摘　要

可编程控制器（PLC）技术作为一种以微处理器及其存储器为控制中心的自动化装置，在工业自动化控制领域发挥着越来越重要的作用。然而，PLC 的发展与计算机技术、半导体技术、控制技术、数字技术以及网络通讯技术等高科技的发展有着紧密的联系。任何一项技术的发展都有可能直接导致 PLC 技术的革新。

PLC 发展到今天，其迅猛的发展势头为我们所有目共睹。与此同时，随着工业制造技术的不断进步以及过程的不断复杂化，对所需的控制程序提出了新的要求。纵观 PLC 的发展道路，长期以来 PLC 的研制走的是一条专用化的道路，使得在其获得成功的同时也带来了许多不便。

IEC61131-3 作为第一个为工业自动化控制系统的软件设计提供标准化编程语言的国际标准极大地改进了工业控制系统的编程软件质量，提高了软件开发效率。它定义的一系列图形化语言和文本语言，不仅对系统集成商和系统工程师的编程带来很大的方便，而且对最终用户同样会带来很大的方便。实践证明，采用 IEC61131-3 国际标准将是 PLC 控制系统发展的必由之路。

本论文首先对 PLC 系统的现状和发展趋势进行了相关调研，进而提出了本次课题的研究方向。然后对 IEC61131-3 标准作了详细的阐述，并依照此标准对软件系统进行了可行性分析、UML 建模以及部分功能的系统 C++语言实现。最后，对整个论文的研究情况以及今后所要开展的工作进行了简单的总结。

关键词：IEC61131-3，面向对象方法论，统一建模语言

Abstract

Programmable Logic Controller is a kind of automation device with microprocessor and memorizer as its control center, which play a more and more important role in the field of industry. However, the development of PLC have close relationship with the computer technology, semiconductor technology, control technology, digital technology and network communication technology etc. The improvement of any kind technology would lead to the reform of PLC control system.

As we can see that the step of PLC's development is very fast. At the same time, with the improvement of the manufacture technology and the increase of the process complication, it have a newer request for the control program. During the course of PLC's development, the production of PLC, from a long time is always used within a specially manufacture, because of it ,some inconvenience has been come out.

IEC61131-3 standard, as the first international standard for the software design of industry automation control system, increasingly improve the quality of programming language in industry control system and the efficiency of software manufacture. The series of graphics language and textual language defined in IEC61131-3 standard, take a great of convenience to not only the system integration manufacturers and system engineers but also the direct users. The practice proved that, the adoption of IEC61131-3 standard is the only way for development of PLC control system.

Firstly ,the paper investigate the actuality and the developing trend of PLC control system, then put forward the study direction of this paper. With the detailed expatiation on IEC61131-3 standard, the author made the feasibility analysis, set up the software system modeling and finally realized it with the advanced language C++. At last, summarize the investigation production of this paper and the work in further.

Keys: IEC61131-3, Object—Oriented Methodology, Unified Modeling Language

郑重声明

本人的学位论文是在导师指导下独立撰写并完成的，学位论文没有剽窃、抄袭、造假等违反学术道德、学术规范和侵权行为，否则本人愿意承担由此而产生的法律后果和法律责任，特此郑重声明。

学位论文作者（签名）：

2005年 6月 1日

1. 绪论

1.1. PLC 系统的现状及发展趋势

1. PLC 定义[1]

可编程控制器（简称 PLC：Programmable Logic Controller）是一种以微处理器及存储器为控制中心的自动化控制装置，它利用来自输入/输出装置的反馈信号及存储的程序，控制机械或程序的动作。在国际电工委员会（IEC）1987 年的可编程控制国际标准的第三稿中，对其具体定义如下："可编程控制器是一种数字运算操作的电子系统，专为在工业环境下应用而设计的。它采用了可编程序的存储器，用来在其内部存储执行逻辑运算、顺序控制、定时、计数和算术运算等操作的指令。并通过数字式和模拟式的输入和输出，控制各种类型的机械或生产过程。可编程控制器及其有关外部设备，都应按易于使工业控制系统形成一个整体，易于扩充其功能的原则而设计。"

在 PLC 问世之前，工业控制领域中的顺序控制主要是继电器控制占主导地位。继电器控制系统有着十分明显的缺点：体积大、耗电多、可靠性差、寿命短、运行速度慢、适应性差，尤其当生产工艺发生变化时，就必须重新设计、重新安装，这样就造成了时间和资金上的严重浪费，为了改变这一现状，以在激烈竞争的汽车工业中占有优势，因而提出开发出这样一种新的工业控制装置来取代继电器控制装置。美国电气制造商协会（NEMA）经过 4 年的调查，于 1980 年把这种新型的控制器正式命名为可编程控制器（Programmable Controller），英文缩写为 PC，而后由于个人电脑（Personal Computer）的兴起，而个人电脑的缩写也是"PC"，所以可编程控制器才更名为"Programmable Logic Controller"，简称 PLC。

2. PLC 的现状及发展趋势[2]

PLC 发展与计算机技术、半导体技术、控制技术、数字技术、通讯网络技术等高科技的发展息息相关。这些高新技术的发展极大的推动了 PLC 的发展，而 PLC 的发展又对这些高新技术提出了更高、更新的要求，从而促进它们的发展。PLC 从诞生到今天，发展速度十分惊人，目前用可编程控制器设计自动控制系统已经成为了世界的潮流。纵观其发展过程，大致可以分为以下四个阶段。

①初级阶段：1969～1972 年。这一阶段的产品主要用于逻辑运算和计时、计数运算。它的 CPU 由中小型规模数字集成电路组成，控制能力比较简单。

②扩展阶段：1973～1978 年。这一阶段产品的主要控制功能得到了较大扩展。扩展的功能包括数据的传送、数据的比较和运算、模拟量的运算等功能。在继承并扩展了逻辑控制功能的前提下，增加了对模拟量的控制。

③通信阶段：1979～1984 年。这一阶段的产品与计算机通信系统的发展有关，并形成了分布式通信网络系统。但是，由于制造商各自为政，通信系统也各自成为独立的系统，使各制造商产品较难实现互通。另外，该阶段的 PLC 产品的功能也得到了发展，数学运算功能也得到了较大的扩充，同时产品的可靠性进一步提高。

④开放阶段：1984 年至今。由于国际化标准组织提出了开放性系统互连参考模型，使得可编程控制器在开放功能上有较大发展，主要表现在通信系统的开放性。这一阶段的产品规模增大，功能不断完善，大中型产品多数采用标准化软件系统，系统的扩展也因通信功能的改善而变得方便。

虽然 PLC 只有 30 多年的历史，但其发展势头迅猛，目前 PLC 的生产增长率仍然保持在 30%～40%的水平，成为当今增长速度最快的工业控制器，而且还将要继续发展下去

随着工业制造技术的不断进步，制造过程的不断复杂化，所需的控制程序也越来越复杂，所需的资料量也越来越大。然而，长期以来 PLC 的研制走的是专门化的道路，使其在获得成功的同时也带来了许多的不便。由于目前的 PLC 都是封闭式的系统架构，各个厂家的 PLC 系统以及编程工具的独立性，导致了 PLC 平台的不兼容，即各种类型的 PLC 制造商都有本身指定的一套 PLC 编程软件，以及对外的通讯传输协议，这样就使得系统的使用者和程序设计人员，需要花费许多的时间去学习各个厂商的编程软件及设备间的通讯方式。另一方面，传统的 PLC 编程语言本身也存在着许多难以克服的缺点，而至今仍然采用的诸如指令表、梯形逻辑或控制系统功能图等传统的 PLC 编程方法已经达到其极限。未来的 PLC 应用程序，除了顺序控制外，人机界面、通讯及监控组态等都将是程序的重要组成部分，如果 PLC 程序的开发方法得不到改进，程序的开发时间、错误率都将会相对的提高，而对于大型程序的编制、纠错和维护也将是个巨大的工程。因此用户迫切地需要统一的、独立于制造商的语言概念的高级编程语言和开发工具，类似于早已在 PC 领域中存在多年的高级编程语言和开发工具。最近 10 年来，随着控制技术和现代化的 PC 机的不断发展，使不断提高效率的 PLC（可编程逻辑控制器）编程工具的开发成为可能[3]。

早在上个世纪 80 年代，IEC（International Electrotechnical Commission 国际电工技术委员会）国际标准化组织的第六工作组就开始着手制定统一的可编程控制器的国际标准，以对 PLC 未来的发展指定一种方向或框架，并且相继颁布了一系列的 PLC 标准，其中包括：

- IEC61131-1：General Information（一般信息）；
- IEC61131-2：Equipment Characteristics And Test Requirement（设备特性与测试

特性）；

- IEC61131-3：Programming Language（编程语言）
- IEC61131-4：User Guidelines (用户导则)
- IEC61131-5：MMS Companion Standard （制造信息规范伴随标准）

工业界正按照上述标准朝着硬件及软件两方面为 PLC 谋求新的发展。在硬件方面，有着 PC－based PLC 的产生；而在软件方面，于 1993 年正式颁布了 IEC 61131-3 标准，它是第一个为工业自动化控制系统的软件设计提供标准化编程语言的国际标准。该标准极大地改进了工业控制系统的编程软件质量，提高了软件开发效率。它定义的一系列图形化语言和文本语言，不仅对系统集成商和系统工程师的编程带来很大的方便，而且给最终用户同样会带来很大的方便。

正是由于 IEC61131-3 标准的公布，和国际 PLCopen 组织的长期努力，许多 PLC 制造商先后推出了符合该标准的 PLC 产品。美国 A-B 公司的许多 PLC 产品都带符合 IEC61131-3 标准中的结构文本的软件选项；德国西门子公司的 SIMATIC S7-3000、S7-400、C7-620 均采用了 SIMATIC 软件包，软件包中的梯形图部分完全符合 IEC61131-3 标准，而一些其他的可任选软件 S7-SCL、S7-GRAPH 则提供了 IEC61131-3 标准中的结构文本和顺序功能图编程方式。

实践证明，IEC61131-3 标准的引入势必将引起一场空前的工控技术革新，既促进了市场的繁荣，同时也加强了行业之间的良性竞争，采用先进的国际标准是自动控制厂商发展的必由之路。

3. 软 PLC 技术[4,19,50]

软 PLC 控制技术亦称为 Soft logic 和基于 PC 的控制技术，目前对于它还没有一个准确而统一的定义。西门子公司将软 PLC 定义为“集控制、人机界面、数据处理、通讯等功能于一台 PC 的解决方案”；3S 公司定义 SoftPLC 为“一种能将工业 PC 机转换为高端 PLC 的软件”；http://it-div.web.cern.ch 网站上称“SoftPLC 是一种能够使用户在无 PLC 硬件支持下，在普通 PC 上开发和运行程序的软件解决方案”。因此，软 PLC 可以说是传统 PLC 的软件化解决方案，能够在 PC 机上依靠一定的软件平台，完成 PLC 的所有功能，并且有开放的体系结构。

与传统的 PLC 相比，软 PLC 具有符合现代工业控制的许多优点：

(1) 具有开放的体系结构，SoftPLC 具有宽范围的 I/O 端口和多种现场总线的接口，支持多种硬件，能够解决传统 PLC 互补兼容的问题，并具有第三方软件接口，可支持多种语言编程，可允许用户根据需要，灵活扩展系统功能。

(2) 遵循国际工业标准和事实上的工业标准，例如 IEC61131 标准和 IEC61499 标准。

(3) 能充分利用 PC 机的资源，如大容量的内存，高速 CPU 及其它硬件。

(4) 具有更强的数据处理分析能力。

(5) 具有友好的人机界面，便于操作。

(6) 强大的网络通讯功能。

(7) 节约成本和培训费用。

在上述众多优点中，软 PLC 技术发展的一个重要的条件就是 IEC61131 标准的指定，其中，IEC61131-3 是 PLC 编程语言的标准，它详细的定义了句法、语义和五种 PLC 编程语言的表达方式。同时，它还允许用户在同一个程序中使用多种语言混合编程，方便了编程者选择合适的语言来进行设计。可以说 IEC61131-3 标准的指定，是软 PLC 技术的重要条件和保证。

1.2. 论文研究的意义和主要的工作

伴随着可编程控制器功能需求的不断增长，其硬件性能以每年约 25%的速度增长，软件的性能也以每五年提升一倍的速度在增长，因此开发 PLC 程序的难度也随之提高。未来的工业应用程序，除了顺序控制外，人机界面，通讯及统计监控程序也是程序系统的重要组成部分。然而，传统的 PLC 编程方式（如指令表语言和梯形图语言等），其功能已经达到极限，很难再有本质上的发挥空间。如果开发方法不改变，程序的开发时间和错误率都将会相对的提高，同时对于大型程序的纠错和维护也将是一项非常艰巨的工程。

与此同时，鉴于目前工业界的大多数控制装置都是构架在专有的品牌 PLC 之上，而大多数的可编程控制器仍然是封闭性的，其编程软件的操作、编程语言的使用以及对外的通讯都各不相同，遵循不同的协议和标准，这样就使 PLC 软件应用系统产生了一系列的问题，主要表现在一下几个方面：[5]

1. 不同厂商的 PLC 产品，其编程语言的符号和编程规则存在很大差异，程序的可移植性差；
2. 传统 PLC 语言编写的程序可复用性差。现代的软件编程方法特别强调程序的可重复性使用，而传统的诸如梯形图语言所编制的程序很难通过子程序的调用完成模块程序的反复使用。
3. 缺乏足够的程序封装能力。现代软件的开发，一般要求将一个复杂的程序分解为若干个不同功能的程序块（或者称为子程序）。人们在编程时，总是希望用不同的功能块组合成一个复杂的程序，而传统的 PLC 编程语言难以实现程序的模块化以及内部数据的封装和安全性能保证。
4. 传统的 PLC 编程语言不支持数据结构。传统的梯形图语言不支持数据结构，无法实现将数据组织成诸如 Pascal、C 语言等高级语言中的数据结构

那样的数据类型，对于一些复杂应用的编程，它几乎无能为力。

5. 传统的 PLC 程序执行具有很大的局限性。由于传统的 PLC 均是按扫描方式循环执行程序，因此整个程序的指令代码完全按顺序逐条执行。对于要求即时响应的程序，具有很大的局限性。
6. 传统 PLC 编程语言在算术运算和处理、字符串或文字处理等方面均不能提供强有力的支持；

同时，在完成 PLC 与 Windows 或 Unix 系统为主的网络及资料系统的整合时，系统的设计者不得不花费更多的时间去学习各个厂家的程序设计软件及设备的通讯方式，这样大大提高了 PLC 使用的成本费用。

为了改善上述的缺点，工业界正朝着硬件和软件两方面寻求解决的方法。在硬件方面，有 PC-based PLC 的产生，而在软件方面，IEC6113-3 标准的出现更是为工业控制的软体设计提供了强有力的技术支持,其构架思想就是根据 IEC 组织所制定的 IEC61131-3 标准的程序编辑方式，提供程序编辑环境，然后通过代码的编译和转换，最后将可执行程序下载到控制器环境中去，以达到与传统 PLC 相同的功能。

鉴于以上的讨论，本论文研究的目的是通过对 IEC61131-3 国际标准的研究，并试图提供一个在 Windows 或 Unix 系统下运行，遵循 IEC61131-3 标准的工业控制程序开发环境，它要求具有良好的可视化人机界面，拖拉式的编辑环境，并提供使用符合标准的五种编程语言。整个系统配置如下图 1-1 所示：

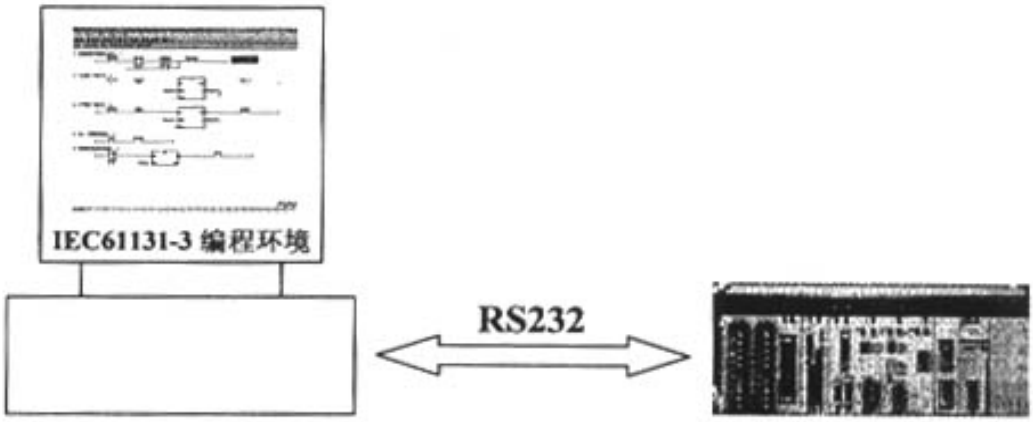

图 1-1：系统的配置图

1.3. 论文的结构

本论文主要从五个部分阐述了基于 IEC61131-3 标准编程环境的设计与开发过程。各个章节的内容说明如下：

第一章：绪论。介绍了 PLC 系统的现状及其发展趋势，并由此确定了论文的研究方向和开展的主要工作；

第二章：详细介绍了工业自动化系统软件编制的国际标准——IEC61131-3，并从中并总结得出了该标准对工业控制软件编程系统的各项要求。

第三章：对系统进行可行性分析与规划。通过对软件开发方法和系统建模方法的讨论与研究，最终确定了面向对象的迭代增量法以及统一建模语言——UML 的运用。

第四章：软件系统的 UML 建模与实现。主要是介绍运用 UML 对系统进行建模，并利用面向对象的高级编程工具 C++ Builder6.0 对局部功能进行实现。

第五章：结论与展望。

2. IEC61131-3 国际标准

2.1. 相关知识背景

1. IEC61131-3 标准[2,9]

国际电工委员会（International Electrotechnical Commission：简称 IEC）成立于 1906 年，是为电工、电子和相关技术领域指定和出版国际标准的世界性组织，总部设在日内瓦。它的宗旨是通过其成员促进电工、电子和相关技术领域的一切电工标准化及相关事务的国际合作，自其成立以来颁布了一系列的国际化标准。早在上个世纪 80 年代，IEC 国际标准化组织的第六工作组就开始着手制定统一的可编程控制器的国际标准，以对 PLC 未来的发展指定一种方向或框架，并于 1993 年正式颁布了关于 PLC 的国际标准——IEC61131 标准，其中包括以下部分[20]：

- **IEC61131-1**：General Information（一般信息），这部分包含通用定义和 PLC 区别于其它系统的典型功能特征，这包括标准 PLC 的特性，例如：循环处理具有存储输入/输出值映象区的应用程序，或编程设备、PLC 以及人一机接口；
- **IEC61131-2**：Equipment Characteristics And Test Requirement（设备特性与测试特性），这部分定义了设备的电气、机械和功能要求以及相应的产品质量合格性测试。列出了控制器及编程设备的环境条件（温度、空气湿度等）和重要性等级，对这部分标准的修订目前正在进行中；
- **IEC61131-3**：Programming Language（编程语言），这部分将世界上已经广泛使用的 PLC 语言组合到一个协调且面向未来的语言版本中来。通过形式定义、词法、语法和部分语义的描述及示例，定义了基本的软件模型和编程语言。
- **IEC61131-4**：User Guidelines (用户导则)，这部分试图作为一个导则，对从事自动化项目的所有阶段的客户提供帮助，提供面向实际的信息，其主题包括从系统分析，装置选择直至维护等；
- **IEC61131-5**：MMS Companion Standard （制造信息规范伴随标准），这部分是关于不同制造商的 PLC 之间以及它们和其它设备之间的通信。

其中 IEC61131-3 作为它的第三部分——关于编程语言的标准。该部分标准规范了可编程控制器的编程语言及其基本元素，是第一个为工业自动化控制系统的软件设计提供标准化编程语言的国际标准。该标准一经诞生就得到了包括有美国 A-B 公司、德国西门子公司等世界范围的众多厂商的推动和支持，它极大地改进了工业控制系统的编程软件质量，提高了软件开发效率。它定义的一系列图形化语言和文本语言，不仅对系统集成商和系统工程师的编程带来很大的方便，而且对最终用户同样会带来很大的方便。

2. PLCopen 组织介绍[6]

成立于 1992 年的 PLCopen 组织，是一个独立于 PLC 制造商和 PLC 产品的国际组织，总部位于荷兰。它致力于 IEC61131-3 标准的使用和推广，并着力于改善标准本身，如特定应用领域的行为规范，兼容级别等方面的研究。到目前为止，PLCopen 组织在欧洲和世界范围共有 88 个会员，来自世界 19 个国家的 PLC 制造商、软件公司和独立的研究机构和用户。该组织的宗旨是促进 PLC 兼容软件的开发和使用，而实现该宗旨的方法则是基于以下几点：

- 采用 IEC61131-3 国际标准；
- 受PLCopen会员的委托生产或采用符合IEC61131-3国际标准的PLC产品；
- 共同支持市场策略，例如，积极开展专题研讨会和展览会；
- 支持国家标准化委员会的工作；
- 为更好的评估编程系统建立符合性等级，并由独立的机构测试以执行必要的检验；

需要指出的是，PLCopen 组织并不是像 IEC 是一个标准化委员会，而是一个具有共同利益的集团，它分为若干个委员会，每个委员会处理所关心的专门领域，例如，技术委员会拟定共同政策的指导原则，宣传推广委员会则负责市场策略。这个集团的目的是希望现有的标准获得国际上的接受，并能够提供一些帮助。其目标是提供一套编程语言的标准集合，即 IEC61131-3 标准所说的 PSE 编程支持环境，该编程环境可以在多个 PLC 开发环境中实现，而不是开发单一的独立 PLC 编程环境。

2.2. IEC61131-3 标准的产生背景及主要特点

随着 PLC 系统的日趋复杂化，将直接导致其成本的持续上升，具体表现在[7,8]：

- 培训应用程序的编制人员方面；
- 编制容量不断加大的程序方面；
- 越来越复杂的编程系统的实现方面；

所以通过标准化和最佳化 PC 机为主流的 PLC 编程系统可以大大降低上述费用，使制造商和客户双方均能从中受益。然而，传统的梯形图在实现复杂系统的编程方面已经很难满足要求。与此同时，传统的 PLC 编程方式本身也存在许多难以克服的缺点，表现在以下几个方面[5]：

a) 不同厂家 PLC 产品的梯形图符号和编程变化很大；
b) 有限的程序可重用性：程序可重用性是现代编程的一个发展趋势，传统的 PLC 不能通过重复调用相同的逻辑策略和算法，实现程序重复使用。
c) 有限的数据封装能力：传统的梯形图程序很难将一个复杂的程序分解为数个简

单的程序部分。现在的梯形图编程，一个程序块的内部数据还缺乏对外部隐藏其数据的封装能力，因而，一个大的程序要想分解为几个简单的小程序，并且各个小程序之间具有的清晰的接口是很困难的。

d) 不支持数据结构：在许多复杂的应用中，程序需要把一些数据组织成类似高级语言 PASCAL 、C 中的数据结构那样的数据类型，而目前的梯形图程序还不支持数据结构。

e) 对顺序操作功能的编程支持有限：传统的 PLC 梯形图编程对顺序操作的处理方法是，为每一个顺序状态提供一个状态位，这种对顺序操作的处理能力是很有限的。

f) 程序执行的局限性：PLC 程序是顺序执行的，执行一次程序的时间取决于程序的长短和复杂性，对很大和很复杂的程序， 执行一次程序的时间就较长，这对有些对时间有苛刻要求的应用，是有很大的局限性的。

g) 执行算术操作的局限性：传统的 PLC 梯形图程序对算术操作处理是很有限的。

与传统的 PLC 编程语言相比，IEC61131-3 标准具有比较全面的优点，主要表现在以下几个方面：[9,10]

（1） 标准的编程系统开放性

该标准规定的软件模型是一个国际标准的软件模型，它不是针对具体的 PLC 系统，即 IEC61131-3 编程系统不依赖于不同的制造商硬件平台，不同的软件制造商逻辑组态软件数据可相互导入导出，具有很强的适用性。与此同时，IEC 编程器支持离线仿真，多种调试手段（如断点、单步、跳进跳出、单循环执行），支持多种通讯协议（如 TCP/IP、CAN、Profibus、RS232 等），对不同 HMI(人机界面软件)的驱动适应能力，算法库的兼容能力，二次开发接口的能力等。

（2） 面向对象的 IEC61131-3 标准编程方法

为使 IEC61131-3 标准的编程语言成为不依赖于硬件平台的多平台通用语言，该标准定义了基本的大量的函数和功能块，而且允许用户按照输入变量、输出变量、中间变量等既定接口来自定义函数和功能块。每个功能块实例就是一个独立的可完成特定逻辑功能的活动对象，不同的程序、不同的任务都可以定义和调用各种功能块的应用实体，每个调用实体都占用独立的内存，保留独立的逻辑状态。这种面向对象编程的方法，具有非常明确的现实意义。

（3） 结构化的程序编程思想

IEC61131-3 标准编程方式强调结构化的程序结构，采用自顶向下的编程方法，通过硬件配置与软件逻辑的分离、IEC 任务调度、IEC 程序调用等手段，使应用工程的逻辑结构更加清晰。五种语言元素的使用，使从块和数据的定义，到硬件的配置清晰明了，便于应用程序的服务及维护。

（4） 统一的编程语言标准

它所定义的5种编程语言（SFC、LD、FBD、IL、ST），基本上涵盖了整个工业控制应用领域，可以最大限度的在不同的PLC系统，甚至是DCS和HMI以及现场总线系统中运行，这样更有利于降低工程成本、统一文档、提高工程效率。

（5） 变量定义的方便性及安全性

IEC61131-3按照一定的语法，将一定字节长度及地址偏移的硬件地址IO映射为相应数据类型的变量，通过此变量实现对硬件的读写操作，使易用性及安全性大大提高；IEC61131-3标准的编程工具通过关键字自动识别全局变量和局部变量，保证在不同的程序中均可正确访问全局变量。不同的子程序允许定义自己的局部变量，不与全局变量矛盾；IEC61131-3必须为每个变量指定唯一的数据类型，变量定义采用统一的格式；用户可自定义数组及结构类型的变量（如枚举、联合等），使实际编程更为简洁和方便；所有类型数据可以自由定义初始值或使用该类型变量的缺省值；用户自定义的数据类型允许定义相应的初始值。

（6） 方便、开放的功能库管理

用户在编程的时候，既可以使用系统提供的函数库，也可以自己编辑内部库函数，这些函数可以任意选择6种编程语言进行逻辑实现，通过接口变量定义来确定调用接口，并以特定方式存为内部库文件。用户在组态工程中包含库文件后，即可进行任意的库函数实例定义与调用。对这些内部库函数的封装，既是基于面向对象编程思想的体现，又增加了程序的可读性。很多的内部库，如数控逻辑内部库、连续控制系统PID调节内部库等，大大丰富了标准的应用范围，易于实现工程的规模化。

综上所述，这个独立于制造商的标准可显著的降低PLC程序员的培训和熟悉时间，程序的编写将会更加可靠，且PLC系统的功能性可跟上当代由PC机提供的功能强大的软件开发环境。IEC61131-3标准将促使PLC制造商和用户放弃原有的习惯而采用一个全新的，符合未来软件发展方向的现代化的编程技术。

2.3. IEC61131-3标准的软件模型[5,8,11]

IEC61131-3标准提出的软件模型，如图2-1所示，它定义了诸多概念，包括：配置（Configuration）、资源（Resource）、任务（Task）、程序（Program）、功能块（Function Block）以及功能（Function）和它们之间的连接。且该软件模型是一种

分层式的结构模型，它定义了许多的层次，每一层隐藏了其下一层的许多特性。有了这种分层结构软件模型，人们就能完整地理解除编程语言外的全部内容。

在配置本软件模型的过程中，在其最上层把解决一个具体控制问题的完整的软件概况称为一个“配置”。在一个由多台 PLC 构成的控制系统中，每一台 PLC 的应用程序就是一个独立的“配置”。在一个“配置”中可以定义一个或多个“资源”。在一个“资源”内可以定义一个或多个“任务”，“任务”被配置后可以控制一组程序或功能块，这些程序和功能块可以是周期性地执行，也可以由一个事件驱动予以执行。一个 IEC 程序可以用符合 IEC61131-3 规定的编程语言来编写，而典型的 IEC 程序由许多互连的功能块和函数组成。

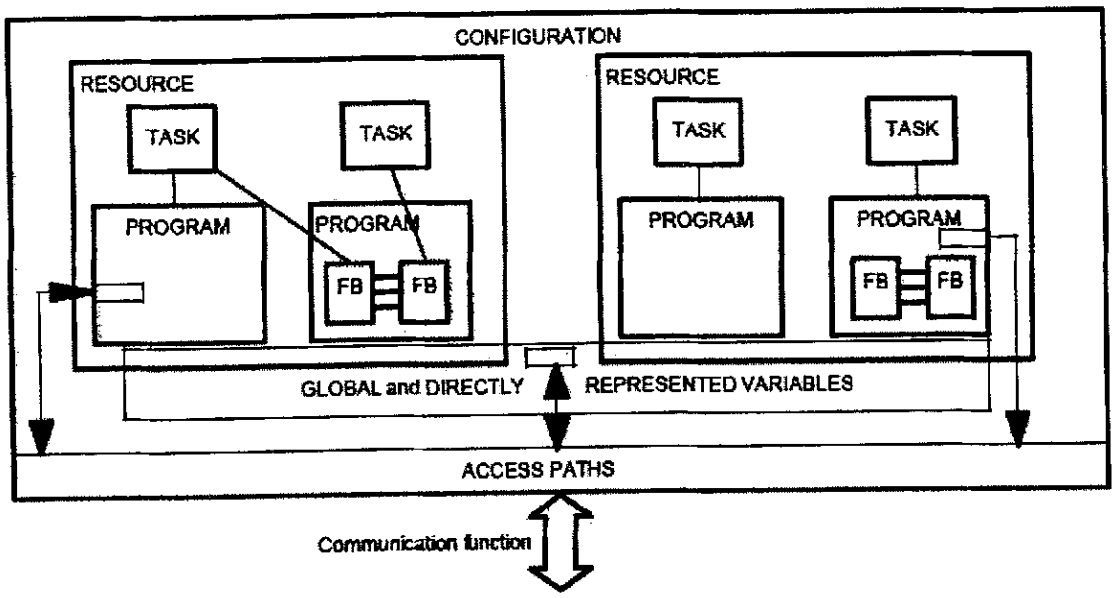

图 2-1：IEC61131-3 的软件模型[12]

1. PLC 的配置（CONFIGURATION）[13]

在软件等级中，最高等级是“配置”，它定义了单元的结构，专指一个特定类型的控制系统，包括硬件装置、处理资源、I/O 通道的存储地址和系统能力，等同于一个 PLC 的应用系统。如图 2-2 所示为一个带有多个 CPU 连接的 PLC，一个配置包括一个或若干个资源，它构成了一个 CPU，资源的程序由任务来控制，任务表示一个可执行的程序单元。

IEC61131-3 使用配置将 PLC 系统的所有资源结集成组，并给它们提供数据交换的手段，一个“配置”的文本性说明如下图 2-3（左）所示。在一个配置内，可以作出对整个 PLC 项目全局有效的类型定义，这在其它的结构元素中是不可能的。

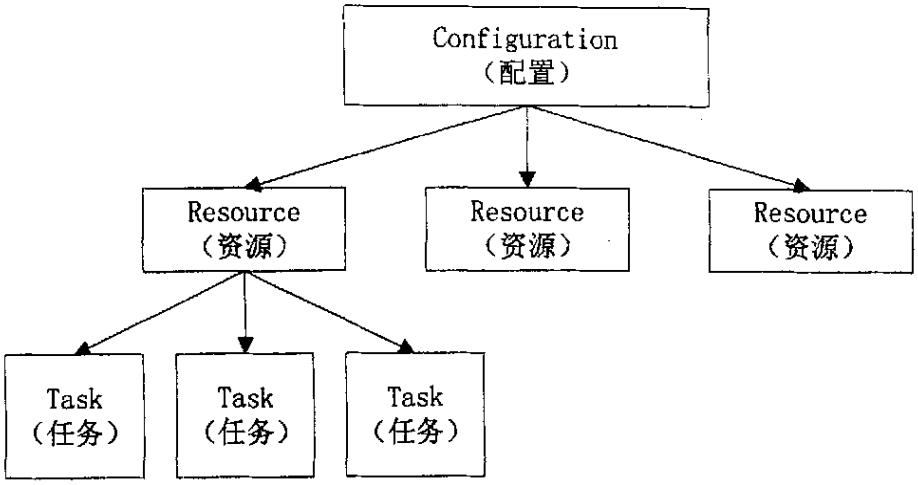

图 2-2：配置、资源、任务

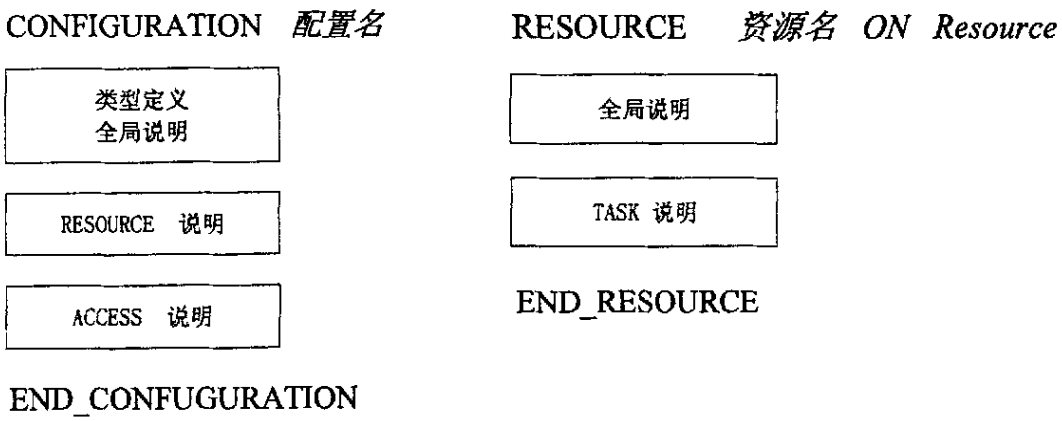

图 2-3：配置、资源文本化定义

2. 资源（RESOURCE）

在每一个配置中，有一个或多个“资源”，“资源”不仅为运行程序提供了一个支持系统，而且它反映了 PLC 的物理结构，在程序和 PLC 物理 I/O 通道之间提供了一个接口。只有在装入了“资源”后才能执行 IEC 程序。一般而言，通常“资源”放在 PLC 内，当然它也可以放在其它支持 IEC 程序执行的系统内。与此同时，一个 IEC 程序只有在装入了“资源” 后才能执行。

IEC61131-3 标准中对“资源”的定义是为了将“任务/（Task）”分配给一个 PLC 系统的物理资源。构成一个资源的元素的文本性定义如图 2-3（右）所示：

资源名将一个符号名赋予 PLC 中的一个 CPU。编译系统提供 PLC 系统内该资源（每个 CPU 的命名）的类型和数量，并检验这些资源的类型和数量以确保能正确地使用它们。

3. 具有运行期程序的任务（TASK）

在PLC的配置中，定义任务（TASK）的目的在于规定程序及其功能块的运行期特性。“任务”被配置以后，可以控制一系列程序或功能块周期性地执行程序或由一个特定的事件触发而开始执行程序。也就是说，IEC61131-3 标准规定在标准PLC 中的 IEC 程序或功能块通常保持完全的待用状态，只有当是由一个特定的被配置后的任务来周期性地执行，或当一个特定的变量状态发生改变而引起触发执行的情况时，IEC 程序或功能块才会执行。

“任务”的种类有很多，例如：周期任务、时间控制任务（时间间隔任务）、事件控制任务（事件任务）、中断任务等。它文本化的说明结构如下：

TASK　　*任务名*　*（任务属性）*

PROGRAM　　*程序名*　WITH　*任务名：（程序接口）*

以上的格式，定义了一个“任务”，并将该任务与一个“程序”相关联。“任务属性”给出了该任务的参数值，而“程序接口”则为形式参数提供了实际参数。带有“程序（PROGRAM）”的任务（TASK）定义了一个具有“程序名”的运行期程序。这是在说明中给出的调用接口的程序实例名。另外，对“任务”的定义除了能以文本的形式给出外，还可以以图形的形式给出，如图 2-4 所示。

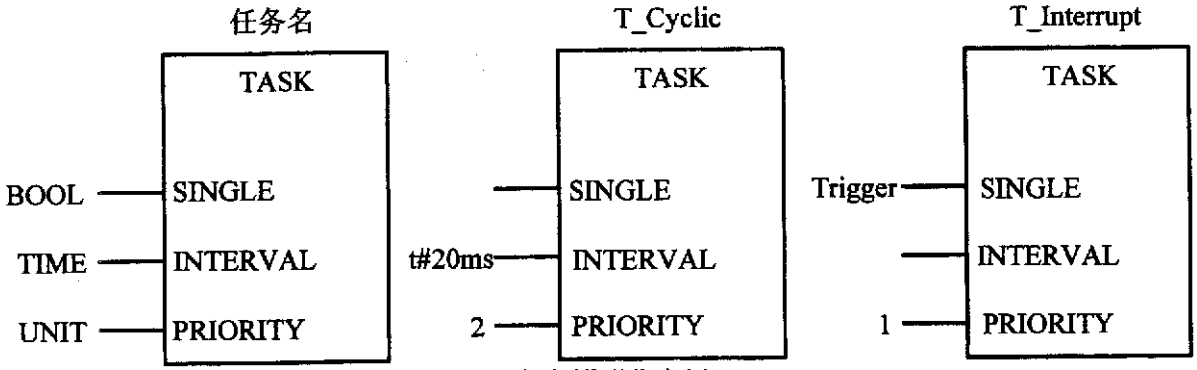

图 2-4：任务图形化实例

左侧是对于“任务”图形表达式的通用形式，中间和右侧是“任务”的两个实例。对于其通用形式的各个输入参数说明如下：

表 2-1：作为输入参数的“任务”特性说明

TASK 参数	说明
SINGLE	在这个输入的上升沿，调用与 TASK 相关联的程序，并执行一次；
INTERVAL	如果调用的时间值不是零，则周期性地循环执行与 TASK 相关联的所有程序。提供的时间是两次调用之间的时间间隔。此时，该时间值可用于设置和监视循环时间。如果输入值为零，则程序将不被调用。
PRIORITY	与其它同时运行的程序相比较，这个输入定义所关联程序的优先级。

4. 程序组织单元（POU）[10]

为了与传统的 PLC 编程领域的程序块、组织块、顺序块和功能块相对应，并同时对块的多样性加以限制，IEC61131-3 标准引入了新的构成 IEC 程序和项目的块，即程序组织单元（Program Organization Unit）——POU，如图 2-5 所示。POU 是标准 PLC 系统用户程序中最小的、独立的软件单元。

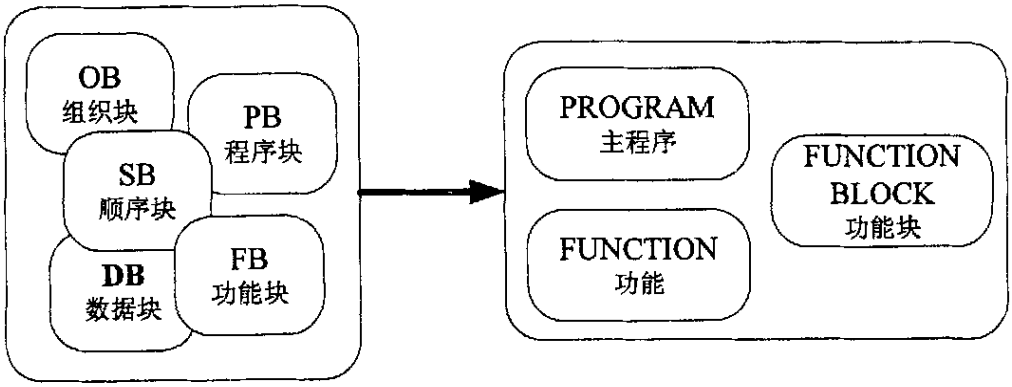

图 2-5：程序块的演变

传统 PLC 系统的块类型　　IEC61131-3 标准系统中的 POU

正如上图 2-5 所示，IEC61131-3 标准将传统 PLC 制造商的块类型的种类减少为 3 种统一的基本类型。按它们功能性的递增顺序，分别为 Function（FUN，功能）、Function Block（FB，功能块）和 Program（PROG，程序）。对于 3 种 POU 类型及其含义见下表：

表 **2-2**：POU 含义说明

POU 类型	关键字	含义
Function	**FUNCTION**	具有功能值，用于扩展 PLC 的基本预算、操作集；
Function Block	**FUNCTION_BLOCK**	带输入和输出变量的块，这是最常用的 POU 类型；
Program	**PROGRAM**	主程序，包括 I/O 的分配、全局变量和存取路径；

➢ 功能（FUN）

IEC61131-3 对于“功能（FUN）”定义的基本思想是：在功能主体内，作用于输入变量值的指令产生一个单值的功能值。在这个意义上，功能可以视为制造商专用或应用专用的 PLC 操作运算或指令集的扩展。它的简单规则是，相同的输入值总是产生相同的功能（返回）值。这和调用功能的频繁度以及何时调用功能无关，这也是区别于功能块（FB）的显著标志。

为了简化和统一 PLC 系统的基本功能性，IEC61131-3 预定义了一系列经常使

用的标准功能集，其中包括 50 种功能和 12 种功能块，对于这些功能的特性、运行时的行为特定以及调用接口都进行了标准化。这些元素就是人们所熟悉的标准功能和标准功能块，它们的名称保留为关键字。（详细的标准功能和功能块的描述可参阅 IEC61131-3 标准）

- 功能块（FB）

“功能块（FB）”是构成 PLC 程序的主要积木式的部件，可以由程序和功能块所调用，它们自身也能调用功能和其它功能块。功能块可以赋予参数并具有静态变量（带有记忆），当以相同的输入参数调用时，功能块（FB）的输出值取决于其内部变量（VAR）和外部变量（VAR_EXTERNAL）的状态，这些变量在功能块的这一次执行到下一次执行的过程中是保持的。

提到功能块（FB），我们不得不提到“FB 的实例化”，这个概念在 IEC61131-3 标准中显得特别重要。它是 POU 类型之间互相区别的必要准则。我们知道，“变量的实例化”就是程序员在说明程序部分指定变量名和数据类型来建立变量，而“功能块（FB）”也可以象变量那样进行实例化，从物理空间上，也就是开辟专门的存储空间，用于存放实例化后的变量（这里指的变量，不是单单指的简单数据变量，也包括特定意义上的结构变量）。

直到现在，我们使用的大部分制造商的 PLC 中用于计数或计时的功能块分别简称为计数器和定时器，主要就是根据其类型进行定义的，并由用户给出一个编号，例如计时器“C05”。IEC61131-3 标准不使用这种以绝对值表示的编号，它要求以一种变量名形式给出，并组合所期望的定时器或计数器的详细说明，然而这必须要在对 POU 的说明部分给予说明。将 POU 编译为 PLC 机器码时，编译系统能自动生成这些 FB 变量的内部绝对编号。借助于这些变量名，PLC 程序员能够以透明的方式使用具有相同类型的不同定时器或计数器，而不需要检验其名称有无冲突。

- 程序（PROG）

功能和功能块构成“子程序”，而 PROGRAM 类型的 POU 则构成 PLC 的“主程序”。在有多个任务能力的控制器上，能同时执行数个主程序，这一点体现了程序与功能块（FB）的不同。整个程序的实时性表现在程序能在 CPU 中运行，是由分配给任务的程序来解决的，一个程序能分配给若干个任务，亦即若干个程序背景在不同的实时性质下生成。程序中的一个主程序被分配给 PLC 外部设备、全局变量和访问路径。

- POU 之间的调用方式

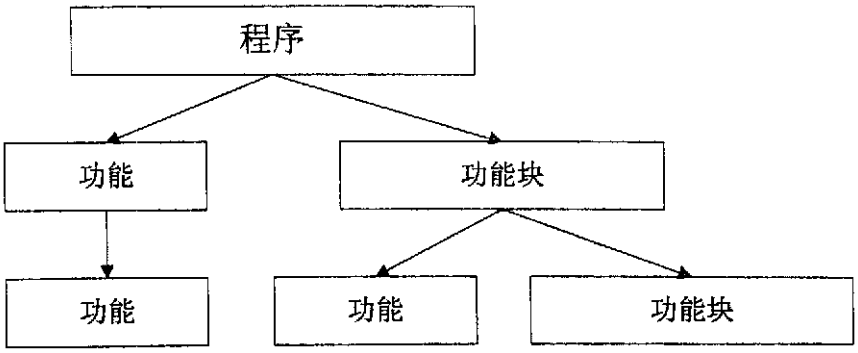

图 2-6：POU 的调用规则

如上图 2-6 所示，3 种 POU 的相互调用规则为：

1） “程序”可调用“功能块”和“功能”，但不允许反方向调用；

2） “功能块”可调用“功能块”；

3） “功能块”可调用“功能”，但不允许反方向调用；

4） POU 不能递归调用，即 POU 不能直接或间接地调用它自身的一个实例；

5. 通用语言单元[14]

- 标识符

标识符是一些字母和数字的字符串，PLC 程序员可使用标识符来命名变量、功能块和程序，并对它们进行寻址。通过 IEC61131-3 标识符定义后的单元，可以支持程序的可读性。标识符以一个字母或下划线字符开始，随后是按照需要的一定数量的字母、数字和下划线字符。且 IEC61131-3 标准对于标识符中英文字母的大写和小写不作区别。例如，变量“INPUT_OFF”和变量“Input_Off”或“input_off”都是等同的。

- 关键字

关键字是标准的标识符，能作为单个的语法助记单元，其拼写和期望目的均由 IEC61131-3 明确的规定，因此，关键字不能用于用户定义的变量名或其它名称。对于关键字而言，用大写字母或小写字母 IEC 不作要求，但是为了更好的加以区别，IEC61131-3 鼓励关键字以大写字母表达。

IEC61131-3 标准中保留的关键字包括以下内容：基本数据类型的名称、标准功能/功能块名、标准功能/功能块的输入参数、图形编辑语言中的 EN 和 ENO 变量、指令表语言中的运算符、结构化文本语言中的语言元素、顺序功能图语言中的语言元素。（对于具体的保留字详见 IEC61131-3 标准的“关键字”附录）。

- 注释

同其它高级语言一样，IEC61131-3 的编程语言中也允许在程序中加入一部分注释，用来帮助理解程序，是重要的沟通方法。注释允许在任何位置以所有文本

编辑的形式，而且必须以特殊的字母序列开始和结束，每一个网络段能对它的功能注释成一段文本。

- 数据类型

传统的 PLC 编程语言包括的数据类型，如浮点表达式，BCD 码或定时器和计数器值等，它们通常具有完全不兼容的格式和编码。大部分传统的编程系统统一地使用 BIT、BYTE、WORD 和 DWORD。但是，即使是简单地整数值，不同 PLC 制造商的系统之间仍然有细微的特殊区别，例如，有无符号，存储位地的数量。因此，在大多数情况下，与不相兼容的数据类型程序的接口，需要对程序作大量的修改。

由于 IEC61131-3 定义了 PLC 编程最常用的数据类型，因而在 PLC 领域内，这些数据类型的含义和使用都是统一的。这对 PLC 制造商，以及使用来自不同制造商的多台 PLC 和编程系统的工程技术人员来说，将会带来明显的利益。

➢ 基本数据类型

在 IEC61131-3 中，有一组预定义的、标准化的数据类型，称之为基本数据类型。如下表 2-3 所示：

表 2-3：基本数据类型

布尔类型/位串	有符号整型数	无符号整型数	浮点数（实型数）	时间、持续时间，日期和字符串
BOOL	**INT**	**UINT**	**REAL**	**TIME**
BYTE	**SINT**	**USINT**	**LREAL**	**DATE**
WORD	**DINT**	**UDINT**		**TIME_OF_DAY**
DWORD	**LINT**	**ULINT**		**DATE_AND_TIME**
LWORD				**STRING**

➢ 导出数据类型

在基本数据类型之上，PLC 程序员可以建立自己的“用户自定义”的数据类型，它包括枚举类型、数组类型、结构类型等，这使程序员能为其应用，最方便地实现数据模型。对类型定义必须采用文本的表达方式，对此 IEC61131-3 并没有提及图形表达方式。类型定义由关键字 **TYPE...END_TYPE** 构成。如下例所示：

```
TYPE
  LongFloatNum     :  LREAL;          (*从 IEC 基本数据类型直接导出*)
  FloatingPoint    :  LongFloatNum;   (*从用户定义的数据类型导出*)
  InitFloatNum     :  LREAL :=1.0;    (*用新的初始值导出*)
  Tcountrol        :  BOOL :=TRUE;    (*用新的初始值导出*)
END_TYPE
```

- 变量

在传统的 PLC 编程中，通常使用“操作数”来直接存取 PLC 存储器中的地址，例如，欧姆龙编程系统中的按槽位划分方式（“0.01”）和西门子 PLC 编程系统中的按操作数类型的划分方式(“I0.1”)。这些地址或者是在 PLC 中央处理单元(CPU)的主存储器中，或是在例如 I/O 模块中。这意味着，以物理地址编址的存储区可用于不同的目的，因而每种存储器单元也就有着特定的数据格式（8 位，16 位，32 位）。然而，这些数据格式一般互不兼容，程序员必须记住在每一个程序中 PLC 的编址可能会使用的格式。当指定一个不正确的存储器地址，或者使用错误数据格式的地址时，常常导致程序的错误。因此，对于多数 PLC 系统而言引入“符号”，它可以用来等效地替代 PLC 的绝对地址，从而确保 PLC 程序更具有良好的可读性。IEC61131-3 标准对此作了更深一步的改进，通过对变量的定义和使用，来替代硬件地址和符号，这就如同高级编程语言中的常规变量。变量是由程序员指定的标识符（名称），其作用如同“位置标识符”，它包含程序的数据值。

1. 变量的类型

IEC61131-3 定义了以下 5 种不同的变量类型：

1) 全局变量
2) 局部变量
3) 输入变量
4) 输出变量
5) 输入/输出变量

其中，局部变量不能连接到外部，即它们只能在程序内部的一部分进行寻址；全局变量能被所有的程序组织单元（POUs）寻址；输入、输出和输入/输出变量是与程序（Program）、功能（Function）和功能块（Function Block）有关的，它们能在被分配的 POU 内通过读或写来改变，而要在 POU 外部改变时必须进行定义（输入、输出和输入/输出），在原文件之间变量要加以说明。通常，每个变量在冷启动之后被初始化。默认的初始值为“0”或“假”，可以通过另一变量符号“：＝”在说明中指定其初始值。

2. 变量的属性

IEC61131-3 在定义变量的同时，也定义了变量的属性或限定符，并通过它们将附加的特性赋给变量。

- RETAIN　　变量具有保持的特性（电池后备）
- CONSTANT　　常数变量（不能修改）
- R_EDGE　　上升沿
- F_EDGE　　下降沿
- READ_ONLY　　写保护（只读）

- READ_WRITE　　可进行读和写
- AT　　具有固定的存储器地址（固定地址）

在以上的变量属性限定符中，RETAIN 和 CONSTANT 限定符是在变量类型的关键字后立即指定的，也就是说这 2 个限定符总是涉及变量说明的整个段（直至 END_VAR）；其他 4 个属性或限定符是单独地分配给不同地变量说明，它们不能与上述 2 个限定符相组合。

需要特别指出的是，为了直接存取程序中处理器和 I/O 模块的数据区，IEC61131-3 为该类型的变量提供了 2 种可能性，直接表达的变量和符号变量。在对直接表达变量的说明中，物理存储位置由关键字 AT 指定，后接由特殊字母序列构成的地址。这些地址以“%”开始，其后紧跟表示范围的前缀——字母 I（输入）、Q（输出）或 M（标志/存储器）。然后为表示长度的前缀——X（单个位）、B（字节，8 位）、W（字，16 位）、D（双字，32 位）。对于符号变量的说明及其使用如同正常变量的说明和使用，只不过其存储位置不能由编程系统自由的指定，而是限于由用户以“AT”指定的地址。这些变量对应于预先由分配表或符号表指定的地址。

例如：

```
AT%IX1.0.2      (*输入位 2*)
AT%QD3          (*输出双字 3*)
AT%MX1.3        (*标志字 1 中的第 3 个标志位*)
```

3. 变量的说明

在 IEC61131-3 标准中，变量用于初始化、处理和存储用户数据。在每个 POU 的开始部分，必须对变量予以说明，这就包括对变量数据类型，变量属性（如电池后备、初始值或物理地址赋值等）的定义。

对不同的变量类型，POU 变量的说明分为不同的段/说明块，每个段/说明块对应于一种变量类型，并可以包括一个或多个变量，且相同变量类型的块的次序和数量可以自由决定。如下所示，

```
(*局部布尔变量*)
 VAR              VarLocal : BOOL ;      END_VAR (*局部布尔变量*)
(*调用接口：输入参数*)
 VAR_INPUT        VarIn : REAL;          END_VAR (*输入变量*)
 VAR_IN_OUT       VarInOut : UNIT;       END_VAR (*输入和输出变量*)
(*返回值：输出参数*)
 VAR_OUTPUT       VarOut : INT;          END_VAR (*输出变量*)
(*全局接口：全局/外部变量和存取路径*)
```

VAR_EXTERNAL VarGlob : WORD; **END_VAR**（*外部，来自其它POU*）

VAR_GLOBAL VarGlob : WORD; **END_VAR**（*全局，用于其他POU*）

VAR_ACCESS VarPath : WORD; **END_VAR**(*到配置的存取路径*)

除了文本形式定义变量以外，对于 POU 接口的简单变量的说明，IEC61131-3 提供了图形表达的定义方式。但是，必须说明的是，对于数组，持变量或初始值的说明，必须使用文本化的表达方式。

6. 通信模式[14]

对于通讯方式，IEC61131-3 的目标是提供一个标准化的通信模型，从而能够建立良好结构化的 PLC 程序，它方便于调试和诊断，以及编制更完善的文档。为了使程序的不同部分之间进行数据交换，IEC61131-3 定义了以下的几种通讯方式：

- 内部变量通讯；
- 全局变量通讯；
- 调用参数通讯；
- 使用存取路径通讯；
- 直接表达的变量
- 通讯块

其中，前三种方法用于一个配置内的通信，通过内部变量和全局变量的建立可以在一个配置内的程序、功能块和功能之间相互连接形成一个网络，数据信息可以通过这个内部的网络进行通讯。存取路径的通讯方式用于各个配置之间的数据交换，也就是要跨越一个 PLC 系统的范围，它可用于配置和程序层。存取路径是全局变量的一个扩展，它只在一个配置内有效，由 **VAR_ACCESS...END_VAR** 语言结构所定义给出，通过符号名，一个配置的变量可为其他配置所认知。对于 PLC 直接表达的变量（例如%I，%Q 和%M）允许在一个应用程序的不同部分之间进行有限的通信，因为它们可在一个系统上被全局地存取，这些变量只能在程序层或更高层（即全局层）中说明。通讯块是用于从发送方向接收方传送数据包的专用功能块，因为这些功能块链接到一个程序，所以它们局限于一个配置，且对外部不可视。对于标准通讯块，在 IEC61131－5 中专门地作出了定义，这里就不详细介绍。

2.4. IEC61131-3 标准的编程语言

IEC61131-3 标准中的编程语言分为两类，图形化编程语言和文本化编程语言，

其中图形化编程语言包括：梯形图（LD——Ladder Diagram）、功能块图（FBD——Function Block Diagram）、顺序功能图（SFC——Sequential Function Chart），文本化编程语言包括：指令表（IL——Instruction List）和结构化文本（ST——Structured Text）。这五种编程语言基本上涵盖了全部工业控制领域的编程方式，这样做的原因是为了满足不同国家不同领域的编程人员对编程语言的要求，正如在德国广泛采用梯形图，而在美国则较多的使用指令语句。

以下先就这五种编程语言作简单的介绍，在以后的章节将针对本课题所要实现的两种编程语言（LD 和 FBD）作具体的分析[11,15]。

梯形图（LD——Ladder Diagram）来源于美国，是基于图形表示的继电器逻辑，用来描述一个 POU 的网络自左向右的能量流，是 PLC 编程中被最广泛使用一种图形化语言，因此 IEC61131-3 将它列为标准的编程语言之列。IEC61131-3 标准中定义的梯形图编程语言是对各个 PLC 生产厂家的梯形图语言的合理的吸收和借鉴，语言中的各种符号与各个厂家的基本一致。IEC61131-3 标准的梯形图编程语言中的图形符号包括以下几类，它们的具体性质在这里不作详细描述。

1. 接点类：常开接点、常闭接点、上升沿接点和下降沿接点；
2. 线圈类：一般线圈、反向线圈、置位/锁存线圈、复位/解锁线圈、保持/记忆线圈、置位保持/记忆线圈、复位保持/记忆线圈、正转换线圈、下降沿线圈；
3. 功能和功能块：包括标准的功能和功能块以及用户自己定义的功能块；

梯形图语言中使用网路的概念，一个 LD 网络的边界是在左侧和右侧所谓的电力轨线（Power rails）。左侧的电力轨线，名义上是为“功率流”从左向右沿着水平梯级通过各个触点、功能、功能块、线圈等提供能量，“功率流”的终点是右侧的电力轨线。期间的每一个触点代表了一个布尔变量的状态，每一个线圈代表了一个实际设备的状态，功能或功能块与 IEC61131-3 中的标准库或用户创建的功能或功能块相对应，根据这些元素的逻辑状态来决定或是允许能量流通过，或是中断能量流。由此，便构成了所需要的逻辑程序。

功能块图（FBD——Function Block Diagram）是一个开放格式的图形编程语言，它起源于信号处理领域，对一个硬件工程师而言，它如同是一个电子电路图或逻辑电路图。在程序中，它可以看作两个过程元素之间的信息流。在工业控制器领域，FBD 作为一个普遍采用的语言其重要地位已经确立。FBD 语言同其他的工业控制图形化语言相类似，由多个方块组成，即所谓的功能块，各个功能块之间以线条相连接。功能块用矩形表示，每个功能块的左侧有不少于一个的输入端，右侧有不少于一个的输出端，并且上部标有该功能块的功能块实例名，内部还标有功能块的类型名称。在 FBD 网路中，线条代表的是信号的流向，如同梯形图中的

电力轨线一样，但是和梯形图的轨线有所不同的是，这些表示信号流向的连线不再是单纯的“ON”或“OFF”（导通或截止），它们所传递的信息可能是一个布尔数值、整型数值、实数或者是一个字符串，因此 FBD 语言适合于信息流和控制元件间信息的相互传递。

顺序功能图（SFC——Sequential Function Chart），它是一种面向图形的编程语言，适合于用户描述具有时间序列的不同作用的程序。SFC 的方法论来自人们所熟悉的 Petri 网和顺序（级联）方法论，在 IEC61131-3 标准将 SFC 作为一种编程语言标准之前，法国使用的 GRAFCET 和德国使用的 S5－GRAPH，S7－GRAPH 都是属于这种类型的编程语言。SFC 在 IEC61131-3 标准中的规划是用来作为组织程序内部构架，也就说将它作为一种逻辑化的方法来定义其中的顺序步骤以说明整体的应用架构，所以 SFC 可以用来设计顺序和并行过程。

指令表（IL——Instruction List）和结构化文本（ST——Structure Text）是两种文本化的语言。指令表是一种较“低级”的编程语言，它类似与计算机的汇编语言，常常作为其他文本化语言和图形语言转译过程中的公用中间语言。指令语言的代码由一系列的指令组成，而一条指令只能执行一个动作，且严格要求由一个行来表述，因此其可读性较差，被过多地应用在较小的程序或对程序空间有特别要求的场合，目前欧洲及东南亚地区的一些用户偏向于该语言的使用。

相比而言，结构化文本在 IEC61131-3 标准编程语言中则是一种高级语言，它类似于高级语言中的 C 语言、Pascal 语言等，有四十多组关键字和十种不同的状态，具有可读性高，程序结构化明显等特点。虽然如此，毕竟 ST 只是一种小型的工业控制高级语言，这种语言并不如梯形图那样受到广泛的应用。

IEC61131-3 标准所规定的编程语言是 IEC 工作组在对世界范围的 PLC 厂家的编程语言的合理吸收和借鉴的基础上形成的一套针对工业控制的系统的国际化的编程语言标准，这些语言标准不仅仅适用于目前的 PLC 系统，而且还涵盖了范围更加广泛的整个工业控制领域，例如 DCS、HMI 以及现场总线控制系统，并对它们的软件设计产生了很大的影响，这样有利于进一步地降低工程成本、统一文档、提高工程效率。

2.5. IEC61131-3 标准存在的不足[5,16]

IEC61131-3 标准在具备了众多优点的同时，也暴露出了自身的一些缺陷。

一方面 IEC61131-3 标准沿用了直接跟硬件相关的物理变量表示方法，这就在一定程度上妨碍了符合标准的 PLC 系统之间做到真正意义上的程序可移植性的优点。这一点表现在，如果想把在某一个厂商的 PLC 系统中运行成功的程序原样拷贝到另一个厂商的 PLC 中，用户要做的必须先从技术文件中找到有关与硬件资源

相关的变量的定义，然后再在另一个硬件资源中对此物理变量进行重新定义。但是，跟以前的编程方式相比，已经有了很大的提高，至少不存在与硬件相关变量之间的转换。

另一方面，目前的IEC61131-3标准只给出了单一的集中PLC系统的配置机制，所定义的按顺序调用块的方法也不是一种适用于分布式系统中的程序结构的方法，这显然不能适应分布式结构的软件要求。用户在实现了对单一资源配置管理的前提下，还期望能以图形方式显示程序的拓扑分布、程序的总体结构以及与分布式自动化项目的其他部分的互连，这些均发生于比前面所阐述的编程方式的层次更高，也更为抽象。

以上两方面的缺陷，并不能够通过几个人，几个厂商，甚至几个国家的努力就能克服的，必须在PLCopen组织的领导下，进行全面的规划和积极的推广。值得庆幸的是，IEC标准化工作组为此目标而进行了统一语言元素的工作，这就是正在制定的国际标准IEC61499[17]所规范的内容，该标准是对现IEC61131-3标准的进一步补充。

2.6. IEC61131-3标准的产品化[18]

正是由于IEC61131-3标准的公布，和国际PLCopen组织的长期努力，许多PLC制造商先后推出了符合该标准的PLC产品。美国A-B公司的许多PLC产品都带符合IEC61131-3标准中的结构文本的软件选项；德国西门子公司的SIMATIC S7-3000、S7-400、C7-620均采用了SIMATIC软件包，软件包中的梯形图部分完全符合IEC61131-3标准，而一些其他的可任选软件S7-SCL、S7-GRAPH则提供了IEC61131-3标准中的结构文本和顺序功能图编程方式。另外，还有德国Backhoff公司的WinCAT工控软件，Infoteam公司的OpenPCS软件以及法国CJ International公司的ISaGRAF软件等[19]，它们都在很大程度上向IEC61131-3标准靠近，而且基本或者已经完全满足了该标准对工控编程软件的各项要求。

特别的，法国International公司的ISaGRAF软件是目前符合IEC61131-3标准的三大组态软件之一[20,44]。在符合标准的同时，它也是世界上第一个用Windows平台为开发环境的软件平台，它支持全部5种PLC语言，具有强大的开放性和弹性。另外，它还具有程序的编辑，程序的纠错（Debugging），程序产生器（Code Generation），线上即时监视（On-line Monitoring）及离线的仿真（Off-simulation）等多种功能，并且还支持用户自定义的符合IEC61131-3标准的函数和功能块。ISaGRAF应用范围广泛，可以完成从简单的机械控制到复杂的高速过程控制的高效、高可靠性的控制工程。

需要指出的是，IEC61131-3标准对控制领域的影响并不仅限于PLC，它还使

用于 DCS、PC、现场总线控制，运动控制，甚至 SCADA 系统。国内在这方面有了很大的进步，最早引入 IEC61131-3 标准的是冶金部自动化研究院智能装备所，该所在其生产的 EIC2000 现场总线控制系统中采用了该标准。北京和利时公司也较早地认识到了 IEC61131-3 标准的重要性，并着力开发了基于该标准的控制系统，所以他们在 HS2000DCS 中就已经采用了梯形图、功能块和结构化文本的标准编程，并且在 1999 年推出的 FOCS 带现场总线的控制系统和 MACS 先进控制系统中，采用了 IEC61131-3 标准中的全部 5 种编程语言。另外，电力行业有上海新华控制工程公司的 XDPS-400 DCS 系统，化工行业有浙江威盛自动化公司推出的支持现场总线的 FB-3000 DCS 系统等，它们均采用了符合 IEC61131-3 标准的控制组态软件。实践证明，IEC61131-3 标准的引入势必将引起一场空前的工控技术革新，既促进了市场的繁荣，同时也加强了行业之间的良性竞争，采用先进的国际标准是自动控制厂商发展的必由之路[12]。

3. 系统分析与整体规划

本文的研究方向是针对 IEC61131-3 标准所提出的工业控制软件国际标准进行分析与实现，根据 IEC61131-3 所提出的各种规范和要求，尝试着设计并开发出一个视窗化的满足 IEC61131-3 标准的程序开发环境。在这一章中，将依据 IEC61131-3 标准对所要开发的软件系统进行全面的功能性分析，同时确定软件的开发模式和建模方法。

3.1. 功能需求分析[21,22]

IEC61131-3 所规范的是可编程控制器的程序语言部分，其中包含了五种方式的编程语言，分别是指令表 IL、结构化文本 ST、梯形图 ID、功能块图 FBD 和顺序功能图 SFC。这些语言被定义为可以自由地以交叉方式构成相应的应用软件，然后通过编译，将它们连接到单一的可执行的程序中，最后将生成的可执行程序下载到底层硬件之上执行。

由以上的分析可知，要开发出满足 IEC61131-3 标准的编程环境，必须从两方面来着手实施，一方面是编程方式上的要求，即必须提供 IEC61131-3 标准所支持的五种编程语言中至少一种方式的编程语言，这是使用者从界面上接触到的最直接的功能。另外一方面是软件的内在功能，这些功能不是直接显示在界面中，而是运行于软件的后台，是整个开发环境的后台程序，其功能包括，将图形化程序转换成可编译的程序代码、将所有的程序编译连接成单一的可执行程序、将可执行程序下载到硬件系统上执行以及进行相关的 I/O 状态连接以达到即时监视功能等。以下将分别就这两个方面进行讨论。

在 IEC61131-3 标准所支持的图形化和文本化的两种编程方式中，目前比较流行的是图形化的编辑方式，而其中的梯形图 LD 和功能块图 FBD 两种语言使用范围就更加广泛，遍布工业控制的各个领域。梯形图起源于美国，主要与 PLC 结合以取代继电器系统图，其符号表示法与原先的继电器线路图相似，因此受到工程师接收的程度高。功能块图（FBD——Function Block Diagram）是一个开放格式的图形编程语言，它起源于信号处理领域，对一个硬件工程师而言，它如同是一个电子电路图或逻辑电路图，FBD 作为一个普遍采用的语言其重要地位已经确立。因此，本论文在实现编程环境所支持的语言时，便以梯形图 LD 语言和功能块图 FBD 语言作为优先考虑，对于其他语言编程方式我们则预留了足够的空间用于今后功能的扩展。

为了满足 IEC61131-3 标准对编程环境各项功能的要求，我们将软件的内在功能划分为以下几个基本部分，分别是程序代码生成，编译，下载、在线监视。代

码生成功能是将编辑完成的图形或文本语言转换为 C 语言程序代码。编译功能是对生成的 C 语言代码进行编译、纠错，并进一步将其联结成下位机可执行代码格式文件。下载功能是将生成的可执行程序下载到 PLC 的存储器中去，以便程序的执行。在线监视功能则是通过串行通讯的数据连接，在软件界面上实时地反映程序地执行情况。

综合考虑 IEC61131-3 标准的要求，现将整个软件系统划分为以下基本功能块，如图 3-1 所示。其中最主要的有五大功能块，包括：程序编辑器、编译器、可执行程序下载、在线监视器和数据库。其具体功能划分如下：

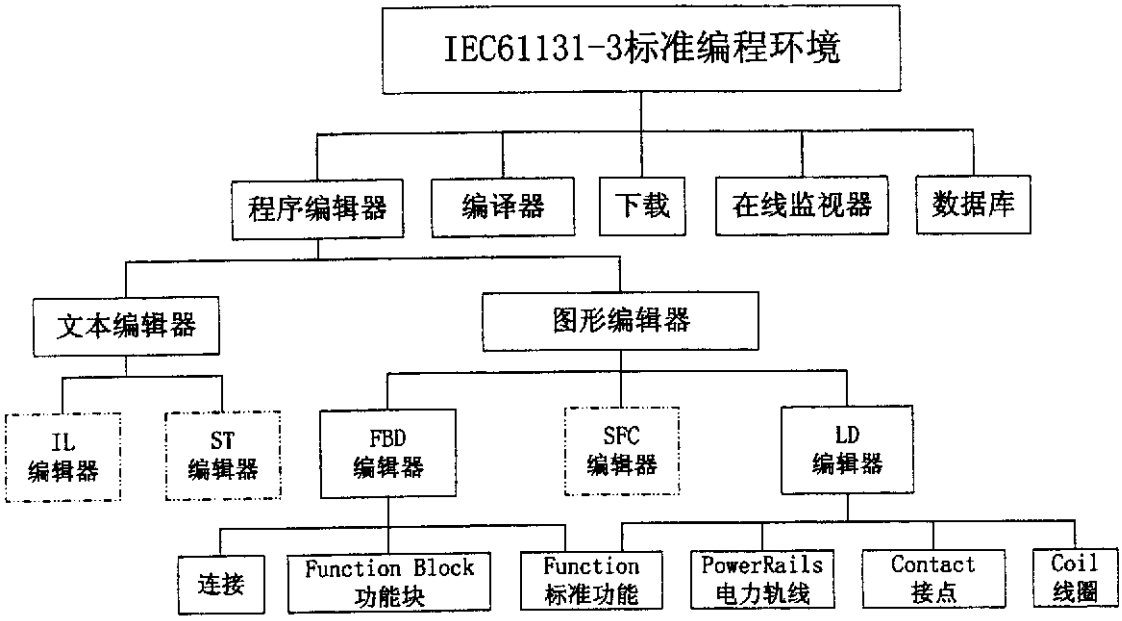

图 3-1：系统整体构成

- 程序编辑器：支持 IEC61131-3 五种编程语言中的梯形图 LD 和功能块图 FBD 语言编程，并提供方便快捷的组件拖动式的编程方式。
- 编译器：通过内含的代码生成器来产生与图形逻辑等效的 C 语言程序代码，并将所有纠错无误的档案连接成单一的可执行文件。
- 可执行程序下载：将生成的单一可执行程序下载到硬件系统；
- 在线监视：即时了解程序的执行情况和参数的运行情况；
- 数据库：提供对软件数据的集中管理、运算和存储；

除了以上五个主要功能模块外，软件系统还应具有其他的一些基本功能，包括项目文件的管理，PLC 的通讯以及在线帮助等功能，对于这些功能我们可以将其分别内嵌到了以上的主要功能模块当中，因此，在这里我们不单独进行讨论[3]。

3.2. 软件开发方法

3.2.1.软件开发方法模型[23]

软件成为工程化产品已经有三十多年的历史，软件开发方法也经历了不断的更新和发展，人们总结软件开发过程和经验，提出了许多软件开发方法的模型，其中最为流行的有结构化的开发方法、软件生命周期法、快速原型法和面向对象的迭代增量法。

a) 结构化设计方法（Structure Method）

人们在解决复杂问题的时候，常常都采用各个击破的方法，把一个大的复杂问题分解成若干小的子问题，逐个予以解决。结构化设计方法正是基于这种思想，它将大规模的程序至顶向下，逐步求精；采用模块化技术、分而治之的方法，将系统按功能分解成若干模块；模块内部由顺序、分支、循环基本控制结构组成；各部分独立建立，然后再组合。从 20 世纪 60 年代初提出结构化程序设计方法，到 20 世纪 70、80 年代结构化分析何结构化设计方法，一直到现在，在很长一段时期，结构化方法是应用最广泛的方法。

但由于结构化方法将过程和数据分离为相互独立的实体，程序员在编程时必须时刻考虑所要处理的数据的格式，而不得不针对不同格式的数据来编写具有相同功能的不同程序，这样就影响了结构化程序的可复用性。另一方面，由于数据与程序的相互独立，程序员常常要花额外的精力去保持数据与程序的相容，以避免发生错误的数据调用正确的程序模块或正确的数据调用错误的程序模块的情况。以上这些问题在软件开发过程中的不断积累，将导致完整的开发过程的无法控制，造成开发进度的失控。

b) 瀑布式软件生命周期法

瀑布式软件生命周期法把软件开发的整个过程严格地划分为6个阶段：需求分析、系统设计、结构设计、编码、测试和维护，对每个阶段和时期所要完成的任务和注意事项都作了详细的规定，程序员只有在一个阶段的任务完成后，才能进行到下一个阶段，这如同盖房子一般，要先由地基，才能继续往下做。这种方法规定了软件开发的阶段性和顺序性，从时间角度对软件开发和维护的复杂性问题进行了分解，是一种分时法。尽管这种方法是一种成熟的、被广泛使用的方法，然而，随着软件开发实践的加深和拓宽，也暴露出了自身的一些弱点。首先，这是一种理想化的，写在纸上的方法，实际的开发过程并非按着预先的设定顺序进行，期间还存在着大量的修改和反复。其次，在软件开发的不同阶段进行修改需要付出的代价是大不相同的，早期引入变动涉及的面积小、代价较低，若后期引入变动所需付出的代价会高2～3个数量级。另外，用户的需求本身具有模糊性，他

不能预先准确定义需求，开发的结果也会觉得与需求相距甚远。

c) 快速原型法

快速原型法采用快速的，低成本的方法建立反映实际应用系统主要功能的计算机模型——原型。它主要是借助一些软件开发工具或环境尽可能快地构造一个实际系统的简化模型。使用户在大规模的软件开发之前，能够尽快的看到目标系统的概貌。用户可利用原型进行验证和修改需求。这种方法的基本思想是开发人员与用户的不断交互，通过原型的演进不断适应用户需求的改变，不断改进和完善原型，从而能早期改正错误，加快程序的开发。与瀑布式模型相比，快速原型法更符合人类认识真理的过程和思维活动。

然而，原型法认为需求分析是个动态的定义过程，因此它对初始的需求分析比较重视。另一方面，它要求必须有快速建立系统原型模型的软件工具与环境。这两方面的要求从一定程度上限制了快速原型法的使用范围。

d) 面向对象的迭代增量法 [24]

面向对象的迭代增量法是新生的面向对象技术与快速原型法的有效结合。

众所周知，面向对象技术是软件工程领域中的重要技术，这种全新的软件开发思想出现于70年代末期。一方面，面向对象方法与现实世界存在着自然而且直接的对应关系，采用它为现实世界建模,能充分地描述与表达现实世界。另一方面，面向对象吸收了结构化的基本思想和主要优点，它将数据与操作放在了一起，作为一个相互依存、不可分割的整体来处理，综合了功能抽象和数据抽象，采用数据数据抽象和信息隐蔽技术。同时，面向对象方法将问题求解看做是一个分类演绎的过程，更贴近人们认识事物和解决问题的过程和思维方法。因此，该方法一经诞生就具有强大的生命力。

面向对象方法为大量复杂的大型软件工程提供比传统方法更为优越的新方法,并已逐步形成从面向对象分析(OOA)→面向对象设计(OOD)→面向对象编程语言(OOL)→面向对象系统(OOS)这样一套完整的软件开发方法学。

在实际的软件开发过程中，由于没有形式化的系统建模工具，使得快速原型法对软件开发人员的依赖性很大，再加上本身开发过程的随意性也很大，这给软件的维护带来了很大的隐患。面向对象(Objected－Oriented)技术的产生，使得很容易对快速原型方法进行改进，为大型复杂的软件找到一条准确与用户交互、弄清系统需求、缩短软件开发周期、降低成本、提高软件质量的有效途径。

针对以上分析，如果能将快速原型法和面向对象技术结合起来，就非常自然地解决了传统瀑布模型的不足，与此同时，将面向对象建模技术作为快速原型法的形式化工具，能够方便地构造出更好的初始原型，减少开发过程的随意性，防止一些垃圾原型的生成，这对提高开发效率，降低开发成本意义重大，用此方法

开发出的软件也更加易于维护和扩充。

面向对象的迭代增量法正是基于以上的功能而产生的，它把一个需要很长时间才能看到的大目标分为多个小目标，按相当小的增量构造软件，且总是保持一个可执行的构造。每一个迭代都可以得到一个可运行构架系统，每隔一段时间就能看到一定的改进。它是一种循序渐进的开发方法，使项目开发中的问题不至于完全遗留到项目的末期,使问题得以及时解决。

3.2.2.面向对象的迭代增量法的确定

在软件开发过程中，完整、正确地捕获并准确地描述用户需求，是软件项目需求工程的关键。开发人员都希望尽早得到正确的需求，但事实上需求在整个开发阶段都有可能变更[24]。在项目开发的早期，用户往往对系统只有一个模糊的想法，一般也不清楚计算机将如何实现，因此，很难完整地表述需求，往往需要先看到软件大概是个什么样子才能提出明确具体的需求。另一方面计算机开发人员又可能由于对业务不是很熟悉，在与用户沟通和捕获需求方面存在困难，用户和计算机人员之间的这种矛盾往往使需求分析工作信息失真或存在信息损失，这就意味着很难得到一个完整准确的或一成不变的软件需求规格说明。因此项目开发必须要有易于更新、易于增加新的需求的途径和方法[25]，并能评估变更所带来的后果。

瀑布式软件生命周期法代表的是理想的开发过程，是一种接力长跑式的作业方式，开发活动按阶段顺序进行。需求人员做完需求后交给分析设计人员进行分析设计，然后再交给编程人员编码实现，然后是测试。但实际的项目并非如此，开发者并不总是在完成一个阶段之后才开始下一阶段。有时他们常常会发现前一阶段的工作有错，因而有一部分必须重做。有些时候，开发人员需要重复进行分析、设计甚至编程的过程，需要有意识地反复进行，这些活动都是同时进行，不断迭代的。

面向对象增量迭代式的软件开发方法对于解决以上沟通困难，需求变更的问题，是一个理想的方法。与传统的瀑布式的软件设计方法相比，它不再是一口气做完所有的需求，而是将整个需求工程划分为多个阶段，分解到不同的迭代周期中去，在每个迭代周期中，需求工作一般都只需识别并描述部分功能的需求。并通过分析、设计、编码及测试，得到该迭代周期的阶段系统，即一个可以运行的构架基线。因此用户可以切切实实地看到整个软件系统到底像个什么样子，还可以启发他们提出更加明确具体的需求，从而方便了开发人员获得用户需求，以达到各方对系统的共同理解，使用户的业务需求得以准确描述出来。

迭代增量总是在原有的构架基线的基础上进行的，在每一次迭代周期中，可

能都有需求、分析、设计、编码及测试等活动，但它不是开发过程的简单重复。在不同的迭代周期中，它们并不是总占有相同比例的工作量。在早期的迭代中，需求或分析的工作量大些；随着迭代的推进，需求或分析的工作量逐渐减少；而在系统后期的迭代周期中，可能编码或测试的工作成为其主要工作。每一次迭代都总是在原有构架基础上阐明剩余的需求或新的需求，把它们添加到原有构架上(即将一个用例增量添加到已有的构架基线上)，并基于已建立基线的构架完成系统开发。因此，迭代增量的过程就是不断扩充完善构架的过程，每一次迭代的结果就是新的构架基线，一个新的向目标系统更进了一步的可运行构架系统。

下图 3-2 是在软件开发过程中运用迭代增量法的高层次视图[27]：

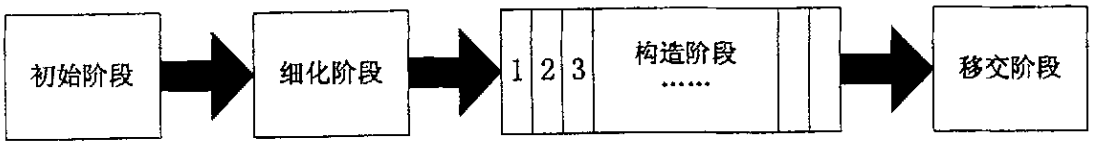

图 3-2：迭代增量式软件开发过程

如图所示，在项目开发过程中，首先要做的工作是选择一些功能点，然后完成这些功能，随后再选择别的功能点，如此循环往复。显然，作这个计划需要时间。前两个阶段是初始阶段和细化阶段。在初始阶段，需要考虑的是项目的效益，并确定项目的使用范围，这一阶段需要与用户进行讨论，在细化阶段，需要收集更为详细的需求，进行高层次的分析和设计，并为构造阶段制定计划。采用迭代增量式开发方法，不是在项目结束时一次性提交软件，而是分块逐次开发和提交。尤其是在构造阶段，它是由多次开发组成，每一次开发都包含编码、测试和集成，所得到的软件应满足项目需求的某一子集，或提交给早期的用户，或纯粹是内部提交，每一次迭代都包含了软件生命周期的所有阶段，即：分析、设计、实现和测试阶段。

由于本次课题研究的最终的目标是独立开发出一套符合 IEC61131-3 标准的程序编辑环境，虽然目前仅仅涉及到了 IEC61131-3 标准的五种语言中的两种（LD、FBD），但是，我们的还是以支持 IEC61131-3 标准所规范的五种语言为最终目标，因此，在软件开发过程中，我们必须留有足够大的空间用于今后的扩充，有待进一步的完善，由此决定了我们应该采用这种面向对象的迭代增量法作为本次软件系统的开发方法。

3.3. 统一建模语言——UML[28]

3.3.1.软件系统建模

建模是一项经过检验并广为接受的工程技术，已应用于社会生活以及科学研

究等众多领域，例如建筑业、机械制造业、经济学等等。对待解决问题进行建模，是由于人们对复杂问题的理解能力有限，通过建模对现实世界进行一定程度的抽象，可以大大简化所研究问题的难度。特别是当我们要解决一个大而复杂的系统时，可以采用“各个击破”的手段，将要解决的问题分解为一系列的小问题，即在建模过程中一次只注重研究它的一个方面的问题，这样解决了所有的小问题也就解决了整个问题。

软件开发的最终目的是得到满足一定要求的、具备特定功能的软件系统。在当今软件项目的复杂性和规模日益膨胀的情况下，要产生合格的软件就必须由一套关于系统结构、过程和工具的规范。如果没有对体系结构、过程和工具规范做任何考虑就毫无计划地着手实施软件开发，必然会导致软件项目的失败。为此，我们必须借助于建模活动来达到上述目的。

所谓建模，便是系统抽象化（abstraction）的过程，其目的在于建立可以描述系统功能的一组模型（Model），用来了解此系统，并作为实现该系统的依据。对软件系统建模有助于理解正在构造的系统，并在简化系统和软件复用方面提供很好的支持。最常用的软件建模方法有两种：一是从算法的角度建模；二是从面向对象的角度建模。传统的软件开发都是从算法的角度建模，而现代软件开发多是从面向对象的角度建模。

面向对象的建模语言出现于20世纪70年代中期，经过90年代初的“方法之争”，Booch方法、Coad/Yourdon方法、Jim Rumbaugh等人的OMT和Ivar Jacobson的OOSE方法在面向对象软件开发界得到了广泛的认可。具体而言，Booch方法在项目的设计和构造阶段的表达能力较强，OOSE方法对以用例作为一种途径来驱动需求捕获、分析和高层设计提供了极好的支持，而OMT适合于分析和描述数据密集型系统这些方法的主导思想基本一致，但表达形式存在较大的差异，给用户的选择带来了一定的困难。为了使面向对象方法向一致的方向发展，Grady Booch、James Rumbaugh和Ivar Jacobson三位面向对象领域的著名专家联合创立了统一建模语言（Unified Modeling Language，简称UML）。UML一经确立便得到了工业界和学术界的广泛支持和普遍的采用，在美国，截止1996年10月，已有700多家公司表示支持采用UML作为建模语言，1996年底，UML已稳定地占领了面向对象技术市场的85%，成为可视化建模语言事实上的工业标准。1997年11月，UML1.1版被OMG采纳作为基于面向对象技术的标准建模语言，目前UML最新的版本为2.0。

3.3.2.UML简介[29,30]

1. UML的定义

在OMG发布的白皮书中，UML的定义为：UML是一种用于对软件密集型系

统制品进行可视化、详述、构造和文档化的可视化建模语言，主要适用于软件开发的分析和设计阶段，主要特点是表达能力丰富。标准建模语言 UML 的定义包括 UML 语义和 UML 表示法两个部分。

- UML 语义：描述基于 UML 的精确元模型。元模型为 UML 的所有元素在语法和语义上提供了简单、一致、通用的定义性说明，使开发者在语义上取得了一致，消除了各种因人而异的表达方法所造成的不良影响。此外，UML 语义还支持对元模型的扩展定义。
- UML 表示法：定义了 UML 的表示符号，为建模者和建模支持工具的开发者提供了标准的图形符号和正文语法。这些图形符号和文字所表达的是应用级的模型，在语义上它是 UML 的元模型的实例。使用这些图形符号和正文语法为系统建模可建造出标准的系统模型。

这里，需要指出的是 UML 是一种建模语言，而不是一种方法。在原理上，任何方法都应该由建模语言和建模过程两部分所构成。其中，建模语言提供的这种方法用于表示设计的符号（通常是图形符号）；建模过程则描述进行设计所需要遵循的步骤。标准建模语言 UML 统一了面向对象建模的基本概念、术语及其图形符号，为人们建立了便于交流的共同的语言。然而，人们可以根据所开发的软件的类型、环境和条件，选择不同的建模过程[27]。

2. UML 的构成[31]

UML 语义构造块也就是 UML 模型元素，是用来抽象、描述现实世界的 UML 的语义部分。UML 的模型元素由三部分组成：事物，关系和图。事物是建模元素本身，是对模型中最具代表性的成分的抽象；关系把事物结合在一起，用来说明两个或多个事物是如何语义相关的；图是 UML 模型的视图，它们展现事物的集合。

(1) 事物

UML 中描述语义的四种事物：结构事物、行为事物、分组事物、注释事物。

a) 结构事物

结构事物是 UML 模型中的名词，它们通常是模型的静态部分，描述概念或物理元素。共有七种结构事物：类（Class）、接口（Interface）、协作（Collaboration）、用例（Use Case）、主动类（Active Class）、组件（Component）和结点（Node）。

b) 行为事物

行为事物是 UML 模型的动态部分。它们是模型中的动词，描述了跨越时间和空间的行为。行为事物在 UML 中分两类：交互和状态机。

交互是在特定的语境中共同完成一定任务的一组对象之间交换消息的动作集合体。

状态机则是这样一种行为，它描述了一个对象或一个交互在生命周期内响应

事件所经历的状态序列。交互和状态机两种元素是可以包含在 UML 模型中的基本行为事物。在语义上，这些元素通常与各种结构元素（主要是类、协作和对象）相关。

c) 分组事物

分组事物是 UML 模型的组织部分。它们是一些由模型分解成的“盒子”。在所有的分组事物中，最主要的是包（Package）。包是把元素组织成组的机制。它把语义上相关的建模元素分别作为内聚的单元，是用来组织 UML 模型的基本分组事物。它有诸如框架、模型和子系统等变体。

d) 注释事物

注释事物是 UML 模型的解释部分。这些注释事物用来描述、说明和标注模型的任何事物，如注释（Note）。

(2) 关系[32]

UML 中的各种事物是通过关系结合在一起的。在 UML 中模型构造块之间的基本关系有四种：依赖（Dependency）、关联（Association）、泛化（Generalization）和实现（Realization）。

依赖是描述两个事物之间的一种语义联系，其中一个事物变化是会影响或者提供信息给另一依赖的事物。UML 中描述了四种基本依赖类型：使用（Usage）、抽象（Abstraction）、授权（Permission）、绑定（Binding）。

关联描述了一种链，是指模型块之间的结构联系，两者存在结构性的链接（Link）。寻找关联的依据是：如果两个对象之间存在链接，这些对象的类之间必定存在关联。有一种特殊的关联叫聚合（Aggregation），表示结构的整体和部分之间的关系。

泛化是表示一种特殊与一般的关系，特殊元素完全符合一般元素，但包含更多的信息。在很多面向对象的编程语言中，这种关系被称作继承（Inheritance）。

实现指定了一类元对另一类元保证执行的契约，其中一个规定了一组约定（协议），另一个负责实现它们。实现多用于接口和类或组件之间，用例和实现它们的协作之间。

下图 3-3 给出了 UML 中的各种关系的图形表达方法：

图 3-3：UML 中关系的图形表示法

(3) 图

图（Diagram）是一组元素的图形表示，是模型元素的图形化符号。在 UML

中，提供了以下九种基本的模型图：用例图（Use Case Diagrams）、类图（Class Diagrams）、对象图（Object Diagrams）、状态图（State Diagrams）、顺序图（Sequence Diagrams）、协作图（Interaction Diagrams）、活动图（Activity Diagrams）、部件图（Component Diagrams）和部署图（Deployment Diagrams）。通过这九种可视化的图形，可对世界上的任何复杂的事物进行可视化描述，从而充分显示了 UML 的灵活性。

a) 用例图（Use Case Diagrams），用于显示若干角色及这些角色与系统提供的用例之间的关系。其中，角色是与系统进行交互的外部实体，可以是系统用户，也可以是其他系统或硬件设备；用例是系统提供的功能。
b) 类图（Class Diagrams），用来表示系统中的类和类之间的关系，如关联、依赖、聚合等，也包括类的内部结构（类的属性和操作）。类图描述的是一种静态关系，所以在整个系统的整个生命周期都是有效的。
c) 对象图（Object Diagrams），是类图的实例。因为对象的存在是有生命周期的，所以对象图只能在系统的某一时间段存在。
d) 状态图（State Diagrams），用来描述类的对象所有可能的状态及时间发生时的状态的转移条件。一个状态图就是一系列状态以及状态之间的转移。通常，用状态图来表示单个对象在生命周期中的行为。
e) 活动图（Activity Diagrams），描述满足用例要求所要进行的活动以及活动之间的约束关系，有利于识别并行活动。
f) 顺序图（Sequence Diagrams），描述对象之间的动态的交互关系，它强调对象之间消息发送的顺序，同时显示对象之间的交互。
g) 协作图（Interaction Diagrams），描述相互合作的对象之间的交互关系和连接关系。
h) 部件图（Component Diagrams），用来反映代码的物理结构。各部件之间也存在关系，它可以方便的分析一个部件的变换会给其他的部件所带来的影响。
i) 配置图（Deployment Diagrams），定义系统中软硬件的物理拓扑机构以及在此机构之上执行的软件。它可以显示实际的计算机和设备之间的连接关系，也可以显示连接的类型及部件之间的依赖性，还可以显示网络之间的通信路径。

以上的九种视图就是 UML 表达语义的最基本的符号。其中，用例图、类图、对象图、部件图和配置图是静态模型视图，描述 UML 的静态建模机制，状态图、活动图、顺序图、协作图是动态模型视图，描述的是 UML 的动态建模机制。

从应用的角度看，当采用面向对象技术设计系统时，首先是描述需求；其次根据需求建立系统的静态模型，以构造系统的结构；第三步是描述系统的行为。其中在第一步与第二步中所建立的模型都是静态的，包括用例图、类图、对象图、部件

图和配置图等五个图形，是标准建模语言 UML 的静态建模机制。其中第三步中所建立的模型或者可以执行，或者表示执行时的时序状态或交互关系。它包括状态图、活动图、顺序图和协作图等四个图形，是标准建模语言 UML 的动态建模机制。因此，标准建模语言 UML 的主要内容也可以归纳为静态建模机制和动态建模机制两大类[33]。

3.3.3.UML 的应用领域[34]

UML 的目标是以面向对象的方式来描述任何类型的系统，具有很宽的应用领域。其中最常见的是建立软件系统的模型，适用于系统开发过程中从需求到测试的软件开发生命周期全过程。

1. 需求分析阶段。可以用用例捕获用户需求，通过用例建模，描述对系统感兴趣的外部角色和他们对系统的功能要求。
2. 分析阶段。主要关心问题域中的主要概念（抽象、类和对象）和机制，需要识别这些类以及它们相互间的关系，并用 UML 类图来描述。
3. 设计阶段。任务是通过综合考虑所有的技术限制，扩展和细化分析阶段的模型，并得到可行的技术解决方案。
4. 实现阶段，又称为构造阶段。是对类进行编程的过程。其任务是选择合适的面向对象编程语言将来将设计阶段的类转换成实际的代码。在实现阶段，可以选取下列图的说明来辅助编程：
   - 用例图和规格说明：显示系统需求和结果。
   - 类图：显示类的静态结构和类之间的关系。
   - 类的规格说明：每个类的规格说明详细显示了必要的属性和操作。
   - 状态图：显示类的对象可能的状态，所需处理的转移以及触发这些转移的操作。
5. 测试。在完成了构造过程之后，UML 模型还是测试阶段的依据。系统常常需要经过单元测试、集成测试、系统测试和验收测试。

此外，UML 还可以用于描述不带任何软件的机械系统、一个企业的机构或企业过程等，如处理复杂数据的信息系统、具有实时要求的工业系统或工业过程、嵌入式实时系统、分布式系统、系统软件、商业系统等。简而言之，UML 是一个通用的标准建模语言，可以对任何具有静态和动态行为的系统进行建模。

3.4. 系统开发工具的选择[35]

综合考虑需求分析和系统开发的周期长短，决定采用 C++ Builder 6.0 作为前台开发工具[35]以及它本身所提供的本地数据库 Paradox 为数据库系统。

C++ Builder 是 Inprise 公司推出的基于 C++语言和可视化组件库（VCL）的全新可视化编程环境，它是最早推出的面向 C++的真正 RAD（Rapid Application Development）工具之一，也是唯一提供真正基于组件拖放式编程的 RAD 工具。

C++ Builder 开发环境中集成了众多功能强大的组件，适用于 32 位 Windows 应用程序的快速开发。利用 C++ Builder 编程，可以实现用最小的代码开销编写出高效率的 Windows 应用程序，是一个十分理想的软件开发平台。另外它还引入了如下的概念：

1. 它引入了类的概念，具有非常贴近现实实体的表示特点，方便了较为复杂程序的设计。
2. 引入了类模板的概念，并在此基础上建立了包容类，提供了抽象数据类型 ADT。

我们在对数据库的选择上，考虑到 C++ Builder 6.0 内嵌的数据库接口 BDE（Borland Database Database Engine）直接提供了对 Paradox 数据库的引擎驱动，选择 Paradox 作为系统的数据库，就省去了解决多线程存取数据时产生的数据锁定的问题[37]。

综上所述，选择 Paradox 作为系统的数据库和 C++ Builder 6.0 作为前台开发工具是可行的，完全有能力在较短的时间内开发出满足系统要求的软件。

4. 系统的 UML 建模与实现

在完成系统分析和设计工作之后，此次 IEC61131-3 编程环境系统就可以开始实施编码了，本章将利用 UML 提供的各种图形工具对软件系统进行建模，并利用高级语言工具 C++ Builder 6.0 对编辑器功能块中的 LD 语言和 FBD 语言环境进行实现[28,38]。

4.1. IEC61131-3 标准的要求[7,39]

IEC61131-3 是一套工业控制系统软件设计的国际化标准，此次在设计“基于 IEC61131-3 标准的编程环境”时，为了尽可能的满足这一国际化的标准，在进行软件开发之前，我们必须充分了解 IEC61131-3 标准对工业控制软件的各项要求，并以此作为整个软件设计的依据和重点。以下便是对 IEC61131-3 标准研究之后，作出的一些归纳：

1. 编程语言。IEC61131-3 允许使用者使用图形化和文本化两种方式的编辑环境，其中图形化的编程语言有 LD、FBD 和 SFC，文本化的编程语言有 IL 和 ST。
2. 变量的使用。IEC61131-3 标准允许使用者定义变量的名称、类型（数据类型和作用域）和地址。
3. 在线监视功能。程序在执行时，可以实时地进行在线监视（Online Monitoring）。
4. 对于各种语言编辑环境，必须提供基本的语言元素，例如 LD 语言编辑器中必须有电力轨线、线圈和接点。
5. 充分体现程序的结构化和可重复使用。
6. 快速、便捷的可视化程序开发环境。

4.2. 静态结构模型图的建立

4.2.1.用例图（Use Case Diagrams）的建立

通过上一节对 IEC61131-3 标准要求的分析，我们基本明确了该系统的基本功能，接下来将根据这些基本功能来建立“基于 IEC61131-3 标准编程环境”的用例图。用例图描述的是系统外部的执行者与系统提供的用例之间的某种联系。用例模型包括以下组成部分[40]：

- 角色（Actor）——角色代表了以某种方式与系统交互的人或事。根据定义，他们是在系统外的。我们将注意力放在参与者上，是为了确保系统可以做出有意义的事来。
- 用例——用例代表了系统为其角色所进行的有价值的操作。它并不是简单的功

能或特性，也不能进行分解。用例有一个名称和简要的说明，还拥有一些详细的描述，这些描述从更本上阐述了角色如何使用系统来做他们认为重要的事情，以及系统为了满足这些需要而做的事情。

首先，我们了解软件与环境之间的关系，找出直接操作的对象，即 UML 中所谓的“角色（Actors）”。“角色”用于描述与系统功能有关系的外部实体。需要注意，“角色”并不一定都是人所扮演的角色，也可以是外部系统。这里我们从工作的项目中可以分析出程序编辑人员、系统运行人员、受控对象 PLC 和控制程序的编制有关，它们就组成了外部的“角色（Actors）”，通过进一步的分析我们建立起了系统的用例模型，如图 4-1 所示。

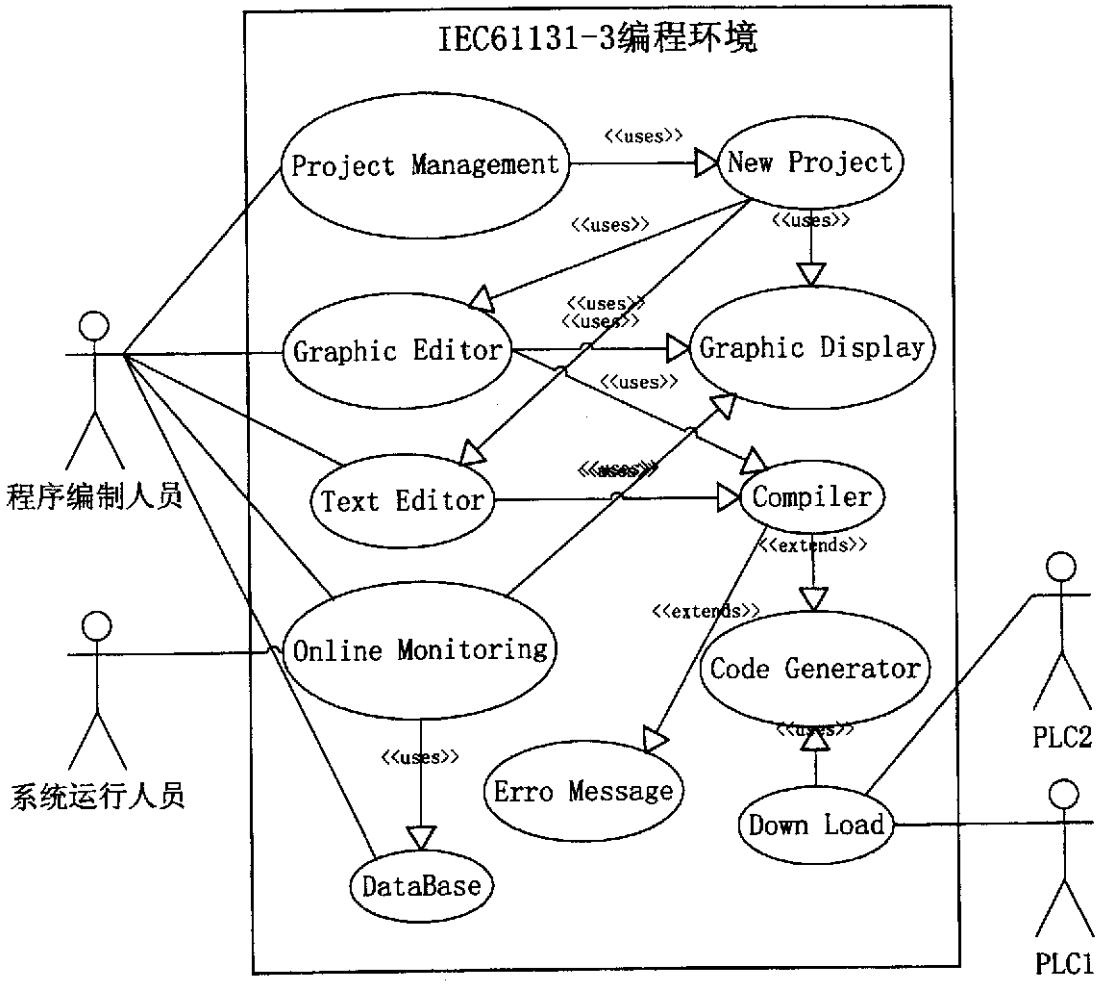

图4-1：IEC61131-3标准编程环境—UML用例图

图 4-1 中项目管理器（Project Management）用来管理项目文件，每个项目文件中又含有一个或多个程序文件。程序编制人员可以通过项目管理器来创建和管理项目，通过图形编辑器（Graphic Editor）来编辑 LD 或 FBD 程序，也可利用文本编辑器（Text Editor）来编辑 IL 和 ST 两种文本化的程序（有待实现）。在程序

编辑完成后，图形化程序又可以由图形显示器功能将图形程序显示出来。当所有的程序编辑完成后，经代码生成器（Code Generator）产生 C 语言的程序格式码，并利用 C 语言的编译器（Compiler）将所有的程序文件进行编译，生成可执行程序，最后利用下载功能（Down Load）通过与外部的串行通讯连接（RS232 串口通讯）下载到外部的“角色”中去。

此外，当外部的“角色”（PLC）在执行程序的过程中，会通过专门的通讯模块将状态参数以字符串的形式送出，上位机软件将接收到的字符串数据送到专门的数据库（Database）中进行分析，并且进行实时的数据存储，再利用在线监视功能（Online Monitoring）将数据与图形程序联系起来以图形的方式显示在画面上。

4.2.2.类别图（Class Diagrams）的建立[41,47]

在需求分析阶段所建立的需求模式，主要是描述系统的外部对象与系统之间的行为。用例图的建立表示了系统外部的行为者（Actors）与系统内部的用例（Case）之间的动作，这只是从用户的观点完成的需求分析。完成用例图之后，必须进一步描述系统内部的结构，因此，我们有必要根据用例图中的各个用例对象，从结构的观点出发，建立描述系统结构的类别图（Class Diagrams）。

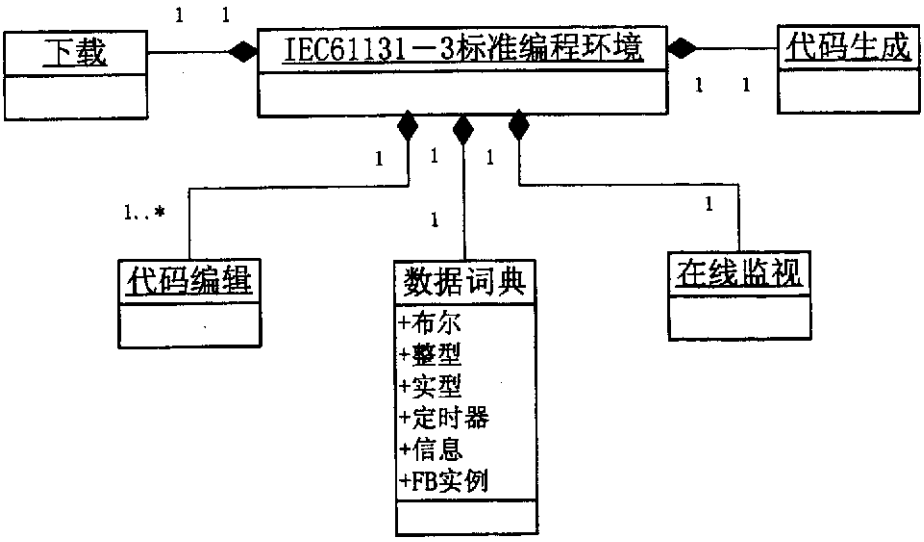

图 4-2：IEC61131-3 标准编程环境类别图

类别图展现的是一组对象、接口、协作之间的关系。在面向对象的系统建模中所建立的最常见的视图就是类别图。类别图给出的是系统静态设计视图。类与类之间的关系都体现在类别图的内部结构之中，并通过类的属性和操作等术语反映出来。一个典型的系统中通常包含有若干个类图，一个类图不一定包含系统所有的类，一个类可以加到几个类图中。

由用例图 4-1，我们可以得出 IEC61131-3 标准编程环境的基本类别图，如图 4-2 所示。以下分别就这五部分进行简单介绍：

数据词典（Data Dictionary），用于集中创建和管理程序中的变量。在程序编制之前或者编制过程中，可以随时调用该功能对象，来定义程序中所用到的变量的名称、类型、初始值和地址等属性。我们将 IEC61131-3 标准所规定的变量的类型划分为了以下六种：布尔变量、整型变量、实型变量、定时器变量、信息变量和 FB 实例。

代码生成器（Code Generator），本对象的主要功能是利用中间转换的方式，将图形编辑器种所编辑的图形化程序，转换成对应的 C 语言程序代码，并且驱动编译程序对所产生的代码进行编译和纠错。如果生成的代码无误，则直接生成可执行文件，若生成的代码有错误，则将错误信息以对话框的形式展现给用户，并提示错误信息，以便用户进行修改。

在现监视器（Online Monitoring），主要是由数据库中提取由 PLC 串行上传的实时数据，以动态的图形方式或文本方式显示在编辑界面上，便于用户进行在线监视 PLC 的运行状态。

下载（Download），用于将编译器生成的可执行代码经串行通讯连接下载到 PLC 的程序存储器中去，并且提供驱动该程序在 PLC 上运行。

对于代码编辑器中的图形编辑器，我们对其进行了进一步的分解，从而建立出更为详细的两种图形语言（LD 和 FBD）编辑器类别图，如图 4-3 和 4-4 所示：

图 4-3 所示的为梯形图程序编辑器类别图，图中给出了各个内部类（Class）的类别标识/命名，类的属性、类的操作以及类之间的关系。梯形图的主要类有接点（Contact）、线圈（Coil）和电力轨线（Power Rail），这三个子类都将继承来自 LD 对象这个父类，对于父类 LD 对象的所有的特性和操作，接点、线圈和电力轨线都将拥有。与此同时，由接点和线圈构成的父类又将派生出下一级的子类。具体的是由接点类派生出常开接点、常闭接点、上升沿接点和下降沿接点，线圈类派生出一般线圈、方向线圈、置位线圈以及复位线圈。对于它们之间的关系，在类图中也进行了相应的标识，其构成法则是对于一个 LD 对象必须有 2 条电力规线，在两条电力轨线之间（即左电力轨线与右电力轨线），必须有线圈（1...*），线圈之前可以有 0 个以上（0...*）的接点。

图 4-4 是功能块图语言编辑器类别图，在类的继承方式上与梯形图编辑器类相似，只是在子类的属性和操作上有所不同。这里不再一一介绍。

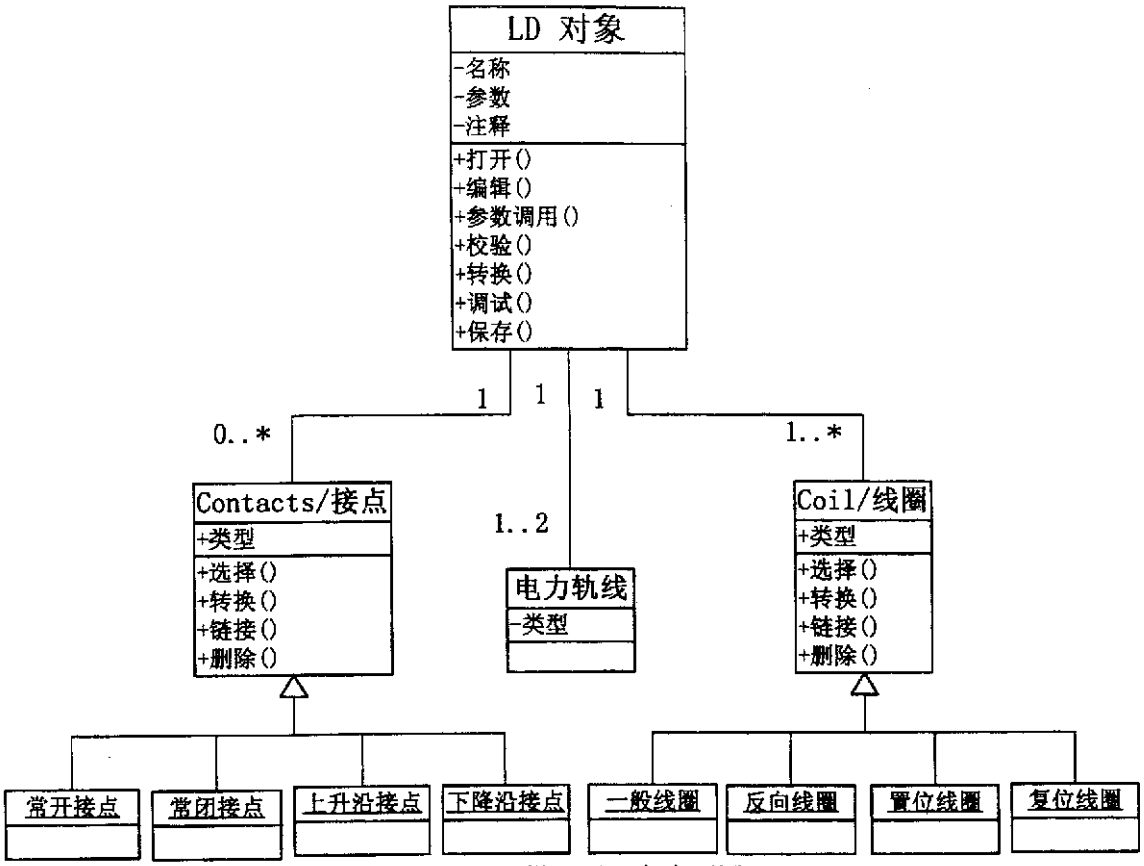

图 4-3：LD/梯形图语言类别图

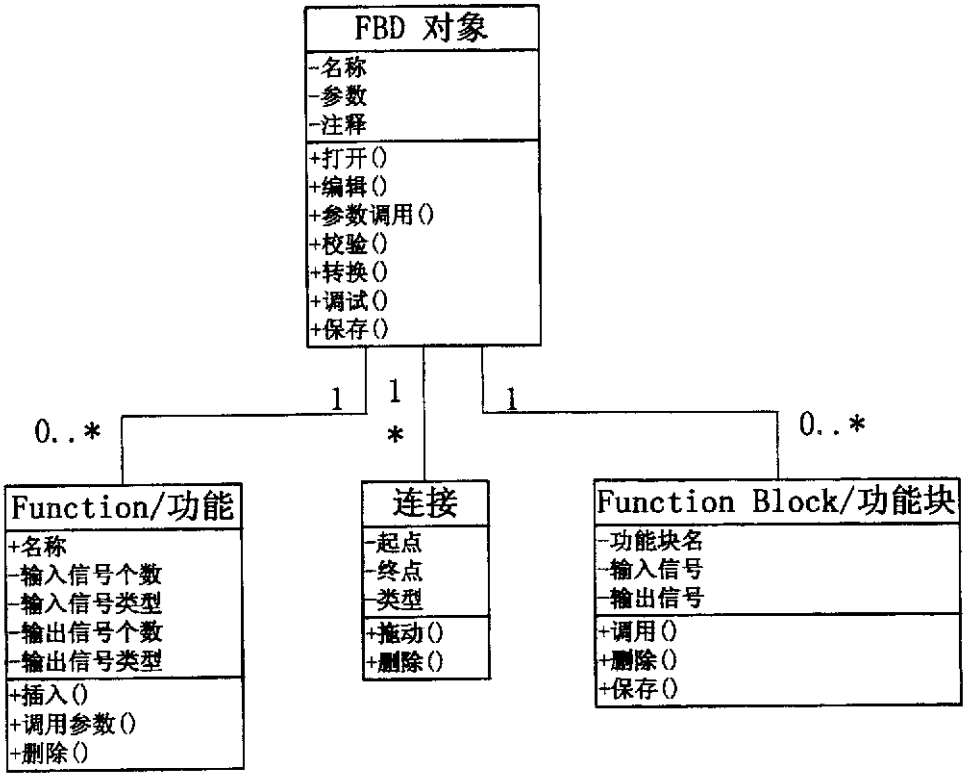

图 4-4：FBD/功能块图语言类别图

4.2.3.配置图（Deployment Diagrams）的建立

配置图是用来描述系统硬件的物理拓扑结构以及在此基础上执行的软件。它可以清楚地描述硬件设备的配置、通信以及在各硬件设备上各种软件构件和对象的配置。下图 4-5 就是标准编程环境的部署图，整个软件系统是安装在个人计算机的 Windows 2000NT 系统之中，它包括程序编译器构件、标准语言编辑器构件、在线监视器构件、通讯下载构件和数据库构件五大部分。

在硬件配置上，标准编程系统利用计算机串口（1 或 2）与下位机 PLC 的串口建立起 RS-232 串行通讯连接。在线监视器功能则是通过数据库构件与 PLC 数据存储单元构件之间的数据传输来实现。

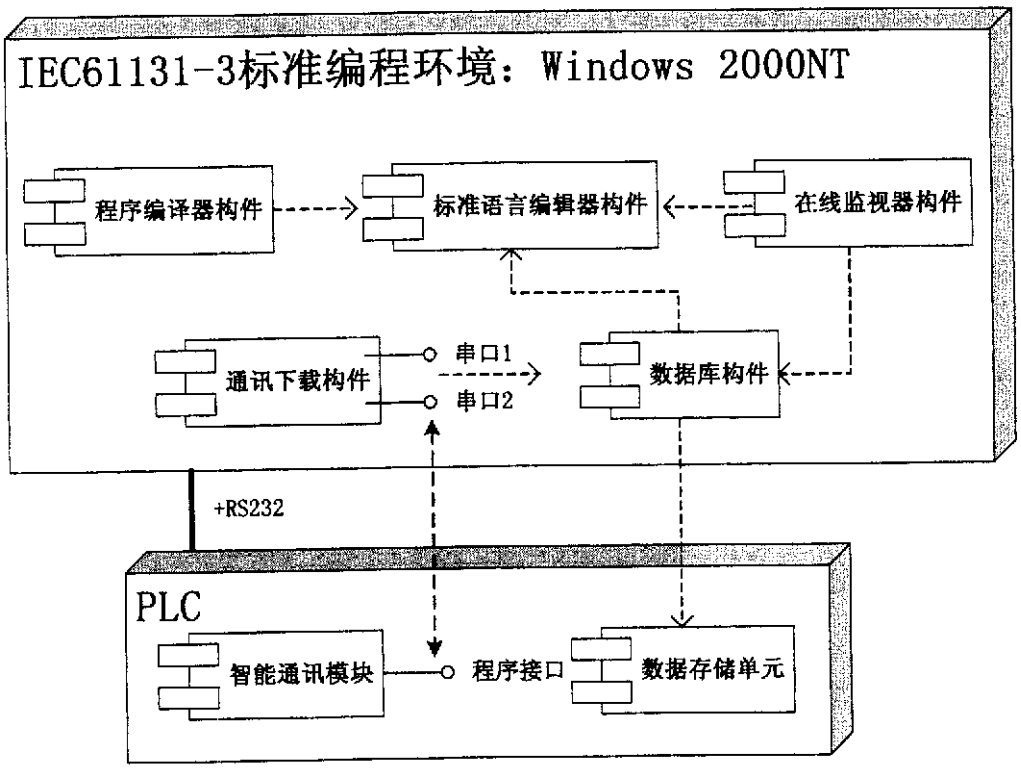

图 4-5： IEC61131-3 标准编程环境的部署图

4.3. 动态结构模型图的建立

4.3.1.交互图（Interaction Diagrams）的建立

用例图、类别图和配置图的建立，只是帮我们静态地分析了系统本身的框架以及与系统外界的关系。我们需要进一步明确地表示对象之间的信息传递关系，此时，需要建立系统的交互图。

交互图主要是用来描述对象之间的动态合作关系以及合作过程中的行为次

序。它常常用来描述一个用例的行为，显示该用例中所涉及的对象和这些对象之间的消息传递情况。交互图有两种形式，即顺序图和合作图，它们从不同的侧面来描述对象间的交互关系。选择哪一种形式的交互图根据不同人的喜好而定，但是一个最基本的原则就是哪种图更简明清楚则选择哪种图。以下是我们建立的系统各个功能对象之间的顺序图。

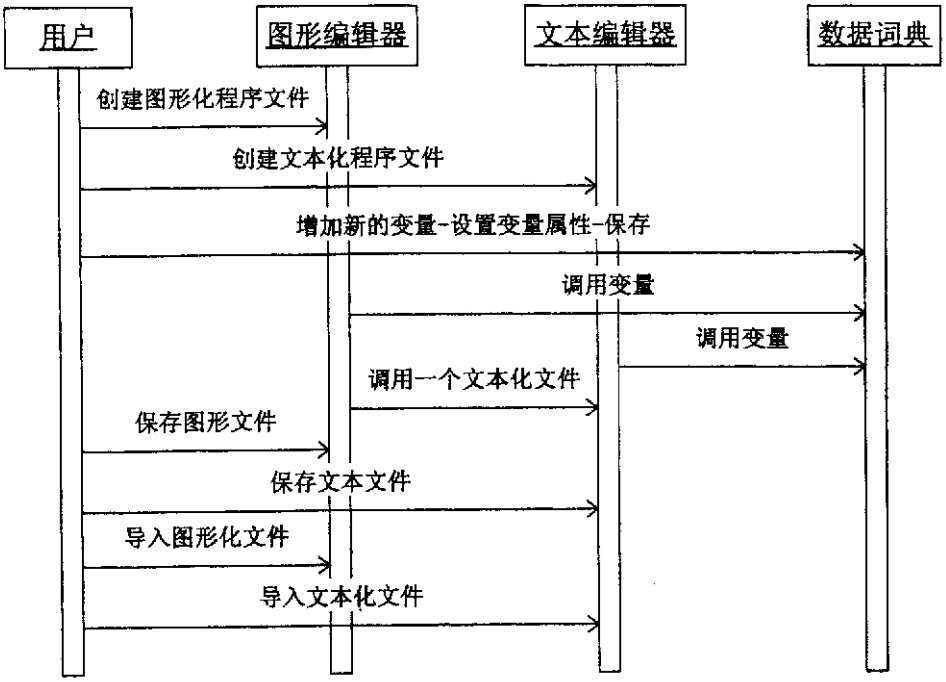

图 4-6：编辑环境交互图

图 4-6 表示了在编辑器环境中，各个子功能模块之间的交互过程。它描述了从用户创建一个文件（图形文件或文本文件）到在编辑过程中往数据词典中添加新的变量，设置变量的属性，再到变量的调用以及图形文件和文本文件的相互交叉编辑，最后再将编辑完的程序文件存档这一用户活动过程中对象间的消息传递。

图 4-7 所描述的是将用户编辑完成的程序进行编译，并进一步下载到控制装置（PLC）中去的编译、下载过程交互图。在对整个软件的操作过程中，我们将使用程序编辑器编写程序的过程作为单独的过程，进行交互图的建立，编辑环境交互图已经向我们展示了这一过程。接下来，用户向软件发出对编译的请求时，必须由图形或文本编辑器来驱动内部的编译器功能，编译完成后编译器会以对话框的形式回复用户编译信息，其中，涉及到编译进程、错误报告、警告报告、文件大小和变量的信息等等。

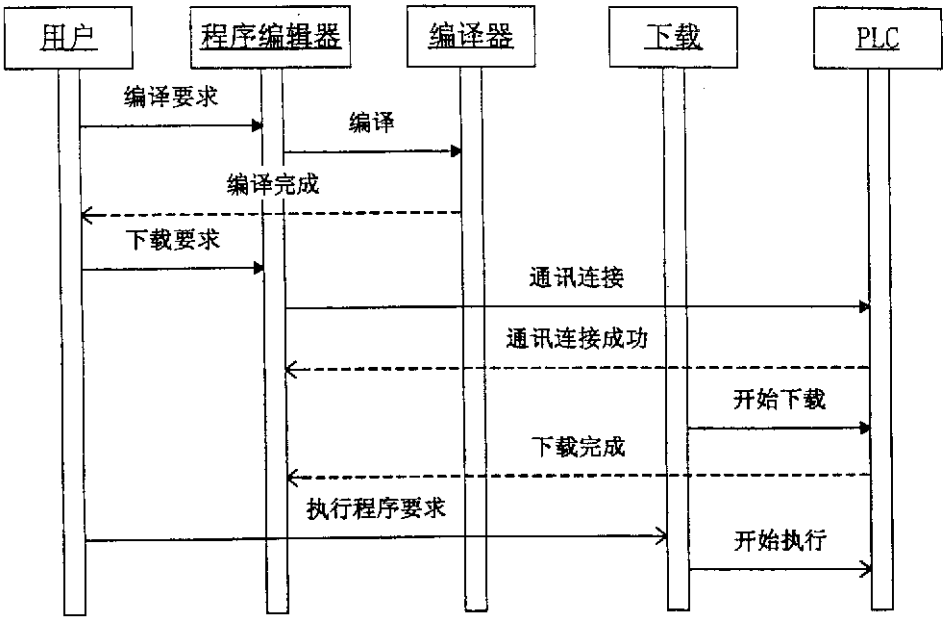

图 4-7：编译、下载过程交互图

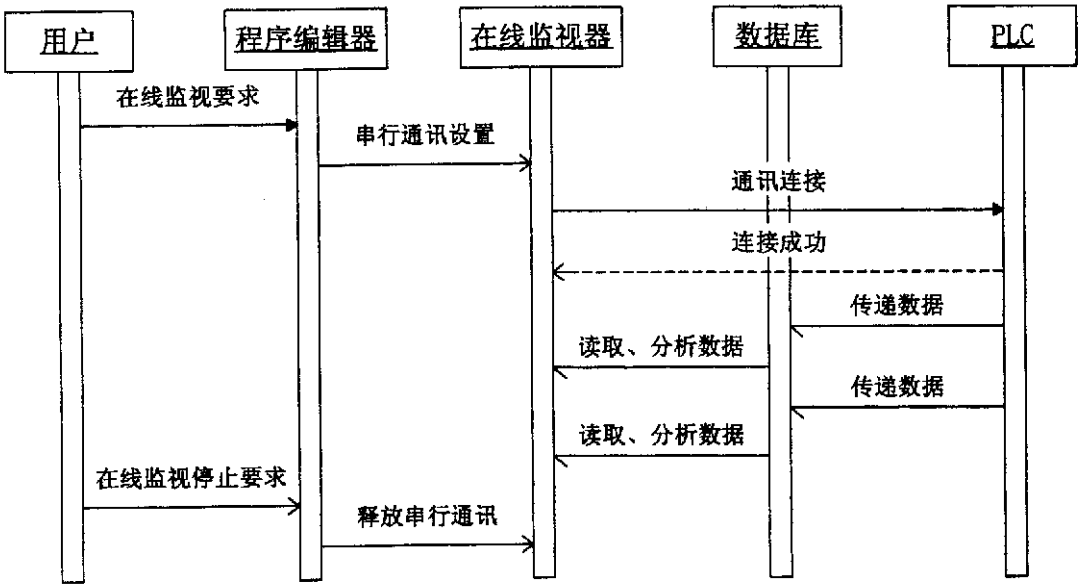

图 4-8：在线监视交互图

图 4-8 是使用在线监视器功能时的各个物件之间的信息交互过程。当使用者通过图形或文本编辑器来驱动在线监视这一功能时，首先必须确认程序已经下载到 PLC 中去，且 PLC 此时处与运行状态。其次，要建立上位机软件系统与 PLC 的串行通讯连接，包括通讯方式的选择，波特率的设置，数据格式的确定。当成功建立起通讯连接后，软件才可以接收来自 PLC 的数据信息，并在软件的专用数据库中建立起对现场数据的分析、存储和更新，经在线监视器（Online Monitoring）对数据库中的数据进行相关的链接和调用，并以动态的图形方式或文本方式显示在图形编辑器上，以实现对现场运行状态的在线监视。

4.3.2.状态图（State Diagrams）的建立

状态图适合于描述跨越多个用例的单个对象的行为，而不适合描述多个对象之间的行为协作。这就是为什么常常将状态图与其他的技术（交互图或活动图）组合使用来描述对象的动态行为的原因。然而，对于系统运行方式的描述，状态图是一种非常好的手段,它描述了一个特定对象的所有可能的状态以及由于各种时间的发生而引起状态之间的转移。

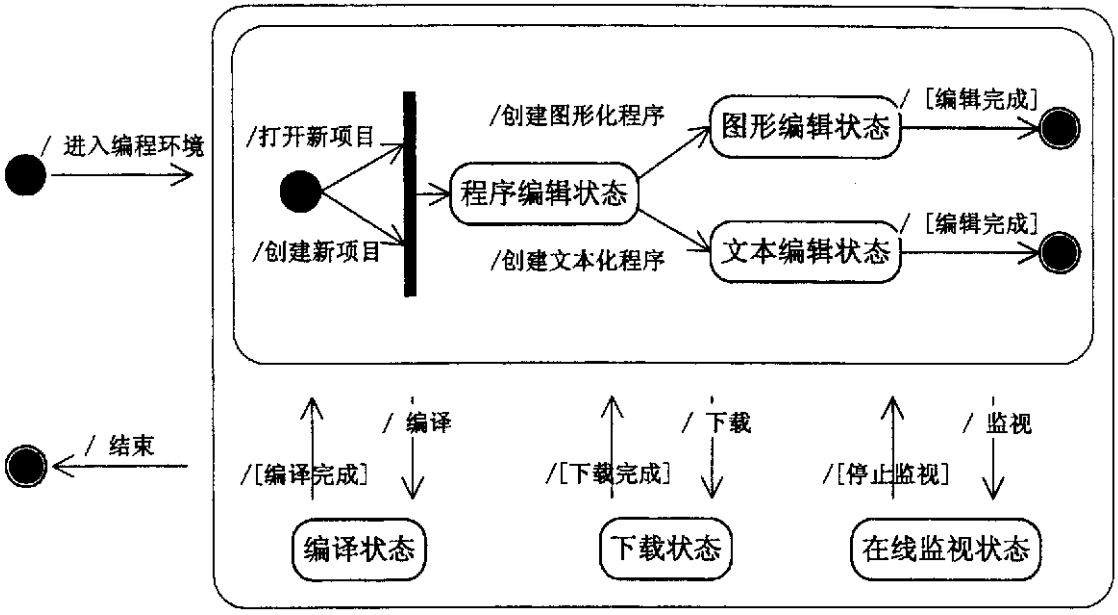

图 4-9：基于 IEC61131-3 标准的编程环境系统状态图

图 4-9 就是基于 IEC61131-3 标准编程环境的系统状态图。分别描述了软件系统在不同的功能对象中所处的不同状态，以及它们之间的转移情况。首先，软件一开始便进入程序编辑器对象中，根据编程方式的不同要求来选择进入相应的程序编辑环境（图形化编辑环境或文本化编辑环境）。在程序编辑完成后，由用户下达不同的命令，包括编译（Compile）、下载（Download）或在线监视（Online Monitoring），分别转移到相应的功能状态下。一般整个软件的状态过程是，用户先编辑程序，编辑完成后便可对程序进行编译，编译完成且程序纠错无误后便可将生成的可执行程序下载到 PLC 上。需要注意的是只有当 PLC 处于程序运行状态时，才可直接调用在线监视功能。

在编辑器状态图 4-10 中，当用户进入图形编辑器状态时，选择所要求的程序编辑语言并进入到相应的编辑环境。在任意的编辑环境中，用户只需调用相应的菜单选项“进入 FBD 编辑环境”或“进入 LD 编辑环境”（两者相互锁定），可以随时改变当前的编辑环境。这两种编辑状态的相互调用，体现了 IEC61131-3 标准中对不同种语言实现交叉编辑的要求。

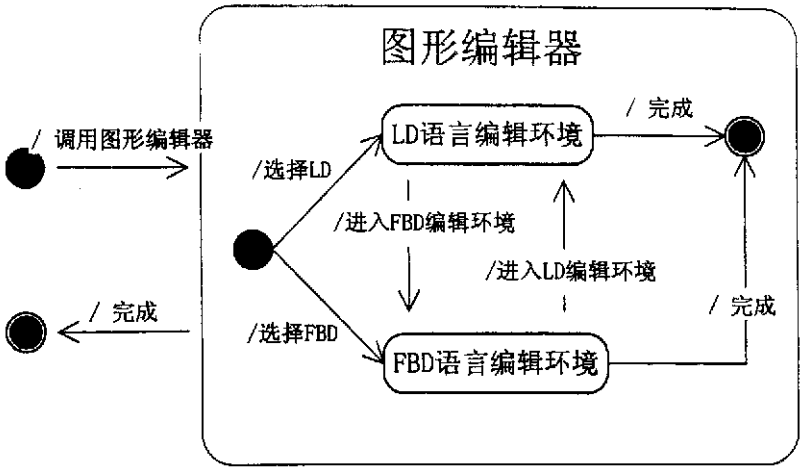

图 4-10：图形编辑器状态图

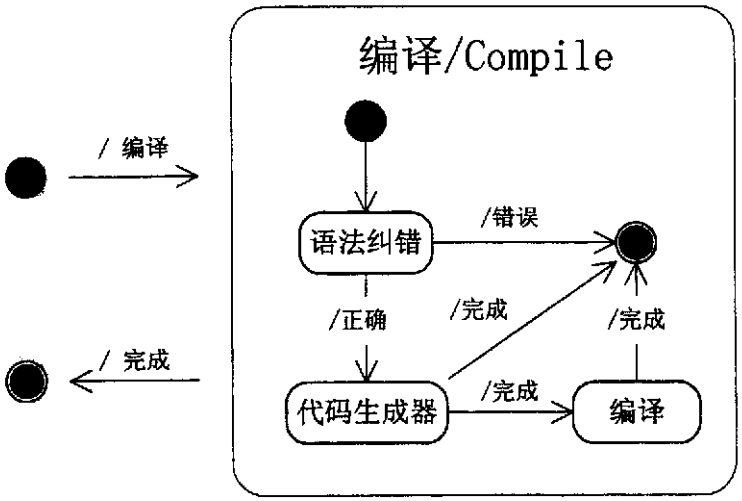

图 4-11：程序编译器状态图

图 4-11 是编译功能对象的状态图，当用户在程序编辑器中完成程序设计之后，通过语法、语义的纠错功能对程序进行纠错。对于程序中出现的错误，系统会给出相应的提示，并指导用户对程序进行修改；通过再次纠错且确认程序无误后，方可驱动 C 语言代码生成器将所设计的程序转换成对应的 C 语言代码，最后利用 C 语言编译器，将程序代码生成 PLC 可识别的可执行程序代码。在编译过程的每一步，系统都会以对话框的方式向用户展示编译过程的进度，并给予适当的提示，引导用户完成整个编译过程.

当所设计的程序编译完成后，即已经生成了 PLC 可执行的程序代码。此时可以立即驱动下载功能，其状态图如图 4-12 所示，该功能被内嵌在了程序编辑器中。必须注意的是，用户在下载程序之前，必须成功地建立软件系统与 PLC 的通讯连接，包括一些相关串行通讯参数的设置（串口的选择、比特率的确定、字符串的格式等）。期间，如发生通讯连接失败、中途放弃下载或文件选择超时等情况，下

载功能将会自动终止。

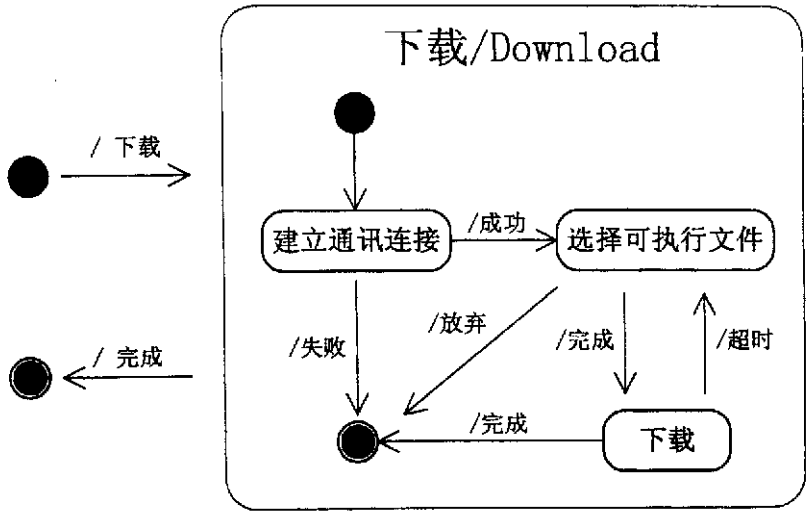

图 4-12：程序下载状态图

图 4-13 所示为在线监视器（Online Monitoring）的状态图。用户在使用在线监视功能时，必须满足两个条件：一是用户程序已经下载到 PLC 上，且 PLC 此时处于运行状态；二是要建立起软件系统与 PLC 之间的通讯连接，以保证数据的即时反馈。在满足了以上两个条件后，软件系统处于等待接收 PLC 数据的状态，当系统阶段性的接收到数据的同时，将数据在数据库中进行相应的分析与处理，并随之在图形程序中显示出来。这一系列的过程是周期性的发生，数据显示完成后又回到等待接收数据状态，然后再分析、处理，再显示，如此反复以确保数据的即时有效。

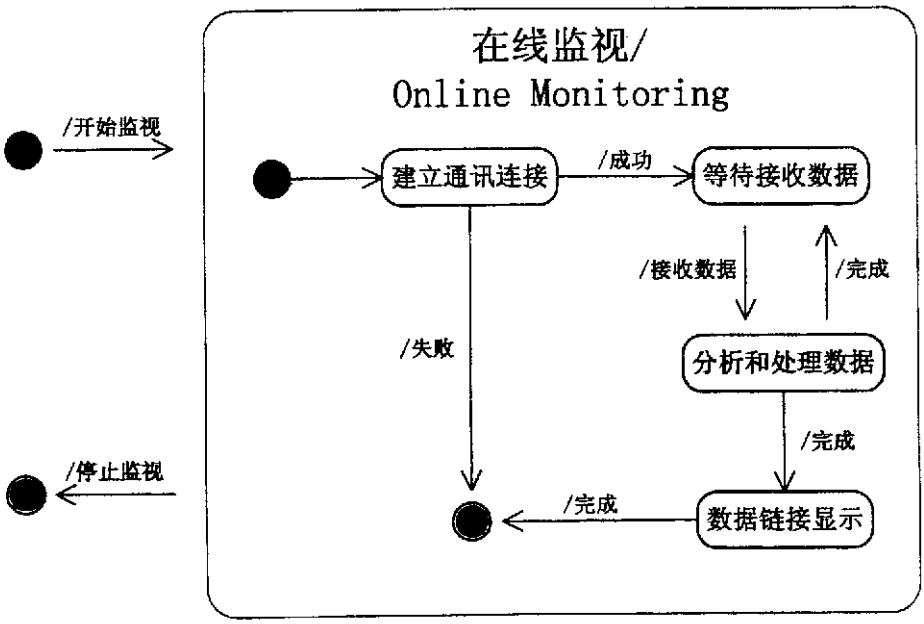

图 4-13：在线监视器状态图

4.3.3.活动图（Activity Diagrams）的建立

活动图的建立向我们展示了那些需要做的事情（即活动）以及那些必须的和不得不遵守的工作顺序。换句话说，活动图从抽象程度较高的层次上表示了并行过程的发生。这一点也是它与流程图的根本区别。下图 4-14 即为基于 IEC61131-3 标准编程环境的活动图，其中显示了程序编辑器（Programmer）、编译器（Compiler）、下载（Download）和在线监视器（Online Monitoring）的基本操作顺序。对于编辑器活动图，我们单独给出，如图 4-15 所示。从图中我们可以清楚地看出各个功能在执行时的顺序，并以此与所建立的状态图进行对照。

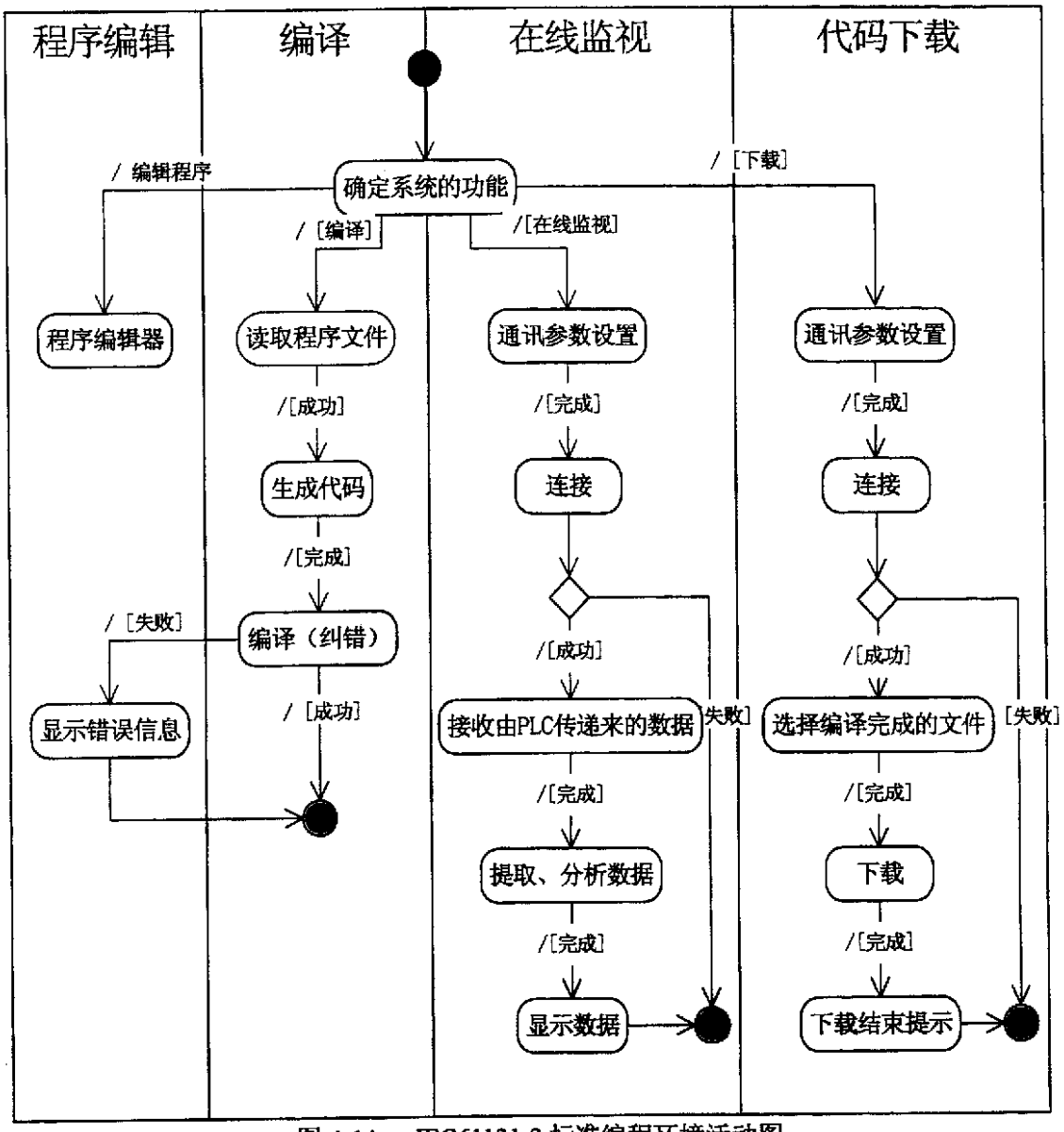

图 4-14： IEC61131-3 标准编程环境活动图

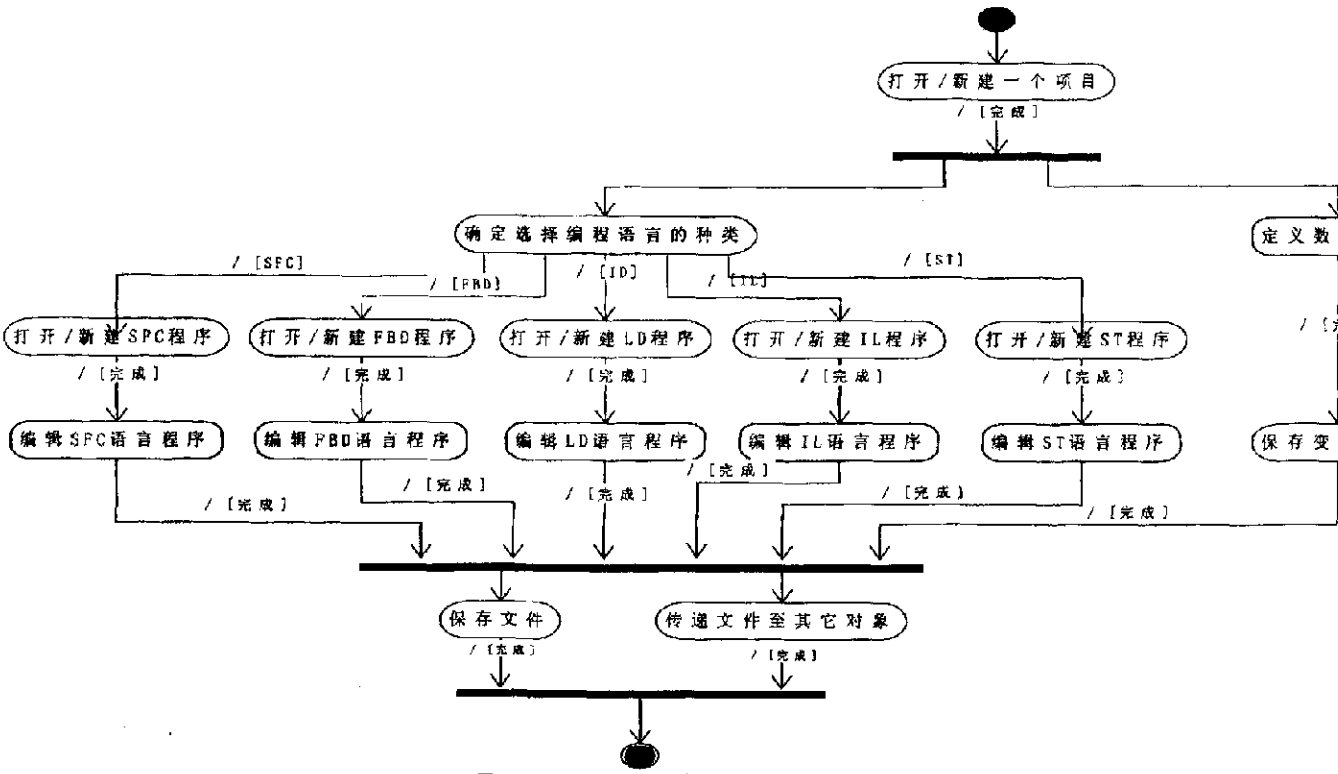

图 4-15：IEC61131-3 标准编程环境编辑器活动图

4.4. 编程环境的实现

4.4.1.系统功能展示

编程环境的系统构成如图 3-1 所示[46]，整个软件框架是建立在遵循 IEC61131-3 标准之上。从功能上软件系统主要可划分为五大部分，分别对应着五大用例对象，它们是程序编辑器对象（Program Editor）、编译器对象（Compiler）、下载（Download）、在线监视对象（Online Monitoring）以及数据库对象（Database）。

根据已经建立起的梯形图语言类别图 4-2 和功能块图语言类别图 4-3，我们在高级语言系统开发环境 C++ Builder 6.0 中对其进行了基本功能的实现。下面我们将对已经实现的 LD 语言和 FBD 语言的开发环境作简单介绍，并给出相应代码的设计思想和实现。

运行系统，首先显示的就是项目管理界面，如图 4-16 所示。在该界面中，用户可以进行的操作有建立项目、打开项目、设置项目注释文本，以及导入和导出项目等。整个项目管理器，将各个程序文件以项目的形式进行划分、归类，使用项目管理窗口中的“打开”命令或鼠标双击项目，就可以打开项目文件。

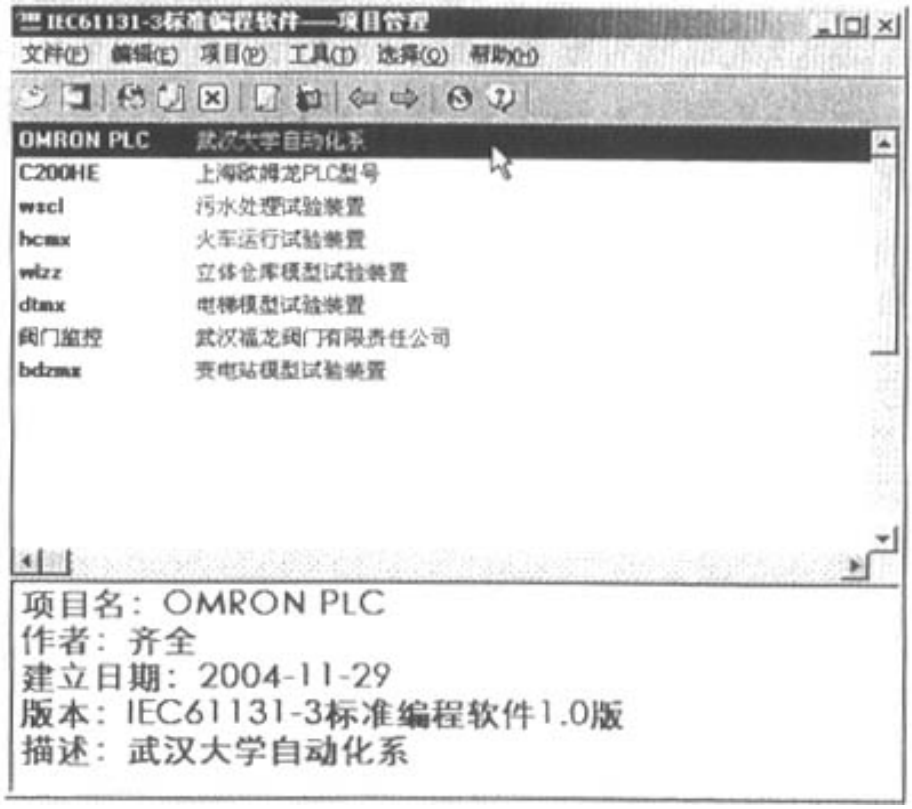

图 4-16：项目管理器界面

如图 4-17 所示为梯形图编辑环境界面[36,42,49]，当用户打开 LD 语言编辑文件后，呈现在面前的就是 LD 语言编辑区。在系统工具栏的下面，我们提供了梯形图

编程语言专用的工具箱，在工具箱中有各种图标按钮，每个按钮分别对应着梯形图的主要语言元素（接点、线圈以及功能等），用户可以通过对图标按钮的点击和托放，在“程序编辑区”中完成梯形图程序的编辑。以下我们分别对工具箱中的元素进行介绍：

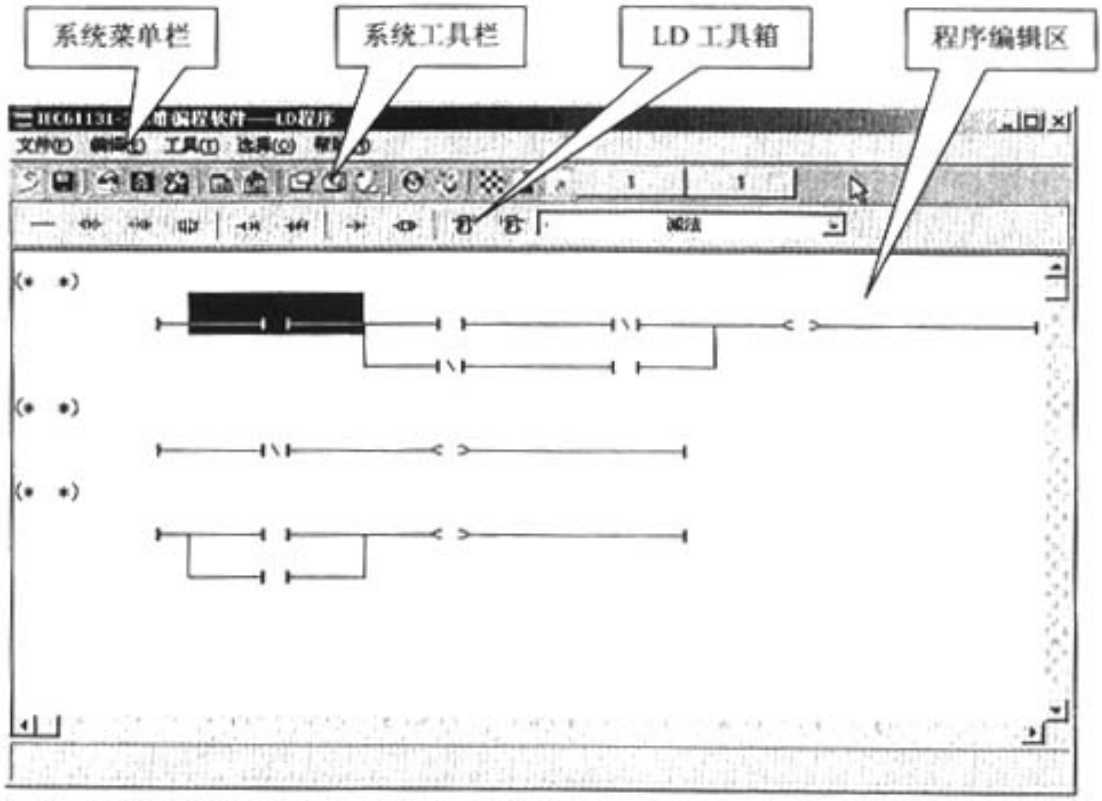

图 4-17：梯形图编辑环境界面

- ：水平连接线，连接两图元；
- ：与一个变量相关的接点；
- / ：添加接点于所选图元的左侧或右侧的接点；
- ：实现并联连接的接点；
- / / ：与一个输出变量或一个内部变量相关的线圈（常开，常闭接点，输出接点）；
- / ：添加功能函数于所选图元的左侧或右侧；
- ：有条件或无条件的跳转符号，可被用于控制梯形图的执行；
- ：“返回标号”在程序中用作一个输出，它表示该程序的一个有条件的结束。在一个返回符号的右侧不能进行连接。

FBD 语言是许多不同类型的编程方式的一种图形表示法。操作符被表示为矩形的功能块。功能的输入被连接到矩形块的左侧；输出连接到块的右侧。图形的输入和输出(变量)通过逻辑链接连接到功能块。一个功能块的输出可以连接到另一个块的输入。

图 4-18 为 FBD 语言程序编辑环境界面。其中在工具栏的下部是编辑 FBD 语言的专用工具箱，与 LD 编辑环境相类似，在工具箱上也有数个图形按钮，分别对应 FBD 图中的主要语言元素。其操作方式与 LD 相同。工具箱的上方是一般的系统工具栏和菜单栏，其中包括对程序文件的打开，删除和保存等一般文件操作。以下就工具栏上的主要图形按钮作简单的介绍。

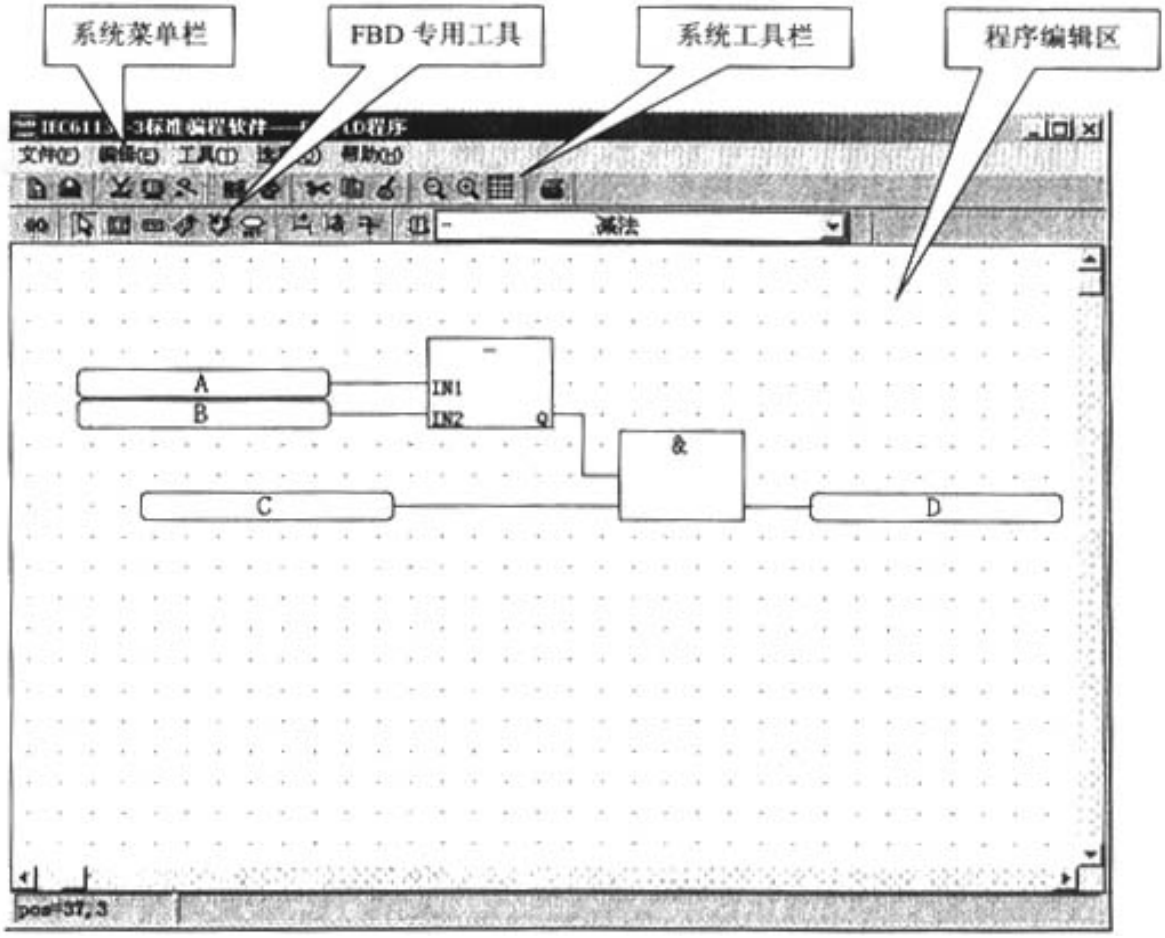

图 4-18：FBD 语言编辑环境界面

- ：选择对象。用于在图形程序中选择一个或多个图形元素。
- ：功能块。在 FBD 图中一个块可以表一个功能、功能块、子程序或运算符。
- ：标号。可以放在程序内的任何地方，用以提高程序的可读性。
- ：跳转符号。指向位于程序中某个地方的标号。

-  ：返回标号。必须与布尔变量相连接，并根据其值决定返回与否。
-  ：变量。在程序图表中变量在小的矩形框中表示，它的左侧或右侧连接到图表的其它元素。
-  ：连接线。连接 FBD 程序的各图形元素，且总是沿着数据流的方向从一个输出到一个输入。
-  ：表示逻辑非的连接。这种逻辑“非“的连接被表示为在其末端带有一个小圆圈。

以上介绍的两种语言的图形编辑器均是采用的拖拉式（Drag and Drop）的可视化编辑方式，用户只需利用鼠标点击并选中工具箱中的图元，然后将其放置在程序编辑区，就可以快速而且方便地创建出所需的图形控制程序。再利用变量连接对话框，将先前定义好的变量与程序中的图元相连接，数据的映射。这样就完成了一次程序的设计与编辑。

限于篇幅，在这里我们不对软件系统其它的功能界面一一介绍。

4.4.2. 程序代码的设计思想[43]

在运用高级语言对软件系统的实现过程中，我们充分体现了面向对象的编程思想，并灵活运用了 C++语言所提供的类、派生和继承的概念与模式，对软件对象进行编码和实现。

在对图形编辑器编码的设计中，我们运用 C++ Builder 所提供的 TCanvas 类作为图形对象操作的基类，因为它对 Windows 提供的图形设备接口（GDI）进行了很好的封装，我们只需在此基础上，派生出软件系统的编辑器类。派生后的类的对象均继承了基类的所有特性（数据成员和成员函数），并且保护性（piblic）地拥有对专用成员数据和操作函数的调用。

在对图形编辑器实例化的过程中，图形编辑器就被视为一个对象，在这个对象中包含了其他对象，即梯形图编辑器对象（LD Net）和 FBD 编辑器对象（FBD Net）。梯形图编辑器对象又由自身专用的对象成员（LDObjects）——图形元素组成，其中包括电源轨线（左和右）、线圈（常开接点、常闭接点、上升沿接点、下降沿接点）和线圈（正向线圈、方向线圈、置位线圈和复位线圈等），这样的建模结果与实际需求的结果是相同的。

以下我们以 LD 编程环境为例，给出程序的部分代码[36,49]，通过代码的结构可以比较清晰的体会到面向对象方法在软件设计中的运用：

```
class CIECDoc : public CDocument
{
  public:
```

```
    CLDNet      theLDNet ;
    CFBDNet     theFBDNet ;
    ..........
} ;

class CLDObject
{
  public:
        LDObject(int LDid ,CString,LDname,POINT LDposition );
        Virtual void Paint(CDC * );                    //虚函数－绘制图形；
        Virtual void Save(Carchive &ar);               //虚函数－保存图形文件；
        Virtual void Load(CString in);                 //虚函数－载入图形文件；
        Virtual ～CLDObject();                         //析构函数；
        ..........
  protected:
        int         LDid;
        CString     LDname;
        POINT       LDposition;
        ..........
};

class CLDContact : public CLDObject                    //梯形图－接点类
{
  public:
        CLDContact(int ConType,CString ConName,POINT c,)
        //构造函数（初始化接点类型，名称）；
        Virtual ～CLDContact();
        //析构函数；
        ..........
  protected:
        Virtual void CConvertion(CString Contype);     //接点类型转换；
        Virtual CString CallParameter(CString ConName); //调用变量参数；
        int   ConType;                                 //接点类型变量；
        ..........
};

class CLDCoil : public CLDObject                       //梯形图－线圈类
```

```
{
  public:
        CLDCoil(int ClType,CString ClName,POINT c,)
        //构造函数（初始化线圈类型，名称）；
        Virtual ~CLDCoil();                                    //析构函数；
        ……
  protected:
        Virtual void CConvertion(CString Cltype);              //线圈类型转换
        Virtual CString CallParameter(CString ClName);         //调用变量参数；
        int   ClType;                                          //线圈类型变量；
        ……
};
```

5. 结论与展望

5.1. 结论

在由国际电工委员会（IEC）制定的关于可编程控制器的国际标准（IEC61131）中，IEC61131-3 作为它的第三部分——编程语言部分，定义了可编程控制器通用的一套编程方式。它在工业控制系统软件应用方面的一致性不仅降低了程序开发的成本，还极大地改进了编程软件的质量，提高了软件的开发效率。

在本次系统的设计与开发过程中，作者首先深入研究了 IEC61131-3 标准，并在此基础上，总结得出了该标准对工业控制编程系统的各项要求，然后通过对大量国内外文献和资料的查阅和研究，及时掌握了目前相关产品的现状和发展趋势，并通过进一步相关产品的调研，最终确定了此论文研究的方向。

在软件的开发方法上，我们采用了目前比较流行且较为先进的面向对象的迭代增量法；在系统模型的建立上，我们灵活运用了集 OMT、Booch 以及 Jacobson 方法于一体的统一建模语言——UML。在代码的实现上，我们利用高级语言编程工具 C++ Builder6.0 作为系统开发工具，且充分体现面向对象的编程思想，逐一对模型系统中的主要功能模块予以实现。

需要指出的是，IEC61131-3 标准对控制领域的影响并不仅限于 PLC，它还应用于 DCS、PC、现场总线控制，运动控制，甚至 SCADA 系统。它的这一特点在目前的工业控制领域已经得到了证实。

5.2. 展望

本次的设计与开发虽然取得了一定的成果，但是由于时间的限制和 IEC61131-3 标准本身所涉及到的内容的复杂性，以及软件迭代开发的周期性的要求，我们不可能在短期内实现 IEC61131-3 标准所要求的全部功能，只是实现了软件系统的一部分功能。在论文的研究初期，我们已经预见到了此次所要开发的软件系统功能的庞大，和课题的难度，清楚地意识到要最终实现它是一个漫长的过程，需要我们作出坚持不懈的努力。

就目前国内这方面的研究情况来看，也展开了部分基础技术的研究工作，但是由于起步较晚，至今尚未有成熟的产品出现。接下去所要进行的工作可以从以下几方面着手：

1. 完善 IEC61131-3 所支持的其他编程语言环境。IEC61131-3 标准共支持 LD、FBD、SFC、IL 和 ST 五种语言的编程方式，本次论文研究的范围仅限于 LD 和 FBD 两种，且只是实现了基本图形的编辑，对于另外三种语言编辑方式有

待进一步研究。

2. 尽可能实现软件系统的多线程运行，不同语言的交叉编译以及它们之间的相互调用。因为这些特点在 IEC61131-3 标准中作出了明确的要求。
3. IEC61131-3 标准在自动化领域的应用并不仅限于 PLC 控制系统，还可以应用于 DCS、PC、现场总线控制，运动控制，甚至 SCADA 系统。所以，接下来我们要尝试着将 IEC61131-3 标准应用到他控制领域中，即实现软件系统与不同的硬件设备的连接。

致 谢

三年的求学路，短暂而漫长。一路奔跑，不敢懈怠，才有了今天的这篇论文，这是值得深刻回味的。其间倾注了很多人的希望和心血，在此真心的对他们表示感谢。

尤其是我的导师，从论文的选题，到课题的研究，再到最后的撰写、修改和定稿无不凝聚着导师辛勤的汗水。三年来，导师不仅为我提供了良好的学习环境，而且在学业上、生活上都给予了我极大的关心、支持和帮助。他渊博的知识、严谨的治学态度、积极进取的精神和谦逊务实的工作作风都时时刻刻影响着我。在学习中，导师悉心指导，循循善诱，使我的学术水平和实际动手能力都有了很大的提高；在工作中，导师不辞辛劳，以身作则，为我作出了极好的榜样。从他身上，我们学到了许多书本上没有的东西，这些都将使我终身受益。

再一次表示我对导师崇高的敬意和深深的感激之情！

除此之外，在我研究生和本科学习过程中，得到了自动化系各位老师的教育和指导，在此一并表示衷心感谢！

同时，我要向给予我最无私的爱的家人表示深深的谢意，是他们的关心和支持给了我无穷的动力！

最后，向曾经给予我关心和帮助的所有老师、同学和朋友表示衷心的感谢，并致以崇高的敬意和美好的祝福！

万 涛

2005 年 4 月 28 日

参考文献

[1] 何衍庆，戴自祥等.可编程控制器原理及其应用技巧[M].化学工业出版社，2003.5 第 2 版(8).

[2] 章文浩.可编程控制器原理及其应用[M].北京.国防工业出版社，2003

[3] Jiang—Hai Huang，Yu—Cheng Li，etc.The design of a new-type PLC based on IEC61131-3[J].Machine Learning and Cybernetics，2003 International Conference on Volume 2，2-5 Nov. 2003 Page(s):809 - 813 Vol.2.

[4] 周峰，王新华等.软 PLC 技术的发展现状及应用前景[J].计算机工程与应用，2004 第 24 期.

[5] 彭瑜.工控编程语言 IEC61131-3 的现状和发展[J].国内外机电一体化技术，2004 年第一期.

[6] 邢建春，王双庆 等.IEC 61131-3—工业自动化系统的控制逻辑组态软件标准.世界仪表与自动化，2003 第 7 卷第 8 期.

[7] Karl-Heinz John.Michael Tiegelkamp.IEC61131-3：工业自动化系统的程序编制[M].中国机电一体化技术应用协会 秘书处翻译.

[8] R. Lewis.Programming Industrial Control Systems Using IEC61131-3.IEE[M]，Michael Faraday House，1998.

[9] PC 编程语言标准 IEC1131—3 的优点[J].机床电器，1999 第一期.

[10] 丁军.浅谈 IEC61131-3 程序组织单元 POU 及其应用[J].自动化博览，2002.10.15.

[11] Karl-Heinz John.Michael Tiegelkamp.IEC61131-3:Programming Industrial Automation Systems[J].Springer，2001

[12] JAMESH.CHRISTENSEN，Ph.D.International Standards for Open Distributed Automation[J].Rockwell Automation，2000.6.23.

[13] 陈忠华.细说 IEC61131-3[J].(第一节)自动化博览，2003.5.

[14] 陈忠华.细说 IEC61131-3[J].(第二节)自动化博览，2003.6.

[15] 陈忠华.细说 IEC61131-3[J].(第五节)自动化博览，2004.1.

[16] 彭瑜.IEC61131-3 的现状与发展(下)[J].世界仪表与自动化，2002.6 卷 2 期.

[17] 杨磊，徐蓉萍等.IEC61499：工业控制技术的发展[J].电气传动自动

化,2002.12 第 24 卷第 6 期.

[18] 方原柏.IEC1131－3 编程标准及其影响[J].石油化工自动化，2000.3(38)

[19] 方原柏.PLC 发展的新动向——IEC1131－3 编程标准和开放式结构[J].昆明理工大学学报，1999.12 第 24 卷第 6 期(102-103).

[20] 李平康，赵书君.ISaGRAF 集成化编程技术及在 DCS 中的实现[J].微计算机信息，1999.第 6 期.

[21] 刁成嘉.面向对象技术导论－系统分析与设计[M].机械工业出版社，2004.9 第一版.

[22] Jaffe M.S.,Leveson N.G,etc.Software requirements analysis for real-time process-control systems[J].Software Engineering,IEEE Transactions on Volume 17,Issue 3,March 1991,Page(s):241-258.

[23] 汤庸.软件工程方法与管理[M].冶金工业出版社，2002.9 第一版(8-12)

[24] 韩承双.增量迭代的软件开发方法[J].合肥工业大学学报(自然科学版)，2004.6 第 27 卷第 6 期.

[25] Wiegers KE. Software requirements[J]. Redmond (Washington):Microsoft Press,1999.1-76.

[26] CockburnA.Writing effective use cases [M] Reading (Massacusetts): Addision-Wesley，2001.20.38.

[27] 刘超 张莉.可视化面向对象建模技术[M].北京航空航天大学出版社，1999.7(19)

[28] 安建伟.基于 UML 的软件系统建模研究[C]. 西南交通大学，2002.4.18(2-3)

[29] James Rumbaugh，Ivar Jacobson，Grady Booch.The Unified Modeling Language Reference Manual[M].Addison-Wesley，1999.

[30] Young Jong Yang，Soon Yong Kim， etc.A UML-based object-oriented framework development methodology[J].Software Engineering Conference，1998.

[31] Cybulski,J.L,Linden,T.Learning systems design with UML and patterns[J]. Education，2000.

[32] 高焕兵.UML 面向对象技术在分布式监控系统中的应用与研究[C].西南交通大学， 2004.4.1(19-20)

[33] 柳立峰.UML 概述及其在面向对象软件设计上的具体应用[J].江西通讯科技，

2001.9，第三期

[34] Karlsson, J. Software requirements prioritizing[J]. Requirements Engineering, 1996.

[35] 张峰.基于ERP的实验室综合管理信息系统的研究与开发[C].武汉大学自动化系，2003.

[36] 张启忠，杨纪春等.PLC梯形图编辑器软件的设计[J].机电工程，1997第6期.

[37] 刘光.C++ Builder数据库系统的设计与开发[M].清华大学出版社，2003.8第一版.

[38] 吴福川.SoftPLC控制规划软体之设计与实现[C].

[39] 邵维忠，杨芙清.面向对象的系统设计[M].清华大学出版社，2003.2第一版.

[40] Kurt Bittner，Lan Spence.用例建模[M].清华大学出版社，2003.8 第一版(1-3)

[41] Vidgen, R. Requirements analysis and UML use cases and class diagrams[J]. Computing & Control Engineering Journal, Volume 14, Issue 2, April/May, 2003, Page(s):12-17.

[42] 陈战林，张万里等.C++ Builder组件大全[M].电子工业出版社，2002.1第一版.

[43] 吕凤翥.C++ 语言基础教程[M].清华大学出版社，1999.3第一版.

[44] "ISaGRAF Automation".http://www.isagraf.com

[45] 骆智.可编程控制器(PLC)运行系统设计与实现[C].北方工业大学，2004.6

[46] 黄江海，李宇成.基于IEC61131-3的新型的PLC系统的设计[J].北方工业大学学报，2004.3

[47] 韩瀛.UML及基于UML的统一开发过程应用研究[C].天津财经学院.2002.5(24-25)

[48] 斯可克.可编程控制器的标准化编程——简介IEC1131-3标准[J].微计算机信息，1997.第2期.

[49] 蒲志新，熊永超等.PLC梯形图语言编辑功能的软件实现[J].机械，2003第30卷第3期.

[50] 黄延延，林跃等.软PLC技术研究及实现[J].计算机工程，2004.第一期

附录 1　攻读学位期间发表论文目录

- 汪小澄, 万涛, 张海艳. 基于 89C2051 的阀门监控系统. 机械与电子, 2004. 7 总第 139 期;