 的运行系统可以对精简系列面板仿真，这种仿真功能对于学习精简系列面板的组态方法是非常有用的。

## 8.2 精简系列面板的画面组态

### 8.2.1 使用 HMI 设备向导生成 HMI 设备

打开 STEP 7 Basic，生成一个名为 PLC\_HMI 的新项目（见随书光盘中的同名例程）。

双击项目树中的“Add new device”，添加一个新设备。点击打开的对话框中的“SIMATIC PLC”按钮（见图 8-3），选中中间窗口的 CPU 1214C，点击“OK”按钮，生成名为“PLC\_1”的新 PLC。

双击项目下面的“Add new device”，点击打开的对话框中的“SIMATIC HMI”按钮（见图 8-3），选中中间窗口的 KTP600 PN，HMI 列表中的 Portrait 表示画面竖直放置。点击“OK”按钮，生成名为“HMI\_1”的面板，出现“HMI Device Wizard: KTP600 PN”（HMI 设备向导）视图（见图 8-4）。

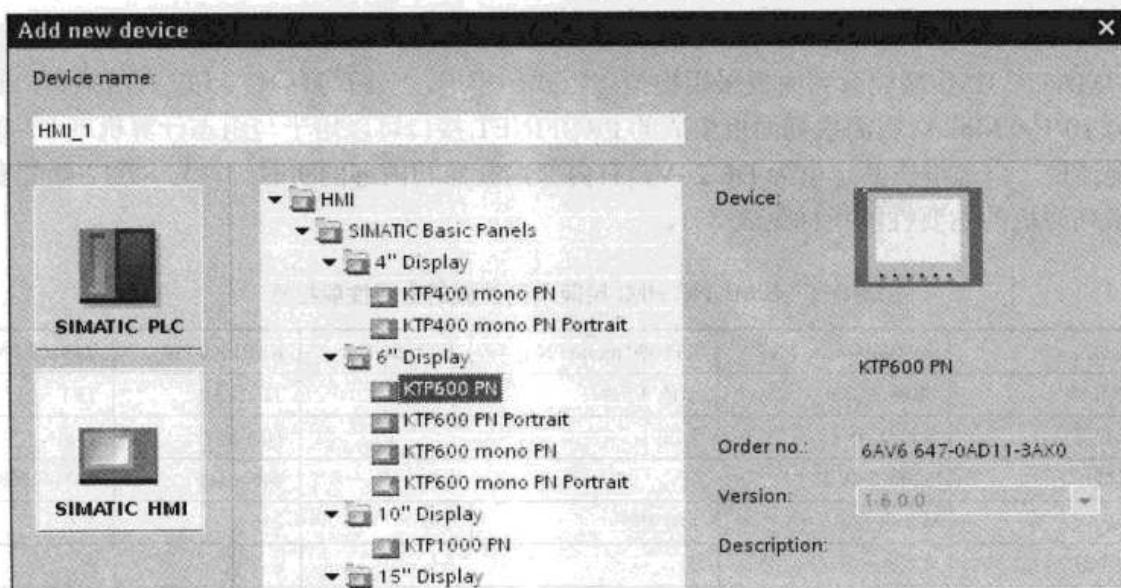


图 8-3 添加 HMI

首先打开的是 PLC 连接 (PLC connection) 对话框 (见图 8-4)，左边的橙色“圆球”用来表示当前的进度。最右边的选择框上面的文字 “No PLC selected” 表示没有选择 PLC。点击该选择框右边的 ▾ 按钮，双击出现的 PLC 列表中的 PLC\_1，出现 PLC 图标和两台设置之间的连线 (见图 8-5)。

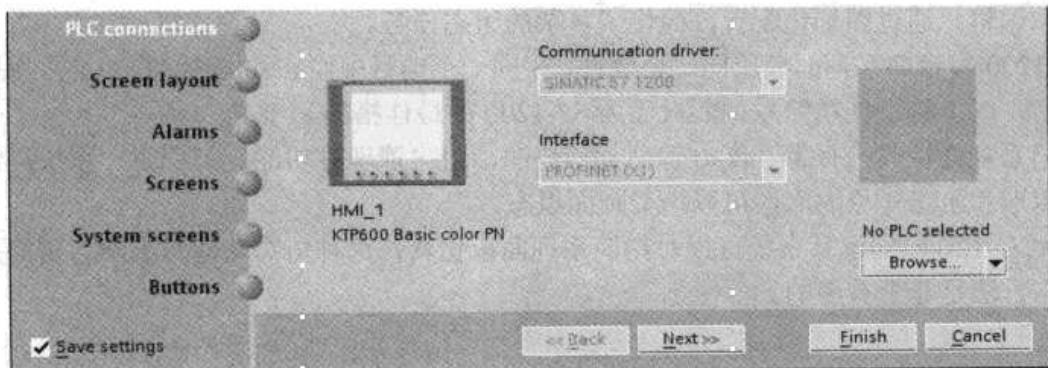


图 8-4 HMI 设备向导

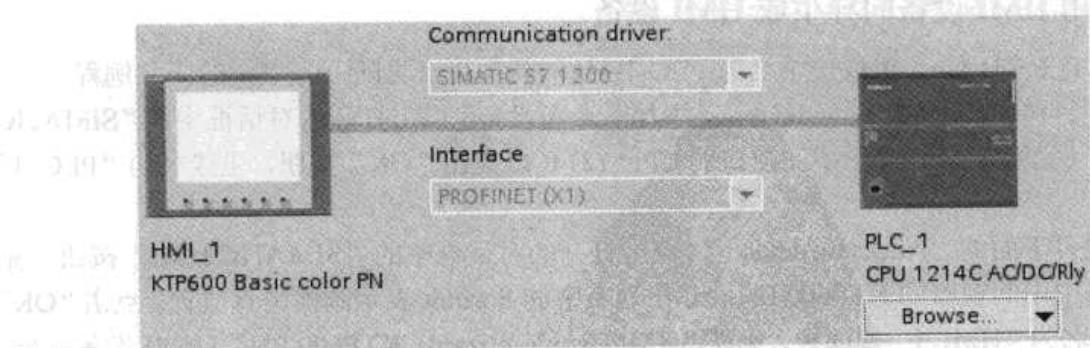


图 8-5 组态 HMI 与 PLC 的连接

点击“Next”按钮，打开画面布局（Screen layout）对话框（见图 8-6），右边的“Preview”区是生成的画面的预览。点击“Background color”（背景色）选择框右边的  按钮，用出现的颜色窗口将画面的背景色改为白色。点击左边的复选框“Page header”，使“”消失，取消页面的标题。

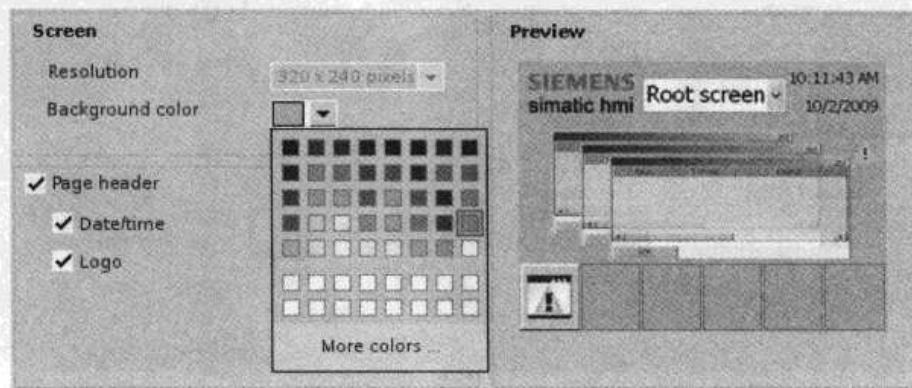


图 8-6 组态画面布局

点击“Next”按钮，打开报警（Alarms）对话框（见图 8-7）。点击复选框“Unacknowledged alarms”（不需确认的报警）和“Active system events”，使“”消失。画面预览中的报警窗口由 3 个减少为 1 个，只保留了激活的报警窗口（Active alarms）。

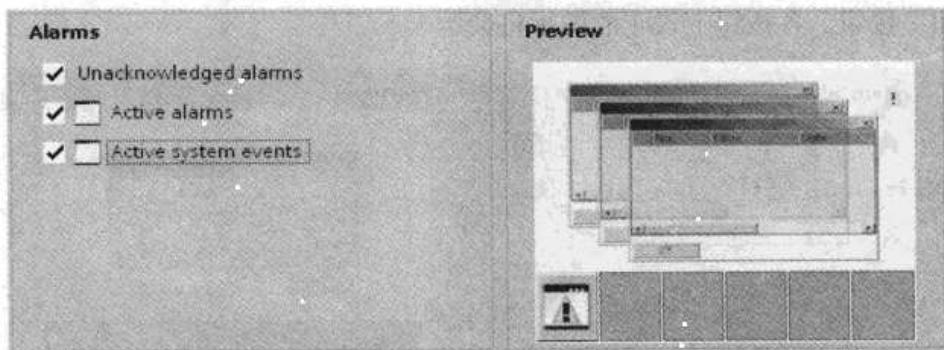


图 8-7 组态报警

点击“Next”按钮，打开画面（Screens）对话框（见图 8-8）。开始时只有初始画面（Root screen）。每点击一次它里面的  按钮，将生成一个下一级的画面（例如 Screen0）。点击 Screen0 里面的  按钮，将生成一个再下一级的画面（例如 Screen2）。选中某个画面，可以用对话框上面的工具栏上的按钮对选中的画面进行改名或删除的操作。

本项目只生成了 Root screen 和 Screen0。如果需要，以后还可以在项目树中生成新的画面。

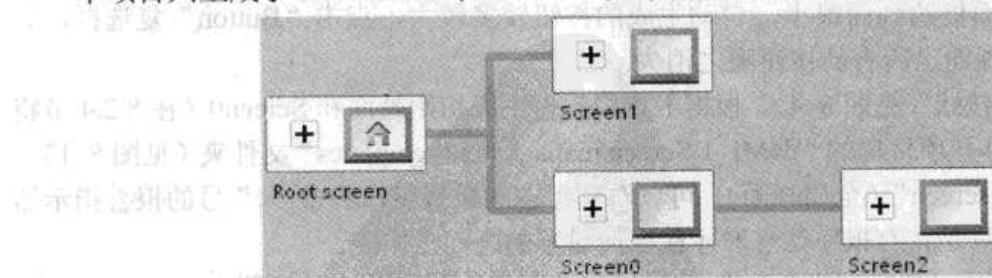


图 8-8 生成画面

点击“Next”按钮，打开系统画面（System screens）对话框。图 8-9 生成了系统画面和用户管理画面。本项目未生成任何系统画面。

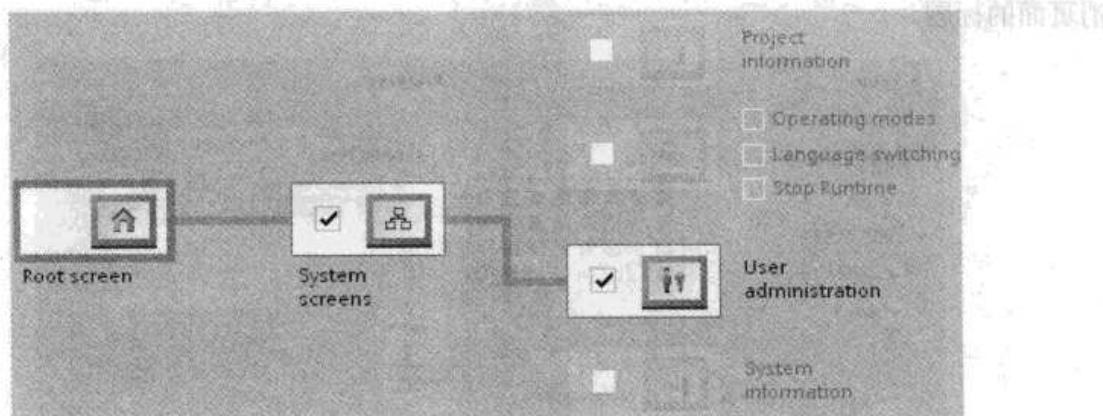


图 8-9 生成系统画面

点击“Next”按钮，打开按钮（Buttons）对话框（见图 8-10）。点击左边的某个系统按钮（System button），该按钮上的图标将出现在画面上最左边未放置图标的位置上。选中“Button area”（按钮区）中的“Left”或“Right”复选框，将在画面的左边或右边生成新的按钮。点击“Reset all”按钮，各按钮上设置的图标将会消失。

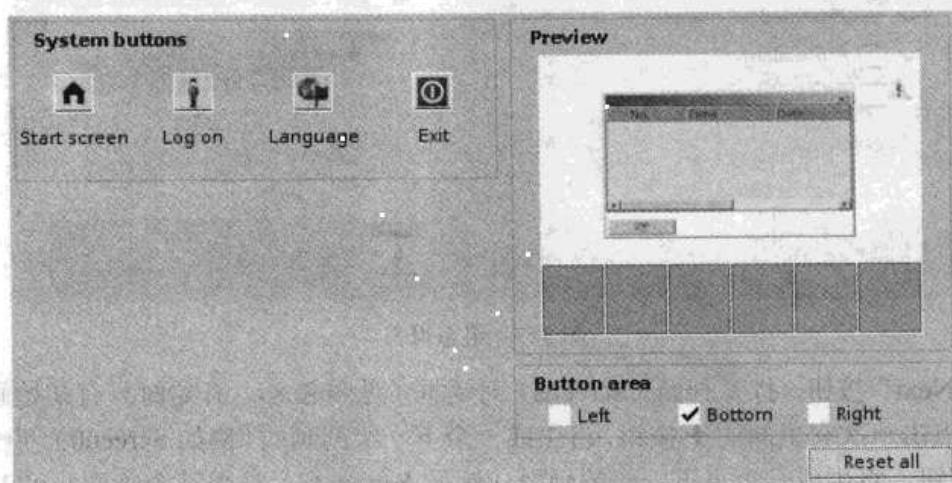


图 8-10 组态按钮

考虑到选中的面板的画面很小，自动生成的按钮较多较大，点击“Button”复选框，其中的“√”消失，画面上所有的按钮随之消失。

点击“Finish”按钮（见图 8-4），根据上述组态，生成初始画面和 Screen0（在 8.2.4 节将它改名为 Level）。打开项目树的“\HMI\_1\Screen management\templates”文件夹（见图 8-13），双击其中的“Globe screen”（全局画面），可以看到组态的报警窗口和带“!”号的报警指示器（见图 8-10）。它们只是在有报警信息时才在当前显示的画面上出现。

下一次生成 HMI 对象时，将会使用本次在 HMI 设备向导中组态的设置。

## 8.2.2 组态指示灯

### 1. 编写 PLC 程序

在 PLC 变量表 (PLC tags) 中生成 5 个变量 (见图 8-11)，M2.0 和 M2.1 是面板的画面上的按钮提供给 PLC 的控制信号。MB0 和 MB1 一般保留给时钟存储器和系统存储器使用。在 OB1 中生成水泵的控制程序 (见图 8-12)，MOVE 指令的操作没有任何实际的意义，只是用来检验 PLC 和 HMI 之间的数据传送是否成功。

PLC tags			
Name	Datatype	Address	Retain
1 Pump	Bool	%Q0.0	
2 Start	Bool	%M2.0	
3 Stop	Bool	%M2.1	
4 PresetLevel	UInt	%MB12	
5 ActualLevel	UInt	%MB14	

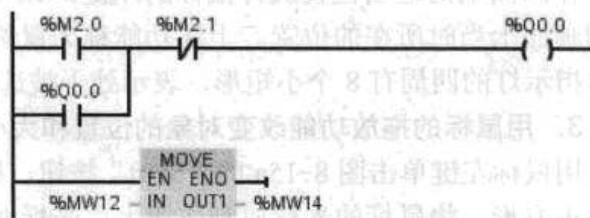


图 8-11 PLC 变量表

图 8-12 OB1 中的梯形图程序

### 2. 生成指示灯

双击打开项目树中的“Root screen”(初始画面)，选中画面上的欢迎文本 (Welcome to HMI\_1) 并删除它。右边任务卡窗口自动打开了工具箱 (Toolbox)。图 8-13 的工作区中是已组态好的初始画面。

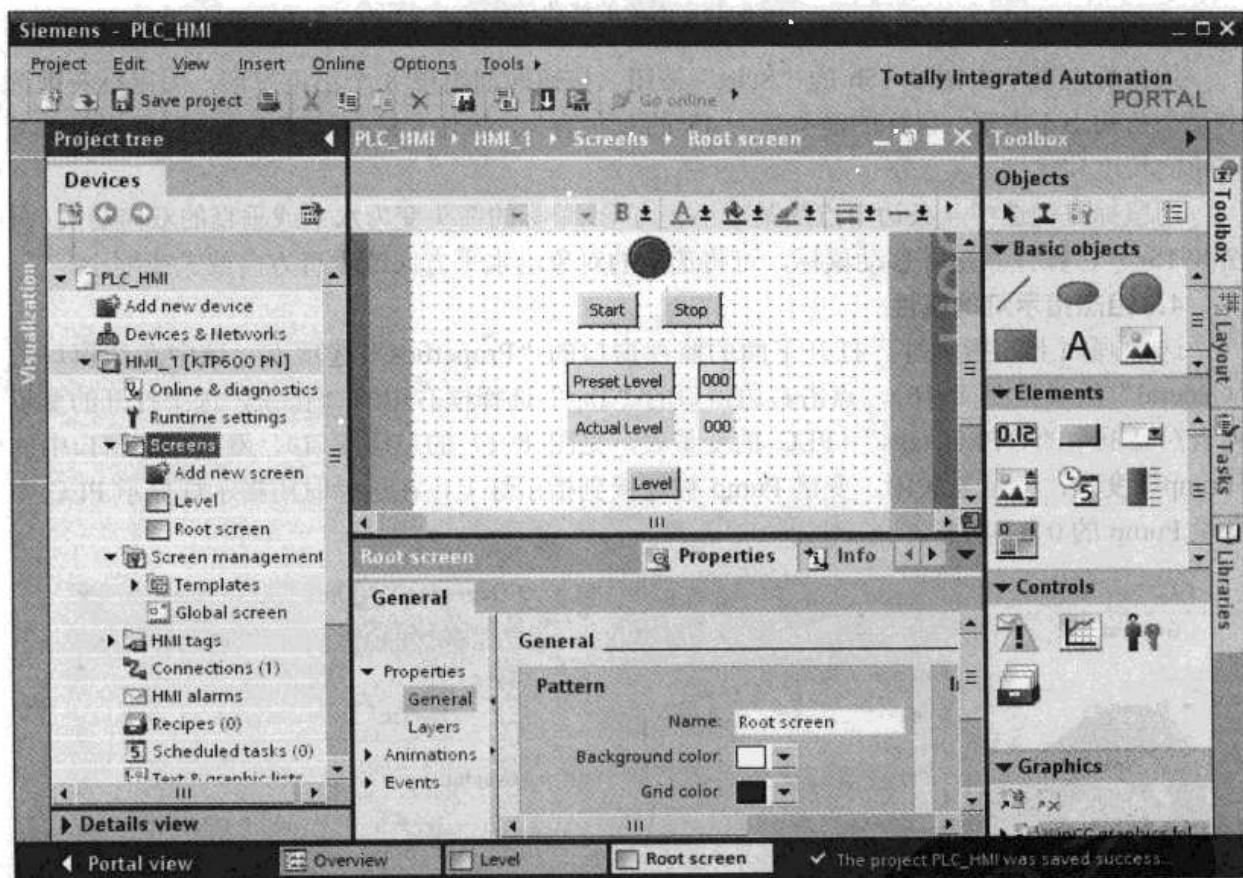


图 8-13 WinCC Basic 视图

点击最右边垂直放置的“Libraries”（库）按钮，打开全局库（Global libraries）中的 PlotLights（指示灯，见图 8-14）文件夹，用鼠标左键按住其中的绿色指示灯 PlotLight\_Round\_G 不放，同时移动鼠标，矩形的指示灯图形跟随鼠标的光标（禁止放置）一起移动，移动到画面工作区时，鼠标的光标变为（可以放置）。

在画面上的适当位置放开鼠标的左键，该按钮对象便被放置到画面上当时所在的位置。上述功能称为鼠标的拖放功能。此时指示灯的四周有 8 个小矩形，表示处于被选中的状态。

### 3. 用鼠标的拖放功能改变对象的位置和大小

用鼠标左键单击图 8-15a 的“Start”按钮，按钮四周出现 8 个小正方形。将鼠标的光标放到按钮上，光标变为图中的十字箭头图形。按住鼠标左键并移动鼠标，将选中的对象拖到希望的位置，松开左键，对象被放在该位置。



图 8-15 对象的移动与缩放

用鼠标左键单击图 8-15b 的“Start”按钮，按钮四周出现 8 个小正方形，用鼠标左键选中某个角的小正方形，鼠标的箭头变为 45° 的双向箭头，按住左键并移动鼠标，可以同时改变按钮的长度和宽度。

用鼠标左键选中 4 条边中点的某个小正方形，鼠标的箭头变为水平或垂直的双向箭头（见图 8-15c），按住左键并移动鼠标，可将选中的对象沿水平方向或垂直方向放大或缩小。

### 4. 组态指示灯的属性

选中画面上的指示灯，打开下面的监视窗口的“Properties”选项卡，选中左边窗口的“General”组（见图 8-16），点击右边窗口的“Tag”选择框右边的…按钮，选中打开的变量列表对话框中的“PLC tags”（PLC 的变量表，见图 8-17 的左边窗口），双击右边窗口中的“Pump”变量，变量 Pump 被连接到指示灯上，在运行时用指示灯显示 PLC 中变量 Pump 的 0、1 状态。

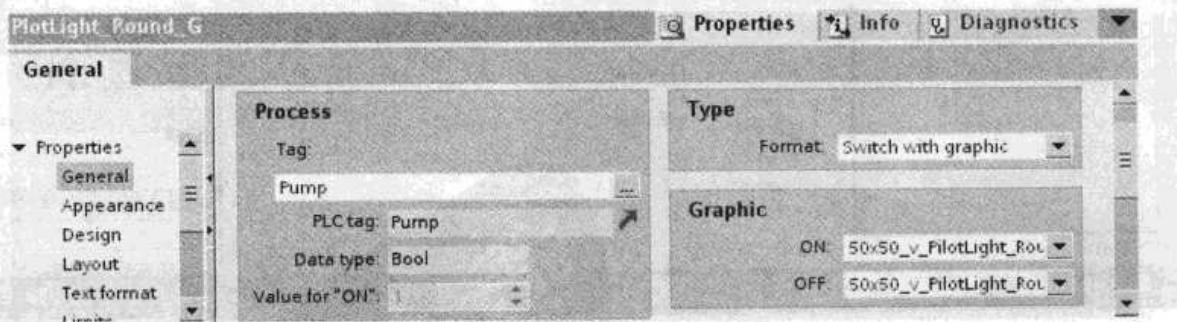


图 8-16 组态指示灯的属性

打开图 8-17 左边窗口中的 HMI tags(HMI 的变量表), 可以看到 PLC tags 中的变量 Pump 出现在 HMI tags 中。

点击图 8-16 中间窗口的 按钮, 将在工作区打开 PLC 变量表编辑器。

刚刚生成指示灯 时, 其正方形的背景色为灰色。选中图 8-16 左边窗口的“Appearance”(外观), 将其背景色 (Background) 改为与画面的背景色相同的白色 (见图 8-18)。

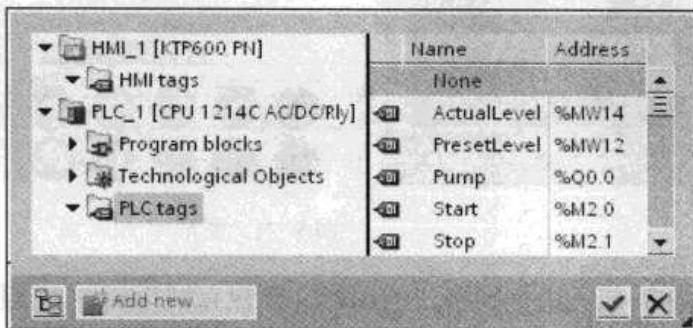


图 8-17 变量列表

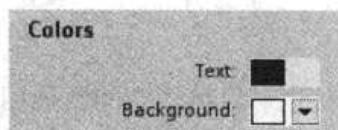


图 8-18 组态背景色

## 5. 用图形 IO 域生成指示灯

本例使用的指示灯是用工具箱的“Elements”组中的开关元件生成的, 它可以显示位元件的状态, 同时又有控制位元件的功能。在运行时点击它, 可以使它连接的 PLC 的变量 Pump 的值变为相反的值, 指示灯的状态随之而变。“真正”的指示灯应该只有显示功能, 没有控制功能。可以用工具箱中的图形 IO 域实现这种指示灯的功能。

将工具箱的“Elements”组中的图形 IO 域的图标 拖放到画面上, 选中它以后, 打开下面的监视窗口的“Properties”选项卡 (见图 8-19), 选中左边窗口的“General”组, 在右边窗口中, 设置连接的变量为“Pump”, “Type”区的“Mode”(模式)为默认的“Two status”(双状态)。点击右边的“On:”选择框右边的 按钮, 打开项目的小图库 (见图 8-20), 选中前述的全局库中的指示灯用于 ON 状态的图形, 右边显示的是选中的图形。点击图 8-19 右边的“Off:”选择框右边的 按钮, 用同样的方法, 选中前述的全局库中的指示灯用于 OFF 状态的图形。

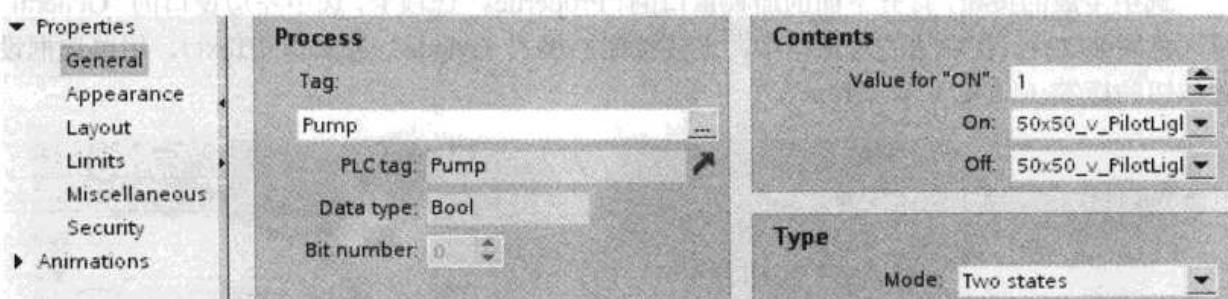


图 8-19 组态图形 IO 域的属性

可以将图库中的图形用于指示灯。打开工具箱最下面的图库 (Graphics) 文件夹中的“3-DPushbuttons Etc”(3 维按钮等)文件夹 (见图 8-21), 将左边的绿灯和暗绿灯图形拖放

到生成的图形 IO 域中，分别用它们来作指示灯 On 和 Off 状态的图形。

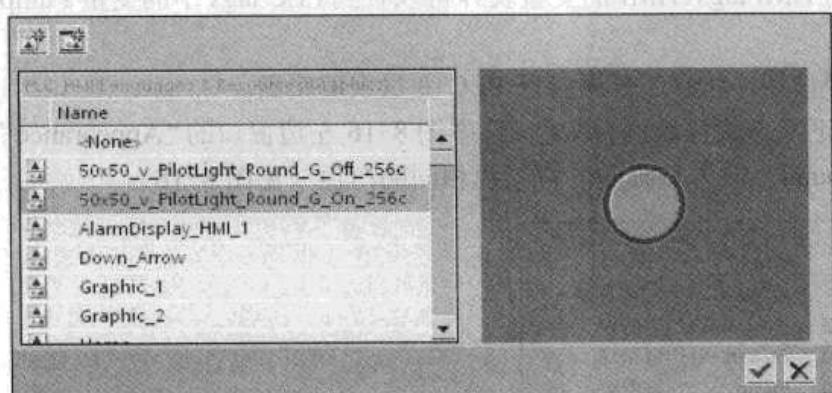


图 8-20 图形列表

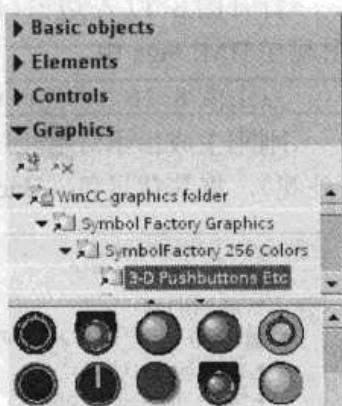


图 8-21 图形库

也可以将任意的其他图形用于图形 IO 域。用标准格式的图形文件保存图形后，点击图 8-20 左上角的 按钮，导入来自文件的新图形，然后将它用于图形 IO 域。WinCC Basic 几乎支持所有的图形格式的文件。

### 8.2.3 组态按钮

#### 1. 按钮的生成

画面上的按钮与接在 PLC 输入端的物理按钮的功能相同，主要用来给 PLC 提供数字量（即开关量）输入信号，通过 PLC 的用户程序来控制生产过程。画面上的按钮元件不能与 PLC 接收硬件输入信号的输入点（例如 I0.0）连接，应与存储器位（例如 M2.0）连接。

将工具箱的“Elements”（元件）组中的按钮图标 （见图 8-13）拖放到画面上，按钮图标跟随光标一起移动，按钮左上角在画面上的 X、Y 轴的坐标（X/Y）和按钮的宽、高（w/h）尺寸（均以像素为单位）也跟随光标一起移动。放开鼠标左键，按钮被放置在画面上。此时按钮的四周有 8 个小正方形，可以用前面介绍的鼠标的使用方法，根据需要来调整按钮的位置和大小。

#### 2. 按钮的属性设置

选中生成的按钮，打开下面的监视窗口的“Properties”选项卡，选中左边窗口的“General”组（见图 8-22），在右边的对话框中，设置按钮的模式（Mode）为文本（Text），用单选框设置按钮的标签（Label）为“Text”。

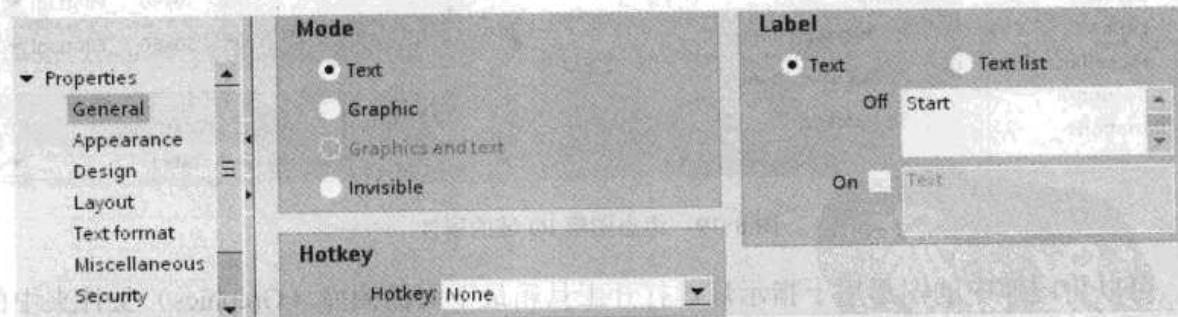


图 8-22 组态按钮的常规属性

如果选中复选框“On”，可以分别设置“On”（按下）和“Off”（弹起）的文本。未选中该复选框时，按钮按下时和弹起时显示的文本相同。

选中图 8-22 左边窗口的“Appearance”（外观），可以在右边窗口修改它的背景色（Background）和文本（Text）的颜色（见图 8-23）。选中图 8-22 左边窗口的“Design”（图样），可以用右边窗口的复选框设置按钮是否有三维（3D）效果（见图 8-24）。

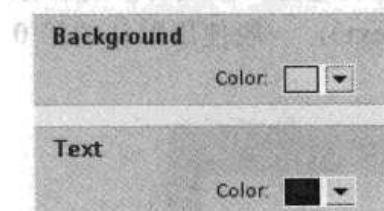


图 8-23 组态按钮的外观



图 8-24 组态按钮的图样

在运行时如果点击了该按钮，它称为焦点（Focus），它的周围出现一个方框。按钮上的字符“Start”周围出现一个虚线的方框。焦点颜色（Focus Color，见图 8-24）用来设置表示焦点的虚线框的颜色，一般采用默认的设置。

选中图 8-22 左边窗口的“Layout”（布局）组，如果选中右边窗口的复选框“Fit object to contents”（自动调整大小，见图 8-25），将根据按钮上的文本字数和字体大小自动调整按钮的大小。

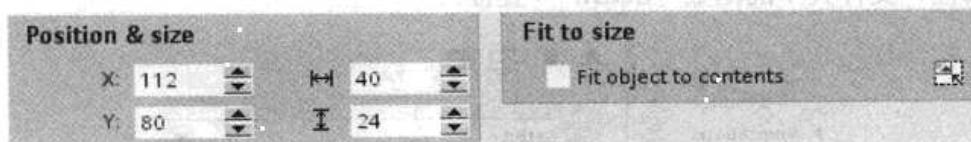


图 8-25 组态按钮的布局

可以通过“Position & size”区的输入框修改对象的 X、Y 坐标和尺寸（均以像素为单位）。一般在画面上直接用鼠标设置画面元件的位置和大小，这样比在布局属性对话框中修改参数更为直观。

选中图 8-22 左边窗口的“Text format”（文本格式），点击右边窗口的“Font”（字体）选择框右边的...按钮（见图 8-26），可以用打开的“Font”对话框设置文本的字体和以像素为单位的大小，以及设置文本的特殊效果，例如加粗、斜体和下划线等。

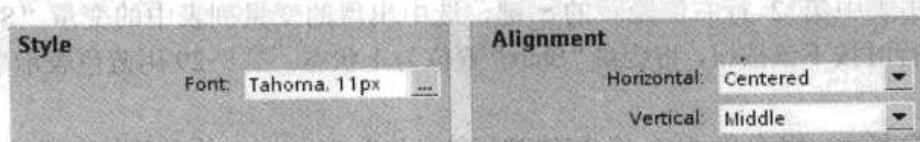


图 8-26 组态按钮的文本格式

在“Alignment”区可以用“Horizontal”（水平）选择框设置文本居左、居中或居右，用“Vertical”（垂直）选择框设置文本偏上、居中或偏下。

选中图 8-22 左边窗口的“Security”（安全）组，激活右边窗口中的复选框“Allow operator

control”(图 8-27), 在 HMI 运行时才允许操作员对按钮进行操作。

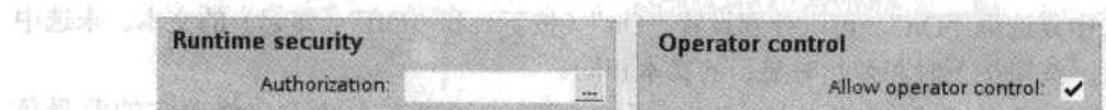


图 8-27 按钮的安全组态

选中图 8-22 左边窗口的“Miscellaneous”(其他), 可以在右边窗口修改对象的名称(见图 8-28), 设置对象所在的“层”, 和输入信息文本(Infotext), 一般使用默认的第 0 层。

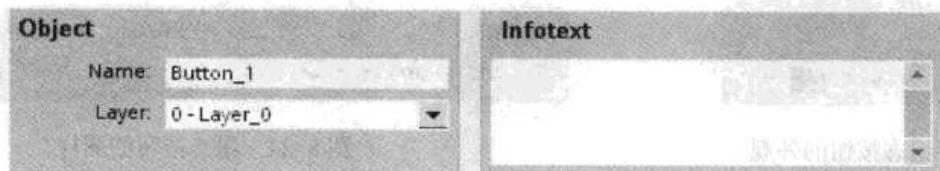


图 8-28 按钮的其他组态

### 3. 按钮功能的设置

选中画面上文本为“Start”的按钮, 打开下面的监视窗口左边的“Events”(事件)文件夹的“Press”(按下)组(见图 8-29), 点击右边窗口标有“Add function”的最上面一行, 再点击它的右侧出现的 $\square$ 键(在点击之前它是隐藏的), 在出现的系统函数列表中, 双击“Edit bits”(编辑位)文件夹中的函数“SetBit”(置位)。

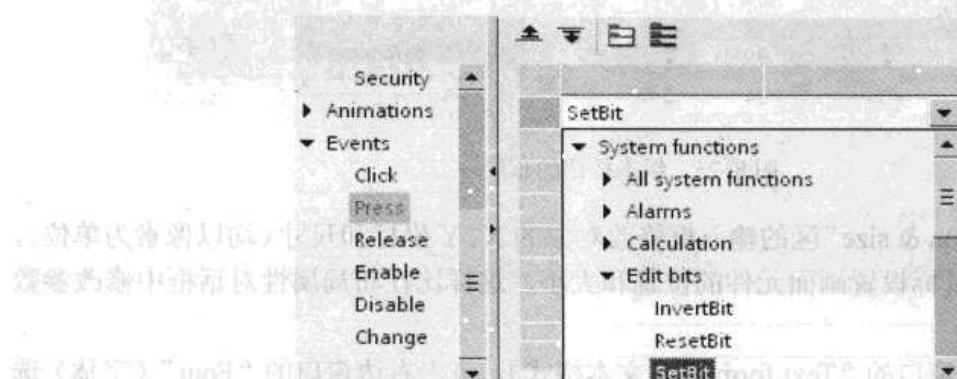


图 8-29 组态按钮按下时执行的函数

直接点击表中第 2 行右侧隐藏的 $\square$ 键, 选中出现的变量列表中的变量“Start”(见图 8-30), 在运行时按下该按钮, 将变量“Start”置位为 1 状态。图 8-29 用蓝色表示事件“Press”设置了功能。

用同样的方法, 组态在“Start”按钮被释放(Release)时, 执行系统函数“ResetBit”, 将变量 Start 复位。该按钮具有点动按钮的功能, 按下按钮时变量“Start”被置位为 1 状态, 放开按钮时它被复位为 0 状态。

组态好启动按钮后, 选中该按钮, 复制和粘贴一个相同的按钮, 将按钮上的文本改为“Stop”, 按下和释放按钮时分别将变量“Stop”置位和复位。

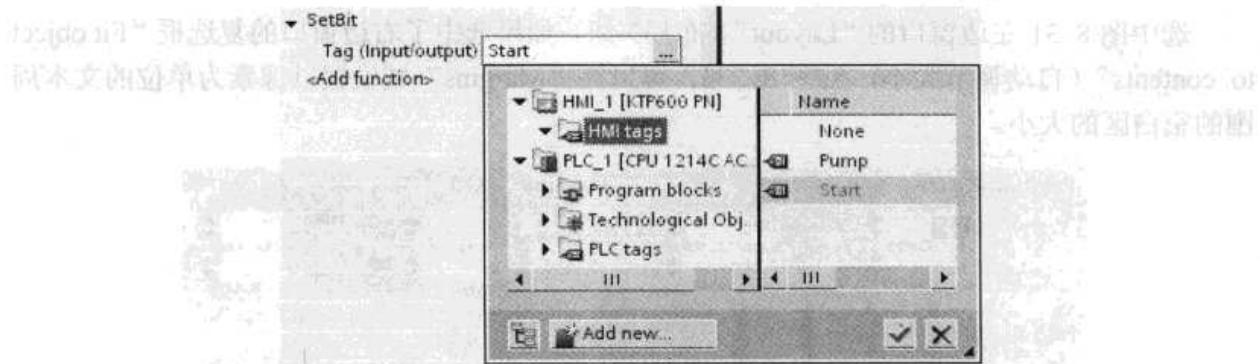


图 8-30 组态按钮按下时操作的变量

## 8.2.4 组态文本域与 IO 域

### 1. 生成与组态文本域

将工具箱中带有“**A**”的文本域图标（见图 8-13）拖放到画面上，默认的文本为“Text”。双击生成的文本域，可以直接在文本域中输入需要的文字。也可以选中下面的监视窗口左边的“General”组（见图 8-31），在右边的“Text”文本框中输入文本。

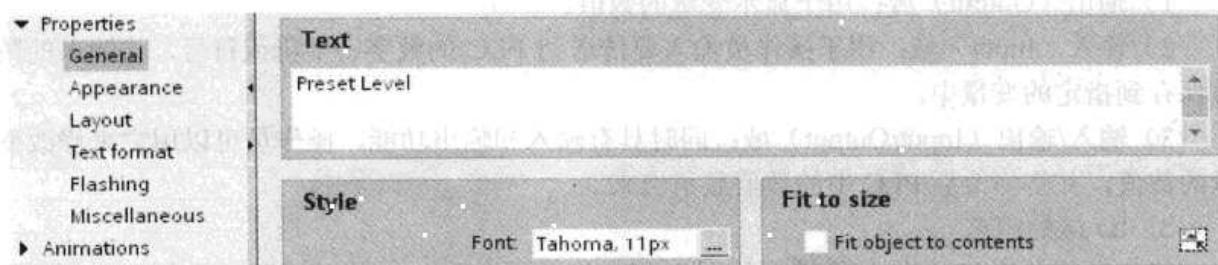


图 8-31 组态文本域

可以用“Font”（字体）选择框设置文本的字体和以像素为单位的大小，以及设置文本的特殊效果。选中复选框“Fit object to contents”（自动调整大小），将根据文本的字数和字体的大小自动调整文本域的大小。

选中图 8-31 左边窗口的“Appearance”（外观）组，可以在右边窗口（图 8-32）修改它的背景色（Background）、填充式样（Fill pattern）和文本的颜色。边框（Border）域中的“Style”选择框可以选择没有边框（None）和有边框（Solid），用“Width”输入域设置边框的宽度，用 Color 选择框设置边框的颜色，还可以用复选框设置是否有三维（3D）效果。

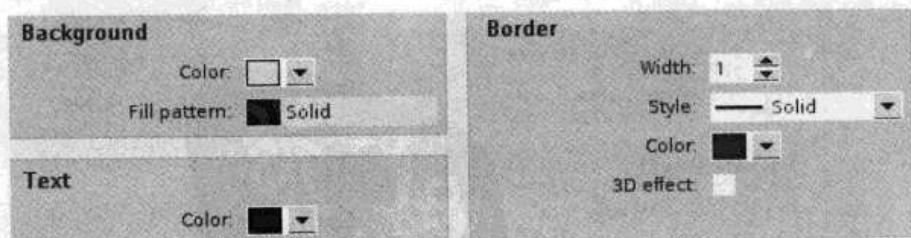


图 8-32 组态文本域的外观

选中图 8-31 左边窗口的“Layout”(布局)组,如果选中了右边窗口的复选框“Fit object to contents”(自动调节大小,见图 8-33),可以在“Margins”域设置以像素为单位的文本周围的空白区的大小。

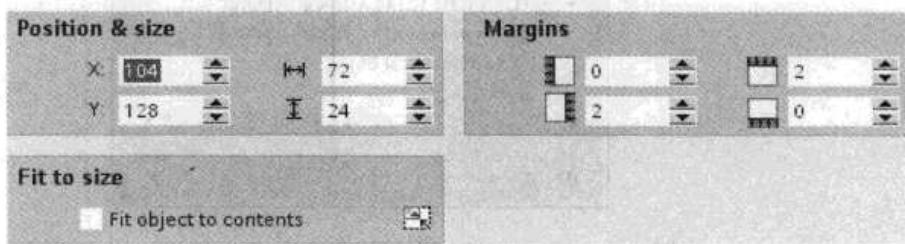


图 8-33 组态文本域的布局

监视窗口中的“Text format”(文本格式)对话框与按钮的相同。

选中画面上生成的文本域,执行复制和粘贴功能,将生成的文本域的文字改为“Actual Level”,没有边框。

## 2. IO 域的分类

IO 域分为 3 种模式:

- 1) 输出 (Output) 域: 用于显示变量的数值。
- 2) 输入 (Input) 域: 用于操作员输入要传送到 PLC 的数字、字母或符号, 将输入的数值保存到指定的变量中。
- 3) 输入/输出 (Input/Output) 域: 同时具有输入和输出功能, 操作员可以用它来修改变量的数值, 并将修改后 PLC 中的数值显示出来。

## 3. IO 域的组态

将工具箱中的 IO 域图标 (见图 8-13) 拖放到画面上, 选中生成的 IO 域, 打开下面的监视窗口的 Properties 选项卡, 选中左边窗口的 General 组 (见图 8-34), 用 Mode 选择框设置 IO 域为输入 (Input) 域。点击右边窗口的“Tag”选择框右边的 按钮, 选中打开的变量列表对话框中的“PLC tags”, 双击其中的“PresetLevel”变量, 该变量被分配给 IO 域。

在“Format”格式域, 显示模式为默认的十进制 (Decimal), 小数点后的位数为 0, 数据长度为 3 位, 不显示左边的无效 0 (Leading zeros)。因为变量 PresetLevel 的数据类型为最大 255 的 UInt (无符号字节), 显示格式 (Format pattern) 被自动设置为 999 (3 位整数)。

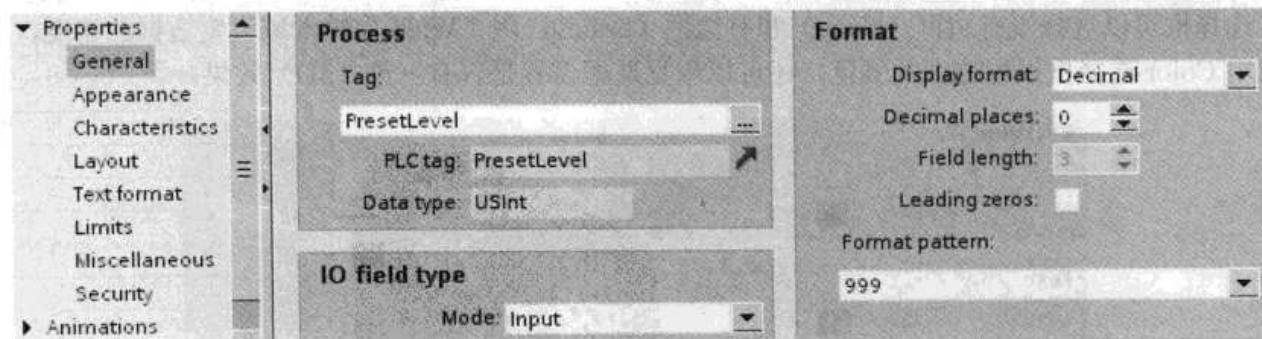


图 8-34 组态 IO 域

IO 域的监视窗口中的“Appearance”（外观）、“Layout”（布局）和“Text format”（文本格式）对话框与文本域的基本上相同。其区别在于 IO 域的“Layout”对话框中的“Margins”域默认的上、下、左、右的空白区均为 2 点像素。

选中图 8-34 左边窗口的“Characteristics”（特性），可以在右边窗口用复选框选择是否隐藏输入值（见图 8-35）。如果选中它，输入的数值用“\*”号显示。

选中图 8-34 左边窗口的“Limits”（限制），可以在右边窗口用选择框设置超过变量的上限值和下限值时的背景色（见图 8-36）。



图 8-35 组态 IO 域的特性

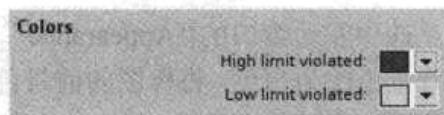


图 8-36 组态 IO 域超限的显示

选中画面上生成的 IO 域，执行复制和粘贴操作。将新生成的 IO 域放置好后选中它，在下面的监视窗口左边选中 General 组（见图 8-34），在右边窗口中组态该 IO 域连接的变量（Tag）为 ActualLevel，模式（Mode）改为 Output（输出），其余的参数不变。

#### 8.2.4 棒图的组态与画面切换

##### 1. 生成画面切换按钮

用鼠标右键点击项目树的“\HMI\_1\Screens”中的画面“Screen0”，执行出现的快捷菜单中的命令“Rename”，将它的名称改为“Level”。然后双击打开该画面。

该画面上有自动生成的画面切换按钮，上面的字符为“Root screen”（初始画面）。选中它后打开下面的监视窗口的“Properties”选项卡，可以看到自动生成的点击（Click）按钮时执行的切换画面的系统命令（见图 8-37）。

打开初始画面，可以看到自动生成的画面切换按钮，上面的字符为“Screen0”。删除该按钮后，将项目树中已改名称的画面“Level”的图标拖放到初始画面上，自动生成的画面切换按钮上的字符为 Level，这种生成画面切换按钮的方法特别简单直观。

##### 2. 用棒图对象显示液位

打开画面“Level”，将工具箱的“Element”组中的棒图图标拖放到画面上（见图 8-38）。用鼠标调节它的大小和位置。选中它后打开下面的监视窗口的“Properties”选项卡，选中左边窗口的“General”组，设置棒图连接的变量为“ActualLevel”（见图 8-39）。

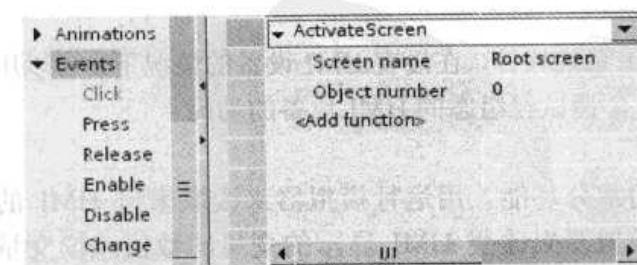


图 8-37 画面切换按钮的事件组态

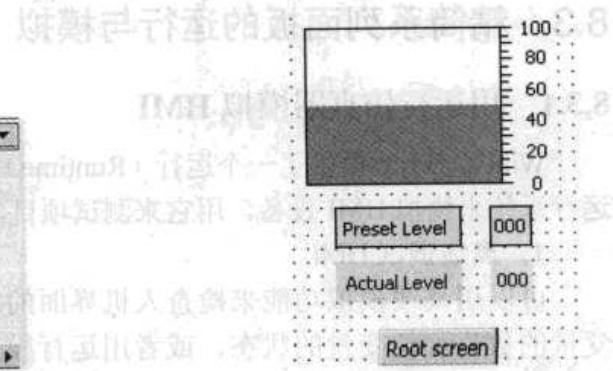


图 8-38 画面 Level

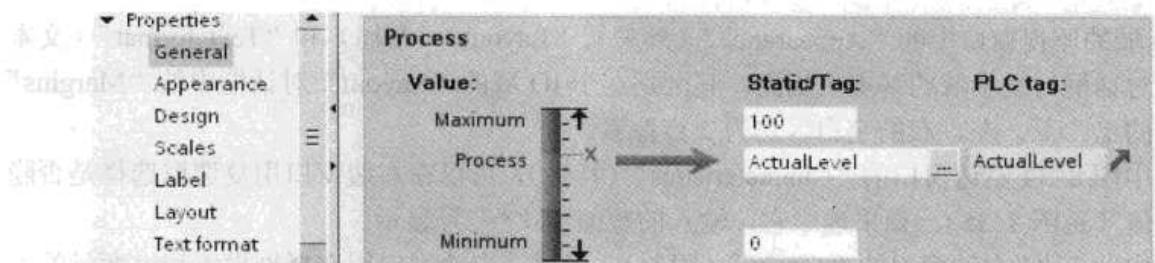


图 8-39 组态棒图

选中左边窗口的“Appearance”（外观，见图 8-40），将背景色（Background）和棒图（Bar）的背景色改为白色，将棒图的前景色（Foreground）改为绿色。点击“Limits”限制区的“Lines”复选框，去掉其中的“√”，禁止显示限制功能。

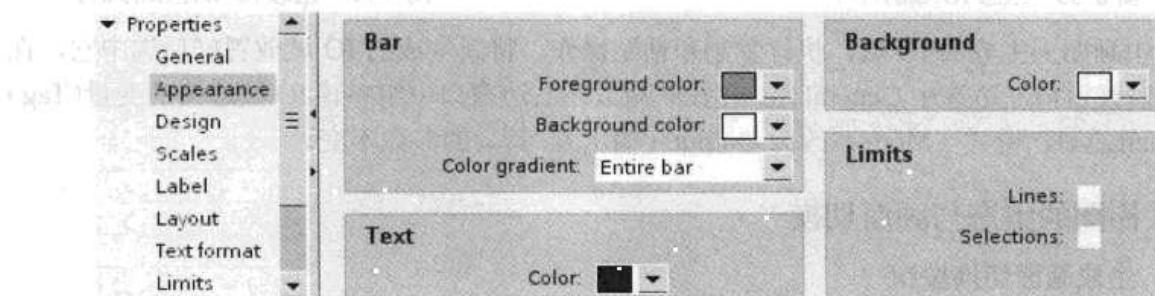


图 8-40 组态棒图的外观

选中图 8-40 中的“Scales”，可以选择是否显示刻度和刻度值，以及刻度的条数和刻度值的个数。修改参数后马上可以看到效果。

选中图 8-40 中的“Label”，可以设置刻度值的显示方法。

选中图 8-40 中的“Layout”，可以设置刻度的位置、方向和性质（双向或单向变化）。

“Text format”和“Limits”的设置方法与 IO 域的相同。

### 3. 复制画面对象

用鼠标左键同时选中初始画面（见图 8-13）上的两个文本域和两个 IO 域。点击工具栏上的复制按钮 ，将选中的对象复制到剪贴板。打开画面“Level”，点击工具栏上的粘贴按钮 ，将剪贴板上的对象粘贴到画面上（见图 8-38），适当调节粘贴的对象的位置。

## 8.3 精简系列面板的运行与模拟

### 8.3.1 用运行仿真器模拟 HMI

WinCC Basic 提供了一个运行（Runtime）系统软件，在没有 HMI 设备的情况下，可以用运行系统来模拟 HMI 设备，用它来测试项目，调试已组态的 HMI 设备的功能。

#### 1. 离线调试 HMI

可以用离线模拟功能来检查人机界面的部分功能。用运行模拟器来读取来自 HMI 的变量的数值和位变量的状态，或者用运行模拟器来改变 HMI 显示的变量的数值或位变量的状态。

因为没有运行 PLC 的用户程序，离线模拟只能模拟实际系统的部分功能。

## 2. 启动运行模拟器

选中 HMI\_1 站点，执行菜单命令“Online”→“Simulate runtime”→“With tag simulator”，启动运行模拟器。如果启动模拟器之前没有预先编译项目，则自动启动编译，编译成功后才能模拟运行。编译出现错误时，应改正错误。

编译成功后，将会同时出现模拟运行的画面（见图 8-41）和运行模拟器（见图 8-42）。

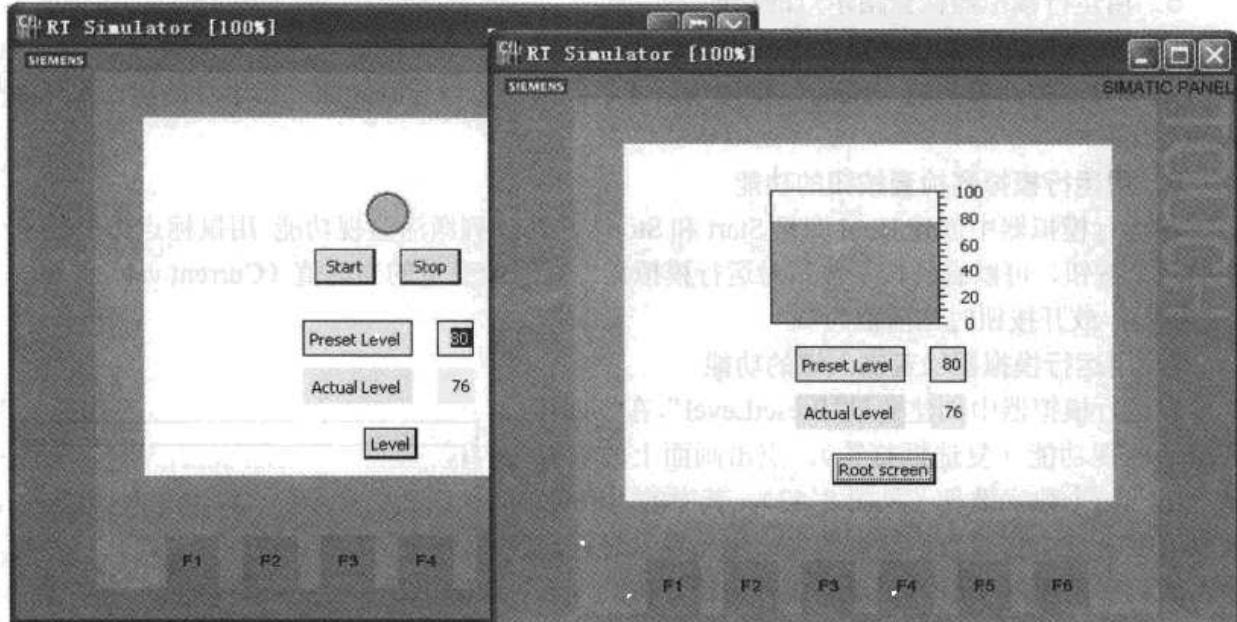


图 8-41 运行系统模拟运行的画面

Untitled - WinCC flexible Runtime Simulator												
Tag	Data Type	Current val.	Format	Write cycle (s)	Simulation	Set value	MinValue	MaxValue	Cycle	Start		
Stop	BOOL	0	Dec	1.0	<Display>	-1	0			<input checked="" type="checkbox"/>		
Start	BOOL	0	Dec	1.0	<Display>	-1	0			<input checked="" type="checkbox"/>		
PresetLevel	BYTE	80	Dec	1.0	<Display>	0	255			<input checked="" type="checkbox"/>		
ActualLevel	BYTE	76	Dec	1.0	<Display>	0	255			<input type="checkbox"/>		
Pump	BOOL	-1	Dec	1.0	<Display>	1	-1	0		<input type="checkbox"/>		
Pressure	INT	1000	Dec	1.0	<Display>	-32768	32767			<input type="checkbox"/>		

图 8-42 运行模拟器

## 3. 生成需要监控的变量

点击运行模拟器空白行的“Tag”列右边隐藏的按钮，双击出现的变量列表中某个要监控的变量，该变量出现在运行模拟器中。

## 4. 运行模拟器的参数

变量的名称和数据类型（Data Type）是变量本身固有的。其他参数是运行模拟器自动生成的。白色背景的参数可以修改，例如显示格式（Format）、改写周期（Write cycle）、最小值

为 1s)、模拟模式 (Simulation) 和设定值 (Set value)。对于位变量，模拟模式 (Simulation) 可选显示 (Display) 和随机 (Random)，其他数据类型的变量还可以选正弦 (sine)、递增 (Increment)、递减 (Decrement) 和移位 (Shift bit)。灰色背景的参数不能修改，例如图 8-42 中的 MinValue (最小值) 和 MaxValue (最大值)。模拟模式被选择为正弦、递增和递减时，可以用 Cycle 列设置以秒 (s) 为单位的变量的变化周期。可以将当前值 (Current value) 列视为 PLC 中的数据。

对于需要监视的变量，应选中 Start 列的复选框 (用鼠标打勾)。

#### 5. 用运行模拟器检查指示灯的功能

在运行模拟器中创建变量 “Pump” (见图 8-42)，在 “Set value” 列中分别设置该变量的值为 0 和 1，按计算机的〈Enter〉键确认后才起作用，可以看到画面上指示灯的状态发生相应的变化。

#### 6. 用运行模拟器检查按钮的功能

在运行模拟器中创建 Bool 变量 Start 和 Stop，在 Start 列激活监视功能。用鼠标点击图 8-41 中的两个按钮，可以看到按下按钮时运行模拟器中对应的变量的当前值 (Current value) 为 -1 (应为 1)，放开按钮时当前值为 0。

#### 7. 用运行模拟器检查输入域的功能

在运行模拟器中创建变量 “PresetLevel”，在 “Start” 列激活监视功能 (复选框打钩)。点击画面上的输入域，出现一个数字键盘 (见图 8-43)。其中的 “ESC” 是取消键，点击它后数字键盘消失，退出输入过程，输入的数字无效。“BSP” 是退格键，与计算机键盘上的〈Backspace〉键的功能相同，点击该键，将删除光标左侧的数字。“+/-” 键用于改变输入的数字的符号。 和 分别是光标左移键和光标右移键， 是确认 (回车) 键，点击它使输入的数字有效 (被确认)，将在输入域中显示，并退出键盘。

用弹出的小键盘输入数据 80，在运行模拟器的当前值列出现同样的数据。

#### 8. 用运行模拟器检查输出域的功能

在运行模拟器中创建变量 “ActualLevel”，在 “Set value” 列输入常数 76，按〈Enter〉键确认，可以看到画面上的输出域显示出相同的数值 (见图 8-41)。

#### 9. 画面的切换

点击画面下部的画面切换按钮，可以在两个画面之间相互切换。用运行模拟器修改变量 “ActualLevel”的值，画面 “Level” 中棒图显示的“液位”的高度随之而变。

### 8.3.2 用 HMI 的控制面板设置 HMI 的参数

#### 1. 启动 HMI 设备

接通 HMI 的 DC 24V 电源后，启动期间显示进度条。启动后将会出现装载 (Loader) 对话框 (见图 8-44)。装载对话框中的按钮功能如下：

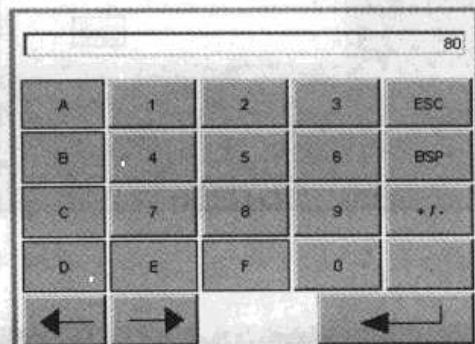


图 8-43 数字键盘

“Transfer”（传送）按钮用于将 HMI 设备切换到传送模式。  
“Start”（启动）按钮用于打开保存在 HMI 设备中的项目，显示初始画面。  
“Control Panel”按钮用于打开 HMI 设备的控制面板。  
如果 HMI 设备没有装载任何项目，在启动时将会自动切换到“Transfer”（传送）模式，出现“传送”对话框（见图 8-45）。点击“Cancel”（中止）按钮返回装载对话框。

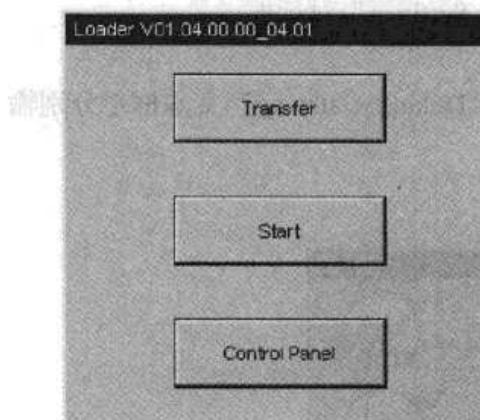


图 8-44 装载对话框

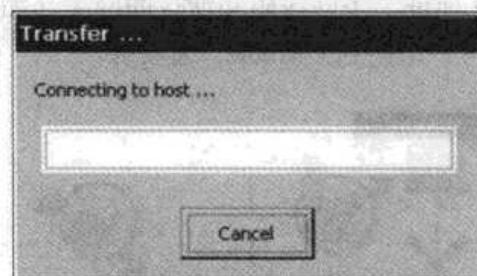


图 8-45 传送对话框

启动时如果 HMI 设备已经装载了项目，出现装载对话框后经过一定时间的延时，将自动启动项目。

## 2. 控制面板

精简系列面板使用 Windows CE 操作系统，与普通的 Windows 操作系统一样，用控制面板设置 HMI 设备的各种参数。点击图 8-44 的装载对话框中的“Control Panel”（控制面板）按钮，打开 HMI 设备的控制面板（见图 8-46）。

可以用控制面板进行下列设置：

- 1) 双击“Profinet”按钮，设置以太网接口的通信参数（见图 8-47）。
- 2) 双击“OP”按钮，设置显示方式为立式或横式、启动时的延迟时间，和对触摸屏进行校准。
- 3) 双击“Password”按钮，设置对控制面板的密码保护。
- 4) 双击“Transfer”按钮，选中激活通道（Enable Channel）和远程控制（Remote Control）复选框（复选框中出现×）。
- 5) 双击“Screensaver”按钮，设置屏幕保护程序的等待时间。输入为 0 将禁用屏幕保护。

如果画面内容显示的时间过长，切换画面后可能在背景中留下模糊的影像（虚像）。经过一段时间后，“虚像”会自动消失。相同的内容在画面中显示的时间越长，滞留的残影消失所需的时间就越长。屏幕保护程序有助于防止出现残影滞留。建议使用屏幕保护程序。

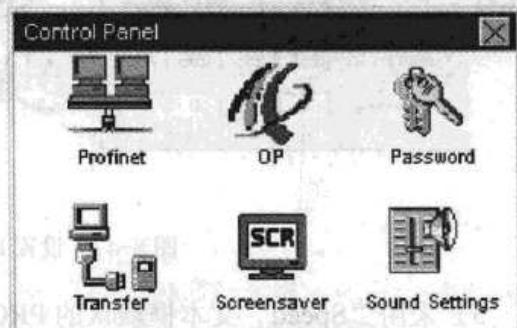


图 8-46 HMI 的控制面板

6) 双击“Sound Settings”按钮，设置是否在触摸屏幕或显示消息时产生声音反馈。

详细的情况请参阅随书光盘中的文件《精简系列面板操作说明》。

### 3. 设置以太网接口的通信参数

使用精简系列面板之前，应在它的控制面板中设置好它的通信参数，具体的操作步骤如下（图 8-47 给出了各步骤的编号）：

- 1) 双击控制面板的“Profinet”按钮，打开“Profinet Settings”对话框。
- 2) 用单选框选择由用户指定 IP 地址。
- 3) 用屏幕键盘在“IP Address”、“Subnet Mask”和“Default Gateway”文本框中分别输入 IP 地址、子网掩码和默认的网关（如果使用）。
- 4) 切换到“Mode”选项卡。

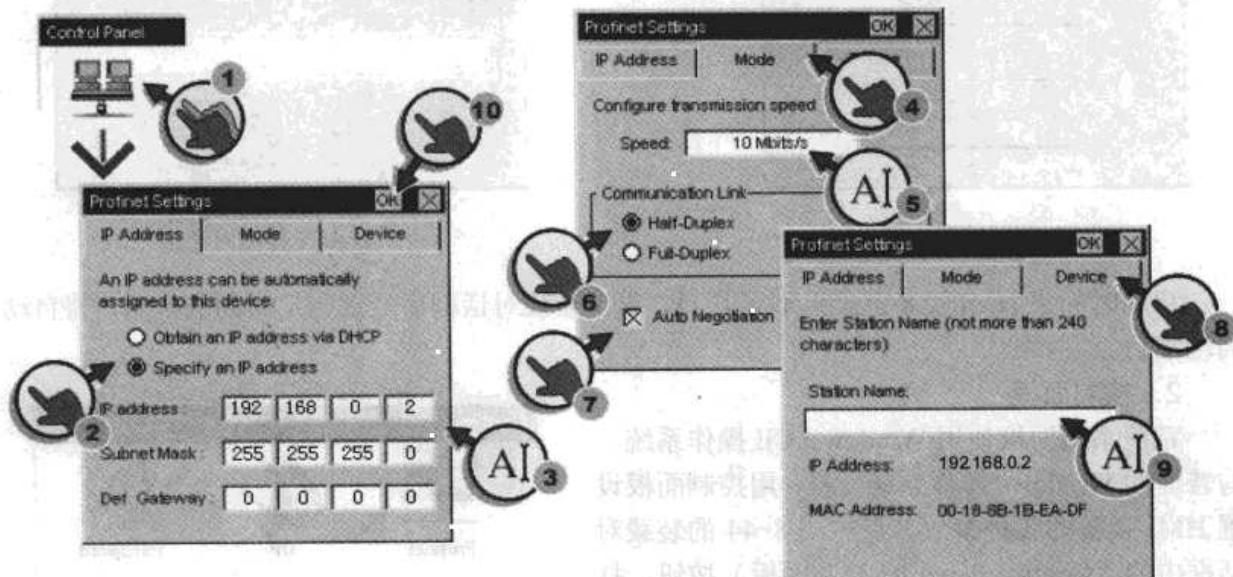


图 8-47 设置 HMI 的 PROFINET 接口参数

- 5) 采用“Speed”文本框默认的 PROFINET 网络的传输速率（10 Mbit/s）。
- 6) 用单选框选择连接模式为“半双工”或“全双工”。
- 7) 选中复选框“Auto Negotiation”后，激活以下功能：自动检测和设置 PROFINET 网络的连接模式和传输速率，激活“自动交叉”功能。
- 8) 切换到“Device”选项卡。
- 9) 为 HMI 设备输入站的名称。该名称最大长度 240 个字符，不要使用特殊字符。如果系统只有一块面板，可以使用默认的站名，不必修改它。

点击“OK”按钮退出设置，重新启动面板，使设置的参数有效。

### 8.3.3 HMI 组态信息的下载与运行

#### 1. 将组态信息下载到 HMI

为了实现计算机与 HMI 的通信，应做好下面的准备工作：

- 1) 组态 CPU 和 HMI, HMI 的组态信息是项目的一部分, 可以在项目内部组态和下载 HMI 的信息。
- 2) 生成 HMI 时, 建立 PLC 与 HMI 之间的逻辑网络连接(见图 8-5)。
- 3) 在 STEP 7 Basic 中设置 CPU 和 HMI 的 IP 地址分别为 192.168.0.1 和 192.168.0.2。设置计算机的子网地址为 192.168.0 (见图 3-20), IP 地址的第 4 个字节只要不与 CPU 和 HMI 的冲突就可以了。它们的子网掩码均为 255.255.255.0。
- 4) 用 HMI 的控制面板设置它的 IP 地址为 192.168.0.2, 子网掩码为 255.255.255.0。
- 5) 用以太网电缆连接好计算机与 HMI 的 RJ 45 通信接口。
- 6) 接通 HMI 的电源, 点击出现的装载对话框中的 Transfer 按钮, 打开传送对话框(见图 8-45), HMI 处于等待接收上位计算机(host)信息的状态。
- 7) 选中 STEP 7 Basic 的项目树中的 HMI\_1, 点击工具栏上的下载按钮 , 下载 HMI 的组态信息。出现下载预览对话框(见图 8-48 中的大图)后, 首先自动地对要下载的信息进行编译, 编译成功后, 显示“Ready for loading”(准备好下载), 选中“Overwrite all”(改写全部)复选框。点击“Load”按钮, 开始下载。如果是第一次下载新的 HMI, 出现需要更新操作系统的对话框(见图 8-48 中的小图), 点击其中的“Yes”按钮, 首先下载更新操作系统的文件。更新成功后, 自动重新启动 HMI, 启动成功后, 自动下载 HMI 的组态信息。

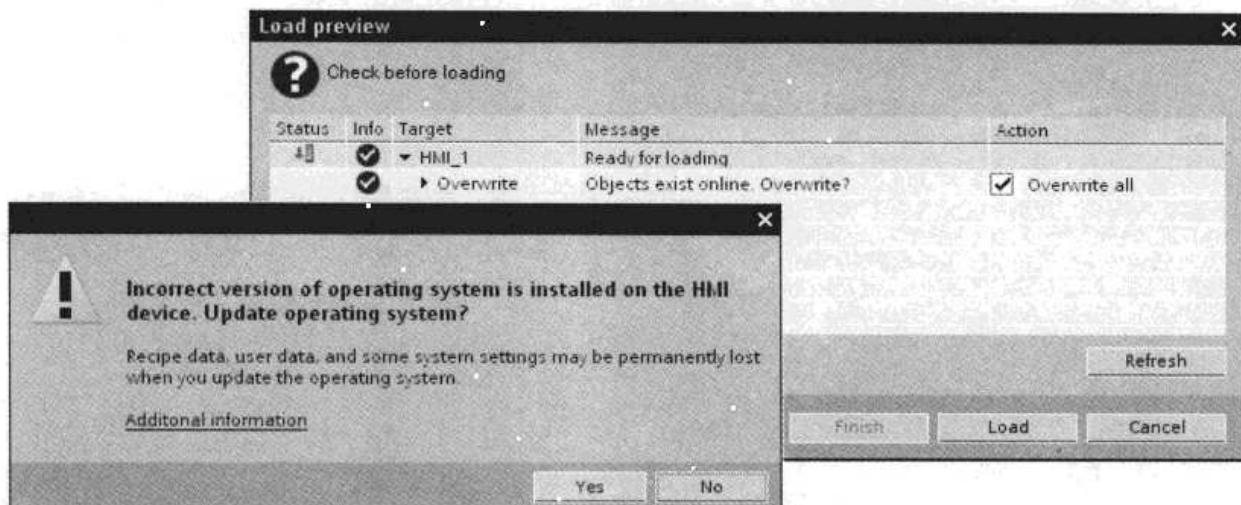


图 8-48 下载预览对话框与更新操作系统的对话框

下载结束后, HMI 自动打开初始画面, 点击图 8-48 中的“Finish”按钮, 结束下载过程。

HMI 中的项目正在运行时, 也可以下载项目文件, 首先自动关闭 HMI 设备的运行系统, 下载期间将出现图 8-45 中的传送对话框, 传送结束后自动显示初始画面。

## 2. 验证 PLC 和 HMI 的功能

首先分别下载 CPU 和 HMI 的用户程序和组态信息, 然后用以太网电缆连接 CPU 和 HMI 的以太网接口(见图 8-49)。两台设备通电后, 经过一定的时间, 面板显示初始画面。用手指点击初始画面上的“Start”和“Stop”按钮, 通过 PLC 中的程序, 来控制 PLC 中的变量 Q0.0。控制的结果用画面上的指示灯显示出来(见图 8-50)。

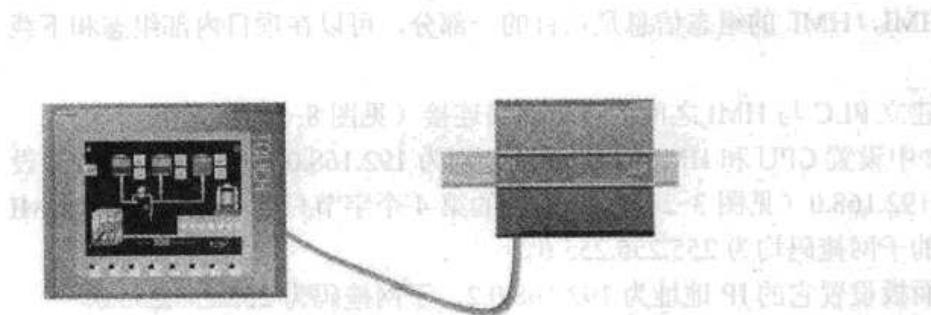


图 8-49 HMI 与 PLC 的通信连接

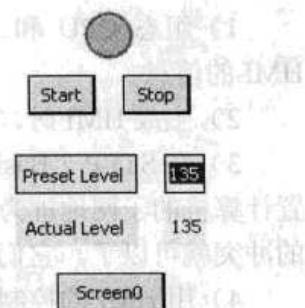


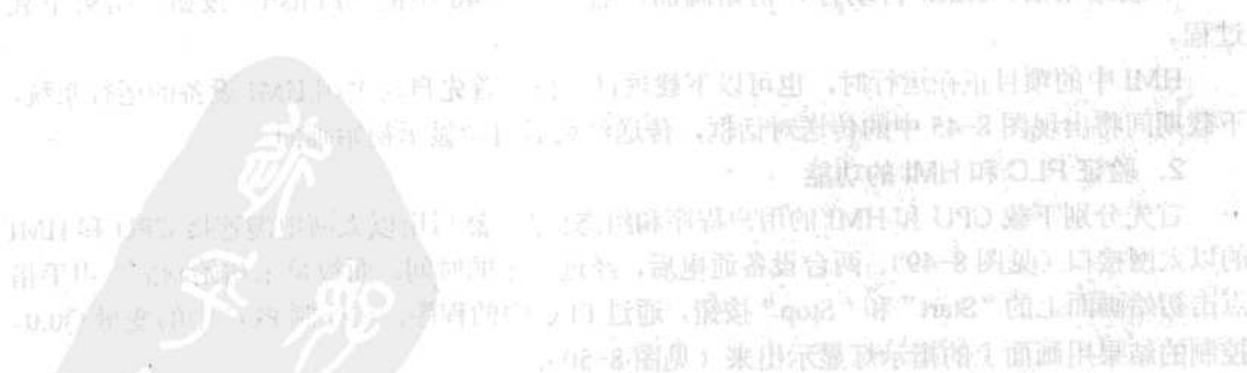
图 8-50 运行时的初始画面

点击画面上的文本域 Preset Level 右边的输入域，用弹出的小键盘输入数据，由于 PLC 梯形图中 MOVE 指令的作用（见图 8-12），输入的数据被传送到变量 ActualLevl，并用画面上 Actual Level 文本域右边的输出域显示出来，因此输入域和输出域显示的数值相同。



图 8-51 屏幕截图显示了 HMI 上的梯形图编辑器

图 8-52 展示了“Preset”软元件的梯形图。该梯形图由梯形图块 M001 和常开触点 X0 组成。



# 第9章 PLC应用中的其他问题

## 9.1 PLC控制系统的应用与设计

### 9.1.1 系统设计

PLC控制系统的应用与设计如图9-1所示。

#### 1. 深入了解被控系统

这一步是系统设计的基础，设计前应熟悉图样资料，深入调查研究，与工艺、机械方面的技术人员和现场操作人员密切配合，共同讨论，解决设计中出现的问题。应详细了解被控对象的全部功能，例如机械部件的动作顺序、动作条件、必要的保护与连锁，系统要求哪些工作方式（例如手动、自动、半自动等），设备内部机械、液压、气动、仪表、电气几大系统之间的关系，PLC与其他智能设备（例如别的PLC、计算机、变频器、工业电视、机器人）之间的关系，PLC是否需要通信联网，需要显示哪些数据及显示的方式等，电源突然停电及紧急情况的处理，以及安全电路的设计。有时需要设置PLC之外的手动的或机电的联锁装置来防止危险的操作。

对于大型复杂的控制系统，需要考虑将系统分解为几个独立的部分，各部分分别用单独的PLC或其他控制装置来控制，并考虑它们之间的通信方式。

#### 2. 人机界面的选择

人机界面（HMI）用于操作人员与PLC之间的信息交换。人机界面一般特指用于操作人员与控制系统之间进行对话和相互作用的专用设备。使用单台PLC的小型数字量控制系统一般用指示灯、报警器、按钮和操作开关来作人机界面。

如果系统需要输入和显示的参数较多，或者系统对人机界面功能的要求较高，可以使用专门为S7-1200配套的精简系列面板，根据系统对人机界面功能和成本的要求，选用某种型号的精简系列面板。

#### 3. 通信方式的选择

选择通信方式时，应考虑通信网络允许的最大节点数、最大通信距离和通信接口是否需要光隔离等问题。选择通信速率时应考虑网络中单位时间内可能的最大信息流量，并应留有一定的余地。通信速率与通信线路的长度有关，通信距离增大时最大通信速率降低。

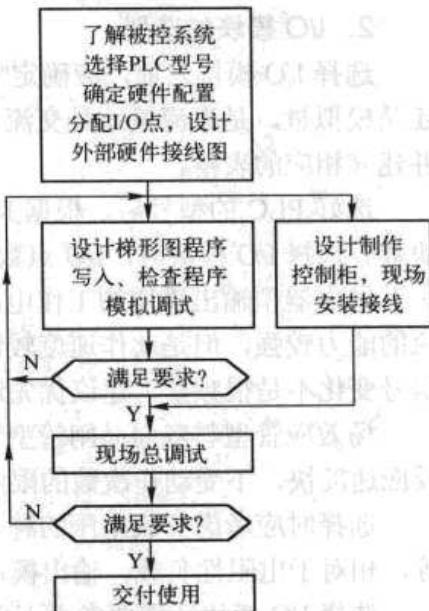


图9-1 设计调试过程示意图

可以用 S7-1200 CPU 集成的以太网接口实现 CPU 与计算机、HMI 和其他 CPU 的通信。如果需要与其他设备用串行通信接口进行点对点的通信，可以选用适当的通信模块。配备通信模块后，可以使用 Modbus 协议与其他设备进行通信，还可以使用 USS 协议与西门子的变频器进行通信。

### 9.1.2 PLC 硬件的选型

#### 1. CPU 型号的选择

S7-1200 不同的 CPU 模块的性能有较大的差别，在选择 CPU 模块时，应考虑 CPU 集成的 I/O 点数和模块的扩展能力，程序存储器与数据存储器的容量，CPU 集成的高速计数器、高速输出和中断的功能，在满足要求的前提下尽量降低硬件成本。

#### 2. I/O 模块的选型

选择 I/O 模块之前，应确定哪些信号需要输入给 PLC，哪些负载由 PLC 驱动，是数字量还是模拟量，是直流量还是交流量，以及电压的等级；是否有特殊要求，例如快速响应等，并建立相应的表格。

选好 PLC 的型号后，根据 I/O 表和可供选择的 I/O 模块的类型，确定 I/O 模块的型号和块数。选择 I/O 模块时，I/O 点数一般应留有一定的裕量，以备今后系统改进或扩充时使用。

继电器型输出模块的工作电压范围广，触点的导通压降小，承受瞬时过电压和瞬时过电流的能力较强，但是动作速度较慢，触点寿命（动作次数）有一定的限制。如果系统的输出信号变化不是很频繁，建议优先选用继电器型的。

场效应管型与双向晶闸管型输出模块分别用于直流负载和交流负载，它们的可靠性高，反应速度快，不受动作次数的限制，但是过载能力稍差。

选择时应考虑负载电压的种类和大小、系统对延迟时间的要求、负载状态变化是否频繁等，相对于电阻性负载，输出模块驱动电感性负载和白炽灯时的负载能力降低。

选择 I/O 模块还需要考虑下面的问题：

1) 外部传感器或电子设备（例如变频器）的输出电路的类型应与输入模块的输入电路匹配，使二者的输入/输出端能直接相连。

2) 选择模拟量模块时应考虑变送器和执行机构的量程是否能与 PLC 的模拟量输入/输出模块的量程匹配。模拟量模块的 A/D、D/A 转换器的位数反映了模块的分辨率，模拟量模块的转换时间反映了模块的工作速度。

3) 使用旋转编码器时，应考虑 PLC 的高速计数器的功能和工作频率是否能满足要求。

### 9.1.3 硬件软件设计与调试

#### 1. 系统硬件设计

1) 确定系统的总体结构，例如是否需要通信和联网，选用什么样的通信网络和通信协议，需要什么通信接口或通信模块，在此基础上确定 PLC 的型号，选择 CPU 模块和与通信有关的硬件。根据系统的要求确定是否需要人机界面和具体的型号。

2) 根据数字量、模拟量输入/输出的性质和点数确定 I/O 模块的种类和块数。

3) 给各输入、输出变量分配地址。梯形图中 I/O 变量的地址与 PLC 的外部接线端子号直接关联，这一步为绘制硬件接线图作好了准备，也为梯形图的设计作好了准备。

4) 画出 PLC 的外部硬件接线图, 以及其他电气原理图和接线图。

5) 画出操作站和控制柜面板的机械布置图和内部的机械安装图。

## 2. 软件设计

软件设计包括设计主程序、子程序、中断程序、故障应急措施和辅助程序等, 小型数字量控制系统一般只有主程序。

首先根据总体要求和控制系统的具体情况, 确定用户程序的基本结构, 画出程序的流程图或数字量控制系统的顺序功能图。它们是编程的主要依据, 应尽可能地准确和详细。

较简单的系统的梯形图可以用经验法设计, 复杂的系统一般用顺序控制设计法设计。画出系统的顺序功能图后, 根据它设计出梯形图程序。可以在 PLC 变量表、代码块的接口区和共享数据块中给变量命名。变量名称可以在程序中显示, 便于程序的阅读和调试, 变量名称的定义应简短和易于理解。

## 3. 软件的模拟调试

设计好用户程序后, 一般先作模拟调试。用 PLC 的硬件来调试程序时, 用接在输入端的小开关或按钮来模拟 PLC 实际的输入信号, 例如用它们发出操作指令。或者在适当的时候用它们来模拟实际的反馈信号, 例如限位开关触点的接通和断开。通过输出模块上各输出点对应的发光二极管, 观察输出信号是否满足设计的要求。

调试顺序控制程序的主要任务是检查程序的运行是否符合顺序功能图的规定, 即在转换实现的两个条件都满足时, 该转换所有的前级步是否变为不活动步, 所有的后续步是否变为活动步, 以及各步被驱动的负载是否发生相应的变化。

在调试时应充分考虑各种可能的情况, 对系统各种不同的工作方式、顺序功能图中的每一条支路、各种可能的进展路线, 都应逐一检查, 不能遗漏。发现问题后及时修改程序, 直到在各种可能的情况下输入信号与输出信号之间的关系完全符合要求。

如果程序中某些定时器或计数器的设定值过大, 为了缩短调试时间, 可以在调试时将它们减小, 模拟调试结束后再写入它们的实际设定值。

在编程软件中, 可以用程序状态监视功能或状态表来监视程序的运行。

## 4. 硬件的调试与系统调试

在对程序进行模拟调试的同时, 可以设计、制作控制屏, PLC 之外其他硬件的安装、接线工作也可以同时进行。完成硬件的安装和接线后, 应对硬件的功能进行检查, 观察各输入点的状态变化是否能送给 PLC。在 STOP 模式用监视表改变物理输出的状态, 可以检查 PLC 的负载 (例如外部的电磁阀和接触器) 的接线是否正确。

完成上述的调试后, 将 PLC 置于 RUN 状态, 运行用户程序, 检查控制系统是否能满足要求。在调试过程中将暴露出系统中可能存在的硬件问题, 以及程序设计中的问题, 发现问题后在现场加以解决, 直到完全符合要求。

## 5. 整理技术文件

根据调试的最终结果整理出完整的技术文件, 至少应将 PLC 的外部接线图提供给用户, 以便于系统的维护与改进。技术文件应包括:

1) PLC 的外部接线图和其他电气图样。

2) PLC 的编程元件表, 包括定时器、计数器的设定值等。

3) 带注释的程序和必要的总体文字说明。

## 9.2 PLC 控制系统的可靠性措施

PLC 是专门为工业环境设计的控制装置，一般不需要采取什么特殊措施，就可以直接在工业环境使用。但是如果环境过于恶劣，电磁干扰特别强烈，或者安装使用不当，都不能保证系统的正常安全运行。干扰可能使 PLC 接收到错误的信号，造成误动作，或使 PLC 内部的数据丢失，严重时甚至会使系统失控。在系统设计时，应采取相应的可靠性措施，以消除或减少干扰的影响，保证系统的正常运行。

干扰主要来自控制系统供电电源的波动和电源电压中的高次谐波，以及因为线路和设备之间的分布电容和分布电感产生的电磁感应。

实践表明，系统中 PLC 之外的部分（特别是机械限位开关和某些执行机构）的故障率，往往比 PLC 本身的故障率高得多，因此在设计时应采取相应的措施，例如用可靠性高的接近开关代替机械限位开关，才能保证整个系统的可靠性。

### 1. 电源的抗干扰措施

电源是干扰进入 PLC 的主要途径之一，电源干扰主要是通过供电线路的阻抗耦合产生的，各种大功率用电设备是主要的干扰源。

在干扰较强或对可靠性要求很高的场合，可以在 PLC 的交流电源输入端加接带屏蔽层的隔离变压器和低通滤波器。可以在互联网上搜索和选用电源滤波器或净化电源产品。

隔离变压器可以抑制从电源线窜入的外来干扰，提高抗高频共模干扰能力。高频干扰信号不是通过变压器绕组的耦合，而是通过初级、次级绕组间的分布电容传递的。在初级、次级绕组之间加绕屏蔽层，并将它和铁心一起接地，可以减少绕组间的分布电容，提高抗高频干扰的能力。

### 2. 布线的抗干扰措施

数字量信号传输距离较远时，可以选用屏蔽电缆。模拟信号和高速信号（例如旋转编码器提供的信号）应选择屏蔽电缆，通信电缆应按规定选型。

PLC 应远离强干扰源，例如大功率晶闸管装置、变频器、高频焊机和大型动力设备等。PLC 不能与高压电器安装在同一个开关柜内，在柜内 PLC 应远离动力线（二者之间的距离应大于 200mm）。系统的动力线应足够粗，以降低大容量异步电动机起动时的线路压降。不同类型的导线应分别装入不同的电缆管或电缆槽中，并使其有尽可能大的空间距离。

I/O 线与电源线应分开走线，并保持一定的距离。如果不得已要在同一线槽中布线，应使用屏蔽电缆。交流线与直流线应分别使用不同的电缆，数字量、模拟量 I/O 线应分开敷设，后者应采用屏蔽线。当数字量输入、输出线不能与动力线分开布线，且线路较长时，可以用继电器来隔离输入/输出线上的干扰。

传送模拟信号的屏蔽线的屏蔽层应一端接地，为了泄放高频干扰，数字信号线的屏蔽层应并联电位均衡线，其电阻应小于屏蔽层电阻的 1/10，并将屏蔽层两端接地。如果无法设置电位均衡线，或只考虑抑制低频干扰时，也可以一端接地。

如果模拟量输入/输出信号距离 PLC 较远，应采用 4~20mA 的电流传输方式，而不是易受干扰的电压传输方式。干扰较强的环境应选用有光隔离的模拟量 I/O 模块，使用

分布电容小、干扰抑制能力强的配电器为变送器供电，以减少对 PLC 的模拟量输入信号的干扰。

### 3. PLC 的接地

良好的接地是 PLC 安全可靠运行的重要条件，PLC 与强电设备最好分别使用不同的接地装置，接地线的截面积应大于  $2\text{mm}^2$ ，接地点与 PLC 的距离应小于 50m。

在发电厂或变电站等场合，有接地网络可供使用。各控制屏和自动化元件可能相距甚远，若分别将它们在就近的接地母线上接地，强电设备的接地电流可能在两个接地点之间产生较大的电位差，干扰控制系统的工作。为防止不同信号回路接地线上的电流引起交叉干扰，必须分系统（例如以控制屏为单位）将弱电信号的内部地线接通，然后各自用规定面积的导线统一引到接地网络的某一点，实现控制系统一点接地的要求。

### 4. 防止变频器干扰的措施

现在 PLC 越来越多地与变频器一起使用，经常会遇到变频器干扰 PLC 的正常运行，变频器已经成为 PLC 最常见的干扰源。

变频器的主电路为交-直-交变换电路，工频电源被整流成直流电压信号，输出的是基波频率可变的高频脉冲信号，载波频率可能高达数千赫兹。变频器的输入电流为含有丰富的高次谐波的脉冲波，它会通过电力线干扰其他设备。高次谐波电流还通过电缆向空间辐射，干扰邻近的电气设备。

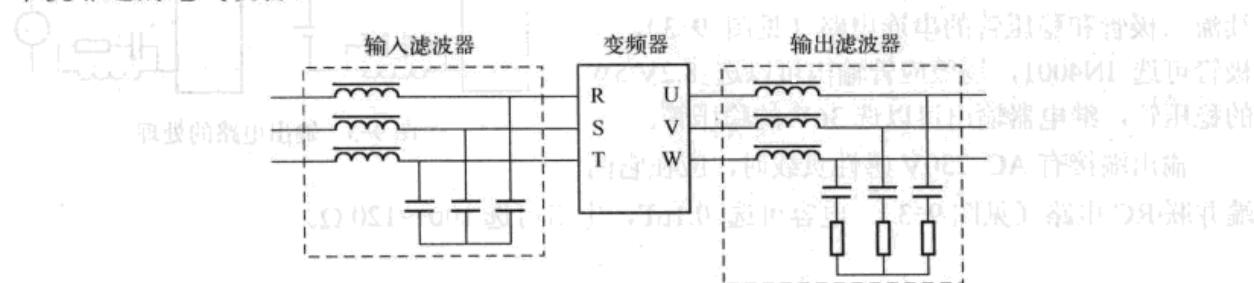


图 9-2 变频器的输入/输出滤波器

可以在变频器输入侧与输出侧串接电抗器，或安装谐波滤波器（见图 9-2），以吸收谐波，抑制高频谐波电流。

将变频器放在控制柜内，并将其金属外壳接地，对高频谐波有屏蔽作用。变频器的输入、输出电流（特别是输出电流）中含有丰富的谐波，所以主电路也是辐射源。PLC 的信号线和变频器的输出线应分别穿管敷设，变频器的输出线一定要使用屏蔽电缆或穿钢管敷设，以避免辐射干扰和感应干扰。

变频器应使用专用接地线，且用粗短线接地，其他邻近的电气设备的接地线必须与变频器的接地线分开。

对受干扰的 PLC 可以采用屏蔽措施，尽量远离电磁辐射干扰区，可以在 PLC 的电源输入端串入滤波电路或安装隔离变压器，以减小谐波电流的影响。

### 5. 强烈干扰环境中的隔离措施

一般情况下，PLC 的输入/输出信号采用内部的隔离措施即可以保证系统的正常运行。因此一般没有必要在 PLC 外部再设置干扰隔离器件。

在发电厂等工业环境，空间极强的电磁场和高电压、大电流断路器的通断将会对 PLC 产

生强烈的干扰。由于现场条件的限制，有时很长的强电电缆和 PLC 的低压控制电缆只能敷设在同一电缆沟内，强电干扰在输入线上产生的感应电压和电流相当大，可能使 PLC 输入端的光耦合器中的发光二极管发光，使 PLC 产生误动作。在这种情况下，对于用长线引入 PLC 的数字量输入信号，可以用小型继电器来隔离。S7-1200 的输入模块允许的最大逻辑 0 信号电流仅 1mA，而小型继电器的线圈吸合电流为数十毫安，强电干扰信号通过电磁感应产生的能量一般不会使隔离用的继电器误动作。来自开关柜内和距离开关柜不远的输入信号一般没有必要用继电器来隔离。

为了提高抗干扰能力，对长距离的 PLC 的外部信号、PLC 和计算机之间的串行通信信号，可以考虑用光纤来传输和隔离，或使用带光耦合器的通信接口。

## 6. PLC 输出的可靠性措施

如果用 PLC 驱动交流接触器，应将额定电压为 AC 380V 的交流接触器的线圈换成 220V 的。在负载要求的输出功率超过 PLC 的允许值时，应设置外部继电器。PLC 输出模块内的小型继电器的触点小，断弧能力差，不能直接用于 DC 220V 的电路，必须通过外部继电器驱动 DC 220V 的负载。

与 PLC 装在同一个开关柜内的电感性元件，例如继电器、接触器的线圈，应并联 RC 消弧电路。

输出端接有直流感性负载时，应在它两端并联续流二极管和稳压管的串连电路（见图 9-3），二极管可选 IN4001，场效应管输出可以选 8.2V/5W 的稳压管，继电器输出可以选 36V 的稳压管。

输出端接有 AC 230V 感性负载时，应在它两端并联 RC 电路（见图 9-3），电容可选 0.1μF，电阻可选 100~120 Ω。

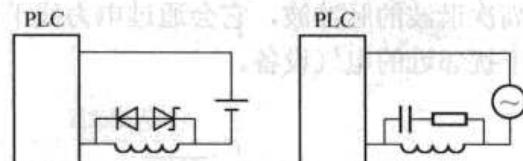


图 9-3 输出电路的处理

## 9.3 PLC 在模拟量闭环控制中的应用

### 9.3.1 模拟量闭环控制系统与 PID 控制器

#### 1. 模拟量闭环控制系统的组成

PLC 模拟量闭环控制系统如图 9-4 所示，虚线部分在 PLC 内。在模拟量闭环控制系统中，被控量  $c(t)$ （即系统的输出量，例如压力、温度、流量、转速等）是连续变化的模拟量，大多数执行机构（例如直流调速装置、电动调节阀和变频器等）要求 PLC 输出模拟信号  $M(t)$ ，而 PLC 的 CPU 只能处理数字量。 $c(t)$  首先被测量元件和变送器转换为标准量程（例如 DC 4~20mA 和 0~10V）的直流电流信号或直流电压信号  $pv(t)$ 。

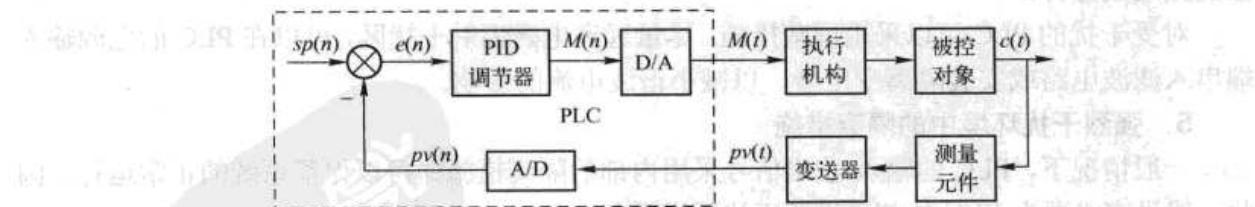


图 9-4 闭环控制系统的方框图

以加热炉温度闭环控制系统为例,用热电偶检测炉温  $c(t)$ , 温度变送器将热电偶输出的微弱的电压信号转换为标准量程的电流或电压  $pV(t)$ , 然后送给 PLC 的模拟量输入模块, 经 A/D 转换后得到与温度成比例的数字量  $pV(n)$ , CPU 将它与温度设定值  $sp(n)$  比较, 并按某种控制规律(例如 PID 控制算法)对误差值  $e(n)$  进行运算, 将运算结果  $M(n)$ (数字量) 送给模拟量输出模块, 经 D/A 转换后变为标准量程的电流信号或电压信号  $M(t)$ , 用来控制电动调节阀的开度, 通过它控制加热用的天然气的流量, 实现对温度的闭环控制。 $c(t)$  为系统的输出量, 即被控量, 例如加热炉中的温度。

模拟量控制系统分为恒值控制系统和随动系统。恒值控制系统的给定值由操作人员提供, 一般很少变化, 例如温度控制系统, 转速控制系统等。随动系统的输入量是不断变化的随机变量, 例如电动调节阀的开度控制系统就是典型的随动系统。闭环负反馈控制可以使控制系统的反馈量  $pV(n)$  等于或跟随给定值  $sp(n)$ 。以炉温控制系统为例, 假设输出的温度值  $c(t)$  低于给定的温度值, 反馈量  $pV(n)$  小于给定值  $sp(n)$ , 误差  $e(n)$  为正, 控制器的输出量  $M(t)$  将增大, 使执行机构(电动调节阀)的开度增大, 进入加热炉的天然气流量增加, 加热炉的温度升高, 最终使实际温度接近或等于温度给定值。

天然气压力的波动、工件进入加热炉, 这些因素称为扰动量, 它们会破坏炉温的稳定。闭环控制可以有效地抑制闭环中各种扰动的影响, 使被控量趋近于给定值。

闭环控制系统的结构简单, 容易实现自动控制, 因此在各个领域得到了广泛的应用。

## 2. 变送器的选择

变送器分为电流输出型和电压输出型, 电压输出型变送器具有恒压源的性质, PLC 模拟量输入模块的电压输入端的输入阻抗很高, 例如  $100k\Omega \sim 10M\Omega$ 。如果变送器距离 PLC 较远, 通过线路间的分布电容和分布电感产生的干扰电流在模块的输入阻抗上将产生较高的干扰电压。例如  $1\mu A$  干扰电流在  $10M\Omega$  输入阻抗上将产生  $10V$  的干扰电压信号, 所以远程传送模拟量电压信号时抗干扰能力很差。

电流输出具有恒流源的性质, 恒流源的内阻很大。PLC 的模拟量输入模块输入电流时, 输入阻抗较小(例如  $250\Omega$ )。线路上的干扰信号在模块的输入阻抗上产生的干扰电压很低, 所以模拟量电流信号适于远程传送。

变送器分为二线制和四线制两种, 四线制变送器有两根信号线和两根电源线。二线制变送器只有两根外部接线(见图 9-5), 它们既是电源线, 也是信号线, 输出  $4\sim 20mA$  的信号电流, 直流  $24V$  电源串接在回路中, 二线制变送器一般用配电器供电。通过调试, 在被检测信号量程的下限时输出电流为  $4mA$ , 被检测信号满量程时输出电流为  $20mA$ 。二线制变送器的接线少, 信号可以远传, 在工业中得到了广泛的应用。

## 3. 闭环控制反馈极性的确定

闭环控制必须保证系统是负反馈(误差 = 给定值 - 反馈值), 而不是正反馈(误差 = 给定值 + 反馈值)。如果系统接成了正反馈, 将会失控, 被控量会往单一方向增大或减小, 给系统的安全带来极大的威胁。

闭环控制系统的反馈极性与很多因素有关, 例如因为接线改变了变送器输出电流或输出电压的极性, 或改变了位移传感器(编码器)的安装方向, 都会改变反馈的极性。

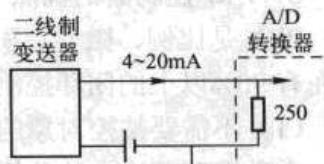


图 9-5 二线制变送器

可以用下述方法来判断反馈的极性：在调试时断开 D/A 转换器与执行机构之间的连线，在开环状态下运行 PID 控制程序。控制器中有积分环节，因为反馈被断开了，不能消除误差，D/A 转换器的输出电压会向一个方向变化。这时如果假设接上执行机构，能减小误差，则为负反馈，反之为正反馈。

以温度控制系统为例，假设开环运行时给定值大于反馈值，D/A 转换器的输出值不断增大，如果形成闭环，将使电动调节阀的开度增大，温度升高，闭环后温度反馈值将会增大，使误差减小，由此可以判定系统是负反馈。

#### 4. 闭环控制系统主要的性能指标

由于给定输入信号或扰动输入信号的变化，使系统的输出量发生变化，在系统输出量达到稳态值之前的过程称为过渡过程或动态过程。系统的动态过程的性能指标用阶跃响应的参数来描述（见图 9-6）。阶跃响应是指系统的输入信号阶跃变化（例如从 0 突变为某一恒定值）时系统的输出。

一个系统要正常工作，阶跃响应曲线应该是收敛的，最终能趋近于某一个稳态值  $c(\infty)$ 。系统进入并停留在  $c(\infty)$  上下  $\pm 5\%$ （或  $2\%$ ）的误差带内的这段时间  $t_s$  称为调节时间，到达调节时间表示过渡过程已基本结束。

系统的相对稳定性可以用超调量来表示。设动态过程中输出量的最大值为  $c_{\max}(t)$ ，如果它大于输出量的稳态值  $c(\infty)$ ，定义超调量

$$\sigma\% = \frac{c_{\max}(t) - c(\infty)}{c(\infty)} \times 100\%$$

超调量越小，动态稳定性越好。一般希望超调量小于  $10\%$ 。

通常用稳态误差来描述控制的准确性和控制精度，稳态误差是指响应进入稳态后，输出量的期望值与实际值之差。

#### 5. PID 控制器的优点

PID 是比例、微分、积分的英文缩写，PID 控制器是应用最广的闭环控制器，有人估计现在有  $90\%$  以上的闭环控制采用 PID 控制器。这是因为 PID 控制具有以下的优点：

##### （1）不需要被控对象的数学模型

自动控制理论中的分析和设计方法主要是建立在被控对象的线性定常数学模型的基础上的。这种模型忽略了实际系统中的非线性和时变性，与实际系统有较大的差距。对于许多工业控制对象，根本就无法建立较为准确的数学模型，因此自动控制理论中的设计方法很难用于大多数控制系统。对于这一类系统，使用 PID 控制可以得到比较满意的效果。

##### （2）结构简单，容易实现

PID 控制器的结构典型，程序设计简单，计算工作量较小，各参数有明确的物理意义，参数调整方便，容易实现多回路控制、串级控制等复杂的控制。

##### （3）有较强的灵活性和适应性

根据被控对象的具体情况，可以采用 PID 控制器的多种变种和改进的控制方式，例如 PI、PD、带死区的 PID、被控量微分 PID、积分分离 PID 和变速积分 PID 等，比例控制一般是必

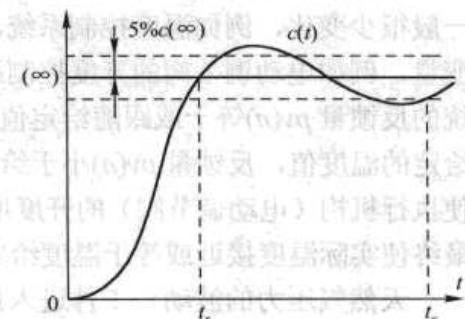


图 9-6 被控对象的阶跃响应曲线

不可少的。控制器的参数整定是 PID 控制器应用的难点，随着智能控制技术的发展，可以实现 PID 控制器的参数自整定，使 PID 控制器具有经久不衰的生命力。S7-1200 提供了 PID 参数自整定功能。

#### (4) 使用方便

现在已经有很多 PLC 厂家提供具有 PID 控制功能的产品，例如 PID 闭环控制模块、PID 控制指令和 PID 控制系统功能块等，它们使用起来非常方便，只需要设置一些参数即可。S7-1200 的 PID 指令和 STEP 7 Basic 的 PID 调试窗口的使用非常简单方便，形象直观。

### 6. PID 控制器的数字化

典型的 PID 模拟量闭环控制系统如图 9-4 所示。对于模拟量 PID 控制系统， $sp(t)$  是给定值， $pv(t)$  为过程变量（反馈量），误差  $e(t) = sp(t) - pv(t)$ ，模拟量 PID 控制器的输出

$$M(t) = K_C \left[ e(t) + \frac{1}{T_I} \int_0^t e(t) dt + T_D \frac{de(t)}{dt} \right] + M_{\text{initial}}$$

即控制器的输出 = 比例项 + 积分项 + 微分项 + 输出的初始值，式中  $M(t)$  是控制器的输出； $M_{\text{initial}}$  是 PID 控制器输出的初始值； $K_C$  是控制器的增益； $T_I$  和  $T_D$  分别是积分时间常数和微分时间常数。

比例（P）、积分（I）、微分（D）部分分别与误差、误差的积分和误差的微分成正比。如果取其中的一项或两项，可以组成 P、PD 或 PI 控制器。需要较好的动态品质和较高的稳态精度时，可以选用 PI 控制方式；控制对象的惯性滞后较大时，应选择 PID 控制方式。

PLC 周期性地执行 PID 控制程序，执行的周期称为采样周期  $T_S$ 。

### 9.3.2 PID\_Compact 指令与 PID 技术对象的组态

S7-1200 使用 PID\_Compact 指令来实现 PID 控制，该指令的背景数据块称为 PID\_Compact 技术对象。PID 控制器具有参数自整定功能和自动、手动模式。

PID 控制器连续地采集测量的被控制变量的实际值（简称为实际值或输入值），并与期望的设定值（Setpoint）比较。根据得到的系统误差，PID 控制器计算控制器的输出，使被控制变量尽可能快地接近设定值或进入稳态。

PID 控制器的输出值由以下 3 个分量组成：

- 1) 与系统误差成比例的比例分量。
- 2) 与系统误差的积分成比例的积分分量。
- 3) 与系统误差的变化率（微分）成比例的微分分量，系统误差减小时，微分部分为负值。

#### 1. 生成一个新的项目

打开 STEP 7 Basic 的项目视图，生成一个名为 PID 的新项目。双击项目树中的“Add new device”，添加一个 PLC 设备。CPU 的型号为 CPU 1214C。

#### 2. 调用 PID\_Compact 指令

调用 PID\_Compact 的时间间隔称为采样时间，为了保证精确的采样时间，用固定的时间间隔执行 PID 指令，在循环中断 OB 中调用 PID\_Compact 指令。

打开项目视图中的文件夹“\PLC\_1\Program block”，双击其中的“Add new block”，点击打开的对话框中的“Organization block”（组织块）按钮，选中“Cyclic interrupt”，生成循环中断组织块 OB200，设置适当的循环时间间隔。点击“OK”按钮，自动生成并打开 OB200。

选中 OB 下面的监视窗口左边的“Cyclic interrupt”组（见图 6-27），可以用右边的“Scan time”文本框来修改循环时间间隔。

打开任务卡的“Extended instruction”窗口的 PID 文件夹中，将其中的“PID\_Compact”指令拖放到 OB200 中，对话框“Call options”被打开。将默认的背景数据块的名称改为 PID\_DB，点击“OK”按钮，在“Program block”文件夹中生成名为“PID\_Compact”的功能块 FB1130（见图 9-7）。生成的背景数据块 DB1 在文件夹“Technology objects”（技术对象）中。

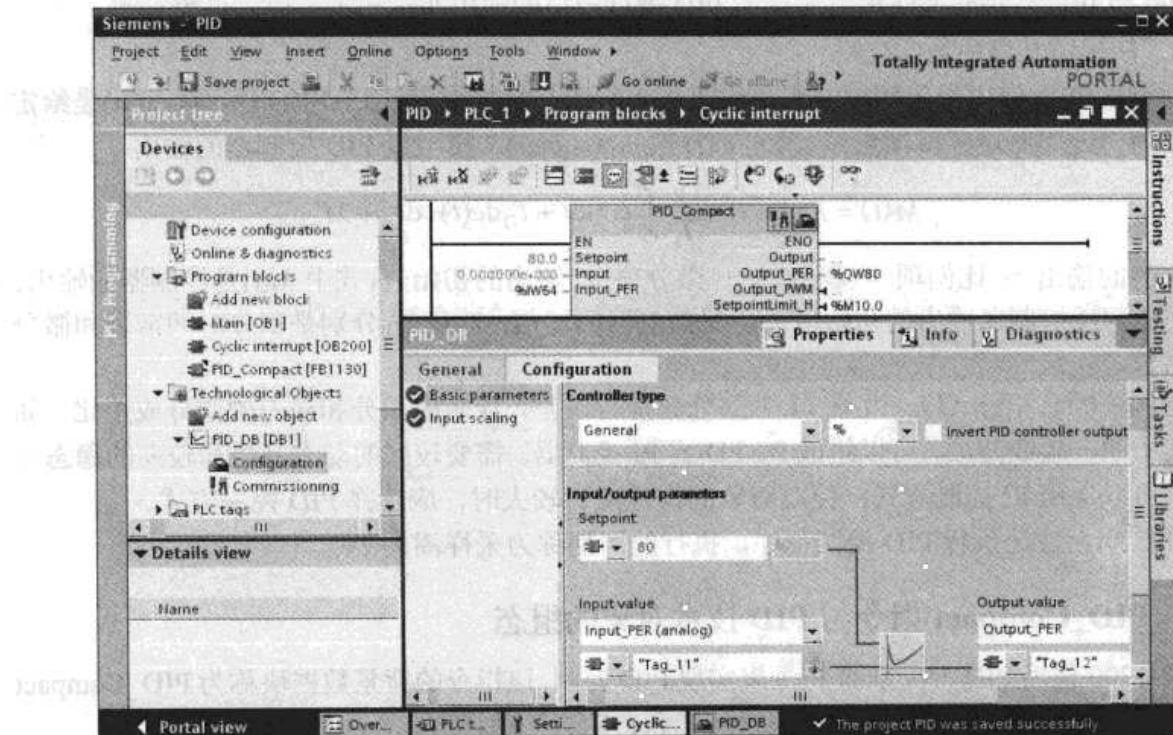


图 9-7 组态 PID 控制器的基本参数

### 3. PID 指令的模式

#### (1) Inactive 模式

PID Compact 技术对象被组态后首次下载到 CPU 之后，PID 控制器处于 Inactive（非活动的）运行模式，此时需要在调试窗口进行自整定初始启动。在运行时出现错误，或者点击了调试窗口的“STOP measurement”按钮（见图 9-13），PID 控制器将进入 Inactive 模式。选择其他运行模式时，活动状态的错误被确认。

#### (2) 自整定模式

打开 PID 调试窗口，可以选择进入初始启动自整定模式或运行点自整定模式。

#### (3) 自动模式

在自动模式，PID\_Compact 技术对象根据设置的 PID 参数进行闭环控制。

满足下列条件之一时，控制器将进入自动模式：

1) 成功地完成了自整定初始启动和运行点自整定的任务。

2) 在组态窗口选中了“Use manual PID parameters setting”复选框（见图 9-11）。

#### (4) 手动模式

在手动模式，PID 控制的输出变量用手动设置。

满足下列条件之一，控制器将进入手动模式：

- 1) 指令的输入参数“ManualEnable”为 1 状态。
- 2) 在调试窗口选中了“Manual”(手动)复选框。

#### 4. 组态基本参数

打开 OB200，选中“PID\_Compact”指令（见图 9-7），然后选中监视窗口左边的“Basic parameters”，在右边窗口中设置 PID 的基本参数。

##### (1) 控制器类型 (Controller type)

默认值为“General”(普通)，设定值与实际值的单位为%。可以用下拉式列表选择控制器类型为控制具体的物理量，例如转速、压力、流量等。被控量的单位随之而变。

##### (2) 反向调节

有些控制系统需要反向调节，例如在冷却系统中，增大阀门开度时降低液位，以增大制冷作用来降低温度。为此选中“Invert PID controller output”复选框，在 PID 控制器的输出值增大时，减小被控制的实际值。

##### (3) 控制器的输入/输出参数

控制器的输入/输出参数分别为设定值(Setpoint)、输入值(Input value, 即被控制的变量的反馈值)和输出值(Output value)。可以用各数值左边的按钮选择数值来自功能块(function block)或来自背景数据块(instance data block)。用输入值上面的下拉式列表选择输入值为来自用户程序的“Input”或“Input\_PER(analog)”(模拟量外设输入，即直接指定模拟量输入的地址)，用输出值上面的下拉式列表选择输出值为来自用户程序的“Output”、“Output\_PWM”(脉冲宽度调制的数字量开关输出)或“Output\_PER”(外设输出，即直接指定模拟量输出的地址)。可以用下拉式列表设置参数，也可以直接输入参数的绝对地址或符号地址。

图 9-7 中的“Tag\_11”和“Tag\_12”分别是 IW64(CPU 集成的模拟量输入通道 0)和 QW80(信号板的模拟量输出)。

#### 5. 组态输入值缩放比例

选中图 9-8 的监视窗口左边的“Input scaling”，可以缩放输入值，或给输入值设置偏移量。默认的比例为模拟量的实际值(或来自用户程序的输入值)为 0.0%~100.0%时，A/D 转换后的数字为 0.0~27648.0(见图 9-8)，可以修改这些参数。

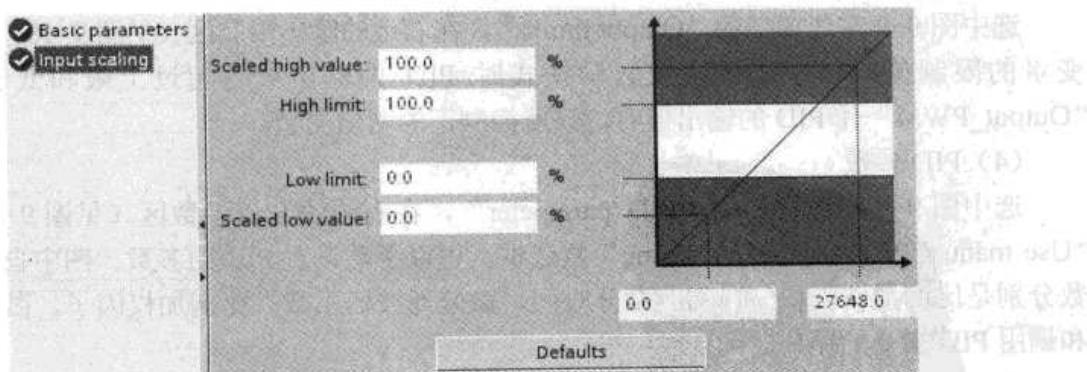


图 9-8 组态输入值比例

可以设置输入的上限值 (High limit) 和下限值 (Low limit)。在运行时一旦超过上限值或低于下限值，停止正常的控制，输出值被设置为 0。点击“Default”按钮，用默认值替换现有的值。

## 6. 组态控制器的高级参数

为了设置 PID 的高级参数，打开项目树中的文件夹“\PLC\_1\Technological Objects\PID\_DB”，双击其中的“Configuration”（见图 9-7），打开 PID\_Compact 技术对象。点击图 9-7 中 PID\_Compact 指令右上角的图标，也可以打开 PID\_Compact 技术对象（见图 9-9）。

选中左边窗口的“Advanced settings”，在右边窗口设置高级参数。

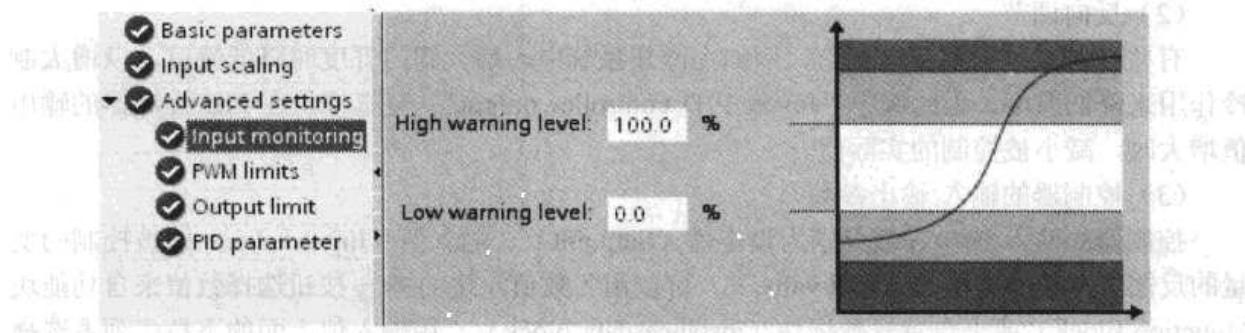


图 9-9 组态 PID 控制器的高级参数

### (1) 输入监视

选中左边窗口的“Input monitoring”，在右边的输入监视区（见图 9-9），可以设置输入的上限报警值（High warning level）和下限报警值（Low warning level）。运行时如果输入值超过设置的上限值或低于下限值，指令的 Bool 输出参数“InputWarning\_H”或“InputWarning\_L”将变为 1 状态。

### (2) PWM 限制

选中图 9-9 左边窗口的“PWM limits”，在右边的 PWM 限制区，可以设置 PWM 允许为 ON 和 OFF 的最长时间。该设置影响指令的输出变量“Output\_PWM”。PWM 的开关输出受“PID\_Compact”指令的控制，与 CPU 集成的脉冲发生器无关。

### (3) 输出限制

选中图 9-9 左边窗口的“Output limits”，在右边的输出限制区（见图 9-10），设置输出变量的限制值，使手动模式或自动模式时 PID 的输出值不超过上限和低于下限。用“Output\_PWM”作 PID 的输出值时，只能控制正的输出变量。

### (4) PID 参数

选中图 9-9 左边窗口的“PID parameter”，在右边的 PID 参数区（见图 9-11），选中“Use manual PID parameters setting”复选框，可以手动设置 PID 的参数。图中自上而下的参数分别是比例增益、积分时间、微分时间、微分滤波器系数、比例加权因子、微分加权因子和调用 PID 指令的循环时间。

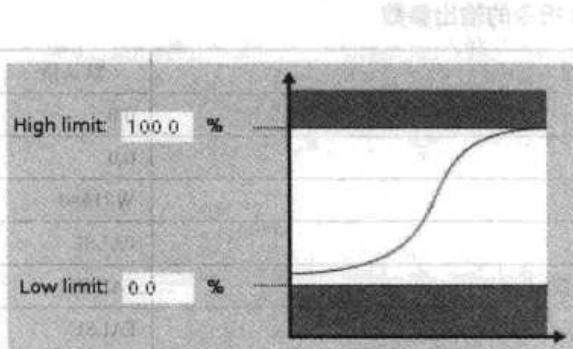


图 9-10 PID 控制器的输出限制

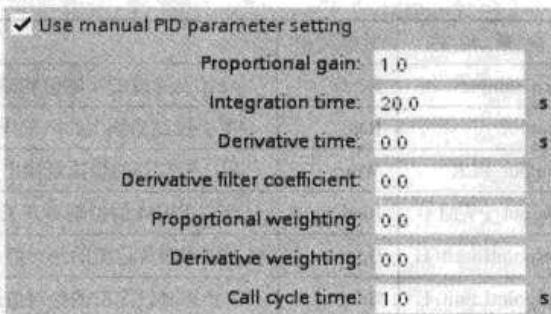


图 9-11 PID 参数设置

### 7. 用 PID 指令设置 PID 控制器的参数

除了在 PID\_Compact 技术对象的组态窗口和指令下面的监视窗口中设置 PID\_Compact 指令的参数外，也可以直接输入指令的参数，未设置（采用默认值）的参数为灰色。点击指令方框下沿向下的箭头，将显示出更多的参数（见图 9-12）。点击图中指令方框下沿向上的箭头，将不显示指令中灰色的参数。点击某个参数的实参，可以直接输入地址或常数。

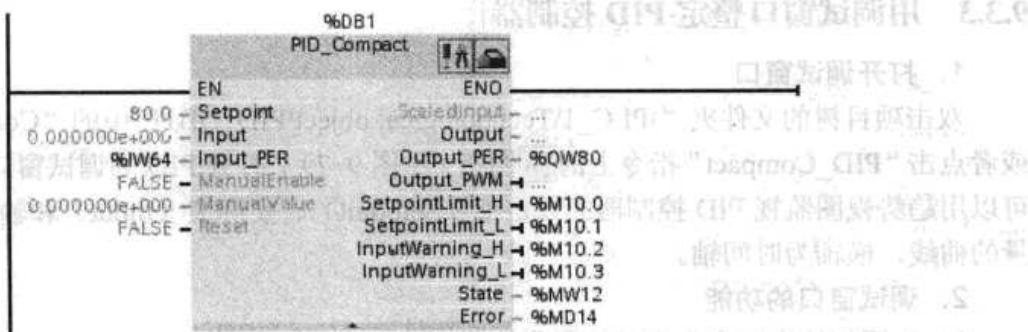


图 9-12 PID 指令

### 8. PID\_Compact 指令的输入/输出变量

PID\_Compact 指令的输入/输出变量见表 9-1 和表 9-2。

表 9-1 PID\_Compact 指令的输入参数

参数名称	数据类型	说 明	默 允
Setpoint	Real	自动模式的控制器设定值	0.0
Input	Real	作为实际值（即反馈值）来源的用户程序的变量	0.0
Input_PER	Word	作为实际值来源的模拟量输入	W#16#0
ManualEnable	Bool	上升沿选择手动模式，下降沿选择最近激活的操作模式	FALSE
ManualValue	Real	手动模式的 PID 输出变量	0.0
Reset	Bool	重新启动控制器，1 状态时进入 Inactive（不活动）模式，控制器输出变量为 0，临时值被复位，PID 参数保持不变	FALSE

表 9-2 PID\_Compact 指令的输出参数

参数名称	数据类型	说 明	默认值
Scaledinput	Real	经比例缩放的实际值的输出	0.0
Output	Real	用于控制器输出的用户程序变量	0.0
Output_PER	Word	PID 控制器的模拟量输出	W#16#0
Output_PWM	Real	使用 PWM 的控制器开关输出	FALSE
SetpointLimit_H	Bool	1 状态时设定值的绝对值达到或超过上限	FALSE
SetpointLimit_L	Bool	1 状态时设定值的绝对值达到或低于下限	FALSE
InputWarning_H	Bool	1 状态时实际值达到或超过报警上限	FALSE
InputWarning_L	Bool	1 状态时实际值达到或低于报警下限	FALSE
State	Int	PID 控制器的当前运行模式, 0~4 分别表示 Inactive、初始启动阶段的自整定、运行点自整定、自动、手动模式	16# 0000
Error	DWord	错误信息, 为 0 没有错误; 非零有 1 个或多个错误, 控制器进入 Inactive 模式。错误代码见 PID 指令的在线帮助	

可以组态使用输入 Input 或 Input\_PER，可以同时使用变量 Output、Output\_PER 和 Output\_PWM。

### 9.3.3 用调试窗口整定 PID 控制器

#### 1. 打开调试窗口

双击项目树的文件夹 “\PLC\_1\Technological object\PID\_DB1” 中的 “Commissioning”，或者点击 “PID\_Compact” 指令上的图标（见图 9-7），打开 PID 的调试窗口（见图 9-13）。可以用趋势视图监视 PID 控制器的设定值（Setpoint）、实际值（Input）和输出（Output）变量的曲线，横轴为时间轴。

#### 2. 调试窗口的功能

- 1) 使用初始启动自整定功能优化控制器。
  - 2) 使用运行点自整定功能优化控制器，可以实现最佳调节。
  - 3) 用趋势视图监视当前的闭环控制。
  - 4) 通过手动设置控制器的输出值来测试过程。
- 建立与 CPU 的在线连接后，才能使用调试窗口。

#### 3. 基本操作

- 1) 用上面的 “Sample time” 下拉式列表选择采样时间。

2) 点击 “Start Measurement”（开始测试）按钮，启动调试窗口的运行，开始用趋势视图记录设定值、实际值和控制器的输出值。

点击 “Stop Measurement” 按钮，结束调试功能。可以用趋势视图中的曲线来分析 PID 控制的效果。关闭调试窗口后，趋势视图中的记录被停止，记录的数据被删除。

#### 4. 显示模式

可以用启动、停止测试按钮下面的选择框，选用下面的显示模式来显示趋势记录：

1) Strip（连续显示）：新的趋势值从趋势区的右边进入，较早的趋势值显示到趋势区的最左边为止，时间轴不能移动。

2) Scope (区域跳跃显示)：新的趋势值从趋势区的左边开始往右移动，到达趋势区的右边缘时，监控区往右移动一个视图宽度，原有的趋势曲线消失，新的趋势值又从左边出现。

3) Sweep (旋转显示)：趋势曲线固定不变，出现一根从左往右移动的垂直曲线，曲线左边的背景色为白色，是新出现的趋势值，曲线左边的背景色为浅绿色，是原来的趋势值。曲线移动到最右边后，返回最左边，又往右移动。时间轴不能移动。

4) Static (静态区域显示)：趋势视图的写入被中断，在后台记录新的趋势值。显示的是趋势的历史曲线，时间轴可以在整个记录区间移动。

趋势区左上角的图例是当前的时间和趋势值（见图 9-13）。

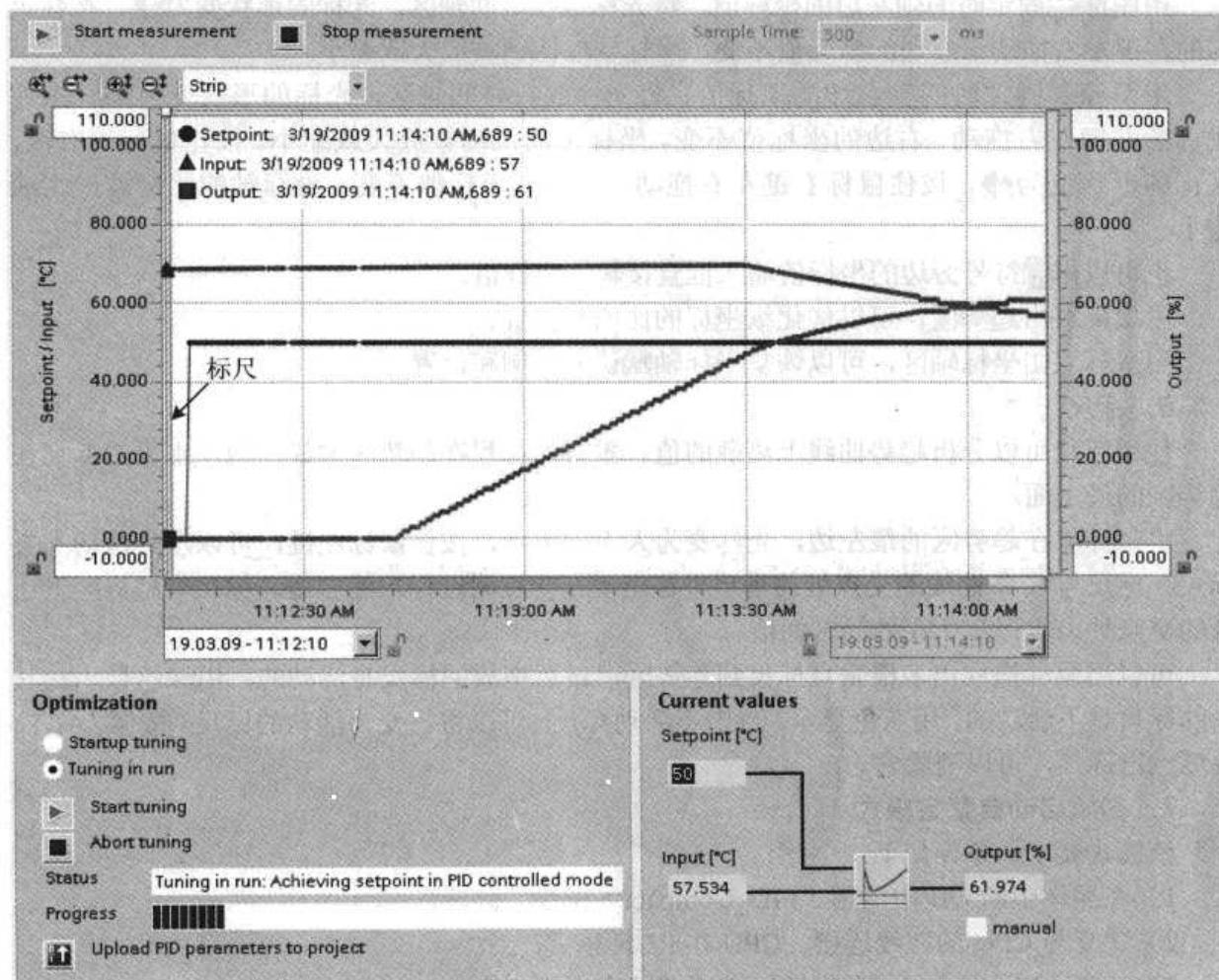


图 9-13 PID 控制器的调试窗口

## 5. 移动坐标轴与改变坐标轴的比例

点击坐标轴两端的图标，可以将一端的坐标值锁死（）或解除闭锁（）。

可以用左上角的按钮拉伸或压缩时间轴，用按钮拉伸或压缩左边的设定值/实际值轴和右边的控制器输出变量轴。如果用图标固定了一端的坐标值，只能用上述按钮调节另一端的坐标值。如果固定了两端的坐标值，不能调节坐标轴的比例。

两端的坐标值没有被闭锁时，将光标放在某个坐标轴区，光标的形状变为 $\pm$ 或 $\times$ ，按下鼠标的左键上下拖动或左右拖动，可以整体移动坐标轴上的刻度值。只要坐标轴的一个端点值被闭锁，就不能进行这一操作。

用图标 $\square$ 固定某个纵坐标轴下面的坐标值，将光标放在纵坐标轴区，光标的形状变为 $\blacktriangleup$ ，按住鼠标的左键上下拖动，下面的坐标值不变，坐标轴的比例被放大或缩小。

用图标 $\heartsuit$ 固定某个纵坐标轴上面的坐标值，将光标放在纵坐标轴区，光标的形状变为 $\blacktriangledown$ ，按住鼠标的左键上下拖动，上面的坐标值不变，坐标轴的比例被放大或缩小。按住鼠标的右键，光标的形状变为 $\blacktriangleright$ ，按住鼠标右键上下拖动，下面的坐标值不变，坐标轴的比例被放大或缩小。

用图标 $\blacksquare$ 固定时间轴左边的坐标值，将光标放在时间轴区，光标的形状变为 $\blacktriangleleft$ ，按住鼠标的左键左右拖动，左边的坐标值不变，坐标轴的比例被放大或缩小。

用图标 $\blacktriangleleft\blacktriangleright$ 固定时间轴右边的坐标值，将光标放在时间轴区，光标的形状变为 $\blacktriangleleft\blacktriangleright$ ，按住鼠标的左键左右拖动，右边的坐标值不变，坐标轴的比例被放大或缩小。按住鼠标的右键，光标的形状变为 $\blacktriangleright$ ，按住鼠标右键左右拖动，左面的坐标值不变，坐标轴的比例被放大或缩小。

也可以用 $\square$ 符号旁边的坐标值输入框直接输入坐标值。

用鼠标双击趋势区，可以优化纵坐标的比例和位置。

用鼠标双击坐标轴区，可以恢复坐标轴默认的比例和位置。

## 6. 标尺

使用标尺可以分析趋势曲线上离散的值。垂直的标尺在趋势区的最左边，水平的标尺在趋势区的最上面。

将鼠标放在趋势区的最左边，光标变为人手的形状，按住鼠标左键，可以左右拖动垂直标尺。标尺与 3 条曲线的交点处的垂直坐标出现在交点附近标尺的右边。标尺与趋势区下边缘的交点处出现标尺所在处的时间值。

可以用鼠标拖放出多根垂直标尺到趋势区，最后拖放的标尺是活动的，用深色显示，其余的标尺是不活动的，用灰色显示。点击不活动标尺，可以将它变为活动的标尺。按住 $\langle ALT\rangle$ 键后点击标尺，可以消除它。

## 7. 初始启动自整定模式

该模式要求的条件如下：

- 1) 在循环中断 OB 中调用“PID\_Compact”指令。
- 2) 建立与 CPU 的在线连接，CPU 在 RUN 模式。
- 3) 点击“Measuring on”按钮，激活调试窗口的功能。
- 4) 未选中“Manual”（手动）复选框。
- 5) 设定值与实际值在组态的限制值之内。
- 6) 设定值与实际值的差值大于 50%。

操作步骤如下：

- 1) 用单选框选中调试窗口的“Optimization”区的“Start tuning”选项。
- 2) 点击“Start tuning”按钮 $\blacktriangleright$ ，启动自调整（见图 9-14）。状态域（Status，见图 9-13）显示当前的步骤和可能出现的错误。进度条（Progress）显示当前步骤的进展。

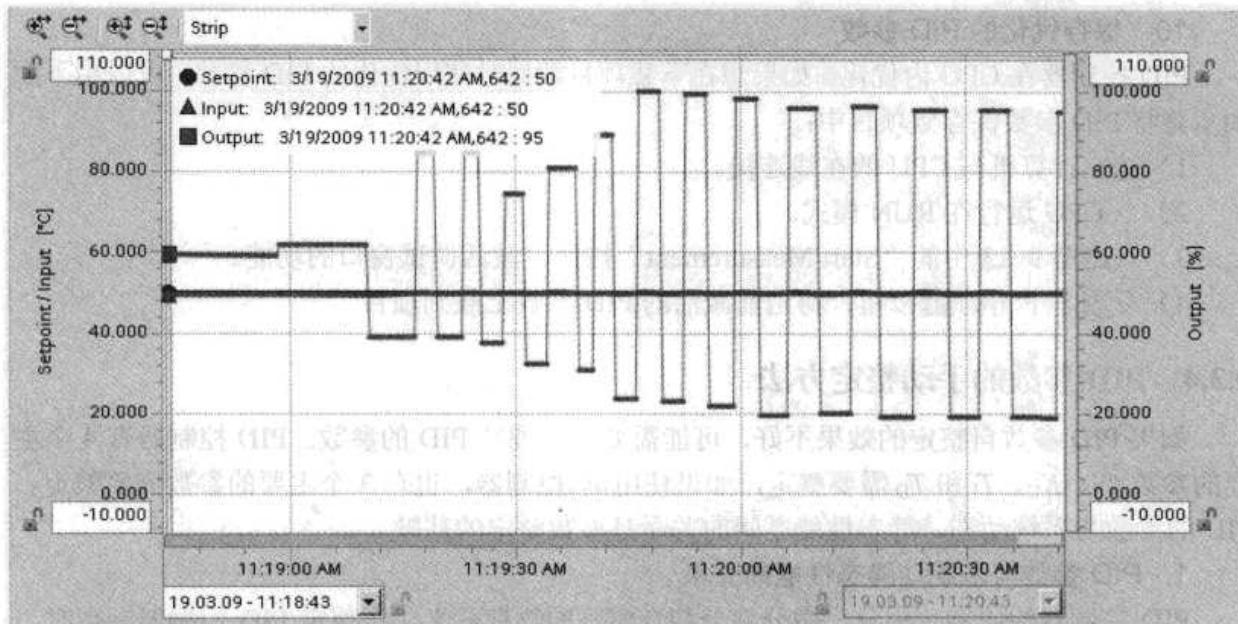


图 9-14 PID 控制器的调试窗口

当进度条达到 100%，认为自调整功能被阻塞时，点击“Abort tuning”按钮■，检查技术对象的组态，如果有必要，重新启动自调整。

如果自调整没有出错，PID 的参数被优化。将 PID 控制器切换到自动模式，使用优化的参数。上电和重新启动 CPU 时，优化的参数被保持。

### 8. 运行点自整定模式

该模式要求的条件与初始启动自整定模式的基本上相同。其区别在于第 6 条改为设定值与实际值的差值小于 50%。如果大于 50%，应首先进行初始启动自整定，完成后自动进行运行点自整定。操作步骤如下：

- 1) 用单选框选中调试窗口的“Optimization”区的“Tuning in run”（运行中调试）选项。
- 2) 点击“Start tuning”按钮▶，启动自调整。状态域显示当前的步骤和可能出现的错误。进度条显示当前步骤的进展。

自调整过程观察到的现象、运行的结果和处理的方法与初始启动自整定相同。

### 9. 手动模式

如果想通过手动设置控制器的输出值来测试过程，可以使用调试窗口的手动模式。该模式要求的条件与初始启动自整定模式的前 3 条相同。操作步骤如下：

- 1) 选中“Current values”（当前值）区的“Manual”（手动）复选框（见图 9-13 的右下角），控制器输出与控制变量输出之间的连接被断开，闭环控制最后的控制输出变量作为手动的控制器输出变量。
- 2) 在“Output”域输入以%为单位的希望的控制器输出变量的值。
- 3) 按〈Enter〉键，控制器输出变量被写入 CPU，并被立即激活。

控制器连续地监视实际值，实际值超过限制将在“Status”域显示出来，PID 控制器进入 Inactive 模式。

## 10. 保存优化的 PID 参数

PID 控制器在 CPU 内优化。如果想在下载项目数据时使用优化的 PID 参数，可以用下面的步骤将 PID 参数保存到项目中：

- 1) 建立计算机与 CPU 的在线连接。
- 2) 令 CPU 运行在 RUN 模式。
- 3) 点击图 9-13 中的“Start Measurement”按钮，激活调试窗口的功能。
- 4) 点击左下角的  按钮，将当前激活的 PID 参数上载到项目。

### 9.3.4 PID 参数的手动整定方法

如果 PID 参数自整定的效果不好，可能需要手动调节 PID 的参数。PID 控制器有 4 个主要的参数  $T_S$ 、 $K_C$ 、 $T_I$  和  $T_D$  需要整定，如果使用 PI 控制器，也有 3 个主要的参数需要整定。PID 的参数与系统动态、静态性能之间的关系是参数整定的基础。

#### 1. PID 参数与系统动静态性能的关系

PID 控制器的比例、积分、微分部分都有明确的物理意义，在整定 PID 控制器参数时，可以根据控制器的参数与系统动态、稳态性能之间的定性关系，用试验的方法来调节控制器的参数。有经验的调试人员可以较快地得到较为满意的调试结果。

在调试中最重要的问题是在系统性能不尽如人意时，知道应该调节哪一个参数，应往什么方向调整，即该参数应该增大还是减小。

##### (1) 比例增益

比例部分与误差同步，它的调节作用及时，较积分控制的反应快。在误差出现时，比例控制能立即给出控制信号，使被控制量朝着误差减小的方向变化。

如果  $K_C$  太小，会使系统输出量变化缓慢，调节时间过长。如果系统中没有积分作用，比例调节存在稳态误差，稳态误差与  $K_C$  成反比。增大  $K_C$  使系统反应灵敏，上升速度加快，且可以减小稳态误差。但是  $K_C$  过大会使超调量增大，振荡次数增加，调节时间加长，导致动态性能变坏， $K_C$  过大甚至会使闭环系统不稳定。

如果 PID 控制器中有积分作用（例如采用 PI 或 PID 控制），积分能消除阶跃输入的稳态误差，这时可以将  $K_C$  调得小一些。

##### (2) 积分时间

积分部分与误差对时间的积分成正比。因为积分时间  $T_I$  在积分项的分母中， $T_I$  越小，积分速度越快，积分作用越强。

控制器中的积分作用与当前误差的大小和误差的历史情况（累加值）都有关系，只要误差不为零，控制器的输出就会因积分作用而不断变化，误差为正时积分项不断增大，反之不断减小。积分项有减小误差的作用，一直到系统处于稳定状态，这时误差恒为零，比例部分和微分部分均为零，积分部分才不再变化，并且刚好等于稳态时需要的控制器的输出值。因此积分部分的作用是消除稳态误差和提高控制精度，积分作用一般是必须的。

但是积分作用具有滞后特性，不像比例部分，只要误差一出现，就立即起作用。积分作用太强会使系统响应的动态性能变差，超调量增大，甚至使系统不稳定。因此积分作用很少单独使用，它一般与比例和微分联合使用，构成 PI 或 PID 控制器。

PI 控制器既克服了单纯的比例调节有稳态误差的缺点，又避免了单纯的积分调节响应慢、

动态性能不好的缺点，因此被广泛使用。

综上所述，积分作用太强（即  $T_I$  太小）使系统的稳定性变差，超调量增大；积分作用太弱（即  $T_I$  太大）系统消除稳态误差的速度减慢， $T_I$  的值应取得适中。

### (3) 微分时间

微分部分的输出与误差的微分（即误差的变化速率）成正比，反映了被控量变化的趋势，其作用是阻碍被控量的变化。在图 9-6 启动过程的上升阶段，当  $c(t) < c(\infty)$  时，被控量尚未超过其稳态值，超调量还没有出现。但是因为误差  $e(t)$  不断减小，误差的微分和控制器输出的微分部分为负，减小了控制器的输出量，相当于提前给出制动作用，以阻碍被控量的上升，所以可以减少超调量。因此微分控制具有超前和预测的特性，在超调量出现之前，就能提前给出控制作用。

闭环控制系统的振荡甚至不稳定的根本原因在于有较大的滞后因素，因为微分项能预测误差变化的趋势，这种“超前”的作用可以抑制滞后因素的影响，适当的微分控制作用可以使超调量减小，缩短调节时间，增加系统的稳定性。其缺点是对干扰噪声敏感，使系统抑制干扰的能力降低。

对于有较大惯性或滞后的被控对象，控制器输出量变化后，要经过较长的时间才能引起反馈量的变化，如果 PI 控制器的控制效果不理想，可以考虑在控制器中增加微分作用，以改善系统在调节过程中的动态特性。

微分时间  $T_D$  表示了微分作用的强弱， $T_D$  越大，微分作用越强。但是  $T_D$  太大可能会引起频率较高的振荡。

如果将微分时间设置为 0，微分部分将不起作用。

### (4) 采样周期

采样周期  $T_S$  越小，采样值越能反映模拟量的变化情况。但是  $T_S$  太小会增加 CPU 的运算工作量，相邻两次采样的差值几乎没有什么变化，将使 PID 控制器输出的微分部分接近为零，所以也不宜将  $T_S$  取得过小。

确定采样周期时，应保证在被控量迅速变化时（例如启动过程中的上升段）能有足够的采样点数，以保证不会因采样点过稀而丢失被采集的模拟量中的重要信息。

表 9-3 给出了过程控制中采样周期的经验数据，表中的数据仅供参考，实际的采样周期需要经过现场调试后确定。S7-1200 中 PID 的采样时间精度用循环中断来保证。

表 9-3 采样周期的经验数据

被控制量	流 量	压 力	温 度	液 位	成 份
采样周期/s	1~5	3~10	15~20	6~8	15~20

## 2. PID 参数的调整方法

在手动调节时，为了减少需要整定的参数，可以首先采用 PI 控制算法。为了保证系统的安全，避免出现系统不稳定或超调量过大的异常情况，在调试开始时应设置比较保守的参数，例如增益不要太大（可以小于 1），积分时间不要太小。给出一个阶跃给定信号后，观察系统输出量的波形。根据输出波形提供的系统性能指标的信息，以及 PID 参数与系统性能的关系，反复调节 PID 的参数。

如果阶跃响应的超调量太大，经过多次振荡才能稳定或者根本不稳定，应减小增益、增大积分时间。如果阶跃响应没有超调量，但是被控量上升过于缓慢，过渡过程时间太长，应按相反的方向调整参数。

如果消除误差的速度较慢，可以适当减小积分时间。

如果反复调节  $K_C$  和  $T_I$ ，超调量仍然较大，可以加入微分， $T_D$  从 0 逐渐增大，反复调节  $K_C$ 、 $T_I$  和  $T_D$ 。

总之，PID 参数的调试是一个综合的、互相影响的过程，实际调试过程中的多次尝试是非常重要的步骤，也是必须的。

变频器一般也有 PID 控制功能。如果用 S7-1200 控制变频器，并且需要控制的变量直接与变频器有关，例如用变频水泵控制水压，可以优先考虑使用变频器的 PID 功能。

## 附录 随书光盘内容简介

1. S7-1200 的编程软件 STEP 7 Basic V10.5
2. 用户手册与资料
  - S7-1200 可编程控制器系统手册。
  - S7-1200 system manual。
  - S7-1200 Catalog ST 70 N (产品样本)。
  - S7-1200 入门手册。
  - S7-1200 入门指南。
  - 从 S7-1200 过渡到 S7-1200。
  - 紧凑型交换机模块 CSM 1277 操作说明。
  - 精简系列面板操作说明。
  - 基本面板入门教程。
  - 精简面板下载参考手册。
3. 书中的例程
  - TimerCounter: 定时器计数器指令应用例程。
  - Dolly: 小车的顺序控制例程。
  - ComplexSFC: 有选择系列与并行序列的顺序控制例程。
  - Drilling: 专用钻床顺序控制例程。
  - Manipulator: 机械手顺序控制例程。
  - HSC\_COUNT: PWM 脉冲发生器与 HSC 计数例程。
  - FrequencyTest: 用 HSC 测量频率的例程。
  - FB\_FC: 功能、功能块和多重背景数据块的应用例程。
  - Interrupt: 中断组织块应用例程。
  - HWInterrupt1: 硬件中断组织块应用例程。
  - HWInterrupt2: 硬件中断组织块应用例程。
  - PLC\_PLC: 开放式用户通信例程。
  - PLC\_HMI: 精简系列面板组态与应用例程。
  - PID: PID 控制例程。

## 参 考 文 献

- [1] 廖常初. PLC 编程及应用[M]. 3 版. 北京: 机械工业出版社, 2008.
- [2] 廖常初. S7-300/400 PLC 应用技术[M]. 2 版. 北京: 机械工业出版社, 2008.
- [3] 廖常初, 陈晓东. 西门子人机界面(触摸屏)组态与应用技术[M]. 2 版. 北京: 机械工业出版社, 2008.
- [4] 廖常初, 祖正容. 西门子工业网络的组态编程与故障诊断 [M]. 北京: 机械工业出版社, 2009.
- [5] 廖常初. FX 系列 PLC 编程及应用 [M]. 北京: 机械工业出版社, 2005.
- [6] Siemens AG. S7-1200 system manual, 2009.
- [7] Siemens AG. STEP 7 Basic 的帮助文件, 2009.
- [8] Siemens AG. S7-1200 Catalog ST 70 N, 2009.
- [9] Siemens AG. 精简系列面板操作说明, 2009.