摘 要

数控系统内部控制逻辑是由通用或内装式可编程控制器(PLC)来实现的。目前，PLC的编程方法主要有专用编程器和基于PC的软件编程器两种。华中世纪星系统采用的是“内装型”PLC，采用DOS环境下的C语言编程，这种编程方法没有采用Windows操作系统方便和快捷，而且使用C语言编程同使用传统的梯形图编程相比，对编程人员的技术水平要求高，不便推广使用。因此，本文研究了基于 Windows 操作环境的数控系统PLC编程及仿真软件，论文的主要工作有：

分析了数控系统PLC的需求，在此基础之上采用模块化设计方法，将PLC软件集成环境分解成项目管理模块、编辑模块、编译模块、模拟仿真模块等。同时，设计出PLC软件各模块间的数据联系，处理好各模块之间的数据、信息传递和功能驱动。

编辑器选用形象、直观的梯形图语言作为编程语言，采用面向对象的建模方法，建立PLC系统类层次结构和元件库模型对象，以图形输入方式来编程。

编译器以树结构为中介将梯形图和PLC指令联系起来，实现了从梯形图到PLC指令的转换。转换以梯级为单位，采用“正向深度优先扫描算法”来构造树结构，然后对树结构进行遍历，得到相应的PLC指令集。

通过对PLC工作原理的分析和研究，实现了PLC指令解释器算法的仿真，模拟了PLC运行过程。

关键字：数控系统，梯形图，模块化，树结构

Abstract

The internal control logic of Numerical Control (NC) system is implemented by general or built-in Programmable Logic Control。At present, there are mainly two methods of PLC programming, which are special programmer and software programmer based on PLC。Huazhong Century Star NC system adopts built-in PLC , and C language for programming under DOS environment. The method is less convenient and slower than under Windows operation system, furthermore, compared with using traditional ladder diagram for programming, it requires the person to be highly capable so it isn't easy to be popularized. The paper has done research on NC system PLC programming and simulation software based on Windows, the main tasks are as follows::

Adopt building block design based on the analysis of the demand of NC system PLC. The software consists of modules which are item management, edition, compilation, and analog simulation, etc. Simultaneously, data connection between the modules of this platform is designed, which deals with data and information transmission and drive between them.

Editor adopts visual ladder diagram language for programming and object-oriented modeling method to build class layer structure and model object of element library, and program by way of graphic input.

Compiler links ladder diagram with PLC instruction through structure tree to implement the transformation from ladder diagram to PLC instruction. The transformation adopts step unit and priority scanning algorithm of positive-direction depth to construct tree structure, then traverses tree to get corresponding PLC instruction collection.

Implement the emulation of PLC instruction explanation algorithm by analysis and study of PLC operating principle, and simulate PLC operation process.

Key words: NC system, ladder diagram, modularization, structure tree

独创性声明

本人声明所呈交的学位论文是我个人在导师的指导下进行的研究工作及取得的研究成果。尽我所知，除文中已标明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：韩伟

日期：2006 年 10 月 25 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在_____年解密后适用本授权书。

本论文属于

不保密☑

（请在以上方框内打“√”）

学位论文作者签名：韩伟　　指导教师签名：

日期：2006 年 10 月 25 日　　日期：2006 年 10 月 25 日

1 绪论

1.1 课题背景

当前，华中数控系统 PLC 控制是基于 DOS 环境使用 C 语言而设计的，该软件已经不太符合现代计算机大多采用 Windows 操作系统的实际要求，设计人员使用起来不直观、方便， 而且使用 C 语言编程同使用梯形图编程相比，对编程人员要求较高，根据这些情况，有必要研发一个基于个人计算机的能为用户提供直观、方便、高效的编程环境的 PLC 软件开发平台。

1.1.1 课题来源

根据华中世纪星数控车床开放性数控实验台 PLC 编程系统存在的问题，迫切需要研发一个基于个人计算机的能为用户提供直观、方便、高效的编程环境的 PLC 软件开发平台，要求建立 PLC 元器件符号库，提供以图形输入方式来建立梯形图程序。利用先进的计算机技术来实现可视化、规范化的梯形图设计以及 PLC 指令序列的自动生成，开发一种基于 Windows 2000，面向数控 PLC 的使用梯形图语言编写 PLC 程序的开发平台。使数控实验台在最大程度上发挥它的积极效用，使学生能够直观、方便地了解和掌握数控系统中 PLC 的编程和仿真操作。

1.1.2 数控系统概况

随着科学技术的发展及制造技术的进步，社会对产品多样化的需求越来越强烈，产品的更新换代周期也越来越短，中小批量生产的比重明显增加，从而对制造设备提出了更高的要求。数控机床作为一种自动化的加工设备而被广泛采用。

自 1952 年美国麻省理工学院与帕森斯公司合作发明了世界上第一台三坐标数控铣床以来，机床数控系统已经成为制造业自动化的核心技术和基础技术，是当今世界机械制造技术的高技术之一，可编程控制器是计算机技术与自动控制技术有机结合的一种通用工业控制器。在 PLC 出现之前，机床的顺序控制是以机床当前运行状态为依据，使

机床按规定好的动作依次地动作。这种控制方式的实现，是由传统的继电器逻辑电路完成的，这种电路是将继电器、接触器、开关、按钮等机电分立元件用导线连接而成的控制回路，由于它存在体积大、耗电多、寿命短、可靠性差、动作迟缓、柔性差、不易扩展等许多缺点，正逐渐被 PLC 组成的顺序控制系统所代替。

早期的数控系统 PLC 都是在 DOS 环境下设计的，DOS 环境下 PLC 操作起来不方便、不直观，对设计人员的编程水平要求很高；随着计算机应用技术的发展，计算机的操作系统由原来的 DOS 系统逐渐被 Windows 系统所代替，Windows 系统跟 DOS 系统相比，Windows 是多任务操作系统，具有丰富的资源，十分友好的人机界面，所以它已经成为数控系统软件开发人员的主要选择之一。

1.1.3 数控系统中的可编程控制器

（1）可编程控制器的概念

可编程控制器（Programmable Controller）是 20 世纪 60 年代末发展起来的一种新型自动化控制器。最早在美国通用汽车公司的自动装配线上使用并获得了成功。由于该控制器当时只是为了解决设备在运行中的开关量信号和逻辑控制问题，即用于替代传统的继电器控制装置，且只有逻辑运算、定量、计数及顺序控制等功能，因此把这种装置称为“可编程逻辑控制器”（ProgrammableLogic Controller），简称 PLC。后来随着技术的进步，其控制功能已远远超出了逻辑控制的范畴，人们就称其为 “可编程控制器”（ProgrammableController），简称 PC。在数控领域内，人们习惯称其为可编程逻辑控制器（PLC）或可编程机器控制器（PMC）。

（2）PLC 在数控系统中的类型

数控机床所用 PLC 可分为两类。一类是专为实现数控机床顺序控制而设计、制造的“内装型”PLC[1]，它从属于 CNC 装置的一部分；另一类是 PLC 独立于 CNC 装置，有完备的硬件和软件功能，能够独立完成规定的控制任务，以满足数控机床或其它顺序控制领域要求的 “独立型”PLC。

“内装型”PLC 与 CNC 之间的信号传送在 CNC 内部即可实现，PC 与机床侧的信息传送则通过 CNC 的 I/O 接口电路实现。一般这种类型的 PLC 不能独立工作，它只是 CNC

向 PLC 功能的扩展，两者是不可能分离的。

（3）可编程控制器的结构

可编程控制器的硬件主要由中央处理单元 CPU、存储器、输入/输出（I/0）模块以及电源组成。“内装型”PLC 可与 CNC 共用一个中央处理器（CPU），也可以单独使用一个 CPU。由于 CNC 的功能和 PLC 的功能就一起考虑，因而这种类型的系统在硬件和软件的整体结构上合理、实用、性能价格比高。PLC 和 CNC 之间没有多余的连线，而且 PLC 上的信息可以在 CNC 的显示器上显示，PLC 的编程更为方便，而且故障的诊断功能也有所提高。

华中世纪星系统采用的是“内装型”PLC，PLC 和 CNC 共用一个 CPU、输入/输出、电源。

可编程控制器的软件结构分为两部分，一部分是面向其内部的程序，即系统软件。这些软件的设计与数控系统软件设计的思路和方法基本一致。另一部分是面向用户或面向生产过程的“应用程序”（Application Program），也称 PLC 程序（PLC Program）或用户程序（User Program）。

（4）可编程控制器的编程方法

国际电工委员会 IEC 制定的国际标准 IEC1131-3 中规定了在 PLC 中使用的五种编程语言：顺序功能图、逻辑功能图、梯形图、指令表、C 语言等。其中梯形图和 C 语言是最通用的 PLC 编程语言[2]，华中数控 PLC 采用的是 C 语言编程方式。

编程器通过编程语言将用户程序送入可编程控制器。因此，编程器是 PLC 的主要辅件。编程器用作用户程序的编制、调试、监视、修改和编辑，并最后将程序固化在 RPROM 中。编程器还可通过其键盘去调用和显示 PLC 的一切内部状态和参数，并通过接口与 CPU 联系。

（5） PLC 的工作过程

用户程序通过编程器顺序输入到用户存储器内 CPU 对用户程序循环扫描并顺序执行。这是编程控制器的基本工作方式。

对用户程序的循环扫描执行过程，分为输入采样、程序执行和输出刷新三个阶段，如图 1- 1 所示。

输入采样阶段以扫描方式、顺序读入所输入端的状态（接通状态或断开状态），并将此状态存入输入映象寄存器中，接着转入程序执行阶段。

在程序执行阶段，即使输入状态变化，输入映象寄存器的内容也不会改变，状态的变化只能在下一个工作周期的输入采样阶段才被读入。程序执行总是按先左后右、先上后下的步骤对每条指令进行扫描，并从输入映象寄存器中读入所有输入状态。若程序中需要读入某输出状态，则也在此时读入，然后进行逻辑运算。运算结果再存入元件映象寄存器中。对于每个元件而言，元件映象寄存器所寄存的内容会随着程序执行的过程而变化。

所有指令执行完毕，元件映象寄存器中所有输出继电器的接通或断开状态在输出刷新阶段转存到输出锁存电路，再驱动输出线圈，这时的输出就是实际工作的输出。

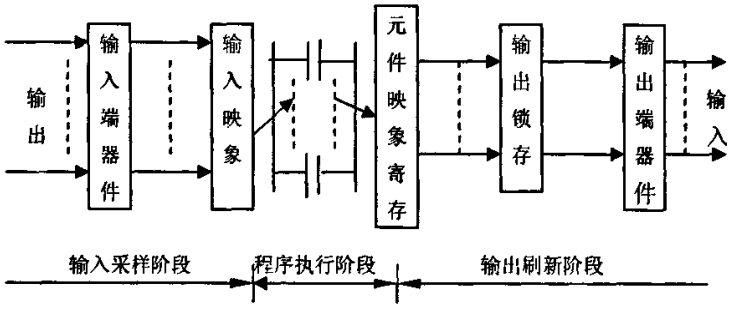

图 1-1 PLC 程序执行过程图

1.2 国内外研究现状

目前，数控机床的主要配套件大多选用经济型数控系统以及电动刀架等，而其内部的控制逻辑是由可编程控制器（PLC）来实现的。如何使机床各部件的动作有效可靠，关键在于 PLC 梯形图程序编制，设计 PLC 梯形图程序就成为工程人员的主要任务。

在国外，采用智能化标准化编程语言和编程工具是 PLC 编程的一个发展方向，也是 PLC 广泛使用的需要。梯形图语言形象、直观、通用、易于标准化，是首选的编程语言。

编程工具是开发应用和检查维护 PLC 以及监控系统运行不可缺少的外部设备。编

程工具的主要作用是用来编辑程序、调试程序和监控程序的执行，还可以在线测试 PLC 的内部状态的参数，与 PLC 进行人机对话等。编程工具可以是专用编程器，也可以是配有专用编程软件包的个人计算机。

（1）专用编程器

专用编程器是厂家提供的与本公司 PLC 配套的编程工具。专用编程器分为简易编程器和图形编程器两种。

简易编程器不能直接输入梯形图程序，只能输入语句表程序。用简易编程器编程时，编程器必须与 PLC 相连接。简易编程器的优点是价格低、体积小、重要轻、方便携带。有的简易编程器可以直接插在 PLC 主机的编程器插座上，如 OMRON 公司的 P 型机等。有的简易编程器要用专用电缆与 PLC 相连。

图形编程器可直接输入梯形图程序。图形编程器分手持式和台式。台式编程器具有用户程序存储器的功能，它可以把用户输入的程序存放在自己的存储器中，也能将用户程序转存到磁带上或打印出来，有的带有磁盘驱动器，可将程序转存到磁盘上。图形编程器的优点是屏幕大，显示功能强，但是其价格昂贵。

（2）使用个人计算机辅助编程

当前 PLC 的更新换代速度很快，因此专用编程器的使用寿命有限，价格一般也比较高。现在的发展趋势是使用个人计算机为基础的编程系统，PLC 厂家向用户提供在 PC 机上使用的编程软件。

采用通用计算机作为编程器的主要优势是使用了价格便宜、功能很强、通用的个人计算机。因此，用户可以使用已有的个人计算机，用最少的投资，得到高性能的 PLC 程序开发系统。对于不同型号、不同厂家的 PLC，只需要更换编程软件就可以了。它的另一个优点是可以用一台个人计算机为所有的智能控制设备编程。

现在，世界上各主要 PLC 厂家都提供了使用个人计算机的程序开发软件。这种软件的功能是相当强的。它可以编制、修改 PLC 的用户程序；监视系统运行；打印文件；采集和分析数据；作为实时图形操作器和文字处理机；对工业现场和系统仿真；将程序存储在磁盘上;实现计算机和 PLC 之间的程序相互传送等。利用它的网络软件，还可以作为网络管理器或通用的网络节点工作站。

以日本三菱公司生产的 FX2 系列 PLC 的梯形图编程软件为例，该软件能进行一系列的文档处理，如编辑（指令形式或梯形图）、存贮、打印、查错、转换类型、程序传送、图形监控等。但这种软件是专用的梯形图编程软件，而且价格昂贵，通用性差。

在国内，数控 PLC 梯形图编程软件设计梯形图程序时，绘图、修改、重绘图的工作量非常大，使控制工程技术人员把大部分的精力消耗在描绘元器件简单线条的重复性工作上，不能拿出更多的精力进行控制方案的优化设计。存在下面一些主要问题：

（1）华中系统 PLC 采用 C 语言编程，使用者必须学习并灵活掌握 C 语言，对使用者的编程水平要求较高。

（2）绘制 PLC 程序要做大量重复性的工作，并且一旦要修改，则需重新编制，造成人（程序员）计算工作量大、复杂。

（3）人机界面不友好，缺少汉字提示，操作不便。

（4）各大 PLC 厂家使用专用的梯形图编程软件，但是这些软件价格昂贵，通用性差。

（5）华中 C 语言 PLC 控制软件必须在 DOS 环境下运行。

目前，Windows 是 PC 上最通用的操作系统。Windows 是多任务操作系统，具有丰富的资源，十分友好的人机界面，所以它已经成为数控系统软件开发人员的主要选择之一。本课题研究的重点就是开发一种基于 Windows 2000 面向数控 PL 的使用梯形图语言编写 PLC 程序的通用开发平台。该系统建立 PLC 元器件符号库提供以图形输入方式来建立梯形图程序。这样不仅快速准确，清晰美观，输出方便而且修改也非常容易。

1.3 课题研究的目的和意义

PLC 是数控器系统程序的一部份，虽然有标准梯形图，但因为各种不同设备的动作千变万化，因此其梯形图往往必须加以修正，或建立新的梯形图来满足设备的特殊动作。华中 PLC 在 DOS 环境下采用 C 语言编程，程序对控制过程的描述不形象直观，对设计人员编程水平要求较高，而且目前，大多计算机不再使用 DOS 系统，人们已经习惯了使用 Windows 操作系统，为此开发出一个基于 Windows 2000 下友好的、针对性强的 PLC

梯形图编程系统，集矢量编辑、智能编译、打印等功能为一体，来修正或建立梯形图。该系统实现了可视化、规范化的梯形图设计以及 PLC 指令序列的自动生成。直观的编辑环境、灵活的编辑方式，方便了用户使用，提高了 PLC 编程效率。

1.4 课题研究的主要内容

梯形图是 PLC 编程的标准语言之一，它直观易学，所编程序很容易跟现场实际相结合，因此梯形图程序以图形方式表达应用程序的逻辑，直观明了。梯形图方便直观，在计算机上和控制技术上被叫做 “面向生产过程的语言” 。PLC 虽然历经了许多重大发展但它却一直沿用至今，而且仍然是编程人员的首选语言。

本课题从华中 PLC 编程系统的特点出发，采用 C++ Builder 作为开发工具，在开发设计过程中采用了面向对象的方法，由梯形图绘制作为切入点，实现了梯形图到 PLC 指令代码的转化，最后在 Windows 系统下模拟 PLC 运行过程，实现了 PLC 指令解释器的仿真，实现 PLC 仿真。

本课题的主要工作如下：

（1）项目的需求分析和总体设计。根据收集的技术资料，分析系统的总体需求，基于现有技术，确定出系统的总体技术路线。

（2）用户界面的设计与实现。为用户提供直观、方便、高效的编程环境。

（3）梯形图编辑模块的设计。为工程师提供了一个绘制梯形图的工具，是人机交互的重要手段。它可以对用户程序的触点和线圈加上注释，并能对某一程序段加注说明，使程序容易阅读和理解。该模块是整个软件的基础，对于这个功能块最重要的一点是要达到界面友好，简单易用的效果。

（4）编译模块的设计。编译是“从上至下，从左至右”，以梯级为单位，按 PLC 逻辑顺序逐个单元编译的。梯形图经检查无误后，可以转换成唯一的一个指令集。

（5）PLC 仿真模块的设计。它允许计算机对生产过程和系统进行仿真，使设计者在系统实际建立之前，通过仿真处理，发现设计中存在的问题，避免不必要的浪费和因设计不当造成的损失，缩短系统设计、安装和调试的总工期。

强的数控 PLC 梯形图编程系统，集矢量编辑、智能编译、打印等功能为一体，来修正或建立梯形图。该系统实现了可视化、规范化的梯形图设计以及 PLC 指令序列的自动生成。直观的编辑环境、灵活的编辑方式，方便了用户使用，提高了 PLC 编程效率。

1.5 本章小结

本章介绍了本课题的背景、国内外 PLC 的现状，了解了“机床数控 PLC 编程及其仿真系统的研究”的目的和意义，基本的研究、设计思想，明确了研究本课题需要进行的主要工作。

2 数控梯形图编程系统的整体设计

2.1 数控系统 PLC 的基本特点

在数控器系统中，CNC 和 PLC 协调配合共同完成数控机床的控制。PLC 主要完成与逻辑运算有关的一些动作，没有轨迹上的具体要求，它接受 CNC 的控制代码 M（辅助功能），S（主轴转速），T（选刀、换刀）等顺序动作信息，对其进行译码，转换成对应的控制信号控制辅助装置完成机床相应的控制信号控制辅助装置完成机床相应的开关动作，如工件的装夹、刀具的更换等一些辅助动作;它还接受机床操作面板的指令一方面直接控制机床的动作，另一方面将一部分指令送往 CNC 用于加工过程的控制。内嵌式 PLC 与 CNC 间的信息传送在 CNC 内部实现，PLC 与机床间的信息传送是通过 CNC 的 1/0 接口电路实现。

PLC 内嵌在数控装置中，可以进行读写操作，称为嵌入式 PLC。嵌入式 PLC 能充分借助 CNC 系统的全部硬件资源，具备友好的界面接口，便于用户编程和操作。

图 2－1 给出了 CNC 系统的机床是如何通过 PLC 联系起来并通过 PLC 互相传递信息的。图 2－1 中 G 为 PLC 到数控系统的输入信号，F 代表数控系统对 PLC 的输出信号，X 代表机床侧向 PLC 的输入信号，Y 代表 PLC 对机床的输出信号。由图 2-1 可以看出 PLC 通过 4 组信号将机床与 CNC 有机地联系在一起。

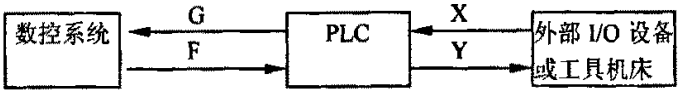

图 2-1 CNC、PLC、机床信号流向图

2.2 系统的整体设计方案

系统参照日本三菱公司 FX2 系列 PLC 编程标准而设计的，是基于 Windows 系统上的图形开发系统，进行图形化编程。以图形输入方式来建立梯形图程序，界面直观，操作简单，能快速绘制 PLC 梯形图，具有方便的编辑环境，灵活的编译方式。在数控

器系统中，PLC 编程主要用到的指令有 LD 指令、逻辑运算指令、算术运算指令、定时指令、记数指令、分支指令、控制指令等。数控系统调用 PLC 程序入口，调用初始化程序，对系统进行初始化，初始化完成后，对 PLC 源程序进行编译，编译成功后，生成名为 PLC null.COM 的文件，然后，更改数控系统配置文件 NCBIOS.CFG，系统启动时自动调用 PLC 文件，从而实现 PLC 与数控装置的联系。

这种高度可移植性的软件系统不仅使软件操作起来简便易行，在线维护方便，而且可视化的图形界面使梯形图编程软件具备了良好的人机交互界面。梯形图编程系统总体流程图如图 2-2 所示。

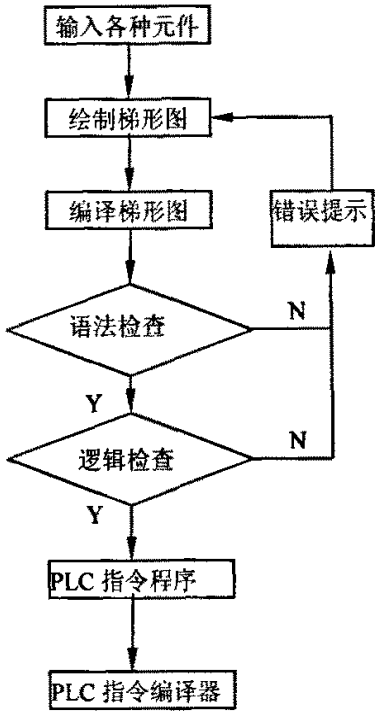

图 2-2 梯形图编程系统总体流程图

本系统是针对 Windows2000 的开放式 PLC 控制系统，能提供以图形输入方式来建立梯形图程序。PLC 软件集成开发的总体环境可以分解成一些相对独立的功能模块，因此，软件采用模块化方法设计。集成环境主要由项目管理模块、编辑模块、打印模块、模拟仿真模块等组成。系统框架结构如图 2-3 所示，现分别介绍各模块的功能。

（1） 项目管理模块

该功能模块主要用于管理和归档所有数据，包括用户编辑的源程序、编译生成的指令代码。

（2）梯形图语言编辑模块

编辑模块是集成环境最基本的功能模块，主要负责 PLC 梯形图编辑，能够实现梯形图语言的输入、编辑、存储、显示等功能，为工程师提供了一个绘制梯形图的工具。对于这个功能模块最重要的一点是要达到界面友好、简单易用的效果。绘制梯形图时，需要把常开触点、常闭触点、线圈、定时器、计数器等元件图形化，图形化的手段是以工具的形式出现，就像 Windows 中画笔程序提供的刷子、笔、颜料一样。

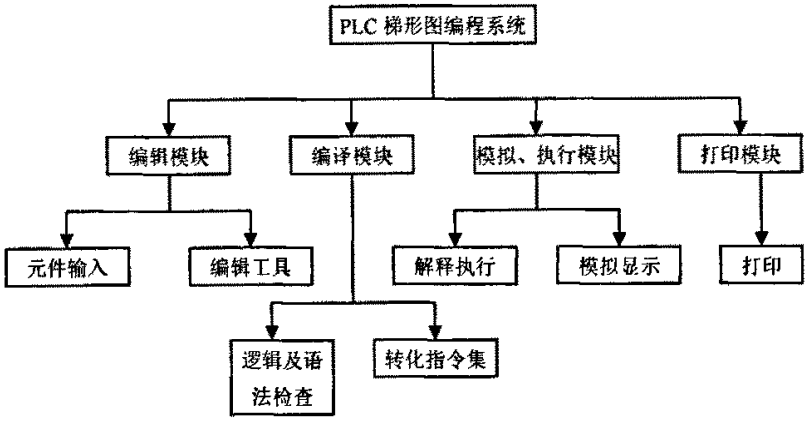

图 2-3　系统框架结构

（3）编译模块

编译是“从上至下，从左至右”，以梯级为单位，按 PLC 逻辑顺序逐个单元编译的[7]。梯形图经检查无误后，可以转换成唯一的一个指令集。应具有以下功能：

- 语法检查功能。检查用户 PLC 图形程序有无指令标记重复、指令地址越界的语法错误。如检查梯形图是否有短路或空路电路。
- 逻辑关系检查功能。检查 PLC 图形程序有无逻辑错误，如指令位置不正确、梯级中缺少输出指令等，编译梯形图。
- 报警功能　重复使用不能再使用的元件组成被查出时，会显示报警。
- 错误提示功能　编译结果有错误，错误信息会显示在梯形图错误信息窗口显示出来。

- 档案管理功能　编译结果良好，适当的信息会显示在梯形图错误信息窗口，同时在状态栏也会显示。除此之外，在同样的档案目录下，会产生梯形图组成元件明细表档案。

（4）打印模块

该模块用来打印出梯形图，以便检查。

（5）仿真模块

仿真模块实现了系统的离线仿真，可以对用户 PLC 程序的逻辑错误进行检查修改。这个功能模块主要实现了 PLC 指令解释器的仿真，PLC 运行过程的仿真，提供了模拟 PLC 输入输出的工具。

除上述五大模块之外的一些其他功能，如集成环境界面管理、联机帮助等，这些功能的实现大都分散或内嵌在以上各模块之中，所以，没有将他们单独列出。

集成环境虽被分解成五个相对独立的模块，但这些模块之间仍存在着密切的关系，处理好它们之间的信息传递和驱动，是集成环境能否设计成功的关键。图 2-4 给出了 PLC 软件开发平台的系统模块间的数据流向图。

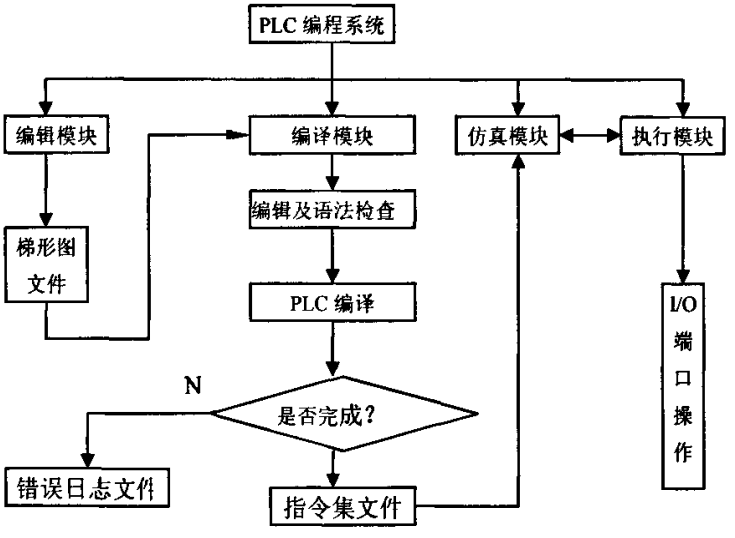

图 2-4　PLC 编程系统模块间的数据流向图

2.3 开发工具的选择

根据设计本程序的需要，选择 C++ Builder 5.0 为开发工具。

C++ Builder 5.0 是著名的 Borland 公司开发的可视化开发工具。它基于 Microsoft 公司的 32 位操作系统 Windows 系列，具有高效、最优化、可视化应用程序开发环境和强大的数据库开发能力。C++ Builder 5. 0 使用的程序语言是 C++语言，C++是一门高效实用的程序设计语言，它既可进行过程化程序设计，也可以进行面向对象的程序设计，强调对高级抽象的支持，用它开发出的应用程序具有可重用的特点[8]。

C++ Builder 5. 0 语言中采取的面向对象技术。经过二十多年的发展，已渐成熟与完善，并广泛应用于科学与工程等多个领域。高质量的代码能有效降低软件的复杂度和提高开发效率。面向对象方法的继承性是一种代码重用有效途径。在软件设计时可以利用一些已被精心设计好并且经过测试的代码这些可重用的代码被组织和存放在程序设计环境的类库中。由于类库中的这类的存在，使以后的程序设计过程变得简单，程序的复杂性不断降低、正确不断加强，也越来越易于理解、修改和补充。利用面向对象的方法，能够建一个定义良好的接口，以帮助系统的设计、实现、维护和程序的重用。

2.4 面向对象技术的应用

面向对象技术是计算机应用领域在九十年代迅速发展起来的一个新生事物，它的出现被认为是程序设计方法学方面的一场实质性革命。

它与传统的结构化程序设计相比较，具有三个鲜明特性：

- 封装：指将一个数据和与这个数据有关的操作集合放在一起，形成一个能动的实体—对象，用户不必知道对象行为的实现细节，只需根据对象提供的外部特性接口访问对象即可；
- 多态：它是指同一个消息可以根据发送消息对象的不同采用多种不同的行为方式；
- 继承：它表达的是一种对象类之间的相交关系。它使得某类对象可以继承另外

一类对象的特征和能力。

特别是在面向对象的系统中，引入继承机制后具有如下优点：

（1）减少代码和数据的冗余度，大大增加程序的重用性；

（2）是在一些比较一般的类的基础上构造、建立和扩充新类的最有效的手段。在梯形图绘制中，元件有自己的特性，更有共性，所以，面向对象技术适合于建造梯形图的元件类[9]。

2.5 软件工程的思想

20 世纪 60 年代以来，随着计算机应用需求的驱动，软件生产的复杂性和成本都日益提高，大型软件的生产出现了很大的困难，即出现了软件危机，主要表现在：生产成本高、进度无法控制、可维护性差等方面。认识早期软件开发中所存在的问题和产生问题的原因，并设法克服是解决软件危机的开始。人们发现将传统工程学的原理、技术和方法应用于软件开发可以起到使软件生产规范化的作用，它有利于组织软件生产，提高开发质量，降低成本和控制进度。

提出软件工程化的思想，反映了人们想使软件生产走上正规化，人们曾从不同的角度给软件工程以定义，但它们的核心内容都是 “以工程化的方式组织软件的开发”，其中涉及软件的计划、开发和维护几个阶段。

本软件的算法比较复杂，需要实现的功能很多，而且大量功能的交织有一个协调问题，从程序的宏观把握上相对困难一些。本软件中的各种状态参数，临时变量就达数百之多，还有众多的元件需要设置状态，因此有必要引入软件工程的方法，组织数据字典、记录数据流图，并且书写大量的软件文档，为日后的软件维护提供参考。为了保证本软件的开发质量，软件设计时应以软件工程中的规范为指导，对软件从定义到开发的各个环节做出合理的规划，认真实施，很好地实现预定的功能，才能做到运行可靠，便于维护。

它提供许多基本算法、数据结构。STL 是一个通用库，即可以充份定制，几乎所有的 STL 组件都是模板，可以仅仅使用这些现成的组件。STL 现在是 C++的一部分，因

此不用额外安装什么，它被内建在编译器之内。list 容器易于使用，是可动态改变大小、顺序，将传统工程学的原理、技术和方法应用于软件开发起到了使软件生产规范化的作用，有利于组织软件生产，提高开发质量，降低成本和控制进度。

2.6 数据结构的设计

一个梯形图程序是一个梯形图元件的集合，包含的元件数量可以有许多。因此，必须采用恰当的数据结构来组织好它们之间的关系。

对于数据结构类型的选择，在本课题中主要考虑了如下方面：首先是要满足对于待处理的数据元素及其关系的描述；其次，在能够完整描述问题空间所有数据元素及它们之间关系的基础上，应采用尽可能简单的数据结构，以避免复杂数据结构带来的复杂操作；同时，应考虑与其它模块之间数据结构的通用性。基于以上考虑，本课题中采用了具有线性特性的数据结构来实现梯形图编辑过程中元件添加、删除及修改时设计数据的保存和读取数据的高效性[9][10]。

系统采用了标准模板库 （STL ）中的标准顺序容器 list。标准模板库（Standard Template Library），是标准 C++库类的一部分，它为存储和处理数据提供了标准步骤。STL 包含多类实体，最重要的三类是容器、算法和迭代器它提供许多基本算法，数据结构。STL 是一个通用库，即可以充份定制，几乎所有的 STL 组件都是模板。STL 是标准化组件，是 C++的一部分，内建在编译器之内。list 容器相当易于使用，list 是可动态改变大小的顺序存储的线性表，也叫 “双向链表”。

STL 容器可以保存对象，内建对象和类对象，并定义能够操作的这个对象的接口。因此，在 STL 容器中的对象很安全。STL 算法是标准算法，应用在那些容器中的对象上，给对象排序，删除，记数、比较，找出特殊的对象，合并到另一个容器中，以及执行其他有用的操作。STL 所有的东西，就是容器、算法和允许算法工作在容器中的元素上 iterator（迭代器）。算法以合适、标准的方法操作对象，STL iterator 就像容器中指向对象的指针。

在本软件中采用了 list：容器来存储设计过程中涉岌到的梯形图数据，主要对象容

器定义如下：

（1）梯级链表：Typedef list<CRung*> CRungList;

（2）行链表：Typedef list<CRow*> CRowList;

（3）元件链表：Typedef list<CElement*> CElementList;

通过 list 容器模板自带的 push_ back 成员函数将要加入的对象压入到相应种类容器中，通过这样的操作就把各对象逐一的放入容器中，再通过其他成员函数如 erase, insert 等对它们进行操作，实现各对象数据的集中管理。

当用梯形图编程系统绘制梯形图时，每在计算机屏幕上画一个梯级，就会生成该梯级类的一个对象，同时程序把该对象加入到梯形图类中的 CRungList 链表当中；每画一个行，就会生成该行类的一个对象，同时程序把该对象加入到对应的梯级对象中的 CRowList 链表当中:每画一个元件，就会生成该元件类的一个对象，同时程序把该对象加入到对应行对象中的 CElementList 链表当中所以，在 CElementList 链表中的一个结点对应梯形图中一个水平连线、元件或功能块等。

2.7 本章小结

本章通过应用工程软件的思想，对数控 PLC 集成开发平台进行了需求分析和总体设计，选择了 C++ Builder 5.0 为开发工具，采用模块化方法利用面向对象技术对编程系统进行设计，将 PLC 软件集成环境分解成项目管理、编辑、编译、模拟仿真等模块，并对内部数据结构进行了设计，以得出各部分之间的数据联系。

3 PLC编辑器的设计

3.1 引言

梯形图是 PLC 中最典型的、最基本的一种编程方式。它采用图形语言，沿用了继电器的触点、线圈、串并联等术语和图形符号，并增加了一些继电接触控制没有的符号[11]。梯形图形象、直观，对于熟悉继电器表示方式的人来说，非常容易接受，而且不需要学习更深的计算机知识。这是一种最为广泛的编程方式，适用于顺序逻辑控制、离散量控制、定时/计数控制等操作。设计好梯形图编辑器对于梯形图编程系统来说是非常重要的。首先介绍系统类层次结构与元件库对象的设计，然后介绍梯形图绘制的具体实现和储存。

3.2 编程器的设计原则

梯形图起源于继电器逻辑和执行线路，它用不同的图符来表示不同的指令，用串、并联等概念组织图符的顺序位置来表述逻辑。梯形图语言作为一种标准 PLC 编程语言，在编制时必须遵循一定的规则，具体如下：

（1 ）触点应画在水平线上，不能画在垂直分支上且应遵循自左至右，自上而下的绘制原则。

（2 ）梯形图的每一行都是从左侧母线开始画起，线圈和指令画在最右边，线圈或指令的右边只能画右母线（右母线可以省略）。

（3 ）线圈和指令不能直接跟左侧母线连接（除极少数没有执行条件的指令，如 END 等）。

（4 ）梯形图必须遵循从左到右、从上到下的顺序编写。

（5）程序结束时安排 END 指令，否则程序不被执行。

所以，在梯形图编程器开发过程中必须遵循梯形图的编程原则来设计。

3.3 系统类层次结构

梯形图一般由多个不同的梯级组成，每一个梯级又可以由“一行”或“数行”组成，每行由一个或几个输入元件及一个输出元件组成。输出元件应出现在梯级的最右边，而输入元件则出现在输出指令的左边，如图 3-1 所示。

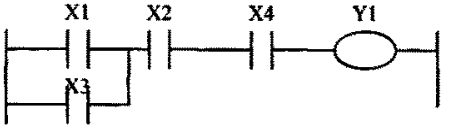

图 3-1 一个典型的梯形图

从计算机角度看，图 3-1 是一幅位图，它由一个个像素组成。计算机没有能力分析出这样一个梯形图，并判断出每个元件之间的逻辑关系，所以，需要一种描述方法，让计算机能“看懂”梯形图。为此，采用面向对象设计方法来开发梯形图编辑器，其关键步骤如下：

（1 ）首先分析和识别梯形图中的不同对象。以图 3-1 的梯形图为例，此梯形图是一个由两行组成的梯级。第一行由元件 X1, X2, X4, Y1 及向下的连接线组成，第二行由元件 X3 及向上连接线组成。梯形图可分为梯级、行、元件三部分，其数据结构图如 3-2 所示。所以，系统由梯形图对象、梯级对象、行对象及元件对象组成。

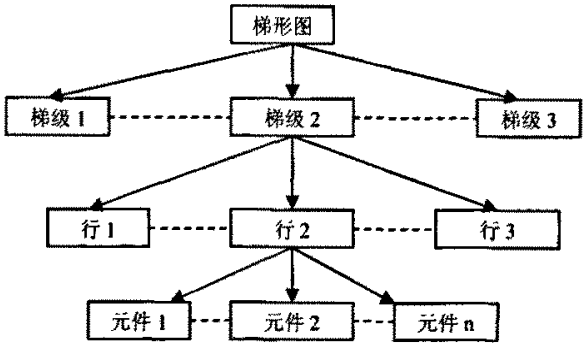

图 3-2 梯形图的数据结构图

（2） 根据对象的性质和功能抽象归并为不同类，建立类的层次结构。面向对象软件设计主要是类的设计，而不是对象的设计，因为各种对象是在梯形图对应的编辑过

程中动态产生的。通过对系统的梯形图对象、梯级对象、行对象及元件对象进行抽象，建立了梯形图编程系统的类层次结构如图 3-3 所示，其类层次结构中六大类对象如下：

1) 类 CLadder，是梯形图类，每个梯形图都对应一个梯形图类对象，处于最上层，通过类中的梯级链表 CRungList 来管理下面的梯级信息。当插入一个梯级时，就把生成梯级对象的地址指针插入到链表中对应位置。

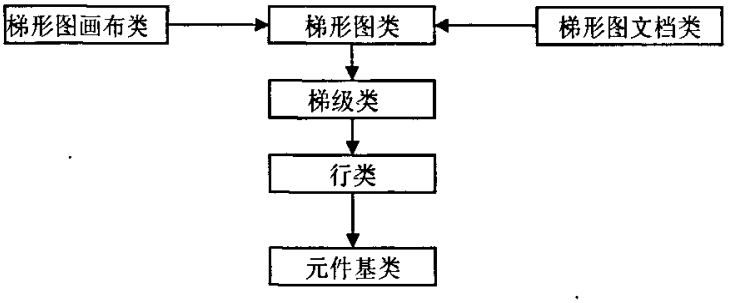

图 3-3 编辑模块的类层次结构

2）类 CRung，是梯级类，是梯形图的基本单位，即相互影响的行组成的最小单元，是梯形图中一个功能相对独立的控制单元，通过行链表 CRowList 管理下面的行信息。

3）类 CRow，是行类，是梯级的基本单位，通过元件链表 CElementList 管理下面的元件信息.

4）类 CElement，是元件基类，不表征具体的对象，但它定义了所有元件类具有的共同属性和操作接口，是其它元件类的父类。

5） 类 CLadderCanvas，是梯形图画布类，负责管理梯形图的绘制。

6） 类 CLadderDoc，是梯形图文档类，负责管理梯形图的存储操作。

3.4 元件库对象的设计

通过对梯形图及其设计规则的分析，用梯形图语言编制程序的过程实质上是一个用梯形图图符来表示操作指令、用图符的串并联及位置顺序来表示操作指令之间逻辑关系的过程。由此，一个梯形图可以看作是由梯形图图符和连接符组合而成，其中：

（1）梯形图图符分为:触点类（Contact）、指令类（Coil）、线圈类（Box）。

（2）连接符分为:向上线、向下线、向左线、向右线。

所以，梯形图的元件库是由梯形图图符和连接符组成。每一个元件建立一个类，并且根据元件之间的关系结合成一个整体，形成一个与现实元件关系一致的元件组合—元件类库。有了元件的类库后，用类生成很多实例对象，这些对象就构成一幅梯形图画面，生成的对象不同，对象的属性不同，得到的梯形图画面也就不同，从而可以实现梯形图的编辑。

要描述一个元件，主要是描述它的各种属性，在软件当中主要是体现图形属性和物理属性。图形属性是指:工程师使用梯形图编程软件将元件画在计算机屏幕上，形成梯形图，当元件被以元件图形画在计算机屏幕上时，描述该元件的坐标、大小等属性就被称为元件的图形属性。物理属性就是指元件的名称、注释等属性。

纵观要描述的元件的属性，有些属性是每个元件都具有的属性，例如元件类型、所在位置、名称等。其中，元件类型表示元件的种别，如水平连线、常开触点、常闭触点、常开线圈、定时器、计数器等;要表示所在位置将整个梯形图看成一个平面，再将这个平面划分成一个网格矩阵,使用元件在网格矩阵中的行列就可以表示;名称也是元件的一个必要属性，因为它代表 PLC 硬件中的具体地址，如 X0 表示 PLC 硬件的第 0 个输入点。为此设计一个基类,这个类就封装了这些共有的属性。再从这个基类派生出各个元件类，它们封装了每种元件对象所特有的属性。

描述元件除了描述它的各种属性以外，还有一些函数，例如在计算机屏幕上绘制、选择、存储元件图形的函数，以及设置、获得元件属性值的函数等。这些方法有些也是所有元件共有的，只是具体的实现方法不一样而已，对于这些方法把它们在基类当中声明为虚函数，在各个元件派生类当中重载。例如在计算机屏幕上绘制元件图形的函数，它就是每个元件都具有的函数，只是具体的实现方法不一样，可以在基类当中声明为虚函数，在各个元件派生类当中重载，在重载程序中具体写出绘制相应元件的函数。这样将大大提高程序的可读性、灵活性，节省代码。对于某种元件特有的函数，就把它写在相应元件的派生类当中去[16]把各种元件定义为相应的类以后，每当在计算机屏幕上画一个元件时，程序就生成一个该元件类的对象，并且这个对象的属性和函数就描述了相应的元件。程序就是通过判断或运算这些对象的属性来实现各种功能的。

3.4.1 元件基类设计

元件库模型的基类 CElement 是从梯形图编程系统所支持的所有元件抽象出来的一个类，它定义了其它类的共有操作接口和属性，是其它图元类的父类。模块中所有的元件子类都根据 C++的继承机制继承了基类的属性，并根据自己支持的图元形状有选择地重载基类相应的操作，以满足子类的需要。所以基类的定义在梯形图编辑模块的实现过程中是非常关键的一步 [17]。

采用的规则是：将具有相同性质，包括相同外部性质和内部处理能力的对象归为一类作为最低层次，然后采用自下而上逐步抽象的方法，将具有共性的类的公共性质再并入一个相对于被抽取共性的类的基类中；被抽取共性的类便为导出类，抽取成形的类为基类。如此类推，不断产生更多的基类，最终建立了类的层次结构。在对所有梯形图元件进行共性抽象之前，做了如下处理：

（1 ）梯形图元件除元件自身信息外，还应包含其连接线的信息。

（2） 把梯形图的空格和水平连接线作为特殊元件。

（3） 在梯形图中，常开触点、常闭触点等元件有一个操作参数；定时器、计数器等元件有两个操作参数；空格和水平连接线等元件没有操作参数。为了对所有梯形图元件进行共性的抽象表示，每个元件设置两个操作参数。例如常开触点只有一个操作参数时，另外一个操作参数设置为空。

抛开梯形图的图像表象，梯形图中每个元件包含的共同属性如图 3-4 所示。元件基类定义元件的所在位置、持久化和事件处理等操作，其定义描述如下：

（1） 标识信息：元件唯一的功，类型及元件的操作参数。

（2） 位置信息：元件的顶点位置，长度等。

（3） 绘制行为：绘制元件图符，绘制元件包含的线型。

（4） 命中测试行为：鼠标的位置是否命中图元以及具体部位。

（5） 持久化行为：存盘和读取。

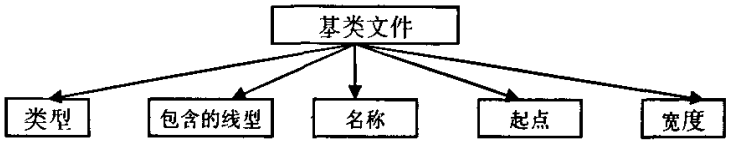

图 3-4 基类元件的共同属性

梯形图元件基类的关键代码如下：

```
Class CElement
{
  Protected:
  Typedefs truct//元件信息数据结构
  {
    LD_ELEMENT__TYPE  elemType;  //元件类型
    int  nWire;  //元件包含的线型
    int  nElementID;  //元件 ID
    }ELEMENT_INFO;
    AnsiString  m_strName;  //元件的操作参数一
    AnsiString  m_strParamName;  //元件的操作参数二
    bool  m_f Selected;  //鼠标的位置是否命中元件
    void  SetWidth ( );  //设置元件的长度
    void  SetType ( );  //设置元件的类型
    int  GetLeft (void);  //获得元件的顶点
    //元件包含线型的处理函数
    int  GetWire ( )  const {return  m_elemInfo.nWire; }
    void  SetWire (const  int  nWire)  {m_elemInfo.nWire= nWire; }
    void  AddWire (const  int  nWire)  {m_elemInfo.nWire} | = nWire; }
    void  RemoveWire (const  int  nWire)  {m_elemInfo.nWire | =~nWire; }
  Public:
    //元件类的构造函数和析构函数
    bool  Open (TFileStream*pms);  //存盘
    bool  Save (TFileStream*pms);  //读取
    virtual  void Draw ( );  //绘制元件图符
    void  DrawWire ( );  //绘制元件包含的线型
```

```
    ......
};
```

3.4.2 各元件类的设计

有了这个基类，以它为父类，根据每个元件的不同特性，为每个元件设计一个子类。各元件类以相应的英文单词来命名，如常开触点类（Normally Open Contact）命名为CContactNO。每一种元件都在本程序的工程组当中单列一个的单元，并把相应元件类的定义写进去。工程组当中的其他单元，只要在接口部分调用该单元，就可以使用该类了。图 3-5 描述了元件库中各类的继承派生关系。

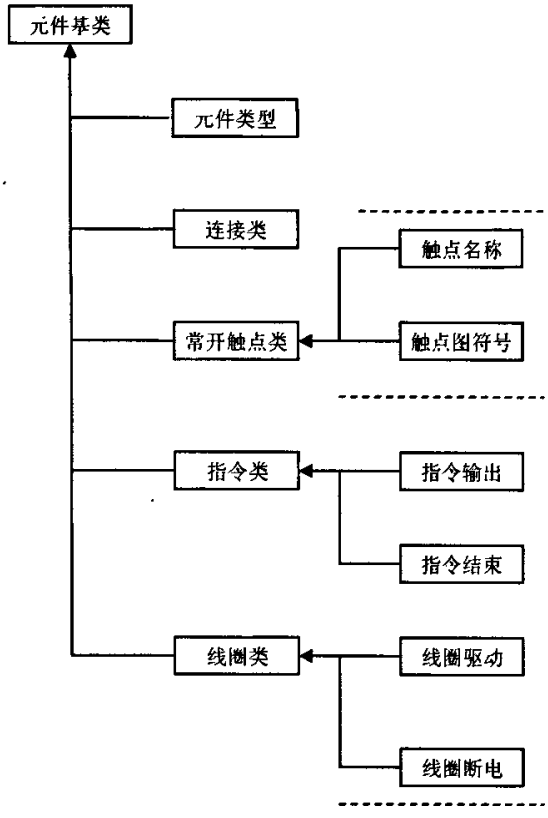

图 3-5 元件类结构图

下面以常开触点作为例子，起其代码如下：

```
Class CContactNO：public CElement
 {
   //类的构造函数和析构函数
   //绘制常开触点图符（—| |—）的绘图函数
  Void  draw  （TCanvas*pCanvas，int nLeft，int nTop，int nWidth，int nHeight，int
              nContactWidth）；
  }
```

绘图函数的具体实现如下：

Draw （TCanvas*pCanvas，int nLeft，int nTop，int nWidth，int nHeight，int

```
                nContactWidth)
{
    OrgX= nLeft+ nWidth/2; //计算原点
    OrgY= int nTop + int nHeight; //计算原点
    //绘制常开触点的左边
    DrawLine (pCanvas, nStartX, orgY, orgX- nContactWidth/2, orgY);
  DrawLine (pCanvas, orgX+1- nContactWidth/2, orgY-8,
              orgX+1- nContactWidth/2, orgY+8);
    ...../绘制常开触点的右边
}
```

3.5 编辑器的具体实现

一个高效、舒适和直观的编辑器不但能极大提高用户的编程效率，并能降低出错的机率，梯形图编辑模块所要实现的主要功能有：设计一个类似于 Word 的任意浮动的工具栏，在这个工具栏当中把各种梯形图元件用按钮表示，形成一个元件图形库；实现元件的操作的功能，包括元件图形的绘制、选择、插入、删除、复制、粘贴等操作；在用户绘制包含有一些基本语法错误的元件时，提示并取消操作；支持滚动，在编辑内容超过当前窗口范围时，自动产生滚动条；提供“撤消”和“重做”的编辑功能；支持对梯形图中的元件属性的更改和梯形图文件存盘、读盘的功能[18]。

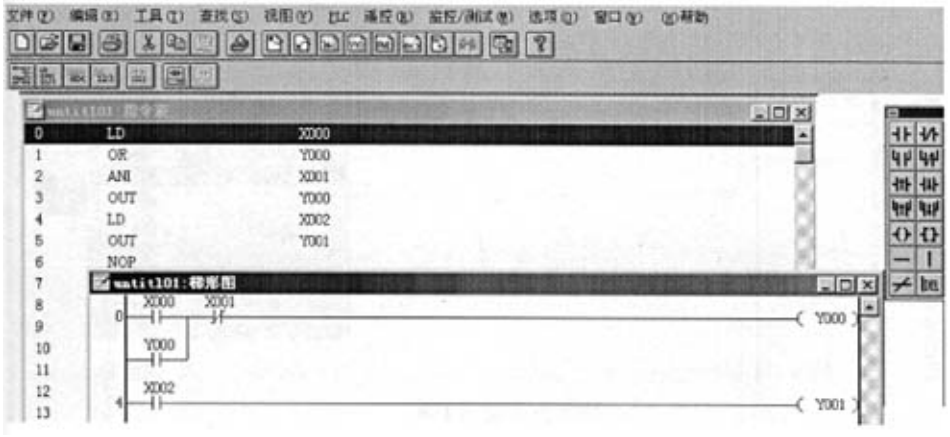

图 3-6 梯形图编程系统主界面

图 3-6 显示了梯形图编程软件的主界面，整个软件界面有五个窗体构成[19]。

（1）主窗体：显示系统主菜单，工具栏，绘制梯形图所需的基本图符按钮；

（2）绘图窗体：用于梯形图的绘制；

（3）代码生成窗体：显示由所绘梯形图编译生成的 PLC 指令代码。

（4）参数输入窗体：为绘制的每个图符设置参数。

（5）出错信息窗体：报告梯形图编译成 PLC 指令时的出错信息。

图 3-7 所示的是双击屏幕中某个元件时弹出的对话框，在其中可以对元件属性进行更改。

图 3-7 元件属性对话框

图 3-8 所示的是插入对话框，提供在选择元件处的前后，插入一个或多个梯级、行和列的操作。

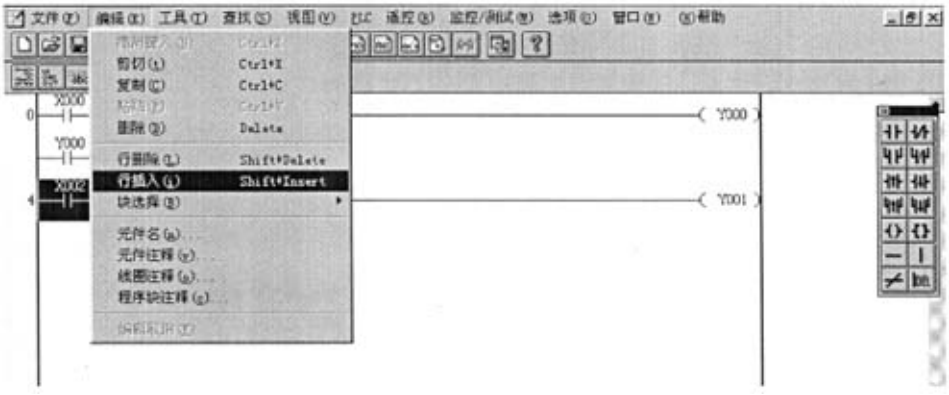

图 3-8 插入对话框

3.5.1 梯形图的绘制

画梯形图主要是在画布TPainBox控件的TPaintBoxMouseDown事件当中使用TPainBox控件中的TCanvas类。该类封装了Windows GUI的高、中、低级函数，它主要有Pen（画笔）、Brush（画刷）、Font（字体）属性和绘制各种几何图形的方法。Pen属性用于设置画笔颜色、宽度和模式等；Brush属性用于设置画刷的填充颜色、模式等；而Font属性用于设置画布上文本的字体、大小、颜色等。另外TCanvas类还具有绘制各种几何图形的方法，可绘制直线、矩形、圆、椭圆等几何图形。把梯形图画布类（CLadderCanvas）中的画布指针与TPainBox控件中的TCanvas类接上，可以进行梯形图的绘制。

梯形图画布类（CLadderCanvas）应该具有下列功能：

（1）提供利用工具栏、键盘和鼠标来绘制梯形图；

（2）实现画面的重画；

（3）响应拉动滚动条事件；

（4）实现修改元件的图形属性和操作属性的方法；

（5）绘制梯形图后，重新计算梯形图画布大小。

其定义描述如下：

```
Class CLadderCanvas
private:
```

```
CElement*m_pCursor;  //选中元件
CrungList m_SelectionList; //选中梯级所在链表
CView*m_pView;  //窗体指针接口
TCanvas*m_pCanvas; //画布指针接口
public:
CLadderCanvas (CView *view) ;
virtual ~CLadderCanvas () ;
// 重画函数
Void Refresh  (int nStartX, int nStartY, int nWidth, int nHeight)
void OnMouseEvent (MouseEvent event) ;  //鼠标事件
void OnMouseDClick (MouseEvent event) ; //鼠标双击事件
void OnKeyDown ( KeyEvent event) ; //键盘事件
void OnEditDelete (void) ; //编辑删除
void OnEditInsert (void) ; //编辑插入
void OnEditElements (void) ; //编辑元件
void OnEditWire (int nId) ; //编辑连接线
void ResetCanvas (void) ; //重设画布
void ResizeCanvas () ; //重新计算画布大小
void RecalcLadderElemSizes () ; //重新计算元件大小
 void RecalcRow ( CRow*pRow) ; //重新计算行
 ......
}
```

为了方便的绘制梯形图，在新建一个梯形图工程时，设计预先生成一个梯形图。它由30个梯级组成，每个梯级有一行，每行由11个水平连接线元件和一个NOP元件组成。在梯形图编辑过程中，主要是元件、行、梯级的绘制和编辑。

（1）元件的绘制

画某类元件时，用户先用鼠标选中在梯形画布上想绘制元件的某个网格，然后再点

击工具栏中该类元件的按钮，就可以画出该元件了。绘制元件的实质是先创建元件类对象的构造函数中，把网格区选中元件的图形属性赋给该对象，然后释放选中元件，删除行链表中选中元件的对象指针，插入新生成的对象指针，最后调用元件的绘图函数，就实现了绘制元件的功能。

（2）行的绘制

画行时，点击菜单中插入行的选项，就会在当前鼠标选中元件所在行下面插入一行。绘制行的实质是先创建一个由12个空格元件组成的行的对象，再把该行对象的地址指针插入对应的梯级链表中。

（3） 梯级的绘制

梯级的绘制过程与行的绘制过程类似。

3.5.2 元件的选择

在梯形图的编辑过程中，只有对处于被选状态的元件才能进行编辑。画在梯形画布上的元件，它的属性一般不能完全满足要求，还需要对它进行一定的修改。在修改某个元件属性之前，应该将这个元件处于区别于其他元件的被选状态，并且用一个指针指向这个元件对象，以便对被选元件进行各种操作。在本程序当中，如果将鼠标点击在画布上某个已画元件的时候，程序将使该元件用蓝色矩形框重画，表示其处在被选状态，并用指针m_pCursor指向该元件，同时将元件对象中的一个已定义的布尔型变量m_fSelected的值由初始化时False该为True。

具体的实现方法是如下:当鼠标点击画布的某个地方时将触TPaintBox控件的TPaintBoxMouseDown事件，把获得鼠标点击处的坐标X, Y传给函数FindElement（int x, int y）后，先遍历梯级链表，再遍历梯级对象中的行链表，最后遍历元件链表来找到元件对象。

```
CElement* CLadderCanvas::FindElement (int x, int y)
{
  while （ 遍历梯级链表）
    {
```

```
        //获得梯级对象中的行链表
          while（遍历行链表）
          {
            nTempY+=pRow->GetRowHeight（）；//获得当前行纵坐标大小
            if（ nTempY>y）
              {
                //获得行对象中的元件链表
                  while （遍历元件链表）
                    {
                      //获得当前元件横坐标大小
                    nTempX=pElement->GetLeft（）+pElement—>GetWidth（）；
                      if（ nTempX > x）
                        return  pElement;  //返回元件类对象
                    }
                }
            }
      }
  }
```

3.5.3 响应拉动滚动条事件

梯形图用户程序不一定能在一屏之内画完，所以要实现滚屏功能。设计编辑器时，考虑使用滚动条，包括水平滚动条和垂直滚动条。首先介绍 一下元件图形的绝对坐标和相对坐标的概念。相对坐标是指元件所在的矩形框在计算机屏幕上的坐标，它以计算机屏幕左上角为零点，拉动滚动条该坐标需随之改动。绝对坐标是指以整张图纸的左上角为零点的元件所在的矩形框的坐标，拉动滚动条该坐标不会改变。滚动条控件有一个Position属性，该属性记录了滚动条滚动的距离。所以，绝对坐标减去滚动条的Position属性，就是相对坐标。

在绘图窗体的右边界和下边界时，分别添加竖直和水平滚动条。实现该功能的重画函数Refresh（）被TscrollBar控件的OnChange事件调用来实现。

下面介绍实现这个功能的具体办法，主要代码如下：

```
void CLadderCanvas::Refresh (int nStartX, int nStartY,
                             int nWidth, int nHeight )
{
  //清除画布上次的内容
    m_pCanvas->FillRect (Rect (nStartX, n StartY, n EndX, n EndY) );
    While ( 遍历梯级链表)
    {
      …//获得梯级对象中的行链表
        While (偏历行链表)
        {
          //把绝对坐标转换为相对坐标
          pRow->SetLeft (nStartX) ;
          pRow->SetTop (nStartY) ;
          if (pRow->GetTop () >nEndY) //在计算机屏幕区域内的行才画
          break;
          …//获得行对象中元件链表
         while (遍历元件链表)
         {
          if (元件在计算机屏幕区域内)
            …//调用元件对象的绘图函数
            if (元件是否处于选中状态)
            {
              …//画蓝色的选中光标
            }
```

```
        }
      }
}
```

说明： nStartX 是水平滚动条的 Position 大小，nStartY 是垂直滚动条的 Position 大小，nWidth 是屏幕视窗的宽度，nHeight 是屏幕视窗的高度。

3.5.4 梯形图的存储

绘制完一个梯形图后，操作系统对存储在外部介质上的数据以文件为单位进行管理。存取数据时，要先建立一个文件，再对它输出数据；读取文件里的数据时，要先按文件名找到该文件，再对其进行操作。

在本课题中，创建了一种*.pr j 的文件，专门用来存取梯形图。保存文件时，必须保存其再次打开时需要的所有信息，所以，本课题采用结构化存储方式，在程序中定义了梯级、行、元件等对象信息的结构体，把它存到文件中，当要调用的时只需把文件里的数据读出来，再根据数据重建相应的对象即可[22]。编辑模块对梯形图的文件存储采用三级方式，由梯形图（Ladder）工程文件结构、梯级（Rung）文件结构、行（Row）文件结构组成。

第一级为梯形图（Ladder）工程文件结构，它由工程文件头、梯级索引表和梯级数据组成。工程文件头保存了文件标识、版本号、工程名、梯级数目等整个工程的描述信息。梯级索引表中记录了每个梯级数据在文件中的偏移地址及数据大小。梯级索引表后面存储的是每个梯级的数据，依次存放。

第二级为梯级（Rung）文件结构，用于存储一个梯级的数据信息，它由梯级文件头、行索引表和行数据组成。梯级文件头保存了行数目等梯级描述信息。行索引表中记录了每行数据相对于本梯级数据的偏移地址及数据大小。行索引表后面依次存储了本梯级上所有行对象的数据信息。

第三级为行（Row）文件结构，它由行文件头和元件数据存储区构成。行文件头记录该行包含的元件个数。元件数据存储区依次记录了每一个元件的类型、包含的线型、元件索引值、操作参数的名称和文字长度。实现文件的存盘功能，使用C++Builder的TFileStream类。该类采用流机制来读写文件，它可以用于存储大量的同一记录类型的

记录。另外，在梯形图的梯级类，行类，元件类当中都有一个名为save的成员函数，它的作用是将对象中的数据存储到定义的结构体当中去。程序通过遍历梯级链表、行链表和元件链表，依次调用链表中各对象的save方法，即依次把对象的数据存储到文件流当中去。这样不断重复，直到把链表中所有的对象都存储为止，从而实现存盘。下面以梯级类的存储函数为例，主要代码如下：

```
bool CLadder::Save (TFileStream *pms)
{
    int nCount=m_RungList.size ( ) ; //获得梯级数目
    pms->Write (&nCount, sizeof (nCount) ) ; //存储梯级数目
    //定义个迭代器，用于遍历梯级链表对象
    CRungList::const_iterator it=m_RungList.begin ( ) ;
    while  (it!=m_RungList.end ( ) )
    {
      CRung*pRung=*it;
      pRung->Save (pms) ; //调用梯级的save方法将数据存入流中
      it++; //存储下个梯级
      }
  }
```

工程文件的读取是存盘的逆过程，以读写方式打开文件后，再从头至尾读取文件流当中的每条记录，在这个过程关键是如何重建每个梯级、行、元件对象。如读到元件记录时，就根据该记录的第一个数据（元件类型属性值），可知该记录存储的是一个什么元件，从而调用生成一个该类的元件对象，并调用对象的
Open ( ) 方法把记录当中的数据赋值给元件对象，这样文件流中的每一个记录都将转换成一个对应的元件对象。并在每个对象生成的同时，将其加入到相应的行链表即完成读取过程。

3.6 本章小结

本章根据编程器的设计原则，研究了系统类层次结构和元件库对象，设计出Windows环境下的梯形图编辑模块，该模块的主要功能有：设计出了一个元件图形库，实现元件的操作功能，包括元件图形的绘制、选择、插入、删除等，并能检查错误，支持滚动，提供“撤消”和“重做”功能，支持梯形图中元件属性更改和梯形图文件存盘、读盘功能。

4 系统编译模块的开发

4.1 引言

编译器是“PLC集成开发平台”中最重要的部分，因为只有通过它们的“翻译”用户程序才能变成PLC硬件能理解的内容。任何符合规范的梯形图逻辑关系都可以通过树结构来表示，对这棵树进行遍历，就可以得到相应的PLC指令集。首先介绍梯形图到指令转化的算法，然后介绍梯形图编译的具体实现方法。

4.2 编译模块建模

梯形图形象，直观，对于熟悉继电器表示方式的人来说，非常容易接受，而且不需要学习更深的计算机知识，但不适合可编程控制器的CPU识别。指令表是一种用助记符形式表示用户程序的编程语言，已经接近于机器语言，能很容易变为CPU能识别的机器码，然而使用者用指令表编写PLC程序必须学习和记忆指令。

采用梯形图进行编程， CPU能很容易识别程序，将形象化的梯形图语言自动转化为指令表。但梯形图转换为指令表有一定的难度，因为在编辑梯形图时存储的是行与行之间的顺序以及每行中各元件的信息，并没有存储各个元件的逻辑关系，这将给梯形图转化带来一定的难度。

4.2.1 以树为中心的转换方法

梯形图是一种图形语言，而指令表则是一种类汇编的文本语言，要实现从梯形图到指令表的编译，得找出它们之间的本质联系。采用一个本质的方式表达PLC用户程序的逻辑关系，将有利于转换的实现。经过考察，发现数据结构中的“树”是比较合适的方案。树是一种最为常用的非线形数据结构，是以分支关系定义的层次结构。树是n个结点的有限集。在任意一棵非空树中：

（1）有且仅有一个特定的根结点；

（2）当n>1时，其余结点可分为m （m>0）个互不相交的有限集，其中每一个集

合本身又是一棵树，并且称为根的子树[20]。

直观地说，树是由1个根结点和若干棵子树组成，这若干棵子树每一棵都有1个根结点和它自己的若干棵子树，依此类推。用树的叶结点代表具体的元件，而用非叶结点则表示其左右子树的结合方式（串联/并联）且对应梯形图中的一个基本块，那么每个梯形图网络都可以表示成一棵树，而整个PLC用户程序就是一系列树的有序排列，也就是森林。

本程序以树结构为中介将梯形图和PLC指令联系起来，实现从梯形图到PLC指令的转换。采用树结构表达梯形图各元件间的逻辑关系后，对其进行一次后序遍历之后，就能得出其对应的用指令语言表示的程序。如图4-1中的梯形图的逻辑关系可以表示成图4-2的树结构，图4-2中的“AND”表示“逻辑与”，“OR”表示“逻辑或”。通过对这棵树设计遍历算法，就可以唯一得到相应指令集，如图4-3所示。图4-1，4-2，4-3表示了梯形图、树和指令表三者之间的对应关系。 所以，任何符合规范的梯形图逻辑关系都可以表示树结构，通过对这棵树设计遍历算法，可以唯一得到相应指令集。

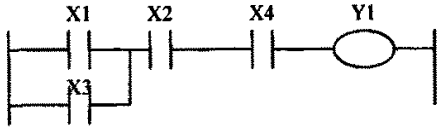

图4-1 梯形图示意图

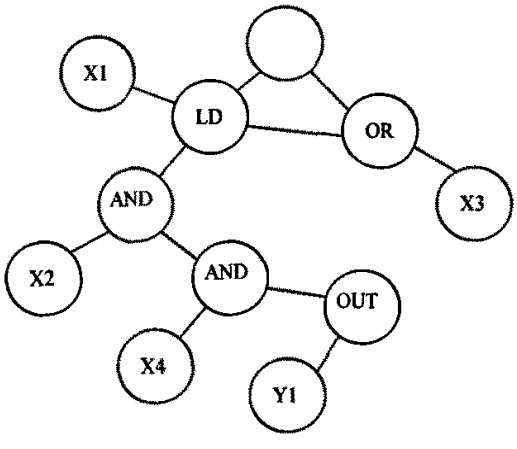

图4-2 梯形图转化的树

LD X1

```
OR  X3
AND X2
AND X4
OUT Y1
```

图 4-3 转化成功的指令表

4.2.2 逻辑树对象模型

对图 4-2 梯形图转化的树进行分析，逻辑树树结点有三种类型：根结点、逻辑结点和元件结点。根结点没有实际意义，元件结点代表梯形图中具体的元件，逻辑结点则表示其左右子树的结合方式（串联/并联）且对应梯形图中的一个基本块。

树结点除了表示本身信息以外，为了能遍历整棵树还必须包含父结点和子结点信息。采用双亲表示法，每个结点包含父结点的对象指针和子结点链表。其中，通过访问父结点指针对象，能获得该结点的父结点对象;在子结点链表中，每个结点对应该结点的一个子结点对象。由于根结点是没有父结点，所以根结点的父指针总是空的。元件结点没有子结点，所以元件结点的子链表总是空的。在这种表示法中，求任一结点的子结点很方便，只要遍历子结点链表就可找到全部子结点。

本论文充分利用面向对象方法的封装、继承等特点实现了逻辑树对象模型的建立，如图 4-4 所示。将结点类型 LDObject 视为基类，结点类型 LDRung，LDRungSeg 和 LDElement 视为派生类。其中 LDRung 指的是根结点，LDRungSeg 指的就是上面所说的 AND 结点或 OR 结点，而 LDElement 指的就是元件结点了。

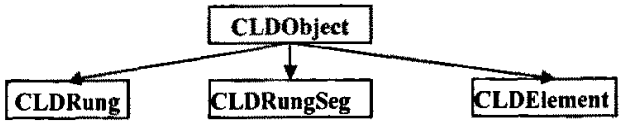

图 4-4 逻辑树对象模型

基类 CLDObject 是从逻辑树中所有树结点抽象出来的一个类，它定义了其它类的共有操作接口和属性。其定义描述如下：

（1）父结点对象指针：通过指针可以访问父结点对象。

（2）子结点链表：遍历链表可以获得结点的所有子结点。

（3）结点的逻辑类型：逻辑或和逻辑与。

（4）编译函数：实现树结点的遍历过程中转化为指令表。

（5）结点所在列：元件在梯形图网格中所在列。

基类主要代码如下：

```
class CLDObject
{
  protected :
    CLDObject *    m_p LDParent; //父结点指针
    CLDObjectL ist   m_LDObjectList; //子结点链表
    int   m_nColumn; //结点在梯形图中列号
    typedef   enum //枚举结点类型
    {
     LD _OBJECT_ ELEMENT,
     LD _OBJECT_ RUNG_ SEG,
     LD _OBJECT_ RUNG
     }ILD _OBJECT__TYPE;
     LD_ OBJECT_ TYPE   m_type; 区别为何种类型结点
     LD_ LOGIC_ TYPE   m_logicType; //逻辑类型
     Public:
    CLDObject（LD_ OBJECT_ TY Pe   type);
    Virtual   ~CLDObject（ );
    virtual bool Compile（OP__CODE__TYPE   ocType, int& nAddress,
    ArrayString*pstrProgram); //编译函数
    int   GetCount（void); //子结点的个数
    virtual   void   ResetData（ ): //清除结点
    virtual   void   Append（CLDObject*p LDObject); //设置父子关系
```

```
virtual  void  Merge（CLDObject*pLDObject)；//合并树
//删除结点的子结点
virtual  void  Delete（int  nStartCol，  int nEndCol)；
void  SetLogicType（ )：//设置逻辑类型
… …
}；
```

4.3 编译器的具体实现

梯形图编译器所要实现的主要功能是:将形象化的梯形图语言自动转化为指令表。在编译的过程中，对梯形图进行语法检查。如果梯形图不符合格式要求，则会给出错误信息。用户通过查看编译过程中所给出的错误信息，进行相应的修改，直到没有错误为止。符合规范的梯形图就会转换成相应的指令集。在由梯形图向指令表转化过程中，需要进行语法检查、梯形图到树的转变、由树产生指令几个步骤[22]，编译模块的流程图如图4-5所示。

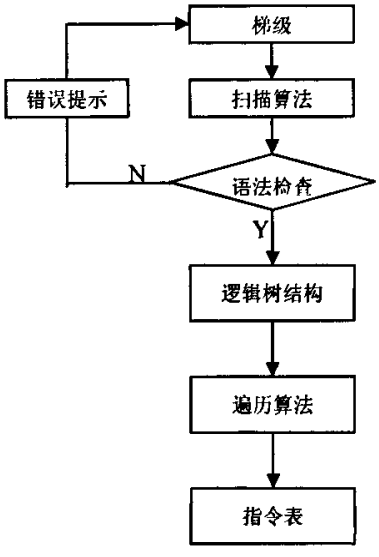

图 4-5 梯形图编译模块的流程图

从第三章中可知，编辑模块在存储梯形图时，使用的数据结构是链表，而不是现在

使用的树，所以，在编译梯形图程序的时候，第一步是梯形图进行扫描，将其转换树。

在编译梯形图的过程中同样需要进行错误处理：如各种继电器线圈不可直接与左边母线相连；输出线圈的右边不可放置触点，触点必须放在线圈的最左边等。

4.3.1 语法检查

用户编写和输入梯形图时，可能会出现不符合系统要求的问题，所以梯形图必须经过全面检查是否符合规范，只有在确定没有错误后，才能被转化为指令表。对用户梯形图编译时只进行语法和词法错误检查，而对逻辑错误不进行检查，用户程序的逻辑错误通过系统离线仿真进行检查、修改。

根据梯形图的特点，先检测每个梯级是否有输出线圈或者执行线圈，没有的话报警。其次检查是否有短路、断路情况的发生。短路的情况就是两并联线之间存在着直线直接相连的情况。检查时就是顺次检查两相邻并联线间是否有直线直接相连的支路，可以通过记录两并联线之间的元件个数来进行这一过程，若元件个数为零就是短路。至于断路，可以这样来判断：凡扫描过的元件都进行记录，扫描过程结束后若发现还有未扫描到的元件就判断为断路。

4.3.2 梯形图转化为树

梯形图转化为树的过程是“从上至下，从左至右”，以梯级为单位，采用“正向深度优先扫描算法”来正确有效地确定梯形图程序的扫描走向和读取顺序，建立构造逻辑树结构。“正向”就是从梯形图程序左侧母线连接的第一行的第一列开始按程序编写的格式，从左至右、从上到下的顺序读取扫描。“深度优先”指的是从梯形图内部逻辑关系入手，按照手工翻译的思想，首先编译模块内深层的逻辑关系，然后再编译模块间的关系。梯形图编译时以梯级为单位，梯级是相互影响的行组成的最小单元。

4.3.2.1 扫描函数

为了更方便论述构造树的过程，先介绍扫描过程中用到的函数及其功能：

（1） 函数 CompileRung（ ）：用来对梯级逐个进行扫描，具体对某个梯级进行扫

描时，从梯级的起始列开始逐列扫描到结束列。算法的描述如图 4-6 所示:

（备注：nLastCol 是函数 GetSegment（）所记下列数。Y 和 X 分别表示函数 GetRungSegment（ ）的参数 nPrevBStar 和&pnLastBStart，gpnLastBStart 作为函数的返回参数赋给 nPrevBStart，作为已知参数进入下一次循环。）

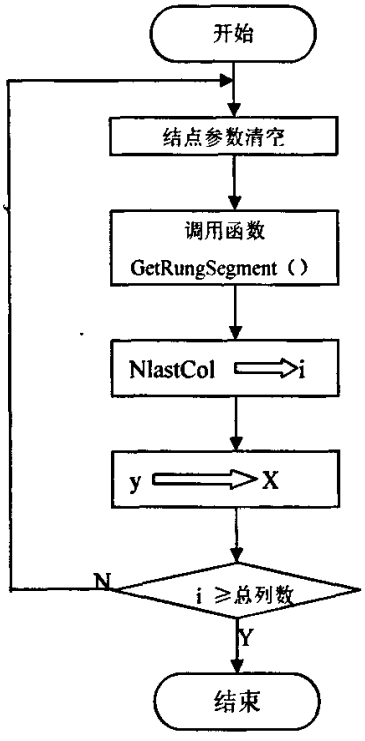

图 4-6 函数的流程图

（2） 函数 GetRungSegment（） 作为实现整个扫描过程的核心函数，包含以下几个函数:

①函数 G etSegment（ ）：从左至右依次扫描各个元件，如果找到含有向下线（LADDER_WIRE_ DOWN）的元件，返回当前所在列（nLastCol）和一个表明是否找到向下线的 bool 变量。

② 函数 GetConnectingRows（）：从当前行找到下一行的当前列（当前列为传入的参数）的前一列，判断元件类型是否为空（ELEMENT-TYPE-FREE）。为空的话则函数返回的参数 RowList 为空，不为空的话则递归向下扫描，判断再向下一行当前列的前一

列是否为空，直到为空为止，用分支链表 RowList 存储相关行的信息。

③ 函数 IsBranch ()：从当前列（当前列为传入的参数）向前扫描，是否有向上线（LADDER_ WIRE _UP）存在，如果含有向上线的元件的上一行恰好就是基准行（每个梯级的第一行），则该元件所在行就是分支，函数返回 bool 变量为 true，如果含有向上线的元件的上一行不是基准行就递归向上扫描（对应列索引号不变），再次判断所在行是否为基准行，倘若是基准行就是分支，用分支链表储存起来，不是的话就不是分支，函数返回 bool 变量为 false。

④函数 GetRungBranches ()：利用函数 IsBranch () 判断是否为分支，并将分支的信息（分支起始列，结束列以及行信息等）储存在分支链表中并返回。

（3） 函数 GetBranchSegment () 和函数 GetOutputSegment ()：这两个函数与函数 GetRungSegment () 的扫描过程基本相同。

4.3.2.2 扫描和构造树的过程

下面介绍函数 GetRungSegment () 的扫描和构造树的过程，具体的过程如下：

（1） 新建一个 AND 结点（LD_LOGIC_ AND），函数 GetSegment () 从左至右开始扫描，在未发现向下线之前，将一个个扫描过的元件都作为元件结点指定为该 AND 结点的子结点。如果扫描到向下线，立即跳出函数 GetSegment ()。

（2）调用函数 GetConnectingRows () 进行判断，若返回的参数 RowList 为空直接将结点 LDObject（结点 LDObject 为传入函数 GetRungSegment 的参数，具体为哪一个结点由函数 CompileRung 传出的参数决定）替代 AND 结点，即结点 LDObject 成为元件结点的父结点，继续向后扫描，返回步骤（1）。若返回的参数 RowList 不为空则执行步骤（3）。

（3） 从行链表（RowList）中逐个读出存储的信息，调用函数 GetRungBranches () 判断是否为分支，倘若为分支，获取分支的信息，用分支链表（BranchList）存储。

（4） 调用函数 GetBranchSegment ()：扫描分支，新建一个 OR 结点（L D__LOGIC__OR）。

扫描分支时，为了解决分支与分支之间的包容问题，将对分支的扫描分成如下几种

情况区别对待：

①分支的起始列小于上一分支的起始列参数 nPrevBStart；

②分支的起始列等于上一分支的起始列参数并且分支的起始列小于上一分支的参数 nLastCol。

③ 除去以上两种情况以外的情况。

如果分支的情况符合①②，将根结点以下的位于分支起始列和结束列之间的所有已建立的结点全部删掉，新建一个 AND 结点，将该 AND 结点作为函数 GetBranchSegment（）的实际参数（LDObject 为形式参数）输入，对该分支进行扫描，该分支所有扫描到的结点都将归于该 AND 结点之下，将该 AND 结点指定为根结点的子结点并将 OR 结点指定为 AND 结点的子结点，然后将 OR 结点作为函数 GetBranchSegment（）的实际参数，该分支作为函数 GetBranchSegment（）的主分支参数，再次对位于分支起始列和结束列之间的整个电路进行扫描，扫描到主分支时直接跳出。

如果分支的情况符合③，直接 OR 结点指定为 AND 结点的子结点，然后将 OR 结点作为函数 GetBranchSegment （）的实际参数，对位于分支起始列和结束列之间的整个电路进行扫描，各结点均归于 OR 结点之下。

4.3.2.3 应用实例

以图 4-7 所示的梯形图为例，具体谈谈构造树的过程。

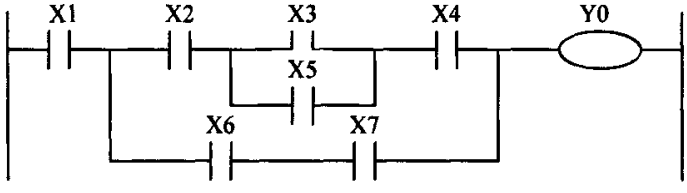

图 4-7 梯形图示例

第一步：扫描元件 X1，执行 4.3.2.2 中的（1）和（2），并返回（1），

第二步：扫描元件 X2，执行 4.3.2.2 中的（1）和（2），并返回（1）。

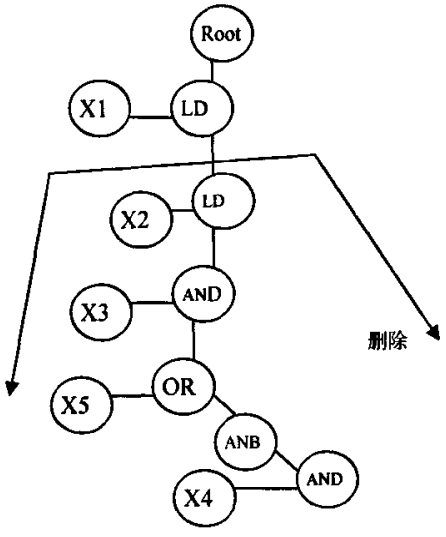

图 4-8 结点的删除

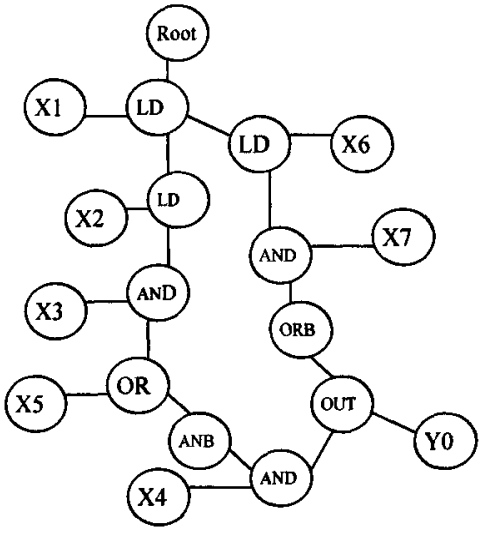

图 4-9 完整的树结构

第三步：扫描元件 X3 和 X5，执行 4.3.2.2 中的（1）完成对 X3 的扫描，接着执行（2），（3），（4），完成对 X5 的扫描，此时根据在函数 CompileRung（）中的 GetRungSegment（ ）中预先输入的参数，X5 所在的分支属于（2）中的第③种情况，返回（1）。

第四步：扫描元件X4，X 6和X7，执行4.3.2.2中的（1），完成了对X4的扫描，接着执行（2），（3），（4），发现X6和X7所在分支属于（2）中的第①种情况。

第五步：扫描输出元件Y0。在上面的过程4.3.2.2中的（4）中，扫描到X6和X7所在分支时，有一个将树的部分结点删除再重建的过程，删除部分如图4-8所示，扫描此梯形图最终所得到的构造树如图4-9所示。

4.3.3 树转化为指令表

把符合规范的梯形图逻辑关系表示成树结构后，需要设计树的遍历算法，来实现树到指令集的转换。树的遍历算法采用的是递归法，从根结点的子结点链表开始，依次调用链表中各对象的Compile方法。当对象为元件结点时，重载LDElement类型的编译函数，根据元件的类型和逻辑关系查表生成对应的指令；当对象为逻辑结点时，重载LDRungSeg类型的编译函数，遍历完逻辑结点的子结点链表后，最后把逻辑结点生成对应的指令。这样不断重复，直到把链表中所有的对象都编译为止，从而实现树到指令的转换。

（1）基类CLDObject的编译函数主要代码如下：

```
bool CLDObject：Compile（OP_ CODE_ TYPE  ocType，  int& nAddress，
                        ArrayString * pstrProgram ）
{
 //定义个迭代器，用于遍历子结点链表对象
 CLDObjectList::const_iterator  it =m__LDObjectList.begin（）；
while（it!=m__LDObjectList.end（ ））//遍历本结点的所有子结点对象
 {
  //派生类重载编译函数
   （*i t）  —> Compile（ocType，nAddress，pstrProgram）；
  it+ +；//编译下个子结点
  ocType=AND；//与上个子结点的逻辑关系
 }
}
```

（2）类 LDRungSeg 的编译函数主要代码如下：

```
bool CLDRungSeg::Compile（ ）
{
   //递归调用 CLDObject 的编译函数来遍历本结点的子结点链表
   CLDObject::Compile（ocType, nAddress, pstrProgram）;
   if（包含子结点数个数>1）
   switch（m_logicType）/逻辑树结点生成对应的逻辑指令
   {
     case LD_ LOGIC_ AND:
          strText= "ANB": //产生 ANB
          break:
     case LD__LOGIC_OR:
     ... ... //产生 ORB
     }
   pstrProgra m— > pushesback（strText）; //把生成的指令加入字符串数组
  }
```

（3）类 CLDElement 的编译函数主要代码如下：

根据元件结点的元件类型（m_elemType）和逻辑关系（ocType）查询对应的指令集表，获得对应的指令语句。

```
bool  CLDElement::Compile（OP__CODE__TY PE  ocType,  int&  nAddress,
                          Array String * pstrP rogram ）
 {
  switch（ ocType）
  {
  case LD:
       //产 生 LD 系列指令
       strLine =Format（"%-8d%—los%-IOs%—lOs",  （nAddress,
```

```
                    Get  ContactOpCode （ocType , melem Type） ,
                    mstrName. c_ str（ ） ， m__st rParamName.c _s t r （））；
        break；
        case OR：
              …… //产 生 OR 系 列 指 令
        case AND：
                  // 产 生 AN D 系 列 指 令
        }
        pstrProgram— > pushesback（strLine）；//把生成的指令加入字符串数组
    }
```

4.3.4 编译结果

用户使用本软件编写 PLC 程序时，先根据控制要求绘制出梯形图，再由梯形图转换成对应的指令表。图 3-6 就是利用本 PLC 梯形图编程软件绘制的一个 PLC 梯形图实例及编译后生成的指令表。

4.4 本章小结

本章主要介绍了系统编译模块的开发，以树结构为中介将梯形图和PLC指令联系起来，对梯形图到指令语言的转换算法进行了研究，采用树结构的遍历算法，将形象化的梯形图语言自动转化为指令表。

5 PLC的仿真实现

5.1 引言

PLC 仿真模块实现了 PLC 指令解释器的仿真，提供了模拟 PLC 输入输出的工具，模拟了 PLC 运行过程。总之，仿真模块实现了系统的离线仿真，可以对用户 PLC 程序的逻辑错误进行检查、修改。首先介绍 PLC 工作原理与 PLC 指令解释器的实现算法，然后介绍梯形图仿真模块的具体实现和仿真结果。

5.2 PLC 工作原理

PLC 开始工作后，它先经过输入采样，把现场所有信息（包括输入数据和一些中间输出数据）都放入输入映像区。然后扫描执行用户程序，执行用户程序所需的所有现场信息都从输入映像区取用，而不直接到外设去取；对于被控对象的控制信息，也不是形成一个就输出一个，而是把它们先放到输出映像区。最后当扫描结束后，将所有被控对象的信息集中输出，称这个阶段为输出刷新，并以此改变被控对象的状态。对于那些在一个扫描周期内没有变化的变量状态，就输出一个与前一周期相同的信息，因而不会引起外设工作的变化。I/0 映像区的建立，使 PLC 工作时只与内存有关地址单元内的所有信息状态发生关系，而系统输出也只是给内存某一地址单元设定一个状态。在图 5-1 中详细地描述了一个扫描周期内 PLC 工作的过程。PLC 在扫描执行程序一个扫描周期后，接着开始第二次、第三次，……，第 n 次执行程序，直至停机。

P LC 模拟仿真就是仿照硬 PLC 工作原理设计的。它既要有输入采样，还要有输出刷新，更要体现循环扫描，当然，正确执行程序是第一位的。现在将整个的流程图总结如图 5-1 所示[23]。

（1）输入采样

输入采样是 PLC 模拟运行的第一步。本课题中，建立了一个输入文件，把它作为

输入映像区。此文件存放着现场的所有输入信息，其中的数据随着现场信息的改变实时改变。输入采样数据就是从这个文件得到，然后用这些变化了的信息不断替换原来的信息，这样就可以得到不同的输出，进而进行逻辑控制。

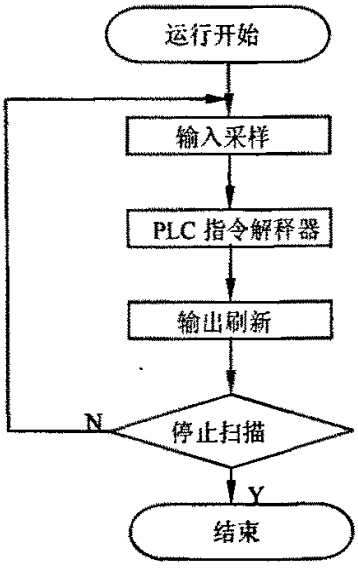

图 5-1 PLC 模拟流程图

（2） 程序执行

要实现不同的逻辑控制，必须首先执行用户程序得到正确的输出结果。根据输入映像区中存储的信息来执行用户程序。根据 PLC 工作的原理，用户程序执行的结果要放到输出映像区里，最后集中进行输出刷新。为此，构造一个输出映像区，用它来存放各输出元素状态。

（3）输出刷新

当执行完用户程序后，PLC 就进入了输出刷新阶段。本课题中，以输出文件作为输出映像区，所有被控对象的控制信息都从此文件读出，进行输出刷新，改变输出模块各地址位的状态。

（4）循环扫描工作方式

硬 PLC 是按照循环扫描方式工作的，软 PLC 也是按照这种工作方式工作的。只有这样，才能把 PLC 控制系统对被控对象控制的实现是相关逻辑关系的实现这一点体现

出来。

5.3 PLC 指令解释器的实现算法

PLC 指令是一种类似于汇编语言的助记符编程表达式，易计算机处理。每条指令由序号、操作码和操作参数组成。如 LD X0 中 LD 就是操作码，表明要执行的指令，X0 就是操作参数，指出信息来源。PLC 指令解释器用于对 PLC 指令的程序逻辑错误进行检验，并对 PLC 运行过程的仿真起指挥和控制作用。下面介绍了如何将由逻辑指令和元件组成的 PLC 程序在 PC 机中实现解释执行。

首先看一段简单的 PLC 程序，见程序清单：

（1） LD X4

（2） OR X6

（3） ORI M102

（4） OUT Y5

（5） LDI Y5

（6） AND X7

（7） OR M108

（8） ANI X10

（9） ORI M110

（10） OUT M103

（11） END

定义 OR，ORI，ORB 操作为加法“+”，AND，ANI，ANB 操作为乘法“*”取反操作为“[]”。则上述程序清单中的输出 Y5 和 M103 分别等于：

Y5= X 4+ X6+ [M102]

M103=（[Y5]* X7+M108）*[X10]+[M110]

上式的所有运算符只会连接两个变量或者一个变量和一个表达式。由于扫描过程是顺序执行的，因此可以为上述计算过程建立计算堆栈。建立递增的堆栈，则计算过程和

堆栈中的数据变化见表 5-1：

表 5-1 PLC 解释器堆栈数据变化

步数	指令	栈顶内容	栈顶指针	操作说明
1	LD X4	X4	0	X4 压入堆栈
2	OR X6	X4+X6	0	栈顶元素弹出与 X6 相加，结果压入堆栈
3	ORI M102	X4+X6+[M102]	0	栈顶元素弹出与[M102]相加，结果压入堆栈
4	OUT Y5	X4+X6+[M102]	0	栈顶元素赋值 Y5，堆栈状态和内部数据不变
5	LDI Y5	[Y5]	1	将 Y5 压入堆栈
6	AND X7	[Y5]*X7	1	栈顶元素弹出与 X7 相乘，结果压入堆栈
7	OR M108	[Y5]*X7+M108	1	栈顶元素弹出与 M108 相加，结果压入堆栈
8	ANI X10	（[Y5]*X7+M108）*[X10]	1	栈顶元素弹出与 X10 相乘，结果压入堆栈
9	ORI M110	（ [Y5]*X7+M108 ）*[X10]+[M110]	1	栈顶元素弹出与[M110]相加，结果压入堆栈
10	OUTM103	（ [Y5]*X7+M108 ）*[X10]+[M110]	1	栈顶元素赋值 Y5，堆栈状态和内部数据不变
11	END		空	

观察表 5-1 中的处理过程可以发现每一个输出只占了一个堆栈寄存器。由此软 PLC

程序可定义如下的计算法则：

（1）出现 LD 或 LDI 指令，将指令的操作数压入计算堆栈的栈顶。

（2）出现 OR，ORI，AND 和 ANI 指令，运算符的两个操作数一个来源于堆栈，另一个来源于指令之后跟随的操作参数。将计算堆栈栈顶元素弹出与指令操作参数进行相加或相乘运算，结果重新压入栈顶。

（3）出现 OUT 指令，栈顶元素的值赋给指令操作参数。现在考虑指令中出现块的逻辑操作指令时的情况。块的逻辑操作程序见程序清单：

（1） LD X0

（2） OR X1

（3） LD X2

（4） AND X3

（5） LDI X4

（6） ANI X5

（7） ORB

（8） OR X6

（9） ANB

（10） OR X7

（11） OUT Y6

（12） END

在这一段程序中，输出可以用输入按照如下算式计算得到。

Y6= （X0+X1）*（X2*X3+[X4]*[X5]+X6）+X7

观察上式，发现情况复杂了一些，例如第一个出现的乘号和第二个出现的加号，他们连接的分别是两个计算表达式。仍然采用计算堆栈，运算过程见表 5-2 所示。

表 5-2 PLC 解释器堆栈数据变化

步数	指令	栈顶内容	栈顶指针	操作说明
1	LD X0	X0	0	X0 压入堆栈
2	OR X1	X0+X1	0	栈顶元素弹出与 X1 相加，结果压入堆栈
3	LD X2	X2	1	X2 压入堆栈
4	AND X3	X2*X3	1	栈顶元素弹出与 X2 相乘，结果压入堆栈
5	LDI X4	[X4]	2	[X4] 压入堆栈
6	ANI X5	[X4]*[X5]	2	栈顶元素弹出与[X5]相乘，结果压入堆栈
7	ORB	[X4]*[X5]+ X2*X3	1	连续弹出堆栈中两个元素，并将它们相加，结果压入堆栈
8	OR X6	[X4]*[X5]+X2*X3+X6	1	栈顶元素弹出与 X6 相加，结果压入堆栈
9	ANB	（X0+X1）*（X2*X3+[X4]*[X5]+X6）	0	连续弹出堆栈中两个元素，并将它们相乘，结果压入堆栈
10	OR X7	（X0+X1）*（X2*X3+[X4]*[X5]+X6）+X7	0	栈顶元素弹出与 X7 相加，结果压入堆栈
11		（X0+X1）*（X2*X3+[X4]*[X5]+X6）+X7	0	栈顶元素赋给 Y6
12			0	空

在处理块操作逻辑指令的时候，操作符的两个操作参数均来自堆栈，因此要将最后进栈的两个元素弹出用于计算。这样，增加软 PLC 程序的计算法则：

（4） 出现 ANB 或 ORB 指令，将最后进栈的两个元素弹出堆栈，进行加法或乘法运算，结果重新压入栈顶，对于 增 加 了更复杂线圈和指令的 PLC 程序可以按照这种类似的方法进行解释执行。

5.4 仿真程序的具体实现

PLC 模拟仿真是在计算机上仿照 PLC 的工作原理，以梯形图转化的指令程序为输入，逐行解释 PLC 指令的方式执行，来实现 PLC 运行过程的仿真。根据 PLC 指令解释器的实现算法，依次执行各行指令即可遇到诸如程序段串并联的时候，可采用堆栈的方式保存每个程序段运行时的中间状态。例如，遇到 LD 指令进栈，ANB, ORB 或 OUT 指令出栈。根据上面的分析，设计 PLC 仿真模块的执行流程如图 5-2 所示。模拟仿真的基本思路如下:

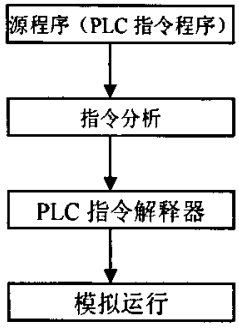

图 5-2 仿真模块的流程图

1）将输入信号读入输入映像寄存器。

2）读取一行 PLC 指令，提取出指令字和信号（如果有的话）。

3）根据指令字执行相应的功能或功能块。

4）如果还有 PLC 指令，则返回 2），否则进入 5）。

5）将输出映像寄存器写到实际输出信号。

6）一个运行周期完成，返回 1），直到 PLC 程序被人为停止。

5.4.1 虚拟 I/0 映像寄存器建立

由于 PLC 与继电器控制电路相比，以软器件代替了硬器件，以软触点代替了硬触点，以软接线代替了硬接线。元件的符号地址反映实际的 I/0 模块中的某个地址所对应的数据。

为了实现系统的离线仿真，需要建立一个虚拟 I/0 映像寄存器。用虚拟 I/0 模块来模拟各个数据位状态的改变，把元件的符号地址和虚拟地址相对应，能用符号地址操作虚拟地址。所以，为 PLC 每一个触点资源都分配一个对应的元件数据结构，在内存中建立了模拟 I/0 映象区。元件数据结构表示如下：

```
type defstruct
{
    bool   m_fOutput; //元件的输出状态
    bo l   m_fLastInVal; //元件的当前状态
    bool   m_fReset; //是否重新计时/计数
    bool   m_fActive; //元件是否激活
    long   m _1Pres etAmem; //预设值
    long   m_ lAmem;  //元件实际地址的当前值
}P LC _E L EMENTS, * PPLC_E LEMENTS;
```

虚拟 I/ 0 映像寄存器定义如下：

PLC_ELEMENT m_Elements [ELEMENT_SIZE]。

其中：ELEMNT_ SIZE 为 PLC 触 点资源的总个数。

定义一个“堆栈寄存器”用来保存每个程序段运行时的中间状态，定义如下：

```
bool   m_fStack [300];
```

5.4.2 指令分析

指令分析主要完成如下工作:先输入PLC指令程序后，以指令行为单位，对构成指令行的字符串进行扫描和分解，识别出指令行编号，指令操作码，操作参数一和操作参数二，然后在操作码数据表和操作参数数据表中进行查询。在查询过程中分析指令的操作码是否存在，操作数代码是否符合操作码的要求和操作数是否在操作数代码的允许范围之内。如果指令的操作码不存在或操作数不在操作数代码的允许范围之内，程序就会报错。如果找到对应的索引号，就会生成一个指令行类对象存入操作码，操作参数及对应表中的索引号信息，同时程序把该对象加入到指令行链表当中。这样不断重复，以完成

整个指令程序的指令分析。其中，生成的指令行链表是后面模拟仿真运行的基础。

对PLC指令进行分析，得到如下结论：

（1） LD, LDI, AND, ANI, OR, ORI等指令操作码只有一个操作参数。

（2） LDLT, LDGE等指令操作码有两个操作参数。

（3） Timer指令中操作参数“KXXX”，"K”为标识，"XXX” 为预设值。

所以，每行PLC指令代码由指令行编号、指令操作码、操作参数一和操作参数二组成。指令的操作参数一和操作参数二的值分两种情况:对应工/O模块中的某个触点地址和对应某个常量值。根据面向对象设计方法的抽象与继承机制，把每行PLC指令代码抽象成一个指令行类，类中应该包含以下信息：

（1） 指令行编号

（2 ）指令操作码及其索引号

（3 ）操作参数一及其索引号

（4 ）操作参数二及其索引号

建立PLC指令程序行类关键代码如下：

```
class CProgramLine
{
public:
  CProgramLine ();
  Virtual~CProgramLine ();
  long m_lStepNum; //指令行号
  AnsiString  m_szMnemonic://指令操作码
  Short  m_slnstruction; //指令操作码的索引号
  bool  m_fParml; //操作参数一
  long  m_lParml;
  bool  m_fParm2; //操作参数二
  long  m_lParm2;
};
```

当操作参数为I/O模块中的某个触点地址，布尔型变量m_fParm1为False,变量m_1Parm1为操作参数一就是对应操作参数表中的索引号;当操作参数为某个常量值时，布尔型变量m_fParm1为True，变量m_1Parm1为对应值。当指令语句只有一个操作参数时，布尔型变量m_fParm1为True，变量m_1Parm1为0.指令分析由创建操作码数据表、操作参数数据表、查询过程组成，具体过程如下:

（1）创建操作码数据表主要代码如下:

```
void CPLCCtrl::LoadMnemonics ()
{
    ArrayString*m  Mnemonics=new ArrayString; //创建存储操作码的字符串数组
    //依次把操作码加入到字符串数组
    m_Mnemonics->Add ("LD") ;  刀LD系列常开触点索引号为。
    M_Mnemonics->Add ("LDI") ;  //LD系列常闭触点索引号为1
    M_Mnemonics->Add ("AND") ;
    M_Mnemonics->Add ("ANI" ) ;
    … …
    //加入其他操作码
    //AND系列常开触点索引号为8
    //AND系列常闭触点索引号为9
   }
```

（2）创建操作参数数据表主要代码如下:

```
bool CPLCCtrl::LoadProgram (TStringList *strProgram)
{
  ArrayString *elements=new ArrayString//创建存储参数的字符串数组
  AnsiString sbase; //定义操作数代码
  AnsiString sindex; //定义操作数
  sbase="x"; // 操作数代码为 "X"
    for ( index =0;  index<XLEN;  index++) //其中：XLEN为操作数代码X的允许范
```

```
围
    {
     elements->Add (sbase+sindex) ; //把操作参数加入到字符串数组
     }
     …… //加入其他操作数代码及范围
  }
```

(3) 查询过程主要代码如下:

```
  for (int nLineNum=0;  nLineNum<nNumProgLines;  nLineNum++)
{
  pProgLine=new CprogramLine; //创建一个指令行对象
  s=strProgram->Strings[nLineNum]: //获得指令行
  //字符串进行扫描和分解
  token[0]=s.SubString (1,8) .TrimRight () ; //获得指令行号
  …… //获得指令操作码
  .… //获得操作参数一
  …… //获得操作参数二
for (int i=1;i<5;i+ +)
    switch (i)
    {
      case1 : // 1为指令行号
      pProgLine -> m_lStepNum=token[i].ToInt () ;
      ……
      case 2: //2为指令操作码
      index = m_Mnemonics->IndexOf (token[i]) ; //查表获得索引号
      ProgLine->m_szMnemonic=token[i]:
      pProgLine->m_sInstruction=index;
      ……
```

```
case 3: //3为操作参数一
{
  if (token[i].Su bString (1,1) ==”K”)
    {
      //操作参数一的第一个字符为K时，其后字符串表示常量
      rest=token[i].SubString (2,token[i].Length () —1) ;
      pProgLine->m_1Parml=rest.ToInt () ;
      pProgLine->m_fParml=TRUE;
      breaK;
    }
    else
    {
      //操作参数为I/0模块中的某个触点地址
      index=elements->IndexOf (token[i]) ; //查表获得索引号
      pProgLine->m_fParml=FALSE;
      pProgLine->m_1Parml=index;
      break;
     }
  }
    case 4: //4为操作参数二
    …… //同操作参数一的处理
   }
   m_Program.pushesback (pProgLine) ; //该对象加入到指令行链表
}
```

5.4.3 PLC指令解释器实现

(1) 输入采样:把模拟仿真中对应的输入元件的数据传给虚拟工/0模块中的输入触点。

```
for ( int ulElemIndex=0;  ulElemIndex<8; ulElemIndex++)
{
 //把输入元件的输出状态传给虚拟工/0模块中对应的输入触点
 m_Elements[XINDEX+ulElemIndex].m_foutput
 =Forml->GetInputData (ulElemIndex) ;
//把输入元件的当前值传给虚拟I/0模块中对应的输入触点
m_Elements[TINDEX+ulElemIndex].m_lAmem
=Forml->GetAIData (ulElemIndex) ;
}
```

(2) PLC指令解释器:实现由逻辑指令和元件组成的PLC程序在PC机中解释执行。主要代码如下:

```
switch (m_Program[nLineNum]->m_sInstruction)
{
 case 0: //0为LD指令索引号
 {
   if (m_Program[nLineNum]->m_fParml==false)
  {
   //通过操作码的索引号获得对应1/0映射区中元件的输出状态
   btemp=m_Elements[m_Program[nLineNum]->m_lParml].m_foutput;
   }
 m_fStack[bstack index]=btemp; //指令的操作数压入计算堆栈的栈顶
 bstack_index++;
 ......
 }
 ......
 case 16://16为OR指令索引号
 {
```

```
 ......
//堆栈栈顶元素弹出与指令操作数进行相加
 m_fStack[bstack index-1] |= btemp
}
......
case 28://28为OUT指令索引号
{
 //栈顶元素的值赋给指令操作数
 m_Elements[m Program[nLineNum]->m_lParml].m_foutput=
                                    m_fStack[bstack index-1];
 ......
}
 ......
case 25:// 25为ANB指令索引号
{
  bstack index--; //最后进栈的两个元素弹出堆栈
  m_fStack[bstack_ index-1]= (m_fStack[bstack_index-1]&&
  m_fStack[bstack_ index]) ://进行加法后，结果重新压入栈顶
  ......
   }
   ......
}
```

(3) 输出刷新:扫描虚拟I/O模块中的输出触点地址所对应的数据，传给模拟仿真中对应输出的元件。

```
for (int ulElemIndex=0;u lElemIndex<8;ulElemIndex++)
    For ml->SetOutputData (ulElemIndex,
        mElements[YINDEX+ulElemIndex].m_fOutput)
```

（4）循环扫描工作方式

本课题中 ，用C++B uilder提供的TTimer控件来实现循环扫描。程序开始运行时，用函数SetTimer0设置定时器，然后在程序结束时用函数KillTimer（）消除定时器。循环扫描在OnTimer（）函数里实现。只要定时器不取消，PLC程序就一直在循环执行，输出结果在实时变化。

5.5 仿真结果

当按下“Load”按钮调入要模拟运行的PLC指令程序，再按下“Start”按钮时，PLC系统开始仿真。仿真时，运行系统不与I/O接口板和CNC通信。只是从界面上的输入框获得输入值，经处理后，把输出值输出到界面上的输出框。当PLC程序编程者在没有确定PLC程序运行结果之前，不想对I/O接口板进行操作时，就可采用仿真功能[26]。

仿真结果的正确性通过输入输出继电器指示灯来进行判断。

通过系统的离线仿真，可以对用户PLC程序的逻辑错误进行检查、修改。

5.6 本章小结

本章通过对PLC工作原理的分析，进行了PLC程序仿真技术和算法的研究，具体实现了PLC指令释器实现算法的仿真，模拟了PLC的运行过程。并且通过仿真系统的离线仿真，可以对用户PLC程序的逻辑错误进行检查、修改。

6 总结与展望

6.1 研究总结

PLC是数控机床上联系机床和NC的桥梁和纽带，其用户软件开发平台已经成为PLC系统中重要的组成部分。本课题的主要任务是开发出一个基于个人普通计算机的梯形图编程软件，为数控系统提供一个直观、方便、高效的编程环境。从数 控 PLC编程系统的特点出发，针对目前PLC梯形图编程软件的缺陷和不足，采用C++ Builder 5.0集成开发环境和面向对象的方法，开发了PLC梯形图的编辑环境，实现了梯形图到PLC指令代码的转换，并设计了仿真部分，弥补了原有同类型软件的不足。本系统集编辑模块、编译模块、模拟仿真模块等功能模块为一体，提供以图形输入方式来建立梯形图程序。要完成这样一个目标是需要进行大量的工作的，经过一年多的努力，作者主要完成了以下工作：

（1）项目的需求分析和总体设计，内部数据结构的详细设计；

（2）用户界面的设计与实现；

（3）梯形图编辑器的设计与实现；

（4）梯形图到指令语言的转换算法的设计与实现；

（5）用户文档管理的设计与实现；

（6）用户程序的仿真功能；

（7）安装文件的制作。

6.2研究展望

从软件的最后运行效果来说，该软件的功能基本上达到了一个PLC软件开发平台所应具有的最基本的一些要求，能满足用户编辑梯形图和逻辑仿真的功能，达到了较好的效果。但由于时间和能力所限，本软件离“为用户提供一个直观、方便、高效的PLC软件开发平台”的目标还有一定的差距，还需要下届研究生的继续努力。软件的测试结果显示平台的总体设计是比较合理的，因此以后的工作重心应该更多地放在软件的测试稳定运行和具体功能的改进上，后续迫切需要解决的问题如下：

（1）梯形图到指令语言的转换算法的优化；

（2）编译器的错误处理能力；

（3）用户程序的调试功能；

（4）仿真功能进一步完善。

致 谢

本学位论文是在导师唐小琦教授的亲切关怀和悉心指导下完成的。唐老师严峻的治学态度、广博的知识和丰富的实践经验使我受益匪浅，并将激励我终生奋发向上。在此期间，还受到周向东老师的悉心指导，我在学习和科研中取得的每一点成绩都包含了唐老师和周老师的心血，在此向两位老师致以真诚的谢意和崇高的敬意。

最感谢的是老师们严谨的治学态度和奋斗不息的工作作风，它们将指引我一生的工作和学习，支持着我奋斗不息！

参考文献

[1] 蒲志新，谷艳丰，康文龙.PLC在数控机床控制系统中的应用.机床与液压，2004（12）:158—159

[2] 李左章，周云飞，胡建中一种基于IPC的内嵌式PLC的实现方法.机械与电子，2000（5 ）:3-6

[3] 夏燕兰.PLC在数控机床上的应用.南京工业职业技术学院学报，2002 （6） :16-18

[4] 罗华丽，李斌等.开放式数控系统中的软件PLC技术研究.组合机床与自动化加工技术， 2003（2） :38-43

[5] 吴孜越，刘陆群，吕战争.基于CNC系统的嵌入式PLC组件的设计与实现.河南科技大学学报（自然科学版），2005 （6）: 18-21

[6] 游华云，叶佩青，杨开明，汪劲松.基于RTLinux的软件PLC的研究与开发.计算机工程与应用，2002 （22）:134-136

[7] 吕俊白，施敏芳.PLC梯形图可视化编辑与语句表的自动生成.自动化仪表 2005（3）:28 一 3 0

[8] 余明心，吴明哲等（著）.Borland C++ Builder 6程序设计经典.科技出版社，2000（1 ）

[9] Robert Lafore（著），邓子梁，胡勇（译）.C++面向对象程序设计.中国电力出版社，2004（2）

[10] [10]卢艳军，任立义，闻邦春基于软件PLC的I/0控制研究.机械制造，2005 （7）: 37—39

[11] 蒲志新，熊永超，熊晓红.PLC梯形图语言编辑功能的软件实现.机械2003（3 ）:54-55

[12] 张汉兵，叶伯生，杨道善.数控系统中内嵌式PLC梯形图编程的软件实现.组合机床与自动化加工技术，2001（ 12）: 26-28

[13] 谭锦洁，程良鸿.嵌入式PLC梯形图的一种数据结构描述方法计算机工程，2004（5）:85- 87

[14] 谭锦洁，程良鸿，殷学鹏，嵌入式PLC中梯形图到AOV图的映射，计算机测量与控制，2004 （ 12） :993-995

[15] 崔小乐，周卓岑.可编程控制器的梯形图语言与语句表语言的互换算法.微电子学与计算机 ， 2000 （1）: 26-30

[16] 周峰，王新华等.软PLC编辑系统的设计与实现.计算机工程程与应用，2005 （7）: 111—113

[17] 张运波.PLC梯形图设计中的关键技术.长春工程学院学报，2001 （1） :30-32

[18] 陈营.基于AutoCAD的PLC梯形图的绘制.潍坊学院学报，2002 （4） :75-79

[19] [19」王俊梅，王建军等.面向对象组态软件流程图CAD的开发与实现.工业控制计算机，20 00 （. 3 ）: 25-27

[20] 严蔚敏，吴伟民（著）.数据结构.清华大学出版社，2001 （2）

[21] 蒲志新，熊永超，李赢.PLC梯形图编辑功能的软件仿真机床电器，2003 （2）:37-40

[22] 董朝霞，陈青华，范斗.电力仿真系统图形编辑器面向对象的设计与实现.继电器，2002（6 ）: 4 0— 42

[23] 赵建东，王广炎，王小椿梯形图编辑系统的设计.机械与电子，2000 （4） :41-44

[24] 苏宏田，王秀玲.二叉树理论在配电网潮流计算中的应用内蒙古电力技术，2001 （1）:18 — 1 9

[25] 张承瑞，程荣亮，王恒.面向对象技术在软件PLC编译器中的应用.计算机工程，2004 （2 ）: 7 6— 7 7

[26] 赵建东，王广炎等.智能化嵌入式可编程控制器集成系统的研制.制造技术与机床，2002 （ 12）: 41-44

[27] 张礼兵，杨文通等基于Visual C++6环境的软件PLC的研究与开发.机床与液压，2004 （1 1） : 15 8 -166

[28] 郝用兴，华林，松下可编程控制器模拟仿真软件设计.微计算机信息，2005 （7）: 3-6

[29] 朱文凯，王卫华，丁汉，熊有伦基于嵌入式PC的开放式软PLC.机械与电子。2002 （3 ）: 3 — 7

[30] 马斌.基于WINDOWS系统的PLC模拟实验台的开发重庆工商大学学报（自然科学版），2004 （ 4 ）:206-208

[31] 张礼兵，吴婷.基于软PLC编译系统目标代码生成的研究与实现.微计算机信息，2005（7 ）:85—86

[32] 白艳艳.基于SERCOS接口的开放式数控体系中的软PLC系统.机械管理开发，2005（8）:78—79

[33] 高金刚，陈建春，刘雄伟.数控系统的软PLC系统开发.计算机测量与控制，2004（12）:254—256

[34] 刘家亮，钟庆，黄树槐.“软件PLC”的设计与实现研究.电子机械工程2001（2）90-91

[35] 张礼兵吴婷基于软PLC编译系统目标代码生成的研究与实现.微计算机信息.2005（1） 85-87

[36] 卢艳军，任立义，闻邦春.基于软件PLC的I/O控制研究.机械制造.2005（7）37-39

[37] 罗华丽 李斌 汤志斌.开放式数控系统中的软件PLC 技术研究.组合机床与自动化加工技术2003（2） 38-41

[38] 张承瑞，程荣亮，王恒.面向对象技术在软件PLC编译器中的应用.软件技术与数据库·2004（3）76-77

[39] Peter Deussen. Partial Order Verification of Programmable Logic Controller. Lecture Notes in Computer Science. Vol. 2075, 2001: 144

[40] Andreas Schenk .SIMATIC S7-400F/FH: Safety-Related Programmable Logic Controller. Lecture Notes in Computer Science. Vol. 1943,2000: 286

[41] Nanete Bauer, Sebastian Engel], Ralf Huuck, et al. Verification of PLC Programs Given asSequential Function Charts. Lecture Notes in Computer Science. Vol 3147,2004: 517 — 540

[42] [Jin Hyun Kim, Su-Young Lee, Young Ah Ahn, et al. Development of RTOS for PLC Using Formal Methods. Lecture Notes in Computer Science. Vol. 3299 2004:479

[43] Kevin J. McDermot, Wenlong Albert Yao.Developing a Hybrid Programmable Logic Controller Manufacturing System. International Journal of Flexible Manufacturing

Systems.Vol.9, 1997（4）: 367—374

[44] St6phane Klein, Georg Frey, Mark Minas.PLC Programming with Signal interpreted Petri Nets. Lecture Notes in Computer Science. Vol.2679, 2003（8）: 440 — 449

[45] Ed Brinksma, Angelika Mader, Ansgar Felmker. Verification and optimization of a PLC control schedule. International Journal on Software Tools for Technology Transfer（STTT）. Vol. 4, 2002（1）: 21 — 33

[46] A. Ramirez-Serrano, S. C. Zhu, S. K. H. Chan, et al. A hybrid PC/PLC architecture for manufacturing-system control-theory and implementation. Journal of Intelligent Manufacturing. Vol. 13, 2002（4）: 261—281

[47] Can Saygin, Firat Kahraman. Web-based Programmable Logic Controller laboratory for manufacturing engineering education. The International Journal of A Advanced Manufacturing Technology. Vol. 24, 2004（7-8）: 590—598

[48] J. S. Lee, P-L. Hsu. An improved evaluation of Ladder logic diagrams and Petri nets for the sequence controller design in manufacturing systems. The International Journal of Advanced

[49] Manufacturing Technology. Vol.24, 2004（3-4）: 279—287