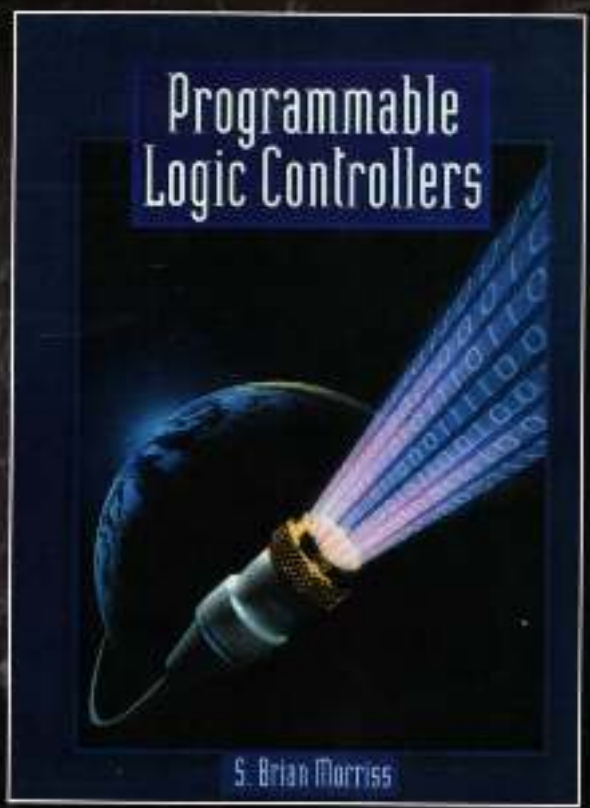
计 算 机 科 学 丛 书

可编程逻辑控制器

（美）S. Brian Morriss 著　　杨智 王琅 朱海锋 译



Programmable
Logic Controllers

机械工业出版社
China Machine Press

本书全面介绍可编辑逻辑控制器(PLC)，涉及三个主流制造商的五大PLC系列：

- 罗克韦尔自动化公司的Allen-Bradley PLC-5系列
- 罗克韦尔自动化公司的Allen-Bradley SLC 500系列
- 西门子AG公司的Simatic 5(S5)系列
- 西门子AG公司的Simatic 7(S7)系列
- 欧姆龙电子公司的CQM1系列

本书详细讲解PLC编程，以及整个数据字和数据文件的操作；怎样编程来控制连续的模拟过程；如何配置PLC的中断操作，以及中断处理的过程。

本书还讨论了PLC的通信，并且涵盖了结构化编程技术，甚至覆盖了“开放的”结构化编程的IEC 1131-3标准，并介绍了设置PLC和对PLC进行故障诊断的方法。

www.PearsonEd.com

ISBN 7-111-18427-0

9 787111 184270

HZ BOOKS

华章图书

上架指导：计算机/硬件设计

PEARSON Education

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

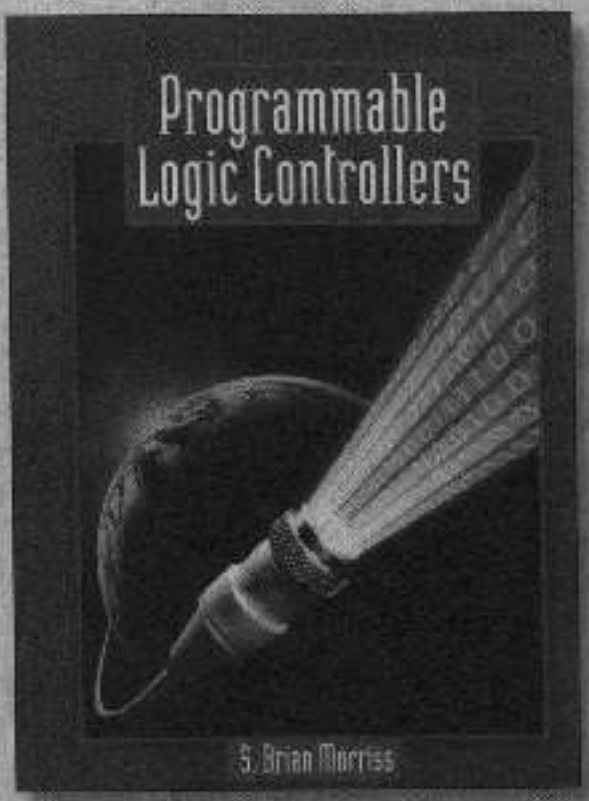
投稿热线：(010) 88379604
购书热线：(010) 68995259, 68995264
读者信箱：hzjsj@hzbook.com

ISBN 7-111-18427-0
定价：66.00 元

计 算 机 科 学 丛 书

可编程逻辑控制器

（美）S. Brian Morriss 著　　杨智 王琅 朱海锋 译



Programmable Logic Controllers

机械工业出版社
China Machine Press

本书以国际著名自动化厂家美国罗克韦尔、德国西门子、日本欧姆龙公司生产的典型可编程逻辑控制器（PLC）为技术背景，系统、完整地阐述了可编程控制器的系统构成与工作原理。内容包括：PLC的组成与原理、软件操作指令与数据处理、存储器的组织与数据操作、编程语言与程序设计、PLC的安装与配置、中断、过程控制、通信、PLC应用、故障检测及PLC的未来。

本书适合作为高等院校电气、自动化、电子、计算机、机电类专业学生的可编程逻辑控制器教材，也可供相关领域的技术人员参考。

Simplified Chinese edition copyright © 2006 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *Programmable Logic Controllers* (0-13-095565-5) by S. Brian Morriss, Copyright© 2000.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Prentice-Hall, Inc.

本书封面贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书法律顾问　北京市展达律师事务所

本书版权登记号：图字：01-2003-1988

图书在版编目（CIP）数据

可编程逻辑控制器/（美）默里斯（Morriss，S.B）著；杨智，王琅，朱海锋译．—北京：机械工业出版社，2006.5

（计算机科学丛书）

书名原文：Programmable Logic Controllers

ISBN 7-111-18427-0

Ⅰ.可…　Ⅱ.①默…　②杨…　③王…　④朱…　Ⅲ.可编程序控制器　Ⅳ.TP332.3

中国版本图书馆CIP数据核字（2006）第007276号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：范运年

北京诚信伟业印刷有限公司印刷·新华书店北京发行所发行

2006年5月第1版第1次印刷

184mm×260mm·35.5印张

定价：66.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

译 者 序

随着当今科技的飞速发展，自动化技术已经深入到了各个领域中，无论是神舟飞船的顺利上天还是奥运赛事的正常进行，自动化技术都发挥着极其重要的作用。

说到自动化就不能不提到可编程逻辑控制器（Programmable Logic Controller，PLC），它是以微处理器为基础，综合了计算机技术、自动控制技术、通信技术的工业控制产品，是在硬接线逻辑控制技术和计算机技术的基础上发展起来的。在进入崭新的 21 世纪的工控领域，PLC 正日益发挥着举足轻重的作用。因此，如果有一本能够详细、透彻地介绍 PLC 的书籍，将会对广大从事自动化的工程技术人员起到事半功倍的效果。这也正是本书诞生的初衷。

也许有人该摇头了，诚然，PLC 的重要性不言而喻，但现在关于 PLC 的书籍多得已经令人目不暇接了，还有必要再出本新的吗？对此，译者表示有足够的信心能够让读者满意。这并不是一句空话，而是由本书以下的一些特点决定的。

1. 本书作者

本书作者 S. Brian Morriss 从事自动化研究领域已逾 20 年了，是加拿大安大略省剑桥市自动化工具系统公司（ATS）的一名设计工程师。ATS 是世界上定制自动化操作系统的大型供应商之一。Morriss 最近在加拿大的 Kitchener 大学的 Conestoga 学院任教，并且参与了一个很成功的以机器人和自动化机械技术程序为背景的项目。他负责 Conestoga 学院 PLC 训练站的几种 PLC 的采购、安装和配置，并且教授 PLC 编程已逾十年。Morriss 还是《Automated Manufacturing Systems》（New York：Glencoe/McGraw-Hill，1995）一书的作者，此书于 1996 年在新加坡发行国际版本，并被译为韩文和中文。

2. 本书组织架构

本书的组织架构清晰明确，内容详实、透彻，有相当的深度、广度和前瞻性。从结构上讲，每章基本上都由以下几个部分组成：

（1）学习目标：以凝练、精确的语言简要介绍该章将要学习的知识，力求使读者在阅读每章前做到心中有数，从而全盘掌握整章的内容，以便根据个人的情况有所侧重。

（2）核心内容：是每章内容的重点，围绕着每章的主题展开介绍，并且写作手法由浅入深，由简到繁，先从相关知识引入，再层层铺开，中间结合需要适当插入了许多相关的指令程序和插图，让读者能够结合事例，融会贯通。

此外，本书内容和其他同类书相比最大的一个特色就是，作者在讲解 PLC 的每块知识时都先以基础知识开始，然后再结合具体厂家、具体型号一一进行介绍。这样做的好处不言而喻，使读者不至于始终面对枯燥的理论知识，可以结合实际，活学活用，这样读者在阅读本书的同时，就能够学以致用了。

在选择 PLC 的代表厂家和型号时作者也费了一番心思，囊括范围很广，分别选择了北

美自动化的领导者罗克韦尔自动化公司的 Allen-Bradley 系列的 PLC、引领着欧洲市场并正逐渐占领北美市场的西门子 AG 的 SIMATIC PLC、亚洲的欧姆龙电子的 CQM1，厂家不可谓不大，品种不可谓不多，代表着当今 PLC 市场的主流和发展方向，应该说满足我们日常工作、学习需要已绰绰有余了。

(3) 故障检修：这部分也是本书的一个特色，在这里作者假设了读者在具体的操作过程中可能会犯的各种情况的错误，并且给出具体的例子和解决方法，这就给初学者以很大的帮助，使他们尽量少犯或不犯这些错误。

(4) 课后习题：每章的课后都附带有少而精的习题，用于复习整章学习的内容，考查形式多种多样。

(5) 编程练习（不需要 PLC)：此部分编制了一些不需要 PLC 实际参与情况下的编程练习，读者在亲自使用 PLC 前就能进行模拟的编程实验，极大地方便了读者。

(6) 建议做的 PLC 实验室练习：这部分就要应用到 PLC 了，首先设计一个需满足的条件系统，然后实际地编写程序来操作和控制系统的运行，让读者进行实战演习。

3. 本书的翻译

我们翻译的原则是不仅力求准确无误、贴近原文，还会根据实际情况适当取舍和变通。翻译外文书籍时，最忌讳的是一字一句的死板翻译，这样看起来好像是尊重了原书意思，但事实上却使读者读起来如坠入云雾中。因为大家都知道，外国人的写作手法、思考方法、表达方法和中国人有很大区别，因此，我们在翻译时尽量使自己站在中国读者的角度来进行翻译，使翻译出来的东西能够适合中国读者的思维和理解习惯，让他们轻松、顺利地了解和掌握世界最新的产品和技术，这是我们翻译的最终目的。

有了以上一些特点，我们有理由相信它是读者系统地、科学地、透彻地学习 PLC 知识的一本好书，因此把它郑重地介绍给高等院校信息大类、电气自动化、自动化、机械类学生的教学用书；当然，本书也可作为工程技术人员学习 PLC 的参考书。

本书第 1 章、第 3 章、第 4 章、第 12 章、第 14 章、第 16 章由杨智翻译，第 2 章、第 5 章、第 6 章、第 8 章、第 9 章、第 11 章、附录由王琅翻译，第 7 章、第 10 章、第 13 章、第 15 章由朱海锋翻译。全书由朱海锋统一校对。

本书在翻译过程中得到了王李冬子、韩恺、刘飞、徐子峰、陆和健、曾海等同学的大力支持和帮助，他们对译稿做了许多贡献，这里向他们表示诚挚的感谢！

由于译者水平有限，虽经多次审核与校对，但难免有不当之处，敬请读者批评指正。

杨　智

2005 年 3 月 12 日于中山大学

前　言

本书以向读者介绍可编程逻辑控制器（PLC）和对 PLC 编程进行简单的布尔量操作作为开篇。与其他著作不同，本书接下来的章节描述了现代 PLC 先进的、可利用的功能特征。读者将会了解到在 PLC 编程中，如何操作整个数据字和数据文件；如何通过编程来控制连续的模拟过程；以及如何配置 PLC 使它中断正常程序的扫描循环并转向需要立即响应的控制过程。通信、结构化编程技术和开放的结构化编程标准 IEC 1131-3 也将会讨论到。读者还将了解到如何设置 PLC 和检修 PLC 的故障。

本书的每一章都描述了 PLC 如何进行一类特殊的操作，以及为什么程序员要利用 PLC 的这些特点。通过使用与普通编程语言、PLC 开放界面一致的语言可以理解这些基础的计算机操作。在每一章中，我们都将使用来自三个主流制造商的各自的专用术语来描述他们生产的五种系列的 PLC。

罗克韦尔（Rockwell）自动化公司的 Allen-Bradley 系列的 PLC 是北美自动化领域的领导者。事实上，在 PLC 的特征和性能方面，他们为其他 PLC 制造商提供了必须遵守的标准。Allen-Bradley 正在积极组织开发开放的标准。

1）Allen-Bradley PLC-5 是一种标准的 PLC，有增强型系列和经典系列。一个经典的 PLC-5 并不能解决所有本书提到的问题。

2）Allen-Bradley SLC 500 是一种小型的 PLC。SLC 500 有和 PLC-5 系列类似的特征。SLC 5/04 是最新的版本。低档的 SLC 500 可能不会提供这里描述的所有特征，尽管最小型的 SLC 500（也叫做集成模式）具有一些大型模块不具备的功能，这些功能也将会介绍到。这里介绍的编程语言也适用于 Allen-Bradley 的 MicroLogic 系列。

西门子（Siemens）AG 公司的 SIMATIC 系列 PLC 领导着欧洲的市场，并开始逐渐在北美市场占据显著的地位。他们以前在北美市场使用西屋（Westinghouse）和德州仪器（Texas Instruments）的商标，但现在开始使用自己的品牌。西门子的 PLC 系列通常提供了编程方面的最新创意，这样使得 PLC 功能非常强大，但有时对于用户来说难以入手。西门子 PLC 一直致力于使用开放的标准，所以可以说西门子已经成为推动开放标准的一个主要力量。

3）西门子 AG 的 Simatic 5（S5）通过 STEP 5 编程。STEP 5 和其他 PLC 一样包括容易使用的梯形图逻辑，但在本书中我们也经常使用 STEP 5 的 STL 编程语言。任何了解梯形图逻辑的读者将会很容易地使用 STEP 5 的梯形图语言，但是一些 S5 的特征只能使用 STL 语言编程。我们将专门讨论 S5 PLC 的两个版本：标准的 S5-115U 型和小型的 S5-100U 型，它们和 S5 其他成员的编程都是类似的。

4）西门子 AG 的 Simatic 7（S7）通过 STEP 7 编程。STEP 7 以 IEC 1131-3 标准为基础，IEC 1131-3 是一个描述一系列通用编程语言的非强制性国际标准。使用 IEC 1131-3 语

言编写的程序可以下载到任何支持 IEC 1131-3 的 PLC 上。S7 系列 PLC 刚刚发布，所以它们可能与未来的 PLC 最为接近。我们将特别讨论 S7 PLC 系列的两个版本：S7-300 型和 S7-400 型，它们和 S7 家族的其他成员具有类似的编程方式。

欧姆龙（OMRON）电子公司的 CQM1 系列 PLC——欧姆龙电子公司的 PLC 领导着日本的市场。欧姆龙是北美的传感器供应商之一，直到最近才开始积极地推动自己的 PLC 系列。欧姆龙之所以拥有一大部分市场份额，很大程度上是因为价格低，但同时也因为他们的 PLC 具有良好的品质。

5）欧姆龙电子公司的 CQM1 是一种小型的模块化的 PLC。最近引进的 CQM1 具有很多欧姆龙标准 C200H PLC 的特征，并且它们的编程语言（几乎）一样。一些类型的 CQM1 PLC 提供特殊用途的功能，这里也将涉及到。但欧姆龙的梯形图语言与其他梯形图语言有很大的区别。

本书不包括共接近 30 本手册的所有细节。读者应该可以从需要使用的 PLC 的使用手册中查到列表数据、语法信息和其他特殊信息。

本书的组织架构

本书的第 1、2、3、4 章介绍了 PLC 的编程并描述了 PLC 的组件和独特的操作系统特性。包括 PLC 的发展历史、与传感器以及执行器的连接、布尔逻辑编程的使用以及定时器、计数器的使用等。

第 5、6、7 章涵盖了 PLC 编程的一些主题，如存储结构、数据类型、寻址方式和微处理器基础。也将涉及到具有各种位（如字节、字、文件、数组、结构）的数据元素、数学和逻辑操作的指令。

第 8、9 章介绍了结构化编程的技巧，从最初通过跳转和子程序的主控继电器指令，到通过使用数据声明和参数传递来实现数据存储的严格控制的现代要求。也会介绍用于开放编程语言的 IEC3 1131-3 标准的要求。

第 10 章描述了如何启动和配置一个 PLC。

第 11、12、13、14 章介绍了特殊应用的高级主题。会讨论使用中断来定制一个现代 PLC 的响应特性，还会涉及到使用 PLC 的过程控制、配置和 PLC 间编程数据的通信。通过一个例子展示了 PLC 如何与机器人共享一个工作块。

第 15 章描述故障检测。

第 16 章分析 PLC 技术的发展趋势以帮助读者了解将来 PLC 的技术特征。

致谢

感谢我的学生们，他们检查了本书中的大部分材料。也感谢罗克韦尔自动化公司、西门子 AG 公司、CBS 的西屋分公司和欧姆龙电子公司的许多工作人员。没有他们，我们学校不可能拥有一个 PLC 实验室。最后，感谢我的妻子——琳达，在我为了本书交稿而工作时，她单独度过了去年夏天的大部分时间。

我也感谢下面的公司允许我使用他们的图表和图形：Allen-Bradley 公司 LLC、西门子公司、欧姆龙公司以及 CBS 的西屋分公司。

我还感谢以下评审对本书有益的反馈意见：密歇根理工大学的 Tony Farrell，田纳西技

术大学的 Bill Maxwell 和加拿大 Conestoga 大学的 John Tielemans、Jeff Uniac。

关于作者

S. Brian Morriss 从事自动化领域研究已经 19 年了，他是加拿大安大略省剑桥市自动化工具系统公司（ATS）的一名设计工程师。ATS 是定制自动化操作系统的全球最大的供应商之一。Morriss 最近在加拿大安大略省 Kitchener 的 Conestoga 学院任教，并且参与了一个成功的以机器人和自动化机械工程技术为背景的项目。他负责 Conestoga 学院 PLC 训练站的几种 PLC 的采购、安装和配置，并且教授 PLC 编程已逾十年。Morriss 是《Automated Manufacturing Systems》（New York：Glencoe/McGraw-Hill，1995）一书的作者，此书于 1996 年在新加坡发行国际版本，并被译为韩文和中文。

1977 年，Morriss 先生在 Waterloo 大学获得系统设计工程专业的科学学士学位。

目　录

译者序
前言
第 1 章　什么是 PLC …… 1
1.1　学习目标 …… 1
1.2　PLC 基础 …… 2
1.2.1　PLC 的结构 …… 2
1.2.2　操作系统和应用程序 …… 3
1.2.3　PLC 用户程序 …… 5
1.3　选择合适的 PLC …… 6
1.4　PLC 的革新 …… 8
1.5　故障检修 …… 11
习题 …… 12
第 2 章　PLC 组件 …… 13
2.1　学习目标 …… 13
2.2　CPU 模块 …… 14
2.3　框架或总线 …… 15
2.4　电源 …… 15
2.5　I/O 模块 …… 16
2.5.1　数字 I/O 模块 …… 17
2.5.2　模拟 I/O 模块 …… 24
2.5.3　智能 I/O 模块 …… 27
2.6　编程器 …… 29
2.7　故障检修 …… 30
习题 …… 30
第 3 章　二进制逻辑编程（布尔逻辑） … 32
3.1　学习目标 …… 32
3.2　按位操作的梯形图 …… 33
3.2.1　梯形图元素 …… 34
3.2.2　创建梯形图程序 …… 35
3.3　按位操作的指令表程序 …… 37
3.3.1　指令表布尔逻辑元素（STL） … 38
3.3.2　创建指令表程序（STL） …… 38
3.4　一些常见的二进制逻辑编程技巧 …… 40
3.4.1　一次翻转法 …… 40
3.4.2　锁定和封装法 …… 40
3.4.3　顺序器法 …… 42
3.5　故障检修 …… 43
习题 …… 44
编程练习（不需要 PLC） …… 45
推荐的 PLC 实验室练习 …… 46
第 4 章　计数器和定时器 …… 48
4.1　学习目标 …… 48
4.2　计数器指令 …… 49
4.3　定时器指令 …… 49
4.3.1　Allen-Bradley 计数器和定时器 …… 50
4.3.2　SIEMENS S5 计数器和定时器 … 53
4.3.3　SIEMENS STEP 7 计数器和定时器 …… 57
4.3.4　OMRON CQM1 计数器和定时器 …… 58
4.4　故障检修 …… 59
习题 …… 60
编程练习（不需要 PLC） …… 61
推荐的 PLC 实验室练习 …… 61
第 5 章　存储器组织和数据操作 …… 62
5.1　学习目标 …… 62
5.2　存储器概述 …… 62
5.3　数据类型 …… 63
5.4　寻址方式 …… 64
5.5　PLC 中可寻址的数据存储 …… 65
5.5.1　ALLEN-BRADLEY PLC 中的数据文件和可寻址数据 …… 65
5.5.2　ALLEN BRADLEY PLC-5 中的

可寻址数据 …… 72
5.5.3 ALLEN-BRADLEY SLC 500 中的可寻址数据 …… 75
5.5.4 SIEMENS STEP 5 中的可寻址数据 …… 76
5.5.5 Siemens STEP 7 中的可寻址数据 …… 85
5.5.6 OMRON CQM1 中的数据寄存器和可寻址数据 …… 102
5.6 故障检修 …… 105
习题 …… 106
第 6 章 操作简单数据元素 …… 109
6.1 学习目标 …… 109
6.2 微处理器基础 …… 109
6.3 数据操作指令 …… 110
6.4 简单数据元素 …… 111
6.4.1 简单数据元素的移动 …… 111
6.4.2 简单数据元素的比较 …… 117
6.4.3 简单数据元素的数学、逻辑和转换操作 …… 121
6.5 故障检修 …… 133
习题 …… 135
推荐的 PLC 实验练习 …… 137
第 7 章 文件、块、数组和结构体中的数据处理 …… 139
7.1 学习目标 …… 139
7.2 文件、块、数组和结构体定义 …… 140
7.2.1 ALLEN-BRADLEY 的数据文件 …… 140
7.2.2 SIEMENS STEP 5 的数据块 … 140
7.2.3 SIEMENS STEP 7 的数据块、数组和结构体 …… 141
7.2.4 OMRON CQM1 的数据集合 … 141
7.3 位数组和移位指令 …… 141
7.3.1 ALLEN-BRADLEY 的移位指令 …… 142
7.3.2 SIEMENS STEP 5 的移位指令 …… 144
7.3.3 SIEMENS STEP 7 的移位指令 …… 144
7.3.4 OMRON CQM1 的移位指令 … 145
7.4 数组移位指令（包括 FIFO 及 LIFO） …… 146
7.4.1 ALLEN-BRADLEY 的 FIFO 及 LIFO 指令 …… 146
7.4.2 SIEMENS STEP5 的数组移位指令 …… 148
7.4.3 SIEMENS STEP 7 的数组移位指令 …… 148
7.4.4 OMRON CQM1 的数组移位指令（包括 FIFO 和 LIFO） …… 148
7.5 文件、数组和结构体的移动 …… 151
7.5.1 ALLEN-BRADLEY 的文件移动、顺序器和块传送指令 …… 151
7.5.2 SLC 500 专用：顺序器差分和交换指令 …… 153
7.5.3 PLC-5 专用：块传送指令 …… 153
7.5.4 SIEMENS STEP 5 数据集的移动指令（带有传送和接收功能块） …… 154
7.5.5 SIEMENS STEP 7 数据集的移动（使用系统函数） …… 157
7.5.6 OMRON CQM1 文件、数组和结构体移动指令 …… 160
7.6 文件、数组和结构体的比较 …… 162
7.6.1 ALLEN-BRADLEY 文件比较指令 …… 162
7.6.2 PLC-5 的文件搜索与比较（FSC）、文件位比较（FBC）以及诊断检测（DDT） …… 163
7.6.3 SIEMENS 文件、数组和结构体比较指令 …… 165
7.6.4 OMRON CQM1 文件、数组和结构体比较指令 …… 165
7.7 文件、数组和结构体的数学及逻辑指令 …… 166
7.7.1 ALLEN-BRADLEY 的文件数学及逻辑指令 …… 166
7.7.2 SIEMENS 文件、数组和结构体的数学及逻辑指令 …… 167

7.7.3 OMRON CQM1 文件、数组和结构体的数学及逻辑指令 …… 167
7.8 故障检修 …… 169
习题 …… 170
推荐的 PLC 实验室练习 …… 171
第 8 章 程序结构和结构化编程 …… 173
8.1 学习目标 …… 173
8.2 在单独的程序中影响执行的指令 … 174
8.2.1 主控继电器 …… 174
8.2.2 跳转指令 …… 179
8.2.3 循环 …… 184
8.3 在程序扫描的过程中影响子程序或函数执行的指令 …… 185
8.3.1 ALLEN-BRADLEY 的子程序调用 …… 186
8.3.2 SIEMENS STEP 5 的函数调用 …… 188
8.3.3 Siemens STEP 7 的函数调用 … 191
8.3.4 OMRON CQM1 的子程序调用 …… 202
8.4 影响程序执行的配置 …… 202
8.4.1 对 ALLEN-BRADLEY PLC-5 中结构化编程配置 …… 203
8.4.2 ALLEN-BRADLEY SLC 500 的结构化编程配置 …… 204
8.4.3 Siemens STEP 5 的结构化编程配置 …… 204
8.4.4 Siemens STEP 7 的结构化编程配置 …… 205
8.4.5 OMRON CQM1 中结构化编程配置 …… 206
8.5 故障检修 …… 206
8.5.1 主控继电器的故障检修 …… 207
8.5.2 跳转和循环指令的故障检修 …… 207
8.5.3 子程序调用的故障检修 …… 207
8.5.4 允许离开扫描循环的程序配置的故障检修 …… 208
习题 …… 208
推荐的 PLC 实验室练习 …… 211
第 9 章 IEC 1131-3：通用编程语言 …… 213
9.1 学习目标 …… 213
9.2 IEC 1131 概述 …… 213
9.3 IEC 1131-3 编程语言 …… 214
9.4 IEC 1131-3 结构化程序的通用元素 …… 215
9.4.1 算法和数据类型 …… 215
9.4.2 配置 …… 215
9.4.3 资源 …… 216
9.4.4 任务 …… 217
9.4.5 程序 …… 217
9.4.6 功能块 …… 219
9.5 程序组织单元 …… 219
9.5.1 程序 …… 220
9.5.2 函数 …… 220
9.5.3 功能块 …… 223
9.6 变量和变量声明 …… 224
9.6.1 配置层的变量声明 …… 224
9.6.2 资源层的变量声明 …… 227
9.6.3 程序层的变量声明 …… 228
9.6.4 功能块层的变量声明 …… 230
9.6.5 函数层的变量声明 …… 231
9.7 IEC 1131-3 的编程语言 …… 231
9.7.1 梯形图 …… 232
9.7.2 指令表 …… 234
9.7.3 结构文本 …… 235
9.7.4 顺序功能图 …… 236
9.7.5 功能块图表 …… 239
9.7.6 连续功能图 …… 241
9.8 总结 …… 242
9.9 故障检修 …… 243
参考文献 …… 244
习题 …… 244
第 10 章 PLC 的设置和配置 …… 245
10.1 学习目标 …… 245
10.2 安装和配置新的 PLC …… 245
10.3 安装硬件 …… 245
10.3.1 ALLEN-BRADLEY PLC-5

的硬件安装 …… 246
10.3.2 ALLEN-BRADLEY SLC 500 硬件安装 …… 248
10.3.3 SIEMENS S5 硬件安装 …… 249
10.3.4 SIEMENS S7 硬件安装 …… 251
10.3.5 OMRON CQM1 的硬件安装 …… 254
10.4 为一个应用准备的 PLC 系统的第一次配置 …… 256
10.4.1 Allen- Bradley PLC-5 的第一次配置 …… 256
10.4.2 ALLEN- BRADLEY SLC 500 的第一次配置 …… 265
10.4.3 SIMENS S5 的第一次配置 …… 269
10.4.4 SIMENS S7 第一次的配置 …… 276
10.4.5 OMRON CQM1 的第一次配置 …… 285
10.5 在 PLC 程序重启过程中重新配置 …… 289
10.5.1 ALLEN-BRADLEY 重启配置 …… 290
10.5.2 SIEMENS S5 重启配置 …… 290
10.5.3 SIEMENS S7 重启配置 …… 290
10.5.4 OMRON CQM1 重启配置 …… 292
10.6 故障检修 …… 293
习题 …… 295
第 11 章 中断 …… 297
11.1 学习目标 …… 297
11.2 问题 …… 297
11.3 中断的解决方案 …… 298
11.4 关于中断响应的更多细节描述 …… 300
11.4.1 立即输入和立即输出指令 …… 301
11.4.2 I/O 中断 …… 306
11.4.3 定时中断 …… 324
11.4.4 出错程序中断 …… 338
11.4.5 初始化中断 …… 346
11.4.6 通信中断 …… 349
11.5 总结 …… 351
11.6 故障检修 …… 352
习题 …… 354
推荐的 PLC 实验室练习 …… 356
第 12 章 过程控制 …… 357
12.1 学习目标 …… 357
12.2 过程控制导言 …… 357
12.3 PLC 在过程控制中的应用 …… 359
12.4 改进 PLC 程序在过程控制中的性能 …… 361
12.4.1 过程变量和控制变量的标度变换 …… 362
12.4.2 对过程变量进行限幅 …… 363
12.4.3 减少过程控制的扫描时间延时 …… 364
12.4.4 定时中断 …… 368
12.4.5 输出量计算中的其他量 …… 371
12.4.6 复杂的过程控制程序 …… 374
12.4.7 系统的手动控制 …… 420
12.4.8 根据检测到的情况的不同而选用不同的计算控制输出方法 …… 420
12.5 故障检修 …… 423
习题 …… 424
推荐的 PLC 实验室练习 …… 425
第 13 章 通信 …… 427
13.1 学习目标 …… 427
13.2 PLC 的通信能力 …… 427
13.3 ALLEN-BRADLEY PLC 的通信 … 429
13.3.1 ALLEN-BRADLEY PLC-5 的通信 …… 431
13.3.2 ALLEN-BRADLEY SLC 500 的通信 …… 436
13.4 使用 PROFIBUS 的 SIEMENS 的 PLC 通信 …… 442
13.4.1 使用 SIEMENS S5 的通信 …… 443
13.4.2 使用 SIEMENS S7 PLC 的通信 …… 447
13.5 OMRON CQM1 的通信 …… 454
13.5.1 CQM1 通信通道的配置 …… 456
13.5.2 对 CQM1 编程实现通信 …… 457

13.6 故障检修 …… 459
习题 …… 460
推荐的 PLC 实验室练习 …… 461
第 14 章 机器人技术、自动化和 PLC …… 462
14.1 学习目标 …… 462
14.2 车间里的机器人和 PLC …… 462
14.3 机器人控制器与 PLC 的不同 …… 462
14.4 机器人控制器与 PLC 的相似点 …… 464
14.5 编程使机器人和 PLC 共同工作 …… 464
14.6 机器人程序 …… 466
14.7 PLC 程序 …… 468
第 15 章 故障检修 …… 472
15.1 学习目标 …… 472
15.2 系统方法 …… 472
15.3 PLC 外围硬件的故障检修 …… 472
15.4 PLC 硬件、配置及编程的故障检修 …… 474
15.5 ALLEN-BRADLEY PLC-5 的故障检修 …… 475
15.5.1 PLC-5 的硬件故障检修 …… 475
15.5.2 PLC-5 启动出错检测的配置 … 475
15.5.3 PLC-5 硬件状态 …… 475
15.5.4 PLC-5 通信通道状态 …… 477
15.5.5 PLC-5 的 CPU 状态 …… 477
15.5.6 Allen-Bradley 的主要出错及次要出错 …… 478
15.5.7 PLC-5 程序故障检修 …… 480
15.5.8 PLC-5 编程器在调试中的特性 …… 482
15.6 Allen-Bradley SLC 500 的故障检修 …… 482
15.6.1 SLC 500 硬件故障检修 …… 482
15.6.2 配置 SLC 500 启动时的出错检测 …… 482
15.6.3 SLC 500 硬件状态 …… 483
15.6.4 SLC 500 主要及次要出错 …… 483
15.6.5 SLC 500 程序故障检修 …… 483
15.7 SIEMENS S5 的故障检修 …… 484
15.7.1 S5 致命及非致命错误 …… 484
15.7.2 STEP 5 逻辑错误调试工具 … 489
15.7.3 普通编程错误 …… 490
15.7.4 STEP 5 编程器的调试特性 …… 490
15.8 SIEMENS S7 的故障检修 …… 490
15.8.1 STEP 7 的编程器问题 …… 491
15.8.2 S7 硬件故障检修 …… 491
15.8.3 S7 的故障检修配置 …… 491
15.8.4 S7 状态信息 …… 492
15.8.5 出错响应组织块 …… 496
15.8.6 使用编程器观察的 S7 诊断信息 …… 496
15.8.7 STEP 7 逻辑错误调试工具 … 497
15.9 OMRON CQM1 的故障检修 …… 498
15.9.1 CQM1 故障检修配置 …… 498
15.9.2 CQM1 致命和非致命错误 …… 499
15.9.3 CQM1 逻辑错误调试工具 …… 501
15.10 总结 …… 504
习题 …… 505
第 16 章 未来：PLC 前途是否黯淡 …… 506
16.1 学习目标 …… 506
16.2 明天的 PLC …… 506
16.3 现场总线和传感器——执行器网络 …… 507
16.4 SCADA 系统 …… 509
16.5 软件逻辑 …… 510
16.6 过程仿真 …… 511
16.7 反射性存储器 …… 511
16.8 OMAC 运动与过程控制 …… 512
习题 …… 512
附录 A Allen-Bradley PLC-5 状态文件结构体 …… 513
附录 B Allen-Bradley SLC 500 状态文件结构体 …… 516
附录 C OMRON CQM1 SR 和 AR 存储区域 …… 518

附录 D Allen-Bradley 比较指令算子 …… 526
附录 E Allen-Bradley 计算指令算子和优先权 …… 527
附录 F Siemens S7 被数学和逻辑操作影响的状态位 …… 529
附录 G Siemens S7 系统函数（FC）、系统功能块（SFB）和 IEC 函数（FC） …… 531
附录 H Allen-Bradley PLC-5 主要和次要出错位及代码 …… 536
附录 I ALLEN-BRADLEY SLC 500 主要出错代码 …… 541
附录 J ALLEN-BRADLEY PLC-5 PID 控制块 …… 548

第1章　什么是PLC

1.1　学习目标

本章您将了解到：

- PLC与其他计算机相区别的硬件及操作特性；
- 梯形图；
- 如何选择PLC；
- PLC的简要发展历史。

可编程逻辑控制器（PLC）是一种为工业控制特别设计的专用计算机。因为它易于设置和编程，运行可预估，甚至在恶劣的生产环境下还可以保持正常工作，所以广泛应用于工业控制。

可编程逻辑控制器有时又被称作可编程控制器，但PLC的称呼更为常见。PLC看上去与标准的个人计算机不同，最明显的是PLC没有键盘和显示器。图1-1展示了几款PLC（本书将详细介绍）。

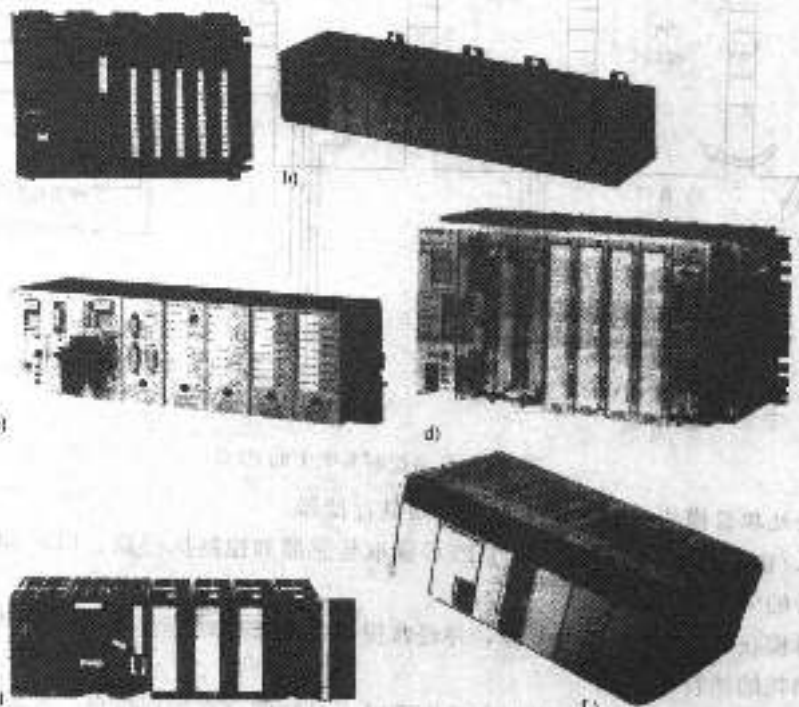


图1-1　可编程逻辑控制器：Rockwell的a) PLC-5和b) SLC 500；Siemens AG的c) S5-100U、d) S5-115U和e) S7-300；OMRON的f) CQM1

1.2 PLC 基础

外观只是 PLC 与其他计算机之间区别的一部分。PLC 区别于大多数计算机有两个主要方面：1）PLC 的结构使用户易于组建一个 PLC 控制系统；2）PLC 可以通过操作系统预编程序，并通过应用程序使之优化并应用于工业控制。

1.2.1 PLC 的结构

有的 PLC 被集成为一个独立的单元，而另一些却是模块化的。因为尺寸小，集成 PLC 有时称作鞋盒或者砖块 PLC。如果集成 PLC 具备用户所需的性能，那么它通常是最经济的选择。模块化 PLC 由可选择的组件组成。这些组件可由用户选择和装配，使 PLC 可以应用于更复杂的控制系统。

购买模块化 PLC 时需要单独购买的 PLC 组件将在第 2 章详细说明。它们包括以下组件，如图 1-2 所示：

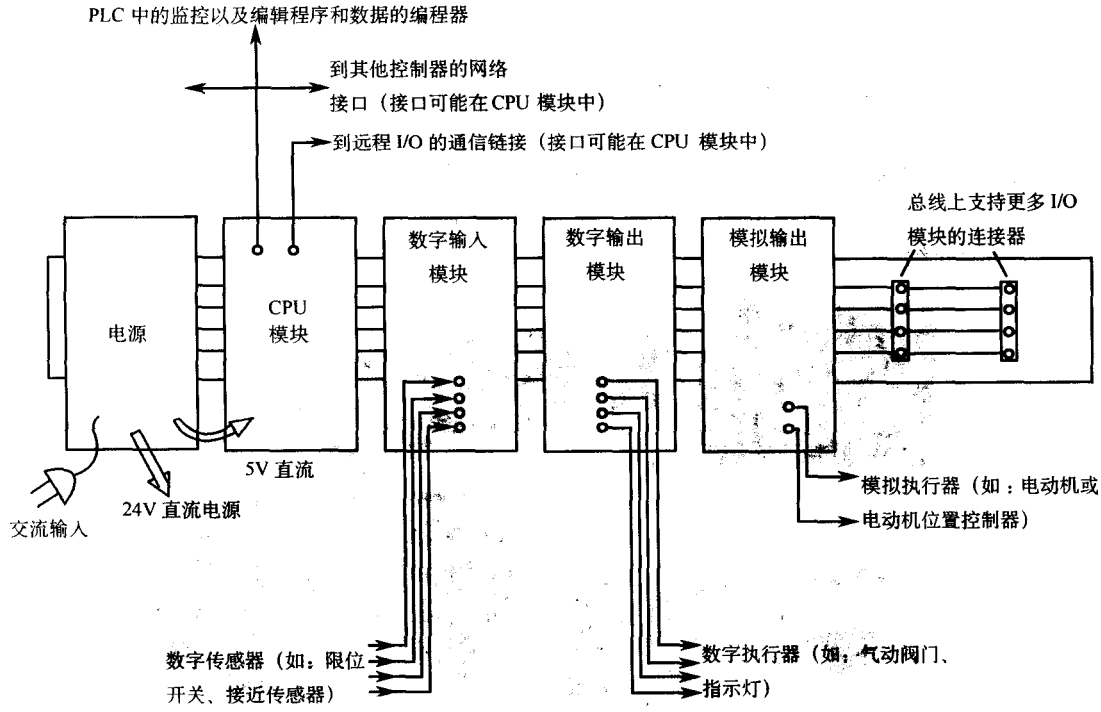


图 1-2 自动控制系统中的 PLC

- 中央处理器模块，包含中央计算机及其存储器。
- 输入/输出（I/O）模块，允许 PLC 读取传感器和控制执行器。PLC 制造商提供各种型号的 I/O 模块。
- 电源模块，给 CPU 提供电源，并经常提供电源来驱动连接在 I/O 模块上的传感器和低功耗的执行器。
- 框架或总线，CPU 模块和 I/O 模块可以通过它们交换数据信息。在有些 PLC 中，这部分不是必需的，因为每个模块直接插入它的邻近模块。

包含这些组件的 PLC 系统足以控制一个自动化系统。如果 PLC 必须在使用前编程，则

需要其他组件：

- 编程器，建立用户程序并将其传送至 PLC CPU 模块的存储器内，这个是必需的。

其他可选 PLC 组件包括：

- 远程 I/O 的通信适配器，通过它中央控制器可以连接远程的传感器和执行器。
- 网络接口，允许 PLC 和其他控制器相互连接组成分布式控制系统。
- 操作员界面设备，允许操作员输入和监控数据。

1.2.2 操作系统和应用程序

与大多数其他计算机中的 CPU 相比，PLC 的 CPU 模块具有非常不同的操作系统，PLC 的应用程序可以预先编程并直接存储到 CPU 的存储器中。电源接通后，操作系统启动 PLC；当 PLC 转换到运行模式时，运行用户程序；通过运行适合的应用程序，响应用户命令。该应用程序允许用户把程序和数据发送至 PLC 的存储器。这种用户可以访问的存储器的某一部分在 PLC 断电后也不会丢失数据。

> 不像 PLC，标准的个人计算机在其存储器里只有很少量的预编程的操作系统。断开电源后，它只有很少甚至没有存储器来保存数据。当计算机启动时，个人计算机的操作系统先进行简单的自检，然后必须从硬盘加载附加的操作系统程序（例如 DOS，Windiows 或者 UNIX）至存储器。直到应用程序（例如字处理程序、CAD 程序或程序设计语言如 BASIC）也加载完毕，个人计算机才可正常工作。当电源断开时，少量能够保持用户数据的存储器单元只能记录计算机配置信息（如硬件驱动安装的类型）。

在 PLC 关闭甚至没有连接电源时，PLC 仍在保持性存储器中（有时又称作非易失性存储器）保留它的操作系统、应用程序、用户程序和一些数据。因此，尽管 PLC 经常设计为在重启前需要操作员操作（出于安全考虑），一旦电源恢复 PLC 就可重新运行用户程序。

PLC 的操作系统使 PLC 运行用户程序的方式同其他计算机的运行方式不同。PLC 操作系统每次进入运行模式时执行初始化一次，然后只要 PLC 保持在运行模式就重复使 PLC 执行顺序扫描循环。图 1-3 显示了所有的 PLC 内部的基本扫描循环。虽然在不同的 PLC 中存在一些细微差别，尤其是初始化过程以及生产商用来描述扫描循环的术语存在着差别，但是三步扫描循环都是 PLC 控制自动化系统的基础。如图 1-3 所示，PLC 的操作系统使 PLC 执行如下步骤：

1）预编程的初始化步骤，在每次 PLC 进入运行模式，且在三步重复扫描循环步骤第一次执行前，此步骤只执行一次。

2）重复的三步扫描循环，包括[⊖]

（a）输入扫描。PLC 从所有的输入模块读取数据（从连接到输入模块的传感器获取数据）。这些输入数据存放在保留给输入数据映像的 CPU 模块的存储器中。

（b）用户程序扫描。从开始到结尾，用户编写的控制程序只运行一次。这些程序包含检测输入映像数据的指令和决定 PLC 需要输出什么值到执行器。PLC 不会写输出数据到输出模块，而是将它们储存在 CPU 的 RAM 存储器中保留给输出数据映像的空间。用户编写的程序可检测和改变所有 RAM 存储器中可寻址的区域。（这意味着输入映像数据可以被用户

⊖ 在这个循环中，功能更强大的 PLC 可能会执行附加的步骤。一些附加的步骤可以预编程作为操作系统的一部分，通常用来控制串口通信。一些 PLC 可以由用户编程或配置而进行附加的操作。

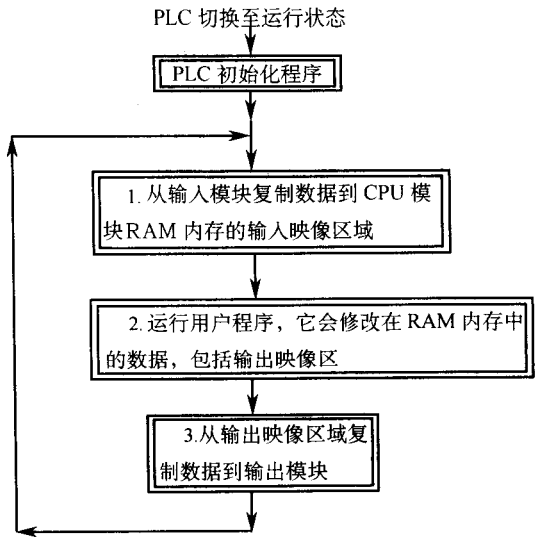


图 1-3 标准 PLC 扫描循环

程序改变并且输出映像数据可以被检测。）一些 RAM 存储器不可寻址，所以它不可以被用户程序改变。例如，用户程序不在可寻址存储器中。

（c）输出扫描。在这一步中，PLC 从 CPU 的 RAM 的输出映像复制所有数据到输出模块中。

每次 PLC 结束一个扫描循环并开始另一个时，操作系统也重新启动一个*看门狗定时器*。看门狗定时器在扫描循环执行时运行。如果看门狗定时器在重新启动前达到其预设值（如果完成一个扫描循环花费的时间超出正常很多），PLC 会立即进入出错状态并停止运行。出错后 PLC 需要操作员介入才可继续运行。在故障检修中起作用的操作系统诊断程序将会在第 15 章中介绍。

PLC 操作系统的其他部分提供了用户认为的任何计算机都应该具备的性能，但它对整个 PLC 控制系统的质量有着显著影响。在选择 PLC 前往往要确定操作系统的应用程序和驱动程序的质量。

所有 PLC 都带有预编程的*应用程序*。通过运行应用程序来响应从编程器、操作面板或连接到 PLC 的其他计算机发到 PLC 的命令。应用程序允许用户写入或保存程序和数据到 PLC 的读写存储器中，且允许用户指示 PLC 运行程序以及发送状态信息到操作员界面终端，允许监控程序执行和监控 PLC 存储器的数据。

*驱动程序*是其他程序（扫描循环程序和部分用户程序）调用来操作 CPU 模块内的 I/O 控制电路的子程序。驱动程序是由 PLC 生产商在只读存储器 ROM 上预先编程的。需要驱动程序的 I/O 功能包括：

1）连接到编程器。所有 CPU 模块都包含驱动器和通信硬件，让程序员通过编程器监控和更改用户程序及工作数据。出于这个目的，编程器通常与专用的 PLC 串行端口连接。

2）扫描循环时本地 I/O 模块数据的读写。所有模块化 PLC 的 CPU 模块包含与当前的 I/O 模块交换数据的驱动软件和硬件，当前的 I/O 模块即指通过框架或总线里并联的导线直接与 CPU 模块连接的 I/O 模块。

3）接收和发送远程 I/O 数据。远程 I/O 站点可能是 PLC 控制的 I/O 模块的分离的框架，可能是没有连接到框架的独立的 I/O 模块，甚至也可能是带嵌入式通信适配器的独立传感器和执行器。PLC 的 CPU 模块可能包含驱动器和接口硬件，从而允许 CPU 模块内远程 I/O 站点通过串行的通信链接来交换数据。如果 CPU 模块没有嵌入远程 I/O 通信驱动器和硬件，则对远离主 PLC 的过程进行控制必须有分离的通信模块。

4）接收和发送扩展的 I/O 数据。扩展 I/O 站点是由 PLC 控制的单独框架上的 I/O 模块，但它架设在主框架附近，因此与远程 I/O 通信相比，利用不同驱动程序和硬件进行的数据交换会更快。

5）在局域网（LAN）里接收和发送数据。如果 PLC 有适当的驱动器和硬件，CPU 模块可以通过共享的串行的通信链接直接与其他控制器系统（包括其他 PLC）连接。因此 PLC 的用户程序包括数据的编程交换。一些 PLC 只有一个通信端口，但是有可选择的驱动器，因此用户可以选择将 CPU 模块连接到局域网。编程器和操作面板有时通过局域网端口连接到 PLC，并且有时可以与 CPU 共享使用局域网。如果编程器以这种方式通过局域网端口连接，则可以利用编程器去监控和更改所有在局域网上的 PLC 的 CPU 模块的存储器。一些 CPU 模块有多个通信端口。如果 CPU 没有必需的局域网驱动器或硬件，则可以利用一个独立的通信 I/O 模块。第三方的厂家出售接口硬件和驱动软件以便连接多数流行的局域网类型，因此你可以连接西门子的 PLC 到 Allen-Bradley 的局域网。

6）从设备（例如串行打印机、电脑条码阅读机）读写。通过标准通信协议如 RS-232 的串行链接来完成此任务。

1.2.3 PLC 用户程序

用户程序不属于购买 PLC 时已预编的程序。它们必须由程序员通过编程器输入 PLC 的 RAM 存储器内，之后编程器可以与 PLC 断开。PLC 在存储器中保存用户程序，它既不会受到断电的影响，也无需一个长效电池来维持。用户程序保存在存储器里直到用编程器更改它为止。

PLC 的用户程序常用梯形图来编写。梯形图程序是基于图形化编程的。它们看起来有一点像在绘制折梯的图形，也有点像工业电工使用的继电器逻辑电路图。图 1-4 显示了一个简单的梯形图程序及其控制的系统。梯形图程序中的每个梯级由一个可以表示真或伪的逻辑条件组成，它可以控制梯级的输出功能是否执行。例如，如图 1-4 所示，程序的第一个梯级控制喷漆执行装置。只要系统控制开关打开且传感器同时检测到需要油漆的零件，喷漆器就会打开。如果油漆用完或者传送带因为某些原因停止了，第二个梯级会打开警示灯通知操作员。如果你不能完全明白程序如何工作，不用担心，第 3 章将详细介绍梯形图程序是如何工作的。

PLC 以毫秒为单位的间隔重复执行它的扫描循环，包含用户程序。通过图 1-3 所示的扫描循环流程图，很明显可以看出在传感器检测到变化和执行器做出响应之间有一个短的延时。最坏的情况，如果在一个扫描循环内，例子中零件感应器在它的值被读取后立即检测到零件，则喷漆执行器不会打开，直到下一个扫描循环结束。因为 PLC 执行它的扫描循环的间隔只有几个毫秒，所以延时通常不成问题。

程序员应该始终记住用户程序检测到的状态只是输入状态的最近映像。同样，用户程序没有改变实际输出，只是改变最终会写入输出端的映像。还要记住，用户程序的前一个梯级可能已经改变这个梯级使用的输出或输入映像值。经验不足的程序员所犯的一个常见的编程

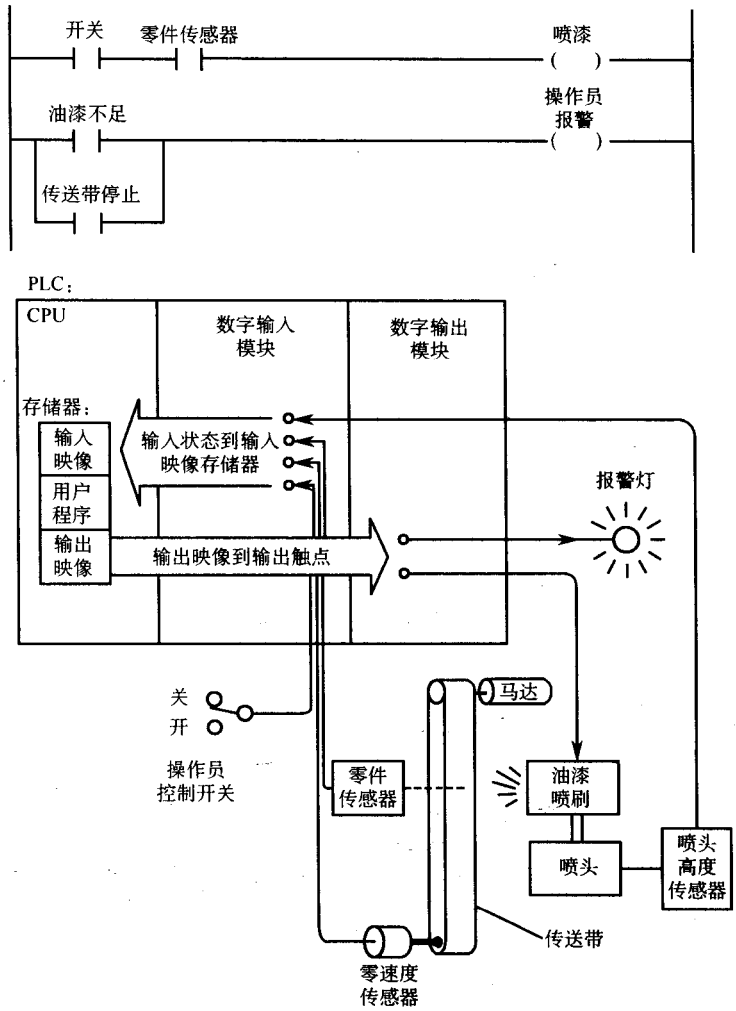


图 1-4 两梯级梯形图

错误是：设计两个独立的梯级来控制相同的输出映像。第二个梯级总是会在输出映像实际复制到输出模块前覆盖第一个指令的结果。

如果编程正确，图 1-4 的程序会保证经过的每个零件都涂好，在零件到达与喷漆器动作之间没有可察觉的延时。如果不需要就不会喷漆。程序会一直控制喷漆器，只有在 PLC 检测到已编程要求监控的情况下，才需要操作员注意。这套基于 PLC 的自动系统可以很快向会计师证明购买它是值得的。

1.3 选择合适的 PLC

PLC 有各种各样的类型和性能，要根据任务选择合适的 PLC。大部分组织开始以一个集成 PLC 作为控制器来实现他们的第一个简单的自动化系统。Allen-Bradley 的 Micrologix 1000，Siemens 的 S7-200 或 OMRON 的 CPM1（如图 1-5 所示）都是典型的小型但是功能强大的集成 PLC。大部分此类 PLC 都有内置 24V 的电源可以去驱动典型传感器。内置电源只能驱动最小

的执行器，所以需要由 PLC 控制的独立的电源去驱动执行器。用户还需要编程器。PLC 生产商出售可以下载到个人计算机的编程软件，个人计算机可作为编程器，用导线来连接个人计算机和 PLC。PLC、编程软件和连接线的总价值比一台计算机要少得多。Allen-Bradley 的 Micrologix 1000和较大一些的 SLC 500 PLC 可以用相同的软件编程；Siemens 的 S7-200 和 S7-300 可用相同的软件；而 OMRON 的 CPM1 和 CQM1 可用相同的软件。

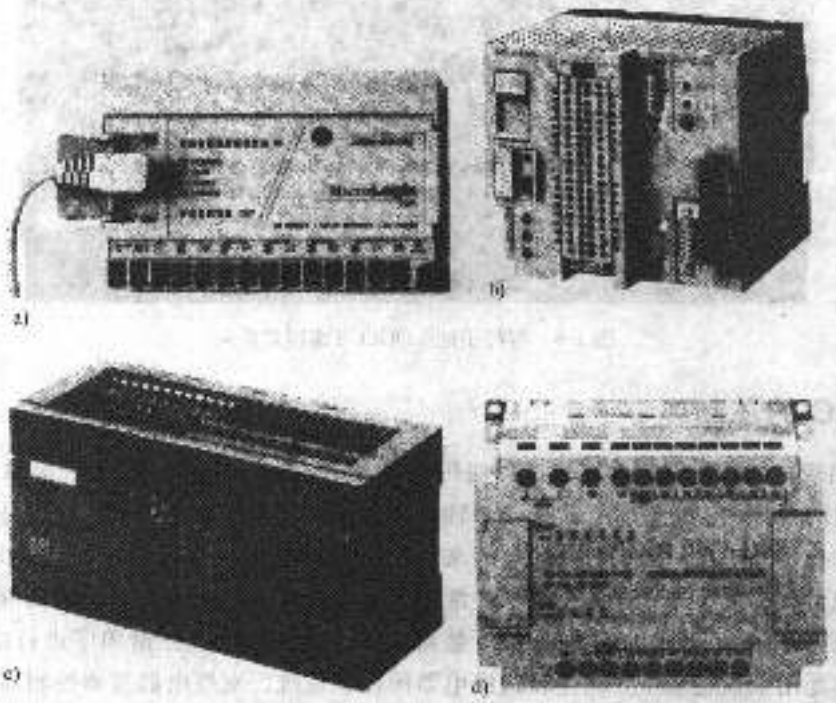


图 1-5 小型 PLC：a) Rockwell 的 MicroLogix 1000；b) Siemens 的 b) S5-95U 和 c) S7-200；OMRON 的 d) CPM1

上述简单的 PLC 能够控制大约有 20 个开关传感器和执行器的系统。一些集成的 PLC 甚至通过一个或两个输出端子提供模拟输出信号，或通过一个或两个模拟传感器或编码器接收信号，它们主要还是用于传感器信号和执行器（状态为开或关）是数字式的系统。这些小型 PLC 使用的编程语言与本书介绍的语言相同，尽管这里介绍的一些指令可能不能用到。

如果系统需要一个只有很少输出和输入的、全数字化的控制器，那么购买一个更小型的可编程逻辑模块就更有意义，如 Siemens 的 LOGO 逻辑模块，如图 1-6 所示。用户的购买订单还可以包括西门子公司在装配时写进逻辑模块的程序，所以这种 PLC 控制器不仅便宜，用户甚至不需要购买编程器。

如果上述的小型 PLC 不能满足需要，用户可购买模块化 PLC（第 2 章将详细介绍）。购买模块化 PLC 时，购买者只要购买需要的接口模块使 PLC 连接到数字传感器和（或）执行器、非数字传感器和（或）执行器，或其他的计算机控制器。

一些制造商提供软件和接口卡使个人计算机可作 PLC 使用。这些系统被称作软逻辑控制器。软逻辑控制器也要与传感器和执行器相连，所以需要能够与软件逻辑模块通信的 I/O 模块。软逻辑控制器会在第 16 章讨论。

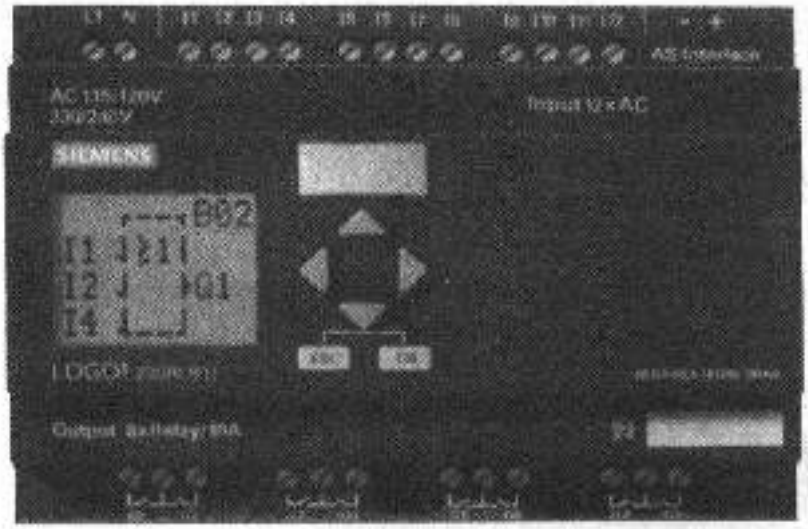


图 1-6 西门子的 LOGO 可编程逻辑块

1.4 PLC 的革新

PLC 出现之前，自动化的制造过程使用硬件设备控制。例如，旋转凸轮轴可以保证在生产循环里的事件相继地发生，就像凸轮轴还用来保证汽车引擎正确运行，但是，凸轮轴需要凸轮轴和控制操作之间的机械连接。后来，控制系统利用继电器去控制非常分散的系统。一些电路包含控制继电器开关的传感器。继电器通过控制其他电路的电流来控制电气执行器或其他的继电器。通过系统的定时器和计数器，继电器系统可以控制顺序进行的制造过程。继电器逻辑是用来描述基于互相连接的继电器的控制系统。对继电器逻辑控制器编程可以创建继电器逻辑控制系统。要想修改继电器逻辑程序，则控制系统需要重新创建。调试错误的继电器逻辑控制系统要求对电气元件和接触点进行繁琐的电气故障检修。

计算机首先以顺序器（sequencer）的形式用作工业控制器。为了对顺序器编程，程序员输入一系列二进制数模式（数据字）到计算机存储器。顺序器通过输出二进制字控制制造过程，每次一个字。数据字的每个二进制位控制一个执行器，所以当一个输出数据字被下一个代替时，一些执行器会启动而另一些会关闭。有两种控制输出字更改的方式：

1）在基于时间的顺序器中，程序员指定每一步的延时时间。

2）在基于事件的顺序器中，如图 1-7 所示，程序员为每个输出数据字输入一个输入数据字。在输出一个数据字后，顺序器比较输入数据字及相应的传感器的开关状态模式。当输入模式与输入数据字相匹配时，则输出下一个输出数据字。（一个错误输入会导致基于事件的顺序器完全失效。）

存在混合基于时间和事件的顺序器。一些现代的 PLC 提供顺序器指令，所以 PLC 程序可包括顺序的控制。顺序器指令会在第 7 章介绍数据文件操作时涉及。

继电器逻辑控制器在很多方面比顺序器更好：通过多重并联电路，每个控制电路独立工作，因此一个单独的错误不会引起整个控制系统瘫痪。而且，所有多重并联电路同时执行它们的控制职责，如有需要，每一个都可实时改变其输出。继电器逻辑控制器可实时操作，只要有需要，它们就提供控制信号。顺序器不是实时控制器。

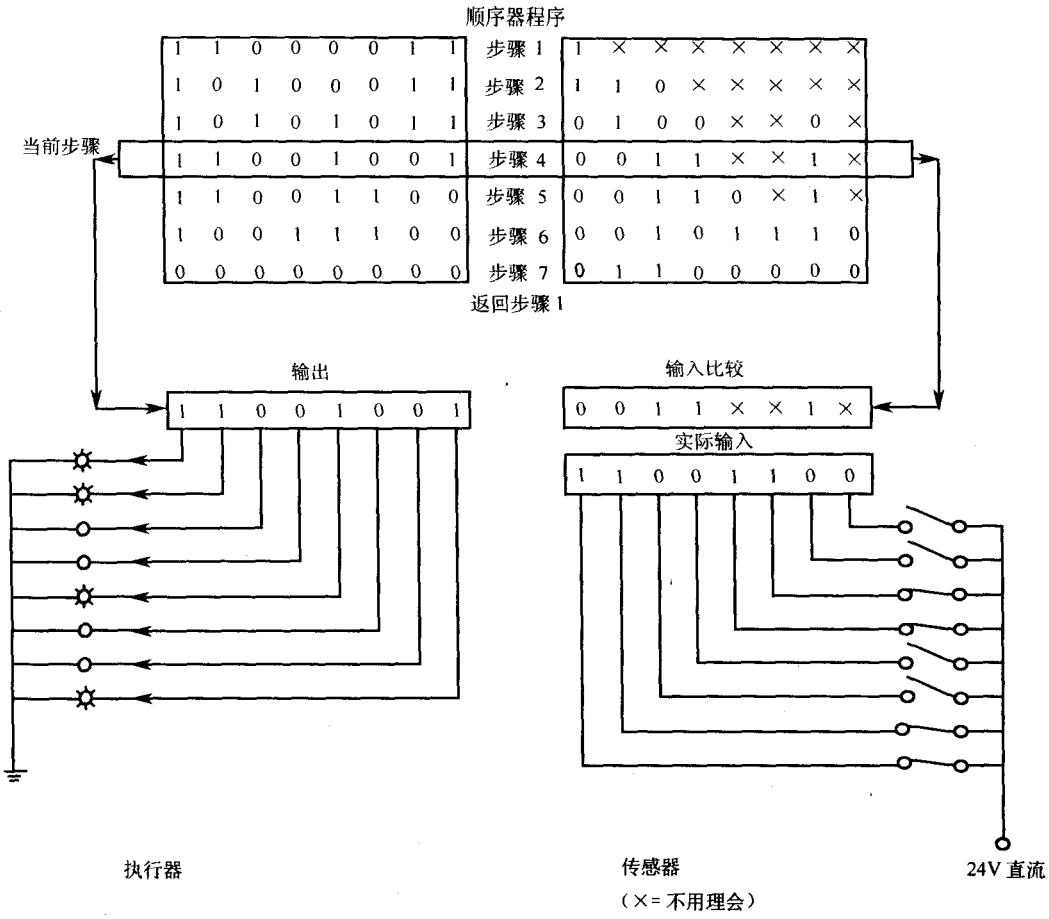


图 1-7 基于事件的顺序器

最初的 PLC 称作可编程控制器，是设计在尽可能接近实时模式下操作的早期计算机。因为计算机同一时间只能做一件事，按编程时指定的顺序，早期的可编程控制器需要操作系统使它看起来好像同时控制很多独立的过程。早期的可编程控制器操作系统采用三步扫描循环。首先，可编程控制器将所有输入触点的状态读入至存储器的输入映像区，这有利于快速了解输入情况。然后，可编程控制器运行用户程序，用户程序由梯形图梯级组成，每一级独立地控制存储器的输出映像区中的单个位的状态，就像每个并联的继电器逻辑线路控制一个执行器一样。（梯形图程序甚至设计得像继电器逻辑线路图，因此工业电工可以编写可编程控制器程序。）最后，执行完用户程序后，可编程控制器立即复制输出映像区的内容到输出触点，使执行器开或关。这种控制不是真正的实时，因为在意识到需要改变输出到可编程控制器使输出改变之间总有延时，但是读写大块的输入输出信息时的效率使可编程控制器操作足够快，从而延时很短。扫描循环也使控制具有确定性，即可编程控制器的响应是可预料的和可靠的。

可编程控制器在 20 世纪 80 年代被迫改名，可编程控制器的缩写是 PC，为了与个人计算机缩写 PC 相区别，可编程控制器现在称作可编程逻辑控制器，缩写是 PLC。

PLC 还有其他革新。最早的 PLC 只能用布尔梯形逻辑元素来检查和控制输入输出映像数据的开关状态（布尔逻辑会在第 3 章介绍）。之后不久，新一代的 PLC 除了输入输出映像

存储器，还提供工作数据存储器与可编程定时器和计数器指令。可编程定时器和计数器使用存储器的整个数据字去存储计数值和计时值。在许多应用场合，PLC 并不要求具有比计数器、定时器、工作数据的存储更高的功能。第 4 章将介绍定时器和计数器。

一个改进不可避免地会引出另一个改进。一旦 PLC 可存储工作数据，例如计数值，用户就要求永久性存储，因此数据在 PLC 程序停止后不会失去。备用电池存储器和电可擦除的只读存储器（EEPROM）可在断电后用来保留所选的存储器的指定区域的内容（包括用户程序）。

定时器和计数器指令实际上是函数程序的调用，它包含在 PLC 的操作系统里。生产商发明了功能块图形元素以表示这些在梯形图中预编程的子程序的调用。功能块图形元素使函数程序需要什么输入数据和它控制什么输出元素这些概念更清晰。在利用编程器监控 PLC 的运作时，功能块有时显示工作数据值（例如计数累加值）。第 4 章后会涉及到在 PLC 编程语言中功能块元素应用的范例。

PLC 现在提供定时器和计数器指令，它们执行数据字的算术操作和比较操作，下一逻辑步骤是提供允许程序员在存储器内储存字的指令，检索那些数据字，以及执行对那些数据字的其他的算术、比较和逻辑操作运算。模拟 I/O 模块是 PLC 用来读写模拟 I/O 值，像读写数字数据一样，还利用串行通信去和其他计算机交换数据字。第 6、7 章会介绍处理个别数据要素和数据集的指令。在第 5 章我们会描述 Allen-Bradley、Siemens、OMRON 的 PLC 的数据储存性能，以及描述用户编写的程序如何在那些存储器寻址数据。

大部分功能块指令和数据处理指令通过调用作为一部分包含在 PLC 操作系统的预编程的函数或子程序工作。很快，PLC 开始允许程序员写他们自己的函数和子程序，并且可以通过编写布尔逻辑语句跳转到这些函数和子程序。使用有条件的包含或排除部分用户程序的指令意味着确定性的 PLC 控制变得更加多变。如果调用函数或子程序，扫描循环需要花费更长的时间。提供其他 PLC 指令进一步减少了 PLC 控制的确定性。主控继电器和转移指令允许程序不考虑（或完全忽略）部分用户程序。通过提供立即 I/O 指令，允许程序员在执行用户程序时，从输入模块读数据和写数据至输出模块进一步摆脱扫描循环的限制。大部分现代的 PLC 甚至提供中断性能，因此扫描循环可以在任何时候被中断，去执行重要的控制程序，从而响应 I/O 模块的信号或定时器的信号，或处理检测到的 PLC 出错情况。中断结束后，PLC 继续扫描循环，就像没有中断过一样。第 8 章将介绍结构化编程技术，第 11 章介绍中断。

当 PLC 和 PLC 程序设计语言变得更强大时，即使有功能块元素，梯形图的局限性也变得更明显。PLC 的功能块和存储器寻址方式变得不那么标准化，因为生产商不愿意使用最好的方法以实现更先进的功能。一些 PLC 甚至开始提供其他程序设计语言。为了尝试找到 PLC 的程序设计标准和存储器访问标准，国际电工委员会（IEC）确定了一个非强制性的标准，称作 IEC 1131-3，它为某些程序设计语言及它们的元素提供指导，并将 PLC 编程分类成如梯形图、功能块、指令表、结构化文本和顺序流程图等。我们在第 9 章介绍这些 IEC 兼容的语言。（IEC 1131 的标准集最近被重新编号为 IEC 6-1131，但大多数供应商仍称它们为 IEC 1131。）

现在一些 PLC 包含不止一个微处理器，因此 PLC 可以同时执行多个操作。附加的微处理器有时在智能 I/O 模块里，完全有自身的操作系统程序和存储器，有时和主微处理器一起在 CPU 模块里。附加的处理器是从处理器，它们在单一的主处理器的控制下运行。主处理器执行扫描循环，因此用户程序仍控制 PLC 的一举一动。至少对于一款最近推出的 PLC 而言，其扫描循环中的 I/O 扫描步骤是由一个微处理器执行的，而其他微处理器运行用户程序。

PLC 的通信能力也有所发展。最早的 PLC 可以读输入触点和写输出触点。所有 PLC 都可与编程器相连，因此用户可以更改用户程序和存储器的工作数据区，以及监控 PLC 运行时的程序和存储器。某些 PLC 可以通过串行通信处理器与 I/O 模块的远程框架相连，那些通信处理器可能在分离的通信模块里，或可能直接嵌入在 CPU 模块里。目前串行接口允许 PLC 与传感器/执行器网络相连，因此 PLC 可读写一些没有与 PLC 的 I/O 模块相连的远程传感器和执行器，PLC 生产商提供可以使 PLC 和编程器互相连接的专有的局域网。网关有时可能连接其他供应商的 PLC 和计算机到一个专有的局域网。某些 PLC 已经使用现场总线标准的局域网，这个标准可用于与任何 PLC、任何兼容的传感器、执行器，和其他控制设备的相互连接。第 10、13 章会讨论标准通信能力，第 16 章会讨论通信的发展。

有人认为，总有一天，个人计算机可以代替 PLC 在工业控制上的应用（我认为仅以 PC 的名字来代替是不够的）。关于这个是否会发生，两方面都有很好的理由。假使你现在想进一步了解本书有什么观点，我想指出的是，个人计算机的拥护者曾经预言 PLC 会在上个 20 年消失，然而在那段时期 PLC 的销售持续快速上升。第 16 章会讲述支持和反对个人计算机的论点，其他影响 PLC 发展的趋势将尽可能客观地描述。

1.5 故障检修

把 PLC 引入车间的困难常常是由于缺乏了解 PLC 可以（和不可以）做什么，或由于对改变的抵制（至少在某种程度上，这种了解的缺乏导致了对 PLC 的抵制）。为了使 PLC 的引入更加容易，为操作和决策的人员提供培训是一个好方法，由此他们会了解通过 PLC 可以（及不可以）做什么。这种培训应该包括实际的操作，受培训的人员可以使用 PLC 来做实验。

克服缺乏经验的最好方法是找一个简单的可被 PLC 控制的过程，然后实现这个过程的 PLC 控制。这样可以迫使大家在工程的所有阶段（从计划到硬件选择和装配、编程和调试，再到车间设备的安装和操作员的培训）去面对他们知识的贫乏和他们对改变的惯性抵制。当然，这个过程应该并不重要，因为它并不是马上就要运行的。它又不能太不重要，否则，如果它比预料更难，就没有任何动力去完成它。一旦完成了一些简单的项目，实施小组将会学到足够的知识，因此他们会奇怪为什么第一个工程看上去很难。还有，他们会渴望实现他们的新技能。

PLC 培训应把重点放在扫描的顺序上。大部分用户从编写包含冲突逻辑语句的程序开始，然后想知道为什么其中一个条件不能工作。如图 1-8a 所示的程序，如果油漆用完，操作警示灯不会开；只有传送带停了，警示灯才会开。一旦用户理解如何在输出动作前对整个用户程序进行评估，从第一梯级上的最先的逻辑要素到最后的梯级上的最后的逻辑要素，用户会明白为什么如果控制一个输出语句的完美的逻辑条件后面跟着另一个控制相同输出端的逻辑条件时，控制的输出语句会没有效果。图 1-8a 的程序不能工作，因为两个梯级尝试控制同一输出，第一个条件打开操作员警示灯，而第二个条件令警示灯保持灭的状态，除非传送带停止。当警示灯由单独的条件控制（如图 1-8b）时，警示灯会在任一报警条件发生时报警。

理解 PLC 扫描循环还会帮助程序员理解为什么图 1-9 的程序不能工作。在执行用户程序时，PLC 不会暂停，即使在等待执行器完成其操作时；并且 PLC 不会忽略任何逻辑条件，即使它们的功能已实现。在这个程序中，当检测零件时，PLC 就启动钻孔进给。一会儿，重复的程序扫描会证实零件在适当的地方（因此进料将会保持），且钻头没有钻到足够远而达到极限。第二梯级无效。最终，钻头会达到其极限，此时第二梯级会开始收回钻头。收回输出是在双进

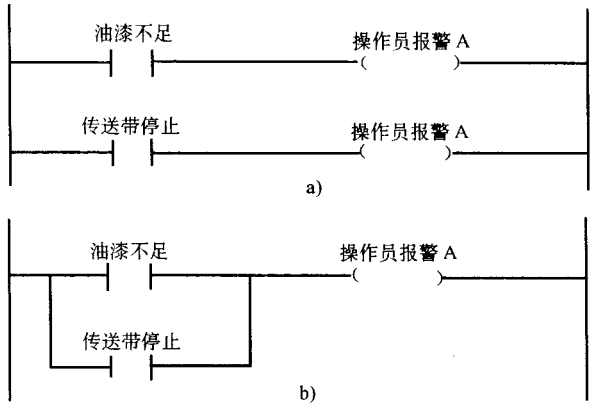


图 1-8 冲突逻辑条件：a）错误版本；b）正确版本

给控制的第二级，因此它会覆盖第一级的扩展输出，所以钻头开始收回，但只要钻头稍微收回，第二个条件停止为真，收回输出立即停止。零件仍在那个地方，所以第一级扩展逻辑条件继续为真，因此又伸长钻头。钻孔进给装置会在前进和后退的快速转换中失效。

在第 3 章，我们会知道如何创建克服图 1-9 指出的问题的顺序程序，第 8 章我们会学到一些可使 PLC 跳过指定梯级的结构化编程指令。

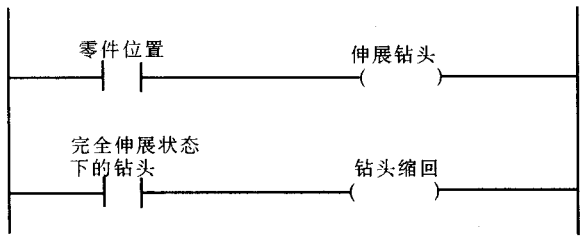


图 1-9 另一个错误程序

习题

1. 集成 PLC 同模块化 PLC 有什么不同？
2. 模块化 PLC 需要什么组件（或组件的类型）？
3. 电源供应模块提供什么类型的电源？
4. （1）列出并解释 PLC 的 3 步扫描循环的步骤。
   （2）一个典型的扫描循环耗时多少？
   （3）扫描循环结束后会发生什么？
5. 如果 PLC 必须放置在远离传感器和执行器的地方，你会用什么方法选择远程 I/O 设备？
6. 什么是输入映像？
7. 在 PLC 和其他基于计算机的控制器出现之前，是用什么机械设备控制自动化过程？
8. 基于事件的顺序器和基于时间的顺序器有什么不同？
9. 为什么使用梯形图？为什么不是其他更传统的计算机语言，如 Basic、Pascal、C？
10. 列出 IEC 1131-3 标准许可的 5 种编程语言。
11. 定义实时控制。PLC 扫描循环如何使 PLC 继承了实时控制应用的优点？

第 2 章　PLC 组件

2.1　学习目标

本章你将了解到：

- CPU 模块的操作和它如何使用内存；
- 电源模块和总线（或框架）用来做什么；
- 数字、模拟和智能 I/O 模块：
  - 如何连接传感器和执行器；
  - 它们做些什么；
  - 如何在用户程序中寻址传感器和执行器数据；
- 编程器：它们做什么以及有什么可选项。

正如我们在第 1 章学到的，一些 PLC 被集成到一个独立的单元，而另一些则是模块化的。模块化的 PLC 由很多的可以插入背通总线或框架的组件构成。每个 PLC 需要：

- CPU 模块
- 电源模块
- 最少一个 I/O 模块

集成式 PLC 将所有组件包含在一个单独的盒子里，所以它的 I/O 性能由制造商而不是用户决定。一些集成式 PLC 可以通过扩展插槽增加附加 I/O 模块，使它们在某种程度上是模块化的。

单独购买时，模块化 PLC 必须包含 CPU 模块、电源支持模块和 I/O 模块。它们可以全部插在一起或插在同一框架上。一些制造商提供带有内置高速 I/O 性能的 CPU 模块，使模块化 PLC 某种程度上是集成的。典型的模块化 PLC 的配件如图 2-1 所示。

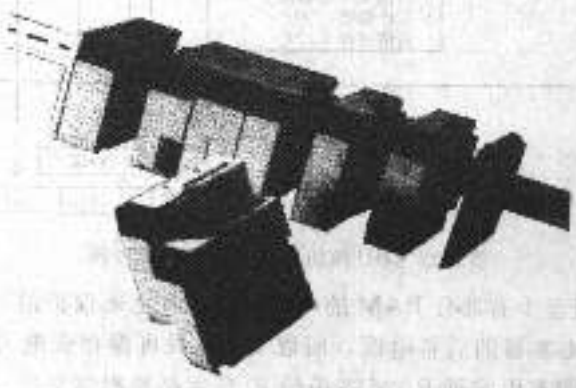


图 2-1　模块化 PLC 的配件 OMRON CQM1

模块化 PLC 有各种各样的大小，习惯上分为小型、中型或大型。这种分类部分基于 CPU 模块里的电源，部分基于 CPU 模块提供给输入输出映像数据的存储器的大小。I/O 映

像表的大小限制了连接在 PLC 上的传感器和执行器的数量。随着越来越多的更强大的信息处理器、更便宜的存储器以及 PLC 之间的更高速的数据通信接口的出现，这种分类方法变得没什么意义。某些现在的小型 PLC 使得几年前的大型的 PLC 看上去像个玩具。在本书中，我们仅区分模块化 PLC 和集成式 PLC。

2.2 CPU 模块

如图 2-2 所示，CPU 模块包含中央计算器及其存储器。存储器包括存放操作系统、驱动程序和应用程序的预编程只读存储器 ROM，以及存放用户编写的程序和工作数据的随机访问存储器 RAM。PLC 制造商提供多种型号的保持性存储器，使得断电后仍可保存用户程序和数据，因此只要通电 PLC 就可继续执行用户所写的控制程序。如果 PLC 选择使用了以下的保持性存储器，则每次启动时都不需要再编程，所以 PLC 不需要键盘和监视器。

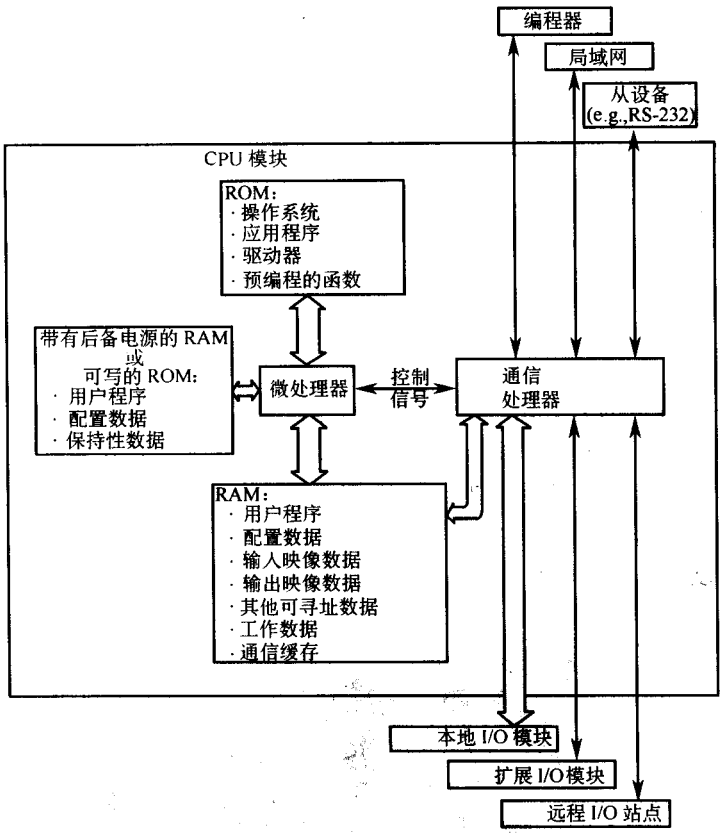


图 2-2 CPU 模块中的存储器和处理器

1）大部分 PLC 里至少有部分 RAM 的内容是被长效电池保护的，可以使用很多年。其他 PLC 只有使用基于电容器的后备电源，所以 RAM 只可保存到电力停止供应为止的短暂时间（以小时计）。只带有电容器 RAM 备份的 PLC 也必须提供至少一个下面提到的选择。

2）很多 PLC 还提供可擦除的存储器模块，它们可插入到 CPU 模块。用户可在可擦除的存储器模块复制用户程序和数据到电可擦除的只读存储器（EEPROM）芯片。用户要购

买 EEPROM 写入器，但一些 PLC 包含了特殊的电路系统，它需要在比 CPU 模块平时正常使用更高的电压下写数据到存储器模块。PLC 可以配置成只要电源开关一打开就可复制存储器模块的内容到读写存储器（RAM）。EEPROM 模块可以插入任何一款设计相同的 PLC，所以在 PLC 间复制程序和数据时它们十分有用。

3）近来，PLC 开始包含闪存。闪存有点像 EEPROM，但它可以更快的写入且不需要特殊的电路系统。闪存用于可擦除的存储器模块，代替原来使用的 EEPROM 存储器芯片。但闪存有时也内置在 CPU 模块，在 PLC 运行时它可自动备份 RAM 存储器的部分内容。如果运行中带闪存的 PLC 突然断电，则电源重启后 PLC 会继续运行而不会失去任何重要的工作数据。

现代的 CPU 模块经常包含不止一个微处理器。主微处理器芯片的工作是执行扫描循环，而从微处理器为了处理通信功能，需要通过局域网与功能日益强大的 I/O 模块、远程传感器和执行器以及其他控制器交换数据。从微处理器的工作是响应主微处理器的命令，或响应通过串联通信连接到该 CPU 模块的微处理器的信息。从微处理器可以直接访问与主微处理器共享的数据存储区（在这种情况下，存储器内容可以被更改，不受主处理器的控制），从微处理器还有它们自己的内存，可作为主微处理器扫描循环的部分而由主微处理器读写，或用来响应从微处理器产生的中断信号。第 13 章和第 11 章的通信中断部分会更详尽的介绍从微处理器用作通信处理器的方法。

2.3 框架或总线

在每个扫描循环期间，CPU 模块读写作为模块化 PLC 一部分的 I/O 模块。CPU 模块通过称作总线的并联导线连接到这些 I/O 模块中的任一个。在一些模块系统，总线在框架底板的电路卡中，所有 PLC 模块都插到框架的插槽里。在其他的模块系统，I/O 模块是插到 CPU 的旁边，或插到已经与 CPU 连接的 I/O 模块的旁边，所以总线是通过 I/O 模块连接的。

总线用于 CPU 与 I/O 模块间发送和接收数据，每次数位。CPU 必须明确指定想读哪些 I/O 模块或写哪些模块。I/O 模块地址是根据在总线上该 I/O 模块离 CPU 模块的距离而自动分配的。某些总线是用于在 CPU 模块和 I/O 模块间传输各种各样的控制信号，以及提供运行在 I/O 模块内的电路系统的电源。总线不提供操作连接到 I/O 模块的传感器和执行器的电源。

2.4 电源

如图 2-3 所示，电源模块将市电转换成为 CPU 和 I/O 模块内部的电路系统所要求的直流电。在北美，市电是 60Hz 120 伏交流电或 220 伏交流电，尽管电源模块也可用于其他的输入电源特性。输出电源必须是 5V 直流电，用来驱动计算器电路系统。电源模块可以连接到总线或连接到模块化 PLC 系统的 CPU 模块。

部分（不是全部）电源包含电源转换电路，通过电源模块的端口输出 24V 直流电。这些 PLC 提供的电源可以驱动数目不多的连到 I/O 模块的传感器和执行器。

如果基于 PLC 的控制系统要求更高功率的电源来驱动传感器和执行器，或需要 5V 或 24V 之外的直流电，或需要其他电信号特性，用户必须提供附加的电源（用于需要高压、应用典型的某些执行器），甚至可能需要提供继电器、光耦或其他电路隔离设备。

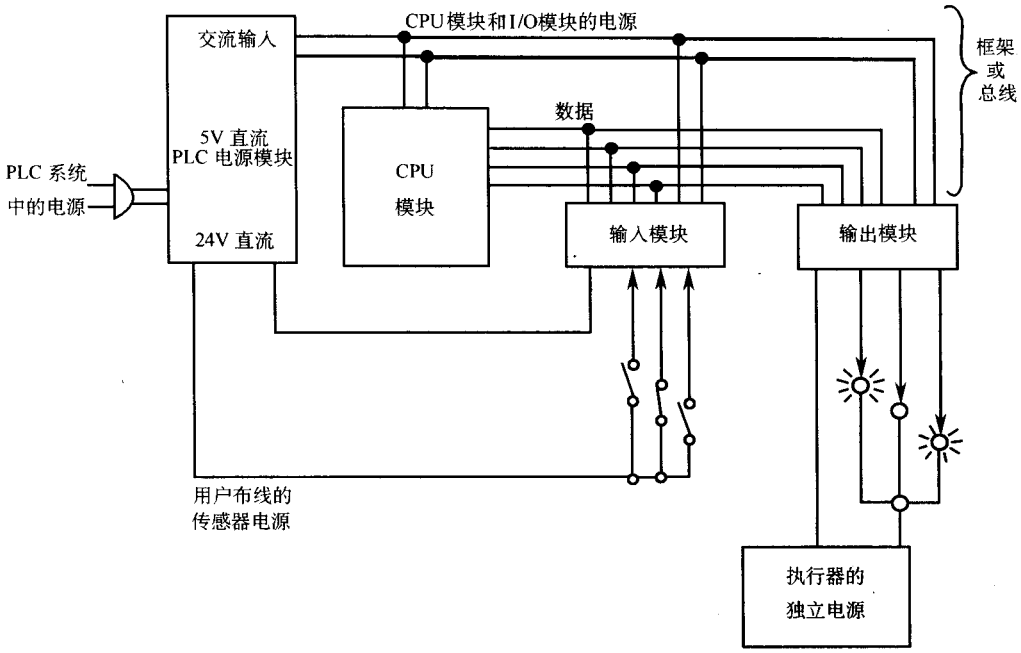


图 2-3 PLC 系统中的电源

2.5 I/O 模块

输入输出模块（I/O 模块）允许 PLC 连接到传感器和执行器。I/O 模块把 PLC 内部使用的低电压、低电流的信号和大多数传感器和执行器要求的高功率电路隔离开来。用户购买他们所要使用的传感器和执行器需要的 I/O 模块，用户可连接很多不同或相同类型的 I/O 模块到 PLC 总线。PLC 制造商提供的 I/O 模块设计成可以与相同制造商提供的 CPU 模块共同工作，所以用户应相信兼容性将不是问题。大部分制造商都有很多 I/O 模块可供用户选择，包括：

1）数字 I/O 模块：用于连接 PLC 和只可以控制开或关的传感器和执行器。可用于多种直流和交流电压或电流。每个模块可连接多个数字传感器和/或多个相同电气特性的数字执行器。

2）模拟 I/O 模块：用于连接 PLC 和可以提供与测量值成比例的电信号的传感器，或连接根据从输出模拟模块接收到的电信号而按比例更改输出的执行器。一个单独的模拟 I/O 模块只可连接不多的传感器或相似电气特性的执行器。

3）各种各样的智能 I/O 模块：每个都有自己的内置微处理器和内存。智能 I/O 模块是设计实现特殊目的的，如计算高频信号或提供电动机的伺服控制。

4）通信接口模块：通过通信链接处理数据交换的智能 I/O 模块。CPU 的用户程序写数据到通信接口模块，模块保证数据放置在通信网络。类似地，通信接口模块可通过通信网络接收来自其他计算机的数据，然后保存它直到 CPU 从模块中读取为止。现代的 CPU 模块可直接连接到通信网络，所以通信接口模块只有在通信请求超出 CPU 的内置性能时才需要。

在本章，我们把通信接口模块作为智能 I/O 模块。通过串行通信的数据交换会在第 13 章讨论。

大多数 PLC 通电后执行自我检测，包括搜索总线以确定有多少模块存在，以便最有效地执行每个扫描循环都会进行的数据交换。PLC 经常与不同型号的 I/O 模块交换不同数量的信息，所以最优化的通信也要求 PLC 知道在框架插槽上的是什么型号的模块。在确实最优化的数据交换中，PLC 不会浪费时间去读输出模块，不会尝试去写数据到输入模块。(CPU 模块每个插槽的存储器仍会包含输入映像和输出映像数据，但数据不会反映传感器和执行器的状态。) 某些 I/O 模块要求交换不同数量的数据，所以 PLC 必须辨认出这些模块。

1) 有些 PLC 制造商要求 PLC 用户根据插入该位置的模块的型号在框架设置开关。当 PLC 通电时，PLC 会读取开关量以配置自身。

2) 有些 PLC 制造商在 I/O 模块设置开关，由此，I/O 模块能以其中一种可选的方式操作。当 PLC 通电并检测到 I/O 模块，它会辨认出不同数据交换请求间的细微差别。

3) 有些 PLC 制造商预先在 I/O 模块上设置确定的特性，所以 PLC 可检测每个在总线上的 I/O 模块，并在通电时配置自身。

4) 有些 PLC 制造商要求程序员在 PLC 进入运行模式前输入配置数据到 PLC 存储器，由此 PLC 会执行适当的数据交换功能。只有在最复杂的 I/O 模块类型才要求程序员的配置。

2.5.1 数字 I/O 模块

数字输入模块允许 CPU 从模块中读取输入映像数据字。数据字的每个单独的位反映了独立开关或开关传感器的开关状态。数字输出模块从 CPU 模块接收输出映像数据字。数据字的每个位会控制一个独立执行器的开或关。

数字 I/O 模块主要提供在低功率的 PLC 内部电路与包含传感器、执行器的高功率电路之间的电气隔离。数字输出模块也提供缓存特性，因此在 CPU 向输出模块写数据字后，模块会保留数据字（保持一些执行器保持开或关）直到下一个扫描循环。此时，新的输出映像数据将被写入输出模块。

数字输出模块 一些数字输出模块以继电器作为隔离设备（见图 2-4）。发送到输出模块字的数据字的每一位可能是 1（5V dc）或 0（0V dc），每一位的保存位置用导线连接到控制继电器线圈。如果该位为 1，线圈被激励，合上开关让电流通过包含执行器的电路。I/O模块每个继电器可能有两个外部的触点，所以独立的执行器和电源可以连接到继电器控制电路。每个输出电路可以在不同极性的不同电平（在继电器的额定范围内）操作，一些甚至可以包含交流电源。更便宜的继电器输出模块中每个继电器可能只有一个外部的触点，一个公共的触点对应一组输出（或对应所有的输出），因此该组的所有电路必须连接到相同的电源。输出模块的公共接线端必须与直流或交流电源的一个端口相连接，其他电源触点通过独立的执行器连到继电器触点。

其他输出模块包含晶体管，CPU 可用它来控制外部电路的开关。(还有附加的组件将内部电路和外部电路电气隔离，但没有在下面的简化图显示出来。)

晶体管可用作电气控制的开关。如果 NPN 型三极管的基极接入正向电，则三极管开关会允许从阳极到阴极的常规电流导入集电极，从发射极流出。如果对应于发射极电势的基极电势为负时，PNP 三极管从发射极向集电极传导常规电流。场效应管（FET）也是用于通

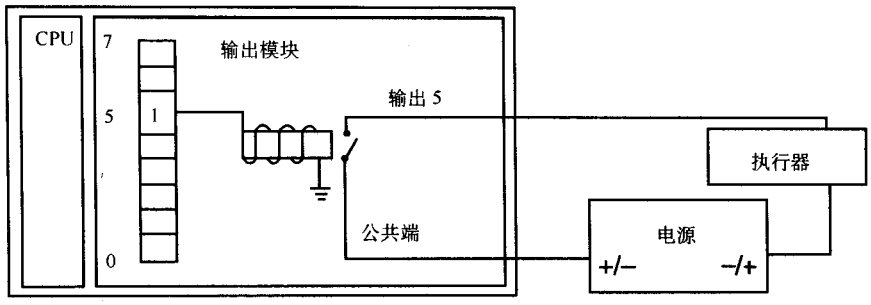


图 2-4 继电器输出模块（只显示了一个输出电路）

过改变对场效应管的栅极触点电势来控制外部电路的开和关。

晶体管输出模块可以是电流源或电流灌入型。电流源输出模块（见图 2-5）通过独立的执行器输出触点提供常规直流电到执行器。电流通过一个标记着“公共端”（Common）的

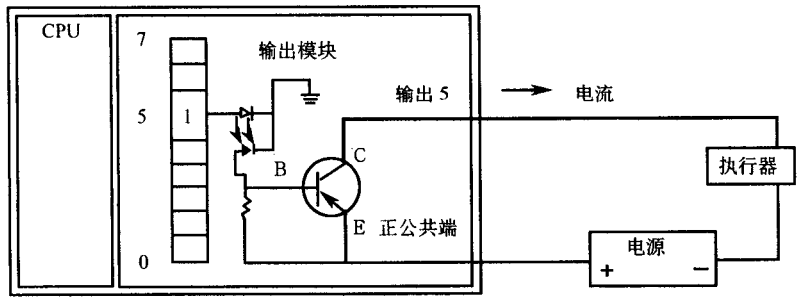


图 2-5 源晶体管输出模块（仅显示了一个输出电路）。有时被称为 PNP 输出或开发射极输出

触点从外部电源流入。在电流源输出模块中使用的 PNP 三极管比 NPN 三极管稍贵且对于滥用的耐受能力更差。NPN 三极管用于更常用的电流灌入晶体管输出模块，如图 2-6 所示。在电流灌入输出模块，输出模块允许（常规）电流通过公共的触点从执行器流到独立的输出触点，最后返回到电源。首次使用 PLC 的用户经常连接直流电源返回到电流灌入型输出模块。

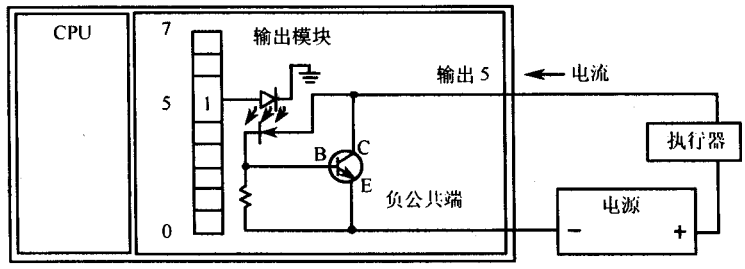


图 2-6 灌入式二极管输出模块（只显示出了一个输出电路）。有时叫做 NPN 输出或开集电极输出

因为三端双向晶闸管开关元件（Triac）价格下降，所以越来越多的被用来代替晶体管。如图 2-7 所示，当输出模块的电路系统对开关元件的触点充电时，Triac 控制的输出电路允许电流从任意方向流过 Triac。虽然用户必须保证在输出模块里，相同的极性在每组执行电路共享一个公共触点，但其实电源极性并不重要（甚至交流电源也可以）。

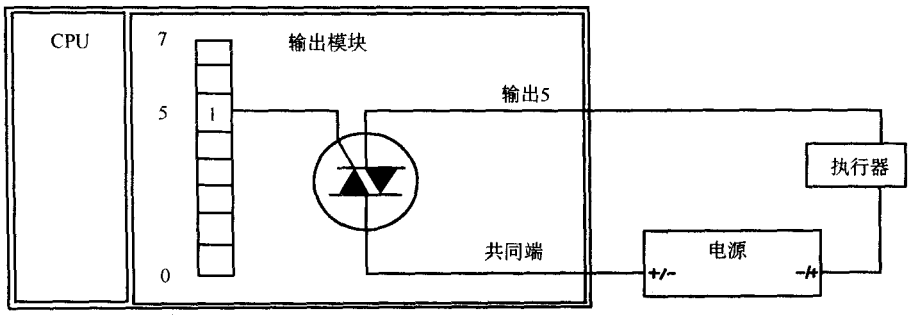


图 2-7 三端双向晶闸管开关元件输出模块（只显示了一个输出电路）。源或灌入输出

数字输入模块 输入模块通常在每个传感器控制的电路中都包含光隔器。当外部传感器开关闭合，电流从外部电源流过光隔器输入触点时，光隔器中的发光二极管就会发光。在光隔器的光敏二极管在它接收到光时允许电流流入低功率 PLC 内部电路。光隔器允许外部电路在两者之间没有任何电连接就可以控制 PLC 内部电路。因为发光二极管是二极管，带传感器的外部电路必须通过正确的极性连接到数字输入模块，所以数字输入模块也可根据它们是否要求常规电流流入或流出独立传感器触点，而分为电流源或电流灌入型，如图2-8和 2-9 所示。虽然可合理地假定输入模块会接收从传感器流过来的电流，但是大部分输入模块是电流源的，且必须连接到传感器，由此合上传感器开关会允许电流从输入模块流出，经过传感器，到电源阴极。电流源输入模块更常见，因为 PLC 使用的输出模块通常是电流灌入型，之前已解释过原因。如果（电流源）输入模块可以从（电流灌入型）输出模块接收信号，那么输入输出模块是相互兼容的。

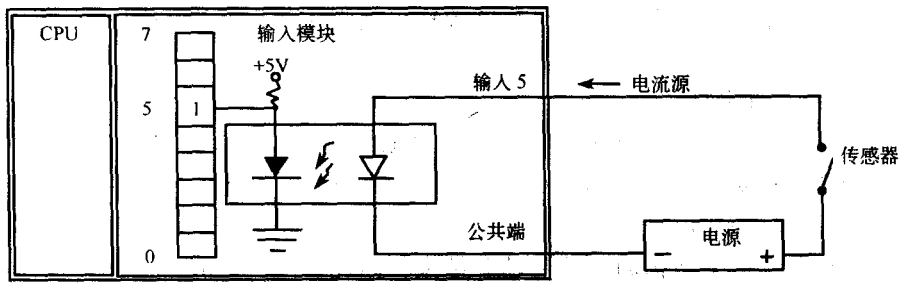


图 2-8 灌入式光隔器输入模块（只显示了一个输入电路）

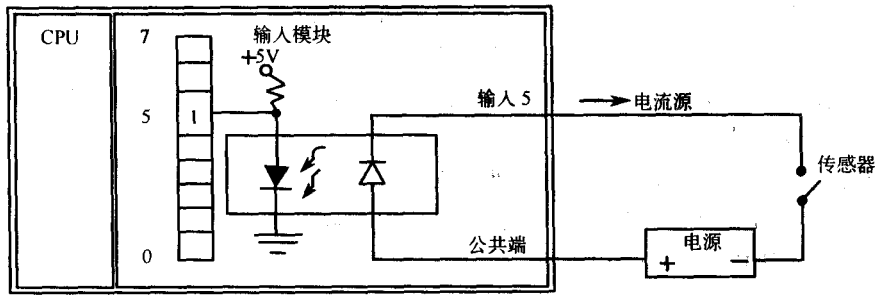


图 2-9 源光隔器输入模块（只显示了一个输入电路）

带两个发光二极管的光隔器，如图 2-10 所示，正在降价，所以更多的输入模块制造成这种形式。无论电流向哪个方向流动，其中一个 LED 都将发光，所以电源极性不再重要（除了在一组触点共享一个公共触点），而交流电源也使用。

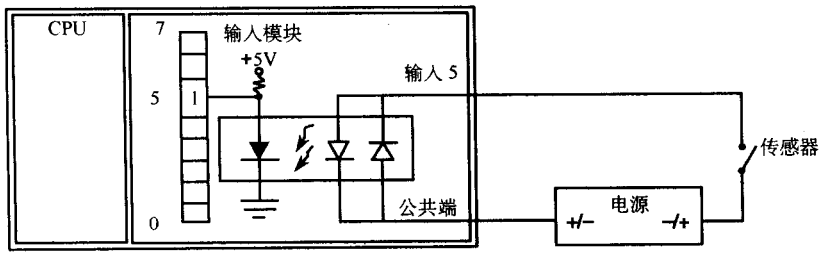


图 2-10 灌入式或源光隔器输入模块（只显示了一个输入电路）

数字 I/O 的寻址：寻址简介

检测数字传感器或控制数字执行器的 PLC 程序指令必须包括传感器的输入映像位或执行器的输出映像位的地址。I/O 映像地址反映传感器和执行器的实际位置。在这一节我们描述一下数字 I/O 寻址和 I/O 模块的位置的关系以及连接传感器或执行器的触点。I/O 寻址会在第五章详细介绍。

PLC-5

1. ALLEN-BRADLEY PLC-5 的数字 I/O 寻址

Allen-Bradley PLC-5 的数字 I/O 寻址包含四种标识符和两种独立特性，如下：

I：123/04

1）首字符可以是 I，表示输入映像位；也可以是 O，表示输出映像位。所有输出映像位保存在输出映像文件，有时称作文件 0。所有输入映像位保存在输入映像文件，有时称作文件 1。在存储器内有独立的区域存放输入输出数据字：I：123/04 与 O：123/04 的地址不同。

2）“:”要跟着首字符。

3）“:”后的头两个数字指出是在哪个 I/O 框架安装 I/O 模块。PLC-5 系统可能包含有 24 个 I/O 模块框架。框架编号为 00～07，10～17 和 20～27。（注意到这些是八进制的数字，不包括数字“8”和“9”。）CPU 模块必须位于 00 或 01。Allen-Bradley PLC-5 可被设置成寻址框架不会在金属框架结构的末端启动和结束。因此，Allen-Bradley 把物理框架称作底盘来区别地址代表的逻辑框架。

4）第 3 个数字指出一组 I/O 模块。框架中最左边的组是第 0 组，而最右边的组是第 7 组。（也没有“8”和“9”）。PLC-5 包含一个存放输入映像的 16 位的数据字和一个存放输出映像的 16 位数据字。一个组因此能表示最多 16 个传感器或 16 个执行器。每个数据字中的某些位不一定要用到。例如，如果只有一个输入模块插入到保留给一个组专用的框架位置，那么没有输出映像位让该组使用。

“组”概念之所以被使用，是因为早期的 Allen-Bradley I/O 模块只有 8 个输入和 8 个输出触点，所以相邻插槽的两个 I/O 模块共享 16 位输入输出映像字。

- 8 位 I/O 模块仍可用，所以 PLC-5 底盘仍可配置成双插槽寻址（见第 10 章的配置）。任意两个数字化 I/O 模块可被插入相邻的插槽，只要它们没有包含超过 16 个输入触

点和 16 个输出触点的总数。

- 现代 I/O 模块通常包含 16 个传感器触点和（或）16 个执行器触点。PLC-5 系统允许底盘配置成单插槽寻址，底盘的每个插槽看作有 16 位输入映像字和 16 位输出映像字的一个组。
- 进一步的微型化使 32 个输入触点和 32 个输出触点组合在一个单独的 I/O 模块上成为可能，因此，Allen-Bradley 也允许底盘被配置成 1/2 插槽寻址。底盘上每个物理插槽可看作 2 个组。在单个 I/O 模块上，有 2 个输入映像字和 2 个输出映像字表示传感器和执行器的状态。

5）“/”用来分隔 3 位数字的映像字地址和 2 位数字的位地址。程序员可用“.”代替“/”（所以，例子也可写成：I：123.04）。当输入梯形图程序时，“/”和“.”将不会显示出来。3 位数字的编号会列在指令上面，而 2 位数字的编号则列在下面，如图 2-12 所示。如果指令使用整个字，省略“/”和后面的位识别号。

6）最后 2 个数字表示指令在使用 I/O 映像字的哪个位。正如图 2-11 显示的那样，16 位被编号为 00～07，10～17（也是八进制的）。

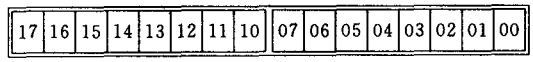


图 2-11 Allen-Bradley I/O 映像数据位寻址

图 2-12 展示了 PLC-5 数字寻址的一级梯形图程序。它检测在输入映像数据字的位 6，该位代表了在框架 04，组 5 的输入模块。如果该位是开的，程序将会打开输出映像数据字的位 16，这一位将代表框架 05，组 3 的输出模块。（没有其他 I/O 映像位受到该程序影响，所以它们的值没有在此显示出来。）

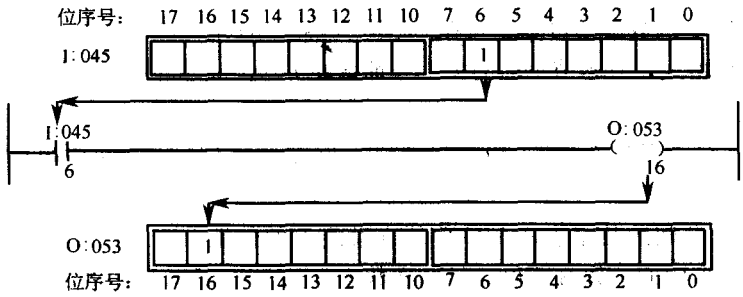


图 2-12 在 PLC-5 中使用的梯形图和 I/O 映像文件

2. ALLEN-BRADLEY SLC-500 数字 I/O 寻址

SLC-500

Allen-Bradley SLC 500 的数字 I/O 寻址与 Allen-Bradley PLC-5 相似，但更简单。字和位都可寻址。I/O 映像位以下面的形式寻址：

I：12/04

1）首字符可以是 I，表示输入映像位；也可以是 O，表示输出映像位。所有输出映像位保存在输出映像文件，有时称作文件 0。所有输入映像位保存在输入映像文件，有时称作文件 1。输入映像文件和输出映像文件是完全隔开的，所以 I：12/04 与 O：12/04 的表示不同地址。

2）“：”要跟着首字符。

3）“:”后的 2 个数字指出 I/O 模块占用总线的位置。I/O 映像数据字 0 是为 CPU 模块带有 I/O 性能的映像保留的。I/O 映像字 1 到 30 为从 CPU 旁边的 I/O 模块（插槽 1）到第 30 个插槽的 I/O 模块保留。

4）“/”通常用来区分 2 位 I/O 映像字地址和 2 位数字的位地址。在一些编程软件中，程序员可用“.”代替“/”（比如这个例子可写成 I：12.04）。如果指令使用整个字，省略“/”和后面的位识别号。

5）最后 2 个数字表示指令在使用 I/O 映像字的哪个位。如图 2-13 所示，16 个位用编号 00 到 15 表示。

图 2-13 展示了带 SLC 500 数字寻址的梯形图程序的一个梯级。它检测了输入映像数据字的位 6，该位代表了插槽 4 中的输入模块。如果该位是开的，程序会打开输出映像数据字的位 14，这一位代表了插槽 5 中的输出模块。（没有其他 I/O 映像位受此程序影响，所以它们的值没有在此显示。）

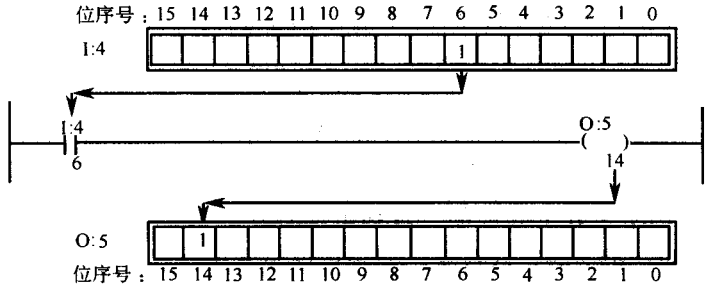


图 2-13 SLC 500 中的梯形图和 I/O 映像文件的使用

S5

3. SIEMENS S5 的数字 I/O 寻址

Siemens S5 PLC 的数字 I/O 数据可在 STEP5 程序上寻址整个字、字节或个别的位。为了了解字或字节寻址格式，看本节下面的关于寻址智能 I/O。位寻址通常利用布尔逻辑，就如：

I4.5

1）第 1 个字符可以是 I，表示输入映像位，也可以是 Q，表示输出映像位。（Q 是多了条尾巴的 O，以区别 0）。

2）字符和“.”间的数字表示 I/O 模块在总线上的位置。

- 在更小的 S5 PLC，如 S5-100U，标准数字 I/O 模块不能有超过 8 个输入或输出触点，总线上的每个插槽由一个过程映像输入（PII）数据字节和一个过程映像输出（PIQ）字节表示。I0 和 Q0 是为插槽上最接近 CPU 模块的 I/O 模块保留的，I1 和 Q1 则为下一个模块保留，如此类推，直到 I31 和 Q31 为第 31 个模块保留。超过 8 个输入或输出触点的新的 I/O 模块，可看作是模拟 I/O 模块。
- 在大一些的 S5 PLC，如 S5-115U，I/O 模块可以有 32 个输入触点或 32 输出触点，要求 4 个 I/O 映像字节来表示在单个 I/O 模块插槽的数字 I/O 值。I0 到 I3 及 Q0 到 Q3 是为最接近 CPU 模块的插槽保留的。I4 到 I7 及 Q4 到 Q7 为下一个插槽保留，如此类推，直到 I63 和 Q63 为第 16 个 I/O 模块保留。

3）“.”是必需的。

4）“.”后，在 0 到 7 的数字，表示 8 位中寻址哪一位。（S5 的存储器的每个字节只有 8 位）

图 2-14 表示带 STEP 5 数字寻址梯形图程序的一级。它检测输入映像数据字节的位 6，该位代表在插槽 4 的输入模块。如果该位是开的，程序会打开输出映像字节的位 4，这一位代表在插槽 8 的输出模块。（没有其他 I/O 映像位受程序影响，所以它们的值没有在此显示。）

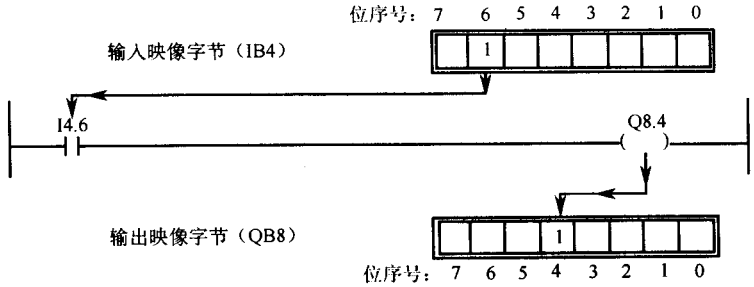


图 2-14 S5 PLC 梯形图和 I/O 映像区的使用

4. SIEMENS S7 的数字 I/O 寻址

Siemens S7 PLC 中 STEP 7 程序使用的数字寻址与 STEP 5 程序的系统十分相似。字和字节都可寻址，如本节下面关于智能 I/O 寻址所描述的，但程序经常会检测或控制一个单独的位。单独位的地址可能如下：

I4.5

1）第 1 个字符可以是 I，表示输入映像位；也可以是 Q，表示输出映像位。（Q 是多了条尾巴的 O，以区别 0）。

2）字符和“.”间的数字表示 I/O 模块在总线上的位置。S7 I/O 模块最多有 32 个输入触点或 32 个输出触点，要求 4 个 I/O 映像字节来表示在单独的 I/O 模块插槽的数字 I/O 值。I0 到 I3 及 Q0 到 Q3 是为最接近 CPU 模块的插槽保留的。I4 到 I7 及 Q4 到 Q7 为下一个插槽保留，如此类推，直到 I127 和 Q127 为第 32 个 I/O 模块保留。

3）“.”是必需的。

4）“.”后，在 0 到 7 的数字，表示 8 位存储器字节中寻址哪一位。

图 2-15 表示带 STEP 7 数字寻址的一级的梯形图程序。检测输入映像数据字节 4 中的位 6，该位代表连接到从 CPU 模块旁边算起的第 2 个 I/O 模块的传感器。如果该位是开的，程序会打开输出映像数据字节 8 的位 4（来控制连在第 3 个 I/O 模块的传感器）。（没有其他 I/O映像位受程序影响，所以它们的值没有在此显示。）

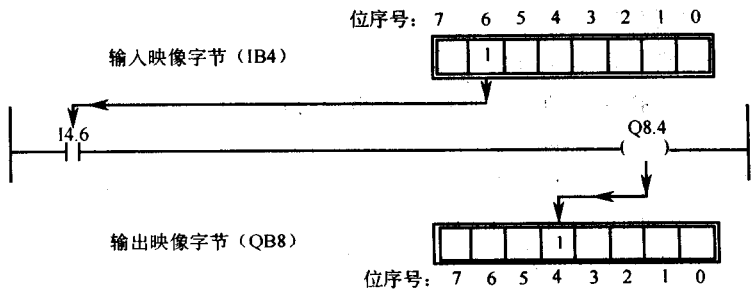


图 2-15 S7 PLC 中的梯形图和 I/O 映像区的使用

CQM1

5. OMRON CQM1 的数字 I/O 寻址

OMRON 的 CQM1 PLC 使用的数字寻址如下：

00405，实际上即 IR 00405

1）IR 前缀表示存储器的 I/O 寄存器空间在被访问存取。程序员不输入 IR 前缀，因为 IR 通常是默认的。

2）第 1 个数字可以是 0，表示输入映像；也可以是 1，表示输出映像。

3）第 2 个数字 0 表示 I/O 模块在总线上的位置。

- x00xx 表示为 CPU 模块带有的 I/O 性能保留的输入映像和输出映像数据字。x01xx 到 x11xx 表示为最多 11 个输入模块和 11 个输出模块的映像保留的数据字。
- 输入映像地址 001xx 是为最接近 CPU 的输入模块保留的，即使还有更接近的输出模块。002xx 是第二接近的输入模块，如此类推，到第 11 个输入模块。
- 输出映像地址同样分配到输出模块。101xx 是为最接近的输出模块保留的，即使还有更接近的输入模块。102xx 是第二接近的输出模块，如此类推，到第 11 个输出模块。

4）最后 2 个数字，编号在 00 到 15 之间，表示（最大）16 位中哪一位被寻址。如果使用的指令可运用整个 16 位字，可以省去位识别符（例如，只输入 004）。

图 2-16 表示带 CQM1 数字寻址梯形图程序的一级。检测输入映像数据字节的位 6，该位代表从 CPU 模块旁边算起的第 4 个输入模块。如果该位是开的，程序会打开输出映像数据字节的位 14，它代表从 CPU 模块旁边算起的第 5 个输出模块。（没有其他 I/O 映像位受程序影响，所以它们的值没有在此显示。）

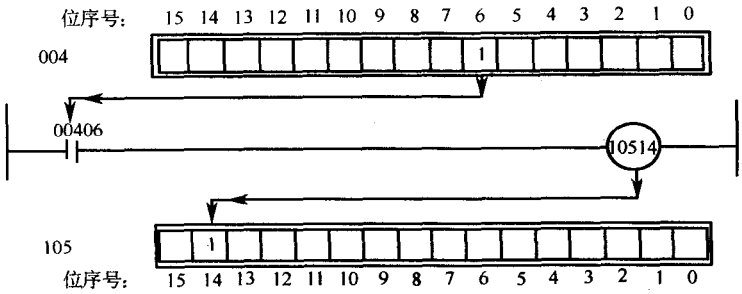


图 2-16 CQM1 的梯形图和 I/O 映像文件的使用

2.5.2 模拟 I/O 模块

模拟输出模块使用数模转换（DAC）芯片把二进制数转换成与数字值大小成比例的直流电压或电流信号。模拟输入模块包含模数转换（ADC）芯片，把模拟直流信号转换成二进制数。到目前为止，我们见到的布尔梯形图指令可用于读写代表模拟值的二进制数，每次一位，在第 6 章会介绍更有效的指令。

模拟 I/O 模块处理在很多标准范围内的模拟信号，包括 4 到 20mA 或 0 到 20mA、0 到 5V 或 0 到 10V 和－5 到＋5 伏或－10 到＋10 伏。一些模拟 I/O 模块可让用户通过在模拟 I/O 模块设置 DIP 开关来选择要用的范围，或有时通过软件配置模拟 I/O 模块。（见第 10 章的 I/O 模块配置。）

由于 DAC 或 ADC 芯片使用二进制位，因此模拟 I/O 模块的分辨率是有限的。每个附

加的位使模块可分辨的模拟值的数字翻倍，因而将可分辨值间的差距减半。在 PLC 系统中的典型 DAC 和 ADC 使用 12 或 13 位二进制值，所以有 4096 或 8192 的分辨率。如果使用 13 位 ADC，在 20V 的变化范围内的输入信号（如－10 到＋10V），理论上可在 8192 个 0.0025V 的子区域之间被识别出来。事实上，大多数模拟 I/O 模块为超出范围的值保留了部分区域，因此－10 到＋10V 范围内的输入模块可能实际上接收和转换在－20 到＋20V 范围内的模拟值，使 13 位的分辨率接近于 0.005V 的大小。由于这分辨率比典型传感器的质量好，12 位 ADC 已经足够应付多种应用。同样地，模拟输出模块很少需要超过 12 或 13 位的分辨率的 DAC。

PLC 只可读写 8、16、32 或 64 位的二进制数，所以代表模拟值的 12 或 13 位二进制数必须作为 16 位数的一部分处理。一些模拟输出模块忽略接收到的 16 位数的低位（如 Siemens PLC），另一些忽略高位（如 Allen-Bradley PLC-5）。模拟输入模块同样将 12 或 13 位数作为 16 位数的高位或低位，但其他位经常用作描述输入模块的状态。图 2-17 显示 Siemens S5 模拟输入模块产生的 16 位二进制数值的结构。代表模拟输入值的 13 位数字是高位（0000 _ 0100 _ 0000 _ 0）。然而，PLC 程序不能将这个 13 位的数作为有效值使用，因为低两位状态位中的一位显示模拟输入模块在它的模拟输入电路检测到一个问题。

程序员应查看所使用的 I/O 模块的说明书，以保证用户程序使用在 I/O 模块处理范围内的二进制数值，还应保证当程序想要中断它们时，模拟 I/O 模块会中断程序：

1）一些模拟 I/O 模块把二进制数看作无符号二进制整数。12 位二进制数的范围从 0000 _ 0000 _ 0000 到 1111 _ 1111 _ 1111，会被翻译为在 0 到 4095 之间的十进制数。

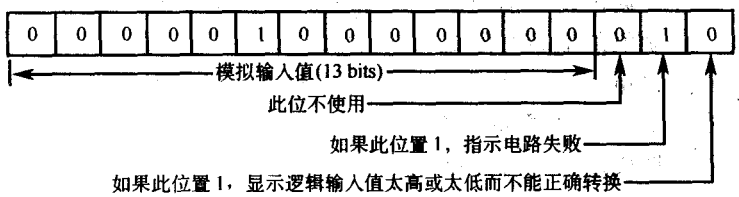


图 2-17 来自 Siemens S5 模拟输入模块代表一个模拟输入值的 16 位二进制数

2）其他模拟 I/O 模块把二进制数看作有符号的二进制整数。12 位二进制数的范围从 1000 _ 0000 _ 0000 到 0111 _ 1111 _ 1111，代表－2048 到＋2047 之间的十进制数。

3）还有模拟 I/O 模块把二进制数译为 BCD 码。16 位二进制 BCD 数 0000 _ 0000 _ 0000 _ 0000 到 0100 _ 0000 _ 1001 _ 0110，必须被写进模块或从模块读取，代表十进制数 0000 到 4096。模拟 I/O 模块通过 ADC 或 DAC 在 BCD 格式和 12 位二进制格式间转换。

一些制造商也提供同时带有输入和输出通道的模拟模块。一些模拟 I/O 模块包含简单的处理器，它能做比在二进制和模拟值之间进行转换更多的事情。这些模拟 I/O 模块会在关于智能 I/O 模块的小节中讨论。

模拟输出模块 模拟输出模块必须保存由 CPU 写入模块的二进制值，模块的数模转换芯片连续产生与二进制数的大小成比例的模拟输出信号，所以只要 CPU 模块写入一个改变了的数到输出模块，模拟输出值就会改变。模拟输出模块经常有 2 至 4 个输出通道，即它们可以同时把很多数字值转换为模拟信号，还可通过单独的一组触点分别为每个通道提供模拟

信号。电源必须与 DAC 相连以提供电源，DAC 允许通过第三个触点输出部分电压或电流，如图 2-18 所示。

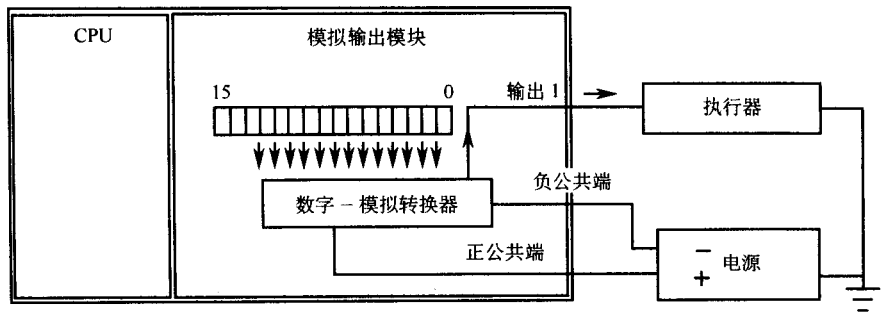


图 2-18 模拟输出模块（只显示了输出通道）

模拟输入模块 模拟输入模块必须将电压或电源信号转换成 PLC 的 CPU 模块可以读的数值。有很多类型的 ADC 可以实现这种转换。有一种是 Flash ADC，可以连续地产生二进制数值，但对于大部分 PLC 的使用而言太贵了。其他类型，最常用的是逐次逼近和双积分式，需要时间来执行模数转换，所以需要对模拟信号重复采样，然后对每个采样数据进行转换。采样数据之间的模拟输入信号值的变化可以忽略。最近的转换过程的结果保存在模拟输入模块的存储器内以便 CPU 模块读取。代表模拟输入信号的二进制值总有轻微的延时，但如果用户小心选择带适合的采样频率的模拟输入模块，那么延时就不成问题。

模拟输入模块经常提供比模拟输出模块更多的通道，因为一个独立的 ADC 常常被多个通道共享（使二进制值的更新时间更长）。图 2-19 显示每个模拟输入通道要求至少三个触点：两个是为了连接电源，一个是为了从传感器来的单端信号，如果传感器使用同一电源的话。有些模拟输入模块测量差分的信号，所以每个通道必须有两个传感器触点，它们之间有电位差或形成连续电流环路。

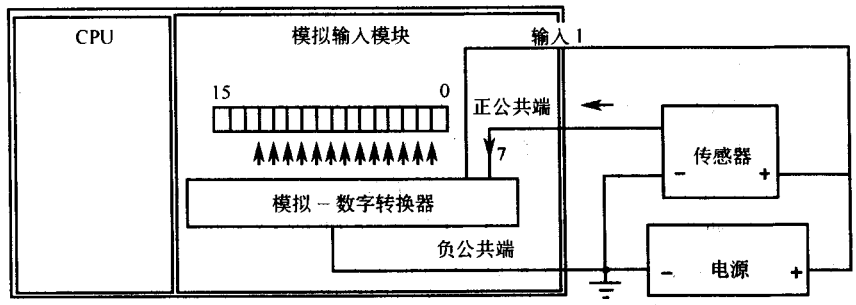


图 2-19 单端模拟输入模块（只显示了输入通道）

模拟值寻址 目前所有 PLC 语言都包括了可以同时操作整个数据字的指令，而不需要像布尔梯形图指令那样每次操纵一个位。我们将在第 6 章讲到进一步的指令，必须使用代表整个数据字的地址，模拟 I/O 模块有它们自带的存储器，还经常带有内置“智能”。模拟 I/O值寻址会在后面的部分同其他智能 I/O 模块一起讨论。

2.5.3 智能 I/O 模块

一些 I/O 模块包含完整的计算机。本书把它们称作智能 I/O 模块，虽然每个制造商都为它们起了特有的名字。现在，甚至一些似乎简单的数字 I/O 模块都带有内置的处理器，使它们成为简单的计算机。不易描述什么是智能 I/O 模块，因为计算机可以完成很多工作，且计算机有很多不同的大小形状。

除读写数据之外，智能 I/O 模块的用户程序还要包括发送命令或配置数据到智能 I/O 模块内的计算机的指令，以及读取模块运行的结果或描述模块状态信息的指令。

模拟 I/O 模块是简单的智能 I/O 模块的例子。一些模拟 I/O 模块可被配置来对模拟量进行标度变换（通过写配置数据字到模块），所以 CPU 内运行的用户程序可被设计为与模块相互转换工程单位数据。例如，用户程序可发送数字 25 到模拟输出模块，使一个直流电动机以 25rpm（转/分）速度旋转。还有其他智能 I/O 模块既可用于实现一些简单功能，例如快速计算从传感器如光学编码器送来的脉冲的信号，又可用于实现复杂的功能，例如使电动机驱动的传送带执行一系列的传动。

智能 I/O 模块的寻址 没有全球都接受的读写智能 I/O 模块数据的程序。一些 PLC 自动分配几个地址给每个智能 I/O 模块，然后，用户程序使用与读写数字 I/O 模块相同的指令读写智能 I/O 模块。其他的 PLC 要求用户程序包含在 I/O 模块和 CPU 数据存储区之间传送含有若干数据字的块的指令。在第二种系统，只有当数据在 CPU 的数据存储区时，用户程序才可处理智能 I/O 模块的数据复制。一些 PLC 含有在 CPU 和智能 I/O 模块间传送数据块并自动交换数据的系统。模拟 I/O 模块经常被看作智能 I/O 模块。一些数字 I/O 模块也可以被看作智能 I/O 模块。

1. ALLEN-BRADLEY PLC-5 的模拟 I/O 寻址

PLC-5

Allen-Bradley PLC-5 提供一种智能 I/O 模块，Allen-Bradley 把它叫做块传输（BT）I/O 模块。模拟 I/O 模块是 BT 模块。BT 模块没有独立的可寻址存储器。必须利用块传输指令将整个数据块传送到 BT 模块或从 BT 模块读取整个数据块，我们将在第 7 章介绍。数据字和位必须写入 CPU 的工作数据存储区，然后，块传输指令可用于复制该数据块到 BT 模块。块传输指令还可用于从 BT 模块复制数据到 CPU 工作数据存储区的任何区域，然后用户程序可从数据存储区读取单独的字或位。在第 7 章，我们描述任何寻址 PLC-5 的数据存储区。数据块的结构很重要。每个 BT 模块都有描述从 CPU 接收或写入 CPU 的数据块的使用手册，程序员必须了解数据块的结构以便知道如何对数据块的字进行操作。

2. ALLEN-BRADLEY SLC 500 的模拟 I/O 寻址

SLC 500

Allen-Bradley 把 SLC 500 智能 I/O 模块称作专用 I/O 模块。模拟 I/O 模块也是专用I/O 模块的一种。数据必须从专用 I/O 模块直接读取，或直接写入专用 I/O 模块。因此，模块的存储器必须以这种形式寻址：

M0：12.3/45

1）M 表示数据存放在专用 I/O 模块的存储器内。

2）M 后的数字可以是 0 或 1。0 有时表示输出值，而 1 有时表示输入值，但 0 和 1 只可表示模块的存储器里的独立区域。

3）“:”是必需的。

4）“:”和“.”间的数字表示总线上专用 I/O 模块的位置。离 CPU 最近的是插槽 1，继续计数直到 30 表示之后的 29 个插槽位置。

5）“.”是必需的。

6）“.”后的数字可能是从 0 开始的数字，代表数据字在专用 I/O 模块存储器的地址。专业 I/O 模块有不同大小的存储器。

7）“/”是可选的。只有当程序指令对单独的位操作时才需要“/”和它后面的数字。如果指令是对整个 16 位的数据字操作的，不用包含“/”。

8）“/”后的数字从 0 到 15。表示指令是对 16 位字的哪一位操作的。

S5
S7

3. SIEMENS S5 和 S7 的模拟 I/O 寻址

S5 系列 PLC 有两种不同类型的智能 I/O 模块。一些可看作模拟模块，且寻址形式与寻址数字化 I/O 数据相似，除非数据通常是以下面的形式整数据字来读写的：

PW72（或 IW72、QW72）

1）如果第一个字符代表 I/O 模块的存储器位置，则它必须是 P。在中型 S5 PLC 如 S5-100U 中，对模拟 I/O 模块的扫描包含在扫描循环内的读写 I/O 模块过程中，因此程序员可用字符 I 或 Q 去读写在 CPU 的输入或输出映像表中的数据值。在这些 PLC，用 P 作前缀是不允许的，除非在中断服务程序，将在第 11 章讨论。

2）下一个字符是 W 表示一个完整的 16 位数据字，或者是 Y 表示一个 8 位字节。对位寻址，可以省略这个字符并且使用前面在介绍数字 I/O 模块寻址时描述的格式。如果 P 做前缀，不能在模拟 I/O 模块里对单独的位寻址。

3）接下来的数字表示模拟 I/O 模块的位置和该 I/O 模块的通道。每个通道代表一个字的存储，由两个字节的数据组成。当一个指令占用了一个字，它对两个字节操作，如图 2-20 所示：

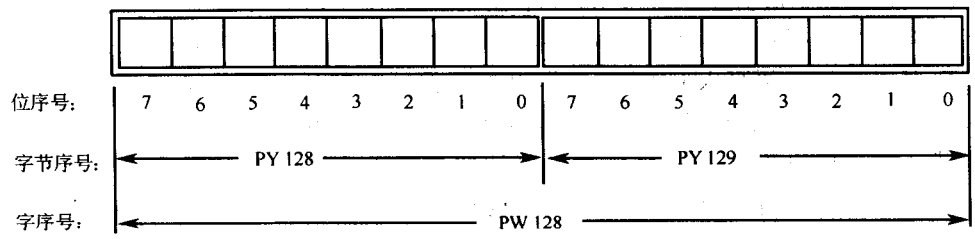


图 2-20 S5 模拟通道使用一个字，它包含两个字节

- 中型 S5 PLC 如 S5-100U 分配 4 个输入映像字和 4 个输出映像字给每个模拟 I/O 模块都可插入的插槽。只有 8 个最接近 CPU 模块的插槽可插入模拟 I/O 模块。如图 2-21 所示，最近的插槽包括模拟输入和输出寻址，从 64 字节开始到 71 字节，最后的模块插槽的最后的字节是 126 字节。
- 大型 S5PLC 如 S5-115U 上包含 PLC 程序能读写的 32 字节的存储器的模拟 I/O 模块。只有 4 个最接近 CPU 模块的插槽可插入模拟 I/O 模块。如图 2-21 所示，最近的插槽包括模拟输入和输出寻址，从 128 字节开始到 159 字节，最后一个模块的插槽的最后一个字节是 255 字节。

插槽号	0	1	2	3	4	5	6	7
CPU		I/O 模块		I/O 模块			I/O 模块	
中等尺寸的 S5	IB 64-71	IB 72-79	IB 80-87	IB 88-95	IB 96-103	IB 104-111	IB 112-119	IB 120-127
	QB 64-71	QB 72-79	QB 80-87	QB 88-95	QB 96-103	QB 104-111	QB 112-119	QB 120-127
大尺寸的 S5	PY 128-159	PY 160-191	PY 192-223	PY 224-255				

图 2-21 S5 PLC 中的逻辑地址分配

STEP 5 编程语言不允许在模拟 I/O 模块的存储器内读写单独的位。有些 S5 系统的复杂智能 I/O 模块，尤其是通信处理器（CP）和接口模块（IM）类型，它们包含了在 CPU 模块和智能 I/O 模块的存储器间自动复制的数据的存储器。两个区域的 CPU 存储器可用来实现此操作：

1）一个区域叫做接口标志区域。当配置 S5 PLC 时（见第 10 章），需要指定哪一部分的 PLC 数据存储区来作此种用途。用户程序应该包括布尔指令以监测包含 I/O 模块状态信息的接口标志，且通过控制其他接口标志来控制 I/O 模块。见第 5 章关于对 S5 的工作存储区寻址的内容。

2）另一个区域是接口区域，是 S5 保存智能 I/O 模块间相互交换的数据存储器区域。这个存储器区域不包含可用传统方式寻址的存储器，但 S5 PLC 包含数据处理功能块，即提前写入的可用来复制数据字到接口存储区或从接口存储区复制数据字的子程序。为了控制智能 I/O 模块，程序必须把单独的字或位写到可寻址的工作存储器，然后使用数据处理功能块从工作存储区复制数据到接口存储区。读取 I/O 模块的状态信息要求使用数据处理功能块从接口存储区复制数据到工作存储区，然后从工作存储区读取单独的字和位。见第 5 章介绍的对 S5 的工作存储区的寻址。在第 10 章我们描述怎样配置 I/O 模块使 CPU 知道哪个部分的接口存储区要用于从模块复制数据或复制数据到模块。在第 7、13 章我们介绍如何使用数据处理功能块。每个智能 I/O 模块都有手册介绍模块和 CPU 相互交换的接口数据块的结构。程序员有责任了解这些信息以便生成数据块，用以控制 I/O 模块和从模块翻译状态信息。

4. OMRON CQM1 模拟 I/O 寻址

OMRON CQM1 模拟 I/O 模块及其他智能 I/O 模块采用与数字化 I/O 模块寻址相同的寻址方式，除非整个数据字通常被作为单独的单位来操作，使用以下寻址方式：

004 事实上即 IR 004

注意到除了不包括区分位的 2 个数字外，字的地址与位地址相同。

每个模拟/智能 I/O 模块可使用不止一个输入或输出映像字。一些 CQM1 CPU 模块有脉冲宽度调制特性，可用来输出模拟信号，但它们要用到一系列特殊的指令，这些会在第 11 章介绍。

2.6 编程器

PLC 需要用户程序，并且在它用于控制工业过程前可能需要在工作存储区对数据初始

化。程序员因而需要一个输入程序和数据到存储器的途径，尽管一般的 PLC 没有键盘或显示器。

PLC 制造商出售各种类型用来编程和输入数据的编程器。编程器还可用来监控程序的执行及监控在 PLC 运行时的工作数据存储区的内容。最简单的编程器是悬挂器。悬挂器有连接导线使它可插入 PLC 的编程端口。悬挂器通常只有很少的有特殊用途的按键和一个很小的显示器件，用于显示几行文本或简单的梯形图。悬挂器容易携带，但难以用于对复杂的 PLC 编程和监控。

曾几何时，大多数 PLC 制造商都提供专用的编程器，就像个人计算机，但是它们只可用来编程和监控 PLC。这种类型的编程器现在已经很少见了。现在，PLC 制造商提供复杂的编程软件，可以安装在个人计算机上，还有接口硬件，因此个人计算机可连接到 PLC 的编程端口。有时，接口硬件由一系列的电缆组成，有时还有 RS-232C 接口到电流回路转换器，因此用户可使用个人计算机的 RS-232C 端口来和 PLC 交换数据。RS-232C 接口通信标准不满足高速交换大量数据的需要，因此有些制造商提供了另一选择：高速通信接口卡，可安装在个人计算机上，通过接口电缆连接到 PLC。为避免携带编程器到各种不同的工作场所，一些编程软件和接口硬件允许作为编程器的个人计算机连接到 PLC 局域网，由此，编程器可用来修改和监控在同一局域网上的 PLC。

2.7 故障检修

选择适合的 PLC 模块并正确连接它们只是创建 PLC 控制系统的一部分。如果控制系统仍不能工作，可就进行以下问题检查：

1）因为 PLC 的 I/O 模块不是通过 PLC 连接到电源的，而是必须在外部连接电源。你有没有把电源同 I/O 模块及传感器或执行器串联在一起？（所选择的电源是否正确?）邻近的开关连接好没有，电源是否可以供电给传感器和 PLC 输入电路？

2）直流电源的极性是否正确？I/O 模块连接时通常有什么极性（如果有）的要求？（作者曾经见过由于使用电流灌入输出模块而造成问题，因为它们不提供电流给执行器。）

3）PLC 程序的寻址是否正确？尤其对于没经验的程序员，很容易把输入地址打成输出地址，且很难找出这类错误。如果程序看上去可以工作（PLC 运行程序时，程序可以被监控），而当程序监测器说 PLC 输出模块上的 LED 开时，它们没有开，则小心留意当前输出的程序地址。

4）PLC 程序是否正确？如果梯形图的一个梯级看上去应该可正常操作，是否有其他梯级与它有冲突？大部分 PLC 编程软件同时提供“前后参照”的特性，可以报告所有在程序中涉及的地址。检测每个梯级，以保证它们都没有引起问题。

习题

1. 模块化 PLC 有哪三种组件？
2. 哪个组件包含了 PLC 的主存储器？断电后，有什么方法可以防止 PLC 用户存储区的内容消失？
3. 扫描程序是在 PLC 的 RAM 还是 ROM 存储器中？用户程序保存在 RAM 还是 ROM 存储器？

4. 什么是通信处理器?
5. 你会用什么类型的输出模块?
   (1) 简单的控制直流电动机的开关?
   (2) 通过主程序控制直流电动机的速度?
   (3) 允许主程序初始化一连串动作而不控制这一连串动作的执行?
6. 简述你如何连接机械开关和电流源型数字直流输出模块。
7. 简述你如何连接感应极性开关和电流源型数字直流输出模块。开关应该是 NPN 型传感器还是 PNP 型传感器?
8. 如果数字传感器连接到第 2 个输入模块的第三个输入触点，远离 CPU 模块（第 4 个远离 CPU 的 I/O 模块)，在用户程序中，你用什么地址去检测传感器的输入映像?
   (1) PLC-5（单插槽寻址）
   (2) SLC 500
   (3) S5
   (4) S7
   (5) CQM1
9. 你用什么地址来对习题 8 所提到的模块上的数字传感器的状态的 16 位输入映像寻址?
10. 如果你的 PLC 使用智能 I/O 模块来处理在 PLC 和其他连在局域网的计算机之间的信息传递，CPU 必须从模块读写什么不同的数据（除接收或发送的信息数据外)?
11. 读取 PLC-5 模拟输入模块，你不只需要专用的地址，还需要专用的指令。你需要什么指令?
12. 你刚写了一个可以检测到在模拟输入的微小变化的程序，它是通过检测输入字的最小位，来查出小的变化。模拟值在改变，但因为某种原因，你的程序找不出变化，低位一直为 0，不改变。为什么这个位从不改变?（假定不是你的程序问题。)
13. 什么存储器区域可用来在西门子 CPU 模块和通信处理器模块间交换数据?
14. 可以使用什么类型的编程器?

第 3 章　二进制逻辑编程（布尔逻辑）

3.1　学习目标

本章您将了解到：

- 布尔逻辑指令元素以及如何将其结合到布尔状态以控制工业过程；
- 梯形图指令和指令表（Siemens 的 STL 语言）等；
- 数据位；
- 编程技巧，如一次翻转法、锁定、顺序器编程。

对于多数控制应用，执行器仅仅需要开、关两种状态，这依赖于传感器的开关状态。例如，当一个盒子被检测到位于传送带上时，驱动马达“开”；当盒子位于传送带末端斜板上待取走时，传动马达“关”。如图 3-1 所示。

布尔逻辑控制程序检测并控制开关状态。每个布尔逻辑程序可包含多个条件语句。以下是一个布尔类型的语句例子：

If（一个盒子在传送带上）AND（斜板里没有盒子）THEN（打开传送带的电机）

也可以用描述传感器和执行器的语句，如下所示：

If（传感器 A 开）AND（传感器 B 没有开）THEN（打开执行器 C）

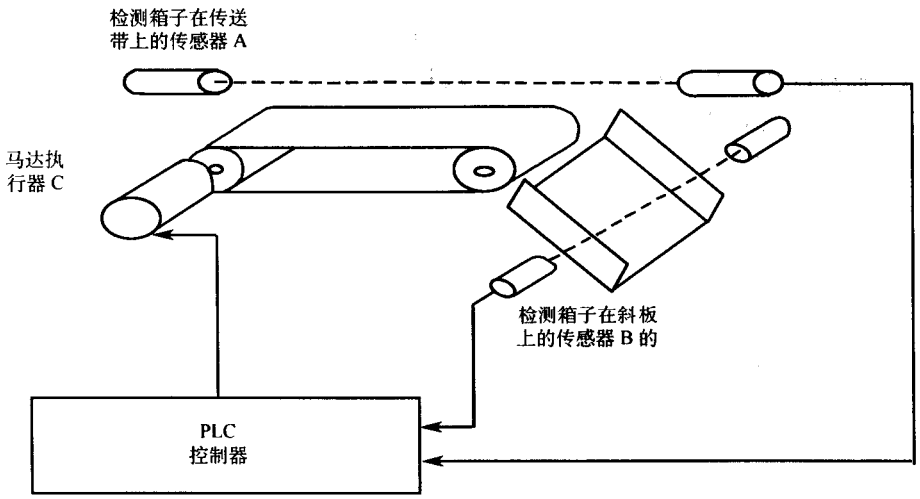


图 3-1　传送带马达控制系统

在布尔逻辑中，只考虑“真”“假”两种状态。上述的布尔逻辑语句中，有两个条件，每个条件可能是“真”或“假”。如果传感器 A 为“on”，则第一个条件为真；否则为假。如果传感器 B 为“off”，则第二个条件为真；否则为假。如果整个布尔逻辑语句取值为真，

则其控制的输出变量为真（即执行它控制的操作；例如，驱动马达 C 启动）。在计算机里，二进制数字“1”表示“开”或“真”，二进制数字“0”表示“关”或“假”。

由布尔逻辑语句可知，有四种可能的传感器开关状态的组合，也就是说四种可能的逻辑操作结果（Siemens 称为 RLO）。仅当两个语句条件都为真时，布尔类型的与操作有且仅有一个结果为真，如表 3-1。

表 3-1

传感器状态		布尔逻辑值状态		
传感器 A	传感器 B	传感器 A 是否工作?	传感器 B 是否没有工作?	执行器 C
关	关	假	真	假（关）
关	开	假	假	假（关）
开	关	真	真	真（开）
开	开	真	假	假（关）

每个传感器和执行器的开关状态用一个二进制数字在存储器的 PLC 输入映像或输出映像存储区表示。在第 1 章中提到，PLC 执行扫描循环期间 PLC 读出连接输入模块的传感器的状态到输入映像存储器，并从输出映像存储器复制数据到连接执行器的输出模块。在第 2 章已讨论过存储器地址和 I/O 模块位置的关系。布尔逻辑程序也可以在 CPU 模块存储器的其他位置读写数据位。

最早的 PLC 只可以用布尔逻辑编程。即使是今天，对于大多数的 PLC，布尔逻辑依然是基本的 PLC 控制程序的控制语言。PLC 程序语言通常使用基于图形的程序设计语言——称为梯形图，有时也使用其他基于布尔逻辑的程序语言指令，包括汇编语言，例如指令表程序语言、更高级的结构文本语言，如 Basic、Pascal、C。在这一章将介绍梯形图和指令表程序语言。

各种 PLC 程序语言间差异很大，即使梯形图也是如此。国际电工委员会（IEC），国际标准组织[⊖]（ISO）的同盟，最近致力于制定 PLC 编程语言标准。事实上，梯形图（LD）、指令表（IL）和结构化文本（ST）是由 IEC 命名并确定的五种标准化语言中的其中三种。IEC1131-3 标准（现改为 IEC 6-1131-3）见第 9 章。

3.2 按位操作的梯形图

梯形图程序和梯形图元素都是基于图形的。梯形图程序和电气工程师所用的继电器逻辑电路图相似。梯形图程序由画在两个垂直线之间的水平梯级构成，因此程序就像一幅阶梯图。和电气图术语一样，左边的竖线可以认为是电源线，右边的竖线为共用连接线。每条梯级包括检查存储器位的指令元素和至少一个控制一位存储位的输出元素。如果在一个简单梯级上的按位检测的元素为真，则整个逻辑操作结果为真，并且将由梯级的输出元素控制的位打开。与电路相似，如果存在一条路径使得电流从电源线通过开关向公共线流动，电流就可以启动在电路中的执行器。然而与电路不同的是，梯形图程序的输出元素通常只能作为每个梯级的最右端元素输入。

如上所述，请记住 PLC 反复执行梯形图用户程序，每次一条梯级，从左上方第一个条

⊖ ISO 不是国际标准化组织（International Standards Organization）的缩写，Iso 是一个意思为“一”或者“一起”的希腊单词。

件元素到最后右下方的最后一个元素。此外，用户程序的执行是三步 PLC 扫描循环的第二步，用户程序只使用输入输出状态的映像。实际输入条件在用户程序开始执行之前由第一步扫描循环读出，实际输出状态在用户程序结束后由第三步扫描循环改变。

3.2.1 梯形图元素

梯形图布尔类型元素（指令）的基本元素包括：

┤ ├ *检查开*，有时也叫*检查输入闭合*，当该位为“1”时表示“真”。（“1”所表示的还可以是“真”、“开”、“置位”）

┤/├ *检查关*，有时也叫*检查输入断开*，当该位为“0”时表示“真”。（“0”所表示的还可以是“假”、“关”、“复位”）

—( )— *激励输出*，有时也叫*输出线圈*，控制一个存储器位。存储器位的值：

- 当前述的逻辑语句中此元素为真时，该位为 1；
- 当前述的逻辑语句中此元素为假时，该位为 0。

*激励输出线圈*的两个常用附加变量：

—(L)— *输出锁定*，控制存储器位，当前述逻辑语句中此元素为真时，该位为 1（开）；但即使该逻辑语句中此条件变为假，该位也不会变为 0。

—(U)— *输出解锁*，控制存储器位，当前述逻辑语句中此元素为真时，该位为 0（闭），但即使该逻辑语句中此元素变为假，该位也不会变为 1。

以下这些元素必须包含被检测或控制的 CPU 存储器地址位：

1. *检查开和检查关元素* 能检查任何数据位，包括输入映像数据表、输出映像数据表中的数据位，以及包含其他工作数据存储器区域中的位。某些 PLC（例如 Siemens S7），如果指令包括一个特殊地址表明指示指令与状态位有关，则检查 CPU 状态位。（例如 BR、OV、OS、>0 等等。详见第五章）

2. 输出元素（*激励输出*、*输出锁定*和*输出解锁*）可以控制在输出映像数据表或在存储器的工作数据区的数据位。大多数 PLC 也提供改变在输入映像数据表中的数据位的输出指令，尽管下次它从输入模块中读取新值时，扫描循环将覆盖输入映像数据位。如果以后的梯级检查到改变的输入映像，那么改变输入映像表数据的程序会出现一些有趣和混乱的结果。

每个品牌的 PLC 都要求程序员用一个特殊的专有寻址公约。在第 2 章已经讨论过输入和输出映像位的寻址，并将在第 4 章重新讲到。要检查专有地址公约，读者可以记住这样一个模式：所有寻址系统包括三个方面：见表 3-2。第一栏规定存储器区域，第二栏确定 I/O 模式，第三栏确定与上述 I/O 模式相关的传感器和执行器。可以用相同的方式确定 PLC 存储器的其他数据位[⊖]。

表 3-2 标准位地址格式

数据类型标识符	数据字地址	数据位序号
• 输入 • 输出 • 数据	• 与 CPU 模式相关的 I/O 模式的位置 • 惟一的存储器字或字节地址	• 传感器/执行器的触点位置 • 存储器位

⊖ 在某些 PLC 中，如果输入的地址没有前缀，则表示被寻址的输入或输出映像。

布尔梯形图适用于所有 PLC，所以可以很容易地熟悉梯形图，无须依赖任何一个专有寻址系统。（在后面的章节里将分别集中讨论每个 PLC 的编程。）除非有必要用专有寻址，我们将用以下简化（假设）寻址系统：

标识符：	IN	用于输入映像数据
	OUT	用于输出映像数据
	D	用于工作数据存储器
数据字：	0到7	
数据位：	0到7	

例如：

IN. 3. 6	输入映像数据位，表示输入模块3的触点6
OUT. 0. 7	输出映像数据位，表示输出模块0的触点7
D. 2. 4	第2个存储器字中的第4个数据位（与任何输入输出数据无关）

3.2.2 创建梯形图程序

每个梯形图的梯级必须有一个输出元素作为最右端元素。“检查”指令通常在输出元素前，见图 3-2 的梯级 1。在这个梯级例子中，要输出为“开”，则三个条件必须同时为真。这就是众所周知的布尔“与”语句（AND），相当于一个串联电路。（值得一提的是，基于 PLC 的布尔编程比电路功能更强大——因为“检查关”指令随时可用，但电路开关断开时不能导电！）

在一些梯形图语言中，即使没有任何检查元素同样可以设计梯级，如图 3-2 所示的梯级 2。这个梯级上的输出元素控制的位恒为 1（真）。在其他梯形图语言，没有条件限制的布尔语句是不允许的，所以设计者常常提供始终为真的数据位，以便使这一位的检测结果恒为真。但其他布尔逻辑语言提供用于设置布尔逻辑操作结果为真的语句。

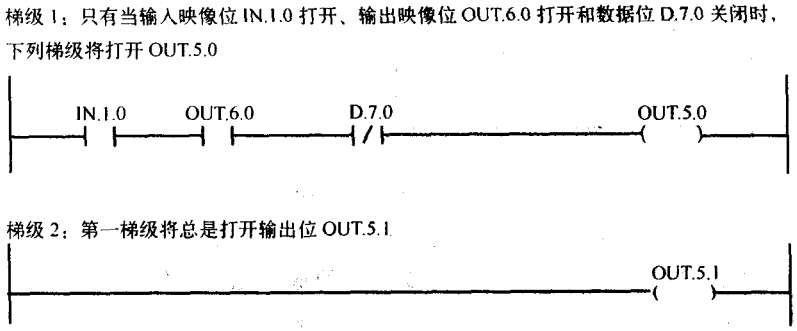


图 3-2 简单的梯形图梯级

一条梯级可以包括从主梯级出发的分支，就像并联电路一样，所以可有多条选择路径来打开输出位，或由同一条梯级控制一个以上的执行器。图 3-3 第一条梯级说明梯形图的布尔语句是如何用多选条件置输出位为开。这就是布尔“或”语句（OR）。

部分梯形图编程语言允许分支上有“输出”指令，但输出元素常常不得不设计为梯级的最右端元素。图 3-3 梯级 2 说明单个逻辑语句是怎样控制两个输出位的。梯级 3 说明怎么添加附加逻辑到输出分支。[⊖]

⊖ Allen-Bradley 和 ORMON PLC 允许分支包括输出，但 Siemens 梯形图编程语言不可以。

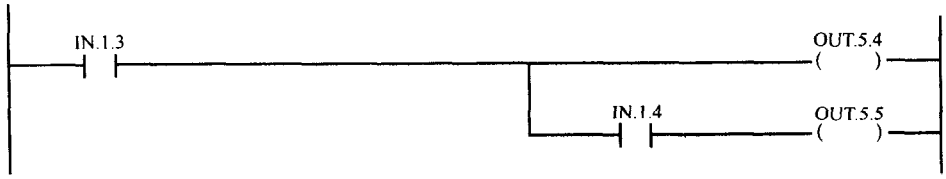
梯级 1：如果 IN.1.1 打开或 D.7 打开，输出位 OUT.5.2 将打开



梯级 2：只要 IN.1.2 打开，OUT.5.3 和 D.7 都会打开



梯级 3：OUT.5.4 和 OUT.5.5 都需要 IN.1.3 打开，但 OUT.5.5 也需要 IN.1.4 打开



梯级 4：如果下列位打开，OUT.4.1 将会打开（(IN.2.1 OR IN.2.2) AND (IN.2.3 OR (IN.2.4 AND IN.2.5))) OR IN.2.6

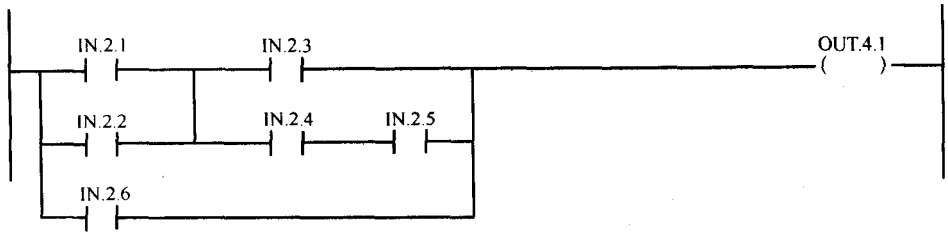


图 3-3 较复杂的梯形图语句

将“与”和“或”逻辑条件结合起来可以构成复杂的网络梯形图语句，但终点不是输出元素的分支必须通过主分支重新连接到主分支左边的输出元素。任何时候从主梯级分出来的分支重新连到主梯级（分支终点的右端）时，PLC 在进行下一步和其他逻辑语句的“与”操作前需要先计算“或”逻辑的结果。图 3-3 梯级 4 说明一个由“与”和“或”语句构成的网络怎样从梯级的左上开始计算。为了简单起见，例子中的逻辑元素都设为常开，输入地址选择为顺序显示逻辑计算结果。

“输出锁定”和“输出解锁”指令允许逻辑语句分别控制一个二进制位的开关状态。图 3-4 说明一对梯级如何用于锁定或解除锁定单个输出映像位。在梯级逻辑程序中，控制锁定或解锁位的两条（或多条）梯级无须连续编程。如果两个逻辑语句在扫描循环中都为假，则输出位从在早先的状态左移一位；如果两个逻辑语句在扫描循环中都为真，则最后一个执行的梯级会覆盖之前的梯级的影响。如图 3-4，如果 IN1.1 和 IN1.2 都为真，则程序结果使 OUT.2.0 为假，因为有解锁指令的梯级最后执行。

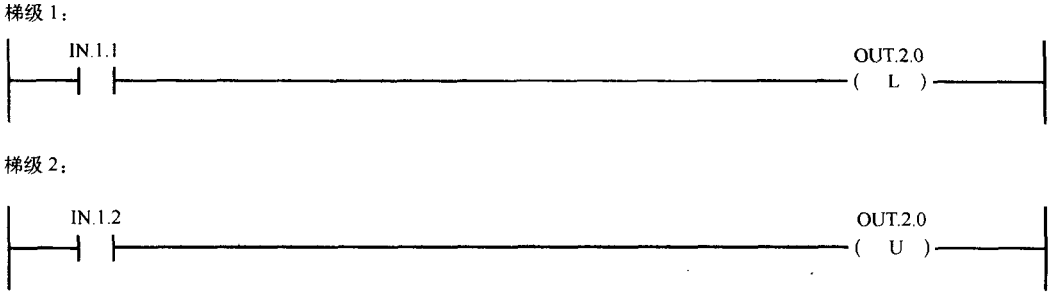


图 3-4 输出锁定和输出解锁例子

OUT. 5. 5 只在 IN. 1. 3 和 IN. 1. 4 都为真时才能打开

OUT. 5. 4 在 IN. 1. 3 为真时打开

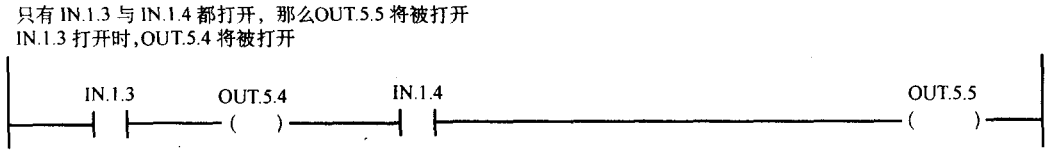


图 3-5 部分梯级逻辑控制的输出元素

一些 PLC 允许输出元素不在梯级的最右端，但梯级仍必须有一个最后输出元素在最右端⊖。图 3-5 显示的梯级以所有梯形图梯级控制 OUT. 5. 5，以部分梯级控制 OUT. 5. 4。此梯形图结果不适合电路模拟——因为执行器（OUT. 5. 4）即使电路不完整（当 IN. 1. 4 断开时）也可以为开。注意，此梯级与图 3-3 的梯级 3 有同样效果。

3.3 按位操作的指令表程序

指令表由 IEC1131-3 命名，是编程语言的一种，与汇编语言十分相似。指令表语言在 IEC1131-3 标准中有详细说明，与 Siemens 语句表（STL）相同。

真正的汇编语言程序对每种可在微处理器芯片执行的机器语言指令都有单独的指令，指令名称简述了微处理器对机器语言指令的响应。例如，在真正的汇编程序中，汇编语言里的 SUB 指令指从一个数里减去另一个数。指令必须包括告诉向微处理器用什么数据参数。

指令表指令不是真正的汇编语言指令，但相比用梯形图编程和其他 PLC 编程语言，指令表指令允许程序员指定 PLC 的微处理器能更精确地做什么。指令表允许程序员编写较短、执行较快的程序，而无须在微处理器执行指令之前进行更多的翻译。Siemens 梯形图程序在进入 Siemens PLC 前会由编程器自动翻译为 STL 语言。因为程序员不能控制翻译的过程，所以翻译后的 STL 程序可能并不是速度最优的形式。

对于所有的布尔程序，从第一个到最后一个逻辑元素对逻辑语句状态的评估过程中，PLC 必须一直跟踪每个逻辑语句的逻辑结果。对于指令表语言（如 Siemens 的 STL 语言），逻辑操作结果（RLO）的当前状态位是已知的。PLC 在开始检测指令表逻辑语句前，即设

⊖ Siemens STEP 7 编程语言对不在最右边的输出元素前加一个“#”。

定 RLO 位为 1；如果 RLO 位在整个布尔语句计算完成后仍然为 1，则执行输出语句。与其他一些指令表数据操作指令不同的是，布尔输出指令依赖于 RLO，这将在以后的章节中讲到。不依赖于 RLO 的输出指令执行时与 RLO 状态无关，且对之前执行的布尔逻辑语句的 RLO 没有影响。

3.3.1 指令表布尔逻辑元素（STL）

Siemens STL 布尔逻辑指令包括：

A "与"指令，后跟正在和 RLO 进行与操作的位的地址。例如：

A IN. 4. 0 若地址为 IN. 4. 0的数据位为1，则该指令结果为真。

O "或"指令，同样需要一个地址位。此指令在 STL 语言里等价于梯形图分支通过主梯级重新连接。

AN "与非"指令。若在特殊地址的数据为0，则指令结果为真。

ON "或非"。

STL 布尔输出语句包括：

= 若 RLO 为真，则使特殊地址位为开；若 RLO 为假，则使特殊地址位为关。例如：

```
A IN. 4. 0
O IN. 4. 1
= OUT. 5. 0
```

当输入映像位 IN. 4. 0 或 IN. 4. 1 其中一个为开时，输出映像位 OUT. 5. 0 为开。

S STL 语言中与梯形图"输出锁定"等价的指令。若 RLO 为真，则特殊地址位为开（置位）；但即使 RLO 再为假，特殊地址位也不会为关。

R 若 RLO 为真，则特殊地址位为关（复位）；但若 RLO 再为假，特殊地址位也不会为开。

复合逻辑语句等价于梯形图里的复合分支梯级，这里括号是必需的，它用来指出 PLC 从语句的一部分取出 RLO 保存，以便把该 RLO 连接到另一部分。以下的括号指令可用，输入时不占地址位：

A (使 PLC 计算从下一程序行开始到结束括号指令逻辑语句的 RLO 结果，然后和该语句之前的 RLO 结果相与。

O (计算紧接的 RLO 结果，然后和之前的 RLO 结果执行"或"操作

) A(或 O(指令的结束标志

STL 程序必须以 BE 结束：

BE 块结束指令（"BE"由一些程序编辑器自动加上）

指令表语言，譬如 STL，通常包含不可用于梯形图的操作指令。例如，STL 提供测试位（TB 和 TBN）指令检查用梯形图或是直接用 STL 的 A 和 O 指令不能检查的存储器位。STL 还包含可以无条件置位和复位的输出指令（SU 和 RU），因此布尔逻辑语句并不是必要的。

3.3.2 创建指令表程序（STL）

在梯形图中，每个逻辑语句必须是独立的梯级，Siemens 称之为"段"。在 STL 可能把布尔逻辑程序分为段，但并非必要。无论何时 PLC 在执行一条将 RLO 输出到一个位的指令（=，S，R 布尔逻辑指令 1 之后，遇到可以影响 RLO 结果的指令（A、O、AN、AC、或者

O（布尔逻辑指令）时，PLC 会自动把 RLO 复位为 1，使后续的逻辑语句取值与前面的语句结果无关。在输出指令后 RLO 不会重新载入，除非遇到一条新的输入指令，因此，可使用一条单独的布尔语句来控制多位输出。注意，你不能编写一个 Siemens 的梯形图语句来控制多个输出，但你能写一条 STL 程序来做同样的事情。

以下 STL 程序例子执行本章前面提到的梯形图例子中的布尔逻辑语句，同样使用二进制寻址系统。加入空格行仅为阅读方便，不是必要的。分号后的注释与 STL 程序 STEP 5 中的作用一样是为了更好地解释程序。

```
A       IN. 1. 0        ;等价于图3-2的梯级1
A       OUT. 6. 0       ;简单的与语句
AN      D. 7. 0
=       OUT. 5. 0

A       D. 7. 0         ;等价于图3-2的梯级2
ON      D. 7. 0         ;逻辑语句,它仅能导致一个需要的真 RLO,跟随任何其他逻辑语句
=       OUT. 5. 1

A       IN. 1. 1        ;等价于图3-3的梯级1
O       D7. 1
=       OUT. 5. 2

A       IN. 1. 2        ;等价于图3-3的梯级2
S       OUT. 5. 3       ;控制两个输出的逻辑语句
=       D. 7. 1

A (                     ;等价于图3-3的梯级3,或等价于图3-5单独的一级
A       IN. 1. 3
=       OUT. 5. 4       ;注意括号能被用来保存一个 RLO,即使 RLO 已经被用来控制和
)                       ;输出
A       IN. 1. 4
=       OUT. 5. 5

A       IN. 2. 1        ;等价于图3-3的梯级4
O       IN. 2. 2        ;注意一个最终 RLO 的建立不必需要和期望一样多的括号
A(
A       IN. 2. 3
O(
A       IN. 2. 4
A       IN. 2. 5
)
)
O       IN. 2. 6
=       OUT. 4. 1

A       IN. 1. 1        ;等价于图3-4的两个梯级
S       OUT. 2. 0
A       IN. 1. 2
R       OUT. 2. 0

BE                      ;STL 程序结尾
```

3.4 一些常见的二进制逻辑编程技巧

布尔逻辑 PLC 程序员利用三步扫描循环的效果设计了一些布尔逻辑的技巧。以下介绍其中三种。

3.4.1 一次翻转法

如图 3-6，两梯级程序在每次输入位（IN. 1. 0）从关到开的改变时，使由梯级 1 控制的位持续一个扫描循环时间。另一个梯级控制一个数据位，且必须在第一条梯级后（不一定紧跟）。要允许一次翻转法执行，必须有第二条梯级。以下为执行过程：

1）假设输入为关（即 IN. 1. 0 为 0），结果使 OUT. 3. 0 和 D. 2. 0 都为关（0）。

2）在 IN. 1. 0 变成 1 后的第一次扫描循环，梯级 1 中的逻辑语句全为真，于是输出映像位（OUT. 3. 0）变成 1。完成梯级 1 后，PLC 开始计算梯级 2 得到数据位（D. 2. 0）为 1。在三步扫描循环的第三步，PLC 根据输出映像位（OUT. 3. 0）打开执行器。

3）IN. 1. 0 变成 1 后的第二个扫描循环，PLC 重新计算梯级 1，因为 D. 2. 0 不再为 0，所以逻辑语句为假，于是输出映像位（OUT. 3. 0）关闭。在扫描循环结束时，与 OUT. 3. 0 相关的执行器在惟一的 PLC 扫描循环中持续为开后也被关闭。

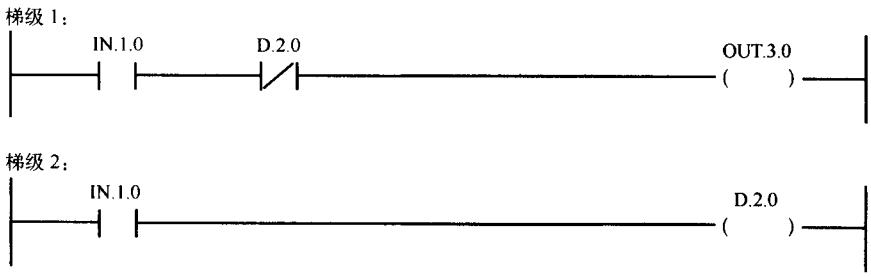


图 3-6 一次翻转法布尔逻辑结构

4）一次翻转法仅在输入位为关（D. 2. 0 为关）、然后再开后才能再起作用。

多数 PLC 制造商承认，一些程序员学习上述的一次翻转法概念有困难，于是在梯形图指令集中加入了布尔一次翻转指令有些 PLC 称微分指令。图 3-7 说明了由本书提到的三个 PLC 制造商提供的一次翻转法的不同图形语句。（本章所有例子使用位地址格式。）所有一次翻转指令控制一个指定用户位（该位不一定为输出映像位），都要求一个数据位从之前的扫描循环中记录控制逻辑语句状态。Allen-Bradley 和 Siemens 的 PLC 要求程序员指定一个数据位，但 OMRON DIFU（13）一次翻转指令利用了内部工作位，使程序员无须指定数据位地址。大多数 PLC（包括本书内提到的）提供附加的一次翻转指令，当布尔逻辑的逻辑语句从真跳变为假时，可以使用一次翻转法。

3.4.2 锁定和封装法

应用布尔逻辑编程使用“输出锁定”和“输出解锁”指令来封装（seal）一个输出。布尔逻辑封装结构如图 3-8 所示。布尔逻辑封装的结构由以下部分组成：

1）由两部分相与的逻辑语句，当语句结果为真时，可以打开输出。

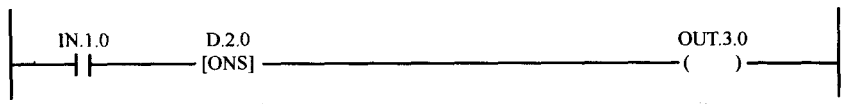
■ 逻辑语句的第一部分通常为假，但如果 IN. 1. 0 为开，则输出打开。

■ 与逻辑语句的第二部分通常为真，但如果 IN. 1. 1 为关，则输出关闭。

2）由或逻辑语句组成的分支。由与语句第一部分的状态（通常为假）与输出位的状态相或。

一旦输出已经打开，或语句的顶部分支可以恢复为“常假”状态，底下分支的输出保持（或封装）为“开”，保持开的时间和主分支中“常真”的与逻辑元素保持为真的时间一样长。封装逻辑和标准化开关电路的布线相似，在开关电路一个常开开关可以关闭，使继电器打开一个执行器，以及继电器一直持续执行，直到一个常闭开关打开，中断该继电器的电源。

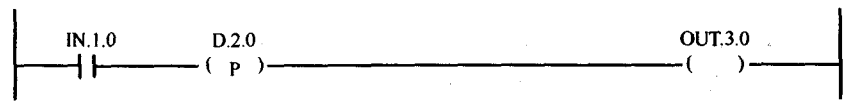
Allen-Bradley:



Siemens S5:



Siemens S7:
在梯形图中



In STL:

```
A     IN.1.0
FP    D.2.0
=     OUT.3.0
```

OMRON:

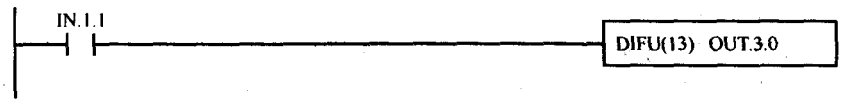


图 3-7 一次翻转指令

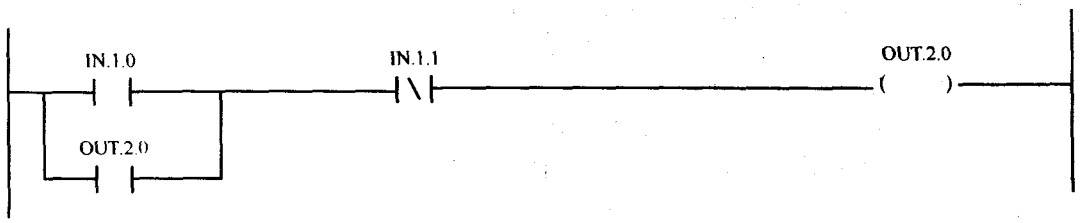


图 3-8 封装梯形图结构

我们已经看到，“输出锁定”和“输出解锁”指令执行和封装结构相同的操作。由于封装结构使得所有控制输出位的逻辑统一到一条单个梯级，这样问题检测会更容易，所以一些程序员愿意用封装结构。而有些程序员则愿意使用“输出锁定”和“输出解锁”指令，因为这些指令允许程序员通过其他开/关执行器的逻辑来控制锁定/解锁输出的梯级。使用“输出锁定”和“输出解锁”，程序员也可以在完全不同的条件下输入几个独立的梯级以开/关执行器。

一些 PLC 提供把“输出锁定”和“输出解锁”指令结合起来的单个梯形图元素，执行和封装结构相同操作。例如 OMRON 提供如图 3-9 所示的 KEEP 指令。当标有 S（Set）的输入梯级上的逻辑为真，在 KEEP 指令中寻址的数据位打开并保持，直到在标有 R（Reset）的输入梯级上的逻辑为真。(OMRON 还提供“置位”和“复位”指令执行“输出锁定”和“输出解锁”操作。)

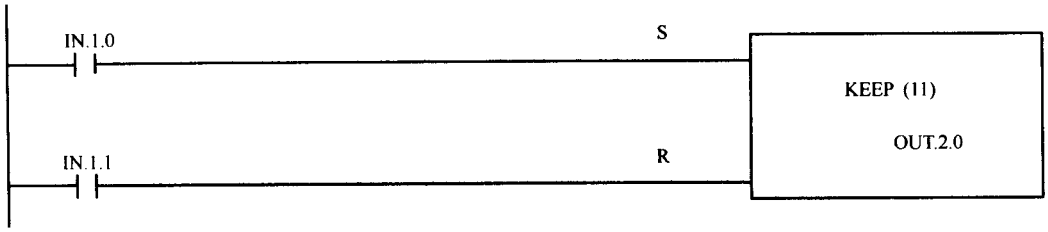


图 3-9 OMRON 封装输出位的 KEEP 指令

3.4.3 顺序器法

锁定（或封装）可以在 PLC 程序里用于控制每次只能按预定顺序执行一步的过程。数据位常常用于跟踪当前哪一个步骤是有效的。图 3-10 所示为一个三步的顺序器法程序。一个 PLC 程序每秒可执行许多次，但由 PLC 控制的产生顺序的每一步，可能需花费几分钟甚至几小时。

第一条梯级只有当三个数据位都不为真时，才执行其输出语句，而这三个数据位只当已排序的进程不是当前进程时才为真。当其他条件也允许已排序过程开始执行时（IN. 1. 1 为真），此梯级开始执行“锁定 OUT. 2. 1”的操作，并锁定一个数据位（D. 3. 1），以指示序列第一步为当前执行。

下一条梯级当且仅当序列的第一步已开始时才执行它的输出语句。当条件允许结束第一步并开始下一步时（IN. 1. 2 变为开），第二条梯级终止由梯级 1 开始的操作且初始化一个新操作（OUT. 2. 1 解除锁定，OUT2. 2 锁定），数据位状态改变，指示当前次序为第二步。

第三条梯级和第二条相同。它控制第二步到第三步的过渡。

最后一条梯级控制第三步终止，并解除最后一个数据位的锁定，于是当在梯级 1 的条件语句变为真时，第一步可再次执行。

当序列顺序必须为可变，或假如开始或结束每一步的条件复杂时，梯形图程序变得相当麻烦。一些 PLC 的高级编程性能可以使编写复杂的顺序器（顺控器）程序变得简单。本书后面将讨论众多方法中的两种。第 7 章讨论特殊顺序器指令、第 9 章讨论流程图语言的编程。

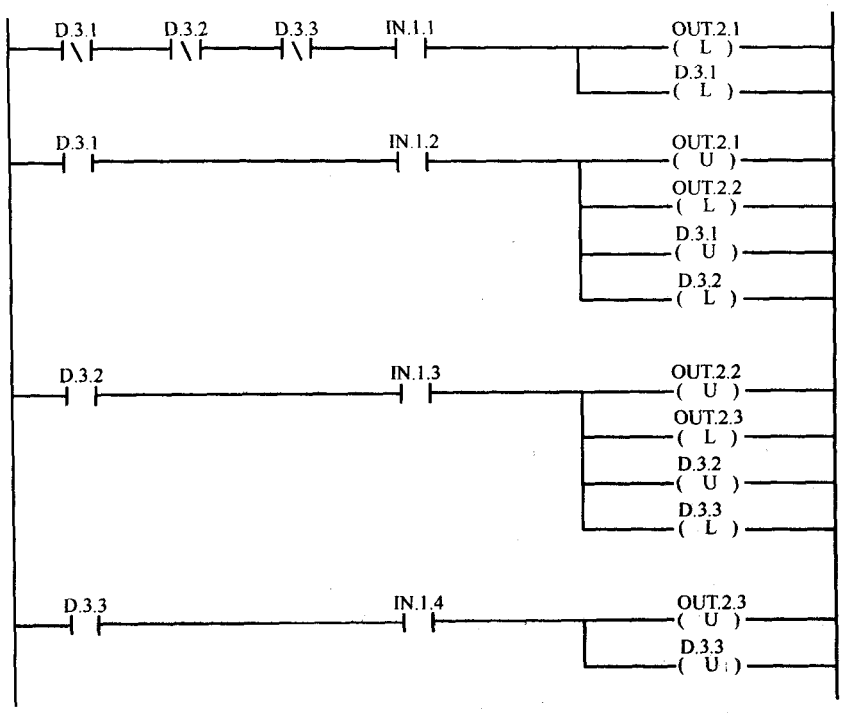


图 3-10 梯形图的三步顺序器程序

3.5 故障检修

PLC 编程新手容易犯的一些二进制编程错误：

1）输入错误的地址前缀。例如本该输入“检查开”指令检查 IN. 2. 1，但程序员输入地址 OUT. 2. 1——这虽然是有效地址但不在程序员预定的存储器位置。类似的，输出指令可能会意外地控制了 IN. 2. 3 而不是 OUT. 2. 3。在第二个例子，不仅与 OUT. 2. 3 相关的执行器不产生响应，而且当程序的下一个梯级尝试检查与 IN. 2. 3 相关的传感器状态时输入映像位的状态会出错——因为它已经被覆盖。

2）两个条件语句控制同一个数据位：

- 如果对同一个输出映像位，一个语句控制“输出锁定”且其他语句控制“输出解锁”，则当两个条件语句者为真时，相关的执行器会一直保持为“关”，这里程序里的语句顺序非常重要，因为最后执行的指令控制着执行动器的状态。按提供所需默认状态顺序编写逻辑条件，或把逻辑包含到条件中使得当其他条件为真时某条件无效。
- 如果两个条件通过“输出激励”指令控制同一数据位，且如果第一条梯级包括真逻辑同时其他梯级包括假逻辑，则用户程序执行过程中存储器位将先开、然后关。如果该位为输出映像位，则 PLC 在 PLC 扫描循环第三步期间关闭相关的执行器，即使逻辑语句指示执行器本应为开。所有控制单个执行器的逻辑都应与一个单独语句相连接。如果有两个使执行器应为开的不同条件，则对同一梯级的条件语句执行 OR 操作。
- 如果一条梯级包含“输出激励”指令且另一条包含“输出锁定”或“输出解锁”，

两者控制同一个数据位，则实际输出状态由 PLC 执行的最后一条梯级控制。如果控制“锁定”或“解锁”指令的条件都为假，则梯级都不处于活动状态，但有“输出激励”指令的梯级在每个程序扫描都处于活动状态。此结构导致控制系统的间歇性失效。

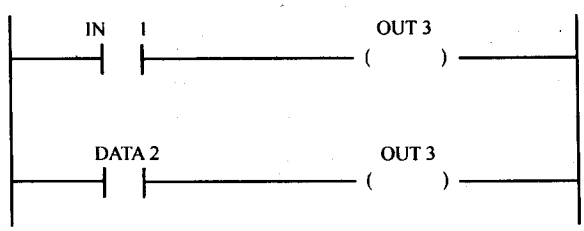
3）把顺序器编程的 PLC 从运行模式切换到修复机械故障，重启 PLC 后，程序重新执行的次序很可能不正确了。PLC 会对一些数据位置位，并且一些输出映像位锁定在 PLC 被切换出运行模式之前的状态，这由正在执行的步骤决定。数据位可能仍为开，甚至对于有些 PLC 输出映像位也为开。PLC 最后运行时会尝试重建最后一次运行时执行的步骤的次序。如果操作员在机械系统清除了器件，则次序的重建可能不成功。操作员和程序员最好确保机械系统和（或）数据存储器内容都处于比转换 PLC 返回运行模式级别更高的适当优先级。欲了解更多，请看第 4 章关于数据位部分。第 10、11 章（尤其是初始化中断和错误中断两节）介绍了如何对 PLC 编程、配置以正确的自动重启。

4）在重启 PLC 的过程中，一次翻转法不会继续执行，即使是使它们执行的条件没有发生改变。多数（不是全部）PLC 在进入运行模式时，自动清除输入映像和输出映像存储器。因此尽管在运行单元中的当前条件不变，在第一或第二次程序扫描时，PLC 可以“看见”条件语句逻辑从假到真的改变。对于第一次也可能是第二次扫描循环，程序可能不得不包含禁止一次翻转法逻辑执行的逻辑。可以参考 11 章初始化中断部分。

少数 PLC 允许输出元素放在固定位置而不是在一条梯级的最右端。

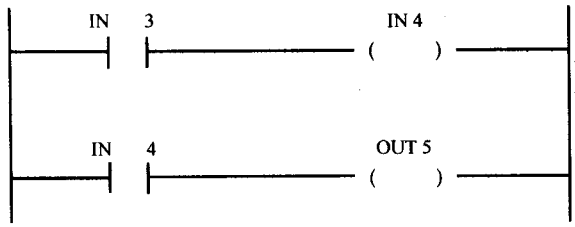
习题

1. PLC 用什么值表示“真”？用什么值表示“假”？
2. 在梯形图中，对位的值进行操作的三个图形元素是什么？
3. 以下哪个 CPU 模块的存储器区域可以由用户程序修改？
   - 操作系统程序区
   - 用户程序区
   - 输入映像区
   - 输出映像区
   - 数据区
4. 假定 PLC 处于运行模式，执行一个扫描循环。按顺序准确地描述 PLC 在每一个扫描循环步骤及每一步骤结束时的工作（如果需要的话，可以使用图表）。
5. 如果以下程序正在运行，在程序执行结束后输出的状态是什么？假设输入和数据位变化如表中所示。



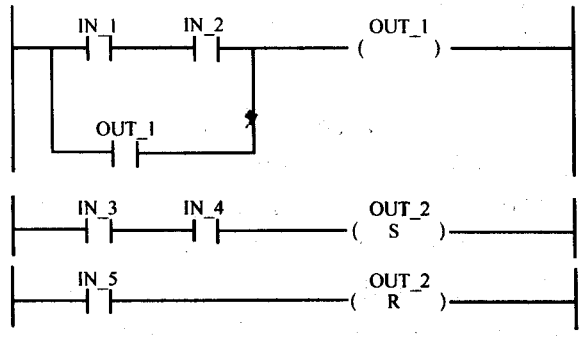
IN 1	DATA 2	OUT 3
关	关	
关	开	
开	关	
开	开	

6. 为什么要避免在多于一条梯级上用“激励输出”指令控制相同的输出映像位？
7. 假设以下程序正在运行，在程序执行完毕后输出状态应该是什么？假定决定输入映像状态的传感器如下表所示变化。



SENSOR FOR:		
IN3	IN4	OUT 5
关	关	
关	开	
开	关	
开	开	

8. 观察以下梯形图梯级，简述它们的工作原理。



编程练习（不需要 PLC）

1. 用一次翻转法写一个两梯级程序，只使用简单的“检查开”，“检查关”，和“输出”指令（若是 Siemens PLC，用梯形图、STL 语言）
2. 写一个梯形图程序，完成以下功能：
   (a) 仅当 IN. 2. 1 为关、IN. 2. 0 为开时打开 OUT. 1. 0。
   (b) 每当打开 OUT. 1. 0 的条件为真时打开 OUT. 1. 1，但 OUT. 1. 1 必须保持为真，即使

OUT.1.0 的条件不再为真时也保持为开。

(c) 每当 IN.2.3 变为开时重新关上 OUT.1.1。若打开 OUT.1.1 的条件和关闭 OUT.1.1 的条件同时为真，前者必须覆盖后者。

3. 重写题 2 的程序，使当 IN.2.2 为开时 OUT.1.0 也为开，但程序其他部分不受影响。
4. 写一个指令表（Siemens STL）程序，完成题 2 的功能。
5. 同题 4 重写指令表程序，使当 IN.2.2 为开时 OUT.1.0 也为开，但程序其他部分不受影响。
6. 写一个梯形图程序，任何时候 IN.4.0 从假变为真（开）时打开 OUT.3.0 一个周期，只使用标准化布尔逻辑语句。
7. 修改题 6 的程序，使得每次 IN.4.0 变为开或关时，OUT.3.0 持续一个周期为开。
8. 写一个梯级逻辑顺序器程序，使用封装结构，于是瞬间触点启动按钮启动传送带，带动车辆往油漆车间，在油漆车间，一旦检测到有车进入则开动油漆喷枪（在喷漆期间车不会停下）。喷枪在车完全经过后关闭，传动带也全部倒卷。（当传动带倒卷时喷漆完的车的位置自动由未喷漆的车取代。这一部分无须编程。）重复喷漆过程。当按下一个瞬间触点停止按钮时，当前循环结束，但不立即开始新循环。

推荐的 PLC 实验室练习

假设一个系统：

- 四个控制面板开关：输入 0 到 3
- 四个指示灯或可视输出模式灯 LED：输出 A 到 D
- 两个弹簧复位阀门控制气缸：输出 E 和 F
- 一个锁销阀门控制气缸：输出 G 和 H
- 三个传感器，检测每个气缸的伸展

1. 写一个梯形图程序完成以下功能：
   (1) 伸展气缸 E：如果开关 0 为开且开关 1 为关，或如果开关 2 为开则不考虑开关 0 和开关 1 状态。
   (2) 当气缸 E 已伸展或伸展中，伸展气缸 F。此梯级应检查与气缸 E 的输出以决定是否伸展，并检查气缸 E 的传感器是否已伸展。(调节气缸气流使得伸展延迟)
   (3) 伸展气缸 G—H：仅在两个气缸 E 和 F 都在伸展期间。检查传感器以决定气缸是否在扩展。(使气缸伸展仅仅是工作的一半。当气缸不需要伸展时，需要用另一条梯级取消)
2. 在一个网络里，写一个 STL 语言程序，完成上述梯级逻辑程序功能。
3. (1) 在写一个程序前，启动开关和传感器同时观察输入模式灯 LED。注意哪一个开关与输入模式中的哪一个触点对应。
   (2) 对 PLC 编程：
      1) 当控制开关 0 为开时，打开显示灯 A。
      2) 当开关 1 为开时，打开显示灯 B，但同一时间开关 2 必须不为开。
      3) 当显示灯 A 或 B 任意一个为开时，关闭显示灯 C。检查输出。
      4) 当开关 1 为开时，打开存储器的第一个可用数据位。
      5) 当数据位为开时，同时打开显示灯 C 和 D，但开关 2 和 3 不能同时为开。

4. 写一个程序，用锁定法（或封装法）执行以下序列：
   - 当开关 0 变为开，伸展气缸 E。
   - 气缸 E 伸展后（用传感器）且开关 1 打开，伸展气缸 F。
   - 气缸 F 伸展后且开关 2 打开，收缩气缸 E 和 F。
   - 当气缸 E 和 F 收缩时，伸展气缸 G－H，但是仅在显示灯 A 和 B 都亮时。
   - 当开关 2 为开时（不管余下的程序执行到哪一步）的任何时候，打开显示灯 A 和 B。
   - 当显示灯 A 和 B 熄灭时，收缩气缸 G－H——但是仅当气缸 G－H 已经在正常顺序下伸展过。
5. 修改题 4 的程序，使得假若气缸 E 在之前的伸展后已经完全收缩，则它可以在整个程序序列结束前再一次伸展，但气缸 F 要直到气缸 G－H 收缩才能再次伸展。

第 4 章　计数器和定时器

4.1　学习目标

本章您将了解到：

- 事件计数和通过计算时间单元计时；
- 计数方向和预置值；
- 累计值、状态位、存储器需求和某些数据的可操作性；
- 向上和向下计数；
- 开延时、关延时，保持计数器和脉冲计数器；
- Allen-Bradley 的详细描述，Siemens、OMRON 的计数器和定时器。

控制过程通常需要可以计算事件发生次数（如放到啤酒箱里的酒瓶的个数）的 PLC，或者是可以对事件计时（如在移走已满箱子代之以空箱子时灌满另一瓶啤酒前该延时多长）的 PLC。图 4-1 所示为一个简化的程序，该程序包含一个计数器和一个定时器以控制啤酒装箱系统。

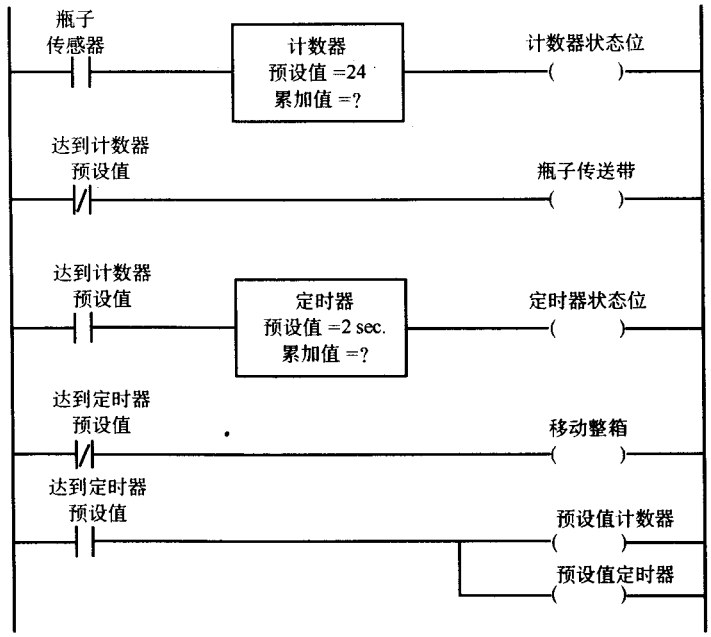


图 4-1　用来显示计数器和定时器的简化控制程序

所有 PLC 编程语言都包括计数器和定时器指令。计数器计算逻辑语句由假到真的次数。例如，一个逻辑语句可用于检查探测传送带上的啤酒瓶的传感器，以便计算啤酒瓶数。定时器负责计算时间单元。例如，在把一个已满的啤酒瓶箱子移离传感器时即启动一个定时器。计数

器和定时器可以用于控制布尔操作，例如启动传送带使一个已满的啤酒箱离开装箱地点或停止啤酒瓶向装箱地点移动。由于计数器和定时器指令影响保存时间值和计数值的存储器单元里的数据字，因此尽管定时器和计数器在布尔程序里使用，但它们都不是简单的布尔编程要素。

计数器和定时器指令作为由布尔逻辑条件控制的输出要素输入。每个计数器或定时器对能被其他布尔逻辑语句检查的状态位进行操作，状态位可显示计数或定时是否结束。PLC必须存储计数器和定时器累计值作为独立的数据字，且在计数或计时指令执行时改变累计值。对于高级的编程运算，布尔逻辑语句可以直接用于控制计数器和定时器状态位，操控数据字的指令可以用于检查或改变累计值。

4.2 计数器指令

有些PLC包含计数到0的计数器。这些计数器包含指示何时累计值等于0的状态位。PLC程序必须包括初始化累计值为预设值的指令，以及从累计值递减到零的指令。指令也可以递增地计数并复位累计值为零（此方式不常用）。

其他PLC计数器使用递增计数至预设值的方式。每次计数值改变，计数器指令就比较累计值和预设值，当累计值大于或等于预设值时改变状态位的值。这些PLC需要保存预设值和累计值。PLC程序需包括复位累计值为零的指令和递增指令，也可以用指令递减计数并设累计值为一个非零值（此方式不常用）。

4.3 定时器指令

实际上，定时器指令是当输入逻辑语句保持为真（或某些情况下当输入逻辑语句保持为假）时，计算时间单元的计数器。就像计数器一样，每个定时器要求存储器（至少）有一个数据字的空间保存定时器的累计值和（至少）一个状态位指示定时器“计时结束”。

和计数器不同的是，多数种类的定时器在控制逻辑禁止时自动重启，然后恢复计时，所以定时器复位指令不是必需的。其他定时器是保持性的，即当被控制逻辑禁用时停止计时、而重新启用时继续计时。保持性的定时器必须有一个定时复位指令。一些PLC提供一次计时定时器，一旦开始计时，即使控制逻辑状态改变也照样运行到计时完毕。

有些PLC的定时器指令和计数器的一样，使用令累计值递减至零的方式。有些PLC编程语言仅提供增量定时器，使累计值递增至预设值。所有PLC定时器当累计值达到零或预设值时改变状态位。有些PLC提供附加状态位和（或）附加的定时器类型，利用它们的状态位指示定时器处于运行状态。

多数PLC定时器指令允许程序员选择时间单位的大小，并指定计算多少个时间单元。定时器仅在一个完整的时间单元结束后才能改变其累计值，所以选择一个过大的时间单位会减少定时器的精确度。定时器指令只有当指令执行时候才能改变定时器状态位，所以长程序也会导致低精确度[⊖]。例如，假如程序需要50ms执行一个扫描循环，则即使定时器的时间单位小于50ms，定时器仍可能有±50ms的误差！

许多PLC提供实时时钟功能，用实时时钟PLC可以跟踪时间变化而不需要用户程序发出指令。用户程序可包含检查甚至改变实时时钟累计值的指令。

⊖ 大多数PLC制造商提供独立于扫描循环时间、计数更快的定时器和计数器。高速计数器和定时器在11章讨论。

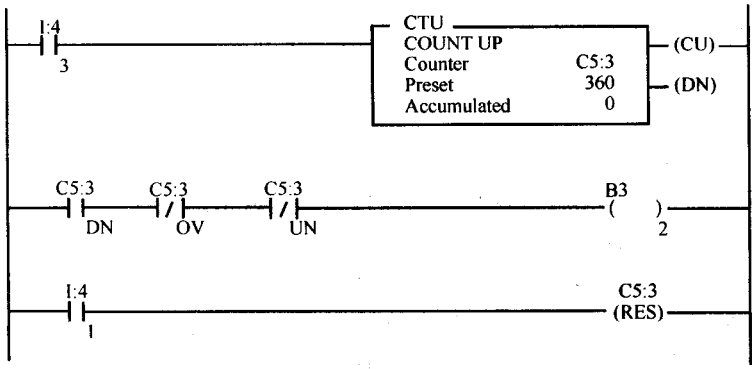
PLC-5
SLC 500

4.3.1 Allen-Bradley 计数器和定时器

在 PLC-5 和 Allen-Bradley PLC 的 SLC 500 系列中预留了一个三字数据结构体给每一个计数器或定时器。如图 4-2 所示，这种三字数据结构体包含两个 16 位的有符号二进制数，代表累计值和预设值，还包含状态位。每个状态位和数据字都可在用户程序里单独寻址。

PLC-5 和 SLC 500[⊖]里有三条可用的计数器指令。如图 4-2 程序所示，每次控制增计数指令的语句从假跳变为真时，增计数指令使得累计值递增。每次控制减计数指令的逻辑从假跳变为真时，累计值递减。复位指令使一个计数器的累计值返回到零，并复位所有状态位。复用增计数和减计数指令可以共享一个计数器数据结构体，以便对同一个累计值和状态位进行操作。

程序员需在输入一个增计数或减计数指令同时输入一个常数作为计数器预置值。预置值可以是－32 768～＋32 767 之间的任何常数。在此范围内累计值可以递增或递减计数。如果计数器计数超过＋32 767，则认为是溢出，数值为－32 768。如果小于－32 768，则认为是下溢，数值为＋32 767。操作同一个计数器数据元素的增计数和（或）减计数指令应给定相同的预置值，因为计数器结构体里的预设值在每次程序员输入一条计数器指令时都会改变。



结构体 C5:3

	CU	CD	DN	OV	UN	*	
C5:3.0(status)	1	0	0	0	0	0	
C5:3.1(.PRE)	+360						
C5:3.2(.ACC)	17						

* 在字 0 的位 10 中，SLC500 高速计数器元素有一个附加状态位（UA）

图 4-2 Allen-Bradley 计数器指令及其控制的计数器要素数据结构体

计数器数据结构体有五个状态位[⊜]，由布尔逻辑程序指令检查，如图 4-2 所示。DN（done）位在累计值大于或等于预设值时为开；CU（count up）和 CD（count down）位反映最后执行的增计数或减计数指令的控制逻辑的状态。OV（overflow）和 UN（underflow）位在累计值溢出或下溢时锁定。当 OV 或 UN 位为开时，表示累计值为非法，则 DN 位的状态可能不正确。布尔逻辑

⊖ 紧凑型的 SLC 500 和 SLC 501 也提供高速计数指令。高速计数器操作在第 11 章的定时中断小节中介绍。

⊜ 也有第 6 个状态位，UA 位，只在高速计时器里有，见第 11 章。

输出指令也可控制上述状态位，但程序员应该避免使用此功能，因为这样可能使计数器无效。

注意，以下两个条件下 Allen-Bradley 计数器指令也使累计值增加（或减小）：1. 控制逻辑为真同时复位计数器后，当 PLC 切换到运行模式同时计数器指令的控制逻辑为真；2. 执行结构化程序，使控制逻辑看起来好像已经从假跳变为真（如第 8 章所述）。

图 4-2 描述了由一个十分简单的逻辑语句控制的增计数指令，指令检查一位单一的输入映像位（在 SLC 500 中的地址是 I：4.3）。仅当从最近一次计数器复位后起累计值没有溢出或下溢，若计数器的 DN 位为开（DN 在累计值大于或等于预设值时为开），则另一条梯级打开一个比特文件位（B3/2）。复位指令将复位累计值为零，且当另一个输入映像位(I：4/1)为开时，复位计数器状态位。其他数据操作指令可以作用于计数器预设值及累计值。移动指令、数学计算以及在后续章节将提到的比较指令，都是重要的例子。如图 4-2 所示，定时器的三字数据结构体里的格式存储了状态位、预设值和累计值。

实际上，Allen-Bradley 定时器是计算时间单元的计数器。和定时器指令一起输入的预设值必须是一个在 0～＋32 767 之间的正数，这个数值代表了时间单元数量。在低级 SLC 500，基本时间单元只能是 0.01s，但在较好的 SLC 500 型号及 PLC-5，程序员可以选择 0.01 或 1.0s 为基本时间单元。定时器只可增加累计值，当累计值达到预设值时停止增值。定时器的数据结构里只有三个状态位：DN（done）位、TT（timer timing）位、EN（enable）位。EN 位反映了定时器的控制逻辑状态，图 4-3 说明了其他状态位是怎样工作的。

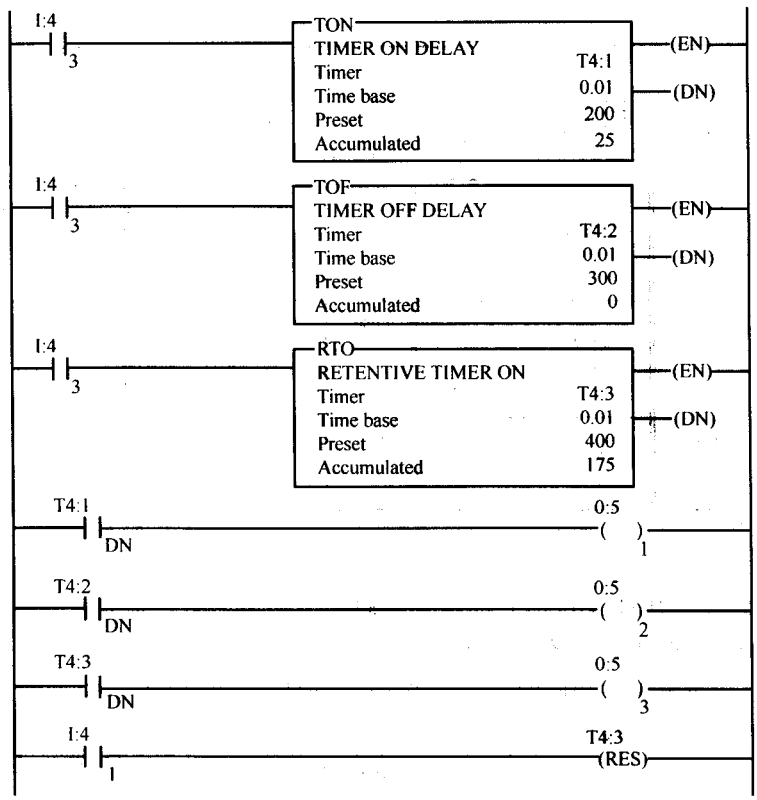
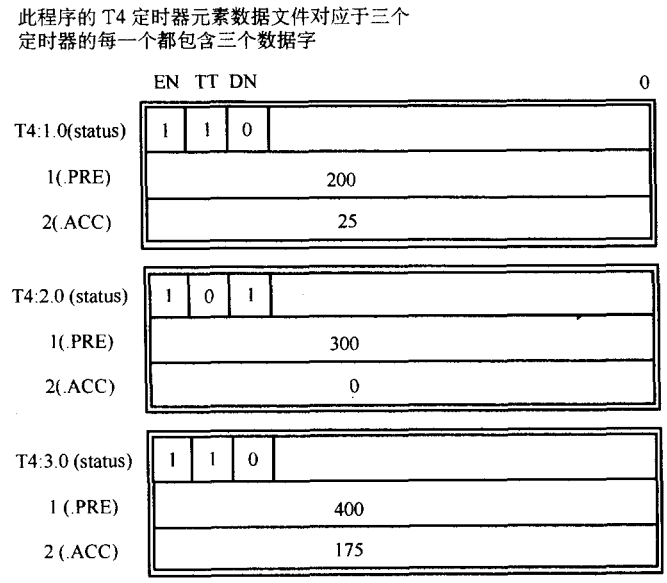


图 4-3 Allen-Bradley 定时器指令、定时器要素结构体以及定时器操作表（未完待续）

Allen-Bradley 有三个定时器指令和一个定时器复位指令，这些指令的用途如图 4-3 所示。三个定时器指令的控制逻辑对它们的影响如图 4-3 的表格所示。



定时器类型	复位累加值和状态位当：	DN 位，当复位	定时器，当控制逻辑是：	DN 位，当定时达到预设值	TT 位，当定时达到预设值	DN 位，当累加值等于预设值
TON	控制逻辑变为假	0	True	0	1	1
TOF	控制逻辑变为真	1	False	1	1	0
RTO	RES 指令执行	0	True	0	1	1

图 4-3（续）

1）每当定时器开延时（TON）的控制逻辑变为假时，累计值及所有状态位自动复位，且当输入逻辑变为真时重新计数。只有在执行定时器指令时才能更新累计值；要提高更新频率可在用户程序里重复编入相同的指令。当定时器时间到达预设值时 TT 状态位为开，之后关闭。当累计值时间达到预设值时，DN 位打开。

2）定时器关延时（TOF）与 TON 的操作几乎完全相反。相同之处在于 TOF 在计时时也要打开 TT 位。

3）保持定时器开（RTO）定时器和 TON 定时器一样，除了当 RTO 的控制逻辑变为假时，

RTO 不复位累计值。当 RTO 定时器的控制逻辑为假、RTO 定时器“暂停”时，TT 位变为关。由于 RTO 不自动复位，RTO 定时器的累计值必须由一条复位（RES）指令复位。

注意！当定时器指令的控制逻辑为真时，如果 PLC 切换到运行模式，定时器的操作就像它们的控制逻辑已经从假切换到真一样。对于结构化编程，定时器的操作好像它们的控制逻辑已经变为假或变为真——即使逻辑语句并没有真正改变状态。

图 4-3 所示为三种定时器类型之间的一些不同点。此例中的所有定时器均由相同的简单控制逻辑控制，该控制逻辑当前为真（如 EN 位所示），且已经保持 0.25s（如 T4：1.ACC 的累计值所示）。因为 TON 的控制逻辑为真，但它的 DN 位仍未打开，所以说 TON 处于运行状态。由于 TOF 的控制逻辑为真，且 DN 位未开（DN 位在控制逻辑跳变为假时开始计时，且当计时完毕后关闭 DN 位），因此 TOF 被复位。由于 RTO 的控制逻辑最后一次跳变为假时不会复位它的累计值，因此可知 RTO 已经在之前的 1.50s 累计值后附加计时 0.25s。RTO 的 DN 位仍未打开。当 I：4/1 打开，RTO 指令在 T4：3.ACC 的累计值复位（即变为 0）——即使 RTO 仍在计时。

4.3.2 SIEMENS S5 计数器和定时器

S5

当 Siemens 计数器或定时器启动，必须存储一个初始化预设值作为计数器或定时器的累计值。因为 Siemens 计数器和定时器仅在累计值已经减到零后才能改变状态位，所以累计值必须减量计数至零。

为更清楚地了解向定时器或计数器存储区存储初始值的 STL 语言程序，先要了解加载（LOAD，L）指令，LOAD 指令将在后续章节中详细介绍。LOAD 指令提供一个值，可以被复制到计数器或定时器的累计值。例如：

```
L FW004
SP T000
```

从存储器地址 FW004 获得一个值，并将其复制到为定时器 0（T000）保留的存储器地址。另一个例子：

```
L KC+ 15
S C001
```

在为计数器 1（C001）保留的存储器中放入计数器格式的常数（KC）15。

STEP 5 计数器指令至少需要两个布尔语句。如图 4-4 所示的 STL 和梯形图程序，一个布尔语句通过控制计数器设置（S）输入来设置一个预设值到计数器的累计值存储器地址。另一个语句控制减计数（CD）输入使累计值减少。布尔语句也可以控制梯形图计数器元素的增计数（CU）和计数器清零（R）输入，但这不是必要的。这四个计数器控制中的每一个都可在 STL 里单独编程，如图 4-4 所示，例子里的 STL 部分不包括在梯形图例子里没用到的控制输入的语句。

在至少一个其他支持 S5 的编程软件包里，增计数、减计数、计数器设置和计数器清零功能是在单独的梯形图的块里完成，仔细考虑这些功能不都是必须的，且不一定要放置在连续的程序存储器单元里。

图 4-4 列举了使用被称为计数器线圈（Cx）的计数器状态位的逻辑语句编程的两种方法。计数器线圈当累计值大于零时为开；当累计值达到零时关闭。在梯形图中，地址（如 Q5.0）

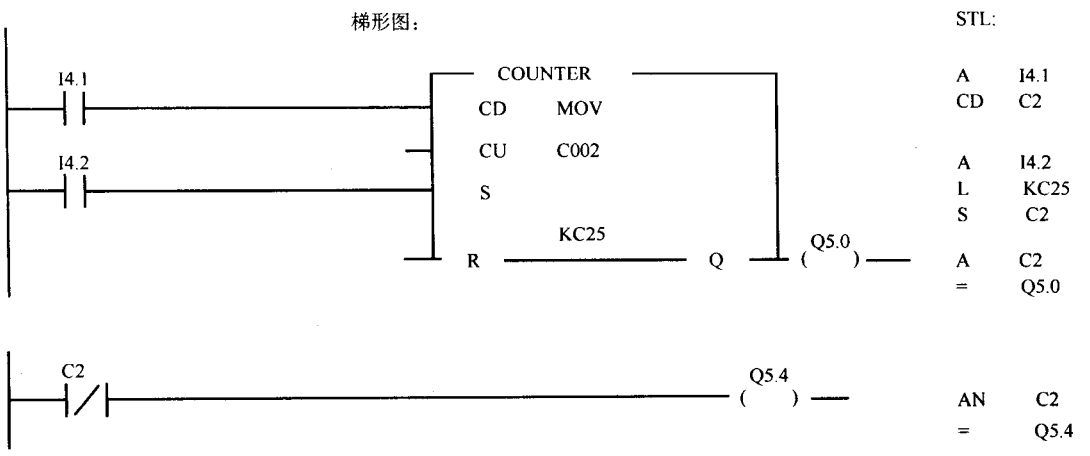


图 4-4 STEP5 计数器设置和增计数指令

可以在计数器元素的输出触点（Q）输入，或者单独的梯级可像检查布尔逻辑语句的一部分一样检查同一个计数器的状态位。对于 STL 可用同样的编程方式实现上述两种方法。

附加计数器控制指令只能在 STL 里编程，详见后文。这些附加计数器控制指令包括使能定时器/计数器（FR）、无条件置位（SU）和无条件复位（RU）指令。

一些计数器控制指令的执行过程和一次翻转法相似。每次布尔控制逻辑语句的 RLO（逻辑操作结果）从假跳变为真时，这类计数器控制指令便执行一次。计数器设置、减计数、增计数和启动定时器/计数器指令就是工作于这种方式。另一方面，计数器清零指令的执行时间为每个扫描循环中 RLO 保持为真的时间长度，而无条件置位和无条件复位（SU 和 RU）指令的执行与 RLO 无关。

由计数器设置指令写入到计数器存储器地址的初始化预设值必须在 0～999 的范围内。在梯形图中，程序员一般向梯形图的功能框输入一个计数器常量或输入一个包含预设值的存储器单元地址。在 STL 中，加载指令（L）必须出现在计数器设置指令（S）前，且 LOAD 指令必须包括计数器常量或存储器地址。计数器常量输入格式为 KCx，其中 x 为 0～999 间的常数。如果输入的是地址，则地址必须包含一个 BCD 格式的常数。（BCD 格式的三个低半码元组代表一个 0～9 之间的十进制数，高半码元组为全零）。PLC 把预设值翻译为二进制并存储在计数器的存储单元（Cx，其中 x 为计数器编号）。增计数（CU）或减计数（CD）指令分别对含此 BCD 格式的常数的存储单元增和减计数。加载（L）指令从计数器存储器中读出当前累计值，如果计数器的累计值是由加载计数器/定时器（LD）指令读出，则 PLC 在读出累计值时把它转换为 BCD 码。

启动定时器/计数器（FR）指令（仅在 STL 里可编程且只能在功能块里编程）每当 RLO 跳变为真时执行一次，使得计数器复位状态位，这些状态位包含增计数、减计数、计数器设置和计数器清零控制逻辑的最后的状态。这样可以使这些指令即使在下一次 RLO 为真、但控制逻辑语句实际上未先跳变为假时能被再次执行。

Siemens 定时器由布尔量的启动控制逻辑启动。图 4-5 所示的定时器每当启动控制 RLO 从假跳变为真时，重新加载预设值并开始减量计时。定时器启动后，PLC 必须重新执行定时器指令使累计值减量，并当定时器已经计数到零时改变定时器的定时器线圈

(Tx)。待复制的定时器线圈（Tx）的地址可输入到梯形图定时器元素的输出（Q），或定时器线圈（Tx）可当作另一个布尔逻辑语句的一部分来检查（如图 4-5 的第二个程序网络所示）。

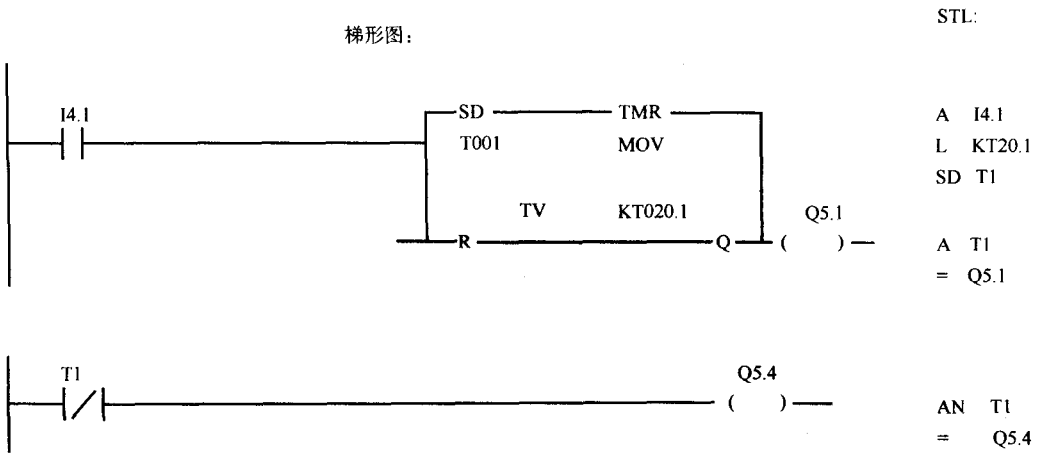


图 4-5 包含 STEP5 时器的程序，用梯形图和 STL 语言

定时器启动指令包含的初始化预设值必须为定时器常量或存储器单元地址，存储器地址必须包含定时器格式的 16 位数。定时器常量如表 4-1 所示。定时器常量必须带前缀“KT”，后跟代表时间单元数量的三位十进制数，接着是一个时间段，最后是由十进制数代码指定的时间单元大小。如果程序包含 PLC 获得定时器常量值的地址，则该地址所指的 16 位二进制字必须为定时器格式。如图 4-6 所示，定时器格式的 16 位数中的低 3 个字节必须都为 BCD 码（十进制数 0～9，则所设的最大时间为 999 个时间单元），最高 2 位必须为零，剩下的 2 位二进制数为时间单元的大小，其值为四种编码中的一个。PLC 把定时器常量和定时器格式的数字翻译为自然二进制数、并存储到定时器的存储单元（地址为 Tx，x 为定时器编号）。Siemens 不会把存储时间单元大小编码的存储单元清零。LOAD 指令用于读定时器累计值存储器的内容，若程序包括加载指令去读计时器累计值的内容，则 PLC 仅读出时间单元数量，但不恢复数字为定时器常量或定时器格式。而如果定时器值是由加载计数器/定时器（LD）指令读出，则 PLC 可以把二进制数转换为三位十进制数的 BCD 码，但指示的时间单元大小的 BCD 码将丢失。

定时器可精确到±1 个时间单元。附加的错误率可由定时器操作完成和定时器指令修改定时器状态位之间的延时、状态位改变和用户程序检查控制输出位之间的延时、扫描循环内输入（读）和输出（写）之间的延时造成。

图 4-5 例程所示为 S5 PLC 支持的梯形图开延时（SD）定时器指令。（旁边的 STL 程序不是对梯形图程序的直接翻译。）此例中，I4.1 用于控制定时器的启动。当 I4.1 从关切换到开，定时器指令向定时器 1（T001）的存储器单元置入初始化时间值，然后开始减计数。如图 4-5 所示，定时器常量 KT20.1，设置定时器计时 20 个时间单元，每个时间单元长 0.1s。梯形图定时器指令提供一个定时器复位（R）输入，当（可选的）控制逻辑语句为真时，该输入用于清除定时器的存储单元（此例中没用到复位控制逻辑）。在定时器计时到零后，开延时定时器打开定时器线圈状态位。如果程序员输入另一 Q 触点的位地址，定时器线圈状

时间 单元		
编码		大小
十进制	二进制	
0	00	0.01
1	01	0.1
2	10	1
3	11	10

例子：

定时器常量	定时器格式	定时器设置
KT205.0	0000_0010_0000_0101	2.05
KT555.1	0001_0101_0101_0101	55.5
KT125.2	0010_0001_0010_0101	125
KT999.3	0011_1001_1001_1001	9990

图 4-6 定时器编码、定时器常量、定时器格式的数字及其所表示的时间

态位会由梯形图指令复制到另一个地址。定时器 1（T001）定时器线圈位可在程序的其他位置的布尔条件中检查，此位地址 T001。

如上所述，复位（RESET，R）指令清除定时器的存储单元和状态位、并当复位控制逻辑保持为真时，使存储单元和状态位保持为零。如果定时器尚未结束计时，启动定时器/计数器（FR）指令使定时器从初始值开始重新计时。FR 指令仅能在 STL、且只能在功能块里编程，并只有当在布尔逻辑控制下 RLO 从假跳变为真的时候执行。R 和 FR 指令对不同定时器的运行有不同作用，如下所述：

Siemens 提供五种不同类型的定时器：

1）开延时（SD）定时器。这种定时器当置位（S）的 RLO 为假时，关闭定时器线圈位；当 S 逻辑变为真时设置时间值并开始计时（减计时）；计时完毕后打开定时器线圈位。在开延时定时器的启动逻辑保持为真、但定时器已工作完毕，后跟 FR 指令的 R 脉冲将重启定时器。

2）保持开延时（SS）定时器。这种定时器当置位（S）变为真时设定时值并开始计时；当打开定时器线圈位时继续运行直到结束计时，即使 S 逻辑变为假；被复位后才会重启，之后在 S 控制逻辑变为真时也能重启。当保持开延时定时器的 S 逻辑保持为真、但定时器已工作完毕，后跟 FR 指令的 R 脉冲将重启定时器。

3）脉冲（SP）定时器。这类定时器当置位（S）的 RLO 为假时关闭定时器线圈位；当 S 逻辑变为真时，设置定时值、打开定时器线圈位、并开始计时；当计时完毕时关闭定时器线圈位。定时器的 S 逻辑保持为真期间，FR 指令将重启脉冲定时器，即使定时器已经计时完毕。

4）脉冲（SE）定时器。复位、打开定时器线圈位、并当置位（S）的 RLO 变为真时开始计时；当重新关闭定时器线圈位时继续运行直到计时结束，即使它的 S 逻辑变为假。在定时器的 S 逻辑保持为真期间，FR 指令将重启脉冲定时器，即使定时器已经计时完毕。

5）关延时（SF）定时器。这类定时器当置位（S）的 RLO 变为真时打开定时器线圈位；当 S 逻辑变为假时设置定时值并开始计时；当计时完毕时关闭定时器线圈位。S 逻辑

保持为假期间，FR 指令不能在关延时定时器计时完毕后重启关延时定时器。

布尔逻辑指令：无条件置位和无条件复位（SU 和 RU）可用于打开/关闭指定的存储器位，包括计数器和定时器状态位：开或者关。这些指令的执行不受 RLO 影响，但只能出现在 STL、且只能在功能块里。

4.3.3 SIEMENS STEP 7 计数器和定时器

STEP 7 的计数器和定时器支持 S7 PLC，和 STEP5 定时器相同，除了以下两种方式：

1）如图 4-7 所示，改变了梯形图要素的外观以便符合 IEC 1131-3 标准。这里用三个单独的梯形图计数器指令代替单一的一个：增-减计数器、新的增计数器和减计数器。每个指令有一个置位（S）和复位（R）输入以及计数控制输入（CU 和/或 CD）。程序员可以在任何输入里编辑控制语句，在输入量里键入地址或常量来对计数器的预设值（PV）进行编程。（如图 4-7 所示，一个由 STEP 7 的 BCD 前缀识别的 BCD 数已作为参数写入定时器）。当前值（CV）和 BCD 码的当前值（CU _ BCD）有附加输出。如果程序员在输出端输入地址，则指令将复制计数器的当前值到输入的地址。

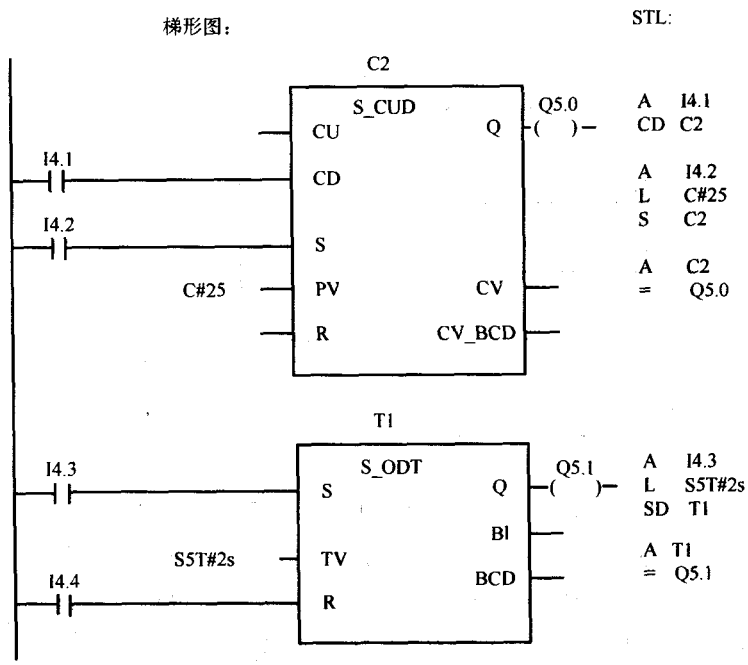


图 4-7 S7/IEC 1131-3 梯形图计数器和定时器指令

梯形图定时器指令修改后，允许定时器把它的剩余时间值输出到普通二进制或 BCD 格式的特定地址。定时器元素仍允许用逻辑语句编程来控制启动（S）输入和复位（R）输入，并约束输出（Q）位。在时间值输入处输入一个常量或地址。如图 4-7 所示为以 STEP 7 的常量格式表示的 2s 输入到 S5 的定时器。

2）FR 指令只在 STL 语言里可用，但仍允许在组织块、函数或功能块里对它进行编程。

CQM1 4.3.4 OMRON CQM1 计数器和定时器

CQM1 定时器和计数器都有一系列保留的存储器地址（TC 0 到 TC5 11），这样可以保证定时器和计数器不使用相同的数字（例如，有一个名为 TIM 6 的定时器，则不能有计数器 CNT 6）。CQM1 提供常规定时器/计数器和高速定时器/计数器。高速定时器/计数器将在第 11 章讨论，与此相关的内容如对照表指令和脉冲输出指令将一起讨论。本章只讨论标准型计数器和简单标准型定时器两种类型。

CQM1 计数器其中一种类型为可逆计数器（CNTR（12）），如图 4-8 所示。当在复位（R）输入的控制逻辑变为真时，计数器的值“0”将复制到 TC 区域的为计数器的当前值（PV）保留的存储器单元。PV 值保持为零直至 R 控制逻辑变为假。若复位（R）逻辑为关，则在每次 II 输入的控制逻辑变为真时，计数器的 PV 值增 1，或每次计数器的 DI 输入的控制逻辑变为真时，PV 值减 1。（如果 II 和 DI 输入逻辑条件两者同时都变为真，则 PV 值不改变。）CNTR（12）指令的 PV 值是循环的：可以减量到 0 以下，这种情况下 PV 自动重新载入设定值（SV）并由设定值开始减量计数；或可以在 SV 值上增量计数，在这情况下，PV 值跳到 0 并从 0 开始增量计数。当计数已减少至小于 0、等于 SV，或已增量至大于 SV、等于 0 时，CNTR（12）指令的完成标志（CNT 6）变为开，但将一直保持开到下一次计数改变。计数器的 SV 值可以输入作为 CNTR（12）指令的一个参数，SV 值可以是一个四位十进制数常量（例如，＃0025，如图 4-8 所示）或是存储 16 位 BCD 码的地址。

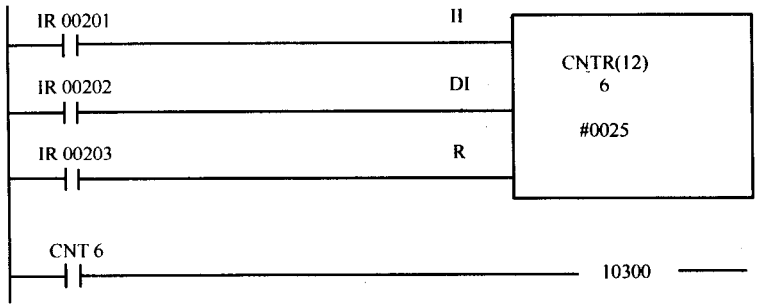


图 4-8 CQM1 保持计数器和使用了保持计数器的完成标志的一条指令

其他程序指令可以通过寻址计数器的存储器单元 CNT x 读或写计数器的 PV，x 为计数器的惟一编号。（PV 值须是 BCD 格式。）计数器的完成标志也可以通过对 CNTx 寻址来获得。CQM1 通过指令类型区分 CNTx 代表 PV 还是完成标志。

更简单的计数器（CNT）指令是一个减计数器。该计数器只能减计数，但不能小于 0。当在复位输入的控制条件变为真时，将设定值复制到计数器的 PV 值的存储器单元（且在 R 为真的持续时间内不改变）。当 PV 值为 0，CNT 指令的完成标志打开。

对于计数器，CQM1 只提供开延时类型。输入定时器（TIM）指令时需要带有一个惟一的定时器识别码，设定值（SV）是作为定时器指令的一个参数输入。SV 值可以是一个四位十进制数常量或存储 16 位 BCD 码的地址，SV 值代表定时器的运行时间，数量级为 10s（例如，＃0125 代表 12.5s）。每当定时器单输入的控制逻辑变为假时，SV 值就复制到为定时器当前值（PV）保留的存储器单元，且定时器的完成标志置为关。当控制逻辑变为真时，定

时器的 PV 开始减量计数。当 PV 值达到零时，定时器的完成位打开（即使定时器指令从 PV 值减至零后未执行过）。在其他 COM1 指令里可以用 TIM x 对定时器的 PV 里保存的时间值寻址，TIM x 还可以用于定时器的完成位寻址，这里 x 为定时器指令输入时指定的定时器编号。

4.4 故障检修

定时器或计数器不能正常工作的最常见原因之一是：编程错误使得定时器/计数器指令的控制逻辑为假，永远不可能变为真，因此定时器/计数器指令永远得不到执行（反之亦然），即定时器永远不能启动计时或计数器不能计数。试观察当 PLC 进入运行模式后，定时器/计数器指令执行一次，但在 PLC 切换出运行模式前不能再运行的情况，这种情形和编程错误相似。有几个定时器/计数器指令只能“看到”其控制逻辑为真的可能原因：

1）程序员可能未输入任何控制逻辑，所以控制逻辑不能为假。

2）控制逻辑一直为真，可能是因为某个地方编程错误。可以在 PLC 执行期间检测控制语句，还可以考虑使用数据管理屏幕而不是仅使用程序监控屏幕。

3）在一定的条件下，结构化编程技巧可以令 PLC 跳过程序中的一部分。结构化编程技术将在第 8 章中讨论。结构化程序仅当控制逻辑为真（或假）时执行定时器/计数器指令，即使定时器/计数器的控制逻辑可能已经改变了状态。

如果仍旧认为控制逻辑状态应该改变，但又怀疑它因改变太快而不能在速度较慢的编程器观察窗口里显示，可以临时增加一条特定的计数器指令。它使用相同的逻辑，紧接在不工作的定时器/计数器前面或后面！如果新计数器的累计值不改变，则控制逻辑在程序的这一点上不改变。尝试在程序里临时给第一条梯级和最后一条梯级加上相同的逻辑控制两个新计数器。如果两个计数器都没改变，那么程序正在寻找一个未发生的状态改变。（是否存在失效的传感器或在 PLC 外的存在一条损坏的电路?）

如果一个或两个临时增加的计数器都改变了它们的累计值，则需要查找其他的错误：

1）是否对两个不同的计数器或定时器使用了相同的地址？编程软件可能会提供一个交叉引用功能部件，告诉你在其他什么地方用了这个定时器或计数器地址。有些 PLC（如 OMRON CQM1）对定时器和计数器使用同一组地址，所以要注意不能把相同的地址数字用于一个计数器后再赋给另一个定时器！

2）检查影响定时器/计数器状态位的布尔逻辑语句。（这些条件可能被定时器/计数器的输出位覆盖了。）

3）检查复位定时器的语句。看这些语句是否禁止了定时器工作？试试临时加入某种逻辑语句使复位逻辑为假，观察定时器/计数器是否恒为工作状态？注意，复位定时器/计数器的状态位和累计值不能使定时器/计数器再次计时/计数；计时逻辑和定时器启动逻辑的状态必须改变（或用 Siemens STL 指令 FR 使定时器/计数器认为控制逻辑已经改变了状态）。

4）检查可能影响定时器/计数器的累计值或预设值（如果预设值可以改变）指令。有几条指令可以改变这些值，包括 MOVE 指令。

5）如果 PLC 设定定时器/计数器值减量计数至零，检查这些值是否正确设置为非零以

允许计时/计数？当定时器/计数器试图对初始值减量计数时，这些值是否反复设置为初始值？注意在 PLC 运行时监视累计值。

6）若 PLC 设计为增计数/计时至预设值，检查计数器/定时器累计值是否复位为零，以允许计数器/定时器工作？当计数器/定时器试图增计数时，这些值是否反复复位（为零）？

如果是定时器出错，可能是因为误用了定时器类型或定时器状态位。注意，当控制逻辑变为假时，绝大多数的定时器会自动复位。若控制逻辑限制定时器不能运行足够长的时间，则可以加上一个锁定以保持 Allen-Bradley 定时器或 OMRON 定时器的逻辑为真，或使用不要求控制逻辑保持为真的 Siemens 定时器。当定时器的状态位显示定时器已经运行完毕时可解除其逻辑锁定。（笔者的学生在第一次遇到开延时定时器出错时，尝试对关延时定时器编程以避开问题，但这是毫无帮助的，因为他们即使使用关延时定时器最终也会碰到同样的问题，并且可能更多。）

另一个常见的定时器和计数器问题是它们的累计值在不应改变时改变了。每个计数器/定时器都有其单独的存储单元保存累计值，并有少数状态位在保留的存储单元，记住这点可以避免以下错误：

1）每个定时器或计数器必须有其独有的识别码，识别码可以使 PLC 确定保存定时器/计数器的累计值和状态位的地址。不要把一个定时器/计数器地址再用于另一个定时器/计数器，或用两条指令试图改变同一累计值或用相同的状态位！大多数 PLC 给定时器和计数器保留各自的存储区域所以你可以同时使用定时器 12 和计数器 12，但 OMROM PLC 对于定时器时计数器只有一个存储区。若 OMRON 程序已有定时器 12，则不能再有计数器 12。

2）不要用多于一条指令对一个单独的计数器增计数（或用多于一条指令对计数器减计数），这是因为只有一个状态位可以再调用最近的增计数的状态（另外还有一个状态位可再调用减计数的状态）。如果用两条逻辑语句来增计数，其中一条逻辑语句为真而另一条为假，每次程序扫描时 PLC 会检测到增计数的状态位从假到真的改变，并在每次程序扫描时给对应的计数器增加累计值。

习题

1. 定时器是如何计数的？
2. 与 PLC 的类型相关，预设值可以保存也可以不保存，但所有 PLC 都需要为一个定时器或计数器的_____值存储一个值。
3. 在怎样的布尔条件下可以：
   （1）增计数的计数器只计数一次？
   （2）开延时定时器开始计时并连续计时？
   （3）关延时定时器开始计时并继续计时？
4. 保持型开延时定时器和标准开延时定时器如何区分？（提出至少两个不同点）
5. 如果（保持）定时器已经设置时间单元为 1s，且定时器的逻辑已经持续一个 3.5s 的时间间隔及 10 个 1/2s 的时间间隔为真，则现在定时器的累计值寄存器里的时间是多少？
6. 如果计数器存储一个 16 位有符号数作为累计值，则计数器的计数范围是什么？
7. 完成下面关于非保持型开延时定时器和关延时定时器的输出（开或关）表。所有定时器都设置为 5s 运行时间。

	定时器输出	
输入条件	开延时	关延时
10s 前跳变为真		
1s 前跳变为假		
6s 前跳变为假		
1s 前跳变为真		
6s 前跳变为真		

编程练习（不需要 PLC）

1. 画一个 STEP 7 梯形图网络，包括一个完整的增一减计数器，可以在输入（I4.0）变为开 4 次后关闭一个输出（Q5.0）。在网络里添上其他任意的必要部件，但“只能是绝对必要”的。解释网络里的计数器中每个未用到的触点的用途。
2. 写一个你常用的 PLC 的程序，注意使用适当的地址。此程序可以：
   （1）计算经过传送带上经过传感器而被检测到的物体数量。
   （2）计算传感器每次被阻塞的时间，并在传感器阻塞多于 2s 时打开一个输出。
   （3）计算定时器达到 2s 的次数。
   （4）在每次定时器超过 2s 时，复位定时器从而反复计时，（因此 3）部分的计数器每隔 2s 计数一次）。
   （5）当 3）部分的计数器达到 100 时打开报警器。

推荐的 PLC 实验室练习

对于一个系统有：

- 四个控制面板开关：输入 0 到 3
- 四个指示灯或可视输出模式灯 LED：输出 A 到 D
- 两个弹簧复位阀门控制气缸：输出 E 和 F
- 一个锁销阀门控制气缸：输出 G 和 H
- 三个传感器检测每个气缸的伸展

写一个 PLC 程序：

（1）每次开关 0 打开，准确地伸展气缸 E 4s。无论开关持续为开多长时间气缸伸展都为 4s。

（2）气缸 E 伸展后，准确地伸展气缸 F 2s，并当气缸 E 压缩时气缸 F 也压缩。用气缸 E 的传感器控制定时器。

（3）开关 0 为关期间，用一个保持定时器跟踪气缸 F 持续为开的时间。当计数超过 10s 时，打开显示灯 A。当开关 1 变为开时，复位计时值。

（4）写一个计算开关 0 的动作次数的计数器程序。开关 1 令计数器反方向计数。开关 2 把计数值复位为初始值。

（5）若开关 0 动作比开关 1 多三次，气缸 G－H 先伸展 3s，然后压缩。（气缸 G－H 需要靠单独的螺线管的激励来伸展和压缩。注意不能同时向两个螺线管输出，对每个螺线管的激励只能是 1s）。

第5章　存储器组织和数据操作

5.1　学习目标

本章您将了解到：

- PLC 存储数据单元的类型和大小，从位到结构体和文件；
- 在用户程序中可能存在的寻址方式：常数寻址、绝对寻址、符号寻址、变址寻址和间接寻址；
- 指针和参数传递；
- 举例说明对 Allen-Bradley、Siemens 和 OMORON PLC 的存储器寻址；
- IEC 1131-3 数据类型、变量声明和功能块中的立即数。（如在 Siemens STEP 7 中执行）

最早的 PLC 只可以操作输入和输出映像数据位的开关状态。现代的 PLC 能操作整个数据字节和设置整个数据字节。在本章中，我们讨论在现代 PLC 中数据如何存储，数据是如何寻址的。对这些数据操作的指令将在以后的章节中说明。

5.2　存储器概述

PLC CPU 模块的存储器中必须包含程序和数据，PLC 需要这些程序和数据来进行操作。存储器分为以下三种类型：

1）用户不可以访问的部分。存储器的这个区域包含 PLC 的专有操作系统程序和数据，这些由 PLC 提供。

2）存储器的第二个区域存储用户程序和配置数据的特殊类型，只能使用编程器才能改变它们。当 PLC 在运行模式，通常它们是不可以被改变的。在典型的 PLC 中，这部分很少能让用户程序访问。这部分存储器可以被寻址，可以被用户程序读其中内容，但是不能够改变其内容。

3）存储器的第三部分是完全可以被访问的，这意味着在用户程序中的指令可以读此存储器并且可以向其中写入内容。输出映像存储器一定是可以访问的，当然，一些 PLC 允许用户程序改变存储器的输入映像区域的内容（直到扫描循环重写用户程序，它才变化）。附加的可寻址存储区域可用来储存工作数据，这些数据包括计数器值、定时器值和其他的数据值。PLC 某些部分的配置存储器是可以寻址的，因而 PLC 可以通过编程动态地改变工作特性（当它运行时）。

除了 CPU 模块中的存储器，一些近期的 PLC 带有 I/O 模块，这些模块包含数据存储器，它们可以通过用户程序直接地寻址。处理器模块中的数据存储器芯片可以是标准的动态随机存取存储器，在掉电的时候，会丢失数据。现代的 PLC 通常内部用电池来给动态随机存取储存器供电，在外部电源断开的时候，可以保护数据不丢失。但是也有一部分 PLC 存储器没有内部电池保护。一些 PLC 带有电容，可以有充足的电能来保证在短期断电时避免数据丢失。最近的 PLC 包含闪存芯片，能在不需要电容器或电池的情况下，保存数据。在

PLC 掉电的情况下，数据不会丢失的存储器叫做非易失性存储器或者保持性存储器。尽管现在可以把所有的 PLC 存储器都变成非易失性存储器，但是一些 PLC 的生产商仍然将存储器或者存储器的一部分设计为易失性的。当 PLC 上电时，或有时当 PLC 进入运行模式后，PLC 会清除那些存储器区域中的内容。我们将在第 11 章的初始化中断一节中讨论存储器内容的清除。在第 10 章中讨论存储器使用的选项问题。

大多数 PLC 的编程器包括一个数据监视屏幕，用来给程序员或 PLC 操作员监控存储器的内容，即使在 PLC 程序运行时也可以。在许多 PLC 中，只需要简单地打入 [Alt] [D] 就可以启动数据监视屏幕。由扫描循环控制的一些程序数据（例如，输入映像数据和用户程序写入的存储空间），其他一些数据可以在 PLC 运行的时候，通过数据监视屏幕来改变！当操作员修改工作数据后，用户程序会动态地调整 PLC 的工作状态，使其符合要求。

5.3 数据类型

通过标识符可以访问 PLC 中的每个可寻址存储位置，这个标识符指明了各存储器区域是保留给哪些类型的数据的。地址标识符也表明那种数据类型的数据项需要多少比特数据。

在程序中，如果数据作为常数输入，通常包括前缀，用来识别数据的格式。这样在数据输入后，PLC 会知道该如何把它编译成二进制数据。

1）位　由一个二进制的数字组成。位时常被描述为布尔型数据。现代计算机不读/写存储器的单一位。位以集合（字节或字）的形式存在存储器中，因此，每个位的地址是每一个要被读取或写入的字节或字的地址，因而一定要指明哪一位被使用。

2）字节　由 8 个位组成，可以存储一组 8 个独立的位，但是通常用来存储一个在 0 和 255 之间的无符号整型数。字通常是 16 位。一个 16 位的存储单元可以储存 16 个独立的位或一个无符号的二进制数，代表 0 到 65 535 之间的无符号整型数，或者存储一个有符号的二进制数，表示－32 768 和＋32 767 之间的有符号整型数。更新的 PLC 提供双字数据元素，有 32 位，可以存储大约－20 亿到＋20 亿间的有符号整型数据，或者大约从 0 到 40 亿间的无符号整型数据。

3）几种 PLC 在存储器中保留了 16 位字，用来存储定时器或计数器格式的数据，这些数据由这些 PLC 的定时器和计数器指令使用。PLC 生产商已经发展了他们自己的系统，但是典型的定时器字用 16 位字的部分位存储累计值，其他位用来指示状态，例如定时器/计数器是否达到它的预先设定值。计数器和定时器数的格式有向标准化发展的趋势。

4）一些 PLC 用 32 位存储单元来保存浮点数。IEEE 754 格式通常许可 32 位数表示从 $\pm 1.1754944 \times 10^{-38}$ 到 $\pm 3.4028238 \times 10^{+38}$ 范围内的实数。仍然有其他的 PLC 为保存更大的科学计数法而使用 64 位的储存格式。

5）ASCII 码代表键盘字符，通常被称为字符型数据元素。一个 ASCII 码需要 8 位存储空间。（1 个字节）

6）几款近期的 PLC 允许程序员存储一组相同类型的数据，这些相同的数据叫做数组（举例来说：整数或浮点数），然后把这个数组作为一个单一的数据来操作。通常把 ASCII 码组成的数组叫做字符串。

7）一组不同类型的数据元素组成的集合叫做结构体，在一些现代的 PLC 中结构体可以

当作单一的数据单元来操作。结构体中可以包含数组，甚至其他结构体。在一些 PLC 中，定时器和计数器数据储存在结构体中。

8）指针是存储器地址，以 PLC 可以识别的格式存储在数据存储器中。指针总是用于下节将要介绍的间接寻址。

PLC 的生产商提供存储区域来存储指定类型的数据，这并不意味着程序员总是由于这个原因而使用存储器。无法阻止用户程序拷贝整字输入映像数据到用来存储有符号整型数据的区域中，也无法阻止用户通过编写布尔型指令来检查或者改变包含整型数或者 ASCII 码型数据存储器独立的位。(PLC 生产商通常在它们的编程软件中加入保护功能，举例来说，阻止程序改变输入映像位或 ASCII 码的一部分。使用者通常可以取消这些保护措施)。

一些指令只能对特定类型的数据进行操作，如果输入错误的地址符号，这些指令将不能工作。(事实上，大多数的编程软件不允许输入错误类型的数据)。举例来说，布尔类型的 EXAMINE ON 指令需要地址的单一位，而 MOVE 指令（在大多数的 PLC 程序语言中都是这样）需要一个数据字节、字、双字或浮点数作为目的数。布尔指令或 MOVE 指令不能操作数组或者结构体。

5.4 寻址方式

在程序中输入数据存储器地址时，PLC 现在允许程序员选择下面几种寻址方式：

1）常数是程序中的一个数，程序员不需要知道它在程序存储器中所占的地址。但是，因为常数占用电脑中一个存储单元的位置，所以是一种寻址方式，这种方式是常数寻址。

2）在程序中，最常见的用来指示数据存储器地址的寻址方式是绝对寻址。绝对地址是数据存储的实际地址，通常输入一个字母与数字混合编排的前缀来识别在存储区域的数据类型，紧接着的数字确定在这个存储区域中寻址的确切位置。所有的 PLC 都允许数据的绝对寻址。

3）符号寻址是绝对寻址的一种变化形式。程序员使用编程器输入一个表单，用来识别各个绝对寻址的符号名称。在输入这个表单之后，程序员就可以在输入程序时使用符号名来代替存储器地址。在把程序传送给 PLC 之前，编程器将用符号代替实际的绝对地址。

4）一些 PLC 现在提供变址寻址，或者允许程序员使用内部可改变的数据块。这两种寻址技术很相似。使用变址寻址，用户程序必须先要将一个偏移量放入存储器保留给所有偏移量。(一些指令会自动地执行)。任何时候当 PLC 指令遇到包含变址寻址符号的基地址，PLC 把偏移量加到基地址上，计算出被操作数据的绝对地址。(例如，在 Allen-bradley PLC 中，任何前面带有“#”的地址都被认作变址寻址)。为了使用数据块，用户程序必须首先“调用”其中一个数据块（从而建立一个偏移地址)。程序中数据字的地址，默认是从存储器中偏移区域的起始位置开始计算。

5）一些 PLC 提供间接寻址方式。在任何时候 PLC 遇到以间接格式输入的地址时，PLC 将会读取指定地址单元中的内容，并把它的值作为将要处理的数据项的绝对地址。第一个存储位置包含一个指针用来指到第二个存储位置。(在 Siemens S7 PLC 中，在地址前面标上“#P”是使 PLC 将存储单元的内容作为间接地址的一种办法。)

6）一些 PLC 编程软件包允许定义形式操作数名来代表一个程序向另一个被其调用的程序（“子程序”或者“函数”，等等）提供的数值。字母与数字混编的形式操作数名用来代替被调用程序中的实际地址。IEC 1131-3 标准包括这样的要求，就是每个程序或者函数应该

有一个存储区域，这个区域只能被这个程序或者函数使用，而不能被其他的程序或函数使用。随着这个 IEC 1131-3 要求的被执行，更多的 PLC 程序语言将会允许使用形式操作数。

5.5 PLC 中可寻址的数据存储

在第 2 章中我们讨论了输入和输出映像寻址，在第 4 章中我们讨论了计数器和定时器寻址，因此我们已经介绍了一些寻址协议。在这一节中，我们重新复习这些寻址格式，并且介绍其他的一些可寻址的存储器，以及在我们选择的流行的 PLC 中可以使用的寻址方式。

在这一节中，将通过介绍 MOVE 指令来显示这些选出的 PLC 存储数据的能力，尽管 MOVE 指令将在第 6 章正式地阐述。MOVE 指令可以把在存储单元中的数据从一个存储单元拷贝到另一个存储单元，而不需要改变源存储单元的内容。(MOVE 指令实际上是一个拷贝指令)。

PLC-5 SLC 500

5.5.1 ALLEN-BRADLEY PLC 中的数据文件和可寻址数据

Allen- Bradley PLC-5 和 SLC 500 PLC 有非常相似的存储结构。在这一节中，我们将介绍相似的数据寻址特征。它们之间有哪些重要的差别将在后面几节分别介绍。

在这两种 Allen-Bradley PLC 中，CPU 存储器包含一个处理器文件，它由 PLC 在运行时需要的程序文件和数据文件组成。程序文件包括用户程序和一些操作系统配置信息，PLC 需要它们才能运行。(程序文件不在这章中讨论)。数据文件包括程序文件需要的用户数据和系统配置数据。Allen-Bradley PLC 的数据文件中存储的数据一直保持不变，直到程序专门改变这些数据或者清除这些数据，只要存储器后备电池工作，甚至 PLC 不在运行状态，或者把 PLC 的电源断开，这些数据也会保持不变。Allen- Bradley PLC 不同于本书中讨论的其他 PLC，因为其他 PLC 有非保持性数据存储器。当 PLC 不在运行状态或者它们的电源断开时，非保持性存储器的内容会由 PLC 自动清除。

1. Allen-Bradley 常数数据值

常数可以直接输入到一些用户程序指令中，而不需要输入数据值的存储地址。下面的值可以直接输入。在一些情况下，必须加前缀：

1) 有符号的十进制整型数的值在－32 768 到＋32 767 范围内是可以接受的。

2) 如果 PLC (和指令) 能够使用浮点数的话 (不是所有的 SLC 500 都可以)，更大的十进制数可以被输入，可以有小数点或者没有。举例来说，9876543210 是可接受的，就像－10.5一样。可接受数的大小是有限制的，其极限与下面所述的科学计数法相同。

3) 非常大或非常小的十进制数，在允许的范围内，可以用科学计数法 (例如，9.87E25，意思是 9.87×10^{25}，或者 －9.8E-25，意思是 -9.8×10^{-25})，但是只有在符合 IEEE 754 范围内的 32 位浮点数可以表示 (1.17549944×10^{-38}到 $3.4028238\times10^{+38}$)。

4) 如果正确地使用前缀，十六进制、八进制、二进制和 ASCII 常数在一些指令中是可以接受的。前缀是：

&H	十六制	从 0 到 FFFF(例如，&HFF12)
&O	八进制	从 0 到 177777(例如，&O177000)
&B	二进制	(例如，&B1111000011110000)
&A	ASCII 码	(例如，&Ahi)

2. Allen-Bradley 的数据文件

那些不用输入到用户程序中的数据作为常数必须在数据文件中的可寻址存储器中。每

个数据文件用来储存一个单一类型的数据。三种类型的数据文件对 PLC 来说是至关重要的：

O 输出映像数据字；在这一节中通过 PLC-5和 SLC 500处理器来介绍
I 输入映像数据字；在这一节中通过 PLC-5和 SLC 500处理器来介绍
S 状态字；包含对 PLC 的配置数据和描述 PLC 运行情况的状态数据

附加的数据文件可用来存储专门类型的用户数据。这些附加数据文件中的一部分可以默认为是用来存储专门的数据类型的。默认数据类型包括：

B3 十六位数据字中的位存储
T4 有三个数据字的定时器数据结构体
C5 有三个数据字的计数器数据结构体
R6 有三个数据字的控制数据结构体
N7 整型数据字存储
F8 在 PLC-5和较好版本的 SLC 500中的浮点数存储

SLC 500 中的数据文件，从 10 到 255，或者 PLC-5 中的文件，从 9 到 999，也是可用的，但是这些数据文件没有默认的数据类型。它们可以用来附加存储位、定时器、计数器、控制元素、整型数、浮点数或者在以下的小节介绍的关于 SLC 500 和 PLC-5 的任何其他数据类型。在编程过程中输入可用数据文件的文件序号，可以自动地保存那些数据文件，它们用来存储指定文件序号的数据类型。例如，创建数据文件 12 作为位存储的附加数据文件，写一个使用下面地址的程序：

```
B12/x or B12:x or B12:x/x
```

这将保留文件 12 给位存储使用。("x" 也必须是一个有效的位或者字地址。位文件的地址的结构体将在下面描述)。通过在编程器中的文件创建特性，数据文件也可以保留给指定的数据类型。

3. Allen-Bradley 的状态文件

默认状态文件是惟一存在的状态文件。它包含 Allen-Bradley 定义的状态字。每个 16 位的状态字包含 PLC 的配置数据或者在 PLC 运行的时候描述 PLC 运行状态的状态数据。任何状态字都可以通过用户程序检查。大多数状态字是动态的，用户程序可以往状态字写内容，当用户程序运行的时候，可能会改变 PLC 的配置。另一些状态字是静态的，只有在 PLC 处于编程模式时才可以改变，或者偶尔，用户程序改变它们，但 PLC 只有在下次进入运行模式的时候才可以识别它们。状态字以下面的形式寻址：

S:e "e"是16位状态字元素的序号(例如，S：1是一个数据字，包含16个处理器状态位)

首个状态字编号为 0，最后一个由 SLC 500 或者 PLC-5 的型号决定。SLC 500 和 PCL-5 的状态字在附录 A 和 B 中列出。注意 PLC-5 和 SLC 500 对于相同的功能使用相同的状态字地址，这是可能的，但状态文件并不相同。

每个状态位可以用下面的形式在布尔逻辑条件中寻址。

S：e/b⊖ "b"是从0到15中的一个数，表明操作哪个位(例如，S：1/15是字1的最高位；这个重要的状态位可以在每次 PLC 进入运行模式时保持开一个扫描循环)。

4. Allen-Bradley 的位文件

位文件用来作位存储。文件 3 默认保留给位文件，它的地址前缀是 "B3"。编程保存数

⊖ 只有在 PLC-5 中，程序员可以输入一个点 (.) 来代替斜杠。

据文件作为另外的位存储的过程中，前缀“B”可以被任何没有使用过的文件序号使用。当用作位存储的时候，在位文件 3 中的单一位通常以以下形式寻址：

B3/b “b”是在 SLC 500中从0到4096间的数，或者在 PLC-5中从0到15999中的数。

Allen-Bradley 允许一个替代的方法寻址位存储文件中的位。认识到所有的位实际是 16 位数据字的一部分，可以用下面的形式寻址：

B3：e/b “e”是字元素，在 SLC 500中从0到255，或者在 PLC-5中从0到999；“b”是从0到15的数值，表明16位字中的一个位。

图 5-1 显示了位文件存储器的存储空间，并且展示了寻址同一个位的两种方法。例如，“B3/20”意思是文件 B3 的第 21 个位。这个位也可以用“B3：1/4”表示，意思是第二个状态字的第 5 个位。

位文件由 16 位的状态字组成，可以作为整个字操作。如果按照以下方式寻址，整个“位”数据字可以被操作：

B3：e “e”是字元素序号，在 SLC 500中是从0到255，在 PLC-5从0到999。

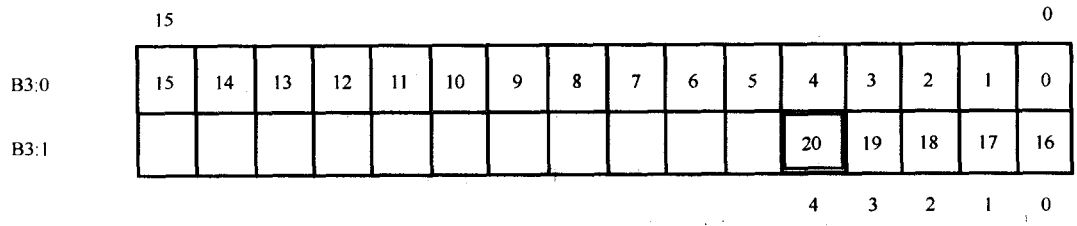


图 5-1 两种方法寻址相同的 Allen-Bradley 位文件的数据位

5. Allen-Bradley 的定时器结构体文件

文件 4 默认保留给定时器数据结构体，并且所有在这个文件中的数据寻址必须有前缀“T4”。每一个定时器元素结构体由 3 个 16 位数据字组成，如图 5-2 所示。

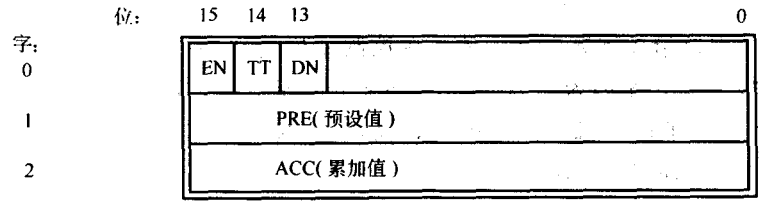


图 5-2 在 Allen-Bradley PLC 中的定时器数据结构体

当使用定时器指令编程的时候，程序员以下面形式输入整个定时器结构体的地址。

T4：e “e”是3字结构体元素的序号(在 SLC 500中从0到255，在 PLC-5中从0到999。)

在定时器数据结构体中的每个数据字可以作为字来寻址的，如以下形式：

T4：e.m “e”是3字结构体元素的序号(在 SLC 500中从0到255，在 PLC-5中从0到999)。“m”是 Allen-Bradley 定义的助记词，表示定时器结构体中的一个数据字(例如，T4：3.ACC 是为定时器3的累加值寻址的一种方法)

早期的 Allen-Bradley PLC，程序员不能够使用助记词，但是不得不用数字寻址数据字，如下：

T4：e.s “s”是3字子元素(0到2)的序号(例如,T4：3.2是定时器3的累计值的地址)。

一些程序设计软件允许程序员选择数字或寄存器寻址。程序员应该学习寄存器方法，因为一些程序设计软件需要在定时器结构体寻址中使用寄存器。

在3字结构体中的各个位可以用下面格式寻址：

T4：e/m “m”是Allen-Bradley定义的字0中的可寻址状态位的寄存器(e.g.,T4：3/TT是另一种寻址定时器3的TT位的方法)

T4：e.m 在PLC-5中,使用点(.)代替斜线(/)是允许的,并且甚至是Allen-Bradley建议使用的(SLC 500编程中不使用点)

6. Allen-Bradley 计数器结构体文件

文件5默认保留给计数器数据结构体。当在这个文件中寻址数据的时候，前缀“C5”必须使用。每个3字结构体以如图5-3所示的格式保存。

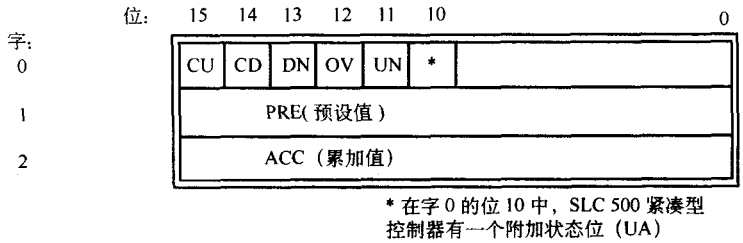


图5-3 在Allen-Bradley PLC中的计数器数据结构体

计数器数据寻址和定时器数据寻址类似，除了用“C5”代替“T4”，如下：

C5：e 3字计数器元素
C5：e.m 结构体的数据字子元素
C5：e/m 字子元素0的单一状态位(在SLC 500中)
C5：e.m 字子元素0的单一状态位(在PLC-5中)

可以在SLC 500程序中使用数字值代替寄存器值，但是程序员应该学习寄存器方法。

7. Allen-Bradley 的控制结构体文件

文件6默认保留给控制数据结构体。在控制结构体文件中的数据必须有前缀“R6”来寻址（使用前缀“R”，是因为在Allen-Bradley引进控制元素前，“控制（Control)”的前四个字母已经分别用作计数器（Counter)、输出（Output)、整型数（Integer）和定时器(Timer))。3字结构体格式如图5-4所示：

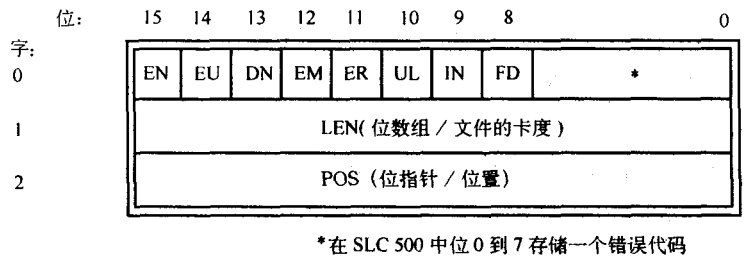


图5-4 在Allen-Bradley PLC中的控制数据结构体

控制数据可以以如下方式寻址：

R6：e　　3字计数器元素
R6：e.m　　结构体的数据字子元素
R6：e/m　　字子元素0的单一状态位(在 SLC 500中)
R6：e.m　　字子元素0的单一状态位(在 PLC-5中)

SLC 500 程序中可以使用数字值代替寄存器，但是程序员应该学习寄存器方法。

8. Allen-Bradley 整型数文件

文件 7 默认保留给－32 768 到＋32 767 之间的整型数据值的存储。每个整型数保存为一个 16 位的有符号二进制数。这个文件中的每个数据字可以如下方式寻址：

N7：e　　“e”表示文件7中的哪一个数据字(在 SLC 500中从0到255，在 PLC-5中从0到999)

数据字中的每个位可以以如下方式寻址：

N7：e/b　“b”是从0到15的位的序号

9. Allen-Bradley 浮点数结构体文件

文件 8 是 PLC-5 和好一些的 SLC 500 保留用来存储浮点数的。浮点数的范围是 $\pm 1.1754944\times10^{-38}$ 到 $\pm 3.4028238\times10^{+38}$，使用 IEEE 754 浮点数格式编码为二进制数，这种格式需要 32 位，所以每个浮点数元素是一个含有两个 16 位数据字的结构体。单独的位和 16 位数据字不可以寻址，所以浮点数存储元素可以以如下方式寻址：

F8：e　　“e”是元素序号(在 SLC 500中从0到255，在 PLC-5中从0到999)

10. Allen-Bradley ASCII 字符文件

ASCII 码字符没有默认的文件，但任何未被使用的文件都可以用来保存 ASCII 字符。ASCII 字符包括原始的标准电报键盘字符集。对每个字符，二进制 ASCII 码由 7 个位组成。(ASCII 码仍是最常用的把键盘字符二进制化的格式。) 在 Allen-Bradley 字符文件中，每个代码保存为 16 位字的一半，这样一个字可以保存两个字符。由于 Allen-Bradley 不允许字节(8 位元素) 的寻址，每个可寻址的 ASCII 码文件元素包含一对 ASCII 码字符。ASCII 文件数据以如下方式寻址：

Af：e　　“f”是文件序号，“e”是文件中字的序号(每个字是一个16位的数，包含两个 ASCII 码，代表2个 ASCII 字符)

ASCII 数据文件元素的每个位可以以如下形式寻址：

Af：e/b　　“b”是数据文件“f”(附加数据文件之一)的双 ASCII 码(在 SLC 500中从0到255，在 PLC-5中0到999)中一个位的序号。
Af：e.b　　(在 PLC-5编程中，点(.)是 Allen-Bradley 推荐的)

11. Allen-Bradley 字符串数组文件

没有默认的文件保留给字符串数据存储，但是任何没有使用的数据文件都可以用来存储。每个字符串数组由 42 个 16 位数据字组成，第一个数据字包含一个数字，用以描述存储在当前字符串中的 ASCII 码字符的实际数字。在以下的 16 位数据字中包括两个 8 位 ASCII 码，它们代表 ASCII 字符，如图 5-5 所示 (ASCII 字符如图 5-5 所示，尽管实际被保存的是 ASCII 码)。

Allen-Bradley PLC 提供操作字符串的指令。字符串元素字和位只有通过使用那些指令来操作，因为 Allen-Bradley PLC 不允许对字符串的单独的字或者位寻址，除了首字，首字中包含了字符串的长度。这个首字可以通过以下地址访问：

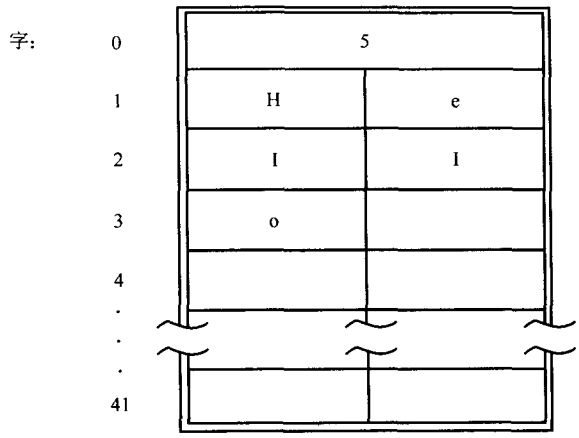


图 5-5 保存在 Allen-Bradley 字符串数据文件中的 ASCII 码字符串

STf：e.LEN "f"是文件序号,"e"是字符串元素序号

图 5-6 表示使用一些 Allen-Bradley 寻址的惯例。它显示了一个梯形图的梯级，用以检测一个位文件的第七位（首位是位 0）。如果该位为开，则梯级：

1）移动十进制常数值（0）到 16 位的位文件字中，包括 B3/16 到 B3/31，全部关闭它们。

2）拷贝文件 4 的定时器结构体 8 的第三个数据字（累计值）到一个整型数据文件中（N7：12）。

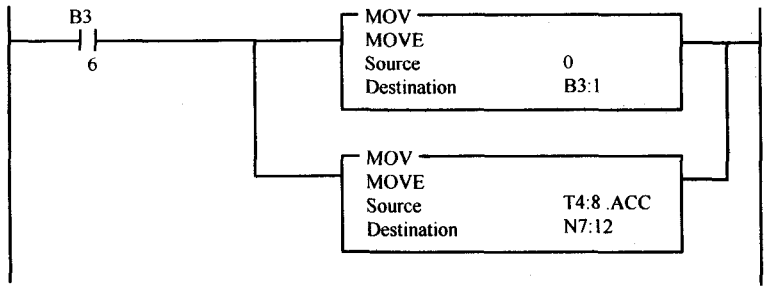


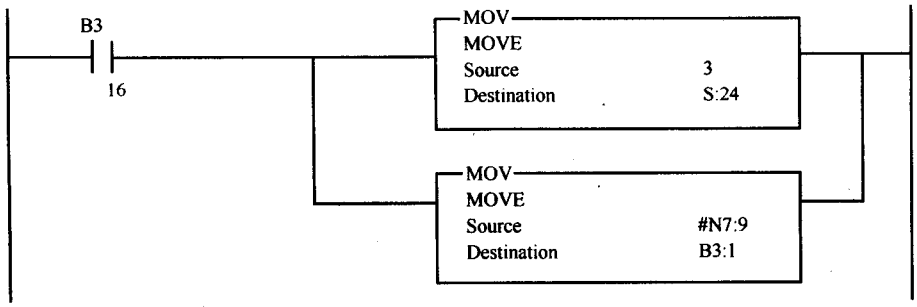
图 5-6 有条件的执行两条 MOVE 的 PLC-5 或者 SLC 500 程序梯级

12. Allen-Bradley 变址寻址

Allen-Bradley PLC 也可以编程对数据存储器使用变址寻址。如果在地址之前输入“#”，在操作这个地址之前，PLC 将计算生成一个元素数。计算包括将从状态文件字 S：24 得到的数（叫做偏移量）加到以变址寻址格式输入的元素数上（“#”和基数）。这个加法生成了被操作的数据元素的绝对地址。在图 5-7 的例程中包括一个 MOVE 指令，这个指令把偏移地址值放到 S：24，第二个 MOVE 指令包括一个变址寻址格式的基数。在图 5-7 的表中显示是如何计算将要被移动到 B3：1中数据的绝对地址，并且显示其他的变址寻址的例子。

因为在状态文件字 24（S：24）中的偏移量影响变址寻址所指的实际地址，程序员需要确保 S：24 包含适当的值，通常在用户程序中使用 MOVE 指令来把数值放到S：24中（如图

5-7 所示)。可以输入正数或者负数值(受限于 16 位有符号二进制数的表示范围),并且在重复变址寻址指令前,S:24 的值可以被改变。必须注意需要被寻址的元素是确实存在的。有一些没有地址 N7:−6,因而把 S:24 的偏移量−15 加到基地址 #N7:9将会产生一个无效的地址。如果 N7:10 是在 N7 中的最后一个数据字,N7:12 也可能是无效地址。Allen-Bradley 把生成不存在的地址称为*突破数据文件边界*,因为 N7:−6实际存在。N7:−6是先前数据文件 R6 的一个数据字。SLC 500 PLC 可以被配置成允许突破数据文件边界或当程序这样做时出错。如果数据文件边界被突破时,PLC-5 PLCs 将出错。



变址寻址	S:24 内容	实际地址
#N7:9	3	N7:(3+9)=N7:12
#N7:9	−3	N7:(−3+9)=N7:6
#N7:9	−15	见文本
#C5:2.ACC	5	C5:(5+2).ACC=C5:7.ACC

图 5-7 Allen-Bradley 变址寻址例子

一些指令自动地使用变址寻址,并在执行时修改 S:24 的内容。这些指令要求程序员在输入地址前加“#”,但不需要用户编写指令写数值到 S:24 中。在编写需要在指令中使用变址寻址的程序时,这些指令不会自动地控制 S:24 的内容,在这些指令之前立即移动期望的偏移量到 S:24 中是好的编程方法,特别是包含可以自动地改变 S:24 内容的指令的程序。

13. Allen-Bradley 间接寻址

PLC-5 和较好的 SLC 500 提供间接寻址,间接寻址在 Allen-Bradley 中叫作指针的使用。在间接寻址中,地址的数字部分可以作为包含绝对数的数据存储空间的地址来输入。(迷惑?)例如,如果 B3:4包含数 7,间接格式的地址“N[B3:4]:1”实际表示“N7:1”。方括号很重要,以便 PLC-5 可以识别间接寻址。如在下面的例子中,文件序号、字序号和二进制文件位序号都可以在地址中间接地引用。结构体中的数据字仅可以间接地在定时器、计算器和控制结构体中引用。

N[N7:1]:5	如果 N7:1包含“16.”,意味着 N16:5
B3:[N7:1]	如果 N7:1包含“16.”,意味着 B3:16
B3/[N7:1]	如果 N7:1包含“16.”,意味着位 B3/16
N9:5/[N7:1]	不工作。位的间接寻址只能用于位文件中的位
O:[N7:1]	SLC 500中,如果 N7:1包含“16.”,意味着 O:16 PLC-5中,如果 N7:1包含“16.”,意味着 O:020。指向 PLC-5的 I/O 映像地址的间接地址被转换成八进制。

(十进制的“16”= 八进制的“20.”)

C5：[B3：1].ACC 如果 B3：1包含“2.”，意味着 C5：2.ACC

C5：2.[B3：1] 如果 B3：1包含“2”，意味着 C5：1.2。在 PLC-5中不工作，因为对于地址字，必须使用寄存器寻址。

ST9：[B3：1].5 不工作。字符串元素的间接引用是不允许的

ST9：1[B3：1] 不工作。除了定时器、计数器或控制元素结构的结构字，不允许间接引用。

SC10：[B3：1].TIM

(仅 PLC-5) 如果 B3：1包含“2.”，意味着 SC10：2.TIM

14. Allen-Bradley 符号寻址

Allen-Bradley PLC 编程时也可以使用符号寻址。程序软件允许程序员定义字符与数字混编的符号名来代表绝对地址。定义符号名之后，在用户程序中可以用符号名代替绝对地址。

15. Allen-Bradley 形式参数

在 Allen-Bradley 程序中不可以输入字母与数字混编的形式参数（至少使用 PLC-5 6200 编程软件或者 SLC 500 APS 编程软件时不可以）。顺便说一下，当您阅读本书时，形式参数可能已经允许使用了。

当 Allen-Bradley PLC 程序包含“跳转到子例行程序（JSR）”指令，JSR 指令允许程序员指定输入参数地址。匹配的子例行程序（SBR）指令也允许地址当作输入参数被输入。无论何时，调用子程序，PLC 从在 SBR 指令中列出的输入参数地址中拷贝数据到在 SBR 标号中列出的输入参数地址中。

类似地，输出参数地址在一个子程序结尾的 JSR 指令和返回（RET）指令中列出。无论何时，子程序结束，PLC 拷贝在 RET 指令中列出的输出参数地址到在 JSR 指令中列出的输出参数地址。

当子程序执行时，它可以操作数据，这些数据被拷贝到输入参数地址中，而不会影响原值，并且可以写结果到输出参数地址中，而不需要过早地改变那个数据最终目的地址。

(PLC-5) 5.5.2 ALLEN BRADLEY PLC-5 中的可寻址数据

在前面的几节中，我们讨论了数据存储和 PLC-5 和 SLC 500 通用的寻址特性。在这一节中，我们讲述 PLC-5 特殊的数据存储特性。

1. Allen-Bradley PLC-5 I/O 映像数据文件

数据文件 0 是保留给输出映像数据的，并且地址必须有前缀“O：”，数据文件 1 是保留给输入映像数据的，它们必须有前缀“I：”来寻址。输出映像和输入映像数据文件的安排如图 5-8 所示。注意使用的是八进制系统，为了和早期的 Allen-Bradley PLC 兼容。

16 位输出映像和输入映像文件数据字寻址必须以如下的格式：

p：e 前缀“p”是字母 O(对输出)或者 I(对输入)；“e”是一个三数字的八进制数⊖：

- 前两个数字表示逻辑框架。
- 最后的数字表明 I/O 组，与映像相关的 I/O 模块的实际位置(PLC-5 I/O 组在第2章中描述)。

[例如，I：123是16位输入映像字的地址，与框架12的 I/O 组3中的数字输入模块相关。

例如，O：027是16位数据字的地址，与在框架02中的最后一个 I/O 组(组7)的数字输出模块相关。]

⊖ 八进制数只使用数字 0 到 7。因此框架编号为 00 到 07，10 到 17，等等。当前最大的 PLC-5 系统的框架排到 27。I/O 组编号为 0 到 7。

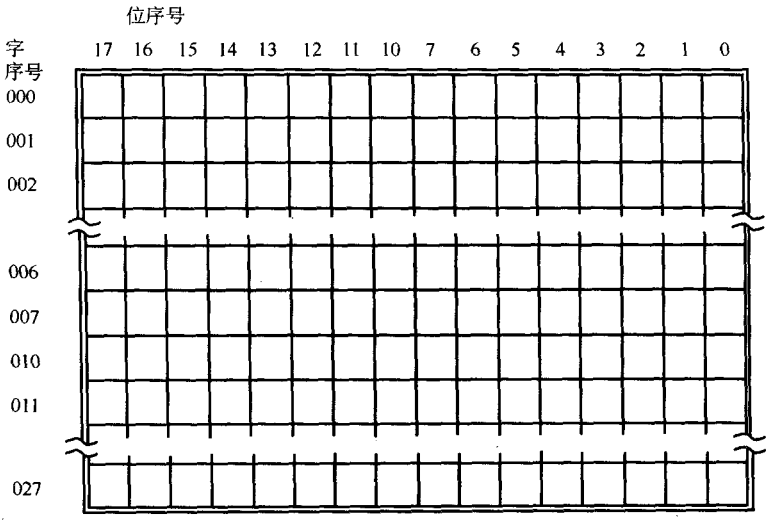


图 5-8 Allen-Bradley PLC-5 的映像文件格式

I/O 映像数据字的每个位可以以下面形式寻址：

p：e/d "d" 表明（八进制）16 个位中的哪一位将被操作（I/O 映像数据字的 16 个位的序号是从 00 到 07，从 10 到 17；

例如，I：123/04 是输入映像数据字 123 位 4 的地址；它代表一个传感器的状态，这个传感器连接到在逻辑框架 12 上 I/O 组 3 中的输入模块的第 5 个螺丝终端。）

2. Allen-Bradley PLC-5 BCD/十六进制数据文件

未使用的数据文件可以分配用来保存代表 BCD/16 进制值的 16 位的数。可以保存 4 位的 BCD/16 进制数据值（BCD 码在十进制数 0000 和 9999 间，或者 16 进制数从 0000 到 FFFF）。每 4 位数被保存作为在 16 位数（4 位=1 半位元组（nibble））中一个分开的 4 比特代码。当使用编程器来监控包括 BCD/十六进制数据字地址的程序时，编程器显示一个 4 字符的十进制/十六进制数来显示地址里的内容。当为了显示目的而将 16 位二进制数转化时，16 位二进制数不能被看成是有符号数。除了在编程器上如何显示数字，PLC-5 不会把 BCD/十六进制数区别于整型数。PLC-5 的算术操作把 BCD/十六进制数作为有符号数来代表整型数。

当一些输入和输出接口设备接收或者产生 BCD 值，它们使用一个专门的惯例来使 BCD 码代表有符号数。4 半位元组二进制数的低三个半位元组的每一个包含一个二进制数，代表从 0 到 9 的十进数字符。高半位元组使用二进制数 0000 来代表正符号，二进制数 1111 代表负符号。可以代表在－999 到＋999 间的十进制数。在 PLC-5 的编程器中，这些数将显示如 0999 和 F999。

十六进制值经常作为压缩的方法来代表二进制值，所以如果应用 16 位二进制数以十六进制而不是二进制或者十进制形式来显示，有时更容易理解。

BCD/十六进制数据字有如下寻址方式：

```
Df：e   "f"是任何可使用的数据文件(3到999),"e"表明一个16位字元素(0到999)
        (例如,D9：4是在数据文件9的数据字4中以 BCD 码格式存储的十进制数的地址)
```

BCD/16 进制数据位可以以下面的形式寻址：

```
Df：e/d "d"表明一个序号从0到15的位
        (例如,D9：4/15是被保存在文件9中的字4中的16位值的最高位;如果这个位是1,十进制数是负数)
```

3. Allen-Bradley PLC-5 块传送结构体文件

在增强型 PLC-5 中，没有被使用的输入文件可以保留来存储块传送数据元素，这些块传送数据元素由 6 字的结构体组成。当 CPU 传送数据到智能 I/O 模块，或者从智能 I/O 模块接收数据，块传送元素用作控制目的。Allen-Bradley 有时称智能 I/O 模块为块传送模块。在早期增强型 PLC-5 的经典 PLC-5 中，程序员不得不在整型数文件中使用一套 5 个数据字的集合，用来控制和智能 I/O 模块之间的每个数据交换。

在本书中没有块传送结构体格式的图表，因为程序员不需要知道它的格式。块传送数据元素的每一个数据位和字都不可以被数字寻址。它们只能通过 Allen-Bradley 寄存器寻址。块传送元素以如下方式寻址：

BTf：3　对整个的6字结构体来说，“f”是可用数据文件(从3到999)中的一个，“e”是文件中的一个元素(从0到999中)。

BTf：e.m 来自元素的16位数据字或者单独的状态位(字使用的“m”寄存器包括 RLEN，DLEN，FILE 和 WORD，单独的状态位使用的“m”寄存器包括 EN，ST，DN，ER，CO，EW，NR，TO 和 RW)

4. Allen-Bradley PLC-5 消息控制结构体文件

在增强型 PLC-5 中，未被使用的数据文件可以分配用来存储 56 字的结构体，这个结构体给消息控制数据使用。当用户程序包括 MSG（消息）指令，通过 Allen-Bradley 的一个数据高速公路局域网来传送或者接收数据，MSG 命令必须指定一个下面形式的消息控制元素，以便 PLC 可以使用这些消息控制结构体来处理数据交换：

MGf：e　“f”是可用的数据文件(从3到999)，“e”是在哪个文件中56字的消息数据控制元素的序号(从0到999)

56 字的结构体包括状态位和字，它们可以在用户程序中用其他指令寻址，形式如下：

MGf：e.m　“m”是给状态位或者16位数据字使用的寄存器

5. Allen-Bradley PLC-5 PID 结构体文件

在增强型 PLC-5 中，未被使用的数据文件可以被分配用来保存 82 字的 PID 数据结构体，PID 指令使用这些数据。PID 指令和 PID 数据结构体元素在第 12 章中详细讲述。在增强型 PLC-5 出现之前，程序员不得不在整型数文件中保留一个有 23 个数据字的块，用来存储对 PID 指令重要的数据，但现在 PID 指令可以保留并使用改进的 82 字 PID 元素，通过下面的方法寻址：

PDf：e　“f”是可用的数据文件(从0到999)，“e”是在文件中的82字的 PID 数据元素的元素序号(从0到999)

Allen-Bradley 新的 PID 数据元素的结构体包括数据字和状态位，并且也有浮点值。每个值能使用以下这种寄存器方式寻址：

PDf：e.m　“m”是存储状态位，16位数据字或32位浮点数的寄存器序号

6. Allen-Bradley PLC-5 SFC Step 定时器结构体文件

在增强型 PLC-5 中，未被使用的数据文件可以被分配来存储 SFC step 定时器数据元素，每一个由三个数据字组成。使用 SFC step 定时器，编程器可以监控以 SFC 编写的程序。在监控时显示每个 SFC step 的定时状态。用户程序也可以对 SFC 定时器元素为了每一个 SFC 步骤建立的字和状态位寻址，如以下形式：

SCf：e　三字计数器元素

SC5：e.m 数据字子元素，或者结构体的状态位(字使用的寄存器包括 PRE 和 TIM；状态位使用的寄存器包括 SA、FS、LS、OV、ER 和 DN)

5.5.3 ALLEN-BRADLEY SLC 500 中的可寻址数据

SLC 500

我们已经讨论了 PLC-5 和 SLC 500 共有的数据存储和寻址特性。在这一节中，讲述 SLC500 特别的数据存储器特性。

1. Allen-Bradley SLC 500 I/O 映像文件

输出映像数据字保持在数据文件 0 中。输入映像数据字保持在文件 1 中。在这些存储器区域中的 16 位字以如下面格式寻址：

p：e 或者 p：e.s

在这里：

- 前缀“p”是字母 O 代表输出映像数据字，或者 I 代表输入映像数据字。
- 元素序号“e”表示包含映像字所代表的输入或者输出模块的插槽。数 0 表示 CPU 模块，只有当 SLC 500 是紧凑型控制器时，CPU 模块包含 I/O 触点。从 1 到 30 的数代表 CPU 模块的插槽数。
- 只有当 I/O 模块需要多于一个的 16 位输入映像或者输出映像的时候，子元素“s”才需要。0、1、2 等表示第一、第二、第三等 16 位映像数据字。如果没有“s”值指定，首数据字（字 0）是默认的。“s”的最大值是 255，可以构造强大的专用 I/O 模块。

每个 I/O 映像数据位可以如以下格式寻址：

p：e/d 或者 p：e.s/d　“d”是被操作的二进制位的地址(从0到15)

2. Allen-Bradley SLC 500 模块数据文件（M0/M1）

SLC 500 允许数据文件寻址，这些数据文件实际上包含在 I/O 模块中。这种类型的数据文件寻址只有在安装了自带存储器的专用 I/O 模块时是可能的。如果这样，模块数据文件（M0 或 M1）可以按如下方式寻址：

Mf：e.s　“f”是数0或者1，“e”是模块所在插槽的序号(从1到30)，子元素“s”表示在模块存储器中的哪一个字将被操作(0到模块的最大值)

M0 或 M1 数据文件也可以按以下方式按位寻址：

Mf：e.s/b　“b”是位序号(从0到15)

只有在 PLC 不运行时，配置数据可以写到专用 I/O 模块中。因为这个类型的 I/O 模块数据存储器（叫做 G 文件）在用户程序中不可寻址，在这里我们不讨论。

3. Allen-Bradley SLC 500 数据文件 9

数据文件从 10 到 255，可以用来保存任何类型的数据（除了 I/O 映像、状态或者 M0/M1 数据）。读者可能会问为什么文件 9 不可用。在第 13 章将看到文件 9 有时需要留给某种特定类型的通信数据。

在图 5-9 中，SLC 500 梯形图的梯级检查在二进制文件中的位。如果这个位置为开，则梯级：

1）从插槽 8 中的输入模块移动一个 16 位的值到整型文件数据字中（N7：2）。输入模块不使用超过 16 个输入映像位，所以不需要“s”参数。

2）移动一个双 ASCII 码数据字（一个 16 位的值代表 ASCII 码字符“H”和“i”）到在专门 I/O 模块存储器中文件 0 的第 4 个数据字（字 3）中。I/O 模块在插槽 12 中。

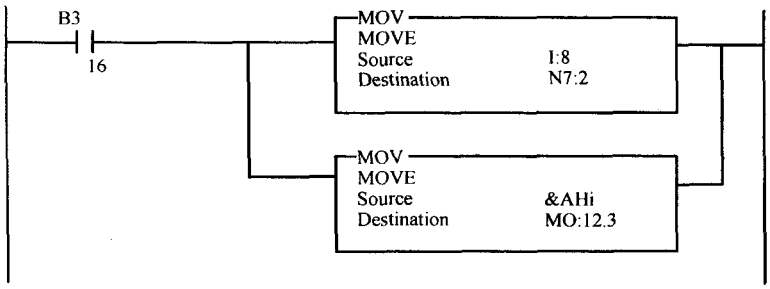


图 5-9 SLC 500 程序梯级，有条件地执行两个 MOVE 指令

S5 5.5.4 SIEMENS STEP 5 中的可寻址数据

STEP 5 操作系统中允许程序员输入常数或者地址给 I/O 映像、定时器、计数器、标志、数据块和保留的系统存储器中的数据。所有的数据可以作为字节（8 位）或者字（16 位）被输入和被寻址。

1. Siemens STEP 5 常数数据值

常数值可以在用户程序中输入。每个输入数据的前缀表明输入数据的类型，以便 PLC 把那些数据正确地转换成二进制。通过使用以下的前缀（为了可读性使用了下划线，不要在程序中使用它们），数据类型是可识别的：

KM 从0到1111_ 1111_ 1111_ 1111的二进制值(例如,KM 1001_ 1111_ 0011_ 1110将存储为1001_ 1111_ 0011_ 1110)

KH 从0到 FFFF 的16进制数(例如,KH 9F3E 将也存储为1001_ 1111_ 0011_ 1110)

KF 从－32 768到＋32 767中的有符号十进制数(例如,KF -24 771也可以保存为1001_ 1111_ 0011_ 1110)

KY 从0到255中的每对无符号十进制值(例如,KY 12 255将存储为0000_ 1100_ 1111_ 1111)

KS 一对 ASCII 字符(例如,KS 's','F'将保存为0111_ 0011_ 0100_ 0110)

KC 计数器累加值的专门代码形式[例如,KC 50在程序中被保存为计数器格式(BCD)(例如,0000_ 0000_ 0101_ 0000);在计数器存储器中以自然二进制形式(例如,0000_ 0000_ 0011_ 0010)]⊖

KT 定时器时间保持值的专门的代码形式[例如,KT 50.2在程序中以定时器格式保存(改进 BCD 码)(例如,00_ 10_ 0000_ 0101_ 0000);在定时器存储器的以自然二进制数形式(例如,0000_ 0000_ 0011_ 0010)]

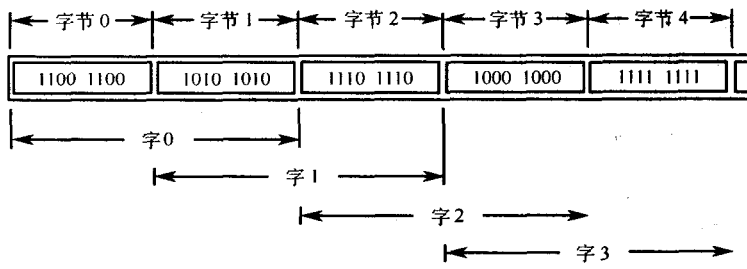


图 5-10 在 S5 PLC 中字节和字的关系

⊖ 对于 Siemens PLC，定时器/计数器存储器地址的低 10 位可以用来存储定时器/计数器的累加值。高位可能被用来作为状态位，但当用户程序读这些地址，S5 将高位置 0。

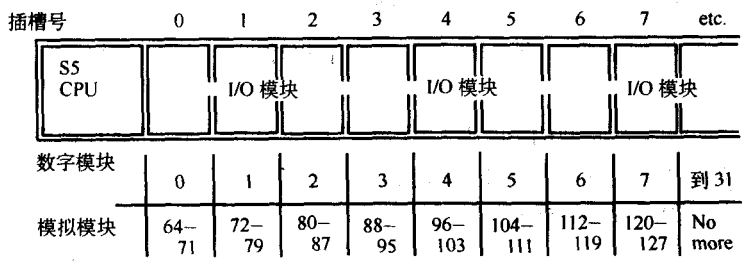
2. Siemens STEP 5 可寻址数据存储器

S5 系列 PLC 的数据存储器被安排在各自的可寻址字节中。STL 语言中“加载（L）”指令从存储器拷贝数据到 16 位的累加器中，“传送（T）”指令从 16 位累加器中拷贝数据到存储器中。（梯形图 MOVE 指令等于 STL-语言的“加载”，后面跟着一条 STL 传送指令。）

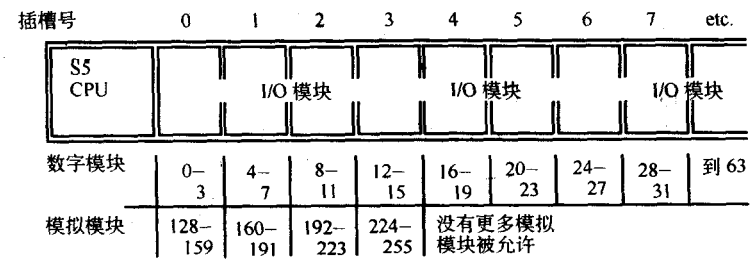
数据字可以在加载或者传送指令中寻址，但每个字由来自两个独立的字节所组成，如在图 5-10 中所示。“加载”指令可以获得来自字 2 中的数据，例如，可以读来自地址 2 的一个 8 位值和地址 3 中的另一个 8 位值。字 3 由字节 3 和字节 4 组成。通常，程序员仅使用寻址字中的偶数，这样每个字是一个 16 位的惟一的集。如果加载指令包括字节的地址，S5 PLC 拷贝来自单一地址的字节数据到累加器的低字节中，并清除累加器的高字节。当传送指令包括字节地址，它拷贝累加器低字节的内容到指定的地址，但不拷贝累加器的高字节到存储器中。

3. Siemens STEP 5 I/O 数据

输入和输出映像保存在存储器区域中，这个区域被 Siemens 叫做过程映像输入（PII）和过程映像输出（PIQ）表。数字 I/O 模块插入的每一个插槽都有 PII 存储器和 PIQ 存储器，分配来代表可能的 I/O 数据。S5 PLC 有 128 字节的可寻址 PII 和 128 字节的 PIQ 存储器。中型的 S5 PLC（例如，S5-100U）保留 1 个字节的 PII 和 1 个字节的 PIQ 存储空间给每个数字 I/O 模块，给每个模拟 I/O 模块 8 个字节的 PII 和 PIQ 存储空间。大型的 S5 PLC（例如，一个 S5-115U）保留给每个数字模块 4 个字节 PII 和 PIQ 存储空间，但对模拟模块没有 PII 或者 PIQ。这些大型的 PLC 保留 32 个直接存取地址来读取来自 I/O 模块的输入数据字节，或者写数据字节到模拟I/O模块中。当用户程序读来自地址的数据或者写数据到地址中去，PLC 实际上读或写在 I/O 模块中的存储器。（模拟 I/O 模块不需要包含全部的 32 字节的输入或者输出存储器）图 5-11 显示的字节地址对应中等（S5-100U）和大型（S5-115U）PLC 的 I/O



a) 中等大小的 S5



b) 大型的 S5

图 5-11 S5 PLC 中 I/O 地址分配：a）在中型 S5 PLC 系统中寻址；b）在大型 S5 PLC 系统中寻址

模块位置。注意模拟 I/O 模块插到插槽中，可以得到与数字 I/O 模块插到相同插槽中得到的不同字节地址。

在 STEP 5，I/O 映像数据字节以如下绝对寻址格式寻址：

IB x　过程映像输入(PII)字节，或者

QB x　过程映像输出(PIQ)字节，“x”是从0到127之间的数，对应 I/O 模块位置，或者

PY x　在大型 S5 PLC 中的可直接访问 I/O 模块的存储器，“x”是在0和255间的数(S5没有保持字节地址大于127的 CPU 模块中的 PII 或者 PIQ 存储器，所以必须使用直接访问来对在这些地址的 I/O 模块寻址。)

中型 S5 PLC I/O 模块的直接访问也可能在某些情况下使用“PY”前缀（查看第 11 章中 STEP5 中关于立即寻址的章节。）

STEP 5 也允许访问同样的 I/O 映像数据存储器作为 16 位字以下面格式寻址：

IW x　过程映像输入字，或者

QW x　过程映像输出字，“x”是在0和126之间的数，或者

PW x　直接访问大型 S5 PLC 中的 I/O 模块存储器，或者直接访问中型的 S5 PLC 中的 I/O 模块(“x”是在0和254间的偶数)

通常，“加载（L)”指令寻址存储器字获得来自指定地址的累加器的高字节数据，和来自下一个地址的累加器的低字节。例如，L IW072，获得来自 IB072 和 IB073 的数据，如图 5-12a 所示。

在小型的 S5 PLC 中，对每个数字输入模块只保留一个输入映像（PII）字节，PLC 阻止加载字指令读取来自超过 1 个输入模块的数据，并阻止传送字指令写数据到多于一个的输出模块。如图 5-12b 所示，加载和传送指令按字节寻址数据，导致 S5 PLC 只使用累加器的低字节（在加载过程中清除高字节）。图 5-12c 显示小型 S5 PLC 的带有字地址的加载指令读取来自指定的输入映像地址中一个字节的数据到累加器的高字节，并把 0 输入到低字节。同样地，传送字指令将拷贝来自累加器的高字节到指定输出映像字节，忽视可能在累加器的低字节的任何数据。图 5-12d 显示如果程序员不小心把字和字节寻址混淆，将发生什么。

图 5-12d 和 e 显示因为大型的 S5 PLC 有数字 I/O 模块，它们没有被限制在 8 位的 I/O 能力，整个字可以从这些模块的 PII/PIQ 表地址中读和写。图 5-12f 来提醒读者在大型的 S5 PLC 中模拟模块没有 PII/PIQ 存储器。

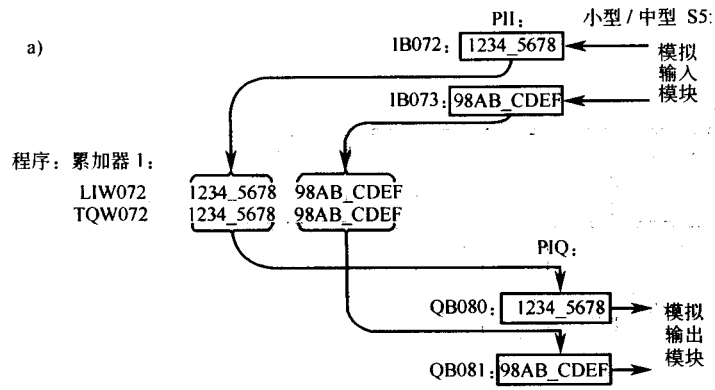


图 5-12　Siemens S5 PLC 以字和字节形式导入输入数据和写输出数据。小型或中型 PLC a) 按字寻址模拟模块 PII/PIQ 数据

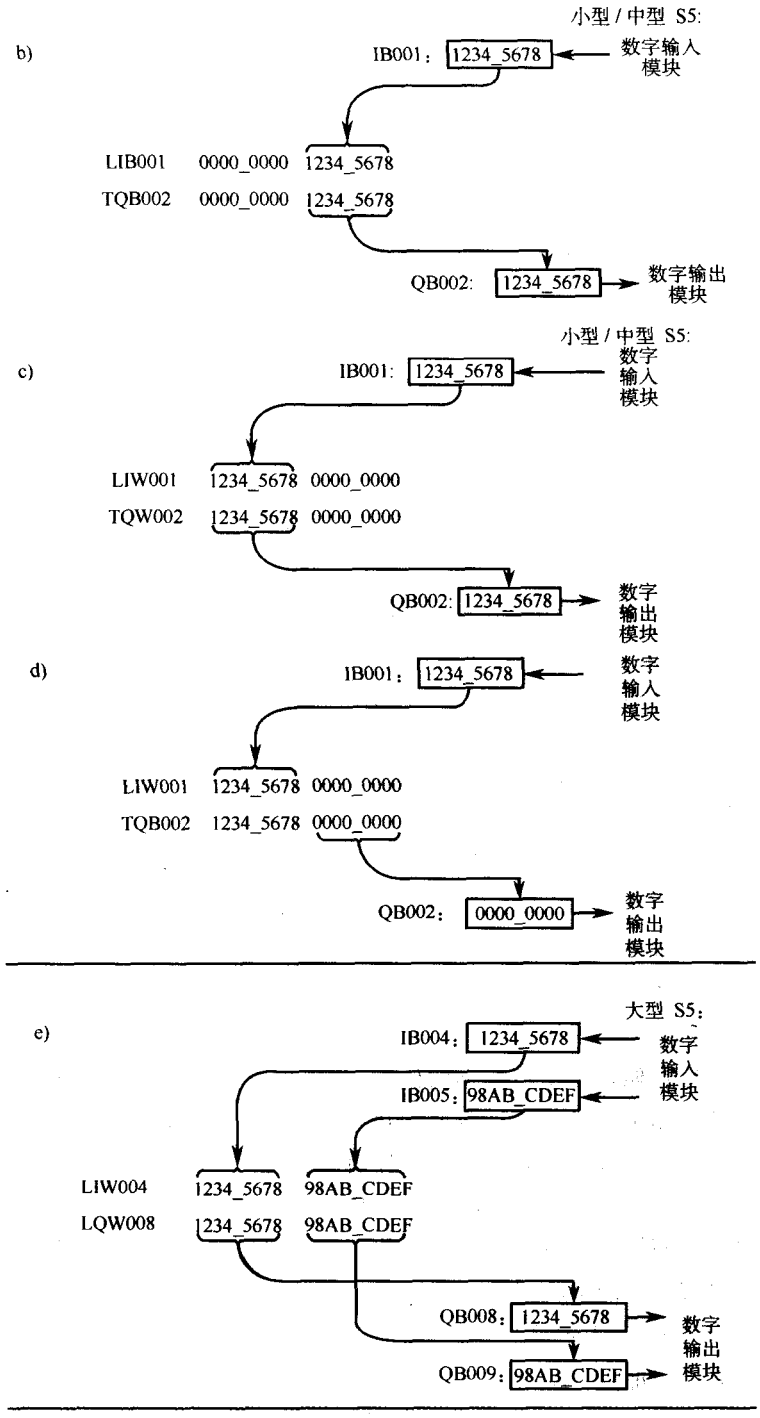


图 5-12 （续）

b）按字节寻址数字模块 PII/PIQ 数据；c）按字寻址数字模块的 PII/PIQ 数据。

d）按字和字节混合寻址数字模块 PII/PIQ 数据大型 PLC： e）按字寻址数字模块 PII/PIQ 数据

f)

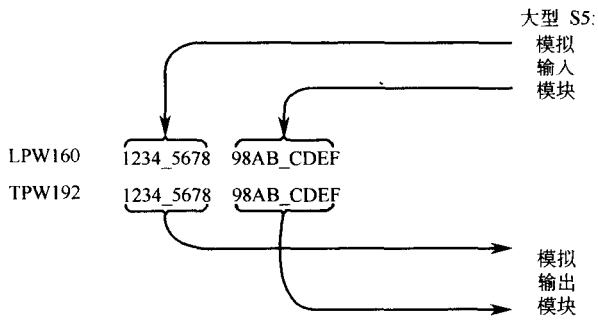


图 5-12 （续）

f) 按字寻址模拟模块 PII/PIQ 数据

每一个 I/O 数据位可以按如下格式中寻址：

Ix.b 或者 Qx.b "x"是字节序号，"b"是从0到7的位序号

直接寻址不能用来读或写单独的位，所以输入地址“Px.b”将发生错误。

4. Siemens STEP 5 系统数据

PLC 的状态和配置数据保存在 PLC 保留的系统数据存储器区域中。其中一些在用户程序中可寻址，但是大多数不可以。通过改变合适的系统数据字内容，用户程序可以改变 PLC 的配置。在第 8 章中，我们将看到系统数据字不像其他数据存储器那么容易寻址[⊖]。当可以的时候，如下面形式寻址：

RSx "x"是在范围0到255的可寻址的存储器地址(关于可寻址的保留系统存储器地址和它们的使用请查看编程手册)

通过测试位（TB）或者测试位非（TBN）指令，每个系统数据字的独立的位可寻址。通过使用无条件置位（SU）和无条件复位（RU）指令，每个系统数据字的单独位可以改变，与系统数据字寻址的情况类似，以下面的形式：

RSx.d "d"是位序号(从0到15)

5. Siemens STEP 5 计数器数据

每个计数器使用存储器单个数据字的低 10 位，来存储自然二进制数格式的累加计数器值。另外的存储器位用来存储每个计数器的状态位。

当设置计数器值时，程序必须提供一个计数器常数（KC）或者必须提供存储空间的地址，它包含计数器格式的数。(查看第 3 章关于计数器常数和计数器格式的描述。) 当 PLC 传送它们到计数器存储器的位置，计数器常数和计数器格式数自动地转化成 10 比特自然数。计数器累加值可以用以下的形式寻址：

Cx "x"是0到127之间的数(例如,C25)

如果累加值通过标准的“加载”（L）指令来读计数器地址，PLC 将提供一个 10 位无符号自然二进制数（高 6 位是 0）。如果用“加载 BCD”（LC）指令来读计数器累加器的值，PLC 将把自然二进制数转换为 12 位的 BCD 数。

计数器累加器值的位可以通过“测试位”指令（TB 和 TBN）和“无条件设置/复位”

⊖ RS 数据地址只可以在数据块程序中使用。

指令（SU 和 RU）单独寻址。这些指令仅能在 STL 功能块中编程。可寻址位以如下方式寻址：

Cx. b　　“x”是定时器数，“b”是位从0到15中的一位

低位数的计数器累加值是保持型的，而其他的累加值不是（看指定地址的手册）。保持型的计数器累加值即使在 PLC 的电源断开时仍保持。

计数器状态位也可以保持在 S5 的存储器中，来记录一些事情，如什么时候计数器到 0，及计数器的向上和向下计数的逻辑语句上一次评估是否是真，以便计数器检测何时逻辑从错误变为正确。在这些位中，只有计数器线圈状态位是可寻址的。当计数器的累加值是 0 这个位关闭。这一位的寻址方式如下：

Cx　　“x”是在0和127间的数(例如，C25)

没错，同样的地址可以给累加器值和计数器线圈使用。S5 PLC 通过指令的类型来判断程序员使用这些地址的用途。如果“C005”在布尔“检查开”指令中使用，PLC 将提供计数器线圈的布尔状态给计数器 5，因而如果“C005”在“加载”指令中使用，PLC 将提供 10 位的自然二进制累加数。

6. Siemens STEP 5 定时器数据

每个定时器使用存储器的一个数据字来保存开始运行后经过的时间量。定时器的存储器的使用和计数器相似：时间保持值保存在每个定时器数据字的低 10 位，并且定时器状态位保持在 PLC 的存储器中。当定时器被编程后，程序必须包括一个预设值作为常数（使用定时器常数格式），或者指定读哪个地址来获得预设值。如果预设值来自存储器，则在存储器中的值必须以定时器格式存储。在第 3 章讲述了定时器常数和定时器格式的数。

S5 PLC 转换定时器常数和定时器格式数到 10 位自然二进制数，存储在保留给定时器时间值存储器地址的低 10 位。表示定时器使用时间单元的大小的代码保存在相同数据字的位 12 和位 13 中。通过使用“加载（L）”或“加载 BCD”指令来读取时间保持值，在读取的过程中，它们使 PLC 转换 10 位自然二进制数到 12 位 BCD 数。这两个指令剥去读入的数的时间单元码。定时器的时间保持值可以按以下形式寻址。

Tx　　“x”是在0和127间的数

定时器保持值和定时器单元码的每一位可以通过“测试位”（TB 和 TBN）指令来检查，可以通过功能块中的“设置/复位无条件”指令来改变（SU 和 RU）。位可以用以下的格式来寻址：

Tx. b　　“x”是定时器的序号，“b”是从0到15中的一位

定时器状态位也可以保持在 S5 的存储器中，因此定时器可以启动并运行到完成，并可以表明什么时候定时器结束。这些位，只有定时器线圈状态位是可以寻址的。当定时器启动后，这个位可以打开和关闭，当定时器停止运行或者当时间值到 0 时候，这个位将改变状态，取决于定时器的类型。这个位的寻址如下：

Tx　　“x”是在0和127中的数(例如 C25)

同样的地址可以给定时器的时间保留值和定时器线圈使用。S5 PLC 通过指令的类型来判断程序员使用这些地址的用途。如果“T005”在布尔“检查开”指令中使用，PLC 将给定时器 5 提供定时器线圈的布尔状态，而如果“T005”在“加载”指令中使用，PLC 将提

供给 10 位自然二进制数值。

7. Siemens STEP 5 标志数据

标志数据字节在 S5 PLC 中可用来存储工作数据。低地址的标志字节是可保持的，然而，每次 PLC 进入运行模式时候，其他字节会被清除。（查看用户手册来决定 S5 PLC 有多少标志存储器，和有多少个是可保持的。）标志字节以这个格式寻址：

```
FY x      "x"是从0到255间的数
```

标志存储器也可以以如下格式按字寻址：

```
FW x
```

像 I/O 存储器，一个标志字由同地址标志字节和下一个标志字节组成，如图 5-13 所示。单个的标志位可以如下形式寻址：

```
F x.b     "x"是字节序号,"b"是从0到7的位序号
```

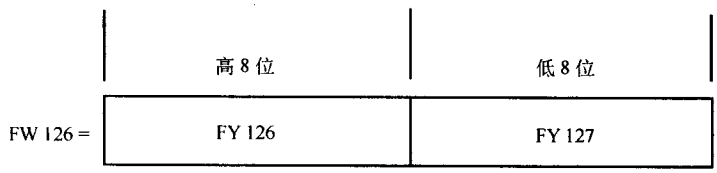


图 5-13 一个 S5 标志字包括两个标志字节

8. Siemens STEP 5 数据块

S5 PLC 可以在数据块中保存用户数据。直到包含数据字的数据块被创建，数据块地址不可以在程序中使用。数据块可以通过编程器创建，或者用户程序可以包含指令来创建数据块。每个在数据块中的数据字也必须在数据块的创建过程中被创建。数据块可以包含从 0 到 255 的数据字。

在梯形图程序中，数据块中的数据字可以按如下方式寻址：

```
DBx：DWn  "x"是已存在的数据块的序号,从0到255(STEP 5操作系统使用数据块0和1),"n"是在数据块中存在的数
          据字的数量,从0到被创建的(最大数255)实际的最后字。
```

每个数据字的高和低字节可以分别以如下形式寻址：

```
DBx：DRn    寻址低(右边)字节
DBx：DLn    寻址高(左边)字节
```

STL 程序中，数据字和数据字节地址不包括“DBx：”。但是，在程序试图使用数据字或者数据字节前，程序必须包括一个指令来“调用”数据块。以下的 STL 指令调用数据块：

```
C DBx
```

在调用数据块后，STL 程序可以寻址在下面形式的数据块中的数据字或者字节：

```
DWn    or    DRn    or    DLn
```

单独的数据字的位可以通过使用 TEST BIT 指令（TB 和 TBN）来检查，并可以通过使用功能块中的无条件置位/复位指令（SU 和 RU）来改变。数据字的位是按以下形式寻址：

```
Dx.b      "x"是数据字数(在当前打开的数据块),"b"是从0到15的一个位
```

一次只能打开一个数据块（通过使用 CALL 指令）。调用数据块会自动关闭先前调用的数据块。结构化的程序可以包括跳转到其他程序，其他程序可能包括 CALL 指令来调用它们自身的数据块，但当其他程序结束时，S5 PLC 会在跳转到其他的程序之前，自动地重新调用已打开的数据块。

当创建数据块时，程序员必须提供一个初始化的数据值，并且必须表明在数据块中每个数据字的默认数据形式。默认数据形式可以是常数数据的任何允许格式，如在这章早前描述的。编程器显示数据块中的默认数据格式的每一个字，尽管用户程序可以操作任何格式的数据字、字节或位。

当创建数据块数据字时在类型识别符和初始化数据值之间使用一个“=”（例如，KH=1CF3）。

图 5-14 显示 STEP 5 梯形图程序和等价的状态表（STL）程序，无条件地：

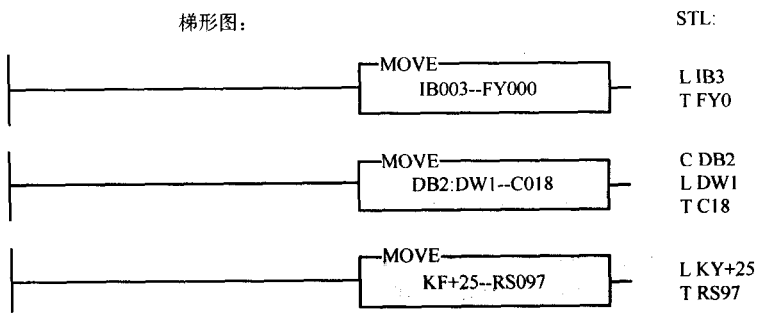


图 5-14 显示数据字使用的 S5 PLC 程序（梯形图和 STL）

1）从过程映像输入表中移动一个字节的数字输入映像数据（IB3）到第一个标志数据字节（FY0）。

梯形图 MOVE 指令与 STL LOAD（L）和 TRANSFER（T）指令功能相同。

2）从数据块 2 中移动第二个数据字（DW1）到计数器 18（C18）的累加值存储空间。

3）移动有符号二进制数的常数值（KF25），到一个系统数据字中（RS 97）。在用户程序中可寻址的系统数据字仅可以在一种类型的用户程序中寻址：功能块。

9. Siemens STEP 5 符号寻址

STEP 5 允许程序员进入一个符号库中。符号地址在编程过程中不能用来输入地址。它们仅能在编程器屏幕中程序员必须输入的绝对地址和块地址的旁边被显示。

符号名字不是 S5 PLC 中程序的一部分。它们仅能在 STEP 5 编程器和通过编程器保存到硬盘文件中的 STEP 5 程序中保持。如果用另一个编程器来读来自 PLC 的程序，那么使用原来编程器定义的字母名字将不再显示。

10. Siemens STEP 5 变址寻址

STEP 5 允许程序员使用两种变址寻址：

1）数据块寻址（Data block addressing），如上所述，实际上是变址寻址的一种形式。当数据块被调用的时候，保留给数据块的存储器区域的起始地址变成基地址，数据字地址是来自数据块起始位置的偏移地址。如果程序用来在一系列特殊条件下调用数据块，或者在不同条件下调用其他数据块，程序指令中的数据字地址（偏移地址）可以指示每个数据块（基地址）的数据。

2）处理操作（PROCESSING OPERATION，DO）指令（仅用在 STL 的功能块中），在变址寻址中使用。在下面的 STL 语言例子中显示，DO 指令必须包括标志字（FW）或者数据字（DW）地址，它们必须包含地址的偏移部分。DO 指令后面必须跟着指明通过变址寻址将执行什么操作的指令以及地址识别符指明地址的基地址部分。基地址识别符必须表明 Siemens S5 存储区域，必须表明需要的数据单元的大小（例如，I 表明是输入映像位，IW 表明是输入映像字，C 表明是计数器字或者计数器状态位，由使用的指令所确定）。

STEP 5 需要包含地址识别符的数字地址，所以必须在 DO 后面跟随的指令中输入带有地址识别符的地址 0 或 0.0。16 位偏移数存储在标志或者数据字中，它们在 DO 指令中被识别，其中低字节的数指明一个字节或者字地址序号，（如果需要）在高字节中的数指明位序号。如果位地址不需要，高位字节应该是 0。例如：

```
L    KH0203        ; 把字节地址"03"放入 FW4的低字节
T    FW 4          ; 并把位地址"02"放入 FW4的高字节
DO   FW 4          ; 使用 FW4中的数字作为偏移量
S    Q0.0          ; 在输出映像存储器中设置从字节3和位2开始的偏移量的位
                   ; (=Q3.2)
```

和

```
L + 21             ; 把数字"21"放入 DB5：DW6
C DB5              ; (小于256的数仅使用低字节)
T DW6
DO DW6             ; 使用在 DB5：DW6中的数作为偏移量
L TO               ; 加载来自定时器存储区域的数
                   ; 偏移量来自定时器区域开始的21(= T21)
```

11. Siemens STEP 5 间接寻址

两种形式的间接寻址在 S5 PLC 是可用的。一个方法是使用间接地加载寄存器（LIR）或者间接地传送寄存器（TIR）指令和一个累加器来表明（指向）要使用的地址。另一种方法使用间接过程（DI）指令。

使用寄存器间接指令，代表在 PLC 存储器地址的一个 16 位数必须首先加载到累加器 1 中。间接地加载寄存器指令可以用来加载来自被指示地址的数到累加器中，或者间接地传送寄存器指令用来传送来自累加器的一个数到指定地址中。这些指令只可以在 STL 的功能块中编程。指令也必须表明哪一个累加器应该加载或者传送，形式如下：

```
LIR 0    把来自存储器的值加载到累加器1中(在 LIR 指令执行前重写在累加器1中的间接地址数)
LIR 2    把来自存储器的值加载到累加器2中(在累加器1中的间接地址没有被覆盖)
TIR 2    传送累加器2中的内容到由累加器1中的地址指定的地址中
```

为了使用这些指令，程序员必须知道实际的 16 位存储器地址，这个地址可以被读或者写，而不是通常的在数字前加前缀形式的地址。S5 PLC 手册识别实际地址范围，它们用来给 I/O 映像、标志、计数器、系统数据等等。

间接处理指令也是一种形式的间接寻址，仅在 STL 的功能块中可使用。（读者可以忽略这一节，直到读过第 8 章后）。间接处理指令和处理操作（DO）指令相似，因为它将影响在 DI 指令后执行的指令将用哪一个地址。DI 指令不同于 DO 指令，因为当 DI 指令执行的时候，它使用累加器 1 中的指针来选择来自参数列表的地址，当块被调用时候，参数传递给这个功能块。

下面的例子描述了这个过程。在这个例子中，当被调用的时候，调用程序传递 3 个字

(IW4，FW6 和 FW40) 的地址给 FB4。FB4 包括一个声明段，这个段识别这 3 个地址作为输入参数 (IWs) 给功能块。在简化的例程序中，在 DI 指令执行前，FB4 把数 "3" 输入到累加器 1 中。在这个例子中，如果前面没有 DI 指令，DI 指令后的 "传送" 指令通常拷贝来自累加器 1 的内容 (数 3) 到 QW6。但是因为 DI 指令在 "传送" 指令之前，PLC 知道使用在累加器 1 中数 (3) 来表明哪个输入参数用来作为传送给 QW6 的源数据。因为第三个输入参数是 FW40，FW40 的内容被加载，然后传送给 QW6。在我们的例子中，调用的程序是：

```
JU FB3
p1 : IW4
p2 : FW6
p3 : FW40
```

和功能块 (FB4) 是：

```
DECL : p1 IW
DECL : p2 IW
DECL : : p3 IW
L KF +3
DI
T QW6
BE
```

12. Siemens STEP 5 形式操作数

用户定义的绝对存储器或者块名字被定义为形式操作数，但是仅在功能块中。一旦定义过，形式操作数可以在向功能块中输入程序时代替绝对地址。在第 8 章显示如何在给功能块编程时声明形式操作数的名字，以及在功能块中如何使用形式操作数的名字。

形式操作数在预编程的功能块中已经定义，这些功能块在 S5 PLC 中。形式操作数的名字不是 S5 PLC 程序的一部分。它们仅在 STEP 5 编程器和利用编程器存储在磁盘文件中的 STEP 5 程序中保持。如果其他的编程器用来读来自 PLC 的程序，通过使用原编程器定义的字母与数字混编的名字将不出现 (除非它们也在其他编程器中定义过)。

5.5.5 Siemens STEP 7 中的可寻址数据

S7

当控制程序事实上包含几个用户程序，S7 的存储器允许高级结构化编程。一些存储空间是所有用户程序共享的，但其他区域保留给单一程序。STEP 7 提供各种各样的数据类型，并且鼓励使用符号和变量名编程。

1. Siemens STEP 7 常数数据值

通过使用 IEC 1131-3 指定的数据类型和常数格式，以及不包括在 IEC 1131-3 标准内的一些类型，数据值可以作为 STEP 7 程序的常数输入。IEC 1131-3 定义的数据元素类型包括：

- 布尔型：布尔值，输入如下：

  二进制形式：0或1

  文本形式：true 或 false

- 字节型：8 位二进制数，代表从 0 到 255 的值，作为 16 进制常数 (基 16) 输入如下：

  B#16#0至 B#16#FF

- 字型：16 位无符号二进制数，代表从 0 到 65 535 的值，或者 16 位 BCD 码数代表从

－999 到＋999 的值[⊖]。可以输入常数：

二进制：2#0至2#1111_ 1111_ 1111_ 1111
十六进制：W#16#0至 W#16#FFFF
BCD：C#-999至 C#999
无符号十进制：B#(0,0)至 B#(255,255)

- 双字型：32 位的无符号二进制数，代表从 0 到（大约）40 亿的值，输入：

二进制：2#0至2#1111_ 1111_ 1111_ 1111_ 1111_ 1111_ 1111_ 1111
十六进制：D#16#0至 D#16#FFFF_ FFFF
无符号十进制：B#(0,0,0,0)至 B#(255,255,255,255)

- 整型：16 位有符号二进制数，代表从－32 768 到＋32 767 间的整数。可以输入常数：

整数值：- 32 768至+ 32 767

- 长整型数：32 位的有符号二进制数，代表从－2 147 483 648 到＋2 147 483 647 间的整型数，输入：

“长”整数：L#-2 147 483 648至 L#2 147 483 647

- 实数型：32 位 ANSI/IEEE 格式的二进制数，可以代表从 $\pm 3.402823 \times 10^{38}$ 到 $\pm 1.175494 \times 10^{-38}$ 间的浮点数。常数可以使用小数点或者科学计数法符号（例如，－1.553 或者 4.7e－25）
- 字符型：代表 ASCII 字符的 8 位 ASCII 码。ASCII 字符必须加单引号输入（例如，‘R’）
- 时间型：32 位有符号整型值代表午夜后的微秒值，输入天/小时/分钟/秒/微秒的代码形式：

T#- 24D_20H_31M_23S_648MS 至 T#24D_20H_31M_23S_648MS

- 日期型：16 位二进制数代表从 1990 年 1 月 1 日至今的天数。
常数必须是正数，并以年/月/天的格式输入：

D#1990-1-1至 D#2168-12-31

在 STEP 7 中允许包括非 IEC 1131-3 的单元素数据类型：

- TIME_OF_DAY：32 位的小时/分钟/秒/微秒时间系统，以与 IEC 1131-3 时间值同样的二进制格式保存，但是常数可以更简单地输入，以如下形式：

TOD#0：0：0.0至 TOD#24：59：59.999

- 计数器型：不用声明的数据类型；然而，有一种计数器型常数，它由一个 16 位数组成，其低 12 位是 3 个 BCD 码数字，其高 4 位无效。在程序执行过程中，计数器值加载到 S7 PLC 的 32 位累加器的低字中，然后用来设置计数器。尽管 S7 PLC 可以转换 12 位 BCD 码数为 10 位自然二进制数，在它被写入计数器存储器区域的时候，它必须以 BCD 码形式加载到累加器中。它可以作为常数输入：

C#000至 C#999

- S5TIME：16 位数，其低 12 位中保存 3 个 BCD 码，位 12 和位 13 识别时间单元大小，2 个高位是无用位。数值时间小时/分钟/秒/微秒以如下格式输入：

S5T#10ms 至 S5T#2h_ 46m_ 30s(例如，S5T#2m_ 10s)

⊖ BCD 数的高半码元组全为 1，则这个数是负数。

或者可以在时间单元代码值后输入 3 个 16 进制数值。可用的时间单元代码是 0，1，2，3 表示：0.01-，0.1-，1-，或者 10-s 的时间单元。以 16 进制表示的 S5TIME 值显示如下：

```
#16#2130（表明130秒，或者2分+ 10秒）
```

- 块、计数器和定时器：程序员创建这些元素时，块、计数器或者定时器数以 16 位二进制数形式存储。当在程序中需要时，块、计数器和定时器以名字和数字输入（例如，把计数器数传递给另一个程序）。

STEP 7 也允许另外的非 IEC 1131-3 复合数据常数。复合数据包含其他数据元素的数组或者结构体。S7 复合数据类型包括：

- DATA _ AND _ TIME：8 位 BCD 码数据值的集合代表一个年/月/日/小时/分钟/秒/微秒/星期日特性，保存在如图 5-15 中的存储器中。DATE _ AND _ TIME 常数以如下格式输入：

```
DT#1999-12-31-24-59-59-999-7
```

- 指针：识别其他数据元素存储器中的位置的数据元素。STEP 7 提供两个不同方法使用指针。两种方法都允许一些可选的指针形式：

  一种类型的指针数据在以下的间接寻址格式中使用。（STEP 7 间接寻址在这章中其他地方描述。）

1）在间接寻址使用前，间接寻址的存储器间接型需要程序设置一个指针指示存储空间。存储器间接指针可以是：

- 16 位的无符号二进制数值（代表在 0 和 65 535 间的十进制数值）指向定时器、计数器、数据块、函数或者功能块。
- 32 位双字，其中低三位表示一个数（从 0 到 7），接下来的低 16 位表示存储器地址（从 0 到 65 535），高 13 位是 0。这个类型的指针用来指向除了定时器或者计数器外的存储器地址。

2）寄存器间接型寻址使用 32 位指针，在间接寻址使用前，程序必须加载到地址寄存器中。当输入常数的时候，指针必须是以下的一种形式：

```
p#x.d     区域内部指针；“x”是从0到65 535间的数，“d”必须是从0到7间的数
p#ax.d    跨区域指针；“a”是前缀（P、I、Q、M、DBX或者 DIX），表明指向存储器中的哪个地址，“x”是从0到65 535间的一个数，“d”必须是从0到7间的数
```

当在存储器中存储的时候，区域内部指针是与上面提到的 32 位存储器间接指针形式相同的一个 32 位数。跨区域指针数据值以相同的 32 位数据格式保存在存储器中，除了最高位是 1 和位 24、25 和 26 包含一个 3 位代码，它们识别指向的存储器区域（如图 5-15 所示）。

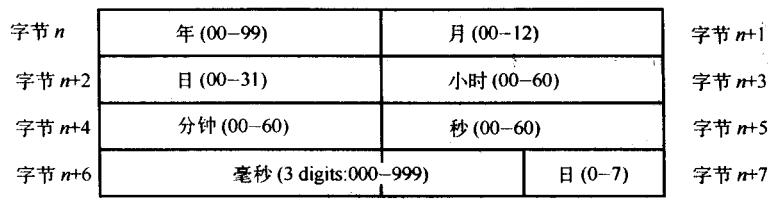


图 5-15　S7 DATE _ AND _ TIME 元素结构体

第二个类型的指针用来识别一组存储空间的起始位置，它们包含复合数据元素，如 DATE _ AND _ TIME 元素，或者数组，或者结构体。这些类型的指针在程序间传递复合数据元素时使用。不用拷贝程序存储器空间的整个复合数据元素到其他程序空间，取而代之的是只有指针被传递。然后两个程序使用相同的复合数据元素。为了这个目的，指针常数以如下格式输入：

```
p#area_ identifier byte_ number ? bit_ number
```

例如，位数组：

```
p#M25.4
p#DB30.DBX25.4
```

字节/字/双字：

```
p#MB25.0
p#DB30.DB25.0
```

即使在指针指向一个复合数据项，包括字节、字或者双字，输入位数也是必需的。位数应该是 0，除非指针用来指向这个位。S7 PLC 以 6 字节格式存储指针，如图 5-16 所示：

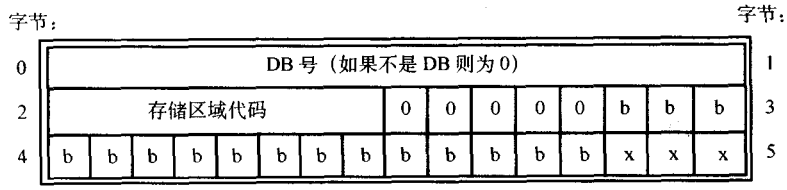


图 5-16 S7 指针指向复合数据元素

“0”表示位没有使用

“b”表示字节地址

“x”表示位地址

存储器区域代码（以 16 进制显示）包括：

81 对应 I （输入映像存储器）

82 对应 O （输出映像存储器）

83 对应 M （共享数据存储器）

84 对应 DB （共享数据块存储器）

85 对应 DI （立即数据块存储器）

85 对应 L （局部数据存储器）

86 对应先前使用的局部数据存储器

- 字符串：一组字符元素跟在两个头字节后面，顺序的保存在存储器字节中。在字符串头的第一个字节表明保留给字符串的存储器字节数（包括头），第二个字节指示当前包含有效数据的存储字节数。在字符串头的后面，每个存储字节包含一个 ASCII 码。字符串默认最大有 256 个字节，包括字符串头，但是短的字符串也可以被创建。在程序中用单引号标明字符串常数，如下：

  ```
  'This 50-character string will need 52 memory bytes'
  ```

- 数组：一维到六维由类似数据元素组成的集合。数据元素按顺序保存在存储器中。在程序中，数组中的数据不可以作为常数数据值输入。程序可以包含指针常数来识

别保留给数组的存储地址的起始地址，包含初始值的数组可以在变量声明表和/或数据声明表中声明，它们将在本章后面讲述。当一个数组作为命名的变量被创建，S7 PLC 保存在存储器中的 6 字节指针来识别保留给命名的数组使用的存储器的起始地址。

- 结构体：包括任何类型的数据元素（甚至其他结构体元素）的结构集合。结构体如同数组，在程序中数据不可以作为常数数据值输入，但是程序可以包含指针常数，来识别保留给结构体的存储器的起始地址。结构体可以被创建，也可以在变量声明表/数据声明表中设定初始值，它们将在这章中后面讨论。像数组数据、结构体数据可以在顺序的存储空间保存，除非，如果结构体包括其他复合数据元素（例如，数组或者其他结构体），结构体实际上仅包含一个指针来指向其他数据元素。结构体可以包含的其他结构体嵌套最多不超过 8 层。当结构体作为命名的变量而创建时，S7 PLC 在存储器中保存六字节的指针来识别在结构体中一组存储位置的起始地址。
- 任意（ANY）型：一种复合数据元素，有点像数组，数组可以包括相同类型的多个数据元素，但任意型数据可以任何类型的元素，甚至结构体元素或者其他任何元素。任意型数据元素可以按顺序存储位置保存，包含基本数据元素或者复合数据元素的指针。当创建任意型数据元素作为命名的变量的时候，类似一个指针的 10 字节数据集将保存在存储器中。这个数据集识别任意型数据元素中的数据的起始地址，并包括它包含的元素的类型的描述和元素的数量。

程序包含指针常数来识别任意型数据元素。它稍微不同于前面描述的指针常数，指针格式如下：

```
p#area_ identifier_ and_ byte.bit data_ type how_ many
```

例如：

```
p#M25.0 byte 5      开始于 M25的5字节数组(代表基本数据类型必须以绝对地址的符号输入)
p#M30.0 struct 3    开始于 M30的包括3个结构体数据元素的数组(如果程序员愿意,复合数据起始地址可以使用
                    它们的符号名字输入)
```

(位通常是 0，除非 data _ type 是布尔型。)

10 字节任意型指针以类似于 6 字节标准指针的格式保存，其实际格式根据任意型数组中是什么类型的元素决定。(查看你的手册)。当功能程序使用任意型参数，PLC 需要 10 字节的任意型指针来决定哪个存储空间应该实际被使用。

2. Siemens STEP 7 可寻址数据存储器

S7 PLC 的可寻址存储器包含：

- I/O 映像数据
- 位存储器
- 定时器数据
- 计数器数据
- 系统数据
- 数据块
- 外部 I/O 存储器

外部 I/O 存储器实际位于 I/O 模块中，但是如果它在 CPU 模块中，则可以在程序中使用。

■ 局部数据堆栈（L 堆栈）存储器

每个 S7 PLC 的程序可以有它自己的局部数据存储器，每个程序的局部数据存储器被保护起来以防止其他程序改变它。

所用存储区域都可以使用绝对寻址。绝对地址有一个前缀来识别存储器区域和数据单元大小，然后一个惟一的数来识别特殊的存储器。STEP 7 也可以允许间接寻址、变址寻址和使用指针来识别存储器。

按照 IEC 1131-3 要求，每个 PLC 程序应该有一个其他程序不可以访问的存储区域，STEP 7 提供新的在 STEP 5 中不可用的立即数据块和上面提到的局部数据堆栈。STEP 7 可以给程序员比 IEC 1131-3 标准规定的更多的权力，指针和/或任意型数据可以用来允许程序访问其他程序的局部堆栈存储器。

STEP 7 鼓励使用符号寻址。符号表和数据声明表可以定义符号名字来代表绝对地址。每个 STEP 7 程序包括在它们头文件中的变量声明表，可以用来声明（局部）符号名，在程序中代替绝对地址。使用符号名可以使程序更容易地拷贝到其他 PLC 中，即使其他 PLC 有不同的存储器安排，因为程序员可以容易地修改符号表，数据声明表和变量声明表，而无须用不同的地址重新写整个程序。IEC 1131-3 标准的要求程序可以容易的从一个 PLC 移植到另一个 PLC。

3. Siemens STEP 7 输入映像数据

输入映像数据保存在过程映像输入表中（PII），它包含来自数字输入模块的 128 个字节的输入数据。PII 数据可以用以下绝对地址形式寻址：

IB x　8位字节寻址，“x”是从0到127间的数，反映数字输入模块的位置。地址可以自动地分配给 I/O 模块，作为 PLC 配置使 PLC 识别 I/O 模块。低字节代表靠近 CPU 模块的模块。Siemens 保留另外的 IB 地址，最多可达65 535，作为将来扩展使用。

I x. d　“d”是在字节“x”中的位序号，从0到7。

IW x　16位数据字寻址，“x”是从0到126的数，表示头两个连续字节组成的16位字的地址。2个字节连接起来成为一个字，如图5-17a 所示。

ID x　32位双字，“x”是从0到124的数，表示头4个连续的字节组成的32位双字。4个字节连接起来组成一个双字，如图5-17b 所示。

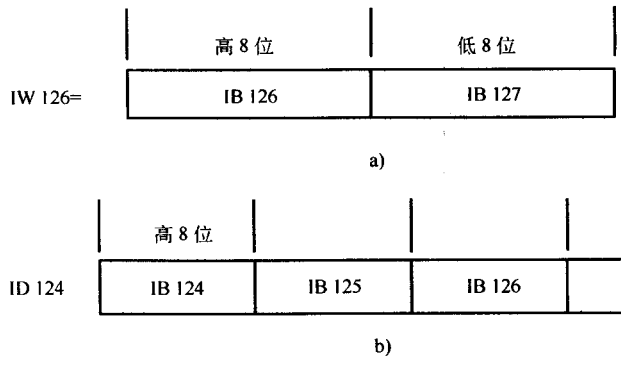


图 5-17　S7 字和双字结构

4. Siemens STEP 7 输出映像数据

输出映像数据存储在过程映像输出表中（PIQ），它包含数字输出模块的 128 字节的输出数据。PIQ 数据可以像 PII 数据一样寻址，以如下的绝对地址形式：

QB x　8位字节，x是从0到127的数(Siemens保留了附加的QB地址，直到65 535，作为将来扩展使用。)
Q x. d　d是字节中的位序号，从0到7
QW x　16位字寻址，x是从0到126的数
QD x　32位双字，x从0到124

5. Siemens STEP 7 外围输入数据

外围输入数据和PII数据使用方式相同，但是实际上在一个输入模块的存储器中。外围输入数据反映当前输入状态，而不是扫描循环读输入模块数据到输入映像存储器时的输入状态。单独的外围输入数据位不能寻址，但在下面绝对地址形式寻址是允许的：

PIB x　8位字节，“x”是从0到65 535间的数。注意字节地址从0到127的输入模块可以直接读取，尽管它们带有在每个扫描循环刷新的输入映像数据存储器。带有外围I/O地址的指令比带有PII或者PIQ地址的指令需要更长的执行时间
PIW x　16位数据字寻址，“x”是从0到65 534的数
PID x　32位双字，“x”是从0到65 532的数

6. Siemens STEP 7 外围输出数据

外围输出数据和外围输入数据的使用方式相同。单独的外围输出数据位不可以寻址，但以下面的绝对地址形式寻址是允许的：

PQB x　8位字节，“x”是从0到65 535间的数
PQW x　16位数据字寻址，“x”是从0到65 534间的数
PQD x　32位双字，“x”从0到65 532间的数

7. Siemens STEP 7 位存储器

位存储器（有些时候叫做标记存储器）用来存储用户数据的位、字节、字或者双字。位存储器默认有16个字节（从MB0到MB15）是保持型的，但程序员可以增加或者减少保持型位存储器地址的数目。位存储器可以以如下绝对地址形式寻址：

MB x　8位字节，“x”是从0到255之间的数
M x. d　“d”是在字节中位的序数，从0到7
MW x　16位数据字寻址，它由两个连续的字节组成，“x”是从0到254之间的数
MD x　32位双字，它由4个连续的字节组成，“x”是从0到252之间的数

8. Siemens STEP 7 定时器数据

像S5定时器数据字一样，定时器数据字在低10位中包含10位自然二进制数，代表定时器所剩时间。Siemens没有告诉高位用来做什么，但它们可能用作给定时器状态位。不像S5 PLC，定时器数据值不是保持型的，除非程序员专门地配置一些定时器字作为保持型。二进制的剩余时间可以通过梯形图指令或者STL加载（L）指令读入。STL加载BCD（LC）指令也可以读定时器时间剩余值并转换为BCD码。在读剩余时间值时，PLC仅读位0到位9的值，送到累加器中。定时器剩余值可以通过如下绝对寻址方式寻址：

T x　“x”是从0到255的数

定时器的状态位也可以通过绝对寻址模式寻址，如下：

T x　“x”是从0到255的数

状态位的绝对地址与时间剩余值相同，但PLC可以通过包含地址的指令的类型来决定使用哪一个值。

S7 PLC也可以提供一个实时的定时器，它可以保留日期和时间，就像用户对现代计算机的期望一样。在配置过程中（查看第10章），程序员指定一个地址来保存实时数据。实时

时间可以在程序中被检测来使某些功能以程序指定的时间或者间隔时间执行。

9. Siemens STEP 7 计数器数据

与 S5 PLC 一样，S7 中的计数器数据字在低 10 位包括 10 位自然二进制数。这个数代表计数器的累加值，可以通过梯形图指令读取或者在 STL 中使用加载（L）指令读取。在累加值读到累加器的过程中，STL 加载 BCD（LC）指令把累加值转化为 BCD 码。默认只有八个计数器数据字（从 C0 到 C7）是保持型的，但可以配置更多的保持定时器。计数器累加值数据字可以用如下绝对地址的形式寻址：

```
C x        "x"是从0到255的计数器数
```

计数器的状态位也可以通过以下的绝对寻址模式寻址：

```
C x        "x"是从0到255的值
```

状态位的绝对地址和累加值的地址相同，但 PLC 通过包含地址的指令的类型决定使用哪一个值。

10. Siemens STEP 7 系统数据

S7 PLC 可以使系统数据按以下的方式给程序员使用：

1）每次 S7 操作系统调用某一个用户编写的组织块程序时，自动地把系统数据值作为参数传递给它。程序员必须知道 S7 PLC 传递什么系统值，被参数传递的局部堆栈的绝对地址或者符号名。例如，带有符号名 Event _ ID 的值可以传递给每个 S7 操作系统调用的组织块（用户程序不可以调用组织块，只有操作系统可以）。Event _ ID 识别组织块被调用的原因，在响应前，组织块中的程序可以检验 Event _ ID 中的值。用户手册中包含 Event _ ID 数的更多信息。

2）通过允许用户写程序包含调用内嵌系统功能程序，SFC 51，也可以以 RDSYSST 为名称运行。RDSYSST 可以用来拷贝来自系统数据区域存储器的任意的系统状态列表变量到程序员指定的存储空间。图 15-6 显示了“部分”可以读取的系统状态列表，以及在调用 RDSYSST 来读那个数据时必须包括的系统状态列表数（SZL-ID）。更多的信息包含在 STEP 7 手册中。

3）通过一些 MPU 寄存器的绝对寻址。可寻址的 MPU 寄存器包括：

(a) 16 位状态字，如图 5-18 所示，以如下方式寻址：

```
STW (e.g., L STW reads the current status word)
```

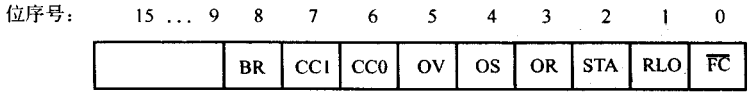


图 5-18 S7 状态字

一些状态字的单独状态位可以独立地寻址，因此它们可以通过布尔逻辑语句检测。其他状态字的位不可以分开寻址，但一些指令可以自动地使用。

- CC0，CC1，BR，OV 和 OS 是条件代码位，它们受 MPU 最近操作或生成的数据值影响。（查看用户手册可以知道不同指令类型如何影响这些位。）
- FC，RLO，STA，OR 和 BR（BR 是有时作为 ENO 寻址）。在 PLC 执行指令时，被 PLC 执行的逻辑操作的中间或最终结果影响这些位。如果 RLO 位是 1，将执行条件输出语句。如果一个内置的字操作指令、内置函数或者功能块程序正确地执行，BR/ENO

位被置 1。BR/ENO 位也可以通过用户编写的函数或者功能块写入。在逻辑条件执行的过程中，其他位可以被 S7 PLC 使用以产生最后的 RLO 结果。(查看用户手册)

(b) MPU 的地址寄存器，如果指令寻址为 AR1 或者 AR2，地址寄存器中的数据可以在寄存器间接寻址中使用（在这节后面讲述）。

11. Siemens STEP 7 共享数据块

S7 PLC 提供两个类型的数据块：共享数据块，在任何 S7 PLC 程序中可寻址；立即数据块，保留给立即数据块涉及的功能块程序使用。在这一节中，我们主要讨论共享数据块，立即数据块在后面讨论。

在数据块的创建过程中，程序员创建一个用户数据类型（UDT）元素，它们看起来像数据块，并且以与数据块相同的方式创建，但是它实际只是一个复合数据元素的模板。UDT 不是数据元素，也不可以作为数据元素使用。UDT 只存在于编程器中，在共享或者立即数据块或者在结构体数据元素的创建过程中使用。

数据块是数据元素的集合。数据块在程序中使用之前，每个数据块必须被程序员或者用户程序的指令创建。一旦被创建，在读写数据块内容之前，用户程序必须先打开这个已经创建的数据块。打开已存在的数据块的梯形图指令和 STL 指令，如图 5-19 所示。一次只能打开一个共享数据块，因此打开一个共享数据块会自动地关闭先前打开的共享数据块。默认只有一个数据块（DB1）是保持型的，但是程序员可以增加或者减少这个数字。

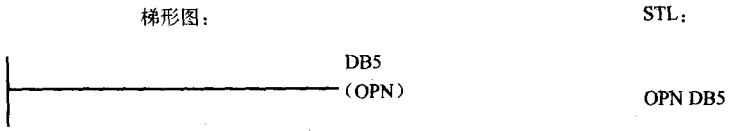


图 5-19 打开一个数据块的 STEP 7 指令

12. 共享数据块的数据声明和数据类型

共享数据块可以通过 STEP 7 程序软件创建，声明一个数据块作为数据元素结构体，如表 5-1 所示。表 5-1 的顶部显示编程软件正在为 PLC 创建数据块，这个 PLC 在编程软件中被认作“Stn _ 3”，编程软件把这个 PLC 所在的网络叫做“Plant _ 2”，这个编程器把这些数据保存在硬盘 C:\ 上。被创建的数据块是 DB4。DB4 可以在编程软件的其他地方分配一个符号名，但符号名不出现在这个编程屏幕上。

表 5-1 使用编程器创建数据块时显示的 STEP 7 数据声明表

数据块 c:\ Plant _ 2\ Stn _ 3\ DB4				
地 址	名 字	类 型	初始值	注 释
0.0		STRUCT		
+0.0	latch _ 1	BOOL	0	
+0.1	latch _ 2	BOOL	0	
+2.0	count-up	WORD	20	
+4.0	big _ num	DINT	0	
+8.0	ratio	REAL	9.5	
+12.0	message	STRING [20]		
=32.0		END _ STRUCT		

数据声明表包括关键字 STRUCT 和 END _ STRUCT 来定义构成数据块的一套数据元素的起始和结束。编程软件插入这些关键字。在这些关键字中，在数据声明表中每个记录（行）定义一个数据元素，表明：

1）每个数据元素的绝对地址在数据块中被创建（表 5-1 的地址纵列）。程序员不需要在这个纵列中输入内容。程序员完成表的每一行时，STEP 7 编程软件分配地址。每个绝对地址识别数据元素保存在哪里，即数据块在存储器中的开始地址。在例子中，两个布尔位保留给头两个元素，并且 STEP 7 分配第一个可用的数据字节的低两位。下一个数据元素，一个字，分配两个字节（开始于字节 2）。另外的空间保留给一个 DINT 数、一个实数和一个字符串。注意字符串声明包括大小声明，因而它只包括给 20 个 ASCII 码的空间（加上一个 2 字节的头），而不是默认的 254 个 ASCII 码。整个数据块使用 32 个字节。程序通过绝对地址操作在数据块中位、字节、字和双字。一旦创建和打开数据块后，共享数据块中的数据可以通过以下的绝对寻址格式寻址：

```
DBX x.b   独立的位,"x"是字节序号(从0到65 535,当前的 S7 PLC 少一些),"B"是位序号(从0到7)
DBB x     字节序号"x"
DBW x     字序号"x";包括两个连续的字节
DBD x     32位双字,由4个字节组成
```

2）给每个元素一个符号名，符号名是可选的，但通常建议使用。如果输入一个符号名，通过使用那个名字，在用户程序中可以对数据项寻址。符号名可以用来寻址位、字节、字或者双字，以及更大的数据元素。

3）数据类型列定义保留多少存储空间给数据项，下一列允许程序员（选择性地）分配初始值（以常数形式）给每个数据项。几乎任何类型的数据都可以声明作为共享数据块中的数据元素，包括定时器常数，但不包括定时器、计数器或块识别数，或指针数据类型，或任意型数据。

复合数据类型，如数组和结构体，也可以声明作为数据块中的元素，并且如果它们没有定义为 UDT，甚至可以在数据块中定义。因此在数据块中的复合数据类型元素可以在用户程序中作为单独的数据项操作，或者组成复合数据元素的简单的数据元素可以单独地操作。表 5-2 显示一个数据声明表，它保留四个复合数据类型元素在存储器的数据块中：DATE _ AND _ TIME 元素、字符串元素、数组和结构体元素。注意变量名可以声明作为复合数据元素，有时可以作为复合数据项中的数据元素。符号名的使用通常是复合数据项和它们的组件的寻址的惟一实际办法。

在表 5-2 的数据声明表中，符号名分配给 DATE _ AND _ TIME 元素、字符串元素、数组元素、结构体元素和结构体中的简单元素，但不可以是数组中的简单元素。一旦数据声明定义的数据块被用户程序打开后，声明的符号名可以用来代替数据元素的绝对地址。

表 5-2 声明了复合数据元素的 STEP 7 数据声明表

数据块 c:\Plant _ 2\Stn _ 3\DB5				
地 址	名 称	类 型	初始值	注 释
0.0		STRUCT		
+0.0	today	DATE _ AND _ TIME		
+8.0	message	STRING [20]		
+28.0		STRUCT		

（续）

数据块 c:\Plant_2\Stn_3\DB5				
地　址	名　称	类　型	初始值	注　释
+0.0	matrix	ARRAY [0..4, 0..9]		
*2.0		WORD		
=100.0		END_STRUCT		
+128.0	misc_data	STRUCT		
+0.0	light	BOOL		
+2.0	sum	WORD		
=4.0		END_STRUCT		
		END_STRUCT		

数据声明表也定义了保留给被声明元素的存储器数量。字符串元素声明给足够包含 20 个 ASCII 码字符元素。数组声明作为二维的，五行（从 0 到 4），10 列（从 0 到 9）的相似数据项（字）。关键字 STRUCT 和 END_STRUCT 必须用来开始和结束数组声明。数组的大小声明在关键字 STRUCT 后面，紧接着是数组中数据元素的类型。数据块声明表也定义了结构体（名字 misc_dat），包含两个简单的元素（叫作 light 的布尔元素和叫作 sum 的字元素）。关键字 END_STRUCT 表明结构体的定义完成了。（下一个 END_STRUCT 完成数据块的声明。）

一旦用户程序打开数据块，在数据块中的数组元素可以通过数组的符号变量名寻址，接着是数组维数的一个元素。例如：

```
matrix[0,9]        字的地址，它占用在表5-2中声明的数组的首行，最后一列
```

为了寻址结构体数据元素的单独的子元素，使用通过点（.）分开的变量名，起始为高层数据元素，向下工作到底层。例如：

```
misc_data.sum    结构体的字子元素的地址
```

如果结构体（例如 more_data）定义包括另一个结构体（例如 sub_data），sub_data 包含一个布尔型的三维数组（例如 set），数组中的每一位可以用如下方式寻址：

```
more_data.sub_data.set [2,5,3]
```

如果结构体和数组数据类型不能提供用户所需的灵活性，程序员可以使用 STEP 7 编程软件的数据类型编辑器定义其他的用户数据类型（叫作 UDT 数据类型），与程序员创建共享数据块的方法完全相同。除了 STEP 7 编程软件的选项，创建过程是相同的，在实际创建数据块前，必须选择 Data Block Referencing a User Data Type 来代替对话框中的 Data Block 选项。如果 UDT 在创建共享数据块前被定义，UDT 可以被声明作为在数据块中的元素，因而给在数据块中 UDT 数据元素的所有元素保留存储空间。UDT 型元素的组件的初始值、符号名、数据类型在创建 UDT 时被声明，并且不可以在创建 UDT 中使用的共享数据块的过程中改变。

13. Siemens STEP 7 立即数据块

立即数据块用来存储数据，这些数据只能在特定的功能块程序中使用。

功能块是其他程序可以调用的子程序。功能块被调用的时候，调用必须包括将要被打开的立即数据块序号，如图 5-20 所示。（注意：STEP 7 对于立即数据块和共享数据块都使用

符号“DB”。）

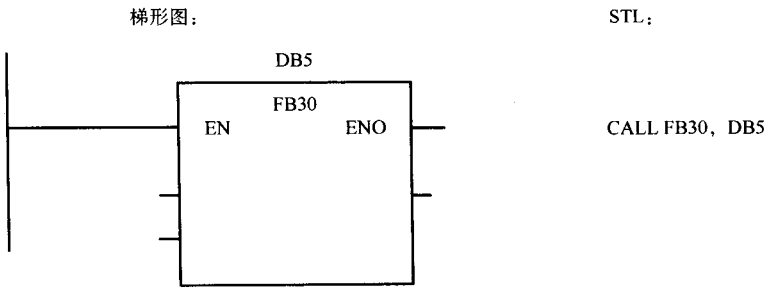


图 5-20 STEP 7 指令（梯形图和 STL），调用一个功能块程序。它包括立即数据块的特性。

立即数据块的创建不同于共享数据块的创建。当程序员使用 STEP 7 编程软件创建立即数据块的时候，必须从创建数据块的对话框中选择读“data block referencing a function block”的菜单选项。程序员随后输入使用这个立即数据块的功能块序号。STEP 7 自动地创建一个立即数据块，包含数据元素和在功能块变量声明表中（除了声明作为 temp 的数据元素）声明的初始化值。在程序员创建立数据块前，功能块必须与一个完全的变量声明表一起编程。每个功能块能够创建几个立即数据块。表 5-3 是在功能块中的变量声明表的例子。涉及这个功能块的立即数据块包含带有定义在变量声明表中的符号名、数据类型和初始值的数据元素，除了变量声明作为“temp”。当看到立即数据块内容的时候，编程器显示声明类型列。

功能块中的变量声明表表明：

1）立即数据块或者局部堆栈存储器（在这一章的下一节中将讲述局部堆栈存储器）中的每个数据元素的绝对地址（在表 5-3 的地址列）。程序员在这一列中不用输入内容；程序员完成每个表行的输入后，STEP7 编程软件会分配这些绝对地址。在这个例子中，立即数据块中保留两个位给两个布尔数据单元来代表一个开关和一个执行器的状态。另外两个字节（从位 2.0 开始）保留给一个 INT（整型值），开始于位 4.0 的另 2 个字节保留给另一个 INT。下面的绝对地址形式可以用来寻址立即数据块的内容：

DIX x.b 单独的位，“x”是字节序号（从0到65 535，目前的 S7 PLC 要少一点），“b”是位序号（从0到7）

表 5-3 简单的变量声明表，STEP 7 功能块程序的一部分

地 址	声明类型	名 称	数据类型	初始值	注 释
0.0	in	switch _ 1	BOOL		
0.1	out	cylinder	BOOL	False	
2.0	in _ out	loop _ cnt	INT	20	
4.0	stat	work _ dat	INT	0	
6.0	temp	push _ cnt	INT	0	

DIB x “x”是字节序号
DIW x “x”是字序号，包括两个连续的字节
DID x “x”是32位的双字序号，由4个字节组成

注意绝对地址在给声明类型为“temp”的元素的 6.0 开始。这反映了这样一个事实，

变量保存在存储区域而不是立即数据块中。声明为“temp”的变量只保存在局部堆栈区域存储器中，将在下一节中讨论。

2）*声明类型列识别源、目的和每个数据的生命周期。*可以给每个声明变量声明任意数据元素的序号，但它们必须在变量声明表中按以下顺序声明：

- “in”参数在“out”参数之后，“out”参数在“in-out”参数之后。这些参数声明类型定义是否把数据元素值提供给功能块（“in”），是否从功能块（“out”）写入，或者两者皆可（“in-out”）。我们将在第 8 章的参数传递中讨论。
- 接下来是“stat”参数。这些数据元素通过功能块给工作值使用。“stat”参数值在立即数据块中保持，直到下一次功能块被相同的立即数据块调用。
- “temp”参数列在最后。在功能块结束时，它的值将丢失，尽管功能块的变量声明表可以包括初始值，在下一次执行带有任意立即数据块的功能块时，“temp”变量将被设置。

3）*局部块符号名*可以被输入但不一定要输入。如果输入，这个功能块可以通过一个“#”后跟随局部块符号名来寻址数据项，代替通过绝对寻址寻址数据项。使用符号名字允许程序员更容易的寻址一些数据项。下面的例子显示在表 5-3 中来自变量声明表中的数据怎样通过局部块符号名寻址：

```
#cylinder        意思与 DIX 0.1相同
#work_ dat       意思与 DIW 4相同
```

STEP 5 允许程序员使用一个符号表来定义全局符号名给共享的存储器地址。这些名字也可以在功能块中使用。在寻址局部块符号名的数据的时候，一定要包括“#”，或者程序可能通过相同的名字偶尔寻址全局变量。

4）*数据类型*列定义必须保存给一个数据项多少个局部堆栈数据存储器，*初始值*列允许程序员以常数形式分配初始值给每个数据项。可用的数据类型和常数格式包括前面介绍的所有数据类型，加上一些 Siemens 描述的类型，包括：

- *定时器*：导致功能程序保留两个字节的存储空间，接收参数值，它由字母“T”和从 0 到 255 之间的数组成，从调用程序中识别定时器。
- *计数器*：像定时器参数，除了调用程序必须传递字母“C”和在 0 和 255 间的数，识别一个计数器。
- BLOCK _ FB、BLOCK _ FC、BLOCK _ DB、BLOCK _ SDB：每个都会导致函数保留 2 个字节的存储空间，并接收给逻辑块的地址（FB 或者 FC）或者给来自调用程序的数据块（DB 或者 SDB）。调用的程序可以通过全名传递参数——后接从 0 到 255 的数的块类型识别符（FB、FC、DB 或者 SDB）——或者可以通过符号名指定一个数据块。
- *指针*：用来传递一个地址类型到函数或者功能块程序中，以便函数/功能块程序可以读取或者写那个地址。指针通常可以被传递给函数或者功能块，但不一定声明是“out”数据声明类型。（查看用户手册。）记住指针类型参数保存在 6 字节的存储空间，如图 5-16 所示。

存储区域的许可代码包括给先前局部堆栈存储器的代码。这表明程序可以调用函数（或者功能块）并可以传递一个指针参数，它允许函数（或者功能块）访问属于调用程序的局部

堆栈存储器。

- 任意型：立即数据块保留 10 个字节的空间，包含一个到任意型数据集的任意型指针。由于有一个标准的指针参数，任意型指针参数可以给功能块来访问存储区域，这个存储区域包含一个或者更多的简单或者复合数据类型的数据元素，包括调用程序的局部堆栈存储器的存储区域。

因为对应同一个功能块可以创建多于一个的立即数据块，所以调用功能块的程序必须包括与功能块一起打开的立即数据块的地址。功能块可以关闭一个立即数据块并打开另一个不同的立即数据块。图 5-21 显示如何用梯形图和 STL 编程打开新的立即数据块。注意打开指令可以使用地址区分立即数据块（DIX）和共享数据块（DBX）。

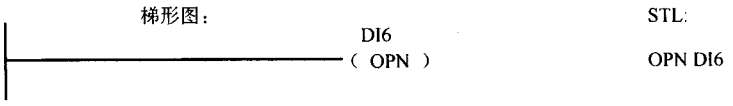


图 5-21 打开一个立即数据块的 STEP7 指令

14. Siemens STEP 7 局部堆栈数据

局部堆栈存储器用来保存 STEP 7 程序使用值，但当 STEP 7 程序结束时，局部堆栈存储器将不再需要。为了便于理解，需要对结构化编程有一定了解。结构化编程在第 8 章将详细讲述。

STEP 7 操作系统支持组织块（OB）程序，由用户编写，在扫描循环中执行。这些组织块中的一个（OB1）包含一个在每个扫描循环中执行的主用户程序。其他组织块包含的用户程序只有在特定情况下运行。OB1 比其他组织块有更低的优先权，以便当需要另一个组织块运行的时候，操作系统中断 OB1，当其他组织块完成的时候，重新执行 OB1。STEP 7 包含可以提供多达 80 个不同优先级的组织块程序。每个优先级有一个固定量的局部堆栈存储空间可以使用，它和给其他优先级程序使用的局部堆栈存储空间是分开的[⊖]。

在任何组织块中的程序可以调用子程序，这些子程序被称作函数和功能块。每个组织块、函数和功能块程序被分配一些局部堆栈存储器，当执行完程序后，释放局部堆栈存储器。为了 STEP 7 操作系统定义的目的，一些局部堆栈存储器可以自动地使用。

每个组织块、函数和功能块寻址起始于地址 L 0.0 的局部堆栈区域，尽管一些程序可以同时分配来自局部堆栈的存储空间。每个用户编写的程序必须包括一个变量声明表，变量声明表表明程序需要多少局部堆栈存储空间。表 5-3 显示了一个对于功能块的简单变量声明表的形式和内容。组织块和函数也包括变量声明表。

在功能块程序中的变量声明表一部分用来作为立即数据块模板，另一部分声明功能块需要的“临时”变量。当功能块被调用的时候，STEP 7 保留局部堆栈存储器给“temp”变量。“temp”变量（如在表 5-3 中定义的整型变量）可以通过绝对地址（例如 LW 0）或者符号地址（例如 # push _ cnt）寻址。

函数变量声明表看上去和功能块的变量声明表相似，如表 5-4 所示。注意，函数地址列

⊖ S7 PLC 的可用存储器的数量是不同的，参看你的手册集。如果程序试图使用太多同一优先级的局部堆栈存储器，或者所有程序使用的各优先级的所有存储器太多，PLC 将出错。

起始在 0.0，当它遇到首个“temp”变量时，不再重新计数。函数的所有变量保存在局部堆栈存储器。因为函数不可以有立即数据块，所以它的变量声明表不能包括声明类型为“stat”的变量㊀。可以在功能块中声明的变量的数据类型与可以在函数中声明的变量的数据类型有一点不同，但是声明变量的规则与变量寻址的规则与在功能块中是相同的。

表 5-4 STEP 7 函数的一个变量声明表的格式和内容

地　址	声明类型	名　称	数据类型	初始值	注　释
0.0	in	switch _ 1	BOOL		
0.1	out	cylinder	BOOL	0	
2.0	in _ out	loop _ cnt	INT	20	
4.0	temp	push _ cnt	INT	0	

组织块的变量声明表只能包含声明为“temp”的变量，并且它的所有工作数据必须在局部堆栈存储器中存储（除非组织块打开一个共享数据块）。当 STEP 7 操作系统调用组织块时，它自动地打开 20 个字节的局部堆栈存储器，并把数据放到存储器中，给在组织块中的程序使用。每个 STEP 7 组织块预编一个变量声明表，定义了它使用的 20 个字节的局部堆栈器，但是程序员可以把声明的更多的“temp”变量加到变量声明表中。

15. Siemens STEP 7 存储器的间接寻址

存储空间可以通过使用存储器间接寻址格式寻址。存储器中的指针数据有一个地址，表明将从哪里读入数据，或者把数据放到哪里㊁。在存储器间接寻址中使用的指针不同于在数据常数节和本章的立即数据块节中描述的 6 字节的指针数据类型。在存储空间间接寻址中使用的指针是一个数，它必须以 16 或 32 位保存，如图 5-22 所示：

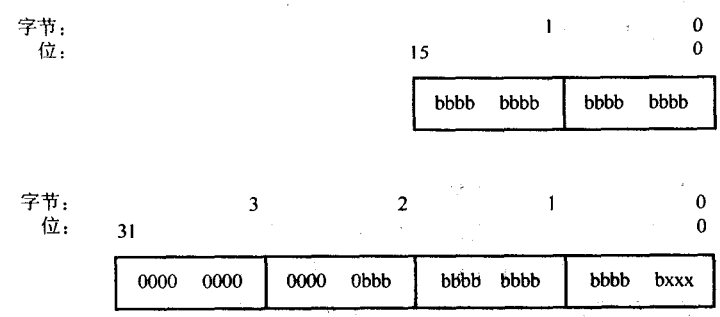


图 5-22 在存储器间接寻址中使用的 S7 指针

- bbb bbbb bbbb bbbb b 是无符号16位数，指明一个字节地址(从0到65 535)。
- xxx 是一个3位的值，指明一个从0到7的数；如果指针“指向”字节、字或者双字，它必须是0。

16 位指针格式用来指向计数器、定时器、数据块、函数或者功能块。它由一个在 0 和 65 535 间的无符号二进制数组成。32 位指针格式用来指向在存储器中的一个字节或者指向哪个字节中的一位。

㊀ 任何程序（函数，功能块，组织块）都可以打开一个共享数据块，并可以读写这个数据块中的数据，所以共享数据块可以在程序结束后获得工作数据

㊁ 因为有时 STEP 7 操作系统自动地使用局部堆栈存储器，Siemens 建议不要对局部堆栈存储器的区域使用间接寻址

存储器间接寻址格式用在下面格式的指令中：

```
instruction p1s1[p2s2x]
```

“instruction”是可以使用存储器间接寻址的指令。

“p_1”是识别存储器区域的前缀，存储区域包含指针指向的数据值。

“s_1”表明数据值的大小（如果有必要指定大小）。

方括号告诉 PLC 这是存储器间接寻址，在括号里的地址是指针被保存的地址。

“p_2”是前缀，表明指针存储在存储器的哪一区域（M. L. DB 或者 DI）。

“s_2”表明指针数据的大小（W 或者 D）。

“x”是指针的地址号。

例如：

```
L+ 25
T MW4          // 把“25”放入 MW4作为指针。
L C [MW4]      //使 PLC 从 MW4读取指针(+25),然后从指向(C25)的计数器加载累加值。因为计数器的累加值总是16
               位的,所以它不需要指定大小。
```

和

```
L PW [MD8]     //使 PLC 从 MD8读取一个32位的指针双字,然后从指针指向的外围 I/O 字加载一个16位值。
```

给存储器间接寻址的 32 位指针，必须通过使用在本章描述数据常数节中描述的指针常数格式来创建。这个指针常数作为 32 位数保存。使用 32 位指针创建它们的过程如下：

```
L p# 24.6
T MD8          //把32位指针放入 MD8中
A I [MD8]      //检测输入映像存储器的字节24中的位6
```

符号地址可以在本节的例子中代替绝对地址使用。如果符号名 My _ Pointer 声明代表地址 MD8，上面显示的程序可以写成：

```
L p# 24.6
T “My-Pointer”
A I [“My-Pointer”]
```

16. Siemens STEP 7 寄存器间接寻址（变址寻址）

寄存器间接寻址是一个非常有效的变址寻址格式。它用指针作为基地址和偏移地址。S7 的 CPU 包括两个 32 位的地址寄存器，它可以用来存储 32 位的指针作为基地址值给寄存器间接寻址。指针常数值通过以下指令保存到地址存储器：

```
LAR1或者 LAR2  加载一个常数或者存储空间的内容到地址寄存器1或者2
```

寄存器间接寻址中使用的 32 位数格式和在存储器间接寻址的 32 位数的格式有点不同。它必须如图 5-23 格式显示：

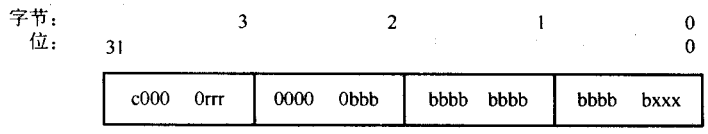


图 5-23 S7 寄存器间接指针格式

■ bbb bbbb bbbb bbbb b 和 xxx 是字节和位数，和在存储器间接寻址中一样。

- c 是：
- 0 如果指针是区域内部型，意味着使用这个指针的指令也必须包括一个被存取的专用的存储器区域。
- 1 如果这个指针是跨区域型，这个指针表明被存取的存储器区域。
- rrr 是一个 3 位代码，如果指针是跨区域型，如果 c＝1，指示哪个存储器区域被使用。代码包括：

```
000代表 P    010代表 Q    100代表 DBX
001代表 I    011代表 M    101代表 DIX
111保留给调用这个程序的程序的局部存储器
```

如果指针的最高位是“0”，表明是区域内部指针，使用寄存器间接寻址的指令必须包括下面格式的地址：

```
instruction    ps[ARn,0]
```

- “P”是指令操作的数据值的区域识别符（I、Q、P、M、L、DB 或者 DI）。
- “S”是这个指令操作（B、W 或者 D）.A 的数据值的大小。如果指令使用位值，则不需要大小值。
- 对地址寄存器 1（AR1），“n”为 1，对地址寄存器 2（AR2），“n”为 2。AR1 或者 AR2 必须包括一个 32 位的寄存器间接寻址指令作为变址地址的基端口。
- “o”是变址寻址的偏移地址。它可以以 32 位存储器间接指针常数格式输入，或者作为地址来保存 32 位存储器间接地址指针。如果作为指针常数输入，必须以如下的形式：

```
P# x.n
```

“x”是基地址字节数，“b”是基地址位序号。如果指针指向一个字节、字或者双字，“b”必须是 0。

如果指针最高的位是“1”，表明是跨区域指针，使用寄存器间接寻址的指令必须包括下面格式的地址：

```
instruction    S[ARn,b]
```

和区域内部寻址相同，除了不需要使用存储区域标识符。

下面例子显示寄存器间接寻址的使用。（加载指针常数和传递它们给指针寄存器的指令不是必须在使用寄存器间接寻址的指令前）：

例 1　使用两个指针常数：

```
LAR1 p# 24.6          //加载区域内部指针的常数到寄存器1地址的字节34,位6
A I [AR1,p# 2.1]      //检查输入映像字节26(24+2=26),位7(6+1=7)
```

例 2　使用来自存储器的偏移指针和对应基地址的指针常数

```
L p# Q5.2             //加载对应跨区域指针的常数到累加器1的输出映像位 Q5.2
T MD4                 //传送该常数到数据存储器地址 MD4
LAR2 MD4              //复制来自 MD4的指针到地址寄存器2
S [AR2,p# 1.2]        //输出映像位 Q6.4置位
```

例 3　使用来自存储器的两个指针

```
L p# M6.0             // 加载对应跨区域指针的常数到数据存储器字节6
```

```
T LD8                  // 传送该常数到本地堆栈存储器 LD8
L p# 1.0               // 为一个指针加载32位存储器间接类型指针常数
T LD12                 // 传送该常数到局部堆栈内存 LD12
LAR1 LD8               // 从 LD8加载带有指针的地址寄存器1
L D [AR1,LD12]         // 从数据存储器字节地址 MD7加载一个双字
```

两个指令，+AR1 和+AR2，可以用来增加或者减少地址寄存器中的指针值。+AR1 和+AR2 在第 6 章的关于 Siemens STEP 7 算术/逻辑指令小节中详细讲述。

CQM1

5.5.6 OMRON CQM1 中的数据寄存器和可寻址数据

CQM1 有本书中讨论的 PLC 的最不复杂的工作数据存储器结构体，并且有最少的常数输入方法。

1. OMRON CQM1 数据常数值

数据常数在 CQM1 程序中只可以作为一个 10 进制或者 16 进制值来输入，并且必须有一个前缀“#”。操作存储器数据的一些 CQM1 指令假定，在使用这些数据前数据是 BCD 码格式，而其他指令假定是无符号二进制数或有符号二进制数。一些指令自动地使用 16 位数据字，其他自动地使用 32 位双字。输入的数据常数必须以十进制或者十六进制形式，必须包括一个合适的十进制或者十六进制数字的序数。

2. OMRON CQM1 存储寄存器区域

这是一个有 8 个独立的可寻址存储区域。

(1) OMRON CQM1 I/O 映像数据

内部寄存器（IR）存储区域保留给包含输入映像和输出映像数据的 16 位的数据字，但比输入和输出映像需要更多的存储空间。这个存储区域的剩余部分可用来存储其他工作数据。MACRO 指令自动地使用这个存储区域的一部分。在 CQM1 的一些模块中，另外一些存储空间保留给 PLC 模块的指定特性。OMRON 的文献表明 OMORN 打算在未来的 CQM1 模块中使用部分额外的其他特殊特性的内部寄存器存储空间，程序员应该避免使用这些地址。在外部寄存器（IR）的存储区域中数据字可以有如下的寻址格式：

000到243

（如果没有前缀输入，“IR”前缀是默认的）

- 对输入映像，从 000 到 011。
- 对输出映像，从 100 到 111。
- 特殊函数或者工作数据可以用其他地址。

可以在如下格式中寻址每一个数据位：

xxx00到 xxx15　　“xxx”从000到243

(2) OMRON CQM1 定时器/计数器寄存器

定时器/计数器寄存器（TC）存储区域用来保存 16 位 BCD 数，它们代表或者是定时器或者是计数器的当前值（PV）（也就是说，不能同时是计数器 5 和定时器 5）。通过使用下面寻址格式来寻址存放当前值的字，由地址是包含定时器（TIM）还是计数器（CNT）决定：

TIM 000到 TIM 511　或　CNT 000到 CNT511

使用相同的地址检测定时器或者计数器的一个位结束标志。指令表明应该使用当前值或

者完成标志，并且 PLC 将存取正确的数据输入。例如：

```
TIM 025    可以用来作为定时器25的当前值地址,及作为定时器25的完成标志的地址。
```

(3) OMRON CQM1 数据存储器

数据存储（DM）区域包含 16 位数据字，它们中的一些可以自动地用来作为配置和状态信息，剩余部分给用户数据存储使用。数据存储字可以以如下方式寻址：

```
DM 0000到 DM 6655
```

DM 6144 到 DM 6655 用来给配置和状态信息使用，用户程序不可以改变它们。当 PLC 不在运行模式时，一些配置字可以通过编程器读取或者改变。

CQM1 不允许寻址 DM 存储区域中的单独位。

(4) OMRON CQM1 保持寄存器

保持寄存器（HR）存储区域是 EEPROM 存储器，所以它们包含的 16 位用户数据字不受电源掉电影响。在下面范围内，字是可寻址的：

```
HR 00到 HR 99
```

并且每一位可寻址：

```
HR xx00 到 HR xx15    "xx"从00到99
```

(5) OMRON CQM1 专用寄存器

专用寄存器（SR）的存储区域包含 8 位字节，CQM1 使用它们中的单独位作为状态和控制目的。字节按以下形式寻址：

```
244到255
```

在 IR 存储区域序号的结束处开始计数。"SR" 前缀是默认的，不需要输入。

程序员通常以下面形式寻址 SR 字节的每一位：

```
xxx00 到 xxx07 "xxx"从244到255
```

附录 C 中包含 SR 位的描述。

(6) OMRON CQM1 辅助数据存储器寄存器

辅助数据存储器寄存器（AR）区域包含其他 16 位数据字，包含控制和状态位，可以按字的方式寻址：

```
AR xx00 到 AR xx15    "xx"从00到27
```

附录 C 包含 AR 位的描述。

(7) OMRON CQM1 临时寄存器

临时寄存器（TR）仅包含一个数据字，它只能按位寻址，格式如下：

```
TR 0 到 TR 7
```

执行复杂布尔梯形图语句过程中，CQM1 操作系统使用这些位来包含临时保存的逻辑操作状态结果。当它们评价分支梯级时，所有的 PLC 必须保存临时的逻辑结果状态，但 OMRON 允许寻址这些保存的状态，所以它们可以拷贝到其他存储器位中，便于以后使用。

(8) OMRON CQM1 连接寄存器

在通过 RS-232 端口进行数据通信的过程中，连接寄存器（LR）区域是可用的，这个将在第 13 章看到，但是，如果不需要进行数据通信，这个存储区域可以用作其他数据存储。

16 位字可以按如下方式寻址：

LR 00 到 LR 63

单个位寻址如下：

LR xx00 到 LR xx15 “xx”从00到63

图 5-24 显示检测专用寄存器字 253 的位 15 的梯级，（每次 PLC 运行的循环里，这个位工作）。如果这个位工作，梯级执行两个 move 操作：

1）它拷贝来自保持寄存器 10 的 16 位数（包含 PLC 最后一次运行后的值）到输出模块 102 的输出映像数据字中（因而 16 个数字输出执行器按预定的状态启动）。

2）拷贝常数值 2000 到数据存储空间 0011。

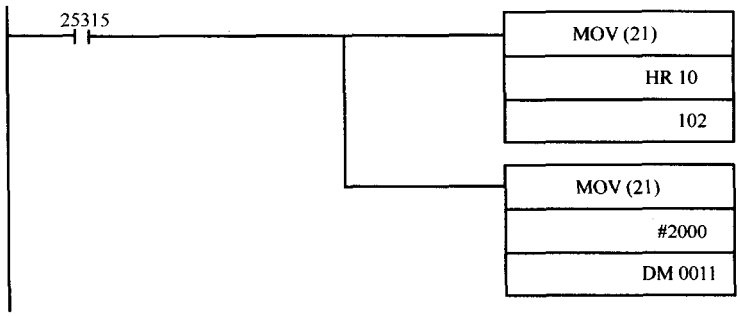


图 5-24 传送两个数据字的 CQM1 程序

3. OMRON CQM1 间接寻址

在间接寻址中需要两个数据存储器（DM）存储空间。其中一个 DM 存储空间必须包含一个 BCD 数，表明另一个 DM 空间的地址。“另一个”DM 存储器包含指令操作的数据值。例如，如果 MOV(21)指令，通过间接寻址，写一个值到 DM 2000，目的地址数（2000）必须在另一个 DM 存储器空间中存在。（例如，在 DM0010）。MOV(21)指令包含“＊DM 0010”形式的目的地寻址，“＊”告诉 CQM1 这是间接寻址。CQM1 使用来自 DM 0010 的 BCD “2000”值作为实际的 DM 目的地址（DM 2000）。间接寻址可以用来访问在 DM 存储区域的一个存储地址。

4. OMRON CQM1 变址寻址

只有两个 CQM1 指令允许变址寻址。单字传递指令 DIST(80)，复制来自源地址 16 位数据字，并复制它到目的地址中，目的地址通过使用基地址和偏移地址来指定。类似的，数据收集指令 COLL(81)，复制来自源地址的 16 位数据字到目的地址，但源地址可以是基地址和偏移地址。两个指令在图 5-25 中显示。注意两种都需要控制（C）参数。如果控制参数值在 DIST(80)指令中低于 9000，或者在 COLL(81)指令中低于 8000，它作为偏移值使用[⊖]。控制参数可以用常数形式的指令输入（例如，＃0045），或者指令可以包括实际偏移量的地址（以 BCD 码形式），以便在程序中的其他指令可以操作偏移地址。基地址（在 DIST(80)中的 DB 参数，或者在 COLL(81)中的 SB 参数）只可以作为地址输入，因而在那个地址的 BCD 码值也可以通过其他程序指令操作。

⊖ 如果控制字超出这些范围，DIST(80). COL(81)指令与简单地移动一个数据字的操作稍有区别（见第 7 章）。

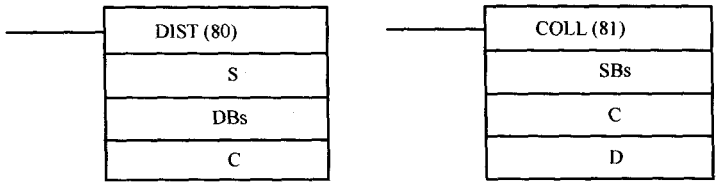


图 5-25 允许 CQM1 在移动数据字变址寻址的指令

5.6 故障检修

你使用存储器遇到过困难吗？很容易遇到！并且有时候这些困难的原因很难发现。一个有效的问题解决工具是在线数据监视特性（Siemens 称作变量访问表）。所有好的编程软件都提供数据监控功能。当程序运行时，程序员可以使用这个屏幕来监控（并改变）存储器的内容。有时程序软件需要程序员使用其他屏幕（例如，系统状态屏幕或者系统配置屏幕）来访问数据屏幕不可以访问的存储器。有时程序可以拷贝来自一部分存储器的数据到另一部分，这样问题更容易解决（例如，允许在相同的数据屏幕的不同存储区域查看数据）。

在一些 PLC 程序包中提供直方图特性，可以用来帮助找到与数据相关的问题。PLC 运行时，程序员可以使用直方图特性使编程器跟踪某一地址数据（或者有时是同时多个地址），然后显示一个时基图表，表明被监控过程中，这个数值是怎样变化的。如果在相同的时间段生成多个直方图，程序员可以在这个时间段检测相关数据是如何变化的。在第 15 章将描述直方图。

一个普遍的程序错误是偶尔地输入错误地址到程序中。如果写数据到错误寻址的存储空间，可能后面不能够发现，但是如果偶尔地写那些数据到包含了不想丢失的数据存储区的时候，问题变得更加复杂。

记住定时器和计数器的操作存储空间。如果偶然地对两个定时器使用相同的定时器地址，在程序中有两个指令操作相同的存储器，其中一个想保持时间，而另一个想把时间设置成 0。如果给两个计数器相同的地址，可能会发现计数器的值在每个扫描循环都增加，因为一个计数器指令有假控制逻辑，而另一个是真控制逻辑，因而计数器会认为它的逻辑在每个扫描循环是真。

在程序中被寻址的存储空间必须是存在的，因而需要检查手册来决定哪些存储空间是可用的，哪些存储空间可以改变，这是很重要的。在一些 PLC 中（例如，Allen-Bradley），数据存储器必须在使用它之前“创建”。当然，存储器在这个过程中不用实际创建，它只是保留给指定类型的数据。程序员在程序中写地址时，Allen-Bradley 程序软件通常保留存储器，但必须遵守一些规则（例如，如果数据文件在一部分程序中用来保存一个类型的数据，相同的程序文件不可以用在其他地方保存其他类型的文件）。在程序运行时，一些存储区域，经常是包含系统配置和状态的区域，是不可以被写入的，或者只可以通过使用高级的程序结构或者指令写入。当程序员向保留存储地址输入数据时，程序软件不会警告程序员，所以，程序员不得不在程序执行时进行监控，来发现这个程序是否改变存储器的内容。

在一些 PLC 中，一些存储空间是保持型的，表明 PLC 不在运行状态时，甚至是掉电的情况下，它们中的内容可以保留。当 PLC 恢复运行它的程序时，在这些存储器中的值和 PLC 最后一次运行时的值是一样的。在 Allen-Bradley PLC 中，所有的存储器（甚至输出映像）也是保持型的。一些 Allen-Bradley 程序元素（例如，计数器、延时定时器等等）可能

会在 PLC 进入运行模式时改变它们控制的存储器，这是因为它们认为控制逻辑是真。任何时候 Allen-Bradley 程序从 PLC 上载到主机时，上载的数据文件包括所有的数据存储空间中的内容，当程序后来下载到 PLC 时，所有的数据也重新下载到 PLC 中。其他的 PLC 有易失性存储器，这表明 PLC 将在开关打开或者转换到运行状态时清除它们，或者有的部分存储器区域是非易失性的，部分是易失的。Siemens PLC 有计数器，标志/位和数据块存储器，它们中的部分是非易失的，部分是易失的。程序员有责任读用户手册来决定，在 PLC 开关打开后，哪些存储器自动地清除，哪些将不清除。

在程序执行时（改变地址指针或者偏移值），高级的寻址模式：变址寻址和间接寻址，可能会导致寻址问题，因为程序可以改变地址。程序可能会产生并不存在或者不应该被写入的地址。如果程序试图访问不存在的存储器地址，一些 PLC 会自动出错，其他 PLC 将允许程序访问“超出存储器界限”的数据。例如，如果程序试图重置一个高过最高允许的计数器地址的预设值，一个在计数器累加值存储位置外的存储地址可能被清除。如果 PLC 可以超出存储器界限访问数据，这通常（至少）将设置一个错误位，程序应该在扫描循环结束前检查这个位。（你会明白为什么大多数 PLC 生产商宁愿他们的 PLC 出错。）

习题

1. 什么是输出映像列表？在标准 PLC 循环中的哪部分 PLC 使用这个数据？
2. 如果 PLC 在运行模式，并使用数据屏幕改变在输入映像地址的值，PLC 会按命令改变存储器的值吗？解释你的回答。
3. 以下数据单位有多少位：
   1）字？
   2）字节？
   3）浮点数（最普通的大小）？
   4）位？
   5）ASCII 码？
4. 一些 PLC 允许使用浮点数。什么是浮点数？
5. 什么是指针？是在变址寻址还是在间接寻址中使用？
6. 概括地描述使用符号寻址的好处。
7. 如果 PLC 有模拟输入模块，并且一个有符号 16 位数值用来代表＋10 到－10Vdc 的范围：
   （a）什么值代表＋10V？
   （b）什么值代表－10V？
8. PLC 中的所有 RAM 都可以寻址吗？（是或者不是；不需要解释）
   对于你正在学习使用的 PLC 类型
9. 程序中的数据值可以用 16 进制格式输入。当输入常数的时候，可以使用其他什么格式来代表数据。（就你正在学习使用的 PLC 做出回答，对每个格式做出解释。）
10. 以下数据在存储器区域的第一个可用地址是多少：
    （a）输出映像数据？
    （b）定时器累加数据？
    （c）定时器完成位？

(d) 有符号整型字?

11. 列出其他存储器区域，各举一个例子，并指出各个存储区保存什么内容。对于不代表CPU模块存储器的地址，同样作出回答。

12. 填表

(a) 如果表格输入的是地址，指示一个数字输入或数字输出触点，使用正确的列来描述你将会在PLC中的哪里找到一个连接到传感器或执行器的螺旋触点。

(b) 如果表格输入的是常数，使用正确的列来描述它代表什么类型的数据。

(c) 如果表格输入的是另一个类型的地址，指出它的存储器区域将要包括什么。

PLC地址/常数	数据大小	存储区域类型/IO位置/常数类型	PLC地址/常数	数据大小	存储区域类型/IO位置/常数类型
PLC-5			Siemens S5		
I：223/14			F0.1		
O：020			FY2		
B3/20			FW2		
N7：20.2			IB4		
T4：0			Q2.14		
T4：0.ACC			PY4		
T4：0.TT			12		
F8：6			C2		
5.3			KH12		
&H1234			KT12.2		
SLC 500			Siemens S7		
I：12/14			M10.1		
O：10			MW10		
B3/20			IB4		
N7：20.2			Q2.14		
T4：0			PIB4		
T4：0.ACC			>0		
T4：0.TT			B#16#12		
M0：2.6			T#2H5M		
5.3			P#M10.0		
&H1234			OMRON CQM1		
			10304		
			003		
			HR 04		
			DM 1304		

对于你正在学习使用的Allen-Bradley PLC

13. 在Allen-Bradley PLC：B3/4相同的位是B3：____，B3/____和B3：1.2是相同的位。

14. 存储器中保留____数据字给Allen-Bradley PLC中的每个计数器。这些数据字中包含什么?

15. 对于每一个定时器，Allen-Bradley PLC的存储器中保留了____数据字。这些数据字包含什么?

如果你正在学习使用Siemens S5 PLC

16. 如果在Siemens S5 PLC中的模块的模拟地址起始是64，你将使用什么地址来写（传送）

一个16位的值到在第二个插槽中智能I/O模块的第一个通道？如果通过检测来自那个模块的数据的首字的低位，检查I/O模块的状态。你将使用什么位地址？

17. Siemens S5/Westinghouse PC503的地址T001代表的两个数据项是什么？

如果你正在学习使用Siemens S7 PLC

18. 一次可以打开多少S7数据块？分别是什么类型的数据块？
19. 哪里可以声明只在单一的（局部）程序中有效的符号名？
20. 什么是复合数据类型？使用一句话描述每个数据类型包括什么类型的数据？
21. 在哪里可以声明一个立即数据块的结构体？
22. 在功能块的数据声明段声明的“in-out”参数类型是什么？

如果你正在学习使用OMRON CQM1 PLC

23. 在OMRON CQM1 PLC中识别3个数据存储区域，指出存储区域应该保持的存储区域的前缀和数据类型。

对于一个系统包括：

- 4个控制板开关：输入0到3
- 4个指示灯或者可见的输出模块LED：输出A到D
- 2个弹簧复位阀门控制气缸：输出E和F
- 1个锁销阀门控制气缸：输出G和H
- 3个传感器来检测每个气缸的伸展

（1）清除你的PLC中的程序存储器，然后把PLC置于运行模式，打开数据屏幕

1）使用数据监控屏幕来显示反映所有输入模块的状态的输入映像内容（如果使用PLC-5，模拟输入在它们配置前不可以监控）。观测数据屏幕时，改变一些开关和传感器位置，改变显示的数据格式。

2）PLC仍然在运行模式时，通过使用编程器的写数据能力，试图改变输入图像位。回答：为什么你不能改变它们？

3）显示输出模块的输出映像内容（PLC-5模拟输出模块必须首先配置）。当观察执行器时，使用数据写特性来改变在数字输出模块的输出，并使用电压测试来检测模块的输出（当前输出，测试通过电阻后的电压降）。

（2）写一个程序，包括定时器、计数器和一个监控数据位的梯级来控制气缸E的输出位。

1）程序运行时，使用数据监视器来监控在定时器、计数器和数据存储器中数据值。检查来自编程器的哪个值将被改变。

2）显示和改变你的PLC提供的每个其他类型的数据存储器的内容。在监控数据内容前，可以创建数据元素（在Allen-Bradley PLC）或者数据块（在Siemens PLC）。

3）写一个程序移动来自控制开关的数据字或者字节到输出灯和一个数据字中，但是如果气缸E伸展，改变开关的位置，通过写数据来改变控制气缸E的数据位状态。

（3）写一个带定时器的程序

1）包括一个梯级，复制定时器累加器的值到一组数据值中的第三个数据字。使用带有偏移量目的地址的变址寻址。

2）包括一个梯级，复制一个定时器的累加值到存储数据字的第15个可用存储器。对目的地址使用间接寻址。

第6章　操作简单数据元素

6.1　学习目标

本章您将了解到：

1）微处理器的计数器、状态字以及在数据操作中的其他寄存器的使用；

2）Allen-Bradley、Siemens 和 OMRON 指令（或者功能块），用于：

- 移动单个的数据单元（字节，字，双字或者部分字）；
- 在布尔逻辑语句中比较单个的数据单元；
- 对单个的数据单元进行数学运算；
- 对单个的数据单元进行逻辑运算；
- 转换数据类型；

3）在数据操作指令中的数据的间接和变址寻址。

如果你正在使用 Siemens 的 PLC，你应该在读第 6、7 章前先读第 8 章。

对工业过程控制常常要求数据操作。两个计数器的累计值的相加这样的简单操作，可能是把在两条装配线上的良好的零件数量相加起来这样的工作所要求的。如果你想 PLC 执行统计的过程控制操作，希望以此计算是否一个生产过程在规程内仍然安全的话，那么数学操作需要变得更加复杂。现代的 PLC 可以进行任何其他计算机可以执行的相同的数学运算，但是程序员必须小心保证要执行的数据操作的总量不会占有太多时间，否则它将会干扰 PLC 的主要工作：快速采样和执行控制生产过程的程序。

理解 PLC 怎样操作数据的关键是，记住 PLC 只是有特殊用途的计算机。它使用任何其他的计算机都使用的微处理器芯片，但是它的 ROM 存储区包含一些程序，这些程序让它具有与其他类型的计算机不同的性能。

6.2　微处理器基础

微处理器有时叫做微处理器单元（MPU）。它是一个中央处理器（CPU）的核心组成部分。PLC 一般使用其他计算机同样使用的 MPU 芯片（也就是，当 MPU 用于 IBM 兼容的计算机中，它们常常是由 Intel 公司制造；当 MPU 用于苹果计算机和 PowerPC，它们由摩托罗拉公司制造）。PLC 制造商通常不会标出他们使用的 MPU。

一些 PLC（例如 Allen-Bradley）可以轻易地由并不了解 MPU 怎样工作的用户进行编程。其他的 PLC 生产商（例如 Siemens 和 OMRON）相信只有用户完全了解 MPU 的操作后，PLC 才能在用户使用时发挥它的最大潜力。即使你打算使用 Allen-Bradley 的 PLC，了解一点关于微处理器的操作对你来说也是有益的，因为 Allen-Bradley 的 PLC 正在努力使熟

悉 MPU 的用户能够写出更好的控制程序成为可能。

微处理器包含几个类型的寄存器，这些寄存器保存它每一次执行一步程序所要使用的信息。这些寄存器包括：

(1) 累加器

累加器保存 MPU 处理的数据字。典型的 PLC 编程语言允许直接使用一个或两个累加器。累计器可以保存的位的数目规定了 PLC 一次可以操作的最大容量的数据元素。Allen-Bradley 的 PLC-5 和 SLC 500 的 PLC 有 16 位累加器，Siemens S5 PLC 和 OMRON CQM1 也一样。Siemens S7 系列的 PLC 有 32 位累加器。

(2) 状态寄存器

状态寄存器包含受微处理器操作的数据影响的状态位。一些 PLC 数据操作指令能自动地复制状态寄存器位到可寻址存储器，所以用户程序可以检测它们。典型的 PLC 程序员可以使用的状态位包括：

1) 零状态位，如果最近操作的数字为（或者变为）零时，这个位就会被置位。

2) 符号位，如果最近操作的数字在它的最高位（MSB）有一个“1”时，这个位就会被置位。有符号的二进制数中在 MSB 位的“1”代表一个负数。

3) 进位位，如果最近的数学运算生成的二进制数超过了微处理器可以容纳的位数，这个位就会被置位（所以结果是一个不正确的答案）。

4) 溢出位，如果在增加一个数的值时导致它的最高位变为“1”，或者在减少一个数的值时导致它的最高位变为“0”，这个位将会被置位。这会指明一个无意的符号改变，例如，当 16 位有符号的二进制数 3 2767 加上 1 时，结果将会是－32 768。

5) 逻辑运算的结果（RLO）位，PLC 的复杂的布尔逻辑指令会使用这个位（1 为真值，0 为假值）。在 S7 的 PLC 中，程序可以访问 RLO 状态位。S7 也允许程序员对二进制结果（BR）状态位进行寻址，这个状态位也叫做 ENO 位，能够指明一个子程序的执行是成功（1）还是失败（0）。OMRON 的 CQM1 在它的真值寄存器中保存 RLO 位。

(3) 地址寄存器

地址寄存器有时也叫做偏移寄存器由 MPU 使用，这样可以记住当前使用的存储区各个段的首地址，并且也可以记住用户程序存储区和各个寻址数据存储区的首地址。越来越强大和复杂的存储区访问技术需要更多的地址寄存器。

6.3 数据操作指令

虽然大多数存储空间使用布尔指令可以每一次操作一个位，现代 PLC 提供了更强大的指令系统。这些更加强大的指令允许程序操作整个多位数据元素，就像单元素的数据类型。（例如，字节、字符、16 位数据字、浮点指针数、时间值等等）。在本章，我们会介绍可以编程对 PLC 作些什么，但是我们在细节的解释方面只提供一些典型数据操作指令的例子。如果你想了解更多关于 PLC 指令系统的详细描述，可以参阅你的编程手册。

当我们检查数据操作指令时，记住 PLC 通过重复地执行三步扫描循环来操作。记住这个特性后，你会更容易理解数据操作指令在存储器和在 PLC 控制的过程的作用。三步扫描循环如下：

1）把输入值从输入模块读入到输入映像存储器。

2）在执行用户程序时，从开始到结束都是一次执行一条指令。

3）从存储器的输出映像区域复制数据到输出模块。

6.4 简单数据元素

字节、字和双字都是单独的简单数据元素[⊖]。保存为“简单”数据元素的二进制数可以表示有符号或无符号的整数值、实数、定时器/计数器的累加值，或者也可以是用作字符的专用编码的数据单元，例如计时器常数、间接寻址的指针，或者是其他的用途。一个多位的数据元素只表示一组数字传感器或者数字执行器的开/关状态。无论一个二进制数据元素代表什么，对于 PLC 来说它都仅仅是一个二进制数，并且可以通过以下描述的指令来操作。

6.4.1 简单数据元素的移动

我们已经看过一些用于移动单个数据值的 PLC 指令的例子；在第 5 章中，它们用于显示数据元素的寻址。例如，移动指令可以用于把一个 16 位的值从工作数据存储器复制到一个输出映像数据字，从而用单一的指令来控制 16 个数字执行器的状态。

一些移动指令是有条件的，这意味着它们只有在移动指令前面的布尔逻辑为真时，它们才会执行。在其他的 PLC，移动指令被执行是无条件的，这意味着在对移动指令编程时，即使控制逻辑为假，PLC 也总是执行移动。

大体上，移动指令把数据从一个存储位置复制到另一个大小相同的存储位置。作为一台计算机，PLC 实际上必须对一个数据值复制两次来移动它，第一次是从它的源地址复制到微处理器的累加寄存器，第二次是从累加寄存器复制到目的地址。梯形图指令使这两步的操作看起来像一步完成的，但是一些 PLC 编程语言（例如在 IEC1131-3 标准中描述的指令表（IL）语言和 Siemens 的 STL 语言。）要求程序员对两个复制操作分别地编程。这些语言也允许程序员在把数据值复制到微处理器的累加器之后，在把数据值复制到目的地址之前，对数据值进行运算。运算可以改变在后面的地段所覆盖的数据值。

几个 PLC 提供梯形图移动指令，仅移动二进制数据的一部分，并把其他部分留在不受移动所影响的目的地址。一些 PLC 也提供一种移动指令，在移动过程中，把数据值从一种数据类型转变为另一种数据类型。

在第 7 章中，我们会讨论更多有用的指令，这些指令能操作整个的简单数据元素集。

1. ALLEN-BRADLEY 的移动指令

PLC-5 SLC 500

在 Allen-Bradley PLC5 和 SLC 500 中，移动（MOV）指令（如图 6-1 所示）可以用于把源地址的值（一个常数，或者包含一个 16 位数据字的存储区的内容，或者一个 32 位的浮点指针元素）复制到目的地址。当从浮点数元素移动一个 16 位数字，或把一个 16 位字移动到一个浮点数元素时，MOV 指令能使数字在有符号 16 位二进制数和 32 位浮点数的格式之间相互转换。浮点数的十进制部分是整数值。图 6-1 显示的 MOV 指令

⊖ 位也是简单数据元素，位操作已经在第 3 章讲述。

可以编程为无条件地执行，但是如果布尔逻辑在 MOV 指令的左边编程，可以使 MOV 指令成为有条件的。

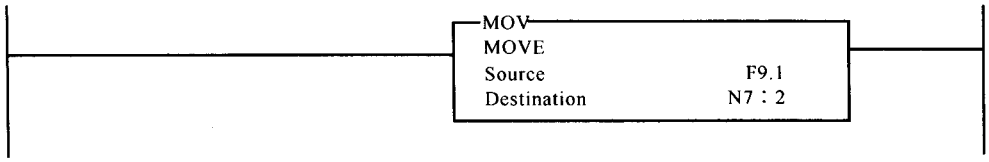


图 6-1 Allen-Bradley 的移动指令

Allen-Bradley 的两种 PLC 都提供了*带掩模的移动*（MVM）指令，这个指令能够用于把一个 16 位数字的一部分复制到目的地址，而且不需要改变目的地址其他的内容。*掩模*必须指定为一个 16 进制的常数（如图 6-2）或者是一个可以取回 16 位掩模值的地址。如果来自源地址的位要覆盖对应目的地址的位，16 位掩模对应的位必须为 1。在例子中，16 进制掩模 00FF 的低 8 位都为 1，所以只有目的地址的低 8 位内容会被源地址的低 8 位覆盖。

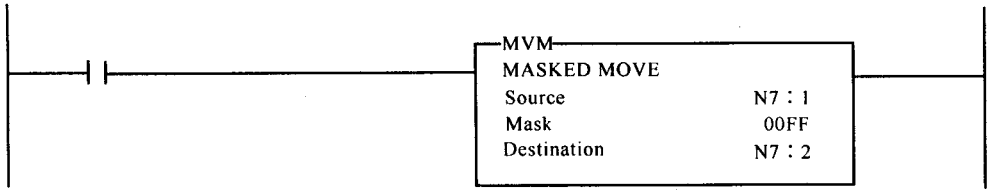


图 6-2 Allen-Bradley 的带掩模的移动指令

PLC-5 也有*位段落传递*（BTD）指令，这个指令允许程序员把一连串的连续的位从 16 位数据源地址复制到 16 位数据目的地址的任何地方。程序员指定要移动多少连续的位、源地址中的起始位序号以及目的地址中的起始位序号。

PLC-5 和 SLC 500 PLC 都有一些状态位⊖，受到 MOV、MVM 或者 BTD 指令后的目的地址的结果数字的影响。

SS

2. SIEMENS STEP 5 *的移动指令（加载和传送）*

在学习 Siemens S5 的数据操作能力前，你应该知道怎样从其他程序中调用 STEP 5 程序，而且你应该知道功能块（FB）程序。一些指令只可以在功能块中编程，一些地址只可以由功能块中的程序访问。（在开始这一章前，你应该先读完第 8 章。）

STEP5 的用户程序可以使用梯形图或者 STL 语言编写。STL 跟 PLC 微处理器实际使用的机器语言相似。所有的梯形图指令都有 STL 的等价指令，通常每个梯形图指令由几个 STL 语句组成。事实上，梯形图指令在被送到 PLC 的存储区之前，总是由编程器转换为 STL 的同等指令。一些 STL 指令执行的操作不能在梯形图中编程。

本节描述的 STL 的*加载*（L）和*传送*（T）指令和它们变种都是无条件执行的！任何时候，当 PLC 在程序中遇到它们时，它们就会被执行，而不用理会逻辑运算的结果（RLO）

⊖ S：0/2（零位）或者 S：0/3（符号位）可以被置位，这依赖于移动后目的地址的值。如果浮点数太大而不能转换为整数，S：0/1（溢出位）可以被置位。S：0/0（进位位）可以被 MOVE 指令清除。

的状态。如果你需要使加载和传送指令有条件地执行，你必须使用结构化的编程技术。你可以在功能块中对有条件的梯形图 MOVE 指令进行编程，但是编程器会自动地把它转换成一个结构化的包含无条件加载和传送指令的 STL 程序。

如图 6-3 的程序所示，STEP 5 的梯形图移动指令等价于 STL 的传送指令后面再加一条 STL 的加载指令，但在功能块中除外。正在加载的数字可以来自一个地址（例如来自图 6-3 的 IW2），或者是一个常数值。STL 的例子也显示了正在被操作的数据值，在程序被操作时，这些数据值会在编程器的监视器中显示（以十六进制）。加载指令复制一个数字（“B500”）到 MPU 的累加器 1，把累加器 1 以前的内容（以“xxxx”显示）推到累加器 2。累加器 2 以前的内容（以“YYYY”显示）会丢失。然后传送指令把数据从累加器 1 复制到指令（FW4）指定的存储区地址，而不会影响累加器 1 或累加器 2 的内容。（这个例子包含一个潜在的缺点，我们在后面会讨论。）



In STL：

	ACCUM 1	ACCUM 2
:	xxxx	yyyy
: L IW2	B500	xxxx
: T FW4	B500	xxxx

图 6-3 STEP5 的梯形图 MOVE 指令，对应的 STL 指令显示程序执行时累加器的内容

加载和传送指令（或者移动指令）必须包含被读或写的存储位置的地址，或者被加载的常数值。记住在 S7 PLC 的大多数可寻址存储位置只包含一个字节（8 位）。如果加载或者传送指令对一个字节使用一个地址，这个值会在那个地址和累加器 1 的低字节之间复制。另一方面，如果加载或者传送指令为一个字（16 位）使用一个地址，然后有 8 个位就会在 16 位的累加器的高字节和在指令中指定的数字的地址之间复制，并且另外 8 个位会在累加器的低字节和下一个存储器位置的位置之间复制（如果有的话）。

图 6-3 中的程序在一个中型的 PLC（例如 S5-100U）中运行。它使用字寻址，所以加载指令从 IB2 取得 8 位的累加器的高字节的值（16 进制：B5），因为 Siemens 的 S5-100 数字输入模块只可以提供 8 位数据，所以累加器的低位会设为零。然后传送指令会把累加器的高字节复制到 8 位的存储器空间 FY4，并从累加器的低 8 位的零复制到 FY5，覆盖那里的任何有用数据。显然这是一个有潜在的错误程序。

如图 6-4 所示，一个更好的程序应该对 8 位存储器按字节寻址。这个程序把 8 位输入映像值（B5）从 IB2 加载到累加器的低字节并清除高位（高位在加载字节时总是会被清除），然后把低字节从累加器 1 传送到 FY4，并且 FY5 会不受影响。

出错的程序会更糟糕。字节和字会混合在一起。在图 6-5a 中，输入映像数据按字来加载，所以它进入累加器的高字节。传送指令使用字节寻址，所以累加器为零的低位会写到 FY4，有效的数据在任何地方都不会被复制。在图 6-5b 中，输入映像字节按字节来加载，

所以它进入累加器的低 8 位，然后它被作为一个字来传送。清除了的 8 个高位会被传送到 FY4，有效的数据会被送到错误的地址，即 FY5 中。

```
                    ACCUM 1      ACCUM 2
:                   xxxx         yyyy
: L   IB2           00B5         xxxx
: T   FY4           00B5         xxxx
```

存储器	FY4	FY5
程序执行前	aa	bb
程序执行后	B5	bb

图 6-4 改进后的程序，用于操作字节

```
                    ACCUM 1      ACCUM 2
:                   xxxx         yyyy
: L   IW2           B500         xxxx
: T   FY4           B500         xxxx
```

存储器	FY4	FY5
程序执行前	aa	bb
程序执行后	00	bb

a)

```
                    ACCUM 1      ACCUM 2
:                   xxxx         yyyy
: L   IB2           00B5         xxxx
: T   FW4           00B5         xxxx
```

存储器	FY4	FY5
程序执行前	aa	bb
程序执行后	00	B5

b)

图 6-5 Siemens 加载并传送混合了字和字节寻址的指令

如果加载或传送指令在数据块中对数据寻址，STL 程序必须包括 STL 指令的第三个形式：如图 6-6 所示，调用数据块（C）指令必须在加载和/或传送指令之前。在梯形图中，数据块的调用包括在梯形图指令中，所以地址必须指定数据块和数据字。源地址为 DB4：DW5的梯形图移动指令等价于调用数据块 4，然后加载数据字 5 的 STL 指令。

在 STL 程序中加载 BCD（LD）指令可以代替加载指令，但是只能在指令从定时器或计数器存储器加载一个值时。这个指令使 PLC 把在定时器或计数器的地址的低 10 位的自然二进制数值以 BCD 码传送到累加器的低 12 位（并清除最高位 4 位）。MOVE BCD 梯形图指令

包含一个加载 BCD 指令，然后是一个传送指令。

交换累加器内容（TAK）指令能交换在累加器 1 和累加器 2 的数据值。它只可以在 STL 中编程，并且只能在功能块中编程。

```
C DB4
L DW5
C DB6
T DW7
```

图 6-6 在两个 Siemens 数据块中移动数据

只要加载和传送指令（还有移动指令）在功能块中编程，它们就可以读/写存储器的系统区域（RSn）。在小型和中型的 S5 PLC 中，使用外围存储器（PWn，PYn）的直接寻址的加载和传送指令只允许出现在中断程序里。（OB2 或者 OB3；见第 11 章）。

3. 在 STEP 5 中使用变址和间接寻址移动数据

下面指令的寻址技术已经在第 5 章中的 STEP 5 变址寻址节中详细地描述了。

变址寻址可以用于标准的 STL 语言的加载和传送指令，如果此时加载和传送指令跟随一个处理操作（DO）指令，这个指令指定一个位移值，又或者如果加载或传送指令必须指定数据将要加载的或传送的存储区域。

专用的加载和传送指令可以使用 S5 PLC 的间接寻址。一个 16 位的指针值必须加载到累加器 1；然后间接加载寄存器（LIR）指令可以用于从指针指向的地址加载一个数字到累加器 1 或累加器 2，或者间接传送寄存器（TIR）指令可以用于从累加器传送一个数字到指针指向的地址。这些指令只可以在 STL 中编程，并且只能在功能块中编程。

只有在功能块程序中，并且一串可能的地址被传送到功能块作为参数时，STL 的加载和传送指令也可以使用数据的间接寻址，后面跟随一个间接处理（DI）指令。

4. SIEMENS STEP 7 的移动指令（加载和传送）

对 STEP 7 的结构化编程有一个基本的理解对理解 STEP7 的移动指令是很有帮助的。STEP 7 的结构化用户程序由一个或更多的主程序组成，叫做组织块（OB），包括调用功能块（FB）程序的指令或者更简单的函数（FC）程序。（在阅读这一章之前，你应该先阅读第 8 章。）

梯形图的移动指令由 STL 语言的加载指令（复制一个数到累加器 1）和传送指令（把数从累加器 1 复制到存储器位置）组成。除非结构化程序使 PLC 回避加载和传送指令，否则这两条指令会无条件地执行。

Siemens STEP 7 编程语言没有对移动指令，以及 STL 的加载和传送指令做出重大的改变。前面介绍的 S5 系列 PLC 的 STEP5 指令也可以同样应用于 S7 系列 PLC 的 STEP7，以下情况除外：

（1）S7 PLC 有 32 位的累加器，它与 S5 的 16 位累加器有些不同。图 6-7 显示了带有三种不同源地址大小的加载指令：

1）按字节寻址存储器，使 PLC 在那个地址和累加器 1 的低 8 位之间复制 8 位的数据值。加载一个字节值会使累加器的高 24 位被清除。传送一个字节会把累加器的最低位写到存储器的指定字节。

2）在加载或传送指令中，按字寻址存储器使 PLC 在指定的地址及下一个地址，和累加

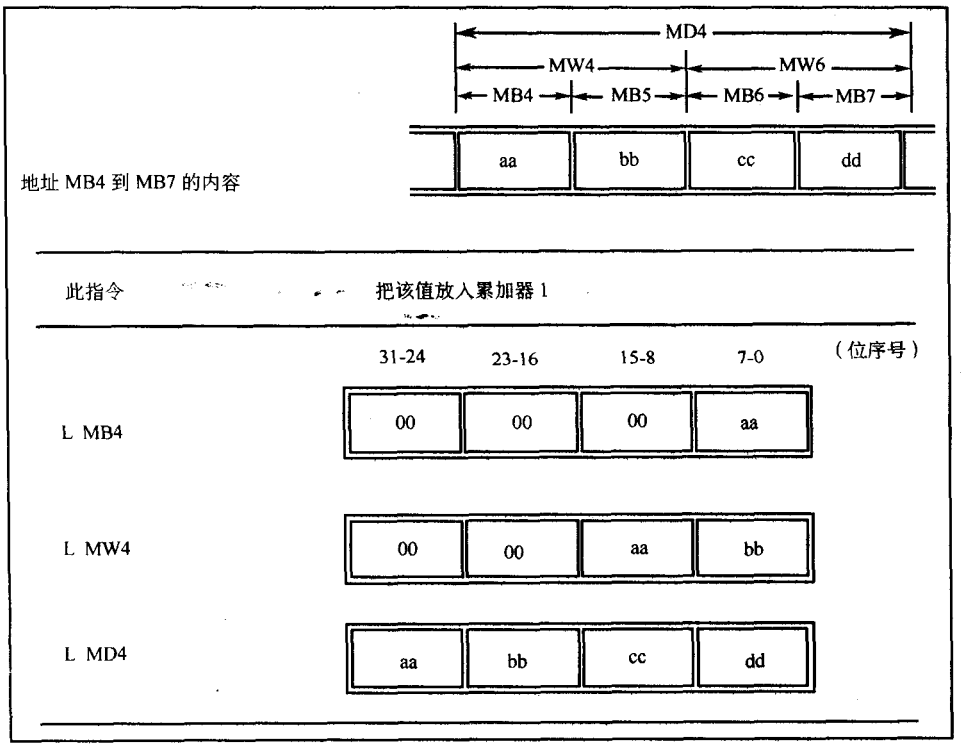


图 6-7 按字节、字或双字来寻址 S7 的效果

器最低的 16 位之间复制数据的两个字节。加载一个字的值使累加器 1 的高 16 位被清除。

3) 按双字寻址存储器，使 PLC 在指定的地址及接着的三个地址，和 32 位的累加器之间复制一个 32 位的数据值。

(2) 除了 STEP 5 的交换累加器内容指令（在 STEP7 中叫做 TOGGLE ACCUMULATOR）外，还有四个另外的指令，只能在累加器之间移动数据。Push（PUSH）把累加器 1 的内容复制到累加器 2，而不改变累加器 1。Pop（POP）把累加器 2 的内容复制到累加器 1，而不改变累加器 2。有两个交换字节顺序指令：CAW 交换累加器 1 中低字的两个字节，CAD 使累加器 1 的双字的四个字节的顺序颠倒。(在 CAD 执行后，最低位的字节与最高位字节交换，第二最低位字节与第二最高位字节交换)

(3) STL 指令的加载 BCD，可以加载定时器或者计数器的值并在这个过程中把它转换成 BCD 码，这个指令现在叫 LC，而不是 LD。

(4) 存储器可以使用符号名来寻址，另外间接寻址方法现在叫作存储器间接寻址和寄存器间接寻址（在第 5 章中解释），和直接（绝对）寻址的格式一样。STEP5 初步的 LIR 和 TIR 指令不再使用。

(5) 其他的 STL 指令可以把 32 位的指针值加载到 MPU 的地址寄存器，或者能从地址寄存器传送指针值。LAR1 或者 LAR2 用于把一个指针加载到地址寄存器 1（AR1）或者地址寄存器 2（AR2）。指针可以在 LAR 指令中作为常数来输入，如果 32 位指针在存储器中，那么 LAR 指令会包括一个地址规定，如果来自 AR2 的指针要加载到地址寄存器 1，那么参数“AR2”会与 LAR1 指令一起使用。除非指令指定另一个源，否则 LAR1 和 LAR2 会使

用累加器 1 作为指针的源。

TAR1 或者 TAR2 指令从 AR1 或 AR2 复制一个指针到指定的存储位置，如果 TAR1 或 TAR2 指令不包括地址，这个指针就会被复制到累加器 1。CAR 指令交换 AR1 和 AR2 的指针。

在本章这一节后面的 Siemens 的 STEP 7 的数学/逻辑指令中，我们会检查＋AR1 和＋AR2指令，这两个指令可以用于增加或者减少 AR1 和 AR2 的指针值。

(6) 当前激活的数据块的数和大小都可以读，并可以加载到累加器 1。

L DBNO 加载共享数据块序号
L DBLG 加载共享数据块大小（以字节）
L DINO 加载立即数据块序号
L DILG 加载立即数据块大小（以字节）

5. OMRON CQM1 的移动指令

CQM1

几个 CQM1 的编程指令都可以用于从一个存储位置（或者从一个常数）复制数据到另一个存储位置，但是只有其中三个能够复制单一的数据值而不改变数据的值。在数据被移动时改变数据的指令我们将在下一章讨论。

在图 6-8 中，只要控制逻辑为真，移动指令 MOV(21)就能够把一个源数据字复制到目的地址。源作为第一个参数输入移动指令，而且可以是任意 16 位常数或者任意地址，详细指明数字将从哪里复制（在图 6-8 中，使用常数＃1234。）第二个参数指明复制到的目的地，可以是任意地址，但是存储器的 TC 区域除外（在图 6-8 中，一个输出映像字地址被指定）。

MOV(21)指令可以使用间接寻址，在第 5 章 OMRON CQM1 的间接寻址小节介绍过。

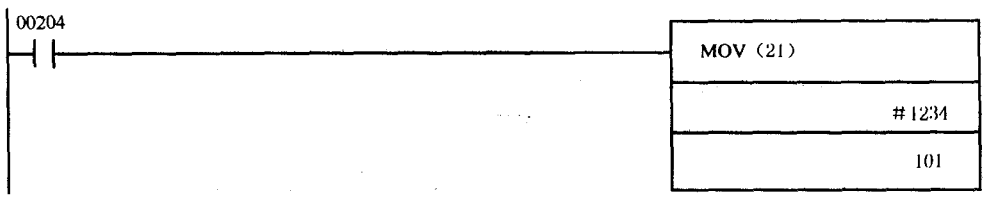


图 6-8 CQM1 的 MOVE 指令

其他两种移动指令可以完成 MOV(21)指令的工作，但是在移动一个 16 位数据字时，它们允许使用变址寻址来识别源地址和目的地址。单字传送指令 DIST(80)，可以复制一个 16 位数据字到变址的目的地址；数据收集指令 COLL(81)，可以从变址的源地址复制一个数据字。在第 5 章 OMRON CQM1 变址寻址小节中，我们描述这些指令怎样使用变址寻址。至于这两个指令怎样用于超过一个的字移动，我们将会在第 7 章讲述。

6.4.2 简单数据元素的比较

大多数 PLC 都提供可以比较两个简单的数据值或比较数据值和常数的指令。比较两个数据值通常是用来了解它们是否大小相等或者是一个比另一个大。比较指令的结果通常是真或假，并且可以用于控制另一个 PLC 指令的布尔逻辑语句中。例如，反映不同传送带上有多少零件的两个计数器的累加值，可以通过比较来使结果能用于控制一个输出映像位，这样

执行器就能把零件推到有最少零件的传送带上。

PLC-5 SLC 500

1. ALLEN-BRADLEY 的比较指令

Allen-Bradley 的比较指令产生布尔结果。它们可以作为梯级的逻辑语句控制梯级上的输出指令。一个阶梯可以包含几个比较指令，并允许用复杂的比较要求来构造布尔逻辑语句。

Allen-Bradley 的比较指令允许 16 位的有符号的二进制数和/或存储器的浮点数互相比较，或者与常数比较。常数可以是十进制，或者带前缀的 16 进制、二进制或者八进制（分别为 &H、&B 或者 &O）。

在 CMP 指令中使用浮点数时，计算机化的数学运算经常会出现舍入的错误，例如一个很简单的表达式（F9：1=0）并不一定总是真。

2. PLC-5 的比较指令

PLC-5 允许程序员写他们自己的逻辑语句。表达式写成比较（CMP）指令形式。PLC-5 最近的增强型系列甚至允许逻辑表达式包括括号，以及在比较运算之前还有进行字处理的数学表达式。（附录 D 包括一个可用算子的完全列表。留意字符“|”用于分隔。）比较表达式的括号不能用于建立复杂的比较条件。每个 CMP 指令只能进行一次比较运算。

图 6-9 演示典型的 CMP 指令是怎样使用的。注意表达式首先计算括号内的数学运算（N7：1除以 4）以及字逻辑运算（把 N7：1和 N7：3中的 16 位字节进行与运算），之后再将这两个结果进行逻辑比较。如果 16 位有符号整数的除运算结果大于字的与运算的 16 位有符号整数值结果，那么 CMP 指令的结果是布尔真。在这个例子中的 CMP 指令是一个复杂的布尔逻辑语句的一部分（它的结果会与检查位 B3/5 的结果相与），并且最后的布尔逻辑结果控制一个输出单元（移动指令）。

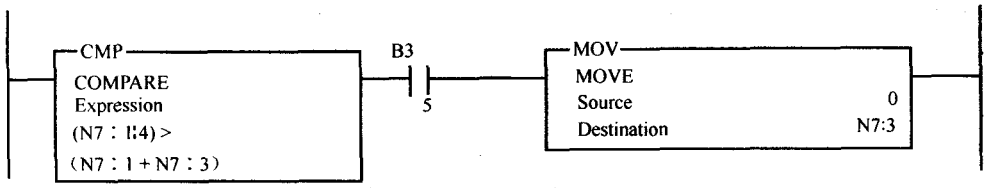


图 6-9 PLC-5 的比较指令，作为控制移动语句的一部分

COMPARE 表达式可以包括与梯形图执行相同的逻辑运算的布尔逻辑语句。例如可以使用以下算子：

> 大于

= 等于

布尔逻辑算子不能与处理字逻辑的算子结合使用。如下算子：

#not# 对一个字中的所有位执行 NOT 操作

#AND# 对两个字中的所有位执行 AND 操作

不能用在比较表达式中。

比较指令比下面介绍的更简单的比较指令需要明显更长的执行时间。

3. 其他的 Allen-Bradley 比较指令

SLC 500 PLC 不提供比较指令。它们要求程序员使用老的、简单的比较指令，只能比

较两个源输入值。算术和字逻辑的运算在这些比较指令中不能实现。PLC-5 也允许简单的比较指令的使用。

简单的比较指令，用法如图 6-10 所示，包括 6 个简单的比较：

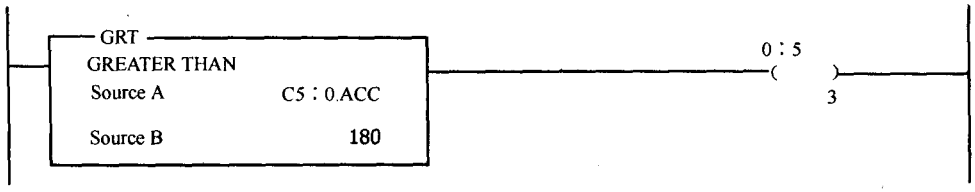


图 6-10 Allen-Bradley 比较指令，显示控制一个输出激励指令。

- 等于（EQU）：如果两个源的值相等则为真。
- 不等于（NEQ）：如果两个源的值不相等则为真。
- 小于（LES）：如果源 A 比源 B 小则为真。
- 小于或等于（LEQ）：如果源 A 小于或者等于源 B 则为真。
- 大于（GRT）：如果源 A 比源 B 大则为真。
- 大于或等于（GEQ）：如果源 A 大于或者等于源 B 则为真。

还有两个功能更强大的指令：

- 极限测试（LIM）：如果测试值输入的范围是在“上限”和“下限”之间，则为真。
- 带掩模的等于比较（MEQ）：如果源的 16 位数字选择的位与要比较的 16 位数字的相同位相等则为真。一个 16 位的掩模指明要比较哪一位。掩模以十六进制常数或内存地址的形式输入，掩模中为 1 的位会被比较。

4. SIEMENS STEP 5 的比较指令

Siemens 的比较指令比较累加器 2 和累加器 1 的值，把这两个值都看成是 16 位的有符号二进制数。加载指令把一个值放到累加器 1，并移动累加器 1 原来的值到累加器 2，所以一对加载指令会填满两个累加器。

STEP 5 比较指令的结果或者为真或者为假，而且比较指令可以用的地方，就是其他的布尔逻辑输入指令在条件语句中用来影响 RLO 的任何地方（影响是否执行一个条件输出指令）。比较指令也影响 MPU 的状态寄存器中的两个条件代码位（CC0 和 CC1）。对于功能更强大的编程，条件跳转指令可以跟随比较指令用于检查 CC0 和 CC1 位。（有条件的跳转会在第 8 章讨论。）

图 6-11 显示一个条件语句（梯形图和 STL），这个语句加载两个值并比较是否第一个值（来自 IW64）比第二个值大（来自数据块 4 的数据字 3），如果是则打开输出。

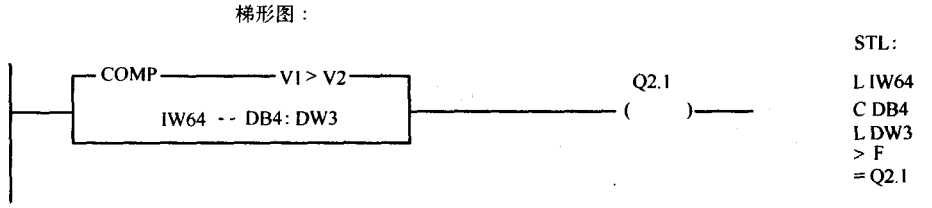


图 6-11 在一个条件语句中的 STEP5 比较指令

可以在梯形图或 STL 中比较是否第一个加载的值是：

！=F　　等于第二个加载的值

<>F　　与第二个加载的值不相等

>F　　大于第二个加载的值

>=F　　大于或者等于第二个加载的值

<=F　　小于或等于第二个加载的值

<=F　　小于或者等于第一个加载的值

在上面列出的 STL 比较指令中的“F”暗示程序员 PLC 会把累加器中的 16 位数看作代表（有符号的）定点整数值的有符号二进制数。

S7

5. SIEMENS STEP 7 的比较指令

STEP7 提供与 STEP 5 相同的比较算子。STEP 7 改变的包括：

1）“等于”算子从“！=”改变为“==”，“不等于”变为“<”。

2）在比较指令中用符号“I”代替“F”，可以比较 16 位的有符号定点整数（例如，==I是比较在累加器 1 和累加器 2 中的两个 16 位有符号二进制数的大小是否相等）。

3）双字和浮点数可以比较。如果比较双字（表示有符号整数的有符号 32 位二进制数），那么“D”会取代“I”（例如，==D）。如果浮点数（表示实数）被比较，那么“R”会取代“I”（例如，>R）。当心：浮点数系统中的突发事件有时会使应该相等的浮点数变得不相等。最好比较判别一个浮点数是远大于还是远小于另一个浮点数（或者一个常数），而不只是比较它们是否相等。

4）如图 6-12 所示，梯形图指令的出现作了轻微的改变，来遵守 IEC-1131-3（我们会在第 9 章介绍）。CMP 指令可以连接到布尔逻辑的梯级，通过在模块左边和右边的最高的连接线。将要被比较的地址或常数值可以从模外输入到 IN1 和 IN2 的左边。IN1 表示将会放到累加器 2 的值，并且 IN1 的值会在比较执行之前进入累加器 1。在图 6-12 中，如果 IN1 比 IN2 大，语句的结果为真。

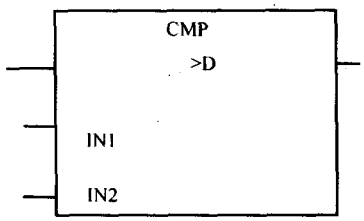


图 6-12　STEP7 的梯形图比较指令

CQM1

6. OMRON CQM1 的比较指令

与本书讲到的其他 PLC 编程语言的比较指令不同的是，CQM1 的比较指令 CMP(20)，被认为是输出指令。它必须是梯级上最右边的指令，并且只有在它左边的布尔控制逻辑条件为真时才会被执行。如果执行了，它会影响存储器的 SR 区域的位。它不会产生在布尔逻辑语句中直接使用的真的或假的结果；必须在比较之后检查它影响的 SR 的存储器位。

如图 6-13 所示，每次 CMP(20)指令执行时，这个指令会比较第一比较值（Cp1，第一个参数）和第二比较值（Cp2，第二个参数）。在指令中指定的两个比较的值可以是存储器

位置的地址，或者是作为比较指令的一部分而输入的常数。数值总是被看成无符号的二进制数，除非这个值是一个定时器或者计数器的当前值，在这两种情况下，数值都会被认为是BCD的格式。当CMP(20)指令执行时，它会打开下面的三个状态标志位（并且把其他的关掉）。在存储器的状态寄存器（SR）区域的这三个状态标志位包括：

1）25505（GR标志位）：如果第一比较值＞第二比较值，开启

2）25506（EQ标志位）：如果第一比较值＝第二比较值，开启

3）25507（LE标志位）：如果第一比较值＜第二比较值，开启

如图6-13所示，接着的指令可以检查状态位，在此处只要计数器8的当前值大于5，输出映像位10300会被开启。注意在这个例子中，如果使CMP(20)指令执行的逻辑为真，输出10300可以继续。这个构造阻止其他的可能影响SR22505的指令意外地使输出10300启动。

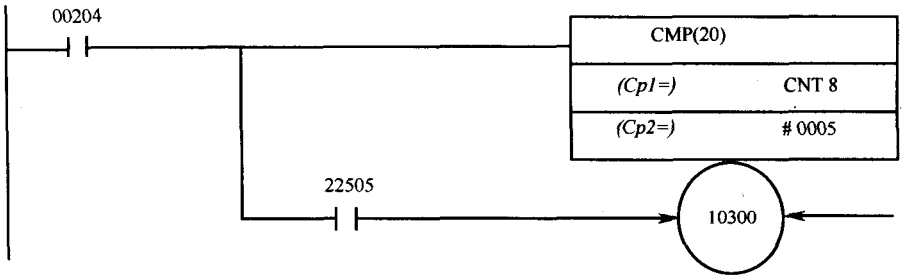


图6-13 CQM1的比较指令有条件地执行，并且有一个使用这个结果的分支

CMP1提供一些另外的用于其他类型二进制数的指令。像CMP(20)指令，它们都被认为是在布尔逻辑控制语句为真时执行的输出指令，并且它们都影响GR、EQ或者LE标志位。

1）双比较指令CMPL(60)，比较两个32位无符号二进制数（或者BCD值）。Cp1和Cp2参数可以是地址，且每个地址都指明32位数的低16位在哪里。高16位来自以Cp1和Cp2输入的地址上面的地址。

2）有符号二进制比较指令CPS(-)，还有双有符号二进制比较指令CPSL(-)，除了16位或32位数字翻译为有符号二进制数之外，与CMP(20)和CMPL(60)指令很相似。OMRON要求程序员输入一个必须为000的第三参数。32位的版本不允许参数以常数输入。

3）区域范围比较指令ZCP(-)，用于设定GR、EQ或者LE的状态位，依赖于被比较数据值是超过/等于/低于较低限制值（LL，第二参数）和较高限制值（UL，第三参数）之间的范围。双区域范围比较指令ZCPL(-)，以相同的方式工作，但比较的是32位数，这个数只能够作为地址说明而输入。

6.4.3 简单数据元素的数学、逻辑和转换操作

PLC提供越来越强大的指令来进行数学运算和逻辑运算，从而改变整个数据字（不像布尔指令只影响单独的数据位）。早期的PLC可以将两个数据值相加，但是减法、乘法、除法这些指令是最近才出现的，而三角函数、对数等现在也可以使用。增加了用于对整个数据实现逻辑AND和OR运算的指令，接着是其他的字逻辑指令。PLC生产商希望能够允许程序员写自己的数学/逻辑方程式，而不用每个运算的独立指令。所以现在一些PLC提供一种

“计算机”类型的指令，这个指令允许程序员输入方程式，从而更好地使用PLC，因为PLC毕竟是一部计算机。

对于不熟悉字一级的布尔逻辑的读者，非、或、与以及异或逻辑运算在这里讨论：

1）非转换二进制数据元素的每个位的状态。例如，NOT指令可以从输出映像表取得一个数据字，对它执行一个NOT运算，然后把结果放回原来的地址。结果会把所有开启的执行器关掉，并把所有关闭的执行器打开：

```
NOT          1111_ 0000_ 0000_ 1010
=            0000_ 1111_ 1111_ 0101
```

2）与运算组合两个二进制数元素来产生第三个二进制元素。一个源值的每个位和其他源值相应的位进行AND运算，并把结果放进相应位。只有当两个相应的源位都是1，结果位才为1（例如，如果源A为真值，源B也为真值，则结果为真）。AND运算常常用于屏蔽一个数据字的一部分。例如，一个源值可能来自输出映像数据字，另一个源可能是由指令提供的常数（叫做掩模），并且结果会返回原来的输出映像地址。程序员可以使用掩模常数，从而强制把输出映像的一些位关掉（掩模位中的0），而不会影响其他位（掩模位中的1）。

```
             1111_ 0000_ 1010_ 0101 (源 A)
AND          0011_ 0011_ 0011_ 0011 (源 B)
=            0011_ 0000_ 0010_ 0001 (结果)
```

3）或运算使两个二进制数据元素结合。如果在源A或源B的相同的位上有一个1，则在这个位的位置的结果数据值为1。（如果在源A或源B的相同的位上都为1，则结果仍然为1）。OR运算可以用于屏蔽运算，从而强制一些结果位打开（在此处掩模常数有一个1），而不改变源的其他位的值（在此处掩模位为0）。

```
             1111_ 0000_ 1010_ 0101 (源 A)
OR           0011_ 0011_ 0011_ 0011 (源 B)
=            1111_ 0011_ 1011_ 0111 (结果)
```

4）异或运算与或运算相似，但是只有在两个源值的同一个位置上只能有其中一个为1时，结果位才为1。异或运算在控制应用中没有AND或者OR的使用程度频繁。

```
             1111_ 0000_ 1010_ 0101 (源 A)
Exclusive OR 0011_ 0011_ 0011_ 0011 (源 B)
=            1100_ 0011_ 1001_ 0110 (结果)
```

PLC-5 SLC 500

1. ALLEN-BRADLEY 数学、逻辑和转换指令

（1）ALLen-Bradley的计算指令

PLC-5和较好的SLC 500提供计算（CPT）指令。在CPT指令中，程序员可以输入一个数学/逻辑表达式，这个表达式包含存储器地址和/或常数（带有适合的前缀）。程序员也必须输入目的地址给浮点数或者整数的结果。CPT表达式可以允许四函数的代数、三角和对数运算，并且可以对整个数据字执行逻辑运算。CPT指令可以使16位的有符号整数和浮点数互相转换，也可以使二进制数和BCD数互相转换，还可以使弧度值和度数值互相转换。如图6-14，CPT指令被认为是输出指令，当它的（可选的）控制逻辑语句为真时，它就会执行。图6-14，展示了一个数学/逻辑表达式，这个表达式使一个常数和N7：1地址中的值进行与运算，然后再把F9.1的值加上3，接着乘以地址F9：1中的值的

结果。最后的结果保存在整数文件 N7：100 中。前缀 E 包括数学/逻辑算子的全部列表和优先的 CPT 指令。

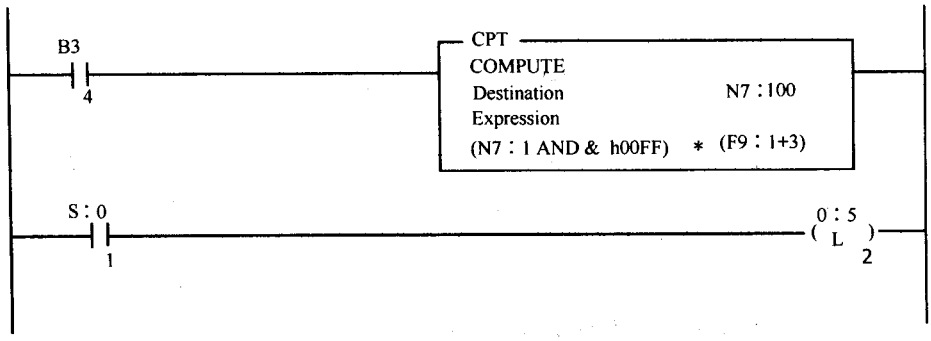


图 6-14 带有 COMPUTE 指令的 Allen-Bradley 程序

CPT 指令的执行影响状态字 0（S：0)㊀的数学标志位。状态字中的三个位（进位、零、符号）只反映在 CPT 指令结束后产生的最后结果的大小。但是一个状态位（溢出）会保持置位，如果表达式中任何运算使它置位。在图 6-14 的例子中，如果由于溢出而产生错误的答案，一个梯级就会把 SLC500 锁定为开。CPT 指令的执行时间明显长于在下面描述的更简单的单操作数学运算。

（2）单操作的 Allen-Bradley 数学和逻辑指令

单操作的数学和逻辑运算指令可以在任何的 PLC-5 或者 SLC 500 程序中编程使用。它们包括基本的加（ADD）、减（SUB）、乘（MUL）和除（DIV）指令。如图 6-15 所示，每个指令都使用两个源操作数（使用地址引用或常数）来产生一个保存在目的地址的存储器的结果，在此处乘法运算会无条件地执行（没有可选的控制条件）。

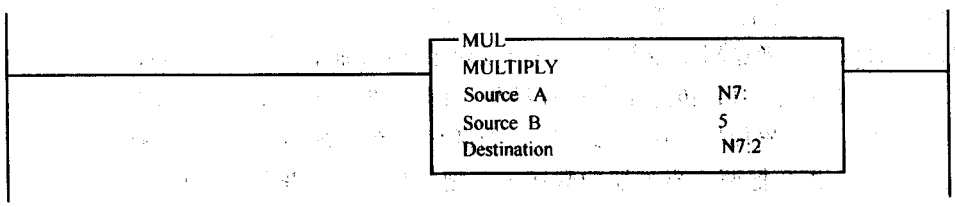


图 6-15 无条件乘指令

PLC-5 和 SLC500 都提供另外的数学指令来清除（CLR）或者取反（NEG）一个数字并且对一个数字执行对数（SQR、LOG、LN、XCP）和三角函数（SIN、COS、TAN、ASN、ATN）运算。两个 PLC 都有非（NOT），与（AND），或（OR）和异或（XOR）等指令，能够对整个数据字进行逻辑运算。

一些 SLC 500 不提供浮点数学运算，所以 SLC 500 有一个特性，能够实现 32 位整数的加法和减法。SLC 500 只可以操作 16 位整数，如果加法或减法导致溢出的话，SLC500 就不会保存目的地址实际 16 位结果。作为代替，SLC 500 常常保存最大或最小的有符号 16 位二进制数（32 767 或者－32 768）。尽管如此，如果数学的溢出选择位（S：2/14）在数学运算的扫描循

㊀ S：0.0＝进位，S：0.1＝溢出，S：0.2＝零，S：0.3＝符号。

环之前被设定，SLC 500 会保存实际的结果作为目的地址的 32 位数学运算的低 16 位。无论何时出现一个数学溢出，不管溢出选择位的状态怎样，SLC 500 都会锁定溢出位（S：0/1）和溢出陷阱位（S：5/0）。对于 32 位加法（或者减法），程序应该包括另一个数学指令，把溢出陷阱位加到另一个代表 32 位数学运算结果的高 16 位值（或者把它从高 16 位的数减去，根据导致溢出的数学指令的类型而定）。溢出陷阱位和溢出位应该在每个 32 位数学运算后解锁。

SLC500 提供附加的双精度除法（DDV）指令，以及求一个数的绝对值（ABS）指令或者对一个数进行标度变换的指令。对一个数进行标度变换的指令要求程序员沿着标度轨迹线（SCP 指令）输入两个指针的坐标值，或者输入一个偏移和比率乘数（SCL 指令）。

PLC-5 提供附加的指令对一串数字分类（SRT）并找出这一串数字的平均值（AVG）和标准差（STD）。这些指令对多个存储器位置的数据进行运算，它们会在第 7 章中说明。

数学和逻辑运算影响 S:0 中的偏移位、溢出位、零位和符号位。

S5

2. SIEMENS STEP 5 的数学、逻辑和转换指令

Siemens S5 PLC 只提供两类 STL 语言数学指令：加法和减法。S5 PLC 没有提供更复杂的数学和逻辑指令，它有预先编好的功能块来执行乘法、除法、测量数字、转换二进制数格式等运算。大多数 STEP 5 的数学和逻辑指令对两个累加器上的值进行运算，把结果保留在累加器 1。梯形图的数学指令包括加载和存储指令，这样梯形图程序员就不需要留意累加器的使用。数学和逻辑运算的数字结果影响状态位，见附录 F。

数学指令只能在功能块中编程。它们无条件地执行，这意味着它们会改变累加器 1 的内容，如果指令遇到 CC0 和 CC1 的状态位，指令会改变这些状态位。结构化编程技术可以使 PLC 绕过这些指令。如果使用梯形图编程，那么带有条件逻辑控制数学指令的梯级会被自动地翻译为等价的 STL 指令。当指令中有“F”，数学指令常常把 16 位值看成有符号二进制数，Siemens 把这些有符号二进制数叫做定点整数。

梯形图的数学或者逻辑指令等价于一个或两个 STL 加载指令，后面接着对应的 STL 数学/逻辑指令，然后是 STL 传送指令保存结果。图 6-16 显示一个减（－F）指令，这个指令能使第一加载值（在累加器 2 中）减去第二加载值（在累加器 1 中），并把结果放到累加器 1 中。加（＋F）指令把两个累加器的数相加，把结果保存在累加器 2 中。

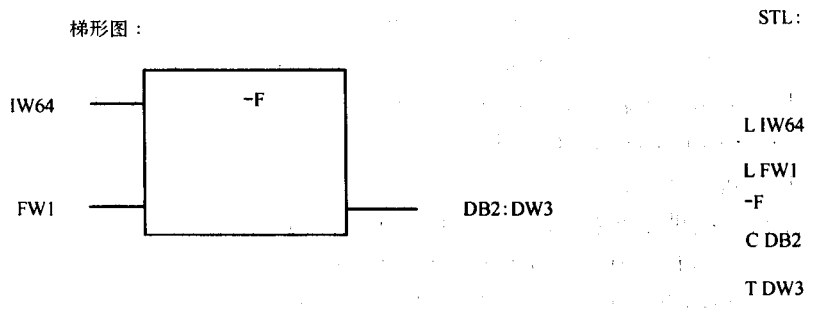


图 6-16 STEP 5 的 SUBTRACT 指令

STEP 5 也提供增（I）和减（D）指令，这两个指令用于增加或减小累加器 1 中低字节

的值。这个字节被看作是 8 位无符号数。累加器的高位不受影响。指令必须包含累加器 1 要加或减的值。例如，STL 指令“D 13”会从累加器的低位中减去 13，但是不会影响高位。

加（ADD）指令也可以用来增加一个有符号的常数值到累加器 1 的低字节（BF）或者整个 16 位的数（KF）。ADD 指令把累加器的值作为有符号数。例如，STL 指令“ADD BF −13”会增加 −13 到累加器 1 的低字节。

二进制补码（CSW）指令改变累加器中有符号 16 位数的符号。增、减、加和指令只可以在 STL 中使用，并且只能在功能块中。

STEP5 提供四个指令来执行逻辑运算。其中三个指令在逻辑上结合了从累加器 1 到累加器 2 的 16 位，并把结果保存在累加器 1。这些指令是与字（AW），或（OW），和异或字（XOR）。第四个指令，二进制反码（CFM），相当于在累加器 1 的字的位上执行 NOT 运算。这些指令都是无条件地执行。它们只可以在 STL 中使用，而且只能在功能块中。

S5 PLC 有一些预编的功能块[⊖]（还有一个组织块），可以由用户程序调用。有些功能块会自动地被梯形图指令调用。预编的功能块包括：

1）FB240 能把数字从二进制转换为 BCD 码，FB241 能把数字从 BCD 码转换为二进制。包括地址和常数值的参数为：

BINARY	16 位二进制数
BCD	16 位 BCD 数，用于 FB240
BCD1	FB241 的低 16 位 BCD 码输出
BCD2	FB241 的高 8 位 BCD 码输出
SBCD	BCD 码数字的符号位（1 是负）

2）FB242，用于执行乘法，能接受两个有符号 16 位二进制数作为输入，并产生一个 32 位的有符号二进制的结果（在两个 16 位字中）。一个位输出也指明是否乘法结果为零。参数包括：

Z1 和 Z2	相乘的两个 16 位数
Z31	结果的低 16 位
Z32	结果的高 16 位
Z3=0	指明结果为零的位（1 为不是）

3）FB243，用于除法，使用有符号 16 位输入值来产生一个有符号 16 位整数的商输出和一个 16 位的余数输出。生成四个输出状态位。参数包括：

Z1	被除数
Z2	除数
Z3	16 位整数结果（去掉一些位，不是舍入数）
Z4	余数（16 位）
Z3=0	指示结果为 0 的位（1 为不是）
Z4=0	指示余数为 0 的位（1 为不是）
FEH	指示被零除的位（1 表明是）
OV	指明除法溢出的位（1 表明是）

4）FB250，根据调用程序提供的参数来对一个模拟输入值进行标度变换；FB251 给模拟输出值进行标度变换。参数包括：

⊖ 在第 8 章讲述如何使用及如何写一个功能块。

BG	模拟输入或者输出模块地址的16位的数字部分
KNKT	由两个数字组成的常数，两个数字由逗号分开，指明模拟通道数，然后是一个代码识别模拟I/O模块的类型（请看你的操作手册）
OGR和UGR	指示标度变换范围的上限（OGR）和下限（UGR）的常数（有符号16位范围）。
XE	16位值，用于FB251的标度变换输出
XA	FB250的16位标度变换结果
EINZ	一个位，为1时能使FB250执行一次
TBIT	在FB250执行由EINZ命令的转换运算时，FB250保持这个位
BU	一个指明标度变换结果在范围以外的位，范围由OGR和UGR定义（1表示在范围之外）
FEH	一个指明KNKT、OGR或者UGR参数的误差的位，由FB251检测（1表示有误差）
FB	一个指明FB250检测到硬件错误或非法地址的位（1表示错误）

5）OB251，能计算伺服控制（PID）方程式的结果。OB251会在第12章详细介绍。

S7

3. SIEMENS STEP 7 数学、逻辑以及转换指令

所有的SETP 7 STL语言的数学及逻辑指令都会无条件地执行，将累加器中的内容作为操作数，将结果保存在累加器1中，同时影响状态寄存器标志位（CC0、CC1、OV以及OS标志位，详见附录F）。

梯形图的数学/逻辑指令包括了STL加载及传送指令，附加在STL数学/逻辑指令中。梯形图逻辑数学指令（不是逻辑指令）同样也包含了STL指令，用来在数学操作后检查状态寄存OV位，从而测定结果是否合法，如果合法就将一个输出（ENO）位置位。图6-17显示了一条将两个16位整数数值相加的梯形图逻辑指令，如果结果合法（如果状态寄存器OV位未置位）就将一个输出位置位，并保存结果。

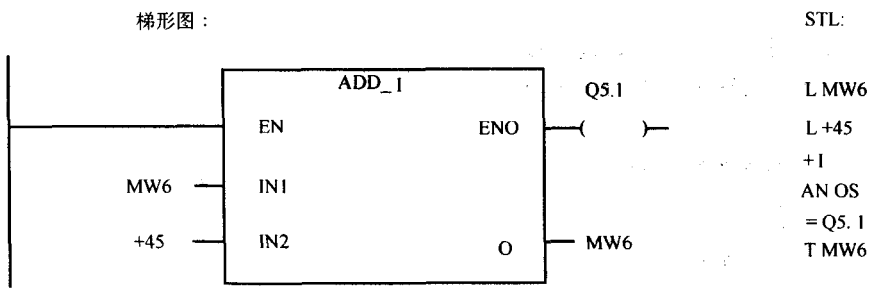


图6-17 STEP7整数加法指令

在STL程序中，编程人员必须使用结构化编程来确保数学/逻辑指令是条件执行的，同时必须编写对状态位的检查程序。如图6-17中，和无条件梯形图ADD_I指令等价的STL语言程序并没有包含任何被用来使指令条件化STL指令，但会检测OV状态位。

本章的余下部分，我们不会通过名称来识别Siemens梯形图逻辑指令。除非特别注明的情况下，每一条讨论的STL指令都有一个等价的梯形图指令。由于STEP 7梯形图指令兼容全部的STL指令，因此下面的描述同样也描述了梯形图的性能。

STEP 7允许执行数学操作使用有符号16位数字（I）代表整数，有符号32位双字（D）代表整数，或者32位浮点数（R）代表实数。新的无条件指令已经加入，用以实现对在累

加器 1 中数字的二进制数格式转换。

1）将有符号 16 位整数转换成有符号 32 位双字整数（ITD）。

2）将 BCD 数转换为 16 位有符号二进制整数（BTI）或者其求反的值（ITB），或者将 BCD 数转换为 32 位有符号二进制整数（BTD）或者其求反的值（DTB）。注意加载 BCD（LD）指令也可以用来将一个定时器或者计数器格式的数值转换成与载入累加器 1 中数值类型相同的 BCD 数。

3）将有符号 32 位二进制双字数值转换成浮点实数（DTR），或者从浮点数值转换成取整的有符号 32 位二进制数值（RND），强制舍入（RND＋及 RND－）或者截去尾数（TRUNC）。

在数值载入累加器 1 与 2 后，可以由四个基本数学操作来处理：

- 加（例如：＋I）
- 减（例如：－I）
- 乘（例如：＊I）
- 除（例如：/I）

第二个载入（累加器 1）的数加减或者乘除先前载入（累加器 2）的数。结果放入累加器 1。

1）使用 32 位有符号数（例如：＋D 和＋R）的基本数学运算会产生 32 位有符号数。如果因为需要 33 位才能正确显示的结果或者非法的符号变换而产生了错误结果，那么状态字中的相关状态位会置位（见附录 F）。程序会包含在数学运算之后监视状态位的语句。

2）16 位带符号数（例如：＋I）的加减运算使用累加器 1 与 2 中的低 16 位数值。得到的 16 位数值结果并不影响累加器 1 中的高 16 位，即使是操作得到一个 17 位的数值。程序会包含监视状态的语句。

3）16 位有符号数的乘法（例如：＊I）会导致一个 32 位有符号二进制结果被放入累加器 1 中，因此保存结果的传送指令应该将结果以双字形式传送。如果结果适合 16 位有符号数值，那么累加器 1 的低 16 位数值是正确的，并且可以作为字来传送。在累加器 1 中的低 16 位容纳不了计算结果时，S7 PLC 会将 OV 与 OS 状态位置位。

4）16 位有符号数的除法（例如：/I）会在累加器 1 的低 16 位中产生一个 16 位的数值，代表舍去符号后的整数值，而除法运算产生的余数存入累加器 1 的高 16 位。如果除法运算的结果作为一个双字传送到存储器，则程序中其他的指令会将结果与余数分别作为单字来读取。

5）32 位双字的除法运算（例如：/D）会在累加器中产生一个取整的 32 位数值，不保存余数。STEP7 提供了第五种数学运算指令，除并保存余数（MOD），该指令也同样执行 32 位除法，但 MOD 将余数存入累加器 1，并且抛弃商。/D 指令之后可以跟随 MOD 指令（在重新加载操作数后），这样 32 位商与余数都可以分别产生和保存，如下：

```
L   MD4
L   MD8
/D                  //产生 MD4/MD8 的整数商结果
T   MD12
L   MD4
L   MD8
MOD                 //产生 MD4/MD8 的余数
```

```
T  MD16
```

某些指令仅能对整数值进行操作：

1）增（INC）与减（DEC）指令仅在累加器1最低字节上操作。将指令中包含的无符号数值加到字节上，同时将该字节视为无符号整数，例如：

```
DEC 25    从累加器1的低字节中减去25
```

2）加一个整数（+）指令用于将一个有符号常数加到累加器1有符号的低16位字上，如果该常数是以十进制来输入；或者将常数加到累加器1带符号的32位双字上，如果该常数是以有符号的长整数常数格式输入。例如：

```
+   -45       在累加器1中加入-45到低字节
+   -L# 45    在累加器1中加入-45到双字
```

3）整数的二进制补码（INVI）指令将累加器1中的低16位字的符号改变，双字的二进制补码（INVD）指令改变了整个32位双字的符号。

累加器1中的浮点数（R）可以由几个新的无条件指令来操作（结果仍在累加器1）中，包括：

1）改变数字符号的指令：实数取反（NEGR）与绝对值（ABS）指令。这些指令的梯形图版本有一个当溢出状态位打开时关闭的ENO输出，即便如此这些指令也不会影响OV位。由于先前执行的指令可以将OV位置位，梯形图的程序员应当十分小心地使用ENO输出来控制子程序操作。下面的例子表明了如何使用NEGR指令：

```
L  MD4            //改变存储在MD4中的34位浮点数的符号
NEGR
T  MD4
```

2）对数及三角函数的数学指令，包括SQR、SQRT、LN、EXP、SIN、COS、TAN、ASIN、ACOS及ATAN。

STEP7提供了四个基本的逻辑操作符，都是无条件执行的，并且都影响状态寄存器。与字（AW）、或字（OW）、异或字（XOW）及整数的二进制反码（INVI）指令对16位字进行操作，将16位的结果保存在累加器1的低字中。（反码指令进行一个NOT操作。）相同指令的32位操作包括与双字（AD）、或双字（OD）、异或双字（XOD）以及双精度型整数的二进制反码（INVD）指令。AND、OR、XOR指令可以包含一个16或32位数值的地址，或者包含一个16或32位常数㊀，输入累加器1中（例如：AW#16#12FF）。如果没有地址或者常数包含在指令中，那么累加器2中的数值（由指令决定是低16位或者是全部的32位）会进入累加器1中。

在梯形图的逻辑指令中，ENO输出位并不反映任何状态寄存器位的状态。它仅仅是输出RLO，RLO在逻辑指令执行之前就已经存在。如果需要的话，程序员必须编程检测状态位。

STEP7寄存器间接寻址的附加指令

STEP7提供了两条可以用作高级数据寻址的指令。将常数加入寄存器1（+AR1）和将常数加入寄存器2（+AR2）指令，可以用来改变地址寄存器AR1及AR2的指针。改变一个指针使得在下一次一条使用寄存器间接寻址的语句执行时㊁，PLC对从不同地址来的数据

㊀ 不允许二进制常数，使用十六进制。

㊁ 函数和功能块执行时，AR2被STEP 7的操作系统自动使用，所以程序员要注意AR2如何变化。

进行操作。(寄存器间接寻址在第 5 章中有描述) 如果+AR1 与+AR2 指令不带任何附加参数来执行，那么累加器 1 的低字节中的 16 位数值便被加到地址寄存器中的 16 位指针中。如果+AR1 与+AR2 指令包括了以 P#byte. bit 形式的 32 位指针常数，则该常数值会被加到地址寄存器中的 32 位指针中。(注意指针常数不包括存储区域标识符。不能使用+AR1 或+AR2 来改变 AR1 或 AR2 中的跨区域存储区域标识符)

4. OMRON CQM1 数学、逻辑以及转换指令

CQM1

CQM1 有几个基本四则数学操作指令 (加、减、乘、除) 的集合，每一个集合对应 CQM1 对数值进行二进制编码的一种方法。如图 6-18 所示，有针对 16 位、32 位 BCD 数值的指令集合，有针对 16 位、32 位无符号二进制数值的指令集合，也有针对 16 位、32 位有符号二进制数值的指令集合。当然，CQM1 也提供了在这些二进制编码系统中转换的指令。还有几个特别的数学指令，如求平方根、平均值、三角函数以及标度变换。

数学 I 指令和参数要求

指令：	加	减	乘	除
16 位 BCD 码	ADD(30)	SUB(31)	MUL(32)	DIV(33)
32 位 BCD 码	ADDL(34)	SUBL(35)	MULL(36)	DIVL(37)
16 位无符号二进制数	ADB(50)	SBB(51)	MLB(52)	DVB(53)
32 位无符号二进制数	ADBL(-)	SBBL(-)	-	-
16 位有符号二进制数	ADB(50)*	SBB(51)*	MBS(-)	DBS(-)
32 位有符号二进制数	ADBL(-)*	SBBL(-)*	MBSL(-)	DBSL(-)

*可用于有符号或无符号二进制数的操作。用于有符号数时，上溢和下溢标志位可以用到，在后面讨论标志位时会讲到。

在数学指令中使用的参数	加	减	乘	除
第一	Au: 被加数	Mi: 被减数	Md: 被乘数	Dd: 被除数
第二	Ad: 加数	Su: 减数	Mr: 乘数	Dr: 除数
第三	R: 结果	R: 结果	R: 结果	R: 结果

图 6-18 数学 I 指令和参数要求

OMRON 的加减指令影响内存中状态寄存器区域 (位 25504) 的进位 (CY) 位，同时它们在计算时也包含了上次 CY 位的状态，通过如下方式：

加： R = Au + Ad + CY

加法运算后，如果结果大到 R 中存不下，则 CY 置位。

减： R = Mi - Su - CY

减法运算后，如果结果为负（例如，如果 Su-CY 大于 Mi）则 CY 置位。

自带的附加 CY 位和减法允许程序员编写顺序的加法和减法，因此也能对长于 32 位的二进制数字操作！有一些指令是特定地用来对 CY 置位：STC(40)，或者对 CY 复位：CLC(41)。

数学指令的参数，例如图 6-19 所示的 ADD(30)指令，可以是常数或者地址。

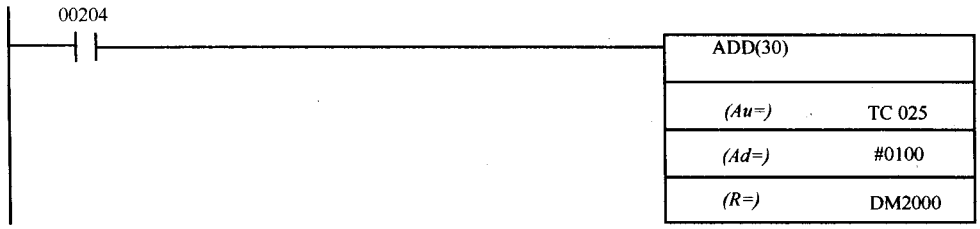


图 6-19 CQM1 数学指令，把 BCD 数 100 与代表定时器或者计数器当前值的 BCD 数相加，并把结果保存到数据存储器里

加法和减法指令得到的结果位数与他们的操作数位数相同。乘法指令得到的结果位数为他们操作数位数的两倍。每个参数只可以输入一个地址，即使指令使用了大于 16 位的数据字作为操作数。如果操作数要求为大于 16 位的数，或者如果得到的结果大于 16 位，数学指令自动地把下一个顺序存储器的位置增添到指定的存储器位置。指令中指定的作为参数的地址会保存这个较大的数的低 16 位，高一位的地址保存高 16 位数字。图 6-20 显示了一个乘法 MULL(56)指令。这个指令把两个 32 位 BCD 数（8 个十进制数）相乘，并产生一个 64 位的 BCD 数（16 个十进制数）。其中 8 位十进制数来自 DM2100（低四位数字）和 DM2101（高四位十进制数），而另外 8 位十进制数来自 DM2102 和 DM2103。相乘所得的 16 位十进制数结果保存到四个顺序的存储单元里：从 DM2104 到 DM2107 单元，其中 DM2104 保存低四位十进制数，DM2107 保存高四位十进制数。

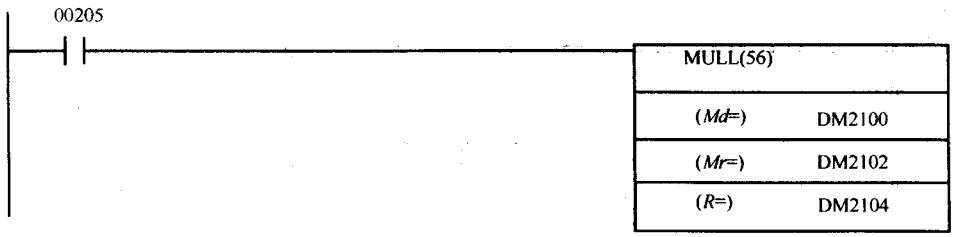


图 6-20 使用大于 14 位数的 CQM1 数学指令

除法指令得到的结果（商）的位数与操作数的位数相同，但同时也会产生一个余数。余数的位数也与操作数的位数相同，并自动地保存到商所在单元的上一个单元里。例如，包含结果指定地址 DM1000 的双有符号二进制除指令，会把 32 位商的低 16 位保存到 DM1000，把高 16 位保存到 DM1001，同时它也把除法所得的余数保存到 DM1002（低 16 位）和 DM1003（高 16 位）里。所以，你应该为乘法和除法的结果预留足够的空间。

还有一些另外的 CQM1 数学指令供程序员使用：

1）INC(38)和 DEC(39)对指令中指定地址里的 BCD 数字进行加 1 或者减 1 操作。

2）NEG(-)和 NEGL(-)，只在 CQM1-CPU4x 中才有，对这两个指令的二进制的 16 位或者 32 位数取反，并把结果保存到存储器单元的另一个位置。

3）ROOT(72)指令，用于取一个 32 位 BCD 数的平方根，产生一个对应的保存在另一个存储单元里的 16 位 BCD 结果。

4）MAX(-)和 MIN(-)，可用于找出存储器的一串顺序数据字里最大或者最小的数字，并把结果的数值保存到目的地址里。使用这条指令时，必须输入三个参数。首先是控制参数（C），指把数据值看作是有符号还是无符号二进制数、在多少个数据字里搜索最大或最小值，并且指出是否输出所找到的最大/小值的地址。第二个参数是范围（R1）参数，它指出了从数据列表的哪个地址开始搜索数据。最后一个是目的（D）参数，指出把最大或最小值写到哪个地址，如果选择了那个特性，那么它后面还会跟着搜索到数据值的地址。

5）SUM(-)，可用于把存储区里的一个连续区域内的一连串字或者字节加起来。控制（C）参数指数据是否为 BCD 码、无符号数还是有符号的二进制数，指是否把整个 16 位数据字相加，还是只把高字节的数或者只把低字节的数相加，并指出在这个相加的操作里包含有多少个数据字。另外，范围（R1）参数指出相加的数据从哪个地址开始。最后，目的（D）参数指将要保存 32 位结果的两个地址的第一个地址。

6）AVG(-)，当这个指令的控制逻辑为真时，计算一个单独的源字 64 次扫描循环以来的平均值。每次当控制逻辑从假变为真时，就开始一次新的平均值计算。这个指令需要三个参数。首先是源（S）参数，指在每次扫描循环计算平均值时读取的数值所在的单一的地址。读到的数值在计算平均值时被看作是普通二进制数。第二个是次数（N）参数（地址或者常数）指出（以 BCD 的形式）参与计算平均值（最大值＝64，尽管你输入的是一个更大的数）的扫描循环次数的最大值。最后是目的（D）参数，指保存计算出的平均值的地址。同时紧接着这个平均值结果地址后还有多达 65 个地址被占用。CQM1 用这些紧接着目的地址的地址单元来保存工作数据，而再紧接着的（多达 64）地址单元被用来保存从源地址读来的单独数据值。

7）APR(-)，被称为算术过程指令，因为它可用于两种不同的数学计算。这个指令需要一个控制（C）参数、一个源（S）地址参数以及一个目的（D）地址参数（按顺序）。

APR(-)可用来计算一个数的正弦或者余弦值。如果控制参数是常数＃0000，源地址的数值将被看作为以十分之一角度为单位的角度（以 BCD 形式），计算这个角度的正弦值，并保存（以 BCD 形式）到目的地址。控制常数是＃0001 时，就会使 APR(-)指令计算余弦值。

APR(-)还可用来对一系列的 X-Y 坐标进行线性近似。如果控制（C）参数是一个地址，APR(-)将会计算沿着一条线性近似线上的两个点。控制参数指定地址里的数值被看作为一个控制字，指出输入和/或输出的数字是 BCD 数字还是二进制数字，以及在这个计算中包含了多少个 X-Y 坐标。在紧接着控制字的地址单元里，程序员必须输入一个 X_M 值，用于计算直线上两点中一点的 Y_M 值，而这条直线是 APR(-)指令将要计算的那条线性近似线。接下来的地址单元里必须含有 X 和 Y 坐标数据，用来计算那条近似直线：X_0 定义为 0，所以它

不需要输入，因此列表中第一个数值是 Y_0 的值，跟着的是 X_1，然后 Y_1、X_2，然后 Y_2，如此类推，直到列表的结尾（X 值的输入必须是从最低到最高的顺序，并且不能大于 X_M）。源（S）参数指已输入的两个计算点中另一个点的 X 值的地址，而目的（D）参数指出 APR(-)指令保存对应那个 X 值的 Y 值的地址。

8）标度变换指令 SCL(66)，利用一条直线上的两个 X-Y 坐标点来计算沿着这条直线的第三个点的 X 值所对应的 Y 值。由于只有 ORMON 自己才知道的原因，X 值是无符号的二进制数，而 Y 值是 BCD 形式的数。第一个参数源（S）指第三个点的（二进制形式）X 值。第二个参数参量（P1），是四个已知点的值（Y_0、X_0、Y_1 和 X_1）中的第一个值的地址。最后，结果（R）参数指 CQM1 将把第三个点的（BCD 形式）Y 值写到哪个地址。

9）在 CQM1-CPU4x 及更高系列里，还提供了一个有符号十六进制到 BCD 标度变换指令 SCL2(-)和一个 BCD 到有符号十六进制标度变换指令 SCL3(-)。

在 SR 254 和 SR 255 中的数学指令可影响以下标志位：

1）进位标志位（SR 25504），加法和减法可影响进位标志位。这个标志位还可用于大于标准的 16 位和 32 位数字的加法和减法！例如，如果两个 64 位数保存在 DM2000（到 2003）以及 DM2004（到 2007）、DM2000（到 2001）和 DM2004（到 2005）中的一个 32 位加法后，紧接着另一个 DM2002（到 DM2003）和 DM2006（到 2007）的 32 位加法。第二个加法会自动地把来自第一个加法地进位标志位（如果有的话）考虑进去。如果你不希望进位标志位影响结果，一定要在任何加法和减法之前清除所有进位标志。

如果一个大的无符号数字（BCD 或者二进制数）被一个稍小的无符号数相减，进位标志位也会被置位，因此它指出了结果是一个负数，尽管 BCD 数和无符号数不可以正确地产生负数。如果 0[⊖]（在清除了进位标志后，所以进位标志不影响减法）被这个不正确的“负数”减，它可以转换成一个有效的 BCD 或者无符号二进制数字（它的绝对值）。否则，程序员必须通过某些方法记住这个 BCD 或者无符号数实际上代表的是一个负数的值（例如，通过对一个用户选择的状态位置位）。

进位标志位也可有由乘法来置位。没有简单的方法可以从产生过大的结果以致存储器不能保存的乘法中恢复过来，但至少程序可以辨认出结果是不正确的。

2）如果无符号加法或者减法产生一个结果，当这个结果向上超出或者向下超出有符号二进制数所允许的范围时，上溢出标志位（SR 25405）或者下溢出标志位（SR 25404）置位。在 CQM1-CPU4x 以下的 CQM1 不提供这些状态位，所以只有当溢出会引起进位错误时，才把标志溢出引入到这些 PLC 里。如果这些 PLC 的程序员希望使用一个图 6-18 中标有 * 的指令来执行有符号二进制数的加法或者减法时，程序应该包含有监视结果最高位变化的指令。

3）错误标志位（SR 25503）置位，PLC 不能读或写操作数的地址（可能因为地址不存在），或者间接地址偏移量含有不可识别为 BCD 码的位模式，或者除法指令被 0 除。

4）等于 0 位（SR 25506），当结果等于 0 时，这个位置位。

数据转换指令包含有：

1）BIN(23)和 BCD(24)。BIN(23)由源（S）参数（常数或者地址）指出，把 16 位

⊖ 对于 CQM1-CPU4x 或更高级的 PLC，一个负的无符号二进制数也可以通过一个补码指令 NEG(-)或 NEGL(-)转换为一个实的无符号数（绝对值）。

BCD 数值转换成 16 位无符号二进制数值，并把结果送到目的（D）地址。BCD(24) 把 16 位无符号二进制源数转换成 16 位 BCD 目的数，除非 BCD 数值可能超过 9999，这样的话将不能成功转换，但 CY 状态位（SR 25504）置位。

2) CQM1-CPU4x 以及更高级的 CQM1 也提供 BINL(-)和 BCDL(-)，它们可实现 32 位 BCD 和 32 位无符号二进制数之间的转换。

3) 使用 HMS(-)可把代表秒数（0 到 35 999 999）的 32 位无符号二进制数转换成时/分/秒格式。源（S）参数指出 32 位数保存的单元地址，目的（D）参数告诉 CQM1 把 BCD 时/分/秒结果存放到哪个单元。小时（0 到 9999）存放到 D 以上的地址，分（0 到 59）存放到 D 的高位字节，而秒（0 到 59）存放到 D 的低位字节。SEC(-)可用于把源（S）地址的时/分/秒格式转换成 32 位的只有秒数的二进制数保存到目的（D）地址。

4) MLPX(76)读取 16 位源（S）字中的一个单独的 4 位“字符”（半位元组）的值（0 到 15），并在 16 位结果（R）字里把对应位设为同样大小（0 到 15）。DMPX(77)的操作与 MLPX(76)相反，可以把多达四个 16 位源（S）数值，每个都至少有一个位已经设置，转换成一个单独目的（D）地址中的 4 位半位元组数值。(参考你的用户手册)

5) ASCII 转换指令 ASC(86)和 ASCII-TO-HEX 转换指令 HEX(-)，除了它们实现的是半位元组数值（16 进制的 0 到 F）和每个那些 16 进制数值的 8 位 ASCII 码之间的转换外，它们分别与 MLPX(76)和 DMPX(77)相类似。

6) 七段解码指令 SDEC(78)，可用于把 16 位源字的半位元组数值转换成适合驱动标准 7 段显示码的 8 位数值。

CQM1 提供了一套标准的逻辑指令，用于对 16 位数据字进行操作。取补码指令 COM(29)把被指定为字（Wd）参数（就在那个地址改变数值）的那个地址里的一个字的全部 16 位取反（执行非操作）。逻辑与 ANDW(34)、逻辑或 ORW(35)和异或 XORW(36)指令，使用两个输入字（I1 和 I2）来产生一个逻辑结果字（R）。同或 XNRW(37)等同于 XORW(36)后紧接着执行 COM(29)。输入参数（I1 和 I2）可以输入地址或者 16 进制的常数。结果参数（R）必须是一个地址。

6.5 故障检修

PLC 是一台计算机。它被设计为快速并精确地进行数学和逻辑操作。所以为什么你的程序会得出错误的答案呢？答案就是，如果程序有问题，计算机会按照程序执行得到错误的结果，或者进行一些超过自身计算能力的计算。

当你的程序尝试移动数据值，或者保存结果到不存在的存储单元时，会发生相关的容易发现的问题：如果发生这些情况，你的 PLC 很可能出错（停止运行），而且保存出错代码，这样程序员就可以找出问题所在。即使 PLC 不停止运行，也很可能会保存出错代码或者错误位到存储器里（第 11 章和 15 章就有关于出错中断和出错检修更详细地论述）。在有些 PLC 里，存储器必须先分配才能使用。然而当程序员编写程序时，存储器常常是自动分配的，存储器寻址使用间接或者变址寻址必须由程序员指定，因为编程软件是不能预先知道实际上需要哪些存储空间的。高级的寻址技术有时候会使 PLC 尝试写到程序员要使用的存储区域限制范围以外的地址。

可能问题应该归咎于你的数据编码成二进制数的方式。二进制编码方案包括二进制编

码数（BCD）、无符号二进制数、有符号二进制数以及浮点数。预编到 PLC 里的数学指令都被设计成专门针对某种编码格式的二进制数进行操作。只有当它们对适当的二进制数进行操作时，它们才有可能得到正确的答案。如果你使用一个 PLC 指令来把两个 BCD 数相加，而这个指令被设计为对有符号二进制数操作时，你得到的结果将会是错误的。大多数 PLC 的数学指令都是为有符号二进制数而设计的，但无符号数学和 BCD 数学指令仍然普遍，浮点数学指令更变得越来越普遍。数字编码系统要求使用正确的指令。检查并确认实际保存在存储器里的数字是如你所想的格式：你的编程器应该可以以二进制形式显示存储内容，或者可以写一个指令，把二进制数值输出到一个没有连接到执行器的数字输出模块，这样就可以看到 on 和 off 的模式是怎样的。如果需要的话，可使用数据变换指令来使数据编码与指令匹配。

问题也有可能归咎于数据单元的大小。如果数据数值是以有符号二进制字（16 位）的形式来存储，而数学或者逻辑指令是设计为对有符号二进制的双字（32 位）进行操作的，那么结果的低 16 位有时会是正确的，但有时结果也会出错，因为指令会在高 16 位里寻找指示负数的代码（可能你没意识到，每当你执行这个数学指令时，那些高位都一直在改变；现在你该知道为什么存储结果的地址上面的地址会自己改变内容了吧）。检查实际操作数和结果数据元素的大小，并与指令期望使用的数据单元的大小进行对比。

要记住，固定长度的二进制数可以表示的数值的范围是有限的，并且范围的极限是依赖于编码格式的。例如，16 位二进制数（一个字）只可以有 2^{16}（或者 65 536）种不同的模式，所以它可表示的数字是在－32 768 和＋32 767 之间，或者在 0 到 65 536 之间（决定于这个二进制模式被看作为有符号还是无符号的二进制数）。如果你把 1 加到表示 65 536 的无符号二进制数上的话，计算机将会得到结果 0（除非 PLC 被预设为禁止这种操作——参看你的用户手册）。类似地，把 1 加到表示 32 767 的有符号二进制数上，得到的结果就会是－32 768!如果这些情况有可能发生，你应该使用更大的数据元素大小和指令来对那些更大的数据元素进行操作。

大多数 PLC（本书中论述到的所有 PLC）都提供有状态位（或者标志位），它们会受数学操作或者有时候是逻辑操作的影响。状态位可以通过程序里的布尔指令检测到（永久的，或者只是在调试中），并且有时可以在编程器屏幕上显示。典型的状态位都包括一个当数学操作的结果为 0 时（如果加 1 到 65 536 时，就可能表示一个错误）置位的状态位，一个当结果可能表示负数（如果你把 1 加到 32 767 上想得到＋32 768，那么就会出现错误）时置位的状态位，一个当结果对数据元素大小过大而置位的进位标志位，还有一个溢出标志位，当它置位时表示结果过大以致不能正确地以有符号二进制数形式来表示。

大多数 PLC 编程软件都含有直方图特性，它通常用来记录 PLC 运行时一个数的历史值。直方图特性可以用来追踪程序员怀疑有可能会接近数字限制大小的数的变化，这样就可以评估并预防可能发生的数学错误。

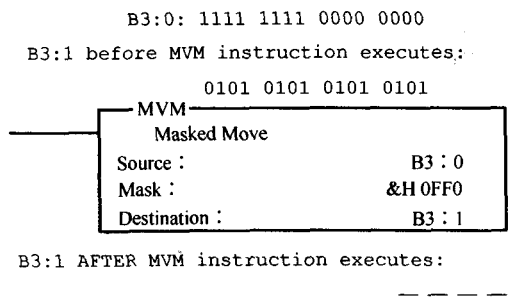
浮点数用来表示很大或者很小的数。它使用科学记数法的二进制形式来表示真实的数字。记住，尽管如此，标准的 32 位浮点数（已经出现 64 位版本，但还没普及到 PLC 的应用中）能表示的数字大小还是有一定限制的[⊖]。也许更重要的是，要记住 32 位二进制数可表示的只能

⊖ 见第 5 章限制范围。

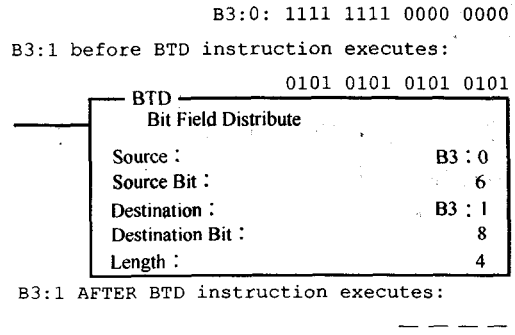
是2^{32}种不同模式，这就意味着不能用浮点数来表示在它的最大和最小值范围以外的任意可能的十进制数字。浮点数数学指令使结果更精确地接近可能的32位模式。你可能会发现，将一个浮点数转换成另一种二进制格式可能会因为尾数的舍入造成误差而导致结果出错。记住把一个浮点数转换成另一个二进制数的格式时，通常会包括舍入或截断的问题。

习题

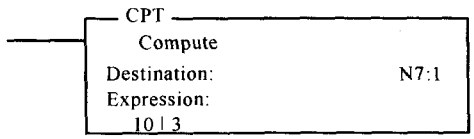
1. 什么是PLC CPU模块里的中心部件？它控制其他哪些类型的组成部件？
2. 识别典型状态寄存器中各个状态位的用途，并描述在PLC程序里这些位是如何使用的？
3. 当PLC执行MOVE指令时，通常使用哪个MPU寄存器？
4. 诸如非、与、或和异或等的逻辑操作可用于改变整个数据字里的位。其中哪个指令可以把所选的位置位？
5. 允许你写一个顺序器程序，用于读取数字输入的模式，忽略其中的一些输入，并把剩下的模式与输入掩模进行比较，如果模式匹配，就输出一个数字输出的模式。哪个（些）梯形图或者STL指令可代替编写这些顺序程序而在PLC里实现同样的操作？
   (a) 读一个含有8个输入映像位的模式。
   (b) 屏蔽某些位。
   (c) 对结果和输入掩模进行比较。
   (d) 输出一个模式到输出映像的8个位。
   Allen-Bradley PLC
6. PLC执行以下指令后在B3：1单元里将会是什么内容。



7. PLC执行以下指令后在B3：1单元里将会是什么内容。



8. 当 PLC 执行以下指令后，在 N7:1 中将是什么数（用十进制数回答）？



9. COMPUTE 指令执行一个数学运算操作，并把结果送到目的地址里。根据数学运算操作的结果，还有 4 个其他的位可能受影响。确定那 4 个位的每一位将会怎样设置。
10. 有些指令要求（或允许）你在一个地址里使用方括号，就像这个例子：T4：[N7：5]．DN. 那么 PLC 是如何知道这个地址是关于哪个定时器的“done”位的？

Siemens S5 PLC

11. 假设 Siemens S5 的输入模块 3 上的全部 8 个传感器都在运行中，并且没有其他输入运行，在每个以下的程序后，输出模块 4 的 8 个执行器将会如何变化？

程序1：

```
L IB3
T QB4          将打开执行器：_____
```

程序2：

```
L IW3
T QW4          将打开执行器：_____
```

程序3：

```
L IB3
T QW4          将打开执行器：_____
```

程序4：

```
L IW3
T QB4          将打开执行器：_____
```

程序5：

```
L KH0000
T QW4
A I2.0
L IB3
T QB4          当 I2.0开时，将打开执行器：_____
               当 I2.0关时，将打开执行器：_____
```

Siemens S7 PLC

12. 假设 8 个传感器都连接到提供数据给 IB3 的 Siemens S7 输入模块，而且假设当控制其他输入映像位的传感器都停止运作时，所有这些传感器都启动了，那么这 8 个由 QB4 控制的执行器在每个以下的程序后是启动还是停止？

程序 1：

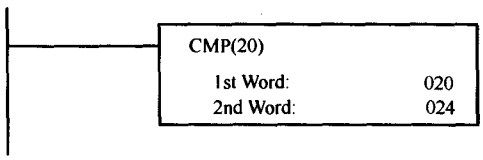
```
L IB3
T QB4          将打开执行器：_____
```

```
程序 2:
L IW3
T QW4          将打开执行器:_____
程序 3:
L IB3
T QW4          将打开执行器:_____
程序 4:
L ID3
T QB4          将打开执行器:_____
程序 5:
L B# 16# 00
T QB4
A I2.0
L IB3
T QB4          当 I2.0开时,将打开执行器:_____
               当 I2.0关时,将打开执行器:_____
```

13. STEP 7 中把两个 16 位数相加（ADD _ I）的梯形图指令包含有一个标为 ENO 的输出触点。如果 ADD-I 指令是“使能和操作”，这个 ENO 输出应该为 on。如果 ADD-I 指令使能时这个 ENO 输出不是 on。为什么不是 on？（不要只是说“它没有正确操作”，解释可能有什么地方出错!）

OMRON CQM1 PLC

14. OMRON CQM1 的比较指令不像 Allen-Bradley 或者 Siemens PLC 里的比较指令会影响逻辑操作的结果（RLO）。那么 OMRON 的比较指令会有些什么影响呢？
15. 以下的 OMRON CQM1 梯级含有一个比较两个数据字的指令。如果你希望 020 的数与 024 的数相等时，把一个输出置为 on，你需要在这个程序里加什么（说出描述这个位函数的位地址和 OMRON 名称，或者这个位代表的状态）



推荐的 PLC 实验练习

一个系统有：

- 四个控制面板开关：输入 0 到 3
- 四个指示灯或者输出模块的 LED 灯：输出 A 到 D
- 两个弹簧复位阀门控制的气缸：输出 E 和 F
- 一个锁销阀门控制气缸：输出 G 和 H

■ 三个传感器用来检测每个气缸的伸展

对于 PLC，你该学会使用：

1. 编写一个程序：
   (1) 当面板开关 0 是 on 时，复制一个常数到输出映像来伸展所有的气缸。
   (2) 包括一个计数器，以及每当开关 0 从 off 变为 on 则设置计数器的计数值为 5 的指令。
   (3) 当计数器的计数值在 10 到 15 之间时，指示灯 A 以 1/4 的循环速度闪烁。检查你的程序，使用数据屏幕来修改计数器的累加值的内容。
   (4) 当气缸 E 处的传感器启动后，使用逻辑算子屏蔽连接指示灯的输出映像的所有偶数位触点（包括 0），让指示灯持续灭 2 秒。
   (5) 每当开关 2 从 off 变为 on 时，把存储器里的一个有符号二进制数的大小加倍（如果你的 PLC 不提供乘法指令，就使用加法），然后：
   ■ 在这个有符号二进制数超过 16 位字的允许范围时，把一个输出锁存为 on，然后
   ■ 清除这个数，并且每当开关 3 从 off 变为 on 时锁存输出。
2. 编写一个程序：
   (1) 复制四个输入映像位到数据存储区的一个字里，通过这种方法，使一个开关控制数据字的高位。
   (2) 使用 COMPARE 指令，这样当数据字是：
   ■ 大于 0 时，就传送另一个数据字数值（常数或来自存储器的）到输出映像，以此来伸展所有的气缸。
   ■ 小于 0 时，移动单独数据字来收缩所有气缸。
   ■ 等于 0 时，把两个未使用的存储单元字的数值相加，并把结果保存到另一个单元里（使用数据监控屏幕来输入和改变数据值，并观察结果数值）。
3. 写一个程序：
   (1) 使一个计数器用来计算开关 0 变为 on 的上升次数，以及开关 1 变为 on 的下降次数。
   (2) 使用另一个计数器来计算第一个计数器的计数值改变的次数，但只是当计数值大于 5 后才开始计数。这暗示着：比较当前计数值与它在上一次扫描循环时的值。

第7章　文件、块、数组和结构体中的数据处理

7.1　学习目标

本章您将了解到：

- 数组、结构体、数据块和数据文件的定义；
- Allen-Bradley、Siemens、OMRON 的指令（或者功能块）用来完成：
  - 数组的移位或循环移位；
  - 数据字数组的移位：
    - FIFO 及 FILO 移位；
    - 顺序器指令；
  - 在 CPU 存储器中移动数据字的集合；
  - I/O 模块数据字集合的交换：
    - I/O 模块数据的块传送（Allen-Bradley）；
    - 通过共享存储器的数据进行发送与接收（Siemens）；
    - 串行数据字的读与写（OMRON）
  - 比较数据字集合，使用布尔逻辑中的状态位；
  - 数据字集合的数学、逻辑以及转换操作；
- 检测存储器中不需要的数据的指令（Allen-Bradley）。

若使用 Siemens PLC，则应在学习第 6 章与第 7 章之前先读第 8 章。

如果控制系统需要对数据字进行数据处理操作，则通常必须对连续数据字使用相同的操作。同时也有必要对两个相同的制造系统中的每一对计数器的累加值进行求和。同样的统计过程控制（SPC）的计算也可能需要对几个独立的传感器上的数据进行操作。或许你已经编好了一个在每次采样执行 SPC 计算时不会浪费时间的控制程序，但它仅仅是保存了一系列的采样值而已。你的 PLC 程序应当在时间允许的前提下，使用已经保存的采样值的文件来执行运算。数据文件用于顺序操作的控制，以及存储器中或者 CPU 与其他控制器之间的数据块的交换。智能 I/O 模块中的配置数据通常保存在 CPU 存储器中的一个数据文件中，这样就可以在需要时作为单独的数据块来传送。

数据文件、数据块、数组以及结构体都是用来描述保存在连续存储器的地址中的简单数据集合的。但是，每个 PLC 厂商都有独特的指令集以及处理文件、块、数组与结构体中的数据术语。

7.2 文件、块、数组和结构体定义

数据文件及数据块这两个名字，并没有在数据集合中的简单数据元素类型的任意标准定义。每个品牌的 PLC 在关于是什么组成了文件或者块上都有自己的定义。数组与结构体这两个名词的含义更好接受，但即使是 IEC 在 IEC 1131-3 标准中也已经避免将它们定义为标准数据单元，避免 PLC 厂商出现不支持者。

数组是一系列相同的数据单元。这些数据单元通常是简单的数据元素，例如位、字节、整数或者浮点数。数据位的数组可以占据几个数据字的空间（一个数据字有时会被当作位数组，而不用特别声明它是个数组）。ASCII 编码的字节数组通常称为字符串，数组可以是一维的（例如同一行的数据元素），也可以是二维的（例如数据元素的矩阵），还可以是多维的。一维数组需要一个变址号来确定数组中的第一个成员（例如 STUFF [4] 代表了 STUFF 数组中的第 4 个数据元素）；二维的数组需要两个变址号（例如 MORE [2，5] 代表了 MORE 数组中的第 2 行，第 5 列中的数据元素）；更多的维数需要附加相应的变址号。

结构体是数据元素的组合，这些数据元素不一定相同，甚至可能不是简单元素。一个结构体，举例来说，可以是包含了若干位、整数、数组甚至是其他结构体的组合。结构体通常是单维的，因此一个变址号就可以标注结构体中的元素，即使该元素本身是数组或者结构体。附加的变址号可能会用来标注子数组或者子结构内的简单数据元素（例如，MYSTRUCT.4 [2] 代表了一维数组中的数据元素 2，而该数组本身是结构体 MYSTRUCT 中的第 4 个数据元素）。复杂的足以支持结构体数据类型的 PLC 通常都允许符号寻址，因此数据元素名称可以用来代替变址号（例如，MYSTRUCT.STUFF [4] 也可以代表数据元素 MYSTRUCT.2 [4]）。

PLC-5 SLC 500

7.2.1 ALLEN-BRADLEY 的数据文件

Allen-Bradley PLC 的每一个数据文件都是用来保存通用类型的数据元素，其中一些数据类型是结构体或者数组，已经在第 5 章中有所提及。数据文件中的数据寻址也已经在第 5 章中提及。

S5

7.2.2 SIEMENS STEP 5 的数据块

S5 数据块用来存储 16 位数据字，数据块和数据块中的数据字必须在读写数据字的程序运行之前创建。最大可以创建 256 个数据块，每个数据块最大可以包含 256 个数据字。

一次仅允许一个数据块被激活。程序在使用该数据块中的数据字之前，必须执行数据块调用（C）指令。当新的数据块被调用时，前面运行中的数据块会停止运行。如果程序调用一个程序块，接着执行一个到其他程序的跳转，则在子程序运行时该数据块仍在运行状态；如果子程序调用了新数据块，则该新数据块在子程序结束之前都处在运行状态；当子程序结束而调用程序继续时，在跳转之前运行的数据块又会重新进入运行状态。

7.2.3 SIEMENS STEP 7 的数据块、数组和结构体

S7

S7 PLC 可以存储相似的数据内容的数组、不同数据内容的结构体以及数据块，数据块就像结构体，在其中包含了多种不同大小的数据元素。数组、结构体以及数据块可以在 STEP 7 中的变量声明表格中定义，也可以通过在编程软件中使用数据块创建属性来创建。用户数据类型（UDT）模板可以定义并用来创建结构体和/或数据块，如何在一组数据中创建元素及寻址在第 5 章中介绍过。

一次仅允许两个 S7 数据块处于开（激活）状态：一个是共享数据块，一个是立即数据块。共享数据块直到程序中的打开数据块（OPN）指令执行并激活它时才能够使用；而立即数据块则在相关的功能块程序被调用时自动加载，在该程序结束时自动结束。任何程序都可以打开新的数据块来代替当前运行的数据块（共享或者立即数据块皆可）。当程序被更高优先级的程序中断或者程序调用子程序时，S7 PLC 将在中断或者调用之前记录处于激活状态的数据块序号，保存在存储器区域中的 B 堆栈中。当子程序或者中断服务例行程序结束时，之前被激活的数据块会重新被激活。STEP 7 有一个交换共享和立即数据块（CDB）指令，可以将激活的共享数据块转换为一个激活的立即数据块，反之亦然。

如果程序试图使用并不存在的数据存储区，则 PLC 会执行同步出错例行程序（如果有的话），然后停止运行。第 11 及 15 章有更多相关的介绍。

7.2.4 OMRON CQM1 的数据集合

CQM1

OMRON 并不使用类似文件、数据块、数组或者结构体这样的词汇，但是 CQM1 可以预配置作为数据单元数组来使用的顺序存储器空间，第 5 章中介绍了这些存储器区域。由于程序员不用创建这些存储器区域，因此他们必须时刻注意文件处理指令使用的存储器是否真实存在。

如果一条指令试图处理超出了存储器区域边界的一组存储器（例如，使用 20 个保持寄存器地址，从 HR90 开始，在 HR99 之后都超出存储器区域边界），错误状态位（SR 25506）会置位；错误状态位在程序试图处理只读的 DM 存储器部分时也会置位，这些 DM 存储器部分是保留给系统配置和状态的。

7.3 位数组和移位指令

大部分 PLC 程序都有将单一数据字或者一组数据字向左或者向右移位的指令，如图 7-1 中所示。因为计算机仅可以处理完整的数据单元（典型的如 16 位数据字），一个正在移位的数据位的数组必须是该数据单元大小的整数倍。某些 PLC 指令会使位数组的大小看起来并不是完整数据字的整数倍，但要记住实际上并不是这样的。

注意到图 7-1a 中的移位的例子，最左边的位在移位后显示为“?”。当然事实上移位后要么是 0 要么是 1，但是究竟是哪一个取决于移位指令如何运行。某些 PLC 移位指令总会移入 0，而其他的 PLC 则允许程序员控制到底移入哪个值。PLC 有时会提供另一种移位指令，该指令称为循环移位指令。在循环移位中，位数组的一端移出的位会移入该数组的另一端，某些 PLC 循环移位指令会在执行循环移位的位数组中包含 CPU 状态位寄存器的进位信息。图 7-1b 显示了一个这样的循环移位，进位来的 0 循环移入数组的最左端，而新的进位则从

位数组的右端移出。

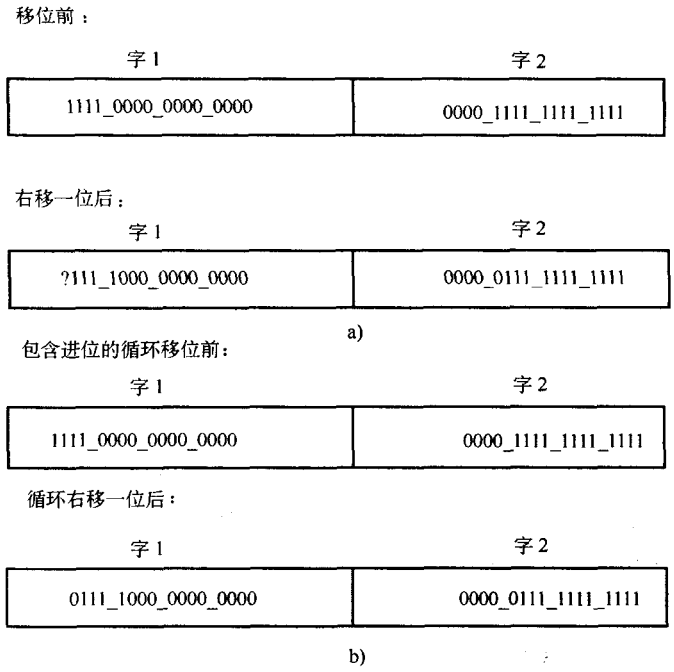


图 7-1 双字位数组：a）右移；b）循环右移

也可以使用位数组的移位。例如，在一个自动系统中，沿传送带排列的工作站跟踪工件的状态位，当传送带上的工件被发现出错时，相应跟踪该部件的状态位复位为 0。每次传送带传动时，PLC 都会执行一次移位操作，这样每个工件在存储器内的状态位都会在其沿着传送带传送一站的时候移位。PLC 程序会在告知工作站开始工作之前先检查相应的每个工作站的状态位，出错的工作站当然不允许再继续工作。新移入的位会显示新进入传送系统的工件情况，而移出数组的位，就好像传送完毕的工件一样，会被送入下一个自动生产系统中。

PLC-5
SLC 500

7.3.1 ALLEN-BRADLEY 的移位指令

PLC-5 以及 SLC 500 PLC 提供了*位左移*（BSL）、*位右移*（BSR）指令，可以将连续的数据字作为一个位数组来操作。移位指令在每次指令的控制逻辑判断为真时执行一次移位操作，每次移一位。Allen-Bradley 的移位指令会使位数组的大小看上去并不是 16 位的整数倍，就好像它们影响了数据字的一部分，而另一部分未受影响一样。但实际上并不是这样的，16 位数据字整体都会移位。

BSL 指令，如图 7-2 所示，会在其控制语句由假转为真时执行（当输入映像位 I：4/3 置位时）。由程序员输入的 BSL 指令数据一般包含：

1）文件地址（本例中为＃B3：2）。任何字地址都可以被输入，该字的最右位会包含在准备移位的位数组中，无论该指令是 BSL 还是 BSR。而“＃”是必须的，告诉 PLC 使用 S：24 以及变址寻址来跟踪每次必须移一个字的数据字。B3：[2＋S.24]是先作为B3：(2＋0)，然后

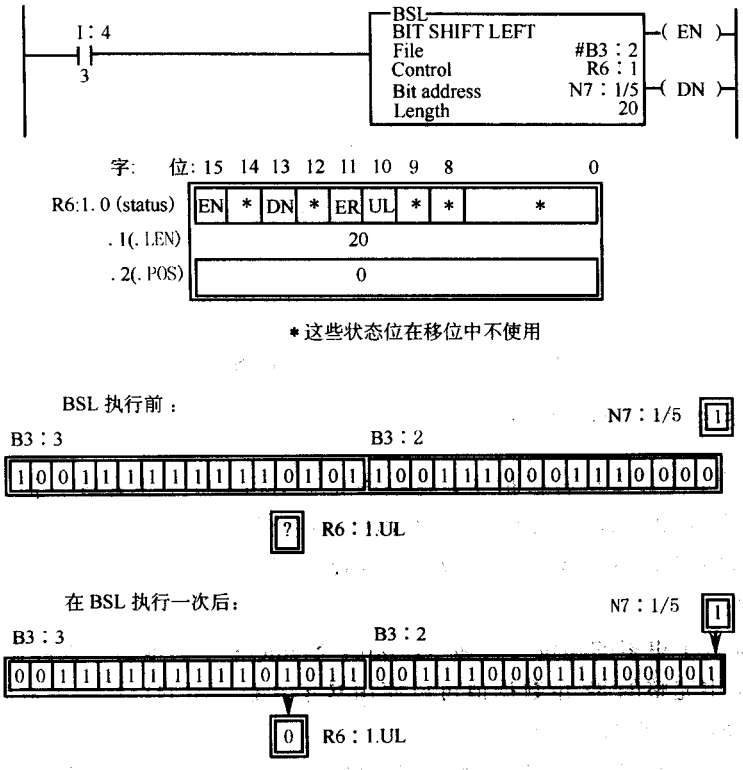


图 7-2 Allen-Bradley 位左移指令，控制块以及对存储器内容的影响

是 B3:(2+1)，接着是（如果位数组比本例中更长的话）B3:(2+2) 等一系列值来进行计算的。

2）长度表示了程序员希望在移位中包含的位的数量。位是从“文件”数据字的最右位向最高位计数的，如果长度大于 16 位的话，则进入下一个数据字。PLC 会将长度保存在控制元素的“.LEN”数据字中，同时 PLC 将该数保存到下一次 16 个数据位的全增时，来决定有多少位需要移位。图 7-2 中，为了移动 20 位，事实上移动了 32 位。

3）控制元素（Rf:x），PLC 用来保存需要在执行文件处理指令中使用的数据。控制元素的结构体中的“.POS”字用来跟踪究竟已经有多少位被移动。在移位指令中使用四个状态位：

- UL （未加载位）包含了在移位操作时，程序员打算移动的最后一个位的数值（1 或 0）。
  在 BSR 指令中会从文件地址字的最低位取值，在 BSL 指令中会从控制元素的“.LEN”字指示的数据位中取值（例如图 7-2 中的第 20 位）。
- EN （使能位）代表指令控制逻辑的状态（必须为真时指令才能执行）。
- DN （结束位）指令执行结束后为真，并一直保持到指令逻辑为非时。
- ER （错误位）指令试图进行操作但失败时为真，通常是由于编程错误引起的，有可能是因为位数组包含了并不存在的位地址。当控制逻辑为非时该位复位。

4）位地址，被指定用来指示 PLC 从哪儿获得数值来移入位数组中。从该地址中获得的值（本例中 N7:1/5），BSL 指令会移入文件地址字的最低位，而 BSR 指令会移入“.LEN”数据字指示的位中。

S5 7.3.2 SIEMENS STEP 5 的移位指令

STEP 5 提供了*字左移*（SLW）和*字右移*（SRW）指令，在程序中无条件地对累加器 1 中的 16 位数值进行移位。指令后跟随的数值（0 到 15）显示了要移多少位，而移出后空出的位由 0 来填充，移出的位通过 CC1 位移出。因此如果没有其他指令影响 CC1 位，移位操作执行后可以通过检测 CC2 位来确定最后一个移出的位。例如：

	累加器1	CC1位
L KH F05F	1111 0000 1100 1111	?
SRW 3	0001 1110 0001 1001	1

SRW 以及 SLW 指令仅可以在功能块中编程。梯形图版本中也提供了这两个指令。

S7 7.3.3 SIEMENS STEP 7 的移位指令

STEP 7 提供字左移及字右移指令，这些指令在 STEP 5 中也存在，但是在 STEP 7 中它们可以在任意程序块中编写，而且工作方式也有些许不同：

1）*字左移*（SLW）与*字右移*（SRW）指令仅对累加器 1 中的低字（16 位）进行移位，对高位没有影响；

STEP7 也加入了更多的移位与旋转（rotate，Siemens 中的术语，也即循环移位）指令。

2）累加器 1 中完整的 32 位数据值可以使用*双字左移*（SLD）及*双字右移*（SRD）指令来移位，其后须加上一个数（0 至 32）。

3）累加器 1 中的低字，或者完整的双字，可以使用*有符号整数移位*（SSI）或者*有符号双字整数移位*（SSD）指令来移位，而无须改变字或者双字最左边的符号位。一个数字（0 至 15 或者 0 至 32）显示有多少位需要移动。

4）累加器 1 中的 32 位数值可以循环左移或者右移。循环移位时，在双字一端的位值会立即移入双字的另一端，因此位模式的序列并未丢失信息。*双字左循环移位*（RLD）与*双字右循环移位*（RRD）指令，其后必须跟随一个指示有多少位需要移动的数值（0 到 32）。

5）上述的任何移位与循环移位指令，都可以在默认指示需要移动多少位的情况下输入程序（仅在 STL 中）。在这种情况下，S7 PLC 会读出累加器 2 中的数值，来决定究竟在累加器 1 中有多少位需要移位。例如：

	累加器 1	累加器 2
L + 3	3	?
L MD4	11111111111111000000000000000000	3
RLD	11111111111000000000000000000111	3

6）一个 33 位的循环是可能的，但仅限于 STL 编程。第 33 位是 CC1 位，该位接收一端移出的位，同时在累加器 1 的另一端移入一位数值。这些指令不需要指示需要移动多少位，因此它们每执行一次只循环移动一位。这些指令包括*通过 CC1 循环左移*（RLDA）指令以及*通过 CC1 循环右移*（RRDA）指令。

大部分上述的移位与循环移位指令都有相应的梯形图指令。必须给梯形图输入一个数（N），以指示有多少需要移动的位，但该数值必须为正值，最大为十六进制数 FFFF。梯形图指令的 ENO 输出与 CC1 位的意义是相同的，都代表了最近的移位或循环移位操作的位状态。

7.3.4 OMRON CQM1 的移位指令

CQM1

CQM1 提供了四个移位及循环移位指令来改变单一的数据字，提供了两个指令用来对更大的位数组进行移位的指令。下面的指令在每次扫描循环中，当控制逻辑为真时，对存储器中一个单一的 16 位数据字进行移位或者循环移位，一次移一位。如果以微分形式编程（带一个@前缀），则当控制逻辑由假变为真时，指令只执行一次移位。这些指令影响进位（SR25504）、零位（当 16 位结果为零时 SR25506 置位），同时影响错误位（SR25503）。

1）*算术左移* ASL(25)，*以及算术右移* ASR(26)。ASL(25) 让 16 位数值向最高位（位 15）移动一位，第 15 位移位到进位（SR 25504），而进位位中原先的数据丢失。数值 0 被移入最低位（位 0）。ASR(26) 会让 16 位数值向最低位移动一位，位 0 移入进位位中，而数值 0 移位到位 15 中。

2）*循环左移* ROL(27)，*以及循环右移* ROL(28)。ROL(27) 让 16 位数值向最高位（位 15）移动一位，第 15 位移入进位位（SR 25504），而进位位中的数值移入最低位（位 0）。ROR(28) 会让 16 位数值向最低位移动一位，位 0 移入进位位，而进位位中的数值移入位 15。

下列移位指令可以对一组数进行移位。它们的编程方法有很大区别。

1）*寄存器移位指令* SFT(10)，如图 7-3 中所示。有两个参数，第一个参数是地址的起始字（St），（图中的 IR 130），第二个参数是地址的终止字（E），（图中的 IR 132）。因此该指令可以使输出字 130、131、132 中的 48 位数值同时移位，SFT(10) 指令编程时最多可以有三个布尔逻辑控制输入。当控制逻辑在脉冲（P）输入由假变为真时，移位寄存器中所有的位（从起始字到终止字）向着最高地址的最高位移动一位（字 IR132 的第 15 位），移位寄存器的最高位丢失（字 IR132 的第 15 位），输入位（I）的控制逻辑的状态位（1 为真，0 为假）移入最低字的最低位（字 IR130 的 0 位）。如果重启（R）输入的控制逻辑为真，则移位寄存器的所有位均重置为 0，且直到控制逻辑再次变为假，才继续允许移位操作。SFT(10) 指令并无微分形式（即不允许加前缀@），因为每次脉冲逻辑由假变为真时它只执行一次。

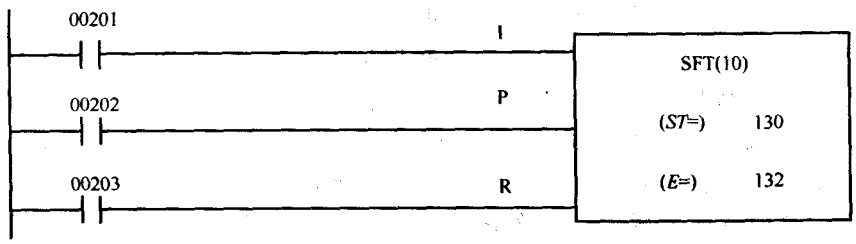


图 7-3 程序中的 CQM1 移位寄存器指令

2）*可逆的寄存器移位指令* SFTR(84)，与 SFT(10) 指令类似，但编程方法不同，如图 7-4 的例子所示。同 SFT(10) 指令一样，起始字与终止字地址都是必须的，但是作为第二和第三参数输入，第一参数是控制字地址（C）参数（图 7-4 中的地址 IR012 是工作区存储器），必须是一个地址，而不是常数。SFTR(84) 指令有一个可选的微分形式（本例中使用），因为在其非微分形式指令下，在每次控制逻辑持续为真时进行的扫描循环都会移动一位数值。在微分形式下（加@前缀），每次控制逻辑由假变为真时，仅移动一位。

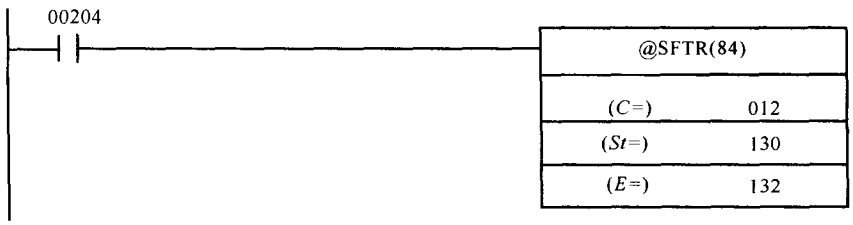


图 7-4 程序中的 CMQ1 可逆移位寄存器

控制字中的位执行了 SFT(10) 指令中的控制逻辑输入的功能，以及更多其他功能。(控制地址中的位可以由用户程序中的其他梯形图梯级来控制)。如果复位位（位 15）关闭，则整个移位寄存器的内容（字 IR 130 至字 132）都会复位为零，不再移动。用户程序必须先打开复位位（本例中的 IR 01215），这样 SFTR(84) 指令才可以工作。输入位（本例中的位 13，IR 01213）包括了被移入移位寄存器最右端的数值（如果操作是左移位的话），或者被移入寄存器最左端的数值（如果操作是右移位的话）。位 12 是移位方向位，控制着移位的方向（1 为右，0 为左）。位 14 称为移位脉冲位，但可以认为是启动位。移位寄存器在移位脉冲位关闭时停止工作，因此用户程序也必须将此位打开。位 0 至位 12 在控制地址中无效。

其他 CQM1 移位指令影响了若干位的组合（4 位半位元组或者 16 位字），在随后描述数组操作的章节中会予以讨论。

7.4 数组移位指令（包括 FIFO 及 LIFO）

就像位可以移动一样，某些 PLC 也允许完整的数据字（或者部分字）在连续存储器区域中移位。一组存储器区域，举例说明，可以用来保存独立的工件的描述性信息，而这些描述性数据可以当工件在自动化系统中传送时，在描述符堆栈中上下移动。

先入先出（FIFO）以及后入先出（LIFO）用以描述从存储器区域的堆栈中数据存入及获取的两种方式。存入存储器区域中的数据数值被压入堆栈的顶部。数据可以从堆栈的底部获取，这样在堆栈中停留时间最久的数据最先被移出（FIFO）。如果数据从顶部获取，那么最近放入堆栈中的数据最先被移出（LIFO）。PLC 在将数据压入堆栈顶部以及将数据推出栈时都需要保存堆栈控制信息。

即使 PLC 不提供任何数组移位指令，程序员通常也可以通过使用变址寻址或者直接寻址方式来达到同样的效果。如果每次一条指令运行时，它的变址寻址偏移量或者直接寻址指针会增量或者减量，那么它就可以用来对连续的数据值进行操作。地址就这样被移位，而不是切实地将存储器中的数据移位。FIFO 及 LIFO 指令通常在内部以这种方式使用变址寻址。

PLC-5 SLC 500

7.4.1 ALLEN-BRADLEY 的 FIFO 及 LIFO 指令

PLC-5 及 SLC500 均提供了 FIFO 与 LIFO 指令。图 7-5 表示了一个使用 FIFO 加载（FFL）指令和 FIFO 卸载（FFU）指令的程序。每次控制逻辑由假变为真时（例如，当 B3/4 开启时），FIFO 加载指令将一个 16 位数据值载入 FIFO 数组中。FIFO 卸载指令在每

次 B3/5 开启并且仅当 FIFO 非空（例如，如果 R6：2EM 关闭）时将最早的信息从 FIFO 中清除出去。在输入 FFO 及 FFU 指令时，程序员必须输入：

1）源地址（仅对于 FFL），用以指准备压入堆栈的数据字是从哪儿复制过来的。在本例中，堆栈中已经有三个数据字，因此从源地址来的下一个数据字会被压入第四个位置。堆栈的顶部是最大编码的地址，包含了已经被压入堆栈的数据字，同时当数据出栈入栈时堆栈顶部的地址也随之改变。

2）目的地址（仅对于 FFU），用以指准备出栈的数据字复制到什么地方去。在图 7-5 中，堆栈中最早的数据总是处在最低编码的地址位，当它出栈时，PLC 必须对存储器区域中的所有剩余数据值移位，这样保证下一个最早的数据移入最低编码的地址中。

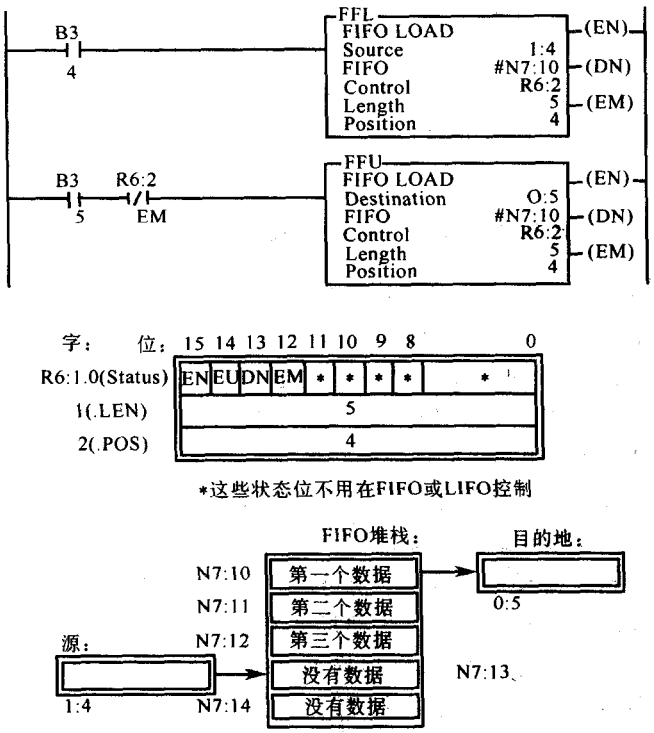


图 7-5 Allen-Bradley FIFO 加载与 FIFO 卸载指令，控制块及 FIFO 数据堆栈

3）FIFO 地址，带有“#”标记，指示了堆栈底部地址的位置（最低地址的编码）。

4）长度由 FIFO 地址起始，指多少存储器空间用作堆栈。最大长度值为 64 字。

5）控制元素必须输入每一个 FIFO 堆栈中。FFL 及 FFU 指令都必须使用同样的控制元素，这些元素包含了指示 FIFO 最大容量（长度数值）的数据和目前在 FIFO 堆栈中的数据字的序号（皆可被 FFL 和 FFU 改变），控制元素包含下列状态位。

- EN （加载使能），当 FFL 指令的控制逻辑为真时打开。
- EU （卸载使能），当 FFU 指令的控制逻辑为真时打开。
- DN （完成），当 FIFO 堆栈充满时为真。当充满时，PLC 不会再向堆栈压入新的数据字。
- EM （空值），当堆栈中无数据字时为真（所有压入的字都已经被弹出）。它会被当作 FFU 或者 LFU 的控制逻辑的一部分来检查，用以确认在卸载数据之前堆栈中是有数据的。

LIFO 指令，LIFO 加载（LFL）及 LIFO 卸载（LFU）指令，与 FFL 及 FFU 指令是类似的。LIFO 与 FIFO 的区别在于数据是从 LIFO 堆栈的顶部（最高位地址）卸载（弹出）的，因此对于 PLC 而言用不着将堆栈中的数据字移上或者移下。

S5

7.4.2 SIEMENS STEP5 的数组移位指令

S5PLC 系列并不提供数据字数组的移位指令。程序员必须编写自己的能够在连续存储器空间中移动数据的程序段。第 5 章中讨论过的变址寻址过程操作（DO）指令和过程间接（DI）指令可以使用，同时第 8 章中讨论的结构化的编程技巧也会有所帮助。

S7

7.4.3 SIEMENS STEP 7 的数组移位指令

STEP 7 也不提供任何专门针对结构和数组中的数据移位指令。程序员必须自己编写相应的程序。存储器间接寻址和寄存器间接寻址技术，以及在数组和结构（在第 5 章中有讨论）中使用的寻址技术，都可以用来简化这部分工作。在第 8 章中讨论过的结构化编程技巧也可以使用。

CQM1

7.4.4 OMRON CQM1 的数组移位指令（包括 FIFO 和 LIFO）

尽管 OMRON 技术方面的语法规则并没有使用数组这个字眼，但是 CQM1 是可以对 16 位数据字数组，或者 4 位半位元组（OMRON 中称为数字）数组进行移位操作。OMRON 异步移位指令就特别适合对数据字数组的移位，通过与产品在带有多种存储缓冲区的生产系统中传送相类似的方式。FIFO 及 LIFO 可以结合数据收集指令，使用单一字分配指令来编程。CQM1 也有专门设计的用来将键盘数值转换成小型数组的指令。

字移位指令 WSFT(16)，可以用来对一个存储器区域中数组内 16 位数据字进行移位。当字上移时，16 位数值 0 移入最低位地址中，最高位存储器地址中的内容移出数组，自然丢失掉。WSFT(16) 指令需要两个参数：*起始字地址*（ST）和*终止字地址*（E）。WSFT(16) 有微分指令形式：@WSFT(16)，这种形式是需要的，因为，在非微分指令状态时，WSFT(16) 在它的控制逻辑为真时每次扫描循环都会执行数组移位操作一次。附加指令（例如 MOV 指令）也可以包含在程序之中，用以在 WSFT(16) 执行之前从终止字地址中复制出数据字，在 WSFT(16) 执行之后将新的数据字复制入起始字地址中。

有两条指令可以将 4 位数字数组移位：*左移一位数字指令* SLD(74) 和*右移一位数字指令*，SRL(75)（OMRON 中称半位元组为数字，因为 CQM1 通常使用 BCD 编码，在其中使用二进制半位元组来表示十进制数字）。数字移位指令需要一个起始字参数（ST）和一个终止字参数（E），但在一个字内移动半位元组时，ST 可以与 E 拥有同样的地址。当 SLD(74) 指令执行时，它将 4 个零位移入起始字地址的最低四位中。SRD(75) 指令将零移入终止字地址的最高四位中。其他的半位元组都会被左移或者右移四位。左移出一个字高字节的半位元组，移入下一个较高字的低字节中，反之亦然。而移出半位元组串端点的四位则被丢弃。除非使用微分形式的指令（@SLD 与@SLR），否则当执行控制逻辑持续为真时，每一次扫描循环都会执行一次移位操作。

异步寄存器移位操作 ASFT(17)，在数据字数组中将数据字上移或下移，但仅当数组中

有足够保存字的空闲空间时，移入一个字。(带有数值 0 的存储器被认为是空闲空间) 于是移出数据字的存储器会成为空闲空间 (带有数值 0)，当下一次 ASFT(17) 指令执行时就会有一个数据字移入那个空间中。每一个数据字都移入数组中存在的每一个空闲空间中 (在其后留出一个空闲空间)。ASFT(17) 在其控制逻辑持续为真时，每次扫描循环都会产生一组新的移位，除非使用微分形式的指令，@ASFT(17)。ASFT(17) 需要一个控制参数 (C)，其后还要跟随一个起始字参数 (ST) 及一个终止字参数 (E)。当然 ASFT(17) 指令的控制逻辑必须为真时指令才执行，但当指令执行时，控制字才能确定它应该做什么。控制字中方向控制位 (位 13) 的 1 使 ASFT(17) 指令将存储器中的字下移，而 0 意味着上移。移位起动位 (位 14) 必须打开，否则 ASFT(17) 不会对任何数据字进行移位。当复位位 (位 15) 开启时，数组清空 (所有字都变为 0)。这些控制字位可以由用户程序中的其他指令来操作。而其他的控制字位是不能够被 ASFT(17) 指令使用的。

FIFO (先入先出) 堆栈数组和 LIFO (后入先出) 堆栈数组可以使用单字分配指令 DIST (80)，将数据字放在堆栈之上来实现，可以使用数据收集指令 COLL(81)，将数据从堆栈顶端或者底部移除。DIST (80) 在堆栈超出其分配的大小的情况下将错误状态位置位 (SR25503)。(这两条指令也可以在变址寻址移动的类型中使用，在第 6 章中有讨论) 图 7-6 中的例子显示了如何使用 DIST(80) 及 COLL(81) 来实现 FIFO 和 LIFO 堆栈的过程。

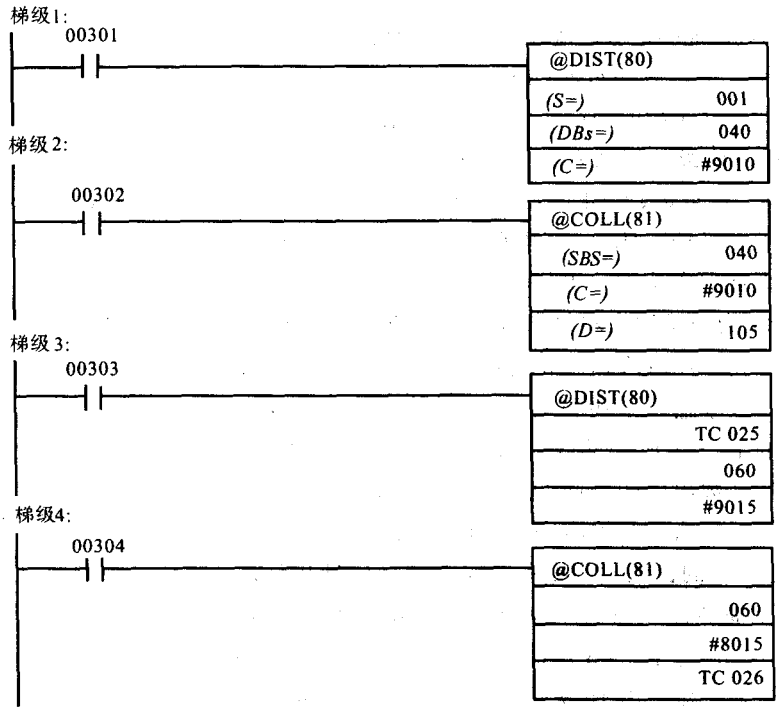


图 7-6 程序中的 CQM1 FIFO 与 LIFO 指令

在运行图 7-6 中的程序之前，程序员已经将数据存储器 IR 040 至 IR 060 中的内容清空。梯级 1 处包含了一个@DIST(80) 指令，该指令有一个将 BCD 数值 9010 作为常值输入的控制字 (C)。控制字的最高位为 9，因此@DIST(80) 会执行一个堆栈操作，而其后的数字

010显示了堆栈有大小为10字。当输入模块3（IR 00301）的位1由假变为真时，微分指令@DIST(80)将堆栈指针增加，同时从源字（S=IR 001）中复制一个数据字至10字大小堆栈的顶部，同时将其他字向堆栈中压。一个堆栈指针保存在由目的基准字参数（DBS=IR 040）确定的地址中，堆栈的顶部实际上是一个存储器区域的高位（IR 041）。当新字被压入堆栈顶部时（IR041是顶，尽管在堆栈中它是最低编号的地址），堆栈中的其他字也必须压入更高的存储器区域中。因此在堆栈中停留最久的字可以在一个加上了目的基参数地址（IR040）和其所包含的堆栈指针（偏移量）数值的存储器区域中寻址得到。例如，图7-6的程序中，如果三个字压入了堆栈中，那么IR 040中的内容就会从其初始值0000增加到0003。目的基参数地址是IR 040，因此堆栈中停留时间最久的数值在IR 043中。

梯级2处，一个微分形式的COLL(81)指令在每次输入模块3中的位2由假变真时执行一次。由于@COLL(81)的控制字（C）在其最高位上是9，@COLL(81)指令会从堆栈的数据字中执行一个FIFO移除数据字的操作，将其写入目的字（D）中（本例中IR105）。堆栈长度为10，如控制字中的低三位所显示。源基准字（SBS）参数是地址IR 040，因此该指令使用了与先前梯形图梯级使用的相同的堆栈。

当@COLL(81)从堆栈的数据字中执行一个FIFO移除数据字操作时，它将堆栈中最早的字移除（本例中目前应为IR 043），接着将在IR040中的堆栈指针偏移量数值减1。注意到IR 040在@DIST(80)与@COLL(81)指令中都被当作堆栈指针使用，因此梯级1可以持续将数据压入堆栈之中，只要梯级2能够足够快地移除它们，使得堆栈永远不超过10字的容量即可。

在梯级3及4中，另一对@DIST(80)与@COLL(81)，对从IR 061开始的15位字堆栈进行操作，同时在IR 060存储了一个堆栈指针。计数器（TC 025）中的当前值通过梯级3放入堆栈，通过梯级4复制入另一个计数器（TC 026）。此时，@COLL(81)指令的控制字的最高位为8，因此@COLL(81)指令执行一条LIFO，将堆栈中的数据字移除。在LIFO数据移除过程中，堆栈（IR061）顶端的数据字总会被移出堆栈（如果顶端有数据字的话），而堆栈中的字则被移向顶端，堆栈指针减1。

某些CQM1指令是用来将键盘数据转换成小型数组的。

1）10键输入指令TKY(18)，可以检测一个输入字的低10位中哪些位是打开的（IW参数通常指示的是数字输入模块的地址），同时也会记录该位的序号（0至9），并将该序号作为双字数组中的下一个BCD数字，该双字数组起始于第一个寄存器地址字（D1）参数所指定地址的高四位。数组中的先前的半位元组右移（移入D1下一个地址的低四位中），接着再移出，因此最近的八个开关激励可以保存在D1和D1+1中。指定作为第二寄存器地址字（D2）参数的地址中的位显示出哪一个输入位最后打开及究竟该位是否继续打开。

2）十六进制键输入指令HKY(-)，当它依次顺序激活一个输出字（OW）地址的低四位时，用来检测一个输入字（IW）地址的低四位。如果一个十六进制键区连接着那些输出位和输入位，则结果会显示出16键位（0至F）中哪些被按下。并且该十六进制数值会送入由第一个寄存器字（D）地址参数所指定的存储器的高四位中，该地址以及其下一个的地址中先前的四位数值全部右移一位，最高字的低四位会丢失。8键盘入口就这样被保存在数组中了。第一个寄存器字之上的第二个字的位显示了最近被激活的十六进制键（0至15表示0至F），当键始终被按压时输出字地址的4位开启。由于操作需要占用若干扫描循环才能获取一个键击值，CQM1在HKY(-)执行时将一个状态位（SR 25408）置位。将HKY(-)的

控制逻辑关闭，以防它连续不断地重复读取键值。

7.5 文件、数组和结构体的移动

某些 PLC 提供了很强大的移动指令，可以从 PLC 存储器的一块区域中复制全部的或者部分的数据到另一块区域中。举例来说，这样的指令可以用在每一段时间保存记录时，将输入模块的整体框架中的当前输入状态复制到存储器中，或者用来在每当传感器检测到输入工作站的产品类型发生改变时，改变全部的工作数据值。在某些 PLC 中，块传送指令用来在 CPU 存储器与智能 IO 模块存储器之间复制数据数组。

顺序器是现代 PLC 的前身。顺序器执行文件移动操作，从连续的存储器区域中复制数据字，一次一个到一个单出口端口地址中，以先前定义好的状态集来使一组执行器执行一步。某些现代 PLC 包括了顺序器指令。顺序器输出指令可以用布尔逻辑语句来控制，用以检测定时器输出（为输出改变的定时间隔）或者输入位（为事件触发输出改变）。逻辑条件控制顺序的输出甚至还包括顺序器比较指令。

7.5.1 ALLEN-BRADLEY 的文件移动、顺序器和块传送指令

PLC-5 SLC 500

PLC-5 与 SLC 500 都提供了文件复制（COP）指令，在文件复制指令的控制逻辑持续为真时，每次扫描循环从存储器的一个区域中复制全部的连续数据元素到另一块区域中（并非当控制逻辑为真时仅执行一次）。任何类型的数据元素都可以被复制，甚至用于计数器控制的三字长结构体也可以被复制。COP 指令事实上将数据以 16 位数据字形式复制，因此数据可以在不同数据文件类型之间复制传送（即使是在整数文件与计数器元素文件之间），但要注意 COP 指令并不会在复制数据的时候转换数据。在拥有不同数据元素大小的数据文件之间复制传送数据时，COP 指令会按要求复制一定数量的长度的元素来填充由程序规定长度的目的元素。如图 7-7 所示，指令的源参数与目的参数必须都加上“#”前缀，因为 PLC 必须使用变址寻址（自动使用状态字 S：24）来跟踪它在源文件和目的文件中使用的地址。本例中，并没有条件控制 COP 指令，因此每次扫描它都会从地址 N7：4至 N7：33 的地址段中复制 30 个整数数据字到 C12：2至 C12：11 的十个三字计数器单元中。

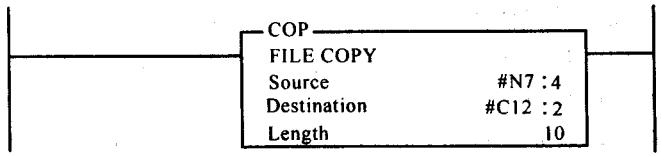


图 7-7 Allen-Bradley 文件复制指令

文件填充（FLL）指令，将单一地址中的内容（或者常数）复制到目的地址的所有元素中。当然，#标记符在源地址中不是必须的，因为仅需要单一地址。

Allen-Bradley PLC 提供了顺序器输出（SQO）指令，如图 7-8 所示。每次指令的控制逻辑条件为真时，SQO 指令会从连续的存储器区域中复制下一个数据字到目的地址中。连续的数据字会在 SQO 指令使用它们之前输入存储器中。程序员必须输入：

1）连续数据字的起始文件地址。#标记符是必须的，因为 PLC 使用变址寻址。

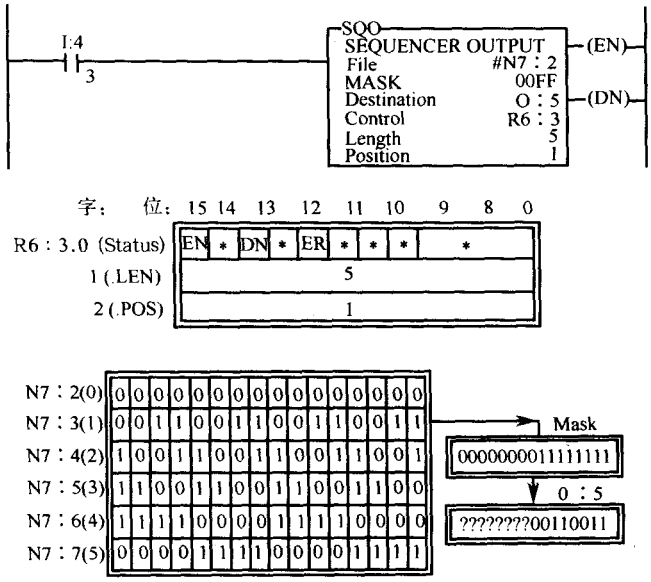


图 7-8 Allen-Bradley 顺序器输出指令、控制元素及顺序器输出字的一个典型文件

2）长度号，指在连续序列输出数值中有多少数据。存储器中的数据字文件必须比该长度号多一个数据字，因为第一个数据字（字 0）不被视为标准序列的一部分。SQO 指令仅当顺序器位在 0 位且控制逻辑在 PLC 切换到运行模式时为真时将数据输出到字 0 中。在顺序器将其最后一个数据字输出到顺序循环后，下一个应当输出的数据字会是字 1，而非字 0。

3）掩模必须以十六进制数值输入。掩模显示了目的地址中的 16 位的哪些位会在顺序器进行操作时改变，以及哪些位不受顺序器影响。16 位二进制掩模中的一个二进制 1 表示 SQO 指令会在目的地址中的相应位置写入信息覆盖该位。

4）目的地址，是顺序器输出数值写入的位置。一般要使用输出映像地址。

5）控制元素是必须的，这样 PLC 可以跟踪顺序的一系列输出字的位置。

在 SLC 500 例子中，如图 7-8 所示，I：4/3 仅在第一次打开，因此 SQO 指令会通过掩模（HEX 00FF）将数据字 1（N7：3，而非 N7：2）中的内容复制到目的地址（O：5）中，该掩模允许 SQO 指令改变目的地址的低 8 位。而目的地址的高 8 位不会由 SQO 指令改变。只要输入逻辑保持为真，那么每个扫描循环 N7：3会被复制到 O：5中。当 SQO 指令的控制逻辑为假时，位数值（在 R6：3. POS）会变为 2，同时 SQO 指令会停止向输出地址中写入。当控制逻辑重新为真时，地址 2（N7：4）中的内容又会被复制到输出地址中。

顺序器加载（SQL）指令也是可用的，因此用户程序可以将数据放入连续的数据字中，每次 SQL 的控制逻辑为真时放入一个。当然，SQL 指令包括了对源数据地址而非目的地址的规范。SQL 指令最初是用来将数据字放入文件中的，该文件可以由另一个顺序器指令使用，但是 SQL 指令可以用来从任何源存储器区域中获取数据并存入连续的存储器区域中。

PLC-5 提供了顺序器输入（SQI）指令。SLC 500 提供了相似的指令但是称为顺序器比较（SQC）。这些指令都是用来控制 SQO 指令的，当我们在本章中学习文件比较指令时便会看到它们的用处了。

7.5.2 SLC 500 专用：顺序器差分和交换指令

SLC 500

当用 SLC500 PLC 中的顺序器指令编程时，必须为掩模输入一个地址，并且也可以为屏蔽位和目的地址使用变址寻址。较高级的 SLC 500（SLC5/03 及以上版本）提供了交换指令（SWP），可以用来将位、整数、ASCII 码及字符串文件中一系列 16 位存储器区域的高位与低位交换。该指令使用起始地址和需要操作的字元素的序号进行变址寻址来编程。

7.5.3 PLC-5 专用：块传送指令

PLC-5

在 PLC-5 系统中，智能 I/O 模块被称为*块传送模块*，因为 CPU 与 I/O 模块交换数据块，用以响应用户程序中的*块传送读*（BTR）指令或*块传送写*（BRW）指令。一个 BTW 指令如图 7-9 所示。块传送指令使得 PLC 将数据交换请求传送到 CPU 模块的通信处理器中。在进入下一个指令之前，PLC 不需要等待通信处理器结束与 I/O 模块要求的数据的交换[⊖]。通信处理器处理与 I/O 模块之间的数据交换，一次交换一个数据，按照它从程序中接受请求的先后顺序排序。由于块传送指令发送的数据交换请求并不一定要在同一个扫描循环中完成，通信处理器必须提供可寻址状态信息给主微处理器，这样用户程序才能编写成在数据交换结束后能做出合理的反应。如果程序在一个较早的块传送完成之前，发送了第二个块传送请求给 I/O 模块的话，通信错误便会发生。下面在图 7-9 中介绍了块传送读指令以及状态信息的交换：

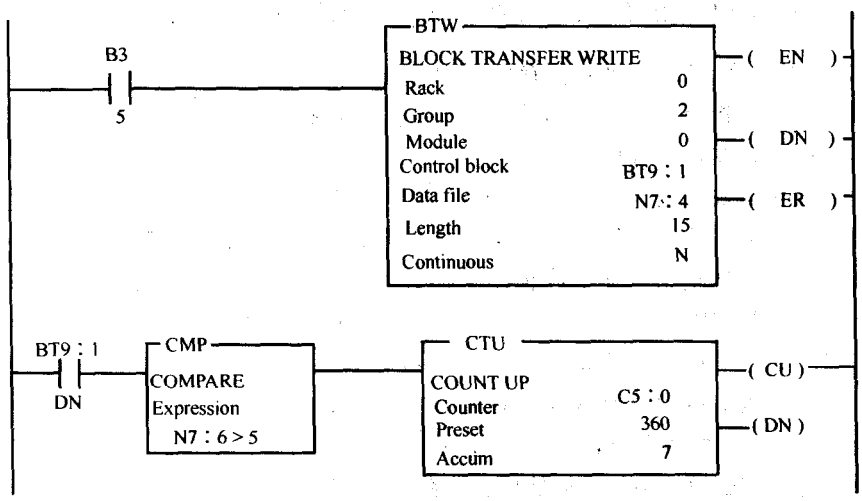


图 7-9 PLC-5 编写的使用块传送读指令来从智能 I/O 模块中读取数据的程序以及对这些数值中读取超过数值 5 的次数进行计数

1）每次 BTR 控制逻辑由假变为真时，BTR 指令会执行一次（发送一个请求给通信处理器），因为程序员在输入 BTR 时对于连续操作选择了 N（no）。如果程序员输入 Y（yes），则 BTR 指令会在每次数据组从 I/O 模块成功读出时，自动初始化一个新的读请求，只要

⊖ 当 BTR 和 BTW 指令用在中断程序中时，PLC 会等待它们的执行，我们将在 11 章看到。

BTR 指令的控制逻辑保持为真。

2）用以读取的 I/O 模块的位置是由框架（本例中的 rack0）、组（2）以及模块（始终为 0，除非当右边插槽为模块 1 时，框架为每组的两个插槽配置）来指定的。框架/组/模块寻址有时被视为一个 RGM 地址。

3）数据文件地址是通信处理器准备将其从 I/O 模块（图 7-9 的 N7：4）读入的数据存入的连续存储器区域的第一个地址。#标记符不需要。程序员必须输入数据块的长度来读取 16 位数据字。（图 7-9 中指定了 15 个字，因此最终 15 个数据字会由通信处理器复制到地址 N7：4至 N7：18 中。）如果是块传送写（BTW）指令，则通信处理器会从这些地址中复制数据字进入 I/O 模块中。程序会确认在初始化 BTW 数据传送请示之前这些地址中是否存在有效数据。

4）与 CPU 模块存储器中的文件处理使用的简单的三字控制元素不同的是，六字块传送控制元素结构体（图 7-9 中的 BT9：1）为一个控制块而指定。该 6 个字保存了 I/O 模块地址和 CPU 存储器地址，并且在处理器执行与 I/O 模块的数据交换时，通常由通信处理器来保存工作数据和状态位。程序员需要使用（至少）块传送控制元素的 DN（完成）位。（其他状态位在 13 章中有涉及）用户程序可以检查 DN 位来决定何时通信处理器响应 BTR 指令，成功读取了数据（或者响应 BTW 指令而写入数据）。如果 DN 位未开，则数据交换未完成，而用户程序不会从存储器中读取数据，任何时候这些存储器都可以被通信处理器在写覆盖。（DN 位在 BTR 或者 BTW 指令请求一个新的数据交换时会回到低状态）图 7-9 包括了一个梯形图梯级来计算从智能 I/O 端口获得的数值大于 5 的次数。只有在块传送控制元素的 DN 位（BT9：1. DN）为高时才执行，表示一个数据块已经从 I/O 模块中读取。

S5

7.5.4 SIEMENS STEP 5 数据集的移动指令（带有传送和接收功能块）

S5 PLC 不提供在 PLC 标准数据存储器中移动整个数据数组（或者数据块）的指令。程序员必须编写有多个指令的程序来移动数据字组。处理指令（DO）、间接处理（DI）指令及程序化编程技巧可以在这类程序中使用。

预编程序的功能块在较大的 S5 系列 PLC 中用来在 CPU 存储器与智能 I/O 模块的存储器之间移动数据字组（数组，尽管 S 系列不将其称为数组）。传送函数（FB 244）和接收函数（FB 245）可以跳转至与智能 I/O 模块交换数据组。每次 PLC 进入运行模式时，同步函数（FB 249）可以与 CPU 以及与 CPU 进行大量数据交换的智能 I/O 模块同步执行一次。控制函数（FB 247）可以用来确定带有智能 I/O 模块的通信状态。复位函数（FB 248）可以用来中止一个智能 I/O 模块函数和与其相关的数据交换。

与智能 I/O 模块交换数据时，PLC 通过 CPU 模块中的块共享存储器，从（向）智能 I/O 模块复制数据帧。CPU 模块中的数据处理芯片在 CPU 数据存储器与 CPU 共享存储器之间复制数据，也在 CPU 共享存储器与 I/O 模块存储器之间复制数据。除了它们由上述的功能块程序分配任务以外，数据处理器在 PLC 主微处理器上独立工作。传送和接收功能块告知数据处理器需要从（向）CPU 的数据存储器的什么地址复制数据。太大而放不进共享存储器空间（或者智能 I/O 模块）的数据帧可以由数据处理器分拆成几个较小的数据组，一次传送一组。或者，用户程序可以使用一系列连续的带有传送和接收的函数来传送大数据帧的部分。当被分配任何数据传送任务时，状态信息由数据处理器保存，同时状态位可以由用户程序使用控制功能块来读取。

同步函数（FB249）调用是指定的参数包括：

1）SSNR（接口序号），两字节常数，通常为 KY 0,y 格式。“y”出现在低字节，为了响应在系统设置时分配给智能 I/O 模块的接口序号，必须为 0 到 255 之间的数值。接口序号包括在所有的写入共享存储器块的数据组中，因此数据处理器可以确定究竟它是为哪一个智能 I/O 模块服务的。如果任何一个不为 0 的数字输入到接口号码的高字节，则指明是间接寻址。带有智能 I/O 模块的通信间接寻址在本书中未涉及。

2）BLGR（帧大小），两字节常数，通常为 KY 0,y 格式。“y”指定了每次扫描循环中的 I/O 模块可以用来交换的最大数据量（参考使用手册中指定智能 I/O 模块允许的代码及帧大小）。

3）PAFE（错误字节），在功能块不能正常工作时，指功能块在哪应该写入一个字节错误代码的地址。标志位通常也会被设定。

复位函数（FB248）调用时指定的参数包括：

1）SSNR 和 PAFE，同上。

2）A-NR（工作号），两字节常数，通常为 KY 0，y 格式。“y”是在 1 到 233 之间的数值，用以指示哪一个智能 I/O 模块操作需要复位或者哪一个通信工作需要取消。每一个智能 I/O 模块带有一个指定它可以执行的工作及其工作序号的操作手册。

控制函数（FB247）调用时指定的参数包括：

1）SSNR，A-NR，PAFE，同上。

2）ANZW（工作状态字），功能块可以写入第一个状态字的两位数据字（四字节）的地址。工作状态字可以保存在标志位存储器中或者保存在当功能块跳转时打开的数据块中，因此该参数要么以 FWx、要么以 DWx 形式输入。“x”是标志字或者数据字的序号。这些功能块通过写入两个标志位或者数据字的方法，报告由 SSNR 参数指定的 I/O 模块的通信请求的状态。高字保存了目前为止数据字交换的数目，而低字包含了下列重要的状态字：

0 和 1：指示了通信处理器 I/O 模块的状态；

2：当数据处理器成功完成了由成对工作状态字指定的功能块的工作请示时，置位。用户程序监视该位，以确保有效的数据处于交换状态中；
当带有工作状态字的功能块要求数据处理器执行新工作时立即复位；

3：当数据处理器完成了工作请求但在执行过程中发现错误时置位；
当带有工作状态字的功能块要求数据处理器执行新工作时立即复位；

4：当数据帧响应传送或者接收功能块请求，在 CPU 模块与智能 I/O 模块之间传送时置位。传送一个大的数据帧可能需要几个扫描循环；
数据帧全部传送完毕后复位；

5：当一个由传送函数请求的数据传送完成时置位。用户程序不应改变输出数据或者初始一个传送指令给智能 I/O 模块，直到该位置位为止。
在被读取后由用户程序复位，或者当下一次带有相同工作状态字的传送指令执行时复位。

6：当一个由接收函数请求的数据传送完成时置位。用户程序不应对输入数据求值，直到该位置位为止。
由用户程序复位，用来确定数据的接收，或者当下一次带有相同任务状态字接收指令执行时复位。

7：置位时（由用户程序），传送和接收请求被拒绝。也可以在用户程序读或者写新的数据时保存数据存储器区域。

8-11：包含描述导致位 3 被置位的通信错误的代码的状态位。

12-15：不使用。

传送与接收函数，实际上是导致了数据字数组在 CPU 存储器与智能 I/O 模块存储器之间复制的功能块。这些功能块可以由条件跳转（JC）指令来初始化，如图 7-10 所示，这样仅当数据传送确实需要时它们才会初始化一个数据传送请求。当程序中遇到一个跳转到传送或接收函数指令的条件，并且条件跳转的控制逻辑条件为非时，函数不能完全被忽略：PLC 会更新传送或接收操作的状态字，这样其后的检查状态位的下一梯级可以正常工作。如果传送/接收函数在每个扫描循环中都出现的话，控制函数（FB247）便不需要更新状态字[⊖]。

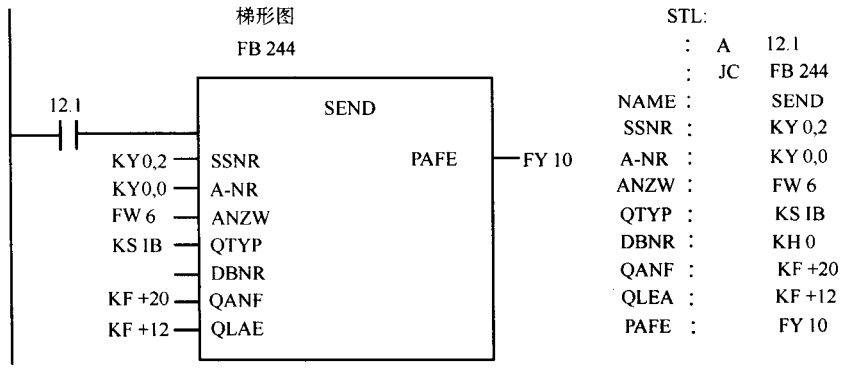


图 7-10 STEP 5 有条件跳转到数据传送功能块的指令

传送功能块（FB244）指定的参数包括：

1）SSNR、ANZW、PAFE，如上所述。如图 7-10 所示，数据被传送到由接口 2 配置的 I/O 模块中。状态数据被保存在标志字 6 和 8 中（标志字节 6 到 9）。如果提供给功能块的数据有错的话，则错误代码会被返回至标志字节 10 中。

2）A-NR（工作序号），在此处的意义与复位函数中的意义有所不同。仍然需要输入一个 2 字节常数，以 FY 0,y 的形式，但此时如果“y”=0（如图 7-10 所示），则数据处理器会被指示来传送整帧的数据，即使数据处理器芯片需要将其作为一系列子集来传送。如果“y”是介于 1 和 255 间的数值（例如，15），那么意味着传送函数正在请求数据处理器仅传送较大数据帧的该部分（15 部分）。

3）数据源指定使用下列几种参数，包括：

- QTYPE（数据源类型），必须以双 ASCII 码常数形式输入，通常以 KS xx 的形式。“xx”必须是以下字母和数字混编代码中的一种：DB、QB、IB、FB、TB、CB 或者 AS[⊜]。表示准备传送的数据应当是从数据块中复制出的数据（DB）；或者从输出、输入及标志位存储器区域中复制的字节（QB、IB 或 FB）；或者是从定时器、计数器存储器区域（TB 或 CB）或将被指定为绝对地址（AS）的地址中复制出的字。
- DBNR（数据块序号），以双字节常数输入，通常以 KY 0,y 的形式。“y”必须是数据块序号，同时当 QTYPE 参数代码为 DB 时可忽略该序号。
- QANF（源数据帧的起始地址），以一个 16 位有符号二进制常数形式输入，通常以

⊖ 如果发送和接收指令作为中断程序的一部分或者在初始化程序中被执行，或者如果它们在不经常执行的结构化程序中的一部分，这时需要一个“控制（control）”函数。

⊜ 另外还有三个可能的代码可以用来说明数据源将用间接寻址来指示，本书中没有讲到。

KF x 的形式。“x”是数据字、输出字节或者其他包含了将要传送的数字值的第一个连续单元的数字值部分。关于绝对地址，参考使用手册。

- QLAE（源数据帧长度），以一个 16 位有符号二进制常数形式输入，通常以 KF x 的形式。“x”是即将传送的数据单元的序号（QTYPE 同样也定义数据单元为字或字节），包括从起始地址及起始地址之后的连续存储器区域中的数据单元。

图 7-10 中的程序指定了被传送的 12 个输入字节，从 IB20 到 IB31。

接收功能块（FB245）指定的参数包括：

1）SSNR、ANZW、PAFE，如上所述。

2）A-NR，如传送函数中所述。

3）数据目的参数，指接收函数会将其接收的数据值放到何处。该参数与其他的传送函数参数相类似。ZTYPE、DBNR、ZANF 和 ZLAE 参数与前述的 QTYPE、DBNR、ZANF 及 ALAE 有相同的含义，除了它们有不同的名字来指示目的地址，而非源地址。

S7

7.5.5 SIEMENS STEP 7 数据集的移动（使用系统函数）

S7 PLC 系列不提供独立的指令来移动数据项的数组，尽管一个数组类型的数据项可以在 STEP 7 中定义。由用户编写的程序必须使用存储器间接寻址、寄存器间接寻址以及结构化的编程技巧。

S7 PLC 系列包含了一些已经编写好的系统函数和系统功能块[⊖]，它们可以由用户程序来调用，用以移动数据集或者执行顺序程序。调用是无条件执行的。（结构化的编程技术可以导致 PLC 绕过一条调用指令。）某些在单一 S7PLC 系统中移动数据集的系统函数包括：

1）SFC 20(BLKMOV)，将 CPU 数据存储器中的任何一个区域的数据块（除去数据块组、计时器数据或者计数器数据）复制到 CPU 存储器的另外一块区域中（除去数据块组、计时器数据或者计数器数据）。在 STL 中，带有所需要的输入和输出参数的完整调用，是如下的形式。

```
CALL SFC 20
SRCBLK    := x
RET_ VAL  := y
DSTBLK    := z
```

在这个调用的例子中，“x”是一个任意型指针，指定了即将移动的数据元素的位置和数量（源块），“z”是另一个任意型指针，指示了数据将要移动的目的地（目标块）。“y”是一个字地址，如果不能复制数据，则由 BLKMOV 写入一个错误代码。可以输入符号名称。如第 5 章中所知，一个任意型指针常量都可以按如下格式输入：

P#Memory _ prefix _ Byte. Bit Data _ type Number _ of _ items

例如：P#M20.0 Word 5，表示：由 MB20 起始的 5 个字（10 字节）

2）SFC 21（FILL），复制小数据块中的内容（BVAL 输入参数必须为指向该小数据块的任意型指针），不断重复操作直到它填充了一块较大的存储器块（由 BLK 任意型输入参数作为指针指向该存储器块）。

⊖ 系统函数在第 8 章有详细介绍。附录有系统函数的完整列表。

3) SFC 79 (SET) 与 SFC 80 (RSET)，I/O 存储器区域中的置位或复位。为 SA 参数输入一个指针，指示从哪儿开始置位或者复位，同时为 N 参数输入一个整数值来指示有多少位被置位或者复位。

4) SFB 32 (DRUM) 是一个顺序器。由于它是一个功能块程序（不仅仅是一个 SFC)，它使用立即数据块。在下述的每一个立即数据块中的静态 VAR 变量都必须在 SFB32 使用之前赋值。

- OUT _ VAL，二维的位数组（最大 16 位，通过 16 步)。必须为每一个最大为 16 步的并且最大为 16 DRUM 的输出包含布尔值。作为指针来输入到布尔元素数组中。
- S _ MASK，关于屏蔽位的二维位数组。屏蔽位中的 1 允许将从 OUT _ VAL 中获得的数据位输出。作为指针来输入到数组中。
- DSP，在复位后启动的 OUT _ VAL 步的序号。输入一个 1 到 16 之间的数。
- 如果需要基于时间的顺序，则必须为 S _ PRESET 和 DTBP 输入参数。S _ SET 作为每一步的时间延迟数组的指针输入（最大 16 个字)，DTBP 则作为一个指示使用的时基（以毫秒计）的字来输入。

其他参数也可以在被调用时传送至“DRUM”。如果这些参数未能传送，则它们会保留上一次 DRUM 使用这个立即数据块执行操作后它们的数据值（可能是使用上述静态变量赋的默认值)。输入参数包括：

1) DRUM _ EN，布尔输入，用以选择是基于时间的顺序（若 DRUM _ EN 为 1 时）还是基于事件的顺序。

2) JOG，布尔输入，在布尔控制逻辑为真时，使得基于事件的顺序步进执行。

3) EVENT1 至 EVENT16，布尔输入，如果 DRUM _ EN 为 1，为每一个可能的 16 位输出步启动计时器。对于纯粹基于时间的顺序，这些位必须始终为开。若在基于事件的时序中使用时间延迟，则布尔逻辑可以将这些位开启，以在每一步执行时启动时间延迟。

4) LST _ STEP，返回步 1 之前指示最后一个允许执行的步的序号的字节（最大为 16)。

5) RESET，用以复位顺序器的布尔输入。见下述的 Q 参数。

输出参数包括：

1) OUT _ WORD，输入一个 DRUM 写 16 位顺序器输出的地址。

2) OUT1 至 OUT16，输入 DRUM 能写独立的布尔输出的位地址，指示 OUT _ WORD 的 16 位。

3) Q，布尔位，最后一步输出时置位，同时指示 DRUM 在其能够继续改变输出值之前必须复位。

4) ERR _ CODE，字，当 DRUM SFB 出错时，DRUM 能够将错误代码写入其中。

许多系统函数都是为了读或者写系统存储器区域或者 I/O 模块配置数据存储器中的信息，因此 PLC 程序可以响应和改变其自身的操作环境。这些 SFC，我们在第 10 章中会再次见到它们，包括：

1) SFC51 (RDSYSST)，从系统数据存储器区域中复制一个系统状态列表集合（包括状态信息）到数据存储器中（M、L、D、I 或者 Q 存储器区域)。在 STL 中，对 RDSYSST 的调用看起来像图 7-11 中的例子一样。

SFC51 (RDSYSST) 需要的输入参数有：

- REQ 输入参数，布尔型。尽管 RDSYSST 被称为是无条件的，但如果 REQ 输入位为

非（例如，如果图 7.11 中的 I2.1 关闭），那么 SFC 不会复制任何数据。

- 两个参数，SZL _ ID 及 INDEX，为 16 位数据字，包含了能够告知 PLC 哪些状态列表代码从系统数据存储器中复制出来。（如图 7-11 所示，用以从系统存储器的错误诊断区域中读取最近 15 个记录的代码）

输出参数包括：

- RET _ VAL 地址，当 RDSYSST 出错不能正常工作时，会将错误信息写入该地址中。
- BUSY，布尔位，RDSYSST 复制数据时将其置位。
- SZL _ HEADER，双字结构体，RDSYSST 会告知它从系统存储器区域中读取的数据记录的字节数，以及它已经读取的数据记录的个数。如图 7-11 的程序执行后，MYSTRUCT.0 会包含 300，表示有 300 字节被读取，同时 MYSTRUCT.1 会包含 15，因为 15 个记录已经读取。

```
CALL SFC51
REQ             := I 2.1
SZL_ ID         := W# 16# 01A0
INDEX           := W# 16# 000F
RET_ VAL        := MW10
BUSY            := Q 7.1
SZL_ HEADER     := MYSTRUCT
DR              : = P # M40.0 Word
150
```

图 7-11 STEP7 调用 SFC51 来从系统存储器中复制状态数据块到数据存储器中

- DR，一个任意型指针，告知 PLC 将从系统存储器区域中读出的数据放在何处。在图 7-11 中，从 MW40 起始的 150 个存储器字被使用，而每一个错误记录有 10 个字长。

2）SFC58（WR _ REC），用来复制数据块而不是配置数据到智能 I/O 模块中。图 7-12 中举出一个使用 WR _ REC 的 STL 例程。

SFC58（WR _ REC）需要的输入参数包括：

- REQ，布尔输入，在其为真时允许 WR _ REC 将数据块传送至智能 I/O 模块。
- IORD，单字节代码，确定在外部输入存储器区域（IORD=W#16#54）或者外部输出区域（IORD=W#16#55）中的数据的目的位置。
- LADR，确定智能 I/O 模块的逻辑地址的整数值。在图 7-12 中，该数据会写入 I/O 模块的 PQ128 地址中。
- RECNUM，确定该数据集记录数量的整数，决定了智能 I/O 模块将数据集放在其存储器中的位置。

```
CALL SFC 58
REQ             := I 2.1
IORD            := W# 16# 55
LADDR           := 128
RECNUM          := 4
RECORD          : =  P #  MW60.0
Word 5
RET_ VAL        := MW56
BUSY            := M 55.0
```

图 7-12 STEP7 调用 SFC58 将数据块复制到智能 I/O 模块中

■ RECORD，一个任意型指针，描述了将要复制到 I/O 模块中的数据的位置和大小。

有两个输出参数必须被指定。在 RET _ VAL 地址中：

■ WR _ REC 在出错时会写入一个单字的错误代码，并且 BUSY 是一个位；

■ WR _ REC 在向目的智能 I/O 模块中复制数据块时为真。

3) SFC59（RD _ REC）可以用来从智能 I/O 模块中读取数据块。参数分配与 SFC58（WR _ REC）相同。如果 RECNUM 参数有一个超过 1 的值，那么它就是 I/O 模块中读入的用户数据，而不是状态信息。

4) SFC52（WR _ USMSG）被用来将一个用户自定义的单字长错误事件 ID 号码（作为 EVENTN 输入参数提供）和三字长的用户自定义的错误信息（在 INFO1 及 INFO2 输入参数中）复制进入系统存储器的诊断缓冲器的记录中，作为标准的 10 字诊断信息的一部分；或者将 4 字输入数据传送至外围设备中，例如操作面板中。该系统函数在第 10 章中讨论。

5) SFC55（WR _ PARM）、SFC56（WR _ DPARM）、SFC57（PARM _ MOD）以及 SFC59（RD _ REC）将包含配置数据的数据块写入智能 I/O 模块中，或者从 I/O 模块中读取状态信息块。在第 10 章也有讨论。

CQM1 7.5.6 OMRON CQM1 文件、数组和结构体移动指令

CQM1 从不使用数组这个术语，有一个明确的指令分类方法，可以移动位、半位元组（OMRON 称为数字）、字节和字的连续数组。某些指令可以在 16 字数组的行和列中转换。其他的指令也是特别设计用来从输入设备中读取数值进入字节数组中，或者可以从字节数组中复制数值进入显示设备中。还有其他一些指令（例如 MSG(46)、TXD(48) 及 RXD(49)）用来通过通信网络移动数据单元的数组，这些在第 13 章中有讨论。

1) 块传送指令 XFER(70)，最多可以复制 9999[⊖]个数据字（第一个参数 N，必须为 0000 至 9999 之间的 BCD 数），从起始于起始源字地址的存储器区域（S，第二个参数）至起始于起始目的字地址的存储器区域（D，最后一个参数）。除非使用差分形式，XFER(70) 指令在其控制逻辑为真时，每次扫描循环都会执行一次。图 7-13 中展示了该指令的使用。本例中，每次 PLC 进入运行模式时（当第一次循环标志位 SR315 置位时），24 个数据字从永久存储存储器地址 HR10 至 HR33 中复制到存储器区域 DM1000 至 DM1023 中。



图 7-13 OMRON 复制数据值数组的 XFER(70) 指令

⊖ 当然，源地址和目的地址的整个范围必须存在。如果任意指令试图向一个不存在的或超出允许范围的（例如，HR99 是保持寄存器区域的最后地址）的目的地址写，错误状态位（SR 25503）将打开。

2）块置位指令 BSET(71)，将源字地址（S，第一个参数）中的单 16 位字复制到从起始字地址（St，第二个参数）到终止字地址（E，最后一个参数）之间的所有存储器区域中。

3）数字移动指令 MOVD(83)，从源字地址（S，第一个参数）中将所有的或部分的四个半位元组（即一个 16 位数据字）移动到目的字地址中（D，第三个参数），同时在该过程中将数字（半位元组）移位。数字指示值（Di，第二个参数）以 BCD 码指示了源数字中哪一个最先移动（Di 的低四位为 0 至 3），目的字中哪四位需要移入（Di 的 8 至 11 位为 0 至 3），以及有多少附加的数字需要移动（Di 的次低四位为 0 至 3）。最高四位设置为零。

4）传送位指令 XFRB(-)（仅在 CQM1-CPU4X 中使用）可以最多复制 255 位的位数组至存储器中的其他区域。数组可以起始于源字指定的存储器中的任何位（S，第二个参数），并且可以扩展到更高的地址中。它也可以被复制，因此它可以起始于目的字指定的存储器中的任何位（D，第三个参数），并且可以扩展到更高的地址中。一个控制字（C，第一个参数）必须在其低四位半元组中包含一个数值来显示源字中的起始位（十六进制数值 0 至 F 代表源位号 0 至 15 位）。第二个四位中的值指示目的字中的起始位（0 至 F），并且高字节中的值指示有多少位要拷贝（00 至 FF 代表 0 到 255 位）。

5）两条指令在 16 字数组的列与单 16 位字（称为行）的位之间复制 16 位数值。行列转换指令 LINE(-) 指令，将第一个起始于源地址（S，第一个参数）的 16 个数据字中的一位，并且扩展进入更高编码的地址中，复制到一个单一的目的字地址中（D，第二个参数）。一个列位指示值（C，第三个参数）显示了从源字（输入一个 BCD 数值，在 0000 与 0015 之间）数组中的哪一位复制该位。最低序号的地址为目标地址提供了 0 位。COLM(-) 指令，执行了相反的操作：将一个 16 位数据字复制入一个 16 字长数组的一列中。

6）数字开关输入指令 DSW(87)，将 4 位或者 8 位中的标准数字开关集合读入 4 个或者 8 个半位元组数组中。输入字地址（IW，第一个参数）和输出字地址（OW，第二个参数）指在对开关位置译码时使用哪一个输入和输出模块。输出模块的低四位在读取过程中会顺序开关，一次驱动一个数字开关（如果读取 8 位开关则一次驱动两个）。在驱动时，数字开关输出一个通过输入模块的低四位读取的四位 BCD 码（如果读取 8 位开关，则同时也会通过后续四位输入触点）。每四位结果保存在由第一个寄存字（R，第三个参数）指定的地址的四位中，如果读取 8 位开关，则同时也会存入后续四位地址中。状态位 SR25410 在 DSW（87）执行操作时置位。一次操作占用 12 个扫描循环。只要控制逻辑保持为真，读取操作就一直循环执行。

7）七段码显示输出指令 7SEG(88)，从存储器中复制半位元组数据（用十六进制或者 BCD 字符表示）至输出模块中，该输出模块可以连接到七段码显示芯片上。4 位（或 8 位）独立的显示芯片是可写的。7SEG(88) 将 4 位数值，一次 4 位（或 8 位），复制入输出模块的低 4 位（或低 8 位），接着反复执行直到全部 4 位（或 8 位）都已经写入显示芯片中。7SEG(88) 也可以在输出模块输出四位数值时，顺序将输出模块的接着的较高的 4 个端口转换以选择当前 4 位半位元组将要作用的显示芯片（或者芯片对）。4 位半位元组的值从第一个源字（S，第一个参数）所指定的地址中复制出来（如果读 8 位开关则为 8 位），送入到由输出字指定的（O，第二个参数）输出映象地址中。一个控制数据数值（C，第三个参数）必须包含一个代码指示究竟是输出 4 位还是 8 位字符，以及是否有必要将 4 位半位元组数值或者芯片选择状态转化为显示硬件（参见参考手册中的代码）。状

态位 SR 25409 和一个输出映像位在操作时需要的 12 个扫描循环中持续为高，当控制逻辑持续为真时写操作反复执行。

7.6 文件、数组和结构体的比较

一些 PLC 提供了可以将整个数据数组与其他数值数组，或者与其他单个数值比较的指令。单独的比较结果可以用作控制其他指令。由顺序输出位模式（本章中前部分讨论过）控制的生产过程通常由等待输入正确的状态来控制需要的顺序。对每一个顺序执行的输出步来说，在下一个输出步被初始化之前，都有一个必须与之相匹配的输入状态模式。

PLC-5
SLC 500

7.6.1 ALLEN-BRADLEY 文件比较指令

PLC-5 顺序器输入（SQI）及 SLC 500 顺序器比较（SQC）指令非常相似，但并不是完全相同。两者都是将源地址（通过屏蔽位）中的数值与顺序执行文件中的数据字集中的一个数值进行比较，并且两者都仅当其控制逻辑为真时执行比较指令。屏蔽位指示了源地址中的哪一位必须匹配顺序执行文件字，因为该数据字被认为是相等的。16 位屏蔽位中的 0 位表示在该位中并没有发现任何匹配。

在 SQI 和 SQC 指令中，输入文件的第一个数据字（字 0）的第一个数据字被用作启动模式，而当下一次顺序器重新循环执行时也不会包含在比较序列中。如果程序员想要重新使用第一个数据字，则必须使用另一条编程指令（例如 MOV）来将数值 0 写入 SQI 或者 SQC 的控制元素的.POS 字中。下面会解释其中不同。

1. PLC5 的顺序器输入

PLC5 的 SQI 指令用在布尔逻辑条件中。SQI 指令将输入模式与它的控制元素指向的每一个字进行比较。如果相等，则 SQI 指令为真，否则为假。SQI 指令不会将其控制元素中的位置字.POS 增加。控制元素中的.POS 值必须被改变来增加指针值。可以由另一条指令选择一个新的输入文件值与源地址中的数值比较改变.POS 值（例如 MOV 或者 CPT 指令）。

顺序器输出（SQO）指令（在前面讨论过）在每次控制逻辑为真时会自动增加它的控制元素中.POS 的数值，因此当 SQI 指令被用来控制 SQO 指令时，它们有相同的控制元素，这样 SQO 指令就会同时增加 SQI 与 SQO 的操作步。图 7-14 展示了一个当 SQO 顺序步进，反过来增加 SQI 顺序时，SQI 是如何控制的。两种指令都在图中的位置 2（R6：3.POS=2），因此 SQO 指令正在将其输出文件（N7：2+2=N7：4，由十六进制屏蔽位 00FF 来输出）中的字 2 的低 8 位数值写入目的地址（O：124）中。SQI 指令将源地址（I：123）的低 12 位数值（屏幕位=十六进制的 0FFF）与其输出文件（N7：20+2=N7：22）中的字 2 进行比较。当 SQI 产生一个真值（当 I：123=N7：22）时，SQO 指令会增加 R6：3.POS 的值同时从数据文件中输出下一个字（N7：5），同时 SQI 指令会开始比较源数值与输入文件中的下一个数据字（N7：23）。这是仅有的两条指令可以分享的相同控制元素的部分（除去 FIFO 与 LIFO 指令）。

2. SLC 500 顺序器比较指令

SQC 指令在 SLC 500 程序中被视作一个输出元素。本身不能对自己控制元素中的.POS

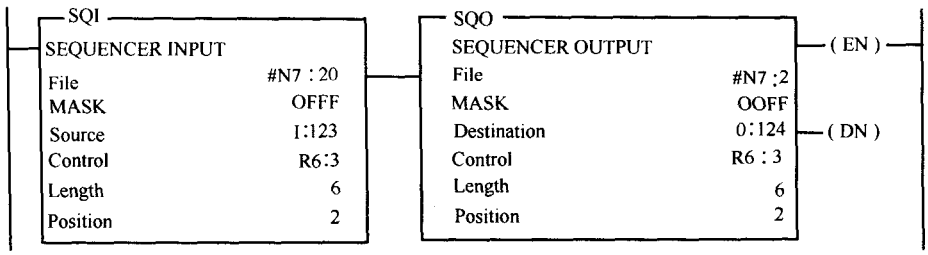


图 7-14 PLC-5 顺序器输入及顺序器输出对

字进行增量操作，仅当其控制逻辑由假变为真时可以执行。由于一个输出元素不能在布尔逻辑条件中使用以控制另外的输出元素（即便是控制一个顺序器输出指令），一个附加的状态位便加在了 SQC 的控制元素上。FD（found）位在源输入类型与被比较的顺序文件字类型相匹配时置位，不匹配时复位。FD 位可以由执行 SQO 与 SQC 增量操作，从而使得它们同时增量的布尔逻辑条件检查。为了在 SLC 500 中工作，图 7-14 中的 PLC-5 程序必须修改成如图 7-15 中那样。

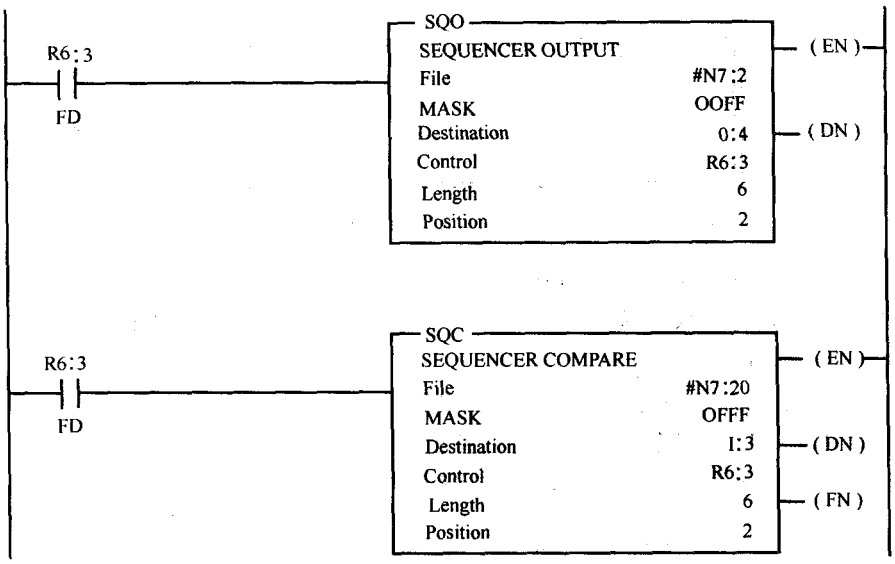


图 7-15 SLC 500 带有顺序器输出的顺序器比较指令

PLC-5

7.6.2 PLC-5 的文件搜索与比较（FSC）、文件位比较（FBC）以及诊断检测（DDT）

PLC-5 提供文件搜索与比较指令（FSC），将文件中的数据字集合与另一个文件中相应的数据字进行比较，或者将文件中的每一个字逐次与作为常值或者地址的输入数据值进行比较。当寻找到匹配时，比较操作停止，指令的控制元素中指示发现匹配的状态位置位，可参看图 7-16 程序中的 FSC 指令。必须和 FSC 指令一起输入的数值包括：

1）长度，指示需要进行比较的字的个数。

2）表达式，与在简单比较指令（CMP）中输入的表达式类似，除了 FSC 表达式中的地址可以加上 # 标记符前缀。该标记符表示此地址为文件的第一个地址，此文件可以逐次逐位

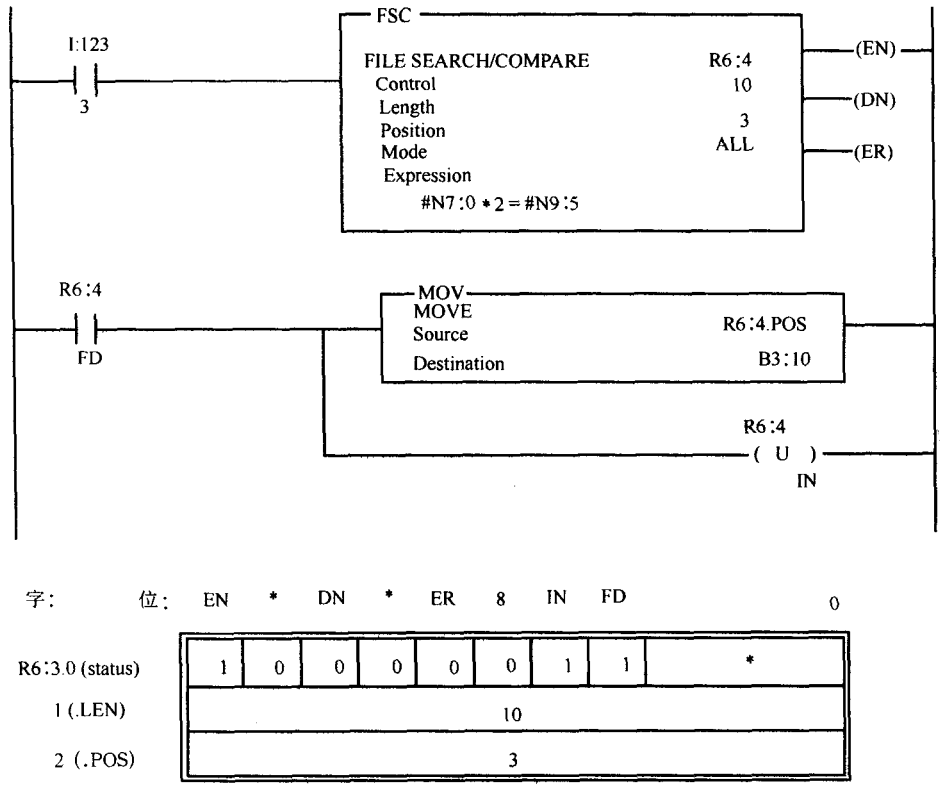


图 7-16 PLC-5 文件搜索及比较指令和它的控制元素

被检查，同时允许 PLC-5 在变址寻址中使用 S：24。在图 7-16 范例中，N7：0被乘以 2，然后其结果与 N9：5进行比较，接着 N7：1进行乘法运算后再与 N9：6进行比较，如此下去直到 10 次比较全部做完。

3）控制元素，用来跟踪哪些文件的数据字已经完成比较操作。在使用一般的 EN、DN 及 ER 位的基础上，FCS 指令还使用以下两种：

- FD （发现），每次发现表达式为真时锁定，表达式为假时打开。
- IN （阻止），当 FD 位置位时锁定，只要 IN 置位，FSC 指令就会停止执行，因此使得表达式为真的文件中字的位置会保存在控制元素的 .POS 字中。程序中解锁的 .IN 位允许 FSC 指令继续比较操作。

4）下列类型的一种模式：

- ALL 指示当每次 FSC 的控制逻辑由假变为真时，所有的比较操作都执行了一次。如果寻找到匹配，则比较操作停止，但复位控制元素的 .IN 位允许 FSC 指令在余下的文件中继续比较数值。
- INC 在每次控制逻辑为真时（在文件最后一个数值比较完后会重启序列）使得序列中的下一个比较操作执行。
- n n 是一个从 1 到长度数值的号码。每次控制逻辑为真时，在每次扫描循环中都会使得相应号码的比较操作执行，直到文件中最后一个数值被比较完。若想重新启动比较，控制 FSC 指令的逻辑条件必须重新由假变为真。

在图 7-16 所示的程序中，FSC 指令在每次 I：123/3 位为高时比较所有的 10 个数据值。如果寻找不到匹配，则 FSC 指令等待下一次 I：123/3 位再次为高时继续执行比较操作。如果寻找到匹配（如例中），FSC 指令就不会继续执行比较，而是将发现（.FD）与限制（.IN）状态位置位。例子中的程序的梯形图的下一步标记会因为 FD 位（R6：4.FD）置位而继续执行。该步程序会将文件中发现匹配的位置的信息（R6：4.POS 包含 3）保存在位文件数据字中，

接着复位. IN 位这样 FSC 就可以在下一个扫描循环中继续比较余下的 7 个数值。

PLC-5 的文件位比较（FBC）指令是设计用来支持控制程序调试的。实现该功能是通过将输入映像数据字与存储器中参考数据字的第二个文件（可能反映了程序员期望输入映像是什么?）进行比较，并且将任何不匹配的位置记录在第三个文件中。第 15 章中将会涉及文件位比较指令。

诊断检测（DDT）指令与 FBC 指令相同，除了在任何时刻发现不匹配，参考文件位会改变以反映出输入数据字中不匹配位的状态。因此 DDT 指令可以更好的检测输入情况中的改变，并把它们按照改变的次序记录起来。

S5
S7

7.6.3 SIEMENS 文件、数组和结构体比较指令

Siemens 的 S5 与 S7 PLC 系列并不提供任何比较数据值集合的指令或者预先编好的函数。如果需要的话，程序员必须自己编写程序实现该功能。

CQM1

7.6.4 OMRON CQM1 文件、数组和结构体比较指令

CQM1 有三个指令，可以用来将单一数据值与数组中的数值进行比较；有一个指令可以用来将一个数组中的数值与另一个数组中的数值进行比较。

1）表格比较指令 TCMP(85)，用来将一个 16 位比较数据字（CD，第一个参数）与在 ORMON 系列中起始于最低地址（TB，第二个参数）、称为比较表格的连续存储器区域中的 16 个字进行比较。当其控制逻辑保持为真时，每个扫描循环它都会对整个比较集合进行操作，将 1 写入单一 16 位结果字中（R，第三个参数）来显示与比较数据字相同的字的位置。举例来说，结果字的位 5 中的 1，代表比较表格中的字 5 与比较数据字相同。如果不同就将 0 写入结果字中。微分指令@TCMP(85) 在每次其控制逻辑为真时，都会将所有 16 个数据字与比较数据字进行比较。

2）数据搜索指令 SRCH(-)，同样也是将单一数据字与其他数据字中的数组进行比较，但是比较数据值可以与任何字的数来进行比较。N 是第一个参数，表示有多少字进行比较；R_1 是第二个参数，表示与比较数据数值进行比较的数值范围的最低地址；CD 是第三个参数，必须提供将与数值范围中的数值进行比较的单一数值的地址。数值的整个集合在每次控制逻辑为真时的扫描循环中都会执行比较操作，如果寻找到任何匹配，则 EQ 位（SR25506）置位，同时第一个匹配（最低地址）值的地址写入比较数据地址之上的存储器地址中。而该最低地址的匹配数值的位置可以由两种方法来记录：如果数值范围处于 DM 存储器区域范围中，则保存 DM 地址；如果处于其他存储器区域范围中，则保存在比较数据地址之上的存储器地址中的数是在数值范围起始处地址与发现匹配处地址之间的存储器地址之间的偏移量。

3）块比较指令 BCMP(68)，比较单一的比较数据数值（CD，第一个参数，可以为常数）来检查它是否在一个或多个 16 个数值范围之内。每一个 16 个范围都是由一个下限与上限标识的，因此 32 个上下限数值必须在执行 BCMP(86) 指令之前输入连续存储器区域中，从最低位地址中的第一个范围的下限起始，其后与相应的上限匹配，接着是下一个范围的下限，如此执行下去。输入第一个下限的地址作为第一个比较块字（CB，第二个参数）。如果比较数据数值在相应的范围之内，BCMP(86) 指令将 1 放入一个单一 16 位结果字地址（R，第三个参数）

的一位中，如果不在则放入0。例如，如果第七个范围为40至200，而比较数据值为200，那么结果字的第7位则会置位，以表示200（刚刚好）在第七个范围之内。只要BCMP(86)控制逻辑保持为真，所有的16位比较都会在每次扫描循环中执行，但是微分指令@ BCMP(86)也是可用的。

4）多字比较指令MCMP(19)，将从表格1地址（TB1，第一个参数）的第一个字起始的16个字与从表格2地址（TB2，第二个参数）的第一个字起始的16个字进行比较。如果比较数值相等，则将0（注意不是1）写入相应结果字地址（R，第三个参数）的位中。如果数值不相等，则写入1。所有16个数值在控制逻辑持续为真时的每次扫描都进行比较，除非指令以微分形式@MCMP(19)出现，它用来在控制逻辑由假变为真时，强制使得比较操作仅执行一次。

7.7 文件、数组和结构体的数学及逻辑指令

为了在数据集合中执行数学或者逻辑操作，PLC对其执行操作的数据集合中的每一个数值都重复相同的数学/逻辑操作。执行这些操作的指令有内置的循环操作，它们使用变址寻址以便于在每次重复循环时对不同地址的数据进行操作。一个好的程序员会编写执行使用循环功能、变址寻址及一次在一个数值上进行操作的数学/逻辑指令的数学/逻辑操作的函数，但有一部分PLC已经提供了相关指令，便没有必要再编写这样的循环程序了。

PLC-5 SLC 500

7.7.1 ALLEN-BRADLEY的文件数学及逻辑指令

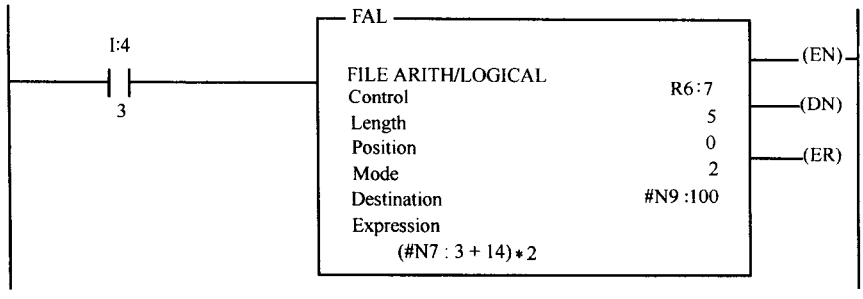
Allen-Bradley的文件数学及逻辑指令（FAL）与Allen-Bradley的计算指令（CPT）执行相同的数学及逻辑操作，同时它还可以执行使用文件中的数据操作并将结果放回文件中。如图7-17所示，FAL指令包括了一个描述数学/逻辑操作的表达式，并且指定了目的位置，这与CPT指令是一样的。但是，FAL指令可以包含带有#标记符前缀的地址，告诉PLC这是变址寻址。#标记符可以根据程序员的需要出现在表达式及目的地址中的许多地址中。像简单的CPT指令一样，FAL指令使得PLC执行一次，检查表达式同时如果没有地址带有#标记符则将结果放入存储器中，但是FAL指令接着会将所有带有#标记符的地址加1，并且重复执行该过程，按需要如此反复执行多次。程序员应当保证对于带有#前缀的每一个操作数地址，存储器中都有一个操作数数组与其对应，并且在目的数组中有足够的存储器空间保存结果数值。

图7-17也显示了程序员必须输入的附加信息，与我们在其他文件数据操作指令中的条目相似。这些信息包括：

1）控制（control）元素地址及长度数，用以指示表达式须被求值多少次。PLC使用控制单元的.POS数据字来跟踪执行计算集合操作时FAL指令执行循环的次数。

2）模式（mode）入口，在FBC指令中，可以为：

- ALL 这样全部的计算集合都会在每次FAL指令控制逻辑为真时执行。
- INC 这样仅会完成一个计算，同时指针会在每次指令的控制逻辑为真时加1来指向下一计算。
- n 从1到长度数值的序号。每次指令的控制逻辑保持为真时，在每次扫描循环中都会使得相应序号的计算操作执行，直到整个文件操作完毕。若想重新启动FAL指令，指令的控制逻辑必须先变为假，再变为真。图7-18显示了一个编写好的FAL指令在这种模式下执行的情况。



扫描数	I:4/3	这些计算执行:
1	False	None
2	True	N9:100=(N7:3 + 14) *2 N9:101=(N7:4 + 14) *2
3	True	N9:102=(N7:5 + 14) *2 N9:103=(N7:6 + 14) * 2
4	True	N9:104=(N7:7 + 14) *2
5	True	None
6	False	None
7	True	N9:100=(N7:3 + 14) *2 N9:101=(N7:4 + 14) *2
8	False	None
9	True	N9:100=(N7:3 + 14) * 2 N9 :101=(N7:4 + 14) * 2

图 7-17 PLC-5 的文件用来计算表达式及保存结果的文件算术及逻辑指令

7.7.2 SIEMENS 文件、数组和结构体的数学及逻辑指令

S5 S7

Simens 的 S5 与 S7 系列都不提供任何在数据值的集合中执行数学或者逻辑操作的指令或者预先编好的函数。程序员在需要时必须自己编写带有循环及变址寻址的程序来实现这些功能。

7.7.3 OMRON CQM1 文件、数组和结构体的数学及逻辑指令

CQM1

CQM1 仅提供了非常有限的用于操作数据值数组的指令集合。程序员必须自己编写带有循环和变址寻址功能的程序，用以在需要的时候执行其他功能。

某些检查多种数据值的指令在前面已经提到过了。如第 6 章中的 MAX(-) 及 MIN(-) 是用来从存储器中的数值数组中寻找单一的最大或者最小值的。AVG(-) 计算数据值集合的平均值，但每次扫描时独立的数值都会从相同的存储器区域中读取一次，而不是从已经位于存储器中的数据值数组中，所以在第 6 章中也提到了，但是 AVG(-) 在从单一存储器区域读取数据值时，也创建了一个数据值数组并将它们保存起来，是因为 AVG(-) 在数组完成之前是不会完成计算平均值操作的。

CQM1 在数据数组中操作的数学/逻辑指令包括：

1) SUM(-) 指令，如图 7-18 所示将从连续存储器区域中得到的字或者字节相加求和。当其控制逻辑为保持为真时，每次扫描循环都会再计算一次，除非使用微分形式@SUM(-)指令，在这种情况下每当控制逻辑由假变为真时只会再计算一次。控制字（C，第一个参数）代表了用于相加的数据值的大小、数目及类型。C 的低 12 位必须是一个介于 000 与 999 之间的 BCD 数，用以指示包含需要相加的数据地址号码。C 的高两位指示了数据值会被译码为 BCD（如果为 00）还是译码为无符号二进制数（如果为 01），再或者译码为有符号二进制数（如果为 11）。C 的另外两位（位 12、13）代表了将所有 16 位数据字相加（如果为 00），或者仅将数组中每个字的最右边字节相加（如果为 11），或者将所有字节相加（如果为 10）。接着，范围字（R1，第二个参数）代表了需要相加的数值的起始地址。最后，目的字（D，第三个参数）代表了 32 位结果将要存入的第一个双字地址。低 16 位存入指定地址中，高 16 位存入紧跟其后的地址中。

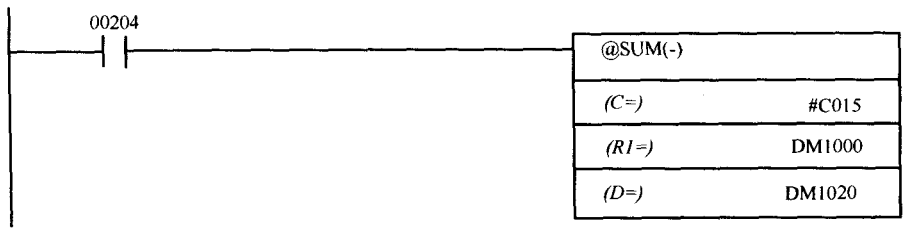


图 7-18 OMRON 的 SUM(-) 指令

在图 7-18 中，使用了 SUM(-) 的微分形式，因此每次 IR00104 由假变为真时，求和运算仅执行一次。控制参数被当作常数输入，代表 15 个数值需要求和（如果 R1 参数为 DM1000，那么便从 DM1000 至 DM1014）。从这些地址得到的数值被当作 16 位有符号二进制数值（控制字参数的高 16 位为 C，在二进制下为 1100）。32 位有符号二进制结果保存在地址 DM1020 和 DM1021 中。

2) 帧求和指令 FSC(-)，使用所有包含在连续存储器区域中的字或者字节来执行异或操作。该指令也要求输入与 SUM(-) 指令相同的参数，但是其控制字参数的高位应当始终为 00。该异或指令的结果只是一个单一的字节或者字，因此可以存储在一个单一的目的存储器区域中。微分形式@FSC(-)，也是可用的（见第 10 章，在连续通信中使用）。

有两条指令可以用来将半位元组中的十六进制数值与 ASCII 代码数组之间互相转换：

1) ASCII 转换指令 ASC(86)，当其控制逻辑保持为真时，每次扫描循环都会将最多四个十六进制数值转换为 ASCII 码。微分形式的指令@ASC（86）也是可用的。源字（S，第一个参数）代表包含了一个将被译码为四个十六进制数值的 16 位数值的地址。数字指示字（Di，第二个参数）代表了源字的哪一位最先转换（Di 的低四位中的号码 0 至 3），有多少来自源地址的附加数字需要转换（接下去四位中的号码 0 至 3），ASCII 码的奇偶（如果 Di 高四位为 0，则无奇偶，为 1 则为偶，为 2 则为奇），以及究竟将第一个 ASCII 码写入目的地址的低字节还是高字节（如果 Di 的次高四位为 0，则写入低字节，如果为 1，则写入高字节）。输出数组可以被分三种目的地址。目的字（D，第三个参数）代表了 ASCII 码将要存入的最低位的地址。

2) ASCII 至 HEX 指令 HEX(-)，与 ASC(86) 指令恰恰相反。将从源字（第一个参数）

代表的地址起始的字节中的 ASCII 代码转换为十六进制代码存入单一目的地址中（第三个参数）。数字指示器（第二个参数）也与 ASC(86) 中拥有相同的含义。如果 ASCII 代码的奇偶性与 Di 所指示的奇偶性不相匹配，则错误标志位（SR25503）置位。

7.8 故障检修

在第 6 章中涉及的指令（一般来说仅对于一个或者两个存储器区域中的数据进行操作）与本章中的指令（对更多的存储器区域中的数据进行操作）之间存在一个主要的区别，就是执行操作的时间。带有几个有条件的文件/数组/结构体指令的程序在其本质上来讲是非常不确定的。这就意味着它们的执行时间会变化很大，可能变化的范围足以导致控制错误的出现。为了避免程序执行时间可变性大的问题，程序员希望限制每次扫描中执行的文件/数组/结构体指令的数目，或者希望编写只需几步就可以对大容量数组进行操作的指令（这样数组操作可以在几个扫描内执行）。如果数组操作被编写成占用几个扫描循环便可完成，那么程序就需要使用状态位，给某些 PLC 指令使用以检测操作何时结束，或者程序要包含其他的检测数组操作何时会结束的机制。另一个使得控制程序操作以更加确定的方式运行的方法是配置 PLC，使它在内部关键操作中使用定时中断，或者在固定的最小扫描时间内执行。这些技术会在第 11 章中讨论。

与智能 I/O 模块交换数据块指令也占用很长的时间来执行。某些 PLC 有独立的通信处理器芯片从用户程序中接受数据传送请求，同时也会按照接收到的请求顺序每次执行一个数据传送，有时在扫描循环中也不对数据传送进行同步控制。这些指令典型地用来对更多完整的状态字或位的集合进行操作，而程序员应当使用可获得的状态信息来确保输入输出数据块不会在不适当的时候改变。

在执行带有文件/数组/结构体操作指令的程序时发生的问题大部分都与在使用操作单独数据字指令时发生的错误类型相同，但却更容易发生。当指令被设计期望使用起始于程序员已知存在的地址的 100 个数据字集合时，很容易忘记的一点是，有可能在该存储器区域中已经不够提供 99 个更高的地址空间了。同样，由于程序列表仅包括了起始地址，也很容易忘记的一点是，编写另一条指令来改变本不应当改变的 100 个地址中某一个的内容。当程序员开发一个程序时，他/她可能会保留一个到目前为止程序使用存储器情况以及每个存储器中保存了什么数据类型的记录。（记录数据类型有助于程序员避免偶然使用了原本准备在另一种数据类型上进行操作的指令）

要记住，PLC 实际上就是一个计算机，而计算机是设计用来一次操作大量数据字的。如果参考手册建议一条指令处理部分数据字而将该字的剩余部分维持不变（就好像某些位数组移位指令做的那样），那可要小心了。仔细阅读参考手册，更好的方法是试着运行一下该指令（处理不用的数据位）来看看该指令在进行字操作时对其他位做了些什么。

通常情况下，PLC 在使用一个并不存在的地址时会出错，同时会保存描述错误情况的错误位或者错误代码，但并不总是这样。例如，某些 OMRON 的 CQM1 的 PLC 在地址 DM 1024 到 DM 6143 之间并没有数据存储器，但是所有的 CQM1 的 PLC 都允许程序读或写这些并不存在的地址。CQM1 同时还允许指令对于 DM 6144 到 DM 6655 之间的只读数据存储器部分进行写操作。

在结构化的 Simens 程序中，记住数据块在使用之前一定要被调用，同时也要记住当控制在组织块、程序块及其他程序之间跳转时，PLC 可能会自动改变正在运行的数据块。尽管通常这是

有用的，因为它避免了程序员在每次跳转之后都要重开一次数据块，但也可能会导致问题。

在程序进入 PLC 存储器之前，如果编程软件可以检测到不正确的存储器引用的话，尽可能找出来。离线编程时，配置编程软件使其与程序将要在其上进行的 PLC 的运行环境完全一致。这会使得编程软件检测到与该 PLC 运行模式不相关的存储器基址。一些编程软件可以配置用来监视某些类型的编程错误，同时忽略其他一些错误。

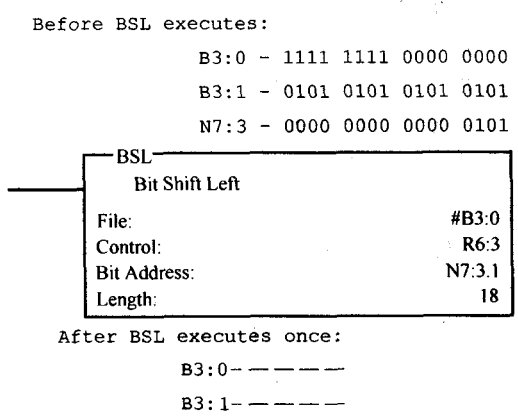
大部分编程软件提供了一个交叉引用选项，可以告诉你每一个包含一个地址引用的梯形图标记。使用它来校验文件、数组及不被其他指令所使用的结构体操作指令所使用的地址。

文件/数组/结构体操作指令在内部使用了变址寻址。为了这样做，它们必须自动使用某些地方的存储器区域来保存偏移量。此时可能会出现两种错误：1）如果同样的存储器区域存储了两个不同函数的偏移量，则文件/数组/结构体指令会改变该偏移量，导致其他指令使用不正确的存储器基址；2）如果文件/数组/结构体指令被中断，另一条指令可以在指令恢复执行之前影响偏移量值。例如，有几种 Allen-Bradley 指令自动使用 S：24 中的偏移量，而 S：24 在变址寻址中也是必须的。

某些 PLC 提供了一些指令，可以用来检测数组内容的意外改变，或者检测数组中的数据与原本应存入数组中的数据之间存在的不同。Allen-Bradley 的文件位比较（FBC）和诊断检测（DDT）指令便是个例子。这些指令在第 15 章中有所涉及。大部分 PLC 提供了一些可以用来使 PLC 出错的指令。如果程序使用了简单的比较或者文件比较指令，调试程序时产生可编程错误就是一种情况，不期望的存储器内容改变（或者未改变成功）都会被捕捉到并被诊断。在 11 章中我们描述了由用户产生的错误及它们如何产生。

习题

1. 数组与结构体之间的不同之处在哪里？
2. 在 Allen-Bradley 执行了下列指令后，B3：0 与 B3：1 位中应该为什么内容？

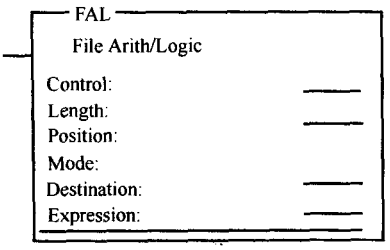


3. （1）如果模块需要大于 16 位字（例如模拟 I/O 模块）的数据，那么在 CPU 和 I/O 模块之间交换数据的 Allen-Bradley PLC-5 中的指令应该使用哪一个？
   （2）如果由（1）中的指令指定的模块地址为 111，那么 I/O 模块在哪里？画出草图清晰准确地标出模块在框架中的位置。
   （3）作为指令的一部分，必须指定一个块传送控制元素，该单元须包含一个完成位。

PLC-5 中的什么元件会将该完成位置位？

4. 填写下图 Allen-Bradley 文件数学/逻辑指令（FAL）中的空白，使得每次其控制逻辑变为真时下列三条运算都会执行。

```
N7:10 = N7:0 * 3 + N9:0
N7:11 = N7:1 * 3 + N9:0
N7:12 = N7:2 * 3 + N9:0
```



5. 一个 Allen-Bradley 控制元素（例如 R6：0）包含了多少数据字？PCL 在这些数据字中的每一个都保存了什么内容？
6. 某些 PLC-5 指令需要（或允许）在地址中使用＃标记（例如使用＃N7：1而不是 N7：1）。该标记代表什么？简要回答。如果仅重复参考手册中关于变址寻址的内容，是不会得满分的，除非你能证明你已经弄清楚了什么是变址寻址。
7. Allen-Bradley 的 FBC（文件位比较）指令可以在调试程序中使用。那么 FBC 指令使用多少数据文件？需要多少控制元素？

推荐的 PLC 实验室练习

一个系统有：

1. 四个控制面板开关：输入 0 到 3
2. 四个指示灯或者可视化输出模块的 LED 灯：输出 A 到 D
3. 两个由弹簧复位阀门控制气缸：输出 E 和 F
4. 一个锁销阀门控制气缸：输出 G 和 H
5. 三个传感器用来检测每个气缸的伸展

编程：

1. 编写一个带有文件操作指令（或者函数调用）的程序，用来：
   （1）当开关 0 开启时，从数据存储器区域中复制五个数据字到另一块区域中。使用 PLC 数据监视和数据写入功能将数据输入到第一个数据区中，同时观察第二块数据区域中的变化；
   （2）将控制指示灯的输出映像数据字（或字节）的所有位进行移位。移位操作在每次开关 1 由断开到闭合时将所有的位上移一位。包括复制映像位状态的逻辑，该位是从字（或字节）中移出并移入映像字（或字节）中最低位。
   （3）每次传感器 2 由断开到闭合时将五个数据字的数组中的 16 位数据字移一位。新的数据字（移入的）必须是从数据存储器中复制过来的（使用数据屏可以改变）。在数组已经移入了五个数据字后，将一个输出位置高，并且复制每一个顺序移出至数组以

外的存储器区域的数据值。

2. 编写一个使用四步输出的顺序程序（如果可以的话使用顺序指令）。程序应该可以按照如下要求监视四个开关及控制气缸：
   (1) 如果所有开关均关闭，则开始伸展所有的气缸。
   (2) 当开关 0 与 1 皆为开时将所有气缸收缩。
   (3) 当所有的开关再次关闭时伸展气缸 G 至 H。
   (4) 当开关 1 与 2 再次开启时将 G 至 H 收缩，同时伸展 E。
   (5) 重复以上操作。
   在顺序器程序中加入定时器，这样在输出改变之间至少会有 3 秒的延迟。
3. 对 PLC-5 编程时，当输入开关保持开状态时，编写一个带有块传送的程序，将 N7：1中的内容连续复制到一个模拟输出模块中。
4. 如果 PLC 提供了一个比较命令，可以对文件进行操作（Allen-Bradley 与 ORMON），编写程序在每次输入位为高时，检测五个数据字集合，同时如果任何数据字为 25 时将输出打开。

第 8 章　程序结构和结构化编程

8.1　学习目标

本章您将了解到：

- 运用主控继电器（MCR）和互锁禁止用户程序段的执行；
- 通过向前跳转或向后跳转来跳过或者重复用户程序段；
- 子程序（如子程序、函数、功能块等）的条件执行；
- 参数传递，使程序可复用；
- 禁止或跳过指令，如定时器和一次翻转法；
- 对 CPU 进行配置来改变 PLC 的扫描循环特性。

这一章介绍了现代 PLC 设备所提供的结构化编程性能：在单扫描循环中允许跳过用户程序段的一部分或者让这个部分多次重复执行的技术。通过把所有的输出关闭或者在最终状态时把他们的输出状态锁存的方法，使用户程序的一段禁止执行。条件执行的部分可以包含只有在必要时才能调用的用户编写的控制程序。PLC 程序员现在需要这种灵活性，甚至这意味着 PLC 的扫描循环可能变得不规则。

在这本书里我们已经强调了 PLC 扫描循环的重要性，扫描循环限制 PLC 重复以下的三个步骤：

1）在扫描循环第一个步骤像拍快照一样创建一个输入条件的静态映像。

2）在第二个步骤执行用户编写的程序。用户程序建立 PLC 保存的输出映像直到用户程序已经执行完毕。从第一条指令开始，PLC 每次执行一条用户程序的指令。梯形图的指令执行从左上角开始，每次只求出一条分支的值，直至执行完右下角的输出元素。

3）在扫描循环的第三个步骤中，像投射一副图像般复制输出映像数据到输出模块中。

到现在为止，这本书所覆盖的编程技术都是一些简单的非结构化编程：所有的指令都执行，并且都是按照他们在程序中的位置依次执行；没有包含子程序、函数调用、跳转指令使程序背离从上到下的执行顺序。早期的 PLC 中用户程序都必须按照这种简单的非结构化形式编写，因为这样更容易模拟实时控制。非结构化编程使输入映像快照与输出映像投影间的间隔尽可能地有规则。但必须注意：尽管结构化编程功能很强大，但是它允许扫描时间是可变的。在第 11 章我们将讨论中断程序的编写，它可以用来将确定性的行为恢复到高度结构化的程序。

我们将通过下面的顺序检查三个主要类型的结构化编程技术：

1）某些条件下，允许 PLC 跳过、重复执行程序中梯级的指令或者禁止所选程序中梯级的指令输出。（专业的程序员可能会说这些特色并不是“真正的”结构化编程技术，但它们确实改变了 PLC 程序的结构。）

2）当条件要求执行时，使 PLC 执行其他的程序如函数、子程序、功能块或者程序文件

的指令。除了在那些处理特殊的 PLC 章节中，这本书我们将函数用作专有名词。提供可写用户函数能力的 PLC 通常也允许在函数中使用变量名替换存储器的绝对地址。(专业程序员对这些结构化编程技术很满意。)

3) 导致 PLC 偏离正常的扫描循环的配置选项，可以在每个扫描循环中执行一个以上的程序，可以改变哪个程序要被扫描，或者可以增加额外的 I/O 扫描步骤。(一些专业程序员甚至不知道这意味着什么。)

正在学习如何对 Siemens PLC 编程的读者请注意：

Siemens PLC 编程语言广泛地运用了结构化编程技术，所以最好在学习简单的指令之前就要理解这些技术。这就是为什么强烈推荐在学习第 6 章和第 7 章之前，先学习这一章。

这一章中涉及函数使用的部分是最重要的，它将在学完跳转指令部分后介绍。现在你可能会希望跳过这些章节，回到第 6 和第 7 章，在学完第 7 章之后再完整的学习这一章。

8.2 在单独的程序中影响执行的指令

8.2.1 主控继电器

主控继电器（MCR）是一种相当于普通使用的电子闭锁电路的软件，用来切换部分自动运行的系统使其关闭而同时让余下的系统继续运行。软件 MCR 与硬件 MCR 有一些显著的不同，所以建议用户不要依赖软件 MCR 来表现安全切断操作性能。软件 MCR 应该被用来启动/禁止与安全不相关的控制功能。

软件 MCR 禁止一系列梯级上的输出。在以下两条梯级上必须包含 MCR 指令：第一条梯级有控制 MCR 启动指令的条件语句，通常叫做 MCR 的*初始声明*。第二条梯级，在 MCR 将要转向控制后输入，必须包含 MCR 的结束指令，通常叫做 MCR 的*结束声明*。

当 MCR 的初始声明的梯级上有如下条件时：

真 MCR 没有效果，在 MCR 的初始声明和结束声明之间的梯级上的指令正常执行。

假 MCR 被激活，在初始声明和结束声明之间的梯级上的输出结果以控制逻辑语句为非的方式执行，而不考虑控制逻辑语句的实际布尔运算结果。例如：

- 如果某条梯级上包含标准-（）-输出指令，则不管这条梯级上的控制语句是否为真，它所控制的位都被关闭。
- 如果某条梯级上包含-（L）-或-（U）-输出指令，则它所控制的位不被这些梯级上的逻辑真所影响，所以它保持在 MCR 的区域被激活之前的那个状态不变。

在一些 PLC 中，MCR 仅仅能影响特殊的输出指令，然而，也有一些其他的 PLC，MCR 可以影响任何的条件输出指令。程序员应该明白，如果 PLC 的操作系统允许 MCR 影响某些功能，而硬件设计者一般不会在硬件 MCR 电路中嵌入这些功能，这时 MCR 可能会导致一些不正常的效果。例如：

- 开延时定时器在 MCR 的控制逻辑为非和 MCR 区域被激活时将会复位，当 MCR 区域逻辑为真以后它又会重新开始计时，甚至控制定时器的逻辑语句并没有为非。

- 关延时定时器如果在 MCR 区域被激活之前它的控制逻辑为真，那么当 MCR 区域的控制逻辑变为非时（这种情况将会禁止很多其他的函数执行），关延时定时器才开始执行。
- 移动指令在 MCR 区域被激活时可以使输出值为零，而不是锁定输出值。

1. Allen-Bradley 主控继电器

PLC-5 SLC 500

图 8-1 是一个 MCR 在 Allen-Bradley PLC-5 或 SLC 500 中编程应用的例子。在初始声明和结束声明之间有 7 条梯级，所以这些梯级的输出都受控制 MCR 初始声明的条件语句的影响。如图所示，MCR 的结束声明必须是无条件的。在程序中如果没有结束声明指令，用户程序结束时的指令将被默认为 MCR 结束声明。

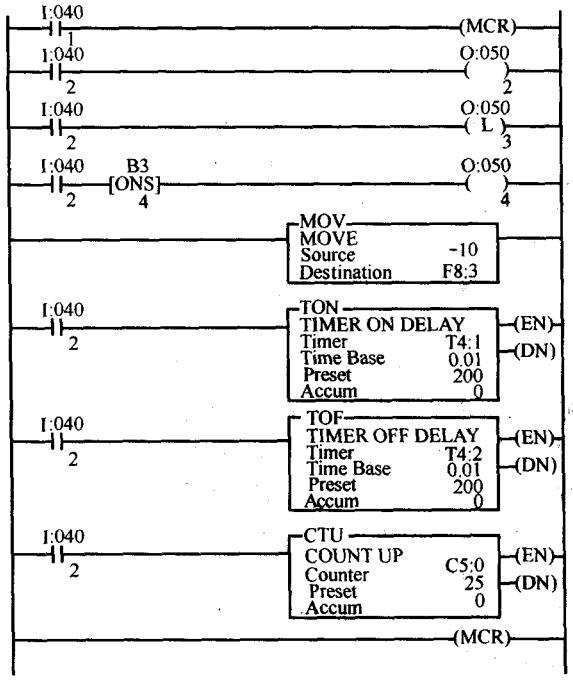


图 8-1 Allen-Bradley 主控继电器示例

如果控制 MCR 初始声明的逻辑语句为非时（例如 I：040/1 断开），初始声明与结束声明之间的梯级上的指令会暂停执行，无论控制受影响的梯级的逻辑语句是否为真（例如 I：040/2处于接通状态）。然后：

- O：050/2 的输出将会断开。
- O：050/3 不能被锁存在 I：040/2。如果 I：040/2 在 MCR 区域被激活之前已被锁存，则 O：050/3 将一直为接通状态并保持下去。如果这个程序某个其他地方的梯级将 O：050/3 关闭，那么在 MCR 区域内的这条梯级不能将它重新锁存。
- 一次翻转法位：B3/4 和 O：050/4，这些被一次翻转法控制的，不考虑 I：040/2 的状态，将继续保持断开。
- 尽管 MOV 指令被编成无条件执行指令，但在这里它不会被执行，所以地址 F8.3 的输出不会被 MOV 指令所影响。

- 接通延时继电器的累加的值（T4：1. ACC）将会复位（T4：1. TT 和 T4：1DN 将会断开）。
- 断开延时继电器（T4：2）将会启动，正如 I：040/2 断开一样（所以在接下来的 2 秒内 T4：1. TT 和 T4：1. DN 在 2S 后会接通）。
- 向上计数元素（C5：1）将会让它的 CU 的状态位（C5：1. CU）关闭。

当 I：040/1 接通时，将会使得 MCR 停止并且重新让 MCR 区域内的梯级的指令运行，然后会有：

- 如果 I：040/2 处于接通状态，那么 O：050/2 和 O：050/3 将会接通。
- 如果 I：040/2 接通，一次翻转法将使它的输出位 O：050/3 工作一个循环，即使在 MCR 处于活动状态时 I：040/2 没有改变状态，并且它还会将位 B3/4 接通。如果 I：040/2是关闭的，将不会产生一次翻转法，即使在 MCR 处于活动状态时 I：040/2 确实改变了状态。
- 如果 I：040/2 是接通状态，那么接通延时继电器将从 0 开始重新启动计时，即使当 MCR 被激活时 I：040/2 处于接通状态而且一直没有改变状态。
- 如果 I：040/2 处于接通状态时断开延时继电器将置零，但是如果 I：040/2 处于断开状态则它将不受影响。
- 计数器元件（C5：1）将它的 CU 状态位（C5：1. CU）变回接通状态，并且会增加它的计数值如果 I：040/2 是接通状态，即使它不曾断开过。

如果 Allen-Bradley PLC 的程序包含跳转到子程序的语句并且这个语句处于活动的 MCR 区域，那么将不可以跳转到这个子程序（子程序是在这章后续的章节中要介绍的结构化编程特色之一）。被不允许执行的子程序控制的函数将保持冻结状态直到该子程序被允许再次执行。如果 MCR 的初始声明在一个子程序中，而这个子程序不包含结束声明（或者如果该子程序中有指令在 MCR 的结束声明梯级执行之前结束这个子程序），则这个子程序的结束指令被当作 MCR 的缺省的结束声明。

Allen-Bradley PLC 中不允许有重叠的 MCR 区域。当 MCR 区域变成活动后程序的执行将稍微变快，因为控制逻辑能够被当作非的状态。

S5

2. Siemens STEP 5（无 MCR）

STEP 5 不提供 MCR 指令。为完成与 MCR 相同的功能，STEP 5 程序员必须另外编写一个使在独立的程序段中控制逻辑变为非的程序，就像使 MCR 区域被激活一样。例如：控制 MCR 初始声明的逻辑要在每一个程序段中被编写，否则该程序段就必须处于初始声明与结束声明之间。如果 MCR 的控制逻辑是复合的，那么它能够用来控制一个能够被一些程序段检测的标志位。

后续章节中将要介绍的其他结构化编程的特色（如条件跳转到程序块或者功能块）能够被用来使所选的输出断开的条件地执行逻辑，不考虑其他的不控制他们的状态的逻辑。但是要注意，如果在程序中不止一个逻辑语句尝试去控制同一个输出，最后执行的语句将实际上控制输出的状态。

S7

3. Siemens STEP 7 主控继电器

Siemens 已经在 STEP 7 的操作系统中添加了主控继电器的特色，极大地满足了美国的市场。STEP 7 的 MCR 比已经在美国的 PLC 中提供了数年的 MCR 拥有更多的优点，它们

严格限制了自己能够影响的输出指令的类型。

能够被 STEP 7 的 MCR 所影响的仅有的指令如下：

- STL 语言的输出指令：＝以及梯形图中的等价指令：—（）和-（#）-。
- STL 语言中的置位和复位指令：S 和 R，以及梯形图中的等价指令-（L）-、-（U）-，置位-复位触发器和复位-置位触发器。
- STL 语言中的传送指令：T，以及梯形图中等价的移动指令。（注意 STL 中的加载指令 L 不仅仅在 MCR 中执行。）如果传送（或者移动）指令在一个活动的 MCR 区域内，它将会向目的地址写入 0。

像前面几章一样，在这章中我们继续主要讨论 STL 语言中的指令。这里包含一些与每条 STL 指令等同的 STEP 7 中的梯形图的指令。在 STEP 7 中，MCR 区域必须从有条件的开始 MCR 指令开始，MCR（，和以一个无条件的结束 MCR 指令结束，）MCR。每一个 MCR 指令必须有一个 MCR 指令与之配对。Siemens PLC 要求使用另外两条 MCR 控制指令：无条件的 MCR 激活和 MCR 解激活指令，MCRA 和 MCRD。如下面的程序例子所示，MCR 仅仅影响处于 MCR（与）MCR 指令对和 MCRA 与 MCRD 指令对之间的指令。

在单个程序中的单个 MCR 区域：

```
A  M0.0          //打开一个 MCR 区域
MCR (
A  I4.0          //在 MCR 区域中的语句
=  Q5.0
MCRA             //激活任何开放的 MCR 区域
A  I4.1          //在激活的 MCR 区域中的语句
=  Q5.1
S  M3.0
R  M3.1
L  +15
T  MW2
CU C1
) MCR            //关闭 MCR 区域
A  I4.2          //在 MCR 区域外的语句
=  Q5.2
MCRD             //抑制任何开放的 MCR 区域
```

在这个程序中：

- Q5.0 仅受 I4.0 控制，即使 M0.0 为非，因为 ＝Q5.0 在 MCRA 指令之前被写入程序。同样的，Q5.2 仅受 I4.2 控制，因为它在 MCR 指令后被写入。
- Q5.1 受 I4.1 控制，除非 M0.0 处于非的状态，这样的话 Q5.1 将一直处于断开状态。
- 当 M0.0 为非的时候，M3.0 不能被 I4.1 置位，M3.1 不能被 I4.1 复位。
- 值 ＋15 将被加载到累加器 1 中，而不管 M0.0 的状态。如果 M0.0 是真，＋15 将被传递到 MW2。如果 M0.0 的状态是非，值 0 将会被传送至 MW2。
- 增计数（CU）指令不受 MCRs 的影响，所以 C1 将在每次 I4.1 变为真的时候加 1，而不管 M0.0 的状态。

STEP 7 调用其他程序块的指令（这章的后面将讨论函数和功能块程序）不受 MCR 的

影响，所以其他程序块可以从活动的 MCR 区域内被调用。MCR 不会影响在程序块中的指令除非这个程序块执行 MCRA 指令。在一个被调用的程序块中 MCRA 与 MCRD 指令的使用不影响 MCR 在所调用程序中是否被激活。下面的程序示范了在 MCR 区域中程序块的调用。

在结构化程序中的单个 MCR 区域：

```
A   M0.0          //打开和激活一个 MCR 区域
MCR (
MCRA
A   I4.3          //函数 10 的有条件的调用
CC  FC10
A   I4.5          //在一个调用后 MCR 区域中的语句
=   Q5.5
) MCR             //结束和抑制 MCR 区域
MCRD
FC10
                  函数 10，在 MCR 区域中调用
A   I4.3          //先于 MCR 激活的语句
=   Q5.3
MCRA
A   I4.4          //跟随 MCR 激活的语句
=   Q5.4
MCRD              //在函数 10 中抑制 MCR 区域
BE                //结束函数 10
```

在上面的例子中：

- 当 I4.3 为真时函数 10 被调用，即使 M0.0 为非，因为条件调用指令不受 MCR 的影响。在函数 10 中：
  1）Q5.3 仅受 I4.3 影响，即使 M0.0 为非，因为 MCR 区域自动地在程序块被调用时变为不活动。
  2）如果 M0.0 为非，则 Q5.4 将也为非，而不管 I4.4 的状态，因为函数 10 是从被 M0.0 控制的 MCR 区域中被调用，还因为函数 10 中的 Q5.4 是在 MCRA 指令执行以后被控制的。
- 如果 M0.0 为非，则 Q5.5 将也为非，而不管 I4.5 的状态，并且不管函数 10 是否被执行。函数 10 包含一个 MCRD 指令，但是当一个被调用功能块结束处于该功能块的末尾的 MCRA 与 MCRD 语句的作用时，MCR 将继续保持调用之前的活动性。

STEP 7 允许 MCR 嵌套，最大的嵌套深度可以达到 8！可以有多达 8 个的有条件 MCR（指令在第一个）MCR 指令之前被执行。第一个遇到的 MCR 声明结束由最近的 MCR（指令所开始的 MCR 区域。在一个以上的 MCR（与）MCR 指令对中指令执行都取决与所有他们所在的 MCR 区域的开始条件。如下面的例子中所示，甚至如果这些 MCR（指令中的一个的控制条件为非（并且一个 MCRA 指令已经被执行），那么所有在 MCR 区域中处于非的控制条件下的指令都会受 MCR 的影响。

使用 MCR 区域的嵌套的程序：

```
MCRA          // 激活所用的 MCR 区域
A     M0.0    // 打开第一个 MCR 区域
MCR(
A     I4.6    // 在第一 MCR 区域层的语句
=     Q5.6
A     M0.1    // 打开第一层内的第二 MCR 区域层
MCR(
A     I4.7    // 在第二嵌套区域中的语句
=     Q5.7
)MCR          // 关闭第二 MCR 区域
A     I4.6    // 在第一 MCR 区域层的语句
=     M3.6
)MCR          // 关闭第一 MCR 区域
MCRD
```

如上面的程序中所示：

- 如果 M0.0 为非，Q5.6 和 M3.6 将断开，不管 I4.6 的状态。因为 Q5.6 在第二个 MCR 区域开始之前被控制，并且 M3.6 在第二个 MCR 区域结束之后被控制，所以他们不受 M0.1 影响。
- 如果 M0.0 与 M0.1 中的任何一个为非或者两者都为非，那么 Q5.7 将断开，而不考虑 I4.7 的状态。

在下面的章节中我们将讨论在一个单扫描循环中被用来跳过或重复 STEP 7 中的程序段的指令。建议程序员尽量避免会跳过或者重复 MCR 指令的跳转，尤其是有条件跳转时。

即使你确信能处理可能会发生的嵌套 MCR 区域和激活的混乱的交织网，当你去到管理层时，想想那个可能会不得不修改你的程序的可怜的程序员吧。

4. OMRON CQM 1 的互锁（主控继电器）

CQM1

CQM1 具有 MCR 的特色，但是 OMRON 把它叫做互锁。对梯级的互锁范围的初始声明必须包含逻辑去控制 IL（02）（互锁）指令，并且结束声明必须包含一个无条件 ILC（03）（互锁清除）指令。如果初始声明中的逻辑为真，在 IL（02）与 ILC（03）指令之间的梯级将正常执行，但是如果逻辑为非，则所有这些互锁的梯级上的逻辑语句都被看作为非。输出指令将把他们的逻辑当作为非或刚变为非的状态执行，所以被非锁存输出指令控制的位都将断开，定时器都将复位，计数器都将停止计数。

被 DIFU（13）或 DIFD（14）指令（Omron 把一次翻转指令叫做微分指令）控制的一次翻转在 IL（02）的逻辑变回真时可能会执行，但只有在当互锁停止变为活动时他们的控制逻辑是不同的条件下才会执行。例如，如果当互锁变成活动或者互锁不再活动时控制 DIFU（13）指令的逻辑为真，DIFU（13）指令将不产生一个一次翻转，即使当互锁活动时 DIFU（13）的控制逻辑改变了数次。另一方面，如果当互锁变为活动时 DIFU（13）的控制逻辑为非，并且当互锁不再活动时控制逻辑为真，DIFU（13）指令将产生一个（延时的）一次翻转。

重叠的互锁区域是允许的（尽管编程软件可能会发出错误警告），但是遇到的第一个 ILC（03）指令将终止所有活动的互锁区域。

8.2.2 跳转指令

跳转指令允许在一个扫描循环内用户程序跳过或者重复一些程序中的指令。尽管这个功

能并不是非 PLC 程序员所谓的真正的结构化编程组成之一，但它确实允许程序员去写顺序而不是仅仅从上到下执行的结构化程序。

跳转指令是输出指令，所以有跳转指令梯级上的布尔逻辑条件控制了 PLC 是否将立即跳转到程序中别的地方的有标号的梯级上。除了在跳转指令被执行时，标号指令不影响程序执行的。被向前跳转指令跳过的梯级上的输出都被有效地锁定，因为梯级没有被执行。如果跳转指令使部分程序被跳过，程序执行时间将减少。

一些 PLC 不允许向后跳转，因为这样有可能会导致死循环，导致看门狗定时器故障。因为当 PLC 重新执行一个扫描循环时，PLC 会固定地重复用户程序，所以无论如何向后跳转都是不必要的指令。即使没有向后跳转指令，整个的程序将在几毫秒内再次被执行。

PLC-5 SLC 500

1. ALLEN-BRADLEY 跳转指令

当控制 PLC-5 或 SLC 500 跳转（JMP）指令的有条件语句为真时，PLC 将通过程序文件向前或向后跳转到由标号（LBL）指令开始的梯级，标号指令的数字与 JMP 指令的数字相同。可以有几个 JMP 指令拥有相同数字的目的标号，但是每个 LBL 指令必须有独特的数字编号（通过你的手册查看允许的数字范围），并且 LBL 指令必须是它所在的梯级上的第一个元素。LBL 指令必须在一条至少有一个输出指令完整的梯级上，并且这条梯级的执行不受 LBL 的影响。

在图 8-2 的例子中展示了怎样使用跳转指令编一个 PLC-5 程序：

1）当 I：040/1 为真，处于 JMP 指令与 LBL 指令之间的梯级将被跳过，所以：

- O：050/2 的输出保持锁定在它的最后状态，即使 I：040/2 的状态改变了。
- MOV 指令没有执行，即使它是无条件的。
- 开延时继电器的累加值（T4：1. ACC）锁定，因为 PLC 没有像执行梯级那样本应该正常地做每个扫描循环去更新它的值。. TT 与 . DN 位保持它们最后的状态。（当 TON 指令继续执行时，. ACC 将继续增加，好像没有时间流逝。）

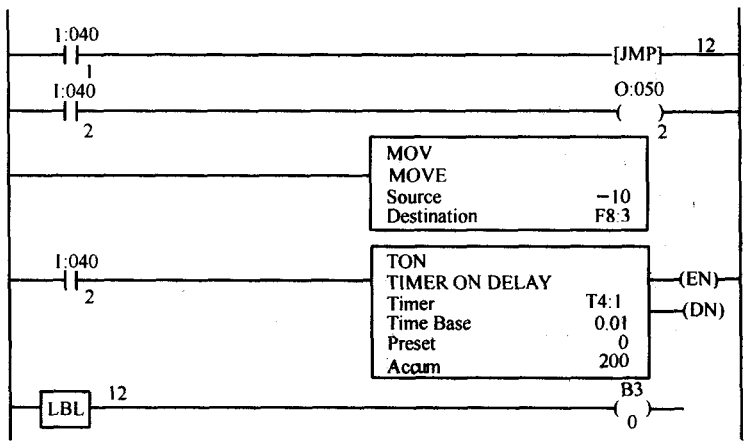


图 8-2 有向前跳转的 Allen-Bradley PLC-5 程序

2）当 I：040/2 为非时，程序像没有 JMP 或 LBL 指令一样被执行。

不管 JUMP 指令是否执行，B3/0 位都将无条件的接通。它在这里被编程出来是为了显示 LBL 指令必须在一个完整的梯级上。一般地，有 LBL 指令的梯级上将包含一个有用的控

制操作。

在图 8-3 中的例子显示了一个在 Allen-Bradley 程序中不应该被编写的跳转。考虑一下在扫描循环中将会出现什么情况，在你阅读书中对这个错误的解释之前，你能否指出这个程序中发生了什么错误。程序看起来好像会与 I：040/3 关联的输入的接通时间一样长时间地重复地监控与 I：040/4 关联的输入情况去控制与 O：050/4 相关的输出。（我的措辞应该会给你一些提示。）

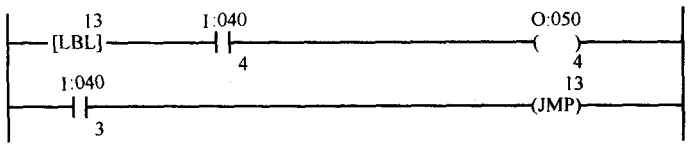


图 8-3 使 PLC 出错的 Allen-Bradley 跳转程序

这里解释了图 8-3 中的程序出错的原因：当与输入映像位 I：040/3 相关联的输入保持断开时 PLC 将不会出错。PLC 将重复执行它的扫描循环，输出映像位 O：050/4 控制的输出接通或关断就像控制输入映像位 I：040/4 的输入接通或断开一样。JUMP 指令将不执行，所以正常的扫描循环不受 JMP 指令的影响。

在与输入映像位 I：040/3 相关联的传感器接通后，输入映像位 I：040/3 在下一个输入扫描步骤时将被当作真，然后用户程序将开始执行。输入映像位 I：040/4 的状态将被写入输出映像位 O：050/4，然后 JMP 指令将执行。没有改变的 I：040/4 的状态将再次被写入 O：050/4，然后跳转也将被再次执行，因为在程序中没有什么改变输入映像位 I：040/3 的状态。即使控制 I：040/3 的传感器断开，PLC 也不能从用户程序中跳出来去执行输入扫描，所以输入映像位 I：040/3 将永远不会断开。注意到输出扫描也不会执行，所以被 O：050/4 控制的激励器将保持在它最后的状态（在循环执行之前的状态）而不管 I：040/4 的状态。

PLC 被设计成不允许这种类型的死循环执行很久。当扫描循环执行时所有的 PLC 都运行一个看门狗定时器，如果看门狗定时器超过预设的时间时，PLC 将会出错（停止运行）。在第 11 章中将详细的检视看门狗定时器的缺点，并且那时我们将会明白如何去改变默认的看门狗定时器的预设值。

2. Siemens STEP 5 的跳转指令

在单个 STEP 5 的程序中的跳转指令仅能在功能块中⊖编写并且仅能使用 STL 语言。跳转指令能向前跳转或向后跳转。下面的例子显示了在 STL 程序中典型的 STEP 5 条件跳转指令（JC）：

```
      : L  KH00  ;  加载8位的全零数进入累加器
      : A  I4.0  ;  如果输入 I4.0置位，
      : JC = past;  跳过下一条指令
      : L  KHFF  ;  更新累加器的内容为全1
past  : T  QB5   ;  发送累加器中的所有内容到一个输出模块
```

单词 past 是程序员编写的由字母或数字组成的标号，它不能超过 4 个字符，并且它的第一个字符必须是字母，同时要注意标号必须在冒号的左边输入。跳转命令必须指出所要跳

⊖ Siemens PLC 默认将组织块作为正常扫描循环来执行，并且 STEP5 不允许在组织块中程序的跳转。这里介绍的在一个程序中跳转的信息，只有在你学习了本书后面章节介绍的怎样在功能块中跳转之后才有用。

向的标号，并且标号的标识符之前必须要有“=”以指出要跳转到功能块内的一个标号。跳转既能向前也能向后，并且一些跳转指令能跳转到同一个标号。当然，在同一个程序中不应该有两个同样的标号。

个别的 STL 指令（如加载和传送）无条件地执行，但它们也可以通过编写有条件跳转指令来跳过它们从而有条件地执行。上面的程序例子中示范了 JC 指令怎样被用来进行有条件的加载（条件加载，LKHFF）。一些梯形图指令可以在功能块中但不能在其他 STEP 5 程序中进行条件化编写（如移动）。在将指令送到 PLC 之前 STEP 5 会自动地将梯形图程序转换为 STL 语言，并且如果它们在程序块中被编写，则可以插入跳转指令（在梯形图中不可见）使他们条件化。某些梯形图指令仅在功能块中是可用的，个别的会自动包含跳转指令当作他们如何运行的部分。

除了在上面所示的有条件跳转指令（JC）之外，还有一些其他的 STL 跳转指令。如无条件跳转（JU），还有个别指令将导致跳转取决于操作系统维持的状态位的状态，就像它操作数据字那样。由状态字决定的跳转指令包括 JZ、JN、JP、JM 和 JO，如果最后处理的数据分别是 0、非 0、正数，负数或超出累加器的 16 位有符号数容量的溢出，它们将会使跳转发生。

S7

3. Siemens STEP 7 中的跳转指令

在 STEP 7 中，跳转可以向前也可以向后并且可以在任何类型的程序块中编写。每个目的标号必须是独特的不超过 4 个字符由字母或数字组成的标号，并且第一个字符必须是字母。有个别跳转指令能够跳转向相同的目的标号，但是在同一个程序中不能有两个相同的目的标号。

在 STL 与梯形图中，跳转指令看起来会有显著的不同，所以现在示例一个梯形图的例子。图 8-4 显示了梯形图中的跳转（JMP）指令和目的网络中的标号。JMP 指令能够被条件化编写（如图所示）和无条件编写。如果之前的条件（如果任何一个）为真，JMP 指令将执行。梯形图中为非时跳转（JMPN）指令则在它的控制语句为非时执行，并且它必须进行条件化编写。注意，目的标号必须是目的网络的第一个元素并且在梯形图中要在网络之前出现。

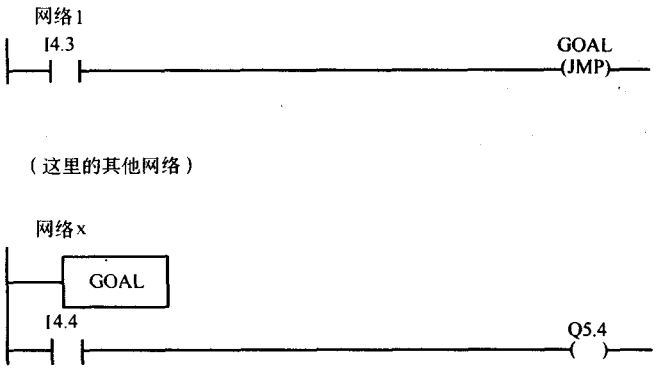


图 8-4 STEP 7 梯形图跳转指令

STL 等同于图 8-4 梯形图的程序如下，它包含了 STL 条件跳转指令（JC）：

```
A    I4.3
JC   GOAL
```

```
          (这里的其他指令)
GOAL: A   I4.4
      =   Q5.4
```

在上面的 STL 程序中，目的标号必须同冒号（:）一起被输入，因为 STEP 7 编辑器像 STEP 5 编辑器一样每一行都不显示冒号，但是目的标号仍然必须要在冒号的左边并且程序仍然必须在冒号的右边。在 STEP 5 中 JC 命令能够被用来调用其他程序块，但是在 STEP 7 中它则不能，所以 JC 命令不需要“=”和目的标号在一起出现。

如果用 STL 语言编写，必须要注意许多 STL 指令是无条件执行的⊖。甚至用来复制数据值进出累加器加载（L）指令与传送（T）指令都必须无条件地执行。你可以通过编写条件跳转来跳过它们使无条件指令有条件地执行。如果你将一些梯形图指令转换为 STL 语言，你将会看到它们会转换成一系列 STL 指令，包含通过有条件跳转跳过无条件的 STL 指令。

STEP 7 仅仅提供上面所包含的两个梯形图跳转指令，但是给喜欢 STL 语言的程序员提供了大范围的跳转指令。与梯形图跳转指令等同的 STL 跳转指令包括：

- JC，条件跳转指令。
- JU，无条件跳转指令。
- JCN，如果之前逻辑语句的 RLO 为非，则产生一个跳转。

在梯形图中没有与之相当的 STL 跳转指令，它包括以下指令：

- JCB 与 JNB，分别同 JC 与 JCN 一样，但是又复制 RLO 到第一个状态字的 BR 位。
- JL，它允许 PLC 从可跳转目的标号的列表中选择，当 JL 指令执行时取决于累加器 1 中是什么数字。（要了解更多的细节请查看编程指南。）
- JBI 与 JNBI，如果 BR 位是置 1（JBI）或被清 0（JNBI）时将产生跳转。
- 取决于许多最近执行的数学运算结果的导致跳转的指令有：JZ，结果为 0；JN，结果为非 0；JP，结果为正值；JM，结果为负；JPZ，结果为正或者 0；还有 JMZ，如果结果为负或者为 0。
- JOV，如果最近的数学运算产生的有符号的二进制数字超出累加器范围则导致跳转；JOS，如果任何数学运算结束前最后一次的 OS（溢出设置）状态位被清零，它将导致跳转。
- JUO，如果最近的数学运算是浮点数学运算，但是产生的浮点数字不是有效的浮点形式，那么将导致跳转。
- LOOP，在程序中被用来编写重复的循环，它将在后续章节被介绍。

4. OMRON CQM1 跳转指令 CQM1

不像其他的 PLC，OMRON 的跳转在控制跳转指令的控制逻辑为非时执行。图 8-5 展示了 OMROM 的条件跳转指令 JMP（04）和无条件跳转指令 JME（05）的使用。只有在输入映像位 IR 00503 处于断开状态，跳转才会执行。

OMRON 的 JMP（04）指令必须包含与 JME（05）指令的数字相匹配的数字标号（01-99）。同一程序中，相同的标号数字不能在其他的 JMP（04）或者 JME（05）指令中使用。

⊖ 你可以在 STL 程序中将条件语句放在无条件执行指令（如 LOAD 指令）之前，但是当你运行程序时你会发现 LOAD 指令的执行并不受条件语句是否为真值（RLO）的影响。

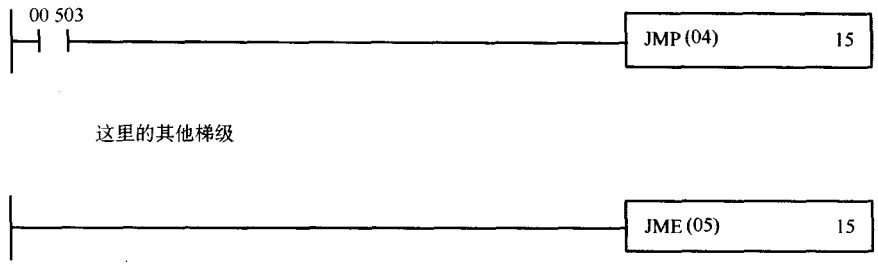


图 8-5 有跳转指令的 OMRON PLC 程序

JME（05）指令是跳转目标，但是必须编写成无条件的输出指令。在 OMRON 的 PLC 中不允许编写向后跳转指令。

标号数字 00 能够被用来替代 01-99 之间的数字，但是这样规则就有一些不同。任何 JMP（04）和 JME（05）指令的数字都可以使用标号数字 00。跳转到标号 00 的指令将会向前跳转到下一个标号为 00 的 JME（05）指令。

8.2.3 循环

循环在很多结构化编程语言中是普通元素，但是直到最近它才在 PLC 的编程语言中可用。在结构化语言的循环中，中央处理器向后跳转去执行一系列指令并且保持一个计数器来确保向后循环执行预先确定的次数。

对数据元素队列进行操作的 PLC 指令包含嵌入循环（PLC 程序员是看不见的），这些指令每次只对队列中的一个元素进行操作。在这一节中，我们简要的考查 PLC 程序员能够用来创造循环的指令。这本书里没有详细地介绍循环指令，因为在 PLC 中它们有些多余。PLC 的操作系统使得整个程序不管怎样都重复的（循环）扫描！循环在某些应用中是有用的，例如在单扫描中有多重数据字要被移动，但是循环的误用可能会使看门狗定时器导致 PLC 的故障。

这本书里仅有两种 PLC 提供了循环指令，讨论如下。

PLC-5

1. ALLEN-BRADLEY PLC-5 中的 FOR-NEXT 循环指令

PLC-5 中的 FOR（FOR）和 NEXT（NXT）指令对被用来使梯级产生明确次数的循环，这些梯级处于控制 FOR 指令的有条件梯级和包含 NEXT 指令的无条件梯级之间。当控制 FOR 指令的条件语句保持为真时，每个扫描循环都执行一组循环。

在 FOR 指令中，程序员必须录入和与 FOR 配对的 NXT 指令的标号数相同的标号数字，并且必须输入地址作为索引以便让 PLC 能够保持跟踪在循环多少次后跳回那里。同样在 FOR 指令中，程序员必须输入初始值、终结值和步长，以便让 PLC 知道重复循环过程的频率。（例如：输入 0、5、1 分别作为初始值、终结值和步长将会循环 5 次；输入 100、110 和 2 则也是循环 5 次。）

处于 FOR 与 NXT 指令中间的梯级中使用可选的 BREAK 指令（BRK）可以使循环提前（有条件的）结束。

S7

2. Siemens STEP 7 中的循环指令

当 S7 PLC 执行循环指令时，它让累加器 1 中低位字中的 16 位数（把这个数字当作无

符号二进制数）递减。如果结果数值为非 0，PLC 会读出 LOOP 指令后面的标号并在程序块内部跳转到之前有相同标号数的指令那里。跳转典型的都是向后跳转，以便在同一个扫描循环中让循环指令再次被执行。

为了使用循环指令，程序必须首先在累加器 1 中加载一个正数以指出循环的次数（不超过 16 位无符号二进制数的最大值 64，565）。程序在循环指令执行以后应该立即保存（传送出）累加器 1 中的循环计数器的数值（在循环中之前有标号的指令将在下一步执行），并且在每次循环指令执行前要马上将循环计数器中的值恢复（加载）到累加器 1 中；否则，在循环范围内的指令可能会影响累加器 1 中的内容。下面的例子中循环指令将使循环内部的其他指令执行 5 次：

```
      L    + 5
More: T    MW10
      (循环中的指令)
      L    MW10
      LOOP more
```

8.3 在程序扫描的过程中影响子程序或函数执行的指令

使用单个 PLC 程序控制几个过程，或者甚至控制一个复杂的过程时，可能会变得十分混乱。比较好的方法是编写一个主程序，来调用需要的子程序。复合程序结构可以让人很容易明白程序中的哪个部分控制哪个过程，同时它也方便了几个程序员一起去编写同一个工程，每个程序员各自编写自己的子程序部分。

当主程序调用子程序时，在子程序运行由开始到完成这段时间主程序会暂时停止执行。在调用完成后，主程序会从调用子程序指令的下一条指令开始继续运行。子程序可以调用其他子程序。图 8-6 显示了在三个步骤的循环周期中用户程序步如何受子程序调用的影响。

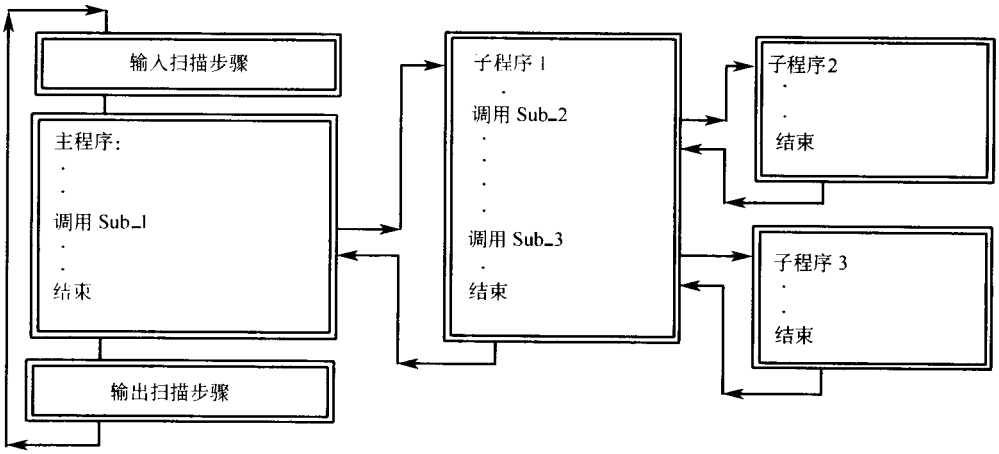


图 8-6 主程序中的子程序调用

如果多程序工程被详细地规划好，一些子程序是可以重复使用的。可重复使用的子程序可以被主程序多次调用来控制多个过程，同时主程序要为被调用的该子程序每次提供不同的参数。例如，图 8-7 中包含了一个可重复使用的子程序来比较传感器温度与设定的期望温度的差，并且通过加热或冷却来减小这种差。相同的子程序被主程序调用了两次，并且主程序

在每次调用时都提供了不同的参数，让子程序知道在每次调用时使用哪一个温度传感器，设定的温度是多少，以及应该用哪一个加热器。

子程序可共用被其他主程序、子程序使用的数据区，或者可能会使用它自己的私有的数据区域（如 IEC 1131-3 所要求的）。有自己数据区的子程序在它们被调用时需要从主程序那里接收输入参数数据并且当它们运行结束后要将结果参数传回给主程序。在图 8-7 中，主程序提供了两个输入参数（分别是传感器地址和设定温度值）给子程序，并且为子程序运行的结果提供了一个目的参数（加热器的地址）。

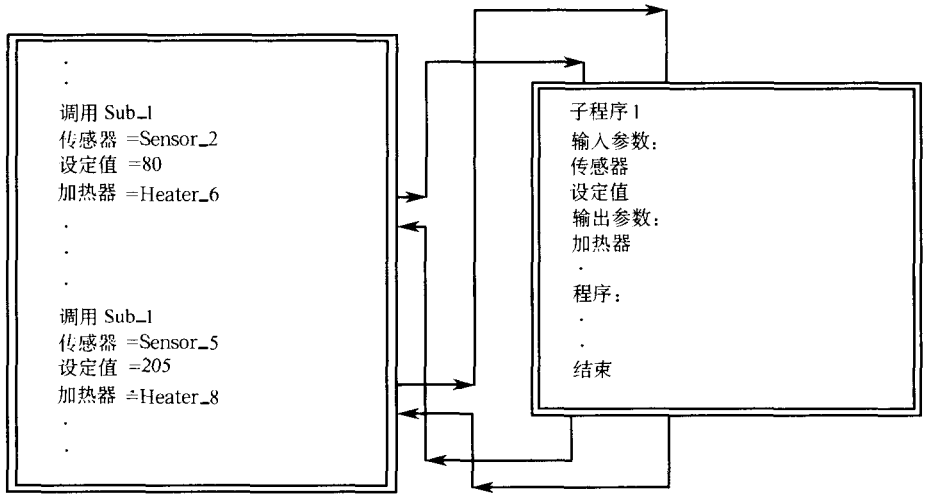


图 8-7 可重复使用的子程序的多重调用

PLC 厂商都有它们自己对子程序的叫法。有些把它们叫做子程序，有些使用术语程序文件，其他的会使用名词如功能块。IEC 1131-3（国际标准化组织对 PLC 编程标准的说明，将在第 9 章中介绍）规定程序、功能块和函数可以被任何其他的程序、功能块和函数调用。（程序、功能块和函数有些许的不同特征，在 IEC 1331-3 有明确定义。）

PLC-5 SLC-500

8.3.1 ALLEN-BRADLEY 的子程序调用

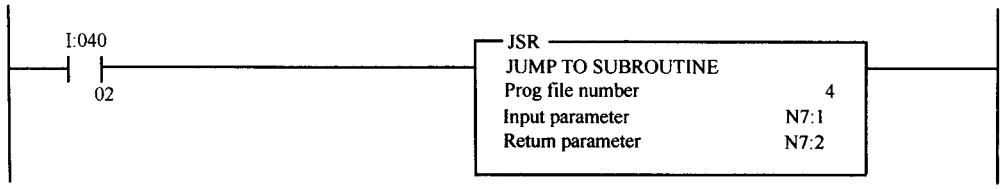
最新生产的 PLC-5 或 SLC-500 型号的 PLC 在它的扫描循环中自动把程序文件 2 中的程序当作用户程序执行。编程软件都这样执行，所以程序文件 2 是当程序员输入程序编辑码时系统默认产生的程序。（这个默认设置可以更改。）

任何 Allen-Bradley 程序文件（包括文件 2）都包含跳转到子程序（JSR）指令来调用任何其他的程序文件，这些程序文件由子程序（SBR）指令开始并已经被标识为子程序，并且可以接收调用程序试图传递的任何输入参数。SBR 指令必须是子例行程序文件中第一条梯级上的第一个元素，并且它不会影响所在的梯级的逻辑状态。子程序能够轮流使用另外的 JSR 指令来调用第二个程序文件直到最大嵌套调用的深度达到 8 层。图 8-8 展示了无嵌套的子例行程序调用，第一个调用不计在调用嵌套层数内。每一个被调用的程序文件都必须由 SBR 指令开始。

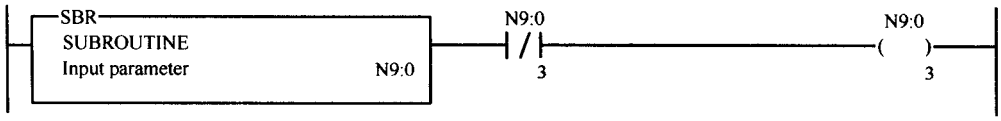
每个子例行程序都应该以一条返回（RET）指令结束，通过它指定返回参数，产生一个返回使程序回到调用子程序的程序文件。返回参数（如果有的话）从子例行程序那里被接

收；然后在 JSR 指令后面的指令被执行。所有的子程序都以一条 RET 指令结束是很重要的，因为这样调用第一个子程序的第一个主程序最终可以继续执行。

如果输入 I：040/2 处于接通状态，则这条指令（或许在 MCP 文件2中)将调用子程序文件（文件 4)。程序员已经为一个输入和一个输出参数指定了地址。

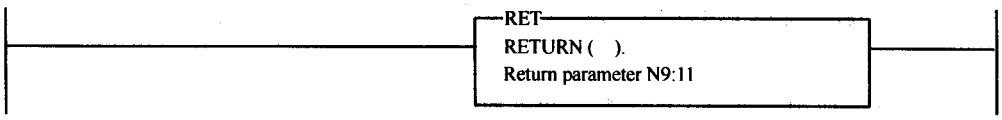


程序文件 4 中的 SBR 指令要从调用程序中接收一个输入参数。这里，它接收了来自 N7：1 的一个数据，并且把数据传给 N9:0，然后文件 4 处理了这个来自 N7:1 的数据（改变了 N9:0)。



文件 4 中接下来的梯级（不可见）处理了 N9：11 中的数据。

文件 4 中最后的梯级包含一个传回控制到文件 2 的 RET 指令，并且给文件 2 一份来自 N9：11 的新数据。



文件 2 中的 JSR 指令直到 PLC 把来自 N9：11 中的数据复制到 N7：2 中才完成，然后文件 2 的下一条梯级才执行。

图 8-8　跳转到另一个文件中的子程序的 PLC-5 程序

Allen-Bradley PLC 仅有一个数据存储区，叫做数据表，它是由一些数据文件组成的。任何程序文件都可以在任何数据表中读或写，所以被子程序改变的数值可能会影响使用同一数据地址的其他程序文件的操作。因为仅有一个公用数据表，所以子程序和调用该子程序的程序之间没有必要传递参数，除非该子程序是可以重复调用的。

调用程序可能（选择的）传递输入参数到被编写用来重复调用的子例行程序文件中。在 JSR 指令中传递输入参数常数或地址将会产生由子程序的 SBR 指令指定的存储地址的输入数据的副本。调用程序中的 JSR 指令也能够（选择的）指定返回参数的返回地址。当子程序结束时，PLC 从由子程序的 RET 指令指定的数据存储区地址中复制数据到由主程序的 JSR 指令指定的存储区地址中。图 8-8 显示了有跳转到子程序指令和输入、返回参数说明的 PLC-5 程序。

S5 8.3.2 SIEMENS STEP 5 的函数调用

默认状态下，在 S5 PLC 的扫描循环中组织块 1（OB001）被当作用户程序执行，并且编程器默认的自动编辑 OB001。任何块（包括 OB001）都能够调用任何其他的块作为子程序（当然除了数据块）。使用无条件跳转或有条件跳转指令可以使块的嵌套调用深度达到 16 层，如图 8-9 中梯形图和 STL 语言程序所示。

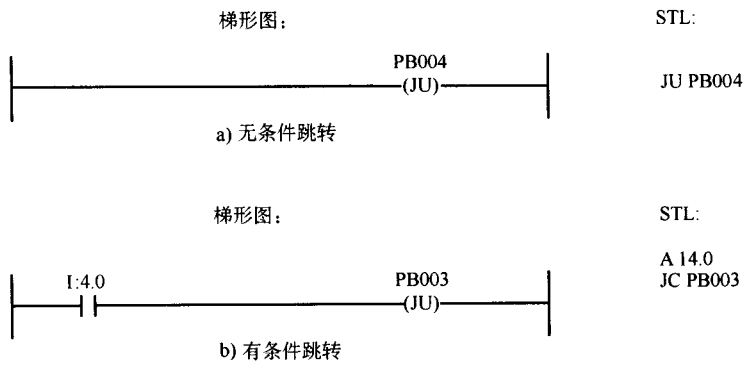


图 8-9 在 STEP 5 中跳转到程序块

在 S5 PLC 中，所有的程序和函数都共用相同的标志存储区和相同输入和输出映像存储区。STEP 5 程序能够拥有多达 255 个数据块，一次只能打开其中的一个，所以程序能被编写成使每个子程序有它自己的数据块。任何主程序或子程序能够调用任何数据块，所以 STEP 5 的数据块不对子程序的存储区空间之间的互用提供完善的保护。当数据块激活时，任何随后调用的子程序会默认使用这个数据块，除非子程序调用它自己的数据块。当每个子程序结束，返回控制到调用它的程序时，在调用之前活动的数据块会再次自动地激活。

在 STEP 5 中使用三种类型的主程序和子程序：

1）组织块（OB 1 到 255） 规定作为主程序使用，去组织仅由有条件和无条件的 FB、PB 或其他的 OB 的调用组成的 PLC 操作。当需要时某些 OB（如 OB1、OB2、OB13）能被操作系统自动调用。

2）程序块（PB 1 到 255） 规定用来做简单的控制程序但不提供任何 OB 没有提供的 PLC 程序功用。OB 和 PB 作为子程序被调用时不需要包含标号来声明它们是子程序并且不需要返回指令。当 OB 或 PB 函数结束时，PLC 会自动继续运行调用 OB 或 PB 的程序。

3）功能块（FB 1 到 255） 允许比 OB 和 PB 有更多强大的指令和存储访问功能。例如，只有在 FB 中，程序才能读或写 RS 存储区去改变 PLC 的配置，或使用跳转指令跳过载入和传递指令使它们变成有条件的。FB 使用起来更困难一些。

为了在程序中包含功能块接下来的步骤是必需的：

1）建一个功能块（FB），它必须在被另一个程序调用 FB 的指令编写之前完成。当你创建 FB 时，编程软件将提示你输入：

- 函数名（不超过 8 个字符）。在下面的例子中，名称 Control 已经被指定。你可以不输入名称：只要按下确认键。

- 不超过 40 个声明参数。在下面的例子程序中 4 个参数已被声明。你不需要输入任何参数：如果你不再有参数要声明按下确认键。在声明参数时，要声明形式变量名称。然后 FB 程序可以使用变量名替代它要操作的数据地址。提示，在 DECL 中：每个参数由 3 个参数域组成，它们要在同一行中输入。

形式操作数　　参数类型　　数据类型

其中：

- 形式操作数是变量名，长度不超过 4 个字符。使用一个名称代表数据值或数据地址，将在后面的例子中介绍。在下面的例子中，第一个形式变量名的声明是“sw1”。
- 参数类型可以是：
- I，表示输入参数。当 FB 被调用时，调用程序必须提供现行的 FB 能够使用的数据值的地址。（这个地址可以不必是输入映像地址。）
- Q，表示这个函数将返回一个输出参数，当 FB 结束时，函数返回到被调用程序指定的地址。（可以不必是输出映像地址。）
- D，表示调用程序在调用 FB 时将提供常量输入参数（用常量值代替值的地址）。
- B，表示调用程序将提供在 FB 执行期间由形式变量名所代表的数据块编码、功能块号码或者程序块号码组成的输入参数。
- T 或 C，表示调用程序将提供形式变量名所代表的由定时器或计数器地址组成的输入参数。
- 数据类型是：
- BI，BY 或者 W（如果参数类型是 I 或 Q），表示输入或输出参数的大小（可以是位、字节或字）。来自输入、输出、标志位或数据块的数据可以被 FB 传递或返回。
- KF，KH 等等（如果参数类型是 D），表示调用程序将提供的常量的格式。
- 如果参数类型是 T、C 或者 B，则不需要有数据类型标志。这些数据类型意味着由调用程序提供的数据将是定时器编码、计数器编码或者是数据、函数、程序、组织块的编码。

在命名函数和声明变量后（或者不这样做），你就可以输入程序到功能块中。你就能够在 FB 中使用比在 OB 或 PB 中功能更强大的指令。STEP 5 中把这些额外的指令（其中的一些仅能在 STL 中编写）叫做辅助指令和系统操作，它们包括如下的指令：

- 在 FB 内跳转；
- 有条件定时器和计数器控制操作；
- 使用定时器或计数器的累加值和数据块中的数据来执行位逻辑；
- 每次利用整个的数据字的逻辑操作和移位操作；
- 改变 PLC 的 RS 数据区，来改变 PLC 的配置，包括 PLC 如何处理中断。

如果使用参数传递，功能块必须用形式变量名给出参数数据的地址。任何形式变量名前必需要有“=”（例如下面的例子中的“=sw1”）。如例子中所示的，在 FB 中使用已命名的参数来代表数据或数据地址，当它在结构化 PLC 程序中被多处调用时，通过拥有不同的输入和输出地址，可以使 FB 被重复使用。FB 将在它传递的任何数据上执行同样的操作。

2）在将调用 FB 的程序中，程序 JU 或 JC 可以通过功能块序号来声明调用功能块。编程器在你建立 FB 时将立即显示你输入的函数名，并且提示你为每一个在编写 FB 时声明的

变量输入实际地址或者常量值。

下面的功能块例子，FB001，声明了4个有形式变量名的参数。它把前两个（sw1和mem）当作从主程序接收的输入参数（I），把第三个（act）当作FB将返回到主程序的输出参数（Q），把第四个（dat）当作块（B，这个例子中是数据块。）。所以的数据类型都是单个的位（BI），除了数据块参数，它不需要指定大小。FB001的程序是否改变输入参数（act），取决于两个输入参数的值；使用DO命令[⊖]调用新的数据块（dat）；从到逻辑输出模块（QW072）的数据块中复制数据字（DW001）；然后结束。注意如果你愿意的话可以在功能块中使用绝对地址（DW001和QW072）。（黑体字指示了程序员必须要输入的内容。）

```
FB001
   NAME  :  control
   DECL  :  sw1   I   BI
   DECL  :  mem   I   BI
   DECL  :  act   Q   BI
   DECL  :  dat   B
         :  A     = sw1
         :  AN    = mem
         :  =     = act
         :
         :  DO    = dat
         :  L     DW001
         :  T     QW072
         :  BE
```

调用FB001的程序应与下面的相似。在编写跳转指令调用FB001期间，编程器提示程序员输入在FB中声明的参数地址。（程序员要输入的都已用黑体显示。）

```
      :  C  DB002
      :  JU  FB001
name  :  control
sw1   :  I4.2
mem   :  F3.2
act   :  Q5.4
dat   :  DB006
      :
      :  L  DW001
      :  T  QW74
      :  BE
```

程序例子在调用FB001之前先调用了DB002（数据块2）。在FB001已被调用并结束后，Q5.4将被FB001提供一个新值，来自DB006中的DW001的值将作为类似值被复制到QW72输出，并且在这以后主程序将复制DB002中的DW001的值到QW74中。（因为在FB001被调用前DB002是活动的数据块，所以在FB001结束后它将再次激活。）

另一个“JU FB001”指令可以放在结果化程序的其他地方，那里程序员可以通过输入

⊖ STEP 5中DO命令只能与形式块参数（OB、PB、FB、SB、DB）一起使用。如果形式参数代表组织块、程序块、功能块或顺序块、DO意味着JU（跳转）。如果形式参数代表数据块，DO意味着C（调用）。

完全不同的一系列地址来回应参数的提示。FB001 将接收这些不同的地址；为它们指定名称如“sw1”、“mem”等；执行它的程序；然后返回结果。在使用不同的数据值和/或地址时能执行相同的一系列操作，只要 FB 被传递参数并且使用了形式变量名。

Siemens 的一些 FB（240 到 251）被预先编写好（在 ROM 中）用来执行用户可能会使用的特殊任务。如果调用这些块，程序就必须为每个在预编写的功能块中声明的参数提供现行的常量或现行的地址。其中的一些功能块已在书中描述，或者可以查看每个预编写的 FB 所需要的指定参数的手册。

也有一些 Siemens 的 OB（31 到 251）被预先编写好（在 ROM 中）用来执行用户可能会使用的特殊的任务。如果程序调用 OB251，OB251 将读写数据字，所以程序必须在特殊表格中提供数据块并且在调用 OB251 之前要打开其中的一个。阅读第 12 章，查看对 OB251 所需要的数据块结构的解释。

一些 OB（1、2、3、13、20、21 和 34）没有被预编，但是它们是存在的，当 STEP 5 需要它们运行时它们将被 STEP 5 的操作系统调用，而不管程序员打算把它们编写用来干什么。阅读第 11 章，查看为何这些 OB 会执行的解释。

8.3.3 Siemens STEP 7 的函数调用

接下来，先让自己摆好一个舒适的姿势以准备多花一点时间来阅读下面的内容。如果你一直在想快点喝上一杯咖啡，现在正是时候。STEP 7 要求程序员用使结构化编程更简单和更强大的程序准备过程，但是它与大多数 PLC 程序员习惯的编程方式有很大的不同。在我们进入实际的编写结构化程序之前在几分钟里描述一下需要的结构。

1. 工程和 CPU 程序

在输入任何程序之前，程序员必须首先使用 STEP 7 编程软件创建一个工程，然后在工程内部创建（最少）一个 PLC 站点（在西门子的文献中有时叫做 CPU 程序）。程序员必须为站点中的 PLC 系统输入配置信息，完成后输入程序，Siemens 称它为逻辑块，并且输入逻辑块所需要的数据块。在逻辑块和数据块的编写之前或之间，程序员通过向符号表输入内容来定义可以在这个 PLC 的任何程序中使用的符号名称。配置和符号表成为被保存在编程器中同时又将被下载到 PLC 中的站点的一部分。

拥有包含站点的工程观念对由单个 PLC 组成的系统程序员来说似乎是不必要的，尽管拥有由分离配置、逻辑块、数据块和符号表成分组成的站点甚至在简单的系统中都是有意义的。现在许多控制系统至少由一个拥有智能 I/O 模块和远程 I/O 模块的中心 PLC 组成。工程结构开始变得更有意义，当你认识到现在许多 PLC 通过多重局域网在多重 PLC 互连的制造系统中使用，并且它们一起工作。

单一的工程（如制造设备）应该由几个站点构成，或者甚至由多重的子工程组成，这些子工程每个又包含几个站点。被局域网互连的 PLC 系统的每个站点内的配置数据都必须包含配置文件，这些配置文件用来描述 PLC 的通信连接和局域网中惟一的地址。STEP 7 的工程也能够包含全局数据表，它定义了在 PLC 存储器和网络中其他 PLC 存储器器之间局域网必须复制的数据项目。在整个系统中使用工程系统定义结构、互连和数据交换需要意味着程序员不必要分别地编写或配置每个 PLC 来交换数据。我们将在第 13 章中详细地查看局域网和全局数据。

在图 8-10 中显示了一个简单完整工程的例子（Project _ A）及其组成部分⊖。Project _ A 中包含一个符号表，两个局域网（MPI Network _ X 和 Profibus-DP Network _ Y），其中的一个包含全局数据表，两个站点（S7-300 Station _ AA 和 S7-300 Station _ AB），还有一个子工程（Sub _ Proj _ AC）。站点 Station _ AA 包含有描述 PLC 控制下的硬件和硬件初始化参数的配置数据的系统数据块。Station _ AA 还包含两个智能模块，一个是连接到特殊网络的通信处理器，另一个（当然）是 CPU 模块。CPU 模块包含两个逻辑块（OB1 和 FB4，分别叫做 Program _ 1 和 Program _ 2）、一个公用数据块（DB2）和一个实例数据块（DB3）。

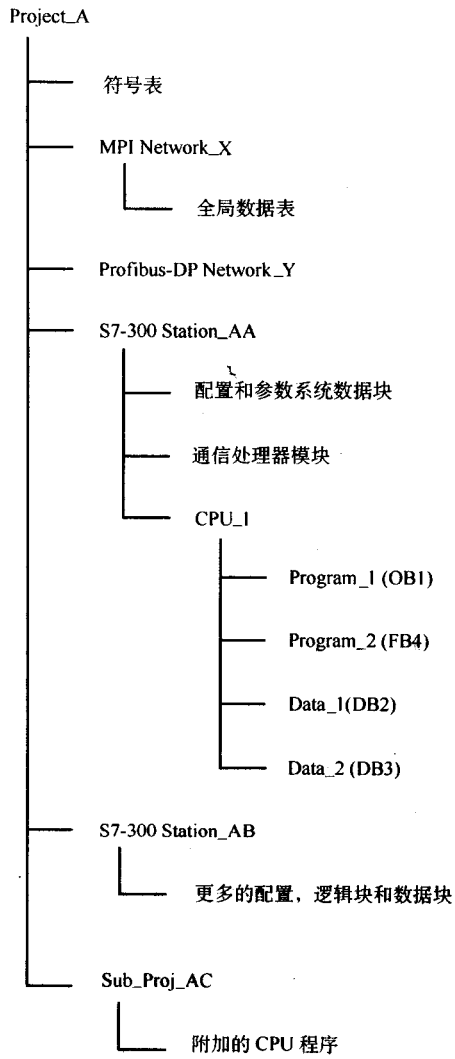


图 8-10 简单的 STEP 7 工程的构成

⊖ STEP 7 编程软件屏幕显示的项目树有一点不同。一些组件（如硬件配置）的项目只有在主窗口中被选中时才能在二级窗口中显示。

2. 站点的配置数据

工程创建和站点配置将在第 10 章中介绍，在这节中我们给出一个简单的概述。

在工程创建中，程序员必须输入描述 CPU 的类型和位置、供电模块和所有的 I/O 模块（包含信号模块（SM）、函数模块（FM）和通信过程（CP））的数据。通过 STEP 的对话框[一]，参数都被指定到需要进一步配置的模块中。STEP 7 的编程器把配置数据集成到系统数据块（SDB）中，然后把它们再下载到 CPU。CPU 在它的保持存储器中存储系统数据块并且每次 PLC 被启动后重写模块配置数据到合适的模块中，所以一旦配置数据被输入，程序员就不必再担心它直到 PLC 的系统需要改变时。

3. 调用组织块的 STEP 7 操作系统

默认情况下，当 S7 PLC 在运行模式时，每个循环周期中它都把组织块 001（OB1）当作用户程序来执行。程序员在结构化程序中通常使用 OB1 作为主程序并且对 OB1 进行编程使它包含有条件和/或无条件对子程序的调用。

无论何时 S7 PLC 转换到运行模式，它总是寻找组织块 100（OB100）。如果 OB100 存在，在它执行扫描循环之前，STEP 7 的操作系统将执行一次它包含的程序。OB100 应该被编成，例如，使它执行任何以前的程序员要 PLC 每次转换开始时执行的操作。

STEP 7 的操作系统也自动的尝试执行在确定条件下的其他组织块。其他组织块和它们运行的条件将在后面关于配置选项的小节中描述，在第 11 章中还会有更详细的介绍。

4. STEP 7 中的结构化编程

完整的 S7 PLC 用户程序在组织块 1 中（OB1）可以被当作单程序编写[二]，但是 STEP 7 允许并提倡结构化编程，在程序里主程序（例如 OB1）可以调用子程序而且依次的子程序可以调用其他子程序。西门子所涉及的主程序和子程序如逻辑块有 3 种类型：

1）组织块（OB）是能够调用其他逻辑块的主程序。OB 执行会响应 STEP 7 的操作系统的调用。例如，每个扫描循环 OB1 被调用一次，用户不能编写到 OB 的调用但是能够在 OB 中输入程序。

2）功能块（FB）能够从任何用户可写逻辑块中调用，甚至可以从另外一个功能块中调用。与西门子 S7 PLC 一起提供的一些预编写功能块[三]，叫做系统功能块（SFB）。每次对 FB 或者 SFB 的调用必须包含用 FB/SFB 打开的立即数据块的名称。下面的例子展示了用立即数据块 25 调用功能块 3 和用立即数据块 125 调用系统功能块 4 的 STL 指令：

```
Call  FB 3,  DB25
Call  SFB 4,  DB125
```

3）函数（FC）能够从任何用户编写的逻辑块中调用，甚至从另一个函数中调用。Siemens S7 PLC 的预编程序中包含一些函数，称为标准函数（FC）和系统函数（SFC）。下面的例子显示了调用函数 20 与调用系统函数 46 的 STL 指令。

[一] 如果用户购买了一个 Siemens 称为功能模块（FM）或通信处理模块（CP）的特殊 I/O 模块，这个模块可以在安装时成为 SETP 7 编程软件的一部分，并包含了必要的参数配置。

[二] 如果你正在使用 STL 编程，你甚至可以将整个程序输入一个网络，但是将一个逻辑块分开放入多个网络可以提高程序的稳定性，并且不会影响它的性能。

[三] 检查你的 CPU 模块的特性页，并不是所有的 CPU 都有 SFB。

```
Call FC 20
Call SFC 46
```

Siemens S7 PLC 中仅有 4 个标准函数，它们是：FC4、FC6、FC7 和 FC8，所有的预编程序都在日期和次数与被称为 DATE _ AND _ TIME 的标准常数格式之间进行转换，这些已经在第 5 章中描述。标准函数要求程序传递参数到 FC。请查看手册中对参数的介绍。

在 Siemens S7 PLC 中有相当多的系统函数（SFC），它们都被预编写用来进行一些如数据集移动的操作，系统函数与看门狗定时器或系统时钟（Siemens 中对实时时钟的称谓）一起运行，并对中断进行控制等等。大部分系统函数都要求程序传递参数。（在上面的例子中使用的 SFC 46，仅仅使 PLC 转到停止模式，它并不需要参数。）在其他的章节中还包含另外的一些系统功能块，查看手册以便了解完整的系统功能块列表和它们所需要的参数。）

5. STEP 7 结构化程序中的参数传递

程序员能在整个结构化程序的逻辑块中使用公用存储区（输入/输出映像、位存储区、公用数据区等等）以便让所有的逻辑块都共用相同的地址空间，但是 STEP 7 允许并提倡逻辑块之间的参数数据传递，以便每个逻辑块在它自己的存储空间中控制它自己的数据，并且不能偶然的改变其他逻辑块需要的数据值。可以容易的读和修改包含有符号名的参数声明程序，并且在程序中使用符号名替代绝对地址。

参数传递中关键是每个逻辑块都能够包含的变量声明。当用梯形图或 STL 语言进行编程时，程序员在变量声明表中输入变量的声明，其中变量声明表是每个逻辑块的主要部分（逻辑块中其他的部分是对数据进行操作的程序。）。系统功能块（SFB）、系统函数（SFC）和预编标准程序（FC）当然是与变量声明表一起被预编好。逻辑块中的变量声明表如表 8-1 所示的那样（它一般会在右上角显示逻辑块的名称），将在下面作出说明。

表 8-1 STEP 中的变量声明表（好像在寻找功能块）

声明表（工程/CPU 程序/逻辑块名称）					
地址	声明类型	名称	数据类型	初始值	备注
0.0	in	switch _ 1	BOOL		
0.1	out	cylinder	BOOL	False	
2.0	in _ out	loop _ cnt	INT	20	
4.0	stat	work _ dat	INT	0	
0.0	temp	push _ cnt	INT	0	
2.0	temp	my _ pie	REAL	3.14	

第 5 章中当介绍 STEP7 的立即数据块时，我们已经见到表 8-1 中所示类似的变量声明表，但是这次我们从参数传递和保留逻辑块所使用的存储区的角度来查看变量声明表的内容。表 8-1 中的变量声明表包含六个变量声明的记录，每个记录包含以下的几栏：

1）当逻辑块执行时，为数据元素保留的存储区的绝对地址（Addr.）。程序员不必输入绝对地址的数值：STEP 7 的编程软件在程序员完成输入每个表行数据后自动分配它们。在

这个功能块变量声明表例子中，第一个立即数据块字节中的两位被保留用来表述两个布尔数据单元的转换和激励的状态，然后立即数据块中四个以上的字节（从第二个字节开始）被保留用来存放两个 INT 值（整数数值）。注意，在第四个记录后地址的编号方式从 0.0 重新开始，这是因为接下来的记录在本地堆栈存储区中为整数数值（两个字节）和实数值（四个字节）保留空间。在功能块中，变量声明表必须总是首先声明立即数据块变量（表 8-1 中的前面的四个变量），然后为本地堆栈存储区域声明“temp”变量（表 8-1 中的后两个变量）。组织块和函数中不可以有立即数据块，因此所有它们声明的变量都在本地堆栈存储区中被分配空间。

表 8-1 中包含变量声明表的程序可以使用绝对地址来存取立即数据块或本地堆栈存储区中的数据。例如，地址 DIX0.1 可以被用来检测或输出到第二条记录（变量名为 # cylinder）中声明的布尔位。地址 DIW2 可以被用来操作第三条记录（# loop _ cnt）中声明的整数数值。（在后面将解释字符“#”的意义。）本地堆栈存储区中的实元素（# my _ pie）能够用它的绝对地址 LD2 赋予地址值。

2）*声明类型*栏标明了每个数据值的源、目的地和生存期。数据元素在变量声明表中必须依照下面的次序声明：

- 被另外的逻辑块调用时，In 参数是传递到该逻辑块的数值。传递到子程序的数据值（如常量、地址或指针）必须与在子程序的变量声明表中所声明的变量数据类型相匹配。在西门子的文献中，“in”参数也被描述为 VAR _ INPUT 类型。
- 程序结束时，Out 参数使该逻辑块返回变量值到调用逻辑块。调用逻辑块必须指出子程序写输出参数所需要的（合适的大小）地址。“out”参数也被描述为 VAR _ OUTPUT 类型。
- In-out 参数从调用逻辑块接收初始值并且返回变量的最终值到调用逻辑块中。调用程序必须提供地址或指针。它也被描述为 VAR _ IN _ OUTPUT 类型。

组织块不能使用任何 in、out、in-out 参数。功能块复制 in 和 in-out 参数值到与功能块一起被调用的立即数据块，然后从立即数据块复制 in-out 和 out 参数值到调用逻辑块作为功能块的结束。函数复制 in、out、in-out 参数指针到本地堆栈存储区中，以便当函数运行时它实际读和写的是存储区中被调用逻辑块指定的特定区域的值。如果调用逻辑块不传递参数，在立即数据块中的预设值（如果存在）将被使用，并且如果没有立即数据块，则变量声明表中的初始值将被使用（下面将介绍初始值）。如果上面所列的各项都没有提供数值，则变量被赋予数值 0。

- Stat 参数在逻辑块调用期间中不用来传递参数。“stat”参数仅能在功能块的变量声明表中声明。声明 stat 变量将在立即数据块中保留空间，以便让程序能在立即数据块中储存变量值直到下一次功能块被调用。“stat”参数也被简称为 VAR 类型。
- temp 参数在逻辑块调用期间也不用来传递参数。对 temp 参数的声明仅仅在本地堆栈存储区中保留空间直到逻辑块结束，当保留的本地堆栈存储区空间变回一般聚集区时，存储在里面的值将丢失。它也被称为 VAR _ TEMP 类型。

3）*本地块符号名*（Block-local symbolic name）应该要被输入（但不是必须）。如果输入了，这个逻辑块程序能够使用“#”后面加上本地块符号名标明数据项的地址，而不必用

绝对地址来标明数据项的地址。复杂的数据元素仅能使用符号名来标明地址。在程序中使用符号名允许程序员通过改变变量声明表中的某个地方的实际地址来修改程序地址，而不必在程序中的许多地方进行修改。下面的例子显示了表 8-1 中变量声明表的数据如何使用它们的本地块符号名来标明地址：

```
# switch_ 1      means the same as DIX 0.0
# my_ pie        means the same as LD 2
```

如果没有在左边标明“#”前缀，STEP 7 的编辑程序将添加它到你的程序，前提是符号名与在变量声明表中声明的名称相称。

STEP 5 也允许程序员使用符号表来为共享的存储器地址定义（非局部的）符号名。如果它们被双引号括住（如“big _ data”），（非局部）符号名能够在 S7 PLC 运行的任何逻辑块中使用。如果没有双引号，则在符号表中搜寻之前，STEP 7 将先在变量声明表中寻找该符号名的本地块声明。当在标明非局部符号名时或在程序中碰巧用相同的名称来标明本地块的地址时，确信要包含双引号。

4）*数据类型*栏定义了必须为数据项保留多少立即数据块或本地堆栈存储区，*初始值*栏允许程序员为一些数据项类型指定常值形式的初始值。允许的数据类型和常值形式包括在第 5 章中描述的类型，包括：

- 基本的数据类型：
  BOOL
  BYTE 和 CHAR
  WORD 和 DWORD
  INT 和 DINT
  REAL
  TIME、DATE、TIME _ OF _ DAY（TOD）和 S5TIME
- 复合数据类型（复合数据类型不能被赋予初始值）
  DATE _ AND _ TIME (DT)
  STRING
  ARRAY
  STRUCT
- 用户所拥有的任何复合数据类型都是通过创建*用户数据类型*（UDT）数据块来定义的。数据类型栏要包含有编号的 UDT 数据块（如 UDT3）。

数据类型也可以是被西门子描述为*参数数据类型*的数据类型之一，它包括下面所列的几种（不可以赋予初始值）：

1）*定时器*：使被调用逻辑块保留两个字节的存储区并且接收来自调用逻辑块的由字母“T”和 0 到 255 之间的数字组成的参数值，来标明定时器。

2）*计数器*：与定时器参数类似，但是调用逻辑块必须传递由字母“C”和 0 到 255 之间的数字组成的值来标明计数器。

3）Block _ FB，Block _ FC，Block _ DB，Block _ SDB：它们当中的每一个都会使被调用逻辑块保留两个字节的存储区并且接收来自调用逻辑块的逻辑块（FB 或者 FC）或者数据块（DB 或 SDB）的编号。调用逻辑块必须用参数的全名来传递参数：块类型标识符（FB、FC、DB 或

SDB）后面由 0 到 255 之间的数字组成，或者调用逻辑块通过块的符号名来指定某个块。

4）指针：使被调用逻辑块保留空间用来存储指针常量形式的地址说明。无论什么时候当被调用逻辑块读或写值到该变量中时，它将实际读或写到由调用程序指定的地址。指针可以总是用来传递值到函数和功能块中，但是不必要被声明为“Out”数据声明类型。（请查看手册。）记住指针参数在存储器中占据 6 个字节，存储的格式如图 8-12 中所示。存储器区域允许的编码包含指出“前一个”本地堆栈存储区域的编码。这意味着逻辑块可以调用函数（或功能块），而且可以传递指针参数，该指针参数允许函数（或功能块）访问属于调用逻辑块的本地堆栈存储器。

5）任意型（ANY）：在这种情况下被调用逻辑块将为 ANY 指针保留 10 个字节的空间。因为 ANY 指针包含它所指向的数据类型的描述和数据项的编号，所以调用逻辑块可以用它来传递任何基本的或复合的数据元素到被调用逻辑块中（例如，如果该逻辑块在主程序中被两次调用，主程序就可以在第一次调用中传递 ANY 指针到整数值，而在第二次就传递 ANY 指针到数组。）。像标准的指针参数一样，ANY 指针参数可以提供函数（或功能块）访问调用程序使用的不同存储区域，包含调用程序的本地堆栈存储区。

6. STEP 7 的调用指令和中止逻辑块指令的使用

有两个梯形图指令可以被编写用来调用另一个逻辑块，但是 STL 语言中有三条这样的指令。梯形图也提供两条指令来结束逻辑块，但是 STL 语言中则有多种不同的方法来结束。

梯形图中的 CALL FC/SFC FROM COIL 指令可以有条件地或无条件地调用没有 in、out、in-out 参数的函数（FC），在这种情况下程序将显示如下面所示的形式：

FC11

---------- (Call)

除了调用函数的情况下，上面所示的梯形图的梯级也会影响状态字中的一些状态位（请查看手册）。

图 8-11 为 STEP 7 的指针格式。

字节：																		字节：
0	DB 编号（如果不是 DB 则编号为 0）																	1
2	存储区域的代码								0	0	0	0	0	b	b	b		3
4	b	b	b	b	b	b	b	b	b	b	b	b	b	x	x	x		5

图 8-11 STEP 7 的指针格式

为编写梯形图中对已经声明了 in、out、in-out 参数的 FB、SFB、SFC 的调用或者 FC 的调用，程序员可以使用 STEP 7 的浏览器把逻辑块当作指令一样来选择。（明显地，正在被调用的逻辑块必须已经被编写好。）梯形图的元素如图 8-12 中所示的那样。调用程序必须为被调用逻辑块所需要的 in、out、in-out 参数提供数据常量或地址。用西门子的术语来说，变量声明表创建形式参数，调用程序必须提供实际参数。在图 8-12 例子中梯形图的调用是对有表 8-1 中所示的变量声明表的功能块的调用。图 8-12a 中显示了在程序员输入实际参数（或实例数据块）之前梯形图形式和与之对等的 STL 语言指令，而图 8-12b 中显示的是在程序员输入典型的实际参数后的相同程序。

注意，所有的 in 和 in-out 参数都在梯形图元素的左边显示。In 参数的实际参数可以是

常量或地址，但是 in-out 参数则必须是地址。Out 参数在梯形图元素的右边显示，并且其实际参数必须作为地址输入。当你输入实际参数地址是，有几条规则要记住：

1）基本数据元素的地址可以是绝对地址、符号名或指针形式。

2）复合数据元素的地址必须是指针常量或符号名。

3）符号名不能被用来传递：

- 复合数据，如果调用是由函数组成。
- 指针或 ANY 指针，除非调用是由组织块组成。
- 定时器、计数器或块，如果函数正从另一个函数中被调用。

图 8-12 中的梯形图元素包含标为 EN 的布尔输入，程序员可以用逻辑运算使调用有条件的执行。梯形图元素也包含标为 ENO 的输出，它提供了程序员可以用来控制另一个操作的布尔输出，即使 ENO 输出不包含在表 8-1 所示的对 FB3 的变量声明表中。所有的梯形图函数和功能块都有这个 ENO 输出，它输出状态字中的 BR 状态位的状态。Siemens 提供的所有 SFC 和 SFB 程序都包含复位 BR 位的逻辑，任何原因的 SFC 或 SFB 执行失败就使 BR 位复位，否则使 BR 位置位。在梯形图中编程时，在所编写的函数或功能块中包含控制 BR 位指令会使编程更加方便。如果没有这样做并且程序中包含来自 FC 或 FB 的 ENO 输出的逻辑状态控制的梯形图语句，那你的程序可能会出现很大的问题。

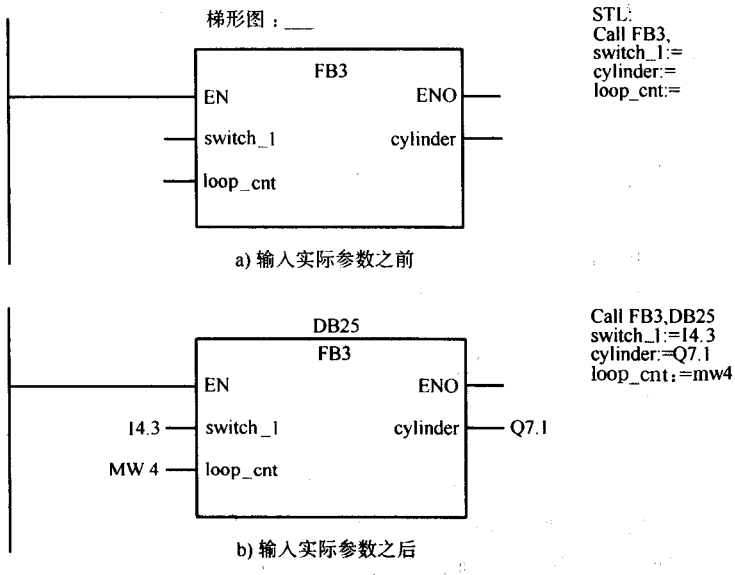


图 8-12 调用 STEP 7 的包含如表 8-1 中所示的变量声明表的功能块

为了控制 BR 位（也因此控制 ENO 输出），可以在所编写的每个 FC 或 FB 程序结束时包含保存（SAVE）输出元素或返回（RETURN）输出元素。保存指令仅影响 BR 位，而返回指令还会影响其他的状态位（请查看手册）。如果保存或返回指令被编写成无条件执行，BR 位将一直被置位。如果保存或返回指令在程序中是有条件执行（如图 8-13 所示），BR 位仅在条件语句的 RLO 为真时被置位。

Siemens 的 SFC 和 SFB 一般包含名为 RET _ VAL 的输出参数，它包含代码来确定为何

出错。错误代码将在第 15 章中介绍。如果你想的话，你可以包含名为 RET _ VAL 的输出变量，并且可以编写你的逻辑块使它写入 RET _ VAL 中。



图 8-13 STEP 7 的返回条件

调用其他逻辑块的三个 STL 指令包括无条件调用（UC）、条件调用（CC）和调用（CALL）指令。调用指令是无条件执行的，所以它等同于图 8-12 中所示的梯形图中的无条件调用。调用指令必须用来调用已声明了 in、out 或 in-out 参数的逻辑块。由于调用指令是无条件执行，所以如果想使调用指令有条件执行，必须编写一个有条件跳转，使它跳过调用指令，就像我们在这章前面的部分看到的那样。

UC 指令与调用指令类似，只是 UC 指令仅能被用来调用不需要任何的 in、out 或 in-out 参数的 FC 或 FB。使用 UC 指令的一个优点是拥有 UC 指令的 FC 或 FB 的编号可以以间接形式在存储区中被指向(如 FC5 可以用指令 UC FC [MW4] 来调用，如果 MW4 中包含数字 5)，或者 UC 指令能够调用编号已经被传递到该逻辑块中作为 BLOCK _ FC 类型的参数的函数（如 UC ＃my _ prog 调用 FC5，如果当传递到该逻辑块时，变量名＃my _ prog 是实值为 5 的 BLOCK _ FC 类型的参数)。

CC 指令与 UC 指令类似，只是它可以跟随在条件语句的后面，该语句将决定 CC 指令是否导致函数被调用。如果 CC 指令没有使 FC 被调用，则 CC 指令不影响 RLO。

如所有的 Siemens SFC 和 SFB 程序一样，编写在结束时使 BR 位置位或者复位的 STL 函数或功能块是可能的。STL 中的保存指令复制 RLO 位状态到 BR 位，所以如果保存被逻辑语句控制，则 BR 位将存储逻辑语句的结果。另外，也可以在无条件执行保存指令之前用 STL 指令 SET、NOT 或 CLR 来置位、取反或清除 RLO 位实现对 BR 位的控制。

所有的 STL 逻辑块必须有块结束（BE）指令作为它们的最后指令。当 PLC 执行这条指令时，它总是终止现行的逻辑块并且返回控制到调用该函数或功能块的逻辑块，或者返回控制到 STEP 7 的操作系统，如果该逻辑块是将要结束的组织块。STL 程序也能够包含无条件块结束（BEU）指令，它与 BE 指令有相同的效果。但与仅能是最后指令的 BE 指令不同的是，BEU 指令可以在逻辑块中的任何地方被编写。在 STL 程序中，程序员也可以编写条件块结束（BEC）指令。如果当 BEC 执行时 RLO 为真，在该逻辑块中没有进一步的指令要执行，这正像执行了 BEU 指令一样。

STL 语言的 FC 或 FB 也能够声明 RET _ VAL 参数和写入错误代码，以便它们像西门子写好的 SFC 和 SFB 一样操作。如果使用文本编辑器替代 STEP 7 的梯形图或 STL 编辑器来编写函数，你可以为每个所写的函数声明数据类型，然后（除非声明的类型是空值型（VOID））在将文本文件转译为 S7 PLC 可执行程序时，STEP 7 将自动添加记录到变量声明表来创建一个名为 RET _ VAL 的输出参数。㊀

㊀ 在本书中我们没有覆盖以文本格式编辑程序，文本格式编程基于 STL，是容易学习的，并且提供了一些其他编程方式不能提供的优点，例如可以让你创建一个 STEP7 拒绝显示的程序。

7. STEP 7 的结构化编程对微处理器寄存器的影响

无论何时逻辑块调用另外的逻辑块，STEP 7 的操作系统会自动地存储来自一些微处理器寄存器的信息（关于逻辑块处理调用的信息）到被称为块栈（B 栈）的存储区域里，以便当被调用逻辑块结束时使该逻辑块能够继续执行。存储的信息包括：

1）程序中调用的逻辑块和下一条指令的名称

2）在调用时处于活动状态的共享和立即数据块的编号

3）本地堆栈中调用逻辑块开始时的地址

块栈可以包含八条这样的数据，它限制了 S7 PLC 的嵌套使用的深度，使 S7 PLC 只能有八层逻辑块对其他逻辑块调用。㊀

如果正被调用的逻辑块是功能块，STEP 7 的操作系统将在把新的立即数据块的编号放入寄存器之前转换包含立即数据块和共享数据块的编号的微处理器寄存器中的内容（特别的是使立即数据块转换为共享数据块）。

当被调用的逻辑块结束时，块栈中的内容被放回到取出它们的微处理器寄存器中，所以调用逻辑块从调用发生之前的正在使用的数据块和本地堆栈存储区的访问地址处开始继续执行。

调用和执行另外的逻辑块会影响一些微处理器寄存器，程序员应该注意这点。如果被调用逻辑块操作复合数据元素，或者即使它的变量声明表声明了如 in、out、in-out 参数的复合数据元素，STEP 7 的系统必须使用公用数据块寄存器和地址寄存器 1。这些寄存器的内容在它们使用之前应该一直存储在被调用逻辑块中。从被调用逻辑块返回后，没有在块栈中保存的微处理器寄存器可以被被调用逻辑块改变内容。这些寄存器包括累加器、地址寄存器和状态字的一些位（包括 RLO 位）。一些状态字的位会自动地被调用改变（请查看手册）。

8. STEP 7 结构化程序中的符号

符号名可以在 CPU 程序的符号表中定义，也可以在逻辑块的变量声明表或数据块定义中定义。㊁符号名必须由字母字符开始并且区分大小写（如“1spot”不是有效的名称，符号名“onespot”与“Onespot”是不同的）。

表 8-2 中显示的是一个符号表示例。符号表可以包含分配符号名到 I/O 地址（I 和 Q 存储区域，而不是 P 存储区）、位存储区（M）、数据块（DB 和 UDT）、定时器和计数器（T 和 C）、甚至是到逻辑块（OB、FB、FC、SFB 和 SFC）的记录项。数据类型栏让 PLC 知道每个符号名所代表的数据项是多大。符号名在符号表中被定义了之后，就可以在 STEP 7 的编程软件中的任何地方被用来替代它所代表的绝对地址，只要你继续使用相同的编程器。符号名保存在编程器的存储区里面，而不是在 PLC 的存储区里。在用户程序中，符号名可以被输入在双引号之间（如“big _ var”）。编译器在没有找到类似的以命名块的本地符号的情况下，将添加双引号。

㊀ STEP7 操作系统可以中断一个结构化程序来使另一个组织块执行。每一个组织块代表一个优先级，并且每一个优先级有它自己的 B 堆栈，所以即使被中断的组织块已经有几层调用，新的组织块仍可以有八层新的嵌套调用。

㊁ 你不能在创建数据块时为数据块和用户数据类型声明符号名。他们必须在符号表中声明。

表 8-2 典型的 STEP 7 的 CPU 工程中的符号表

（工程和 CPU 程序的名称）\ 符号			
符号	存储区地址	数据类型	注释
Cylinder _ 1	Q 7.1	BOOL	坏的部分释放
Bad _ parts	C 6	COUNTER	坏的部分数量
GoodBad _ rat	MD 4	REAL	比率：坏/好
Template _ 1	UDT 5	UDT 5	数据结构
Program _ AB	FB 5	FB 5	检查部分质量
Data4 _ progAB	DB 34	DB 34	FB 5 的实例

使用符号名来代表 I/O 地址允许 I/O 触点或模块的位置改变而不需要主程序的改变。在 I/O 触点或模块移动后，程序员在符号表中仅需改变绝对地址一次。任何使用符号名的逻辑块都将使用新的绝对地址。指定符号名到非 I/O 地址提高了程序的可读性并且允许为一种类型的 PLC 编写的程序可以被另一个不同存储结构的 PLC 更容易地修改。

在逻辑块的变量声明表中声明的或在数据块中声明的符号名都被认为是本地块符号。本地块符号仅在逻辑块或数据块被定义为开放性时有效。虽然在不同的逻辑块中你可以声明相同的符号名或者声明与已在符号表中定义的相同符号名，但是并不提倡这样做。程序员可以在本地块符号前输入“#”字符（如#small _ var）或让编译器添加字符“#”。在变量声明表中和数据块中声明的变量的符号名都被存储在编程器的存储区中并且不会被下载到 PLC 的存储区中。如果从 PLC 上传了程序到另一个程序单元并且查看该程序，则该程序中不包含符号名。

9. STEP 7 中输入结构化程序的次序

可能已经很明显，STEP 7 中必须依照特殊的次序来编写结构化的程序：

1）必须已经创建工程和站点，并且完成了 PLC 需要的所有配置信息的输入。

2）如果打算使用用户数据类型作为变量声明表中的其他数据块或数据结构的模板，那么现在就创建它。（参见第 5 章）

3）如果想使用在所有的逻辑块中都有效的符号名（例如用来代表输入和输出），现在就开始计划并且在符号表中输入。如果你想的话可以像在后面编写程序一样存取该表来更改或添加项目。STEP 7 的逻辑块中在使用符号名之前并不绝对需要创建该符号名，但是你将不能编译该模块直到你已经定义了符号表中的所有非本地符号。

4）首先由最低等级的函数开始创建函数（FC），然后创建调用已编写好的函数。如果任何函数调用功能块（FB），不要创建那些函数直到你已经创建了它们所需要的功能块（在 6 步后）。如果你想的话，现在你可以用刚才的变量声明表创建函数，然后再输入程序。

5）编写你的功能块（FB），至少要达到可以完成变量声明表的的程度。可以声明类型为 UDT 的数据元素，因为你已经在第 2 步中创建了 UDT。你可以不必要编写 SFB 程序，也不必下载它们到 PLC 中。你的编程器软件和 PLC 中都包含副本。

6）创建实例数据块（DB）。在创建的过程中，你必须指出每个实例数据块中所涉及的功能块。在该功能块中的变量声明表将自动定义实例数据块中的内容。如果有必要的话，为单独的功能块创建几个实例数据块。如果要修改初始值就查看实例数据块，但是不允许改变任何其他的项目。

7）创建调用已经被创建的 FB 和 FC 的组织块（OB）。

8）创建公用数据块（DB）。（你可以在完成第 2 步后的任何时候完成这一步，但是必须要在你运行你的程序之前完成。）

CQM1

8.3.4 OMRON CQM1 的子程序调用

在 OMRON 的程序中子程序被称为子例行程序并且被编写在主程序的底部和 END（01）语句之前。任何程序（包括子例行程序）都可以包含子例行程序输入指令 SBS（91）N，在梯级的逻辑为真时来调用编号在 0 到 255（N）之间的子例行程序，如图 8-14 中所示。可以允许有 16 层的子例行程序嵌套调用。

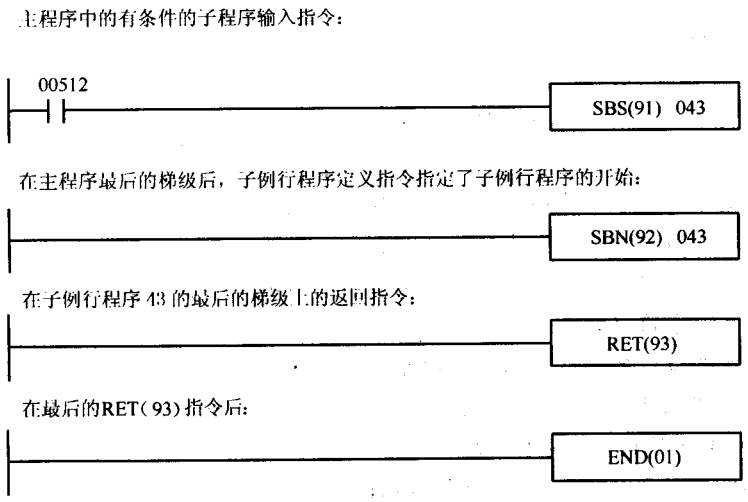


图 8-14 在 OMRON PLC 程序中的子程序调用和子程序定义

每个子程序必须由子例行程序定义（输出）指令开始，SBN（92）N，指出子程序的编号，必须由返回指令结束，RET（93）。遇到的第一个 SBN（92）N 指令告诉 CQM1 主程序将要结束，所以在你的主程序结束前不要输入 SBN（92）N 指令。在最后的子程序的 RET（93）指令后面，需要有一个 END（01）指令。

参数传递不可以通过 CQM1 的子程序调用来完成。所有的 CQM1 的程序和子例行程序都共用相同的数据空间。

CQM1 的子例行程序注释

配置和编写 OMRON PLC 使它的操作系统调用子例行程序有时脱离主程序的控制是可能的。在本章的下一节中我们会简要地查看这些方法，在第 11 章中将有更详细地介绍。为了避免在 SBS（91）指令调用的子例行程序和操作系统自动调用的子例行程序之间可能会出现的冲突，程序员应该避免使用 SBS（91）指令来调用编号为 000 到 003 的子例行程序和 STIM（69）指令所涉及的子例行程序，还应避免在 CTBL（63）指令中所使用的数据文件中的子例行程序。

8.4 影响程序执行的配置

PLC 被期望用来执行 3 个步骤的扫描循环，这使得 PLC 与其他类型的计算机不同。PLC 使用扫描循环，因为这已经被证明允许自动化过程的相对快速、可预测和无故障的控制。现在，运用这章中我们所见的强大的结构化编程特色，程序员编写的程序可能使 PLC

在既不快、不可预测，又不是无故障的方式下运行。

听起来可能会奇怪，许多 PLC 现在都提供让用户改变 PLC 的配置选项以便让 PLC 偏离标准扫描循环。PLC 可以被用来做这样的事情，如在扫描循环中运行不止一个主程序或者定时执行输入和输出扫描，而不是在主程序扫描之前或跟随主程序扫描后马上进行。

大部分现代的 PLC 都可以进行设置以便 PLC 的操作系统能够用精确的时间间隔中断标准的扫描循环或响应输入模块中的改变，从而执行不是扫描循环的用户编写的标准程序。这些选项允许程序员迫使 PLC 在可预测的时间间隔上执行结构化的用户程序，或快速地响应一些输入条件。通过把一些用户程序带离标准的扫描循环，程序有时能够被简化以便它将再次无故障执行。第 11 章中将详细地介绍中断，所以这章我们只描述通过使用中断程序员可以使 PLC 做些什么。

8.4.1 对 ALLEN-BRADLEY PLC-5 中结构化编程配置

PLC-5

默认情况下，程序文件 2 是在 PLC-5 的扫描循环中执行的仅有的 MCP（主控程序），但是 PLC-5 的配置可以被改变用来在每个扫描循环中执行多达 16 个像 MCP 文件一样的程序文件，在每个 MCP 执行期间有或者没有 I/O 扫描。

改变 MCP 程序文件的配置使它包含在扫描循环中可以通过使用 PROC CONFIG 界面来完成。为（不超过）16 个 MCP 中的每个文件输入程序文件编号（1 到 999）。对列表中的每个 MCP 文件，程序员可以指出 PLC-5 在 MCP 文件执行后是否要执行 I/O 扫描（0 意味着包含），并且能够暂时禁止个别的 MCP 执行而不需要把它们从列表中移除（1 意味着禁止）。

当程序被存储到硬盘时，数据表的内容（在状态文件中包含配置数据）也会被保存。当程序被恢复时，包含 PLC 的配置数据的状态文件也被恢复。

MCP 配置可以不使用配置界面，而通过程序来改变。改变 PLC 配置的指令可以包括在每次 PLC 进入运行模式时的调用子程序中，使用*第一扫描位*（S：1/15）来调用子程序。当 PLC 被启动时，这种方法保证了在每次转换时配置被复位一次。高级 PLC 编程可能需要在 PLC 运行时动态地改变 PLC 的配置，这取决于工作地点的传感器状态。为了改变 MCP 的配置，程序必须包含修改下面配置数据的指令：

- 储存在每个第三状态数据字位置的 MCP 文件的编号，从 S：80 开始并延伸到 S：125（在每个 MCP 文件编号后的两个状态数据字都被 PLC-5 用来记录 MCP 最近的执行时间和该 MCP 最大的扫描时间）。
- 控制 I/O 扫描是否跟随在从 MCP A 到 MCP P 的 16 个 MCP 中的每个之后执行的 16 个位都是状态文件位 S：78/0 到 S：18/15。
- 任意的 MCP 都可以通过在 S：79 中设置 16 位中的一位来禁止它执行。

如果 SFC 程序已经包含在 MCP 程序的列表中，PLC-5 可以被配置用来执行用*顺序流程*图语言编写的程序。（程序文件 1 为 SFC 程序被保留，但是任何文件都可以包含一个 SFC 程序。）SFC 编程将在第 9 章中介绍。

在这里简要地列出可配置的*中断属性*（在第 11 章中将详细地介绍）：

- PLC-5 程序可以包含*立即输入和立即输出*指令，它们直接读或写 I/O 模块而不用等待输入扫描或输出扫描。
- PLC-5 可以被配置用来以精确的时间间隔中断扫描循环来运行可选定时中断（STI）

程序文件。

- PLC-5 可以被配置用来中断扫描循环，执行处理器输入中断（PII）以响应输入模块中输入状态的改变。

SLC 500 8.4.2 ALLEN-BRADLEY SLC 500 的结构化编程配置

默认情况下，程序文件 2 是仅有的主控程序（MCP），并且用户不可以改变。程序员仅可以在文件 2 中编写在其他程序中有条件调用子程序的程序。条件可以被编写用来模拟顺序流程图（SFC）编程，以便让每个子程序文件在下一个子程序被调用前必须完成控制函数。

SLC 500 有一个仅当 PLC 切换到运行模式后的第一个扫描循环中自动变为真的优先扫描位（S：1/15）。程序员可以使用优先扫描位来调用子程序，以保证在 PLC 启动时一些控制功能（如今天生产的零件数）每次转换时复位一次。

下面简要地列出了 SLC 500 的中断特性（在第 11 章中将详细地介绍）：

- 立即输入和立即输出指令可以直接读和写 I/O 模块而不用等待输入扫描或输出扫描。
- SLC 500 可以被配置用来以精确的时间间隔中断扫描循环来运行 STI 程序文件。
- SLC 500 可以被配置用来中断扫描循环来执行离散输入中断（DII）程序文件以响应输入模块中的输入状态的改变。SLC 500 也提供另外称为 I/O 中断的变量。

S5 8.4.3 Siemens STEP 5 的结构化编程配置

默认情况下，STEP 5 的操作系统重复地扫描组织块 1（OB1）。这个配置是不能改变的。然而，程序员可以编写 OB001 来包含到其他块的跳转指令（OB、PB 和 FB），尤其是包含那些每个扫描循环中被执行块列表中的其他块。通过使跳转成为有条件的，程序员可以定义禁止一些块执行的条件。

S5 PLC 仅在整个程序执行后扫描输出和输入，且配置不可以改变。如果程序员希望在块的执行期间执行 I/O 扫描，就必须创造性地编写块执行条件。例如，OB001 每个循环可以锁定一个标志位，标志打开时仅执行 PB001，标志位关闭时仅执行 PB002。在每次执行 PB001 和 PB002 之间，有一个完整的扫描循环和一个 I/O 扫描。

STEP 5 提供可选的控制系统流程图（CSF）编程语言，它可以用来编写顺序块。CSF 允许创建流程图式的程序，在程序中步程序重复每个扫描直到“转变”状态变为真，CSF 允许 PLC 停止重复过去的步程序和开始执行下一个步程序（它也有一个转变条件）。如果没有购买到 CSF 编程软件，创造性的程序员可以在组织块中编写有条件的跳转，以便每个控制函数在它被禁止和接下来的控制程序开始执行之前必须完成它的工作。仅仅会丢失很小的流程图显示部分。

下面的中断选项在第 11 章中都有详细的介绍：

- 在大的 S5 PLC 中，直接访问寻址（有“P”前缀的地址）导致 PLC 读或写 I/O 模块，所以 PLC 不必等待输入或输出扫描来更新数据。
- OB 20 在 S5 PLC 通电后每次都被执行，OB 21 每次在 PLC 从编程模式切换到运行模式时运行。如果 OB 20 和 OB 21 存在，它们可以在正常的 OB 001 重复扫描开始前用来初始化 PLC。
- OB 13（在一些 S5 PLC 中也包括 OB 10 到 OB 12）按时间间隔运行。
- 当输入模块产生硬件中断信号时，OB 2（在一些 S5 PLC 中也包含 OB 3 到 OB 5）运行。

8.4.4 Siemens STEP 7 的结构化编程配置

S7

在 PLC 处于运行模式时，S7 PLC 重复地执行扫描循环，从输入模块中读取数据到输入映像存储器中，然后在复制输出映像表中的内容到输出模块之前执行组织块 1（OB1）中的程序。程序员不能改变这个循环。程序员可以编写程序以便 OB1 可以使逻辑块能在顺序的方式下运行，从而每个逻辑块被禁止直到输入条件显示先前的活动逻辑块已经工作完毕。

Siemens 提供可选的编程语言，称为 S7 Graph 和 S7 HiGraph，它有助于顺序程序结构的创建。S7 Graph 语言与称为结构化流程图（SFC）的标准 IEC 1131-3 编程语言类似。Siemens 也为结构化文本提供可选的基于 IEC 1131-3 标准的编程语言。Siemens 的结构化文本实际上是用 C 语言和/或 Visual Basic 语言编写，所以它们可以使用非 PLC 程序员所认识的结构化编程语言来编写。

用户程序可以使用系统函数 SFC 47（等待）来使 PLC 短时间的停止执行用户程序，然后继续执行。程序员输入参数指出需要等待多久，等待延长了扫描循环的时间。

S7 PLC 在扫描循环中不会扫描所有的 I/O 模块。S7 PLC 仅扫描数字化 I/O 模块地址范围（不超过字节 127；请看第 5 章）的 I/O 映像存储区。为了编写对其他 I/O 地址的读写，程序员必须使用外围输入（PI）或外围输出（PQ）地址前缀，这样迫使 PLC 读写 I/O 模块而不是存储器中的 I/O 数据映像。程序员甚至可以使用外围输入或外围输出寻址来读写数字化 I/O 模块，而跳过 I/O 映像存储器。

无论何时 S7 PLC 切换到运行模式，STEP 操作系统就立即执行组织块 100（OB100）中的程序，如果程序员创建了 OB100 的话，然后开始执行包含 OB1 的扫描循环。

当程序员为不同于 OB1 的组织块创建程序时，程序员使操作系统能够使用该组织块作为中断服务子程序！每个组织块有它自己的优先级，并且组织块不会中断正在执行的程序，除非它有更高的优先级。[⊖]OB1 有最低的优先级（1），所以 STEP 7 操作系统可以中断 OB1 去执行任何其他的组织块。OB100 有几乎最高的优先级（27），所以它一般不能被中断。第 11 章将更详细的介绍中断，但是中断 OB 的列表遵循：

OB 35　在精确控制的循环间隔条件下执行。程序员在配置 CPU 程序的过程中可以设置循环时间。OB 35 的优先级为 12。Siemens 为附加的循环中断保留了 OB 30 到 OB 38。

OB 40　被调用来尽快的执行 I/O 模块对硬件中断的请求。当为 CPU 程序输入配置数据时，程序员可以在参数说明过程中配置 I/O 模块来产生中断请求，或者当 PLC 程序运行时，程序能包含到 SFC55、56 或 57 的调用来改变 I/O 模块的配置参数。OB 40 的优先级别为 16。Siemens 为附加的硬件中断保留了 OB 40 到 OB 47。

OB 10　被调用来在特定的时间和日期下执行，不是一次就是在每个小时的同一时间（或者每星期、每月，等等）。程序员可以通过调用 SFC 28（“SET _ TINT”）来设置时间和 OB 10 的重复，并且可以使用 SFC 30（“ACT _ TINT”）来激活中断。悬挂时间和日期中断可以用 SFC 31（“QRY _ TINT”）来检查并且可以用 SFC 29（“CAN _ TINT”）来取消。OB 10 的优先级是 2。Siemens 为附加的“时间和日期”（time-of-day）中断保留了 OB 10 到 OB 17。

OB 20　在 SFC 32（“SRT _ DINT”）执行后的指定延迟时间后执行。悬挂中断可以用

⊖ 在有些 S7 PLC 中有可能改变优先级：参考你的手册。

SFC 34（“QRY _ DINT”）来检查并且用 SFC 33（“CAN _ DINT”）来取消。OB 20 的优先级是 3。Siemens 为其他的时间延迟中断保留了 OB 20 到 OB 23。

其他的组织块响应程序员平常不希望发生的事件，如当程序出错或检测到硬件故障。运行来响应错误和故障的组织块将在第 11 章中介绍。

Siemens 也提供四个系统函数（第 11 章有描述），在程序中可以调用它们来改变 PLC 响应中断的方式：

- SFC 39（“DIS _ IRT”）用来禁止一个或更多中断的优先级，这取决于程序为调用提供的参数值。
- SFC 40（“EN _ IRT”）用来使“DIS _ IRT”禁止的中断重新生效。
- SFC 41（“DIS _ AIRT”）使比正执行的中断有更高优先级的中断等待，直到该组织块结束。（较低优先级的中断一直等待。）
- SFC 42（“EN _ AIRT”）取消“DIS _ AIRT”的效果。当所有的“DIS _ AIRT”调用都被取消时，任何正在等待的高优先级中断都能执行而不用等待该组织块结束。

CQM1

8.4.5 OMRON CQM1 中结构化编程配置

只有 CQM1 的主程序能被扫描，并且每次扫描从开始执行到结束。流程图编程语言可以用 STEP（08）、SNXT（09）、SR 25402 和“步开始标志”来模拟。（请查看手册。）

CQM1 可以用最小扫描时间来配置，以便新的扫描循环不会开始直到前一个扫描循环开始后最少等待该时间长度才开始。在 PLC 的配置数据中指定最小扫描时间将迫使 PLC 在重复的时间间隔上执行它的扫描循环，使 PLC 的响应变得更可预测，但是速度会变慢。可以配置 OMRON PLC 以便用户程序总是直接写入输出模块，而不用仅在输出扫描步期间来改变输出状态。

下面简要地列出了 OMRON PLC 中断性能（在第 11 章中将有更详细的介绍）：

- 程序可以包含 STIM（69）指令来设置操作系统来中断主扫描循环并且在精确的时间间隔上执行子程序。
- 也可以配置 OMRON PLC，以便任何时候输入地址 IR 00000（或 IR 00001、IR 00002、IR 00003）的输入状态改变，操作系统能自动的中断扫描循环来扫描少数的输入模块，然后执行子程序 0（或 1、2、3）。配置数据必须被输入到合适的数据存储器字中，使输入位打开以便初始化中断，也可以指定对哪一个输入模块进行扫描，注意该程序必须包含 INT（89）指令来使使能中断性能。
- 高速计数器可以计算 IR 00004 到 IR 00006 的输入状态改变的次数（或一个光编码器输入端口），即使脉冲改变得比扫描循环通常能够检测到的速度还快。PLC 可以进行配置以中断它的扫描循环来扫描少数的输入模块，然后当计数溢出预设的范围时运行子例行程序。配置数据必须输入到数据存储器中以使高速计数器计数并且指定要读的输入模块，必须要使用 CTBL（63）指令和数据表来指定要运行的子例行程序和运行子例行程序的计数器值。

8.5 故障检修

在结构化的 PLC 程序中会出现什么错误？答案很多！每个你使用的结构化编程程序都

使 PLC 的操作方式与设计的操作方式不同。PLC 将不再简单的执行三步扫描循环，并且它将不必从上到下地运行用户程序。

8.5.1 主控继电器的故障检修

主控继电器（用任何名称）通过使受影响的梯级上的逻辑变为非进行工作。如果指令被设计用来当它的控制逻辑改变时执行一些操作，则 MCR 能导致该指令运行，即使它的逻辑没有改变，或者可以使该指令忽略一个逻辑改变，或者可以通过延迟逻辑状态的改变来延迟指令的操作。确定你已了解 MCR 的活动性对 PLC 中的其他指令的影响。

如果程序有跳转语句，你的程序有可能执行跳转跳过 MCR 的结束语句，这样 MCR 的影响区域将比你所打算的要更广。还要注意 MCR 的嵌套使用。在一些 PLC 中允许 MCR 的嵌套，但是在大多数 PLC 中第一个 MCR 的结束语句结束所有的嵌套 MCR 的影响区域。

8.5.2 跳转和循环指令的故障检修

如果向后跳转被重复足够的次数使看门狗定时器的时间溢出，向后跳转和循环可能会导致 PLC 出错。记住当用户程序在执行时，PLC 将不会执行输入扫描或输出扫描，所以当等待一个输入改变时使用向后循环没有任何意义，除非你使用了立即输入指令，并且甚至可能会使看门狗定时器出错。

向前跳转的结果会使 PLC 不计算跳过的梯级上的逻辑，并且不更新被这些指令控制的输出指令状态。结果是被这些梯级上的指令控制的输出状态保持锁定。输入逻辑状态改变时进行操作的指令（例如计数器和一次翻转）可能会丢失实际已改变的状态。操作多重扫描的指令（定时器和一些文件操作指令）执行时就好像这些扫描没有发生一样。已开始计时一个 5s 的尖峰脉冲定时器可能会计时为 1s，因为跳转跳过它使其变成锁定状态，然后当跳转停止时 3 个星期后结束计数剩下的 4s，这样导致了很多浪费。

使用跳转标号时要注意。许多 PLC 允许个别跳转指令跳到一个单标号，但是不允许几个标号有相同的编号。也有其他的 PLC（如 OMRON）对每一个跳转指令都需要一个独特的标号，除了 label 00，你可以有几个编号为 00 的标号。

8.5.3 子程序调用的故障检修

嵌套子程序调用结构的疏忽可能会导致如向后跳转同样的问题，但是通常它使 PLC 更快出现故障。例如，如果子程序 A 调用子程序 B，而子程序 B 之后又调用子程序 A，你将会遇到持续的循环并且扫描循环中的用户程序不能结束。因为 PLC 需要在堆栈存储器中保存信息以便从调用中返回，而且典型的 PLC 仅有足够的堆栈存储器来允许大约 16 个未决的调用返回，所以 PLC 通常装满堆栈存储器并且远在看门狗定时器溢出之前出错。如果设计不好的结构化程序尝试调用一个有很多嵌套的子程序，PLC 就会出错。编程软件通常包含程序调试工具，它可以查看子程序堆栈的内容并且告诉你堆栈溢出是否是问题产生的原因，甚至可以让你查看当错误产生时哪个程序正在等待子程序结束。

不跳转到子程序也可能会导致与跳转指令跳过部分程序出现的相同问题。没有执行的逻辑不能够更新它所控制的输出，所以输出保持锁定在它们最后一次逻辑执行的状态。

对缺乏经验的程序员来说，在编写第一个结构化程序时定时器可能是很大的问题。如果

程序使用相同的逻辑来控制子程序的调用和控制该程序中的定时器，当逻辑为非时定时器指令将永远不会执行，所以它就不会复位。(初级程序员可能没有认识到相同的定时器编号能够在程序中其他地方的另外定时器指令中使用，在这个地方把定时器的控制逻辑看为非，只要程序是结构化的，那在同一个扫描循环中 PLC 将两者都不执行。)

结构化程序通常包含有条件的调用子程序，所以在任何扫描循环期间任何相关子程序都可能执行。如果多于一个子程序使用相同的存储地址（甚至定时器或计数器地址），则其中的一个子程序将使其他子程序不正确的操作。使用参数传递以便子程序仅能获得原始数据的一个拷贝，允许每个子程序处理它自己的拷贝而不影响原始数据或其他子程序。当使用了参数传递，程序员就必须注意所有需要的参数都要被传递，并且是以正确的顺序传递，而且调用程序和被调用程序的数据形式必须匹配。当传递大的数据元素和一系列数据，并且使用间接地址、变址地址或作为地址指针时，程序员必须保证 PLC 不会尝试使用比可用空间更多的存储空间，否则 PLC 将会出错。(查看第 5 章中的“故障检修”。)

Siemens PLC 提供数据块，它在保持数据集合间的相互独立时是很有用的工具，但是程序员在使用数据块时必须要小心。用户程序在它能被使用之前必须要打开一个数据块，并且用户程序仅能读/写已被创建的数据元素，否则 PLC 将出错。如果子程序被用来为主程序打开数据块，就可能会出错。PLC 会记住在子程序被调用之前哪一个数据块是打开的，并且当子程序结束时，PLC 重新打开之前的数据块。在 S7 PLC 中，如果数据块在复合数据元素传递给它之后在子程序中被使用，则也有可能会出错，因为在调用期间 PLC 使用它的共享数据块指针寄存器并且共享数据块编号将丢失。

特别在 STL 语言中，使用微处理器寄存器的数学运算和比较运算可能会出错，因为子程序能够改变 PLC 的累加器或其他寄存器中（包括状态位）的内容。调用指令自身可能不会影响累加器和其他寄存器，但是在被调用程序中的指令可能会！梯形图中的元素通常将大多数的微处理器寄存器管理得很好，但是 Allen-Bradley 偏移寄存器（S：24）是一个在子程序结束时可能改变和影响程序的寄存器。

8.5.4 允许离开扫描循环的程序配置的故障检修

如果已经配置好你的 PLC 使 PLC 的操作系统允许或导致离开标准的扫描循环，这时要记住扫描循环不再执行。即使你的 PLC 不会由于太多的嵌套中断级而出现故障，但它也可能会因为在扫描循环结束之前太多的扫描循环使看门狗定时器的时间溢出而出错。编程软件应该能鉴别出该类型的问题并且可以指出在出现故障时正等待继续执行的程序。一些 PLC 维持保存观察到的最长扫描时间的状态字。程序员可以检查该状态字，查看如何关闭出错的 PLC，即使 PLC 没有出错。

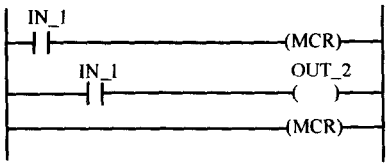
一些 PLC（如 Siemens 的）自动地尝试执行中断子程序，所以确信你没有编写程序到操作系统要查找的组织块中，除非你希望发生中断。其他的 PLC（如 OMRON 的）自动的寻找特殊的子程序执行，但是仅在明确的被指明中断为可用时。如果你认为有一天你可能需要使用 OMRON 的中断，请不要使用编号为 00 到 03 的子程序。

习题

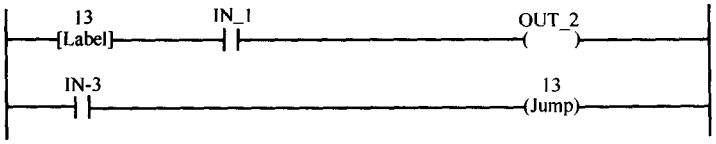
1. PLC 通常被描述为确定性控制器，是因为它们的控制间隔是可预测的和可重复的。为何

没有用结构化编程的 PLC 的主控制程序比用结构化编程方式编写的更具有确定性？

2. 当 MCR 的条件为非时，在主控继电器的初始声明与结束声明之间的非保持型延时输出会出现什么情况？当条件为真时呢？
3. 当 MCR 的条件为非时，在主控继电器的声明之间的保持型延时输出会出现什么样的情况？当条件为真时呢？
4. 下面的程序有什么错误？



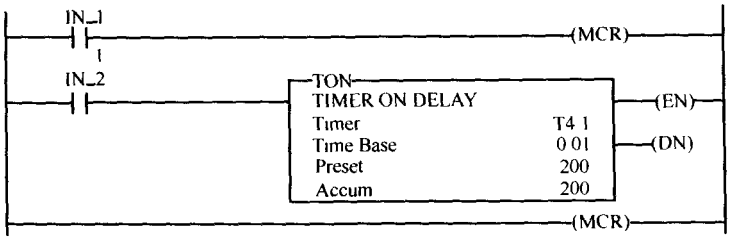
5. 当控制跳转指令的逻辑语句变为真时（在 OMRON PLC 中为非），在跳转和目的标号 LBL 间的梯级会受到怎样的影响？
6. 下面的程序有什么错误？将会发生什么结果，为什么？



7. 为何参数传递的使用可以使你的子程序/函数重复使用？
8. 当在子程序或函数中遇到返回指令时，PLC 将如何操作？
9. 定时中断如何影响标准的扫描循环？

Allen-Bradley PLC

10. 对于下面的程序：
    1）当 IN _ 1 断开而 IN _ 2 保持接通时，将会出现什么情况？在每个梯级下的空白处写下答案。（提示：定时器由 3 个数据字组成。）
    2）当 IN _ 1 变回为接通状态时，定时器将会出现什么情况？（注意：请解释清楚。）

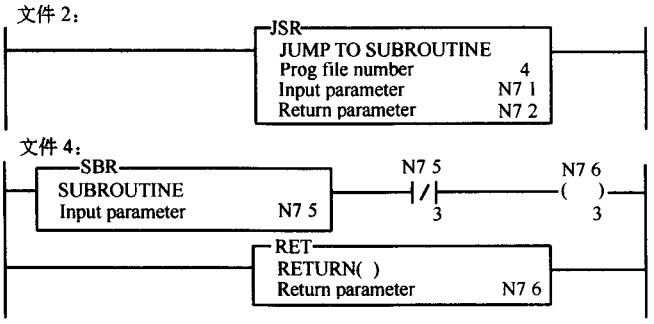


11. 在下面的程序执行后 N7：1和 N7：2中将包含什么内容？

程序执行前：N7：1 - 0000 0000 0000 0000

N7：2 - 0000 0000 0000 0000

程序执行后：N7：1 - ____ ____ ____ ____



N7：2 -____ ____ ____ ____

12. Allen-Bradley PLC-5 在每个扫描循环中执行_____作为主控制程序。（除非改变了 PLC 的配置。）
13. 如果你改变了 PLC-5 的配置使文件 3 而不是文件 2 是 MCP（主控制程序），请问 PLC-5 的扫描循环将如何改变？
14. 下面的程序中有什么错误？（“x”是一个时间，它没有错误。）

```
OB1
   A   I4 0
   JC  "Function_ 1"
"Function_ 1"
   A   I4 0
   L   X
   SP  T1
   A   T1
   =   Q7 0
   BE
```

15. Siemens PLC 在每个扫描循环中执行_____作为主控制程序。S _____ PLC 中的循环定时中断应该在_____。

Siemens S5 PLC

16. 在执行下面的程序之后，Siemens S5 PLC 中的 FY25 中将是什么内容？

程序执行之前：FW0 包含 KH0008

C0 包含 KH0008

```
      L   FW0
      L   C0
      = = F
      JC + past
      L   KHFF
past T  FY25
```

FY25 中包含_____

17. 在编写 PLC 程序时 STEP 5 中有五种不同的块，分别是哪五种？它们之间有什么不同？
18. STEP 5 中在传递参数到功能块时，参数的形式操作数、参数类型和数据类型必须要在功能块中指定（声明）。如果形式操作数的名称为“mine”的参数被声明为：

DECL：mine Q BY

(1) 主程序和功能块之间交换的数据单元是什么类型的?

(2) 主程序是否提供数据给功能块或从功能块中接收数据?

Siemens S7 PLC

19. 在执行下面的程序后，Siemens S7 PLC 的 MB25 中将是什么内容?

程序执行之前：MWO 中包含 W#16#0005

C0 中包含 W#16#0008

```
      L   MW0
      L   C0
      < > 0
      JC   past
      L    B#16#FF
past:T  MB25
```

MB25 中包含_____

20. 在执行下面的程序后，Siemens S7 PLC 的 MB25 中将是什么内容?

程序执行之前：MWO 中包含 W#16#0005

C0 中包含 W#16#0008

```
      L   MW0
      L   C0
      = =  I
      JC  past
      L   B#16#FF
past:T  MB25
```

MB25 中包含_____

21. 下面的声明类型中哪些在 Siemens 的功能块可行而在函数中是不可行的? 请解释为什么?

IN

OUT

IN_OUT

STAT

TEMP

推荐的 PLC 实验室练习

对于一个系统有：

- 四个控制面板开关：输入 0 到 3
- 四个指示灯或可视输出模块二极管：输出 A 到 D
- 两个弹簧复位阀门控制气缸：输出 E 和 F
- 一个锁销阀门控制气缸：输出 G 和 H
- 三个检测每个气缸范围的传感器

1. 用 MCR、跳转和子程序/函数来编写程序：

1) 当开关 0 接通时，打开指示灯 A，但是在开关 1 接通时就跳过这条梯级。

2) 当编写 Allen-Bradley 或 OMRON PLC 程序时：

- 在开关 0 接通后用接通延时定时器点亮指示灯 B 0.75s。现在添加一个由开关 2 控制

的主控继电器，如果开关 2 接通则定时器工作。保持开关 0 处于接通状态并用 MCR 进行试验。如果定时器需要它的控制逻辑从非变为真来复位定时器，那么为什么在每次 MCR 区域可用时定时器会重新计时？

当在西门子 S7 中编程时：

- 使用 MOVE 指令来复制四个开关位的值到指示灯。添加主控继电器，它允许 MOVE 仅当气缸 E 的传感器没有被触动。设置一些开关，然后触动气缸 E 上的传感器。

3）如果传感器 3 是触发状态程序将跳转到子程序或函数中。当开关 0 接通时子程序或函数应该点亮指示灯 C。（仅当参数传递时绝对基本的。）

2. 编写一个结构化 PLC 程序（如果编写 Siemens PLC 就使用 STL 语言）：

1）PLC 一变为运行模式时就使三个气缸全部伸展。

2）在主程序中，监控控制开关并且调用其他程序的文件/函数/子程序，这取决于开关位置。这些其他的程序可以包含跳转和调用到仍是其他的程序。

- 当开关 0 从关闭到打开时，撤销所有的气缸而不用考虑它们在什么地方和开关何时接通。
- 当开关 0 保持接通时，循环气缸。当传感器指出所有的气缸都被撤销时，再次延伸它们。当传感器检测到所有的气缸都被伸展后，再次撤销它们。（限制气流来减慢冲动。）当开关 0 断开时，气缸应该停止，而不论它们在哪个地方，不论何时开关断开。
- 当开关 1 接通时，撤销所有的气缸而不用考虑开关 0 的状态。
- 当开关 2 接通时，所有的阀螺线管将立即断开，而不考虑其他的开关是接通的。重复地闪烁输出指示灯。当开关 2 变回断开状态时，所有的灯都要关闭。

3. 使用（Siemens）功能块或（Allen-Bradley）子程序来编写程序，其中功能块或子程序接收输入参数并且产生输出参数（用 OMRON 就不可能）。如果 PLC 允许命名参数则适当地命名该参数，程序要包括：

1）如果输入参数位关闭，则功能块/子程序应在不执行任何工作的条件下结束；否则，它应该：

- 从调用程序传递到现行的值数据字中（它也将从主程序中被传递）添加一增量值，保存结果为暂时参数；
- 如果结果高于上限，值将输入作为第二个输入参数，把它设置为与上限相同的值；
- 如果结果低于在功能块/子程序中包含的下限，复制该下限值到暂时字中；
- 复制暂时字的值到输入参数。

2）包含两个到功能块/子程序调用的主程序，每个都提供独立的参数。（如果编写 Siemens PLC 则通过数据块。）

- 仅当开关 0 接通时第一个调用工作。通过参考地址来传递包含开关 1 的输入映像、正的增量值（作为常量）、现行值的参数并且传递上限值。为现行值从功能块/子程序中接收新的值。
- 仅当开关 0 断开时第二个调用工作。传递开关 2 的地址、作为常量的负增量值、被其他调用传递的相同现行值和用其他调用传递值的一半大小的上限值地址。为现行值接收一个新的值。

3）在测试你的程序时可以使用数据监控器和写数据窗口来输入上限值和观察现行值。（如果使用 Siemens PLC，你可能必须添加横线到你的主程序中，在数据监控器能够存取的数据块和存储区域之间复制数据。）

第 9 章　IEC 1131-3：通用编程语言

9.1　学习目标

本章您将了解到：

- IEC 1131 标准概述；
- 由配置、资源、任务和程序组织单元（POU）构成的普通程序结构的 IEC 1131-3 要求，并附有例子；
- 用变量声明去定义：
  - 数据单元大小；
  - 局部变量（只有单个 POU 和它的子程序可用）或者全局变量（其他 POU 可用）；
  - 临时变量，可以在本地 POU 和调用它的 POU 之间互相传递，或者作为指针保存到 POU 的共享存储空间；
- 包含自己数据的功能块和算法的例子；
- 标准函数；
- 标准编程语言（LD、IL、ST、SFC，FBD，也可能是 CFC）；
- 标准指令。

9.2　IEC 1131 概述

1979 年，国际标准组织（ISO）的兄弟组织——国际电工委员会（IEC）成立了一个部门发展一项用于可编程控制器的共同标准，IEC 1131。1997 年，IEC 标准重新编号，于是 IEC 1131 也随之重新编号为 IEC 6-1131。由于用户仍然会接触到旧编号的标准，所以在本书中，我们也会继续沿用。

IEC 1131 标准分为 6 个部分。本书主要介绍第 3 部分，IEC 1131-3，用来定义在 PLC 中使用编程语言的要求。这个标准允许组合 5 种不同的编程语言进行编程，并且有压力要求允许第 6 种编程语言。其他的部分包括：

IEC 1131-1	标准术语
IEC 1131-2	PLC 硬件及其测试要求
IEC 1131-4	选择和安装 PLC 的组件
IEC 1131-5	PLC 之间的通信
IEC 1131-6	PLC 程序中的模糊逻辑

IEC 1131 委员会包括大多数主要 PLC 生产商的代表，Rockwell（拥有 Allen-Bradley）和 OMRON 是其中的成员，Siemens 虽然不是直接的成员，但是由于它对标准化的支持，使它在全欧洲的未加入 IEC 1131 的制造商中最有影响力。为了达成众多生产商的共识，IEC 1131 标准包含很多妥协而且是非强制性的标准。即使标准没有被全部接受（而且它不是第一个遭此命运的国际标准），它仍然会为将来的发展奠定基调（就像其他失败的国际标

准一样)。

在20世纪70年代，ISO建立了一种非强制性的计算机之间通信的标准，叫做开放系统互连（OSI）标准。通用汽车以它作为一项强制标准的基础，并尝试强加给它的所有电脑部件供应商。但通用汽车公司的生产商自动化协议（MAP）标准失败了，可能是因为遵守所有协议要求的成本太高。尽管MAP失败了，但是关于计算机通信服务的OSI的七层模型仍然成为计算机通信网络的基础，其中也包括符合PLC通信要求的IEC 1131-5标准。

PLCopen是一个鼓励接受IEC 1131-3标准的组织。PLCopen的代表也包括主要的PLC生产商，包括Rockwell、OMRON和Siemens。PLCopen选择IEC 1131-3标准的一部分来实施、开发接受性测试，并且对兼容IEC 1131-3的PLC编程语言进行商业认证。PLCopen也作了妥协，它建立了三层兼容体系：基础层，可移植层和完全兼容层。在每层都有五种编程语言的兼容和测试要求，所以供应商可以寻求每种语言的认证。符合基础层要求仅仅意味着编程语言包括基本指令集并且允许程序使用IEC的结构化编程技术。可移植层的认证要求其他指令，但更重要的是，它要求编程软件可以将程序与一种中立的文件格式互相转换，这样写给一个PLC的程序就可以下载到使用不同编程语言的其他品牌的PLC中。建立标准的过程是缓慢的，PLCopen的编写已经差不多完成了基础层兼容要求，并开始在可移植层的要求下工作。不可避免地，PLCopen发现它的成员希望PLCopen提倡改变IEC 1131标准，所以PLCopen现在着手要求IEC接受在前面提到的第6种编程语言。

9.3 IEC 1131-3 编程语言

为PLC编程语言开发一个标准的最终目标是允许程序员在不知道程序在什么PLC上运行的情况下，也可以用一种所有PLC程序员都使用的语言和格式来编写PLC程序。这个标准指出只要程序包含5种（或6种）语言里面的一种或者多种，任何单独的PLC程序都应该可以工作。在讨论语言之前，我们必须检查所要求的程序结构。

IEC 1131-3标准的作者预言有朝一日PLC程序员可以购买标准程序组件，并把它们集成为有特别用途的控制程序[⊖]。为此，IEC 1131-3标准要求所有程序以一种通用的结构化方式来包含标准的程序组件。这种结构是好的，但是IEC 1131-3用来定义结构组件的术语可能是接受标准的最大困难。不要让下面章节的术语使你在认识利用这些术语所描述的有用结构时感到困惑。

学过Siemens STEP 7编程语言的读者会发现，尽管一些不同的地方会引起困难，但是STEP 7跟IEC 1131-3标准描述的编程语言非常接近。STEP 7中使用的术语和IEC 1131-3的术语很接近，使得两者之间的不同难以区分。STEP 7也提供了一些在IEC 1131-3标准中没有提到的特征（例如把指针作为参数来传递），并且满足一些IEC 1131-3中不常用的方法（例如当功能块和立即数据块一起被调用时）。记住IEC 1131-3的兼容性是非强制性的，PLCopen还没有决定IEC 1131-3的哪一部分要求超过基础层的认证，也没有任何组织阻止继续发展。STEP 7的程序员一定会原谅我在本章第一部分的例子中仅使用梯形图，因为其他读者还没有学习其他的编程语言。

⊖ 编写C语言的程序员在购买带有标准C语言程序的工具箱时，就是这样做的。

9.4 IEC 1131-3 结构化程序的通用元素

9.4.1 算法和数据类型

在写程序之前，未来的程序员甚至将拥有一些标准算法和数据结构的模板，程序员会声明（创建）其他的类型（算法和数据模板）作为他或她事业的延续。标准的*算法类型声明*包括使用标准 IEC 1131-3 指令编写的执行通用任务的例程。例如，一个标准算法可能用于驱动执行器来校正期望的系统输出和测得的系统输出之间的不同，而且算法在工作时应该像控制马达速度一样来控制房间的温度。标准的*数据结构体类型声明*包含一个标准 IEC 1131-3 数据元素安排定义好的顺序。一个数据结构体的例子可能是两个十六位二进制有符号数，紧接着是两个布尔元素（位）。这个数据结构体类型可以用于在这两个十六位二进制有符号数中包含一个来自操作员站的速度设定值，以及一个来自传感器观测到的速度值，用于我们在前面介绍的标准控制算法。这个算法可以置位或者复位这两个数据位来驱动马达正转或反转。同一个数据结构体也可以包含一个温度设定值和一个测得的实际温度值，所以它可以被标准算法用于控制这两个位来指示需要增加温度还是降低温度。

IEC 1131-3 标准定义了标准的算法和数据结构体类型声明的格式，在我们看过整体 IEC 1131-3 系统结构体后，我们将会逐一检查这些格式。

如果有标准算法和数据结构体，编程的所有工作可能是把标准的组件组成一个结构体以执行特殊制造过程要求的控制操作。写新程序将使用编程器声明（输入）一个自顶向下的结构体描述，编程器将编译最终的程序，翻译成特定型号 PLC 可以识别的二进制机器语言，并下载到 PLC（或多个 PLC）。

9.4.2 配置

自顶向下的编程，要求程序员在编程时首先声明 IEC 标准调用什么样的配置。配置声明语句描述了在配置中一起工作的 PLC“资源”系统、共同使用的数据值以及定义配置以外的电脑可访问的数据（例如，这样可以使一个车间的监控电脑系统监控生产过程，或者使一个管理员可以下命令改变生产）[⊖]。以下是 1993 年 IEC 1131-3 标准中的一个配置声明例子的全局和访问变量段。在重写它的时候，用大写字母表示标准的 IEC 术语，用小写字母表示用户定义的名字。例子中的某些部分只有继续阅读本书才能明白。

```
CONFIGURATION cell_1
   VAR_GLOBAL w: UINT;
   END_VAR
   .
   .
   .
   VAR_ACCESS
   able     :station_ 1.% IX1.1     :BOOL READ_ ONLY
   baker    :station_ 1.p1.x21      :UINT READ_ WRITE;
```

⊖ Siemens STEP 7 使用不同的术语。在 STEP 7 中配置被称为 Project，资源被称为 station，全局数据在符号表中声明。

```
    charlie   :station_ 1.z1          :BYTE;
    dog       :w                      :UINT READ_ ONLY;
    alpha     :station_ 2.p1.y1       :BYTE READ_ ONLY;
    beta      :station_ 2.p4.hout1    :INT READ ONLY;
    gamma     :station_ 2.z2          :BOOL READ_ WRITE;
    END_ VAR
END_ CONFIGURATION
```

9.4.3 资源

在配置声明中，程序员必须为每个资源输入一系列的声明语句。当IEC第一次遇到需要定义的资源时，IEC将其视为单独的计算机，可以编程来控制一组内部有联系的执行器和传感器，尽管它们允许这样解释这个定义，这样使得作为从属CPU模块运行的智能I/O模块就可以被称为主CPU的一个分离资源。从那时起，计算机的发展使CPU模块包含多处理器阵列，所以单一的CPU模块可能实际上包含了几台计算机。其他发展给我们带来了快速的多任务处理操作系统，所以单一的电脑可以同时运行几个独立的程序，这就好像有几台计算机在运行。资源的准确定义已经变得有些不确定（最少可以这样说），但是程序员仍然可以把资源看成是一台控制一个单一过程的计算机。资源声明语句标识这台计算机的微处理器的厂家和型号，列出资源必须执行的“程序”和/或“功能块”（从可用的程序模板中），并且用“任务”来建立资源的操作系统，从而定义资源执行每个程序所需要的条件。资源定义声明中的其他语句声明全局共享的数据值（仅在这个资源的程序中），并声明配置外的计算机可以访问的这个资源中的数据值。

上面提到的配置声明的例子可以向下扩展来包括含有变量声明的资源定义。增加的语句在下面以粗体字表示。

```
CONFIGURATION cell_ 1
    VAR_ GLOBAL w: UINT; END_ VAR
    RESOURCE station_ 1 ON processor_ type_ 1
        VAR_ GLOBAL z1: BYTE; END_ VAR
    .
    .
    END_ RESOURCE
    RESOURCE station_ 2 ON processor_ type_ 2
        VAR_ GLOBAL z2      :BOOL;
                    AT % QW5        :INT;
                END_ VAR
                .
                .
                END_ RESOURCE
                VAR_ ACCESS
                able        :station_ 1.% IX1.1        :BOOL READ_ ONLY;
                baker       :station_ 1.p1.x21         :UINT READ_ WRITE;
                charlie     :station_ 1.z1             :BYTE;
                dog         :w                         :UINT READ_ ONLY;
                alpha       :station_ 2.p1.y1          :BYTE READ_ ONLY;
                beta        :station_ 2.p4.hout1       :INT READ ONLY;
                gamma       :station_ 2.z2             :BOOL READ_ WRITE;
```

```
        END_ VAR
END_ CONFIGURATION
```

9.4.4 任务

资源定义包括定义任务的声明语句。任务声明语句定义计算机操作系统初始化程序或功能块的执行条件，但是并没有指定是哪一个程序或者功能块去执行。任务声明语句必须指定一个任务的名字以及任务的初始化条件是一个单一的事件（输入传感器的上升沿）还是以精确的时间间隔发生的。任务声明语句可以（选择性地）指出一个优先级。如果输入优先级，将建立一个有优先权任务的系统，高优先级的任务可以中断低优先级的程序或功能块的执行。（当插入的任务完成时，被中断的任务会继续执行。）如果没有输入优先级，将建立一个没有优先权的系统，任务将以定义语句变为真的顺序执行，而不管需要等多久才能执行，并且任务一旦开始执行将一直执行到结束。

下面是来自 IEC 1131-3 标准的例子，任务声明语句会以粗体字表示。

```
CONFIGURATION cell_ 1
  VAR_ GLOBAL w: UINT; END_ VAR
  RESOURCE station_ 1 ON processor_ type_ 1
     VAR_ GLOBAL z1: BYTE; END_ VAR
     TASK slow_ 1 (INTERVAL := t# 20ms, PRIORITY := 2);
     TASK fast_ 1 (INTERVAL := t# 10ms, PRIORITY := 1);
     .
     .
  END_ RESOURCE
  RESOURCE station_ 2 ON processor_ type_ 2
     VAR_ GLOBAL z2            :BOOL;
                 AT % QW5      :INT;
     END_ VAR
     TASK per_ 1 (INTERVAL := t# 50ms, PRIORITY := 2);
     TASK int_ 1 (SINGLE := z2, PRIORITY := 1);
     .
     .
  END_ RESOURCE
  VAR_ ACCESS
  able        :station_ 1. % IX1. 1        :BOOL READ_ ONLY;
  baker       :station_ 1. p1. x21         :UINT READ_ WRITE;
  charlie     :station_ 1. z1              :BYTE;
  dog         :w                           :UINT READ_ ONLY;
  alpha       :station_ 2. p1. y1          :BYTE READ_ ONLY;
  beta        :station_ 2. p4. hout1       :INT READ ONLY;
  gamma       :station_ 2. z2              :BOOL READ_ WRITE;
  END_ VAR
END_ CONFIGURATION
```

9.4.5 程序

资源定义必须包括定义要被资源运行的程序声明语句。每个程序声明语句必须指定要使用的标准程序模板名字，每次程序模板的使用都需要一个独一无二的名字，以及执行这个程

序初始化任务的名字。如果没有指定任务，这个程序将会作为每次 PLC 扫描循环执行的主用户程序，但是会被指派为最低优先级，这样它就会被任何有优先权的任务中断。如果有多个程序被声明去执行一个任务（包括没有任务执行的情况），程序将会以在任务中声明的顺序来执行。作为程序声明的一部分，声明语句也必须能够识别程序执行时使用的简单数据元素和数据结构体单元。这些变量声明语句在配置中要定义那些必须在程序之间传递的数据值，变量声明使程序可以使用标准的程序模板。另外一些程序定义语句可以声明全局数据值，这些数据值由组成程序的函数和功能块共享，或者声明一些这个程序的数据，使资源可以由配置获得，并使其他计算机通过这些数据访问这个配置。

我们使用的 IEC 1131-3 标准的例子可以进一步发展。在下面的程序中，程序声明会以粗体字表示。

```
CONFIGURATION cell_ 1
  VAR_ GLOBAL w: UINT; END_ VAR
  RESOURCE station_ 1 ON processor_ type_ 1
     VAR_ GLOBAL z1: BYTE; END_ VAR
     TASK slow_ 1 (INTERVAL := t#20ms, PRIORITY := 2);
     TASK fast_ 1 (INTERVAL := t#10ms, PRIORITY := 1);
     PROGRAM p1 WITH slow_ 1 :
            f(x1 := % IX1.1);
     PROGRAM p2 :
          g(out1 = > w,
          .
          .                );
  END_RESOURCE
  RESOURCE station_ 2 ON processor_ type_ 2
     VAR_GLOBAL z2          :BOOL;
              AT % QW5      :INT;
     END_ VAR
     TASK per_ 1 (INTERVAL := t#50ms, PRIORITY :=2);
     TASK int_ 1 (SINGLE :=z2, PRIORITY:=1);
     PROGRAM p1 WITH per_ 1 :
            f(x1 := z2,
            x2 := w);
     PROGRAM p4 WITH int_ 2 :
            h(hout1 = > % QW5,
                .
                .          );
  END_ RESOURCE
  VAR_ ACCESS
  able          :station_ 1.% IX1.1        :BOOL READ_ ONLY
  baker         :station_ 1.p1.x21         :UINT READ_ WRITE;
  charlie       :station_ 1.z1             :BYTE;
  dog           :w                         :UINT READ_ ONLY;
  alpha         :station_ 2.p1.y1          :BYTE READ_ ONLY;
  beta          :station_ 2.p4.hout1       :INT READ_ ONLY;
  gamma         :station_ 2.z2             :BOOL READ_ WRITE;
  END_ VAR
END_ CONFIGURATION
```

9.4.6 功能块

给 PLC 编程可以从资源层直接地执行单个的功能块，而不需要使用程序来调用功能块。这种性能看来是 IEC 1131-3 标准最后增加的，因为它要求在一个定义功能块的程序声明语句中，定义功能块的调用与程序没有关系。功能块声明语句跟程序声明语句非常相似。它识别功能块模板、功能块使用时的名字、初始化功能块的任务（调用包含功能块定义的程序任务是不需要的）以及传递到功能块的变量列表。

下面是 IEC 提供的例子，指派功能块到任务。

```
CONFIGURATION cell_ 1
  VAR_ GLOBAL w: UINT; END_ VAR
  RESOURCE station_ 1 ON processor_ type_ 1
     VAR_ GLOBAL z1: BYTE; END_ VAR
     TASK slow_ 1 (INTERVAL :=t#20ms, PRIORITY :=2);
     TASK fast_ 1 (INTERVAL :=t#10ms, PRIORITY :=1);
     PROGRAM p1 WITH slow_ 1 :
              f(x1 :=  %IX1.1);
     PROGRAM p2 :
            g(out1  =>w,
              FB1 WITH slow_ 1,
              FB2 WITH fast_ 1      );
  END_ RESOURCE
  RESOURCE station_ 2 ON processor_ type_ 2
     VAR_ GLOBAL z2          :BOOL;
               AT  %QW5      :INT;
     END_ VAR
     TASK per_ 1 (INTERVAL := t#50ms, PRIORITY :=2);
     TASK int_ 1 (SINGLE := z2, PRIORITY :=1);
     PROGRAM p1 WITH per_1 :
              f(x1 := z2,
              x2 :=w   );
     PROGRAM p4 WITH int_ 2 :
            h(hout1 = >  % QW5,
               FB1 WITH per_ 2 );
  END_ RESOURCE
  VAR_ ACCESS
  able        :station_ 1.% IX1.1        :BOOL READ_ ONLY
  baker       :station_ 1.p1.x21         :UINT READ_ WRITE;
  charlie     :station_ 1.z1             :BYTE;
  dog         : w                        :UINT READ_ ONLY;
  alpha       :station_ 2.p1.y1          :BYTE READ_ ONLY;
  beta        :station_ 2.p4.hout1       :INT READ_ ONLY;
  gamma       :station_ 2.z2             :BOOL READ_ WRITE;
  END_ VAR
END_ CONFIGURATION
```

9.5 程序组织单元

IEC 1131-3 定义了三种类型的程序组织单元（POU），包含用梯形图或者另一种允许的

编程语言的指令。程序组织单元包括程序、功能块和函数[⊖]，函数也叫做过程。IEC 1131-3标准要求如果一个 POU 正在执行并且包含对其他 POU 的调用时，这些调用是不能递推的。这意味着一个没有完成执行（它可能被中断了）的 POU 在完成之前不能被再次调用。标准也规定一个 POU 正在使用的数据不能被另一个 POU 使用，直到第一个 POU 完成了使用。这些要求强制 PLC 生产商小心地处理以下三者的相互关系——用户程序的执行（包括中断其他用户程序的用户程序）、I/O 扫描、读写数据存储区的串口通信。大体上，操作系统必须确保每个程序组织单元拥有它自己的存储器，这样它就可以在执行之前，复制所有存储区需要的数据，并且可以保持它的运行数据，其他的 POU 和交换机都不能改变这个正在运行的数据。在 POU 完成执行时，POU 的输出数据必须从 POU 的存储器中复制。

POU 的三种类型是相似的，因为它们必须包括算法使用的变量声明段，后面紧跟包含算法的段。变量声明段给每个算法需要的数据元素分配变量名字，指出名字所代表的数据类型，指出实际数据值是否必须由 POU 提供或输出，还声明变量的初始值，并且指出变量是否有特殊的特征，例如即使在 PLC 断电后，仍然有保存这个变量的能力。在我们介绍 POU 的三种类型后，我们要在下一节详细介绍变量的声明。

9.5.1 程序

程序执行可以通过在资源中定义的任务来初始化，这些资源依次是配置中的一部分。程序可以由用户编写，但是为了使程序组织单元模板的重复使用变得方便，程序必须包含比功能块和函数的条件和/或无条件调用更多的东西。一个程序不能调用另一个程序。

9.5.2 函数

函数可以从任何 POU 中调用：程序、功能块，甚至另一个函数。函数可以操作从调用它的 POU 中传递来的数据，并且总是生成一个单一的结果给调用它的 POU。因此，写一个函数有点像创建自己的指令。IEC 1131-3 标准包括多种函数的定义，这些函数可以有效地扩展 IEC 1131-3 语言的指令集。例如，图 9-1 所示的梯形图中的 COS 函数。程序员必须输入一个数字、一个地址或者是一个变量名到方框之外输入变量（IN）的旁边。COS 函数会计算这个值的余弦，并且会输出结果（可能用变量名来表示）到输出（OUT）的地址。只要连接到使能端（EN，ENABLE）的布尔逻辑为真，这个函数就会执行。标准的函数有一个使能输出（ENO），当函数成功执行完后，输出就会变为真。ENO 输出可以被随后的布尔语句使用（直至另一个函数被调用）。在图 9-1 中，COS 无条件地执行，它从存储地址 4 中取得一个 32 位的数（在 MD4 中的“D”表示一个 32 位的双字），把它当作浮点格式的实数，计算其余弦；接着把 32 位浮点结果送到地址 MD8。这个例子没有显示 ENO 输出有任何连接，所以这个程序不能检测 COS 函数是否正确工作。

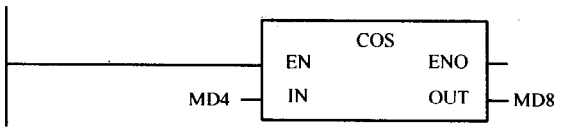


图 9-1 梯形图中的函数

⊖ Siemens STEP 7 称程序为组织块。

标准函数 IEC 1131-3 定义标准函数[⊖]来实现一些基本算法和数值运算：

ADD、MUL、SUB、DIV、MOD、EXPT、MOVE、ABS、SQRT、LV、LOG、EXP、SIN、COS、TAN、ASIN、ACOS、ATAN

选择几个输入数值中的一个：

SEL、MAX、MIN、LIMIT、MUX

逻辑运算：

AND、OR、XOR、NOT

比较数值：

GT、GE、EQ、LE、LT、NE

移位：

SHL、SHR、ROR、ROL

转换数据类型：

* _ TO _ * *

此处“*”和“* *”是 IEC 1131-3 定义的数据类型

操作字符串：

LEN、LEFT、RIGHT、MID、CONCAT、INSERT、DELETE、REPLACE、FIND

大多数 IEC 1131-3 的标准函数都可以接受任意格式的二进制数并且输出具有相同格式的结果，这种功能叫做过载[⊜]。COS 函数是一个只有部分过载能力的例子。如果图 9-1 的输入地址是 ML4，COS 函数会翻译 64 位输入（L 是指一个 64 位长数）成 64 位浮点数，并且会生成一个 64 位浮点结果。另一方面，如果输入变量指定为 MW4（W 是指 16 位字），SIN 函数会失效，因为它必须生成一个 16 位的结果，而实数不能用 16 位表示，这样，SIN 函数的过载能力就不允许它工作在 16 位的模式下。其他 IEC 1131-3 的函数可能有更多的过载性能。例如，ADD 可以接受两个或者更多的以任意二进制格式（同一种格式）编码的输入数字，并且可以产生一个同样格式的结果。更特别的是编码为 Time _ and _ Date 的数值也可以相加，因为 Time _ and _ Date 是一个 IEC 1131-3 标准的格式。甚至 MOV 函数的输入不需要是数字。MOV 可以移动任何类型的数据元素，甚至是 ASCII 字符数组或者是用户定义的数据结构体。在允许过载的地方，程序员当然必须确保目标地址的大小与输入地址的大小匹配。

用户编写的函数 程序员可以写他自己的函数。图 9-2 以构造块的形式显示了一个函数类型声明，包括一个使用标准的 IEC 1131-3 函数的梯形图算法。例子中的函数计算 16 位有符号二进制数的正弦值，这个函数接受一个输入值，这个值是真正的弧度值的 100 倍；输出一个整数值，这个整数值是实际的余弦值的 100 倍（这样结果可以包括两个十进制位）。花一点时间去阅读并理解图 9-2。在函数类型声明的第一行声明函数的名字（INT _ COS _ X _ 100）和函数输出的数据单元大小（SINT 意思是一个 16 位有符号整数），函数必须将输出结果写到与函数同名的变量中。在构造一个函数时，程序员声明函数有一个数据类型，也定义了函数结果的数据类型。函数定义也必须声

⊖ 你可以在因特网上找到完整的 IEC 61131-3 标准。在本书写作时，可以在 Allen-Bradley 的 FTP 网址 FTP//FTP. CLE. AB. COM/STS/SC65BW67TC13 下载。

⊜ IEC 61131-3 标准确切定义了每一种标准函数的过载性能。

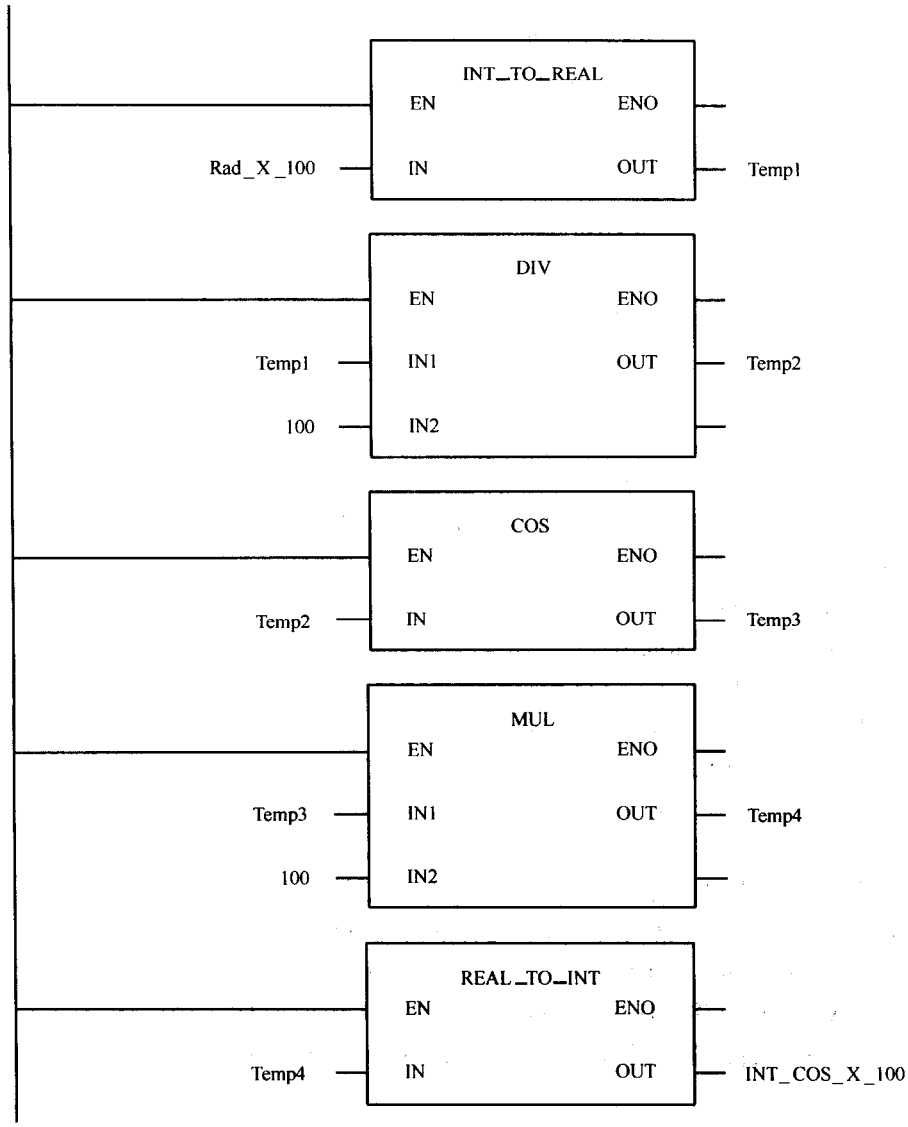


图 9-2 梯形图中用户编写的函数

明变量，这些变量为调用这个函数的程序所提供的数值（在 VAR_IN 和 END_VAR 之间），或者在程序执行时保存临时数据（在 VAR 和 END_VAR 之间）。IEC 1131-3 明确地禁止程序员编写的函数有过载的性能，只有 IEC 1131-3 程序软件包中购买的标准函数可以有过载性能。

为了让例子简单，图 9-2 中没有使用 ENO 输出，每个子函数都编程为无条件地执行（如果每个函数在前面的函数成功执行时能够有条件地执行，则会更好）。

当函数被调用时，PLC 的操作系统留出存储空间去保存在变量声明部分中声明的每个变量的数值。当函数完成执行时，指定的存储器会释放作为其他的用途，所以函数不可以保留工作数据作为将来使用。例如，不能使用一个函数在每次进行新的测量后重新计算平均

值，因为这个函数不能保存旧的测量结果以进行新的平均值计算。

9.5.3 功能块

功能块是 IEC 1131-3 标准中最重要的程序组织单元，它可以是可重复使用的算法。无论功能块在何时被调用，它都是作为功能块模板的实例来调用的。除了某些方面之外，功能块跟函数类似。功能块变量声明部分可以声明超过一个的输出参数。更加重要的差异是无论何时执行功能块实例，PLC 操作系统会留出内存区域来永久保存数据。数据成为功能块实例的一部分，并且可以被功能块实例在每次执行时重复使用[⊖]。因此，在每次测量之后，功能块实例可以用来重新计算新的平均数，因为功能块实例可以保存旧的测量结果。

因为功能块实例是一部分算法和一部分数据，所以有时它会被看做算法，有时会被看做数据。

1）功能块可以被其他 POU（程序、另一个功能块甚至是函数）调用来执行它的算法。功能块甚至可以直接通过任务来调用，如果资源声明建立了从任务到功能块连接的话。

2）在功能块可以被 POU 或者任务调用之前，功能块必须声明，这就好像功能块是数据元素一样。功能块实例可以被声明作为全局变量或在一个单一的 POU 内使用的局部变量。这个声明告诉操作系统保留数据内存空间给功能块实例使用，声明会指定使用哪一个功能块作为模板并指定一个独一无二的名字给这个功能块的实例。之前作为类型声明而输入的功能块模板，描述了每次使用功能块所需要的存储空间。为每次功能块实例保留内存叫做“实例化”，这个词可以在你下一次面试时使用。你的面试官可能会在下一次测试时让你定义实例化，以考查你是否曾经阅读过这方面的内容。

3）功能块实例可以作为参数而传递，因为它们部分是数据。PLC 操作系统为每个功能块保留专有的内存空间，所以传递功能块实例到另一个 POU，有时是使一些数据值能被共享的惟一方法。

标准功能块 IEC 1131-3 定义一些标准功能块，这些功能块执行的操作与标准的 PLC 程序指令类似。标准的功能块如下：

TON	on-delay timer
TOF	off-delay timer
TP	timed pulse
RTC	real-time clock
CTU	up counter
CTD	down counter
CTUD	up-down counter
R_ TRIG	one-shot, rising edge
F_ TRIG	one-shot, falling edge
SR	bistable latch, SET input dominant
RS	bistable latch, RESET input dominant
SEMA	semaphore; useful in creating a busy signal

图 9-3 显示了 IEC 1131-3 梯形图程序中的 CTUD 功能块（最复杂的标准功能块）。在方

⊖ Siemens STEP 7 功能块以其他方式实现同样的效果。STEP 7 功能块实例必须调用一个独一无二的实例数据块，数据保存在这个实例数据块中，而不是功能块实例中。

框上边，功能块的实例名字（my _ counter）显示在它的正常位置。其他的计数器有其他实

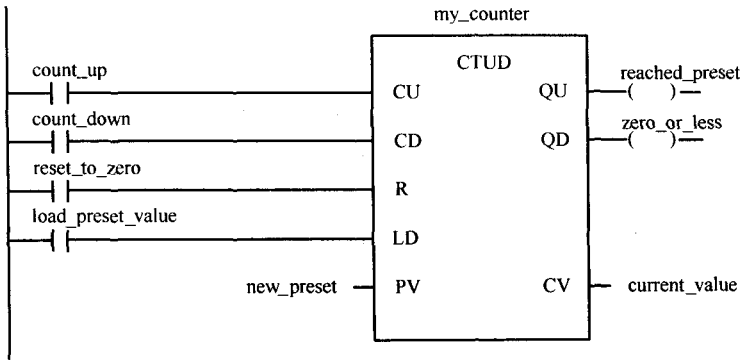


图 9-3 一个可增减计数的计数器功能块

例名字，可以用于程序的其他地方，或者同一个计数器实例也可以在其他地方调用。每个输入和输出变量已经输入了布尔逻辑语句和参数（用小写字母表示）。变量名“count _ up”代表一个布尔元素，这个元素可以在程序的其他地方操作。如果 count _ up 为真，则计数器的当前值（在内部使用作为“CV”，输出作为“current _ value”）会加 1。类似地，如果 load _ present _ value 为真，则在内部保存为“PV”的 new _ preset 的值，将会在 CTUD 这个实例中复制到“CV”。

IEC 1131-3 标准为 PID 伺服控制和统计过程控制操作提供了一些功能块的例子。（当你读到这里时，可能这些或其他功能已经有标准功能块了。）

9.6 变量和变量声明

在配置外的计算机与配置内的组件之间的变量使用和参数传递是 IEC 1131-3 标准的核心。从 PLC 构造的外部来看，配置的重要方面包括它控制的执行器和它可以接受的控制命令。从功能块实例来看，配置的最重要方面包括以下的能力——提供工作数据、保存旧的数值，并且在功能块实例结束时能够接受输出数值。配置必须使它的客户（在销售部门的人员和在结构深层的函数）的要求得到满足，这样它们都可以访问到它们需要的数据，并且只能访问它们需要的数据。在变量需要的层里，变量声明使这个需要得到满足。

9.6.1 配置层的变量声明

最高层的变量声明定义 PLC 系统到执行器、传感器和其他计算机的连接。最高层是配置层，在定义 PLC 系统的计算机文件开始时键入“CONFIGURATION”，配置层就会马上开始工作[⊖]。程序员应该在这一层声明变量，这些变量只是提供大多数或所有的资源所需要的数值。

1. 配置层的全局变量

键入 CONFIGURATION 后，变量声明的第一个部分必须加上前缀 VAR _ GLOBAL，

⊖ IEC 61131-3 标准没有规定怎样将关键字和其他需要的文件语法元素输入文件，所以编程软件允许程序员汇编文件或使用 GUI 界面以任意顺序向菜单输入数据。

并且以键入 END _ VAR 结束。在这些关键字之间，单独声明必须以以下格式输入（方括号表示可选项，斜体表示用户选择的项）：

```
var_ name [AT % dir_ rep_ addr]: data_ type [:: 5 init_ value];
```

例如，

```
motor_ 1 AT % QW42 : INT :: = + 35
```

在此处，

1）var _ name 是任意的但必须是独一无二的字母与数字混编的字符。[⊖]大写字符跟小写字符一样，第一个字符必须是一个字母，下划线（_）可以忽略，但是你也可以为了阅读方便而输入它们。IEC 1131-3 标准要求 PLC 只检查前六个字符（不计算下划线），所以程序员应该确保所有变量名字的前六个字符有不同的样式。

2）AT %dir _ rep _ addr 定义变量为直接表示的变量，意思是实际的 PLC 地址。实际的 PLC 地址必须在前面加符号%，接着是一个两字母的代码和一个数字。代码中第一个字母必须是：

I	表示输入存储区地址
Q	表示输出存储区地址
M	表示内部数据存储区地址
S	表示 PLC 状态（为将来的标准而保留）

第二个字母必须是：

X	表示位（如果没有第二个字母输入，则可以默认）
B	表示字节（8 位）
W	表示字（16 位）
D	表示双字（32 位）
L	表示长字（64 位）

数的大小只受 PLC 存储区大小的限制。它可以由几个独立的数字组成，以点来区分存储区结构。例如，%IX4.3 可以表征在存储区的输入映像区域的字节 4 的位 3。

3）data _ type 可以是任何 IEC 1131-3 能识别的数据类型，或者是用户定义的数据类型。IEC 1131-3 识别的数据类型包括：

- INT，用于有符号的 16 位整型数。
  - 前缀 S、D 或者 L（例如 SINT、DINT、LINT）分别改变大小为 8 位、32 位或者 64 位。
  - 前缀 U（例如 UINT、USINT、UDINT、ULINT）改变数据类型为无符号的二进制数。
- BOOL，用于位。

当程序组织单元（我们在后面将会介绍）的非全局声明中用作 data _ type 时，BOOL R _ EDGE 和 BOOLF _ EDGE 是允许的 BOOL 的变体，这些变体定义 var _ name 为一个位，在 dir _ rep _ addr 分别变为真或者变为假时，数值的布尔状态在程序组织单元执行的时候，这个位会设置为真。

- BYTE 用于字节（8 个位），WORD 用于 16 位的字，DWORD 用于 32 位的双字，

⊖ IEC 1131-3 指定了 ISO 646 基本代码表字符。如果你坚持使用标准字母数字字符，也不会有任何问题。

LWORD 用于 64 位的长字。

- REAL 用于 32 位浮点数，LREAL 用于 64 位浮点数。
- TIME、DATE、TIME _ OF _ DAY 和 DATE _ AND _ TIME 是专用格式的数据类型，用于保存特殊用途的数据。TIME _ OF _ DAY 或者 DATE _ AND _ TIME 可以用缩写形式 TOD 和 DT。
- STRING 用于保存 ASCII 字符的字符串。
- 包含在功能块实例中的数据结构体。输入功能块实例的名字。功能块在编程器的类型声明文件中已经声明好了。在后面会说明功能块类型声明定义，可以定义保留哪一个存储区。
- 用户定义的数据类型，在 IEC 1131-3 标准中称为派生数据类型。必须使用编程器预先在类型声明文件中定义好。派生数据类型声明文件必须以关键字 TYPE 开始，接着是一个独一无二的名字，包括一连串的变量名和数据类型，并且必须以关键字 END _ TYPE 结束。程序员要为上面声明的 data _ type 输入派生数据类型名。派生数据类型模板包含：

- 简单的变量声明（上面显示的类型中的一种）。在这种情况下，结果就好像对数据类型重新命名。例如，

```
TYPE my_ real : LREAL ;
END_ TYPE
```

- 单一的变量声明，接着是两个数字定义这个变量允许的范围。例如，

```
TYPE my_ real : LREAL(-4.3,+4.3);
END_ TYPE
```

- 一组字母数字混编的名字，通过程序组织单元可以用于输入和输出数值，而不用直接传递数值。例如，

```
TYPE mode-switch (load, run, unload) ;
END_ TYPE
```

- 一个数据结构体，由一维或多维的相似数据类型元素组成的数组组成。关键字 ARRAY 必须跟随关键字 TYPE（name），而且数组的大小和数组元素的数据类型必须指定。下面的例子定义了一个 4 * 3 的整数数组。例如，

```
TYPE my_ array
     ARRAY [1. . 4, 1. . 3] OF INT ;
END_ TYPE
```

- 数据结构体，由几个不同数据类型的元素所组成。声明必须是在关键字 STRUCT 和 END _ STRUCT 之间，而这两个关键字又是在关键字 TYPE（name）和 END _ TYPE 之间。例如，

```
TYPE my_ struct
 STRUCT
   switch : BOOL
   speed : INT
   position : my_ array
 END_ STRUCT
END_ TYPE
```

注意一些导出数据类型要求其组成成分的数据类型声明，可以使用导出数据类型作为在另一个导出数据类型声明中的数据类型。例如，你可以创建一个由一个结构体组成的数据类型，这个结构体包含一个数组（就如在上面提到的结构体例子）。在导出数据类型中声明的变量名字允许分等级的寻址，这些寻址是属于导出数据类型结构的各个部分。例如，如果一个全局变量定义为：

```
my_ mess : my_ struct
```

这样“my _ mess”的单个部分可以寻址为如下：

```
my_ mess. switch                对单个的位
my_ mess. position              对整个的 4×3 的数组
my_ mess. position [2. 2]       对数组中的一个整数
```

4）init _ value 可以是一个放进存储区的初始值，这个存储区是在配置初始化时分配给这个变量的，随后的程序执行可以改变这个值。初始值必须以 IEC 1131-3 标准定义的直接量格式输入。如果没有输入初始值，PLC 的操作系统会把所有的值设为零，日期值除外，日期值将会设定为 0001 年 1 月 1 日。

2. 配置层的访问路径声明

紧接着配置层的全局变量声明，它以关键字 END _ VAR 结束，IEC 1131-3 标准要求程序员为资源输入声明。在最后的资源声明和最后的关键字 END _ RESOURCE 之后，程序员可以列出变量，这个变量允许配置外面的计算机访问。声明列表以关键字 ACCESS _ VARS 开始，以关键字 END _ VARS 结束，之后是关键字 END _ CONFIGURATION 结束整个配置文件。访问路径声明以下面的格式输入：

```
global_ name : local_ name : data_ type [parameter]
```

其中，

1）global _ name 是远程计算机用来读和写变量的名字，与全局变量的限制相同。

2）local _ name 可以是：

- 在配置层定义的全局变量名；
- 分等级的变量名，可以定义（在必要时候）资源、程序和功能块、输入和输出名，或者是在该层声明的全局变量；
- 直接表示的变量地址。

3）data _ type 与全局变量的相同。

4）可以加入一个可选参数，限制来自配置外的对这个变量的访问。参数的关键字包括 READ _ WRITE 或者 READ _ ONLY，这个就不需多作解释了。READ _ ONLY 是默认值，所以不需要输入。

9.6.2 资源层的变量声明

只有全局变量可以在资源层声明，它使用与前面描述相同的 GLOBAL _ VAR/声明内容/END _ VAR 结构。这一层的变量声明紧接在 RESOURCE..ON.. 之后，RESOURCE... ON... 分配一个名字到资源，并定义 PLC 的处理器类型。在这一层声明的变量名和它们代表的存储空间是配置中其他资源不能访问的。在这里使用变量声明来分配变量

名到 I/O 地址，这些地址由资源里面的程序组织单元所控制，也可以分配存储区给资源的程序组织单元必须共享的值。

如果配置的 ACCESS _ VAR 段使用一个分等级的结构来定义 local _ var 的名字，该结构由资源名、一个点和声明的变量名组成，那么在这一层所声明的全局变量配置外面的计算机也可获取。

9.6.3 程序层的变量声明

程序员在程序层可以声明几种变量类型。程序层是可以包含算法来操作数据的第一层。有些声明会定义在上一层还没定义的变量，但是其他声明要求 PLC 保留一些存储空间复制已经在更高层声明过的变量。维护同一个数据元素的多个拷贝是有充分理由的。每个程序可以操作自己的共享数据拷贝，并且可以在任意长的时间内操作，而不用担心其他程序改变这个数据（甚至是中断这个程序执行更高优先级的程序）。如果每个程序必须使用共享存储空间的数据，程序就可以相互改变工作数据。

程序层的变量声明不会出现在包含配置、资源和任务声明的文件中。如图 9-4 所描述的那样，程序层变量声明会出现在每个程序组织单元的模板中，而这些单元都是包含在编程器的程序类型声明文件中。程序类型声明文件必须以一个包含关键字 PROGRAM 的声明语句开始，然后是程序模板名、变量声明，接着是程序的算法，文件必须以关键字 END _ PROGRAM 结束。可以包括几组变量声明，每一组都必须以关键字 END _ VAR 结束（图 9-4 中的程序包括了可以在程序中声明的所有变量类型）。算法必须用 IEC 1131-3 允许的编程语言来编写。在算法中，所有数据必须使用在变量声明段中声明的变量名。

图 9-4 中的例子显示了刚好够的算法演示怎样使用程序层变量名。记住程序不应该做太多的数据操作，它应该调用功能块和函数来操作数据。就像例子中显示的那样，当程序调用功能块或函数时，程序必须把参数传递给较低层的程序组织单元，其中变量由每个函数的 IN 端提供。

如图 9-5 所示，配置文件的资源定义部分必须调用程序实例，程序实例必须在资源声明这个程序语句中指定一个独一无二的名字。对于每个有独一无二名字的程序实例的调用，PLC 会分配内存空间给每个在程序数据类型模板中声明的变量。图 9-5 中，“my _ prog”（图 9-4 中）用作一个程序类型模板，并且由叫做“run _ it”的程序实例所使用。task _ 2 初始化时，run _ it 就会执行；程序实例被调用时，资源必须把参数传递给程序实例；结束时，必须接受来自程序实例的输出参数。程序类型模板的变量声明定义了参数传递的要求。例如，在图 9-5 中，参数 my _ int 是为程序的输入变量“five”而传递的。

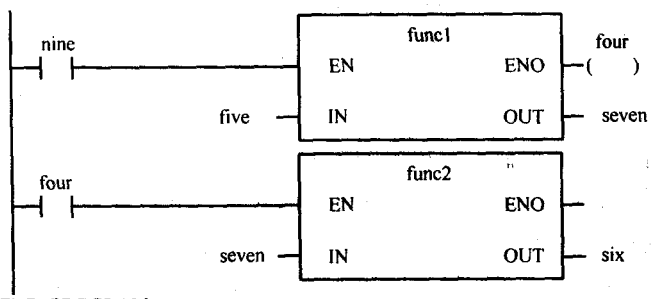
图 9-4 的例子显示了可以在程序中声明的变量类型（每一种类型都有一个例子）。单个的变量声明与我们所介绍的配置层的全局变量声明没有什么不同，但是它确实有不同之处，因为声明是在变量段里面。在段里声明的变量以下面的关键字开头：

1）VAR 在程序实例执行的时候，在 PLC 的内存区有保留的空间。当被任意变量声明初始化时，除非变量声明定义一个初始值，否则存储区内容会设定为零（通常地）。调用程序不会为 VAR 变量传递参数。

2）VAR _ TEMP，在 1997 年增加的，与 VAR 的意思相同。

3）VAR _ INPUT 保留内存区域，也要求资源为“five”传递一个有值的参数。例子中

```
PROGRAM my-prog

VAR
        one:INT;
END_VAR

VAR RETAIN
        two : REAL;
END_VAR

VAR CONSTANT
        three : INT : = 100;
END_VAR

VAR_TEMP
        four : BOOL : = FALSE;
END_VAR

VAR_INPUT
        five : INT := -10;
END_VAR

VAR_OUTPUT
        six : INT;
END_VAR

VAR_IN_OUT
        seven : REAL;
END_VAR

VAR_EXTERNAL
        eight : INT;
END_VAR

VAR_GLOBAL
        nine AT%IX20.2 : BOOL;
END_VAR
```




```
END_PROGRAM
```

图 9-4 程序类型声明

的声明包括初始值（－10），只有资源不传递参数时，这个值才会被使用。注意图 9-5 的资源例子传递其中一个先前声明的变量（my _ int）给实例程序，以此来复制到保留给“five”的存储区域。

4) VAR _ OUTPUT 为一个值保留内存空间，程序应该分配一个值给这个空间。当程序结束时，必须给资源传递一个值。注意在这个例子中，资源提供一个直接代表的变量地址

```
PROGRAM run_ it WITH task_ 2
     my_ prog ( five: = my int,
                  six=>% QW4,
                seven:=my_real)
END_ PROGRAM
```

图 9-5 创建程序实例的资源声明

(%QW4) 给程序实例写入。

5) VAR _ IN _ OUT 突破了一些 IEC 1131-3 的程序组织单元的内存使用规则。程序实例保存一个指针到一个地址，这个地址是调用程序作为参数 (my _ real) 来传递的。程序实例执行时，程序可以读和写那个地址保存的值。在这个例子中，如果程序 run _ it 包含写一个值到 "seven" 的语句，PLC 实际上会把值写到 my _ real。VAR _ IN _ OUT 变量在功能块中比在程序中有更多的值。

6) VAR _ EXTERNAL，像 VAR _ IN _ OUT 一样，让程序实例可以访问在它自己存储区之外的存储区。在这种情况下，变量 "eight" 必须是一个在资源层或配置层声明为全局的变量名。

7) VAR _ GLOBAL 创建全局参数，这些参数可以由在程序控制下执行的功能块和函数所共享。在这个例子中，一个直接引用的变量地址 (I×20.2) 在程序类型模板中被定义为名为 "nine" 位的数据源。程序层的声明是最低一层的声明，在这一层可以使用直接引用变量地址。

除了 VAR _ GLOBAL 之外，任何变量声明段都可以包括关键字 RETAIN 或者 CONSTANT 。下面的两个例子显示了在 VAR 声明段中添加这些关键字的作用。

1) VAR RETAIN 为变量在 EEPROM 存储器中保留空间，这样在程序实例执行时，如果突然断电它们的值也不会消失 (一些 PLC 把所有的数据都保存在 EEPROM 或者在有备用电池的 RAM 中，所以这个关键字没有作用)。功能块实例不能声明为已经保留了的变量。

2) VAR CONSTANT 是标准的 VAR 变量，但是声明为 CONSTANT 的变量不能被算法改变，所以声明要提供初始值 (在这个例子中，"three" 的初始值是 100)。CONSTANT 应该仅仅用于 VAR 或者 VAR _ TEMP 变量。

9.6.4 功能块层的变量声明

功能块模板是通过在编程器中创建类型声明文件来定义的。功能块类型声明文件以关键字 FUNCTION _ BLOCK 开始，然后是功能块模板名，并以关键字 END _ FUNCTION _ BLOCK 结束。在这些关键字之间，必须声明变量，就好像它们在程序类型声明文件中声明一样，但是有如下不同之处。在变量声明后面，功能块类型声明文件必须包含一个算法，应该编写功能块算法，这样它们就可以重复使用。

功能块实例化后，PLC 操作系统保留存储空间给功能块类型声明段中声明的变量。PLC 在复位之前不会释放存储空间，存储区的数据在功能块实例执行期间被保存。因为数据的持续性，变量声明有一点不同：

1) VAR 变量在功能块实例算法执行过程中保存它们的值。在功能块类型声明中定义的

初始值，在功能块实例第一次执行后便不再使用。

2）VAR _ TEMP 变量，是仅有的在功能块实例执行期间不保存的值。

3）VAR _ INPUT 值，在功能块实例被调用并且如果 POU 传递新的参数时，VAR _ INPUT 值仅接受新的值。

4）VAR _ OUTPUT 值，只有在再次执行功能块实例时才能被改变。

5）VAR _ IN _ OUT 和 VAR _ EXTERNAL 值在功能块实例没有被激活时，可以被其他的 POU 改变，因为这个两个关键字都与功能块实例存储区外的值有关。

6）VAR _ GLOBAL 变量，不能在这一层声明。

直接表示的变量地址不能在功能块的任何地方使用。如果希望功能块实例操作直接表示的变量，必须在程序层或更高层声明一个直接表示的变量，并且把这个变量传递到功能块实例作为一个输入、输出、In _ Out 或者外部变量。

9.6.5 函数层的变量声明

函数模板作为函数类型声明文件写入，就好像创建功能块模板一样，但也有一些区别。类型声明文件必须以关键字 FUNCTION 开始，当然，也必须包括函数名。第一行必须为函数声明一个数据类型。函数的一个数据类型声明会自动地声明函数的仅有的输出参数，这个参数与函数有相同的名字。其他的变量声明可以在第一行后面的变量声明段中，然后算法必须在最后一个变量声明后面。变量声明和允许的变量组与前面介绍的程序类型声明中的一样，VAR _ OUTPUT，VAR _ IN _ OUT 或者 VAR _ EXTERNAL 声明除外。惟一可能的输出变量也已经被声明了，函数应该写一个值给函数命名的变量。无论何时 POU 调用一个函数，函数都会返回这个值。VAR _ GLOBAL 变量不可以在这一层声明。

直接表示的变量不能在函数中使用。你必须在程序层或更高层声明一个直接表示的变量，并且将那个变量作为输入或者函数类型传递给函数。

9.7 IEC 1131-3 的编程语言

IEC 1131-3 定义了五种许可的语言，但是却存在压力要求接受第六种语言。所有这六种语言都是现在在控制系统中使用的编程语言。这六种语言包括三种基于文本的语言：

1）梯形图（LD），以从继电逻辑标准发展出来的梯形图语言为基础。梯形图在北美是最常用的 PLC 编程语言。

在梯形图程序中看到的图形都代表相当简单的文本指令。一些程序软件实际上允许使用文本键盘输入梯形图程序。

2）指令表（IL），建立在汇编程序语言（所有微处理器均具有）基础之上。在欧洲指令表语言普遍用于 PLC 编程。

3）结构文本（ST），跟当前个人计算机中最受欢迎的 C 语言很相似，但是更加接近 Pascal。

两种基于图形语言现在都包含在 IEC 1131-3 中：

4）顺序功能图（SFC），建立在 Telemechanique 的 GRAFCET 语言基础之上。（Telemechanique 现在是 Groupe Schneider 的一部分）。Rockwell 为 Allen _ Bradley PLC-5 的编程提供一种类似的语言。SFC 编程通过图形的方法来对子程序按顺序调用。

5）功能块图表（FBD），与带有梯形图的电子图表有一点相似——好像放入编程元素一样。

第六种是基于图表的语言，如果这种语言的支持者成功的话，这种语言也会包括在 IEC 1131-3 中：

6）连续功能图（CFC），跟 FBD 在某些方面有点相似，但是它使用的符号和连接都是建立在过程控制和运动控制工业所使用的图表基础上。IEC1499（一个独立的标准）正在准备使 CFC 语言标准化。

在本书中，我们不打算教你这六种语言中的任何一种，我们只是在细节上描述这些语言，让你认识每种语言的优点。知道在什么地方使用每种语言是很重要的，因为 IEC 1131-3 标准规定一个单一的配置可以由使用五种（到目前为止）语言里面的一种或多种语言编写的程序组织单元组成。

现在有人声称能够提供五种 IEC 1131-3 语言的编程软件，并且很多新的编程软件包声称能够提供至少一种 IEC 1131-3 语言。当你阅读这些声明时，记住 IEC 1131-3 标准的兼容性是非强制性的，这些软件并没有经过 IEC 的证明。PLCopen 计划将 IEC 1131-3 标准分为一个基础层、一个可移植层和一个完全兼容层，但是仅仅开始基础层的认证工作。PLCopen 要求制造商表明他们没有达到 IEC 1131-3 的哪一部分，如果他们声明是部分兼容的话。

一个值得考虑的问题是，IEC 1131-3 标准没有清晰地说明，用户定义的数据元素和算法模板的类型声明、变量名和程序注释的存储区域。当输入一个 PLC 配置的时候，在编程器中必须可以获得它们，当然，标准并没有说明它们必须被下载到 PLC 的存储区域。编程器可以去除所有的注释和变量名，甚至当编程器把 IEC 1131-3 兼容的配置编译成机器语言时，编程器可以改变程序的结构。除非标准达成一致，否则使用另一个编程器来控制系统去监控或调整是不可能实现的。

9.7.1 梯形图

IEC 1131-3 中的梯形图跟其他的梯形图编程有些类似。IEC 1131-3 将梯级称为网络，并且网络可以有分支。除了标准的布尔指令检查开（EXAMINE ON）和检查关（EXAMINE OFF），检测位的正跳变（POSITIVE transition）和负跳变（NEGATIVE transition）的元素也是标准化的。

—| |—	—| / |—	—| P |—	—| N |—
EXAMINE ON	EXAMINE OFF	POSITIVE transition	NEGATIVE transition

标准的布尔输出指令有几种，其中四种应该不需要描述：

—( )—	—(/)—	—(S)—	—(R)—
COIL	NEGATED COIL	SET COIL	RESET COIL

也有一些新的布尔输出元素，当布尔逻辑控制语句改变状态时，其中两种只提供一次翻转的真值：

—(P)—	—(N)—
POSITIVE transition	NEGATIVE transition

其中三种用于保存布尔输出到以关键字 RETAIN 声明的布尔变量，这样即使断电后，它们的值也会被 PLC 保存：

—(M)—	—(SM)—	—(RM)—
RETENTIVE	SET	RESET
MEMORY	RETENTIVE	RETENTIVE
COIL	COIL	COIL

其他常见梯形图指令，例如移动（MOVE）和定时器（TIMER），通过调用标准函数或者标准功能块来实现。

在 IEC 1131-3 内定义的标准函数都包括 EN 输入端，这样它们可以根据布尔逻辑语句来执行；并且它们都包括 ENO 输出，这样函数的成功完成就可以控制其他的布尔元素。用户编写的函数或者功能块可以在梯形图中使用，无论它们是用哪种语言编写的，只要它们包括至少一种布尔输入使得它们可以接入梯形图网络。一个包括标准功能块调用的梯形图网络如图 9-6 所示，图中“value”在每次“switch”从假值变成真值时都会增加 2，但是最大不能超过 100。一旦程序员习惯了输入参数作为图形方框外面的变量名（例如“value”和“temp”）或者作为直接量（例如+2 和+100）时，IEC 1131-3 梯形图中的修改程序是很容易的。

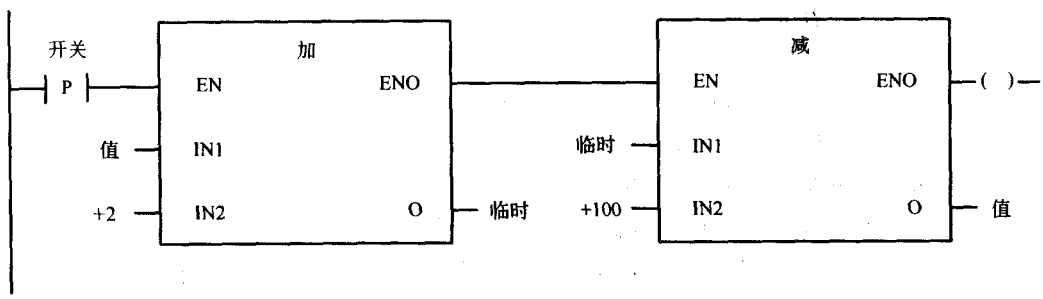


图 9-6 带有标准函数的梯形图网络

图 9-7 所示是一个标准的开延时定时器功能块，它可能会用于梯形图网络。功能块类型声明名为“TON”。功能块实例由程序员赋予名字“my _ ton”。输入定时器的预设时间（PT）（在这个例子中是 10s），并且由变量“light”表示的布尔位会在布尔位“switch”置位后的 10 秒置位。程序中的其他网络可以读出已用的时间（ET，elapsed time），这个时间由功能块实例输出到变量“so _ far”。

尽管 IEC 1131-3 在梯形图中包含一个标准的 JUMP 指令，但它建议 jump 不应该用于梯形图的组织单元的内部编程。

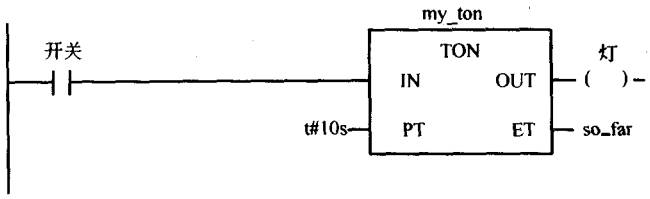


图 9-7 梯形图网络中的开延时计时器功能块

9.7.2 指令表

指令表编程语言有点像 Siemens 的语句表（STL）编程语言，所以除了指令有一点不同之外，其他部分我们都已经在前面讲过。布尔逻辑指令包括：

LD、AND（or &）、OR、XOR、ST、S、R）

在此处 LD（加载）可以加载一位值来初始化一个布尔逻辑语句，并且 ST（保存）可以把布尔操作的结果写到输出地址。

除了 S、R 和 ）之外，任何一个布尔算子都有一个修正器，使它们检查或控制的位的状态翻转。例如，如果正在加载的位的值是假，LDN 会加载 TRUE。

建立布尔逻辑结果的算子（例如 AND，OR，XOR）也可以包含一个“（”修正器，以便在已经开始的语句中建立另一个布尔语句。例如：

```
LD      switch1
XOR     ( switch2
AND     switch3
)
ST      light
```

如果“switch1”是开，或者如果“switch2”和“switch3”同时开，“light”就会置位，但是如果三个开关都开着的话，“light”就不会置位。

非布尔数据值操作指令包括

LD、ST、ADD、SUB、MUL、DIV

LD（加载）和 ST（保存）可以用于布尔逻辑，或者也可用于把非布尔值复制入和复制出微处理器的累加器。跟随 LD 或 ST 指令的地址决定是否一个布尔或者非布尔操作被请求。

算子可以操作任何标准的 IEC 数据类型的数据值，并返回一个具有相同数据类型的结果。除了 LD 和 ST 之外，所有算子都可以包含“(”修正器，在未决的操作之前，强制执行紧接着的数据处理操作。例如，

```
LD      value1
MUL (   value2
ADD     value3
)
ST      result
```

在加法运算结果与“value1”相乘之前，强制 PLC 把“value2”加到“value3”上。最后的结果保存为“result”。

数据比较指令包括：

GT、GE、EQ、NE、LE、LT

这些指令可以接受任何标准数据类型，并会返回一个布尔逻辑结果。所有这些算子都包含一个“（”修正器。

结构化编程指令包括：

JMP、CAL、RET

其中任意一个指令都包含一个“C”修正器，这样使它根据前面的布尔逻辑语句进行有条件的操作，与/或一个“N”修正器来翻转它的执行条件。例如，

```
LD        switch
JMPCN     there
```

在“switch”不为真的条件下，会导致一个到标签“there”的跳变。

CAL 指令必须跟随一对括号，里面包含一系列参数（如果有的话），传递给被调用的函数或者功能块。例如，如图 9-7 所示的梯形图格式的标准开延时计时器功能块（TON），输入名为 IN 和 PT 的输入变量和名为 OUT 和 ET 的输出变量。图 9-7 与以下的指令表语言相似：

```
CAL my_ ton (IN: switch, PT: t#10s, OUT: light, ET: so_ far)
```

没有取反的无条件调用不要求使用 CAL 指令。上面的指令也可以写成：

```
my_ ton (IN: switch, PT: t#10s, OUT: light, ET: so_ far)
```

IEC 1131-3 包含特别设计的指令，用于把新值写入功能块实例的输入变量。输入变量名作为指令来使用，并且实例功能块名也包括在指令内。例如，开延时计数器功能块（TON）有名为 IN 和 PV 的输入变量，这些输入变量名可以作为指令表指令来使用，就像下面的例子一样：

```
LD          switch
IN          my_ ton
LD          t#10s
PT          my_ ton
My_ ton     (OUT: light, ET: so_ far)
```

功能块调用中的变量传递可以通过以下方法完全消除——使用以“实例名．变量名”方式寻址的标准指令表指令来读和写功能块的变量。就像下面的例子一样：

```
LD          switch
ST          my_ ton. IN
LD          t#10s
ST          my_ ton. PT
my_ ton
LD          my_ ton. OUT
ST          light
LD          my_ ton. ET
ST          so_ far
```

如果 IEC 1131-3 兼容的程序是真正可移植的，它必须使写给任何 PLC 的程序能够装载到任何类型的编程器，这些编程器能够运行任何供应商的编程软件，然后再装载到 PLC 中。[⊖]为了实现这个过程，当程序文件从一个供应商产品转移到另一个供应商的产品时，程序文件会请求一个中立语言格式。定义数据的中立文件格式定义已经达成一致，但是算法的中立格式仍然在讨论之中。欧洲制造商提议将指令表作为算法的中立语言格式。指令表语言接近于所有微处理器都使用的机器语言，并且所有其他的 IEC 1131-3 语言都可以用指令表表达（虽然确切的翻译可能会引起争议）。北美的供应商对使用指令表的提议感到不满意。

9.7.3 结构文本

很多非 PLC 程序员都用 C 语言或者 Pascal 语言来编程。如果要让不懂指令表或者梯形

⊖ 由于各个 PLC 使用的微处理器和存储器仍然不同，所以还是需要为每一种 PLC 写各自专用的下载软件。

图的程序员接受 PLC 作为标准的控制计算机，那么这些 PLC 必须以一种结构化的文本语言来编程，例如 C 或 Pascal 语言。

声明变量、分配变量名和数据类型以及传递变量的概念，对多数 PLC 程序员来说都是新的，但是对于 C 语言和 Pascal 语言的程序员来说又是很熟悉的。C 和 Pascal 都允许函数或过程的定义，这些函数或过程可以从另一个程序中调用，所以可重用算法的概念也是熟悉的。声明一个数据结构体作为一个数据类型，然后传递那个数据类型的数据元素，是面向对象编程（OOP）和即插即用的核心部分。

结构文本所基于的语言（Pascal）包括在布尔类型数据元素中有较好表现的逻辑操作指令，并且提供在 PLC 中使用的其他数据类型和数据操作指令。如果你了解 Pascal 或者 C 语言，你就可以学会使用为结构化文本定义的 IEC 1131-3 指令集。

结构文本提供一些在 Pascal 和 C 的结构化编程中具有的高级编程特点。这些特点包括：

1）IF..THEN..［ELSE..END_ELSE］..END_IF 结构，它们可以使用一个条件语句从一到两个选项中选择一个操作来执行。

2）CASE..OF..［ELSE］..END_CASE 结构，可以用于选择一个单一的操作，从而执行一连串可能的动作。

3）FOR..DO 结构，WHILE..DO 结构，REPEAT..UNTIL 结构，可以用于通过一段算法来使程序重复地循环，直至条件语句变成真时才停止（或者直至这个条件变成假时）。

9.7.4 顺序功能图

PLC 从出现起就用于控制生产过程的顺序。我们在第 3 章介绍了如何写顺序器程序。通过建立图表显示顺序中的每个步骤，使一个步骤完成和下一个步骤开始的跳变事件，以及显示可选的步骤顺序，顺序功能图（SFC）语言为程序员提供一种简单的方法来写非常复杂的顺序器程序。在 SFC 中，每一个步骤之后必须要有一个跳变，而且每个跳变只能使一个步骤工作。

如果比较简单的话，步骤的动作和跳变条件可以用 SFC 图表编程，并且可以在任何一种 IEC 1131-3 语言中编程。如果步骤的动作算法或者跳变条件算法比较复杂，它们就可以作为 SFC 调用的函数或者功能块写入。当然，跳变算法必须返回一个布尔结果，来使跳变生效或失效。步骤的算法可能会更加复杂，并且有一些所有程序和功能块都服从的规则限制，其中包括关于变量声明和参数传递的规则。一个 SFC 甚至可以调用由 SFC 语言写的功能块。

1. SFC 步骤动作

在 SFC 算法中，SFC 的图表步骤必须有步骤名，而且必须指明当轮到这个步骤时，应该执行哪一个动作。动作实际上是 SFC 程序调用的函数或功能块实例，条件语句的跳变将控制调用。SFC 图表包括在步骤旁边的函数或功能块实例名。一个步骤可以有零个或者更多的动作（如果没有动作，在 SFC 执行时，PLC 并不完成任何工作，直至下一个跳变变成真）。每个动作都可以用一个可选择的动作限定词来编程，这样可以详细说明在这个步骤激活后，步骤动作应该从何时开始继续执行并且继续执行多久。选择性的指示变量也可以输入到 SFC 程序。指示变量是动作的函数或功能块的输出变量，并用于监视那个步骤的行为。

图 9-8 显示了一个跳变——步骤——跳变的顺序，并且有一个梯形图中的跳变控制程序。当“tank_full”变为真，之前的步骤会变得无效，步骤“Warm”开始执行。动作的限定器是“N”，意思是动作必须在 SFC 执行时执行，直至下一个跳变变为真时才停止。完成

的动作包含在名为“Heat _ 23”的功能块“实例”中。Heat _ 23 在这里没有显示，但是它所要求的参数可以告诉我们它的作用。它要求一个温度设定值、一个要读取的传感器地址、一个要打开的加热器地址和一个表示“完成”的布尔输出。当变量“Temp _ 23”的值低于35 度时，Heat _ 23 会让加热器（Elem _ 23）打开。当温度到达 35 度时，Heat _ 23 会把 Elem _ 23 关掉，并且把 temp23 _ OK 打开。把 temp23 _ OK 打开能够启用下一个跳变并且让这个步骤失效。

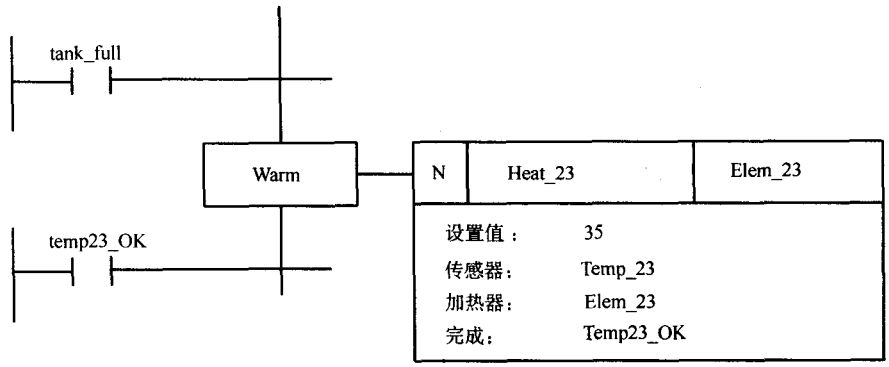


图 9-8 跳变—步骤—跳变的顺序

动作限定词“N”用于刚才的例子。允许的动作限定词包括：

N 步骤激活时，在 SFC 程序的每一次扫描使动作执行（没有输入动作限定器时也有同样的效果）。

P 步骤激活时，使相关的动作执行一次。

L time “time”指明动作执行的最大时间，除非动作被激活后很快又停止。

SL time “time”指明在每次步骤被激活时动作要执行的时间，即使这个步骤在时间用完前停止。

D time “time”指明动作开始执行之前的一个延时。只有在步骤被激活时，动作才会继续执行。

S 开始执行相关动作并且一直执行它，即使这个步骤结束。

SD time “time”指明在动作开始执行之前的一个延时。即使这个步骤停止，执行也会开始并继续。

DS time “time”指明在动作开始执行之前的一个延时。当时间用完后，除非步骤仍然处于激活状态，否则执行不会开始，但是一旦开始，它会在步骤停止后一直执行。

R 停止一个以 S、SD 或者 DS 的动作限定词开始的动作。

SFC 程序可以包括一些检查程序中步骤状态的指令。对于每个步骤，PLC 维持一个布尔步骤激活标志，设定地址为＜step _ name＞. X，一个已用时间，设定地址为＜step _ name＞. T。这个时间保存在 IEC 1131-3 的 TIME 格式中。使用步骤的已用时间来触发下一个跳变是在固定时间内保持步骤激活的一个方法。例如，

- 当图 9-8 中的步骤激活时，“Warm. X”为真值。
- “Warm. T”有一个等于 Warm 最后一次激活持续的时间值。

2. SFC 执行顺序

以下我们描述 SFC 程序是怎样进行构造来控制它们如何执行的。

从现在开始，下面的例子只显示步骤块和跳变标记，而不显示它们必须包括的动作识别块或者跳变条件。

必须有一个显然不同于其他步骤的初始步骤，因为它有双竖线，如图 9-9 所示。



图 9-9 初始步骤

初始步骤是 PLC 启动时执行的第一个步骤[⊖]。每次 SFC 算法执行时，PLC 会再次执行与这个步骤相关的动作（基于动作限定词），直至下一个跳变为真时[⊜]。在一个步骤之后的跳变变为真时，这个步骤会停止，并且下一个步骤会被激活。

SFC 图表可以分流为一种或多种可供选择的路径。这里有两种不同的分流编程方法。在一个多选一分流（exclusive divergence）中，SFC 算法会从几种可能的垂直路径中选择一种，而其他路径的步骤将不会被执行。SFC 流程图必须显示下一个步骤可能有的多个路径，每个路径由一个跳变开始。如图 9-10 所示。

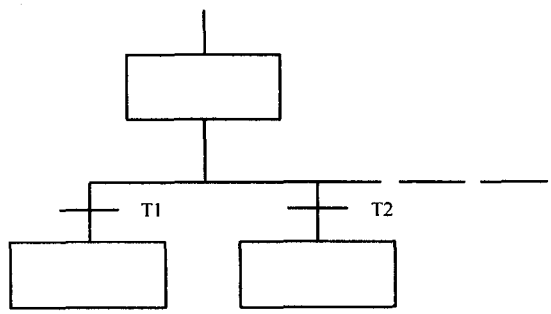


图 9-10 多选一分流

采用上面所示的方法编程，程序员必须写入跳变条件（T1 和 T2 等），这样任何时候都只有一个跳变为真；否则，就不可预测 PLC 选择的实际路径。多选一分流将会首先通过编程来强制 PLC 评估最左边的跳变条件，然后检查下一个最左边的跳变，直至发现一个真的条件为止。为了强制执行这个过程，在分流点输入一个星号（*），如图 9-11 所示。

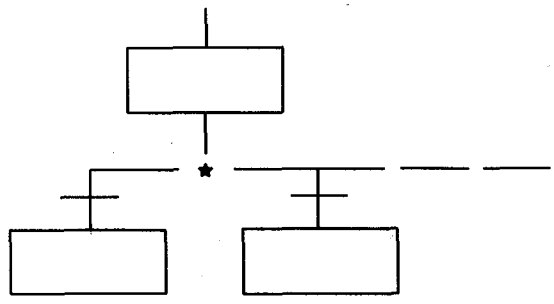


图 9-11 强制由左到右的跳变评估的多选一分流

通过输入星号和输入指明顺序的数字（1，之后 2，之后 3，等）来评估跳变条件，程序

⊖ 有些 PLC 制造商允许 PLC 配置成可以从上一次断电时顺序停止的地方重新开始。

⊜ 你可能在其他地方读到一个步骤可以在它的跳变为真之后再执行一次。对于 1997 年修改的标准，这是错误的。

员就可以强制另一个评估跳变的顺序，如图 9-12 所示。

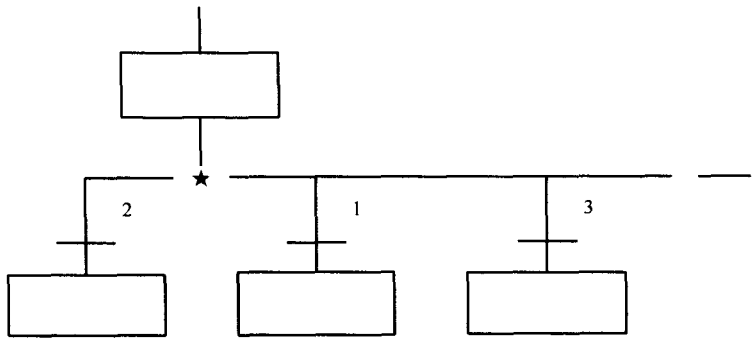


图 9-12 强制程序员选择跳变评估顺序的多选一分流

在每条平行的路径跳变之后，分支的路径应该汇集于一点，如图 9-13 所示。注意跳变要求出现在汇集点之前。

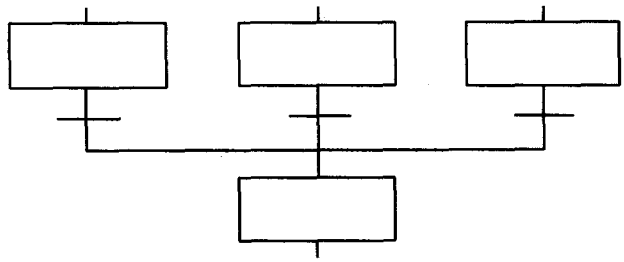


图 9-13 多选一分流路径的汇集

多选一的分流路径不需要包括任何步骤，所以它可以用作一种跳过或者返回去重复步骤的方法，如图 9-14 所示。

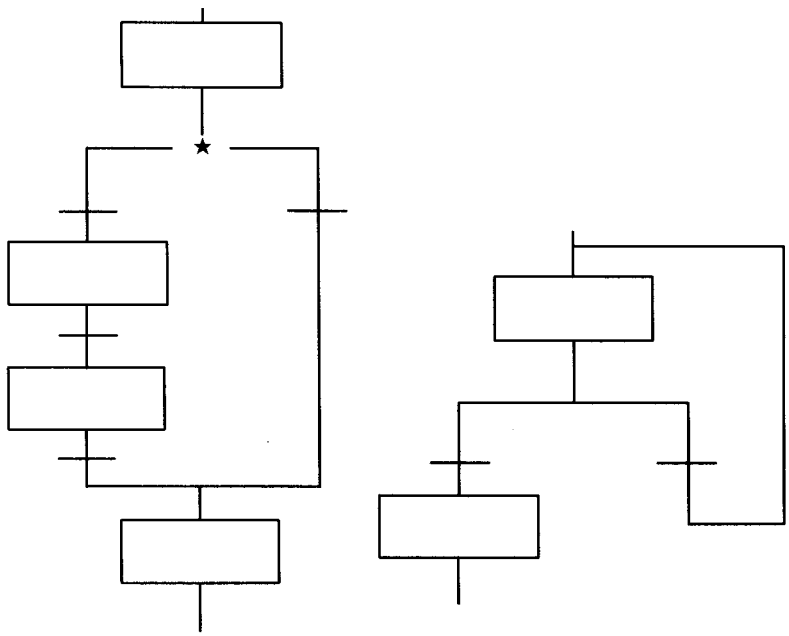
对分流编程的另一种方法是使用同步的顺序分流。跳变必须在分流前编程。双线指明了分流和汇集点。SFC 的同步顺序分流和汇集点如图 9-15 所示。在分流前面的跳变变为真后，PLC 开始执行所有分流路径的步骤（事实上，计算机一次只能完成一件事，所以它实际上首先执行最左边路径的步骤，然后再执行右边的另一条路径的步骤，等等）。每条路径都包含几个步骤和跳变（甚至包含其他的分流和汇集点），但是路径在它们最后的步骤中不能包含独立编程的跳变。作为替代，单一的跳变会在汇集点下面编程。仅当在所有分流路径的最后一个步骤被激活时，这个跳变才会被评估。

9.7.5 功能块图表

对功能块图表（FBD）编程与对函数和功能块的梯形图调用编程很相似，不同的是不需要梯形图的布尔逻辑图。对于布尔逻辑，必须使用有布尔输出参数的功能块元素，例如 AND 和 OR。功能块编程时也可以包含布尔逻辑语句，这些语句能够影响它们的内部操作。

每个 FBD 元素（表示一个函数或者功能块）描述为一个方框，其中输入变量显示在左边，输出变量显示在右边，并且参数的形式名显示在方框的相应的端点。函数或者功能块名显示在方框内部，功能块实例名显示在方框上方。

FBD 算法和梯形图功能块调用的不同之处在于怎样显示变量在 FBD 网络元素之间的传



a) 跳过所选步骤　　b) 重复所选步骤

图 9-14　多选一的分流路径

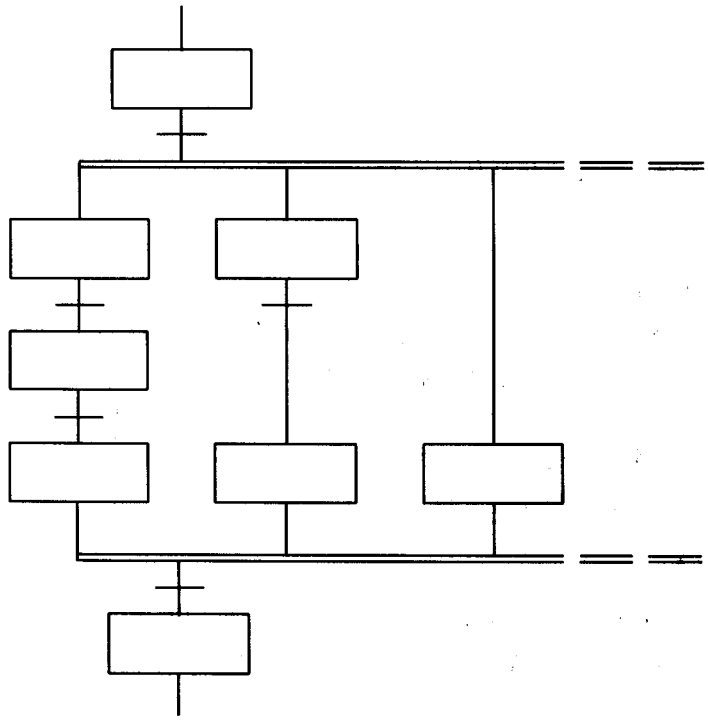


图 9-15　同步的顺序分流和汇集点

递。在 FBD 中，一个元素的输出参数端点可以连接到另一个元素的输入参数端点，甚至可以通过参数传递连接线将参数传递到生成这些参数的元素的左边或上方的元素，如图 9-16 所示。my _ speed 的转矩输出会反馈到 my _ position，这样如果一个大的请求转矩没有使位置误差减

少，my _ position 就可以检测到一个错误。虽然 FBD 的编程好像是没有结束的循环，在下一次 FBD 图表执行时，PLC 会使用 my _ speed 的转矩输出作为 my-position 的 Requested _ Torque 的输入。IEC 1131-3 规定程序员应该有能力选择 FBD 的功能块的执行顺序（“want”，“posn”，“max”，“drive”和“stop _ motor”是这个 FBD 与调用它的程序交换数据的参数）。

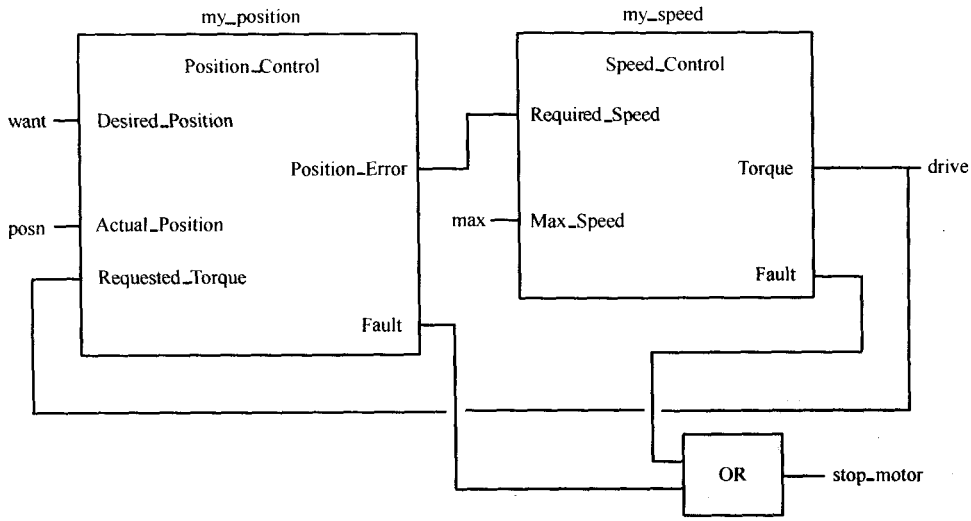


图 9-16 带有反馈的功能块图表网络

9.7.6 连续功能图

连续功能图（CFC）语言目前还不是 IEC 1131-3 认可的语言，如果它变为认可的语言，会分配一个不同的名字给它。当前过程控制和运动控制软件的用户都想使用 CFC 语言，因为这种语言与他们现在使用的一些语言相似。CFC 语言包括过程控制和运动控制的标准算法。

CFC 算法与 FBD 算法相似的地方是它们都是由这样的元素组成，从一个元素的输出变量可以连接到另一个元素的输入变量。图 9-17 显示图 9-16 的 FBD 算法在 CFC 中的用法（CFC 的输入和输出参数被除去了）。

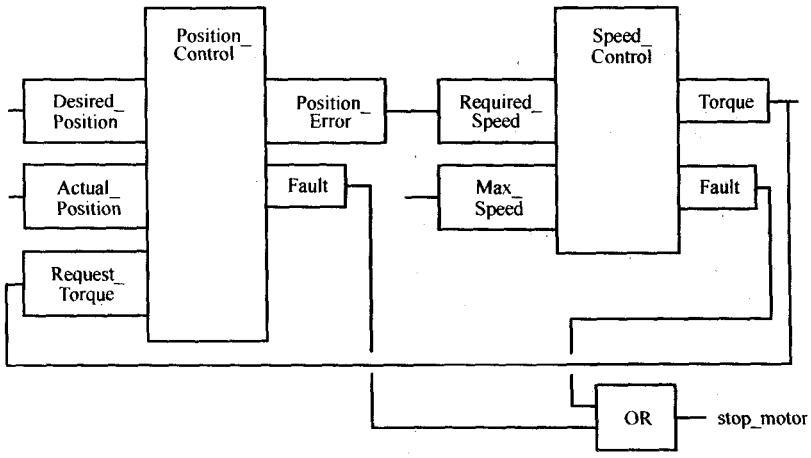


图 9-17 带有反馈的连续功能图网络

FBD 和 CFC 有明显的不同，因为形式参数名标识的附件显示在 CFC 元素方框外面，而不是在方框里面。变量的外部特征反映了 CFC 语言和现有的 IEC 1131-3 语言的一个很大差别。CFC 程序忽视了函数和功能块之间的不同。程序员不需要声明一个功能块实例来获得可以使用长期存储空间来保存参数的程序元素。图像元素的外部标识表示 CFC 程序变量，只要控制程序继续操作（或更长）就可以保留，这些变量并不是函数的一部分，但是函数会使用它们。

建立一套可以自动地为每一个 CFC 对标准功能块的调用创建一个新的实例的 CFC 编程软件是可能的，这样 CFC 会遵守 IEC 1131-3 的结构要求，但是如果实例对程序员来说是不可见的，那么传递参数值的能力就会变得相当有限。

9.8 总结

虽然 IEC 1131 已经被重编号为 IEC 61131，但是大众仍然通俗地称它为 IEC 1131。第 3 部分（IEC 1131-3）定义了一系列标准，这些标准允许任意 PLC 程序下载到任意 PLC 中。PLCopen——一个与 IEC 没有官方联系的组织，正在要求开发可以用于检查程序软件是否与 IEC 1131-3 标准相符的认证。

IEC 1131-3 描述了五种可以用于 PLC 编程的语言。编程软件并不需要提供全部五种语言。五种语言都是允许的，因为它们都是 PLC 在不同历史时期的应用。梯形图（LD）语言是由北美的继电逻辑图发展而来的；指令表（IL）与汇编语言相似，并且在欧洲非常流行；结构文本（ST）与 Pascal 和 C 语言很相似，所以它允许“主流”的程序员（那些没有使用过 PLC 的程序员）使用一种和他们现在使用的语言非常相似的语言；顺序功能图（SFC）是一个相当近期的图表语言，用于编写顺序的控制程序；功能块图表（FBD）是电子工程师和过程控制系统设计员常用的一种图表语言。存在压力要求将连续功能图 CFC 增加为第六种标准语言。CFC 在过程控制和运动控制工业中都很常用。

所有的五种语言都必须能够使用 IEC 1131-3 强加给程序的结构，使它们可以在 PLC 之间移植。除了简单地使指令系统标准化之外，IEC 1131-3 要求有变量声明和参数传递，使程序有权更改可以用于其他程序的数据。

IEC 1131-3 描述了程序组织单元（POU）。每个 POU 有一个算法段和一个变量声明段。变量声明定义用于这个算法的值的形式参数名，并给每个变量指定数据类型，并且一个 POU 被另一个 POU 调用时，能指明是否要求传递这个变量的一个参数。变量声明和参数传递允许每个 POU 对它所需要的数据有独占的权利，并允许在 POU 之间传递数据项目的拷贝。数据元素可以在更高层声明，并允许它们被几个 POU 在全局使用。

程序组织单元有三种类型。函数只生成一个输出参数，并且当它有相同的输入参数值时，它们的输出参数值也总是相同的。功能块需要永久存储它的一些数据，因为它可以通过编程来使用之前执行的数据。为了指明功能块使用哪一组数据，需要为每个功能块和它专有的数据集指定一个实例名，并且功能块由实例名调用。程序应该包含功能块和函数的调用，所以程序应该安排函数和功能块调用的顺序和条件。

功能块和函数可以互相调用，但是程序不能被任何 POU 调用。程序员可以为一个任务声明一个程序，并且必须声明任务调用程序的条件。任务可以响应一个条件来执行调用的程序，或者在重复的时间间隔中执行，或者如果声明没有初始任务，程序会在每一个扫描循环

执行。任务可以分配优先级，这样会停止低优先级的任务并允许高优先级的任务执行。一个单独的 PLC 称为资源，并且可以配置来执行多重任务。资源是一个较大型控制系统的典型组成部分，并且资源之间可以进行通信。整个控制系统叫做配置。图 9-18 是 IEC DIS 1131-3 的图 19b 的拷贝，显示了控制系统的封装。

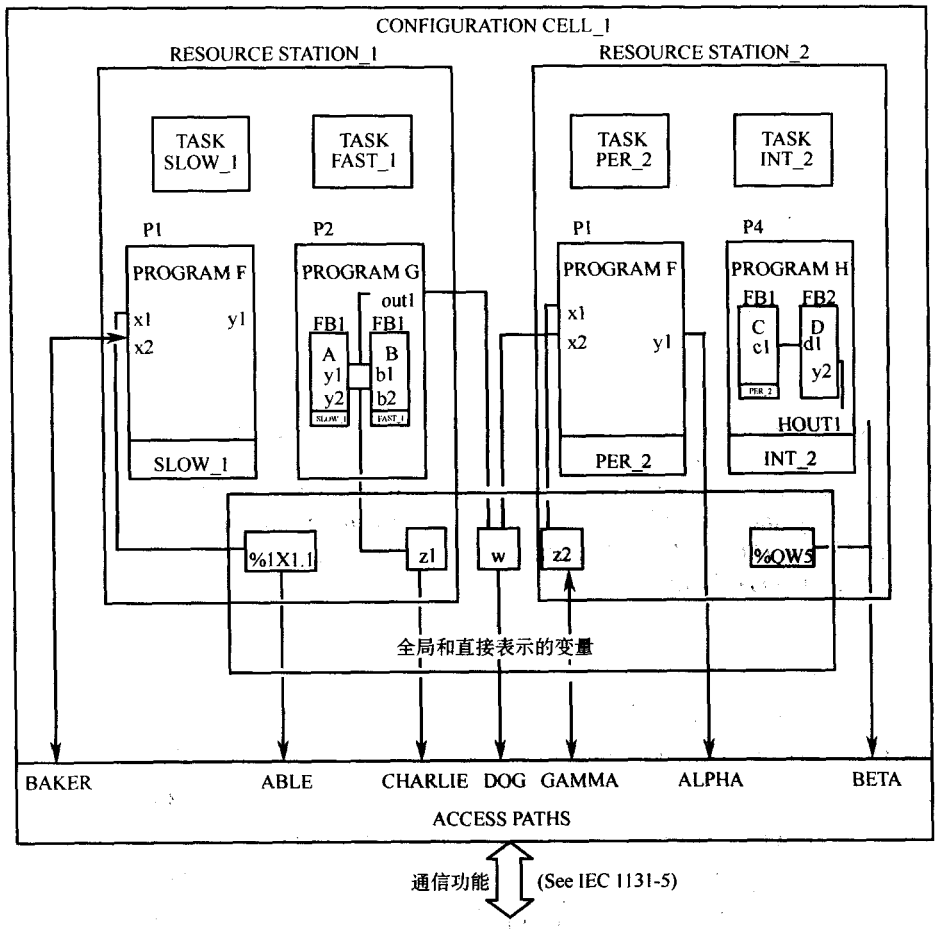


图 9-18 IEC 1131-3 软件模型

9.9 故障检修

IEC 1131-3 没有定义任何工具来确定为什么一个程序不工作。事实上，自从函数和程序在完成执行时释放它们的存储空间后，程序员不能总是检查存储器来发现什么时候和什么地方出现了不正确的结果。小心使用时，IEC 1131-3 要求的程序结构可以帮助避免编程的问题，因为新的程序应该完整地调用经过测试的函数和功能块。IEC 1131-3 也要求编程器应该检测数据类型的不匹配（例如，传递实数到需要整数的函数），所以应该不再有这些类型的问题。

直至标准被完全理解，并且直至 PLCopen（或者其他组织）建立兼容性的认证，用户可以预见的问题也就是 IEC 1131-3 正在尝试解决的问题：不兼容性。你可能发现当你尝试把用指令表编写的程序（完全遵守 IEC 1131-3 标准）下载到另一个 PLC 时（这也遵守标准），它却不能下载，因为第二个 PLC 不能把你的指令表程序翻译为它所认识的语言；当你

尝试把程序从一个 PLC 上传到一个 IEC 1131-3 的编程器时，你会发现其他问题，即在 PLC 存储器的汇编程序不能被编译，因为程序没有从原来的编程器上下载足够的信息。IEC 1131-3 没有定义必须汇编并下载什么信息到 PLC 的存储器中。

PLCopen 已经存在的认证要求包括制造商如果声明他们的产品与 IEC 1131-3 兼容，他们也必须准确地标出他们没有遵守标准中的哪一部分。IEC 1131-3 标准包括很多编号的段落组成的表格。每个编号的段落定义标准的一个部分。在购买 PLC 或编程软件时，购买者应该查看兼容性的陈述，并理解这个陈述的意思。PLCopen 有一个因特网址（http：//www. plcopen. com），提供标准发展的信息。购买者如果需要一个 IEC 1131-3 标准的拷贝，可以从 IEC 或者所在国家的标准组织购买，也可以从因特网上的 Allen-Bradley 的 FTP 站点中下载（ftp//ftp. cle. ab. com/sts/iec/sc65bw67tc13）。记住 IEC 1131 在 1997 年后变成 IEC G1131，要使这个数字在各个地方都改变，还需要一段时间（营销人员最近几年都在推广“IEC 1131-3”，他们可能根本不想改变这个数字[⊖]）。

参考文献

Brendel, W., and Tiegelkamp, M., *Uniform PLC Programming in Accordance with IEC 1131-3*, PLCopen, The Netherlands.

Lewis, R. W., *Programming Industrial Control Systems Using IEC 1131-4*, Institute of Electrical Engineers, London, 1995.

IEC Subcommittee 65B/WG7, *IEC DIS 1131-3*, IEC, Geneva, 1993.

IEC Subcommittee 65B/WG7/TF3, *Committee Draft: Amendments to IEC 1131-3*, IEC, Geneva, 1997.

习题

1. 与 IEC 1131-3 兼容是强制性的吗？IEC 会测试和证明兼容性吗？
2. 配置声明语句描述 PLC 系统一起工作的_____，以及它们共享的_____。资源声明语句能识别_____，列出资源必须执行的_____和_____，建立_____，在资源里面声明_____。任务声明语句定义了一些条件，在这些条件下操作系统应该_____。程序声明语句指明给程序模板使用的_____，以及这个程序初始化执行的_____的名字。功能块声明语句能分配一个名字给功能块的_____，初始化功能块的_____，以及一个_____的列表，从而传递这个功能块实例。函数总是产生_____返回调用这个函数的程序。
3. 什么是过载的功能？
4. 函数和功能块在使用存储区方面有什么不同？在你的答案中应正确地使用实例化这个词语。
5. 什么是全局变量、访问变量、临时变量、静态变量、输入变量和输入输出变量？
6. 辨别 IEC 1131-3 语言指定的五种当前的语言，分别用一个句子描述每一种。
7. 什么是在 SFC 语言程序中使用的动作限定词？带有一个 S 动作限定词的步骤与带有一个标准 N 限定词的步骤有什么不同？
8. SFC 多选一的分流与标准（非多选一）的分流有什么不同？

⊖ 你知道大部分电脑都有的 RS-232C 端口吗？其实在 1990 年代早期它就被改称为 EIA-232D 端口了，但是计算机的营销部门并没有更改他们的习惯称谓。

第 10 章　PLC 的设置和配置

10.1　学习目标

本章您将了解到：

- 如何安装 Allen-Bradley、Siemens 或者 OMRON PLC 系统，包括远程 I/O、局域网和编程器，也包括 DIP 开关和跳线的设置；
- 针对用户选择的应用设置编程器和 PLC 系统所需要的配置；
- 怎样给 PLC 编程让 PLC 启动时进行初始化配置以及在它运行时修改配置。

在这一章中，我们不讨论如何给 PLC 连接电源及 PLC 元件间的电源连接。出于安全原因考虑，必须按照 PLC 提供的安装说明来装配。

10.2　安装和配置新的 PLC

当你打开装有你刚买回来新的 PLC 系统盒子时，你会发现：

1）使用手册（指南）。手册描述了如何对你已经买回的 PLC 安装、配置、编程，以及对 PLC 故障检修。在将 PLC 设备互连或者给 PLC 供电前，要先熟悉使用手册。在这一章，我们告诉你在一般情况下如何安装 PLC 系统。你的使用手册会准确地告诉你应该怎么做。

2）CPU 模块，有时也叫做处理器，或者 PC。

3）编程器。编程器可能由来自 PLC 制造商的软件组成，你必须安装这个软件到你的个人电脑。一些制造商出售完整的编程器，它们是已经安装软件的工业个人计算机。一些 PLC 的制造商出售不完全的编程器，这种编程器足以用来给小型的 PLC 控制器编程。在这章中，我们认为你已经有编程软件，并将安装到个人电脑上。

4）通过 PLC 的接口硬件将 CPU 模块连接到编程器。这些接口硬件可能只是串行通信电缆，或者包括 RS-232C/电流环路转换器。硬件可能包括安装在个人电脑上的通信卡，它用来高速编程和监控程序或者用来连接包括多个 CPU 模块的局域网络。

5）电源模块（除非你的 PLC 是内置电源的集成型）。

6）I/O 模块对你的应用特别重要，除非你的 CPU 模块有足够的内置 I/O 能力来控制你的小型系统。在本章我们假设你有一个模块化的 PLC 系统，包括数字、模拟和智能 I/O 模块。

7）连接 I/O 模块和 CPU 模块的总线系统。在本章中我们假设你的 PLC 系统有一个中心框架（包括 CPU 和一些 I/O 模块）和 I/O 模块的远程框架。小型的传感器和执行器组越来越多地通过传感器/执行器总线或者现场总线局域网连接到 PLC 系统中。传感器/执行器总线和现场总线将在第 15 章讨论。

10.3　安装硬件

一些组件可能有 DIP 开关或者跳线，在装配组件前需要进行设置。如果 PLC 有远程框

架，它们必须连接到主框架上。最后，编程器必须连接到 CPU 上或者连接到包括一些 CPU 的局域网络中。

PLC-5

10.3.1 ALLEN-BRADLEY PLC-5 的硬件安装

1. PLC-5 底盘设置

PLC-5 插入 Allen-Bradley 称作底盘的框架中。底盘上的跳线必须被设置用来提示电源模块是否从外部安装在底盘的尾部或者插入底盘。必须设置底盘上的 DIP 开关，开关指示：

1）在这个框架中的模块使用的寻址方式：

■ 双插槽寻址。一对相邻的插槽称为一组，它们分别代表 16 位字的输入表和 16 位字的输出表。完整的八组逻辑框架需要一个 16 插槽的底盘。

两个插槽每组可以包括：

● 两个 8 位 I/O 模块，左侧的插槽（插槽 0）代表在输入或者输出模块中的 16 位字的低 8 位，右侧的插槽（插槽 1）代表 I/O 映像字的高字节。

● 一个 16 位的输入模块和一个 16 位的输出模块。

● 一个 16 位的输入模块和一个 8 位的输出模块。

● 一个 8 位的输入模块和一个 16 位的输出模块。

● 一个单插槽块传输模块和一个 8 位输入或者输出模块[⊖]。

● 两个单插槽块传输模块。

● 一个双插槽块传输模块。

■ 单插槽寻址。每个插槽分别代表一个 16 位输入映像字和一个 16 位输出映像字。每组只有插槽 0，没有插槽 1。一个框架中只需要有八个插槽，这被认为是完整的逻辑框架。16 插槽底盘有两个逻辑框架。单一插槽组可以包含：

● 任何 8 位或者 16 位 I/O 模块或者一个块传输模块（包括一个双插槽传输模块，它们被认为是占用了左边的两个插槽而寻址。）。

● 32 位 I/O 模块！每对相邻的插槽必须包含不超过一个 32 位输入模块和一个 32 位输出模块。PLC-5 将自动地分配低的 I/O 映像地址给每个模块的低 16 位，分配高的I/O 映像地址给每个模块的高 16 位。

■ 1/2 插槽寻址。每个插槽代表两个 16 位的输入映像字和两个 16 位输出映像字。来自模块的低 16 位 I/O 模块由低地址的字代表。一个完整的逻辑框架只需要 4 个插槽，每个插槽可以包含 I/O 模块列表来给插槽寻址。如果包含 32 位输入触点和 32 位输出触点的 I/O 模块是可用的，那么在每个底盘插槽上应该安装其中的一个模块来配置 1/2 插槽寻址。

2）如果检测到硬件出错，在这个框架中的输出模块会怎样影响它们的输出。选项包括“复位所有的输出”（通常是安全选项）或者“保持最后的状态”。即使“保持最后的状态”设置被使用，在一些情况下，硬件出错事件发生的时候输出将复位。查看用户手册。

3）存储器控制选项，如果这个底盘包括一个 CPU 模块。选项包括：

⊖ CPU 和块传输模块在数据的块传输中使用 I/O 映像表中的 8 位，所以你不能在同一个组中安装 16 位 I/O 模块作为数据传输模块

■ 允许或者禁止从编程器中清除 CPU 存储器。

■ 在复位时总是传送 EEPROM 数据给存储器，或者不传送，或者只有当 CPU 存储器检测到故障（例如，如果后备电源失效）才这样做。

4）重启选项，如果这个底盘是带有远程 I/O 适配器模块的远程 I/O 框架而不是 CPU。在出错后，或者需要一个开关连接到底盘上用来复位底盘后，选项允许 CPU 模块复位框架。

2. PLC-5 CPU 模块设置

在 PLC-5 CPU 旁边的 DIP 开关用来：

1）设置 DH＋地址，如果这个 CPU 在 DH＋局域网中；

2）配置串行接口通道；

3）给 CPU 模块选择扫描模式或者适配器模式，并且如果选择的是适配器模式：

■ 框架号和第一个 I/O 组号；

■ 与扫描模式 CPU 交换的数据单元大小；

（查看第 13 章获得关于数据高速公路，串行通信和远程 I/O 扫描的详细资料）。

3. PLC-5 远程 I/O 适配器设置

远程 I/O 框架必须有一个 CPU 模块被配置作为适配器，或者在左边插槽中有包括 CPU 模块的 1771-ASB 远程 I/O 适配器（RIO）模块。在 1771-ASB 远程 I/O 适配器模块边上的 DIP 开关用来：

1）分配框架号和第一个 I/O 组号；

2）选择是否忽略（不扫描）这个底盘上的最后 4 个插槽；

3）选择串口通信速度为扫描模式 CPU 交换数据；

4）选择是否把这个底盘作为主底盘或者作为补充底盘。如果另一个包括 RIO 模块的底盘被配置作为主底盘，那这个 RIO 模块可以被配置作为有相同框架号和 I/O 组号的补充底盘，这样允许使用尚未使用的 I/O 映像空间。例如，如果主底盘有一个输入模块，它给 I：123提供数据，而不是接收来自 O：123 的数据的输出模块，补充底盘可以包括输出模块来接收数据（补充底盘也可以有输入模块来补充主底盘的输出模块。）。

4. PLC-5 块传输和智能 I/O 模块

一些 I/O 模块有 DIP 开关和跳线，在安装模块到底盘前，可以设置它们。

■ 模拟输入模块，例如，有跳线来配置模块的电压或者当前输入信号。

■ 模拟输出模块可以有跳线，可以用来使输出达到最小值、最大值或者中等范围，或者当 PLC 程序停止运行时，保存它们最后的值。

块传输模块和智能 I/O 模块可以查阅用户手册。

5. PLC-5 编程器通信接口卡

通过串口可以进行更快速的数据交换，Allen-Bradley 给个人计算机提供通信接口卡的选择（大多数在它们的识别号中有“KT”字样。）。

当“即插即用”技术正逐渐减少现代计算机对使用接口卡进行硬件设置的需要时，大多数接口卡仍然有 DIP 开关，用来：

1）为通信卡选择一个地址，使个人计算机可以通过通信卡读和写数据到 PLC 的 CPU 模块中。如果使用通信卡上的默认地址设置，不需要改变程序软件的默认配置来使用接口卡。

2）选择通信卡将产生哪种中断来中断个人计算机。如果个人计算机已经有一个产生默

认中断的接口设备，则必须选择另一个中断。两个 Allen-Bradley 通信卡可以共享相同的中断。可以选择“没有中断”，尽管这不是推荐的，因为这样用户必须对个人计算机编程来确保重复地轮询接口卡以接收来自 PLC 的数据。

6. 连接远程 I/O 框架到 PLC-5 主框架

PLC-5 扫描模式 CPU 的简单菊花链通道 1B 到远程 I/O 适配器模块，或者到 I/O 模块的远程底盘适配器模式的 CPU[⊖]。Allen-Bradley 推荐 Belden Blue 电缆，通过沿着菊花链的远程底盘触点 1 和触点 2 连接一个 82 欧姆的终端电阻（尽管这种方法很难实现，但随后你将看到配置扫描模式的 CPU 使用远程 I/O 是多么容易）。

通道 1B 是默认的扫描通道。如果需要，也可以使用另一个通道，但不得不改变默认 CPU 通信口的配置，这将在后面讲述。

7. 连接 PLC-5 CPU 到数据高速公路＋（DH＋）

简单地通过菊花链连接 PLC-5 CPU 模块通道 1A 到其他 PLC-5 CPU 的相同通道。Allen-Bradley 推荐使用 Belden Blue 电缆，安装一个 82 欧姆的终端电阻，这个电阻通过在 DH＋的菊花链连接终点上连接 1 和 2。

通道 1A 是默认的 DH＋通道。如果需要，也可以使用另一个通道，但是不得不改变默认的 CPU 通信端口配置，这将在后面讲述。

8. 连接 PLC-5 编程器

安装通信卡到个人电脑的插槽中，插编程器接口电缆到通信卡中（或者到 RS-232C 连接器中，如果没有购买通信卡），连接通信电缆的另一端到插在 DH＋连接器的最近的 CPU 模块上（DH＋连接器现在在通道 1A；如果改变通信配置来移动 DH＋接口到通道 2，必须在 DH＋改变后移动编程器连接。）。

SLC 500 10.3.2 ALLEN-BRADLEY SLC 500 硬件安装

1. SLC 500 CPU 模块设置

在 SLC 500 CPU 模块中的跳线被设置用来选择交流 120 或者 240V 输入电源，并表明安装在 CPU 模块中 EEPROM 模块的大小。

2. 连接 SLC 500 远程 I/O 框架到主框架上

包括一个 CPU 模块的 SLC 500 模块化 PLC，可以被带有 1747-DCM 模块的扫描模式 PLC 看作远程 I/O 框架。扫描模式 PLC 可以是 PLC-5，或者是 SLC 5/02 或者是更高级的带有 1747-SN 模块的 SLC 500 PLC。连接到 SLC 500 的远程 I/O 是通过菊花链将 1747-DCM 模块连接到 1747-SN 模块。

3. 连接 SLC 500 CPU 到 DH-485 或者 DH＋

SLC5/04 CPU 模块可以直接连接到带有 PLC-5 CPU 和其他 SLC 5/04 模块的 DH＋网络，通过 3 线的 DH＋使用菊花链将这些 CPU 连在一起。通道 1 是默认的 DH＋端口。

任意 SLC 500 CPU 通过连接 CPU 模块到它自己的 1747-AIC 模块，可以连接到 DH-485，然后通过 4 线的 DH-485 使用菊化链连接所有的 1747-AIC 模块。在菊花链的每一个端

⊖ 在菊化链中主模块和第一远程模块被连接在一起，另一个惟一的电缆被用来连接第一远程模块的相同触点到第二远程模块，接着连接第二模块到第三模块，依此类推。

点，连接器的终端 5 和 6 必须连接来提供终端电阻。在菊花链的一个端点（不是两个），终端 1 和 2 必须连接来使电缆接地。在 SLC 5/04 中，通道 0 是默认 DH-485 端口。

你可以通过一个端口连接 SLC 5/04 到 DH+，通过其他端口到 DH-485，以便 SLC 5/04 可以传输来自一个网络的信息到其他网络。

4. 连接 SLC 500 编程器

有几种方法来连接编程设备到 SLC 500 或者到包含 SLC 500 的网络，最常用的方法是使用 1747-PIC 接口电缆连接 SLC 500 到包含程序软件的个人计算机的串口。对高速的数据交换，可以购买 Allen-Bradley 的通信接口卡，并安装该卡到个人计算机上。如果要安装通信接口卡，阅读 PLC-5 编程器通信接口卡的小节。SLC 500 设置手册包括其他连接方法的细节。

10.3.3 SIEMENS S5 硬件安装

S5

1. S5 底盘设置

在 S5-100U 系列 PLC 中，电源、CPU 模块、总线单元和一个（可选择的）接口单元被挂在 DIN 导轨上。电源必须通过导线连接到 CPU 和总线单元上，但其他组件是通过插入来自一个模块的电缆到下一个来连接的。在插 I/O 模块到总线单元之前，总线单元中的选位开关必须与 I/O 模块背后的标识相匹配，这样你就不会不小心把错误的 I/O 模块类型插到不同 I/O 模块类型的插槽中。开关设置也告诉 CPU 在这个插槽需要读或写多少个数据。

在 S5-115U 系列 PLC 中，所有的模块可以插到 Siemens 的底板上。一个机械的编码元件可以安装到底板上，以便只有一种 I/O 模块类型可以插入到底板上，但是没有电气型的代码元件。

2. 带有中断能力的 S5 I/O 模块

一些带有中断信号生成能力的 S5-115U 数字输入模块通过把模块插到底板前移走跳线，从而禁止中断能力。

3. S5 智能 I/O 模块

智能 I/O 模块有它们自己的计算能力，本书使用的智能 I/O 模块的定义中包括 Siemens 智能 I/O（IP）模块和 Siemens 模拟 I/O 模块。通信处理模块和接口模块也是智能的，在下一节中讨论。在智能 I/O 模块和模拟 I/O 模块上的开关可用来选择如下的事情：

1）模拟信号类型和范围（有时通过插入适当的测量范围模块来选择）；

2）二进制格式，假定在二进制数和模拟数之间转换；

3）使用了多少模块的模拟输入通道（来允许更高速的转换，如果只有少量信号需要转换。）；

4）在模拟信号电路中打开/关闭电线损坏检测；

5）模拟输入模块的交流电源频率（允许那个频率的交流噪声被滤除）；

6）一些传感器类型的接口或者补偿需要；

7）定时器或者计数器模块预设值；

8）来自增强型编码器的脉冲，被计数为 1X（计算一个通道的正信号）、2X（计算一个通道的所有信号变化）或者 4X（计算两个通道的所有信号变化）。

4. 连接 S5 扩展单元框架到主框架

有两种方法连接 I/O 模块的扩展单元框架到包含 S7 CPU 模块的核心框架上：

1）在核心框架系统中。整个电缆长度必须小于或等于 2.5 米，电缆互相连接每个框架中的接口模块（IM），包括中心框架。每个 IM 模块必须装配在底板或导轨的最右边。中心框架必须有一个电源，但是扩展单元框架没有。

2）在分散框架系统中（对 S5-100U 不允许），扩展单元框架和带有 S5-115U CPU 模块的中心框架之间的距离最长可以达到 3 公里（由选择的 IM 模块决定），最多可以有 64 个框架互相连接。每个框架（包括中心框架）必须有一个 IM 模块，通常作为最右边的模块，只有中心系统 IM 模块可以装在分散系统 IM 模块的右边。

每个互连到分散系统的框架也可以是中心框架系统的一部分，如图 10-1 所示。分散系统的 IM 模块可以在中心系统的任意一个框架中。当然，分散－核心复合系统中只有一个框架需要 CPU 模块。

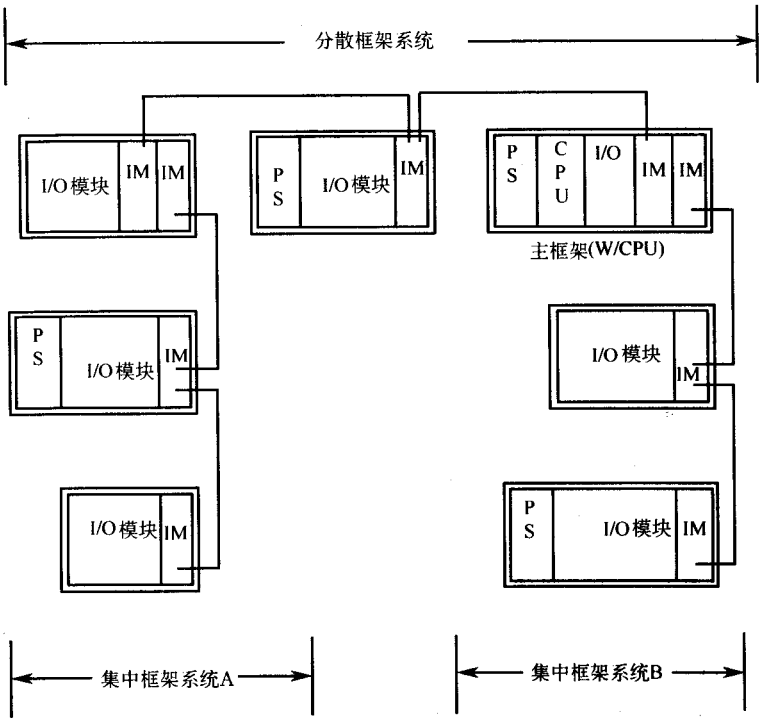


图 10-1 包含一个中心框架和数个扩展框架的分散－核心复合系统

如果只需要一个扩展单元，可以使用预先装配的由两个 IM 模块组成的组件和一条电缆；如果需要更多的扩展单元，IM 模块、电缆和终端器必须分别购买，并且 IM 模块要有 DIP 开关和可能设置为以下方式的跳线：

1）给每个扩展单元框架选择地址范围；

2）显示每个框架包含多少 I/O。一些模块需要 32 位的数字 I/O 或 16 通道的模拟 I/O，而其他小型的模块有 16 个数字位或者 8 个模拟通道就足够了；

3）使能一个或两个接口模块的通信端口（必须在一个没有使用的端口中插入终端器。）；

4）指明连接到（分散的）IM 模块上的最长的连接电缆的长度；

5）指明如何响应如果在一个（分散的）端口检测到一个“未准备好”状态；

6）为每个 IM 模块指定一个特殊的接口号（如果 IM 模块有 1K 字节的存储空间来和 CPU 交换数据，如果 CPU 也配置来给页帧通信）。

远程 I/O 框架可以通过 Profibus 局域网连接（在下一节中解释）。

5. 连接 S5 CPU 到 L1 局域网

S5 CPU 模块通过被称为编程端口的 CPU 串口连接到 L1 接口单元，接口模块将 L1 电信号转换为程序端口信号，L1 接口单元通过带屏蔽的两对双绞线组成的菊花链连接在一起，屏蔽必须在 L1 网络一端接地。在网络中需要一个 L1 网络主站，它可以在个人计算机或 S5-115U PLC 中。S5-100U PLC 仅可以作为 L1 网络上的从站。

6. 连接 S5 CPU 到 Profibus 局域网

有三种类型的 Profibus 局域网，所以连接到 Profibus 网络需要正确的接口。最早的一种就叫做 Profibus，但有时用字母“FDL”或者描述符“part1”来表示与其他两种不同。另外两种是 Profibus-DP 和 Profibus-FMS（不同的地方在第 13 章中解释。）。

S5-115U CPU 模块可以通过接口（IM 308-C）模块连接到 Profibus-FDL 或 Profibus-DP 网络。S5-115U 可以是活动节点，表明可以包含一个程序来命令通过 Profibus 网络交换数据；S5-115U 也可以作为 Profibus 控制器。CP 5431 模块可以作为一个 Profibus-DP 或 Profibus-FMS 网络上的主站，执行控制由 S5-115U CPU 中的用户程序命令的功能。

S5-100U 或 S5-115U CPU 模块通过通信处理模块（CP 541）可以连接到 Profibus-FDL 或者 Profibus-DP 网络。CP 541 通过程序端口而不是通过 PLC 的总线系统连接到 S5 CPU。S5-100U CPU 只能在 Profibus-DP 网络中作为被动组件，这意味着它们只有响应 Profibus 中的主动节点命令才可以传送或接收数据。

如果你正在使用 Profibus-DP 网络，这样安装了 S5-100U I/O 模块的一个框架可以用作其他控制器的从站，你就可以安装这些 I/O 模块到一个分散 I/O 模块（如带有 CP 318 Profibus-DP 接口模块的 ET200U），而不用购买 S5-100U CPU 和 CP 541。

其他带有 CP 模块的 ET200 分散 I/O 模块，也可以连接到 Profibus 网络。接口可用来连接到执行器-传感器接口（AS-I）网络，作为 Profibus-DP 从站。有 Siemens 提供的接口卡的个人计算机也可以被连接并作为主动组件。

通信处理器（CP）模块有 DIP 开关和跳线，它们可以被设置为：

1）通过在 CPU 存储器中保留的通信标志，使 CP 模块能够与 CPU 模块交换状态和控制信息；跳线选择 CP 模块和哪一个 32 位的通信标志区域一起工作；

2）指定一个惟一的接口号给这个 IM 模块（如果 IM 模块有 1K 的存储空间来和 CPU 交换数据，以及如果 CPU 配置给页帧通信）。

7. 连接 S5 编程器

带有内置的 RS-232C 到 4-20mA（当前电压）转换的电缆可以用来个人计算机的 RS-232C 端口和 S5 CPU 模块的编程端口之间的连接。

10.3.4 SIEMENS S7 硬件安装

S7

1. 安装 S7 CPU 和 I/O 模块

除了把 S7 PLC 装到控制面板，并用电线把它连接到交流电源、传感器和执行器上，S7 PLC 系统只需要很少的硬件设置。使用模块提供的导线将中心框架中的模块简单地插在一起。

一些 S7 CPU 模块中可以插入存储器模块卡，当电源关闭后，带有闪存 EPROM 存储器的存储卡存储关键数据（也可以使用带后备电池的存储器，一些集成的功能模块 CPU 包含 EEPROM 存储器㊀，因而不一定需要存储卡。）。软件配置（后面将讲述）告诉 CPU 什么数据是可保持的，什么数据应该保存在存储卡上或在 EEPROM 中。存储器卡的一部分可以编程（使用一个分离的程序设备），使用 CPU 模块不可以改变的数据。操作者可以清空几乎所有的 CPU 的 RAM 存储器和保持存储器的内容㊁，并通过执行存储器复位从存储卡（或者从 EEPROM）中重新加载数据。如果 CPU 从 RAM 或保持型存储器㊂中检测到一个问题，那必须存储器复位。为了执行一个完整的存储器复位，使用 CPU 模块上的开关键如下：

1）PLC 开关打到 STOP，然后

2）打到 MRES，保持直到 STOP LED 灯闪烁，然后

3）打到 STOP，然后

4）再次打到 MRES，并保持直到 STOP LED 灯停止闪烁。

一些模拟 I/O 模块在插入到 PLC 框架中之前需要硬件配置。这些模块有一个范围选择器，它们可以移去、翻转，然后重新插入来选择电压或者电流模拟信号范围。

电线连接器是终端插头单元，它可以插到 I/O 模块的前部，必须压下 I/O 模块上的连接器锁来允许插入连接器。每个 I/O 模块带有一个传送给插入 I/O 模块的第一个电线连接器的编码键。编码键然后阻止电线连接器插入到不同类型的 I/O 模块中（如果需要，可以从电线连接器中移走代码键。）。

Siemens 给 S7 框架插槽的编号和其他 PLC 中的编号有一点不同㊃：它是从最左边开始的 S7 插槽编号。

1）插槽 1 包含电源模块；

2）插槽 2 必须包括 CPU，或者被认为是空的；

3）插槽 3 必须包括 IM 模块，或者被认为是空的；

4）插槽 4 到 11 可以包含任意 I/O 模块的组合。

Siemens 提供 3 种类型的 I/O 模块：

1）数字或者模拟 I/O 的信号模块（SM）；

2）功能模块（FM），有自己的微处理器来进行专门的处理器控制的智能 I/O 模块；

3）通信处理器（CP），连接 PLC 到通信通道，除了那些通过内置串口来连接的 CPU。

2. S7 分散 I/O 系统安装

分散 I/O 系统有一个中心框架，它连接到近距离的扩展框架上，这样单一 CPU 模块可以控制更多的 I/O 模块，而且每个中心框架和扩展框架必须有一个接口模块（IM 模块）。局域网如 Profibus 可以使更多远距离的 I/O 模块受到中心 CPU 模块的控制，但这需要更复杂的通信处理器（CP）模块接口，在这一节中不讨论。

预先装配好的连接电缆用来连接扩展框架（带有 IM 模块）到中心框架上（包含 CPU

㊀ Siemens 的手册中在讨论电可擦除 PROM 型存储器时使用“EPROM”，其实更正确的叫法应该是 EEPROM。

㊁ 诊断缓存中的错误消息，本 CPU 的 MPI 节点地址，以及操作计数器中的内容不会被清除。

㊂ 在将新程序从编程器传送到 CPU 之前存储器需要复位。程序员可以从编程器初始化这次复位。

㊃ I/O 模块所在的插槽影响它的地址，见第 5 章的介绍。一些 PLC 允许用户分配不同于由插槽确定的地址。在这本书中我们不讨论用户分配的地址。

模块和一个 IM 模块)。一些 S7-400 IM 模块的寻址需要设置一个硬件开关来选择一个框架号，但 S7-300IM 是自我配置的，甚至不需要终端器。

3. S7 局域网安装

Siemens 提供 3 种类型的局域网接口：

1) 多点接口 (MPI) 网络。任意 S7-300 (或者更高型) 的 CPU 可以通过编程端口连接 (Siemens 将程序端口称为 MPI 端口)，MPI 连接器必须以菊花链配置进行电线连接。每个 MPI 连接器包括一个终端电阻，但是只有在菊花链的每一段的端点上的连接器通过开关打开终端电阻。在通过电线连接的 MPI 连接器将 CPU 内部互连之前，每个 CPU 和每个编程器必须配置它们自己的 MPI 地址，如在本节中的 S7 CPU 配置中介绍的那样。

一个 MPI 段可以包含 32 个节点，但一些段可以互相连接，允许多达 126 个节点。RS-485 转发器可以把段连接在一起，在一个段中加入 RS-485 转发器，可以减少一个段中允许的节点数，但转发器不需要 MPI 地址，并不减少整个 MPI 网络上可寻址的节点数。为了将一个 S7 CPU 连接到若干个 MPI 段 (或其他类型的网络)，S7 PLC 必须为每一个附加的网络连接配置一个通信处理器 (CP)。与固定布线的安装不一样的是，这里不需要进行硬件配置。

2) Profibus 网络有三种类型：灵活的消息服务 (FMS)、分散的外围 I/O (DP) 和过程自动化 (PA)。大多数 S7-300 CPU 需要 Profibus CP 模块来连接到 Profibus 网络上，但 S7-400 CPU 可以通过它们的程序端口，在它们配置了惟一的 Profibus 地址之后，直接连接到 Profibus-DP 网络，如后面所述。

对于 MPI 网络，在单个 Profibus 段中只能有 32 个节点，但 RS-485 转发器可以用来将段相互连接，这样可以连接 125 个可寻址节点。

两个类型的组件可以连接到 Profibus 网络上：控制器和从站。控制器可以是 S7-400 CPU 或者任意 S7 PLC 系统中的 CP 模块。每个段中的控制器必须指定作为 Profibus 主站 (节点 1)，通过其他控制器来控制网络的使用。从站和扩展 I/O 框架相似，在那里没有 CPU 模块 (尽管 Profibus CP 模块允许带有 CPU 的 S5 PLC 在 Profibus 网络作为从站)。从站可以只响应来自控制器的通信命令，但不可以初始化数据交换。一些从站设备有 DIP 开关用来指定节点数目，但不需要 Profibus 网络的其他硬件配置。

3) 工业以太网[⊖]，需要 CP 模块。工业以太网基于互连计算机 IEEE 802.3 的标准，它是办公室网络的最普通通信协议，其与 Profibus 和 MPI (以及大多数 PLC 使用的网络) 不同的是：以太网不需要为每个控制器安排一个令牌。一个单独的工业以太网子网最多可以连接 2024 个站点，并且无论何时只要发现网络是可用的，各个站点都有试图传输的权利。

4. S7 编程器安装

编程设备 (包括装有 STEP 7 软件的个人计算机) 使用一条 MPI 接口电缆，通过 RS-232C 端口可以直接地连接到 S7 的 MPI 端口。需要适配器使编程器和 MPI 网络可以共享 S7 CPU 的一个 MPI 端口。连接到接入 MPI 网络的一个 CPU 的编程器能够访问网络上的其他节点。

⊖ 尽管有 Profibus 支持者的反对，快速以太网 (100M 或更快) 已经被选作开放现场总线标准控制器网络的主干。

对于永久性地安装在 MPI、Profibus 或工业以太局域网的编程器，一定要安装网络接口卡，且 Profibus 网络只能有一个编程器这样连接。S7 网络接口卡是“即插即用”的，所以它们不需要硬件配置。如果个人计算机已经将默认的中断和/或存储器用作其他目的时，必须为接口卡分配新的中断号和存储器空间（通过软件，在下面解释）。

CQM1 10.3.5 OMRON CQM1 的硬件安装

CQM1 模块可以互相插入彼此的侧面，并且通过模块背后的滑动闭锁键固定在一起。组装好的 CQM1 被挂在 DIN 轨道上，通过先压低每个模块后面的闭锁键，然后把 CQM1 挂在轨道上，接着抬升闭锁键，将每个模块锁到轨道上。

CQM1 为小型的 PLC 系统使用，它不允许增加远程 I/O 的框架，但是 CQM1 可以连接起来互相通信。

1. CQM1 CPU 设置

CQM1 CPU 模块表面的六个 DIP 开关，可以在 CPU 装入 PLC 之前或之后设置。DIP 开关允许你选择：

1）Pin 1。关闭时，保护从 DM 6144 到 DM 6655 地址的程序存储器和配置存储器不被写。如果你要修改程序或配置，或当它启动时你希望 PLC 自动地载入来自卡式存储器的程序和配置，那你必须打开这个开关。如果选择保护模式，一定要安装后备电池，用来保留这些区域的数据。

2）Pin 2。打开时，当 PLC 启动时，导致 PLC 把拷贝卡式存储器的内容拷贝到 RAM。当然，开关 1 必须关闭，一个装有程序和配置数据的卡式存储器必须插入到 CPU 模块中。

3）Pin 3。打开时，当 PLC 启动时，使 PLC 在编程器上显示英文消息。如果 Pin 3 关闭，显示日语消息。

4）Pin 4。关闭时，阻止默认的扩展指令集的改变。在编程手册中一些扩展指令没有函数号（举例来说，AVG (-)），其他一些有函数号（举例来说，PULS (65)）。如果你使用没有默认函数号的指令，你必须打开 Pin 4，而且你必须先给这个指令重新分配一个默认函数号，再在程序中使用这个函数号（举例来说，AVG (-) 变成 AVG (65)，而且 PULS (65) 变成 PULSE (-))。我们将在后面看到如何重新配置函数号。

5）Pin 5。打开时，分配一个默认的协议到 CPU 的 RS-232C 端口。默认配置是 1 个偶校验位、7 个数据位、加上一个开始位和二个停止位，9600 bps。当 PIN 5 断开时，CPU 的软件配置指定另一个协议。

6）Pin 6。打开时，保持 AR 0712 开；断开时，关闭 AR 0712 。AR 0712 被称为 DIP 开关 Pin 6 的标志（明显的理由），但它不能被 CQM1 为任何目的而自动地使用。程序员可以编写程序来监控这个位。

对 Pin 1 和 Pin 2 的解释表明：你可以在 CPU 中插入一个卡式存储器，卡式存储器可以包含程序和配置数据，对这些卡式存储器可以使用几种类型和大小的存储器芯片，而且通过设置它上面的一对 DIP 开关来指示内部的存储器芯片的大小。存储器芯片可以是 EPROM 芯片——使用 PROM 烧写器来编写 CQM1 的程序和配置，也可以是 EEPROM 芯片——它可以通过一个 CQM1 CPU 模块写入（设置 AR 1400 使 CPU 下载当前的程序和配置到存储器模块）。为了防止意外的 EEPROM 重写，卡式存储器有一个写保护 DIP 开关，当它打开

时防止重写。当 PLC 的电源打开时，不要移出或者插入带有 EEPROM 芯片的存储器磁带。

如果 CQM1 的安装没有保护程序和配置数据的后备电池，程序和配置数据应该保存在卡式存储器中，CPU 的 Pin 1 和 Pin 2 应该允许将卡式存储器的数据载入 RAM，并且程序员应该知道 CQM1 其他存储器的内容在重启 PLC 后可能是不可靠的。DM、HR、AR 和 TC 存储器区域可能没有被清除；输出关闭位（SR 25215）在它应该关闭时可能没有关闭。用户程序应该在使用这些存储器之前准备它们：可能使用块设置指令、BSET（72），清除 HR、AR、CNT，以及 DM 存储器没被配置的部分，并且通过使用总是 OFF 标志来保持输出 OFF 位关闭。

在一些 CPU（CQM1-CPU42E）中，CPU 模块上的四个旋转接口能被用来调整存储器 IR 220 到 223 的内容。CPU 将在每个扫描循环扫描这些接口，并且把从 0000 到 0200 之间的 BCD 码放入存储器来反映他们的状态（接口设置可能被用来人工设定定时器预设值）。

2. 连接 CQM1 I/O 模块

只有模拟 I/O 模块需要设置。模拟输入模块需要设置一些 DIP 开关对，从而选择每个输入通道的电压或者电流范围。另外一个 DIP 开关选择给模拟输入值保留二个还是四个输入映像字，还有一个 DIP 开关用来选择输入数值是否要取平均。模拟输出模块的双通道的每一个都有跳线，跳线允许或禁止通道输出负模拟信号。模拟 I/O 模块的供电由插入 PLC 的供电模块提供，好像它是一个 I/O 模块，然后被连接到模拟 I/O 模块上。

CQM1 有一个特殊的 I/O 寻址系统。插槽从 CPU 模块开始编号（CPU 是插槽 000），输入模块被分配的 I/O 映像地址是从 001 到 011，输出模块被分配的 I/O 映像地址是从 101 到 111。不管有多少个输出模块插在他们之间，被分配的第一输入映像地址总是 001，第二个是 002，等等；也不管有多少输入模块插入他们之间，输出模块被同样地分配地址 101，102，等等。框架上的模拟电源不影响寻址。

每个数字 I/O 模块被分配 16 个位（一个字）的 I/O 映像存储空间，但是模拟 I/O 模块有二或四个字的 I/O 映像，这取决于他们有多少个通道。

3. 连接 CQM1 编程器

连接 CQM1 的外围端口到运行编程软件的个人计算机的 RS-232C 端口，需要一条包含协议转换器的连接电缆，或者用户可以用一条 RS-232C 电缆（在 CQM1 的设置手册中有说明）来连接 CQM1 的 RS-232C 端口到个人计算机的 RS-232C 的端口。

4. 为通信互连 CQM1 CPU

CQM1 CPU 可以通过 RS-232C 端口连接外围（例如打印机、条形码设备等），或通过 RS-232C 端口互相通信。如果三线的 RS-232C 电缆把两个 CQM1 RS-232C 端口的信号地（SG 引脚）连接在一起，并且将每个 CQM1 的 RXD/RD[⊖] 引脚连接到另一个 CQM1 的 TXD/TD 引脚，那两个 CQM1 CPU 可以配置成点对点的通信。一个局域网最多可以连接 32 个 CQM1，它们将作为单一主机（不是一个 CQM1）的从站，因而主机可以控制从站进行通讯。CQM1 端口和主计算机的信号地（SG 引脚）必须连接在一起，CQM1 的 TXD/TD 线必须互相连接在一起，然后连接到主计算机的 TXD/TD 线。

⊖ 老的 RS-232C 标准使用如 RXD 和 TXD 这样的引脚名。RS-232C 标准已经更名为 IEA-232D，并使用新的引脚名（对应为 RD，TD）。但是大多数使用者（除了 OMRON）仍使用旧的名称。

10.4 为一个应用准备的 PLC 系统的第一次配置

如果你已经购买了一台带有经过预配置的编程器或手持式挂件的小型 PLC，你只需要简单地打开电源，就可以开始对 PLC 编程了。在你准备编程之前，更强大和更灵活的系统需要更多的步骤。你可能要：

1）在你的个人计算机上安装编程软件。在本书中，我们很少介绍如何安装软件。

2）配置软件离线编程特性。离线编程允许程序员在 PLC 没有连接到编程软件的时候，编写程序和数据文件，程序和数据稍后可以被下载到 PLC 中。

3）配置编程软件以便它能与你的 PLC 的 CPU 模块进行通信，配置它的在线程序编程特性。

4）按用户习惯配置 PLC 的操作特性。

5）配置 PLC CPU 模块与其他控制器和远程 I/O 进行通信。

6）配置智能 I/O 模块，按照应用需要运行。

PLC-5

10.4.1 Allen- Bradley PLC-5 的第一次配置

1. 安装 PLC-5 编程软件和离线编程

在软件安装时，你将会被问到是否需要使用特权密码保护。如果你回答是，你将能够建立一个带有密码的系统来保护你的程序、数据和配置，不会被未经许可的用户改变（如果你回答是，你一定要按照这章所说的方法对特权密码进行配置，以防止一个未经许可的用户在你之前这样做，使你不能使用）。运行安装程序来装载编程软件到你的个人计算机的硬盘上后，启动编程软件。

Allen- Bradly 的 6200 编程软件在“显示属性”配置菜单中有一个链接模式（link mode）选择。任何时候当个人计算机传送数据包到 PLC，开启链接模式会导致个人计算机请求一个来自 PLC 的 ACK 信号（确认接收信号）。开启连接模式将会因此减慢通信速度，但是可以增强个人计算机的编辑软件和 PLC 之间通信的可靠性，如果数据包在传输过程中被损坏，这种模式将有利于避免可能存在的不安全状况。

编程软件的离线编程配置包括：选择程序下载到什么型号的 PLC-5CPU 中以及选择显示属性，例如编程器如何显示梯级、梯级标号、符号等等。离线配置不在这里讨论。

2. 连接 PLC-5 CPU 和在线编程

为了使个人计算机上的编程软件与 PLC 通信，它的在线编程属性必须进行如下选择性设置：

（1）包含字母和数字的编程终端名字（可选择的）；

（2）PLC 连接到个人计算机的哪一个通信端口。这一选择由以下组成：

1）识别安装在个人计算机上的通信卡型号；

打开卡上的 DIP 开关之后，要求程序员输入通信卡的地址（如果你没有记录它，Allen-Bradley 的 ABHELP 软件通常能够读取卡的 DIP 开关设置）。如果你使用串口来代替通信卡，你会被要求设定所选串口的通信协议。

2）识别连接编程器到 PLC-5 CPU 的数据高速公路的类型，对于在线编程，它是默认的。

■ 选择 DH＋连接到一个 CPU，这个 CPU 作为编程终端被连接到相同的 DH＋（尽管你或许已经在 PLC-5 CPU 模块的连接器上插入编程电缆，但是它实际上被连接到包括这个 PLC-5 的 DH＋上。）。图 10-2 显示了一台个人计算机连接到包含两个PLC-5 CPU 的 DH＋中，其中任意一个 PLC 可以设置为在线编程的默认 PLC。

作为一个 network access（网络访问）选项，选择 Local，然后输入：

- CPU 模块的 PLC 地址（DH＋地址）；
- 编程终端的终端地址（DH＋地址）。

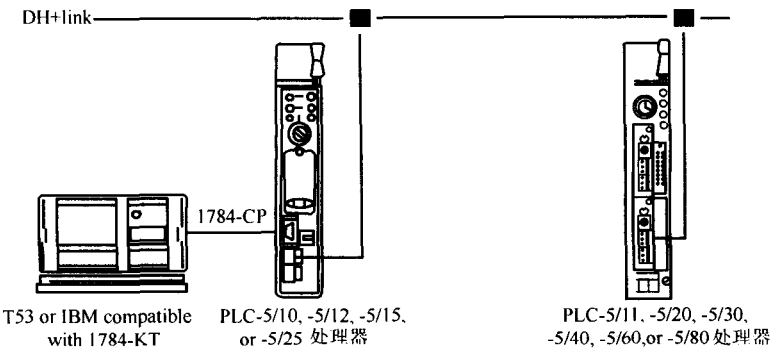


图 10-2 一个本地 DH＋上的一个编程器和两个 PLC-5CPU

■ 选择 DH＋也允许你连接到在其他 DH＋上的 CPU，这两个 DH＋通过数据高速公路连接。图 10-3 显示一台个人计算机连接到 DH＋上，这个 DH＋又通过数据高速公路与另一个 DH＋互连，在这两个 DH＋上的任意一个 PLC-5 可以被设置为在线编程的默认 PLC。作为一个网络访问选项，选择 Remote（DH＋），然后输入：

- CPU 模块的 PLC 的地址（DH＋地址）；

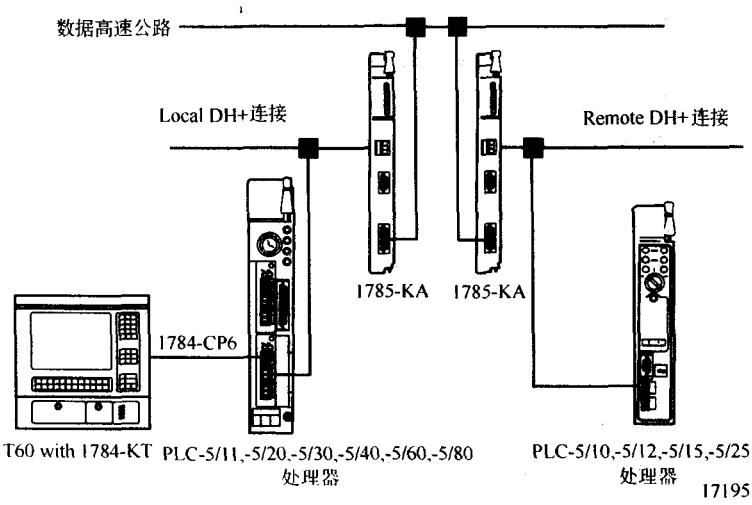


图 10-3 一个 PLC-5 编程器，两个 DH＋，一个数据高速公路

- 编程终端的终端地址（DH＋地址）；
- 网络中的 1785-KA/KE 模块的数据高速公路（不是 DH＋）地址，1785-KA/KE 模块

作为局域网桥来连接一个包括远程 DH＋的数据高速公路系统；

- 远程 DH＋中的 1785-KA/KE 模块的数据高速公路（不是 DH＋）地址，1785-KA/KE 模块作为远程网桥用来连接 DH＋到数据高速公路系统。

■ DH＋路由　允许连接 PLC-5，如果到其他 CPU 的电气路径包括 Allen-Bradley 金字塔积分器底板，你也必须输入：

- CPU 模块的 PLC 的地址（DH＋地址）；
- 编程终端的终端地址（DH＋地址）；
- 连接这个 DH＋网络到底板的网桥 5130-RM/KA 模块的局域 DH＋地址；
- 连接远程 DH＋网络到底板的目的连接站点 5130-RM/KA 模块的地址。

■ DH II　允许通过 II 型数据高速公路和其他 PLC-5 连接。你必须输入以下的信息来辨识：

- 源（SRC）链接，包括编程器地址（源用户），在局域 DH＋上的 1779-KP5（源节点），DH II（源链接）的地址，通过这个 DH II，源节点和在远程 DH＋上的目的节点可以通信。如果源节点和目的节点经单一的 DH II 链接在一起，源链接的地址可以输入 0。如果其他链接包括在这个 DH II 和远程 DH II 之间，输入局域 DH II 网络的实际地址 。
- 目的（DST）链接，包括 PLC-5 地址（目的用户），远程 DH＋上的 1779-KP5 节点地址（目的节点），以及 DH II（远程链接）的地址；通过这个 DH II，1779-KP5 和远程DH＋通信。如果 DH II 直接把目的节点链接到源节点上，它的远程链接地址可以输入 0。如果其他链接包括在此 DH II 和远程 DH II 之间，输入远程 DH II 网络的实际地址。

3. 配置 PLC-5 CPU

既然你的编程器已经被告诉该如何连接到 PLC-5 上，通过选择在线编程你就能连接到 CPU 模块上。在线编程模式中，你能编辑程序并且改变存储器内容，但是在这么做之前，你需要调用配置屏幕和输入数据来使 PLC 的实际应用操作特性符合用户需要。

当你使用配置屏幕输入配置数据时，CPU 的配置数据被存储在存储器区域的状态文件（S:）中（PLC-5 存储器的状态文件区域的结构在附录 A 中，它也包括反映 PLC 状态的字。）。PLC 程序可以读状态文件的任意一个数据。一些配置数据是动态的，这意味着 PLC 程序能改变它们，并且这种改变可以立刻生效。其他的配置数据是静态的，表明只有当 PLC 开关打到运行模式，这种改变才能生效，甚至在程序执行时也可以改变它。

PLC-5 带有备份电池，因此，包括配置变化的状态文件数据将保持有效（即使没有使用电源），直到程序员或程序再一次改变了配置。当你把程序保存到编程器时，你也保存了状态文件，而且当你再次装载程序到 PLC 时，你也要再次装载了配置文件。

增强型 PLC-5 CPU 的处理器配置屏幕如图 10-4 显示。在经典的 PLC-5 状态屏幕中包含很多相同的配置选项，但是它们看起来不同。如程序员移动光标经过屏幕区域，配置选项的实际状态文件地址在屏幕的底部显示出来，而且程序员能输入新的数据。

处理器配置屏幕使程序员可以访问配置选项，它们包括下列各项（我们在这一本书中不会包括所有的配置选择项）：

（1）启动配置

用户控制位（在 S:26）的低级位可以被置位或复位，来控制你的 PLC 如何重启，在运行模式下电源被重新加入时或者每次 PLC 开关打到运行模式时。

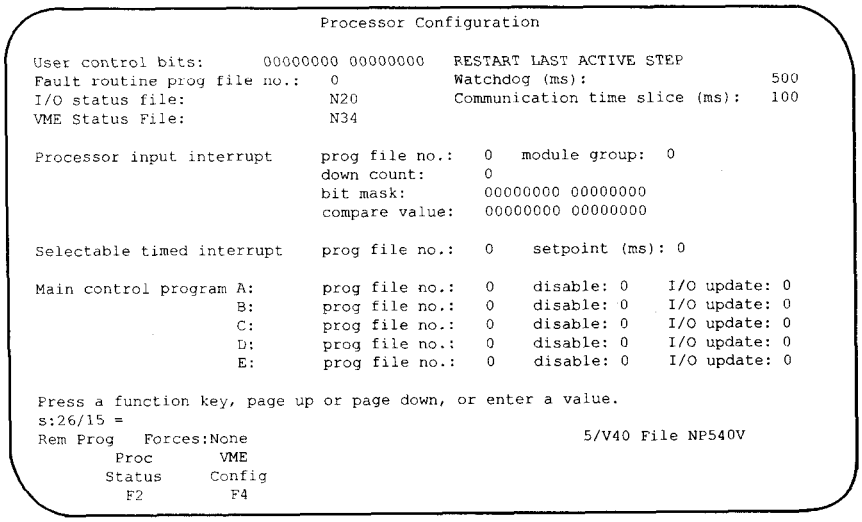


图 10-4 处理器配置屏幕（增强型 PLC-5 处理器）

如果你置位位 S：26/1，然后每当 PLC 重启，它将会设定一个主出错位（S：11/5）以便 PLC 立刻试着运行出错例行程序，出错例行程序的文件号（S：29）也必须使用这一配置屏幕输入。如果出错例行程序清除主出错位，出错例行程序运行结束时，PLC 开始执行标准的扫描循环。PLC 重启时，如果位 S：26/1 没有被置位，PLC 立刻开始执行标准的扫描循环。出错例行程序将在第 11 章中描述。

位 S：26/0 只影响顺序流程图（SFC）程序的执行。如果 PLC 在运行模式中被重新启动，或者 PLC 转到运行模式时，S：26/0 置位，PLC 将会像通常一样开始执行扫描循环，但是会从 PLC 上次中断时已在执行的那一步开始继续顺序执行 SFC 程序。PLC 重启时，如果 S：26/0 没有置位，SFC 程序从初始步骤重启。SFC 在第 9 章描述。

（2）主控程序

PLC-5 可以配置成在每个扫描循环周期中执行多达 16 个叫做主控制程序（MCP）的用户程序，或一个也不执行。使用处理器配置屏幕输入 MCP 文件号，文件号存储在每个文件的第三状态文件字中，从第一个 MCP 文件（MCP “A”），它的文件号在 S：80 中，到第十六个 MCP（MCP “P” 在 S：125 中）。状态文件中的每个 MCP 文件号之后，下一个字存储最近的 MCP 扫描的持续时间，再下一个字存储 MCP 最长的扫描时间。

PLC-5 可配置成在任意或者所有的 MCP 之后，是否执行输出扫描和输入扫描。字 S：78 的 16 位包含 16 个可能的 MCP 文件的 I/O 更新位。对这个字中的一位置位（例如，位 S：78/0），在对应的 MCP 之后禁止 I/O 更新（例如，MCP “A”）。

处理器配置屏幕可以用来禁止任意或所有的 MCP 的执行，而不需要将它们从配置中移出。使用字 S：79 的 16 位，如果一个位置位，对应的 MCP 将不执行。

（3）看门狗定时器

每个扫描循环开始时，PLC-5 启动一个看门狗定时器。如果定时器所定时间在扫描循环结束之前溢出，PLC 将会出错（查看第 11 章获取更多的细节资料）。程序员可以使用处理器配置屏幕为定时器看门狗输入一个以毫秒为单位的时间，这个值放入 S：28 中（每个扫描

循环结束时，PLC-5 把扫描所花费的时间写入 S:8)。

(4) 中断

使用处理器配置屏幕，可以配置三种类型的中断。全部在第 11 章中详细地说明。

1) 输入（到 S：29）出错程序文件号；

2) 输入（到 S：31）可选择的定时中断（STI）文件号，并输入（到 S：30）执行 STI 文件的间隔时间；

3) 输入（到 S：46）处理器输入中断（PII）文件号。程序员可以输入一个模块组号和一个屏蔽位（分别为 S：47 和 S：48），来配置哪一输入模块和哪一触点可以是中断源，并且可以输入比较值和向下计数值（分别为 S：49 和 S：50），来配置位传输类型和表明在中断发生前需要传输多少位。

(5) I/O 状态

如果通过 PLC-5 模块扫描远程 I/O 框架，需要使用处理器配置屏幕输入 I/O 状态文件号（使用 DIP 开关把 CPU 设置在扫描模式）。

输入一个文件号（例如，13），PLC-5 将这个文件号保留给整型文件（N13），来保存保持在 I/O 框架中的 I/O 状态。在 PLC-5 系统中，I/O 状态文件将包含对应每个 I/O 框架的两个 16 位的数据字（N：13.0 和 N：13.1 对应框架 0；N：13.2 和 N13.3 对应框架 1，等等）。第一个数据字包含状态信息，第二个数据字节用来配置和控制 I/O 框架（控制和配置 I/O 框架 1 的数据写入 S：13.3 中）。

在数据字中，每个数据位包含两个 I/O 组的状态或配置信息，因此，需要 4 个位来配置或控制一个标准的八组 I/O 框架：

1) 框架配置字中的位 0 到位 3 置位时禁止 I/O 组（例如，如果 S：13.3/0 置位，框架 1 中的组 0 和组 1 被禁止，并且不能给 CPU 提供输入映像数据，或接受来自 CPU 的输出映像数据。）。一般情况下，整个框架会同时被禁止，而不只是二个组。

2) 如果有一个出错，框架配置字中的位 8 到位 11 用来复位 I/O 组。如前面讨论，框架中的 DIP 开关可以设定允许 CPU 提供复位或需要手动复位（例如，S：13.3/9 置位可以复位框架 1 中的 I/O 组 2 和组 3。）。一般情况下，整个框架会同时被复位。

3) 位 4 到位 7 和位 12 到位 15 没有使用。

框架状态字包含用来指出 I/O 模块的当前状态和哪一个 I/O 框架出错的状态位（位 0 到位 3）（例如，如果位 S：13.2/2 置位，则框架 1 出错。）。I/O 框架中的任一错误都会引起 4 个错误位置位。

(6) 通信时间片

PLC-5 系统通过 DH1 系统与其他的 PLC-5 系统进行串行通信，通过远程 I/O 扫描系统与远程智能 I/O 模块串行通信，并且经过其他串口与其他设备通信。通信处理芯片承担大部分工作，但是在扫描循环的 I/O housekeeping 步骤，CPU 的主处理器必须服务于通信处理器（通信请求和交换数据）。(在第 13 章有较详细的解释。）通信任务可以由用户程序中的命令提出，或者是响应来自其他 PLC-5、智能 I/O 模块的请求，或者其他外围设备的请求，因此每一扫描循环中需要的清理时间是可变的。

如果通信时间片输入时间 0，PLC-5 将会在每次扫描时执行所有通信任务，因此扫描时间的长短更不确定。如果输入一个不为 0（毫秒）的时间，那么在每次扫描时，PLC-5 都将

使用这个数量的时间（加上 housekeeping 所用的基本 3.1 毫秒）来进行通信服务，而不管通信任务实际需要多少时间。

4. 配置 PLC-5 的通信通道

PLC-5 在面板上有一些带连接器的通信端口。在线编程中，通信端口的配置是比较容易的；在离线编程中，通讯端口也可以配置。为了改变配置，必须使用如图 10-5 所示的编程器的通道屏幕。屏幕显示了每一个通道的默认配置使用，允许操作员改变默认值，而且允许程序员选择一个通道进行进一步的配置，或者对其通信状态进行监控。

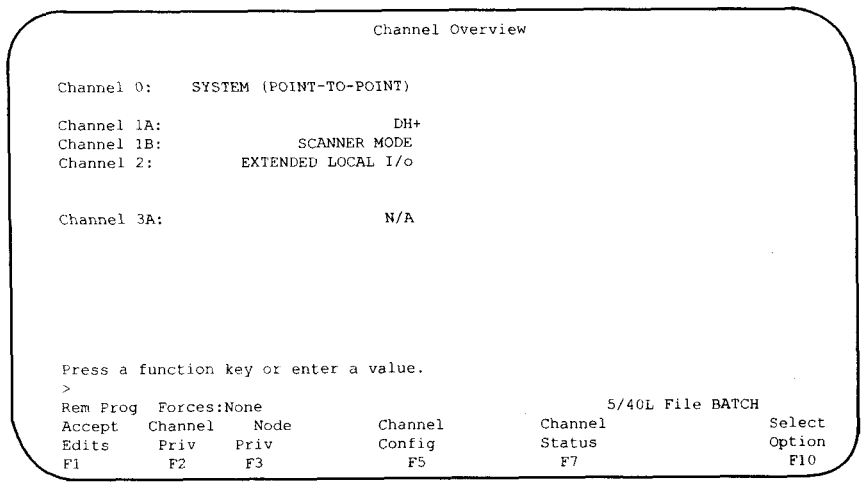


图 10-5 PLC-5 通信通道观察屏幕

保存程序时，也保存了通信配置数据。当程序下载到 CPU 模块时，通信配置数据也下载了。通信通道可以做如下的配置：

1）数据高速公路加（DH+）。通道 1A 是默认的 DH+通道，如果你必须改变通道 1A，首先为 DH+配置其他的通道，然后把编程器连接到其他通道上。可以有超过一个通道被配置成 DH+通道，这样 PLC-5 可以被用作两个 DH+网络之间的链接。

把一个通道配置为 DH+，包括输入两个没有使用的整型文件地址：其中一个，为了在此节点上的 DH+通信，存储 40 个字的诊断文件；另一个为 DH+网络存储一个 64 字的全局状态标志文件。Channel 1A 的 DH+的波特率必须设为 57.6 千波特，但是通过 Chanel 1B 或 2B 也可以设置 230 千波特。通过 Channel 1A 连接的 PLC-5 的节点地址是通过 DIP 开关设置固定的，但是输入的一个节点地址不是给 Channel 1A，而是给 DH+通道的。如果 DH+被连接到一个有多于一个 DH+网络的系统时，每一个 DH+必须要有惟一的链接 ID 号，配置通道的时候，你一定要输入 DH+的链接 ID 号。当然，连接到相同 DH+上的所有节点必须配置相同的链接 ID 号。

2）远程 I/O 扫描。远程 I/O 扫描可以让 PLC-5 作为扫描模式的 CPU（使用 DIP 开关），对远程 I/O 适配器模块或者作为适配器模式 CPU 的 PLC-5 模块，读输入数据和发送输出数据。

通道 1B 是默认的远程 I/O 扫描通道。进一步的配置包括指定一个未被使用的整型文件作为诊断文件，指定通道的波特率（必须和远程 I/O 适配器模块的 DIP 开关设置的波特率

相匹配），指示补充 I/O 框架是否存在（如果把远程 I/O 适配器模块配置成一个补充框架），并且建立一个扫描列表。扫描列表识别出哪一 I/O 框架和部分框架存在，所以 PLC-5 不会浪费时间向不存在的 I/O 点读取或写入数据。如果配置是在在线编程中完成的，并且 CPU 模块不处在运行模式，如果使用处理器配置屏幕指定了 I/O 状态文件，并且没有 I/O 状态文件的框架禁止位为开，PLC-5 能为你建立一个扫描列表！你只要简单地选择“clear list”，然后选择“automatic configuration”，然后读扫描列表以确认它与你已经连接的框架是否相匹配。

3）如果配置 PLC-5 作为（通过它的 DIP 开关）适配器模式的 CPU 而不是扫描模式的 CPU，一个通道（通常是 channel 1B）必须被设置作为远程 I/O 适配器模式通道。PLC-5 将会提供输入数据给扫描模式 CPU，从一个扫描模式的 CPU 接收输出数据。

进一步的配置包括输入配置数据，你将使用远程 I/O 适配器模块上的 DIP 开关（例如，框架数、波特率等。），但是你也必须为这个 PLC-5 输入两个未被使用的 BT 元素的地址（或两个不用的整型文件），使用它作为控制元素和扫描模式的 PLC-5 进行数据交换。

4）通道 0 有一个标准的 25 针连接器，它用来将 PLC-5 连接到标准计算机和外围设备。DIP 开关选择通道 0 的部分串行通信协议（RS-232C 或 RS 422B 或 RS 423），但是程序员可以通过通道观测屏幕配置其他的串行通信协议，以如下选择之一开始：

- DF1 点对点协议（默认，用来连接 PLC-5 到其他的计算机）；
- DF1 从站模式协议或 DF1 主站模式协议（允许多个计算机和 PLC-5 互连）；
- ASCII 协议，也叫用户模式（用来连接到外围，如条形码设备或串行打印机）。

串行通信选项在第 13 章中将再次提到，但是它们在本书的讨论范围之外。

还有其他通道配置（扩展 I/O 框架扫描、以太网通信、专用协处理器接口）不在这里讨论了。

5. 配置 PLC-5 的智能 I/O 和模拟 I/O

PLC-5 的智能 I/O 模块叫做块传输（BT）模块。模拟输入和输出模块是 BT 模块，其他 BT 模块包括高速计数器模块和电源监控模块。

PLC-5 CPU 必须配置成可以识别智能 I/O 模块，在块传输数据开始前，必须与智能I/O 模块进行通信。当 PLC-5 程序保存到磁盘时，CPU 模块包含的所有智能 I/O 配置数据也同时被保存。把程序上传给 PLC 时可以重新加载配置数据。对块传输 I/O 模块的 I/O 配置过程有以下几个步骤：

1）查看 BT 模块的用户手册，找出需要多少块传输（多少 BT 元素）和每个数据块必须包括多少个数据字；

2）写一个包含与 BT 模块通信需要的块传输读（BTR）和块传输写（BTW）指令的程序。输入 BTR 和 BTW 指令将会自动创建 BT 元素（整型文件控制元素）和数据块，你将在下一步的 I/O 配置中需要它们。带有指令（I/O 模块地址、数据块的位置和大小）的数据也会放到 BT 控制元素中（提示：记录你已经对哪一个 BT 控制元素编程和每一个用来做什么。）；

3）选择“I/O overview”选项，或者用光标指到 BTW 或者 BTR 指令上，选择“I/O edit”，然后选择“add new module”，并且使用下面的菜单来选择你所要使用的 CPU 的 I/O 模块的准确类型。一旦模块类型选择好，将显示图 10-6 的屏幕。在第二步中输入你要创建

的 BT 控制元素。(你不希望按推荐的方式保存记录 I?) PLC-5 将找到块传输数据文件的地址、长度，并在屏幕上显示这些数据。当接受配置后，你将在 I/O 观察屏幕上再次看到，并且配置包括新选择的 I/O 模块描述、地址、BT 元素和数据文件，如图 10-7 所示；

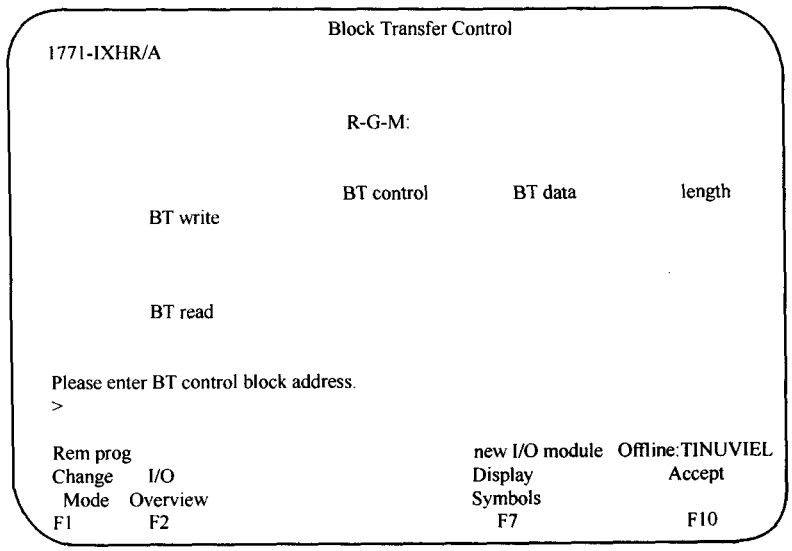


图 10-6 为 PLC-5 BT 模块输入 BT 元素

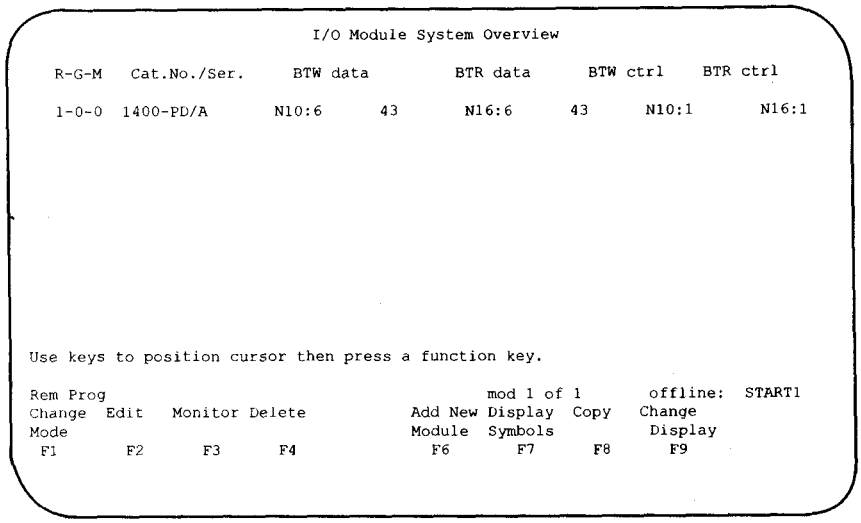


图 10-7 在加入一个 BT I/O 模块后的 PLC-5 I/O 观测屏幕

4) 选择“EDIT”，你将看到模块编辑屏幕，指定 BT I/O 模块的类型，在那里进一步输入配置数据。还需要一个或更多的其他配置屏幕，通过选择 edit/setup、counter setup、output/device setup 或者类似的选项，根据你所配置的 BT I/O 模块类型来定。

模块编辑配置数据包括：

■ 模拟输入模块：信号采样率、输入滤波、模数转换过程中生成数的二进制数格式和

传感器类型的选择；

- 模拟输出模块：转换为模拟输出信号的二进制数格式。

在通道编辑屏幕，你可以选择各种属性，如模拟信号的类型和范围；你可以指定模拟信号范围，它们分别是代表两个端点的最大值和最小值，这样模拟输入模块将把模拟输入信号标度变换到工程单位，或者模拟输出模块把工程单位值标度变换为模拟输出信号；你也可以为模拟输出通道输入一个新的输出值。

在这一步骤和下一步骤中，PLC-5 将会接受你写到数据文件中适当位置处的输入内容，BTW 指令将最终写到 BT 模块中。当 BTW 执行时，这个数据将用来配置智能 BT I/O 模块。

如果你工作在在线编辑模式，I/O 编辑软件将提供一个选项，可以立刻传输新的配置数据到 BT I/O 模块中，而不用等待 BTW 指令执行。如果你想改变一个模拟输出值，这个选项特别有用。

5）编辑你的 PLC 程序并执行：

- 每次重新启动 PLC 执行用户程序时，BTW 指令把配置数据写到 BT 模块至少一次。I/O 模块的其他 BTW 或 BTR 指令不能执行，直到配置数据传送成功。作为其他 BTW 和 BTR 执行的一个条件，对于将配置数据写到 BT 模块的 BTW 指令，检查 BT 控制元素（BTx. y. DN）的“done”位；
- 根据应用需要，BTR 指令经常读取来自 BT I/O 模块的状态信息（包括输入数据）；
- 根据应用需要，BTR 指令经常把其他控制数据（包括模拟输出值）写到 BT I/O 模块中；
- 在 BTR 指令从 BT I/O 模块中成功地读取信息之后，执行使用这些刚读入的数据指令；对于确定 BTR 指令存储的特殊数据项的实际地址，设置手册和（或）I/O 监控屏幕有很大帮助。你可能要写程序使每个 BTR 指令成功执行后，对接受数据进行操作，这由 BTR 的 BT 元素的“done”（. DN）位控制。
- 把要输出的数据写到数据块中的指令，这些数据块由 BTW 指令写到 BT I/O 模块中。使用你的设置手册和（或）I/O 监控/编辑窗口，确定要写入的数据块地址。你可能希望写程序使每次输出数据改变后，仅对 BTW 指令执行一次；这可以通过锁住控制 BTW 指令的指令，或者在 BTW 的控制逻辑保持为真时，通过把 BTW 的 BT 元素的“enabled”位（. EN）锁定在关状态来实现。

6. 配置访问特权

你可以通过分配特权来阻止未授权的读和/或修改程序、数据或者 PLC-5 的配置，而授权的用户可以访问（通过执行本节介绍的配置，你会明白阻止未授权的改动意义）。你可以在软件安装时打开或禁止这个功能。

即使没有安装口令和特权选项，你仍然可以使用处理器功能菜单给 PLC-5 的存储内容分配（或删除）一个口令。下一次编程器试图访问 PLC-5 的存储器时，用户将被要求输入口令，如果用户不知道口令，监控和编辑将被禁止。

如果你已经安装了口令和特权选项，就可能得到有更多访问选择的系统。你可以分配一个允许访问 PLC-5 存储器的特权系统，每一个 PLC-5 必须分别地配置访问特权。如果想充分利用特权选项，可以按以下的一些步骤操作：

1）在 processor functions 菜单中，选择 PASSWORDS AND PRIVILEGES，屏幕将会显示 4 个特权等级，特权选项也会出现。

2）选择 MODIFY PASSWORD，输入一个特权级别名，四个特权级别名（级别 1 到 4）都是可以输入的。输入口令后，你将得到相应特权级别。这个口令将会保存在 PLC-5 的处理器中，也会保存在从 PLC 中下载的程序里。如果忘记口令，你需要把后备电池和 EEPROM 数据存储设备取出，PLC 在断电时口令将会消除，并且在 PLC-5 存储器中的内容也会消除。（提示：现在就记住密码）

3）选择 TOGGLE PRIVILEGES，选出你给每个用户的特权级别，给监督者分配最大的特权级别 1，监督者可以根据需要改变其他等级的特权。特权包括创建、删除或者编辑程序和数据文件的权利，从 CPU 中上传和下载文件的权利，使用 I/O 端口的权利，甚至从远程终端改变处理器模式的权利（运行和编程模式）。

4）在 General Utilities 菜单的 channel overview 中选择 CHANNEL PRIVILEGES。默认的，连接编程终端的通道显示特权级别 1。既然级别 1（应该）有权利去改变特权，你的编程终端能被用来指定默认（更低）的等级来使将来的用户通过每一个通道访问这个 PLC-5。

5）通信通道的单一节点可以被分配比步骤 4 中更高级别的特权。你也许需要一个工程部门的编程器，或者一个运行监控程序的 PLC，它们可以有比默认权利更高的特权级别。在同样的菜单中选择 NODE PRIVILEGES 选项，你会找到 CHANNEL PRIIVLEGES，使用 select privileges 锁住通道，输入工作站的地址和节点的连接 ID（如果在另一个 DH＋网络上），然后输入（更高的）特权等级。

6）在编程器允许查看和改变文件前，在 PLC-5 中的单独程序文件和数据文件可以被配置成编程器需要的指定特权级别。使用处理器功能菜单来选择程序文件，或者存储器表菜单来选择数据文件，然后通过 TOGGLE PRIVILEGES 选项为文件锁定特权级别要求。

任何用户可以通过选项 Enter Password 获得比默认更高的特权级别，并为更高级特权输入口令。

10.4.2 ALLEN- BRADLEY SLC 500 的第一次配置

SLC 500

1. SLC 500 离线编程配置

如果你的 SLC 500 PLC 系统包括带有 M0/M1 存储器的特殊 I/O 模块，你可能需要在软件配置期间选择 Enable M0/M1 monitoring。为了写离线程序，首先要选择 Offline Configuration。在离线配置中，你必须选择写入程序的 CPU 模块的型号，而且你要为 CPU 模块将包含的程序输入文件名。

你也必须使用 Offline Configure 菜单来输入 I/O 配置数据，这些数据包括由 CPU 控制的 PLC 系统每个插槽中的框架和 I/O 模块的描述。可以从菜单中选择模块的描述，或者通过 Read Configuration 选项，它导致编程器连接到 PLC 上、检测 PLC 系统以及自动地生成 I/O 配置系统描述。

因为 Read Configuration 需要编程器连接到 PLC 上，所以 I/O 配置菜单包括 Online Configure 选项。在线配置在下面将要说明。

2. SLC 500 在线编程配置

进入在线编程之前，你必须执行“Online Configuration”。在线配置中，指定编程器和PLC 之间安装哪一通信接口硬件，通过从接口选项的菜单中选择，完成接口的指定。

你必须也选择“Driver Configuration”来为接口硬件做更进一步的配置。选择接口硬件的驱动接口配置屏幕出现，在这个屏幕上，程序员必须配置如下项目：

1）程序终端地址（0 到 31），所以网络上的一个 PLC 系统可以由多个编程器操作；

2）在线编程初始化后（0 到 31），SLC 500 CPU 模块要连接的地址。默认的，连接到节点 1，节点 1 是新买的 SLC 500 CPU 的默认地址。如果在配置过程中你改变了 CPU 的地址（CPU 的配置将在下面说明），你必须修改在线编程配置参数；

3）个人计算机中的通信接口卡（如果使用）的地址，所以编程软件可以通过它进行通信。这个数应该和卡上的 DIP 开关的设置相匹配；

4）通信接口卡的中断请求（IRQ）号，以便在它接收来自 PLC 的数据后，可以通知编程软件。（在默认中断号给你的个人计算机中的其他设备使用前，你可以使用默认的 IRQ 号）；

5）通信波特率；

6）如果在编程器和 SLC 500 中的接口硬件包括硬件连接 DH＋、DH-485、调制解调器等等，接口模块的地址和任何其他通信协议的描述也必须输入。查看你的用户手册获取更多的信息。

3. 配置 SLC 500 CPU

CPU 的配置数据保存在状态文件（S:）的存储区域，通过处理器状态屏幕你可以配置 SLC 500 CPU。SLC 500 CPU 的处理器状态屏幕显示存储区域的状态文件的内容，包括标号来识别各数据项的意义。处理器状态屏幕在数据监控菜单中，当光标在配置项时，程序员可以输入新的配置数据。当程序员移动光标到处理器状态屏幕区域时，每个配置的实际状态文件地址或者状态选项会在编程器屏幕的底部显示出来。

SLC 500 有后备电池，因此，状态文件数据包括配置更改，将会保持有效（即使电源断开），直到程序员或程序再一次改变配置。每当你把程序存入编程器，你同时保存了状态文件，当你把程序重新装载到 PLC，你也装载了配置。

PLC 的程序可以读状态文件中任意数据。一些配置数据是动态的，这意味着 PLC 程序能改变它们，而且改变立刻生效。其他配置数据是静态的，这意味着 PLC 程序可以改变它们，但只有 PLC 在运行状态时这种改变才生效。状态文件也包含反映 PLC 状态字，SLC 500 存储器的状态文件区域的结构显示在附录 B。

可以被修改的配置选项，包括下面所讨论的。（只有较好的 SLC 500 型号可以提供所有的这些选择项；附录 B 表明了哪种处理器提供哪种选项。）

（1）启动配置

位 S：1/10 到 S：1/12 是装载存储器模块控制位，它们用来选择在什么条件下 PLC 将把 EEPROM 模块中的内容拷贝到 RAM 中。

位 S：1/9 被称为启动保护出错位。如果该位置位，PLC 工作在远程运行模式；当电源出错，PLC 将会运行出错例行程序，错误例行程序文件号必须使用这个状态窗口输入到 S：29中。如果出错例行程序清除了主错误停止位(S：1/13)，然后当出错例行程序完成执行

后，PLC 将开始执行标准的扫描循环。当 PLC 在重启时，如果 S：1/9 位没有置位，需要看 S：1/8 位来决定是否重启。出错例行程序将在第 11 章中说明。

位 S：1/8，上电出错跳过位，当电源打开时，将引起 PLC 清除它的主错误停止位（S：1/13），使 PLC 再一次运行程序。电源恢复时，如果 S：1/8 位没有置位，S：1/13 位将不会清除，所以操作员必须在每个主要出错后清除 S：1/13 位。

（2）数学控制

置位 S：1/14 位，数学溢出选择位。如果你打算使用 32 位加和减，这个位会影响出现上溢出或下溢出时的保存结果。

在使用浮点数的数学操作期间，S：34/2 置位使数学标志无效。位 S：0/0 到位 S：0/3 和位 S：5/0 不会在进位、溢出、0、负数或者在溢出中的小错误发生时置位（S：0/0 在浮点运算中用作其他目的）。

（3）看门狗定时器

每个扫描循环开始时，SLC 500 启动看门狗定时器。如果在扫描循环结束前，定时器定时结束，PLC 将会出错（看第 11 章）。程序员可以给看门狗定时器输入一个以 10ms 为单位的时间，这个值保存在 S：3H 中，它是字 S：3的高字节（每个扫描循环结束后，SLC 500 会把完成扫描循环所用的时间写到 S：3L 中的低字节。）。

通过 S：33/13 选择扫描时间的时基，如果该位置位，时基将会从默认的 10ms 变化到 1ms。PLC 写入 S：22 和 S：23 中的平均和最大扫描时间也会受到影响。

（4）实时时钟

S：37 到 S：42 分别包含年、月、日、小时、分钟和秒的数值，它们在 PLC 运行时被更新（直到 65535 年，年的数值都是没有问题的!）。通过写 0 到这些字中，你可以让实时时钟计时无效。

（5）中断

使用处理器状态窗口，可以配置四种类型的中断（根据 SLC 500 的类型）。所有的这些将在第 11 章详细说明。

1）出错例行程序文件号可输入到 S：29 中；

2）可选择的定时中断（STI）文件号可以输入到 S：31 中，用来执行 STI 文件的时间间隔写入 S：30 中。通过 S：2/10（置位时是 1ms）来选择 STI 定时器的定时单元（1 或者 10ms），通过改变 STI 使能位（S：2/1）的默认状态使 STI 无效；

3）离散输入中断（DII）文件号可输入到 S：46 中。程序员输入插槽号和屏蔽位（分别为 S：47 和 S：48）来配置哪个输入模块和哪个触点是中断源，可以输入比较值和计数器预设值（分别为 S：49 和 S：50）来配置位传输类型和在中断发生前需要多少计数。如果改变 S：2/12 的默认状态，DII 中断将保持，直到这个位再次置位；

4）专用的 I/O 模块可以配置来生成 I/O 中断请求。在 I/O 配置过程中（将在下面讨论）输入响应 I/O 中断的程序文件号，但是程序员可以使用处理器状态屏幕使 I/O 中断无效（插槽 1 到 31），通过分别对 S：27/1 到 S：28/14 位清 0。这些位默认为置位。

（6）I/O 插槽使能

通过对 S：11/0 到 S：12/14 位复位，程序员（或者程序）可以使 31 个 I/O 模块中的任何一个 I/O 无效（插槽 1 到 31 和插槽 0 中的 CPU 模块）。位 S：11/0 控制插槽 0，位 S：11/1 控

制插槽 1，等等，这些位都默认为置位。当一个 I/O 插槽无效时，CPU 的输入映像表将不受输入模块变化的影响，而改变 CPU 的输出映像表，输出模块的状态也不受影响。

(7) 通信和消息控制

在这一节中，我们说明使用状态文件（S：）字配置选项。使用 SLC 500 进行通信将在第 13 章详细阐述。

置位 S：1/14 可以拒绝编程设备访问这个 PLC 的程序和数据，除非它已经有处理器文件的拷贝了。

你可以限制在 I/O 扫描循环中服务通信通道所花费的时间。如果它不受到限制，每次扫描循环中用来服务于通信通道的时间变化很大。

1) 每次通道 1 被服务时，置位 S：2/15 来限制 CPU 给通道 1 一个通信服务任务。

2) 置位 S：33/7 来限制 CPU 服务一个通道 1 的通信任务，即使 S：2/15 没有置位，也可以通过消息指令使其初始化。

3) 每次通道 0 被服务时，置位 S：33/5 来限制 CPU 给通道 0 一个通信服务任务。

4) 置位 S：33/6 来限制 CPU 服务一个通道 0 的通信任务，即使 S：33/5 没有置位，也可以通过消息指令使其初始化。

5) 置位 S：34/0 来禁止 CPU 在通道 1 的 DH＋连接和在通道 0 的 DH-485 连接之间传送信息。

6) 置位 S：34/5 来允许 CPU 在通道 1 的 DH＋连接和通道 0 的 DF1 连接之间传送信息。

7) 置位 S：34/1，并使用“DH＋ active”表，来使这个 CPU 保持一个 DH＋上哪一个站是激活的记录。

位 S：33/14 和 S：33/15 分别在通道 0 配置和控制 DTR 信号。

通过置位 S：2/8，通用接口文件（CIF）中的默认数据单元大小可以从默认的 16 位变到 8 位。使用 PLC-2 读/写类型的消息（MSG）指令可以自动地读/写 SLC 500 的 CIF（详细内容可以看第 13 章关于 CIF 文件的介绍），SLC 500 的数据文件 9 自动地用作 CIF 文件（因而聪明的程序员可以保留数据文件 9）。其他 PLC-2 兼容的设备也可以自动地访问 SLC 500 的 CIF 文件，一些设备需要 8 位的数据单元大小（特别是那些通过网桥或网关访问 SLC 500 的设备）。

一些保存在状态文件中的配置数据实际上通过使用通道配置屏幕来输入（在下面将解释），但是这些数据可以被用户程序覆盖。这些状态文件值包括：

1) S：15L（在字 S：15 的低字节）包括了节点地址（DH-485 的 1～31，DH＋的 1～63）。

2) S：15H（S：15 的高字节）包括 DH-485 网络的波特率代码号。

3) S：34/3，如果置位，导致 CPU 通过 DH＋网络传递全局状态字（S：99 中的内容）。S：34/4 位，如果置位，导致 CPU 接收来自其他 DH＋网络上 CPU 的全局状态字，拷贝它们到从 S：100 至 S：163 的地方（从节点 0 的 S：100，节点 1 的 S：101，等等。）。用户程序可写数据字到 S：99 中，并可从 S：100 到 S：163 读取数据。

4. 配置 SLC 500 通信通道

通过使用“Channel Configure”屏幕，对一些通信通道的配置选项进行选择。这些配置

参数保存在 CPU 模块中，但不一定要保存在状态文件中，当 SLC 500 程序下载到个人计算机后，这些参数也可以保存到各自的文件中。任何时候当把保存的程序从个人计算机上传到 SLC 时，通道的配置启动信息也被上传。记住当你改变一个节点的某些配置参数时（如波特率或者节点数），你也必须相应地改变其他节点或者编程器，或者和这个节点交换数据的操作员接口面板。

使用通道配置对两个通道（0 和 1）进行配置。任一通道可以配置成如下：

1）DH＋通道。必须给 CPU 输入节点地址（八进制的 1 到 77），并且波特率可以增加到 57.6 千波特。如果在通信系统中有超过一个 DH＋，就要输入链接号来确定这个通道连接到哪个 DH＋中。通过屏幕可以改变全局静态字的传输使能和接收使能状态位（S：34/3 和 S：34/4，将在下一节中讨论）。

2）DH-485 通道。也必须给 CPU 输入节点地址（从 1 到 31）。最大的节点地址可以从 31 开始减少，如果只有很少的控制器连接到 DH-485 网络上，这样可以减少通信时间。也可以把令牌保持因子增加到 4，每次轮到它“发言”时可允许 CPU 传送这个数字的消息。波特率可以从 19200 变化。

3）RS-232 C DF1 通道。允许 CPU 和一个或更多其他电脑（包括调制解调器和其他的 SLC 500 组件）上的串口进行双向通信，在 SLC 500 中的 RS-232C 串口通信协议的配置必须和其他的电脑协议相匹配，不同配置窗口的一些 DF1 选项是许可的（查看 SLC 500 用户手册和电脑的 RS-232C 协议）。

4）ASCII 通道。允许 CPU 传送和接收用户程序中 ASCII 码命令控制的 ASCII 信息。

5. 配置 SLC 500 智能 I/O

专用的 I/O 模块是 SLC 500 的智能 I/O 模块，每一个模块有自己的存储器和配置要求。为了配置专用 I/O 模块，在首次配置离线编程时，或者在离线编程选择“Processor Change”选项的时候，使用 Configure I/O 菜单中的“SPIO Configure”选项。在为 PLC 插槽选择一个专用 I/O 模块后，SPIO 的配置菜单允许你做如下的选择：

1）执行 ISR 程序文件来响应此插槽的一个来自模块的 I/O 中断；

2）在这个插槽中为专用 I/O 模块保留的输入和输出映像字的数量，以及在每次 I/O 扫描中包含有多少字；

3）这个模块的 M0 和 M1 数据字的数量，如果允许 M0/M1 监控，可以对它们监控（最开始的配置选择之一）；

4）专用 I/O 模块中 G 数据文件字的数目和那些 G 数据字的内容。G 数据字可以用来配置 I/O 模块，查看用户手册，了解专用 I/O 模块的 G 数据意义。

10.4.3 SIMENS S5 的第一次配置

S5

1. 连接 S5 CPU 和在线编程

选择在线编程（“编辑 PLC”），不需要配置。

2. 配置 S5 CPU

S5 CPU 配置数据以 16 位字的形式存储在 CPU 的系统数据（RS）存储器区域。可选型的配置软件使配置更容易，但不是必需的。使用数据窗口可以把数据输入到系统数据存储器中，或者使用包括功能块的用户程序把数据写到系统数据存储区域中。（只用 STEP 5 的功

能块程序能读或写系统数据存储器)。

下面的系统数据存储区域包含可修改的配置数据:

1) RS 96 的低字节包含看门狗定时器的设置。在一个不常见的长扫描循环使 PLC 出错前,这个数是以 10ms 为单位的数。可以把设定值改为 200ms,功能块中的指令如下:

```
L  KF + 20  ;  (20) 10-ms time units = 200 ms
T  RS 96
```

2) RS 97 控制 OB 13 作为定时中断服务例行程序而被调用的频率。RS 97 默认包含"10",并且单位是 10ms,因此对 OB 13 的默认设置是每 100ms 执行一次。默认定时中断是使能的。S7-115U CPU 在 RS 98 到 RS 100 中有时间值,分别通过 OB 12 到 OB 10 来控制循环中断。(查看第 11 章获取详细信息)

为了禁止定时中断和过程输入中断,可以在一个功能块程序中执行命令 IA。

3) RS 8 到 RS 10 控制一个实时时钟 (RTC)。这三个字包含两个 3 字节的指针:一个指向最多 44 个字节的时钟数据应该保存的位置,另一个指向 RTC 状态 (和控制) 字。RTC 时钟数据不需要与在 RTC 状态字有相同的存储区域。

3 字节的 RTC 数据指针包括:

- 一个代码,在 RS 8 高字节中,代表保存实时时钟数据的存储区域。十六进制的 44 代表一个数据块,45 代表标志存储器,49 代表输入映像 (PII) 存储器,51 代表输出映像 (PIQ) 存储器。
- 一个数,在 RS 8 低字节中,指出从 2 到 255 的数据块编号,或者指出在标志或 PII 或 PIQ 中的起始地址。记住,你需要从这个地址开始的 44 个字节的存储空间。
- 一个数,在 RS 9 高字节中,代表起始数据字地址 (当然,除非 RTC 数据在数据块中,否则忽略。)。这是包含了 44 个字节的时钟数据的 22 个数据字的低字。

3 字节的 RTC 状态字指针包括:

- 一个代码,在 RS 9 的低字节中,代表包含状态字的存储器区域。(和上面的 RTC 数据有同样的代码)。
- 一个数,在 RS 10 的高字节中,指出数据块 (2 到 255),或者指出包含状态字高字节的标志/PII/ PIQ 地址。(第二高位的标志/ PII/ PIQ 地址会包含状态字的低字节)。
- 一个数,在 RS 10 的低字节中,表明包含状态字的数据字 (除非状态字在数据块中,否则忽略。)。

如果你已经为实时时钟输入指针数据,你将要设置时钟。如果要设置时钟,输入以下的 RTC 数据到 RTC 时钟数据的 44 个字节中 (或者 22 字) (第一个字节叫做字节 0):

1) 实际时间,输入到从字节 8 到字节 15 的区域中。当你操作设置 RTC 时间的状态字位时,这个数据可以拷贝到从字节 0 到字节 7 的正在工作的实时时钟中。把下面的 BCD 码值 (使用 KC 常量) 输入到字节:

```
8   直到一个闰年的年数(KC 0到3)(这个数字只能内部使用,并且不能拷贝到字节0。)
9   星期数(KC 1到7)
10  天(KC 1到31)
11  月(KC 1到12)
12  年(KC 00到99)
13  小时(KC 0到23是24小时循环,KC 1到12是上午,KC 81到92是下午。)
```

14 分钟(KC 0到59)
15 秒(KC 0到59)

2) 快速装载间隔时间，输入到从字节 16 到字节 23 的区域。在你设置状态字位后，会引起 CPU 使用快速装载，CPU 在间隔时间中会重复地置位状态字的高字节的位 5（输入无效的数字使间隔时间取消。）。把下面的间隔时间输入到字节中：

16 (未使用)
17 星期数(KC 1到7)
18 天(KC 1到31)
19 月(KC 1到12)
20 (未使用)
21 小时(KC 0到23是24小时循环,KC 1到12是上午,KC 81到92是下午)
22 分钟(KC 0到59)
23 秒(KC 0到59)

3) 为操作小时计数器设置新的时间，输入到从字节 30 到字节 35 的区域中，这个值被拷贝到从字节 24 到字节 29 的正在工作的操作小时计数器中。正在工作的操作小时计数器跟踪 CPU 在运行模式的时间。输入下列时间到字节中：

30 (未使用)
31 秒(KC 1到59)
32 分钟(KC 1到59)
33 小时(KC 0到99)
34 100小时(KC 0到99)
35 10 000小时(KC 0到99)⊖

你也可以充分利用系统数据存储器指针中的第二个指针指向的状态字。你将要在配置实时时钟时使用的位包括：

1) 低字节位（标志/PII/PIQ 字节地址指向的下一个字节，或者数据字的低字节指向的下一个字节。）：

- 位 2，置位时，导致 CPU 从数据字节 8 到 15 拷贝实时时钟设置到正在工作的时钟字节 0 到 7。在成功完成拷贝后，S7 操作系统清除这个位。如果拷贝成功，但是操作系统发现错误数据，位 0 将被置位。
- 位 1，置位时，以 12 小时模式计算小时（1 到 12 和 81 到 92）。清 0 时，使用 24 小时循环（0 到 23）。
- 位 4，置位时，导致 PLC 更新实时时钟（字节 0 到 7）；甚至当 PLC 在停止模式，也可以更新。
- 位 5，置位时，导致 PLC 记录上一次离开运行模式的时间。时间将以如下的格式保存在数据字节 36 到 43 中：

36 (未使用)
37 星期数(KC 1到7)
38 天(KC 1到31)
39 月(KC 1到12)
40 年(KC 00到99)
41 小时(KC 0到23或 KC 1到12/81到92)

⊖ 是不是 Siemens 真的希望这些 PLC 之一能运行 115 年？

42 分钟(KC 0到59)
43 秒(KC 0到59)

2）高字节位（标志/PII/PIQ 字节地址指向的，或者指向数据字的高字节的。）：

- 位 2，置位时，导致 CPU 从数据字节 30 到 35 拷贝新的操作小时计数器的时间到正在工作的操作小时计数器的字节 24 到 29 中。在成功地完成拷贝后，S7 操作系统清除该位。如果拷贝成功，但操作系统发现数据有错误，位 0 将被置位。
- 位 1，置位时，使能操作小时计数器。清 0 时，操作小时不被计数。
- 位 6，置位时，导致 CPU 使用在数据字节 16 到 23 中的快速装载时间设置。S7 操作系统在拷贝成功后，把此位清 0。如果操作系统发现数据错误，位 4 将置位。

每次快速时间用完后，STEP 5 操作系统给相同字节的位 5 置位。可能你的程序包括一个响应例行程序，此例行程序在位 5 保持时执行，并且它将复位位 5。

3. 配置 S5 通信通道

在这一章中，将要描述允许 S5 PLC 经过串行端口或者经过通信处理器模块进行通信所需要的配置。通信在第 13 章中描述。

S5 CPU 的串行端口有时也叫做程序员端口，这是因为串行端口用来连接编程器。串行端口在点对点配置中可以用来连接两个 S5 PLC，或者在单一的 L1 网络中可以用来连接一个 L1 主站和最多 30 个 L1 从站。在这两种情况下，你必须向 CPU 系统的数据存储器区域输入网络地址和指针[⊖]。

1）如果你要连接到 PLC 的编程器可以通过 L1 网络连接到其他的 CPU 上，RS 57 的高字节可以包含一个表示编程器的节点数。

2）RS 57 的低字节必须包括一个惟一的节点数。如果 L1 网络正在使用，输入 L1 的从站数（1 到 30）。（S5 CPU 模块只能作为 L1 从站。L1 主站将在下面解释。）在点对点的连接中，一个 CPU 必须是节点 0，另一个 CPU 是节点 1。

3）RS 58 到 RS 63 必须包含指向一个接收协调字节、一个发送协调字节、一个发送信箱和一个收到信箱的位置数据。协调字节和信箱在数据块或标志存储区中。

- 接收协调字节：

RS 58,高字节　　数据区 ID
RS 58,低字节　　DB 或标志字节
RS 59,高字节　　数据字

- 发送协调字节：

RS 59,低字节　　数据区 ID
RS 60,高字节　　DB 或标志字节
RS 60,低字节　　数据字

- 发送邮箱：

RS 61,高字节　　数据区 ID
RS 61,低字节　　DB 或标志字节
RS 62,高字节　　数据字

- 接收邮箱：

⊖ 在某些 S5-115U CPU 中有预编程的 FB 239。FB 239 将提供给 FB 239 作为参数的配置数据拷贝到适当的系统数据存储器。

RS 62,低字节　　数据区 ID
RS 63,高字节　　DB 或标志字节
RS 63,低字节　　数据字

这里

■ 数据区域 ID 必须是：

● 如果保留的存储区在数据块中，D 的 ASCII 码（十六进制：44）；

● 如果保留的存储区在标志存储区域中，F 的 ASCII 码（十六进制：4D）。

■ DB 或者标志字节表示哪个数据块或者标志存储器在使用。输入：

● 如果数据区域的 ID 是 D，数据块号，从 2 到 255；

● 如果数据区域的 ID 是 F，标志字节号，从 1 到 255，表示数据的起始。

■ 数据字表示：

● 数据字在它的低字节中包含协调字节，或者如果数据区域 ID 是 D，它的低字节是邮箱数据开始的位置；

● 如果数据区域 ID 是 F，不相关。

在第 13 章将描述协调字节和邮箱，在 L1 网络上的 PLC 之间的数据交换将用到它们。

对于 S5-115U CPU 模块，它们的 CPU 串口中建有 ASCII 码驱动，它们的配置包括：

1）写数 1 到 RS 46 的高字节，为串口选择 ASCII 驱动器。

2）RS 48 到 RS 55 必须包含以下格式的数据（配置 L1 网络的数据格式是类似的，但不完全相同。）：

■ ASCII 码参数设置：

RS 48,高字节　　数据区 ID
RS 48,低字节　　DB 或标志字节
RS 49,高字节　　数据字

■ 发送邮箱：

RS 49,低字节　　数据区 ID
RS 50,高字节　　DB 或标志字节
RS 50,低字节　　数据字

■ 接收邮箱：

RS 51,高字节　　数据区 ID
RS 51,低字节　　DB 或标志字节
RS 52,高字节　　数据字

■ 发送协调字节：

RS 52,低字节　　数据区 ID
RS 53,高字节　　DB 或标志字节
RS 53,低字节　　数据字

■ 接收协调字节：

RS 54,高字节　　数据区 ID
RS 54,低字节　　DB 或标志字节
RS 55,高字节　　数据字

在配置 L1 网络的时候，数据区域 ID、DB 或者标志字节和数据字描述如下：

ASCII 码参数设置是一组 12 个字节的 ASCII 通信协议，如波特率，每个 ASCII 码字符

所用的位，打印页控制等等。查看你的用户手册获取更多的信息。邮箱和协调字节中的数据使用方式与 L1 邮箱或者协调字节中的数据使用方式是类似的，但不完全相同（查看第13 章）。

L1 网络必须有一个主节点，它通常是 S5-115U PLC 的框架中的 CP 530 通信处理器模块。S5-115U CPU 模块有数据处理功能块，它用来把配置参数写到 CP 模块中，也可以通过 CP 模块发送和接收数据。如果 CP 530 模块没有自己的非易失性存储器，如 EEPROM 模块，那么每次 PLC 进入运行模式，CP 模块的配置需要重新输入。在第 7 章已经介绍如何给数据处理功能块编程，但没有讨论怎样使用它们把配置数据写到通信处理器模块中。这些数据处理功能块包括：

1) FB 246，同步，在其他数据处理功能块前使用，用来为 CPU 模块和 CP 模块的数据交换规定最大的数据块大小，以及这两个模块之间的同步；

2) FB 244，发送，在 L1 网络操作之前，写配置数据到 CP 模块；在配置后，发送控制数据和消息数据到 CP 模块，或经过 L1 网络发送数据信息。在使用“发送”配置 L1 网络时，下面的参数很重要，它们包括：

A-NR，工作（job）号。输入如 KY 0,y，“y”是一个工作号，指出这个数据传输到 CP 模块的目的。对于下面每个执行特殊配置操作的工作号必须执行一次“发送”：

222 通过写新的 L1 系统 ID 参数设置（SYSID）到存储器的接口区域来配置 CP 530 模块。程序员输入到存储器的数据设置必须是 ASCII 编码数据，其中段落之间必须用 ASCII 码的回车符分开（即使在某一段落没有数据输入，在每段后回车符也是必需的）。段落以如下顺序排列：

- 四个可选择的标识段。头两个段落可以包含最多八个 ASCII 字符，第三段落最多 19 个字符，最后一个段落最多八个字符。
- 二个空白段（输入回车符代码）。
- 第 7 段必须被输入。包含一个两数字的编程器节点号（如果编程器用来通过 L1 网络访问从节点），然后一个“/”符。在“/”后没有输入，表明这个节点数是 L1 主节点，编号 0（如果你编程使 CP 530 作为 L1 从站，你可以输入 1 到 30 间的从站号。）。
- 五个空白段。
- 第十二字段必须包括接口号。默认的接口号是 1，但如果在 SYSID 数据中你输入一个不同的数，那你可以重新编程设置 CP 模块的接口号。下次使用功能块和这个模块通信时，必须输入新的接口号，如 SSNR 参数。如果你拷贝 CP 模块存储器的内容到 EEPROM，接口号可以永久性的重新配置。
- 第十三个段落可以包含一个代表“是”的“Y”或代表“否”的“N”，指出在电源恢复之后，CP 是否立刻自动恢复 L1 主站的操作（默认为 Y）。
- 第十四个段落可能包含给 L1 网络的一个新波特率（默认为 9600）。

43 通过接口存储器向模块中写入一个新的轮循列表来配置 CP 530。如果从站想要通过 L1 网络来进行传输，那么轮循列表必须有 64 个节点数组成（在存储器的连续字节）来确定从站被使用的次序（可以让一些节点输入多于一个的节点号从而使这些节点在每个轮循循环中获得更多的机会）。

44 通过接口存储器向模块中写入一个新的中断列表来配置 CP 530 模块。中断列表有

30 个节点数组成（在存储器的连续字节），标出发送高优先级数据的节点，并在两个节点的请求同时发生的时候，确定每个节点的优先级水平。（每个节点号只能输入一次）。

42 写一个模式控制字节来启动或停止 L1 网络。如果字节的位 0 置位，L1 网络将停止；如果位 1 置位，L1 网络将会启动运行。当 L1 网络在停止模式，配置数据仅可以写入一个 CP 530 模块中；但是在它被配置之后，工作 42 可以用来使 L1 网络进入运行模式。

3）FB 247，控制，读取 CP 530 模块的状态。

4）FB 245，接收，读取 CP 530 模块状态，但是也能读取 PLC 通过 CP 530 收到的消息。

5）FB 248，复位，取消任何数据处理功能块的工作请求。

一些 CP 模块使用内部处理器通信标志控制 CP 模块和 CPU 模块之间的通信。在硬件配置期间，CP 模块的 DIP 开关置位，表明它们应该使用 CPU 内部处理器通信标志存储区域的哪一区域与 CPU 模块进行消息交换。（内部处理器通信标志保持在一个 0 到 255 的特殊存储区域中。）配置 CPU，使其使用内部处理器通信标志，写配置数据到数据块 1 中（这就是为什么我们建议只使用数据块地址的 2 到 255 作其他用处的原因。）数据块 1 中的配置数据告诉 CPU 哪一个字节包含输入位（CP 模块可以写入其中），哪一块包含输出位（CP 模块仅可以读）。在 PLC 处于运行模式时，你可以创建和编辑 DB1，但在 PLC 下次运行前，PLC 不会承认修改过的配置。为了配置内部处理器通信标志的使用，可以创建一个数据块 1（DB1）包括：

1）一个由 ASCII 字符组成的头：

KS= ‘MA’ in DW0

KS= ‘SK’ in DW1

KS= ‘01’ in DW2

2）十六进制代码 CE00 代表输入标志字节列表的开始，紧跟着的是用作输入字节的内部处理器标志存储器的字节地址，通过它 CPU 能读取 CP 模块的状态，例如：

KH= CE00 in DW3

KF= +10 in DW4

Kl = +11 in DW5

KF= +20 in DW6

3）十六进制代码 CA00 代表输出标志字节号列表的起始地址，紧跟着的是用作输出字节的内部处理器标志存储器的字节地址，通过它 CPU 能控制 CP 模块，例如：

KH= CA00 in DW7

KF= +15 in DW8

KF= +30 in DW9

KF= +31 in DW10

4）十六进制代码 EEEE 指出内部标志存储字节分配列表的结束地址，例如：

KH= EEEE in DW11

4. 配置 S5 智能 I/O

在这一节中，我们简要地介绍产生中断的数字输入模块的配置。对更加复杂的模块配置要求，可以参考用户手册对那些模块的说明。

带有 8 个具有中断能力触点的数字输入模块可以通过传送 16 位数到此模块中来进行配置。在启动块（OB 21 或 OB 22）中，使用传送指令编程和使用直接访问地址，以便在 PLC 启动后，模块就可以工作。高 8 位用来使 8 个触点中的 1 个或多个打开中断（为 1 时打开触点），低 8 位的每一位使一个打开中断的触点在电压升高（如果配置位＝0）或者降低的时候，使触点产生中断。在下面的程序中，插槽 1 中的数字输入模块的高 6 个触点（触点 1.2 到 1.7）可以产生中断，它们中的 5 个触点在电压上升的时候产生中断信号，另一个触点（1.2）在电压下降的时候产生中断。

```
L  KM  11111100 00000100
T  PW  1
```

第 11 章有中断的更详细说明。

S7 10.4.4 SIMENS S7 第一次的配置

1. 安装 STEP 7 编程软件

STEP 7 编程软件的安装程序将在没有用户的任何帮忙下（除了换软盘），把编程软件装载到个人计算机。但是用户必须在安装之前或安装过程中安装一个授权码。最容易的方法是在安装时使用带有授权的磁盘，这样当安装程序需要它的时候，你插入它就可以了。

STEP 7 的编程软件在你的个人电脑硬盘上的一个隐藏路径（C：\ AX NF ZZ）中以隐藏文件形式存储授权文件，如果它稍后不能找到授权，将不工作，并建立一个缺陷簇。不要删除这个缺陷簇。

在新的操作系统的安装过程中，在硬盘格式化期间，在压缩磁盘空间过程中，或当简单地删除包含缺陷簇的文件时，授权可能丢失。在做这些动作之前，用户应该再插入授权磁盘，并且运行 Authors 程序来把在硬盘中的授权移到软驱中，直到潜在丢失授权的危险消失。通过使用 Authors 程序，授权可以重新装载到硬盘中。

2. 设置 STEP 7 编程器和 PLC 接口

在安装 STEP 7 软件过程中，Windows 95/NT 控制面板自动地配置默认的编程器接口参数，但是你必须手动完成这个过程。如果你使用个人计算机的串口和 MPI 接口电缆来直接连接 S7 CPU 的 MPI 端口，选择 WINDOW 的“我的电脑”，然后进入“控制面板”，在那里你会发现 STEP 7 的“SETTING PG/PC INTERFACE”（设置 PG/PC 接口）。

1）选择“设置 PG/PC 接口”。

2）设定应用的访问点到 S7ONLINE。

3）选择“INSTALL”（安装）并且选择接口。（经过 PC/MPI 适配电缆到编程端口。）

4）选择“PROPERTIES”（属性），然后为 MPI 网络的编程器选择“local station address”（局域站地址）（与任何其他编程器的局域站地址不同），如果还有其他编程器，检查“Not the Only Active Master”（非惟一激活的主站）。检测并确定 COM 端口号是正确的。

如果你通过计算机上的网络接口卡（而不是通过个人计算机的串口）把个人计算机连接到 PLC 上，安装 STEP 7 的默认设置包括中断（IRQ）号，或其他个人计算机的组件使用的地址。为了找到它，可以通过选择 Windows 的“控制面板”，然后选择“系统”，然后进入“设备管理器”，然后进入“电脑”，然后打开中断和内存选项来显示中断（IRQ）和接口卡地址的分配。查看你的接口卡和其他电脑组件使用的中断和默认存储器地址之间的冲突，如

果发现，记录可用的中断和存储器空间，返回到控制面板，执行上面四个步骤，并把这一步作为第 5 步，纠正发现的任何冲突。

5）选择接口卡的型号并且显示它的属性。修改需要更正的默认值。你可以使用中断 5、10、11、12、15，或有时是 9。如果没有自动分配存储器空间地址，使用 F00000H 到 F4004FH，或 F80000H 到 FC004FH。

3. 创建 STEP 7 工程

在你能配置或编程 PLC 之前，必须创建一个带有站点的工程[1]。在创建工程过程中，编程器的存储空间分配给你的 PLC 系统中将要创建的和将要传输到 PLC 的工程数据。工程数据包括通常被程序员认为是程序的一部分，但不一定要写到 S7 CPU 中的数据。举例来说，符号名是在编程过程中创建的并保存在工程中，但是不会下载到当前的 S7 CPU 中。如果通过相同的编程器，你可以重新连接到 CPU 上，也可以观察符号名，但是如果使用不同的编程器连接到 CPU 程序，你只可以看到绝对地址。[2]

工程结构帮助保持数据库的逻辑顺序，数据库中包括配置、编程和内部连接的 PLC 系统中所有 PLC 的数据存储器分配。每个 PLC 系统应该创建它自己的工程。图 10-8 显示了一个工程例子（名字叫 Project _ A），它如何构建包含与 PLC 系统中的所有 CPU 相关的数据（例如符号表），选择 PLC 的网络（例如全局数据表），或者在系统中指定 CPU（例如配置和 Station _ AA 的参数系统数据）。图 10-8 中的树型结构和 STEP 7 编程器中显示的有一点不同，STEP 7 编程器使用旁边的窗口显示一些组件。

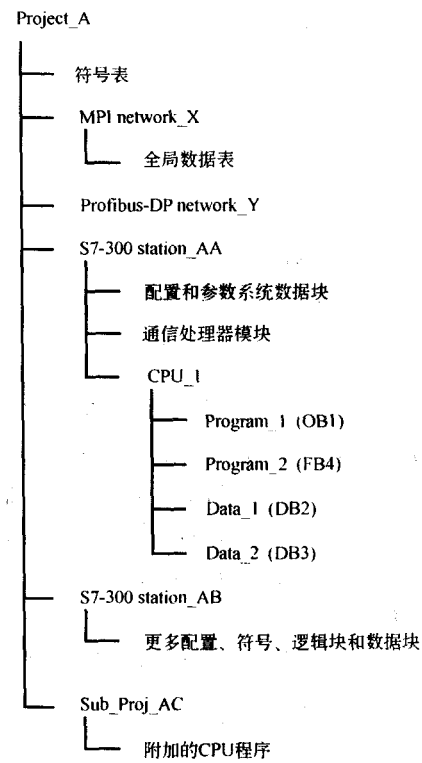


图 10-8 小型 STEP 7 工程，显示了一些组件

创建一个工程：

1）从 Windows 的桌面通过选择“SIMATIC Manager”来启动 STEP 7 编程软件。第一个屏幕会告诉你选择 CPU、MPI 地址和工程中一个站的逻辑块。通过你安装的 PLC，选择 CPU 类型，来回答上述的问题，为逻辑块选择 OB1，为工程输入名字，然后选择“Make”。（下次启动 SIMATIC manager 的时候，可以取消打开菜单，然后通过“File-Open-Project”进入工程。）

2）如果需要创建一个工程，并在没有创建工程的时候，从步骤 1 的对话框中退出来，选择“File”，然后选择“New”，然后选择“Project”，然后输入合适的“Name”。

---

[1] 你可以在创建工程之前在程序库里创建程序模板（特别是功能块）和数据块模板（用户数据类型）。

[2] Siemens 说未来的 S7 PLC 将接受符号名作为下载程序的一部分。PLCOpen 指定有限的自然文件交换格式，他们希望这些格式可以被扩展到允许所有的 PLC 编程软件创建可以在任何 PLC 中使用的程序。PLCOpen 的成员仍在争论符号名等是否应该作为 PLC 程序的一部分。

在创建工程后，可以创建和编辑程序和网络或者在网路上的单一站点的配置文件，并分配可以在工程的任意程序中使用的符号名。

4. 配置连接到网络上的 PLC 系统的一个 S7 站

对于第一次安装，在你写任何程序之前，先为 PLC 系统创建配置和参数系统数据块。(虽然你可以先写程序)。

配置数据包括系统中的站点描述和连接站点的通信网络描述。如果允许 SIMATIC Manager 创建一个工程，在软件启动后，跳到下面的步骤 2，创建配置文件：

1) 从 SIMATIC Manager 的顶部菜单中，选择 "File-Open-Project"，然后选择为这个 PLC 系统创建的工程。窗口将一分为二，如图 10-9 所示。

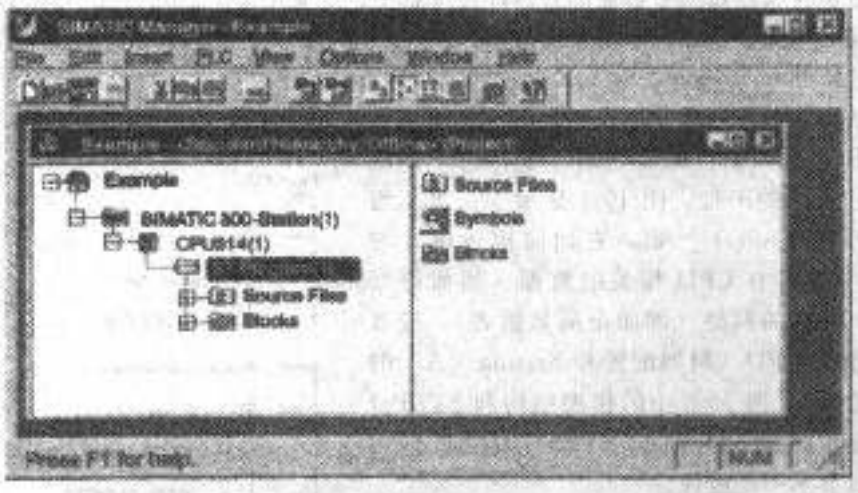


图 10-9 典型的 STEP 7 工程窗口

左边的窗口显示代表保存在你的工程中的数据项的内部图标。右边的屏幕将显示你在左边的窗口中点击的图标的组成条目。

2) 如果为通信目的而使 PLC 内部连接，最好现在就配置 PLC 子网，以便在下一步中，配置站点的通信。当你创建一个工程后，一个 MPI 子网就自动地插入，如果你不需要更多的子网，可以跳过第一部分。

- 点击工程名字，然后选择 "Insert Subnet"，选择内接类型：MPI，Profibus，工业以太网，或只是简单的点对点网络，为你的 MPI 或其他网络选择一个最高的地址。(为 MPI 子网选择 15、31、63 或 126；默认值是 15)。为 PLC 项目中的每一个互联的子网重复以上步骤。
- 通过选择子网，给每个子网（包括自动插入的 MPI 子网）输入配置数据，然后选择 "Edit-Object Properties"。出现对话框，输入子网的名字、传输速率和最高的节点地址。每个子网都这样做。

3) 现在输入描述系统中的每个 PLC 的配置数据。如果你已经为一个站点选择了 CPU 型号，就不需要另外的站点，跳到下面的第二部分。

- 选择 "Insert"，然后选择 "Station"，然后选择要插入站的类型（S7-300，S7-400，程序员，等等。），并为要配置的站输入合适的名字。尽管每个站点可以包括几个智

能 I/O 模块（IM、CP 或者 FM 模块），但是在工程中每个站[⊖]中通常只有一个 CPU 模块。

- 通过点击在左边窗口的新站点图标，选择“Hardware”，然后双击右边窗口的硬件，弹出硬件配置表窗口。
- 在硬件配置窗口中的组件反映了真实世界中的 PLC 系统。选择“View”，然后选择“Catalog”（如果目录没有自动地出现），一个新的窗口显示列出标准的 Siemens PLC 组件。
- S7-300 站点硬件配置将有一个引导项；但对于 S7-400，找一个中心框架或和你系统中相应的一个，然后把它拖到配置表窗口中。
- 找到实际系统中包括的电源、CPU 和其他一些模块，然后一次一个地把它们拖到配置表窗口中合适的插槽处。（记住：插槽 2 仅供 CPU 使用，插槽 3 仅供接口模块使用。）
- 如果在系统中包括扩展框架：
  - 为每个附加的框架重复框架和模块的选择。当然每个模块必须有一个接口模块（IM）。

如果站点包括 S7-314 IFM 模块，你就不能把一个 I/O 模块插入到框架 3 的插槽 11 中，因为集成到 CPU 中的数字和模拟 I/O 节点安排的地址是 124 到 129[⊜]，地址正常地会应用到框架 3 的插槽 11 中。

  - （仅用于 S7-400）通过双击发送 IM 来连接 IM 模块，然后选择“Connection Tab”，选择扩展框架并点击发送 IM 的“connection buttun”。每个扩展框架都这样做。
- 如果系统中包括 Profibus-DP 主站和从站：
  - 当 DP 主站插到配置表中后，“Properties”对话框打开。选择这个模块要连接的子网，输入一个 Profibus 节点地址。
  - 从目录中拖动 DP 从站模块到硬件配置窗口中的连接图标。“Properties”对话框打开，在对话框中，可以输入 Profibus 节点地址和其他的 Profibus 参数。（查看第 13 章获得更多的详细资料。）
  - 对于代表 I/O 框架接口的 DP 从站，一个新的框架插入到硬件配置窗口。对于像 DP 从站的 AS-i 控制器来说，可以允许选择并将 AS-i 从站插到 AS-i 系统中。

4）为这个站中的每个配置模块输入配置参数。通过双击硬件配置表中的可配置模块，选择一个可配置模块（例如，CPU、CP 等等）。每个模块显示一个“Properties”对话框，输入想要的配置参数。如果模块有一个时钟，选择“PLC-Set Time and Date”来设置。在这章的以下几节中，将提供关于 S7 CPU 和其他模块配置的更详细信息。

5）选择“Station>Save and Compile”。编译选项使 STEP 7 软件将配置信息翻译给系统数据块（SDB）。如果 SDB 在程序下载到 CPU 模块时存在，它们将被传送到处理器中，

⊖ S7-400 PLC 在一个中心框架中可以有超过一个的 CPU。这样的系统叫做多处理器系统，本书没有覆盖这部分内容。

⊜ 20 个数字输出：Q124.0 到 Q126.3；16 个数字输入：1124.0 到 1125.7；4 个模拟输入通道：IW128 到 IW135；一个模拟输出通道：QW128。

并可以改变 CPU 和它的可配置 I/O 模块的配置。“Station>Save option”选项不是用来创建系统数据块，应该在编辑程序而不要改变系统的配置时使用。

6）检查确定 SDB 已经创建。（如果输入的数据有明显错误，STEP 7 将不会创建 SDB。）选择“Station-Consistency Check”，STEP 7 将告诉你创建 SDB 失败的原因。

7）使 PLC 准备接收新的配置数据：使用本章 S7 硬件安装节中的按键方法，执行存储器复位，或者通过以下的软件方法：

- 将 PLC 切换到停止或者运行状态，然后
- 选择“File-Open-Accessible Nodes”，输入 PLC 的节点数，如果 CPU 没有配置，节点数是 2。（在配置过程中，分配新的节点数；记录给每个 CPU 配置的节点数。）
- 选择“PLC-Operating Mode..”，并点击“STOP”；
- 选择“PLC-Clear/Reset”。（在 PLC 复位存储器时，注意 STOP LED 灯的闪烁。）

8）（仍然在硬件配置窗口）选择“PLC-Download-To Module..”，然后让对话框帮助确认选择。

在创建或者修改模块配置的时候，即使处于在线编程，新的配置数据也不要写到 CPU 中。它必须先编译，然后下载。在下一次 PLC 进入运行状态时，新的配置参数才会生效。

9）退出硬件配置窗口，保存最终的工程到编程器的存储器中，使用“File-Save As”。输入的名字必须有扩展名 S7P，否则数据不能被正确地保存。（在一定的时间间隔里，有经验的个人电脑用户会保存他们的工作）。

STEP 7 编程语言包括一些标准函数（SFC），这些函数可以在初始化配置完成后，用来修改可编程 I/O 模块的配置信息。它们不会引起 CPU 存储器中系统数据块（SDB）的改变，因而在重启 PLC 后总是会重新保存原始配置。改变配置的 SFC 将在稍后的一节中讨论。

5. 配置 S7 处理器

双击站点的硬件配置表中的 S7 CPU 模块（如上面步骤 4 所示），将出现包含一些带制表符页的对话框。制表符允许选择寄存器类型，可以为寄存器类型输入配置参数。CPU 的配置寄存器在下面将描述。

（1）通用寄存器

从屏幕中选择 MPI 按键，然后为 CPU 选择一个新的 MPI 地址（从 2 到 126，默认为 2），MPI 子网（CPU 将可以通过 MPI 子网通信），波特率（默认：17.5 千波特，不可以改变）。

Siemens 建议把所有 CPU 的 MPI 地址设置为 2。不要分配地址 0、1 或者 2，保留他们给你不得不连接改变位置的 PG、OP 或者 CPU 单元（分别地）到 MPI 网络上时节点号 0、1、2 是那些设备的工厂设置的地址。在 MPI 网络上的所有节点必须有不同的 MPI 地址，所有的节点必须有同样的最高级的 MPI 地址输入。

必须激活“This Node Is Connected to the Selected Subnet”工具箱。

如果站点是 S7-300 型的，并且它包含附加的需要 MPI 地址的通信处理器（CP）模块或者功能模块（FM），那在配置 CPU 模块前，必须做站点硬件选择，这样在输入 CPU 的 MPI 地址时，可以自动地配置给紧接着的 CPU 的 MPI 地址的下一个 MPI 地址。举例来说，如果站点包括一个 CPU 和两个 CP 模块，配置地址 12 给 CPU，其他两个插槽中低位的 CP 模块可以自动地分配 MPI 地址 13，地址 14 分配给另一个 CP 模块。（因此不要配置 MPI 地址 13 或者 14 给其他 CPU。）

(2) 启动寄存器

选择 CPU 每次重新启动（通过将开关打到运行模式）是否必须测试硬件（默认为非）。S7-315-DP CPU 模块可以配置成拒绝进入运行模式，如果检测到的 Profibus-DP 网络配置不同于软件配置。

可以设置一定量的时间作为时间限制，在此时间中 CPU 模块从可配置的模块中等待一个完成信号，或者把参数写到可配置模块中。(站点配置参数只有在 CPU 模块中是保持的，因而每次在 PLC 开关进入运行模式时，CPU 一定会自动地重新写必要的配置参数到所有的 I/O 模块中。)

(3) 循环/时钟存储寄存器

选择最大循环时间（在这本书的其他地方叫做看门狗定时器时间，默认值：150ms），而在一些 CPU 中，选择最小周期时间。如果任何扫描循环超过最大循环时间，PLC 将会出错。如果任何循环的时间小于最小扫描时间，PLC 将在下一个周期启动前延迟。S7-400 CPU 允许 CPU 等待最小扫描循环结束的空闲时候执行 OB90 中的后台程序。

可以为一个时钟存储器指定一个存储器地址（默认值：没有）。如果你要使用时钟存储器，选择一个数据存储器字节（M0 到 M255）。PLC 将以如下的速率操作那个地址的 8 个位：

位：	7	6	5	4	3	2	1	0
速率：(s)	2.0	1.6	1.0	0.9	0.5	0.4	0.2	0.1

也可以指定经过 MPI 网络的通信时间不超过整个扫描时间的百分比（默认：百分之二十）。一个低的百分比将保证更多的重复扫描时间，但会降低 PLC 系统的站点间传输数据的速度。

也可以限制花费在测试 CPU 存储器的扫描时间的百分比（默认值：零百分比，表示没有测试）。

(4) 保持型存储器区域寄存器

选择哪个地址可以保存在非易失性 RAM 存储器中（NVRAM），以及 PLC 开关从停止到运行模式时，或者电源掉电后恢复时，哪个地址不被清除。如果 PLC 有后备电池，所有的数据块将会保持，而不管你如何配置这个 CPU 的寄存器。PLC 重启时，无论输入的配置是什么，存储器卡或者 EPROM 存储器中的数据将被拷贝到 RAM 中。

根据 CPU 的型号，可以配置下面容量的保持型存储器：

1) 从 M0 到最高 M255 的位存储器（默认值：M0 到 M15）。

2) 从 T0 到 T127 的定时器（默认值：没有）。

3) 从 C0 到 C64 的计数器（默认值：C0 到 C7）。

4) (仅对 S7-300) 最多八个数据块，DB1 到 127（默认值：DB1）。保存每一个首字节的地址（DBB 0 到 8191）和数据字节数（0 到 4096）。

(5) 中断寄存器

仅对 S7-400 CPU，通过使用这个配置选项，改变中断的优先级。

(6) 时间日期寄存器

对 S7 CPU 提供时间日期中断（在第 11 章将作解释），可以在这里激活中断，设置启动日期和时间以及它的间隔时间（默认值：禁止）。

(7) 周期性中断寄存器

给 S7 CPU 提供周期性中断（在第 11 章中解释），可以通过设置执行间隔时间（默认值：100ms）。改变执行间隔时间到 0ms，以禁止周期性中断。

(8) 诊断/时钟寄存器

选择 CPU 在记录其他检测到的出错（默认值：没有）之外，是否应该在诊断缓存中记录 PLC 事件的扩展历史（如 OB 调用）。配置 I/O 模块时（将在后面说明），可以包括指定是否可以传送出错事件消息给 CPU。选择是否每次 CPU 进入停止模式时，让 CPU 传送最后的诊断缓存的拷贝到另一个 MPI 模式中去。诊断缓存输入表明为什么 PLC 会进入停止工作模式（默认值：是）。

选择怎样使 CPU 的时钟和其他时钟同步（默认：没有同步）。可以选择 CPU 作为同步主站或从站，可以选择同步执行的间隔时间。经过 S7-400 底板（也叫做 K-bus）的同步现在是可能的（例如，CP 模块连接到 Profibus 或者以太网中），通过 MPI 端口的同步可能在将来的 S7 PLC 中使用。在发现 CPU 的时钟展示多少错误后，可以输入一个更正因子来帮助保持同步间隔之间的同步。

(9) 集成功能寄存器

如果 PLC 有一个 IFM 型 CPU（I/O 嵌入到 CPU 模块中），可以配置一些数字输入，如硬件中断源（在第 11 章中将解释）。输入可以配置如下：

1) 在 S7-312 IFM 中从 I 124.6 到 I 125.1

2) 在 S7-314 IFM 中从 I 126.0 到 I 126.3

选择是否把数字输入作为中断输入源（来调用 OB40），或者作为高速计数器，频率计，并行 A/B 计数器，或者位置传感器（例如，编码器）输入信号源。如果选择输入中断使用，可以分别地使能每 4 个输入位（默认值：禁止），并选择当信号上升或者下降的时候，是否产生中断。（当调用 OB 40 来响应输入中断时，STEP 7 操作系统传递一些参数给 OB 40，包括 OB40 _ POINT _ ADDR，它是一个 32 位字，其中低 4 位表示哪个输入位初始化中断）。其他中断选项将在第 11 章中讨论。

(10) 保护寄存器

最终，Siemens 计划允许用户设置口令，其他用户不能改变，甚至不能监视 CPU 的存储器内容。

在完成输入后，记住返回到步骤 5 的配置过程来保存、编译和下载这些配置参数。

6. 配置其他 S7 可编程模块

在站点硬件配置过程中，双击硬件配置表行，它里面包含可编程 I/O 模块的规格说明，适合该模块的对话框将出现。当使用者购买了功能模块或者通信处理器（CP），所带的软件会变成安装的 STEP 7 编程软件的一部分，软件中包含必要的配置程序。

当可编程 I/O 模块的配置数据从默认值改变并被编译后，将放入系统数据块（SDB）的两个记录中（记录 0 和 1）。每个可编程模块[⊖]有一个独立的 SDB 记录集。CPU 重启时，把 SDB 配置数据写到可编程模块中（如果程序员改变了默认配置）。记录 0 中的配置通常只可

⊖ 你的用户手册会告诉你每一个 SDB 用作什么目的，以及描述这些记录的格式。只有在你要编写用户程序来更改配置时才需要知道这些事情。我们将在后面讨论。

以用来使能或禁止模块产生编程错误消息，并传送给 CPU 模块的诊断缓存。记录 1 包含另外一些配置，包括可检测的出错类型定义，这些错误将产生可编程消息并产生诊断中断使 CPU 模块立即调用 OB 82。(在第 11 章中将讲述中断响应)。即使记录 0 禁止传送信息，一些可检测的硬件出错类型（非编程错误）将仍然会导致消息和诊断中断传送给 CPU。

用户程序可包含系统函数（SFC 55、SFC 56 或 SFC 57）来动态地改变模块的配置，在这章中后面会见到。执行这些 SFC 中的一个，不需要改变 CPU 中的 SDB，所以重新启动 PLC 将恢复模块配置到原始状态。记录 0 的模块配置只可以通过 S7-400 中的 SFC 来改变，但记录 1 的配置可以动态地改变。

除了已经描述过的诊断响应特性外，其他配置设置可以通过使用可编程 I/O 模块的对话框输入。根据模块的类型，可能有以下的选项：(如果没有其他指定，下面所有的配置选项成为 I/O 模块记录 1 的一部分。)

1) 数字输入模块被配置用来延迟输入状态的改变，从 0.5ms 到 20ms，这个配置成为记录 0 的一部分。数字输入模块可以被配置用来响应数字输入状态的变化，从而产生输入中断信号（引起 CPU 调用 OB 40)。(在第 11 章中讲述中断响应。)

2) 用“替代”输出值配置数字输出模块，数字输出模块可以被配置用来保持当前输出状态，或者输出替代状态。很少在第一次配置就配置一个模块保持或者替代输出值。这些配置参数可以被用户程序改变，以响应检测到的环境条件。

3) 模拟输入模块给每个通道配置一个可选择的类型和模拟输入信号的范围（如果模块没有硬件配置)。当输入信号超过最大限制或低于最小限制时，模拟输入模块还可以被配置用来产生一个输入中断信号（需要 CPU 调用 OB 40)。限制值必须在这里输入。

4) 模拟输出模块可以被配置用来给每个通道输出指定类型和输出信号的范围（如果模块中没有硬件配置)。模拟输出模块可以用一个替代输出值配置（每个通道)，模拟输出模块可以配置成保持最后的输出信号，或者配置成当 PLC 进入停止模式后，输出替代值（通常是 0 值)。

在完成输入后，记住返回到步骤 5 对配置过程保存，编译和下载这些配置参数。

7. 配置 S7 通信

在这一节中，将对配置过程中的步骤作一个简要概述。对 S7 PLC 通信的配置将在第 13 章详细地介绍。首先，需要确定每个 S7 CPU 如何与其他节点通信，需要经过什么类型的网络进行通信。

可以建立全局数据循环以便 CPU 存储器中的数据可以自动地通过 MPI 网络拷贝到其他少量的 S7 节点的存储器中。配置 S7 CPU 以便参与 MPI 网络中的全局数据循环：

1) 配置一个工程使其包括至少两个站点，它们被配置成连接到 MPI 网络中。

2) 选择 MPI 子网，然后选择“Options”，然后选择“Define Global Data”，一个 MPI 子网的配置全局数据配置表将打开。

3) 选择站点来参与全局数据循环。

4) 给每个站点输入地址，用来提供给全局数据循环通信。

5) 为了和其他 S7 CPU 交换数据，创建其他全局数据循环。

6) 选择“GD Table Compile”创建来自全局数据配置中的系统数据块（SDB)。如果正确地输入数据，当成功地完成这个阶段 1 的编译步骤后，软件将会报告。

7) 可以选择性地设置一个扫描速率或者创建一个 GD 状态字，然后再编译一次来执行

阶段 2 编译。

8）保存你的工作。SDB 将被下载到全局数据循环的站点中，然后可以下载程序到 PLC 中。

用户程序的标准函数（SFC）可以用来传送数据到 MPI 网络上的其他节点中，或者从其他节点获得数据（参考第 13 章）。

如果初始化通信的节点已经配置用来连接到其他节点上，如上描述，S7-400 PLC 通过使用系统功能块（SFB），可以在 MPI、Profibus-DP、或者工业以太网的节点之间进行平等的交换数据。配置 S7-400 PLC 到其他控制连接上：

1）在工程中创建和配置 MPI，Profibus，工业以太网，或者点对点子网。对于子网至少要配置两个可以通信的模块。如果已经按以上说明操作，这个工作已经完成。

2）选择想要连接到其他节点上的节点，选择“Connections”，然后选择“Eidt-Open Object”，一个配置连接窗口将打开。

3）选择“Insert-Connection”，一个新的对话框将打开。

4）选择一个伙伴站点和模块来建立连接，为这个连接选择通信类型。如果局域和伙伴都是 S7 PLC，选择 S7 单一站点，或者根据局域和伙伴之间的连接选择另一个类型。

5）选择“Edit-Object Properties”，另一个对话框将打开。在启动后，如果想要这个节点建立起通信连接，选择“Active”[⊖]。想要这个 PLC 每次在模式改变时通知它的伙伴（停止、运行、出错），选择“Send Operating Mode Message”。也可以为单向通信选择一个通信方向，（对于双向通信，两个控制器要配置成一个连接）。

6）注意局域 ID 号和可能分配的伙伴 ID 号。这些 ID 号需要作为用户程序中用于通信来调用的 SFC 或 SFB 的参数。

7）下载连接表到节点。

8）连接表不可以从 CPU 上载，所以在每次修改后，保存配置连接表到编程器中，否则你不能再次看到这个连接。

Profibus-DP 网络主站有一块存储区域，可以为每个 DP 从站保留最多 4 个输入字节和 4 个输出字节。Profibus-DP 网络自动地在这些存储空间和适当的 DP 从站的存储器之间拷贝数据，就像从站在 DP 主站框架系统的插槽中一样。配置 Profibus-DP 通信：

1）在工程中创建和配置 Profibus 子网。

2）在硬件配置屏幕上配置每一个有 DP 能力的模块时，选择 Profibus 子网，并激活“This Node Is Connected to the Selected Network”框。

- 配置 DP 主站时，使用对话框设置最多 1024 字节的 Profibus 主站 I/O 存储空间，通过它们主站可以读或写每一个从站的 I/O 地址。可以选择性地为每个从站建立一个诊断地址，从站来的一块诊断数据将保存在那里。
- 对于 DP 网络上没有 DIP 开关选择地址范围的 DP 从站，你必须使用对话框来配置特殊的从站存储空间来参与 Profibus 通信。智能从站可以设置一个诊断地址，从站可以在这个地址保留一些数据来跟踪与主站连接的 Profibus 的状态。

⊖ 你只能为 S7 PLC 配置动态通信。对于动态连接，两个节点都不会自动设置，但可以通过用户程序中的指令这样做。

Profibus-DP 主站可以在用户程序中使用 SFC 14 “DPRD _ DAT” 或者 SFC 15 “DPWR _ DA”来和 DP 从站交换最多 122 个字节的数据。配置连接必须按上面的介绍来设置。

在完成输入后，记住要返回到步骤 5 的配置过程来保存、编译和下载配置参数。

8. S7 符号表

下载其他工程组件时，符号表不用写到 CPU 中。符号表仍然在编程器中，允许程序员和操作员使用符号名而不是绝对地址。在第 5 章中已经讲述怎样创建和编辑符号表。在第一次配置 PLC 系统的过程中，程序员希望把 Siemens 的标准符号表导入工程中。标准符号文件，SYMBOL. SDF 可以从 C: \STEP7 _ V2 \ S7DATA \ SYMBOL 导入。

9. S7 逻辑块和数据块

组织块（OB）、函数（FC 和 SFC）和功能块（FD 和 SFB）是逻辑块。逻辑块包含程序算法和变量声明表。逻辑块可以下载到 CPU 中。数据块（DB、DI 和 SDB）也可以下载到 CPU 中。变量声明表和数据块已经在第 5 章和第 9 章中描述。逻辑块中使用的编程元素已经在前八章中阐述。

1）选择工程中的一个站点的 CPU，然后选择 “Insert”，然后选择 “S7 Block”，然后选择块类型（OB、FB、FC 或者 DB）。

2）如果你已经选择了逻辑块类型（OB、FC 或 FC），选择你希望使用的编程语言（梯形图、STL、FBD 或你付费购买的其他语言）。

10.4.5 OMRON CQM1 的第一次配置

CQM1

1. 配置 CQM1 处理器

为了配置 CQM1，必须写数据到从 DM 6600 到 DM 6655 地址的数据存储器中。这些存储器只可以通过编程器而不是程序写入。这些区域的配置数据在不同的情况下生效：

1）当 PLC 是在编程或监控模式，DM 6645 到 DM6655 立刻生效并且可以改变。（除了可以用编程器来显示存储器内容和程序状态，监控模式和运行模式差不多）。

2）当 PLC 切换到运行或监控模式时，DM 6615 到 DM 6644 生效，并且只有当 PLC 在编程模式时才可以改变。

3）当 PLC 电源关闭，然后接通，DM 6600 到 DM 6614 生效，并且只有当 PLC 在编程模式时才可以改变。

下面配置选项的描述使用 “x” 来表明 16 进制字符位置，这不会影响配置选项。如果配置选项没有被使用，或者用户手册表明输入值是多少没有影响，养成对这些配置项输入 0 的习惯是一个好的想法。事实上，在一些地方，如果 0 没有被输入到地址中，配置或者指令将不会工作。OMRON 计划扩展 CQM1 系列 PLC 的功能，并且 CQM1 的很多功能和 C 系列的 OMRON PLC 是相似的。

尽管必须输入合适的配置数据到数据存储器中（DM 6600 到 DM 6655），来使能 CQM1 配置选项，但一些特性也需要用户程序执行指令来激活，并且在特性正确地工作前，指令经常需要程序员输入数据到其他的存储区域。在本书的其他一些章节中，将阐述指令和数据的要求；在这节中，将为下面的配置过程，讲述一些关键的数据存储器配置。

(1) 启动操作

(从 DM 6600 到 DM 6614，运行或者监控模式下有效。)

1) 电源恢复后，DM 6600 中的数据值控制 CQM1 输入哪种模式。选项是：

00xx 如果连接了编程控制台，编程控制台上的开关控制 PLC 的模式；否则，PLC 从运行模式开始。

01xx 在电源掉电后，PLC 恢复它先前的模式。

0200 在编程模式中启动

0201 在监控模式中启动

0202 在运行模式中启动

2) 电源恢复后，DM 6601 中的数据影响 I/O 映像的更新、工作存储器以及 IR 和 LR 存储区域中（连接寄存器）的通信消息。选项是：

00xx 复位所有 IR 和 LR 存储器。也复位 SR 25211和 SR 25212。

x1xx 电源失效后，如果 IOM 保持位(SR 25212)置位，存储器中的 IR 和 LR 数据将被保存；当进入运行模式或者监控模式时，操作系统将会保留那些数据。当电源失效时如果 SR 25212中是0，复位 IR 和 LR 存储器。

1xxx 当电源失效时，如果强制状态保持位(SR 25211)置位，存储器中任意的被强制开或者关的 IR 和 LR 数据会以这些状态保存；并且当它进入运行或监控模式时操作系统不会清除这些强制状态。当电源失效时，如果 SR 25211被清除，那么所有的强制状态都将被清除。

3) DM6611 和 DM6612 中的数据被用作补偿值来纠正 CPU44-E 的端口 1 和 2（分别地）的绝对编码器的值。不要手动输入这些值。手册描述了如何使 CQM1 输入数值到这些地址中。

4) DM 6611 中的数据选择使用 CPU43-E 的端口 1 和 2 作为高速计数器模式输入端口（假定是 0000）或者作为脉冲输出端口（假设为 0001）。

5) 在这个范围中的其他数据将不被使用。

(2) 脉冲输出

(DM 6615，运行或者监控模式下有效）输入 xxNN，NN 是输出通道数，写脉冲响应 SPED（64）指令。SPED（64）指令（在第 11 章中阐述）建立一个最高到 1kHz[⊖] 的脉冲频率和选择哪个输出位产生脉冲。

循环时间设置（从 DM 6616 到 DM 6619，运行或者监控模式下有效。）

1) DM 6618 中的值设置循环监控时间（在本书的其他地方叫做看门狗定时器时间)：

0000 默认的120-ms 设置

0btt 设定时间：

- “b” = “1” 代表 10ms 单位，“2” 代表 100ms 单位，“3” 代表 1s 单位。
- “tt” 表示时间单位数（00 到 99)。

CQM1 记录最长的扫描循环时间的长度到 AR26 中，记录最近完成的扫描循环长度到 AR 27 中。

2) DM 6619 用来设置最小循环时间。0000 使选项无效，但任何其他 BCD 码（0001 到 9999）设置 ms 量级的最小扫描时间。如果任何扫描循环在少于最小循环时间之前完成，在启动下一个扫描循环前，CQM1 一直等待直到最小循环时间结束。如果扫描循环比最小的循环时间长，CQM1 打开长时间循环标志位，AR2405，在下一个扫描循环开始前，执行长

⊖ SPED（64）也可以用来初始化通过 CPU43-E 的端口 1 或 2 的更快的脉冲。在这种情况 DM 6615 的配置是不相关的。

扫描循环到完成。

3）DM 6615 可以用来设置扫描循环百分比的上限值，它用来服务 RS-232C 端口。

0000 不限制 RS-232端口的服务时间。
01nn "nn"是限定值的百分比(百分00到99)。

4）DM 6616 限制外围端口的服务时间。如上 DM 6615 设置。

（3）中断设置

（从 DM 6620 到 DM 6639，运行或监控模式下有效。）

1）DM 6620 到 DM 6625 用来为单独的输入映像字节的位选择等待时间。除非在整个等待时间里输入电压保持改变，否则位值不能改变。

xxxN 属于 DM 6620,影响位 IR 00000到 IR 00007
xxNx 属于 DM 6620,影响位 IR 00008到 IR 00015
xNxx 属于 DM 6620,影响位 IR 00100到 IR 00107
Nxxx 属于 DM 6620,影响位 IR 00108到 IR 00115

xxxN 属于 DM 6621,影响位 IR 00200到 IR 00207
xxN(等等),x 属于 DM 6621,影响位 IR 00208到 IR 00215

这里“N”是：

"0" 代表 8 ms(默认)
"1" 代表 1 ms
"2" 代表 2 ms
"3" 代表 4 ms
"4" 代表 8 ms
"5" 代表 16 ms
"6" 代表 32 ms
"7" 代表 64 ms
"8" 代表 128 ms

2）DM 6628 的四个数字各自可以使能一个 I/O 中断或者一个高速计数器中断。一个 I/O中断可以中断主程序来执行一个子例行程序。一个高速计数器可以给输入脉冲计数，并且当达到设置值（SV）时，中断主程序来执行子程序。（可以看第 11 章关于 I/O 中断的讨论，获取更多的细节。）如果 DM 6628 包含：

0000 没有输入中断被允许。
xxx1 IR 00000 能引起 I/O 中断0(调用子程序0)或者能改变高速计数器0的当前值(PV)。
xx1x IR 00001 引起 I/O 中断1(子程序 1)或 HSC1。
x1xx IR 00002 引起 I/O 中断2(子程序 2)或 HSC2。
1xxx IR 00003 引起 I/O 中断3(子程序 3)或 HSC3。

即使 DM 6628 使能中断，中断仍然被屏蔽（被阻止），直到一个 INT（89）指令消除屏蔽。另一个 INT（89）指令选择是否让输入位引起一个 I/O 中断，或者引起高速计数器当前值（PV）的改变。

3）DM 6629 可以用来建立最多 16 个高速定时器。不像标准的定时器，在每次定时值消失时高速定时器的当前值（PV）将会更新，而不只是在定时器指令执行的时候。（查看第 11 章定时器中断的讨论，获取更详细的信息。）如果 DM 6629 包含：

0000 16个高速定时器(TIM 000~TIM 015)被使能
0100 没有高速定时器（其他定时中断将更可靠地执行）

01NN NN 定时器(01~15)被使能,开始于 TIM 000

4) 从 DM 6630 到 DM 6638 表明一个中断服务子程序执行前，哪个输入映像字要立刻刷新（通过读输入模块)。举例来说，在 I/O 中断子程序执行前，DM 6630 表明哪个输入映像字必须刷新。如果 DM 6630 包含：

0000 (默认)不刷新任何输入映像
NNPP 刷新开始于地址 PP(000到011)的 NN(01~12)个输入映像字

在第 11 章中，讲述 I/O 中断前，哪个数据存储器（DM）字用来配置刷新高速计数器，或者定时中断。

(4) 输出刷新方法

(DM 6639，运行或者监控模式下有效)，如果 DM 6639 是：

xx00 (通常)在每次扫描循环后输出映像值被写到输出模块中
xx01 当程序改变它们时,输出映像值被立即被写到输出模块中

(5) 数字开关输入

(DM 6639，运行或者监控状态下有效）如果有一个数字开关连接到输入模块中，可以使用 DSW (-) 指令来读开关量，DM6639 应该配置如下：

00xx 读取一个4位的数
01xx 读取一个8位的数

(6) 高速编码脉冲的计数

(从 DM6642 到 DM 6644，运行或者监控模式下有效)

1) DM 6642 用来配置任意 CQM1 以便对和输入映像位 IR 0004 到 0006 相关的输入触点的输入信号变化（这些脉冲来自增量编码器）进行计数。任何时候 CQM1 在这些输入触点中的一个接收到一个脉冲，CQM1 就会中断它的处理来改变高速计数器 0 的当前值。如果 DM 6642 是：

0000 不计数(没有编码输入)。
01x0 在-32 767和+ 32 767之间计数
01x4 在0和65 535之间计数
010x 在程序打开复位位(SR 25200)后重新计数,然后输入触点00006开
011x 在程序打开复位位(SR 25200)时重新计数

查看第 11 章中的 I/O 中断，可获取更详细的资料。

2) DM 6643 和 DM 6644 用来配置 CQM1-CPU43-E 的增量编码器的输入或 CQM1-CPU44-E 的绝对编码输入的附加输入端口（DM 6643 给端口 1，DM6644 给端口 2)。

(7) RS-232C 端口和外围端口的通信设置

(从 DM 6645 到 DM 6654，当改变后立刻生效）使用这些数据存储器来配置 CPU 模块上的两个通信端口的通信协议。端口可以配置如下：

1) 标准 RS-232C 通信协议，用来传送和接收来自外围或者其他计算机的数据信息。

2) 一对一通信，用来和其他的计算机共享存储器。

3) 和局域网中的一些其他 CQM1（通过主机电脑）进行主机链接通信。

在第 13 章中，描述怎样配置通信，并描述程序指令使用 CQM1 的通信能力。直到读到第 13 章，把所有这个范围内的配置参数设置为 0，然后打开 CPU DIP 开关 5。当开关 5 打开时，CQM1 的 RS-232C 端口的硬件配置为默认配置，在 DM 6645 到 DM 6649 中的设置无效。

(8) 错误日志设置

(DM 6655，改变时立刻生效) 控制怎样记录 PLC 的错误信息。

xxxN 这里"N"影响如何在错误日志中存储 PLC 的出错记录
如果 N=
1 只保存最初的10个错误记录
0 保存所有的错误记录。旧的错误记录将丢失，因为错误日志只能保存10个记录
2到 F 不保存错误记录
x0xx 如果超出最小的循环时间，AR 2405置位
x1xx 如果超出最小的循环时间，AR 2405不置位
0xxx 如果检测到电池电量低，SR 25308置位
1xxx 如果检测到电池电量低，SR 25308不置位

2. 其他 CQM1 CPU 的配置

DM 6144 到 DM 6568 是数据存储器的一个附加区域，它也是写保护的，因此程序员可以使用编程器保存只读数据到这个区域。DM 6569 到 DM 6599 也是只读的，因此它不可以从程序中改变，但是 CQM1 的操作系统会自动地使用它来保存错误日志（将在第 11 章的出错中断一节讲述，在第 15 章将再次介绍）。

3. 配置 CQM1 扩展指令

18 个函数代码保存给扩展指令使用。OMRON 预先分配 18 个扩展指令给函数代码，但程序员可以改变分配。括号内部没有号码的指令是没有分配的扩展指令，可以分配给如下的函数代码：

1) 选择 "UTILITIES" 菜单，然后选择 "SET INSTRUCTIONS" 来获取设置指令菜单。

2) 选择 "EDIT INSTRUCTIONS" 来获取可用的函数代码表，完成默认扩展指令。

3) 选择 "WRITE"，然后把光标移到函数代码，在那里可以分配一个不同的扩展指令(改变那个代码的默认指令)。

4) 按下 [F2] 来获取扩展指令表，把光标移到你想分配函数代码的地方，然后按 [Enter] 键。

5) 按下 [F10] 来记录编程器存储器中的变化。现在可以使用程序中新分配的函数代码。

6) 如果想保存当前分配的扩展指令给以后使用，选择 "SAVE INSTRUCTIONS"。

4. 保存 CQM1 配置到 EEPROM 存储卡中

保存新的配置到到存储卡中，必须插入存储卡并把它的写保护给关掉。记住当 PLC 电源打开时不要插入或者移出存储卡。置位 AR 1400，并把存储器中的配置和程序拷贝到存储卡中。

10.5 在 PLC 程序重启过程中重新配置

既然用户可以改变 PLC 的配置，用户程序可以每次在 PLC 进入运行模式时，重新初始化配置。一些配置选项可以在执行 PLC 程序时改变，因此，当环境条件改变时，可以写入改变 PLC 配置的用户程序。

PLC-5 SLC 500

10.5.1 ALLEN-BRADLEY 重启配置

Allen-Bradley PLC-5 和 SLC 500 使用后备电池，当 PLC 关闭或者 PLC 进入编程模式时，保留它们存储器中的所有数据。当 PLC 重启时，一些状态数据将从状态文件中消失，非保持性定时器将重新设置，输入映像字将刷新，输出映像数据被清除，但在状态文件中的所有配置数据，甚至如计数器累加值将被保存。用户想编程 PLC 以便重启后重置一些存储内容到默认值。在第 11 章，讲述了怎样使用一个错误例行程序重启 PLC，或者程序员使用首次扫描位特性。

PLC-5 或者 SLC 500 在运行模式时，它们会自动地在第一次扫描期间对首次扫描位（S:1/15）置位。PLC 程序可以包括一个梯级，以检测首次扫描位来控制一个配置选项的复位（通过写一个新值到适当的状态文件字中）。这个梯级甚至可以控制一个到子程序的跳转，这个子程序包含一套完整的重新配置语句。

当一个程序改变状态字的时候，动态配置选项可以改变。只有当 PLC 重启后，静态配置选项才会生效。如果必须改变静态配置字，用户程序应该也设置一个主出错位，这样 PLC 会停止，并且操作员强行将开关打到运行模式。（当然，只有当程序检测到需要改变静态状态字的时候，程序才设置主出错位。）手册会告诉哪一个状态字是动态的。Allen-Bradley PLC 可以配置成在每次重启后，执行出错例行程序，取代使用第一扫描位。重新配置指令可以在出错例行程序中。

所有的智能 I/O 模块，包括模拟 I/O 模块，当 PLC 进入编程模式或者断电时，将丢失它们的配置信息。用户程序应该确保在每次 PLC 进入运行状态后，配置数据重新写入智能 I/O 模块中。

S5

10.5.2 SIEMENS S5 重启配置

Siemens S5 CPU 模块可以把配置信息保存在有后备电池的 RAM 中，或者 FLASH 存储器中，或者 EEPROM 存储器中（依靠 CPU 处理和模式），以便在 CPU 处于停止模式或者电源断电后，CPU 配置可以保存。一些配置参数可以通过用户程序重新写入，但是如果 CPU 提供 EEPROM 或者 flash 存储器，它们包含默认启动参数值，重新装载默认参数值是在每次电源打开后自动进行的。有必要编程启动块（OB 21 或 OB22）来重新装载值，这些值不包括来自 EEPROM 或者 flash 存储器的，或者如果启动配置是根据 PLC 检测环境变化的。

除了一些智能 I/O 模块和通信处理模块，当 PLC 不在运行模式时，I/O 模块配置将丢失。每次在 PLC 重新启动后，如果 I/O 模块有 EEPROM 模块保存配置，那么 S5 PLC 应该编程启动块（OB21 或者 OB22）来写配置参数到 I/O 模块。

S7

10.5.3 SIEMENS S7 重启配置

S7 PLC 在系统数据块（SDB）中保存配置数据，甚至在 CPU 没有供电时也会保存。PLC 重启时，包含 I/O 模块配置的系统数据块可以自动地写入 I/O 模块中。S7 CPU 可以有两种方法启动：

1）如果当电源中断时 S7-400 PLC 的运行没有问题，并且它的模式或存储器内容没有变化，当电源恢复后，执行重启。在重启过程中，PLC 要重写 I/O 模块配置参数到 I/O 模块

中，执行重启组织块（OB 100），然后恢复执行中断点的程序。非易失性定时器、计数器或位存储器和 I/O 映像在此种类型的重启中不会复位。

2）如果 S7-400 存储器的内容已经改变，如果一个出错被检测到，或者如果 PLC 已经转换回运行模式，将执行一个完全重启。其他 S7 CPU 只可以使用完全重启方法重启。在完全重启过程中，I 堆栈和 B 堆栈内容被清空，I/O 映像和非易失性存储器将复位，完全重启组织块（OB101）被执行，然后扫描循环从头开始执行。

改变 PLC 的配置作为每天工作的一部分：

1）操作员可以执行初始化存储器复位。存储器复位可以由模式开关手动设置（查看 Siemens S7 硬件安装）或者通过编程终端设置。当执行存储器复位后，CPU 删除 RAM 和 NVRAM 存储器中的几乎所有数据，包括 SDB 中的配置数据。没有被删除的配置参数只有 MPI 地址和波特率，以便 CPU 仍然可以接收来自 MPI 网络上的编程器的数据。如果 CPU 上装有存储卡，那存储卡中的内容（典型的，默认配置和程序）可以自动地下载到 RAM 中。如果 CPU 没有存储卡，必须从编程器中下载一个新的 CPU 程序到 CPU 中。

2）当程序执行时，包括调用 SFC 来改变 I/O 配置。在每次 PLC 重启时，SFC 可以在 OB 100 和/或 OB 101 中被调用来建立一个默认的配置，或者 SFC 可以在用户程序的其他地方改变 PLC 的配置，这些配置在通常程序执行过程中需要。

你可以记录包含 CPU 存储器中的其他配置，以便写到需要的 I/O 模块中。手册解释了每个类型的 I/O 模块 SDB 数据记录的结构。可以调用三个系统函数来初始化 I/O 模块的参数交换：

SFC 55 “WR _ PARAM” 可以用来修改部分 I/O 模块配置（记录 1 中的部分），当 CPU 写 SDB 数据到模块中的时候，可以改变写到模块中的设置。图 10-10 包含一个程序例子，如果模拟输入值超过报警输入值，这个程序可以使用 “WR _ PARAM” 启动一个模拟输入模块来产生硬件中断。

```
      A     I  4.0        // 无论何时输入 I4.0 从关跳到开，
      FP    M  4.0
      S     M  2.0        // 对 " WR_PARAM " 的 REQuest 位 M2.0 置位，
      R     M  2.1        // 并且对 " WR_PARAM " 的 BUSY 位 M2.1 复位，
      JC    init          // 然后向前跳到标号 " init "

      A     I  4.0        // 但当输入 I4.0 为开
      A     M  2.1        // 并且 " WR_PARAM " 没有完成这次传输
      BEC                 // 跳过这个块的剩余部分的执行

      JU more             // 如果两个条件为真，跳过这个例子

init: L B#16#80           // 用于使能到数据块 2 的第一个字节的极限
      T DB2.DBB0          // 中断的二进制模式

      L B#16#10           // 十六进制值 10 放入数据块 2 的字节 6
      T DB2.DB6           // 作为模拟通道组

      L B#16#00           // 0 的上限值
      T DB2.DBB7          // 没有其他 DB2 中的配置字节被改变
```

图 10-10 STEP 7 程序通过使用 WR _ PARAM 来配置一个模拟输入模块以产生硬件中断，尽管它原来不是配置来做这些的

```
CALL "WR_PARAM"                          // SFC 55，可以改变记录 0 或 1
  REQ       :=M2.0                       // M2.0 为真时写模块
  IORD      :=B#16#54                    // 十六进制 54 表示一个输入模块
  LADDR     :=256                        // 框架 0 第一个插槽的模块
  RECNUM    :=B#16#1                     // 十六进制 1 指示改变记录 1
  RECORD    :=P#DB2.DB×0.0 Byte 14       // 如果“WR-PARAM”失败，14 个字节的数据
                                         // 记录的地址 会被送到模块 MW4，用于包含
  RET_VAL   :=MW4                        // 失败原因
  BUSY      :=M 2.1                      // 保持置位直到“WR-PARAM”结束
more:                                    // 程序的剩余部分 …
```

图 10-10 （续）

SFC 56（WR _ DPARAM）可以保存一个 I/O 模块配置的一个记录（你来指定哪一个）到它的保存在 SDB 模块的原始配置中，通过把一个记录从 SDB 写到模块中。

SFC 57（PARM _ MOD）发送一套完整的 SDB 参数到模块中，保存它的整个配置和其他程序特性。

其他 SFC 可以用来读模块的状态（SFC 59，RD _ REC）或者写记录（SFC 58，WR _ REC0）到 I/O 模块中。

CQM1 10.5.4 OMRON CQM1 重启配置

在 CQM1 中，当 PLC 在运行或者监控模式，CQM1 数据存储器中地址 DM 6144 到 6655 中的配置参数是不可以改变的。既然在这一章中讨论的配置参数都在这个范围中，好像不可以写用户程序来动态的改变 CQM1 的配置。动态改变配置在某些时候是不可能的。举例来说，改变扫描时间设定，或者在中断发生时改变输入字，或者改变高速定时器或计数器的数目，或者改变附加端口的设置（对 CQM1 有另外的端口）。

另一方面，一些 DM 参数仅可以启动 CQM1，通过执行合适的指令，用户程序必须打开（或者关闭）。为打开（关闭）输入中断，用户程序必须执行模式控制指令：INI（61），并且有时必须输入设置值到状态寄存器（SR）中。为了使用高速计数器，必须执行寄存器比较表指令 CTBL（63），并且数据集必须输入到存储器中。通过执行一些指令建立和启动高速脉冲发生器，包括速度输出指令：SPED（64）。（这些和其他动态配置的例子将在第 11 章中讲述。）用户程序可以监控可配置特性的输入状态，并且可以根据需要打开或者关闭这些特性。程序员应该锁定状态位，并使用不同形式的初始化指令来防止浪费时间给已经初始化过的特性重新初始化。

然而其他的配置特性可以通过 DM 参数使能，当配置选项被调用的时候，如果用户程序控制位置位。通过编程终端，程序员把这些位打开，用户程序也可以打开它们。I/O 存储器保持配置在 DM6601 中，例如，在启动时使能 I/O 存储器的保持性，但是当 PLC 停止运行时，仅当 SR25212 为开，I/O 存储器才保持不变。通过写到 SR 存储器，一些其他的动态配置选项是可以控制的，它们中的一些将在本书的其他地方讲述（在一些章节中，将处理它们影响的特性）。

有两个另外的状态寄存器位可以用来动态地配置 CQM1：

1）首次循环标志位，SR 25315，在每次 PLC 重启它的扫描循环时开。程序员可以在用

户程序中写程序来检测这个位，并在每次 PLC 进入运行模式后，执行需要的指令来初始化需要的特性。

2）输出结束位，SR25215，置位时，保持所有的输出关闭。

10.6 故障检修

在设置和配置过程中的困难：

1）编程器不能在线连接 PLC，或者不能读或改变选择的存储区域。

2）PLC 错误，或者执行不正确。

3）通过局域网和其他 CPU 连接失败，或者通信缓慢。

4）配置属性不正确。

如果发现不能连接 PLC 在线编程：

1）确认编程器正在试图通信。如果通信经过 RS-232C 端口，使用 RS-232C 的协议检测设备，当试图使用软件传送数据到 PLC 中时，检查传输（TXD/TD）和接收（RXD/RD）线是否连通。

- 如果没有连通，编程器的通信软件设置不正确。试着改变 COM 端口设置，或者将接口电缆插入编程器的另一个串口中。
- 如果连通，除了传送和接收线，检测连接电缆是否正确，特别地，如果 PLC 制造商没有设置它。使用一个连接检测器来保证从电缆的一端到另一端的正确连接（并且没有其他的连接）。

2）检测编程器和 PLC 是使用相同的协议语言。

- 你已经改变 PLC 默认通信协议的一部分吗？举例来说，通过使用 DIP 开关，可以改变 PLC 节点的地址或者波特率，导致编程器试图连接到不存在的节点。改变编程器通信参数来使那些 PLC 相匹配。
- 如果配置参数已经写到 PLC 中，或者如果存储器卡安装在 PLC 的 CPU 模块中，PLC 可以使用不同于默认的通信协议。如果不能确认当前的协议设定，移去任何存储器卡或者后备电池，并且重新设置 PLC 的存储器。在重新设置通信参数后，将删除用户程序和数据存储器的内容。在 S7 系列 PLC 中，存储器复位不会导致 CPU 恢复到默认的节点号，所以如果节点数已经改变，你将不得不试着连接非默认的节点。

3）如果通信经过编程器中的接口卡，确认软件的中断设置（IRQ 号）和个人计算机的地址设置与接口卡上的那些设置相匹配，并且确认不会和已经给个人计算机用作其他目的的端口产生冲突。如果有必要，改变接口卡的设置。这可能需要改变卡上的 DIP 开关或者跳线设置，也需要改变软件中的通信设置。

如果不再访问部分 PLC 存储器来监控或者改变它：

1）PLC 的存储器保护特性被 DIP 开关控制（像 PLC-5），检查这些 DIP 开关。

2）PLC 为选择的存储器区域提供密码保护？在设置密码前，是否有其他人建立密码？如果没有找到密码，可以重新下载带配置的 PLC 程序，这些程序不需要密码（如果想在配置 PLC 后，立刻保存一份拷贝）。在最坏的情况下，可以清除 PLC 存储器。如果不允许通过编程器清除存储器，可以使用存储器复位硬件过程，或者可以移去存储器的后备电池删除 RAM 存储器的内容。

3）一些程序信息不可以通过编程器下载到 PLC 中。如果试图从另一个编程器监控程序，不能使用保存在源编程器的信息。举例来说，Siemens PLC 不包括符号名，或者在跳转中不使用程序标号。从源编程器拷贝遗失信息，使新的编程器可以使用这些遗失数据。

如果 PLC 重复出错或者运行不正确：

1）使用编程器检查 PLC 存储器中的出错代码历史纪录。（CQM1 中保存的出错消息可以关掉，所以可以改变 CQM1 的配置并先保存它们。）代码将告诉你存在什么类型的问题。在 PLC 第一次启动和配置过程中，问题可能是如 I/O 模块的电气出错，循环时间出错（如果看门狗定时器设置太短），或者试图使用不存在的程序（例如，中断文件），或者不存在的存储器。

2）是否正确配置 CPU 和远程 I/O 模块之间的通信？检查远程框架中的 LED 灯来确保远程框架正常地运行。检查为远程框架或者来自远程框架的输入状态保持的状态位。

3）地址控制 DIP 开关是否设置正确？PLC-5 框架必须设置为双插槽、单一插槽或 1/2 插槽，根据你使用的 I/O 模块决定。一些远程 I/O 接口模块和一些智能 I/O 模块有 DIP 开关设置，这些设置影响地址，这些地址在 CPU 内存中使用。

4）检查确定没有偶然地输入不希望输入的配置数据。例如，使 PLC-5 中的主控程序无效，或者把 CQM1 上的输出位关闭，都将会阻止 PLC 控制它应该控制的输出。

如果 PLC 不能通过局域网通信，或者如果通信缓慢：

1）如果 PLC 可以发现在网络上工作的其他 PLC，检测并查看它们。一些 PLC 保持激活节点的列表，可以通过编程终端读这些节点，或者编程终端可以自己保持激活节点的列表，PLC 能够找到它们。如果在网络上的节点失效，或者它的配置和其他的不同，将阻止网络工作。

2）检测并查看这个 PLC 在扫描循环中，是否配置有足够的时间进行通信。通过禁止对通信时间的控制的测试，所有请求的消息都可以响应。连接到 PLC 网络中的 PLC 的问题可能是通信处理时间太少，PLC 不得不处理并不是由它自己的用户程序提出请求的消息传送。

3）你的用户程序是不是想简单地通过网络传送太多的数据？

如果 PLC 不能使用配置的特性，或者如果 PLC 不允许为一个特性输入配置数据：

“RTM”代表阅读手册。（一些沮丧的支持者加入另外一些字，这些字在这本书中不使用。）给工业设备和软件的手册很少会被读者仔细阅读，但在一些地方经常包含需要的信息！阅读可以找到想使用的特性资料。查找在用户手册中关于特性的参考资料的索引，并阅读描述相关特性的章节。

1）确定你使用的 PLC 型号提供你想要的特性！在本书中，将讲述各种型号的 PLC 可以做的事情。

2）确定你已经输入执行一些特性需要的程序或数据。特别是在 OMRON 的 PLC 中，通过配置使能特性后，经常需要执行指令来启动特性。（那些指令打开特性，还是实际执行了吗?）

3）直到 PLC 跳出运行模式，然后再次进入运行模式，一些配置改变才可以被接受。其他一些配置只有在 PLC 电源关闭，然后再打开时，配置变化才可以接受。

4）当 PLC 电源关闭时，配置被保存了吗？大多数 PLC 有一个后备电源来保存存储内容。检查电源。如果不再需要电池来保持 RAM 中的内容，请在电源开时移走电池（另一方面，当电源关闭后，移出电源是清除 RAM 存储器内容的一种很好方法。）

5）设置 DIP 开关，或者改变 CPU 的配置参数来允许保存配置到存储卡中，或者当 PLC 启动后，允许存储卡的内容拷贝到 CPU 的 RAM 存储器中。另一方面，如果空白存储

卡或者有错误配置数据的存储卡被安装后，可以设置 DIP 开关，使 PLC 打开后，存储卡数据不会覆盖有用的配置。是否在电源开时插入或者拔出 EERPROM，从而损坏了它们？

6）配置 Allen-Bradley 块传输 I/O 模块必须小心，否则 PLC-5 发现矛盾，不让你继续。在开始编辑 I/O 配置前保存程序，这样你可以清除 PLC 存储器，重新下载预设的 I/O 配置程序，如果有必要的话。

习题

1. 哪里需要使用终端电阻？
2. 使用硬件配置，什么是典型的关于模拟输入模块的可选项？
3. 一个有三个分布式框架的系统中有多少 CPU 模块？
4. 有三个节点的局域网中存在多少 CPU 模块？

   Allen-Bradley PLC-5
5. 为 PLC-5 在线编程配置编程软件，必须输入两个 DH＋地址，这些地址是做什么用的？
6. 1）在线编程过程中，通过输入时间，或者通过选择自动操作，配置 PLC 通信时间片。在通信时间片中什么通信类型将会发生？

   2）如果为通信时间片输入时间代替允许它保持自动，将会发生什么？
7. 举三个例子，可能改变 Allen-Bradley PLC 的处理器配置？
8. 1）PLC-5 的双插槽寻址系统和 1/2 插槽寻址系统有什么不同？

   2）如果 PLC-5 设置为使用单插槽寻址，每一个输入模块可以提供几个输入映像数据字？
9. PLC-5 的 CPU 模块的 DIP 开关用来做什么？
10. 远程 I/O 模块上的 DIP 开关可以用来做什么？
11. 在配置 PLC-5 的过程中，为什么必须建立起一个远程 I/O 框架扫描表？
12. 如果想阻止没有授权的用户改变程序，在启动 PLC-5 后，将做什么？
13. 智能 I/O 模块有嵌入式微处理器，可以编程来执行特殊功能（在 PLC 的 CPU 模块的控制下）。每次 PLC 进入运行模式的时候，必须执行什么指令重新配置智能 I/O 模块？

    Allen-Bradley SLC 500
14. 什么时候配置 SLC 500 来使能 M0/M1 监控？
15. 哪一个数据字包括使 I/O 插槽禁止位？
16. SLC 500 有什么类型的通信通道？

    Siemens S5 PLC
17. S5 PLC 怎么知道在当前总线单元插槽中有什么类型的 I/O 模块？
18. 中心框架系统和分布式框架系统有什么不同？
19. 接口模块（IM）和通信处理器（CP）有什么不同？
20. 怎样配置循环中断？
21. L1 从站怎样互相交换数据？

    Siemens S7 PLC
22. S7 通信有什么类型的局域网接口可用？
23. S7 PLC 通过默认 MPI 网络交换全局数据前，需要什么配置？
24. 在电源关闭后，智能 I/O 模块的配置数据保存在哪里？

25. 如何配置循环中断？

OMRON CQM1 PLC

26. 如何配置 CQM1，使从 DM6144 到 DM 6655 的配置不允许变化？

27. 1）用户程序可以改变 DM 存储器中的配置数据吗？

2）哪一些 DM 存储器包含的配置数据在改变后立刻生效？

28. 怎样设置 CQM1，当用户程序改变输出映像时，所有的输出映像被立即传送到输出模块中？

29. CQM1 没有定时中断来保证严格的控制操作间隔的一致。怎样配置 CQM1 来以一致的间隔时间进行循环？

第11章 中 断

11.1 学习目标

本章您将了解到：

- 程序中断后发生了什么，以及中断服务程序（ISR）结束后程序如何恢复；
- 在用户程序执行期间，立即 I/O 指令或地址怎样让 PLC 读或写一个 I/O 模块；
- 当一个输入信号变化时，配置 PLC 立即执行一个 ISR；
  - 高速计数器；
  - 频率监控；
- 配置 PLC 按一定时间间隔或在预先设定的时间或延时之后执行一个 ISR；
  - 脉冲发生和脉冲宽度调制；
- 配置 PLC 在程序执行时或掉电后，执行一个出错程序，使 PLC 从检测到的出错中恢复；
- 初始化例行程序；
- 通信功能和中断；
- 修改 PLC 中断响应的指令和/或函数。

现在许多 PLC 都提供中断处理特性，能够用来使 PLC 更快的响应工作环境的变化或使 PLC 在更加一致的时间间隔中响应。一些控制应用，例如位置控制，需要一致的控制间隔；另外一些控制应用，例如在一个机器人车间里响应一个闯入者时，需要立即响应。

11.2 问题

如图 11-1 所显示的那样，当 PLC 运行时[⊖]，标准的 PLC 操作系统引起 PLC 执行一个三步扫描循环。这种扫描循环的最初设计目的就是让 PLC 能够快速地执行控制程序，并且能够持续的按一定时间间隔响应。在每次程序执行时，同时读所有的输入和写所有的输出要比它们分开单独执行效率更高。一致的程序重复时间使得预测控制系统如何响应变得更容易。这种扫描循环仍然是值得考虑的，尽管它有一些不受欢迎的缺点：

1）它内在地引起一个响应延时。完成这三步的总耗时很短，通常都为几微秒，但是在系统控制需要快速响应的情况下，这种延时也是令人讨厌的。例如，如果一个工件在高速传输带上进入一个粉刷棚，但是却在粉刷开始前 15 毫秒已经跨过喷头，那么这个工件就将有一部分没被粉刷到。

2）响应时间是变化的。一个输入状态的变化是在输入扫描步骤开始之前还是之后，决

⊖ 现代 PLC 扫描循环可能也包括处理串行数据通信的步骤，并且有几款 PLC 允许步骤顺序的更改。在这一章，假定使用三步扫描循环。

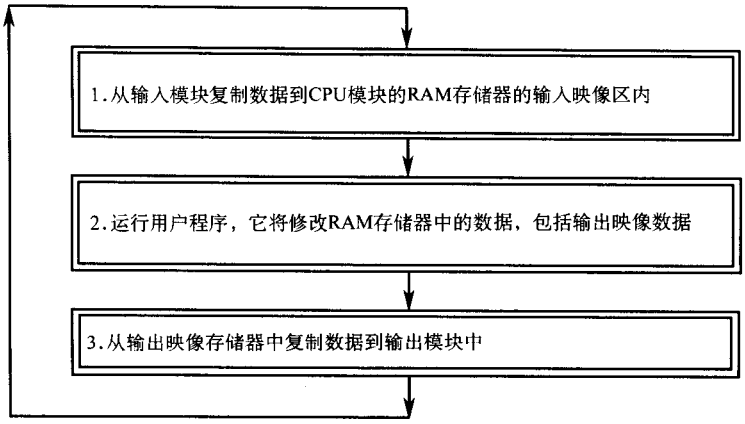


图 11-1 标准的 PLC 扫描循环

定着响应的延时可能是一次或两次扫描循环。在一个实例中，一旦检测到一个靠近的货盘，PLC 就伸展气缸使得货盘离开传输带，并且保证机器人能够对圆盘里的工件进行操作。因为货盘是以 3m/s 的速度运行着，在 PLC 响应延时的变量意味着气缸推动的定位机械装置通常会错过货盘，有时甚至会达到 3cm 之多。

3）结构化的用户程序的执行时间相差很大，使得响应时间不可预测（更少确定性）。程序能包含一个条件跳转到子程序或带有长执行时间的条件执行指令（例如，计算）。如果条件是这几个子程序和长执行时间指令只在一个扫描循环中执行，不在下一个中执行，那么执行时间将变化少一些。

11.3 中断的解决方案

大部分新的 PLC 都提供中断。当程序员所定义的中断情况发生时，PLC 应该立即去响应。当检测到有中断情况发生时无论 PLC 当时正做什么都要立刻停止，而且 PLC 正使用的数据应该保存起来以备后用。中断响应程序随即开始，当中断响应程序结束后，被中断的程序的数据重新获得，同时被中断的程序恢复。中断响应程序正确地叫做*中断服务程序*（ISR），它是这一章我们所要提到的。PLC 的制造商喜欢包装他们自己的技术，所以你会发现 OMRON 把 ISR 叫做*子程序*，Siemens 叫做*功能块*，Allen-Bradley 简单地叫做*程序文件*。图 11-2 展示了当一个扫描循环被中断时发生了什么。

中断特征的六种类型在 PLC 中是可以利用的。在以后接下来的部分被介绍的六种中断类型将更详细地被描述。术语学中的术语也使用在计算机工业中。每个 PLC 的制造商都想发明它们自己的中断类型术语，独立的制造商生产的不同 PLC 型号间的术语经常是互相矛盾的。（本章中被专门的 PLC 供应商使用的术语在处理其他专门的 PLC 时也被用到了。）

1）*立即输入和立即输出指令*引起 PLC 短暂地延缓执行用户程序，从一个输入模块中复制数据字到输入映像区或从输出映像区复制到输出模块，然后执行用户程序的下一条指令。一些 PLC 通过使用特殊地址来提供立即 I/O 功能，这样就使标准 PLC 指令能够触发立即输入和输出。

第一种类型的中断可称作软件中断，因为它只在程序执行一个类似于中断程序的指令时

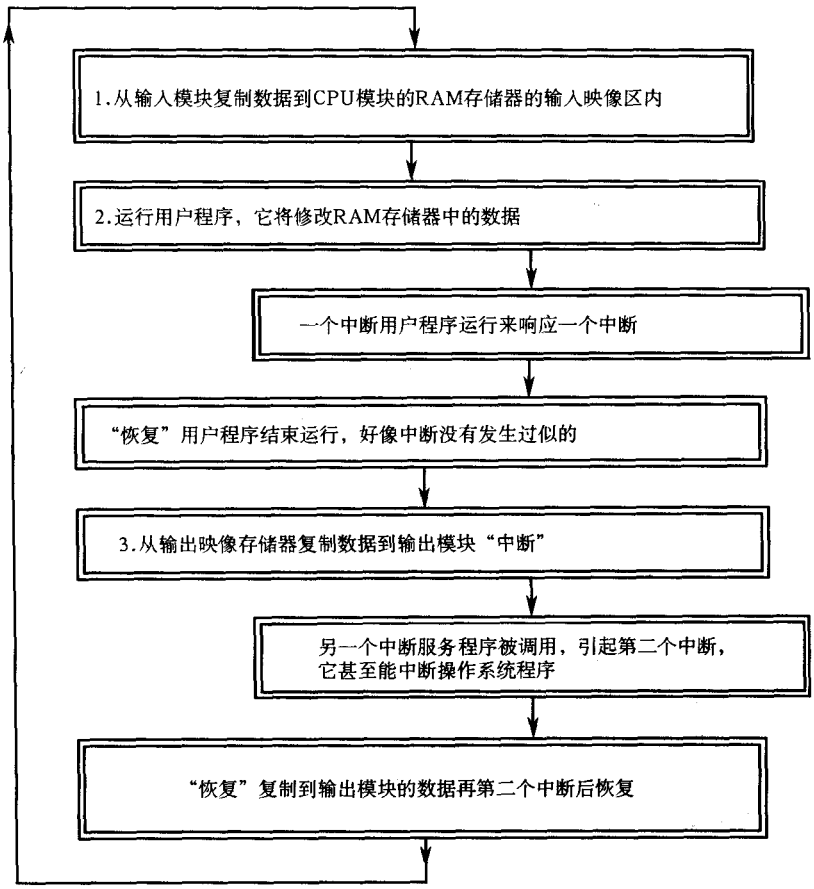


图 11-2　中断的 PLC 扫描程序

才发生。其他五种中断被归类成硬件中断。它们由硬件装置所引起，里面的 PLC 微处理器显示它们需要立即被响应。

2）I/O 中断引起 PLC 响应一个输入触点上的变化。输入信号不再等待输入扫描而直接被 CPU 模块读取。PLC 立即推迟目前所做的事情而去执行一个用户编写的 ISR。一些 PLC 提供一个叫做高速计数器的 I/O 中断变量，内置的 ISR 仅通过它来进行计数增量。

3）定时中断保证在用户编写的 ISR 中产生持续的间隔，或者保证 ISR 能够在一个特殊的时间执行。一些 PLC 提供一个叫做高速脉冲输出的定时中断变量，或者脉宽调制特性，通过它内置的 ISR 在精确的时间间隔使一个数字输出开或关。

4）出错程序中断在 PLC 程序出错或硬件失败的情况下，引起出错程序 ISR 执行。程序出错或硬件失败并不是用户故意引起的，但是用户编写的出错程序却允许程序员控制 PLC 如何响应这些不愿见到的情况。

5）初始化中断让程序员在 PLC 被停止后控制它如何重新启动。PLC 操作系统程序经常包含内置初始化 ISR，当 PLC 程序开始运行时，它们也被执行。初始化例行程序在 PLC 的存储器中做一些例如设定或清除选定数据值的事情（输出映像数据被大多数 PLC 清除），同时自动地执行用户初始化程序。一些 PLC 允许用户编写自己的初始化

程序。

6）通信中断使得包括在 PLC 中的串行通信链接的使用达到最优化。大部分通信控制 ISR 建立在提供给 PLC 的操作系统中，并且不允许 PLC 用户修改。对于一些 PLC，程序员能够选择 PLC 的配置选项来最优化通信和/或编写用来检查串行通信通道的状态从而触发通信程序的程序。

中断服务程序（ISR）不能被扫描循环控制（立即 I/O 类型除外）。如果 ISR 只有在特殊情况才被执行，这些特殊情况很少发生，PLC 为了节省时间可以不检查这些情况。如果在一个单扫描循环中，情况可能发生数次，ISR 将在扫描循环中执行数次。当然，扫描循环被中断将比没有中断花费更多的时间。

所有具有中断能力的 PLC 都定义了中断的优先权。当一个 ISR 运行时它只能被比它优先级高的中断而不能被比它低或同级的中断所响应。所有的 ISR 都有足够的优先权去中断主扫描循环程序。出错程序基本上都有很高的优先权，因此它们能引起其他类型的中断。定时中断的优先权通常都很低。大多数 PLC 都对一些当前没有运行的中断进行排队等候。当一个高优先级的 ISR 结束后，级别较低的就会接着响应。许多 PLC 也提供状态位来显示中断请求是否延时或丢失。在要求严格的应用中，用户程序在执行一个 ISR 时能够通过检查状态位来检测延时。

程序员通过仔细使用中断允许和中断禁止指令来有效地改变中断的标志位。许多 PLC 的中断是默认允许的，直到一个中断禁止指令被执行。当主程序中的一个关键部分被执行时，程序员可能想禁止中断（可能是想阻止 ISR 在正在运行的主程序中修改存储器数据）或想直到出现一个特殊的输入时才开中断。程序员可以在一个 ISR 中包含中断禁止指令，这样这个 ISR 不会被其他 ISR 中断。（ISR 结束之前应该执行一个中断允许指令）。

11.4 关于中断响应的更多细节描述

如果读者对微处理器有很好的了解，会对 PLC 在中断发生时发生了什么感兴趣。其他读者可以直接跳到描述立即 I/O 中断的部分。

对于 I/O 中断或定时中断，下列步骤依次发生：

1）输入端口控制芯片或定时器芯片把连接到微处理器的中断线上的电压拉低。

2）当中断线上的电压下降时，微处理器结束正在运行的中断指令或梯级（一些 PLC 允许用户改变 PLC 的响应速度）。如果允许中断，那么微处理器将在 RAM 的堆栈存储器中储存中断寄存器的内容。微处理器中的堆栈指针寄存器能记录很多储存在堆栈中的数据，这样数据将来能够得到恢复。

3）微处理器在存储器中一块预留的地方重新获得 ISR 的起始地址。这个地址放在微处理器的程序寄存器中，使下一条指令是 ISR 的第一条指令。

对于这一点，中断的响应对于每一个 PLC 是不同的。有时，PLC 已经获得的地址是用户编写的中断服务程序的地址，所以用户编写的 ISR 就立即开始执行。在其他的 PLC 中，PLC 获得的地址是 ROM 存储器的一个地址，同时 PLC 在调用用户编写的 ISR 前（后）执行了许多额外的步骤。这些步骤可能包括一个典型的三步扫描的简化版本：

■ 执行（通常被简化的）输入扫描

■ 执行跳转到用户编写的 ISR

■ 在用户编写的 ISR 结束后，执行输出扫描

4）ISR 最后的指令是结束或返回指令。它使微处理器重新获得在中断时储存在堆栈里的数据，并且能把它们送回原来在微处理器里的寄存器。被中断的程序重新开始，好像从没有被中断似的。（被恢复的数据项中的其中一项内容就是中断前的程序计数器）

立即 I/O 指令的响应和上面的描述很相似，除了它是由程序指令而不是电压的改变引起的，并且它的 ISR 全部存放在 ROM 中。响应一个出错中断稍有不同。如果出错 ISR 不存在，或者在出错中断结束时引起出错中断的问题没有得到修复，那么 PLC 将不会重新运行被中断的程序。代替它的是，PLC 将进入停止模式，同时操作者必须发现并修复出错。在修复出错并使 PLC 转到运行状态后，PLC 通常重新运行它的用户程序，而并不是从它被中断的地方开始运行。

11.4.1　立即输入和立即输出指令

立即输入指令引起 PLC 中断它的用户程序转去直接从输入模块中读取输入数据，而不是使用在输入映像存储器中较早以前的数据。立即输出指令引起 PLC 中断它的用户程序转而直接向输出模块中写入数据，避免等待 PLC 扫描循环向输出模块拷贝数据。在程序中立即 I/O 指令并不阻止 PLC 在扫描循环时从 I/O 模块读或写数据，因此输入和输出数据可能不只一次在扫描循环中被读和写。

立即输入和立即输出指令通常要比从 I/O 映像存储器区中读或写数据花费更长的时间，所以编程人员只有在必需的时候才用它。更快的 PLC 之所以简化了立即 I/O 指令的执行时间，是因为一些 PLC 必须要对某些 I/O 模块使用立即 I/O 指令。它们没有为那些地址设置的 I/O 映射存储器区，（显然）在扫描循环中它们也不包含输入或输出扫描。

立即输入和立即输出中断是软件触发的中断。它们是由程序员在程序中的指令所引起的，并且它们只有当 PLC 执行包含这些指令的梯级时才执行。本章所讨论的其他类型的中断将不受程序控制而被触发，并且在 PLC 循环中能在任意时刻发生。

1. 立即 I/O 指令和 ALLEN-BRADLEY PLC-5

PLC-5

当 PLC-5 遇见含有输入模块（它和 CPU 具有同样的框架）地址的立即输入（IIN）指令时，PLC 从输入模块复制完整的数据字到输入映射文件中。如果输入模块是一个远程 I/O 框架或扩展本地框架，PLC 将从输入模块（通信数据缓存）[⊖] 中复制大多数最近获得的数据，并且把这些数据放到输入映射文件中去。接下来的指令就可以使用到在输入映射文件中最近更新的数据了。

当 PLC-5 遇见一个立即输出（IOT）指令时，如果输出模块是 CPU 的本地框架，它将从输出映射表中复制一个完整的数据字到输出模块；如果该模块是一个远程框架或在一个扩展本地框架里，就复制到要发送到该模块的通信缓存区中。

立即 I/O 指令比其他大多数 PLC 指令要花费更多的时间，但是它们在一些情况中使用，

⊖ PLC-5 在 RAM 存储器的缓存区存储串行通信数据，包括从远程 I/O 框架的输入映像数据。在 PLC 扫描循环的 I/O 扫描部分，CPU 将缓存中的数据拷贝到输入映像文件中。类似地，在 I/O 扫描中，将送往远程 I/O 框架的输出映像文件数据从输出映像文件拷贝到缓存中，并存储在那里，直到这些数据通过串行通信通道被发送。这个过程将在第 13 章详细描述。

使得它们不用执行每一个扫描。对于这些指令来说，不能为了特别区分输入和输出模块地址，而使用前缀“I：”和“O：”。⊖

图 11-3 展示了一个 PLC-5 程序，这个程序使用了立即输入指令从一个本地输入模块中读取一个 16 位的数据字。之后，程序就依靠新的输入映像数据改变输出映像的数据，并且利用立即输出指令发送一个修改过的输出映像字到本地输出模块。

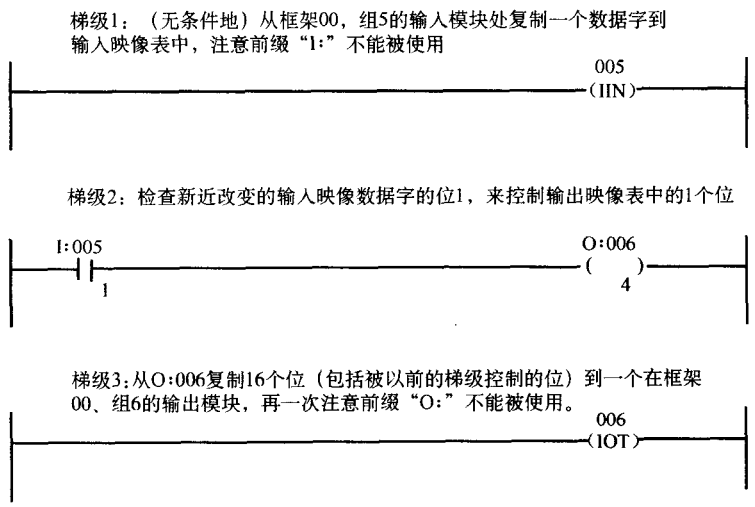


图 11-3 带有立即输入和立即输出指令的 PLC-5 程序

块传输读（BTR）和块传输写（BTW）指令也被看作是立即 I/O 指令，但仅当它们在中断服务程序（ISR）中被执行时才是，这些 ISR 是用来响应定时中断、I/O 中断或出错的（这些中断将在以后的章节中描述）。在 ISR 中当 BTR 或 BTW 出现时，PLC 停止执行 ISR，直到数据块被读取或写入被寻址 I/O 模块。如果 I/O 模块是远程框架，块传输的完成将花费很长的时间，所以增强型 PLC-5 模型是设计用来利用等待时间的。当 ISR 自身等待远程块传输完成时，增强型 PLC-5 将运行 ISR 中断的程序（除非 ISR 是一个出错程序）。

SLC 500

2. 立即 I/O 指令和 ALLEN-BRADLEY SLC 500

Allen-Bradley 的 SLC 500 PLC 的 I/O 指令相对 PLC-5 来说有了几个改善。SLC 500 的指令通常被叫做带掩模的立即输入（IIM）和带掩模的立即输出（IOM），允许程序员指定一个数据字中 16 个位的哪一位从输入模块复制到输入映像数据表中（或者从输出映像表复制到一个输出模块）。在输入映像表或输出模块中的其他位则不受这些指令的影响。另外，SLC 500 的 IIM 和 IOM 指令也允许程序员从一个单独的输入模块输入或输出一串数据字或向输出模块输出一串数据字。

SLC 500 同时也提供一个更新（REF）指令，这个指令引起 PLC 中断自己的程序而在

⊖ 如果指定一个带有前缀的地址，PLC-5 读这个地址的低 9 位，并将它解释为立即 I/O 指令将要读或写的 I/O 模块的一个三数字的八进制地址。例如，如果 N9：27 包含二进制值 1010 1010 1011 0010，IOT N9：27 将读取 N9：27 的低 9 位，010 110 010，解释为一个八进制地址，262，并把输出映像地址的 O：262 的内容写到框架 26、组 2 的数字 I/O 模块。

恢复执行程序前执行一次完整的输入扫描、通信服务扫描和输出扫描。[⊖] 一个服务通信（SVC）指令在恢复程序扫描前中断程序扫描去服务一个专门的通信通道。

一些专用的 SLC 500 I/O 模块必须在使用立即 I/O 类型时才能读或写。在输入或输出数据扫描步骤，SLC 500 能被配置成被专用模块使用的数据字没有或很少被 CPU 读或写，所以 CPU 不用为这些模块维持一个完整的复制 I/O 数据的过程。I/O 模块在它们自己的存储器文件中储存数据。专用 I/O 模块不使用我们本章中所研究的立即 I/O 指令，取而代之的是使用标准 I/O 指令进行读和写，但是利用 M0 或 M1 寻址告诉 PLC 在模块存储器而不是 CPU 存储器[⊜] 中存储数据文件。图 11-4 展示了从一个专用 I/O 模块读和写的过程。与 Allen-Bradley I/O 映像表寻址和数据表寻址相比，M0 和 M1 寻址像个交叉。举例来说，M1：2.3/5意味着位于插槽 2 的专用模块数据文件 1 的字 3 的位 5。前缀“M0”最初是用来表示输出模块文件，“M1”意味着输入模块文件，但现在专用模块中 0 和 1 仅仅意味着两个数据文件中的一个而已。

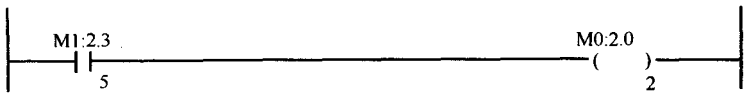


图 11-4　SLC 500 的专用 I/O 模块的 M0/M1 的立即 I/O 寻址

3. Siemens STEP 5 的立即 I/O 指令　S5

Siemens STEP 5 并没有为立即输入和输出提供特殊指令。作为代替的是，S5 PLC 使用带有直接访问寻址的标准输入和输出指令。直接访问寻址和普通的过程输入映像（PII）和过程输出映像（PIQ）寻址很相似，只是寻址前缀字母“P”被“I”和“Q”所替代而已。举例来说，在插槽 4 中直接从输入模块中读取 16 位的值时，指令 L IW004 将被 L PW004 所取代。在插槽 6 中直接写入 16 位的值时，要用指令 T PW006 代替 T QW006。同时，直接写入输出模块的输出值将自动地复制到 PIQ。

一些稍大些的 S5 PLC 没有包含任何模拟 I/O 模块的输入映像存储器（PII）和输出映像存储器（PIQ）。在这些 PLC 型号中，模拟 I/O 值必须利用直接访问寻址方式，从输入模块中读取或写入输出模块。如果存在对应的 PIQ 地址，那么直接写数据到一个模块的输出指令同时也复制数据到 PIQ 表中去，这样在扫描循环时，当 PIQ 表中的内容被写入输出模块时，它并没有改变被直接访问寻址指令所写入的数据的值。当利用直接寻址方式从一个输入模块中读取数据时，S5 PLC 不会自动的用新值来覆盖 PII 表中的数据。如果程序员希望 PII 中包含有新的数据，那么程序必须向 PII 地址写入新值，就像下面例子所展示的那样：

```
L  PW005    ;直接从插槽5的输入模块读一个字
T  IW005    ;更新插槽5的 PII 表的值
```

一些中型的 S5 PLC（例如，S5-103U）实际并不能进行立即 I/O 操作，尽管它们允许程序员用直接访问寻址的方式写入程序。在这些 PLC 中，直接访问寻址仅仅在中断服务程

⊖ 扫描循环中的通信服务步骤是在 PLC 控制通信处理器并与其交换数据时，通信处理器通过串行通信链接来发送或接收数据。更多的信息见第 13 章。

⊜ 一次翻转上升指令（OSR）不能使用 M0 或 M1 数据文件地址。数据移位，顺序器和堆栈（LIFO 和 FILO）指令不能在 M0 或 M1 数据文件中处理数据。

序（ISR）响应一个 I/O 中断、定时中断或初始化的块时才能被应用（在以后的章节中我们都将涉及。当一个 I/O 中断或定时中断发生时，这些 PLC 自动地从输入模块中复制数据到一个特别的中断过程映像输入表（interrupt PII）中，它和普通的 PII 表不同。ISR 中的指令能够用直接寻址的方式从中断 PII 中读取数据或向中断过程映像输出（interrupt PIQ）表中写入数据。在 ISR 或初始化块结束之后，中断 PIQ 表中的值自动地复制到输出模块，但仅当在 ISR 或初始化块执行的过程中，中断 PIQ 的值改变的情况下才可能发生。就像大型 S5 PLC 一样，使用直接访问寻址方式输出一个值的同时也引起普通 PIQ 表中的值更新。因为中断 PII/PIQ 表只有在中断或重启 PLC 时才被更新，所以在一个标准扫描循环程序中使用直接访问寻址方式将会导致 PLC 出错。

直接访问寻址和在 PII 及 PIQ 表中寻址不同之处在于，程序员不能为单独的输入或输出位使用直接访问寻址。指令 A P4.4 和 = P 5.3 不再起作用。使用直接访问寻址去检测独立的输出位需要三个步骤：首先一个完整的字或字节必须用直接访问寻址的方式读出，之后字或字节必须被储存在标志位、数据、PII 或 PIQ 存储器中，然后标志位/数据/PII/PIQ 地址中的独立位才能被检测到。利用直接访问寻址方式写独立的输出位：独立位的值必须被写入数据/标志位/PII/PIQ 存储器中，之后标志位/数据/PII/PIQ 字或字节必须用直接访问寻址方式复制到输出模块（或中断 PIQ）。图 11-5 显示了一个使用直接访问寻址方式的 STL 程序块读一个位和控制一个位。不可避免的是，直接访问输出将同时把其他七个标志位的数据位复制到输出模块，所以例子使用了字逻辑指令（在第 6 章已讲到）保证输出模块中仅有一个位确实被改变了。

```
PB002
包含立即 I/O 语句
L PY002    ;直接从插槽2的输入模块读取低8位
T FY002    ;拷贝一个值到标志字节2,使得每一个位都能被单独检测到
A F2.0     ;如果与插槽2中的输入模块的位0连接的传感器为开,则
=F6.0      ;打开在标志字节6的一个标志位(在这个程序中的标志字节6的其他位总是0
L QB004    ;插槽4中的输出模块获得反映当前低8位状态的数据字节
L KH FE    ; 获取一个除了最低位的每一个位都包含"1"的常数字节值,然后
AW         ;对两个值逻辑与,结果的最低位将变为0,但其他位保持和 QB004中的一样
L FY006    ;从标志字节6(可能全为0,除了 F6.0)获取8位值
OW         ;把它和逻辑与的结果逻辑或,这样最低位被改变为 F6.0的值,但其他位保持
           ;不变,然后
T PY004    ;直接传输结果到插槽4中的输出模块
BE         ;返回调用 PB002的块
```

图 11-5 STEP 5 中的立即 I/O 指令的使用

S7

4. SIEMENS STEP 7 的立即 I/O 指令

Siemens 的 STEP 7 PLC 程序能够直接从输入模块中读取数据和直接向输出模块中写入数据。事实上，过程映像输入表和过程映像输出表仅仅为了可寻址的输入或输出的最低 128 字节才存在。在以上的 127 个字节地址中读取数据或写入数据需要程序员使用立即 I/O 操作。在 Siemens 的宣传资料中，他们把立即 I/O 存储叫做直接访问。STEP 7 同时也提供了四个系统函数来控制 I/O 模块地址组中的立即输入、输出、置位或复位。

和 STEP 5 的语言相比，STEP 7 并没有包含 STEP 5 的立即 I/O 指令。程序员使用标

准指令集，但要输入一个地址作为 STEP 7 的宣传品中所谓的外围输入存储器（例如，“L PIW 003”代替了“L IW 003”），来强制 PLC 从输入模块而不是过程映像输入表中读取数据，或者输入一个地址作为外围输出存储器（例如，“T PQW 007”代替了“L QW 007”），来强制向输出模块写数据。过程映像输出寻址（字节 0 到 127 位）的任何数据写到外围输出存储器时都要自动地向过程映像输出表写入数据。

STEP 7 含有允许外围输入和输出存储器寻址字节（例如 L PIB 003）和 16 位的字（例如 L PIW 003）的指令，并且增加了寻址 32 位双字（例如，L PID 003）的能力，但是仍旧不允许寻址独立的位。比如，你不能输入指令“A PI 003.1”。为了能立即检测输入位，你的程序必须复制整个外围输入字节、字或双字到过程映像存储器、位存储器、本地存储器或在位能够检测到之前复制到一个数据块中。为了能立即向一个独立输出位写入数据，过程映像、位存储器、本地或数据位一定要被用户程序改变，然后包含该位的整个字节、字或双字才能够被传输到外围输出存储器。这个过程和图 11-5 所示的基本相似。

为了能立即从输入模块中复制多个数据值到 PII 表，程序可以执行 SFC26（“UPDAT _ PI”）。程序必须为 PART 参数提供一个数字（0 到 8），来区分 PII 表中的哪一部分要更新。“0”表示更新全部的 PII，其他数字显示 PII 有多大程度的更新。PII 表能够通过使用编程软件来部分地进行配置。SFC 27（“UPDAT _ PO”）能够用来复制所有或部分 PIQ 表到输出模块。

SFC79（“SET”）用来在输出模块中对一系列的位置位。如果这些位是用来寻址哪个 PIQ 含有数据，那么该 PIQ 相应的位将被置位。有了这个“SET”，程序员必须提供一个指针（该指针在第 5 章已介绍）作为 SA 参数识别一个开始地址，同时必须提供一个数字来识别对于参数 N 来说有多少个位被置位。SFC80（“RESET”）能够被调用来在输出模块中复位一定范围的位。

5. 立即 I/O 指令和 OMRON CQM1

CQM1

CQM1 提供一个立即 I/O 指令，I/O 刷新指令：IORF（97）。有了这条指令，程序员可以识别出第一个（开始的）I/O 字和最后的（结束的）I/O 字。IORF（97）指令引起 PLC 立即从输入模块复制数据到指定范围的输入映像数据区和从输出映像区数据字复制数据到指定范围的输出模块。图 11-6 显示了这条指令，当输入模块 5 的输入位 12 开时，这条指令将在每个扫描过程中执行。指令复制所有的输入模块数据（插槽 000 到 011 中的模块）到存储器的输入区（IR 000 到 IR 011），但仅仅复制两个数据字（IR 100 和 IR 101）到输出模块（模块槽 100 和 101），因为指定区域终点结束在 IR 101。

CQM1 也提供了一个在 Allen-Bradley 和 Siemens PLC 中不存在的特征。CQM1 能配置为直接刷新[⊖]。当为直接刷新而配置时，只要 IR 数据被程序改变，则它将被复制到适当的输出模块。你不能限制直接刷新输出的范围。直接刷新不影响输入并且不存在为实行自动的立即输入而做的配置。（接下来我们会看到 I/O 中断和定时中断。无论何时中断程序被执行，CQM1 能够被配置去执行自动地输入刷新并且存在很多方法使得中断程序执行）。

⊖ 对于直接刷新，写数据字 xx01 到地址 DM 6639。当配置为直接刷新，CQM1 也将继续写输出到输出模块，包括在 I/O 扫描循环过程中。DM6639 在 PLC 处于运行模式时不能改变。

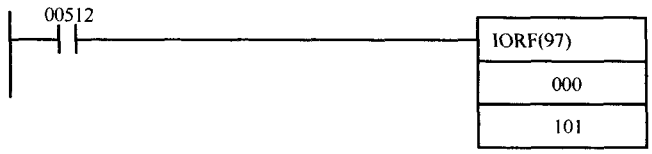


图 11-6 在一个 CQM1 程序的立即 I/O 刷新

11.4.2 I/O 中断

I/O 中断被一个来自输入模块的信号初始化，通常是因为连接着输入模块的一个传感器的改变。当该信号从输入模块到达时，CPU 模块放下它正做着的事情去执行为输入条件而分配的中断服务程序（ISR）。一旦 ISR 结束执行，在收到中断请求信号同时 PLC 重新恢复正在做的事情。I/O 中断一直被认为处于最高优先级的中断等级中。甚至 I/O 中断有时能够中断 I/O 扫描。

I/O 中断被归入硬件中断，因为它是被从设备（硬件）发出的信号而不是被程序（软件）中的指令初始化。一些 PLC 需要使用一些特殊的输入模块，需要引起硬件中断信号的能力。其他一些 PLC 能配置成任何 CPU 模块连接的输入模块都能够引起中断。但仍旧有些 PLC 仅仅允许 I/O 中断响应直接连接 CPU 模块的传感器。

一些 PLC 能够设置成计算输入模块中的变化和仅当计数器到达一个预设值时初始化 ISR。在 CPU 模块中含有内置输入的 PLC，可能有能力很快地计算输入的改变并且可能提供高速计数器中断特征。如果用户程序包含一条能够实现这个特征的指令，无论何时输入信号改变，CPU 就中断去执行一个高速计数器 ISR。ISR 简单地自增或自减一个存储的计数值。当累计的计数值达到一个用户事先选择好的设定值时，用户程序中的指令能检测和响应。高速的计数器输入连接到一个传感器，例如视觉译码器，这个传感器产生的脉冲比大多数 PLC 执行它们扫描循环快，但是需要在计数中没有丢失脉冲。一些没有高速计数器特征的 PLC 提供分离的计数器模块，由它们自己的微处理器来计数在传感器（例如视觉译码器）上发生的改变。

PLC-5

1. I/O 中断和 ALLEN-ERADLEY PLC-5

与 Allen-Bradley 增强型 PLC-5 相比，Rockwell 把 I/O 中断叫做处理器输入中断（PII）。（不错，同样的字母就像 Siemens STEP5 的过程映像输入。这样我们对它有一定的了解而非混淆不是很好吗?）

如果输入组和 CPU 模块在相同的框架，那么在一个模块组中[⊖]PLC-5 能够配置成对来自任何数字输入模块的处理器输入中断（PII）进行响应。输入映像的 16 位中任何一位或多位能够被监视器的一组位所包含，其他位将被忽视。在每个位被监控时，PLC 能配置成反映无论是从错误到正确还是从正确到错误的跳变（当一些位是正确到错误时，另一些就是错误到正确）。PLC 能被配置成运行它的 PII 响应每次程序中的一个位的改变，或去计算改变发生的时间长短，同时运行 PII 响应每次程序计数值达到一个预先设定的值，直到 32 767。

⊖ 一些限制：I/O 模块必须在本地框架中，而不能在远程或扩展本地框架中。PLC-5 框架不能配置为双插槽寻址。如果一个 32 位模块在一个被配置为单插槽寻址的框架中，这个模块不能被监控。如果在一个框架中有一个块传输模块，块传输时某些被监控的位的传送会丢失。

（当 PLC 从编程模式转到运行模式时，计数器重新清零。）当 PII 响应程序时，PLC-5 能配置成运行任何程序文件。尽管当 PLC-5 跳到 PII 程序时不做输入扫描，但在输入模块组被监控引起 PII 时[⊖]，它却保存第一次位改变的记录。PII 程序能够通过检查这次记录来准确决定哪一个改变了状态的传感器持续引起中断。

如果在先前的 PII 程序结束运行之前，一个新的 PII 中断发生，那么这个新的 PII 就等待直到先前的 PII 结束为止。PLC-5 将置位 S：10.12，这是一个用来显示 PII 发生了重叠现象的次要出错位。

图 11-7 显示程序员使用的处理器配置屏幕。在程序编辑过程中为一个程序指定程序文件号，这个程序作为 PII 中断服务程序执行。模块组的地址显示了由于位的改变而应该被监控的输入模块的框架和组号；一个下降计数器数显示了在 ISR 将被执行前，位多久改变一次；对被监控的每一个位，一个 16 位的二进制位掩模值的对应位为“1”；对于每一个要对从错误到正确的跳变计数的位，一个 16 位的二进制比较值的对应位为“1”（“0”意味着正确到错误的跳变要计数）。

```
                          Processor Configuration

User Control Bits     00000000 00000000  RESTART LAST ACTIVE STEP
Fault routine prog file no.:    0          Watchdog (ms):                  500
I/O status file:                0          Communication time slice (ms):    3
VME Status File:              N34

Processor input interrupt    prog file no.:   0    module group:  0
                             down count:      0
                             bit mask:        00000000 00000000
                             compare value:   00000000 00000000
```

图 11-7　定义 PLC-5 处理器输入中断：处理器配置屏幕

示例：

如果下列数据在处理器配置屏幕被输入：

处理器输入中断：

```
program file no.:  12        module group: 05
down count:        6
bit mask:          00000000  00000011
compare value:     00000000  00000010
                                  ||└bit number 0
                                  |└bit number 1
```

然后来自输入模块或在框架 0 或 1 的组 5 的模块的最低的 2 位（含有 CPU 模块）将被中断请求所监控。每次位 0 变假或位 1 变真时，事件被计数。在这些被计数的事件发生 6 次后，PLC 将中断它正做的任何事情，执行程序文件 12 中的程序，然后复位计数器使得中断将不发生，直到六次相似的事件被计数为止。

⊖ 这个“返回掩模”记录被存储在数据字 S：51。S：51 是保持性的，这意味着 PII 调用时是开的位在 PII 结束时仍为开，除非 PII 程序中的指令清除它们。如果当 PII 完成时 S：51 不为 0，下一个“返回掩模”的记录将再次无用，并且 PLC-5 将对 PII 重迭位（次要出错位 S：10.12）置位。

在使用处理器配置屏幕输入数据时，程序员实际上把配置字输入到状态文件 2 中⊖。状态数据字能被 PLC 程序改变。它们不一定使用处理器屏幕才能被输入。取而代之的是，程序员能在一个程序中包含多条指令使其把程序文件数、模块组、下降计数器、位掩模和对比值数据放到数据字合适的位置中。一些程序员包含这些指令在一个初始化程序文件中，这个文件运行在每次 PLC 进入运行模式的时候，以确保一个 PLC 的 PII 配置保持相同，即使操作者不小心修改了屏幕参数的配置。除了 PII 文件数目，仅当 PLC 从程序模式切换到运行模式时，PLC-5 PII 配置的改变才生效，所以写入 PII 配置状态字的初始化程序应该和新的配置而不是旧的相比较，如果发现不同，应该强制用户程序再次将 PLC 转到运行模式。

程序员可以写入一个能够改变哪一个 PII 响应程序文件被调用的程序。图 11-8 显示的 PLC 程序包含一个移动指令，它可以为 PII 程序文件改变存储在状态文件的数据。

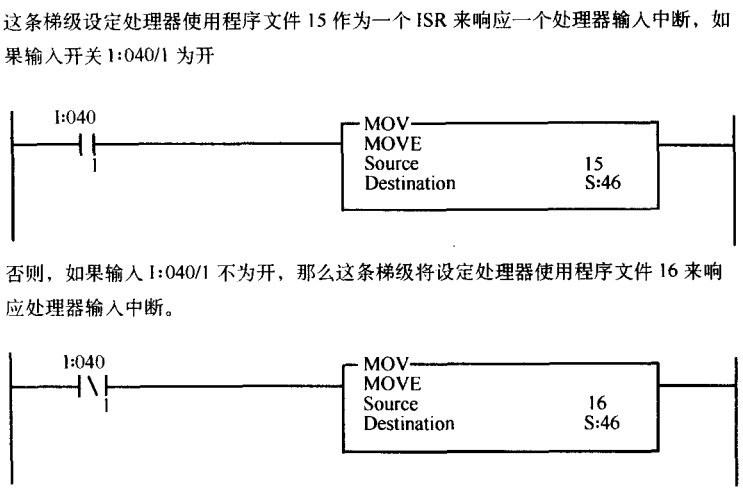


图 11-8 Allen-Bradley PLC 改变 PLC 的中断配置的指令

SLC 500

2. I/O 中断和 ALLEN-BRADLEY SLC 500

Allen-Bradley 的 SLC 500 提供几乎同样的中断，但是叫做离散输入中断（DII），而在 PLC-5 中他们被叫做 I/O 中断处理器输入中断（PII）。大多数 SLC 500 同时也提供一个额外的、低级的 I/O 中断，实际上叫做 I/O 中断，它仅能被专用 I/O 模块初始化。

为了描述离散输入中断（DII），建议读者阅读一下前面的关于 PLC-5 的处理器输入中断的部分。甚至连配置和状态字地址都一样，尽管 SLC 500 的文档有时使用不同的术语。

SLC 500 的 I/O 中断特征之所以和 DII 相似，在于当 I/O 模块产生一个中断请求时，PLC 放下自己正做的事情，去执行一个 ISR，然后再恢复被中断的任务；然而在优先级、结构和中断信号初始化方面也存在一些差别。只有专用 I/O 模块才引起 I/O 中断。中断程序文件号在配置专用 I/O 模块时被使用，而不是在配置 PLC 处理器时。因此，每个专用 I/O 模块能够拥有它们自己的 ISR。(Allen-Bradley 的术语中使用的 ISR 常常特定为被 I/O 调用

⊖ 数据文件号被存储在 S：46 中，模块组在 S：47 中，位掩模在 S：49 中，下降计数在 S：50 中。S：51 指示哪一个被监控的输入模块位最近被改变，导致计数增加。S：52 包含从上一次 PII 运行以来，或者自从 PLC 进入运行模式，被监控的位被改变的次数。

的中断服务程序，但是术语中却说 ISR 为中断子例行程序—好的，任何标准化都好过没有）一个被 SLC 500 I/O 调用的 ISR 必须让中断子例行程序（INT）指令作它的第一条指令，返回（RET）指令作它的结束指令。I/O 中断的优先级低于离散输入中断。事实上，在 SLC 500 PLC 中定义的四种硬件中断中，I/O 中断的优先级最低。（出错中断的优先级最高，其次时 DII，然后是定时中断调用的 STI，最后才是 I/O 中断）。I/O 中断需要一些状态和控制位来防止有时它们被完全忽视。和 DII 中断不一样的是，SLC 500 包含有下列用在 I/O 中断的状态位和字[⊖]：

1）每个插槽都有 I/O 插槽使能位，如果在输入和输出扫描步骤中插槽含有一个模块，这个位必须置位。如果插槽使能位为零，那么 CPU 将不会发送（或接收）任何数据（甚至中断请求）到此插槽中的模块。如果插槽含有一个专用 I/O 模块，那么当 I/O 槽使能位的值改变时，CPU 将向模块发出信号，因为一些专用 I/O 模块被禁止时，它们将改变它们的运行方式。

2）每个插槽都有 I/O 中断使能位，这个位用来控制此插槽中来自专用模块的中断是否引起了一个 ISR 运行。如果这个位被用户程序里的一条指令或使用 PLC 配置屏幕的程序员所清除，那么插槽就不能引起 I/O 中断了。SLC 500 提供能够用来清除或设置独立 I/O 中断使能位的 I/O 中断禁止（IID）和 I/O 中断使能指令。

3）每个插槽都有 I/O 中断悬挂位，无论在 I/O 中断使能位没有置位的过程中 I/O 模块产生一个中断请求，还是因为一个更高优先级的 ISR 已经运行，来自此插槽中的 I/O 中断请求信号没有引起一个中断时，该位都将被置位。如果条件允许，I/O 中断悬挂位被 SLC 500 稍后用来允许 I/O 中断 ISR 运行。复位悬挂中断（RPI）指令可用来为选定的插槽清除 I/O 中断悬挂位，取消还没被服务的中断请求。

4）I/O 中断执行字包含专用 I/O 模块的插槽地址，如果允许的话，初始化目前正在运行的 I/O 中断 ISR。

Allen-Bradley Micrologix 1000 PLC 拥有一个能够最多使用 4 个输入位的高速计数器中断。该计数器能用多种方法配置，包括使用光学编码器。未来的 SLC 50 PLC 将可能提供更先进的高速计数器，可能到那时情况就和你在这里读到的一样。在允许高速计数器之后，当一个信号改变发生时，PLC 监控输入触点 I：0/0 至 I：0/3，并且产生中断去更新计数器。高速计数器能够设置成当高速计数器累计值到达它的预设值时，PLC 将中断来立即写入一个字到 O：0，和/或去执行程序文件 4。用户程序中至少需要两条指令来使用高速计数器。

1）第二条指令是高速计数器（HSC）指令。在你的程序中仅仅存在有一条 HSC 指令。C5：0自动被高速计数器使用，并且包含 16 个状态位。HSC 指令的执行必须有一个真条件语句，使得 PLC 能够设置一个硬件高速计数器。在 HSC 指令中，输入：

- 计数器类型，告诉 PLC 四个输入中有多少到监控器，以及怎么使用信号。八个计数器类型能够被选择，从一个只有在 I：0/0 处计数上升信号的简单上升计数器，到作为 A 和 B 编码器信号来监控 I：0/0 和 I：0/1 的一个（双向）的编码计数器，I：0/2 作为一个 Z（初始化）信号，I：0/3 引起计数器保持状态。

⊖ I/O 插槽使能位对于插槽 1 到 30 在 S：11/1 到 S：12/14，I/O 中断使能位在 S：27/1 到 S：28/14，I/O 中断悬挂位在 S：25/1 到 S：26/14，中断执行字在 S：32。

- 当 HSC 指令初次被执行时，高预置值（C5：0. PRE）被复制到硬件高速计数器，其他时候则依赖于所选择的计数器型号。
- 累加值（C5：0. ACC）。它被高速计数器使用和复制到 C5：0. ACC，仅仅当 HSC 指令执行和/或当程序打开更新的累计值位 C5：0UA。

除了通常的状态位（DN、CU、CD、OV 和 UN）和以上提到的 UA 位，C5：0还包含显示以下情况的位：当累积值高于高预设值（HP）或低于低预设值（LP）；当中断文件 4 由于这些情况（IH 或 IL）或由于 16 位有符号的二进制编号系统（IV 和 IN）已经被调用；当（低优先级）中断文件 4 正等待它的机会去执行或在它能够执行（LS）前已经被调用的情况。

2）用户程序中的第一条指令应该是高速计数器加载（HSL）指令，它为硬件计数器提供比在 C5：0 计数器结构体被 HSC 指令调用更多的配置。有了 HSC 指令，用户可以指定一个源地址，这个源地址显示了五个数据字中的第一个包含了额外的配置数据，每次 HSL 随着真值控制逻辑电路执行时，这个额外数据被复制到硬件高速计数器（就是写入这个逻辑电路使得它仅当配置是必须的时候才为真）。因此，五个配置字按顺序包括：

- 输出掩模。16 位值中的每个“1”显示了一个在 O：0的触点，当高速计数器到达它的预设值时它会被写入。（在硬件配置中 0 默认为禁止向 O：0写入。）
- 高输出源。当到达高预设值时 16 位值将被写入到 O：0（通过掩模）。
- 高预设值。如果 HSL 在 HSC 后执行，那么将输入 HSC 指令覆盖高预设值。在几种定时器类型中，每当到达硬件高速预设值时（看你的使用手册），来自 HSC 的高预设值将被重新设置。
- 低输出源。如果在 HSC 指令中一个双向计数器类型被选择，那么当低预设值到达时，16 位值将写入 O:0（通过掩模）。
- 低预设值。仅当一个双向计数器类型被使用到时才使用。

Micrologix 同时也提供了高速计数器中断允许（HSE）和禁止（HSD）指令，这些指令能被用来允许或禁止调用文件 4 作为 ISR。默认的，文件 4 在 HSC 执行后能够被立即调用。高速计数器复位累加器指令（RAC）能够被用来改变硬件累加器的内容和同时复位（RES）指令在复位 C5：0时复位硬件计数器。

在 CPU 中拥有输入触点的 SLC 5/01 是仅有的在写入时能够提供高速计数的 SLC 500。它拥有一个有限的高速计数器，只能在 I：0/0 计数改变，并且只能当 CPU 模块中的一个跳线被切断时。SLC 500 的高速计数器并不初始化一个中断服务例行程序，当它达到一个预设值时，也不向输出写入数据。在图 11-9 的例子中显示了一个高速计数器稍微不同的版本。HSC 指令是 SLC 500 惟一能使用的高速计数器指令。

S5

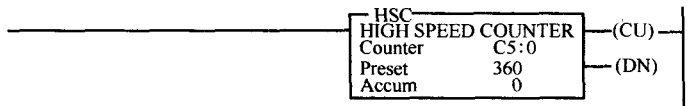
3. Siemens STEP 5 的 I/O 中断

当一个输入信号变高或变低时，Siemens S5 PLC 响应 Siemens 所谓的过程中断（process interrupt）。它响应下面所列举的中断类型的发生：

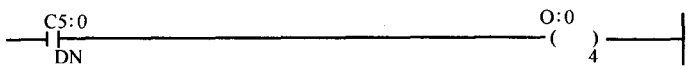
1）过程中断有一个合理的高优先级。当 MPU 完成正在工作的指令后，S5 的扫描循环或定时中断（以后将解释）将取消。甚至 I/O 扫描操作将因为一个过程中断而被中断。初始化例行程序和出错例行程序（以后将提到）将不被中断。

2）在不能使用直接访问寻址来读或写 I/O 模块的中型 S5 PLC 中，执行简化的输入扫

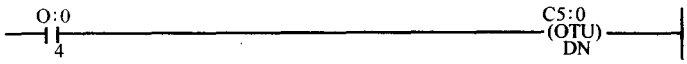
这条梯级设定和允许高速计数器来监控输入 I:0/0 和计数它的跳变直到有 360 个跳变，此时计数器的完成位将关闭，计数器将自动重启。



当高速计数器达到 360 并把它的完成位打开，输出 O:0/4 被梯级打开。在完成位保持开时 O:0/4 将保持开。



为了响应高速计数器到达它的预设值，关闭计数器的完成位。(注意 O:0/4 将因此仅在程序的一个循环保持开。)



每次梯级被扫描时，这个梯级无条件地从高速计数器拷贝累加值到计数器元素的累加值存储字。



高速计数器的累加值（实际上是在先前梯级执行时的值）能和一个不是高速计数器预设值的值相比较，来打开另一个输出。

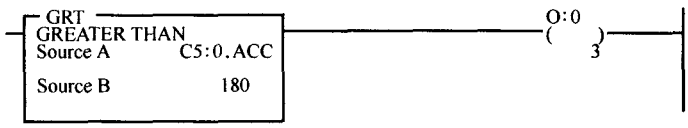


图 11-9　SLC 500 的高速计数器

描。从带有生成硬件中断能力的模块被安装的地址中可以读出输入数据⊖。输入数据不能被复制到标准过程映像输入（PII）表，而是复制到一个特殊的中断 PII 表。直接存储寻址必须被用来读/写中断 PII。

3）程序员已经输入到结构块 2（OB002）的中断服务程序运行。直到 OB002 结束，没有其他的硬件中断被响应。OB002 可能包含跳转到其他块的指令，因而在 ISR 中包括其他块。在 ISR 运行时，已经被中断的功能块不能被调用。如果程序员还没有为 OB002 输入一个程序，来自输入模块的中断信号将被忽视。

稍大型的 S5-115U 型号提供更多的过程中断，记为中断 A、B、C、D。中断 A 引起 OB002 执行，中断 B 引起 OB003 执行，中断 C 引起 OB004，中断 D 引起 OB005 执行。所有的中断都有相同的优先级别，所以如果一个过程中断 ISR 已经运行，其他的三个中断都不能中断它。在这些 PLC 中直接访问寻址读/写 I/O 模块。

⊖ 在 S5-103U，过程中断只能由 4 位数字输入模块或比较器模块来生成。这些模块必须是槽位 0 或 1，并且这些槽位必须在中断型总线单元内。当一个过程中断发生，S5-103U 只能从每一个插槽中拷贝一个字节到中断 PII 的低 2 字节。

4）在 OB002 中，当遇到块结束指令时，那些含有中断 PII 和 PIQ 表的 PLC 执行一个简化的输出扫描，仅复制在 OB002 执行过程中被改变的中断 PIQ 表数据到其影响的输出模块中。

5）中断扫描循环在它被中断处恢复。如果在 OB002 执行过程中其他硬件中断或定时中断信号被接收到，那么只有最经常接收的中断信号才能被响应，即使引起中断请求的条件不存在。

在图 11-10 所示的 STEP 5 的例子中，在插槽 0 或插槽 1 的模块中，OB002 决定中断是否被输入模块产生，从而决定是否跳转到程序块 10（PB010）或程序块 11（PB011）。PB010 检查独立的位（被独立传感器控制的）来决定合适的响应。通过在输入/输出模块和标志位存储器中直接复制数据字节，程序可以完成对数据的位一级的直接访问寻址。为了这个目的，标志字节 5（FY005）和 6（FY006）在位一级的指令中被使用。程序 PB011 的内容在例子中没有被显示出来。

过程中断能被两种方法禁止。如果在 OB002 中不存在程序，PLC 将忽略中断信号。如果用户程序已经执行了 IA（中断禁止）指令，中断响应将被延迟，直到程序执行一个 RA 指令（重新使能指令），在这段时间，大多数新近延迟的 ISR 将执行。

```
OB002
每次输入模块产生一个硬件中断信号时，程序将运行
L IB000      ; 为插槽 0 中的输入模块加载 OLD（预中断）PII 数据
L PY000      ; 直接从插槽 0（或从中断 PII）中的输入模块加载当前数据
> < F        ; 比较这两个值，如果不同（指出这个模块产生了中断），然后
JC PB010     ; 跳转到程序块 010
L IB001      ; 与上面类似，如果中断来自插槽 1 的输入模块，则引起一个到 PB011 的跳转，
L PY001      ;
> < F        ;
JC PB011     ;
BE           ; 返回到被中断的程序（在 PB010 或 PB011 之后直接写新输出数据到输出模块或到
             ; 中断 PIQ）

PB010
被 OB002 调用时运行
T FY005      ; 复制在累加器 1（这是在 OB002 中从 PY000 复制的值）的值到标志字节 5，所以位能被检测到
A F5.0       ; 如果连接到输入模块 0 的位 0 的传感器开，然后
= F6.0       ; 打开标志字节 6 的一个标志位
             ; 如果在输入模块 0 的其他传感器响应这个中断，那么其他逻辑也将响应
L FY006      ; 从标志字节 6（包括 F6.0）中复制 8 位值直接到在插槽 4（或者到中断 PIQ 表的）的输出模块，
T PY004      ; 并自动到 QB004，相同模块的 PIQ 表地址
BE           ; 返回到 OB002
```

图 11-10 STEP 5 ISR

S7

4. Siemens STEP 7 的 I/O 中断

在 Siemens 的术语中，有些地方 STEP 7 把 I/O 中断叫做硬件中断，其他地方叫做过程中断。当接收到一个硬件中断信号时，PLC 将中断任何它正在做的事情，同时调用 OB 40。用户必须先编写 OB 40，否则 PLC 将发生出错。在 S7 PLC 中硬件中断以三种方式使用：

1）硬件中断能被数字输入模块检测到的输入变化初始化，如果模拟输入值超过一个报警极限值，输入变化也能被模拟模块检测到，或者被 CP（通信处理器）或一个 SP（特殊）

I/O 模块检测到。

2）一些 PLC（例如，S7-315-DP）能通过 Profibus DP 局域网发送一个信号到 DP 主站的 PLC，来引起 DP 主站 CPU 执行硬件中断 OB 40。

3）较小型的集成功能模块（IFM）S7 PLC 拥有内置于 CPU 模块的数字输入能力。这些 CPU 的输入能被用来触发（高速）计数器和频率监控中断（这些在 STEP 7 中不会引起硬件中断，不会触发 OB 40 的执行）。

当 PLC 的操作系统程序调用 OB 40 时，它自动地提供 20 个 OB 40 能检测到的本地变量参数。这些参数包含初始化中断请求信号的模块地址，一个指示数字输入号码的序号（如果是一个数字模块来初始化这个信号），或模拟模块的当前状态（如果是一个模拟模块初始化来这个信号）。包括调用的数据和时间，被调用的组织模块的名字，以及它的优先级别，有时还有一些描述为什么产生调用的附加信息，所有这些其他参数将自动地传递给 OB 40（和 OB 40 能够检测到的程序）。如果要看它们分配的 20 个参数和变量的名字的完整列表，请查阅用户手册。

在 STEP 7 系统中，硬件中断是第二高级的中断，所以它将中断任何程序或者除出错 ISR 程序之外的其他任何正在运行的 ISR。如果 OB 40 已经在运行，新的硬件中断信号将不会引起 OB 40 被中断。

为了将来额外的硬件中断 ISR，STEP 7 保留了 OB 41 到 OB 47。所有的硬件中断具有相同的优先级，因此它们不能相互中断。

如果程序员在 OB 40 执行时，想要禁止硬件中断，或禁止其他的中断，有两个选择：

1）编程调用 SFC 39（“DIS _ INT”），告诉 PLC 禁止中断请求。在 DIS _ INT 禁止中断请求后，接收到的中断请求将完全被忽视（尽管与每一个中断请求相联系的信息将在发生中断请求时被存储在诊断缓存）。程序员提供的调用 DIS _ INT 的参数指示是否 PLC 仅仅忽略一种特殊类型的中断请求，指定优先级的所有中断请求，或者除了响应编程错误的出错例行程序之外的所有中断（在 STEP 5 中叫做同步中断）。OB 40 运行时，“DIS _ INT” 能被 OB 40 调用来禁止其他中断。尽管只有出错中断有足够的优先级来中断 OB 40，在 OB 40 运行时禁止中断，将在 OB 40 结束后阻止那些中断被服务。

为了重新使能中断，程序将不得不调用 SFC 40（“EN _ INT”）并确切地指定哪个中断类型或中断优先级将被重新使能。EN _ INT 无须与 DIS _ INT 在同一扫描中被调用。如果一个程序扫描终止时，中断仍被禁止，那么中断在输入和输出扫描步骤中也将保持被禁止的状态。

图 11-11 显示，在 STL 语言中，一个程序如何禁止所有的硬件中断并在之后重新使能它们。在 SFC 39（DIS _ INT）的调用中，模式参数“1”（一个十六进制的数字，前缀“B#16#”）指出一个完整的优先级将被禁止。OB _ NR 参数 40 是硬件中断调用的 OB 号，所以 DIS _ INT 将禁止所有将来的硬件中断调用。RET _ VAL 是一个 DIS _ INT 产生的 32 位数字，对 DIS _ INT 的调用指示将这个数字存储到在地址 MW100 的位存储器。如果 DIS _ INT 成功地禁止中断，这个值将是 0；否则，一个出错代码将被 DIS _ INT 放进 MW100。程序应该包括在调用 DIS _ INT 后检查 MW 100 的指令。对 SFC 41 EN _ INT 的调用包括一个类似的参数值的集合，因为它将重新使能硬件中断。仅仅是 RET _ VAL 参数返回的地址有些不同，因为程序员想要为关于 DIS _ INT 和 EN _ INT 的成功或失败的信息保持不同的地址。

2）编程调用 SFC 41（“DIS _ AINT”）来告诉 PLC 延迟对所有中断请求的响应。在 DIS

```
CALL "DIS _ INT"                    //调用 SFC 39 来禁止中断
 MODE          :=B#16#1             //选择一个完整的优先级
 OB _ NR       :=40                 //指定硬件中断级别
 RET _ VAL     :=MD 100             //接受 SFC 39 的 32 位回复

       :                            //此处的指令不能被硬件中断中断

CALL "EN _ INT"                     //调用 SFC 40 来重新使能中断
 MODE          :=B#16#1             //选择一个完整的优先级
 OB _ NR       :=40                 //指定硬件中断级别
 RET _ VAL     :=MD 104             //接受 SFC 40 的 32 位回复

       :                            //此处的指令不能被硬件中断中断
```

图 11-11 暂时禁止硬件中断的 STEP 7 程序

_ AINT 调用后接收到中断请求会被保持，直到 SFC 42（“EN _ AINT”）被调用，或者直到第一次调用“DIS _ AINT”的组织块结束。如果在 EN _ AINT 被调用前 DIS _ AINT 被调用超过一次，那么在组织块被中断来执行 ISR 以响应中断前，EN _ AINT 必须被调用相同的次数。DIS _ AINT 或 EN _ AINT 不需要任何参数，但是两者都输出一个 RET _ VAL 参数，该参数指示在中断被再次允许前，“EN _ AINT”必须被调用的次数。图 11-12 包含了一个延迟中断响应的 STL 程序。

如果 DIS _ AINT 在 ISR 中被调用，那么它仅仅影响具有比 ISR 组织块优先级更高的中断。低优先级的中断将被推迟直到更高优先级的组织块结束。

```
CALL "DIS _ AINT"                   //调用 SFC 41 来延时中断
 RET _ VAL     :=MD 108             //接受 SFC 41 的 32 位回复，指出 SFC 41 已被调用的
次数

       :                            //此处的指令不能被任何中断中断

CALL "EN _ AINT"                    //调用 SFC 42 来重新使能中断
 RET _ VAL     :=MD 112             //接受 SFC 42 的 32 位回复，
                                    //指出在中断被允许前 SFC 42 必须被调用的次数
       :
                                    //如果返回值为 0，那么此处中断能被任何中断中断
```

图 11-12 暂时延时中断的 STEP 7 程序

Profibus-DP 从站 PLC（例如，S7-513-DP）能够调用 SFC 7（“DP _ PRAL”），通过 Profibus-DP 局域网发送一个信号来引起 DP 主站 PLC 中断和运行 OB 40。当从站调用 DP _ PRAL，它将提供参数显示虚拟的外围输入或外围输出地址。从站 PLC 通过网络传递这些参数，使得它们能被用来作为 DP 主站的 OB 40 的输入参数。OB 40 应该被编程来适当地响应来自虚拟外围 I/O 模块的硬件中断信号。

集成功能模块（IFM）是把数字 I/O 内置于 CPU 模块的 S7 CPU 模块。这些 IFM 模块能够被配置成监控选择的输入，并且当这些位中的一个的状态改变时，中断来更新一个高速计数器。IFM 模块甚至能被配置成比较计数器值是否超出限制值，并且能够在返回被中断程序前进行控制操作。STEP 7 文件并没有为这些高速计数中断指示优先级，但是在它们执

行时，甚至硬件中断被延时都会被警告。这里有四种集成高速中断功能可以利用，但是它们都使用相同的内置数字输入，所以用户只能选择一种：

1）频率监控功能可以用来记录在一个时间段内一个输入触点㊀。在每个时间间隔结束时，CPU 将重复计算频率。STEP 7 编程软件必须被用来配置 CPU 使用 0.1-、1-或 10-s 时间间隔，并且指定一个立即数据块（默认 DB 62）。在用户程序中，执行 SFC 30（“FREQ _ MES”）可以读取频率，并且为 32 位结果提供一个地址，FREQ。调用 SFC 30 的其他可选择的参数包括 32 位有符号二进制的上和下频率极限（PRES _ U _ LIMIT 和 PRES _ L _ LIMIT，MHz），以及强制 SFC 30 开始使用这些极限的 Boolean 位（SET _ U _ LIMIT 和 SET _ L _ LIMIT）。如果极限已经被定义，那么接下来的 SFC 30 的调用能传递输出参数地址以读取当前的极限值（U _ LIMIT 和 L _ LIMIT）和状态位，如果实际频率高于上极限（STATUS _ U）或低于下极限（STATUS _ L），状态位将打开。

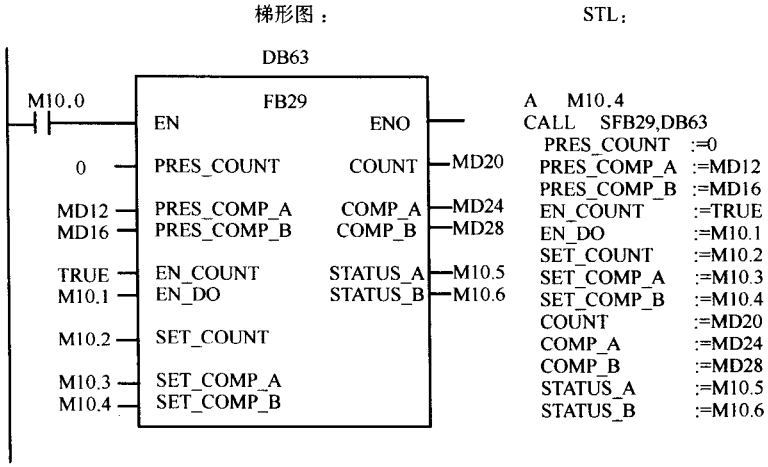
2）计数器功能能够在两个输入触点㊁计数高速脉冲（到 10kHz），如果计数达到两个极限值时，能够立即提供输出和/或初始化一个硬件中断（调用 OB 40）。当在需要的情况下，计数值达到一个值时，Siemens 把它叫做“计数事件”发生了。两个输入（一个上升计数和一个下降计数）中的任何一个，STEP 7 的编程软件必须被用来配置 CPU 计算上升和下降沿，并且还要配置成能当计数高于或低于这两个比较的值时，计数事件是否会发生。配置也指定了在计数事件发生时发生了什么控制操作：初始化一个硬件中断和/或置位（复位）两个输出触点。当计数事件发生时，也可以配置来强制重置计数器和/或设定新的极限值。DB 63 将用作立即数据块，除非默认配置被改变。

计数器将不会计数直到用户程序调用 SFC 29（“HS _ COUNT”）至少两次（就像在图 11-13 中显示的那样），一次先随着 Boolean 参数 SET _ COMP _ A 和 SET _ COMP _ B 为假，之后再为真；第二次调用也应该为计数极限（PRES _ COMP _ A 和 PRES _ COMP _ B）提供 32 位的有符号值，应该通过传输一个真或假的 Boolean 值到 EN _ DO，来允许或禁止输出触点的控制，并且必须通过传输一个真的 Boolean 值到 EN _ COUNT 来允许计数。如果方向控制输入触点和开始/停止触点保持通电，PLC 将记录这两个主要触点的上升计数脉冲和下降计数脉冲的数量。如果方向触点电压下降，计数方向将反向；如果开始/停止触点电压下降，那么计数方向将被冻结。被配置的计数器事件动作将自动地执行。用户程序能再次执行 SFB29 来读取 32 位有符号二进制的当前计数参数（COUNT），读取当前的极限值（COMP _ A 和 COMP _ B），或者检查两个状态位来判断当前的计数是否超出任一个极限值（STATUS _ A 和 STATUS _ B）。

3）S7 314 IFM 允许配置 CPU 来运行两个高速计数器，各带有一个极限值和一个配置好的计数器事件响应。有两个输入（I 126.0 和 I 126.1）用在计数器 A，另外两个输入（I 126.2 和 I 126.3）用在计数器 B。两个输入能够配置成一个上升和一个下降计数器，或一个计数器输入加上一个计数器方向控制输入。其他配置选项在 2）中因为高速计数器而已经介绍过，例外的是必须有两个配置。除非这些默认被改变，计数器 A 使用 DB 60 作为立

㊀ （S7 312 CPU 监控 I 124.6 作为频率监控。S7 314 监控 I 126.0）的上升沿的次数，并在这个时间段结束时计算频率（MHz，到 10，000，000）。

㊁ S7 312 CPU 监控 I 124.6 到 I 125.1 分别作为上、下、方向和开始/停止。S7 314 监控 I 126.0 到 I126.3。这两种 CPU 都控制 Q124.0 和 Q124.1 作为输出 A 和 B。



（记住，因为存在一个调用 SFB29 的参数的立即数据块，所以不需要在每次调用时对每一个参数声明实际值。）

图 11-13 在 STEP 7 梯形图网络中和在 SFC 程序中显示的 SFB 29（HS _ COUNT）

即数据块，计数器 B 使用 DB 61。用户程序必须包括一组对 SFB 38（“HSC _ A _ B”）的调用（带着各自的立即数据块），来设置每个计数器的极限值和使能每个计数器。SFB 38 也允许一个布尔输入来复位计数（对于任何在配置时输入的值）。SFC 38 不提供布尔状态输出。

4）S7 314 IFM 还提供一个马达控制功能块，SFB 39（“POS”），包括对来自增量编码器的高速脉冲计数。POS 把当前的计数值（实际的马达位置）与要求的计数值（期望的马达位置）相比较，并且产生输出来驱动马达更正任何误差。位置误差的大小影响到 POS 驱动马达更正误差的强弱。除非默认配置被改变，POS 使用立即数据块 59。

在用户程序中，能调用 POS 来轻推马达前进或后退（通过设置 POS _ MODE _ 1 或 POS _ MODE _ 2 布尔参数）。当计数器以正确的方向改变时，打开 REF _ ENABLE 来初始化计数器。当在输入触点 I 126.2，或者当 POS 再次被调用时在参数 SET _ POS，观测到一个上升沿，计数器将被初始化（由输入参数 REF _ VAL 提供的有符号 32 位数值）。在初始化发生后，POS 第一次运行时，将置位输出参数 REF _ VALID。

在初始化之后，POS 可以伴随着以下情况被再次调用：一个位置设定值作为输入参数 DEST _ VAL，在 SET _ POS 输入一个上升沿来使 POS 接受位置值，在 POS _ STRT 输入一个上升沿来告诉 POS 开始驱动马达。参数 SWITCH _ OFF _ DIFF 的一个无符号 16 位字的值在设定点附近给 POS 一个允许的范围，在这个范围之内马达将不会被驱动。POS 的输出参数包括 ACTUAL _ POS、当前的有符号 32 位计数和 POS _ READY，它意味着最近期望的位置已经达到或最近的推进动作已经关闭。

CQM1

5. I/O 中断和 OMRON CQM1 PLC

OMRON CQM1 PLC 提供三个 I/O 中断的变量。一个变量叫做输入中断；另一个变量被用作一个高速计数器中断，并且只能对输入信号的改变计数；第三个变量叫做高速脉冲输出，精确地产生时钟输出信号，在定时中断一节将讨论。

为了响应一个输入信号而执行的输入中断，被 OMRON 定义为最高级的 CQM1 中断，所以它们能中断其他任何可能正在运行的 ISR（但不包括 CQM1 中预编好的出错例行程

序)。PLC 在 CQM1 CPU 模块（模块 000）的四个最低序号的输入终端（00 到 03）中的一个或多个接收输入信号。对于这四个输入中断，OMRON 也设定了优先级：中断 0（来自终端 00）的优先级最高，中断 3（来自终端 03）的优先级最低。用户必须把 ISR 程序输入到特定的子例行程序：输入 00000 运行子例行程序 000 来响应一个中断信号，而子例行程序 001 对应 00001，SBR 002 对应于 00002，SBR 003 对于 00003。

OMRON 的 CQM1 必须被配置成响应来自四个输入触点的输入中断信号。[⊖]配置数据必须在 PLC 处于程序模式时输入（PLC 程序在运行时，输入中断配置不能改变）。在 CQM1，中断默认被屏蔽（禁止），所以在为输入中断配置 PLC 后，用户编写的程序必须执行一个中断控制指令，INT（89），来解除输入中断的屏蔽。如图 11-14 所示，INT（89）指令执行时必须带有三个参数。第一个参数是控制代码（CC），第二个参数总是 000，第三个参数是控制数据（D），它的使用依赖于控制代码。控制代码包括：

1）CC＝000 被用来屏蔽或解除屏蔽一些或所有输入中断。D 参数必须是四数字的 BCD 值，其中高三位总是 000。低位代表一个 4 位编码，指示是否屏蔽或解除屏蔽四个输入中断中的任何一个。如果位 3＝1，中断 3（来自触点 03）将被屏蔽（禁止）。如果位 3＝0，中断 3 将被解除屏蔽。类似的，参数 D 的位 2、1、0 控制中断 2、1、0。

2）CC＝002 被用来读取哪个输入中断目前被屏蔽。D 参数必须是一个地址。INT（89）将向地址 D 的低四位写入数据来反映四个中断的屏蔽状态。如果位 3 为开，那么意味着来自触点 03 的中断被屏蔽，等等。

3）CC＝100 屏蔽了所有的中断（不仅仅是输入中断），CC＝200 将恢复在 CC＝100 的 INT（89）执行前中断的状态。在带有 CC＝100 的 INT（89）执行时接收到的中断信号将被记录，并且在带有 CC＝200 的 INT（89）执行时将被响应。由于某些原因，如果在中断 ISR 中执行，CQM1 将不能认出带有 CC＝100 的 INT（89）。D 参数没有使用但应该被设置为 0000。

4）CC＝001 被用来清除正在等待执行的输入中断请求。如果因为一个屏蔽被置位，或由于一个高优先级的 ISR 在运行，一个输入中断被禁止，当输入中断被解除屏蔽或其他的 ISR 结束时，它将被执行（一次），除非带有 CC＝001 的 INT（89）被用来清除等待的输入中断。对应四个中断的 D 中的低 4 位中的 1 将清除任何等待的相应中断类型的中断。

5）CC＝003 被用来设置计数器模式中的输入。在计数器模式，每次中断相联系的输入信号打开时，中断不一定都执行。取而代之的是 CQM1 使用计数器来计算输入信号，并且仅在计数器达到设定值时使 ISR 子例行程序运行。当计数器到达预设值时，计数器也会自动复位为 0，于是输入信号的计数能够重新开始。当 CC＝003 被使用时，参数 D 指示四个中断中的哪一个将被设置为计数器模式。参数 D 的位 0 到 3 中若出现 1 则意味着相应的输入中断不会被这条指令影响，但若是 0，则将使相应的输入中断解除屏蔽，并且在计数器模式时被设置（并且会使计数器重新更新回 0000，所以程序员将可能会使用这条指令的不同形式，这样每次它的输入逻辑为真时，只执行一次）。举例来说，如果 16 位的 D 值的低 4 位是 0111，那么输入中断 0、1 和 2 不受影响，但是输入中断 3 将随着计数器重新清零为 0000 而

⊖ 在 DM 6628 的四个十六进制字符的选定位置写入 1，指示哪一个输入位能够导致输入中断。最低位的 1 使能输入位 IR 00000 来触发子例行程序 000 的 ISR 的执行。倒数第二位对应 IR 00001，倒数第三位对应 IR 00002，最左边的位对应 IR 00003（例如，在 DM6628 的 0101 将使能位 IR 00000 到 IR 00002 来触发输入中断子例行程序 000 到 002，但是 IR 00001 和 IR 00003 不会触发输入中断）。

变成计数器模式。在 0000 和 FFFF（0 到 65，535）之间的十六进制计数器的设定值必须被存储在 SR 244 到 SR 247 之间，分别对应中断 0 到 3，并且在程序执行时能被改变，但是直到 INT（89）在 CC＝003 的情况下被再次执行时，改变才生效。将设定值设为 0000 将禁止中断。计数器的累计值将存储在 SR 248 到 SR 251，分别对应中断 0 到 3。

在 CPU 的四个最低序号的输入终端（00000 到 00003）处，只有从假到真的跳变能够初始化输入中断（或在计数器模式中被计数）。如果这些输入信号的其中一个打开，并且如果输入已经被配置成一个输入中断源，并且已经解除屏蔽（参考图 11-7 的例程），CQM1 将中断它正做的工作，并且：

1）执行一个精减的输入扫描，仅仅读取已经被配置成要读取的输入模块。㊀

2）运行输入中断信号对应的中断服务例行程序。（OMRON 调用 ISR 的子例行程序，同时这些子例行程序被程序员像普通的子程序一样输入，只是指令跳转到子程序不需要被输入。）输入终端 00 引起子例行程序 000，输入 01 引起子例行程序 001，等等。

3）返回被中断程序而不需要发送新的输出数据到任意输出模块（除非 CQM1 已经被配置成直接刷新）。如果需要立即输出，在 ISR 子例行程序中，程序员必须包括 I/O 刷新指令。

在图 11-14 的例子程序中：

1）第一个梯级允许两个中断。它解除中断 0 的屏蔽，把值 5 送入数据字 SR 246 中作为中断 2 的设定值，然后设置中断 2 为计数器模式。（记住，在使用带有 CC-003 的 INT（89）设置计数器模式时，参数 D 的低 4 位中的 0 将设置和允许计数器，所以 1 必须被用来显示哪个中断不会被指令影响。十六进制数字 B 在二进制中是 1011。）输入 00002 的五个激励将引起中断 2。每个指令的差分形式（使用@）被用来保证无论何时输入 IR 00203 打开时，指令都将仅仅执行一次。仅当输入 IR 00203 从假跳变为真时，使用差分形式才能确保计数器将复位。

2）如果输入 IR 00204 为开，第二个梯级禁止所有的中断。

3）如果 IR 00204 为关，第三个梯级在清除任何未决的对中断 0 的调用后重新使能所有的中断。

中断 0 和 2 需要子例行程序 000 和 002（和中断一样的数字）。在使 PLC 进入运行模式之前，程序员必须已经输入 0101 到 DM6628，使能输入 00000 和 00002 作为输入中断源，并且对于此例已经输入 0305 到 DM6630 和 DM6631，指示和 IR 005 一起开始的三个数据字在每一个中断开始时必须刷新。

1）SBR 000 从输入模块 005 立即复制刷新过的数据字到输出模块 104，使用一个 MOV（21），接着是 IORF（97），被“总是开”位控制。

2）此例的程序中 SBR 002 没有显示。

OMRON 的高速计数器（HSC）利用一个 I/O 中断来初始化一个预编好的 ISR，以便在 CQM1 的 CPU 模块的输入触点对接收到的输入信号计数。大多数 CQM1 的型号仅能使用 HSC0 来对连接到触点 00004 到 00006 的信号计数，但是一些型号在 CPU 中有两个附加的连接端口，并且也能使用 HSC1 和 HSC2 在这些端口对输入信号计数。在本节中我们先描

㊀ 对于中断 0（被 IR 00000 初始化），DM 6630 的最高位 BCD 数字必须包含要读取的输入字的序号（00 到 08），低位的数字必须指示读取的（00 到 07）第一个模块地址。DM 6631、DM 6632 和 DM 6633 类似地各自用于被位 IR 00001、IR 00002 和 IR 00003 初始化的中断。（例如，如果 DM 6632 包含 0305，当 IR 00002 导致中断 2 时，在运行 ISR 子例行程序 002 之前，CQM1 将来自输入模块 005、006 和 007 的输入数据拷贝到 IR 005 到 IR 007。）

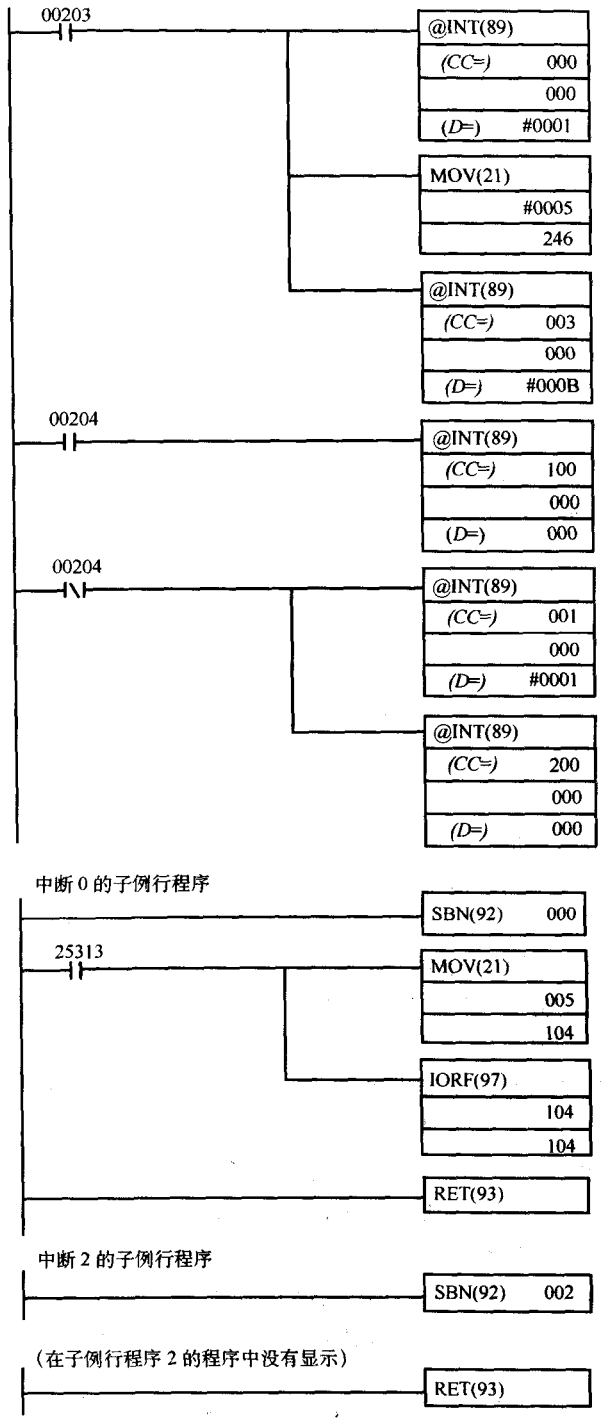


图 11-14　使用 CQM1 输入中断的程序

述 HSC0，之后讨论 HSC1 和 HSC2。在 CQM1 中高速计数是最低优先级的中断，所以如果另外一个 ISR 已经运行或如果 INT（89）已经屏蔽掉所有的中断，那么一个高速计数器中断服务例行程序可能被延迟。

在任何程序能使用 HSC0 之前，CQM1 必须通过在 DM 6642 中的十六进制值 01xx 来配置使用 HSC0。㊀（最后两个数字也会影响计数器并且会在以后讨论。）配置可能也包含向 DM 6638 输入一个数据字来指示无论何时一个 HSC0 中断服务例行程序执行时，CQM1 都应该从最多 12 个输入模块中读取数据来开始工作。㊁只有当 CQM1 处于程序模式时，配置值才能被改变。如果配置成使用 HSC0，那么每当 CQM1 工作在运行模式时，CQM1 将复位 HSC0 的预设值，同时也会计算在运行模式中，到达输入触点 00004（也可能是 00005）的输入信号。高速计数器中断默认被屏蔽，所以用户程序必须包括一个寄存器比较表指令，CTBL（63），来“登记”（提供）与当前值比较的设定值表，以及与对应表中各项运行的 ISR 列表。CTBL（63）指令（下面将描述）也能被用来对比较初始化。在 CTBL（63）执行来登记设定值表后，模式控制指令，INI（61），可以开始或结束比较。计数器的当前值在每个扫描循环自动地复制到数据存储器地址 SR 230 和 SR 231，但是高速计数 PV 读取指令，PRV（62），能在程序执行时更新 SR 230 和 SR 231。

HSC0 能被配置成使用两个方法来计数，这依赖于在 DM 6642 的最低十六进制字符位置的配置值。

1）如果 DM 6642 包含 01x4，HSC0 处于增量模式。它将计算在 00004 的输入信号打开的次数，只能向上计数，从 0 到 65 535。

2）如果 DM 6642 包含 01x0，HSC0 在上升/下降模式。如果一个增量编码器被用作位置传感器，应该选择这种模式。无论何时，在 00004（编码器的 A 相输出）或 00005（编码器的 B 相输出）的信号改变，那么 CQM1 将增或减计数。计数器的当前值范围将会是 −32 767到＋32 767。㊂

为了复位 HSC0 的当前值，三个方法中的一个可以被使用。当 PLC 进入运行模式时，如果 DM6642 包含 011x，程序能够对 SR 25200 置位来使 HSC0 的当前值复位为 0，随后 SR 25200 将复位，然后在输入 00006 的信号将打开。这个信号来自于一个增量编码器的 Z 相输出。不管 DM 6642 的值，模式控制指令，INI（61），能被用来改变 HSC0 的当前值到 0 或到任何其他的值。模式控制指令将在以后描述。

注册比较表指令，CTBL（63），显示了一个八字符 BCD 值的表，用来与 HSC0 的当前值相比较，并且如果发现匹配，还提供被调用的子例行程序号。程序员必须在 CTBL（63）执行前，输入数值表到顺序的存储器位置。这里有两种类型的比较表可能被用到：

1）目标表，它含有最多和 16 个目标值相关的子例行程序号，如图 11-15 所示。一般来说，当计数器的当前值已经被增加到目标值时，子例行程序将运行，但是如果子例行程序号的第一个字符为 F，子例行程序将在 HSC0 已经减少到目标值后执行。在 CTBL(63) 已经登记图 11-15 中的示例目标表后，并且比较已被初始化，那么当 HSC0 的当前值增加到＋10 300时，子例行程序 15 将被调用。计数将继续，如果计数超过＋21 111，子例行程序 16

㊀ 如果设置 DM6642 来使 CQM1 使用 HSC0，CQM1 不能使用间隔定时器 2。

㊁ DM 6638 的最高两个 BCD 字符指示有多少输入模块要读取（00 到 11），最低位字符指示从哪一个输入地址开始（000 到 011）。

㊂ CQM1 以 BCD 码格式存储当前值。SR 230 保持当前值的四个最低 BCD 字符。SR 231 在低 4 位包含高 BCD 字符值。除非当前值为负（在这种情况最高位将是 F），SR231 的其他 BCD 字符将是 000。例如，如果 HSC0 的当前值是－12，345，SR 230 将包含值 2345，SR 231 将包含 F001。

将被调用，然后再减计数到+21 111。（如果目标值的高四位 BCD 数字第一个字符为 F，那么目标值为负）

	EG:
表中目标值（BCD）的数目	0002
第一目标：低 4 位 BCD 数	0300
第一目标：高 4 位 BCD 数	0001
第一子例行程序：如果 PV 达到第一目标值	0015
第二目标：低 4 位 BCD 数	1111
第二目标：高 4 位 BCD 数	0002
第二子例行程序：如果 PV 达到第一目标值	F016
以下是不相关的值	

图 11-15　CQM1 CTBL（63）指令中需要的目标表

2）范围表，必须包含八个范围，能以图 11-16 所示的形式输入。只要 HSC0 的当前值输入一个范围，不论增加还是减少，子程序都将立即执行。对于八个可能范围中的每一个，

	EG:
第一范围：低限：低 4 位 BCD 数字	0333
第一范围：低限：高 4 位 BCD 数字	0001
第一范围：高限：低 4 位 BCD 数字	0444
第一范围：高限：高 4 位 BCD 数字	0002
第一子例行程序，如果 PV 在第一范围	0017
第二范围：低限：低 4 位 BCD 数字	0555
第二范围：低限：高 4 位 BCD 数字	F003
第二范围：高限：低 4 位 BCD 数字	0666
第二范围：高限：高 4 位 BCD 数字	F002
第二子例行程序，如果 PV 在第一范围	0018
等等	

图 11-16　CQM1 CTBL（63）指令需要的范围表

数据表必须包含数据，即使不是八个范围都需要。(每个没有用到的范围都必须输入子例行程序 FFFF)。就像在目标表中，如果一个极限值的高四位 BCD 数字的第一个字符为 F，那么极限值为负。在 AR 11 的八个状态位反映了当前值是否在范围表的范围中。如果输入图 11-16 中的示例值，并且 CTBL (63) 已经登记它们，同时比较已经被初始化，那么当 HSC0 的当前值出现在范围+10，333 到+20，444 之间时子程序 17 将被调用，当 HSC0 的当前值处在-20，266 到-30，555 之间时，子程序 18 将被调用。

CTBL (63) 应该以差分形式使用，@CTDB (63)，因为它有长的执行时间，因此只有在比较表中的数据需要被初始化或被改变时才执行 (例如，每次 PLC 进入运行模式)。CTBL (63) 必须提供三个参数：

1) 第一个是 P (端口限定) 参数。因为高速计数器 0 使用端口 0，P 必须是 000 (CPU 输入触点 00004 到 00006)

2) 第二个参数是 C (控制) 参数。C 指示被输入的是什么类型的比较表 (000 或 002 意味着目标表；001 或 003 意味着比较表)，以及指示表的使用是否将立即被初始化 (000 或 001 初始化比较和允许中断；002 或 003 意味着等待直到 INI (61) 指令初始化比较和中断)。

3) "TB" (表) 参数是最后一个，它必须在比较表中包含第一个数据字的地址。

高速计数器中断的开始和中止，可以通过 INT (89) 屏蔽或解屏蔽所有中断，或利用模式控制指令 INI (61)。如果被屏蔽，那么计数器将继续但没有 ISR 运行，并且在 AR11 的状态位将继续反映当前值在通常是否将引起 ISR 执行的范围之内。INI (61) 必须包括三个参数：

1) 第一个参数，P (端口限定)，必须是 000 来控制 HSC0。

2) 第二个参数，C (控制)，如果是 000 则解屏蔽 (初始化) 比较表的使用来引起中断，如果是 001 则屏蔽 (停止) 它的运行。002 意味着接下来的参数为计数器提供新的当前值。004 被用来控制脉冲输出，但不为高速计数器使用。

3) 第三个，P1 (第一个 PV) 将是 000，除非计数器的当前值将改变。如果是这样的话，P1 必须指示为当前值而包含一个新的八位 BCD 字符的值的两个地址中的第一个地址。(就像在比较表中，第一个地址必须包含低四位 BCD 数字，下一个地址必须包含高四位 BCD 数值。) 推荐使用差分形式，@INI (61)；否则，在这条指令的控制逻辑为真时，当前值将在每次循环被改变。

如果计数超出允许的范围之外 (增模式是 0 到 65 535，增/减模式是 -32 767 到 +32 767)，HSC0 将停止工作。在计数高出范围之后，八位的 BCD 当前值将被设置为 0FFF FFFF，或在计数低于范围后，被设置为 FFFF FFFF，所以可以监控第六位和第七位数 (它们应该总是为 0)，检查出这种情况。如果非常少的情况下，计数器必须复位，并且在高速计数中断恢复之前，比较值重新启动。

图 11-17 显示了一个使用高速计数器中断特性的程序。在使 PLC 进入运行模式前，DM 5542 包含 0104 来配置 CQM1 使用 HSC0，需要一个来自输入 00006 的信号来复位计数器 PV 值和使用增/减计数。DM 6638 包含 0203，在每次 HSC0 调用一个中断子例行程序时，配置 CQM1 读取输入模块 003 和 004 到 IR003 和 IR004。含有范围数据的两个表已经被准备；第一个在 DM 0000 到 DM 0039，第二个在 DM 0040 到 DM 0079。程序应该也包含子例行程序，使得 HSC0 能引起执行，但是此例中并不包括子程序。

在图 11-17 的程序中：

1）梯级 1 使用第一扫描位（25315）来记录开始于地址 DM 0000 的范围数据表，以及初始化 HSC0 的当前值与表的比较。因为第一扫描位仅仅在第一次扫描时为开，所以不需要 CTBL（63）的差分版本。

2）当输入 IR 00102 打开时，梯级 2 执行 INI（61）来停止为 HSC0 而使用的比较表，执行 CTBL（63）来改变范围表，但不重新启动比较的使用来初始化中断，并且最后打开 HSC0 的软件复位位，使得当在输入 00006 处接收到一个信号时，计数将复位为 0。

3）当输入 IR 00102 重新变为关时，梯级 3 初始化新的比较表的使用。

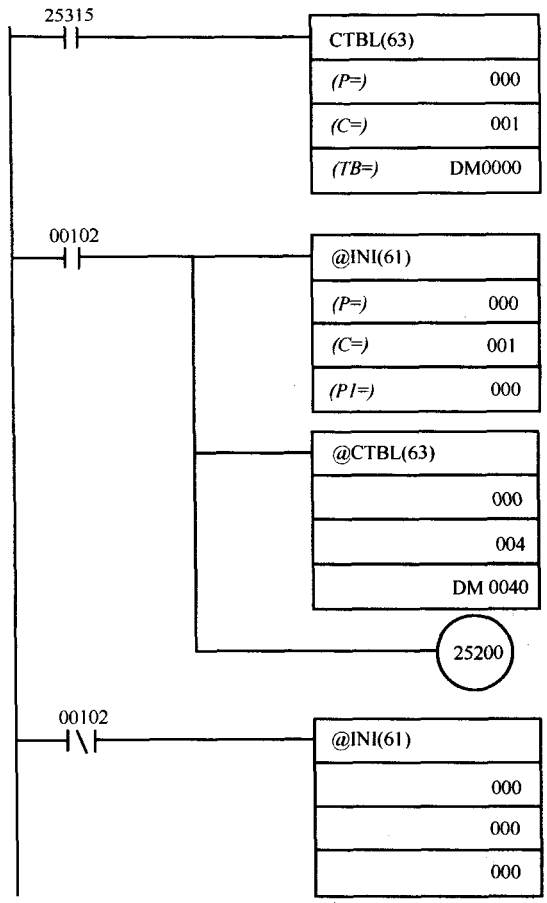


图 11-17　OMRON CQM1 的高速计数器中断

CQM1-CPU43-E 提供另外两个高速计数器，HSC1 和 HSC2，它们的操作与标准 HSC0 相似，只是 HSC1 对在输入端口 1 处接收到的输入信号计数，HSC2 对在端口 2 处接收到的信号计数，所以 CTBL（63）的 P 参数、INT（61）和 PRV（62）必须是 001 或 002。HSC1 和 HSC2 的配置要求和数据字的使用与上面描述的 HSC0 有微小的差别：

1）DM 6611 选择使用端口 1 和 2 作为高速计数器输入或作为脉冲输出端口。（脉冲输出将在以后章节中描述）

2）DM 6634 被用来配置当 HSC1 初始化一个中断时哪个输入需要更新。DM 6635 被 HSC2 使用。

3）DM 6643 被用来配置复位 HSC1 的当前值的方法。DM 6644 被 HSC2 使用。

4）在每一个扫描，HSC1 的当前值被拷贝到 SR 232 和 SR 233。HSC2 使用 SR 234 和 SR 235。

5）AR 05 包含 HSC1 的状态标志位。HSC2 使用 AR 06。

在高速计数器 0 和高速计数器 1 和 2 之间存在一些不同：

1）DM 6643 和 DM 6644 配置的计数模式是不同的。HSC0 所谓的增/减计数对于 HSC1 和 HSC2 叫做差分相位模式。对于 HSC1 和 HSC2 来说，增/减模式意味着在 A 相输入的信号引起了计数器增加，B 相引起了计数器的减少。增加了一个额外的脉冲/方向模式，在这种模式下，一个在 A 相的信号显示了在 B 相的信号是否引起计数器的增或减。

2）DM 6643 和 DM 6644 选择线性或环形模式计数。在线性模式中，HSC1 或 HSC2 在 −8 388 607和+8 388 607 间计数。如果它的计数值超出了这个范围，计数器停止。在环形模式中，计数器在 0 到 64 999 之间以任一方向计数。环中点的数目（1 到 65 000）必须在目标或范围值前，由比较表中的第一个双数据字中提供。如果计数器超过了最大值，那么它将重新从 0 开始（如果它在 0 以下计数，那么将从最大值开始）。

3）HSC1 有比 HSC2 高的优先级。这两个的优先级都比间隔定时中断或 HSC0 高，但是比输入中断的优先级低。

要了解更多的细节，可查阅 CQM1 的使用手册。

CQM1-CPU44-E 也提供了 HSC1 和 HSC2，但是他们被设计用来对来自 8-、10-或 12-位绝对编码器的信号的改变计数。不像上面 CQM1-CPU43-E 的描述，在端口 1 和 2，DM6611 和 DM 6612 没有被用来在计数器输入或脉冲输出间进行选择（CPU44-E 不能通过端口 1 或 2 输出脉冲），但取代的是，能包含一个补偿值，它从 HSC1 的当前值中减去。另一个不同是 DM 6643 和 DM 6644 被用来配置 HSC1 和 HSC2，以监控 8-、10-或 12-位的绝对编码器，并且指示是否转换当前值为以度为单位的数，或以 BCD 的形式保留它们。如果选择转换到度，那么 PV 的最大值将是 359，并且必须为目标或范围极限准备带有单个数据字的比较表。

11.4.3 定时中断

中断服务程序（ISR）可以通过定时中断以精确的时间间隔运行。中断定时器产生中断请求，同时 PLC 的微处理器中断它正做的事情，运行 ISR，然后恢复被中断的过程。当然，间隔必须有足够长的时间保证每个定时中断 ISR 能在下一个开始前结束。

定时中断被用来控制需要一致控制间隔的过程，并且用来控制不需要经常调节的过程。将不是经常需要的控制例行程序放入定时中断 ISR，意味着它们不是在每个扫描循环中都执行，所以主程序将执行得更快（当然，除了那些中断来执行 ISR 的循环）。

有时整个用户程序被编写为定时中断 ISR。将整个程序放入定时中断（设置中断时间最少等于可能的最大扫描时间间隔）强制 PLC 的控制间隔一致，一致的间隔时间必须足够长。一些 PLC 允许程序员配置 PLC 在 PLC 扫描循环之间有一个最小时间，这样做和将主程序作为定时中断 ISR 一样有效。

定时中断通常比 I/O 中断优先级低。这意味着定时中断将不会中断响应 I/O 中断的 ISR，但是 I/O 中断能中断定时中断 ISR。一些 PLC 在执行定时中断 ISR 前执行输入扫描循环，有些则不用。如果 ISR 使用过时的输入数据，使用定时中断的好处可能不能体现，所以如果 PLC 不能自动执行输入扫描，那么 ISR 应该包括立即输入指令来读取和使用最近可能的输入数据。简单来说，在定时中断 ISR 完成之后，一些 PLC 执行输出扫描循环来把修改过的输出映像表数据写入输出模块，但有些则不用。程序员可能希望在 ISR 中使用立即输出指令来控制写输出数据到输出模块。记住立即 I/O 指令比大多数 PLC 指令花费更长的执行时间。

一些 PLC，如果在 PLC 关闭时它们有时钟继续运行，能被编程来按照用户编程的日期和时间执行定时中断 ISR ，或者在相同的天、周和月份等等。到目前为止，PLC 都需要打开并在运行模式，否则它们将不会运行定时 ISR。在你阅读到这里的时候这些可能已经改变。甚至 VCR 能在预设的时间自己打开并记录程序。

一些近来的 PLC 提供可编程的高速脉冲输出。一些甚至为模拟输出提供脉宽调制脉冲输出。这些 PLC 使用定时中断来调用脉冲控制的 ISR 在一个精确控制的时间间隔来打开和关闭输出位。这些脉冲控制算法在 ROM 中的 PLC 操作系统程序中。程序员需要输入指令和/或配置数据，配置数据指定输出位将被脉冲开或关的地址，而指令用来开始和停止脉冲，以及用来指定脉冲的频率和可能的占空比。(对于脉宽调制)。

一些 PLC 提供可编程的延时特征，利用它用户程序能包括一条指令，在指令执行后的一个精确控制时间里引起中断。举例来说，程序可能执行一个立即输出指令，然后初始化一个 ISR 的延时执行，这个 ISR 将执行一个立即输入和检查立即输出的效果。在延时时间里，PLC 将正常地执行它的扫描循环。

每个 PLC 有一个看门狗定时器，它是定时中断的一种类型。无论何时一个新的扫描开始，看门狗定时器将重启。如果扫描时间超过了看门狗定时器的设定值，扫描将被中断并且 PLC 将进入出错模式。扫描时间能包括主程序的扫描时间再加上运行任何可能执行的 ISR 的时间。大多数 PLC 允许通过改变 PLC 的默认配置数据，修改看门狗定时器的设定时间。大多数 PLC 也允许用户输入一个出错程序，程序在看门狗定时器达到它的预设值时执行。你可以使用看门狗定时器和出错程序来完成时间延时的响应（如果你的 PLC 不提供时间延时中断)。程序可以包含写一个小的时间值到看门狗定时器的设定值的指令。在出错程序执行前，这将是延时时间。当然，延时不能超过当前扫描的结束，因为看门狗定时器在每次循环中被复位。出错程序必须包括出错和看门狗定时器的复位，这样中断程序能被返回，否则，PLC 将停止运行。

1. 定时中断和 ALLEN-BRADLEY PLC-5

PLC-5

PLC-5 能被配置成在定时间隔执行一个选择的定时中断（STI）程序文件。通过使用和配置处理器输入中断（PII）同样的处理器配置屏幕输入 STI 的配置数据。程序员仅仅需要输入程序文件号，它们包含 ISR 和以毫秒为单位的设定间隔时间。PLC-5 允许的间隔时间是从 1 到 32，767ms 之间。输入一个 0ms 的时间将禁止 STI。

示例

如果在 PLC-5 的处理器配置屏幕输入下列数据：

```
Selectable Timed Interrupt: prog file: 11 setpoint:1500ms
```

只要 PLC 进入运行模式，将打开一个定时器。当定时器到达 1.5s 时，PLC 将中断它正做的事情，并执行程序文件 11。当程序 11 结束后，PLC 将恢复被中断的程序或 I/O 扫描。STI 程序文件开始执行时，定时器复位，所以这个过程将每隔 1.5 秒重复执行。

STI 中断是最低优先级的 PLC-5 中断。STI ISR 程序文件将不会中断出错程序、处理器输入中断或先前开始的 STI 程序文件。STI 将等待并在高优先级的 ISR 结束后开始执行。如果 STI 被一个高优先级的中断所中断，那么它将在高优先级的中断 ISR 完成后重新执行。如果因为一个 STI 中断程序已经在运行，所以另一个 STI 不能够执行，PLC-5 将设置一个“STI 重叠”次要出错位（S：10/2）。

在 STI 程序文件中，块传输请求将不参加排队。取代的是，它们将在块传输指令后立即被执行。PLC 将等待，直到在 STI 程序中的下一条指令被执行前块传输指令完成时。（在 STI 中，连续的块传输不能被编程；PLC-5 将它们改变为非连续。）

PLC-5 的其他特征和 Allen-Bradley SLC 500 PLC 中的一样，将在本节中描述，在本节中我们讨论 Allen-Bradley PLC-5 和 SLC 500 的定时中断。

SLC 500

2. 定时中断和 ALLEN-BRADLEY SLC 500

SLC 500 能被配置为在定时间隔处执行可选定时中断（STI）程序文件，它和 PLC-5 的配置十分相似。通过使用与配置离散输入中断（DII）相同的处理器配置屏幕，输入 STI 的配置数据。程序员仅需要输入包含 ISR 的程序文件号和设定间隔时间。SLC 500 的 STI 间隔可以以 10ms 为单位从 1 增加到 255（或在一些 SLC 500 型号中以 1ms 增加）。输入时间 0 禁止 STI。

示例

如果在 SLC 500 的处理器配置屏幕输入下列数据：

```
Selectable Timed Interrupt: prog file: 11 setpoint:150 10ms
```

只要 PLC 进入运行模式，将打开一个定时器。当定时器到达 1.5s 时，PLC 将中断它正做的事情，同时执行程序文件 11。当程序 11 结束后，PLC 将恢复被中断的程序或 I/O 扫描。当 STI 程序文件开始执行时，定时器复位，所以这个过程每隔 1.5 秒重复执行。

STI 中断是最低优先级的中断，除了在 Allen-Bradley SLC 500 的术语中实际上叫做 I/O 中断的类型。因此 STI 程序文件将不会中断出错程序、离散输入中断或已经执行的 STI 程序文件，但是将中断 I/O 中断 ISR。另一方面，如果高优先级的出错或离散输入中断发生，STI ISR 将被中断。在高优先级的中断 ISR 完成之后，被高优先级中断延时或中断的 STI 将开始（或恢复）执行。（如果出错程序不清除引起它运行的任何出错，那么 PLC 将退出运行模式并且 STI 将被取消。STI 被保持仅在更高级别的 SLC 500 的出错程序中。）

SLC 500 PLC 与 PLC-5 相比，拥有一套更为完善的 STI 控制和状态位。一些位能被用户程序改变（例如，在一个取消任何被延时的 STI 中断的出错程序）。其他的位仅能被用户程序监控（例如，STI 能检查看它是否被延时和能被编程来使得操作变得不同）。在一些低级的 SLC 500 型号中并没有提供的这些位，包括：

1）STI 使能位（S：2/1），如果被清除，将禁止 STI 中断，但并不阻止定时器运行。

2）STI 执行位（S：2/2），它在一个 STI 的 ISR 运行时自动置位。

3）STI 未决位（S：2/0），它在 STI 延时时自动置位，因为它没有足够的优先权来中断

一个已经运行的 ISR，或因为 STI 使能位没有置位。

4）STI 丢失位（S：36/9）和次要出错溢出位（S：5/10），如果仍有一个 STI 未决，STI 被使能，以及定时器时间溢出时，两个位都将自动置位。如果在这时中断不被允许，那么仅仅溢出位置位。

5）STI 分辨率位（S：2/10），程序员对它置位来使定时器以 1-ms 为单位增计数而不是默认的以 10ms 为单位。

6）中断潜在位（S：33/8），使中断响应时间变短。如果当 STI 定时器值溢出时，这个位置位，那么当它结束正在操作的数据字时，PLC 将发生中断。如果这个位没有置位，那么 PLC 将结束正在执行的程序梯级，或结束读或写一个完整插槽的输入/输出数据，或结束在被中断前正在发送或接收的完整的数据包。

在下面的小节，我们描述 SLC 500 和 PLC-5 共同的 STI 特性。

3. 定时中断和 ALLEN-BRADLEY PLC-5 和 SLC 500

PLC-5 SLC 500

既然 Allen-Bradley PLC 在运行 STI 程序文件之前没有执行输入扫描，并且当 STI 程序文件结束后没有执行输出扫描，那么强烈推荐在 STI 程序中使用立即 I/O 指令。这会确保在输入采样和输出控制之间的间隔与 STI 程序文件的执行持续同样的时间。其他软件中断指令［临时结束（TND）、刷新（REF）和服务通信（SVC）］不能在 STI 程序文件中被执行。

STI 能被禁止或重新使能，可通过以下几种方法：

1）间隔时间能够被改变到 0 来禁止 STI 中断，或设置为一个正值（对于 PLC-5 超过 10ms 或对于 SLC 500 超过 2ms）来重新使能它们。当禁止了，再对它们重新使能，第一个间隔时间可能不正确。

2）设置 STI 文件序号为 0 来禁止 STI 中断，设置它为一个已经存在的程序文件号来重新使能 STI 中断。

3）使用中断禁止指令和中断允许指令。PLC-5 的用户中断禁止（UID）指令禁止所有中断，除了出错程序，并且用户中断使能（UIE）指令重新使能所有的中断。SLC 500 的可选择时间禁止（STD）指令仅仅通过锁定 STI 允许位禁止定时中断，并且可选择时间允许（STE）指令通过锁定相同的位重新使能它们。SLC 500 的 STI 定时器继续运行，所以一个 STI 中断可能在 STE 指令执行时执行。

为了强制所有的程序扫描在一致的时间间隔执行，PLC-5 或 SLC 500 能被配置成使得所有的主控程序（MCP）被禁止，以及配置 STI 程序文件在频率间隔运行。（如果没有 MCP 文件，你不能配置这些 PLC。如果你想试试，PLC 进入运行模式时自动地在配置中插入一个 MCP）。不像主控程序，在每个扫描中仅有一个可选定时中断程序能被执行，但是可以编写一个 STI 程序来调用其他程序文件。即使扫描循环不包括任何 MCP 文件，输入和输出扫描将在每次扫描循环中继续执行。

在 STI 的配置屏幕输入的配置数据字被存储在状态文件中[⊖]。它们也能被带有移动指令的用户编写的程序输入或改变为合适的状态字。SLC 500 也提供可选时间开始（STS）指令来专门改变 STI 配置。用户可以编写 PLC-5 或 SLC 500 程序，每当 PLC 进入运行模式时能

⊖ 在 PLC-5 和 SLC 500 中，程序文件都被存储在 S：31，设定值在 S：30。

建立 STI 配置来避免一些情况，例如用户可能意外地禁止 STI 中断。当程序运行时也能改变它本身的 STI 配置。（尽管紧跟一个改变了的间隔时间的第一个 STI 间隔将不正确）。一个依靠外部条件可以改变自身 STI 中断间隔和 STI 响应程序的程序如图 11-18 所示。

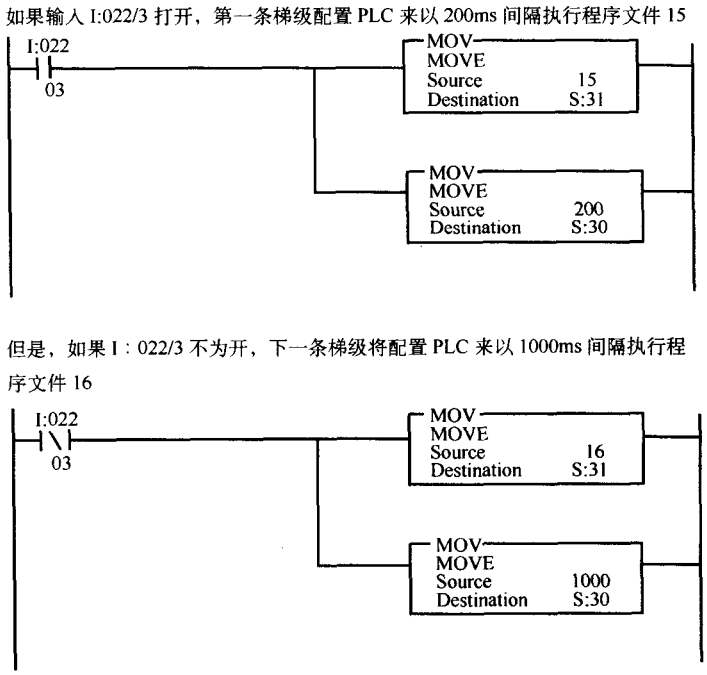


图 11-18 改变自身 STI 配置的 PLC-5 程序

可以编程使 PLC-5 和 SLC 500 PLC 在运行时改变看门狗定时器间隔。[⊖]如果看门狗定时器时间故意设置得比 PLC 扫描循环短的时间（包括程序扫描时间、I/O 扫描时间和通信时间），事实上，出错程序变成在程序扫描开始后的一个固定时间执行的定时中断例行程序。（出错程序和出错代码的更多的细节将在本章以后的部分中讲到。）

S5

4. Siemens STEP 5 的定时中断

不是所有的 S5 PLC 都提供定时中断。在提供定时中断的 PLC 中，响应定时中断与响应 I/O 中断有如下几个方面的区别：

1）定时中断是最低优先权的中断，所以它将不能中断响应过程中断（STEP 5 叫 I/O 中断）的中断服务程序（ISR），也不能中断一个已经运行了的定时中断 ISR。定时中断 ISR 将被推迟直到其他的 ISR 结束。另一方面，过程中断将中断正在运行的响应定时中断的 ISR。

2）定时中断比标准 I/O 扫描的优先级低，所以只有在 I/O 扫描结束后才能响应定时

⊖ 在 SLC 500，看门狗定时器间隔的设置可以通过在状态字 S：3的高字节（S：3H）写一个在 2 到 250 之间的值来实现，指示以 10ms 间隔为单位的时间（20 到 2500ms）。同一个状态字的低字节（S：3L）包含从用户程序的当前扫描开始的结束时间。在 PLC-5，以 ms 为单位的定时间隔的设定通过往 S：28 写一个 10 到 32 767 之间的值来实现。S：8包含从用户程序的当前扫描开始的结束时间，以 1ms 或 10ms 为单位，这依赖于 PLC-5 的型号。

中断。

3）在 S5-103U 中当一个定时中断响应时，它将做一个完整的输入扫描，复制输入模块的内容到中断过程映像输入表（中断 PII）。在中断 PII 中，直接访问寻址将导致 PLC 来读和写数据。（大型 S5 PLC 不做额外的输入和输出扫描。直接访问寻址引起大型 PLC 执行立即输入或输出到 I/O 模块。）

4）定时中断导致组织块 13（OB 013）运行（像 OB 002 一样，它将在 I/O 中断的事件中被调用）。OB013 包含到其他结构、程序或功能块的跳转。

一些大型 PLC 提供调用 OB 010、OB 011 和 OB 012 的附加定时中断。

5）在 S5-103U 中，直接访问寻址引起输出数据被写入中断 PIQ 表（并且同时写入普通 PIQ 表）。OB 013 结束后，一个短输出扫描在这些 PLC 中被执行，复制所有改变了的中断 PIQ 数据到输出模块。

6）程序员必须通过写一个值到保留系统字 97（RS 097）来定义定时中断间隔，可以直接由程序员完成，也可以使用编程软件数据屏幕，或在用户程序中完成，输入 0 和 255 之间的值。定时中断间隔将是以 10ms 为单位的数。如果没有值被输入，RS 097 将包含默认值 10，所以 OB 013 的定时调用每 100ms 发生一次。

提供附加定时中断的大型 PLC 需要在 RS 098 的值来设定 OB 012 的间隔时间，对于 OB 011 是 RS 099，对于 OB 010 是 RS 100。

通过输入值 0 到 RS 097，通过不为 OB013 输入一个程序，或通过执行 IA（中断禁止）指令，定时中断可以被禁止。如果一个定时中断已经发生但还没被 IA 指令禁止，那么它将在 RA 指令执行后被执行。

在用户程序中设置定时中断的一个好处是，无论 PLC 何时进入运行模式，都执行初始化程序一次。（OB 021 保留用作一个初始化程序，将在本章节中讨论。）图 11-19 的程序将设置 PLC 以精确的 5s 间隔运行 OB 013 中的程序。这个例子不显示 OB 013 做什么。

```
OB021
每次当 PLC 转换到运行模式时，这个组织块将运行一次
    JU FB001          ；在一个功能块中仅能写系统字一次
       Time_ Int



   BE
FB001
设定定时中断每 5s 发生一次，通过指定两次中断之间有多少 10ms 的间隔来完成
Name:  Time_ Int
    L KF+ 500         ；500×0.01s=5.00s
    T RS097           ；放入系统字的位置
    BE

OB013
任何作为 OB013 被输入的程序将以精确的 5s 间隔运行
```

图 11-19　配置 S5 PLC 每 5s 执行一个 ISR 的 STL 程序

为了强制 S5 PLC 以一致的间隔执行标准程序扫描循环，程序员可以通过向 RS097 写入数据来设定时间间隔，可以把主程序放进 OB 013，并且可以不输入任何程序进入任何默认

的主程序块，OB001 或 PB001。

在一个 S5 PLC 中，*扫描时间监控*（针对看门狗定时器 STEP 5 的名字）不能被使用来触发一个用户编写的出错程序。当扫描时间监控时间溢出时，PLC 进入停止模式。程序员可以访问扫描时间监控器，通过写一个 1 和 255 之间的数到 RS 096，根据在 0.01 和 2.55s 之间的扫描时间设定。在程序扫描中，如果 OB031 被调用，或者如果它包含至少一个 BE（块结束）指令，累计时间将被复位为 0，所以超出扫描时间监控器的时间就可以避免。

S7

5. Siemens STEP 7 的定时中断

高于 S7-312 IFM 的 Siemens S7 PLC 提供一个大的定时中断选项的扩展范围。除了在 STEP 7 中叫做*循环中断*的标准定时中断外，其他定时中断包括时间-天中断，它能被设定在一个预先设置的未来的日期和时间处发生一次，或在每天的同一时间，或甚至在每一分钟、小时、星期、月或年的相同时间发生一次；以及时间延时中断，它可被设定成在触发它的程序指令之后的一个预先设定的时间发生。这里也存在一个*循环时间出错*中断，有很多原因可以触发它，包括当 PLC 扫描循环超过设定在 1ms 和 6s 之间的循环时间（看门狗定时器）。在 S7-312 IFM 等级的 S7 PLC 或者更低型号不提供定时中断，但是它们能被配置成以一个特殊的循环速率执行它们的主组织块程序。

在 STEP 7 系统中，定时中断是最低优先级的中断，所以如果有更高优先级的 ISR 正在运行，响应定时中断的 ISR 的执行将被延时。一些 STEP 7 定时中断有比其他定时中断更高的优先级。每一个优先级将在以后讨论。S7 PLC 响应定时（或其他类型的）中断时，在堆栈中存储数据，所以当 ISR 结束时，中断程序能被恢复。

如果定时中断被使用，程序员必须输入中断服务程序（ISR）到特殊的组织块。如果任意定时中断组织块（除了循环中断）被调用，或者组织块没有被编写（带有至少一个块终端指令，如 BEU 或一个调用“STP”），出错程序 OB 85（如果存在）将运行，或者如果 OB 85 不存在，PLC 将立即进入停止模式。在下面部分，我们将告诉你如何配置和允许定时中断，以及对于每个类型的定时中断，哪个组织块需要你编程。

在用户程序中，“DIS _ INT”和“EN _ INT”系统函数调用能被用在任何地方以禁止和重新使能一些或所有中断类型，或在一个单独的程序扫描中，“DIS _ AINT”和“EN _ AINT”系统函数能被调用来从低优先级的中断中延时或停止延时更高优先级的中断。（在 STEP 7 关于 I/O 中断的部分中，这些系统函数将讨论）

S7 PLC 能被配置来以 1ms 和 60s 之间的固定间隔执行循环中断，开始于 PLC 转换到运行模式后当初始化程序（OB 100）结束执行时。在一些 S7 PLC 中，固定的间隔能被配置成带有一个最高到 60s 的相位失调，所以第一个时间间隔的开始将在 OB 100 结束后被延时一定的时间。循环中断有一个默认的优先级 12，STEP 7 定时中断中的最高优先级中断，但是在一些 S7 PLC 中，优先级能被改变为 0（以禁止循环中断）或改变到从 2 到 24 的优先级。

当 PLC 在运行模式时，配置循环中断将导致一个间隔定时器工作，发布中断请求信号，所以程序员将在 PLC 程序激活循环中断使之开始之前输入一个 ISR 程序（尽管如果一个循环中断 ISR 丢失时，PLC 不发生出错）。ISR 必须作为 OB 35 被输入[⊖]。当操作系统调用OB 35 时，它传递一些参数到 OB 35，包括一些指示中断是激活的和 OB 35 已经被调用的参数，当前

⊖ Siemens 保留 OB 30 到 OB 37 给将来的循环中断能力使用。

循环中断的配置数据（优先级，频率设定，相位偏移），以及 OB 35 开始的日期和时间。如果当 OB 35 仍就运行在一个先前的循环中断中时，循环中断定时器时间溢出，那么 PLC 将执行出错程序 OB 80（如果它存在），或者如果 OB 80 不存在，PLC 立即出错而进入停止模式。

最多有 8 个独立的时间-天中断能通过使用 STEP 7 编程软件或通过调用用户程序中的 STC 28（“SET _ TINT”）被配置。配置选项允许选择被配置作为时间-天中断的组织块的编号（OB10 到 OB17），第一次中断将发生的日期和时间，以及在时间-天中断将运行的间隔周期。间隔选项包括仅在特殊时间和日期中断一次，或在开始后的每一分钟、小时、天、星期、月或年重复。

一旦被配置，在用户程序中，时间-天中断能通过使用 STEP7 编程软件或通过调用 SFC 30（“ACT _ TINT”）被激活，指定哪一个配置好的时间-天中断被激活（OB 10 到 OB 17）。在激活后，操作系统将在指定时间调用时间-天 ISR。时间-天中断有一个默认的优先级 2，它仅比标准扫描循环的优先级 1 高，所以它将不能中断任何其他 ISR，但是将被保持直到没有更高级别中断请求剩余。如果一个时间-天中断组织块被调用但还没被编程，PLC 将执行出错程序 OB 85，如果 OB 85 不存在，PLC 将立即出错并进入停止模式。如果系统时钟向前移动到超越几个时间-天中断，并且因此使 OB 80 运行，STEP 7 操作系统将在 OB 80 结束后产生同样数量的时间-天中断。程序员可能希望利用再次调用 SET _ TINT 来取消这些仍在等待的中断调用。

图 11-20 显示了使用 SET _ TINT 和 ACT _ TINT 来配置和每小时初始化时间-天中断来调用 OB 11，在每小时后的 10 分钟，从 1997 年 12 月 1 号，星期五的上午 8:10 开始。在本例中变量 # start 已经被声明作为一个 DATE _ AND _ TIME 类型的变量。例子中的程序通过调用系统函数 FC 3（D _ TOD _ DT）分配一个值到 # start，这个系统函数把一个日期值常量（D#）和一个时间值常量（TOD#）转换到 DATE _ AND _ TIME 格式。被系统函数 SET _ TINT 和 ACT _ TINT 返回的 RET _ VAL 参数将包含显示配置失败或如果它们失败将激活时间-天中断的位类型。你的程序应该包括观察这些出错代码的指令。

```
网络 1：
    Call "D_ TOD_ DT"                    //函数 FC 3 定义了变量“start”
    IN1         :=D# 1997-1-12           //在电影“2001”中的 HAL 的生日
    IN2         :=TOD# 8: 10: 0. 0
    RET_ VAL    :=#Start

网络 2：
    Call SFC "SET_ TINT"                 //系统函数 SFC28，用来配置时间-天中断
                                         //传递到 SET _ TINT 的参数包括：
    OB_ NR      :=11                     //OB11 正被配置
    STD         :=#Start                 //第一个间隔的开始时间
    PERIOD      :=W#16#0401              //代码 0401 指示按小时中断
    RET_ VAL    :=MW 100                 //到 MW100 的返回值

网络 3：
    Call "ACT_ TINT"                     //系统函数 SFC 30 激活中断配置：
    OB_ NR      :=11                     //OB 11
    RET_ VAL    :=MW102                  //到 MW102 的 ACT _ TINT 的返回值
```

图 11-20　配置和开始时间-天中断的 STEP 7 STL 程序

SFC31（“QRY _ TINT”）能被调用以决定时间-天中断的状态，它带有一个参数指定哪个中断的状态被询问（OB 10 到 OB 17）。QRY _ TINT 返回的状态字包含显示时间-天中断是否被允许的位类型，前提条件是时间设定有效，中断被激活，中断已经发生，要求的组织块存在，以及一个测试功能有效㊀（时间-天中断在程序处于测试模式时被禁止）。查阅你的 STEP 7 编程手册来研究一下状态字代码和它们的含义。

在用户程序中系统函数 SFC 29（“CAN _ TINT”）能被调用以取消所有未来的对八个时间-天中断的任一个的调用。时间天-中断也能通过使用 S7 编程软件改变时间-天中断的优先级为 0 来禁止。如果在时间-天 ISR 开始之后的几秒时间内 SET _ TINT 被调用来初始化一个时间-天中断，或正在等待更高优先级的 ISR 结束，正在等待的中断（或正在运行的时间-天中断 ISR 的其余部分）将被取消。因此任何 ISR 能够包括 SET _ TINT 指令来取消正在等待的时间-天中断而不需要禁止未来的时间-天中断的运行。

在 STEP 7 程序中，如果控制对 SFC 32（“SRT _ DINT”）的调用的条件语句允许 SRT _ DINT 被调用，时间延时中断将被初始化。在 SRT _ DINT 已经被调用来开始时间延时中断后，PLC 继续运行它的用户程序，就像 SRT _ DINT 还没有被调用，但是 PLC 也开始运行一个定时器。如果 SRT _ DINT 被调用在延时溢出之前再次重启相同的时间延时中断，那么延时时间将复位并且延时将重新启动。当定时器达到它的预设值时，即使在发生了一些扫描循环以后，PLC 中断扫描循环来运行时间延时 ISR。如果在定时器溢出时，PLC 不在运行模式，那么只要 PLC 进入运行模式，时间延时 ISR 将执行。延时时间精确到 1ms 内。

时间延时中断拥有相对低的优先级 3，这意味着它仅能中断时间-天 ISR，但它能被其他任何中断类型中断。如果时间延时中断定时器到达预设值时，一个高优先级的 ISR 已经在运行了，那么时间延时 ISR 将等待高优先级的中断结束之后再开始运行。如果时间延时定时器到达预设值时，另外一个时间延时 ISR 正在运行，那么 OB 80 被调用，或如果 OB 80 不存在，PLC 将进入停止模式。如果定时器到达它的预设值时时间延时 ISR 不存在，那么 PLC 将调用 OB 85，如果 OB 85 不存在，就进入停止模式。

最多有四种不同的时间延时中断 ISR（OB 20 到 OB 23）存在。当 SRT _ DINT 被调用时，调用必须标明哪一个组织块正被调用，标明延时时间为毫秒级（1 到 60 000ms），并且必须提供一个标识符，在延时时间后组织块被调用时，这个标识符将被传递到组织块中。Siemens 把这个标识符参数字叫做一个符号。每次 SRT _ DINT 被调用时，延时定时器复位。如果 SRT _ DINT 在延时定时器溢出之前和第一次调用之后被调用，那么时间延时中断将发生，并从上一次 SRT _ DINT 被调用时开始。

通过系统函数 SFC 33（“CAN _ DINT”），时间延时中断能被取消。利用系统函数 SFC 34（“QRY _ DINT”）可以决定中断状态。这些调用必须指定它们取消和询问的是哪个 ISR（OB 20 到 OB 23）。被 QRY _ DINT 返回的状态字包含与 QRY _ TINT 函数返回的相同的信息（在上述时间-天中断的描述中已介绍）。

图 11-21 显示了一个程序如何有两个相同时间延时中断 ISR（OB 20）的不同调用。因

㊀ 如果程序员使用编程器在线调试程序，一个测试函数将被激活。调试时，PLC 需要花费更长的时间来执行每一个扫描循环，因为它必须将状态信息发送给编程器。

为每次被调用时，SRT _ DINT 复位延时时间，如果在 OB 20 开始之前两个调用的条件都变为真，被接收到的上次调用将被响应。在一个输入信号打开后，除了第一次扫描，每个调用将被跳过。每个调用初始化相同的时间延时中断（OB 20），使用不同的延时时间和提供不同的 16 位符号参数作为在 OB 20 开始时被传递到 OB 20 的标识符。OB 20 中的程序（没有显示）能包括对依赖于接收到的两个符号参数而作出不同响应的逻辑。当 I4.4 保持开时，CAN _ TINT 在每次扫描循环被调用，在它们发生前取消任何延时中断。

```
网络 1：
    A  I  4.2                          //如果输入 I4.2打开
    FP M  1.0
    JNB m001
    CALL "SRT_ DINT"                   //调用 SFC _ 32开始一个中断定时器
    OB_ NR     :=20                    //为了时间延时 OB20
    SDT        :=T#100MS               //带有100ms 的延时
    SIGN       :=W#16#1111             //设置十六进制标识符"1111"
    RET_ VAL   :=MW 100                //在 MW100存储 SRT _ TINT 回复
m001:NOP 0


网络2：
    A  I  4.3                          //和网络 I 一样，但被 I4.3触发
    FP M 1.1
    JNB m002
    CALL "SRT_ DINT"
    OB_ NR     :=20
    SDT        :=T#100MS               //延时时间比200ms 长
    SIGN       :=W#16#2222             //标识符是十六进制数"2222"
    RET_ VAL   :=MW 100
m002:NOP 0


网络3：
    A  I  4.4                          //但是如果输入 I4.4打开
    JNB m003
    CALL "CAN_ DINT"                   //调用 SFC _ 33，取消中断
    OB_ NR     :=20                    //为了时间延时 OB20
    RET_ VAL   :=MW 102                //存储 CAN TINT 的回复
m003:NOP 0
```

图 11-21　当两种情况之一被检查到，可能初始化一个时间延时响应的 STEP 7 程序

6. 定时中断和 OMRON CQM1

OMRON 的设计者好像喜欢使用定时中断，因为它们有几种方式可以使用：

1）间隔定时器中断被用来在一定时间间隔重复运行 ISR，或在延时之后运行一次。

2）在一个标准化的最小循环时间内，CQM1 能被配置成执行它的扫描循环，或当扫描时间超过循环监控时间，停止。

3）定时中断用在脉冲输出中。

4）高速定时器能被配置来在精确的时间间隔中断普通过程以更新定时器当前值。

5）PID 指令设置定时中断来读取输入值。PID 指令在 12 章描述。

CQM1 能被配置来执行三个间隔定时器中断。和其他 CQM1 中断一样，程序员必须在运行程序之前输入配置数据到存储器，并且程序必须在它们中断主程序之前包括允许定时中断的指令。输入中断和高速计数器中断 HSC1 和 HSC2 有最高的定时中断优先级。间隔定时器中断仅次于它们，间隔定时器 0 的优先级比间隔定时器 1 高，而间隔定时器 1 又比 2 高。

间隔定时器 0 被 CQM1 自动地用来对输出到输出模块的脉冲输出定时（在本节的以后部分讨论），并且在脉冲被计时时不能被使用。因此脉冲输出和间隔定时器 0 有相同的优先级。类似的，高速计数器 0（HSC0）使用中断定时器 2，CQM1 必须被配置来使能 HSC0 或中断定时器 2。如果选择 HSC0，它将取代中断定时器 2 而作为 CQM1 系统中最低优先级的中断。DM 6642 必须包含 00xx 来使能间隔定时器 2 或 O1xx 来使能高速计数器 HSC0。

DM 6636 到 DM 6638 能包含配置值，使 CQM1 在 ISR 被间隔定时器 0、1、2 分别调用时，执行一个部分输入扫描[⊖]。程序必须执行间隔定时器指令，STIM（69），来开始或停止间隔定时器中断，来设置定时器设定值，来显示哪个子例行程序作为 ISR 运行，并且选择中断模式（仅执行一次或重复模式）。STIM（69）通常使用差分形式，@STIM（69），所以当它的输入逻辑保持为真时，它不继续执行每个扫描。STIM（69）指令需要三个控制参数，C1、C2 和 C3。

1）如果 C1 是 000、001 或 002，间隔定时器中断 0、1、2 以一次翻转模式开始，这意味着 ISR 将在 STIM（69）指令执行后的一个延时时间时仅执行一次。003、004 或 005 的 C1 值将以时间表中断模式开始间隔定时器中断 0、1、2，这意味着 ISR 将从 STIM（69）指令执行时开始按一定时间间隔重复执行。C1＝006、007 或 008 复制间隔定时器 0、1、2 的当前值到数据存储器。C1＝009、010、011 停止间隔定时器 0、1、2 并取消任何等待执行的中断。

2）如果 C1 引起间隔定时器开始工作，那么 C2 为定时器设定值使用。C2 可能包含指示以毫秒为单位的设定值的常数（＃0000 到＃9999），或能包含存储设定值的两个数据字中的第一个的地址[⊜]。如果 C1 使 STIM（69）来读取间隔定时器当前值，C2 指示两个数据字的当前值的第一个将被写入的地址。

3）如果 C1 引起这条指令开始一个按时间表的定时器中断，那么 C3 必须包含要执行子例行程序的序号。如果 C1 引起间隔定时器当前值的读取，C3 包含 CQM1 将存储一个显示时间的数字的地址，转换成以 10 毫秒为单位的数，因为当前值是最后被更新的（定时器时间单位都大于 10 微秒）。

间隔定时器中断能通过执行带有 CC 参数 100 的 INT（89）指令而被屏蔽，或通过执行带有 CC＝200 的 INT（89）指令来解屏蔽。屏蔽不会停止定时器的运行，并且在解屏蔽之后，在屏蔽时被间隔定时中断请求的 ISR 将执行。

图 11-22 显示了 STIM（69）正被使用来开始两个定时间隔中断。ISR 子程序没有显示。

⊖ 最多 12 个输入模块可以被扫描，从 000 到 011，可以从任一模块开始。

⊜ 在 C2 中指定的地址必须包含一个指示设定值时间单位的数值的 BCD 数（0000 到 9999），第二个地址必须包含一个 BCD 数（0005 到 0320），指定以 10ms 为单位的每个时间单位的大小。可能的间隔时间范围是 0.5ms 到刚刚低于 320s。

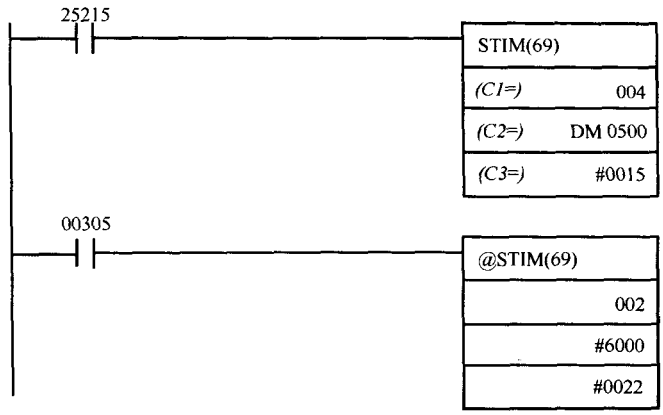


图 11-22　CQM1 的间隔定时器中断

1）在梯级 1，当 PLC 进入运行模式时，第一扫描位（SR 25215）使 STIM（69）运行一次。间隔定时器 1 被设置来引起时间表间隔定时器中断，每 25s 调用子程序 015，因为 DM 0500 包含 0025，设定值的时间单位的数值，以及下一个存储器，DM 0501，包含 1000，显示 1000ms（1s）的时间单位。

2）梯级 2 使用了 STIM（69）的差分形式，所以每次输入 IR 00305 打开时将执行一次。这条梯级在一次翻转模式下开始间隔定时器 2，所以子程序 022 将在 STIM（69）指令执行后的 6000ms 被调用一次。

CQM1 能被配置成带有一个标准最小循环时间，这样它在上一个扫描循环开始后的一个固定时间开始每次扫描循环。如果在 DM 6619 中，CQM1 已经被配置成带有一个非零的时间值（0001 到 9999ms），然后在用户程序的每次扫描后，在写到输出模块并且继续下一次扫描循环前，CQM1 将等待直到循环时间监控器到达那个时间值。这个值是最小扫描时间值。花费更长时间执行的扫描循环将总是在下一个扫描循环开始之前结束，所以程序员应该确信在 DM 6619 的时间比最长的扫描循环时间还长。（记住，被中断的扫描循环将比不被中断的扫描循环花费的时间长）。

在 DM 6618 中，如果任意扫描循环超出了循环监控时间的设定，那么 CQM1 将立即停止扫描，关闭所有输出，把一个出错代码放进出错日志，同时打开在 CPU 面板上的 ERR/ALM 和 POWER 指示器。循环监控定时中断没有像特意编程的定时中断那样的值，所以将在“出错程序和 OMRON CQM1”中描述。

CQM1 提供可编程的高速脉冲输出，它引起 CQM1 中断扫描循环来执行立即输出来开或关一个单独的输出触点。所有的 CQM1 PLC 允许程序员在被控制的每个输出模块中选择一个位，但 CQM1-CPU34-E 还允许通过 CPU 模块中的 15-pin 连接端口有额外的脉冲输出。

通过任意数字输出模块触点，任意 CQM1 PLC 能使用脉冲输出来输出 50%占空比循环脉冲[⊖]。程序员必须通过输入一个值（0000 到 0011）到 DM 6615 来指定输出模块，DM 6615 指示哪一个输出模块（100 到 111）包含控制触点，同时必须配置 CQM1 通过输入值 xx01 到 DM 6639 来使用直接输出。直接输出意味着当程序改变它们时，所有的输出值被写

⊖　50%占空比循环脉冲意味着每个循环的输出有一半为开。

入输出模块。(脉冲输出使用间隔定时器中断 0 来改变一个输出位的值。)

用户程序必须执行速度输出指令，SPED（64），来开始脉冲输出。SPED（64）必须包含一个 P 参数值，从 010 到 150 来分别选择位 01 到 15；一个 M 参数包含 000 来选择独立模式或 001 来选择连续模式；以及一个 F 参数包含一个在 0002 到 0100 之间的值对应在 20 到 1000Hz 之间的频率。如果连续模式被选择，那么脉冲输出将继续直到另一个 SPED（64）指令改变频率到 0Hz 或直到 INI（16）指令（在“I/O 中断和 OMRON CQM1”中介绍）被执行。如果独立模式被选择，那么设置脉冲指令，PULS（65），必须在 SPED（64）指令之前指示在停止前输出多少脉冲。[⊖]两个指令都应该以差分形式使用，@SPED（6）和@PULS（65），所以当它们不需要被执行时，就不会减慢循环的速度了，并且避免在每次扫描循环中重启脉冲输出定时器和计数器。

图 11-23 显示了一个 CQM1 程序，在这个程序中当输入 IR 00405 关闭时，输出脉冲通过输出模块 103（DM 6615 先前被程序员设置为 0003）的触点 2（P=020）来持续地（M=001）处于 60Hz（F=#0006）。(在运行程序前，0001 被输入到 DM 6639 来指示直接输出。）当输入 IR 00405 打开时，执行 INI（61）来停止 60Hz 输出，然后 75 000 个脉冲通过相同输出模块的触点 11 以 250Hz 输出。(DM 0200 包含 5000，DM 0201 包含 0007，在执行梯级 2 之前。)

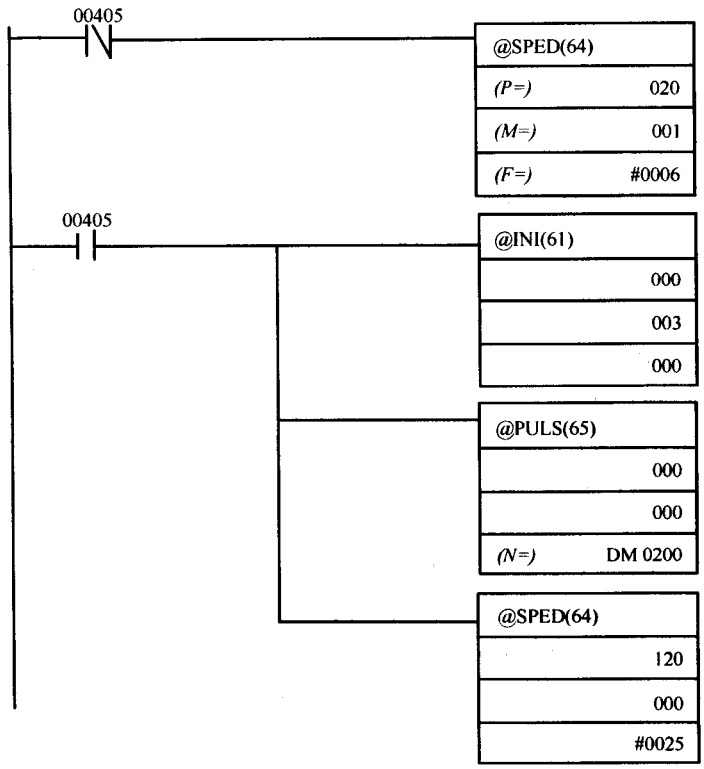


图 11-23 使用脉冲输出的 CQM1 程序

⊖ PULS（65）有 P 参数和 C 参数，对于通过输出模块的脉冲，两者都为 0；N 参数，指定包含脉冲的 8 数字 BCD 数的两个数据存储器的第一个的地址。第一个地址包含四个低 BCD 数字，第二个地址包含四个高 BCD 数字。最多允许 16 777 215 个脉冲。

在 CQM1 的 CPU43-E 的版本中，当脉冲通过输出模块输出时，脉冲可以通过 CPU 端口 1 和 2 输出。还有更多的选项，在这里只介绍几个常用的，请查阅你的 CQM1 编程手册。一些差异包括：

1）输出到端口 1 或 2 的脉冲输出需要高速计数器 1 或 2（分别），所以在它们处在脉冲输出时，HSC1 和 HSC2 不能用作计数器。

2）SPED（64）和 PULS（65）参数指示通过 CQM1-CPU43-E 端口输出脉冲。三个脉冲输出能独立地同时输出。SPED（64）默认是连续脉冲输出除非 PULS（65）已经被用来指示一个到输出的脉冲数。

3）在 PLC 进入运行模式前，DM 6643 的值指示端口 1 和 2 是否将输出标准的 50％占空比的循环脉冲或占空比可调脉冲（脉宽调制）。

4）如果选择标准脉冲输出，那么被选择的端口在两个触点输出相同的脉冲，其中一个触点的脉冲超过另一个 90 度（适合驱动步进电机）。PULS（65）被用来指定哪个输出信号超前另一个（控制步进电机的旋转方向），也能指定脉冲频率应该在停止前减少到 0Hz。PLS2（-）能被用来代替 PULS（65）和 SPED（64），当它们开始时以一个可控制的速率来逐渐增大脉冲频率，从 0 到指定的最大值，当它们停止时，返回到 0Hz。在被其他命令启动后，ACC（-）能被用来上升或下降脉冲频率。这些特性适合于用来控制步进电机的速度和加速率。PLS2（-）和 ACC（-）需要一个高速计数器，仅在 DM 6611 被设置为 0001 时使用，保留 HSC0 给脉冲输出使用。

5）如果占空比可调脉冲输出被选择，那么脉冲在每个端口仅通过一个触点输出。它们由 PWM（-）指令开始，他们所带的参数指示端口，选择三种频率中的一个，并且指示了在 1％到 99％之间的占空比。占空比可调脉冲输出可以用来输出模拟直流量。

6）每次循环中，已经输出的脉冲数目的计数将被更新在端口 1 的 SR 236 和 SR 237 中或端口 2 的 SR 238 和 SR239 中。几个状态位在 AR 04 和 05（端口 1）和 AR 04 和 06（端口 2）处被更新。

7）INI（61）用来在端口 1 或 2 停止脉冲输出。PRV（62）用来读取脉冲输出的比在 AR04 到 AR06 更多的立即更新的状态位。

当前值（PV）需要更新时，高速定时器（TC 000 到 TC 015）中断扫描循环，而不是等待直到每次扫描循环结束时。因为 ORMON 不把高速定时器描述为中断，所以也没有告诉我们它们的优先级，但是使用手册警告我们高速定时器能降低其他中断操作的等级，所以它们必须有相当高的优先级。当 DM 6629 包含值 00xx 时，所有 16 个高速定时器被使能。用户能指定有多少定时器使用中断，通过输入值 01xx 到 DM 6629，这里的 xx 是在 00 到 15 之间的值，指示最多有多少用于中断 PV 更新使能的高速定时器。程序员可能希望输入一个值来限制正在运行的高速定时器的数目，这样它们不会降低其他中断的等级（特别是如果脉冲输入的频率要求超过 500Hz 时）。

高速定时器不同于标准定时器的地方在于高速定时器指令，TIMH（15），被使用而不是标准定时器指令，TIM. TIMH（15）的设定值（SV）以 0.01s（从 00.00 到 99.99）为单位输入，而不是 0.1s。在扫描时间超过 0.01s 时，使用带中断的高速定时器更新它的 PV 值。但是为一个定时器使用的 TIMH（15）指令不需要中断它的 PV 值的更新，这将导致扫描时间超过 0.01s 时，定时器的 PV 值不准确。和其他定时器一样，高速定时器预设值（PV）

从它的 SV 值减少到 0，此时它的完成标志打开，但是不像其他的普通定时器，在程序扫描的任意时刻完成标志可以被打开。和其他定时器一样，如果在一个跳转开始（JMP（04））和带有跳转 00 的跳转结束（JME（05））之间有 TIMH（15）指令，高速定时器将停止。

11.4.4 出错程序中断

当 PLC 检测到存在一个严重的问题时，自动执行一个出错程序。出错程序具有最高的中断优先级，所以无论 PLC 正做什么事情都将被中断来允许出错程序执行。早期的 PLC 发现一个严重的出错时，只是简单地进入出错模式。PLC 停止运行，打开 CPU 面板上的出错 LED 灯，并且把所有的输出都关闭（或可能冻结在它们的最后状态）。大多数 PLC 进入出错模式时保存一个出错代码在存储器中。在 15 章中我们描述了如何使用出错代码来找到为什么 PLC 发生出错。

当一个可检测到的出错情况被识别时，好的 PLC 能被配置来执行用户编写的出错例行程序。出错程序可能被编写来纠正一些类型的问题，给操作者发送消息，或仅仅以一个有秩序的方式关闭 PLC。在出错被出错程序清除后，PLC 将重新执行检测到出错时被中断的程序。就像上面描述的那样，如果错误不能被识别或纠正，所有的 PLC 都进入出错模式。

能导致出错程序运行的典型的出错包括以下一些事情，例如当看门狗定时期达到它的预设值（指示程序扫描循环花费太长时间）时试图运行的程序文件不存在，与 I/O 模块或其他的 CPU 通信时检测到失败，或后备电池电压过低。一些 PLC 也允许程序员定义和使用他们自己的用户定义的出错。用户编写的出错程序能作为一种软件中断类型使用，仅当程序包含一个引起用户定义的出错的指令时运行。

PLC-5 SLC 500

1. 出错程序和 ALLEN-BRADLEY PLC

PLC-5 和 SLC 500 都被预编程来检测一些主要和次要出错类型。（附录 H 包含出错列表）当检测到一个出错时：

1）PLC 中断任何它正运行的程序。出错中断具有最高优先级，所以在一个出错发生时，ISR 也将被中断。

2）一个（或多个）出错位在适当的状态字中置位，并且一个出错代码被放入另外一个状态字。程序文件号和当出错发生时运行的程序的梯级号被记录在其他状态字。状态字能被显示在编程软件的处理器状态屏幕上。

如果出错是次要出错类型，PLC-5 置位 S：10 或 S：17 中的出错位，或者如果出错是主要类型，置位 S：11 中的出错位。保存主出错的出错码到 S：12。出错发生时，程序文件号和正在运行的梯级号被分别放入状态字 S：13 和 S：14。

SLC 500 对于次要出错对 S：5中的位置位，或对于任何主要出错，单独对 S：1/13 置位。它存储主要出错代码到 S：6。出错发生时，程序文件号和正在运行的梯级号被分别放入状态字 S：20 和 S：21。

3）如果出错是次要出错，被中断的程序立即恢复，而不运行出错例行程序。在 15 章中，我们描述了将被置位的次要出错位以及程序如何使用它们。

4）如果出错是一个可恢复的主要出错（下面将描述），PLC 试图运行出错例行程序。

■ 如果程序员从配置屏幕或通过程序中的指令，已经输入一个出错例行程序文件号到

地址 S：29。出错程序可以调用其他程序文件。

- 如果主要出错位在出错程序结束执行前被程序清除，那么 PLC 将重新运行被中断的程序文件，从出错发生时正运行的指令的下一条开始。

5）如果：

- 主要出错是一个不可恢复的出错类型，或者
- 没有找到出错例行程序文件号，或者
- 出错程序执行时出现另一个主要出错，或者
- 出错程序结束执行时没有清除所有的主要出错位，之后 PLC 将直接进入出错模式。

大多数能引起出错程序执行的可恢复的主要出错是由编程出错引起的。这些能被编程出错引起的可恢复出错包括：在一个程序扫描结束前看门狗定时器时间结束；程序试图执行一个到不存在的标签的跳转或一个到不存在的梯形图程序文件的跳转（包括丢失的 ISR 文件）；在间接寻址中使用一个无效值作为计数器的预设值，或作为其他某种特定的应用；或者子程序的太多嵌套被使用。一个可恢复的主要出错也可能由非编程错误引起，例如存储器失效，或有时由外围设备的出错引起。

PLC-5 和 SLC 500 允许程序生成用户定义的可恢复的主要出错，这种出错以一个出错代码结束。

在 PLC-5 中，如果用户程序包括跳转到出错程序文件的子程序的跳转指令，出错程序能被强制执行。当然，出错程序应该被编程来响应引起 JSR 执行的情况，但是出错程序会作为中断而非子程序被调用。一个差别是 PLC 不能传递普通参数到出错程序。然而，JSR 指令只能带有一个单独的参数：一个 0 和 9 之间的数，它是 PLC 的操作系统在调用出错程序时作为出错代码写入状态字 S：12。（出错程序不需要 SBR 指令来接收这个值到 S：12。）Allen-Bradley 特意不指定 PLC-5 出错代码 0 到 9 的意义，所以程序员能使用这些数字作为自己的出错识别代码。当 JSR 指令初始化出错程序中断时，PLC-5 也置位一个主要出错位（S：11/7），该位指示可修复主要出错是被 JSR 指令引起的。和其他任何主要出错位一样，出错程序必须在结束前或 PLC 进入出错模式前清除这个位。

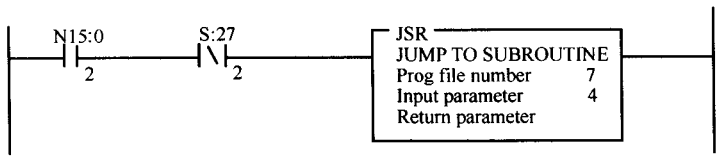
在 SLC 500 程序中，为了强制出错程序来运行用户程序，必须置位 S：1/13（可恢复的用户出错位）。如果程序员要使用一个自己生成的出错代码，程序必须在对位 S：1/13 置位之前写出错代码到 S：6。Allen-Bradley 推荐用户使用（十六进制）代码 FF00 或 FF0F，这意味着 Allen-Bradley 在将来的 SLC 500 型号中避免使用这些出错代码。出错程序将在结束前或 PLC 将进入出错模式之前复位位 S：1/13。

图 11-24 显示了一个包括了 JSR 指令的 PLC-5 程序，当一个不正常的可检测到的出错情况发生时引起出错程序运行。（本例中是一个框架故障）本例中的出错程序检查出错代码来确定为什么程序使它执行。（它可能已经被 Allen-Bradley 定义的出错引起）然后出错程序禁止出错框架和清除包含主要出错位的数据字。出错程序结束时，PLC 将重新运行在梯级 18 的 MCP 文件 2，即引起出错中断的梯级的下一条梯级。（记住，这仅仅是一个例子。在实际应用中，仅禁止出错框架是不够的。出错程序至少也将通知维护和重新使能框架。）

文件：2
MCP 文件的一部分

梯级：17

这个梯级调用文件 7，出错程序，如果在 S:27/2（如果框架 2 被禁止，框架控制位将工作）关闭时整数位 N15:0/2（一个 I/O 框架状态位指示远程框架 2 的出错）打开



文件：7

出错程序。PLC 的出错文件状态字，S:29，包含数字 7，配置程序 7 作为出错程序

梯级：1

这条梯级（在一个更长的出错程序的一个梯级）检查出错代码是否是用户定义的出错 4。如果是，梯级清除主要出错状态字（S:11）并把框架 2 的I/O 禁止位打开，这样 PLC 将停止扫描出错的远程 I/O 框架。

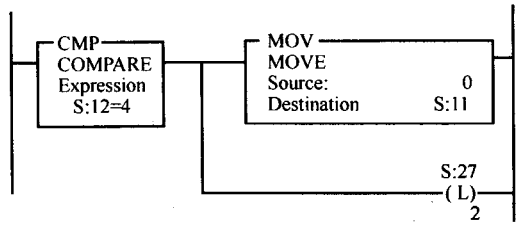


图 11-24 用户定义的 Allen-Bradley 出错应用

每次在电源关闭和重新打开后 PLC 进入运行模式时，PLC-5 和 SLC 500 也能被配置来运行一个出错程序一次。出错程序作为初始化例行程序使用，将在关于初始化中断的部分讨论。在故障检修一章，我们将描述如何避免看门狗定时器时间到期而引起中断。

S5

2. Siemens STEP 5 的出错程序

当发现出错时，S5 PLC 的操作系统自动地中断运行的程序，对存储器的 ISTACK 区域中的出错位置位，然后可能调用一个组织块（OB）作为出错程序。S5 PLC 仅提供几个作为出错程序的 OB，所以对于大多数出错，PLC 将进入停止模式。操作员必须找到和纠正出错，见 15 章描述。确实存在的结构块能被编程来纠正出错和允许中断程序的恢复，或只是简单地影响控制 PLC 关闭。

STEP 5 提供停止指令（STS）来使 PLC 像发生出错一样反应；提供另一个停止指令，STD，允许在引起 PLC 发生出错之前，完成当前的扫描循环。在调试程序期间，STS 和 STD 是有用的，在 15 章讨论。

被调用来响应出错的组织块中的程序，可能包括检查 ISTACK 中的出错识别位状态的指令，并试图纠正出错，尽管在程序中不是所有的 ISTACK 数据都可以访问。如果问题不能被解决和/或在执行一个关闭过程之后，OB 甚至可能包括 STS 或 STP 指令以使 PLC 进入停止模式。如果一个出错发生并且响应的组织块没有被编程，那么 PLC（通常）将进入停止模式。作为用户可编程的出错程序而提供的 OB 包括：

- OB 34，被调用来响应一个电池失效。如果它不存在，PLC 不发生出错。
- OB 23，如果程序中使用了直接访问寻址但被寻址模块不响应时，将被调用。
- OB 24，如果一个模块在 I/O 扫描时不响应，或处理器之间的通信失败时，将被调用。
- OB 27，如果功能块需要的形式参数和功能块调用中的实际参数不匹配，将被调用。
- OB 32，如果一个程序试图访问没有被调用的数据块，一个不存在的数据字，或如果程序试图在数据块中创建新数据字但没有足够的 PLC 存储器时，将被调用。

在故障检修一章，我们将描述如何避免由于看门狗定时器的时间到期而引起的中断。

3. Siemens STEP 7 的出错程序

S7

比 S7 312 IFM 更高型号的 Siemens S7 PLC 提供几个组织块（OB）来作为出错程序，每一个被不同类型的 PLC 出错触发。Siemens 定义的出错属于两个不同类型中的一个。同步错误出错由程序中的错误引起（所以它们的时间设定和程序扫描时间同步）；异步错误出错由设备失败[⊖]引起。在这两种情况下，当错误发生时 PLC 存储关于出错的信息到诊断缓存，如果错误在 I/O 模块中发生，存储出错信息到诊断状态表。被存储的信息包括一个描述出错的“事件 ID”错误代码、出错发生的日期和时间，以及（如果 PLC 已经被配置来包括扩展的诊断输入）被调用来响应出错的组织块的编号（参考你的手册，了解事件 ID 代码的意义）。

和 S7 312 IFM 级别相同或更低的 S7 PLC 不含有执行响应出错的组织块。如果一个致命出错发生，这些 PLC 仅存储诊断数据到诊断缓存，然后进入出错模式。如果在出错信息存储后，被调用的出错响应组织块不存在，那么 PLC（在大多数情况下）直接进入停止模式。如果组织块（OB）已经被编程，OB 以 BEC 或 BEU 指令结束，使被中断的程序恢复（当然在采取一些措施来纠正出错后），或者可能调用 SFC 46（“STP”）来使 PLC 进入停止模式（或许在执行一个关闭过程后）。STEP 7 系统为出错响应组织块提供了几个选项来确定出错的原因，这样它能在被中断的程序恢复之前被纠正。

1）当操作系统调用组织块时，它将传递几个参数到 OB，包括相同的出错事件 ID 代码，中断调用的时间和日期，以及被写入诊断缓存的其他信息。OB 能包括指令来评估这个信息使得它能适当地被响应。

2）系统状态表信息，包括比诊断缓存里更多的信息，能够被 OB 重新获得，通过调用带有指示什么数据的子列表被需要的参数的 SFC 51（“RDSYSST”）。看 15 章有关系统状态列表的更多信息。

图 11-25 显示了一个简单的 STEP 7 STL 程序，它能引起一个定时器寻址错误，以及一个在 OB 121 的出错响应程序，这个出错响应程序在同步错误发生时（例如定时器寻址错误）被调用。OB 121 中的程序检查事件 ID 代码，这个代码在 OB 121 被调用时作为一个来

⊖ 在 STEP 7 文件的某些位置，Siemens 将 I/O 中断描述为出错程序的某些类型。他们认为 I/O 中断经常被用来检测和响应指示控制过程出错的报警信号。在本书中 Siemens 的过程中断 ISR 并不归为出错程序。

自操作系统的参数被接收，如果是定时器寻址错误，程序将纠正这个错误，然后返回到被中断的程序。如果错误是其他任何类型的错误，那么 OB 121 将使 PLC 进入停止模式。

```
OB1
  这个程序包括梯级 25，它偶尔会引起一个同步错误。

网络 25：

     A  I 4.2                    //引起一个脉冲定时器运行
     SP T [MW 10]                //来自存储器地址 MW 10 中的定时器编号，
                                 //可能包含一个不能被定时器使用的数字

OB 121
  当发生同步错误时，组织块被调用

网络 1：

     L "OB_121_SW_FLT"           //读取操作系统提供的事件 ID 的低字节
     L b#16#27                   //并把它与一个对应无效计数器地址的已知
     <>1                         //事件 ID 相比较
     JC m001                     //如果不同，则跳到调用 STOP

     L w#16#5
     T MW 10                     //否则，把一个有效的定时器地址放入 MW10
                                 //使得下一次定时器工作

     BEU                         //然后返回到被中断的程序
m001:CALL  "STP"                 //SFC46，STP，让 PLC 进入停止模式
     BE
```

图 11-25 STEP 7 出错程序来纠正一个特殊类型的误差，或停止 PLC

同步错误使 PLC 的操作系统调用 OB 121 或 OB 122，这取决于程序运行时出现的错误类型：

- OB 121，“编程错误” OB，被调用来响应错误，例如调用一个不存在的函数、功能块或数据块；试图使用没有被分配的或指定给不同数据类型的数据存储器；或试图使用无效的数据值。
- OB 122，“模块存储错误” OB，当 PLC 正努力读取或写入一个 I/O 模块时，如果一个误差发生，则调用。

OB 121 和 OB 122 本身没有优先级！它们具有和它们中断的程序相同的优先级。（这是惟一的情况，S7 PLC 允许程序被具有相同优先级的 OB 中断。）因此 OB 121 和 OB 122 被认为是它们中断的程序的一部分。事实上，当 OB 121 和 OB 122 被调用时，它们不会使 PLC 在中断前保存微处理器寄存器中的值。如果 OB 121 或 OB 122 改变微处理器寄存器的内容，原始内容将丢失[⊖]。这样使得 OB 121 和 OB 122 可以纠正出错的工作数据，因为引起中断的

⊖ OB 121 或 OB 122 可以改变的。微处理器寄存器包括累加器，在累加器中保存加载指令获得的值；状态寄存器，保存在微处理器执行指令时被置位和复位的位；地址寄存器，保存微处理器正在使用的数据的指针；数据块寄存器，保存使用的数据块的地址。某些寄存器，包括那些指向本地数据堆栈，块堆栈，中断堆栈的顶部的寄存器，不会被 OB 121 或 OB 122 影响。

数据可能在微处理器寄存器中。程序员应该注意被中断的程序在恢复时所需要的好的数据没有被改变，或许可以通过保存受影响的微处理器寄存器中原始值，并在结束前恢复它们。STEP7 提供一个系统函数，SFC 44（“REPL _ VAL”），它允许累加器 1（在这里经常可以发现引起同步错误的坏的数据值）的内容被改变而不影响累计器 2 的内容。如果 REPL _ VAL 被调用，一个作为调用的参数的数据字被放入累计器 1 中。

不像其他类型的中断，OB 121 和 OB 122 不能通过调用系统函数 DIS _ INT、EN _ INT、DIS _ AINT 或 EN _ AINT（已在 Siemens STEP 7 中的 I/O 中断部分描述）禁止或重新使能。但是，它们能被其他系统函数来禁止或重新使能：

SFC 36（“MSK _ FLT”）能被调用来使程序开始屏蔽（忽略）选择的同步误差的类型。程序员必须为调用 MSK _ FLT 提供两个参数。一个参数是 32 位的编程错误滤波器，另一个是 32 位的访问错误滤波器。设定一个（或多个）这些 64 位将禁止一个（或多个）同步错误的指定类型引起编程错误程序（OB 121）或存储错误程序（OB 122）的运行。在 MSK _ FLT 对一个滤波器位置位后，它保持为开再次调用 MSK _ FLT，带有一个相同位为 0 的新参数，不会将那一位复位为 0，并且不能重新使能那一类型的同步错误。MSK _ FLT 返回两个参数，指示哪个位仍被置位。没有 64 种不同的同步错误的原因，所以某些位没有用。在返回参数中那些位总是 1。察看 STEP 7 的使用手册，了解每个滤波器位的用途。当被屏蔽时，同步错误仍将使 PLC 在错误缓存中记录一个事件 ID 错误代码。诊断缓存仍被使用。

SFC 37（“DMSK _ FLT”）必须被调用来允许一个程序来解除（恢复响应）被 MSK _ FLT 屏蔽的同步错误，这样当这些错误再次发生时，OB 121 或 OB 122 将运行。和 MSK _ FLT 一样，CALL DMSK _ FLT 指令必须带有编程错误滤波器参数和访问错误滤波器参数。如果掩模中的一个位是 1，它能重新允许被 MSK _ FLT 禁止的同步出错类型。（是的，在 MSK _ FLT 使用一个 1 来屏蔽，然后在 DMSK _ FLT 使用一个 1 来解除屏蔽。）DMSK _ FLT 返回和 MSK _ FLT 相同的参数。在使用一个 1 来解除屏蔽同步错误类型后，被返回的参数在相同位的位置上应该有一个 0 来显示它被解除屏蔽。

SFC 38（“READ _ ERR”）能被调用来检查当错误由于被屏蔽而不能被响应时，放置在误差缓冲区的事件 ID 错误代码。对于 READ _ ERR 的调用，与调用 MSK _ FLT 或 DMSK _ FLT 程序一样，必须提供编程错误滤波器和访问错误滤波器参数，除了在对应参数位的位置上是 1 来指示程序员要 READ _ ERR 返回误差缓存中的事件 ID 代码。READ _ ERR 为编程错误事件 ID 码返回一个 32 位的参数，为访问错误事件 ID 码返回另一个 32 位的参数。在这两个 32 位返回参数中的 3 个 16 位字保留给特殊类型的事件 ID 代码，第四个字（访问错误返回参数的一半）没有使用。READ _ ERR 在每次调用时可以返回 3 个 16 位的事件 ID 代码，或者通过几次调用 READ _ ERR 来返回所有的错误代码。举例来说，如果 5 个访问错误事件 ID 代码在错误缓存，那么需要对 READ _ ERR 调用 5 次来返回这 5 个代码。每次 READ _ ERR 返回一个事件 ID 代码，它删除来自错误缓存的代码，所以当 READ _ ARR 返回全 0 时，它已经返回所有的请求类型的事件 ID 代码。

图 11-26 包含一个例程，有条件地屏蔽两种类型的同步错误（坏的定时器数和坏的 BCD 数字），这样它们不能引起中断。网络 2 有条件地解屏蔽一个错误类型（坏的定时器数）。当中断被禁止，网络 3 为每次坏定时器数发生时读取错误代码，并且每次发生时运行 FB13 一次。

任何异步错误使 S7 PLC 调用一个 OB 80 和 OB 87 之间的组织块，这取决于错误的

```
网络 1:
    A  I  4.2                          // 如果输入 I4.2 开
    FP M 1.0
    JNB m001                           // 调用 SFC 36 来禁止误差
    CALL "MSK_FLT"                     // 当坏的定时器数被使用
    PRGFLT_SET_MSK:=DW#16#42           // 以及当非 BCD 数被当作 BCD 数时
                                       // 但没有访问错误
    ACCFLT_SET_MSK:=DW#16#0
    RET_VAL        :=MW100             // 存储 MSK_FLT 的错误信息
                                       // (本例中没有评估,
                                       // 但一个真实的程序应该评估)
    PRGFLT_MASKED:=MD102               // 存储显示哪一个错误正被屏蔽的返回值
    ACCFLT_MASKED:=MD106
m001:NOP 0
网络 2:
    A  I  4.2                          // 如果输入 I4.2 返回关
    FN M 1.1
    JNB m002                           // 调用 SFC37 来重新使能
    CALL "DMSK_FLT"                    // 只有坏定时器数同步错误类型
    PRGFLT_RESET_MSK:=DW#16#40
    ACCFLT_RESET_MSK:=DW#16#0
    RET_VAL             :=MW98         // 存储 DMSK_FLT 的错误信息
                                       // (这个例子中没有评估,
                                       // 但一个真实的程序中应该评估)
    PRGFLT_MASKED     :=MD102          // 存储显示哪一个错误正被屏蔽的返回值
    ACCFLT_MASKED     :=MD106
m002:NOP 0
网络 3:
    A  I  4.2                          // 如果输入 I4.2 已经关闭
       Fn M 1.2
       JNB m003                        // 重新从错误缓存中获得代码
m005:Call "READ_ERR"                   // 但仅为坏定时器数
    PRGFLT_QUERY      :=DW#16#40
    ACCFLT_QUERY      :=DW#16#0
    RET_VAL           :=MW96           // 存储 READ_ERR 的错误信息
    PRGFLT_CLR        :=MD110          // 存储包含来自错误缓存的
    ACCFLT_CLR        :=MD104          // 事件 ID 代码的返回值
    L MD110                            // 比较返回值和一个坏定时器数的事件 ID
    L W#16#15260000                    // 如果不等于
    <>1                                // 跳出此循环
    JC m004                            // 否则调用一个功能块一次,
    CALL FB15,DB10                     // 并再次返回去读取误差缓存
    JU m005
m004:NOP 0
```

图 11-26 屏蔽和解屏蔽被选择的同步中断和读取错误缓存的 STEP 7 程序

类型:

- OB 80,“循环时间出错”OB,被调用来响应一个定时误差,例如因为以前的定时中断尚未结束,而导致一个定时器中断不能运行,或看门狗定时器溢出。
- OB 81,“电源出错”OB,响应电源失效。如果它在被调用时不存在,那么这将是惟一的不会使 PLC 进入停止模式的出错例行程序。
- OB 82,“诊断中断”OB,响应在 I/O 模块被配置允许诊断中断功能的条件下,I/O 模块失效的情况。

诊断中断能通过使用 STEP 7 编程软件来使能,或通过执行一个用户程序中的 CALL WR _ PARAM 指令来发送新的配置数据到 I/O 模块。能够产生诊断中断的 I/O 模块也提供描述出错的一套参数值。当 OB 82 被调用时,它提供参数识别失效的 I/O 模块。OB 82 包括一个 CALL RD _ PARAM 指令来从失效的模块中读取出错识别参数,以识别为什么发生出

错。(查阅 STEP7 手册中的参数列表。诊断中断可能包括硬件中断)

- OB 83 运行，如果一个 I/O 模块被移走或插入不正确的模块。
- OB 84 运行，当出现 MPI[⊖]错误时，或如果存储器卡被取出或替换时
- OB 85，“OB 没有被加载出错” OB，其调用条件是，如果程序试图运行一个不存在的组织块，或当更新过程映像输入（PII）或过程映像输出（PIQ）表时，发生访问外围模块的错误。
- OB 86 运行，如果在分散电源系统中检测到出错，或和远程 I/O 框架进行通信时出错。
- OB 87，“通信出错”，在局域网里其他 PLC 共享的数据里检测到错误。

STEP 7 系统中异步错误具有最高优先级。OB 80 到 OB 87 有优先级 26。如果当异步错误被检测到时正在运行 OB100（初始化程序），那么异步误差 OB 被临时指定为优先级 28，这样它能中断优先级 27 的 OB 100。异步错误 OB 不能被中断。如果当 OB 80 和 OB 87 运行时，发生其他的异步错误，那么它们响应的 OB 将在当前运行的 OB 结束后被调用。

异步误差出错程序可以通过调用 DIS _ INT 来禁止，在这个中断请求被忽视之后，或者可以通过调用 EN _ INT 而重新使能。如果 DIS _ AINT 已经被调用，那么出错程序也能被延时直到程序扫描结束，然后允许再次开始，前提是 EN _ AINT 在程序扫描结束前被调用。在本章讨论 Simens STEP 7 的 I/O 中断的小节中解释了这些标准函数的使用。

3. 出错程序和 ORMON CQM1

CQM1

当检测到一个错误时，CQM1 不执行用户编写的出错程序。当一个出错发生时，CQM1 存储一个错误代码到 FAL 存储器区域，存储错误代码和发生的时间到错误日志存储区（如果错误日志特性被允许)[⊜]，并且为了更进一步地描述错误，可能打开存储器中的状态位。仅有三个两数字的 BCD 错误码能存储到 FAL 存储器区域。最近存储到 FAL 存储器区域的错误代码也被存储到 SR 253 的低字节。错误日志从 DM 6569 开始，扩展到 DM 6599。DM 6569 包含一个指向必须被放进日志的下一个入口的指针，跟着是最多 10 个错误的记录，每个错误的记录有 3 个字。每一个记录包含一个错误分类（致命或不致命）、错误代码和发生的时间、日期。CQM1 错误代码和状态位在图 15-7 和图 15-8 中。

对于致命错误，在存储错误信息后，CQM1 立即停止，关闭所有的输出，打开 CPU 模块上的 ERR/ALM 指示器。在 15 章中，我们描述了操作员如何能发现和纠正主要出错。

对于非致命错误，CQM1 不停止运行。它在 FAL 区域和错误日志中存储错误代码，对特殊的状态位置位，打开 CPU 模块上的 ERR/ALM 指示器，然后进入扫描循环。非致命错误包括在 CPU 和存储器模块之间的数据传输错误，PLC 在 DM 存储区设置数据字的问题，超过 100ms 的循环时间，或一个坏掉的后备电池。通过外围端口的通信问题也被认为是非致命错误，但是没有错误代码或状态位被存储；取而代之的是，端口连接器的显示灯停止闪烁。用户程序可以包括寻找错误代码或被非致命错误导致的状态位的指令，也能引起 CQM1 执行响应子程序。响应子程序能发送消息或清除来自 FAL 存储区的出错代码，但是它们在

⊖ MPI 是使 PLC 与其他计算机可以通信的局域网

⊜ 在 CQM1 进入运行模式时 DM 6655 的最低位决定错误日志的配置。xxx0 使 CQM1 记录最近 10 个错误的代码和时间；xxx1 指示只有最先的 10 个错误的代码和时间被记录；xxx2 指示错误日志中没有任何错误被记录。

主程序控制下执行，所以它们不会被中断。

CQM1 允许用户定义自己的用户出错代码。当程序执行失败警报（FAL（06）指令）或严重的失败警报（FALS（07）指令）时，CQM 将在 FAL 存储器区中或在出错日志中存储用户出错代码。如果 FALS（07）指令执行，CQM1 将作为致命错误来响应，而 FAL（06）指令作为非致命错误来响应。在任一情况下，指令都需要一个参数：CQM1 存储在 FAL 存储区或差错日志中的用户出错编码。CQM1 系统为用户出错代码保留了代码 01 到 99。推荐 FAL（06）的差分形式，@FAL（06），这样每次指令条件为真时仅有一个代码被存储。因为它们都在程序的控制下，所以 FAL（06）和 FALS（07）都被认为是软件中断。在本章的故障检修部分，我们讨论如何避免由扫描循环花费太长时间而引起的中断。

11.4.5 初始化中断

当 PLC 进入运行模式或当 PLC 正在运行时，如果电源在被中断后恢复，PLC 将在重新开始执行扫描循环以前执行一个初始化过程。这个初始化过程被认为是中断服务程序，尽管它的很多执行都将推迟实际的中断源（离开运行模式或关闭电源）。在初始化中断过程中 PLC 做什么，取决于用户如何配置 PLC 和用户已经编写的程序。第一次配置和重启初始化在第 10 章已描述，但是我们在这里讨论初始化程序的中断特征。

当 PLC 进入程序模式或电源发生中断时，正在运行的程序的实际中断发生了。PLC 立即[⊖]停止扫描循环和关闭所有的输出（如果电源仍为有效，可能冻结它们中的一些最后的状态）。然后，当 PLC 切换回运行模式或当电源恢复时，PLC 通过执行初始化 ISR 完成中断过程，之后从循环的起始处重新开始扫描循环的执行（不是从 PLC 被切换出运行模式而使循环被中断的地方）[⊜]。由于电源中断或由于切换到程序模式而使 PLC 停止，重启的过程将有轻微区别，当然对于不同的 PLC 型号也有所不同。

如果 PLC 因为切换到程序模式而停止，如果电源在程序模式时没有丢失，重启相对简单。当它进入程序模式时，它可能把输出关闭，但是不会丢失存储器中的任何数据，也不会丢失输出映像数据。当 PLC 在程序模式时，操作员可能已经使用编程终端来改变一些数据存储器，包括程序、工作数据、甚至是配置数据。当它们在程序模式时，一些 PLC 继续扫描输入，所以输入映像数据可能改变。当 PLC 转换到运行模式时，PLC 将使用最新的配置数据开始，将清除非保持性的数据存储器（当上一个程序在它结束之前终结时，最小化可能出现的危险），将执行一些第一次扫描操作，最后将开始执行标准扫描循环过程。在一些 PLC 中，第一次扫描操作仅仅是对状态存储器第一扫描位置位，所以用户程序能包括监控这个位的指令，这条指令能引起 PLC 在每次切换到运行模式时执行指定的功能。其他的 PLC 寻找用户编写的初始化程序，如果找到，在扫描程序开始前执行。一些 PLC 在运行任何用户程序前读取新的输入值，其他的 PLC 在读取新的输入值前执行用户编写的初始化程序。如果 PLC 在读取最新的输入值前运行用户程序，那么不需要编写初始化程序来响应输

⊖ 一些 PLC 提供后备电源，使 PLC 在电源中断后继续运行几个毫秒，但是这种特性只能延迟中断，或使 PLC 忽略短暂的电源中断。

⊜ PLC 不是每次重启时都需要执行同样的操作，因为在它的存储器中的数据是不同的。例如，顺序功能图程序不必重启它们的编程的过程。

入映像存储器的数据。

如果当电源被中断时 PLC 在程序模式，PLC 将丢失用户编写的程序、工作数据和所有 RAM 存储器（除了在非易失性存储器的 RAM 存储器区域）的配置数据。大多数 PLC 提供非易失性 RAM，例如电源备份 RAM，电可擦除只读存储器（EEPROM）模块，或被用来在电源中断时存储程序、数据、配置的闪存模块。当电源恢复时，这些 PLC 寻找 EEPROM 或闪存装置作为上电初始化程序的一部分。如果找到，PLC 将自动复制 EEPROM 或闪存的内容到 RAM 存储器。带有后备电池的 RAM 内容仅当没有 EEPROM 或闪存时才需要。EEPROM 存储器通常被程序员写入一次，它包含一个完整的用户程序和启动数据集，这样 PLC 可以在每次打开时以同样的方式启动。闪存和带有后备电池的 RAM 存储器可以在 PLC 运行程序时被 PLC 改变，所以它也可能包含反映 PLC 上次运行时做了什么的工作数据。

如果当 PLC 处在运行模式时，电源失效，当电源恢复时，大多数 PLC 自动地切换到程序模式作为上电初始化程序的一部分。这时操作员必须将它们切换到运行模式，然后 PLC 按与前一种情况一样的方式重启。其他 PLC 能被配置成经过短暂的电源中断后自动在运行模式重启。显然，用户程序、关键数据和配置数据，必须保持在带有后备电池的 RAM 或 EEPROM 或闪存中，在电源恢复后拷贝到 RAM。这些 PLC 通常在重启标准扫描循环前寻找和执行一个初始化程序，这个初始化程序不同于当 PLC 转换为运行模式时将执行的程序。

1. *初始化中断和* ALLEN-BRADLTY PLC

PLC-5
SLC 500

当它们开始或停止时，PLC-5 和 SLC 500 不清除任何数据存储器，甚至是输入和输出映像文件数据。当在程序模式时，它们继续扫描输入，所以当 PLC 返回运行模式时，输入映像数据将被更新。转换到运行模式后在这些 PLC 中的一个开始执行它的第一个扫描循环前，它自动地对第一扫描状态位（S:1.15）置位。在它结束第一次扫描循环后，PLC 关闭这个位。程序员能包括指令到主程序中以利用第一扫描位使初始化过程执行。任何以第一扫描位为条件的指令（包括跳转到子程序）将作为第一扫描循环的一部分被执行，但是将不再执行直到下次 PLC 停止和重新启动。

默认的，如果当 PLC-5 或 SLC 500 处于运行模式时电源丢失，将自动地重启到运行模式。扫描循环将从开始处重新开始，而不是从被中断处重新开始。这些 PLC 能被配置成在每次电源失效后自动地执行出错程序。如果配置成这样，那么每次电源被重新打开时，它们就对主要出错状态位置位。[⊖]因为主要出错有最高的优先级，所以出错程序能在 PLC 试图恢复扫描循环时执行，当 PLC 处在运行模式时电源中断就立即执行，而不在运行模式时就延时执行。如果 PLC 没被配置成带有一个出错程序，或者出错程序结束时没清除主要出错位，那么 PLC 将直接进入出错模式，这需要操作员来进行干涉。如果出错程序在结束前复位主要出错位，那么出错程序结束后，PLC 将重新开始标准扫描循环，而不需要操作员干涉。

PLC-5 能被配置成无论何时它重启时，重新恢复运行 SFC 文件，这个文件从上一次完

⊖ 为了配置 PLC-5 在电源恢复时执行出错程序，置位 S：26.1。对于 SLC 500，置位 S：1/9。PLC-5 检测到电源上电时，通过对主要出错位 S：11.5 置位来触发出错程序。

成的步骤之后开始，而不是从第一步开始。PLC-5 可以通过设置状态位 S:26.0 来配置。(SLC 500 不提供 SFC 编程) 这个配置位在梯形图程序中没有影响，因为梯形图程序总是从开始处重启。

S5

2. Siemens STEP 5 的初始化中断

当 S5 PLC 从程序模式切换到运行模式时，清除输入和输出过程映像 (PII 和 PIQ) 以及非保持型存储器，执行组织块 21，然后使能输出和重启标准扫描循环。如果程序员没有为 OB 21 输入一个程序，那么扫描循环将不用任何初始化重启，而不会清除 I/O 映像和非保持型存储器。注意 OB 21 不应该编程来通过检查 PII 表值响应输入条件，因为输入映像保持清除直到 OB 21 结束后。大型的 S5 PLC 能在 OB 21 使用直接访问寻址，因为直接寻址使 PLC 直接从输入模块中读取输入数据，而跳过 PII 表。注意直到第一标准扫描循环结束，输出将保持关闭。在 OB 21 结束后，它们不会立即改变。

如果当 S5 PLC 在运行模式时，电源中断，然后恢复，像上面描述的那样，PLC 执行相同的程序，例外的是执行 OB 22 而不是 OB 21。初始化中断例行程序 OB 21 和 OB 22 只能被出错中断中断。

S7

3. Siemens STEP 7 的初始化中断

S7-400 系列 PLC 提供两种方式启动，而低型号的 S7 PLC 仅提供一种方式。所有 S7 PLC 都提供的启动方式叫做*完全重启*，而由 S7-400 PLC 提供的方式叫做*重启*。当 PLC 没有错误地执行一个程序时，如果电源在丢失后恢复，S7-400 将执行*重启* (仅在 CPU 的重启选择开关在 WRST 而不是 CRST 时，并且 PLC 的存储器和模式开关没有变化时)。S7-400 CPU 模块也能被配置成在切换到程序模式后执行重启，然后返回运行模式。在这种重启类型中，PLC 重写配置参数到智能 I/O 模块，执行 OB 101，使能输出，然后恢复执行用户程序 (从电源丢失时被中断的一步开始)。PLC 不改变 I/O 映像，除非 PLC 已经被配置在重启时清除它们。

每次 S7 300 或更低型号的 PLC 切换到运行模式时，或者它处于运行模式时掉电并重新打开电源，或者当 S7-400 PLC 在不允许重启的条件下启动，执行完全重启。在完全重启中，PLC 清除非保持型数据存储器，清除 B 堆栈和 I 堆栈，清除警报，并加载配置参数。如果 PLC 在运行模式时电源丢失，那么它也写参数到智能 I/O 模块。然后在使能输出和开始执行主程序循环之前，PLC 执行 OB 100，开始程序。

OB 100 和 OB 101 能包括用户编写的初始化程序。两者都有优先级 27，所以它们只能被诊断中断 OB 82，或编程出错 OB 121，或模块访问错误 OB 122 中断。[⊖] 当操作系统调用 OB 100 或 OB 101，它传递几个用户编写的初始化程序可使用的参数。这些参数包括：

1) 一个代码，定义切换到运行模式时，或 PLC 处在运行模式时电源中断后再恢复，PLC 是否启动。

2) 一个代码，定义为什么 PLC 进入停止模式。它可能是由操作员切换到停止模式，或者可能是一个出错导致的。程序检查这个代码和先前提到的代码，使得初始化程序能被编程为由于 PLC 停止运行的原因不同而进行不同的操作。

⊖ OB 82 通常具有优先级 26，但如果在执行 OB 100 或 OB 101 时，它由一个检测到的硬件出错引起，它会被分配优先级 28。OB 121 和 OB 122 被分配与它们中断的组织块具有相同的优先级。

3）一个 32 位数据字，它包含显示当 PLC 进入停止模式时，配置数据是否已经被改变的状态位，是什么初始化这个到运行模式的切换，以及在 OB 100 或 OB 101 前的启动程序是否已经成功重启 PLC

直接访问寻址能被用来执行立即从输入模块读取，或当 OB 100 或 OB 101 执行时，执行到输出模块的立即输出。当 S7 PLC 进入停止模式时，它使它的输出进入预先设定的状态（在配置中选择）而不是清除它们。当处在停止模式时，程序员能改变 CPU 中的存储器配置。

4. 初始化中断和 OMRON CQM1　CQM1

默认的，当 CQM1 进入运行模式时，输入映像和输出映像被清除。当它进入运行模式时，程序员能配置 CQM1 跳过清除 I/O 映像，通过设定 I/O 存储器保持位（SR 25212）和设定在 DM 6601 的第二高的 BCD 位为 1。如果 DM 6601 不以这种方式被配置，在初始化例行程序中 SR 25212 将被自动清除，然后 I/O 映像也将被清除。

CQM1 由一个启动模式状态配置字（DM 6600），它指示了当电源中断后重新恢复时，PLC 如何被响应。默认的，DM 6600 包含 0000，它配置 PLC 在运行模式启动。如果 DM 6600 被改变为 0100，那么在电源被中断时，CQM1 将在原来的模式重启。如果 DM 6600 被设置为 02xx，那么在电源恢复后最后两个数字决定 CQM1 将进入的模式：0200 是程序模式，0201 是监控模式，0202 是运行模式。如果 CQM1 连接了编程器，那么它将以编程器设定的模式启动，而不管 DM 6600 中的配置值。

CQM1 有第一循环标志位（SR 253 15），它在 PLC 切换到运行模式后或在电源中断后 PLC 返回运行模式后运行第一次扫描循环。主程序在第一次扫描循环时能包括检查这个位的指令以引起初始化操作的执行。

11.4.6　通信中断

现代 PLC 包括通信处理器和被用来处理串行通信的独立的微处理器。主微处理器通过监控通信处理器的状态和发布通信请求来服务通信处理器。通信处理器经常直接访问主存储器。因为服务能在一个扫描循环的 I/O 步骤中完成，与其他计算机不同，PLC 中的服务于通信处理器的中断不是至关重要的！在这个部分我们讨论一些通信中断的特性，在 13 章介绍其他可编程通信能力。

一些 PLC 编程语言包括使 PLC 中断用户程序来立即执行服务而不是等待直到 I/O 扫描的指令。这些 PLC 通常提供状态位，它们被通信处理器正做的事情所控制，所以当它们需要改善通信效率时，PLC 能被编程来执行这些通信服务指令。

分配给通信处理器的数据交换任务通常被加到通信处理器的任务队列的结尾，但是一些 PLC 提供使一个任务被放到队列的开始的指令，和/或使用户程序中断直到通信处理器已经完成数据交换任务的指令。

1. 通信中断和 ALLEN-BRADLEY PLC-5　PLC-5

PLC-5 提供一些用户可访问的指令或配置特性来影响通信中断，部分原因可能是 PLC-5 有很强的通信能力，用户需要花费更多的时间来研究它。防止用户改变 PLC-5 操作系统处理通信的方式是安全的。PLC-5 CPU 模块包含通信处理器，并且在一个扫描步骤的通信时间片期间，主微处理器控制它们以及和它们进行交换数据。这些不是中断，在 13 章将讨论。

用户程序能控制通信中断。如果块转移请求在 ISR 中或在出错程序中用来响应可选定时中断（STI）和过程输入中断（PII），PLC-5 自动中断 ISR 的执行并执行块传输。直到通信处理器完成数据的传输，ISR 才恢复执行。如果在块传输读或块传输写（BTR 或 BTW）指令执行前执行用户中断禁止（UID）指令，程序员能够阻止 ISR 由于块传输指令而中断。在 BTR 或 BTW 后执行用户中断使能（UIE）指令，来重新使能中断。在 UID 和 UIE 之间的块传输指令将请求的块传输放在通信队列中，所以数据传输最终将执行。

接下来，我们描述 PLC-5 如何在通信处理器和主程序间共享对通信通道的访问。当程序员不能改变这些特性时，对于 PLC-5 的程序员来说知道 PLC-5 如何处理通信是很重要的：

1）立即 I/O 指令（IIN 和 IOT）与本地或扩展框架的块传输相冲突，这是因为当用户程序执行时，它们使用相同的本地通信总线，导致数据总线上可能的数据崩溃[⊖]。如果程序员必须为本地框架编写块传输和立即 I/O 指令，那么立即 I/O 指令应该是有条件的，即没有激活的块传输在执行，这能通过检查 ST（start）位的状态来检测。

2）CPU 所在的框架中的模块的块传输干扰对来自 CPU 所在的框架的 PII 信号的监控。

3）到远程 I/O 框架或来自远程 I/O 框架的块传输使 PLC-5 中断连接到串行通信链接的框架的远程 I/O 扫描，这样块传输能执行，所以远程数字 I/O 状态的更新就变得更慢了。一个单独的块传输能轻易地使远程 I/O 扫描的时间加倍（例如，通过 230.4kbps 的串行通信，它花费了 2.7ms 来传输 10 数据字的块，而远程 I/O 扫描时间通常仅是 3ms）。PLC-5 在每个远程 I/O 扫描循环时，通过对每个远程框架执行一个块传输来限制延时（不用对 ISR 的块传输请求计数，它能发生任意次）。

4）由 ISR 中的块传输读（BTR）或块传输写（BTW）指令产生的通信请求，比在通信队列中的其他请求的优先级更高，除了已经激活的请求，所以中断时间被最小化。如果是远程 I/O 框架内一个模块的数据块传输请求，那么当 ISR 等待块传输完成时，新的增强型 PLC-5 执行 ISR 中断的 MCP 程序。

SLC 500

2. 通信中断和 ALLEN-BRADLEY SLC 500

SLC 500 的通信服务功能在扫描循环中的 housekeeping 步骤被执行。直到扫描循环的 housekeeping 部分被执行，才有数据被接收或被发送，所以通信的执行通常不需要任何通信中断。没有中断的通信在 13 章详细讨论。

对于 SLC 500，从 5/02 开始的型号提供两个指令，它们可以用来使用户程序立即中断服务通信。服务通信（SVC）指令引起串行通信通道的立即服务；I/O 刷新（REF）指令（在早期关于立即 I/O 中断的部分已讨论过）引起一个完整的 I/O 扫描，包括串行通信通道的服务。新的 SLC 500 型号（5/03 和更高级的）允许程序员指定两个串行通信通道中的一个响应 SVC 或 REF 指令。SVC 和 REF 指令不能用于其他的中断服务程序，这暗示着通信中断分配了相对低的优先级。SLC 500 为串行通信提供了几个通信状态位，用在用户程序中控制 SVC 或 REF 指令是否应该执行。这些位包括：

1）输入命令未决位，如果已经从另一个控制器接收到一个通信请求，该位置位。通信服务清除这个位。位 S：2/5 指示请求通过串行通道 1 已被接收到。S：33/0 指示请求通过

⊖ 对远程框架的立即 I/O 指令只能读或写来自远程 I/O 扫描缓冲存储器的数据，所以它们不能干扰到远程框架的块传输，但是它们实际不能执行立即 I/O。

在 SLC 5/03 或更高型的串行通道 0 被接收。

2）消息回复未决位，如果另一个控制器已经发送信息给早前的消息请求控制器，该位置位。当这个控制器服务即将输入的通信时，该位清零。通道 1 使用 S：2/6，通道 0 使用 S：33/1。

3）输出信息控制未决位，当可能发送一个来自控制器消息队列的消息时，该位置位。当通信通道的服务导致消息传递开始，该位清零。位 S：2/7 指示通道 1 准备好，S：33/2 指示通道 0 准备好。

3. Siemens STEP 5 的通信中断

S5

STEP 5 用户手册没有描述任何通信中断控制指令或用户可以利用的配置设置。

4. Siemens STEP 7 的通信中断

S7

S7 PLC 必须被配置和被编程来在局域网中和其他计算机交换数据。高效的网络软件完成大部分数据交换操作而不需要 PLC 的中断，但是存在一些系统函数使用网络通信的中断。

系统函数 SFC 35（“MP _ ALM”），利用 MPI 或 Profibus 网络来中断所有网络中的其他 PLC，在所有的 PLC 里同时开始 OB 60（“多计算中断” OB）的执行。每个 PLC 在它的 OB 60 中有不同的程序，如果 PLC 缺乏一个在 OB 60 里的程序，将不会出错。OB 60 有一个优先级 25，所以如果它存在，当一个信号通过局域网到达来使其运行时，它将中断几乎任何其他的 OB。网络信号可以被任何 PLC 初始化，PLC 执行 SFC 35 MP _ ALM。为了防止由于两个 PLC 都试图执行 MP _ ALM 而发生的冲突，如果 OB 60 正被网络中的任意 PLC 执行，没有 PLC 能识别开始执行 OB 60 的信号。调用 MP _ ALM 的 PLC 程序必须提供一个由 1 到 15 的数字组成的 JOB 参数。这个值被发送到其他带有中断信号的 PLC，它作为 OB60 _ JOB 参数而提供给 OB 60。OB60 的程序能检查这个值来知道为什么中断被调用。

在配置过程中，S7 PLC 能被设置来参与全局数据循环。通常情况下，网络在这个循环中循环地在 PLC 之间复制数据，但是存在两个系统函数能被调用来中断网络，从而发送或接收全局数据。SFC 60（“GD _ SND”）能被调用来立即发送一个全局数据包。需要的参数包括 CIRCLE _ ID，用来发送数据的子网号（1 到 16），以及包括 BLOCK _ ID，发送数据的特殊 PLC 的数字（1 到 3）。SFC 61（“GD _ RCV”）以一种相似的方式使用，但它使一个特殊子网中的全局数据被立即从一个特殊的 PLC 中读取。

5. 通信中断和 ORMON CQM1

CQM1

CQM1 能使用三种方式通信，其中没有一种可以使用被用户程序控制的中断能力。CQM1 的通信能力在 13 章详细介绍。

11.5　总结

当一个中断服务程序（ISR）执行时，中断将延缓扫描循环，然后通常情况下恢复成它们好像没被中断那样。高优先级的中断能中断低优先级的中断。

程序指令能导致来自输入模块的立即输入或到输出模块的立即输出。特殊寻址能被用来代替特殊指令。一些 PLC 允许通过一个单独的指令发送或接收 I/O 映像数据的多个字，OMRON CQM 1 能被配置来直接输出所有程序产生的输出数据到输出模块。

定时中断被用来强制 PLC 以精确的时间间隔执行 ISR 程序，因而使控制具有更多的确定性。程序员可能会在定时中断 ISR 中使用立即 I/O 指令，这样输入的采样和输出的改变

执行的时间间隔与定时中断 ISR 的执行时间间隔相同。定时中断也可以用来使 PLC 在每个月、天、小时、甚至分钟的相同时刻执行 ISR。定时中断能被用来开始一个延时定时器，使得一个 ISR 在定时器开始后的一个精确时间执行。定时中断通常在 PLC 中断类型中具有最低优先级，这意味着如果一个高优先级的中断已经执行，那么它们将不能开始，一直要等待到 ISR 结束。如果这种情况发生，状态位经常被置位，这样程序能响应任何由这个问题引起的困难。

I/O 中断使一个 ISR 执行来响应连接到输入模块的传感器所产生的信号。一些 PLC 能被配置成监控多重信号，有时可以对正在迅速改变的输入信号计数，当计数器到达预设值时执行 ISR。一些 PLC 仅能监控内置于 CPU 模块的输入触点。I/O 中断的优先级比定时中断高，但是比出错中断的优先级低。

出错中断，作为最高优先级的中断，在 PLC 检测到故障时发生。如果被检查到的问题是次要的，那么 PLC 仅中断足够长的时间来存储出错代码，并可能在存储器中对一些出错位置位，使得用户程序或操作员能找到它们。如果问题比较严重，那么 PLC 在存储出错代码后，可能寻找并执行一个用户编写的出错程序 ISR，然后离开运行模式。出错程序有时可以编写为能够纠正出错和清除出错位，这样 PLC 能恢复被中断的程序，但严重的出错通常引起 PLC 在保存出错标志数据后停止。一旦 PLC 以这种方式进入出错模式，那么被中断的扫描循环不能恢复。出错被纠正时，仅能从最开始的地方重启。大多数 PLC 提供能导致出错中断的指令，或许仅允许程序来停止 PLC 运行，但有时也允许触发高优先级的出错程序。

初始化中断，几乎具有和出错程序一样的优先级，响应 PLC 离开运行模式。当 PLC 重新打开或返回运行模式，中断在重启通常的扫描循环前通过引起一个初始化 ISR 的执行来结束。一些 PLC 在此时清除部分存储器，通常包括 I/O 映像数据。初始化 ISR 的其余部分通常在新的输入值被读入前执行。Allen-Bradley 和 ORMON 的 PLC 设置一个第一扫描位，程序员可以在主用户程序中包含指令来监控这个位，并执行任何可能需要的初始化。Siemens PLC 在重启扫描循环前自动执行一个用户编写的初始化程序。如果电源已经被中断 PLC 可能以与上次执行不同的方式开始，或许执行出错程序，或许强制操作员手动切换 PLC 到运行模式。

现在 PLC 可以内置仅处理串行通信的微处理器，一些 PLC 提供少量的通信中断。通信中断被用户程序中的指令初始化。这些（通常是低优先级）指令使 PLC 立即中断扫描循环来提供它需要的通信处理器。

11.6 故障检修

带有中断能力的程序的执行顺序不像没有中断能力的程序一样具有可预测性。子程序只有在它被调用时执行，但是中断服务程序（ISR）能在其他程序执行的任何时候执行。如果你发现你的 PLC 程序以不应该的方式执行，这可能是 ISR 的问题！一些 PLC 保持最近发生的中断的记录（例如 Siemens S7 PLC 包括在它的诊断缓冲区的中断事件）。如果你的 PLC 保留一个 ISR 调用的历史记录，那么就很容易决定一个 ISR 是否已经执行了，因为一个 ISR 可能在你察看程序监控屏幕之前已经被调用和结束了。如果 PLC 不保留中断历史记录，那么你可以使用一个简单的窍门，如编一个翻转计数器，就像图 11-27 显示的那样，然后监控

计数器的累加值来观测一个 ISR 程序是否正被执行。(在 ISR 的执行期间使用一个保持的数据位)。

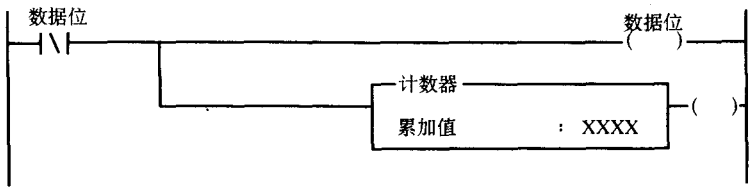


图 11-27　一个简单梯形图的 ISR 中的梯级来计算每次 ISR 执行的时间

近来引入的一个非强制性的 IEC 1131-3 标准要求严格的区分数据存储器，使得 ISR 不能修改被中断的程序正使用的存储器的内容，也要求操作系统确保中断不能发生，直到所有 ISR 需要的数据可以使用为止，但是大多数 PLC 的操作系统设计在标准发布之前。如果你发现数据值在不该改变时改变了，那么再次怀疑你的 ISR 程序。大多数编程软件包提供一个前后参照特性，它能报告一个地址在程序中所有被使用的地方。使用前后参照来观测一个 ISR 是否有能力来改变数据值。小心使用用来保护数据值不被 ISR 修改的中断禁止和重新使能指令。

一些中断默认是使能的（例如 Siemens 定时中断或循环中断），而其他中断默认是禁止的（例如 Allen-Bradley STI 定时中断）。一些中断需要被配置和解屏蔽来执行（例如，OMRON 间隔定时中断）。如果 ISR 不执行，或者如果它们不按预期的执行，那么检查 PLC 中有什么被要求来开始或停止它们。

理论上中断信号的响应是立即的，但是存在几种原因使在 ISR 执行前可能有延时。当 PLC 响应中断请求时，一些 PLC 允许程序员通过配置来调节这个延时：在扫描循环的任何时间或仅在用户程序执行时，在每一个梯级后，在每一个指令后，或一个长指令的中途。如果 PLC 仅在一个梯级结束后允许中断，那么程序不应该包含有很长执行时间的梯级。定时中断可能不能在应该执行的时刻精确的执行，因为它们没有足够高的优先级来中断其他的 ISR，甚至是其他的定时中断。大多数 PLC 包含指示 ISR 在它执行前是否必须等待的状态位。使用编程器或用户程序 ISR 本身监控这些状态位。如果你怀疑计数器比你所看到的要改变的更快时，使用状态位来驱动计数器。

甚至 ISR 在需要执行的时候执行，它可能不能使用最近可获得的数据。大多数 PLC 当它们调用一个 ISR 时，不更新输入映像表，所以输入映像数据可能在最长的扫描循环时间后过时了（大多数 PLC 保留一个显示最长扫描时间的状态字）。类似的，ISR 写到输出映像表的数据可能在一个完整扫循环时间里没有被写到输出模块。如果不能接受这些延时，使用立即 I/O 指令或在 ISR 中寻址。

高优先级的 ISR 程序能中断低优先级的 ISR。无论何时任何程序被中断，PLC 都把中断程序的数据压入堆栈存储器的顶部，使得它能在中断结束后被恢复。中断堆栈保留的存储器区域的大小是有限制的，所以存在一个等待恢复的中断程序数目的极限。如果太多的中断优先级被允许，堆栈将溢出并丢失一些需要恢复的中断程序的数据。大致来说，PLC 的制造商指定了少数几个中断优先级，使得它们的 PLC 不会由于堆栈溢出而引起问题。但是，现代的 PLC 提供给程序员越来越多的灵活性，所以将来的程序员不得不小心配置它们的

PLC 来避免堆栈存储器溢出；否则，PLC 将不能够恢复被中断的程序。

每次一个程序执行对另一个程序的普通调用时，堆栈存储器也被用来存储返回地址，一些 PLC 还使用堆栈存储器来存储工作数据。PLC 可能会对所有这三种目使用相同的堆栈，所以由于程序调用和/或存储工作数据，堆栈存储器可能溢出。在 IEC 1131-3 之前，对于堆栈存储器来说，没有统一的标准。⊖

如果你的程序不能严格地响应高速计数器累加值，这可能是因为程序正通过检查是否等于目标值来检查计数器的累加值。记住高速计数器使用中断来允许对比 PLC 扫描快的输入信号的改变计数，所以高速计数器能在扫描之间计数到刚刚超过极限值。监控高速计数器累计值的指令应该通过比较来看它是否超过了极限值，而不是通过检查来看它是否和一个极限值相等。当高速计数器到达它的预设值（例如，SLC 500）时，一些 PLC 能被配置来立即中断它们的扫描循环。

如果你的程序在执行时改变中断配置，但中断又不能正常工作，记住进入运行模式时，一些 PLC 仅接受某种特定类型的配置的改变。它不可能改变你的 PLC 的中断响应，除非当 PLC 进入程序模式时你做了一些改变。如果你想编写一个初始化程序来设置一个默认的中断配置，初始化程序可能不得不把新的配置和旧的配置相比较，如果它们不同，执行一个指令，在写入新值之后产生一个用户初始化出错。PLC 进入程序模式，操作者不得不将其转回运行模式。PLC-5 必须以此方法编程来允许一个初始化程序改变任何 PII 配置，除了 PII 文件号。OMRON PLC 配置数据存储器的值仅当 PLC 在程序模式时可以被改变，但是中断可以通过程序中的指令打开和关闭。

中断将使 PLC 更快的响应或在更多的重复时间间隔响应，但是它们可能使整个程序扫描时间更长或更加不可预料。它甚至可能使 PLC 在无差错地运行很长时间后，不预期的进入出错模式，因为在同一个扫描中可能有太多地 ISR 运行或者看门狗定时器时间到。PLC 提供几个特性来帮助程序员预测、避免或从扫描时间的无规律性和看门狗定时器问题中恢复。编程手册经常包含能被用来计算一个扫描循环在最坏地情况下能运行多长时间的公式。大多数 PLC 也保留一个显示最长扫描循环时间的状态字，这使得程序员能把计算出的结果和实际扫描时间相比较。购买时，PLC 通常被配置成有长看门狗定时器设置，但也允许程序员改变看门狗定时器的时间。

习题

1. 写出能被 I/O 中断所中断的三步 PLC 扫描循环的每一步骤的名字。
2. 简要描述 PLC 如何处理两级中断请求，其中一个在扫描循环过程中接收到，另一个高级别的请求在第一个中断的中断服务程序的执行过程中接收到。
3. 对于哪一种中断类型，被中断的程序不能恢复？
4. 对于 MPU 寄存器中的数据（例如，累加器），响应一个中断和响应一个子例行程序调用有何不同？
5. 如果你的 PLC 不得不对一个过程提供伺服控制，这个过程改变的太快以至于不能正确地

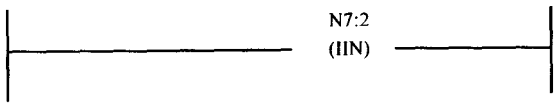
⊖ 对于本书所讲到的 PLC，只有 Siemens S7 系列 PLC 坚持 IEC 61131-3 标准，对于每一个程序块使用不同的存储器。它要求三种类型的堆栈：对于每一个 STEP 7 优先级，L 堆栈保存最多 256 个字节的本地数据，B 堆栈保持的数据可以供八层嵌套调用的程序使用，I 堆栈保持嵌套的中断的数据。

控制，因为普通的 PLC 扫描循环有延时，那么什么类型的指令可以用来读取传感器和输出到伺服器呢？

6. 立即输入/输出指令是否被硬件中断或软件中断或定时中断使用？
7. 配置 PLC，使用定时中断可以让 PLC 做什么？
8. 什么类型的事件将引起 I/O 中断（有时叫做硬件中断）？
9. 对于定时中断或 I/O 中断，哪个有更高级别的优先权？
10. 为什么高速计数器需要使用中断？
11. 解释一下为什么使用中断能使主程序的控制动作具有更少的确定性；你能使用什么类型的中断来使控制动作具有更高的确定性？
12. 对于定时中断有很多实用的限制用来缩短时间间隔，描述这些限制。
13. 什么类型的事件能触发出错中断？
14. 出错中断程序有时能被写入使得它们允许被中断的程序在出错程序结束执行后恢复，出错程序不得不做什么？
15. 如果你正在学习的 PLC 允许用户初始化的出错中断的使用，那么请描述一下你将如何设置和对一个非致命中断编程。
16. 初始化中断需要初始化什么？你如何配置和对你正学习的 PLC 编程来使用一个计数器作为一个初始化中断的一部分？

Allen-Bradley PLC

17. 写一个 PLC-5 的梯级，无条件地读取来自框架 12 中模块 6 的 16 个输入位的状态。PLC 将 16 位的数据放在哪里了？
18. 如果 N7：2 包含值 0，那么下一条指令将做什么？



19. 你的 PLC-5 的看门狗定时器被设置得太短，并且 PLC 偶尔会出错。你想写一个临时的程序，在看门狗定时器每次引起一个出错时稍微地增加看门狗定时器的时间。PLC 应该在每次发生出错后重新自动地恢复操作，并且最终它不会发生任何出错。（在一个小时后你将删掉这个临时程序。）仔细描述这个程序，但你不需要实际地写出或给出实际的状态位或字的地址。
20. 为了使 PLC-5 出错，你能对 PLC 编程来置位一个_____位。

Siemens PLC

21. ■ 无论何时一个 55-100U PLC 执行一个硬件中断（也称作 I/O 中断），OB002 执行。PLC 还做什么来执行组织块？清楚地解释在附加的步骤中将发生什么，并显示附加步骤在什么时候执行？
    ■ 在硬件中断组织块中，你要一个程序来使用输入模块 1 的最近的信息，你将编一个加载指令如下：L _____ B1（填空）
22. 如果你想一个 Siemens PLC 在程序执行时立即出错（来停止程序），那么你能编写_____指令。

23. 写一个 Siemens STEP 5 或 STEP 7 程序来读取来自输入模块（输入字节 25）的数据的一个字节的内容，并检查立即输入数据的一个位（bit 4）。

推荐的 PLC 实验室练习

对一个系统带有：

- 四个控制面板开关：输入 0 到 3
- 四个显示灯或可视的输出模块的 LED 灯：输出 A 到 D
- 两个弹簧复位阀门控制气缸：输出 E 和 F
- 一个锁销阀门控制气缸：输出 G 和 H
- 三个传感器来检测每个气缸的伸展

写一个程序和/或配置你的 PLC 使它执行：

1）一个 3s（或 3000ms）间隔的定时中断程序。当开关 0 打开时，程序应该打开指示灯 A。

2）几个（至少 10）立即输入和立即输出指令，但仅当开关 1 打开时。PLC 处于运行模式，当你打开开关 0 时在处理器状态屏幕监控扫描时间。

3）减少看门狗定时器时间到一个值来引起处理器出错，但仅当开关 0 打开时（当立即输入和输出执行时）。

4）写一个无论何时看门狗定时器引起出错时都将运行的出错程序。出错程序必须清除合适的出错位（仅为看门狗定时器设置的位），并且每次它执行时，必须给看门狗时间设置增加一个时间间隔。（在生产环境下这将不是一个好程序。）

第 12 章　过 程 控 制

12.1　学习目标

本章您将了解到：

- 过程控制的术语
- 不使用任何内置的 PID 指令来编写 PLC 程序进行过程控制，在此我们将用到：
  - 设定值、过程变量和控制变量
  - 变量的标度变换和范围限定
  - 使得控制间隔快速并具有确定性的方法
  - 偏离量、扰动量和前馈量
- Allen-Bradley、Siemens 和 OMRON 的 PID 过程控制指令或功能块：
  - 程序和数据输入
  - 方程：独立增益、非独立增益、变量
  - 比例项、积分项和微分项的作用，它们如何控制稳态误差和超调误差
  - 使用平滑功能和抗积分饱和功能
- PID 控制的替代方法——模糊控制
- 编程实现：当 PID 指令执行时，允许手动控制；当环境约束条件改变时，进行自适应的自动控制。

12.2　过程控制导言

当我们使用 PLC 来监控输入端和改变输出端时，我们就是使用 PLC 进行过程控制。在这一章里，我们将用过程控制一词来表示一些较为复杂的事物：在过程控制中，设定值命令信号告知控制器被控过程应当被驱动到哪里，传感器则测量过程的实际输出并将该信息反馈给控制器，控制器再提供控制信号给执行器，让它使被控过程到达所要求的状态。

这个定义意味着 PLC 必须执行计算功能来决定执行器应该如何动作。许多 PLC 都有这样的计算功能。PID 和模糊逻辑算法是 PLC 实现过程控制的最普遍的两种方法。如图 12-1 所示，这是一个典型的过程控制应用。在这个例子中，PLC 用来控制一个由电动机驱动台架的位置，这个台架是通过滚珠丝杠前后运动的。操作员通过一个操控旋转式的电位器来给出设定值。这个设定的电压值与电位器的旋钮位置成正比，显示了台架与电动机之间应该有的距离。还有一个线性电位器提供另一个电压信号，该信号和台架与电动机之间的实际距离成正比。台架运动位置的上下限由滚珠丝杠的长度来决定。

当操作员改变电位器表盘上旋钮的位置后，PLC 就会命令电动机运转，直至线性电位器提供的电压与电位器表盘上指示的电压值相同为止。图 12-2 给出了本系统的过程控制结构图。这个结构图中包含了由 PLC 控制的过程的一般性结构。

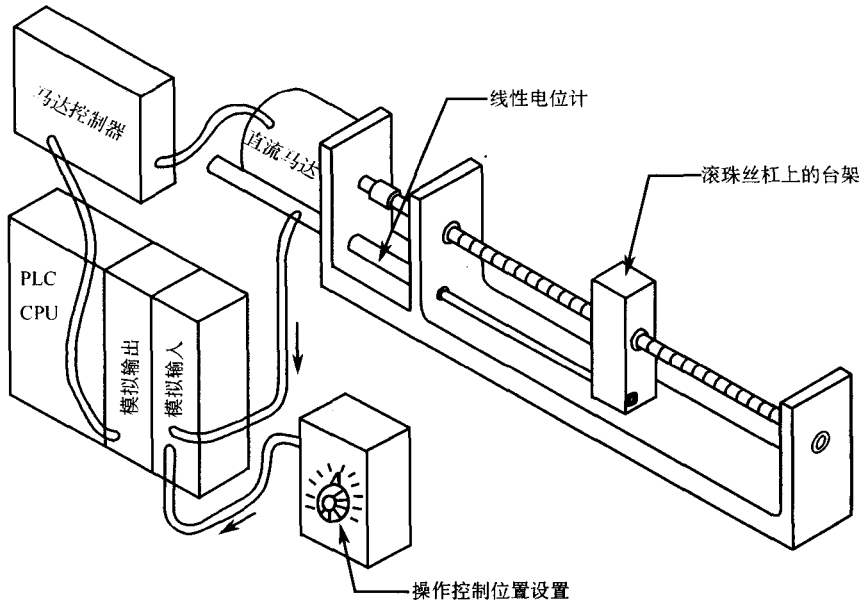


图 12-1 过程（位置）控制

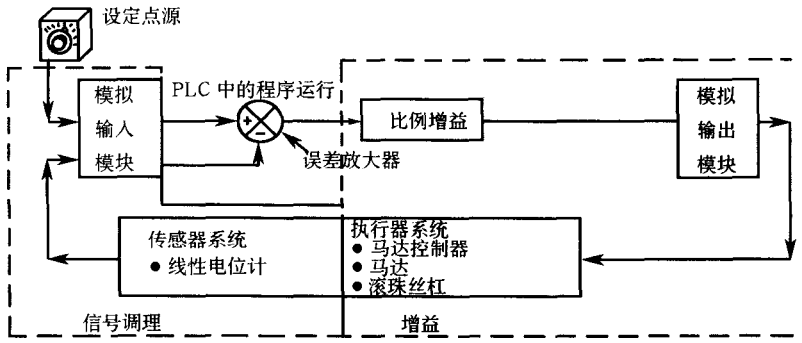


图 12-2 过程控制结构图

1）结构图中的设定值是由旋转式电位器提供的电压信号。它经过 PLC 的模拟输入模块转换成数字量信号后进入 PLC。

2）*误差放大器*，由 PLC 实现此功能，将在后面详细讨论。

3）本例中，整个系统的增益包括 PLC 程序的计算结果，并受到以下因素影响：

- 与 CPU 模块相连的一个模拟输出模块。这个模块输出一个与 CPU 模块给出的数字量成正比的模拟信号。智能模拟模块可以调节此值。
- 电动机控制器，将 PLC 给出的微弱的模拟电压信号放大为高功率的直流信号，提供给电动机。
- 直流电动机，它将直流电流转换成扭矩，进而来改变螺杆的旋转速度。
- 滚珠丝杠和球状螺母，它们的作用是将螺杆的旋转速度转变成台架沿螺杆方向运动的直线速度。

控制器生产商在产品说明中的结构图里一般不画出执行器，这是因为执行器是由用户选

定的。尽管上面这个结构图是这样画的，但一个被控系统的增益不是完全由 PLC 决定的。我们还应当考虑执行器的特性和它所处的环境。

4）此系统的信号调理器由两部分组成：

- 线性电位器，将电位器的游标位置转换成电压信号。（控制器的生产商一般是不考虑传感器的作用的。）
- 模拟量输入模块，将反馈回来的电压信号转换成数字量，也可能会在 CPU 模块读取数据前对结果进行调整。

过程控制系统的结构图中的信号都有自己的名称，如图 12-3 所示。

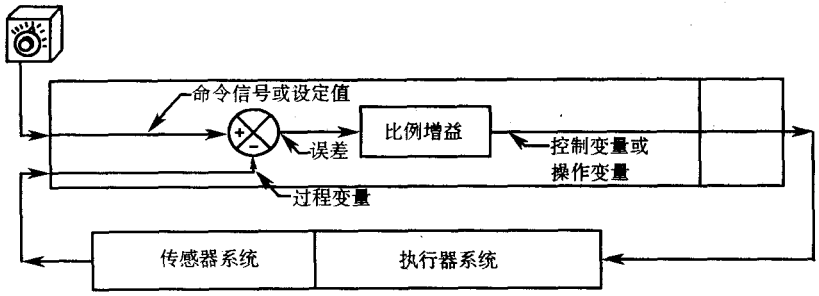


图 12-3 信号名称

1）命令信号，有时也叫设定值。

2）由误差放大器送给增益块的信号通常叫做误差信号。

3）系统的最终输出叫控制变量或操作变量，当然也可以叫其他名称。在这个例子中，我们把它叫做台架位置，并且使用长度度量。控制变量可以是速度、温度、压力或任何其他可测的量。如果结构图中未画出执行器，那么控制器传送给未知执行器的信号就叫控制输出。

4）从传感器得到的反馈信号有时被称作过程变量，因为它是随着测量的被控过程的状态成比例地变化的。

12.3 PLC 在过程控制中的应用

在传统的过程控制理论中，误差放大器只是简单地将系统的设定值和过程变量相减，再将误差信号输出。而其他的功能都归入到增益函数块中，再用公式描述出控制器的电路特性和数字计算。

PLC 程序员应该把 PLC 看成是误差放大器（用来计算设定值和过程变量间的误差）和控制器增益（对误差值进行数学计算）的组合。PLC 控制器提供控制信号给执行器或执行器系统（比如上例中的电动机和电动机控制器）。执行器（在传统过程控制理论中属于增益功能块的一部分）不可能准确无误地响应 PLC 的输出信号，但对此应采取什么措施已经不是 PLC 的工作了，当然也就不在本书的讨论范围内。

如图 12-4 所示的一个 PLC 程序，它的功能只是最简单的比例过程控制。在这个 Allen-Bradley 梯形图中，PLC 程序将存放在地址为 N7：2 的过程变量与 N7：1 中的设定值相减，来实现误差放大器功能，然后再将所得的误差值乘以一个存放在 N7：50 的数值，这样就完成了控制器增益的功能。改变存放在 N7：50 中的数据值将会改变整个系统的比例增益。这个程序的结果会放入 N7：100 中。

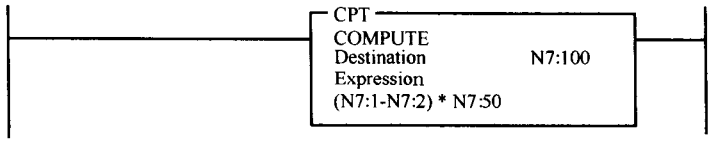


图 12-4 非常简单的 PLC-5 过程控制程序

过程变量的值和（有时）设定值必须通过模拟输入模块从传感器中读取，控制变量输出值必须通过模拟输出模块写入执行器系统，所以程序中必须有对模拟 I/O 模块的读和写。如果模拟 I/O 模块在你所使用的 PLC 中是可直接寻址的，那么这就简单得多。如图 12-5 所示的一个 STEP 5 STL 程序，只是简单的过程控制程序，其中有对模拟 I/O 模块地址的读、写。目前用于 Allen-Bradley PLC-5 系统的模拟 I/O 模块是不可直接寻址的，所以需要用块传送指令在进行计算前读取模拟量的值，并且在计算后将模拟量的值输出，如图 12-6 所示。

```
          L IW072      ;读取设定值
          L IW074      ;取反馈值
          -F           ;二者相减
          T DW0        ;存误差值
          L KY + 0     ;将 DW1清0
          T DW1
          L DW2        ;取乘数
          T DW3        ;将它赋给另一个工作数据字
          JZ = out     ;如果乘数为0则跳过乘法运算
add   :   L DW0        ;将误差值与 DW1中的值相加并存入 DW1中
          L DW1
          + F
          T DW1
          L DW3        ;从 DW3中取乘数
          L KF+ 1      ;逐次减1
          -F
          JZ= out      ;减到0后跳出循环
          T DW3
          JU= add      ;否则,继续循环
out   :   L DW1        ;取结果
          T QW112      ;写入输出模块
          BE
```

图12-5 简单过程控制的 STEP 5 STL 程序

图 12-5 所示的 STEP 5 程序中：

1）计算误差值。

2）将误差值与乘数相乘。STEP 5 中没有乘法指令，所以程序使用误差值循环相加的方法来实现乘法。（注意：为了提高可读性，本程序在相应的数学运算后未对误差值进行检查，所以当溢出时就会产生错误结果。）

3）将所乘的结果写到模拟输出模块的一个地址中。

图 12-6 所示的 Allen-Bradley PLC-5 程序中：

1）第一个梯级的指令从框架 0 的组 1 的左插槽（slot 0）的模拟输入模块中读取两个值。其中第一个值是数字量，代表了模拟设定值，作为字 1 存入了 PLC 的整型文件 7 中；第二

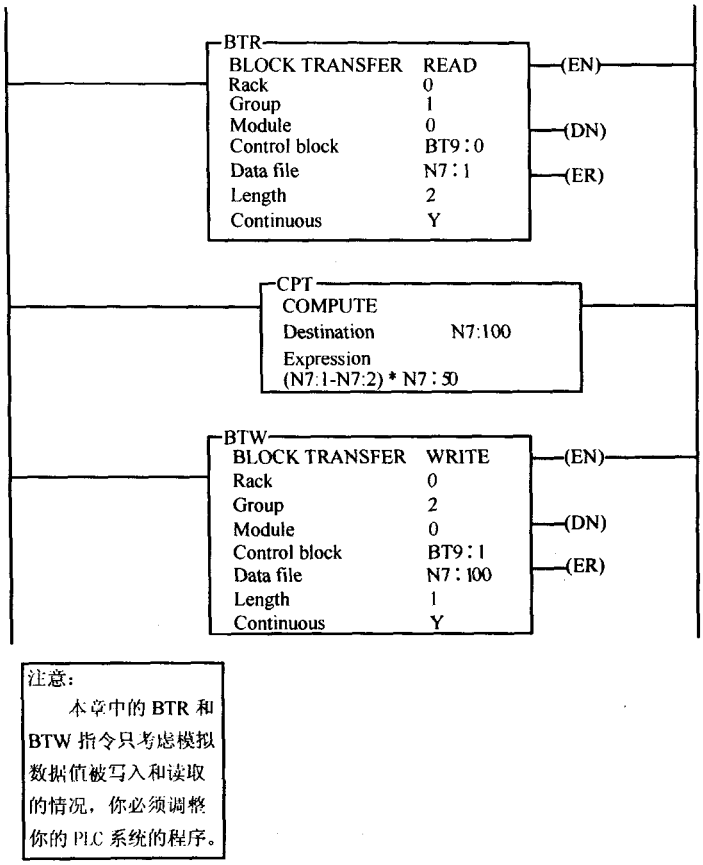


图 12-6 简单过程控制的 PLC-5 程序

个值是从传感器中得到的量化的模拟量，它作为字 2 被存入同一整型文件中。一旦模拟输入模块提供了已量化的数据，块传送读指令就会进行读取动作。

2）第二个梯级是进行简单的过程控制计算。

3）最后一个梯级是块传送写指令，将存储在 N7：100 的结果写入模拟输出模块，地址是框架 0 的组 2 的右插槽（slot 0）。一旦模拟输出模块可写入，就执行块传送写指令。

12.4 改进 PLC 程序在过程控制中的性能

有很多因素使得 PLC 程序员需要编写一些复杂的程序，而不是只使用前面我们所提到的一些简单的比例过程控制的程序，不同之处如下所示。

1）输入值和（或）输出值可能需要标度变换。

2）输出值要介于最大值和最小值之间。

3）控制程序有可能需要按设定的时间间隔来执行。

4）计算中可能会用到其他变量（比如去补偿外界条件）。

5）需要用到更复杂的算法，而不是简单的减法和乘法（比如 PID 算法或模糊逻辑算法）。

6）有时需要操作人员对系统进行手动控制。

7）在不同的条件下，控制输出的计算结果可能会不同。

12.4.1 过程变量和控制变量的标度变换

过程变量在与设定值比较前有时需要进行标度变换。如果设定值采用工程单位的话，那么对被控系统的操作就比较容易理解。例如，位置控制的工程单位可以是英寸，所以当PLC 的设定值为 15 时，就表示位置控制器应当把负载驱动到 15 英寸的位置。当负载在 15 英寸的位置，如果通过模拟输入模块从位置传感器上读入的过程变量值不是 15，就需要进行标度变换。有时即使不使用工程单位，标度变换也是必要的。例如在这样的情况下，电位器提供 0 到 10V 的一个直流电压作为位置设定值，而位置传感器提供的信号范围是 0 到 24V。这时，过程变量必须经过标度变换后才能和设定值进行运算，否则就会得到错误的结果。

控制变量值的标度变换是为了将通过控制算法得到的输出值调节到执行器所能接受的信号范围内。例如，当模拟输出模块要将 13 位有符号数值转换成执行器所需的±10V 直流信号时，就需要对从控制算法中得到的 16 位数据进行标度变换，变成 13 位的数据。为了将控制变量转换成工程单位的数据进行显示，我们还需要一次标度变换。这个标度变换需要两次计算，一个是乘法，一个是加法。乘法是用来调整未缩放值的范围，加法是对调整后的范围进行补偿，以达到所要求的限定范围。

如图 12-7 所示，对一个过程变量进行标度变换，以便与以千分之一英寸为单位的位置控制设定值相匹配。位置控制系统将负载驱动到与中心位置相距±5.000 英寸的范围内（设定值的范围是－5000 到＋5000）。这里的传感器是标有 12 英寸刻度的线性电位器，它的输出直流电压范围是 1 到 9V，对应范围为 10 英寸的负载的移动位置。模拟输入模块把 1 到 9V 的直流信号转换成 204 到 1844 之间的无符号二进制数。将取值范围在 204 到 1844 之间的过程变量（PV）标度变换成范围在±5000 之间的数值。

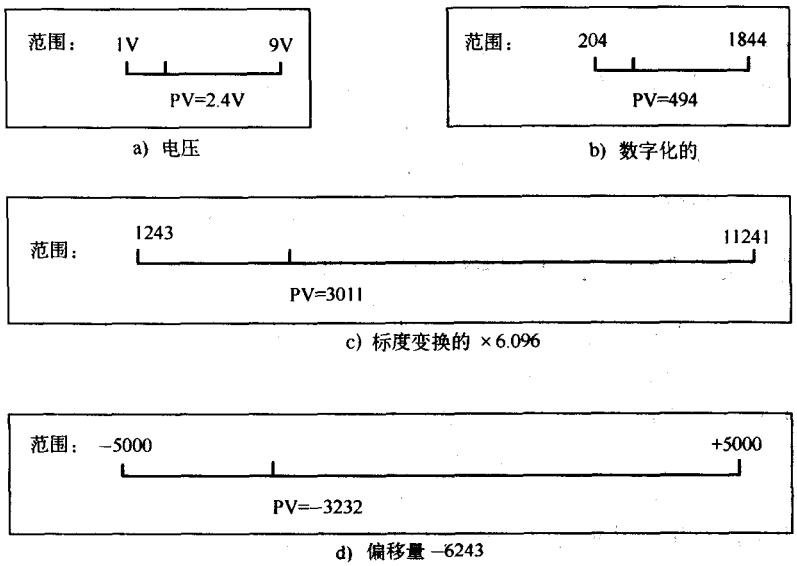


图 12-7 标度变换前后的过程变量：a）从位置传感器得到的电压；b）从模拟输入模块得到的未变换的数据；c）乘以 6.096；d）再加上值为－6243 的偏移量

1）乘法：

$$PV\times\frac{\text{输出范围}}{\text{输入范围}}=PV\times\frac{-5000-(+5000)}{204-1844}=PV\times\frac{-10\ 000}{-1640}=PV\times 6.096$$

注意：这里的负号在运算过程中抵消了，但不是每次都会这样；计算时不要把方向搞反了。

如果你选用的 PLC 不能进行浮点运算，那么就要对从传感器得到的输出电压进行调整，这样就可以用整型的乘数了。

2）加上偏移量：

$PV+$［输入范围的上端－（倍乘的输出范围的上端）］

$=PV+[-5000-(204\times 6.096)]=PV+(-5000-1243)=PV+(-6243)$

图 12-8 所示的梯级添加到图 12-6 的 Allen-Bradley PLC-5 程序中，就可以对存放在 N7：2中的过程变量值在带入到过程控制算法前进行标度变换。注意：乘以浮点数后的结果又存回 N7：2 中，N7：2 所在的内存区域存放着 16 位有符号整型数。PLC-5 将会以浮点形式进行计算，然后将结果转换为有符号整型数存储起来，这样会存在很小的截断误差。

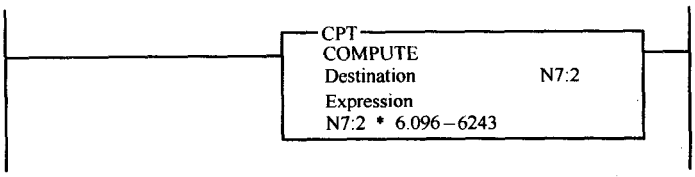


图 12-8 PLC-5 梯形图梯级：对尚未带入到过程控制算法的过程变量进行标度变换

有的智能 I/O 模块带有对模拟量标度变换的功能。如果你要使用的过程控制指令也有标度变换的功能（如本章后面将要讲到的 PID 指令），那么就不用在 I/O 模块中进行标度变换。因为任何额外的计算都会占用 PLC 的执行时间，而且也存在会产生更大误差的可能性。

12.4.2 对过程变量进行限幅

如果通过过程控制计算得出的控制变量的输出值的范围对于模拟输出模块来说太大，或输出的模拟输出信号的范围对于执行器来说太大，那么可以通过使用标度变换在把它们写入模拟输出模块前缩小控制变量；但这样做，信号的强度也随之减弱。一个比较好的解决办法就是对控制变量的最大值和最小值进行限制，保证执行器系统既不会被过度驱动，同时也能很好地允许系统对微弱的误差进行响应。事实上，如果在过程控制计算后对结果进行限幅，那么就可以采用更大的比例系数的乘数，这样对微弱的误差的响应也会更大，而且当误差值很大时还能保证执行器不被过度驱动。

举个例子，假设我们使用的是一个 10 英寸刻度的位置控制系统，它的电动机控制器的控制信号取值为＋8 到－8V 直流信号之间。我们可以对控制变量进行标度变换，那么当负载距离它应在的位置－10 英寸时，模拟输出值将会是＋8V（当位置误差是＋10 英寸时为－8V）。但是，如果位置误差为 0.010 英寸时，控制系统就只能提供＋0.008V 直流信号去消除位置误差。另一个方法是，我们选取一个很大的比例因子（比如 80，那么当位置误差是 0.10 英寸时，将会得到最大输出值），并且我们设定控制系统的输出范围仍在＋8 到－8V 之间。这时，当误差为 0.010 英寸时，执行器将得到一个 0.8V 的直流信号——是刚才我们只采用标度变换时强度的 100 倍，同时，当误差为 10 英寸时，输出也只有 8V。

如果在图 12-4 和图 12-6 的 Allen-Bradley PLC-5 程序中，整个过程控制计算的结果是通过模拟输出模块输出，而这个模拟输出模块会将 13 位有符号数（－2048 到＋2047 之间）转换成±10V 之间的直流信号，我们就要把这些数值的大小限制在±1638 之间，为的是不让模拟量模块产生超过±8V 的直流电压。将图 12-9 的梯形图插入到过程计算语句和模拟输出模块指令之间，就可以完成限制范围的工作。（由于 1638 和 8V 直流电压之间没有什么明显的联系，所以，如果能把计算得到的控制变量转换成以伏特为单位的数值，那么程序的可读性就会更高。）

很多 PLC 都有预编的 PID 指令，通常可以对输出进行限幅，这是该指令的一项功能，我们将会在后面看到。

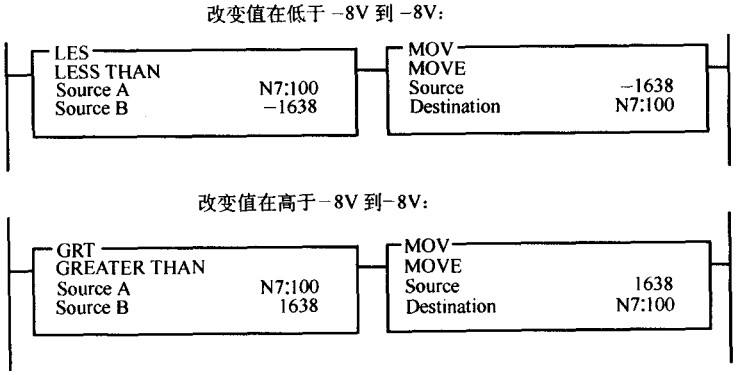


图 12-9 对计算后得到的控制变量输出值进行限幅

12.4.3 减少过程控制的扫描时间延时

过程控制的程序需要按照固定不变的时间间隔执行。有些 PLC 提供的预编过程控制指令（如 PID 指令）内嵌有计时功能，能够控制自身的执行速度。如果有必要的话，程序员可以设置一个计时器来控制过程控制计算按可测的时间间隔执行。

在一些实际应用中，扫描时间延时仍是一个问题。PLC 在同一时间内只能执行一条用户程序的指令，对需控制的状态做出反应总是存在一定的延时。那是因为：

1）扫描循环造成延时。PLC 从输入模块把数据读入输入映像存储区域，然后才执行用户程序。程序执行后，PLC 再将程序结果从输出映像存储区拷贝给输出模块⊖。这样，PLC 做出的反应总是比被感知的状态晚几个毫秒，有时甚至会因此而变成一个错误的结果。

2）PLC 程序的执行时间不尽相同。有些操作是要通过逻辑运算来判断是否要执行的，当逻辑真时，程序的执行时间就会长些，尤其是此后有跳转到某个子程序或文件处理指令。

3）一些 PLC 对 I/O 模块的扫描与它们的扫描循环是异步的。对 I/O 口的扫描是由一个单独的微处理器完成，而扫描循环是由 PLC 的主处理器来执行的。所以当进行过程控制计算时，不可能知道那些输入数据是什么时候得到的，也不知道在什么时候结果会被送到输出模块。Allen-Bradley PLC-5 使用块传送指令同模拟输入、输出模块交换数据。由块传送模

⊖ 更新的 PLC 可以在程序执行时直接从模拟输入模块读取输入数据，并直接写输出数据到模拟输出模块，以响应减少扫描循环延时。

块指令引起的通信请求会排队等待逐个被执行，其顺序是由程序决定的。⊖

4）模拟量与数字量之间的转换不是即刻完成的。当模拟 I/O 模块正在进行转换时，其过程是不可控制的。在图 12-10 的例子中，每当模拟输入模块的前一次的读操作完成时，这个 Allen-Bradley PLC-5 程序就会向该模块请求新的一组输入数据，而该模拟输入模块是当所有的输入通道的模数转换完成后，才会将新一组数据传给 PLC-5 的 CPU。这样的延时是不容易被程序员所发现的。

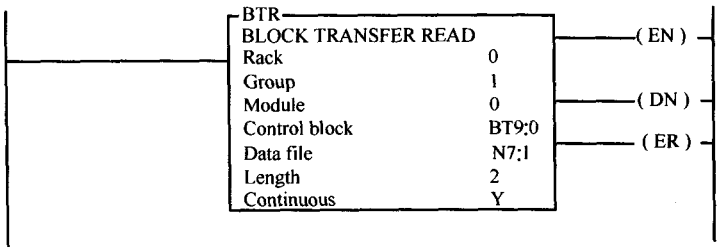


图 12-10 Allen-Bradley PLC-5 块传送指令从模拟输入模块读取数据

控制系统不是完全可信赖的，因为未受控制的可变因素具有不确定性。下面列举出一些方法可以让 PLC 用户减少（但不是消除）延时和不确定性。

1）选择那些工作频率适合被控过程的 PLC 和 I/O 模块。（不是所有的被控过程都需要速度很快的控制器）有的 PLC 可以在 5ms 内将整个程序执行一遍，而典型的 PLC 扫描时间大概是 25ms。当然，随着程序大小不同和选用的指令不同，这些数是可变的。指令执行时间一般在 PLC 手册中可以查到，所以用户可以计算（或估计）出程序的扫描时间。图 12-11 列出了一些 Allen-Bradley PLC-5 指令的执行时间等信息。

2）使用智能过程控制 I/O 模块。有些被控过程需要一些 PLC 的 CPU 无法提供的实时控制，所以 PLC 生产商往往会推出一些用于过程控制的智能 I/O 模块。这些模块插入标准的 PLC I/O 插槽，接收来自用户程序的命令，提供用户程序可读取的状态信息，同时也利用自带的微处理器直接控制与其连接的执行器和传感器。PLC 生产商如今正在研制新一代的 PLC CPU 模块，其中嵌入了专门负责过程控制（比如电动机控制）的微处理器。

3）程序员尽量编写效率更高的程序。如果从占用的存储空间来考虑，程序员就会通过跳转、调用子程序和中断（在本书其他地方都将讲到）等一些结构化的编程技术来把程序编写得简洁、短小，但这样的程序就会比那些不用跳转等指令的、可能会更长的程序耗去更多的执行时间。图 12-12 所示的两个 STEP5 STL 程序做的是同样的事情：将两个数据块中的数据字相加，如果其和大于 15 就打开输出端口（在这个例子中选用 STL 是因为它很接近低端的机器语言）。其中一个程序只用了 15 条指令，另一个是 20 条，但那个有 20 条语句的程序执行所用的时间更短，因为它不包含要调用其他子程序的指令。一些编程语言（非 PLC 的）的编译器允许程序员选择将程序编译成紧凑型还是快速型的。在不久的将来，PLC 编程软件也将能够提供这种选择。

4）一些 PLC 编程语言本身就能生成相对高效的机器语言代码。高级的图形式语言，比如

⊖ 除了在 STI 或 PII 程序中的块传输指令。因为 STI（或 PII）程序在 PLC 执行下一个指令之前暂停并等待块传输完成，所以这些块传输指令是与程序同步执行的。

分类	代码	标题	执行时间（μs） 整数		执行时间（μs） 浮点数	
			True	False	True	False
逻辑	AND	与	5.9	1.4		
	OR	或				
	XOR	异或				
	NOT	非	4.6	1.3		
	MOV	移动	4.5	1.3	5.6	1.3
移动	MVM	掩模移动	5.2	1.4		
	BTD	位移动	10.0	1.7		
比较	EQU	等于	3.8	1.0	4.6	1.0
	NEQ	不等于	3.8	1.0	4.3	
	LES	小于	4.0	1.0	5.1	
	LEQ	小于或等于				
	GRT	大于				
	GEQ	大于或等于				
	LIM	极限测试	6.1	1.1	8.4	1.1
	MEQ	如果等于，屏蔽比较	5.1	1.1		
比较	CMP	所有	2.48+(Σ[0.8+])	2.16+Wr [0.56]	2.48+(Σ[0.8+])	2.16+Wr [0.56]
计算	CPT					

在处理器的数据表中查用大小的效率可以超过 2048 的字
t= 处理 CMP 或 CPT 的每条指令的执行时间
Wr= 在 CMP 表达式中被比较使用的内存字的数目
注意：CMP 或 CPT 指令用加直接寻址来计算

图 12-11 PLC 5 指令执行时间

```
OB1
主程序调用其他程序
      R Q7.0 ,初始化标志
      C DB1 ,第一数据块
      JU PB1 ,第一次跳转
      C DB2 ,第二数据块
      JU PB1 ,第二次跳转
      BE
PB1
在这程序中的了程序
      L DW1
      L DW2 ,过程数据
      +F        ,来自两个数据
      L DW3 ,块
      +F
      L KY+15
      >F
      S Q7.0
      BE
```

a)

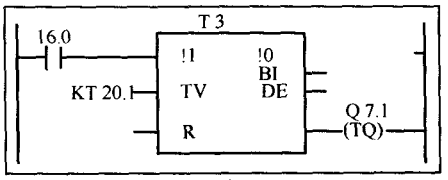
```
OB1
            在一个主程序中的整个程序
      R Q7.0 ,初始标志
      C DB1
      L DW1 ,过程数据来自
      L DW2 ,第一个数据块
      +F
      L DW3
      +F
      L KY+15
      >F
      S Q7.0
      C DB1
      L DW1 ,过程数据来自
      L DW2 ,第二个数据块
      +F
      L DW3
      +F
      L KY+15
      >F
      S Q7.0
      BE
```

b)

图 12-12 a）短程序；b）快程序

梯形图，在被 PLC 执行前必须被编译（翻译）成机器能够识别的指令。编译后的程序中往往会含有一些不需要的指令。图 12-13a 所示的 Siemens S5 梯形图中，程序员选择的有三个端口

的开延时定时器未使用，而图 12-13b 所示的对应的 STL 程序中，表明对于每个未用到的特性它需要一条额外的 STL 指令（注：NOP0 的意思是“没有准备”。Siemens 的编译器可以有足够的智能使没有用到的功能的不需要的机器语言代码最少，但却不能完全消除。）如果程序员用 Siemens 的 STL 语言编写程序时，这三个 NOP 0 指令就可以省略了，如图 12-13c 所示。



a)

```
A     16.0
L     KT20.1
SD    T3
NOP   0      ;
NOP   0      ;   这三个程序
NOP   0      ;   不需要
A     T3
=     Q7.1
```

b)

```
A     16.0
L     KT20.1
SD    T3
A     T3
=     Q7.1
```

c)

图 12-13 STEP 5 定时器指令：a）梯形图形式；b）对应的 STL 形式；c）高效的 STL 程序

5）立即 I/O 指令（也叫直接访问 I/O）可以减少读入过程中变量当前值和输出新的控制变量值之间的延时。在第 11 章我们已经讲到，立即 I/O 指令可以让 PLC 直接从输入模块读取时间，也可以直接将输出数据写入输出模块。使用立即 I/O 指令也无法完全消除延时，因为 PLC 仍需要通过过程控制计算（包括标度变换以及对最大值和最小值的限制等操作）得到控制变量值，而且模拟 I/O 模块也需要时间来进行模拟量与数字量之间的转换。

1. Allen-Bradley PLC-5 中用于过程控制的立即 I/O 指令

PLC-5

PLC-5 中的立即 I/O 指令对于模拟过程控制不起作用。因为它们只能读写数字 I/O 模块，而且是同一框架中的。如果把读写模拟 I/O 模块的指令——块传送读指令和块传送写指令写入到中断服务程序中，比如定时中断程序（STI），那么它们就可以被立即执行。

2. Allen-Bradley SLC 500 中用于过程控制的立即 I/O 指令

SLC 500

在 Allen-Bradley SLC 500 系列 PLC 中，使用带掩模的立即输入和带掩模的立即输出指令可以对模拟 I/O 模块立即读写。如果有必要的话，在标准指令中使用 M0 和 M1 模块寻址特殊的 I/O 模块。

3. SIEMENS S5 中用于过程控制的立即 I/O 指令

S5

Siemens S5 PLC 允许使用强制立即访问 I/O 的方式来直接寻址访问[⊖]。当把地址的前缀换成 P 而不是 I 或 Q 时（例如，PW072 而不是 QW072），程序和使用输入、输出映像表中的数据是一模一样的。图 12-15 的程序中包括直接寻址访问。

4. SIEMENS S7 中用于过程控制的立即 I/O 指令

S7

Siemens S7 PLC 允许使用强制立即访问 I/O 的方式来对 I/O 外围寻址。只有使用 I/O 外围寻址时，模拟模块才能被访问到。当把地址的前缀换成 PI 或 PQ 而不是 I 或 Q 时（比

⊖ 在 S5-103U PLC 中，直接访问地址只能用在定时中断服务程序（OB13）和硬件中断服务程序（OB2）中，或者在这些中断服务例行程序执行时调用的程序块中。

如，PQW072 而不是 QW072），那么程序就和使用输入、输出映像表中的数据一模一样了。

CQM1

5. OMRON CQM1 中用于过程控制的立即 I/O 指令

在 CQM1 程序中，I/O 更新指令（IORF（97））可以用来在指令指定的地址范围内立即读和写 I/O 模块，包括模拟 I/O 模块。如果 PLC 被配置为直接更新（见第 11 章），那么，一旦所有输出存储空间的数据改变时，这些数据就会被全部拷贝到输出模块。

12.4.4 定时中断

我们在第 11 章中看到，有些 PLC 通过配置可以定时执行中断服务例行程序。用定时中断服务例行程序编写的过程控制程序提供的是一种确定性的过程控制。使用了立即 I/O 指令的定时中断程序既有确定性又有快速响应的特性。

PLC-5

1. Allen-Bradley PLC-5 的定时中断在过程控制中的应用

通过配置，Allen-Bradley PLC-5 可以按照一定的时间间隔运行可选择的定时中断程序（STI）文件。PLC-5 的 CPU 在模拟输入/输出模块间的数据的块传送（BTR 和 BTW）通常与执行完块传送指令后的扫描循环是异步发生的，但定时中断程序中的 BTR 指令和 BTW 指令的优先级将会高于任何其他排队等候处理的块传送请求，而且当 STI 程序执行到块传送指令时，它会停下来等待块传送完成后再继续向下执行。当然，程序员必须保证 STI 的调用频率不能太快，要给模拟输入模块留有足够的时间完成模拟输入量与二进制数值之间的转换运算。图 12-14 所示的是一个 STI 程序文件中的块传送读指令，只有当 BTR 完成后，其下面的语句才会开始执行。

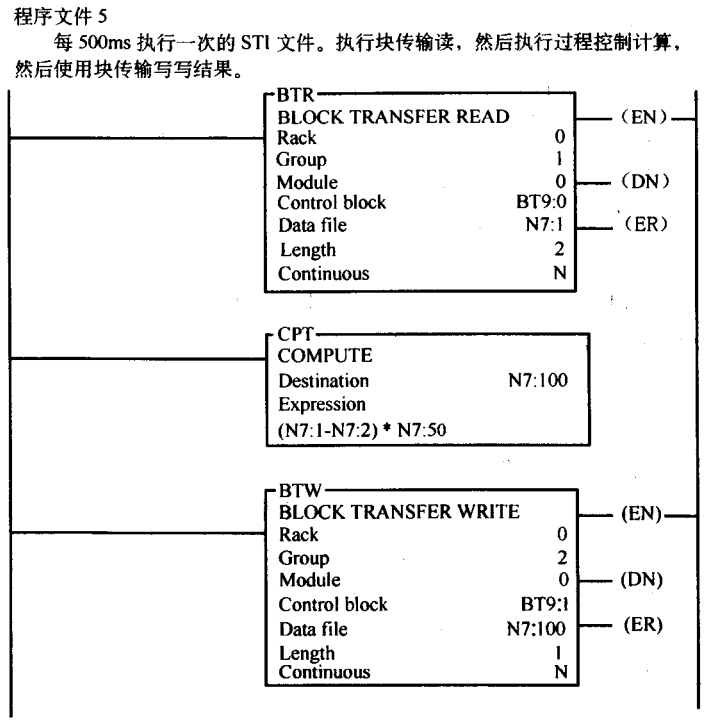


图 12-14 按设定的时间间隔执行过程控制功能的 PLC-5 STI 程序

PLC-5 CPU 与远程的 I/O 模块进行块传送的速度会明显低于与 CPU 处于同一框架中的 I/O 模块之间的速度，所以尽量把模拟 I/O 模块安装在 CPU 附近，这样会使控制的效果更好。

2. Allen-Bradley SLC 500 的定时中断在过程控制中的应用 SLC 500

通过配置，Allen-Bradley SLC 500 可以按照定时间隔来运行 STI 程序文件。要求按固定时间间隔执行的过程控制程序需要被写入到 STI 文件中，可以配置成当程序进入 STI 程序时来执行 PID 指令，也可以利用 PID 内嵌的定时器来控制执行间隔，这样就不用将其编入 STI 中了。

3. Siemens S5 的定时中断在过程控制中的应用 S5

Siemens S5 PLC 能够自动地按定时间隔调用组织块 13（OB13）中的程序。第 11 章我们讲过，这些规定的间隔值保存在保留系统字 97（RS97）中。如图 12-15 所示为使用 OB13 的过程控制程序。

```
FB1
    保留系统字只有在功能块中才能被写入,所以这个功能块要在 PLC 启动时被用户程序调用。
    L KF 40;取十进制数(KF)40到累加器1
    T RS 97;将此数传给保留系统字97,配置定时中断每40×10ms 发生一次;如果没有值写入,则默认为
10,即定时中断每(10×10ms= )100ms 发生一次
    BE      ;结束 FB1
OB13
    此块中的任何程序都将会自动地按照设定的时间间隔执行,时间间隔值由写入到 RS97中的数据(现
在400ms)来决定。
    L PW072;通过直接寻址访问方式从模拟输入模块中读取两个数据
    L PW074;
    JU PB1 ;假设 PB1是所需的过程控制计算
    T PW112;将输出值写给输出模块
    BE
```

图 12-15 使用定时中断的 STEP5 程序

第 11 章中我们详细讲述了 S5 PLC 如何执行定时中断程序，以及定时中断程序中包含直接地址访问（前缀为 PW 而不是 IW 和 QW）的情况。大型 S5 PLC 与中型 S5 PLC（比如 S7 103U）的工作原理是不同的。

4. Siemens S7 的定时中断在过程控制中的应用 S7

大部分的 S7 PLC 都提供循环式中断，它们能够定时中断主程序的扫描，定时值在配置 S7 CPU 模块时设定（见第 10 章）。OB30 中的程序就是循环式中断例行程序。

5. OMRON CQM1 定时中断在过程控制中的应用 CQM1

CQM1 既有循环式中断，又可以对主扫描循环程序设置最短扫描时间，所以程序员既可以把过程控制程序放到循环式中断子程序中也可以把它放到主程序中。OMRON 的预编 PID 指令中带有的定时器与定时中断的使用相冲突，所以在编写 OMRON 的程序时，不要把 PID 指令写入定时中断程序中。

对于有些 PLC 系统，把 I/O 模块与 CPU 模块的数据交换设定为定时交换或严格地要在几秒钟内完成的情况是可能的。当 I/O 模块检测到一个需要立即执行控制程序的事件时，I/O中断就会立即去运行所需的过程控制程序。

PLC-5

6. Allen-Bradley PLC-5 I/O 模块控制的数据交换

在 Allen-Bradley PLC-5 系统中，有的模拟输入模块可以配置为定时地进行模拟输入量到数字量的转换，而不去理睬 CPU 模块发来的块传送读（BTR）请求，但当数据改变时例外。如果 PLC 主控制程序（MCP）中含有连续型 BTR 指令，那么输入模块就定时给 CPU 传送数据。如果，只有当新的数据从输入模块传来时，MCP 才执行过程控制计算，那么输入模块又可控制程序的执行时间间隔。既然 MCP 的执行时间间隔不能被控制，而 MCP 发出的块传送请求也要和其他的块传送请求一起等待执行，那么控制程序就不如 STI 程序具有确定性了。但是，往往一些控制应用实例要求有一定的确定性。

图 12-16 为配置模拟输入模块来允许定期地执行块传送指令时所出现的配置画面。在实时采样率后输入 0.5，模块就会在响应一个 BTR 请求后过 0.5s 再响应下一个。图中还画出了需要用来初始化 BTR 的指令。使用连续型块传送读指令时，前一个块传送读操作完成后发出新的请求，但模块会将前后两次请求的响应间隔控制为 0.5s。

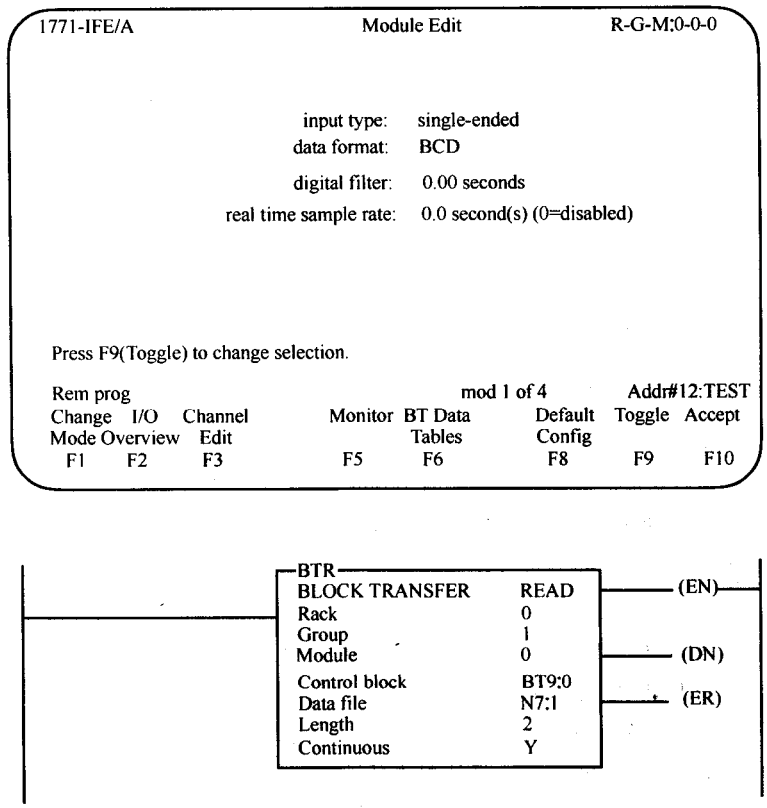


图 12-16 配置 PLC-5 模拟输入模块定时给块传送语句提供数据的配置画面，其中的 BTR 指令要求连续型块传送

连续型块传送指令不能在中断服务例行程序中使用。但是，假设 STI 例行程序中有（非连续型）BTR 指令，用来从模拟输入模块中读取输入值，而这个输入模块已经配置为按一定的时间间隔来产生数字量数据的（此时间间隔大于 STI 的时间间隔），那么这个模拟输入模块就会在 STI 执行时强行让 STI 去等待数据。虽然这样在每次执行过程控制程序时，

都会浪费一些时间，但是，由于输入采样时间间隔被严格控制，过程控制程序就具有最大的确定性了。

在 PLC-5 中，事件驱动中断可以让程序与输入模块检测到的外部事件同步。在图 12-17 的例子中，当框架 1 中的模块 4 的位 5 两次为真后，PLC-5 就会中断它的扫描循环而立即运行文件 6——过程输入中断程序（PII）。Rockwell 建议不要在 PII 程序中使用立即 I/O 指令和块传送指令，所以使用 PII 来包含事件驱动型控制程序的用途不是很大。

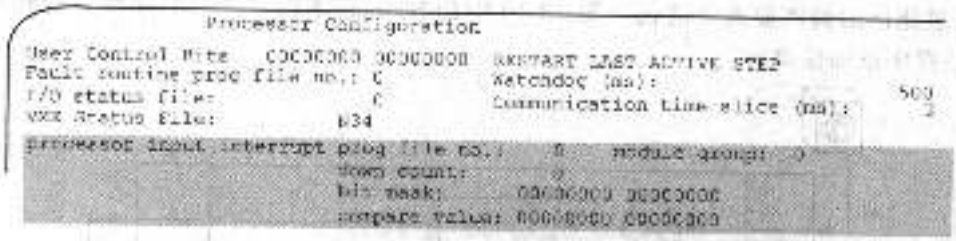


图 12-17 PLC-5 中用 PII 进行过程控制的事件驱动中断配置

7. Allen-Bradley SLC 500 I/O 模块控制的数据交换

SLC 500

通过配置，来自 I/O 模块的两种中断均可被 SLC 500 PLC 响应，或者配置成只响应其中的一种。在第 11 章中我们讲到，特殊 I/O 模块可以初始化两种中断请求：离散输入中断（DII）和 I/O 中断。这两种中断都可以用来初始化过程控制程序。

8. Siemens I/O 模块控制的数据交换

S5 S7

Siemens S5 和 S7 中也可以提供事件驱动中断功能，但是要求必须安装能够产生中断信号的输入模块。一旦安装并配置好后，I/O 模块就能根据所选择的情况（依模块而定）发出硬件中断信号，当硬件中断信号出现后，S5 PLC 就会执行组织块 2（OB2）中的程序（前提是 PLC 未关闭中断），S7 执行的将是 OB40 中的程序。程序员可以在事件驱动中断服务程序中编写与定时中断中一样的程序，包括立即 I/O 访问的使用。

12.4.5 输出量计算中的其他量

本章到目前已讨论过的过程控制计算涉及到的控制输出量是与设定值和过程变量的差值成比例的。当误差为 0 时，控制变量也变成 0。但当增添了偏移量后，情况便有所改变。有了偏移量，当计算所得的误差为 0 时，输出将不再是 0。

在控制变量的计算中添加偏离量、扰动量或前馈量，可以使控制输出信号发生偏移。偏离量、扰动量和前馈量的用法相似，而且名称有时可以互换，但它们还是会体现出所需偏移量的一些细微的不同之处的。

1. 偏离量

偏离量是在控制变量输出值的计算中加入的给定值，通常是在所有其他计算都完成后才加入的。图 12-18 给出了一个需要加偏离量的情况。直流电动机的速度是被控变量，速度传感器提供与电动机的实际速度成正比的过程变量。图 12-18a 中，没有加入偏离量，PLC 利用控制变量输出信号来控制电动机的速度，其中控制变量是由速度误差值乘以比例增益（G）得到的。控制变量通过电动机控制器控制电动机的扭矩。这样的控制是有缺陷的，因为当电动机的速度刚好为所要求的时，控制变量将变成 0，这样，电动机扭矩也就变成了 0，

那么电动机的负载就会把其速度拖慢。

为了让图 12-18a 中的速度控制器正确地工作，需要加入一个速度偏离量。在图 12-18b中，偏离量是由手动控制的电位器提供的。当电动机正在运行时，操作员调整好电位器来弥补能量损失，然后就不再对电位器进行操作了。当电动机的速度达到要求时，误差值就会为 0，而 PLC 输出的控制变量的值就会等于偏离信号的值，正好能够保证电动机的速度。当速度传感器告知电动机的速度与设定值有偏差时，控制变量值就会与速度误差成比例的偏离偏离量的值。图 12-19 中的 Siemens STEP 7 STL 程序告诉我们如何给 PLC 程序添加偏离量。

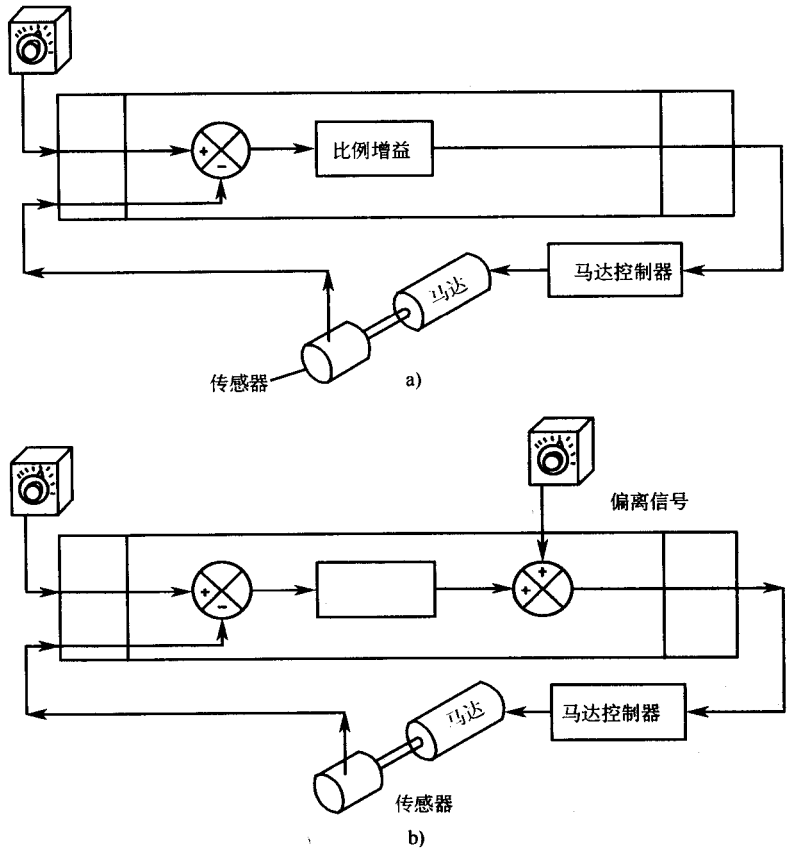


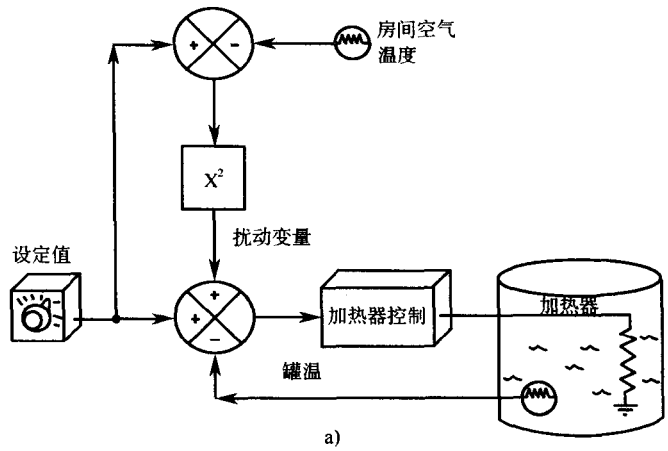
图 12-18 电动机速度控制：a）无偏离量；b）有偏离量

```
L PIW072 ;读取整数设定值
L PIW074 ;读取反馈的整型过程变量
-1       ;计算误差
L DW4    ;载入整型比例常数
*1       ;计算比例控制变量
L PIW076 ;读取偏离量
+ 1      ;将偏离量加到控制变量中
T PQW112 ;将最后结果写入执行器
BE
```

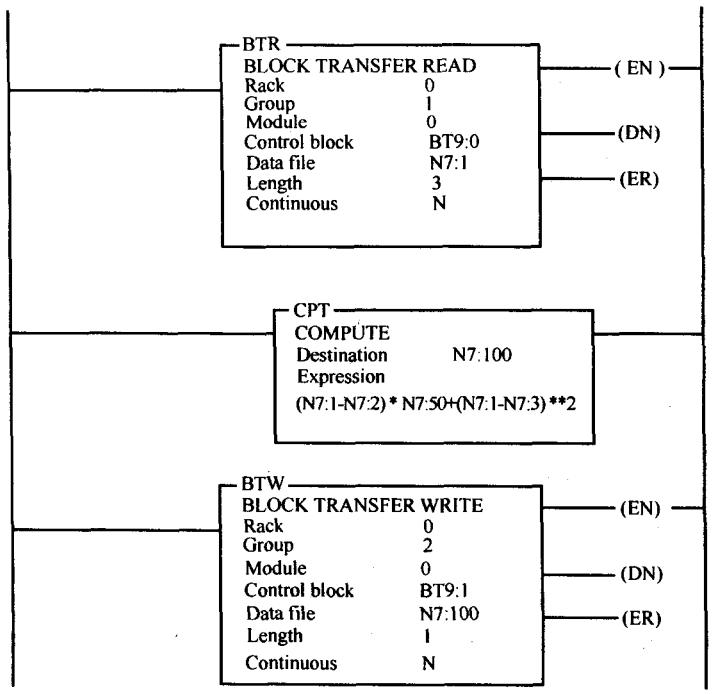
图 12-19 使用偏离量的 STEP7 STL 程序

2. 扰动量

与偏离量相似，在控制程序中起修改控制变量输出值的作用，而且一般也是添加到所有其他计算之后。典型的扰动量是来自处在被控系统所在环境中的传感器（扰动量有时也被叫做环境变量）。在图 12-20a 中的温度控制实例中，加热液体的容器内外各安置了一个温度传感器。容器的热量损失与室温和容器温度的差值的平方成正比，所以加热器提供额外的热量来弥补那些散失的热量。这里我们让扰动量的值等于设定值与室温的差值的平方（即当容器温度为设定值时，所需的用来维持温度的能量）。图 12-20b 给出了利用另一个传感器计算扰



a)



b)

图 12-20 a）补偿环境温度的温度控制器；b）按环境温度输入量计算扰动量的 PLC-5 程序

动偏离量的值的 Allen-Bradley PLC-5 程序。从外部温度传感器读取第三个模拟输入量，现在的过程控制计算中多了：

+ (N7 : 12-N7 : 3) ^ 2

这个新算式添加了扰动量，即设定温度值 (N7 : 1) 与容器周围的空气温度值(N7 : 3)差值的平方。简单起见，我们假定空气温度不会超过设定温度值。(实际上还需要一个乘数来调节绝缘和加热系统的性能。)

3. 前馈量

前馈的定义目前说法不一。有些文献（比如 Allen-Bradley 用户手册）中，前馈是指偏离，实际上就是前面我们所讨论的偏离量。其他的生产商对前馈下的定义与环境变量类似，除了前馈传感器检测原料变化这种情况。如果检测到原料发生了变化，控制系统就会在误差真正产生前预算出误差值和正确值，此时，前馈量便会出现在所有过程控制计算前的误差计算当中。图 12-21a 给出了一个需要引入前馈的例子。例中的温度控制系统要控制水温，而且要适应引入水(即刚进入控制系统的水）的温度变化，所以这里需要传感器来测量引入水的温度。在本例中，根据引入水的温度而调整加热器的控制是必不可少的，因为一旦水经过了加热器再采取措施就为时已晚（实际上，本例不需要测量输出水（即流出控制系统的水）的温度，故未安装此传感器。)。送到加热器的控制变量信号与设定值和引入水的差值成正比。图 12-1b 给出了用 PLC-5 程序实现的过程控制计算公式。当引入水温度降低时，PLC-5 指令中的前馈项 (N7 : 1-N7 : 3) 便会增加。(有前馈环节的控制系统也可以再添加反馈环节。)

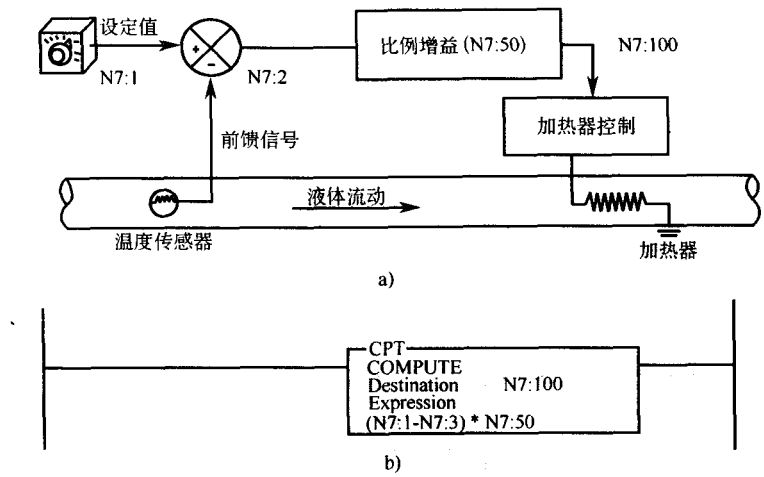


图 12-21 a) 有前馈环节的温度控制系统。b) 根据前馈量计算输出值的 PLC-5 指令

12.4.6 复杂的过程控制程序

PID 控制：有时，引入了偏移量的过程控制计算仍不能得到令人满意的结果，被控系统可能会有我们所不希望的稳态误差或超调误差。使用 PID 算式来计算控制变量能够消除或减少这些误差。

稳态误差，有时也叫残留误差或跟随误差（当被控系统的设定值为变化量时)。如图 12-22所示，当控制系统无法消除设定值和测量到的系统实际输出值之间的差值时，稳态误

差就会存在。比例控制系统中必定有稳态误差，因为当误差小到一定程度时，控制变量就会相应地小到无法控制执行器去消除这仅剩的误差。以电动机为例，当它把台架移动到离指定位置只有 0.002 英寸的距离以内时，电动机就无法产生足够的扭矩来克服摩擦力。这 0.002 英寸的误差，控制系统无法将其消除，便成了稳态误差。

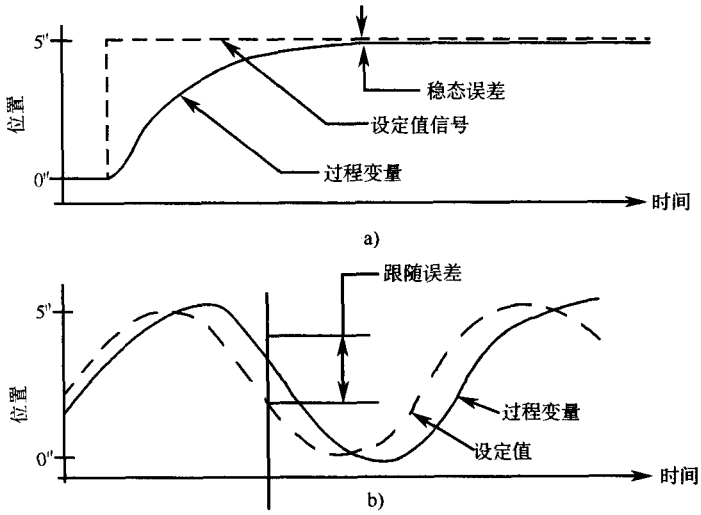


图 12-22 a）稳态误差；b）跟随误差

超调误差，也叫动态误差，有时是导致振动或不稳定的原因，见图 12-23。如果系统的响应动作过大和（或）被控系统的容性和惯性太大，致使当期望的输出已经达到了而被控系统的输出还在改变时，超调现象就出现了。假设读者要将一悬挂的装满酒的瓶子全速向一堵墙推去，当瓶子离墙只有 2 英寸时才去阻止它，想让它停下来，这时我们所不期望的超调就很明显了。但如果我们是以一个较低的速度，或用一个空瓶子，或在摩擦力很大的水中做这

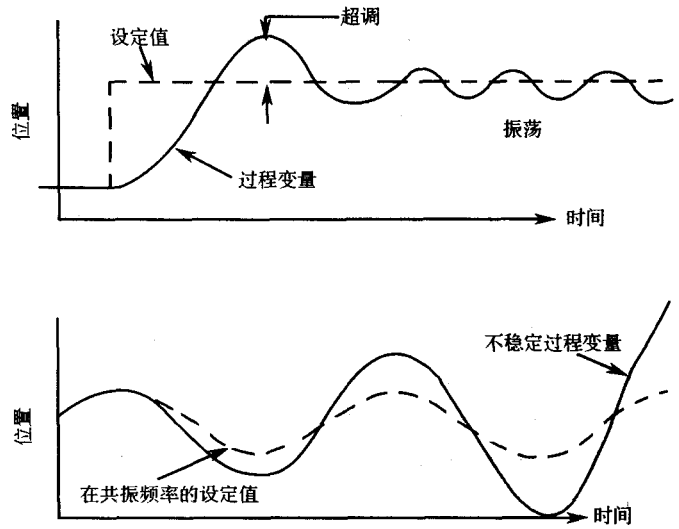


图 12-23 超调、振荡和振荡性的不稳定状态

件事，那么超调就会降低，而且结果也不会像刚才那么糟糕了。如果一个控制系统接收的设定值信号、扰动信号或前馈信号振荡的频率恰好是被控系统的共振频率，那么每一周期超调都会有所增加，最终就会导致系统的不稳定。

许多现代的 PLC 都有能被主程序调用的预编 PID 过程控制程序。PID 的意思是比例-积分-微分，这三种计算方式能够根据误差计算结果得到控制变量输出值。

图 12-24 所示的 PLC-5 控制程序，我们在前面已经见到过，不同之处在于原来的计算指令现在已经换成了 PID 指令。(我们现在只对 PID 作简单的介绍，于是在这里使用了 PLC-5 的 PID 指令，因为此指令看上去不是很复杂，虽然要正确有效地使用它并不容易。) 这个 PLC-5 的 PID 指令中有一个源地址，其中存储的是通过模拟输入模块从传感器读取的过程变量。这个指令还将产生一个控制变量，然后通过模拟输出模块传给执行器。控制块和手动输入值将在后面讨论。

图 12-24 中的 PLC-5 程序应当写入 STI 程序，这样就可以定时执行了。同以往的程序一样，图 12-24 的程序中也有 BTR 指令，用来从模拟输入模块中读取过程变量，然后 PID 指令计算出控制变量，再由 BTW 指令将其写入到模拟输出模块中。

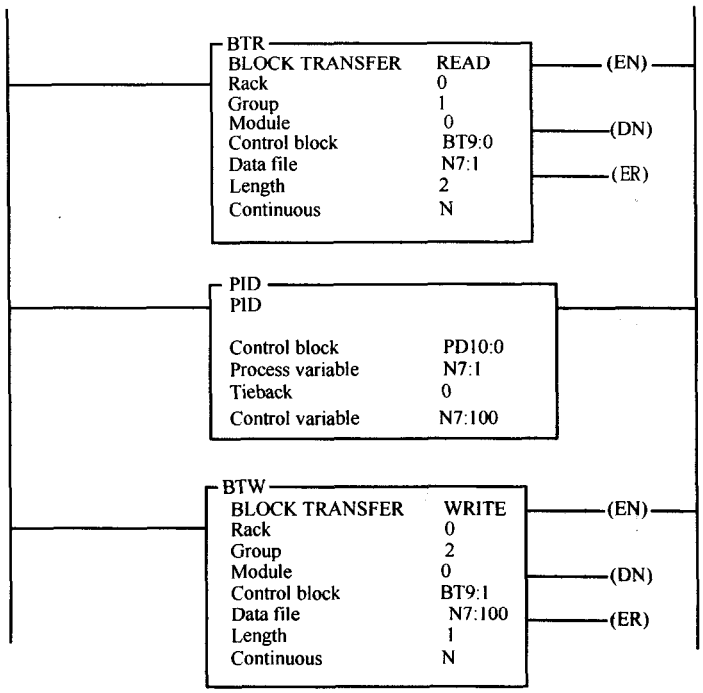


图 12-24 使用 PID 指令的 PLC-5 程序

在本章的开始，我们讲过一个闭环控制系统的方框图（图 12-2），那里，PLC 计算出误差值再乘以一个比例常数。使用了预编 PID 指令的 PLC 程序可以做更多的事情，如图 12-25 所示。PID 指令在比例计算的基础上不仅添加了积分、微分运算，还可以引入偏移量，对输出值进行范围限制，如果需要的话，还可以对过程变量和（或）控制变量进行标度变换。我们还将会看到积分、微分的引入还将使其他的一些改良成为可能。

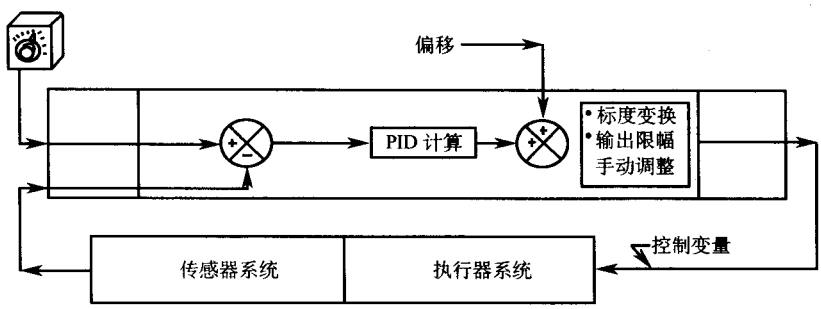


图 12-25 利用 PLC 进行 PID 控制的控制系统结构图

PID 指令能做什么

我们前面讨论过的比例控制所能做的工作是：

$$CV=K_p\times\text{误差}$$

其中 CV 是控制变量（来自 PLC），K_p 是控制器的增益（出现在 PLC 程序中），误差＝设定值－反馈值（出现在 PLC 程序中）。控制变量被转换成模拟信号后传给执行器，根据需要，执行器将其放大后转变成机械动力。如摩擦力和惯性等物理特性，都将影响控制系统的最终输出。如图 12-26 所示的系统中，电动机通过滚珠丝杠来控制台架的位置，设定值和过程变量的含义是期望的和实际的台架位置。当设定值有一个很大的改变时，位置误差就会变的很大，这时电动机就要产生一个很大的扭矩去驱动台架到期望的位置。随着台架越来越接近指定位置，误差也越来越小，电动机的扭矩也会变小，台架的速度也相应的减慢。最终，台架会停下来，但和设定值指示的位置会差一点，因为这时的电动机扭矩不足以克服摩擦力，稳态误差没被消除。调大比例放大系数，稳态误差会减小；也可以加大电动机的动力或换一个动力更大的电动机，或对 PLC 程序作一些修改。

不幸的是，增大比例增益也会增大比例控制系统的其他误差：超调。在我们的位置控制例子中，假如增大 K_p，电动机就会产生更大的扭矩和更快的速度去减小误差（当然要在各部件的承受范围内，而且也不能超过 PLC 程序给出的限制）。电动机的转子、滚珠丝杠和台架相应得会具有更大的动能。当误差变为 0 后，惯性会使台架继续运动而滑过（超调）指定位置，而减小超调的办法就是减小 K_p。

被控系统的特性可以由容性和阻性来描述。容性是对被控系统的吸收能量的能力的一种度量，它会阻碍控制器改变系统的现状（例如台架定位器的动能导致的惯性、加热系统中的容器的大小）。阻性是对被控系统的能量损失快慢的一种度量（例如摩擦力会损耗掉台架控制器的动能、加热容器的对外热辐射）。

在一些被控系统中，容性和阻性的平衡点就是寻找这样一个 K_p，能够使系统兼有可接受的稳态误差和超调量。但有些系统，根本找不到合适的 K_p 去减小这两种误差，或系统在运行时其特性一直在改变，导致能够满足要求的 K_p 的值也一直在变化。这时，比例控制就无能为力了，我们便需要借助积分和（或）微分来达到目的。积分控制可减小稳态误差，微分控制能减小超调量。既有比例，又有积分、微分的算式可以写成：

$$CV=K_p\times\text{误差}+K_i\times(\text{误差和})+K_d\times(\text{误差变化率})$$

其中，CV 指由 PLC 提供给执行器的控制变量输出值，K_p 为比例增益（无单位），K_i 为积分

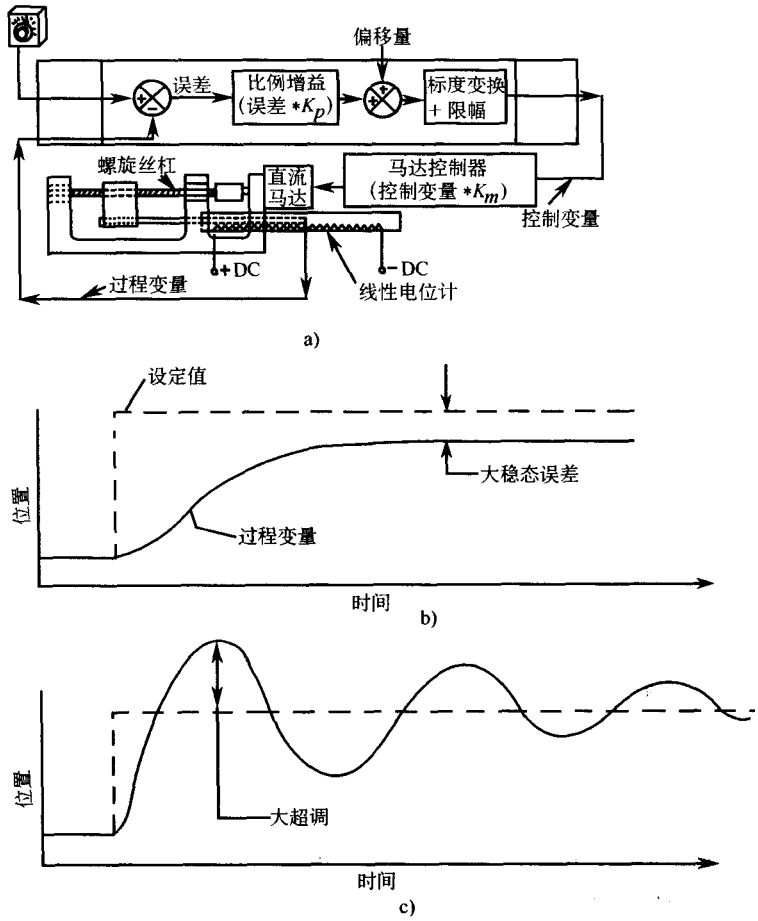


图 12-26 设定值发生阶跃变化后比例控制系统的响应：a）控制系统；b）当 K_p+K_m 较小时的响应；c）当 K_p+K_m 较大时的响应

增益（1/s），K_d 为微分增益（s）。（典型的 PLC PID 公式会更复杂，但我们先来讨论最基本的。）

下面是两个最常见的 PID 算法的形式（其差异我们将在后面讨论）。在有些 PLC 中，程序员可以为 PID 指令选择不同形式的方程。这两种 PID 方程形式包括：

1）*独立增益方程*，它和我们前面提到的那个方程在形式上很像。

$$CV=K_p(E)+K_i\int(E)\mathrm{d}t+K_d\frac{\mathrm{d}(E)}{\mathrm{d}t}$$

其中，CV 是控制变量输出，E 是误差，$\int(E)\,\mathrm{d}t$ 是误差和，$\mathrm{d}(E)/\mathrm{d}t$ 是误差的变化率。

2）ISA[⊖]标准的相关增益方程：

$$CV=K_c\left[E+\frac{1}{T_i}\int(E)\mathrm{d}t+T_d\frac{\mathrm{d}(E)}{\mathrm{d}t}\right]$$

这个方程中的大部分变量与前一个方程相同：K_c 是一个比例因子，影响着整个 PID 计算的结果；$1/T_i$ 是重置增益（重复次数/时间，T_i 有时也叫做积分时间）；T_d 是速率增益

⊖ ISA 是北美一个大型的广受尊敬的组织，提供过程控制工业的标准。

(时间)，有时也叫做微分时间。注意此方程中没有变量 K_p，所以输出中的比例分量不是独立分量。在整个 PID 计算完成后，由比例因子 K_c 对其结果进行标度变换。

PID 指令如何纠正误差

只要误差存在，积分量就会增加，所以当稳态误差未消除时，积分项就会使控制系统的输出稳步增加，直至控制变量大到稳态误差消除。PID 指令在每次重新计算 PID 结果时都按下面的方法来得到误差的和，即积分值。

新误差和＝前一次误差和＋（当前误差×距离上次计算的时间）

积分增益（K_i）和重置增益（$1/T_i$）将对积分输出值进行标度变换。

现在我们来看看如何使用积分控制。假设我们有一个比例控制位置系统，如图 12-26 中的系统。我们已经观察到当 K_p 取 1 时，台架将停止到距离期望位置 0.25 英寸的位置（由于有摩擦力），这时从 PLC 得到的模拟输出信号为 0.25V。既然 K_p 是 1，那么系统的未经放大的稳态误差就是 0.25V。图 12-27 给出了 PLC 比例控制器在台架停止后的模拟输出量。

如果系统的位置误差是 0.25 英寸，我们把比例控制公式换成 PID 独立增益方程，其中 K_p 和 K_d 置为 0，积分增益（K_i）设置为 $1s^{-1}$，那么 PLC 的模拟输出值就会从 0 开始，但会每秒增加 0.25V（因为只有误差存在，所以此值会等于上段中提到的那个输出值）。取 $K_i=2s^{-1}$，其他条件不变，模拟输出就会以每秒 0.5V（输出值的两倍）的速度增加。很明显，积分增益越大，控制变量输出值增加的就越快。图 12-27b 给出了只有积分环节作用时控制变量随时间的变化曲线，图 12-27c 给出了比例控制（$K_p=1$）和积分控制共同作用的结果。这个公式叫做独立增益方程，是因为改变其中一个增益（比如 K_p）后，不会因为其他项而改变输出。图 12-27d 给出了当 K_p 取 3 时比例积分系统的输出曲线（前提是位置误差仍为 0.25 英寸，保持不变）。

相关增益方程中的积分计算与此类似，只是有两点不同：(1) 比例因子（K_c）是与误差、积分还有微分的和相乘，因此误差项没有自己单独的增益，所以控制变量中总有一个与误差成比例的分量；(2) 当稳态误差存在时，积分项的值是每分钟增长一个误差项的大小。重置增益($1/T_i$)的单位是 1/分钟（定义如图 12-28 所示）。图 12-28a 画出了其值保持不变的稳态误差。由误差决定的输出也保持不变。图 12-28b 画出了由积分项单独作用的输出量。如果 $1/T_i=1$，那么它就会在一分钟时上升到大小和由误差所决定的输出相等；如果 $1/T_i=2$，那么它就会在一分钟后变成输出的两倍大小。图 12-28c 给出了有误差和积分共同作用的输出（当 $K_c=1$ 时）。图 12-28d 给出了 K_c 取 3 时，误差和积分项的和。

当然，由于输出量中积分分量的存在，最终将会克服摩擦力，稳态误差也会减小。如果台架向期望位置靠近，但滑过期望位置，到其另一边，这时，误差将变成负值，系统将会以一定的速度降低输出电压，这个速度是由误差大小和积分系数决定的。一段时间后控制变量输出就会大到能够迫使台架回到期望位置。由比例项决定的输出分量在误差为 0 时也会成为 0，但由积分项确定的控制变量输出分量却会有一定的大小。

PID 公式中的微分项用来防止超调。误差的微分，$d(E)/dt$，等于误差变化的速率。如果 PID 控制系统在接近期望位置的过程中，误差在减小，微分项就会为负。如果系统的误差在快速的缩小，那么微分项不仅为负而且还会很大。独立增益方程的微分计算按照每秒误差的变化，而相关增益方程的微分计算是按照每分钟误差的变化。误差的变化速率对 PLC 输出值的影响最后还将会由微分增益（K_d）和速率增益（T_d）来衡量。

还拿前面提到的位置控制系统作为例子，假如台架以 3in./s 的速度向期望位置移动，

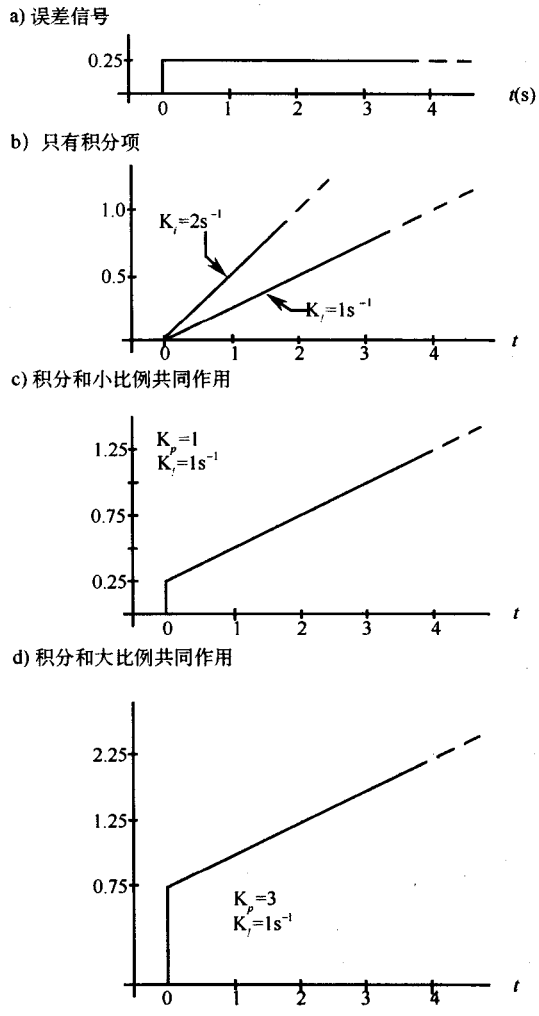


图 12-27 积分增益的作用（K_i 的单位为 1/s）：a）误差信号；b）只有积分项；c）积分和小比例共同作用；d）积分和大比例共同作用

那么误差的变化速率就是－3in./s（英寸/秒）。如果位置传感器发出的反馈信号为 10V/in.，微分项就等于(10V/in.×－3in./s)－30V/s。如果微分增益 K_d 被设为 0.01s，那么独立增益方程中的微分项就会将控制变量减小 0.3V，也就降低了电动机的速度。误差减小的越快，微分项就将被控系统的速度降得越低，降低了系统的动能即减小了超调发生的可能性。如果从微分项得到的（负）输出值大于比例项的（正）输出值（当位置误差很小时），电动机也将会减速，这个可以从下式看出。本例中，台架在距离期望位置不到 1/2in. 的地方，但仍是以 3in./s 的速度运动。这时的台架急需减速，否则一定会跃过期望位置。

$$CV=K_p\times\text{误差}+K_d\times\frac{\mathrm{d}(\text{误差})}{\mathrm{d}t}$$

其中，$K_p=1$，$K_d=1$s，误差＝1/2in.，d(误差)/dt＝－3in./s。此例中电压与距离的对应

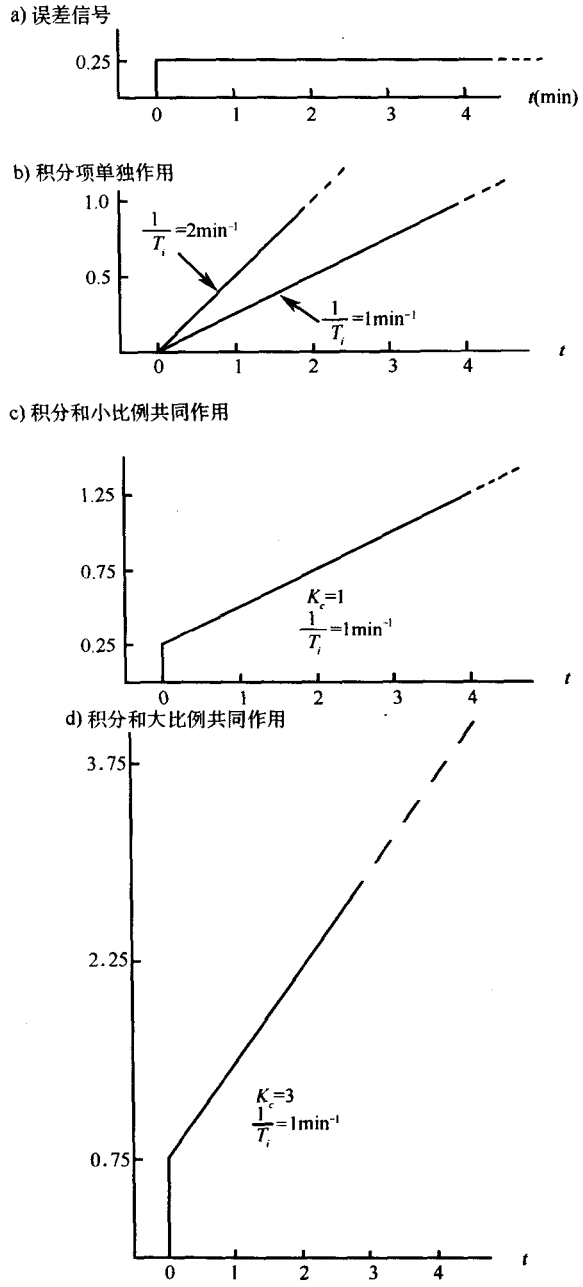


图 12-28 重置增益的作用（$1/T_i$ 的单位是每分钟重复次数）a）误差信号；b）积分项单独作用；c）积分和小比例共同作用；d）积分和大比例共同作用

关系数值是 10V/in.。那么

$$CV=K_p\times\text{误差}+K_d\times\frac{\mathrm{d}(\text{误差})}{\mathrm{d}t}=(1\times\frac{1}{2}\text{in.})+(1\text{s}\times-3\text{in./s})$$

$$=(0.5\text{in.})+(-3\text{in.})=-2.5\text{in.}$$

$$CV\text{（伏）}=-2.5\text{in.}\times10\text{V/in.}=-25.5\text{V}$$

当误差增加时，微分项同样可以起到防止超调发生的作用。当设定值改变时，误差就会变大，这时台架还未作相应的动作。在这种情况下，微分项为正，且将电动机的扭矩增大，让它去尽快提高台架的速度，进而使其位置发生变化。有了微分项就不需要很大的比例分量了。如果比例系数不用取的那么大，那么由比例项导致的超调发生的可能性就会小得多。

相关增益方程中的微分项的作用与独立增益方程中的一样，除非 PLC 是按分钟来计算误差变化，而且速率增益（T_d）的单位是分钟，这种情况下就有所不同了。相关增益方程中的速率增益往往定义为：在误差变化速率相同的前提下，要把输出值提高到某一值，含微分项时所节省的时间与只有误差项时所消耗的时间的比[⊖]。图 12-29 给出这个复杂的表述的一个形象的解释。图 12-29a 给出了一个匀速变化的设定值曲线，它会迫使台架往回运动。（简单起见）假设由于某些因素，台架现在被困在原处不动，因此位置误差就以恒定的速率在增加，那么只含误差项（其中取 $K_c=1$）的控制器的输出也就以同样的速率增长，如图 12-29b 所示。既然误差是以恒定的速率增加，那么误差的微分就是定值，如图 12-29c 所示。如果将微分引入，其中取 T_d 为 2，如图 12-29d 所示。当设定值开始变化时，控制变量的微分分量就会立刻达到某值，而在未引入微分的情况下需要 2 分钟才能到这个值，所以就花多长时间来达到某个输出值来看，使用了 T_d 取值为 2 的微分后，就节省了 2 分钟。依此类推，取 T_d 为 3，就会节省 3 分钟。在相关增益方程中，误差要在与 K_c 相乘前与微分项相加，如图 12-29e 所示。

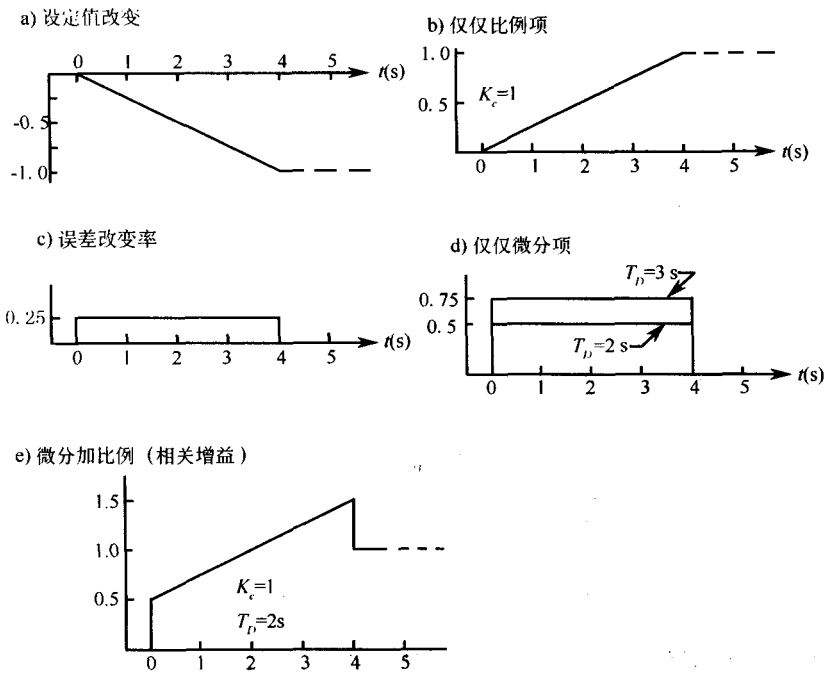


图 12-29 微分项，增加控制变量以响应误差改变 a）设定值改变；b）仅仅比例项；c）误差改变率；d）仅仅微分项；e）微分加比例（相关增益）

选择合适的比例、积分和微分增益不是一件容易的事。在介绍完如何编写使用 PID 控制指令后，我们将详细讨论如何整定 PID 控制。

⊖ 在比例常数 K_p 等于 1 时，同样的定义可以用在独立增益方程的 K_d 项。

1. Allen _ Bradley 的 PID 指令 PLC-5 SLC-500

Allen-Bradley 的 PID 指令有四种形式。一种在 SLC 5/02 中使用，还有一种与第一种很相似，在 SLC 5/03 和更高级的 PLC 中使用（低级 SLC 500 中无 PID 指令）。第三种形式，与 SLC 500 中的很相似，能够在 PLC-5 中使用。PLC-5 的提高版中使用的是改进的第四种形式的指令。因此，下面有很多内容是重复的。斜体字表示被描述的版本与其他 PID 版本之间存在明显的不同。

2. Allen _ Bradley PLC-5 的 PID 指令 PLC-5

使用如图 12-30 所示的 Allen-Bradley PLC-5 PID 指令，用户需输入：

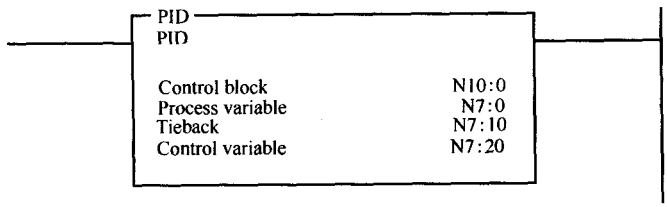


图 12-30 Allen-Bradley PLC-5 PID 指令

1）存储过程变量的地址。每次执行 PID 指令时，将会从这个地址中读取反馈值，然后根据这个值计算误差、误差之和，还有误差的变化率。PID 指令只读取这个地址中的低 12 位，而忽略高 4 位的内容，即从 0 到 4096 的整数，所以模拟输入模块的程序要能把过程变量值标度变换到这个范围内。为了显示目的，PID 指令能够把过程变量值转换成带工程单位的数据（见下），但这些工程单位不会出现在 PID 计算中。

2）存储控制变量的地址。当 PID 指令执行完，其结果 —— 一个 0 到 4095 间的 12 位无符号数，将被放到这个地址所指示的存储单元中。如果不想让输出的取值范围大于 12 位，那么可以对 PID 指令进行相应的配置。如果模拟输出模块需要其他形式的数据，那么程序或模拟输出模块就要对输出进行变换。假如控制变量存储单元中的值超过了 4095，而且程序员为 PID 指令选用的是整型控制块（见下），那么控制变量的值将会被永久改变（这样其他的指令就无法再向此地址写数据了）。

3）存储手动输入值的地址。如果被控系统选择为手动操作模式（见下），那么控制变量值就会从这个地址中得到，而不再是 PID 计算结果。图 12-31 画出了一个含有 PID 指令的 PLC 系统，它的控制变量值既可以是 PID 计算结果，也可以是手动输入值。手动输入值地址中的数据可以来自一个连接模拟输入模块的手动电位器，也可以由操作员通过计算机终端输入，甚至还可以由另一个 PLC 通过网络提供。手动输入值的值必须是 12 位的 0 到 4095 之间的数值。这样，PID 方程才能将其转换成百分比的值来使用。如果不打算使用手动控制，那么就在手动输入值地址处输入 0。

手动输入值也能够在 Allen-Bradley 的无扰特性方面得到利用。当被控过程处于手动控制模式下，每次执行 PID 指令时的积分项的值都是直接取自手动输入值地址，这样在系统转为自动控制时，系统的输出基本等于手动模式下的值，控制变量的值就不会突然有太大的变化。

4）控制块的地址。控制块是一组数据字，其中包含了 PID 配置参数和 PID 计算所需要的数值。其他 PLC 程序语句通过读或写适当的数据字可以改变控制块中的数值。控制块中包含的数据有：

■ 设定值

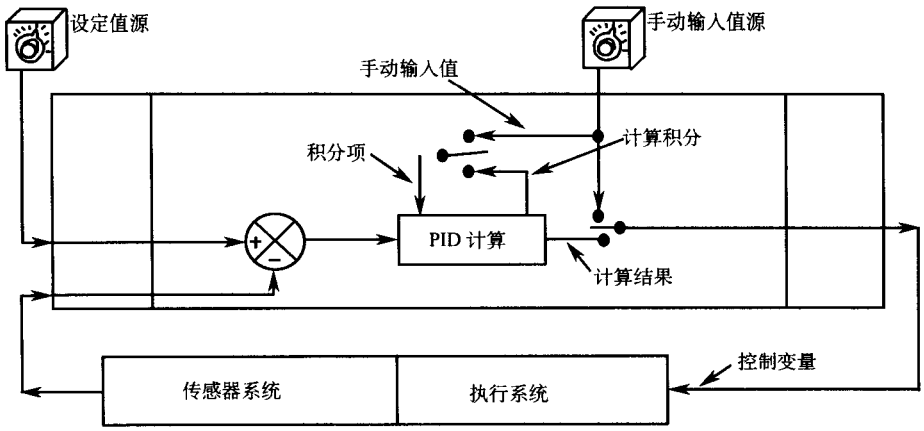


图 12-31 选择手动输入值或 PID 计算结果作为 PID 指令的控制变量输出

- 乘数（K_p、K_i 等）
- PID 方程计算过程中需要存储的临时数据（例如误差和）
- 控制器模式（手动/自动）的选择配置位和其他 PID 计算特性的选择配置位
- PID 计算的结果和状态位

有两种控制块可以使用：整型控制块和 PID 控制块。选择哪种控制块将影响到 PID 指令如何工作。经典的 PLC-5 只能用整型控制块，而对于改进型 PLC-5，整型控制块和 PID 控制块都可使用。

如果输入整型控制块地址（例如 N10：0）：

1）每次控制逻辑由假变为真时，PID 指令会执行一次。

2）计算的精度是 12 位整数。

3）PLC-5 将会创建一个含 23 个整型（16 位）字的整型控制块。将光标移到 PID 指令，然后进入数据画面，出现如图 12-32 所示的画面，这样程序员就可以改变整型控制块中的

```
          equation: 1 (0:AB/1:ISA)            feed forward:   7
              mode: 0 (0:auto/1:manual)   max scaled input: 300
             error: 1 (0:SP-PV/1:PV-SP)   min scaled input:   0
   output Limiting: 1 (0:NO/1:YES)              deadband:  15
  set output mode: 0 (0:NO/1:YES)        set output value%:   0
  setpoint scaling: 0 (0:YES/1:NO)        upper CV limit %: 100
  derivative input: 1 (0:PV/1:error)      lower CV limit %:   0
 last state resume: 0 (0:NO/1:YES)         scaled PV value:  33
   deadband status: 1                          scaled error:   8
upper CV Limit alarm: 0                      current CV %:   7
lower CV Limit alarm: 0
setpoint out of range: 0
           PID done: 0                                setpoint:  25
        PID enabled: 0     proportional gain (Kc)  [.01]:  10
                         res. time (Ti) [.01 mins/repeat]:   1
                          derivative rate (Td) [.01 mins]:   2
                           loop update time    [.01 secs]:  10


 Enter value or press <ESCAPE> to exit monitor.
 N10:0/0 =
 Program    Forces None    Data:Decimal        PLC-5/15 File DRILL1
```

图 12-32 选择整型控制块的 PLC-5 的 PID 指令的数据画面

值，让光标指向你想要作修改的输入口（真正的地址将会在屏幕底端显现），然后输入新值。这个数据画面中显示了许多（但非全部）程序员需要监控和更改的数值[⊖]。

如果数据画面被调用而光标却未在 PID 指令上出现，程序员能够看到并修改整型控制块而无需识别每个入口处旁边的标注。

4）在用户程序中，控制块的字和位都必须用数字标出地址。例如，如果程序需要修改过程的设定值，程序员需要知道设定值是存储在控制块的第三个数据字中（N10：2，假如控制块从 N10：0 开始）。图 12-33 告诉我们如何在执行 PID 指令之前写新的设定值到整型控制块中。PLC-5 的 PID 整型控制块的结构如图 12-34 所示。

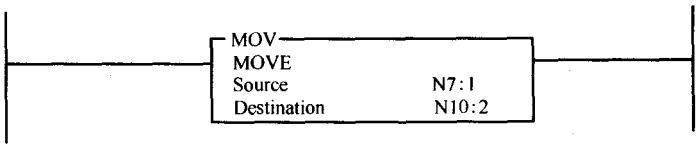


图 12-33 使用 MOVE 指令来写入 PLC-5 整型控制块的设定值

字	包 含	术 语	输入范围
0	位 15 使能（EN） 位 13 完成（DN） 位 11 设定值超出范围 位 10 输出报警，下限 位 9 输出报警，上限 位 8 DB，当误差在死区时置位 位 7 恢复上次状态 （0＝yes；1＝保持上次状态） 位 6 微分作用（0＝PV，1＝误差） 位 5 设定值解标度变换 位 4 设定输出（0＝no，1＝yes） 位 3 输出限幅（0＝no，1＝yes） 位 2 控制（0＝反向，1＝正向） 位 1 模式（0＝自动，1＝手动） 位 0 等于（0＝独立，1＝ISA）		
1	保留		
2	设定值	SP	0-4095（未标度变换） Smin-Smax（标度变换）
3	独立：比例增益×100（无单位）	Kp*	0-32 767
	ISA：控制器增益×100（无单位）	Kc*	0-32 767
4	独立：积分增益×1 000（1/s）	K_i**	0-32 767
	ISA：复位项×100（重复每分钟）	T_i*	0-32 767
5	独立：微分增益×100（秒）	K_d*	0-32 767
	ISA：比率项×100（分钟）	T_d*	0-32 767
6	前馈或偏离	FF/Bias	0-4095
7	最大标度	S_{max}	-32 768-＋32 767
8	最小标度	S_{min}	-32 768-＋32 767
9	死区	DB	0-4095（未标度变换） *Smin-Smax*（标度变换）
10	设定输出	SETOUT	0-100%
11	最大输出极限（输出%）	L_{max}	0-100%
12	最小输出极限（输出%）	L_{min}	0-100%

图 12-34 PLC-5 的 PID 整型控制块的结构表

⊖ 一些控制块的值不能在运行模式下修改，因为这样可能会使设备以一种不可预料的甚至危险的方式运行，见 Allen-Bradley 指令集手册。

字	包　　含	术　　语	输入范围
13	回路更新时间×100（秒）	dt	0-32，767
14	标度变换后的 PV 值（显示）		S_{min}-S_{max}
15	标度误差值（显示）		S_{min}-S_{max}
16	输出（4095 的%）	CV	0-100%
17-22	内部存储；不使用		
重要：带有 * 的术语被输入作为 Yy×100。术语自己是 Yy。带有 ** 术语是被输入作为 Yy×1 000。术语自己是 Yy。			

图 12-34　（续）

改进型 PLC-5 可以选择使用 PID 控制块。输入 PD 文件地址（例如 PD9：0）来选择 PID 控制块：

1）当控制逻辑为真时，PID 指令将不断地执行每个程序扫描。

2）PID 计算基于 32 位浮点运算，所以精确度会高些。（但控制变量输出仍为 12 位无符号二进制数。）

3）一个含有 80 个（16 位）数据字的 PD 数据文件将会被生成。将光标移到 PID 指令，然后进入到数据画面，程序员就可以看到 PID 控制块中的数据。图 12-35 的一个数据屏幕会被显示，

```
Setpoint:                0.000000 Proportional Gain (Kp):        0.000000
Process Var.:            0.000000 Integral Gain (Ki) [/secs]:    0.000000
Error:                   0.000000 Derivative Time (Kd) [secs]:   0.000000
Output %:                0.000000

Mode:                    AUTO     Deadband:                      0.000000
PV Alarm:                NONE     Output Bias %:                 0.000000
Deviation Alarm:         NONE
Output Limiting:         NONE     Tieback %:                     0.000000
SP Out Of Range:         NO       Set Output %                   0.000000
Error Within Deadband:   NO
PID Initialized:         NO

A/M Station Mode:        AUTO
Software A/M Mode:       AUTO
Status Enable (EN ):     0

PID Equation:            INDEPENDENT  Engineering Unit Maximum:     0.000000
Derivative Of:           PV           Engineering Unit Minimum:     0.000000
Control Action           SP-PV
PV Tracking:             NO           Input Range Maximum:          0.000000
                                      Input Range Minimum:          0.000000
Update Time (secs):      0.000000     Output Limit High %:          0.000000
                                      Output Limit Low %:           0.000000

Cascaded Loop:           NO           PV Alarm High:                0.000000
Cascaded Type:           -            PV Alarm Low:                 0.000000
Master To This Slave:    -            PV Alarm Deadband:            0.000000

                                      (+)Deviation Alarm:           0.000000
                                      (-)Deviation Alarm:           0.000000
                                      Deviation Alarm Deadband:     0.000000

Press a function key or enter a value.
PD0:1.PE=
Rem Prog  Forces:Disabled Data:Float          Addr:Decimal PLC-5/40 Addr  4
          Toggle    Specify   PID:1:    Next    Prev    Next    Prev
                    Address   Monitor   File    File    Element Element
          F3        F5        F6        F7      F8      F9      F10
```

图 12-35　PLC-5 PID 控制块的数据屏幕

程序员可以轻易的跳转到另一个画面（如果 PD 数据文件被访问，而无光标指向 PID 指令，那么两幅画面会是一样的。）。与整型控制块相比，更多的 PID 操作和数据可以被监控和修改。⊖

4）用户程序的 PID 控制块中的每个数据字都必须有一个地址助记符。要将设定值写入 PD 文件，我们需要知道 PD 文件号（例如本例中的 PD9.0）和设定值的地址助记符（.SP）。MOVE 指令如图 12-36 所示。图 12-37 所示为本章后面将会讨论的地址助记符。（关于 PD 数据文件中的数据的更详细的解释在附件 J 中。）

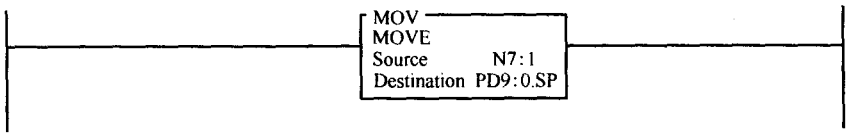


图 12-36 利用 MOVE 指令向 PLC-5 PID 控制块写设定值

字	包 含		范 围
0	控制/状态位		
	位 15 使能（EN）	.EN	
	位 9 串级选择（主、从）	.CT	
	位 8 串级回路（0=no，1=yes）	.CL	
	位 7 过程变量跟踪（0=no，1=yes）	.PVT	
	位 6 微分作用（0=pv，1=error）	.DO	
	位 4 设定输出（0=no，1=yes）	.SWM	
	位 2 控制作用（0=SP-PV，1=PV-SP）	.CA	
	位 1 模式（0=自动，1=手动）	.MO	
	位 0 方程（0=独立，1=ISA）	.PE	
1	状态位		
	位 12 PID 初始化（0=no，1=yes）	.INI	
	位 11 设定点超出范围	.SPOR	
	位 10 输出报警，下限	.OHL	
	位 9 输出报警，上限	.OLL	
	位 8 DB，当误差在 DB 时置位	.EWD	
	位 3 误差过低报警	.DVNA	
	位 2 误差过高报警	.DVPA	
	位 1 过程变量过低报警	.PVLA	
	位 0 过程变量过高报警	.PVHA	
2，3	设定值	.SP	$-3.4E^{+38}$到$+3.4E^{+38}$
4，5	独立： 比例增益（无单位）	.KP	0 到$+3.4E^{+38}$
	ISA： 控制器增益（无单位）		0 到$+3.4E^{+38}$
6，7	独立： 积分增益（1/s）	.KI	0 到$+3.4E^{+38}$
	ISA： 复位项（重复/每分）		0 到$+3.4E^{+38}$
8，9	独立： 微分项（秒）	.KD	0 到$+3.4E^{+38}$
	ISA： 比例项（分）		0 到$+3.4E^{+38}$
10，11	前馈或偏离	.BIAS	−100 到+100%
12，13	最大标度	.MAXS	$-3.4E^{+38}$到$+3.4E^{+38}$
14，15	最小标度	.MINS	$-3.4E^{+38}$到$+3.4E^{+38}$
16，17	死区	.DB	0 到$+3.4E^{+38}$
18，19	设定输出	.SO	0-100%
20，21	最大输出极限（输出%）	.MAXO	0-100%
22，23	最小输出极限（输出%）	.MINO	0-100%
24，25	回路更新时间（秒）	.UPO	
26，27	标度变换后 PV 值（显示）	.PV	
28，29	标度变换后误差值（显示）	.ERR	

图 12-37 带地址助记符的 PLC-5 PID 控制块的结构

⊖ 见前一个注中关于在运行模式修改的警告。

字	包含		范围
30，31	输出（4095 的%）	.OUT	0-100%
32，33	过程变量高报警值	.PVH	$-3.4E^{+38}$到$+3.4E^{+38}$
34，35	过程变量低报警值	.PVL	$-3.4E^{+38}$到$+3.4E^{+38}$
36，37	误差高报警值	.DVP	0到$+3.4E^{+38}$
38，39	误差低报警值	.DVN	$-3.4E^{+38}$到0
40，41	过程变量报警死区	.PVDB	0到$+3.4E^{+38}$
42，43	误差报警死区	.DVDB	0到$+3.4E^{+38}$
44，45	最大输入值	.MAXI	$-3.4E^{+38}$到$+3.4E^{+38}$
46，47	最小输入值	.MINI	$-3.4E^{+38}$到$+3.4E^{+38}$
48，49	手动控制的手动输入值（0-4095）	.TIE	0-100%
51	主 PID 文件号		0-999；对增强型 PLC-5 是 0-9999
52	主 PID 元素号		0-999；对增强型 PLC-5 是 0-9999
54-80	内部存储，未使用		

图 12-37 （续）

如图 12-38 所示为一个使用了整型控制块的 PID 指令的 PLC-5 程序。我们假设这个程

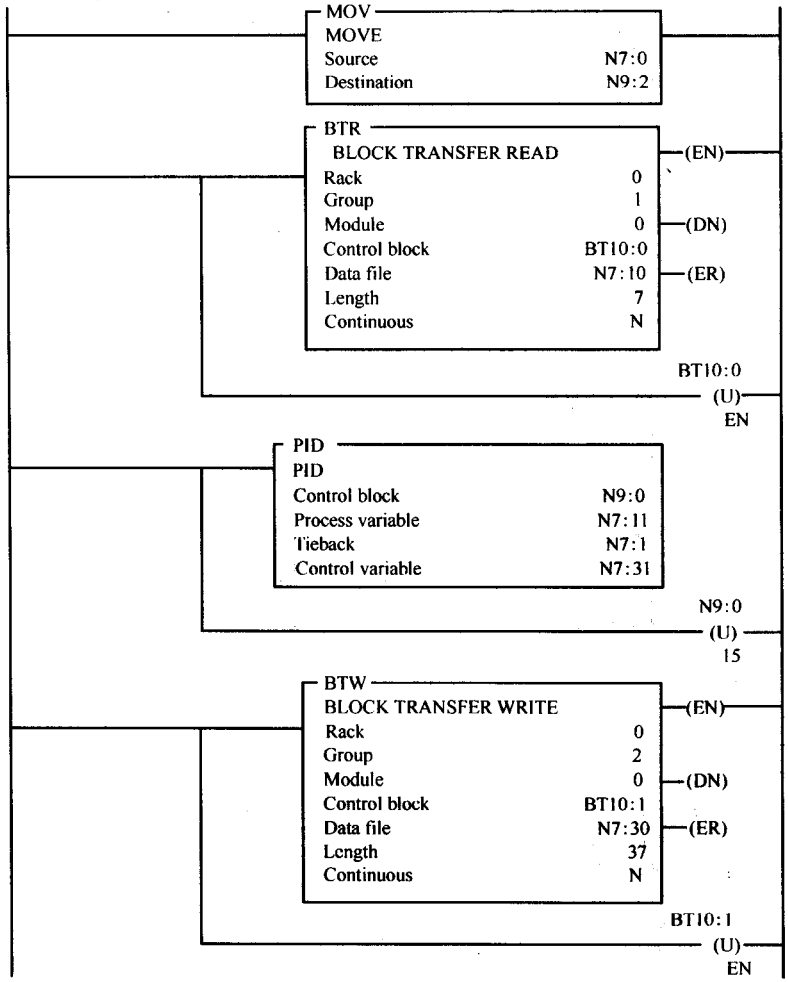


图 12-38 PLC-5 梯形图程序，使用整型控制块的 PID 指令

序是写在了 STI（定时中断）程序中。由于块传送指令在 STI 程序中是无条件执行的，所以在它们执行后，将它们的使能位解锁（unlatch），这样在下一次调用 STI 程序时又可以执行这些指令。既然使用了整型控制块的 PID 指令只有在控制逻辑为真时才执行，那么我们可以同样利用刚才提到的技巧来在下一次 STI 程序执行时，迫使 PID 指令再次被执行（控制块中的第一个数据字的位 15 是 PID 的使能位）。

图 12-39 给出了一个相似的程序，不同之处是 PID 指令使用了 PID 控制块。使用了 PID 控制块的 PID 指令，当控制逻辑为真时，每次都执行扫描循环，所以图 12-39 中的程序没必要将控制块的使能位解锁。

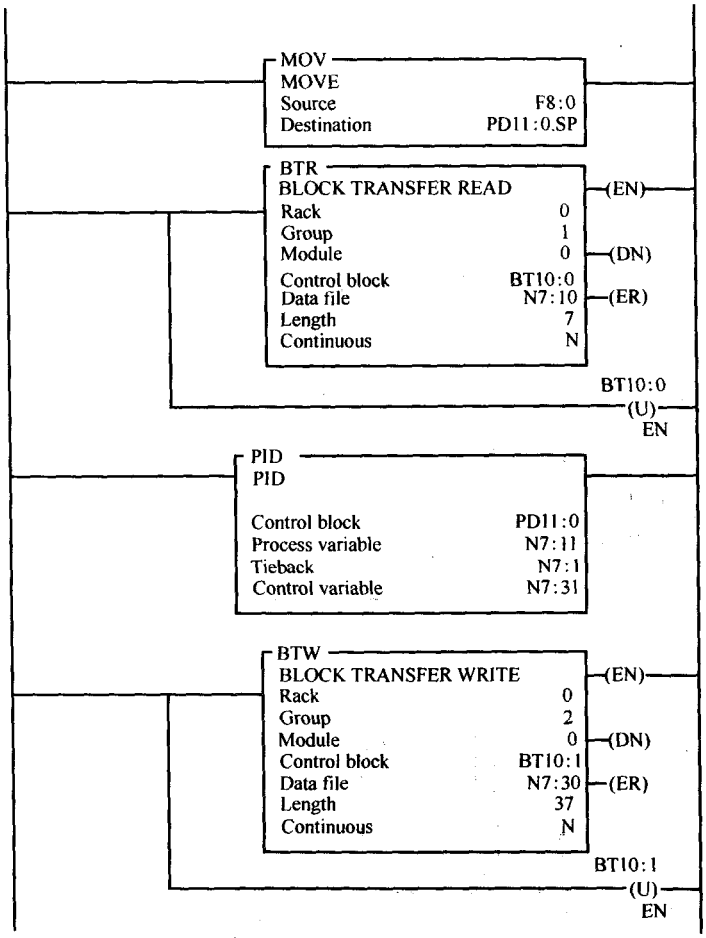


图 12-39 PLC-5 梯形图程序，使用 PID 控制块结构体的 PID 指令

图 12-40 和图 12-41 是从 Allen-Bradley 用户手册上摘下来的流程图。它们告诉我们使用整型控制块（图 12-40）和 PID 控制块（图 12-41）时，PID 指令是如何执行的。注意这两个图之间有很多不同，即使是在术语的使用上。现在请参照这两幅流程图来继续学习下面的有关这两种控制块的内容。

PLC-5 PID 指令的控制块数据

在这一节里，我们要介绍 PLC-5 PID 指令中用到的整型和 PID 型控制块。首先看一下

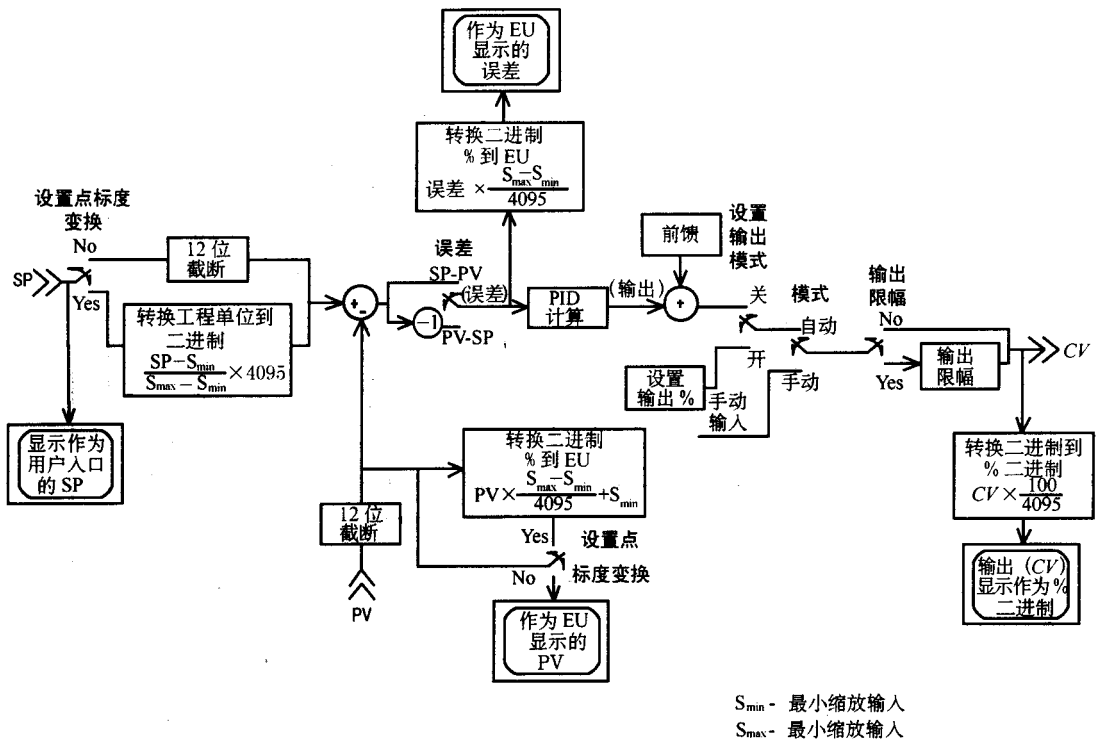
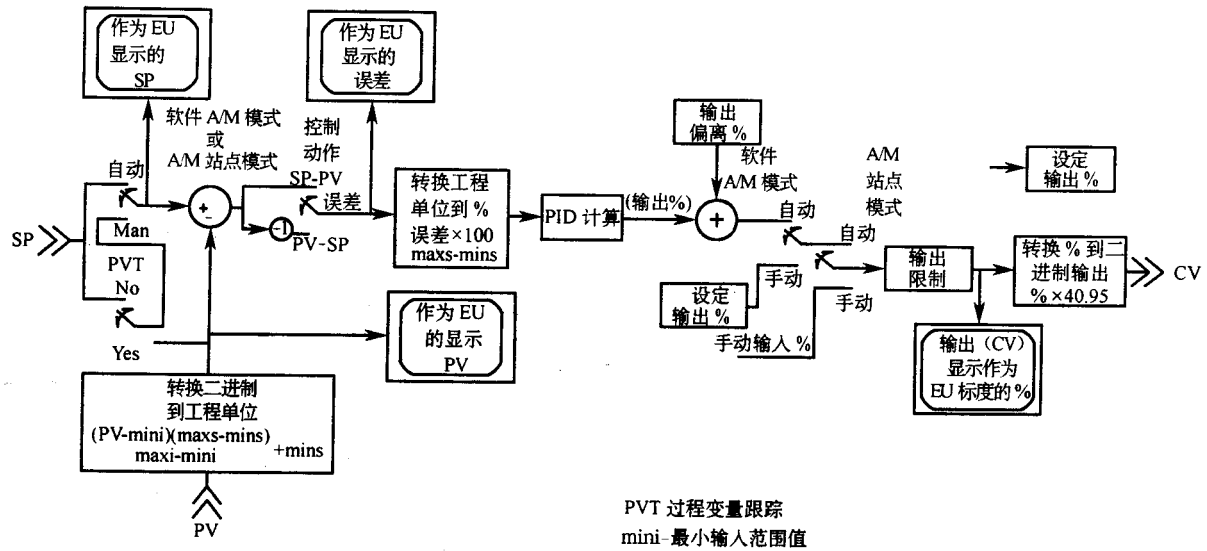


图 12-40　使用整型控制块的 PLC-5 PID 指令的操作



PVT 过程变量跟踪
mini-最小输入范围值
maxi-最大输入范围值
mins-最小工程单位值
maxs-最大工程单位值

图12-41　使用PID控制块的PLC-5 PID 指令的操作

这两种控制块中相似的数据，然后再讨论这两种控制块的不同之处。PID 控制块助记符和整型控制块偏离量都将在对用户程序想要监控和修改的数据的描述中提到。如果没有包括助记符或偏离量地址，那么数据就只能在 PLC 的程序模式下被修改（这里图表和后面的附录给出了所有的地址和助记符）。斜体字表示 SLC 500 PLC 中的 PID 指令与此处 PLC-5 所描述的差异。

1）方程（EQUATION）选择位，用来选择独立增益方程或相关增益方程。方程的输出中会含有偏移量，这个我们在前面的讨论中没有提到。

独立增益方程：

$$CV=K_p(E)+K_i\int(E)\mathrm{d}t+K_d\frac{\mathrm{d}(E)}{\mathrm{d}t}+\text{偏移量}$$

相关增益方程：

$$CV=K_c\left[E+\frac{1}{T_i}\int(E)\mathrm{d}t+T_d\frac{\mathrm{d}(E)}{\mathrm{d}t}\right]+\text{偏移量}$$

2）控制行为选择位，用于选择如何计算误差：

- SP-PV（正作用）这是计算误差的传统做法。如果设定值（SP）大于测量到的过程变量值（PV），那么误差为正，也就是使控制变量为正，使过程变量去跟随设定值。
- PV-SP（反作用）这种方法是将误差值符号取反，这样 PID 计算出的结果将会产生与前一种方法完全相反的动作。例如我们用 PID 指令来控制液体的温度，方法是控制冷水的流入量，如果当前温度（PV）非常低，那么我们就需要减少冷水的流入量，而不是增加。

3）微分输入值选择位

- 误差 这一章只讲了微分分量是按照误差变化率来计算的，并且要加到比例分量和积分分量。将微分作用位设置为 ER（=1），就可以实现上面提到的动作。
- PV 如果这个微分动作位被清 0，那么 PID 指令就会去计算过程变量的变化率而不是误差的变化率。如果设定值很少变化，那么依据过程变量来计算微分项能够对有外界干扰的影响起到更好的控制作用，因为 PID 算法会努力消除因外界影响而产生的过程变量的变化。如果这种微分作用和 SP-PV 控制功能被选中，那么微分项将会按照下式计算：

独立增益方程：

$$CV=K_p(E)+K_i\int(E)\mathrm{d}t-K_d\times\frac{\mathrm{d}(PV)}{\mathrm{d}t}+\text{偏移量}$$

相关增益方程：

$$CV=K_c\left[E+\frac{1}{T_i}*\int(E)\mathrm{d}t-T_d\frac{\mathrm{d}(PV)}{\mathrm{d}t}\right]+\text{偏移量}$$

在直接（SP-PV）控制方程中减去一个与过程变量变化率成比例的值是完全有必要的，这样可以避免超调，因为一个变化很快的过程变量是很容易导致超调的。

4）模式或 A/M 选择模式选择位（PD 助记符=.MO，整型文件偏离量=0/1。）在用户程序中可编写指令来监控人工控制模式的切换，并相应地改变此位：

- 在自动模式（=0），控制变量值由 PID 方程计算得来。
- 在手动模式（=1），控制变量值来自手动输入地址（PID 指令必须输入一个地址或数值）。手动输入地址可以由用户程序从一个与模拟输入模块相连的表盘上读取。在手动模式下，手动输入值也会被拷贝到积分项存储单元中，这样，在自动模式重新启动时，PID 方程的

计算结果就会从与手动模式下的最终输出值相近的数值开始，从而不会突然有太大的控制变量值抖动。

5）设置输出模式或者软件 A/M 模式选择位（PD 助记符=.SWM，整型偏离量=0/4）这个位可由用户程序来控制选择。它的功能与刚才提到的（A/M）模式选择位很相似。

- 在软件自动模式（=0），PID 指令的控制值是按正常方式计算的。
- 在软件手动模式（=1），控制变量值来自设置输出%地址（稍后会有进一步的讨论），这个地址单元中可以包含操作员通过数据监控器输入的数据，也可以是由另一个 PLC 程序提供的。如果设置输出模式位和 A/M 选择位都被置 1，那么控制变量将取决于手动输入值，而不是设置输出%值。

如果使用的是整型控制块，下面的操作必须要选中。但如果使用的是 PID 控制块，这些操作会被自动选中。

1）输出限定选择位（偏离量=0/3）

- 当置为 no（=0）时，输出控制变量被限定在 12 位无符号数值范围内，即 0 到 4095。
- 当置为 yes（=1）时，PID 指令的最大值和最小值是由输入的 CV 上限%参数和 CV 下限%参数来决定（后面会进一步讨论到）。

无论输出限定位是否被选中，当计算得到的控制变量值大于上限（PD 助记符=.OLH，偏离量=0/9）或低于下限（.OLL，0/10）时，PID 指令都会将控制块中的状态位置 1。当控制变量值超过范围时，PID 指令也会停止继续计算积分项，来防止“reset windup”，这个词的意思是积分项变得过大。

2）设定值修定选择位

- 如果是 off（=1），那么修定功能将被禁止。设定值就会被只保留后 12 位，并被当作无符号二进制数字。它的高 4 位会被省略掉而无其他任何修改。
- 如果是 on（=0，默认），设定值（和使用整型控制块时的死区值）就可以以带“工程单位”的形式提供给 PID 指令（也是一定范围的数值，而不是标准的 12 位 0 到 4095 之间的数字）。带有工程单位后，其上下限需要输入到控制块的最大、最小变换输入中或工程单位参数（后面将会讲到）中，这样，PID 指令就能把带有工程单位的设定值转换为 PID 指令能够使用的数据形式。

3）末态保持选择位（偏离量=0/7）

- 如果是 off（=0），那么 PID 计算的积分值会在 PLC 脱离运行模式后变为 0。
- 如果是 on（=1），那么 PID 计算的积分值在 PLC 脱离运行模式后不会变为 0。如果你的模拟输出模块配置为在 PLC 进入程序模式或出错时保持结束时刻的数值，那么在 PLC 再次进入运行模式后，积分值会保持上次的积分数值不变。[⊖]

在整型控制块中还有：

4）PID 使能状态位（偏离量=0/0），PID 完成状态位（偏离量=0/1），这是非常有用的，因为使用了整型控制块的 PID 指令只有在控制逻辑从假变真时才执行一次。图 12-38 中

⊖ PID 指令第一次执行时可能会给积分值分配一个已经使用过的内存，这个存储区域可能包含一个非零值。在末态保持选择位打开前至少执行 PID 指令一次可以清除这个积分值。否则输出值可能会造成不可预知的，甚至是危险的结果。

的程序显示了如何在 STI 程序中使用 PID 使能位。

下面的选项只有在 PD 控制块中才可用：

1）过程变量追踪选择位（助记符=.PVT） 如果置为 1，即选中，PID 指令会在手动模式下将过程变量拷贝给设定值。在手动模式下，PID 计算仍会继续进行，虽然它的结果不会被使用，这样做的目的是防止积分项增长（如果过程变量=设定值，那么误差=0）。当切换到自动运行模式下时，控制变量输出的变化就是最小的。

2）级联回路选择位和级联类型选择位 在级联控制回路中，主控制回路的控制变量输出成为从控制回路的设定值。如果这个 PID 回路是从回路，那么必须输入主 PID 文件号和主 PID 元素号（助记符=.ADDR）。（本书不讨论级联控制）。

在使用 PID 指令之前必须要将一些数据字输入到 PID 指令的控制块中，这些数据字有时可以在控制程序正在执行时被修改，它们是：

1）设定值（PD 助记符=.SP，整型偏离量为字 2） 这个值通常是由你的 PLC 程序中的另一条梯级写入控制块的，如图 12-38 和 12-39 中程序的第一个梯级所示。设定值的大小必须在最大值和最小值之间。如果设定值的标度变换在整控制块配置时被关闭了，那么设定值的大小就要在 0 到 4095 之间。如果输入的设定值超过这个范围，那么 PID 指令就会将一个状态位置位（PD 助记符=.SPOR，整型偏离量=0/11）。

2）比例、积分和微分系数（K_p、K_i 和 K_d）或控制器增益、重置时间和微分时间（K_c、T_i 和 T_d）（PD 助记符=.KP、.KI 和 .KD；整型偏离字分别为 3、4 和 5） 这些数只能是正数，负数会被 PID 计算当作 0 来处理，图 12-34 和 12-37 给出了取值范围和格式要求。注意，整型控制块的输入是以输入乘数值的百分之一或千分之一为单位，选择一个合适的取值方法将会在后面进行讨论。

3）回路更新时间或更新时间 以秒计（在整型控制块中为百分之一秒，在 PID 控制块中为浮点值）的回路更新时间不能为 0，否则 PLC 将会出错，因为回路更新时间是用来计算积分项和微分项的，它的值就是用作 dt 的值[⊖]：

$$\text{新积分项}=\text{旧积分项}+K_i(e_c\,\mathrm{d}t)$$

$$\text{微分项}=\frac{\text{上次计算以来 } e_c \text{ 的变化}}{\mathrm{d}t}$$

更新时间需要选择一个足够快的可以完全控制被控过程但又不能太频繁而影响 PLC 的扫描循环的值。它不能大于被控过程固有频率的五分之一（稍后会进一步讨论到）；也不能太小，应该至少为平均扫描循环的五倍。

程序员一定要保证 PID 指令的执行频率与此处输入的回路更新时间相符。（也许当你学习本书时，PLC 已经能够用自带的时钟计算 PID 指令执行的间隔有多大了，但起码编写此书时，这还不行。）输入一个更新时间而不是实际的计算时间间隔可能会导致不期望的微分项和积分项，输入的值越大，积分项增长的就越快，而微分项起到的作用就会越小。

4）死区值（PD 助记符=.DB，整型偏离量=9）的单位与设定值的相同 如果计算的误差在最后一次大过死区值后过了 0 点，而现在又小于死区值，那么 PID 指令将会把误差

⊖ 这里的公式假设使用独立增益方程，并且微分项基于计算得到的误差。稍微不同的是，方程显示的微分项是简化的，Allen-Bradley 的 PID 计算将 16 个时间间隔的微分项进行平滑处理。

当作 0 来重新计算控制变量。因此控制变量就会等于前面计算得到的积分分量，但积分项不会再增加。这时的控制变量就有可能大到允许执行器系统（例如升降装置）来克服会导致误差的作用力（比如重力），但它却不会去改正这个小而重要的误差。如果系统有物理的死区特性（比如静摩擦），那么我们这里就要输入一个这样的死区值，它应该在积分项大到能够克服系统的物理死区时防止积分器产生陡然的动作。当误差过 0 点但又未超过死区值时，状态位（PD 助记符=.EWD，整型偏离量为 0/8）将会置位。

5）上和下 CV 限定值%，也叫做高和低输出限定值%（PD 助记符=.MAXO 和 .MINO，整型偏离字分别为 11 和 12）输入的值应当是一个 0 到 100 之间的百分数（PD 数据文件中用的是浮点数）。 这两个参数指出了 PID 指令的控制变量输出值的范围在 0 到 4096 之间。如果 PID 指令产生超过范围的值，而且此时输出限定功能也打开了，那么正如前面所描述的，PLC-5 会把输出值降到限定值。手动输入值或软件提供的输出值也会受到限制，被限幅后，控制变量被存储到控制块中（PD 助记符=.OUT，整型偏离字=16）。这个%值在当作控制变量输出前会被转换成 12 位的 0 到 4095 之间的数值。

6）前馈值或输出偏离量（PD 助记符=.BIAS，整型字=6）加到 PID 计算的输出值中 它的取值范围也是 0 到 4095 之间。－100%到＋100%的浮点数可以输入到 PD 控制块中，它与工程单位的取值范围的最小值（－100%）和最大值（＋100%）是一致的。标度变换将在后面讨论。

7）设置输出值（PD 助记符=.SO，整型偏离字=10） 以 0 到 100 之间的百分数的形式输入（PD 数据文件中为浮点数）。如果设置输出模式或软件 A/M 模式选择位置为 1，那么设置输出值指示的是输出值在限幅前占最大输出值的百分比。（后面会进一步讨论）。

8）要求对设定值、过程变量、死区值和误差进行标度变换的数值 使用 PID 控制块的 PID 指令需要不同的标度变换值，而使用整型控制块的 PID 指令却不需要，因为 PID 控制块用的是浮点运算，而整型控制块用的是无符号整型运算。

- 如果 PID 指令用的是整型控制块，那么必须输入下列标度变换参数：
  - 最大值和最小值标度变换输入值 设定值和死区值可以以带工程单位的形式输入，而无需再用标准的 0 到 4095 的形式（这时还需要将设定值解变换选择位置 0 以使能解变换）。PID 指令会在计算前自动将工程量形式的设定值和死区值转换为 0 到 4095 之间的数值。例如，我们想控制电动机的速度在－200 到＋350rpm 之间，那么“－200”和“350”就是最小和最大的比例值，PID 指令能够把这个大小为 550 的量度与 0 到 4095 的标准输入对应起来。75 是－200 与 350 的中间值，其对应的标准值就是 0 和 4095 的中间值 2047。PID 指令用相应的逆方法将过程变量和误差值从 12 位数值转换为工程数据，用以显示。偏离字 14 中存有标度变换的过程变量值，偏离字 15 中是标度变换的误差值，其他程序语句可以将这些数据拷贝到操作显示屏上。
- 如果 PID 指令中有 PID 控制块的话，那么要想直接输入带工程单位的设定值和死区值，就必须输入两套标度变换参数。PID 指令在将过程变量与设定值比较得到误差之前，将过程变量转换成与设定值一样的工程数据，而得到误差后又会将误差值转换成百分数用以计算。需要的标度变换值有：
  - 最大和最小输入范围值（助记符=.MAXI 和 .MINI） 这些浮点数需要反映出能够提供给 PID 指令的可能的最大过程变量值的范围，这个范围由模拟输入模块决定，所以一般是 12 位无符号整数 0 到 4095。如果程序员不想让过程变量有如此大

的取值范围，那么就要在这个参数中输入数值。

- 最大和最小工程单位标度变换值（助记符=. MAXS 和 . MINS） 输入这些工程单位数（用真实数据）必须与所输入的输入范围最大值和最小值相一致。如果 . MAXS 和 . MINS 中输入 0，那么设定值和死区值就被认为是 0 到 4095 之间的数。如果后来提供的设定值超过了开始输入的范围，PID 指令就会置设定值溢出（SETPOINT OUT OF RANGE）状态位（助记符=. SPOR）。

假设 . MINS 和 . MAXS 中输入了－300 和＋300，那么在这个范围的 rpm（分转）速度设定值就可以提供给 PID 指令来控制电动机的速度，再假设过程变量测量系统（传感器、放大器和模拟输入模块）能够测量－600 到＋600rpm 之间的速度（如果模拟输入模块的整个 12 位范围（0～4095）都被使用）。. MINI 中应该输入 1024，因为它是代表最小速度－300rpm（. MINS）的过程变量，而 . MAXI 中应当输入 3092，因为它代表最大速度＋300rpm（. MAXS）。PID 指令将会以工程数据的形式计算过程变量（助记符=. PV）和误差值（. ERR），有相应的梯形图程序可以把这些数据拷贝到显示设备中。

下列的配置字只可以输入到 PID 控制块结构体中：

1）过程变量上限报警值和过程变量下限报警值（助记符=. PVH 和 . PVL） 如果发现过程变量超出界限（工程单位），PID 指令就会置位 PV 上限警报位或 PV 下限警报位（助记符=. PVLA 和 . PVHA）。为防止过程变量在 . PVL 或 . PVH 值上下振荡而导致报警位快速的开和关，可以再输入一个过程变量报警死区值（工程单位）。这个死区值能够有效的引入迟滞现象。一旦过程变量超过 . PVL 或 . PVH 界限，报警状态位就会置 1，但不会在过程变量刚回到 . PVL 或 . PVH 范围内就清 0，而是等其超过死区值后才清 0。

2）误差上限报警值和误差下限报警值，在配置画面中叫做（＋）背离报警和（－）背离报警（Deviation alarm）（助记符分别=. DVP 和 . DVN） 如果误差值（工程单位）超过了上限或下限，那么上限报警位或下限报警位便会置 1（助记符分别=. DVPA 和 . DVNA）。与上面类似，这里也可以输入一个误差报警死区值（有的资料上也叫做背离报警死区）一旦报警位置位，直到误差返回报警限值，报警位不会复位。

PID 控制块结构体中也有手动输入％数据值（助记符=. TIE），0 到 4095 的手动输入值转换到的％输出单元，这个值是用来显示的。

SLC 500

3. Allen-Bradley SLC 500 PID指令

SLC 500 的 PID 指令，如图 12-42 所示，以 16 位有符号整型数进行计算。在任何一种模式下，当控制逻辑为真时，PID 指令不是每个扫描循环都执行的，它按照一定的时间间隔进行，我们在后面将会看到。SLC 500 PID 指令中中通常使用相关增益方程，如下：

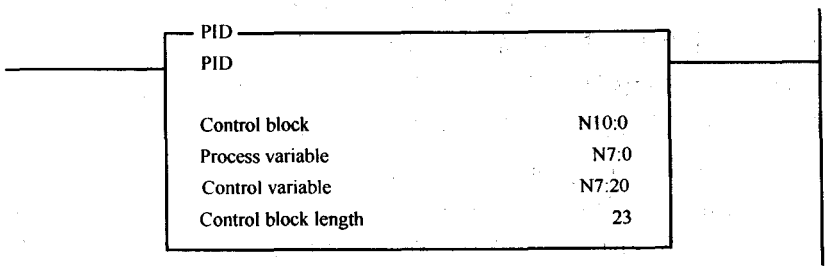


图 12-42 Allen-Bradley SLC 500 PID 指令

$$CV=K_c\left[E+\frac{1}{T_i}\int(E)\mathrm{d}t+T_d*\frac{\mathrm{d}(E)}{\mathrm{d}t}\right]+\text{偏移量}$$

要使用 Allen-Bradley SLC 500 PID 指令，用户必须输入：

1）过程变量的地址　每次 PID 指令执行时，它都会从这个地址读取反馈值，然后计算误差、误差和以及基于这个反馈值的误差变化率。PID 指令要求过程变量值必须在 0 到 16383 之间，所以用户程序需要对过程变量进行标度变换。在对 PID 指令进行配置时（后面会详细讨论），程序员可以把标度变换功能打开，这样数据就能以工程单位的形式显示，但它们不会用于 PID 计算。

2）控制变量的地址　当计算完成后，PID 指令就会把 0 到 16383 之间的计算结果放入这个地址单元。如果我们不希望控制变量的取值范围有 0 到 16383 这么大，我们可以对其进行限幅，后面我们会具体讲到。如果输出模块需要另一个取值范围的数，用户程序必须进行转换。

3）控制块的地址　这个控制块是一组 23 个数据字，它们是 PID 计算需要从一次执行动作传递给下一次执行动作的一些信息。包括：

- 设定值
- 乘数（K_c，T_i等）
- PID 计算需要存储的中间计算结果（例如误差和）
- 控制器模式选择的配置位（手动或自动）和 PID 计算的一些其他属性配置位
- PID 计算结果和状态位

控制块应存储在整型存储区。图 12-42 中指令的第四行不能改变，因为它可以提醒程序员要有 23 个连续的数据字被 PID 指令占用，程序员向数据画面中输入数据也可以通过用户程序来实现[⊖]。图 12-43 给出了数据画面，其中显示了大部分（但非全部）程序员想监控和更改的数据。有些入口在其后面的括号里给出了相应的助记符（例如设定值（SP））或是在显示值的后面给出（例如，time mode bit：1 TM）。用户程序只能用具体的地址来访问这些单元，而不能通过助记符。例如，程序需要改变被控过程的设定值，程序员就要知道设定值是写在了控制块的第三个数据字中（如上例中的 N10：2，因为那里的控制块是从 N10：0 开始的），然后编写一个 MOVE 指令，如图 12-44 所示，把新的设定值写入 N0：2。SLC50 PID 指令控制块的结构如图 12-45 所示。

```
      auto/manual: MANUAL*                          time mode Bit:  1 TM
             mode: TIMED *                         auto/manual bit:  1 AM
          control: E=SP-PV*                       control mode bit:  0 CM
    setpoint (SP):       0          0output limiting enabled bit:  0 OL
     process (PV):       0*                reset and gain range:  0 RG
     scaled error:       0*                 scale setpoint flag:  0 SC
         deadband:       0             loop update time too fast:  0 TF
      output (CV):       0%*            derivitive (rate) action:  0 DA
                                     DB, set when error is in DB:  0 DB
      loop update:       0 [.01 secs]   output alarm,upper limit:  0 UL
             gain:       0 [/10]        output alarm,lower limit:  0 LL
            reset:       0 [/10 m/r]       setpoint out of range:  0 SP
             rate:       0 [/100 min]   process var out of range:  0 PV
       min scaled:       0                                PID done:  0 DN
       max scaled:       0
output (CV) limit:       NO *                          PID enabled:  0 EN
  output (CV) min:       0 %
  output (CV) max:       0 %
```

图 12-43　SLC500 PID 指令的数据画面

⊖ 一些数据输入条不能在运行模式改变，因为这样做可能会使设备从一种不可预料的甚至是危险的方式运行，见 Allen-Bradley 指令集手册。

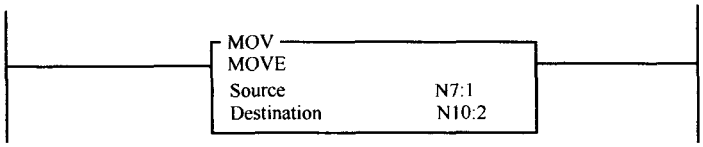


图 12-44 用 MOVE 指令把设定值写入 SLC500 整型控制块中

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	Word
EN DN PV SP LL UL DB DA TF SC RG OL CM AM TM	0
* PID Sub Error Code (MSbyte)	1
* Setpoint SP	2
* Gain K_c	3
* Reset T_i	4
* Rate T_d	5
* Feed Forward Bias	6
* Setpoint Max (Smax)	7
* Setpoint Min (Smin)	8
* Deadband	9
INTERNAL USE DO NOT CHANGE	10
* Output Max	11
* Output Min	12
* Loop Update	13
Scaled Process Variable	14
Scaled Error SE	15
Output CV% (0-100%)	16
MSW Integral Sum 5/03 MSW Integral Sum	17
LSW Integral Sum 5/03 LSW Integral Sum	18
INTERNAL USE DO NOT CHANGE	19
	20
	21
	22

OL, CM, AM, TM ①

① 你可以用你的梯形图改变这些状态。

② 应用到 SLC 5/03 和 SLC 5/04 处理器。

图 12-45 SLC 500 PID 指令的整型控制块结构

图 12-46 为一个使用 PID 指令的 SLC 500 程序。注意，PID 指令的执行是无条件的。它可以按照程序员在配置 PID 指令时给出的时间间隔来定时执行。图中还画出了立即输入、输出指令，假设模拟输入和模拟输出模块使用的数据是 0 到 16383 之间的，与 PID 指令相同。事实上，过程变量和控制变量往往需要标度变换。SLC 500 提供了两个指令，SCL 和 SCP，它们都可以改变数值的大小和偏移量。

SLC 500 PID 控制块数据

在这一节里我们解释控制块的数据输入。其中会提到数据的偏离量，因为这些可能是用

户程序想要监控和修改的。如果有的数据没有偏离量，那么它们就只能在 PLC 的程序状态下被修改，不过读者可以利用图 12-45 查出另一个地址。我们还将继续用斜体字来表示 PLC-5 中的 PID 指令的特别之处。

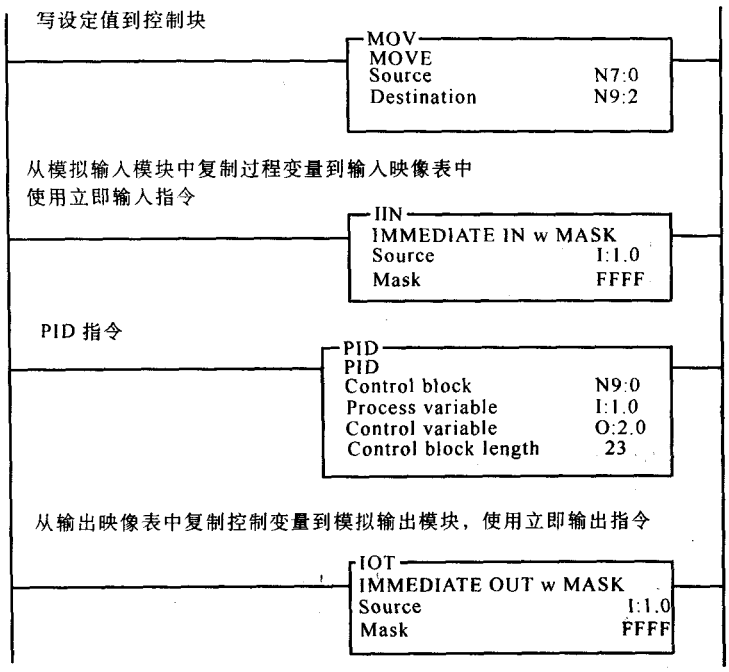


图 12-46 带有 PID 指令的 SLC 500 程序

控制位包括：

1）时间模式选择位 选择 PID 指令如何按照一定的时间间隔重复执行。

- STI 模式 如果＝0，PID 指令将会在每次 STI 程序被调用时执行。那么，程序员必须保证含有 PID 指令的 STI 程序执行的速率与 PID 指令控制块中的回路更新时间相符，否则积分项和微分项的计算结果就会错误。（回路更新时间稍后会谈到）
- 定时模式 如果＝1，PID 指令就会当作主扫描的一部分来执行（通常像图 12-46 中一样无条件执行），但是它还要用到一个定时器，为的是按照控制块中的回路更新时间来定时计算新的过程变量。

2）自动/手动模式选择位（控制块中的偏离量＝0/1） 主要影响控制变量的数据来源。用户程序可以控制此位。

- 自动模式下（＝0），控制变量值由 PID 指令计算得到。
- 手动模式下（＝1），新的控制变量不是来自 PID 计算得到。程序员要在用户程序中编写一些只在手动模式下会被执行的语句，将控制变量值写入控制变量的地址。

控制变量地址中的数值也用于 Allen-Bradley 无扰切换特性。当被控过程处于手动模式下时，每次 PID 指令执行时都会把控制变量地址中的值拷贝到 PID 指令的积分项中，这样当切换到自动模式后，由于积分项而给出的输出是与先前手动模式下相差无几的，并且控制变量值中就不会有太大的扰动。

3）控制模式选择位　用于选择如何计算误差。

- E=SP-PV（=0），这是传统的计算误差的方法。如果设定值（SP）大于测量到的过程变量（PV），误差为正，相应的控制变量为正，最终导致过程变量靠近设定值。
- E=PV-SP（=1），等于改变了上一方法的误差的符号，因此起到的作用正好与上一个相反。例如，如果要利用控制冷水流入量来控制液体的温度，那么当温度（PV）过低时，我们就需要减少，而不是增加冷水的流入量。

4）输出限幅使能选择位（偏离量=0/3）

- 当=0 时，输出控制变量被限制在标准范围 0 到 16383。
- 当=1 时，PID 指令的最大值和最小值就会由最小输出 CV%和最大输出 CV%参数来限制（下面会具体讲到）。

无论输出限幅功能是否被选择，当计算的控制变量高于上限（状态位偏离量=0/9）或低于下限（偏离量=0/10）时，PID 指令都会将控制块中的状态位置位。当计算的控制变量超过限定的范围时，PID 指令会停止积分项的增加，以防止“reset windup”，这个词的意思是积分项允许增长过大。

5）重置和增益范围选择位（SLC5/02 中不可用）

- 当=0 时，作为 K_c 和 $1/T_i$ 的数据在 PID 计算使用前会被缩小 10 倍。（增益和重置因子在后面会讲到）。
- 当=1 时，作为 K_c 和 $1/T_i$ 的数据在 PID 计算使用前会被缩小 100 倍，这样作可以提高控制精度。既然大的 K_c 和 $1/T_i$ 可以用在 SLC5/02 以上版本的模块中，那么就不需要用乘数大小来换取精度了。

6）微分（速率）作用选择位　使用该位可以使 PID 指令去计算下列量的变化率：

- 误差　当为 1 时，PID 指令计算误差的变化率，并把微分项与比例项和积分项相加。
- 过程变量　当为 0 时，PID 指令计算过程变量的变化率而不是误差的变化率。如果设定值基本不变，那么计算过程变量的变化率对于有外界扰动影响的过程来说会有更好的控制作用，因为 PID 计算会根据外界因素对过程变量的影响而采取相应的动作。如果这个功能和 SP-PV 控制功能被选中，那么微分项会按下式计算：

相关增益方程：

$$CV = K_c[E + \frac{1}{T_i}\int(E)dt + T_d\frac{d(PV)}{dt}] + \text{偏移量}$$

有必要在直接（SP-PV）控制方法的方程中减去一个与过程变量变化率成比例的量，因为变化太快的过程变量容易导致超调。

控制块中还有一些其他我们可以进行监控的状态位，其中大部分都受用户输入的参数的影响，所以我们把它们放到一起解释。不受其他参数影响的状态位有：

1）过程变量溢出位（偏离量为 0/12）：过程变量超出 0 到 16383 的范围时，置位。

2）PID 使能位（偏离量为 0/15）：反映控制 PID 指令的控制逻辑状态。

3）PID 完成位（偏离量为 0/13）：每次 PID 指令执行完后置 1，并在下一个循环关闭。

PID 指令在使用前必须将数据字输入到控制块中，但当程序运行时，有的数据字是会被修改的，这样的数据字包括：

1）设定值（偏离量为 2）　该值通常被用户程序的另一个指令写入控制块，如图 12-46

所示的第一行语句。设定值的大小必须在标度变换的最大和最小值之间（后面会解释原因）。如果控制块配置期间未打开设定值的标度变换功能，设定值就必须在 0 到 16383 之间。如果溢出，PID 指令会将设定值溢出位置位（偏离量＝0/11）。

2）增益、重置增益和速率增益（K_c、T_i 和 T_d）的值（偏离量分别为 3、4 和 5） 如果使用的是 SLC5/02，这些值必须在 0 到 255 之间；如果是 SLC5/03 或 5/04，那么取值范围变成 0 到 32767。增益和重置值在被 PID 计算使用前要缩小 10 倍，而速率增益要缩小 100 倍，但当重置和增益范围位被置 1 时，这三个值都要缩小 100 倍。如何选择这些值将在后面讨论。

3）前馈值或偏离值（偏离量为 6） 尽管这个参数不出现在 PID 指令数据画面当中，但可以通过其他数据画面或程序中的指令来改写它的值。这个地址中的值将会加到 PID 计算得到的输出值中，其取值范围是－16383 到＋16383。

4）死区值（偏离量为 9） 这个值的单位与设定值的单位相同，但只能取正值。当计算的误差值过 0 但尚未大于死区值时，PID 指令会把误差当作 0 来计算新的控制变量输出。这样，控制变量就会等于先前计算出的积分项的值，而积分项不再增加。这时的控制变量有可能足够大到可以允许执行器系统克服引起误差的作用力（例如重力），但却无法消除这个已存在的微小误差。对于有死区特性（例如静摩擦）的被控过程，我们可以令积分项不断增加，这样，控制变量最终会大到克服死区并产生一个陡然的动作。当误差过 0 但又未大于死区范围时，一个状态位，DB，误差在 DB 时置位（偏离量为 0/8），将会被置位。

5）回路更新时间 单位一般为百分之一秒。在 SLC5/02 中的取值范围是 1 到 255（0.01 到 2.55），在其他 SLC 500 系列中为 1 到 1024（0.01 到 10.24）。如果这个值选取的太小，PLC 将无法按照这个速率执行 PID 指令，这时，回路更新时间太快状态位（偏离量为 0/6）置 1。回路更新时间用来计算积分项和微分项，所以当太快（TOO FAST）状态位为 1 时，PID 计算的结果错误。计算积分项和微分项的 dt 的值就是回路更新时间的值[⊖]：

$$\text{新积分项}=\text{旧积分项}+K_i(e_c\,dt)$$

$$\text{微分项}=\frac{\text{上次计算以来 } e_c \text{ 的变化}}{dt}$$

更新时间需要足够快到可以完全控制被控过程，但又不能太频繁而影响 PLC 的扫描循环。它不能大于被控过程的固有频率的五分之一（稍后会进一步讨论）；也不能太小，应该至少为平均扫描循环的五倍。

6）标度变换最小值和标度变换最大值 有了这两个参数，设定值和死区值就可以以工程单位的形式输入。在 SLC5/02 中，这两个参数的取值范围是－16383 到＋16383，SLC5/03 和 SLC5/04 中为－32768 到＋32767。PID 指令会利用这两个参数把工程单位值转换成 PID 计算中使用的标准的取值范围 0 到 16383 之间的数值。如果输入了标度变换值，PID 指令还会将过程变量和计算出的误差值转换成工程数据的形式。这些经过标度变换的过程变量和误差值不会在 PID 计算中使用，但会被显示到数据画面上，并且存入控制块中的偏离字 14（过程变量）和偏离字 15（误差）中，可以使用其他程序语句把这些值拷贝到显示设备中。

例如，我们想控制电动机的速度在－200 到＋350rpm 范围内，那么“－200”和“350”就

⊖ 这里的公式假设使用独立增益方程，并且微分项基于计算得到的误差。稍微不同的是，方程显示的微分项是简化的，Allen-Bradley 的 PID 计算将 16 个时间间隔的微分项进行平滑处理。

是最小和最大标度变换值，PID 指令会把这个 550 的值域与标准的 0 到 16383 范围对应起来。设定值 75 是－200 和＋350 的中间值，会被调整为 0 和 16383 的中间值 8191。PID 指令用相应的逆运算把过程变量和误差从 0 到 16383 之间的数转换成带工程单位的数值显示出来。

如果标度变换值未输入，SLC500 便会把设定值标志状态（偏离量为 0/5）位置位。

7）输出 CV 最大值和输出 CV 最小值（偏离量分别为 11 和 12） 以 0 到 100 之间的百分数的形式输入。它们指出了 PID 指令允许的 0 到 16383 之间的输出。如果 PID 指令产生的输出结果超出范围，而且输出限制使能位已打开，那么 SLC500 就会把输出值降到这个范围的边沿值。被限幅修改后，控制变量百分值会被存储到控制块的偏离量为 16 的字中，这个百分值在作为控制变量写出之前被转换为 0 到 16383 之间的数值。

不管超出界限使能位的状态，当计算的（或操作员输入的）控制变量值超出了输出 CV 的最大值（最小值）范围时，输出报警位、上限状态位（偏离量为 0/9）和下限状态位（偏离量为 0/10）会被置位。

控制块中的第二个字被用作误差状态字。当 SLC 想要执行 PID 指令时，如果 SLC 检测到一个主要错误，那么 SLC500 会：

1）置主要错误位，S：1/13 为 1。PLC 停止执行程序，除非有相应的错误处理程序，并且程序中含有清除此位的指令。

2）如果误差代码 H0036 被写入 S：6 中，意味着这个主要错误是由 PID 指令引起的。

3）写一个 8 位的误差代码到 PID 指令的控制块的偏离量为 1 的字中，我们就会知道为什么 PID 指令未被执行。大多数误差代码都是用来指出哪个控制块参数中含有不正确的数据，但有一个代码是意味着 PID 指令被频繁的中断，致使其无法正常工作。具体的误差代码见手册。

S5

4. SIEMENS STEP 5 PID 指令

Siemens S5 PLC 没有 PID 指令，但有 PID 组织块：OB251。OB251 可以预编程来进行 PID 计算。为了在 OB251 中运行 PID 程序，用户程序必须先调用含 OB251 所需要的数据的数据块，然后再跳转到 OB251 当中。数据块中的参数有：程序员在创建数据块时输入的一些初始参数，用户程序写入到数据块中的设定值和反馈值，还有 OB251 放入的 PID 计算结果。用户程序需要在 OB251 执行后从数据块读取结果。图 12-47 给出了一个这样的 STL 语言程序。

```
C DB4       ;数据块4中含有 PID 计算所需的数据
L IW72      ;从模拟输入模块读取反馈值
T DW22      ;将其放入 DB004的 DW22中
JU OB251    ;进行 PID 计算
L DW48      ;从 DW48中读取 PID 计算结果
T QW112     ;写入模拟输出模块
```

图12-47 使用 SIEMENS STEP 5 PID 指令

在 STEP 5 程序中，只有从功能块才能跳转到 OB251，所以图 12-47 的这些编码应当写在被用户程序调用的一个功能块里。为了保证 OB251 是按固定的时间间隔执行，一个好的办法是在 OB13 中调用此功能块，其中，OB13 是 STEP5 中固有的定时中断程序块（第 11 章中已讲过）。S5 PLC 可以按照保留系统字 97（RS97）中的数值定时的调用 OB13。STEP5 的初始化组织块可以通过调用有关功能块来向 RS97 中写入数值。图 12-48 给出了一个完整的程序例子。

```
OB21
组织块21是 STEP 5的一个初始化块,每次 PLC 从 STOP 模式切换到 RUN 模式后,它都会自动运行。在这里,
它用来调用功能块1去设定一个定时中断间隔。
JU FB1
    Init_ Int
BE
FB1
这个功能块的功能是设置定时中断为500ms(50个标准的10ms 间隔)
NAME: Init_ Int
DECL:
L KF+ 50
T RS97
BE
OB13
定时中断组织块,在 PLC 为 RUN 模式时,按照 RS97中的值定时的运行。在此,它用来调用 OB251
JU FB2
    PID_ exec
BE
FB2
此功能块用来为 PID 计算更新数据块中的过程变量值,然后跳转到 OB251来进行 PID 计算,最后把计算得
到的控制变量拷贝模拟输出模块。
NAME : PID_ exec
DECL :

C DB4
L IW72
T DW22
JU OB251
L DW48
T QW112
BE
```

图 12-48 定时执行 SIEMENS STEP 5 PID 指令

STEP 5 的另一个特性是提供了直接寻址访问功能（第 11 章中讲到）。直接寻址访问使大型 S5 PLC 能够直接读写 I/O 模块中的数据，也使中型 S5 PLC（例如，S5-103U）能够读写中断映像表。由于中型 S5 PLC 只在调用了中断服务程序（如 OB13）后才更新中断映像表，所以直接寻址访问可以只用在 OB13 这样的中断服务程序中。一个使用了直接寻址访问的程序如图 12-49 所示。在这个例子中，功能块 2 直接从输入模块读取数据（或从中断映像表中，而中断映像表中的数据是当 OB13 调用时从输入模块读入的），同时也直接向输出模块写数据（或向中断映像表中写，当 OB13 执行完成后，这些数据再会被送到输出模块）。

```
OB13
定时中断组织块,按 RS97定义的时间间隔运行,见图12-48
JU FB2
    PID_ exec
BE
FB2
调用 OB251的功能块,地址 IW72和 QW112改变为 PW72和 PW112
NAME: PID_ exec
DECL:

C DB4
L PW72
T DW22
JU OB251
L DW48
T PW112
BE
```

图12-49 使用 STEP 5的直接寻址访问功能

STEP 5 的 OB251 中的 PID 计算可以用做位置算法或速度算法，将独立增益方程和相关增益方程结合起来就是位置算法：

$$CV=K_c\left[K_p(E)+K_i\int(E)\mathrm{d}t+K_d\frac{\mathrm{d}(E)}{\mathrm{d}t}\right]+\text{偏移量}$$

SIEMENS 使用的符号有些差别，其方程为：

$$YA=K\left[R(XW)+TI\int(XW)+TD\frac{\mathrm{d}(XW)}{\mathrm{d}t}\right]+Z$$

其中：$YA=CV=$命令变量或控制器输出

$XW=E=$误差

$Z=$偏离量或扰动量

$K=K_c$[⊖]

$R=K_p$

$TI=K_i$

$TD=K_d$

对独立增益方程还可以有其他一些调整：

1）PID 计算中的微分项可以根据一个随机扰动（XZ）来计算，而不再是误差，XZ 可以是过程变量或任何一个量。后面会进一步讨论。

2）如果程序员希望在 PID 计算过程中微分项当作减数，那么 TD 可以取负值。

3）如果输入一个负值作为 K，那么 PID 计算的结果就正好相反了。

对于速度算法，PID 方程会计算出一个与位置算法结果的变化率成比例的值，方程如下：

$$YA=\frac{\mathrm{d}\left(K\left\{R(XW)+TI\int(XW)\mathrm{d}t+TD[\mathrm{d}(XW)/\mathrm{d}t]\right\}+Z\right)}{\mathrm{d}t}$$

当 OB251 中的输入数据（设定值、过程变量、扰动量和随机变量）保持不变时，速度算法的输出是 0。当其中一个或多个量变化时，速度算法输出将等于位置算法输出值的变化率（前提是使用了位置算法）。当被控系统的特性被检测到有变化时，速度算法可以调整驱动被控系统的信号，这样，被控系统的输出就会基本保持稳定。举个例子，操作员手动设置传送带的速度，然后按下“设置速度”开关，然后让速度算法取代速度控制。当实际速度、外界扰动或设定值改变时，速度算法就会产生一个纠正因子。当 OB251 产生的结果不为 0 时，把速度算法的输出值加到最初手动选择的值上。图 12-50 给出了这个例子的整个程序。

当传送带的速度不变时，图 12-50 中的速度算法的结果为 0，而且 QW112 中的数值也不会改变。但是当设定值改变时，OB251 就会产生一个正值加到位于地址 QW112 中的数值上，而这个数值会被写入模拟输出模块。更高的模拟输出值将使电机转速更快，并导致传送带加速。当速度增加时，过程变量的变化（取自 IW72 并通过 DW22 提供给 OB251）会导致速度算法产生一个负值。负值将加到 QW112，使模拟输出返回到初始值。当电动机的速度增加的值与设定值增加的值相同时，QW112 中的值就会恢复为初始值，这个值足以保持现速。

⊖ K 以单位 1/1024（1/1000）输入，输入 1024（或 1000）意味着 1（1024/1024=1）。

```
PB1
     每当操作员按下"设置速度"开关时,程序块1就会被调用一次。它的功能是读取一个手动的设定值,
并提供给直流电动机作为初始速度。
     L IW72;电动机的初始速度设定值
     T QW112
     BE

OB13
     定时中断块,用来调用 OB251的功能块
     JU FB2
     PID_ exec
     BE

     FB2
     为 PID 计算更新过程变量数值,跳转到 OB251执行 PID 计算的速率算法,然后调整模拟输出值
     NAME : PID_ exec
     DECL :
     C DB4 ;包括配置 OB251作为速度算法
     L IW72
     T DW22
     JU OB251
     L DW48
     L QW112 ;从模拟输出模块读当前输出值,
     + F     ;并将 DW48中的值加给它
     T QW112 ;然后写改变后的值到模块
     BE
```

图12-50 使用 STEP 5的速度算法

配置 STEP 5 的 PID 计算

程序员选择使用 PID 方程的类型并在数据块中输入 OB251 所需要的参数，数据块的 49 个数据字中有些是一定要在程序运行前输入的，其他的一些字由用户程序写入。例如 OB251 要在数据字 22 中得到过程变量值，那么用户程序就需要在程序跳转到 OB251 之前读取反馈传感器，把值写入到 DW22 中。图 12-51 给出了一些重要的数据字，我们也可以从中看出 OB251 如何使用它们，这些数据字在下面会给出解释。

图 12-51 摘自 S5-103U 附带的用户手册，而图 12-52 是根据作者观察得到的。二者之间有一些不同，所以我们可以说现在市场上有两种版本的 OB251。[⊖]图 12-52 中标出了不同之处。现在就来讲讲这两种版本，暂且称为“资料版”和“观察版”。

若想进行 STEP 5 PID 计算，必须输入的数据包括配置位和数据字。其中，配置位在数据字 11 中，SIEMENS 称此字为 STEU。数据字 11 的低 6 位包括：

1）手动/自动模式选择位（位 0，STEP 5 用户手册上称之为 S2） 在自动模式下（S2 =1），DW48 中的输出值是 PID 的计算结果。在手动模式下，OB251 将 DW12 中的手动输入值拷贝到 DW48 中（受位 3 和 4 的限制），这个值不能超过 DW48 的上下限要求。

用户程序监控手动输入开关来控制 S2 位，并可以从人控表盘上读取模拟数据送到 DW12 中。在功能块中使用 STL 语句的无条件置位指令（SU）和无条件复位指令（RU）

⊖ 作者使用的 PLC 是 westinghouse 的 PC503，其实就是 Siemens S7＝103U，在北美由 Westinghouse 销售。

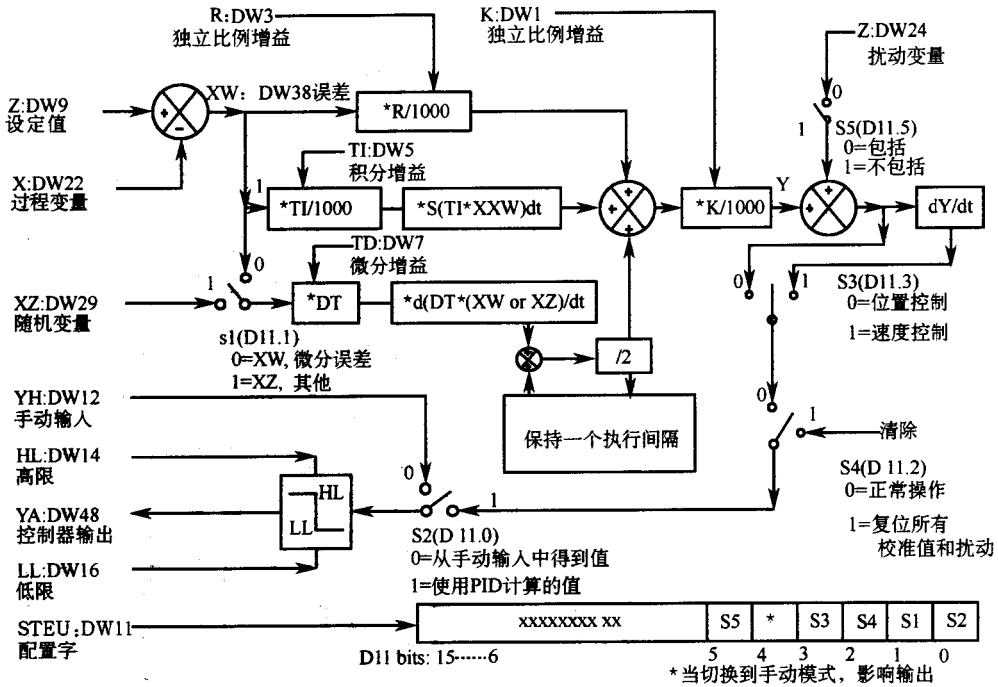


图 12-51　STEP 5 的 OB251，摘自用户手册

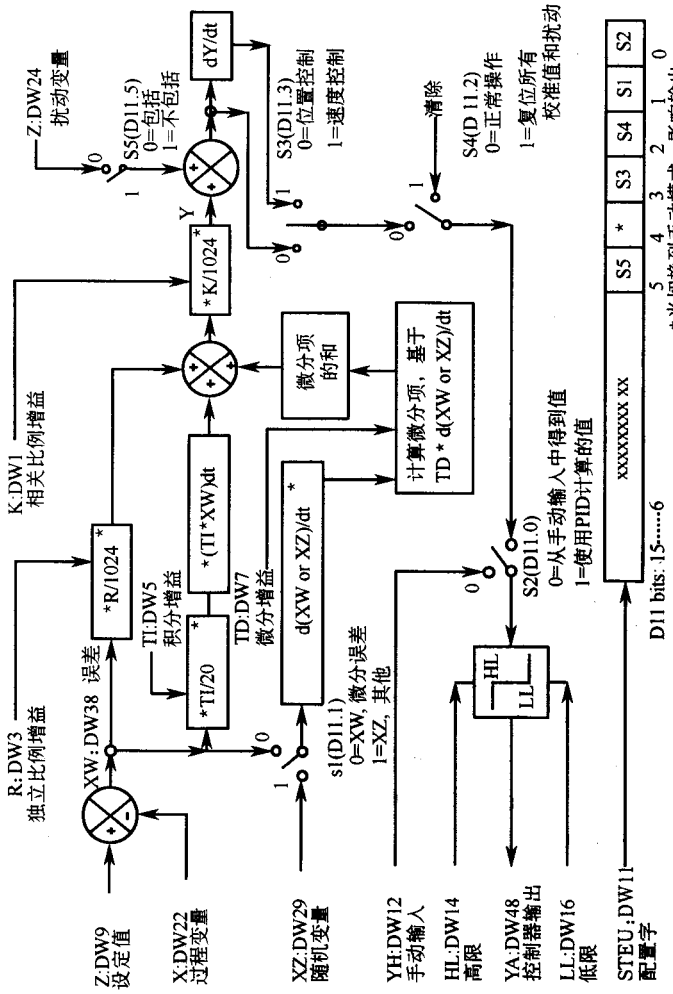


图 12-52 根据观察画出的 STEP 5 OB251 图

控制 S2，因为标准的布尔逻辑指令不能改变数据字中的单独的位。梯形图中无 SU 和 RU 指令；我们只能用字逻辑指令将 S2 置 0（如图 12-53a 所示）或置 1（如图 12-53b 所示）。

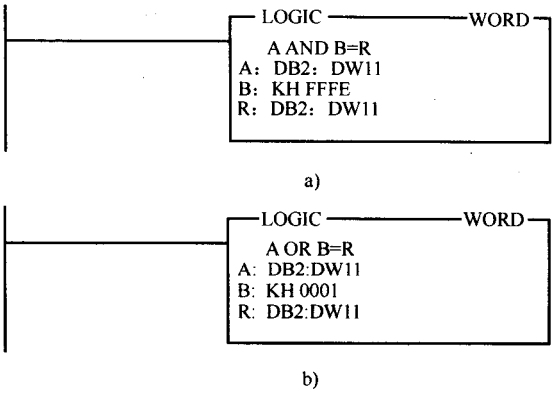


图 12-53 用逻辑指令改变字中的某一位 a）用 AND 指令置 D11.0 为 0；b）用 OR 指令置 D11.0 为 1

2）微分值选择位（位 1，S1） 当 S1＝0 时，微分项根据误差变化率（XW）进行计算是常规的方法。当 S1＝1 时，OB251 将按照 DW29 中的随机变量（XZ）的变化率进行计算，并将其结果用到 PID 方程中。

有时，程序员希望使用过程变量的微分代替误差，如下式：

$$YA=K\left[R(XW)+TI\int(XW)\uparrow TD\frac{\mathrm{d}(XZ)}{\mathrm{d}t}\uparrow\right]+Z$$

所以程序必须把从输入模块得到的过程变量值拷贝到过程变量数据字（DW22）和随机变量数据字（DW29）中。要使微分项充当减数，微分系数必须为负。

随机变量还可以是其他量，而非过程变量。有时会把设定值的微分量加到输出当中去，这样，系统就可以更好的对变化快的设定值作出反应。

3）算法选择位（位 3，S3） 当 S3＝0 时，位置算法被选中；当 S3＝1 时，速度算法被选中。选择何种算法，会影响到 STEU 的第 4 位做什么。

4）NO JERK 选择位（位 4，这是 STEP5 编程手册中惟一一个不以“S”命名的位）如果 OB251 选择使用了位置算法，而且 D11.4＝0，“NO JECK”则被选中。这样，当 PID 计算切换到手动模式后，OB251 将会取 DW12 中的数据作为输出值。当此位为 1 时，OB251 将会把手动模式下的最后一个值作为输出，并保持不变。

如果使用速度算法，而且 D11.4＝0，OB251 将会把手动模式下的输出改为第一个计算间隔时手动输入的数值。当此位为 1 时，OB251 的输出（手动模式下）将为 0。

5）扰动量选择位（位 5，S5） 当 S5＝0 时，DW24 中的扰动量将被引入到计算中。

6）复位位（位 2，S4） 如果 PLC 程序将此位先置 1，后清 0，那么所以的计算结果，包括积分误差、微分项、过程变量和扰动量都将被清 0。而设定值、增益值、输出限定值和手动输入值不会受其影响。

有些数据字必须在 OB251 调用之前输入到数据块中，完成对 PID 计算的配置。其中一些数据字在程序运行时会被改写，来调节计算方式。另一些（例如设定值、过程变量、控制变量）则必须由用户程序进行读写。手册上规定，这些数据字的取值范围是 2047 到－2047（除了下面提到的

例外情况），这个范围是与 12 位有符号二进制数对应的，但作者认为－32768 到＋32768（16 位有符号数）效果会更好些。使用较大的数据，得到的结果超出 16 位数值的范围的可能性会增加。OB251 不会对输入值或输出值进行标度变换，所以用户程序需自己具有此功能。

OB251 数据块中的 49 个数据字中，用户可以对如下几个进行操作：

1）W　即设定值（存入 DW9 中）。通常是由用户程序将其写入数据块，如前所述。

2）X　即过程变量（存入 DW22 中）。从反馈传感器中读取，并在 OB251 执行前被拷贝到数据块中。将其写入到 DW22 中之前，用户程序可能需要对过程变量进行标度变换，以保证其取值范围与设定值的相同。

3）K　即比例动作系数（存入 DW1 中），有时也叫做相关比例增益。手册上规定其取值范围为－32768 到 32768，但在使用前要除以 1000，所以值为 2000 的 K 实际上是 2（但作者发现，实际的除数不是 1000，而是 1024，所以要想得到 2 就要输入 2048[⊖]）。将这个值置为 0，PID 计算的结果就会变为 0，那么 OB251 的控制变量输出值就会完全等于扰动量的值。在 DW1 中放入一个负值，OB251 输出的控制变量就会正好相反。如果 DW1＝1000，PID 方程其实就是独立增益方程。

4）R　即独立比例增益（存入 DW3 中），与相关比例增益一样，取值范围是－32768 到 32768，使用前也会除以 1000（同样观察发现实际除数是 1024）。令 DW3＝1000，PID 方程即为相关增益方程。

5）TI　即积分增益（存入 DW5 中）。手册上规定其范围是 0 到 9999，在使用前同样会除以 1000（但作者发现实际的取值范围比手册规定的要大，而且使用前会被 20 除）。OB251 计算出的输出值超过上下限时，积分输出值不会随其变化。（HL 和 LL 在后面会提到。）

无论实际的两次 OB251 执行间隔是多少，PID 计算总是认为计算是每秒钟执行一次。假设误差保持 30 不变，TI 等于 2000，OB251 每秒钟执行一次，那么在 5S 内，积分项的输出为：

$$\text{积分器输出}=\frac{TI}{1\,000}\times\text{误差}\times\text{重复次数}=\frac{2\,000}{1\,000}\times30\times5=300$$

另一种情况是，OB251 每半秒钟执行一次，那么 5 秒内会执行 10 次，这时的积分输出为：

$$\text{积分器输出}=\frac{TI}{1\,000}\times\text{误差}\times\text{重复次数}=\frac{2\,000}{1\,000}\times30\times10=600$$

所以程序员就要保证 OB251 是准确的每秒钟执行一次，否则就需要对积分项作相应的修改。

6）TD 即微分增益（存入 DW7 中）。在这里，手册上所说的和实际观察到的有很大的差别：

■ 手册上取值范围是 0 到 999，微分方程默认两次的执行间隔为 1s。程序员需要确保 OB251 是每秒钟执行一次，否则就要对 *TD* 作相应的调整。

假如，OB251 每秒钟执行一次，误差以每秒 50 的速率变化，*TD* 为 2，这时，微分项的输出为：

$$\text{微分输出}=TD\times\frac{\text{误差变化率}}{\text{重复次数}}=2\times\frac{50\text{s}^{-1}}{1\text{s}}=100$$

但如果 OB251 是每半秒钟执行一次的话，那么计算时误差每秒钟的变化只有一半（即 25）被观察到，这时就需要将 *TD* 扩大一倍才能弥补失去的那一半误差变化量：

⊖ 读者可以自行检验下手头的 OB251 是哪个版本的，选择位置算法，但不要连接传感器或执行器，置 TI 和 TD 为 0，取设定值和过程变量为明显不同的两个值，然后分别输入 1000 和 1024，看是哪个数会使 DW48 中的被控变量与设定值－PV 相同。

$$\text{微分输出} = TD \times \frac{\text{误差变化率}}{\text{重复次数}} = 4 \times \frac{25\text{s}^{-1}}{1\text{s}} = 100$$

手册中还提到，每次计算得到的新微分值都会和前一次的结果求平均，这样微分项的输出会平滑许多，如图 12-51 所示。求平均后的值才被用到 PID 计算当中，并且会存储起来作为下一个值的前值使用。微分项的平滑度同样受 OB251 的实际执行间隔所影响。如果 OB251 每半秒钟执行一次，突然的误差变化所造成的影响会比 OB251 每秒钟执行一次的情况下小两倍。

- 作者发现－32768 到 32768 之间的数都可以输入，而且（正如图 12-52 所示）微分项的输出不会随着时间而衰减平滑。每次误差变化时，微分项的输出就会有一个与最近误差变化大小（或正或负）和微分增益（*TD*）有关的一个增量（这个增量的大小不会超过有符号 5 位数的取值范围）。如果误差持续增长，微分项的输出会不断累加增量；如果误差开始变小，负的增量会加到微分项的输出中。如果误差的变化率为 0，平滑增量不会被加入，所以微分项的输出不为 0。增量会随着误差变化率的增加和 *TD* 取值的变大而增加，但不是完全的线性关系。经过几次 OB251 的运行后，不平滑现象一般情况下会消失，但程序员应该清醒的认识到微分项可能会累加一个很小但作用明显的偏离量，这样它在应当等于 0 时却不为 0。对于上例，图 12-54 给出了另一个程序，它的不同之处是当上一次 OB251 执行后误差仍以微小值存在并保持，从而清除了 OB251 的累加量。（注意：这样做同样会清除积分项，这可能会引来错误。）

7）HL 和 LL　即输出值的上限（DW14）和下限（DW16）。这是 OB251 允许输出值达到的范围。如果 PID 计算的结果加上扰动量（或手动输入值）后超过了此范围，那么输出值就会被限定为上（或下）限的值。

当输出为限定值时，积分项不再增加，这样可以防止积分项过大增加进而导致超调的发生，所以程序员要选择合适的 HL 和 LL。

8）Z　即扰动变量（DW24），有时也叫做*偏移量*或*偏离量*。通过设置 STEU，可以在计算速度算法输出之前将扰动量加到 PID 位置算法的计算结果中去。实际上，当切换到自动模式时，OB251 计算的并加到 PID 计算结果中去的是扰动量的变化量。

9）YH　即手动输入（DW12）。当 OB251 处于手动模式下（对 OB251 配置时选择）时，这个值取代 PID 计算结果作为输出值。

10）XZ　即随机变量（DW29）。通过配置，可以不计算误差的微分，而改为计算 DW29 中值的微分。如果需要计算输出的微分，用户程序可将过程变量拷贝到 DW29 中，如果希望计算设定值的微分，那么就将设定值拷贝到 DW29 当中。

11）YA　即控制变量（DW48）。OB251 将最终结果存放到这个地址中。每次执行完 OB251 后，用户程序都会把这个地址中的数据拷贝到模拟输出模块以控制过程。在使用前也可能需要对此输出值进行标度变换。

数据块中的其他数据字是 OB251 用来存储数据的，一般不能由程序员改动。

S7

5. SIEMENS STEP 7 PID 指令

只有少数几个 S7 PLC 内置有 PID 控制算法（S7-215、S7-216 和 S7-314IFM）。在这些 PLC 中，PID 算法包含在系统功能块中，所以能够用于控制多个过程，其中每个过程都要求有自己的数据块来存储参数和工作数据。

如果你现有的 PLC 没有内置 PID 算法，那么就只能购买一个功能块或自己将 PID 算法写入。

```
这个块执行早期的例子中的OB251，但是包括一个到纠正累积微分误差的例行
程序的跳转
NAME: PID_exec
DECL:

C DB4
L IW72
T DW22
JU OB251
L DW48
T QW112
JU FB3
          Corrrection

BE

FB3
如果误差很小并稳定，那么纠正累积的微分误差

NAME:Correction
DECL:

L DW38      ; 加载OB251最近计算的误差值
L KH0008    ; 并装载一个小的正数（8）
>F          ; 检查误查是否大于8
BEC         ; 如果是，则终止FB3的执行

L DW38      ; 再次加载误差值
L KHFFF8    ; 并加载一个小的负数（-8）
<F          ; 检查误差是否小于-8
BEC         ; 如果是，终止FB3的执行

L DW50      ; 加载一个仅被FB3使用的字（先前值）
L DW38      ; 来自OB 251的最近的误差
T DW50      ; 为下一次FB3的运行存储新的误差
><F         ;如果新的误差不同于旧的差错
BEC         ;如果是，终止FB3

L KH000     ; 如果误差小则清除PID计算结果
T DW44      ;保持不变直到下次FB3执行
BE
```

图 12-54 清除 STEP5 中累加的微分误差

S7-314IFM 中内置的 PID 控制算法是比较好的位置控制型算法，它使用的是高精度的浮点运算。我们将在本节详细讨论它的工作原理和使用方法，这样，读者也可以依此写出自己的功能块。

S7-314IFM 用于 PID 控制的内置系统功能块叫做 SFB41 “CONT _ C”。可以将过程变量作为一个实数（PV _ IN）提供给 CONT _ C，也可以把 16 位的有符号数作为外围输入的地址（PV _ PER）提供给 CONT _ C。CONT _ C 会把 PV _ PER 转换为实数，并对其进行标度变换。CONT _ C 完成计算后，将以实数的形式（LMV）输出控制变量，同时还以 16 位有符号数的形式（LMV _ PER）送给外围输出。

图 12-55 给出 CONT _ C 如何根据过程变量计算控制变量输出值。如无特殊标注，所有的输入和输出参数都是浮点（实）数或布尔位。下面的内容摘自 STEP7 的用户手册，并且附上了作者自己对 PID 算法的理解：

1）过程变量可以以整型输入参数（PV _ PER）的形式输入。PER _ VAL 由 16 位有符

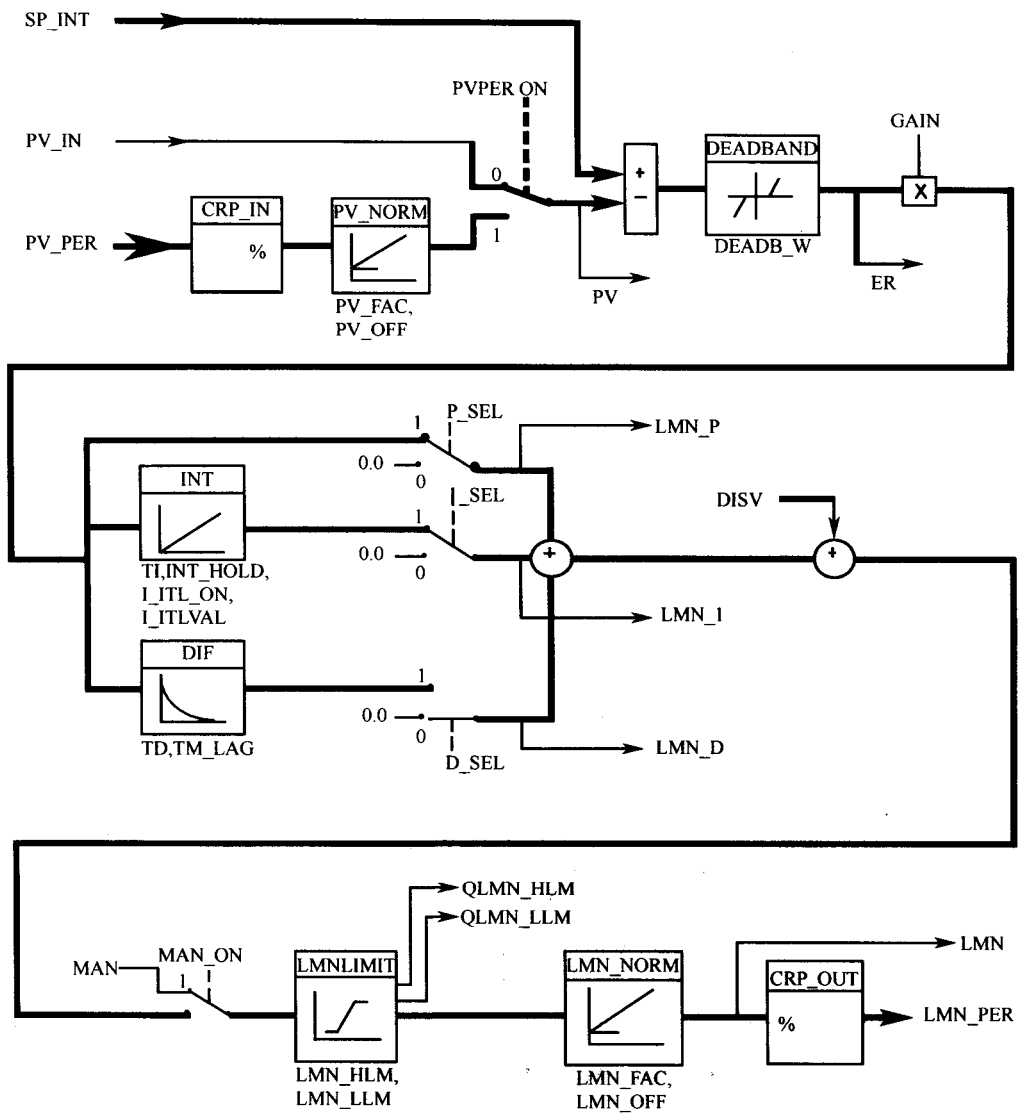


图 12-55 STEP 7 的 CONT _ C 流程图

号二进制数转换成 32 位二进制浮点（实）数，然后在与过程变量的标度变换乘数(PV _ FAC)相乘。得到的结果再与偏离量（PV _ OFF）相加得到最终结果，如果此时 PVPER ON 位为 1 的话，这个最终结果就作为过程变量（PV）；否则，由 PV _ IN 提供的实数作为过程变量（PV）。

2）设定值（SP _ INT）与过程变量的差值会被计算出。

3）计算出误差后，CONT _ C 检查它的绝对值是否大于死区值（DEADB _ W）。如果不大于，CONT _ C 会在以后的计算中令误差为 0 以代替真实误差，这样，微小的误差无法使积分项不断增加。

4）然后令误差值（现叫做 ER）与比例增益相乘，如果这时比例输出（P _ SEL）为真，

那么这个积分结果就会被用于 PID 计算。

5）将 GAIN×ER 的结果的备份除以积分时间（TI，以时间形式输入）再乘以自上次 CONT _ C 执行到此刻的实际时间（CYCLE，以时间形式输入）。如果积分项未被冻结（置 INT _ HOLD 为真可冻结积分项），这个新的计算结果将被加到积分项中。如果积分输出（I _ SEL)为真，那么积分结果将会被用到 PID 计算当中。

6）GAIN×ER 的微分（即变化率）是这样计算的：将 GAIN×ER 的值与上一次 CONT _ C 执行时得到的 GAIN×ER 值相减，再让这个差除以两次 CONT _ C 执行的间隔时间（CYCLE)。将此结果再乘以微分时间（TD，以时间形式输入），然后用时间滞后值（TM _ LAG，以时间形式输入）来平滑突变。此处未解释 TM _ LAG 如何起到平滑作用，但读者如果想做到这点，可尝试如下方法：

- 存储最近几次（10 或 20）计算得到的微分值到一个队列，每当有新值加入队列就抛弃队列最前面的（即最老的）的一个微分值，计算队列中数据的平均值即可。新加入队列的微分值会立刻影响到平均值，它对平均值的影响将会重复 10 或 20 次。
- 将新计算得到的微分值加到已经平滑处理的微分值上，然后再除以 2。将此结果输出，并将之保存充当下一次计算的平滑的微分值。如果想调节最后计算得到的平滑值，可以在新微分值和已处理的平滑微分值相加前将它们分别乘以一个权重。如果已处理的微分值的权重大，那么结果会更加平滑。

经过平滑处理后，如果微分输出 D _ SEL 为真，那么得到的微分结果就将用于 PID 计算当中。

7）将比例项、积分项和微分项相加（LMN _ P+LMN _ I+LMN _ D)。

8）将扰动量（DISV）加入到 PID 结果中。

9）如果 MAN _ ON 为真，计算的结果将会被手动输入值（MAN）所取代。

10）结果还要与上限（LMN _ HLM）和下限（LMN _ LLM）比较。如果超出范围，那么结果将被舍弃，改取限定值，同时，CONT _ C 还会把状态位 QLMN _ HLM 或 QLMN _ LLM 置 1。

11）结果还要和标度变换因子（LMN _ FAC）相乘，然后再加上偏离量（LMN _ OFF)。现在得到的结果就是实数形式的控制变量了，即 CONT _ C 的 LMN 输出参数。

12）实数形式的结果被转换成整型数，成为了 CONT _ C 的 LMN _ PER 输出参数。

上面所讲的计算顺序与相关增益方程的运算等同（如果忽略了数值转换、标度变换、积分冻结、限幅和时间延时等)：

$$LMN=GAIN\left[(ER)+\frac{CYCLE}{TI}\int(ER)\mathrm{d}t+\frac{TD}{CYCLE}\frac{\mathrm{d}(ER)}{\mathrm{d}t}\right]+DISV$$

与 CONT _ C 执行的 PID 计算功能基本相同的一个功能块如图 12-56 所示。这个例行程序不去检查状态字位以保证无错误的操作，也不提供 CONT _ C 中执行的数值转换、标度变换、积分冻结、限幅和时间延时等功能。（作者在其中使用了与 CONT _ C 相同的参数名，但为了以示区别，在每个名字前加了前缀 M _ 。)

S7-314 还提供 SFB 42 “CONT _ S”，它将一个与设定值和过程变量间的误差成比例的量与积分项相加，其中，积分项是与过去的累积误差和 CONT _ C 的累积输出成比例的。这个结果是一种 PI 控制，积分项与未校正的误差成比例。CONT _ C 也会把计算结果从实数

地址	声明类型	名称	数据类型	初始值	注释
0.0	in	M_SP_IN	REAL	0.0	设定值
4.0	in	M_PV_IN	REAL	0.0	过程变量
8.0	in	M_GAIN	REAL	1.0	Kp
12.0	in	M_TI	REAL	0.0	以秒计算
16.0	in	M_TD	REAL	0.0	以秒计算
20.0	in	M_DISV	REAL	0.0	扰动
24.0	in	M_CYCLE	REAL	1.0	以秒计算
28.0	out	M_LMN	REAL	0.0	控制变量
32.0	stat	M_OLD_P	REAL	0.0	保存误差
36.0	stat	M_LMN_I	REAL	0.0	积分
0.0	temp	M_LMN_P	REAL	0.0	比例项
4.0	temp	M_LMN_D	REAL	0.0	微分项

```
网络1
      计算误差并产生比例项
      L #M_SP_IN     // 设定值
      L #M_PV_IN     //减去过程变量
      -R             //等于误差
      L #GAIN        //乘以独立比例增益
      *R             //=比例项
      T #M_LMN_P     //保存让其他网络使用

            网络 2

                  产生和保存积分项

                  L#M_LMN_P          // 带独立比例增益的误差
                  L#CYCLE            // 乘以上次到现在的时间间隔
                  *R                 // 计算
                  L#M_PI             // 除以积分时间
                  /R                 // 等于要积分的值
                  L#M_LMN_I          // 乘以先前的积分值
                  +R                 //= 积分项
                  T#M_LMN_I          // 为网络 4 保存

            网络 3

                  产生微分项，然后保存当前误差为将来使用

                  L#M_LMN_P          // 带独立比例增益的误差
                  L#M_OLD_P          // 从先前值中减去相同的值
                  -R                 // 计算
                  L#M_CYCLE          // 被从上次到现在的实际时间除
                  /R                 // 计算
                  L#M_PD             // 乘以微分时间常数
                  *R                 //=微分项
                  T#M_LMN_D          // 为网络 4 存储

                  L#M_LMN_P          // 为下次计算间隔保存
                  T#M_OLD_P          //
```

图 12-56 STEP 7 完成 PID 控制的功能块

```
网络 4
        将 P、I、D 项相加

        L#M_LMN_P           // 比例项
        L#M_LMN_I           // 加积分项
        +R
        L#M_LMN_D           // 加微分项
        +R                  //=PID 计算结果
        T#M_LMN             // 作为输出参数
        BE
```

图 12-56 （续）

型转换为两个布尔输出，这样可以切换正负能量供给，分为三级（正、0 和负）PWM（脉宽调制）来控制直流电动机。

CONT _ S 自动地调用另一个系统功能块，SFB 43 “PULSE _ GEN”，它将 PI 计算结果转换为 2 位 PWM 输出。用户程序可以调用 PULSE _ GEN 来将任意实数（例如来自 CONT _ C 的输出）转换为三级 PWM 信号。PULSE _ GEN 被用户程序调用来产生：

1）对称三步输出，即整个实数范围与正负 PWM 输出线性的对应起来。

2）非对称三步输出，即正 PWM 输出与负 PWM 输出的增益不同。

3）双极两步输出，即全体实数与正 PWM 输出对应。

4）单极两步输出，即正实数与正 PWM 输出对应，负实数对应为 0。

5）以上任何一种，但是带有最小脉冲持续时间和无脉冲持续时间（使 0 附近的实数产生一个死区），并带有输出激励，如果实数接近它们的最大值，输出激励会产生最大 PWM 输出。

6. OMRON CQM1 PID 指令

OMRON 的 PID 指令只有在 CQM1 CPU4x PLC 中才可以使用，它进行 PID 计算是基于比我们书中目前所提到的都要老一些的相关增益方程。它用比例带取代了无单位的比例增益，并且最后不加偏离量。还有一个不同点是它会根据执行的时间间隔来自我调整。OMRON 的 PID 方程为：

$$CV=\frac{PB}{100}\left[E+\frac{1}{R_i T_n}\int (E)\,dt+R_d T_n\frac{d(E)}{dt}\right]$$

其中，

PB ＝比例带，以百分数形式（%）$=\dfrac{\text{比例输出的变化}}{\text{误差值的变化}}\times 100=K_c\times 100$

在标准的独立增益方程中输入一个你想要的 K_c 的 100 倍的值。

T_n 是 PID 计算的采样周期（执行时间间隔）。为 R_i 和 R_d 分别输入一个比例值，它们的意思是有多少执行间隔时间用作积分时间（T_i）和微分时间（T_d）。

$$R_i=T_i/T_n;$$

$$R_d=T_d/T_n$$

增加或减少 T_n 会自动的改变用于计算的积分时间和微分时间的值。

如图 12-57 所示为 PID 梯形图元素 PID(－)。PID(－)指令没有微分形式。如果控制逻辑为真（例如图 12-57 中的 IR00204 为开），PID 指令执行。首先，从 IW 指定的地址中读取过程变量（本例为 IR005），然后按照从 P1 指定的地址开始的 33 个数据字（本例中为 DM0020 到 DM0052）中的配置进行 PID 计算，最后把结果写入到 OW 参数指定的地址（本

例中为 IR106）中。PID 的配置需要在 PLC 进入运行模式前被输入，在程序运行过程中也可以被改动[⊖]。当 PID 指令逻辑持续为真时，一个配置参数可以令 PID(－)按照一定的速率（由 PID(－)中的定时器控制）重复计算输出。因为 PID（－）指令要检查定时器的累加值，所以执行间隔可能不会严格准确。当累加值大于设定值时，PID 计算便会开始，定时器也会开始重新计时。此时的定时器的累加值不会被清 0，而是被设置为预置值与 PID 计算实际开始执行的时间的差值。这样做可以使执行间隔更准确些。

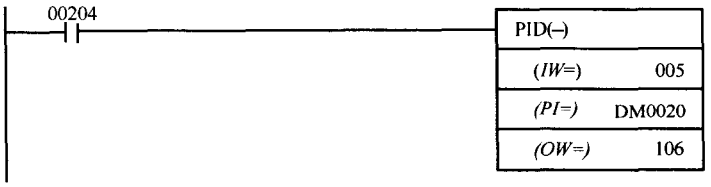


图 12-57 CQM1 PID 指令

由于 PID(－)指令要不断的去更新检查定时器，所以它需要在每个扫描循环都被执行。如果它被放置在可能被跳过的程序段内、或者因互锁而被抑制执行、或者在需要满足一定条件才能被调用的子程序里、或者在中断程序里，PID(－)就无法很好的工作。注意，OMRON 的 PID(－)指令不要放在定时中断程序中。

如果 PID(－)指令的控制逻辑为假，那么它就不能把计算结果写到 OW 地址中，这时，PLC 会利用其他指令向这个地址中写入数据，即系统工作在手动模式下。当 PID(－)指令的控制逻辑由假变真时，它就会按照最新的配置参数进行计算，而输出地址（有 OW 参数指定）中的值会渐变到 PID 的计算结果。这样做是为了防止在 PID 刚开始起控制作用时输出量的抖动过大。

对 PID 指令进行配置的参数被存储到从 P1 指定的地址开始的 33 个连续的地址中，大部分的参数无需用户或用户程序改变，但前 7 个是可以改写的。（只有当 PID(－)指令的控制逻辑由假变真后，这些参数的改变才会对 PID 计算产生影响。）这前 7 个数据字中的参数包括：

1）设定值（SV） P1 范围内的第一个数据字。输入的值应当在输入范围（后面会讲到）指定的工程数据范围内。要使其生效还需要使 PID(－)指令的控制逻辑产生一个正跳变。

2）比例带（PB）的设置 第二个数据字。可以输入 0001 到 9999 之间的 BCD 码，然后会被解释成 0.1 到 999.9%的比例带的值。

3）积分时间比例值（R_i） 第三个数据字。输入 0000 到 8191 之间的 BCD 码。这个值与采样周期（T_n）相乘得到积分时间（T_i）。如果输入 9999，积分项便不可用。

4）微分时间的比例值（R_d） 第四个数据字。输入 0000 到 8191 之间的 BCD 码。这个值与采样周期（T_n）相乘得到微分时间（T_d）。如果输入 0000，微分项便不可用。

5）采样周期（T_n） 第五个数据字。输入 0001 到 1023 之间的 BCD 码，然后会被解释成从 0.1 到 102.3s 之间的时间。这个值即为 PID(－)指令的内置定时器的预置数，当 PID（－）指令的控制逻辑为真时，控制指令的执行间隔。

6）操作说明 第六个数据字的低四位。

⊖ PID（－）指令配置参数的改变只有在 PID（－）指令的逻辑由非跳变为真时才生效。如果想在系统运行时调整算法的增益；你必须编程允许新参数的接受。

- 如果这个低 4 位为 0，那么后作用功能被打开。这时，误差会按照前面所说的常规算法来计算，即用第一个数据字中的设定值减去来自 IW 的过程变量。此方法用在通过加热来控制水温的情况下。当水温过低时，我们就需要加大加热器的输出。
- 如果这 4 位均为 1，那么 OMRON 所称的常规功能被打开。这时，误差的计算方法与我们前面提到的后作用算法类似：设定值从过程变量中减掉。此方法适用于通过控制冷水流入量来控制水温的情况。当水温过低时，我们需要减少冷水的流入量。

7）输入滤波器系数　第六个数据字的高 12 位。可输入 000 到 999 之间的 BCD 码，对应于时间常数 0.0 到 0.999s。高滤波器系数有助于减少由快速变化（包括电子噪声和尖峰脉冲）造成的不期望的影响。PID 计算中使用的是来自 IW 并经过处理的过程变量，这样为的是在一定的时间段内只有 65%的变化会被认可。输入 000 可以关闭输入滤波功能。

如果你有电学理论基础，就比较容易理解观察电容器与电阻串联充电时“时间常数”的含义了。当改变供电电压时，在每个时间常数内只能观察到 65%的变化。

8）输出范围　数据字 7 的低 8 位。可以输入 00 到 08 之间的 BCD 码，作用是在 PID 计算得到的控制变量被送到 OW 指定的地址之前将其转换为 8 到 16 位的数值（带工程单位）。

9）输入范围　数据字 7 的高 8 位。可以输入 00 到 08 之间的 BCD 码，作用是从 IW 指定地址得到的过程变量被用于 PID 计算之前转换为 8 到 16 位的数值（带工程单位）。数据字 1 中的设定值的转换方法与过程变量相同。

注意，PID(—)指令的标度变换功能很有限，它甚至不能加入偏离量。所以在你的 PLC 程序中需要编写语句来对使用前的设定值和过程变量还有计算得到的控制变量进行标度变换。

如果在 PID(—)指令执行期间发现了无效参数，那么 ER 位（SR25603）会被置 1，同时 PLC 进入错误模式。当 PID 指令开始进行 PID 计算时，CY 位（SR25604）会被置 1。如果不执行 PID 计算，此位会被 PID（—）指令清 0。

PID 控制算法的整定

整定 PID 控制算法并不容易，尤其是在数字控制器（如 PLC）中实现的算法，因为有太多的因素会影响被控系统的性能。下面列出的几步可以帮助你做一些近似，但最后还需要由安装人员来调整。

1）最初，按照固定的时间间隔执行只有比例控制的算法（时间间隔可以取 PLC 正常的扫描循环的 3 到 5 倍）。如果安全方面无限制，先不要对输出值的大小进行限幅。

2）测量出被控系统的死区时间：在手动模式下，改变控制变量值使可明显观察的反应出现。计算从控制变量改变到系统做出反应之间的时间 。

3）测出被控系统的自然周期时间（自然的或共振的频率）：在自动模式下，将比例增益取的小一些，输入一个改变了的设定值，然后观察被控系统。正常情况下，系统应该会去追赶设定值，但会很慢。然后逐渐取大一些的比例增益，重复上述试验，直到有明显的超调出现。再继续取大的比例增益直至改变设定值会引起系统无休止的振荡。如果振荡大到限定值，则改用小一点的设定值。观察并记录系统一个完整振荡周期的时间，记录这个自然周期的时间和在自然频率下导致系统共振的比例增益。

4）如果系统的自然周期时间不足死区时间的 4 倍，调整 PID 控制算法就会很麻烦。如果可能的话，调节被控系统，降低它的死区时间或增加自然周期时间（如果死区时间无法降低的话），然后再重复第 2、3 步。

5）改变 PID 计算执行的时间间隔，使其能够在一个系统周期内执行最少 5 次，但也不要太快而使 PLC 中其他的控制功能受影响。一般来说，PID 指令的执行周期不能小于扫描循环的五分之一或被控系统自然周期时间的十分之一。切记：当你改变 PLC 程序调用 PID 指令的频率时，有可能同时也改变了包含回路更新时间在内的 PID 参数。

6）按如下方法计算和输入比例系数、积分系数和微分系数。记住有的 PID 指令会将你所输入的数值除以某个数后再用于 PID 计算。还有，不能将分钟与秒相混淆。

- 将刚才步骤 3 中系统振荡时的比例增益值的一半作为这里的比例系数（K_c 或 K_p）。
- 按 PID 方程的要求输入积分系数：
  - 将观察到的系统自然周期时间作为积分时间（T_i）；
  - 相关增益方程中的积分系数为 $1/T_i$；
  - 用 K_p 除以 T_i 得到独立增益方程的积分系数（K_i）。
- 按 PID 方程的要求输入微分系数：
  - 用系统自然周期时间的八分之一（或上一项中的积分时间的八分之一）作为微分时间（T_d）；
  - 令 T_d 乘以 K_p 得到独立增益方程的微分系数 K_d。

模糊逻辑 模糊逻辑控制算法不需要像 PID 控制算法一样整定，而且现在已经证明在控制复杂系统方面模糊逻辑控制有着无可比拟的优势。模糊逻辑作为一种控制方法在北美并没有被迅速接受，所以直到编写此书时它在 PLC 控制方面的应用仍不普遍。有的 PLC 供应商提供的智能 I/O 模块中包含了模糊逻辑功能，而有的提供了模糊逻辑的软件支持，但本书讨论的 PLC 均未提供预编的模糊逻辑指令。也许在你学习本书时，预编的模糊逻辑指令已经出现，所以我们在这一节里介绍模糊逻辑的有关概念。

模糊逻辑分三步计算最优控制变量值：1）模糊化；2）近似推理；3）去模糊化

模糊化 过程变量与设定值作比较并计算误差，从前一步执行期就开始计算误差的变化量。经过计算，误差（E）和误差的变化量（CE）为原始形式（它们在数值上等于实际的误差和误差的变化）。它们必须在进入近似推理步骤前转化为模糊形式。

在模糊化时，原始的 E 和 CE 值被转化为使用隶属函数表表示的模糊值。沿着 x 轴的隶属函数读 y 轴的原始值，如图 12-58 所示。图 12-58 所示的例子中 E 的原始值从 0 到 4095。原始的 E 值（例如，976）被模糊化为超过一个模糊值（例如 976 在 MN 区域有一个隶属度 0.85，在 LN 区域有一个隶属度 0.23）。

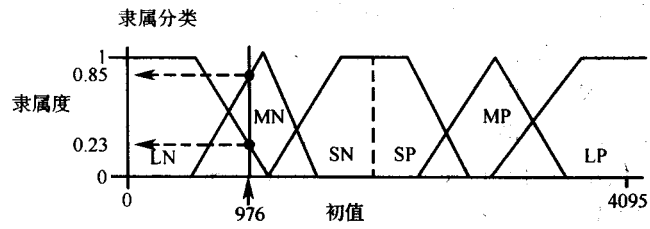


图 12-58 误差（E）的隶属函数表

程序员的一个任务是建立两个隶属函数表。编程软件可以以图 12-58 所示的图表形式显示隶属函数表。

1）隶属函数一般有三到七个分区，其中中央分区（小误差值）被分为两半（SN 为正误差，SP 为负误差）。

2）每一个分区包括一个容易记忆的区域名：L、M、S 代表大、中、小；N 和 P 代表正和负。

3）隶属函数分区的形状一般是的三角形，但也可以是梯形，如本例所示。程序员在建立隶属函数表时输入原始值作为三角形以及梯形的平顶的端点。

4）分区通常是重叠的，每个原始值可以得到两个或更多的模糊值。如果不重叠，每一个原始值只能离散化为一个模糊值。例如“在 MN 分区中的 0.85”这样的模糊值，可以解释为两个原始值（靠近 LN 分类和靠近 SN 分类），所以需要其他的模糊结果来帮助确定真实的输入值。

近似推理 程序员必须输入一系列规则，对于每一个 E 和 CE 的模糊值的组合都必须有一条规则。编程软件通常会生成一个表格，包含所有 E 和 CE 的隶属分区的组合。一个完成的表格如图 12-59 所示。对每一行（一条规则），程序员必须为输出要求的强度输入一个模糊分区名。一些规则是简单的，规则 1 是：如果误差是大的正值（LN），但快速衰减（CE＝LN），则要求一个中等的正输出（MP）（维持误差的衰减）；其他的规则更小心；规则 5 是：如果有中等的负误差但误差在快速减小，则停止输出（并让转矩使它回到设定值）。

规则	E	CE	O
1	LN	LN	MP
2	LN	SN	LP
3	LN	SP	LP
4	LN	LP	LP
5	MN	LN	Z（Z 意味着零）
6	MN	SN	MP
7	MN	SP	LP
8	MN	LP	LP
9	SN	LN	LN
10	SN	SN	Z
11	SN	SP	SN
etc.			

图 12-59 近似推理的规则表

在近似推理过程中，控制器会结合各个要用到的规则，然后为每一个输出隶属函数分区决定一个隶属度。指定输出隶属度的通常方法是指定它等于在 E 或 CE 分区中的更小的隶属度。例如，如果 E 是 LN，隶属度为 0.95，CE 是 LN，隶属度为 0.45，使用规则 1，则输出应该是 MP，隶属度为 0.45。平均 E 和 CE 的隶属度是另一个计算输出隶属度的方法。

既然 E 和 CE 的隶属函数表包含分区的重叠，近似推理将会应用超过一条的规则。拥有超过一条的模糊输出的结果需要以下步骤：去模糊化。

去模糊化 模糊输出分区和隶属度的值必须转化为原始值，这样可以送到模拟输出模块。一个程序员建立的输出隶属函数表如图 12-60 所示。注意在这个例子中，建立了一个最小和最大原始值都在 2048 的中央分区 Z。这个分区的形状是一条竖线。隶属函数分区不能有平顶，因为对每一个模糊输出值只能一个原始输出值。分区的重叠是防止不确定性的关键因素。在图 12-60 中，近似推理产生两个模糊输出值，SP 输出区域的 0.56 和 SM 输出区域的 0.49，注意对每一个模糊逻辑值都代表两个可能的原始值。在 SP 区域，与 0.56 对应有两个原始值：一个接近 Z，另一个接近 MP。另一个规则产生在 SN 区域生成模糊逻辑结果，

告诉控制器选择 SP 区域的哪一个结果。在一些去模糊化例行程序中，有最大隶属度的模糊逻辑结果被用来决定原始值（2155）；在另一些去模糊化例行程序中，使用公式来计算模糊逻辑结果的加权平均。

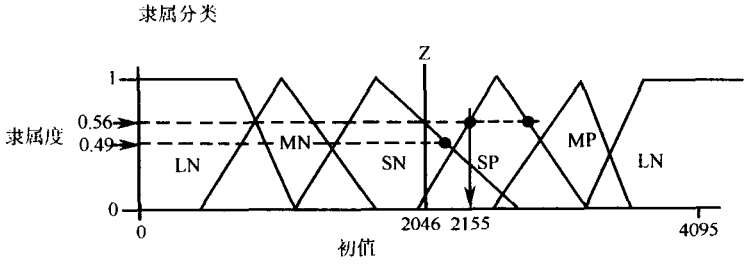


图 12-60 输出的隶属函数表

在决定原始输出值后，去模糊化可以用一个比例系数来乘当前的原始输出。有时以下方程也可以用来累加（积分）所有的历史原始输出值，将累加结果乘以积分增益，然后将这两项相加：

$$\text{输出}=G_p O+G_i \int (O)\,\mathrm{d}t$$

PID 控制也可以在模糊逻辑中使用。比例增益表示为 E 分区和输出的最终计算中的隶属度，积分控制部分表现在原始输出值的累加和中，微分控制部分为 CE 值。

12.4.7 系统的手动控制

即使是最完美的控制系统有时也需要由操作员来控制。例如，在放弃使用只有积分控制的程序前，有必要手动控制液罐的加热器来达到指定温度。假如被控系统的某个部分失灵或出错，操作员就不得不对被控对象进行控制。在这些情况下，操作员需要切换手动/自动模式，同时还需要表盘或计算机终端来提供控制变量值。

预编的 PID 指令通常都能够进行手动配置。当操作员将手动/自动开关拨到手动位置时，PLC 程序只需要改变 PID 指令的配置即可。如果 PLC 程序员已经编写了自己的控制程序，那么用另一个值代替 PID 计算结果来作为控制变量就很容易了。我们可以编写 PLC 程序令其能够根据手动/自动模式开关的位置来选择不同的子程序。如果手动位为关闭，那么调用的子程序进行过程控制计算；如果手动位为打开，那么调用的子程序读取一个人控模拟输入量作为输出值。

安全起见，我们可以配置 PLC 使其在停止模式下，能够自动将对被控过程的控制转到手动控制器。在 PLC 程序中使用无条件输出指令和 PLC 控制的继电器开关可以实现这个目的。图 12-61 给出了我们所需的指令，同时也画出了一个继电器，当 PLC 运行时，它与 PLC 的模拟输出模块相连；当 PLC 停止运行时，它与手控表盘相连。

有时，当开关切换为手动位时，手动/自动选择开关的连线会绕过 PLC。但这不在本书的讨论范围内。

12.4.8 根据检测到的情况的不同而选用不同的计算控制输出方法

我们可以设计控制系统能够自己选择控制程序或调整控制参数，我们称这样的控制系统

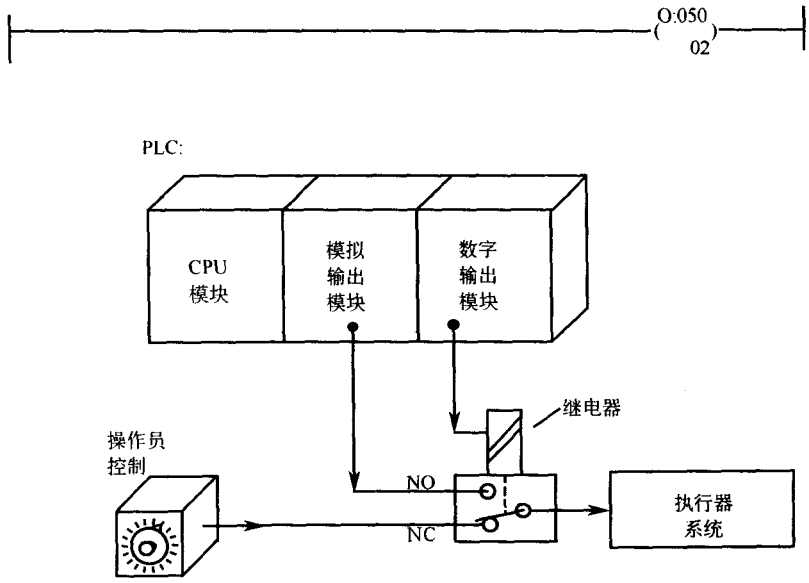


图 12-61 当 PLC 停止时，将过程的控制权交给操作员

为约束控制系统。约束控制有时是生产中需要的。例如，正常情况下控制液罐的温度，和在一天的开始时要将液罐的温度提升到操作温度，这两种情况就需要两个不同的程序来完成，那么在不同的情况下就需要进行选择。

一种改变控制系统行为的方法是在控制程序工作时改变增益参数，一个温控程序需要有监控设定值与过程变量值之间误差的指令。当误差较大时，程序选用较大的比例增益，但不用积分增益，这样控制系统会尽力去消除误差但不会出现大的积分项；误差变小后，程序将比例增益降为 0，同时启用大的积分增益，这时控制系统就会尽力去保持液罐温度，虽然周围环境的温度在逐渐变化。

有时需要将 PID 指令正使用的整套参数都替换掉。PID 程序会将所有的配置参数和工作数据都存储到一套数据字、控制块或数据块中。当你要调用 PID 程序时，需指明 PID 指令的所有数据在哪里。当被控过程所处的环境或外界条件改变时，程序员可以执行一个新的 PID 指令调用（即 PID 指令配置了新的一套数据参数）。例如，一个控制液罐温度的用户程序，要提升液罐温度时可调用一套配置数据，而保持温度时调用另一套配置数据。在有间接寻址的 PLC 中，程序可能只包含一个 PLC 程序调用，但在不同的环境下，我们可以通过不同的地址间接寻址到不同的一套数据。切记 PID 的每套数据中都会存储停止调用时计算出的积分值、微分值和平滑微分值。如果程序在经过一段时间后又重新使用某套数据，这套数据中已有的计算值可能会导致不可预测的结果。如果你的程序出现这样的几套数据之间切换的情况，那么请务必在重新启用一套数据时将其中的已有的计算值清除。

如图 12-62 所示为一个 Allen-Bradley 的 PLC-5 程序。N7：0 作为一个指针，其中存储着整型文件 N10 和 N11 的地址[⊖]。由位（B3/2）来决定 N7：0 指向谁，即选择哪个控制块。图 12-62 包括一个一次翻转指令来检测 N7：0 中的变化，并再次执行 PID 指令，PID 指

⊖ 间接寻址不能用在 Allen-Bradley PID 控制文件的寻址。

令中的 A/M 选择位（控制块第一个字的位 1）被清除以选择手动模式，然后立即复位该位。PID 指令每个周期都会被执行。切换到手动模式，然后再切换回自动模式。这样，原有的积分值和微分值就会清除。

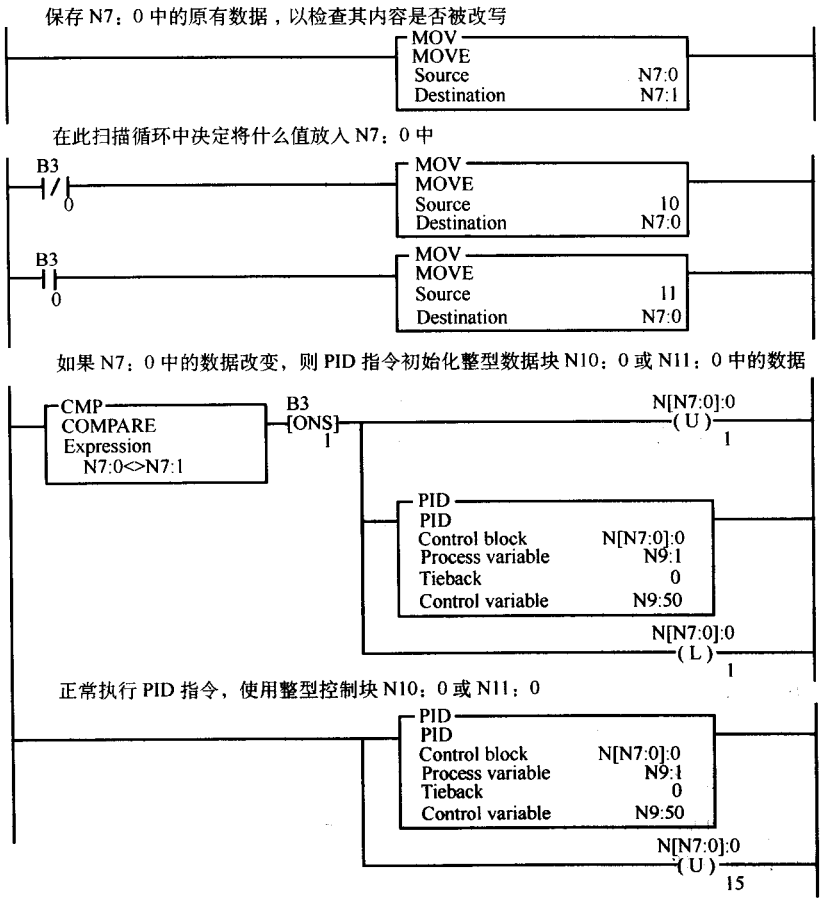


图 12-62 用 PLC-5 的间接寻址方式来为 PID 指令选择不同的控制块

图 12-63 给出了 SIEMENS STEP 5 STL 程序的一部分，它能够自动选择使用数据块 1 或数据块 2。其中，数据块 1 用于保持液罐的温度（它含有一个大积分增益）；数据块 2 用于提升液罐的温度（它含有一个大的比例增益）。程序根据设定值与过程变量之间的差的大小来选择一个合适的数据块，但这个例行程序中没有包括清除停用一段时间后又重新启用的数据块中已有数据的指令。

```
      C     DB1          ;正常操作的最佳参数
      L     FW2          ;装载 OB251的最近计算得到的误差值
      L     KF    + 100  ; 并装载一个小的正值(100)
      > = F              ; 检查看是否系统在正常
                         ; 操作范围(小的正误差)
      JU    = M1         ; 并跳到调用 PID 例行程序的地方
      C     DB2          ;否则,改变到启动参数
M1    JU    OB251        ;然后跳到 PID 例行程序
      L     DW38         ;保存这里计算得到的错误
      T     FW2          ;执行以选择下一个 DB
```

图12-63 用 STL 语言编写的自动选择操作特性的程序

12.5 故障检修

过程控制计算需要仔细的整定，而且容易失调。当系统开始出错时，请务必先检查被控系统。看是否有损坏或老化的部件使用一些合适的测量工具，如振荡分析工具可以在移动零件的系统对制造组件精确定位。使用电信号监视工具（如电压表、安培表、示波器等）可以检查出 PLC 外部的电气设备是否工作正常。

如果你确信问题出在 PLC 程序中，请在修改参数前将其备份。然后寻找下面的问题：

1. 如果被控系统始终向一个方向驱动，那么：

1）检查过程控制计算中使用的设定值是否正确。

2）检查反馈量是否在变化，是否被过程控制计算使用。

3）确保控制方向的正确。考虑应当使用哪种误差计算方法，普通方法（设定值-PV）还是反转方法（PV-设定值）。

4）检查数值的标度变换是否正确。是否需要添加一个偏离量（或应去掉已有的偏离量）。

5）检查过程控制计算的结果是否被使用，是否控制变量的值来自别的地方。

6）输出限幅是否正确？系统是否有其他方式驱动？

7）计算误差在调试时是否不断累积？如果使用的是预编的 PID 指令，将其切换到手动模式（至少一个执行间隔），然后在回到自动模式下，这样可以消除累积的误差。

2. 如果系统在消除误差时，总是产生超调现象，那么：

1）确保过程控制计算是按照给定的时间间隔执行的，而且频率足够大到控制被控过程。监视计算中使用的变量，是否实际误差都已消除而计算的误差仍不为 0？详见 PID 控制整定一节。检查过程控制计算是否按固定的时间间隔被调用。编写一个计时器，在计算的执行间隔期间计时，而且在每次过程控制计算执行时存储一次计时器的累计值。如果过程比较慢，我们可以通过计时器的计时来检查计算的执行间隔是否不变；但如果过程太快，我们需要编写一个计数器来计算时间值超过可接受范围的次数。

> 切记：如果你对 PID 指令的执行间隔作了改动，那么通常情况下你还需要对指令使用的数据中的回路更新时间作相应的修改，使指令能够正确的计算积分项和微分项。如果指令中无回路更新参数，那么就需要根据你所修改的执行间隔来重新计算积分系数和微分系数。如果使用了定时中断，那么回路更新时间要与定时中断间隔相匹配。

2）过程变量是否在过程控制计算之前立即更新，还是它在要进行计算时已经陈旧？详见关于降低扫描时间延时的有关章节。

3）刚计算得到的控制变量是否在第一时间被写到输出模块？详见关于降低扫描时间延时的有关章节。

4）被控过程在向不同的方向驱动时是否需要配置不同的整定？（例如，一个液压气缸，其中的活塞向前推进 1 英寸所需的能量要大于其退后 1 英寸所需的能量，因为活塞前方的油室内有一个活塞杆。）

5）是否在响应设定值变化时误差的积分过大而导致超调？可以试着加大比例增益，而降低输出的限幅。大部分的 PID 指令都有 antiwindup 特性。有的 PID 指令还有死区特性，其作用是当过程变量接近设定值时防止被控过程滑过设定值。

6）过程控制计算中的微分项是否按你所希望的变量进行计算？有的方程计算误差的变化率，而有的则计算过程变量的变化率。有的方程可以让你选择被微分量。

7）微分项的作用太小是否无法阻止超调的发生？或者是太大而导致系统无法跟上设定值？详见调制 PID 整定的有关章节。

8）死区时间是否太长？如果是的话，可以试着降低被控系统的延时。可以考虑使用前馈量来预测所需的更正。

9）可以考虑增加能量损耗（如摩擦力）来避免超调的发生。

3. 如果系统在校正误差时，达不到设定值就停止了，那么：

1）增大增益，尤其是比例增益和积分增益。详见 PID 计算整定的有关章节。

2）降低被控系统的能量损耗（如摩擦力）。

4. 如果设定值或过程变量改变后，系统未作出任何反应，那么：

1）监视 PID 方程所使用的设定值和过程变量。如果它们有变化，那么就要核实在 PID 指令执行前（而不是后）这些量确实被读取了。

2）检查过程控制计算得到的控制变量是否有变化，是否是在计算后（而不是前）被写到了输出模块。

3）查看一下存储输出量的地址单元是否被你的程序中的其他指令所使用。

5. 如果程序在改变计算参数后或换用其他控制计算后，系统突然无法正确工作，那么：

检查你的 PLC 是否需要重新启动 PID 指令（调为手动模式然后再调回自动模式），这样新的参数才能生效；是否是修改了自动模式下被控过程的参数导致了误差的累积？这些潜在的问题一般不会在书或手册中提到，也许是因为编者自己对过程控制计算还未完全理解。可以试着关闭指令的自动模式，然后再打开，如果这样问题可以解决，那么就在你的程序中添加上每当改变参数时就进行模式切换的有关指令。

习题

1. 定义设定值（命令变量）、反馈量（过程变量）和控制变量。以汽车的速度控制系统（或导航控制系统）为例加以说明。
2. 什么是工程单位？试以汽车的导航控制系统为例加以解释。
3. 试解释在汽车的导航控制系统中为何需要对输出值进行限定，如何限定？
4. 早期的导航控制系统允许汽车下坡速度明显快于上坡时的速度，为什么？汽车中用来检测前或后倾角的传感器可用来提供系统校正的偏离量（偏移量、扰动量或前馈量），试解释其道理。
5. 早期的导航控制系统在汽车上坡时其稳态误差会变的更大，为什么？PID 控制系统中的哪一项可以消除本例中的稳态误差，如何消除？
6. 当汽车抵达坡顶时，早期的导航控制系统会出现超调现象（尤其时 PID 控制已经消除了稳态误差），为什么？PID 控制系统中的哪一项可以消除本例中的超调误差，如何消除？
7. 试解释独立增益 PID 方程和相关增益 PID 方程的区别（即什么是独立增益 PID 方程和相关增益 PID 方程）？
8. 什么是死区？为什么在如导航控制系统这样的控制系统中需要死区特性？
9. PID 控制中的 antiwindup 特性有什么用处？
10. PID 控制中的 antijerk 特性的作用是什么？

11. 如果我们将 PID 方程执行的时间间隔扩大为原来的 2 倍，而 PID 方程无法自动作出补偿（多数都不能），那么积分项和微分项会受到什么影响？
12. 为汽车导航系统建立建立模糊逻辑的近似推理表。假设一个简单的系统，仅有：
    - 速度误差仅有三个隶属区域：ZE（零误差），TF（太快，隶属度随误差的增加而增加），TS（太慢）。画一个隶属函数表。
    - 加速有三个隶属函数：ZA（零加速），P（正＝加速），N（负＝减速）。
13. 如果汽车的导航控制系统有一个约束控制选择开关，可以选择感知型（适用于老年人）或娱乐型（适用于青少年）两种控制场景，那么用于感知型的 PID 控制算法如何与娱乐型的区别？假设系统是由你所学的 PLC 加以控制的。

推荐的 PLC 实验室练习

系统需配备：

- 四个控制面板开关：输入 0 到 3
- 一个位置控制执行器（例如，由模拟信号控制速度的电动机或伺服流控执行器）
- 一个检测负载移动位置的传感器（例如，电位器或带有脉冲计数模块的编码器）

模拟过程的开-关控制

位置控制系统，其组成为：一个根据接受到的模拟信号而移动的执行器和一个线性位置传感器或编码位置传感器：

步骤 1：编写程序令 PLC 读和写模拟数据（或从译码输出中读取数据）。编写一个 MCP 程序连续的读和写每个模拟 I/O 模块的数据。

步骤 2：编写程序实现：当操作员按下开始按钮后，执行器重复的移动，直至停止按钮被按下。

1）确定位置 1 和位置 2 所需要的位置设定值。我们可以通过用手移动一个线性位置传感器来测出我们需要的数据。

2）编写 PLC 程序，使其能够：

- 当 PLC 进入运行模式时，将执行器的速度初始化为 0。
- 当瞬时触摸开关（开始开关）按下时，将动作从头开始；当瞬时触摸停止开关按下后，停止一切动作。
- 将速度设定值从一个数据文件（这样下次修改它们时不用重新编辑程序）拷贝到模拟输出模块，然后完成下列动作：
- 执行器向位置 1 缩回。
- 到达位置 1 时，执行器静止 2s。
- 向位置 2 驶去，到达后也静止 2s。
- 重复步骤 1。

3）选取三组不同的速度设定值进行试验，观察并记录负载的实际速度和停止时的位置误差。每组数据至少重复试验 5 次（注意：每次试验，误差是多少？是否恒定？速度和方向对试验有无影响？）

步骤 3：使用定时中断使被控过程有更大的确定性，达到基本的控制要求。

1）将读取传感器数据和写执行器的指令拷贝到一个新的程序中。改变处理器的配置使这个

新程序文件能够按照给定的时间间隔执行。

2）至少用 5 种不同的时间间隔进行试验，观察误差的变化（误差的大小或持续性是否受时间间隔影响?）。

使用 PID 指令进行比例控制

步骤 4：修改先前的程序，加入 PID 指令。（根据所选用的 PLC 的不同，下面的步骤顺序可能会有所改变。）

1）在定时中断程序中加入 PID 指令。从模拟输入模块读取过程变量（测得的位置信号），并将计算出的控制变量（速度信号）送到模拟输出模块。

2）将配置数据输入到数据文件或 PID 指令使用的变量中，如果有数据画面提供，可以使用。注意：PLC 在运行模式时，过程变量、控制变量和一些其他的 PID 变量会在数据画面中显示。

3）修改 MCP 程序，将位置设定值送入到 PID 指令的数据块的有关数据字中，而不再是直接送到模拟输出模块。作者建议将数据存储器中的设定值备份，这样修改起来会比较容易。

4）修改主程序，在位置传感器检测到负载基本已到达指定位置时认为本动作已经完成（然后等待 2s，再开始下一个动作）。提示：你的程序应该从数据存储区域读取一个非常接近的数值，这样就可以很轻易的调整了。

5）在每两个定时中断间隔之间（可能会需要调整回路更新时间），试验用两个不同的比例系数。对于每一个，选取并记录下不会导致系统动作卡壳的最小接近值。对于每一次设置，记录并观察最少 5 个周期的在停止位置的实际误差。这样的系统与开关控制器相比，哪个更精准？哪个更持久？

步骤 5：引入工程单位和死区。

1）选择一个工程单位来测量执行器的运动范围。

2）修改 PID 指令的标度变换系数，使我们能够直接输入带工程单位的设定值。

3）加入死区值，这个值最少与步骤 4 中选择的接近值相等。不过不同的是这个值是带有单位的。

4）修改程序使它能够提供带工程单位的设定值，而且当误差进入死区值范围内时运行延时 2s 的定时器。

5）选择和记录使系统正常运行的最小死区值。要缩小死区值可能还需要调整计算中的增益值和限定值。

步骤 6：使用比例-积分（PI）控制来减小稳态误差。

1）修改程序使当新的设定值写入时，PID 指令的累积积分值被重新初始化（按下开始按钮写入第一个设定值）。

2）慢慢的增大积分系数，而不断的减小比例系数。试验看能得到多低的死区值。

3）测定现在能得到多小的死区值。

步骤 7：使用比例-微分（PD）控制实现更好的加速和减速。

1）将积分增益置为 0，然后根据需要增加死区值。选择合适的比例系数，然后记录完成一个运动周期所需要的时间。

2）逐渐增加微分项的增益或微分时间（根据你所选择的方程而定），适当降低比例系数。记录此时完成一个运动周期所需的时间。

步骤 8：使用比例-积分-微分（PID）控制，整定得到合适的系数使运动周期具有一个小的死区值，而且可以快速的执行。

第13章 通 信

13.1 学习目标

本章您将了解到：

- 现代 PLC 的通信要求；
- 共享通信通道的网络访问方法；
- Allen-Bradley、Siemens 和 OMRON 的 PLC 通信能力；
- 配置 PLC，并使用指令通过串口和局域网进行通信；
- 用户程序中的通信控制。

13.2 PLC 的通信能力

至少在 15 年以前，专家们就已经预言 PLC 将被个人计算机所代替。其中最普遍的理由是 PLC 没有足够的通信能力。本章我们讨论 PLC 通信能力的复杂水平（在 16 章我们会列举为什么说 PLC 正在“死去”的原因）。

PLC 可以在以下几种模式使用通信接口和通信软件：

1）通过串行通信连接到编程器可实现对 PLC 的编程和配置。通过接收来自编程器的程序和数据或者发送状态信息到编程器，PLC 响应编程软件的命令。

2）有些 PLC 要求在 CPU 的数据存储器和智能 I/O 模块之间传送大量的数据。这些数据必须通过本地框架连接 CPU 到 I/O 模块的并行数据总线来传输。

3）现代 PLC 允许多个远程 I/O 框架通过串行接口直接连接到主框架上的 CPU。用户程序以同样方式响应远程 I/O 模块和本地框架中的 I/O 模块：作为输入映像表数据或输出映像表数据。PLC 负责处理与远程 I/O 框架的数据交换。远程框架都配有通信模块以代替 CPU。通信模块从 PLC 框架中的输入模块读取数据然后发送给主 CPU，从主 CPU 中接收输出数据然后写入 PLC 框架的输出模块。

4）现代 PLC 可以连接到局域网，所以 PLC 能通过串行连接与其他的控制器（可能是其他的 PLC）交换数据。PLC 复制存储器中的输出数据到局域网的总线，从局域网总线复制数据到 PLC 的存储器，从而实现数据的交换。

5）随着 PLC 控制的制造工序变得越来越复杂以及被控过程越来越远离 PLC 的主框架，串行链接也越来越多地用于远程传输数据。传感器-执行器网络允许 PLC 通过共享的串行链接与单独的传感器和执行器交换数据。PLC 与单独的带有内置通信适配器的传感器和执行器交换输入和输出映像表数据，而功能更强大的现场总线标准允许多 PLC 中的任何一个或者其他连接到串行链接上的控制器在现场总线系统的任何位置读和写数据。每个 CPU 都必须执行在用户程序中的数据交换指令和响应现场总线上其他控制器发出的数据交换指令。

现代 PLC 的 CPU 模块中都配有多个微处理器。其中主微处理器执行标准程序的 PLC

扫描循环，其他微处理器负责处理 CPU 模块的 RAM 存储器和其他控制器之间的串行数据交换。有些控制器可能在 PLC 的外部（例如个人电脑，其他 PLC 等），也有些可能在 PLC 控制下的智能 I/O 模块（例如高速计数器模块、运动控制模块等）。对每个通信处理微处理器进行特定的编程实现通信，它们可以通过某些路径和方法对主微处理器的存储器进行访问，因此用户程序能够读取通信处理器的状态并对它分配新的任务。

通信处理器的设计目的是通过一组单独的导体与其他控制器进行数据交换，因此它必须共享这些导体。PLC 利用与其他计算机网络相同的网络访问控制方法：

1）在轮询或者主-从系统里，一个单独的计算机被指定为主计算机（通常是节点 0）。它负责发送请求消息到网络中的从计算机，每次只能发送到一个从计算机。从计算机（例如节点 12）响应主计算机请求，要求主计算机传送一个数据包到另一个节点（例如节点 5）或者请求主站从节点 5 接收数据然后传送回节点 12。在本书中提到的所有 PLC 都具备主-从通信能力。

OMRON 的主机链接通信协议就是基于主-从架构的，也是本书中描述最详细的主-从系统。Allen-Bradley 的 PLC 用的是 DF1 协议，而 Siemens PLC 的 L1 网络和 Profibus-DP 网络也包含了主-从架构（Profibus 工业通信网络已在世界范围内被广泛应用而并非只限于 Siemens，存在其他版本的 Profibus）。

2）在令牌传递系统里，就像轮询系统里一样，每个令牌传递网络中的计算机都轮流拥有发送数据或请求接收数据的机会，但不是作为从计算机，而是每台计算机都是平等的。在这个端对端（peer-to-peer）的系统中，一个节点可以直接发送数据包到另一个节点，每个节点都可以按指定的顺序发送数据。当计算机已经发送完需要发送的数据（或者它被分派的时间段结束），它就会发出最后一个信息，这就是令牌，而这时就轮到接收到令牌的计算机发送数据、数据请求或者对之前接收的请求的响应数据。有些令牌传递网络确实有主计算机，但只是用来检测令牌的丢失情况和启动新的令牌环。令牌传递的速度比轮询快，而且当每个节点在预先指定的环中轮流获得令牌而开始工作时能保持内在的可靠性。

20 世纪 70 年代，通用汽车公司的计算机通信采用了 ISO 的开放式系统互联（OSI）参考模型，它的 OSI 最终版本被称为生产商自动化协议（MAP）。尽管很少人购买 MAP 产品，但它却成为后来的计算机网络的模型。MAP 就是利用令牌传递架构实现网络访问的。

Allen-Bradley 的数据高速公路网络和 GE Fanuc 的 Genius 网络以及流行的 Modbus Plus 网络，都是北美企业的事实上的标准，它们全都利用令牌传递。

3）带有冲突检测的载波帧听多路访问网络的访问控制方法是办公室电脑最流行的网络访问控制方法，它被更多的称为是 CSMA/CD 或者以太网。在 CSMA/CD 网络里，只要网络空闲时任何一个节点都能发送数据到网络上。如果两个节点同时发送数据，则发生冲突，此时两个节点都会检测到冲突（在网络上的数据与它正在发送的数据不同），然后停止发送，等待一段随机时间再重新尝试发送。

现场总线基金会是一个代表大多数工业控制器厂商的组织，最近已经接纳快速以太网为将来的开放式工业控制器的通用网络访问方法。现在，在本书中所提及的 PLC 厂商都提供以太网通信模块，而下一代的 PLC 将有可能在它们的 CPU 模块中提供以太网端口。

4）实际上有些网络是上述访问控制方法的结合体。在工业控制中，监控控制器（例如 PLC）读取远程传感器数据和写数据到远程执行器的功能是很重要的。远程 I/O 相对主控制

器是从站，它们有时也被称为网络上的被动节点，相应的主控制器被称为主动节点。要使得生产设备效率更高，同样的网络必须允许连接多个主动节点并且使用令牌传递或者 CSMA/CD 架构以端对端（peer-to-peer）的方式交换数据。

OMRON 的主机链接主-从通信架构在轮询各个从站时有一小段停顿时间，主站在这时能接收从站发来的紧急服务请求。Profibus 是一个有点对点通信能力的主-从系统。Allen-Bradley 提供一个可选的 Controlnet 网络，其中有主-从和点对点通信两个可选网络。

PLC 制造商和第三方供应商提供多种带有软件的接口设备，这些设备拥有把一个网络访问架构转换到另一个架构的功能。如果这些接口只提供网络访问转换功能，它们多数被称为网桥；除此之外，如果接口还提供调整数据格式或执行数据传输控制，它们大多数被称为网关。

开放网络应该不属于任何商业组织。任何人都可以自由地生产和销售开放网络的兼容组件，所以用户不会被强迫只从一个供应商那里购买他们所有的控制组件，或者被限制只能购买供应商提供的选择。最近，已经有几个开放式的标准开发出来并被 PLC 制造商接纳，事实上这些标准已在相互的竞争中（尽管生产商同意相互协调工作，但工业上是不会停止竞争的）。在第 16 章我们将会讨论几个更成功的开放标准。

13.3　ALLEN-BRADLEY PLC 的通信

PLC-5
SLC 500

当 Allen-Bradley 的 PLC 配置好后，很容易对它们编程以实现通信功能。配置 Allen-Bradley 的 PLC 也相当容易，只是有好多选项要选择。因此，理解 Allen-Bradley 的 PLC 通信过程中会发生什么是很有用的（在以下的部分将分别对 PLC-5 和 SLC 500 进行描述）。

当用户程序执行请求由通信端口传输一个数据的指令时，它通常只是被添加到一个含有其他请求获得通信通道的通信请求的队列的尾部。在扫描循环的 I/O 扫描部分，PLC 的主处理器通过检查通信处理器是否已经完成之前的任务，或者是否准备好接受下一个新的任务，而为通信通道提供服务。如果是的话，CPU 就从通信处理器的缓存器中复制最先到达的数据到主存储器中，和/或从主存储器中复制新的输出数据到通信缓存器中。通信任务每次只能执行一个（每个通道），通常按通信处理器接收到的请求的顺序来执行。通信任务与扫描循环是异步执行的，这意味着从用户程序执行一条请求传送一个数据的指令到数据传输完成有可能需要经过多个扫描循环。因此，在一条通信请求指令的执行周期中或者某个通信状态位表明数据传输正在执行时，用户编写的程序不应该尝试检查主存储器中正在输入的数据或者改变主存储器中正在向外输出的数据。有些指令能迫使 PLC 不用等待下一个 I/O 扫描到来而中断扫描循环转去服务通信处理器，也有些情况下一条新的数据传输请求会在队列里比其他请求具有更高的优先权。

Allen-Bradley 的 PLC-5 和 SLC 500 都拥有多条通信通道，能同时响应多个通信请求。有些通信行为有足够的自动化性能，只需要很少甚至不需要任何的配置或者编程，尽管这些性能会影响 PLC 本身的性能，但程序员常常还是不清楚它的用途。还有些通信通道必须配置好，而且只能用于响应某些用户程序指令。Allen-Bradley PLC 的通信通道主要用作以下用途：

1）编程器接口。除了如第 10 章所描述的配置优先级能够限制对 PLC 存储器的访问，PLC 对编程器的命令的响应不能修改。PLC-5 能连接到个人计算机的 RS-232C 串口来编程，

但如果生产车间有多个 PLC，使用个人计算机里 Allen-Bradley 接口卡会更适合。接口卡连接到 PLC-5 CPU 面板上的一个插头，而编程器则连接到 DH＋局域网上（如图 13-1 所示）。编程器和 PLC 之间的消息由 DH＋上的其他路径传送。如果有其他节点正在使用 DH＋，程序监控速度会因此而降低；如果编程器请求大量的数据，DH＋的数据交换速度也会慢下来。在第 10 章我们已经讨论过怎样配置编程软件使它能通过 DH＋连接到 PLC 上。连接到 DH＋网络上的编程器能够连接到 DH＋网络上任何的 PLC 并监控或者改变它们存储器的内容。

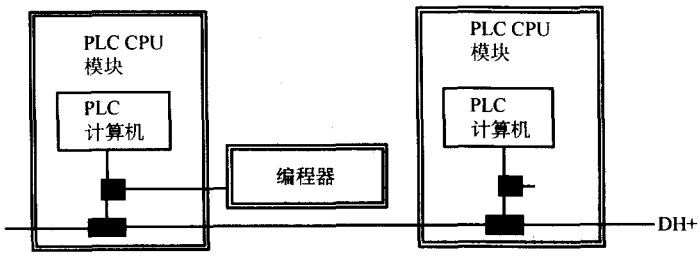


图 13-1 编程器和 Allen-Bradley PLC 在 DH＋上的连接

2）本地 I/O 扫描，主 PLC 扫描循环的一部分，它读或写数字 I/O 模块和 PLC 框架上其他模块的状态。

3）如果一个 I/O 中断被配置，在执行用户程序时，本地框架通信处理器必须周期性地读取这个输入模块的状态。在 PLC-5 中 I/O 中断被称为过程输入中断（PII），而在 SLC 500 中被称为离散输入中断（DII）和 I/O 中断。

4）立即 I/O 指令（IIN 和 IOT）在用户程序运行时，利用本地框架通信处理器在空闲总线上发送或者接收单独的数据字。

5）数据通过执行用户程序中的消息（MSG）指令在数据高速公路通道（DH＋或者 SLC 500 的 DH485）上与其他 PLC 进行交换。

6）其他串行通信。配置集成串行口作如下操作：

①系统模式，作为 DF1 通信通道，通过连接诸如调制解调器之类的链接实现与其他设备通信。通过执行消息（MSG）指令、ASCII 写（AWT）或者 ASCII 设置握手（AHL）指令，能够在 DF1 通道上实现用户程序之间的通信。使用 DF1 口，PLC-5 可以被设置为：

- 点对点模式，直接连接到另外一个设备（默认模式）。
- 从 DF1 模式，每个节点通过接口调制解调器或者专线驱动器连接到局域网，节点可多达 254 个。当主 DF1 允许从站发送时，从站才能发送报文。
- 主 DF1 模式，轮询每个在 DF1 网络上的从站，有请求时就传送请求的消息。

DF1 可容纳比 DH＋更多节点的连接，也给用户提供更多可选择的通信协议，但配置过程并不比 DH＋的容易。由于 DH＋网络能够互联从而可以容纳更多节点，而且操作容易和可靠性强，同时也不需要进行那些传统的配置，所以 DH＋在很多地方都可以代替 DF1 的使用。因此这里也不对 DF1 作详细描述，您可以阅读本章中有关 OMRON 的主机链接通信来了解一下 DF1 的典型工作情况，要了解有关细节可参考 Allen-Bradley 的操作手册。

②用户模式，作为 ASCII 通道，用于与外围设备，例如条形码读卡机或者串行打印机

等，交换 ASCII 码字符串。用户程序可以使用 ASCII 指令通过 CPU 模块的用户模式通道中的缓存进行数据的发送和接收。本书中没有涉及使用用户模式通道的 ASCII 指令知识。用户手册会提供如何使用 ASCII 指令的细节，例如：

- ASCII WRITE（AWT）或者 ASCII APEND（AWA）指令，用于向缓存器写入一个 ASCII 码字符串，然后由用户模式通道传输字符串。
- ASCII READ（ARD）或者 ASCII READ LINE（ARL）指令，用于从用户模式通道读取它接收到的 ASCII 码字符串，然后存储到缓存器中。
- ASCII CHARACTERS IN BUFFER（ACB）或者 ASCII TEST FOR LINE（ABL）指令，用于判断缓存器中有多少字符或者缓存器中第一条消息中有多少字符。
- ASCII HANDSHAKE LINE（AHL）指令，用于控制 RTS 和 DTS 的握手队列。
- 其他对 ASCII 码字符串操作的指令。

13.3.1 ALLEN-BRADLEY PLC-5 的通信

PLC-5

PLC-5 大体上用了一些与如上所述的 Allen-Bradley PLC 相同的通信通道：

1）编程器接口。个人计算机里的接口卡，通常用途是把计算机通过 PLC-5 的编程端口连接到 DH＋上。

2）本地 I/O 扫描。通过本地底盘内的总线导体每次对一组 I/O 进行读或写。凡是带有字母“L”的 PLC-5（例如 PLC -5/40L）在通道 2 都有一个并行连接器，通过这个并行连接器能对扩展的本地框架进行扫描。从你的手册中可以获得更多详情。

3）过程输入中断（PII），一旦配置好，只要运行用户程序，就使得本地底盘通信处理器周期性地读取某个特定的输入模块。

4）立即 I/O 指令（IIN 和 IOT）寻址本地底盘里的数字 I/O 模块，当用户程序运行时，此指令命令本地底盘通信处理器在底盘总线上发送或者接收单独的数据字。

5）DH＋通道。默认的 DH＋通道是通道 1A，但默认通道可以改变。

6）集成串行口。其中通道 0 是默认的串口通道，串行口能配置为使用 RS-232C、RS-423 或者 RS-422A。

PLC-5 还有附加的通信通道的应用，包括块传输指令、远程 I/O 和以太网通信等。

1. 块传输指令

本地底盘内的 BT 模块的块传输指令是对本地底盘通信处理器的通信请求指令。块传输读和写指令，要求通信处理器在 I/O 总线的空闲时间里与 I/O 模块交换块数据（在第 7 章可查阅 BTR 和 BTW 指令）。每个块传输请求都会生成一个简短的块传输确认数据置于通信处理器等待队列的尾部。这个确认数据包含 CPU 中的源数据或目的数据的数据文件地址。按数据传输请求命令进入等待队列的先后次序，每次执行一个数据传输（除了在 PII 或者 STI 程序里，详见以下），所以块传输完成前可能需要花上多个扫描循环。每个块传输指令在存储器里都保存一个块传输控制结构体，通信处理器运行时写这个存储器区域的状态标志位。用户程序通过检查状态标志位，判断数据块传输请求是否还在等待队列中等待（.EW 状态位），已经开始传输数据（.ST），传输完成（.DN），还是工作在不正常状态（.ER 是错误标志位）。

当执行块传输时，通信处理器每次在 CPU 的数据文件和 BT 模块之间传送一个数据字。因此在整个数据块被全部发送出去之前，不应该改变输出数据，否则只能发送部分改变后的

数据；在整个数据块全部接收完成前，不应该读取输入数据，否则程序只能读到部分更新数据！当块传输进程被挂起时，程序应该写监控块传输的控制状态位（例如 .EW，.ST 和 .DN 位）以保证数据段不被更改或者使用。

正如第 11 章提到的，如果在中断程序中（PII 或者 STI）执行 BTR 或者 BTW 指令，这个块传输请求会放在队列的其他块传输请求之前，并且立即执行。中断程序中的下一条指令必须等到块传输完成后才被执行，所以不存在中断程序使用部分块传输的数据的危险。

块传输会与过程输入中断（PII）发生冲突。当本地底盘正在执行一个块传输时，通信处理器不能读取输入模块，所以本该引起 PII 中断的信号改变可能没有被检测到。Allen-Bradley 建议如果要使用 PII 中断就禁止使用本地底盘的 BT 模块，但是两者也可以一起使用，只要在用户程序里在运行 BTR 或者 BTW 之前禁止 PII 中断［使用用户中断禁止（UID）指令］，然后等待块传输完成后重新打开中断［使用用户中断使能（UIE）指令］。

块传输也会与立即 I/O 指令发生冲突。正在执行的块传输优先级高于立即 I/O。如果在程序里同时包含有本地框架的块传输和立即 I/O 指令的话，立即 I/O 执行的前提条件是没有正在执行的块传输，可以通过检查 .ST（开始）位的状态来判断。

2. 远程 I/O

远程 I/O（RIO）框架由 PLC-5 中称为扫描器的通信处理器进行读和写。通过在 RIO 控制器中设置 DIP 开关，每个 RIO 设备都被分配输入映像和/或者输出映像地址，而用户程序以同等方式处理数字 RIO 设备和本地数字 I/O 设备。在第 10 章我们已经讲述了如何配置一个通信通道为扫描通道，默认的扫描通道是通道 1B。

PLC-5 的 CPU 模块在扫描循环的 I/O 扫描期间不能扫描远程 I/O 模块。取而代之的是，PLC-5 在每个扫描循环都读和写它自身的远程 I/O（RIO）缓存器，如图 13-2 所示。扫描器的通信处理器执行它自己的扫描循环，在此期间它通过为 RIO 扫描保留的串口，在 RIO 缓存器和远程 I/O（RIO）控制器之间复制数据。一般情况下，扫描器循环比 PLC 的主扫描循环花的时间更长。而在 RIO 框架或者其他设备，还有另一个扫描循环在执行，在那期间 RIO 控制器负责从扫描器写输出映像数据到输出模块和读传送给扫描器的输入模块数据，下一次扫描器便从此模块读取数据。尽管 RIO 控制器的扫描循环一般比扫描器循环要快，但它还是增加了额外的时延。

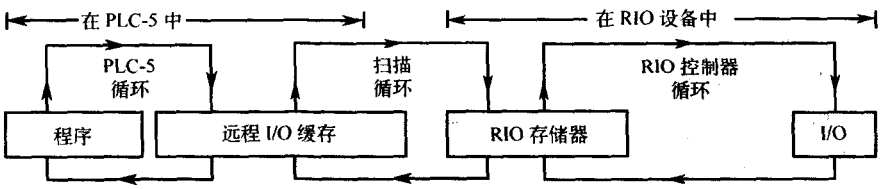


图 13-2 与 PLC-5 远程 I/O 有关的三个扫描循环

这些累加延时意味着输出映像存储器的值真正地写入远程输出模块可能需要经过几个 PLC-5 扫描循环。远程输入的读取也有类似的延时。实际上，如果 PLC-5 的一个输出映像位打开后关得太快，远程输出触点可能来不及改变，同时如果远程输入的触点过快地改变状态，PLC-5 也可能永远检测不到那些变化。

因为到/从远程 I/O 框架里的 BT 模块的块传输指令会触发 PLC-5 中断 RIO 扫描而去执

行块传输，所以远程数字 I/O 的状态更新变慢，单个块传输就很容易加倍 RIO 的扫描时间，因此 PLC 在每个远程框架的每个扫描器循环里只执行一次块传输，以此来限制扫描延时。

正如大家所想，中断程序（STI 和 PII）里到远程 I/O 系统的 BT 模块的块传输请求优先于通道等待队列中所有其他的块传输请求。另外，无论一次 RIO 扫描器循环中提出多少个在 STI 或者 PII 程序中的块传输请求，所有请求都会被执行。在增强型的 PLC-5 模块中，被中断而执行 STI 或 PII 的 MCP 在中断程序等待一个块传输完成的同时，继续执行原程序。

3. 以太网通信

以太网通信可能通过那些名字中带有字母“E”（例如 PLC-5/40E）的 CPU 模块的通道 2，如果 CPU 模块的通道 3 端口插有一个以太网控制协处理器，也有可能通过通道 3。（为实现以太网通信而设的控制协处理器的配置在第 10 章里没有提及，是因为不需要任何配置就能使用）。

1. PLC-5 的通信配置

CPU 配置中只有一个通信设置，这就是通信时间片。

PLC-5 系统在扫描循环 I/O 部分的内务处理步骤期间实现串行通信服务功能。每个扫描循环里所需的内务处理时间可能是不同的，所以如果通信通道的服务时间不加以控制的话，那么每个扫描时间也会因此有所不同。

如果通信时间片设为“0”，PLC-5 会执行每个扫描的所有的通信任务请求，所以每次扫描时间都是可变的。如果将它设为其他非零值，无论实际上有多少个通信任务的请求，PLC-5 都会把这个数字当作单位为微秒的时间，加上内务处理通常需要的基本时间 3.1 毫秒，作为每次扫描循环的通信服务时间。

使用图 13-3 所示的通道概况窗口，可以改变通信通道的几个配置。这个窗口显示了每个通道的当前配置的通信任务，允许程序员选择一个通道改变当前配置，或者监控它的通信

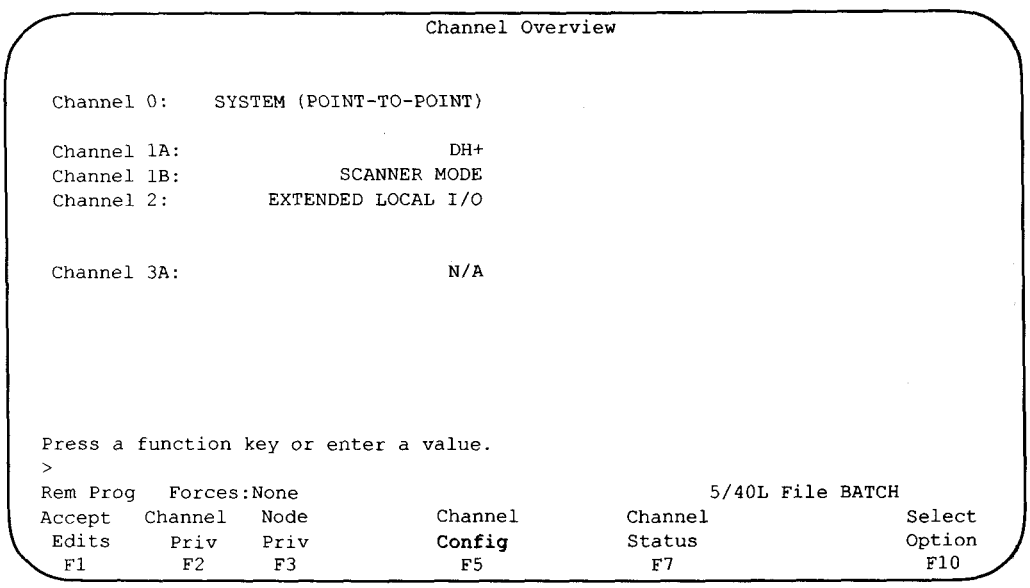


图 13-3　PLC-5 通信通道概况窗口

状态。通道能被配置为 DH＋、远程 I/O 扫描、远程 I/O 适配器模式或者其他的串行口协议。在第 10 章里有更详细的配置信息。

2. PLC-5 的消息指令

消息（MSG）指令用来通过配置成 DH＋或者 DF1 通道的 CPU 端口发送数据到目的 PLC 或者从目的 PLC 中读取数据。如果与远程网络有一个通信链接，那么消息甚至能够发送到远程网络的 PLC 上。DH＋网络之间能通过 PLC-5 连接。其他网络之间能通过 SLC 5/04 或者其他通信处理器连接。

PLC-5 的消息（MSG）指令能在梯形图的梯级上编程，这看起来似乎很简单（参考图 13-4）。MSG 指令的梯形图符号让它看上去好像输入 MSG 指令只需要输入一个信息：控制块的地址。对增强型 PLC-5，程序员应该输入一个惟一的消息控制元素地址（例如 MG10：1），但对经典的 PLC-5 处理器就输入完整的地址，这对某些增强型的 PLC-5 也适用。消息控制元素包含 56 个数据字（整型控制块的大小不同）。当程序员输入控制块地址时，就会弹出 MSG 数据输入窗口。在 MSG 指令输入用户程序前，程序员必须在这个窗口上输入另外的 MSG 参数。当光标指向 MSG 指令时，通过调用数据表可以访问另一个状态信息窗口，更多的 MSG 参数能在这个窗口上输入。在图 13-5 中显示了消息控制元素的数据输入窗口和数据表监控器窗口（整型控制块的窗口有轻微的差别）。

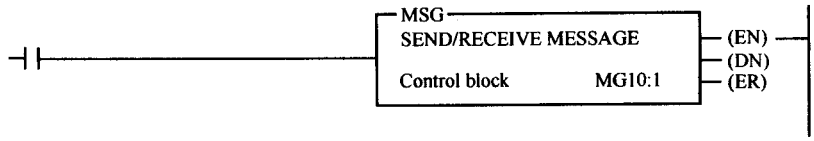


图 13-4 梯形图梯级上的一个 PLC 消息指令

```
MESSAGE INSTRUCTION DATA ENTRY FOR CONTROL BLOCK MG10:10

          Communication Command              PLC-5 Typed Write
          PLC-5 Data Table Address:          N10:30
          Size in Elements:                  1
          Local/Remote:                      LOCAL
             Remote Station:                 N/A
             Link ID:                        N/A
             Remote Link Type:               N/A
          Local Node Address:                00
          Destination Data Table Address:    N20:50
          Port Number                         1A

BLOCK SIZE = 56 WORDS
```

a)

图 13-5 PLC-5 消息控制元素的数据输入窗口和数据监控窗口

```
    MESSAGE INSTRUCTION DATA MONITOR FOR CONTROL BLOCK MG10:10

  Communication Command            PLC-5 Typed Write
  PLC-5 Data Table Address:        N7:0              ignore if timed-out:  0 TO
  Size in Elements:                1                       to be retried:  0 NR
  Local/Remote:                    LOCAL              awaiting execution:  0 EW
     Remote Station:               N/A                         continuous:  0 CO
     Link ID:                      N/A                              error:  0 ER
     Remote Link Type:             N/A                       message done:  0 DN
  Local Node Address:              00               message transmitting:  0 ST
  Destination Data Table Address:  N7:10                 message enabled:  0 EN
  Port Number                      1A

  ERROR CODE: 0



Press a function key to change a value.
MG10:10.TO=
Program    Forces:None      Data:Decimal    Addr:Decimal PLC-5/40 File DRILL1
                Toggle       Specify          Next     Prev   Next       Prev
                 Bit         Address          File     File   Element   Element
                  F3           F5              F7       F8      F9        F10
```

b)

图 13-5　（续）

必须输入或者从数据输入窗口中选择的参数包括：

（1）通信命令

从以下选项中选择一个：

1）PLC-5 TYPED READ 或者 PLC-5 TYPED WRITE，用于从其他 PLC-5 中读数据或者写数据到其他 PLC-5。如果被复制的数据是结构体数据，整个数据结构体就作为数据元素被复制（例如浮点数数字，计数器单元，数据串等）。

2）PLC-5 TYPED READ FROM SLC 或者 PLC-5 TYPED WRITE TO SLC，用于在 PLC-5 和 SLC-500 之间复制整个数据结构体。

3）SLC TYPED LOGICAL READ 或者 SLC TYPED LOGICAL WRITE，用于在 PLC-5 的存储器和 SLC 500 之间以单独的 16 位数据字形式复制数据（结构体也能被复制，但例如一个浮点数结构体，就会当作两个 16 位的数据字）。

4）PLC-3 WORD RANGE READ 或者 PLC-3 WORD RANGE WRITE，在 PLC-5 和 PLC-3 的存储器之间复制 16 位数据字。

5）PLC-2 UNPROTECTED READ 或者 PLC-2 UNROTECTED WRITE，用于一个 PLC-5 中的 PLC-2 兼容文件和 PLC-2 的存储器或者一个工作方式与 PLC-2 类似的控制器之间复制 16 位的数据字。在 SLC 5/03 以下并带有 OS 301 的 SLC 500 的工作方式都设计成类似 PLC-2 设备。

（2）PLC-5 数据表地址

输入起始地址，在源 PLC-5 中，从这个地址开始的数据被写到目的 PLC 中，或者从目的 PLC 中的这个地址开始复制数据。

（3）元素大小

输入一个数字表明有多少个元素要复制。PLC-5 能处理长至 1000 字的消息，但 SLC 500 只能处理最多 120 字的消息。如果通信命令允许结构体的话，要记住结构体是由多字组成的，所以一次只能传输较少的几个元素（除了 ASCII 元素，每个 ASCII 元素只占半个字）。

（4）本地/远程

1）如果选择本地，消息只能发送到同一个（DH＋）网络上的另一个 PLC，并且对本地节点地址：输入目的 PLC 的 DH＋地址。

2）如果选择远程，最多有四个附加的输入，充分地描述数据如何路由到远程网络的另一个 PLC 上，详细内容见操作手册（找一个好的专业建议者）。

（5）目的表地址

在源以外的其他 PLC 中输入一个起始地址，数据将被写入或者被复制到从这地址开始的存储单元。在某些含有多个命令型选项和类型的其他 PLC 里，你可能要输入引号或者其他格式（详细内容见操作手册）。如果目的 PLC 的工作方式类似 PLC-2，输入一个偏移量（10 或者更高的值）表明数据将会被写入到目的 PLC 存储器的哪个部分，就好像这个存储器是在 PLC-2 里一样。在 SLC 500 中，文件 9 通常用作 PLC-2 的兼容性存储空间，它被称为 485CIF 文件。

（6）端口号

可以改变端口号来选择 DH＋端口。除非你更改了默认的配置，否则 DH＋的端口就是端口 1A。如果你选择了一个以太网口，你将要输入一个其他 PLC 的 IP 地址来代替输入本地或远程地址特性。

在 MSG 指令数据监控窗口上大多数的信息都是状态信息，但还有两个消息控制元素可用于配置：

1）连续操作位（.CO）。被置位时，只要 MSG 指令的控制逻辑保持为真，就会使 MSG 指令在每次执行完一个操作后再重复执行一次。

2）时间溢出位（.TO）。被置位时，若 PLC-5 未能成功获取消息，在 30 到 60 秒后它将放弃尝试获取消息，然后置位错误位（.ER），写入一个错误代码到消息控制元素（.ERR），并在数据监控窗口中显示。

状态位在用户程序里用控制元素地址和这个位的助记词（例如 MG 10:1.ER），并且有些位可以通过用户程序写入（例如 MG：1 TO），如图 13-5 所示的数据窗口明确指出了各个状态位的功能。你们大概也已经可以认得大多数状态位的功能，知道哪个消息是由串口传输，以及通信处理器与主 PLC 扫描循环进行异步的串行数据交换。

SLC 500 13.3.2 ALLEN-BRADLEY SLC 500 的通信

某些 SLC 500 的通信通道有如前所述的 Allen-Bradley PLC 的那些用法：

1）编程器的端口。在编程器的串口与单一的 SLC 500 之间的 RS-232 通信是很常见的，因为 SLC 500 通常用于小型应用。你也可以用个人计算机编程器的一个接口卡，通过 SLC 5/04 的可编程端口联接到 DH＋上，或者通过其他 SLC 500 CPU 编程端口联接到 DH-485 网络上。

2）I/O 扫描（理所当然）。

3）在 SLC 5/03 及其以上 PLC 中可以配置离散输入中断（DII）和 I/O 中断，所以当用户程序运行时，本地框架的通信处理器会周期性地读一个输入模块的数据。

4）当用户程序运行时，立即 I/O 指令（IIM 和 IOM）使用本地框架的通信处理器收发总线上单独的数据字。

5）在 SLC 5/04 及其以上 PLC 都提供有一个 DH＋通道。默认的 DH＋通道是通道 1。

6）集成串行口。只有通道 0 能用作串行通道。DF1 或者 ASCII 通信选择如前所述。

除了上述的用法，SLC 500 PLC 还能用作：

7）数据高速公路 485（DH-485）通道。通道 0 能被配置为用于 DH-485 通信而不是用于串行通信。要把通道 0 连接到 DH-485 网络上的共享导体，你需要一个接口适配器（1747-AIC）。

1. SLC 500 CPU 的通信配置

通过处理器状态窗口或者（有时候）通道配置窗口来改变 CPU 的状态文件，就能改变通信配置。这两个窗口都指向同一块配置存储区。

在第 10 章已详细讲述了处理器状态窗口提供的状态文件配置选项，在那章里我们也讲述了 DH-485 的节点地址和/或 DH＋节点地址、默认波特率、通信服务时间的限制和一个决定是否要保留一份记录来登记哪些站点在 DH＋上是主动站点的选项。设置中也包括选择是否使用全局状态字。如果全局状态字有效，当这个 CPU 向 DH＋网络中的下一个 PLC 传递令牌时，它会把 S：99 的内容连同通信令牌一起传递，也会从网络上其他 CPU 中接收全局状态字，然后存储到 S：100 至 S：163 单元中去（分别从节点 0 至节点 63）。

有些通信配置必须通过通道配置窗口进行设定。图 13-6 显示了 Allen-Bradley 的 APS 编程软件的通道配置窗口。它指出了两个通道：通道 0 和通道 1，它们都必须在使用前配置好。点击功能键会弹出另外一些隐含的选择设置选项的窗口。

```
CHANNEL 0 CONFIGURATION
              Current Communication Mode:  SYSTEM
                     System Mode Driver:  DF1 FULL-DUPLEX
                       User Mode Driver:  GPNERIC ASCII
                          Write Protect:  DISABLED
                           Mode Changes:  DISABLED
                Mode Attention Character:  /1b
                   System Mode Character:  S
                     User Mode Character:  U
       Edit Resource/File Owner Timeout:  60  (seconds)
                        Passthru Link ID:  1   (decimal)

CHANNEL 1 CONFIGURATION
                        System Mode Driver:  DH-485 MASTER
                            Write Protect:  DISABLED
       Edit Resource/File Owner Timeout:  60  (seconds)
                          Passthru Link ID:  2   (decimal)
```

图 13-6　APS 编程软件里的 SLC 500 通道配置窗口

根据你所拥有的 SLC 500 的型号，你可以把它配置为一个通道：DH＋系统通道、DH-485 系统通道、RS-232 DF1 系统通道或者 ASCII 用户通道。详细内容见第 10 章。

状态文件包含的一些只含状态信息的位（代替配置参数）。状态文件运行时，由 SLC 500 CPU 来更新这些位。用户程序可检查这些状态位来检测存在哪些主动节点：

■ 通道 0（如果有通道 0，则置位 S：33/4）。SLC 5/03 和 5/04 还保存一个记录哪些 DH-485 节点是主动节点（S：67 和 S：68）的位图。

■ 通道 1（如果有通道 1，则置位 S：1/7）。SLC 5/03 保存一个记录哪些 DH-485 节点是主动节点（S：9和 S：10）的位图，而 SLC 5/04 也保存一个记录哪些 DH＋节点是主动节点（S：83 至 S：86）的位图。

2. SLC 500 的 DH＋、DH-485 和读/写消息（MSG）指令

SLC 500 各型号之间有许多不同之处，它们使用读写消息（MSG）命令来交换数据的用法也有很大的差别：

1）MSG 命令不能出现在固定的 SLC 500 或者 SLC 5/01 PLC 的程序中。尽管如此，当这些 PLC 连接到 DH-485 网络上，其他 PLC 仍能使用 MSG 命令来读写这些 PLC 的数据文件。

2）MSG 命令只能用于 SLC 5/02 PLC 的程序中，通过 DH-485 来交换数据。

3）带有 OS 300 的 SLC 5/03 PLC 能执行 MSG 命令来与一个 DH＋上的 PLC-5 进行通信，只要 DH＋连接到 SLC 5/03 的 DH-485 上（通过 1785 KA5 模块或者配置为中继链接的 SLB 5/04 PLC）。MSG 命令必须选择 485CIF 作为目标设备，PLC-5 也必须拥有一个普通接口文件（CIF）接收来自 SLC 5/03 的数据或者保存 SLC 5/03 将要读取的数据。

4）SLC 5/03（配有 OS 301 或更高版本）PLC 在 MSG 指令里有一个 PLC-5 目标设备选项，所以它能与连接到 DH-485 的 DH＋上的 PLC-5 进行通信（通过一个 1785-KA5 模块或者一个中继 SLC 5/04）。

5）SLC 5/04 及其更高的 PLC 能直接地连接到 DH＋上，它们都有 PLC-5 目标设备选项。它们被 Allen-Bradley 称为具有网络兼容性。因为 SLC 5/04 PLC 同时支持 DH＋和 DH-485 两种协议，所以它可用作一个中继链接，把来自一个网络上的 PLC 的信息转换成另一个网络上的 PLC 能够理解的内容。当任何 SLC 5/04 PLC 接收到一个目的设备是自己的数据包时，它返回一个 ACK（确认）信息到源 PLC，表明数据包已收到。当 SLC 5/04 PLC 接收到一个数据包要经由自己传送到另一个网络上的 PLC 时，它不会发出 ACK 信息，而只是向前传递这个包到另外的网络上。最后，数据包会到达目的设备，然后由目的 PLC 返回一个 ACK，而中继 PLC 必须把这个确认信息传回源 PLC。

读/写消息指令（MSG）通过 DH＋，DH-485 或者 DF1 通道发起一次与目的 PLC 的数据交换。目的 PLC 不需要通过编程就能参与这次数据交换。SLC 5/03 及 5/04 PLC 有一个 MSG 指令的新版本，这个版本与之前的 SLC 5/02 的版本有轻微的差别。这个新版本在本书中已经讨论过，如图 13-7 所示。输入图中的参数后，在编程器中会出现另一个窗口，要求程序员输入另外的信息，其中必须输入的参数有：

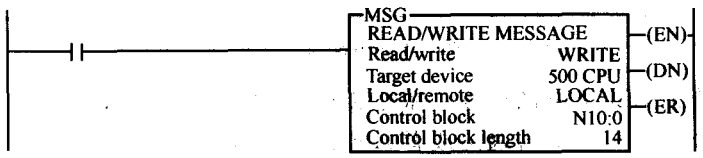


图 13-7 SLC 5/03 或者 SLC 5/04 的 MSG 指令

1）选择“WRITE”，如果你要从 SLC 500 的 PLC 中复制一个数据块到另一个 PLC 中；或者选择“READ”，如果你要从另一个 PLC 复制数据过来。

2）选择目标设备类型。如果目标 PLC 是 SLC 500 PLC，则选择 500 CPU；如果是

PLC-5，则选择“PLC-5”（在 SLC 500 与 PLC-5 PLC 之间不可以进行 I/O 数据文件的交换）。如果你选择了“485CIF”，数据会被当作单独的字或者字节（选择使用位 S：2/8）在 PLC 与另一个计算机里的常用接口文件之间传送。Allen-Bradley 称这种方法为 PLC-2 读/写方式。另一个 SLC 500 PLC 能被用作 485CIF 设备，这时数据通常会被写入它的数据文件 9，或者从数据文件 9 中读取数据。

3）控制块的完整文件地址。SLC 500 PLC 通常会从输入的地址开始保留 14 个数据字的存储空间，用来保存那些 PLC 进行消息传送时所需的参数和工作数据，其中包括用户程序能检查从而决定如何进行消息传送的状态位。图 13-8 显示了当 500 CPU 或者 PLC-5 被选作目标设备时控制块的结构图。当选择 485CIF 时，它的结构图会有所不同，而 SLC 5/02 PLC 的 MSG 指令只保留 7 个字的存储空间。

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	Word
EN ST DN ER CO EW NR TO | 错误代码	0
节点号	1
为字节长度保留	2
文件号	3
文件类型(O,I,S,B,T,C,R,N,F,St,A)	4
元素号	5
子元素号	6
保留（内部信息位） | WQ	7
消息定时器预设值	8
消息定时器	9
信息定时器累加值	10
保留（仅作内部使用）	11
保留（仅作内部使用）	12
保留（仅作内部使用）	13

图 13-8　目标设备是 500 CPU 或者 PLC-5 时，SLC 5/03 或者 5/04 MSG 指令的整型文件控制块的格式

4）选择“LOCAL”，如果目的 PLC 与源 PLC 连接到同一个网络上；或者选择“REMOTE”，如果目的 PLC 与源 PLC 连接到不同的网络上。

①如果选择了 LOCAL，必须输入以下参数：

- 连接到包含目标 PLC 的网络上的 PLC 的通道（0 或者 1）。
- 目标节点就是目的 PLC 的节点号，节点号在通信配置时分配。
- 目的或源文件地址（在源 PLC 里），响应源或者目的 PLC 的目标地址（在目的 PLC 里），以及以元素为单位的描述 PLC 间复制的数据的消息长度。如果目标类型是 485CIF，则指定一个目标偏移量来代替目标地址，这个偏移量表明了数据是从目标计算机里的 CIF 文件的哪个位置开始的。
- 可以输入一个以秒（1 至 255）为单位的消息超时时间。如果输入数值 0，这个消息将永不会“超时”。

②如果选择了“REMOTE”，源 PLC 必须把数据包的格式做稍微的修改才能传输出去。

Allen-Bradley 的文献中称这些包为网关信息包，而在 MSG 指令里要求输入另外一些附加参数。就 DH＋和/或 DH-485 网络互联方式的不同，对应有两种方式可选择参数：

- 如果 SLC 5/04 PLC 同时连接到两个网络上，并且配置为中继链接（有时候称为网桥或者网关[⊖]），这两个网络就通过这个 PLC 互联起来了。这个中继 PLC SLC5/04 必须把两个通道都配置好，这样每个通道都有不同的链接 ID 号（0 到 255）来区分各自连接的是 DH＋还是 DH-485。这个 SLC 5/04 PLC 的工作方式类似 1785-KA5 模块（1785-KA5 模块插到 PLC-5 的一个框架里连接到数据高速公路）。如果目标 PLC 是在另外的网络上，当编辑消息指令时，必须包括以下的附加信息：
- 本地网桥节点地址是用作中继链接的 SLC 的地址，在同一个 DH＋或者 DH-485 上就好比带有 MSG 指令的 PLC（中继链接 PLC 分别为它连接的两个网络各自分配一个单独的节点地址）。
- 远程网桥链接 ID 是目标 PLC 连接到的那个网络的链接 ID 号。
- 远程网桥节点地址是一个模糊的名字，特别是在以下网络通信情形下：

如果目标 PLC 是 SLC 5/03 和 SLC 5/04，或者 PLC-5 PLC，那么远程网桥节点地址是 0。Allen-Bradley 称 PLC-5、SLC 5/03 和 SLC 5/04 PLC 为具备网络能力的设备。

如果目标 PLC 没有网络能力，远程网桥节点地址与本地网桥节点地址一样。这就意味着网桥可能要对消息做一定的转换才能让级别低一点的处理器识别。

- 如果目标 PLC 在更远的远程网络，就要求使用多于一个中继链接和/或 1785-KA5 模块，这样 REMOTE 参数值的选择就由系统里的网络类型所决定。详尽的寻址要求可参考你的操作手册。

3. SLC 500 响应 MSG 指令的方式

如果在程序执行期间 MSG 指令的条件为真，就会发出一个消息请求，放到通信处理器中的等待队列，申请适当的通信通道。新一代的 SLC PLC 会立即尝试把要发送的数据（或者数据请求消息）送到其中一个消息缓存器的一个消息包里。每次执行 I/O 扫描后，PLC 紧接着服务通信处理器，一旦缓存器可用则立即写输出数据包到缓存器中，并从其他缓存器中复制输入数据包。等待队列中的请求按照进入队列的先后次序执行[⊜]。当 MSG 指令尝试进入等待队列时，即使等待队列已经排满了请求，处理器也会在下次 MSG 指令被执行或下次通信服务在运行时作再一次努力尝试。

每个 PLC 都有一个顺序（称为令牌）通过发起数据传输来使用网络。它能发送数据到其他的 PLC 或者要求其他 PLC 发送数据。当其他 PLC 获得令牌时，它们响应这些请求发送数据包。当通信处理器发起数据传输或者响应发起者的数据传输请求时，它写状态文件和

⊖ Allen Bradley 可以忽略网络连接术语的互相矛盾的混用，仅仅是因为大多数其他制造商把这些术语弄混了。事实上，根据 ISO，网桥仅指连接使用不同控制方法和不同信号或寻址方法的网络，而不是不同数据包内容要求，而网关甚至可以通过转换数据包中的数据来连接完全不兼容的网络。ISO 没有定义术语“链接”，所以这是一个可以安全使用的词。在 DH485 和 DH＋协议之间转换似乎要求比网桥稍微多一些，尽管使用一个专有网络，如 Allen-Bradley 提供的，其好处是用户不需要知道技术细节。

⊜ SLC 5/02 只有一个消息缓存器，一些 5/03 有四个，其他 5/03 和所有的 5/04 每个通道都有四个。5/04 和一些 5/03 可以在每个通道等待队列中保存 10 个消息，其他的 5/03 在队列中总共可以保存 10 个请求，而 5/02 据说可以保存“几个”请求。

MSG 指令控制块中的状态位，这些位能被用户程序检测到。状态文件位包括：

1）输出消息命令等待位。PLC 执行 MSG 指令时，产生一个消息请求进入等待队列，但缓存器里却没有数据包要发送时（使用通道 1 时写 S：2/7，使用通道 0 时写 S：33/2），把这个位置位。当下一步是通信服务时，检查最先进入等待队列的消息请求，并且把要发送的数据（或数据请求）装到一个数据包并送到缓存器中去。如果不能按请求集成数据包，MSG 指令控制块中的 ERROR 位置位（参考下面所述），然后执行等待队列中的下一个消息请求。

2）消息回复等待位。当另一个 PLC 已经发送了这个 PLC 的 MSG 指令所请求的数据时（若从通道 1 接收数据则写 S：2/6，若从通道 0 接收数据则写 S：33/1），把这个位置位。在下次通信服务期间，CPU 从输入数据缓存器中读取数据到数据存储器中，并置位 DONE 位。其中 DONE 位是在向其他 PLC 请求数据的 MSG 指令的控制块中。

3）输入命令等待位。当网络中另一个 PLC 已经发送了用户程序中的那条 MSG 指令请求的数据或者一个命令请求数据时（若从通道 1 接收输入数据包则写 S：2/5，从通道 0 接收则写 S：33/0），把这个位置位。在下次通信服务期间，检查刚收到的数据包：

如果数据包里的是其他 PLC 发送给这个 PLC 的数据，则把数据写入存储器，然后发内容为“我已经成功完成你的写请求”的信息包到输出缓存器或输出消息队列中。如果数据不能写入到这个 PLC 的数据存储器中，这个 PLC 会返回“我不能完成你的请求，你出错了”的信息包。

如果接收到的数据包里的是其他 PLC 要读取这个 PLC 的存储器数据的请求时，从数据存储器中复制请求的数据到一个输出消息缓存器中，结尾加上“这就是你请求的数据”的消息包（或者你的请求正在排队争取把数据传到缓存器中去）。如果不能把请求的数据集成到数据包，则返回“我不能完成你的请求，你出错了”的消息包。

每个 MSG 指令都会具体指明一个控制块的起始地址，这个地址是以一个整型文件字地址的形式输入的。PLC 从那个单元起保留 14 个消息控制块和状态字给通信处理器使用。图 13-8 显示了消息控制块的结构。通信处理器在执行 MSG 指令请求的任务时，可改变控制块的状态位如下，除了特殊说明，它们全都是在控制块的第一个字中。

1）使能位 EN。如果 EN（位 15）置位，则消息指令逻辑为真。

2）队列等待位 WQ。如果 WQ（字 7 的位 0）置位，则队列已经排满，消息请求不能再进入队列，直到顺序扫描成功让这个请求进入队列，才复位 WQ。SLC 5/02 PLC 不使用这个位。

3）使能并等待位 EW。如果 EW（位 10）置位，则数据正在缓存器中等待发送（并且不能修改）。

4）开始位 ST。如果 ST（位 14）置位，则目的 PLC 返回 ACK 信息，表明已经接收到数据包。

5）完成位 DN。如果 DN（位 13）置位，则目的 PLC 已经返回一个回复消息包来表明已成功从源 PLC 接收到数据，或者已经发送源 PLC 请求的数据。

6）没有响应位 NR。如果 NR（位 9）置位，则其他 PLC 还没有对数据包响应 ACK 消息。

7）错误位 ER。如果 ER（位 12）置位，则这个 PLC 不能集成数据包发送，或者 NR 位在整个扫描循环期间都保持为 1，或者其他 PLC 通知本 PLC 它不能响应消息请求，或者

消息超时（参考下面关于时间溢出位的叙述）。如果其他 PLC 由于忙而不能接收数据包，这个 ER 位不置位，而消息则保留在队列中等待下一次的尝试。

程序员或者程序可以写控制块中的某些位来控制通信，这些位包括：

8）连续操作位 CO。置位 CO（位 11），则即使错误发生，消息也重复发送。

9）时间溢出位 TO。置位 TO（位 8），则在 MSG 指令已启动但不能在指定的时间内完成之后，清空缓存器和等待队列中的消息。输入一个时间（从 1 到 256 秒）到控制块的字 8 中。

如果你没有置位时间溢出位，一个不能发送的输出数据包会一直占用一个缓存器空间。为了从缓存器中移走这个消息，你必须把时间设为 0 秒，然后置位时间溢出位并启动一条扫描指令。直到你清空时间溢出位和重新启动 MSG 指令，否则不再执行 MSG 指令。

4. 其他的 SLC 500 PLC 通信指令

以 5/02 开头的 SLC 500 型号都提供有使 PLC 中断用户程序的指令。其中，服务通信（SVC）指令使 PLC 不必等到 I/O 扫描就直接执行串行通信通道服务。I/O 更新（REF）指令（在第 11 章已经讨论过）引起一个完整的扫描循环，包括串行通信通道服务的扫描。SLC 500 PLC 只有有限的通信缓存器空间（SLC 5/02 PLC 只有仅能容纳一个单独数据包这么大的缓存器），所以用户程序可以监控输入命令等待、消息回复等待和输出消息命令等待指令的状态文件位（如前所述），以及要请求执行 SVC（或者 REF）命令来改善通信性能。

SLC 5/03 及更高的 PLC 允许程序员指定两个串行通信通道中的哪一个响应 SVC 或者 REF 指令，但不能把 SVC 和 REF 指令编写到响应 I/O 中断、DII 或者 STI 的中断服务例行程序中。

有多个 ASCII 指令可用于通过一个配置成 ASCII 用户通道的通道来发送和接收数据。这些指令都已经列在本章的 PLC-5 通信部分，但本书中没有对它们进行讨论。

S5 S7

13.4 使用 PROFIBUS 的 SIEMENS 的 PLC 通信

虽然 Siemens 的客户都不自觉地认为 Profibus 是属于 Siemens 公司的，但事实并非如此。不过 Siemens 确实在促进 Profibus 网络的发展中扮演了很重要的角色。

现在还有四种类型的 Profibus（过程现场总线）的网络在使用，所以连接到 Profibus 网络需要正确的接口器件。

1）你可能遇到的最旧的 Profibus 版本经常就被称为 Profibus，但有时候会用字母 FDL（现场总线设备级别）或者描述符 part 1 来与其他版本区分开。它主要是一个主-从网络。Siemens 的 L1 网络就是基于这个早期的标准。

2）Profibus-DP（分布式的外围设备或者分散的外围设备）是当今最常用的类型，它用于把主动节点互联起来，让各个主动节点以对等方式互相发送消息，以及发送到从属于主动节点的被动节点。Siemens 有一个 Profibus-DP 早期的版本，它称之为 L2 网络。Siemens 现在使用的 Profibus-DP 与欧洲的标准 EN 50 170 相一致。

3）Profibus-PA（过程自动化）除了使用较低的电压和电流外，与 Profibus-DP 安全一样。安全性高的应用中的计算机化的设备不能产生电火花或者电流噪音，Profibus-DP 是提供给高安全性的应用设备使用的。

4）Profibus-FMS（现场总线消息服务）是最快的和最高级别的 Profibus 网络，它是为

迎合 EN 50 254 标准而设计的。德国的厂商希望它能成为现场总线基金会承认的开放式通信网络的世界性标准。但现在却选择了另一个系统（快速以太网），而作为现场总线基金会可接受的网络标准的 Profibus-FMS 预计大概在 2004 年停止使用（有时候计划并不是能以本来意愿完成的）。

13.4.1　使用 SIEMENS S5 的通信

S5 PLC 能通过自己 CPU 模块中的串口或者接口模块（IM）或者通信处理器（CP）模块进行通信。S5 PLC 的 CPU 模块中主串口用于连接编程器，所以其常常被称为编程端口。有些 S5-115U CPU 模块还有第二个串口，这样它们能同时使用两个通道连接来进行通信。使用串口与另一个 PLC 通信（而不是只连接到编程器）要求配置 CPU 模块使用部分存储器用于协调通信和保存数据包。用户程序必须写输出数据到发送邮箱，然后用发送协调字节初始化数据传送。用户程序必须监控接收协调字节来检测数据的接收情况，然后必须从接收邮箱复制刚收到的数据。S5 CPU 能以三种不同的方式使用 CPU 端口进行通信：

1）在点对点通信中，只有两个 S5 PLC 互相连接来通过它们的邮箱进行数据交换。因此只需要一根普通的通信电缆。

2）在 L1 网络通信中，可有多达 30 个 S5 CPU 作为从站互联到一个 L1 主站。这个 L1 主站可能是 S5-115U 框架中的 L1 通信处理器模块，也可能是配置有 L1 接口卡和通信软件的个人计算机，但单纯的 CPU 模块不能作为主站。通常 L1 主站轮询每个 CPU 从站，读取它的发送协调字节来判断它是否要发送数据，如果是，就执行数据传输请求。对 L1 主站编程能让它响应其他选定的 CPU 模块的中断请求，中断正在处理的普通优先级的消息而传送高优先级的消息。每个 CPU 模块都需要一个 L1 接口模块来把编程端口的当前环路协议转换成共用通信总线上的 RS-485 协议。通信处理器模块也是可用的，这样 S5 PLC 能在 L1 网络上通过 I/O 模块而不是它们自己的编程端口来通信。

3）有些 S5-115U CPU 模块内置有 ASCII 驱动器。这些 CPU 模块能使用 ASCII 串行通信来与另一个计算机交换数据包，或者与串行设备，例如条形码读卡机或者串行打印机等，进行数据包的交换。除了协调字节和邮箱之外，也必须为 ASCII 参数设置分配存储器（在 CPU 配置期间），这个参数设置必须包含描述 S5 CPU 将如何与其他计算机或者外部设备进行通信的参数。接口硬件可能需要把 CPU 中的通用环信号转换成其他计算机/外围设备所要求的信号类型。

使用中心框架和扩展单元框架中恰当的接口模块（IM），S5-115U CPU 可以连接到远程 I/O 系统里。S5 CPU 可以连接到 Profibus 网络、MODBUS 网络或者使用点对点协议的 RS-232C 设备上。这些通信过程需要适当的通信处理器（CP）模块。AS-I CP 模块可用于连接执行器/传感器-接口网络，这样 S5 PLC 能控制它们。

1. S5 L1 与点对点串行通信

S5 CPU 必须配置成 L1 从站的形式或者在点对点模式下通信（L1 主站在后面会讲到）。将 S5 CPU 配置为 L1 从站或者配置为点对点通信，必须输入网络地址和指向 CPU 系统的数据存储区的指针，就如第 10 章所描述的那样。必须输入的配置数据包括一个 CPU 惟一的 L1 从站节点号，以及为协调字节和邮箱的数据包预留的存储区的指针。

协调字节包括用户程序能读写的控制位和状态位，它们有：

位 7，在发送协调字节（CBS）里把这个位置位，则发送发送邮箱里的数据，或者在接收协调字节（CBR）中置位来允许数据接收到接收邮箱里。当 PLC 进入运行模式时，操作系统自动把 CBS 和 CBR 中的位 7 置位。

当 CBS 中的位 7 置位时，禁止 CPU 改变发送邮箱里的内容；当 CBR 中的位 7 置位时，禁止 CPU 读接收邮箱里的内容。

执行完一次数据传输后，STEP 5 操作系统复位位 7（从发送邮箱里传送完数据则复位 CBS 中的位 7，或者把数据接收到接收邮箱后复位 CBR 中的位 7）。你的程序应该读这个位来判断什么时候发生了数据的传输。

位 0，在最近的数据传输请求期间，如果检测到一个错误，STEP 5 操作系统置位位 0。

在协调字节里的以下状态位只用于 L1 网络：

位 4，在发送协调字节（CBS）中这个位置位，表明发送邮箱里的数据应看作为比其他 PLC 的数据传输请求有更高优先级的数据。当传送高优先级的数据时，STEP 5 操作系统置位目的控制器的接收协调字节（CBR）的位 4。

位 1，当 STEP 5 操作系统检测到有个节点关闭，它就置位 CBR 中的位 1。

位 2，当 L1 网络在运行模式下时，STEP 5 操作系统置位 CBR 中的位 2。

位 3，当编程器使用 L1 网络时，STEP 5 操作系统置位 CBS 中的位 3 来阻止其他数据传输。

发送邮箱能容纳一个多达 66 个字节的数据包。接收邮箱应该有足够大的容量来容纳其他节点发送的最大数据包。数据包必须包含有：

在字节 1 里，是一个表明整个数据包长度的数据（2 到 66 字节）。这个数字与数据包的其他字节一起传送。

在字节 2 里，是一个表明哪个/些节点负责传送这个数据包的编码。节点 0[⊖] 是 L1 主站，节点 1 到 30 是其他从站节点，节点 31 意味着要广播这个数据包到网络上其他所有的节点。当这个包写到接收节点的存储器时，STEP 5 操作系统改变字节 2 来表明发送节点的地址。

字节 3 到 66，包含发送节点想要发送的数据。

注意：字节 1 表明实际上总共有多少字节（包括字节 1 和 2 在内）要发送。

2. S5 的 ASCII 串行通信

有些 S5-115U CPU 模块的串口里内置有 ASCII 驱动。要使用 ASCII 通信，必须包括如下配置（参考第 10 章）：为串口选择 ASCII 驱动，指向协调字节、邮箱的指针以及指向 ASCII 参数集的指针。

用户程序对位 7 置位，发送协调字节（CBS）中的位 7 能发起数据传输，而接收协调字节（CBR）中的位 7 允许数据接收。（在完成一次数据传输后，STEP 5 操作系统清空这些位）在数据传输期间，如果检测到错误，CBS 和 CBR 中的位 0 同时置位。

发送邮箱和接收邮箱有最多一千字节的只能保存数据的存储空间。ASCII 参数集表明数据包大小是否适合，或者表明一个特定的控制字符来指示数据集里的数据信号的结尾。

3. 通过接口模块和通信处理器模块的 S5 通信

接口（IM）模块提供信号的转换和一些小型通信控制功能，把 S5 PLC 连接到与它们的

⊖ 节点 32 也表示 L1 网络主站，如果从站通过通信处理器模块而不是编程端口与 L1 网络连接。

串口信号类型不兼容的网络上。通信处理器（CP）模块能用于连接 PLC 到一个类型差异很大的网络上，例如 Profibus 网络。

有些 CP 模块使用 CPU 模块的接口存储区（在第 5 章没有介绍接口存储器，是因为它只用于与 CP 模块的通信）与 S5-115U CPU 模块交换数据。CPU 模块中接口存储器的每个 1k 字节部分是预留给与 CP 模块共享的数据[⊖]。CPU 模块中的通信操作处理器自动处理 CP 模块和接口存储器模块之间数据的往返传输。每个 CP 模块都有一个惟一的接口号，这个接口号是在 CP 模块中使用 DIP 交换的硬件设置期间分配的，或者保存在 CP 模块的 EEPROM 存储器里。接口号有时会改变，只要在这个 CPU 发送到 CP 模块的数据中写入一个新的号码传到 CP 模块里。用户程序必须执行数据处理功能块去读/写 CPU 中的接口存储器（因此读/写 CP 模块的数据）。S5-115U CPU 本身就提供了预编程数据处理块（参考第 7 章和第 10 章）。

有些 CP 模块使用处理器之间的通信标志来控制 CP 模块和 CPU 模块之间的通信。在第 10 章我们已经讨论过如何配置 CPU 来使用处理器之间的通信标志。

4. Siemens 的 L1 网络主站

L1 网络必须有一个主站节点。主站轮流读取每个从站的发送协调字节（CBS）检查数据传输请求，并执行数据传输请求任务。正如关于 CBS 各个位的隐含描述，主站也会响应从站的中断请求来执行更高优先级的数据交换操作。L1 主站可以是 S5-115U PLC 框架里的 CP 530 通信处理器模块，也可以是装有接口卡的个人计算机。

CP 530 模块的配置在第 10 章里已经讨论过。配置包括有命令一个用户程序中的数据处理功能块读和写数据到 CPU 内存的接口存储器。S5 PLC 的通信处理器自动在 CPU 和智能 I/O 模块间（包括通信处理器模块）复制接口存储器的数据内容，然后与 CPU 交换数据。配置完成后（可能还要打开 L1 网络），CP 530 模块不需要其他更进一步的配置或者控制就能作为 L1 网络的主站开始工作了。连接到 L1 网络的从站也能交换数据。

用于配置 CP 模块的同一个数据处理功能块也可以被 L1 主站的用户程序用来发送和接收 L1 网络上的数据。虽然在第 7 章和第 10 章里，我们已经讨论过数据处理功能块的内容，但我们在这里还是要回顾一下它们的通信用途：

FB 246，“同步”，每次 PLC 重启时都要执行一次，使得在 CPU 模块和 CP 模块间进行同步通信（在配置期间就会被调用）。

FB 244，“发送”，用于写配置数据到 CP 530 模块，但配置后，发送用于发送控制数据和消息数据到 CP 530 模块。必须包含在 L1 网络上发送数据的调用的参数包括：

■ A-NR：工作号。输入如 KY 0，y.，其中“y”是一个工作号，表明这个数据传送给 CP 模块。SEND 设置好后，必须连同以下的工作号才能通过 L1 网络实现数据发送：

1 到 31　接口存储器的数据包含有一个发送邮箱的数据集，这些数据从本节点（主站节点）传送到节点号为 1 到 30 的节点，或者数据要广播到所有节点（工作号 31）。发送邮箱的数据集必须包括：

● 一个发送协调字节（CBS），然后是

⊖ 也可以使用分页（page framing）的方式来代替为每一个 CP 模块分配内存的线性方法。在分页中，所有的 CP 模块使用同一个 1k 比特的 CPU 存储空间。见你的用户手册。

- 一个未使用的字节，然后是
- 最多 64 字节的数据

发送协调字节能用来调节目标 CPU 从停止模式（如果位 7 置位）到运行模式（如果位 6 置位）。

51 到 81 包括一个发送邮箱的数据配置的数据，需要以更高的优先级传输到从站 1 至 30（工作 51 到工作 80）或者广播（工作 81）。

- ANZW：两个字的条件码数据地址。当数据块打开时，输入一个标志字地址（例如 FW6）或者一个数据字地址。这两个字的第一个字要包含有通过 L1 网络传送数据到 CP 模块和到另一个 CPU 的请求传输的状态标志位。第二个字会包含一个数字，表明响应这个请求已经有多少数据字在 CPU 和 CP 模块间进行了交换（参考第 7 章和你的操作手册）。
- QLAE：有多少个字或者字节的数据复制到接口存储区。输入如 KF x，其中“x”是数据字的数量或者标志字节的长度。

记住发送邮箱的数据集包括一个放在数据前的双字节的头。

FB 247，“控制”，用于读 CP 530 模块的状态。必须输入的参数包括有：

- ANZW：双字的条件码数据的存储地址。如果由于从站使用了一个适当的工作号而被挑选出来，控制功能块就使用这些字来描述与这个特定的从站之间通信的状态。被传输的字的计数值会存到字的高位，而字的低位则包括以下重要的状态位：

0 当 CP 模块已经从另一个 PLC 接收到一个数据块，这个位置位；当一个接收指令完成从 CP 模块读取数据时，这个位复位。

1 当执行发送函数，向 CP 模块提供数据发送到另一个 PLC 时，或者当获取（FETCH）函数请求 CP 模块从另一个 PLC 获取数据时，这个位置位。当发送数据或者获取请求已经发送给其他的 PLC 时，这个位复位。

2 完成发送或接收而没出错的情况下，这个位置位；当执行新的发送或接收功能块时，这个位复位。

3 发送或接收检测到错误，这个位置位。一个错误代码写入位 8 到位 11 中。当开始一个新的发送或接收时，这个位复位。

4 当 CPU 模块和 CP 模块网络之间的数据传输执行时，这个位置位。

5 由发送功能块置位。在数据从 CPU 传输到 CP 模块之后，这个位复位。

6 由接收功能块置位。在数据从 CP 传输到 CPU 模块之后，这个位复位。

7 可以由用户程序置位来禁止将来的发送或接收功能块访问 CP 的接口存储器，或者由用户程序复位来恢复发送和接收。

- A-NR：请求的条件码状态数据的工作号。可输入任何一个工作号，但可能最有用的应用就是从从站接收数据的工作号，这个工作号可用来决定 CP 530 模块是否含有一个从站发到主站的消息。使用工作号：

101-103 job 101 读取从从站节点 1 发来的消息的状态，而 job 102 读取从从站节点 2 发来的消息的状态，等等。

FB 245，“接收”，用于读取 CP 530 模块状态和 CP 530 接收到 CPU 内存的消息。必须和接收指令一起输入的参数有：

■ ANZW：返回与控制接收到的相同的条件码数据。

■ A-NR：工作号，如发送所述的，但使用一组不同的工作号：

101-130　从 CP 530 模块读取接收邮箱里的数据包到 CPU 内存。job 101 读来自从站节点 1 的数据包，job 102 读取来自从站节点 2 的数据包，等等。每个接收邮箱的数据集由一个 4 字节的头和数据组成：

- 第一个字节是接收协调字节（CBR），紧接着的是
- 这个消息的数据字节个数，然后是
- 发送这个数据的节点号，然后是
- 一个未使用的字节，接着是
- 最多 64 字节的数据

221　读全部系统状态（SYSTAT）的数据集，其中包括一个错误号和一个最近通信问题的错误代码（参考你的操作手册可了解错误号和代码）。

220　只读 SYSTAT 数据的 10 个字节，包括最多三个错误号和代码。

201　读一个 4 字节长的从站节点号列表，这个从站在最近一次轮询循环里没有响应。

223　读当前 SYSID 配置数据（用 job 222 写的数据）

143　读当前轮询列表配置

144　读当前中断列表配置

142　读当前控制字节

FB 248，“复位”。取消所有停留的工作，为该工作重新设置状态。复位的参数是 SSNR、PAFE 和 A-NR，这些已在前面论述过。

5. S5 PLC 和 Profibus 局域网

使用接口模块（IM 308-C），S5-115U CPU 可连接到 RS-485 Profibus-DP 或者 Profibus-FDL 网络。在 Profibus-DP 网络上，S5-115U 是主动节点，这意味着它可以含有一个命令数据交换的程序。S5-115U 可以作为 Profibus 的主站，协调所有 Profibus-DP 的通信量，或者 CP 5431 模块可用作 Profibus-DP 或者 Profibus _ FMS 的主站，在 S5-115U CPU 的控制下执行控制功能。

通过通信处理器（CP 541）模块，S5-100U 或者 S5-115U 的 CPU 模块可连接到一个 Profibus-DP 或者 Profibus-FDL 网络上。CP 541 通过编程端口连接到 S5 CPU 上，而不是通过 PLC 的总线系统。在 Profibus-DP 网络里，S5-100U 的 CPU 只能用作被动组件，这意味着它们只能响应 Profibus-DP 上主动节点的命令以发送/接收数据。

通过 Profibus-DP 网络，主动控制器能控制 S5-100U 的 I/O 模块框架。安装 I/O 模块到分布式的 I/O 模块（例如装有 CP 318 Profibus-DP 接口模块的 ET200U），可以代替 S5-100U 和 CP 541。分布式的 I/O 模块通常是被动节点。

其他安装有 CP 模块的 ET200 分布式 I/O 模块，也能连接到 Profibus 网络上。接口可用作 Profibus-DP 从站节点连接执行器-传感器接口（AS-I）传感器网。装有 Siemens 专用的接口卡的个人计算机也能连接到 Profibus 网络上并可作为主动组件。

13.4.2　使用 SIEMENS S7 PLC 的通信

Siemens S7 PLC 通过子网进行通信。子网可以是任何点对点的连接或者网络，它可以

是一个比它大的网络的一部分。S7 PLC 构成的网络中，最常见的类型包括多处理器接口（MPI）、Profibus（特别是 Profibus-DP）和工业以太网。在第 10 章里讨论了如何组建一个网状 PLC 系统，也大致讨论了如何使用各种类型的通信子网。在这一章，我们会更仔细的介绍网络配置，并讨论怎样使用不同类型的通信子网。

1. S7 PLC 的通信配置

S7 PLC 的通信配置，包括如何使它与其他节点通信和通过哪种类型的网络进行通信两方面。

1）可以在 MPI 网络内配置全局数据环，这样选定的数据会在少数的 S7 节点间自动进行交换。

2）用户程序可以包含标准函数（SFC）与 MPI 网络上的其他节点进行数据交换，而不必预先配置节点间的连接（有时称为无配置连接的通信）。

3）标准功能块（SFB）由 S7-400 CPU 提供，用于任何网络类型的节点间进行数据交换，只要这些节点已经互相配置好连接（已设置连接的通信）。

4）当配置 Profibus-DP 网络时，你要为每个 DP 从站在 DP 主站的存储器里分配地址空间。DP 主站读或写使用外部 I/O 寻址的从站数据中的多达 4 个输入字节和 4 个输出字节，这就好像从站是在 DP 主站框架系统的一个插槽里一样。Profibus-DP 系统负责在从站内存和主站内存之间复制数据。

5）Profibus-DP 主站能使用用户程序里的 SFC 读或写数据集，设置的长度比 DP 从站的 4 输入和 4 输出字节还要长。

如果 S7 CPU 模块检测到通信错误，它会以一个优先级小于 35 的中断来中断所有程序，然后执行通信错误组织块 OB 87。如果没有 OB 87，PLC 会进入出错模式。当调用 OB 87 时，STEP 7 操作系统会产生一个单字节的错误代码（符号名：0B87 _ FLT _ ID _）和两个附加码来识别出现的问题。有关出错中断的更多信息可参考第 11 章和参考操作手册。

2. MPI 网络中的 S7 全局数据环

配置 S7 CPU 加入到 MPI 网络的全局数据环中：

1）必须已经在这个方案中创建和配置好了一个 MPI 子网，而且必须为子网中至少两个模块输入了 MPI 配置。

2）选择 MPI 子网，然后选择“OPTIONS”，然后就是“DEFINE GLOBAL DATA”，就会弹出一个 MPI 子网的全局数据配置的配置表。图 13-9 演示了配置表在你输入完某些全局数据参数时会如何出现，以下面的步骤。

	GD 标识符	Stn 1/CPU 1	Stn 2/CPU 2	Stn 3/CPU 3
	GD 1.1.1	MB100:16	>>IB4:16	MB10:16
	GD 1.2.1		MW40:4	MW60:4

图 13-9 STEP 7 全局数据表配置窗口

3）双击空白栏的头部，就会弹出一个配置好并连接到这个子网上的站点的菜单。选择其中一个站点。对其他要加入这个全局环的站点也重复此操作。在全局数据环中，任何一个站点都能发送数据到最多四个其他站点。

4）表中的每一行都必须描述每个 PLC 将会把全局数据包存到哪里，这些全局数据包将会在全局数据环的成员中传输。在第一行选择一个单元，按 F2 键，然后在该站点输入你要进行全局数据环通信的地址。在输入的地址后面加上冒号，然后是字节的个数（最多是 22），这个数就是你想要在全局数据环中交换的数据包的字节数。地址可以是任何存储区的，除了本地（L）或者外围的 I/O（PI 和 PQ）。就可以指定任何数字和数据单元的大小，最多 22 字节。

如果多于两栏填上了地址（如图 10-21 中的行 1），其中一个站点必须指定为发送方，当光标在适当的单元上时，选择“SELECT AS SENDER”即可实现。图 13-9 中的行 1 配置了 Stn 2 发送本地存储器 IB 4 的内容到 IB 19（16 字节），然后发送到 Stn 1 和 Stn 3。在 Stn 1，它会把这 16 字节覆盖到从 MB 100 开始的位存储单元中，而在 Stn 3 中，它会覆盖数据到 MB 20 开始的位存储单元中。

如果只有两栏填上了地址（如在行 2 中），就不必指定一个发送方。不论何时，其中一个站点写数据到这个 GD 包使用的存储区时，另一个站点的响应存储区也会随之改变。

5）其他行也重复以上的数据包定义。每个全局数据环最多同时交换四个包。

6）创建额外的全局数据环，只要选择“GD TABLE-OPEN-GLOBAL DATA FOR SUBNET”即可，然后重复以上步骤 2 至 5。如果使用的是 S7 CPU 模块，每个站点可以配置为最多 16 个不同的全局数据环的成员（如果是 S7-300，最多只能加入四个环）。

7）选择“GD TABLE COMPILE”。如果你正确地输入数据，编程软件根据你的全局数据配置创建系统数据块（SDB），而且在成功完成这个阶段 1 的编译步骤后会返回一个报告。

8）这是个可选步骤，但你也有可能需要设置扫描频率和/或创建全局（GD）状态字。这些步骤必须在阶段 1 编译完成（也就是现在）之后做好：

- 选择“VIEW-SCAN RATE”，然后输入扫描频率。你选择的扫描频率应该是 PLC 的扫描循环时间的倍数。当发送数据时，S7-300 PLC 的扫描频率应该大于 60ms，而 S7-400 PLC 是 10ms。如果是接收数据，（扫描频率×循环时间）应该比发送数据的频率短。如果输入 0 作为扫描频率，全局数据包将不能自动发送，而只有当响应命令发送或接收全局数据的 SFC（SFC 60、GD _ SEND 或者 SFC 61、GD _ RCV）执行时，全局数据包才能进行交换。
- 为每个你想要的可获得全局数据交换状态的 CPU，选择“VIEW-GA STATUS”，然后输入一个全局数据状态（GDS）的双字可以储存的单元地址。但这样全局数据包的接受方不会有响应，所以如果没有配置 GDS 状态字，就没有办法确定全局数据的传输是否成功。数据交换成功时，GDS 双字中的所有位都将变为 0。当使用编程器监控全局数据的交换时，这个编程器与所有 GDS 双字进行“或”操作并显示一个组状态字（GST）。
- 在输入完扫描频率和/或 GD 状态配置后，你必须再一次选择“GD TABLE COMPILE”。

9）完成所有输入后，保存和下载这些配置参数。

现在，用户程序可以写存储器，它将会复制到全局数据环包中，而数据也会自动复制到全局数据环的其他 PLC 中。在发送方发送新的数据前，应先对其他 PLC 编程来读取接收到的全局数据。用户程序可以调用两个标准功能块（SFC 60、GA _ SEND 和 SFC 61、GD _ RCV）来发起全局数据包的交换，而不等待循环的交换。

3. 无配置连接的 S7 SFC

无配置连接使用 SFC 时，必须要配置好子网，还要配置每个节点在哪个子网上，如第 10 章所讲。只要你执行其中一个 SFC，S7 CPU 就会自动地建立与其他计算机的连接。

以下的 SFC 预编程到 S7 CPU 模块中，它们都能被用户程序调用：

SFC 67，“X _ GET”。从同一个 MPI 网络上的另一个 CPU 中复制一个数据元素。目的 CPU 无需编程就能响应（Siemens 称两个 PLC 都不需要编程的通信为单边通信）。

■ X _ GET 的输入参数包括：

REQ，如果该位为开，当工作（job）在上一次扫描没有进展时，X _ GET 要求通信处理器获取数据元素。数据元素接收到后，如果 REQ 还为真，则再要求获取数据。

DEST _ ID，表明其他 CPU 的 MPI 节点号的字。

VAR _ ADDR，任意型指针，它指向含有从其他 CPU 读取数据元素的数据单元。其中数据元素不可以是结构体或者布尔数组。如果数据单元不能分块传输，那么对于 S7-300 CPU，数据单元不该超过 8 字节长，在小型 S7-400 CPU 之间它不该超过 16 字节；或者对于 S7-414 及其以上的 CPU，不该超过 32 字节。

CONT，如果该位为开，数据发送完成后仍保持连接，这样当下次数据还是发送到同一个目的节点时可节省时间。如果这个位关，则发送完成后断开连接，这样它对其他通信也是可用的。

■ X _ GET 有三个输出参数：

RD，任意型指针，它指向输入数据该存放到 CPU 的那个存储区。它必须与 VAR _ ADDR 一样大小。

BUSY，当通信缓存器接受了工作，或者工作已经在之前的扫描就被接受，BUSY 位置位。如果不能接收这个工作，BUSY 位复位。

RET _ VAL，包含描述工作状态的完整代码（包括连接难点），或者包含 X _ GET 函数不能工作的原因。

SFC 68，“X _ PUT”。写一个单独的数据单元到 MPI 网络上的另一个 CPU。就好像 X _ GET一样，X _ PUT 也是单边的，所以其他 PLC 不需要编程就能接收数据单元。X _ PUT 的参数包括：

REQ、CONT、BUSY 和 RET _ VAL：如前面 X _ GET 所述。

DEST _ ID 和 VAR _ ADDR：和 X _ GET 的一样，除了它们现在指明的 MPI 节点和来自这个 CPU 的数据将写入的存储地址不同。

SD：任意型指针，指向含有一个将要写到其他 CPU 的单独数据元素的存储区。它不能是结构体数据单元，也不能是布尔数组。如果数据单元不能分块传输，那么对于 S7-300 CPU，这个数据单元就不该超过 8 字节；对于小型 S7-400 CPU，不该超过 16 字节；或者对于 S7-414 及其以上的 CPU，不该超过 32 字节。

SFC 65，“X _ SEND”。写一个数据块到 CPU 的通信缓存器。通信处理器会接收数据，

并把数据发送到目标地址。数据发送到的那个 CPU 需要执行 X _ RCV 来从它的通信缓存接收数据，所以 X _ SEND 和 X _ RCV 都是用于单边通信。X _ SEND 的参数包括：

REQ、CONT、BUSY、RET _ VAL 和 DEST _ ID：如上所述。

REQ _ ID：程序员分配的一个双字，用来指定一个惟一的 X _ SEND 工作。它被发送到接收方的计算机，接受方的计算机就用它来确定数据的源节点。

SD：任意型指针，指向含有将要发送到目标 CPU 的数据的存储区。但它也不能是结构体数据元素或者布尔数组。

SFC 66，“X _ RCV”。在 X _ SEND 指令发送完来自于另一个 S7 CPU 的数据后，从通信缓存中读取数据。通信缓存可以保存多于一个的输入数据块。

■ 只有一个输入参数可用于控制 X _ RCV：

EN _ DT：如果这个位是：

- 关：X _ RCV 使用它的布尔 NDA 输出参数来指出在输入通信缓存器里是否有任何数据块（如果有的话，NMA=1）。
- 开：X _ RCV 从通信缓存器中复制最早写入的数据块。
- 来自 X _ RCV 的输出参数有：

NDA：一个可以有两个功能的位，当执行 X _ RCV 的，它的值由 EN _ DT 的状态决定。如果：

- EN _ DT=0，NDA=1，表示有数据在缓存器里。
- EN _ DT=1，NDA=1，表示已经有一个数据块从缓存器复制到 CPU 的内存。

REQ _ ID：一个惟一的双字，指出缓存器里最早的数据块的字节数，就像发送数据的 X _ SEND 指令分配的那样。

RD：任意型指针，指向来自缓存器的数据将要存放到 CPU 的那个存储区。

RET _ VAL：整型数，包含描述不能接收 X _ RCV 指令的原因的错误代码，或者是缓存器里最早的数据块的大小（如果 EN _ DT=0）或者是刚传送给 CPU 的数据块（如果 EN _ DT=1）的大小（以字节为单位），或者如果缓存器为空，则是 16 进制的数值 7000。

SFC 69，“X _ ABORT”。终止与 X _ GET、X _ PUT 或者 X _ SEND 建立的连接。REQ、BUSY、RET _ VAL 和 DEST _ ID 等参数都需要输入。

SFC 72 和 I _ GET，SFC 73 和 I _ PUT 以及 SFC 74 和 I _ ABORT，分别与 SFC 67 和 X _ GET，SFC 68 和 X _ PUT 以及 SFC 69 和 X _ ABORT 相同，只是除了它们用于读/写数据元素或者放弃与同一个站点的通信伙伴的连接，例如放弃与发起通信的那个 CPU 的连接。它们的参数是一样的，除了 DEST _ ID 的参数用以下参数来代替：

IORD：字节，指出通信伙伴是以外围输入模块（IORD=hex：54）的形式来寻址，还是以一个外围输出（hex：55）来寻址。

LADDR：字，指出模块的外围 I/O 地址数。

4. 用于配置连接的 S7-400 SFB

为了使用任何类型的子网进行通信，S7-400 系列 CPU 预编了标准功能块（SFB）。这些 SFB 要求已经把连接配置好。在第 10 章我们已经讨论过如何为 S7-400 PLC 配置连接。在配置期间，编程软件分配本地 ID 号，有时还分配通信对方的 ID 号。在用户程序里，需要这些 ID 号作为参数来让 SFB 发起通信。

一旦连接配置好，S7-400 CPU 里的用户程序就能调用通信系统 SFB 来与连接好的通信方进行通信[⊖]。交换的数据可以分成较小的数据包来传送。通信 SFB 的参数必须在立即数据块（用户程序可读写立即数据块的变量）。通信 SFB 和它们的参数包括：

■SFB 14，“GET”。从另一个 CPU 单边地复制一批数据（不需要对其他 CPU 编程就能提供数据，所以可以是一个 S7-300 CPU）。参数包括：

- REQ：输入位，当 GET 已经不在程序中运行，用它来启动 GET。与用于无配置通信的 SFC 不同，通信 SFB 的启动是在 SFB 的执行期间 REQ 输入参数从假跳变为真时。
- NDR：输出位，响应 GET 把数据接收到 CPU 的存储器中后，这个位置 1。
- ID：字，存储程序员输入的本地 ID 号。当连接配置好时，自动为连接中的其他 CPU 分配本地 ID 号。
- ADDR_1、ADDR_2、ADDR_3 和 ADDR_4：任意型指针，指向在其他 CPU 中包含将要被复制的数据的存储区。
- RD_1，RD_2，RD_3 和 RD_4：任意型（ANY）指针，指向将被拷贝到 CPU 中的数据所在内存区域。
- ERROR：输出位，当调用 GET 时检测到通信错误，这个位置 1。
- STATUS：输出字，包含描述通信错误的代码。

■ SFB 15，“PUT”。单边地复制一批数据到另一个 CPU（不需要对其他 CPU 编程就能接收数据，所以它可以是 S7-300 CPU）。参数包括有：

- REQ、ID、ERROR 和 STATUS：如前所述的 GET 一样。
- SD_1、SD_2、SD_3 和 SD_4：任意型指针，指向 CPU 中将会复制到其他 CPU 的数据的存储区域。
- ADDR_1、ADDR_2、ADDR_3 和 ADDR_4：任意型指针，指向在其他 CPU 中将要被复制的数据的存储区域。
- DONE：输出位，调用 PUT，当其他 CPU 确认接收到数据后，这个位置 1。

■ SFB 8，“USEND”。复制数据到通信对方的 CPU，不过这个 CPU 必须执行 URCV 才能接收数据。既然必须对通信双方的两个 CPU 编程，那么通信是双边的，而且两个 CPU 都必须与另一个 CPU 有一个配置好的连接。USEND 的参数包括有：

- REQ、ID、DONE、ERROR、STATUS 和 SD_1 到 SD_4：如前所述。
- R_ID：含有用户分配的号码的双字。BRCV 必须含有相同的 R_ID 号，这样通信双方的 CPU 使 USEND 和对应的 URCV 匹配（通过 CP 441 模块连接到网络上的 S5 PLC 使用 R_ID 号来实现不同的操作）。

■ SFB 9，“URCV”。使用 BSEND 来接收数据。BRCV 的参数有：

- ID、R_ID、NDR、ERROR、STATUS、R_ID 和 RD_1 到 RD_4：如前所述。
- EN_R：输入位，置位时，允许接收新的数据。如果在其他 CPU 中再执行一次 USEND，新输入的数据会覆盖之前接收到的数据。要阻止这种情况，就需要在 NDR 为开之后（指出成功完成了一次数据接收），把 EN_R 置为关并保持下来，直到准备好接收新的数据。

⊖ STEP 7 操作手册有时将通信系统功能块称为（FB）。

■ SFB 12，“BSEND”。复制可多达 64k 字节的数据到通信对方的 CPU，对方 CPU 必须执行 BRCV 才能接收数据。因为必须对通信双方的两个 CPU 编程，所以通信是双边的，而且两个 CPU 都必须与另一个 CPU 有一个配置好的连接。USEND 的参数包括：

- REQ、ID、DONE、ERROR、STATUS 和 R_ID：如前所述。
- SD_1：任意型指针，指向 CPU 中含有要发送给其他 CPU 的数据的存储区域。
- LEN：指示有多少数据字节要发送的字。BSEND 自动地把数据拆开到几个小数据包里，等到接收方确认接收到一个数据包后，才发送下一个数据包。
- R：输入位，可把它置为开来取消 BSEND。

■ SFB 13，“BRCV”。使用 BSEND 来接收数据。BRCV 的参数包括：

- EN_R、ID、R_ID、NDR、ERROR、STATUS 和 LEN：如前所述。EN_R 和 NDR 是用来防止接收到的数据被覆盖，正如对 URCV 所述那样。
- RD_1：任意型指针，指向输入数据的存储区。

有三个 SFB 能用于控制一个远程的 S7PLC：

■ SFB 19，“START”。通过一次完全的重启，把 PLC 切换到运行模式。

■ SFB 20，“STOP”。把 PLC 切换到停止模式。

■ SFB 21，“RESUME”。通过重启，把 S7-400 PLC 切换到运行模式。

START、STOP 和 RESUME 都是单边的。它们需要以下的参数：

- REQ、ID、DONE、ERROR 和 STATUS：如前所述。
- PI_NAME：任意型指针，指向含有一个 IEC 采用的设备名的 ASCII 串。如果是 S7 PLC，它的设备名必须是以 P_RROGRAM 开头。
- IOSTATE 和 ARG：字节和任意型指针，它们都应该是空白的。最终，IEC 将会使用它们。

还有两个 SFB 和一个 SFC 可以用来：

■ 从另一个 CPU 中请求状态信息（SFB 22，“STATUS”）。

■ 允许另一个 CPU 报告模式的更改（SFB 23，“USTATUS”）。

■ 请求与另一个 CPU 的连接状态（标准函数：SFC 62，“CONTROL”）。

参考你的操作手册可得到更多细节信息。

SFC 16，“RPINT”。通过点对点连接，向打印机发送 ASCII 字符和格式化代码（参考你的操作手册）。

5. 通过 Profibus-DP 网络的 S7 PLC 通信

配置一个使用 Profibus 子网的 S7 工程，当你配置站点硬件时，你可以选择有 Profibus 通信能力的模块，如第 10 章中所述。在配置 DP 主站期间，你必须为每个 DP 从站分配 4 字节的输入映像和 4 字节的输出映像地址空间。通过这些 I/O 映像地址，使用梯形图的 MOVE 或 STL 的 LOAD 和 TRANSFER 指令，主站能从每个从站中读或写 I/O 地址。

在配置期间，你可能还需要配置 DP 主站和一些 DP 从站，来保持共用接口数据区以供 DP 网络使用。Profibus 系统自动地在主站的接口存储器和从站中的相应接口存储器之间复制数据。这个接口数据区可用于与 DP 主站进行多于 4 个输入字节和 4 个输出字节的数据交换。在 DP 主站中的用户程序，通过读或写主站自身存储器的接口区，调用 SFC 能与每个从

站进行多达 122 字节的数据交换。

SFC 14，“DPRD _ DAT”：从接口存储器中读从站数据。

SFC 15，“DPWR _ DAT”：通过接口存储器写从站数据。

DPRD _ DAT 和 DPWR _ DAT 使用的参数包括有：

LADDR：字，指出接口存储器使用的号码（从而选择了从站）。

RECORD：任意型指针，指向本 CPU 内存中的另一个存储区间，DPRD _ DAT 把数据复制到那个存储器区间，或者 DPWR _ DAT 从那个存储器区间复制数据。

RET _ VAL：整型，含有一个当 SFC 失败时，由 SFC 写的错误代码。

DP 从站必须配置有一个存储器的诊断区。DP 主站使用 SFC 13，“DPNRM _ DG”，能读从站的诊断存储器。活动的 DP 从站能执行它们自己的用户程序，使 DP 主站中断并执行它的硬件中断程序，OB40。从站中的那些用户程序含有对 SFC 7（“DP _ PRAL”）的调用（参考你的操作手册和第 11 章）。

CQM1

13.5 OMRON CQM1 的通信

大多数 CQM1 都有两个串行通信口：外围设备端口和 RS-232C 端口[⊖]。编程器通常是连接到外围端口上的，但这两个端口都可以配置为以下三种模式中的一种来进行通信：

1）在一对一模式下，两个 CQM1 PLC 可配置成共享它们的链接寄存器（LR）的一部分。其中一个 CQM1 配置为主站，它能写数据到共享存储器的低地址部分。另一个 CQM1 就配置为从站并写数据到共享存储器的高地址部分。无论其中一个 PLC 写什么到它的链接存储器的共享部分，都会在发送方的 I/O 扫描期间被发送到另一个 PLC，并在接收方 PLC 的 I/O 扫描期间被接收并写入它的链接存储器中。数据复制过程是全自动的，没有任何命令和配置选项可以加快或者减慢串行数据交换的速度。一对一通信只能通过 RS-232C 端口配置，而不能通过外围设备端口。

2）两个端口都默认是在主机链接模式。编写编程器软件来与使用此模式的 CQM1 进行通信，但它在局域网通信里也同样有用。在主机链接模式下，一台主机计算机（不是 CQM1）通过一个共享的串行链接，能与多达 32 个从站 CQM1 连接（每个 CQM1 配置有一个不同的节点号）。这个主机计算机能向任何一个 CQM1 PLC 发出 C-模式命令。有几个 C-mode 命令可以让主机读或写一个 CQM1 的数据、程序、错误代码或者操作状态存储器的任何部分。CQM1 的响应是完全自动的，因此不受 CQM1 程序的控制。主机链接通信使用 C-mode 命令来允许 CQM1 PLC 在以下两种方式中的一种进行网络互联：

- 对主机计算机编程，可让主机通过读每个 CQM1 的存储器的特定部分，一个接着一个地轮询各个 CQM1 从站。必须对每个 CQM1 从站编程，把通信请求消息存到它的存储器中供主机读取。同时也必须对主机编程，解释来自 CQM1 的通信请求消息，并在需要完成功能请求时执行 C-模式命令（例如从一个 CQM1 复制数据到另一个 CQM1）。当一个请求通信服务完成时，主机也必须通报每个 CQM1 从站，通过写状态信息到 CQM1 从站的存储器中的一个单独区域。本章 S5 PLC 使用的 Siemens L1 网络部分有一个例子说明这样的系统是如何工作的。

⊖ CPU11-E 仅有一个外设端口。

- 对任何 CQM1 从站编程，都能让它利用自己用户程序中的传输指令，TXD（48），传输数据包到主机上。对主机计算机编程来解释来自 CQM1 的数据包，如通信请求数据包，并通过执行请求的一组 C-code 命令来响应请求。因此可对 CQM1 从站编程来发起通信（通过主机计算机），而无需等待轮询的到来。

主机计算机发送给 CQM1 的 C-mode 命令都有标准的格式。当数据传输时，CQM1 从站自动地组织数据集到标准帧格式里。C-mode 帧包含有：

- 一个开始代码（字符@）
- 一个目标节点地址
- 一个编码的命令
- 发送给 CQM1 的数据（连同一些命令）
- 一个错误检测数据字节，然后
- 一个终止码（*后面跟着一个回车）

CQM1 的 C-mode 回复帧也是类似结构的帧。

帧里的每个字符通常都是使用 RS-232C 异步协议，以 ASCII 码格式传输的。当发送帧时，CQM1 自动地把数据从 16 进制数转换成 ASCII 码；当接收帧时，又自动地转换成 16 进制数。

关于主机计算机的编程不在本书的讨论范围内，但 OMRON 表示主机计算机的接口和驱动器可从多个第三方厂商获得。如果你想要自己编写主机计算机程序，OMRON 的用户手册里有关于 C-mode 帧格式的完整描述，包含一组命令代码和预编程的 CQM1 回复帧格式。程序员能够轻易设计一组能由 CQM1 发出的通信请求，并对主机计算机编程，发出适当的 C-code 帧来响应那些请求。

3）在 RS-232C 模式下，用户程序可以执行传输指令，TXD（48），通过串口发起数据的传送；或执行接收指令，RXD（47），让 PLC 读取从串口接收到的数据。消息可以传送到外围设备（例如一个打印机或者另一个 CQM1）或者从外围设备接收（例如条码读卡机和另一个 CQM1）。

CQM1 在它的附加寄存器（AR）存储区（AR 08 到 10）中保持 RS-232C 的通信状态位。有些通信状态位指明消息的传输完成（但不是消息是否被外围设备接收到）。其他的通信状态位则指明 RS-232C 消息已经被接收到通信缓存器里。CQM1 的用户程序必须在执行 TXD（48）或者 RXD（47）指令前，对这些状态位进行检查。

RS-232C 消息可以是固定长度的字符串，也可以是可变长度的字符串。在可变长的消息里，必须对 CQM1 配置，为消息开始和结束设定特定的字节值。

大型的 OMRON PLC，能通过 SYSMAC NET 接口模块互联到高速令牌环网，或者通过 SYSMAC LINK 接口模块互联到高速令牌总线网络，或者使用链接适配器互联到端对端网络，或者通过以太网单元连接到以太网，又或者如这里所述的方法连接到主机链接网络上。链接适配器要求要连接到一个使用 RS-422 协议而不是 RS-232C 的主机链接网络。大型 OMRON 系列 PLC 也可以有远程 I/O 框架，通过在主框架里使用一个 SYSMAC BUS 远程主站模块，在每个远程框架里使用 SYSMAC BUS 远程从站模块。RS-485 协议用于 SYSMAC BUS 远程 I/O 系统。

13.5.1 CQM1 通信通道的配置

CQM1 通信的配置是输入配置数据到 CPU 模块的 RS-232C 端口和外围设备端口的数据存储配置区（DM 6645 到 6654）。这个配置数据只能使用编程器才能改变，但这个更改即使 PLC 是在运行模式下也能进行，而且这个更改一旦配置了就马上生效。

如果 CPU 的 DIP 开关 5 开，则 CQM1 的 RS-232C 端口被硬件配置到主机链接模式，并配置其节点号为 0，以及使用默认的标准 RS-232C 通信协议来传送和接收数据，解释如下。你必须关闭开关 5，否则 DM 6645 到 DM 6649 的设置将不起作用。

为与外围设备通信的 RS-232C 配置端口：

■ DM 6645（用于 RS-232C 端口），

DM 6650（用于外围设备端口）必须包括：

1000 用于一个“标准”异步通信字符协议：7 个数据位，1 个奇偶位，1 个起始位和 2 个停止位，波特率为 9600。

1001 用于使用 DM 6646 或者 DM 6651 选择的专用协议

■ DM 6646（用于 RS-232C 端口），

DM 6651（用于外围设备端口）如果选择专用协议，必须包含以下设置：

NN0P 其中 NN 是选择数据、奇偶位、起始位和停止位的代码集合。P 是选择波特率的代码（参考你的操作手册）。

■ DM 6647（用于 RS-232C 端口），

DM 6652（用于外围设备端口）必须包括：

0000 用于传输的字符间没有延迟时间

NNNN 用于强制给传输的字符间的延时，输入数字 NNNN（0000 到 9999）指出要延时多少个 10 毫秒

■ DM 6648（用于 RS-232C 端口），

DM 6653（用于外围设备端口）必须包括：

0000 用于没有起始和结束代码的字符流

x200 使用一个 CR/LF 字符来指出一个字符流的结束（结束代码）

x100 另一个字节数值（后面有对它的定义），指出字符流的结束（结束代码）

1x00 另一个字节数值（后面有对它的定义），指出字符流的起始（起始代码）

■ DM 6649（用于 RS-232C 端口），

DM 6654（用于外围设备端口）必须包括：

1）结束和起始代码，如果 DM 6648 或者 DM 6653 含有要求程序员指定这两个代码的配置：

NNPP 其中 NN 是从 00 到 FF 的 16 进制数值的结束代码，而 PP 是从 00 到 FF 的 16 进制数值的起始代码

2）一个没有起始或者结束代码的定长数据包的大小，如果 DM 6648 或 DM 6653 不需要起始活和结束代码。

NN00 其中 NN 是从 00 到 FF 的 16 进制数值，指出每个数据包以字节为单位的固定大小（从 0 到 258 字节）。

为与其他计算机的一对一通信配置一个端口。

■ DM 6645（用于 RS-232C 端口），

DM 6650（用于外围设备端口）必须包括：

xN00　其中 N 选择了在这个 CQM1 的链接寄存器有多少存储单元用于与其他 CQM1 共享。

■ 如果 N＝0，使用 LR 00 到 63。

■ 如果 N＝1，使用 LR 00 到 31。

■ 如果 N＝2，使用 LR 00 到 15。

2x00　在通信双方中作为从站。从站只能写数据到共享存储器的高位地址单元（也就是 LR 32-63，16-31 或者 08-15）

3x00　在通信双方中作为主站。主站只能写数据到共享存储器的低位地址单元（也就是 LR 00-31，00-15 或者 00-17）

为局域网内的主机链接通信配置一个端口：

■ DM 6645（用于 RS-232C 端口），

DM 6650（用于外围设备端口）必须包括：

00NN　其中 NN 是主机链接局域网内的 CQM1 的节点号。节点的号码从 00（默认）到 31。

■ DM 6645 到 6647（用于 RS-232C 端口），

DM 6650 到 6652（用于外围设备端口）必须配置成与 RS-232C 相同（如上所述）。

13.5.2　对 CQM1 编程实现通信

当配置完成后，使用一对一通信时，CQM1 程序只需要写数据到链接存储器（LM）的可写部分，方法是使用通常用来写存储器的指令：数据位用 OUT 指令，字用 MOV(21)指令等。用标准指令从链接存储器的其他 PLC 可写部分读取数据：数据位用 AND，字用 CMP(20)等等。记住，写入 PLC 链接存储器中的数据不会即时出现在其他 PLC 的链接存储器里。数据要在两个 PLC 的 I/O 扫描期间才进行传输，因此在一个 PLC 的数据可被另一个 PLC 使用前，可能要延时多达两个扫描循环周期（一个 PLC 的扫描循环，加上另一个 PLC 的扫描循环）。

为了在主机被编程来轮询每个 CQM1 从站的系统中使用主机链接通信，必须对每个 CQM1 编程，让 CQM1 写通信请求到它自己的存储器中，请求要写到主机能找到的地址空间，然后还要周期性地检测通信状态位来了解主机什么时候执行通信服务。特定的请求格式和存储器请求决定对主机编程的方法。

为了在主机被编程为接受由任意 CQM1 发来的数据包并将它们按通信服务的要求解释的系统中使用主机链接通信时，也必须对每个 CQM1 编程，让 CQM1 执行 TRANSMIT 指令，TXD(48)来发送含有请求的数据包给主机计算机。数据包不能超过 122 字节的数据。图 13-10 显示了用来发送数据包给主机计算机的 CQM1 指令。如果为主机链接通信配置好一个端口，CQM1 会自动地把数据包转换成 C-mode 帧格式并带有一个指令代码，指出是从站主动发送的消息，同时也要对主机编程，让它解释消息数据并判断消息是哪个从站发送过来的。无论何时 CQM1 执行 TXD(48)指令通过 RS-232C 端口发送数据，它都会自动地把

AR 0805 的通信状态位设为关，当要传送的数据在通信缓存器中等待发送时通信状态位保持为关。当数据的传输是通过外围设备端口时，AR 0813 的用法也相同。TXD(48)指令应该编程为每当新的数据集合要发送给主机时执行一次，并且只有 AR 0805（或者 AR0813）是开时才发送。

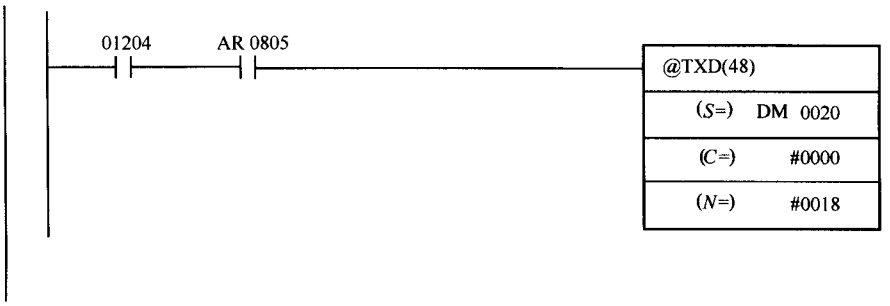


图 13-10 OMRON 的传输一个 18 字节的消息到主机计算机的 CQM1 指令

在图 13-10 的程序里：

1）使用不同形式的 TXD(48)，消息只会发送一次，如果

- 操作位 IR 01204 为开。在梯级上没有显示出来，只要有新的数据集合已经准备好，这个位置位；当 TXD(48)指令开始执行，这个位复位，并且位 AR 0805 置为关。
- AR 0805 为开时，表示外围设备端口还不忙。

2）TXD(48)的第一个参数，S，第一个源字，指出要发送的数据在从 DM0020 开始的数据存储器中。第三个参数，N，字节数的参数，指出 18 字节的数据（从 DM 0020 到 0028 的 9 个字）将要被传输。执行 TXD(48)前，主机可以解释的数据必须存在这 9 个数据存储单元里。

3）TXD（48）的第二个参数，C，控制字，包含有 0000 来指明使用的是 RS-232C 端口（在这里“1000”的意思是使用外围设备端口）。

在为 RS-232C 模式配置的 CQM1 里使用 RS-232C 通信，用户程序必须执行传输指令，TXD(48)，和/或接收指令，RXD(47)。TXD(48)指令使 PLC 复制数据到传输通信缓存器，而 RXD(47)命令用于从接收通信缓存器复制数据。实际串行数据的发送和接收是由 CQM1 CPU 模块中分离的通信操作处理器完成的。TXD(48)指令不会使 PLC 等到数据真正地发送，RXD(47)也不会使 PLC 等到数据真正地接收。通信处理器控制状态位会指出什么时候数据已经接收到通信缓存器中，也会指出什么时候一个请求传输完成了。置位 SR 25209，可复位 RS-232C 端口；置位 SR 25208 可复位外围设备端口。当端口复位完成，SR 25209 和 SR 25208 会自动复位。

RXD（47）使用以下附加寄存器（AR）和状态寄存器（SR）位：

1）AR 0806 置位，当数据已经通过 RS-232C 端口接收完成，并且 AR 09 指出在接收缓存器里有多少字节的数据。当在输入数据里检测到错误，置位 AR 08 的位 0 到位 4 和/或位 7。

2）AR 0814 置位，当数据已经通过外围设备端口接收完成，并且 AR 10 指出接收缓存器里有多少字节的数据。当在输入数据里检测到错误，置位 AR 08 的位 8 到 12 和/或位 5。

因为执行 RXD(47)时，所有这些位都被复位，所以先寻找错误位。

TXD(48)使用以下附加寄存器位：

1）AR 0805 置为关，当 TXD(48)指令为 RS-232C 端口写数据到传输通信缓存器；当数据发送完成，AR 0805 返回置为开。

2）AR 0813 置为关，当 TXD(48)指令为外围设备端口写数据到传输通信缓存器；当数据发送完成，AR 0813 返回置为开。

13.6 故障检修

如果你的 PLC 系统不能像平时一样在系统成员间传送数据，你的系统可能出现了硬件问题、固件（firmware）问题或者软件问题。如果系统不能以你所设想的方式传送数据，你可能没有正确地使用通信特性。

首先应该检查是否为硬件问题，特别是如果系统过去一直能正常通信，但现在出现问题：

- 所有需要的通信组件都启动了吗？这包括了尝试交换数据的控制器、网络的主站，如果有的话（主站应该在其他组件前上电启动），以及通信路径上的任何接口设备。
- 导体都正确地连接到控制器上了吗？总线的配线终止位置正确吗？
- 如果物理层使用一个普通的通信标准（例如 RS-232C 指定了连接器和导体的使用），你就可以购买一个便宜的协议监测器，插到 PLC 和网络其余部分之间，这样可以观察通信线路的状态是否在改变。专有网络的厂商通常提供网络分析器，它能以其他统计性能来报告网络活动。编程软件通常包括一个显示统计量的通道监控窗口，统计量包括尝试建立通信的情况。

固件（软硬件相结合）是当你购买时，软件已经被预编到通信装置里。如果厂商表明你所购买的设备是兼容的，则你就应该不会有韧件的问题。检查所有部件，确定它们都是与同一个版本的网络兼容的，因为网络标准一直在发展和升级中。

预配置端口的使用有一些规则要遵守，如果忽视这些规则有可能导致问题的出现：

- 如果一个通道用于多个用途，这些用途间可能会发生冲突。例如，Allen-Bradley 建议不要在打开了 PII 中断的 PLC-5 模块的底盘里同时执行块传输。如果同时允许 PII 中断和块传输功能，那么当块传输执行的时候，你将得不到中断信号。
- 过多的通信量会减慢所有共享通信通道的传输速度。你的程序不能成功地向通信处理器提交工作（job）吗？有些 PLC 编程软件包附带有通道诊断窗口。检查一下诊断窗口，看看是否有多于你要发送的数据包正在传输，然后检查看看哪些指令的执行导致这种情况。尝试使用计数器，计算那些可疑的指令提交请求的时间。记住，当编程器监控网络上的 PLC 时，它们通常使用共享通信通道，所以编程器可能会发出大量数据请求。

软件问题包括不正确的配置或者用户编程出错。

- 检查你分配给每个网络节点的地址是否惟一，并且要在允许的地址范围内。有些网络系统允许你更改节点地址范围，所以每个节点必须配置成使用相同的范围。你的用户程序是否为尝试进行通信的控制器配置正确的节点地址？当编程器用于更改一个 PLC 的节点地址时，那么这个编程器必须重新为那个新地址的节点建立连接。

- 所有网络节点的通信端口都配置为使用相同的通信协议吗？例如，其中一个节点配置的通信波特率与其他的不同，那么它将不能与其他节点通信。像 Allen-Bradley 的 DF1 网络或者 OMRON 的主机链接网络这样的主-从网络，在如何设置节点这方面有很大的弹性，这也就意味着这些网络有很多潜在的不兼容协议。有时候只是一个节点的协议配置不当，就会在其他所有节点的通信处理器中引起混乱。如果你使用编程器来改变它自己使用的端口协议，那么你必须改变这个编程器的协议来匹配那个新的协议。
- 所有 PLC 都会报告通信错误和错误的原因，这样可以决定什么是可用的，并使用这些报告来帮助确定出现的问题。
- 每个 Allen-Bradley 的通信通道都有一个诊断文件（在配置期间建立的），这个诊断文件保存通道性能的统计数据。Allen-Bradley 消息指令使用消息控制结构体中的状态位和状态文件中的其他状态位来报告消息发送的进度。如果安装了编程器，并使用它来监控消息指令状态，则 Allen-Bradley PLC 也能发送 MSG 指令错误代码到编程器。
- Siemens PLC 保存控制字在通过 L1 网络进行通信的 S5 PLC 中。通信处理器模块使用接口标志或者接口存储器来报告它们的状态。S7 PLC 把所有错误写入诊断缓存器（在第 15 章有更多有关细节的论述）。Profibus-DP 从站保存一个 Profibus 主站能够读取的诊断数据。
- OMRON 的 CQM1 PLC 在存储器的 AR 存储区域中保存通信状态位，把最多 10 个错误消息写入诊断缓存器（可配置为存储 10 个消息）。在第 15 章有更多关于诊断缓存器的详细论述。

对通信性能的不正确运用，可能会导致不合需要的结果。如果数据正在传输（可通过监控接收方 PLC 的存储器来确定），但你的程序似乎使用了错误的数据，那么你的用户程序的定时可能会出现故障。记住，通信处理器不能在用户程序执行指令来分配工作时就马上能完成工作的分配。要确定你的程序按以下顺序来执行下列步骤：

1）写发送的数据到存储器的特定区域，通信处理器将在那里寻找数据来发送，或者从通信处理器存储新数据的存储区（如果你不想丢失旧的数据）复制最近接收到的数据。不要再一次读或写这些存储区，直到按这个顺序执行的所有步骤中的最后一个步骤完成。

2）检查通信处理器的状态，确定通信处理器准备好接受一新的工作。

3）执行发出事件请求给通信处理器的指令。

4）检查通信处理器状态，决定通信处理器什么时候完成一个工作（在某些应用程序中，你可能还需要检查其他的进度报告状态位，例如要确定通信处理器接受了这个工作）。

5）重复以上步骤。

习题

1. 1）解释典型的用于 PLC 局域网的轮询（主-从）、令牌传递和 CSMA/CD（以太）的网络访问系统类型的差别。

2）早期的 PLC 擅长工业控制，部分原因是因为输入采样间隔恒定。哪一种公用网的网络访问方式不允许采样间隔保持恒定？

2. 什么是开放网络协议？
3. 在现代 CPU 模块里能找到的通信处理器是什么？通信处理器如何协调它与主 MPU（运行用户程序）的活动？
4. Allen-Bradley 的 MSG 指令通过 Allen-Bradley DH＋网络发送或接收数据。每个 MSG 指令需要一个_____ _____数据元素来控制消息的传输。这个数据元素包括以下的状态位：EW 位，指出消息是_____；DN 位，当消息已经成功发送，由_____写入的位；ER 位，指出_____。控制元素也包括当输入 MSG 指令时程序员输入的数据，包括有，例如：
   1) _____
   2) _____
   3) _____
5. Siemens 的 L1 网络（Profibus 的早期形式）是主-从网络，系统里每个 PLC 必须在它的存储器里设置一个可由 L1 网络主站读和写的存储区。为什么网络和你的 PLC 程序需要这个存储区？

推荐的 PLC 实验室练习

对 PLC 编程，开关 0 为开时，发送一个 ASCII 消息给另一个 PLC。就你所使用的 PLC，你可能需要使用一个消息指令，一个发送指令，或者可能你需要配置你的 PLC 来让它在每个 PLC 里的邮箱之间交换消息。使用编程器的数据窗口，输入消息到发送方 PLC 的存储器，并在接收方的 PLC 观察存储器的变化。

第 14 章　机器人技术、自动化和 PLC

14.1　学习目标

本章您将了解到：

- 机器人的组成；
- PLC 和机器人控制器的操作系统的不同。区分它们在应用特点上的优点和缺点；
- 描述一个自动化车间使用机器人和 PLC 在制造业流程中如何互相协助工作。

14.2　车间里的机器人和 PLC

PLC 可以应用到各个自动化车间里。事实上，PLC 通常是一个车间的主要控制器。甚至在有机器人的车间里，机器人也只是价格远比它便宜的 PLC 的“奴隶”而已。那为什么不用更昂贵的机器人控制 PLC 呢？（机器人控制器都有数字 I/O 能力，也有类型齐全的模拟 I/O 和通信能力）PLC 是监督控制的最优选择，然而机器人控制器只能用于控制一个单一的活动部件：机器手臂。

当对 PLC 编程，让它在车间里与其他控制器一同运作时，了解各种控制器各自的优点和缺点是很重要的。通常当 PLC 是主要控制器时，最好把一些控制功能编程到其他类型的控制器里。机器人控制器是典型的非 PLC 控制器，所以，在本章我们将探讨如何对 PLC 编程使其与机器人一起工作。

14.3　机器人控制器与 PLC 的不同

当你购买一个完整的伺服机器人（与提放式机器人相对）时，你会看到它通常有三部分：机器人活动部件，有时候叫做手臂，控制器和挂件。

1）机器人的手臂有一组互相关联的传动杆，所以可定位最后的传动杆末端的终端效应器来进行操作。新的机器人通常不包括终端效应器，所以用户必须购买或自己构造一个，以便实现机器人要做的工作。电子机器人的每个传动杆都是由电动机带动，并都含有位置传感器。

2）控制器包括接口电路，这能让它读取手臂的位置传感器并驱动手臂的电动机。控制器也有一个电源为手臂电动机供电，还有一台存储和执行程序来控制机器人的计算机。

3）挂件与控制器相连，并具有使机械手臂的移动更具人性化的控制功能。有些挂件或控制器具有附加显示和操作输入能力，这样就能用于输入机器人程序到控制器的存储器里，但通常是通过个人计算机来编程的，而这时计算机必须运行机器人编程软件和与机器人控制器通信的软件。

到目前为止，从两个方面可以看出机器人与 PLC 的不同：机器人拥有自己的执行器，

并为这些执行器设有内置的电源和接口电路。但还有一个更加重要的区别：机器人只可以执行一系列的顺序动作。机器人能执行的动作包括有：

- 移动手臂。每一次手臂的移动都要求同时驱动所有电动机并监控所有位置传感器。
- 控制执行器而不是手臂的电动机。
- 与车间里其他控制器交换数据。
- 测试传感器的输入信号而不是手臂的位置传感器。
- 通过向前或向后的程序分支来绕过或重复程序段。

尽管机器人能执行的动作和 PLC 能执行的动作没什么两样，但机器人控制器是设计用来执行一些顺序动作的，并且只有前一个动作完成后，才能执行一个新的动作[⊖]。机器人程序可能需要花几分钟到几个小时才能执行一次。另一方面，PLC 每几毫秒执行一次全面的扫描循环，并几乎同时执行完成所有的控制动作。例如，你可以对 PLC 编程，让它在运转着的传送带上检测凸出的部分，并且每当检测到凸出物时就把它从传送带上推下，而不管是在什么时候检测到的（如果传送带运转不是很快的话）。如果用机器人来做同样的工作，凸出物的到来必须是在机器人正在执行顺序程序中的相应步骤时。如果凸出物在任何其他时间到达，机器人将不能把它们推出传送带。

好的，也许你会问，为什么不用 PLC 来控制机器人手臂的运动呢？PLC 能控制电动机和监控位置传感器，也能比机器人控制器更好地监控和控制事先未计划的车间活动，所以为什么还要使用机器人控制器呢？下面是这两个问题的答案：

1）机器人手臂必须通过编程才能执行顺序的移动，并与车间的其他活动同步进行。对 PLC 编程来使机器手臂进行一系列的顺序移动，你需要锁存某些位来保持这些活动的顺序步骤，或者使用顺序器类型的指令（如果 PLC 有顺序器），或者使用一种专为顺序操作而设计的编程语言（就如 IEC 的顺序流程图语言）。用机器人的编程语言编写顺序操作的程序就容易得多，因为从本质上说，它只有在完成一个指令后才执行下一个。

2）机器人控制器是运用机器手臂所提供的一切来控制手臂的最佳选择。机器人控制器的 ROM 存储器里含有协调所有电动机活动的程序，所以所有传动杆都能同时到达它们的目标位置。机器人控制器里的预编 ROM 例行程序可允许程序员指定终端效应器移动的路径，或者指定终端效应器要经由一条由控制器计算中间各点位置的路径移动。机器人控制器的运动控制程序自动地包括伺服控制和位置误差控制的例行程序，因此不需要程序员另行编写。运动控制程序通常允许程序员指定一个可变的工具偏移量，机器人控制器会自动地通过前一个预编的路径，根据移动一个终端效应器的工具中心点的要求，重新计算运动的路径（例如，机器人放下碾磨工具，然后拿起一个漆刷）。内置的例行程序可以控制手臂，使手臂以它可控的最大速度和加速度运动。上面所说的操作都可以编程到 PLC 里，但 PLC 程序会更加复杂和缺乏灵活性，并且手臂会移动的更慢。你可以购买 PLC 运动控制的 I/O 模块，并可获得更高速度和灵活性，但是你不能购买到已经预先配置好来优化任何一个单独机器人性能的 I/O 模块。

⊖ 一些机器人控制器可以通过一些方法改变它们顺序执行的天性。例如在执行一个手臂移动任务时，一些控制器无需等到移动完成就可以执行下一个动作。一些机器人可以编程具有中断性能，所以它们在特定情况下可以中断一个顺序程序而跳到另一个顺序程序。

14.4 机器人控制器与 PLC 的相似点

与 PLC 一样，机器人控制器也是计算机，也具有数字 I/O 能力和通信端口，并且在它们的编程语言里也有使用 I/O 点和通信通道的命令。有些机器人控制器甚至还有附加的模拟 I/O 功能，就像 PLC 一样。

大多数的机器人控制器都能通过编程来解释它们的主程序。它们可以监控 I/O 点，并当输入条件改变时可立即执行中断服务例行程序，就像 PLC 能配置为具有 I/O 中断能力一样。

机器人控制器与 PLC 一样具有有记忆能力的存储器。当它们没有运行程序，甚至断电的情况下，这些存储器仍可存储配置参数、程序和变量（与 PLC 不同，当机器人开始运行程序时，不需要清空存储器。当你通过执行程序重启一个中途断电的机器人时，清空了存储器会导致一些不可意料的后果）。

想要找出更多的相似点，我们可能要考察一下用户不太感兴趣的特性，例如计算机的内部结构或者信号处理能力。实际上，PLC 和机器人控制器之间没有多少功能性的相似点！PLC 和任何一种其他类型的计算机之间也没有多少功能性的相似点。标准的 PLC 操作系统（基于扫描循环）决定了它的不同。

14.5 编程使机器人和 PLC 共同工作

由于机器人控制器和 PLC 之间的差异，使得它们在自动化的车间里成为理想的控制组合[⊖]。你应该分配每个控制器控制它最善于控制的被控过程类型。对机器人控制器编程，可以让它监控影响机器人的顺序操作的环境条件，以及控制与机器人的动作同步的输出。在另一方面，可对 PLC 编程，让 PLC 控制那些不能与机器人的活动同步或者不需要同步的被控过程。PLC 和机器人之间可以交换信号，所以有些 PLC 的操作可以按机器人的要求同步。

在图 14-1 里显示了一个工作车间的实例。这个车间里有一个 PLC 和一个机器人，被设计成完成两个零件的装配，我们称这两个零件为零件 A 和零件 B。下面是这个车间运作的描述（这个车间的例子只是用于学习，所以请暂时忽略它那些明显的缺陷）：

1）A 零件位于车间的左边。在这里操作员把一个零件放到一个滑梭的工作夹具里，把零件传给机器人，然后操作人员按一下按钮，发出滑梭已经装有零件的信号（如果两个滑梭都可用，那么操作员选择哪个是没限定的）。

2）操作员按了按钮后，按钮旁的滑梭就把工作夹具连同 A 零件送到机器人能接触到的地方。有可能两个滑梭同时都在运行。

3）当滑梭接近机器人时，启动一个锁，把 A 零件稳固地固定在一个精确的位置，供后面的装配操作使用。

4）每当重力供给器上的一个传感器检测到供给器上的重量少于 10 个 B 零件时，振动供给器就继续输送 B 零件，直到第二个传感器检测到重力供给器上的重量大于 20 个 B 零件时为止。振动碗是由一个含有 B 零件的送料斗供应零件的。如果送料斗需要重新填满时，送

⊖ 至少一个最近的机器人申明具有包括协处理器的控制器，可以像 PLC 一样执行扫描循环。

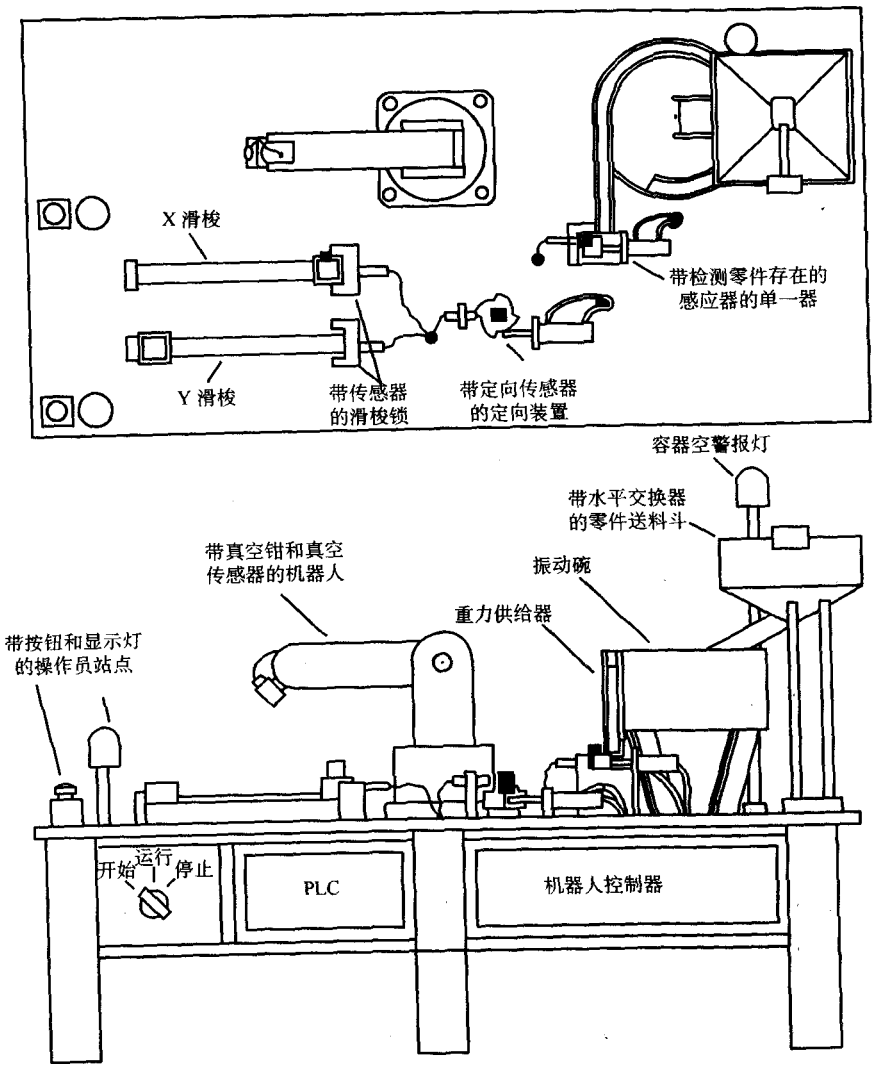


图 14-1 有一个 PLC 和一个机器人的车间

料斗上的警报灯就会点亮。

5）B 零件从重力供给器中滑到底部的一个固定装置里。装置里有一个传感器可以检测到零件。当这个固定装置移到旁边时，落到装置里的 B 零件是惟一的，所以机器人可以不碰到重力供给器上的其他零件就能把 B 零件捡起来。在机器人把 B 零件取出来并且单一器收缩回去之前，不会有其他 B 零件滑落到重力供给器里的固定装置中。

6）机器人利用真空的压力来从单一器中捡起 B 零件。当一个零件没有被捡起或者在以下的操作过程里掉落，终端效应器里的真空传感器会检测到真空的流失。

7）因为 B 零件是对称的，所以振动供给器可以把它们定向在两个方向中的一个。机器人必须把 B 零件放到一个定向装置里，定向装置带有可感应邻近物体的传感器，可以检测到贴在 B 零件的一个面上的金属板。如果 B 零件的放置方向错误，则定向装置会把零件旋转 180 度，然后机器人就能再次捡起这个零件了。

8）然后机器人把 B 零件装配到 A 零件上，该过程中顺序动作如此复杂，没有任何程序员愿意编程两次。在机器人的滑梭末端有两个装配位置，所以可以在机器人程序里使用一个偏移量来允许它在两个位置上以相同的路径执行动作。

9）装配完成后，零件 A 的锁定就会释放，滑梭重新缩回去，并且亮一个灯来通知操作员移动装配工具回去再装另一个 A 零件到滑梭里，开始下一次的循环。

在装卸开始时，应该没有 A 零件在车间里，尽管可能在机器人旁边的滑梭装置里仍有一些之前装卸留下的已装配好的零件，但在重力供给器，振动碗供给器和在顶部右边的送料斗上应该有足够的 B 零件。

1）操作员必须将车间控制开关置于“开始”位置，装着装配好的零件的滑梭会再次返回到操作员的这个位置。

2）然后操作员把车间控制开关置于“运行”位置，并留在那里以控制剩余的装卸。

3）在装卸结束时，操作员必须把车间控制开关置于“停止”位置。直到下一次装配前，滑梭不会再移动到停止位置，但机器人必须完成它旁边的滑梭上仍在等待的所有零件的装配。

有一些操作必须与机器人同步，所以在机器人程序里最好包含有这些同步操作的程序。这些操作有时被称为内部操作，指出它们在顺序生产循环里占用了时间。其他操作应该尽早执行，以避免让机器人等待这些操作的完成。那些可以与顺序生产循环中的操作同时执行的操作，有时候被成为外部操作。外部操作不应该加到生产循环的时间里。想要提高生产率，就应该尽可能多地把操作设计为外部操作。

14.6 机器人程序

首先，让我们看看在机器人程序里必须要有什么。机器人必须：

1）初始化所有变量、配置参数以及需要在转换开始时初始化的输出信号状态。保持型存储器可能含有机器人最后运行的期待以外的状态，特别是当工作循环的操作不是正常结束的时候。

2）移动手臂到靠近单一器的位置，并等待指示装有 B 零件的信号。振动碗、重力供给器以及单一器将以外部操作的方式在 PLC 控制下运转，所以 PLC 必须提供一个 B 零件准备好信号。

3）按以下顺序执行一系列的顺序操作：

a）启动真空发生器；

b）捡起 B 零件；

c）输出一个简短的信号，表示 B 零件已经从单一器中取出；

d）触发一个中断条件，监控真空传感器来检测 B 零件的掉落；

e）移动到定向装置；

f）使中断条件失效；

g）把 B 零件放进定向装置。

因为真空发生器和真空传感器都是作为生产循环的一部分被顺序控制，所以它们必须直接连接到机器人控制器的数字 I/O 连接里。机器人的一个输出需要发送一个 B 零件已取的信号给 PLC。

4）定向装置应该由 PLC 还是机器人来控制？因为机器人必须在定向装置工作的时候进入等待状态，所以它可能也是机器人程序的一部分。感应邻近物体的传感器和定向装置旋转器必须直接连接到机器人的数字 I/O 连接里。机器人程序会：

a）如果金属监测器的信号是开时，跳到步骤 5，或者；

b）把 B 零件旋转 180 度。

5）a）从定位装置里捡起定向的 B 零件。

b）重新启动中断条件，监控真空传感器。

c）移动到靠近滑梭的中间位置。

6）a）如果 PLC 发出信号，指出在靠近机器人的滑梭 X 里已经装有 A 零件，则跳到步骤 8，或者，

b）如果 A 零件在靠近机器人的滑梭 Y 里，则跳到步骤 7，或者，

c）重复步骤 6。

PLC 控制滑梭、锁和滑梭上的指示灯的工作。两个从 PLC 到机器人的信号必须是 X 滑梭准备好或者 Y 滑梭准备好的其中一个。

7）a）如果 Y 位置变量置位，跳到步骤 9；否则，

b）根据同等路径的转换要求，提供一个偏移量，这样在 Y 滑梭上的装配才能够执行。

c）置位 Y 位置的参数。

d）跳到步骤 9。

同等路径在滑梭 X 里有，但滑梭 Y 没有。

8）a）如果 Y 位置变量被复位，跳到步骤 9；否则，

b）移走为 Y 滑梭上的装配而调整同等路径的偏移量，这样在 X 滑梭上的装配才能执行。

c）复位 Y 位置的参数。

9）a）执行装配路径的移动。

b）使中断条件失效。

c）关闭真空发生器，释放 B 零件。

d）如果 Y 位置参数置位，发送简短的 Y 装配完成信号，否则，

e）发送简短的 X 装配完成信号。

X 装配完成和 Y 装配完成信号都是从机器人发送到 PLC。

10）a）如果装卸结束信号为开，并且如果 X 滑梭准备好和 Y 滑梭准备好信号都复位，跳到步骤 11；否则，

b）跳到步骤 12。

PLC 会监控工作站控制开关，并在装卸完成时通知机器人（好的机器人程序会在开始步骤 2 前，先检查装卸结束信号）。

11）停止。

有一块零件从终端效应器上丢失，或者没有被正确地捡起，因此其执行的中断服务例行程序没有在这里展开讨论。主程序必须能够为中断服务例行程序设置不同的变量，通过检测找出机器手臂是在顺序生产线上的哪个步骤，这样程序能返回并重复需要重复的步骤，并调节需要改变的输出。

14.7 PLC 程序

PLC 程序的功能是明显的，因为一开始我们已经精心为车间设计好内部操作程序（机器人程序）。除了一些要按顺序执行的操作，大部分的 PLC 操作都可以认为是同时执行的。在下面的描述里对按顺序执行的步骤进行了编号：

1）如果工作站控制开关是在开始位置，锁存两个滑梭控制输出口，这样滑梭就会缩回到操作员的位置，并把这个程序的主程序部分里所有的滑梭顺序控制位复位。

2）如果工作站控制开关是在“停止”位置，锁存装卸结束信号。

3）如果工作站控制开关刚放到“运行”位置，用一个一次翻转特性来锁存装卸结束信号。

4）如果工作站控制开关不是在“运行”位置，则不执行这个程序的余下部分。

5）执行顺序程序的以下步骤：

a）当操作员按 X 滑梭按钮时，锁存 X 滑梭的输出并移动 X 滑梭到机器人那里，然后打开 X 滑梭输出来把 X 滑梭指示灯点亮。

b）当机器人检测到 X 滑梭时，锁上 X 滑梭的零件锁。

c）零件锁锁上后，锁存 X 滑梭准备好信号到机器人。

d）当接收到机器人的 X 装配完成信号时，打开 X 滑梭锁。

e）当已经打开 X 滑梭锁后，打开 X 滑梭控制，滑梭返回到操作员的位置。

f）当 X 滑梭返回到操作员的位置时，X 滑梭指示灯点亮并保持下来。

6）Y 滑梭的运作要执行另一个顺序程序，并且程序跟 X 滑梭的顺序程序一样。

7）当震动碗里的传感器检测到低位的零件时，点亮警报灯。

8）如果（正常运作）重力供给器上的传感器因检测到少于 10 个零件而关闭的时间超过 2 秒时，锁上重力供给器的输出，然后添加 B 零件。

9）如果（正常运作）重力供给器上的传感器因检测到多于 20 个零件而启动的时间超过 2 秒时，打开重力供给器的输出。

10）执行顺序程序的以下步骤。使用一些可保持的位，这样即使当 PLC 不运行的时候，顺序器位置仍可保持不变。

a）当一个 B 零件到达单一器的夹具位置时，锁上单一器输出来夹住单独的零件。

b）当检测到单一器在前进的位置上，锁存 B 零件准备好信号到机器人上。

c）当接收到机器人的 B 零件已取信号，打开 B 零件准备好信号并打开单一器的输出。

图 14-2 概述了这两个控制程序需要的输入和输出，包括了在 PLC 和机器人控制器之间交换的信号。

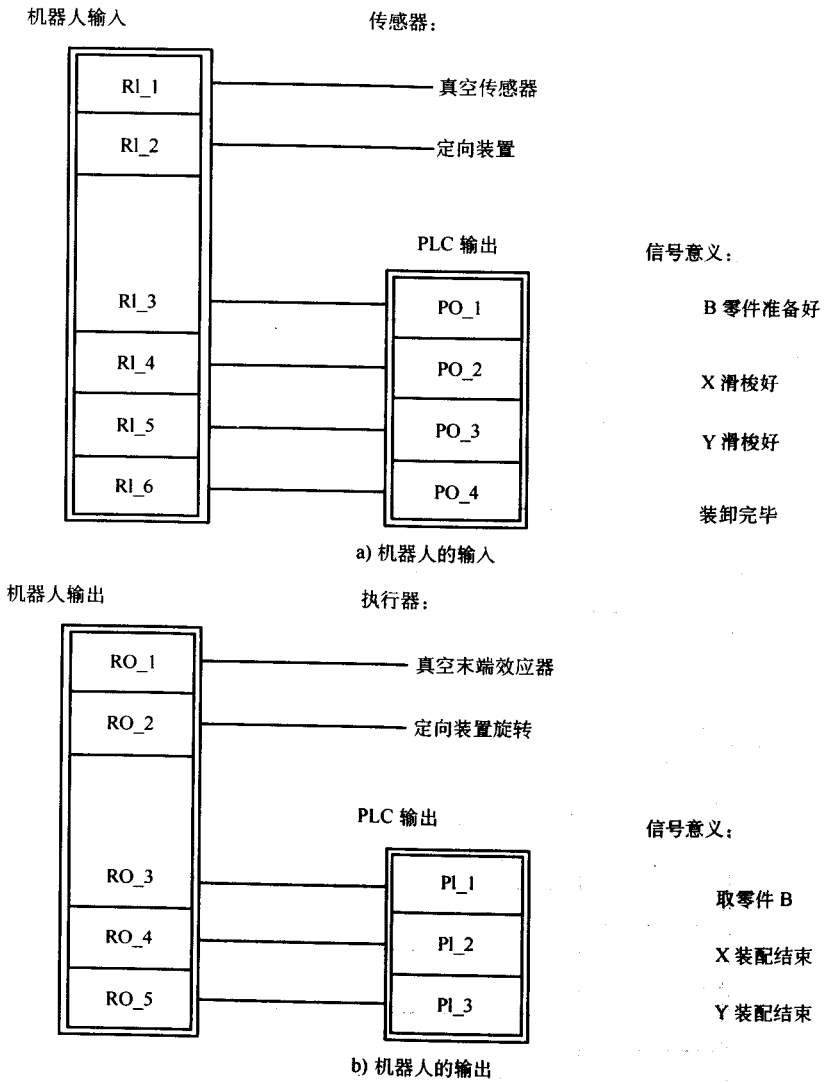
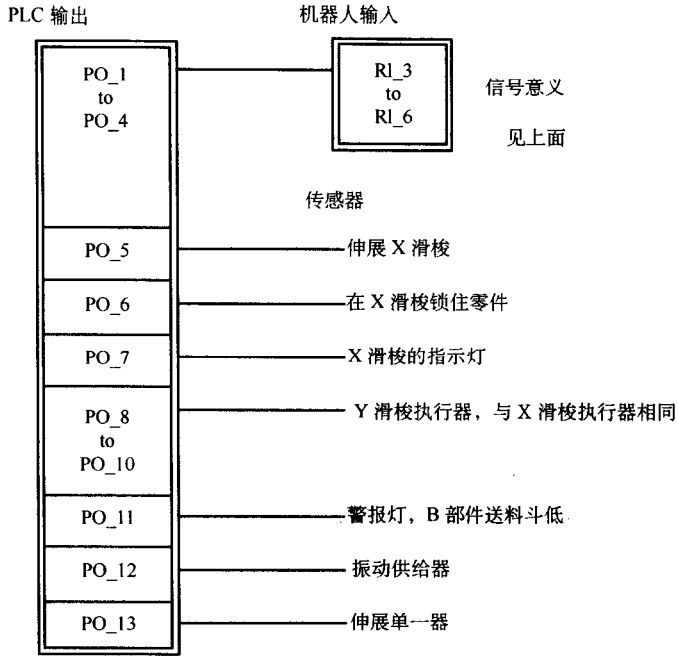

机器人输入
传感器：
RI_1
RI_2
RI_3
RI_4
RI_5
RI_6
真空传感器
定向装置
PLC 输出
PO_1
PO_2
PO_3
PO_4
信号意义：
B 零件准备好
X 滑梭好
Y 滑梭好
装卸完毕
a) 机器人的输入
机器人输出
执行器：
RO_1
RO_2
RO_3
RO_4
RO_5
真空末端效应器
定向装置旋转
PLC 输出
PI_1
PI_2
PI_3
信号意义：
取零件 B
X 装配结束
Y 装配结束
b) 机器人的输出

图 14-2



c) PLC 的输入

图 14-2 （续）



d) PLC 的输出

图 14-2 （续）

第15章 故障检修

15.1 学习目标

本章您将了解到：

- 故障检修的系统方法；
- PLC 外围硬件的故障检修；
- PLC 内部可能导致错误的问题，包括致命错误、非致命错误（包括数学错误），以及编程的逻辑错误；
- Allen-bradley、Siemens 及 OMRON PLC 的错误代码、位、状态信息以及其他出错辨识特征；
- 一些有助于检测不希望行为的原因的指令及编程技巧。

15.2 系统方法

PLC 相关问题可以通过系统方法来解决：

1）确定问题所在；

2）确定正确系统应当如何工作；

3）寻找使系统最优运行的方法；

4）应用最佳的解决方案（回到前面所需的步骤）；

5）验证系统已经被纠错以及系统运行正常。

最早的解决方案往往不是最好的，尤其是对于如今复杂的生产系统来说。PLC 可以帮助你确定问题的所在，但是要做好在 PLC 以外寻找解决方案的准备。自动生产设备中的管理者可以展示许多关于修改非常好的 PLC 程序以改正问题的例子，比如被卡住的滚轴或者是坏掉的电信号线。

使用已经内置于 PLC 中的故障检修功能可以帮助诊断错误。察看 PLC 的外围硬件及接口；然后尝试检查 PLC 本身的硬件和配置；最后再找一找 PLC 用户程序是否有可以修改之处。这些都会找出可能的解决方法。新编写的用户程序往往含有错误，当然新的控制系统往往含有配置不当的组件或者内部连接。

15.3 PLC 外围硬件的故障检修

所有的 PLC 在其 CPU 模块、I/O 模块，有时甚至在电源模块上都有 LED 指示灯。一般情况下，红灯表示有问题存在，绿灯则表示正常。如果 LED 灯在闪烁，通常说明某功能正在执行或者模块正处于等待状态。对 LED 指示灯状态的了解可以节省你大量的故障检修时间。这方面的资料可以通过 PLC 厂商在其产品中附带的 LED 故障检修指导书中查阅得到。

1）如果 PLC 不能进入运行模式，可以用几种不同的方法来确定是软件还是硬件问题。

- 临时输入一个终止循环扫描指令作为用户程序的第一条指令。如果 PLC 可以进入运行模式，则说明问题很可能出在用户程序上，而非硬件问题。（有些 PLC 要求在 PLC 进入运行模式之前清除错误标记位，即使错误已经被改正）
- 重置 PLC 的存储器（仅当已经备份 PLC 存储器内容或者已经不需要这些程序、数据及配置的情况下）。如果 PLC 此时可以进入运行模式，那么问题一般是出在程序中或者配置中或者是存储器损坏。

2）如果你已经启动传感器，并且通过监控输入映像存储器证明你已经操作了传感器，而 PLC 未从传感器中接收到信号，则检查 PLC 输入模块的 LED 灯以确定当某一传感器启动时它们的状态是否改变。

- 如果输入 LED 灯未改变，则使用万用表检查在 PLC 输入模块端的信号是否正在改变状态。如果未改变，则断开 PLC 与传感器之间的连接，单独测试 PLC 的工作。检查是否是外围电源或者直流极性的问题。某些 PLC 直流输入模块是电流灌入型的，所以传感器电路可能会把正极接到 PLC 输入触点。还有一种类型的直流输入模块是电流源型的，所以传感器电路必须把电源的接地端接到输入端口。

如果传感器电路检查完毕，将输入模块用一个仿真模块代替（如果有的话）。如果 PLC 可以识别仿真模块的转换变化，则输入模块确实存在问题。（也有可能你忽略了传感器的电路问题）

- 如果 LED 灯改变状态，则问题很有可能出在程序上。在程序第一行插入一条扫描终止命令，然后运行程序，同时重新监视输入映像，如果此时位随 LED 变化，则问题出现在程序中，一定是在程序写入输入映像标志位的时候改变了它的值所引起的。

如果输入映象标志位仍不随输入模块上的 LED 灯的状态改变而改变，则应该是传感器电路出现了问题。电路的电流负载能力可能不足以改变输入状态，即使它能够改变 LED 灯状态。检查一下输入端口是否有不正常的微小电压变化。

3）如果执行器没有接收到 PLC 准备写入的信号，则观察输出模块的 LED 灯，以确定当 PLC 改变输出状态时它们是否也在改变。

- 如果 LED 灯改变，则使用万用表检查输出模块是否提供了适合于驱动执行器的信号变化以及极性是否正确。如果正确，则断开 PLC 与执行器之间的连接再重新测试一次。（有些输出模块有保险丝，看看是否坏掉了）
- 如果 LED 灯不变，则检查输出电路的电源及它与输出模块之间的配线。（直流输出可以是电流灌入型或电流源型）如果配线正确，断开输出模块与执行器再检查没有驱动执行器的时候 LED 灯是否改变。

用仿真模块替代输出模块（如果有的话），如果 PLC 能够改变仿真的输出状态，则输出模块或配线有问题。

4）如果传感器或者执行器电路工作的话，可以从 PLC 或者 PLC 程序中寻找问题。

15.4 PLC硬件、配置及编程的故障检修

由于PLC系统是由人来启动、配置和编程的，那么主观性的错误难以避免[⊖]。因此PLC生产商们提供了非常全面的工具用以寻找错误。错误可以分为以下几类。

1）致命错误：导致PLC跳出运行模式进入错误模式（与检测到硬件错误时的情况类似）。致命错误可能是由于当PLC起动时，或者是当程序执行时使用组件的过程中，PLC的自检功能检测到错误的PLC组件所引起。一些编程或者配置方面的问题（例如看门狗定时器定时结束，或者试图运行并不存在的程序文件）也会导致致命错误。

当PLC进入出错模式时，LED出错指示灯闪亮，并且所有的输出都将关闭（或者保持在上一次的状态），并且PLC在存储器中保存一个出错代码。这样，程序员可以从出错代码中了解到错误的原因，从而可以解决问题，清除出错代码，然后重启PLC就可以重新进入运行模式。

现代的PLC一般都会保存最近几次出错的详细记录，从而有利于程序员编写在致命错误出现时执行的出错程序。出错程序可以检查出错记录，同时做出相应的反应。在第11章中已经介绍了出错程序。在本章中我们会学习如何寻找和使用错误代码。

2）非致命错误：可以由PLC检测到但不会导致PLC跳出运行模式。一些可以检测到的硬件错误，例如存储器备用电池能量低，会导致非致命错误。它们也可能由配置和程序错误引起（例如，由于优先级较高的程序的运行，或者数学运算在目标存储地址中产生了太大而无法存储的结果，从而推迟定时中断程序的执行）。

非致命错误导致PLC将错误标志位置位，或者将错误代码写进存储器中。当结果字太大或者太小而不能保存时，数据字操作指令在存储器中置位数学标志。用户程序必须检查这些标志位或者代码并且尽可能做出反应，因为如果没有检测到错误，PLC便会继续执行用户程序。一些非致命错误标志位在前面的章节中也有讨论，本章中我们将涉及其他一些类型。

3）编程或配置逻辑错误：PLC不能自动检测到，但是可以在程序中或者在编程器的程序监控特性中使用故障检修指令检测到。

逻辑错误的例子包括：用户程序将原来应该开启的位关闭，用户程序的两个不同段争取同一输出的控制权，或者结构化程序绕过了一个原本必需的程序段。

早期PLC的设计中，不允许类似于同时有两个控制相同输出的梯级或者在同一程序中有向后跳转的逻辑错误出现，但是用户需要更多的灵活性，所以那些安全措施被去掉了。当程序员写程序存在明显的潜在逻辑错误时，一些编程器提供报警信息，但是程序员可以忽略该报警。

编程语言也包含简单的调试工具：可以提前终止扫描循环从而使程序一次只检查一个程序段的指令；可以在特殊情况下产生致命或者非致命错误从而即时终止PLC工作的指令；以及其他一些特别为调试程序而设计的指令。标准指令，例如计数器指令，可以临时插入程序中记录可能丢失的事件。编程软件也包含一些调试工具：当用户程序执行时可用于监视和

⊖ 即使系统设计得再完美，这个世界总有更蠢的人能把事情弄砸。

改变数据存储器；强制 I/O 映像位开或关以观察程序如何做出反应；用特定标志位或者标志字在很短间隔内记录程序变化，从而在直方图表中表示出来；对程序中所有使用了特殊位或字的位置创建 X-ref（前后参照）列表；同时还有一个搜索工具，可以找出在特定地址或者指令中程序的每一步动作。

15.5 ALLEN-BRADLEY PLC-5 的故障检修

PLC-5

15.5.1 PLC-5 的硬件故障检修

Allen-Bradley 操作手册会告诉你如何理解 CPU 模块和 I/O 模块上的 LED 灯的指示状态。总的来说，绿灯表示正常，红灯表示出错，一直闪烁的灯表示未完成操作。例如，一个智能 I/O 模块在它开始工作前需要将一批配置或者数据传送，那么它会使绿灯闪烁。（也可能表示有编程问题）

一些硬件问题可能很难被检测到，包括：

1）当框架出错时可以选择信号值（最小、最大或者在输出范围中）的远程 I/O 框架模拟输出模块中的 DIP 开关设置，与一些底座的 DIP 开关设置发生冲突。从模拟输出模块上断开执行器，导致一个框架出错并测量模拟输出，验证模拟输出信号是否做了我们需要它们做的动作。

2）模拟输入被模块中相邻通道中的信号所影响。在输出信号中选择电压或者电流通道。减少无用通道被连接到常用通道而导致问题的可能性。

3）牢记模拟输入和输出模块偶尔需要校准。

15.5.2 PLC-5 启动出错检测的配置

通过对 PLC 启动特性的配置可以了解 PLC-5 系列如何响应不同类型的错误。在第 10 章和第 11 章中我们已经讨论过。我们了解到：

1）当主要出错导致 PLC-5 停止工作时，PLC 不会返回运行状态，直到在 S：11 的主要出错位被操作者用编程器或者出错程序清除为止。

2）如果将 S：26/1 置位，则 PLC-5 会将运行时发生断电视为一个主要出错，所以当电源恢复时 PLC 不会直接进入运行模式。它会寻找并执行出错程序，并且只有当该出错程序清除了主要出错标志位的时候才会重新返回到运行模式。

3）可以将 PCL-5 配置成总是在第一步（默认）或者在当 PLC 停止运行时（如果将 S：26/0置位）并未执行完的那一步重启它的 SFC 程序（如果它配置有 SFC 程序）。

15.5.3 PLC-5 硬件状态

I/O 状态 全局状态及状态文件中的框架控制字表明了 I/O 框架的状态。对于 24 个框架需要六个字。当你配置 PLC-5 的远程 I/O 扫描通道时，须指定一个整型文件（例如，N9）被作为 I/O 状态文件来存储另外的框架状态（S：16 会包含你输入的整型文件编号）。I/O 状态文件包括为每个 I/O 框架准备的两个状态字，即如果有 24 个框架，则会包含 48 个字。全局状态、框架控制以及 I/O 状态字作用如下：

1. 指示框架或者模块出错

全局状态字（S：7、S：32 和 S：34）在低字节包含了一些当检测到框架出错时置位的标志位。每一位表示一个满框架，同时最低位的地址代表了最低（八进制）编号的框架（例如 S：7/0 至 S：7/7 代表框架 00 至 07，S：32/0 代表框架 10，等等）。举例说明，如果框架 03 出错，S：7 会包含如下二进制模式：

S：7 ？？？？？？？？00001000

I/O 状态文件字包括两个状态字中第一个的低四位，当出错发生在一个框架的四分之一处（例如位 0 代表组 0 和组 1，位 1 代表组 2 和组 3）时会被置位。举例说明，如果 I/O 状态文件标号为 N9，则 I/O 状态文件起始位在字 N9：0，字 N9：6 和 N9：7 是代表框架 03 的状态字。如果因为组 2 中的一个模块出错而导致框架 03 出错，则 N9：6 和 N9：7 则会包含如下模式：

N9：6 ？？？？？？？？00000010

N9：7 ？？？？？？？？？？？？？？？？

低字节的高位（本例中的 N9：6/4 及 N9：6/7）始终为零。

当框架出错时，它不能进入 I/O 扫描，同时：

1）通过设置底板上的 DIP 开关（复位或者保持）可以将框架的输出按指定状态输出；同时也可以通过 I/O 模块上的 DIP 开关设置来规定框架的输出（例如：最大/最小以及模拟范围中的值等）。

2）输入映像保持在它们上一次的状态。用户程序可以将缺省值写入输入映像的出错框架中，如果它们的状态会危及到得到控制的系统。

2. 指示框架的块传送要求队列已满

如果对某一个框架有足够的块传送要求来填满该框架的块传送要求队列时，每个全局状态字（S：7，S：32 及 S：34）的高 8 位置位。每个位代表一个框架，最低地址代表了最低的框架号（例如 S：7/8 代表框架 00 至 07，S：32/8 代表框架 10）。举例说明，如果对于框架 03 及 04 存在足够多的块传送要求，S：7 则会包含如下模式：

S：7 00011000？？？？？？？？

3. 指示框架中 I/O 模块的存在

I/O 状态文件包含了框架的第一个状态字的高字节的低四位。如果一个框架的四分之一包含了 I/O 模块，将它们置位（例如，位 8 代表了组 0 和组 1，位 9 代表了组 2 和组 3）。举例说明，如果框架 03 在组 6 与组 7 中无任何 I/O 模块，描述框架 03 的状态字则会包含如下模式：

N9：6 00000111？？？？？？？？

N9：7 ？？？？？？？？？？？？？？？？

高字节的高位（本例中的 N9：6/12 至 N9：6/15）始终为零。

4. 禁止框架中的框架或模块

每个框架控制字（S：27，S：33 及 S：35）包含了 8 个低位，可以由用户程序（或者编程器的操作）置位以禁止框架，冻结它的输出状态及框架的输入映像。每位代表了一个框架（例如 S：27/0 至 S：27/7 代表了框架 00 至 07，S：33/0 代表了框架 10）。举例说明，如果想要禁止框架 02 及 03，S：27 必须包含如下模式：

S：27 ？？？？？？？？00001100

第二个 I/O 状态文件字的四个低位可以置位以禁止同一框架的两个模块（例如，位 0 禁止组 0 和组 1，位 2 禁止组 2 和组 3）。举例说明：如果仅打算禁止框架 03 中的组 0 和组 1，组 4 和组 5，则 N9：6 与 N9：7 必须包含如下模式：

N9：6 ？？？？？？？？？？？？？？？？

N9：7 ？？？？？？？？00000101

低字节的高位（本例中的 N9：7/4 至 N9：7/7）始终为零。

5. 复位框架中的框架或模块

每个框架控制字（S：27，S：33 及 S：35）的高 8 位，可以由用户程序（或者编程器的操作）置位以复位框架，将其输出关闭（或者进入默认状态）直到扫描循环写入新的数值为止。每位代表一个框架（例如 S：27/7 至 S：27/15 代表了框架 00 至 07，S：33/7 代表了框架 10）。举例说明，如果想要重置框架 00 及 03，S：27 必须包含如下模式：

S：27 00001001？？？？？？？？

第二个 I/O 状态文件字高字节的低四位，可以被置位以复位同一框架的两个模块（例如，位 8 复位组 0 和组 1，位 9 复位组 2 和组 3）。举例说明：如果仅打算复位框架 03 中的组 0 和组 1，组 4 和组 5，则 N9：6 与 N9：7 必须包含如下模式：

N9：6 ？？？？？？？？？？？？？？？？

N9：7 00000101？？？？？？？？

高字节的高位（本例中的 N9：7/12 至 N9：7/15）始终为零。

15.5.4 PLC-5 通信通道状态

配置 PLC-5 通信通道时，指定一个整数文件作为诊断文件，用来保存与通信通道类型相关的状态信息（诊断文件号保存在状态文件中）。不必解释原始数据形式的诊断文件的内容，也不需要编写程序来改变文件内容。使用编程器的通道状态显示功能来显示通道诊断文件的内容，并带有描述每一个数值含义的标签。在配置一个 DH＋网络通道时，也可以指定一个整数文件用作全局状态文件的标志文件。DH＋WHO 动态屏幕显示了全局状态文件标志数据，告诉我们哪一个 DH＋节点是激活的或者为什么它们是没有激活的，而且同时当你选择 DH＋WHO 诊断屏幕时也显示出 DH＋诊断文件数据的信息。以太网 WHO 屏幕同时也可以显示以太网络状态及诊断。

如果想要使程序检查 DH＋节点是否被激活，状态字 S：3 至 S：6 包含了独立标志位，当相应的 DH＋节点激活时该位置 1。举例说明，如果 DH＋节点 4 激活，则 S：3/4 置 1，同时如果最高位节点，节点 63（八进制的 77）被激活，则 S：6/15 置 1。

15.5.5 PLC-5 的 CPU 状态

PLC-5 状态文件包含了各种状态信息，包括：

1）处理器状态标志位（在 S：1 中），用以描述 CPU 的当前操作状态。

2）CPU 上的 DIP 开关的位置（在 S：2 中）。

3）最近一次的扫描时间（在 S：8 中），最大扫描时间（在 S：9 中），独立 MCP，PII 以及扩展本地 I/O 扫描（如果一个扩展的本地 I/O 底盘通过通道 2 相连）的扫描次数。

4）为当前处理器文件而计算得到的校验和（在 S：57 中），用以检测存储器坏区。

5）出错信息（下一部分会讨论）。

6）配置及中断程序状态。

大部分与操作人员相关的故障检修的 PLC 状态都可以通过处理器状态屏幕了解，除了一些更详细的通道状态信息。

PLC-5 SLC 500

15.5.6 Allen-Bradley 的主要出错及次要出错

Allen-Bradley PLC-5 和 SLC 500 对于主要及次要出错的响应是相同的，所以本节把两种 PLC 放在一起讨论以避免重复。状态信息在 SLC 500 及 PLC-5 中都可以获取，用以帮助程序员检测出错的 PLC 的故障。恢复错误的出错程序的使用也会被讨论。关于 Allen-Bradley 系列 PLC 如何响应出错的更详细的叙述已在第 11 章中的出错程序中断章节出现。

1. 出错位及代码

主要出错会导致 Allen-Bradley PLC 跳出运行模式。某些主要出错可以恢复，这就意味着它们可以检测到。PLC 会在 S：29 中寻找出错程序文件号，同时当错误被检测到的时候执行该出错程序。出错程序由用户写入，可以包含清除主要出错标志位的指令。如果出错程序在运行结束之前清除了主要出错的标志位，PLC 就会返回运行模式继续运行在检测到出错之前所执行的程序。可恢复的主要出错通常是由于用户程序内容引起的，所以 Allen-Bradley 的文献通常将它们列为可恢复的主要用户出错。

次要出错一般不会导致 Allen-Bradley PLC 跳出运行模式，尽管某些 SLC 500 的次要出错位（在 S：5 的低 8 位）在 SLC 500 结束执行程序扫描时没有被清除（或者当用户程序执行一个 REF 指令时），也会触发一个主要出错。任何用户程序可以包括监视次要出错位以及纠正次要出错的指令，甚至包含可能当检测到次要出错时也会引发主要出错的指令。算术标志位用以检测可能导致控制问题的次要算术错误。如果操作数字的指令检测到进位或者溢出，和/或显示出结果为零或者为负的时候，数学标志位置位。次要出错位和算术标志位会自身清零。

出错位、代码以及数学标志位会出现在：

1）PLC-5 中：

S：23 包括主要出错位（见附录 H）；

S：12 包括主要出错代码（见附录 H）；

S：13 包括主要出错发生时正在执行的程序文件编号；

S：14 包括主要出错发生时正在执行的梯级编号；

S：10 以及 S：17 包括了次要出错位（见附录 H）；

S：0 包括算术标志位（见第 6 章）。

2）SLC 500 中：

S：1/13 包括惟一的主要出错位；

S：6 包括主要出错代码（见附录 I）；

S：20 包括主要出错发生时正在执行的程序文件编号；

S：21 包括主要出错发生时正在执行的梯级编号；

S：0 包括数学标志位（见第 6 章）；

S：5 包括次要出错位：

■ 如果在扫描循环的用户程序结束前未被程序指令清除，S：5 的低 8 位在某些特殊情况下锁定并成为主要出错。

S：5/0 置位，当检测到数学溢出时（如果 S：0/1 开）；

S：5/2 置位，当用于 LIFO、FILO、位转换或者顺序器指令的控制元素的错误位开启时；

S：5/3 置位，出错程序执行过程中发生新的主要出错时。S：6 中的出错代码记录了最近一次的主要出错；

S：5/4 置位，当程序指令试图使用已经失效的模块存储器（M0/M1 地址空间）中的数据时。

（S：5 的其他低位此时不起作用）

■ S：5 的其他次要出错位不会锁定为开，也不会产生主要出错。

S：5/8 被锁定，当上电时存储器模块中的内容被传送到 RAM（如配置中所要求）；

S：5/9 或者 S：5/13 置位，当上电时存储器模块中的内容由于种种原因（见使用手册）没有被传送到 RAM（如果配置要求如此）；

S：5/10 或者 S：5/12 置位，当 STI 或者 DII 中断程序丢失时（见第 11 章）；

S：5/11 置位，当电池容量低时；

S：5/14 置位，当检测到与通道 0 相连的适配器有问题时（见使用手册）；

S：5/15 置位，当 ASCII 码指令尝试操作一个长度大于允许上限 82 字符的字符串时。

PLC 出错后，检查主要出错位及错误代码以确定错误的原因。可以使用编程器来显示包含错误位的状态文件字中的内容。编程器同时也会显示出描述为何每个出错位被置位的文本消息。附录 H 和 I 包含了主要出错位、次要出错位的列表，以及 SLC 500 与 PLC-5 能够检测到的出错代码列表；同时还包括了状态文件地址，以方便用户或者出错程序寻找。在纠正了主要出错的原因之后，（如果 PLC 已经被设置成将自身的主要出错位清零），操作员可以通过旋转状态控制开关至编程模式，然后至运行模式，从而将 PLC 置回运行状态。操作员必须使用编程终端清除在 PLC-5、REM 运行模式的 SLC 500 中或者在未设置成清除自身主要出错位的 SLC 500 中的主要出错。清除了主要出错位后，PLC 可以重新进入运行模式。

2. 看门狗定时器

当看门狗定时器计时结束时程序会出错。可以显示看门狗定时器的当前设置，并且可以与最大、平均及最近扫描次数做比较。当前时间可以通过编程器显示，也可以在用户程序中读和写。

1）在 PLC-5 中：

■ S：28 包含了看门狗定时器的设置；

■ S：8 和 S：9 分别包含了最近以及最大的扫描次数。

2）在 SCL 500 中：

■ S：3H（S：3 的高字节）包含了看门狗定时器的设置；

■ S：22、S：23 及 S：35 分别包含了最大、平均及最近的扫描次数。

3. I/O 模块中保持的 I/O 状态

I/O 模块可能包含指示出错情况的状态位：

1）在 PLC-5 中，智能（BT）I/O 模块状态包含在能够被块传送读（BTR）指令读的数据中。如果用户程序已经执行 BTR 指令来读 I/O 模块的状态，那么编程器中的 I/O 主菜单便可以用来完全观察到带有描述性标记的状态信息。参考 BT 模块手册中关于状态信息的 BT 文件地址的内容，如果你想编写用户程序来监视 I/O 模块状态的话。

2）在 SLC 500 中，状态位可以从模块 M0 及 M1 存储器中读出。参考使用手册中介绍相关的专门 I/O 模块的状态信息的可用性及地址情况。

现在继续介绍 PLC-5 的故障检修。

PLC-5

15.5.7 PLC-5 程序故障检修

一部分 PLC-5 指令使用控制元素。控制元素包括状态位及在程序调试时不赋值的状态字。一部分控制元素状态位或者状态字会在你监控程序时显示在梯形图程序显示器上（例如：EN，EN，ER，LEN 等）。其他的状态信息则可以通过数据屏幕或者用户程序中的指令来监视，有时也可以由它们来改变。

除去一些被称为控制元素的元素（例如 R6：0）外，一部分指令还包含了嵌入其他用以控制的数据结构体中的控制位及控制字。它们包括了定时器及计数器指令(例如 T4：0，C5：0)、块传送指令（例如 BT9：0)、信息指令（例如 MG10：0)、PID 指令（例如 PD11：0）以及 SFC 程序控制的结构体（例如 SC：12：0)。

某些指令的使用主要是为了程序调试目的。它们包含了下面介绍的一些指令。

1. 始终为非（AFI）

如果在梯形图梯级中插入始终为非（AFI）指令作为第一个元素，那么它会将梯级的布尔运算结果置为非，关闭梯级的输出。

2. 暂时终止（TND）

暂时终止（TND）是一个输出元素。可以有条件或者无条件地使用。当它执行的时候，会立即终止当前的扫描循环，即使它是出现在子程序或者中断中。执行 I/O 扫描并开始运行下一个扫描循环。

3. 数据传送（DTR）

数据传送（DTR）指令用于将单一的源数据字（或者源字中被选出的位）与包含期望位值的参考字做比较。通过使用比较指令，可以被用作布尔逻辑元素。除了以下情况：

1）DTR 仅当两个字（或者部分字）彼此不相等时为真。

2）当两个值不相等时，DTR 将源字中的值复制到参考字中。

3）必须输入一个掩模（作为十六进制值或者地址)。如果掩模中包含 0 的位，源字与参考字中的对应位不能比较。

DTR 指令可以用作一种一次翻转法，每次当一个变化的源字值被检测到的时候扫描一次，且值为真。

4. 文件位比较（FBC）

文件位比较（FBC）指令如图 15-1 所示。在程序中使用该命令用以比较一组数据字与一组期望数据值，同时每次 DTR 指令的控制逻辑从非变为真时，记录下每一位不相同的数

字位。为便于学习，虚线部分用以区别三组中的指令参数（真实情况中没有虚线）。

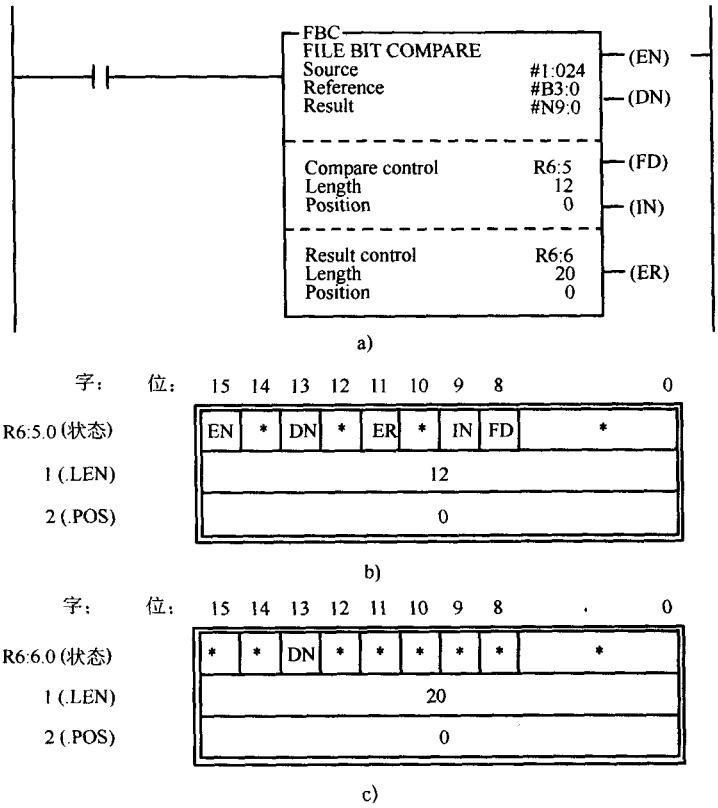


图 15-1 a）PLC-5 文件位比较指令及其使用的两个控制元素；
b）比较控制元素；c）结果控制元素

在第一组 FBC 参数中，输入三个地址，每一个前面标上 # 号以表示它们是这三个文件的起始地址。（记住，# 在 PLC-5 用作变址寻址）。三个地址显示了以下文件的开始地址：

1）源文件：包含拿来与期望值比较的数值，从第一个字的最低位开始计算。为了诊断目的，这个文件可能是一组输入或者输出映像。（在本例中，起始位 I：024/0）

2）参考文件：包含期望的数据值。可能是程序员在执行程序前输入位存储器的模式。（在本例中，起始位 B3：0.0）

3）结果文件：FBC 指令会将不相匹配的位的八进制地址写入其中。例如，如果位 1、8、9 和 17 不匹配，则八进制数 0、7、10、20 会被存入结果文件的前四个字中。（在本例中，结果文件始于 N9：0）

第二组比较参数值指示有多少位需要比较，同时提供一个比较控制元素，用于追踪在比较序列中 PLC-5 的位置。通常控制元素的地址是可读、可写并可以由程序或者程序员监控的，包括 .EN（启用）、.DN（完成）状态位及 .LEN（长度）值。在 FBC 指令中某些比较控制元素的功能有非常特殊的用处：

1）.IN（禁止）位可以被置位（由程序员或者编程指令）来控制 FBC 指令，每次当 FBC 控制逻辑从非变为真时仅仅寻找一个不匹配之处。当找到不匹配之处时，比较控制元

素的 .FD（找到）位置位，同时不匹配位的位置信息也被存入控制元素的 .POS（位置）字中，并在 FBC 指令框图中显示。注意：以十进制存储，而非八进制！当 FBC 指令的控制逻辑再次由非变为真时，FBC 指令继续从刚才停止的地方开始比较，直到找到下一个不匹配。当所有的比较都完成后，FBC 指令不会重新比较另一个组，直到另外一个逻辑跳变发生，即使没有发现任何不匹配。

2）如果 .IN 位没有被置位，则 FBC 指令会将所有的源文件位与参考文件比较，将不匹配记录在结果文件中，并且将比较控制元素的 .FD 位置位（只要找到一个不匹配即可）。

第三组结果参数是用来指定结果文件的长度，并让 PLC-5 了解有多少不匹配被记录在结果文件中的结果控制元素。如果有太多不匹配而超出存储范围，结果控制元素中的 .DN 位会置位，而最近的不匹配位地址也会被存储在第一个结果文件字中，同时比较过程继续。当比较控制元素的 .DN 位置位，并且指令的控制逻辑从非变为真时，结果控制元素复位，这样每次把新的不匹配地址存入第一个结果文件字时都会复位。

5. 诊断检测（DDT）

诊断检测指令看上去像 FBC 指令。除了每次 DDT 指令在源文件字与参考文件字中寻找到一个不匹配后，DDT 指令在记录不匹配信息的时候会将参考字的值改变以外，它与 FBC 的工作方式基本相同。因此 DDT 指令更适合于检测位值中的变化。

6. 由用户产生的主要出错使用跳转到子程序（JSR）指令

在 11 章中，讲述了如果使用跳转到子程序指令去产生一个由用户产生的主要出错。总的来说，如果在 JSR 指令中指定了出错程序文件号作为子程序文件号，同时输入一个数字作为单输入参数，那么出错程序执行时就好像在响应一个主要出错（而且必须清除已经置位的 S：11/7 主要出错位，否则 PLC 会停止工作），而不是仅仅将出错程序视为子程序一样跳转。作为输入参数的数值会被放入主要出错代码字中（S：12）。

15.5.8 PLC-5 编程器在调试中的特性

Allen-Bradley 的编程软件提供常规的程序以及数据监控特性、数据修改特性、I/O 强制特性以及搜索和交叉参考特性。也可以使用 PLC-5 的编程器来生成一个直方图，用以显示数值随时间的改变：指定任何位或者数值（在本章中不考虑浮点运算数值），编程器会对 PLC-5 的存储器进行采样，同时将数值传送次数、新数值以及总的监控次数都显示出来。同时也可以指定掩模，这样仅有一部分数值位处于监控状态，其余的都被忽略。

SLC 500 15.6 Allen-Bradley SLC 500 的故障检修

15.6.1 SLC 500 硬件故障检修

参见 PCL-5 硬件故障检修相关章节。

15.6.2 配置 SLC 500 启动时的出错检测

在第 10 章中已经讨论过如何配置 SLC 500 的启动特性，第 11 章中也叙述过 SLC 500 如何响应出错中断。从中可以了解到：

1）当断电且 S：1/8 置位且 PLC 处于 REM 运行模式时，一旦电源恢复工作，SLC 500

清除其主要出错位（S：1/13）及次要出错位（S：5/0 至 S：5/7）。否则，SLC 500 将进入错误模式。

2）当断电且 S：1/8 未置位时，若 S：1/9 置位，SLC 500 同样也可以在重启后进入 REM 运行模式。PLC 会首先寻找并执行出错程序。如果出错程序清除了主要出错位，则 PLC 会返回 REM 运行模式。

15.6.3 SLC 500 硬件状态

1. I/O 状态

在 SLC 500 中，I/O 模块错误是致命错误，因此 S：1/13 会置位（除非出错程序执行并清除了 S：1/13 位），同时 PLC 停止工作。一个错误代码会写入 S：6 中。错误代码中的高位包括了出错的模块号，而低位则代表了错误类型。（代码见附录 I）I/O 插槽也可以通过清除 S：11 或 S：12 中的位而单独被禁止工作（冻结它们的输出以及输入映像表）。这些位在 PLC 启动时自动置位，同时启动所有的 I/O 模块。

2. 通信通道状态

SLC 500 不会将通道统计表载入诊断文件中。通信通道状态仅通过检查活动节点状态字来观测。

S：9 和 S：10 代表与通道 1 相连的 DH485 网络

S：67 和 S：68 代表与通道 0 相连的 DH485 网络

S：83 和 S：86 代表与通道 1 相连的 DH＋网络

每一个置位的位表示了一个正在网络中参与传送标记符的节点，即使该节点并未传送数据。节点 0 由最低位（S：9/0、S：67/0 或者 S：83/0）来表示，而最高位表示相对较高地址的节点（由 DH485 在 S：10/15 或者 S：68/15 的节点 31，或者 DH＋在 S：86/15 的节点 63）。也可以使用编程器通过选择 WHO 表或者 DH485 WHO 和 DH＋ WHO 中的工作站诊断来显示网络或者节点的动作。

3. CPU 状态

如果不能直接看见 SLC 500，但是可以通过本地局域网的编程器连接，也可以确定究竟使用的是哪一种型号的 SLC 500。SLC 500 在状态字 S：58 到 S：65 中保存了运行模式、存储器以及它所包含的用户程序的描述信息。

大多数与使用者有关的故障检修的 PLC 状态都可以通过处理器状态或者状态文件数据屏幕获得，一些更详细的通道状态信息除外。

15.6.4 SLC 500 主要及次要出错

相关内容见 Allen-BradleyPLC-5 故障检修章节中关于 PLC-5 主要及次要出错部分。

15.6.5 SLC 500 程序故障检修

某些指令规定在调试用户程序的过程中使用，包括：

1. 临时终止（TND）

当 SLC 500 执行一个临时终止（TND）指令时（可以作为布尔逻辑控制的输出单元），即使遇到子程序或者中断，它仍然可以终止当前扫描循环。PLC 会继续执行 I/O 扫描的步骤，重新开始运行用户程序。

2. 延缓执行（SUS）

如图 15-2 所示，如果延缓执行（SUS）指令被执行，PLC 会立即终止扫描循环。终止不是主要出错，尽管处理器状态检测表明发生了一个主要出错。当编写 SUS 指令时，必须键入延缓 ID 码（本例中使用 91）。SUS 指令执行时将延缓 ID 码写入 S：7 中。程序员可以在程序中使用若干个 SUS 指令，每一个都必须有不同的 ID 码，这样程序员可以读出 S：7 中的信息以确定哪一个 SUS 指令导致了 PLC 的工作终止。S：8 在 SUS 指令执行时会包含正在执行的程序文件号。

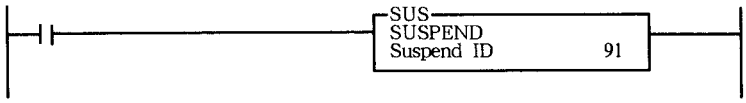


图 15-2 SLC 500 在梯形图梯级中的 SUS 指令

3. SLC 500 的用户出错程序

编写 SLC 500 的程序，使其在特定的环境下出错，将主要出错位 S：1/13 置位。

4. SLC500 编程器在故障检修中的特性

在 PLC 运行的过程中，通过编程器中的数据监视屏幕可以监控输入到控制元素地址中的指令。控制元素包括状态位，这些状态位由每次程序中执行的指令写入。定时器和计数器元素也包含有状态位。PID 指令控制元素也会包括一个字节的出错代码。本书覆盖了每一个指令的状态位和出错代码。

监控程序时，信息指令和一些 ASCII 指令在指令图中显示错误代码。其他方式不能监测到这些错误代码。

SLC 500 程序可以在单步测试模式中每次测试一条梯级。在编程器中选择测试模式而不选择运行或者编程模式时，选择“SINGLE STEP”（单步），接着选择“EXECUTE STEP”（执行步）一次，让 PLC 执行一个完整的扫描循环，再多次选择执行步，每次从梯形图的第一条梯级开始执行程序的一步。也可以使用程序监控及数据监控特性，包括在单步中数据值的修改以及强制 I/O。

也可以在单步测试模式中令 SLC 500 程序运行至断点处。先选择测试模式及单步，选择“SET END RUNG”（设置梯级结束），然后输入需要测试的文件的文件号及梯级号。按下开始键后，PLC 会执行一个完整的扫描循环，接着第二次循环执行到断点处为止。每次按下开始键，PLC 便会完成未完成的扫描循环然后执行下一次扫描循环到断点处为止（也可以在按开始键之前改变断点）。

S5

15.7 SIEMENS S5 的故障检修

用户不能确定 S5 的出错响应（除非写入错误处理 OB）。当出错的原因被排除后，只需要简单地按开关停止 S5 PLC，然后再运行即可。不需要清除任何错误位或者代码。

15.7.1 S5 致命及非致命错误

1. ISTACK

S5 的 CPU 模块在系统数据存储空间的两个区域都保存了 32 字节的状态及出错信息。Siemens将此 32 字节称作 ISTACK，或者中断堆栈存储区。表 15-1 显示了这些 32 字节的内

容⊖。表 15-2 及表 15-3 描述了每一个出错位的作用（Siemens 称为分析位）以及其他状态信息的作用。

ISTACK 的信息可以由编程器在 PLC 处于停止工作模式时显示出来。如果 PLC 遇到一个不能由用户编写的错误处理组织块程序纠正的致命错误时，PLC 会进入停止模式。当致命错误被指出时，这些位置位。可以将光标指向每个位以了解编程器显示出的位的意义的详细解释。当 PLC 进入运行模式时，用户可以监控一些 ISTACK 中的数据（前 6 个字节）。

表 15-1 Siemens S5 ISTACK 存储器内容

2 \ 1	7	6	5	4	3	2	1	0
1			BST SCH	SCH TAE	ADR BAU			
2	CADA	CEDA		REM AN				
3	STO ZUS	STO ANZ	NEU STA		BAT PUF		BARB	BARB END
4						AF		
5	ASP NEP	ASP NRA	KOPF NI		ASP NEEP			
6	KEIN AS	SYN FEH	NINEU					UR LAD
7	不相关							
8	不相关							
9	STOPS		SUF	TRAF	NNN	STS	STUEB	FEST
10	NAU	QVZ	KOLIF	ZYK	SYSFE	PEU	BAU	ASPFA
11								
12	ANZ1	ANZ0	OVFL		OR	STA TUS	VKE	ERAB
13	第六层嵌套					OR	VKE	FKT
14	不相关							
15	第四层嵌套					OR	VKE	FKT
16	第五层嵌套					OR	VKE	FKT
17	第二层嵌套					OR	VKE	FKT
18	第三层嵌套					OR	VKE	FKT
19	嵌套深度（0…6）							
20	第一层嵌套					OR	VKE	FKT
21	数据块的开始地址（高）							
22	数据块的开始地址（低）							
23	块堆栈指针（高）							
24	块堆栈指针（低）							
25	步地址计数器（高）							
26	步地址计数器（低）							
27	状态寄存器（高）							
28	状态寄存器（低）							

⊖ 这里没有给出字地址。列表对于系统数据使用了 SD 前缀，而不是更常用的 RS 前缀。一些 S5 CPU 不能维持表里所示的所有位，而一些更新的 S5 CPU 可能有一些表里没有列出的位。

（续）

2 \ 1	7	6	5	4	3	2	1	0
29	累加器 2（高）							
30	累加器 2（低）							
31	累加器 1（高）							
32	累加器 1（低）							

表 15-2 Siemens S5 ISTACK“分析”（出错）位的含义

出错/错误	出错/错误码	导致 CPU 内存中有缺陷的程序的原因	补 救
不能冷重启	NINEU SYNFEH/KOPFNI	原因： -系统启动出错 -电源错误导致压缩被中断 -电源错误导致程序员与 PC 之间的块传输中断 -程序错误（TIR/TNB/DO FW）	执行一次彻底的复位。重新下载程序
	KOLIF	DB1 编程不正确	重新命名 DB1
	FEST	CPU 自检时发现错误	更换 CPU
出错的子模块	ASPFA	子模块 ID 非法	插入正确的子模块
电池失效	BAU	没电池或电池电量低，要求有保持性特性	更换电源 执行一次彻底的复位 重新下载程序
I/O 未准备好	PEU	I/O 未准备好： -扩展单元有电源错误 -到扩展单元的连接被中断 -中央控制器没有终止符	-检查扩展单元中的电源供应 -检查连接 -在中央控制器中安装终端器
程序扫描中断	STOPS	模式选择在停止模式	将模式选择打到运行模式
	SUF	置换错误： 带有不正确实参的功能块调用	更正功能块的调用
	TRAF	传输错误： -数据字数量大于数据块长度的数据块的语句 -没有首先打开一个 DB 的数据块语句	更正程序错误
	STS	-由语句决定的软件停止（STP） -程序员要求停止 -主 L1 网络要求停止	
	NNN	-一个语句不能解码 -超出最大嵌套深度 -参数溢出	更正程序错误

（续）

出错/错误	出错/错误码	导致 CPU 内存中有缺陷的程序的原因	补　救
	STUEB	块堆栈溢出： -超过块的最大嵌套深度 16	更正程序错误
	NAU	电源错误	
	QVZ	I/O 超时： -程序中一个外围的字节不能寻址 -I/O 模块不认识	更正程序错误或更换 I/O 模块
	ZYK	扫描时间溢出： 程序扫描时间大于设定监视时间	检查程序的连续环路。如果有必要，使用 OB 31 再次触发扫描时间，或者改变监视时间

表 15-3　Siemens S5 ISTACK 分析位和字

控制位寄存器		中断显示寄存器	
ADRBAU	地址列表的构造	ANZ1/ANZ0	00：ACCUM 1=0 或 0 被移位
AF	中断使能		01：ACCUM 1>0 或 1 被移位
ASPNEEP	存储器子模块是 EEPROM		10：ACCUM 1<0
ASPNEP	存储器子模块是 EPROM	ASPFA	非法存储器子模块
ASPNRA	存储器子模块是 RAM	BAU	电池错误
BARB	程序检查	BEF-REG	状态寄存器
BARBEND	请求结束程序检查	BST-STP	块堆栈指针
BATPUF	电源备份 OK	DB-ADR	数据块地址
BSTSCH	请求块移位	DB-NR	数据块编号
CA-DA	处理器之间的通信标志	ERAB	第一次扫描
	输出地址列表可获得	FEST	CPU 自检程序错误
CE-DE	处理器之间的通信标志	FKT	0：O（
	输入地址列表可获得	KE1...KE6	嵌套堆栈入口 1 到 6 输入 A（和 O（
KEINAS	没有存储器子模块		1：A（
KOPFNI	块头不能被中断	KOLIF	处理器之间的通信标志传输列表不正确
NEUSTA	PC 冷重启	NAU	电源错误
NINEU	不能冷重启	NNN	在 PC-2000 中条件语句不能被解释
REMAN	0：冷启动时所有的定时器、计数器和标志复位	OB-NR	组织块编号
		ODER	OR 存储器
	1：冷启动时定时器、计数器和标志的下半部分复位	OVF	算术溢出（+或-）
		PEU	I/O 未准备好：I/O 扩展单元的电源错误；

（续）

控制位寄存器		中断显示寄存器	
SCHTAE	块移位激活（函数：COMP：PC）		到 I/O 扩展单元的连接中断
SD	系统数据（从地址 $EA00_H$）		中央控制器没有终止符
STOANZ	“停止”显示（内部请求）	QVZ	I/O 超时：指引到不存在的模块
STOZUS	“停止”状态（外部请求）	REL-SAZ	相关步地址计数器
SYNFEH	同步错误（块不正确）	SAZ	步地址计数器
URLAD	请求 bootstrapping	STATUS	最后一个条件执行时操作数的状态
		STOPS	模式选择在停止模式
		STS	程序员的停止请求或停止条件导致的操作中断
		STUEB	块堆栈溢出：超过块的最大嵌套深度 16
		SUF	置换错误
		SYSFE	SYSID 块的错误
		TRAF	数据块传送错误：数据字数量＞数据块长度
		UAW	中断显示字
		VKE	逻辑操作结果
		ZYK	扫描超时：超出允许设置的最大程序扫描时间：0.5s

2. BSTACK

除 ISTACK 之外，S5 PLC 同时也在一个 BSTACK（块堆栈）区域中保存了状态信息。编程器可以用来在出现致命错误后显示 BSTACK 的内容。BSTACK 包含了等待继续执行的逻辑块列表。列表中包含了那些包括跳转指令的逻辑块，这些跳转指令调用这些在错误发生时正在执行的逻辑块。列表中也会详细说明任何被中断并等待继续执行的逻辑块的信息。BSTACK 也包含了那些当逻辑块延缓执行时在任何被延缓的逻辑块中激活的数据块的地址。

3. 错误处理组织块

取决于 S5 CPU 的类型，PLC 的操作系统会在接下来的错误处理组织块中寻找一个并执行，而不是当其他某些类型的致命错误发生时就进入错误模式。在执行完一个错误处理 OB 后，S5 CPU 会继续执行致命错误产生时正在执行的程序。错误处理 OB 包括：

- OB 23　响应直接寻址指令时，读或写 I/O 模块的时间超过 160 微秒，执行该命令。CPU 在调用 OB 23 之前将 I/O 模块地址写入 RS 103。
- OB 24　在 I/O 扫描或者更新过程中的通信标志存储器时，读或写 I/O 模块的时间超过 160 微秒，执行该命令。CPU 会在调用 OB 24 之前将 I/O 模块地址写入 RS 103。
- OB 27　程序调用功能块，功能块的变量声明在功能块调用之后发生改变，则执行该命令。

■ OB 32　如果程序试图使用未被调用的数据块，或者并不存在的数据字，或者程序试图产生容量过大的数据块时，执行该命令。

组织块会被调用来响应非致命错误。当然如果错误未被找到，PLC 不会出错：

■ OB 34　当检测到电池容量低时执行。

4. 看门狗时间错误

一种最终的错误处理 OB，用以避免致命错误，而不是从致命错误中恢复：

■ OB 31　可以被调用以重启看门狗定时器，用以避免过长的扫描时间导致的致命错误。

由于超出看门狗时间设置的致命错误，也可以通过延长看门狗定时器的默认 100ms 计时时间来避免。通过将 1 至 255 中的一个值写入 RS 096 中，可以改变看门狗定时器的时间设定。这些数值对应于按 10ms 增量的设定值，因此扫描时间可以被设定在 10ms（对于大多数程序来说太短）与 2.55s 之间。下面的 STL 程序段显示了一个功能块如何配置 PLC 从而使看门狗定时器避免产生错误（除非程序扫描时间超过 300ms）。系统数据字，加上前缀字母 RS，可以仅被功能块程序写入。

```
L KF30     ; 装载十进制值 30
T RS097    ; 传送到 RS097
BE         ; 结束
```

5. 数学错误

与大部分其他 PLC 一样，执行数学运算时，如果产生了过大的正数或者负数而不能存储在存储器中时，S5 可以检测到错误。PLC 将溢出位（OV）置位，然后当作没有出错的情况继续执行用户程序（显然，这属于非致命错误）。在出现这类数学错误的时候程序员可以使用（仅 STL 语言）指令溢出跳转（JO）来检测。跳转目标应该包括响应该错误的程序。其他情况的代码位可以用作检测究竟结果是过大还是过小。在该操作数是一个过大的正数时 CC1 置位，当它置位时正值跳转命令会导致一个至 CC1 的跳转。CC0 在该操作数是一个过大的负数时置位，当它置位时负值跳转命令会导致一个至 CC0 的跳转。如果 CC1 与 CC0 在 OV 置位时同时被清零，则说明数学操作的结果溢出太多以至于 16 位结果变为零。

15.7.2　STEP 5 逻辑错误调试工具

Siemens 的理念是使得每一个可以使用 PLC 最强大功能特性的程序员也可以使用简单的工具编写复杂的功能。STEP 5 保持了这样的风格，提供简单的 STL 语言指令来调试程序：

■ 块结束（BE）指令置于逻辑块列表的最后，所以对于调试程序用处不大。

■ 无条件块结束（BEU）指令可以放在逻辑块中的任何地方。如果 PLC 执行 BEC 指令，当前逻辑块便会终止，同时等待继续执行的逻辑块将继续执行。如果 BEU 遇到 OB 001 指令，I/O 扫描步骤会立即执行，接着 OB 001 重新运行。也可以编写指令跳过 BEU 指令。

■ 有条件块结束（BEC）也可以放在逻辑块中的任何地方。它与 BEU 指令工作方式类似，除了可以被一个布尔逻辑语句控制之外。

■ STS 与 STP 指令可以用于强制使 PLC 终止扫描循环，同时程序员也可以检查 PLC

存储器、CPU 寄存器、ISTACK 以及 BSTACK 在程序终止时的状态。两个指令的执行都是无条件的，尽管程序可以有条件地跳过它们。立即停止（STS）指令可以有效地产生一个致命错误，使 PLC 立即跳出运行模式。程序扫描末端终止（STP）指令允许执行剩余的扫描循环到结束，包括在它使 CPU 跳出运行模式之前，把输出映像写到输出模块。

15.7.3 普通编程错误

在 STEP 5 中有许多非常普通的编程错误，所以也需要在此提一下，尽管它们也会在其他地方提到。

1）不正确的字节与字的混用：当从存储器地址中加载一个字节时，从该地址得到的 8 位数值被放入累加器的低位。当从存储器地址加载一个字时，从该地址得到的 8 位数值被放入累加器的高位，而下一个存储器地址中的 8 位数值被放入累加器的低位。在从累加器到存储器之间的数据传送过程也是如此（见第 5 章）。

2）结构化程序中定时器和/或计数器的放置不正确：绝大部分的定时器在执行过程中遇到控制逻辑错误时都会复位。如果定时器在仅当控制逻辑为真时才调用的逻辑块中，它就不能自己复位，而只能运行一次（带有定时器的逻辑块的第一次执行）。计数器同样也需要由一个改变的逻辑来控制（见第 4 章）。

3）模拟量的不正确使用：在模拟输出模块中的数模转换器（DAC）忽略了 16 位数据字的低四位并且将高 12 位数值转换成模拟信号。某些模块假定二进制数值为带符号的二进制格式，因此如果最高位置位后，模拟输出就会出现负值。而在模拟输出模块中的模数转换器（ADC）产生一个 13 位的数（通常以带符号的二进制形式），存入 16 位数的高 13 位中。此时低三位非常重要！如果模拟输入值超出可以转换为二进制数值的范围，最低位置位；如果模拟输入值超过输入模块名义上的范围但是还足以产生一个正确的数字数值，倒数第二个低位置位；如果输入模块仍处于转换模拟数值为二进制的过程中，倒数第三个低位置位，因此这些数字值还不是最终的正确值（见第 2 章）。

15.7.4 STEP 5 编程器的调试特性

STEP 5 编程器可以用于创建交叉参考列表，该列表用来显示在哪里使用了特别的位、字或者结构体；同时也可以用于观察当程序执行时（同时也允许某些值被改变）PLC 存储器中状态变量屏幕的显示值；也可以在程序执行的过程中强制加载特定内容进入存储器（包括 I/O 映像）。

STEP 5 同样也允许用户强制输出模块触点在 PLC 处于停止模式时关或者开。STEP 5 目前没有包含直方图功能。（据说 Siemens 正在研发适用于 STEP 7 的直方图）

S7 15.8 SIEMENS S7 的故障检修

S7 PLC 有非常复杂的故障检修特性，如果全面介绍，至少得用上整整一本书。本节的内容足以使用户能够使用大部分的特性，并了解 STEP 7 手册描述了哪些其他特性。

当通过编程器连接 PLC 后，工作站硬件窗口的图表显示出每个模块的工作状态。

如果由于配置问题导致某模块不能工作，那么图表中对应该模块的部分会闪烁。X 符号表示该模块出错，而带有强制位的模块也会被它们顶部显示的红线指示出来。在工作站硬件窗口中点击相关模块的图表，出现的对话框会让你了解到模块存储器中存储的诊断信息。

CPU 也在系统状态数据中保存了诊断事件表（有时称为错误事件）。诊断事件会在存储器的诊断缓存中以诊断信息的形式记录下来。用户不能擦掉诊断缓存的入口，即使完全重启存储器也不能。诊断缓存与相关存储器区域用来存储本章中所描述的系统状态表（SZL）。

15.8.1 STEP 7 的编程器问题

如果不能启动 STEP 7，可能是由于 Siemens 为了保护 STEP 7 不受非法拷贝而采用的方法造成的麻烦。装有 STEP 7 的 PC 机必须取得合法的授权，否则 PC 机可能会被类似于 SCANDISK 这样的修复文件命令操作所破坏，或者当硬盘被压缩时，或者当新的操作系统加载时。当你读到这里时，Siemens 可能已经意识到微软的市场目标是为了让每个用户每三天就买一个新的操作系统，那么它也会出台更好的保护方案了。

15.8.2 S7 硬件故障检修

每个 S7 模块都有一个系统出错 LED 指示灯（SF，当模块出错时亮红灯）和多个状态 LED 灯（如果没有检测到电路问题亮绿灯）。CPU 模块有额外的 LED 灯，每一个都有特定的用处（比如指示电源电量低，强制点亮等等）。S7 允许模块和网络连接部分的热插拔（因此可以在 PLC 运行时更换模块），但要注意：

1）当 PLC 运行时，不允许移除或者插入电源、CPU 或者接口模块（IM）；

2）在 CPU 中更换存储卡会导致 CPU 的指示停止的 LED 闪烁，指示在新的存储卡中的内容载入 PLC 存储器之前存储器需要重启；

3）CPU 识别出新加载的信号模块需要的配置数据，以及将配置数据写入模块的过程，可能需要两秒的时间；

4）不允许将模块插入其他型号模块的插槽；

5）在网络段两端的模块都必须加电（并且所有网络终端电阻器都是打开的），否则通信数据会丢失。

15.8.3 S7 的故障检修配置

仅有几个配置的改变可以人为设置，用来影响 S7 PLC 如何响应错误或者如何收集状态信息。一些故障检修的配置选项在第 10 章中已涉及，包括导致 PLC 做以下动作：

1）检查 RAM 存储器，检查硬件配置以确定硬件的连接是否正确，有错误发生时出错；

2）限制扫描时间，如果扫描循环过长则出错；

3）每次当 CPU 进入终止模式时都通过 MPI 网络发一条消息（最近的诊断缓存入口），用以描述为什么 PLC 停止工作；

4）在标准原因之外再附上扩展的原因列表，将诊断数据入口写入诊断缓存中；

5）限制将配置参数写入模块或者等待响应的时间，如果超时则出错；

6）使能带有诊断能力的信号模块（SM），用以产生诊断中断至 CPU，同时当检测到错误时将扩展的诊断信息发至 CPU；

7）告知 PLC 在主和从 Profibus-DP 中何处存储 DP 诊断信息；

S7 CPU 可以配置来执行第 10 章中未提及的故障检修功能：

1）监视选出的布尔元素的状态，当状态改变时发送信息。Siemens 将此功能称为符号相关消息（SCAN）。从监视符号表格列出的位中选出的一位，接着选择 STEP 7 中的“EDIT-SPECIAL OBJECT PROPERTIES-MESSAGES”（编辑-特殊对象属性-消息），这样就可以输入消息文本、监视位的时间框图以及其他属性。该消息随后会被编译并且下载到 CPU 中。在工作站已经被“INSERT-WICCC OBJECT-OPERATOR STATIONS”（插入-WINCC 对象-工作站）识别出后，使用“OPTIONS-PLC-OS-CONNECTION DATA-TRANSFER”（选项-PLC 与操作系统互联数据-传送）选择一个工作站来接收消息。

2）作为报警消息来传送的消息。与块相关的消息文本与消息目的地也会通过编辑-特殊对象属性-信息写入，但同时用户程序必须执行一个消息传送系统函数（SFC 17 或者 SFC 18）或者系统功能块（SFB 33 或者 SFB 37）来初始化消息的传送。

3）关于用户定义的诊断消息。用户定义的诊断消息文本如上所述被输入，但同时程序必须执行 SFC 52，WR _ USMSG，来初始化消息的传送，而且 WR _ USMSG 的输入参数必须指示出消息传送的地点。这种类型的消息仅能传送到诊断缓存和编程器，不能传送到工作站。

15.8.4 S7 状态信息

S7 CPU 模块和其他带有诊断能力的模块保存了可以用编程器读出的状态信息。绝大部分相同的信息也可以通过用户程序中的系统函数读出。

1. CPU 模块状态

在 CPU 模块中，状态信息保存在两个地方：系统状态列表（SZL 表）和状态字中。

2. CPU 状态字

保存在 S7 CPU 模块中的状态字包括反映 CPU 在执行用户程序时其状态的状态字。与其他 PLC 类似，S7 不会仅因为用户程序要求 PLC 进行一个产生了超过 PLC 处理能力的数的数学运算而出错。与其他 PLC 类似，这个非致命错误可以被检测到，程序也可以通过检测受到影响的状态位来检测该错误。状态字包括了与此功能相关的 4 个位，其中两个位可以由布尔逻辑指令来寻址：

OV（溢出） 每当产生一个超出当前使用的计数系统范围的结果时，置位。如果该位置位，则表示结果出错。而下一次数学运算则会将此位置位或者复位。

OS（溢出置位） 与 OV 同样置位，但是会保持置位直到有特定的复位，或者特定的指令（JOS、CALL 或 BE）执行。因于 OS 可以自保持，所以不需要每次数学运算后都去检测。

还有另外两种状态字位，CC1 与 CC0，表示操作结果（或者比较结果）的长度。尽管 CC1 与 CC0 不能直接寻址，但可以通过布尔逻辑指令检测，如果下列代码中的一个在布尔指令中代替一个位地址：

==0 如果结果为零（CC0=CC1=0），置位；

>0　如果结果为正（CC1＝1，CC0＝0），置位；

<0　如果结果为负（CC1＝0，CC0＝1），置位；

<>0　如果结果非零（CC1＝1＋CC0＝0 或者 CC1＝0＋CC0＝1），置位；

<＝0　如果结果非负（CC1＝0＋CC0＝0 或者 CC1＝0＋CC0＝1），置位；

>＝0　如果结果非正（CC1＝1＋CC0＝0 或者 CC1＝0＋CC0＝0），置位；

举例说明，如下 STL 程序段加上两个数字，然后检查结果。如果结果为负或者由于溢出导致其非法，程序会跳过存储结果的指令。

```
      L  MW 10
      L  MW 14
      + I
      A< 0
      o 0V
      JC  Past
      T   MW 18
Past: BE
```

3. CPU 系统状态列表（SZL）

SZL 区域被分为若干信息子列表。每个子列表有一个十六进制的 SZL-ID 码，同时一些子列表还会被进一步划分为带有索引号的段。当使用编程器阅读 SZL 列表信息时，不需要知道 SZL-ID 码或者索引号。但是当在程序中读取 SZL 子列表时，使用 SFC 51 及“RDSYSST”指令时，需要将这些号码用作输入参数。表 15-4 显示出 S7-400 CPU 中可以获取的 SZL 列表信息及每种子列表的十六进制（W＃16＃）SZL-ID 码。可以参考使用手册的设置（或者用 RDSYSST 输出 SZL-ID 0000）了解到在 CPU 中哪些系统状态记录可以获取。（完整的叙述列表有 60 多页!）如图 15-6 所示，SZL 数据可以分为如下几类：

1）系统数据，包括了对如下几类的描写：

- CPU 系统状态列表中可以获取的 SZL-ID 子列表；
- CPU 模式与特点，包括它拥有多少用户和系统存储器；
- 用于描述被编程的块的数据，它们的优先级（在某些 S7 模式中可配置）以及系统数据块列表（SDB2）。（系统数据块包括从 CPU 中下载的已编译的配置数据）；
- 许可的 I/O 框架系统大小（仅对于 S7-300）；
- CPU 及其他模块中当前的 LED 灯状态；
- 启动事件（也叫做错误事件）编号，该编号被标记用以使组织块中断低优先级的程序，同时还有一张优先级列表，这些优先级由于用户程序执行 SFC 而被禁用或者屏蔽；
- 中断及优先级状态，包括了 20 字节被传送到每一个正在运行或者被中断的组织块中的参数数据、每一个 OB 操作状态、可用的中断嵌套级、启动信息的限制以及可以由 OB 使用的本地堆栈数据；
- 当前操作模式、最近模式的改变，以及显示 PLC 如何启动或者为何停止工作的状态。

2）诊断状态数据，包括：

- 通信配置参数与状态的扩展列表，包括波特率设定、与通信伙伴的连接、通信工作活动、全局数据交换、标记符相关及块相关的信息配置和状态（如本章前面所述）；

- 初始化组织块的启动事件列表，每一个组织块的启动信息（这些信息也存储在诊断缓存中）；
- 带有在 PLC 中活动的诊断能力的模块的状态列表，包括框架以及连接到该 PLC 的 DP 网络的工作站。

3）诊断缓存，S7 CPU 在每次诊断事件发生时写入诊断缓存一个入口信息。诊断事件由已经打开了诊断中断功能的 CPU 或者模块进行错误检测。诊断事件可能是简单的模式转换。这些诊断事件被视作启动事件，导致组织块执行或者导致 PLC 停止工作。用户定义的诊断事件信息也可以通过 SFC 53（WR _ USMSG）指令写入诊断缓存。每种诊断事件记录包括：

- 发生的时间及日期；
- 启动事件的 ID 码及启动信息，当错误初始化时被传送到组织块。（启动事件码与文本诊断信息及帮助信息相关，这几类信息都会在编程器中显示）

4）来自于模块或者 DP 从站的诊断数据，通过对模块和 DP 从站诊断地址的定位，该类数据会在系统状态列表中出现。SZL 中的诊断数据包括：

- 从带有诊断能力的模块中记录 0 类诊断。记录 0 是 4 字节的错误位，对任何模块它的格式是标准化的。（参见操作手册设置中的 SZL-ID 00B1）
- 从带有诊断能力的模块中记录 1 类诊断。记录 1 包括该模块类型的特殊数据结构体。（参见操作手册）
- 从 Profibus-DP 从站中得到结构化的标准诊断信息，如在 EN 50 170 中所指定的那样，包括了六种标准格式字节及取决于 DP 从站类型的附加数据。

表 15-4 S7 系统状态列表一览（SZL）

	部分列表	SZL-ID
系统数据	一个模块的所有 SZL-ID 列表	W＃16＃xy00
	模块标识	W＃16＃xy11
	CPU 特性	W＃16＃xy12
	用户存储区	W＃16＃xy13
	系统区	W＃16＃xy14
	块类型	W＃16＃xy15
	优先级	W＃16＃xy16
	编号小于 1000 的允许的 SDB 列表	W＃16＃xy17
	最大 S7-300 I/O 配置	W＃16＃xy18
	模块 LED 灯的状态	W＃16＃xy19
	中断/错误指派	W＃16＃xy21
	中断状态	W＃16＃xy22
	优先级	W＃16＃xy23
	模式	W＃16＃xy24

（续）

	部分列表	SZL-ID
诊断状态数据	通信能力参数	W#16#xy31
	通信状态数据	W#16#xy32
	诊断：设备登录列表	W#16#xy33
	开始信息列表	W#16#xy81
	开始事件列表	W#16#xy82
	模块状态信息	W#16#xy91
	框架/站 状态信息	W#16#xy92
诊断缓存	CPU 的诊断缓存	W#16#xyA0
模块和 DP 从站的诊断数据	模块诊断信息（数据记录 0）	W#16#00B1
	模块诊断信息（数据记录 1），地理地址	W#16#00B2
	模块诊断信息（数据记录 1），逻辑地址	W#16#00B3
	DP 从站的诊断数据	W#16#00B4

4. 模块状态

信号模块（SM）、通信处理器模块（CP）、接口模块（IM）以及带有诊断能力的 DP 从站，都保持了它们自身的诊断数据信息。这些信息可以通过编程器检查，也可以通过在用户程序中调用 SFC 59（“RD_REC”）指令读出（RD_REC 在第 7 章中有叙述）。模块的诊断数据与它保存在 CPU 系统状态列表中时的格式一样（记录 0 与记录 1，或者标准化的 DP 从站诊断结构体）。模块存储器中的诊断数据比 CPU 的 SZL 列表保存了更多的信息，或者说更多的近期信息，尤其是当该模块并未配置用来发送所有的诊断数据至 CPU，或者说近来模块没有传送数据时。

5. 全局数据状态双字

配置 PLC 时，可以建立一个全局状态双字来保存描述 PLC 之间全局数据信息包交换的状态位。状态位可以通过编程器，使用“VIEW-GD STATUS”（察看-GD 状态）来监测，或者可以通过用户程序中的指令来读出。置位的全局状态位会保持置位状态，除非遇到特定的复位情况。当数据丢失或者被破坏时，一个或者多个低 8 位会置位。如果全局数据的传送器重启，倒数第二字节的位 3 置位。当新数据被接收时，最高位会被置位。（其他位均无此功能）

6. SFC RET_VAL 指示的出错

所有 SFC 都会用 RET_VAL 这个符号名返回一个单整数参数。RET_VAL 包括描述 SFC 如何有效工作的状态信息。如果用户程序包括这个输出参数的地址，则编程器可以用来显示 RET_VAL，或者用户程序可以求它的值。如果 RET_VAL 不小于零，则 SFC 工作（某些 SFC 使用 RET_VAL 来返回一个正的结果，用以在 SFC 工作时调用程序）。如果 RET_VAL 最高位是 1，则 RET_VAL 是负值，表示 SFC 现在没有工作。RET_VAL 的低字节包括了指示错误的代码。如果高字节余下所有的位都为零，错误代码特别对应于产生错误的 SFC，否则，代码就是一个可以应用于任何 SFC 的普通错误类型。参见参考手册中

RET _ VAL 的错误代码部分。

15.8.5 出错响应组织块

对于某些类型的致命错误，S7 PLC 会在检测到错误时寻找出错响应组织块。如果出错响应 OB 未找到，则 PLC 会在将描述致命错误的诊断信息写入诊断缓存之后停止工作。

Siemens S7 提供了几种组织块，用来作为出错响应组织块。包括发生异步错误时调用 OB 80 到 OB 87；响应同步错误时调用 OB 121 和 OB 122。同步错误一般由程序中的错误引起（因此错误与程序同步发生），而异步错误是由于类似硬件错误或者模块的诊断中断信号引起的。在第 11 章中的出错程序部分描述了出错响应 OB；同时也描述了如何使用 SFC 36 至 SFC 42 指令来停用、延迟或者屏蔽选出的错误类型，这样它们就不会引起程序的中断，直到被重新启动或者取消屏蔽。

如果一种导致 S7 寻找错误响应 OB 的致命错误发生，那么 OB 会被找到（因为程序员为 OB 编写了程序），S7 会立即中断当前程序的执行，接着启动出错响应 OB。当出错响应 OB 结束时，被中断的程序会继续执行。致命错误因此可以转变为非致命错误。诊断信息仍然会输入到诊断缓存，即使 PLC 不进入停止模式。SFC 46，“STP”，也可以被称为是出错响应 OB，用以阻止被中断的程序继续执行。

其实在程序员创建出错响应 OB 之前，每个 OB 都已经留有 20 字节的变量声明表格，填充着缺省的符号名。S7 操作系统在调用出错响应 OB 时，总会从诊断缓存中传送数据至变量。程序员可以加入更多（临时）的变量声明，同时也可以写一段程序来检查错误记录的参数，从而纠正错误。如果 OB 需要更多的状态数据，则 OB 可以调用 SFC 51，“RDSYSST”，去读 CPU 中系统状态列表（SZL）的任何一个子列表，或者调用 SFC 6，“RD _ SINFO”，去读描述 PLC 上次是如何启动的状态信息。SFC 59，“RD _ REC” 也可以被调用，用来从 I/O 模块中读取数据。SFC 13，“DPNRM _ DG”，也可以用来从 Profibus-DP 读取诊断数据。RSDYSST 在第 11 章中已提及，RE _ REC 在第 7 章中已提及；而 RD _ SINFO 与 DPNRM _ DG 并未在本书中涉及。

允许被中断的程序继续执行的出错响应 OB 通常都需要将参数传回被中断的程序，以表示它们已经进行了一些动作，但是组织块却只能拥有临时变量。有几种不同的方法可以将错误响应 OB 写入输出数据，这些输出数据可以在 OB 结束后被其他程序读出。OB 可以写入数据块一个数值或者一个在标记符表格中定义过的全局变量，或者使用 SFC 58，“WR _ REC”，将数据（非配置数据）写到模块中。同步出错响应 OB 与它们中断的程序共享 CPU 的寄存器，所以当出错响应 OB 结束时，它们对寄存器做的所有改动都会保存下来。

15.8.6 使用编程器观察的 S7 诊断信息

在 PLC 出错而停止工作后，程序员可以使用编程器来在线连接出错的 PLC，查看诊断数据（包括诊断缓存的内容）。这些数据描述了当致命错误发生时 PLC 所处的状态。从工作站硬件窗口中选择错误模块（在诊断标记符中显示），同时会出现一个对话框，提供了相应模块的诊断数据。如果错误模块是 CPU，则诊断数据会包含系统状态列表的数据。

关于未出错模块的诊断数据同样也可以用这种方法显示出来。这种特性并不是为了让使

用者观察实时的事件，因为屏幕不会在 PLC 运行的时候自动更新数据信息。每次从 CPU 中获取一组新的数据来显示，都必须按更新按钮。

诊断数据不会以上述的 SZL 列表及模块诊断数据的格式出现。它们会被组合成下列对话页面，这取决于在任务窗口中选择的模块类型。

1）诊断缓存页面，包括了时间、日期以及每次进入诊断缓存入口的文本信息。指向诊断事件，然后（可选）选择“HELP ON EVENT”（事件帮助）查看更详细的信息。选择“OPEN BLOCK”（打开块）立即跳转到显示当诊断事件发生时正在执行的逻辑块和指令的屏幕。

2）诊断中断，显示模块的诊断数据，分两页显示，对应于记录 0 与记录 1。

3）堆栈页，仅当 PLC 在停止工作模式时显示，有助于确定为何 PLC 突然进入停止工作模式。屏幕会显示如下内容：

- BSTACK，包含了由于被中断（包括因为停止工作而中断的程序）或者由于调用其他程序被暂停的程序的列表。BSTACK 同时也显示了每一个即将执行的暂停的程序中，从哪一部分继续执行。
- ISTACK，包含了 PLC 保存下来便于继续执行的程序信息（这些程序在 BSTACK 中记录为被中断）。这些信息包括了中断时 CPU 寄存器的内容，以及被中断程序的优先级。可以选择打开块命令来跳转到显示每一个被中断的程序信息的屏幕上，包括那些由于转到停止工作模式而被中断的程序。
- LSTACK，显示了每一个在 BSTACK 列表中的程序的本地堆栈（临时指定）存储器的内容。将光标指向 BSTACK 入口并选择 LSTACK。

4）DP 从站诊断，显示了 DP 从站的诊断数据，在子页面上对应为标准数据及模块专用数据在 EN 50 170 中的规格。DP 从站（如果有）的子模块诊断数据也是可以获取的。

5）扫描循环次数，显示最短、最长、最近的扫描循环次数以及设定作为最大扫描时间的时间值。

6）通信页面，显示波特率、CPU 允许的通信连接数目以及正在使用的连接数目。

7）常规信息，描述模块类型、特性、位置以及操作状态。

8）用户存储器使用，显示有多少 CPU 模块的存储器被使用。（也可以选择压缩命令来尽可能得到更多的自由空间）

9）时间信息，显示当前的时间、日期（在配置过程中设置了时钟）以及 CPU 工作时间。

10）性能数据，显示在 CPU 存储器中每一部分可以使用的存储器地址范围。（在下载可能占用相当多空间的程序时应先检查该地址范围）选择 BLOCKS 来察看 CPU 正在使用的组织块列表、对于用户程序块允许的数目及大小以及可用的 SFC 和 SFB 列表。

15.8.7 STEP 7 逻辑错误调试工具

1. 调试指令及系统功能

不同于块结束指令的指令：BEC 与 BEU，Siemens 并不提供任何特别的用于调试的指令。（BEC 与 BEU 在描述调试程序的 STEP 5 指令的小节中有叙述）

某些系统功能（SFC）执行可以用作调试程序的操作。在本章或者其他章节中已经有讨论或提及：

SFC 6，“RD _ SINFO”　　读 OB 开始信息

SFC13，“DPNRM _ DG” 读 DP 从站的诊断数据
SFC17，“ALARM _ SQ” 产生可辨认的块相关消息
SFC18，“ALARM _ S” 产生永久的可辨认的块相关消息
SFC19，“ALARM _ SC” 询问可辨认的块相关消息
SFC31，“QRY _ TINT” 询问 time-of-day 中断
SFC34，“QRY _ DINT” 询问 time-delay 中断
SFC46，“STP” 终止 CPU
SFC51，“RDSYSST” 读系统状态列表或部分列表
SFC52，“WR _ USMSG” 将用户定义的信息写入诊断缓存
SFC59，“RD _ REC” 读数据记录

一些在本章中未讨论到的附加的 SFC，对确认纠错或者避免错误也很有帮助，包括：

SFC38，“READ _ ERR” 读错误寄存器
SFC43，“RE _ TRIGR” 重新触发循环时间监控
SFC44，“REPL _ VAL” 将替换数值送入累加器 1
SFC47，“WAIT” 延时执行用户程序

2. 编程器的调试特性

CPU 参考数据可以由编程器生成，用以描述 S7 CPU 模块中的程序。参考数据包括程序中使用的存储器地址列表，可用存储器列表以及并不存在的标记符名称的地址。同时也可以创建前后参照列表，它用来描述存储器在哪里使用以及对其他程序的调用情况的列表，也可以创建程序结构的描述信息。

在 STEP 7 程序编写过程中，编辑器通常缺省设置为增量方式，因此它会标注出那些可检测到的程序错误（例如使用了未定义的标记符）。

监视程序时，程序员可以建立并保存多种变量表格，然后从中找到一个合适的来监控特定过程。每一个表格都可以有变量的组合，被指定用作绝对地址或者标记符。可以设定变量表格的数据捕捉触发为只发生一次（在实验室测试环境中），或者每次扫描都发生（在过程测试环境中），同时数据获取的触发点可以是扫描循环的起点或终点，也可以是当 PLC 进入停止工作模式时。也可以使用编程器的调试菜单来建立断点。每次运行到断点时，PLC 会进入终止模式，直到选择了继续命令。

在 PLC 处于运行或者停止状态时，可以使用变量表修改变量值。当 PLC 处于停止模式时，变量-启动外围输出（PQ）命令可以用来编辑输出值，而变量-激活外围输出（PQ）命令可以用来写入新的输出值。

CQM1

15.9 OMRON CQM1 的故障检修

当 CQM1 出错时，在 PLC 重新进入运行模式之前必须清除所有的错误代码，即使产生错误的地方已经被修复。

15.9.1 CQM1 故障检修配置

如第 10 章中所述，可以使用 DM 6655 来配置 CQM1。在错误日志存储器区域（DM 6569 至 DM 6599）中存储致命及非致命错误的描述。缺省情况下，CQM1 可以存储最近 10

次的错误描述，也可以仅是最早的 10 次，或者一个也不存储。DM 6655 也可以用做禁止 CQM1 在每次扫描循环时间大于 100ms 时将 SR 25309 置位[一]，或者在检测到电池容量低时将 SR 25308 置位来停止 CQM1。

15.9.2 CQM1 致命和非致命错误

致命错误的原因包括许多方面，比如电源断电，检测到错误存储器，与 I/O 模块通信中发生错误，程序中没有结束指令，或者扫描时间超出了循环监视器的时间设定值。严重错误报警指令的执行：在用户程序中的 FALS（07）会触发一个致命错误。FALS（07）在第 11 章中讲述过。当致命错误发生时，CQM1 首先保存错误信息，关闭所有输出，打开 CPU 模块中的 ERR/ALM 指示灯，然后停止工作。操作者可以检查存储器中的错误信息，解决问题，清除错误代码，再打开 PLC 就可以重新进入运行模式[二]。

非致命错误包括 CPU 与存储器模块之间传送数据时发生的错误，存储器的 DM 区中的 PLC 设置数据出了问题，循环时间超过了 100ms，备用电池损坏，或者错误报警指令 FAL（06）的执行。FAL（06）在第 11 章中讲述过。使用 CPU 端口（RS-232C、外围、脉冲、译码器）的过程中遇到的问题也可以视为非致命错误。当非致命错误发生时，CQM1 保存错误信息，但不会停止工作。CPU 模块上的 ERR/ALM 指示灯开始闪烁，但扫描循环继续运行。

当致命或者非致命错误发生时，CQM1 会：

1）将两位 BCD 码的 FAL 错误代码或者 FALS 错误代码。[二]写入 SR 253 的低字节中（OMRON 称作是 FAL 存储器区域）。CQM1 实际上保存了最近三次的 FAL/FALS 错误代码，但在 SR 253 低字节中仅可获得最近一次的代码。

2）在错误日志中保存错误的描述（受 DM 6655 字的配置影响）。错误日志中的第一部分存储空间（DM 6569）包括了一个指向下一个必须写入日志的入口的指针，接着是三个字的关于每次错误（最多十次）的记录（在 DM 6570 至 DM 6599 中）。每个记录包括了 FAL 或者 FALS 的错误代码以及发生错误的时间和日期。

存储器中 SR 与 AR 区域保存了其他状态位，这些状态位也被置位用来进一步描述非致命错误，同时用户程序也会包含检测与纠正错误的指令。另外，AR 26 保存了自从 PLC 进入运行模式以来最长的扫描循环时间，AR 27 则保存了当前扫描循环执行的时间。

FAL 与 FALS 错误代码在表 15-5，表 15-6 中有具体描述，其他有可能受影响的存储器区域也列了出来。

需要注意的是 CPU 端口的错误并没有 FAL 错误代码，但是当通信错误发生在 RS-232C 或者外围端口时，AR 24 中的位会置位，同时如果脉冲输出端口或者高速输入计数端口存在并且没有工作时（参考使用手册），AR 04 中的位也会被置位。当 CPU 端口问题发生时（总认为是非致命的），端口连接器的指示灯会停止闪烁。

[一] 循环时间溢出标志 SR 25309 与 AR 2405（长循环时间标志）不同。如果一个扫描循环的时间超出在 DM 6619 中的最小扫描循环设置，AR 2405 置位。

[二] 每次执行 FAL 编号 00 的 FAL（06），会删除 PLC 存储器中的三个 FAL 错误码之一，并且第二老的代码会移到 SR 253 的低字节。打开外部设备的 SR 25214，清除整个错误日志，随后 SR 25214 会自动复位。SR 和 AR 存储区域的状态位在每个扫描循环清除他们自己的状态。

[三] FAL 是一个非致命错误，表示失败警告。FALS 是一个致命错误，表示严重失败警告。

表 15-5 CQM1 致命错误代码

消 息	FALS 编号	含义和适当的响应
电源中断 （无消息）	无	电源已经中断最少 10ms。检查电源电压和电源线。试着重新打开电源
MEMORY ERR	F1	AR 1611 开：在 PLC 设置中出现校验和错误（DM 6600 到 DM 6655）。初始化所有 PLC 设置并重新输入
		AR 1612 开：在程序中出现校验和错误，指示一条不正确的指令。检查程序并更正所有检测到的错误
		AR 1613 开：在扩展指令的数据中出现校验和错误。初始化所有扩展指令设置并重新输入
		AR 1614 开：电源开时安装或移去存储器卡。关闭电源，安装存储器卡，再次打开电源
		AR 1615 开：在启动时不能读存储器卡。检查标志位 AR 1412 到 AR 1415 找出错误，改正错误，并重新打开电源
NO END INST	F0	程序中没有 END（01）。在程序结尾地址写 END（01）
I/O BUS ERR	C0	CPU 与 I/O 模块传递数据时发生一个错误。使用标志位 AR 2408 到 AR 2415 确定问题所在，关闭电源，检查 I/O 模块是否有松动，并重新打开电源
I/O UNIT OVER	E1	I/O 模块中的 I/O 字的数量超出上限。关闭电源，重新安排系统以减少 I/O 字的数量，并重新打开电源
SYS FAIL FALS** 看注意	01 到 99	程序中执行了 FALS（07）指令。检查 FALS 号以确定导致指令执行的条件，改正错误，并清除错误
	9F	循环时间超过 FALS 9F 循环时间监控时间（DM 6618）。检查循环时间，如果有必要调整循环时间监控时间

表 15-6 CQM1 非致命错误代码

消 息	FAL 编号	含义和适当的响应
SYS FAIL FAL**	01 到 99	在程序中已经执行 FAL（06）。检查 FAL 编号以决定导致执行的条件，更正这些条件，并且清除错误
	9D	CPU 与存储器卡之间数据传递时发生错误。检查标志位 AR 1412 到 AR 1415 的状态，并直接改正 AR 1412 开：转到编程模式，清除错误，并再次传递 AR 1413 开：传输目的地写保护 如果 PLC 是目的地，关掉 PLC 的电源，确定 CPU 的 DIP 开关的 pin 1 是关，清除错误，并再次传递 如果 EEPROM 存储器卡是目的地，检查电源是否为开，清除错误，并再次传递 如果 EPROM 存储器卡是目的地，改为可写存储器卡 AR 1414 开：目的地的容量不够，检查 AR 15 中源程序大小，考虑使用不同的 CPU 或存储器卡 AR 1415 开：存储器卡中没有程序，或者程序包含错误。检查存储器卡

（续）

消　息	FAL 编号	含义和适当的响应
	9C	脉冲 I/O 功能或绝对型编码器接口功能发生一个错误。检查 AR 0408 和 AR 0415 的内容（两位 BCD），并直接改正。（这里的错误码只用于 CQM1-CPU43-E 和 CQM1-CPU44-E CPU） 01，02：硬件发生错误。关掉电源，并再次打开。如果还有错误，则更换 CPU 03：PLC 设置（DM 6611，DM 6612，DM 6643，DM 6644）不正确，改正配置 04：脉冲输出时 CQM1 操作被中断。检查模块接收脉冲输出是否受到影响
SYS FAIL FAL**	9B	在 PLC 启动时检测到一个错误。检查标志位 AR 2400 到 AR 2402，并直接改正 AR 2400 开：在电源打开时，检测到 PLC 设置的一个不正确的设置（DM 6600 到 DM 6614）。在编程模式下改正设置，并再次打开电源 AR 2401 开：在转换到运行模式时，检测到 PLC 设置的一个不正确的设置（DM 6615 到 DM 6644）。在编程模式下改正设置，并再次打开电源 AR 2402 开：在操作时检测到 PLC 设置的一个不正确的设置（DM 6645 到 DM 6655）。更正设置并清除错误

数据错误可以认为是非致命错误，因为它没有专门的错误代码。错误位受操纵数据值的指令影响。用户程序可以在每次可能导致不正确的结果数值的数据操作指令之后监视这些算术错误位：

SR 25404　如果结果超出了使用中的二进制数字系统允许的正数值的范围，则置位。同时结果会保存在一个比正确答案占用地址小得多的地址中（意即保存的结果不一定是正确答案）。

SR 25405　如果结果低于使用中的二进制数字系统可允许的最低数字范围，则置位。同时结果会保存在一个比正确答案占用地址大的多的地址中（意即保存的结果应该为正确答案）。

SR 25504　如果产生进位时（例如，当使用一个指令操作一个 16 位数值时产生了一个 17 位的结果），则置位。这意味着结果太大，无法保存。

SR 25503　如果检测到其他错误，则置位。通常表示间接寻址出错或者数据值不在数学指令想要操作的二进制表格中。

15.9.3　CQM1 逻辑错误调试工具

有几种指令可以在调试用户程序时使用。

1. END（01）

通过插入 END（01）指令，在正常结束之前终止扫描循环。这样可以允许用户检查程序段位于 END（01）之前的部分是否正确工作。

2. 失败报警，FAL（06）以及严重失败报警，FALS（07）

在某一特定的点上插入严重错误指令 FALS（07），使 PLC 终止扫描循环，进入错误模式。该指令在第 11 章中的出错程序部分叙述过。一个程序可以有一些有条件的 FALS（07）指令[⊖]，每一个都有程序员自己指定的 FALS 错误代码，因此程序员可以清楚究竟是哪一个 FALS 指令导致了程序终止，以及什么时候发生终止。在 FALS 条件判断为真时，当前 CPU 状态，包括它的存储器内容，都会保留下来。

一个或多个错误报警，FAL（06）指令，在每次 FAL 指令执行条件判断为真时，可以用来保存带有用户指定的 FAL 错误代码的错误信息，以及发生错误的时间，并将它们写入错误日志中。FAL（06）不会使 PLC 终止正常的扫描循环。FAL（06）在第 11 章中已叙述。

3. 消息显示，MSG（46）

与 FAL（06）指令的用法类似。当 MSG（46）指令条件判断为真时给编程器发送一个消息。用户可以自行创建最多 16 位扩展 ASCII 码的任何字符消息，同时以 ASCII 码形式存在存储器中（每次输入两位 BCD 数值，存在 8 个连续的存储器地址中）。第一个 ASCII 码进入最低存储器地址的高位。而当 ASCII 码 0D 跟随在字符最后输入时，也可以输入少于 16 个字符的信息（例如 XXXX0D）。MSG（46）指令中包含一个参数：FM（首信息字）参数，代表了 ASCII 代码序列的起始地址。在写操作的同时，编程器会保存三条信息，因此 OMRON 设计出一个消息优先系统：在 LM 存储器中的信息优先级最高，可以取代编程器存储器中的低优先级的消息。其余的优先顺序如下：LM＞IR＞HR＞AR＞TC＞DM。在单一的存储器区域中，低地址段的消息比高地址段的消息享有更高的优先级。

4. 失败点检测 FPD（-）

失败点检测指令用来确定在可接受的时间之内，究竟是其后的梯级中的哪一部分布尔逻辑条件未能为真。输入的时间以十分之一秒计，作为监视时间（T）参数。当 FPD（-）开始运行直到计时超时还没有发现其后的梯级上的条件为真，CQM1 会产生一个非致命错误，同时以正常检查的顺序检查其后的梯级上的每一个布尔逻辑指令。CQM1 会找出第一个为假的语句（即使有多个为假）。

FPD（-）的定时器没有完成位，但是进位标志（SR 25504）会在定时器计时完毕后置位。

CQM1 可以自动为 T 生成一个时间值。详情参考使用手册。

控制（C）参数必须以常数形式输入。C 的低字节必须包含一个惟一的 FAL 编号，该编号可以写入错误缓存及 FAL 存储器区域中。如果 C 参数的最高位是：

0 则 CQM1 在输入目标（D）参数的地址后的第一个存储区域中写入一个 16 位的代码，代码识别由“假”布尔条件语句决定的地址。此代码识别位存储区域和位地址（参考使用手册）。D 地址中的字仅使用两个高位：当地址代码写入下一个字时位 15 置位；当布尔逻辑指令是为假的检查关闭或者常闭（EXAMINE OFF OR NORMALLY CLOSED）指令时

⊖ 除了少数特例，OMRON 的输出指令必须是有条件的。但是使用 SR 25313，可以有效地使输出指令无条件执行。

位 14 置位。

1 CQM1 会将为假的布尔语句的位地址以 BCD ASCII 码的形式写入与输入目标（D）参数相邻的三个地址中，同时会写入一个代码指示到下一个字的常开或常闭。如上所述，CQM1 也会在 D 地址中字的高两位写入信息。FPD（-）接着通过外围端口传送九个字，这九个字以 D 地址中的内容起始。程序员（或者程序）可以在九个地址中的最后四个中写入数据。这九个地址的区域具有如下结构：

- 由 FPD（-）指令控制的地址：
  - D　　如上所述，由 CQM1 控制的高两位；
  - D+1　　高字节：ASCII 中的空格代码（“20”）；低字节：位地址的 MSD
  - D+2　　位地址中的下两个字符；
  - D+3　　位地址中的低位字符；
  - D+4　　高字节：“-”的 ASCII 代码（“2D”）；
    低字节：如果是常开指令，包含 0 的 ASCII 代码（“30”）
    如果是常闭指令，包含 1 的 ASCII 代码（“31”）。
- 由程序员或者程序控制的地址：
  - D+5 至 D+8　最多可以容纳 8 个扩展 ASCII 代码，如果最末位 ASCII 代码为 0D 时可以包含少于 8 个代码（即“XX0D”）。

图 15-3 显示 FPD 指令是如何使用的。IR 00204 在 FPD 指令中触发了定时器的启动。一个＃0030的常数作为 T 参数输入，因此如果在其后的梯形图中没有逻辑语句为真的时间超过 3 秒，则 CQM1 产生一个带有 FAL 码为 55 的非致命中断（从输入的作为控制参数的常数＃0055 的低字节中取得）。CQM1 接着会检查其后的梯形图中的布尔逻辑指令，找出第一个错误的语句（即使存在多个）。如果第一个错误指令正好是含有地址 00214 的检查关闭指令，而且 C 参数（＃0010）最高位为 0，则 FPD 指令会将如下两个字的信息写入地址 DM 0000 与 DM 0001 中：

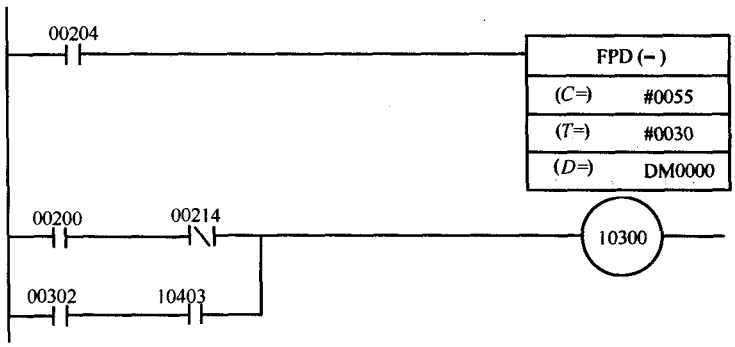


图 15-3　OMRON 的失败点检测指令

- DM 000 0：　1100 0000 0000 0000
  - 位 15　　由于 DM 0001 之后跟有地址而置位；
  - 位 14　　由于指令为一个检查关闭（也被称做常闭）而置位；
  - 其他位　　不使用

- DM 0001：　1000 0000 0010 1110
  - 1000　IR 地址区域的代码（参考使用手册）
  - 0000 0010　字地址的二进制形式：002
  - 1110　位地址的二进制形式：14

5. 框架校验和，FCS (-)

FCS (-) 指令用于从一系列数据字中产生一个校验和[⊖]。它将结果校验和的每 4 位转换成十六进制的字符，接着将十六进制字符转换成 1 字节的 BCD ASCII 代码。例如：

	1100	1111
异或：	1010	0101
二进制结果：	0110	1010
十六进制结果：	6	A
ASCII 码：	36	41

FCS (-) 指令可以再次执行，使用相同的数据再产生一次校验值。如果结果校验值不同，则说明一个或者多个数值改变了。校验和通常被用在数据有可能被破坏的地方，例如数据从一个 PLC 传到另一个时。在传送前产生的校验和会随数据一起发送，接收者再产生一个新的校验和来比较。

FCS (-) 的第一个参数是控制 (C) 参数。如果 C 的位 13 置位，则校验和是使用字节而非字来产生的。如果 C 的位 12 置位，那么在计算校验和中最低字节的使用在最高字节之前。控制参数 (000-999) 的低三个字节的 BCD 码显示了有多少字节或者字包含在校验和的计算过程中。第二个参数是范围首字 (R_1) 地址，表示了数据的起始。第三个参数是目的地址 (D)，表示了代表 8 或者 16 位结果的两个或者四个 ASCII 码的地址。

CQM1 编程器在调试用户程序的过程中会产生一个时间表来协助调试。CQM1 时间表直方图可以随时间同步显示最多 16 位或者三个数据字的状态。为了创建时间表的直方图，可以使用编程器在线连接 CQM1，通过工具菜单来选择时间表命令，接着执行命令。输入一个采样间隔（最小值为 0.3s，即使输入为 0.0）并且选择需要监视的位和字。同时必须指定一个位地址和一个边缘类型（上升或下降），用以告知程序于何处开始采样，也可以输入一个最大为 250 个采样间隔的延时。在触发之前可以输入一个负值的延时，记录至第 249 次采样。当准备就绪后，执行确认指令，选择 Y。

某些 OMRON PLC 比 CQM1 拥有更强大的功能，可以编写程序来记录存储器中的位或者字的值。

15.10 总结

现代的 PLC 保存了广泛的状态信息，可以由用户获得，来帮助未按照预期运行的 PLC 的故障检修。某些 PLC 程序可以使用出错处理程序，该程序可以在 PLC 终止工作之前在主要出错事件中自动执行。非致命错误和数学错误一般不会导致 PLC 终止工作，但是可以编写用户程序来检测这些类型的错误。不要通过改变 PLC 的程序或者配置来纠正错误直到确定该错误不能够通过改变 PLC 外围来解决。

⊖ 校验和的生成是由第一个二进制值与第二个异或，再将结果与下一个异或，直到最后一个。

习题

1. 如果控制系统运行不正常，在考虑修改 PLC 程序或者配置之前，你会做什么？
2. 你所学习的 PLC 有哪些出错识别特性可以告诉你为什么 PLC 终止工作？
3. 当致命错误发生时，PLC 会告诉你哪一个程序正在执行？怎样获取该数据？
4. PLC 会自动执行出错恢复程序么？你可以配置你的 PLC 做这项工作吗？
5. 你的 PLC 会记录哪些不会导致 PLC 终止工作的错误类型？
6. 在你学习编程的过程中，哪些专用指令有助于你寻找逻辑错误？

第 16 章　未来：PLC 前途是否黯淡

16.1　学习目标

本章您将了解到：

- 使 PLC 持续受欢迎的特性；
- 在将 PLC 和其他产品设备整合起来的开放标准中，从单一的传感器-执行器网络到复杂的现场总线网络的发展；
- 使用监控管理和数据采集（SCADA）软件进行的集中管理；
- 软件逻辑：它将取代 PLC？还是只是 PLC 的另一种形式？
- 在没有传感器或者执行器的情况下，仿真测试 PLC 程序；
- 允许在控制器之间快速进行数据交换的反射存储器；
- PLC 内置的运动控制器和过程控制器。

本章的标题是一个双关语。“PLC 的前途是否黯淡？”既问到目前 PLC 发展到什么地步，也问到 PLC 的重要性是否衰落。预测未来是非常冒险的。在 1984 年，我曾经告诉一批面试官，我相信 PLC 将会消亡并被个人电脑所取代。那批面试官必然认为我是一个非常有远见的人，因为我最终得到了那份工作。从那时开始，很明显地，我改变了自己的想法，并且没有一个面试官还在那所雇佣我的公司工作。

人们依然热衷于预测 PLC 将会被个人电脑所取代，然而，PLC 的销售仍在继续增加，因为 PLC 制造商依然在继续改进他们的产品。今天的 PLC 和十年前的 PLC 有显著的不同，未来十年的 PLC 也将明显不同于今天的 PLC。尽管有很多改变，但是今天的 PLC 和过去的 PLC 相比，存在两个重要的共通之处：它们依然传递确定性的控制响应，并且仍是高度可靠的。它们之所以是确定性的，是因为它们基础的扫描循环能使它相对容易确定对被感知状态做出反应的延迟时间。它们之所以是可靠的，是因为它们被设计用于抵抗不如意的环境并被连接到其他设备上。（个人电脑的确定性和可靠性有多少呢？）

16.2　明天的 PLC

明天的 PLC 将会怎样？它将会做到哪些今天的 PLC 无法做到的事情？你能确定的一件事情是，PLC 的某些方面不会发生根本性改变：PLC 依然会是确定性的和可靠的。另一件事情你能确定的就是 PLC 将会根据 PLC 用户的要求来改变而不是根据 PLC 制造商要销售新的型号来改变。工业用户是非常小心的。如果他们正在使用的 PLC 能帮助他们完满的完成工作，他们绝对不会冒着引起产品问题的危险来买一个仅仅有着色彩鲜艳广告的新型号。然而，如果工业用户有信心找到一种相比目前对工作更有帮助的方法，他们一定会探寻能在新系统中运行的 PLC。因此，相对其他产品来说，我们更容易看到 PLC 的未来。我们只需

要从制造业趋势来看，而不需要太多考虑潮流趋势。

在今天的产品世界里，我们看到了什么趋势？适应性的增加，制造过程可见性的增加，对远程的更好控制，当然还有价格的降低。要降低产品的价格，我们可以通过以下途径来实现：降低产品库存，提高生产力，更好的控制产品质量和更好的产品设计。所有这些趋势都要求有更好、更容易的通讯。根据计算机生产商（包括 PLC 生产商）的观点，计算机必须能交换数据，并且能通过程序使用其他计算机上的数据。在这章中，我们讲述的几乎所有的趋势都围绕着 PLC 和其他计算机间更大的相互连通性以及相关的其他趋势。PLC 的制造商们都已经在进行这些改善。个人计算机制造商相对反应更慢，因此 PLC 看起来并没有任何将要消亡的直接危险。

16.3 现场总线和传感器——执行器网络

随着控制器变得越来越强大，它们将在更多领域控制产品生产。这意味着对传感器、执行器和 PLC 控制器分开布线不再可行。由 I/O 模块框架组成的远程 I/O 系统也是一种解决方法，在这个系统里，每个框架都由一个能和单一中央 PLC 控制器交换数据的通讯适配器来控制。但是，远程 I/O 解决方案依然会留给我们一些将大量传感器和执行器连接起来的框架。

传感器/执行器网络，有时也称为*传感网络*，将会变得越来越普通。在传感器/执行器网络里，小的 I/O 模块会相互连接起来，并通过串行通讯电缆连接到网络控制模块中。在广泛分布的 I/O 模块中，你所需要的仅仅是有三根导线的电缆。每个 I/O 模块都能连接在一些数字传感器和执行器上，并且每个 I/O 模块都有自己的节点地址。网络控制器通过将小的数据包写进每个 I/O 模块来控制它的执行器，同时，通过从 I/O 模块中读取小的数据包来读取传感器。根据每个目的，例如分配各自惟一的节点地址，在装置中对单独的 I/O 模块编写程序（有时候仅仅需要设定 DIP 开关）是非常必要的。传感器/执行器网络控制器通常都会直接嵌入在 PLC 中央 I/O 框架中。在 PLC 中央 I/O 框架中，它将 I/O 数据传给 CPU，仿佛它就是一个标准 I/O 模块，这样，我们就不需要为使用远程传感器/执行器模块而编写不同的用户程序。

传感器/执行器网络的接口电路非常便宜，它能非常经济地在单独的传感器和执行器间建立接口。但这里的问题是，如果传感器或者执行器是带有内置的针对某一类型的传感器/执行器网络的接口电路，它将无法在其他类型的传感器/执行器网络中使用。因此，用户必须努力争取采用一个开放的传感器/执行器网络标准。虽然专有的传感器/执行器网络依然非常受欢迎，但 PLC 制造商还是非常支持的。两种更加成功的开放的传感器/执行器网络标准分别是由 Allen-Bradley 设计的基于早期开放式控制器局域网络（CAN）标准并被“贡献”给公共的 DeviceNet，还有就是由与 Siemens 联合的 As-I（传感器/执行器接口）。不用说，即使只有两种传感器/执行器网络，其中一种也是多余的，并且没有一个制造商希望支持一种他们认为将会给竞争者带来优势的标准。

大部分传感器/执行器网络都仅仅善于把数字传感器、执行器和控制器连接起来，因为通常数据包的尺寸都被限制在每个节点只有少数字节。即使我们将一个模拟 I/O 值数字化为一个 10 位数，对于大部分传感器/执行器网络来说，也是太大而难以处理。尽管如此，也有少数传感器/执行器网络能处理更大的数据包，或者有可变的数据包尺寸，因此它们将被

用于连接远程模拟设备和控制器。

如果用户打算购买内置通信接口芯片和数模转换芯片的传感器或者执行器，那么在芯片中植入更多的功能是个很经济的主意，比如植入能够执行校准或者自检的功能。如果你要把传感器和执行器连接到一个网络里，为什么不使用网络把配置字写进芯片里，并使用网络来读取它的状态？为什么不在芯片里建立 ROM 来储存用于描述智能传感器和执行器的数据，这样，控制器就可以读取 ROM 芯片来知道传感器和执行器的类型。如果控制器能查询到连接的装置，我们就能把传感器/执行器网络系统设计成自我配置型（即插即用），你甚至可以设计一个允许传感器和执行器热插拔（无须关掉电源即可插拔）的系统。在早期简单的专有网络的某个发展阶段，仅仅能用复杂的操作系统从远程数字 I/O 读和写到自我配置的网络。传感器/执行器网络的名字已经被现场总线所取代。

几年前，国际标准组织认识到，在建立太多专有系统之前，应该建立一个为达到控制目的而用于传递数字化模拟值的开放标准。在对目标进行了一系列交锋和修订之后，一个由工业代表组成的单独小组承担起发展独立开放现场总线标准的责任。这个小组被称为现场总线基金会（FF），而这个发展中的标准被定义为 SP50，它是 ISA/IEC 标准的结合体。SP50 定义了通讯网络要求的四个层次，并定义了在运用现场总线中 PLC（在第五层）必须做的事情。这个四层结构的模型不仅仅包括用于计算机通讯的基于 ISO 的 OSI（开放系统相连）三层结构模型，还加上了一层新的用户层标准。事实上 SP 50 是一套标准，因此用户可以根据不同功能要求来选择 SP 50 中合适的设备。图 16-1 显示了一个 FF 现场总线的结构。

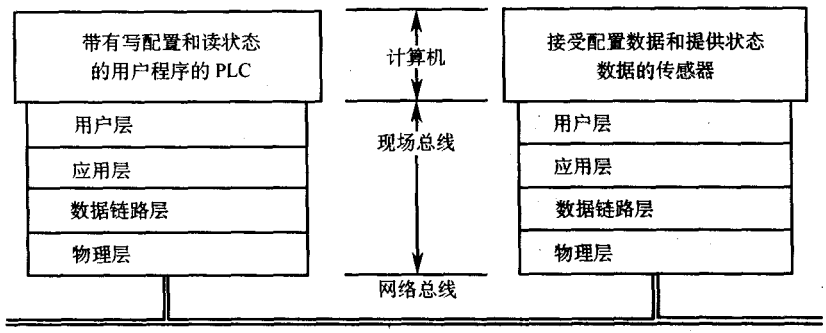


图 16-1 SP 50 现场总线上两个计算机节点

图 16-1 显示了连接在现场总线中的两台计算机。其中一台是 PLC，另一个仅仅是一个传感器。传感器和执行器都必须包含内嵌的控制芯片，这样它们才能足够灵活地通信、使用配置数据、报告状态，更重要的是，执行某一类型设备特定的控制程序。（SP 50 接口模块依然是可用的，因此不灵活的传感器依然可以被连接。）低智能的传感器可能只能从 PLC 中接受定值，或者计算数字化的输出值来让 PLC 读取。而用于电机控制的高级的智能执行器可以从 PLC 中接受位置的设定值，甚至拥有它自己的伺服控制程序（例如，PID 功能块）。它能从现场总线中各个分离的传感器（例如，一个位置传感器）中读取过程变量值，并能通过使用由 PLC 或者现场总线中其他控制器提供的控制参数来驱使电机到达要求的位置。

当你听到现场总线基金会以下的陈述时，一定会感到非常高兴：用户不需要知道应用层和数据链路层如何运作。然而，我们依然要求用户能了解到有关这两个物理层间的一些知识，因为用户必须通过网络总线来连接现场总线接口。H1 型物理层提供了足够取代大部分

传感器/执行器网络的活动级别，而 H2 型物理层提供了足够的速度和数据处理能力来完成上面描述的完整现场总线的所有功能。作为 PLC 程序员需要了解更多的是用户层。用户层通过现场总线来传递和请求数据，并对用户程序指令做出反应，同时自动为系统执行一些现场总线要求的操作。

现场总线系统所执行的一些用户层服务包括以下几个方面：

1）在系统启动或操作过程中，对现场总线设备的自动检测和辨识。每个设备都包含一个设备描述语言格式（DDL）的定义（由制造商安装）。现场总线基金会指定一组虚拟设备格式来限定各种设备定义。

2）通过自动分配地址以及维护地址目录，用户可以编写使用符号名的用户程序。

3）通过输入和输出的循环读和写（就像在 PLC 里）来更新 I/O 映像。

4）无需等待循环更新，直接交换高优先级的消息。当用户程序（在应用层）执行 IEC 1131-5 通讯功能块时，现场总线用户层将做出相应的反应：执行请求的数据交换。有些通讯要求现场总线能中断其他的控制器，这样它就能响应警告消息。

5）在用户程序中，通过运行标准控制功能块来进行远程控制。

6）所有计算机时钟以周期间隔同步。

现场总线基金会的 SP 50 标准是否被普遍接受？（如果你猜错的话，你是否会失去你的工作呢？）SP 50 应该是做得很好的，因为它在几乎所有的专有标准之前已经存在，而且工业界极需一个开放的现场总线标准。在 SP 50 包含 Allen-Bradley 的 ControlNet、Siemens 支持的 Profibus 网络以及一些小公司的网络之前，一些专有网络凭借广泛的应用基础还可以挑战 SP 50。传感器/执行器网络已经拥有一个足够大的安装基础，假如可以改进，人们更喜欢简单地升级它们，而不是用 SP 50 兼容的系统来取代它们。

16.4 SCADA 系统

监控管理和数据采集（SCADA）系统允许用户在个人计算机中把控制器（例如 PLC）中的数据显示出来，并使用个人计算机改变控制器存储空间中的数据值，你也可以使用编程器。然而，从以下一些方面看来，SCADA 系统可以比编程器做得更多：

1）运行在个人计算机上的 SCADA 软件能同时从数台控制器中收集数据；

2）SCADA 软件能对获取的数据进行分析；

3）SCADA 软件提供制图功能，因此用户能操作包含一些特性的操作显示屏。这些特性包括可识别的特性（例如，泵及油罐图表）和使状态解释更容易的特性（例如，油罐图表可以显示其满位而泵可以在开启的时候变色）。如果还有触摸屏等，操作者就可以直接控制过程，而不需要通过键盘和鼠标。

SCADA 软件对已获取的数据进行的数据分析经常是将它们绘制在时基图上。在操作屏上显示的图表能让操作者察觉到系统输出的趋势。更高级的 SCADA 数据操作有时候还包括图表数据的 SPC 分析，以及被控过程在控制范围以外时触发操作警告。

随着控制器网络的发展，人们逐渐要求 SCADA 软件允许管理人员监控操作系统。目前，最好的 SCADA 数据包是由第三方提供的（既不是设备供应商也不是设备购买商），当你读到这里，可能 PLC 制造商已经开始做最大的努力来维持 SCADA 市场了。

16.5 软件逻辑

如果你的个人电脑包含了能模拟 PLC 循环扫描的软件，或者能运行标准 PLC 程序并通过接口卡完成远程 I/O 框架的 I/O 更新，你就不需要在你的 PLC 系统里装备 PLC CPU 模块。你需要一个软件逻辑处理器。应用于个人计算机的软件逻辑软件和接口正是预言 PLC 将会消亡的最新理由。

PLC 技术的发展趋势证明了软件逻辑的成功。让我们先从以下典型的小型制造过程来看看发展周期。

1）在生产车间中大概有一打 PLC，和一个装有 PLC 编程软件的个人计算机以及一系列接口电缆的编程器。个人计算机并不是工控机，因为工控机的费用更高，因此所有者会因为担心个人计算机的损坏而寻求更可靠的系统。

2）通过使用带有内置局域网能力的 PLC 可以组建一个更复杂的系统。人们可以在个人计算机中加入连接 PLC 网络的接口卡。如今，个人计算机可以从工程办公室连接到任何 PLC 上。

3）由于目前个人计算机已经永久地连接到 PLC 网络上，因此我们可以通过安装一些 SCADA 软件来管理整个控制系统。我们也可以安装一些网络分析软件。

4）为了改进整个系统的集成性，我们可以用单独的中央控制框架和 11 个远程 I/O 框架（也可以用由传感器/执行器网络或者现场总线连接起来的远程 I/O）来代替分离的 PLC 控制器。线路要求基本一样，但是你需要为中央 PLC 编写一个程序。为了满足你需要的速度和存储容量，可能你要为中央 PLC 购买一个更大的 CPU，这样，你就可以更自由、灵活地修改远程节点。

5）现在你应该认识到事实上你拥有两个中央计算机：PLC 的 CPU 模块和用于运行程序软件/SCADA 软件、网络分析软件的个人计算机。相对于更昂贵的 PLC CPU 模块来说，个人计算机可能速度更快、存储容量更大。为什么我们不通过在个人计算机中运行软件逻辑来代替在 PLC 上升级 CPU 模块呢？它允许运行在个人计算机上的其他软件更快更完全地接近控制系统的状态。更重要的是，它意味着通过加入标准个人计算机组成部分后，我们能更容易地扩展 PLC 存储器、I/O 功能以及通信通道能力。

事实上，也存在一些观点要求保持真实的 PLC 而不使用软件逻辑控制。我们可以预测，这些观点必然考虑到可靠性以及控制间隔的确定性。软件逻辑控制软件和接口硬件供应商声明这些问题是可以解决的。下面就是一些关于软件逻辑正反两面的观点，你可以自己决定：

反面：对于生产环境，个人计算机母板和电源装置还不够完善，经常会发生故障。

正面：母板已经不再由小型电子供应商生产。Intel 已经获取了全球大半市场。它们的质量更好并更加持久。上一次你看到发生故障的母板（不算连接器）是什么时候？下一代的母板将是密封的模块，而它也将被 PLC 制造商使用。电源依然是一个问题，但是只要你肯多付一点钱就可以配置一台好的计算机电源。当然，你应该在工厂中安装电源调节器，这样譬如毛刺等电源问题就不会影响到电源供应了。

反面：个人计算机操作系统不够稳定。它们经常会将在同一时间运行的所有软件挂起，以至于必须重新启动计算机。

正面：大部分评论家认为从 NT 以后，Windows 的操作系统非常稳定，并值得信赖。

对于那些拒绝相信操作系统的人来说，一些软件逻辑系统在接口卡上运行 PLC 模拟软

件，因此只要电源供应没有被中断，即使计算机被挂起，它依然可以继续操作。事实上，你有 PLC CPU 模块，只是它在个人计算机中，其他个人计算机软件可以更好地访问它。如果你需要关闭个人计算机并重新启动其他计算机程序，同样会有问题，但是至少在控制系统停止前你可以控制它。

反面：个人电脑的控制间隔是不确定的。事实上个人计算机在同一个时间里只可以执行一个程序。假如同时打开几个窗口并且每个都在执行程序，操作系统必须在那些不需要维持确定性的行为的程序之间来回转换。当一个需要响应的情况发生时，或软件逻辑控制器执行要求的控制输出，很难预测两者之间的延迟。

正面：现代操作系统通常具有一些特点，称为易写的软件逻辑控制程序，可以减少或者消除控制间隔的不确定性。即使这些特点还不能使用，但是个人计算机的迅速发展使控制间隔的不确定性变得不明显。

在操作系统之前，可以加载一些软件逻辑控制器软件。软件逻辑程序在可重复的间隔里运行，并且仅仅当不需要软件逻辑功能时，才允许操作系统使用计算机。不确定性只取决于植入电脑中的偏离程序（对于 PLC 要求的数据交换类型不一定要求最优化）。

如果软件逻辑控制器嵌入在接口卡中，那么只有当它需要使用个人计算机里的资源，例如个人计算机或者存储器里的通信通道时，它的控制间隔才会受到个人计算机的影响。如果所有这些特性都嵌入到接口卡中，接口卡的功能将甚至比 CPU 模块更强大。

如果软件逻辑控制能成功取代 PLC，我们也需要对软件逻辑 PLC 进行编程。控制语言可以由软件逻辑程序的供应商来指定，最值得信赖的当然是由主要 PLC 供应商指定的软件逻辑。例如 Rockwell，现在提供了一个称为 softLogix 5 的 Allen-Bradley 的软件逻辑系统，被称为是“PLC-5 系列处理器的一员”。现在所有流行的软件逻辑程序都能提供 IEC 1131-3 标准 PLC 编程语言，因此程序员不必再学习其他计算机编程语言。依然要求有 I/O 模块，而这个市场依然是由主要的 PLC 供应商占领着。

总的来说，即使软件逻辑无法完全取代 PLC，大部分 PLC 用户依然无法说出两者的区别。

16.6 过程仿真

为了帮助编写和调试 PLC 程序，目前一些供应商为个人计算机提供了软件和接口卡，这样，程序员能在个人计算机编写程序来模拟 PLC 控制的系统。我们可以使用 PLC 程序来控制仿真系统，从而测试这个 PLC 程序。

Siemens 提供了被称为 S7-PLCSIM 的过程仿真软件包。你可以通过把用户程序加载到 S7-PLCSIM 里来测试它，因此你甚至不需要 PLC。第三方销售商提供了基于个人计算机的能被真正 PLC 驱动的仿真系统。例如，SST（S-S 技术）提供了被称为 PICS 的程序，以及允许个人计算机像远程 IO 模块一样连接到 PLC-5 上面的接口卡。个人计算机会模拟对在 PLC-5 中运行的用户程序的系统响应。

16.7 反射性存储器

作为标准 PLC 的完整替代品，VMIC 提供了被称为 IOWorks 的软件逻辑程序，同时出售包含了 VMIC RTnet 通信适配器及 VMIC I/O 模块的远程 IO 框架。在中央计算机（运行 IOWorks）与框架之间的数据交换是通过反射性存储器系统完成的。通过反射性存储器，网

络可以对远端框架的 RTnet 适配器的存储器进行直接存储访问（DMA），使 IO 数据在个人计算机与远程框架之间复制，其速率最高可以达到 29Mps。

16.8 OMAC 运动与过程控制

OMAC 这个名字（开放模块化架构控制器）是由美国三大汽车制造商提出的，三大汽车制造商推动了一个开放的 PLC 控制器标准定义的出台。OMAC 控制器定义包含一个标准操作系统（可能是 Windows NT 系统的修改版本），一个标准的编程接口，标准的编程语言（通过 IEC1131-3 定义），以及为运动控制及其他标准机器功能设定的标准功能块。其他控制器的用户都加入了这个体系，包括过程控制设备的用户。过程控制工业的代表正在推动第六种编程语言的采用，以方便标准过程控制的操作。

PLC 厂商做出回应，为它们自己的 PLC 提供了新的 CPU 模块，这些模块带有运动控制 I/O 能力以及内置的高速控制算法。Siemens 生产了一种新的 S7-600 级的 CPU，而 Allen-Bradley 推出了新的 Logix 5 处理器。像 PLCOpen 的标准化组织致力于为 IEC1131 开发扩展标准，这样可以在尽可能不损害已有的基于其他 PLC 编程语言的 IEC（6）1131-3 标准协议的基础上满足连续流程图支持者的需要。

所以，PLC 的前景不会黯淡。

习题

1. 不用把传感器连接到标准的 PLC 的 I/O 模块上，将它们通过类似 DeviceNet（Allen-Bradley 最常用）或者 AS-i（Siemens 最常用）的传感器总线连接到 PLC 上。当然，PLC 应该有传感器总线接口模块，但是与传感器之间的连接可以是不同的。描述一下（或者画出电路图）如何将八个传感器连接到传感器总线接口模块上？
2. 现场总线网络中的过程通过网络控制器的作用彼此同步，网络控制器如何保持系统同步？
3. 现成总线网络中的过程彼此通信，就好像彼此是不同的电脑一样。那么过程与电脑之间究竟有什么区别？
4. 除了有规律地预定的数据交换外，现场总线还允许_____类型的消息的传递。
5. 每一个连接到现场总线网络的组件都必须有 DDL 标识符，为什么？
6. 1）在不久的将来，个人计算机是否会在工厂的车间取代 PLC？
   2）如果个人电脑确实取代了 PLC，PLC 的程序员是否会发现其中的不同？解释一下。

附录 A　Allen-Bradley PLC-5 状态文件结构体

涉及型号	包括这些 Allen-Bradley 处理器
经典 PLC-5 处理器	PLC-5/10、-5/12、-5/15、-5/25 和-5/VME 处理器
增强型 PLC-5 处理器	PLC-5/11、-5/20、-5/30、-5/40、-5/40L、-5/60、-5/60L 和-5/80 处理器 重要：除非特别说明，增强型 PLC-5 处理器包括以太网 PLC-5 和 VME PLC-5 处理器
以太网 PLC-5 处理器	PLC-5/20E、-5/40E 和-5/80E 处理器
VME PLC-5 处理器	PLC-5/V30、-5/V40、-5/V40L 和-5/V80 处理器。参见 PLC-5/VMEVMEbus 可编程控制器用户手册，1785-6.5.9，可获取更多信息

表 12-A　处理器状态文件地址

状态文件的字		存　储
S：0		算术标志 • 位 0＝进位 • 位 1＝溢出 • 位 2＝零位 • 位 3＝符号
S：1		处理器状态和标志
S：2		开关设置信息： • 位 0～5：DH＋站点号 • 位 7：1＝扫描器；0＝适配器 • 位 11，12：硬件寻址 位 12　位 11 0　0　无效 1　0　1/2-插槽 0　1　1-插槽 1　1　2-插槽 • 位 13：1＝从 EEPROM 中装载 • 位 14：1＝RAM 备份未配置 • 位 15：1＝内存未被保护
S：3～S：6		激活的节点列表 字　位　DH＋站点＃ 3　0～15　00～17 4　0～15　20～37 5　0～15　40～57 6　0～15　60～77

（续）

状态文件的字		存　储
S：7	（经典 PLC-5 和增强型 PLC-5 处理器）	全局状态位： • 低 8 位－框架 0～7 的框架出错位 • 高 8 位－框架 0～7 的框架队列满位
S：32	（PLC-5/40、-5/40L、-5/40E、-5/V40、-5/V40B、-5/V40L、PLC-5/60、-5/60L、-5/80、-5/80E、PLC-5/V80）	• 低 8 位－框架 10～17（八进制）的框架出错位 • 高 8 位－框架 10～17 的框架队列满位
S：34	（PLC-5/60、-5/60L、-5/80、-5/80E、-5/V80）	• 低 8 位－框架 20～27（八进制）的框架出错位 • 高 8 位－框架 20～27 的框架队列满位
S：8		上次程序扫描（单位 ms）
S：9		最大程序扫描（单位 ms）
S：10		次要出错（字 1）
S：11		主要出错
S：12		出错代码
S：13		出错发生的程序文件
S：14		出错发生的梯级数
S：15	PLC-5/VME 处理器	VME 状态文件
S：16		I/O 状态文件
S：17	增强型 PLC-5 处理器	次要出错（字 2）
S：18		处理器时钟年
S：19		处理器时钟月
S：20		处理器时钟日
S：21		处理器时钟小时
S：22		处理器时钟分钟
S：23		处理器时钟秒
S：24		变址寻址偏移量
S：25	（PLC-5/12、-5/15、-5/25）	I/O 适配器映像文件
S：26		用户控制位： • 位 0 和 1：启动程序 • 位 2：本地框架地址
S：27	（经典 PLC-5 和增强型 PLC-5 处理器）	框架控制位： • 低 8 位－框架 0～7 的 I/O 框架禁止位 • 高 8 位－框架 0～7 的 I/O 框架复位位
S：33	（PLC-5/40、-5/40L、-5/40E、-5/V40、-5/V40L、PLC-5/60、-5/60L、-5/80、-5/80E、-5/V80）	• 低 8 位－框架 10～17 的 I/O 框架禁止位 • 高 8 位－框架 10～17 的 I/O 框架复位位
S：35	（PLC-5/60、-5/60L、-5/80、-5/80E、-5/V80）	• 低 8 位－框架 20～27 的 I/O 框架禁止位 • 高 8 位－框架 20～27 的 I/O 框架复位位

（续）

状态文件的字		存　储
S：28		程序看门狗设置值
S：29		错误程序文件
S：30		STI 设置值
S：31		STI 文件号
S：46	（增强型 PLC-5 处理器）	PII 程序文件号
S：47	（增强型 PLC-5 处理器）	PII 模块组
S：48	（增强型 PLC-5 处理器）	PII 位掩模
S：49	（增强型 PLC-5 处理器）	PII 比较值
S：50	（增强型 PLC-5 处理器）	PII 减计数器
S：51	（增强型 PLC-5 处理器）	PII 改变位
S：52	（增强型 PLC-5 处理器）	PII 自从上次中断发生的事件
S：53	（增强型 PLC-5 处理器）	STI 扫描时间（单位 ms）
S：54	（增强型 PLC-5 处理器）	STI 最大扫描时间（ms）
S：55	（增强型 PLC-5 处理器）	PII 上次扫描时间（ms）
S：56	（增强型 PLC-5 处理器）	PII 最大扫描时间（ms）
S：57	（增强型 PLC-5 处理器）	用户程序校验和（ms）
S：59	（PLC-5/40L、-5/60L）	扩展的本地 I/O 通道离散传输扫描（ms）
S：60	（PLC-5/40L、-5/60L）	扩展的本地 I/O 通道离散最大扫描（ms）
S：61	（PLC-5/40L、-5/60L）	扩展的本地 I/O 通道离散块传输扫描（ms）
S：62	（PLC-5/40L、-5/60L）	扩展的本地 I/O 通道最大块传输扫描（ms）
S：77	（增强型 PLC-5 处理器）	通信清理函数的通信时间片（ms）
S：78	（增强型 PLC-5 处理器）	MCP 后的 I/O 扫描
S：79	（增强型 PLC-5 处理器）	MCP 禁止位
S：80～S：127	（增强型 PLC-5 处理器）	MCP 文件号 MCP 扫描时间（ms） MCP 最大扫描时间（ms）

附录 B　Allen-Bradley SLC 500 状态文件结构体

状态文件 S：包含下列字

字	功能（应用于所有的处理器）
S：0	算术标志
S：1	处理器模式状态/控制
S：2	STI 位/DH485 Comms
S：3L	当前/上个扫描时间
S：3H	看门狗扫描时间
S：4	自由运行的时钟
S：5	次要出错位
S：6	主要出错代码
S：7，S：8	延缓编码/延缓文件
S：9，S：10	激活节点（DH-485）
S：11，S：12	I/O 插槽启用
S：13，S：14	数学寄存器
S：15L	节点地址
S：15H	波特率

字	功能（应用于 SLC5/02、SLC5/03、SLC5/04 处理器）
S：16，S：17	单步检测－开始－梯级/文件
S：18，S：19	单步检测－断点－梯级/文件
S：20，S：21	检测－出错/断电－梯级/文件
S：22	观察到的最大扫描时间
S：23	平均扫描时间
S：24	索引寄存器
S：25，S：26	I/O 中断禁止
S：27，S：28	I/O 中断允许
S：29	用户出错程序文件号
S：30	可选择的定时中断设置值
S：31	可选择的定时中断文件号
S：32	I/O 中断执行

字	功能（应用于 SLC5/03、SLC5/04 处理器）
S：33	扩展处理器状态和控制字
S：34	禁用 Passthru（仅限 SLC5/04）
S：35	上 1 毫秒扫描时间
S：36	扩展的次要错误位
S：37	时钟/日历　年
S：38	时钟/日历　月
S：39	时钟/日历　日
S：40	时钟/日历　小时
S：41	时钟/日历　分钟
S：42	时钟/日历　秒
S：43	STI 中断时间（SLC5/03 和 SLC5/04）
S：44	I/O 事件中断时间（SLC5/03 和 SLC5/04）
S：45	DII 中断时间（SLC5/03 和 SLC5/04）
S：46	离散输入中断－文件号
S：47	离散输入中断－插槽号
S：48	离散输入中断－位掩模
S：49	离散输入中断－比较值
S：50	离散输入中断－预设值
S：51	离散输入中断－返回掩模
S：52	离散输入中断－累加器
S：53，S：54	保留
S：55	上次 DII 扫描时间
S：56	观测到的最大 DII 扫描时间
S：57	操作系统分类号
S：58	操作系统序列
S：59	操作系统 FRN
S：60	处理器分类号
S：61	处理器序列
S：62	处理器校正
S：63	用户程序类型
S：64	用户程序功能索引
S：65	用户 RAM 大小
S：66	闪存 EEPROM 大小
S：67，S：68	通道 0 激活节点

字	功能（应用于 SLC5/04 处理器）
S：69～S：82	保留
S：83～S：86	DH＋活动节点（仅在通道 1 SLC5/04）
S：87～S：98	保留
S：99	全局状态字
S：100～S：163	全局状态文件

附录 C　OMRON CQM1 SR 和 AR 存储区域

SR 区域

字	位	功　能
SR 244	00～15	输入中断 0 计数器模式 SV 当输入中断 0 用在计数器模式时的 SV（4 位十六进制数，0000～FFFF）。（当输入中断 0 不用在计数器模式时能被用作工作位）
SR 245	00～15	输入中断 1 计数器模式 SV 当输入中断 1 用在计数器模式时的 SV（4 位十六进制数，0000～FFFF）。（当输入中断 1 不用在计数器模式时能被用作工作位）
SR 246	00～15	输入中断 2 计数器模式 SV 当输入中断 2 用在计数器模式时的 SV（4 位十六进制数，0000～FFFF）。（当输入中断 2 不用在计数器模式时能被用作工作位）
SR 247	00～15	输入中断 3 计数器模式 SV 当输入中断 3 用在计数器模式时的 SV（4 位十六进制数，0000～FFFF）。（当输入中断 3 不用在计数器模式时能被用作工作位）
SR 248	00～15	输入中断 0 计数器模式 PV 减一 当输入中断 0 被用在计数器模式时计数器 PV－1（4 位十六进制数）
SR 249	00～15	输入中断 1 计数器模式 PV 减一 当输入中断 1 被用在计数器模式时计数器 PV－1（4 位十六进制数）
SR 250	00～15	输入中断 2 计数器模式 PV 减一 当输入中断 2 被用在计数器模式时计数器 PV－1（4 位十六进制数）
SR 251	00～15	输入中断 3 计数器模式 PV 减一 当输入中断 3 被用在计数器模式时计数器 PV－1（4 位十六进制数）
SR 252	00	高速计数器 0 复位位
	01	CQM1-CPU43-E：高速计数器 1 复位位 打开时复位高速计数器 1（端口 1）的 PV 值 CQM1-CPU44-E：绝对高速计数器 1 初始补偿位 打开时设置绝对高速计算器 1（端口 1）初始补偿。当补偿值设置在 DM 6611 中时自动关闭
	01	CQM1-CPU43-E：高速计数器 2 复位位 打开时复位高速计数器 2（端口 2）的 PV 值 CQM1-CPU44-E：绝对高速计数器 1 初始补偿位 打开时设置绝对高速计算器 2（端口 2）初始补偿。当补偿值设置在 DM 6612 中时自动关闭

（续）

字	位	功　能
SR 252	03～07	未使用
	08	外设端口复位位 打开时复位外设端口（当外设设备被连接时无效）。当复位完自动关闭
	09	RS-232C 端口复位位 打开时复位 RS-232C 端口。复位后自动关闭
	10	PLC 设置复位位 打开时初始化 PLC 设置（DM 6600～DM 6655），复位后自动关闭。仅当 PLC 在编程模式时有效
	11	强制状态保持位 关：从程序模式转换到监控模式时，强制置位/复位位清除 开：从程序模式转换到监控模式时，强制置位/复位位保持
	12	I/O 保持位 关：开始或停止操作时，IR 和 LR 位复位 开：开始或停止操作时，IR 和 LR 位保持
	13	未使用
	14	错误日志复位位 打开时清除错误日志。操作完成后自动关闭
	15	输出关闭位 关：正常输出状态 开：所有输出关闭
SR 253	00～07	FAL 错误代码 当错误发生时错误代码（一个两位的数字）被保存在这里。当 FAL（06）或 FAL（07）执行时，FAL 数被存在这里。通过执行 FAL00 指令或从外设中清除错误，这个字可以被复位（到 00）
	08	低电池标志位 当 CPU 电池电压下降时打开
	09	循环时间溢出标志位 循环时间溢出时打开（即，当循环时间超过 100ms 时）
	10～12	未使用
	13	总是开标志位
	14	总是关标志位
	15	第一循环标志位 第一循环开始操作时打开
SR 254	00	一分钟时钟脉冲（30s 开，30s 关）
	01	0.02s 时钟脉冲（0.01s 开；0.01s 关）
	02～03	未使用

（续）

字	位	功　能
SR 254	04	CQM1-CPU4□-E：溢出标志（OF） 计算结果超出有符号二进制数上限时打开
	05	CQM1-CPU4□-E：下溢标志（UF） 计算结果超出有符号二进制数下限时打开
	06	微分监控完成标志 微分监控完成时打开
	07	STEP（08）执行标志 仅在基于 STEP（08）的过程的开始一个周期打开
	08	HKY（—）执行标志 HKY（—）执行期间打开
	09	7SEG（88）执行标志 7SEG（88）执行期间打开
	10	DSW（87）执行标志 DSW（87）执行期间打开
	11～14	未使用
	15	CQM1-CPU43-E：脉冲 I/O 错误标志（FALS：9C） 当使用端口 1 或 2 的脉冲 I/O 功能出现错误时打开 CQM1-CPU44-E：绝对高速计数器错误标志（FALS：9C） 当使用端口 1 或 2 的绝对高速计数器出现错误时打开
SR 255	00	0.1 秒时钟脉冲（0.05s 开；0.05s 关）
	01	0.2 秒时钟脉冲（0.1s 开；0.1s 关）
	02	1 秒时钟脉冲（0.5s 开；0.5s 关）
	03	指令执行错误（ER）标志 指令执行期间发生错误时打开
	04	进位（CY）标志 指令的结果有进位时打开
	05	大于（GR）标志 比较操作的结果是“大于”时打开
	06	等于（EQ）标志 比较操作的结果是“等于”时打开
	07	小于（LE）标志 比较操作的结果是“小于”时打开
	08～15	未使用

注意：以下字不能写入：SR 248～SR 251，SR 253～SR 255

AR 区域

字	位	功　能
AR 00～AR 03		未使用
AR 04	08～15	CQM1-CPU43/44-E：脉冲 I/O 或绝对高速计数器状态编码 00：正常 01，02：硬件错误 03：PLC 设置错误 04：当脉冲输出时 PLC 停止
AR 05	00～07	CQM1-CPU43/44-E：高速计数器 1 范围比较标志 00 开：计数器 PV 在比较范围 1 01 开：计数器 PV 在比较范围 2 02 开：计数器 PV 在比较范围 3 03 开：计数器 PV 在比较范围 4 04 开：计数器 PV 在比较范围 5 05 开：计数器 PV 在比较范围 6 06 开：计数器 PV 在比较范围 7 07 开：计数器 PV 在比较范围 8
	08	CQM1-CPU43/44-E：高速计数器 1 比较标志 关：停止 开：比较
	09	CQM1-CPU43/44-E：高速计数器 1 溢出/下溢标志 关：正常 开：溢出或下溢
	10～11	未使用
	12～15	CQM1-CPU43-E：端口 1 脉冲输出标志 12 开：减速指定（关：没有指定） 13 开：脉冲数指定（关：没有指定） 14 开：脉冲输出完成（关：没有完成） 15 开：脉冲输出递增（关：没有脉冲输出）
AR 06	00～15	CQM1-CPU43/44-E：高速计数器 2/端口 2 脉冲输出标志 与 AR 05 中的高速计数器 V 端口 1 脉冲输出标志一致
AR 07	00～11	未使用
	12	DIP 开关 pin6 标志 开：CPU 的 DIP 开关 pin6 关 关：CPU 的 DIP 开关 pin6 开
	13～15	未使用
AR 08	00～03	RS-232C 通讯错误码（1 位数） 当运行 LSS/SSS 的计算机连接到外设端口时编码为“F”
	04	RS-232C 错误标志 当 RS-232C 通信错误发生时打开
	05	RS-232C 传输允许标志 仅在主机链接，使用 RS-232C 通信时有效

（续）

字	位	功　能
AR 08	06	RS-232C 接收完成标志 仅在使用 RS-232C 通信时有效
	07	RS-232C 接收溢出标志 仅在使用 RS-232C 通信时有效
	08～11	外围设备错误码（1 位数） 当运行 LSS/SSS 的计算机连接到外设端口时编码为“F”
	12	外围设备错误标志 当外围设备通信错误发生时打开
	13	外围设备传输允许标志 仅在主机链接，使用 RS-232C 通信时有效
	14	外围设备接收完成标志 仅在使用 RS-232C 通信有效
	15	外围设备接收溢出标志 仅在使用 RS-232C 通信有效
AR 09	00～15	RS-232C 接收计数器 4 位 BCD 数；仅当 RS-232C 通信被使用时有效
AR 10	00～15	外围设备接收计数器 4 位 BCD 数；仅当RS-232C 通信被使用时有效
AR 11	00～07	高速计数器 0 范围比较标志 00 开：计数器 PV 值在比较范围 1 01 开：计数器 PV 值在比较范围 2 02 开：计数器 PV 值在比较范围 3 03 开：计数器 PV 值在比较范围 4 04 开：计数器 PV 值在比较范围 5 05 开：计数器 PV 值在比较范围 6 06 开：计数器 PV 值在比较范围 7 07 开：计数器 PV 值在比较范围 8
	08～15	未使用
AR 12	00～15	未使用
AR 13	00	存储卡安装标志 当电源打开时存储卡已安装时该位打开
	01	有时钟标志 如果带有时钟的存储卡被安装时打开
	02	存储卡写保护标志 当安装了 EEPROM 存储卡并写保护或者安装了 EPROM 存储卡时打开
	03	未使用
	04～07	存储卡编码（1 位数） 0：无存储卡被安装 1：EEPROM，4K 字存储卡安装 2：EEPROM，8K 字存储卡安装 4：EPROM 型存储卡安装

（续）

字	位	功　能
	08～15	未使用
AR 14	00	CPU 到存储卡传输位 从 CPU 传输到存储卡时打开，操作结束时自动关闭
	01	存储卡到 CPU 传输位 从存储卡传输到 CPU 时打开，操作结束时自动关闭
	02	存储卡比较标志 当 PLC 的内容和存储卡比较时打开，操作结束时自动关闭
	03	存储卡比较结果标志 开：发现差别或不能比较 关：内容相同
	04～11	未使用
	12	编程模式传输错误标志 由于处于程序模式，传输不能被执行时打开
	13	写保护错误标志 由于写保护，传输不能被执行时打开
	14	容量不足标志 由于目的地的容量不足，传输不能被执行时打开
	15	没有程序标识符 由于存储卡中没有程序，传输不能被执行时打开
AR 15	00～07	存储卡程序模式 编码（2 位数）显示储存在存储卡中的程序的大小 00：没有程序或没有存储卡被安装 04：程序小于 3.2K 字长 08：程序小于 7.2K 字长
	08～15	CPU 程序模式 编码（2 位数）显示储存在 CPU 中的程序的大小 04：程序小于 3.2K 字长 08：程序小于 7.2K 字长
AR 16	00～10	未使用
	11	PLC 设置初始化标志 当一个校验和错误在 PLC 设置区域发生时打开，然后所有设置恢复到默认状态
	12	程序无效标志 当一个校验和错误在 UM 区域发生时打开，或者当一个不正确的指令被执行时打开
	13	指令表初始化标志 当一个校验和错误在指令表发生时打开，然后所有设置恢复到默认状态
	14	存储卡增加标志 若电源打开时存储卡已安装，该位打开
	15	存储卡传输错误标志 当 DIP 开关第二个脚置为开时（也就是说，设置为上电时自动传输存储卡的内容），如果一个传输不能被成功执行时打开

（续）

字	位	功　能
AR 17	00～07	当前时间的“分钟”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
	08～15	当前时间的“小时”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
AR 18	00～07	当前时间的“秒”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
	08～15	当前时间的“分钟”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
AR 19	00～07	当前时间的“小时”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
	08～15	当前时间的“日期”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
AR 20	00～07	当前时间的“月”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
	08～15	当前时间的“年”部分，2 位 BCD 值（仅当带有时钟功能的存储卡被安装时有效）
AR 21	00～07	当前时间的“星期几”部分，2 位 BCD 值［00：周日～06：周六］（仅当带有时钟功能的存储卡被安装时有效）
	08～12	未使用
	13	30s 调节位 （仅当带有时钟功能的存储卡被安装时有效）
	14	时钟停止位 （仅当带有时钟功能的存储卡被安装时有效）
	15	时钟设置位 （仅当带有时钟功能的存储卡被安装时有效）
AR 22	00～07	输入字 输入位的字号（2 位 BCD 值）
	08～15	输出字 输出位的字号（2 位 BCD 值）
AR 23	00～15	断电计数器（4 位 BCD 值） 记录电源被关闭的次数，从外部设备写“0000”可清除计数
AR 24	00	上电 PLC 设置错误标志 当在 DM 6600～DM 6614 中（在通电时，PLC 设置被读取的区域）存在一个错误时打开
	01	启动 PLC 设置错误标志 当在 DM 6615～DM 6644 中（在开始操作时，PLC 设置被读取的区域）存在一个错误时打开
	02	运行 PLC 设置错误标志 当在 DM 6645～DM 6655 中（PLC 设置总是被读取的区域）存在一个错误时打开
	03，04	未使用
	05	长循环时间标志 如果实际的循环时间长于在 DM 6619 中设置的循环时间时打开
	06，07	未使用
	08～15	显示一个检测到的 I/O 总线错误字编码（2 位十六进制） 00～07：对应输入字 000～007 80～87：对应输出字 100～107 FF：结束覆盖不能被确认

（续）

字	位	功　能
AR 25	00～07	未使用
	08	FPD（－）教学位
	09～15	未使用
AR 26	00～15	最大循环时间（4 位 BCD 值） 存储从操作开始以来的最长循环时间。它在开始时被清除，而不是在操作结束时。 单位由 DM6618 的设置决定。默认是 0.1ms；“10ms”设置：0.1ms； “100ms”设置：1ms；“1s”设置：10ms
AR 27	00～15	当前循环时间（4 位 BCD 值） 存储操作过程中最近一次循环的时间。当操作停止时，当前循环时间不被清除。 单位由 DM6618 的设置决定。默认是 0.1ms；“10ms”设置：0.1ms； “100ms”设置：1ms；“1s”设置：10ms

附录 D　Allen-Bradley 比较指令算子

操　作	使用的算子
二进制运算	+、−、*、|
	OR、* *
	AND、XOR
一元运算	-（求反）
	LN
	FRD、TOD、DEG、RAD、SQR、NOT、LOG、SIN、COS、TAN、ASN、ACS、ATN
比较运算	=、<、>
	<>、<=、>=

附录 E　Allen-Bradley 计算指令算子和优先权

在 CPT 表达式使用的有效操作

类型	算子	描　述	操作例子
复制	无	从 A 复制到 B	在表达中输入源地址，在目的中输入目的地址
清除	无	设置一个值为 0	0（在表达中输入 0）
算术	+	加	2+3 2+3+7（增强型 PLC-5 处理器）
	−	减	12−5 （12−5）−7（增强型 PLC-5 处理器）
	*	乘	5*2 6*（5*2）（增强型 PLC-5 处理器）
	|	除	24|6 （24|6）*2（增强型 PLC-5 处理器）
	-	求反	-N7:0
	SQR	平方根	SQR N7:0
	**	指数*（x 的 y 次幂）	10**3
	LN	自然对数*	LN F8:20
	LOG	以 10 为底的对数*	LOG F8:3
三角函数	ACS	反余弦*	ACS F8:18
	ASN	反正弦*	ASN F8:20
	ATN	反正切*	ATN F8:22
	COS	余弦*	COS F8:14
	SIN	正弦*	SIN F8:12
	TAN	正切*	TAN F8:16
位逻辑运算符	AND	位与	D9:3 AND D10:4
	OR	位或	D10:4 OR D10:5
	XOR	位异或	D9:5 XOR D10:4
	NOT	位非	NOT D9:3
转换符	FRD	从 BCD 转换到二进制*	FRD N7:0
	TOD	从二进制转换到 BCD*	TOD N7:0
	DEG	从弧度转换为角度	DEG F8:8
	RAD	从角度转换为弧度	RAD F8:10

注：* 仅在增强型 PLC-5 中存在

CPT 表达式的操作顺序

顺序	操作符	描　述
1	* *	指数（x^y），仅在增强型 PLC-5 中
2	-	取反
	NOT	位非
3	*	乘
	|	除
4	+	加
	—	减
5	AND	位与
6	XOR	位异或
7	OR	位或

附录 F　Siemens S7 被数学和逻辑操作影响的状态位

评估状态字中的位　　数学指令影响状态字中的下列位：

- CC1 和 CC0
- OV
- OS

有效结果　表中位栏的符号（—）表示问题中的位不受整型数学操作的影响

整型数学结果在有效范围内的状态字的位状态

整型数（16 位）和双整型数（32 位）结果的有效范围	状态字的位			
	CC1	CC0	OV	OS
0	0	0	0	—
I：−32，768<＝结果<0（负数） D：−2，147，483，648<＝结果<0（负数）	0	1	0	—
I：32，767>＝结果>0（正数） D：2，147，483，647>＝结果>0（正数）	1	0	0	—

无效结果

整型数学结果不在有效范围内的状态字的位状态

不在双整数（32 位）结果的有效范围	状态字的位			
	CC1	CC0	OV	OS
I：结果>32，767（正数） D：结果>2，147，483，647（正数）	1	0	1	1
I：<−32，768（负数） D：结果<−2，147，483，648（负数）	0	1	1	1

双整型数学指令＋D、/D 和 MOD92 的状态字的位状态

指　令	状态字的位			
	CC1	CC0	OV	OS
＋D：结果＝−4，294，967，296	0	0	1	1
/D 或 MOD：发生了被 0 除	1	1	1	1

在有效范围里浮点数结果的状态字的位状态

浮点结果有效范围（32 位）	状态字的位			
	CC1	CC0	OV	OS
+0，−0	0	0	0	—
−3.402823E+38<结果<−1.175494E−38（负数）	0	1	0	—
+1.175494E−38<结果<3.402823E+38（正数）	1	0	0	—

不在有效范围里浮点数结果的状态字的位状态

不在浮点结果有效范围（32 位）	状态字的位			
	CC1	CC0	OV	OS
−1.175494E−38<结果<−1.401298E−45（负数）下溢	0	0	1	1
+1.401298E−45<结果<+1.175494E−38（正数）下溢	0	0	1	1
结果<−3.402823E+38（负数）溢出	0	1	1	1
结果>−3.402823E+38（正数）溢出	1	0	1	1

在一个比较指令后的 CC1 和 CC0 位的设置

条　件	状　态		可能用到的指令 A，O，X，AN，ON，XN
	CC1	CC0	
ACCU 2>ACCU 1	1	0	>0
ACCU 2<ACCU 1	0	1	<0
ACCU 2=ACCU 1	0	0	= =0
ACCU 2<>ACCU 1	0 或 1	1 或 0	<>0
ACCU 2>=ACCU 1	1 或 0	0 或 0	>=0
ACCU 2<=ACCU 1	0 或 0	1 或 0	<=0

在一个实数比较指令（IEEE）32 位浮点数后的状态字的位设置

条　件	CC1	CC0	OV	OS
= =	0	0	0	无用
<>	0 或 1	1 或 0	0	无用
>	1	0	0	无用
<	0	1	0	无用
>=	1 或 0	0 或 0	0	无用
<=	0 或 0	1 或 0	0	无用
UO	1	1	1	1

附录 G　Siemens S7 系统函数（FC）、系统功能块（SFB）和 IEC 函数（FC）

系统函数，用数字排列

序　号	缩　写　名	功　　能
SFC0	SET _ CLK	设置系统时钟
SFC1	READ _ CLK	读系统时钟
SFC2	SET _ RTM	设置运行时间记录器
SFC3	CTRL _ RTM	开始/停止运行时间记录器
SFC4	READ _ RTM	读取运行时间记录器
SFC5	GADR _ LGC	询问一个通道的逻辑地址
SFC6	RD _ SINFO	读取 OB 开始信息
SFC9	EN _ MSG	允许块相关、符号相关和组状态信息
SFC10	DIS _ MSG	禁止块相关、符号相关和组状态信息
SFC13	DPNRM _ DG	读取 DP 从站的诊断数据
SFC14	DPRD _ DAT	读取一个标准 DP 从站的一致数据
SFC15	DPWR _ DAT	写一致数据到一个标准 DP 从站
SFC17	ALARM _ SQ	生成通知块相关信息
SFC18	ALARM _ S	产生永久通知块相关信息
SFC19	ALARM _ SC	询问上个 ALARM _ SQ 信息的通知状态
SFC20	BLKMOV	复制变量
SFC21	FILL	初始化一个存储器区域
SFC22	CREAT _ DB	创建数据块
SFC23	DEL _ DB	删除数据块
SFC24	TEST _ DB	检测数据块
SFC25	COMPRESS	压缩用户存储器
SFC26	UPDAT _ PI	更新过程映像更新表
SFC27	CPDAT _ PO	更新过程映像输出表
SFC28	SET _ TINT	设置时间-天的中断

（续）

序 号	缩 写 名	功 能
SFC29	CAN _ TINT	取消时间-天的中断
SFC30	ACT _ TINT	激活时间-天的中断
SFC31	QRY _ DINT	询问时间-天的中断
SFC32	SRT _ DINT	开始延时中断
SFC33	CAN _ DINT	取消延时中断
SFC34	QRY _ DINT	询问延时中断
SFC35	MP _ ALM	触发复合计算中断
SFC36	MSK _ FLT	屏蔽同步错误
SFC37	DMSK _ FLT	不屏蔽同步错误
SFC38	READ _ ERR	读取错误寄存器
SFC39	DIS _ IRT	禁止新中断和异步错误
SFC40	EN _ IRT	允许新中断和异步错误
SFC41	DIS _ AIRT	延时高优先级中断和异步错误
SFC42	EN _ AIRT	允许高优先级中断和异步错误
SFC43	RE _ TRIGR	重新触发循环时间监控
SFC44	REPL _ VAL	传输到累加器 1 的替代值
SFC46	STP	使 CPU 停止
SFC47	WAIT	用户程序延时执行
SFC48	SNC _ RTCB	同步从站时钟
SFC49	LGC _ GADR	询问属于某个逻辑地址的模块插槽
SFC50	RD _ LGADR	询问一个模块的所有逻辑地址
SFC51	RDSYSST	读一个系统状态列表或部分列表
SFC52	WR _ USMSG	写一个用户定义诊断事件到诊断缓存
SFC55	WR _ PARM	写动态参数
SFC56	WR _ DPARM	写默认参数
SFC57	PARM _ MOD	分配参数到一个模块
SFC58	WR _ REC	写一个数据记录
SFC59	RD _ REC	读一个数据记录
SFC60	GD _ SND	发送一个 GD 包
SFC61	GD _ RCV	获取一个收到的 GD 包
SFC62	CONTROL	询问一个属于通信 SFB 实例连接的状态
SFC63	AB _ CALL	编译编码块
SFC64	TIME _ TCK	读系统时间
SFC65	X _ SEND	发送数据到本地 S7 站点外的通信伙伴

（续）

序　号	缩　写　名	功　　能
SFC66	X_RCV	接收来自本地 S7 站点外的通信伙伴的数据
SFC67	X_GET	读取来自本地 S7 站点外的通信伙伴的数据
SFC68	X_PUT	向本地 S7 站点外的通信伙伴写入数据
SFC69	X_ABORT	中止与本地 S7 站点外的通信伙伴的连接
SFC72	I_GET	读取一个来自本地 S7 站点内的通信伙伴的数据
SFC73	I_PUT	向一个来自本地 S7 站点内的通信伙伴写入数据
SFC74	I_ABORT	中止和来自本地 S7 站点内的通信伙伴的连接
SFC79	SET	设置输出范围
SFC80	RSET	复位输出范围

* SFC63 的“AB_CALL”仅存在于 CPU614，更详细的解释，参考相应的手册。

系统功能块，按数字排列

序　号	缩　写　名	功　　能
SFB0	CTU	增计数
SFB1	CTD	减计数
SFB2	CTUD	增/减计数
SFB3	TP	产生一个脉冲
SFB4	TON	产生一个开延时
SFB5	TOF	产生一个关延时
SFB8	USEND	数据的不协调发送
SFB9	URCV	数据的不协调接收
SFB12	BSEND	发送分段数据
SFB13	BRCV	接收分段数据
SFB14	GET	读取来自一个远程 CPU 的数据
SFB15	PUT	向一个远程 CPU 写入数据
SFB16	PRINT	发送数据到打印机
SFB19	START	初始化一个远程设备的完全重启
SFB20	STOP	停止一个远程设备
SFB21	RESUME	初始化一个远程设备的重启
SFB22	STATUS	询问一个远程伙伴的状态
SFB23	USTATUS	接收来自一个远程伙伴的状态
SFB29	HS_COUNT	计数器（高速计数器，集成功能）
SFB30	FREQ_MES	频率计（集成功能）
SFB32	DRUM	实施一个顺序器

（续）

序 号	缩 写 名	功 能
SFB33	ALARM	生成带有通知显示的块相关信息
SFB34	ALARM_8	生成不带 8 信号值的块相关信息
SFB35	ALARM_8P	生成带 8 信号值的块相关信息
SFB36	NOTIFY	生成不带有通知显示的块相关信息
SFB37	AR_SEND	发送档案数据
SFB38	HSC_A_B	计数器 A/B（集成功能）
SFB39	POS	位置（集成功能）
SFB41[1]	CONT_C	持续控制
SFB42[1]	CONT_S	步骤控制
SFB43[1]	PULSEGEN	脉冲产生

[1]SFB 41 “CONT_C”，42 “CONT_S”，43 “PULSEGEN” 仅存在于 CPU 314 IFM。

IEC 函数，按数字排列

序 号	缩 写 名	功 能
FC1	AD_DT_TM	给一个时间增加一个时间值
FC2	CONCAT	组合两个 STRING 变量
FC3	D_TOD_DT	组合 DATE 和 TIME_OF_DAY 到 DT
FC4	DELETE	删除一个 STRING 变量
FC5	DI_STRNG	数据类型转换 DINT 到 STRING
FC6	DT_DATE	从 DT 中取出 DATE
FC7	DT_DAY	从 DT 中取出星期中的天
FC8	DT_TOD	从 DT 中取出 TIME_OF_DAY
FC9	EQ_DT	比较 DT（等于）
FC10	EQ_STRNG	比较 STRING（等于）
FC11	FIND	找到一个 STRING 变量
FC12	GE_DT	比较 DT（大于等于）
FC13	GE_STRNG	比较 STRING（大于等于）
FC14	GT_DT	比较 DT（大于）
FC15	GT_STRNG	比较 STRING（大于）
FC16	I_STRNG	数据类型从 INT 转换到 STRING
FC17	INSERT	插入一个 STRING 变量
FC18	LE_DT	比较 DT（小于等于）
FC19	LE_STRNG	比较 STRING（小于等于）
FC20	LEFT	STRING 变量的左部分

（续）

序　号	缩　写　名	功　　能
FC21	LEN	STRING 变量的长度
FC22	LIMIT	极限
FC23	LT _ DT	比较 DT（小于）
FC24	LT _ STRNG	比较 STRING（小于）
FC25	MAX	选择最大值
FC26	MID	STRING 变量的中间部分
FC27	MIN	选择最小值
FC28	NE _ DT	比较 DT（不等于）
FC29	NE _ STRNG	比较 STRING（不等于）
FC30	R _ STRNG	数据类型从 REAL 转换到 STRING
FC31	REPLACE	替换一个 STRING 变量
FC32	RIGHT	STRING 变量的右部分
FC33	S5TI _ TIM	数据类型从 S5TIME 转换到 TIME
FC34	SB _ DT _ DT	两个时间值相减
FC35	SB _ DT _ TM	从一个时间中减去一个时间值
FC36	SEL	二进制选择
FC37	STRNG _ DI	数据类型从 STRING 转换到 DINT
FC38	STRNG _ I	数据类型从 STRING 转换到 INT
FC39	STRNG _ R	数据类型从 STRING 转换到 REAL
FC40	TIM _ S5TI	数据类型从 TIME 转换到 S5TIME

附录 H　Allen-Bradley PLC-5 主要和次要出错位及代码

状态字 II

响应主要出错

位	显示错误	错误是
00	错误程序文件	可恢复： 出错程序能引导处理器来清除出错然后恢复扫描程序
01	梯级程序中的错误地址（见出错代码 10～19）	
02	编程错误（见出错代码 20～29）	
05	启动保护出错（见出错代码 26，位 1） 处理器置位位 5；如果你的出错程序不复位该位，处理器禁止启动	
06[1]	外围设备出错	
07	用户产生的出错；处理器跳转至出错程序（见出错代码 0～9）	
08	看门狗出错	
11[1]	MCP 不存在或不是一个梯级文件	
12[1]	PII 文件不存在或不是一个梯级文件	
13	STI 文件不包含梯形图或不存在	
03	处理器检测到 SFC 出错（见出错代码 74～79）	不可恢复： 处理器进入出错模式而不扫描出错程序
04	当编译一个梯级程序文件时，处理器检测到一个错误（见错误编码 70）	
09	系统配置错误；	
10	不可恢复的硬件错误	
14	出错程序不包括梯形图或不存在	
15	出错程序不包含梯形图	

[1] 表示仅出现在 PLC-5/11，-5/20，-5/30，-5/40，-5/40L，-5/60，-5/60L 或-5/80 处理器。

可能的主要出错：

状态字	如果状态位值为：15……8 7……0	位序号	出错是
S23	＊＊＊＊＊＊＊＊ ＊＊＊＊＊＊＊1	0	坏的用户程序文件（见出错代码 10～19）
S23	＊＊＊＊＊＊＊＊ ＊＊＊＊＊＊10	1	非法的操作地址（见出错代码 20～29）
S23	＊＊＊＊＊＊＊＊ ＊＊＊＊＊100	2	编程错误（见出错代码 30～49）
S23	＊＊＊＊＊＊＊＊ ＊＊＊＊1000	3	SFC 出错（见出错代码 71～79）
S23	＊＊＊＊＊＊＊＊ ＊＊＊10000	4	程序编译错误（见出错代码 70）
S23	＊＊＊＊＊＊＊＊ ＊＊100000	5	上电保护出错
S23	＊＊＊＊＊＊＊＊ ＊1000000	6	通道 3 设备出错[1]

（续）

状态字	如果状态位值为：15……8 7……0	位序号：	出　错　是
S23	* * * * * * * * 10000000	7	用户产生的出错（见出错代码 0～9）
S22	* * * * * * * 1 00000000	8	看门狗定时器出错
S22	* * * * * * 10 00000000	9	坏的系统配置（见出错代码 80～89）
S22	* * * * * 100 00000000	10	硬件错误
S22	* * * * 1000 00000000	11	MCP 不存在或不是梯级[1]
S22	* * * 10000 00000000	12	PII 不存在或不是梯级[1]
S22	* * 100000 00000000	13	STI 不存在或不是梯级[1]
S22	* 1000000 00000000	14	坏的出错程序
S22	10000000 00000000	15	非梯级文件

每一个 * 代表一个位，状态可以是 0 或 1

[1] 仅对增强型 PLC-5 处理器

字 1 中可能的次要出错（存储在 S：10）

如果状态位值为：15……8 7……0	位序号：	出　错　是
* * * * * * * * * * * * * * * 1	0	电池是坏的或没有电池
* * * * * * * * * * * * * * 10	1	DH＋表改变
* * * * * * * * * * * * * 100	2	STI 重叠
* * * * * * * * * * * * 1000	3	EEPROM 传送
* * * * * * * * * * * 10000	4	继续编辑先前的 SFC
* * * * * * * * * * 100000	5	无效的 I/O 状态文件
* * * * * * * * * 1000000	6	存储器电源电压低[1]
* * * * * * * * 10000000	7	没有更多的命令块存在
* * * * * * * 1 00000000	8	EEPROM 太小，刻录失败
* * * * * * 10 00000000	9	没有 MCP 被配置来运行
* * * * * 100 00000000	10	MCP 不允许
* * * * 1000 00000000	11	PII 字号不在本地框架内[1]
* * * 10000 00000000	12	用户 PII 例行程序重叠[1]
* * 100000 00000000	13	没有命令块存在来得到 PII[1]
* 1000000 00000000	14	算术溢出发生[1]
10000000 00000000	15	SFC “lingering” 功能重叠[1]

每一个 * 代表一个位，状态可以是 0 或 1

[1] 仅对增强型 PLC-5 处理器

可能的次要出错—仅限于增强型 PLC-5（存储在 S：17）

如果字 2 中的状态位值为：15……8 7……0	位序号：	出　错　是
* * * * * * * * * * * * * * * 1	0	在本地和远程 I/O 的队列满
* * * * * * * * * * * * * * 10	1	服务于通道 1A 的队列满
* * * * * * * * * * * * * 100	2	服务于通道 1B 的队列满
* * * * * * * * * * * * 1000	3	服务于通道 2A 的队列满

（续）

如果字 2 中的状态位值为：15……8 7……0	位序号：	出 错 是
* * * * * * * * * * * 10000	4	服务于通道 2B 的队列满
* * * * * * * * * * 100000	5	在串行端口无 modem
* * * * * * * * 1000000	6	* 在本地框架表中的远程 I/O 框架，或者 * 远程 I/O 框架大于映像区大小。如果本地框架被设置为八进制密度扫描并且 I/O 映像表小于 64 个字（8 个框架），出错也可以由本地框架导致
* * * * * * * * 10000000	7	错误没有定义
* * * * * * * 1 00000000	8	ASCII 指令错误
* * * * * * 10 00000000	9	复制节点地址
* * * * * 100 00000000	10	DF1 主站轮循列表错误
* * * * 1000 00000000	11	错误没有定义
* * * 10000 00000000	12	错误没有定义
* * 100000 00000000	13	错误没有定义
* 1000000 00000000	14	错误没有定义
10000000 00000000	15	错误没有定义

主要出错代码

代码	出 错
00～09	保留给用户定义的出错代码
10[1]	运行时间数据表检测失败
11[1]	坏的用户程序校验和
12	坏的整型操作数类型，重装新处理器存储器文件
13	坏的混合模式操作类型，重装新处理器存储器文件
14	指令的操作数不足，重装新处理器存储器文件
15	指令的操作数太多，重装新处理器存储器文件
16	错误指令，可能因为装载了不兼容的处理器存储器文件
17	找不到表达式结束语；重装新的处理器存储器文件
18	丢失编辑区的结束语；重装新的处理器存储器文件
19[1]	下载失败
20	向一个间接地址输入了一个太大的元素编号
21	向一个间接地址输入了一个负的元素编号
22	试图访问一个未定义程序文件
23	你使用了一个负的文件号，你使用了一个大于存在文件号的文件号，或你试着间接寻址文件 0、1 或 2
24	你试着间接寻址一个错误类型的文件
30	你试着跳到一个有太多嵌套的子程序文件
31	你没有输入足够的子程序参数
32	你跳到一个无效的文件
33[1]	你输入一个不是 68000 编码的 CAR 程序文件
34	你在一个定时器指令中输入一个负的预设值或累加值
35	你在一个 PID 指令中输入一个负的时间变量

（续）

代码	出　错
36	你在一个 PID 指令中输入一个超出范围的设定值
37	你在块传输、立即输入、立即输出指令中寻址一个无效模块
38	你输入一个来自非子程序文件的返回指令
39	丢失 NXT 的 FOR 指令
40	PID、BTR、BTW 或 MSG 指令的太小的控制文件
41	丢失 FOR 的 NXT 指令
42	你试着跳到一个被删除的标签上
43[1]	文件不是一个 SFC 文件
44～69	保留
70	处理器检测到复制的图标
71[1]	处理器试着开始一个已经运行的 SFC 子图表
72[1]	处理器试着开始一个没有运行的 SFC 子图表
73[1]	处理器试着开始更多的允许的子图表
74	检测到 SFC 文件错误
75	SFC 有太多激活的函数
77	SFC 丢失文件或有错误类型的步、动作、传输；或子图表已经创建但为空；或在 SFC 中指定的 SC 或定时器文件为空或太小
78	在电源丢失后处理器不能继续运行 SFC
79	你试着下载一个 SFC 到一个不能运行 SFC 的处理器；或指定的 PLC 不能支持这个增强型 SFC
80	你错误的在一个 1-插槽配置中安装一个 32 点的 I/O 模块（PLC-5/15，-5/25）；你有一个 I/O 配置错误（PLC-5/11，-5/20，—5/30，-5/40，-5/40L，-5/60，-5/60L，-5/80）
81	你非法地设置一个 I/O 底盘底板开关；开关 4 或 5 必须为关
82[1]	选择操作的非法的 EEPROM 卡类型
83[1]	用户看门狗出错
84[1]	在用户配置适配器模式块传输中的错误
85[1]	坏的存储卡
86[1]	存储卡和主机不兼容
87[1]	扫描框架列表重叠
88[1,2]	没有足够的时间来处理远程 I/O
90[1]	协处理器扩展内存检测失败
91[1]	协处理器无定义信息类型
92[1]	协处理器无效索引
93[1]	协处理器无效最大调查大小
94[1]	协处理器无效 ASCII 信息
95[1]	协处理器报告出错
96[1]	协处理器当前信号丢失
97[1]	协处理器无效最小调查大小
98[1]	协处理器第一/最后 16 字节 RAM 检测失败
99[1]	协处理器数据传输错误

（续）

代码	出　　错
100[1]	处理器到协处理器的传输失败
101[1]	协处理器扫描传输结束失败
102[1]	通过协处理器传输的原始数据指定的文件号是一个无效值
103[1]	通过协处理器传输的原始数据指定的元素号是一个无效值
104[1]	通过协处理器请求的传输大小是一个无效大小
105[1]	协处理器的原始传输块的偏移量是一个无效值

[1] 仅在 PLC-5/11，-5/20，-5/30，-5-40，-5/40L，-5/60，-5/60L，和 -5/80 处理器

[2] 处理器不能处理所有的数据和扫描通道过度使用远程 I/O 缓存

附录Ⅰ ALLEN-BRADLEY SLC 500 主要出错代码

地址	错误代码（十六进制）	上电错误	等级			处理器			
			非用户	用户		Fixed5/01	5/02	5/03	5/04
				不可恢复	可恢复				
S：6	0001	NVRAM 错误	×			•	•	•	•
	0002	硬件看门狗时间溢出	×			•	•	•	•
	0003	存储模块的内存错误。这种错误也能在进入 REM 运行模式的时候发生	×				•	•	•
	0005	保留			×			•	•
	0006	保留			×			•	•
	0007	存储器模块传输失败	×					•	•
	0008	内部软件错误	×					•	•
	0009	内部硬件错误	×					•	•
	0010	处理器不能满足需要的修订等级	×			•	•	•	•
	0011	可执行程序文件号 2 缺少	×			•	•	•	•
	0012	梯级程序有一个内存错误	×			•	•	•	•
	0013	需要的存储模块丢失或 S：1/10 或 S：1/11 不是满足程序需要的设置			×	•	•	•	•
	0014	内部文件错误	×			•	•	•	•
	0015	配置文件错误	×			•	•	•	•
	0016	电源丢失后的启动保护。位 S：1/9 被置位时上电，或在运行时断电，所存在的错误情况			×		•	•	•
	0017	NVRAM/存储模块用户程序不匹配		×				•	•

（续）

地址	错误代码（十六进制）	上电错误	等级			处理器			
			非用户	用户		Fixed5/01	5/02	5/03	5/04
				不可恢复	可恢复				
	0018	不兼容用户程序-操作系统类型不匹配。这种错误也可以在上电时发生	×					•	•
	0019	丢失或重复的标号被检测到		×				•	•
	001F	在线编辑时发生一个程序完整性问题	×					•	•
	0004	在运行模式中发生的存储器错误	×				•	•	•
	0020	在扫描结束时一个次要错误位被置位。参考 S：5 次要错误位			×	•	•	•	•
S：6	0021	一个扩展 I/O 底盘的远程电源失效。⚠固定式和 FRN1 至 4 SLC5/01 处理器——如果处理器处于 REM 运行模式时，发生远程电源错误，错误 0021 将导致主要出错停止位（S：1/13）在本地底盘下一次上电时被清除。SLC5/02 处理器和 FRN5. SLC5/01 处理器——本地底盘的电源不需要循环来恢复 REM 运行模式。一旦远程底盘电源恢复，CPU 将重新启动系统。	×			•	•	•	•
	0022	用户看门狗扫描时间超出		×		•	•	•	•
	0023	STI 中断文件无效或不存在		×			•	•	•
	0024	无效的 STI 中断间隔（大于 2550ms 或为负）		×			•	•	•
	0025	过多的堆栈深度/STI 例程的 JSR 调用		×			•	•	•

（续）

地址	错误代码（十六进制）	上电错误	等级			处理器			
			非用户	用户		Fixed5/01	5/02	5/03	5/04
				不可恢复	可恢复				
S：6	0026	过多的堆栈深度/I/O 中断程序的 JSR 调用		×			·	·	·
	0027	过多的堆栈深度/用户出错程序的 JSR 调用		×			·	·	·
	0028	无效或不存在的“启动保护”出错程序文件值	×				·	·	·
	0029	变址地址在全部数据文件区域外 ⚠ SLC5/02 处理器将一个表示 0 的变化值赋给紧跟着错误恢复的出错指令			×		·		
				×				·	
	002A	变址寻址引用超出了指定的引用数据文件		×			·	·	··
	002B	文件号存在，但他不是正确的文件类型或文件号不存在			×			·	·
	002C	间接引用元素不存在，但文件类型正确并存在，例如，T4：[N7：0] N7：0＝10，但 T4 仅是 T4：9			×			·	·
	002D	一个子元素被不正确引用或者一个间接引用已经应用在一个 M-文件			×			·	·
	002E	无效 DII 输入插槽			×			·	·
	002F	无效或不存在的 DII 中断文件		×				·	·

I/O 错误

错误码：下列编码中的字符 XX 代表插槽号，以十六进制表示。如果准确插槽不能被检测到，字符 XX 变为 1F。

可恢复 I/O 出错（仅包含 SLC 5/02，SLC 5/03，SLC 5/04 处理器）：许多 I/O 出错是可以恢复的。为了恢复，你必须在用户出错程序中禁止某些指定的插槽 XX。如果你不禁止插

槽 XX，处理器将在扫描结束时出错。

注意：一个严重毁坏的 I/O 卡可能导致处理器指示一个错误存在于插槽 1，即使损坏的卡不是安装在插槽 1。

十六进制的插槽序号

插槽	XX	插槽	XX	插槽	XX	插槽	XX
0	00	8	08	16	10	24	18
1	01	9	09	17	11	25	19
2	02	10	0A	18	12	26	1A
* *3	03	11	0B	19	13	27	1B
4	04	12	0C	20	14	28	1C
5	05	13	0D	21	15	29	1D
6	06	14	0E	22	16	30	1E
7	07	15	0F	23	17	*	1F

* 这个值指示插槽没有找到（SLC 5/01，SLC5/02，SLC5/03，SLC5/04 处理器）

* * 这个值指示插槽没有找到（500 紧凑型控制器）

地址	错误代码（十六进制）	用户程序指令错误	出错等级			处理器			
			非用户	用户		Fixed5/01	5/02	5/03	5/04
				不可恢复	可恢复				
S：6	0030	尝试跳到一个有太多嵌套子程序的文件。这个代码也意味着一个程序含有潜在的递归子程序		×		•	•	•	•
	0031	一个不支持的指令引用被检测到		×		•	•	•	•
	0032	一个顺序器长度/位置参数点超过了数据文件的结束处			×	•	•	•	•
	0033	LFU、LFL、FFU、FFL、BSL、BSR 指令的长度超过了数据文件的结束点			×	•	•	•	•
	0034	一个定时器累加器的负值或预设值被检测到			×	•	•	•	•
		仅带有 24VDC 输入的紧凑型处理器：一个负的或零 HSC 预设值在 HSC 指令中被检测到			×	•			

（续）

地址	错误代码（十六进制）	用户程序指令错误	出错等级			处理器			
			非用户	用户		Fixed5/01	5/02	5/03	5/04
				不可恢复	可恢复				
S：6	0035	TND、SVC、REF 指令在一个中断或用户出错程序中被调用		×			·	·	·
	0036	一个无效值被使用作为 PID 指令参数			×		·	·	·
	0038	一个 RET 指令在非子程序文件中被检测到	×			·	·	·	·
	xx3A	试图写一个位于有常数数据文件保护的文件中的间接地址			×			·	·
	1f39	无效字符串长度在一个字符串文件中被检测到			×			·	·
	xx50	一个底盘数据错误被检测到			×	·	·	·	·
	xx51	一个“固定”运行时间错误在 I/O 模块中被检测到		×		·	·	·	·
	xx52	检测到一个用户程序需要的模块丢失或移动			×	·	·	·	·
	xx53	当运行时，一个用户程序声明一个插槽不使用，并且这个插槽被检测到有一个 I/O 模块插入。这也可以表示一个 I/O 模块已经自我复位			×	·	·	·	·
		对于一个空的底盘，试图进入运行或检测模式			×			·	·
	xx54	检测一个用户程序需要的模块是错误类型			×	·	·	·	·
	xx55	一个用户程序需要的离散 I/O 模块被检测到有错误 I/O 计数。这个代码也意味着一个特殊卡的驱动不正确			×	·	·	·	·

（续）

地址	错误代码（十六进制）	用户程序指令错误	出错等级			处理器			
			非用户	用户		Fixed5/01	5/02	5/03	5/04
				不可恢复	可恢复				
S：6	xx56	检测到用户程序指定的底盘配置不正确	×			•	•	•	•
	xx57	一个特殊 I/O 模块没有在要求的时间极限中响应一个锁共享内存命令			×	•	•	•	•
	xx58	一个特殊 I/O 模块产生一个普通出错。模块的状态字节的卡出错位置位		×		•	•	•	•
	xx59	一个特殊 I/O 模块没有在规定时间里响应一个表示完成的命令			×	•	•	•	•
	xx5A	硬件中断问题			×		•	•	•
	xx5B	G 文件配置错误—用户程序 G 文件大小超过了模块容量			×		•	•	•
	xx5C	M0-M1 文件配置错误：用户程序文件 M0-M1 的文件大小超过了模块的容量			×		•	•	•
	xx5D	处理器不支持中断服务请求			×		•	•	•
	xx5E	处理器 I/O 驱动（软件）错误			×		•	•	•
	xx60-xx6F	识别一个 I/O 模块特殊的可恢复主要错误。参考特殊模块提供的用户手册			×		•	•	•
	xx70-xx7F	识别出一个 I/O 模块特殊的不可恢复主要错误。参考特殊模块提供的用户手册		×			•	•	•
	xx90	被禁止的插槽的中断问题		×			•	•	•
	xx91	一个被禁止的插槽出错		×			•	•	•
	xx92	一个无效的或不存在的模块中断子程序（ISR）文件		×			•	•	•

（续）

地址	错误代码（十六进制）	用户程序指令错误	出错等级			处理器			
			非用户	用户		Fixed5/01	5/02	5/03	5/04
				不可恢复	可恢复				
	xx93	不支持 I/O 模块特殊的主要错误		×			•	•	•
S：6	xx94	在 REM 运行或 REM 检测模式中，检测到在带电情况下一个模块被插入。这也意味着一个 I/O 模块自己复位了		×			•	•	•

附录 J　ALLEN-BRADLEY PLC-5 PID 控制块

参　数	地址助记符	描　　述
设定值	.SP	在 PID 配置屏幕的工程单位中输入一个浮点数。有效范围时$-3.4E^{+38}$到$+3.4E^{+38}$
过程变量	.PV	将来自模拟输入模块的数据标度变换为与你选择的设定值相同的工程单位，并显示
错误	.ERR	显示下列错误值的一种： 反作用：错误＝SP－PV
		正作用：错误＝PV－SP
输出%	.OUT	显示 PID 算法控制输出值（0～100%）
模式	.MO .MO＝0 .MO＝1 .SWM＝1	显示操作模式： AUTO－自动 PID 控制 MANUAL－来自一个手动控制站点的控制 SW MANUAL－来自数据监控器或梯级程序的模拟手动控制
PV 报警	.PVHA＝1 .PVLA＝1	在 PID 配置屏幕上显示是否超出 PV 值的报警极限。显示下列中的一种： NONE－PV 值在报警极限范围内 HIGH－PV 超过报警极限值的上限（与死区联用） LOW－PV 超过报警极限值的下限（与死区联用）
偏差报警	.DVPA＝1 .DVNA＝1	在 PID 配置屏幕上显示错误是否在报警极限内。显示下列中的一种： NONE－错误在偏差报警极限内 HIGH－错误超过报警上限（与死区联用） LOW－错误超过报警下限（与死区联用）
输出极限	.OLH＝1 .OLL＝1	显示指令是否将输出限制在你在屏幕上选择的上下极限值（.MAXO 和 .MINO）。显示下列中的一种： NONE-输出没有受限制 HIGH-输出钳在上限端（.MAXO） LOW-输出钳在下限端（.MINO） PID 算法有一种抗积分饱和特性可以阻止当输出达到高或低报警限值时，积分项变得太大。达到限值时，算法将停止计算积分项，直到输出回到范围内
SP 超出范围	.SPOR＝0 .SPOR＝1	显示设定值是否在你通过 PID 配置屏幕选择的工程单位之外。显示下列中的一种： NO－SP 在范围之内 YES－SP 在范围之外 重要：当指令第一次被使能时，如果 SP 超出范围，一个主要处理器出错发生
DB 内的错误	.EWD＝0 .EWD＝1	显示错误是否在你输入到屏幕上的死区值内或是超过该值。显示为下列情况之一： RESET－错误存在于死区区间 SET－错误穿过死区中线

（续）

参　数	地址助记符	描　述
PID初始化	.INI=0 .INI=1	每次你在控制块中改变一个值，PID指令花费超过第一次扫描时间两倍的时间去执行（直至初始化）。显示为下列情况之一： NO—在你改变控制块值之后，PID指令没有初始化 YES—因为你没有改变任何控制块值，PID指令保持初始化 注意：不要在运行时改变输入的范围或工程单位。如果你必须做这些，你必须复位这个位来重新初始化。否则，指令将会失效，可能会损坏设备或使人员受伤
A/M站点模式	.MO=0 .MO=1	输入0（自动）或1（手动）。显示下列情况之一： AUTO(0)—自动PID控制 MANUAL(1)—手动PID控制 手动控制指定一个来自手动控制站点的输出覆盖PID算法的计算值 重要：选择手动会覆盖设置输出模式
软件A/M模式	.SWM=0 .SWM=1	为了软件仿真控制，输入0（自动PID控制）或1（设置输出模式）。显示下列情况之一： AUTO（0）—自动PID控制 SW MANUAL（1）—软件仿真PID控制 单回路时，你可以用数据监控器仿真一个手动控制站点。为此，设置.SWM为SW MANUAL，并且输入一个设置输出百分比值 多回路时，你可以用梯形图仿真控制站点。设置.SWM为SW MANUAL，并移一个值到设置输出元素.SO
状态使能	.EN=0 .EN=1	使用（1）或禁止（0）使用来显示梯级条件的位。利用该位你可以知道PID指令是否执行。显示下列情况之一： 0—指令不执行 1—指令执行
比例增益	.KP	输入一个浮点值。独立或标准增益的有效范围是 0到$3.4E^{+38}$
积分增益	.KI	输入一个浮点值。独立增益的有效范围是0到$3.4E^{+38}$/s；标准增益的有效范围为每次重复0到$3.4E^{+38}$分钟
微分增益	.KD	输入一个浮点值。独立增益的有效范围是0到$3.4E^{+38}$s；标准增益的有效范围为每次重复0到$3.4E^{+38}$分钟
死区	.DB	在PID配置显示屏上的工程节点输入一个浮点值。有效范围是0到$3.4E^{+38}$，看.EWD
输出偏离%	.BIAS	输入一个值（−100到+100）来表示你想要的前馈输出比例或用来作为输出的偏移。偏离值能补偿来自系统能量的稳定损失 梯级程序也能输入一个前馈值来破坏扰动预测的输出。此值也常用来控制一个带有转换日志的过程