摘　要

IEC61131 标准作为工业自动化控制系统编程语言的国际标准，其开放性和先进性受到了广泛的关注和重视。研究基于 IEC61131 标准的控制系统意义重大。

本文以 IEC61131-3 标准所定义的编程模型以及 IEC61131-5 所定义的通信模型为基础，实现了以功能块执行为核心的下位机程序设计以及以上位机功能块组态程序的设计。

下位机以 Rabbit2000 处理器为核心的现场 I/O 控制器为基础，实现了基于 IEC61131 标准的可编程控制器的软件开发，包括功能块的执行、变量通信、系统管理、用户程序的上载下载、I/O 扩展等内容。

上位机基于 Visual C++ 6.0 开发了 PLC_Config 功能块组态程序。采用模块化设计将整个软件分解成现场设备管理模块、功能块编辑编译模块、现场设备监控模块、文件管理模块、通信服务模块及信息报告模块。可对现场所有控制器资源进行配置和规划，使现场网络成为一个有机整体，协同工作完成控制目的。

以水泥混合配料自动控制工程为背景，基于本文设计的可编程控制器实现了控制系统的设计、工作流程、性能参数配置以及程序实现，达到了预期的控制效果，现场运行良好。通过本文的论述和工程应用，证明了基于 IEC61131 标准的嵌入式可编程控制器的开放性和易用性，在节省人力资源、控制系统维护以及软件可重用性等方面具有明显的优势。

关键词：IEC61131；可编程控制器；组态；功能块

Abstract

As a standard programming language of automation control system,IEC61131 standard is widely used in the world. Many Companies have developed their control systems based on this standard,therefore It's very important to develop the control systemof that style.

Based on the content about Function Block configuration in IEC61131-3 and communication module in IEC61131-5,Develop the basic programming concept and instruction rule. The Implementation of PLC base on IEC61131 is the main task in this project,Including the definition of programming module and communication module.The programming module includes software programming and variable communication.The Communication module supplies a way for PLC communication and I/O extension.

The Program in the controller is developed based on Rabbit2000 CPU. The management program include I/O communication,user code download and upload,password protection. User program include execution of the program and so on.

Develop a Configuration software named PLC_Config. This software is developed on Windows operating system and VC++. It is a visible and graphic configuration software. All the software and hardware design of the devices can be configed using PLC_Config. As a moudled software,It's divided into many parts like Device management,Visible configuration,Device monitor,File management and Communication Management.

Based on a project of cement mixing control system, design a control system using this control module. Develop the system design, chart flow, parameter setting and reach to the system requirement. By the research and application in the field, The controller is verified in the theory and field application. It has many advantages in system opening, human resource saving,software reusing.

Key Words: IEC61131; PLC; Configuration; Function Block

独创性说明

作者郑重声明：本硕士学位论文是我个人在导师指导下进行的研究工作及取得研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得大连理工大学或者其他单位的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的贡献均已在论文中做了明确的说明并表示了谢意。

作者签名：郭福坤 日期：2006年6月28日

大连理工大学学位论文版权使用授权书

本学位论文作者及指导教师完全了解“大连理工大学硕士、博士学位论文版权使用规定”，同意大连理工大学保留并向国家有关部门或机构送交学位论文的复印件和电子版，允许论文被查阅和借阅。本人授权大连理工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，也可采用影印、缩印或扫描等复制手段保存和汇编学位论文。

作者签名：郭福川

导师签名：仲崇权

2006 年 6 月 28 日

1 绪论

1.1 现代自动控制系统和 IEC 61131-3 标准

在工业控制领域，PLC 技术的发展在上世纪 90 年代经历了一次高潮。这得益于微电子、网络通信和控制技术的迅猛发展。但随着技术的进步和市场要求的提高，传统的 PLC 越来越暴露出其在数据封装能力，程序可重用性，顺序操作编程等方面的不足。这些缺陷导致了不同制造商 PLC 平台的不兼容，也给工程技术人员的学习和操作带来了极大的不便。统一的编程规则成为技术人员的迫切需求。IEC61131 标准正是在这种情况下应运而生。IEC61131-3 是 IEC 61131 国际标准的第三部分，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。该标准提供给用户一种良好结构、自上而下或自下而上的程序开发方法，提供全套的配置集成，允许程序分解成功能块和软件元素，进行完全的程序控制。程序不同部分在不同时间，以不同周期或平行的运行。提供了一套统一的应用于 PLC 的语法和语义[1]。IEC61131-3 包括 5 种编程语言，即指令表、结构化文本、梯型图、功能块图和顺序功能图。规范了编程语言、PLC 与编程系统的接口、字符集和工程管理，使得所有 PLC 使用相同的概念，平台程序可以互相移植，从而整体降低自动化控制系统的费用。这些都是在工业控制系统所阐述的软件设计的概念和软件模型等的基础上制定的，适应了当今世界软件、工业控制系统的发展方向[2]。

符合 IEC 61131-3 的软件系统是一个结构完善、可重复使用、可维护的工业控制系统软件。标准最初主要用于可编程序控制器 PLC 的编程系统，但随着可编程序控制器 PLC 技术、编程语言等的不断进步也在不断地进行着补充和完善。目前该标准同样也适用于过程控制领域、分散型控制系统、基于控制系统的软逻辑、SCADA 等。IEC 61131-3 国际标准在技术上的实现是高水平的，因此有足够的发展空间和变动余地，这也使得该标准能很好地适应工业控制的发展和要求。因此 IEC 61131-3 标准在 1993 颁布之后被国际用户和开发商团体广泛接受，目前得到全世界的认可。世界顶尖的工业控制商接受了这个编程模型，各种软件公司提供相应的开发工具。IEC 61131-3 国际标准已对整个控制领域形成了巨大的冲击，采用或应用符合 IEC 61131-3 国际标准的组态产品，已经成为国际工业控制领域的一大趋势[3-4]。

1.2 PLC 控制系统与 DCS 控制系统

目前的工业自动化控制系统绝大多数是 PLC 或 DCS 系统。基于计算机的控制始于 1960 年，PLC 也同时产生。PLC 的市场最初在基于继电器的离散制造业控制中，其应

用具有目的倾向，专门由电气人员使用。由于PLC的高可靠性及使用的方便性，其占领着很大的市场份额。在市场的需求及DCS的挑战下，PLC也在不停地发展，主要表现在[5]：

(1)分布式I/O连接能力方面，许多PLC支持远程I/O及现场总线网络，符合现场总线标准IEC 61158（如Pro-fibus等），使PLC构建分布式应用成为可能；

(2)信息管理通道方面，许多PLC支持高速网络，如Ethernet，ControlNet，Profibus等；

(3)软件的开放性方面，采用可编程逻辑程序语言IEC 61131-3，支持OPC标准；

这些技术的进步推动PLC市场的继续扩大，PLC厂商更是竭尽全力提升产品的附加价值。目前PLC的发展走两个方向，一个是小型系列化，结构功能和使用均较为简单，用于一般工业场合，用户使用起来很容易上手，且控制功能容易实现；另一个是复杂化，功能强大，结构复杂，可靠性高，多有失效安全保障措施，应用于高端场合。但两种方向的发展也有统一的一面，即控制用软件的标准化IEC 61131-3的应用。

控制算法组态是传统PLC的一大缺陷，但通过IEC 61131-3提供的梯形图逻辑及丰富的功能块，在PLC上组态复杂算法已经比较简单。控制功能的不断增强，使得PLC有能力向传统DCS占领的过程控制领域进军。传统DCS具有相对封闭的控制算法组态技术，虽然包括梯形图、助记符、功能块等，但多数不符合IEC 61131-3。而IEC 61131-3编程标准，不仅适合于PLC系统，也同样适于DCS。工程技术人员不希望因控制对象的不同而改变已熟悉的控制设备及编程语言，这又是PLC系统的一个优势。但PLC本身也有局限性，PLC是一种通用控制器，应用到某些场合如电力系统中时，需做许多硬件和软件上的改进工作，并配置额外的设备和电路。PLC控制系统的数据库组态很不方便，它所能提供的只是寄存器，需要借助第三方或专用软件将其转换成有意义的工程参数，对于DCS而言数据测点定义相对完备[6]。

DCS产生于上世纪70年代中期，主要市场在过程控制；但PLC系统在中小型应用中由于低成本、配置方便、维护简单等因素对DCS构成威胁。强大灵活的控制功能是DCS的绝对优势，但随着技术的进步PLC正迎头赶上。DCS为了守住其市场，也必须进行技术革新[7]。实际上DCS的确也在不断地发展。在分布式现场I/O连线上采用符合IEC 61158规范的现场总线技术，在管理层及信息层采用TCP/IP、工业以太网等事实标准，软件普遍支持OPC及COM、DCOM、DNA等技术。这些工业控制领域的新技术在DCS上获得应用，体现了现代控制系统的开放性要求，但其逻辑组态软件仍有许多工作要做。IEC 61131-3对于DCS系统仍然适合，尤其是其中的功能块编程语言。目前尽管各个DCS厂商均推崇其DCS的开放性，但一般而言，不同的DCS需要不同的编程软件环境和编程方法。这一点无法与PLC相比，PLC的开放性不仅仅在于硬件系统的可互换，还表现在统一的编程语言上，即符合IEC 61131-3的软件开发环境。

1.3 国内外发展现状

德国 SIEMENS 的 PCS7 是新一代 DCS 控制系统，面向所有过程控制应用场合，能体现系统全集成自动化思想，冗余的操作站同时运行，能保证控制过程的可靠性。它使用 STEP 7 编程语言，完全符合 IEC 61131-3 标准，技术人员可以为不同的用户需求和应用场合灵活选择 PLC 解决方案。STEP 7 是 SIEMENS 通用软件开发环境，可直接用于组态、管理和维护各种 SIEMENS 产品构成的自动控制系统。我国很多 DCS 厂商也采用北京亚控科技符合 IEC 61131-3 标准的 KingACT 作为其控制逻辑的编程环境，已在多个工程实际中应用。和利时已开始采用 Infoteam 公司 OpenPCS 技术开发新一代的 DCS。从 80 年代中期，基于 PC 的控制开始出现，并在中小系统中迅速地抢占市场份额。到今天它仍是自动化控制系统发展的一个亮点。基于 PC 的控制是自动化控制应用的一个趋势。它一般包括开放式结构的 PC 平台如工业 PC 机，各种 PCI 总线数采卡，强实时操作系统如 NT、QNX、DOS、Windows CE、VxWorks 和统一的控制逻辑编程标准 IEC 61131-3[8]。

基于 PC 的控制相对于 PLC 和 DCS 来说有许多好处。开放的结构，用户可以选择来自不同厂商的不同产品，为应用提供更大的系统柔性；PC 工业的软硬件变化很快，用户可以得到更高性价比的合适产品。提供有力、柔性的连网能力，可以使用标准 TCP/IP 以太网和网卡。能运行复杂任务（如趋势分析）。一些 PLC 厂商也认识到这种基于 PC 控制的优势，并着力开发这方面的产品和市场，如 SIEMENS 公司，在继续发展其 SIEMENS 系列 PLC 的同时，推出软逻辑软件 WinAC 并取得很好的市场业绩。

基于 PC 的控制有两个核心内容，即符合 IEC 61131-3 的编程软件和运行控制逻辑的实时内核。不同产品的实时内核实现机理不同，可比性不强，因此符合 IEC 61131-3 成了各厂商占领市场的一个重要砝码[9]。德国 Beckhoff 公司的 TWinCAT 可编程控制器是一个运行于 PC 机上的功能强大的多 PLC 系统，有 4 个 PLC 运行系统，每个可执行 4 个任务。除 IEC 61131-3 的 5 种标准编程语言外，还提供连接 C 代码的接口。Altersys 公司的 ISaGRAF 软件提供硬件独立的编程环境，完全支持 IEC 61131-3 标准中的 5 种编程语言，并提供对 IEC 61131-5 通信功能块的支持，增加了通过域总线的地址通信标准。用户可以使用梯型图和功能块图在同一个图表中混合编程；同时集成标准 C 语言编辑器，可使用 C 语言编写功能块和函数，并在多任务系统中与 ISaGRAF 程序并行运行。ISaGRAF 可以把任何 PC 或 Soft PLC 集成到高性能低造价的软逻辑开发和控制环境，并进行编程、仿真、调试和在线监视。产生的代码可以被任何硬件平台所用，而不需要刻意调整，不同的硬件可以共存于一个系统，目标硬件的选择仅仅依赖于用户应用的功能性能需求。它的实时核心（ISaGRAF PRO runtimes）可用于 Windows NT、VxWorks 和 OS-9 等操作系统。Altersys 还提供了对于其他操作系统和硬件平台的特定接口的软件开发包。

1.4 IEC61131-3 及其功能块编程

可编程序控制器（PLC）的早期阶段，由于没有一个统一的国际标准，各制造商根据自己的习惯，使用自己的编程语言。这些编程语言从内容到形式都很不相同，给用户带来极大的不方便，使用不同公司产品编制的程序完全不通用，用户被迫要去熟悉不同公司的编程语言，要额外的购置不同的编程工具，要想在一个大型的工程项目中使用多家公司的产品，几乎是不可能的事。在上世纪 80 年代国际电工技术委员会 IEC 的第六工作组（IEC/TC65B/WG6）就开始着手制定统一的可编程序控制器标准，并于 1993 年正式颁布了这一标准，即 IEC 1131-3 国际标准。近几年由于自动化系统的发展，迫切需要制定涵盖更广领域（如 PLC、DCS、HMI 以及现场总线等）的标准。于是 IEC 的第七工作组（IEC/SC65B/WG7）制定了新的 IEC 61131-3 标准，第七工作组包括来自不同的 PLC 制造商、软件公司和用户代表，这样制定的标准可以做为一个导则，为大多数 PLC 制造商所接受，IEC 61131 标准的五个部分总结了当今 PLC 系统的要求，这些要求涉及 PLC 的硬件和编程系统。新标准包括了早已在 PLC 编程中使用的通用概念，同时也增加了新的编程方法。其本身只做为 PLC 的编程指导，而不是强制的规则。IEC 61131 标准提供给用户一种良好结构、自上而下或自下而上的程序开发方法，提供全套的配置集成，允许程序分解成功能块和软件元素，进行完全的程序控制，程序不同部分在不同时间，以不同周期或平行的运行。IEC 61131 将特定应用的控制系统称为配置，包括硬件的分配、过程资源划分、输入输出通道分配、内存地址分配及系统的性能分析。一个配置中可定义一个或多个资源，资源可以理解为可执行的过程处理设备，像一个 CPU。一个资源中可以定义一个或多个任务。由任务控制一套程序或/和功能块的执行，可以周期或由事件驱动。程序可以使用 5 种语言的任何一种。

IEC 61131-3 的 5 种语言中，FBD 最有生命力和发展前途。FB（功能块）是控制系统的基本构件，是一个包装好的控制程序，可以是任何 IEC 61131-3 语言编写的控制逻辑和策略包装成的软件元素，可以在相同程序的不同部分或/和分散的其他程序中使用。功能块能够封装数据和逻辑，超过了 FORTRAN 和 C 语言所写的子程序，有面向对象的含义，其组成及对控制编程软件的贡献很像是现代电子电路中的集成芯片。功能块的使用提高了系统可靠性。数据封装避免了许多错误源，用户不必关心具体实现细节，只需关心与外部的接口和如何使用。开发人员只需注重于实现，而不必关心使用。功能块允许来自不同程序、项目、位置、公司甚至国家的不同组件的结合。IEC 61131-3 标准保证了功能块定义接口的使用，即定义的输入和输出参数。由不同程序员设计的功能块可借助输入和输出参数进行交互，当然输入和输出参数必须是标准中定义的数据类型。FB 不仅利于结构化程序设计，长远地看还能加速应用开发，尤其对相近的应用开发有效。现代控制系统的一个目标是代码重用，相同的控制逻辑无论硬件是 PLC、DCS 或是 PC 均有相同的程序源代码，这个目标只有通过 FB 实现。功能块的支持使得远程控制成为

可能。符合 IEC 61131-3 标准的 DCS 系统编程软件，必不可少地会使用 FB。DCS 中所有控制单元的控制逻辑一般都以 FB 的形式提供在编程环境中。DCS 还需要提供一个现场总线通信系统中用于分散处理的 FB。开放式现场总线控制系统 FCS 通过组态软件生成的参数及算法，不仅可以在控制器中运行，还可以在远程 I/O 或智能设备上运行，这就需要定义好的 FB，可以在智能仪表及执行机构中进行运算，实现真正的分布式控制。

1.5 课题的主要内容和本文结构

本课题的内容主要是在深入了解 IEC61131-3 标准以及西门子 S7 系列 PLC 的功能块编程后提出设计方案，在现有的硬件平台上实现可编程控制器 DUT7000 的软硬件编程以及调试。考虑到存储空间、执行速度以及指令执行等问题，将多种不同的功能块指令抽象化，形成统一的指令格式。在现有基本控制器的硬件基础上，通过串行总线将系统的 I/O 数量进行了扩展。并把系统分为上位机功能块组态程序以及下位机执行程序两部分分别加以介绍。本文主要分为以下几个部分：

第一章是绪论，首先介绍了课题的来源背景，提出了现代自动化系统以及 IEC61131 标准的发展以及功能块编程相对于其他编程方式的特点和优势。对比了国外、国内的研究现状。最后说明本文的研究内容和结构。

第二章主要介绍 IEC61131 标准的相关内容。介绍了 IEC61131-3 软件模型，包括组态内部的资源、程序组织单元、变量通信模型。同时也介绍了 IEC61131-5 通信模型。

第三章主要说明以 Rabbit2000 为核心的下位机系统的设计。介绍了 PLC 的发展过程和 PLC 扫描技术以及梯形图的语言特点。重点介绍了 DUT7000 的指令集定义、用户程序的执行、I/O 模块的扩展、通信以及用于程序上载/下载以及调试等内部管理程序。

第四章是介绍了上位机功能块组态与及管理程序的实现，介绍了上位机程序的层次结构以及各功能实现的思想。

第五章主要以现场应用来说明本文所设计的 DUT7000 可编程控制器在控制系统中的应用。介绍了控制系统的组成，I/O 扩展以及控制系统的程序设计流程，最终实现设计目标并稳定运行。

1.6 小结

本章主要介绍了本课题的来源背景，提出了现代自动化系统以及 IEC61131 标准的发展以及功能块编程相对于其他编程方式的特点和优势。简要说明了本论文的主要研究内容和结构框架。

2 IEC61131 标准及编程规范

2.1 IEC61131 标准概述

可编程序控制器（PLC）的早期阶段，由于没有一个统一的国际标准，各制造商根据自己的习惯，使用自己的编程语言。这些编程语言从内容到形式都很不相同，给用户带来极大的不方便，使用不同公司产品编制的程序完全不通用，用户被迫要去熟悉不同公司的编程语言，要额外的购置不同的编程工具，要想在一个大型的工程项目中使用多家公司的产品，几乎是不可能的事。在上世纪 80 年代国际电工技术委员会 IEC 的第六工作组（IEC/TC65B/WG6）就开始着手制定统一的可编程序控制器标准，并于 1993 年正式颁布了这一标准，即 IEC 1131-3 国际标准。近几年由于自动化系统的发展，迫切需要制定涵盖更广领域（如 PLC、DCS、HMI 以及现场总线等）的标准。于是 IEC 的第七工作组（IEC/SC65B/WG7）制定了新的 IEC 61131-3 标准，第七工作组包括来自不同的 PLC 制造商、软件公司和用户代表，这样制定的标准可以做为一个导则，为大多数 PLC 制造商所接受，IEC 61131 标准的五个部分总结了当今 PLC 系统的要求，这些要求涉及 PLC 的硬件和编程系统。新标准包括了早已在 PLC 编程中使用的通用概念，同时也增加了新的编程方法。其本身只做为 PLC 的编程指导，而不是强制的规则。

IEC 61131-3 是统一PLC 编程的基础，它帮助用户再一次应用已有测试和标准化软件部件，适用于生成这些部件的软件工程方法，以一个复杂的观点考虑问题的解决，以较小的模块结构成一个复杂的任务，明确定义接口，更容易地将程序转移到其他系统[10]。

2.2 IEC61131-3 标准软件模型[11]

IEC 61131-3 的软件模型描述了诸多概念，包括组态（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。为了定义这些术语，标准是基于功能最强的PLC，提供以下性质：能应用多处理器，多任务是可能的，模拟量输入，输出和数字量输入，输出是无限的并能与其他PLC 和PC 进行通信。

2.2.1 组态内部的资源

在软件等级中的最高等级是组态，它定义了单元结构，图2.1 所示为一个组态包括一个或若干个资源，它构成一个CPU，资源的程序是由任务来控制，任务表示一个可执行的程序单元。

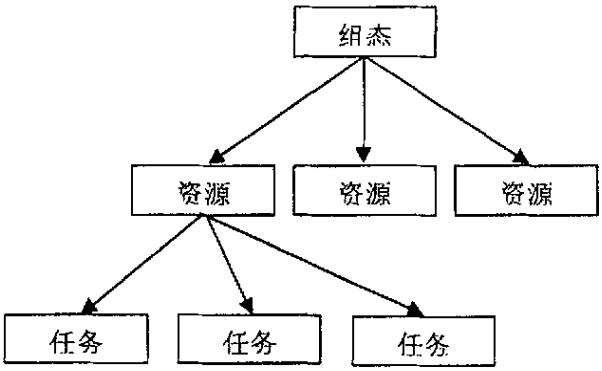

图 2.1 组态的结构

Fig. 2.1 Structure of configuration

任务能周期地或由于一定的事件来处理，它们具有优先权等级，优先权是定义在资源内部分配给CPU 的时间段。有若干种类型的任务如周期任务，时间控制任务，事件控制任务，中断任务。任务说明是由任务名、它的优先权级以及任务执行时的条件组成，条件可以是时间间隔、一个事件或一个中断。每一个任务能分配若干个程序，这些程序将由任务来激活。程序是按照所指示的顺序来处理的。带有条件的任务，在条件满足时将被执行，例如，当指示的时间间隔已经超出，或变量的地址“伪”改变成“真”；如果若干个任务都满足条件，则具有最高优先权级的任务将被执行；不允许将同一个优先权级分配给多个任务（优先权级0＝任务禁止是例外）；在另一个任务正在被处理时，如果具有较高优先权级的任务的条件被满足，则较低优先权级的任务将被中断，只有另一任务已被完成后，再继续处理。

2.2.2 程序组织单元 POUs

IEC 61131-3 定义程序、功能块、功能作为程序的组织单元。POUs 的性质允许用户程序广泛的模块化以及重复应用已经实现和经过测试的软件模块。为了程序模块能访问一个POU，至少需要有请求接口的说明，在进行说明之后，一个POU 对所有其他POU是存在的。

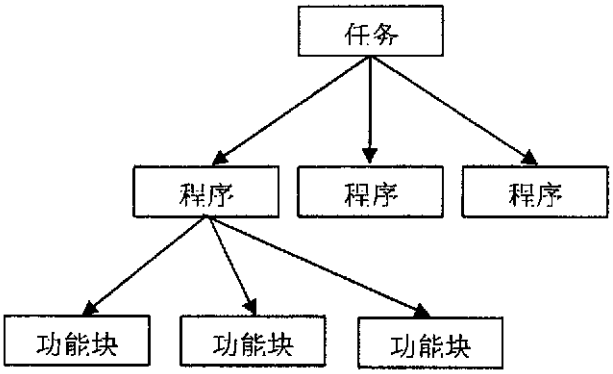

图 2.2 任务的结构
Fig. 2.2 Structure of task

整个程序的实时性质，程序能在CPU 中运行，是由分配给任务的程序来解决的，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，如图2.2。程序中的一个是主程序被分配给PLC 外部设备、全局变量和访问路径。

IEC 61131-3 应用标准功能和功能块来标准化典型PLC的功能。这一标准库是统一的，是不依赖于制造商的PLC 系统编程的重要基础。功能块可以比作集成电路，它包括一定的控制功能，它们用来设置输入/输出和内部变量，功能块的状态要求被保留从一个周期到另一个周期，只有功能的输入和输出变量能被请求的程序寻址。一个功能块能被另一个功能块调用。

IEC 61131-3 提供功能块背景，一个背景是一种结构，在调用功能块时，它保留所有的内部输入和输出变量。一个程序它调用FB1 三次，则具有三个FB1 背景，每次调用一个。程序则会精确地计算请求而不会有边外效应（side effects）。请遵守，所有背景应用相同的程序码，亦即程序码的改变对所有三个请求具有相同的效应。软件工具，通过自动说明对背景提供帮助，在FB 调用时指定背景名，这一名词管理调用的数据结构。

与功能块不同，功能没有内部变量的缓冲区。这样，功能不能使用全局变量访问功能的组织单元和直接说明地址变量。所有功能具有一个共同点，如果功能的输入参数是相同的，则它们将提供相同的输出参数。

2.2.3 IEC61131-3 标准变量通信模型

通信模型从理论上描述了不同程序组织单元（POU – Program Organization Units）之间如何交换信息的方法。程序组织单元包括程序，功能块和功能。IEC 61131-3 的通信方式使用访问路径（Access paths）、全局变量（Global variables）、参数调用（Call parameters）、通信组织单元（Communication organization，IEC 61131-5）来说明组态单元的数据交换，如图2.3。访问路径定义访问路径允许组态单元相互之间和PLC 系统实现通信。全局变量能容易地在程序之间实现通信，它们能在组态、资源、程序内进行说明和应用。参数调用在程序内部，数据交换是通过参数（如输入和输出变量）调用来实

现的，参数调用定义了值转移的接口。通信组织单元提供了通信服务，这些服务将在IEC 61131的第5 部分进行定义。

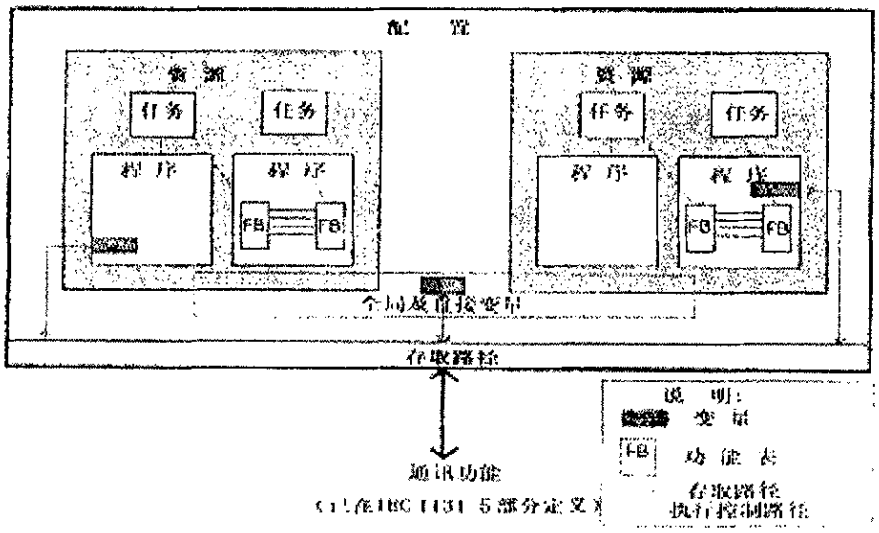

图 2.3 存取路径

Fig. 2.3 Access path

在配置、系统资源、程序、功能或功能块内，可以声明和使用局部变量、全局变量、直接变量。局部变量是仅仅能在配置、资源、程序、功能或功能块内声明和存取的变量；全局变量在一个程序（或配置）内声明，它能被程序（配置）内的所有软件元件存取；直接变量是PLC程序的内存区直接用地址变量来表示的变量。

存取路径提供了在不同的配置之间交换数据和信息的设备。每一配置内的变量可被其它远程配置存取。配置之间存取数据和信息可采用基于以太网的网络，现场总线或通过底板总线交换数据。

2.2.4 通用语言单元

IEC 61131-3 的通用语言单元是标识符（Identifiers）、关键词（Keywords）、注释（Comments）、文字（Literals）、数据类型和变量（Data types and variables）。以下对它们进行详细描述。

（1）标识符

标识符用来寻址变量、功能、程序等，它们是--些单元且能支持程序的可读性。标识符是一个字母数字和下划线的序列，以一个字母或下划线开始。以下各项不可以做为标识符：空格和数字。

（2）关键字

关键字是清楚的字母组合，能做为单个的语法助记符单元。关键字不能用做为标识符，IEC 61131-3 关键字举例：ABS、SIN、BOOL、FALSE、TRUE、FOR、NEXT、IF、

THEN、VAR、GLOBAL、DATE、TIME、FUNCTION。

（3）注释

注释或程序的一部分用来帮助理解程序且是重要的沟通方法。评论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（*and*）开始和结束。每一个网络段能对它的功能评论成一段文本。

（4）文字

IEC 61131-3 说明文字做为字母、数字和时间的序列。字母序列 字母序列文字具有0 或更多字母，而且以“反逗号”开始和结束（例如：Character sequence）数字有两种不同类型的数字文字：整数和实数。整数能定义带有基数，十进制数能具有（+或－）的符号，实数能表示成指数形式。

（5）数据类型

IEC 61131-3 定义了不同的标准数据类型，它们帮助编译、推导以及用户定义数据类型。每一个标识符被分配到一个数据类型，数据类型决定了多大的存储容量将被保留以及什么值相应于存储器的内容。其中常数可为字节，字或者双字。DUT7000 以二进制方式存取所有的数据。编程时支持十进制表示，格式有三种：十进制、二进制以及十六进制。如表 2.1 所示：

表 2.1 数据表达方式

Tab.2.1 Data expressing

数据	表示法
1000	10#1000
1.2345	10#1.2345
1	2#00000001
255	16#00FF

标准数据类型包括波尔型（真/伪），字节、字、双字、带符号的整型数、不带符号的整型数、16 位整形数、不带符号的 16 位整形数、双整形数、不带符号的双整形数（整形数据类型），实数（浮点数据类型）等。位变量采用的格式为 1 个 bit。取值为 0 或者 1。字节变量采用 8 位来表示，以字节长度存取。字变量采用 16 位表示，以字长度存取。实数（或浮点数）采用 32 位单精度数来表示，其格式是正数： +1.175495E-38 到+3.402823E+38；负数：-1.175495E-38 到-3.402823E+38 按照 ANSI/IEEE754 1985 标准格式，以双字长度来存取。如表 2.2：

表 2.2 DUT7000 数据类型

Tab.2.2 Data types of DUT7000

数据大小	无符号 十进制	无符号 十六进制	有符号 十进制	有符号 十六进制
Bit（位变量）	0	0	0	0
1 位值	到	到	到	到
	1	1	1	1
B（字节）	0	0	-128	80
8 位值	到	到	到	到
	255	FF	127	7F
W（字）	0	0	-32768	8000
16 位值	到	到	到	到
	65535	FFFF	32767	7FFF
D（双字）	0	0	-2147483648	80000000
32 位值	到	到	到	到
	4294967295	FFFFFFFF	2147483647	7FFFFFFF

（6） 变量

固定地址变量在进行说明时，通过关键字变量能被分配一个物理存储器位置。地址用特殊字母序列来指示，字母序列的起始用%符号，跟随一个范围前缀和一个数据前缀（数据类型）表示数据长度。最后是数字序列表示存储器的位置。范围前缀：I（输入）、Q（输出）、M（标志，内部存储器范围）。长度前缀：X（单个位）、B（字节，8 位）、W（字，16 位）、D（双字，32 位）。

2.2.5 IEC61131-3 功能块编程语言及规约

IEC 61131-3 国际标准的编程语言包括梯形图（LD—Ladder Diagram）、功能块图（FBD—Function Block Diagram）、顺序功能流程图（SFC—Sequential Function）、指令表（IL-Instruction List）、结构化文本（ST-Strutured Text）。

梯形图可被用来描功能，功能块和程序即程序组织单元(POU-Porgramm Orgnization Unit）的行为，以及顺序功能图（SFC—Sequential Function Charts）中的行为和转移。

结构化文本是一种高级的文本语言，可以用来描述功能，功能块和程序的行为，还可以在顺序功能流程图中描述步、动作和转变的行为。结构化文本（ST）语言表面上与PASCAL语言很相似，但它是一个专门为工业控制应用开发的编程语言，具有很强的编程能力用于对变量赋值、回调功能和功能块、创建表达式、编写条件语句和迭代程序等。对于熟悉计算机高级语言开发的人员来说，结构化文本（ST）语言更是易学易用，易读易理解，特别是当用有实际意义的标识符、批注来注释时，更是这样。

指令表（IL-Instruction List）语言是一种低级语言，与汇编语言很相似，可以用来描述功能，功能块和程序的行为，还可以在顺序功能流程图中描动作和转变的行为。指令表语言能用于调用，如有条件和无条件地调用功能块和功能，还能执行赋值以及在区段内执行有条件或无条件的转移。指令表语言不但简单易学，而且非常容易实现，可不通过编译和连编就可以下载到PLC。IEC 61131-3的其它语言如功能块图、结构化文本等都可以转换为指令表语言。

功能块图用来描述功能、功能块和程序的行为特征，还可以在顺序功能流程图中描述步、动作和转变的行为特征。功能块图与电子线路图中的信号流图非常相似，在程序中，它可看作两个过程元素之间的信息流。功能块图普遍地应用在过程控制领域。功能块用矩形块来表示，每一功能块的左侧有不少于一个的输入端，在右侧有不少于一个的输出端，功能块的类型名称通常写在块内，但功能块实例的名称通常写在块的上部，功能块的输入输出名称写在块内的输入输出点的相应地方。

采用功能块图用来描述功能、功能块和程序的行为特征，还可以描述步、动作和转变的行为特征。功能块图与电子线路图中的信号流图非常相似，在程序中，它可看作两个过程元素之间的信息流。功能块图普遍地应用在过程控制领域。功能块用矩形块来表示，每一功能块的左侧有不少于一个的输入端，在右侧有不少于一个的输出端，功能块的类型名称通常写在块内，但功能块实例的名称通常写在块的上部，功能块的输入输出名称写在块内的输入输出点的相应地方。下面以整数加法和整数减法说明了本章使用的功能块格式以及对它的解释，如图2.4。

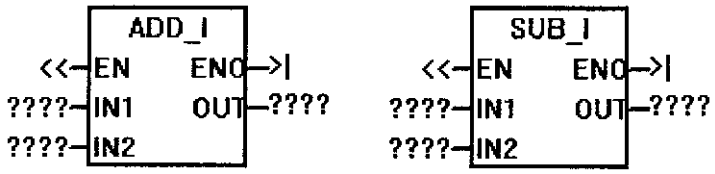

图 2.4 功能块实例

Fig.2.4 The demonstration of function block

在本例中，功能块图中有三个输入（输入总是在图形的左边）和两个输出（输出总是在图形的右边）。在功能块图中使用“能量流”的术语表示流过功能块图逻辑模块的控制流概念，通过功能块图的逻辑“1”称为能量流。在功能块图中，能量流的起点和终点可以直接分配给一个操作数。EN，ENO 为能量流的传输途径。在 EN=1 的条件下，整数加法和整数减法把两个 16 位整数 IN1，IN2 相加或者相减，得到一个 16 位的结果 OUT，并将能流输出到 ENO。如果 EN=0 那么不执行整数加法和整数减法运算。

（1）网络：在功能块图中，程序被分为一些称为网络的一些段，一个网络是功能块图的有序排列。这些元件连接在一起组成一个完整的电路（不存在短路，开路，反向能流）。PLC_Config 允许以网络为单位对程序进行注释。

（2）EN/ENO 定义：EN（允许输入）是功能块中的 BOOL 输入，对要执行的功能块，这个输入必须存在能量流，功能块才会被执行。ENO（允许输出）是功能块中的 BOOL 输出，如果 EN 存在并切功能块正确的执行，那么 ENO 将把能流传递到下一个单元，如果在执行过程中存在错误，那么能流就在出现错误的功能块上停止。

（3）条件/无条件输入：条件输入必须在具备条件的情况下功能块才可执行，无条件输入功能块将被无条件执行，如图 2.5 所示。以下为条件输入指令（JMP），无条件输入指令（NEXT）。

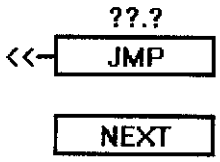

图 2.5 条件输入和无条件输入功能块

Fig.2.5 Conditional and conditionaless input function block

（4）FBD 规约：操作数上的“<<”表示需要一个位变量或者一个能流。操作数上的“>|”表示输出是一个可选的能量流，用于指令的级连。在 FBD 中的取非圆圈表示操作数或者能量流的逻辑非条件用输入端的小圆圈表示。在图中，表示的是 Q0.0 等于 I0.0 的非和 I0.1 的与操作，如图 2.6 所示。

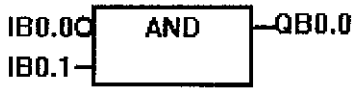

图 2.6 功能块输入取反

Fig.2.6 Invertion input of function block

（5）增加操作数：在 FBD 中对于 AND，OR 功能块如果需要多个位变量进行操作，可对功能块进行增加操作数操作，最多可增至 8 个输入，如图 2.7 所示。

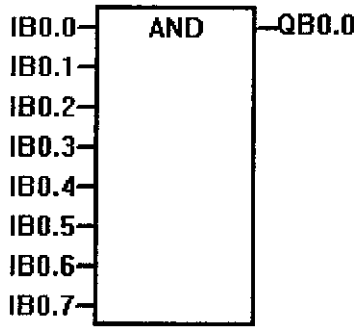

图 2.7 功能块增加操作数

Fig.2.7 Adding inputs for function block

2.3 IEC61131-5 通讯模型[12]

IEC61131-5描述的是PLC的通讯问题，通讯服务包括两个方面：服务器设备和客户设备。 IEC61131-5以国际标准化组织（ISO）的网络的七层协议模型为基础，在第七层应用层之上建立了IEC61131-5的通讯模型，所以从理论上来说，IEC61131-5允许各PLC之间通过任何类型的网络进行通讯[13-14]。

一个PLC可以相当于一个服务器，为客户提供信息和对客户的请求做出反应，也可以相当于一台客户，向服务器请求信息和要求服务。其它的设备，诸如监控系统和其它非IEC61131-3相容设备也可以作服务器或客户。通讯协议（如以太网）允许非限定数量的PLC服务器和客户共存在同一个网络中。在许多情况下，一台PLC既可以一些PLC的服务器，又可以作为其它一些PLC的客户。IEC61131-5标准仅仅定义了在PLC之内的通

讯设备，如PLC A 和B 的通讯设备，并没有定义其它外部的客户的通讯设备，如图2.8。

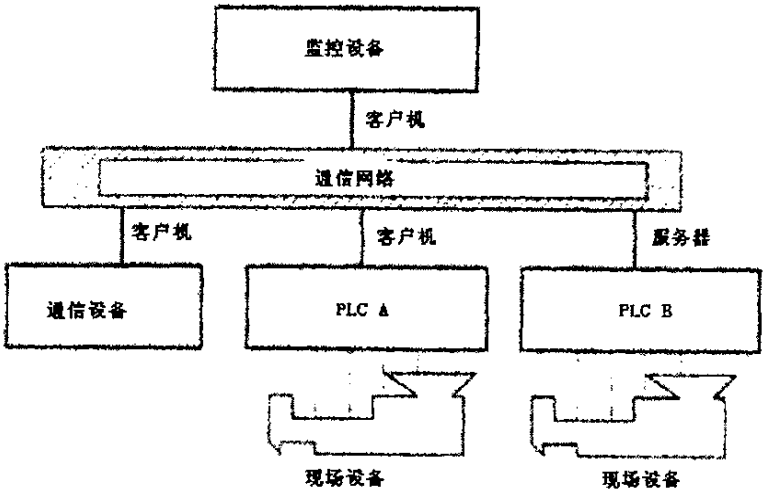

图 2.8 网络结构

Fig. 2.8 Network tropy

2.4 小结

本章主要讲述IEC61131标准的相关内容。介绍了IEC61131-3软件模型，包括组态内部的资源、程序组织单元、变量通信模型等内容。同时也介绍了IEC61131-5通信模型。

3 基于 IEC61131 标准可编程控制器的软件开发

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。它采用可编程序的存贮器，用来在其内部存贮执行逻辑运算、顺序控制、定时、计数和算术运算等操作的指令，并通过数字的、模拟的输入和输出，控制各种类型的机械或生产过程。可编程序控制器及其有关设备，都应按易于与工业控制系统形成一个整体，易于扩充其功能的原则设计。

功能块编程是 PLC 程序编制的一种方式。功能块是现场设备的软件核心，它将现场设备各种功能封装成一个个软件功能单元，每一个功能块由它们的输入、输出、内含参数以及对这些参数进行操作的算法组成。输入、输出参数是功能块提供给外界的接口，内部参数在使用中是不可见的。

功能块包括位逻辑运算（与/或/上升沿/下降沿/线圈/置位/复位/置位优先触发器/复位优先触发器）、数值比较运算（字节/字/双字>，>=，<，<=，!=运算）、字符（串）转换（各种数据类型的转换）、计数器（向上记数器/向下记数器/向上向下记数器）、浮点数值运算（+，-，*，/，SQRT，SIN，COS，TAN，LN，EXP）、整数运算（字/双字的+，-，*，/运算）、字逻辑运算（字节/字/双字的与/或/非/异或运算）、程序流程控制（循环，跳转，停机）、 移位运算（字节/字/双字的左移，右移，循环左移，循环右移指令）、定时器运算（接通延时定时器，关断延时定时器，有记忆接通延时定时器）等[15]。

3.1 软硬件开发环境

在本课题中所使用的设备是本教研室的嵌入式控制模块 DUT5000，软件开发语言则是 Dynamic C。

3.1.1 嵌入式控制模块 DUT5000

DUT5000 以 Z-World 公司的 RCM2250 处理模块为核心，RCM2250 处理模块以 Rabbit2000 嵌入式 8 位微处理器为基础，外扩 512K SRAM 和 512K Flash ROM，采用 RTL8019 作为以太网协议芯片。Rabbit2000 处理器运行速度较快，外接晶振频率为 11.0592MHz，内部使用倍频技术，处理器和外设工作频率均达到 22.1184MHz。与普通 8 位处理器比较，Rabbit2000 内部指令运行所需时钟周期更少。RCM2250 处理模块内嵌掉电检测电路，可以使处理器在上电或电源不稳时被可靠复位，其工作电压范围是 4.75～5.25V[16]。控制器硬件系统结构如图 3.1。

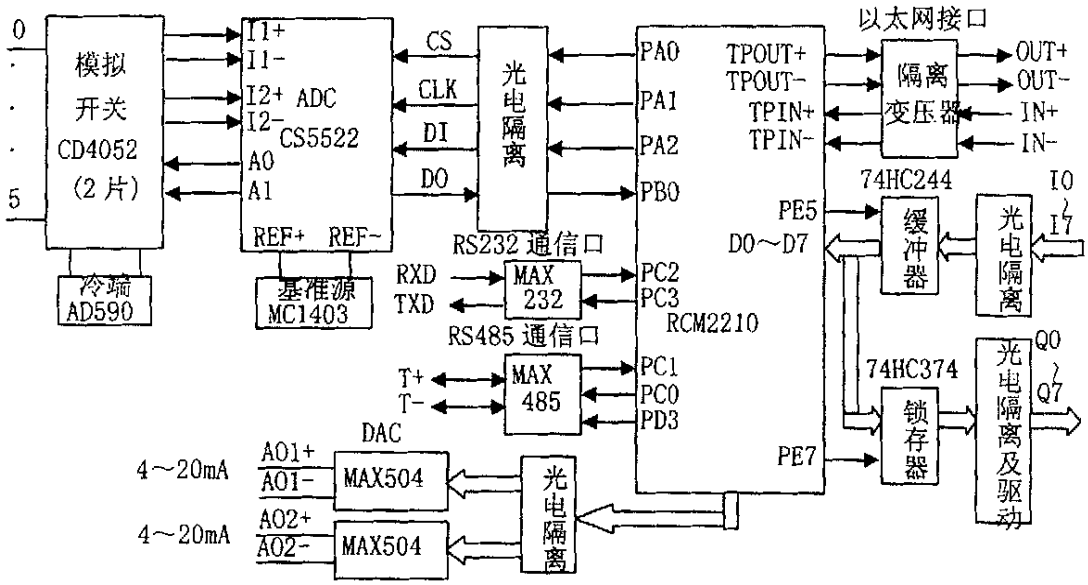

图 3.1 DUT5000 工作原理图

Fig.3.1 The schematic of DUT5000

基于以上设计，DUT5000的资源如下：

（1）开关量输出：8 路隔离 OC 输出，最大驱动能力 500mA。

（2）开关量输入：8 路光偶隔离输入，可接入无源触点，模块给触点提供电源。

（3）模拟量输入：8 路电流/电压 0-5V、4-20mA、各种热电阻、热电偶。

（4）模拟量输出：2 路电流信号输出 4-20mA 或 0-20mA。

（5）通讯接口：2 路隔离 RS485 串行通讯接口，1 个 10 兆以太网端口。

处理器与和编程平台之间通过一条编程仿真接口相连，这是一条十针标准可编程接口的转换电缆，能够通过计算机的串行口控制 Rabbit 处理器实现冷启动、装载程序和在线仿真调试。

3.1.2 开发语言 Dynamic C

应用程序的开发采用 Dynamic C。Dynamic C 是一个专门开发 Rabbit 处理器的嵌入式、多任务、实时操作系统，它含有一个集成开发环境，其中包括：C 编译器、编辑器、链接器、装载器和调试器。使用 Dynamic C 的一个重要特征在于启动时不要求在对象中包含 BIOS 核心程序，这是因为 BIOS 核心程序已经作为一段 C 源程序存储在动态 C 中了，启动时，动态 C 可实现对 BIOS 核心程序进行编译并把它装载到对象中，用户还可以直接查看和修改 BIOS 核心程序。Dynamic C 支持套接字编程，实现了 TCP/IP 协议栈，

并将其封装成几个网络函数库。其中，DCRTCP.LIB 是最主要的网络函数库，该函数库包装了 DNS.LIB、IP.LIB、NET.LIB、TCP.LIB 和 UDP.LIB 等几个网络函数库，这几个库分别实现了 DNS、IP、TCP、UDP 协议。DCRTCP.LIB 和 ARP.LIB、ICMP.LIB 等函数库一起实现了 TCP/IP 协议栈的网络层和传输层[17]。

3.2 IEC61131-3 标准变量通信模型及访问路径的实现

3.2.1 变量通信模型的定义

如第二章所述IEC 61131-3 的通信方式使用访问路径（Access paths）、全局变量（Global variables）、参数调用（Call parameters）、通信组织单元（Communication organization，IEC 61131-5）来说明组态单元的数据交换。

变量的存储是通过定义全局数组实现的。组态内的各程序均可对全局变量进行访问以实现变量通信。其结构如表 3.1 所示：

表 3.1 存储器结构

Tab.3.1 Structure of memory

内存区	内存区名称
I[MEM_SYS_I_LENGTH]	输入过程映像
Q[MEM_SYS_Q_LENGTH]	输出过程映像
S[MEM_SYS_S_LENGTH]	顺序控制存储器
SM[MEM_SYS_SM_LENGTH]	特殊寄存器
……	……

输入映象是用来存储数字量输入的存储器，DUT7000 有 8 路数字量输入，使用一个字节（8 位）便可以存储，但是考虑到以后的扩展方便，采用了宏定义将 I 区域定义为 8 个字节。

```
#define MEM_SYS_I_LENGTH          8
BYTE   I[MEM_SYS_I_LENGTH];     //输入映象
```

输出映象是用来存储 DUT7000 自身的数字量输出的存储器，DUT7000 有 8 路数字量输出，使用一个字节（8 位）便可以存储，但是考虑到以后的扩展方便，采用了宏定义将 Q 区域定义为 8 个字节。

S 变量区是用来存储 DUT7000 步进控制的存储器，DUT7000 在完成步进控制的过程中，要用到一些变量步数，可以步数保存到这个区域，采用了宏定义将 S 区域定义为 32 个字节。

3.2.2 访问路径的研究与实现

控制器将信息存储于不同的存储器单元，每个单元都有唯一的地址，也就是访问路径。用户可明确指出需要存取的存储器地址。这样就允许用户程序直接存取这个地址。

若要存取存储器区域的某一位，则必须指定地址，包括存储器标识符，字节地址以及位号。位寻址如下图所表示，也称为字节.位寻址。在这个例子中，存储器区以及字节地址，I 为过程输入映象寄存器，B 表示字节寻址，3 表示字节 3，字节 3 和位地址（第 4 位）之间用点号“.”相隔开，如图 3.2 所示。

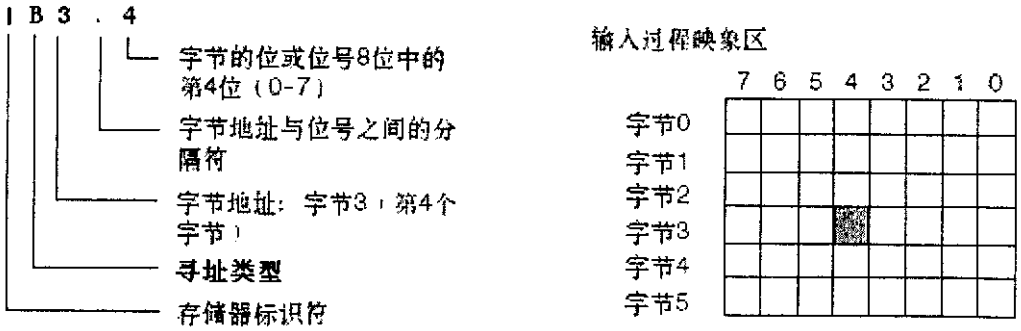

图 3.2 操作数寻址

Fig.3.2 Oprand addressing

如果存取 DUT7000 中的一个字节，字或者双字数据，则必须以类似位寻址的方式给出地址。包括区域标识符，数据大小以及该字节，字或者双字的起始字节地址，如图 3.3 所示。

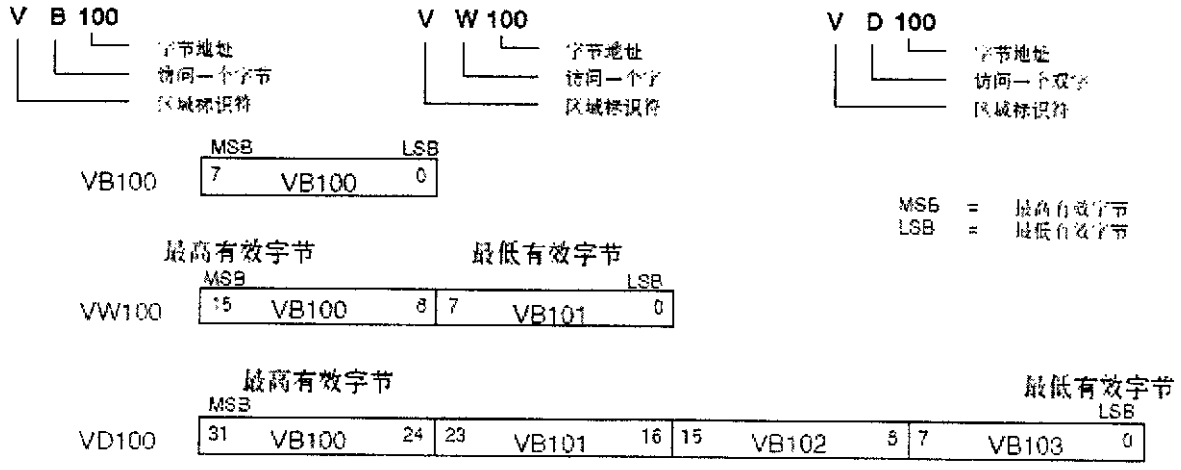

图 3.3 操作数寻址

Fig.3.3 Oprand addressing

下面举例说明 DUT7000 的寻址方式。在程序运行过程中，CPU 对输入数字量点进行取样，并将结果保存到输入映象寄存器中。可按位、字节、字、双字来存取输入映象寄存器。存储器区域的寻址格式如表 3.2 所示：

表 3.2 访问路径举例

Tab.3.2 Addressing examples

寻址方式	访问路径格式	举例
位	IB[起始字节地址].[位地址]	IB0.0
字节	IB[起始字节地址]	IB0
字	IW[起始字节地址]	IW4
双字	ID[起始字节地址]	ID2

程序设计中，需要对访问路径进行寻址的时候，首先要取得要寻址区域的基地址指针，基地址指针定义如下，他是各个寄存器区域也就是各个数组的首地址，并且要防止被意外修改，所以定义为 const 型的指针变量。如下：

```
BYTE   *const pByteBaseV     =V;//变量区域的基地址指针
```

在具体寻址时，比如要寻址 VB35，那么程序首先获得 V 区的基地址指针，调用如下函数可获得，其中函数参数 OprandDomain 表示了当前要寻址的区域是哪个区域，然后根据这个代号返回这个区域的基地址指针。

```
BYTE *GetByteBasePointer(BYTE OprandDomain)
{
   BYTE *pByteTemp;
   switch(OprandDomain)// 根据 OprandDomain 跳转
   {
     case 0:  //V 存储区
       pByteTemp=pByteBaseV;
       break;
      ......
     case 7:  //T 存储区
       pByteTemp=pByteBaseT;
       break;
```

```
        case 8:  //C 存储区
          pByteTemp=pByteBaseC;
          break;
        default:
          break;
      }
      return pByteTemp;//返回基地址指针
    }
```

在获得基地址指针后，根据 VB35 中的 35 可以知道是 V 数组中的第 35 个变量，所以将基地址加偏移量 35 可以获得 VB35 地址的指针。再根据 VB35 中的 B 可以知道要寻址的是字节变量，所以将指针制向的变量取出就是要寻址的变量。如果是 INT 型的变量那么取 2 个字节就是这个要寻址的变量。由于存储的数据和 INT 型的数据大小端对齐方式的不同，所以下面的转换函数中做的一个转换将高低字节顺序颠倒。

```
INT ReadIntFromPByte(BYTE *pBytePointer)
{
   union
   {
     BYTE    byte[2];
     INT     intValue;
    }un;
   un.byte[0]=pBytePointer[0];//存放顺序相反
   un.byte[1]=pBytePointer[1];
   return un.intValue;//返回数据
}
```

如果是位变量，如 VB35.6 表示 V 区域第 35 字节的第 6 位，那么根据下面这个函数可以获得这个位是 0 还是 1。函数根据指针和变量的第几位来得到需要的位变量的值。

```
void WriteBitToPByte(BYTE *pBytePointer,BYTE bitLocation,BYTE bitValue)
{
  BYTE temp;
  temp=1;
```

```
    temp=temp<<(bitLocation);//求 2 的 bitLocation 次方
    if(bitValue==1)//如果要写 1
    {
    *pBytePointer=(*pBytePointer)|temp; //或
    }
    else
    {
       *pBytePointer=(*pBytePointer)&(255-temp); //与
    }
}
```

3.3 IEC61131-3 标准功能块的研究与实现

3.3.1 浮点数加法功能块的定义

由于 IEC61131-3 标准只定义了功能块的外部接口，因此对功能块内部需要自行定义，下面定义的是浮点数加法功能块的结构体。

```
typedef struct tagFloatingPointMathADD_ROprand
{
    WORD    Opcode;//操作码，指示这个指令将进行什么操作
    WORD    OpcodeEx;//操作码扩展
    WORD    NetworkID;//网络 ID
    WORD    FBRunOrder;//功能块执行顺序
    NetworkPosition Position;//功能快在网络中的位置
    Oprand OprandEN;//输入使能信号
    Oprand OprandIN1;//加法操作数 1
    Oprand OprandIN2;//加法操作数 2
    Oprand OprandENO;//输出使能操作数
    Oprand OprandOUT;//加法运算结果
}FloatingPointMathADD_ROprand;
```

其中多次出现了 Oprand 这个结构体，这是定义的操作数结构体。在下位机程序的实现中，控制器的操作数是通过这个结构体描述的。操作数结构体 Oprand 定义如下：

```
typedef struct tagOprand
{
    union OprandHeaderFlag oprandHeaderFlag;//操作数头标志
    union OprandUnion oprandUnion;//操作数内容
}Oprand;
```

操作数头标志表示了这个操作数是地址还是数据（bAddressDataIndicator），如果是地址那么还要说明输入是否取反（**bInputInverse**），因为在输入取反的情况下默认的操作数一定是一个地址，所以使用公用体定义。只占用一个字节。操作数头标志OprandHeaderFlag 定义如下：

```
union OprandHeaderFlag
{
   BYTE bAddressDataIndicator;
   BYTE bInputInverse;
};
```

在使用 OprandHeaderFlag 指示了操作数是地址还是数据的情况下，使用OprandUnion 来表示操作数的具体内容。OprandUnion 是一个公用体，其中 DataAddress 是表示这个操作数是一个地址，DataValue 表示这个操作数是一个数据。操作数内容OprandUnion 定义如下：

```
union OprandUnion
{
   union  DataValue   Data;
   struct DataAddress Address;
};
struct DataAddress
{
   BYTE OprandDomain;        //操作数变量域
   WORD OprandAddress;       //操作数地址
   BYTE OprandEx;            //操作数变量位地址或者扩展
};
union DataValue
{
   BYTE       byteValue[4];
```

```
    INT         intValue[2];
    WORD        wordValue[2];
    DINT        dintValue;
    DWORD       dwordValue;
    REAL        realValue;
};
```

操作数地址定义如下，其中 OprandDomain 表示操作数的区域，也就是这个操作数保存在哪个寄存器区域（实际上是表示保存在哪个数组中）。OprandAddress 表示操作数地址，表示的是在这个数组中的第几个数组下标。在位变量寻址的时候，要使用 OprandEx 表示这个操作数是数组中一个 BYTE 元素的第几位。

```
struct DataAddress
{
    BYTE OprandDomain;          //操作数变量域
    WORD OprandAddress;         //操作数地址
    BYTE OprandEx;          //操作数变量位地址或者扩展
};
```

操作数数据定义如下：

BYTE 寻址时是取 DataValue. ByteValue[0]
INT 寻址时是取 DataValue. intValue [0]
WORD 寻址时是取 DataValue. wordValue [0]
DINT 寻址时是取 DataValue. dintValue
DWORD 寻址时是取 DataValue. dwordValue
REAL 寻址时是取 DataValue. realValue

```
union DataValue
{
    BYTE        byteValue[4];
    INT         intValue[2];
    WORD        wordValue[2];
    DINT        dintValue;
    DWORD       dwordValue;
    REAL        realValue;
```

};

3.3.2 浮点数加法功能块的执行

功能块执行阶段，将对上一节定义的功能块进行具体分析，译码，执行[18]。下面以浮点数加法为例讲述程序的译码执行过程，功能块如图 3.4 所示。

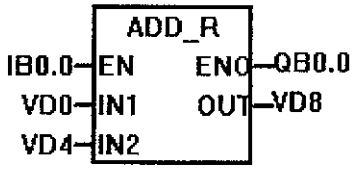

图 3.4 浮点数加法功能块

Fig. 3.4 Floating-point Add function block

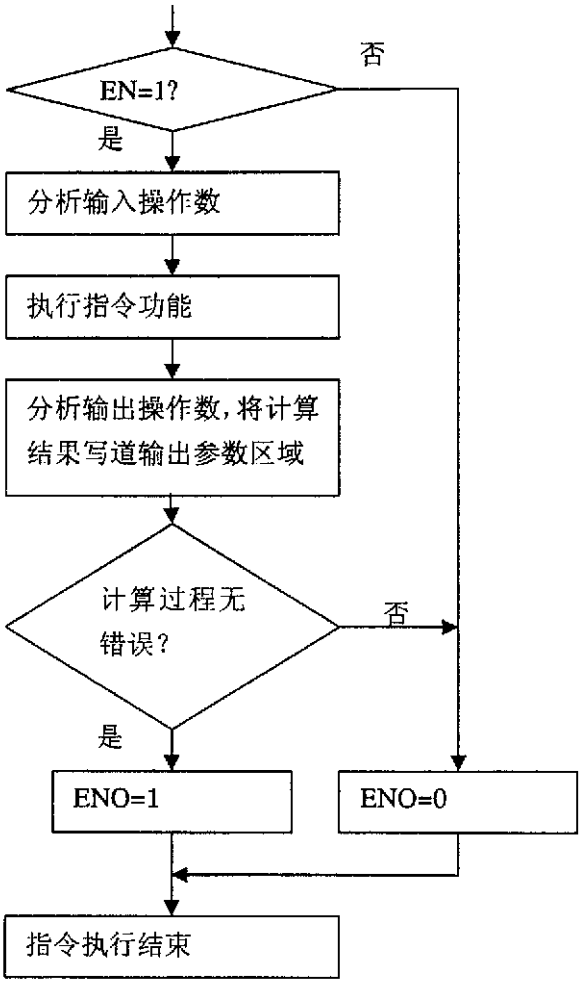

图 3.5 功能块执行过程

Fig. 3.5 Process of instruction

在程序中，首先获得输入使能参数，如果输入使能参数为 0，那么此时功能块不允许执行，程序将直接返回。如果 EN=1，表示功能块允许执行，此时开始执行功能块程序。读出输入参数 1，输入参数 2，进行浮点数加法运算，然后把计算结果写入输出存储器中。在计算结束以后，如果计算过程出现错误，那么 ENO=0，也就是不允许输出使能。如果执行过程没有任何错误，那么输出允许使能。程序返回，执行下一个功能块。其流程如图 3.5 所示。

```
void ExeFloatingPointMath_ADD_R()
{
    //变量定义
```

```
    //读取指令内容 floatingPointMathADD_ROprand
    //获得输入使能参数
    EN=ReadBitFromPByte(pByteMem,OprandBitLocation);
    if(EN==0)//根据输入使能参数进行相关计算
    {
      //获得 ENO 输出参数
      ENO=0;//不允许使能输出，程序返回
      return;
    }
    IN1=ReadParamIN1();//读加法参数 1
    IN2=ReadParamIN2();//读加法参数 2
    OUT=IN1+IN2;//计算输出结果
    ENO=0;//写使能输出
    If(Error==0)//如果计算过程无错误
    {
     ENO=1;
    }
    Return;
}
```

3.3.3 可编程控制器任务的扫描执行

PLC 是靠执行用户程序来实现控制要求。在存储器中设置 I/O 映象区，分别存放执行程序前的各输入状态和执行过程中各结果的状态。PLC 对用户程序的执行是以循环扫描方式进行的，用来描述 CPU 对程序顺序、分时操作的过程[19-20]。

PLC 开始运行时，首先清除 I/O 映象区的内容，然后进行自诊断，自检 CPU 及 I/O 组件，确认正常后开始循环扫描。每个扫描过程分为三个阶段进行，即输入采样、程序执行、输出刷新。三个阶段的执行时间就是一个工作周期（或扫描周期）。

（1）输入采样阶段

在输入采样阶段，PLC 以扫描方式按顺序将所有输入端的输入信号状态并存入映象寄存器去，也称作输入刷新。接着进入程序执行阶段。在输入采样阶段结束后，即使输入信号发生改变，输入映象区中的状态也不会改变。

（2）程序执行阶段

在这个阶段，PLC 对程序按顺序进行扫描，也是程序处理阶段。若是程序用梯形图表示，则总是按先上后下、先左后右的顺序对由接点构成的控制线路进行逻辑运算，然后根据逻辑运算的结果，刷新输出映象寄存器区或系统 RAM 区对应位的状态。在此阶段，只有输入映象寄存器区存放的输入采样值不会发生改变。

（3）输出刷新阶段

程序执行后，进入输出刷新阶段。此时，将输出映象寄存器区中所有输出继电器的状态转存到锁存电路，通过输出端驱动用户输出设备（负载），这就是 PLC 的实际输出。

用户程序将以以下方式执行，待指令下载到 DUT7000 的用户程序存储器后，其格式是顺序存储的。执行时采用了取指令、指令译码，指令执行这 3 个步骤循环交替[18]，如图 3.6 所示。

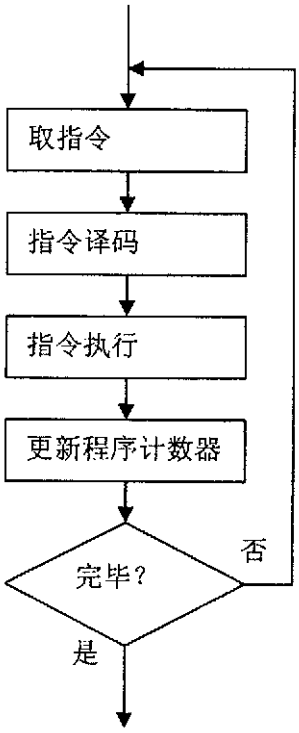

图 3.6 指令执行

Fig. 3.6 Execution of instruction

取指令阶段，系统将从用户程序区取出指令码，在程序设计中，定义了 PC 指针，也就是程序计数器，他表示了当前要执行指令的地址，根据这个地址读出指令码，指令码表示了指令要执行何种操作。如下程序所表示，调用系统函数将指令码从用户程序取读出：

xmem2root((BYTE *)&OpcodeValue,UserCodeHeader+PC,sizeof(WORD));

取得操作码以后，程序将对操作码进行分析，判断出指令要进行何种操作，然后进行不同的功能调用。程序将首先根据读出的操作码的类型判断需要执行的操作的类型，如下所表示：

```
switch(OpcodeValue)//根据操作码执行不同的操作
{
        case 1:
        ExeBitLogic_ANDFB();
        break;
        case 2:
        ExeBitLogic_ORFB();
        break;
        case 3:
        ExeBitLogic_PFB();
          break;
        ……
    Default:
       Break;
}
```

在找到相应的操作后，对程序进行功能调用，以实现功能快要执行的功能。

3.4 I/O 扩展模块通讯处理

输入和输出是系统的控制点，输入信号来自现场设备（如传感器和开关），而输出则控制生产过程中的泵，电动机或者其他设备。可以使用本机 I/O 或扩展 I/O 提供。DUT7000 本身具有一定数量的本机 I/O 数字点。在不满足应用的情况下，可以使用扩展 I/O 点来满足需求。DUT7000 可通过串行口（RS485）对 I/O 量进行扩展，如图 3.7 所示。

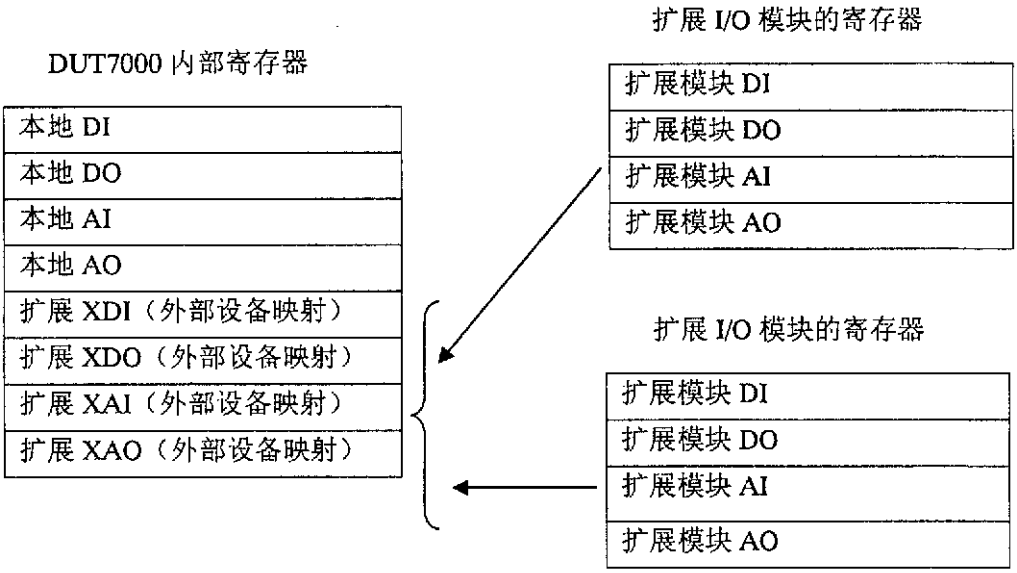

图 3.7 存储器映射

Fig. 3.7 Memory mapping of register

DUT7000 提供的本机 I/O 具有固定的 I/O 地址，当使用串行总线扩展 I/O 的时候，扩展 I/O 的模块的 I/O 寄存器地址是由该模块的模块号决定的。通过寄存器映射将外部扩展 I/O 模块的寄存器映射到 DUT7000 的寄存器中。对扩展模块的 I/O 编程时只需对映像寄存器编程。一个 DUT7000 可以支持多达 16 个外部扩展 I/O 模块。目前支持的模块如表 3.3 所示：

表 3.3 DUT7000 支持的扩展模块

Tab.3.3 Module supported by DUT7000

模块类型	DI 数量	DQ 数量	AI 数量	AQ 数量
DIO1000	24	12	0	0
DUT4000	4	9	8	0
DUT6000	4	9	8	0

为了在各种模块下保持程序的兼容性，将每个设备的存储区域定义为各种模块中最大的一个。也就是 DI 占用 3 字节，DQ 占用 2 字节，AI 占用 16 字节。

每个模块根据他的模块地址占用存储器中的位置。如一个 DIO1000 模块，他的模块地址为 2，那么他占用了 XIB[6]-XIB[8]，XQB[4]-XQB[5]，虽然他没有 AI 可是也占用了 XAI[32]-XAI[37]。

如表所示是一个本机 I/O 和扩展 I/O 举例，DUT7000 主控制器扩展了一个 DUT4000，一个 DUT6000 以及两个 DIO1000。其存储器分配如表 3.4 所示：

表 3.4 存储器分配

Tab.3.4 Mempry allocating

主控制器	I/O 扩展模块 1	I/O 扩展模块 2	I/O 扩展模块 3	I/O 扩展模块 4
DUT7000	DUT4000	DIO1000	DIO1000	DUT6000
IB0	XIB0-XIB2	XIB3-XIB5	XIB6-XIB8	XIB9-XIB11
QB0	XQB0-XQB1	XQB2-XQB3	XQB4-XQB5	XQB6-XQB7
AIW0-AIW12	XAI0-XAI15	XAI16-XAI31	XAI32-XAI47	XAI48-XAI63
AQW0-AQW4				

DUT7000 本身具有一定数量的本机 I/O 数字点。在不满足应用的情况下，可以使用扩展 I/O 点来满足需求。DUT7000 可通过串行口与扩展 I/O 模块通信，将扩展 I/O 的模拟量已以及数字量进行扩展。串行通信采用 Modbus 协议，不加中继的情况下，总线上可以挂载 32 个设备。其结构如图 3.8 所示：

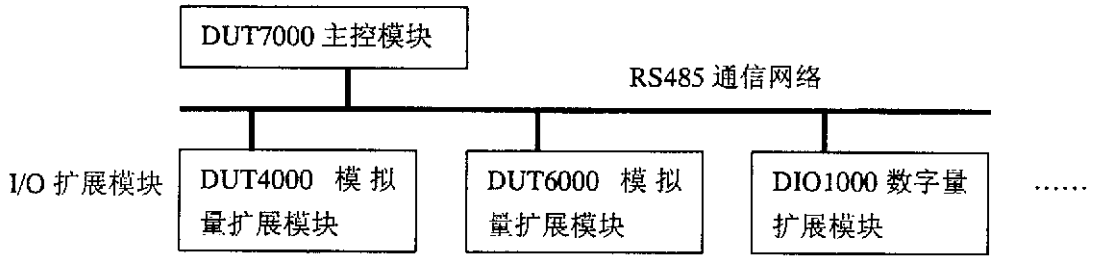

图 3.8 DUT7000 作为主控制器的系统结构

Fig.3.8 The system architecture of DUT7000 used as area controller

每个串行口均有一个串行通信缓冲区和一个串行通信数据区，接收数据时，将接收字符首先放入通信缓冲区，当缓冲区满或接收的两个字符之间的时间间隔超过 3.5 个字

符，则认为一次接收操作完成，并将缓冲区中字符读入到通信数据区。然后根据协议设置，对接收数据按照不同协议语法进行检查。若检查有错误，则说明接收字符不正确，予以丢弃，清空通信缓冲区和数据区，并置串行口为接收状态，开始下一次接收操作；若检查正确，则根据具体协议进行译码，执行相应操作，并对命令做出响应。串行通信的发送和接收操作均由发光二极管指示。通信错误时，置通信错误标志位，由主程序的错误处理模块进行处理。

串行通讯处理函数，在用户程序执行最后的时候被调用，用于刷新 DUT7000 内部缓冲区中的数据。串行通讯管理采用完全异步管理方式，两个串行缓冲队列自动维护消息的发送和接收，如图 3.9 所示。单个 DUT7000 目前可以带 16 个从设备，根据需要还可以继续扩充。

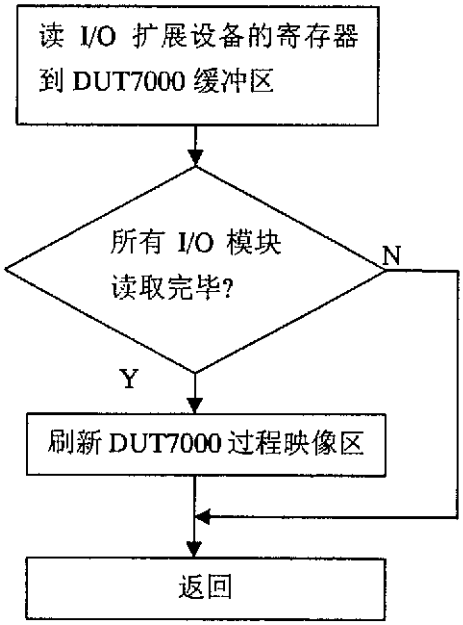

图 3.9 I/O 寄存器刷新

Fig.3.9 I/O register refreshing

3.5 用户程序的上载与下载

用户程序的上载/下载是通过以太网通信实现的。对于以太网 UDP 协议，在程序初始化时打开 UDP 端口，支持读服务、写服务和变量发布服务，流程图如图 3-5 所示。以太网通信的发送和接收操作分别由 udp_send() 和 udp_recvfrom() 函数完成，其中

udp_recvfrom 可以从以太网链路层直接获得发送数据包的主机 IP 地址，根据接收报文的第一个字节决定服务类型，然后分别由不同的服务响应函数给出响应。当以太网发送或接收错误时，置以太网错误标志，由主程序的错误处理模块进行处理，其流程如图 3.10 所示。读写服务根据报文中的 Destination AppID，Destination ObjectID 和 SubIndex 来区分对不同变量区，不同变量的操作。下面介绍一下用户程序的下载和上载过程[21-24]。

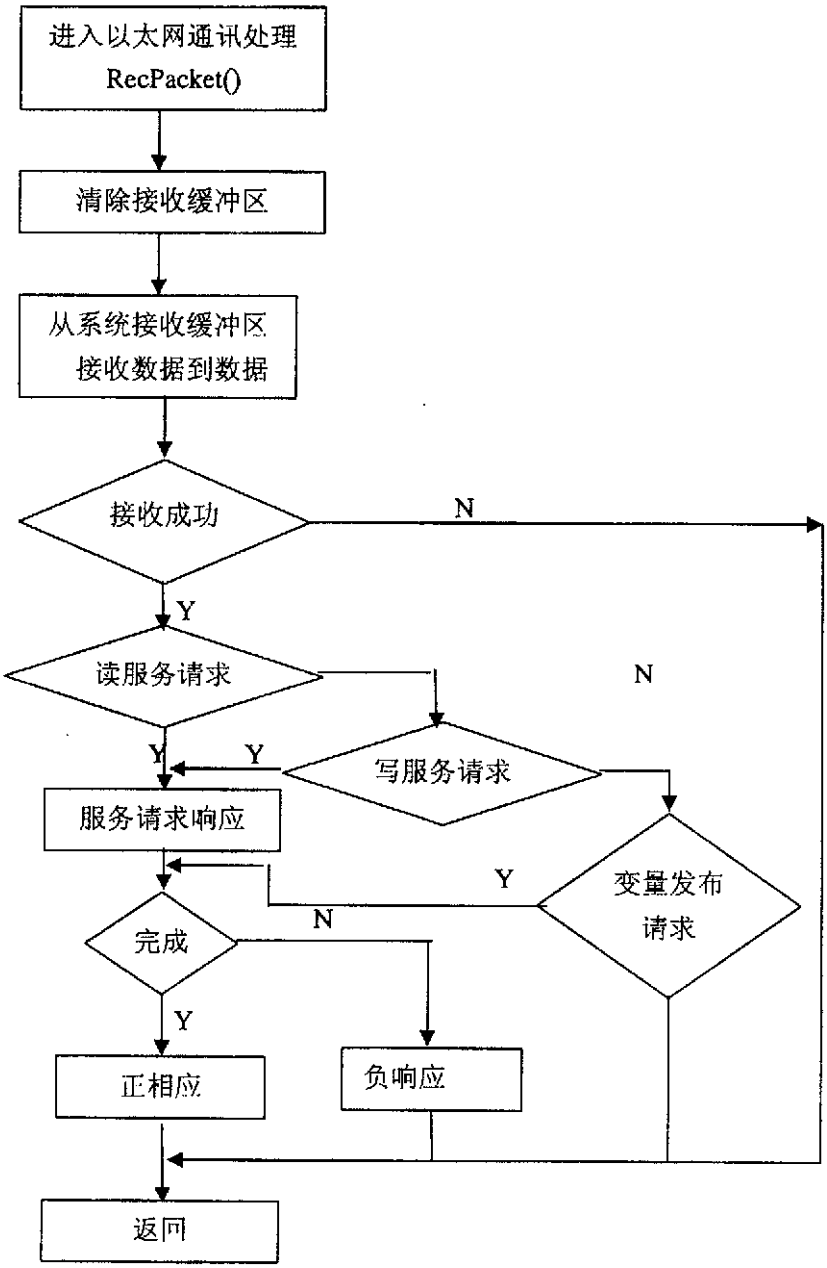

图 3.10 以太网通信处理

Fig.3.10 Processing of ethernet communication

3.5.1 用户程序的无扰动下载

控制器中运行的数据是从工程师站组态后下装到控制器中的。在早期的 DCS 控制器中，都是将程序和数据写入 EPROM 中，如果修改了程序和数据，便要将 EPROM 片子从控制器上拔下，通过 EPROM 写入器擦除原有内容并重新写入新的程序和数据。目前大多数 DCS 系统都提供了静态随机存储器 SRAM，用来存储下装的数据和控制程序。计算机系统通过网络就可以直接下装程序和数据。一般，计算机控制组态完成后，经过与数据库的联编成功后，便可通过下装软件下装到控制器中运行。数据和控制程序一次下装之后，如果没有变化，不应每次启动都下装。但实际上，大多数控制系统都不可能做到一次下装后再也不修改，系统在运行过程中总免不了对组态进行修改。因此，控制系统的数据下装分为两种，一种是生成全部下装文件，一种是生成增量下装文件。全下装是全部组态数据编译后生成的全联编，联编成功后，进行系统库全部下装，此种下装模式，需要对控制器重新启动；增量下装是只下装修改和追加部分的内容，控制器以一种增量方式追加在原数据库中。增量下装为一种无扰在线下装模式，不需要停止控制器的运行，便可实现对控制方案的修改。

3.5.2 下装涉及到的关键技术问题

要实现稳定安全的下装，有下列技术问题需要考虑：

（1）实时通讯技术：上一小节讲到，现代的控制器多采用网络来进行控制程序和数据的下装，这样，可靠的实时通讯就显得十分重要。以太网技术在商业网络系统中得到了广泛的应用，但由于传统的以太网技术采用总线式的拓扑结构和多路存取载波侦听(CSMA/CD)通讯方式，信息碰撞和网络堵塞的概率非常大，在数据传送实时性要求很高的工业场合是不能容忍的。现在网络的通讯速率大大提高，同时又增加了一些新的技术，比如在现场级利用星型拓扑结构代替总线结构，采用环冗余结构等，形成了新的工业以太网标准，能够适应工业现场的需要。

(2)冗余技术：冗余技术指的是用多于一种的途径来完成某一规定功能的设计技术。冗余技术可提高系统或设备的任务可靠性，但会降低系统或设备的基本可靠性。只有在采用更好的元器件和采用简化设计、降额设计等方法都无法满足系统或设备可靠性要求时，或改进元器件所须费用比系统或设备采用冗余技术的费用更高时，或当开发更可靠的基础元器件的速度赶不上新装备、新设备的发展速度时，冗杂技术才成为可供采用的方法。冗余技术分为工作冗余和非工作冗余（备用冗余）两类。

（3）控制器上的实时任务调度：控制器上往往需要有许多复杂的任务在同时执行，比如在无扰下装的时候，既要建立通讯连接来接收下装的程序和数据，又要不间断 IEC 运算的执行，同时还要维持与操作员站的通讯以及与从机的冗余。因此，控制器上通常建立多线程的任务结构，采用嵌入式实时操作系统来调度任务的执行。实时操作系统（RTOS）是指能对外部事件在限定的时间内作出并完成响应的多任务操作系统，它是嵌入式计算机中的重要的系统资源，与通用平台的操作系统不同，它往往嵌入到目标机硬件设备内部运行。常用的嵌入式实时操作系统有 VxWorks，QNX，ucos、嵌入式 Linux 等。当然，控制器也可以采用单任务结构，但是要精确地控制程序的执行流程，确保实时性和可靠性，起到和多任务结构相当甚至更好的执行效果。

3.5.3 无扰动下载方法的设计与实现

用户程序的下载是将程序员编写好的程序通过与 PLC 的通信，下载到 PLC 中，传统的 PLC 系统在用户编写程序后，需要将 PLC 停止，然后下载程序到 PLC，若属于调试程序阶段，停止 PLC 运行不会影响到生产现场，如果 PLC 在生产现场处于运行状态，停止 PLC 可能会影响到整个生产线。故提出了 PLC 的无扰动下载技术。采用双用户程序区域，2 个区域互为程序运行区域与待下载区域。如图 3.11 所示：

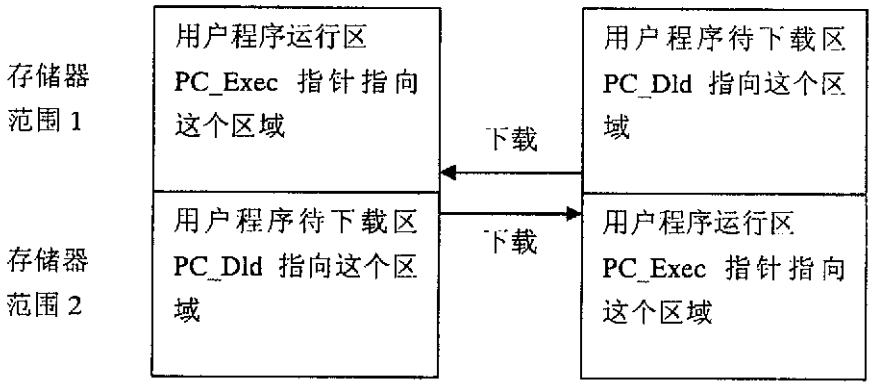

图 3.11 程序下载

Fig. 3.11 Program downloading

程序实现中将用户程序存储器分为两个范围，也就是存储器范围 1 和存储器范围 2。用户程序下载到存储器范围 1 后，PC_Exec 指针指向这个区域，PC_Dld 指向存储器范围 2。其中 PC_Exec 是程序执行使用的程序计数器。PC_Dld 是供运行期间下载使用的程序

计数器。此时，程序将在存储器范围 1 内读取指令并执行。存储器范围 2 待下载新的程序。

当需要下载新的程序时，程序接收到下载命令后，并不需要停止 PLC 的运行，只需要将程序存储于存储器范围 2 内，在下载结束后，将 PC_Exe 指针指向存储器范围 2，而 PC_Dld 指向存储器范围 1。这样便可实现 2 个程序空间的互相切换，也就实现了 PLC 程序的无扰动下载。

下载过程使用以太网 UDP 通信，在上层管理程序中保证传输数据的完整性和连续性。其流程如图 3.12 所示：

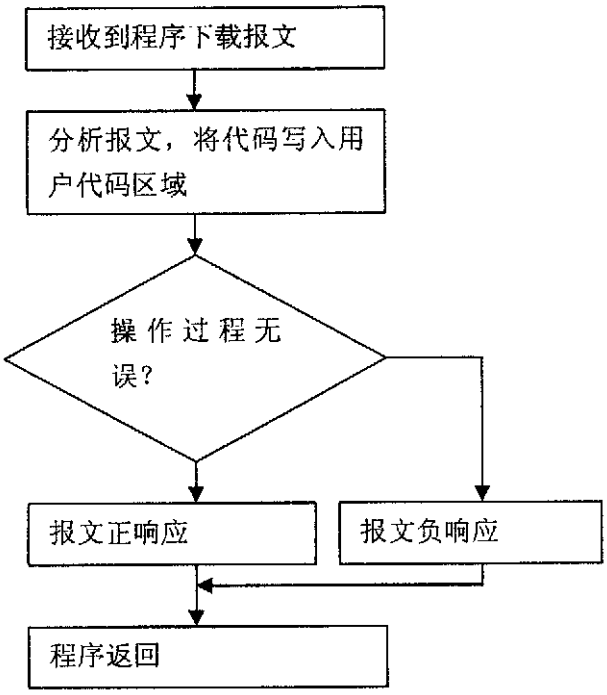

图 3.12 程序下载流程

Fig. 3.12 Process of program downloading

在接收到报文信息后，首先判断报文信息的正确性，如果报文正确，那么进行用户代码写入到存储器的操作，如果操作顺利完成，那么系统将给予正响应报文，以通知上位机发送下一个用户代码报文。如果操作过程中出现错误，那么进行报文负响应，以通知上位机进行相应的处理。

3.5.4 用户程序的上载以及程序比较和加密

在实际应用过程中，由于编程者的疏忽或者由于软件版本控制不当，常将控制器内的程序丢失或者不知道控制器内当前运行程序的版本号。有时为了保护知识产权不受到侵犯，常长使用程序加密的措施，使程序不受到不法侵害。基于以上原因设计时考虑了PLC 程序的上载，程序比较以及加密措施。

程序的上载主要是主机发送命令，将 PLC 中的代码信息传送到上位机，上位机对代码进行解释分析，并将功能块程序复现。

在将控制器程序上载后，可以实现与现存程序的比较，比较将给出程序的哪个网络相同，哪个网络不同，如果不同，还将给出网络中哪个功能块不同，或者功能块的哪个参数不同。

程序加密则使用密码保护的方式，由用户在控制器内设置密码，在上载程序之前先要求输入密码，如果密码正确则可以上载，如果密码输入不正确，则不能对现有程序进行上载，以起到保护程序的作用。

3.6 小结

本章介绍了 DUT7000 可编程控制器的指令格式以及定义，简述了 DUT7000 的操作数结构以及支持的功能块类型。通过介绍可编程控制器 PLC 扫描执行技术，详细讲述了用户程序的上载、下载以及执行，并重点讲述了实现这些功能所需要的通信程序支持。

4 基于 IEC61131-3 标准的功能块组态软件的设计和实现

一个控制系统的组成由网络上相互协调工作的多个设备组成，为了完成特定控制任务，需要对系统进行组态配置，其中包括功能块程序的编制，模块信息的设定，以及网络层次以及 I/O 设备的组态。组态软件收集整个控制网络的设备信息及网络信息，根据控制要求及控制目的，对现场网络的所有资源进行有效的规划和配置，最终将整个网络构建成一个控制系统[25-29]。组态软件的开发工具采用 VC++6.0，本课题所开发的组态软件的名称为 PLC_Config。

4.1 概述

PLC_Config 主要有以下几个模块组成：现场设备管理模块、工程管理模块、可视化功能块组态模块、现场设备监控模块、通信服务模块及信息报告模块，各个模块之间的功能块组态模块、现场设备监控模块、通信服务模块及信息报告模块，各个模块之间的关系如图 4.1 所示[30]。

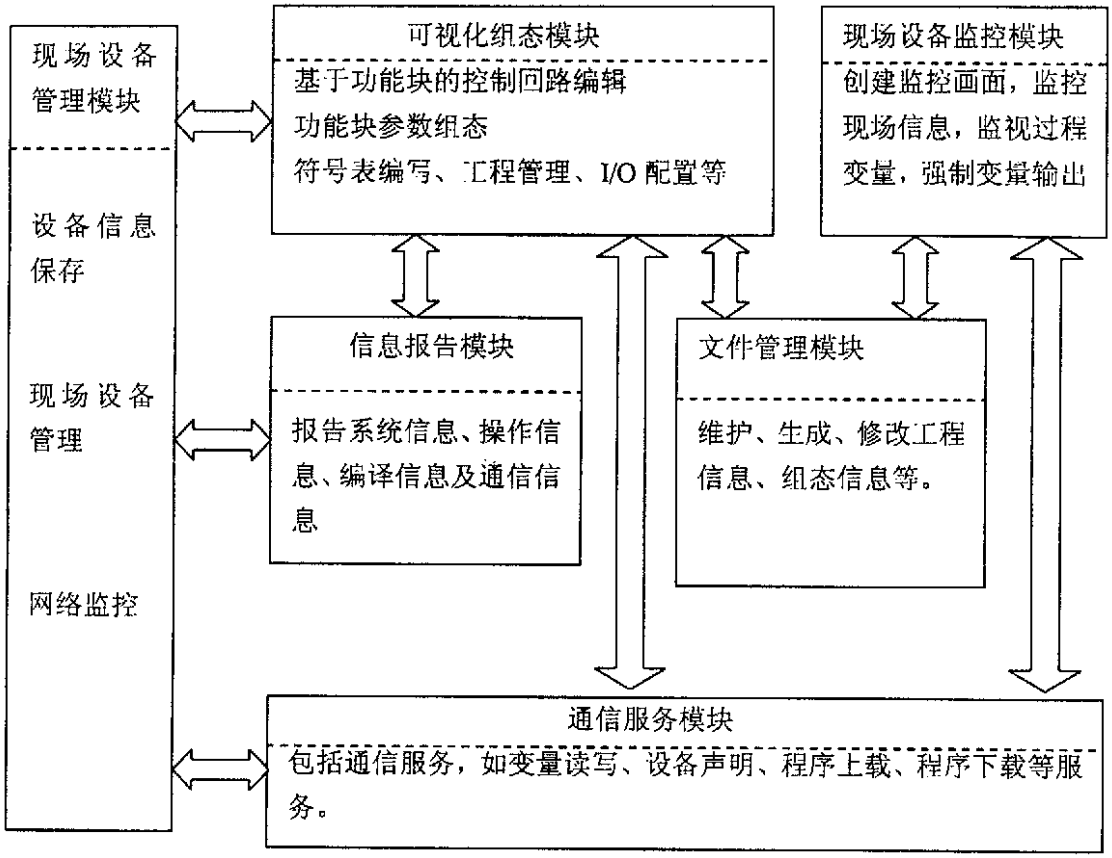

图 4.1 PLC_Config 的总体模块划分结构

Fig.4.1 The module structure of PLC_Config

可视化组态模块是整个组态软件的核心部分，实现了 PLC 应用组态的可视化操作。现场设备管理模块负责维护、修改控制策略。现场设备监控模块负责监控现场网络的运行情况，同时也可强制过程变量的值。文件管理模块维护、生成、修改工程信息、组态信息等，并保存功能块回路的组态信息。通信服务模块提供了系统中的各类通信服务。信息报告模块负责及时通报各类系统信息，如设备信息、网络信息、操作信息、编译信息等。

PLC_Config 以工程的方式对组态信息进行管理。一个工程对应一个工程目录，工程目录结构如图 4-2 所示。每个工程目录包含一个工程文件，功能块程序文件，状态表，符号表等文件。工程文件描述整个工程的总体框架，对工程下的所有文件进行有效的组织管理。程序块文件保存了设备的功能块组态信息，这些信息可以形成代码，并下载到控制器。符号表主要是为了使用符号编程，方便程序的维护和修改，以及程序的可读性。

状态表用于过程变量的监视以及强制。通信配置可配置当前网络的信息，配置扩展 I/O 模块的信息。其结构如图 4.2 所示：

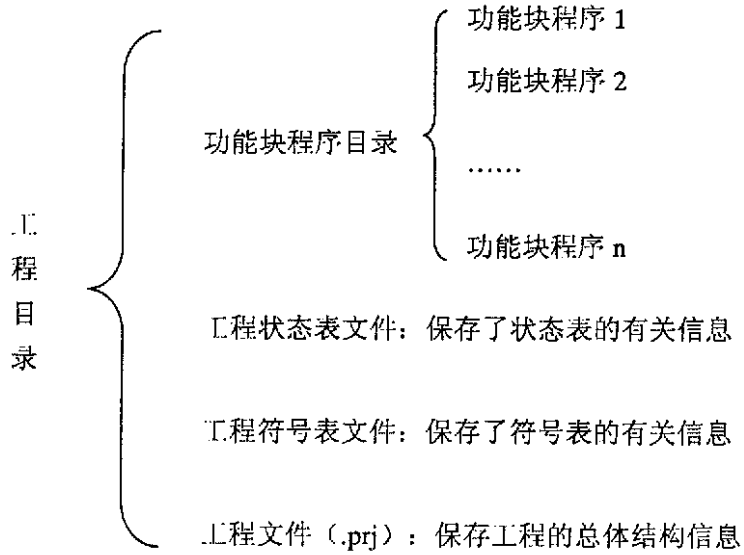

图 4.2 PLC_Config 所创建工程的目录结构

Fig.4.2 The project structure created by PLC_Config

PLC_Config 启动时首先要调用文件管理模块、现场设备管理模块的初始化函数，对这两个模块进行初始化，然后新建或打开一个工程，启动工程，同时打开网络套接字映射接口对象，监听网络设备信息，每发现一个在线设备就在设备管理窗口中添加该设备相关信息。创建一个功能块文件，选择功能块添加到控制回路编辑窗口中，通过鼠标操作建立链接对象，确定功能块之间的连接关系及功能块的执行顺序，形成一个完整的控制回路，对各个功能块进行参数配置。最后对组态信息进行编译，下载组态信息到现场设备中。

4.2 现场设备管理模块

现场设备管理模块负责侦听网络设备信息，维护和管理现场设备信息。本模块至底向上可分为三层：网络层、设备管理层、后台数据库层。如图 4.3 所示。

实时网络监控层：启动套接字映射接口对象，监听网络上的设备信息，将获取的设备相关信息转交给现场设备管理层处理。

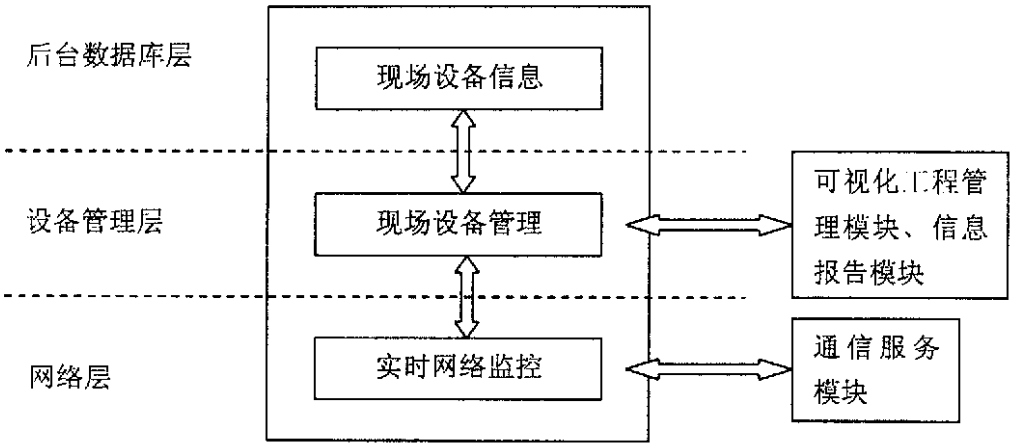

图 4.3 现场设备管理模块的三层体系结构

Fig.4.3 The three layer structure of device manage module

现场设备管理层：该层负责接收来自实时网络监控层的消息，对该消息进行处理，将消息报告给可视化工程管理模块、信息报告模块，维护后台数据库。该层是现场设备管理模块的核心层，是该模块和其他模块交互的主要接口。

现场设备信息层：该层维护一个 Access 数据库，存放现场设备的信息。该数据库的主表是一个名为 DevOnline 数据表。DevOnline 数据表是一个动态变化的数据表，该数据表存放的是目前在线的设备，根据设备在线、离线状态，及时地删除和添加设备信息，供组态使用[31]。

4.3 文件管理模块

在 PLC 应用系统中用文件来描述设备信息，保存回路信息以及符号表和状态表。PLC_Config 需指定一个专用目录来存放各类应用程序文件，将这个目录定义为应用程序根目录。该目录下的各种文件就是应用程序生成的工程相关的文件。

PRJ 后缀的文件是用来保存工程信息的，其中包含工程名，工程所在的根目录，以及工程的配置信息等。

FBD 后缀的文件是用来保存功能块回路的组态信息的。其中包括对功能块网络的保存，网络中功能块的保存以及功能块参数的保存、网络注释的保存等编译需要使用的信息。

STC 后缀的文件是用来保存状态表配置的，状态表是用来调试的时候观察 PLC 内部数据的。同时也可以对内部数据进行强制。

SBL 后缀的文件是用来保存符号表的，符号表是用来保存变量定义的，为了增强程序的可读性以及修改程序的方便性，需要使用符号表。

系统中文件的创建，保存，以及修改都是通过 MFC 框架的序列化函数实现的。Serialize()序列化函数是一个虚函数，因此可以借助这个函数来实现对文件数据的读取，保存或者修改[32-33]。下面以状态表为例，讲述其保存以及打开的代码：

```
void CStatusChartDoc::Serialize(CArchive& ar)
{
    int i,TableSize;
    if (ar.IsStoring())
    {
        // TODO: add storing code here
        ar<<m_StatusChartName;//保存状态表名称
        TableSize=m_StatusNodeTable.GetSize();//获得状态表大小并保存
        ar<<TableSize;
        for(i=0;i<TableSize;i++)//保存状态表内容
        {
            StatusNode *pStatus;
            pStatus=m_StatusNodeTable.GetAt(i);//获得链表节点指针
            ar<<pStatus->AppID;//保存数据
            ar<<pStatus->BitPosition;
            ar<<pStatus->DataPosition;
            ar<<pStatus->ObjID;
            ar<<pStatus->strAddress;
            ar<<pStatus->strDataType;
            ar<<pStatus->strNewData;
            ar<<pStatus->strSymbolName;
            ar<<pStatus->SubIndex;
        }
    }
    else
    {
        // TODO: add loading code here
        m_StatusNodeTable.RemoveAll();//清所有内容
        ar>>m_StatusChartName;//读状态表名称
```

```
            ar>>TableSize;//读状态表大小
            for(i=0;i<TableSize;i++)//读状态表内容到链表
            {
                StatusNode *pStatus=new StatusNode;//动态生成一个新节点
                ar>>pStatus->AppID;
                ar>>pStatus->BitPosition;
                ar>>pStatus->DataPosition;
                ar>>pStatus->ObjID;
                ar>>pStatus->strAddress;
                ar>>pStatus->strDataType;
                ar>>pStatus->strNewData;
                ar>>pStatus->strSymbolName;
                ar>>pStatus->SubIndex;
                m_StatusNodeTable.Add(pStatus);
            }
        }
    }
```

首先判断 ar.IsStoring()函数，以判断当前是保存还是读取文件。如果是保存文件，那么将状态表链表中的内容依次读出，并将其保存到文件中。保存数据的操作是通过重载运算符<<完成的。如果是读取文件，那么借助>>操作符将文件中的数据读出并保存到链表中。

4.4 可视化组态模块

可视化组态模块主要由以下几个部分组成：基于功能块的控制回路编辑、功能块参数配置、可视化工程管理、符号表编写、状态表编写等[34]。

4.4.1 可视化工程管理

PLC_Config 组态软件是以工程的方式来管理和组织信息，而这是通过可视化工程管理窗口来实现的。图 4-5 表示一个新创建的工程，工程名为 project1.prj。如图所示，一个新创建的工程会自动为工程添加一个节点，节点名为 project1.prj。在 project1.prj 节点下面有四个节点：程序块节点、符号表节点、状态表节点、通信配置节点、指令节点。

程序块节点负责管理当前由用户创建的所有程序。程序块节点负责管理所有功能块控制策略。可通过鼠标右键操作，创建一个新的控制策略文件，每一个控制策略文件对

应一种控制策略。一个控制策略可包含多个网络，每个网络中由各种基本功能块组成复杂的运算逻辑，用户可创建并选择合适的回路文件编译并下载。双击某一控制策略节点，将打开控制策略编辑窗口，在该窗口中完成控制回路组态。控制策略编辑窗口是基于 VC++的 Document/View 结构，在本程序实现中，CFBDDoc 类、CFBDView 类和 CFBDFrame 类构成了控制策略编辑窗口的三位一体的 Document/View 结构。如图 4.4 所示。

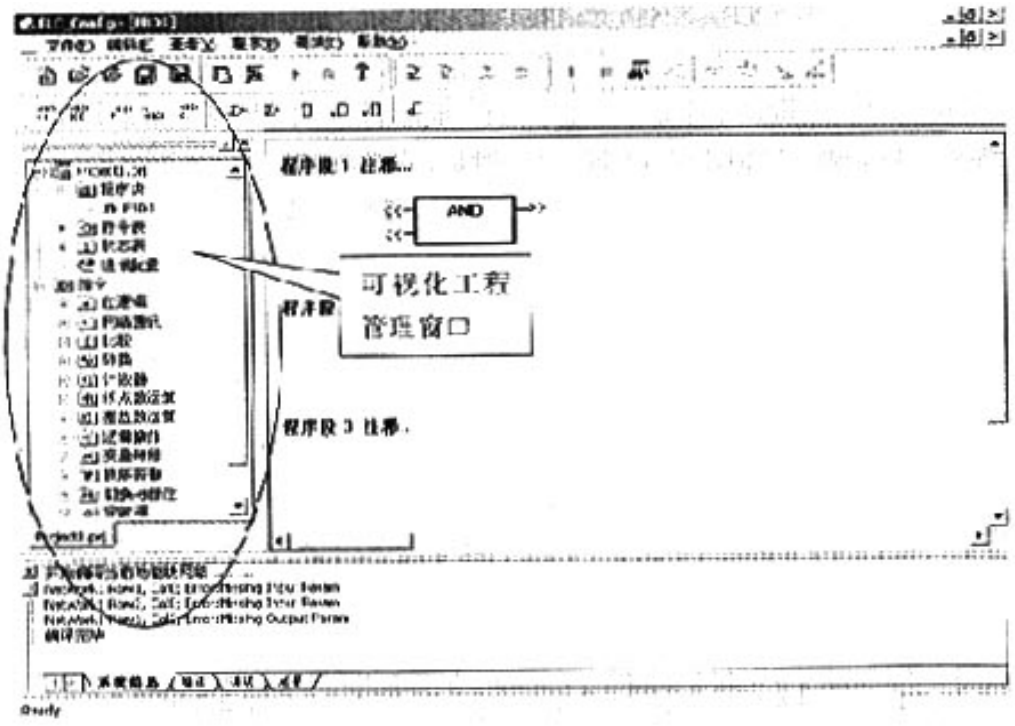

图 4.4 PLC_Config 中的工程窗口

Fig.4.4 The project window of PLC_Config

通讯配置节点负责配置所有在线的 DUT7000 的通信参数，扩展 I/O 模块的参数。通过鼠标双击可弹出通讯配置对话框，完成上述操作。工程管理窗口对应的视图类 CProjectView、文档类是 CProjectDoc，如图 4.7 所示。项目是工程管理的基本组织单位，图 4.5 表示的是项目的数据结构图。

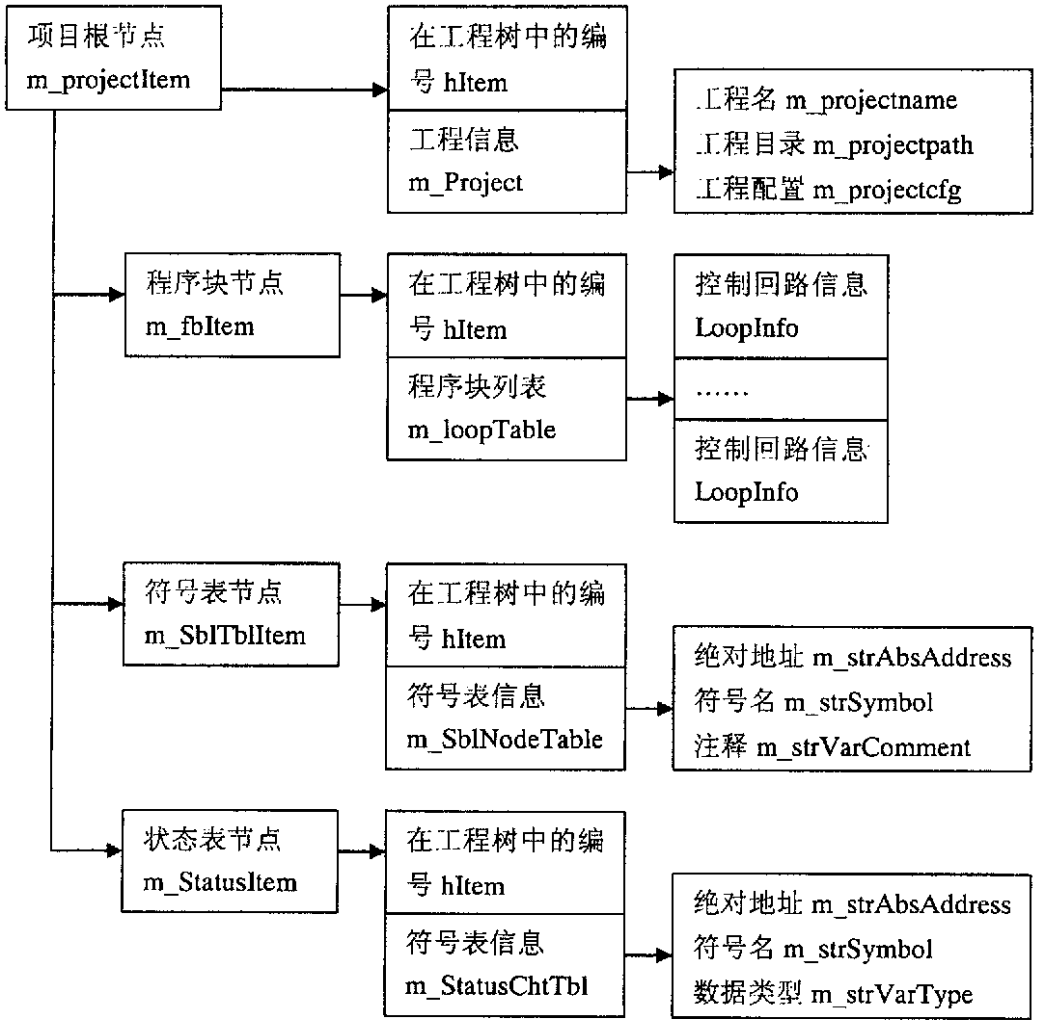

图 4.5 工程结构

Fig.4.5 Project structre

4.4.2 基于功能块的控制回路策略编辑

PLC_Config 组态程序的组态是基于功能块的应用组态程序，通过从需要的功能块，将这些功能块按照控制要求联系在一起，组成控制回路（功能块应用进程），设置各个

功能块的具体参数值，编译形成代码信息并且确定功能块的执行顺序，并下载组态代码，运行现场控制回路。

打开一个控制策略编辑窗口就可以对功能块控制回路进行组态。从工程管理窗口设备节点中选择特定功能块，通过双击鼠标左键便可将功能块添加到当前策略编辑窗口中。每个功能块都可重复使用，用户可先选择网络中的一个位置，也就是点选将要放置功能块的位置，然后可将功能块添加到应用回路中。

控制策略编辑窗口中的功能块有两种状态：激活状态、无效状态。在功能块上点击鼠标，便可选择功能块，此时有一红色外框在其外部，此时可进行拖拽。按删除可删除此功能块。

4.4.2.1 功能块的图形表示

图 4.6 是一个功能块的图形表示。一个典型的功能块从图形结构上可以分为两个部分：功能块参数部、功能块体部。功能块体部包含功能块名称，功能块参数部包含功能块的输入和输出参数单元[35]。

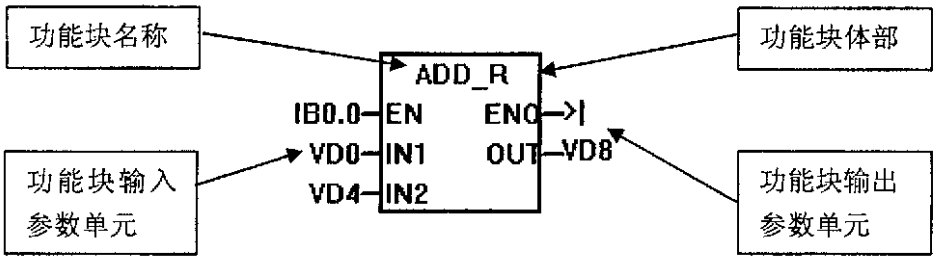

图 4.6 功能块的图形

Fig.4.6 The figure of function block

功能块的软件模型是由 CFBClass 类和 CFBParamClass 类构成的。类 CFBClass 用来描述功能块的图形结构，类 CFBParamClass 用来描述功能块的参数单元。类 CFBClass 中有两个成员变量：

CTypedPtrList<CObList, CFBParamClass*> m_FBInParam;

CTypedPtrList<CObList, CFBParamClass*> m_FBOutParam;

这两个成员变量是两个链表，分别对应功能块输入参数链表、功能块输出参数链表。功能块的大小主要由功能块的体部大小决定。功能块体部的大小取决于该功能块的输入输

出参数的个数及功能块参数单元本身的大小。CFBClass 类根据用户需要使用功能块的类型获取这些信息。功能块参数单元类 CFBParamClass 主要包含单元格的位置和大小、参数名称、参数 ID 等信息。CFBClass 是一个通用类，它适用于任何功能块，它可以根据用户所需要的功能块，画出功能块及功能块参数单元格的图形，自动调节功能块的大小。

4.4.2.2 功能块参数配置

功能块参数配置模块：该模块主要包含如下几部分功能，用户可双击参数区域从而直接配置功能块参数，也可以在参数区域上点右键，选择一个现存的符号表参数，将其配置到功能块参数中。用户双击参数区域时，系统会动态生成一个 CEdit 控件，然后根据参数所在的位置来初始化 CEdit 控件的位置，并调用 CEdit 控件的 Create 函数[36]，并将输入焦点设置于 Edit 控件上。实现了对参数的编辑，其实现如下：

```
m_pEdit = new CEdit;//动态创建一个编辑框控件
CPoint ptLTParam;//编辑框位置坐标
ptLTParam=m_pFBParamClass->m_ptLTParam;
ptLTParam.x=ptLTParam.x-ptScrollViewOffset.x;
ptLTParam.y=ptLTParam.y-ptScrollViewOffset.y;
CRect rcEdit(ptLTParam,m_pFBParamClass->m_szParam);//建立编辑框矩形
m_pEdit->Create(WS_THICKFRAME|WS_CHILD|WS_VISIBLE,rcEdit,this,NULL);//创建
m_pEdit->SetWindowText(m_pFBParamClass->m_strParamVarName);//设置编辑框文本
m_pEdit->SetSel(0,-1);//设置为选择所有文本
m_pEdit->SetFocus();//设置输入焦点
```

右键点击参数区域时，将要求用户选择一个现存的符号表参数，也就是用户定义的符号表参数，可选择其中的项目并将其配置到功能块参数中，如图 4.7 所示。

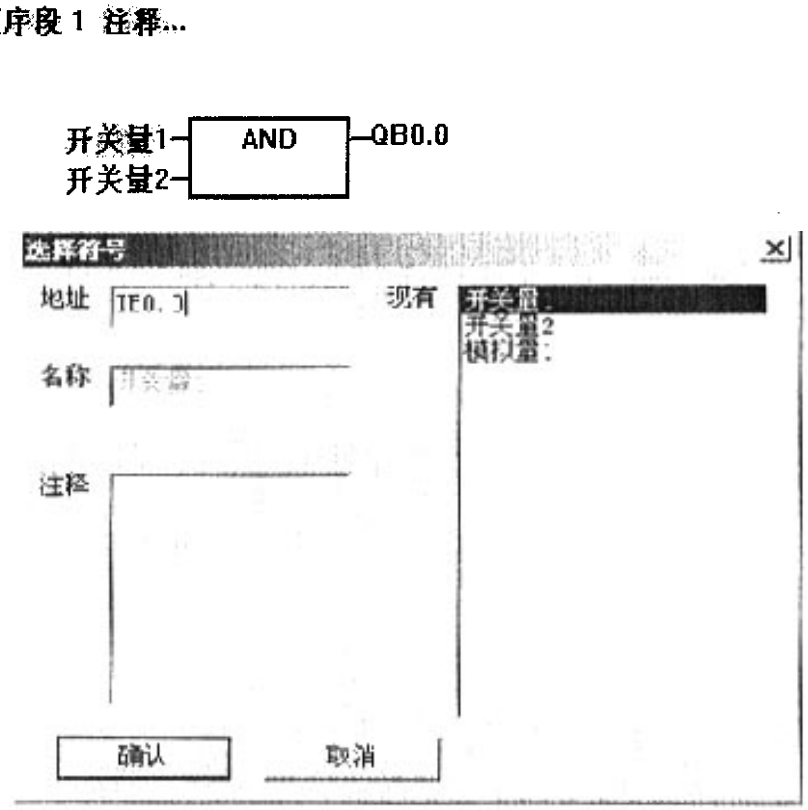

图 4.7 功能块参数配置

Fig4.7 Parameter configuration of function block

4.4.2.3 功能块的链接关系

将各个功能块添加到控制策略编辑窗口后，需要建立功能块之间的连接关系。功能块之间的连接关系是通过建立链接对象来实现的，如图 4.8 所示。链接对象确定功能块参数和参数之间的连接关系。定位一个功能块参数需要三部分信息：参数的数据类型、所在功能块实例、参数 ID，一个链接对象包含两部分信息----源功能块参数地址信息和目标功能块参数地址信息。

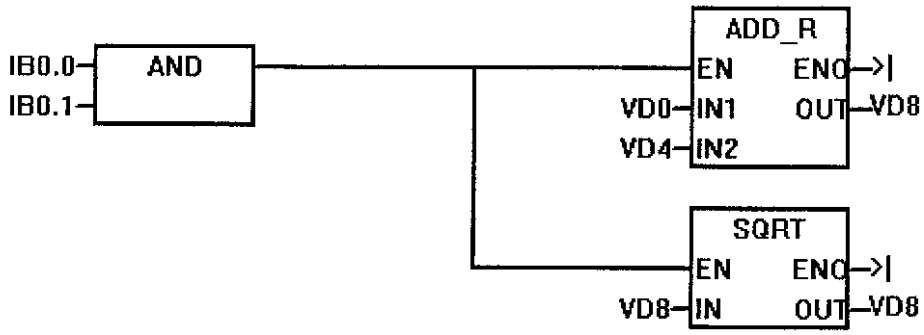

图 4.8　链接对象与应用进程实例

Fig.4.8 The figure of linkage object and application instance

在控制策略组态窗口中，用鼠标左键双击源功能块参数单元格，再用鼠标左键双击目标功能块参数单元格，系统就会自动创建一个连接这两个功能块参数的链接对象。在本程序的设计中，链接对象由类 CLinkageObj 来描述。该类除包含链接对象的基本信息外，还包含链接对象的图形信息。程序如下：

```
class CLinkageObj : public CCmdTarget
{
public:
    CCmdTarget *m_pSourceFB,* m_pDestFB;//链接关系所连接的功能块
    CCmdTarget * m_pSourceParam,* m_pDestParam;//链接关系所连接的功能块参数
    CPoint m_point[4];//链接线的位置
};
```

图 4.9 是一个功能块应用进程实例，通过链接对象将图中的几个功能块组成一个控制回路。由该图可发现，对于功能块中的任意一对输入输出参数，都可以通过三条线段将其连接在一起，其中两条为平行线段，一条为垂直线段。因此链接对象的图形描述可表示成三条连续的线段。进一步分析可发现，由于只能是输入和输出参数之间进行连接，故链接对象有两种基本线型，一种是处于两个功能块之间的线型，是直线。另外一种是三折线。

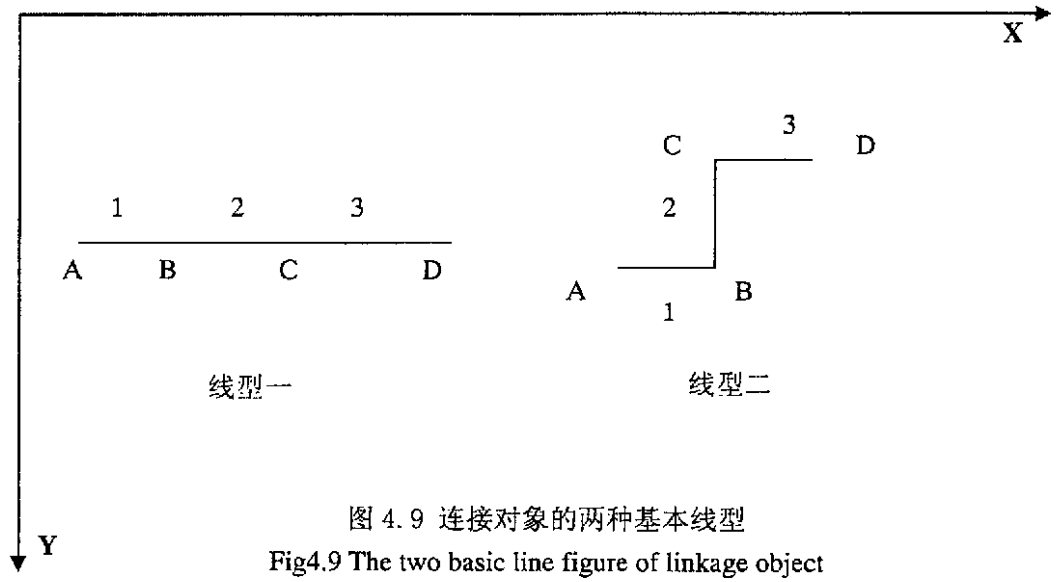

图 4.9 连接对象的两种基本线型

Fig4.9 The two basic line figure of linkage object

如图 4.11 所示，以线型二为例，链接对象有三条线段组成，可通过 A、B、C、D 四个点来确定，假定这四个点的坐标依次存放于 CPoint m_point[4]数组中，A 点的坐标为 out_x、out_y，D 点的坐标为 in_x、in_y，中间的竖线取两个端点的中间位置。由于在程序中采用 MM_TEXT 的坐标映射方式[36]，即水平增量方向为从左到右，垂直增量方向为从上到下，所以有如下的坐标关系：

```
m_point[0].x=out_x;
m_point[0].y=out_y;
m_point[1].x=out_x+(in_x-out_x)/2;
m_point[1].y=out_y;
m_point[2].x= out_x+(in_x-out_x)/2;
m_point[2].y=in_y;
m_point[3].x=in_x;
m_point[3].y=in_y;
```

由以上的等式可知只需要知道 out_x、out_y、in_x、in_y、的值就可以完全确定四个点的坐标，从而确定链接对象的图形表示。out_x、out_y、in_x、in_y 的值取决于功能块的位置，因此当功能块的位置发生变化的时候应同时计算出这四个值。这样就实现了线型一的可视化操作，同理亦可实现线型二的可视化操作。

4.4.2.4 功能块的执行顺序

功能块应用程序是由多个功能块实例组成，为了实现正确的控制，这些块必须以正确的顺序执行，即每个功能块的执行都必须遵循某种调度策略。本程序设计中考虑到使用者的先验经验，主要采用按 PLC 的一般执行顺序完成。也就是功能块的执行按照从左往右，从上到下的顺序依次完成。功能块的执行顺序是在编译时确定的，编译时首先将同一网络内的功能块形成一个树的数据结构。根据功能块所在的位置，确定功能块在树中的位置，最后按照一定的方法遍历树，遍历树的次序就是树的节点的执行顺序。如图 4.10 所示：

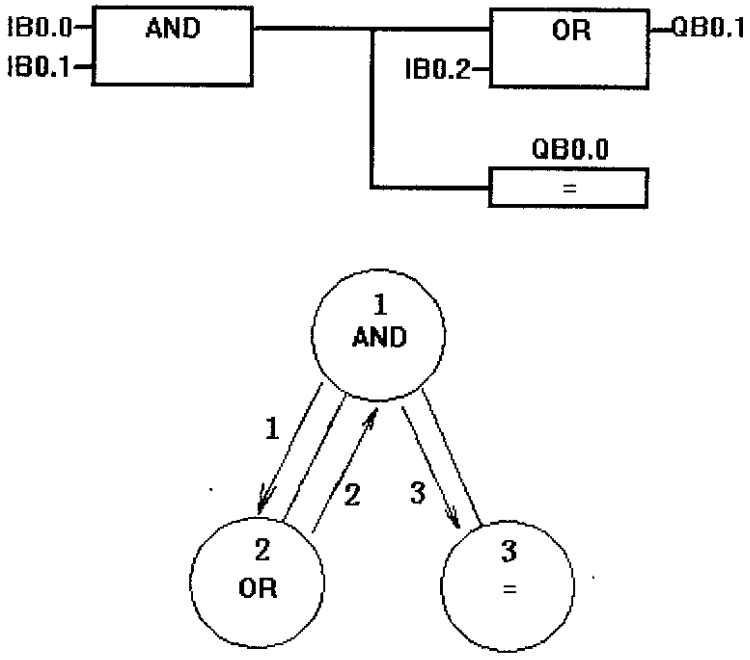

图 4.10 功能块的执行顺序

Fig4.10 The run order of function block

4.4.2.5 组态信息的编译和下载

控制回路组态完毕之后，需要对组态信息进行整理，转换成要下载的信息格式，这个过程称为组态信息的编译。在本程序设计中，采用 CFBSourceCode 类来有效的组织组态后的代码信息。组态程序为每个功能块创建一个 CFBSourceCode 类，并将功能块的信息存储于其中。一个控制回路所包含的所有代码信息以 CFBSourceCode 链表的方式进行组织管理。CFBSourceCode 的定义如下，其内部随功能块参数的不同而变化：

```
struct CFBSourceCode
{
    WORD    Opcode;//操作码
    WORD    OpcodeEx;//操作码扩展
    WORD    NetworkID;//网络 ID
    WORD    FBRunOrder;//执行顺序
    NetworkPosition Position;//功能块位置
    Oprand OprandEN;//输入使能
    Oprand OprandIN1;//操作数 1
    Oprand OprandIN2;//操作数 2
    ……
    Oprand OprandINn;//操作数 n
    Oprand OprandENO;//输出使能
    Oprand OprandOUT1;//运算结果 1
    Oprand OprandOUT2;//运算结果 2
    ……
};
```

组态程序根据各功能块编译的结果，填写各个功能块的 CFBSourceCode 内容，实现从组态信息到下载信息的编译过程。编译成功后，调用通信服务下载组态信息。在下载的过程中，如遇到错误将报告下载错误信息，否则将报告组态信息下载成功。

4.5 通信服务模块

通信服务模块主要包括实时通信服务和套接字映射接口对象的实现。

实时通信服务模块中的各项通信服务以全局对象的形式，由组态程序的其他模块所调用。当组态程序用户层需要与网络通信时，调用相应服务的请求原语（Requset）将请求报文交给实时通信服务，而通信服务在收到通信请求后，将调用套接字映射接口对象的数据发送接口将请求报文发送到网络上。当套接字映射接口对象通过数据接收接口接收到来自网络的通信请求后，将调用相应服务的指示原语（Indicate）把请求转交给相应通信服务，通信服务收到通信请求后把该请求转交给用户层。如果该请求需要回应，用户层会调用相应通信服务的响应原语（Response）把回复报文转交给通信服务，通信服务通过调用套接字接口对象的数据发送接口把回复报文发送到网络上。当套接字映射接口对象通过数据接收接口接收到来自网络的通信回复后，将调用相应服务的确定原语

（Confirm）把回复报文交给通信服务，然后再由通信服务把回复报文转交给用户层。图 4.11 描述了的通信过程。

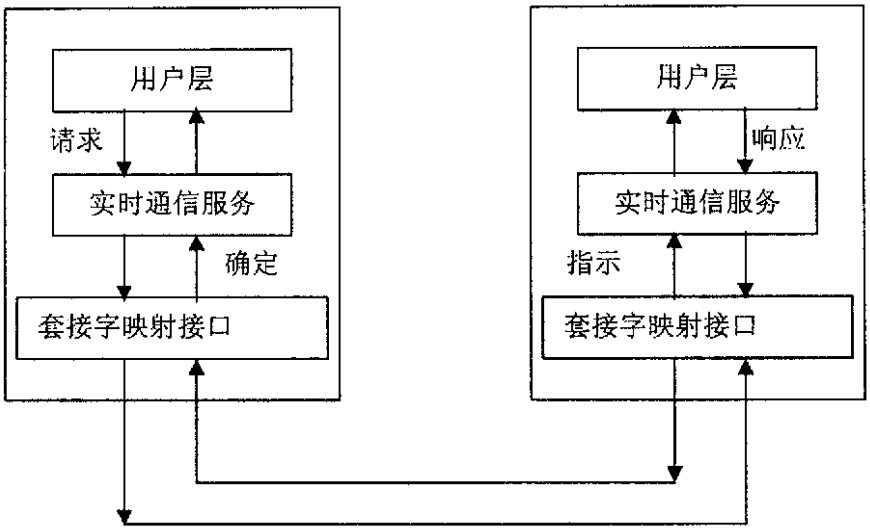

图 4.11 通信过程

Fig.4.11 The communication process

4.5.1 实时通讯服务的实现

实时通信服务分为应用层实时通信服务和管理服务两类。

实时通信服务提供四种服务原语：请求、指示、响应、确定。服务原语是通信服务提供给外部的调用接口，同时服务原语也表示一种执行方向。在本程序中，将通信服务设计成对象的模式。每个通信服务在程序中表现为一个全局对象，称为通信服务对象。每一个通信服务对象都具有以下几个参数：服务序号 ID（m_MessageID）、服务使用标志（m_beUsed）、服务请求完成标志（m_reqFinished）。服务序号 ID 参数表示本服务被调用的次数，该参数初始值为 0，服务每调用一次，该值加 1。服务使用标志参数用于表示通信服务对象是否正在被调用，通常情况下，在同一时刻，通信服务对象只能响应一个服务请求，即只有本次服务调用结束通信服务对象才能响应下一个服务请求。服务请求完成标志表示本次服务请求是否结束，对于确定性服务，其服务请求需要得到响应，它有一定的等待时间，如果服务请求超过等待时间仍没有收到响应，则认为本次请求失败。

4.5.2 套接字映射接口对象的实现

4.5.2.1 套接字映射接口对象

套接字映射接口对象（PLCSocket）封装了各个网络端口，如管理功能端口、应用层服务端口。套接字映射接口提供了实时通信服务与 TCP（UDP）/IP 之间的映射。在网络层次结构上，套接字映射接口介于应用层和传输层之间，向上它提供面向所有实时通讯服务的调用接口，向下它提供向网络发送数据的调用接口。

4.5.2.2 套接字映射接口对象的报文发送管理

组态程序在调用管理功能块服务和应用层服务发送数据时，需要将数据传送给套接字映射接口对象。套接字映射接口对象首先按发送优先级，将这些待发送的数据分别缓存在不同的队列中，以等待发送，优先级最高的报文最先发送。在系统中，定义了三级优先级：设备间周期性信息发布具有最高的优先级，以确保控制系统的连接正常运行。对于事件信息、设备信息等的广播发布具有次高优先级，而点对点之间的单播通信的优先级最低。

来自应用层服务的请求报文，有些是需要确认的，有些是不需要确认的。对于不需要确认的应用层服务请求报文，套接字映射接口对象只需将该报文进行打包，并发送出去即可。

对于需要确认的应用层服务请求报文，套接字映射接口对象在向网络上发送该报文时，将根据发送该消息的服务 ID（ServiceID）以及报文序号（MessageID），创建一个定时器对象，并开始启动定时，并作以下处理：如果在当前报文响应时间 ActiveMsgTime（即最大响应等待时间）到之前收到正确的响应报文，则由套接字映射接口对象将该响应报文发送到相应的服务接口，并同时将定时器清零，并删除该定时器对象；如果在当前报文响应时间 ActiveMsgTime（即最大响应等待时间）到之前收到错误的响应报文，则由套接字映射接口对象将该响应报文发送到相应的服务接口，并同时将定时器清零，并删除该定时器对象；服务将不对该报文进行处理，并直接通知用户功能块实例，由用户功能块实例作出判断并处理。如果定时时间超过当前报文响应时间 ActiveMsgTime(即最大响应等待时间），套接字映射对象仍未收到相应于该请求报文的响应报文，则向应用层服务返回一个负响应，及超时响应错误类型，同时删除该定时器对象。

4.5.2.3 套接字映射接口对象的报文接收管理

组态程序启动之后，套接字接口对象就打开管理功能块端口、应用层服务端口，监听来自现场网络的所有消息。当接收到消息时，由套接字映射接口对象对消息头进行解包，并根据消息所接收的端口的不同，将消息分送到不同的功能模块处理。

套接字映射接口对象接收到来自管理功能块端口或应用层服务端口的消息后，根据服务标识（ServiceID）进行判断与处理，如服务标识无效，则将报文丢弃，不作处理；如果服务标识 ID 有效，则根据其服务类型，判断是否作出应答或响应：如果收到的消息是来自其他现场设备的请求消息，则根据其服务类型，判断是否作出应答或响应；如果收到的消息是组态程序发送出去的请求消息的响应，则根据消息序号（MessageID）、以及肯定或否定的响应，由用户层作出判断与处理。

4.6 信息报告模块

信息报告模块将系统收集到各类信息及时通知给用户，使用户能清楚地得知目前的操作状态。图 4.12 是信息报告窗口的一个示意图。 信息报告窗口分为四个部分：系统信息（System Info）、编译信息（Compile）、操作信息（Operator Info）及通信信息（Communication）。

系统信息：主要记录各类设备管理信息。

编译信息：记录控制回路组态信息的编译信息，报告编译错误等。

操作信息：记录用户操作信息。

通信信息：记录包括功能块参数读写、控制回路组态信息下载等涉及通信服务中应用层管理服务的信息。

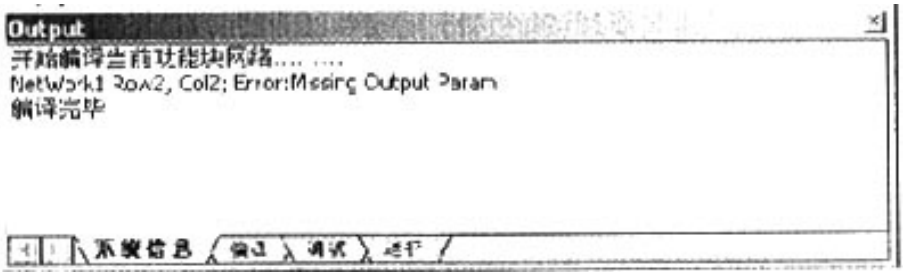

图 4.12 信息报告窗口

Fig.4.12 Information report window

4.7 小结

PLC_Config 是基于功能块的分布式控制系统的可视化组态程序。它通过创建链接对象，建立现场设备中各个功能块之间连接关系，构建成一个个控制回路，将组态信息下载现场设备中，完成对控制网络的组态工作。整个组态过程以一种类似于搭建积木的方式，通过鼠标拖拽、点击完成，而不需要复杂的套接字编程，使得建立功能强大的应用变得非常容易，大大降低了对工程师技术水平的要求，使用和操作起来更加简单、方便，缩短了整个工程的实施周期，降低了生产成本。

5 可编程控制器现场应用

本章以一个水泥混合配料自动控制工程为背景，基于符合 IEC61131-3 标准的 DUT7000 可编程控制器的编程，同时利用 DUT6000 进行了模拟量采集和 I/O 扩展的控制系统的设计思路、工作流程、性能参数，以及配合生产操作的上位机监控系统设计思想以及实现方法等。

5.1 应用背景

近年来，我国建材工业迅猛发展，水泥在建材生产中起着重要作用，水泥配料的比例是影响水泥质量和性能的关键因素，因此对水泥配料的生产控制有着重要意义。针对水泥配料控制过程特点采用 DUT7000 设计了水泥混合配料自动控制系统，应用于大连金德利建材有限公司水泥配料的实际生产中。系统硬件主要由一个 DUT7000 和两个 DUT6000 组成。DUT7000 是系统的主控制器，使用功能块 FB（Function Block）图形化开发语言，DUT6000 是其 AI、DI 和 DO 扩展。DUT7000 主控器与 DUT6000 从设备的通信采用 RS485。利用功能块编程语言，完成了三个流程：称料斗、搅拌机和空压机的程序设计。应用于大连金得利建材有限公司的水泥混合配料控制系统稳定运行在生产现场。本文主要以一个水泥混合配料自动控制工程为背景，介绍基于 IEC61131-3 标准的 DUT7000 软 PLC 控制器的编程，同时利用 DUT6000 进行了模拟量采集和 I/O 扩展的控制系统的设计思路、工作流程、性能参数，以及配合生产操作的上位机监控系统设计思想、实现方法等。在设计过程解决了以下一些问题：

1、根据大连金得利建材有限公司的水泥混合配料控制系统的基本功能要求，设计设备配置：分别由三个机柜与操作台等组成；
2、设计 6 个电机的软启动过程，并由 DUT7000 进行控制启动过程；
3、进行模拟量采集并进行处理，通过传感器、DUT6000 和 DUT7000 结合完成数据处理，从而上料过程进行自动控制操作；
4、根据生产要求，设计了生产流程，并利用 DUT7000 完成自动控制；

5.2 应用总体结构

大连金得利建材有限公司的水泥混合配料控制系统，要完成两种配料从料仓中通过上料电机添加到称料斗中，其中要求按照一定的配比、重量完成上料过程；然后称重记

录后将混和配料放料到搅拌机中，进行定时搅拌；最后打开阀门下料到运输车中，实现整体的控制过程。整个控制过程要完成 6 个电机的软启动过程，2 个阀门的打开、关闭控制，搅拌机的定时搅拌，上料过程中称料斗重量的采集，时时监控控制系统状态等工作。在工作模式方面，支持全自动、半自动和手动控制，充分考虑了工业生产的实际情况。

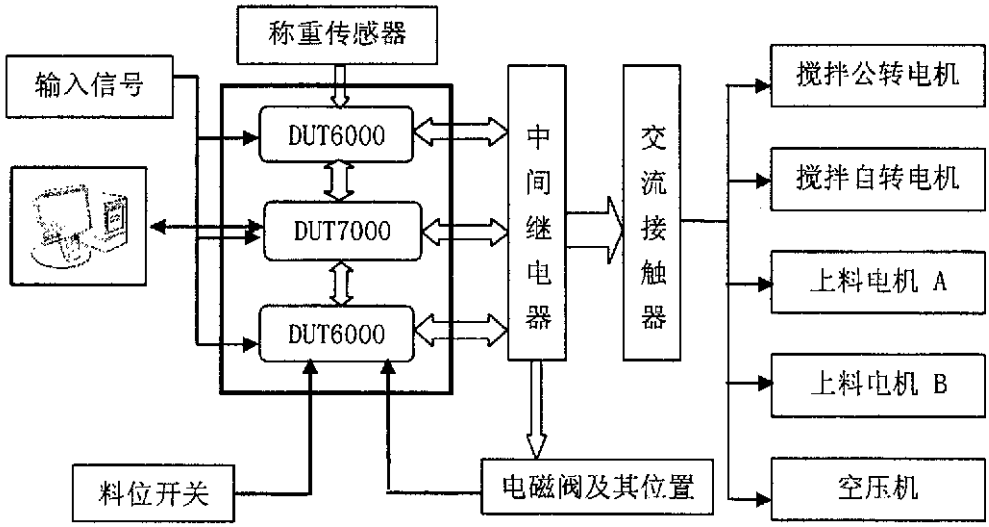

图 5.1 系统结构图

Fig. 5.1 System tropy

5.2.1 硬件设备设计

系统由强电控制柜、主控制柜、现场分布控制箱、人机操作终端、现场操作终端、执行器和传感器组成。主控制柜主要安装了 DUT7000、一个 DUT6000 和全部继电器；强电控制柜安装了全部的交流接触器（空气开关）；现场分布控制箱安装了另一个 DUT6000 进行称重信号的采集和 I/O 的扩展，如图 5.1。强弱电分开，增加操作的安全性，也减少了干扰。另外设计了一个现场操作终端（控制台）安装人机操作终端、信号灯和按钮。工程中使用了两个软启动器，分别启动空压机、两个螺旋输送电机、搅拌自转电机、搅拌公转电机和出料电机。

工程设备是分布在 4 层楼中，一楼供给运输车出入，二楼是主控制室，三楼放置搅拌机，四楼安装了称料斗和两个上料电机。

在控制室内，安装了三个控制柜，由两个强电控制柜和一个主控制柜组成。

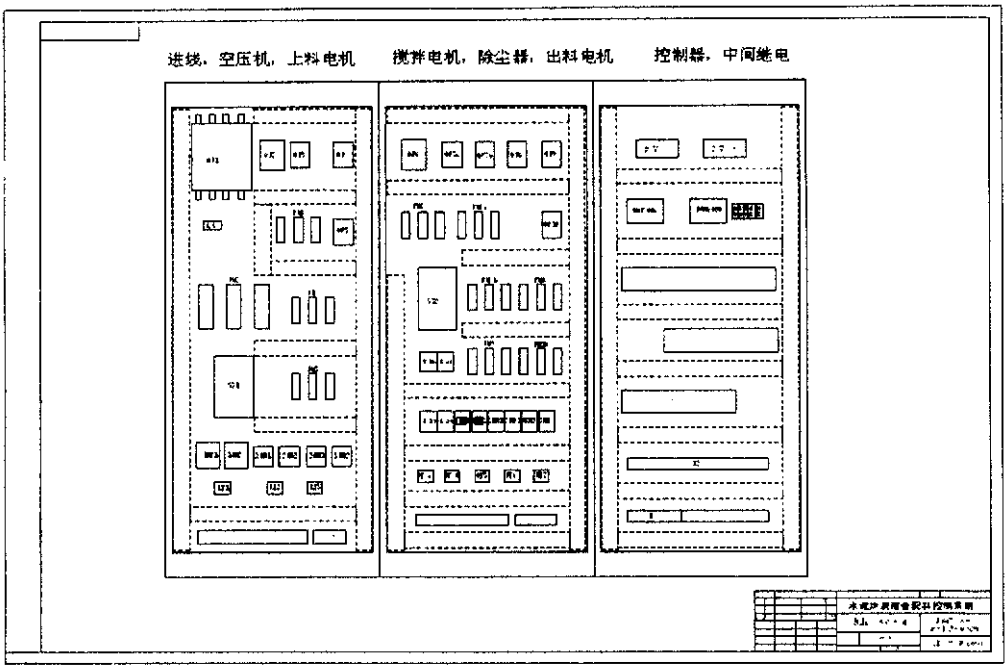

图 5.2 控制柜

Fig. 5.2 Control cabinet

如图 5.2，第一个机柜为强电控制柜，主要安装了 380V 电源、主开关、空压机开关和两个上料电机的开关和控制以上 3 个电机的软启动器。第二个机柜是强电控制柜，安装了 380V 电源、搅拌电机（公转、自转）的开关、除尘电机开关、出料电机的开关以及软启动器。第三个为主控制柜，主要安装了 220V 电源、DUT7000、DUT6000-1 和全部的中间继电器，主控制柜再与操作台相连，同时连接到四楼的称料斗旁的分布控制箱，组成了整个控制系统的全部设备。

现场还设置了一个操作台，以供操作员进行操作和现场监控使用。操作台按钮包括：（1）急停按钮；（2）称料斗模式选择旋钮（3）称料斗执行按钮；（4）A 上料；（5）B 上料；（6）蝶阀开；（7）蝶阀关；（8）搅拌；（9）停止搅拌；（10）搅拌机模式选择旋钮；（11）搅拌机执行按钮；（12）闸阀开；（13）闸阀开；（14）空压机开关旋钮；（15）出料方向旋钮；（16）报警确认按钮；（17）出料结束按钮；（18）除尘器开关旋钮。现场操作台指示灯包括：（1）报警灯（内置蜂鸣器）；（2）24V 电压显示灯。如图 5.3 所示：

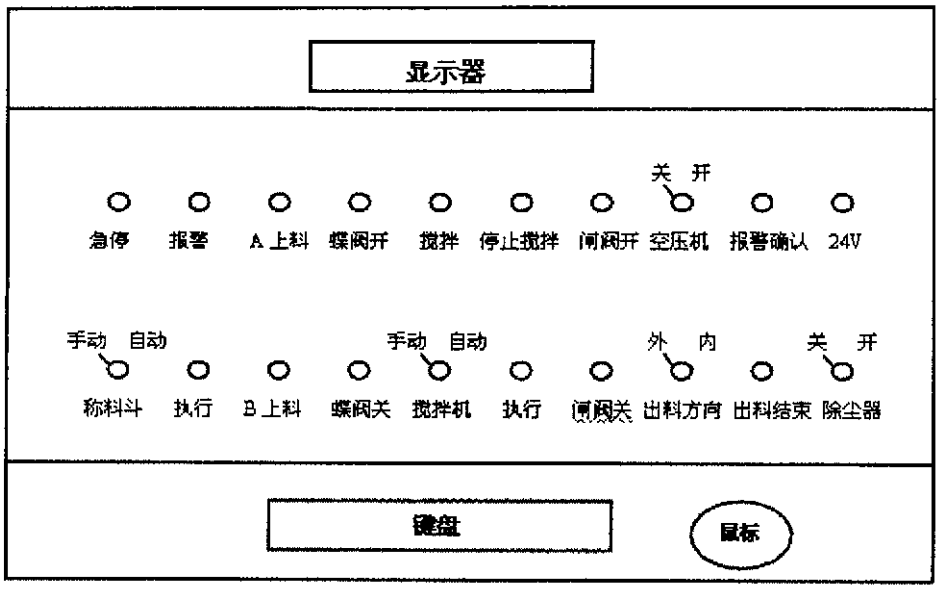

图 5.3 控制台

Fig. 5.3 Control panel

5.2.2 PLC 资源分配

工程中使用了一个 DUT7000 和两个 DUT6000，其资源分配如下：

（1）DUT6000-1：位于四楼称料斗旁，控制箱内。直接连接称重传感器 3 个，称料斗阀门位置信号 2 个，阀门开闭控制信号 2 个。其他开入信号（开入 2，模拟改开入 5）引到控制柜内转接，开出使用 2 个，转接信号中开入 2 用于搅拌机阀门位置检测，其他备用。

（2）DUT6000-2 和 DUT7000：在二楼主控制柜内，操作按钮连接 DUT7000 的开入 8，电机过流连接 DUT6000-2 的模拟改开入。

5.3 PLC 程序设计

根据系统工作流程，将整体流程分为三个子功能，分别是空压机子程序、称料斗子程序和搅拌机子程序，如图 5.4 所示。空压机子功能负责空压机的启动、停止（分时分步骤进行软启动）。称料斗子功能负责按照上位机的设定值完成两种料的上料；对上料重量进行采集计算；完成下料到搅拌机过程。搅拌机子功能负责将按照配比上完的料在搅拌机中进行搅拌，达到均匀后放料到运输车中。

为了方便用户的操作，设计了全自动、半自动和手动三种工作模式。三个子程序控制了电机的软启动过程，上料过程，定时起停电机和数据处理功能。在程序中还加入了急停和报警功能，这也是在工业生产中必不可少的功能。

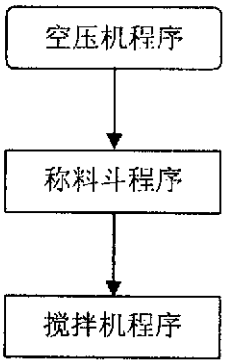

图 5.4 系统流程图

Fig. 5.4 System flow chart

程序中首先启动空压机，因为在称料斗和搅拌机程序中，蝶阀和闸阀的开关是靠空气开关控制的，如果没有启动空压机，阀门的开关会出现问题，也会造成生产事故。因此在程序中，没有启动空压机时，称料斗和搅拌机则不能启动。启动空压机后，进入称料斗流程，等待称料斗按照配比完成上料后，自动进入搅拌机流程，进行混合配料的搅拌。最终放料，完成一次生产流程。如图 5.5 所示。

下面以启动空压电机为例，首先启动空压机的软启动继电器；1s 后启动软启动器，回路接通，空压机以启动电压 Us 开始运行，电机转速会随着软启动器的输出电压的增加而加快，直至电压达到电压 Ur；软启动器返回一个结束信号，这时接通空压机旁路继电器；1s 后将空压机软启动继电器和软启动器释放；软启动过程结束。

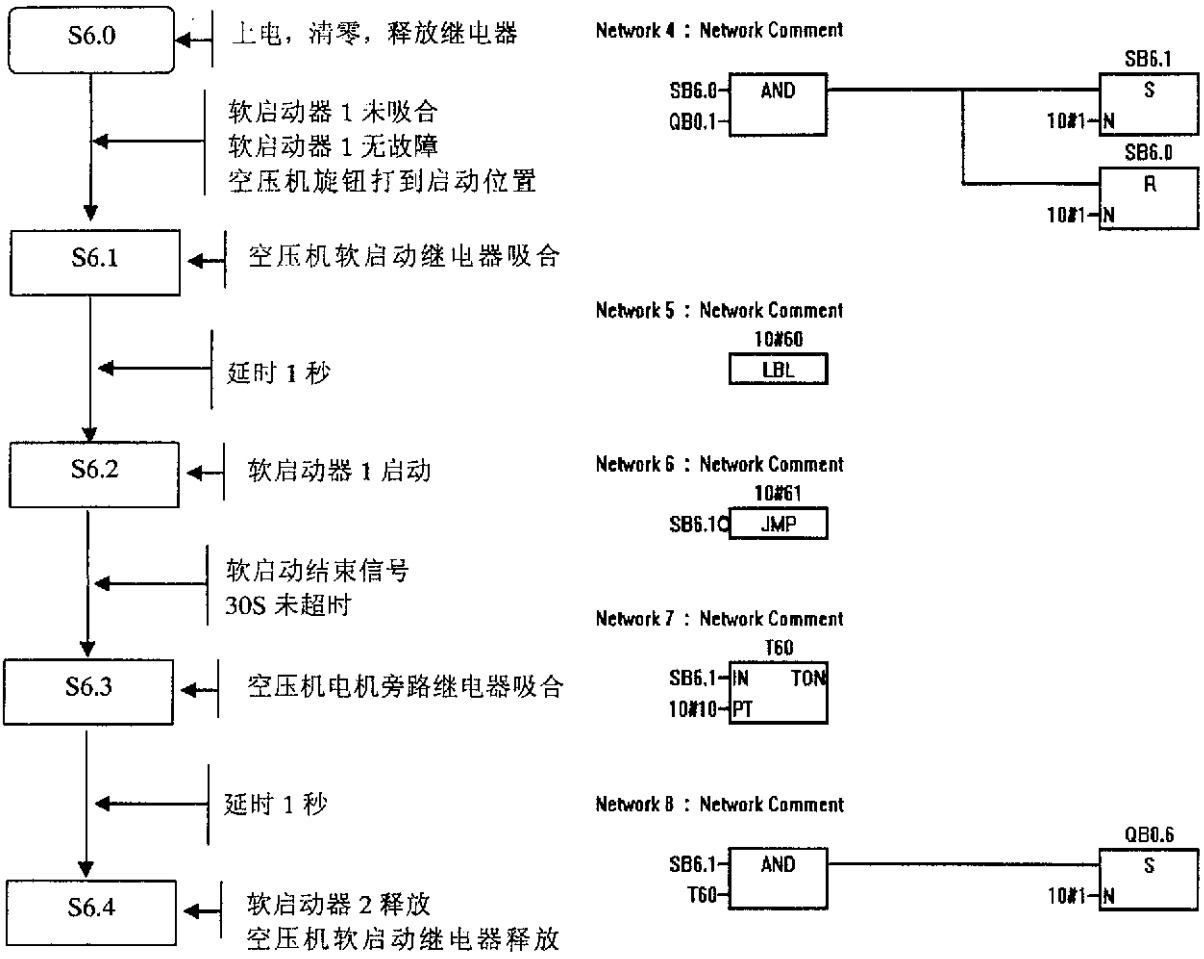

图 5.5 空气压缩机启动流程图

Fig. 5.5 Flow chart of air compressor starting

5.4 小结

本章以一个工程应用为中心，介绍了 DUT7000 可编程控制器的软硬件设计思想，给出了 I/O 端口配置，程序整体流程以及程序实现。对于程序的核心部分，比如空压机软启动的处理方法，做了详细的描述和介绍。从 2005 年 6 月至今，应用于大连金得利建材有限公司的水泥混合配料控制系统稳定运行在生产现场。

结论

本课题的主要工作是研究 IEC61131 标准的内容并在此基础上开发基于 IEC61131-3 标准的可编程控制器，目前已基本完成，研究工作主要包括两个部分：上层组态软件 PLC_Config 的设计和实现以及下位机程序的设计和实现。上层代码主要以 Visual C++ 6.0 为开发环境，编写了约四万行左右的代码；下位机可编程控制器的底层程序用 Dynamic C 实现，约一万五千行左右的代码。

本课题所作的研究工作主要有以下几个创新点：

（1）遵循国际上最新制定的 IEC61131-3 功能块标准，实现分散控制；

（2）组态软件基于以太网络，统一了监控层和现场设备层的网络类型；

由于研究的内容涉及面广、工作量大、技术新颖，是一个复杂而艰巨的工程，需要一个团队长期努力，因此，尽管本人在这一年多的时间内尽了很大的努力，但还是有很多地方需要进一步完善。需要在以下方面作进一步的工作：

（1）通过使用更强的处理器提高运算以及处理速度，提高可编程控制器的扫描周期以及数据处理能力；

（2）充分利用 Internet 网络技术。由于组态软件应用于以太网，而基于 Ethernet+TCP/IP 有大量的成熟可靠的网络技术，将这些技术应用到工业以太网是今后工作的一个主要方向。比如，可以用 IE 浏览器作为现场的监控软件；

（3）开发过渡性控件。由于工业以太网技术是一种崭新的技术，将其普遍应用于现场还需要几年甚至更长的时间，因此需要开发一些组件，使组态程序能够兼容目前广泛使用的设备。例如，可开发通信块，支持 RS485 通信。

（4）改进组态编辑窗口的可视化操作，使组态界面更加友好，组态过程更加简单。支持 IEC61131-3 标准的各种语言的互相转换。

当前，世界上各大组织对 IEC61131-3 都处于初期阶段，所以开发具有我国自主知识产权的控制系统具有重大的实际意义。

参考文献

[1]李宇成,刘锡荟.GB/T 17165.3及IEC 61131介绍.辽宁工程技术大学学报,2001,5(11):567-570.
[2]IEC61131-1 general information Ed.2.International Engineering Consortium,2002.
[3]Van der Wal,E..Creating reusable,hardware independent motion control applications via IEC 61131-3 and PLCopen motion control profile. IEEE/ASME International Conference, 2001.Toronto:769-774.
[4]Karl John.IEC 61131-3 programming industrial automation systems.Springer,2003.
[5]刘奋强.DCS与PLC的区别和应用.湖北电力,2005,29(4):107-108.
[6]汤媛.PLC与现场总线控制系统的关系.石油化工建设,2005,27(4):58-63.
[7]黄明铸.PLC DCS FCS三大控制系统的特点和差异.中华纸业,2005,12(4):48-50.
[8]Kajihara,S.,Ono,M.,Houzouji,H.,Taruishi,H..Development and products of the object-oriented engineering tool for the integrated controller based on IEC 61131-3.SICE 2004 Annual Conference,2004.Japan:1952 - 1956.
[9] 北京亚控自动化软件软件科技有限公司.组态王 5.1 使用手册.北京:北京亚控自动化软件软件科技有限公司,2005.
[10]陈忠华.细说IEC 61131-3.自动化博览,2003,5(3):8-14.
[11]IEC61131-3 Programmable controllers Ed.2.International Engineering onsortium,2002.
[12]IEC61131-5 Communications Ed.2.International Engineering Consortium,2002.
[13]孙进生,姜建国.现场总线的发展趋势—工业以太控制网络.河北理工学院学报,2002,24(2):153-157.
[14]周晓兵,费敏锐.以太网在工业自动化领域中的应用现状和发展前景.自动化仪表,2001,22(10):1-4.
[15]西门子公司.SIMATIC S7-200可编程控制器系统手册.德国:SIEMENS AG出版社,2001.
[16]Z-World公司.Rabbit 2000 microprocessor user's manual.美国:Z-World公司,2005.
[17] Z-World公司.Dynamic C User's Manual.美国:Z-World公司,2005.
[18]戴梅萼,史嘉权.微型计算机技术及应用.北京:清华大学出版社,2000.
[19]De Sousa,M.,Carvalho,A..An IEC 61131-3 compiler for the MatPLC.ETFA'03.IEEE Conference,2003.Germany:485-490.
[20]Turnbull,G.Open-ness and IEC 61131-3.IEEE Int'l Conf on Computational Cybernetics and Simulation,1997:4104-4108.
[21]魏庆福.现场总线技术发展的新动向.工业控制计算机,2000,5(1):11-12.
[22]李继容,鲍芳,何湘初.以太网在工业自动化领域的应用及研究.计算机应用研究,2002,11(9):126-128.
[23]宋真君,李京,凌志浩,吴勤勤.基于工业以太网的分布式控制系统的通信研究.厦门大学学报(自然科学版),2001,40(8):269 - 273.

[24]Potter D..Using ethernet for industrial I/O and data acquisition. Instrumentation and Measurement Technology Conference, IMTC/99.Proceedings of the 16th IEEE,1999.United Kingdom:1492-1496.
[25]徐晓东,杨振坤.中小型DCS组态软件的设计与开发.计算机工程与应用,2001,1:25-27.
[26]马国华.监控组态软件及其应用.北京:清华大学出版社,2001.
[27]吕涌,皇甫正贤.组件化结构的组态软件的研究与开发.电气传动自动化,2000,13(5):39-42.
[28]欧金城,欧世乐.组态软件的现状和发展.工业控制计算机,2002,15(4):1-5.
[29]美国国家半导体公司.NI-FBUS configurator user manual.美国:国家半导体出版,1997.
[30]毛卫良,盛焕烨,单德根.一个基于分层模型的组态软件的设计及实现.小型微型计算机系统,2000,21(12):1282-1285.
[31]王荃,金海东,李福中.工控组态软件实时数据库系统的设计与实现.化工自动化及仪表,2000,27(3):40-43.
[32]David J.Kruglinski,Scot Wingo,George Sheperd.Visual C++6.0技术内幕.北京:希望电子出版社,1999.
[33]王华,叶爱亮.Visual C++6.0编程技巧与实例.北京:机械工业出版社,1999.
[34]Bonfe,M.,Fantuzzi,C..Object-oriented approach to PLC software design for a manufacture machinery using IEC 61131-3 norm languages.IEEE/ASME International Conference,2001.Germany:787-792.
[35]Heverhagen,T., Tracht,R.. Integrating UML-RealTime and IEC 61131-3 with function block adapters.Fourth IEEE International Symposium,2001.Japan:395 - 402.
[36]侯俊杰.深入浅出MFC.武汉:华中科技大学出版社,2001.

攻读硕士学位期间发表学术论文情况

[1]郭福帅. 基于 IEC61131-3 标准的嵌入式可编程控制器的研究与实现（属于学位论文第 2, 3, 4 章）. 大连理工大学网络学刊, 已录用.

致　谢

值此论文完成之际，我要深深的感谢我的导师仲崇权教授以及教研室的杨素英副教授。研究生求学三年期间，仲老师以他那广博的学识、严谨的学风、丰富的经验和缜密的思维启发和指导着我在研究过程中一步步的向前进展，从他身上我不仅学到了许多有益的知识与经验，更重要的是学到了仲老师治学严谨，精神乐观的人生态度，这将使我受益终生；杨老师在学习上给予的支持和生活上给予的关怀增强了我在研究过程中克服困难的信心；仲老师和杨老师带领我们共同营造的良好的学术氛围和生活空间，为本论文的完成奠定了学术基础。在大工七年的学习和生活给我留下了美好的回忆，其中在计算机控制教研室的岁月更是令我骄傲。

在课题研究期间，张立勇老师、李卓函老师给了我许多建设性的意见，使我受益匪浅。与袁晓峰老师、张钰和龚志峰的讨论给予我新的启发。感谢白景春、康宁、李学忱等各位师兄对我的支持。正是他们在我论文的攻关阶段给以我无私的帮助才使我能顺利完成工作。

感谢我的家人以及朋友，在我遇到困难的时候，给我鼓励和支持；在我取得进步的时候，给我默默的祝福；在我行学偏差的时候，给我指正和帮助。