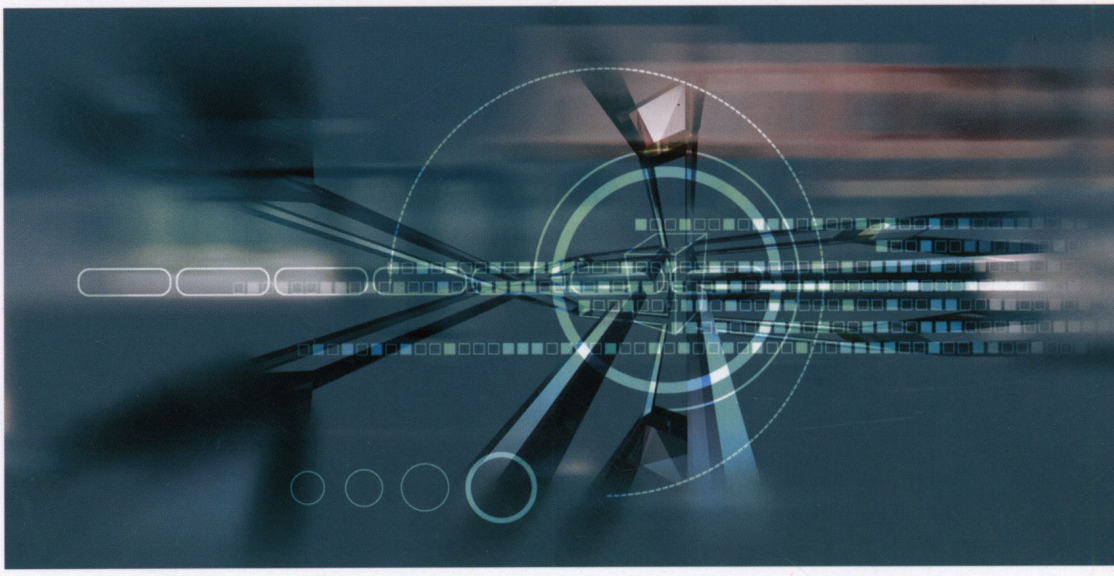
PLC实用技术教程

从基础到实践
——PLC与组态王



韩晓新 主编
邢绍邦
刘海燕 副主编

机械工业出版社
CHINA MACHINE PRESS

PLC 实用技术教程

从基础到实践——PLC与组态王

韩晓新　主　编
邢绍邦
刘海燕　副主编

机 械 工 业 出 版 社

本书从工程实践的角度出发，强调基础、重视应用。本书简单介绍了可编程序控制器（PLC）的来源、现状及发展，并以当前国内外最具代表性的 PLC——三菱 FX 系列为例，介绍了其类型、硬件构成、软件编程方法、应用开发，常用组态软件的选择、使用、编程方法及基于 PLC 与组态王等组态软件的系统开发等知识，通过实训项目对组态王软件的具体应用给出全程指导，介绍了 PLC 控制系统的设计及编程方法，并附有习题和实例参考。

本书可作为普通高等教育电气工程与自动化（应用型本科）规划教材，对于机电行业的广大技术人员、技术工人也是一本更新知识结构及新技术应用的入门教材。

图书在版编目（CIP）数据

从基础到实践：PLC 与组态王/韩晓新主编．—北京：机械工业出版社，2011.6

（PLC 实用技术教程）

ISBN 978-7-111-34776-7

Ⅰ．①从…　Ⅱ．①韩…　Ⅲ．①可编程序控制器　Ⅳ．①TM571.6

中国版本图书馆 CIP 数据核字（2011）第 093651 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）

策划编辑：王　欢　　责任编辑：赵玲丽

版式设计：张世琴　　责任校对：肖　琳

封面设计：赵颖喆　　责任印制：乔　宇

三河市国英印务有限公司印刷

2016 年 11 月第 1 版第 1 次印刷184mm×260mm·18.25 印张·449 千字

5401—6600 册

标准书号：ISBN 978-7-111-34776-7

定价：49.80 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务　　网络服务

社服务中心：（010）88361066　　门户网：http://www.cmpbook.com

销售一部：（010）68326294　　教材网：http://www.cmpedu.com

销售二部：（010）88379649

读者购书热线：（010）88379203　　**封面无防伪标均为盗版**

前　言

众所周知，可编程序控制器（Programmable Logic Controller，PLC）的应用是自动化控制领域的核心技术之一，自产生以来，经过40多年的不断发展，特别是随着半导体芯片技术、计算机技术和数字化控制理论的发展，PLC技术日益成熟，功能日益强大，可靠性不断提高，已越来越受到人们的重视。由于PLC系统构成简单、配置灵活、编程方便、可靠性高，能适用于不同类型的控制对象和各种复杂环境，已经在自动化控制领域获得了非常广泛的应用。因此，学习和掌握PLC应用技术已成为工程技术人员和电气信息类专业学生的基本需求和紧迫任务。鉴于上述原因，目前大专院校机械、电气、机电一体化专业及其他有关专业都普遍开设了PLC这门课程。但PLC的产品和其他电子产品一样，具有种类较多、发展较快的特点，因此常常造成教材的内容和企业、学校教学专用设备脱节，内容滞后于产品技术发展的现状。有感于此，我们查阅了大量国内外的PLC资料，选择了非常有代表性的日本三菱FX系列PLC，介绍其原理、应用及编程方法。FX系列PLC有多种模块、功能齐全、性能价格比极高，是大专院校购置实验装置较佳的选择。同时，掌握一种型号的PLC之后，再选择使用其他型号的PLC时也可触类旁通，相互参照。

与此同时，自20世纪80年代以来，随着自动化技术进步和自动化程度不断提高，自动化系统和设备之间的通信需求也在不断增长，特别是随着计算机网络技术的迅速发展和生产信息化程度的提高，使工控组态软件应运而生。工业控制系统组态技术广义上亦应属于数字仿真范畴，组态王（KINGVIEW）以其丰富的指令集合与友好的视图窗口使控制系统设计者的工作质量和工作效率发生了革命性的飞跃。对于从事系统设计、试验分析、运行维护工作的工程技术人员和电气信息类专业学生来说，掌握该项技术的实现工具——工控组态软件，是非常必要的。

本书主要介绍了三菱FX系列PLC的类型、硬件构成、软件编程方法、应用开发和常用组态软件的选择、使用、编程方法、基于PLC与组态王等组态软件的系统开发等知识。内容上与PLC应用实际紧密联系，突出实践性，侧重典型性，淡化纯理论性，强调综合应用性的特点。在以往的教学中，综合应用性往往容易被忽视。因为在工业自动化控制领域，一个实际使用的控制系统往往是很复杂的，本书把机械、电工、电子、液压、气动和计算机等知识与PLC知识有机地联系综合起来，增设的一些实践教学项目可充分培养学生分析问题和解决问题的能力，培养学生的创新和综合应用的职业能力，真正实现学生毕业后在PLC技术与组态软件技术应用领域“零距离上岗”的最终教学目标。

本书由江苏技术师范学院韩晓新副教授任主编并负责编写第1、4、9章，副主编邢绍邦讲师负责编写第5、6章，副主编刘海燕讲师负责编写第2、3章，于冬梅讲师负责编写第7章，吴晓庆讲师负责编写第8章，朱品伟讲师负责本书所有插图、表的设计与编排，全书由韩晓新负责统稿并审阅全稿。此外，江苏技术师范学院教务处为本书的编写提供了立项资助，编者在此表示衷心感谢。

由于编者水平有限，书中难免有不少遗漏和错误之处，敬请广大读者批评指正。

编　者

目　　录

第 1 章　可编程序控制器概述

可编程序控制器是近几年迅速发展并得到广泛应用的新一代工业自动化控制装置。它的产生和发展与继电器有很大的关系。继电器是一种弱电信号控制强电信号的电磁开关，在复杂的继电器控制系统中，故障的查找和排除是非常困难的，如果工艺要求发生变化，就得重新设计电路连线安装，严重影响生产。显然，将计算机功能的完备、灵活、通用与继电器控制系统的简单易懂、操作方便、价格便宜等优点结合起来，制造一种新型的工业控制装置成为电气控制系统的发展方向。

1969 年，由美国数字设备公司（DEC）研制出第一台可编程序控制器并在 GM 公司汽车生产线上首次应用成功，实现了生产的自动化控制。此后，日本、德国等相继引入可编程序控制器，使之迅速发展起来。这一时期它主要用于顺序控制，虽然也采用了计算机的设计思想，但当时只能进行逻辑运算，故称为可编程逻辑控制器，简称 PLC（Programmable Logic Controller）。

20 世纪 70 年代后期，随着微电子技术和计算机技术的迅速发展，可编程序控制器在原有逻辑运算的基础上，更多地具有了计算机的功能，不仅用逻辑编程取代硬连线逻辑，还增加了运算、数据传送和处理等功能，真正成为一种电子计算机工业控制装置，而且做到了小型化和超小型化。这种采用微电脑技术的工业控制装置被简称为 PC（Programmable Controller），但由于 PC 容易和个人计算机（Personal Computer）混淆，所以人们仍习惯沿用 PLC 作为可编程序控制器的缩写。

PLC 的可靠性高、编程简单、通用性好并且易于安装维护。它的几种常见的控制类型为逻辑控制、生产监控、模拟量控制、闭环调节控制，组成大型控制网络。正是由于它的这些强大的控制功能，使其被广泛应用于冶金、化工、机械、印刷、电子、电力、建筑建材、交通等几乎所有的工业控制领域。

1.1　分类及特点

PLC 的种类很多，其实现的功能、内存容量、控制规模、外形等方面均存在较大的差异。因此，PLC 的分类并没有一个严格的统一标准，而是按结构型式、控制规模、实现的功能大致进行分类。

1. 按结构型式分类

PLC 按照硬件的结构型式可以分整体式和组合式。整体式 PLC 外观是一个长方形箱体，又称为箱式 PLC。组合式 PLC 在硬件构成上具有较高的灵活性，其模块可以像拼积木那样进行组合，构成具有不同控制规模和功能的 PLC，因此这种 PLC 又称为积木式 PLC。

（1）整体式 PLC　整体式 PLC 的 CPU、存储器、I/O 安装在同于机体内（如三菱 FX 系列）。其结构的特点是结构简单、体积小、价格低。

（2）组合式 PLC　组合式 PLC 为总线结构。其总线做成总线板，上面有若干个总线槽，

每个总线槽上可安装一个 PLC 模块，不同的模块实现不同的功能。PLC 的 CPU、存储器和电源等做成一个模块，该模块在总线板上的位置一般来说是固定的，而且该模块也是构成组合式 PLC 所需的。其他模块可根据 PLC 的控制规模、实现的功能进行选取，安装在总线的其他任一总线槽上。组合式 PLC 的总线板又称为基板。组合式 PLC 的特点是系统构成较灵活，可构成具有不同控制规模和功能的 PLC，价格较高。

2. 按控制规模分类

I/O 点数是衡量 PLC 控制规模的重要参数。因此，按控制规模可分为小型 PLC、中型 PLC 和大型 PLC。

（1）小型 PLC　I/O 点数在 256 点以下。

（2）中型 PLC　I/O 点数在 256 ~ 2048 点之间。

（3）大型 PLC　I/O 点数在 2048 点以上。

3. 按实现的功能分类

按照 PLC 所能实现的功能不同，可以把 PLC 大致分为低档 PLC、中档 PLC 和高档 PLC 三类。

（1）低档 PLC　具有逻辑运算、计时、计数、移位、自诊断、监控等基本功能，还具有一定的算术运算、数据传送和比较、通信、远程和模拟量处理功能。

（2）中档 PLC　除了具有低档 PLC 的功能外，还具有较强的算术运算、数据传送和比较、数据转换、远程、通信、子程序、中断处理和回路控制功能。

（3）高档 PLC　除了具有中档 PLC 的功能外，还具有带符号的算术运算、矩阵运算、函数、表格、CRT 显示、打印机打印等功能。

4. 按生产地域分类

目前，世界上 PLC 产品按地域可分成三大类：美国、欧洲和日本产品。美国和欧洲的 PLC 技术是在相互隔离的情况下独立研究开发的，产品有明显的差异性；日本的 PLC 技术是从美国引进的，对美国的 PLC 产品有一定的继承性。但是，日本的主推产品定位在小型 PLC 上，而欧美以大中型 PLC 为主。

我国从 1974 年也开始研制 PLC，1977 年研制成功了以 MC14500 微处理器为核心的可编程序控制器，并开始工业应用。令人遗憾的是，国产 PLC 占有市场份额很小。

（1）美国 PLC 产品　美国是 PLC 生产大国，有 100 多家 PLC 生产厂商，其中著名的有 A-B 公司、通用电气（GE）公司、莫迪康（MODICON）公司、德州仪器（TI）公司、西屋公司等。其中 A-B 公司是美国最大的 PLC 制造商，其产品约占美国 PLC 产品市场的一半。

（2）欧洲 PLC 产品　德国的西门子（SIEMENS）公司、AEG 公司、法国的 TE 公司是欧洲著名的 PLC 制造商。德国西门子公司的电子产品以性能精良而久负盛名，在中、大型 PLC 产品领域与美国的 A-B 公司齐名。

（3）日本 PLC 产品　日本的小型 PLC 最具特色，某些需要用欧美的中型机或大型机才能实现的控制，日本的小型机就可以解决。在开发复杂的控制系统方面明显优于欧美的小型机，所以十分受用户欢迎。日本有许多 PLC 制造商，如三菱、欧姆龙、松下、富士、日立、东芝等公司，在世界小型 PLC 市场上，日本产品约占 70% 的份额。

1.2 可编程序控制器的组成和工作原理

PLC 的种类很多，但其组成的一般原理基本相同，其实质是一个新型的工业控制计算机。本书以三菱 FX 系列 PLC 为例，以后不再说明。

1.2.1 可编程序控制器的硬件组成

PLC 主要由单板机、输入/输出接口、电源、扩展接口和编程器接口等几部分组成，其结构框图如图 1-1 所示。

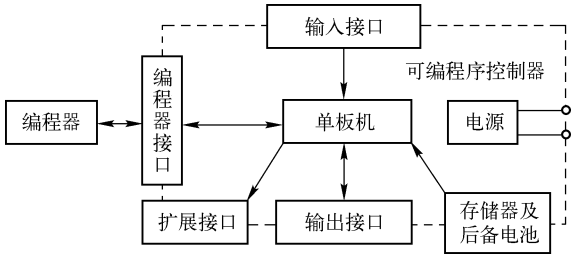


图 1-1 可编程序控制器的硬件结构框图

1. 单板机

可编程序控制器的单板机即为 CPU 板。它包括一台基本计算机必需的部件：中央处理器 CPU，存储器 RAM、ROM，并行接口 PIO，串行接口 SIO，时钟 CTC。单板机的作用是对整个可编程序控制器的工作进行控制。它的工作分为两部分：一部分是对系统进行管理，如自诊断、查错、信息传送时钟、计数刷新等；另一部分就是根据用户程序执行输入输出操作、程序解释执行操作等。

单板机中的 CPU 主要采用通用微处理器、单片机和双极行位片式微处理器三种类型。常用的通用微处理器有 8 位微处理器，如 Z80A、8085、M6800、M6809 等；16 位微处理器，如 8086、M68000 等。常用的单片机有 8039、8031、M6801 等。常用的双极行位片式微处理器有 AMD2900、AMD2903 等。可编程序控制器的档次越高，CPU 的位数也越多，功能指令也就越强。FX_{2N}系列 PLC 使用的微处理器是 16 位的 8096 单片机。

单板机中的存储器主要用于存储系统监控程序及系统数据，并且用于生成用户环境。其容量的大小取决于系统的工作能力及系统程序的质量。

串行接口和并行接口是用于 CPU 与接口器件交换信息的，它的数量取决于系统规模的大小。

单板机中的定时器/计数器是用于产生系统时钟及用户时钟信息的。这里我们要注意，在一台单板机中，CTC 的数量是很有限的，但经过系统监控程序的处理，可以产生几十个甚至数百个相对独立的计数器和定时器。

可编程序控制器的用户程序及参数的存储器有三种类型。一种是 RAM，通常都是 CMOS 型的，耗电极微。在可编程序控制器中通常都用锂电池作后备，这样在失电时不会损坏程序，在调试时就很方便了。为了防止偶然操作失误而损坏程序，还可采用 EPROM 或 EEP—ROM，在程序调试完成后就可以固化。但 EPROM 的缺点是写入时必须用专用的写入器，擦

除时要用专用的擦除器，这对于用户是很不方便的。因此，最近的发展是采用电可擦除只读存储器（EEPROM），它的写入和擦除只需编程器即可，而不再需要其他装置。

2. I/O

I/O 接口是 PLC 与现场 I/O 设备或其他外部设备之间的连接部件。PLC 通过输入接口把外部设备的状态或信息读入 CPU，用户程序运算与操作后，把结果经输出接口传送给执行机构。输入接口对输入信号进行滤波、隔离、电平转换等，把输入信号的逻辑安全可靠地输入到 PLC 的内部。输出接口是把程序执行的结果输出到 PLC 的外部，输出接口具有隔离 PLC 内部电路和外部执行元件的作用，还具有功率放大的作用。

（1）输入接口电路　各种 PLC 的输入接口电路大都相同，通常有两种形式。一种是直流输入，其输入器件可以是无源触点或传感器的集电极开路晶体管。它又进一步分为源型（共 + 端）和漏型（共 – 端）。另一种是交流输入，这实际上是将交流信号经整流、限流后再通过光耦传入 CPU。

图 1–2 所示是 FX 系列 PLC 的输入电路（包括 RUN 输入）的例子。此例为直流源型，它包括如下几部分的内容：

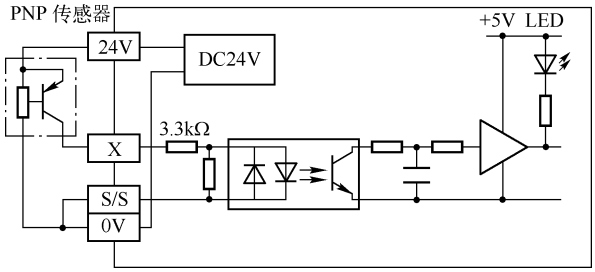


图 1–2　FX 系列 PLC 输入电路

1）输入端子：当电流通过输入端子时，输入信号接通。公共端为“ + ”的是源型；公共端为“ – ”的是漏型。对于源型机，将 S/S 端与 0 V 相连；对于漏型机，将 S/S 端与 24 V 连接。输入信号为 ON 时输入指示灯亮。所有输入的公共端是 S/S 端子（而不是接地端）。

2）输入电路：输入电路的一次电路与二次电路用光耦合器隔离。二次电路中设有 RC 滤波器，这是为防止由于输入触点的颤振、输入线混入的噪声引起误动作而设计的。因此，外部输入从 ON→OFF 或 OFF→ON 变化时，PLC 内部约有 10 ms 的响应滞后。

3）输入灵敏度：PLC 的输入电流为 7 mA（DC24 V）。引起输入动作的最小电流为 2. 5 ~ 3 mA，但为了确实启动，必须取 4. 5 mA 以上；为了确实切断，必须取 1. 5 mA 以下。

4）传感器用外部电路：PLC 的输入电流是由 PLC 内部的 DC 24 V 电源供给的，因此，光电开关等传感器用外部电源驱动时，该外部电源须为 DC 24 V ± 4 V，传感器的输出晶体管须为 PNP 集电极开路型（对于源型）或 NPN 集电极开路型（对于漏型）。

（2）输出接口电路　输出通常有三种形式：一种是继电器输出型，CPU 接通继电器的线圈，继而吸合触点，而触点与外线路构成回路；另一种是晶体管输出，它是通过光耦合器使开关晶体管通断以控制外电路；再一种就是晶闸管输出型，这里的晶闸管是采用光触发型的。

1）公共点：输出端子有两种接法：一种输出是各自独立的（无公共点）；另一种为每

4～8 个输出点构成一组，共用一个公共点。在输出共用一个公共端子范围内，必须用同一电压类型和同一电压等级，但不同的公共点组可使用不同电压类型和等级（如 AC 200 V、AC 100 V、DC 24 V 等）的负载。各输出公共点之间是相互隔离的。

2）电路隔离：继电器输出型是利用输出继电器的触点和线圈，将 PLC 的内部电路与外部负载电路进行电气隔离；SSR 输出型是在 PLC 的内部电路与输出元件（三端双向晶闸管开关器件）之间用光敏晶闸管进行隔离；晶体管输出型是在 PLC 的内部电路与输出晶体管之间用光耦合器进行隔离。

3）响应时间：继电器输出型响应时间最长，从输出继电器的线圈通电或断电到输出触点 ON 或 OFF 的响应时间均为 10 ms。SSR 型和晶体管型的响应时间都很短，SSR 型从光敏晶闸管驱动（或断开）到输出三端双向晶闸管开关器件 ON（或 OFF）的时间为 1 ms 以内，晶体管型从光耦合器动作（或关断）到晶体管 ON（或 OFF）的时间为 0. 2 ms 以内（在 24 V、20 mA 时）。

4）输出电流：FX 系列 PLC 输出接口电路如图 1-3 所示。

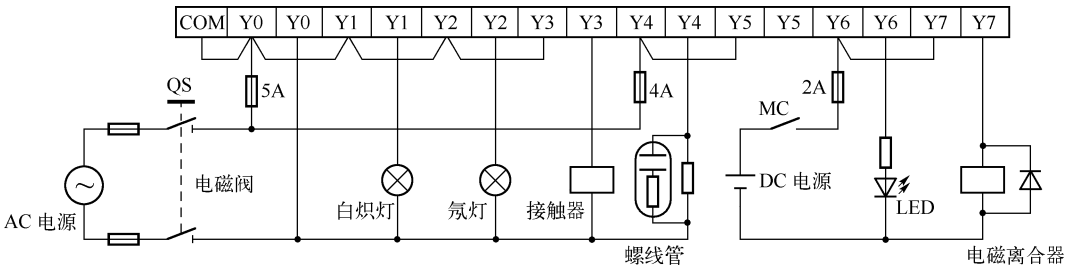


图 1-3　FX 系列 PLC 输出接口电路

继电器型在 AC 250 V 以下电路电压时可驱动负载为纯电阻负载 2A/1 点；感性负载 80 V · A 以下（AC 100 V 或 AC 200 V）；灯负载 100W 以下（AC 100 V 或 AC 200 V）。

感性负载时，容量越大，触点寿命就越短。另外，根据输出触点的不同，直流感性负载启闭时，要将该负载与并联分流二极管连接，最大电压要在 DC 30 V 以下。

SSR 型每点的输出电流最大为 0. 3A，但是，为了防止温度上升，每 4 点的总电流必须在 0. 8A 以下（1 点平均 0. 2A）。浪涌电流大的负载，开关次数频繁时，电流有效值要取 0. 2A 以下。

晶体管型每个输出点可有 0. 5A 电流。但是，因有温度上升的限制，每 4 点输出总电流不得大于 0. 8A（1 点平均 0. 2A）。因为输出晶体管的通态电压降约为 1. 5 V，所以驱动半导体器件等时，要考虑这 1. 5 V 的压降。

5）开路漏电流：继电器型输出触点 OFF 时没有漏电流。

SSR 型输出端子处接有关断用 RC 吸收电路，故有开路漏电流（1 mA/AC 100 V，2 mA/AC 200 V），可能引起微小电流负载的误动作，所以负载应在 0. 4 V · A/AC 100 V 或 1. 6 V · A/AC 200 V 以上。若小于该指标或灯负载时，还应并接浪涌电流吸收电路。

SSR 输出型与继电器输出相比，有较大的开路漏电流，它能使 MY—1、LY—2 等小型继电器保持动作，使用时应加以注意。晶体管输出型的开路漏电流在 100 μA 以下。

3. 电源

PLC 的供电电源是 220 V 交流电，也有用 DC 24 V 供电的。PLC 对电源稳定度要求不高，

内部有开关式稳压电源，电源的交流输入端一般由尖峰脉冲吸收电路，以提高抗干扰能力。允许电源电压在额定值 -15% ~10% 的范围内波动。现在大部分 PLC 电源部分还有 DC 24 V 输出，用于对外部传感器等供电，但电流一般都不超过 100 mA。

4. 扩展接口

扩展接口是用于扩展 I/O 单元的，它使可编程序控制器的点数规模配置更为灵活。这种扩展接口实际上为总线形式，或已配接开关量的 I/O 单元，也可配置如模拟量、高速脉冲等单元以及通信适配器等。在大型机中，扩展接口为插槽扩展基板的形式。

5. 编程器接口

可编程序控制器本体上通常是不带编程器的。为了能对可编程序控制器编程及监控，可编程序控制器上专门设置有编程器接口，通过这个接口可以接各种形式的编程装置，还可以利用此接口做一些监控工作。

6. 存储器接口

为了存储用户程序以及扩展用户程序存储区、数据参数存储区，可编程序控制器上还设有存储器扩展口，可以根据使用的需要扩展存储器。其内部也是接到总线上的。

7. 编程器

上面所述都是可编程序控制器本体上的电路。对于正常使用来说，通常不需编程器，因此，编程器设计为独立的部件。编程器的层次很多，性能、价格都相差很悬殊。

最简单的编程器至少包括一个键盘、一些数码字符显示器。这里的键盘不是单板机上的那种键盘，而是直接表示可编程序控制器指令系统的键盘，因而使用很方便；其显示部分包括三部分，即序号、指令码和元件号（在讲指令系统时详述）。它具有输入编辑、检索程序的功能，同时还具有系统监控的功能；有些还设有存储转接插口，用于将可编程序控制器中的程序转储到诸如盒带、软盘等存储介质中去。

这种编程器的缺点就是无法以梯形图图形的方式输入并编辑程序和监控运行。因此，层次稍高的编程器上就设置了一小块液晶显示器，用于图形编辑、监控。这种编程器对于习惯于使用梯形图的人员来说，无疑方便了许多。

对于较复杂的控制电路图，用这种小型的图形编程器显示就非常不方便。因此，再上一个层次就是通用的大型液晶显示屏编程器。它的功能就相当于一台独立的专用计算机，为了便于现场使用，将它设计得非常轻巧。它采用更换程序存储器的方法更换系统程序。这种编程器可脱机独立编辑程序，与可编程序控制器联机后可将程序写入/读出可编程序控制器，并且可以监控运行，还具有打印机接口、录音机接口等外设接口。

为了进一步完善功能，近来发展了不少功能极强的专用图形编程器。这种编程器就像一台便携式计算机，本身带有 CRT。软盘驱动器，还有许多接口（如打印机接口、串行接口等），程序编辑功能也极强。它还可以作为工作站使用，即把它挂在可编程序控制器网络上，对各站进行监控、管理、调试等工作。

1.2.2 可编程序控制器的软件系统和编程语言

PLC 是一种工业控制计算机，不仅有硬件，软件也是必不可少的。在 PLC 中软件分为两大部分，即系统程序和用户程序。

1. 系统程序

系统程序是 PLC 赖以工作的基础，采用汇编语言编写，在 PLC 出厂时就已固化于 ROM

型系统程序存储器中，不需用户干预。系统程序分为系统监控程序和解释程序。系统监控程序用于监视并控制 PLC 的工作，如诊断 PLC 系统工作是否正常，对 PLC 各模块的工作进行控制，与外设交换信息。根据用户的设定使 PLC 处在编制用户程序的状态或者处在运行用户程序状态等，解释程序用于把用户程序解释成微处理器能执行的程序。当 PLC 处在运行方式时，系统监控程序启动解释程序，解释程序将用户利用梯形图或语句表编制的用户程序编译成处理器可以执行的指令组成的程序，处理器执行这些处理后的程序来完成用户的控制任务。与此同时，系统监控程序对这一过程进行监控并控制，如发现异常立即进行报警并做出相应的处理。

2. 用户程序

用户程序又称为应用程序，是用户为完成某一特定任务而利用 PLC 的编程语言而编制的程序。用户程序通过编程器输入到 PLC 的用户存储器中，通过 PLC 的运行而完成这一特定的任务。

3. 编程语言

各种型号的 PLC 都有其自己的编程语言，以三菱 FX 系列 PLC 为例，主要有以下几种。

（1）指令表编程语言　指令表编程语言是以“LD”、“AND”、“OUT”等基本指令或功能指令助记符输入的方法。

例如：

步	指令	软器件号	
0	LD	X000	
1	OR	Y005	
2	ANI	X002	
3	OUT	Y005	
4	LD	X001	
5	MOV	K5	DO
10	END		

这种以列表形式表示的程序，控制内容难以看懂，但如果使用简易编程器（手编）输入程序时，必须以这种方法输入。

（2）梯形图编程语言　梯形图编程语言是类似于继电器控制电路的一种编程语言，它面向控制过程，直观易懂，是 PLC 编程语言中用得最多的一种语言。

（3）顺序功能图编程语言　顺序功能图（SFC）编程语言是应用步进指令和状态器进行编程的方法，一般是在顺序控制时应用。在应用步进指令编程时，一般先根据控制要求画出 SFC 流程图（详见本书第 3 章），再根据流程图转化成梯形图。

以上这三种程序表达方式可以互相转换。

1.2.3　可编程序控制器的工作方式

PLC 采用循环扫描的工作方式，整个工作过程可以分为输入处理、程序处理、输出处理三个阶段。工作过程如图 1-4 所示。

1. 输入处理

程序执行前，可编程序控制器的全部输入端子的通/断状态读入输入映像寄存器。在程序执行中，即使输入状态变化，输入映像寄存器的内容也不变，直到下一扫描周期的输入处

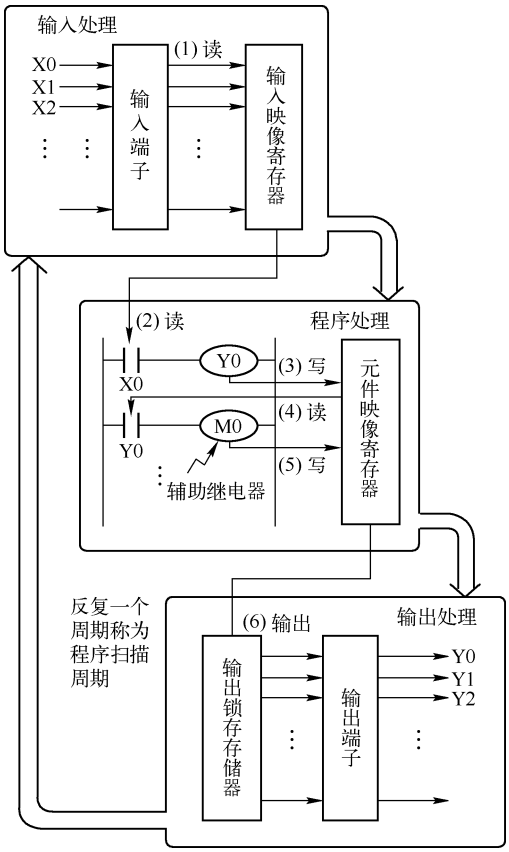


图 1-4　PLC 的工作过程

理阶段才读入这变化。另外，输入触点从通（ON）→断（OFF）［或从断（OFF）→通（ON）］变化到处于确定状态为止，输入滤波器还有一段响应延迟时间（约 10 ms）。

2. 程序处理

对应用户程序存储器所存的指令，从输入映像寄存器和其他软元件的映像寄存器中将有关软元件的通/断状态读出，从 0 步开始顺序运算，每次结果都写入有关的映像寄存器，因此，各软元件（X 除外）的映像寄存器的内容随着程序的执行在不断变化。

输出继电器的内部触点的动作由输出映像寄存器的内容决定。

3. 输出处理

全部指令执行完毕，将输出 Y 的映像寄存器的通/断状态向输出锁存寄存器传送，成为可编程序控制器的实际输出。

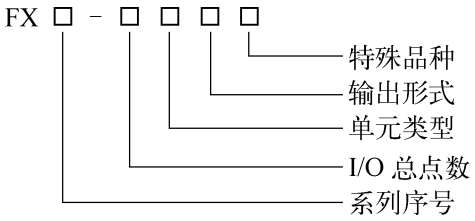
可编程序控制器内的外部输出触点对输出软元件的动作有一个响应时间，即要有一个延迟才动作。

以上方式称为成批输入/输出方式（或刷新方式）。

1.3　三菱公司产品概述

1.3.1　FX 系列 PLC 型号命名方式

FX 系列 PLC 型号命名的基本格式为



系列序号：1S、1N、2N、2NC，即 FX_{1S}、FX_{1N}、FX_{2N}、FX_{2NC}。

I/O 总点数：14～256 点。

单元类型：M 代表基本类型；E 代表 I/O 混合扩展单元及扩展模块；EX 代表输入专用扩展模块；EY 代表输出专用扩展模块。

输出形式：R 代表继电器输出；T 代表晶体管输出；S 代表晶闸管输出。

特殊品种：D 代表 DC 电源，DC 输入；A1 代表 AC 电源，AC 输入；H 代表大电流输出扩展模块（1A/点）；V 代表立式端子排的扩展模块；C 代表接插口 I/ O 方式；F 代表输入滤波器 1 ms 的扩展模块；L 代表 TTL 输入型扩展模块；S 代表独立端子（无公共端）扩展模块。

若特殊品种一项无符号，说明通常指 AC 电源，DC 输入，模式端子排，继电器输出 2A/点，晶体管输出 0.5A/点，晶闸管输出 0.3A/点。

1.3.2 FX 系列 PLC 概述

FX 系列 PLC 包括 FX_{1S}、FX_{1N}、FX_{2N}、FX_{2NC}等型号。各型号的 PLC 在性能上都有所区别，了解各型号 PLC 的特点和性能是选择 PLC 的基本前提。

1. FX_{1S}型 PLC

FX_{1S}型 PLC 是一种卡片大小的 PLC，适合在小环境中进行控制。

（1）特点

1）控制点数为 10～30 点（主单元：10 点、14 点、20 点、30 点）。

2）结构紧凑，性价比高。显示模块和扩展板使系统升级变得很容易。

3）高速运算。基本指令是 0.55～0.7 μs/指令；应用指令是 3.7 μs/指令至几百微秒/指令。

4）具有可靠的、大规模的寄存器规格。EEPROM 寄存器达 2000 步。

5）丰富的器件资源。辅助继电器 512 点、定时器 64 点、计数器 32 点、数据寄存器 256 点。

（2）功能

1）具有实时时钟功能。使用标准型号的实时时钟满足对时间的应用要求。

2）联网能力。串行扩展板通过 RS-232、RS-422 或 RS-485 使联网更容易。

3）电源使用范围很宽。世界上任何地方的标准电压都适合，也可以使用直流电源。

4）基于 Windows 软件。使用 GX-Developer 或 FX-PCS-Win-C 软件能快速、容易地开发程序。

5）脉冲输出和定位功能。一个 PLC 单元中每相能同时输出 2 点 100 kHz 脉冲。PLC 配备有 7 条特殊的定位指令，包括零返回、绝对位置读出、绝对或相对地址表达方式以及特殊脉冲输出控制等。

6）模拟电位器。使用模拟电位器 VR_1、VR_2能容易地改变模拟量或定时器的值。

7）密码保护。使用一个 8 位数密码保护程序。

8）远程维护。通过调制解调器通信，可以监测、上传或下载程序和数据到远处的编程软件上。

2. FX_{1N}型 PLC

FX_{1N}型 PLC 是一种普遍选择方案，最多可达 128 点控制，并且能增加特殊功能模块或扩展板。通信和数据链接功能选项使得 FX_{1N}型 PLC 在体积、通信和特殊功能模块和能源控制等重要的应用方面非常完美。

（1）特点

1）控制点数为 14 ~ 128 点（主单元：14 点 A40 点、60 点）。

2）结构紧凑，性价比高。显示模块和扩展板使系统升级变得很容易。

3）具有可靠的、大规模的寄存器规格。EEPROM 寄存器达 8000 步。便于电池安装，维护也很方便。

4）丰富的器件资源。辅助继电器 1 536 点、定时器 256 点、计数器 235 点、数据寄存器 8 000 点。

（2）功能

1）实时时钟功能。使用标准型号实时时钟功能，满足对时钟的应用要求。

2）网络能力。完全补充的网络模块，使数据通信更容易。

3）电源使用范围很宽。世界上任何地方的标准电压都适合，可以使用直流电源。

4）增加了过程控制。在系统要求精确控制时使用 PID 指令。

5）特殊功能模块。增加了大量的特殊功能模块满足单个需要。

6）基于 Windows 软件。使用 GX-Developer 或 FX-PCS/Win-C 软件能快速、容易地开发程序。

7）模拟电位器。使用模拟电位器 VR_1、VR_2能容易地改变模拟量或定时器的值。

8）脉冲输出和定位功能。一个 PLC 单元中每相能同时输出 2 点 100 kHz 脉冲。PLC 配备有 7 条特殊的定位指令，包括零返回、绝对零、绝对或相对驱动以及特殊脉冲输出控制等。

9）密码保护。使用一个 8 位数密码保护程序。

10）远程维护。通过调制解调器通信可以监测、上传或下载程序和数据到远处的编程软件上。

3. FX_{2N}型 PLC

FX_{2N}型 PLC 是 FX 系列中最高级的模块，它拥有无以匹敌的速度、高级的功能、逻辑组件以及定位控制等特点。

（1）特点

1）控制点数为 16 ~ 256 点（主单元：16 点、32 点、48 点、65 点、80 点、128 点）。

2）灵活的配置。除了具有满足特殊要求的大量特殊功能模块外，6 个基本 FX_{2N}单元中的每一个单元可扩充到 256 点的 I/O。

3）高速运算。基本指令 0. 08 μs/指令；应用指令 1. 52 μs/指令至几百微秒/指令。

4）突出的寄存器容量。FX_{2N}型 PLC 包括 8 000 步内置 RAM 寄存器，用一个寄存器盒可扩充到 1 6000 步 RAM 或 EEPROM。

5）丰富的器件资源。辅助继电器3 072点、计时器256点、计数器235点、数据寄存器8 000点。

（2）功能

1）实时时钟功能。使用标准型号实时时钟功能可以满足对时间的应用要求。

2）增加了过程控制。使用FX_{2N}的PID指令或FX_{2N}-2LC温控模块可以实现过程控制。

3）特殊功能模块。增加了大量的特殊功能模块满足单个需要，FX_{2N}主单元可配置总计达8个特殊功能模块，为工厂自动化应用提供最大的灵活性和控制能力。

4）网络能力。可以连接到世界上最流行的开放式网络（CC-Link、Profibus-DP和DeviceNET）或者采用传感器层次的网络（如AS-l或I/O Link），解决通信的需要，串行通信选项包括RS-232、RS-422或RS-485。完全充足的网络模块，使数据通信容易实现。

5）很强的数学指令集。使用32位处理、浮点数、方根和三角几何指令满足数学功能。

6）基于Windows软件。使用GX-Developer或FX-PCS/Win-C软件能快速、容易地开发程序。

7）定位控制功能。通过定位模块（FX_{2N}-10GM、FX_{2N}-20GM）或脉冲输出模块（FX_{2N}-1PG）最多可达16轴的控制。

8）密码保护。使用一个8位数密码保护程序。

9）远程维护。通过调制解调器通信可以监测、上传或下载程序和数据到远处的编程软件上。

4. FX_{2NC}型PLC

FX_{2NC}型PLC，是在保留了强大功能特色的前提下实现了卡片式大小的PLC。I/O接口降低了接线成本并节省了空间。

（1）特点

1）控制点数为16～256点（主单元：16点、32点、64点、96点）。

2）体积小、灵活的配置。体积小，可以安装到比标准PLC小很多的位置上。基本单元可扩充到256点的I/O。可以使用高达4个特殊功能模块来扩展FX_{2NC}型PLC的能力。

3）高速运算。基本指令0.08 μs/指令，应用指令1.52 μs/指令至几百微秒/指令。

4）突出的寄存器容量。FX_{2NC}型PLC包括8 000步内置RAM寄存器，用一个寄存器盒可扩充到16KB步。

5）丰富的器件资源。辅助继电器3 072点、计时器256点、计数器235点、数据寄存器8 000点。

（2）功能

1）实时时钟功能。使用标准型号实时时钟功能可以满足对时间的应用要求。

2）定位/模拟量控制。可以连接多达4个定位/模拟量等特殊功能模块，利用它内置的功能能控制两轴（包括插补）或通过增加扩展单元能控制多轴。

3）网络能力。通过连接扩展板或特殊适配器能实现多种通信和数据链接。利用CC-Link、Profibus-DP和DeviceNET可以连接到世界上最流行的开放式网络，串行通信选项包括RS-232C、RS-485。数据链接包括N:N连接、并行连接、计算机连接及I/O连接。

4）基于Windows软件。使用GX-Developer或FX-PCS/Win-C软件能快速、容易地开发程序。

5）密码保护。使用一个 8 位数密码保护程序。

6）远程维护。通过调制解调器通信可以监测、上传或下载程序和数据到远处的编程软件上。

7）很强的数学指令集。使用 32 位处理、浮点数、方根和三角几何指令满足数学功能。

8）定位控制功能。通过定位模块（FX_{2N}-10GM、FX_{2N}-20GM）或脉冲输出模块（FX_{2N}-1PG）最多可达 16 轴的控制。

1.4 FX 系列 PLC 的编程器件

1.4.1 输入/输出继电器（X/Y）

输入/输出继电器（X/Y）电路如图 1-5 所示。

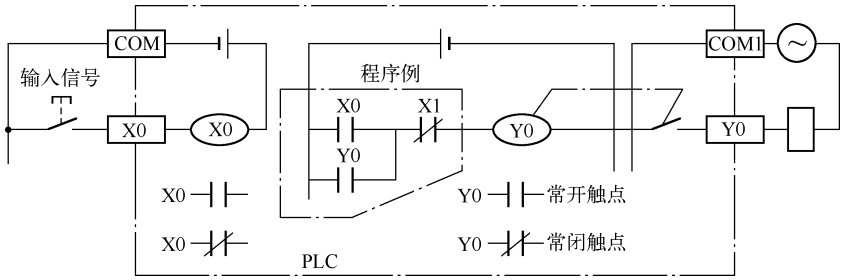


图 1-5 输入/输出继电器（X/Y）电路

1. 输入继电器（X0～X7，X10～X17，X20～X27，…）

PLC 的输入端子是从外部开关接收信号的窗口。与输入端子连接的输入继电器（X）是光隔离的电子继电器，其常开触点（a 触点）和常闭触点（b 触点）的使用次数不限。这些触点在 PLC 内可以自由使用。

FX_{2N}的输入继电器最多可达 256 点，且不能用程序驱动。

2. 输出继电器（Y0～Y7，Y10～Y17，Y20～Y27，…）

PLC 的输出端子是向外部负载输出信号的窗口。输出继电器的外部输出触点（继电器触点、双向晶闸管 SSR、晶体管等输出器件）接到 PLC 的输出端子上。输出继电器的电子常开触点和常闭触点使用次数不限，在 PLC 中可自由使用，但外部输出触点（输出元件）与内部触点的动作有所不同。

FX_{2N}的输出继电器最多可达 256 点。

扩展单元和扩展模块的输入/输出器件号是接续着其紧靠的单元开始采用八进制编号。

输入、输出虽然各有 256 点，但总点数不可超过 256 点。

1.4.2 辅助继电器（M）

PLC 内有很多辅助继电器。辅助继电器的线圈与输出继电器一样，由 PLC 内各软元件的触点驱动。辅助继电器的电子常开触点和常闭触点使用次数不限，在 PLC 内可以自由使用。但是，这些触点不能直接驱动外部负载，外部负载的驱动必须由输出继电器实行。

在逻辑运算中经常需要一些中间继电器作为辅助运算用。这些元件不直接对外输入、输出，经常用作状态暂存、移位运算等。它的数量常比 X、Y 多。另外，在辅助继电器中还有

一类特殊辅助继电器，它有各种特殊的功能，如定时时钟、进/借位标志、启动/停止、单步运行、通信状态、出错标志等。这类元件数量的多少，在某种程度上反映了可编程序控制器功能的强弱，能对编程提供许多方便。

1. 通用辅助继电器 M0～M499（500 点）

在逻辑运算中经常需要一些中间继电器作为辅助运算用，用作状态暂存、中间过渡等。通用辅助继电器的特点是线圈通电，触点动作；线圈断电，触电恢复；当系统断电时，所有状态复位。

2. 停电保持辅助继电器 M500～M1023（524 点）

停电保持辅助继电器可用参数设置方法改为非停电保持用。

PLC 在运行中若发生停电，输出继电器和通用辅助继电器全部成为断开状态。再运行时，除去 PLC 运行时就接通（ON）的以外，其他仍断开。但是，根据不同的控制对象，有的需要保存停电前的状态，并在再运行时再现该状态的情形。停电保持用辅助继电器（又名保持继电器）就是用于这种目的的。停电保持由 PLC 内装的后备电池支持。

3. 停电保持专用辅助继电器 M1024～M3071（2 048 点）

停电保持专用辅助继电器的停电保持特性不可改变。

4. 特殊辅助继电器 M8000～M8255（256 点）

这些特殊辅助继电器各自具有特定的功能，可以分成以下两大类：

1）只能利用其触点的特殊辅助继电器（只读），线圈由 PLC 系统驱动，用户只可以利用其触点。例如

M8000　运行（RUN）监控（PLC 运行时接通）

M8002　初始脉冲（仅在运行开始瞬间接通）

M8012　100 ms 时钟脉冲

2）可驱动线圈型特殊辅助继电器（读/写），用户驱动线圈后，PLC 做特定动作。例如

M8030　使 BATT LED（锂电池欠电压指示灯）熄灭

M8033　PLC 停止时输出保持

M8034　禁止全部输出

M8039　定时扫描

注：未定义的特殊辅助继电器不可在用户程序中使用。

1.4.3　状态元件（S）

状态元件（S）在步进顺控程序的编程中是重要的软元件。它与后述的步进顺控指令 STL 组合使用。有以下 4 种类型：

1）初始状态 S0～S9（10 点）；

2）回零 S10～S19（10 点）；

3）通用 S20～S499（480 点）；

4）保持 S500～S899（400 点）；

注：通用与保持型 S 元件数的分配可以通过参数设置方式加以改变。

各状态元件的常开触点和常闭触点在 PLC 内可以自由使用，使用次数不限。不用步进顺控指令时，状态元件（S）可作为辅助继电器（M）在程序中使用。

1.4.4 报警器

一部分的状态元件可用作外部故障诊断输出。作报警器用的状态元件为 S900 ~ S999（100 点）。S900 ~ S999 中任意一个置 1（ON），特殊继电器 M8048 就动作，接通故障显示输出 Y10。由外部故障诊断程序接通的状态元件可按 X5 键复位（置 0）。每次按下 X5 健，已动作的状态按元件号由小到大依次复位。

1.4.5 指针（P/I）

1. 分支指令用指针 P0 ~ P127（128 点）

分支指令用指针用来指定条件跳转、子程序调用等分支指令的跳转目标。

2. 中断用指针

FX 系列 PLC 的中断事件的类型由外部输入中断、定时器中断和高速计数器中断。

FX 系列有 6 点与 X0 ~ X5 对应的外部输入中断，中断指针为

I □ O □ 输入中断（6 点）

0：下降沿中断

1：上升沿中断

输入号（0~5）

每个输入只能用一次

例如，I001 为输入 X0 从 OFF→ON 变化时，执行由该指针作为标号后面的中断程序，并根据 IRET 指令返回。

FX_{2N} 型有 3 点定时器中断，中断指针为

I □ □ □ 定时器中断（3 点）

10~99 ms

定时器中断号（6~8）

每个定时器只能用一次

例如，I610 即为每隔 10 ms 就执行标号 I610 后面的中断程序，并根据 IRET 指令返回。

FX_{2N} 型有 6 点高速计数器中断，中断指针为

I 0 □ 0 高速计算器中断（6 点）

1~6

例如，用了 I100 就不能用 I101，用了 I610 就不能用 I620。

注意事项如下：

1）中断指针必须编在 FEND 指令后面作为标号。

2）中断点数不能多于 15 点。

3）中断嵌套级不多于 2 级。

4）中断指针中百位数上的数字不可重复使用。

5）用于中断的输入端子，就再也不能用于 SPD 指令或其他高速处理。

1.4.6 定时器（T）

定时器在可编程序控制器中的作用相当于一个时间继电器，它有一个设定值寄存器（字）、一个当前值寄存器（字）以及无数个触点（bit）。对于每一个定时器，这三个量使用同一名称，但使用场合不一样，其所指也不一样。通常在一个可编程序控制器中有几十至数百个定时器，可用于定时操作。

1. 普通型定时器（T0 ~ T245）

如图 1-6 所示，定时器线圈 T200 的驱动输入 X0 接通时，T100 的当前值计数器对 10 ms 的时钟脉冲进行累积计数，当该值与设定值 K123 相等时，定时器的输出触点就接通，即输

出触点是在驱动线圈后的 1.23 s 时动作。

当驱动输入 X0 断开或发生停电时，计数器就复位，输出触点也复位。

注：若在子程序和中断程序中使用 T192 ~ T199，则在执行 END 指令时计时值变更。当到达设定值后，在执行线圈指令或 END 指令时，输出触点接通。

2. 积算型定时器（T246 ~ T255）

如图 1-7 所示，定时器线圈 T250 的驱动输入 X1 接通时，T250 的当前值计数器开始累积 100 ms 的时钟脉冲的个数，当该值与设定值 K345 相等时，定时器的输出触点接通。

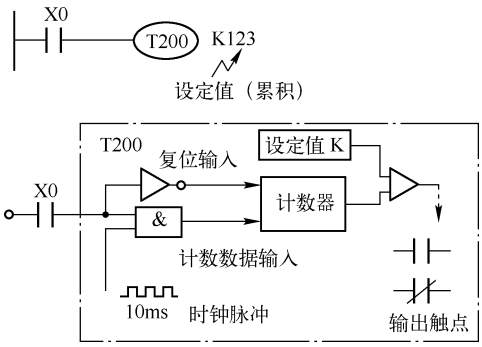


图 1-6　普通型定时器

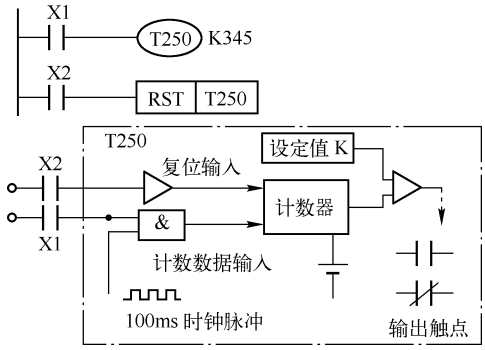


图 1-7　积算型定时器

计数中途即使输入 X1 断开或发生停电，当前值可保持。输入 X1 再接通或复电时，计数继续进行，其累积时间为 34.5 s 时触点动作。

当复位输入 X2 接通，计数器就复位，输出触点也复位。

注：若在子程序或中断程序中使用中断定时类型的 1 ms 定时器，在到达设定值后，执行该定时器第一个线圈指令时，输出触点就接通。

3. 触点的动作时序及精度

定时器在其线圈被驱动后开始计时，到达设定值后，在执行第一个线圈指令时输出触点动作如图 1-8 所示。

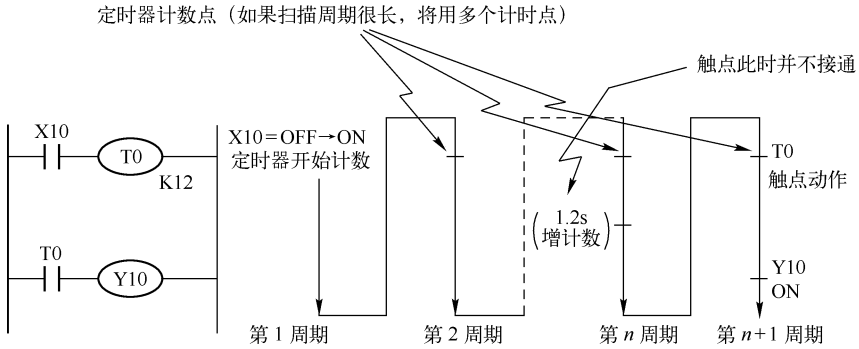


图 1-8　触点的动作时序

因此，从驱动定时器线圈到其触点动作，计时触点的动作精度大致可用下式表示：

$T^{+T_0}_{-\infty}$　　T：定时器设定时间（s）

∞：1 ms，10 ms，100 ms 定时器对应 0.001 s，0.01 s，0.1 s

T_0：扫描周期（s）

如果编程时定时器触点指令写在线圈指令之前，在最坏的情况下，定时器线圈触点动作误差为 $+2T_0$。

但当定时器的设定值为 0 时，在下一扫描周期执行线圈指令时，输出触点就动作。

另外，1 ms 定时器在执行线圈指令后，以中断方式对 1 ms 时钟脉冲计数。

定时器的详细动作如图 1-9 所示。

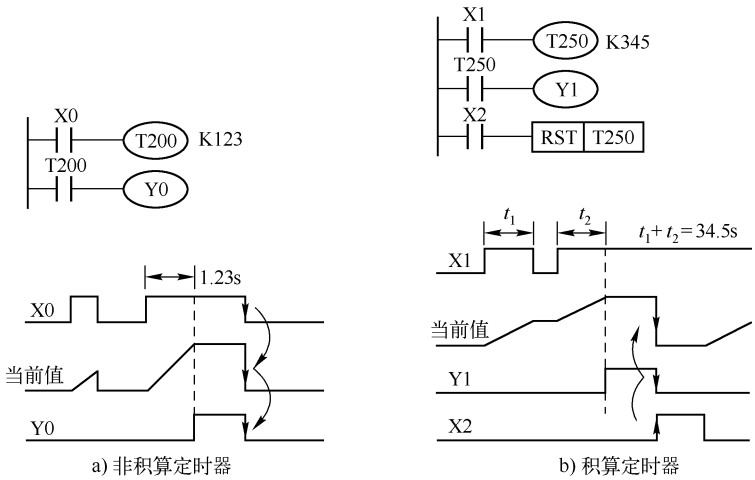


图 1-9　定时器的详细动作

1.4.7　计数器（C）（字，bit）

1. 内部信号计数器

内部信号计数器是在执行扫描操作时对内部元件（如 X、Y、M、S、T 和 C）的信号进行计数的计数器。因此，其接通（ON）时间和断开（OFF）时间应比 PLC 的扫描周期稍长，通常其输入信号频率大约为几个扫描周期/秒。

（1）16 bit 增计数器（设定值：1 ~ 32 767）　有两种类型的 16 bit 二进制增计数器：

1）通用：C0 ~ C99（10 点）；

2）停电保持用：C100 ~ C199（100 点）。

其设定值在 K1 ~ K32767 之间。设定值 K0 与 K1 含义相同，即在第一次计数时，其输出触点动作。

使用计数器 C100 ~ C199 时，即使停电，当前值和输出触点的置位/复位状态也能保持。

通用/停电保持型计数器数目分配可以通过参数设置加以改变。

如图 1-10 所示，X11 为计数输入，每次 X11 接通时，计数器当前值增 1。当计数器的当前值为 10 时，即计数输入达到第 10 次时，计数器 C0 的输出触点接通，之后即使输入 X11 再接通，计数器的当前值都保持不变。当复位输入 X10 接通（ON），执行 RST 指令，计数器当前值复位为 0，输出触点也断开（OFF）。

计数器的设定值，除了可由常数 K 设定外，还可间接通过指定数据寄存器的元件号来设定，如指定 D10 而 D10 的内容为 123，则与设定 K123 等效。

如果将大于设定值的数置入当前值寄存器（例如用 MOV 指令），则当计数输入端 ON 时，计数器继续计数。其他计数器也是如此。

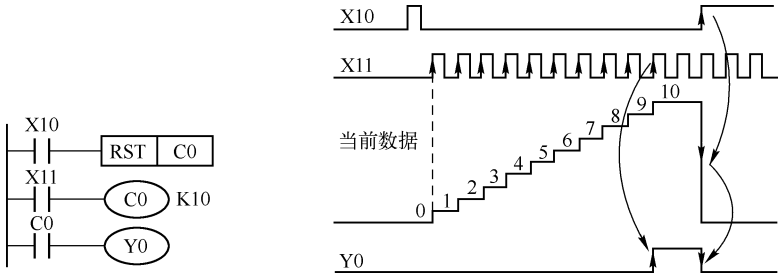


图 1-10　16 bit 增计数器

（2）32 bit 双向计数器（设定值：－2 147 483 648 ~ ＋2 147 483 647）　有两种 32 bit 的增/减计数器：通用计数器 C200 ~ C219（20 点），保持计数器 C220 ~ C234（15 点）。其设定值为－2 147 483 648 ~ ＋2 147 483 647，计数的方向（增计数或减计数）由特殊辅助继电器 M8200 ~ M8234 设定。

如图 1-11 所示，用 X14 作为计数输入，驱动 C200 线圈进行加计数或减计数。

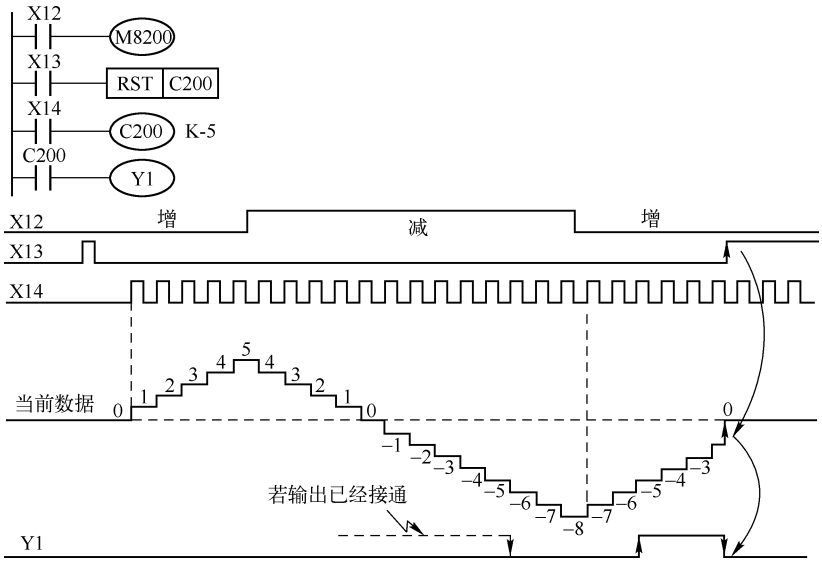


图 1-11　32 bit 双向计数器

当计数器的当前值由 －6→－5（增加）时，其触点接通（置 1）；由 －5→－6（减少）时，其触点断开（置 0）。

当前值的增减虽与输出触点的动作无关，但从＋2147483647 起再进行加计数当前值就成为－2 147 483 648。同样从－2 147 483 648 起进行减计数，当前值就成了＋2 147 483 647（这种动作称为循环计数）。当复位输入 X13 接通（ON），计数器的当前值就为 0，输出触点也复位。

使用停电保持的计数器，其当前值和输出触点状态均能停电保持。

32 bit 计数器可当作 32 bit 数据寄存器使用，但不能用作 16 bit 指令中的操作元件。

2. 高速计数器

虽然 C235 ~ C255（共 21 点）都是高速计数器，但它们共享同一个 PLC 上的 6 个高速

计数器输入端（X0～X5）。即如果输入已被某个计数器占用，它就不能再用于另一个高速计数器（或其他用途）。也就是说，由于只有 6 个高速计数的输入，因此，最多同时用 6 个高速计数器。另外，还可用作比较和直接输出等高速应用功能。

高速计数器的选择并不是任意的，它取决于所需计数器的类型及高速输入的端子。计数器类型如下：

1）1 相无启动/复位端子：C235～C240；

2）1 相带启动/复位端子：C241～C245；

3）2 相双向：C246～C250；

4）2 相 A—B 相型：C251～C255。

上列所有的计数器均为 32 bit 增/减计数器。各种计数器对应输入端子的名称见表 1-1。

表 1-1　高速计数器（X0，X2，X3：最高 10 kHz；X1，X4，X5：最高 7 kHz）

输入	1 相						1 相带启动/复位					2 相双向					2 相 A—B 相型				
	C235	C236	C237	C238	C239	C240	C241	C242	C243	C244	C245	C246	C247	C248	C249	C250	C251	C252	C253	C254	C255
X0	U/D						U/D			U/D		U	U		U		A	A		A	
X1		U/D					R			R		D	D		D		B	B		B	
X2			U/D					U/D			U/D		R		R			R		R	
X3				U/D				R			R			U		U			A		A
X4					U/D				U/D					D		D			B		B
X5						U/D			R					R		R			R		R
X6										S					S					S	
X7											S					S					S

注：U——增计数输入；D——减计数输入；A——A 相输入；B——B 相输入；R——复位输入；S——启动输入。

X6 和 X7 也是高速不同输入，但只能用作启动信号而不能用于高速计数。不同类型的计数器可同时使用，但它们的输入不能共用。

注：输入端 X0～X7 不能同时用于多个计数器，例如，如果使用了 C251，下列计数器和指令就不能使用：C235，C236，C241，C244，C246，C247，C249，C252，C254，I0**，I1** 及 SPD（FNC 56）指令的有关输入。

高速计数器是按中断原则运行的，因而它独立于扫描周期，选定计数器的线圈应以连续方式驱动，以表示这个计数器及其有关输入连续有效，其他高速处理不能再用其输入端子。

[例 1-1]　高速计数器应用如图 1-12 所示，当 X20 接通时，选中高速计数器 C235，根据表 1-1，C235 对应计数输入 X0，因此，计数输入脉冲应从 X0 而不是 X20 输入。当 X20 断开，线圈 C235 断开；同时，C236 接通，因此，选中计数器 C236，其计数输入为 X1 端。

警告：不要用计数输入端作计数器线圈的驱动触点，如图 1-13 所示。

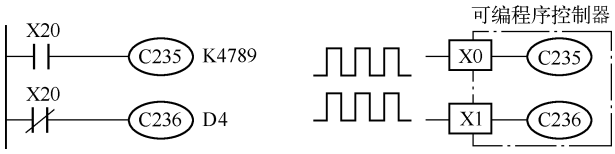


图 1-12　高速计数器应用示例

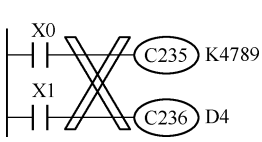


图 1-13　高速计数器错误应用举例

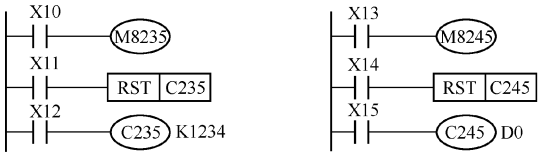
（1）1 相计数器（C235 ~ C245）　1 相高速计数器有如下两组：

C235 ~ C240	无启动/复位端	设定值范围
C241 ~ C245	有启动/复位端	−2 147 483 648 ~ +2 147 483 647

上列两组计数器的计数方式及触点动作与前文中讲述的普通 32 bit 计数器相同。做增计数器时，当计数值达到设定值时，触点动作并保持；作减计数时，到达计数值则复位。

1 相计数器的计数方向取决于其对应标志 MS△△△，△△△为对应计数器号（235 ~245）。

图 1-14a 所示为 1 相无启动/复位（C235 ~ C240）高速计数器。每个计数器只用一个计数输入端。其动作如下：



a) 无启动 / 复位 (C235 ~ C240)　　b) 带启动 / 复位 (C241 ~ C245)

图 1-14　1 相（C235 ~ C245）高速计数器

1）方向标志 M8235 为 ON 时，计数器 C235 减计数；为 OFF 时，计数器 C235 增计数。

2）X11 接通，C235 复位至地触点 C235 断开。

3）当 X12 接通，C235 选中，从表 1-1 中得知，对应计数器 C235 的输入为 X0，C235 对 X0 输入的 OFF→ON 信号计数。

图 1-14b 所示为 1 相带启动/复位（C241 ~ C245）高速计数器。这些计数器各有一个计数输入及一个复位输入。计数器 C244 和 C245 还另有一个启动输入。其动作如下：

1）当方向标志 M8245 为 ON 时，C245 减计数；M8245 为 OFF 时，C245 增计数。

2）当 X14 接通，C245 像普通 32 bit 计数器一样复位。从表 1-1 可知，C245 还能由外部输入 X3 复位。

3）计数器 C245 还有外部启动输入端对。当 X7 接通，C245 开始计数；对断开时，C245 停止计数。

4）X15 选通 C245，对 R 输入端的“OFF→ON”计数。

注意：对 C245 设置 D0 实际上是设置 D0、DI，因为计数器为 32 bit；外部控制启动（X7）和复位（X3）是立即响应的，它不受程序扫描周期的影响。

（2）2 相双向计数器（C246 ~ C250）　这种计数器具有一个输入端用于增计数，另一个输入端用于减计数。某些计数器还具有复位和启动输入。

以图 1-15a 所示计数器 C246 为例，其动作如下：

1）当 X10 接通，C246 以普通 32 bit 增/减计数器一样的方式复位。

2）从表 1-1 可知。C246 计数器用 X0 作为增计数端，X1 作为减计数端，X11 必须接通以选通 C246，以使 X0、X1 输入有效：

X0 “OFF→ON”，C246 增 1

X1 “OFF→ON”，C246 减 1

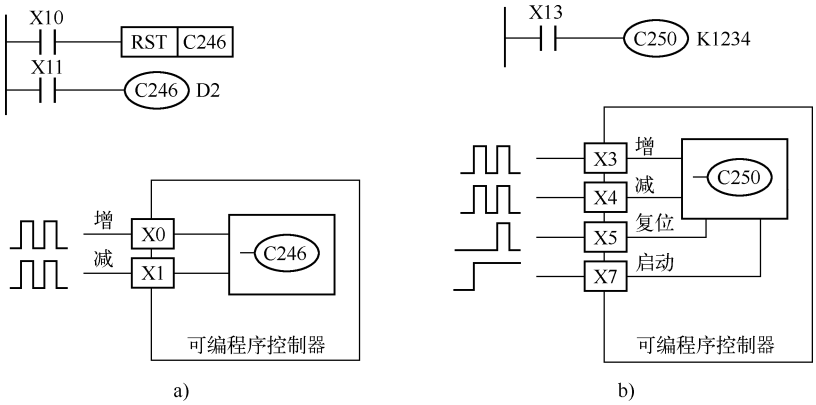


图 1-15　2 相双向计数器

又如图 1-15b 所示计数器 C250：

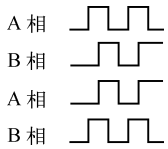
1）由表 1-1 得知，双向计数器 C250 将 X5 作为复位输入，X7 作为启动输入，因此，可由外部复位，而不必使用 RST C250 指令。

2）要选通 C250 必须接通 X13，启动输入 X7 必须接通，以开始脉冲计数，X7 为 OFF 时，停止计数。

3）增计数输入：X3；减计数输入：X4。

4）计数方向可由监视相应的状态寄存器 M8△△△得到，△△△为计数器号。ON：减计数；OFF：增计数。

（3）2 相 A-B 相计数器（设定值：－2 147 483 648 ~ ＋2 147 483 647）　2 相 2 输入（C251 ~ C255，1 个或 2 个，电池后备）最多可有 2 个 2 相 32 bit 二进制增减计数器，其对于计数数据的动作过程与前文中描述的 32 bit 计数器相同。对这些计数器，只有表 1-1 中所示的输入端可用于计数。它是采用中断方式计数，与扫描周期无关。这些计数器还有一些独立于逻辑操作的执行比较和输出操作的应用指令。选定计数器元件号后，对应的启动、复位及输入信号就能使用。A 相和 B 相信号决定了计数器是增计数还是减计数。当 A 相波形为 ON 状态时：B 相输入 OFF—ON：增计数；B 相输入 ON ~ OFF：减计数。



如图 1-16 所示，检查对应的特殊继电器 M8△△△可知道计数器是增计数还是减计数：

1）在 X11 接通时，C251 对输入 X0（A 相）、X1（B 相）的“ON/OFF”事件计数。

2）选通信号 X13 接通时，一旦 X7（S 启动输入）接通，C255 立即开始计数，计数输入为 X3（A 相）和 X4（B 相）。

3）X5（R 复位输入）接通时，C255 复位。在程序中编入虚线所示指令，则 X12 接通时也能使 C255 复位。

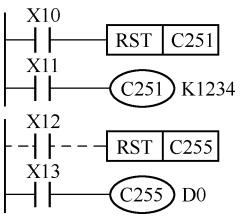


图 1-16　2 相 A-B 相计数器应用实例

最高计数频率：计数器的最高计数频率受两个因素约束：各个输入的响应速度；全部高

速计数器的处理时间。

1）各输入端的响应速度：它由硬件所限制，只用一个计数器时各输入点的最高响应频率为：X0、X2、X3：最高 10 kHz；X1、X4、X5：最高 7 kHz。

2）全部高速计数器的处理时间：这是高速计数器的主要速度限制。计数器操作是采用中断方式，因此，计数器用得越少，则可计数频率就越高。但如果某些计数器用比较低的频率计数，则其他计数器可以较高的频率计数。

使用的全部计数器的频率总和应低于 20 kHz。频率总和是指同时在 PLC 上出现所有信号的最大频率的总和。为使高速计数器准确计数，这个频率总和必须小于 20 kHz。

注意，FX_{2N}的 X0、X1 是特殊的硬件计数器，当 C235、C236、C246 作为单相高速计数器时，最高频率可达 60 kHz；C251 作为两相计数器时，最高频率可达 30 kHz。使用高速比较指令采用 X0、X1 作为输入点时也有频率限制：FNC53、FNC54 最高可达 11 kHz；FNC55 最高可达 5.5 kHz。

3. 相计数器

（1）双向型　设计成增计数信号和减计数信号不会同时发生。实际上，在特定时刻只使用 1 相信号。因此，它们可按单相计数器计算方法来计算频率总和。

当增、减计数信号脉冲同时到达计数器，则作为 2 相信号计数器来计算频率总和。

当使用具有顺时针和逆时针输出形式的编码器时，这些双向计数器可用比 A—B 相型计数器高得多的频率计数，而不会影响结果。

（2）A—B 相型　是另一种类型的计数器，它可同时将 A 相与 B 相信号解码，自动确定增计数或减计数。

在使用 1 个或 2 个这种计数器后，建议不要用高于 2 kHz 的频率对其计数。

当计算频率总和时，每一个计数器的最大信号频率应乘以 4 后再与其他计数器的频率相加。

虽然求得频率总和（19 kHz）低于 20 kHz，但双向计数器（C246）的输入 X1 的硬件响应频率限制为 7 kHz。因此，C246 的信号频率需从 8 kHz 降为 7 kHz。

当使用多个计数器或多种类型的计数器时，其频率总和必须低于 20 kHz（记住将 A—B 相计数器频率乘以 4）。

［**例 1-2**］　1 相计数器：10 kHz（1 个），2 kHz（1 个）；

A—B 相计数器：2 kHz（1 个）；

频率总和 = 10 kHz + 2 kHz + 2 kHzX4 = 20 kHz。

1.4.8　数据寄存器（D）（字）

可编程序控制器用于模拟量控制、位置量控制、数据 I/O 时，需要许多数据寄存器存储参数及工作数据。这类寄存器数量随机型的不同而不同，较简单的只能进行逻辑控制的机器没有此类寄存器，而高档机中可达数千个。

每一个数据寄存器都是 16 bit（最高位为符号位）；可以用两个数据寄存器合并起来存放 32 bit 数据（最高位为符号位），称为数据寄存器对，如图 1-17 所示。

（1）用数据寄存器 D0 ~ D199（200 点）（可通过参数设置改为保持型）　只要不写入其他数据，已写入的数据不会变化，但是，PLC 状态由运行（RUN）到停止（STOP）时，全部数据均清零。

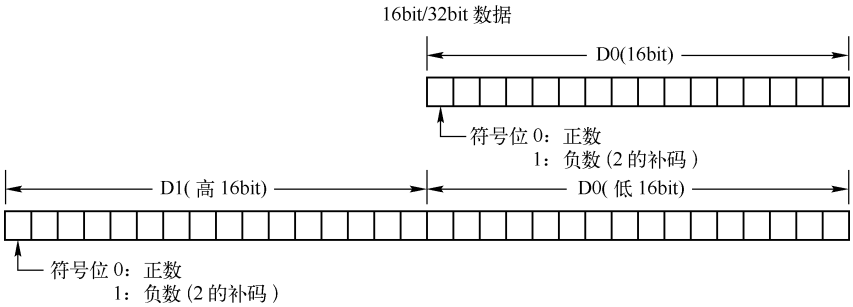


图 1-17　数据寄存器

注：若将特殊辅助继电器 M8033 置 1，在 PLC 由 RUN 转为 STOP 时，数据可以保持。

（2）停电保持数据寄存器 D200 ~ D511（312 点）（可通过参数设置改为通用型）　同上，除非改写，否则原有数据不会丢失。不论电源接通与否，PLC 运行与否，其内容也不变化。在两台 PLC 作点对点通信时，D490 ~ D509 被用作通信操作。

（3）停电保持专用数据寄存器 D512 ~ D7999（7488 点）　参数设置无法改变其保持的性质，但通过参数设置可将 D1000 以后的最多 7000 点设为文件寄存器。

（4）特殊数据寄存器 D8000 ~ D8255（256 点）　这些数据寄存器供监控 PLC 中各种元件的运行方式之用，其内容在电源接通（ON）时写入初始化值（先全部清零，然后由系统 ROM 安排写入初始值）。

例如，D8000 所存放警戒监视时钟（Watchdog Timer）的时间是由系统 ROM 设定的，要改变时，用传送指令将目的时间送入 D8000。该值在运行（RUN）→停止（STOP）时保持不变。

注：未定义的特殊数据寄存器请用户不要使用。

（5）文件寄存器 D1000 ~ D7999　文件寄存器是一类专用数据寄存器，用于存储大量重要数据，例如采集数据、统计计算数据、控制参数、配方等。FX_{2N} 可编程序控制器的数据寄存器区域，从 D1000 开始，以 500 点为一个子文件。最多可设置 14 个子文件，即 500 × 14 = 7000 点作为文件寄存器。D1000 ~ D7999 中不作文件寄存器的部分，仍可作为一般使用的停电保持型数据寄存器。

在这里要注意，FX_{2N} 的文件寄存器同时储存在机内两个不同的地方。存在程序存储器中（RAM，EEPROM）的称为［A］部，存在系统 RAM 中的称为［B］部，如图 1-18 所示。［A］与［B］的地址相同。

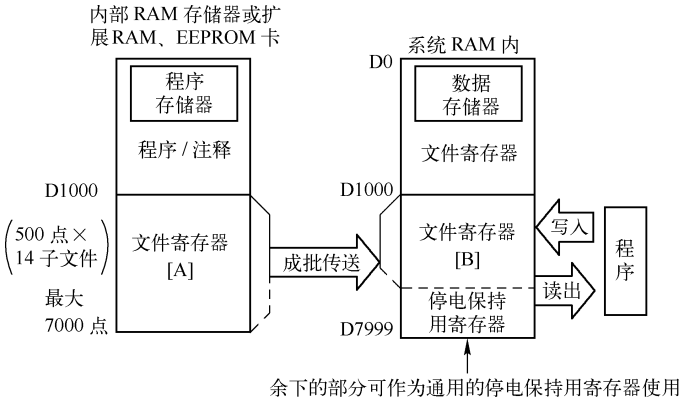


图 1-18　文件寄存器传送

用外围设备（编程器等）写文件寄存器时，是把数据写入［A］部。然后系统自动将［A］的内容复制（copy）到文件寄存器［B］。程序运行时，一般情况下，无论读/写操作，均访问文件寄存器［B］，文件寄存器［A］的内容不受影响。在［B］中没有被设置成文件寄存器的部分，尽管是停电保持的，但在下一次开机时（由 STOP→RUN），系统自动将［A］的内容复制到［B］，即将［B］初始化了。如果要改变［A］的内容就要用 FNC15（BMOV）指令，该指令的用法稍后再述。

从外围设备监视文件寄存器时，读出的是文件寄存器［B］的内容。而从外围设备对 PLC 的文件寄存器做“当前值变更”、“强制复位”、“PLC 存储器全清”操作时，操作对象则是［A］，随后系统自动将［A］的内容复制到［B］。进行上述操作时，EEPROM 卡的写保护开关必须置于 OFF 位置。

至于程序运行期间文件寄存器［A］与［B］相互间数据的交换，可以利用 FNC15（BMOV）指令来实现。例如，用该指令将文件寄存器［B］的内容读出，如图 1-19 所示。当 X0 有上升沿时，将文件寄存器［B］的内容读出。［S·］指源，［D·］叫指目标，*n* 指文件数目。即将 D200 开始的 400 个数据寄存器的内容复制到 D200 开始的 400 个数据寄存器中，当然源和目标也可以同在文件寄存器内。上述读出操作，数据只在系统 RAM 内流动。

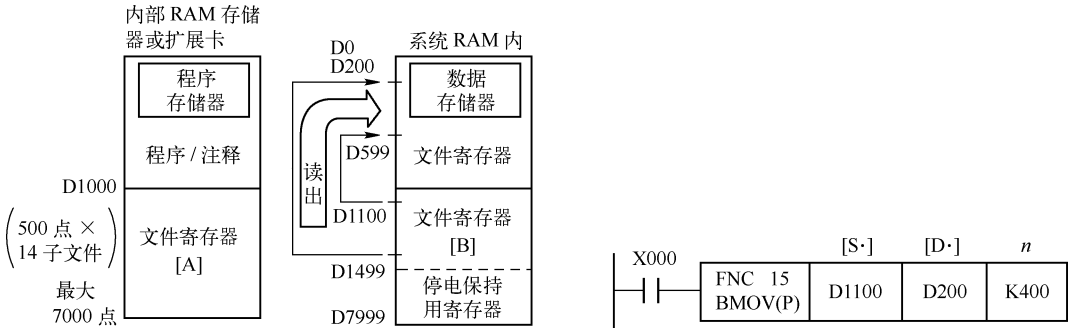


图 1-19　读文件寄存器

也可用 FNC15 BMOV 指令来写文件寄存器，如图 1-20 所示。当 X1 有上升沿时向文件寄存器［B］及［A］写入数据；若［A］不能写入（写保护），则只向［B］写入。

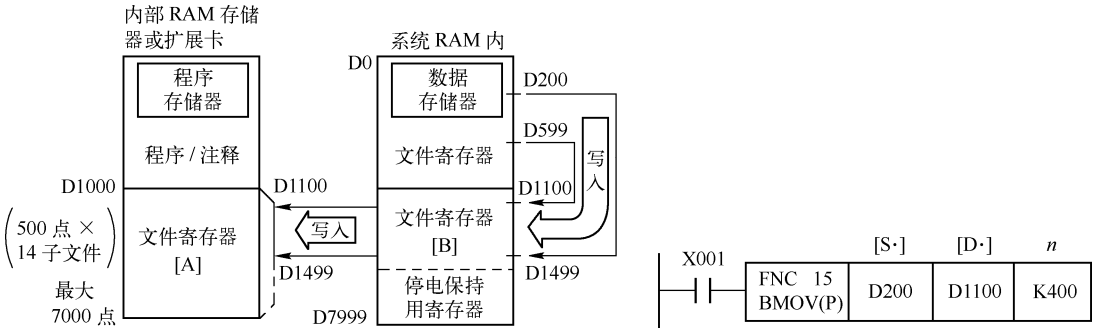


图 1-20　写文件寄存器

1）控制 FNC15（BMOV）指令的方向标志 M8024，可以改变数据传送方向，如图 1-21 所示。

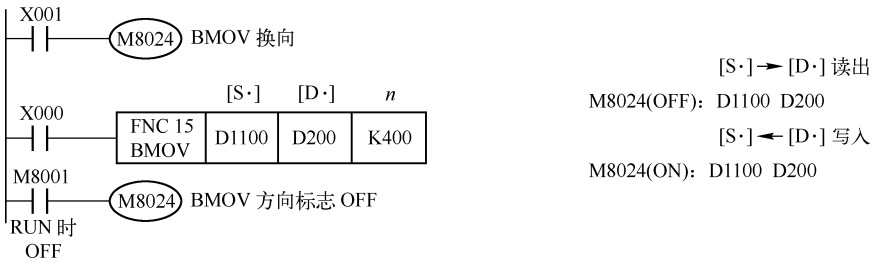


图 1-21　改变数据传送方向

2）若 FNC15 BMOV 指令中，源［S・］和目标［D・］指定的地址相同，则数据在文件寄存器［A］与［B］之间传送，方向仍由 M8024 的 ON/OFF 来决定。

M8024 为 OFF 时，从［A］→［B］，如图 1-22 所示，M8024 = OFF，当 X2 在上升沿时，把文件寄存器［A］从 D1100 开始的 400 个数据寄存器的内容复制到文件寄存器［B］D1100 开始的 400 个数据寄存器中。

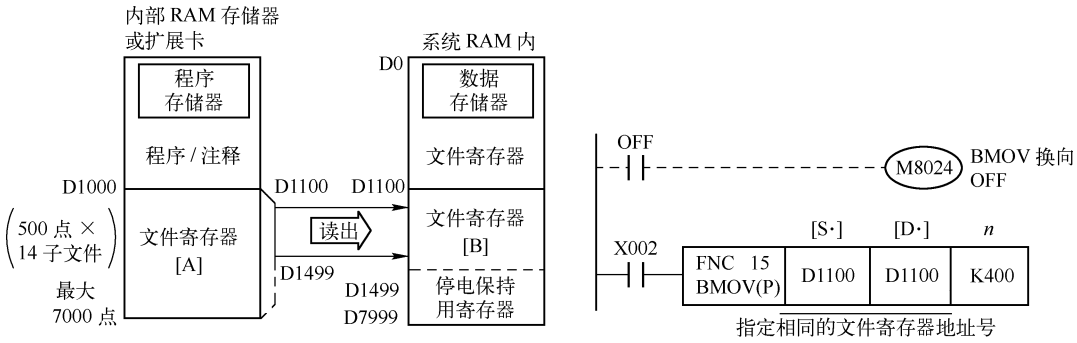


图 1-22　读出示例

M8024 = ON，从［B］→［A］，如图 1-23 所示，X1 = ON 时，M8024 = ON，数据从［B］→［A］，当 X3 有上升沿时，文件寄存器［B］从 D1100 开始的 400 个数据寄存器的内容复制到文件寄存器［A］的相应文件中。

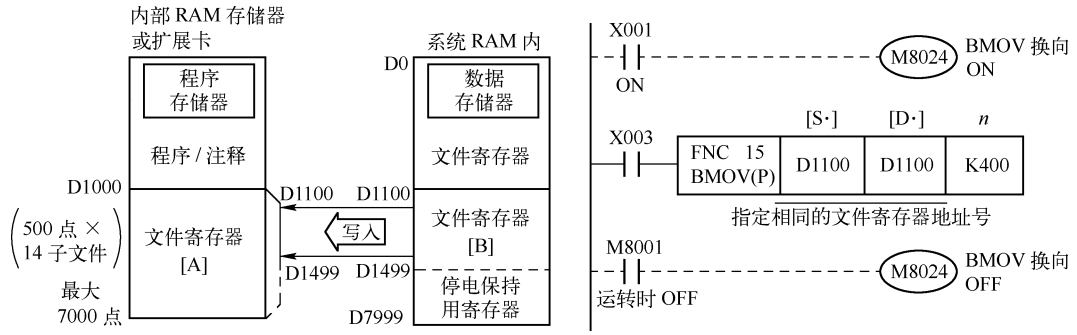


图 1-23　写入示例

注意，利用 FNC15 BMOV 指令实现［A］和［B］间的数据传送时，除应使源和目标地址相同外，还要注意使 n 的数目不应跨越不同的子文件，否则指令不能执行。

1.4.9 变址寄存器（V/Z）（字）

变址寄存器 V 与 Z 和普通用途的数据寄存器一样，是进行数据的写入和读出的 16 位数据寄存器。进行 32 位运算时，将两者组合使用，指定 Z 变址寄存器中（即 Z 为低位，V 为高位）V 和 Z 的主要功能是能够修改软器件的地址，能够修改的软器件有 X、Y、M、S、P、T、C、D、K、H、KnX；但不能修改 V 与 Z 本身以及指定用的 Kn 本身；如 K4M0Z0 有效，K0Z0M0 无效。变址应用示例如图 1-24 所示。

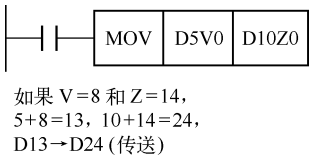


图 1-24　变址应用示例

习题

1. PLC 由哪几部分组成？简述各部分的功能。
2. PLC 有哪几种编程器件？各有什么特点？
3. 什么叫做 PLC 的扫描周期？其长短与什么有关？
4. 在一个扫描周期中，如果程序执行器件输入状态发生变化，输入映像寄存器的状态及输出状态是否随之变化？试说明原因。

第 2 章　基本逻辑指令

基本逻辑指令是 PLC 中最基础的编程语言，掌握了基本逻辑指令也就初步掌握了 PLC 的使用方法。

FX_{2N}型 PLC 共有 27 条基本指令，本章主要介绍这些基本指令的功能、梯形图的编制方法以及对应的指令表程序。最后，通过几个实例说明如何从实际控制要求出发，编制能满足要求的 PLC 程序。

2.1　逻辑取及输出线圈（LD/LDI/OUT）

逻辑取及输出线圈指令见表 2-1，其应用如图 2-1 所示。

表 2-1　逻辑取及输出线圈指令表

符号、名称	功　　能	电路表示及操作元件	程序步
LD（取）	常开触点逻辑运算开始	X,Y,M,S,T,C	1
LDI（取反）	常闭触点逻辑运算开始	X,Y,M,S,T,C	1
OUT（输出）	线圈驱动	Y,M,S,T,C	Y，M：1 特 M：2 T：3 C：3～5

LD：取指令。表示一个与输入母线相连的常开触点指令。

LDI：取反指令。表示一个与输入母线相连的常闭触点指令。

OUT：线圈驱动指令，也叫输出指令。

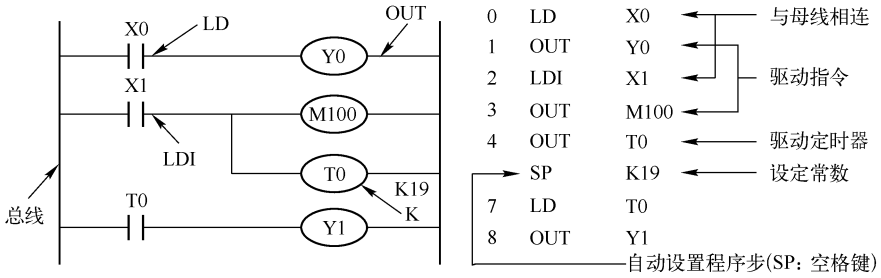


图 2-1　LD/LDI/OUT 指令应用示例

说明：

1）LD、LDI 两条指令可以与后述的 ANB 指令组合，在分支起点处也可使用。

2）OUT 指令是对输出继电器、辅助继电器、状态继电器、定时器、计数器的线圈的驱

动指令，对于输入继电器不能使用。

3）并行输出指令可多次使用（如图 2-1 中的 OUT T0 和 OUT M100）。

4）LD、LDI 是一个程序步指令，这里的一个程序步即是一个字。OUT 是多程序步指令，要视目标元件而定。

5）OUT 指令的目标元件是定时器 T 和计数器 C 时，必须设置常数 K。

2.2 触点串联（AND/ANI）

触点串联指令见表 2-2，其应用如图 2-2 所示。

表 2-2 触点串联指令表

符号、名称	功　　能	电路表示及操作元件	程序步
AND（与）	常开触点串联连接	X,Y,M,S,T,C	1
ANI（与非）	常闭触点串联连接	X,Y,M,S,T,C	1

AND：与指令。用于单个常开触点的串联。

ANI：与非指令。用于单个常闭触点的串联。

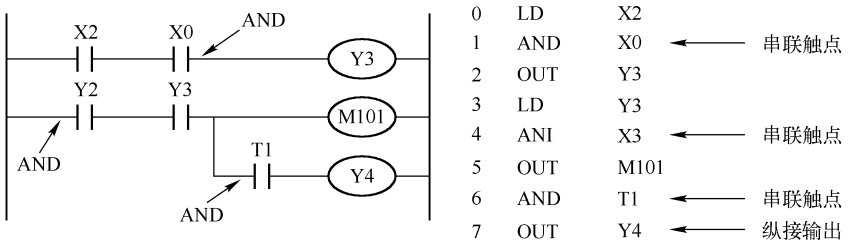


图 2-2 AND/ANI 指令应用示例

说明：

1）用 AND、ANI 指令，可进行触点的串联连接。串联触点的个数没有限制，该指令可以多次重复使用。

2）用 OUT 指令后，通过触点对其他线圈使用 OUT 指令称为纵接输出或连续输出，如图 2-2 中的 OUT　Y4。这种纵接输出，如果顺序不错，可以多次重复。但如果驱动顺序换成图 2-3 所示的形式，则必须应用后文中提到的 MPS 指令。这会导致程序步增多，因此不推荐使用。

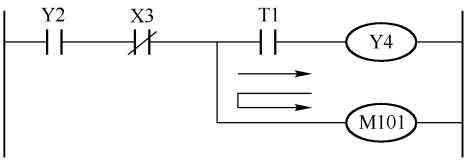


图 2-3 驱动顺序的形式

2.3 触点并联（OR/ORI）

触点并联指令见表 2–3，其应用示例如图 2–4 所示。

表 2–3 触点并联指令表

符号、名称	功　　能	电路表示及操作元件	程序步
OR（或）	常开触点并联连接	X,Y,M,S,T,C	1
ORI（或非）	常闭触点并联连接	X,Y,M,S,T,C	1

OR：或指令。用于单个常开触点的并联。

ORI：或非指令。用于单个常闭触点的并联。

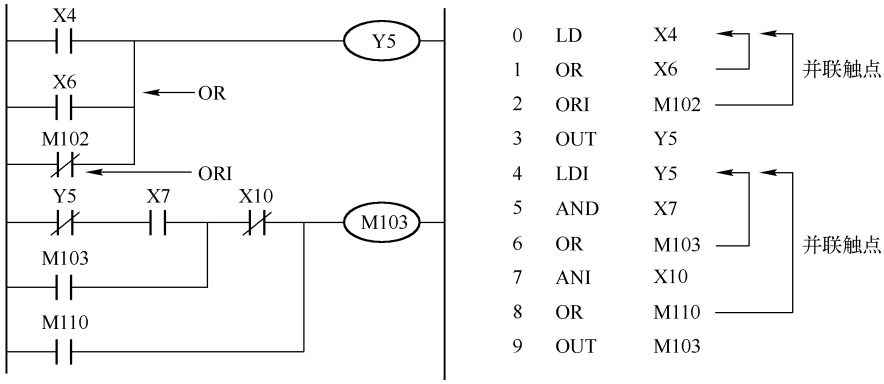


图 2–4 OR/ORI 指令应用示例

说明：

1）OR、ORI 用作 1 个触点的并联连接指令，2 个以上的触点串联连接的电路块的并联连接时，要用后述的 ORB 指令。

2）OR、ORI 指令是从该指令的当前步开始，对前面的 LD、LDI 指令并联连接。并联连接的次数无限制，但由于编程器和打印机的功能对此有限制，所以并联连接的次数实际是有限制的（24 行以下）。

2.4 串联电路块的并联（ORB）

串联电路块的并联指令见表 2–4，其应用示例如图 2–5 所示。

表 2–4 串联电路块的并联指令表

符号、名称	功　　能	电路表示及操作元件	程序步
ORB（电路块或）	串联电路的并联连接	操作元件：无	1

说明：

1）2 个以上的触点串联连接的电路称之为串联电路块。串联电路块并联连接时，分支开始用 LD、LDI 指令，分支结束用 ORB 指令。

2）ORB 指令与后述的 ANB 指令等均为无操作元件号的指令。

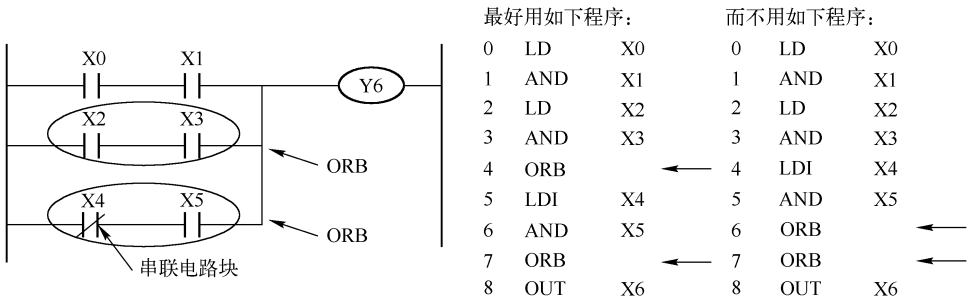


图 2-5　ORB 指令应用示例

注意：

1）每一电路块使用 ORB 指令（参见上述编得好的程序），则并联电路块数无限制。

2）ORB 指令也可连续使用（见上述编得不好的程序），但这样使用时，重复使用 LD、LDI 指令的次数限制在 8 次以下。这点要特别注意。

2.5　并联电路块的串联（ANB）

并联电路块的串联指令见表 2-5，其应用示例如图 2-6 所示。

表 2-5　并联电路块的串联指令表

符号、名称	功　　能	电路表示及操作元件	程序步
ANB （电路块与）	并联电路块之间 的串联连接	操作元件：无	1

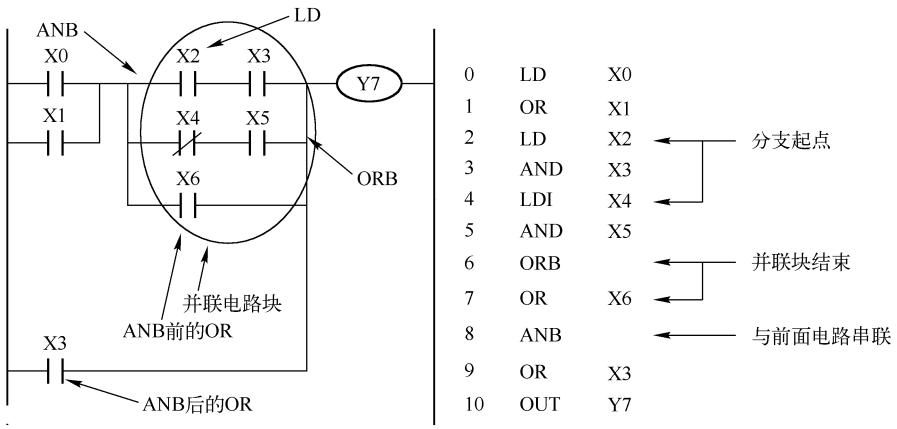


图 2-6　ANB 指令应用示例

说明：

1）分支电路并联电路块与前面电路串联连接时，使用 ANB 指令。分支的起始点用 LD、LDI 指令。并联电路块结束后，使用 ANB 指令与前面电路串联。

2）若多个并联电路块顺次用 ANB 指令与前面电路串联连接，则 ANB 的使用次数没有限制。

3）虽然可以连续使用 ANB 指令，但这时与 ORB 指令同样，要注意 LD、LDI 指令的使用次数限制（8 次以下）。

2.6 多重输出电路（MPS/MRD/MPP）

多重输出电路指令见表 2-6。

表 2-6 多重输出电路指令表

符号、名称	功　　能	电路表示及操作元件	程序步
MPS（Push）	进栈	MPS	1
MRD（Read）	读栈	MRD	1
MPP（POP）	出栈	MPP	1

MPS、MRD、MPP 为无操作元件指令，都为一个程序步长。这组指令用于多输出电路。可将连接点先存储，因此可用于连接后面的电路。

FX 系列 PLC 中，有 11 个存储运算中间结果的存储器，被称为栈存储器，MPS 和 MPP 指令必须成对使用，而且连续使用的次数应少于 11 次。

使用一次 MPS 指令，该时刻的运算结果就压入栈的第一层。再次使用 MPS 指令时，当时的运算结果压入栈的第一层，先压入的数据依次向栈的下一段压移。

使用 MPP 指令，各数据依次向上层托移。最上层的数据在读出后就从栈内消失。

MRD 是最上层所存的最新数据的读出专用指令。栈内的数据不发生下压或上托。使用 MPS 指令、MPP 指令、MRD 指令栈内的数据存储如图 2-7 所示。

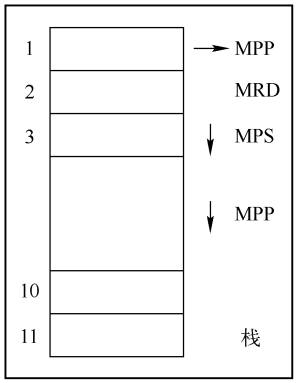


图 2-7　栈存储器

与多重输出电路指令相关的使用，分别如图 2-8 ~ 图 2-11 所示。

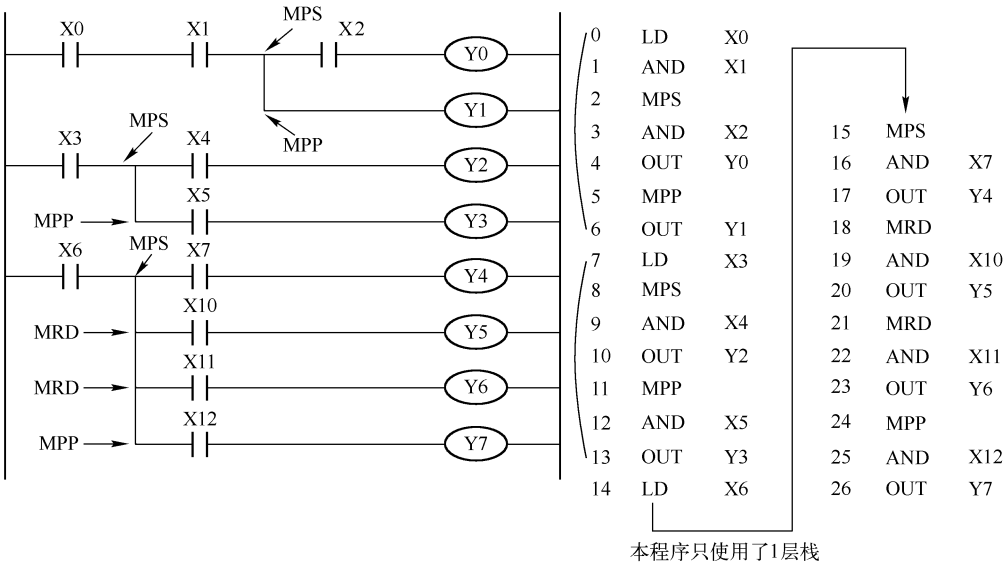


图 2-8　MPS/MRD/MPP 简单电路例（1 层栈）

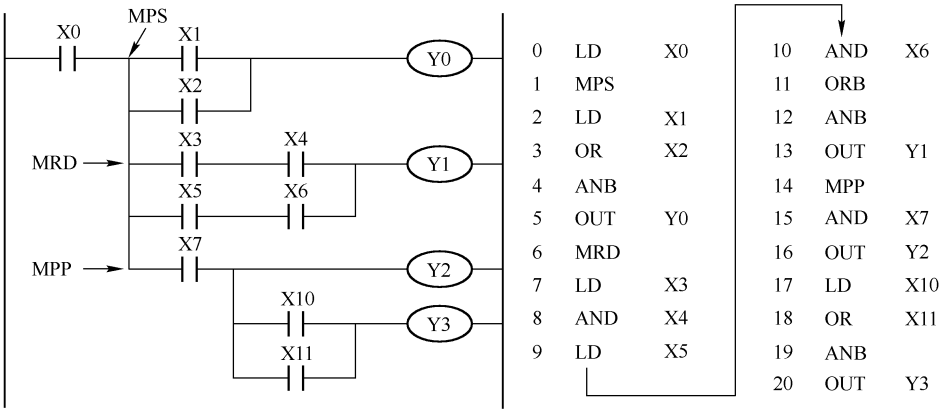


图 2-9　1 层栈和 ANB、ORB 指令应用示例

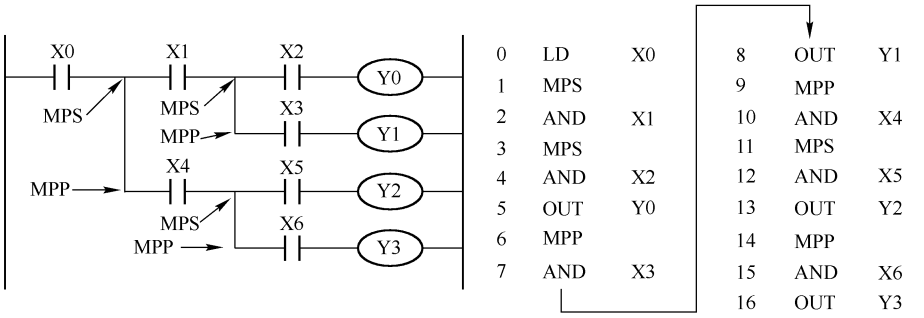


图 2-10　2 层栈应用示例

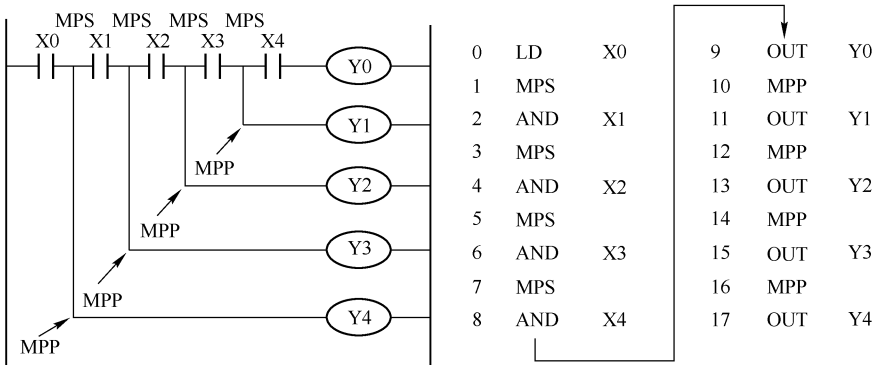


图 2-11　4 层栈应用示例

2.7　主控触点（MC/MCR）

主控触点指令见表 2-7。

表 2-7　主控触点指令表

符号、名称	功　　能	电路表示及操作元件	程序步
MC（主控）	主控电路块起点	MC N Y,M；Y,M 不允许使用特M	3
MCR（主控复位）	主控电路块终点	MCR N	2

MC 为主控指令，用于公用串联触点的连接。

MCR 为主控复位指令，即 MC 的复位指令。

在编程时，经常遇到多个线圈同时受一个或一组触点控制。如果在每个线圈的控制电路中都串入同样的触点，将多占用存储单元，应用主控指令可以解决这一问题。只用主控指令的触点称为主控触点，它在梯形图中与一般的触点垂直。它们是与母线相连的常开接触点，是控制一组电路的总开关。MC、MCR 应用示例如图 2-12 所示。

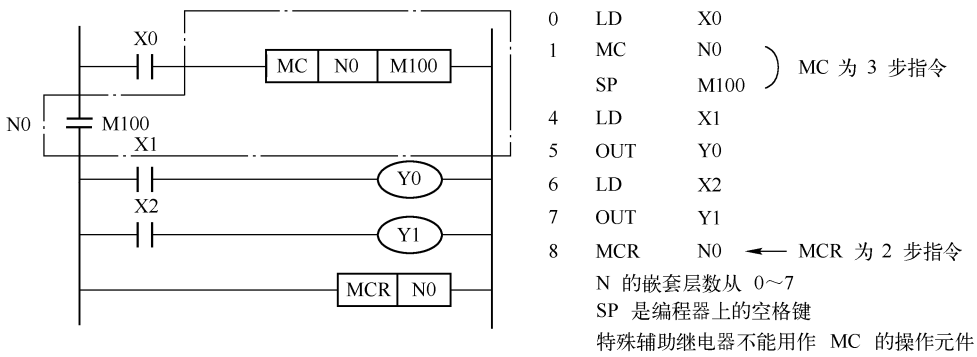


图 2-12　主控触点指令应用示例

1. 说明

1）输入 X0 接通时，执行 MC 与 MCR 之间的指令。输入 X0 断开时，成为如下形式：

① 保持当前状态的元件：积算定时器、计数器及用 SET/RST 指令驱动的元件。

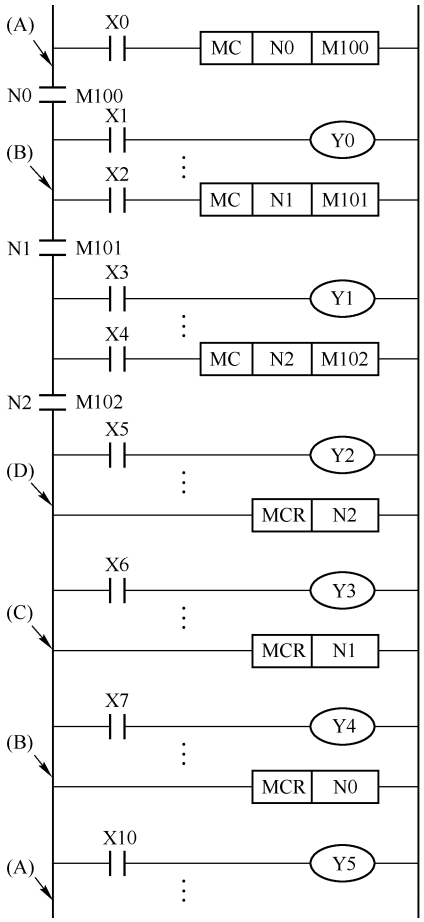
② 变成断开的元件：非积算定时器，用 OUT 指令驱动的元件。

2）MC 指令后，母线（LD，LDI 点）移至 MC 触点之后，返回原来母线的指令是 MCR。MC 指令使用后必定要用 MCR 指令。

3）使用不同的 Y、M 元件号，可多次使用 MC 指令。但是若用同一软元件号，就与 OUT 指令一样成为双线圈输出。

4）在 MC 指令内再使用 MC 指令时，嵌套级 N 的编号就顺次增大（按程序顺序由小到大）。返回时用 MCR 指令，就从大的嵌套级开始解除（按程序顺序由大至小）。

2. 嵌套级（见图 2-13）



级 N0
母线 B 在 X0 接通时成为有效状态

级 N1
母线 C 在 X0、X2 同时接通时成为有效状态

级 N2
母线 D 在 X0、X2、X4 全部接通时成为有效状态

级 N1
根据 MCR N2，母线返回 C 的状态

级 N0
根据 MCR N1，母线返回 B 的状态

初始状态
根据 MCR N0，母线返回开始的 A 状态
因此，输出线圈 Y5 的通断只取决于 X10 的通断，而与 X0、X2、X4 的通断无关

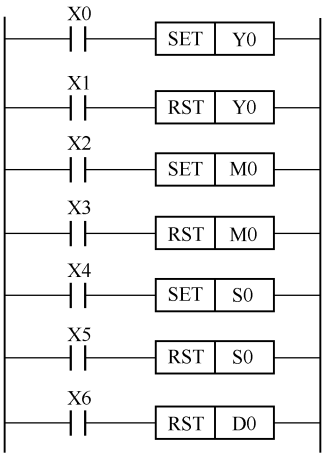
图 2-13　嵌套级示例

2.8　自保持与解除（SET/RST）

自保持与解除指令见表 2-8。其应用示例如图 2-14 所示。

表 2-8　自保持与解除指令表

符号、名称	功　　能	电路表示及操作元件	程序步
SET（置位）	令元件自保持 ON	SET Y,M,S	Y，M：1 S，特 M：2
RST（复位）	令元件自保持 OFF 清数据寄存器	RST Y,M,S,D,V,Z	D，V，Z，特 D：3



0	LD	X0
1	SET	Y0
2	LD	X1
3	RST	Y0
4	LD	X2
5	SET	M0
6	LD	X3
7	RST	M0
8	LD	X4
9	SET	S0
11	LD	X5
12	RST	S0
14	LD	X6
15	RST	D0

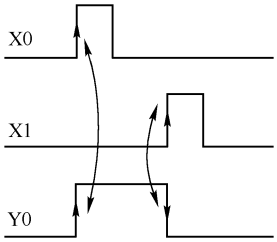


图 2-14　SET/RST 指令应用示例

说明：

1）X0 一接通，即使再变成断开，Y0 也保持接通。X1 接通后，即使再变成断开，Y0 也将保持断开。对于 M、S 也是同样。

2）对同一元件可以多次使用 SET、RST 指令，顺序可任意，但在最后执行的一条才有效。

3）要使数据寄存器 D、变址寄存器 V、Z 的内容清零，也可用 RST 指令（用常数为 K0 的传送指令也可得到同样的结果）。

2.9　计数器、定时器（OUT/RST）

计数器、定时器（OUT/RST）指令见表 2-9。其应用示例如图 2-15 所示。

表 2-9　计数器、定时器指令表

符号、名称	功　　能	电路表示及操作元件	程序步
OUT（输出）	驱动定时器线圈 驱动计数器线圈	T,C　K△△	32 bit 计数器：5 其他：3
RST（复位）	复位输出触点 当前数据清零	RST T,C	2

1. 积算定时器（1 ms 定时器，100 ms 定时器）

输入 X1 接通期间，T246 接收 1 ms 时钟脉冲并计数，到达 1234 时 Y0 就动作。

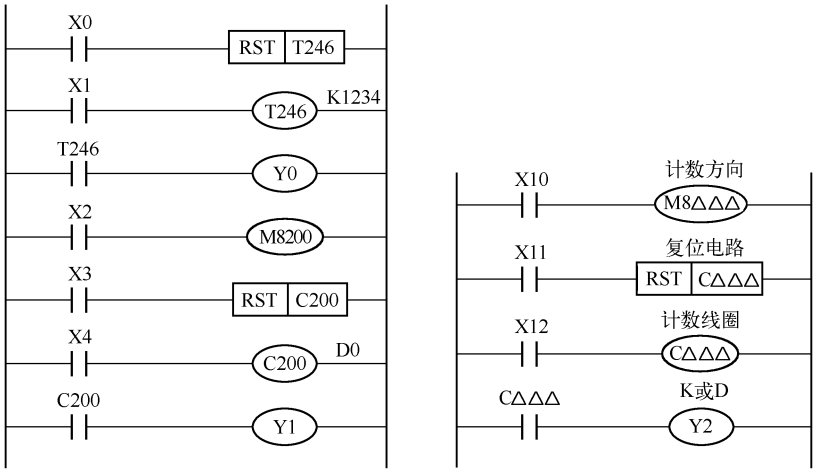


图 2-15　计数器、定时器应用示例

X0 一接通，输出触点 T246 就复位，定时器的当前值也成为 0。

2. 内部计数器

32 bit 计数器 C200 根据 M8200 的 ON/OFF 状态进行计数（增计数、减计数），它对 X4 触点的 OFF→ON 的次数进行计数。

输出触点的置位或复位取决于计数方向及达到 DI、D0 中存的设定值。

输入 X3 接通后，输出触点复位，计数器当前值清零。

3. 高速计数器

1）对于 C235 ~ C245 的单相单输入计数器，须用特殊辅助继电器（M8235 ~ M8245）指定计数方向。X10：接通时增计数；X10：断开时减计数。

2）X11 接通，计数器 C△△△的输出触点就复位，计数器的当前值也清零。对于带有复位输入的计数器（C241 ~ C255 等），当复位输入接通时，不必其他编程也可实现同上动作。

3）X12 接通时，高速计数器 C235 ~ C240 分别对由计数输入 X0 ~ X5 输入的通/断进行计数，对于带有启动输入的计数器（C244、C245、C249、C250、C255），启动输入不接通就不进行计数。

4）计数器的当前值随计数输入的次数而增加，当该值等于设定值（K 或 D 的内容），计数器输出触点接通。

2.10　脉冲输出（PLS/PLF）

脉冲输出指令见表 2-10。其应用示例如图 2-16 所示。

表 2-10　脉冲输出指令表

符号、名称	功　　能	电路表示及操作元件	程序步
PLS（上升沿脉冲）	上升沿微分输出	PLS Y,M	2
PLF（下降沿脉冲）	下降沿微分输出	PLF Y,M	2

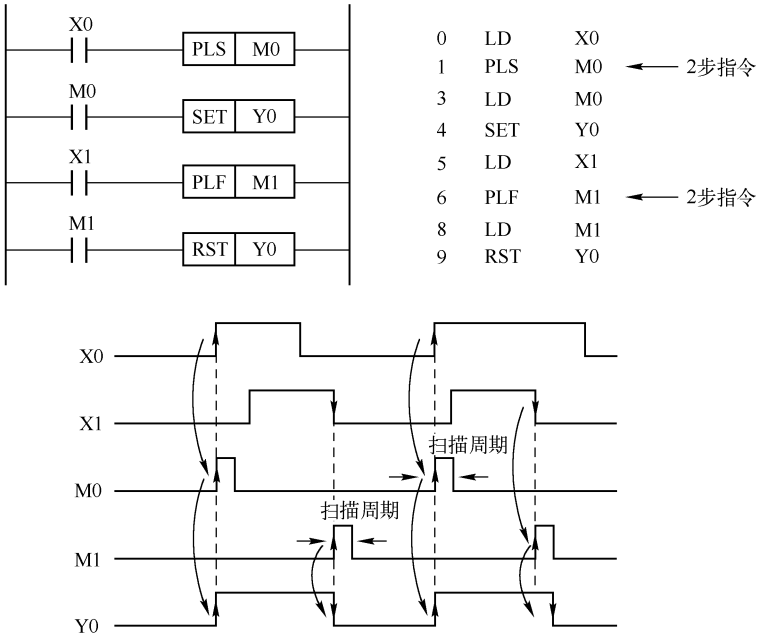


图 2-16　PLS/PLF 指令应用示例

说明：

1）使用 PLS 指令，元件 Y、M 仅在驱动输入接通后的一个扫描周期内动作（置 1）。

2）使用 PLF 指令，元件 Y、M 仅在驱动输入断开后的一个扫描周期内动作。

3）特殊继电器不能用作 PLS 或 PLF 的操作元件。

4）在驱动输入接通时，PLC 由运行→停机→运行，此时 PLS M0 动作，但 PLS M600 断电时由电池后备的辅助继电器不动作。这是因为 M600 是保持继电器，即使在断电停机时其动作也能保持。

2.11　脉冲式触点指令（LDP/LDF/ANDP/ANDF/ORP/ORF）

脉冲式触点指令见表 2-11。

表 2-11　脉冲式触点指令表

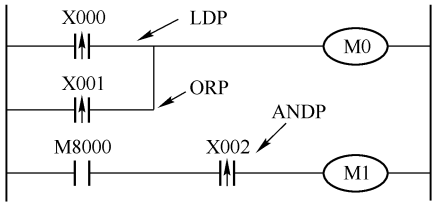
符号、名称	功　　能	电路表示及操作元件	程序步
LDP（取上升沿脉冲）	上升沿脉冲逻辑运算开始	X,Y,M,S,T,C	2
LDF（取下降沿脉冲）	下降沿脉冲逻辑运算开始	X,Y,M,S,T,C	2
ANDP（与上升沿脉冲）	上升沿脉冲串联连接	X,Y,M,S,T,C	2
ANDF（与下降沿脉冲）	下降沿脉冲串联连接	X,Y,M,S,T,C	2

（续）

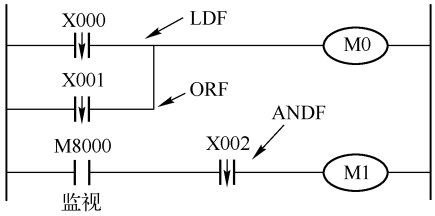
符号、名称	功　　能	电路表示及操作元件	程序步
ORP（或上升沿脉冲）	上升沿脉冲并联连接	X,Y,M,S,T,C	2
ORF（或下降沿脉冲）	下降沿脉冲并联连接	X,Y,M,S,T,C	2

说明：

1）这是一组与 LD、AND、OR 指令相对应的脉冲式触点指令。指令中 P 对应上升沿脉冲，F 对应下降沿脉冲。指令中的触点仅在操作元件有上升沿/下降沿时导通一个扫描周期。如图 2-17 所示，X000～X002 由 OFF→ON 时或由 ON→OFF 变化时，M0 或 M1 接通一个扫描周期。



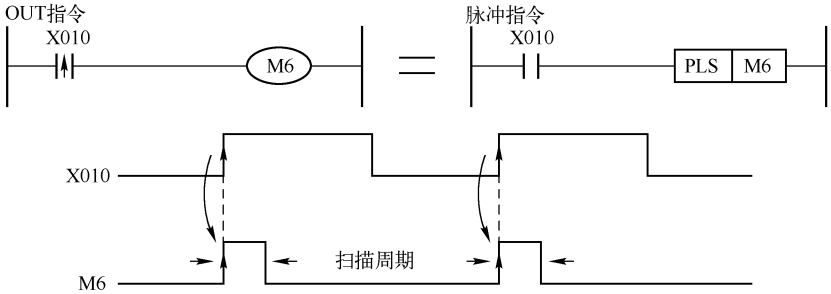
0	LDP	X000
2	ORP	X001
4	OUT	M0
5	LD	M8000
6	ANDP	X002
8	OUT	M1



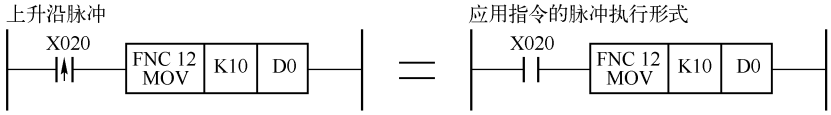
0	LDF	X000
2	ORF	X001
4	OUT	M0
5	LD	M8000
6	ANDF	X002
8	OUT	M1

图 2-17　LDP/ORP/ANDP/LDF/ORF/ANDF 指令应用示例

2）这组指令只是在某些场合为编程者提供了方便。图 2-18 所示为等效编程方法。



a) 结果都是在 X010 由 OFF➤ON 变化时，M6 接通一个扫描周期



b) 结果都是在 X020 由 OFF➤ON 变化时，只执行一次 MOV 指令

图 2-18　等效编程方法

3）当这组指令以辅助继电器 M 作为操作元件时，M 的序号会影响程序的执行情况。如图 2-19a 所示，M0 ~ M2799 作为操作元件时程序的执行是普通情况下的形式，X0 为 ON 后，M0 接通，① ~ ③接通一个扫描周期，而④是 LD 指令，故① ~ ④各行程序都执行。M50 ~ M53 都为 ON。

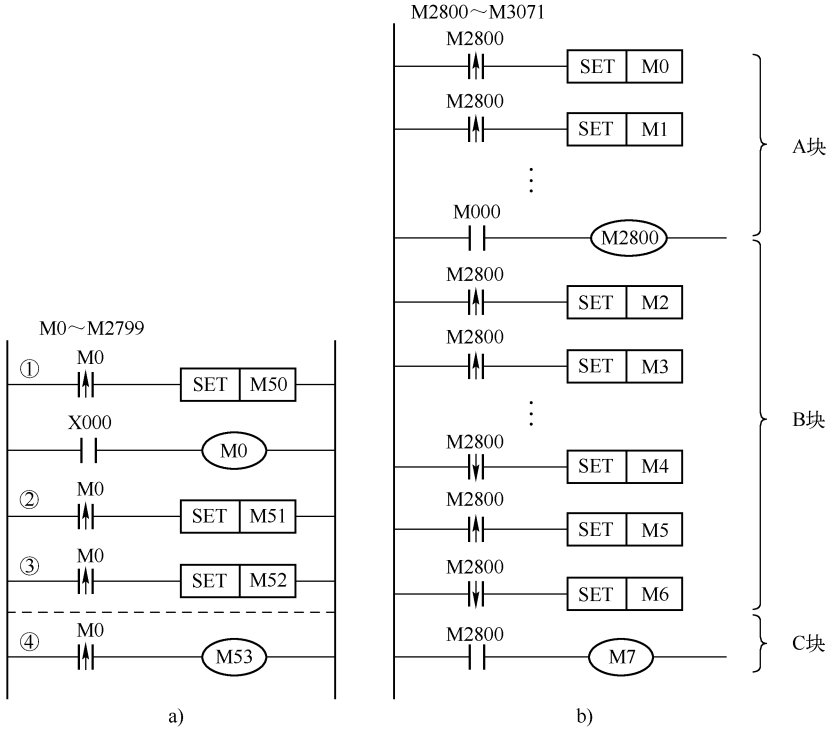


图 2-19　辅助继电器 M 作为脉冲式触点指令的操作元件

而以 M2800 ~ M3071 作为这组指令的操作元件时程序的执行就特殊了。当 M2800 的状态发生变化时，在其后一个扫描周期内只有第一个碰到的 M2800 的脉冲触点起作用。图 2-19b 中，当 M2800 由 OFF 到 ON 时只有 SET M2 被执行；当 M2800 由 ON 到 OFF 时，只有 SET M4 被执行。M7 的驱动用的是 LD 指令，不受上述说明影响。

这一特性在顺控步进指令中有重要用途。

2.12　逻辑运算结果取反（INV）

逻辑运算结果取反指令见表 2-12。其应用示例如图 2-20 所示。

表 2-12　逻辑运算结果取反指令

符号、名称	功　　能	电路表示及操作元件	程序步
INV（取反）	逻辑运算结果取反	操作元件：无	1

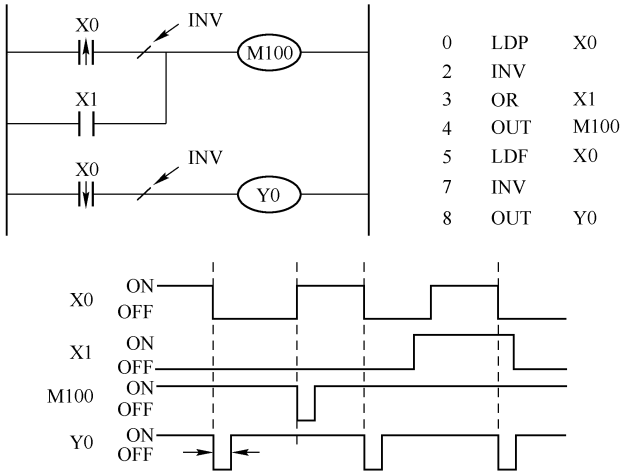


图 2-20　INV 指令应用示例

说明：INV 指令是把指令所在位置当前逻辑运算结果取反，取反后的结果仍可继续运算。INV 指令无需操作元件。

2.13　空操作指令（NOP）

空操作指令见表 2-13。图 2-21 所示为用 NOP 指令改变电路的例子。

表 2-13　空操作指令表

符号、名称	功　　能	电路表示及操作元件	程序步
NOP（空操作）	无动作	无元件	1

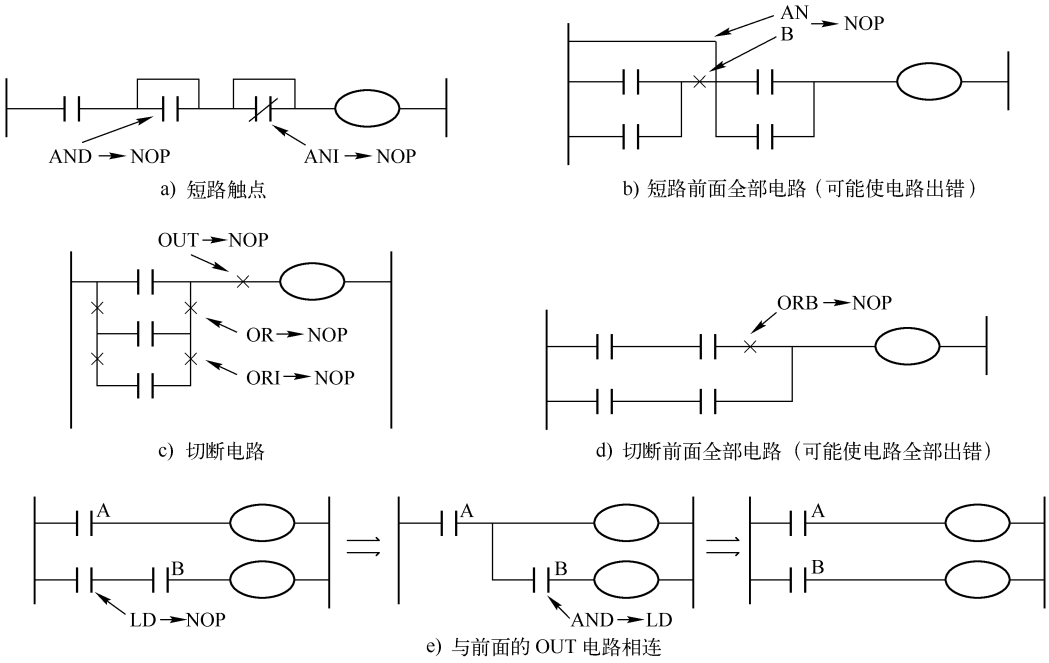


图 2-21　NOP 指令应用示例（用 NOP 指令改变电路）

说明：

1）程序若加入 NOP 指令，改动或追加程序时，可以减少步序号的改变。另外，用 NOP 指令替换已写入的指令，也可改变电路。

2）LD、LDI、ANB、ORB 等指令若换成 NOP 指令，应注意电路构成将有较大幅度变化。

3）行程全清操作后，全部指令都变成 NOP。

2.14 程序结束指令（END）

程序结束指令见表 2-14。END 指令应用示例如图 2-22 所示。

表 2-14 程序结束指令表

符号、名称	功　　能	电路表示及操作元件	程序步
END（结束）	输入输出处理 程序回第 0 步	END　无元件	1

PLC 反复进行输入处理、程序运算、输出处理。若在程序最后写入 END 指令，则 END 以后的程序步就不再执行，直接进行输出处理。在程序调试过程中，按段插入 END 指令，可以顺序扩大对各程序段动作的检查。采用 END 指令将程序划分为若干段。在确认处于前面电路块的动作正确无误之后，依次删去 END 指令。注意，执行 END 指令时，也刷新警戒时钟（Watchdog Timer）。

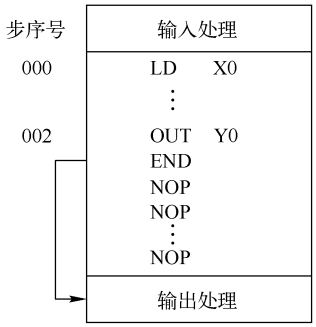
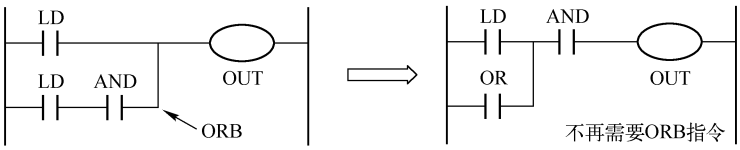


图 2-22 END 指令应用示例

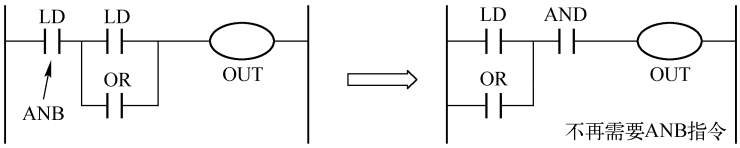
2.15 编程注意事项

1）程序应按自上而下，从左至右的方式编制。

2）适当的编程顺序可减少程序步数，如图 2-23 所示。



a）串联多的电路应尽量放在上部

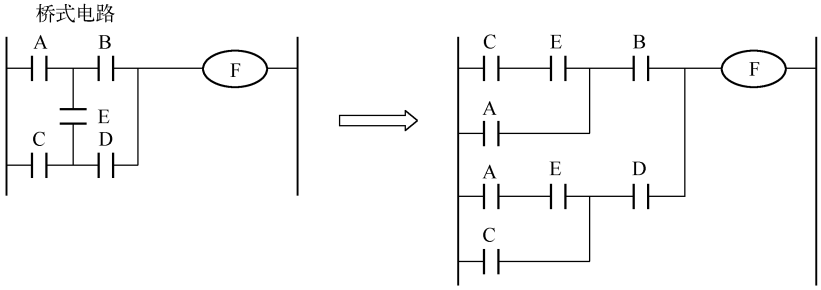


b）并联多的电路应尽量靠近母线

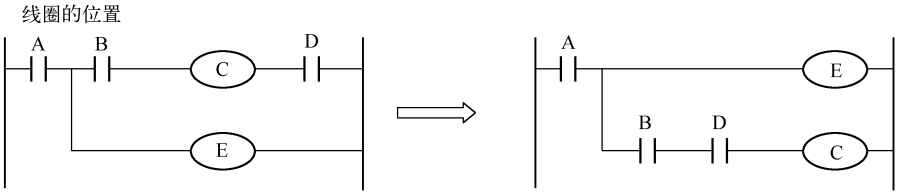
图 2-23 减少程序步数示例

3）重新安排不能编程的电路，如图 2-24 所示。

4）双线圈输出不可用，如图 2-25 所示。



a）将电路改成右图的形式才能编程



b）不能将触点画在线圈的右边，只能在触点的右边接线圈

图 2-24　重新安排不能编程的电路示例

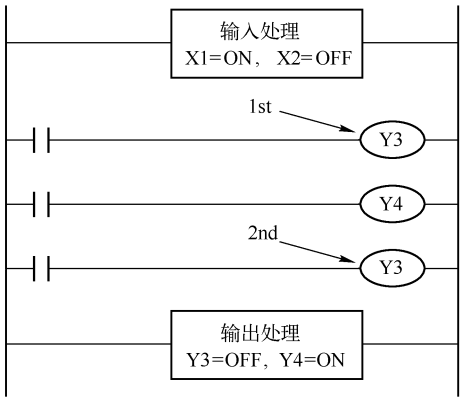


图 2-25　双线圈使用示例

2.16　编程实例

2.16.1　简单程序

1. 延时断定时器（见图 2-26）

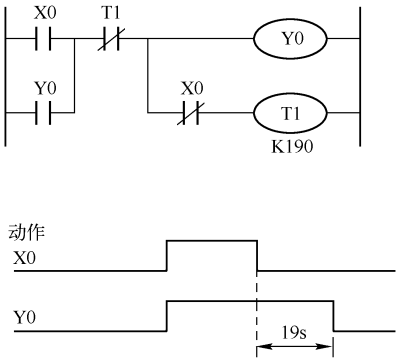
2. 振荡电路（见图 2-27）

3. 脉冲输出电路（见图 2-28）

说明：

1）X0 第一次闭合，Y1 立即接通。X0 再次闭合时，Y1 断开。

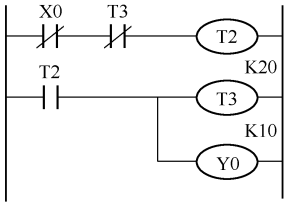
2）M103 只在一个扫描周期内接通（脉冲输出）。在程序中 M104 排在 M103 之后是很重要的。



步序	指令	
24	LD	X0
25	OR	Y0
26	ANI	T1
27	OUT	Y0
28	ANI	X0
29	OUT	T1
30	SP	K190

Y0 在 X0 断开后 19s 关断
在输入触点开断后，输出触点延时开断的定时器被认为是一个延时断定进器

图 2-26　延时断定时器



步序	指令	
31	LDI	X0
32	ANI	T3
33	OUT	T2
34	SP	K20
35	LD	T2
36	OUT	T3
37	SP	K10
38	OUT	Y0

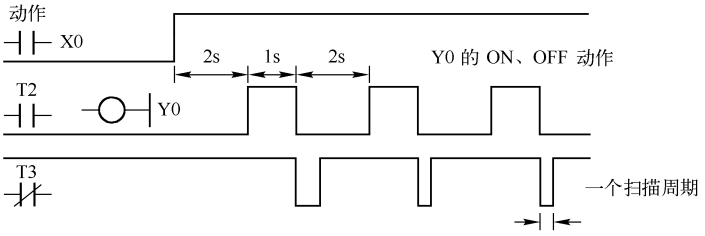
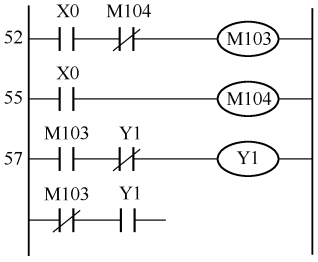


图 2-27　振荡电路



步序	指令	
52	LD	X0
53	ANI	M104
54	OUT	M103
55	LD	X0
56	OUT	M104
57	LD	M103
58	ANI	Y1
59	LDI	M103
60	AND	Y1
61	ORB	
62	OUT	Y1

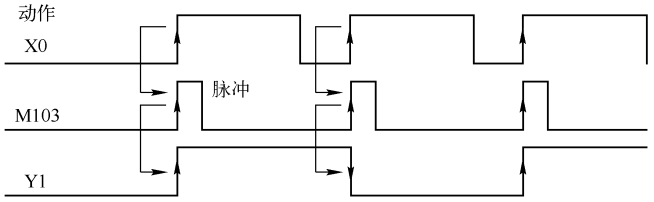


图 2-28　脉冲输出电路

3）按图 2-29 所示电路编程是错误的。

4）实际上，使用 PLS 指令就可很容易地获得像 M103 这样的脉冲输出。

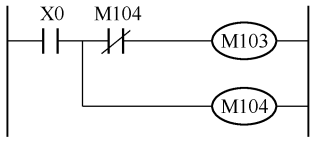


图 2-29　错误的编程电路

2.16.2　实例

1. 抢答显示系统（见图 2-30）

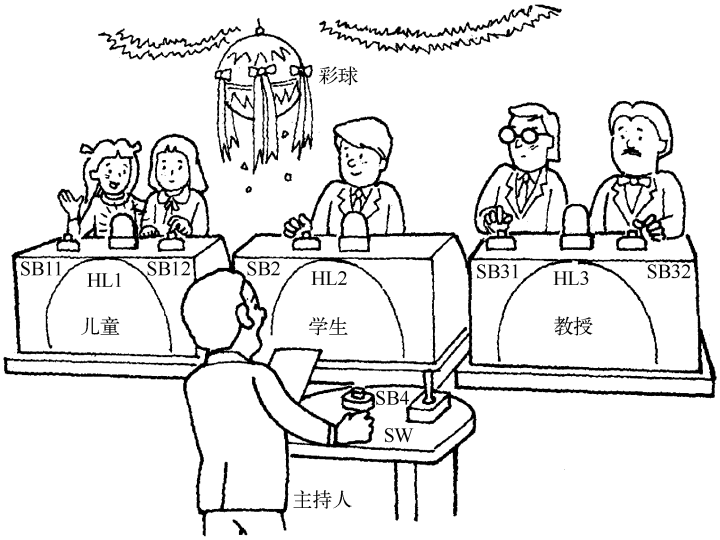


图 2-30　抢答显示系统

（1）控制要求

1）竞赛者若要回答主持人所提问题时，须抢先按下桌上的按钮。

2）指示灯亮后，须等到主持人按下复位键 SB4 后才熄灭。为了给参赛儿童一些优待，SB11 和 SB12 中任一个按下时，灯 HL1 都亮。而为了对教授组做一定限制，HL3 只有在 SB31 和 SB32 键都按下时才亮。

3）如果竞赛者在主持人合上 SW 开关的 10 s 内按下按钮，电磁线圈将使彩球摇动，以示竞赛者得到一次幸运的机会。

（2）本例的目的　设计用互锁和自锁电路为基础构成各输出电路的简单程序。

（3）选定输入/输出设备　在程序控制设计中，首先要确定需要使用哪些输入、输出，然后分别给它们标上可编程序控制器的端子号（见表 2-15）。

表 2-15　选定输入/输出设备

输 入 装 置	输入端子号	输 出 装 置	输出端子号
按钮 SB11	X000	灯 HL1	Y000
按钮 SB12	X001	灯 HL2	Y003
按钮 SB2	X003	灯 HL3	Y005
按钮 SB31	X004	电磁开关 SOL	Y007
按钮 SB32	X005		
按钮 SB4	X002		
选择开关 SW	X007		

（4）可编程序控制器外部接线图（见图 2-31）

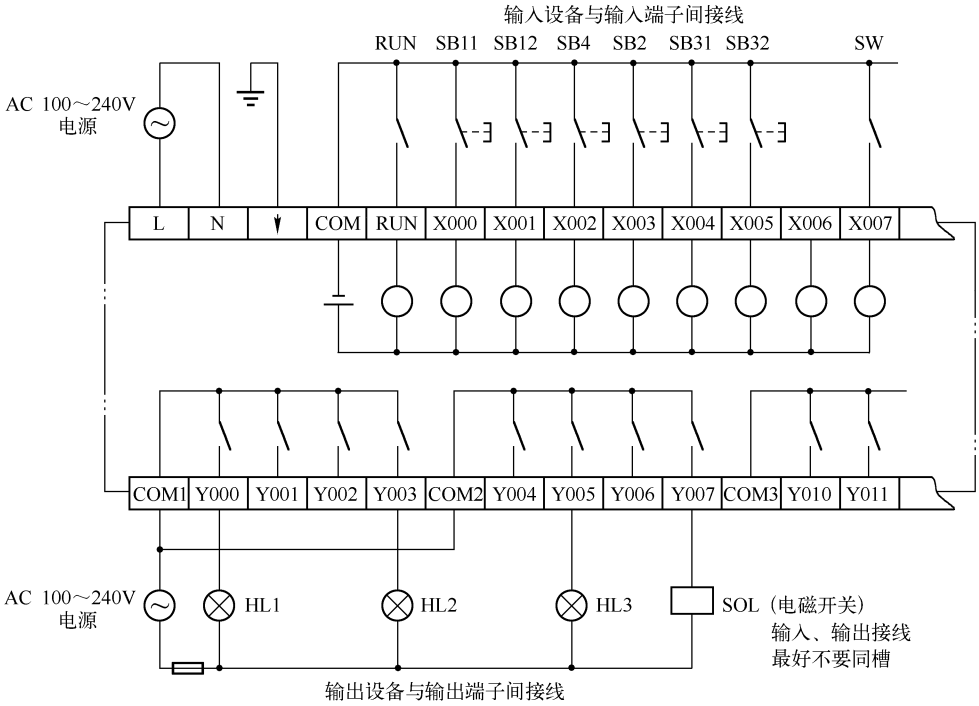


图 2-31　PLC 外部接线图

（5）画出控制逻辑图（梯形图）（见图 2-32）

可编程序控制器的控制逻辑电路设计与继电器箱完全相同。

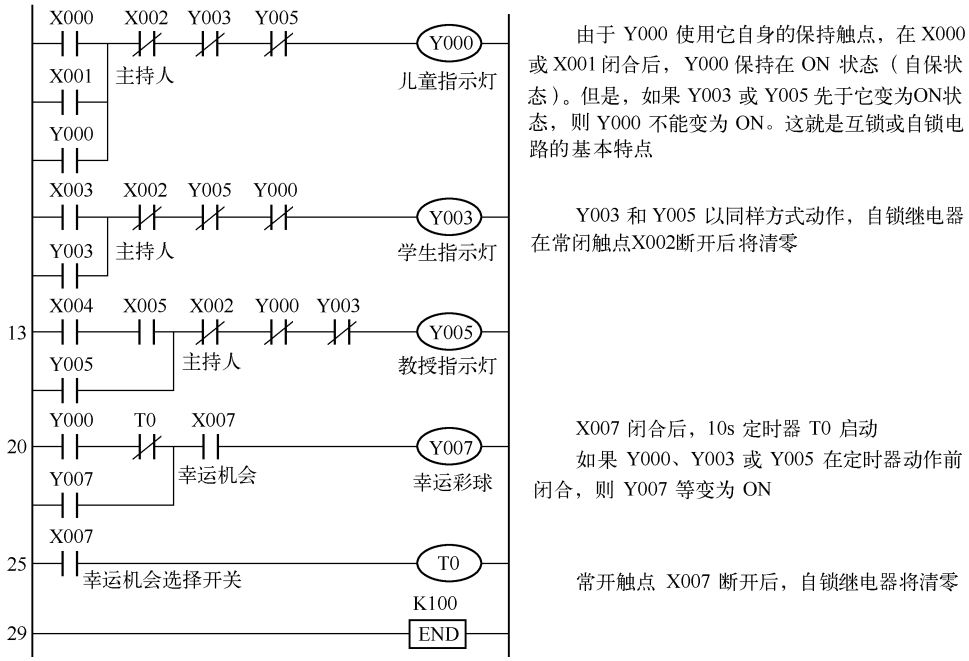


图 2-32　梯形图

（6）写出指令表（见表2-16）

表 2-16　指令表

序　号	指　　令	序　号	指　　令	序　号	指　　令
0	LD　X000	10	ANI　Y005	20	LD　Y000
1	OR　X001	11	ANI　Y000	21	ANI　T0
2	OR　Y000	12	OUT　Y003	22	OR　Y007
3	ANI　X002	13	LD　X004	23	AND　X007
4	ANI　Y003	14	AND　X005	24	OUT　Y007
5	ANI　Y005	15	OR　Y005	25	LD　X007
6	OUT　Y000	16	ANI　X002	26	OUT　T0 SP　K100
7	LD　X003	17	ANI　Y000		
8	OR　Y003	18	ANI　Y003	29	END
9	ANI　X002	19	OUT　Y005		

2. 料箱盛料过少报警系统（见图2-33）

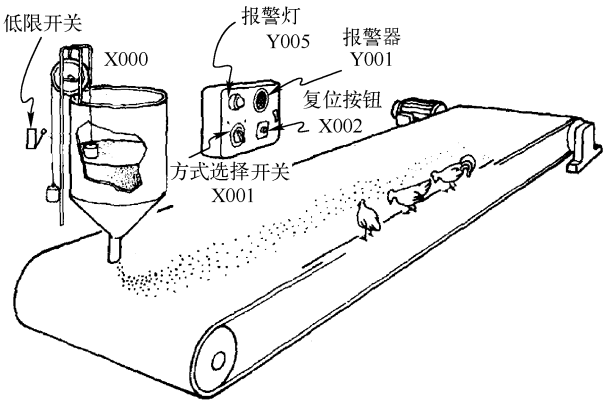


图 2-33　料箱盛料过少报警系统

（1）控制要求

1）自动方式（X001 = OFF）。当低限开关 X000 变为 ON 后，报警器 Y001 开始鸣叫，同时报警灯 Y005 连续闪烁 10 次（亮 1.5 s、灭 2.5 s）。此后，报警器停止鸣叫，灯也熄灭。此外，RESET（复位）按钮 X002 可以使两者中止。

2）手动方式（X001 = ON）。当低限开关 X000 变为 ON 后，报警器 Y001 开始鸣叫，同时报警灯 Y005 开始闪烁。当按下 RESET 按钮 X002 时，两者中止。

（2）本例的目的　设计一个包括计数器和定时器的闪烁电路的控制程序。

（3）画出控制逻辑图（梯形图）（见图2-34）

通过练习，应能直接根据逻辑电路编制程序。

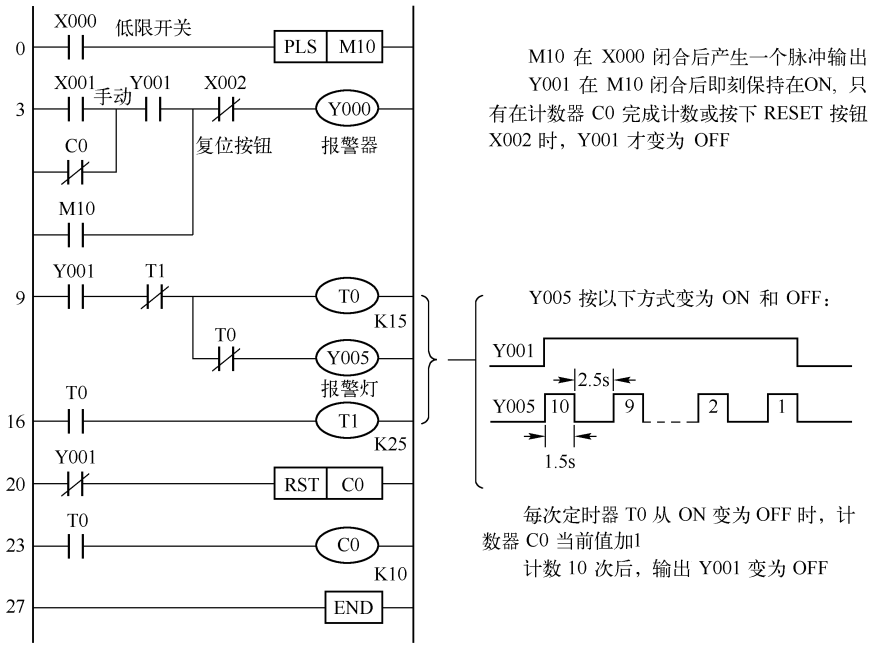


图 2-34　梯形图

（4）写出指令表（见表 2-17）

表 2-17　指令表

序　号	指　　令	序　号	指　　令	序　号	指　　令
0	LD　X000	9	LD　Y001		SP　K25
1	PLS　M10	10	ANI　T1	20	LDI　Y001
3	LD　X001	11	OUT　T0	21	RST　C0
4	ORI　C0		SP　K15	23	LD　T0
5	AND　Y001	14	ANI　T0	24	OUT　C0
6	OR　M10	15	OUT　Y005		SP　K10
7	ANI　X002	16	LD　T0	27	END
8	OUT　Y001	17	OUT　T1		

3. 按钮人行道（见图 2-35）

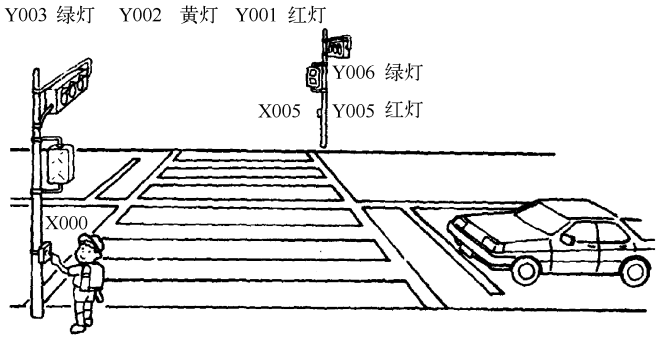


图 2-35　按钮人行道

（1）控制要求　当人行道口的按钮被按下时，交通灯按图 2-36 所示顺序变化。如果交通灯已经进入运行变化，按钮将不起作用。

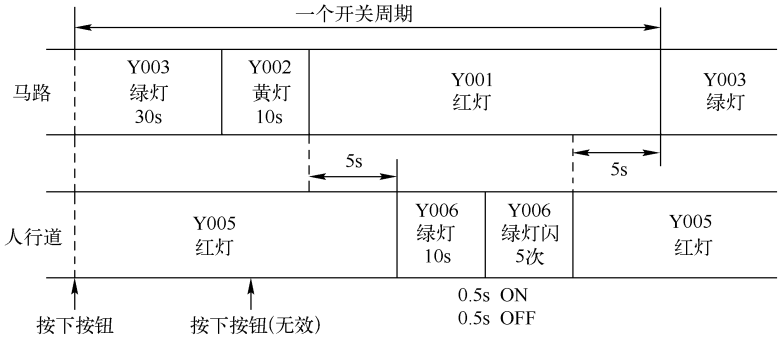


图 2-36　交通灯变化顺序

（2）本例的目的　在使用前例中同样的闪烁器和计数电路的情况下，采用时序图来设计一个更复杂的逻辑控制并编程。

（3）绘制时序图（见图 2-37）

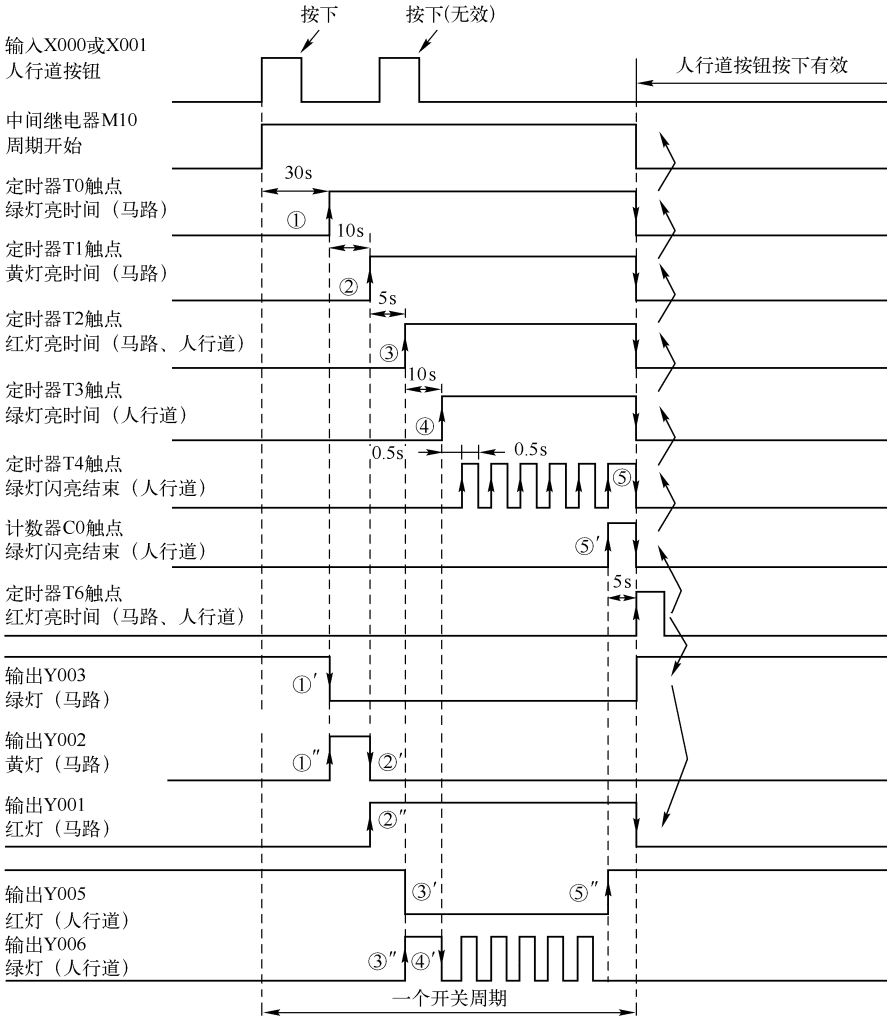


图 2-37　时序图

（4）设计逻辑电路（见图 2–38）

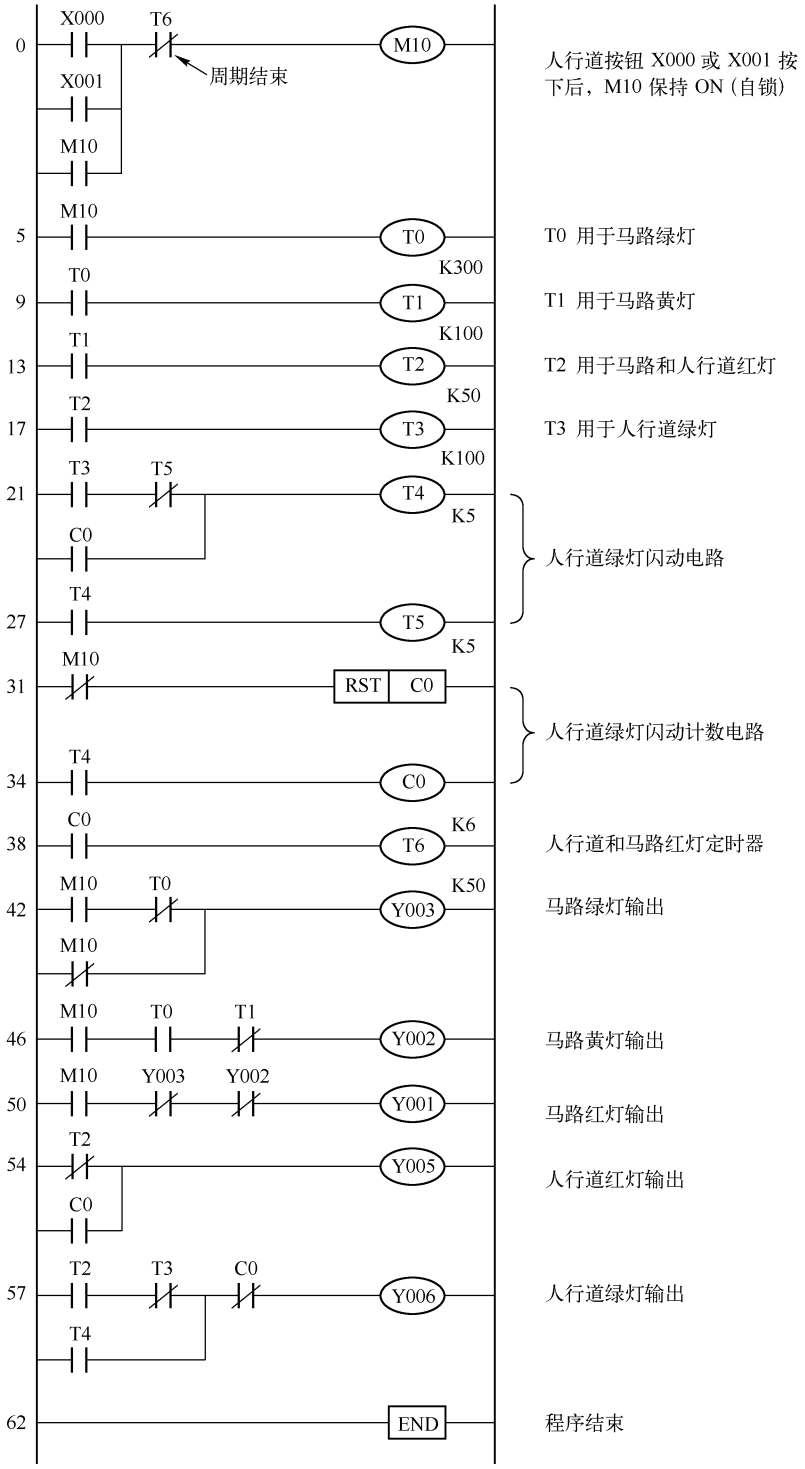


图 2–38　逻辑电路

（5）写出指令表（见表 2–18）

表 2–18　指令表

序　号	指　　令	序　号	指　　令	序　号	指　　令
0	LD　X000	23	OR　C0	47	AND　T0
1	OR　X001	24	OUT　T4 SP　K5	48	ANI　T1
2	OR　M10			49	OUT　Y002
3	ANI　T6	27	LD　T4	50	LD　M10
4	OUT　M10	28	OUT　T5 SP　K5	51	ANI　Y003
5	LD　M10			52	ANI　Y002
6	OUT　T0 SP　K300	31	LDI　M10	53	OUT　Y001
		32	RST　C0	54	LDI　T2
9	LD　T0	34	LD　T4	55	OR　C0
10	OUT　T1 SP　K100	35	OUT　C0 SP　K6	56	OUT　Y005
				57	LD　T2
13	LD　T1	38	LD　C0	58	ANI　T3
14	OUT　T2 SP　K50	39	OUT　T6 SP　K50	59	OR　T4
				60	ANI　C0
17	LD　T2	42	LD　M10	61	OUT　Y006
18	OUT　T3 SP　K100	43	ANI　T0	62	END
		44	ORI　M10		
21	LD　T3	45	OUT　Y003		
22	ANI　T5	46	LD　M10		

习题

1. 将图 2–39 写成指令表。

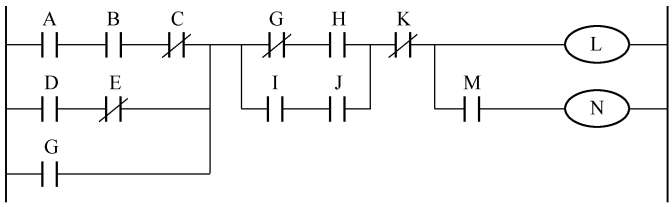


图 2–39　习题 1 的图

2. 如图 2–40 所示，若传送带上 20 s 内无产品通过则报警（Y00 为 ON）。

3. 设计三层电梯控制程序，要求写出梯形图及指令表。

规定 I/O 编号（PB□为□层呼叫按钮；LS□为□层限位开关；□F 为□层指示灯）：

输入：PB3　X00；PB2　X01；PB1　X02；LS3　X03；LS2　X04；LS1 X05

输出：上升 Y01；下降 Y00；1F Y10；2F Y09；3F Y08

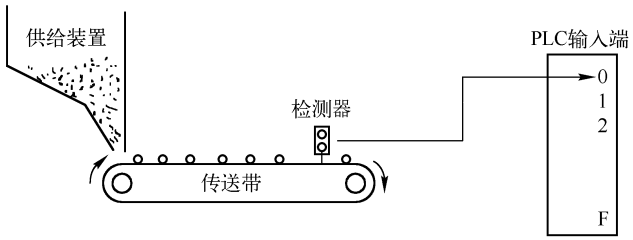


图 2-40　习题 2 的图

动作要求：

1）电梯停在 1 层或 2 层时，按 3F，则电梯上升至 LS3 停。

2）电梯停在 3 层或 2 层时，按 1F，则电梯下降至 LS1 停。

3）电梯停在 1 层时，按 2F，则电梯上升至 LS2 停。

4）电梯停在 3 层时，按 2F，则电梯下降至 LS2 停。

5）电梯停在 1 层时，而 2F、3F 均有人呼叫时，则电梯上升至 LS2 时暂停 2 s 后继续上升至 LS3 停。

6）电梯停在 3 层时，而 1F、2F 均有人呼叫时，则电梯下降至 LS2 时暂停 2 s 后继续下降至 LS1 停。

7）当电梯在上升途中，任何反方向（下降）按钮均无效。

8）当电梯在下降途中，任何反方向（上升）按钮均无效。

4. 对如下十字路口交通灯问题编制控制程序，写出梯形图及指令表。

规定　I/O 编号：

输入：启动 X00；

输出：（1）绿 Y00；（1）黄 Y01；（1）红 Y02 ；（2）绿 Y04；（2）黄 Y05；（2）红 Y06；

动作要求：两组交通灯时序图如图 2-41 所示，关断电源后再启动则从头开始。

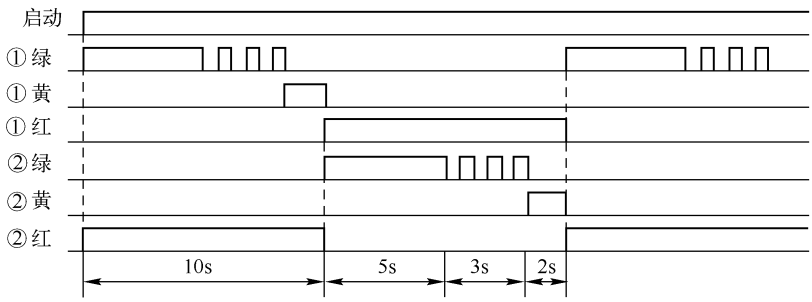
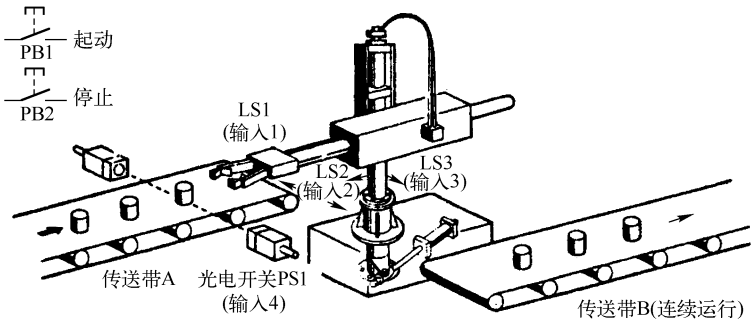
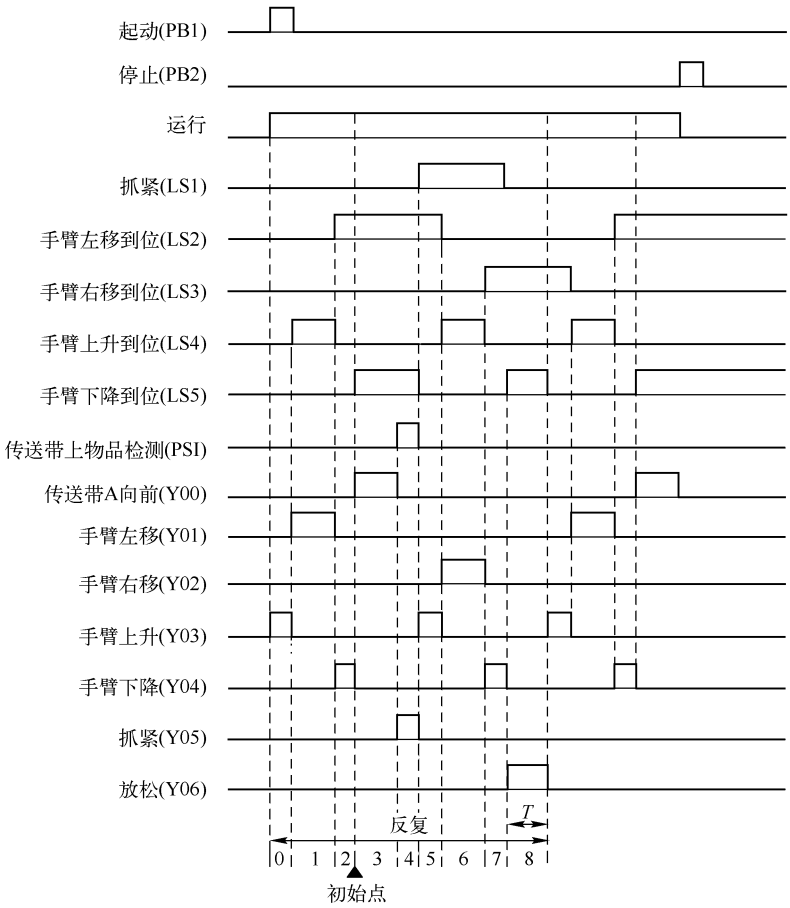


图 2-41　习题 4 的图

5. 如图 2-42a 所示，机械手将传送带 A 上的物品送到传送带 B 上，写出梯形图及指令表。输入编号：PB1 X00；PB2 X01 ；LS1 X02；LS2 X03；LS3 X04；LS4 X05；LS5 X06；PS1 X07。输出编号：传送带 A Y00；左移 Y01；右移 Y02；上升 Y03；下降 Y04；抓 Y05；放 Y06。图 2-42b 所示为时序图。



a)



b)

图 2-42　习题 5 的图

第 3 章　状态图与顺控指令

用梯形图或指令表方式编程固然广为电气技术人员所接受，但对于一个复杂的控制系统，尤其是顺序控制程序，由于内部的联锁、互动关系极其复杂，其梯形图往往长达数百行，通常要由熟练的电气工程师才能编制出这样的程序。另外，如果在梯形图上不加注释，则这种梯形图的可读性也会大大降低。

近年来，许多新生产的 PLC 在梯形图语言之外加上了符合 IEC61131—3 标准的 SFC（Sequential Function Chart）语言，用于编制复杂的顺控程序。另外，这种方法也为调试、试运行带来了许多方便。

IEC61131—3 中定义的 SFC 语言是一种通用的流程图语言。不同厂家生产的可编程序控制器中用 SFC 语言编制的程序极易相互变换。三菱公司的小型 PLC 在基本逻辑指令之外增加了两条简单的步进顺控指令（STL，意为 Step Ladder），同时辅以大量状态元件，就可以用类似于 SFC 语言的状态转移图方式编程。

本章将详细说明步进顺控指令的内容以及对实际顺序控制的处理方法。

3.1　状态转移图

3.1.1　状态的功能

要用继电器梯形图编制顺序控制程序需要有些经验，并且所编的复杂程序也难以读懂。若机械动作用所谓状态转移图（或按英文直译为顺序功能图）表示，则编程就很方便。

称为“状态”的软元件是构成状态转移图的重要元素。FX_{2N} 可编程序控制器的软元件中有 900 点状态（S0 ~ S899）可用于构成状态转移图。其中 S0 ~ S19 用作后面将会叙述的特殊目的，如 S0 ~ S9 称为初始状态，是状态转移图中的起始状态。

如图 3-1 中状态 S30 有效时输出 Y10，Y11 动作，程序等待转移条件 X20 动作。X20 瞬时接通，动作状态就从 S30 向 S31 转移。S30 转到 S31，使 Y10 OFF，Y12 ON。SET 驱动的 Y11 保持接通。

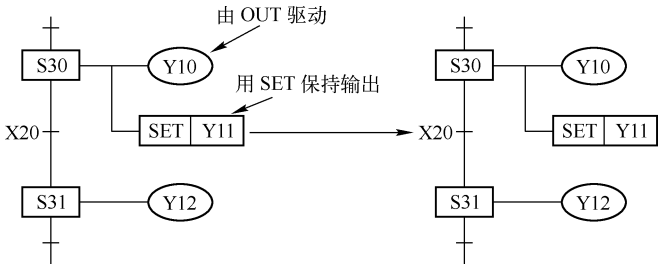


图 3-1　状态转移图

注意：①虽然通常用单独触点作为转移条件，但是实际上，X、Y、M、S、T、C 等各种元件触点的逻辑组合（复杂的串联、并联）连接时也可用作转移条件。②各种负载（Y、M、

S、T、C）和功能指令可由“状态”的触点驱动，也可由各种元件触点的逻辑组合驱动。

3.1.2 简单流程的状态转移图

图 3-2 所示机械手的工作，是将工件从 A 点向 B 点移送。以此为例说明状态转移图。

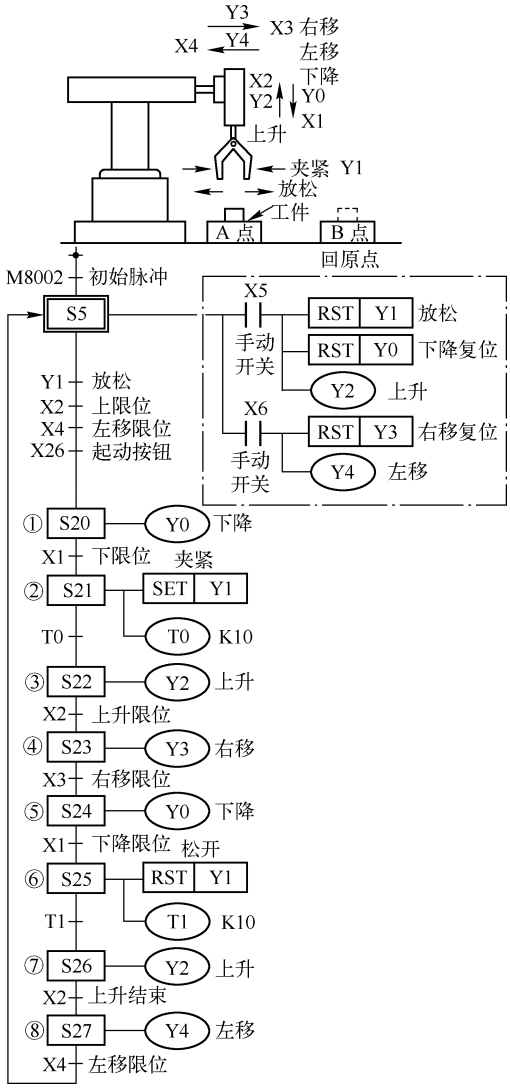


图 3-2　机械手工作系统

例中，上升/下降、在行/右行等分别使用了双螺线管的电磁阀（在某方向的驱动线圈失电时能保持在原点位置上，只有驱动反方向的线圈才能反方向运动），夹钳使用单螺线管电磁阀（只在有电时能夹紧）。

1. 手动操作

这是初次运行时将机械复归左上原点位置的程序。状态 S5 是在 PLC 从停机转为运行的瞬间，用特殊辅助继电器 M8002 置位的。

2. 半自动单循环运行

1）用手动操作将机械移至原点位置，然后按动起动按钮 X26，动作状态从 S5 向 S20 转

移，下降电磁阀的输出 Y0 动作，接着下限开关 X1 接通。

2）动作状态 S20 向 S21 转移，下降输出 Y0 切断，夹钳输出 Y1，保持接通状态。

3）1 s 后定时器 T0 动作，转至状态 S22 使上升输出 Y2 动作，不久到达上限，X2 接通，状态转移。

4）状态 S23 为右行，输出 Y3 动作，到达右限位置 X3 接通，转为 S24 状态。

5）转至状态 S24，下降输出 Y0 再次动作，到达下限位置 X1 立即接通，接着动作状态由 S24 向 S25 转移。

6）在 S25 状态，先将保持夹钳输出 Y1 复位，并启动定时器 T1。

7）夹钳输出复位 1 s 后状态转移到 S26，上升输出，Y2 动作。

8）到达上限位置 X2 接通，动作状态向 S2 转移，左行输出 Y4 动作。一旦到达左限位置 X4 接通，动作状态返回 S5，成为等待再起动的状态。

3.1.3 选择性分支与汇合

图 3-3 所示为使用传送机，将大、小球分类后分别传送的系统。

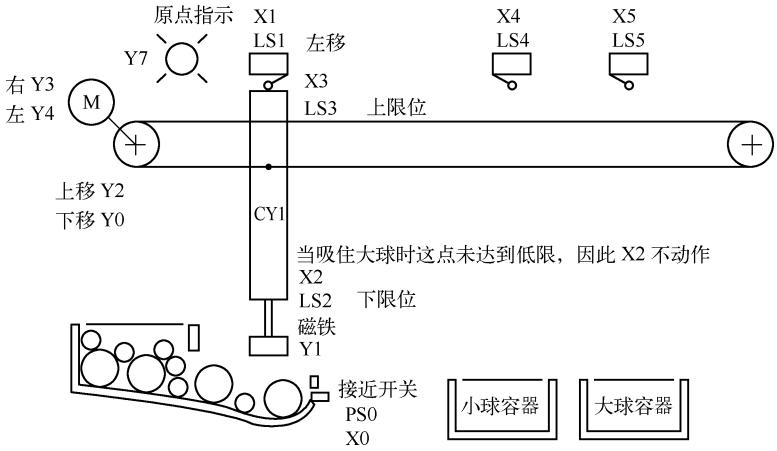


图 3-3 将大、小球分类后分别传送的系统

左上为原点，动作顺序为下降、吸收、上升、右行、下降、释放、上升、左行。另外，机械臂下降，若电磁铁吸住大球，下限开关 LS2 断开，若吸住小球，LS2 接通。

从多个分支流程中选择某一个单支流程，称之为选择性分支。

图 3-4 所示的选择性分支示例中，X0、X10、X20 在同一时刻最多只能有一个为接通状态。这一点是必要前提。比如 S20 动作时，X0 一接通，动作状态就向 S21 转移，S20 就变为“0”状态，因此以后即使 X10 或 X20 接通，S31 或 S41 也不会动作（不会为“1”状态）。汇合状态 S50，可由 S22、S32、S42 中任意一个驱动。

分类处理流程图如图 3-5 所示。

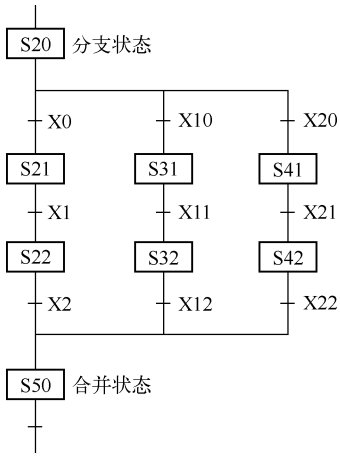


图 3-4 选择性分支示例

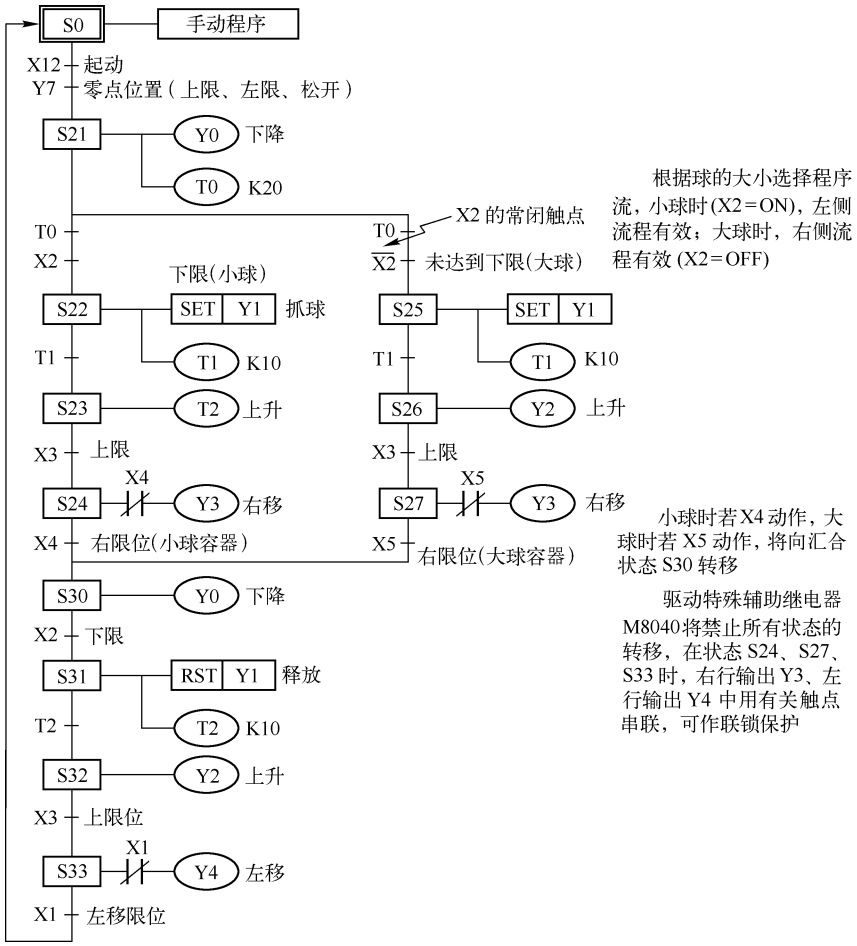


图 3-5　分类处理流程图

本例中，用手动使机械达到初始位置。后面将阐述如何自动使机械达到初始位置。

这里所举的例子是按一下起动按钮 X12，实现单个循环半自动运行的流程。自动连续重复运行方法和每一工序的步进运行方法稍后说明。

3.1.4　并行型分支/汇合

产品 A、B、C 各自并行加工，加工完成后欲将其组合起来时，有必要设计有并行分支，最后再汇合的状态转移图。

图 3-6 所示为按钮式人行横道线的例子，用并行型分支/汇合的方式实现。

并行分支是指同时处理的程序流程。

图 3-7 所示为在 S20 动作时，X0 一接通，S21、S24、S27 就同时动作，各分支流程开始动作。待各流程的动作全部结束时，X7 接通，然后汇合状态 S30 动作，S23、S26、S29 全部恢复为“0”状态。

这种汇合有时被称为排队汇合。

图 3-8 所示为按钮式人行横道线的状态转移图，其中人行横道线的绿灯闪烁部分为重复动作的流程。

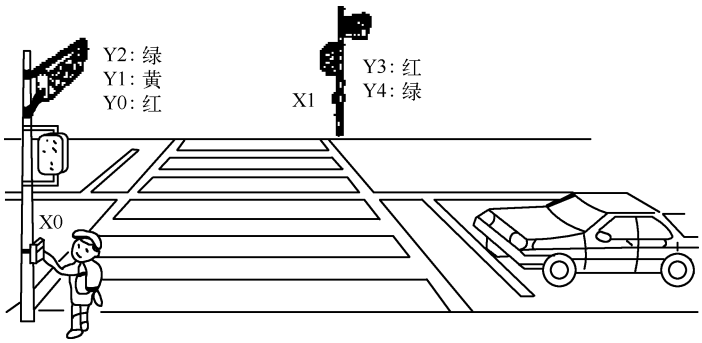


图 3-6　按钮式人行横道线

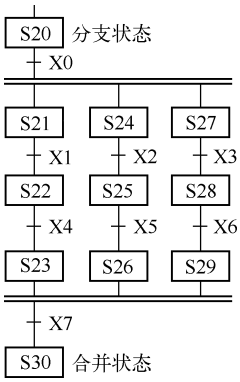
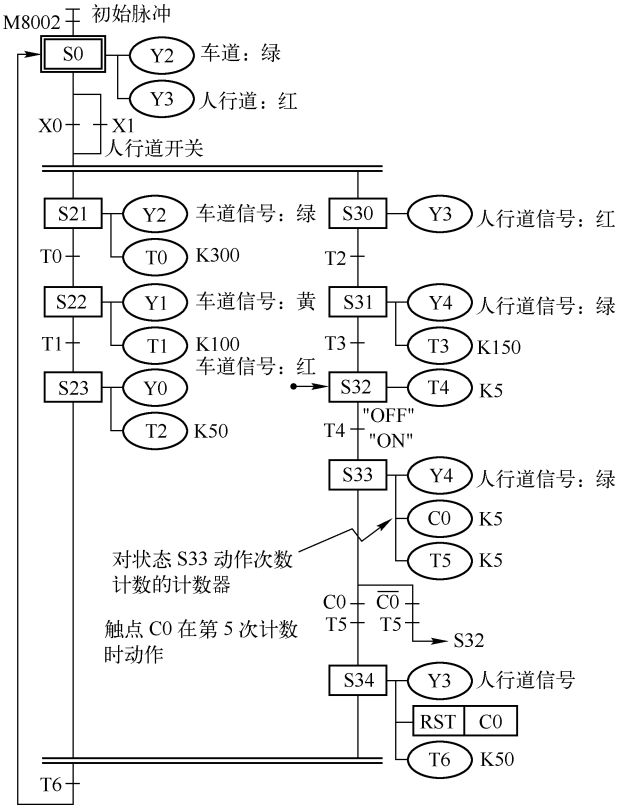


图3-7　并行分支示例



PLC 在由停机转入运行时，初始状态 S0 动作，通常为车道 = 绿，人行道 = 红（通过 M8002）

若按人行横道按钮 X0 或 X1，则状态 S21 为车道 = 绿。S30 为人行道 = 红，红绿灯状态不变化

30s 后车道 = 黄，再过 10s，车道 = 红以后定时器 T2（5s）启动。5s 后 T2 触点接通，人行道 = 绿

15s 后人行道绿灯开始闪烁（S32= 灭，S33= 亮）

闪烁中 S32、S33 的动作反复进行，计数器 C0（设定为 5 次）触点一接通，状态向 S34 转移，人行道 = 红，5s 后返回初始状态

在状态转移过程中，即使按动人行横道按钮 X0，X1 也无效

图 3-8　按钮式人行横道线的状态转移图

3.2　编程方法

3.2.1　步进顺控的程序示例

从状态转移图中有代表性地抽出一个状态，如图 3-9a 所示，每个状态具有驱动负载、指定转移条件以及指定转移目标三个功能。

步进顺控图（见图 3-9b）是用继电器顺控电路表达的状态转移图。

程序用状态转移图或用步进顺控图表达都可以运行。编程顺序为先进行负载的驱动处理，接着进行转移处理。当然没有负载的状态不必进行负载驱动处理。

图 3-9c 所示为程序指令表。

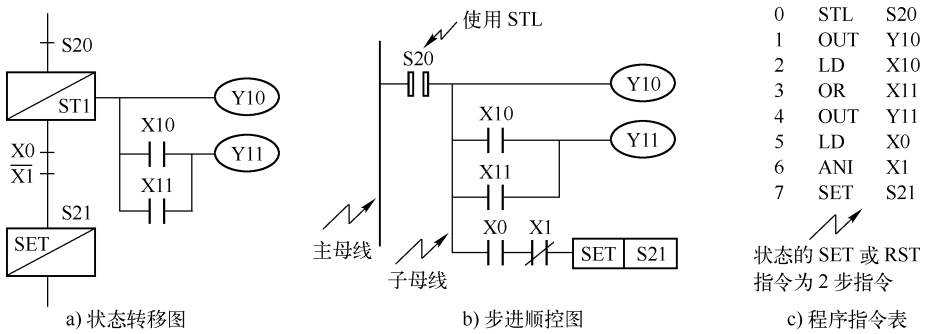


图 3-9　步进顺控示例

STL 指令为与主母线连接的常开触点指令，接着就可以在副母线里直接连接线圈或通过触点驱动线圈。连接在副母线上的触点使用 LD、LDI 指令。若要返回原来的主母线时，使用 RET（返回）指令。但是，通过 STL 触点驱动状态 S，在该 S 前面的那个状态就自动复位。

状态转移图和步进顺控图表达的都是同一个程序，它的优点是可以让编程者每次只考虑一个状态，而不用考虑其他状态，使编程更容易。另外，状态的顺序可自由选择，不一定按 S 编号的顺序选用。但是在一系列 STL 指令的最后，必须写入 SET 指令。

3.2.2　初始状态编程

初始状态编程的例子如图 3-10 所示。

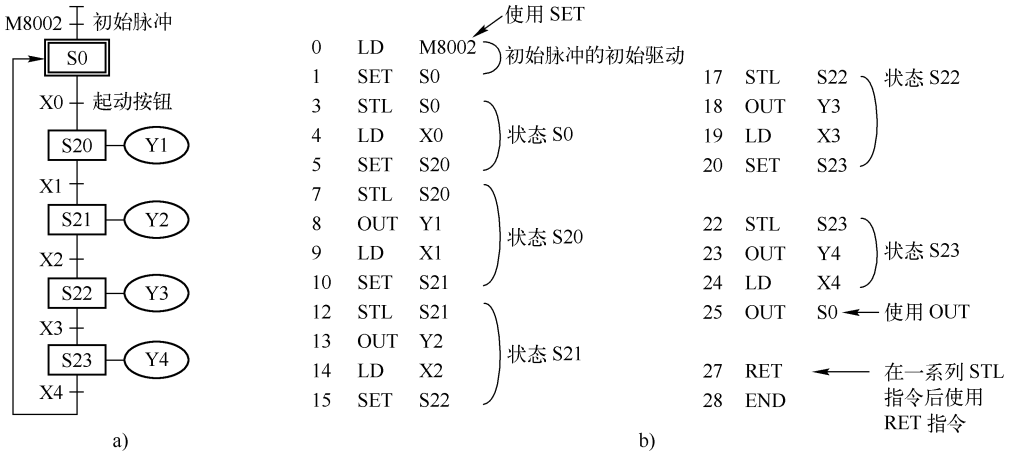


图 3-10　初始状态编程示例

在状态转移图起始位置的状态即是初始状态，S0 ~ S9 可用作初始状态。

初始状态也可由其他状态元件（本图中为 S23）驱动。最开始运行时，初始状态则必须用其他方法预先驱动，使之处于工作状态（即 S23 先置“1”）。

图 3-10 中，初始状态最初是由 PLC 从 STOP 到 RUN 切换瞬时动作的特殊辅助继电器 M8002 驱动，使 S0 置“1”。

除初始状态之外的一般状态元件必须在其他状态后加入 STL 指令才能驱动，不能脱离状态而用其他方式驱动。

编程时必须将初始状态编在其他状态之前。

注意：

状态元件 S500 ~ S899 是由银电池后备的，在运行中途发生停电，再通电时要继续运行的场合，请使用这些状态元件。

在这种场合应重新考虑初始化的方式。驱动初始状态（S0 ~ S20）的较好方法见 3.2.3 节。

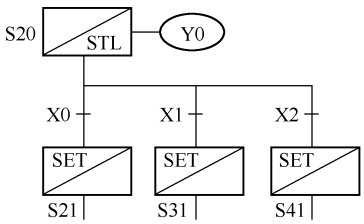
如图 3-10 所示，程序中在一系列 STL 指令的最后必须有 RET 指令。

如果没有 RET 指令，周期性操作的 PLC 将第 0 步的 LD 指令看成为与 STL S23 的副母线连接，引起误动作（实际上造成程序出错，不能运行）。

3.2.3 选择性分支、汇合的编程

1. 选择性分支举例（见图 3-11）

与对一般状态的编程一样，先进行驱动处理，然后设置转移条件，编程时要由左至右逐个编程。



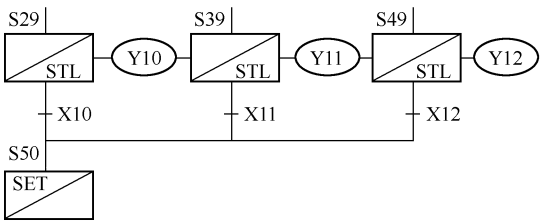

```
STL   S20
OUT   Y0   ←— 驱动处理
LD    X0
SET   S21  ←— 转移到第一分支状态
LD    X1
SET   S31  ←— 转移到第二分支状态
LD    X2
SET   S41  ←— 转移到第三分支状态
```

图 3-11　选择性分支示例

2. 选择汇合举例（见图 3-12）

先进行汇合前状态的输出处理，然后朝汇合状态转移，此后由左至右进行汇合转移。这是为了自动生成 SFC 画面而追加的规则。

注意：分支、汇合的转移处理程序中，不能用 MPS、MRD、MPP、ANB、ORB 指令。



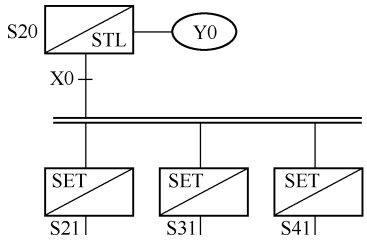

```
STL   S29
OUT   Y10    输出处理
  ⋮
STL   S39
OUT   Y11    输出处理
  ⋮
STL   S49
OUT   Y12    输出处理

STL   S29
LD    X10    从第一分支转
SET   S50    移到汇合点
STL   S39
LD    X11    从第二分支转
SET   S50    移到汇合点
STL   S49
LD    X12    从第三分支转
SET   S50    移到汇合点
```

图 3-12　选择汇合示例

3.2.4 并行分支/汇合的编程

1. 并行分支举例（见图 3-13）

与对一般状态编程一样，先进行驱动处理，然后进行转移处理。转移的处理要从左到右依次进行。

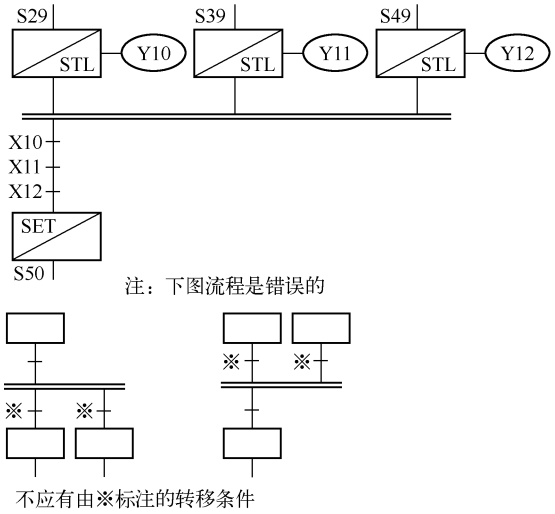


STL	S20	
OUT	Y0	← 驱动处理
LD	X0	
SET	S21	← 转移到第一并行分支状态
SET	S31	← 转移到第二并行分支状态
SET	S41	← 转移到第三并行分支状态

图 3-13　并行分支示例

2. 并行汇合举例（见图 3-14）

汇合前先对各状态的输出处理分别编程，然后从左到右进行汇合处理。



⋮		
STL	S29	
OUT	Y10	← 输出处理
⋮		
STL	S39	
OUT	Y11	← 输出处理
⋮		
STL	S49	
OUT	Y12	← 输出处理
⋮		
STL	S29	
STL	S39	
STL	S49	
LD	X10	合并转移处理
AND	X11	
AND	X12	
SET	S50	
⋮		

连续写 STL 指令表示并行汇合，STL 指令最多可连续使用 8 次

图 3-14　并行汇合示例

3.2.5 分支、汇合的组合

图 3-15 所示为分支、汇合的组合示例。

如果程序连续直接从汇合点转移到下一次分支，而没有中间状态的话，请如图 3-15 所示加入一个中间状态。S100 和 S103 的转移触点可以省略。

图 3-16 中分支条件※和汇合条件※是错误的，参见 3.1 节。

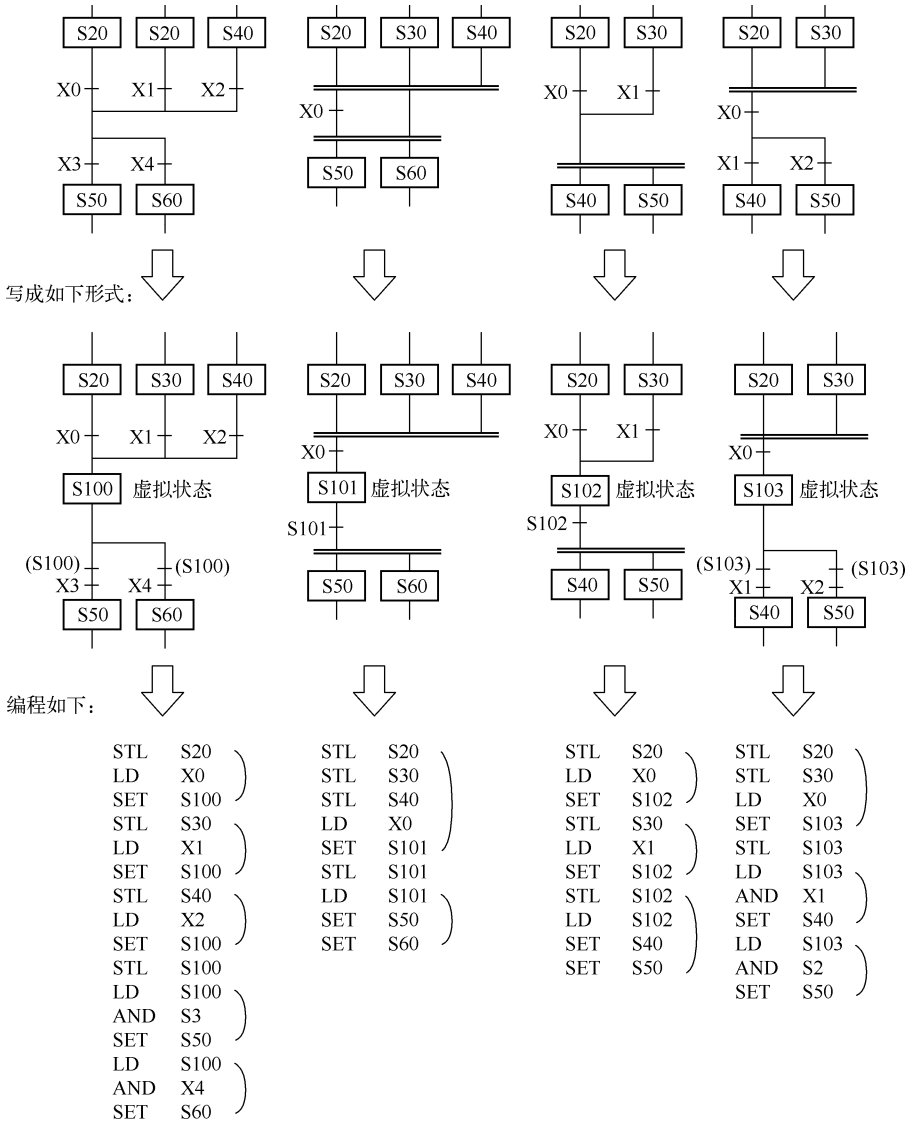


图 3-15　分支、汇合的组合示例

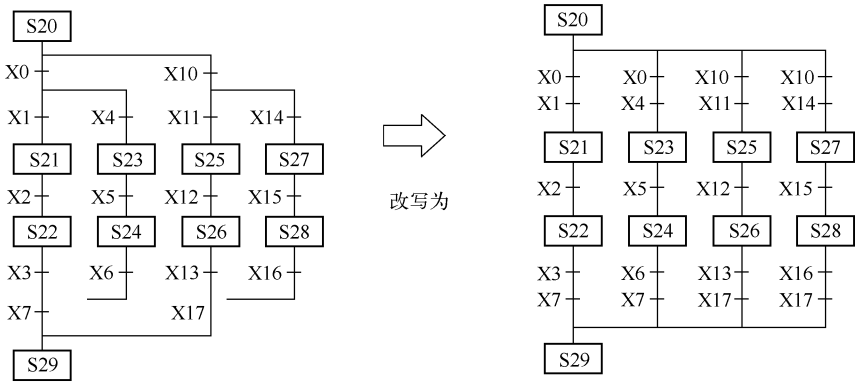
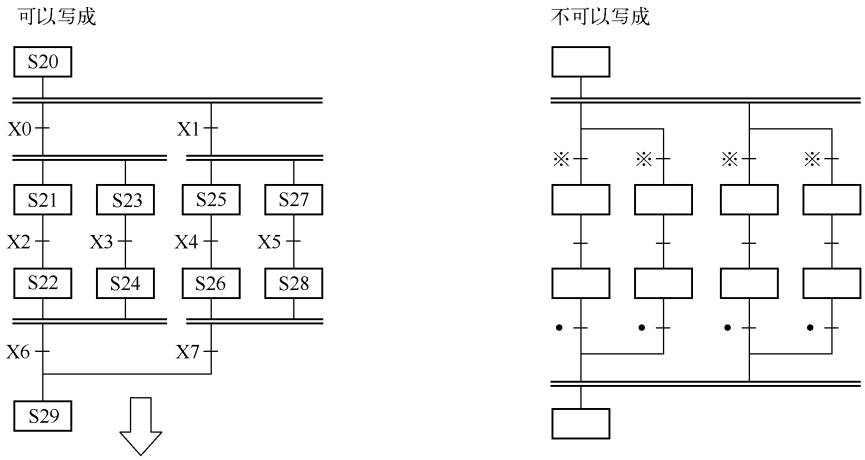


图 3-16　分支、汇合的组合示例



写成程序

分支程序		汇合程序	
STL	S20	STL	S22
LD	X0	STL	S24
SET	S21	LD	X6
SET	S23	SET	S29
LD	X1	STL	S26
SET	S25	STL	S28
SET	S27	LD	X7
		SET	S29

图 3-16　分支、汇合的组合示例（续）

3.2.6　跳转处理、复位处理

跳转处理、复位处理应用示例如图 3-17 所示。

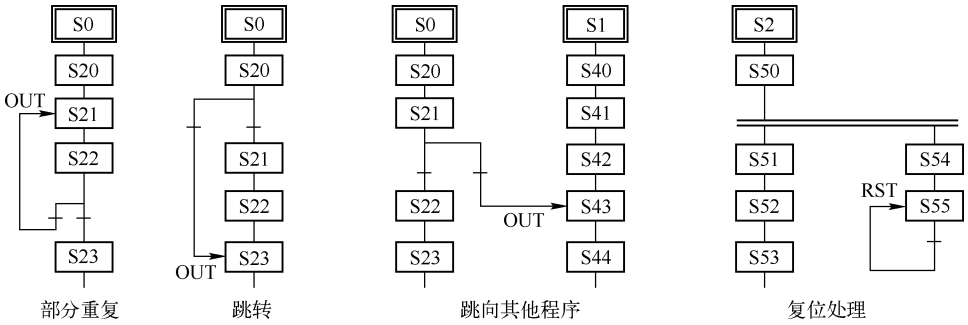


图 3-17　跳转处理、复位处理应用示例

除分支及合流（如前节所述）以外，状态还可向较远的状态转移（非连续的状态），如图 3-18 所示。

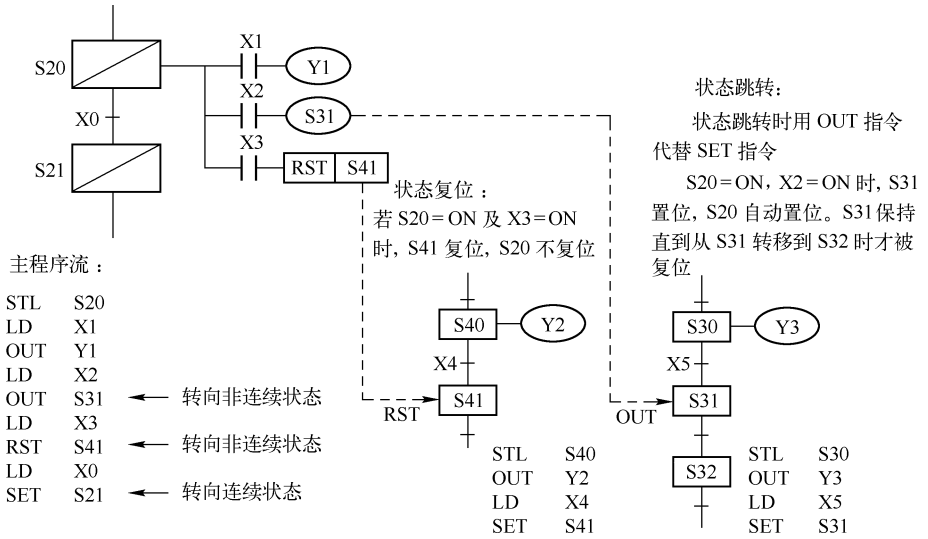


图 3-18　状态跳转示例

1. 分支数的限制

对所有的初始状态（S0～S9），每一状态下的分支电路数总和不能大于 16 个，并且在每一分支点分支数不能大于 8 个，如图 3-19 所示。

注意：直接从汇合线或汇合前状态向其他远处状态的跳转处理或复位处理是不允许的，此时，必须设定虚拟状态，以执行上述状态转移（远距离跳转或复位）。

2. 分离程序流

具有多个初始状态的状态转移图的程序，要按各初始状态分开编程。

如图 3-20 所示，从属于初始状态 S3 的状态 S20～S39 的 STL 指令程序全部结束之后，再进行与下一初始状态 S4 有关的程序编制。在这两部分分离的程序流中，用 OUT 指令代替 SET 指令可实现相互间的跳转。

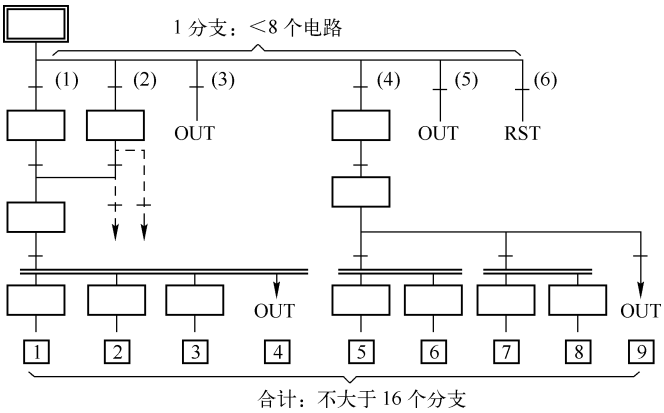


图 3-19　分支数的限制

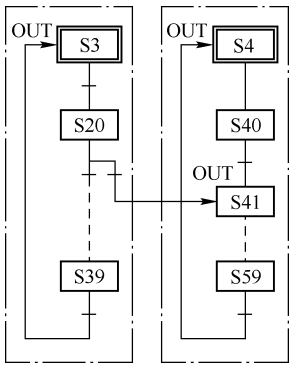


图 3-20　分离程序流

3.2.7　同一信号作为多个状态之间转移条件的处理方法

在某些应用中，流程中各个状态之间的转移条件是同一信号。原本的意思是当这个信号

来时流程上走一步，信号再来时再走一步。但若编程时写成如图 3-21 所示的例子，则当 M0 号来时整个流程会“走通”，即一次通过全部状态。对这种情况可采用以下两种方法处理：

方法 1：　在每个状态中设置一个阻挡元件，以防止“走通”现象。如图 3-22 所示，进入 S30 时，M1 脉冲阻止进一步转移；M0 下一个脉冲到来时，阻挡脉冲消失，可顺利向下转移。每个状态中都设一个阻挡元件，保证 M0 来一个脉冲向下走一步。

方法 2：　利用脉冲触点指令（LDP、LDF、ANP 等）与 M2800 ~ M3071 辅助继电器配合可得到方法 1 同样的结果，如图 3-23 所示。

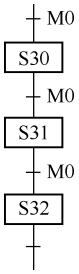


图 3-21　流程“走通”

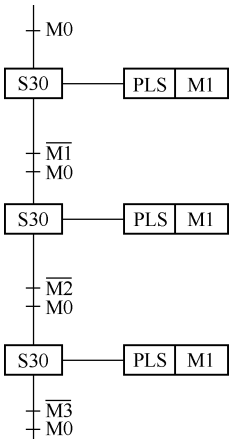


图 3-22　防止“走通”方法 1

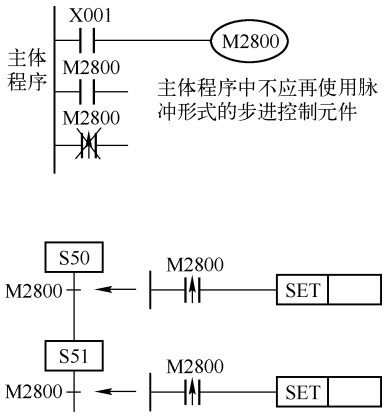


图 3-23　防止“走通”方法 2

3.3　状态的详细动作

3.3.1　STL 指令的动作

当 STL 触点接通（ON），与此连接的电路就运行。

当 STL 触点断开（OFF），与此连接的电路停止运行，在其负载复位后一个扫描周期，这部分线路的指令就被跳过，不再执行。

如果不同的状态驱动同一个输出，如图 3-24a 中的输出 Y2、S21 或 S22 任一个接通，Y2 动作；S21 和 S22 都断开，则 Y2 不动作。若用这种 STL 电路，可使用双线圈，但在 STL 电路中，与其他电路中共用一个线圈（即所谓双线圈），必须多加注意。

状态元件是绝对不能重复使用的。

状态转移过程中，在一个扫描周期中两状态同时为 ON 的情况也可能出现。因此，如图 3-24b 所示不能同时为 ON 的一对输出之间（同一电动机的正转与反转）必须加上联锁，防止同时为 ON。

相邻的状态不能重复使用同一个定时器，因为指令会互相影响，使定时器无法复位。对于分隔的两个状态（图 3-24c 中的 S40 和 S42）可以使用同一定时器（T1）。

用同一信号顺次作为转移条件的场合，要用脉冲，如图 3-24d 所示，当 M0 为通时，S50 刚动作，M1 立即断开，避免状态马上直接转移到 S51。到下一个 M0 脉冲到来时才能转移到 S51。

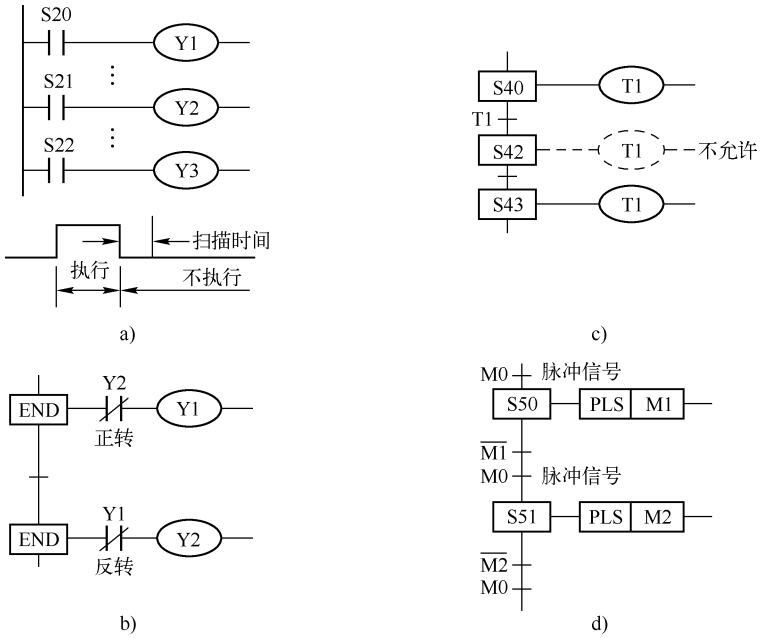


图 3-24　STL 指令的动作示例

3.3.2　对状态的各种指令的处理

STL 指令仅对状态元件 S 有效；对状态元件 S，还可以用 LD/LDI、AND/ANI、OR/ORI、OUT、SET/RST 等触点和线圈指令。

图 3-25 中，即使 S20 驱动 S30 或 S21 后，S20 也不复位；另外，如果 S20 断开（OFF），S30 即停止动作，这是因为 S21 和 S30 没有通过 STL 指令而直接由 S20 驱动。

程序中没用过 STL 指令时，状态 S 可作为通用辅助继电器。

对于 STL 指令后的状态 S，OUT 指令与 SET 指令具有同样的功能，即都能使转移源自动复位，另外还具有停电保持功能。但是，OUT 指令在状态转移图中，只用于向分离的状态转移，而不是向相邻的状态转移，如图 3-26 所示。

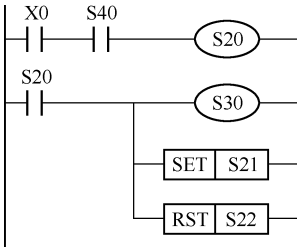


图 3-25　不通过 STL 指令驱动状态元件的示例

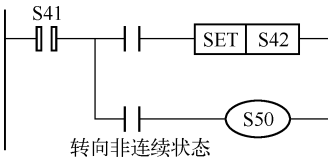


图 3-26　转向非连续状态的示例

另外，STL 触点后，不能紧接着使用 MPS 指令。

从 STL 指令开始到下一个 STL 指令或到 RET 指令为止之间，可使用的顺控指令和不能使用的顺控指令见表 3-1。

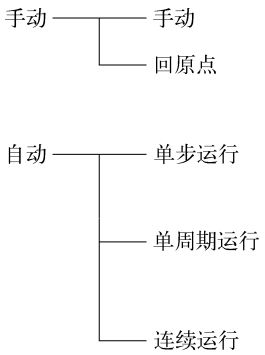
表 3-1　顺控指令

状态		LD/LDI OUT，NOP AND/ANI　SET/RST OR/ORI　PLS/PLF	ANB/ORB MPS/MRD/MPP	MC/MCR
初始状态 一般状态		可用	可用	不可用
分支/汇合 状态	输出处理	可用	可用	不可用
	转移处理	可用	不可用	不可用

后面将述及的中断程序和子程序中不能使用 STL 指令。STL 指令内虽然不禁止使用跳转指令（CJ00），但由于使用后会使动作复杂化，建议不要使用。

3.4　操作方式

3.4.1　操作方式的概念



设备的操作方式大致分为手动方式和自动方式，见右图。它们又可再分为其他运行方式，图 3-27 所示的操作面板就是据此考虑的。其运行及紧急停止的按钮与 PLC 程序无关。用这两个按钮使 PLC 外部负载的电源接通（ON）或断开（OFF）。

用各自的按钮使各个负载单独接通或断开的方式。

该方式下按动回原点按钮时，机械自动向原点回归。

按动一次起动按钮，前进一个工步（或工序）。在原点位置按动起动按钮，自动运行一遍再在原点停止。若在中途按动停止按钮就停止运行；再按起动按钮，从断点处开始继续运行，回到原点自动停止。

在原点位置按动起动按钮，开始连续地反复运行。若中途按动停止按钮，动作将继续到原点为止才停止。

另外，对于单步运行，可不必对各负载都设置按钮，因为可以使用编程器进行强制 ON/OFF。

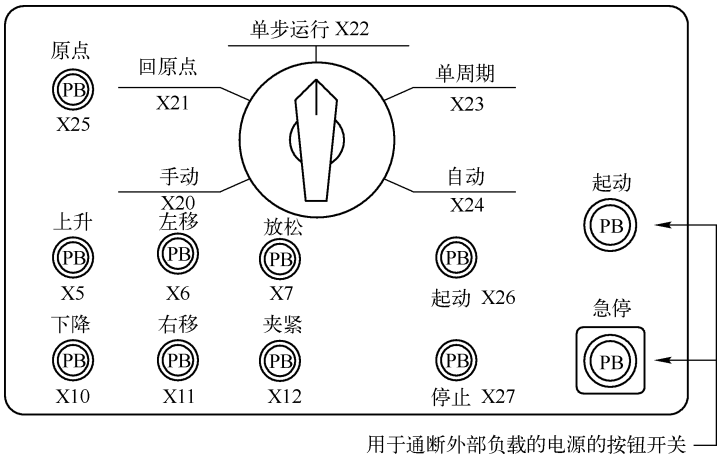


图 3-27　操作面板

若使用后面（4.8 节）将要叙述的功能指令 FNC 60（IST），能自动设定与各个运行方式相对应的初始状态。

为了使用这个功能指令，必须指定如下具有连续编号的输入点。如果无法指定连续编号，则要使用辅助继电器（M），重新安排输入编号，在设置 FNC 60（IST）时（M）作为首输入元件号。

X20：手动	X23：单周期运行	X26：自动起动
X21：回归原点	X24：连续运行	X27：停止
X22：单步运行	X25：回原点起动	

为了使 X20～X24 不会同时接通，可使用选择开关。

重新安排输入编号示例如图 3-28 所示。

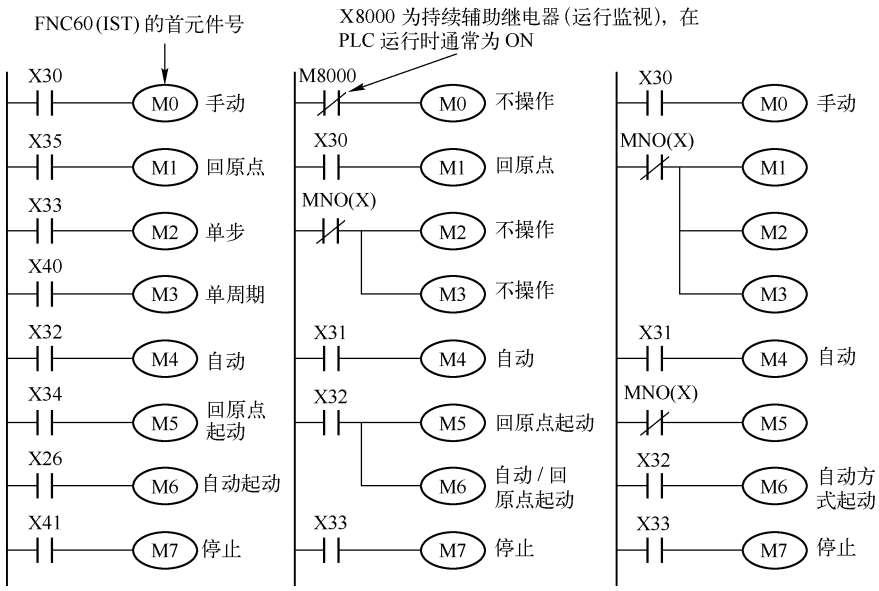


图 3-28　重新安排输入编号示例

用非连续输入	只用自动及回原点方式时	只用自动及手动方式时
X30：手动	X30：回原点	X30：手动
X35：回原点	X31：自动	X31：自动
X33：单步执行	X32：自动/回原点起动	X32：自动方式起动
X40：单周期执行	X33：停止	X33；停止
X32：自动运行		
X34：回原点起动		
X26：自动起动		
X41：停止		

3.4.2　初始状态指令

如图 3-29 所示，驱动功能指令 FNC 60（IST），下面的初始状态及相应特殊辅助继电器就自动被指定为如下功能：

LD	M8000		S0：手动初始状态
FNC	60	1 步	S1：回原点初始状态
X20			S2：自动运行初始状态
S20		各 2 步	M8040：禁止转移
S29			M8041：开始转移
			M8042：起动脉冲
			M8047：STL 监控有效

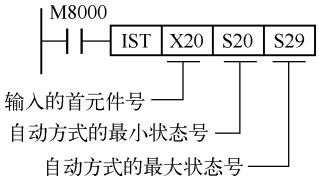


图 3-29　驱动功能指令示例

初始状态的动作：由 IST 指令自动指定的初始状态 S0 ~ S2，根据运行方式的切换，按照图 3-30 所示的形式进行。

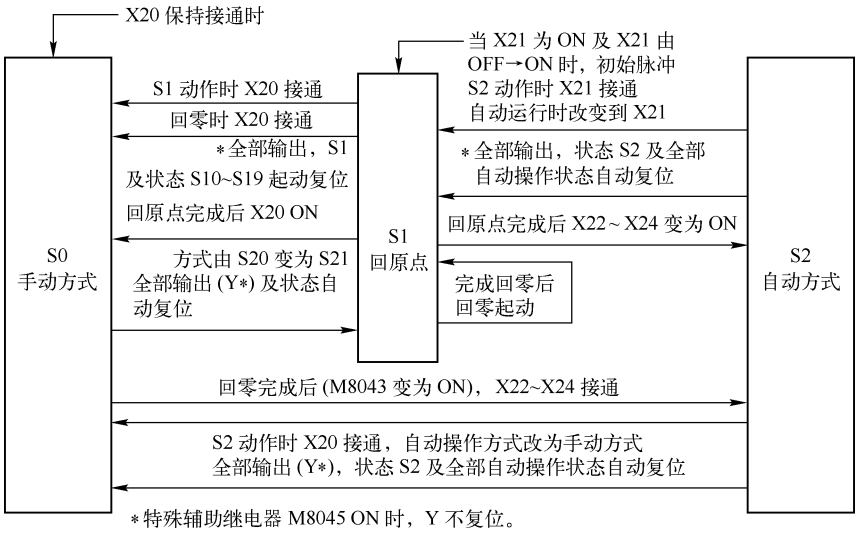


图 3-30　S0 ~ S2 运行方式的切换

根据 IST 指令自动动作的特殊辅助继电器 M8040 ~ M8042。M8047 的动作内容等效于图 3-31 所示的电路。

M8047 为 ON 时，状态 S0 ~ S899 中正在动作的状态号从最低号开始顺序存入特殊数据寄存器 D8040 ~ D8047，最多可存 8 个状态号。

禁止转移（M8040）：这个辅助继电器接通后就禁止所有的状态转移：

① 手动状态下：M8040 总是接通。

② 回原点、单周期运行：按动停止按钮后一直到再按起动按钮期间 M8040 一直保持为 1。

③ 单步执行：M8040 常通，但是在按动起动按钮时变为 OFF，使状态可以从 M8040 转移一步。

④ 其他：PLC 在 STOP→RUN 切换时 M8040 保持接通，按动起动按钮后 M8040 断开。

转移开始（M8041）：从初始状态 S2 向另一状态转移的转移条件辅助继电器：

① 手动回原点：不动作。

② 步进、单周期：仅在按动起动按钮时动作。

③ 自动：按起动按钮后保持为 ON，按停止按钮后 OFF。

④ 起动脉冲继电器（M8042）在起动按钮按下的瞬时接通。

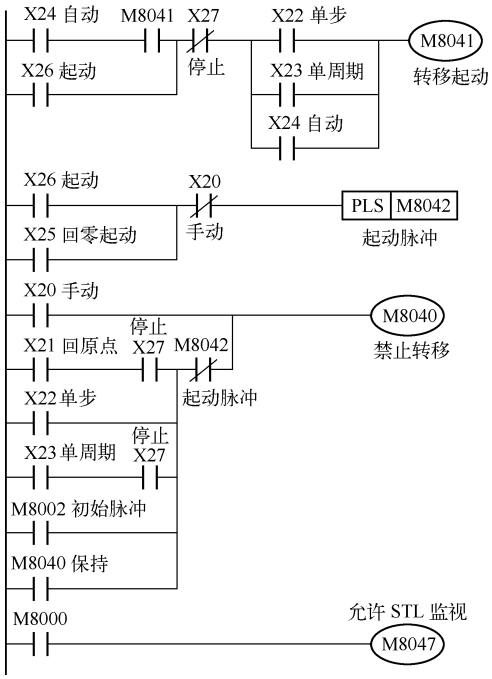


图 3-31　特殊辅助继电器的动作内容等效图

3.4.3　工件移送机构举例

图 3-32 所示为用机械手移送工件的机械系统。左上作为原点，工件按下降、夹紧、上升、右移、下降、放松、上升、左移的次序依次运行。下降/上升、左移/右移中使用双螺线

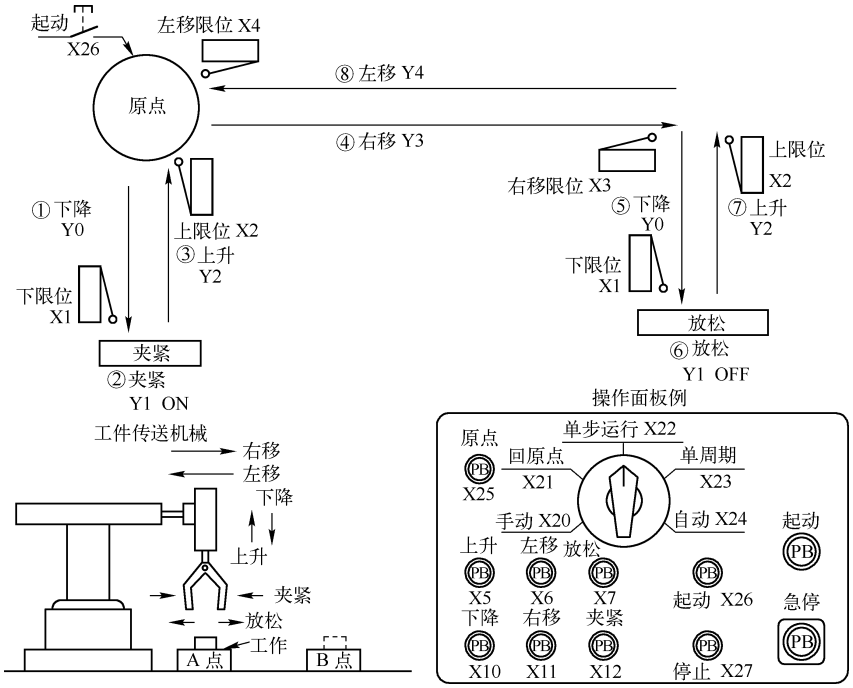


图 3-32　用机械手移送工件的机械系统

管的电磁阀，夹紧使用的是单螺线管的电磁阀。图 3-33 所示为状态转移图。

特殊辅助继电器：

M8044（原点位置条件）：这个元件由原点的各传感器驱动，它的 ON 状态作为自动方式时的允许状态转移的条件。

M8043（回原点完成）：用状态 S10 ~ S19 作回零操作，在最后状态中在自我复位前先将 M8043 置 1。

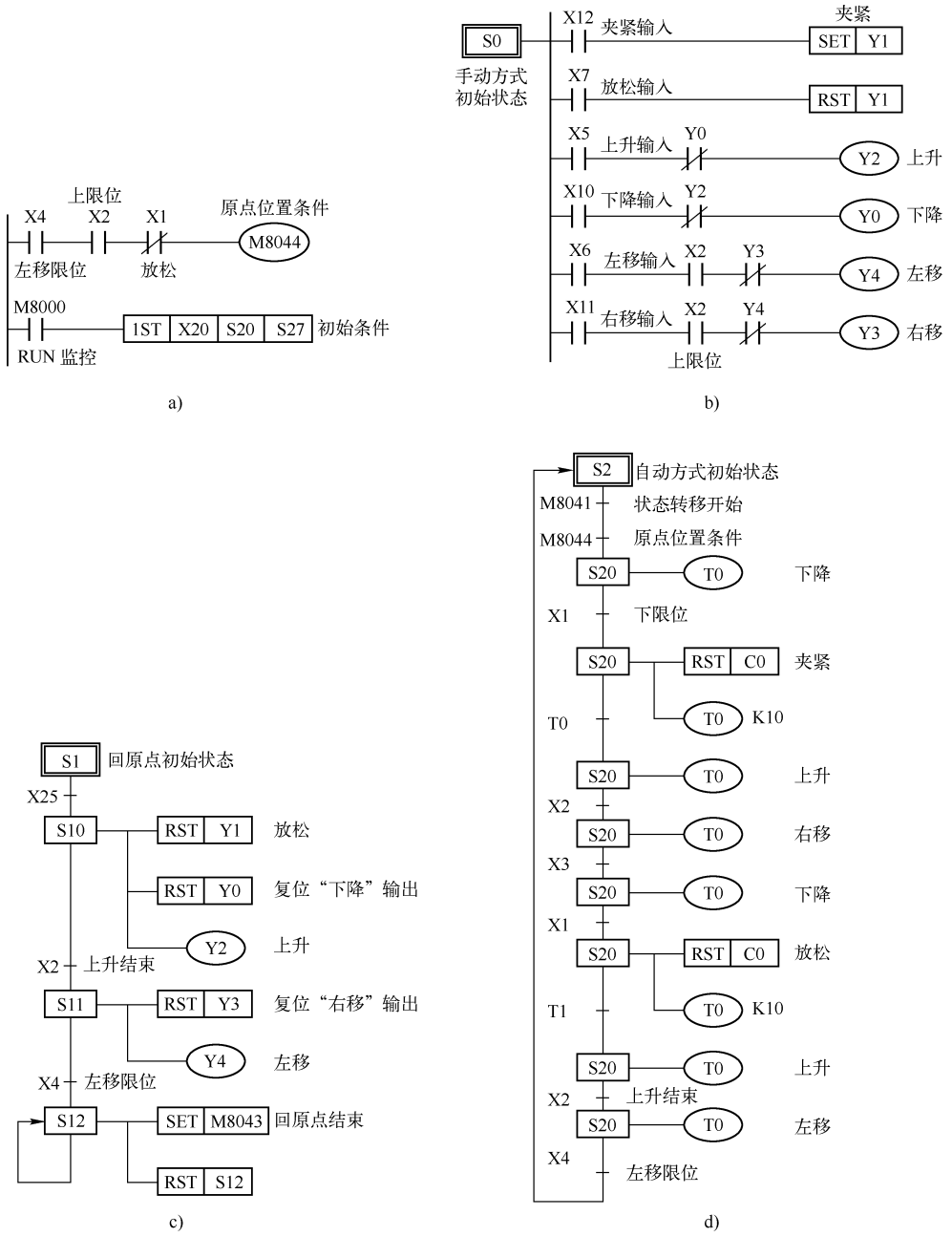


图 3-33　状态转移图

下面是图 3–33 所示状态转移图对应的程序。

0	LD	X4
1	AND	X2
2	ANI	Y1
3	OUT	M8044
5	LD	M8000
6	FNC	60
		X20
		S20
		S27
13	STL	S0
14	LD	X12
15	SET	Y1
16	LD	X7
17	RST	Y1
18	LD	X5
19	ANI	Y0
20	OUT	Y2
21	LD	X10
22	ANI	Y2
23	OUT	Y0
24	LD	X6
25	AND	X2
26	ANI	Y3
27	OUT	Y4
28	LD	X11
29	AND	X2
30	ANI	Y4
31	OUT	Y3
	(RET)	
32	STL	S1
33	LD	X25
34	SET	S10
36	STL	S10
37	RST	Y1
38	RST	Y0
39	OUT	Y2
40	LD	X2
41	SET	S11
43	STL	S11
44	RST	Y3
45	OUT	Y4
46	LD	X4
47	SET	S12
49	STL	S12
50	SET	M8043
52	RST	S12
	(RET)	
54	STL	S2
55	LD	M8041
56	AND	M8044
57	SET	S20
59	STL	S20
60	OUT	Y0
61	LD	X1
62	SET	S21
64	STL	S21
65	SET	Y1
66	OUT	T0
		K10
69	LD	T0
70	SET	S22
72	STL	S22
73	OUT	Y2
74	LD	X2
75	SET	S23
77	STL	S23
78	OUT	Y3
79	LD	X3
80	SET	S24
82	STL	S24
83	OUT	Y0
84	LD	X1
85	SET	S25
87	STL	S25
88	RST	Y1
89	OUT	T1
		K10
92	LD	T1
93	SET	S26
95	STL	S26
96	OUT	Y2
97	LD	X2
98	SET	S27
100	STL	S27
101	OUT	Y4
102	LD	X4
103	OUT	S2
105	RET	
106	END	

注：（）中的指令不是必需的。

习题

1. 按第 2 章习题 4 的要求用步进顺控指令编程。
2. 按第 2 章习题 5 的要求用步进顺控指令编程。

第 4 章　功 能 指 令

由于 PLC 是由取代继电器开始产生、发展的，且早期的 PLC 绝大部分用于顺序控制，于是许多人习惯于把 PLC 看作是继电器、定时器、计数器的集合，把 PLC 的作用局限地等同于继电控制系统、顺控器等。其实，PLC 就是工业控制计算机。PLC 系统具有一切计算机控制系统的功能，大型的 PLC 系统就是当代最先进的计算机控制系统。

小型的 PLC 由于运算速度及存储容量的限制，功能自然稍弱。但为了使 PLC 在其基本逻辑功能、顺序步进功能之外具有更进一步的特殊功能，以尽可能多地满足 PLC 用户的特殊要求，从 20 世纪 80 年代开始，PLC 制造商就逐步地在小型 PLC 中加入一些功能指令（Functional Instruction）或称为应用指令（Applied instruction）。这些功能指令实际上就是一个个功能不同的子程序。随着芯片技术的进步，小型 PLC 的运算速度、存储量不断增加，其功能指令的功能也越来越强。许多技术人员梦寐以求甚至以前不敢想象的功能，通过功能指令就成为极容易实现的现实，从而大大提高了 PLC 的实用价值。

一般来说，功能指令可以分为以下几类：程序流控制、传送与比较、四则与逻辑运算、移位与循环移位、数据处理、高速处理、方便命令、外部 I/O 处理、外部设备、实数处理、点位控制、实时时钟处理、外部设备和触点比较。

功能指令采用计算机通用的助记符 + 操作数（元件）方式，稍有计算机及 PLC 知识的人极易明白其功能。例如

这段程序意为：当执行条件 M100 为 ON 时，把源常数 K123 送到目标元件 D500。

4.1　功能指令通则

4.1.1　功能指令的表现形式

功能指令按功能号（FNC00 ~ FNC250）编排。每条功能指令都有一助记符。如图 4-1 所示，FNC 45 助记符为“MEAN”，在编程时用“HELP”，可显示功能号与对应助记符的清单。在读出程序时，功能号与助记符同时显示。

某些功能指令只需指定功能号即可，但许多功能指令在指定功能号的同时还必须指定操作元件。操作元件由 1 ~ 4 个操作数组成。下面将操作数说明如下：

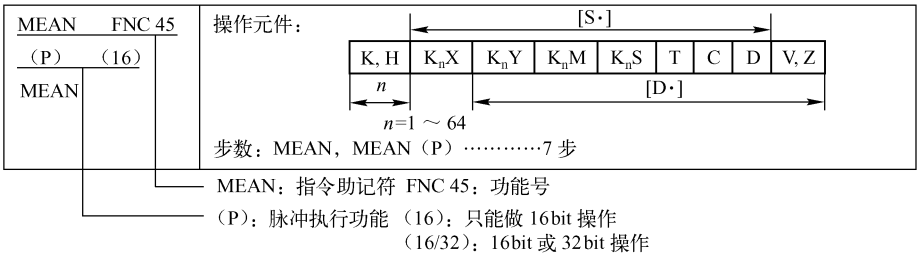
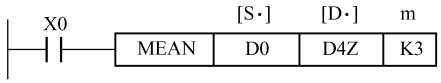


图 4-1　功能指令的表现形式

图 4－2 所示是取平均值的功能指令 $\frac{D0+D1+D2}{3}\rightarrow D4Z$。图中［S・］指取值首元件。$n$ 指定取值个数。［D・］指定计算结果存放地址。



0	LD	X0
1	MEAN	45
		D0
		D4Z
		K3
8	…	

图 4－2　取平均值指令示例

［S］：（SOURSE）源操作数。若可使用变址功能时，表示为［S・］。有时源不止一个，可用［S1・］、［S2・］表示。

［D］：（DESTINATION）目标操作数。若可使用变址功能时，表示为［D・］。目标不止一个时，用［D1・］、［D2・］表示。

m：其他操作数。常常用来表示数制（十进制、十六进制等）或作为源和目标的补充注释。需注释的项目多时，也可采用 m1、m2 等形式。

功能指令的功能号和助记符占 1 个程序步；操作数占 2 个或 4 个程序步，取决于指令是 16 bit 还是 32 bit 的。

例中的 X0 为指令执行条件。X0 = ON，执行该指令；X0 = OFF，不执行该指令。

注意，某些功能指令在整个程序中只能出现一次，即使用跳转指令使其分处于两段不可能同时执行的程序中也不允许。但可利用变址寄存器多次改变其操作数。

4.1.2　数据长度及指令的执行方式

功能指令可处理 16 bit 的数据和 32 bit 的数据，如图 4－3 所示。

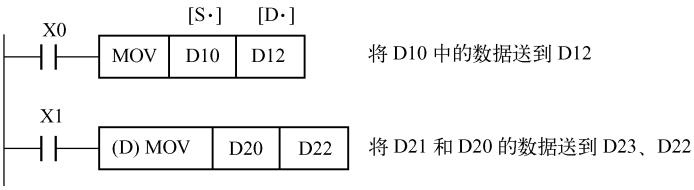


图 4－3　处理 16 bit 的数据和 32 bit 的数据示例

功能指令中附有符号（D）表示处理 32 bit 数据，如（D）MOV、FNC（D）12、FNC12（D）。

处理 32 bit 数据时，用元件号相邻的两元件组成元件对。元件对的首元件号用奇数、偶数均可。但为避免错误，元件对的首元件建议统一用偶数编号。

32 bit 计数器（C200 ~ C255）不能用作 16 bit 指令的操作数。

4.1.3　位元件

1. 位元件和字元件

只处理 ON/OFF 状态的元件，例如 X、Y、M 和 S，称为位元件；其他处理数字数据的元件，例如 T、C 和 D，称为字元件。

但位元件组合起来也可处理数字数据。位元件组合由 Kn 加首元件号来表示。

2. 位元件的组合

位元件每 4 bit 为一组组合单元。KnM0 中的 n 是组数。16 bit 数操作时为 K1 ~ K4。32 bit数操作时为 K1 ~ K8。例如，K2M0 即表示由 M0 ~ M7 组成 2 个 4 bit 组。

当一个 16 bit 的数据传送到 K1M0、K2M0 或 K3M0 时，只传送相应的低位数据，较高位的数据不传送，32 位数据传送时也一样。

在做 16 bit 数操作时，参与操作的位元件由 K1、K2、K3 来指定，高位（不足部分）均为 0。这就意味着只能处理正数（符号位为 0）。在做 32 bit 数操作时也一样。

被组合的位元件的首元件号可以是任意的，但为避免混乱建议采用以 0 结尾的元件。如 M0、M10 等。

4.1.4 整数与实数

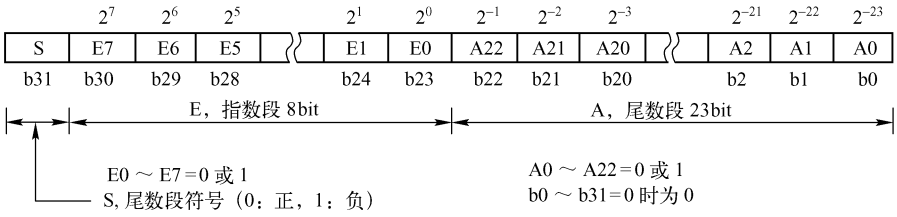
1. 整数

在 PLC 中，整数的表示及运算采用 BIN 码格式，可以用 16 bit 或 32bit 元件来表示整数，其中最高位为符号 bit，0 表示正数，1 表示负数。负数以补码方式表示。

整数可表示的范围：16 位时为 −32 768 ~ +32 767，32 位时为 −2 147 483 648 ~ +2 147 483 647。除表示范围受限制外，进行科学运算时产生的误差也较大，所以需要引入实数。

2. 实数的浮点格式

实数必须用 32 bit 来表示，通常用数据寄存器来存放实数。实数的浮点格式如图 4–4 所示。



实数值 $= S \cdot (1 \cdot A)(2^{E-127})$

实数值 $= \pm (2^0 + A22 \times 2^{-1} + A21 \times 2^{-2} + \cdots + A0 \times 2^{-23}) \times 2^{(E7 \times 2^7 + E6 \times 2^6 + \cdots + E0 \times 2^0)} / 2^{127}$

（例） A22 = 1，A21 = 0，A20 = 1，A19 ~ A0 = 0

E7 = 1，E6 ~ E1 = 0，E0 = 1

实数值 $= \pm (2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + \cdots + 0 \times 2^{-23}) \times 2^{(1 \times 2^7 + 0 \times 2^6 + \cdots + 1 \times 2^0)} / 2^{127}$

$= \pm 1.625 \times 2^{129} / 2^{127} = \pm 1.625 \times 2^2$）

图 4–4 实数的浮点格式示例

标志：零（M8020），借位（M8021），进位（M8022）：

零标志：结果为 0 时置 1。

借位标志：结果未达到最小值但不是 0 时置 1。

进位标志：结果的绝对值超过可能处理的最大数值时置 1。

浮点式实数的范围：$\pm 3.403 \times 10^{38} \sim \pm 1.175 \times 10^{-38}$。

3. 实数的科学记数格式

PLC 内实数的处理是采用上述浮点格式的，但浮点格式不便于监视，所以引入实数的科学记数格式。这是一种介于 BIN 与浮点格式之间的表示方法。用这种方法来表示实数也需占用 32 bit，即两个字元件。通常也用数据寄存器对（例如 D1、D0）来存放科学记数式实数。这

时，序号小的数据寄存器（D0）存放尾数，序号大的数据寄存器存放以 10 为底的指数。

科学记数格式实数 = 尾数 $\times 10^{指数}$

尾数范围为 ±（1000 ~ 9999）或 0；

指数范围为 −41 ~ +35。

注意：尾数应以 4 位有效数字（不带小数）表示，例如 2.34567×10^5 应表示为 2345×10^2。

4.1.5 变址寄存器（V、Z）

变址寄存器在传送、比较指令中用来修改操作对象的元件号。其操作方式与普通数据寄存器一样。

图 4-5 中表示从 K_nY 到 V、Z 都可作为功能指令的目标元件。在［D·］中的点（·）表示可以加入变址寄存器。对 32 bit 指令，V 为高 16 bit，Z 为低 16 bit。32bit 指令中用到变址寄存器时只需指定 Z，这时 Z 就代表了 V 和 Z。

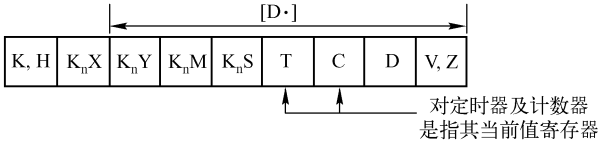


图 4-5　变址寄存器

图 4-6 中，因为 K10 送到 V0，K20 送到 Z0，所以（V0）、（Z0）的内容分别为 10、20。

（D5V0）+（D15Z0）→（D40Z0）

即

（D15）+（D35）→（D60）

在 32bit 指令中，V、Z 自动组队使用。V0 已由 MOV 指令指定为“0”。

（D1，D0）+（D3，D2）→（D25，D24）

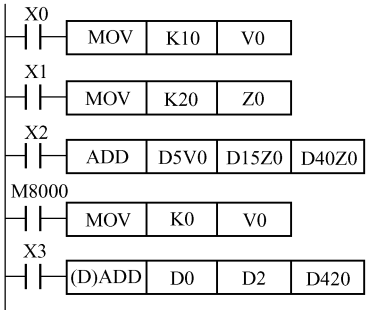


图 4-6　变址寄存器操作示例

某些应用会用到各种标志。例如：

M8020：零标志（Z）

M8021：借位标志（Br）

M8022：进位标志（Cy）

M8029：执行完毕标志（F）

如果功能指令的语法或操作元件号有错误，就会出错。出错时，出错标志 M8067（E）将被置地。

功能指令每次执行时都会影响标志的状态（SET 或 RESET），而功能指令不执行或出错时，标志状态不受影响。许多功能指令都会影响标志的状态，编程时要多加小心。

4.1.6 功能指令简表及某些功能指令使用次数的限制

在本章中介绍的功能指令是 FX 系列 PLC 总共可使用的功能指令，具体到某个型号（例如 FX1s），就只能用其中的一部分，哪些可用、哪些不能用从功能指令简表（见表 4-1）中可以查出。另外，有些指令的使用次数也有限制，见表 4-1，表中○表示相应指令功能可用；—表示相应指令功能不可用。另外，有些指令的使用次数也有限制，见表 4-2。

表 4-1　功能指令简表

分类	FNC NO.	指令助记符	功　　能	对应可编程序控制器				分类	FNC NO.	指令助记符	功　　能	对应可编程序控制器			
				FX_{1S}	FX_{1N}	FX_{2N}	FX_{2NC}					FX_{1S}	FX_{1N}	FX_{2N}	FX_{2NC}
程序流控制	00	CJ	条件跳转	○	○	○	○	四则与逻辑运算	23	DIV	BIN 除法	○	○	○	○
	01	CALL	转子程序	○	○	○	○		24	INC	BIN 加 1	○	○	○	○
	02	SRET	子程序返回	○	○	○	○		25	DEC	BIN 减 1	○	○	○	○
	03	IRET	中断返回	○	○	○	○		26	WAND	字与	○	○	○	○
	04	EI	允许中断	○	○	○	○		27	WOR	字或	○	○	○	○
	05	DI	禁止中断	○	○	○	○		28	WXOR	字异或	○	○	○	○
	06	FEND	主程序结束	○	○	○	○		29	NEG	求补	—	—	○	○
	07	WDT	警戒时钟	○	○	○	○	循环移位与移位	30	ROR	循环右移	—	—	○	○
	08	FOR	循环区起点	○	○	○	○		31	ROL	循环左移	—	—	○	○
	09	NEXT	循环区终点	○	○	○	○		32	RCR	带进位循环右移	—	—	○	○
传送与比较	10	CMP	比较	○	○	○	○		33	RCL	带进位循环左移	—	—	○	○
	11	ZCP	区间比较	○	○	○	○		34	SFTR	右移位	○	○	○	○
	12	MOV	传送	○	○	○	○		35	SFTL	左移位	—	—	○	○
	13	SMOV	移位传送	—	—	○	○		36	WSFR	字右移	—	—	○	○
	14	CML	取反传送	—	—	○	○		37	WSFL	字左移	○	○	○	○
	15	BMOV	块传送	○	○	○	○		38	SFWR	FIFO 写入	○	○	○	○
	16	FMOV	多点传送	—	—	○	○		39	SFRD	FIFO 读出	○	○	○	○
	17	XCH	交换	—	—	○	○	数据处理	40	ZRST	区间复位	○	○	○	○
	18	BCD	BCD 变换	○	○	○	○		41	DECO	解码	○	○	○	○
	19	BIN	BIN 变换	○	○	○	○		42	ENCO	编码	○	○	○	○
四则与逻辑运算	20	ADD	BIN 加法	○	○	○	○		43	SUM	ON 总数	—	—	○	○
	21	SUB	BIN 减法	○	○	○	○		44	BON	ON 位数判别	—	—	○	○
	22	MUL	BIN 乘法	○	○	○	○		45	MEAN	平均值	—	—	○	○

（续）

分类	FNC NO.	指令助记符	功　　能	对应可编程序控制器			
				FX_{1S}	FX_{1N}	FX_{2N}	FX_{2NC}
数据处理	46	ANS	报警器置位	—	—	○	○
	47	ANR	报警器复位	—	—	○	○
	48	SQR	开平方	—	—	○	○
	49	FLT	整数—实数变换	—	—	○	○
高速处理	50	REF	刷新	○	○	○	○
	51	REFE	刷新和滤波时间调整	—	—	○	○
	52	MTR	矩阵输入	○	○	○	○
	53	HSCS	高速计数器置位	○	○	○	○
	54	HSCR	高速计数器复位	○	○	○	○
	55	HSZ	高速计数器区间比较	—	—	○	○
	56	SPD	速度检测	○	○	○	○
	57	PLSY	脉冲输出	○	○	○	○
	58	PWM	脉宽调制	○	○	○	○
	59	PLSR	带加减速的脉冲输出	○	○	○	○
方便指令	60	IST	置初始状态	○	○	○	○
	61	SER	数据搜索	—	—	○	○
	62	ABSD	绝对值式凸轮顺控	○	○	○	○
	63	INCD	增量值式凸轮顺控	○	○	○	○
	64	TTMR	示教定时器	—	—	○	○
	65	STMR	特殊定时器	—	—	○	○
	66	ALT	交替输出	○	○	○	○
	67	RAMP	斜坡信号	○	○	○	○
	68	ROTC	旋转台控制	—	—	○	○

分类	FNC NO.	指令助记符	功　　能	对应可编程序控制器			
				FX_{1S}	FX_{1N}	FX_{2N}	FX_{2NC}
方便指令	69	SORT	数据排序	—	—	○	○
外部I/O设备	70	TKY	10 键输入	—	—	○	○
	71	HKY	16 键输入				
	72	DSW	数字开关	○	○	○	○
	73	SEDG	7 段解码	—	—	○	○
	74	SEGL	带锁存的 7 段显示	○	○	○	○
	75	ARWS	方向开关	—	—	○	○
	76	ASC	ASCⅡ码转换	—	—	○	○
	77	PR	打印	—	—	○	○
	78	FROM	读特殊功能模块	—	○	○	○
	79	TO	写特殊功能模块	—	○	○	○
外部设备	80	RS	串行数据传送	○	○	○	○
	81	PRUN	并联运行	○	○	○	○
	82	ASCI	HEX→ASCⅡ变换	○	○	○	○
	83	HEX	ASCⅡ→HEX 变换	○	○	○	○
	84	CCD	校验码	○	○	○	○
	85	VRRD	读变量	○	○	○	○
	86	VRSC	变量整标	○	○	○	○
	88	PID	PID 运算	○	○	○	○
实数处理	110	ECMP	实数比较	—	—	○	○
	111	EZCP	实数区间比较	—	—	○	○
	118	EBCD	浮点数—科学记数变换	—	—	○	○

（续）

分类	FNC NO.	指令助记符	功　　能	对应可编程序控制器			
				FX_{1S}	FX_{1N}	FX_{2N}	FX_{2NC}
实数处理	119	EBIN	科学记数—浮点数变换	—	—	○	○
	120	EADD	实数加法	—	—	○	○
	121	ESUB	实数减法	—	—	○	○
	122	EMUL	实数乘法	—	—	○	○
	123	EDIV	实数除法	—	—	○	○
	127	ESQR	实数开方	—	—	○	○
	129	INT	实数—整数变换	—	—	○	○
	130	SIN	正弦函数	—	—	○	○
	131	COS	余弦函数	—	—	○	○
	132	TAN	正切函数	—	—	○	○
	147	SWAP	高低 byte 互换	—	—	○	○
点位控制	155	ABS	当前绝对位置读取	○	○	—	—
	156	ZRN	回原点	○	○	—	—
	157	PLSV	变速脉冲输出	○	○	—	—
	158	DRVI	增量驱动	○	○	—	—
	159	DRVA	绝对位置驱动	○	○	—	—
实时时钟处理	160	TCMP	时间比较	○	○	○	○
	161	TZCP	时间区间比较	○	○	○	○
	162	TADD	时间加法	○	○	○	○
	163	TSUB	时间减法	○	○	○	○
	166	TRD	读实时时钟	○	○	○	○
	167	TWR	写实时时钟	○	○	○	○
	169	HOUR	计时表	○	○	—	—

分类	FNC NO.	指令助记符	功　　能	对应可编程序控制器			
				FX_{1S}	FX_{1N}	FX_{2N}	FX_{2NC}
外围设备	170	GRY	GRY—BIN 转换	—	—	○	○
	171	GBIN	BIN—GRY 变换	—	—	○	○
	176	RD3A	模拟量读出	—	—	○	—
	177	WR3A	模拟量写入	—	○	—	—
触点比较	224	LD =	[S1・] = [S2・]	○	○	○	○
	225	LD >	[S1・] > [S2・]	○	○	○	○
	226	LD <	[S1・] < [S2・]	○	○	○	○
	228	LD < >	[S1・] ≠ [S2・]	○	○	○	○
	229	LD≤	[S1・] ≤ [S2・]	○	○	○	○
	230	LD≥	[S1・] ≥ [S2・]	○	○	○	○
	232	AND =	[S1・] = [S2・]	○	○	○	○
	233	AND >	[S1・] > [S2・]	○	○	○	○
	234	AND <	[S1・] < [S2・]	○	○	○	○
	236	AND < >	[S1・] ≠ [S2・]	○	○	○	○
	237	AND≤	[S1・] ≤ [S2・]	○	○	○	○
	238	AND≥	[S1・] ≥ [S2・]	○	○	○	○
	240	OR =	[S1・] = [S2・]	○	○	○	○
	241	OR >	[S1・] > [S2・]	○	○	○	○
	242	OR <	[S1・] < [S2・]	○	○	○	○
	244	OR < >	[S1・] ≠ [S2・]	○	○	○	○
	245	OR≤	[S1・] ≤ [S2・]	○	○	○	○
	246	OR≥	[S1・] ≥ [S2・]	○	○	○	○

表 4-2　限制使用次数的指令

指令名	使用次数	
	FX_{1S}，FX_{1N}	FX_{2N}，FX_{2NC}
FNC52（MYR）	1	1
FNC58（PWM）	1	1
FNC60（IST）	1	1
FNC62（ABSD）	1	1
FNC63（INCD）	1	1
FNC68（ROTC）		1
FNC69（SORT）		1
FNC70（TKY）		1
FNC71（HKY）		1
FNC72（DSW）	无限制	2
FNC74（SEGL）	无限制	2
FNC75（ARWS）		1
FNC77（PR）		2
FNC57（PLSY） FNC59（PLSR）	同时驱动时有限制	

4.2　程序流控制（FNC 00 ~ FNC 09）

4.2.1　条件跳转

1. 条件跳转指令

CJ　FNC 00 （P）（16） 条件跳转	操作元件：指针 P0 ~ P63（允许变址修改） P63 即 END，无需再标号 程序步数：CJ 和 CJ（P）……3 步 标号 P × ×……1 步

CJ 和 CJP 指令用于跳过顺序程序中的某一部分，这样可以减少扫描时间，并使“双线圈操作”成为可能。

在图 4-7 所示的例子中，当 X0 为 ON 时，程序从第 1 步跳到第 36 步（标号 P8 的下一步）。如果 X0 为 OFF，跳转不执行，程序按原顺序向下执行。跳转时，被跳过的那部分的指令不执行。

即使触点状态变化，线圈的状态也不变，见表 4-3。

表 4-3　图 4-7 示例中跳转前后触点、线圈状态

元件	跳转前触点状态	跳转后触点状态	跳转后线圈状态
Y，M，S	X1，X2，X3　OFF	X1，X2，X3　ON	Y1，M1，S1　OFF
	X1，X2，X3　ON	X1，X2，X3　OFF	Y1，M1，S1　OFF
10ms 100ms 定时器	X4　OFF	X4　ON	定时器不动作
	X4　ON	X4　OFF	定时器停止 X0　OFF 后重新计时
1ms 定时器	X5　OFF X6　OFF	X6　ON	定时器不动作
	X5　OFF X6　ON	X6　OFF	定时器停止 X0　OFF 后重新计时
计数器	X7　OFF X10　OFF	X10　ON	计数器不动作
	X7　OFF X10　ON	X10　OFF	计数器停止
			X0　OFF 后重新计数
功能 指令	X11　OFF	X11　ON	除 FNC52 ~ FNC58 之外的 其他功能指令不执行
	X11　ON	X11　OFF	

注：1. Y1 成了双线圈，其操作由 X0 的 ON/OFF 状态决定，即 X0 OFF 时，Y1 由 X1 驱动；X0 ON 时，Y1 由 X12 驱动。双线圈中，一个在跳转程序之内、一个在跳转程序之外是不允许的。

2. 对积算型定时器及计数器的 RESET 指令在跳转程序中时，即使跳转生效，RESET 指令仍会被执行。

3. 定时器 T192 ~ T199、高速计数器 C235 ~ C255 一经驱动，即使处理指令被跳过，也会继续工作，输出触点也能动作。

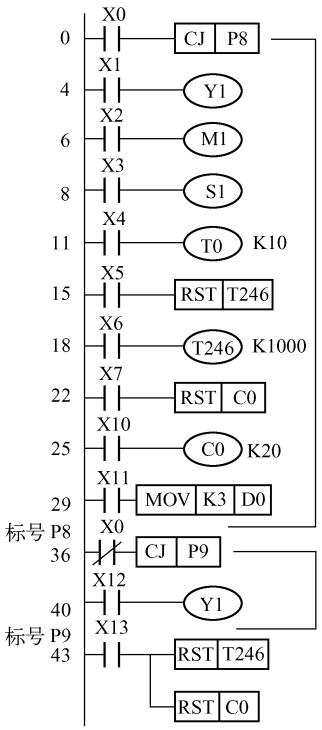


图 4-7　条件跳转示例

在程序中，两条跳转指令使用相同的指针号时（见图 4-8a），执行情况如下：

1）如果 X20 为 ON，第一条跳转指令生效，从这一步跳到标号 P9。如果 X20 为 OFF，而 X21 为 ON，则第二条跳转指令生效，程序从这里开始跳到标号 P9 处。

2）一个标号只能出现一次，如出现多于一次，则会出错。

标号可以出现在跳转指令之前，如图 4-8b 所示。但这时如 X22 ON 的时间超过 100ms，会引起警戒时钟出错。

图 4-9 所示为无条件跳转，因为 PLC 运行时 M8000 总为 ON。用 CJ（P）指令时，跳转仅在 X22 由 OFF 变为 ON 时生效。

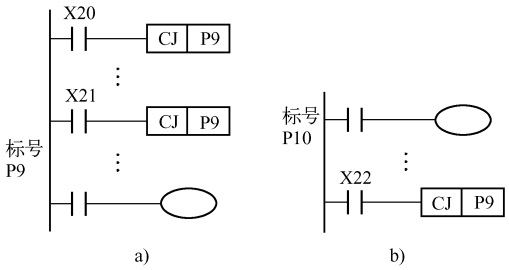


图 4-8　条件跳转

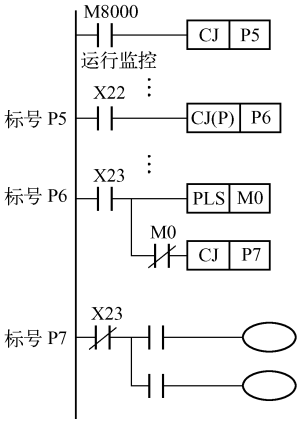


图 4-9　无条件跳转

在图 4-9 所示程序中，使“CJ P7”指令仅在 X23 由 OFF 变为 ON 后执行一个扫描周期。用这种方法，“CJ P7”和标号 P7 之间设置输出的程序执行完后，在以后的扫描周期中就被跳过（设置初始化条件时很有用）。

标号的编程方法如图 4-10 所示。

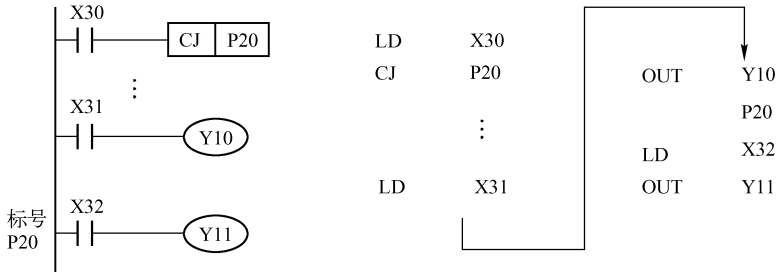


图 4-10　标号的编程方法

2. 跳转与主控区

图 4-11 所示为跳转与主控区的例子。

1）跳过整个主控区（MC ~ MCR）的跳转不受限制。

2）从主控区外跳到主控区内时，跳转独立于主控操作，CJ　KP1 执行时，不论 M0 状态如何，均作 ON 处理。

3）在主控区内跳转时，如 M0 为 OFF，跳转不可能执行。

4）从主控区内跳到主控区外时，M0 为 OFF 时，跳转不可能执行；M0 为 ON 时，跳转条件满足可以跳转，这时 MCR 被忽略，但不会出错。

5）从一个主控区内跳到另一个主控区内时，当 M1 为 ON 时，可以跳转。执行跳转时不论 M2 的实际状态如何，均看作 ON。MCR N0 被忽略。

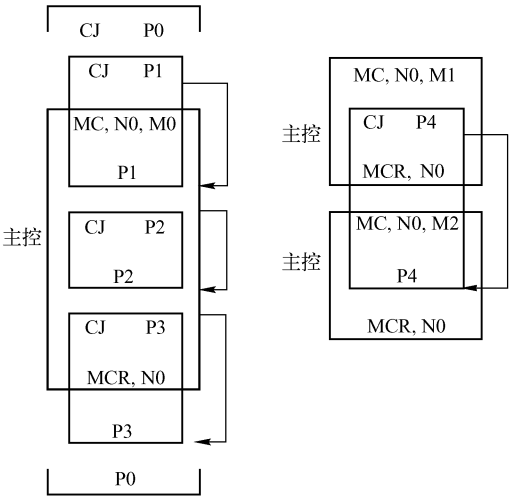


图 4-11　跳转与主控区例

4.2.2　子程序

CALL　FNC 01 （P）　（16） 转子程序	操作元件：指针 P0 ~ P62（允许变址修改） 步　　数：CALL 和 CALL（P）……3 步 标号 P × ×……1 步 嵌　　套：5 级
SRET　FNC 02 子程序返回	操作元件：无 步　　数：1 步

如图 4-12a 所示，当 X0 为 ON 时，CALL 指令使程序跳到标号 P10 处，子程序被执行。

在 SRET 指令执行后程序回到 104 步处。标号应写在 FEND 之后（FEND 指令在 4.2.4 节中解释）。标号范围为 P0～P62，但同一标号不能出现多于 1 次；CJ 指令中用过的标号不能重复再用。但不同的 CALL 指令可调用同一标号的子程序。

在图 4-12b 中，CALL（P）P11 仅 X1 在 OFF 变 ON 时执行一次。在执行 P11 子程序时，如果 CALL P12 指令被执行，则程序跳到子程序 P12。在 SRET（2）指令执行后，程序返回到子程序 P11 中 CALL P12 指令的下一步。在 SRET（1）指令执行后再返回主程序。

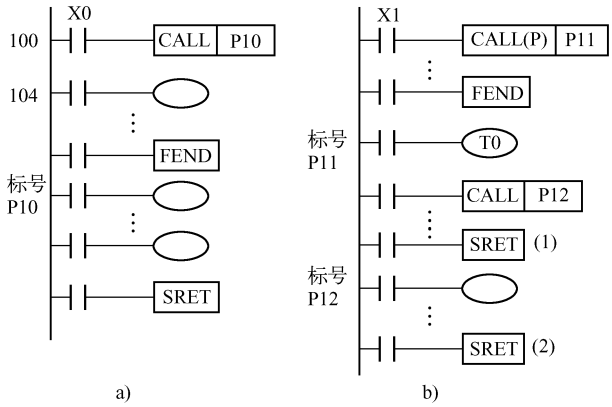


图 4-12　子程序指令示例

注意：

1）子程序中可套 CALL 子程序，形成子程序嵌套，总数可有 5 级嵌套。

2）在子程序和中断子程序中使用的定时器范围规定为 T192～T199 和 T246～T249。

3）可使用的标号 P 的数量：FX_{1S} 为 64，P0～P63；FX_{1N}、FX_{2N}、FX_{2NC} 为 128，P0～P127。

4.2.3　中断

1. 中断指令

IRET　FNC 03 中断返回	操作元件：无 步　　数：1 步
EI　FNC 04 允许中断	操作元件：无 步　　数：1 步
DI　FNC 05 禁止中断	操作元件：无 步　　数：1 步

PLC 通常在禁止中断状态。

如图 4-13 所示，指令 EI 与 DI 之间的程序段为允许中断区间。当程序处理到允许中断区间时，X0 或 X1 为 ON 态，则转而处理相应的中断子程序（1）或（2）。每个子程序处理到 IRET 指令时返回原断点。

在禁止中断区间（即 DI～EI 指令之间）有中断事件发生，则 PLC 记住该中断，留待 EI 指令后执行中断子程序（滞后执行）。

当相应的特殊辅助继电器置 1 时，中断子程序不能执行。例如，当 M805△置 1 时，相

应的中断子程序 I△＊＊ 不执行。

在一个中断程序执行中时，其他中断被禁止。但是，在中断程序中编入 EI 和 DI 指令可实现 2 级中断嵌套。

在子程序或中断子程序中可用定时器为 T192 ~ T199 和 T246 ~ T249。

注意：

1）1 个中断指针（I＊＊＊）占一步，可设置中断点的数参见本节下述内容。

2）指定中断指针的顺序参阅本节下述内容。

3）其中 2 个中断点可产生多层中断。

4）中断信号的脉冲必须超过 20μs。

5）多个中断信号顺序产生时，最先产生的中断信号有优先权。若同时产生 2 个或 2 个以上的中断信号，中断指针号较低的有优先权。

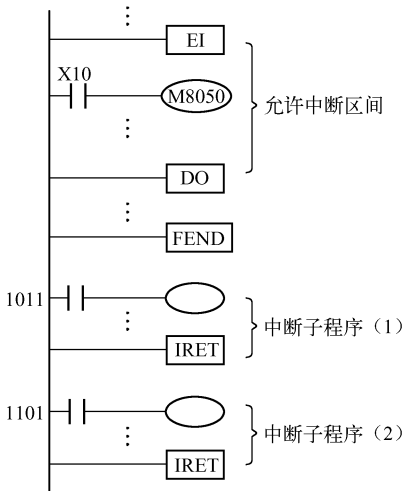


图 4-13　中断示例

6）如果中断信号产生于禁止中断区间（DI 到 EI 范围），则中断信号被存储，并在 EI 指令之后被执行，除非相应的 M8055△为 ON。

2. 中断的种类

中断分为输入中断、定时中断及高速计数器中断 3 种。

I□0□ 输入中断指针（6 点）

0：下降沿中断
1：上升沿中断

输入元件号（X0 ～ X5 对应 0 ～ 5）

输入编号	指针编号		禁止中断特殊 M
	上升沿中断	下降沿中断	
X000	I001	I000	M8050
X001	I101	I100	M8051
X002	I201	I200	M8052
X003	I301	I300	M8053
X004	I401	I400	M8054
X005	I501	I500	M8055

I□□□ 定时中断指针（3 点）

10 ～ 99 ms

定时器中断号 (6, 7, 8)
每个定时器只能用一次

指针编号	中断周期	禁止中断特殊 M
I6□□	在指针名称的□□部分中，输入 I0 ~ I9 的整数。I610 = 每 10ms 执行一次定时器中断	M8056
I7□□		M8057
I8□□		M8058

I0□0 高速计数器中断指针（6 点）

计数器中断号（1 ～ 6）

指针编号	禁止中断特殊 M
I010	M8059
I020	
I030	
I040	
I050	
I060	

注意：指针编号不可重复使用，同一输入点不可同时用上升/下降沿中断。禁止中断特殊辅助继电器置 1 时，禁止相应中断子程序的执行。

4.2.4 主程序结束

FEND FNC 06 主程序结束	操作元件：无 程序步数：1 步

FEND 指令表示主程序结束。执行到 FEND 指令时，机器进行输出处理、输入处理、警戒时钟刷新，完成以后返回到第 0 步，如图 4-14 所示。

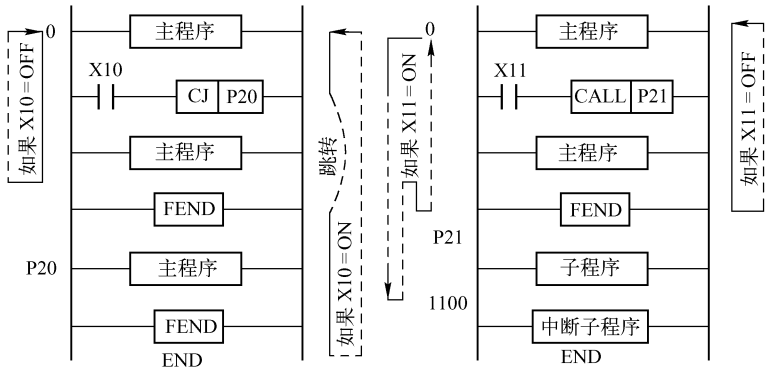


图 4-14 FEND 指令应用示例

子程序应写在 FEND 之后，即 CALL、CALL（P）指令对应的标号应写在 FEND 指令之后。CALL、CALL（P）指令调用的子程序必须以 SRET 指令作为结束。

同理，中断子程序也要写在 FEND 之后。中断子程序必须以 IRET 指令作为结束。

若 FEND 指令在 CALL 或 CALL（P）指令执行之后、SRET 指令执行之前出现，则程序被认为是错误的。另一个类似的错误是使 FEND 指令处于 FOR－NEXT 循环之中。

子程序及中断子程序必须写在 FEND 指令与 END 指令之间。若有多个 FEND 指令，则子程序必须在最后一个 FEND 指令与 END 指令之间。

4.2.5 警戒时钟

WDT FNC 07 （P） 警戒时钟	操作元件：无 程序步数：1 步

WDT 指令刷新顺序程序的警戒时钟。如果扫描周期（从 0 步到 END 或 FEND 指令）超过 200ms，PLC 将停止运行。在这种情况下，应将 WDT 指令插到合适的程序步中刷新警戒时钟，以使顺序程序得以继续执行，直到 END。

4.2.6 循环（FOR－NEXT）

在 FOR－NEXT 间的程序重复执行“n”次（由操作元件指定），然后再执行 NEXT 指令后的程序。循环次数范围：1～32 767，如循环次数设置为－32 767～0 之间时，循环次数作 1 处理。

FOR　FNC 08 （16） 循环区起点	操作元件：[S·] K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：3 步 嵌　　套：5 层
NEXT　FNC 09 循环区终点	程序步数：1 步

FOR 与 NEXT 总是成对出现，且应 FOR 在前，NEXT 在后。FOR—NEXT 循环指令最多可以嵌套 5 层。

利用 CJ 指令可以跳出 FOR—NEXT 循环体。

4.3　传送和比较（FNC 10 ~ FNC 19）

4.3.1　比较

CMP　FNC 10 （P）（16/32） 比较	操作元件：[S1·][S2·] K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z X　Y　M　S [D·] 程序步数：CMP 和 CMP（P）……7 步 （D）CMP 和（D）CMP（P）……13 步

图 4-15 所示的程序是将源［S1 ·］和［S2 ·］的数据比较，结果送到目标［D］中。

这里源数据作代数比较（如 -10 < 2），且所有源数据均作为二进制数值处理。

程序中，M0、M1、M2 根据比较的结果动作。K100 > C20 的当前值时，M0 接通；K100 = C20 的当前值时，M1 接通；K100 < C20 的当前值时，M2 接通。当执行条件 X0 = OFF 时，CMP 指令不执行，M0、M1、M2 的状态保持不变。

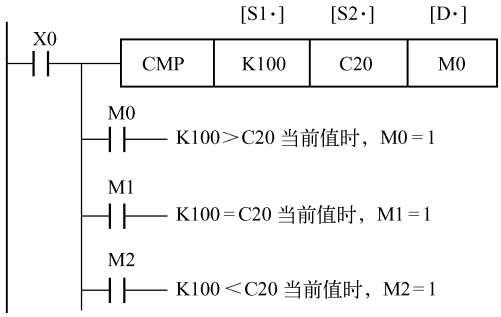


图 4-15　CMP 指令应用例

4.3.2　区间比较

ZCP　FNC 11 （P）（16/32） 区间比较	操作元件：[S1·][S2·][S·] K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z X　Y　M　S [D·] 程序步数：ZCP 和 ZCP（P）……9 步 D（ZCP）和（D）ZCP（P）……17 步

ZCP 指令是将一个数据与两个源数据值比较。

源［S1·］的数据不得大于［S2·］的值。例如，若［S1·］=K100，［S2·］=K90，则执行 ZCP 指令时看作［S2·］=K100。

源数据的比较是代数比较（如 -10 < 2）。

图 4-16 所示的程序中，M3 在 K100 > C30 的当前值时为 ON；M4 在 K100 ≤ C30 的当前值 ≤ K120 时为 ON；W 在 K120 < C30 的当前值时为 ON。

当 X0 = OFF 时，不执行 ZCP 指令，M3、M4、MS 的状态保持不变。

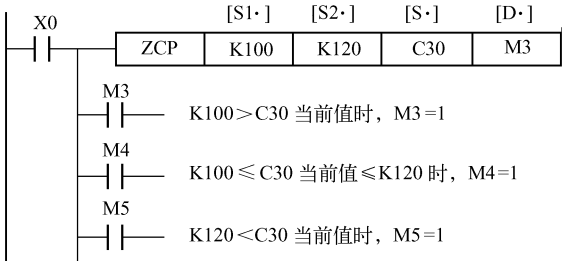


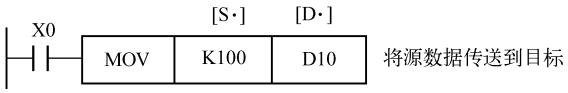
图 4-16　ZCP 指令应用示例

注意：M3、M4、M5 的状态取决于比较结果。

4.3.3　传送

MOV　FNC 12 （P）（16/32） 传送	操作元件：[S·]：K, X　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：MOV 和 MOV（P）……5 步 （D）MOV 和（D）MOV（P）……9 步

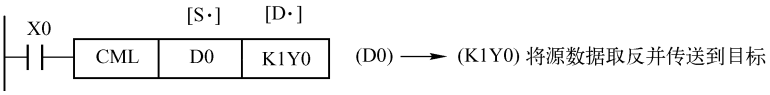
源数据被传送到指定目标。如 X0 = OFF 指令不执行，数据保持不变。



当传送指令执行时，常数 K100 自动转换成二进制数。

4.3.4　取反传递

CML　FNC 14 （P）（16/32） 取反传送	操作元件：[S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：CML 和 CML（P）……5 步 （D）MOV 和（D）MOV（P）……9 步



在源元件中的数据逐位取反（1→0，0→1）并传送到指定目标。若源中数据为常数 K，该数据会自动转为二进制数。

CML 指令用于使 PLC 获得反逻辑输出时非常方便，如图 4-17 所示。

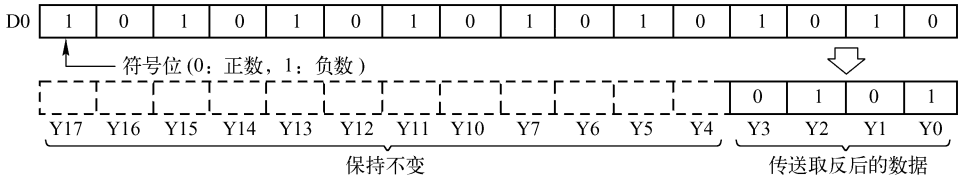
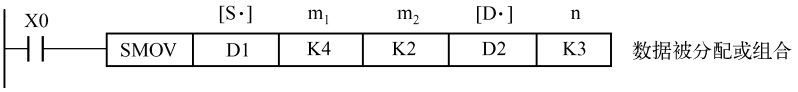


图 4-17　CML 指令应用示例

4.3.5　移位传送

SMOV　FNC 13 （P）　（16） 移位传送	操作元件：[S·]：K, X　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：m_1, m_2, n 程序步数：SMOV 和 SMOV（P）……11 步



如图 4-18 所示，首先源数据（二进制）被转换成 BCD 码，然后将 BCD 码移位传送。源数据 BCD 码右起第 4 位（M1 =4）开始的 2 位（M2 =2）移到目标的第 3 位（n =3）和第 2 位。

然后，目标中的 BCD 码自动转换为二进制码。目标中的第 1 位和第 4 位不受移位传送指令的影响。

BCD 码值超过 9999 时出错。

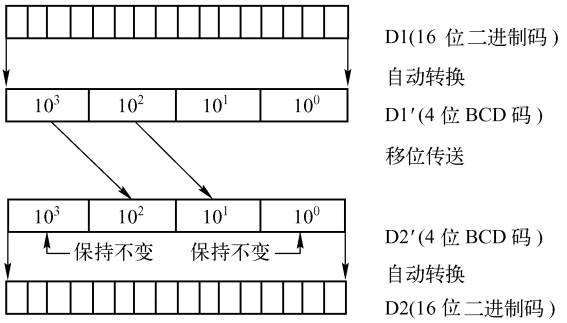


图 4-18　移位传送

例如，图 4-19a 所示，从接在分开的接线端子上的 3 个数字开关的输入组合成数据输入。

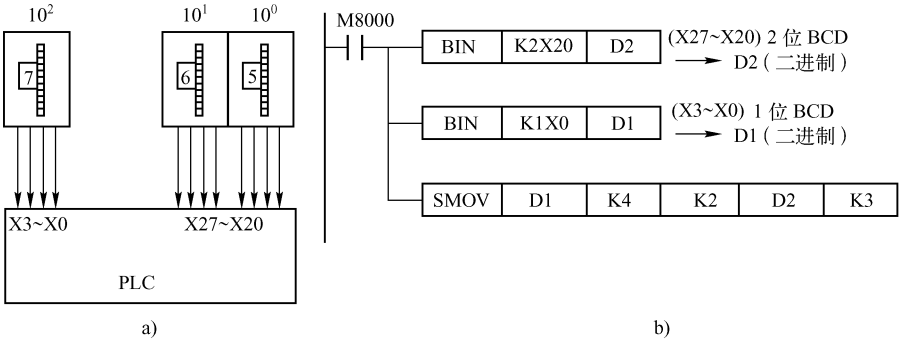


图 4-19　移位传送示例

图 4-19b 中，D1 的 1 位 BCD 码数移送到 D2 的第 3 位上，然后自动转换成二进制数。

从上述操作可知，3 位数字开关上的数字输入组合起来以二进制数形式存于 D2。

4.3.6 块传送

BMOV (P) 块传送　FNC 15 (16)	操作元件：[S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　文件寄存器　V, Z [D·]：K_nY　K_nM　K_nS　T　C　D　文件寄存器 n（n≤512） 程序步数：BMOV 和 BMOV（P）……7 步

从源操作数指定的元件开始的 n 个数据组成的数据块传送到指定的目标。如果元件号超出允许元件号的范围，数据仅传送到允许范围内。

如果源元件与目标元件的类型相同，传送顺序如图 4-20 所示（既可从高元件号开始，也可从低元件号开始）。

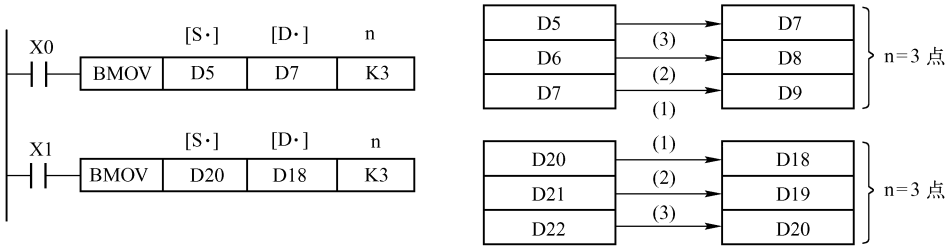


图 4-20　传送顺序

传送顺序是自动决定的，以防止源数据被这条指令传送的其他数据冲掉。

如果用到需要指定位数的 bit 元件，则源和目标的指定位数必须相同。

当 M8024 = ON 时，数据传送方向反转，如图 4-21 所示。

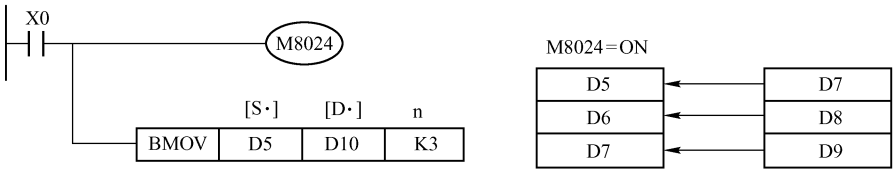
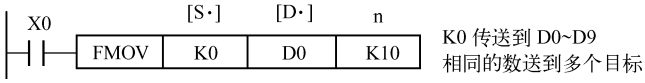


图 4-21　数据传送方向反转

对 FX_{1N}、FX_{2N}、FX_{2NC}而言，通过参数设置可把 D1000 以后的通用数据寄存器设为文件寄存器。当 PLC 从 STOP→RUN 时，程序存储区的文件寄存器会自动被复制到系统 RAM 区中的文件寄存器中。除 FNC15 BMOV 指令外，所有功能指令中所用到的 D1000 以后的元件均指系统 RAM 区中的文件寄存器，只有 FNC15 BMOV 指令有访问程序区中文件寄存器的功能。

4.3.7 多点传送

FMOV FNC 15 （P） （16） 多点传送	操作元件：[S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS　T　C　D　V, Z n（n≤512） 程序步数：FMOV 和 FMOV（P）……7 步

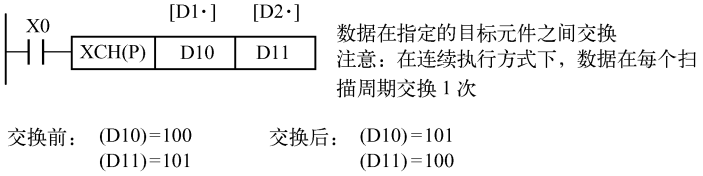


源元件中的数据传送到指定目标开始的 n 个元件中。这 n 个元件中的数据完全相同。

如果元件号超出允许元件号范围，数据仅送到允许范围的元件中。

4.3.8 交换

XCH FNC 17 （P） （16/32） （“1”） 交换	操作元件：[Dl·][D2·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：XCH 和 XCH（P）……5 步 （D）XCH 和（D）XCH（P）……9 步



如图 4-22 所示，当 M8160 = ON，而且［D1·］与［D2·］为同一元件时，该指令的执行使目标元件的高字节与低字节互换，这时本指令与 FNC147SWAP 的功能相同。若 M8160 = ON 时［D1·］与［D2·］元件不同时，出错标志 M8067 = ON，且不执行该指令。

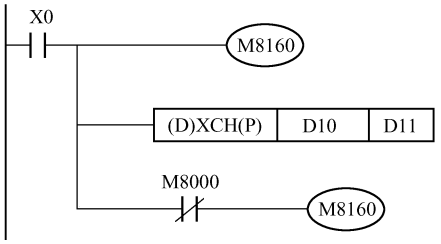
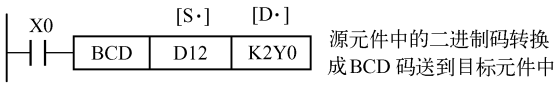


图 4-22 交换指令使用示例

4.3.9 BCD 变换

BCD FNC 18 （P） （16/32） 二进制换成 BCD 码	操作元件：[S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：BCD 和 BCD（P）……5 步 （D）BCD 和（D）BCD（P）……9 步



如果 BCD /BCD（P）指令执行的变换结果超出 0 ~ 9 999 的范围就会出错。

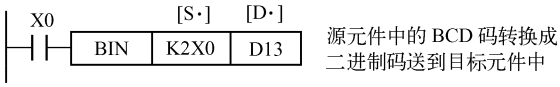
如果（D）BCD/（D）BCD（P）指令执行的变换结果超出 0 ~ 99 999 999 的范围就会出错。例如，可用 BCD 码指令将 PLC 中的 H 进制数变换成 BCD 码输出，以驱动 7 段显示。

4. 3. 10 BIN 变换

BIN 指令用于将从 BCD 数字开关串的设定值输入 PLC 中。如果源元件中的数据不是 BCD 码的就会出错，M8067 ON。但在这种情况下 M8068（操作错误锁存）不为 ON。

常数 K 不能作为本指令的操作元件，因为在任何处理之前它会被转换成二进制数。

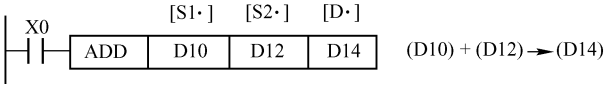
BIN FNC 19 （P） （16/32） BIN 码变换	操作元件：[S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z；[D·]：K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：BIN 和 BIN（P）……5 步 （D）BIN 和（D）BIN（P）……9 步 数值范围：0 ~ 9 999 或 0 ~ 99 999 999



4. 4 四则运算与逻辑运算（FNC 20 ~ FNC 29）

4. 4. 1 BIN 加法

ADD FNC 20 （P） （16/32） BIN 加法	操作元件：[S1·][S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z；[D·]：K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：ADD 和 ADD（P）……7 步 （D）ADD 和（D）ADD（P）……13 步 标　　志：M8020（零标志），M8021（借位），M8022（进位）



指定的源元件中的二进制数相加，结果送到指定的目标元件。每个数据的最高位作为符号位（0 为正，1 为负），运算总是代数运算，如

$$5 + (-8) = -3$$

如果运算结果为 0，则零标志 M8020 置 1。

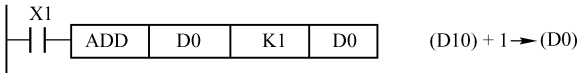
如果运算结果超过 32 767（16 bit 运算）或 2 147 483 647（32bit 运算），则进位标志

M8022 置 l。

如果运算结果小于 -32 767（16bit 运算）或 -2 147 483 647（32 bit 运算），则借位标志 M8021 置 1。

在 32 bit 运算中，用到子元件时，被指定的字元件是低 16 bit 元件，而其下一个元件即为高 16 bit 元件。为了避免重复使用某些元件，建议指定操作元件时用偶数元件号。

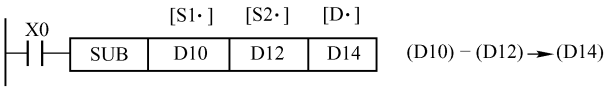
源和目标可以用相同的元件号，若源和目标元件号相同而且采用连续执行的 ADD/（D）ADD 指令时，加法的结果在每个扫描周期都会改变。



上面所示程序，每当 X1 从 OFF 变为 ON 时，D0 的数据加 1。这与 INC（P）指令的执行结果相似。其不同之处在于，用 ADD 指令时，零、借位、进位标志将按前述方法置位。

4.4.2 BIN 减法

SUB FNC 21 （P） （16/32） BIN 减法	操作元件：[S1·][S·]：K, H ｜ K_nX ｜ K_nY ｜ K_nM ｜ K_nS ｜ T ｜ C ｜ D ｜ V, Z；[D·] 程序步数：SUB 和 SUB（P）……7 步 （D）SUB 和（D）SUB（P）……13 步 标　　志：M8020（零标志），M8021（借位），M8022（进位）



［S1・］指定的元件中的数减去［S2・］叫指定的元件中的数，结果送到［D・］指定的目标中。运算是二进制代数法。例如

$$5-(-8)=13$$

每个标志的功能、32 bit 运算的元件指定方法、连续执行和脉冲执行的区别等均与加法指令中的解释相同。

上面所示的运算与执行（D）DEC（P）指令的运算极其相似，区别仅在于用 SUB 指令时可得到标志的状态。

标志的置位情况及标志置位与正/负数的关系如图 4-23 所示。

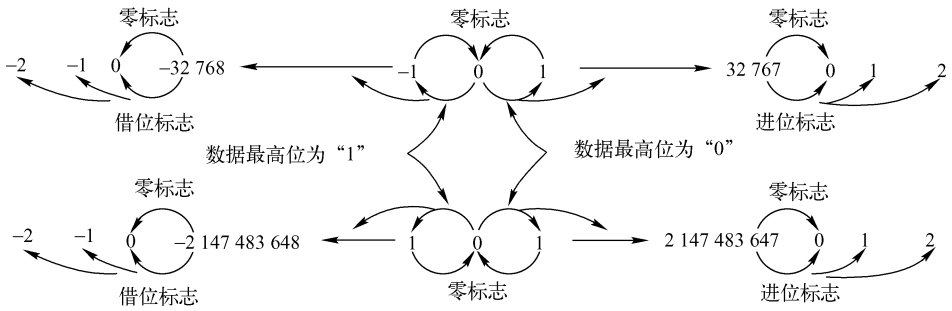


图 4-23　标志的置位情况及标志置位与正/负数的关系

4.4.3　BIN 乘法

MUL　FNC 22 （P）　（16/32） BIN 乘法	操作元件：[S1·][S2·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS　T　C　D　V, Z 程序步数：MUL 和 MUL（P）……7 步 （D）MUL 和（D）MUL（P）……13 步

MUL 指令是将两个源元件中的数据的乘积送到指定目标元件。如果为 16bit 运算，则乘积为 32 位；如果为 32bit 运算，则乘积为 64 位，如图 4-24 所示。

如果目标元件用位元件指定，则只能得到指定范围内的乘积。

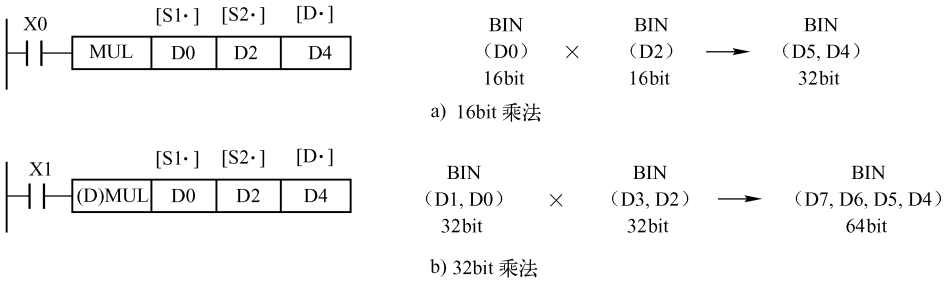


图 4-24　乘法示例

4.4.4　BIN 除法

DIV　FNC 23 （P）　（16/32） BIN 除法	操作元件：[S1·][S2·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS　T　C　D　V, Z　* 程序步数：DIV 和 DIV（P）……7 步 （D）DIV 和（D）DIV（P）……13 步

DIV 指令可以进行 16bit 除法和 32bit 除法。得到商和余数，并将结果送到指定目标元件，如图 4-25 所示。

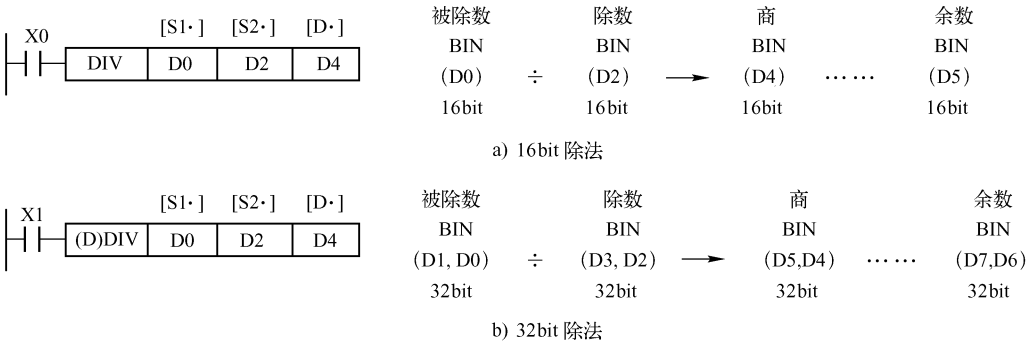
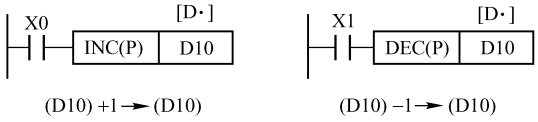


图 4-25　除法示例

对于 16bit 乘、除法，V 不可用于［D·］。对于 32bit 乘、除法，V 和 Z 不可用于［D·］。

4.4.5　加 1 和减 1

INC　FNC 24 （P）（16/32）（“!”） 加 1	操作元件：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z ←[D·]→
DEC　FNC 25 （P）（16/32）（“!”） 减 1	程序步数：INC、INC（P）、DEC 和 DEC（P）……3 步 （D）INC、（D）INC（P）、（D）DEC 和（D）DEC（P）……5 步



1）左边示例为加 1 运算，每次 X0 由 OFF 变 ON 时，由［D·］指定的元件中的数加 1。如果不用脉冲指令，则每个扫描周期加 1。

在 16 bit 运算中，到 +32 767 再加 1 就变为 −32 768，但标志不动作。

32 bit 运算时，+2 147 483 647 再加 1 就变为 −2 147 483 648 时，标志也不动作。

2）右边示例为减 1 运算，每次 X1 由 OFF 变为 ON 时，由［D·］指定的元件中的数减 1。

16 bit 运算时，−32 768 减 1 就变为 +32 767，但标志不动作。

32 bit 运算时，−2 147 483 648 再减 1 就变为 +2 147 483 647，仅标志不动作。

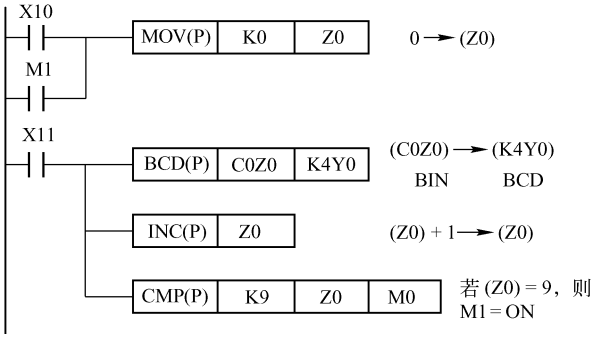


图 4-26　加 1 操作示例

如图 4-26 所示，计数器 C0 ~ C9 的当前值转换成 BCD 码送到输出 K4Y0。Z0 由复位输入 X10 清 0。每次 X11 ON 时，C0 ~ C9 的当前值依次输出。

4.4.6 与、或和异或

WAND FNC 26
(P) (16/32)
字与

WOR FNC 27
(P) (16/32)
字或

WXOR FNC 28
(P) (16/32)
字异或

操作元件：

[S1·][S2·]								
K, H	K_nX	K_nY	K_nM	K_nS	T	C	D	V, Z
		[D·]						

程序步数：

7 步（16bit 操作）		13 步（32bit 操作）	
WAND WAND(P)	(FNC 26) (FNC 26(P))	(D)AND (D)AND(P)	(FNC(D)26) (FNC(D)26(P))
WOR WOR(P)	(FNC 27) (FNC 27(P))	(D)OR (D)OR(P)	(FNC(D)27) (FNC(D)27(P))
WXOR WXOR(P)	(FNC 28) (FNC 28(P))	(D)XOR (D)XOR(P)	(FNC(D)28) (FNC(D)28(P))

16bit 的指令附加“W”

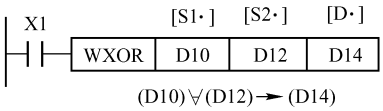
1. 逻辑与

以 bit 为单位作“与”运算：
$1 \wedge 1 = 1$　　$0 \wedge 1 = 0$
$1 \wedge 0 = 0$　　$0 \wedge 0 = 0$

2. 逻辑或

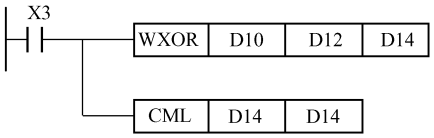
以 bit 为单位作“或”运算：
$1 \vee 1 = 1$　　$0 \vee 1 = 1$
$0 \vee 0 = 0$　　$1 \vee 0 = 1$

3. 逻辑异或



以 bit 为单位作“异或”运算：
$1 \forall 1 = 0$　　$0 \forall 0 = 0$
$0 \forall 0 = 0$　　$0 \forall 1 = 1$

XOR 指令与 CML（FNC14）组合可实现异或非运算。



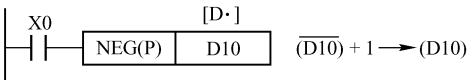
4.4.7 求补

NEG FNC 29
(P)(16/32)
求补

操作元件：

K, H	K_nX	K_nY	K_nM	K_nS	T	C	D	V, Z
		[D·]						

程序步数：NEG 和 NEG（P）……3 步
（D）NEG 和（D）NEG（P）……5 步



NEG 指令只有目标操作元件。它将［D・］指定的元件中的数的每一位取反，然后该数再加 1，结果存于同一元件。本指令实际是绝对值不变的变号操作。

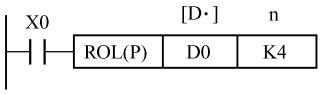
4.5 循环移位与移位（FNC 30 ~ FNC 39）

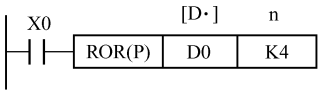
4.5.1 循环移位（左/右）

ROR FNC 30 （P）（16/32） （“!”） 右循环	操作元件：K, H ｜ K_nX ｜ K_nY ｜ K_nM ｜ K_nS ｜ T ｜ C ｜ D ｜ V, Z n (n<16(16bit 指令)；n<32(32bit 指令))　[D·]
ROL FNC 31 （P）（16/32） （“!”） 左循环	程序步数：ROR，ROR（P），ROL，ROL（P）……5 步 （D）ROR，（D）ROR（P），（D）ROL，（D）ROL（P）……9 步 标　　志：M8022（进位）

本指令使 16bit 数据向左向右循环移位。

左循环如图 4-27 所示，每次 X0 由 OFF→ON 时，各 bit 数据向左旋转 n bit，最后移出 bit 的状态存入进位标志 M8022 中。





右循环如图 4-28 所示，每次 X0 由 OFF→ON 时，各 bit 数据向右旋转 n bit，最后移出 bit 的状态存入进位标志 M8022 中。

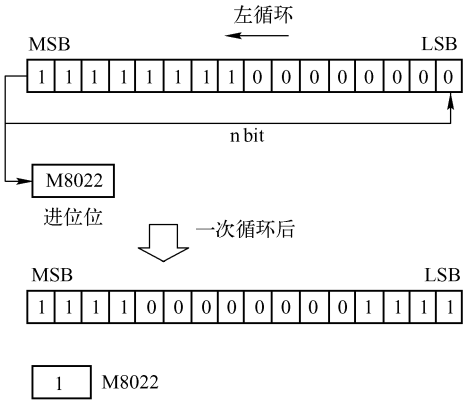


图 4-27　ROL（P）指令

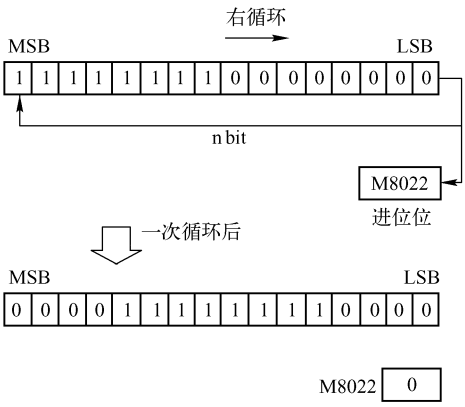


图 4-28　ROR（P）指令

用连续执行指令时，循环移位操作每个周期执行一次。

上面解释 16 bit 指令的 ROR、ROL 的执行情况也适用于 32 bit 指令。

若在目标元件中指定“位”数，则只能用 K4（16 bit 指令）和 K8（32 bit 指令）。例如 K4Y10，K8M0。

4.5.2 带进位的循环

RCR FNC 32 （P）（16/32） （“!”） 带进位右循环移位	操作元件：K, H ｜ K_nX ｜ K_nY ｜ K_nM ｜ K_nS ｜ T ｜ C ｜ D ｜ V, Z [D·] n (n<16(16bit 指令)；n<32(32bit 指令)) 程序步数：RCR，RCR（P），RCL，RCL（P）……5 步 （D）RCR，（D）RCR（P），（D）RCL，（D）RCL（P）……9 步 标　　志：M8022（进位）
RCL FNC 33 （P）（16/32） （“!”） 带进位左循环移位	

执行 RCR、RCL 指令时，各位的数据与进位位 M8022 一起（16bit 指令时一共 17 位）向右（或向左）循环移动 n 位。在循环中移出的位送入进位标志，后者又被送到目标操作元件的另一端。

4.5.3 移位（左/右）

SFTR FNC 34 （P）（16/32） （“!”） 右移位	操作元件：[S·]：X ｜ Y ｜ M ｜ S；[D·]；K, H；n_1, n_2 $n_2 \leqslant n_1 \leqslant 1024$ 程序步数：SFTR，SFTR（P），SFTL，SFTL（P）……9 步
SFTL FNC 35 （P）（16/32） （“!”） 左移位	

本指令使 bit 元件中的状态向右/向左移位，由 n_1 指定 bit 元件长度，n_2 指定移位 bit 数（$n_2 < n_1 < 1\ 024$）。

用脉冲指令时，在执行条件的上升沿时执行。用连续指令时，当执行条件 ON，则每个扫描周期执行一次。

右移位如图 4-29 所示。左移位如图 4-30 所示。

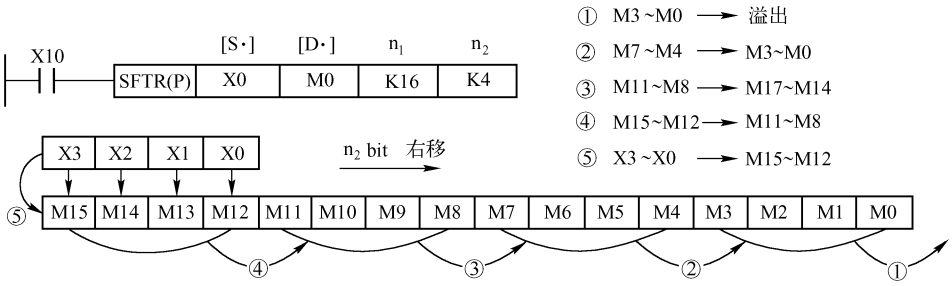


图 4-29　SFTR（P）指令使用示例

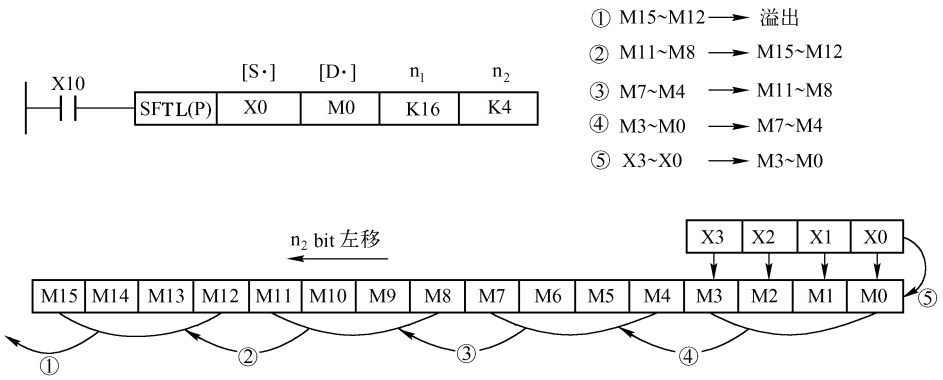


图 4-30　SFTL（P）指令使用示例

4.5.4　字右移/字左移

WSFR　FNC 36 （P）（16）（“!”） 字右移	操作元件：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [S·]　[D·] n_1, n_2（$n_2 \leqslant n_1 \leqslant 512$） 程序步数：WSFR，WSFR（P），WSFL，WSFL（P）……9 步
WSFL　FNC 37 （P）（16）（“!”） 字左移	

本指令使字元件中的状态向右/向左移位，由 n_1 指定字元件长度，n_2 指定移位字数（$n_2 < n_1 < 512$）。

用脉冲指令时，在执行条件的上升沿时执行。用连续指令时，当执行条件 ON，则每个扫描周期执行一次。

由［S·］、［D·］指定的元件若需要指定“位”数时，其位数应相同。

右移位如图 4-31 所示。左移位如图 4-32 所示。

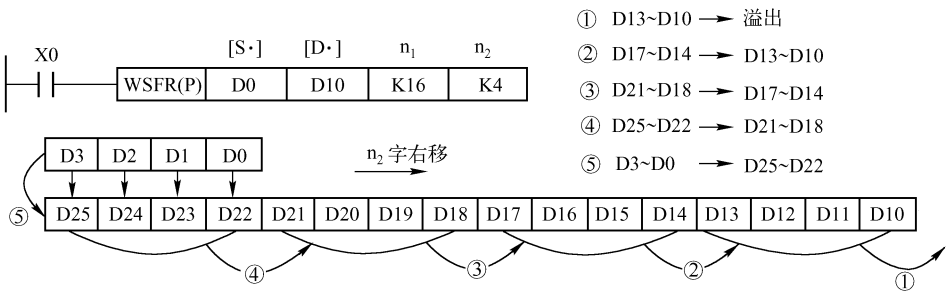


图 4-31　WSFR（P）指令应用示例

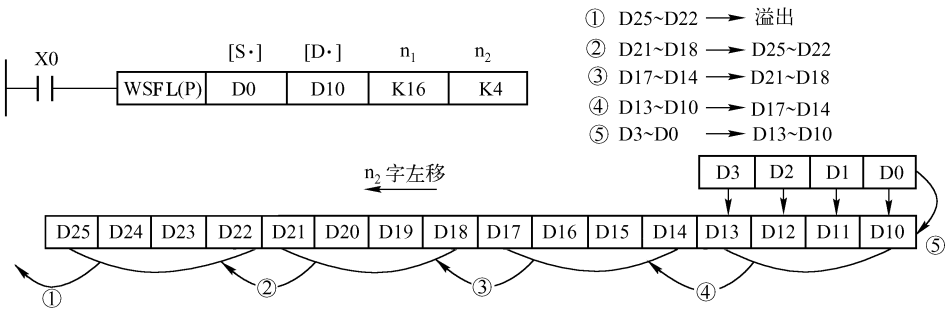


图 4-32　WSFL（P）指令应用示例

4.5.5　先入先出（FIFO）写入

SFWR　FNC 38 （P）（16）（“!”） FIFO 写入	操作元件： [S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS n（$2 \leqslant n_1 \leqslant 512$） 程序步数：SFWR，SFWR（P）……7 步 标　　志：M8022（进位）

如图 4-33 所示，当 X0 变为 ON 时，在源 D0 中的数据写入 D2，而 D1 变为 1（指针），D0 的数据改变后，X0 再变为 ON，则 D0 数据写入 D3，D1 中数据变为 2。依此类推，源 D0 中的数据依次写入寄存器。

数据从最右边的寄存器开始顺次存入，源数据写入的次数（指针）存 D1。当（D1）的内容达到“n→1”后，上述处理不再执行，进位标志 M8022 置 1。

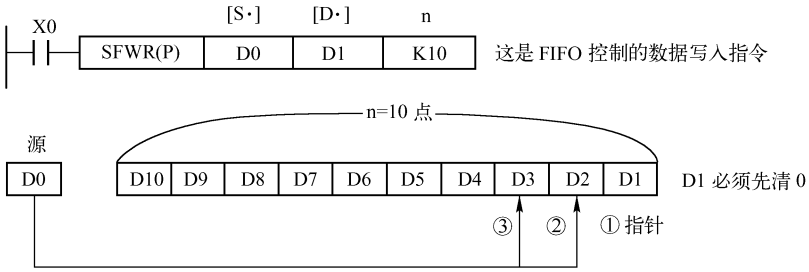


图 4-33　SFWR（P）指令应用示例

4.5.6　先入先出（FIFO）读出

SFRD　FNC 39 （P）（16）（“!”） FIFO 读出	操作元件： [S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：K_nY　K_nM　K_nS n（$2 \leqslant n_1 \leqslant 512$） 程序步数：SFRD，SFRD（P）……7 步 标　　志：M8022（进位）

如图 4-34 所示，X1 从 OFF→ON 时，D2 中的数据送到 D20，同时，指针（D1）减 1，从 D3 到 D10 的数据向右移 1 字。若用连续指令 SFRD，则每个扫描周期数据右移 1 字。

数据总是从 D2 读出。当指针 D1 为 0，不再执行上述处理。零标志 M8020 置 1。

执行本指令的过程中，D10 的数据保持不变。

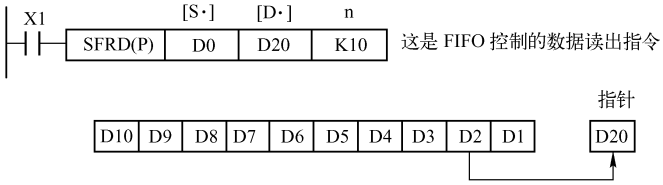


图 4-34　SFRD（P）指令应用示例

4.6　数据处理（FNC40 ~ FNC 49）

4.6.1　区间复位

ZRST　FNC 40 （P）（16） 区间复位	操作元件： K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D1·][D2·] X　Y　M　S [D1·][D2·] ［D1・］号≤［D2・］号 指定同一元件 程序步数：ZRST 和 ZRST（P）……5 步

区间复位指令应用示例如图 4-35 所示。

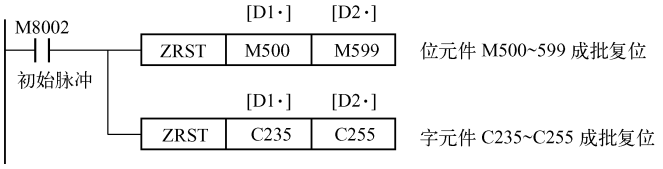


图 4-35　ZRST（P）指令应用示例

［D1・］和［D2・］指定的应为同类元件。［D1・］指定的元件号应小于等于［D2・］指定的元件号。如［D1・］号 >［D2・］号，则只有［D1・］指定的元件被复位。

虽然 ZRST 作 16 bit 指令处理，［D1・］、［D2・］也可同时指定 32 bit 计数器。

注意：［D1・］［D2・］一个指定 16 bit 而另一个指定 32 bit 计数器是不允许的。

4.6.2　解码

DECO　FNC 41 （P）（16/32） 解码	操作元件： [S·] K, H　T　C　D　V, Z [D·] n（n=1~8） X　Y　M　S [D·] 程序步数：DECO 和 DECO（P）……7 步

解码指令应用示例如图 4-36 所示。

若［D・］指定的目标是 T、C 或 D，应使 n≤4。目标元件的每 1 bit 都受控。

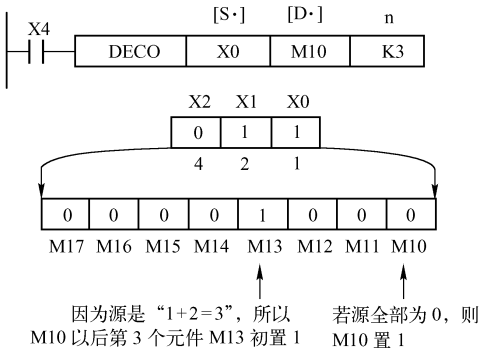


图 4-36　DECO 指令应用示例

4.6.3　编码

ENCO　FNC 42 （P）（16） 编码	操作元件： [S·]：K, H　T　C　D　V, Z　　X　Y　M　S [D·]：n（n=1~8） 程序步数：ENCO 和 ENCO（P）……7 步

编码指令应用如图 4-37 所示。

若［S・］指定的源是 T、C、D、V 或 Z，应使 n≤4。若指定源中为“1”的 bit 不止一处，则只有最高 bit 的“1”有效。

若指定源中所有 bit 均为 0，则出错。

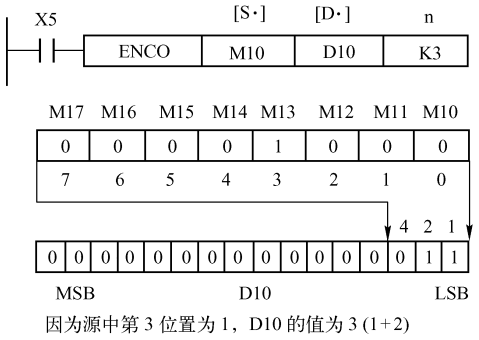


图 4-37　ENCO 指令应用示例

4.6.4　ON 总数

SUM　FNC 43 （P）（16/32） ON bit 总数	操作元件： [S·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]： 程序步数：SUM 和 SUM（P）……7 步 （D）SUM 和（D）SUM（P）……9 步 标　　志：M8020（零）

ON 总数指令应用如图 4-38 所示，D0 中“1”的总数存入 D2。若 D0 中没有为“1”的 bit，则零标志 M8020 置 1。

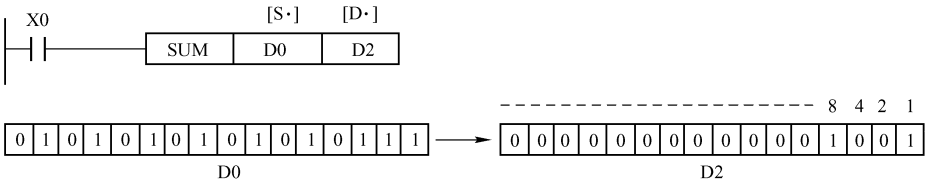


图 4-38　SUM（P）指令应用示例

用到（D）SUM 和（D）SUM（P）指令时，D3 中所有各 bit 均为“0”，因为在 D0 和 D1 32bit 中“1”的总数存入了 D2。

4.6.5　ON 位判别

BON　FNC 44 （P）（16/32） ON 位判别	操作元件： [S·]：K, H ｜ K_nX ｜ K_nY ｜ K_nM ｜ K_nS ｜ T ｜ C ｜ D ｜ V, Z n（n=0~15 或 0~31） [D·]：X ｜ Y ｜ M ｜ S 程序步数：BON，BON（P）……7 步 （D）BON，（D）BON（P）……13 步

ON 位判别指令应用如图 4-39 所示，若 D10 中的第 15bit 为 ON，则 M0 变为 ON。即使 X0 变为 OFF，M0 亦保持不变。

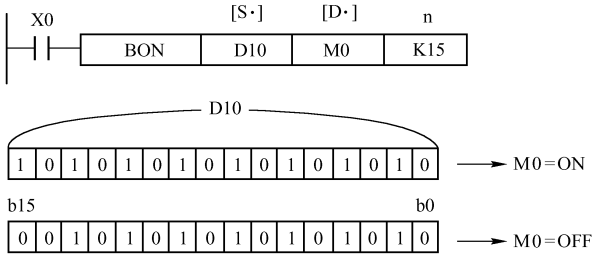


图 4-39　BON 指令应用示例

n 的范围如下：

16 bit 操作：n = 0 ~ 15

32 bit 操作：n = 0 ~ 31

4.6.6　平均值

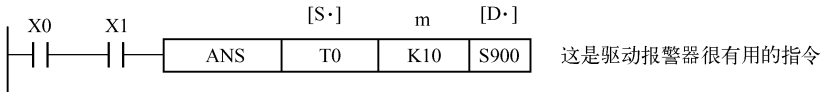
MEAN　FNC 45 （P）（16/32） 平均值	操作元件： [S·]：K, H ｜ K_nX ｜ K_nY ｜ K_nM ｜ K_nS ｜ T ｜ C ｜ D ｜ V, Z [D·] n（n=1~64） 程序步数：MEAN，MEAN（P）……7 步 （D）MEAN，（D）MEAN（P）……13 步

n 个源数据的平均值送到指定目标。平均值是指 n 个源数据的代数和被 n 除所得的商，余数略去。若元件超出范围，n 的值会自动缩小，以取允许范围内元件的平均值。

若指定的 n 值超出 1 ~ 64 的范围，则出错。

4.6.7 报警器置位

ANS FNC 46 (16) 报警器置位	操作元件：[S·]：T0 ~ T199，[D·]：S900 ~ S999 m = 1 ~ 32 767（100 ms 单位） 程序步数：ANS……7 步

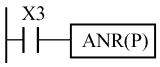


若 X0 和 X1 同时为 ON 超过 1s，S900 置 1。S900 置 1 后，若 X0 和/或 X1 变为 OFF，则定时器复位而 S900 保持为 1。

若 X0 或 X1 在 1s 内再变为 OFF，则定时器复位。

4.6.8 报警器复位

ANR FNC 47 (P)(16)(“!”) 报警器复位	操作元件：无 程序步数：ANR，ANR（P）……1 步



当 X3 变为 ON 时，S900 ~ S999 之间被置 1 的报警器复位。

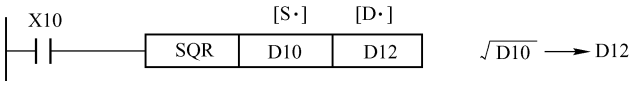
若超过一个报警器被置 1，则元件号最低的那个报警器被复位。

X3 再次变为 ON 时，下一个被置 1 的报警器复位。

注：若用连续指令 ANR，则按扫描周期依次逐个地将报警器复位。

4.6.9 开平方

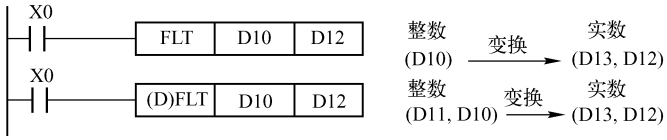
SQR FNC 48 (P)(16/32) 开平方	操作元件： [S·]：K, H；K_nX；K_nY；K_nM；K_nS；T；C；D；V, Z [D·]：D 程序步数：SQR……5 步；（D）SQR……9 步 标　　志：M8020（零标志），M8021（借位标志）



整数开平方运算。[S·] 必须为正数，如为负数，则出错标志 M8067 ON，指令不执行。运算结果舍去小数，取整后送到目标。舍去小数时借位标志 M8021 ON。若结果为零，则零标志 M8020 ON。

4.6.10 整数－实数变换

FLT FNC 49 （P）（16/32） 整数－实数变换	操作元件： [S·] K, H | K_nX | K_nY | K_nM | K_nS | T | C | D | V, Z [D·] 程序步数：FLT……5 步；（D）FLT……9 步



将 BIN 整数变换为实数，常数 K，H 在实数运算中会自动转换，故无须使用本指令。逆变换指令即实数变换成整数的指令为 FNC129 INT。

图 4-40、图 4-41 所示是一个 BIN 整数与实数之间变换及实数四则运算的例子。

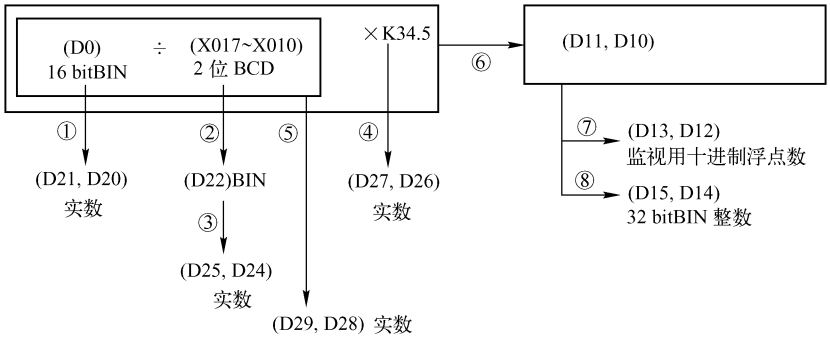


图 4-40　BIN 整数与实数之间变换的例子

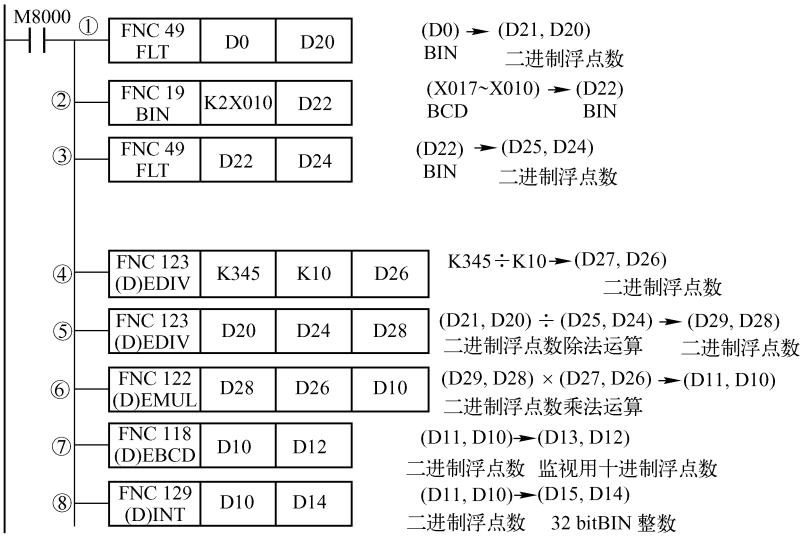


图 4-41　实数四则运算的例子

4.7 高速处理（FNC 50 ~ FNC 59）

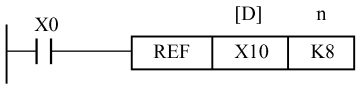
4.7.1 刷新

REF FNC 50 （P）（16）（“!”） 刷新	操作元件：［D］……最低位为 0 的 X 和 Y 元件，如 X0、X10、X20 等 n…… K 或 H 的参数，8 的倍数 程序步数：REF，REF（P）……5 步

如前所述，FX_{2N}型 PLC 用 I/O 批处理的方法，即输入状态是在程序处理之前成批读入到输入映像寄存器的；而输出数据是在 END 指令执行后由输出映像寄存器通过输出锁存器输出到输出端子的。

本指令用于在某段程序处理时开始读入最新输入信息或用于在某一操作结束之后立即将操作结果输出。

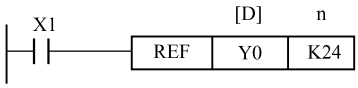
1. 输入刷新



仅限于 X10 ~ X17 的 8 点输入刷新。

若在本指令执行之前 X10 ~ X17 已经变为 ON 约 10 ms（滤波器响应延迟时间），则执行本指令时 X10 ~ X17 的映像寄存器会变为 ON。

2. 输出刷新



对 Y0 ~ Y7、Y10 ~ Y17、Y20 ~ Y27 的 24 点输出刷新。

执行本指令时，与输出 Y0 ~ Y27 对应输出锁存器的状态立即输出到输出端子。

在输出响应延迟时间后输出触点动作。

4.7.2 刷新和滤波时间调整

REFF FNC 51 （P）（16） 刷新和滤波时间调整	操作元件：n K 或 H 的参数：（n = 0 ~ 60）对 X0 ~ X17 （无需指定）数字滤波时间调整（滤波时间：0 ~ 60 ms） 程序步数：REFF，REFF（P）……3 步

在 FX 系列 PLC 中，X0 ~ X7 采用了数字滤波器，使用 REFF 指令可调整其滤波时间。执行 REFF K10 指令时，输入滤波时间为 10 ms。对执行 REFF K20 指令以后的程序步，输入滤波时间为 20 ms。

一般来说，为防止输入噪声的影响，PLC 的输入 RC 滤波时间常数约为 10 ms。

但电子固态（无触点）开关没有抖动噪声，可以高速输入。对于这类高速输入，上述滤波器又成了高速的障碍。当 n = 0 时，实际滤波时间为 50 μs。因为输入接口电路有 RC 滤波网络，所以最小滤波时间不小于 50 μs。

使用 REFF 指令时，应注意以下几点：

1）当 X0 ~ X17 用于高速计数输入或使用速度检测指令以及中断输入时，输入滤波器的

时间常数自动设置为 50 μs。

2）X0 ~ X17 的输入滤波时间初始值（10 ms）存于 D8020 中，推广 MOV 指令可以改写 D8020 中的数值，从而改变 X0 ~ X17 输入滤波时间。

4.7.3 矩阵输入

MTR FNC 52 （16） 矩阵输入	操作元件：［S］：X0、X10、X20 等最低位为 0 的 X ［D1］：Y0、Y10、Y20 等最低位为 0 的 Y ［D2］：Y、M、S 等最低位取 0 n：K、H、n = 2 ~ 8 程序步数：MTR：9 步 标　　志：M8029（完成）

利用 MTR 指令，可用连续排列的 8 点输入与 n 点输出组成 8 列 n 行的输入矩阵，用来输入 n × 8 个开关量信号。指令处理时间为 n × 20 ms。如果用高速输入 X0 ~ X17 作输入点，则读入时间减半。MTR 指令使用格式如图 4-42 所示。

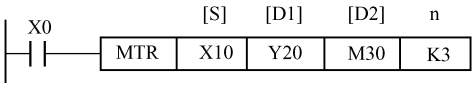


图 4-42　MTR 指令应用示例

如图 4-42 所示，由［S］指定的输入点开始的 8 个输入（这里为 X0 ~ X17）和由［D1］指定的输出开始的 n 个晶体管输出点（这里为 n = 3，即 Y20、Y21、Y22）反复顺次接通；当 Y20 为 ON 时，读入第一行的输入状态，存于 M30 ~ M37，当 Y22 为 ON 时，读入第二行的输入状态，存于 M40 ~ M47，依此类推，如此反复执行。

标志 M8029 在第一个读入周期完成后即置 1。当执行条件（X0）变为 OFF 时，M8029 复位（为 0），M30 ~ M57 的状态保持不变。

4.7.4 高速计数器置位

HSCS FNC 53 （32） 高速计数器置位	操作元件： [S1·]：K, H ; K_nX ; K_nY ; K_nM ; K_nS ; T ; C ; D ; V, Z [S2·]：C235 ~ C255（高速计数器） [D·]：X ; Y ; M ; S 程序步数：（D）HSCS……13 步

HSCS 指令可使置位和输出以中断方式立即执行，即高速计数器以中断方式对相应输入脉冲的下降沿的个数计数。当计数器当前值达到预置值时，计数器的输出触点立即动作。

图 4-43 中当 C255 的当前值由 99 变为 100 或由 101 变为 100 时，Y10 立即置 1。

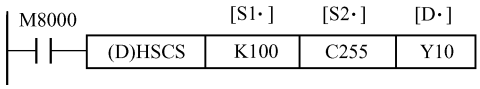


图 4-43　HSCS 指令应用示例

4.7.5 高速计数器复位

HSCR FNC 53 (32) 高速计数器置位	操作元件：[S1·] K, H | K_nX | K_nY | K_nM | K_nS | T | C | D | V, Z [S2·]：C235~C255（高速计数器） X | Y | M | S [D·] 对于［D·］，可以像指定［S2·］一样指定相同的高速计数器 程序步数：(D) HSCR……13 步

图 4-44 中，C255 的当前值由 199 变为 200 或由 201 变为 200 时，Y10 立即复位。

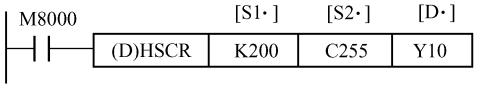


图 4-44 HSCR 指令应用示例

4.7.6 高速计数器区间比较

HSZ FNC 55 (16/32) HSC 区间比较	操作元件：[S1·][S2·] K, H | K_nX | K_nY | K_nM | K_nS | T | C | D | V, Z [S·]：C235~C255（高速计数器） X | Y | M | S [D·] 程序步数：(D) HSZ……17 步

高速计数器区间比较指令 HSC 有三种工作模式：标准模式、多段比较模式和频率控制模式，详细的使用方法请参阅 FX 系列编程手册。

4.7.7 速度检测

SPD FNC 56 (16) 速度检测	操作元件：[S2·] K, H | K_nX | K_nY | K_nM | K_nS | T | C | D | V, Z [D·] [S1·] X0~X5 程序步数：SPD……7 步 指 令 数：X0 ~ X5 每个输入点一条命令以下

速度检测指令 SPD 用来检测在指定时间内从编码器输入的脉冲个数，并计算出速度。

4.7.8 脉冲输出

PLSY FNC 57 (16/32) 脉冲输出	操作元件：[S1·][S2·] K, H | K_nX | K_nY | K_nM | K_nS | T | C | D | V, Z [D·]：Y 程序步数：PLSY……7 步，(D) PLSY……13 步 标　　志：M8029（完成）

PLSY 指令用于产生指定数量的脉冲。它的应用示例如图 4-45 所示。

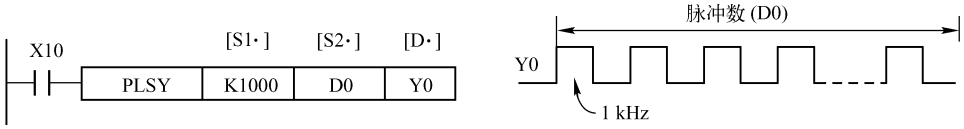


图 4-45　PLSY 指令应用示例

［S1 · ］指定脉冲频率：

FX_{2N}，FX_{2NC}：2 ~ 20 000（Hz）

FX_{1S}，FX_{IN}：16 位指令：1 ~ 32 767（Hz）

32 位指令：1 ~ 100 000（Hz）

在指令执行中更改［Sl · ］指定的字软元件的内容，输出频率也随之发生改变。

［S2 · ］指定产生脉冲的数目。脉冲数范围：16 bit 指令：1 ~ 32 767 个脉冲；32 bit 指令：1 ~ 2 147 483 647个脉冲。

若指定脉冲数为“0”，则产生无穷多个脉冲，即连续脉冲输出。

［D · ］指定脉冲输出元件号。

脉冲占空比为 50%。

脉冲以中断方式输出。

4.7.9　脉宽调制

PWM　FNC 58 （16） 脉宽调制	操作元件： [S1·][S2·]：K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z [D·]：Y　仅限 Y0、Y1 程序步数：PWM……7 步

PWM 指令应用示例如图 4-46 所示。

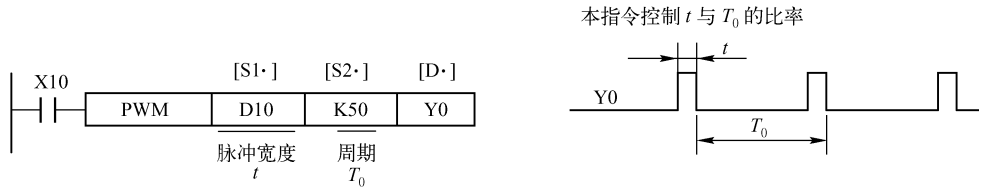


图 4-46　PWM 指令应用示例

［S1 · ］指定脉冲宽度，范围 0 ~ 32 767 ms。

［S2 · ］指定周期 T0，范围 1 ~ 32 767 ms。

［D · ］指定脉冲输出 Y 的元件号。输出的 ON/OFF 状态用中断方式控制。

Y 元件只能用 Y0 与 Y1，而且必须用晶体管输出方式。

4.7.10 带加减速的脉冲输出

PLSR FNC 59 (16/32) 带加减速的脉冲输出	操作元件：[S1·][S2·][S3·] K, H | K_nX | K_nY | K_nM | K_nS | T | C | D | V, Z X | Y | M | S [D·] 仅限于 Y0、Y1 程序步数：PLSR……7 步（D）PLSR……17 步 标　　志：M8029（结束标志）

带加速减速功能的脉冲输出指令，按［S1·］指定的最高频率分 10 级加速，达到［S2·］所指定的输出脉冲数后分 10 级减速。

PLSR 指令应用示例如图 4-47 所示。

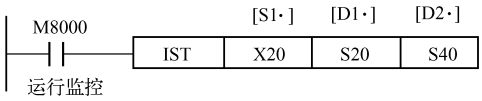
图 4-47　PLSR 指令应用示例

4.8 方便指令（FNC 60 ~ FNC 69）

4.8.1 置初始状态

IST FNC 60 (16) 置初始状态	操作元件：[S·] X | Y | M | S [D1·]：S20~S899 [D2·]：S20~S899 [D1·]< [D2·] 程序步数：IST……7 步

本指令用于自动设置初始状态和特殊辅助继电器。



［S·］指定操作方式输入的首元件。上例中：

X20：手动　　X24：全自动运行

X21：回原点　　X25：回原点启动

X22：单步运行　　X26：自动操作启动

X23：单周期运行（半自动）　　X27：停止

［D1·］指定在自动操作中实际用到的最低状态号。

［D2·］指定在自动操作中实际用到的最高状态号。

本指令执行条件变为 ON 时，下列元件自动受控；其后若执行条件变为 OFF，这些元件的状态仍保持不变。

M8040：禁止转移　　S0：手动操作初始状态

M8041：传送开始　　S1：回原点初始状态

M8042：起始脉冲　　S2：自动操作初始状态

M8047：STL 监控使能

本指令只能使用 1 次。

若不用 IST 指令，状态 S10 ~ S19（用于回原点）可作通用状态。只是在这种情况下，仍需将 S0 ~ S9 作为初始化状态，只是 S0 ~ S2 的用途是自由的。

编程时，IST 指令必须写在 STL 指令之前，即在 S0 ~ S2 出现之前。

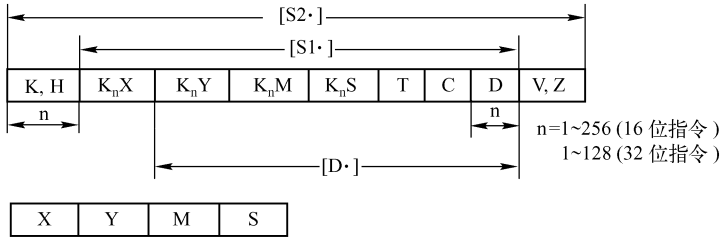
输入 X20 ~ 524 必须用旋转选择开关，以保证这组输入中不可能有两个输入同时为 ON。

若在回原点完成标志 M8043 未置 1 之前改变操作方式选择，则所有输出将变为 OFF。

4.8.2　数据搜索

SER　FNC 61
（P）　（16/32）

操作元件：



程序步数：SER……7 步；（D）SER……17 步

本指令在指定的范围内搜索相同值、最大值及最小值，并将结果存入指定单元中。

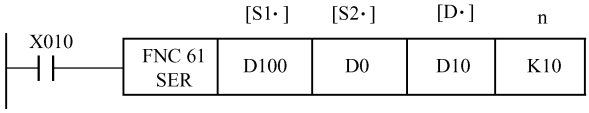


表 4–4 所示为检索表的构成和数据示例，检索结果见表 4–5。

表 4–4　检索表的构成和数据示例

被检索元件	被检索数据示例	比较数据	数据的位置	最大值	相同	最小值
D100	（D100）= K100	D0 = K100	0		相同	
D101	（D101）= K111		1			
D102	（D102）= K100		2		相同	
D103	（D103）= K98		3			
D104	（D104）= K123		4			
D105	（D105）= K66		5			最小
D106	（D106）= K100		6		相同	
D107	（D107）= K95		7			
D108	（D108）= K210		8	最大		
D109	（D109）= K88		9			

[S1 ·] 指定起始元件序号

[S1 ·] 指定的元件顺序

N 指定被检索数据个数

[S2 ·] 指定比较数据

表 4–5　检索结果

元 件 号	内容	备　　注
D10	3	相同数据个数
D11	0	相同数据位置(初始)
D12	6	相同数据位置(最终)
D13	5	最小值最终位置
D14	8	最大值最终位置

[D·] 指定检索结果存放的起始元件号

占用连续 5 点

进行代数上的大小比较（ –10 <2）。

最小值、最大值有多个时，显示后面的位置。

（D）SER 指令中［D·］使用双字节（数据寄存器对）。

在从［S1·］起始的 n 个元件中，检索与［S2·］数据相同的数据，并将其个数存入［D·］中。在以［D·］为起始的 5 个元件中，见表 4–4，存入相同数据个数及最小值、最大值的位置。不存在相同数据时，上例中（D10）～（D12）=0。

4.8.3　绝对值式凸轮顺控

ABSD　FNC 62 （16/32） 绝对值式凸轮顺控	操作元件： [S1·] 奇数 K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z n≤64　X,Y,M 和 S 为 8 的倍数　[S2·] X　Y　M　S [D·] 程序步数：ABSD……9 步；（D）ABSD……17 步

装在机械转轴上的编码器给 PLC 的计数器提供角度位置脉冲，本指令产生一组对应于计数值的变化的输出波形，用来控制最多 64 个输出变量（Y、M 和 S）的 ON/OFF。

4.8.4　增量式凸轮顺控

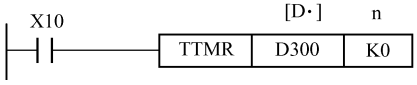
INCD　FNC 63 （16） 增量式凸轮顺控	操作元件： [S1·] K, H　K_nX　K_nY　K_nM　K_nS　T　C　D　V, Z n≤64　X,Y,M 和 S 为 8 的倍数　[S2·] X　Y　M　S [D·] 程序步数：INCD……9 步 标　　志：M8029（完成）

增量式凸轮顺控本指令 INCD 根据计数器对位置脉冲的计数值，实现对最多 64 个输出变量（Y、M 和 S）的循环顺序控制，使它们依次为 ON，并且同时只有一个输出变量为 ON。

4.8.5 示教定时器

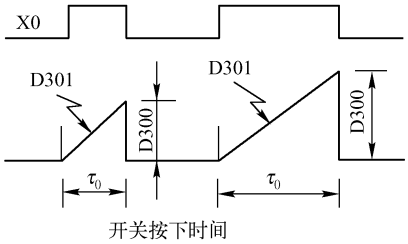
TTMR FNC 64 (16) 示教定时器	操作元件：[D·]：D n：K, H　n=0~2 程序步数：PWM……5 步

使用 TTMR 指令可以用一只按钮调整定时器的设定值。它的应用示例如图 4-48 所示，按钮 X10 按下的时间由 D301 记下。该时间乘以“n”指定值并存入 D300。使用这条指令可利用按钮调整定时器的设定值。



按钮 X10 按下时间即为“0”(s)，存入 D300 的值按“n”指定值而变化，如下所示：

n	D300
K0	τ_0
K1	$10\tau_0$
K2	$100\tau_0$



当 X0 关断时，D301 复位，D300 保持不变。

图 4-48　TTMR 指令应用示例

4.8.6 特殊定时器

STMR FNC 65 (P) (16) 特殊定时器 (Special Timer)	操作元件：[S·]：T　T0~T199 (100 ms 定时器) [D·]：Y M S m：K, H　m=1~32767 程序步数：STMR……7 步

STMR 指令产生延时断定时器、单脉冲式定时器和闪动定时器，它的应用如图 4-49 所示。

“m”指定值是［S·］指定的定时器的设定值。

M0 是延时断定时器，M1 是单脉冲式定时器，M2、M3 是为闪动而设的，如图 4-50 所示的例子。

M3 接法如图 4-50 所示时，M2 和 M1 产生闪动输出。当 X0 关断时，M0、M1、M3 在经过设定时间后关断，T10 同时复位。

在其他程序中不要用这里已使用的定时器。

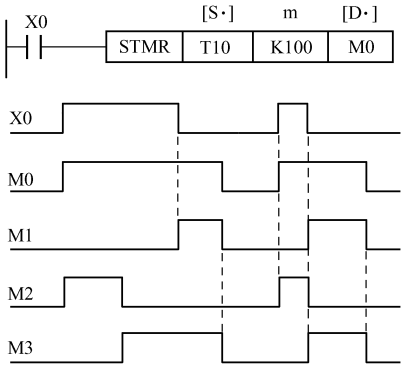


图 4-49　STMR 指令应用示例

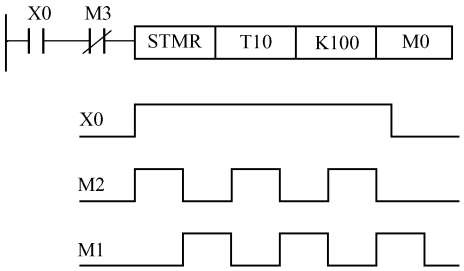


图 4-50　延时断定时器应用示例

4.8.7 交替输出

ALT FNC 66 (P)(16)("!") 交替输出 (Alternate)	操作元件：S Y M S [D·] 程序步数：ALT，ALT（P）……3 步

每次 X0 从 OFF 到 ON 时，M0 的状态改变一次，如图 4-51 所示。若用连续指令 ALT 时，M0 的状态每个扫描周期改变一次。

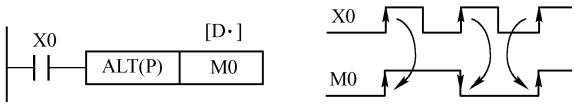


图 4-51 ALT 指令应用示例

ALT 指令具有分频器的效果，使用 ALT 指令，用一只按钮 X0 就可以控制 Y0 对应的外部负载的起停。

4.8.8 斜坡信号

RAMP FNC 67 (16) 斜坡信号 (Ramp)	操作元件：[S1·]，[S2·]，[D·]：D n：K，H　n = 1 ~ 32767 程序步数：RAMP……9 步 标　志：M8029（完成）

RAMP 指令与模拟量输出结合可实现软启动/软停止。设置斜坡输出信号的初始值和最终值后，执行该指令时输出数据由初始值逐渐变为最终值，变化全过程所需的时间用扫描周期的个数来设置。

4.8.9 旋转台控

ROTC FNC 68 (16) 旋转台控制 (Rotary Table Control)	操作元件：[S·]：D m_1、m_2：K，H 程序步数：ROTC……9 步

旋转工作台工作情况如图 4-52 所示，分为 10 个位置。ROTC 指令用来控制工作台的旋转，使得被指定的工件以最短路径转到出口位置。

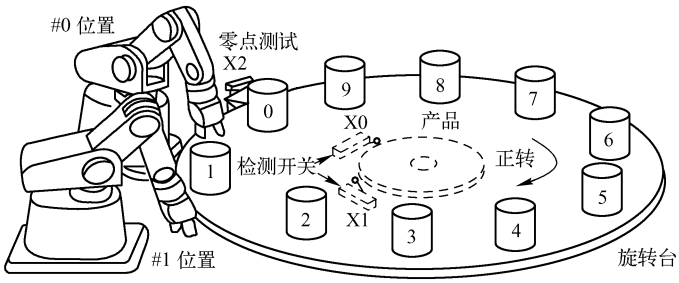


图 4-52 旋转工作台工作情况

4.8.10 数据排序

SORT FNC 69 (16) 数据排序 (Sort)	操作元件： [S][D] K, H | K_nX | K_nY | K_nM | K_nS | T | C | D | V, Z n, m1, m2 指令使用次数：1 次 m1：1~32 m2：1~6 n：1~m2 X | Y | M | S 程序步数：SORT……17 步 标　　志：M8029（完成）

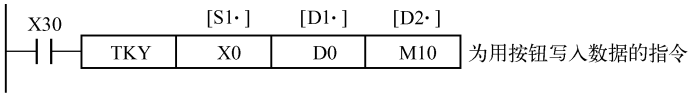
该指令把源数据表格 n 指定的列中的数据按由小到大的顺序重新排列行序后送到目标数据表格中存放。［S］和［D］分别指定源和目标数据表格的首个数据寄存器。表格有 m1 行，m2 列，因而占用 m1 × m2 个数据寄存器。

用 X010 = ON 开始数据排序，执行完毕后标志 M8029 = ON。动作中请不要改变操作数与数据的内容。再运行时，请将 X010 置于 OFF 一次。

4.9 外部 I/O 设备（FNC 70 ~ FNC 79）

4.9.1 10 键输入

TKY FNC 70 (16/32) 10 键 (Ten Key)	操作元件： [D1·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z [S·] X | Y | M | S [D2·] 程序步数：TKY……9 步；（D）TKY……17 步



［**例 4-1**］图 4-53 所示为以 X0 作输入首元件作数字键（0 ~ 9）的例子。

以①、②、③、④顺序按数字键，则 D0 存入数据 2130。若送入的数据大于 9 999，则高位数溢出并丢失（数据以二进制码形式存于 D0）。

用（D）KEY 指令时，D1 和 D0 组对使用，大于　99 999 999 的数据溢出。

当 X2 按下后，M12 置 1 并保持至另一键被按下，其他键亦一样。

M10 ~ M19 的动作对应于 X0 ~ X11。

任一键被按下，键信号 M20 置 1 直到该键放开。

当 2 个或更多的键被按下，则首先按下的键有效。

X30 变为 OFF 时，D0 中数据保持不变。但 M10 ~ M20 全部变为 OFF。

本指令只能用一次。

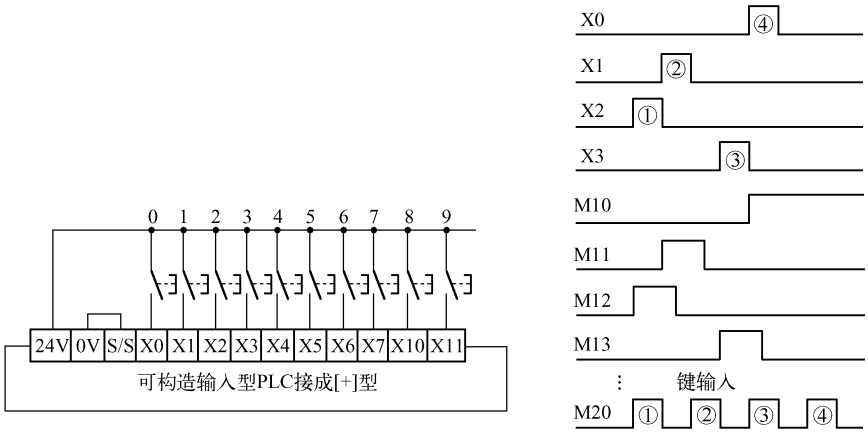
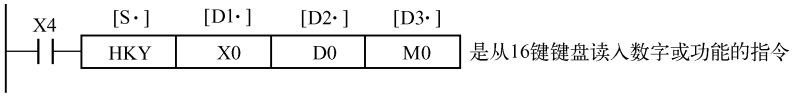


图 4-53　以 X0 作输入首元件作数字键（0～9）的例子

4.9.2　16 键输入

HKY FNC 71 （16/32） 16 键输入 （Hexa Decimal Key）	操作元件： [D2·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D3·]：X　Y　M　S [S·] [D1·] 程序步数：HKY……9 步；（D）HKY……17 步 标　　志：M8029（完成）



［**例 4-2**］图 4-54 所示为从 16 键键盘读入数字的实例。

（1）数字键（见图 4-54a）　输入的 0～ 9 999 数字以 BIN 码存在 D 0，大于 9 999 的数溢出。

用（ D）HKY 指令时，可在 D1 和 D 0 中存放 0～ 99 999 999 的数。

多个键同时按下时，最先按下的键有效。

按下键被扫描到后，标志 M8029 置 1。

（2）功能键（见图 4-54b）　按下 A 键，M0 置位并保持。按下 D 键，则 M0 置 0，M3 置 1 并保持。依此类推，同时按下多个键，先按下的有效。

HKY 指令只能用一次。

（3）键扫描输出（见图 4-54c）　A～F 的任一个键被按下时，M6 置 1（不保持输入1～9 键被按下时，M7 置 1（不保持）。

当 X4 变为 OFF 时，D0 保持不变，M0～M7 全部 OFF。

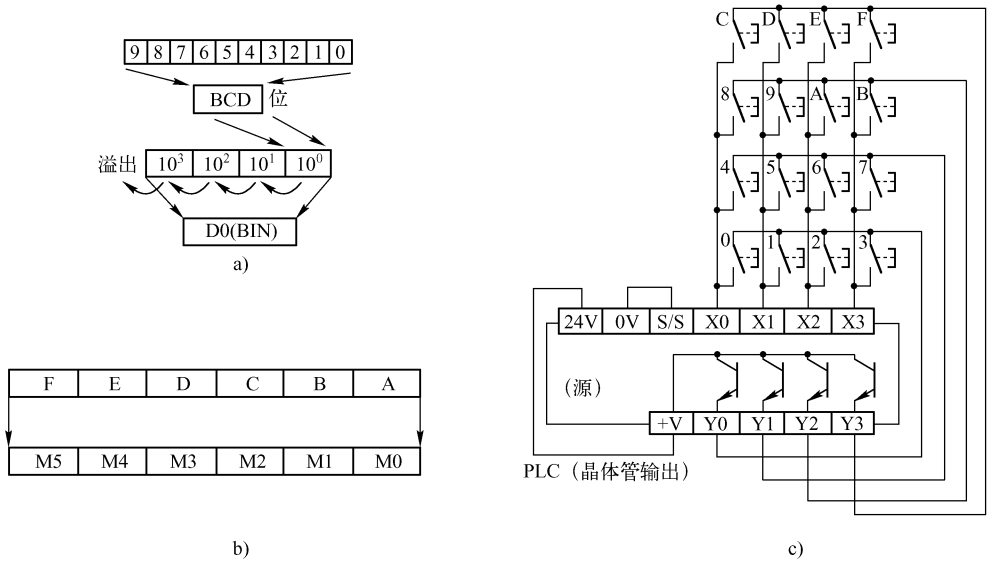
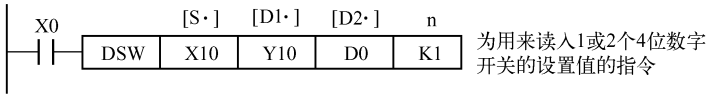


图 4-54　从 16 键键盘读入数字的实例

4.9.3　数字开关

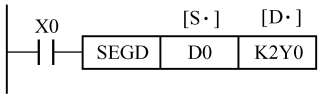
DSW FNC 72 （16） 数字开关 （Digital Switch）	操作元件： [D·] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z n=1,2 X　Y　M　S [S·] [D·] 程序步数：DSW……9 步 标　　志：M8029（完成）



DSW 指令用于读入一组或两组 4 位 BCD 码数字拨盘开关的设置值，占用 PLC 的 4 个或 8 个输入点和 4 个输出点。

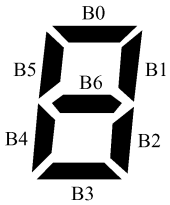
4.9.4　7 段解码

SEGD FNC 73 （P）（16） 7 段解码 （Seven Segment Decoder）	操作元件： [S·] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D·] 程序步数：SEGD，SEGD（P）……5 步



［S·］指定元件的低 4 bit 所确定的十六进制数（0～F）经解码驱动 7 段显示器。解码信号存于［D·］指定元件。［D·］的高 8 bit 不变。段解码表见表 4–6。

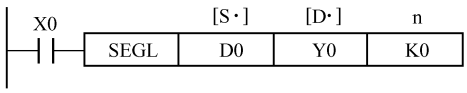
表 4–6　段解码表

［S·］		7 段码构成	［D·］								显示数据
十六进制	二进制		B7	B6	B5	B4	B3	B2	B1	B0	
0	0000		0	0	1	1	1	1	1	1	0
1	0001		0	0	0	0	0	1	1	0	1
2	0010		0	1	0	1	1	0	1	1	2
3	0011		0	1	0	0	1	1	1	1	3
4	0100		0	1	1	0	0	1	1	0	4
5	0101		0	1	1	0	1	1	0	1	5
6	0110		0	1	1	1	1	1	0	1	6
7	0111		0	0	1	0	0	1	1	1	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	0	1	1	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	1	1	1	1	1	0	0	b
C	1100		0	0	1	1	1	0	0	1	C
D	1101		0	1	0	1	1	1	1	0	d
E	1110		0	1	1	1	1	0	0	1	E
F	1111		0	1	1	1	0	0	0	1	F

B0 代表 bit 元件的首位（本例中为 Y0）和字元件的最低位。

4.9.5　带锁存的 7 段显示

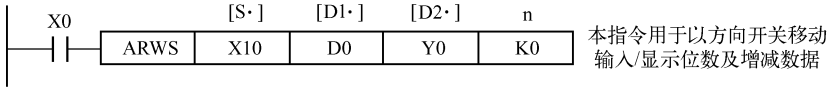
SEGL FNC 74 （P）（16） 带锁存的 7 段显示	操作元件： [S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z n=0~7 [D·]：X　Y　M　S 程序步数：SEGL……7 步 标　　志：M8029（完成）



SEGL 指令用 12 个扫描周期显示一组或两组 4 位数据，占用 8 个或 12 个晶体管输出点。

4.9.6 方向开关

ARWS FNC 75 (16) 方向开关 (Arrow Switch)	操作元件： [D1·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z n=0~3 [S·]：X　Y　M　S [D2·] 程序步数：ARWS……9 步



ARWS 指令用方向开关（4 只按钮）来输入 4 位 BCD 数据，输入的数据用带锁存的 7 段显示器来显示。输入数据时，用左移、右移开关来移动要修改和显示的位，用加、减开关来增减该位的数据。该指令占用 4 个或 8 个输出点。

4.9.7 ASCII 码变换

ASC FNC 76 (16) ASCII 变换 (ASCII Code)	操作元件：[S]：多至 8 个字符，从 A6GPP（A6PHP）键入 [D]：T　C　D 程序步数：ASC……7 步

下面的程序是将字母 A ~ H 换成 ASCII 码并送到 D300 ~ D303，如图 4-55 所示。

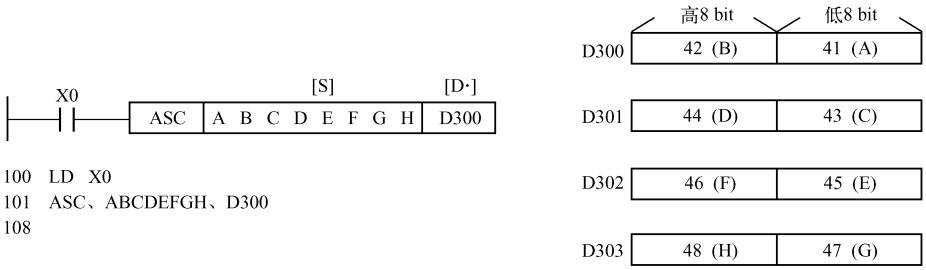


图 4-55　ASCII 码转换示例

ASC 指令适合于用外部显示单元显示出错信息。

表 4-7 所示是数字、字母与相应的 ASCII 码对应关系（以十六进制表示）。

表 4-7　数字、字母与相应 ASCII 码对应关系

数字	ASCII	字母	ASCII	字母	ASCII	字母	ASCII
0	30	A	41	K	4B		
1	31	B	42	L	4C		
2	32	C	43	M	4D	U	55
3	33	D	44	N	4E	V	56
4	34	E	45	O	4F	W	57
5	35	F	46	P	50	X	58
6	36	G	47	Q	51	Y	59
7	37	H	48	R	52	Z	5A
8	38	I	49	S	53		
9	39	J	4A	T	54		

4.9.8 打印

PR FNC 77 (16) 打印 (Print)	操作元件：[S·] T C D [D·] Y 程序步数：PR……5 步

PR 指令用于 ASCII 码的打印输出，常与 ASC 指令配合使用。PR 指令的使用说明如图 4-56 所示。若 D300～D303 中分别存放了字符串“ABCDEFGH”的 ASCII 码，输出 A～H 的 ASCII 码数据输到输出端 Y0～Y7，选通信号 Y10 和执行标志 Y11 也动作。

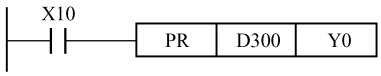


图 4-56 PR 指令应用示例

在指令执行过程中，若 X0 变为 OFF，送数操作停止。当 X0 再次 ON 时，从头开始送数。使用晶体管输出型 PLC。

4.9.9 读特殊功能模块

FROM FNC 78 (P)(16/32) 读特殊功能模块	操作元件：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z n,m1,m2　[D·] 程序步数：FROM、FROM (P) ……9 步 (D) FROM、(D) FROM (P) ……17 步； m1：0～7；m2：0～32767；n：1～32767

FROM 指令的使用如图 4-57 所示，它是将编号为 m1 的特殊功能模块内从缓冲寄存器（BFM）号为 m2 开始的 n 个数据读入基本单元，并存于从［D·］开始的 n 个数据寄存器中。

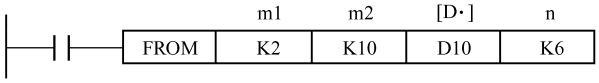
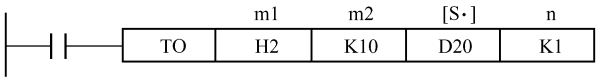


图 4-57 FROM 指令应用示例

接在 FX 系列 PLC 基本单元右边扩展总线上的功能模块，从最靠基本单元的那个开始，其编号依次为 0～7。n 是待传送数据的字数，n＝1～16（16 位操作）或 n＝1～32（32 位操作）。

4.9.10 写特殊功能模块

TO FNC 78 (P)(16/32) 写特殊功能模块	操作元件：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z n,m1,m2　[S·] 程序步数：TO、TO (P) ……9 步 (D) TO、(D) TO (P) ……17 步； m1：0～7；m2：0～32767；n：1～32767



上面的 TO 指令，是将基本单元从［S・］元件开始的 n 个字的数据，写到特殊功能模块 m1 中编号为 m2 开始的缓冲寄存器中。

m1：特殊功能模块号（范围 0～7）。

m2：缓冲寄存器首元件号（范围 0～32 767）。

n：待传送数据的字数（范围 1～32 767）。

4.10 外部设备 I/O 功能指令

4.10.1 串行数据传送

RS FNC 80 （16） 串行数据传送 RS 232C	操作元件： [S·] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z m,n　m,n [D·] m,n:0~4096 X　Y　M　S 程序步数：RS……9 步　　但请指定 n＋n≤8000 标　　志：M8029（零标志），M8021（借位标志），M8022（接收结束），M8129（停工超时标志）

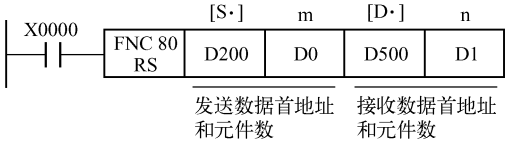


图 4-58　RS 指令应用示例

RS 指令的使用示例如图 4-58 所示。该指令为使用 RS－232－C 及 RS－485 功能扩展板及特殊适配器进行发送接收串行数据的指令。［S・］和 m 用来指定发送数据缓冲区的首地址和数据寄存器的个数，［D・］和 n 用来指定接收数据缓冲区的首地址和数据寄存器的个数。

数据的传送格式可以通过后面所述的特殊数据寄存器 D8120 设定。RS 指令驱动时，即使改变 D8120 的设定，实际上也不接受。

4.10.2 并联运行

PRUN FNC 81 （P）（16/32） 并联运行 FX2－AP/AW	操作元件：［S・］：K_nX，K_nM（n＝1～8）　指定元件的最低位为 0 ［D・］：K_nM，K_nY（n＝1～8） 程序步数：PRUN，PRUN（P）……5 步 （D）PRUN，（D）PRUN（P）……9 步

PRUN 指令利用 PLC 的并行链接适配器，把源数传送到指定的位元件区域，由专用标志 M 控制数据传送。当两台 FX PLC 已经链接，主站的标志 M8070 和从站的标志 M8071 都置为 ON 时，并行链接通信将自动进行，从站不需要为通信使用 PRUN 指令。

主站和从站都只有一台 PLC，主战和从站应分别用 M8000 的常开触点驱动 M8070 和 M8071 的线圈。一旦设置了站标志，它们只能在 PLC 进入 STOP 模式或上电时被清除。

在通信期间，主 PLC 和从 PLC 之间将自动交换图 4-59 中的数据。

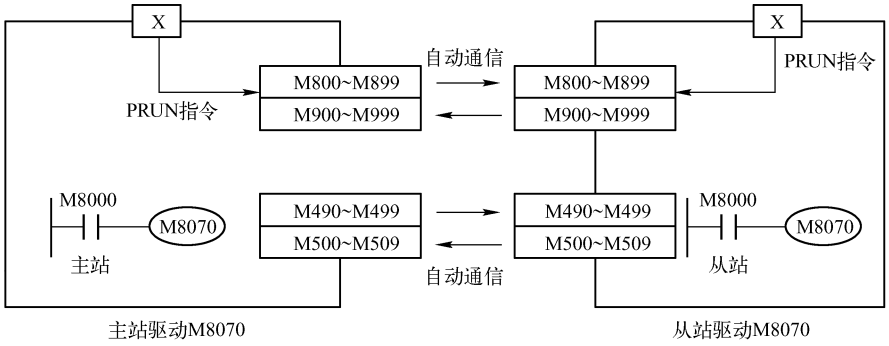


图 4-59　PRUN 指令应用示例

4.10.3　HEX→ASCII 变换

ASCI FNC 82 （P）（16） HEX→ ASCII 变换 HEX→ASCII	操作元件：[S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D·]：n:1~256　X　Y　M　S 程序步数：ASCI……7 步

ASCII 码变换指令 ACSI 使用示例如图 4-60 所示。

十六进制数变换为 ASCII 码存入目标寄存器中，M8161 = OFF 时执行，16 bit 变换模式（M8161 为 RS，HEX、CCD 指令共用的特殊标志）。

［S·］中的 HEX 数据的各位由低位到高位顺序转换成 ASCII 码，向［D·］的高 8 bit、低 8 bit分别传送。转换的字符数由 n 指定。

［D·］目标文件首地址分为低 8 位、高 8 位，存储 ASCII 数据。

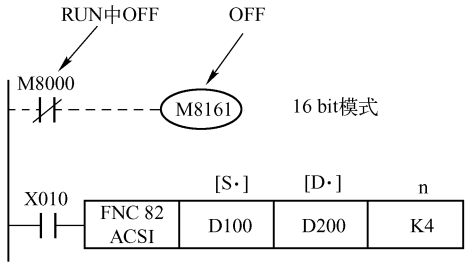


图 4-60　ACSI 指令应用示例

4.10.4　ASCII→HEX 变换

HEX FNC 83 （P）（16） ASCII→HEX 变换 （ASCII→HEX）	操作元件：[S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D·]：n:1~256　X　Y　M　S 程序步数：HEX……7 步

如图 4-61 所示，M8161 为 OFF 时为 16 位，HEX 指令将最多 256 个 ASCII 码转换为 4 位十六进制数，每 2 个 ASCII 码占用一个数据寄存器，每 4 个 ASCII 码转换后的十六进制数

占一个数据寄存器。M8161 为 ON 时为 8 位模式，只转换源操作数低字节中的 ASCII 码。

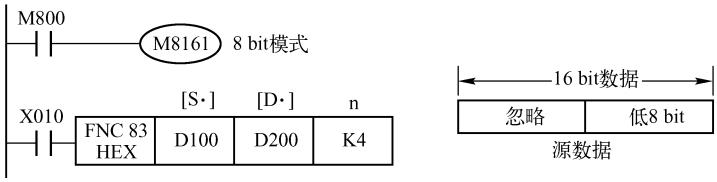


图 4-61　ASCII→HEX 变换示例

4.10.5　校验码

CCD FNC 84 （P）（16） 校验码 （Check Code）	操作元件： [S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z n　n [D·]：X　Y　M　S　n:1~256 程序步数：CCD……7 步

CCD 指令与串行通信指令 RS 配合使用，它将［S・］指定的元件开始的 n byte 组成堆栈（D 的高 byte、低 byte 拆开），将各 byte 数值的总和送到［D・］指定的元件，而将堆栈中垂直奇偶校验值送到［D・］＋1。通信时可将求和或异或的结果随同数据发送出去，对方收到后对接收到的数据也作同样的求和与异或运算，并判别接收到的求和与异或的结果是否等于求出的结果，如不等，则说明数据传送出错。

4.10.6　读变量

VRRD FNC 85 （P）（16） 读变量 FX－8AV （Volume Read）	操作元件： [D·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S·]：变量号(0~7) 程序步数：VRRD，VRRD（P）……5 步

FX－8AV－BD 是内置式 8 位 8 路模拟量功能扩展板，板上有 8 个小型电位器，用 VRRD 指令读出的数据（0～255）与电位器的角度成正比。VRRD 指令应用示例如图 4-62 所示，当 X0 为 ON 时，读出 0 号模拟量的值（［S・］＝0），送到 D0 后作为定时器 T0 的设定值。

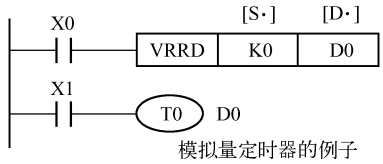


图 4-62　VRRD 指令应用示例

4.10.7　变量整标

VRSC FNC 86 （P）（16） 变量整标 （Volume Scale） FX－8AV	操作元件： [D·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S·]：变量号(0~7) 程序步数：VRSC，VRSC（P）……5 步

VRSC 指令将电位器读出的数四舍五入，整量化为 0 ~ 10 的整数值，存放在［D・］Z 中，这时电位器相当于一个有 11 档的模拟开关。

VRSC 指令应用示例如图 4-63 所示，用模拟开关的输出值和解码指令 DECO 来控制 M0 ~ M10，用户可根据模拟开关的刻度值（0 ~ 10）来分别控制 M0 ~ M10 的 ON/OFF。

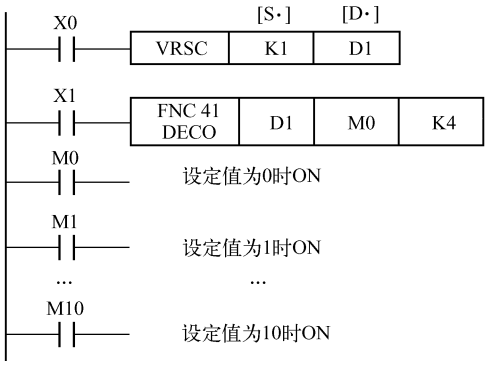


图 4-63　VRSC 指令应用示例

4.10.8　PID 运算

PID FNC 88 （16） PID 运算 PID	操作元件： [S1] [S2] [S3] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z [D] X | Y | M | S　　[S3]:D0~D7975 程序步数：PID……9 步

PID 指令用于模拟量闭环控制。PID 运算所需的参数存放在指令指定的数据区内。［S1］存放置位值，［S2］存放当前值，［S3］是用户 PID 指令定义参数的首址，范围是 D0 ~ D7975，［D］存放输出值，如图 4-64 所示。

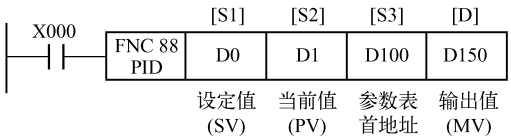


图 4-64　PID 指令应用示例

当 X000 从 OFF 变为 ON 时，执行指令。置位值存入 D0，当前值从 D1 中读出，保留 D100 ~ D124 作为用户定义参数的寄存器，数值存入 D150。

一个程序中可以使用多条 PID 指令，但每条指令的数据寄存器都要独立，以避免混乱。

PID 指令有报警系统，可由用户自己定义的参数设置触点。

PID 指令的出错信息放在 D8067 中。

PID 指令可以用中断、子程序、步进梯形指令和条件跳转指令，但使用时要小心。

采样时间必须大于程序扫描周期。

4.11 实数处理（FNC 110 ~ FNC 147）

4.11.1 实数比较

ECMP FNC 110 （P）（32） 实数比较	操作元件： [S1·] [S1·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z [S2·] [S2·] X | Y | M | S　[D·] 占有3点 [D·] 程序步数：（D）ECMP……13 步

图 4-65 所示程序是将源［S1・］和［S2・］的实数值作比较，将结果送到目标［D・］中。程序中 M0、M1、M2 根据比较结果动作。［S1・］>［S2・］时，M0 = ON，［S1・］=［S2・］时，M1 = ON，［S1・］<［S2・］时，M2 = ON。执行条件 X001 = OFF 时，（D）ECMP 指令不执行，M0、M1、M2 的状态保持不变。

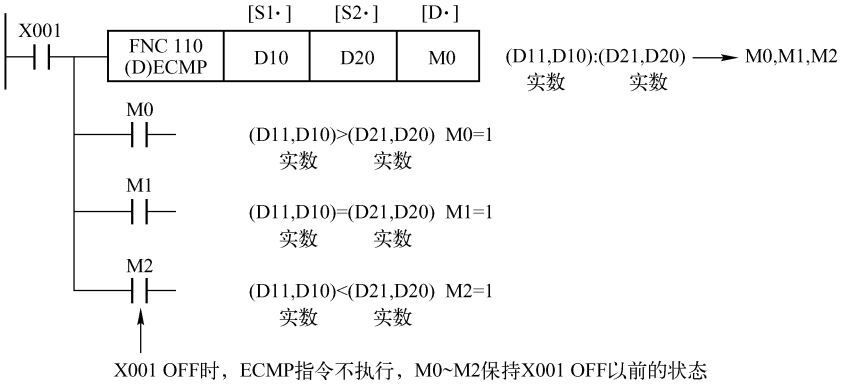


图 4-65　实数比较示例 1

常数 K、H 被指定为源数据时，自动转换成实数处理，如图 4-66 所示。

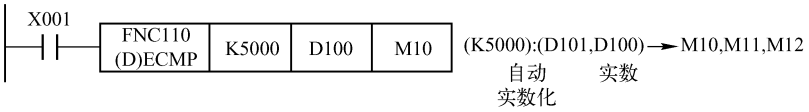


图 4-66　实数比较示例 2

4.11.2 实数区间比较

EZCP FNC 111 （P）（32） 实数区间比较	操作元件： [S1·] [S2·] [S·] [S1·] [S2·] [S·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z X | Y | M | S　[S1·] < [S2·] [D·]　[D·] 占连续的3点 程序步数：（D）EZCP……9 步

EZCP 指令是将一个数据与两个源数据值进行比较，将结果送到［D·］。三个源数据［S·］、［S1·］、［S2·］都必须为实数（二进制浮点方式）；常数 K、H 作为源数据时会自动转为二进制浮点格式。

图 4-67 所示程序中：

（D1，D0）<（D21，D20）时，M3 = ON；

（D21，D20）<（D1，D0）<（D31，D30）时，M4 = ON；

（D1，D0）>（D31，D30）时，M5 = ON。

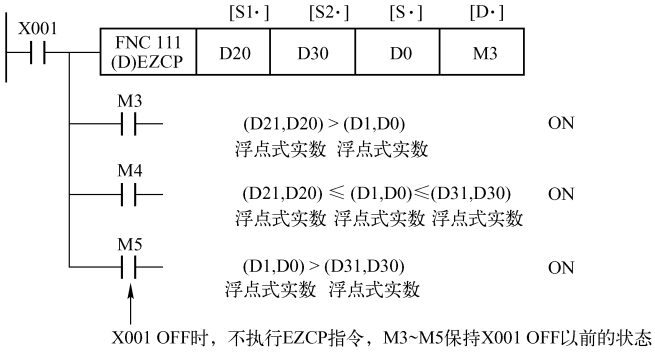


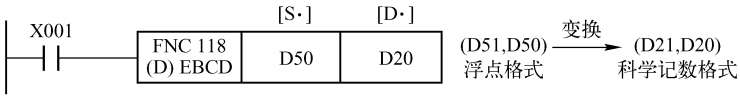
图 4-67　实数区间比较

应注意使［S1·］的值小于等于［S2·］的值。若［S1·］的值大于［S2·］的值，则看作［S2·］=［S1·］（把［S2·］的值看作与［S1·］的值相等）。

4.11.3　浮点数—科学记数变换

EBCD FNC 118 （P）（32）	操作元件： [S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D·]：X　Y　M　S 程序步数：（D）EBCD……9 步

本指令把源［S·］中浮点格式的实数变换成科学记数法的尾数，指数格式存入［D·］中。

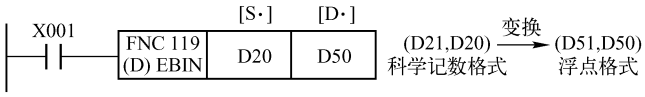


程序中源和目标都占双字（Double Word）。目标［D·］中，D20 存放尾数，D21 存放以 10 为底的指数。尾数 D 应在 1 000 ~ 9 999 范围（0 也可以）。指数应在 -41 ~ + 35 范围内。例如，S = 3. 4567 +10 在 D 中表示为 D = 3 456，D + 1 = -8。

4.11.4　科学记数—浮点数变换

EBIN FNC 119 （P）（32）	操作元件： [S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D·]：X　Y　M　S 程序步数：（D）EBIN……9 步

本指令把源［S·］中科学记数格式的实数变换为浮点格式存入［D·］中。

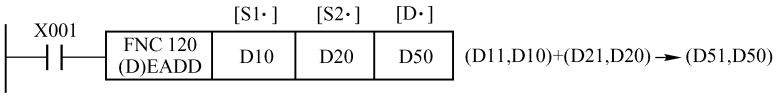


与 EBCD 指令相同，源和目标均占双字（32 bit）。

4.11.5 实数加法

EADD FNC 120 （P）（32）	操作元件： [S1·][S2·] [S1·][S2·] K,H K_nX K_nY K_nM K_nS T C D V,Z X Y M S [D·] 程序步数：（D）EADD……13 步

本指令把指定源元件中的两个实数相加，结果送到指定的目标元件中。



源、目标均各占一个双字 32bit；常数 K、H 用作源时会自动转成浮点实数格式。运算是代数运算，例如

$$2.3456\times10^{2}+(-5.6\times10^{-1})=2.34\times10^{2}$$

源和目标可采用相同的元件号。源和目标元件号相同，而且指令用连续执行方式时，每个扫描周期均执行加法，结果会不断变化。

1）运算结果为零，零标志 M8020 = ON；

2）运算结果超过浮点数可表示的最大值，进位标志 M8022 = ON，且结果表示为浮点数的最大值；

3）运算结果小于浮点数可表示的最小值，借位标志 M8021 = ON，且结果表示为浮点数的最小值。

4.11.6 实数减法

ESUB FNC 121 （P）（32） 实数减法	操作元件： [S1·][S2·] [S1·][S2·] K,H K_nX K_nY K_nM K_nS T C D V,Z X Y M S [D·] 程序步数：（D）ESUB……13 步 标　　志：M8020（零标志），M8021（借位标志），M8022（进位）

本指令将［S1·］指定元件中的实数减去［S2·］指定元件中的实数，结果存入［D·］指定的元件。

注意：各点及标志状态均与 EADD 指令相同。

4.11.7 实数乘法

EMUL FNC 122 （P）（32）	操作元件： [S1·][S2·] [S1·][S2·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z X | Y | M | S [D·] 程序步数：（D）EMUL……13 步

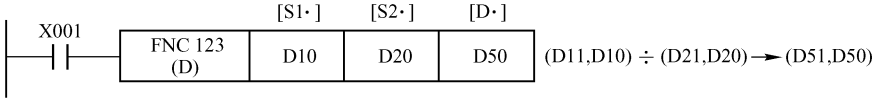
本指令将两源的乘积送到指定的目标。源和目标均为 32 bit 浮点格式实数，常数 K、H 用作源时会自动转换成浮点格式。

注意：各点与 EADD 指令相同。

4.11.8 实效除法

EDIV FNC 123 （P）（32）	操作元件： [S1·][S2·] [S1·][S2·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z X | Y | M | S [D·] 程序步数：（D）EDIV……13 步

本指令将［S1・］指定元件中的实数除以［S 2・］指定元件中的实数，商存入［D・］指定的元件。

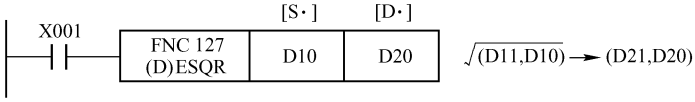


常数 K、H 作为源数据时会自动转换成浮点格式；［S2・］为零时会出错，指令不执行。

4.11.9 实效开平方

ESQR FNC 127 （P）（32）	操作元件： [S·] [S·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z [D·] X | Y | M | S [S·]的内容只有为正数才有效 程序步数：（D）ESQR……9 步 标　　志：M8020（零标志）

本指令将源［S・］指定元件中的实数开平方，平方根送入［D・］指定的元件。



常数 K、H 作为源数据时会自动转为浮点格式。源数据必须为正数；若为负数，出错标志 M8067 = ON，指令不执行。

源和目标均为 32 bit 浮点格式。源中角度以弧度值表示，0≤（S）≤2。0°～360°角度值须变换成弧度值后方可取正弦函数值（COS，TAN 指令时情况相同）。

图 4-68 所示是 45、90 取正弦值的例子。

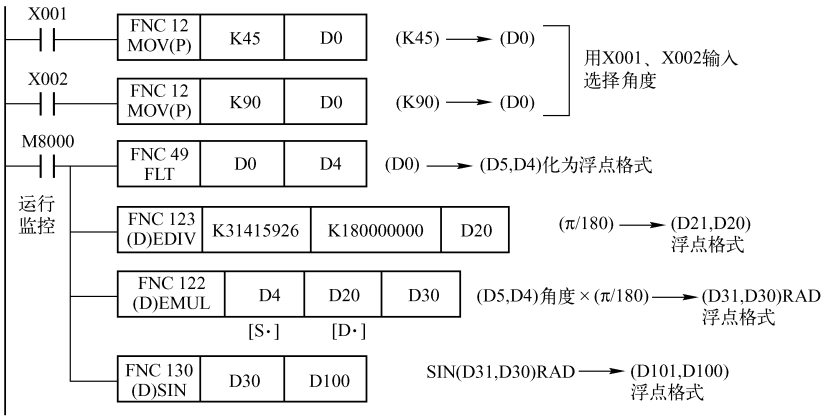
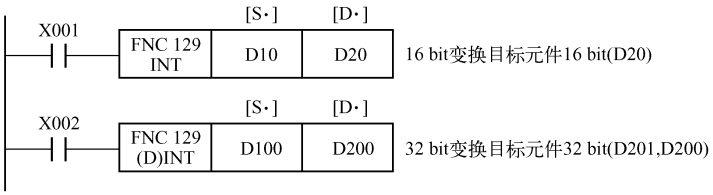


图 4-68　取正弦值的例子

4.11.10　实数－整数变换

INT FNC 129 （P）（16/32） Float to Integer	操作元件： K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z　[S·] X　Y　M　S　[D·] 程序步数：INT……5 步；（D）INT……9 步 标　　志：M8020（零标志），M8021（借位标示），M8022（进位）

本指令将源［S・］指定元件中的浮点格式实数变换为最接近其值的整数，以 BIN 格式存入目标［D・］指定的元件。［S・］必须是 32 bit 浮点格式实数。



本指令为 FLT FNC49 指令的逆变换：

1）结果为零时，零标志　M8020 = ON；

2）结果不足 1 而被舍去时，借位标志 M8021 = ON；

3）结果超过目标元件可表示的数的范围时，进位标志 M8022 = ON，且目标元件中的值无效。

可表示的数的范围如下：

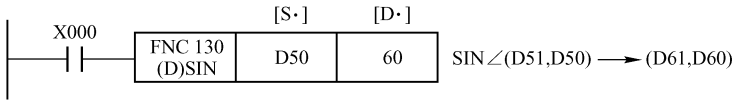
16 bit 运算时：－32 768 ～ ＋32 767；

32 bit 运算时：－2 147 483 648 ～ ＋2 147 483 647。

4.11.11 正弦函数

SIN FNC 130 （P）（32） 正弦函数 SIN	操作元件： [S·] K,H K_nX K_nY K_nM K_nS T C D V,Z [D·] X Y M S　0≤角度<2π 程序步数：（D）SIN……9 步

本指令计算［S·］指定的数据寄存器对中的角度（弧度值）所对应的正弦函数值，把结果存入［D·］指定的数据寄存器对中。

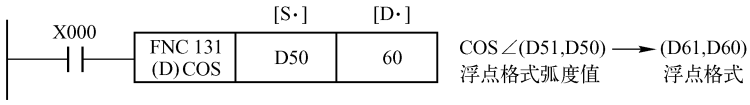


源和目标均为 32 bit 浮点格式。源中角度以弧度值表示，0≤（S）≤2π。0°～360°角度值须变换成弧度值后方可取正弦函数值（COS，TAN 指令时情况相同）。

4.11.12 余弦函数

COS FNC 131 （P）（32） 余弦函数 COS	操作元件： [S·] K,H K_nX K_nY K_nM K_nS T C D V,Z [D·] X Y M S　0≤角度<2π 程序步数：（D）COS……9 步

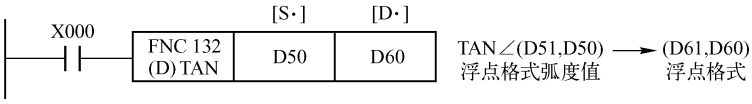
本指令计算［S·］指定的数据寄存器对中角度（弧度值）所对应的余弦函数值，把结果存入［D·］指定的数据寄存器对中。



4.11.13 正切函数

TAN FNC 132 （P）（32） 余弦函数 TAN	操作元件： [S·] K,H K_nX K_nY K_nM K_nS T C D V,Z [D·] X Y M S　0≤角度<2π 程序步数：（D）TAN……9 步

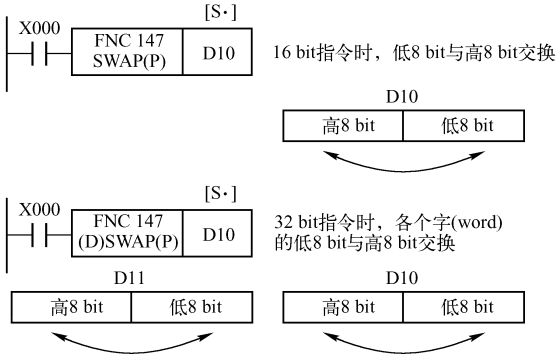
本指令计算［S·］指定的数据寄存器对中角度（弧度值）所对应的正切函数值，把结果存入［D·］指定的数据寄存器中。



4.11.14 高低 byte 交换

SWAP FNC 147 （P）（16/32）“！” 高低 byte 变换 Byte SWAP	操作元件： [S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z X　Y　M　S 程序步数：SWAP……3 步；（D）SWAP……5 步

本指令把源元件中每个字（word）的高、低字节（byte）交换，结果仍存入源元件中。



1）需要注意的是，该指令作为连续执行型执行时，各扫描周期都交换。

2）此指令的作用与 FNC 17（XCH）指令的扩展功能相同。

3）本指令其实是单独性质的指令，与实数处理并无关系。

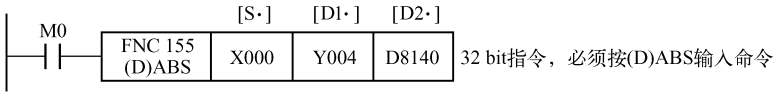
4.12 点位控制功能（FNC 150 ~ FNC 159）

本节介绍的指令只适用于 FX1S 及 FX1N，目的是使这两种低成本的 PLC 可以不借助其他扩展设备就可以实现简单的点位控制。但使用这些指令时最好配合三菱公司的伺服放大器（例如 MR－H 或 MR－J2 型伺服放大器）。本节的程序示例均为与该伺服放大器配合的例子。使用这些指令应注意的共同事项及相关软元件的情况均在本节的最后加以介绍。

4.12.1 当前绝对位置读取

ABS FNC 155 （32） 当前绝对位置读取 （Absolute Current Value Read）	操作元件： K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D2·] [S·] X　Y　M　S [D1·] 程序步数：（D）ABS……13 步

本指令读取当前绝对位置值，将其送到［D2·］指定的元件中。



要用本指令，PLC 应与有绝对位置功能的设备（例如三菱公司的 MR－H 及 MR－2J 型伺服放大器）连接。

［S·］指定来自伺服放大器信号输入点首地址，共占 3 点（上例中为 X000、X001、X002）。

［D1·］指定送到伺服放大器控制信号输出点的首地址，共占 3 点（上例中为 Y004、Y005、Y006）。

［D2·］指定从伺服放大器读取的绝对位置值存放的目标元件，占双字（32 bit），因为当前绝对位置必定存入（D8141，D8140），故通常［D2·］直接指定 D8140。

当执行条件（M0）由 OFF 变 ON 时开始读入绝对位置，读取完成后，完成标志 M8029 置 1。若读入中途执行条件（M0）OFF，则读取动作停止。通常让执行条件常为 ON（用 M8000），即便读取完成后也不应让它 OFF。因为若本指令的执行条件在读取完成后 OFF，伺服放大器的 ON 信号（SON）也会跟着 OFF，伺服放大器不工作。

虽然伺服放大器具有绝对位置读取功能，最好还是在系统最初启动时执行一次回原点操作（ZRN，FNC 156）。

系统设计时，应保证伺服放大器先于 PLC 得电，最起码应令两者同时得电。

上述程序的接线示例如图 4-69 所示。

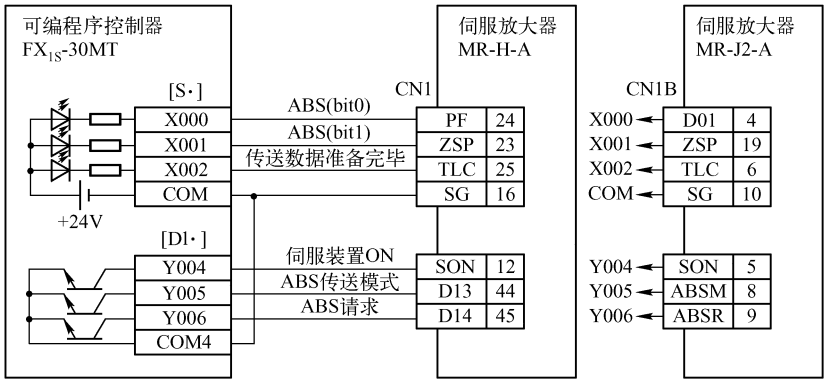


图 4-69　接线示例

如图 4-70 所示，FNC l55（D）ABS 指令驱动且伺服 ON 输出后，驱动 ABS 传送。在传送数据准备完成信号和 ABS 请求信号相互确认的同时，进行 32 +6 bit 数据的通信。数据通过 ABS bit 0，bit 1 回路执行数据传送。

伺服放大器的清零也可利用其本身的清零端，如图 4-71 所示。

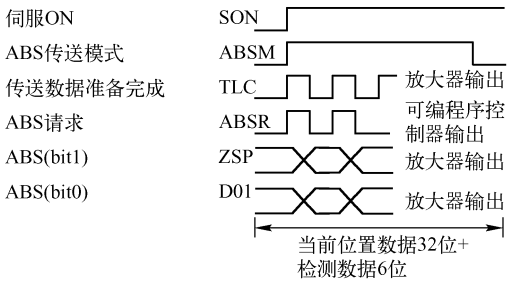


图 4-70　伺服放大器数据通信示例

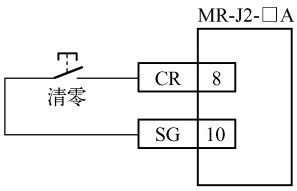
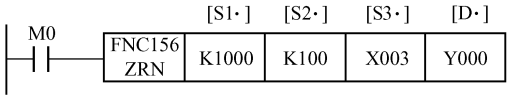


图 4-71　伺服放大器的清零示例

4.12.2 回原点

ZRN FNC 156 (16/32) 回原点 (Zero Return)	操作元件： [S1·] – [S2·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S3·]：X　Y　M　S [D·]：仅限Y000或Y001 程序步数：ZRN……9 步；(D) ZRN……17 步

当 PLC 执行 FNC 158（DRVI）和 FNC 159（DRVA）指令使机器运动时，当前位置值会增加或减少，但 PLC 都可以“记住”这些值，因而也就知道机器的当前位置。但当 PLC 断电时，这些数据就丢失了。要应对这种情况，机器启动时必须执行回原点操作，以校准机械的原点。



［S1・］指定回原点速度，16 bit 指令时为 10 ~ 32 767 Hz，32 bit 指令时为 10 ~ 100 kHz。

［S2・］指定爬行速度，接近点（DOG）信号 ON 后的低速，10 ~ 32 767 Hz。

［S3・］接近点信号，（常开触点）最好用 X，以免受扫描周期影响加大原点误差。

［D・］脉冲输出点，仅限于 Y000 与 Y001，必须用晶体管输出单元。

若在执行 FNC 156（ZRN）指令之前使 M8140 置 1，可使 PLC 在回原点操作完成后向伺服放大器输出清零信号。相应清零信号输出点如下：

1）脉冲输出为 Y000 时，清零输出为 Y002；

2）脉冲输出为 Y001 时，清零输出为 Y003。

清零信号必须以漏型晶体管输出，负载能力大于 200 mA（也适用于 FNC 157、FNC 158）。

若配用三菱公司的 MR – H 和 MR – J2 型伺服放大器，则停电时机械可保持其当前位置，而且可用 FNC 155（D）ABS 读取机械的绝对位置，所以只需在首次起动时执行回原点操作，以后即使断电后再起动也不必再回原点。

因为不具备 DOG 搜索功能，所以回原点前机械必须在 DOG 前方位置。

图 4–72 所示是回原点指令执行情况说明。

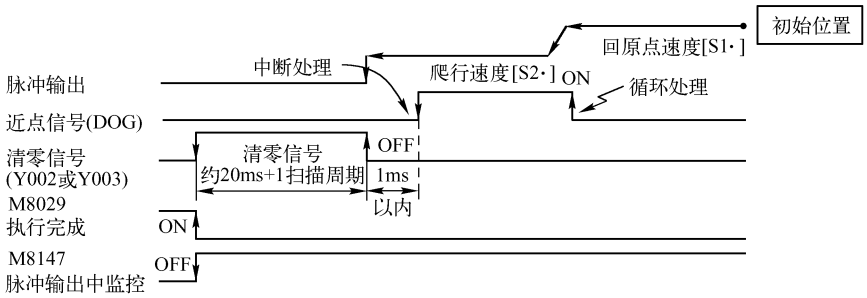
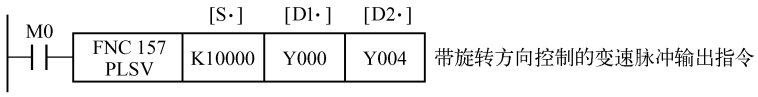


图 4–72　回原点指令执行情况

4.12.3 变速脉冲输出

PLSV FNC 157 (16/32) 变速脉冲输出 (Palse V)	操作元件： [S·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D2·]：X　Y　M　S [D1·]：仅限Y000或Y001 程序步数：PLSV……9 步，(D) PLSV……17 步



[S·] 指定输出脉冲频率：

16 bit 指令：10 ~ 32 767 Hz；

32 bit 指令：10 ~ 100 kHz。

[D1·] 指定脉冲输出点，仅限 Y000 与 Y001，必须用晶体管输出单元；

[D2·] 指定旋转方向控制输出点：

[D2·] = ON 正转；

[D2·] = OFF 反转。

在脉冲输出过程中可自由改变输出脉冲频率，故称变速输出。

在起动/停止时没有加/减速过程，如需软起动/停止，可利用 RAMP　FNC67 指令改变 [S·] 的值。

在脉冲输出过程中指令驱动触点 OFF，则不经减速立即停止脉冲输出。

指令驱动触点 OFF 后，在脉冲输出中标志（Y000 对应 M8147，Y001 对应 M8148）仍为 ON 期间不可再次驱动本指令。

旋转方向的正/反由脉冲输出频率 [S·] 的正、负号来决定。

相关软元件：

1) [D8141（高位），D8140（低位）]：向 Y000 输出脉冲数，反转时减少（32 bit）；

2) [D8143（高位），D8142（低位）]：向 Y001 输出脉冲数，反转时减少（32 bit）；

3) [M8145]：Y000 脉冲输出停止（立即停止）；

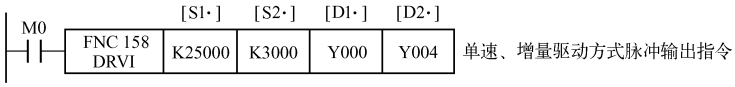
4) [M8146]：Y001 脉冲输出停止（立即停止）；

5) [M8147]：Y000 脉冲输出中监控（BUSY/READY）；

6) [M8148]：Y001 脉冲输出中监控（BUSY/READY）。

4.12.4 增量驱动

DRVI FNC 158 (16/32) 增量驱动 (Drive to Increment)	操作元件： [S1·] – [S2·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S3·]：X　Y　M　S [D1·]：仅限Y000或Y001 程序步数：DRVI……9 步，(D) DRVI……17 步 标　　志：M8029（完成标志）



［S1・］输出脉冲数：

16 bit 指令：－32 768 ~ ＋32 767；

32 bit 指令：－999 999 ~ ＋999 999。

［S2・］输出脉冲频率（最低输出频率限制见本节后述）：

16 bit 指令：10 ~ 32 767 Hz；

32 bit 指令：10 ~ 10 kHz。

［D1・］脉冲输出点，仅限 Y000 与 Y001，必须为晶体管输出。

［D2・］旋转方向信号输出点，ON 为正转，OFF 为反转。

指令执行过程中，输出脉冲数以增量方式存入当前值寄存器对：

输出点 Y000 对应（D8141，D8140）；

输出点 Y001 对应（D8143，D8142）；

正转时数值增加，反转时数值减少。

旋转方向的正/反由［S1・］的符号正/负决定。

指令执行中，改变操作元件的内容也无法在当前运行中反映出来，只有在下一次执行时才生效。

指令执行过程中，执行条件 OFF，将减速停止，但完成标志 M8029 不动作。

指令执行条件 OFF 后，而脉冲输出中标志（Y000 对应 M8147，Y001 对应 M8148）仍为 ON 时，不接受指令的再次驱动。

增量式驱动中，［S1・］指定的输出脉冲数是指由当前位置到目标位置之间应输出的脉冲数，即当前位置与目标位置之间的距离（以脉冲为单位），如图 4-73 所示，脉冲数的 +/－表示运动方向。

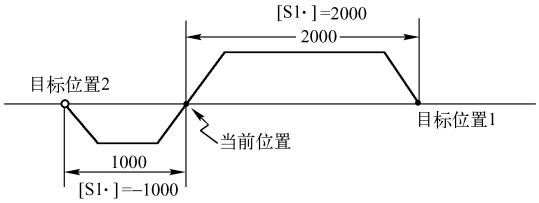


图 4-73　当前位置与目标位置之间的距离示例

增量驱动设置值与速度曲线如图 4-74 所示。

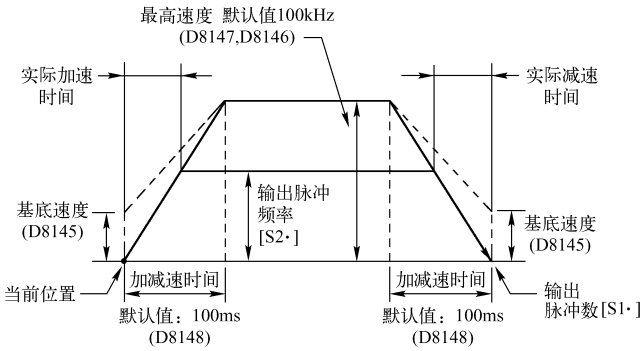


图 4-74　增量驱动设置值与速度曲线

实际能够执行的最低输出脉冲频率由以下公式决定：

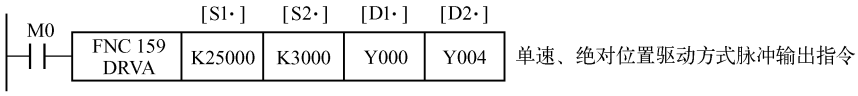
√最高速度［D8147，D8146］Hz ÷（2 × 加速时间［D8148］ms ÷ 1 000）= 输出脉冲最低频率

若输出脉冲频率［S2 ·］指定低于上面计算结果的数值，实际输出频率等于上式计算结果值。

加速初期和减速最终部分的频率也不可低于上述计算结果。

4.12.5 绝对位置驱动

DRVA FNC 169 （16/32） 绝对位置驱动 （Drive to Absolute）	操作元件： [S1·] – [S2·]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S3·]：X　Y　M　S [D1·]：仅限Y000或Y001 程序步数：DRVA……9 步，（D）DRVA……17 步 标　　志：M8029（完成标志）



［S1 ·］目标绝对位置：

16 bit 指令：－32 768 ～ ＋32 767；

32 bit 指令：－999 999 ～ ＋999 999。

［S2 ·］输出脉冲频率，最低脉冲频率由本小节下述公式决定。

［D1 ·］脉冲输出点，仅限 Y000 与 Y001 晶体管输出。

［D2 ·］旋转方向信号输出点，旋转方向由［S1 ·］与当前位置的差的 +／－号决定：

［D2 ·］＝ON 正转（前进）；

［D2 ·］＝OFF 反转（后退）。

指令执行过程中，当前绝对位置存入当前值寄存器对：

输出点 Y000 对应（D8141，D8140）；

输出点 Y001 对应（D8143，D8142）。

指令执行过程中改变操作元件的内容也无法在当前运行中反映出来，只有在下一次执行时才生效。

指令执行过程中，若执行条件 OFF，将减速停止，但完成标志 M8029 不动作。

指令执行条件 OFF 后，而脉冲输出中标志（Y000 对应 M8147，Y001 对应 M8148）仍为 ON 时不接受指令的再次驱动。

绝对值驱动方式中，［S1 ·］指定的是目标位置与原点的距离，即目标的绝对位置，如图 4–75 所示。

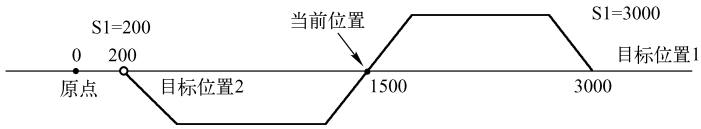


图 4–75　目标绝对位置示例

绝对驱动方式设置值与驱动曲线如图 4–76 所示。

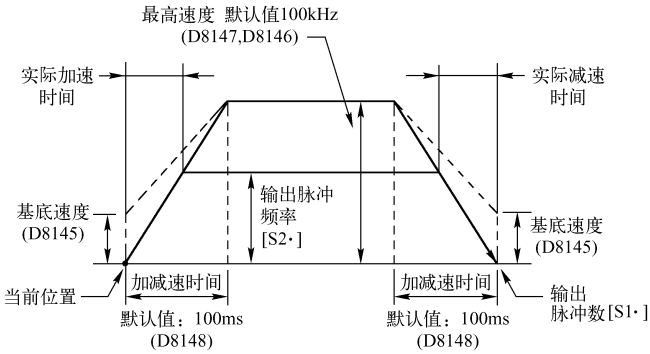


图 4-76　绝对驱动方式设置值与驱动曲线

实际能够执行的输出脉冲最低频率由以下公式决定：

$$\sqrt{\text{最高速度［D8147，D8146］Hz} \div (2 \times \text{加速时间［D8148］ms} \div 1\,000)} = \text{输出脉冲最低频率}$$

若输出脉冲频率数［S2・］指定了低于上面计算结果的数值，计算实际输出频率等于上式计算结果值。

加速初期和减速最终部分的频率也不可低于上述计算结果。

4.12.6　使用点位控制指令注意事项

点位控制指令 FNC 156（ZRN）、FNC 157（PLSV）、FNC 158（DRVI）、FNC 159（DRVA）编程时均可多次使用，但应当注意：

1）不可出现双线圈现象，即不可同时有两条指令驱动同一个目标线圈。因为目标线圈仅限 Y000、Y001，故指令多次使用时极易产生双线圈现象。

2）当指令执行条件 OFF 后，必须等该指令目标元件对应的脉冲输出中监控标志（Y000 对应 M8147，Y001 对应 M8148）OFF 后再经一个扫描周期方可再次驱动同一指令。

3）为了按上面要求编程，最好参照下述的标准程序。

4）FNC 57 PLSY、FNC 59 PLSR 的目标元件也是 Y000 与 Y001，故最好用 FNC 158 DRVI 来代替 FNC 57 和 FNC 59。如一定要使用 FNC 57 和 FNC 59 时，前三条注意事项也适用于这两指令。

5）Y000、Y001 作高速输出时其性能指标如下：

电压：DC 5 ~ 24 V；

电流：10 ~ 100 mA；

输出频率：小于 100 kHz。

相关软元件表见表 4-8、表 4-9。

与 MR - J2 型伺服放大器连接示例如图 4-77 所示。

表 4-8　相关软元件表 I

元件地址		数据长度 16/32 bit	初始值	内　　容
D8140	低位	32 bit	0	作为 Y000 输出定位指令的当前数据寄存器使用 用 FNC 157（PLSV）、FNC 158（DRVI）、FNC 159（DRVA）指令时，对应旋转方向增减当前值 另外，由于 FNC 57（PLSY）、FNC 59（PLSR）指令也使用相同当前值寄存器，因此当执行这些指令时，当前值的数值为脉冲输出数的累加值
D8141	高位			

（续）

元件地址		数据长度 16/32 bit	初始值	内　　容
D8142	低位	32 bit	0	作为 Y001 输出定位指令的当前数据寄存器使用 用 FNC 157（PLSV）、FNC 158（DRVI）、FNC 159（DRVA）指令时，对应旋转方向增减当前值 另外，由于 FNC 57（PLSY）、FNC 59（PLSR）指令也使用相同当前值寄存器，因此当执行这些指令时，当前值的数值为脉冲输出数的累加值
D8143	高位			
D8145		16 bit	0	执行 FNC 156（ZRN）、FNC 158（DRVI）、FNC 159（DRVA）指令时的基底速度 设定范围：最高速度（D8147，D8146）的 1/10 以下，当选用超出该范围的数值时，自动作为最高速度的 1/10 运行
D8146	低位	32 bit	100 000	执行 FNC156（ZRN），FNC158（DRVI），FNC159（DRVA）指令时的基底速度 设定范围：10 ~ 100 000 Hz
D8147	高位			
D8148		16 bit	100	执行 FNC 156（ZRN）、FNC、158（DRVI）、FNC 159（DRVA）指令时，从基底速度（D8145）到最高速度（D8147，D8146）的加减速时间（FNC 158、FNC 159） 设定范围：50 ~ 5 000 ms

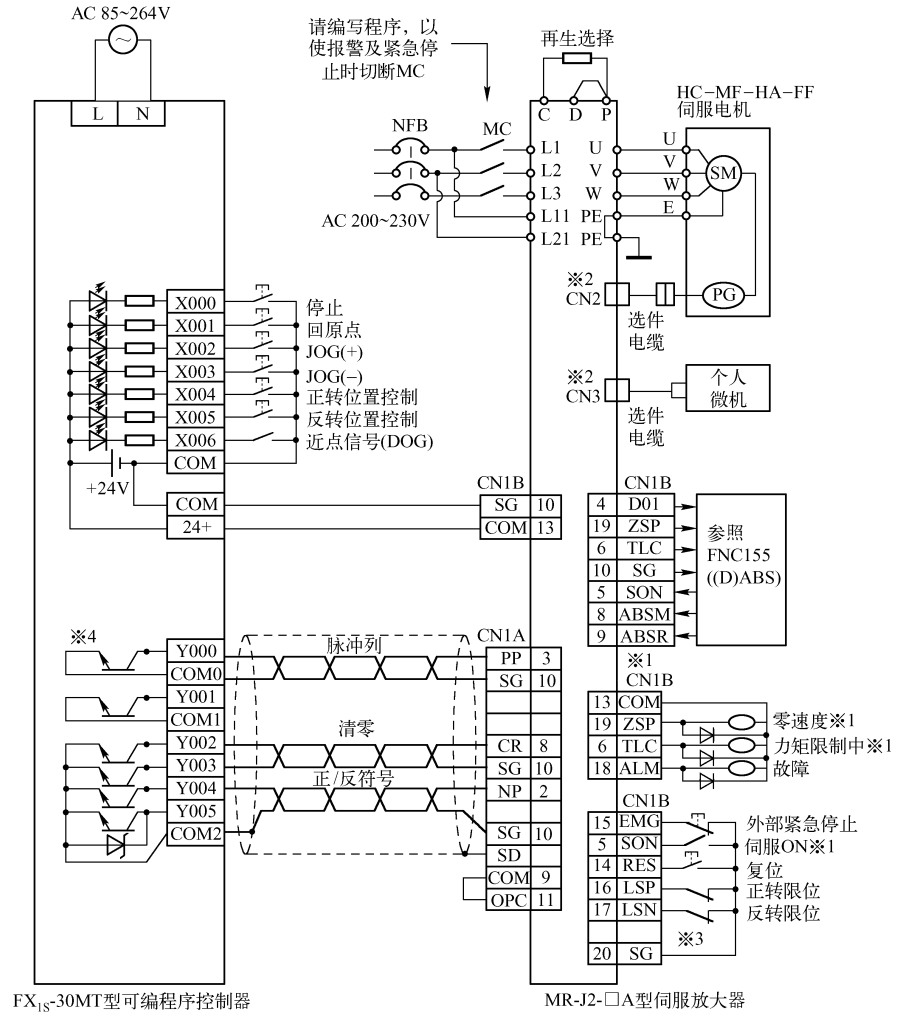


图 4-77　与 MR－ J2 型伺服放大器连接

表 4-9　相关软元件表 Ⅱ

元 件 地 址	属　　性	内　　容
M8145	读/写	Y000 脉冲输出停止标志（立即停止）
M8146	读/写	Y001 脉冲输出停止标志（立即停止）
M8147	只读	Y000 脉冲输出中监控（BUSY/READY）
M8148	只读	Y001 脉冲输出中监控（BUSY/READY）

下面介绍标准程序例子。

输入/输出分配按图 4-77 所示。

运行方式如图 4-78 所示，以绝对位置方式进行定位。

在本例中，加速的第一段和减速的最终段的实际输出频率可以通过下式求得：

$$\sqrt{100\ 000 \div (2 \times (200 \div 1\ 000))} = 500\ \text{Hz}$$

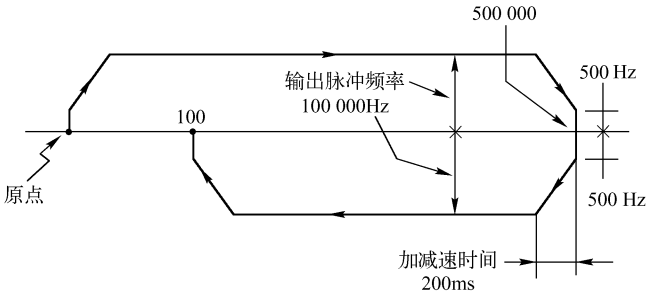


图 4-78　运行方式

图 4-79 所示为使用步进梯形图（STL）进行编程的例子。

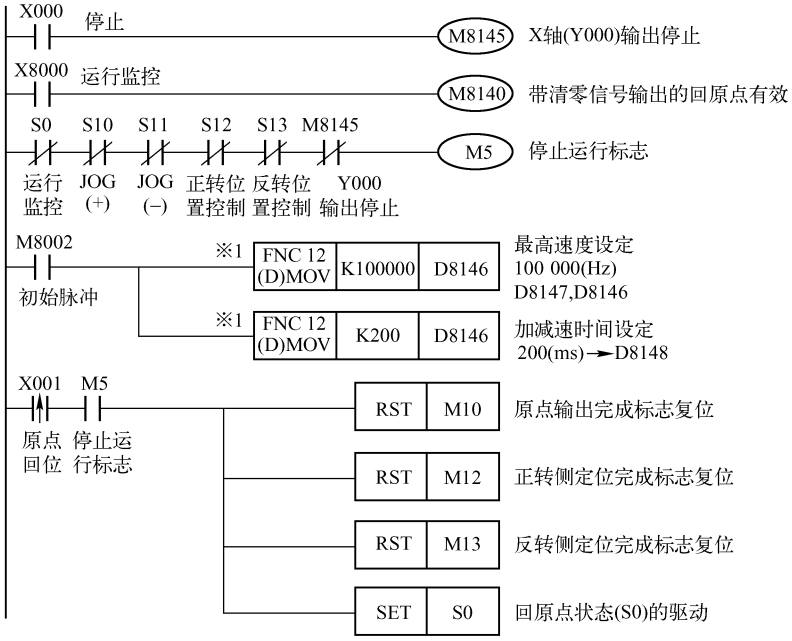


图 4-79　使用步进梯形图（STL）进行编程的例子

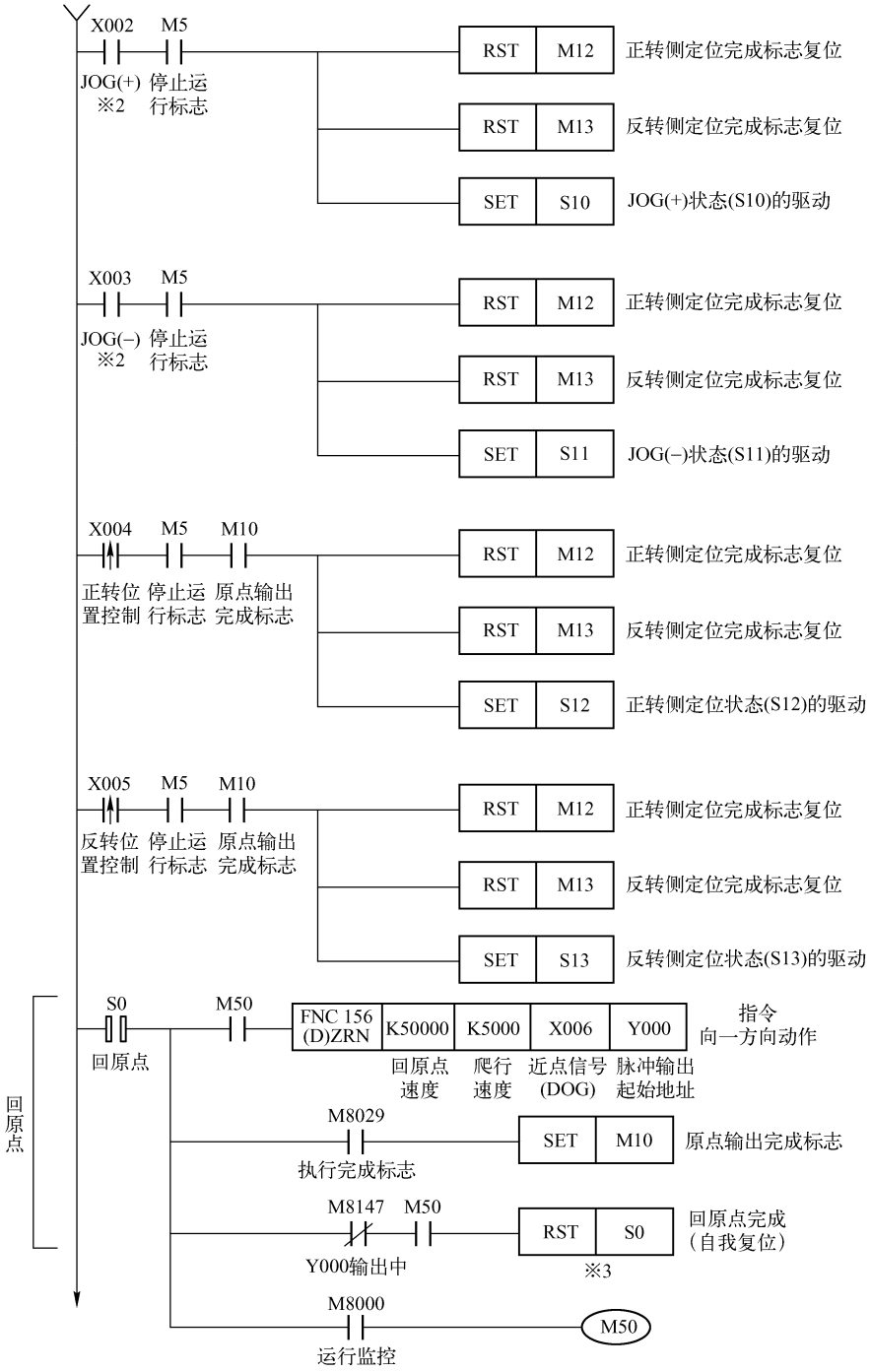


图 4–79　使用步进梯形图（STL）进行编程的例子（续）

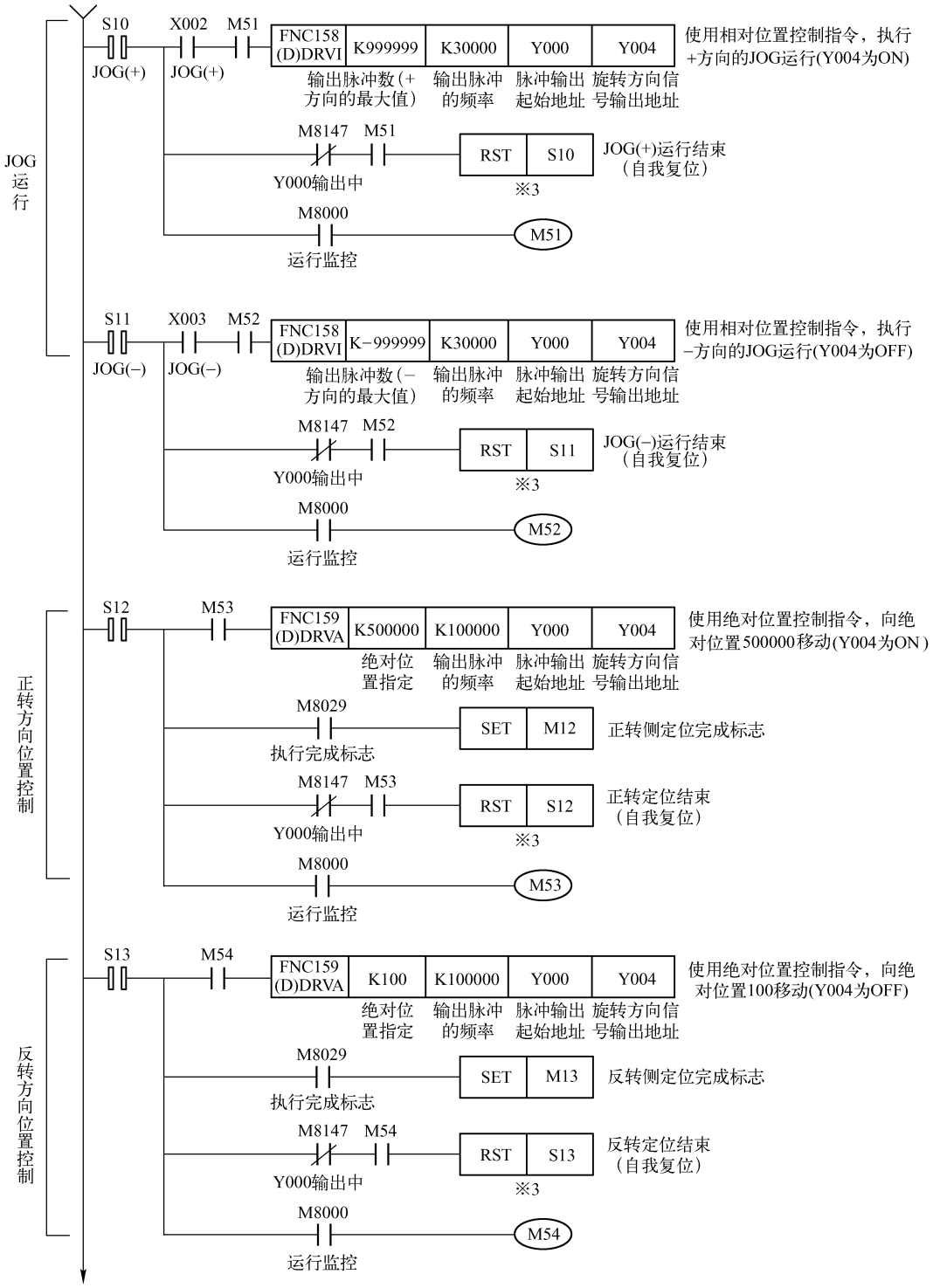


图 4-79　使用步进梯形图（STL）进行编程的例子（续）

4.13 实时时钟处理（FNC 160 ~ FNC 169）

4.13.1 时间比较

TCMP FNC 160 （P）（16） 时间比较 （Time Compare）	操作元件： [S1·] – [S2·] – [S3·] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S·] X　Y　M　S　[D·] 占用3点 [D·] 程序步数：TCMP……11 步

TCMP 指令的源操作数［S1・］、［S2・］、［S3・］用来存放指定时间的时、分、秒。该指令用来比较指定时刻与时钟数据的大小，时钟数据的时间存放在［S・］～［S・］+2 中，结果决定［D・］～［D・］+2 中的 ON/OFF 状态。该指令的应用示例如图 4-80 所示。

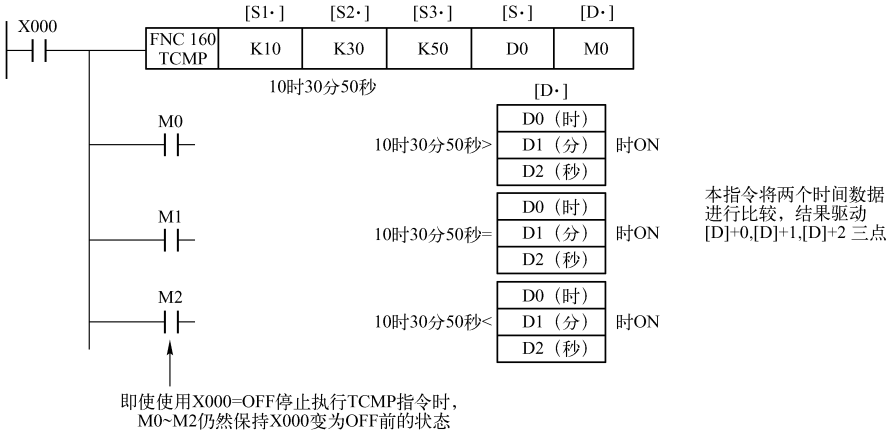


图 4-80　TCMP 指令应用示例

［S1・］：指定比较时间的“时”；

［S2・］：指定比较时间的“分”；

［S3・］：指定比较时间的“秒”；

［S・］：指定时钟数据的“时”；

［S・］+1：指定时钟数据的“分”；

［S・］+2：指定时钟数据的“秒”；

［D・］，［D・］+1，［D・］+2：根据比较结果决定其 ON/OFF 状态。

4.13.2 时间区间比较

TZCP FNC 161 （P）（16） 时间区间比较 （Time Zone Compare）	操作元件： [S·][S1·][S2·] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S·] X　Y　M　S　[D·] 占用3点 请使[S1·]≤[S2·] [D·] 程序步数：TZCP……9 步

本指令将一个时间值与两个源时间数据作区间比较，比较结果驱动［D・］指定元件开始的 3 个 bit 元件。如图 4-81 所示，D20 ~ D22 分别用来存放时、分、秒。

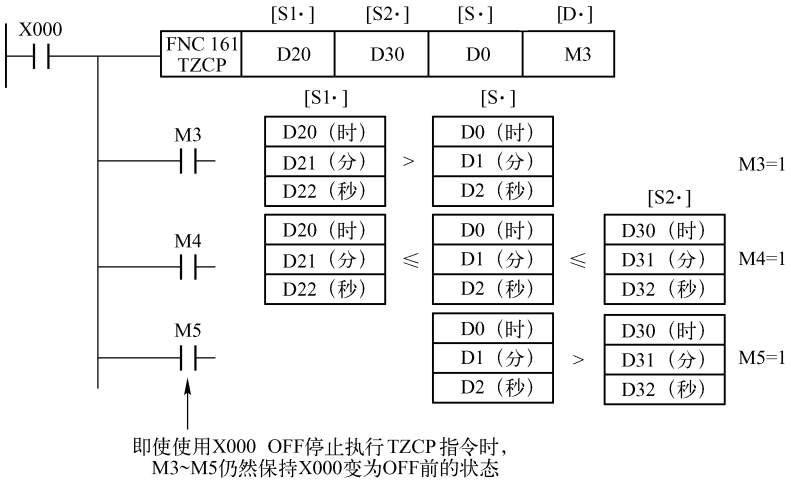


图 4-81　TZCP 指令应用示例

4.13.3　时间加法

TADD FNC 162 （P）（16） 时间加法 （Time Addition）	操作元件： [S1・]–[S2・] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D・] X　Y　M　S 程序步数：TADD……7 步 标　　志：M8022（进位）；M8020（零标志）

［S1・］、［S2・］和［D・］分别指定源和目标元件的首地址。源 1、源 2 和目标各占 3 个字元件，分别为时、分、秒值。本指令是将源 1 与源 2 时间值相加，结果送到目标元件。当运算结果超过 24 h 时，进位标志位变为 ON，并将加法运算的结果减去 24 h 作为运算结果保存。该指令的应用示例如图 4-82 所示。

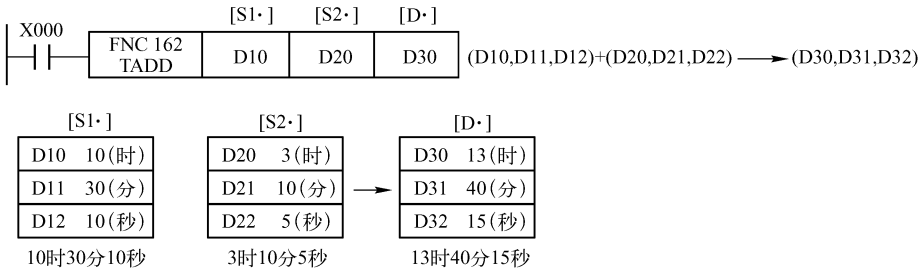


图 4-82　TADD 指令应用示例

4.13.4 时间减法

TSUB FNC 163 （P）（16） 时间减法 （Time Subtraction）	操作元件： [S1·]–[S2·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z [D·] X | Y | M | S 程序步数：TSUB……7 步 标　　志：M8020（零标志）；M8021（借位）

［S1·］、［S2·］和［D·］分别指定源和目标元件首地址。源 1、源 2 和目标各占 3 个字元件，分别为时、分、秒值。本指令是将源 1 与源 2 时间值相减，结果送到目标元件。当运算结果低于 00：00：00 时，借位标志变为 ON，并将减法运算的结果加上 24 h 作为运算结果保存。该指令的应用示例如图 4–83 所示。

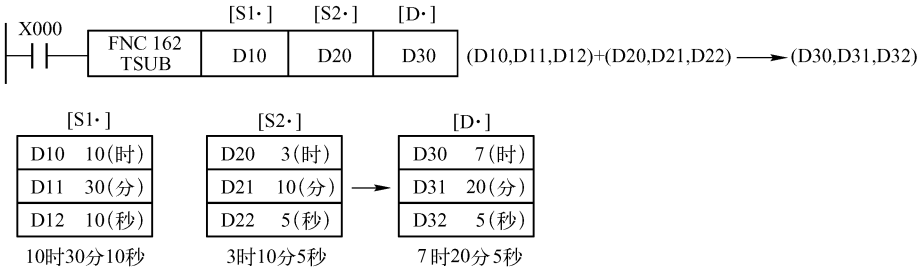


图 4–83　TSUB 指令应用示例

4.13.5 读实时时钟

TRD FNC 166 （P）（16） 读实时时钟 （Time Read）	操作元件： K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z [D·] X | Y | M | S　　[D·] 占用7个字元件 程序步数：TRD……3 步

TRD 指令用来读出内置的实时时钟数据，并存放在［D·］开始的 7 个元件中，实时时钟的时间数据存放在 D8013 ~ D8019 内，D8018 ~ D8013 中分别存放年、月、日、时、分和秒，D8019 中存放星期。该指令的应用示例如图 4–84 所示。

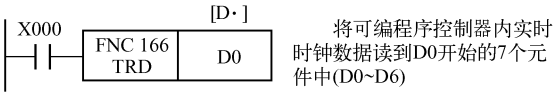


图 4–84　TRD 指令应用示例

4.13.6 写实时时钟

TWR FNC 167 （P）（16） 写实时时钟 （Time Write）	操作元件： K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z [S·] X | Y | M | S　　[S·] 占用7个字元件 程序步数：TWR……3 步

TWR 指令用来将时间设定值写入内置的实时时钟，写入的数据预先放在［S・］开始的 7 个元件中。执行该指令时，内置的实时时钟时间立即变更，改为使用新的时间。该指令的应用示例如图 4-85 所示。

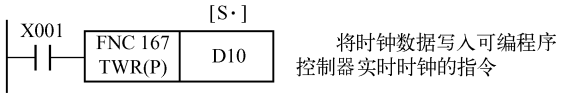


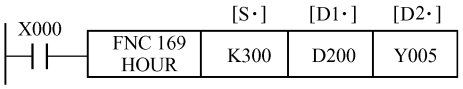
图 4-85　TWR 指令的应用示例

4.13.7　计时表

HOUR FNC 169 (16/32) 计时表 (Hour meter)	操作元件： [S・]：K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D2・]：X　Y　M　S [D1・]：必须指定停电保持用软元件 程序步数：HOUR……7 步

本指令在［D1・］指定元件中累计执行条件为 ON 的小时数，当 ON 时数超过［S・］指定的小时数值时，令［D2・］指定的元件 ON，以产生报警。

1. 16 bit 指令



当 X000 处于 ON 的时间超过 300 h，Y005 变为 ON 将不满 1 h 的当前值以 s（秒）为单位保存在 D201 中。

［S・］：使［D2・］变为 ON 的时间，以 h（小时）为单位；

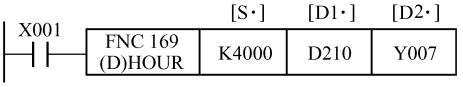
［D1・］：以 h 为单位的累计时间当前值；

［D1・］+1：不满 1 h 的当前值（以 s 为单位）；

［D2・］：报警输出地址。

当前值［D1・］超过［S・］指定的时间时变为 ON（在本例中，当 300 h + 1 s 时变为 ON）。

2. 32 bit 指令



［S・］：使［D2・］变为 ON 的时间；

［D1・］，［D1・］+1：以 h 为单位的累计时间当前值；［D1・］+ 1（高位），［D1・］（低位）；

［D1・］+2：不满 1 h 的当前值（以 s 为单位）；

［D2・］：报警输出地址；当前值［D1・］+1，［D1・］超过［S・］指定的时间时变为 ON（在本例中，40 000 h + 1 s 时变为 ON）。

为了能在切断可编程序控制器的电源后仍能保存当前值数据，［D1・］指定停电保持用数

据寄存器。

即使报警输出［D2・］的输出变为 ON 后，仍然能够继续计时。

当前值到达 16 bit 或 32 bit 的最大值时停止计时。

若希望继续进行计时时，请清除［D1・］~［D1・］+1（16 bit 时），［D1・］~［D1・］+3（32 bit 时）的当前值。

4.14 码制转换指令

4.14.1 BIN－GRY 变换

GRY FNC 170 （P）（16/32） BIN ~ GRY 变换	操作元件： [S・] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D・] X　Y　M　S 程序步数：GRY……5 步；（D）GRY……9 步

格雷码常用于广电码盘编码器，其特点是相邻的两个数（用二进制表示）的各位中，只有一位的值不同。该指令把源［S・］中的整数（BIN 码）变换成等值的格雷码形式存入目标［D・］。该指令的应用示例如图 4-86 所示。

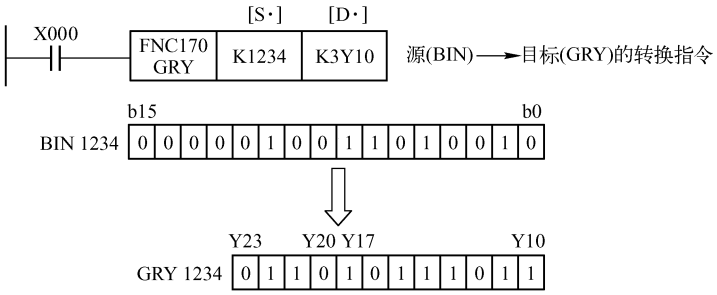


图 4-86　GRY 指令应用示例

4.14.2 GRY－BIN

GBIN FNC 171 （P）（16/32） GRY ~ BIN 变换	操作元件： [m1・]－[m2・] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D・] X　Y　M　S 程序步数：GBIN……5 步；（D）GBIN……9 步

该指令把源［S・］中的格雷码（GRY CODE）值变换成等值的整数（BIN 码），存入目标［D・］。该指令的应用示例如图 4-87 所示。

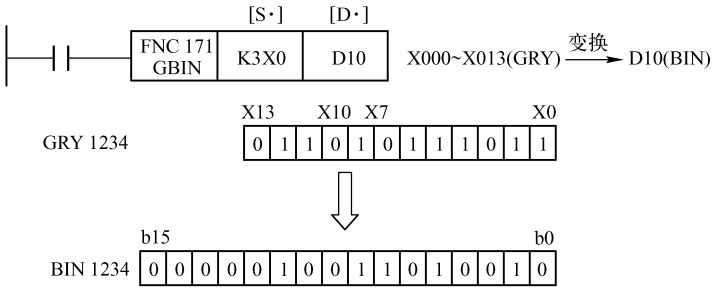
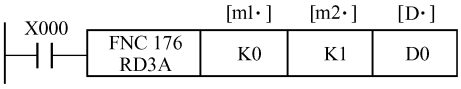


图 4-87　GBIN 指令应用示例

4.14.3　模拟量读（FX_{ON} -3A 专用）

RD3A FNC 176 （P）（16） 模拟量读 （Read Fxon -3A）	操作元件： [m1·]-[m2·] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [D·] X　Y　M　S 程序步数：RD3A……7 步

该指令读取 FX_{ON} -3A 中指定通道的模拟量输入值存入目标［D·］。



［m1·］为特殊模块号：K0 ~ K7；

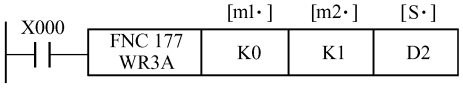
［m2·］为模拟量输入通道号：K1，K2；

［D·］为读入数据存放地址。

4.14.4　模拟量写（FX_{ON} -3A 专用）

WR3A FNC 177 （P）（16） 模拟量写 （Write to FX_{ON} -3A）	操作元件： [m1·]-[m2·] K,H　K_nX　K_nY　K_nM　K_nS　T　C　D　V,Z [S·] X　Y　M　S 程序步数：WR3A……7 步

该指令将源［S·］数据写入 FX_{ON} -3A 中指定的模拟量输出通道：



［m1·］为特殊模块号：K0 ~ K7；

［m2·］为模拟量输出通道号：K1；

［D·］为存放源数据的地址。

4.15 触点比较指令（FNC 224 ~ FNC 246）

有别于先前介绍的比较指令，下面的比较指令本身就像触点一样，而这些触点的通断取决于比较条件是否成立。若比较条件成立，则触点就导通，反之则断开。这样，这些比较指令就可像普通触点一样放在 LAD 程序的横线上，故又称为线上比较指令（inline Comparisions），按指令在线上的位置分为以下三大类。

4.15.1 LD□比较触点

LD □ FNC 224 ~ FNC 230 （16/32） LD□比较触点 □：=，>，<，<>，≤，≥	操作元件： [S1·]-[S2·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z X | Y | M | S 程序步数：LD□……5 步；（D）LD□……9 步

该指令将比较触点接到起始总线上的指令。触点的通/断取决于比较条件是否成立，如表 4-10、图 4-88 所示。

表 4-10 LD□比较触点应用

功能号	16 bit 指令	32 bit 指令	导 通 条 件	不导通条件
224	LD =	LD（D） =	[S1·] = [S2·]	[S1·] ≠ [S2·]
225	LD >	LD（D） >	[S1·] > [S2·]	[S1·] ≤ [S2·]
226	LD <	LD（D） <	[S1·] < [S2·]	[S1·] ≥ [S2·]
228	LD < >	LD（D） < >	[S1·] ≠ [S2·]	[S1·] = [S2·]
229	LD≤	LD（D） ≤	[S1·] ≤ [S2·]	[S1·] > [S2·]
230	LD≥	LD（D） ≥	[S1·] ≥ [S2·]	[S1·] < [S2·]

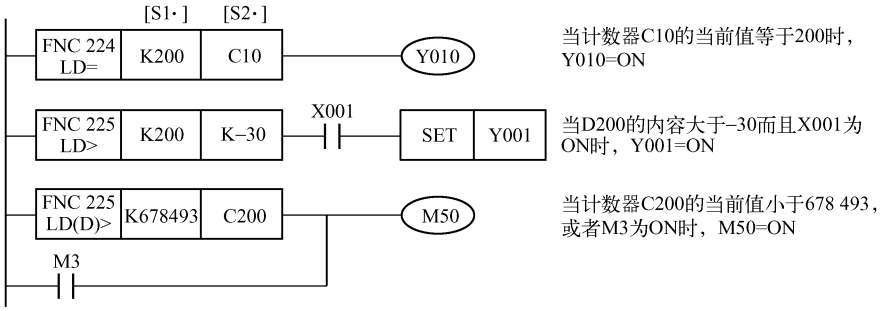


图 4-88 LD□比较触点应用示例

当源数据的最高 bit（16 bit 指令：b15；32 bit：b31）为 1 时，将该数值作为负数进行比较。

32 bit 计数器（C200 ~ C234）的比较，必须以 32 bit 指令来进行。若用 16 bit 指令，会导致程序出错。

4.15.2 AND□比较触点

AND□ FNC 232 ~ FNC 238 (16/32) AND□比较触点 □: =, >, <, <>, ≤, ≥	操作元件: [S1·]=[S2·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z X | Y | M | S 程序步数:AND□……5 步;(D) AND□……9 步

该指令将比较触点作串联连接的指令。比较触点是否导通取决于比较条件是否成立,如表4-11、图4-89所示。

表4-11 AND□比较触点应用

功能号	16bit 指令	32bit 指令	导 通 条 件	不导通条件
232	AND =	AND (D) =	[S1·] = [S2·]	[S1·] ≠ [S2·]
233	AND >	AND (D) >	[S1·] > [S2·]	[S1·] ≤ [S2·]
234	AND <	AND (D) <	[S1·] < [S2·]	[S1·] ≥ [S2·]
236	AND < >	AND (D) < >	[S1·] ≠ [S2·]	[S1·] = [S2·]
237	AND≤	AND (D) ≤	[S1·] ≤ [S2·]	[S1·] > [S2·]
238	AND≥	AND (D) ≥	[S1·] ≥ [S2·]	[S1·] < [S2·]

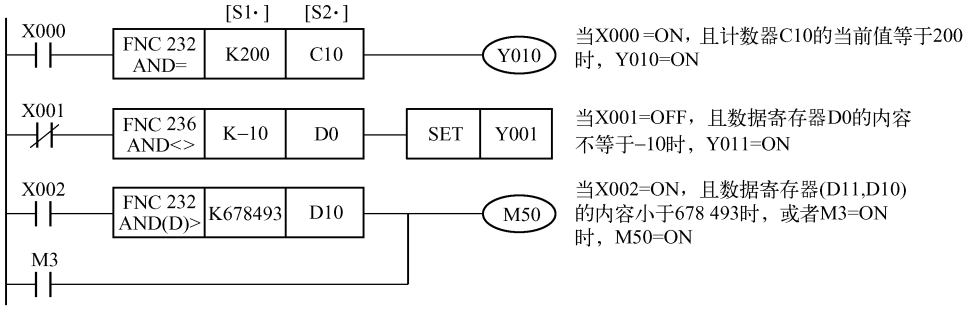


图4-89 AND□比较触点应用示例

当源数据的最高bit(16bit 指令:b15;32bit 指令:b31)为1时,将该数值作为负数进行比较。

32bit 计数器(C200~C234)的比较,必须以32bit 指令来进行。若用16bit 指令,会导致程序出错。

4.15.3 OR□比较触点

OR□ FNC 240 ~ FNC 246 (16/32) OR□比较触点 □: =, >, <, <>, ≤, ≥	操作元件: [S1·]=[S2·] K,H | K_nX | K_nY | K_nM | K_nS | T | C | D | V,Z X | Y | M | S 程序步数:OR□……6 步;(D) OR□……9 步

该指令将比较触点作并联连接的指令。比较触点是否导通取决于比较条件是否成立，如表 4-12、图 4-90 所示。

表 4-12　OR□比较触点应用

功能号	16 bit 指令	32 bit 指令	导 通 条 件	不导通条件
240	OR =	OR（D） =	[S1・] = [S2・]	[S1・] ≠ [S2・]
241	OR >	OR（D） >	[S1・] > [S2・]	[S1・] ≤ [S2・]
242	OR <	OR（D） <	[S1・] < [S2・]	[S1・] ≥ [S2・]
244	OR < >	OR（D） < >	[S1・] ≠ [S2・]	[S1・] = [S2・]
245	OR≤	OR（D） ≤	[S1・] ≤ [S2・]	[S1・] > [S2・]
246	OR≥	OR（D） ≥	[S1・] ≥ [S2・]	[S1・] < [S2・]

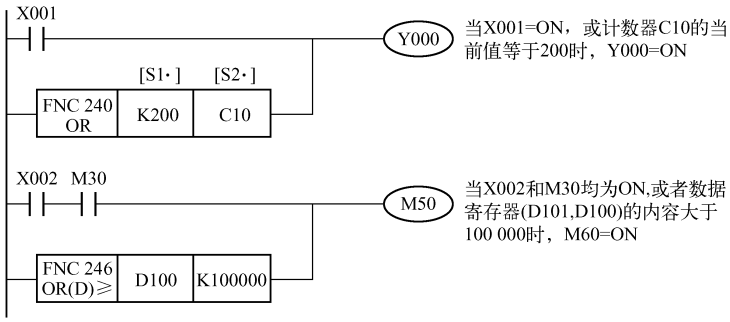


图 4-90　OR□比较触点应用示例

当源数据的最高 bit（16 bit 指令：b15；32 bit 指令：b31）为 1 时，将该数值作为负数进行比较。

32 bit 计数器（C200 ~ C234）的比较，必须以 32 bit 指令来进行。若用 16 bit 指令，会导致程序出错。

4.16　编程注意事项

4.16.1　某些指令使用次数的限制

以下指令（除注明 2 次外）只能使用 1 次：

FNC 52	MTR	FNC 60	IST	FNC 70	TKY
FNC 57	PLSY	FNC 62	ABSD	FNC 71	HKY
FNC 58	PWM	FNC 63	INCD	FNC 72	DSW（2 次）
		FNC 68	ROTC	FNC 74	SEGL（2 次）
		FNC 69	SORT	FNC 75	ARWS
				FNC 77	PR（2 次）

另外，同时 ON 驱动指令限制为 6 个以下的指令有如下几个；

FNC 53　HSCS　　FNC 54　HSCR　　FNC 55　HSZ

1. 使用执行完成标志

FX2N 的软元件中提供了特殊寄存器用于处理上列指令。这些指令中的多数在 PLC 的数个扫描周期中要根据某些特殊寄存器的瞬时状态执行一系列的操作。

比如用 FNC 72（DSW）指令，每隔 0.1 s，4 点输出自动动作一次。若动作结束前切断指令的驱动输入，动作就在此中断。再次驱动指令时动作从头开始。

为避免中断动作，使用执行完成标志 M8029，按图 4-91 所示程序驱动指令。

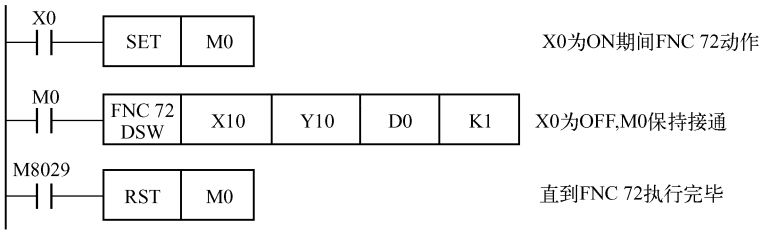


图 4-91　程序

如果这样编程，FNC 72 通过一系列操作读数字开关，但只读一次。这意味着 Y10 ~ Y13 不会连续动作，因此，即使用继电器输出型也不必担心其使用寿命。

2. 变址修饰

对于那些有使用次数限制的指令，若用变址寄存器 V、Z 对目标元件号进行修饰，就可以扩展使用范围，如图 4-92 所示。

图 4-92 中，即使由 FNC 57 产生的一系列脉冲正在输出，转换 X10 的 ON/OFF 状态也要等到这串脉冲全部输出后才能实际转变。

如果想立即中断当前操作，其方法之一就是用图 4-93 所示程序关断指令 FNC 57。

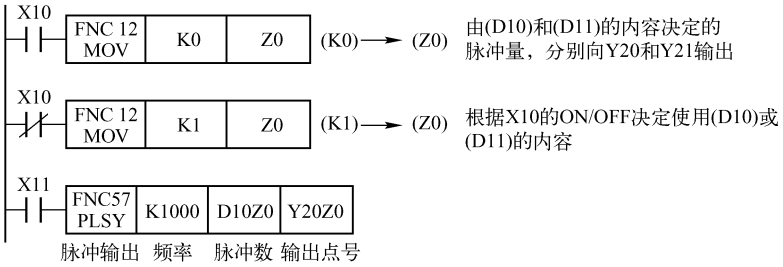


图 4-92　变址修饰示例

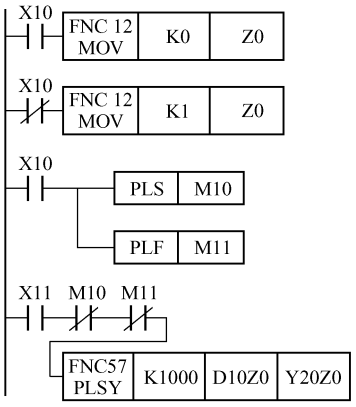


图 4-93　程序

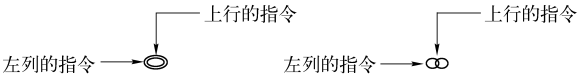
4.16.2 程序流控制指令

有关 MC－MCR 指令及 CJ 指令的相互关系见表 4-13。

表 4-13 MC－MCR 指令及 CJ 指令的相互关系

	MC－MCR	CJ－P	EI－DI	FOR－NEXT	STL－RET	P－SRET	I－IRET	FEND－END	备注
MC－MCR	◎○ 8 级嵌套	◎○	◎○	◎○	◎○	◎× (6608)	◎× (6608)	◎× (6608)	
	◎△	◎△	◎○	◎× (6607)	◎× (6605)	◎× (6606)	◎× (6606)	◎× (6608)	
CJ－P	◎○	◎○	◎○	◎○	◎○	◎△	◎△	◎× (6701)	
	◎△	◎△	◎○	◎△	◎△	◎△	◎△	◎○	
EI－DI	◎○	◎○	◎○	◎○	◎○	◎○	◎○	◎ ※1	
	◎○	◎○	◎○	◎○	◎○	◎○	◎○	◎○	
FOR－NEXT	◎× (6607)	◎○	◎○	◎○ 5 级嵌套	◎× (6607)	◎× (6607)	◎× (6607)	◎× (6607)	※1. 进入漏写 DI 指令状态，不是出错
	◎× (6607)	◎△	◎○	◎ ※3	◎× (6607)	◎× (6701)	◎× (6607)	◎× (6607)	※2.  程序如实线所示控制
STL－RET	◎× (6605)	◎△	◎○	◎○ (在一个 STL 循环内)	◎○	◎× (6605)	◎× (6605)	◎× (6605)	
	◎× (6605)	◎△	◎○	◎× (6607)	◎○	◎× (6606)	◎× (6606)	◎× (6605)	※3. FEND 和 END 为有效，停止所希望的程序流
P－SRET	◎× (6605)	◎○	◎○	◎○	◎× (6606)	◎× (6606)	◎× (6606)	◎× (6709)	除一部分例外，有包含关系的指令的组合是可能的，但是要注意：
	◎× (6608)	◎△	◎○	◎× (6607)	◎○ (6605)	◎× (6606)	◎× (6606)	◎× (6709)	① 不能在 FOR－NEXT、P－SRET、I－IRET 嵌套中使用 MC－MCR
I－IRET	◎× (6606)	◎○	◎○	◎○	◎× (6606)	◎× (6606)	◎× (6606)	◎× (6606)	② 不能在 FOR－NEXT、P－SRET、I－IRET 嵌套中使用 STL－RET
	◎× (6606)	◎△	◎○	◎× (6607)	◎× (6606)	◎× (6606)	◎× (6606)	◎× (6606)	③ MC－MCR、FOR－NEXT、P－SRET、I－IRET 指令间不能用 I、IRET、SRET、FEND、END 等切断
FEND－END	◎○	◎○	◎○	◎○	◎△	◎○	◎○	◎ ※3	
	◎× (6608)	◎× (6701)	◎△	◎× (6607)	◎× (6605)	◎× (6709)	◎× (6709)	◎ ※3	
O－FEND	◎○	◎○	◎○	◎○	◎○	◎× (6606)	◎× (6606)	◎ ※3	
	◎× (6608)	◎○	◎○	◎× (6607)	◎× (6605)	◎× (6709)	◎× (6606)	◎ ※3	
O－END (NO FEND)	◎○	◎○	◎○	◎○	◎○	◎× (6606)	◎× (6606)	◎ ※3	
	◎× (6608)	◎× (6701)	◎ ※1	◎× (6607)	◎× (6605)	◎× (6709)	◎× (6706)	◎ ※3	

注：以上表中，◎为包含关系；◎为区间前后重复的意思。



○：可以组合，无限制。

×：禁止使用的组合。(　) 中的数字为出错编号。

△：虽非禁止使用，因动作将会复杂化而建议尽量不用的组合。

4.16.3 定时扫描方式

特殊继电器 M8039 接通时即为定时扫描方式。其扫描时间由 D8039 指定。

当执行与扫描时间同步的有关指令时，如 FNC 67（RAMP）、FNC 71（HKY）或 FNC 77（PR）时，应选用定时扫描方式。

使用 FNC 71 指令时，特别应注意将扫描时间设定为大于 20 ms，因为输入滤波器的延迟时间是不可改变的。

在不需使用定时扫描方式时，请用定时器中断服务程序。

图 4-94 为采用定时器中断方式时的有关 HKY 指令的程序。

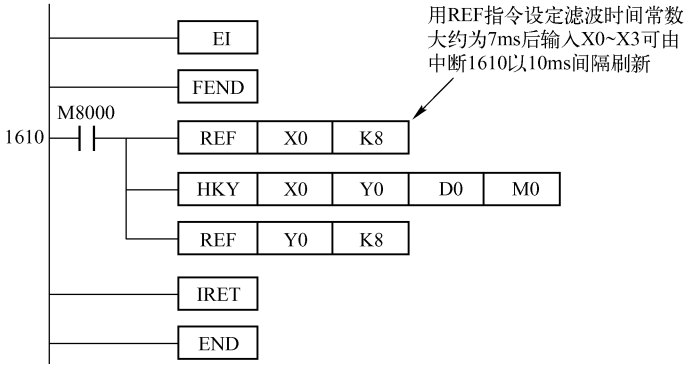


图 4-94　采用定时器中断方式时的有关 HKY 指令的程序

4.16.4 存储器保持

令特殊继电器 M8033 保持为 ON，可使 PLC 由 RUN 变为 STOP 时保持原 RUN 状态时的全部输出状态。

例如，PLC 用于控制加热器时，即使在 PLC 停机时，也可将加热器及其他设备的控制状态加以保持，直到改变程序后重新启动。

习题

1. 自动控制小车运行方向的系统，如图 4-95 所示，请根据动作要求写出梯形图及指令表，尽可能利用功能指令。

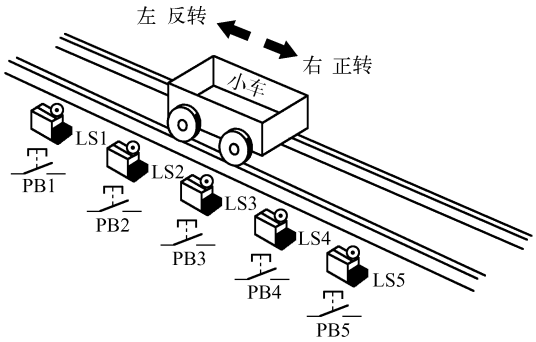


图 4-95　自动控制小车运行方向的系统

工作要求：

1）当小车所停位置 LS 的编号大于呼叫的 PB 的编号时，小车往左运行至呼叫的 PB 位置后停止。

2）当小车所停位置的编号小于呼叫的班的编号时，小车往右运行至呼叫的 PB 位置后停止。

3）小车位置 LS 与呼叫 PB 编号相同时，小车不动作。

2. 利用 PLC 做一电动机转速检测显示及控制装置，写出梯形图及指令表。

电动机上装有一转速检测装置（每转输出 10 个脉冲）。电动机转速由 PWM 输出控制。设计一程序：

1）检测电动机转速，并在 7 段码显示器上显示。

2）当检测值与给定值不同时，改变 PWM 输出，使实际转速与给定值相等。

3）加上 16 键输入电路，用于输入给定值。

3. 用 PLC 对自动售货机进行控制，根据工作要求，写出梯形图及指令表。

工作要求：

1）此售货机可投入 5 元或 10 元硬币。

2）当投入的硬币总值超过 12 元时，汽水按钮指示灯亮；又当投入的硬币总值超过 15 元时，汽水及咖啡按钮指示灯都亮。

3）当汽水按钮灯亮时，按汽水按钮，则汽水排出 7 s 后自动停止。这段时间内，汽水指示灯闪动。

4）当咖啡按钮灯亮时，按咖啡按钮，则咖啡排出 7 s 后自动停止。这段时间内，咖啡指示灯闪动。

5）若投入硬币总值超过按钮所需的钱数（汽水 12 元，咖啡 15 元）时，找钱指示灯亮，表示找钱动作，并退出多余的钱。

I/O 编号：

输入	输出
1 元识别口：X0	咖啡出口：Y10
5 元识别口：X1	汽水出口：Y11
10 元识别口：X2	咖啡按钮指示灯：Y12
咖啡按钮：X3	汽水按钮指示灯：Y13
汽水按钮：X4	找钱指示灯：Y14
计数手动复位：X5	

4. 图 4–96 所示为一定位及减速控制装置，请根据动作要求写出梯形图及指令表。

工作要求：

1）多齿凸轮与电动机同轴转动，由接近开关检测凸齿产生的脉冲信号。

2）当脉冲信号达 5 000 个时，切刀下降将材料切断。

3）电动机转至脉冲数为 4 900 时减速，为 5 000 时停止。

5. 某加工中心刀具用 PLC 进行交换控制（见图 4–97），请按要求写出梯形图及指令表。

加工中心对工件进行加工时，自动将刀具移动至换刀位置后进行换刀动作。

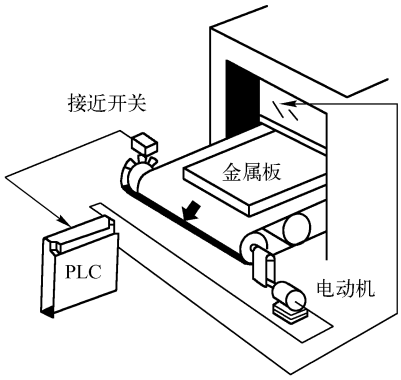


图 4-96　定位及减速控制装置

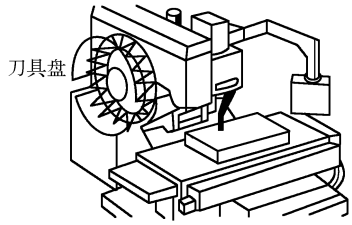


图 4-97　刀具用 PLC 进行控制

1）共有 30 种刀具供选择，按 1 ~ 30 编号。正在换刀位置上的刀具编号称为当前值，希望换上的刀具的编号称为设定值。

2）当设定值减去当前值的差大于等于 15（如 $18-2=16>15$）时，圆盘正转；当设定值减去当前值的差小于 15（如 $16-2=14<15$）时，圆盘反转。

3）当指定刀具转动至离换刀位置差两个号码时减速。

第 5 章　监控组态软件概述

随着工业自动化水平的迅速提高、计算机在工业领域的广泛应用，人们对工业自动化的要求越来越高。种类繁多的控制设备和过程监控装置在工业领域的应用，使得传统的工业控制软件已无法满足用户的各种需求。在开发传统的工业控制软件时，当工业被控对象一旦有变动，就必须修改其控制系统的源程序，导致其开发周期长；已开发成功的工控软件也由于每个控制项目的不同而使其重复使用率很低，导致它的价格非常昂贵；在修改工控软件的源程序时，倘若原来的编程人员因工作变动而要离去时，则必须同其他人员或新手进行源程序的修改，因而更是相当困难。组态软件的出现为解决上述实际工程问题提供了一种崭新的方法，因为它能够很好地解决传统工业控制软件存在的种种问题，使用户能根据自己的控制对象和控制目的任意组态，完成最终的自动化控制工程。

5.1　组态软件的含义

组态软件是一种面向工业自动化的通用数据采集和监控软件，即 SCADA（Supervisory Control And Data Acquisition）软件，亦称人机界面或 HMI/MMI（Human Machine Interface/Man Machine Interface）软件，俗称“组态软件”。

组态软件是一个约定俗成的概念，并没有明确的定义，它可以理解为“组态式监控软件”。“组态（Configure）”的含义是“配置”、“设定”、“设置”等，是指用户通过类似“搭积木”的简单方式来完成自己所需要的软件功能，而不需要编写计算机程序，也就是所谓的“组态”。它有时候也称为“二次开发”，组态软件就称为“二次开发平台”。“监控（Supervisory Control）”即“监视和控制”，是指通过计算机信号对自动化设备或过程进行监视、控制和管理。简单地说，组态软件能够实现对自动化过程和装备的监视和控制。它能从自动化过程和装备中采集各种信息，并将信息以图形化等更易于理解的方式进行显示，将重要的信息以各种手段传送到相关人员，对信息执行必要分析处理和存储，发出控制指令等。

组态软件提供了丰富的用于工业自动化监控的功能，用户根据自己工程的需要进行选择、配置等较为简单的工作来建立自己所需要的监控系统。组态软件和行业无关，它可以广泛应用于机械、钢铁、汽车、包装、矿山、水泥、造纸、水处理、环保监测、石油化工、电力、纺织、冶金、智能建筑、交通、食品、智能楼宇和实验室等凡是涉及自动化监控的任何场合。组态软件既可以完成对小型的自动化设备的集中监控，也能由互相联网的多台计算机完成复杂的大型分布式监控。还可以和工厂的管理信息系统有机整合起来，实现工厂的综合自动化和信息化。

5.2　组态软件的功能和特点

作为通用的监控软件，组态软件都能提供对工业自动化系统进行监视、控制、管理和集

成等一系列的功能。同时也为用户实现这些功能的组态过程提供了丰富和易于使用的手段和工具。利用组态软件，可以完成的常见功能有：

1）可以读写各种各样的 PLC、DCS、仪表、智能模块和板卡，采集工业现场的各种信号，从而对工业现场进行监视和控制。

2）可以以图形和动画等直观形象的方式呈现工业现场信息，以方便对控制流程的监视；也可以直接对控制系统发出指令，设置参数干预工业现场的控制流程。

3）可以将控制系统中的紧急工况（报警）通过软件界面、电子邮件、手机短信、即时消息软件、声音和计算机自动语音等多种手段及时通知给相关人员，使他们及时掌控自动化系统的运行状况。

4）可以对工业现场的数据进行逻辑运算等处理，将结果返回给控制系统，协助控制系统完成它们所不擅长的复杂的运算控制功能。

5）可以对从控制系统得到的以及自己产生的数据进行记录存储。在工程发生事故和故障的时候，利用记录的运行工况数据和历史数据，可以对系统故障原因等进行分析定位，责任追查等。通过对数据的质量统计分析，还可以提高自动化系统的运行效率，提升产品质量。

6）可以将工程运行的状况、实时数据、历史数据、警告和外部数据库中的数据以及统计运算结果制作成报表，供运行和管理人员参考。

7）可以提供多种手段让用户编写自己特殊需要的功能，操作工程中的资源，与组态软件集成为一个整体运行。大部分组态软件提供脚本来完成此功能。易控独创的 C#用户程序功能可以将用户自己开发的代码和易控的功能无缝集成到一起，将用户特需的功能集成到易控中来，对组态软件的功能进行扩展，能实现传统组态软件不能实现的新特性。

8）可以为其他应用软件提供数据，也可以接收数据，从而将不同的系统关联和整合起来。

9）多个组态软件之间可以互相联系起来，提供客户端和服务器架构，通过网络实现分布式监控，实现复杂的大系统监控。

10）可以将控制系统中的实时信息送入管理信息系统，也可以反之，接收从管理系统的管理数据，根据需要来干预生产现场或过程。

11）可以对工程的运行实现安全级别、用户级别等的安全控制。

12）可以非常简单地开发面向国际市场的、能适应多种语言界面的监控系统，可以实现工程在不同语言之间的自由灵活切换，是机电自动化和系统工程服务走向国际市场的保障。目前，组态软件能实现此功能的不多。

13）可以通过互联网发布监控系统的数据，实现远程监控。

通用组态软件主要特点：

1）延续性和可扩展性。用通用组态软件开发的应用程序，当现场（包括硬件设备和系统结构）或用户需求发生改变时，不需作很多修改而方便地完成软件的更新和升级。

2）封装性（易学易用）。通用组态软件所能完成的功能都用一种方便用户使用的方法包装起来，对于用户，不需掌握太多的编程语言技术（甚至不需要编程技术），就能很好地完成一个复杂工程所要求的所有功能。

3）通用性。每个用户根据工程实际情况，利用通用组态软件提供的底层设备（PLC、

智能仪表、智能模块、板卡和变频器等）的 I/O 驱动（Driver）、开放式的数据库和界面制作工具，就能完成一个具有动画效果、实时数据处理、历史数据和曲线并存、具有多媒体功能和网络功能的工程，不受行业限制。

4）实时多任务。例如，数据采集与输出、数据处理与算法实现、图形显示及人机对话、实时数据的存储、检索管理、实时通信等多个任务要在同一台计算机上同时运行。组态控制技术是计算机控制技术发展的结果，采用组态控制技术的计算机控制系统最大的特点是从硬件到软件开发都具有组态性，因此系统的可靠性和开发速率提高了，开发难度却下降了。组态软件的可视性和图形化管理功能也为生产管理与维护提供了方便。

5.3 组态软件的基本构成

在组态软件中，通过组态生成的一个目标应用项目在计算机硬盘中占据唯一的物理空间（逻辑空间），可以用唯一的一个名称来标识，就被称为一个应用程序。在同一计算机中可以存储多个应用程序，组态软件通过应用程序的名称来访问其组态内容，打开其组态内容进行修改或将其应用程序装入计算机内存投入实时运行。

组态软件的结构划分有多种标准，这里以使用软件的工作阶段和软件体系的成员构成两种标准讨论其体系结构。

1. 以使用软件的工作阶段划分

以使用软件的工作阶段划分也可以说是按照系统环境划分，从总体上讲，组态软件是由两大部分构成的。

系统开发环境：是自动化工程设计工程师为实施其控制方案，在组态软件的支持下进行应用程序的系统生成工作所必须依赖的工作环境。通过建立一系列用户数据文件，生成最终的图形目标应用系统，供系统运行环境运行时使用。

系统开发环境由若干个组态程序组成，如图形界面组态程序、实时数据库组态程序等。

系统运行环境：在系统运行环境下，目标应用程序被装入计算机内存并投入实时运行。系统运行环境由若干个运行程序组成，如图形界面运行程序、实时数据库运行程序等。

组态软件支持在线组态技术，即在不退出系统运行环境的情况下可以直接进入组态环境并修改组态，使修改后的组态直接生效。

自动化工程设计工程师最先接触的一定是系统开发环境，通过一定工作量的系统组态和调试，最终将目标应用程序在系统运行环境投入实时运行，完成一个工程项目。

2. 按照成员构成划分

组态软件因为其功能强大，而每个功能相对来说又具有一定的独立性，因此其组成形式是一个集成软件平台，由若干程序组件构成。

其中必备的典型组件包括：

（1）应用程序管理器　应用程序管理器是提供应用程序的搜索、备份、解压缩和建立新应用等功能的专用管理工具。在自动化工程设计工程师应用组态软件进行工程设计时，经常会遇到下面这些烦恼：经常要进行组态数据的备份；经常需要引用以往成功应用项目中的部分组态成果（如画面）；经常需要迅速了解计算机中保存了哪些应用项目。虽然这些要求可以用手工方式实现，但效率低下，极易出错。有了应用程序管理器的支持，这些操作将变

得非常简单。

（2）图形界面开发程序　是自动化工程设计工程师为实施其控制方案，在图形编辑工具的支持下进行图形系统生成工作所依赖的开发环境。通过建立一系列用户数据文件，生成最终的图形目标应用系统，供图形运行环境运行时使用。

（3）图形界面运行程序　在系统运行环境下，图形目标应用系统被图形界面运行程序装入计算机内存并投入实时运行。

（4）实时数据库系统组态程序　有的组态软件只在图形开发环境中增加了简单的数据管理功能，因而不具备完整的实时数据库系统。目前比较先进的组态软件（如组态王®等）都有独立的实时数据库组件，以提高系统的实时性，增强处理能力。实时数据库系统组态程序是建立实时数据库的组态工具，可以定义实时数据库的结构、数据来源、数据连接、数据类型及相关的各种参数。

（5）实时数据库系统运行程序　在系统运行环境下，目标实时数据库及其应用系统被实时数据库系统运行程序装入计算机内存并执行预定的各种数据计算、数据处理任务。历史数据的查询、检索、报警的管理都是在实时数据库系统运行程序中完成的。

（6）I/O 驱动程序　是组态软件中必不可少的组成部分，用于和 I/O 设备通信，互相交换数据，DDE 和 OPC Client 是两个通用的标准 I/O 驱动程序，用来与支持 DDE 标准和 OPC 标准的 I/O 设备通信。多数组态软件的 DDE 驱动程序被整合在实时数据库系统或图形系统中，而 OPC Client 则多数单独存在。

扩展可选组件包括：

（1）通用数据库接口（ODBC 接口）组态程序　通用数据库接口组件用来完成组态软件的实时数据库与通用数据库（如 Oracle、Sybase、Foxpro、DB2、Infomix、SQL Server 等）的互联，实现双向数据交换，通用数据库既可以读取实时数据，也可以读取历史数据；实时数据库也可以从通用数据库实时地读入数据。通用数据库接口（ODBC 接口）组态环境用于指定要交换的通用数据库的数据库结构、字段名称及属性、时间区段、采样周期、字段与实时数据库数据的对应关系等。

（2）通用数据库接口（ODBC 接口）运行程序　已组态的通用数据库连接被装入计算机内存，按照预先指定的采样周期，对规定时间区段按照组态的数据库结构建立起通用数据库和实时数据库间的数据连接。

（3）策略（控制方案）编辑组态程序　策略编辑/生成组件是以 PC 为中心实现低成本监控的核心软件，具有很强的逻辑、算术运算能力和丰富的控制算法。策略编辑/生成组件以 IEC 6－1131－3 标准为使用者提供标准的编程环境，共有 4 种编程方式：梯形图、结构化编程语言、指令助记符和模块化功能块。使用者一般都习惯于使用模块化功能块，根据控制方案进行组态，结束后系统将保存组态内容并对组态内容进行语法检查、编译。

编译生成的目标策略代码既可以与图形界面同在一台计算机上运行，也可以下载（Download）到目标设备（如 PC/104、Windows CE 系统等 PC－Based 设备）上运行。

（4）策略运行程序　组态的策略目标系统被装入计算机内存并执行预定的各种数据计算、数据处理任务，同时完成与实时数据库的数据交换。

（5）实用通信程序组件　实用通信程序极大地增强了组态软件的功能，可以实现与第三方程序的数据交换，是组态软件价值的主要表现之一。通信实用程序具有以下功能：

可以实现操作站的双机冗余热备用；

实现数据的远程访问和传送。

通信实用程序可以使用以太网、RS485、RS232、PSTN 等多种通信介质或网络实现其功能。实用通信程序组件可以划分为 Server 和 Client 两种类型，Server 是数据提供方，Client 是数据访问方，一但 Server 和 Client 建立起了连接，两者间就可以实现数据的双向传送。

5.4 组态软件的历史与发展方向

5.4.1 组态软件的发展史

“组态”的概念是伴随着集散型控制系统（DCS）的出现才被广大的生产过程自动化技术人员所热知的。每一套 DCS 都是比较通用的控制系统，可以应用到很多的领域中。为了使用户在不需要编写代码的情况下便可生成适合自己需求的应用系统，每个 DCS 厂商在 DCS 中都预装了系统软件和应用软件，而其中的应用软件，实际上就是组态软件，但一直没有人给出明确的定义只是将使用这种应用软设计生成目标应用系统的过程称为“组态”或“做组态”。

组态的概念最早来自英文 Configuration，含义是使用软件工具对计算机及软件的各种资源进行配置，达到让计算机或软件按照预先设置自动执行特定任务、满足用户要求的目的。监控组态软件是面向监控与数据采集（SCADA）的软件平台工具，具有丰富的设置项目，使用方式灵活，功能强大。监控组态软件最早出现时，HMI 或 MMI 是其主要内涵，即主要解决人机图形界面问题。随着它的快速发展，实时数据库、实时控制、SCADA、通信及联网、开放数据链接、对 I/O 设备的广泛支持已经成为它的主要内容，随着技术的发展，监控组态软件将会不断被赋予新的内容。

世界上第一个把组态软件作为商品进行开发销售的专业软件公司是美国的 Wonderware 公司。该公司于 20 世纪 80 年代末率先推出第一个商品化监控组态软件 InTouch，此后监控组态软件在全球得到了蓬勃发展，伴随着信息化社会的到来，监控组态软件在社会信息化进程中将扮演越来越重要的角色，每年的市场需求都比较大，未来的发展前景十分看好。

监控组态软件是伴随计算机技术的突飞猛进发展起来的。20 世纪 60 年代，虽然计算机开始涉足工业过程控制，但由于计算机技术人员缺乏工厂仪表和工业过程的知识，使得计算机控制系统在各行业的推广速率比较缓慢。20 世纪 70 年代初期，微处理器的出现促进了计算机控制走向成熟。首先，微处器在提高计算机性能的基础上，大大降低了计算机的硬件成本，缩小了计算机体积，很多从事控制仪器和原来一直就从事工业控制计算机的公司先后推出了新型控制系统。这一历史时期较有代表性的就是 1975 年美国 Honeywell 公司推出的世界上第一套 DCS－2000。随后的 20 年间，DCS 及其计算机控制技术日趋成熟，得到了广泛应用。此时的 DCS 已具有较丰富的软件，包括计算机系统软件（操作系统）、组态软件、控制软件、操作站软件、其他辅助软件（如通信软件）等。这一阶段虽然 DCS 技术、市场发展迅速，但软件仍是专用和封闭的。除了在功能上不断加强外，软件成本一直居高不下，造成 DCS 在中小型项目上的单位成本过高，使一些中小型应用项目不得不放弃使用 DCS。20 世纪 80 年代中后期，随着个人计算机的普及和开放系统（OpenSystem）概念的推广，基于个人计算机的监控系统开始进入市场，并发展壮大。组态软件作为个人计算机监控系统的重要

组成部分，比 PC 监控的硬件系统具有更为广阔的发展空间。这是因为，第一，很多 DCS 和 PLC 厂家主动公开通信协议，加入“PC 监控”的阵营。目前，几乎所有的 PLC 和一半以上的 DCS 都使用 PC 作为操作站。第二，由于 PC 监控大大降低了系统成本，使得市场空间得到扩大，从无人值守的远程监视（如防盗报警、江河汛情监视、环境监控、电信线路监控、交通管制与监控和矿井报警等）、数据采集与计量（如居民水电气表的自动抄表、铁道信号采集与记录等）、数据分析（如汽车机车自动测试、机组各参数测试、医疗化验仪器设备实时数据采集、虚拟仪器和生产线产品质量抽检等）到过程控制，几乎无处不用。第三，各类智能仪表、调节器和基于 PC 的设备可与组态软件构筑完整的低成本自动化系统，具有广阔的市场空间。第四，各类嵌入式系统和现场总线的异军突起，把组态软件推到了自动化系统主力军的位置，组态软件越来越成为工业自动化系统中的灵魂。

5.4.2 组态软件的发展方向

目前看到的所有组态软件都能完成类似的功能。比如，几乎所有运行于 32 位 Windows 平台的组态软件都采用类似资源浏览器的窗口结构，并且对工业控制系统中的各种资源（设备、标签量、画面等）进行配置和编辑；都提供多种数据驱动程序；都使用脚本语言提供二次开发的功能，等等。但是，从技术上说，各种组态软件提供实现这些功能的方法却各不相同。从这些不同之处，以及 PC 技术发展的趋势，可以看出组态软件未来发展的方向。

1. *数据采集的方式*

大多数组态软件提供多种数据采集程序，用户可以进行配置。然而，在这种情况下，驱动程序只能由组态软件开发商提供，或者由用户按照某种组态软件的接口规范编写，这对用户提出了很高的要求。由 OPC 基金组织提出的 OPC 规范基于微软的 OLE/DCOM 技术，提供了在分布式系统下，软件组件交互和共享数据的完整的解决方案。在支持 OPC 的系统中，数据的提供者作为服务器（Server），数据请求者作为客户（Client），服务器和客户之间通过 DCOM 接口进行通信，而无需知道对方内部实现的细节。由于 COM 技术是在二进制代码级实现的，所以服务器和客户可以由不同的厂商提供。在实际应用中，作为服务器的数据采集程序往往由硬件设备制造商随硬件提供，可以发挥硬件的全部效能，而作为客户的组态软件可以通过 OPC 与各厂家的驱动程序无缝链接，故从根本上解决了以前采用专用格式驱动程序总是滞后于硬件更新的问题。同时，组态软件同样可以作为服务器为其他的应用系统（如 MIS 等）提供数据。OPC 现在已经得到了包括 Interllution、Simens、GE、ABB 等国外知名厂商的支持。

随着支持 OPC 的组态软件和硬件设备的普及，使用 OPC 进行数据采集必将成为组态中更合理的选择。

2. *脚本的功能*

脚本语言是扩充组态系统功能的重要手段。因此，大多数组态软件提供了脚本语言的支持。具体的实现方式可分为三种：一是内置的类 C/Basic 语言；二是采用微软的 VBA 的编程语言；三是有少数组态软件采用面向对象的脚本语言。类 C/Basic 语言要求用户使用类似高级语言的语句书写脚本，使用系统提供的函数调用组合完成各种系统功能。应该指明的是，多数采用这种方式的国内组态软件，对脚本的支持并不完善，许多组态软件只提供 IF…THEN…ELSE 的语句结构，不提供循环控制语句，为书写脚本程序带来了一定的困难。

微软的 VBA 是一种相对完备的开发环境，采用 VBA 的组态软件通常使用微软的 VBA 环境和组件技术，把组态系统中的对象以组件方式实现，使用 VBA 的程序对这些对象进行访问。由于 VisualBasic 是解释执行的，所以 VBA 程序的一些语法错误可能到执行时才能发现。而面向对象的脚本语言提供了对象访问机制，对系统中的对象可以通过其属性和方法进行访问，比较容易学习、掌握和扩展，但实现起来比较复杂。

3. 组态环境的可扩展性

可扩展性为用户提供了在不改变原有系统的情况下，向系统内增加新功能的能力，这种增加的功能可能来自于组态软件开发商、第三方软件提供商或用户自身。增加功能最常用的手段是 ActiveX 组件的应用，目前还只有少数组态软件能提供完备的 ActiveX 组件引入功能及实现引入对象在脚本语言中的访问。

4. 组态软件的开放性

随着管理信息系统和计算机集成制造系统的普及，生产现场数据的应用已经不仅仅局限于数据采集和监控。在生产制造过程中，需要现场的大量数据进行流程分析和过程控制，以实现对生产流程的调整和优化。现有的组态软件对大部分这些方面需求还只能以报表的形式提供，或者通过 ODBC 将数据导出到外部数据库，以供其他的业务系统调用，在绝大多数情况下，仍然需要进行再开发才能实现。随着生产决策活动对信息需求的增加，可以预见，组态软件与管理信息系统或领导信息系统的集成必将更加紧密，并很可能以实现数据分析与决策功能的模块形式在组态软件中出现。

5. 对 Internet 的支持程度

现代企业的生产已经趋向国际化、分布式的生产方式。Internet 将是实现分布式生产的基础。组态软件能否从原有的局域网运行方式跨越到支持 Internet，是摆在所有组态软件开发商面前的一个重要课题。限于国内目前的网络基础设施和工业控制应用的程度，在较长时间内，以浏览器方式通过 Internet 对工业现场的监控，将会在大部分应用中停留于监视阶段，而实际控制功能的完成应该通过更稳定的技术，如专用的远程客户端、由专业开发商提供的 ActiveX 控件或 Java 技术实现。总之，基于 Web 的实时控制技术是控制技术与计算机网络技术相结合的一种新型远程监控技术，已成为当前监控组态软件的发展趋势。

6. 组态软件的控制功能

随着以工业 PC 为核心的自动控制集成系统技术的日趋完善和工程技术人员的使用组态软件水平的不断提高，用户对组态软件的要求已不像过去那样主要侧重于画面，而是要考虑一些实质性的应用功能，如软件 PLC、先进过程控制策略等。

随着企业提出的高柔性、高效益的要求，以经典控制理论为基础的控制方案已经不能适应，以多变量预测控制为代表的先进控制策略的提出和成功应用之后，先进过程控制受到了过程工业界的普遍关注。先进过程控制（Advanced Process Control，APC）是指一类在动态环境中，基于模型、充分借助计算机能力，为工厂获得最大理论而实施的运行和控制策略。先进控制策略主要有双重控制及阀位控制、纯滞后补偿控制、解耦控制、自适应控制、差拍控制、状态反馈控制、多变量预测控制、推理控制及软测量技术和智能控制（专家控制、模糊控制和神经网络控制）等，尤其智能控制已成为开发和应用的热点。目前，国内许多大企业纷纷投资，在装置自动化系统中实施先进控制。国外许多控制软件公司和 DCS 厂商

都在竞相开发先进控制和优化控制的工程软件包。据资料报道，一个乙烯装置投资163万美元实施先进控制，完成后预期可获得效益600万美元/年。从以上可以看出，能嵌入先进控制和优化控制策略的组态软件必将受到用户的极大欢迎。

7. 软PLC技术

软PLC产品是基于PC开放结构的控制装置，它具有硬PLC在功能、可靠性、速度和故障查找等方面的特点，利用软件技术可将标准的工业PC转换成全功能的PLC过程控制器。软PLC综合了计算机和PLC的开关量控制、模拟量控制、数学运算、数值处理和通信网络等功能，通过一个多任务控制内核，提供了强大的指令集、快速而准确的扫描周期、可靠的操作和可连接各种I/O系统及网络的开放式结构。所以可以这样说，软PLC提供了与硬PLC同样的功能，而同时具备了PC环境的各种优点。目前，国际上影响比较大的产品有法国CJInternational公司的ISaGRAF软件包、PCSoftInternational公司的WinPLC、美国Wizdom Control Intellution公司的Paradym－31、美国Moore Process Automation Solutions公司的ProcessSuite、美国Wonderware Controls公司的InControl、SoftPLC公司的SoftPLC等。国内推出软PLC产品的组态软件还不多，较为典型的有亚控公司的KingACT。

8. 运行时组态技术

运行时组态是组态软件新近提出的新的概念。运行时组态是在运行环境下，对已有工程进行修改，添加新的功能。它不同于在线组态，在线组态是在工程运行的同时，进入组态环境，在组态环境中对工程进行修改。而运行时组态是在运行环境中直接修改工程。运行时组态改变了以往必须进入复杂的组态环境修改工程应用的历史，给组态软件带来了新的活力，并预示着组态软件新的发展方向。

组态软件作为一种工业信息化的管理工具，其发展方向必然是不断降低工程开发工作量，提高工作效率。易用性是提高效率永恒的主题，但是提高易用性对于提高开发效率是有限的，亚控公司则率先提出通过复用来提高效率，创造性地开发出模型技术，并将这一技术集成到KingView7.0中。这一技术能将客户的工程开发周期缩短到原来的30%或更低，将组态软件为客户创造价值的能力提高到了一个新的境界，代表了组态软件的未来。组态软件的发展必将沿着更好的人机交互、更加逼真的画面、能满足客户个性化需求、具备行业特征和区域特征、具有很好的开放性、信息容易获取和更高的可靠性以及大型SCADA的方向发展。

5.5 常用的组态软件

1. iFLX

iFLX是全新模式的组态软件，思想和体系结构都比其他现有的组态软件要先进。但由于使用了微软公司很多的新技术，经常受微软公司操作系统的影响。具有功能强大的微软标准描述语言；具有标准SQL/ODBC接口，直接集成关系数据库及管理系统；具有真正的实时客户/服器模式，允许最大的规模可扩展性。

2. InTouch

InTouch有最好的图形化人机用户界面（HMI）。InTouch为以工厂为中心和以操作员为

中心的制造信息系统提供了可视化界面，使信息更加容易地在工厂内和不同工厂之间共享。

3. Citech

Citech 是组态软件中的后起之秀，在世界范围内扩展得很快。Citech 产品控制算法比较好，使用的方便性和图形功能不及 InTouch。I/O 硬件驱动相对比较少，但大部分驱动程序可随软件包提供给用户。Citech 的价格略低于 InTouch 和 iFDX。

4. WinCC

功能强大，使用较复杂。如选用西门子硬件，WinCC 是个不错的选择。

5. TraceMode

适用于分布式控制系统的开发，是俄罗斯最畅销的工业控制组态软件。其中包括分布式控制系统整体开发解决方案、方案自动建立、提供信号处理和控制的原始算法、立体矢量图形、统一网络时间和独创的管理工作站图表数据回放技术。TraceMode 是将 SCADA 和 Softlogic 集成为一体的工控软件。

6. 组态王

组态王软件是国内开发较早的软件，界面操作灵活方便，有较强的通信功能，支持的硬件也非常丰富。支持各种数据库，另外还可以选择北京亚控科技发展有限公司研发的实时数据库 KingHistorian 来保存数据。

7. 力控

它是一个面向方案的 HMI/SCADA 平台软件。它基于流行的 32 位 Windows 平台，丰富的 I/O 驱动能够连接到各种现场设备。分布式实时数据库系统，可提供访问工厂和企业系统数据的一个公共入口。内置 TCP/IP 协议的网络服务程序，可以充分利用 Intranet 或 Internet 的网络资源。力控的实时数据库系统也非常有特点。

5.6 组态王软件介绍

5.6.1 组态王概述

北京亚控科技发展有限公司是国内最早成立的专业自动化软件厂商，也是目前国内规模最大的专业自动化软件厂商。从诞生到现在，亚控一直保持着较高的增长速度。

组态王的最新版本（6.53）是亚控公司根据当前自动化技术的发展趋势，面向高端自动化市场及应用，以实现企业一体化为目标开发的一套产品。该产品以搭建战略性工业应用服务平台为目标，集成了对亚控科技自主研发的工业实时数据库（King Historian）的支持，可以为企业提供一个对整个生产流程进行数据汇总、分析及管理的有效平台，使企业能够及时有效地获取信息，及时地做出反应，以获得最优化的结果。

组态王 6.53 保持了组态王早期版本的功能强大、运行稳定且使用方便的特点，并根据国内众多用户的反馈及意见，对一些功能进行了完善和扩充。组态王 6.53 提供了丰富的简捷、易用的配置界面，提供大量的图形元素和图库精灵，同时也为用户创建图库精灵提供简单易用的接口；该款产品的历史曲线、温控曲线以及配方功能进行了大幅提升与改进，软件的功能性和可用性有了很大提高。

组态王 6.53 集成了对 King Historian 的支持，极大地提高了组态王的数据存储能力，能

够更好地满足大点数用户对存储容量和存储速度的要求。King Historian 是亚控公司新近推出的独立开发的工业数据库。具有单个服务器支持高达 10 万点、支持 256 个并发客户同时存储和检索数据、每秒检索单个变量超过 30 000 条记录的强大功能。能够更好地满足高端客户对存储速度和存储容量的要求，完全满足了客户实时查看和检索历史运行数据的要求。

5.6.2 组态王的主要功能特性

1）可视化操作界面，真彩显示图形、支持渐进色、丰富的图库、动画连接。

2）较强的动力和灵活性，拥有全面的脚本与图形动画功能。

3）可以对画面中的一部分进行保存，以便以后进行分析或打印。

4）变量导入导出功能，变量可以导出到 Excel 表格中，方便地对变量名称等属性进行修改，然后再导入新工程中，实现了变量的二次利用，节省了开发时间。

5）强大的分布式报警、事件处理，支持实时、历史数据的分布式保存。

6）强大的脚本语言处理，能够帮助你实现复杂的逻辑操作和决策处理 。

7）全新的 Web Server 架构，全面支持画面发布、实时数据发布、历史数据发布以及数据库数据的发布。

8）方便的配方处理功能。

9）丰富的设备支持库，支持常见的 PLC 设备、智能仪表、智能模块。

10）提供硬加密及软授权两种授权方式。

5.6.3 组态王 6.53 的改进

与组态王 6.51 相比较，组态王 6.53 完善和增加了以下一些主要功能：

1．画面部分

增加画面打印预览功能；添加打印预览命令语言函数，实现打印预览功能。画面的管道图素中增加流水效果，可以将流水的方向和流动的启停关联到变量，通过变量值控制流水。

增加弹出式画面类型，弹出式画面出现后，不能操作其他画面，达到类似模式对话框的效果。加上原来的两种画面类型，目前总共有三种画面类型：覆盖式、替换式和弹出式。

导航图增加关闭按钮。

2．变量定义和使用

当定义 OPC 变量时，可按照字母顺序来排序 OPC 设备的寄存器，方便找到 OPC 设备的寄存器。

支持变量引用函数使用远程变量。

增加变量引用的解除功能。

开发系统在运行配置时，增加只写变量是否在启动时下发一次数据的选择。

开发系统在运行配置时，增加写操作时是否需要数据改变才能下发的选择。

增加变量名称批量修改的功能，实现了先全部替换，然后再更新画面的功能。

增加右键变量组的中的变量模板导入导出功能。

增加一个 DataChanged 数据域，当变量值变化，该域置 1，用户负责清零。

在开发系统中增加是否将报警扩展域 1、扩展域 2 存入数据库选项。

增加一个 dataupdate 域值。当从设备上采集上数据，填充实时库时（不管数据是否变化），该值置为 true，需要手动复位。

增加批量替换中列表中还有未编辑的变量是否继续的提示功能。

增加制作水流控件时，关联一个模拟量，能够根据模拟量的大小变化改变水流的速度功能。

3. 报警和事件

可以使用分表方式记录不同时间的报警数据到报警数据库，提高报警查询速度。报警数据库表中有关时间的字段由原来的文本定义为时间类型，增加了有关时间的毫秒字段。

在开发系统中增加是否将报警扩展域1、扩展域2存入数据库选项。

4. 组态王历史数据库

对于I/O变量可按照每次采集存储历史数据。

数据变化记录增加最长不记录时间。

在组态王开发系统菜单“配置”->“历史数据记录”对话框中增加一项设置，即“存储所在磁盘空间小于__M时报警”，在其中填写数值，当历史数据存储的文件所在的磁盘空间小于设定值（表示多少M）时，弹出无模式警告对话框提示给操作人员。这时仍可正常进行历史数据的记录。当磁盘空间继续减小，当过小（通常小于50M）以致不能存储数据时，会弹出另一无模式对话框告知错误信息，提示历史记录停止运行。

增加历史数据关机时刻和通信失败时的处理。

修复历史库两个故障：变化记录中在数据长时间不变后再发生变化历史曲线无法及时查询到变化的数据；当工程太大（变量点数很多）时，可能会出现组态王运行系统退出后重新进入，历史记录服务不起作用（不能查询、不能保存数据）的现象。

5. 趋势曲线

通用历史趋势曲线控件增加功能：每条曲线可以有自己Y轴单位和范围，选中某一条曲线，显示该曲线的Y轴。Y轴坐标能够根据曲线区域数据的最大值、最小值自动调整。

个性化历史趋势曲线增加功能：Y轴坐标能够根据曲线区域数据的最大值、最小值自动调整。Y轴坐标增加可以设定实际值的功能。

工具箱上面的实时曲线Y轴坐标增加可以设定实际值的功能。

超级XY曲线，增加曲线数目到16，增加对多纵轴的支持，并提供更多的方法。

相对时间曲线控件，放大、缩小操作X轴显示误差较大，提高精度，界面显示也精确到0.001。

通用历史趋势曲线控件打印时，增加可以打印曲线图表的游标和游标显示值的功能。

通用历史趋势曲线控件可以同时显示左右游标的附加信息。

多个历史趋势控件的查询可以同步，即多个历史趋势控件显示不同的变量的数据，当改变一个历史趋势控件的查询范围（数据的起始时间）时，其他控件的查询范围也要进行相同的变化。

修复下列故障：

实时和历史趋势曲线的背景默认为黑色，原来默认的X、Y坐标颜色也为黑色，会出现看不到X、Y坐标的现象，故将默认的标识颜色从黑色改为白色。

工具箱中的历史趋势曲线中的HTSetPenName、HTGetPenName命令在连接远程变量的时候无效。

通用历史趋势曲线控件放大时有时连线有时不连线问题。

超级 XY 曲线控件增加控件方法文件保存和打印功能。

历史趋势曲线控件增加“回退按钮加上了灰色图标，使得其在不可操作时变灰”功能，并修改“曲线变量属性列表中“平均值”列中显示的数值与使用 getVALUE 控件方法得到的平均值有时不一样”问题。

在历史趋势曲线控件中加入最大值和最小值的时间显示字段。

超级 XY 曲线增加一个离散点线型。

增加温控曲线控件中的超级表格控件加入 RunTimeCanEdit 属性，使得用户在运行状态下控制编辑表格。

历史趋势曲线控件添加一个属性或函数，能够获得左右游标的年、月、日、小时、分钟、秒、毫秒时间值，实现方式为增加两个函数 GetRightVernierTimeValue( ) 和 GetLeft VernierTime-Value( )。

6. 报表

在组态王运行系统中，可固定报表的表头若干行和报表的左边若干列。

提高函数 ReportSetHisData ( )、ReportSetHisData2 ( ) 性能和可返回的行数，最大到 20 000行。

ReportSetHisData2 函数中增加是否在变量名下显示变量描述的功能。

ReportSetHisData2 函数变量配置信息保存文件下次可以不必挑选。

增加设置报表的列数和行数的功能 ReportSetColumns( )、ReportSetRows( )。

改进组态王历史库 VBA 访问接口，增加接口函数，修改 Kingreport. xls 示例，具有更加友好的查询界面。

报表创建和使用统一到行数 20 000，列数 52。

7. 增加组态王工程打包工具

可以将制定的组态王工程，以及组态王运行系统和相关驱动进行打包，在目标计算机之间安装该打包文件，不需要再安装组态王。

增加打包密码，可以设置解包密码，只有输入正确的密码才能解包。

使用打包工具安装的运行工程，像正常程序一样提供卸载功能。

8. 增加控件

组态王实时数据访问 ActiveX 控件 KvRealDbClient. ocx：该控件通过 TCP/IP 网络方式访问组态王实时库中变量的值和时间戳，通过该控件的方法和事件可以读取组态王实时库中变量的值和时间戳，也可订阅组态王实时库中变量的值和时间戳。订阅后组态王实时库中变量的值和时间戳的改变都会通知到控件的事件。为组态王的实时库的访问增加了除 OPC 和 DDE 方式外一个新的接口。

组态王实时曲线 ActiveX 控件 KvRealTimeCurves. ocx：该控件通过 TCP/IP 网络方式访问组态王实时库中变量的值，将实时数据显示在曲线上，在显示的时间范围内只每次向上描点，不变更时间轴，当当前时间超过显示的时间范围，才变更时间轴，将当前时间移到时间轴最左端原点或离最左端原点一定距离的位置。可以随时起停曲线数据的采集，可以存储加载曲线，可以打印和预览。

旋转装置显示 ActiveX 控件 KvLqmRevolver. ocx：该控件通过播放指定的序列图片模拟旋转装置的旋转效果，可以模拟旋转装置的旋转和停止，以及故障状态，解决组态王中画面旋转图形旋转的快慢受运行系统基准频率制约的限制。

网络视频控件 KvVideoHaikanServer. ocx 和 KvVideoHaikanClient. ocx：该控件通过 TCP/IP 和 UDP 网络方式可以在客户端计算机中显示服务端计算机通过视频采集卡采集和显示的视频数据。同时支持 8 个客户端对服务端多路视频的分别访问。

增加位图显示控件。

增加报警窗事件转发控件。

增加温控表格控件。

增加温控曲线控件：表格控件的主要功能是设置参数和批量下载，温控曲线控件负责显示。用户通过表格控件把设置的参数保存到数据库中，同时还可以设置参数与实际设备的寄存器的关联关系，达到批量下载的目的。

增加实时数据访问控件。

9. 新增函数及函数改进

EnableNavigate(BOOL is_enable_navigate)：使用函数禁止和开启导航图的使用。

PreviewWindow(“Window”, xScale, yScale, option, xStart, yStart)：画面打印预览命令语言函数。

EnableSaveTagValueToIniValueWhenValueChange(is_enable)：当使用上述函数并将 is_enable 设为 1，那么对于使用了保存值和保持参数特性的变量，当变量值改变时，将变量的值存储到初始化文件 tagname. db 中，下次启动运行系统时，它的初始值为上一次运行系统程序退出时的值，以前只是运行系统正常退出可保存变量的值为变量的初始值。现在不论是正常退出还是异常退出，都可保存变量的值为变量的初始值。

AckByGroupName(“station_name”,“group_name”)：可使用函数 AckByGroupName 对某个报警组的所有变量的报警进行确认，以前函数 Ack(Tagname or GroupName)；中的参数不可以为字符串变量，目前这个函数对其进行补充，可以使用字符串变量进行报警确认。

AckByTagName(“tag_name”)：可使用函数 AckByTagName 对某个变量的报警进行确认，以前函数 Ack(Tagname or GroupName)；中的参数不可以为字符串变量，目前这个函数对其进行补充，可以使用字符串变量进行报警确认。

ReportSetTime(“ReportName”, StartTime, SepTime, “szContent”)：可使用函数 ReportSetTime向报表设置连续的时间字符串，配合函数 ReportSetHisData 设置返回的历史数据的时间，参数设置和 ReportSetHisData 类似，并提高 ReportSetHisData 性能和可返回的行数，最大到 20000 行。

ReportSetColumns(“ReportName”, ColumnNum)：可以使用这个函数设置报表的列数。

ReportSetRows(“ReportName”, RowNum)：可以使用这个函数设置报表的行数。

SetRealDBForInt(“变量名”, long lvalue)：设置整型数据，可以产生操作事件。

SetRealDBForFloat(“变量名”, float lvalue)：设置实型数据，可以产生操作事件。

ReportSetHistData3()：可以控制对关机（通信失败）时期的数据在报表中显示“--”还是最后记录的数据。

GetDateLocal：获取本地日期。

GetTimeLocal：获取本地时间。

SQLUpdateCurrent：解决 SQLUpdateCurrent 对浮点数字段有时不更新的问题，注意，使用 SQLUpdateCurrent 时要求记录体关联的至少一个字段为表中的不可重复字段，并且该字段不为自动编号，不为浮点数。

PlaySound：修改的同步播放方式，以前版本同步会阻塞运行系统主程序，即在播放声音时不进行其他工作，修改后同步播放声音不阻塞主程序，每调用一次将声音文件列表保存，开启辅助线程来播放声音。Flags 增加值 5，意为停止同步播放声音，即清空当前的同步播放的声音文件列表。

ActiveXIsVisible：控制窗体控件隐含。

GetRealDBForBool：获取变量的当前实时离散值。

GetRealDBForFloat：获取变量的当前实时模拟值。

GetRealDBForInt：获取变量的当前实时整型值。

GetRealDBForString：获取变量的当前实时字符串型值。

SetRealDBForBool(“变量名”，bool bvalue)：设置离散型数据，可以产生操作事件。

SetRealDBForString：设置变量的当前实时字符串型值。

WindowSize：对运行画面最大化、最小化的操作。

ShowNavigateWindow：导航窗口的显示与隐藏控制。

GetPictureScrollXPos：获取目标画面当前窗口的左上角的 X 坐标值。

GetPictureScrollYPos：获取目标画面当前窗口的左上角的 Y 坐标值。

GetProjectPath：获取当前工程的路径。

10. 组态王内置控件改进

在 MDropComboBox. ocx 和 MSimpleComboBox. ocx 中增加用户在输入字符串时改变控件相应字符串型变量的值的功能（与使用 MEditCtrl. ocx 有类似的局限，用户需要将鼠标移到控件中，以激活控件，才可以对字符串型变量赋值）。

改进 listDeleteSelection(“ControlName”) 函数：删除一项后自动选择下一项，如果没有下一项，自动选择最后一项，并通知组态王更新与之关联的组态王字符串变量的值，如果全部删空了，与之关联的组态王字符串变量的值更新为空“”；删除不成功返回 0，成功返回 1（针对没有选中项的情况）。

增加 listGetItemCount(“ControlName”) 函数，返回控件中项目数量。

增加 listGetCurSel(“ControlName”) 函数，返回控件中当前选中的项目 ID（从 0 开始），返回值为 -1，说明当前控件没有选中项目。

增加 listSetCurSel(“ControlName”，ItemIndex) 函数，设置控件中的第 ItemIndex（从 0 开始）项为当前选中的项目，返回值说明控件中当前选择的项目 ID（从 0 开始），返回值为 -1，说明没有设置成功（ItemIndex 为负数或超过项目数）。

修复故障：内部控件没有被鼠标点亮时，内部控件的变化没有反映到组态王变量中。

11. 组态王信息窗口改进

增加另存为文本文件的功能。

增加在运行系统启动时如果提示加载协议组件失败在信息窗口中提示出具体是哪个协议组件失败，以便用户定位和解决问题的功能。如果是驱动提示具体的驱动属性，包括设备逻

辑名称、厂家、设备类型；如果是 ocx 控件，提示 ocx 控件名称。

12. web 新增功能和功能改进

增加在 web 中发布实时数据的曲线和列表。

增加在 web 中发布历史数据的曲线和列表。

增加在 web 中发布数据库数据的曲线和列表。

增加记录 web 登录事件到服务端文件中的功能。

开发中配置一批内存变量做为 IE 端的临时变量，运行中 IE 上修改这些变量时，不修改服务器上的值，开发系统在 web 发布时，增加在发布的内存变量中选择内部使用变量的功能，修改 webinfo. dat 的存储格式，运行系统也修改对 webinfo. dat 的读写，web 对内部变量的操作不影响 webserver 上的变量。

IE 上支持显示本机变量的 Quality，TimeYear，TimeMonth，TimeDate，TimeHour，TimeMinture，TimeSecond，TimeMsec，TimeZone 域值。

IE 上支持读取和设置变量的报警使能域。

修复故障：IE 上的报警窗口的 Priority 域不随运行系统而变。IE 上修改 Priority 域值，报警窗口不按照设置的优先级显示。

修复故障：做按钮命令语言：\\本站点\报警窗 1. Group = 报警组 1；IE 上点击该按钮，无效，报警窗报警显示不变。

增加了 IE 端触发音频文件的功能。

增加开发环境改变 web 报警声音文件后，客户端重新打开 IE 后，能自动下载新的声音文件并播放的功能。

增加在 web 上支持内存变量向 I/O 变量的引用功能。

增加 ReportSetHistData 在 web 发布端显示关机时的数据功能。

增加在 web 客户端可以选择显示的数据的形式（关机和不关机的数据）。

增加利用变量状态触发 web 浏览端的报警声音功能。

增加 web 上 I/O 变量的引用功能。

增加 PlaySound 函数在组态王 web 客户端的使用。

13. 菜单

增加分隔符。

字体可以修改大小。

修改组态王帮助菜单为：组态王帮助\驱动帮助\关于。

菜单改为居左。

14. 设备名称

扩展到 64 字节。

15. opc 名称

在开发系统定义 opc 设备的界面，增加订阅频率的设置。

16. 工业库

通过配置可访问的工业库服务器，可以在历史曲线控件、报表控件中显示工业库中的数据。

17. sense4 型时钟锁

增加 sense4 型时钟锁，通过对 sense4 型时钟锁的设置，实现对锁使用时间的控制。

18. 性能提升

运行系统，大画面，多图素，修改画面运行文件 *.act 的生成时建立快速索引的算法，提高 *.act 的生成速度。

开发系统，修改变量排序的算法，提高万点以上变量的加载速度。

运行系统，修改 web 发布初始化变量结构，把变量初始化放在开发系统发布时初始化，提高运行系统的启动速度。

远程站点建立，把远程站点的工程信息导入到本地，修改远程站点的工程文件比较方式，减少多次加载远程站点的工程信息，提高加载速度。

修改实时库的内存分配方式，提高变量访问的速度。

修改脚本中关于控件方法的编译方式，提高编译速度。

修改自定义函数在脚本中的编译方式，提高编译速度。

第 6 章　组态王程序设计步骤

组态王软件结构由工程管理器、工程浏览器及运行系统三部分构成。

工程管理器：工程管理器用于新工程的创建和已有工程的管理，对已有工程进行搜索、添加、备份、恢复以及实现数据词典的导入和导出等功能。

工程浏览器：工程浏览器是一个工程开发设计工具，用于创建监控画面、监控的设备及相关变量、动画连接、命令语言以及设定运行系统配置等的系统组态工具。

运行系统：工程运行界面，从采集设备中获得通信数据，并依据工程浏览器的动画设计显示动态画面，实现人与控制设备的交互操作。

通常情况下，建立一个组态王应用工程大致可分为以下几个步骤：

1）创建新工程：为工程创建一个目录，用来存放与工程相关的文件。

2）定义硬件设备并添加工程变量：添加工程中需要的硬件设备和工程中使用的变量，包括内存变量和 I/O 变量。

3）制作图形画面并定义动画连接：按照实际工程的要求绘制监控画面并使静态画面随着过程控制对象产生动态效果。

4）编写命令语言：通过脚本程序的编写以完成较复杂的操作上位控制。

5）进行运行系统的配置：对运行系统、报警、历史数据记录、网络、用户等进行设置，是系统完成用于现场前的必要工作。

6）保存工程并运行：完成以上步骤后，一个可以拿到现场运行的工程就制作完成了。

需要说明的是，这 6 个步骤并不是完全独立的，事实上，它们常常是交错进行的。在用组态王画面开发系统编制工程时，要依照此过程考虑三个方面：

图形：用户希望怎样的图形画面？也就是怎样用抽象的图形画面来模拟实际的工业现场和相应的工控设备。

数据：怎样用数据来描述工控对象的各种属性？也就是创建一个具体的数据库，此数据库中的变量反映了工控对象的各种属性，比如温度、压力等。

连接：数据和图形画面中的图素的连接关系是什么？也就是画面上的图素以怎样的动画来模拟现场设备的运行，以及怎样让操作者输入控制设备的指令。

本章将以一个电梯监控系统为例，带领大家进入组态王的世界。

6.1　创建新工程

选择菜单“文件\新建工程”或单击“新建”按钮，弹出“新建工程向导之一”对话框，如图 6-1 所示。

单击“下一步”继续，弹出“新建工程向导之二”对话框，如图 6-2 所示。

在方框内输入新建工程的存放路径，也可以单击“浏览”按钮，从下拉列表中选择一个工程路径。这里选择的是“E:\cm 的毕业设计\组态”。单击“下一步”继续，弹出“新建工程向导之三”对话框，如图 6-3 所示。

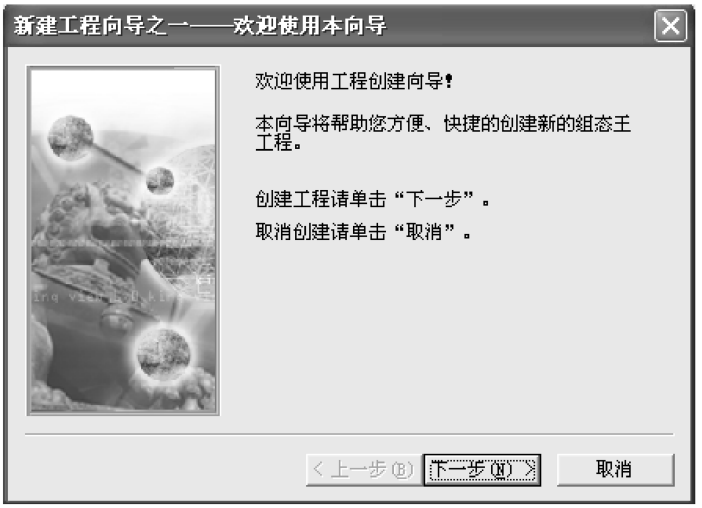


图 6-1　新建工程向导之一

图 6-2　新建工程向导之二

在工程名称的方框内输入工程名称，这里输入的是“我的工程”。在工程描述中输入对新建工程的描述，这里的描述为“电梯的组态监控”。需要注意的是，工程名称长度应小于 32 个字符，工程描述长度应小于 40 个字符。单击“完成”完成工程的新建。系统会弹出对话框，询问用户是否将新建工程设为当前工程，单击“否”按钮，则新建工程不是工程管理器的当前工程，如果要将该工程设为新建工程，还要执行“文件\设为当前工程”命令；单击“是”按钮，则将新建的工程设为组态王的当前工程。定义的工程信息会出现在工程管理器的信息表格中，如图 6-4 所示。双击该信息条或单击“开发”按钮或选择菜单“工具\切换到开发系统”，则可进入组态王的开发系统。

新建工程的路径是向导二中指定的路径，在该路径下会以工程名称为目录建立一个文件夹。完成以上操作就可以新建一个组态王工程的工程信息了。此处新建的工程，实际上并未

图 6-3　新建工程向导之三

图 6-4　工程管理器

真正创建工程，只是在用户给定的工程路径下设置了工程信息，当用户将此工程作为当前工程，并且切换到组态王开发环境时才真正创建工程。

6.2　定义设备与变量

组态王把那些需要与之交换数据的设备或程序都作为外部设备。外部设备包括下位机（PLC、仪表、模块、板卡、变频器等），它们一般通过串行口和上位机交换数据；其他 Windows 应用程序，它们之间一般通过 DDE 交换数据；外部设备还包括网络上的其他计

算机。

只有在定义了外部设备之后，组态王才能通过 I/O 变量和它们交换数据。为方便定义外部设备，组态王设计了“设备配置向导”，引导用户一步步完成设备的连接。

6.2.1 外部设备的定义

组态王支持的硬件设备包括：可编程序控制器（PLC）、智能模块、板卡、智能仪表，变频器等。工程人员可以把每一台下位机看成一种设备，不必关心具体的通信协议，只需在组态王的设备库中选择设备的类型，然后按照“设备配置向导”的提示一步步完成安装即可，使驱动程序的配置更加方便。

工程人员根据设备配置向导就可以完成设备的配置，操作步骤如下：

在工程浏览器的目录显示区，用鼠标左键单击大纲项设备下的成员板卡，则在目录内容显示区出现“新建”图标，如图 6-5 所示。

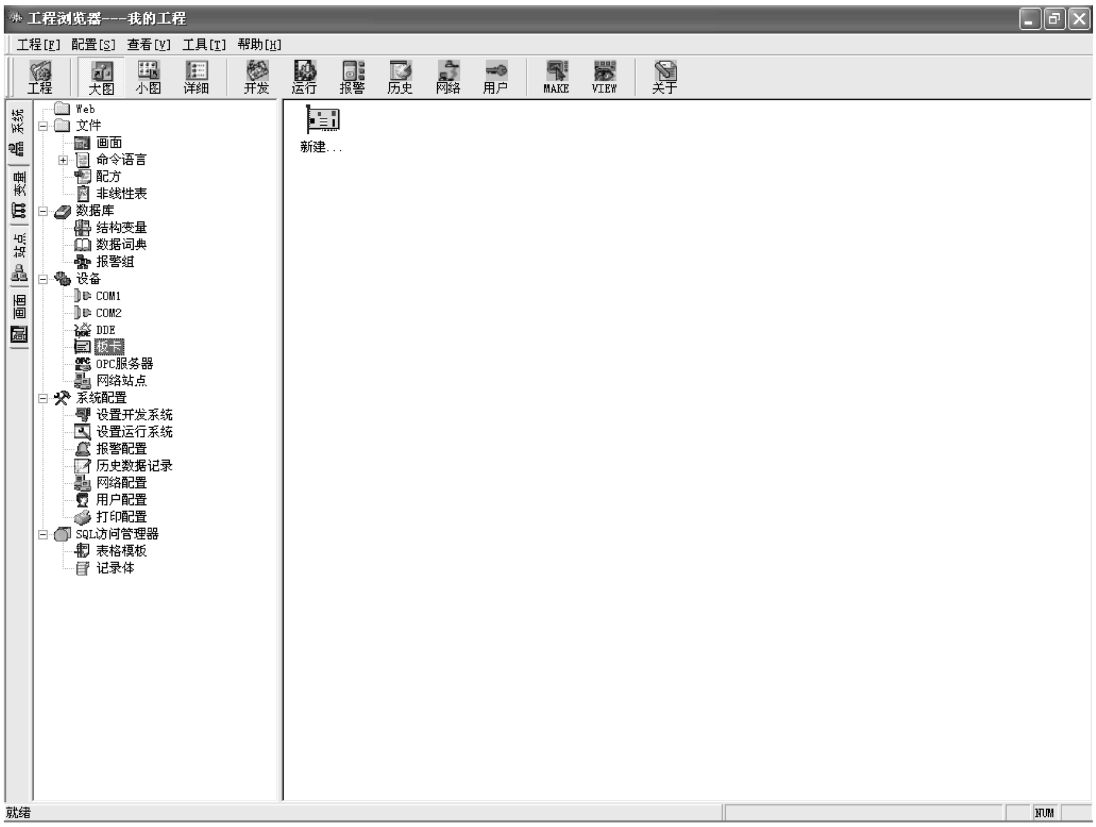


图 6-5　工程浏览器——板卡建立

用左键双击“新建”图标后，弹出“设备配置向导”列表对话框；或者用右键单击，则弹出浮动式菜单，选择菜单命令“新建板卡”，也弹出“设备配置向导”列表对话框，如图 6-6 所示。

从树形设备列表区中选择 PLC 节点。然后选择要配置板卡设备的生产厂家、设备名称。如“PLC/三菱/FX2/编程口”。单击“下一步”按钮，则弹出如下“设备配置向导——逻辑名称”对话框，如图 6-7 所示。

图 6-6　设备配置向导——生产厂家

图 6-7　设备配置向导——逻辑名称

工程人员给要配置的板卡设备指定一个逻辑名称。继续单击“下一步”按钮，弹出如下“设备配置向导——选择串口号”对话框，如图 6-8 所示。

选择连接的串口为 COM1，单击“下一步”按钮，弹出“设备配置向导——设备地址设置指南”对话框，如图 6-9 所示。

图 6-8　设备配置向导——选择串口号

图 6-9　设备配置向导——设备地址设置指南

此处填写设备地址为 2，单击“下一步”，弹出“通信参数”对话框，如图 6-10 所示。

此处采用默认值，单击“完成”按钮，则弹出如下“设备安装向导——信息总结”对话框，如图 6-11 所示。

请检查各项设置是否正确，确认无误后，单击“完成”。

设备定义完成后，您可以在 COM1 项下看到新建的设备“FX2N”。

图 6-10　通信参数

图 6-11　设备配置向导——信息总结

6.2.2　数据变量类型说明和定义

在组态王中，变量的集合形象地称为“数据词典”，数据词典记录了所有用户可使用的数据变量的详细信息。而“数据词典”就位于组态王工程浏览器中的“数据库”项的下拉列表中。

数据词典中存放的是应用工程中定义的变量以及系统变量。变量可以分为基本类型和特殊类型两大类，基本类型的变量又分为内存变量和 I/O 变量两种。I/O 变量指的是组态王与外部设备或其他应用程序交换的变量。这种数据交换是双向的、动态的，就是说在组态王系统运行过程中，每当 I/O 变量的值改变时，该值就会自动写入外部设备或远程应用程序；每当外部设备或远程应用程序中的值改变时，组态王系统中的变量值也会自动改变。所以，那些从下位机采集来的数据、发送给下位机的指令，都需要设置成“I/O 变量”。那些不需要和外部设备或其他应用程序交换，只在组态王内使用的变量，比如计算过程的中间变量，就可以设置成“内存变量”。

基本类型的变量也可以按照数据类型分为离散型、实型、整型和字符串型。

1. 内存离散变量、I/O 离散变量

类似一般程序设计语言中的布尔（BOOL）变量，只有 0、1 两种取值，用于表示一些开关量。

2. 内存实型变量、I/O 实型变量

类似一般程序设计语言中的浮点型变量，用于表示浮点数据，取值范围为 10E－38～10E＋38，有效值为 7 位。

3. 内存整数变量、I/O 整数变量

类似一般程序设计语言中的有符号长整数型变量，用于表示带符号的整型数据，取值范围为 2147483648～2147483647。

4. 内存字符串型变量、I/O 字符串型变量

类似一般程序设计语言中的字符串变量，可用于记录一些有特定含义的字符串，如名称、密码等，该类型变量可以进行比较运算和赋值运算。

特殊变量类型有报警窗口变量、报警组变量、历史趋势曲线变量、时间变量 4 种。这几种特殊类型的变量正是体现了组态王系统面向工控软件、自动生成人机接口的特色。

变量基本属性的说明：

变量名：唯一标识一个应用程序中数据变量的名字，同一应用程序中的数据变量不能重名，数据变量名区分大小写，最长不能超过 32 个字符。用鼠标单击编辑框的任何位置进入编辑状态，工程人员此时可以输入变量名字，变量名可以是汉字或英文名字，第一个字符不能是数字。例如，温度、压力、液位、var1 等均可以作为变量名。变量的名称（包括结构变量）最多为 31 个字符。

变量类型：在对话框中只能定义 8 种基本类型中的一种，用鼠标单击变量类型下拉列表框列出可供选择的数据类型，当定义有结构变量时，一个结构就是一种变量类型。

描述：此项编辑框用于编辑和显示数据变量的注释信息。若想在报警窗口中显示某变量的描述信息，可在定义变量时，在描述项编辑框中加入适当说明，并在报警窗口中加上描述项，则在运行系统的报警窗口中可见该变量的描述信息（最长不超过 39 个字符）。

变化灵敏度：数据类型为模拟量或长整型时此项有效。只有当该数据变量的值变化幅度超过“变化灵敏度”时，组态王才更新与之相连的图素（默认值为 0）。

最小值：指示该变量值在数据库中的下限。

最大值：指示该变量值在数据库中的上限。注意，组态王中最大的精度为 float 型，4 字

节。定义最大值时注意不要越限。

最小原始值：指示前面定义的最小值所对应的输入寄存器值的下限。

最大原始值：指示前面定义的最大值所对应的输入寄存器值的上限。

保存参数：在系统运行时，修改变量的域的值（可读可写型），系统自动保存这些参数值，系统退出后，其参数值不会发生变化。当系统再启动时，变量的域的参数值为上次系统运行时最后一次的设置值。无须用户再去重新定义。变量域的说明请查看在线帮助。

保存数值：系统运行时，当变量的值发生变化后，系统自动保存该值。当系统退出后再次运行时，变量的初始值为上次系统运行过程中变量值最后一次变化的值。

初始值：这项内容与所定义的变量类型有关，定义模拟量时出现编辑框可输入一个数值，定义离散量时出现开或关两种选择。定义字符串变量时出现编辑框可输入字符串，它们规定软件开始运行时变量的初始值。

对于我们的电梯监控系统，可以得到表 6–1 所示的变量列表。

表 6–1　组态变量表

变量名	变量类型	连接设备	寄存器名
1 层上呼按钮	I/O 离散	FX_{2N}	X3
2 层上呼按钮	I/O 离散	FX_{2N}	X4
2 层下呼按钮	I/O 离散	FX_{2N}	X5
3 层下呼按钮	I/O 离散	FX_{2N}	X6
1 层内选按钮	I/O 离散	FX_{2N}	X0
2 层内选按钮	I/O 离散	FX_{2N}	X1
3 层内选按钮	I/O 离散	FX_{2N}	X2
开门按钮	I/O 离散	FX_{2N}	X7
关门按钮	I/O 离散	FX_{2N}	X10
报警按钮	I/O 离散	FX_{2N}	X12
门的开度	内存整型	—	—
升降程度	内存整型	—	—
1 层行程开关	I/O 离散	FX_{2N}	X13
2 层行程开关	I/O 离散	FX_{2N}	X14
3 层行程开关	I/O 离散	FX_{2N}	X15
1 层指示灯	I/O 离散	FX_{2N}	Y2
2 层指示灯	I/O 离散	FX_{2N}	Y3
3 层指示灯	I/O 离散	FX_{2N}	Y4
报警指示	I/O 离散	FX_{2N}	Y7
电梯上升	I/O 离散	FX_{2N}	Y0
电梯下降	I/O 离散	FX_{2N}	Y1
电梯开门	I/O 离散	FX_{2N}	Y5
电梯关门	I/O 离散	FX_{2N}	Y6

（续）

变量名	变量类型	连接设备	寄存器名
开门到位	I/O 离散	FX_{2N}	X16
关门到位	I/O 离散	FX_{2N}	X17
1 层内选回应	I/O 离散	FX_{2N}	Y10
2 层内选回应	I/O 离散	FX_{2N}	Y11
3 层内选回应	I/O 离散	FX_{2N}	Y12
1 层上呼回应	I/O 离散	FX_{2N}	Y13
2 层上呼回应	I/O 离散	FX_{2N}	Y14
2 层下呼回应	I/O 离散	FX_{2N}	Y15
3 层下呼回应	I/O 离散	FX_{2N}	Y16
开门按钮回应	I/O 离散	FX_{2N}	Y17
关门按钮回应	I/O 离散	FX_{2N}	Y20
游标	内存实型	—	—
报表查询变量	内存字符串	—	—
跨度	内存整型	—	—
卷动百分比	内存整型	—	—

6.3 制作图形画面

进入组态王开发系统后，就可以为每个工程建立数目不限的画面，在每个画面上生成互相关联的静态或动态图形对象。这些画面都是由组态王提供的类型丰富的图形对象组成的。系统为用户提供了矩形（圆角矩形）、直线、椭圆（圆）、扇形（圆弧）、点位图、多边形（多边线）、文本等基本图形对象，及按钮、趋势曲线窗口、报警窗口、报表等复杂的图形对象。提供了对图形对象在窗口内任意移动、缩放、改变形状、复制、删除、对齐等编辑操作，全面支持键盘、鼠标绘图，并可提供对图形对象的颜色、线型、填充属性进行改变的操作工具。

组态王采用面向对象的编程技术，使用户可以方便地建立画面的图形界面。用户构图时，可以像搭积木那样利用系统提供的图形对象完成画面的生成。同时支持画面之间的图形对象复制，可重复使用以前的开发结果。

6.3.1 建立新画面

为建立一个新的画面请执行以下操作：在工程浏览器左侧的“工程目录显示区”中选择“画面”选项，在右侧视图中双击“新建”图标，弹出新建画面对话框，如图 6–12 所示。

在画面名称对话框中输入新建画面名称，如“电梯的组态监控”。其他属性默认，单击确定后如图 6–13 所示。

其中，画面的大小可以通过拉动右下角的点来改变。

在工具箱中单击文本工具，在画面上输入文字：“电梯组态监控”。

单击“工具”菜单，在下拉列表中选择“多边形”，然后在画面中绘制一个平行四边形，并复制几个待用，同样地，在“工具”菜单的下拉列表中选择“立体管道”，在画面中绘制一个适当长度的管道，并复制待用。

图 6-12　新画面的建立

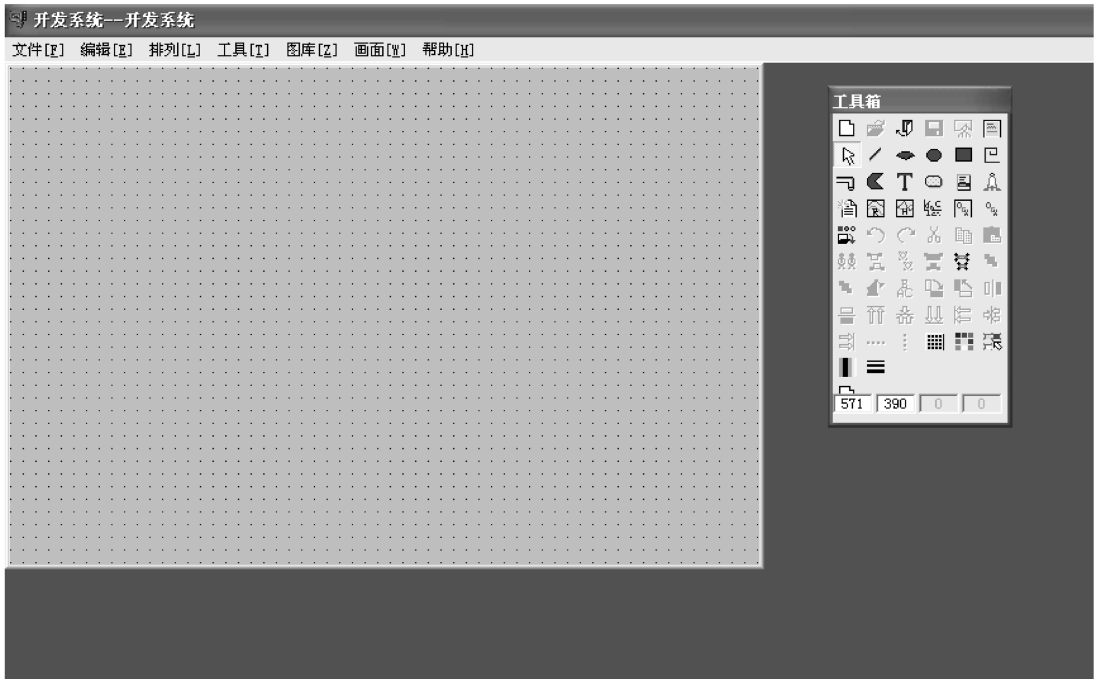


图 6-13　开发系统

将“多边形”和“立体管道”组合起来，如图 6-14 所示。注意多边形之间的距离相等，以及图素前后移动的灵活运用，以达到令人满意的效果。这样电梯的简单框图就绘制完成了。

接下来是电梯轿厢画面的绘制，在这里为了方便可以只画电梯轿厢的三个面，即正面、侧面和上面，以及电梯门的绘制。具体绘制方法与电梯框架的绘制方法相同，只是矩形和平行四边形的简单组合。在这里为了模拟时的效果更好，可以在门的处理上多加一层白色的方

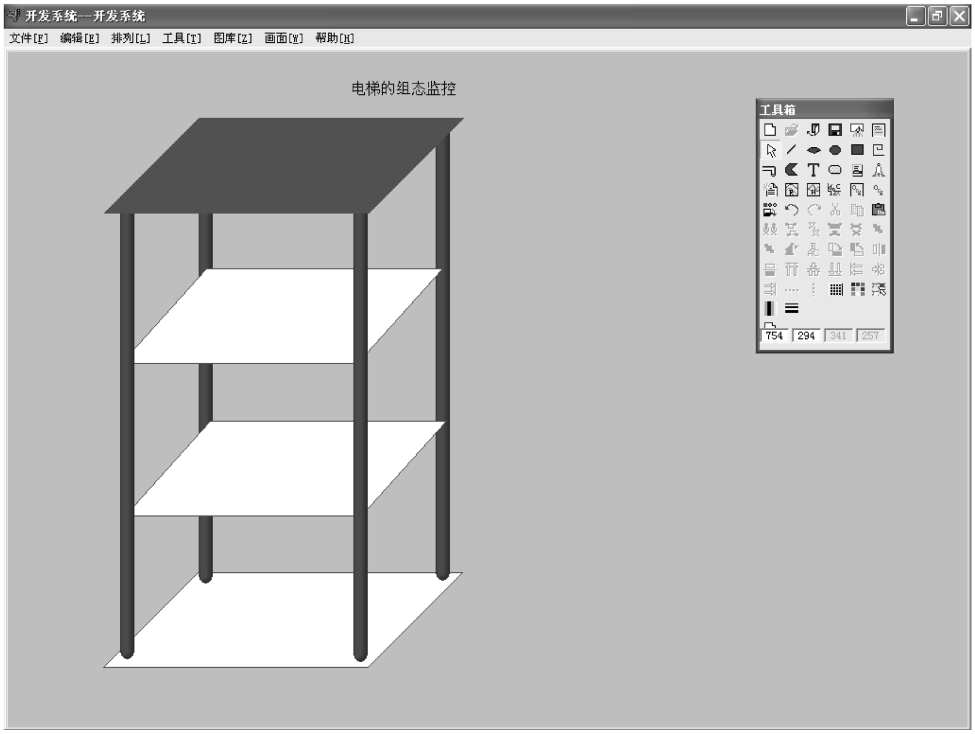


图 6-14　电梯井架

框，大小与两扇门的大小一致，同时注意图素位置的确定（即确定哪个图素在最上面，哪个图素在最下面）。绘制完后效果如图 6-15 所示。

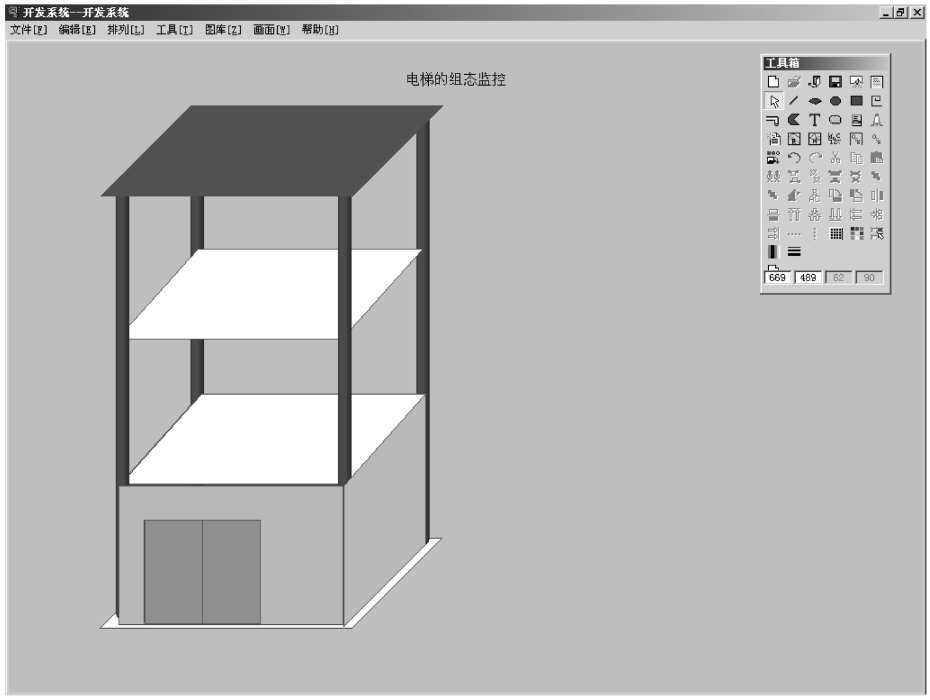


图 6-15　电梯井架和轿厢

6.3.2 使用图库管理器

电梯的框架和轿厢完成以后就只剩下按钮的绘制了，在组态软件中，为了方便用户的使用，专门设计了一个图库管理器，在图库管理器中有一些已制作成型的图素组合。使用图库管理器降低了工程人员设计界面的难度，用户更加集中精力于维护数据库和增强软件内部的逻辑控制，缩短开发周期；同时用图库开发的软件将具有统一的外观，方便工程人员学习和掌握；另外利用图库的开放性，工程人员可以生成自己的图库元素。

选择“图库”菜单中“打开图库”命令或按 F2 键打开图库管理器，如图 6-16 所示。

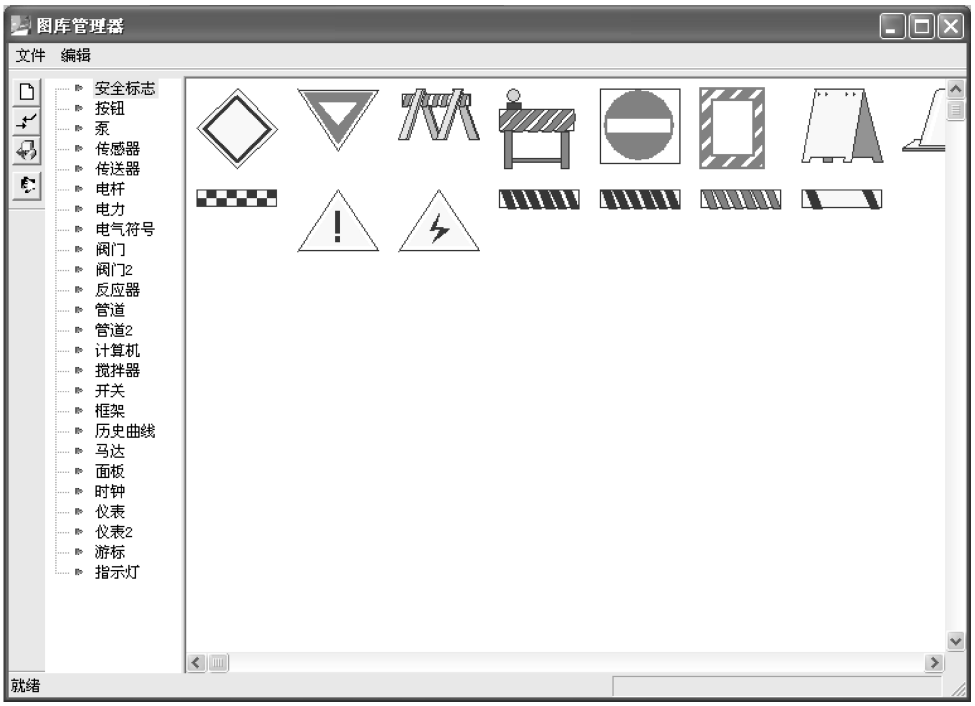


图 6-16 图库管理器

在图库管理器左侧图库名称列表中选择图库名称“按钮”，选中↓（和↑）后双击鼠标，图库管理器自动关闭，在工程画面上鼠标位置出现“L”标志，在画面上单击鼠标，该图素就被放置在画面上并拖动边框到适当的位置，改变其至适当的大小。

考虑到完成后的按钮与我们的要求有一定的距离，我们把按钮稍加变化。具体方法是，先选中我们的按钮，然后单击“图库”菜单，在它的下拉菜单中选择“转换成普通图素”，就能把按钮中的向下（和向上）的箭头拖出来，留着备用。

在工具箱中找到“按钮”，并单击，在画面的适当位置拖动出一个适当大小的方框，出现一个按钮，它的默认文本为“文本”，可以通过右击该按钮选择“字符串替换”来改变按钮的文本属性。在这里我们把它改变成“ <| > ”（表示电梯的开门按钮），这样就完成了开门按钮的制作，把它复制几个，分别改变文本为“ >| < ”（表示电梯的关门按钮）、“1”（表示电梯的 1 层内选按钮）、“2”（表示电梯的 2 层内选按钮）、“3”（表示电梯的 3 层内选按钮）、“!”（表示电梯的警告按钮），同时再复制 4 个，改变文本为“ ”，与上面拖出的向上和向下的箭头组合起来表示电梯的上呼和下呼按钮，在这里要把箭头图素向前移动，在

这里，虽然画面上看与图库中的元素没有差别，但是它们的属性发生了变化，通过改变它的属性可以达到我们的要求。

同样地，对游标进行相同的操作，以达到我们的设计要求。

完成了按钮和游标的制作后，电梯监控系统画面如图 6–17 所示。

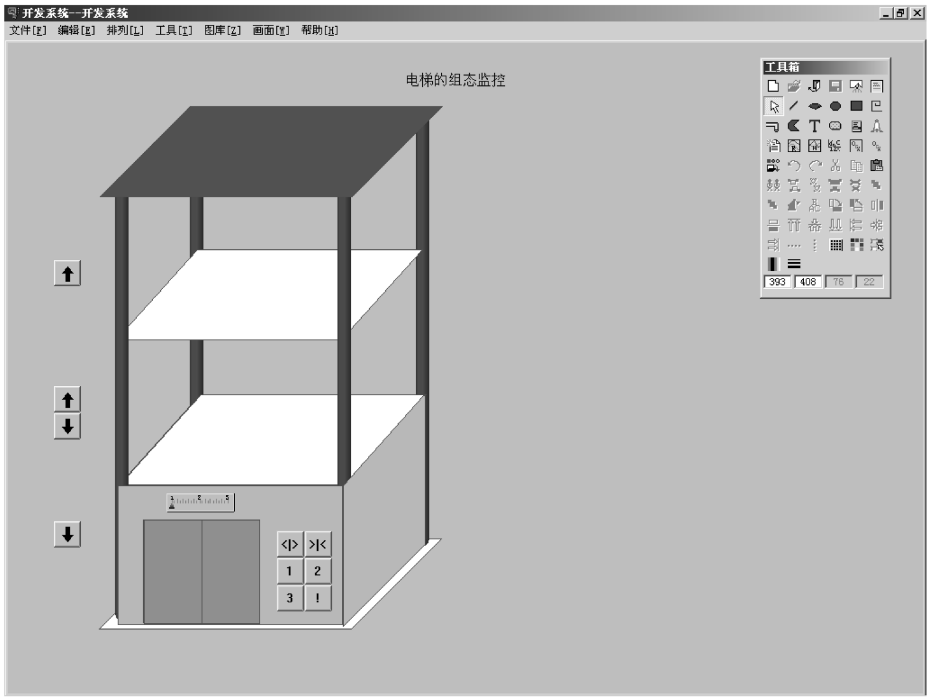


图 6–17　电梯基本画面

这样，电梯的基本画面部分就完成了。

6.4　建立动画连接

所谓“动画连接”就是建立画面的图素与数据库变量的对应关系。

工程人员在组态王开发系统中制作的画面都是静态的，那么它们如何才能反映工业现场的状况呢？这就需要通过实时数据库，因为只有数据库中的变量才是与现场状况同步变化的。数据库变量的变化又如何导致画面的动画效果呢？通过“动画连接”，所谓“动画连接”就是建立画面的图素与数据库变量的对应关系。这样，工业现场的数据，比如温度、液面高度等，当它们发生变化时，通过 I/O 接口，将引起实时数据库中变量的变化，如果设计者曾经定义了一个画面图素，比如指针与这个变量相关，我们将会看到指针在同步偏转。

动画连接的引入是设计人机接口的一次突破，它把工程人员从重复的图形编程中解放出来，为工程人员提供了标准的工业控制图形界面，并且由可编程的命令语言连接来增强图形界面的功能。图形对象与变量之间有丰富的连接类型，给工程人员设计图形界面提供了极大的方便。组态王系统还为部分动画连接的图形对象设置了访问权限，这对于保障系统的安全具有重要的意义。图形对象可以按动画连接的要求改变颜色、尺寸、位置、填充百分数等，

一个图形对象又可以同时定义多个连接。把这些动画连接组合起来，应用程序将呈现出令人难以想象的图形动画效果。

给图形对象定义动画连接是在“动画连接”对话框中进行的。在组态王开发系统中双击图形对象（不能有多个图形对象同时被选中），弹出动画连接对话框，如图 6-18 所示。

图 6-18　动画连接

对话框的第一行标识出被连接对象的名称和左上角在画面中的坐标以及图形对象的宽度和高度。

对话框的第二行提供“对象名称”和“提示文本”编辑框。“对象名称”是为图素提供的唯一的名称，供以后的程序开发使用，暂时不能使用。“提示文本”的含义为：当图形对象定义了动画连接时，在运行的时候，鼠标放在图形对象上，将出现开发中定义的提示文本。

对话框的下面部分为动画连接种类，在本次设计中，要用到如下几种动画连接：垂直动画连接、水平动画连接、缩放动画连接、填充属性连接。

6.4.1　垂直动画连接

垂直移动连接是使被连接对象在画面中的位置随连接表达式的值而垂直移动。移动距离以像素为单位，以被连接对象在画面制作系统中的原始位置为参考基准的。垂直移动连接常用来表示对象实际的垂直运动，如本次设计中的轿厢部分。单击“动画连接”对话框中的“垂直移动”按钮，弹出“垂直移动连接”对话框，如图 6-19 所示。

对话框中各项设置的意义如下：

表达式：在此编辑框内输入合法的连接表达式，单击“?”按钮可以查看已定义的变量名和变量域。在本次设计中我们选为“\\本站点\升降程度”。

向上：输入图素在垂直方向向上移动（以被连接对象在画面中的原始位置为参考基准）的距离。在本次设计中我们选为“320”。

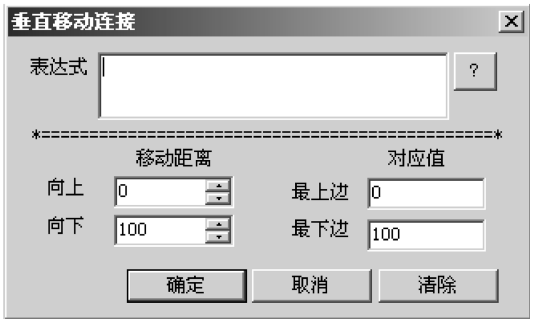


图 6-19　垂直移动连接

最上边：输入与图素处于最上边时相对应的变量值，当连接表达式的值为对应值时，被连接对象的中心点向上（以原始位置为参考基准）移到最上边规定的位置。在本次设计中我们选为“320”。

向下：输入图素在垂直方向向下移动（以被连接对象在画面中的原始位置为参考基准）的距离。在本次设计中我们选为“0”。

最下边：输入与图素处于最下边时相对应的变量值，当连接表达式的值为对应值时，被连接对象的中心点向下（以原始位置为参考基准）移到最下边规定的位置。在本次设计中我们选为“0”。

6.4.2　水平动画连接

水平移动连接是使被连接对象在画面中随连接表达式值的改变而水平移动。移动距离以像素为单位，以被连接对象在画面制作系统中的原始位置为参考基准的。水平移动连接常用来表示图形对象实际的水平运动，如轿厢中游标的指针。

水平移动连接的设置方法为：在“动画连接”对话框中单击“水平移动”按钮，弹出“水平移动连接”对话框，如图 6-20 所示。

图 6-20　水平移动连接

对话框中各项设置的意义如下：

表达式：在此编辑框内输入合法的连接表达式，单击“?”按钮可查看已定义的变量名和变量域。在本次设计中我们选为“\\本站点\游标”。

向左：输入图素在水平方向向左移动（以被连接对象在画面中的原始位置为参考基准）的距离。在本次设计中我们选为“1”。

最左边：输入图素处于最左边时相对应的变量值，当连接表达式的值为对应值时，被连接对象的中心点向左（以原始位置为参考基准）移到最左边规定的位置。在本次设计中我们选为“1”。

向右：输入图素在水平方向向右移动（以被连接对象在画面中的原始位置为参考基准）的距离。在本次设计中我们选为“3”。

最右边：输入图素处于最右边时相对应的变量值，当连接表达式的值为对应值时，被连接对象的中心点向右（以原始位置为参考基准）移到最右边规定的位置。在本次设计中我们选为“3”。

6.4.3 缩放动画连接

缩放连接是使被连接对象的大小随连接表达式的值而变化，如本次设计中的电梯门的设计（将其设置“缩放连接”动画连接属性），以反映变量“门的开度”的变化。

缩放连接的设置方法是：在“动画连接”对话框中单击“缩放连接”按钮，弹出对话框，如图 6-21 所示。

图 6-21 缩放连接

对话框中各项设置的意义如下：

表达式：在此编辑框内输入合法的连接表达式，单击“?”按钮可以查看已定义的变量名和变量域。在本次设计中我们选为“\\本站点\门的开度”。

最小时：输入对象最小时占据的被连接对象的百分比（占据百分比）及对应的表达式的值（对应值）。百分比为 0 时，此对象不可见。在本次设计中我们选对应值为“100”，占据百分比为“0”。

最大时：输入对象最大时占据的被连接对象的百分比（占据百分比）及对应的表达式的值（对应值）。若此百分比为 100，则当表达式值为对应值时，对象大小为制作时该对象大小。在本次设计中我们选对应值为“0”，占据百分比为“100”。

变化方向：选择缩放变化的方向。变化方向共有 5 种，用“方向选择”按钮旁边的指示器来形象地表示。箭头是变化的方向，蓝点是参考点。单击“方向选择”按钮，可选择 5 种变化方向之一。


向下变化


向上变化


向中心变化


向左变化


向右变化

显然，在设计左边的门时应选用“  ”；在设计右边的门时应选用“  ”。

6.4.4 填充属性连接

填充属性连接使图形对象的填充颜色和填充类型随连接表达式的值而改变，通过定义一些分段点（包括阈值和对应填充属性），使图形对象的填充属性在一段数值内为指定值。在本次设计中，PLC 接收到按钮给出的信号之后又返回一个信号，在这个信号的作用下会使相应的按钮持续点亮。这样的效果就是填充属性连接。

“填充属性”动画连接的设置方法为：在“动画连接”对话框中选择“填充属性”按钮，弹出的对话框如图 6-22 所示。

图 6-22　填充属性连接

“填充属性”动画连接对话框的各项意义如下：

表达式：用于输入连接表达式，右边的“?”可以查看已定义的变量名和变量域。选成各个按钮回应，如“\\本站点\一层上呼回应”和“\\本站点\一层内选回应”。

修改：修改选中的分段点。单击修改按钮弹出修改对话框，如图 6-23 所示。

图 6-23　修改

在本次设计中，把阈值设置为“0”和“1”分别对应按钮无效和按钮有效时的按钮颜色。

增加：增加新的分段点。增加对话框用法同修改对话框。

删除：删除选中的分段点。

6.5 命令语言编程

命令语言都是靠事件触发执行的，如定时、数据的变化、键盘键的按下、鼠标的点击等。在本次设计中用到了应用程序命令语言、动画连接命令语言、事件命令语言。组态王中命令语言在语法上类似 C 语言。

6.5.1 应用程序命令语言

在工程浏览器的目录显示区，选择“文件 \ 命令语言 \ 应用程序命令语言”，则在右边的内容显示区出现“请双击这儿进入 < 应用程序命令语言 > 对话框…”图标，如图 6-24 所示。

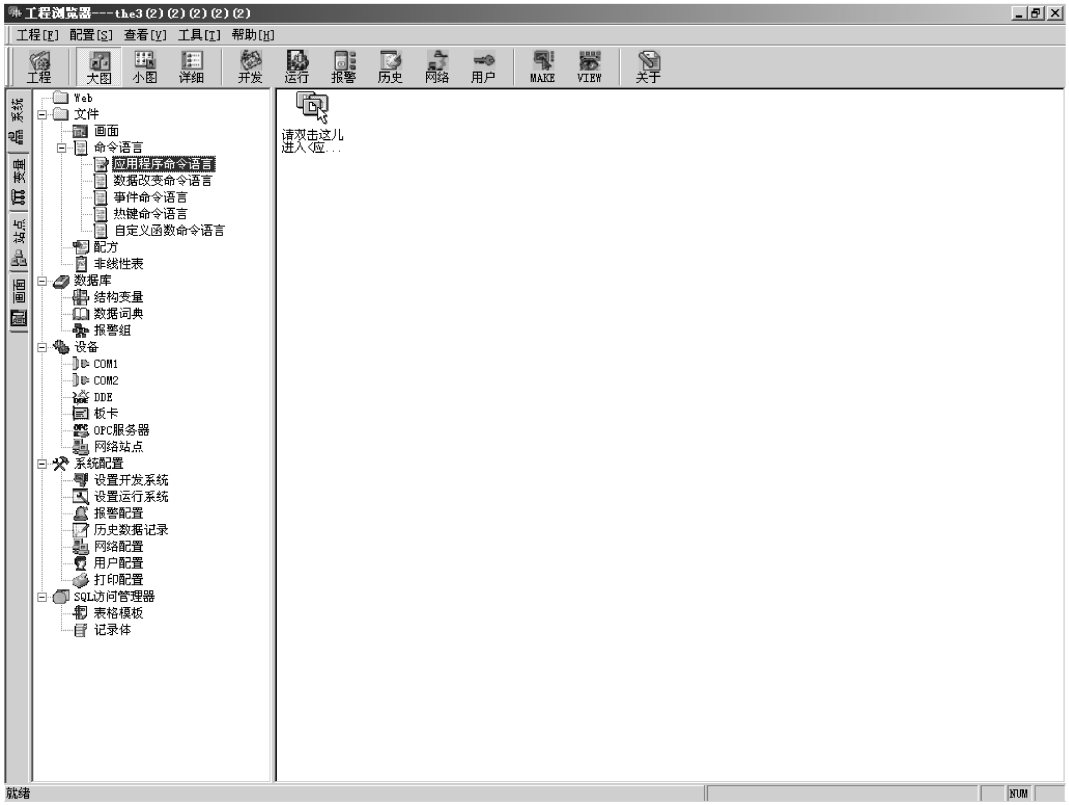


图 6-24 工程浏览器 - 应用命令语言

双击图标，则弹出“应用程序命令语言”对话框，如图 6-25 所示。

应用程序命令语言是指在组态王运行系统应用程序启动时、运行期间和程序退出时执行的命令语言程序。如果是在运行系统运行期间，该程序按照指定时间间隔定时执行。

如图 6-25 所示，当选择“运行时”标签时，会有输入执行周期的编辑框“每……毫秒”。输入执行周期，则组态王运行系统运行时，无论打开画面与否，将按照该时间周期性的执行这段命令语言程序。

图 6-25　应用程序命令语言

选择“启动时”标签，在该编辑器中输入命令语言程序，该段程序只在运行系统程序启动时执行一次。

选择“停止时”标签，在该编辑器中输入命令语言程序，该段程序只在运行系统程序退出时执行一次。

应用程序命令语言只能定义一个。

结合本次设计，我们只需要选择“运行时”标签，在命令输入区输入“\\本站点\游标 = \\本站点\升降程度/160 + 1;”。这里的游标是电梯内部的指示游标，即代表电梯的现在楼层。如此就能把电梯的升降程度这个比较抽象的概念具体化到楼层。

6.5.2　事件命令语言

事件命令语言是指当规定的表达式的条件成立时执行的命令语言。如某个变量等于定值，某个表达式描述的条件成立。在工程浏览器中选择命令语言——事件命令语言，在浏览器右侧双击“新建……”，弹出事件命令语言编辑器，如图 6-26 所示。

事件命令语言有三种类型：

发生时：事件条件初始成立时执行一次。

存在时：事件存在时定时执行，在“每……毫秒”编辑框中输入执行周期，则当事件条件成立存在期间，周期性执行命令语言。

消失时：事件条件由成立变为不成立时执行一次。

事件描述：指定命令语言执行的条件。

备注：对该命令语言作一些说明性的文字。

在本次设计中，我们需要 4 组事件命令语言的定义，分别为

图 6-26　事件命令语言

{事件描述:\\本站点\电梯上升 ==1
存在时:\\本站点\升降程度 = \\本站点\升降程度 +5;
每 55 ms}
{事件描述:\\本站点\电梯下降 ==1
存在时:\\本站点\升降程度 = \\本站点\升降程度 -5;
每 55 ms}
{事件描述:\\本站点\开门 ==1&&\\本站点\门的开度 <100
存在时:\\本站点\门的开度 = \\本站点\门的开度 +25;
每 55 ms}
{事件描述:\\本站点\关门 ==1&&\\本站点\门的开度 >0
存在时:\\本站点\门的开度 = \\本站点\门的开度 -25;
每 55 ms}

6.5.3　动画连接命令语言

对于图素，有时一般的动画连接表达式完成不了的工作，而程序只需要点击一下画面上的按钮等图素才执行，如点击一个按钮，执行一连串的动作，或执行一些运算、操作等。这时可以使用动画连接命令语言。组态王中的大多数图素都可以定义动画连接命令语言。如在画面上放置一个按钮，双击该按钮，弹出动画连接对话框，如图 6-18所示。

在“命令语言连接”选项中包含三个选项：

按下时：当鼠标在该按钮上按下时，或与该连接相关联的热键按下时执行一次。

弹起时：当鼠标在该按钮上弹起时，或与该连接相关联的热键弹起时执行一次。

按住时：当鼠标在该按钮上按住，或与该连接相关联的热键按住，没有弹起时周期性执行该段命令语言。按住时命令语言连接可以定义执行周期，在按钮后面的“毫秒”标签编辑框中输入按钮被按住时命令语言执行的周期。

单击上述任何一个按钮都会弹出动画连接命令语言编辑器，如图 6-27 所示。其用法与其他命令语言编辑器用法相同。

图 6-27　动画连接命令语言

在本次设计中，我们用到“弹起时”有效，在各组命令语言中只要输入相应的按钮等于 1 即可。如“\\本站点\一层内选按钮 = 1;”，“\\本站点\二层下呼按钮 = 1;”等。

6.6　程序运行

组态王软件包由工程管理器 ProjectManage、工程浏览器 TouchExplorer 和画面运行系统 TouchVew 三部分组成。其中工程浏览器内嵌组态王画面制作开发系统，生成人机界面工程。画面制作开发系统中设计开发的画面工程在 TouchVew 运行环境中运行。TouchExplorer 和 TouchVew 各自独立，一个工程可以同时被编辑和运行，这对于工程的调试是非常方便的。本章主要通过 TouchVew 的菜单命令来介绍组态王画面运行系统。

在运行组态王工程之前，首先要在开发系统中对运行系统环境进行配置。在开发系统中单击菜单栏“配置 \ 运行环境”命令或工具条“运行”按钮或工程浏览器“工程目录显示区 \ 系统配置 \ 设置运行系统”按钮后，弹出“运行系统设置”对话框，如图 6-28 所示。

在本次设计中只需要选择“主画面配置”，再选中“电梯仿真”画面，点击“确定”。这样就把“电梯仿真”画面作为了主画面，即进入运行系统后首先跳出的是“电梯仿真”画面。

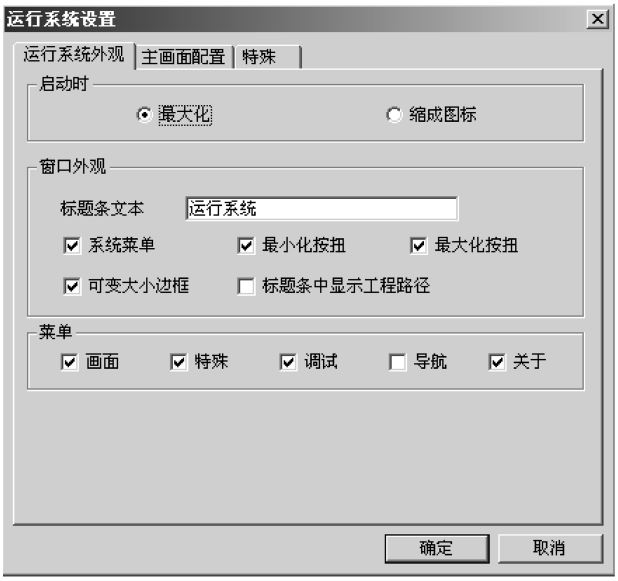


图 6-28　运行系统设置

配置好运行系统之后，就可以启动运行系统环境了。在开发系统中单击工具条“View”按钮或快捷菜单中“切换到 View”命令后，进入组态王运行系统，如图 6-29 所示。

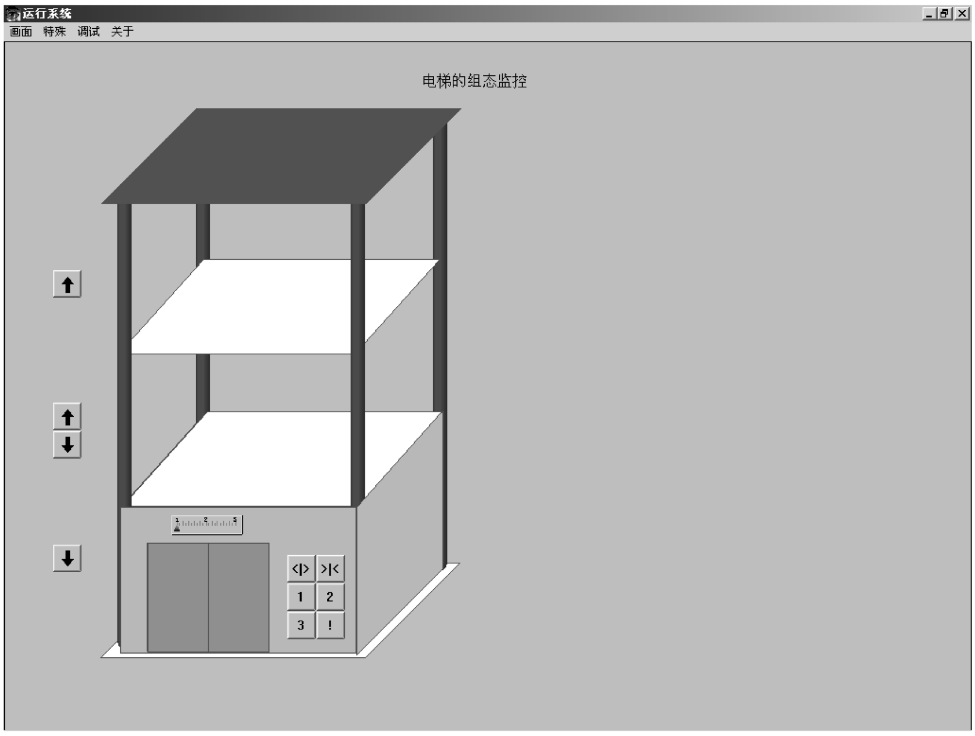


图 6-29　电梯运行画面

这样电梯组态监控部分的基本功能就完成了，接下来要做的是 PLC 的编程和硬件电路的制作。

第 7 章　组态王的深入学习

通过第 6 章的学习，相信大家已经具备了组态王应用的基础。在本章，将带领大家继续深入地学习组态王的相关知识，为大家能够自由灵活地应用组态王打下坚实的基础。

7.1　报警的配置和使用

为保证工业现场安全生产，报警和事件的产生和记录是必不可少的，组态王提供了强有力的报警和事件系统。

组态王中的报警和事件主要包括变量报警事件、操作事件、用户登录事件和工作站事件。通过这些报警和事件，用户可以方便地记录和查看系统的报警和各个工作站的运行情况。当报警和事件发生时，在报警窗中会按照设置的过滤条件实时地显示出来。

为了分类显示产生的报警和事件，可以把报警和事件划分到不同的报警组中，在指定的报警窗口中显示报警和事件信息。

报警配置基本步骤包括：

1）定义报警组；

2）设置变量的报警定义属性；

3）建立报警和事件窗口；

4）建立新画面；

5）绘制报警和事件窗口；

6）报警窗口的操作；

7）报警和事件的输出；

8）文件输出、打印输出、数据库输出等。

7.1.1　建立报警窗口

1．定义报警组

1）在工程浏览器窗口左侧“工程目录显示区”中选择“数据库”中的“报警组”选项，在右侧“目录内容显示区”中双击“进入报警组”图标，弹出“报警组定义”对话框，如图 7–1 所示。

2）单击“修改”按钮，将名称为“RootNode”报警组改名为“化工厂”。

3）选中“化工厂”报警组，单击“增加”按钮，增加此报警组的子报警组，名称为反应车间。

4）单击“确认”按钮关闭对话框，结束对报警组的设置，如图 7–2 所示。

注意：报警组的划分以及报警组名称的设置由用户根据实际情况指定。

2．设置变量的报警属性

1）在数据词典中选择“原料油液位”变量，双击此变量，在弹出的“定义变量”对话

图 7-1　“报警组定义”对话框

图 7-2　设置完毕的报警组窗口

框中单击“报警定义”选项卡，如图 7-3 所示。

设置如下：

报警组名：反应车间/液位

低：10　　原料油液位过低

高：90　　原料油液位过高

优先级：100

2）设置完毕后单击“确定”按钮，系统进入运行状态时，当“原料油液位”的高度低于 10 或高于 90 时系统将产生报警，报警信息将显示在“反应车间”报警组中。

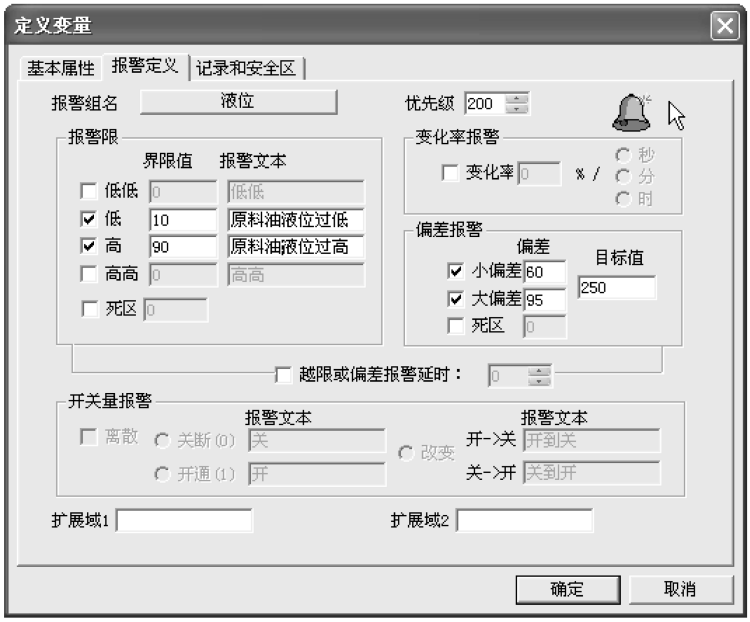


图 7-3　报警属性定义窗口对话框

3. 建立报警窗口

报警窗口是用来显示组态王系统中发生的报警和事件信息，报警窗口分为实时报警窗口和历史报警窗口。实时报警窗口主要显示当前系统中发生的实时报警信息和报警确认信息，一旦报警恢复后将从窗口中消失。历史报警窗口中显示系统发生的所有报警和事件信息，主要用于对报警和事件信息进行查询。

报警窗口建立过程如下：

1）新建一画面，名称为报警和事件画面，类型为覆盖式。

2）选择工具箱中的T工具，在画面上输入文字：报警和事件。

3）选择工具箱中的工具，在画面中绘制一报警窗口，如图 7-4 所示。

图 7-4　报警窗口

4）双击“报警窗口”对象，弹出报警窗口配置属性页，如图 7-5 所示。

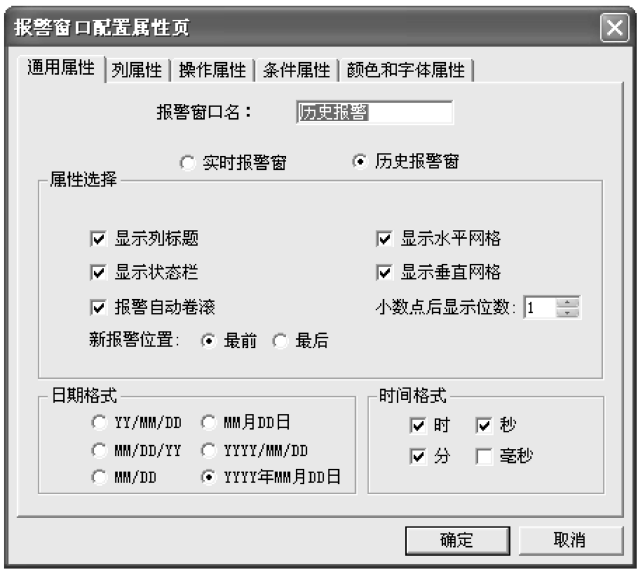


图 7–5　报警窗口配置属性页

报警窗口分为 5 个属性页，即通用属性页、列属性页、操作属性页、条件属性页、颜色和字体属性页。

通用属性页：在此属性页中您可以设置窗口的名称、窗口的类型（实时报警窗口或历史报警窗口）、窗口显示属性以及日期和时间显示格式等。

注意：报警窗口的名称必须填写，否则运行时将无法显示报警窗口。

列属性页：报警窗口中的列属性页窗口，如图 7–6 所示。

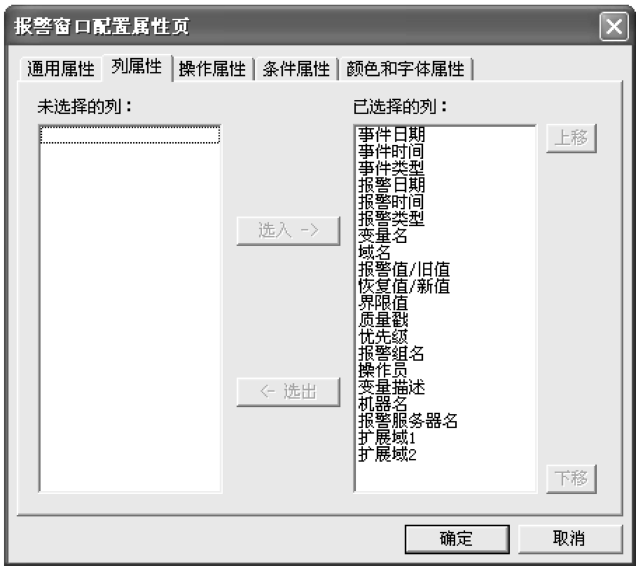


图 7–6　列属性页窗口

在此属性页中您可以设置报警窗中显示的内容，包括报警日期时间显示与否、报警变量名称显示与否、报警限值显示与否、报警类型显示与否等。

操作属性页：报警窗口中的操作属性页窗口，如图 7-7 所示。

图 7-7　操作属性页窗口

在此属性页中您可以对操作者的操作权限进行设置。单击“安全区”按钮，在弹出的“选择安全区”对话框中选择报警窗口所在的安全区，只有登录用户的安全区包含报警窗口的操作安全区时，才可执行如下设置的操作，如双击左键操作、工具条的操作和报警确认的操作。

条件属性页：报警窗口中的条件属性页窗口，如图 7-8 所示。

图 7-8　条件属性页窗口

在此属性页中，您可以设置哪些类型的报警或事件发生时才在此报警窗口中显示，并设置其优先级和报警组。

可将优先级设为 999，报警组设为反应车间。这样设置完后，满足如下条件的报警点信息会显示在此报警窗口中：

① 在变量报警属性中设置的优先级高于 999；

② 在变量报警属性中设置的报警组名为反应车间；

颜色和字体属性页：报警窗口中的颜色和字体属性页窗口，如图 7-9 所示。

图 7-9　颜色和字体属性页窗口

在此属性页中，您可以设置报警窗口的各种颜色以及信息的显示颜色。

报警窗口的上述属性可由用户根据实际情况进行设置。

5）单击“文件”菜单中的“全部存”命令，保存您所做的设置。

6）单击“文件”菜单中的“切换到 View”命令，进入运行系统。系统默认运行的画面可能不是您刚刚编辑完成的“报警和事件画面”，您可以通过运行界面中“画面”菜单中的“打开”命令将其打开后方可运行，如图 7-10 所示。

图 7-10　运行中的报警窗口

4. 报警窗口的操作

当系统处于运行状态时，用户可以通过报警窗口上方的工具箱对报警信息进行操作，如图 7-11 所示。

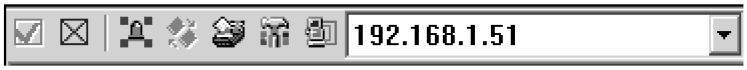


图 7-11　报警信息操作工具箱

报警确认：确认报警窗中当前选中的未经过确认的报警信息；

报警删除：删除报警窗中所有当前选中的报警信息；

更改报警类型：单击该按钮，在弹出的列表框中选择当前报警窗要显示的报警类型，选择完毕后，从当前开始，报警窗只显示符合选中报警类型的报警，但不影响其他类型报警信息的产生；

更改事件类型：选择当前报警窗要显示的事件类型；

更改优先级：选择当前报警窗的报警优先级；

更改报警组：选择当前报警窗要显示的报警组；

更改站点名：选择当前报警窗要显示哪个工作站站点的事件信息；

更改报警服务器名：选择当前报警窗要显示哪个报警服务器的报警信息。

注：只有登录用户的权限符合操作权限时才可操作此工具箱。

5. 报警窗口自动弹出

使用系统提供的“$新报警”变量可以实现当系统产生报警信息时将报警窗口自动弹出，操作步骤如下：

1）在工程浏览窗口中的“工程目录显示区”中选择“命令语言”中的“事件命令语言”选项，在右侧“目录内容显示区”中双击“新建”图标，弹出“事件命令语言”编辑框，设置如图 7-12 所示。

2）单击“确认”按钮关闭编辑框。当系统有新报警产生时，即可弹出报警窗口。

7.1.2　报警信息的输出

对于系统中的报警和事件信息，不仅可以输出到报警窗口中，还可以输出到文件、数据库和打印机中。此功能可通过报警配置属性窗口来实现，配置过程如下：

在工程浏览器窗口左侧的“工程目录显示区”中双击“系统配置”中的“报警配置”选项，弹出报警配置属性页，如图 7-13 所示。

报警配置属性页分为 3 个属性页：文件配置属性页、数据库配置属性页、打印配置属性页。

文件配置属性页：在此属性页中，您可以设置将哪些报警和事件记录到文件中以及记录的格式、记录的目录、记录的时间、记录哪些报警组的报警信息等，文件记录格式如下：

图 7-12　事件命令语言编辑框

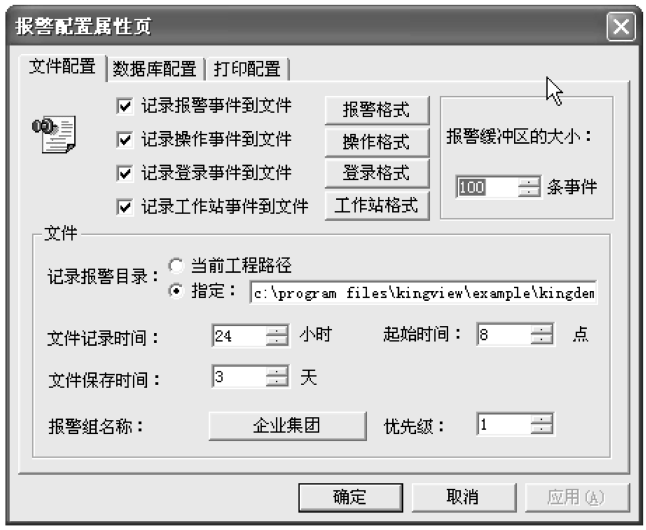


图 7-13　报警配置属性页

示例：工作站事件文件记录：

[工作站日期：2001 年 4 月 28 日][工作站时间：14 时 24 分 7 秒][事件类型：工作站启 动][机器名：本站点]

[工作站日期：2001 年 4 月 28 日][工作站时间：14 时 24 分 14 秒][事件类型：工作站 退出][机器名：本站点]

注意：这里提到的“文件”是组态王定义的内部文件。

数据库配置属性页：数据库配置属性页，如图 7-14 所示。

图 7–14　数据库配置属性页

在此属性页中，您可以设置将哪些报警和事件记录到数据库中以及记录的格式、数据源的选择、登录数据库时的用户名和密码等等。

打印配置属性页：打印配置属性页，如图 7–15 所示。

图 7–15　打印配置属性页

在此属性页中，您可以设置将哪些报警和事件输出到打印机中以及打印的格式、打印机的端口号等，打印输出格式如下：

示例：工作站事件打印：

<工作站日期：2001 年 4 月 28 日 >/<工作站时间：14 时 24 分 7 秒 >/<事件类型：工作站启动 >/<机器名：本站点 >

<工作站日期：2001 年 4 月 28 日 >/<工作站时间：14 时 24 分 14 秒 >/<事件类型：工作站 退出 >/<机器名：本站点 >

注意：建议用户在使用打印设置时，使用带字库的针式打印机。

练一练：

1）完善你的练习工程，对报警组、变量进行相关的配置。

2）在画面中得到报警的显示输出。

3）将报警记录到文件中。

4）将报警记录到数据库中。

7.2 曲线的应用

趋势曲线用来反应变量随时间的变化情况。趋势分析是控制软件必不可少的功能，组态王对该功能提供了强有力的支持和简单的控制方法。趋势曲线有实时趋势曲线和历史趋势曲线两种。在曲线显示坐标系中，X 轴代表时间，Y 轴代表变量值。对于实时趋势曲线，最多可显示 4 条曲线；而历史趋势曲线最多可显示 16 条曲线，而一个画面中可定义数量不限的趋势曲线（实时趋势曲线或历史趋势曲线）。在趋势曲线中，工程人员可以规定时间间距、数据的数值范围、网格分辨率、时间坐标数目、数值坐标数目以及绘制曲线的“笔”的颜色属性。画面程序运行时，实时趋势曲线可以自动卷动，以快速反应变量随时间的变化；历史趋势曲线不能自动卷动，它一般与功能按钮一起工作，共同完成历史数据的查看工作。这些按钮可以完成翻页、设定时间参数、启动｜停止记录、打印曲线图等复杂功能。

7.2.1 实时趋势曲线

在组态王开发系统中制作画面时（比如“我的工程”的“实时趋势曲线”画面），选择菜单“工具｜实时趋势曲线”项或单击工具箱中的“画实时趋势曲线”按钮，此时鼠标在画面中变为“+”字形，在画面中用鼠标画出一个矩形，实时趋势曲线就在这个矩形中绘出，如图 7-16 所示。实时趋势曲线对象的中间有一个带有网格的绘图区域，表示曲线将

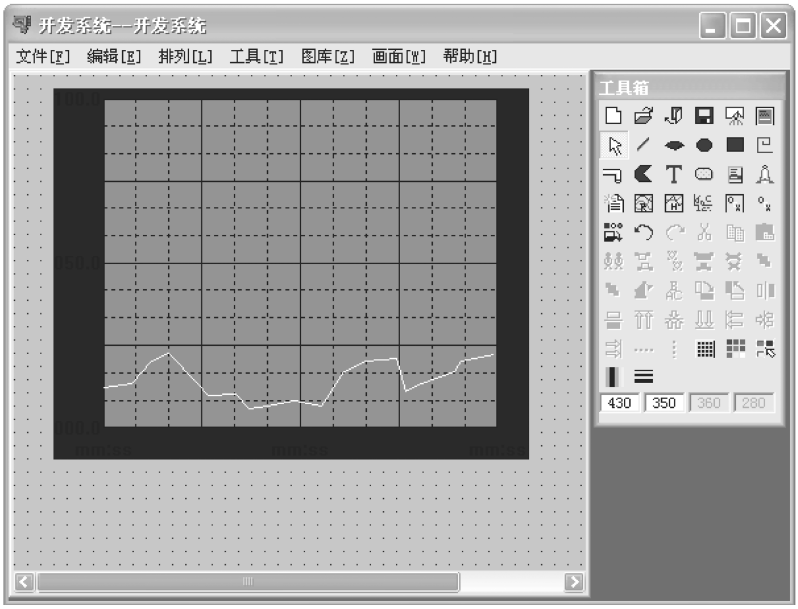


图 7-16 实时趋势曲线

在这个区域中绘出，网格左方和下方分别是 X 轴（时间轴）和 Y 轴（数值轴）的坐标标注。可以通过选中实时趋势曲线对象（周围出现 8 个小矩形）来移动位置或改变大小。在画面运行时实时趋势曲线对象由系统自动更新。

在生成实时趋势曲线对象后，要对实时趋势曲线对象进行参数设置。双击此对象，弹出“实时趋势曲线”对话框，如图 7-17 所示。本对话框通过单击对话框上端的两个按钮在“曲线定义”和“标识定义”之间切换。

图 7-17　“实时趋势曲线”对话框

1. “曲线定义”属性页

“曲线定义”属性页中的各项含义如下：

1）坐标轴：选择坐标轴的线型、线宽和显示颜色。

2）分割线为短线：选择分割线的类型。选中此项后，在坐标轴上只有很短的主分割线，整个图纸区域接近空白状态，没有网格，同时下面的“次/分割线”选择项变灰。

3）边框色、背景色：分别规定绘图区域的边框和背景（底色）的颜色。选择这两个按钮的方法与坐标轴按钮类似，弹出的浮动对话框也与之大致相间，只是没有线型选项。

4）X 轴方向、Y 轴方向：X 轴方向和 Y 轴方向的主分割线将绘图区划分成矩形网格，次分割线将再次划分主分割线划分出来的小矩形，这两种线都可改变线型和颜色。分割线的数目可以通过小方框右边的“加减”按钮增加或减小，也可通过编辑区直接输入。工程人员可以根据实时趋势曲线的大小决定分割线的数目，分割线最好与标识定义（标注）相对应。

5）曲线：定义所绘的 1 ~4 条曲线 Y 坐标轴对应的表达式，实时趋势曲线可以实时计算表达式的值，所以它可以使用表达式。实时趋势曲线名的编辑框中可输入有效的变量名或表达式，表达式中所用变量必须是数据库中已定义的变量。单击右边的“?”按钮可列出数据库中已定义的变量或变量域供选择。每条曲线可通过右边的“线型”和“颜色”按钮来

改变线型和颜色。

2. “标识定义”属性页

在图 7-17 中，单击“标识定义”属性页，弹出的对话框如图 7-18 所示。其中各项含义如下：

1）标识 X 轴——时间轴、标识 Y 轴——数值轴：选择是否为 X 轴或 Y 轴加标识，即在绘图区域的外面用文字标注坐标的数值。如果此项选中，左边的检查框中有小叉标记，同时下面定义相应标识的选择项也由灰变亮。

2）数值轴（Y 轴）定义区：因为一个实时趋势曲线可以同时显示 4 个变量的变化，而各变量的数值范围可能相差很大，为使每个变量都能表现清楚，“组态王”中规定，变量在 Y 轴上以百分数表示，即以变量值与变量范围（最大值与最小值之差）的比值表示。所以 Y 轴的范围是 0（0%）~1（100%）。

3）标识数目：数值轴标识的数目，这些标识在数值轴上等间隔。

4）起始值：规定数值轴起点对应的百分比值，最小为 0。

5）最大值：规定数值轴终点对应的百分比值，最大为 100。

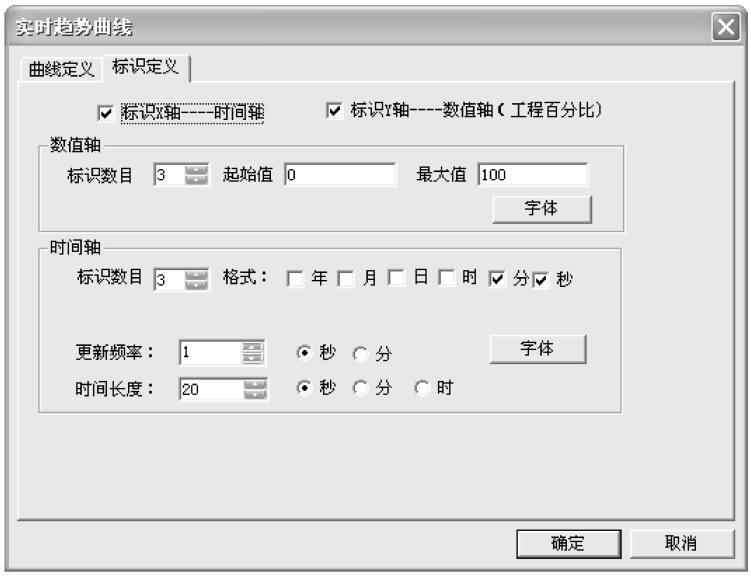


图 7-18 “标识定义”属性页

6）字体：规定数值轴标识所用的字体。可以弹出 Windows 标准的字体选择对话框，相应的操作可参阅 Windows 的操作手册。

7）时间轴定义区。

标识数目：时间轴标识的数目，这些标识在数值轴上等间隔，在组态王开发系统中时间是以 YY：mm：dd：hh：mm：ss 的形式表示，在 TouchView 运行系统中，显示实际的时间，在组态王开发系统画面制作程序中的外观和历史趋势曲线不同，在两边是一个标识拆成两半，与历史趋势曲线区别。

格式：时间轴标识的格式，选择显示哪些时间量。

更新频率：TouchView 是自动重绘一次实时趋势曲线的时间间隔。与历史趋势曲线不

同，它不需要指定起始值，因为其时间始终在当前时间到时间长度之间。

时间长度：时间轴所表示的时间范围。

字体：规定时间轴标识所用的字体，与数值轴的字体选择方法相同。

单击“确定”按钮，关闭此对话框，即完成实时趋势曲线的参数设置。选择菜单“文件｜ 全部存”，保存新的工作成果。如果需要 Y 轴标识实际工程值，可以在标识定义内不选择标识 Y 轴，然后在曲线 Y 轴旁边用“文本”按比例标识实际工程值。

实时趋势曲线建立时，其默认的填充色为“黑色”，所以 X 轴、Y 轴的标识都无法显示出来。这时可以先选中“实时趋势曲线”，再单击工具箱中的“显示调色板”按钮，或选择菜单“工具｜显示调色板”项，弹出“调色板”。在“调色板”顶部的“对象选择”按钮中，选择“填充色”按钮，再在“调色板”中选中其他非黑色颜色即可。

另外，在“实时趋势曲线”画面中新建一个按钮，将按钮文本替换为“切换到监控画面”，双击该按钮，在弹出的“动画连接”对话框中选择“弹起时”项。在弹出的“命令语言”编辑框中输入"ShowPicture（监控中心）”，以便系统在运行时，能从“实时趋势曲线”画面切换回“监控画面”。

3. 为实时趋势曲线建立“笔”

首先画出笔的形状（一般用多边形即可），然后加上垂直移动动画连接，可以通过打开光标位置选择找出实时趋势曲线绘图区域右边的两个顶点，然后用对应的实时曲线定义所用的表达式定义一个移动连接，如图 7－19 所示。“实时趋势曲线”画面的运行效果如图 7－20 所示。

图 7－19　“笔”的垂直移动动画连接

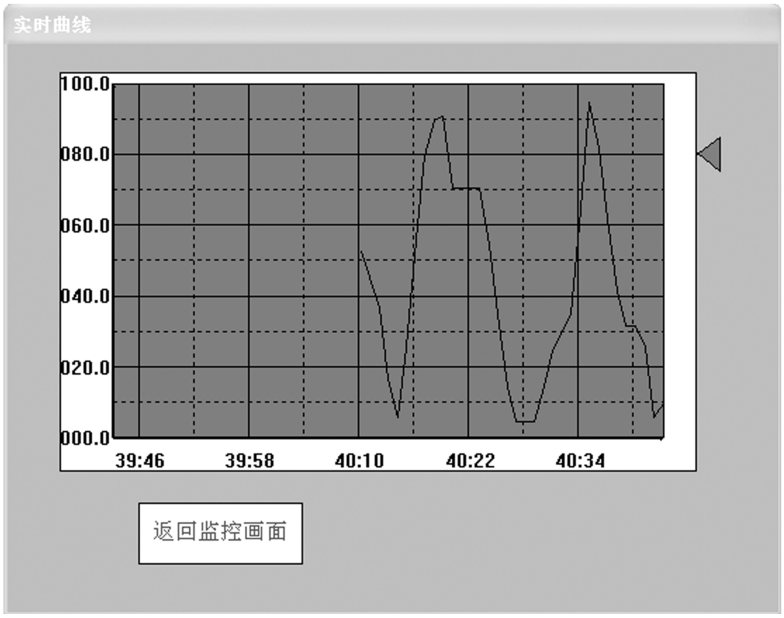


图 7－20　“实时趋势曲线”画面运行效果图

7.2.2 历史趋势曲线

组态王的历史趋势曲线以 Active X 控件形式提供的取组态王数据库中的数据绘制历史曲线和取 ODBC 数据库中的数据绘制曲线的工具。通过该控件，不但可以实现历史曲线的绘制，还可以实现 ODBC 数据库中数据记录的曲线绘制，而且在运行状态下，可以实现在线动态增加/删除/隐藏曲线、曲线图表的无级缩放、曲线的动态比较、曲线的打印等。该曲线控件最多可以绘制 16 条曲线。

组态王提供三种形式的历史趋势曲线：

1）是从图库中调用已经定义好的各功能按钮的历史趋势曲线，对于这种历史趋势曲线，用户只需要定义几个相关变量，适当调整曲线外观即可完成历史趋势曲线的复杂功能，这种形式使用简单方便；该曲线控件最多可以绘制 8 条曲线，但该曲线无法实现曲线打印功能。

2）是从工具箱中调用历史趋势曲线，对于这种历史趋势曲线，用户需要对曲线的各个操作按钮进行定义，即建立命令语言连接才能操作历史曲线，对于这种形式，用户使用时自主性较强，能做出个性化的历史趋势曲线；该曲线控件最多可以绘制 8 条曲线，该曲线无法实现曲线打印功能。

3）是调用 KVHTrend 历史趋势曲线控件，对于这种历史趋势曲线，功能很强大，使用比较简单。本书将主要介绍第三种历史趋势曲线控件。

无论使用哪一种历史趋势曲线，都要进行相关配置，主要包括变量属性配置和历史数据文件存放位置配置。

1. 与历史趋势曲线有关的必备配置项

（1）定义变量范围　由于历史趋势曲线数值轴显示的数据是以百分比来显示，因此对于要以曲线形式来显示的变量，需要特别注意变量的范围。如果变量定义的范围很大，例如 $-999999 \sim +999999$，而实际变化范围很小，例如 $-0.0001 \sim +0.0001$，这样，曲线数据的百分比数值就会很小，在曲线图表上就会出现看不到该变量曲线的情况。

（2）对某变量作历史记录　对于要以历史趋势曲线的形式显示的变量，都需要对变量做记录。在组态王中，离散型、整型和实型变量支持历史记录，字符串型变量不支持历史记录。组态王的历史记录形式可以分为数据变化记录、定时记录（最小单位为 1 min）和备份记录。记录形式的定义通过变量属性对话框中提供的选项完成。在组态王工程浏览器中单击“数据库”项，再选择“数据词典”选项，选中要做历史记录的变量，双击该变量进入“定义变量”对话框，选择“记录和安全区”属性页，如图 7-21 所示。

各选项的含义如下：

1）记录：此选项有效时，则该变量值不进行历史记录。

2）数据变化记录：系统运行时，变量的值发生变化，而且当前变量值与上次的值之间的差值大于设置的变化灵敏度时，该变量的值才会被记录到历史记录中。这种记录方式适合于数据变化较快的场合。

3）变化灵敏度：定义变量变化记录时的阈值。当“数据变化记录”选项有效时，“变化灵敏度”选项才有效。

例：数据库中有一个实型变量，如果需要对该变量的值进行记录，而且规定其变化灵敏度为 1，则其记录过程为如果第一次记录值是 10，当第二次的变量值为 10.9 时，由于 $10.9-10=0.9<1$，也就是第二次变量值相对第一次记录值的变化小于设定的“变化灵敏度”，所以

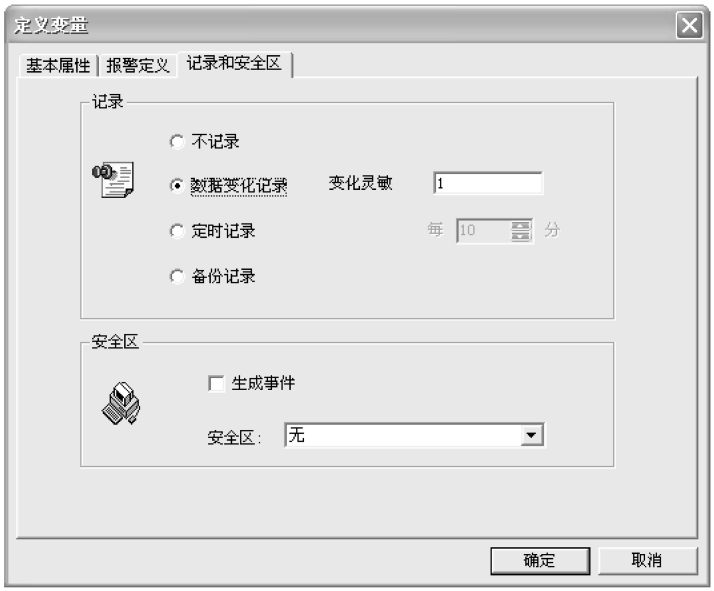


图 7-21　“记录和安全区”属性页

第二次变量值不进行记录，当第三次变量值为 12 时，由于 12 - 10.9 = 1.1 > 1，即变化幅度大于设定的“变化灵敏度”，所以此次变量值才被记录到历史记录中。

4）定时记录：无论变量变化与否，系统运行时按定义的时间间隔将变量的值记录到历史库中，每隔设定的时间对变量的值进行一次记录。最小定义时间间隔单位为 1 min，这种方式适用于数据变化缓慢的场合。

5）备份记录：选择该项，系统在平常运行时，不再直接向历史库中记录该变量的数值，而是通过其他程序调用组态王历史数据库接口，向组态王的历史记录文件中插入数据。在进行历史记录查询时，可以查询到这些插入的数据。这种方式一般用于环境复杂的、无人职守的数据采集点等场合。在这些场合使用的有些设备带有一定数量的数据存储器，可以存储一段时间内设备采集到的数据。但这些设备往往只是简单的记录数据，而不能进行历史数据的查询、浏览等操作，而且必须通过上位机的处理才可以看到。

2. 创建历史趋势曲线控件

KVHTrend 曲线控件是组态王以 Active X 控件形式提供的绘制历史曲线和 ODBC 数据库曲线的功能性工具。该曲线具有以下特点：

1）既可以连接组态王的历史数据库，也可以通过 ODBC 数据源连接到其他数据库上，如 Access、SQLServer 等。

2）连接组态王历史数据库时，可以定义查询数据的时间间隔，如同在组态王中使用报表查询历史数据时使用查询间隔一样。

3）完全兼容了组态王原有历史曲线的功能。最多可同时绘制 16 条曲线。

4）可以在系统运行时动态增加、删除、隐藏曲线，还可以修改曲线属性。

5）曲线图表实现无极限缩放。

6）可实现某条曲线在某个时间段上的曲线比较。

7）无效数据不显示。

8）数值轴可以使用工程百分比标识，也可以用曲线实际范围标识，两者之间自由切换。

9）曲线支持毫秒级数据。

10）可直接打印图表曲线。

11）通过 ODBC 数据源连接数据库时，可以自由选择数据库中记录时间的时区，根据选择的时区来绘制曲线。

12）可以自由选择曲线列表框中的显示内容。

在组态王画面制作系统的“历史趋势曲线”画面中，在工具箱中单击“插入通用控件”或选择菜单“编辑”下的“插入通用控件”命令，弹出“插入控件”对话框，在列表中选择“历史趋势曲线”，单击“确定”按钮，对话框自动消失，鼠标箭头变为小“+”字形，在画面上选择控件的左上角，按下鼠标左键并拖动，画面上显示出一个虚线的矩形框，该矩形框为创建后的曲线的外框。当达到所需大小时，松开鼠标左键，则历史曲线控件创建成功，画面上显示出该曲线，如图 7-22 所示。

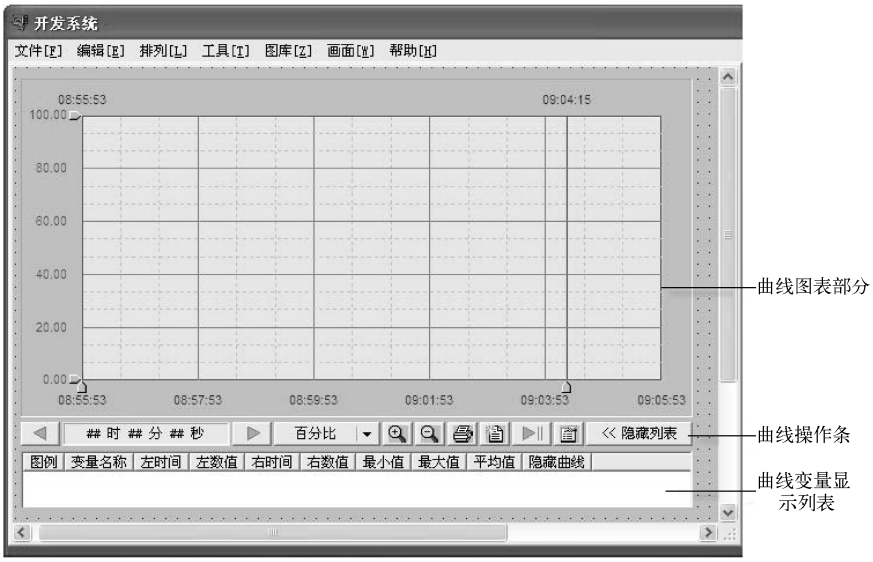


图 7-22　历史趋势曲线控件

3. 设置历史曲线固有属性

历史曲线控件创建完成后，在控件上单击右键，在弹出的快捷菜单中选择“控件属性”命令，弹出历史曲线控件的固有属性对话框。该控件固有属性含有两个属性页：曲线、坐标系，如图 7-23 所示。

1）“曲线”属性页。“曲线”属性页中下半部分的“数据源”是说明定义在绘制曲线时，历史数据的来源。可以选择组态王的历史数据库或其他 ODBC 数据库为数据源。

“曲线”属性页中上半部分的“曲线”列表是定义曲线图表初始状态的变量名称、绘制方式、比较曲线等。各项的含义如下：

① 采集间隔：确定从数据库中读出数据点的时间间隔，可以精确到毫秒。“秒”和“毫秒”不能同时为零，即最小单位为 1 ms。该项的选择将影响曲线绘制的质量和系统的效率。当选择的时间单位越小时，绘制的数据点越多，曲线的逼真度越高，系统效率会有所降

图 7-23　“控件属性”对话框

低。相反，如果选择的时间单位越大，绘制的数据点越少，曲线的逼真度相对降低，移动曲线时，有时会出现在同一个时间点上的曲线显示不同的情况，但系统效率受影响较小。

② 增加：增加变量到曲线图表，并定义曲线绘制方式。单击该按钮，弹出如图 7-24 所示的对话框。各部分的含义如下：

图 7-24　“增加曲线”对话框

变量名称：在“变量名称”文本框中输入要添加的变量的名称，或在左侧的列表框中选择，该列表框中列出了本工程中所有定义了历史记录属性的变量，如果在定义变量属性时没有定义进行历史记录，则此处不会列出该变量。单击鼠标，则选中的变量名称自动添加到“变量名称”文本框中，一次只能添加一个变量，且必须通过单击该画面的“确定”按钮来完成这一条曲线的添加。

曲线定义包括：

线类型：单击“线类型”后的下拉列表框，选择当前曲线的类型。

线颜色：单击“线颜色”后的按钮，在弹出的调色板中选择当前曲线的颜色。

绘制方式：曲线的绘制方式有 4 种。模拟、阶梯、逻辑、棒图，可以任选一种。

隐藏曲线：控制运行时是否显示该曲线。在运行时，也可以通过曲线窗口下方的列表中的属性选择来控制显示或隐藏该曲线。

曲线比较：通过设置曲线显示的两个不同时间，使曲线绘制位置有一个时间轴上的平移，这样，在一个变量名代表的两条曲线中，一个是显示与时间轴相同的时间的数据，另一个做比较的曲线显示有时间差的数据（如一天前），从而达到用两条曲线来实现曲线比较的目的。

数据来源：选择曲线使用的数据来源，可同时支持组态王历史库和 ODBC 数据源。

所在时区：选择数据库记录时间所在的时区。为了统一时间基准，组态王在读取数据库的时间时是按照标准时间——格林威治时间来读取的，所以需要用户定义自己所在的时区。如果时区选择错误的话，可能在图表的相应时间段上找不到曲线。

注意：在使用 ODBC 数据源时，同样需要在变量列表中选择一个变量，用它来确定曲线变量的值的范围。

选择完变量并配置完成后，单击“确定”按钮，则曲线名称添加到“曲线列表”中。如上所述，可以增加多个变量到曲线列表中。选择已添加的曲线，则“删除”、“修改”按钮变为有效。

③ 删除：删除当前列表框中选中的曲线。

④ 修改：修改当前列表框中选中的曲线。

⑤ 显示列表：选中该项，在运行时，曲线窗口下方可以显示所有曲线的基本情况列表。在运行时也可以通过按钮控制是否要显示该列表。列表中的内容可以自定义，但“图例”一项不可删除。单击“列表项”按钮，弹出“列表项选择”对话框，左边列表框中为选出的不用显示的项，右边列表框中为需要显示的内容。选择列表框中的项目，单击“添加”或“删除”按钮，确定要显示的项。单击“上移”、“下移”按钮，排列所选择项的排列顺序。需要注意的是，“图例”一项的位置不可修改。

2）“坐标系”属性页。在如图 7-23 所示的“控件属性”对话框中，单击“坐标系”标签，进入“坐标系”属性设置页，如图 7-25 所示。

① 边框颜色和背景颜色：设置曲线图表的边框颜色和图表背景颜色。单击相应按钮，弹出浮动调色板，选择所需颜色。

② 绘制坐标轴选项：是否在图表上绘制坐标轴。单击“轴线类型”列表框选择坐标轴线的线型；单击“轴线颜色”按钮，选择坐标轴线的颜色。绘制出的坐标轴带箭头的表示 X、Y 方向的直线。

图 7-25　“坐标系”属性页

③ 分割线：定义时间轴、数值轴主次分割线的数目、线的类型、线的颜色等。如果选择了分割线为短线，则定义的主分割线变为坐标轴上的短线，曲线图表不再是被分割线分割的网状结构，如图 7-26 所示。此时，此分割线不再起作用，其选项也变为灰色无效。

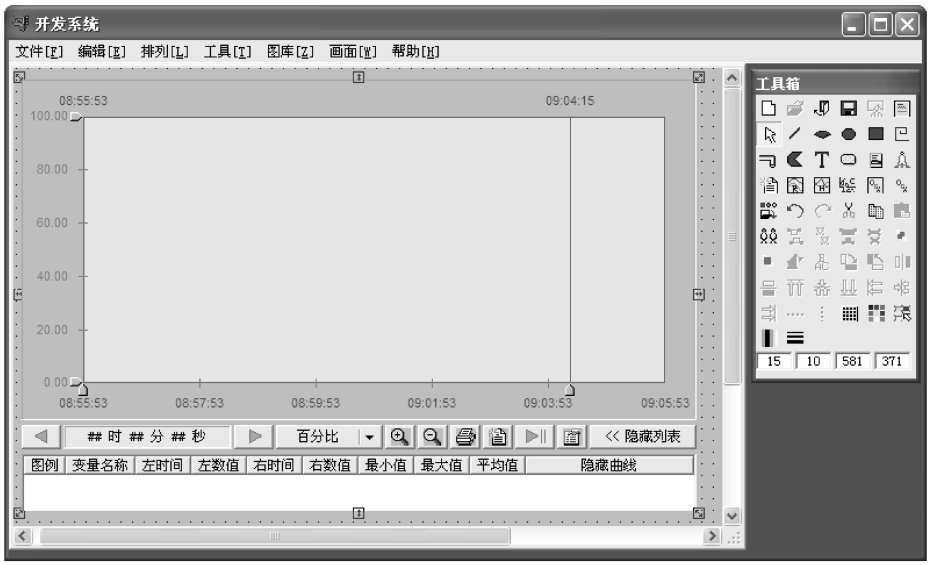


图 7-26　分割线设置为短线后的效果图

④ 标记数值（Y）轴：“标记数目”编辑框中定义数值轴上的标记的个数，“最小值”、“最大值”编辑框定义初始显示的值的百分比范围（0 ~ 100%）。单击“字体”按钮，弹出

字体、字型、字号选择对话框，选择数值轴标记的字体及颜色等。

⑤ 标记时间（X）轴：“标记数目”编辑框中定义时间轴上的标记的个数。通过选择“格式”选项，选择时间轴显示的时间格式。“时间长度”编辑框定义初始显示时图表所显示的时间段的长度。单击“字体”按钮，弹出字体、字型、字号选择对话框，选择数值抽标记的字体及颜色等。

所有项定义完成后，单击“确定”按钮返回。

4. 设置历史曲线控件的动画连接属性

以上所设置的是历史趋势曲线控件的固有属性，在使用该历史曲线时必定要使用到这些属性。由于该历史曲线以控件形式出现，因此，该曲线还具有控件的属性。要在组态王中使用该控件，还需定义其“属性”和“事件”。

用鼠标选中并双击该控件，弹出“动画连接属性”设置对话框，动画连接属性共有 3 个属性页：“常规”、“属性”、“事件”。

（1）“常规”属性页

1）控件名：定义该控件在组态王中的标识名，如“历史曲线”，该标识名在组态王当前工程中应该唯一。

2）优先级、安全区：定义控件的安全性。在运行时，当用户满足定义的权限时才能操作该历史曲线。

（2）“属性”属性页　在“动画连接属性”对话框中，单击“属性”标签，进入“属性”设置页，如图 7-27 所示。该属性页定义控件属性与组态王变量相关联的关系。

图 7-27　“属性”属性页

（3）“事件”属性页　在“动画连接属性”对话框中，单击“事件”标签，进入“事件”属性设置页，历史曲线控件提供了很多控件方法，供用户在命令语言中调用。该属性页定义控件的事件函数和历史曲线的详细事件及含义见组态王的联机帮助手册。

5. 运行时修改历史曲线控件属性

历史曲线属性定义完成后，进入组态王运行系统，运行系统的历史曲线如图 7-28 所示。

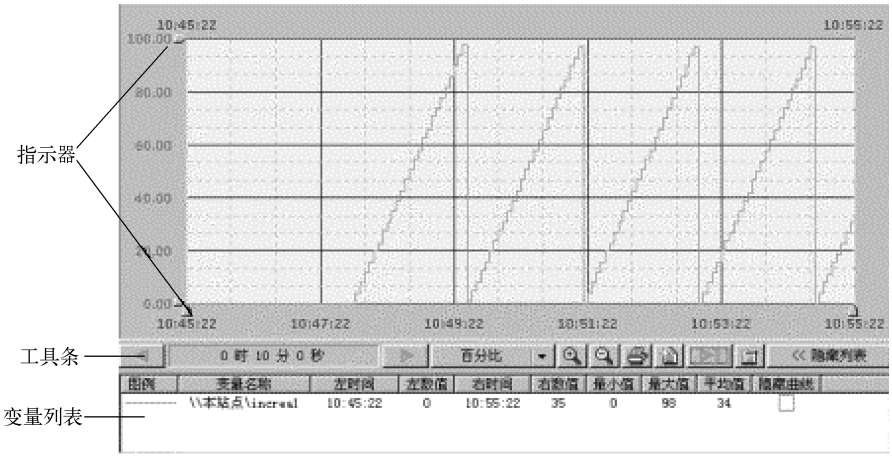


图 7-28　“历史趋势曲线”运行效果

（1）数据轴指示器的使用　数据轴指示器又称数据轴游标，拖动数据轴（Y 轴）指示器，可以放大或缩小曲线在 Y 轴方向的长度，一般情况下，该指示器标记为变量量程的百分比。

（2）时间轴指示器的使用　时间轴指示器又称时间轴游标，拖动时间轴指示器可以获得曲线与时间轴指示器焦点的具体时间，与可以配合 HTGetValueScooter 函数获得曲线与时间轴指示器焦点的数值。

（3）工具条的使用　利用历史趋势曲线窗口中的工具条可以查看变量过去任一段时间的变化趋势以及对曲线进行放大、缩小、打印等操作。工具条如图 7-29 所示。

图 7-29　工具条

1）  时间跨度设置按钮：单击此按钮弹出“输入历史调整跨度”对话框，如图 7-30 所示。

图 7-30　“输入历史调整跨度”对话框

在对话框中输入时间跨度值，如“60 秒”。单击“确定”按钮后关闭此窗口，当点击“◀”或“▶”按钮时，会向前或向右移动一个时间跨度（即 60 秒）。

2）百分比设置 Y 轴标记：设置趋势曲线显示风格：以百分比格式显示或以实际值格式显示。

3）放大所选区域：在曲线显示区中选择一个区域，单击此按钮可以放大当前区域中的曲线。

① 当在曲线显示区中选取了矩形区域时，时间轴最左/右端调整为矩形左/右边界所在的时间，数值轴标记最上/下端调整为矩形上/下边界所在数值，从而使曲线局部放大，左/右指示器位置分别置于时间轴最左/右端。

② 当未选定任何区域，左/右指示器不在时间轴最左/右端时，时间轴最左/右端调整为左/右指示器所在的时间，数值轴不变，从而使曲线局部放大。经放大后左/右指示器位置分别置于时间轴最左/右端。

③ 当未选定任何区域，左/右指示器在时间轴最左/右端时，时间轴宽度调整为原来的一半，保持中心位置不变，数值轴不变，从而使曲线局部放大，经放大后左/右指示器位置分别置于时间轴最左/右端。

4）缩小所选区域：在曲线显示区中选择一个区域，单击此按钮可以缩小当前区域中的曲线。

① 当在曲线显示区中选取了矩形区域时，矩形左/右边界所在的时间调整为时间轴最左/右端所在的时间，矩形上/下边界所在数值调整为数值轴最上/下端所在数值，从而使曲线局部缩小。经缩小后，左/右指示器位置分别置于时间轴最左/右端。

② 当未选定任何区域，左/右指示器不在时间轴最左/右端时，左/右指示器所在的时间调整为时间轴最左/右端所在的时间，数值轴不变，从而使曲线局部缩小。经缩小后，左/右指示器位置分别置于时间轴最左/右端。

③ 当未选定任何区域，左/右指示器在时间轴最左/右端时，时间轴宽度调整为原来的 2 倍，保持中心位置不变，数值轴不变，从而使曲线局部缩小。经缩小后左/右指示器位置分别置于时间轴最左/右端。

5）打印窗口：单击此按钮打印当前曲线窗口。

6）定义新曲线：单击此按钮弹出如图 7-24 所示的“增加曲线”对话框，在对话框中定义新的曲线。

7）将时间轴右端设为当前时间：单击此按钮将历史趋势曲线窗口时间轴右端的时间设置为当前时间。

8）设置参数：单击此按钮弹出参数设置对话框，如图 7-31 所示。

在此对话框中输入历史趋势曲线窗口的起止时间（即您想查询历史曲线的时间）、数据轴的量程范围及游标显示风格等等。

9）<< 隐藏列表 显示/隐藏列表：单击此按钮可显示或隐藏变量列表区。

6. 变量列表区

变量列表区主要用于显示变量的信息包括：变量名称、变量的最大值、最小值、平均值以及动态显示/隐藏指定的曲线等等。

在变量列表区上单击右键弹出下拉菜单，如图 7-32 所示。

图 7-31　参数设置对话框

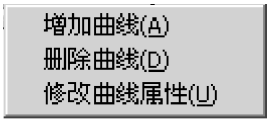


图 7-32　快捷菜单

通过此下拉菜单可对历史曲线窗口中的曲线进行编辑。

练一练：

1）在用户的工程中添加一个实时曲线画面。

2）在用户的工程中添加一个历史曲线画面，熟悉通用历史曲线的的控件的各种使用方法。

3）阅读组态王在线帮助中的历史趋势曲线控件的属性方法。

7.3　控件的使用

控件可以作为一个相对独立的程序单位被其他应用程序重复调用。控件的接口是标准的，凡是满足这些接口条件的控件，包括第三方软件供应商开发的控件，都可以被组态王直接调用。组态王中提供的控件在外观上类似于组合图素，工程人员只需把它放在画面上，然后配置控件的属性进行相应的函数连接，控件就能完成其复杂的功能。

7.3.1　XY 控件

X－Y 轴曲线可用于显示两个变量之间的数据关系，如电流—转速曲线等形式的曲线。

1. 在画面上创建 X－Y 轴曲线

1）新建一画面，名称为 XY 控件画面。

2）选择工具箱中的 T 工具，在画面上输入文字“XY 控件”。

3）单击工具箱中的按钮或选择菜单命令“编辑\插入控件”，则弹出“创建控件”对话框，在“创建控件”对话框内选择 X－Y 轴曲线控件。

4）用鼠标左键单击“创建”按钮，鼠标变成十字形。然后在画面上画一个矩形框，X－Y 轴曲线控件就放到画面上了。可以任意移动、缩放 X－Y 轴曲线控件，如同处理一个单元一样。

在画面上放置的 X－Y 轴曲线控件如图 7-33 所示。

在此控件中 X 轴和 Y 轴变量由工程人员任意设定，因此，X－Y 轴曲线能用曲线方式反应任意两个变量之间的函数关系。

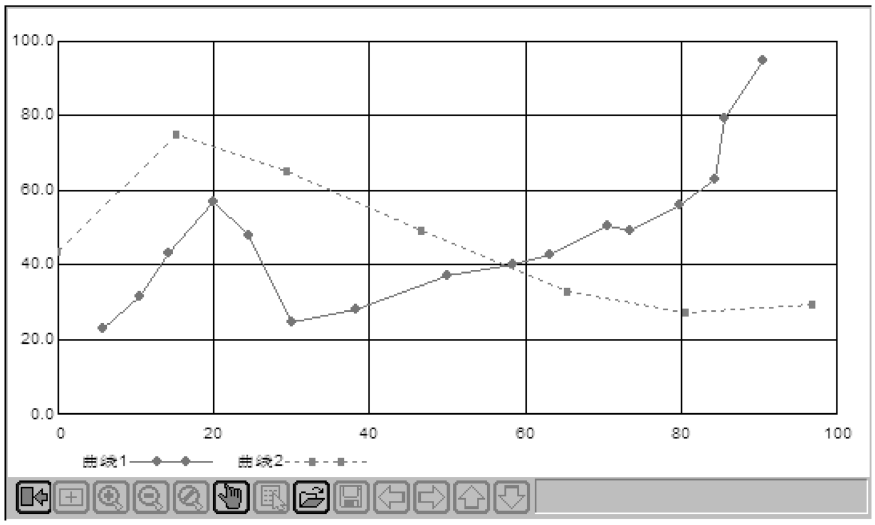


图 7-33　X－Y 轴曲线控件窗口

2. X－Y 轴曲线属性设置

用鼠标双击 X－Y 轴曲线控件，则弹出“X－Y 轴曲线设置”对话框，如图 7-34 所示。

图 7-34　XY 控件属性设置对话框

此对话框中各项的含义分别介绍如下：

背景颜色：此按钮用于设置 X－Y 轴曲线的背景显示颜色。

前景颜色：此按钮用于设置 X－Y 轴曲线横轴和纵轴刻度值的显示颜色。

X 轴（Y 轴）最大值：设定 X 轴（Y 轴）方向的最大刻度值。

X 轴（Y 轴）最小值：设定 X 轴（Y 轴）方向的最小刻度值。

X 轴（Y 轴）分度数：用于指定 X 轴或 Y 轴的最大坐标值和最小坐标值之间等间距的

刻度数。比如，如果 Y 轴的最大坐标值为 90，最小坐标值为 10，设定 Y 轴刻度数为 20，则最小坐标值和最大坐标值之间有 20 等份，每一个等份值就是一个刻度值。

X 轴（Y 轴）小数位数：用于设置 X 轴（或 Y 轴）刻度值的小数位数。

曲线最大点数：用于规定曲线上最多显示点数。

显示操作条：此选项用于显示/隐藏操作条，当此选项有效时，此选项前面有一个符号“√”，“初始状态”单选框由灰色变为正常色。

初始状态：当“显示操作条”有效时，初始状态单选框由灰色变为正常色。此选项决定操作条显示时是按最大化还是最小化方式显示，选中某一种初始化状态，此选项前面有一个“●”的标记。如选中“最大化”，此时 X－Y 轴曲线如图 7–35 所示。

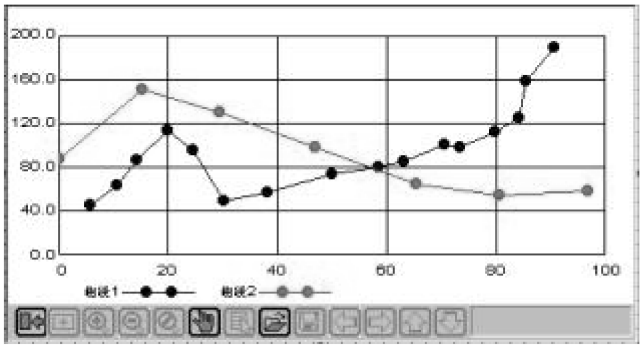


图 7–35　最大化时 X－Y 轴曲线

注意：当操作条按最大化显示时，若要显示出完整的操作条，X－Y 轴曲线在界面上放置需足够大。

操作条如图 7–36 所示。

图 7–36　X－Y 轴曲线控件操作条

各按钮的具体含义解释如下：

1）隐藏或显示操作平台。

2）曲线选定区域放大。

3）曲线放大两倍。

4）曲线缩小两倍。

5）曲线缩放至原状。

6）调整坐标系。单击，弹出如图 7–37 所示对话框，可在此修改坐标。

7）修改曲线属性。单击弹出对话框如图 7–38 所示，可修改曲线颜色、标记和线型。

8）从文件中装载 X－Y 轴曲线。

9）保存 X－Y 轴曲线。

10）曲线右移。

11）曲线左移。

图 7-37　调整坐标系对话框

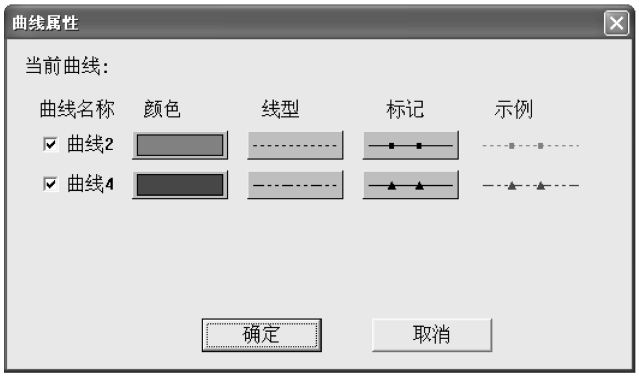


图 7-38　修改曲线属性对话框

12）曲线上移。

13）曲线下移。

不要操作条背景色：此选项用于控制操作条的背景色，当此选项有效时，此选项前面有一个符号“√”。此时操作条的背景色与界面背景色一样；当此选项无效时，此时操作条的背景色与温控曲线的背景色一样。

添加网格线：此选项用于显示/关闭网格线，当此选项有效时，此选项前面有一个符号“√”。

字体：此按钮用于设置标签字符的字体大小、字体样式。单击“标签字体”按钮，则弹出“字体”对话框。

曲线设置：此按钮用于设置曲线的颜色、线型和标记。

显示图例：此选项用于显示/隐藏图例，当此选项有效时，此选项前面有一个符号“√”。“初始状态”单选框由灰色变为正常色。

显示边框：此选项用于显示/隐藏 X－Y 轴曲线的边框，当此选项有效时，此选项前面有一个符号“√”。

3. 添加命令语言

为使 XY 曲线控件实时反应变量值，需要为该控件添加命令语言。

利用函数 XYAddNewPoint 可以在指定的 X－Y 轴曲线控件中增加一个数据点。如果需要在画面中 一直绘制采集的数据，可以在“命令语言”的“存在中”写入如下语句：

XYAddNewPoint（"XY 曲线"，水温，热水阀，1）；

或者是 XYAddNewPoint（"XY 曲线"，30，20，1）；

后面这个语句表示在 XY 曲线中索引号为 1 的曲线上添加一个点，该点的坐标值为（30，20）。绘点的速度可以通过改变“存在中”的执行周期来调整。X－Y 轴曲线最多可以支持 8 条。其他在运行中控制 X－Y 轴曲线的主要功能还包括删除曲线。

在画面命令语言中输入如图 7-39 所示脚本语言。

图 7-39　画面命令语言

4. 保存

单击“文件”菜单中的“全部存”命令，保存您所作的设置。

5. 进入运行系统

单击“文件”菜单中的“切换到 View”命令，进入运行系统。运行此画面，如图 7-40 所示。

7.3.2　Active X 控件

组态王除了支持本身提供的各种控件，还支持 Windows 标准的 Active X 控件，包括 Microsoft 提供的标准 Active X 控件和用户自制的 Active X 控件。Active X 控件的引入在很大程度上方便了用户，用户可以灵活地编制一个符合自身需要的控件或调用一个已有的标准控件来完成一项复杂的任务，而无须在组态王中做大量的复杂的工作。一般的 Active X 控件都具有属性、方法、事件，用户通过设置控件的这些属性、事件、方法来完成工作。

日历控件（Microsoft Date and Time Picker Control 6.0）是一个典型的 Active X 控件，利

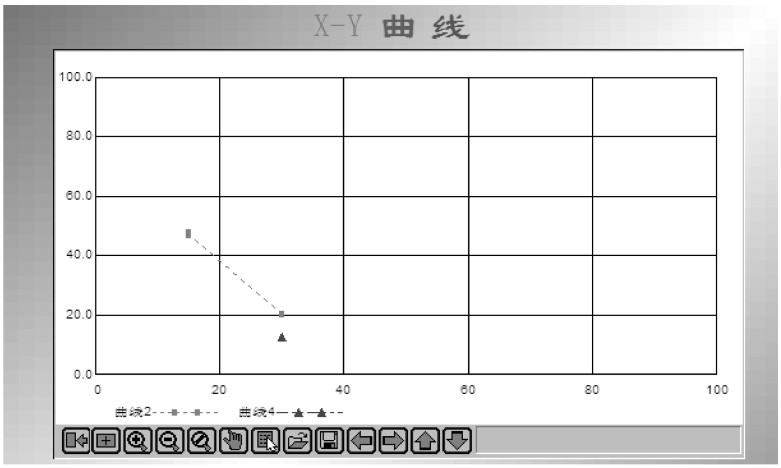


图 7-40　运行中的 XY 控件

用日历控件可实现在组态王中设置任一时间的功能，操作过程如下：

1）在工程浏览器窗口的数据词典中定义三个内存实型变量：

① 变量名：年变量；变量类型：内存实型；最小值：0；最大值：10000 。

② 变量名：月变量；变量类型：内存实型；最小值：0；最大值：12。

③ 变量名：日变量；变量类型：内存实型；最小值：0；最大值：31。

2）新建一画面，名称为：日历控件画面。

3）单击工具箱中的工具，在弹出的通用控件窗口中选择如下控件，如图 7-41 所示。

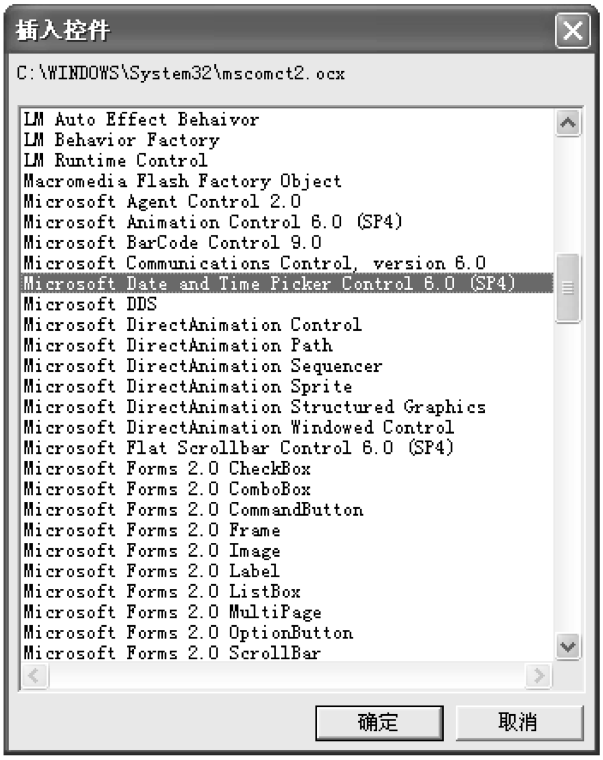


图 7-41　通用控件对话框

单击“确定”按钮，在画面中绘制一日历控件，如图 7-42 所示。

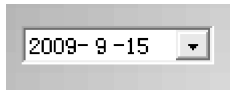


图 7-42　日历控件

4）双击此控件弹出动画连接属性对话框，如图 7-43 所示。

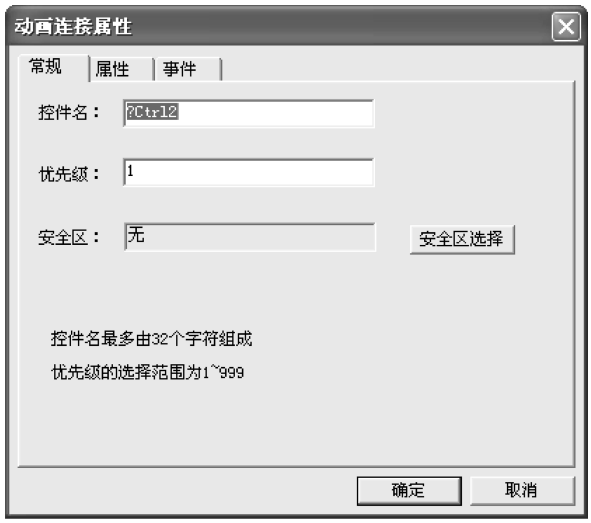


图 7-43　控件动画连接属性对话框

控件属性设置如下：

控件名：ADate。

双击“事件”属性卡中的“CloseUp”事件，在弹出的事件命令语言对话框中输入命令语言，如图 7-44 所示。

图 7-44　事件命令语言对话框

5）关闭对话框，在画面中添加三个文本框，在文本框的“模拟量值输出”动画中分别连接变量\\本站点\年变量、\\本站点\月变量、\\本站点\ 日变量，分别显示在日历控件中选择日期的年、月、日。

6）单击“文件”菜单中的“全部存”命令，保存您所作的设置。

7）单击“文件”菜单中的“切换到 View”命令，进入运行系统。运行此画面，如图 7-45 所示。

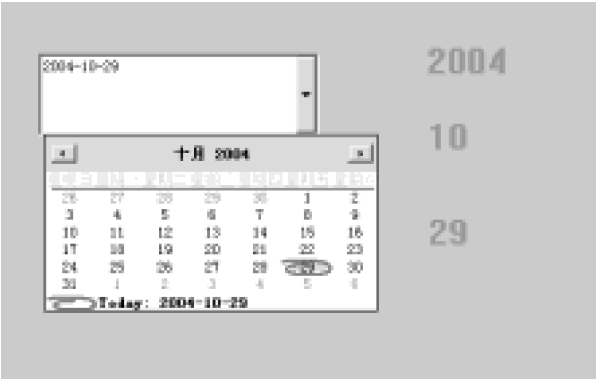


图 7-45　运行中的日历控件画面

单击控件中的下拉按钮，在下拉框中选择设定的日期后，日期的年、月、日分别显示在变量\\本站点\\年变量、\\本站点\\月变量、\\本站点\\日变量所连接的文本框中。

练一练：

1）掌握温控曲线控件的使用。

2）掌握日历控件的使用方法。

7.4　报表的生成

数据报表是反应生产过程中的过程数据、运行状态等，并对数据进行记录、统计的一种重要工具，是生产过程必不可少的一个重要环节。它既能反应系统实时的生产情况又能对长期的生产过程数据进行统计、分析，使管理人员能够掌握和分析生产过程情况。

组态王提供内嵌式报表系统，工程人员可以任意设置报表格式，对报表进行组态。组态王为工程人员提供了丰富的报表函数，实现各种运算、数据转换、统计分析、报表打印等。既可以制作实时报表又可以制作历史报表。另外，工程人员还可以制作各种报表模板，实现多次使用，以免重复工作。

7.4.1　实时数据报表

1. 创建实时数据报表

实时数据报表创建过程如下：

1）新建一画面，名称为：实时数据报表画面。

2）选择工具箱中的 T 工具，在画面上输入文字“实时数据报表”。

3）选择工具箱中的 工具，在画面上绘制一实时数据报表窗口，如图 7-46 所示。

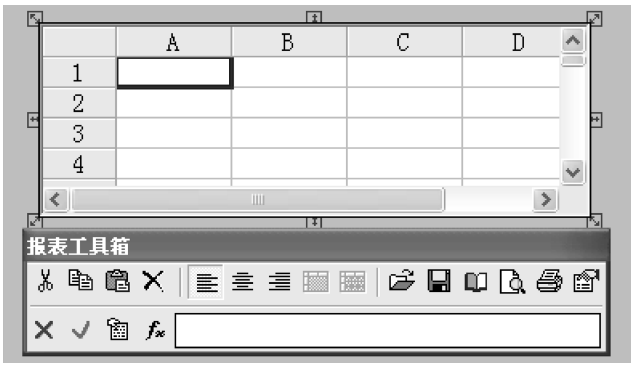


图 7-46　实时数据报表窗口

“报表工具箱”会自动显示出来，双击窗口的灰色部分，弹出“报表设计”对话框，如图 7－47 所示。

对话框设置如下：报表控件名：Report1；行数：6；列数：10。

4）输入静态文字：选中 A1 到 J1 的单元格区域，执行“报表工具箱”中的“合并单元格”命令并在合并完成的单元格中输入“实时数据报表演示”。

利用同样方法输入其他静态文字，如图 7-48 所示。

图 7-47　“报表设计”对话框

图 7-48　静态文字输入

5）插入动态变量：合并 B2 和 C2 单元格，并在合并完成的单元格中输入：＝\\本站点\$日期（变量的输入可以利用“报表工具箱”中的“插入变量”按钮实现）。利用同样方法输入其他动态变量，如图 7-49 所示。

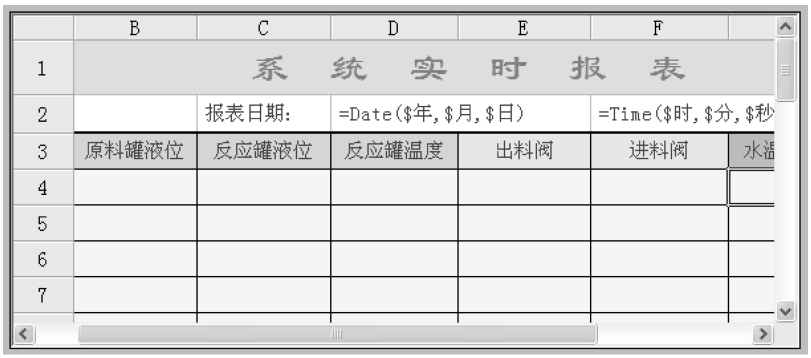


图 7-49　动态变量输入

注：如果变量名前没有添加“ = ”符号的话，此变量被当作静态文字来处理。

6）单击“文件”菜单中的“全部存”命令，保存您所作的设置。

7）单击“文件”菜单中的“切换到 View”命令，进入运行系统。系统默认运行的画面可能不是您刚刚编辑完成的“实时数据报表画面”，您可以通过运行界面中“画面”菜单中的“打开”命令将其打开后方可运行，如图 7-50 所示。

图 7-50　实时报表运行画面

2. 实时数据报表打印

实时数据报表打印设置过程如下：

1）在“实时数据报表画面”中添加一按钮，按钮文本为“实时数据报表打印”。

2）在按钮的弹起事件中输入如下命令语言，如图 7-51 所示。

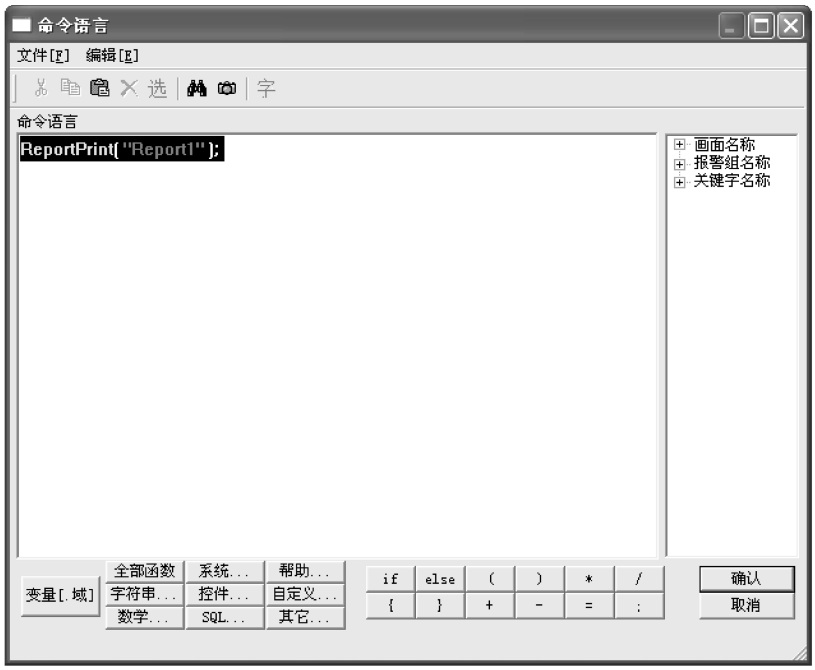


图 7-51　实时数据报表打印命令语言

3）单击“确认”按钮关闭命令语言编辑框。

4）当系统处于运行状态时，单击此按钮，弹出打印属性设置对话框，如图 7-52 所示。

5）在“打印属性”对话框中做相应设置后，单击“确定”按钮，数据报表将被打印出来。

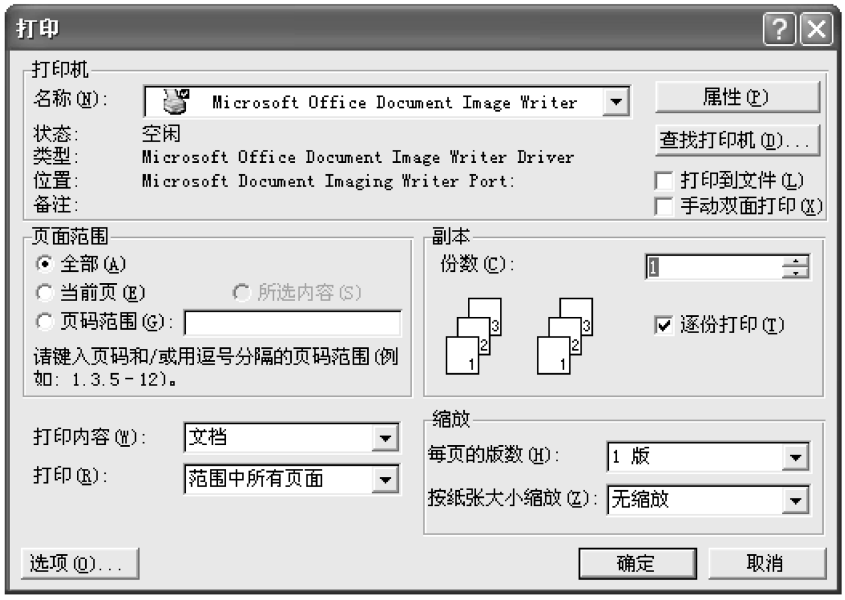


图 7-52　打印属性设置对话框

3. 实时数据报表页面设置过程

1）在“实时数据报表画面”中添加一按钮，按钮文本为“实时数据报表页面设置”。

2）在按钮的弹起事件中输入如下命令语言，如图 7-53 所示。

图 7-53　页面设置命令语言输入

3）单击“确认”按钮关闭命令语言编辑框。

4）当系统处于运行状态时，单击此按钮，弹出页面设置对话框，如图 7-54 所示。

图 7-54　页面设置对话框

5）在“页面设置”对话框中对报表的页面属性做相应设置后，单击“确定”按钮，完成报表的页面设置。

4. 实时数据报表打印预览设置过程

1）在“实时数据报表画面”中添加一按钮，按钮文本为“实时数据报表打印预览”。

2）在按钮的弹起事件中输入如下命令语言，如图 7-55 所示。

图 7-55　打印预览命令语言输入

3）单击“确认”按钮，关闭命令语言编辑框。

4）当系统处于运行状态时，页面设置完毕后，单击此按钮，系统会自动隐藏组态王的开发系统和运行系统窗口，并进入打印预览窗口，如图 7-56 所示。

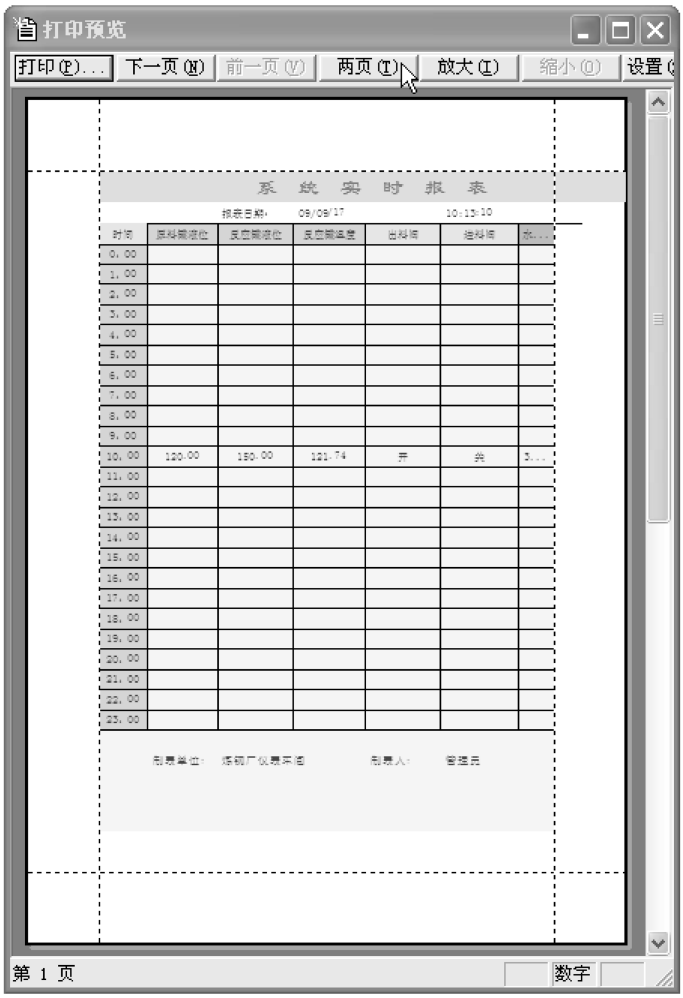


图 7-56　打印预览窗口

5）在打印预览窗口中使用打印预览查看打印后的效果，单击“关闭”按钮结束预览，系统自动恢复组态王的开发系统和运行系统窗口。

5. 实时数据报表的存储

实现以当前时间作为文件名将实时数据报表保存到指定文件夹下的操作过程如下：

1）在当前工程路径下建立一文件夹“实时数据文件夹”。

2）在“实时数据报表画面”中添加一按钮，按钮文本为“保存实时数据报表”。

3）在按钮的弹起事件中输入如下命令语言，如图 7-57 所示。

命令语言如下：

```
string filename;
filename = InfoAppDir( ) + "\实时数据文件夹\" + StrFromReal( \\本站点\ $ 年,0,"f" ) + Str-
```

FromReal(\\本站点\$月,0,"f") + StrFromReal(\\本站点\$日,0,"f") + StrFromReal(\\本站点\$时,0,"f") + StrFromReal(\\本站点\$分,0,"f") + StrFromReal(\\本站点\$秒,0,"f") + ".rtl";

ReportSaveAs("Report1",filename);

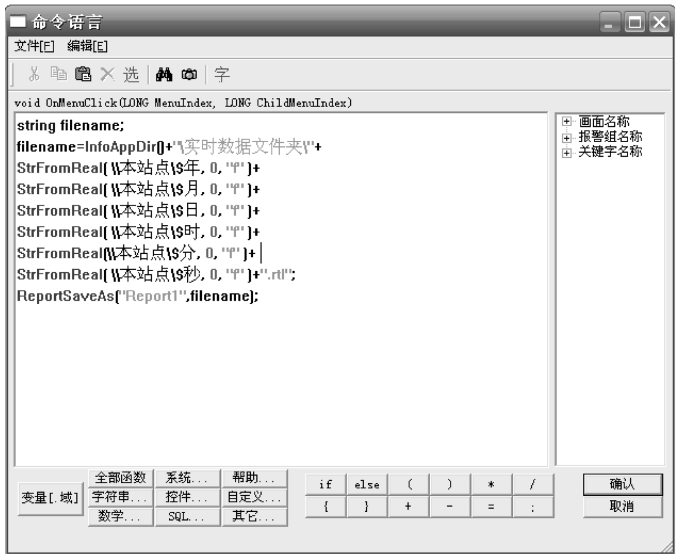


图 7-57　报表存储命令语言输入

4）单击“确认”按钮关闭命令语言编辑框。当系统处于运行状态时，单击此按钮数据报表将以当前时间作为文件名保存实时数据报表。

6. 实时数据报表的查询

利用系统提供的命令语言可将实时数据报表以当前时间作为文件名保存在指定的文件夹中，对于已经保存到文件夹中的报表文件如何在组态王中进行查询呢？下面将介绍一下实时数据报表的查询过程：

利用组态王提供的下拉式组合框与一报表窗口控件可以实现上述功能。

1）在工程浏览器窗口的数据词典中定义一个内存字符串变量：变量名为报表查询变量；变量类型为内存字符串；初始值为空。

2）新建一画面，名称为实时数据报表查询画面。

3）选择工具箱中的T工具，在画面上输入文字“实时数据报表查询”。

4）选择工具箱中的工具，在画面上绘制一实时数据报表窗口，控件名称为 Report2。

5）选择工具箱中的工具，在画面上插入一“下拉式组合框”控件，控件属性设置如图 7-58 所示。

6）在画面中单击鼠标右键，在画面属性的命令语言中输入如下命令语言，如图 7-59 所示。

图 7-58　属性设置

图 7-59　数据查询命令语言输入

命令语言如下：

string filename；

filename = InfoAppDir( ) + " \实时数据文件夹\ * . rtl" ; listClear(" List1") ;

ListLoadFileName(" List1" ,filename) ;

上述命令语言的作用是将已经保存到“当前组态王工程路径下实时数据文件夹”中的实时报表文件名称在下拉式组合框中显示出来。

7）在画面中添加一按钮，按钮文本为“实时数据报表查询”。

8）在按钮的弹起事件中输入如下命令语言，如图 7-60 所示。

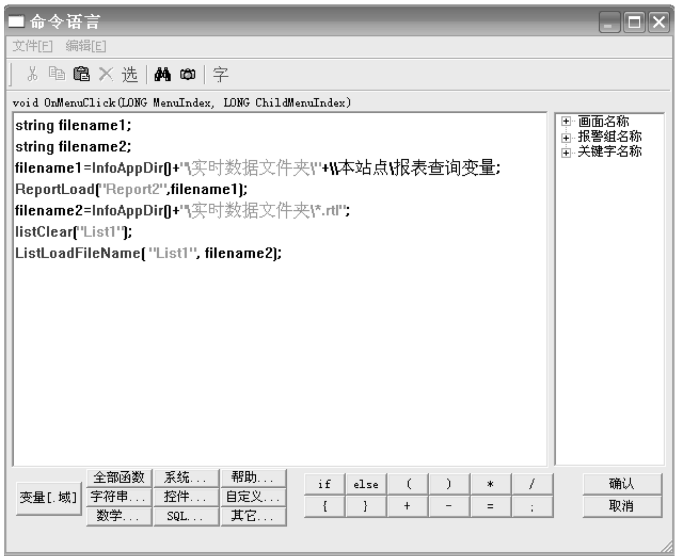


图 7-60　命令语言

命令语言如下所示：

```
string filename1；string filename2；
filename1 = InfoAppDir( ) + " \实时数据文件夹\" + \\本站点\报表查询变量；
ReportLoad( " Report2" ,filename1)；filename2 = InfoAppDir( ) + " \实时数据文件夹\ *. rtl" ;
listClear( " List1" )；
ListLoadFileName( " List1" , filename2)；
```

上述命令语言的作用是将下拉式组合框中选中的报表文件的数据显示在 Report2 报表窗口中，其中\\本站点\报表查询变量保存了下拉式组合框中选中的报表文件名。

9）设置完毕后，单击“文件”菜单中的“全部存”命令，保存您所作的设置。

10）单击“文件”菜单中的“切换到 View”命令，运行此画面。当您单击下拉式组合框控件时，保存在指定路径下的报表文件全部显示出来，选择任一报表文件名，单击“实时数据报表查询”按钮后，此报表文件中的数据会在报表窗口中显示出来，从而达到了实时数据报表查询的目的。

7.4.2 历史数据报表

1. *历史数据报表的两种制作方法*

历史报表记录了以往的生产记录数据，对用户来说是非常重要的。历史报表的制作根据所需数据的不同有不同的制作方法，这里介绍两种常用的方法。

（1）向报表单元格中实时添加数据　例如要设计一个锅炉功耗记录表，该报表为 8 h 生成一个（类似于班报），要记录每小时最后一刻的数据作为历史数据，而且该报表在查看时应该实时刷新。

对于这个报表就可以采用向单元格中定时刷新数据的方法实现。报表设计如图 7-56 所示。按照规定的时间，在不同的小时里，将变量的值定时用单元格设置函数，如 ReportSetCellValue（）；设置到不同的单元格中，这时，报表单元格中的数据会自动刷新，而带有函数的单元格也会自动计算结果，当到换班时，保存当前填有数据的报表为报表文件，清除上班填充的数据，继续填充，这样就完成了要求。这样就好比是操作员每小时在记录表上记录一次现场数据，当换班时，由下一班在新的记录表上开始记录一样。

可以另外创建一个报表窗口，在运行时，调用这些保存的报表，查看以前的记录，实现历史数据报表的查询。

这种制作报表的方式既可以作为实时报表观察实时数据，也可以作为历史报表保存。

（2）使用历史数据查询函数　使用历史数据查询函数从组态王记录的历史库中按指定的起始时间和时间间隔查询指定变量的数据，历史数据查询函数的使用方法请参见报表历史数据查询函数，或组态王函数。

如果用户在查询时，希望弹出一个对话框，可以在对话框上随机选择不同的变量和时间段来查询数据，可使用函数 ReportsetHistData2（StartROW，StartCol）；该函数已经提供了方便、全面的对话框供用户操作。但该函数会将指定时间段内查询到的所有数据都填充到报表中来，如果报表不够大，则系统会自动增加报表行数或列数，对于使用固定格式报表的用户来说不太方便。

如果用户想要一个定时自动查询历史数据的报表，而不是弹出对话框，或者历史报表的格式是固定的，要求将查询到的数据添加到固定的表格中，多余查询的数据不需要添加到表

中，这时可以使用函数 ReportSetHistData（ReportName，TagName，StartTime，SepTime，szContent）；使用该函数时，用户需要指定查询的起始时间，查询间隔和变量数据的填充范围。

组态王报表拥有丰富而灵活的报表函数，用户可以使用报表制作一些数据存储、求和、运算、转换等特殊用法。如将采集到的数据存储在报表的单元格中，然后将报表数据赋予曲线控件来制作一段分析曲线等，既可以节省变量，简化操作，还可以重复使用。用户可按照自己的实际用途灵活使用。

2. 制作历史数据报表实例

历史数据报表的制作步骤如下：

表格设计：组态王历史报表的创建和表格样式与前面所述的实时数据报表的创建方法是一致的，因此用户可以参照实时数据报表的设计方法和步骤。设计的历史报表样式如图 7-61 所示。

图 7-61　历史数据报表

在历史报表画面上新建一个“报表查询”的按钮，在弹起时命令语言中输入历史查询函数：ReportSetHistData2（）。使用该函数，不需要任何参数，系统会自动弹出历史数据查询对话框。

查询历史数据：启动组态王运行系统，打开历史数据报表画面，单击“报表查询”按钮，弹出如图 7-62 所示的“报表历史查询”对话框。该对话框共有三个属性页，用于定义历史数据查询的参数。在对话框中输入合适的查询参数值，然后单击“确定”按钮，同时查询“原料罐液位”、“反应罐液位”、“反应罐温度”。最后生成的历史数据报表如图 7-63 所示。

7.4.3　报表函数

报表在运行系统中单元格数据的计算、报表的操作等都是通过组态王提供的一整套报表

图 7-62　“报表历史查询”对话框

系统历史数据报表

报表日期：	09/09/17		10:30:21	
日期	时间	原料罐液位	反应罐液位	反应罐温度
09/09/17	10:10:03	0.00	0.00	0.00
09/09/17	10:11:03	0.00	0.00	0.00
09/09/17	10:12:03	0.00	0.00	0.00
09/09/17	10:13:03	0.00	0.00	0.00
09/09/17	10:14:03	0.00	0.00	0.00
09/09/17	10:15:03	0.00	0.00	0.00
09/09/17	10:16:03	0.00	0.00	0.00
09/09/17	10:17:03	0.00	0.00	0.00
09/09/17	10:18:03	0.00	0.00	0.00
09/09/17	10:19:03	0.00	0.00	0.00
09/09/17	10:20:03	0.00	0.00	0.00
09/09/17	10:21:03	0.00	0.00	0.00
09/09/17	10:22:03	0.00	0.00	0.00
09/09/17	10:23:03	0.00	0.00	0.00
09/09/17	10:24:03	0.00	0.00	0.00
09/09/17	10:25:03	0.00	0.00	0.00

图 7-63　历史数据报表运行结果

函数实现的。报表函数分为报表内部函数、报表单元格操作函数、报表存取函数、报表统计函数、报表历史数据查询函数、报表打印函数等。与报表相关的函数整理如下：

1. 报表内部函数

报表内部函数是指只能在报表单元格内使用的函数，有数学函数、字符串函数、统计函数等。其基本上都是来自于组态王的系统函数，使用方法相同，只是函数中的参数发生了变化，减少了用户的学习量，方便学习和使用。组态王的报表函数中的参数和有关用报表单元格作为参数的函数，其中的参数引用均为这种方法。当参数为多个单元格时，有以下几种情况：

1）如果是任选多个单元格，则使用方法为用逗号将各个单元格的表示分隔，如“a1，b3，c6，h10”。

2）如果选择的为连续的单元格时，可以输入第一个单元格标识和最后一个单元格标识，中间用冒号分割。如选择了 al ~ c10 间的单元格区域：“a1：c10”。

3）报表内部函数中的单元格参数可以使用组态王变量代替，即报表支持的组态王系统函数可以直接在报表中使用。

2. 报表的单元格操作函数

在运行系统中，报表单元格是不允许直接输入的，所以要使用函数来操作。单元格操作函数是指可以通过命令语言来对报表单元格的内容进行操作，或从单元格获取数据的函数。这些函数大多只能用在命令语言中。

（1）设置单个单元格数值

Long nRet = ReportSetCellValue（String szRptName，long nRow，long nCol，float fValue）；

函数功能：将指定报表的指定单元格设置为给定值。

（2）设置单个单元格文本

Long nRet = ReportSetCellString（String szName，long nRow，long nCol，String szValue）；

函数功能：将指定报表的指定单元格设置为给定字符串。

（3）设置多个单元格数值

Long nRet = ReportSetCellValue2（String szRptName，long nStartRow，long nStartCol，long nEndRow，long nEndCo1，float fValue）；

函数功能：将指定报表的指定单元格区域设置为给定值。

（4）设置多个单元格文本

Long nRet = ReportSetCellString2（String szRptName，long nStartRow，long nStartCol，long nEndRow，long nEndCol，String szValue）；

函数功能：将指定报表指定单元格设置为给定字符串。

（5）获得单个单元格数值

float fValue = ReporGetCellValue（String szRptName，long nRow，long nCol）；

函数功能：获取指定报表的指定单元格的数值。

（6）获得单个单元格文本

String szValue = ReportSetCellString（String szRptName，long nRow，long nCol）；

函数功能：获取指定报表的指定单元格的文本。

String szValue = ReportSetCellString（String szRptName，long nRow，long nCol）；

函数功能：获取指定报表的指定单元格的文本。

（7）获取指定报表的行数

Long nRows = ReportGetRows（String szRptName）；

函数功能：获取指定报表的行数。

（8）获取指定报表的列数

Long nCols = ReportGetColumns（String szRptName）；

函数功能：获取指定报表的列数。

3. 报表存取函数

报表存取函数主要用于存储指定报表和打开查阅已存储的报表。用户可利用这些函数保存和查阅历史数据、存档报表。

（1）存储报表

Long nRet = ReportSaveAs（String szRptName，String szFileName）；

函数功能：将指定报表按照所给的文件名存储到指定目录下。

（2）读取报表

Long nRet = ReportLoad（String szRptName，String szFileName）；

函数功能：将指定路径下的报表读到当前报表中来。

4. 报表统计函数

对指定单元格区域内的单元格进行求平均值、求和运算，结果显示在当前单元格内。

（1）Average

函数功能：对指定单元格区域内的单元格进行求平均值运算，结果显示在当前单元格内。

使用格式：=Average〔“单元格区域”）

（2）Sum

函数功能：将指定单元格区域内的单元格进行求和运算，显示到当前单元格内。单元格区域内出现空字符、字符串等都不会影响求和。

使用格式为=Sum（“单元格区域”）。

5. 报表历史数据查询函数

报表历史数据查询函数将按照用户给定的起止时间和查询间隔，从组态王历史数据库中查询数据，并填写到指定报表上。

（1）ReportSetHistData（）

ReportsetHistData（String szRptName，String szTagName，Long nStartTime，Long nSepTime，String szContent）；

函数功能：按照用户给定的参数查询历史数据。

（2）ReportSetHistData2（）

ReportSetHistData2（StartRow，StartCol）；

函数参数；StartRow。指定数据查询后，在报表中开始填充数据的起始行。

Startcol：指定数据查询后，在报表中开始填充数据的起始列。

这两个参数可以省略不写（应同时省略），省略时默认值都为1。

函数功能：使用该函数，不需要任何参数，系统会自动弹出历史数据查询对话框。

6. 报表打印函数

（1）报表打印函数　报表打印函数根据用户的需要有两种使用方法：一种是执行函数时自动弹出“打印属性”对话框，供用户选择确定后，再打印；另一种是执行函数后，按照默认的设置直接输出打印，不弹出“打印属性”对话框，适用于报表的自动打印。报表打印函数原型为

ReportPrint2(String szRptName)

或者 ReportPrint2（String szRptName，EV－LONGIEV－ANALOGIEV－DISC）

函数功能：将指定的报表输出到打印配置中指定的打印机上打印。

参数说明：szRptName 为要打印的报表名称。

EV－LONGIEV－ANALOGIEV－DISC：整型或实型或离散型的一个参数，当该参数不为 0 时，自动打印，不弹出“打印属性”对话框。如果该参数为 0，则弹出“打印属性”对话框。

例如，自动打印“实时数据报表”：ReportPrint2（“实时数据报表”）或 ReportPrint2（“实时数据报表”，1）；手动打印时，弹出“打印属性”对话框：ReportPrint2（“实时数据报表”，0）。

（2）报表页面设置函数　开发系统中可以通过报表工具箱对报表进行页面设置，在运行系统中则需要通过调用页面设置函数来对报表进行设置。页面设置函数的原型为

ReportPageSetup（ReportName）

函数功能：设置报表页面属性，如纸张大小、打印方向、页眉/页脚设置等。

参数说明：szRptName 为要打印的报表名称。

例如，“实时数据报表”的页面设置 ReportPageSetup（“实时数据报表”）。

（3）报表打印预览函数　运行中当页面设置好以后，可以使用打印预览查看打印后的效果。打印预览函数原型为 ReportPrintSetup（ReportName）。

函数功能：对指定的报表进行打印预览。

参数说明：szRptName 为要打印的报表名称。

例如“实时数据报表”进行打印预览：ReportPrintSetup（“实时数据报表”）。

执行打印预览时，系统会自动隐藏组态王的开发系统和运行系统窗口，结束预览后恢复。

练一练：

1）制作一个实时报表。

2）制作一个日报表。

3）制作一个历史数据查询。

4）练习报表的保存、打印、查询等功能。

7.5　系统安全

在组态王系统中，为了保证运行系统的安全运行，对画面上的图形对象设置了访问权限，同时给操作者分配了访问优先级和安全区，只有操作者的优先级大于对象的优先级且操作者的安全区在对象的安全区内时才可访问，否则不能访问画面中的图形对象。

7.5.1　设置用户的安全区与权限

优先级分 1～999 级，1 级最低 999 级最高。每个操作者的优先级别只有一个。系统安全 区共有 64 个，用户在进行配置时。每个用户可选择除“无”以外的多个安全区，即一个用户可有多个安全区权限。用户安全区及权限设置过程如下：

1）在工程浏览器窗口左侧“工程目录显示区”中双击“系统配置”中的“用户配置”选项，弹出创建用户和安全区配置对话框，如图 7-64 所示。

图 7-64　用户和安全区配置对话框

2）单击此对话框中的“编辑安全区”按钮，弹出安全区配置对话框，如图 7-65 所示。

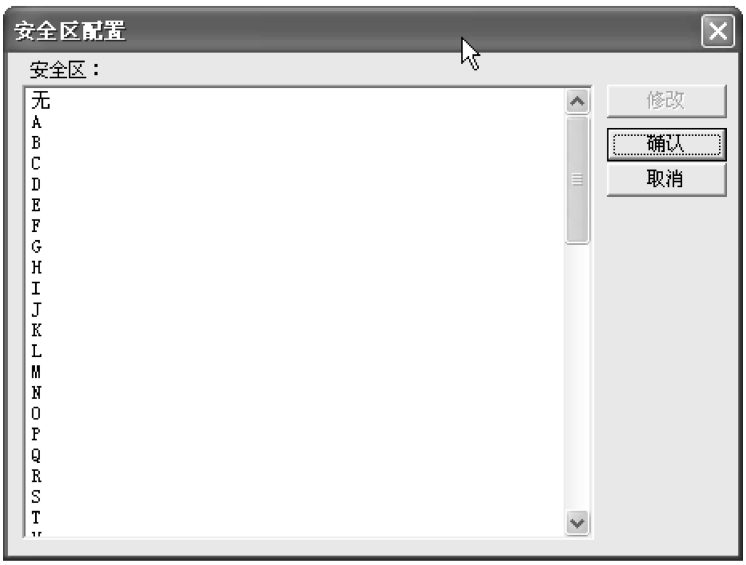


图 7-65　安全区配置对话框

选择“A”安全区并利用“修改”按钮将安全区名称修改为“反应车间”。

3）单击“确认”按钮关闭对话框，在用户和安全区配置对话框中单击“新建”按钮，在弹出的定义用户组和用户对话框中配置用户组，如图 7-66 所示。

对话框设置如下：

类型：用户组；用户姓名：反应车间组；安全区：无。

4）单击“确认”按钮关闭对话框，回到“用户和安全区配置”对话框后再次单击“新建”按钮，在弹出的定义用户组和用户对话框中配置用户，对话框的设置如图 7-67 所示。

图 7-66　定义用户组和用户对话框

图 7-67　定义用户组和用户对话框

用户密码设置为 master。

5）利用同样方法再建立两个操作员用户，用户属性设置如下所示：

操作员 1：

类型：用户；

加入用户组：反应车间用户组；

用户名：操作员 1；

用户密码：operater1；
用户注释：具有一般权限；
登录超时：5；
优先级：50；
安全区：反应车间。
操作员 2：
类型：用户；
加入用户组：反应车间用户组；
用户名：操作员 2；
用户密码：operater2；
用户注释：具有一般权限；
登录超时：5；
优先级：150；
安全区：无。

6）单击“确认”按钮关闭定义用户对话框，用户安全区及权限设置完毕。

7.5.2 设置图形对象的安全区与权限

与用户一样，图形对象同样具有 1～999 个优先级别和 64 个安全区，在前面编辑的“监控中心”画面中设置的“退出”按钮，其功能是退出组态王运行环境。而对一个实际的系统来说，可能不是每个登录用户都有权利使用此按钮，只有上述建立的反应车间用户组中的“管理员”登录时可以按此按钮退出运行环境，反应车间用户组的“操作员”登录时就不可操作此按钮。其对象安全属性设置过程如下：

1）在工程浏览窗口中打开“监控中心”画面，双击画面中的“系统退出”按钮，在弹出的“动画连接”对话框中设置按钮的优先级为 100，安全区为反应车间。

2）单击“确定”按钮关闭此对话框，按钮对象的安全区与权限设置完毕。

3）单击“文件”菜单中的“全部存”命令，保存您所作的修改。

4）单击“文件”菜单中的“切换到 View”命令，进入运行系统，运行“监控中心”画面。在运行环境界面中单击“特殊”菜单中的“登录开”命令，弹出“登录”对话框，如图 7-68 所示。

图 7-68　用户登录对话框

当以上述所建的“管理员”登录时，画面中的“系统退出”按钮为可编辑状态，单击此按钮退出组态王运行系统；当分别以“操作员 1”和“操作员 2”登录时，“系统退出”

按钮为不可编辑状态，此时按钮是不能操作的。这是因为对“操作员 1”来说，他的操作安全区包含了按钮对象的安全区（即：反应车间安全区），但是权限小于按钮对象的权限（按钮权限为 100，操作员 1 的权限为 50）。对于“操作员 2”来说，他的操作权限虽然大于按钮对象的权限（按钮权限为 100，操作员 2 的权限为 150），但是安全区没有包含按钮对象的安全区，所以这两个用户登录后都不能操作按钮。

练一练：

配置两个用户分别能够操作不同的对象。

第 8 章　组态王与外界的通信

一个好的应用程序必不可少地要与外界进行通信，组态王作为一款优秀的工控软件在这方面也提供了强大的支持。本章重点介绍组态王与外部软硬件设备的通信，包括组态王与PLC、单片机等各种硬件设备的通信，组态王与数据库之间的通信，以及组态王对外部设备的管理等。

8.1　组态王与外部设备通信机制

设备驱动程序（I/O Server）是组态软件与 PLC、智能仪表等设备交互通信的桥梁，由于组态软件面向的就是开放式测控设备，因此建立 PC 与设备间的通信链路不存在理论上的障碍。各种测控设备也越来越多地采用标准通信接口，使设备与 PC 间及设备间的互联通信越来越简便。

8.1.1　设备驱动程序在组态软件中所处地位及数据流图

如图 8-1 所示，I/O Server 直接负责从设备采集实时数据并将操作命令下达给设备，I/O Server 的可靠性将直接影响组态软件的性能。在大多数情况下，I/O Server 是与设备相关的，即针对某种设备的驱动程序不能驱动其他种类的设备。一般地，设备制造商会提供 PC 与设备间进行数字通信的接口协议和物理接口标准。物理接口标准规定使用何种通信介质、链路层的接口标准，如 RS-232、RS-485、以太网等；接口协议规定通信双方约定的命令及数据响应格式、数据校验方式等。I/O Server 主要是按照接口协议的规定向设备发送数据请求命令，

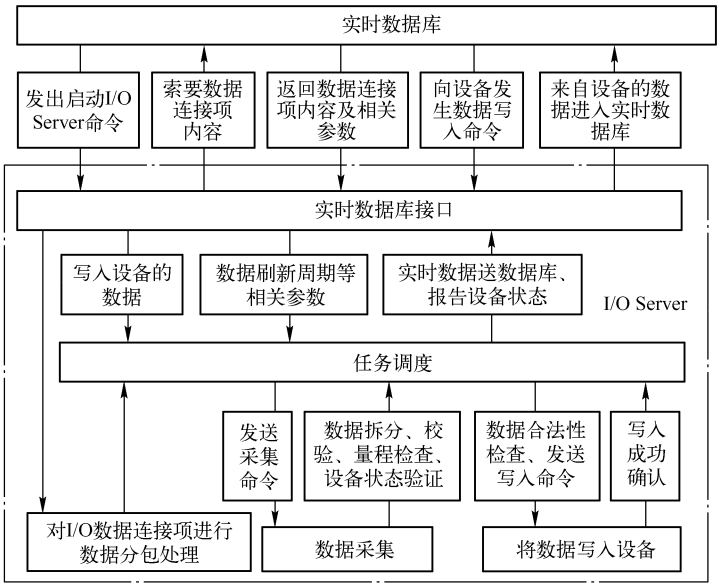


图 8-1　组态软件设备驱动程序的数据流

对返回数据进行拆包，从中分离出所需数据（即组态的数据连接项和设备状态数据）。多数设备的通信接口协议都有若干条读写命令，分别用来读写设备上不同类别的数据，而每一条命令又可以读写同类别的多条数据，具体能读写几条是由接口协议规定的。使用组态软件做I/O数据连接的工程技术人员不必了解这些细节，只需按照I/O Server的说明书组态数据库变量与设备数据项的对应关系即可。I/O Server首先要将组态的数据连接项按照接口协议的要求按照类别分好，把通过一次读写操作能够处理的数据连接项存放在一起，称为一个数据包。I/O Server主要以数据包为单位进行数据处理，这样会大大提高通信效率。

一般设备的接口协议都会提供设备状态信息的访问方法，即组态软件可以直接读取设备的状态信息；如果接口协议不含有设备状态信息的访问方法，I/O Server只能将通信的状态信息送给实时数据库和界面系统，如通信超时、设置数据成功等。I/O Server的通信状态信息将作为系统报警显示在界面系统的系统报警窗中。

8.1.2 设备驱动程序完成的主要功能

1. 从I/O设备采集所需数据进行链路维护

I/O Server要将来自设备的数据转换成实时数据库需要的数据类型（实数、整型数、字符或字符串型等），同时要对越界数据做合理解释，以避免操作人员得到错误的数据。例如，有的设备以“0XFFFF”（十六进制）表示回路断开，有的设备以“----”（ASCII）表示数据超过上限等。

为提高数据的采集效率和数据安全，要将所有数据连接项划分成只读、只写、可读写三种类型，以防止误写、误读的发生。

一个I/O Server可以同时处理多个同类设备，如果其中某个设备出现故障不能响应通信，则I/O Server要花费多余的时间等待其响应，从而减慢了整个系统的响应周期。因此对于故障设备，I/O Server要减少访问频次，以免影响整个系统的数据刷新周期，当故障设备恢复正常后再将设备的采集周期恢复原值。

2. 执行来自操作员的I/O命令管理输出队列

界面系统要对送给I/O设备的命令数据进行第一步检查，检查数据类型是否正确，数据是否越界。实时数据库对数据进行第二阶段检查，主要检查越界情况。I/O Server对来自操作员的I/O命令要针对设备的特殊要求逐级做合法性检查，也是最严格的检查，如首先检查是否只读数据，再检查数值是否越界（如串级回路的回路方式：0表示自动，1表示手动，2表示串级，其他数据则为非法的越界数据）等。

3. 与实时数据库系统进行无缝连接

I/O Server与实时数据库间应采用进程间通信、直接内存映射、OLE方式或其他更为有效的内部通信方式，较采用第三方的DDE设备驱动程序或OPC Server具有更高的通信效率，同时可以迅速报告设备的故障信息，产生系统报警。

I/O Server对来自设备的数据可以进行初步预处理，在一段时间内数值没有发生变化的数据可以不向实时数据库传送或减少传送次数，以进一步提高数据通信的效率。

4. 设备驱动程序能够连接的设备种类

1）DCS、PLC等控制设备；

2）带数字通信接口（RS232/RS422/RS485、以太网、现场总线等）的智能仪表，如流

量计等计量设备、报警器等安防设备、遥测传感设备；

3）PC 总线工业计算机设备；

4）带开放接口的基于 PC 的数据处理系统。

5. 设备驱动程序的技术指标

（1）采样周期　采样周期取决 I/O Server 调度效率，但实时数据库的数据刷新速度则不只是由 I/O Server 决定的，还与设备的响应速度、挂接的设备数量等因素有关。一般的串行设备当波特率为 9600bit/s 时，数据通信速度可以保证不低于 900B/s。

（2）回写数据速度　经常需要由操作员向设备下发控制指令，如修改回路状态、运算系数及对控制过程实施人为干预等。

6. 使用设备驱动程序的注意事项

1）及时处理来自 I/O Server 的系统报警，I/O Server 发出的报警都是重要的系统报警，如设备掉电、物理链路损坏、通信适配器故障等，必须及时处理，否则会引起更为严重的后果。

2）正确设置 I/O Server 的采样周期，因为有些设备如果采样周期太快会引起设备死机或工作不正常，如果采样周期设置太慢又会影响数据刷新速度。

3）I/O Server 必须设置安全密码，以防止非法退出。

4）界面系统的回写数据频率不要大于采样周期的 3 倍，因为 I/O Server 都是写优先的，当实时数据库有回写数据的任务时，I/O Server 会优先处理，而将采集任务作为次要任务拖后处理。当实时数据库频繁不断地将大量回写数据送给 I/O Server 时，I/O Server 将无暇进行数据采集，只能疲于应付数据回写任务。

8.2　组态王中的逻辑设备

组态王对设备的管理是通过对逻辑设备名的管理实现的，具体讲就是每一个实际 I/O 设备都必须在组态王中指定一个唯一的逻辑名称，此逻辑设备名就对应着该 I/O 设备的生产厂家、实际设备名称、设备通信方式、设备地址、与上位 PC 的通信方式等信息内容（逻辑设备名的管理方式就如同对城市长途区号的管理，每个城市都有一个唯一的区号相对应，这个区号就可以认为是该城市的逻辑城市名，比如北京市的区号为 010，则查看长途区号时就可以知道 010 代表北京）。在组态王中，具体 I/O 设备与逻辑设备名是一一对应的，有一个 I/O 设备就必须指定一个唯一的逻辑设备名，特别是设备型号完全相同的多台 I/O 设备，也要指定不同的逻辑设备名。组态王中变量、逻辑设备与实际设备对应的关系如图 8-2 所示。

例如：设有两台型号为三菱公司 FX_2－60MR PLC 作下位机控制工业生产现场，同时这两台 PLC 均要与装有组态王的上位机通信，则必须给两台 FX_2－60MR PLC 指定不同的逻辑名，如图 8-3 所示。

其中 PLC1、PLC2 是由组态王定义的逻辑设备名（此名由工程人员自己确定），而不一定是实际的设备名称。

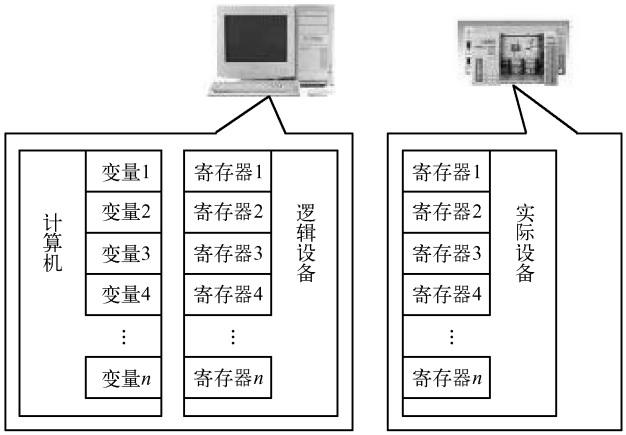


图 8-2　组态王中变量、逻辑设备与实际设备对应的关系

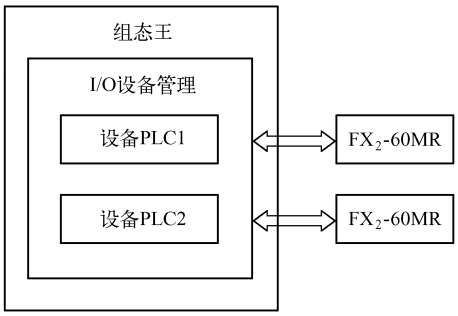


图 8-3　组态王中逻辑设备与实际设备的对应关系

另外，组态王中的 I/O 变量与具体 I/O 设备的数据交换就是通过逻辑设备名来实现的，当工程人员在组态王中定义 I/O 变量属性时，就要指定与该 I/O 变量进行数据交换的逻辑设备名，I/O 变量与逻辑设备名之间的关系如图 8-4 所示。

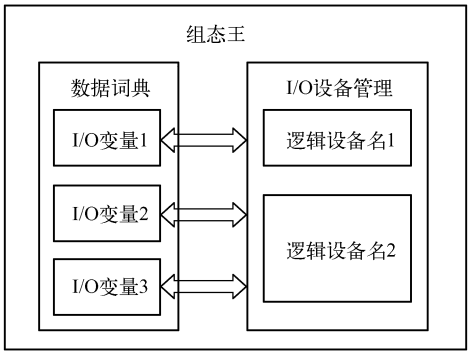


图 8-4　组态王中变量的定义

组态王把那些需要与之交换数据的设备或程序都作为外部设备。外部设备包括：下位机（PLC、仪表、模块、板卡、变频器等），它们一般通过串行口和上位机交换数据；其他 Windows 应用程序，它们之间一般通过 DDE 交换数据；外部设备还包括网络上的其他计算机。

只有在定义了外部设备之后，组态王才能通过 I/O 变量和它们交换数据。为方便定义外部设备，组态王设计了“设备配置向导”，引导我们一步步完成设备的连接。

8.3 组态王中的仿真 PLC

程序在实际运行中是通过 I/O 设备和下位机交换数据的，当程序在调试时，可以使用仿真 I/O 设备模拟下位机向画面程序提供数据，为画面程序的调试提供方便。组态王提供一个仿真 PLC 设备，用来模拟实际设备向程序提供数据，供用户调试。

在使用仿真 PLC 设备前，首先要定义它，实际 PLC 设备都是通过计算机的串口向组态王提供数据，所以仿真 PLC 设备也是模拟安装到串口 COM 上，定义过程和步骤如下：

1）在组态王的工程浏览器中，从左边的工程目录显示区中选择大纲项设备下的成员名 COM1 或 COM2，然后在右边的目录内容显示区中用左键双击“新建”图标，则弹出“设备配置向导”对话框，如图 8-5 所示。

图 8-5 “设备配置向导”对话框

2）选择亚控公司的仿真 PLC，单击“下一步”按钮，在编辑框输入一个仿真 PLC 设备的逻辑名称，例如设定为“simu”。

3）继续单击“下一步”按钮，则弹出“设备配置向导——选择串口号”对话框，如图 8-6 所示。

在下拉式列表框中列出了 32 个串口设备（COM1 ~ COM32）供用户选择，例如从下拉式列表框中选中 COM2 串口。

注意：这里定义的串口是虚拟的，实际仿真 PLC 设备并不使用计算机的 COM 口，而且 COM 口也不需要配置。

4）继续单击“下一步”按钮，则弹出“设备配置向导——设备地址设置指南”对话框，如图 8-7 所示。

图 8-6　串口选择对话框

图 8-7　设备地址设置对话框

在编辑框中输入仿真 PLC 设备的地址。

5）继续单击“下一步”按钮，则弹出“设备配置向导——通信参数”对话框，如图 8-8 所示。

6）继续单击“下一步”按钮，则弹出“设备配置向导——信息总结”对话框。单击“完成”按钮，则设备安装完毕，单击“上一步”，可返回上一次操作进行修改。仿真 PLC 设备安装完毕后，可在工程浏览器进行查看，选择大纲项设备下的成员名 COM2，则在右边的目录内容显示区可以显示已安装的设备，如图 8-9 所示。

图 8-8　通信参数设置对话框

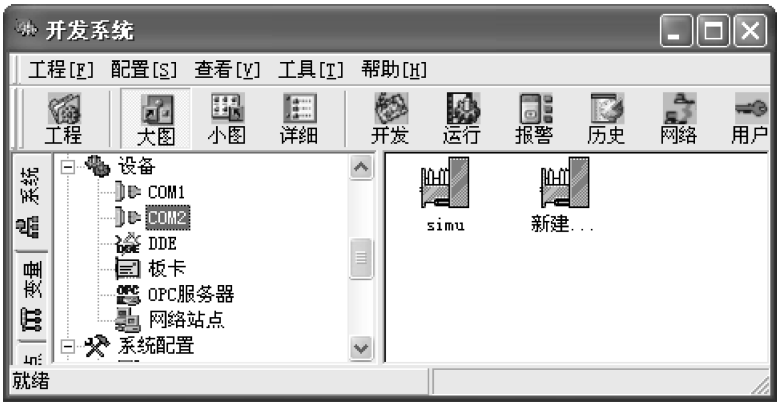


图 8-9　已安装完成的设备

8.4　组态王与 AB PLC 的通信

Rockwell Automation 公司的 AB PLC 是一个著名的可编程序控制器产品。其产品体系基本上涵盖了工业控制产品的各个领域，并以产品的可靠而著称。在国内 PLC 市场的占有率将近 30%。

由于 AB PLC 的种类繁多，且各种类型的 PLC 支持的通信协议不尽相同，使得第三方的 HMI 软件都不能很好地与 AB PLC 进行连接。组态王提供的与 AB PLC 之间的通信方式包括：

直接驱动方式：AB PLC 的串口（编程口）提供 DF1 协议，该协议开放，亚控公司提供了直接驱动。

OPC 通信方式：AB PLC 提供的串口、以太网以及 DH485 接口都可以使用 OPC 方式和组态王通信。但对于除串口以外的通信方式，设备协议都不公开，只能使用 OPC 的通信方式。

RSLinx 软件是 AB PLC 通用的通信配置软件（2.2 版以上版本支持对以太网的网关配置），目前主要有以下几个版本：RSLinx OEM、RSLinx Professional 、RSLinx Gateway、RSLinx SDK 4 个版本，且均支持 OPC 服务（2.3 版以上版本支持 OPC2.0 版）。

组态王软件已内嵌了 OPC 服务功能，支持 OPC 客户端和 OPC 服务器的工作方式。这样组态王就可以很方便地与 AB PLC 进行通信。本节对此作详细介绍。

具体的配置方法如下：

在组态王与 AB PLC 进行 OPC 的配置时，必须要有 AB PLC 及相关的通信附件。

1）安装 AB PLC 、通信卡、相关的软件，并作物理连接；

2）使用 RSLinx 连接 AB PLC 的通信网络。

配置通信卡，如图 8-10 所示，选择“ConFigure Drivers”。

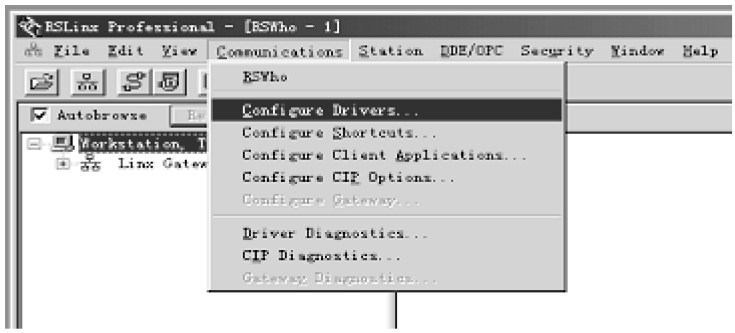


图 8-10　通信卡配置

弹出如图 8-11 所示的配置窗口。

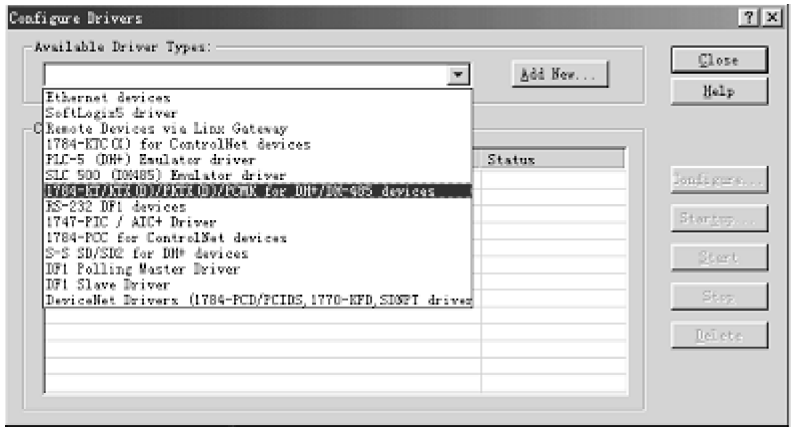


图 8-11　通信卡配置对话框

选择对应的通信卡，并进行端口配置（可参照附带的安装手册），配置成功后，将显示所配置的设备的运行状态。

使用 RSWho 命令查找连接的 PLC 设备，如图 8-12 所示。

RSWho 命令成功地执行后，将在设备列表中列出与本机连接的所有硬件设备。

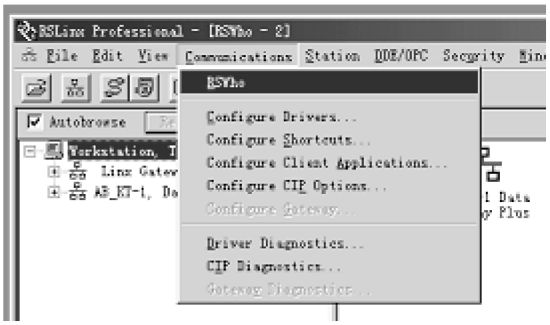


图 8-12　使用 RSWho 命令

3）使用 Topic Configuration 来进行 OPC 服务的配置，如图 8-13 所示。

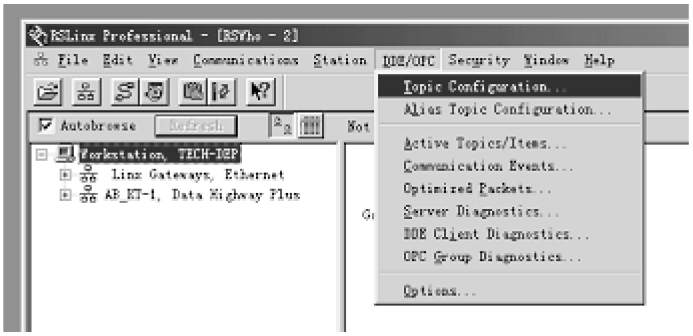


图 8-13　配置 OPC 服务

在 Date Source 配置框中配置 Topic，选择一个物理设备（PLC），按“New”按钮即可建立一个 Topic，单击“Done”即可，如图 8-14 所示。

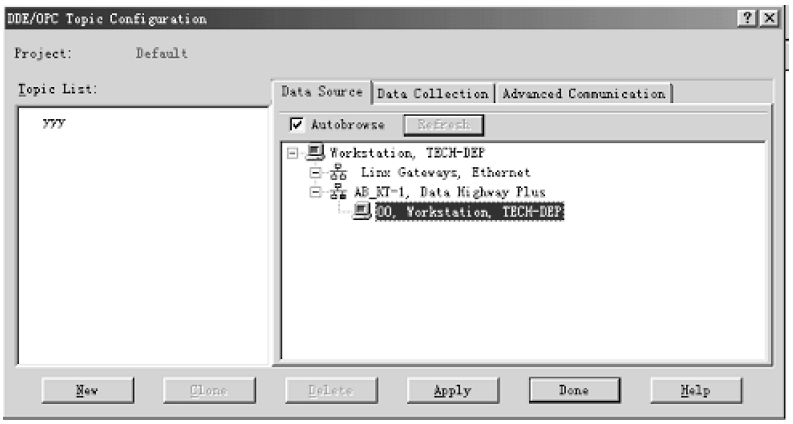


图 8-14　配置 Topic

4）在组态王中建立 y 与 OPC 设备的连接。

添加 OPC 设备。在组态王的工程器中选择设备→OPC 服务器→新建，则弹出如图 8-15 所示的窗口。在 OPC 服务器列表中选择 RSLinx OPC Server，确定即可。

图 8-15　建立与 OPC 设备的连接

5）在数据词典里加入 I/O 点。

在组态王的工程器中选择数据库→数据词典→新建。确定变量名、变量类型等变量属性，在连接设备中选择 RSLinx OPC Server；手工键入寄存器地址，地址的格式为“［Topic 名］变量别名”或“Topic 名：变量寄存器地址”，如图 8-16 所示。单击确认即可。

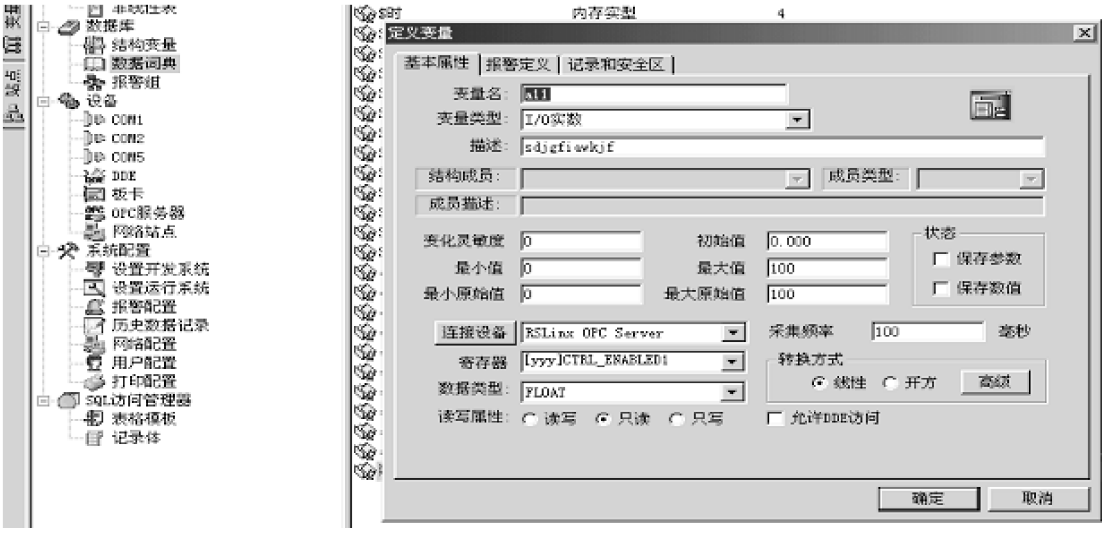


图 8-16　创建变量

在完成上述步骤之后，就可以通过组态王对 AB PLC 进行监视和控制了。

8.5 组态王与西门子 PLC 的通信

组态王提供了多种方式与西门子 PLC 进行通信，具体说明如下：

1. 串口通信方式

硬件连接：此种方式使用 S7－200 系列 PLC 上的 PPI 编程口，使用西门子标准编程电缆或标准 485 电缆连接到计算机串口上。

当使用计算机的 RS－232 通信口时：建议使用西门子专用紫色电缆和网络接头＋常规有源 RS－485/232 转换模块（如研华的 ADAM4520）进行 PLC RS－485 编程口和计算机标准 232 口的连接）。

当使用计算机扩展 RS－485 口时：建议使用西门子专用紫色电缆和网络接头直接进行 PLC RS－485 编程口和计算机 RS－485 口的连接。注意：PLC 的 PPI 口中 9 针口：3 为 DATA＋；8 为 DATA－。

适用场合：一般适用于一个 PLC 和一台 PC 进行直接串口通信的场合，PLC 和 PC 之间距离较近；

组态王对应驱动（常用直接驱动有 3 个）：

1）PLC→西门子→S7－200 系列→PPI；

2）PLC→西门子→S7－200 系列→自由口；

3）PLC→西门子→S7－200 系列→Modbus。

特殊软件需求：组态王所在的计算机不需要安装 S7－200 编程软件。各自具体配置按照组态王对应驱动帮助执行即可。

注意：因为 PPI 协议的特殊性，读取一个数据包一般需要 400 ms 的时间，当用户反馈 PPI 通信速度慢时，请以此标准分析用户工程通信速度慢是否在合理的范围；如果在该驱动所支持的合理范围内，用户仍然不满意时，建议更换为自由口或 Modbus 通信方式。当用户使用自由口或 Modbus 通信驱动时，需要向 PLC 中下载对应协议程序，该程序由亚控公司提供。用户必须在此通信协议基础上继续编写用户自己的逻辑控制程序，该逻辑控制程序中用到的寄存器不能和亚控公司提供的协议中所占用的 V 区地址冲突：PLC 自由口协议程序默认占用了 V300 及以前的地址；PLCModbus 协议程序 占用 V1000 及以前的地址。

2. MPI 电缆通信方式

硬件连接：此种方式使用 S7－300、S7－400 PLC 上的 MPI 编程口，使用西门子标准编程电缆连接到计算机串口或 USB 接口上；

适用场合：一般适用于一个 PLC 和一台 PC 进行直接串口通信的场合，PLC 和 PC 之间距离较近；

组态王对应驱动：在组态王中对应的设备定义向导为：

对于 S7－300 系列

PLC→西门子→S7－300 系列→MPI（串口）；

PLC→西门子→S7－300 系列→MPI（USB）。

对于 S7－400 系列

PLC→西门子→S7－400 系列→MPI（串口）；

PLC→西门子→S7－400 系列→MPI（USB）。

特殊软件需求：组态王所在的计算机必须安装 STEP7 编程软件。

3. MPI 通信卡方式

硬件连接：此种方式使用 S7200 PLC 上的编程口，计算机上插一块西门子公司的 CP5611（或 CP5613 等 MPI 通信卡——具体根据带 PLC 类型和数量，由西门子公司确定使用何种通信卡），通信卡和 PLC 之间建议使用西门子公司提供的标准转换接头和通信电缆实现硬件连接；此种链路常规通信距离为 50 m，超过时请咨询西门子公司，购买中继器扩展通信链路长度。

适用场合：一般适用于一台 PC 和多个 PLC 进行通信的场合；或者多台 PLC 和多台 PC 进行链接（MPI 为多点接口协议，支持多个上位 PC 进行 MPI 通信，具体能够支持的最多上位 PC 数量建议和西门子公司具体确定）。

组态王对应驱动：在组态王中对应的设备定义向导为 PLC→西门子→S7－200 系列→S7－200MPI（通信卡）。

特殊软件需求：组态王所在的计算机必须安装 STEP7 编程软件。具体配置按照组态王对应驱动帮助执行即可（与组态王版本相关的注意事项请参考下一节的 S7－300/400 PLC 对应的 MPI 驱动使用说明部分）。

4. 以太网通信方式

硬件连接：此种方式使用西门子 S7－200 上扩展的 CP243－1 以太网模块，PC 上可以使用普通以太网卡或者西门子公司提供的 CP1613 等以太网卡方式，PLC 和 PC 之间通过以太网线进行连接。适用场合：一般用于对通信速度要求较高的场合；通信距离在局域网允许距离之内。

组态王通信支持的两种方式：

1）组态王提供直接驱动：在组态王中对应的设备定义向导为 PLC→西门子→S7－200 系列→TCP。

特殊软件需求：无，即不需要在组态王所在的计算机上安装 STEP7 或 Simatic net 通信软件，该驱动在组态王 6.52 及以前发布的光盘中还没有（该类版本需要时可以从亚控公司技术部索取）；组态王 6.52 以后的版本中默认提供。

2）使用 OPC 进行和组态王通信：需要本机安装西门子公司提供的 Simatic net6.0 或以上版本基于以太网的授权软件。该软件提供 OPCServer 功能实现和上位机组态王通信。具体请参考西门子公司或亚控发布的相关 OPC 配置文档（网站下载或从亚控公司技术部索取）。

5. Profibus －DP 通信方式

硬件连接：此种方式使用西门子 PLC 上的 DP 接口（可以是 CPU 上集成的 DP 接口或者扩展的 DP 通信模块），计算机上扩展一块 CP5611 或者 CP5613 等通信卡。卡和 PLC 的 DP 接口之间使用西门子标准的网络接头和通信电缆进行连接。一般一块通信卡通过 DP 总线可以连接多台 PLC，具体可以连接的数量根据设备型号向西门子公司确认即可。使用该方式通信时，需要在本机上安装 STEP7 编程软件和 Simatic net 6.0（或以上版本）的通信配置软件和授权。并通过 STEP7 编程软件为 PLC 上的 DP 接口配置为 DP 协议（Slave 站）；通过 Set PG/PC interface 接口为 CP5611（或者 CP5613 卡）配置为 DP 协议（做 master 站）。适用场

合：一般适用于数据交换量少，速度要求较高的场合；组态王通信支持的两种方式：

1）组态王提供直接驱动：在组态王中对应的设备定义向导为 PLC→西门子→S7－200 系列→Profibus→DP。

该驱动只支持 DP 通信卡配置为唯一主站，所有 PLC 等必须配置为从站的工作模式，并且只支持一个上位机组态王和所有从站 PLC 进行通信。

特殊软件需求：需要在组态王所在的计算机上安装 Simatic net6.0（或以上版本）的基于 DP 的授权通信软件，具体使用方法请参考组态王 6.52 或以上版本软件中自带的驱动帮助文档。

2）使用 OPC 进行和组态王通信：需要本机安装西门子公司提供的 Simatic net 6.0 或以上版本基于 DP 的授权软件。该软件提供 OPC Server 功能实现和上位机组态王通信。具体请参考西门子公司或亚控公司发布的相关 OPC 配置文档（网站下载或从亚控公司技术部索取）。

6. Profibus －S7 通信方式

硬件连接：此种方式使用西门子 PLC 上的 DP 接口（可以是 CPU 上集成的 DP 接口或者扩展的 DP 通信模块），计算机上扩展一块 CP5611 或者 CP5613 等通信卡。通信卡和 PLC 的 DP 接口之间使用西门子标准的网络接头和通信电缆进行连接。一般一块通信卡通过 DP 总线可以连接多台 PLC，具体可以连接的数量根据设备型号向西门子公司确认即可。

适用场合：一般用于对数据通信速度要求较高的场合。

组态王通信支持的两种方式：

1）组态王提供直接驱动：在组态王中对应的设备定义向导为 PLC→西门子→S7－200 系列→Profibus→S7。特殊软件需求：需要在组态王所在的计算机上安装 STEP7 编程软件，但不需要安装 Simatic net 软件；具体使用方法请参考组态王 6.52 或以上版本软件中自带的驱动帮助文档。

2）使用 OPC 进行和组态王通信：需要本机安装西门子公司提供的 Simatic net6.0 或以上版本基于 S7 的授权软件。该软件提供 OPCServer 功能实现和上位机组态王通信。具体请参考西门子公司或亚控发布的相关 OPC 配置文档（网站下载或从亚控公司技术部索取）。

以上两种通信方式都支持和多个上位机器组态王同时通信，具体数量请用户和西门子公司咨询，目前了解支持的上位机数量≤7。

7. Modem 通信方式

硬件连接：西门子 S7－200 提供了一款集成 Modem 接口且具有 Modbus RTU 协议的通信模块——CP241。组态王和该模块进行通信时，需要 PC 上插一个主叫的 Modem，CP241 的 Modem 接口就相当于集成了一个普通的被叫 Modem。适用场合：一般用于通信距离较远，希望使用 Modem 链路进行通信的场合。组态王通信支持的驱动：

1）PLC→莫迪康→Modbus RTU→串口；

2）PLC→西门子→S7－200 系列→EM241ModbusRtu→串口。

特殊说明：通信按照常规 Modem 通信进行配置即可。EM241 提供的文档中提供 PLC 的 V、I、Q 各寄存器和 Modbus 功能码地址对应表。组态王中定义变量时，其寄存器和该文档中 Modbus 地址完全一致即可读取到 PLC 的 V、I、Q 等地址。

8.6 组态王与三菱、欧姆龙等 PLC 的通信概述

对于其他众多厂家的 PLC 产品，组态王也提供了广泛的支持，本节对此做一概述。

1. 三菱 PLC

FX 系列：编程口，扩展的 RS－232/485 接口组态王都支持，都有对应的驱动。

A/ANA 系列：

1）编程口需要使用我们的 EZSOCKET 驱动。

2）支持以太网通信。

三菱 Q 系列：

1）编程口：编程口需要使用 EZSOCKET 驱动。

2）扩展 QJ71C24 通信模块（串口）：使用 QJ71C24 专用驱动或者 EZSOCKET 驱动。

3）以太网接口：提供直接的以太网驱动。

2. OMRON PLC

串口通信：hostlink 协议、Omron 系列 PLC 的编程口都支持该协议。

以太网方式：组态王提供直接的以太网驱动向导，方便用户使用。

Controllerlink 专用总线通信方式：本机需要安装 OMRON 公司授权提供的 Fins gateway 软件，通过该软件配置 controllerlink 网络。

3. MODICON PLC

串口通信：Modbus RTU 协议。

以太网通信：Modbus TCP/IP 协议。

备注：凡是满足标准 Modbus RTU 或者 ModbusTCP/IP 协议的设备都可以使用组态王对 MODICON 提供的通信协议进行通信。

8.7 组态王与单片机之间的连接

要使组态王与单片机之间进行通信，最简便的办法还是使用串行通信。虽然 PC 和 80C51 单片机本身都具有串行接口，但 PC 和单片机串行通信接口电平不一致。因此要实现这种连接必须进行接口电路设计。串行接口电路标准通常有 RS－232C、RS－485、RS－422 等。RS－232C 简单、通用；但它的最大传输距离只有 15 m，而且只能实现两台计算机之间的串行通信。要使 PC 与多单片机的远程串行通信，通常采用 RS－485 串行接口，它的最大传输距离可达 1200 m，传送速度可达 100 kbit/s。且可以实现多台计算机之间的串行通信。但一般 PC 和 80C51 单片机都没有安装此接口。如果需要必须为它们设计 RS－485 接口。目前，将 80C51 串行通信接口转换成 RS－485 接口的芯片较多，常用的是 MAXIM 公司生产的 MAX485 芯片。其中 A、B 是 RS－485 总线接口，DI 是发送端，RO 是接收端，分别与单片机串行口的 TXD、RXD 相连；RE、DE 为收发使能端，可接单片机 P1.1 口，由它作为收发使能控制。而数据采集和控制通过 P0 口进行。

8.7.1 编制通信协议

为了保证通信的正确性，上、下位机必须遵从相同的通信协议。同时还要根据通信协议

编写组态王驱动程序。组态王提供了单片机串行通信协议和驱动程序。89C51 单片机的串行通信方式有 4 种，当多机通信时通常使用方式 3。即 1 位起始位，8 位数据位，1 位校验位，1 位停止位。其中校验位作为发送地址/数据的特征位，而数据通信的校验采用方阵码校验法。通信波特率由单片机决定。通信方式多为主从式，即 PC 为主机，单片机为从机。通信过程为应答式，即由主机向从机发送一个命令，从机收到命令后给出一个应答信号便完成一次通信。在组态王中单片机通信协议命令帧格式如下所示：

① 上位机发送的地址命令帧格式为：（此时检验位为 1）

ENQ	Sta	EOT	CRC

② 下位机应答的地址命令帧格式为：（此时检验位为 0）

ACK	Sts	ETX	CRC

③ 上位机读数据命令帧格式：（此时检验位为 0）

ENQ	R	DataType	DataAddr	DataNum	EOT	CRC

④ 下位机应答读数据命令帧格式（正确）：（此时检验位为 0）

ACK	DataLong	Data…	ETX	CRC

⑤ 下位机应答读数据命令帧格式（错误）：（此时检验位为 0）

NAK	ErrorCode	ETX	CRC

⑥ 上位机写数据命令帧格式：（此时检验位为 0）

ENQ	W	DataType	DataAddr	Data…	EOT	CRC

⑦ 下位机应答写数据命令帧格式：（此时检验位为 0）

ACK	ErrorCode	ETX	CRC

其中，Sta：设备地址，占 1 字节；R：读标志，占 1 字节（0x52）；W：写标志，占 1 字节（0x57）；DataType：需要交换的数据类型，占 1 字节（1 代表字节，2 代表字，3 代表浮点型）；DataNum：要读取数据的字节数，占 1 字节；DataAddr：数据起始地址，占 2 字节（的字节在前，高字节在后）；Data：实际传输的数据（低字节在前，高字节在后）；DataLong：单片机返回 Data 的字节数，占 2 字节（低字节在前，高字节在后）；CRC：从第一个字节至 CRC 前的所有字节的异或值，占 1 字节；ErrorCode：应答代码，占 1 字节（0 表示写入数据正确，其他表示写入数据有误）。

按以上协议规定编写单片机的串行通信程序。在组态王中设定通信参数（包括数据位、停止位、波特率、校验方式）时和单片机一致即可。

8.7.2 单片机通信软件设计

根据以上分析，单片机采用工作方式 3 接收，且置 SM2 为 1，允许串行口中断。当 PC 与单片机通信时，PC 首先发送要与之通信的单片机地址码，以辨认目标从机。地址字节与数据字节可用第 9 位来区别，发送地址信息时第 9 位为 1，发送数据信息时第 9 位为 0。当 PC 发送地址命令帧时，所有单片机都引起中断，接收此地址码与自己的地址码相比较，当某一台单片机发现此地址码与自己的相符时，就清除 SM2(=0)，并向 PC 发地址应答信号，表示本单片机已经做好接收数据的准备，可接收 PC 发送来的数据。而其他从机则仍保持

SM2 = 1 状态，这些从机将对进入串行口的数据字节不作响应。这样就保证每次只有一台从机与 PC 进行数据传输。其通信流程图如图 8-17 所示。

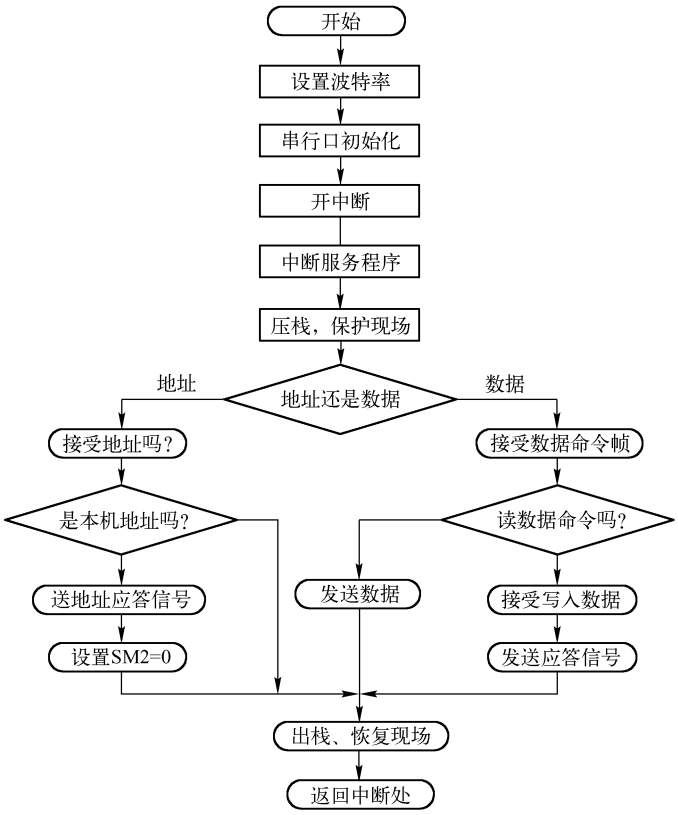


图 8-17　单片机远程通信

8.7.3　PC 上位机通信软件设计

由于通信方式为主从式，PC 是主机，故一切通信工作都由 PC 开始。为保证对下位机的实时监控，在组态王中，PC 对下位机采用定时轮询方式进行读、写操作，其程序框图如图 8-18 所示。

8.7.4　PC 上的通信设置

在组态王 6.51 中，首先用组态王中的“设备配置向导”将单片机定义成串口 I/O 设备，并在智能模块中选择单片机，安装其驱动程序。按通信协议设置串口参数。其次构建数据库，把要和单片机交换数据的变量定义成 I/O 变量，并按组态王中的规定设置其属性。最后建立必要的动画连接，即可进行运行和调试。

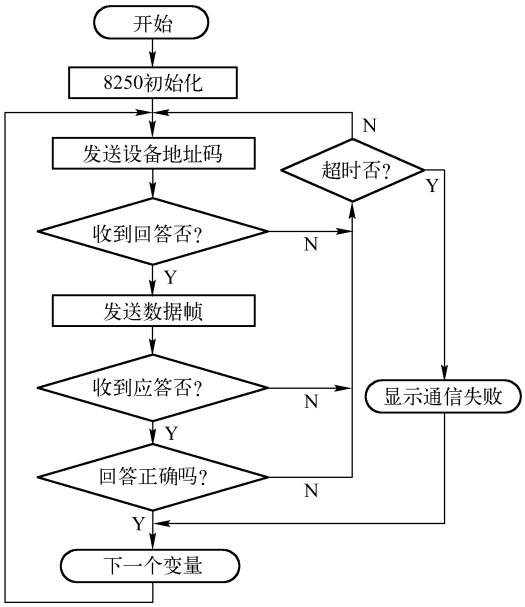


图 8-18　PC 远程通信流程图

8.8 组态王与 DTU 设备的通信

随着移动推出 GPRS 无线数据传输以来，GPRS 的通信速度快、通信费用低、组网灵活等优点，越来越被广大客户看好。GPRS 数传终端，具有 TCP/IP 协议转换功能，不需要用户提供 TCP/IP 的支持。可适用于所有带串口的终端设备，通过 GPRS 网络平台实现数据信息的无线和透明传输，为不具备 TCP/IP 协议处理的终端设备提供了 GPRS 通信的能力。目前，GPRS 数传终端已被广泛的应用于环保、水文水利、油田、电力，工业控制等各个领域，在数据的远程传输和监控方面得到了很好的应用。

组态王的 GPRS 服务程序支持通过 GPRS 数传终端（以下简称 GPRS DTU）与亚控公司开发过驱动的串口设备间的通信。本小节以北京嘉复欣公司的 GPRS DTU 设备为例介绍组态王与其进行通信的相关操作。

8.8.1 设备的连接与配置

嘉复欣公司 GPRS GF－2008 DTU，内嵌 TCP/IP 的 GPRS Modem，分为 IP 模块和 GPRS 模块两部分。有两种工作模式：一个是命令模式：一个是 Serial NET 模式。SOCKET 通信是在命令方式下的通信，期间需要用命令进行干预。而 Serial NET 模式使得连接的设备通过串行线进行 TCP 或 UDP Socket 连接。一旦初始化该模式的参数，进入该模式后，就可以直接按设定的参数进行数据传输。能广泛应用于工业、电力、交通、环境、监测监控、金融交易、零售服务等方面。

用户可以使用两种方式对嘉复欣公司 GPRS GF－2008 DTU 进行设置，一种是通过超级终端进行设置，另一种方式是通过嘉复欣公司提供的 ichipconfig. exe 软件工具进行配置。

1. 通过超级终端进行设置

1）选择正确的 COM 口，并进行端口设置，如图 8-19 所示。

图 8-19　串口设置对话框

2）设置 Serial NET 通信参数：

① 首先确认嘉复欣公司 GPRS GF－2008 DTU 处于配置状态。

键入 AT 回车，如果显示“OK”，说明嘉复欣公司 GPRS GF－2008 DTU 处于配置状态。

如果用户不能键入“AT”字符，说明嘉复欣公司 GPRS GF－2008 DTU 处于 Serial NET 模式，请按“＋”号键 3～5s，使嘉复欣公司 GPRS GF－2008 DTU 回到配置状态。

② 判断模块是否正常。

＞AT＋i ＜cr＞

I/OK

＞AT＜cr＞

OK

以上两条命令通过即可初步断定模块无故障。

③ 检查网络信号强度和 SIM 卡情况。

＞AT＋CSQ ＜cr＞

＋CSQ：＊＊，##

其中＊＊应在 10～31 之间，数值越大表明信号质量越好，##应为 99。否则应检查天线或 SIM 卡是否正确安装。

④ 设置 IP 地址及端口号命令。

AT＋iHSRV＝IP 地址 ：端口号

如 222. 131. 112. 173：5020

At＋ipars 把修改的参数保存下。

⑤ 键入：AT＋I！SNMD，使嘉复欣 GPRS GF－2008 DTU 进入 SerialNET 模式。

2. 通过 ichipconfig. exe 软件工具进行配置

打开 ichipconfig. exe 软件工具，选择高级设置，配置服务器地址和服务器端口，例如 222. 131. 112. 173：5020，如图 8-20 所示。

图中红色 1 字表示的参数项是透明传输模式下的串口波特率设置项，如默认设置为 5、8、N、1、0，其中参数 5 代表串口波特率，对应的串口波特率为 9600，可选内容为 2 对应波特率 1200、3 对应波特率 2400、4 对应波特率 4800、5 对应波特率 9600、6 对应波特率 19200、7 对应波特率 38400、8 对应波特率 57600、9 对应波特率 115200；8 代表数据，可选内容有 7 位或 8 位；N 代表校验位，可选内容有 N 无校验、E 偶校验、O 奇校验；1 代表停止位，可选内容有 1 位停止和 2 位停止；0 代表流控，可选内容有无流控、硬件流控和软件流控。

图中红色 2 字表示的参数项是数据中心的 IP 地址和端口号，前面空格为地址内容可填数据中心的 IP 地址（该地址必须是公网的真实 IP 地址）或者域名（该域名必须是能够被域名服务器解析的域名），后面的空格是对应的连接端口号。

图中红色 3 字表示参数项是 TCP 通信协议选项，可选内容有 TCP 和 UDP。

所有参数设置完成之后，您可以按 save 键保存已经设置好的参数内容，然后退出配置界面，回到主界面。

通过命令进入 Serial NET 模式：

At＋iup

At＋i！snmd

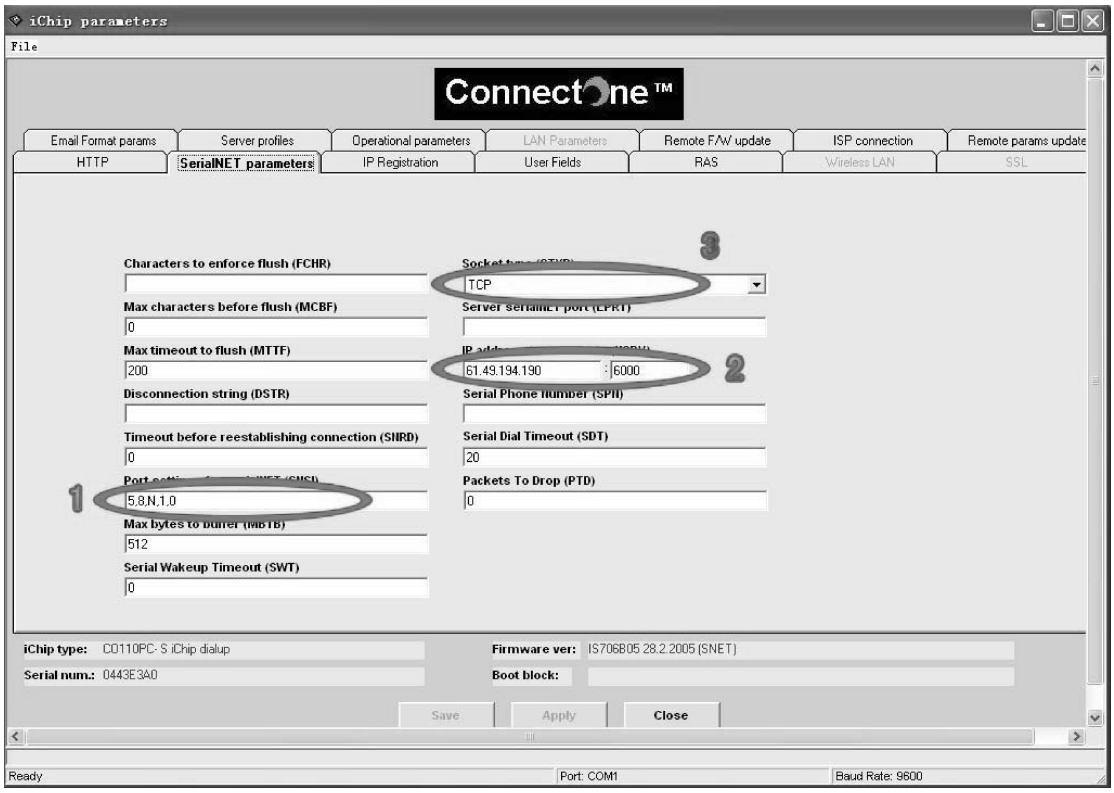


图 8-20　通过 ichipconfig. exe 软件工具进行配置

8.8.2　组态王的设置

首先要新建一个项目，定义项目名称，在组态王工程管理器里能看到建立完的项目如图 8-21 所示。

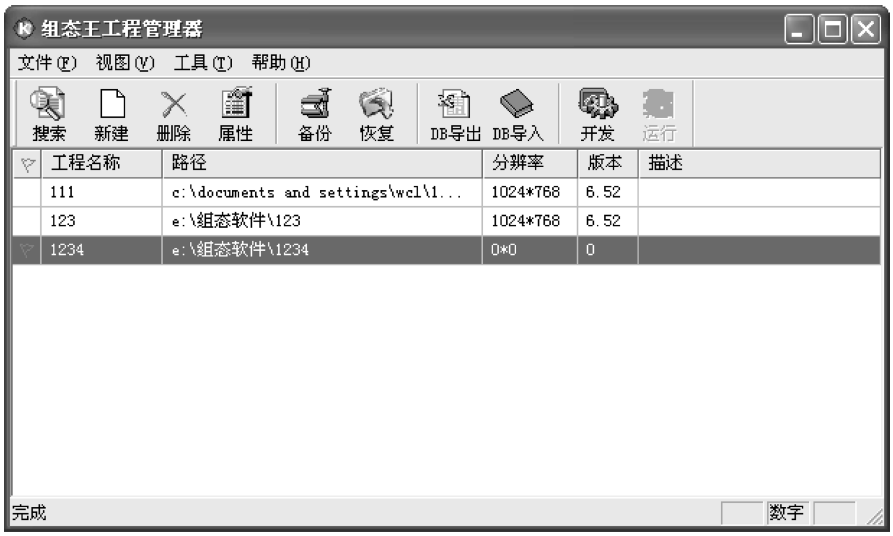


图 8-21　新建工程

组态王与连接用户数据采集终端设备的嘉复欣公司 GPRS GF－2008 DTU 之间的通信是通过虚拟串口实现的。

定义虚拟串口设备：

虚拟串口定义说明：

在一个虚拟串口上只能定义 1 个虚拟设备；

每个虚拟设备可以关联多个实际设备（设备的协议必须相同）；

与每个虚拟设备关联的实际设备地址不能重复；

工程中所有虚拟设备或实际设备逻辑名称不能重复；

不同虚拟设备的标识不能重复；

同一厂家的虚拟设备端口号应该一致。

下面我们就以永宏 PLC，配以嘉复欣公司 GPRS GF－2008 DTU 设备为例进行说明。

第一步：选择串口设备（实际设备，即用户数据采集终端设备），如图 8-22 所示。

图 8-22　新建设备

第二步：给实际设备即用户数据采集终端设备输入一个逻辑名称，如图 8-23 所示。

第三步：给用户数据采集终端设备选择一个虚拟串口，如图 8-24 所示。

第四步：在虚拟串口上定义嘉复欣公司 GPRS GF－2008 DTU 设备，如图 8-25 所示。

第五步：定义设备地址，即 GPRS 下挂的实际设备（如 PLC 的地址），如图 8－26 所示。

至此设备定义完毕，系统会生成两种设备的图标，虚拟串口设备（即 GPRS DTU 设备）和 GPRS DTU 设备下挂的实际设备，如图 8-27 所示。

如果已经定义了设备，在组态王中需要修改虚拟设备的相关信息，可以在组态王开发系统中双击虚拟设备的图标（图 8-27 中的“虚拟设备”），弹出如图 8-28 所示对话框进行修改。

图 8-23　为新建设备设定名称

图 8-24　为新建设备选择虚拟串口

图 8-25　在虚拟串口上定义 DTU 设备

图 8-26　为实际设备定义地址

1. 定义设备的通信参数

假如永宏 PLC 设备（DTU 下挂的实际设备）的通信参数是 9600、7、1、偶校验。那么要把 DTU 的通信参数也配置成 9600、7、1、偶校验。双击图 8-27 中的“KVCOM1”图标，可以对虚拟串口“通信超时”进行设置，建议设置成 15s 以上，如图 8-29 所示。

图 8-27　定义完成的 DTU 设备和实际设备

图 8-28　DTU 设备参数修改

图 8-29　配置 DTU 设备的通信参数

2. 组态王数据词典——I/O 变量定义

组态王数据词典的定义包括两个部分：一是定义 GPRS DTU 设备下挂的实际设备的数据词典，这部分与普通串口设备的数据词典定义相同；二是定义虚拟串口设备的数据词典，见表 8-1。

表 8-1　组态王虚拟串口设备寄存器列表

寄存器格式	寄存器范围	读写属性	数据类型	变量类型	寄存器含义
V_S	—	只读	SHORT	I/O 整数	虚拟设备（即 GPRS DTU 设备）的状态：0 表示没有连接到 GPRS 网络。1 表示已经连接到 GPRS 网络
V_C	—	读写	SHORT	I/O 整数	虚拟设备控制寄存器。0 表示停止虚拟设备工作。1 表示恢复虚拟设备工作

（1）组态王虚拟串口设备寄存器列表

表 8-1 中两个变量的定义的作用是组态王检测 DTU 的网络连接情况，可以不用连接终端设备。

（2）实际连接 PLC 的变量定义　在数据词典中新建一个 HR0 变量。变量类型选 I/O 整数，连接设备选新 I/O 设备，寄存器选 HR 后加个 0，数据类型选 USHORT，读写属性为只读，如图 8-30 所示。

图 8-30　定义连接 PLC 的变量

用同样方法再新建一个 HR1 变量。

3. 新建画面

新建一个画面，在工具箱中选择 T，输入文本 HR0，回车后再输入 HR1。把两个 HR0、HR1 分别用 Ctrl + C、Ctrl + V 复制、粘贴一下，拉开距离，如图 8-31 所示。

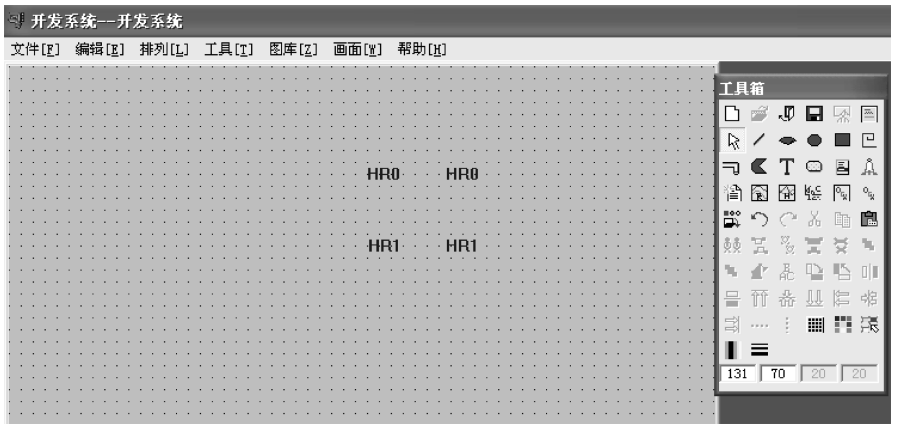


图 8-31　新建画面并设置

对 HR0 进行配置，双击即可进入配置，如图 8-32 所示。

图 8-32　动画连接对话框

HR0 要求配置模拟值输出和模拟值输入，单击模拟值输出，弹出如图 8-33 所示对话框。

在表达式中单击“?”按钮，选择所对应的变量，选中变量 HR0 双击即可。整数位数选择 6。完成后效果如图 8-34 所示。

模拟值输入连接设置如图 8-35 所示。

用同样方法对 HR1 进行配置。

图 8-33　模拟值输出对话框

图 8-34　模拟值输出对话框

图 8-35　模拟值输入对话框

4. 保存配置

配置完后选择全部存，选择切换到 View，打开刚才所建的画面，这时候 PLC 的数据已经传上来了，如图 8-36 所示。

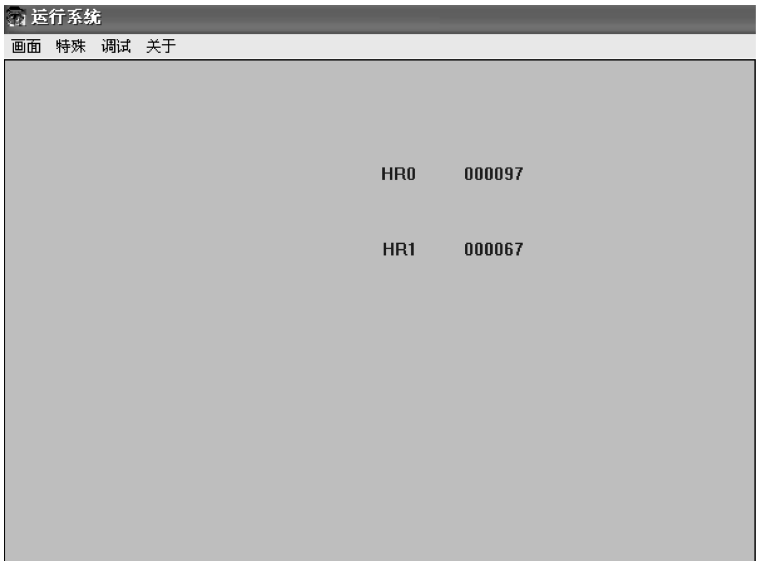


图 8-36　运行效果图

8.9 组态王与数据库连接

组态王 SQL 访问功能实现组态王和其他外部数据库（通过 ODBC 访问接口）之间的数据传输。它包括组态王的 SQL 访问管理器和相关的 SQL 函数。

SQL 访问管理器用来建立数据库字段和组态王变量之间的联系，包括“表格模板”和“记录体”两部分。通过表格模板在数据库表中建立相应的表格；通过记录体建立数据库字段和组态王之间的联系。同时允许组态王通过记录体直接操作数据库中的数据。

8.9.1 创建数据源及数据库

首先外建一个数据库，这里我们选用 Access 数据库（路径：d:\peixun，数据库名为：mydb.mdb）。

然后，用 Windows 控制面板中自带的 ODBC Data Sources（32bit）管理工具新建一个 Microsoft Access Driver（*.mdb）驱动的数据源，名为：mine，然后配置该数据源，指向刚才建立的 Access 数据库（即 mydb.mdb），如图 8-37 所示。

图 8-37 ODBC 数据源的建立

在建立了数据源之后，我们需要在组态王中创建表格模板。

1）在工程浏览器窗口左侧“工程目录显示区”中选择“SQL 访问管理器”中的“表格模板”选项，在右侧“目录内容显示区”中双击“新建”图标，弹出创建表格模板对话框，在对话框中建立三个字段，如图 8-38 所示。

2）单击“确认”按钮，完成表格模板的创建。建立表格模板的目的是定义数据库格式，在后面用到 SQLCreatTable( ) 函数时，以此格式在 Access 数据库中自动建立表格。

接下来我们还要创建记录体。

1）在工程浏览器窗口左侧“工程目录显示区”中选择“SQL 访问管理器”中的“记录体”选项，在右侧“目录内容显示区”中双击“新建”图标，弹出创建记录体对话框，对话框设置如图 8-39 所示。

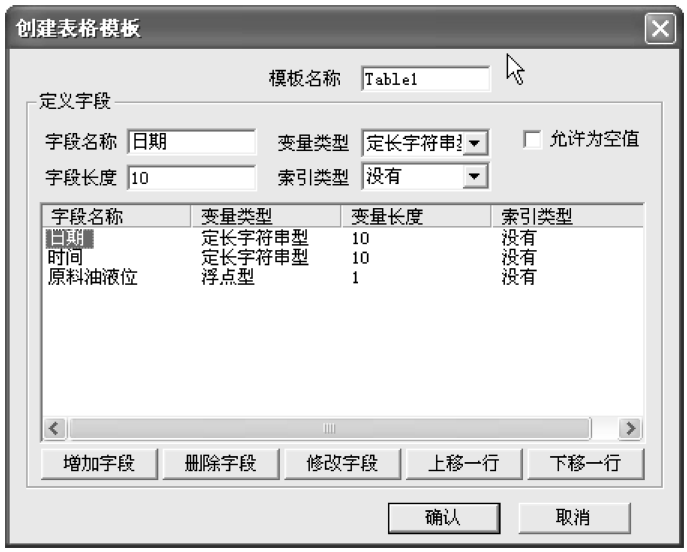


图 8-38　创建表格模板对话框

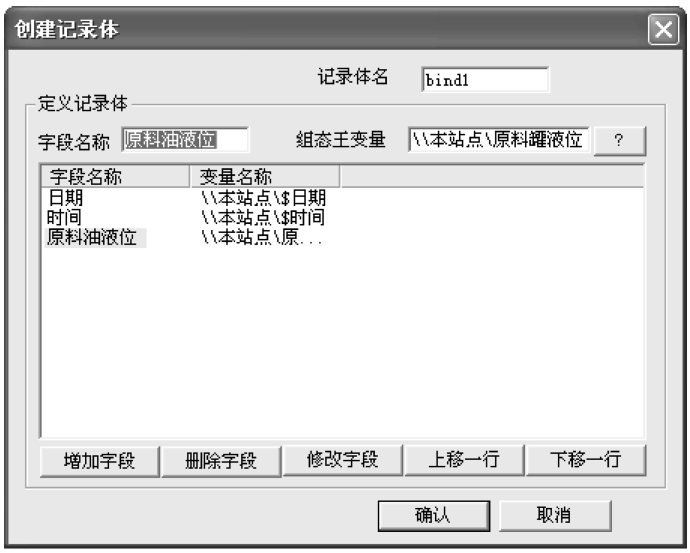


图 8-39　创建记录体对话框

记录体中定义了 Access 数据库表格字段与组态王变量之间的对应关系，对应关系如下所示：

Access 数据库表格字	组态王变量
日期字段	\\本站点\$日期
时间字段	\\本站点\$时间
原料油液位	\\本站点\原料灌液位

即将组态王中\\本站点\$日期变量值写到 Access 数据库表格日期字段中；将\\本站点\$ 时间变量值写到 Access 数据库表格时间字段中；将\\本站点\原料灌液位写到 Access

数据库表格原料灌液位值字段中。

2）单击“确认”按钮完成记录体的创建。

注：记录体中的字段名称必须与表格模板中的字段名称保持一致，记录体中字段对应的变量数据类型必须和表格模板中相同字段对应的数据类型相同。

8.9.2 对数据库的操作

1. 连接数据库

1）在工程浏览器窗口的数据词典中定义一个内存整型变量，变量名为 DeviceID；变量类型：内存整型。

2）新建一画面，名称为数据库操作画面。

3）选择工具箱中的T工具，在画面上输入文字“数据库操作”。

4）在画面中添加一按钮，按钮文本为“数据库连接”。

5）在按钮的弹起事件中输入命令语言，如图 8-40 所示。

图 8-40 数据库连接命令语言

上述命令语言的作用是使组态王与 mine 数据源建立了连接（即与 mydb. mdb 数据库建立了连接）。

在实际工程中将此命令写入：工程浏览器→命令语言→应用程序命令语言→启动时中，即系统开始运行就连接到数据库上。

2. 创建数据库表格

1）在数据库操作画面中添加一按钮，按钮文本为“创建数据库表格”。

2）在按钮的弹起事件中输入命令语言，如图 8-41 所示。

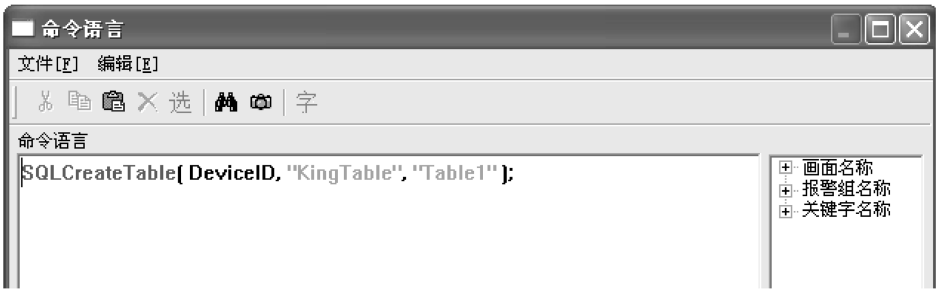


图 8-41 创建数据库表格命令语言

上述命令语言的作用是以表格模板“Table1”的格式在数据库中建立名为“KingTable”的表格。在生成的 KingTable 表格中，将生成三个字段，字段名称分别为日期、时间、原料油液位，每个字段的变量类型、变量长度及索引类型与表格模板“Table1”中的定义一致。

此命令语言只需执行一次即可，如果表格模板有改动，需要用户先将数据库中的表格删除才能重新创建。在实际工程中将此命令写入：工程浏览器→命令语言→应用程序命令语言→启动时中，即系统开始运行就建立数据库表格。

3. 插入记录

1）在数据库操作画面中添加一按钮，按钮文本为“插入记录”。

2）在按钮的弹起事件中输入命令语言，如图 8-42 所示。

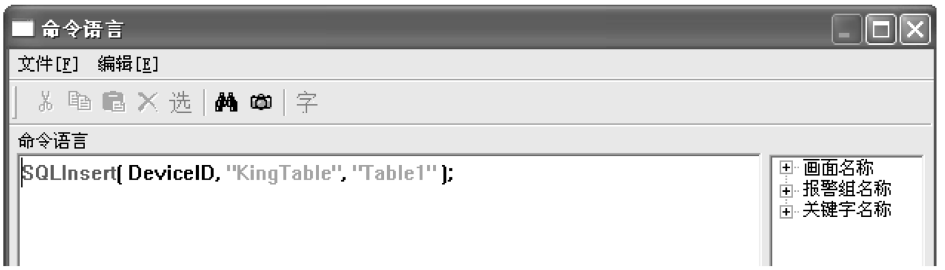


图 8-42　插入记录命令语言

上述命令语言的作用是在表格 KingTable 中插入一个新的记录。按下此按钮后，组态王会将 bind1 中关联的组态王变量的当前值插入到 Access 数据库表格“KingTable”中，从而生成一条记录，从而达到了将组态王数据写到外部数据库中的目的。

4. 查询记录

用户如果需要将数据库中的数据调入组态王来显示，需要另外建立一个记录体，此记录体的字段名称要和数据库表格中的字段名称一致，连接的变量与数据库中字段的类型一致，操作过程如下：

1）在工程浏览器窗口的数据词典中定义三个内存变量，用于存储查询到的值：

① 变量名：记录日期；变量类型：内存字符串；初始值：空。

② 变量名：记录时间；变量类型：内存字符串；初始值：空。

③ 变量名：原料油液位返回值；变量类型：内存实型；初始值：0。

2）新建一画面，名称为“数据库查询画面”。

3）选择工具箱中的 T 工具，在画面上输入文字“数据库查询”。

4）在画面上添加三个文本框，在文本框的“字符串输出”、“模拟量值输出”动画中分别连接变量\\本站点\记录日期、\\本站点\记录时间、\\本站点\原料油液位返回值，用来显示查询出来的结果。

5）在工程浏览窗口中定义一个记录体，记录体窗口属性设置如图 8-43 所示。

6）在画面中添加一按钮，按钮文本为“得到选择集”。

7）在按钮的弹起事件中输入命令语言，如图 8-44 所示。

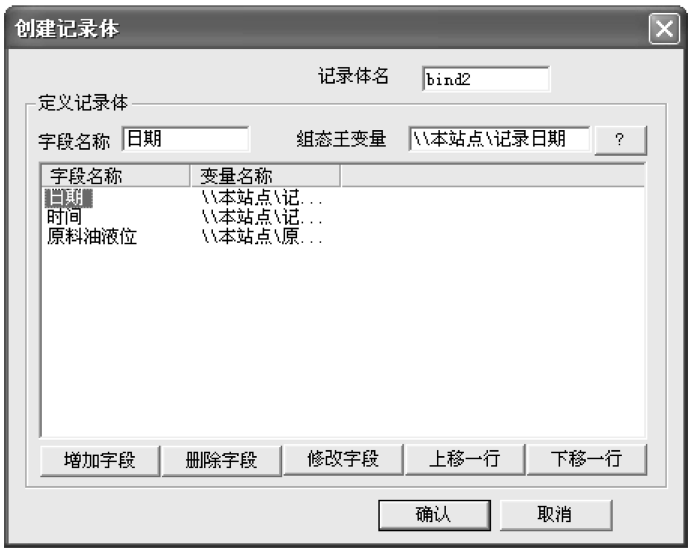


图 8-43　记录体属性设置对话框

图 8-44　记录查询命令语言对话框

此命令语言的作用是：以记录体 bind2 中定义的格式返回 KingTable 表格中第一条数据记录。

8）单击“文件”菜单中的“全部存”命令，保存您所作的设置。

9）单击“文件”菜单中的“切换到 View”命令，进入运行系统。运行此画面，单击“得到选择集”按钮，数据库中的数据记录显示在文本框中，如图 8-45 所示。

图 8-45　数据库记录查询

10）在画面上添加 4 个按钮，按钮属性设置如下：

① 按钮文本：第一条记录“弹起时”动画连接：SQLFirst(DeviceID)；

② 按钮文本：下一条记录“弹起时”动画连接：SQLNext(DeviceID)；

③ 按钮文本：上一条记录“弹起时”动画连接：SQLPrev(DeviceID)；

④ 按钮文本：最后一条记录“弹起时”动画连接：SQLLast(DeviceID)。

上述命令语言的作用分别为查询数据中第一条记录、下一条记录、上一条记录和最后一

条记录，从而达到了数据查询的目的。

5. 断开连接

1）在“数据库操作画面”中添加一按钮，按钮文本为“断开数据库连接”。

2）在按钮的弹起事件中输入命令语言，如图 8-46 所示。

图 8-46 断开数据库连接命令语言

在实际工程中将此命令写入：工程浏览器→命令语言→应用程序命令语言→退出时中，即系统退出后断开与数据库的连接。

8.9.3 数据库查询控件

利用组态王提供的 KVDBGrid Class 控件可方便地实现数据库查询工作，操作过程如下：

1）单击工具箱中的“插入通用控件”工具或选择菜单命令“编辑\插入通用控件”，则弹出控件对话框。在控件对话框内选择“KVDBGrid Class”选项，如图 8-47 所示。

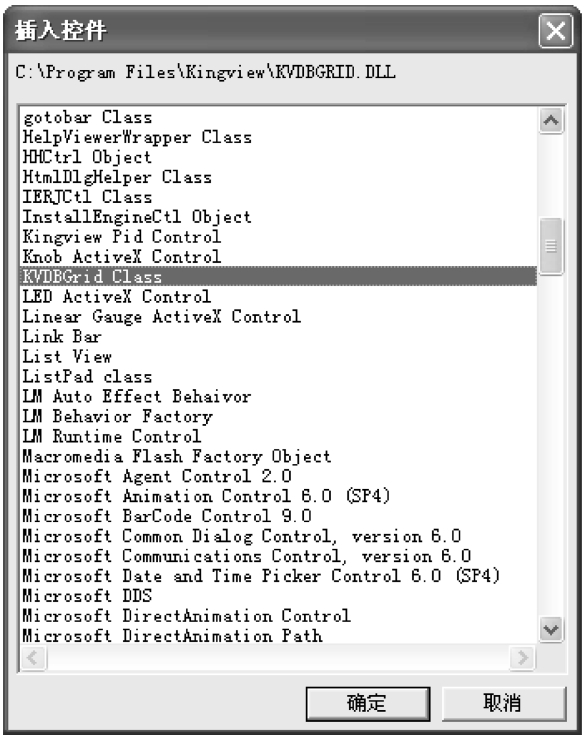


图 8-47 通用控件对话框

2）在画面中添加一 KVDBGrid Class 控件，选中并双击控件，在弹出的动画连接属性对话框中设置控件名称为 grid1。

3）选中控件并单击鼠标右键，在弹出的下拉菜单中执行“控件属性”命令，弹出属性对话框，如图 8-48 所示。

图 8-48　控件属性对话框

单击窗口中的“浏览”按钮，在弹出的数据源选择对话框中选择前面创建的 mine 数据源，此时与此数据源连接的数据库中所有的表格显示在“表名称”的下拉框中，从中选择欲查询的数据库表格（在这里我们选择前面建立的 KingTable 表格），此表格中建立的所有字段将显示在“有效字段”中，利用添加(A)和删除(E)选择您所查询的字段名称并可通过“标题”和“格式”编辑框对字段进行编辑。

4）设置完毕后关闭此对话框，利用按钮的命令语言实现数据库查询和打印工作，设置如下：

按钮一：查询全部记录：grid1. FetchData( )；grid1. FetchEnd( )；

按钮二：条件查询：

```
        long aa;
aa = grid1. QueryDialog( );
        if ( aa == 1 )
{ grid1. FetchData( ); grid1. FetchEnd( );
        }
```

按钮三：打印：grid1. Print( )；

按钮四：保存（将控件查询出的数据以 CSV 为后缀名，保存到指定路径）：grid1. SaveToCSV("d:\peixun \ data. CSV") ；

5）系统运行画面如图 8-49 所示。

练一练：

1）阅读组态王软件帮助中的 KVDBGrid Class 控件帮助说明。

2）制作一个动态数据查询系统来查询数据记录。

图 8–49　数据库操作运行画面

第 9 章　PLC 与组态王应用实例

9.1　三层电梯控制系统

随着工业自动化技术的发展，人们对自动化监控系统的要求越来越高。而如何用最简便的方法设计开发一个实时监控系统成了广大工程技术人员最关心的问题。电梯作为高层建筑物的重要交通工具与人们的工作和生活日益紧密联系，以它作为控制对象来设计实时监控系统无疑具有典型性。本节首先设计了基于 FX_{2N}PLC 的电梯控制系统，在此基础上，用组态王软件设计开发一个全新的电梯监控系统，为实际工程应用奠定了良好的基础。

9.1.1　基于 FX_{2N} –48MR 的电梯控制系统

1. 控制要求

为了与实际应用相匹配，这里所设计的电梯控制系统要求与实际系统一致。电梯在初始状态时，各层显示器都被初始化，各层呼叫灯均不亮；电梯在运行过程中，按下某层呼叫按钮后，该层呼叫灯亮，电梯响应该层呼叫，电梯上行或下行直至该层。各楼层指示随电梯移动而改变。在电梯运行过程中，支持其他呼叫。电梯在到达指定楼层完成开门和关门的动作后，电梯会继续待命，直至新命令产生。详细控制要求见表 9–1。

表 9–1　电梯控制要求

序号	输　入		输　出	
	原停层	呼叫层	运行方向	运行结果
1	1	3	升	上升到 3 层停同时开门，4s 后关门
2	2	3	升	上升到 3 层停同时开门，4s 后关门
3	3	3	停	呼叫无效
4	1	2	升	上升到 2 层停同时开门，4s 后关门
5	2	2	停	呼叫无效
6	3	2	降	下降到 2 层停同时开门，4s 后关门
7	1	1	停	呼叫无效
8	2	1	降	下降到 1 层停同时开门，4s 后关门
9	3	1	降	下降到 1 层停同时开门，4s 后关门
10	1	2，3	升	先升到 2 层，开门、关门结束，再升到 3 层停
11	2	先 1 后 3	先降后升	先下降到 1 层停，开门、关门结束，再升到 3 层停
12	2	先 3 后 1	先升后降	先上升到 3 层停，开门、关门结束，再降到 1 层停
13	3	2，1	降	先降到 2 层，开门、关门结束，再降到 1 层停
14	任意	报警	任意	报警指示灯亮及蜂鸣器响
15	在轿厢运行期间不能开门			
16	轿厢不关门不允许运行			

2. I/O 分配

输入、输出点的分配分别见表 9-2 和表 9-3。

表 9-2　输入点分配

序　　号	名　　称	输 入 点
0	1 层内选按钮 S1	X000
1	2 层内选按钮 S2	X001
2	3 层内选按钮 S3	X002
3	1 层上呼按钮 U1	X003
4	2 层上呼按钮 U2	X004
5	2 层下呼按钮 D2	X005
6	3 层下呼按钮 D3	X006
7	开门按钮	X007
8	关门按钮	X010
9	报警按钮	X012
10	1 层行程开关 SQ1	X013
11	2 层行程开关 SQ2	X014
12	3 层行程开关 SQ3	X015
13	关门到位行程开关	X016
14	开门到位行程开关	X017

表 9-3　输出点分配

序　　号	名　　称	输 出 点
0	电梯上升 A	Y000
1	电梯下降 B	Y001
2	1 层指示 L1	Y002
3	2 层指示 L2	Y003
4	3 层指示 L3	Y004
5	开门指示 L4	Y005
6	关门指示 L5	Y006
7	报警指示 L6	Y007
8	1 层内选回应指示 L7	Y010
9	2 层内选回应指示 L8	Y011
10	3 层内选回应指示 L9	Y012
11	1 层上呼回应指示 L10	Y013
12	2 层上呼回应指示 L11	Y014
13	2 层下呼回应指示 L12	Y015
14	3 层下呼回应指示 L13	Y016
15	开门回应指示 L14	Y017
16	关门回应指示 L15	Y020

按照 I/O 点的分配和项目描述的控制要求，考虑到余量，选择 PLC 为 FX_{2N} -48MR。

3. PLC 编程

根据 3 层电梯的控制要求和逻辑关系即可进行 PLC 编程，详细梯形图如图 9-1 所示。

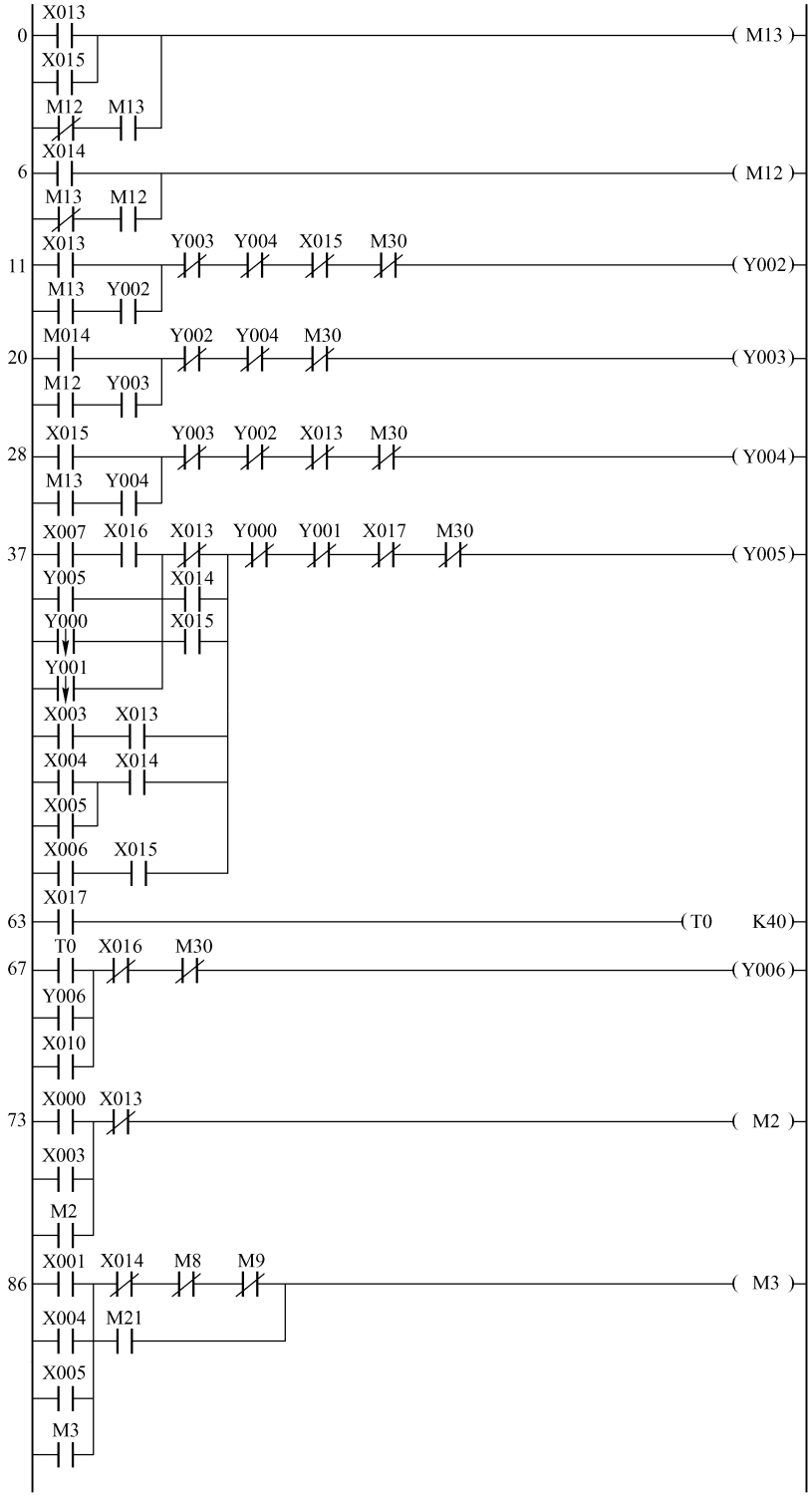


图 9-1　3 层电梯 PLC 控制梯形图

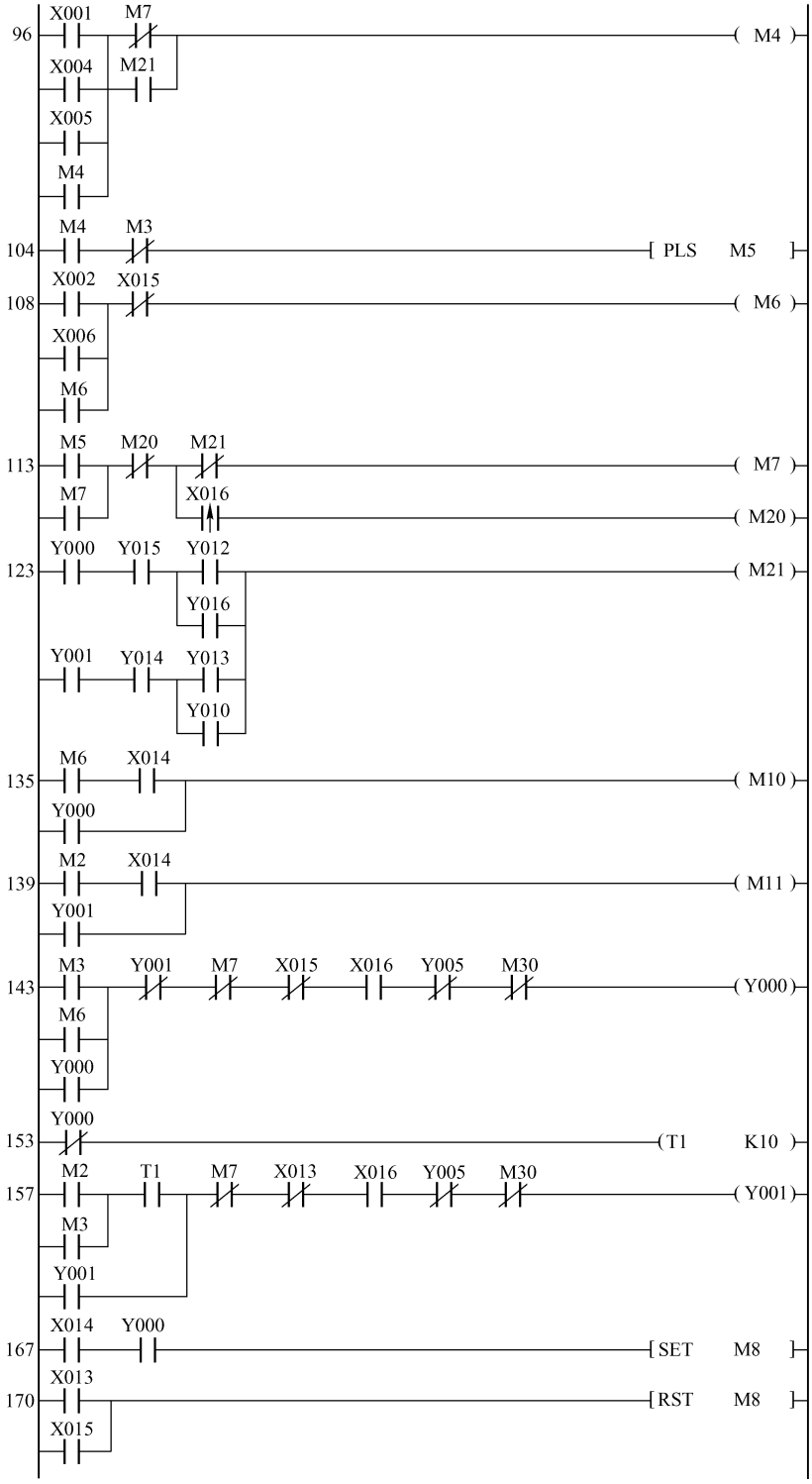


图 9-1　3 层电梯 PLC 控制梯形图（续）

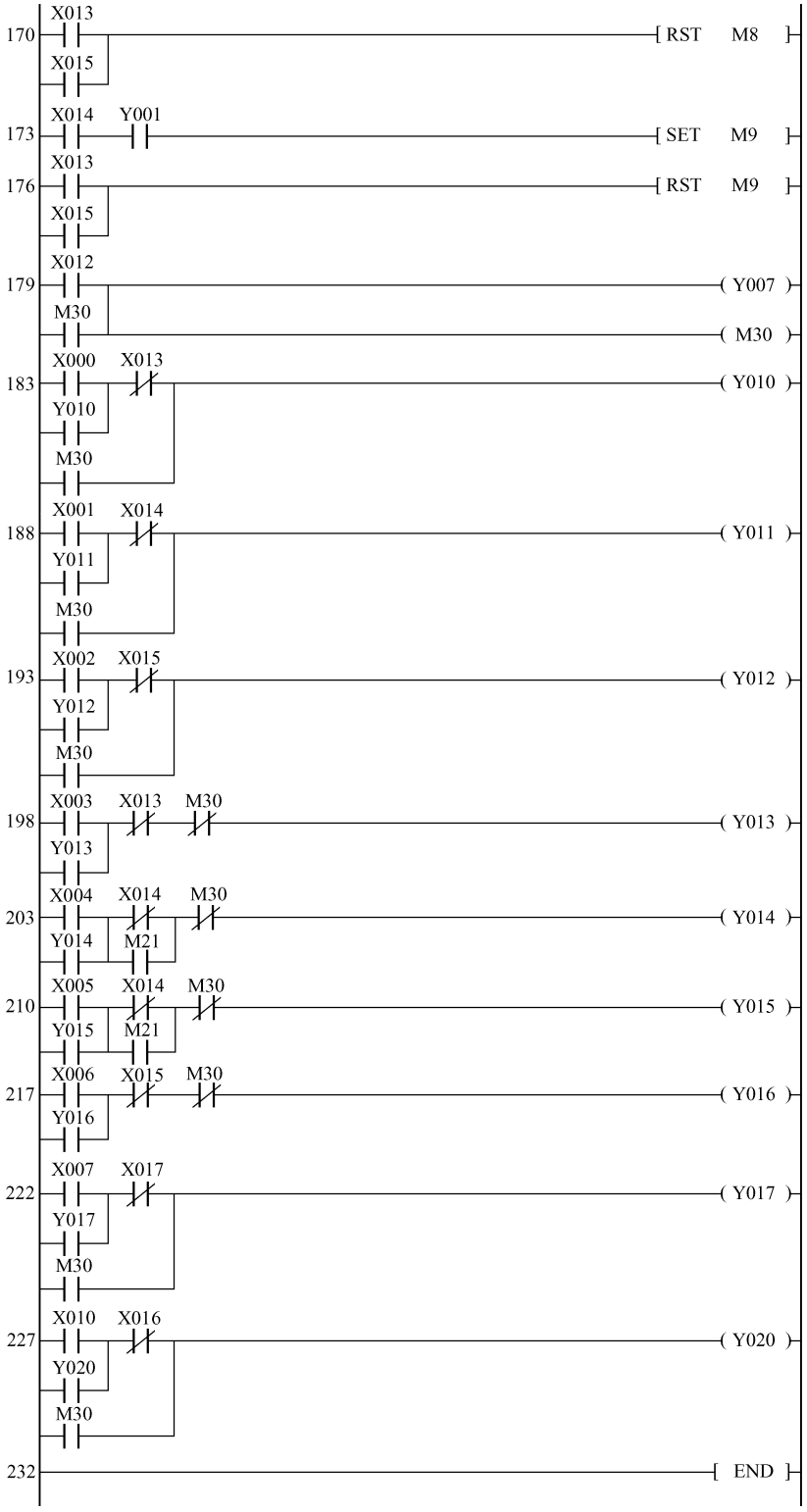


图 9-1　3 层电梯 PLC 控制梯形图（续）

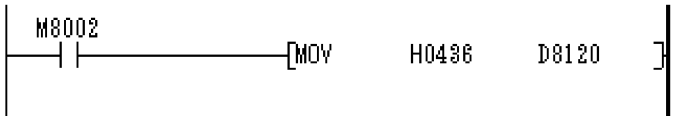
9.1.2 基于组态王的电梯监控系统

1. 准备工作

在利用组态王设计电梯监控系统时，准备工作包括新工程项目的建立与逻辑设备的创建。运行“组态王6.5”，在弹出的工程管理器窗口双击“新建”图标，按照新建工程配置向导建立一个名为“电梯监控系统”的新工程，并把此工程设为组态王当前工程。在创建工程项目后，在工程管理器中双击该工程就会进入工程浏览器界面。此时我们要做的是组态王中定义与所用PLC相对应的逻辑设备。在组态王中，把每台与之通信的设备（包括PLC、智能仪表、智能模块及其他工控设备）看作是外部设备，它内置了大量的设备驱动作为组态王与外部设备的通信接口。在开发过程只需根据“设备配置向导”的提示一步步进行，选择与所用设备相对应的逻辑设备即可完成外部设备的定义。

2. 组态王与PLC的通信设置

在完成逻辑PLC的定义之后，要想实现PLC与组态王之间的通信，还必须进行通信设置。PLC与上位计算机通信时，可以选用通信模块（如232BD、482BD等）与上位机通信，也可以在不添加扩展卡的情况下使用编程口和计算机进行通信，本书采用后者。进行通信设置时，组态王与PLC都需要进行设置，且两者的设置要一致。在组态王中，双击“COM1”图标，弹出“设置串口——COM1”对话框。“通信超时”为默认值，除非特殊说明，一般不需要修改。“通信方式”是指计算机一侧串口的通信方式，一般计算机一侧都为RS-232。在“通信参数”栏中，选择设备对应的波特率为9600、数据位为7、校验类型为偶校验、停止位为1。组态王一侧的通信方式确定之后，PLC也要进行相应设置。PLC的参数设置需要通过D8120寄存器来进行设置。通过查阅资料，本书上述通信设置对应的控制字为H0436。在PLC编程软件中编写如下语句：



选择工具菜单栏中的PLC→Transfers→write后，单击“OK”，即可完成PLC一侧通信参数的设置。

3. 组态王数据库构造

在组态王中，变量的集合形象地称为“数据词典”，数据词典记录了所有用户可使用的数据变量的详细信息。组态王中的变量可以分为基本类型和特殊类型两大类，基本类型的变量又分为内存变量和I/O变量两种。

在电梯控制系统中，所有的呼梯按钮、平层开关、楼层指示灯、运行指示灯都是开关量，只有0、1两种取值，并且它们都是通过驱动程序采集到的，所以这些变量类型都定义为I/O离散变量。在工程浏览器中，选择“数据库\数据词典”，双击“新建图标”，弹出“变量属性”对话框，定义相应的变量属性。按照表10-2和表10-3的I/O分配表定义相应的寄存器，并把连接设备设为“设备配置向导”上设置的新I/O设备。在定义变量属性时要注意的是，寄存器X定义为“只读”时，表示组态王只能从PLC上采集数据，不能向PLC写入数据，而定义为“读写”时，表示组态王和PLC之间为双向的，可以互相交换数据；寄存器Y只能定义为“只读”属性。

定义好的变量属性见表 9–4。

表 9–4　组态变量表

变 量 名	变 量 类 型	寄 存 器 名
1 层上呼按钮	I/O 离散	X3
2 层上呼按钮	离散	X4
2 层下呼按钮	离散	X5
3 层下呼按钮	离散	X6
1 层内选按钮	离散	X0
2 层内选按钮	离散	X1
3 层内选按钮	离散	X2
开门按钮	离散	X7
关门按钮	离散	X10
报警按钮	离散	X12
门的开度	内存整型	—
升降程度	内存整型	—
1 层行程开关	离散	X13
2 层行程开关	离散	X14
3 层行程开关	离散	X15
1 层指示灯	离散	Y2
2 层指示灯	离散	Y3
3 层指示灯	离散	Y4
报警指示	离散	Y7
电梯上升	离散	Y0
电梯下降	离散	Y1
电梯开门	离散	Y5
电梯关门	离散	Y6
开门到位	离散	X16
关门到位	离散	X17
1 层内选回应	离散	Y10
2 层内选回应	离散	Y11
3 层内选回应	离散	Y12
1 层上呼回应	离散	Y13
2 层上呼回应	离散	Y14
2 层下呼回应	离散	Y15
3 层下呼回应	离散	Y16
开门按钮回应	离散	Y17
关门按钮回应	离散	Y20

4. 监控画面制作

图形画面用于模拟实际的工业现场，在系统运行时，通过画面我们就可以监视电梯的运行状态。在组态王中设计画面时，主要利用各种绘图工具和图库管理器完成。本系统中电梯监控画面主要由电梯井道、电梯轿厢、厢内呼梯按钮、轿厢高度指示游标、运行方向指示等图素组成。整体画面如图 9–2 所示。

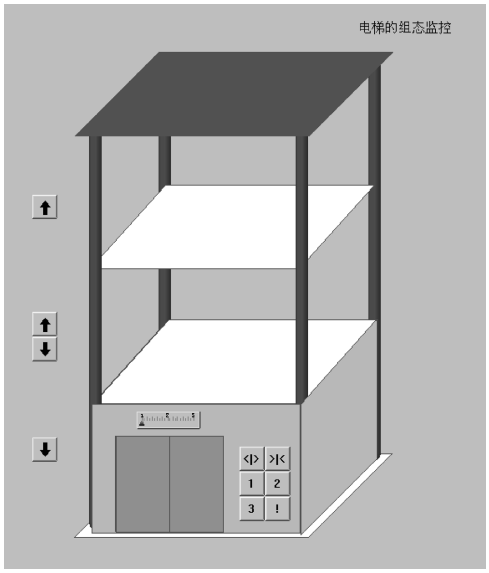


图 9–2　电梯模拟画面

5. 动画连接

所谓“动画连接”就是建立画面的图素与数据库变量的对应关系。本系统中要进行动画连接的图素是电梯轿厢、厢内呼梯按钮、轿厢高度指示、运行方向指示等。在进行动画连接时，除了进行必要的变量关联外，有时还需要进行一定的脚本编程。如在对电梯轿厢进行动画连接时，为了提高轿厢的仿真度，我们进行了如下脚本编程：

{事件描述：\\本站点\电梯上升 ==1

存在时：\\本站点\升降程度 = \\本站点\升降程度 +5；

每 55 ms}

{事件描述：\\本站点\电梯下降 ==1

存在时：\\本站点\升降程度 = \\本站点\升降程度 −5；

每 55 ms}

{事件描述：\\本站点\开门 ==1&&\\本站点\门的开度 <100

存在时：\\本站点\门的开度 = \\本站点\门的开度 +25；

每 55 ms}

{事件描述：\\本站点\关门 ==1&&\\本站点\门的开度 >0

存在时：\\本站点\门的开度 = \\本站点\门的开度 −25；

每 55 ms}

9.1.3 系统整体调试与运行

在完成上述各步骤之后，即可启动组态王运行系统对电梯监控系统进行调试。在外部按下呼梯信号，观察电梯监控的结果（比如轿厢的移动、各个灯颜色的变化等）。在对系统调试过程中，如果出现异常情况，分析造成故障的原因，一般是以下几种：变量设置错误、对象参数设置错误、动画连接错误或命令语言错误。通过返回到开发系统做出相应的修改，然后再次运行观察。经过多次仿真测试，实验结果表明系统运行正常，动画效果良好。

9.2 交通信号灯监控系统

1. 控制要求与 PLC 编程

交通信号灯控制系统（即十字路口红、黄、绿交通信号灯的控制）要求如下：按下起动按钮，交通信号灯开始工作，东西方向绿灯亮 56s，同时南北方向红灯亮 60s，东西方向绿灯亮 56s 后，闪烁 2s，然后过渡到东西方向黄灯，黄灯亮 2s；之后东西方向红灯亮 60s，南北方向绿灯亮 56s 后，闪烁 2s，随之黄灯亮 2s 后灭……

I/O 分配如下：

输入		输出			
起动	X0	东西绿灯	Y1	南北红灯	Y5
停止	X1	东西黄灯	Y2	南北绿灯	Y6
		东西红灯	Y3	南北黄灯	Y7

部分控制程序如图 9-3 所示。

2. PLC 与组态王的通信

PLC 与上位计算机的通信可以利用高级语言编程来实现，但是用户必须熟悉互连的 PLC 及 PLC 网络采用的通信协议，严格地按照通信协议规定为计算机编写通信程序，其对用户要求较高，而采用工控组态软件实现 PLC 与上位计算机之间的通信，则相对简单，因为工控组态软件中一般都提供了相关设备的通信驱动程序，例如西门子公司的 S7 系列 PLC 与工控组态软件 WinCC 之间可进行连接，实现 PLC 与上位计算机之间的通信。

下面介绍组态王 6.0 与 FX_{0N} -60MR PLC 之间通信的实现步骤。

FX_{0N} -60MR PLC 采用 RS-232 或 RS-422 进行通信，占用计算机的一个串行口。在不添加扩展卡的情况下，可以使用编程口和计算机进行通信：

1）设备连接：利用 PLC 与计算机专用的 F2-232CAB 型 RS-232C 电缆，将 PLC 通过编程口与上位计算机串口（COM 口）连接，进行串行通信。串行通信方式使用“组态王计算机”的串口，I/O 设备通过 RS-232 串行通信电缆连接到“组态王计算机”的串口。

2）设备配置：在组态王工程浏览器的工程目录显示区，点击“设备”项下 PLC 与上位计算机所连串口（COM 口），进行参数设置。

FX 系列 PLC 编程口的通信 COM 口参数设置，见表 9-5：

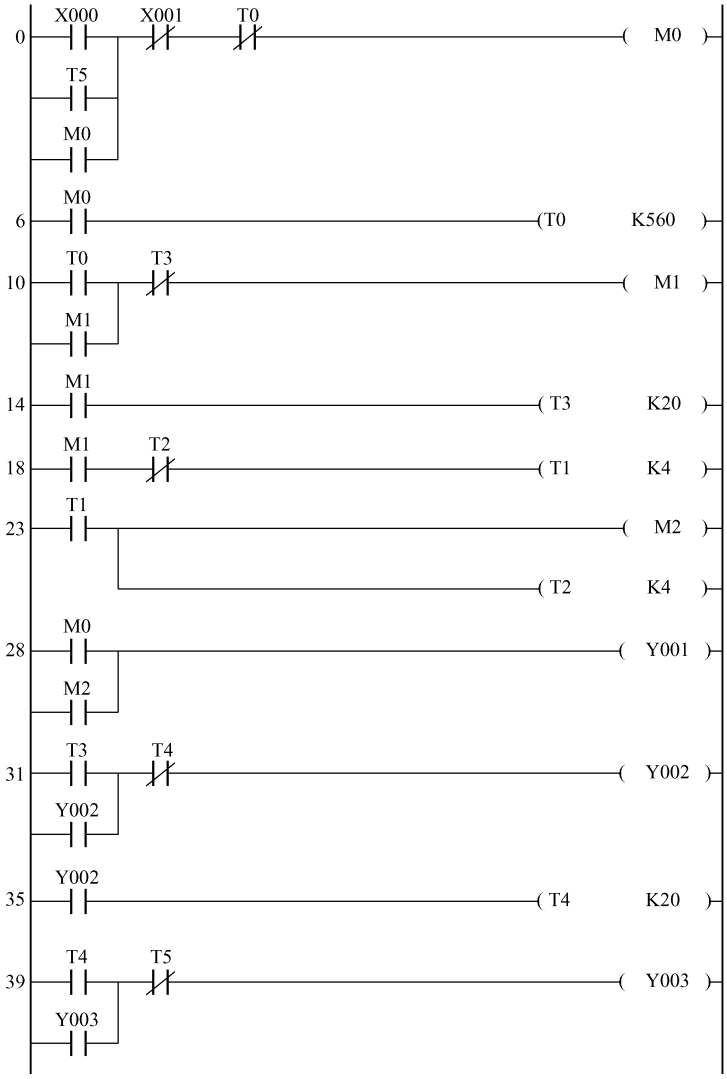


图 9-3　部分控制程序

表 9-5　通信 COM 口参数设置

波特率	数据位长度	停止位长度	奇偶校验位	通信方式
9600	7	1	偶校验	RS－232

然后在组态王浏览器目录内容显示区内双击所设 COM 口对应的“新建”图标，会弹出“设备配置向导”对话框。在此对话框中完成与组态王通信的设备的设置，各属性见表 9-6。

表 9-6　通信 COM 口参数设置

生产厂家	设备名称	通信描述
三菱	FX 系列 PLC	编程口

利用设备配置向导就可以完成串行通信方式的 I/O 设备安装，安装过程简单、方便。在

配置过程中，用户需选择 I/O 设备的生产厂家、设备型号、连接方式，为设备指定一个逻辑设备名，设定设备地址（FX 系列 PLC 在使用编程口进行通信时，不需要设备地址）。

3. 构造数据库

数据库是“组态王”软件的核心部分，在工程管理器中，选择“数据库\数据词典”，双击“新建图标”，弹出“变量属性”对话框。定义 FX_{0N}－60MR PLC 相应寄存器（见表 9-7）。

表 9-7　FX_{0N}－60MR PLC 相应寄存器定义

寄存器名称	寄存器格式	数据类型	变量类型	取值范围
输入寄存器	X*ddo*（位格式）	BIT	I/O 离散	0～43
输出寄存器	Y*ddo*（位格式）	BIT	I/O 离散	0～27
辅助寄存器	M*dddd*（位格式）	BIT	I/O 离散	0～8255
状态寄存器	S*ddd*（位格式）	BIT	I/O 离散	0～999
定时器触点	T*ddd*	BIT	I/O 离散	0～1023
计数器触点	C*ddd*	BIT	I/O 离散	0～1023

斜体字 *dddo*、*dddd*、*ddd* 等表示格式中可变部分，d 表示十进制数，o 表示八进制数，变化范围列于取值范围中。组态王按照寄存器名称来读取下位机相应的数据。组态王中定义的寄存器与下位机所有的寄存器相对应。如定义非法寄存器，将不被承认。如定义的寄存器在所用的下位机具体型号中不存在，将读不出数据。

4. 设计图形界面并建立动画连接

在组态王“画面”上创建十字路口红、黄、绿交通信号灯的控制示意图，如图 9-4 所示，建立起动和停止按钮，并将各个控制信号灯及起动和停止按钮与所建立相应变量关联，进行动画连接。

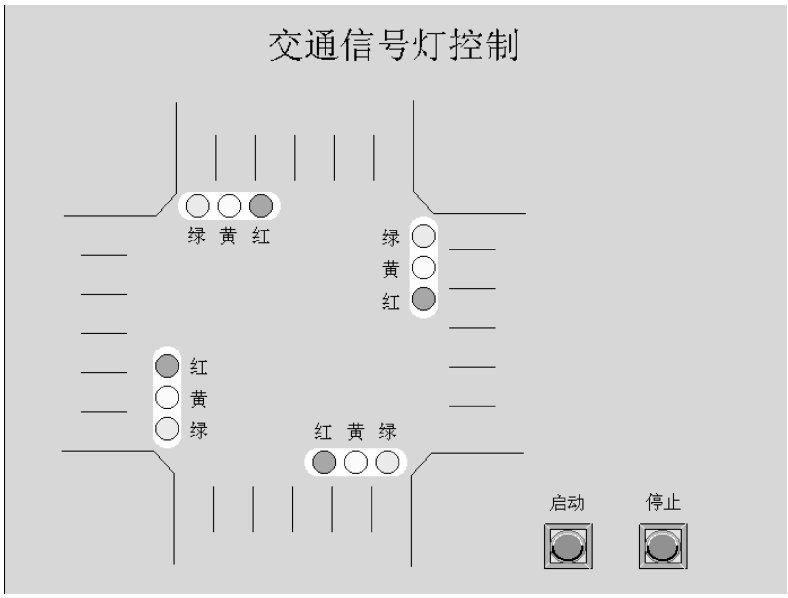


图 9-4　交通信号灯监控界面

5. 系统运行

启动组态王运行系统 TouchView，运行交通信号灯的控制。将 PLC 开关指向“RUN”状态，按下启动按钮，观察交通信号灯系统的控制结果。实验结果表明，系统运行正常，动画效果良好。

9.3 结语

本章以电梯和交通灯作为控制对象，设计开发了基于 PLC 和组态王的电梯监控系统及交通灯控制系统。通过实际运行表明，这种开发自动化监控系统的方法是完全可行的，而且具有很强的扩展性，同时也说明了组态王的出现结合了 PLC 和计算机的长处，降低了对用户的要求，大大缩短了设计周期，使得开发一个监控系统越来越容易，对于其他自动化监控系统的开发具有良好的借鉴意义。

○ ISBN 978-7-111-34776-7
○ 策划编辑：王　欢
○ 封面设计：赵颖喆

地址：北京市百万庄大街22号　邮政编码：100037
电话服务
社服务中心：(010)88361066
销售一部：(010)68326294
销售二部：(010)88379649
读者购书热线：(010)88379203
网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

定价：49.80元

ISBN 978-7-111-34776-7
9 787111 347767 >