LM小型可编程控制器

软件手册

和利时集团

HollySys Group

版权申明

本手册内容，包括文字、图表、标志、标识、商标、产品型号、软件程序和版面设计等，均受《中华人民共和国著作权法》、《中华人民共和国商标法》和《中华人民共和国专利法》及与之适用的国际公约中有关著作权、商标权、专利权或其他财产所有权法律的保护，为北京和利时系统工程股份有限公司专属所有或持有。

本手册仅供商业用户阅读、查询，在未得到北京和利时系统工程股份有限公司特别授权的情况下，无论出于何种原因和目的，均不得用任何电子或机械的方法，以任何形式复制和传递本手册的内容，否则本公司将依法追究法律责任。

我们已经核对本手册中的内容和图表与所述的硬件设备相符，但是误差难以避免，并不能保证完全一致。同时，我们会定期对手册的内容和图表进行检查、修改与维护，恕不另行通知。

本手册的说明、图表、简单程序及应用实例完全出于举例说明的目的，我们对其都进行了测试，但因为软件版本的更新和各种应用有许多未知的变化和要求，我们不承担根据本手册或本手册中的实例而构成的实际应用产生的责任。

北京和利时系统工程股份有限公司保留全部权利。

HOLLiAS、HollySys、和利时、 和利时 HollySys 的字样和徽标均为北京和利时系统工程股份有限公司的商标或注册商标。

Microsoft、Windows 和 WindowsNT 是微软公司在美国和/或其他国家分支机构的商标或注册商标。手册中涉及到的其他商标或注册商标属于他们各自的拥有者。

前　言

LM 小型可编程控制器（PLC）是和利时公司推出的新一代小型一体化 PLC，其中包括多种 CPU 模块和扩展模块。LM 系列 PLC 以其性能稳定、质量可靠、价格适中等优点，广泛应用于自动化领域的众多行业中，也赢得了广大用户的好评。

PowerPro 软件是和利时公司专为 LM 系列 PLC 所开发的基于 Windows 操作系统的编程工具，是 LM 系列 PLC 的硬件配置和软件编程的标准软件包。其主要特点如下：

- 完全符合 IEC61131-3 标准，支持 LD、IL、ST、FBD、SFC、CFC 等多种编程语言；
- 具有 400 余条指令，支持多种数据类型，编程灵活，程序执行效率高；
- 具有丰富的扩展指令，支持用户自定义库，提高程序的复用性和功能扩展能力；
- 强大的数学运算功能，支持浮点数运算，支持多维数组；
- 强大的软件仿真、在线调试及程序检查能力。支持软件仿真，具有单步、单循环、设置断点、强制变量等功能；
- 完善的视图、报警和日志功能，可以通过视图功能实现控制过程的可视化；
- 强大的密码保护功能，可设置 8 个不同等级的密码和权限。

适用范围

本手册的内容适用于 PowerPro 2.1 软件。

如何使用本手册

如果初次使用 PowerPro 软件，需要通读手册。对于有经验的用户，可以通过目录和导读查找相关信息。

相关信息

- 硬件信息请参见《LM 小型可编程控制器硬件手册》。
- 指令信息请参见《LM 小型可编程控制器指令手册》。

目 录

导 读

本软件手册的目的是为了协助您通过PowerPro编程软件设计出一套完善的PLC控制程序，主要介绍了如何使用PowerPro软件和标准编程语言编写控制程序。

第一章　介绍了PowerPro软件的安装、卸载方法以及安装目标。

第二章　对PowerPro做了概述，同时向您详细描述了PowerPro软件编程环境，包括主界面、菜单栏、快捷工具和对象组织器等。若您需要了解PowerPro软件菜单或者快捷方式的选项，可以参考此章的内容。

第三章　快速入门，通过一个简单的例子，向您介绍了PowerPro软件使用的基本步骤和基本使用方法。对于初学者，建议仔细学习此章的内容。

第四章　介绍了LM系列PLC的存储区分配和PowerPro对于变量的管理，包括地址和变量的定义、变量的分类等。假如您在地址或变量使用中有什么问题，可以参考此章的内容。

第五章　主要讲述PowerPro对POU的管理，如：POU的建立、调用等。此章对您在建立程序时碰到的一些问题作了解答。

第六章　在第五章的基础上，此章主要描述了PLC的工作方式以及任务的管理和配置。这部分内容与中断调用相关。

第七章　在了解了PowerPro软件中如何管理地址、变量和POU后，此章主要讲述如何编写程序。并以LD语言为例，介绍了工程的创建和管理，包括新建、PLC配置、程序编写、子程序调用、添加注释等。此章将有助于您完整地编写一个工程。

第八章　编写完程序后的编译、下载和调试等步骤是此章的主要内容。您需要执行的相关操作或遇到的问题都在该章中有所涉及。

第九章　主要介绍了PowerPro软件中的FBD、IL、ST、SFC等编程语言的使用。当您需要使用这些语言，请仔细阅读此章相应内容。

第十章　讲述了PLC的一些特殊功能，包括Modbus通讯、中断、模拟量功能的使用、DP通讯和以太网通讯等。当您用到这些功能时，请查阅此章相应内容。

第十一章　讲述了PowerPro视图的使用。视图是PowerPro软件的高级应用，若您希望在调试时有一个可视化的界面，请参阅此章相应内容。

第1章 安装指南

本章主要介绍 PowerPro 软件的安装和卸载，并对安装目标进行详细的描述。1.1 章节讲述软件的安装过程，1.2 章节讲述软件的卸载。假如您对 Windows 操作很了解，您可以跳过此二节。1.3 章节介绍了如何安装目标。PowerPro 是一款功能强大的 PLC 编程软件，安装目标的目的是将 PowerPro 配置为 LM 系列 PLC 所用的编程软件。若您是第一次接触 PowerPro，希望能仔细阅读此章的内容。

1.1 软件安装

在装有中文 Windows 操作系统的计算机上，将 PowerPro 软件安装光盘插入光驱，在自动弹出的画面界面上选择“LM 系列 PLC 编程软件 PowerPro2.1.3”。或在光盘中找到“\编程软件 PowerPro2.1.3B\Setup.exe”，双击打开此安装程序，出现 PowerPro 安装界面，点击“下一步(N)”按钮，如图 1-1-1 所示。

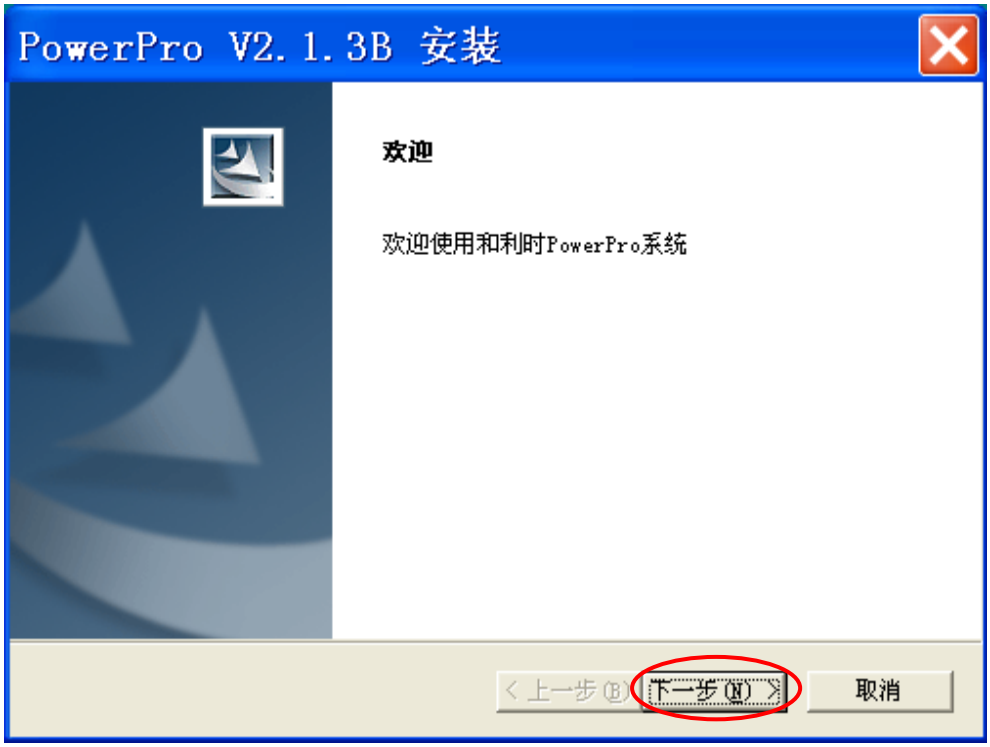


图 1-1-1 安装步骤（1）

同意版权协议，点击“是(Y)”按钮，如图 1-1-2 所示。

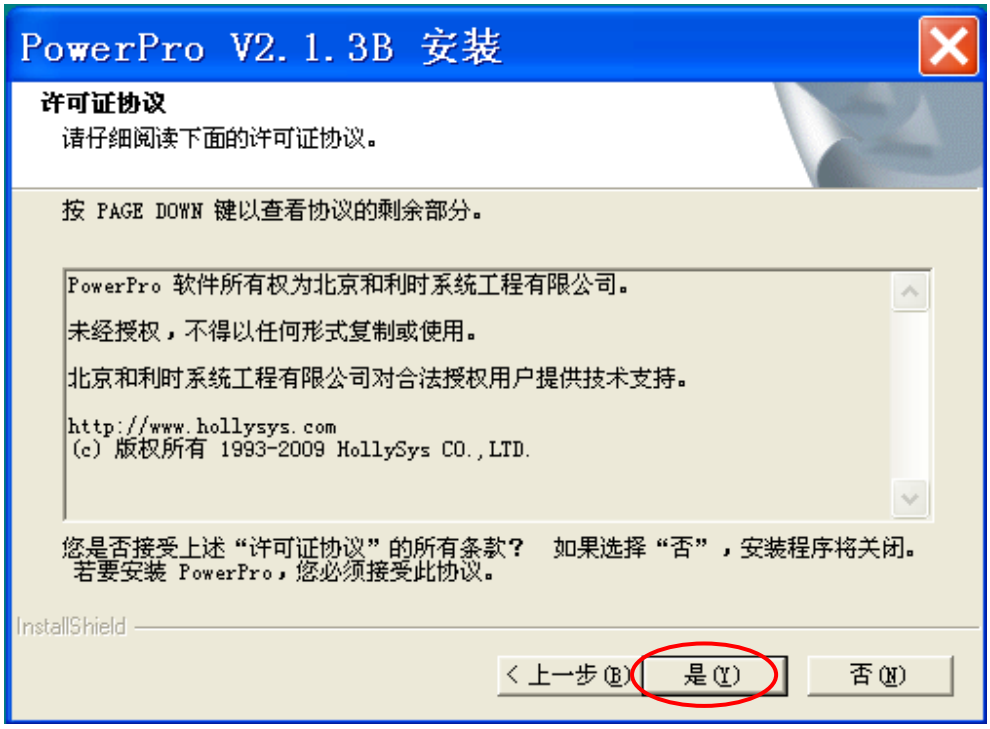


图 1-1-2 安装步骤（2）

选择安装路径。默认的安装路径为 D:\Hollysys\PowerPro，建议不要修改。点击“下一步(N)”按钮开始安装，如图 1-1-3 所示。如果需要修改，可点击“浏览(R)…”按钮选择其它的路径，如图 1-1-4 所示。

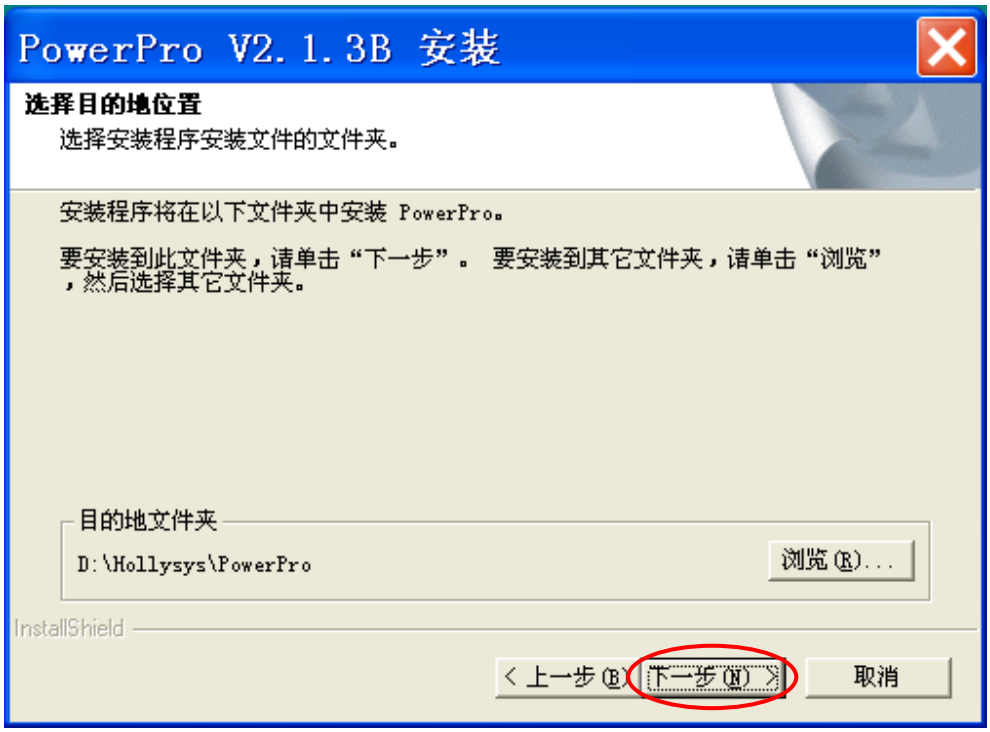


图 1-1-3 安装步骤（3）

图 1-1-4 安装步骤（4）

安装过程中弹出“正在安装”窗口，显示安装进度，若需要取消安装，则鼠标单击“取消”，如图 1-1-5 所示。

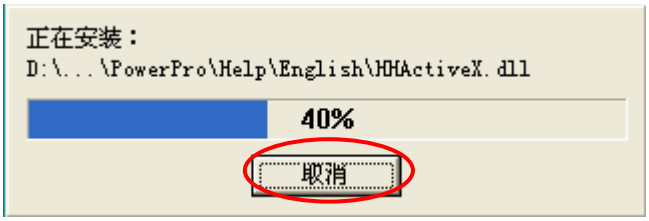


图 1-1-5 安装步骤（5）

安装完毕，弹出“完成”窗口，点击“完成”按钮，PowerPro 软件安装完毕，如图 1-1-6 所示。成功安装的同时，在桌面上弹出“HollySys”窗口，如图 1-1-7 所示。

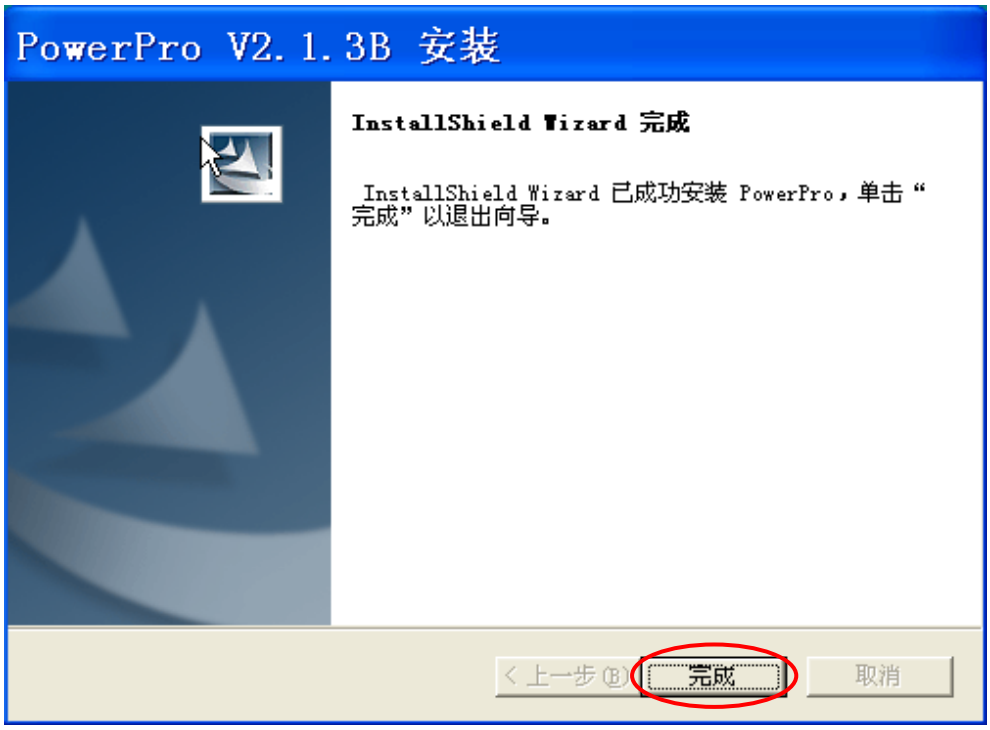


图 1-1-6 安装步骤（6）

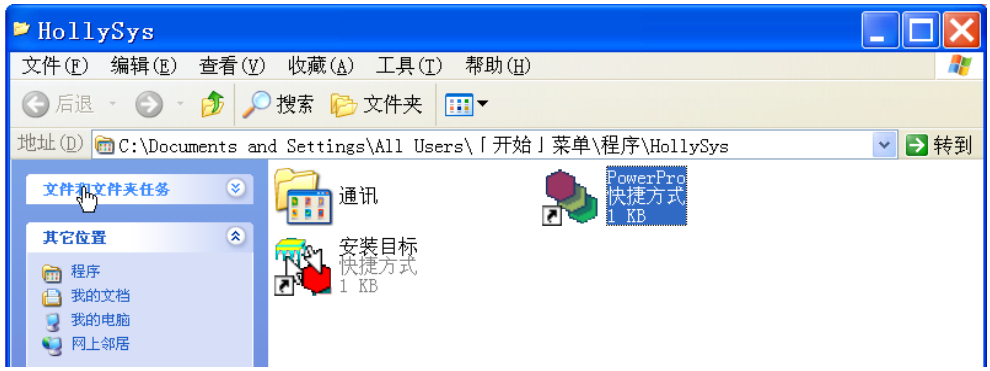


图 1-1-7 安装步骤（7）

1.2 软件卸载

如果计算机中已经安装了低版本的 PowerPro 软件，当安装新版本的 PowerPro 软件时，需要将旧版本的 PowerPro 软件卸载。在“控制面板”/“添加/删除程序”中选择 PowerPro，点击“更改/删除”便可对该程序进行卸载，如图 1-2-1 所示。

图 1-2-1 卸载程序

注意：

在卸载之前，必须要先退出桌面右下角系统托盘中的 Gateway.exe 程序！

1.3 安装目标

PowerPro 软件是 PLC 控制方案的开发平台。在使用 PowerPro 软件之前，必须要先进行“安装目标”，为 PLC 模块选择软件运行的平台。由于安装内容对所有工程通用，因此，使用 PowerPro 软件之前只需要进行一次“安装目标”即可。

“安装目标”的具体步骤如下所述。

在桌面点击“开始”/“所有程序”/“HollySys”/“安装目标”，如图 1-3-1 所示。

图 1-3-1 安装目标（1）

弹出“InstallTarget”窗口，如图 1-3-2 所示。点击“Open...”按钮，在弹出的窗口中选择其中的“C16x_hollysys.tnf”文件，点击“打开”按钮，窗口自动关闭。

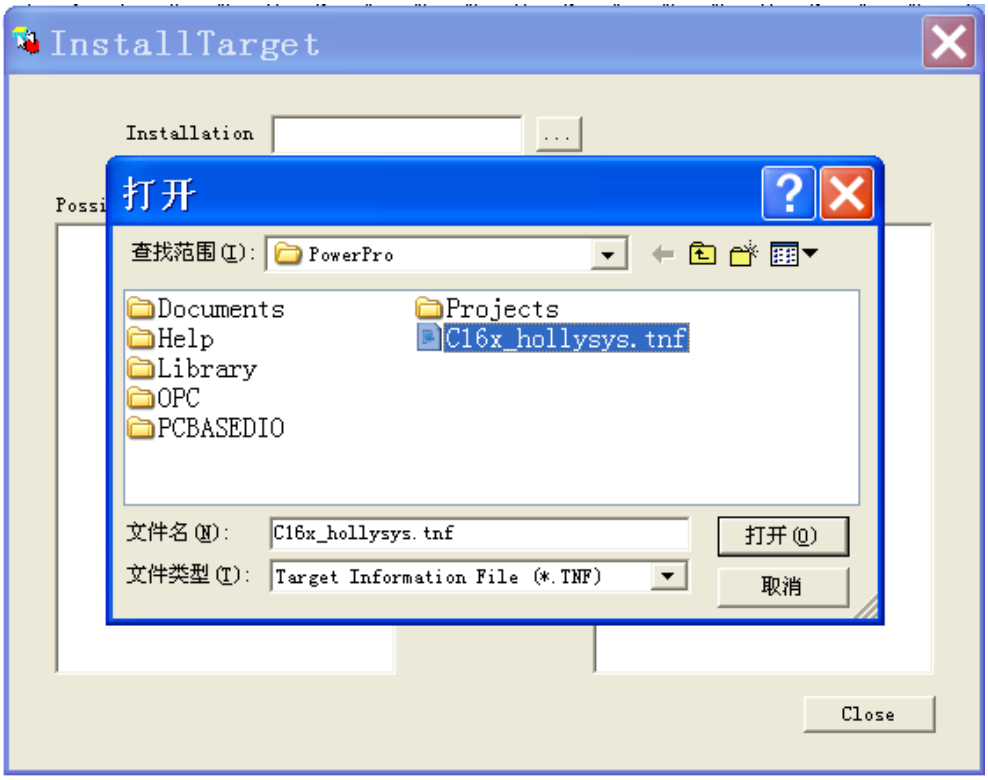


图 1-3-2 安装目标（2）

这时，在“InstallTarget”窗口左侧的可用目标“Possible Targets”窗口中，产生一个目标“Hollysys PLC” 。选中该目标“HollySys PLC”，点击“Install”按钮，如图 1-3-3 所示。

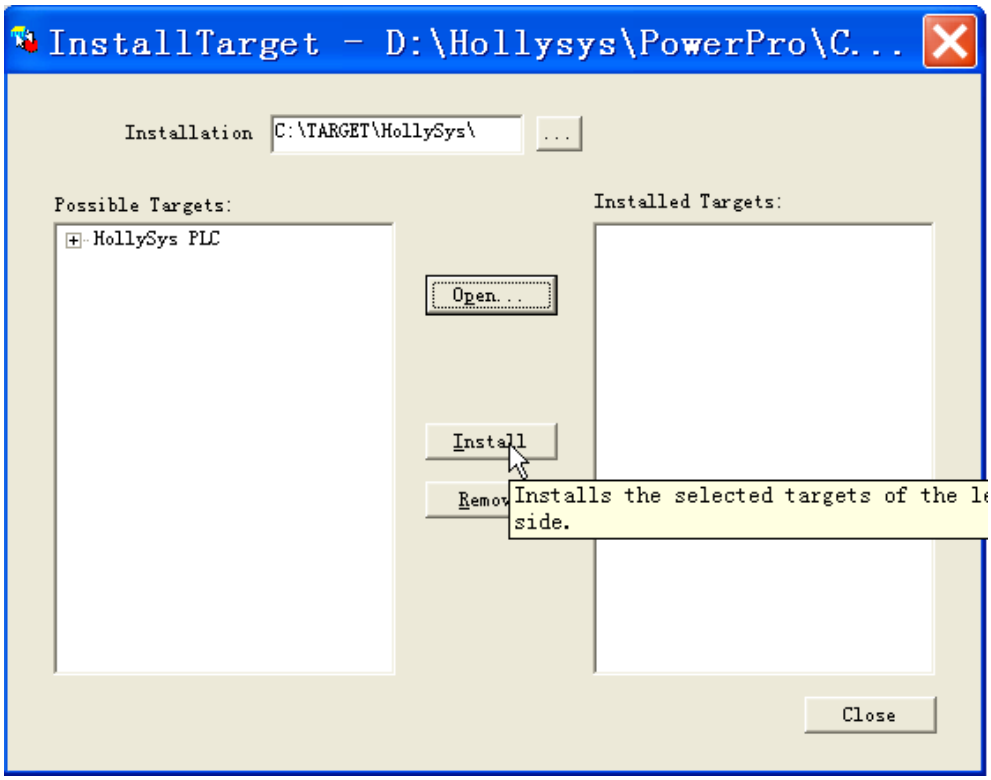


图 1-3-3 安装目标（3）

此时，在右侧已安装目标窗口“Installed Targets”中生成同样的“Hollysys PLC”目标文件。点击“Close”按钮，安装目标结束，关闭此窗口，如图 1-3-4 所示。

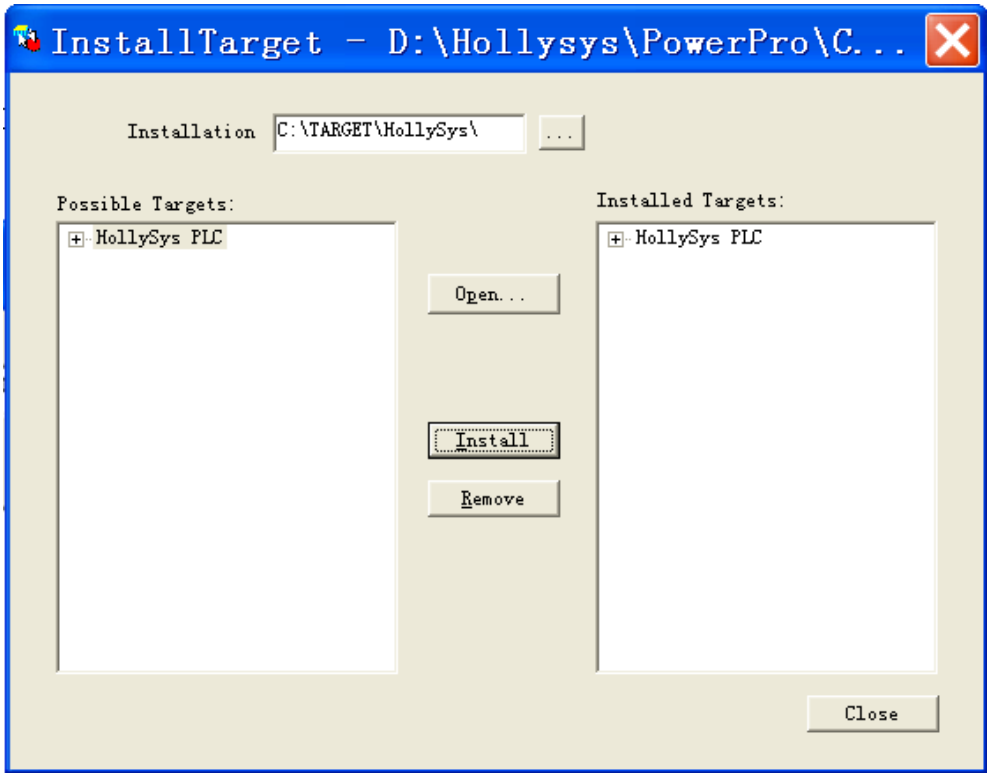


图 1-3-4 安装目标（4）

第2章 PowerPro 概述

启动 PowerPro 后，便进入了 PowerPro 的编程环境。

本章对 PowerPro 的编程环境做了概述，介绍了 PowerPro 的编程界面、菜单命令等，使得初学者能了解和熟悉 PowerPro 的编程环境。当您在使用过程中，需要了解菜单选项的作用时，可以参考本章相关内容。

2.1 PowerPro 简介

PowerPro 软件是和利时公司专为 LM 系列 PLC 所开发的基于 Windows 的编程工具。PowerPro 软件具有控制方案的编辑和仿真调试功能，是 LM 系列 PLC 的硬件配置和软件编程的标准软件包。

PowerPro 与传统的 PLC 编程软件相比，有如下一些特点和功能：

- **程序语言标准化**

在上世纪 90 年代中后期，IEC 发布了自动化行业程序语言的国际标准。先是 IEC1113-3 标准，后来修订为 IEC61131-3 标准，以帮助 PLC、NC 及 DCS 等自动化行业统一编程语言，促进编程技术的进步。PowerPro 是完全符合 IEC61131-3 标准的编程软件，具有 IL、LD、ST、FBD、SFC、CFC 等多种语言编程方式。

- **内部器件变量化**

LM 系列 PLC 没有常规 PLC 那么多的内部器件，如定时器、计数器等，取而代之的是变量。变量是 PowerPro 特有的一个概念，类似于高级语言的形式。这些变量按需要声明，使用多少，就声明多少。变量名还可按其功用命名，比起器件编号更便于辨认。变量还可分为全局与局部、输入与输出、掉电保持与不保持等多种类型。同时，利用 PowerPro 强大的计算功能，还可以定义多种的数据类型。不仅包括布尔型、字节型、字型、双字型，而且还包括指针、枚举、多维数组、单精度浮点数等类型。关于变量的详细说明，请参见 4.4 章节。

- **程序组织模块化**

PowerPro 对程序的组织是完全模块化的。PowerPro 提出 POU 的概念，POU（Program Oginization Unit）即为程序组织单元。PowerPro 的程序组织单元包括程序、函数和功能块。这三者共同完成了一个工程。PowerPro 对程序的组织，都是主程序通过对其他 POU 的调用来实现的。这既便于多人参与编程，又便于程序重用、阅读、调试，还可节省内存，确保程序安全。同时，PowerPro 是一个开放的系统，用户可以根据需要，开发出适合自己的指令。关于程序的组织，请参见第五章内容。

- **模块设定软件化**

PowerPro 是一个开放的系统。一方面，读者可以根据需要开发自己的指令；另一方面，PowerPro 将 PLC 的许多参数和模块设定通过指令的形式开放给用户，用户可以根据自己的需求，在程序中完成设定，诸如：串口通讯参数的设定等。

- **编程监控一体化**

PowerPro 软件有独特的视图和报警功能，可以在运行和调试时提供一个可视化的界面。另外，PowerPro 还提供有非常强大的仿真和调试功能，可以更方便地检查程序逻辑的正确性。关于仿真和调试功能，请参见 8.4 章节。关于视图的内容，请参见第十一章。

2.2 编程界面

启动 PowerPro 软件，进入如图 2-2-1 所示的编程环境主界面。

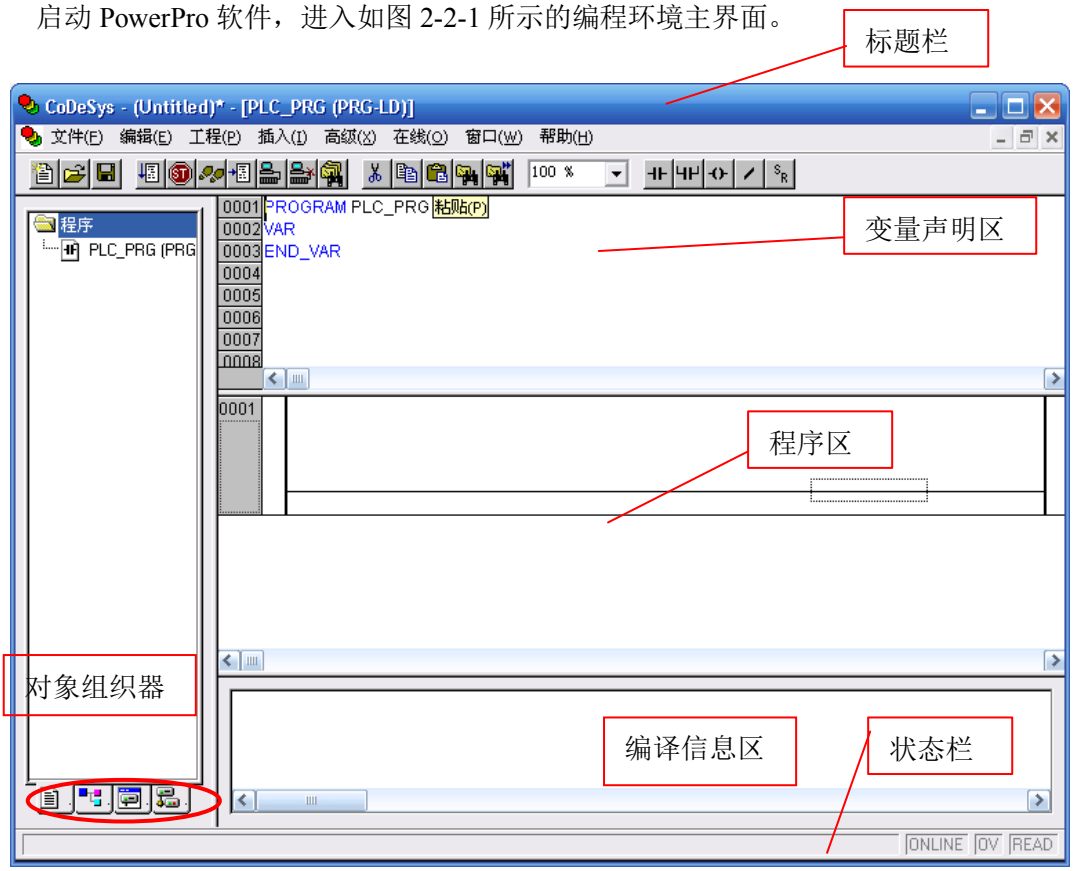


图 2-2-1 主界面

PowerPro 软件主界面中主要有下列组件：

- 标题栏：包括菜单条（操作所需的各种菜单命令）和工具条（可选），部分菜单命令可以通过工具条的快捷图标来快速选择。
- 对象组织器：由程序、数据类型、视图和资源四个选项卡组成。
- 变量声明区：显示在程序中所声明或定义的变量。
- 程序区：程序编辑和修改区域。
- 编译信息区：显示编译后的信息，包括显示程序的基本信息及错误和报警的提示信息等。
- 状态栏：显示当前工程和当前命令的相关信息。

2.2.1 标题栏

PowerPro 软件运行后，顶端标题栏如图 2-2-2 所示，主要包括菜单条（“文件”、“编辑”、“工程”、“插入”、 “高级”、“在线”、“窗口”、“帮助”）和工具条（可选），工具条上的快捷图标可以方便地实现一些常用操作。

图 2-2-2 标题栏

要想知道各个快捷图标的名称，可以把鼠标指针移至快捷键图标上，对应的快捷图标的名称就会出现在提示框中，如图 2-2-2 所示。菜单命令和快捷图标变灰表示该功能在当前窗口禁用。关于菜单中的选项，将在 2.3 章节中介绍。关于快捷图标，将在 2.4 章节中介绍。

2.2.2 对象组织器

主界面左侧的竖条窗口称为对象组织器，由“程序”、“数据类型”、“视图”和“资源”四个选项卡组成，包含了一个工程所必需的基本对象，如图 2-2-3 所示。

程序选项卡用于对程序的管理。诸如新建子程序、新建中断服务程序等都在程序选项卡中完成。数据类型选项卡完成对自定义数据类型功能。PowerPro 支持用户自定义的数据类型。视图选项卡完成视图功能。资源选项卡完成 PLC 硬件配置、添加指令、工程选项及设置中断等功能。

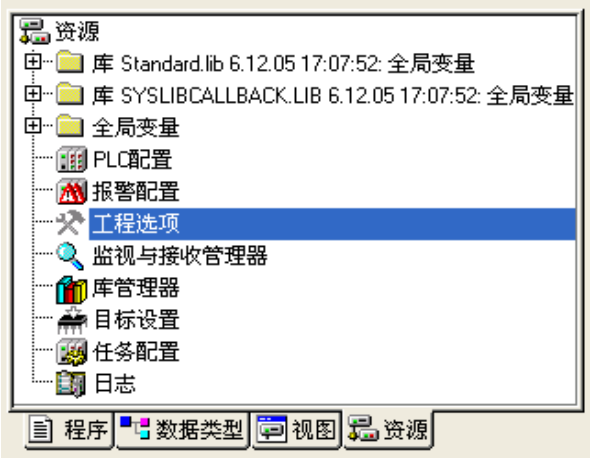


图 2-2-3 对象组织器

2.2.3 变量声明区

变量声明区位于对象组织器的右上角。PowerPro 将数据分为地址和变量两大类。变量可以不指定具体地址，直接用符号来表示，诸如“start”、“run”，同一符号的变量表示同一个变量。变量与地址不同，变量在使用时需要定义，而地址可以直接引用。变量声明区就是用于显示所有定义的变量。

变量的定义有两种方式。一种是在编程时自动定义，并且显示在变量声明区中，如图 2-2-4 所示；另一种就是直接在变量声明区中定义。关于变量的定义，可以参见 4.4 章节。

变量声明区有文本和表格两种显示形式，图 2-2-5 所示为变量表格显示形式。

注意：

1、 不能将同一个变量符号定义为两种数据类型。

2、 地址数据不会显示在变量声明区。

3、 当在程序中删除一个定义的变量时，变量声明区中不会自动删除该变量。

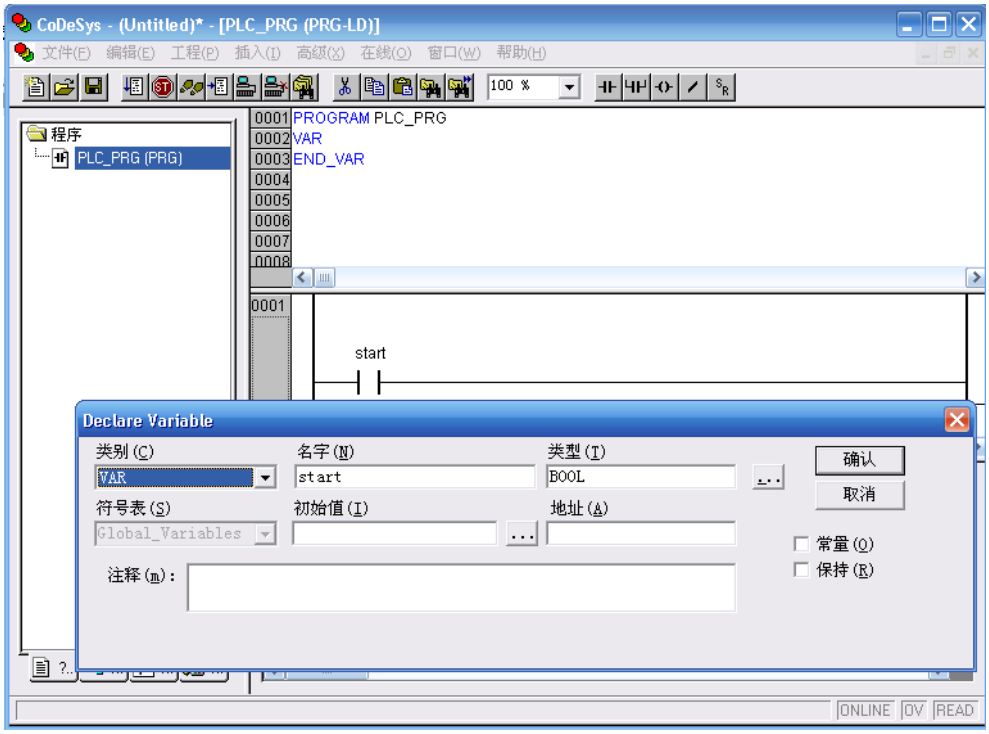


图 2-2-4 程序中自动定义变量

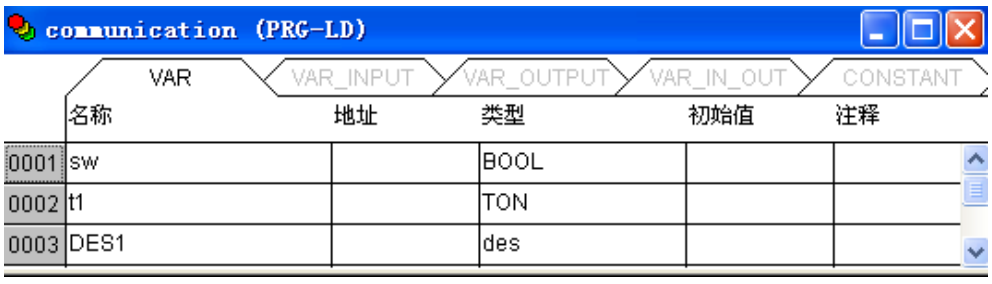


图 2-2-5 变量表格声明

2.2.4 程序区

程序区位于变量声明区的下边。在对象组织器的“程序”选项中，程序区主要指程序、函数和功能块的编辑器窗口，用于编写控制算法。由于选择的编程语言不同，编辑环境也会有所不同。根据编程语言的特点，编程语言可以分为图形编辑语言和文本编辑语言两大类。LD、SFC、FBD 和 CFC 语言的编辑器属于图形编辑器。IL 和 ST 语言的编辑器属于文本编辑器，包含了 Windows 文本编辑器的所有通用功能。

关于程序区的操作，请参见 7.4 章节。关于其他编程语言，请参见第九章。

2.2.5 编译信息区

编译信息区位于程序区的下方，用于实时显示程序关于编译、错误、警告或比较的消息，如图 2-2-6 所示。双击编译信息区中的任一条消息，可以自动跳到编辑器中的相关行，以便查找相关信息。通过“编辑”/“后错误”（F4 功能键）和“编辑”/“前错误”（Shift+F4 组合功能键）命令可以在错误消息行中快速跳转。编译信息区的显示是可选的。当“窗口”下拉菜单里“信息”命令前出现选中符号（出现“√”），则消息窗口打开，否则消息窗口关闭。

另外，编译信息区也可以显示参考数据，诸如未使用变量或重叠内存区等。关于此部分的详细内容，请参见 8.2.3 章节。

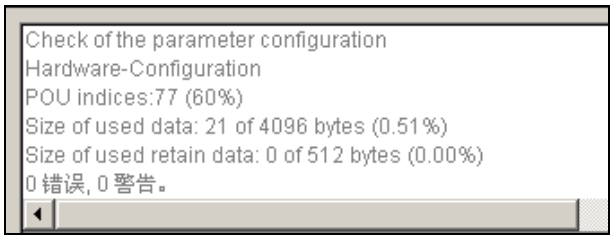


图 2-2-6 消息窗口

2.2.6 状态栏

状态栏位于主窗口边框的底部，用于显示有关当前工程和当前命令的信息。如果某项选中，相关的信息就以黑色的文本出现在状态栏中。

2.3 菜单列表

2.3.1 文件菜单

文件[F]菜单如图 2-3-1 所示，具体功能如下所述。

图 2-3-1 文件菜单

- 文件[F]/新建[N]：创建一个新工程。
- 文件[F]/从模板中新建[T]：用已有的模版来创建新工程。打开此项会弹出对话框，选择相应路径下已有模版的工程，点击“打开”按钮，便可以在相应模板下创建工程。PLC 不支持“从 PLC 打开工程”和“从源代码管理器打开工程”这两项功能。
- 文件[F]/打开[O]：打开一个工程。
- 文件[F]/关闭[C]：关闭当前打开的工程。
- 文件[F]/保存[S]：对当前打开的工程进行保存。
- 文件[F]/另存为[A]：将当前工程以新的文件名或路径保存。
- 文件[F]/保存/邮寄文档[H]：可以自动保存文档并将其压缩为 ZIP 文件进行邮寄。
- 文件[F]/打印[P]：打印当前窗口内容。
- 文件[F]/打印机设置[T]：对打印参数进行设置。打开此项会弹出对话框，可以选择打印机，对页面大小、份数和方向等参数进行设置，还可以对打印质量和打印布局进行设置。
- 文件[F]/退出[E]：退出 PowerPro 软件。

2.3.2 编辑菜单

编辑[E]菜单如图 2-3-2 所示，具体功能如下所述。

图 2-3-2 编辑菜单

- 编辑[E]/撤销[U]：撤消上一次操作。
- 编辑[E]/恢复[E]：重复上一次操作。
- 编辑[E]/剪切[T]：把所选内容复制到剪贴板，并从当前位置删除。
- 编辑[E]/复制[C]：把所选内容复制到剪贴板，但不删除所选内容。
- 编辑[E]/粘贴[P]：把剪贴板上的内容粘贴到当前位置。
- 编辑[E]/删除[D]：删除当前所选内容。

- 编辑[E]/查找[F]：在当前编辑器中查找某一文本。
- 编辑[E]/查找下一个[N]：查找与最后一次所查文本相同的内容。
- 编辑[E]/替换[R]：将查找到的目标替换成所需内容。
- 编辑[E]/输入变量[A]：可以快速输入相关内容。在编辑窗口中的当前光标位置，按F2 功能键，会自动弹出当前位置可以插入的待选项，例如运算符、函数、功能块和变量类型等列表。在左边的列表中选择输入类型，在右边的列表中选择期望的输入，选择所需输入的内容，点击“确认”按钮，则所选内容便被输入。具体请参见 7.4.3 章节。
- 编辑[E]/声明变量：弹出变量定义对话框。
- 编辑[E]/后错误：自前向后查找并显示消息窗口的提示错误或警告。
- 编辑[E]/前错误：自后向前查找并显示消息窗口的提示错误或警告。
- 编辑[E]/宏：PLC 不支持此项功能。

2.3.3 工程菜单

工程[P]菜单如图 2-3-3 所示，具体功能如下所述。

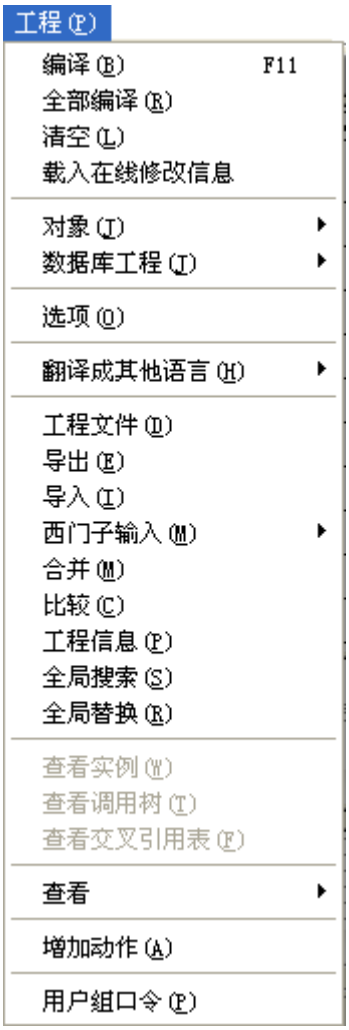


图 2-3-3 工程菜单

- 工程[P]/编译[B]：对当前工程的程序变化部分进行编译。
- 工程[P]/全部编译[R]：对当前工程的所有程序进行全部编译。
- 工程[P]/清空[L]：清除以前产生的下载文件。“清空”的目的是确保系统在下次进行编译时，重新建立下载文件。执行“清空”不影响 PLC 中的用户程序。每执行一次“清空”后再进行“编译”或“全部编译”，则在下次下载程序时，无论用户程序改变与否，均会出现提示：“程序已经更改，是否下载新程序？”，根据相应的提示完成下载。这与“在线/清空用户程序”有所不同。“在线/清空用户程序”清除 PLC 中的程序，重新初始化 PLC 系统，而“工程[P]/清空[L]”清除以前产生的下载文件，确保系统在下次进行编译时，重新建立下载文件。
- 载入在线修改信息：PLC 不支持此项功能。
- 工程[P]/对象[J]：对所选中的对象实现删除、添加、重命名、转换、复制、编辑和属性等操作，如图 2-3-4 所示。

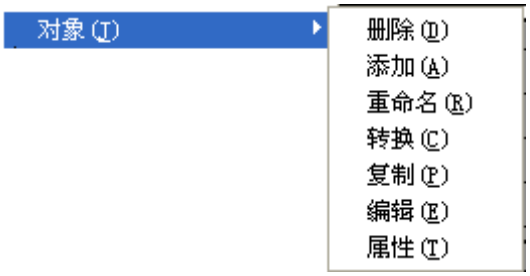


图 2-3-4 对象子菜单

删除：删除当前程序。

添加：在程序组中添加新程序，选择程序语言并命名。

重命名：给当前的程序更换名称。

转换：将当前程序转换成其它语言。例如：可以将当前的 LD 语言转换为 IL 或 FBD 语言，如图 2-3-5 所示。

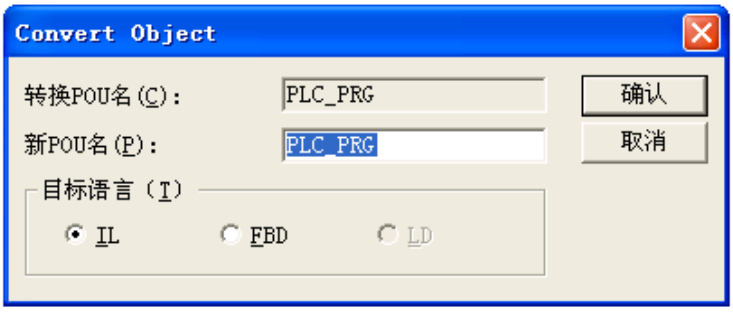


图 2-3-5 语言转换

复制：将当前程序另取名，成为一段新程序。

编辑：打开所选中的程序编辑窗口，也可双击程序名打开编辑功能。

属性：设定用户对当前程序的操作权限，如图 2-3-6 所示。

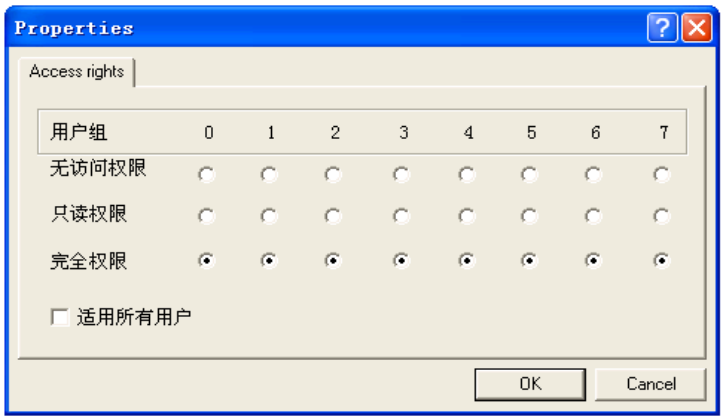


图 2-3-6 属性对话框

- 工程[P]/数据库工程[J]：PLC 不支持此项功能。
- 工程[P]/选项[O]：对当前工程的参数进行设置，例如存储方式、路径或密码等。具体用法请参见 7.6 章节。
- 工程[P]/翻译成其他语言[H]：对语言进行设置，如图 2-3-7 所示。

创建翻译文件：为当前工程创建一个.tlt 后缀的文件。

翻译工程：将当前工程按目标语言进行翻译。

查看已翻译过的工程：可以查看曾经翻译过的工程文件。

锁定翻译：一旦锁定翻译，翻译工程选项便呈现灰色，不可对该工程进行翻译操作。只有当再次选择锁定翻译时，才会允许翻译工程。

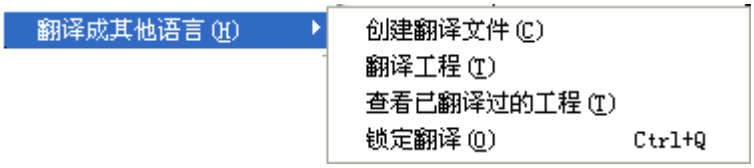


图 2-3-7 语言对话框

- 工程[P]/工程文件[D]：打印输出当前工程中包含的所有文件或部分文件。
- 工程[P]/导出[E]：将当前工程中所选中的程序导出，生成一个*.EXP 文件。
- 工程[P]/导入[I]：将*.EXP 文件导入当前工程，常用于不同工程之间的程序导入。关于导入导出功能，请参见 7.5.2 章节。
- 工程[P]/西门子输入[M]： PLC 不支持此项功能。

图 2-3-8 西门子输入

- 工程[P]/合并[M]：将其它工程中的所需内容合并到当前工程。
- 工程[P]/比较[C]：将当前工程与其它工程中的“程序”、“数据类型”、“视图”、“资源”进行比较，并将比较结果列于信息窗口。
- 工程[P]/项目信息[P]：列出当前工程的相关信息，例如名称和保存路径等。

- 工程[P]/全局搜索[S]：选择在当前工程的部分对象中搜索指定内容。
- 工程[P]/全局替换[R]：选择在当前工程的部分对象中搜索，并替换指定内容。
- 工程[P]/查看实例[W]：打开所要查看的功能块实例。如果想要查看实例，首先将编译好且无错误的程序进行在线登录，然后再进行查看。
- 工程[P]/查看调用树[T]：显示调用树窗口。必须首先对工程进行全部编译，然后才能查看。具体内容请参见 8.2.1 章节。
- 工程[P]/查看交叉引用表[F]：查看当前工程中交叉引用的变量和程序等。具体内容请参见 8.2.2 章节。
- 工程[P]/查看：可以查看与变量有关的信息，如图 2-3-9 所示。具体内容请参见 8.2.3 章节。

未使用变量：工程自动检查有无未使用变量，如果没有未使用变量，则会在消息窗口显示如下提示：No unused variables found（没有未使用变量）。

重叠内存区：工程自动检查重叠内存区，如果没有重叠内存区，则会在消息窗口显示如下提示：No variables with overlapping memory area found（没有重叠内存区）。

同时访问：工程自动检查有无同时访问，如果没有同时访问，则会在消息窗口显示如下提示：No concurrent accesses found（没有同时访问情况）。

多路写输出：工程自动检查多路写输出，如果没有多路写输出，则会在消息窗口显示如下提示：No outputs found which are written to at more than one location（没有多路写输出）

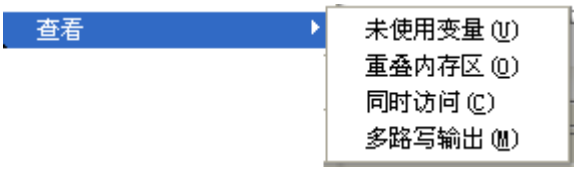


图 2-3-9 查看子菜单

在后面的章节中会介绍到在“资源”选项卡中的“工程选项”/“build”/“自动检查”中的内容，其与“工程”菜单栏里“查看”的内容相同，而且具体功能也类似。唯一不同的是，“工程/查看”只有在编译后才可以通过“查看”选择要查看的内容，例如：未使用的变量等，且只能一一查看。而“自动检查”则可以选择多项，在编译时一起实现自动检查。

- 工程[P]/增加动作[A]：在当前程序中增加程序分支。
- 工程[P]/用户组口令[P]：定义用户口令。在下次打开工程时需写入相应的口令。如果口令一致，允许对其进行操作，否则无法进行操作。

2.3.4 插入菜单

在编写程序时，使用插入工具可以定义变量，调用功能块、运算符和函数。不同的编程语言和当前位置，插入菜单所提示的待插入项目是不相同的。下面以“LD”语言为例，分几种不同的情况来介绍“插入”菜单的不同使用方法。

首先，光标处于工作区域的变量声明区所弹出的下拉菜单，与光标处于工作区域的程序区域所弹出的下拉菜单，所显示的内容是不同的，如图 2-3-10 所示。这里可插入一些 LD 语言中的功能块、输出、注释以及声明关键字、变量类型等内容。至于在具体的程序中如何插入各个选项，请参见 7.4 章节。

a) 光标在变量声明区　　b) 光标在程序区

图 2-3-10 “插入”菜单（1）

其次，在“资源”选项卡中“PLC 配置”下，此时的“插入”是对模块的插入，先插入程序中所需的 CPU 模块，再依次插入相应的扩展模块，如图 2-3-11 所示。如果 CPU 的 I/O 点数可以满足工程的需求，则不需添加任何扩展模块。

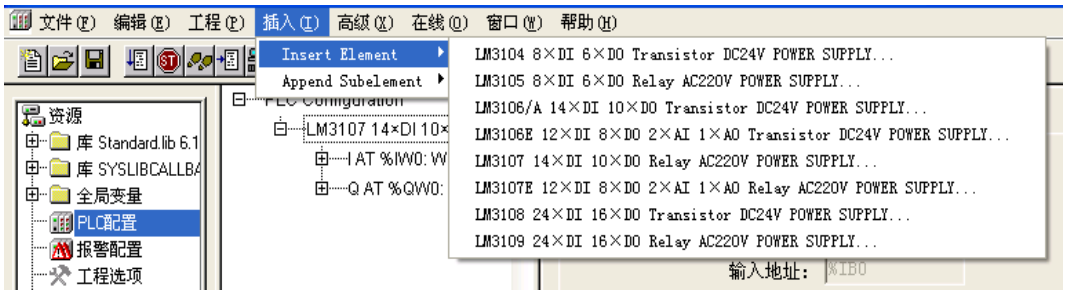


图 2-3-11 “插入”菜单（2）

再次，在“资源”选项卡中的“监视与接收管理器”下，选择“插入”菜单下的“新建监视列表”，或在“监视与接收管理器”列表区的左区点击鼠标右键，从弹出的菜单中选择“增加监视列表”，并为列表输入合适的名称，即可插入新的监视列表，如图 2-3-12 所示。

图 2-3-12 “插入”菜单（3）

2.3.5 高级菜单

使用不同的语言进行编程，或在不同的当前位置下，“高级”菜单所显示的选项不相同。

下面以“LD”语言为例，分几种不同的情况来介绍“高级”菜单。

首先，在“资源”选项卡中的“程序”选项下，“高级”菜单如图 2-3-13 所示。

图 2-3-13 “高级”菜单（1）

- 取反：表示触点和线圈取非。如果线圈取非，则取非以后的值会被保存到对应的逻辑变量中。如果触点取非，则只有当逻辑值是 FALSE 时才能连通。
- 置位/复位：线圈可以定义成置位或者复位状态。用线圈符号（S）表示一个置位线圈。一旦设置为 TRUE 值，置位线圈将一直保持为 TRUE，直到被复位。用线圈符号（R）表示一个复位线圈。一旦设置为 FALSE 值，复位线圈将一直保持为 FALSE，直到被重新置位。
- 功能块帮助：快捷键为“Alt+Enter”。在梯形图中，选中某个功能块，使用“功能块帮助”，则会弹出相应功能块的“帮助”文件，即“库管理器”，从而了解该功能块的应用。
- 选项：在梯形图中，使用“选项”菜单，则会弹出如图 2-3-14 所示的“功能块梯形图选项”对话框，可以进行梯形图的相关参数设置。关于选项，请参见 7.4.9 章节。

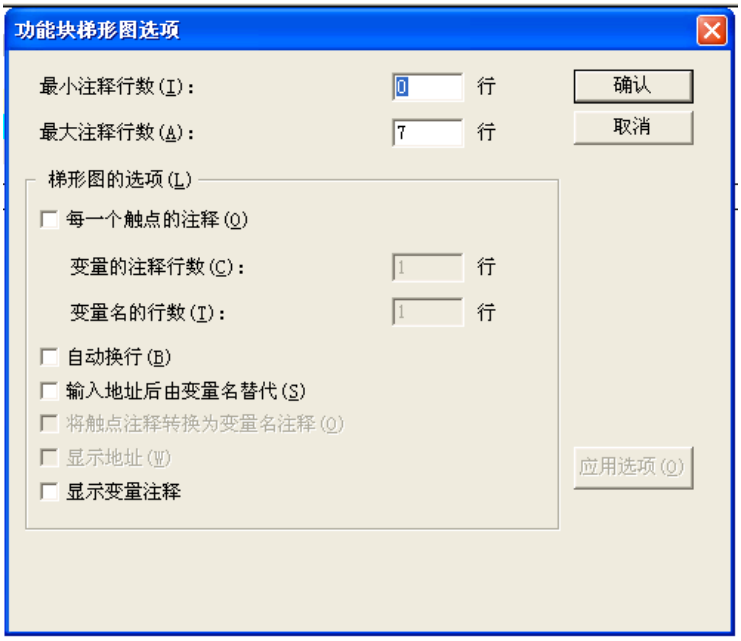


图 2-3-14 “高级”菜单（2）

其次，在“资源”选项卡中的“PLC 配置”选项下，“高级”菜单如图 2-3-15 所示。

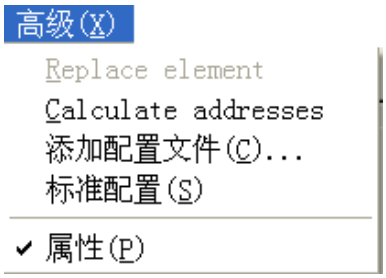


图 2-3-15 “高级”菜单（3）

- Replace element：替换 CPU 模块。PLC 软件不支持此项功能。
- Calculate addresses：自动计算地址。
- 添加配置文件：本软件无须添加任何配置文件。
- 标准配置：选择此项，会弹出如图 2-3-16 所示的对话框。点击“是”按钮，恢复默认配置。否则不改变当前配置。
- 属性：当“属性”前出现“√”，即在激活属性的情况下，会弹出基本参数、模块参数和通道参数等项目栏，否则不会显示此类信息。

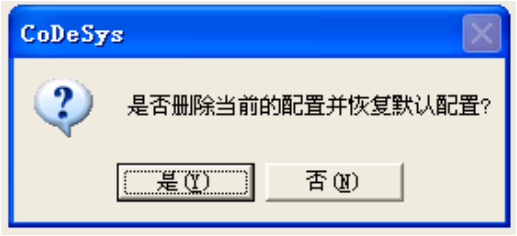


图 2-3-16 “高级”菜单（4）

再次，在“资源”选项卡中的“监视与接收管理器”选项下，“高级”菜单如图 2-3-17 所示。

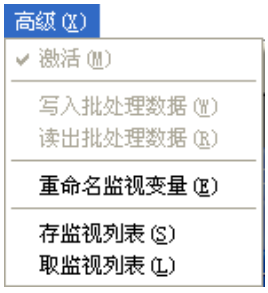


图 2-3-17 “高级”菜单（5）

- 激活：点击该项，使“激活”之前出现“√”，激活监视列表，才可以监控列表中的变量。
- 写入批处理数据：将变量的值改写为设置的变量值。
- 读出批处理数据：读取变量的当前值。
- 重命名监视变量：改变监视列表的名称。
- 存监视列表：保存监视列表，扩展名为*.wtc。
- 取监视列表：重新载入已保存的扩展名为*.wtc 的监视列表。

2.3.6 在线菜单

“在线”菜单是用于程序下载和调试的一组工具，如图 2-3-18 所示。

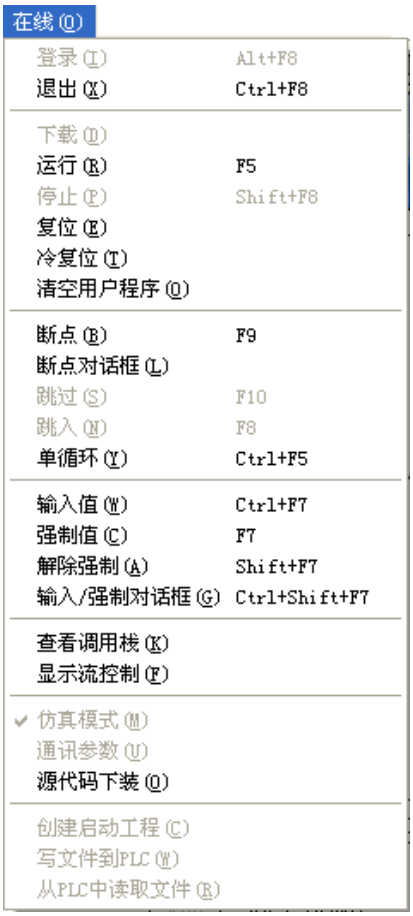


图 2-3-18 “在线”菜单

- 在线[O]/登录[I]：建立 PLC 与 PowerPro 的连接。当 PowerPro 程序与 PLC 内部一致时，自动进入调试状态；当两者不一致时，则提示是否下载程序。具体内容请参见 8.3.3 章节。
- 在线[O]/退出[X]：退出调试状态，切换到程序编辑状态。
- 在线[O]/下载[D]：把工程装载到 PLC 中。这个只有在建立了 PLC 与 PowerPro 的连接以后才有效。关于下载与登录的区别，请参见 8.3.3 章节。
- 在线[O]/运行[R]：启动程序，进入运行状态。
- 在线[O]/停止[P]：停止程序的运行。
- 在线[O]/复位[E]：停止程序的运行，变量置为初始值。retain 型变量维持当前值。
- 在线[O]/冷复位[T]：停止程序的运行，重新初始化所有变量。
- 在线[O]/清空用户程序[O]：清除 PLC 中的程序，重新初始化 PLC 系统。注意，这与“工程/清空”菜单有所不同。“在线/清空用户程序”是指清除 PLC 中的程序，重新初始化 PLC 系统。而“工程/清空”菜单清除以前产生的下载文件，确保系统在下次进行编译时，重新建立新的下载文件。
- 在线[O]/断点[B]：在当前位置设置一个断点或删除已有的断点。如果程序运行后到达断点，则程序终止，相应得程序段会以红色背景显示。为了继续程序的运行，可

用“在线/运行”、“在线/跳过”或“在线/跳入”命令。具体请参见 8.4.6 章节。

- 在线[O]/断点对话框[L]：编辑整个工程中的断点。
- 在线[O]/跳过[S]：单步执行程序，程序在执行之后停止。
- 在线[O]/跳入[N]：如果在当前位置是函数或功能块，则程序将执行到被调用程序的第一条指令。在其它情况下，与“在线/跳过”命令一样。
- 在线[O]/单循环[Y]：程序执行一次循环之后停止运行。具体请参见 8.4.8 章节。
- 在线[O]/输入值[W]：调试时修改变量值。
- 在线[O]/强制值[C]：同样用于调试时对变量赋值。在每个循环结束之后，被强制的变量都被写入强制值，直到执行“解除强制”命令为止。
- 在线[O]/解除强制[A]：终止变量的强制命令。
- 在线[O]/输入/强制对话框[G]：允许对多个变量写入新值，并同时输入到 PLC 中。对于“在线/输入值”，变量只被写一次，而且变量允许立刻被其它程序赋值。对于“在线/强制值”，变量在每一个循环之后被写入强制值，直到执行“解除强制”命令为止。
- 在线[O]/查看调用栈[K]：在仿真模式下，显示在调用堆栈里运行程序的列表。
- 在线[O]/显示流控制[F]：查看程序的运行流程。
- 在线[O]/仿真模式[M]：如果仿真模式被选择，则选中符号“√”将出现在菜单项的前面。在仿真模式下，用户程序运行在操作系统平台下的本地计算机内。此模式可用来检查工程。如果不运行于仿真模式，那么可以直接将程序运行于 PLC 中。
- 在线[O]/通讯参数[U]：设置本地计算机与一个或多个 PLC 模块的通信参数。
- 在线[O]/源代码下载[O]：PLC 不支持此项功能。
- 在线[O]/创建启动工程[C]：将下载到 PLC 中的工程作为 PLC 运行的默认工程。所谓启动工程，是指保存在 PLC 的 flash 中且上电后可以运行的用户程序。如果创建启动工程的代码与上次下载的代码不同，系统会给出相应提示：“当前代码与上次下装代码不匹配！是否继续？”，如图 2-3-19 所示。一般情况选择“是”，从而创建启动工程。在离线模式下，将会产生*.ri 文件。创建启动工程与登录及下载的区别，请参见 8.3.3 章节。
- 在线[O]/写文件到 PLC[W]：PLC 不支持此项功能。
- 在线[O]/从 PLC 中读取文件[R]：PLC 不支持此项功能。

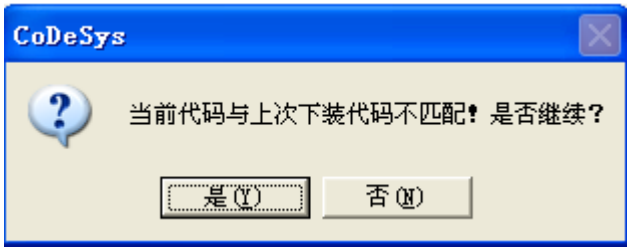


图 2-3-19 创建启动工程

2.3.7 窗口菜单

窗口[W]菜单如图 2-3-20 所示，具体功能如下所述。

图 2-3-20“窗口”菜单

- 窗口[W]/垂直[V]：所有窗口在工作区垂直排列，不重叠，且填充整个工作区。
- 窗口[W]/水平[H]：所有窗口在工作区水平排列，不重叠，且填充整个工作区。
- 窗口[W]/层叠[D]：所有窗口在工作区串联排列，一个叠一个，依次错开。
- 窗口[W]/排列图标[A]：在工作区底部的一行内，顺次排列所有的最小化窗口。
- 窗口[W]/全部关闭[L]：关闭所有打开的窗口。
- 窗口[W]/信息[M]：显示/关闭消息窗口。
- 窗口[W]/库管理器[L]：打开库管理器窗口，添加或删除库函数。
- 窗口[W]/日志[O]：打开日志窗口。

2.3.8 帮助菜单

帮助[H]菜单如图 2-3-21 所示，提供本软件的相关帮助，具体功能如下所述。

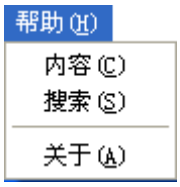


图 2-3-21 “帮助”菜单

- 帮助[H]/内容[C]：弹出帮助主题，并将帮助的相关项目列出，以供方便、快捷地查找帮助项目。
- 帮助[H]/搜索[S]：弹出对话框，提示输入所需查找项目的关键字。
- 帮助[H]/关于[A]：软件名称及版本信息。

2.4 快捷工具

首先介绍工具条中的快捷图标。如果想要知道各个快捷图标工具按钮的名称，可以把鼠标指针移至快捷图标工具按钮上，对应快捷图标工具按钮的名称就会出现在提示框中。菜单命令和工具按钮变灰表示该功能在当前窗口禁用。

2.4.1 文件工具

-  新建工程：新建一个工程。
-  打开文件/工程：打开已经存在的文件/工程。
-  保存：保存文件。

2.4.2 调试工具

-  运行：启动登录程序的运行。
-  停止：停止程序的运行。
-  跳过：单步执行程序，程序在执行之后停止。“跳过”命令遇到功能块或函数时，会跳过功能块执行下一条语句。
-  断点：在程序中设置进程停止的地方。
-  登录：进入调试状态。
-  退出：退出调试状态，回到程序编辑状态。
-  全局搜索：在整个工程中查找所需目标。

2.4.3 编辑工具

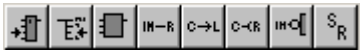
-  剪切：将选中的部分剪切到剪贴板。
-  复制：将选中的部分复制到剪贴板。
-  粘贴：将剪贴板中的内容粘贴到当前窗口。
-  查找：可以在当前编辑器中查找某一文本。打开“查找”对话框，在查找内容字段中输入要查找的字符序列。此外，还可以确定正在查找的文本是否全字匹配，或者是否区分大小写，从当前光标位置向上或是向下查找等。
-  查找下一个：继续查找最近的一次“编辑”/“查找”的文本。

2.4.4 编程工具

编程工具在不同编程语言下的表示形式是不相同的。按照编程语言种类的不同，编程工具的具体功能如下所述：

- LD 语言：  ，

依次为触点、并联触点、线圈、取反、置位/复位等功能的操作符。

- FBD 语言： 

依次为输入、输出、赋值、跳转、返回、运算符、反向、置位/复位等功能的操作符。

- SFC 语言： 

依次为前步转移、后步转移、右选择分支、左选择分支、右并行分支、左并行分支、跳转、转移跳转、使用 IEC 步等功能的操作符。

- CFC 语言： 

依次为输入、输出、块、跳转、标记、返回、注释、反向、置位/复位、使能输入输出、创建宏、输入引脚、输出引脚、退出所有宏、退出当前宏、跳转到宏等功能的操作符。

对于 IL 和 ST 语言，无此类编程工具。具体快捷编程工具的应用，请参见第九章。

2.5 对象组织器

主界面左侧的竖条窗口称为对象组织器，由“程序”、“数据类型”、“视图”和“资源”等四个选项卡组成，包含了一个工程所必需的基本对象，如图 2-5-1 所示。

图 2-5-1 对象组织器（1）

2.5.1 程序

“程序”选项的功能是建立 POU（程序组织单元），用来编写用户程序。在对象组织器“程序”选项卡的空白区域点击右键，弹出如图 2-5-2 所示的菜单，具体功能如下所述。

图 2-5-2 程序

- 添加：在程序组中添加新程序，选择程序语言并给新程序命名。
- 重命名：给当前程序更换名称。
- 编辑：打开所选中的程序编辑窗口，也可双击程序名打开编辑功能。
- 复制：将当前程序另赋予新名称，成为一段新程序。
- 删除：删除当前程序。
- 转换：可以实现不同语言的程序转化。
- 属性：可以设置不同用户组的使用权限。
- 数据库工程：PLC 不支持此项功能。
- 添加动作：可以在当前选定的程序或功能块下创建一个动作。在弹出的对话框中填入动作名，并选择动作实现的语言。“动作”代表一个进一步的功能实现，这种功能也可由其他语言所创建的一般功能块来实现。动作隶属于一个功能块或程序，可以像调用功能块一样来调用功能块的动作。动作调用的格式为：<程序名>.<动作名>或<实例名>.<动作名>。
- 新文件夹：可以自动生成“NewFolder”文件夹。如果某个文件夹被选中，新建文件夹“New Folder”将在其下层创建。另外，在该文件被选中时，点击右键，选择“重命名”还可以对文件夹的名称进行修改。
- 扩展节点：选择“扩展节点”，节点由“⊞”变为“⊟”，将文件夹内的内容进行扩展。
- 合拢节点：选择“合拢节点”，节点由“⊟”变为“⊞”，将已扩展的文件进行合拢。
- 查看实例：此命令可打开和显示当前功能块的实例列表，双击功能块也可以打开该列表。另外，选择“工程”/“查看实例”命令，也同样可以打开该列表。应当注意的是，该命令只有“在线模式”下才有效，即工程编译成功，通过“在线”/“登录”下载到模块中。
- 显示调用树：与“工程”/“查看调用树”命令相同。可以在一个新窗口中显示当前对象

调用程序、函数、功能块的树型结构，直观地指出当前 POU 与工程中其它 POU 的先后调用关系。工程必须通过编译，该命令才有效。

- 保存为模板：系统会自动将当前工程中定义的变量、工程选项中所做的一些设置的修改等，作为模板进行保存。
- 不参与编译：选择此项，当前程序、功能块或函数变为绿色。在编译过程中，并不参与程序的编译。如果想取消此项，需再次选择该项，将“不参与编译”前的“√”删除，便可取消该项，此时，当前程序、功能块或函数由绿色变为黑色。

2.5.2 数据类型

在“数据类型”选项卡中，和系统标准数据类型一样，用户可以定义自己的数据类型，生成结构型、枚举型或引用型等数据类型。例如，在“数据类型”选项卡中点击右键，添加数据类型，定义添加的数据类型“A”，如图 2-5-3 所示。关于数据类型的详细定义，请参见 4.6 章节。

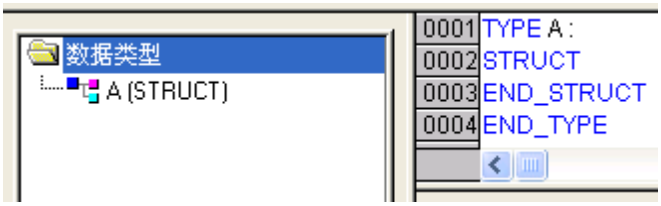


图 2-5-3 数据类型

2.5.3 视图

“视图”选项用于建立视图，观察工程变量，如图 2-5-4 所示。关于“视图”的具体应用，请参见第十一章。

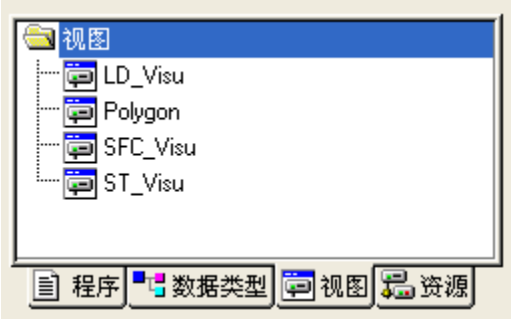


图 2-5-4 视图

2.5.4 资源

“资源”选项用来定义全局变量、配置硬件模块和组织工程，如图 2-5-5 所示。

图 2-5-5 资源

- “全局变量”：用于定义在整个工程中均能使用的全局变量。
- “PLC 配置”：用于系统 PLC 硬件配置。
- “报警配置”：用于配置报警各项参数。
- “工程选项”：用于配置当前工程参数，如存储方式、路径或密码等。
- “监视与接收管理器”：用于“在线模式”下，集中监控同一工程不同程序的变量值。具体应用，请参见 8.4.13 章节。
- “库管理器”：与“窗口”菜单下“库管理器”的内容一致，均可用来显示所有与当前工程连接的库及添加所需的库。关于库的操作，请参见 7.4.4 章节。
- “目标设置”：用于设定目标平台和查看存储区域分配等。
- “任务配置”：用于创建任务，调用程序。
- “日志”：用于显示工程日志信息。

第3章 快速入门

本章的主要内容是通过编写简单的程序，对 PowerPro 的编程方式进行介绍，使初学者对 PowerPro 的编程有一个初步的认识，了解 PowerPro 的基本操作。假如您第一次使用 PowerPro，建议仔细阅读本章节。

本章示例程序的主要功能是在一定的时间间隔内，使开关不断地进行交替通断。

在编程前，首先需要确定硬件配置。对于此程序，无需太多的 I/O 点数，仅需配置一个 CPU 模块即可满足要求，这里选择带有 24 点 I/O 的 CPU 模块 LM3107。

3.1 硬件连接

➢ 所需设备

已经安装 PowerPro 软件且带有 RS232 串口的 PC 机一台。

CPU 模块 LM3107 一台。

220V 交流电源线一根。

连接 PC 机与 CPU 模块的 RS232 通讯专用编程电缆一根。

➢ 选择 CPU 模块

选择 LM3107 模块，按照图 3-1-1 所示的方式连接电源线。注意，当电源线连接好之后，应该把端子盖扣好，以免造成不必要的人身伤害或设备损坏！

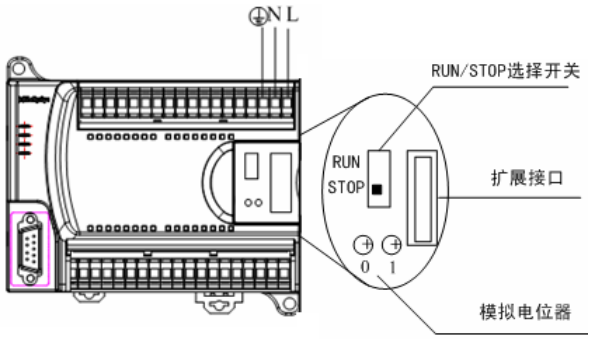


图 3-1-1 连接电源线

电源线连接之后，先不要接通电源。在检查所有电缆连接无误后，再接通电源，并确认 CPU 模块面板上的 RUN 或 STOP 指示灯点亮并显示正常，以保证 PLC 可靠运行。

LM 系列 PLC 的 CPU 模块具有两种运行方式可供选择，具体设置由“RUN/STOP”选择开关来完成，详细说明见表 3-1-1 所示。

如果在 RUN 方式下，不能向 CPU 模块下载程序时，则必须把 CPU 模块置于 STOP 方式下才能进行下载。

表 3-1-1 “RUN/STOP 选择开关”设置说明

开关状态	说明
RUN	CPU 处于运行方式，执行用户程序
STOP	CPU 不执行用户程序，此时用户可以向 CPU 下载用户程序。

➢ 建立 PC 通讯

通过配套的编程电缆，将 CPU 模块连接到 PC 机，建立数据传递通道，如图 3-1-2 所示。

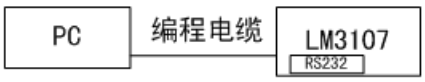


图 3-1-2 连接编程电缆

注意：

通讯电缆连接必须在模块上电之前，否则容易损坏设备！

3.2 启动软件

如果第一次使用 PowerPro 软件，一定要进行目标安装。1.3 章节已经介绍了目标安装的方法。在 Windows 的“开始”/“程序”菜单中，点击“HollySys”/“PowerPro”，启动 PowerPro 软件，如图 3-2-1 所示。

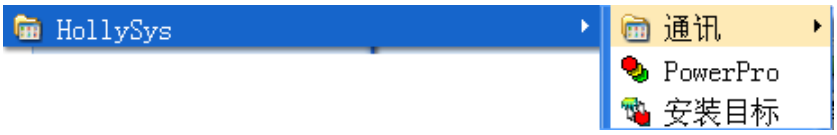


图 3-2-1 启动 PowerPro 软件

图 3-2-2 是 PowerPro 软件的界面。

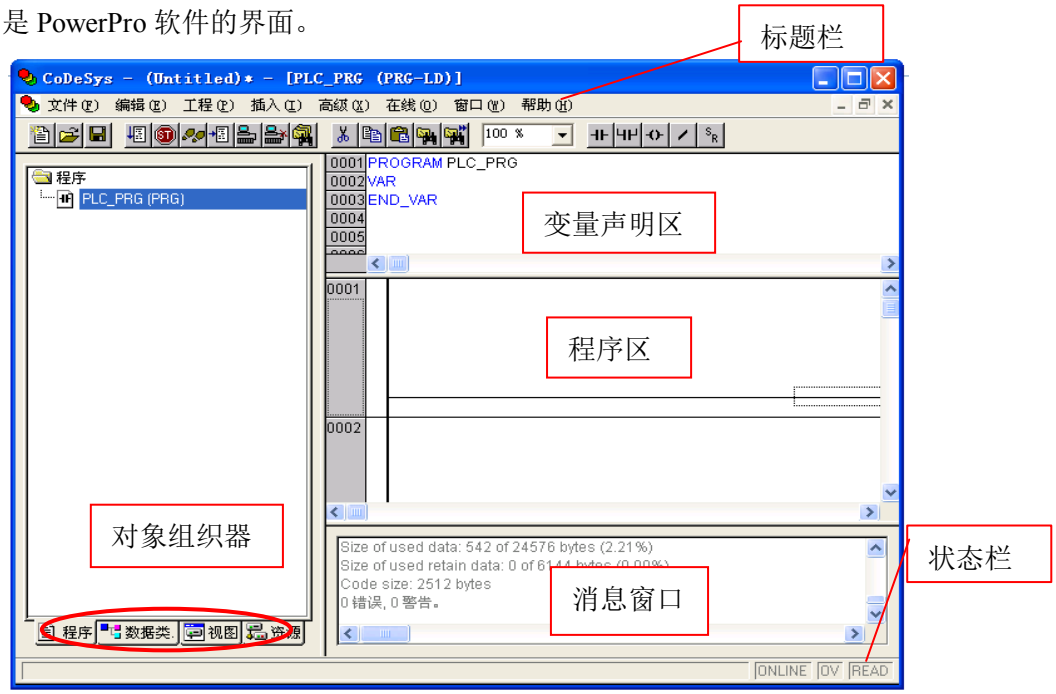


图 3-2-2 PowerPro 软件界面

3.3 新建工程

➢ 目标设置

在主界面中点击“文件”/“新建”菜单，或在工具栏中点击“  ”按钮，随之出现“目标设置”对话框。“目标”是指 PLC 的存储空间，目标设置是指根据所选择的 PLC 的存储空间来进行配置。

在“配置”栏中选中“HOLLiAS-LEC G3 CPU Extend”，此目标为程序存储空间为 120KB 的 CPU 所选用的设置，点击“确认”按钮，如图 3-3-1 所示。如果所使用的模块为存储空间 28KB 的 CPU，则需选择“HOLLiAS-LEC G3 CPU”。若不确定模块的程序存储空间大小，请参见附录。若需编写库指令，则应选择 None。关于库的制作，请参见 7.4.5 章节。

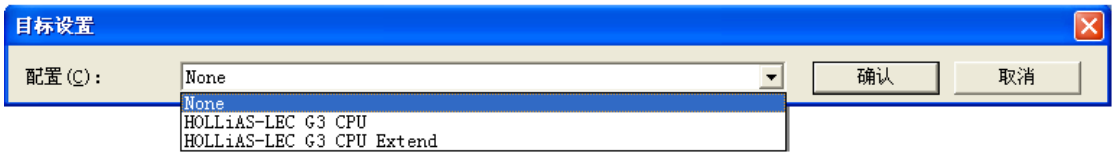


图 3-3-1 选择目标

随后弹出“目标设置”窗口，默认设置已能满足绝大多数应用需求，点击“确认”按钮即可，如图 3-3-2 所示。

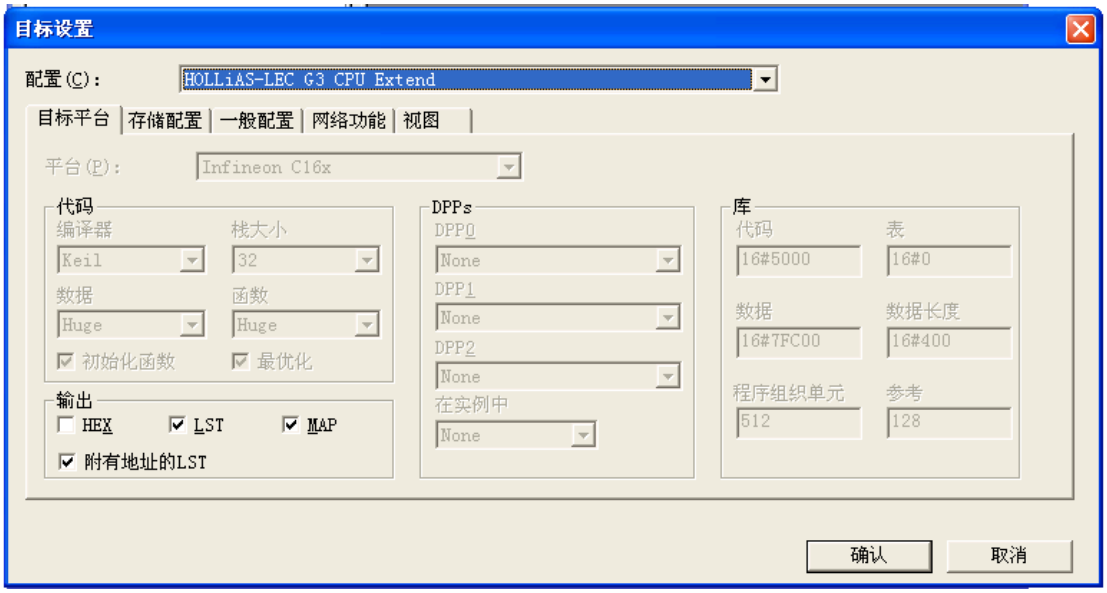


图 3-3-2 目标设置界面

关于目标配置，请参见 7.1 章节。

➢ 创建 POU

POU 是指程序组织单元，是组成工程的基本构件。第五章详细讲述了 POU 的知识。

这里以“LD”语言为例，POU 语言选择“LD”，即梯形图语言，如图 3-3-3 所示。关于其他语言的详细介绍，请参见第九章。

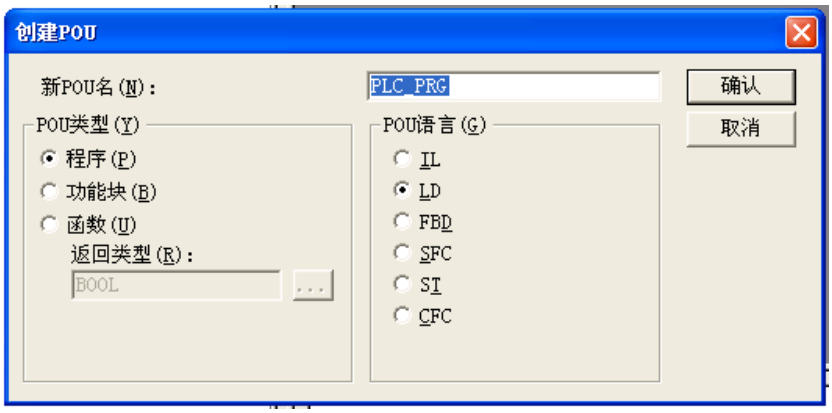


图 3-3-3 创建 POU

POU 类型选择程序，POU 的名称为自动默认的 PLC_PRG。PLC_PRG 是系统自动默认的程序名。关于 PLC_PRG，请参见 5.1.3 章节。

点击“确认”按钮，则会出现如图 3-3-4 所示界面。

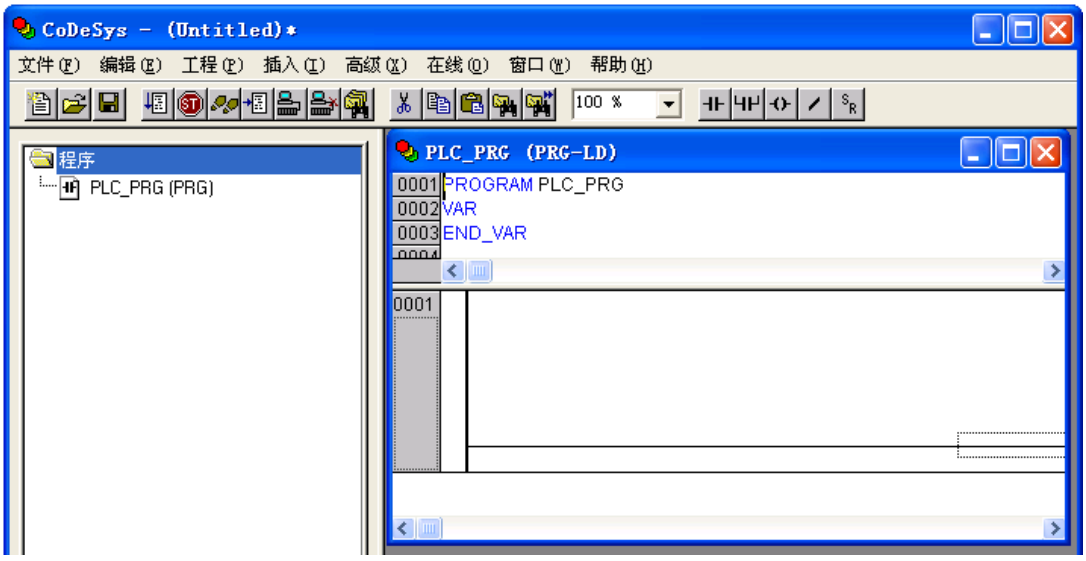


图 3-3-4 工作区域

3.4 PLC 配置

在新建一个工程后，需要进行 PLC 的硬件配置。建议用户在具体编程之前就完成 PLC 配置工作，避免在程序中出现寻址错误的现象。

在“资源”选项卡中双击“PLC 配置”，在“PLC Configuration”上点击鼠标右键，选择“Append Subelement”中的“LM3107”，进行 PLC 的配置，如图 3-4-1 所示。

关于 PLC 配置的详细说明，请参见 7.3 章节。

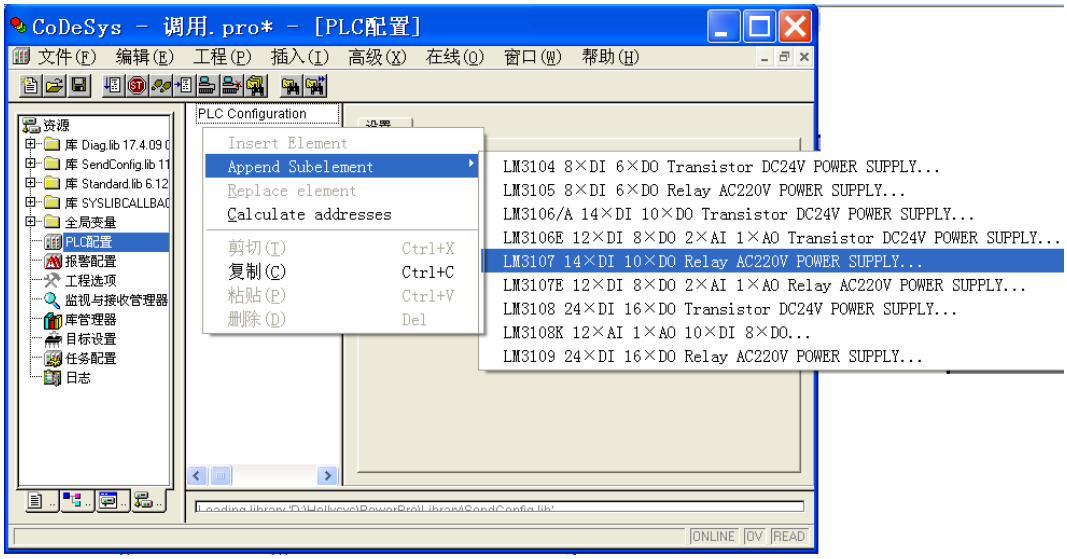


图 3-4-1　PLC 配置

3.5 设置通讯参数

在“在线”菜单中选定“通讯参数”，弹出通讯参数设置对话框，如图 3-5-1 所示。

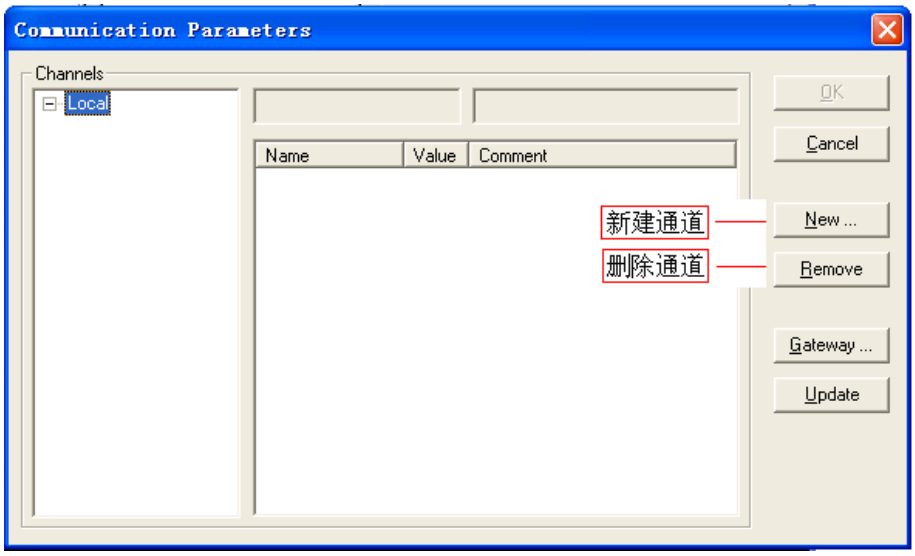


图 3-5-1 通讯参数对话框

选择“New”按钮添加新通道，出现如图 3-5-2 所示的对话框。

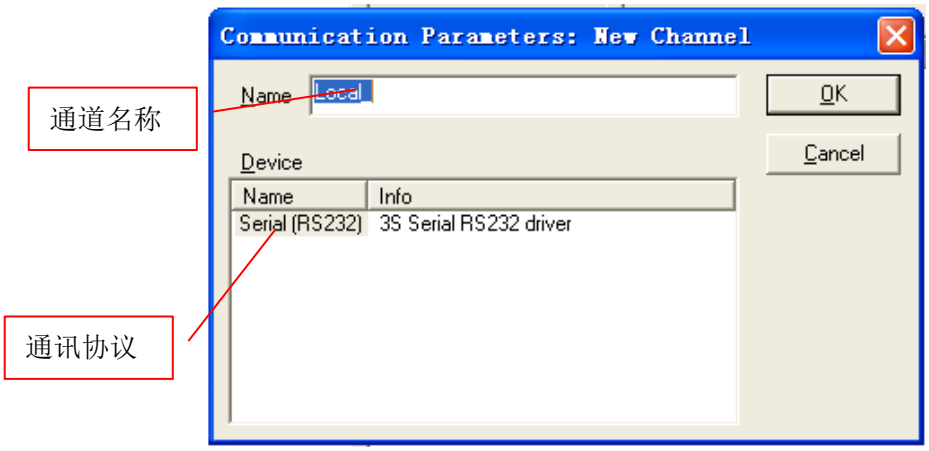


图 3-5-2 添加新信道

其中通道名称默认为“Local_”，通信协议使用缺省的 RS232 协议。点击“OK”按钮后返回通讯参数设置对话框，如图 3-5-3 所示。点击“OK”按钮确认。这样，本地计算机与 CPU 模块之间的通信连接便建立完成。

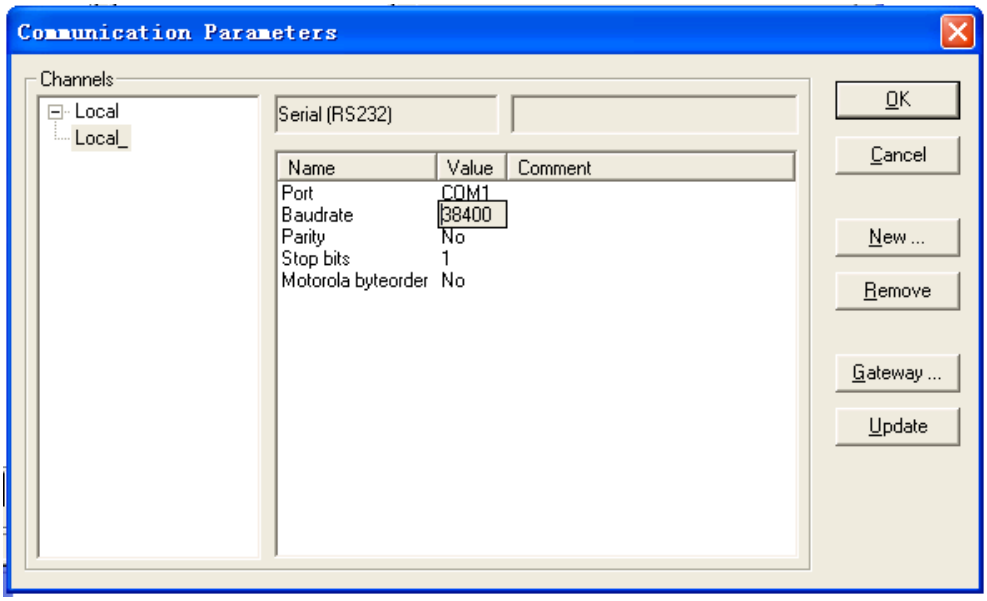


图 3-5-3 通讯速率设置

注意：

要想改变端口设置或者波特率设置，鼠标左键快速双击“Port”、“Baudrate”所对应的“Value”值。

3.6 编写程序

PLC 配置完成后，可以开始进行 PLC 程序的编写。本例是一个简单的定时器应用程序，主要目的是产生一个“1s 断 2s 通”的脉冲信号。

在工具栏里点击表示“触点”按钮“  ”，便在程序区域内的 0001 节中出现了一个触点“  ”，如图 3-6-1 所示。

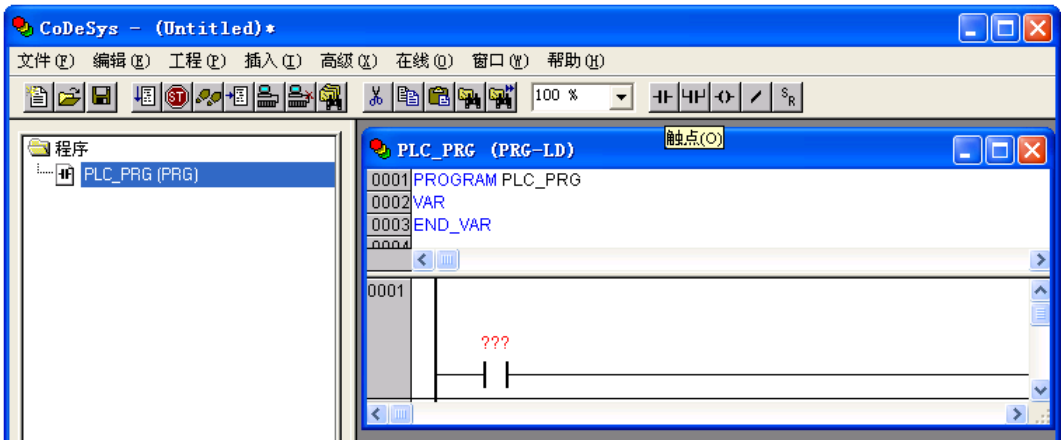


图 3-6-1 程序编写步骤（1）

触点的标记文本缺省值为“???”。点击此文本，输入“%IX0.0”，如图 3-6-2 所示。%IX0.0 表示 PLC 的第一个输入点。关于数据地址，请参见 4.2 章节。

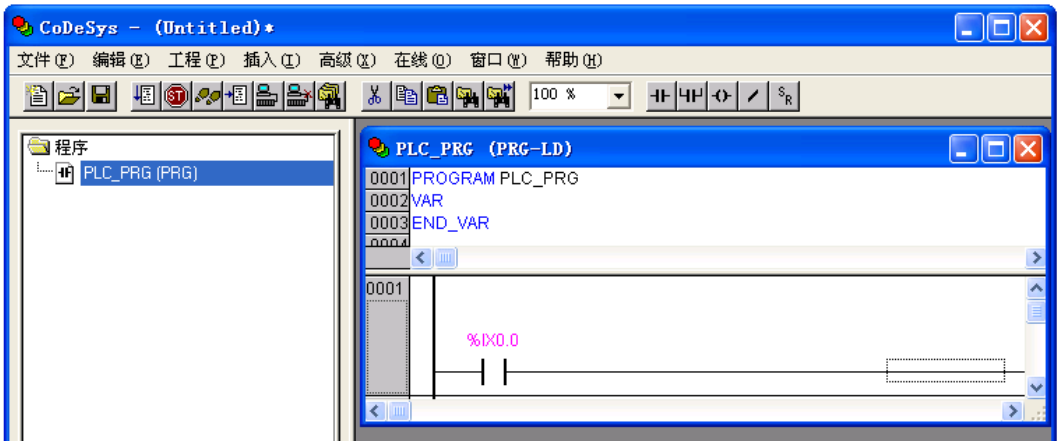


图 3-6-2 程序编写步骤（2）

在“%IX0.0”触点后点击鼠标右键，选择“功能块”，如图 3-6-3 所示。

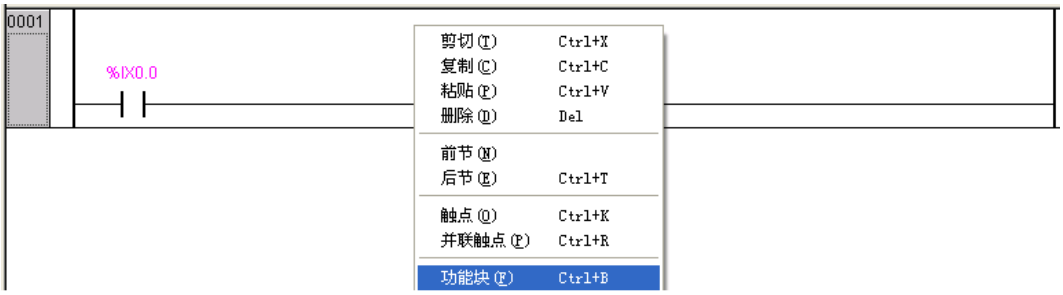


图 3-6-3 程序编写步骤（3）

点击“功能块”，则会弹出如图 3-6-4 所示的对话框，选择“TON（FB）”。TON 指令是通电延时接通定时器。

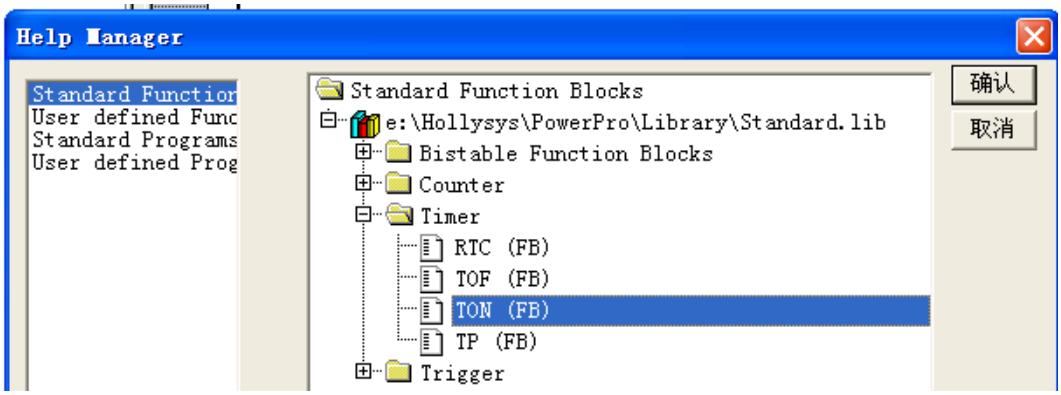
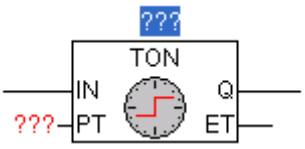


图 3-6-4 程序编写步骤（4）

双击“TON (FB)”或者选中“TON (FB)”后点确认按钮，在“ 

”光标所在位置处，输入 T1 后回车，则会弹出如图 3-6-5 所示的对话框，选择默认类型为“TON”，点击“确认”按钮。T1 是用于标识 TON 指令的一个标识符。对于所有的功能块，都需要用一个标识符来标识该功能块。关于这方面的详细信息，请参见 5.3.2 章节和 7.4.3 章节相关内容。

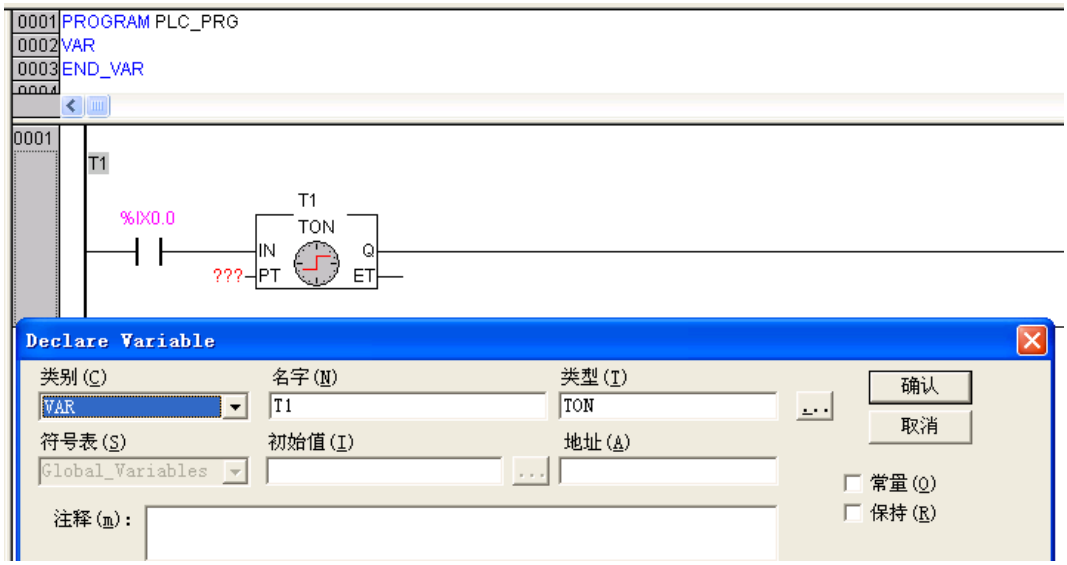


图 3-6-5 程序编写步骤（5）

在梯形图中增加了一个以 T1 命名的通电延时定时器“TON”。在“PT”的标记文本“???”处，点击文本，输入表示延时 1S 的常量“T#1S”。在“ET”处输入变量“ET”，此变量为时间类型变量，类型选择时间类型“TIME”，如图 3-6-6 所示。点击“确认”按钮返回。

PT 变量表示输入的时间参数。在这里可以填写一个时间常量，也可以填入一个时间变量。

在这里填入一个时间常量，定时时间为 1S。关于常量和变量，请参见 4.3 和 4.4 章节。

ET 表示定时器定时后所经过的时间，即当前时间。在 ET 处定义一个时间变量，可以用来观察当前经过时间。

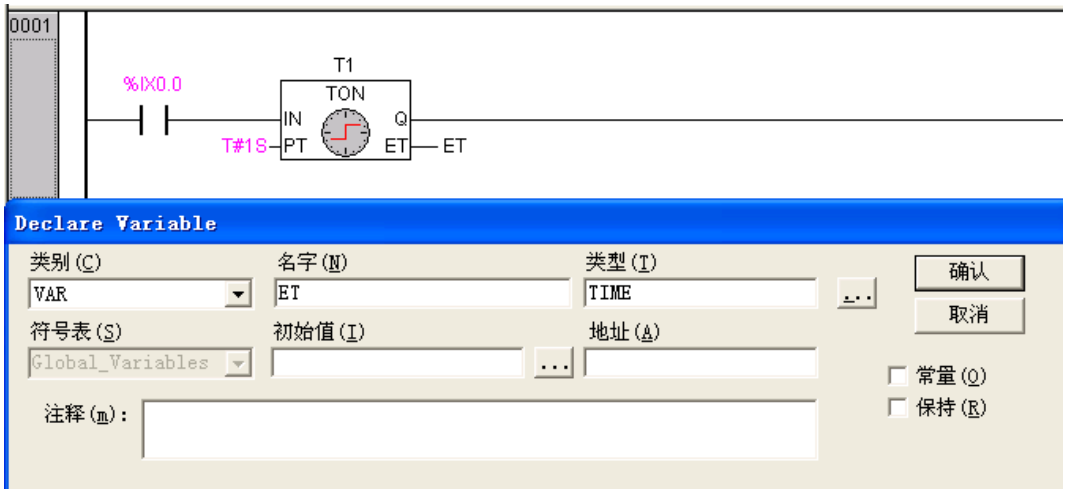


图 3-6-6 程序编写步骤（6）

当光标位置位于“T1”后时，在工具栏中选择表示输出线圈的按钮“  ”，则会在光标处出现如图 3-6-7 所示的输出线圈“???”。

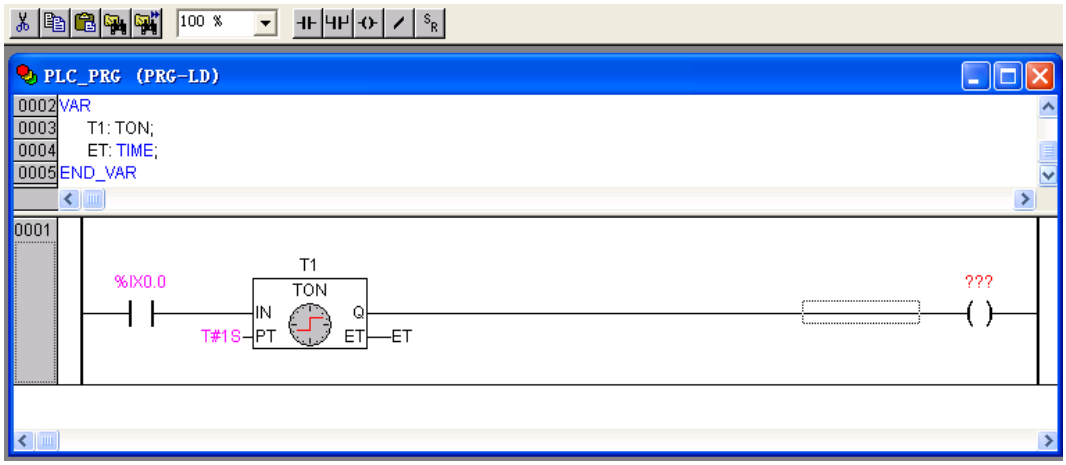


图 3-6-7 程序编写步骤（7）

在相应的标记文本“???”处填入变量名“M”，其类型为“BOOL”，如图 3-6-8 所示。点击“确认”按钮返回。

M 属于中间变量，可以参见 4.4 章节查阅更详细信息。

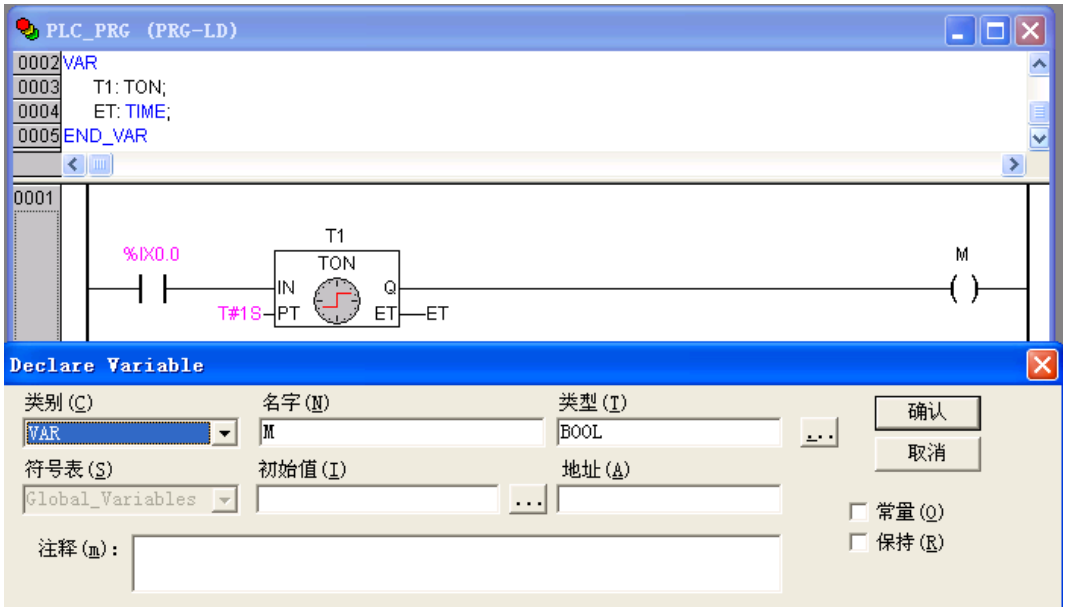


图 3-6-8 程序编写步骤（8）

在工作区域上点击鼠标右键，选择“后节”，如图 3-6-9 所示，进行 0002 节的程序编写。

图 3-6-9 程序编写步骤（9）

0002 节的梯形图如图 3-6-10 所示。其编写方法与 0001 节相同。

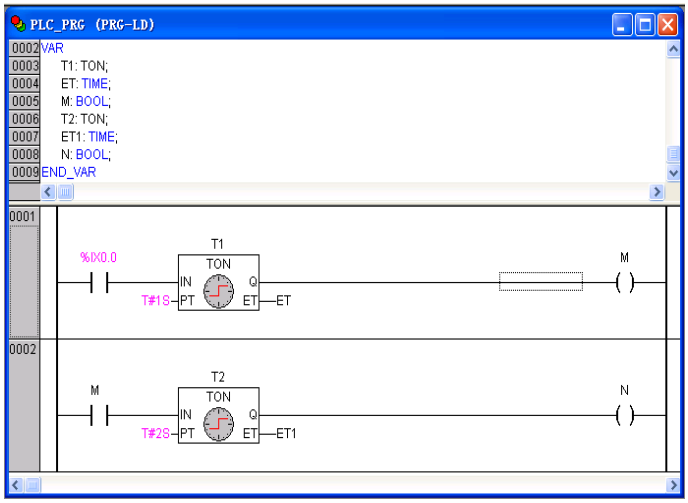


图 3-6-10 程序编写步骤（10）

为了实现开关的交替通断，需要在“T1”前添加“N”开关。选中“T1”，点击工具栏中的“串联”快捷按钮，或点击鼠标右键选择“触点”，添加开关，如图 3-6-11 所示。

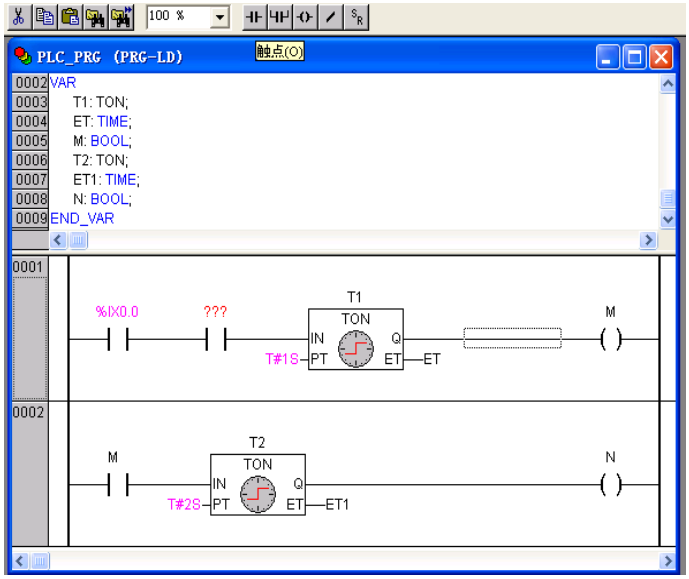


图 3-6-11 程序编写步骤（11）

点击触点标记文本“???”，输入“N”。在工具栏中选择“取反”，便添加了一个取反开关，如图 3-6-12 所示。

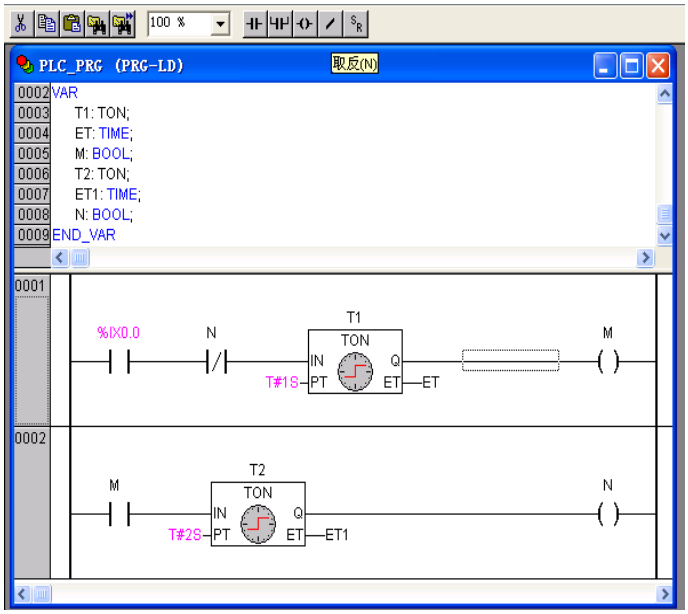


图 3-6-12 程序编写步骤（12）

为了将 M 的值从%QX0.0 输出，可以增加 0003 节。在 0003 节中添加触点“M”，并在其后添加线圈。在工具栏里选择输出按钮，并将其定义为“%QX0.0”，如图 3-6-13 所示。

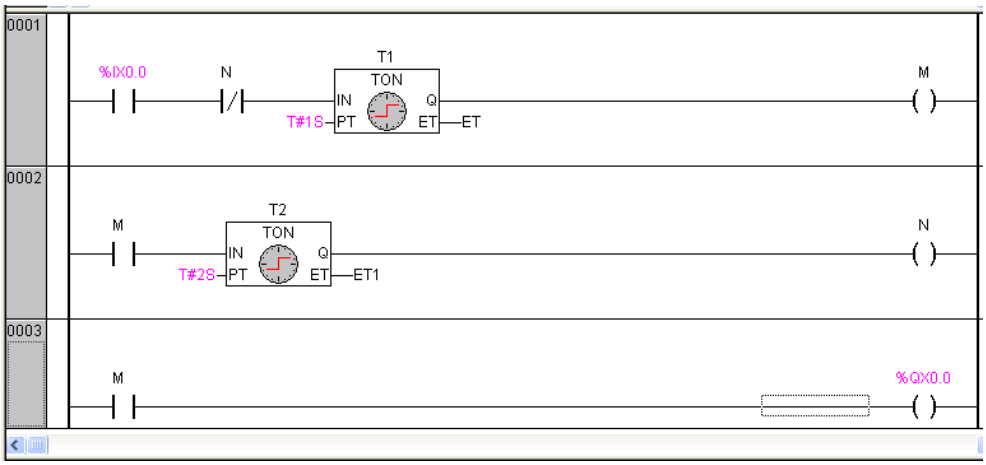


图 3-6-13 程序编写步骤（13）

为了使程序简单易读，一般会在程序中加入相应的注释。例如在 0001 节中，点击右键，选择“注释”，如图 3-6-14 所示。

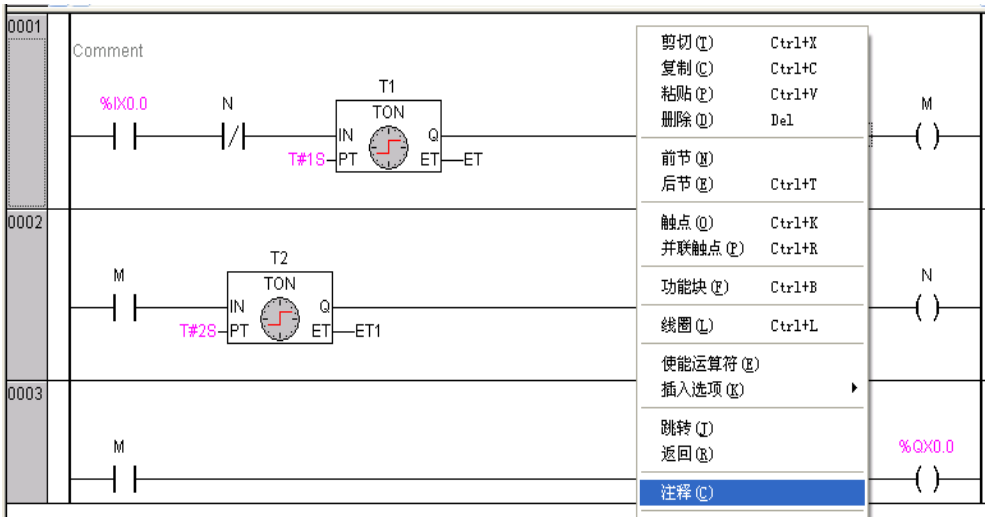


图 3-6-14 程序编写步骤（14）

双击“Comment”，输入相应的注释内容，如图 3-6-15 所示。

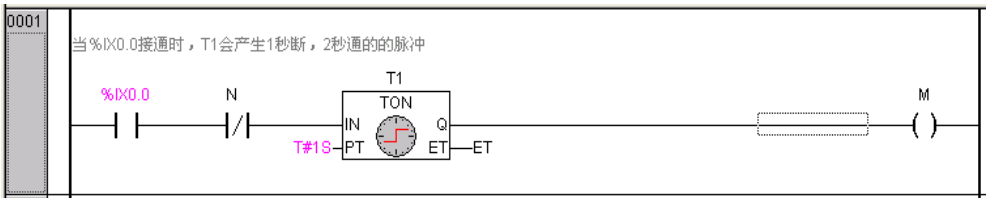


图 3-6-15 程序编写步骤（15）

注释添加完毕，如图 3-6-16 所示。

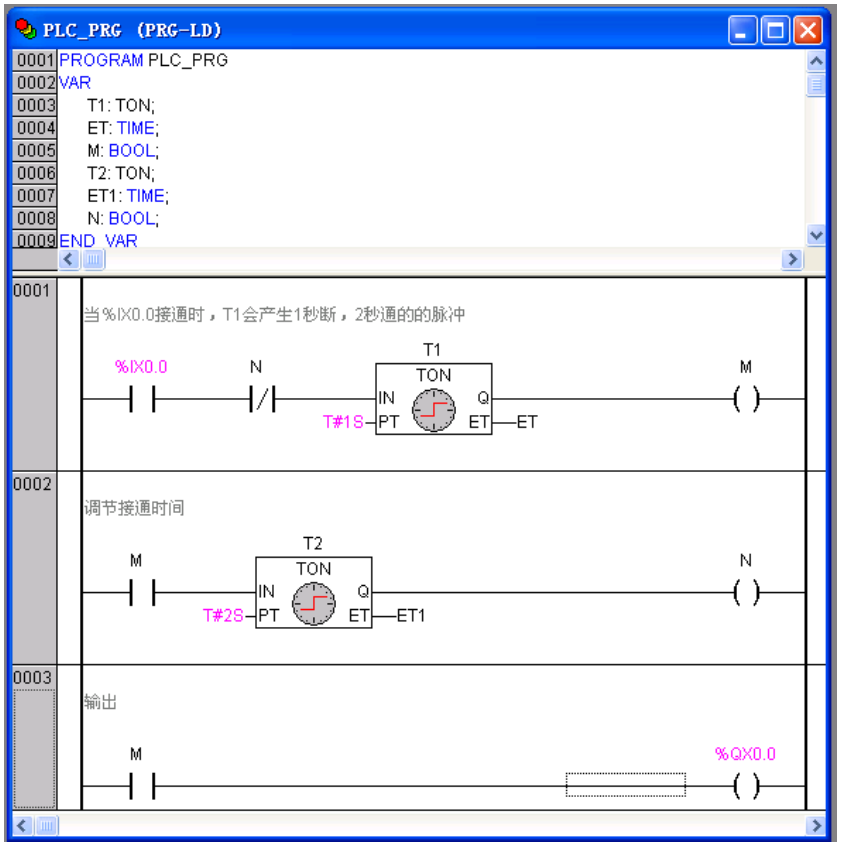


图 3-6-16 程序编写步骤（16）

3.7 编译

程序编写完毕，应该对其进行编译。打开“工程”菜单，选择“全部编译”，如图 3-7-1 所示。

图 3-7-1 编译过程（1）

则在消息窗口会显示如图 3-7-2 所示的信息。

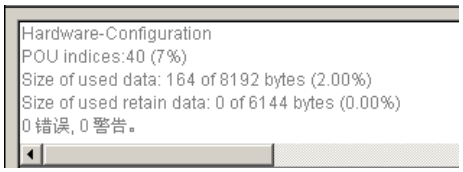


图 3-7-2 编译过程（2）

在“工程”菜单下选择“查看”/“未使用变量”，则会在消息窗口显示如图 3-7-3 所示的信息。“No unused variables found”表示程序中没有未使用的变量。

图 3-7-3 编译过程（3）

如果程序中出现了未使用的变量，如图 3-7-4 所示，如果有一个多余的变量 a，则会在消息窗口中显示信息：PLC_PRG(9)：a。这表示本程序中检查到了一个未使用的变量 a。

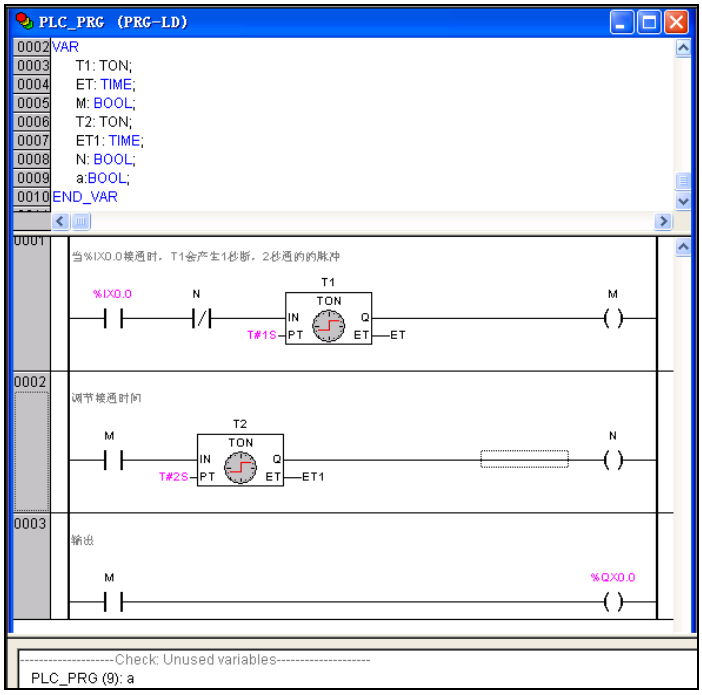


图 3-7-4 编译过程（4）

对于“查看”/“未使用变量”的应用，应该习惯性地在编译通过后进行一下自检。如果存在未使用的变量，建议删除未使用的变量，有助于程序的正确运行。

另外，也可以直接设置。在“资源”选项卡中，选择“工程选项”/“Build”/“自动检查”，选中“未用变量”，在编译时会自动检查未使用的变量。

关于编译，请参见 8.1 章节。关于查看未使用变量，请参见 8.2.3 章节。

编译结束后，执行“在线”/“登录”命令进入调试状态。调试分为在线调试和仿真调试两种方式。在后面的介绍中，将按照在线调试和仿真模式两种方式分别介绍程序的运行情况。

3.8 在线调试

程序下载到 PLC 的 CPU 模块中称为在线调试。编译通过的目标文件在进行下载时，会将全部目标文件下载到 CPU 模块中，同时将 CPU 模块复位，所有变量返回到初始状态。

在“在线”菜单中选择“登录”，建立本地计算机与 PLC 的 CPU 模块的连接，出现系统下

载提示信息，如图 3-8-1 所示。

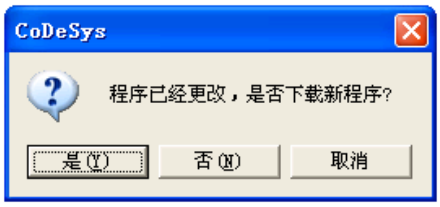


图 3-8-1 下载提示信息

点击“是”按钮，将新程序下载到 PLC 的 CPU 模块中。当出现如图 3-8-2 所示的创建启动工程提示信息时，点击“是”按钮，下载结束。确保 PLC 断电后再上电，运行此下载工程。

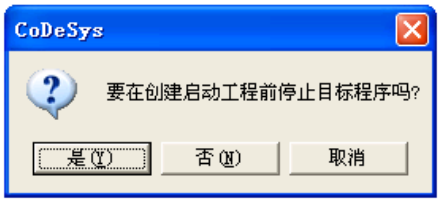


图 3-8-2 创建启动工程提示信息

下载结束后，程序并没有运行，需要手动设置 PLC 程序运行。在菜单栏中打开“在线”下拉菜单，选择“运行”，或者直接按“F5”功能键，均可实现程序的运行，如图 3-8-3 所示。

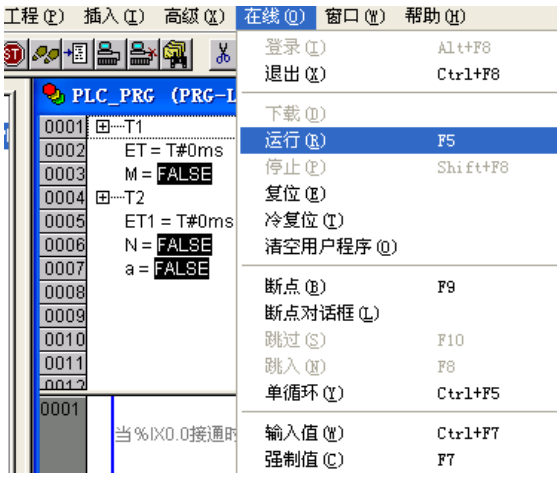


图 3-8-3 程序运行

双击“%IX0.0”，按“F7”功能键强制输入值后，使%IX0.0 闭合，程序开始运行，如图 3-8-4 所示。从运行的结果可见，%QX0.0 通道灯输出“1s 断 2s 通”的脉冲信号。

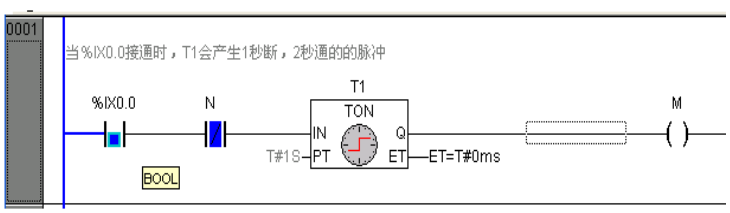


图 3-8-4 强制程序运行

程序运行情况如下所述。程序启动后，当运行时间 t<1s 时，%QX0.0 尚未闭合，PLC 上的通道灯%QX0.0 不亮，如图 3-8-5 所示。

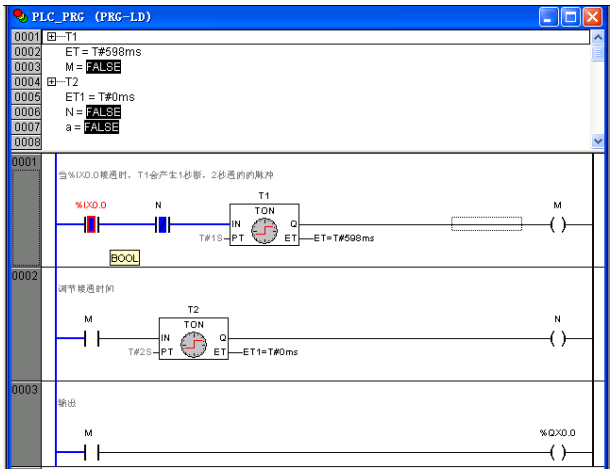


图 3-8-5 程序运行情况（1）

程序运行一段时间后，当运行时间 1s<t<3s 时，%QX0.0 闭合，PLC 上的第一个通道灯%QX0.0 亮，如图 3-8-6 所示。

当运行时间 3s<t<4s 时，触点“M”断开，PLC 上的通道灯%QX0.0 灭。如此反复，%QX0.0 通道灯会出现“1s 亮，并且持续 2s 后再灭，间隔 1s 后又亮”的现象。

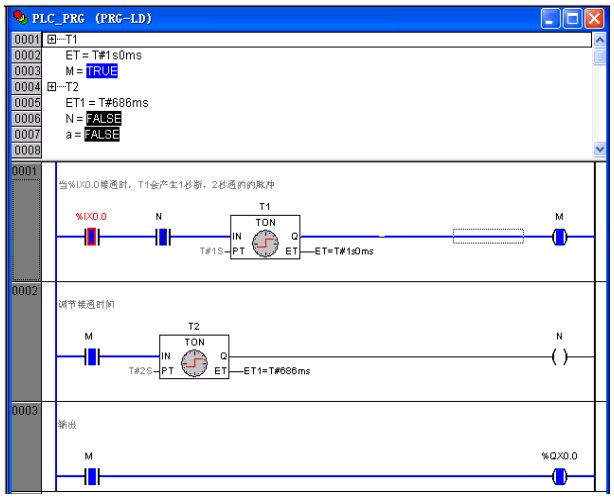


图 3-8-6 程序运行情况（2）

3.9 仿真调试

上述程序实现了在线模式下的运行。如果没有连接 PLC 的 CPU 模块，在本地计算机模拟运行用户程序，称为仿真模式。在菜单栏中打开“在线”下拉菜单，选择“仿真模式”，便进入了仿真模式下的程序运行过程。在“仿真模式”下，选择登录，如图 3-9-1 所示。“在线”/“仿真模式”被选中（出现“√”），登录时便会进入仿真模式。

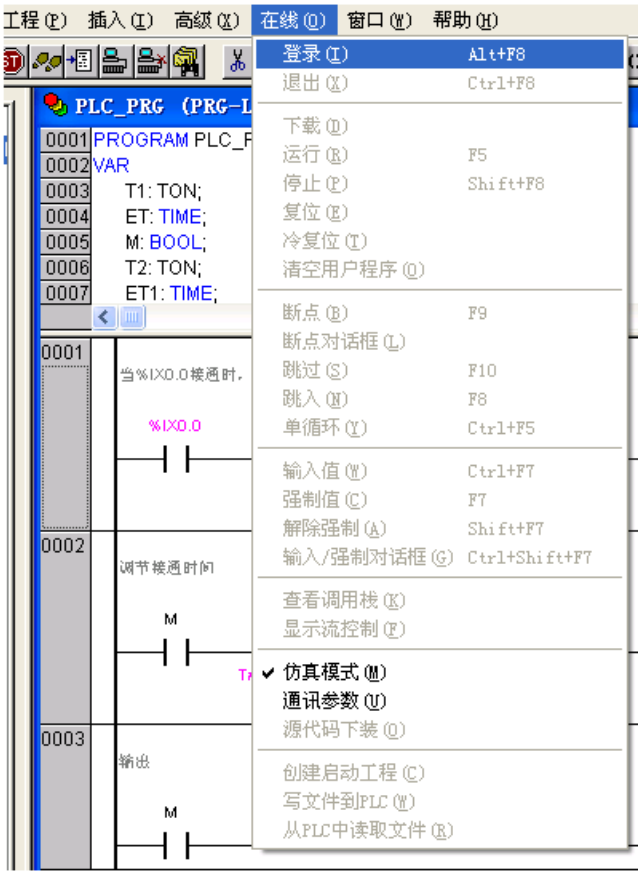


图 3-9-1 仿真模式

双击“%IX0.0”，按“CTRL+F7”组合功能键输入值，或按“F7”功能键强制值，使%IX0.0闭合，再按“F5”功能键，使程序运行，如图 3-9-2 所示。仿真模式下程序运行情况与在线模式下相同。

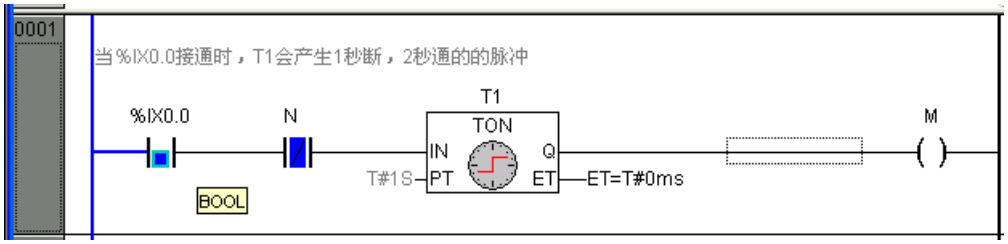


图 3-9-2 仿真模式下程序运行

关于在线调试和仿真调试的操作，请参见 8.4 章节。

第4章 存储区与变量

在编程之前，需要了解 LM 系列 PLC 是如何管理数据的。本章为您讲述 LM 系列 PLC 的存储区分配、数据寻址以及变量使用等概念，为编程打下良好的基础。

LM 系列 PLC 的数据存储区分为输入区（I 区）、输出区（Q 区）、M 区、N 区、掉电保持区共五类，每一类都有自己的特点和使用规则，4.1 章节讲述了各存储区的特点和使用方法。

4.2 章节讲述了地址的寻址方法和存储规则。在 LM 系列 PLC 的数据存储区中，输入区、输出区和 M 区属于寻址方式调用的数据区。

和高级语言类似，LM 系列 PLC 也有常量和变量的概念。所谓的常量就是数值不变的数。4.3 章节详细讲述了常量的分类和使用方法。

变量是 IEC61131-3 标准提出的概念。在程序运行时，其数值会改变的量就是变量。变量用于初始化、存储和处理用户数据。每个变量都有其固定的数据类型。变量的存储位置可由用户指定为 I 区、Q 区或 M 区的地址，亦可不指定地址，由系统自行分配。4.4 章节详细讲述变量的类型和使用方法。

LM 系列 PLC 在数据管理上功能非常强大。4.5 章节和 4.6 章节为您讲述了如何处理数组以及如何自定义数据类型。

4.1 存储区分配

LM 系列 PLC 的存储区，CPU 的内存被划分为不同的存储区，每一部分存储区都有各自的特点和使用规则。

存储区包括以下几大类：

输入存储区（I 区）

输入映像区。CPU 及扩展模块的数字量输入占用输入存储区地址，模拟量输入同样也占用输入存储区地址。另外，一些特殊功能，诸如以太网通讯或者 DP 通讯，也占用输入存储区的地址。I 区最大可存储 512 个字节。

输入存储区通过寻址方式访问，可以按位、字节、字、双字访问。具体访问方法请参见 4.2 章节。

输入存储区是只读的，并且不能掉电保持。在仿真模拟时，输入存储区的地址可以被输入，也可以被强制。但是在在线调试时，只能被强制。输入和强制是 PowerPro 在调试时改变数据的方式，请参见 8.4.9 和 8.4.10 章节。

输出存储区（Q 区）

输出映像区。CPU 及扩展模块的数字量输出占用输出存储区地址，模拟量输出同样也占用输出存储区地址。另外，一些特殊功能，诸如以太网通讯或者 DP 通讯，也占用输出存储区的地址。Q 区最大可存储 512 个字节。

输出存储区通过寻址方式访问，可以按位、字节、字、双字访问。具体访问方法请参见 4.2 章节。

输出存储区的数据是可读写的，并且不能掉电保持。在仿真模拟或者在线调试时，该数据区地址均可以被输入或强制。

M 存储区

M 存储区是 PLC 的中间寄存器区，用于存储和管理中间过程产生的数据或状态。无论

是位数据，还是字数据，均可以在 M 存储区实现。

M 存储区通过寻址方式访问，可以按位、字节、字、双字访问。LM 系列 PLC 的 M 区共有 8KB，按字节来寻址，M 存储区的范围为 MB0~MB8191。具体访问方法请参见 4.2 章节。

M 存储区的数据地址与输出存储区的性质一样，可读写，也可以被输入或强制。

M 存储区的地址中，部分是具有掉电保持功能的，包括 MB300~MB799。其余地址，均不具有掉电保持功能。

另外，要特别注意，M 存储区的前 100 个字节，即 MB0~MB99，是被系统用于自诊断的数据区，可以读取这些存储区的数据，但是不能写入。建议用户在编程时，从 MB100 开始使用。

N 存储区

N 存储区也属于 PLC 的中间寄存器区，用于存储和管理中间过程产生的数据和状态。与 M 存储区不同的是，N 存储区只能通过变量的方式来访问和调用。关于变量，手册在前几章都已提到过，关于变量的详细使用方法，请参见 4.4 章节。

N 存储区中的变量地址，是系统自动分配而用户无法指定的。N 区中的变量数据类型不单有位、字节、字和双字，还有 REAL、TIME、INT 等其他众多数据类型。另外，除了数据变量外，定义的功能块变量也存储在 N 存储区。关于功能块的概念，请参见 5.1 章节。

N 存储区大小为 24KB，即可以存储 24KB 的变量。N 存储区可以读写，可以被输入和强制。

N 存储区的数据是不能掉电保持的。

R 存储区

R 存储区属于掉电保持区，其调用方式与 N 区一致，也是通过变量的方式访问，无法指定地址。

R 存储区的大小为 6KB。R 存储区变量可以读写，可以被输入和强制。

变量定义时，假如没有选择保持功能，或者直接在局部变量中定义，则该变量存储在 N 区，若选择了保持功能或直接在保持型变量中定义，则该变量存储于 R 区，具有掉电保持功能。关于如何定义掉电保持区变量，详细说明请参见 4.4.6 章节。

> 注意：
>
> LM 系列 PLC 有两种方式可以实现数据的掉电保持。一是采用地址方式，选择 M 区的 MB300~MB799 之间的地址即可；另一种采用变量的方式，将变量定义为掉电保持区变量即可。

4.2 地址寻址方式

4.2.1 地址存储映射关系

LM 系列 PLC 的 I 区、Q 区和 M 区是按地址寻址方式访问的，这些存储区都有唯一的、明确的地址。用户可以通过地址寻址方式，即直接使用该存储区地址的方式，来读取和设置该存储区的值。

在介绍访问规则之前，需要先了解这些存储区的存储格式。I 区、Q 区和 M 区的存储格式是一致的，所以这里以 M 区为例说明，如图 4-2-1 所示。

LM 系列 PLC 所有直接寻址的存储区，是按字节存储的。

每个字节包含 8 个位，每 8 个位组成一个字节。每 2 个字节组成一个字，每两个字组成

一个双字。

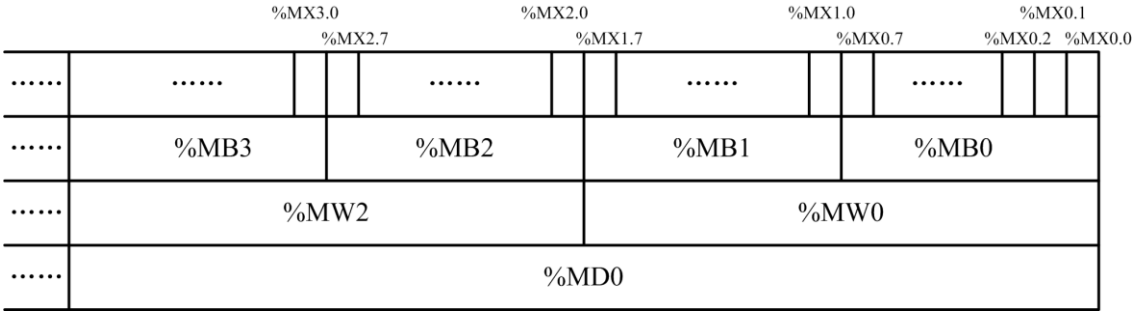


图 4-2-1　M 区存储格式

诸如：%MX2.7 表示%MB2 的第 8 个位。

%MW12 表示有%MB12 和%MB13 两个字节组成的字。

这些存储区的地址，可以按位、字节、字或双字来访问，数据类型为 BOOL、BYTE、WORD 和 DWORD。要了解这些数据类型的范围，请查看 4.4.2 章节。假如需要使用 INT 或者 REAL 等其他数据类型，需要使用变量方式访问，具体内容也请参见 4.4 章节。

要注意，按不同数据类型访问，数据存储区可能是重叠的。诸如%MW0 的数值为 3，则%MB0 的值也为 3，%MX0.0 的值为 TRUE，%MX0.1 的值也为 TRUE。又或者在程序中强制%MX0.0 为 TRUE，因为%MX0.0 是%MB0、%MW0 和%MD0 的第一个位，则同时，这些存储地址的值也被强制为 1。

对于 I 区和 Q 区，也是同样的存储方式。I 区和 Q 区的地址和实际 DI/DO 点如何对应，请参见 7.3 章节。

注意：

按字寻址方式访问，因为一个字由两个字节组成，其数字必为偶数，不能为奇数。如：%MB0 和%MB1 组成一个字%MW0，下一个字则为%MW2，而不是%MW1。%MW1 是无效的地址。按双字寻址也遵守此规则。

4.2.2 地址访问格式

按 IEC61131-3 标准，所有的直接地址都从“%”开始。以 M 区为例，如表 4-2-1 所示。

表 4-2-1 地址访问格式

位寻址	格式	%MXm.n
	描述	X 表示是按位寻址； m 表示在 M 存储区中的字节编号； n 表示位于该字节的第几位，范围为 0~7
	数据类型	BOOL
	示例	%MX0.3、%MX100.0、%MX3212.7

字节寻址	格式	%MBm
	描述	B 表示按字节寻址 m 表示在 M 存储区中的字节编号
	数据类型	BYTE
	示例	%MB103、%MB2000
字寻址	格式	%MWm
	描述	W 表示按字寻址 m 表示在该字存储单元中的首字节的地址，注意 m 必须为偶数
	数据类型	WORD
	示例	%MW150、%MW3000
双字寻址	格式	%MDm
	描述	D 表示按双字寻址 m 表示在该双字存储单元中的首字节的地址，注意 m 必须为偶数
	数据类型	DWORD
	示例	%MD300、%MD432

对于 I 区、Q 区，则把表中的 M 替换为 I 或 Q 即可。

表 4-2-2 为这三个数据区的范围，超过这个范围的地址视为无效的地址。

表 4-2-2 数据存储区及范围

存储区	**范围（按字节）**
I 存储区	%IB0~%IB511（最大为 512 字节，具体大小根据 PLC 确定）
Q 存储区	%QB0~%QB511（最大为 512 字节，具体大小根据 PLC 确定）
M 存储区	%MB0~%MB8191

再次强调的是 M 存储区大小为 8KB，即从%MB0~%MB8191。其中，%MB0~%MB99 作为 PLC 的内部诊断区，建议用户不要使用。%MB300~%MB799 具有掉电保持功能，其余存储区不具有掉电保持功能。

4.3 常量

在 PLC 编程的时候，可能经常要用到一些数值不变的参数，诸如定时器的时间、换算的比例参数等，这些数值不变的参数称为常量。LM 系列 PLC 支持多种数据类型的常量，常见的常量：布尔型、时间型、数字型等，如表 4-3-1 所示。

表 4-3-1 常量的分类及表示方法

常量类型	**表示方法**	
布尔型	描述	布尔常量只有两个：逻辑值 TRUE 和 FALSE（也可表示为 1 和 0），TRUE 等价于 1，FALSE 等价于 0。
	示例	TRUE、0

整数型	描述	数字常量的数值可以是二进制、十进制、八进制和十六进制。如果整数值不是十进制值，可以用“进制”加符号“#”放在整数值前面来表示。十进制的10至15在十六进制中表示为A至F。
	示例	14 (*十进制数 14*) 2#1001_0011 (*二进制数 1001_0011*) 8#67 (*八进制数 67*) 16#AE (*十六进制数 AE*)
实数型	描述	实数常量用十进制小数和指数来表示，遵循标准的科学计数法格式。实数常量的数据类型是 REAL。
	示例	7.4 (*实数 7.4*) 1.64e+009 (*实数 1.64e+009*)
时间型	描述	时间常量一般用来操作时间，由“T#”（或“t#”）加上“时间值”构成，时间值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms）。注意，它们的正确顺序为d、h、m、s、ms。
	示例	T#18ms (*18 毫秒的一个时间常量*) T#100s12ms (*100 秒 12 毫秒的一个时间常量，高单位允许超限*) t#12h34m15s (*12 小时 34 分 15 秒的一个常量*) 下面是错误的时间常量： t#5m68s (*低单位不允许超限*) 15ms (*没有 T# *) t#4ms13d (*顺序错误*)
时刻型	描述	时刻常量用于存储当前时刻，由“TOD#”（“tod#”、“TIME_OF_DAY#”或“time_of_day#”）加上“时刻值”构成。时刻值的格式为：小时:分钟:秒（可以用实数形式输入秒）。
	示例	TOD#00:00:00 (*时刻常量为 0 点 0 时 0 分*) TIME_OF_DAY#15:36:30.123 (*时刻常量为 15 点 36 分 30.123 秒*)
日期型	描述	日期常量由“D#”(“d#”、“DATE#”或“date#”)加上“日期值”构成。
	示例	DATE#2005-05-06 (*日期常量 2005 年 5 月 6 日*) d#1980-09-22 (*日期常量 1980 年 9 月 22 日*)
日期时刻型	描述	日期常量和时刻常量合并起来称为日期时刻常量，由“DT#”（“dt#”、“DATE_AND_TIME#”或“date_and_time#”）加上“日期时刻值”构成。
	示例	DT#1980-09-22-15:45:18 (*时刻日期常量为 1980 年 9 月 22 日 15 点 45 分 18 秒*) date_and_time#2001-03-09-00:00:00 (*时刻日期常量为 2001 年 3 月 9 日 0 点 0 分 0 秒*)
字符串型	描述	字符串常量在两个单引号之间，可以包含空格和特殊字符。
	示例	'Abby and Craig' (*字符串 Abby and Craig *) ':-)' (*字符串:-)*)

注意：

PowerPro 不区分大小写，诸如 T#3s 和 t#3s 属于同一常数，TRUE 和 true 均可以表示布尔型常量。

4.4 变量

所谓变量，就是用字母、数字和下划线组成的一个标识符。

按照数据类型的不同，变量可以分为标准类型和用户自定义类型。其中标准类型包括布尔型（BOOL）、整型（INT）、实型（REAL）、字符串型（STRING）以及时间型（TIME）等。自定义类型包括结构体（STRUCT）和枚举(ENUM)。

按照使用范围的不同，变量可以分为全局变量和局部变量。局部变量只在整个工程的一部分程序中有效，其它程序不能引用。全局变量则可以被整个工程的任意程序引用，在整个工程中均有效。

按照属性的不同，变量分为中间变量、输入型变量、输出型变量、输入输出型变量等。

按照能否掉电保护，变量分为保持型变量和非保持型变量。

4.4.1 变量命名规则

变量命名必须遵循如下的规则：

- 必须以一个字母或者单一的下划线开始，随后是一定数量的字母、数字或下划线。
- 字母与大小写无关，ABC 和 abc 被认为是同一个变量。
- 关键字不能用于变量名。PowerPro 定义了一些关键字，关键字是标准的标识符，其作用和命名已在系统中自动定义，PowerPro 的关键词如表 4-4-1 所示。

表 4-4-1 关键词表

ARRAY	FUNCTION	TYPE
AT	FUNCTION_BLOCK	VAR
CONSTANT	OF	VAR_ACCESS
END_FUNCTION	PERSISTENT	VAR_CONFIG
END_FUNCTION_BLOCK	PROGRAM	VAR_EXTERNAL
END_PROGRAM	READ_ONLY	VAR_GLOBAL
END_STRUCT	READ_WRITE	VAR_IN_OUT
END_TYPE	RETAIN	VAR_INPUT
END_VAR	STRUCT	VAR_OUTPUT

4.4.2 变量数据类型

PowerPro 软件支持的变量数据类型包括标准类型和用户自定义类型。在 PLC 软件中可以查到所支持的标准类型，具体方法如下：选择“编辑”菜单下的“输入变量”，弹出“Help Manager”对话框，再用鼠标左键单击“Standard Types”即可。关于自定义数据类型，请参见 4.6 章节。

PowerPro 支持的标准数据类型及范围，如表 4-4-2 所示。

表 4-4-2 变量数据类型

类型	类型名称	数据下限	数据上限	存储空间	备注
BOOL	布尔型	0	1	1bit	
BYTE	字节型	0	255	8 Bit	
WORD	字型	0	65535	16 Bit	
DWORD	双字型	0	4294967295	32Bit	

SINT	短整型	-128	127	8 Bit	
USINT	无符号短整型	0	255	8 Bit	
INT	整型	-32768	32767	16 Bit	
UINT	无符号整型	0	65535	16 Bit	
DINT	长整型	-2147483648	2147483647	32 Bit	
UDINT	无符号长整型	0	4294967295	32 Bit	
REAL	实数型	-3.402823E+38	3.402823E+38	32Bit	单精度浮点数
TIME	时间型			32Bit	示例： Time1 : TIME := t#3s;
TOD	时刻型				示例： Tod1 ： TOD ： = TOD#00:00:00；
DATE	日期型				示例： Date1 ： DATE ： = D#2008-8-8;
DT	日期时刻型				示例： DT1 ： DT ： = dt#2008-08-08-20:08:08;
STRING	字符串型				示例： Str:STRING(35):=‘hi’;
ARRAY	数组				示例： Arr1:ARRAY[1..5]OF BYTE:=1,2,3,4,5;

4.4.3 变量定义

在使用变量之前，必须先对变量进行定义。PowerPro 针对变量不同的功能，规定了不同的变量类型。在定义变量时，不单要定义数据类型，还要定义变量类型。

在 PowerPro 中，变量可以被定义为很多类型，如：全局变量、局部变量、输入变量、输出变量等。具体变量类型如表 4-4-3 所示。

表 4-4-3 变量类型

变量类型	数据类型
VAR	局部变量，仅在该程序中使用。在其余程序中可以定义相同名称的变量，被认为是两个变量。
VAR_INPUT	输入变量。当调用程序时，输入变量用于实现调用程序时的参数传递。在调用程序时，可以将参数通过输入变量传递至子程序或其余 POU 中，具体参见 5.3 章节。
VAR_OUTPUT	输出变量。当调用程序时，输出变量用于实现调用程序时的参数传递。在调用程序时，可以将参数通过输出变量传递至调用该 POU 的程序中，具体参见 5.3 章节。
VAR_IN_OUTPUT	输入/输出变量。VAR_INPUT 和 VAR_OUTPUT 变量的组合。同样用于参数传递。
VAR_GLOBAL	全局变量。若该变量定义为全局变量，则在任何程序中均可使用该变量。同时，不能再定义名称相同的变量。

VAR、VAR_INPUT、VAR_OUTPUT、VAR_IN_OUTPUT、VAR_GLOBAL 是用于标识变量类型的关键词。定义时根据可需要对类别进行选择，

变量的声明有两种方式：自动声明和手动声明。

自动定义变量

系统支持变量自动定义功能。当程序中出现一个新变量时，系统会自动弹出对话框，要求进行变量定义，如图 4-4-1 所示。其中类别、名字和类型是必须的。

图 4-4-1 自动定义变量

自动定义变量对话框的各项含义，如下所述：

- 类别：类型选择。各类型区别请参见表 4-4-3，如：希望定义的变量在所有的 POU 中都能使用，则定义为全局变量，选择类型 VAR_GLOBAL。
- 名字：声明变量的名称，即标识符。关于变量命名的规则，请参见 4.4.1 章节。
- 类型：数据类型选择。可以直接在输入框中输入，也可以点击按钮，然后在弹出的对话框中选择数据类型。各类数据类型请参见 4.4.2 章节。
- 符号表：只有在“类别”选择“VAR_GLOBAL”时，符号表才可选。默认选项为“Global_Variables”。当定义一个全局变量时，在“资源”选项卡中打开“全局变量”文件夹，可以看到“Global_Variables”，双击“Global_Variables”，刚定义的全局变量便显示在这里，如图 4-4-2 所示。
- 初始值：变量的初始值。这里可以填入一个与变量数据类型对应的常量，完成变量的初始化。
- 地址：定义变量的地址。
- 注释：变量的含义。

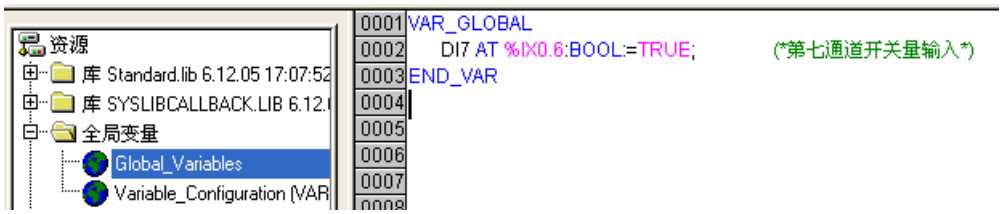


图 4-4-2 声明变量的代码部分

在自动定义变量时，需要注意以下几点：

- 变量可以被指定一个地址，地址的格式与 4.2 章节所讲述的一致。当变量指定地址时，变量存储于该地址所指定的数据区，如图 4-4-1 所示的变量定义，则该变量 temp 存储于 M 存储区，与%MW100 占用同一存储区地址。在程序中，采用直接寻址方

式改变%MW100 的值，则该变量 temp 相应改变。变量定义时，也可以不指定地址，则该变量存储于 N 存储区。

在定义时，可以设置变量的初始值，初始值是一个常量，其类型应与变量的类型一致。诸如定义一时间类型变量，则初始值应是一个时间常量，例如 t#5s。定义初始值后，当 PLC 在上电瞬间，变量被赋值为初始值。

自动定义变量后，将会在变量声明部分会显示刚定义的变量的声明。若定义了图 4-4-1 所示的变量，则在变量声明区有如下声明：

```
PROGRAM    PLC_PRG
VAR
        Temp AT %MW100: WORD := 30;        (*温度参数*)
END_VAR
```

若定义的变量是全局变量，则会显示在资源选项中的全局变量中，而不是在变量声明区。

- 变量自动定义时，在自动定义对话框右下角有两个选项：常量和保持。当选择常量，则将该变量作为一个常量，程序中无法再改变其数值。当选择保持型变量时，表示将该变量设置为具有掉电保持功能，该变量存储在 R 存储区。
- 新建变量时，系统可以自动定义。但当变量被删除时，定义语句不会自动删除，继续保留在编辑器中，因此要注意变量不能定义重复。可使用“工程”/“查看”/“未使用变量”命令查找到这些无用的变量声明。具体使用方法请参见 8.2.3 章节。

手动定义变量

所谓的手动定义变量，就是不通过自动定义对话框进行定义，而是手动在变量声明区按变量声明的格式和规定添加变量。

变量声明的一般格式：

<标识符> {AT<地址>} : <数据类型> {: = <初始值>};

其中在{ }中的部分是可选的。

定义不同类型的变量，需要在不同的位置进行定义。诸如：定义局部变量，需要在 VAR 和 END_VAR 之间定义，而定义输入变量，需要在 VAR INPUT 和 END_VAR 之间定义。

变量声明区也可以定义为表格形式。在“工程”/“选项”/“Editor”对话框里选中“声明为表”项，或在程序编辑区选中右键菜单项“定义为表格”，声明编辑器会显示成表格的形式，如图 4-4-3 所示。

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN | INFO

	名称	地址	类型	初始值	注释
0001	AO1		Analog_OUT		
0002	DP		DP_Slave		
0003	xunhuan		BOOL		
0004	TP1		TP		
0005	TP2		TP		
0006	TON_1		TON		
0007	TON_2		TON		
0008	TON_3		TON		
0009	TON_4		TON		
0010	TON_5		TON		

图 4-4-3 表格声明变量

在表格形式的声明编辑器中，在最初设置时，“名称”字段缺省为“Name”，“类型”字段缺省为“BOOL”。根据需要可以自行更改这些设置，变量名和类型是完成声明变量所必需的选项。

去除右键命令“表格声明”前的“√”，可切换至常规形式。

表格声明变量以简洁的填表方式代替了常规的语句声明方式，建议初学者使用自动变量定义或表格定义。

变量调用和地址调用方式的区别

当采用“变量+地址”方式定义变量，与直接地址调用类似，但是两者还是有区别的。直接地址调用的数据类型可为 BOOL、BYTE、WORD、DWORD 等类型，而用“变量+地址”的方式调用，可定义的数据类型比直接地址调用方式多。

例如：需要定义一 REAL 型变量，其地址为%MD100。若直接使用地址%MD100，其数据类型为 DWORD 型而不是 REAL 型。此时就需要用“变量+地址”的方式定义一个数据类型为 REAL 型的变量，地址为%MW100（只需填入初始地址，长度根据数据类型自动判断），从而实现了在%MD100 上定义一个 REAL 型变量。

4.4.4 保持型变量

在很多工程中，通常需要数据具有掉电保持功能，使 PLC 断电后数据不丢失。在定义变量时，可以直接定义变量为保持型变量，此时变量自动保存于 R 存储区，具有掉电保持功能。

在自动定义时，在自动定义对话框的右下角，选择“保持”选项，变量就自动定义为保持型变量。在手动定义时，将变量定义在 VAR_RETAIN 和 END_VAR 之间，也可以定义该变量为掉电保持变量。

前面讲到存储区（4.1 章节）时曾提到，M 区的部分地址（%MB300~%MB799）同样具有掉电保持功能。将变量定义为保持型变量与将数据放入地址在%MB300~%MB799 的存储区内，具有相同的效果。

4.4.5 指针变量

指针变量是一类特殊的变量。LM 系列 PLC 中，所有的存储区都占用 CPU 的地址，这个地址被称为绝对地址。不管是 I 区、Q 区、M 区还是 N 区和 R 区，都占用 CPU 的一部分地址。而 I 区、Q 区和 M 区通过寻址访问的地址，诸如%MW100，可以认为是相对地址。指针是指数据区的绝对地址而不是相对地址。

相邻的两个相对地址，其绝对地址也是相邻的。因为 LM 系列 PLC 的存储区是按字节存储的，因此假如%MW100 的绝对地址为 pt，则%MW102 的绝对地址为 pt+2。%MW200 的绝对地址就是 pt+100。

在编程时，假如利用指针的这种关系，可以很方便地实现一些比较复杂的功能。在使用之前，同样需要定义指针变量。

指针变量的定义与其他数据类型定义类似，只是其数据类型为 POINTER TO <数据类型>

指针定义的语法格式：

<指针名> : POINTER TO <数据类型/功能块>;

示例：

```
pt:POINTER TO INT;      (*定义一个整型数据的指针 pt*)
Var_int1:INT := 5;      (*定义整型变量 Var_int1，使其等于 5*)
```

```
Var_int2:INT;              (*定义整型变量 Var_int2*)
pt := ADR(Var_int1);       (*取出 Var_int1 变量的地址，将地址值赋给 pt*)
Var_int2:= pt^;            (*将指针 pt 所指地址的值赋给 Var_int2，即 Var_int2=5*)
```

举例说明指针的用法：
在 M 数据区的%MW100 开始的地址中，存放了 100 个 WORD 型变量，现在需要将这些值转移至%MW300 开始的 100 个字内，使其具有掉电保持功能。
因为数据区比较长，采用指针方式，可以很方便地实现数据区的转移。
这里需要定义两个指针变量 pt1 和 pt2，一个指向%MW100，另一个指向%MW300。这里还需要用到一个取地址指令 ADR 和读取指针数值指令^，关于这两个指令的详细信息，请参见《指令手册》。

变量定义如下：

```
VAR
    m: INT;
    pt1: POINTER TO WORD;
    pt2: POINTER TO WORD;
END_VAR
```

其中变量 m 用于传输 100 个字节。
在这里采用 ST 方式编写程序，关于 ST 语言编程，请参见 9.3 章节。
采用一个 FOR 循环语句，每次循环，pt1 和 pt2 的值均加 2（因为是 WORD 类型指针），然后将 pt1 的值赋值给 pt2 即可。
具体程序如下：

```
pt1:=ADR(%MW100);
pt2:=ADR(%MW300);
FOR m:=1 TO 100 BY 1 DO
pt2^:=pt1^;
pt1:=pt1+2;
pt2:=pt2+2;
END_FOR
```

4.5 数组

PowerPro 对数据的管理功能是非常强大的。他不但支持多种数据类型，也支持多维数据。在编程时，可以根据基本数据类型来定义一维、二维和三维数组。数组可以采用自动定义，也可以在变量声明区手动定义。

数组的标识符为 ARRAY。数组定义的语法格式：

<数组名> : ARRAY [<L1>..<U1>, <L2>..<U2>, <L3>..<U3>] OF <基本数据类型>;

其中 L1、L2 和 L3 表示字段范围的最小值，U1、U2 和 U3 表示字段范围的最大值。字段范围必须是整数。假如是一维数组，则只需设置 L1 和 U1 即可；假如是二维数组，则需要设置 L1、U1 和 L2、U2；假如是三维数组，则 L1、U1、L2、U2 和 L3、U3 均需定义。

下图为定义一个数组元素数量为 10 的一个一维数组的自动定义对话框：

图 4-5-1　数组自动定义

在数组定义的同时，给数组中的元素赋值称为初始化数组。在数组定义时，可以初始化数组中所有元素，也可以不进行初始化。

✧　举例：定义数组

Card_game: ARRAY [1..13, 1..4] OF INT; (*定义一个整型的二维数组 Card_game*)

✧　举例：数组的完全初始化

Arr1:ARRAY [1..5] OF BYTE:= 1,2,3,4,5;

Arr2:ARRAY [1..2,1..2] OF INT := 1,3(7) ;　　（*即 1,7,7,7 的缩写形式*）

Arr3:ARRAY [1..2,1..2,1..2] OF INT := 2(0),4(4),2,3;（*即 0,0,4,4,4,4,2,3 的缩写形式*）

✧　举例：数组的部分初始化

Arr1:ARRAY [1..10] OF BYTE:= 1,2;

对于那些没有预先赋值的元素，按照基本数据类型的缺省初始值进行初始化。在此例中，元素[3]到[10]被初始化为 0。

4.6 自定义数据类型

在一些实际应用场合，需要用到一些配方数据。每组配方包含很多参数，而这些参数的数据类型是不同的，诸如可能包含 REAL 型、WORD 型或 TIME 型等，那么这里使用自定义数据类型就能非常方便地实现配方数据的管理。

在 2.5.2 节曾经提到过自定义数据类型。在对象组织器的数据类型选项卡中，就可以自定义数据类型。

首先，点击右键，新建一数据类型，命名规则同变量命名规则。命名完之后，该标志符就可以作为一个结构用来表示这个数据类型。

结构变量以关键字 TYPE 和 STRUCT 开始，关键字 END_STRUCT 和 END_TYPE 结束。

定义结构变量的语法格式：

TYPE <结构名>:

STRUCT

　　<变量声明 1>

　　<变量声明 2>

...
<变量声明 n>
END_STRUCT
END_TYPE
结构是一种可以在整个工程中被识别的数据类型，而且可以象引用标准数据类型一样引用结构。唯一的限制是，结构变量不能指定地址，即不允许进行 AT 声明。
举例：定义名为 Polygonline 的结构

```
TYPE Polygonline:
STRUCT
    Start:ARRAY [1..2] OF INT;
    Point1:ARRAY [1..2] OF INT;
    Point2:ARRAY [1..2] OF INT;
    Point3:ARRAY [1..2] OF INT;
    Point4:ARRAY [1..2] OF INT;
    End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

举例：初始化结构

```
P1:Polygonline:=(Start:=3,3,Point1:=5,2,Point2:=7,3,Point3:=8,5,Point4:=5,7,End:=3,5);
```

举例：结构数组的初始化

```
TYPE STRUCT1:
STRUCT
    p1:int;
    p2:int;
    p3:dword;
END_STRUCT
END_TYPE
A1[1..3] OF STRUCT1:=(p1:=1,p2:=10,p3:= 3),(p1:=2,p2:=0,p3:=2),(p1:=4,p2:=5,p3:=1);
```

访问结构成员的语法：
<结构名>.<结构成员名>
举例
如果结构名为 Week，其中的一个成员名为 Monday，则可以用下面的形式访问该成员：
Week.Monday

第5章 程序组织单元（POU）

本章主要讲述了 PowerPro 的一个重要概念—程序组织单元（POU）。程序组织单元是组成工程的基本结构。任何复杂的工程都是由众多 POU 组成的，POU 包括程序、功能块和函数。

5.1 章节讲述了 POU 的一些基本概念，使得读者对 POU 的概念有一定的了解。

5.2 章节讲述了 POU 的建立，5.3 章节讲述了 POU 之间的调用，这两节重点讲述了 POU 的使用规则。5.4 章节讲述了 PowerPro 软件如何管理 POU。

在本章的学习过程中，若遇到一些概念和规则不易理解，建议初学者可以跳过这些内容，同时在学习过程中采用“边学习、边练习”的方式，这样有助于更快地理解本章的内容。同时建议在学习后面的内容时也采取这种方法。

5.1 POU 的基本概念

POU 是程序组织单元（Program Organization Unit）的简称。POU 可以是函数（Function）、功能块（Function Block）或程序（Program）。其中程序和功能块的编程语言可以是 LD、FBD、IL、ST、SFC 及 CFC。函数的编程语言可以是 LD、FBD、IL、ST 及 CFC，但不能是 SFC。关于 POU 的编程语言，请参见第九章。

5.1.1 POU 的类型

POU 分为程序（Program）、功能块（Function Block）和函数（Function）等三种类型。

- 程序（Program）
  程序是为了完成某项任务而编写的语句序列，是一组指令的集合。程序是唯一可执行的 POU，是逻辑执行的主体。程序可以通过任务组态来激活，也可以通过其它程序来调用。
- 功能块（Function Block）
  功能块是预先编好的、实现某种运算的程序。功能块本身不能单独执行，只能由程序调用功能块执行。在执行时，输入量可以是一个或多个值，输出量可以是一个或多个执行结果。与函数不同，功能块本身没有返回值。
- 函数（Function）
  函数也是预先编好的、实现某种运算的程序。函数在执行时，会针对一系列特定的输入，产生一个输出结果，这个输出结果被赋给函数本身，称为返回值。函数只能被其它 POU 调用，函数本身不能单独执行。

5.1.2 POU 的调用

POU 的调用有两种方法。

- 被其它已经调用的 POU 来调用。
- 通过任务配置来调用，这种方法仅限于程序调用。当程序中没有进行任务配置时，系统会自动调用主程序 PLC_PRG。

POU 的调用要遵循以下原则，如图 5-1-1 所示。

- 程序可以调用函数、功能块和其它程序。
- 功能块可以调用函数和其它功能块。
- 函数可以调用函数。

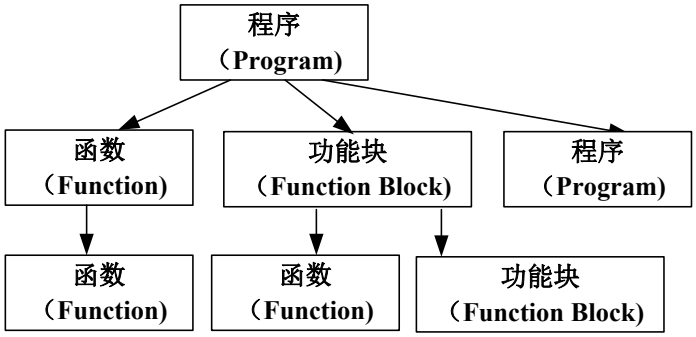


图 5-1-1 POU 的调用

5.1.3 POU 的组成

POU 包含一个声明部分和一个代码部分（程序区）。用户创建 POU 时，必须编辑这两部分。

- 声明部分：在变量区创建、显示 POU 变量。用户可在引用变量之前进行定义，也可以在引用时利用变量定义对话框定义。无论是局部变量还是全局变量，在变量定义过程中必须遵循一定的格式，具体的格式参见 4.4 章节。
- 代码部分：在程序区创建，是 POU 的主体，用户可以选用 IEC 标准编程语言来编写。

5.1.4 主程序 PLC_PRG

程序 PLC_PRG 被默认为主程序，是一个特殊的 POU。每个工程必须包含这个主程序才能正常运行。系统默认每个控制周期调用一次这个 POU，不需要进行额外的任务组态。所以，工程必须以 PLC_PRG 为主程序，通过它来实现对其它 POU 的调用。

5.2 创建 POU

5.2.1 创建程序

在对象组织器中选中“程序”选项卡，在程序列表中点击右键，弹出管理 POU 菜单，管理 POU 菜单的详细介绍详见 2.5.1 章节。选择“添加”，如果列表中无 POU，则新 POU 名默认为“PLC_PRG”。在弹出的“创建 POU”对话框中，选择“POU 类型”为“程序”；“POU 语言”可以选择 IL、LD、FBD、SFC、ST、CFC 之一；“新 POU 名”为程序名，如输入 Fct，名字尽量采用能反映其实际功能的字符，便于识别；点击“确认”，便创建了程序“Fct（PRG）”。如图 5-2-1 所示。

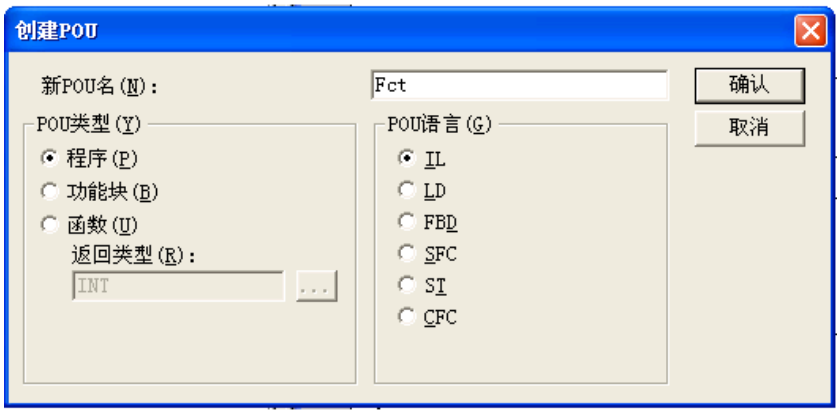


图 5-2-1 创建程序

5.2.2 创建功能块

功能块的创建和程序类似。在对象组织器中选中“程序”选项卡，在程序列表中点击右键，选择“添加”。在弹出的“创建 POU”对话框中，选择“POU 类型”为“功能块”，然后进行编程语言的选择和命名即可。

5.2.3 创建函数

函数的外形结构与功能块类似，区别是函数只有一个输出端，以下说明如何创建函数。

在对象组织器中选中“程序”，选择“添加”。在弹出的“创建 POU”对话框中，选择“POU 类型”为“函数”，“返回类型”、“POU 语言”和“新 POU 名”可根据需要选择和输入，如图 5-2-2 所示。如果新 POU 名为“Fct”，则“Fct”即为函数名。通常，命名要尽量采用能反映其实际用途的字符，便于识别。

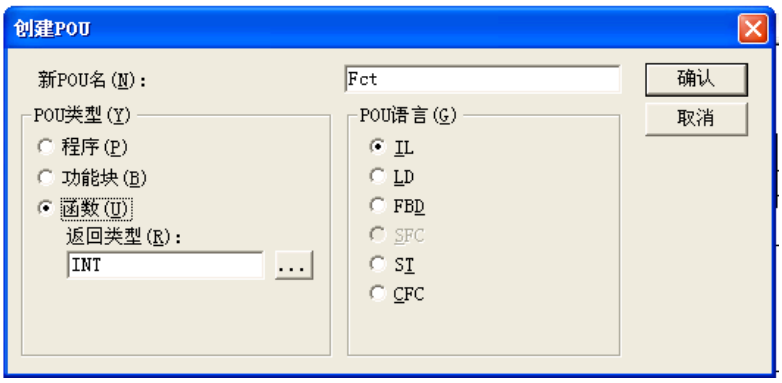


图 5-2-2 自定义函数（1）

设置好后，自动生成函数结构体，可以在里面添加所需要的内容，如图 5-2-3 所示。

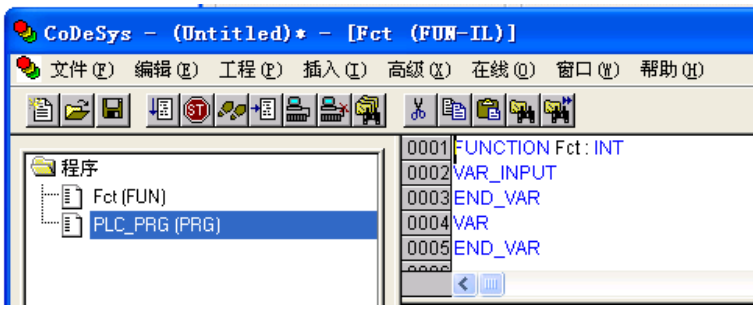


图 5-2-3 自定义函数（2）

下面以制作一个数学计算式为例来介绍函数的编写，如图 5-2-4 所示。从 VAR_INPUT 到第一个 END_VAR 之间用于定义输入变量，从 VAR 到第二个 END_VAR 之间用于定义中间变量，下方窗口进行函数算法编写。

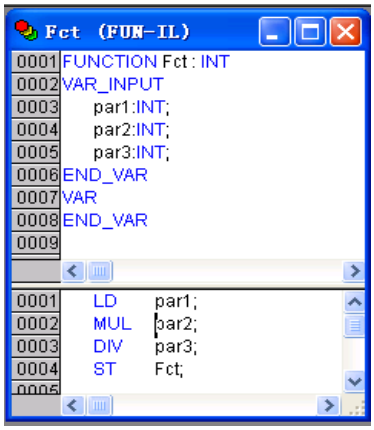


图 5-2-4 自定义函数（3）

编写完毕保存。编译通过后，可以在其它程序中直接调用，如图 5-2-5 所示。

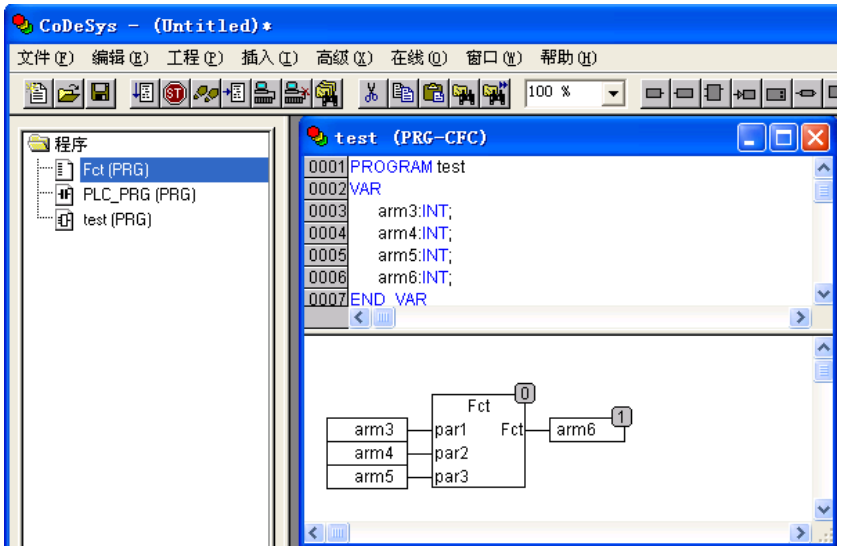


图 5-2-5 自定义函数（4）

5.3 调用 POU

关于 POU 的调用方法和调用原则在 5.1.2 中已经介绍，下面分别讲述了如何调用程序、功能块和函数。

5.3.1 调用程序

程序是唯一可执行的 POU。程序可以调用功能块和函数。程序的声明以关键字 PROGRAM 开始，如图 5-3-1 所示。

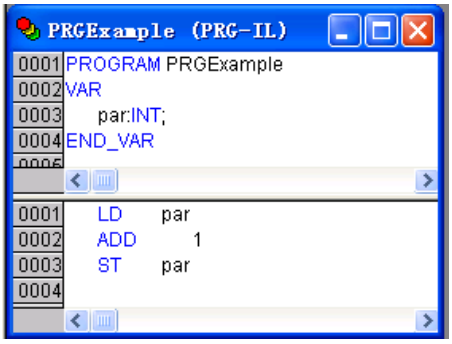


图 5-3-1 程序举例

程序可以被其它程序调用，但不允许在函数中调用程序。

可以采用不同语言调用上述程序“PRGExample”。如果调用使程序的输出值发生变化，程序会保持这个结果，直到下一次被调用。

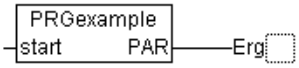
IL 语言调用：

```
CAL PRGExample
LD PRGexample.PAR
ST ERG
```

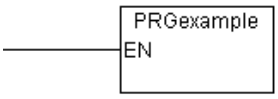
ST 语言调用：

```
PRGExample;
Erg := PRGexample.PAR;
```

FBD 语言调用：



LD 语言调用：



5.3.2 调用功能块

➢ 功能块

一个功能块是一个 POU，本身没有返回值，输出一个或多个值。功能块声明以关键字 FUNCTION_BLOCK 开始。在图 5-3-2 中，用 IL 编写了一个功能块 FUB，有两个输入变量和两个输出变量。其中一个输出是两个输入的乘积，另一个输出是相等比较结果。

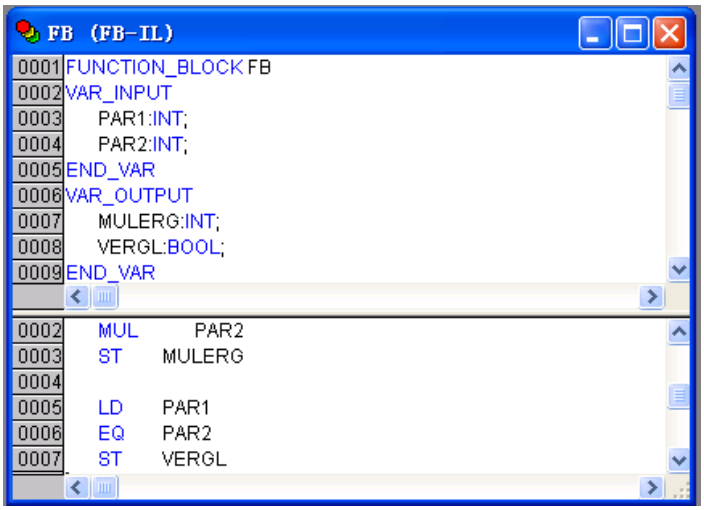


图 5-3-2 功能块举例

功能块实例声明

要想调用功能块，必须对功能块进行实例声明。例如定义一个名称为 INSTANZ 的 FUB 功能块，实例声明如下：

INSTANZ：FUB；

实例针对功能块而言，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功能块的应用实例，每个调用实例都占用独立的内存，保留独立的逻辑状态。通过定义实例实现对功能块的调用。

从外部只能改变功能块输入和输出参数。不允许对功能块的内部变量直接赋值。功能块实例名可以作为函数或功能块的输入。

调用功能块

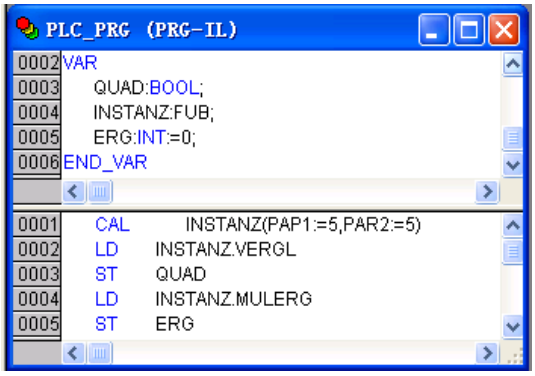
功能块的调用，只改变功能块实例中的值，结果只有当相同实例被调用时才起作用。通过输入“实例名.变量名”，调用功能块中的变量。

在文本语言 IL 和 ST 中，可以在功能块实例名后加圆括号，设置输入参数的初始值。和声明变量时初始化一样，赋值使用符号“:=”。

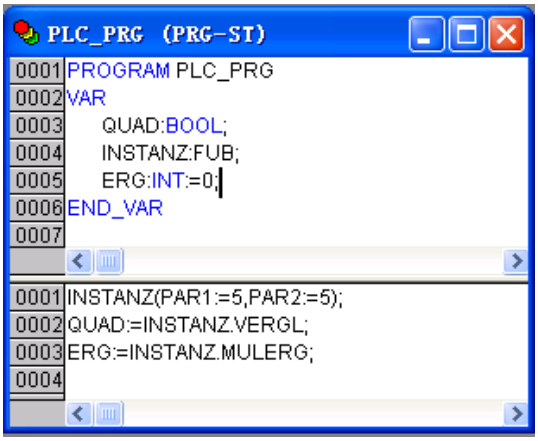
对于 SFC 语言，功能块调用只能发生在单步运行方式下。

功能块执行以后，所有的值都保留到下次执行之前保持不变。由于功能块中存在中间变量，使表面上相同的输入参数实际上可能输出不同的输出值。

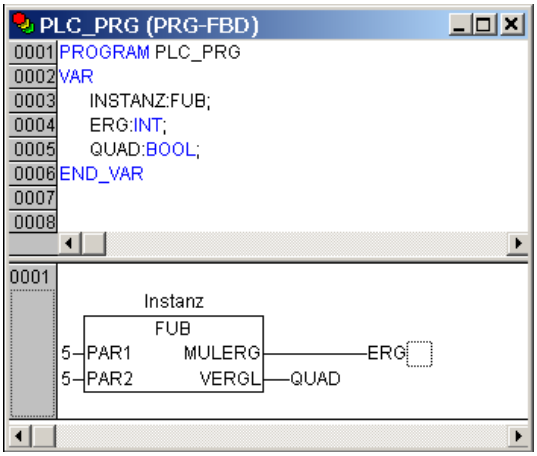
采用不同语言调用上述功能块 FUB 如图 5-3-3 所示。乘积的结果保存在变量 ERG 中，比较的结果保存在 QUAD 中。



IL 语言调用



ST 语言调用（声明部分与 IL 相同）



FBD 语言调用（声明部分与 IL 相同）

图 5-3-3 功能块调用

➢ “功能块”与“使能运算符”区别解析

介绍了功能块之后，下面介绍使能运算符，以便进一步区分功能块与使能运算符的区别。在前面提到过二者的大体区别是功能块与使能运算符的调用方式不同。对于功能块，其自备使能端，不论是否使能，在程序运行时，均会执行该功能块。然而对于使能运算符，只有在使能端 EN 有效时，才可以调用该使能运算符。但是对于具体的应用则没有介绍，这里用 LD 语言编写一个简单程序进行介绍，如图 5-3-4 所示。

从图 5-3-4 中可以看到，通电延时计时器 T1 的输入端 IN 相当于使能运算符的使能端，

即所谓的自备使能端。程序开始运行时，当延时 1s 后，%QX0.0 便置 1，PLC 中的对应通道 Q0.0 灯变亮。然而对于使能运算符 ADD，在添加使能运算符时就产生其使能端 EN，只有使能端有效时，方可调用此使能运算符。即只有当%IX0.0 为 1 时，ADD 使能运算符才会运行。

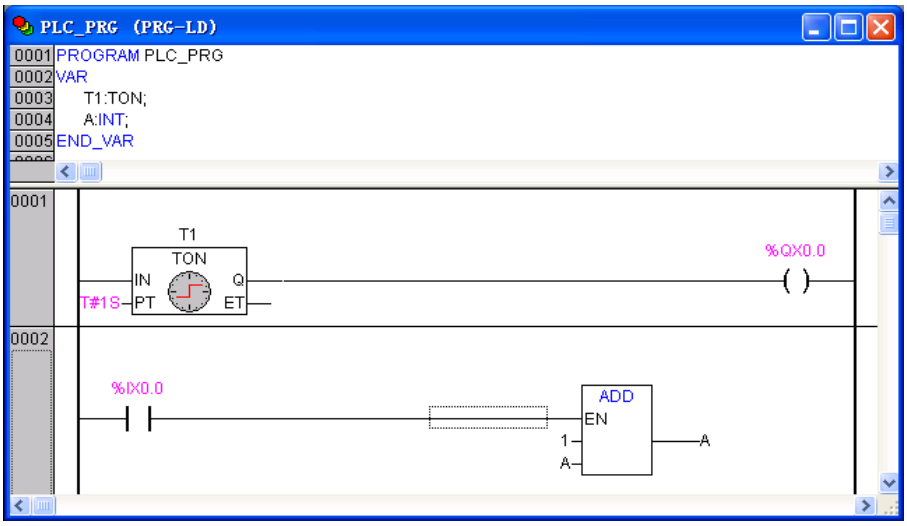


图 5-3-4 功能块与使能运算符

5.3.3 调用函数

一个函数是一个 POU。函数执行时，会对一系列特定的输入产生唯一数据类型的输出结果。相对于功能块而言，函数只有一个输出结果，没有任何内部条件。也就是说，只要给定相同的输入参数，调用函数必定得到相同的运算结果。平时所使用的各种数学运算，例如 SIN(X)等就是典型的函数类型。当定义函数时，函数必须接收一个数据类型作为返回值（返回数据类型）。函数的计算结果赋给函数本身，即函数的输出变量就是函数名本身。函数定义以关键字 FUNCTION 开始。

在图 5-3-5 中，用 IL 语言编写了函数 Fct，有三个输入变量，返回的结果是前两个数的乘积除以第三个数。调用时，函数相当于表达式里的一个运算符，可以赋初值。

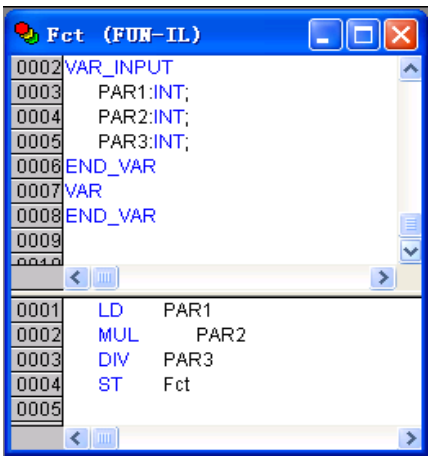


图 5-3-5 函数举例

✧ 举例：调用上述函数 Fct

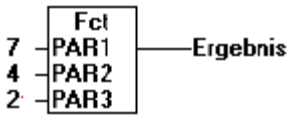
IL 语言调用：

```
LD 7
Fct 4,2
ST Result
```

ST 语言调用：

```
Result := Fct(7, 4, 2);
```

FBD 语言调用：



在 SFC 语言中，函数调用只能用单步或转化来实现。

✧ 举例

下面再列举几个函数调用的例子。

例 1：定义函数 CheckBounds，用这个函数来检查工程变量范围的溢出情况，如图 5-3-6 所示。

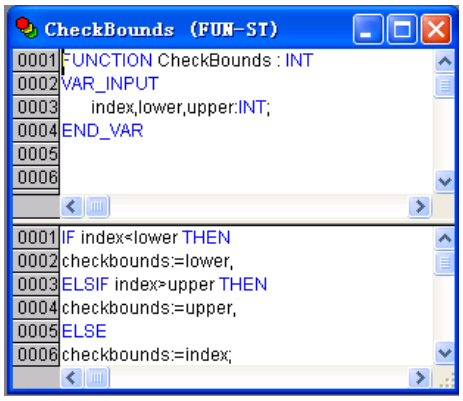


图 5-3-6 函数调用举例（1）

例 2：如果在工程中定义了名为 CheckDivByte、CheckDivWord、CheckDivDWord 和 CheckDivReal 的变量，图 5-3-7 所示为实现 CheckDivReal 函数的例子。

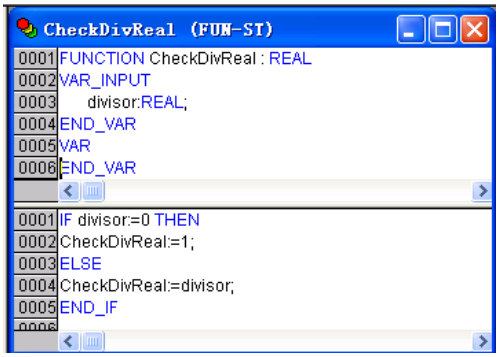


图 5-3-7 函数调用举例（2）

运算符 DIV 使用函数 CheckDivReal 的输出作为除数，如图 5-3-8 所示。避免被 0 除，除数 d 由原值 0 强制成 1，除的结果是 799。

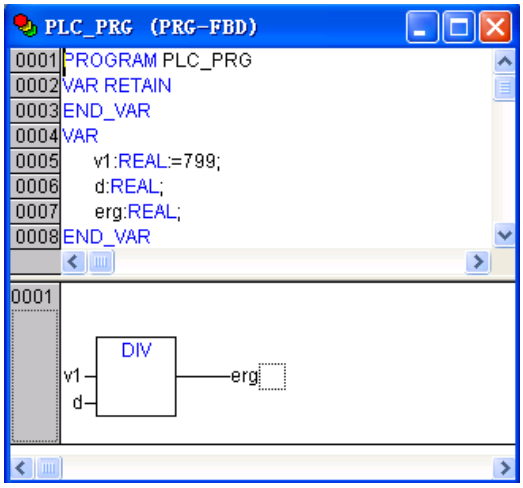


图 5-3-8 函数调用举例（3）

5.4 管理 POU 菜单

在 PowerPro 软件主界面对象组织器的“程序”选项卡中，在程序列表中选择相应的程序，点击鼠标右键，弹出管理 POU 菜单，如图 5-4-1 所示。在 2.5.1 章节对菜单的各项功能进行了简单介绍。

图 5-4-1 管理 POU 菜单

5.4.1 添加动作

动作代表某种功能，隶属于某个程序或功能块，可以被其他的程序或功能块调用。调用动作的格式为：“程序名.动作名”，“实例名.动作名”。调用动作中变量的格式为“程序名.动作名.变量名”，“实例名.动作名.变量名”。

例如，用 LD 语言编写程序 PLC_PRG 和 PRG_1，程序 PRG_1 所隶属的动作 ACT1 也用 LD 语言编写，再让程序 PLC_PRG 调用动作 ACT1 的步骤如下：

首先，编写程序 PRG_1，在程序 PRG_1 下单击右键弹出管理 POU 菜单，选择“添加动作”，弹出 New Action 对话框，确定动作的编程语言以及动作名，命名要尽量采用能反映其实际用途的字符，便于识别。点击“确认”，如图 5-4-2 所示。

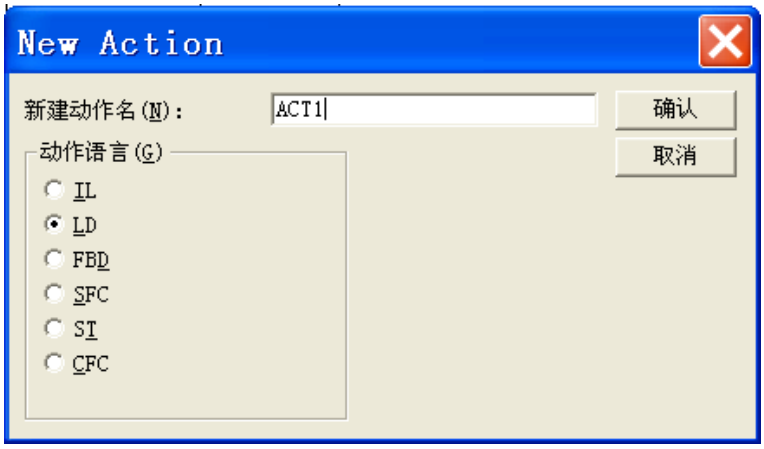


图 5-4-2 创建动作

创建完动作 ACT1，并添加在程序 PRG_1 下，接下来可根据需要在程序区编写动作，如图 5-4-3 所示。

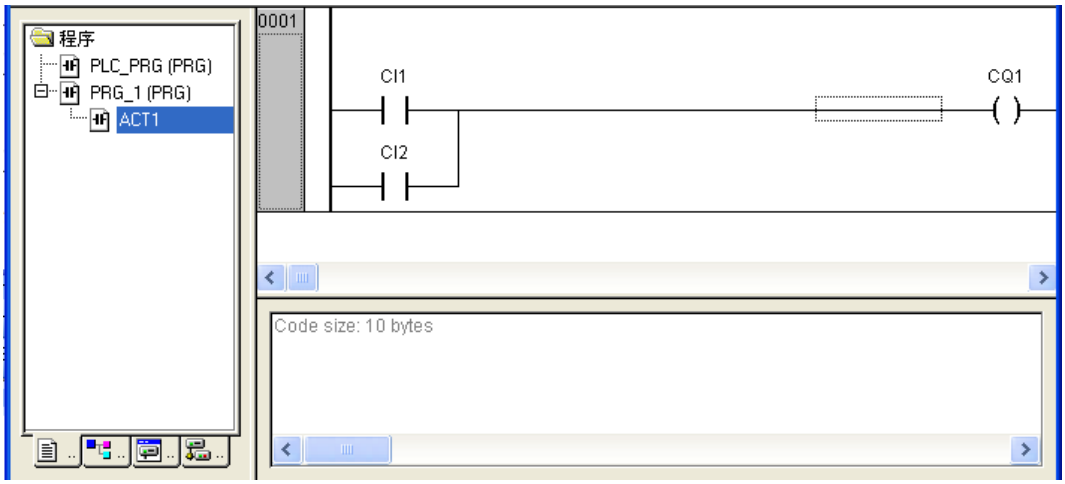


图 5-4-3 动作 ACT1

动作创建完后，可通过“程序名.动作名”，在主程序 PLC_PRG 中调用 PRG_1 所隶属的动作 ACT1，并通过“程序名.动作名.变量名”引用动作的变量 PRG_1.ACT1.CQ1。如图 5-4-4 所示。

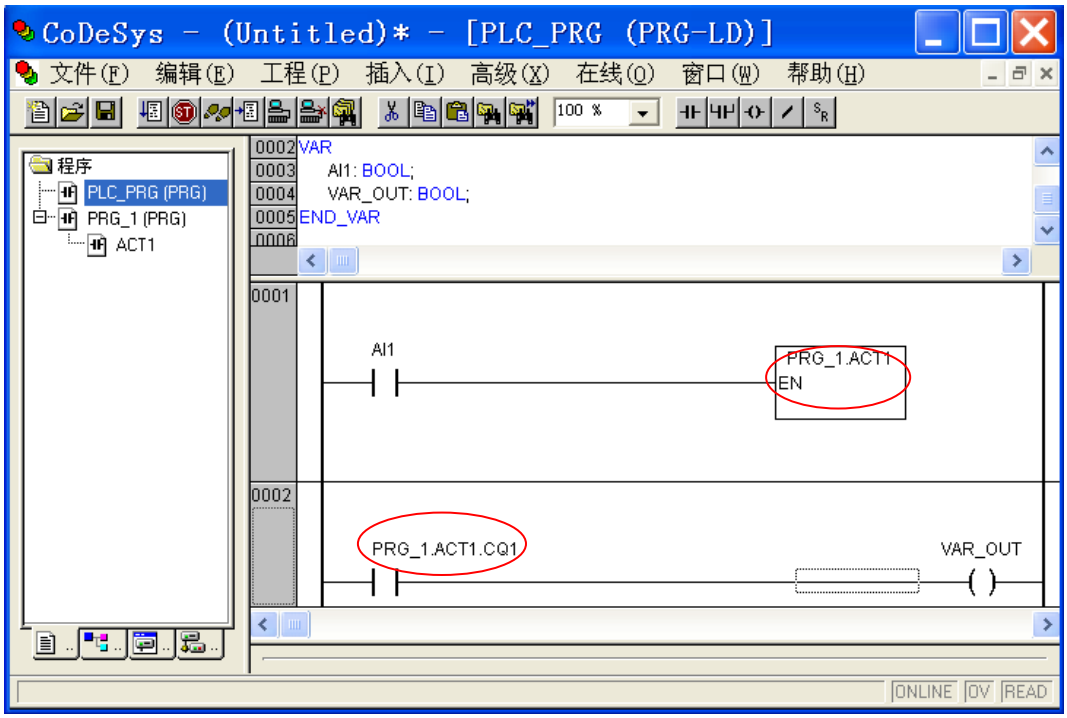


图 5-4-4 主程序调用动作

注意：

由于动作隶属于程序或功能块，所以动作中的变量在所隶属的程序或功能块变量区中声明，即动作自身无变量区。

5.4.2 建立文件夹

对于较大的工程，为了便于管理和查阅，程序、数据类型和全局变量可以统一地组织在文件夹中，并且可以设置多级文件夹（前面带有加号）。通过拖拉可以在对象组织器的范围内移动程序，点中程序并按住鼠标左键拖到指定的位置即可，如图 5-4-2 所示。文件夹使工程的组织构架更加清楚，对程序没有任何影响。“新建组”命令在对象组织器中插入一个新文件夹。如果某个文件夹被选中，新建文件夹“New Folder”将创建在它的下层。如果某个文件被选中，新建文件夹“New Folder”将在它的同层创建。

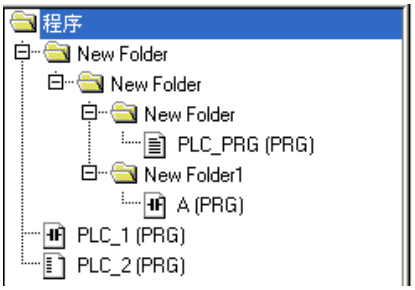


图 5-4-5 文件夹

5.4.3 转换语言

右键快捷菜单中的“转换”命令，可以把当前程序语言转换成 IL、FBD 或 LD 中的一种。该命令对程序、函数、功能块均适用。进行转换前，工程必须通过编译。在“目标语言”中，选择希望转化的语言。如果需要给程序重新命名，则程序名必须是工程中未被使用过的，如果与其它已有程序重名，则无法实现转换。如果新程序名与被转换的程序名相同，则原程序将被新程序覆盖。选定后，新的程序自动添加到程序列表中。

> 注意：
>
> 语言转换过程中会产生一些无用的语句，如果需要在不同语言间多次频繁转换，请先删除转换过程中产生的无用语句，以避免不必要的麻烦。

不同语言之间的转换，很好地体现了编程语言的可移植性，但需要注意以下几点：

- 多层嵌套的 ST 直接转化为 LD 很困难，IF、THEN、CASE、FOR、WHILE 和 REPEAT 格式的语句不能直接转化为功能块网络。
- IL 语言转化为其它语言是非常困难的，除非指令表操作符的使用范围及书写格式受到严格的限制，才有可能实现转化。其它语言则较容易转化为 IL 语言。
- FBD 语言的大部分程序能够转化 IL 和 LD 语言。
- LD 语言、FBD 语言和 IL 语言之间可以相互转化。
- ST 语言可以转化为 LD、FBD 或 IL 语言，ST 程序也能够容易地转化为函数、功能块及其相关的参数值。

从 ST 语言转化到 LD 语言的一个例子如图 5-4-3 所示。

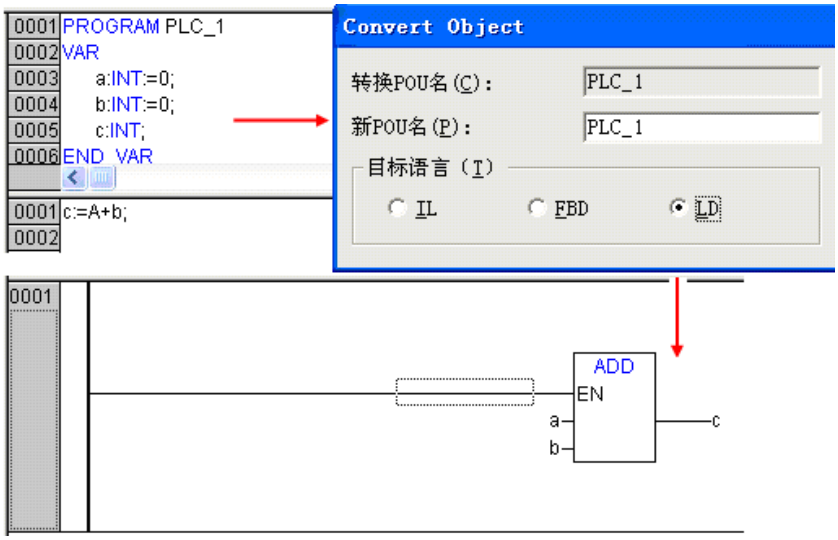


图 5-4-6 程序语言转化

第6章 PLC 工作方式

在学习了 PLC 管理数据和 POU 之后，还需要了解 PLC 是怎么工作的。6.1 章节就为您描述了 PLC 的工作方式，请仔细阅读，将有助于更好地理解程序的执行过程。

任务是 LM 系列 PLC 的一个概念。所谓的任务是指 PLC 所要执行的内容。一般来讲，PLC 是按程序循环扫描，若需要产生中断或触发其他事件，则需在任务配置中进行设置。若需了解任务配置的过程，请参见 6.2 章节。

6.1 PLC 的工作过程

PLC 以扫描方式工作。所谓扫描是指 CPU 连续执行用户程序和任务的循环过程。PLC 的工作过程一般可以分为输入采样、程序执行和输出刷新等三个阶段，如图 6-1-1 所示。

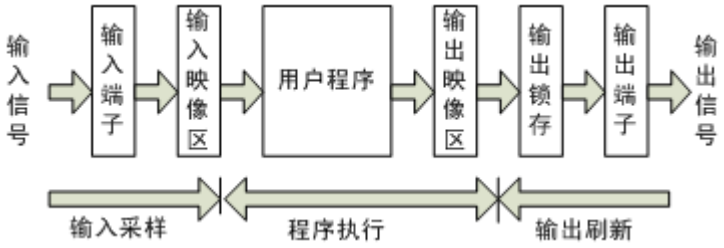


图6-1-1 PLC工作过程

- 输入采样阶段

PLC 以扫描工作方式，按顺序将所有信号读入到寄存输入状态的输入映像寄存器中存储，这一过程称为采样。在本工作周期内，此采样结果的内容不会改变，而且采样结果将在 PLC 执行程序时被使用。

- 程序执行阶段

PLC 按顺序对程序进行扫描，即从上到下和从左到右地扫描每条指令，并分别从输入映像寄存器和输出映像寄存器中获取所需的数据，进行运算和处理。再将程序执行结果写入寄存执行结果的输出映像寄存器中保存。注意，在整个程序未执行完毕之前，程序执行结果不会送到输出端口上。

- 输出刷新阶段

在执行完所有用户程序后，PLC 将映像寄存器中的内容送入到寄存输出状态的输出锁存器中，再去驱动用户设备，这就是输出刷新。

PLC 重复执行上述三个阶段。每重复一次的时间称为一个扫描周期。在一个扫描周期中，PLC 的输入扫描时间和输出刷新时间一般小于 1ms，而程序执行时间因程序长度的不同而不同。PLC 的一个扫描周期一般在几十毫秒之内。

PLC 的一个工作扫描周期主要分为上述三个阶段。但是严格来说，还应当包括下述三个过程，这三个过程都是在输入扫描过程之后进行的。

1、系统自检测。检查程序执行是否正确，如果超时则停止 CPU 工作。

2、与编程器 PowePro 交换信息。在使用编程器输入和调试程序时才执行这一过程。

3、网络通信。当 PLC 配置有网络通信模块时，与通信对象进行数据交换。

当 PLC 投入运行后，重复完成以上三个阶段的工作，即采用循环扫描工作过程，如图 6-1-2 所示。PLC 工作的主要特点是输入信号集中批处理、执行过程集中批处理和输出控制集中批处理。PLC 的这种“串行”工作方式，可以避免继电器—接触器控制系统中触点竞争和时序失配的问题，这是 PLC 可靠性高的原因之一。但是，循环扫描工作过程会导致输出相对输入在时间上的滞后，这是 PLC 的缺点之一。

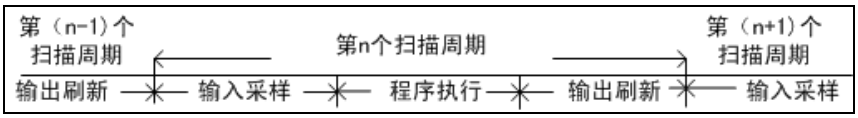


图6-1-2 PLC扫描工作方式

PLC 在执行程序时所使用的状态值不是直接从实际输入端口获得的，而是来源于输入映像寄存器和输出映像寄存器。输入映像寄存器的状态值取决于上一个扫描周期从输入端子采样取得的数据，并在程序执行阶段保持不变。输出映像寄存器中的状态值取决于执行程序输出指令的结果。输出锁存器中的状态值是上一个扫描周期的刷新阶段从输出映像寄存器转入的。

还需指出一点，在 PLC 中经常采用一种称之为“看门狗”（Watch dog）的定时监视器来监视 PLC 的实际工作周期是否超出预定的时间，以避免 PLC 在执行程序的过程中进入死循环，或 PLC 执行非预定的程序而造成系统瘫痪。

注意：

默认状态下，LM 系列 PLC 自动启动看门狗功能，当程序扫描时间超过 500ms 时，系统认为进入死循环，将重新启动，此时，PLC 的 ERR 灯以比较慢的速度闪六下，然后程序复位，重新开始执行。

6.2 任务配置

对于一个工程，可以根据需要配置多个任务，调用不同的程序。在通常情况下，建议只定义一个任务，并通过它调用主程序，其它程序则通过主程序来间接调用。这种调用方法只有程序可以使用，功能块和函数不能这样调用。

严格地说，如果没有使用任务配置，在单任务环境中，系统默认 PLC_PRG 为主程序，自动且唯一调用它，通过它实现对其它程序的调用。如果使用任务配置，程序的调用依赖于任务分配。在一般情况下，PLC 控制单任务环境即可满足要求，不需要进行任务配置。

6.2.1 配置任务

在“资源”选项卡中双击“任务配置”，右侧弹出任务配置窗口，鼠标右键点击窗口中的“任务配置”项，选择“Append Task”项，如图 6-2-1 所示。

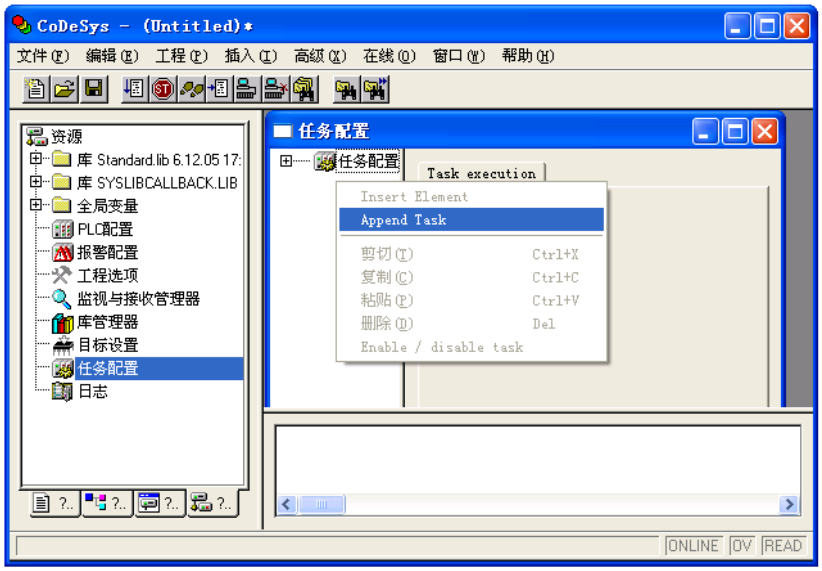


图 6-2-1 任务配置

在弹出的“Taskattributes”窗口中可以分别填入“任务名”、优先级“Priority”、“时间间隔”等信息，如图 6-2-2 所示。

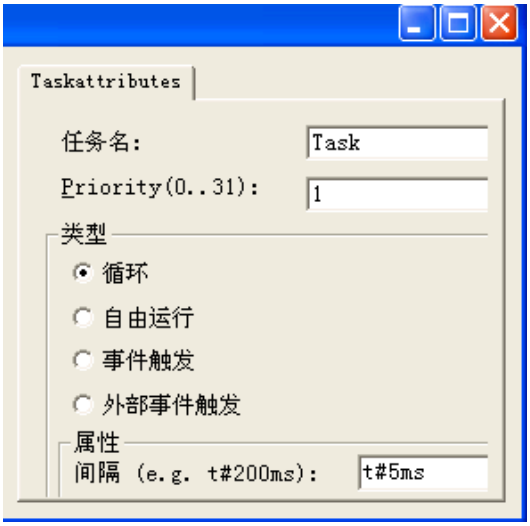


图 6-2-2 任务属性

➢ 任务名

任务的名字，以字母加数字的方式任意填写。

➢ Priority

任务的优先级。对于 LM 系列 PLC 不区分优先级，先配置的任务先调用，所以采用默认值，不需进行任何修改。

➢ 类型

循环：周期地处理任务，周期定义在“属性”的“间隔”中。如果选择“循环”类型，则会发现在新建任务“NewTask”的左侧出现“”图标。

自由运行：只要启动程序，就处理任务，运行一次后自动重新启动程序，如此循环。如果选择“自由运行”类型，则会发现在新建任务“NewTask”的左侧出现“”图标。

事件触发： PLC 不支持此项功能。

外部事件触发：PLC 不支持此项功能。

➢ 属性

间隔：当“类型”中选择“循环”时，要填入间隔时间。任务包含的程序运行的周期根据控制速度的需要填写。填写时注意在前面加“t#”（固定格式），单位可选 ms 毫秒、s 秒、m 分钟、h 小时、d 天等。

6.2.2 系统事件

系统事件用于在工程中调用 POU，而不是用于在任务中调用 POU。当相应事件触发时，调用相应的 POU，例如 T2、T3、T4 定时器溢出和快速外部中断脉冲到达等都会产生中断，调用相应的事件。

在“资源”选项卡中双击“任务配置”，右侧弹出任务配置窗口，点击窗口中“任务配置”下的“System events”，右边窗口显示所有可用的系统事件，如图 6-2-3 所示。在右边窗口中选择需要使用的系统事件，即在系统事件的前面方框里打勾，然后在该系统事件后面的“called POU”处，填入当事件触发时需要调用的程序。

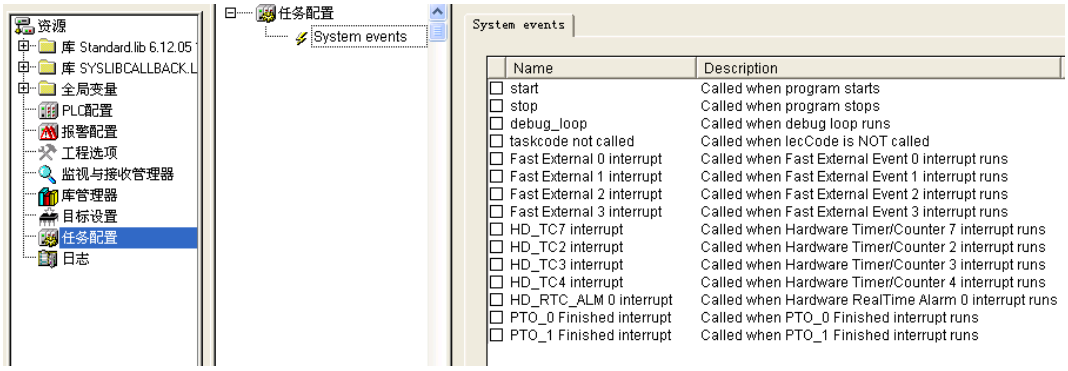


图 6-2-3 System events 对话框

当使用中断时，就需要进行系统事件配置。具体请参见 10.2 章节。

注意：

系统事件不支持仿真模式，只有在程序编译通过且登录运行后，才会响应该事件。

6.2.3 任务调用程序

鼠标右键点击“Task”项，在弹出的菜单上选择“Append Program Call”，弹出“程序调用”对话框，如图 6-2-4 所示。

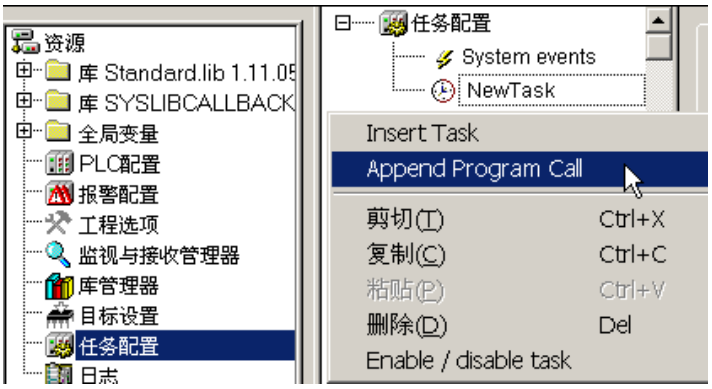


图 6-2-4 添加程序（1）

鼠标点击如图 6-2-5 中的省略号，选择需要调用的程序，点击“确认”按钮返回，程序即被任务所调用。

图 6-2-5 添加程序（2）

图 6-2-6 所示的例子是一个名称为“task”、优先级为“1”、时间周期为“5ms”的任务，通过这个任务调用 reset 程序。

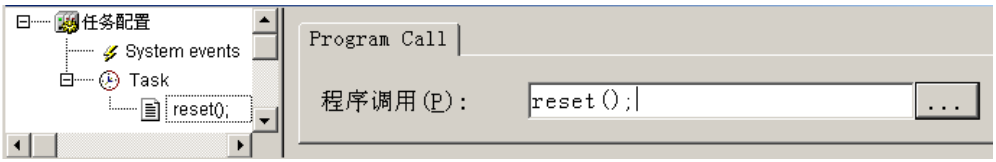


图 6-2-6 任务配置示例

第7章　创建和管理工程

在介绍了 PowerPro 软件的编程环境之后，本章将介绍如何创建一个工程。工程包含 PLC 程序中的所有对象，包括 POU、数据类型、资源和算法等。创建一个新工程的顺序是可以灵活掌握的，基本步骤主要包括目标设置、创建主程序、硬件配置和保存工程等。

7.1 目标设置

在主界面中点击“文件”/“新建”菜单，或在工具栏中点击“  ”按钮，随之出现“目标设置”对话框。“目标”是指 PLC 的存储空间，目标设置是指根据所选择的 PLC 的存储空间来进行配置。

在“配置”栏中选中“HOLLiAS-LEC G3 CPU Extend”，此目标为程序存储空间为 120KB 的 CPU 所选用的设置。点击“确认”按钮，如图 7-1-1 所示。

如果所使用的模块为存储空间 28KB 的 CPU，则需选择“HOLLiAS-LEC G3 CPU”。若不确定模块的程序存储空间大小，请参见附录。

若需要编写库指令，则应选择 None。关于库的制作，请参见 7.4.5 章节。

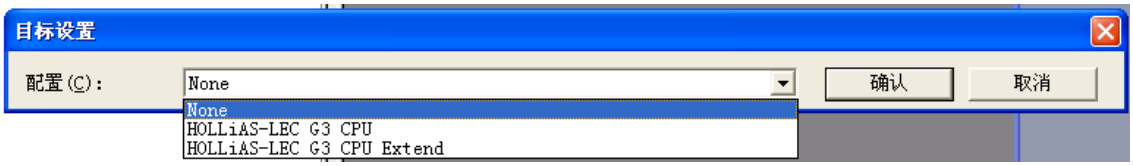


图 7-1-1 选择目标

随后弹出“目标设置”窗口，默认设置已能满足绝大多数应用需求，点击“确认”按钮即可，如图 7-1-2 所示。

图 7-1-2　目标配置

7.2 创建主程序

每一个工程都必须在程序中建立一个主程序。主程序调用其它子程序。在新建工程时，系统会自动指定一个名字为“PLC_PRG”的程序作为用户程序的主程序，不得更改。否则，用户程序无法正常运行。

目标设置完成后，会自动弹出“创建 POU”对话框，如图 7-2-1 所示。POU 程序语言可选 IL、LD、FBD、SFC、ST 或 CFC 中的任意一种，这里选用梯形图 LD 语言。在“POU 类型”中选择“程序”，主程序名默认为“PLC_PRG”，点击“确认”按钮。

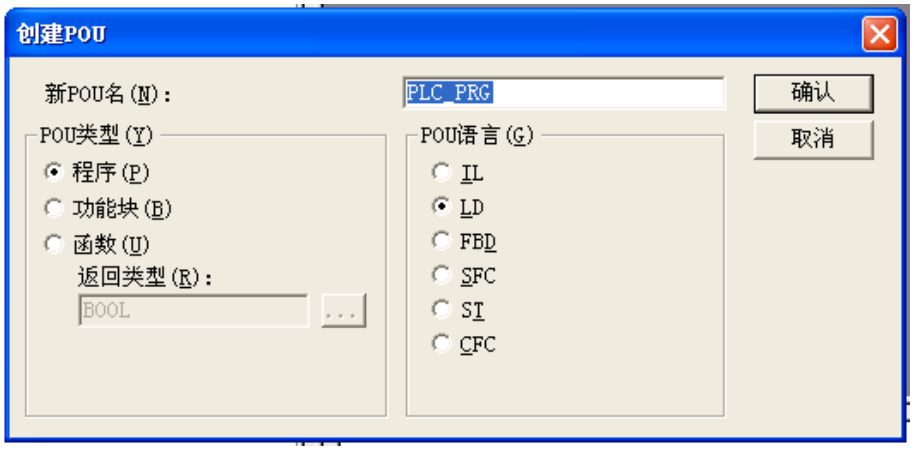


图 7-2-1 创建主程序“PLC_PRG”

7.3 硬件模块配置

PLC 系统通过硬件模块（包括 CPU 模块和扩展模块）采集和处理现场的数据。输入通道采集现场的数据，输出通道控制生产过程中的电气设备。为了完成采集和控制任务，需要根据具体的工程，对 PLC 系统的硬件模块进行相应的配置。

7.3.1 配置 CPU 模块

在“资源”选项卡中选择“PLC 配置”，弹出配置界面。点击鼠标右键，选择 CPU 模块的型号，建立“PLC Configuration”结构树。例如选择 LM3107 模块，如图 7-3-1 所示。

CPU 模块的 I/O 具有固定的 I/O 地址。例如 CPU 模块 LM3107 共有 14 个 DI 通道和 10 个 DO 通道，其中：

输入部分以字为单位，从%IW0 开始，每个输入通道占一个位，存储地址依次为：%IX0.0、%IX0.1、.....、%IX0.7、%IX1.1、%IX1.2、.....、%IX1.7，其中前 14 个地址为有效地址。

输出部分以字为单位，从%QW0 开始，每个输出通道占一个位，存储地址依次为：%QX0.0、%QX0.1、.....、%QX0.7、%QX1.0、%QX1.1、.....、%QX1.7，其中前 10 个地址为有效地址。

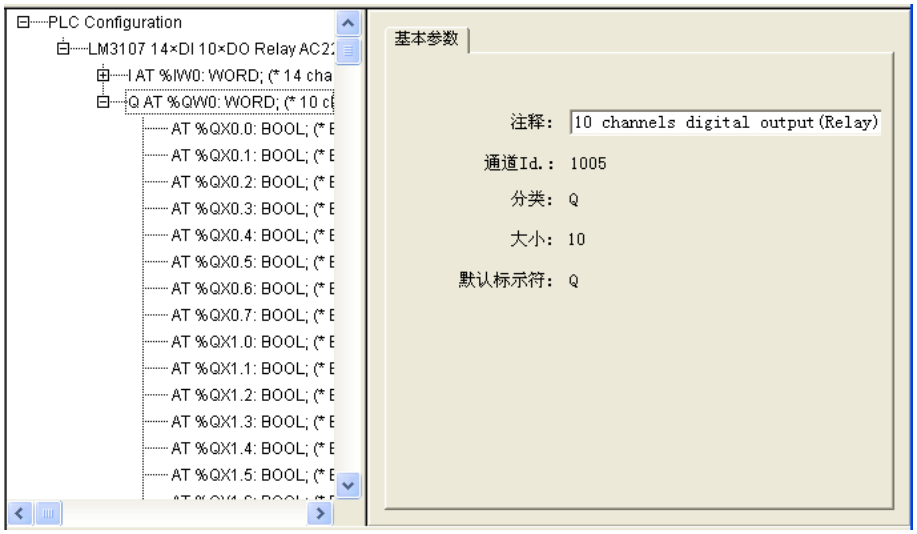


图 7-3-1 LM3107 基本参数设置

对于 CPU 的输入通道来讲，需要配置其输入通道的滤波参数。输入通道的滤波参数设置如图 7-3-2 所示，其默认值为“32”，如果需要修改此参数，可在“Value”中选择。

打开“模块参数”选项，该“模块参数”可以设置输入通道的滤波参数，其中每个字段的具体含义为：

- Index：表示模块参数索引号。
- Name：表示模块参数，对应于相应输入通道。
- Value：设置输入通道滤波参数，默认为 32，可以修改此值。
- Default：输入通道滤波参数默认值为 32。

对“滤波参数”的定义如下：若采集数据在 32 次扫描周期内维持原数据不变，即采集数据有效，完成滤波。这里是指对开关量进行滤波的次数，即：滤波次数为 32 次。

滤波参数对于高速输入通道是无效的。在用到高速输入通道的地方，无需设置相应通道的滤波参数。即便做了相应设置，对高速输入通道也是无影响的。

其他 CPU 模块的参数配置与此类似。根据实际控制要求，选择合适的 CPU 模块后，便可以添加所需的扩展模块。

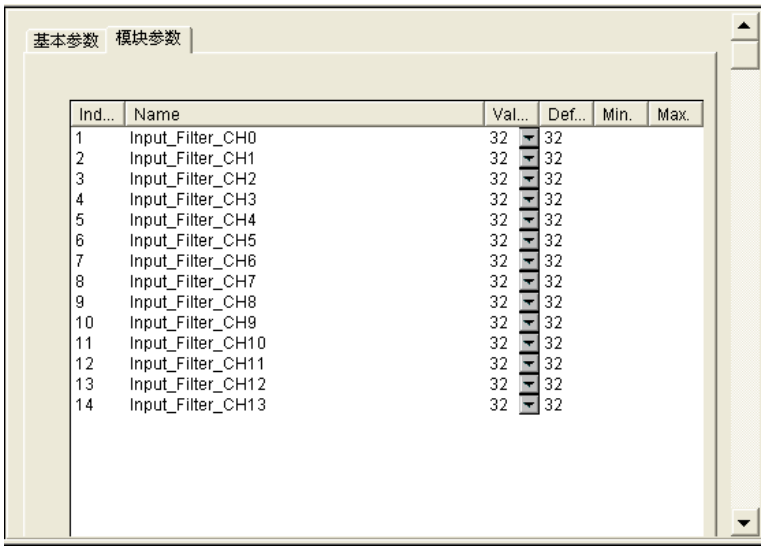


图 7-3-2 LM3107 模块参数设置

7.3.2 配置扩展模块

在选择 CPU 模块后，便要添加所需的扩展模块。在 CPU 模块上点击鼠标右键，出现模块操作对话框，选择其中的“Append Subelement”项，从弹出的扩展模块列表中选择所需的模块，如图 7-3-3 所示。例如，如果选择添加“LM3230”模块，则会在“PLC Configuration”下显示添加的模块，如图 7-3-4 所示。

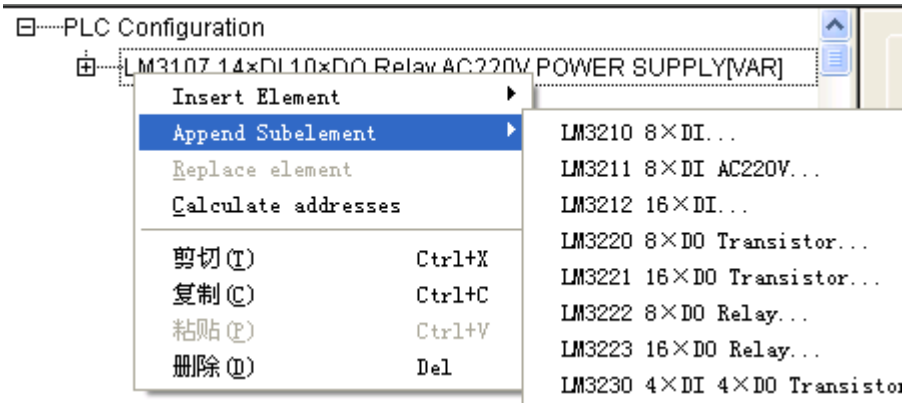


图 7-3-3 添加扩展模块（1）

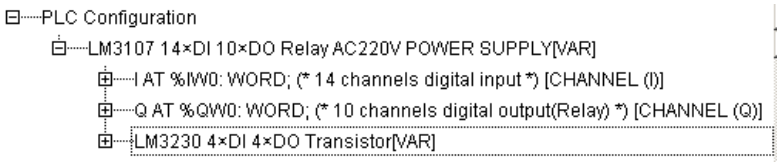


图 7-3-4 添加扩展模块（2）

CPU 模块和扩展模块的每一个 I/O 通道都对应一个实际的物理设备，其对应关系反映在随模块同时显示的“基本参数”中。“基本参数”包括节点号、输入地址、输出地址和诊断地址等信息，如图 7-3-5 所示。其中“节点号”表示模块硬件连接的先后次序，扩展模块根据连接次序，依次为“0”、“1”、“2”等，用户不可随意更改。“输入地址”和“输出地址”表示模块通道对应 I/O 存储区的起始地址。

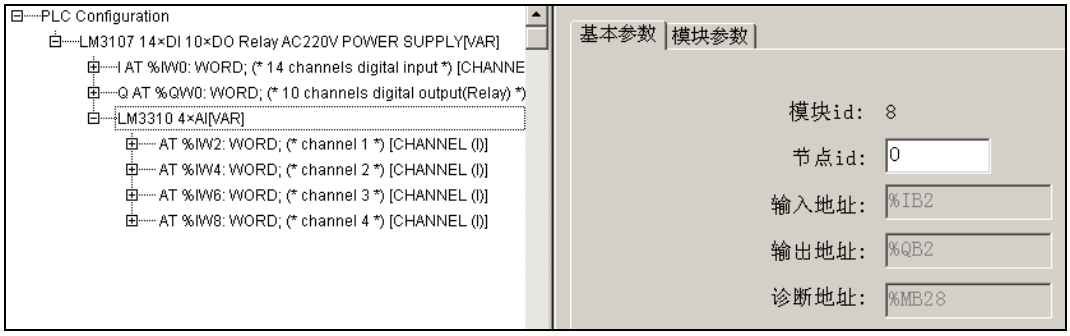


图 7-3-5 扩展模块的基本参数

扩展模块地址受其节点号、输入地址、输出地址和模块类型的影响。例如，如果四路模拟量输入模块 LM3310 为 CPU 模块 LM3107 后边第一个扩展模块，则通道地址分别为%IW2、%IW4、%IW6 和%IW8。双击模块或点击其前面的“+”号，可以看到模块的类型及各通道的具体 I/O 地址。

配置好模块后，如果需要，还可以对模块进行 I/O 变量定义以便访问。双击字符“AT”，激活变量名输入框，键入变量名即可。“AT”后面的字符“%”表示地址，即“%”后面的字符为“AT”前面变量名的地址。

如果需要，还可以为某些通道定义总体的名字，例如 CPU 模块的“%IW0”默认定义为“I”，这样可以用“I.0”直接访问第一个输入点，而无需写成“%IX0.0”。另外，也可以为各通道分别定义变量名，如图 7-3-6 中，将“%IX0.0”定义为“start”，将“%IW2”定义为“temp1”。

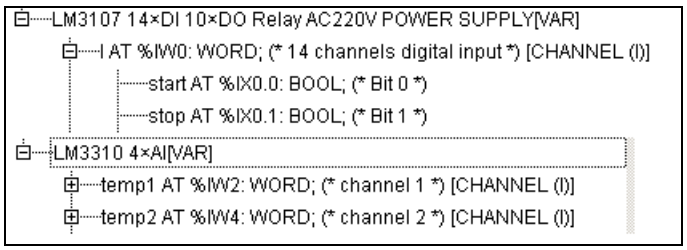


图 7-3-6 I/O 变量定义

当光标点击在“PLC Configuration”处时，可以看到在窗口的右侧有一“设置”选项卡，其具体选项包括“自动计算地址”、“检查重叠地址”和“在工程中保存组态文件”等，如图 7-3-7 所示，其中每个选项的具体含义为：

- 选中“自动计算地址”，表示“基本参数”中的节点号、输入地址、输出地址和诊断地址可以根据模块硬件连接的先后次序自动排序。
- 选中“检查重叠地址”，可以方便地检查出在编程中不小心定义的重叠地址，有利于程序被编译成功。
- 选中“在工程中保存组态文件”，可以方便地在工程中保存组态文件。

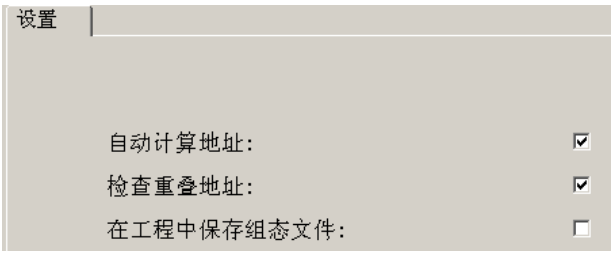


图 7-3-7“设置”选项卡

关于扩展模块的参数配置，请参见《硬件手册》，关于模拟量模块的应用，请参见 10.3 章节。

7.4 程序编写

在完成 PLC 配置后，可以开始进行程序编写。在新建程序时，可以选择程序的编程语言，包括 IL、LD、ST、SFC、FBD、CFC 等。在这里，以常用的 LD 语言为例，介绍 PowerPro 编程的规范。其余语言编程规范，请参见第九章。

LD 是梯形图（Ladder Diagram）的简称。LD 是一种图形化的编程语言。用 LD 可以方便地构造逻辑运算。LD 主要由触点、线圈、功能块和连接线等编程元件组成。LD 通过水平线和垂直线连接成平面网状图。一般称最左边的垂直线为“能量线”，其状态永远是真（TRUE）。各编程元件以一定的规则互相连接，最终连接到这条能量线上，形成一个个“节”、“段”或“网络”，完成特定的逻辑运算，如图 7-4-1 所示。

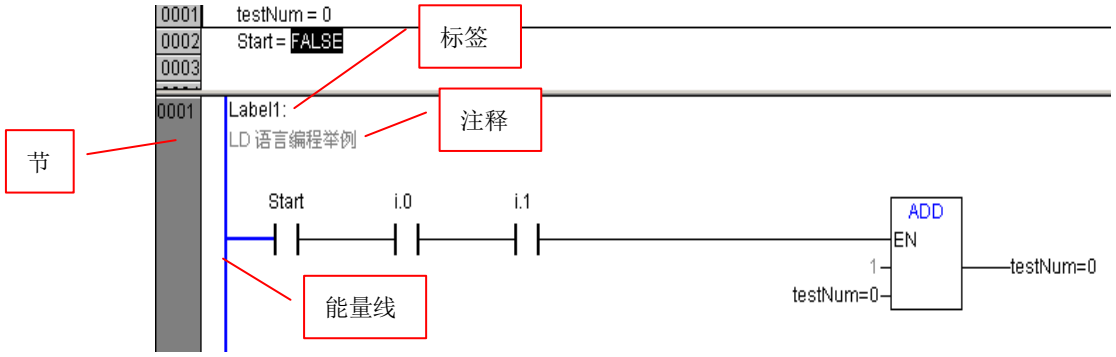


图 7-4-1 梯形图编辑器

在工作区域的代码部分点击右键，在右键快捷菜单中可以看到当前状态常用的命令，如图 7-4-2 所示。

- 前节、后节：在光标所在当前节的前面或后面增加空节。
- 串联、并联：在光标位置添加与之串联或并联的触点。
- 功能块：在光标位置插入功能块。
- 输出：在光标所在点添加线圈。
- 使能运算符：在光标位置添加带有使能端的 IEC 运算符、函数、功能块或子程序。
- 跳转：设置一个跳转。如果条件为真，则跳转。
- 返回：如果当前 POU 被其它 POU 调用，当返回条件为真时，返回到调用它的 POU。
- 注释：可以在每一节中加入注释，增加程序的可读性。

图 7-4-2 右键快捷菜单

7.4.1 节的操作

节是 PowerPro 中的一个重要概念，他是程序的基本单位，每个 POU 都是由节组成。

- 添加节

“插入”/“前节”、“后节”命令在 LD 编辑器中添加一个新节。等同于右键菜单的“前节”、“后节”。

➢ 插入节注释

每个节可有多行节注释。“插入”/“注释”命令可在当前节插入一个注释，默认文本为“Comment”。在“高级”/“选项”的“最大注释”字段中，可以设定输入节注释使用的最大行数（缺省值是 7）。在“最小注释”字段中可设定输入节注释使用的最小行数（缺省值是 0），如图 7-4-3 所示。例如，如果输入数字 2，那么在每一个节起始的标签行之后会有两个空行。如果最小注释大于 0，系统会自动留出注释行，点击注释行可直接输入注释。与程序文本相比，注释文本显示为灰色。

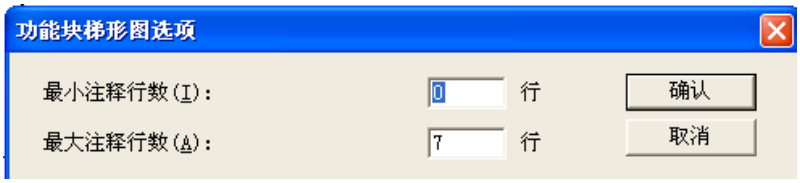


图 7-4-3 注释设定

7.4.2 添加触点和线圈

➢ 插入“触点”

快捷菜单：  。在当前位置前串联插入一个触点。

如果标记位置是一个线圈或触点和线圈之间的连接线，那么新的触点会被顺序连接到前一个触点。

触点标记文本缺省值为“???”。点击文本输入所要的变量或常量。此时，可以使用输入助手（快捷键 F2），直接从变量列表中选择输入，如图 7-4-4 所示。

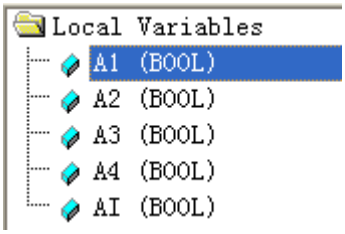


图 7-4-4 变量列表

➢ 插入“并联触点”

快捷菜单：  。在光标的标记位置处并联插入一个触点。

如果标记位置是一个线圈或触点和线圈之间的连接线，那么新的触点会并联到触点上。

触点标记文本缺省值为“???”。点击文本输入所要的变量或常量。此时，可以使用“输入变量”（快捷键 F2），直接从变量列表中选择输入，如图 7-4-4 所示。

➢ 添加“线圈”

快捷菜单：  。在光标的标记位置输出一个线圈。

如果标记位置是一段位于触点和线圈之间的连接线，那么新线圈平行插入到线圈下面。如果标记位置是一个线圈，那么新线圈平行插入到上面。

线圈标记文本缺省值为“???”。点击文本输入所要的变量或常量。此时，可以使用“输入变量”（快捷键 F2），直接从变量列表中选择输入，如图 7-4-4 所示。

线圈只能是并联的，从左向右传递逻辑值，并且把值保存到合适的逻辑变量中。可以预置状态 ON（对应逻辑变量 TRUE 值）或者 OFF（对应逻辑变量 FALSE 值）。

- “取反”操作

快捷菜单：  。触点和线圈取非。

如果线圈取非，则取非后的值会被保存到对应的逻辑变量中。如果触点取非，只有当逻辑值是 FALSE 时，才能连通。

- “置位/复位”操作

快捷菜单：  。线圈可以定义成置位或者复位状态。

用线圈符号(S)表示一个置位线圈，一旦设置为 TRUE 值，置位线圈将一直保持为 TRUE，直到被重新复位。

用线圈符号（R）表示一个复位线圈，一旦设置为 FALSE 值，复位线圈将一直保持为 FALSE，直到被重新置位。

7.4.3 添加指令

PowerPro 的指令主要有两种调用形式：功能块和使能运算符。下面分别讲述这两者的使用方法。

- 使能运算符调用

在 PowerPro 的指令系统中，一些标准指令，诸如初等运算指令、比较指令、移位指令、赋值指令、类型转换指令、逻辑运算指令等，都应采用使能运算符形式调用。

右键菜单/使能运算符，或者在“插入”菜单中选择“使能运算符”，均可插入使能运算符。

当插入一个使能运算符时，会出现带有一个 EN 标志的使能输入端。使能输入端 EN 的输入为 BOOL 类型。当使能输入端 EN 为 TRUE 值时，运算才被执行，如图 7-4-5 所示。

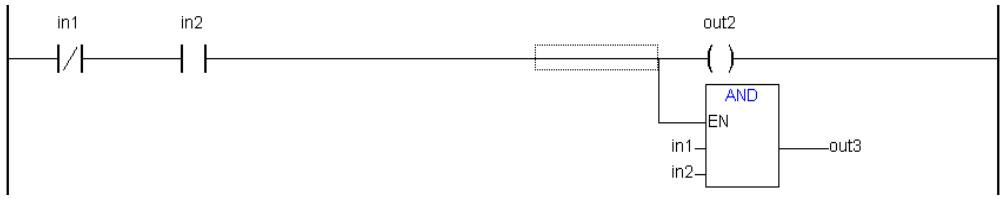


图 7-4-5 插入使能运算符

当选择插入使能运算符时，系统默认的指令为“AND”，可以选中运算符关键字“AND”，直接更改为所要求的指令，诸如赋值指令“MOVE”，也可以借助帮助来输入使能运算符的关键字。用鼠标激活运算符关键字，按下快捷键 F2，或者调用主菜单“编辑”/“输入变量”命令，在弹出的帮助窗口中选择合适的运算符，便添加了相应的使能运算符，如图 7-4-6 所示。

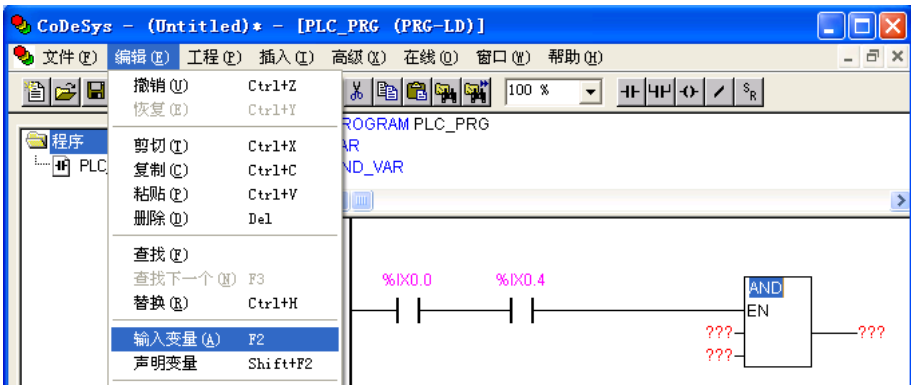


图 7-4-6 自动提示输入运算符

➢ 功能块调用

包括定时器、计数器、边沿触发器、通讯、高速输入输出、模拟量处理指令等在内的指令，应采用功能块的形式调用。

在使用功能块调用这些指令之前，首先需要了解库的概念。PowerPro 软件中，把用来实现这些常用功能的指令集合起来建立专门的库。假如要使用某些指令，首先需要添加该指令的库。关于库的概念及使用，请参见 7.4.4 节。

在添加完相应的库以后，可以在程序中用功能块形式调用该指令。右键菜单/功能块，或者在“插入”菜单中选择“功能块”，弹出如图 7-4-7 所示的对话框，根据库选择所需要的指令即可。

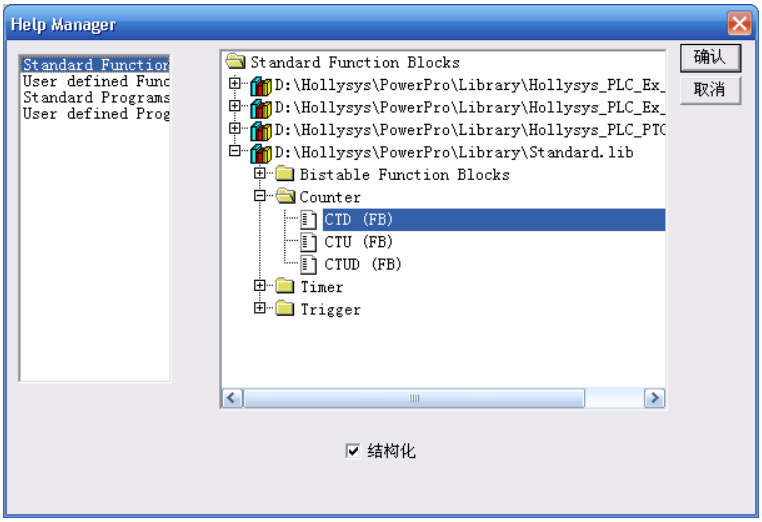


图 7-4-7 添加功能块指令

在应用中经常会混淆“功能块”与“使能运算符”的概念。其实二者是有严格区别的，“功能块”与“使能运算符”具有不同的调用形式。对于“功能块”，其自带使能端，无论是否使能，在程序运行时均会执行该功能块。对于“使能运算符”，只有在使能端 EN 有效时，才可以调用该使能运算符。

注意：

在调用功能块指令时，需要对该功能块进行实例声明，与变量定义类似，需要定义一个变量，数据类型自动默认为该功能块类型。一个程序中假如用到多个该指令，其声明的变量应不同。

7.4.4 添加库

编写 PLC 程序的过程中，经常会引用一些指令或者功能块，如字符串处理指令、触发器功能块、计数器功能块、PID 控制功能块等等。在 PowerPro 软件中，把用来实现这些常用功能的指令和功能块集合起来进行分类，然后建立专门的库。

库是指令与功能块的集合，所有的库都包含“库名.lib”文件（包含指令和功能块的输入输出代码），调用指令和功能块只需载入相应的“库名.lib”文件。

最常用的库主要有标准库（Standard.lib）、应用库（Util.lib、Util_no_Real.lib）、系统库（SysLibC16x.lib、SyslibCallBack.lib）等。标准库和系统库在建立工程时就自动添加到程序中，可以直接调用，其它库则需要添加到工程后才可以调用。系统提供的所有库文件都以库文件*.lib 的形式存在于\Hollysys\PowerPro\Library 目录下。

➢ 库管理器

库管理器用来管理库函数和功能块，包含了系统提供的所有标准函数和功能块。选择“窗口”/“库管理器”选项，打开库管理器，如图 7-4-8 所示。

库管理器可以显示所有和当前工程连接的库，可以像使用用户自定义的程序、功能块、函数、数据类型和变量一样使用函数库里的程序、功能块、函数、数据类型和变量。

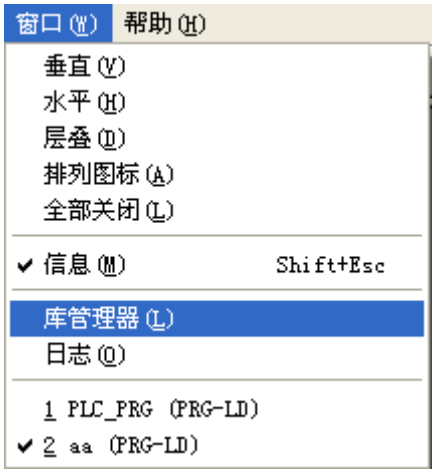


图 7-4-8 打开库管理器

标准库管理器窗口分成 4 个区域，如图 7-4-9 所示。绑定到工程的库列表位于左上区域，左下区域显示所选择库的程序、数据类型或全局变量。选择一个函数或功能块后，其变量声明出现在库管理器的右上区域，右下区域是函数或功能块的图形显示。

学会查看函数库是非常重要的。在函数库中，给出了一些关于该函数非常重要的信息，例如，该函数中有几个输入变量及中间变量，它们各自的数据类型是什么，有哪些中间变量是必须赋予初始值的以及变量的注释等等。

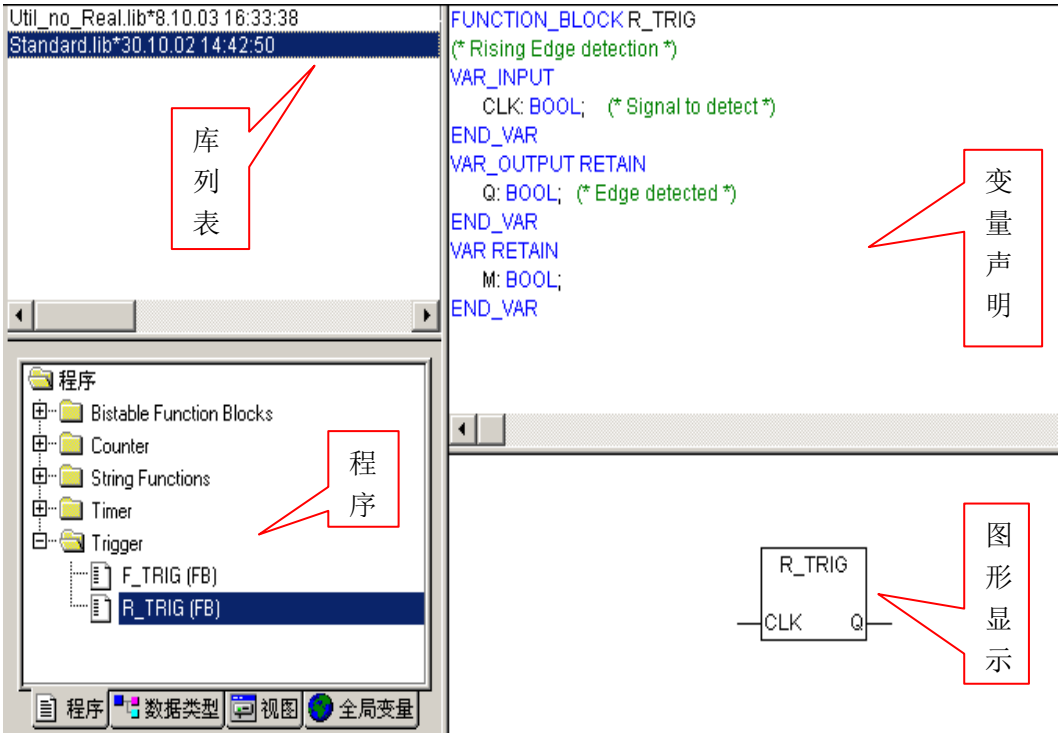


图 7-4-9 库管理器窗口

➢ 添加库

当库列表区的库文件已不能满足目前的编程需要时，则需要添加库。注意，由于 Util.lib 和 Util_no_Real.lib 的数据类型有冲突，会产生编译错误，不允许同时添加。使用库时，需要保证相应的库文件存在于如下目录：PowerPro 安装目录\Library\。添加库的过程如下所述：使用“插入”/“添加库”命令，或在库管理器窗口的库列表位置选择鼠标右键菜单命令“添加库”，如图 7-4-10 所示。

图 7-4-10 添加库

选择所需要的库文件，点击“打开”。不论哪种库，只需要打开对应的*.lib 文件即可，如图 7-4-11 所示。

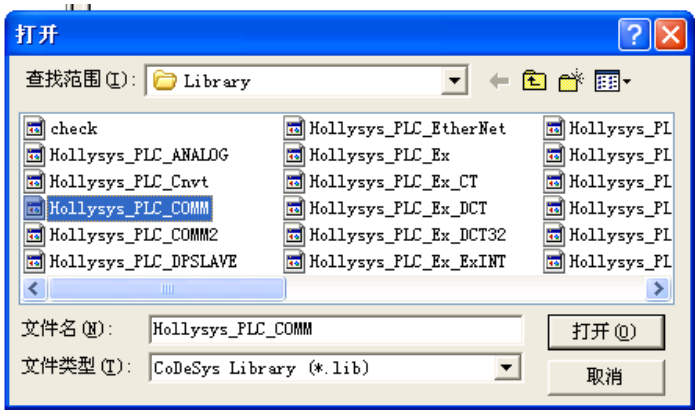


图 7-4-11 打开库

上面所选择的库被添加到库列表中，如图 7-4-12 所示。注意，凡是添加到库管理器的库都会占用用户程序空间，所以建议用户只添加所使用的库。

图 7-4-12 显示库

➢ 删除库

使用“编辑”/“删除”命令，或选择鼠标右键菜单“删除”，可以从工程和库管理器中删除已添加的库。

注意：

在新建工程后，PowerPro 自动添加了标准库（Standard.lib）和系统库（SyslibCallBack.lib），其余库文件需要手动添加。

7.4.5 库的制作

在工程实施中，如果有一些算法或逻辑是大量重复的，而且可以应用于多个工程中，则可以将这部分逻辑制作成库，以方便于程序编写、工程维护管理及技术资源共享。还可以加上口令，以保护库的具体实现算法和逻辑不被查看和修改。以下将介绍制作库的过程：

首先新建一个工程，“目标设置”中“配置”选“None”，如图 7-4-13 所示。

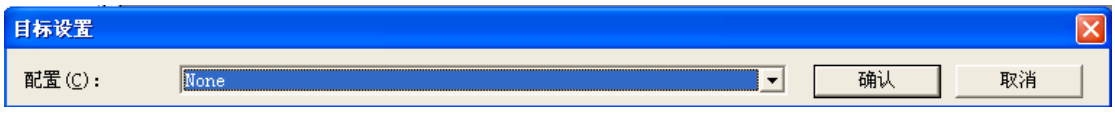


图 7-4-13 自定义库（1）

确认后，弹出“创建 POU”窗口，如图 7-4-14 所示。“POU 类型”选择“功能块”。根据需要填写“新 POU 名”，一般来说命名要尽量采用能反映其实际用途的字符。在图 7-3-2 中，“POU 语言”选择了 ST，“新 POU 名”为“Generate_CRC”，即 CRC 校验。

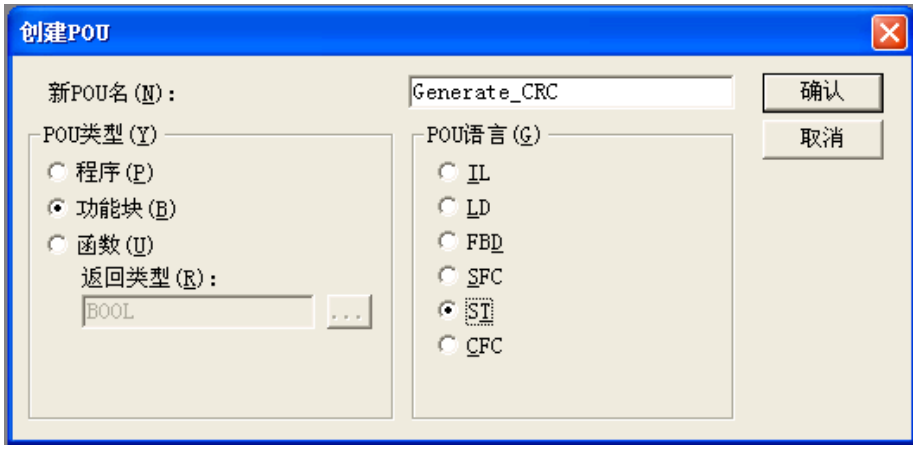


图 7-4-14 自定义库（2）

设置完毕后自动生成一个功能块结构体，可以先保存为库文件，再往里面添加所需要的内容。保存时注意要存在PowerPro\Library目录下，文件名根据需要定义，保存类型要选择内部库（Internal Library），如图7-4-15所示，使算法逻辑在库文件中实现。还有一种是外部库（External Library），其算法逻辑在模块中实现，库文件中只定义使用的变量。

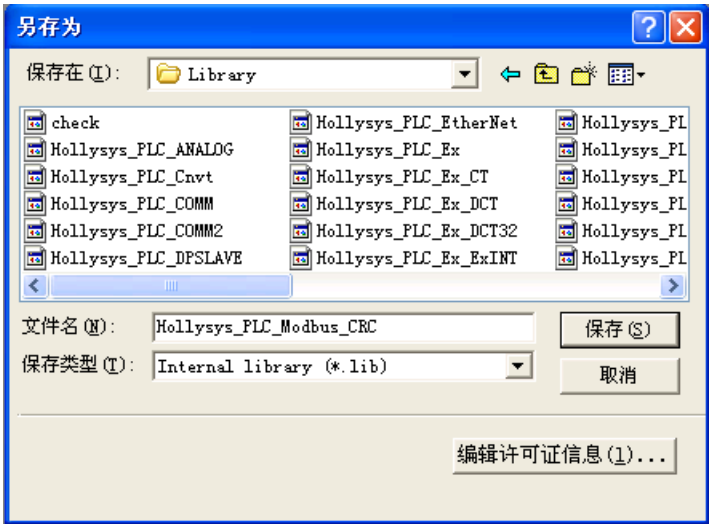


图 7-4-15 自定义库（3）

✧ 举例

本例是专为产生 CRC16 校验而制作的一个 ST 语言的自定义库，如图 7-4-16 所示。

编译通过后，就可以在其它的工程中使用此库。使用前注意在库管理器中先要加入此库。

如果想要在其它的计算机上使用此库，则只需要将此库文件复制到其它计算机的软件安装目录 PowerPro\Library 下，再添加到工程中即可。

一个库中可以包括多个功能块或函数。如果想要添加功能块或函数，只需打开库文件，在程序中添加新的功能块即可。

注意，库中的功能块不允许递归调用，即不允许自己调用自己。

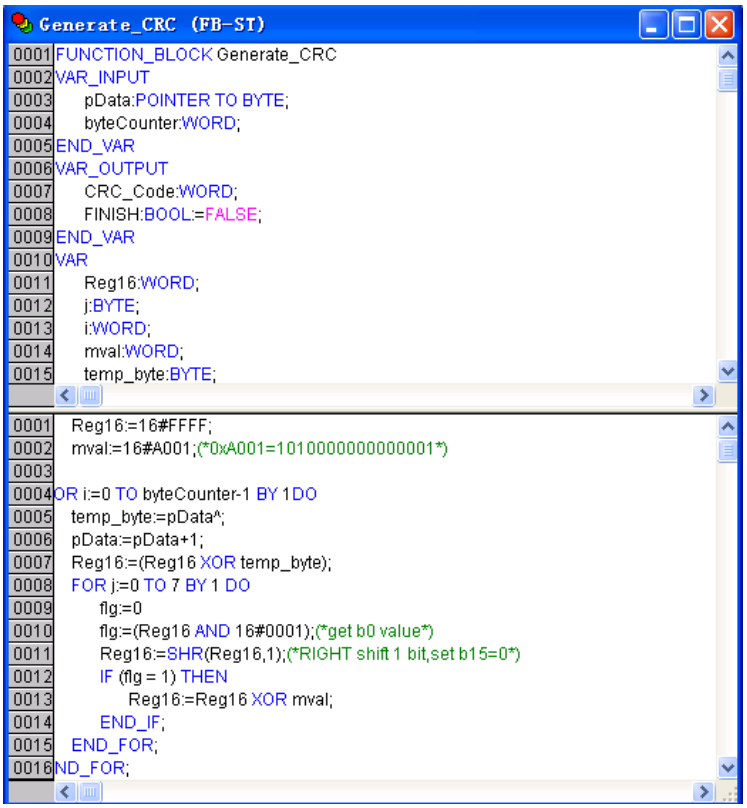


图 7-4-16 自定义库（4）

对于自定义库，可以加口令保护。这与对程序加密的原理相同，只需在“资源”选项卡中，选中“工程选项”，打开“Passwords”，设定相应的口令，即可实现加密，如图7-4-17所示。

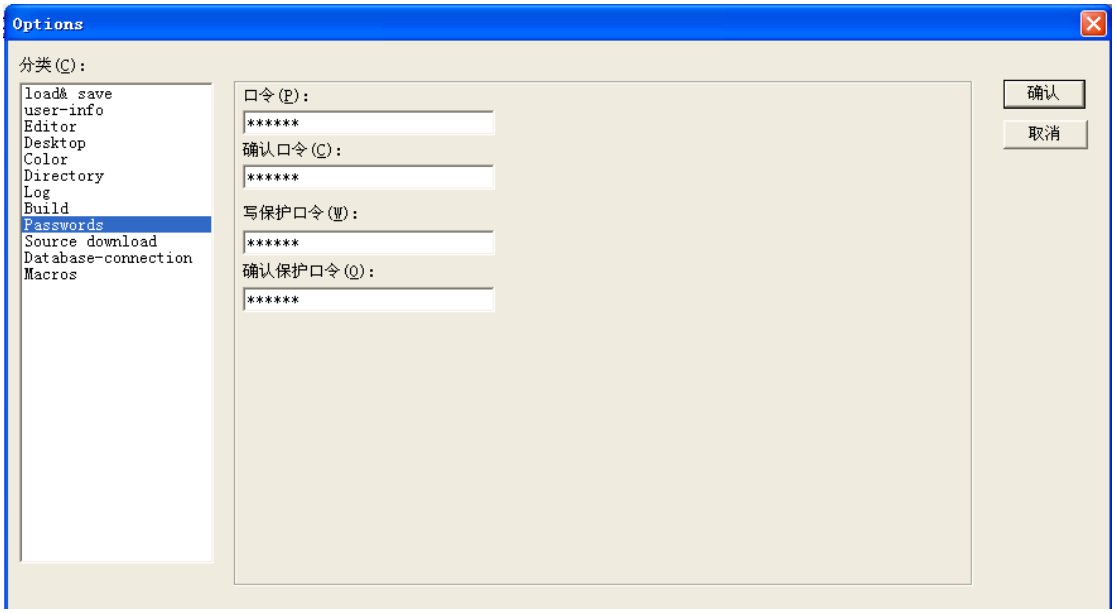


图 7-4-17 自定义库加密

7.4.6 跳转和返回

跳转和返回都是对程序扫描顺序的改变。在正常情况下，PLC 将根据主程序中节的顺序进行扫描。

- 跳转：当条件满足后，则跳转至相应的节。

右键菜单/跳转，或者在“插入”菜单中选择“跳转”，均可插入跳转命令，如图 7-4-18 所示。

插入跳转后，需要输入跳转标签（默认为“Label”）。跳转标签用于识别跳转目的地。如图所示，则跳转条件满足后，直接跳转至第三节，不再执行第二节的程序。

每个网络都有一个标签，缺省为空。标签在每个网络的首行，鼠标直接点击，即可输入标签。

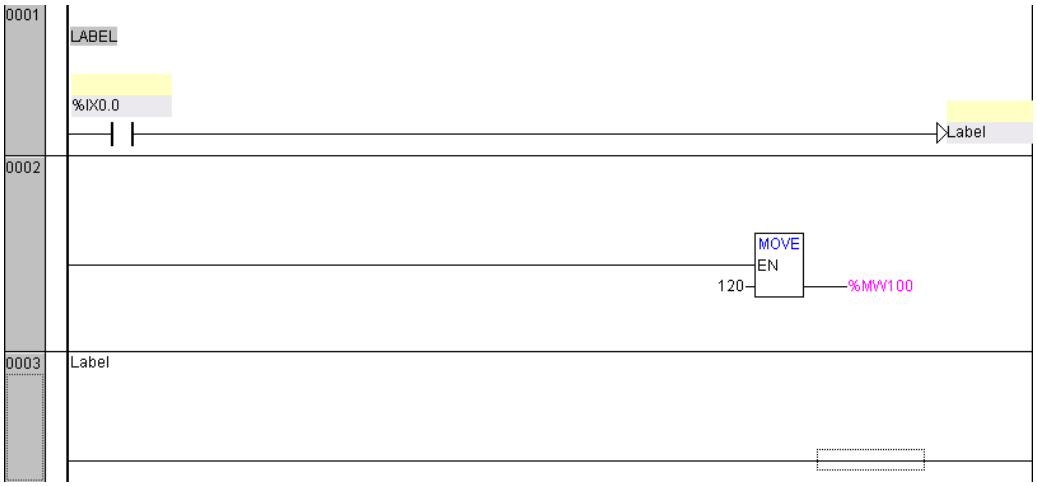


图 7-4-18 跳转

➢ 返回：当调用 POU 时，可以利用返回，当条件满足后，被调用的 POU 不再继续执行，而返回调用的 POU 中。

右键菜单/返回，或者在“插入”菜单中选择“返回”，均可插入返回，如图 7-4-19 所示。

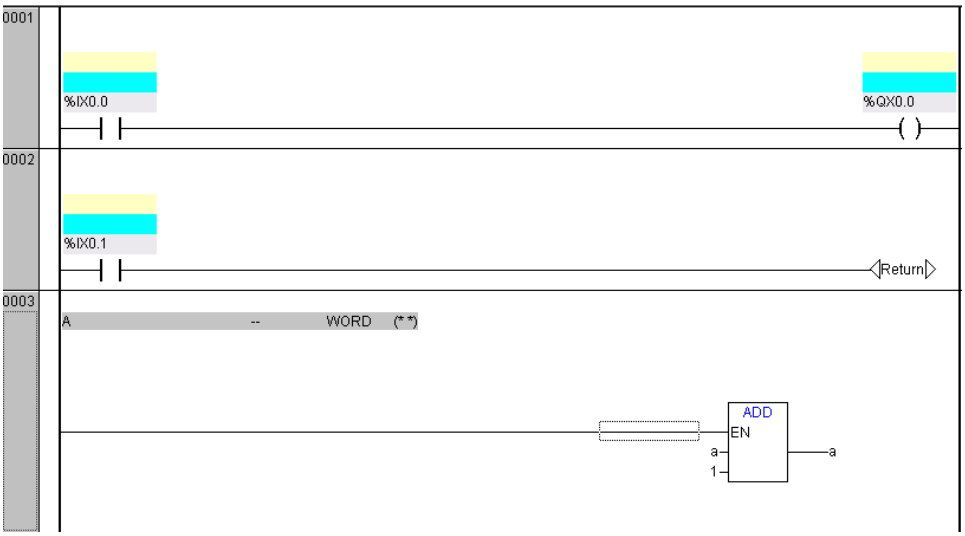


图 7-4-19 返回

7.4.7 子程序调用

当程序比较复杂时，往往需要建立许多程序。在前面的章节中曾多次提到，PowerPro 把程序名为“PLC_PRG”默认为主程序，其余均为子程序。

在调用子程序之前，首先应该建立子程序。关于子程序的建立，请参见 5.2 创建 POU 章节。建立完子程序后，在主程序中，用使能运算符的形式调用，将运算符关键字修改为子程序的名字即可，如图 7-4-20 所示。关于使能运算符调用，请参见 7.4.3 章节。

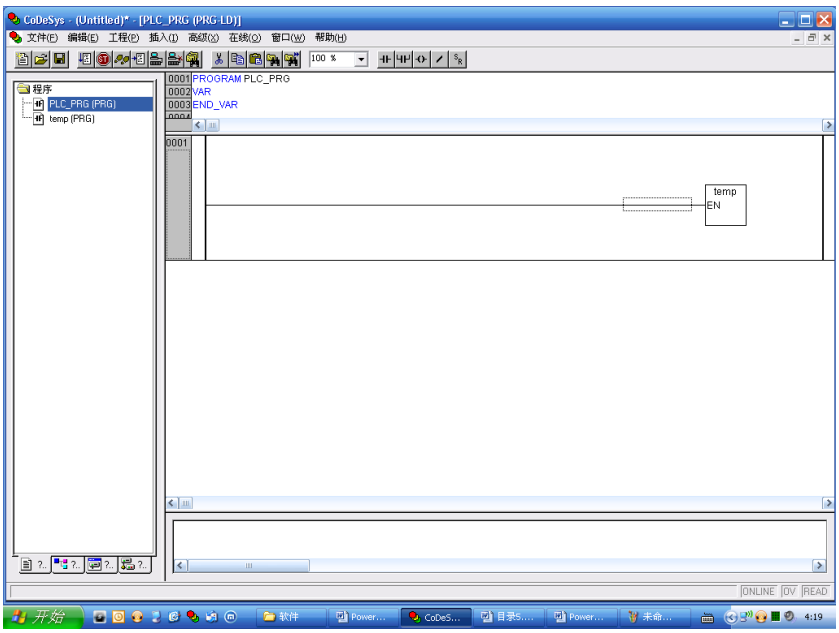


图 7-4-20 子程序调用

当主程序和子程序有变量传递时，需要定义一些输入输出变量，或者定义全局变量。关于变量，请参见 4.4 章节。

7.4.8 添加注释

为了增加程序的可读性，需要给程序、节或者变量和地址添加注释。PowerPro 提供多种方式用于添加注释。

➢ 程序和节的注释

程序的注释与节的注释一致。PowerPro 可以对每一节添加注释，如图 7-4-21 所示。具体注释方法，请参见 7.4.1 章节。

➢ 变量注释

PowerPro 允许对变量进行注释。在定义变量时，可以在变量声明对话框中直接进行注释，也可以在变量声明区进行注释，如图 7-4-21 所示。

➢ 地址注释

假如在程序中用到 I 区、Q 区或 M 区的地址，那么也可以对地址进行注释。在高级菜单/选项中，选择每一个触点的注释，并操作应用选项后，就可以在梯形图中，对地址参数进行注释，如图 7-4-21 所示。关于梯形图选项，请参见 7.4.9 章节。

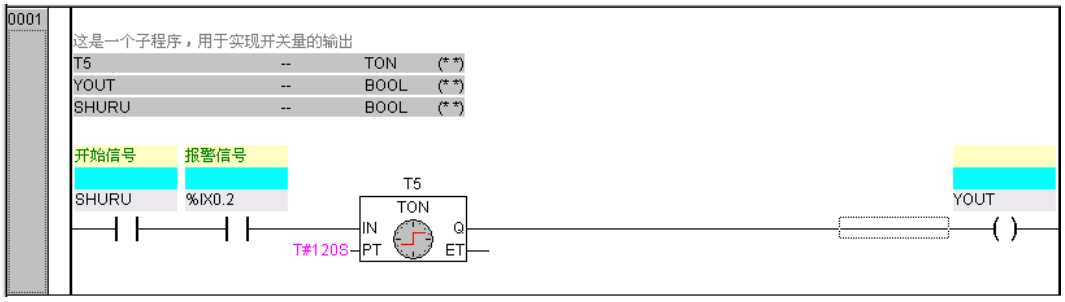


图 7-4-21 注释

注意：

添加节的注释与 7.4.6 章节中提到的跳转标签是有区别的。添加注释，必须通过选择右键菜单/注释或插入菜单/注释来实现。

7.4.9 梯形图选项

打开高级菜单/选项，弹出如图 7-4-22 所示对话框，可以对梯形图显示进行设置。

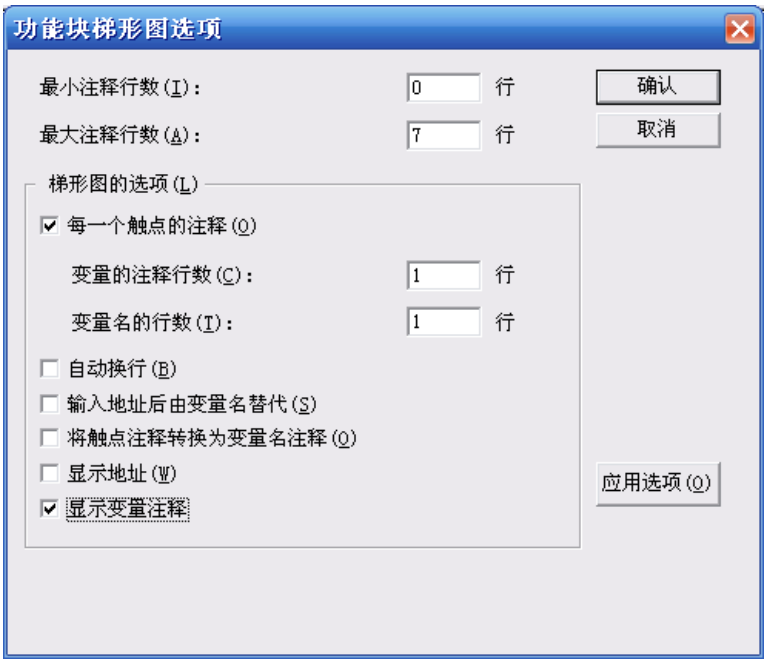


图 7-4-22　梯形图选项

- 最小注释行数和最大注释行数用于选择节注释的行数。
- 每一个触点的注释用于设置触点的注释行数以及变量名的行数。
- 自动换行：当一节的长度超过显示长度后，可以自动换行，保证在一个画面中显示所有的触点、线圈和指令。
- 输入地址后由变量名替代：当变量定义地址后，在程序中输入地址，则自动变为相应的变量名。
- 将触点注释转化为变量名注释：将触点的注释转化为变量名的注释。
- 显示地址：当变量定义了地址后，在程序中输入变量名，自动显示相应的地址。
- 显示变量注释：在每一节中，显示该节所使用的变量名、数据类型、地址及注释。

7.4.10　保存文件

在主菜单中选择“文件”/“保存”菜单，或在工具栏中点击“ ”按钮，可以保存当前工程。

在“文件名”中填入新建工程的文件名，建议使用具有一定实际含义的字母或数字。

“保存类型”选择“*.pro”，工程文件将保存在默认目录\Hollysys\PowerPro\Projects 下，如图 7-4-23 所示。

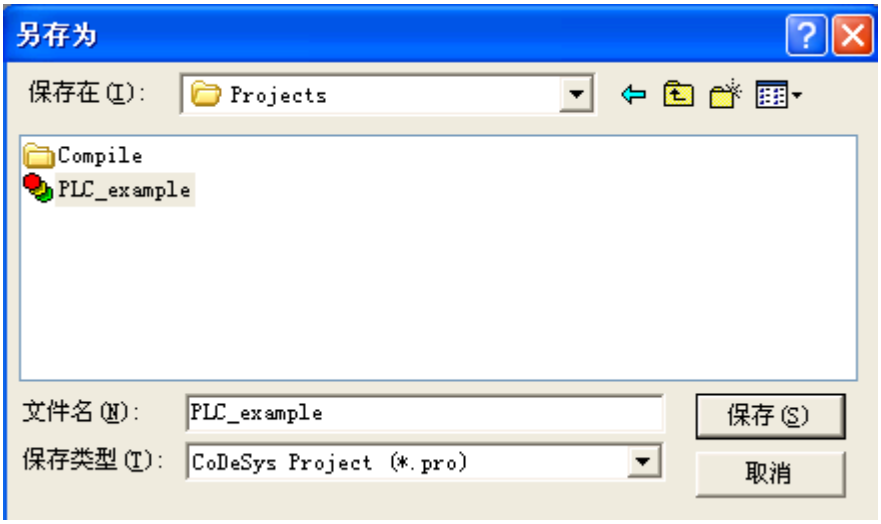


图 7-4-23 保存工程（1）

工程文件保存后，主界面左上角的“（Untitled）*”变为保存的工程文件名，如图 7-4-24 所示。在创建工程的整个过程中，要养成随时存盘的习惯，以免由于误操作而造成数据的丢失。当工程内容改变未保存时，主界面左上角工程名的后面会出现一个“*”号，保存后“*”号消失。

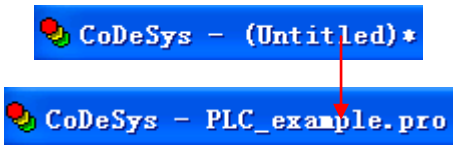


图 7-4-24 保存工程（2）

可以使用“工程”/“选项”菜单或者“资源”选项卡中的“工程选项”，来设置系统的某些属性。使用选择“工程”/“选项”会自动弹出设置对话框。在对话框的左边选定一个分类后，在右边出现相应的设置选项。所做的更改立即生效，同时被保存入初始化文件中，并在下一次启动系统时自动载入。

7.5 管理工程菜单

PowerPro 软件以工程文件的形式来保存用户程序，所有的信息都集成存放在以*.pro 为后缀的工程文件中。PowerPro 软件的默认安装目录结构如图 7-5-1 所示，其中“Library”和“Projects”分别用来存放库文件和工程文件。系统提供了很多针对工程的操作，帮助用户更好地管理工程。

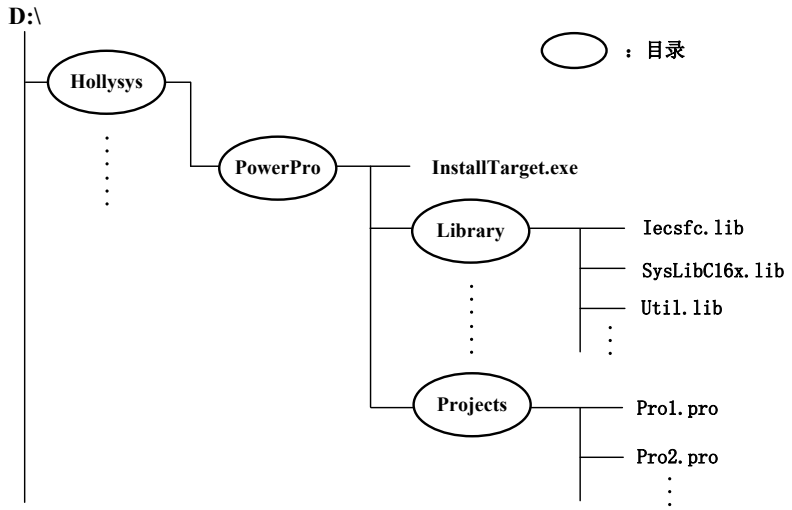


图 7-5-1 软件目录结构

打开主菜单中的“工程”菜单，如图 7-5-2 所示。

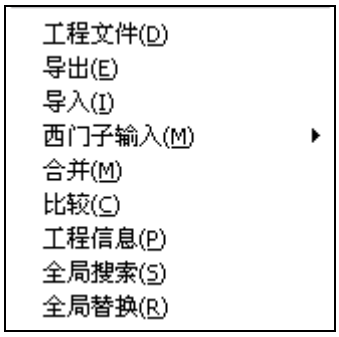


图 7-5-2 管理工程菜单

下面介绍几项常用的工程管理命令。

7.5.1 打印工程文件

使用“工程”/“工程文件”命令可以打印整个工程的文档，或者选择其中的一部分打印。工程文件由工程信息、文档内容、程序和资源等元素构成，如图 7-5-3 所示，其中资源包括全局变量、PLC 配置、报警配置、工程选项、监视与接收管理器、任务配置和参数管理器等内容。

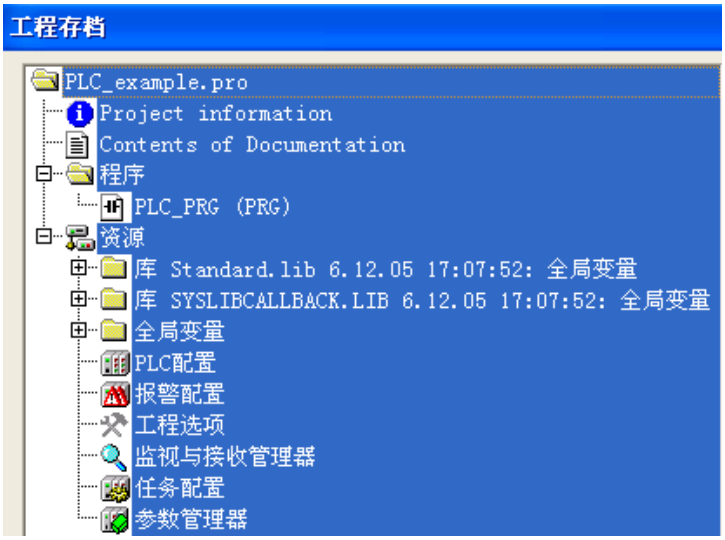


图 7-5-3 打印工程文件

用蓝色高亮可以选择需要打印的区域。如果要选定整个工程，直接在第一行选中工程文件夹即可。如果只想选定一个单独的对象，点击相应的对象。对象的前面带有加号的为多重对象，点击加号可扩展。选好后弹出打印对话框，可设置打印属性。可以按照一定的格式打印一个工程文件。选择“文件”/“打印机设置”，弹出如图 7-5-4 所示的对话框。

图 7-5-4 Documentation Setup 对话框

➢ 文件

在文件字段，输入带“.dfr”扩展名的文件名，页面布局会保存在此文件中。保存设置的缺省文件是 DEFAULT.DFR。

➢ 浏览

如果要保存在一个已存在的布局里，可点击“浏览”按钮浏览目录找到所需的文件。

➢ 编辑

如果点击“编辑”按钮，设置页的对话框则会出现。选择“插入”/“占位符”，可分别插入页、程序组织单元名、文件名、日期、内容。选择好插入的内容及位置后，按住鼠标左键，将其拉伸为矩形后，松开鼠标左键，即可将其放在可打印的页面上和文本区。

➢ 为每一个工程开始新页

如果选中“为每一个工程开始一个新页”，则打印时另起新页，否则不另起新页。

➢ 为每一个子工程开始新页

如果选中“为每一个子工程开始一个新页”，则打印时另起新页，否则不另起新页。

➢ 打印机设置

使用“打印机设置”按钮，打开打印机设置。点击“属性”，同样可以弹出“打印”对话框。

图 7-5-5 所示为一个自定义的页面布局，分别在相应的位置插入页码（Page）、程序组织单元名（POUName）、文件名（FileName）、日期（Date）和内容（Content）等占位符，占位符相当于通过将矩形置于布局中进行占位。在打印输出时，相应的内容会映射在相应位置处。当布局设置好后，点击“确认”。

如果模板被更改，则 PowerPro 软件会提示是否关闭以前保存的信息。

另外，在“工程”/“选项”/“Desktop”下，如果选中“显示打印边界”，则会在程序的相应范围内出现红色虚线，以示打印界限。否则，不会显示该虚线。

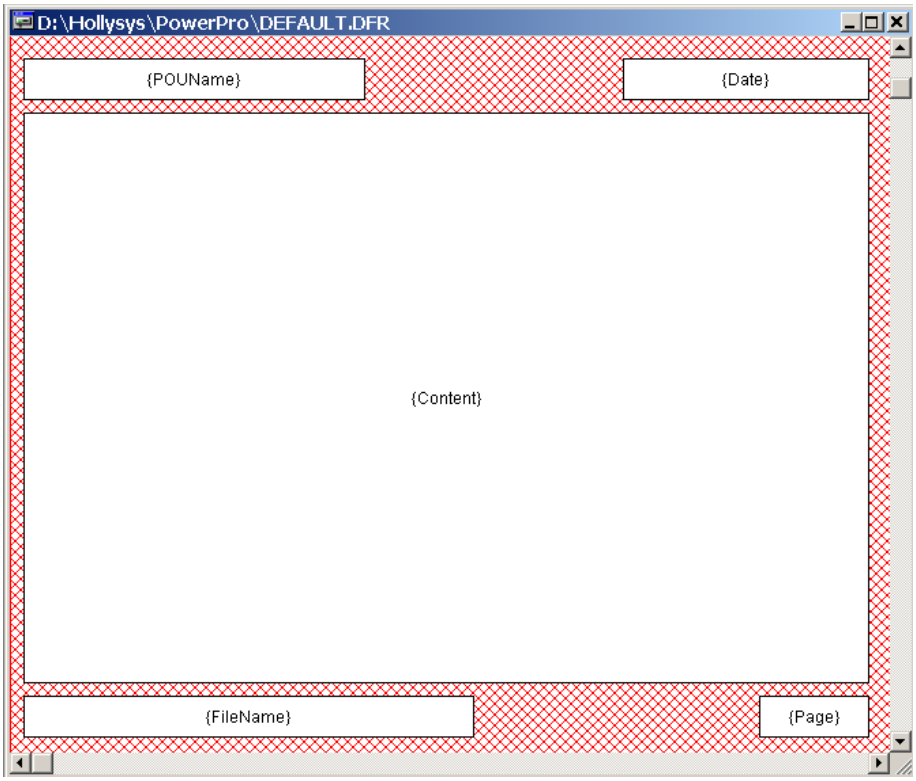


图 7-5-5 页面布局设置

7.5.2 导入导出工程

“工程”/“导入”、“导出”命令可以导入或导出工程对象，以便在不同的工程文件中交换程序。在导出文件时，对话框底部的“每个对象对应一个文件”选项，可以选择对象导出到同一个文件，还是导出到不同的文件中，如图 7-5-6 所示。

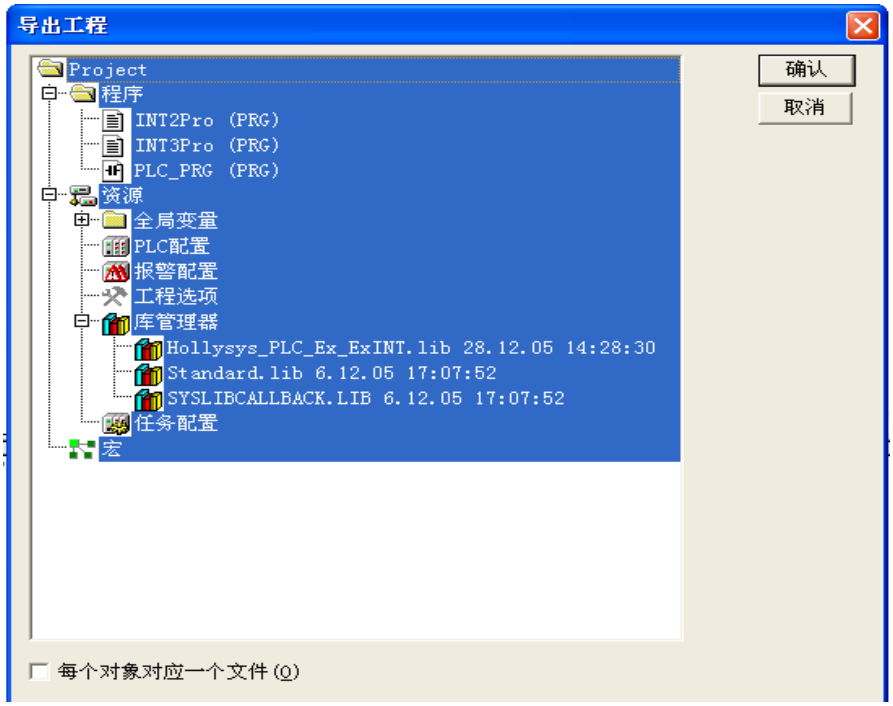


图 7-5-6 导出工程（1）

设定完成后，弹出导出工程对话框，如图 7-5-7 所示。在“目录”下指定导出对象的保存路径，即在指定目录下生成以*.exp 为扩展名的若干导出文件，同时弹出消息窗口列表，显示相关信息。

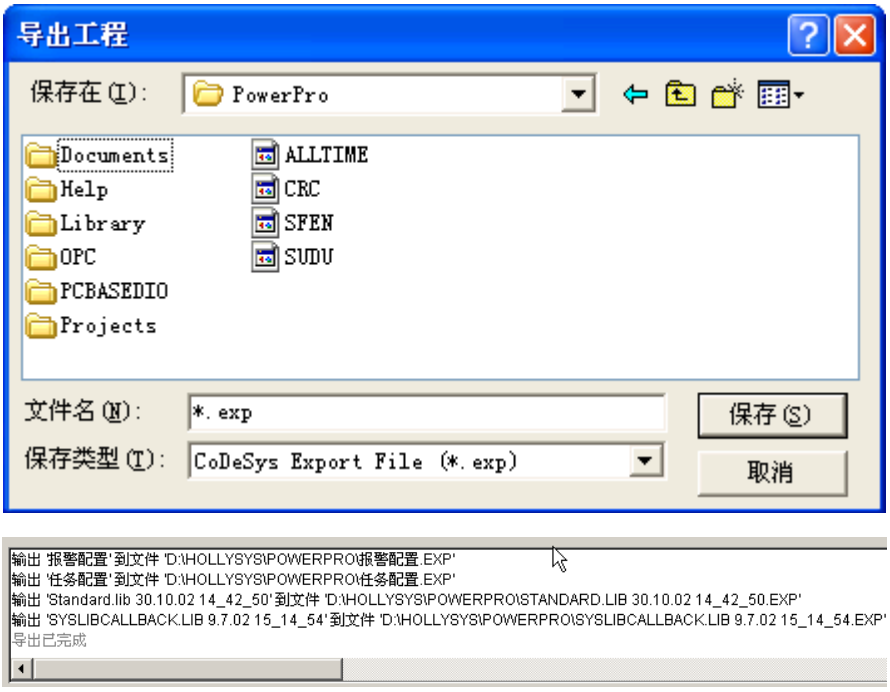


图 7-5-7 导出工程（2）

导入工程对象时，选定导入的文件*.exp，对象即被导入到当前工程的相应窗口中去。如果在该窗口中，已经有一个同名的对象存在，则会出现对话框，询问是否替换它：“对象已存在，要替换它吗?”，如图 7-5-8 所示。

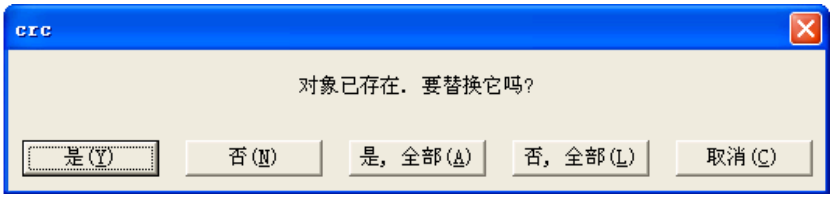


图 7-5-8 覆盖提示

导入的对象和导出的结果都存放在*.exp 文件中。通过*.exp 文件，可以在不同的工程之间搭起一座沟通的桥梁，可以方便地交换工程中的所有对象，如图 7-5-9 所示。

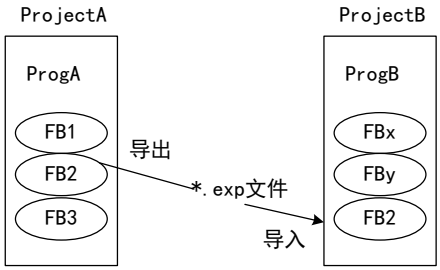


图 7-5-9 导出、导入示意图

7.5.3 合并工程

“工程”/“合并”命令可以把其它工程中的对象合并到当前工程中。打开合并工程对话框，选定一个工程，弹出该工程的所有对象列表，选择需要合并的对象，如图 7-5-10 所示。

对于库或资源的合并，也会弹出对话框提示是否覆盖。

对于程序的合并，会将新程序添加到原程序列表中。如果合并的程序与当前工程中的程序同名，则会弹出对话框询问是否覆盖。

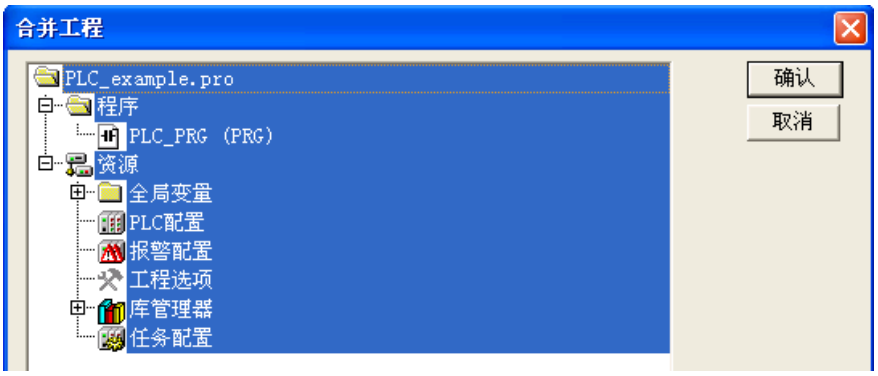


图 7-5-10 合并工程

需要注意的是，在合并工程时，系统事件不会被合并。

7.5.4 比较工程

“工程”/“比较”命令可比较当前工程与另一个工程的所有对象的内容。如果想知道是否在当前工程中做了修改，那么可以比较当前打开的工程和它的前一个版本。当执行此命令后，弹出工程比较对话框，如图 7-5-11 所示。

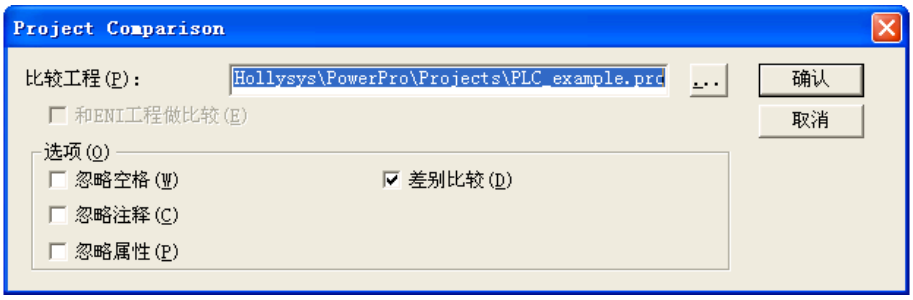


图 7-5-11 比较工程（1）

选定与当前工程比较的目标工程后，弹出比较结果列表，如图 7-5-12 所示。不同颜色的对象文字代表不同的比较结果，共有五种可能的颜色：

- 黑色：对象内容相同。
- 红色：对象内容不一致。
- 蓝色：当前工程中新增加的对象。
- 绿色：当前工程中没有的对象。
- 灰色：两个工程中内容不一致的对象，鼠标双击对象可查看详细内容。

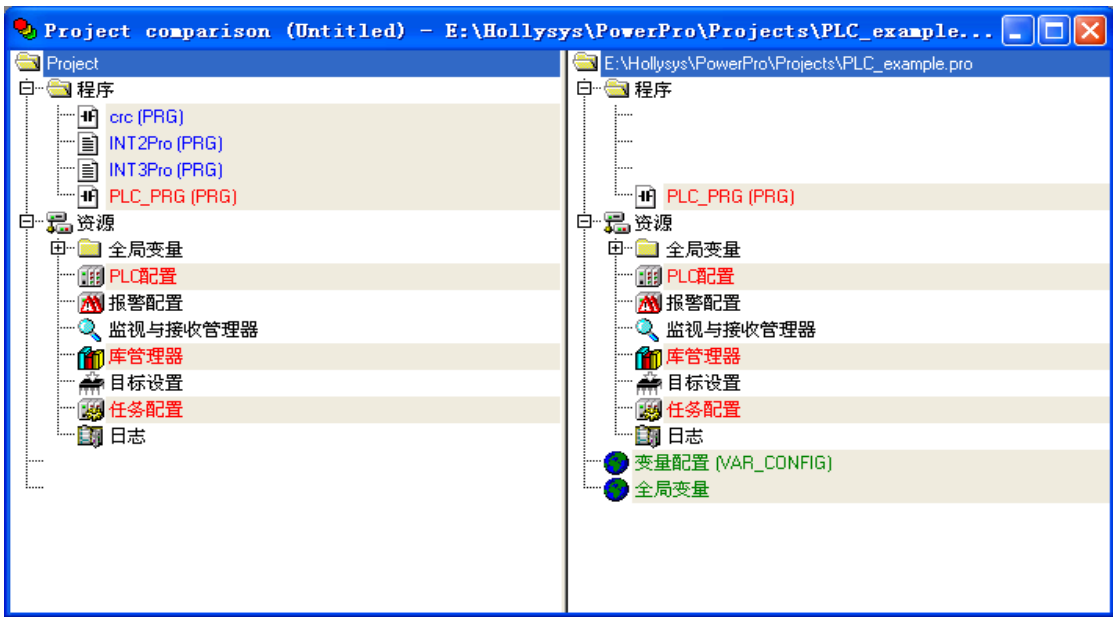


图 7-5-12 比较工程（2）

在具体的程序编写过程中，还可以利用“比较”命令，对修改前后的程序代码进行比较，如图 7-5-13 所示。在比较模式下，不允许编辑工程对象，所有操作功能禁用。只有关闭了比较列表窗口，才能对工程进行编辑。

另外，比较工程不支持硬件配置的比较。

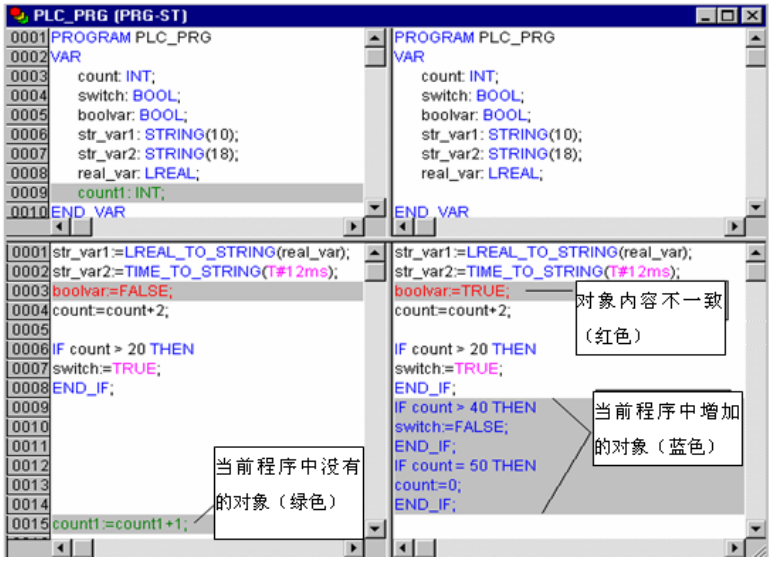


图 7-5-13 比较工程

7.5.5 用户口令

“工程”/“选项”/“Passwords”选项可以设定单一用户口令，保护 CPU 免受未授权的访问，防止文件被随意地打开和更改，如图 7-5-14 所示。其中“口令”字段为工程设置浏览权限，“写保护口令”字段中为工程设置更改权限。

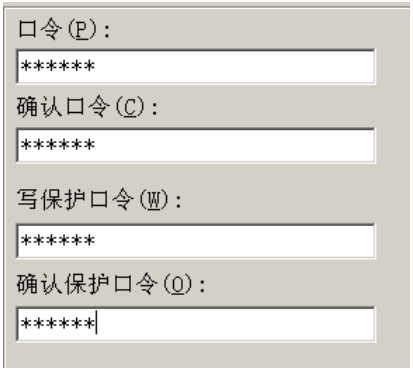


图 7-5-14 设定口令选项

输入口令时，对应每一个字符出现一个星号（*），在“确认口令”字段再次输入同样的内容。设定成功后，下次打开工程时，就需要通过口令验证。如果出现如图 7-5-15 所示的对话框，说明两次输入口令不一致，需要重新输入。

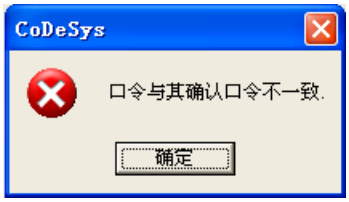


图 7-5-15 口令错误提示

对于一个写保护的工程，在没提供口令的情况下是不能被更改的。如果打开一个文件时系统提示输入写保护口令，而只是简单地点击“取消”按钮，那么工程中所有的编辑设定功能都禁用，只可以使用编译或仿真等查看功能。

如果存在多个用户，需要定义用户组，为每个用户组分配不同的访问权限。系统共提供了 8 个用户组，它们对工程对象拥有不同的访问权限。每个用户组的成员都通过口令来确认身份。“工程”/“用户组命令”用来设定用户组的口令。用户组从 0 到 7，0 组拥有管理员权限，只有 0 组的成员可以决定其它用户组和对象的口令和访问权，如图 7-5-16 所示。

建立一个新工程时，所有的口令初始状态都是空的。当一个口令被设置为 0 组，用户以 0 组成员的身份进入工程时，就可以对其它级别的组设定口令。如图 7-5-16 所示，在左边的组合框“用户组”中选定组，在右边输入相应的口令。

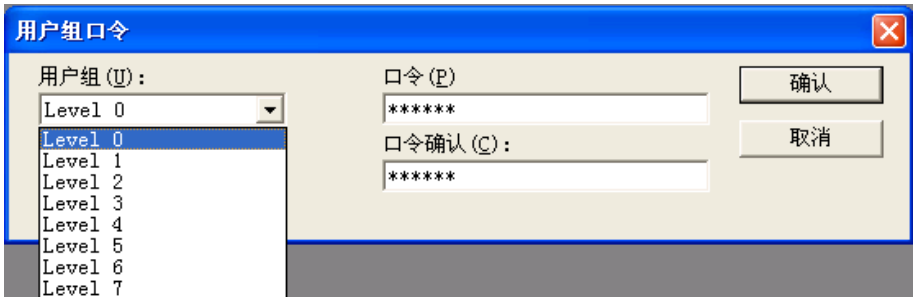


图 7-5-16 用户组命令

使用“工程”/“对象”/“属性”命令可以为用户组设置不同的访问权限，如图 7-5-17 所示。访问权限有三种可用的设置：

- 无访问权限：对象不允许被此用户组的成员打开。
- 只读权限：对象能够被此用户组的成员读入，但不能被更改。
- 完全权限：对象能够被此用户组的成员打开和更改。

访问权限的设置仅对选中的对象有效，每个 POU、硬件配置或全局变量都是对象成员。如果忘记了口令请及时与制造商联系，口令随工程一起保存。

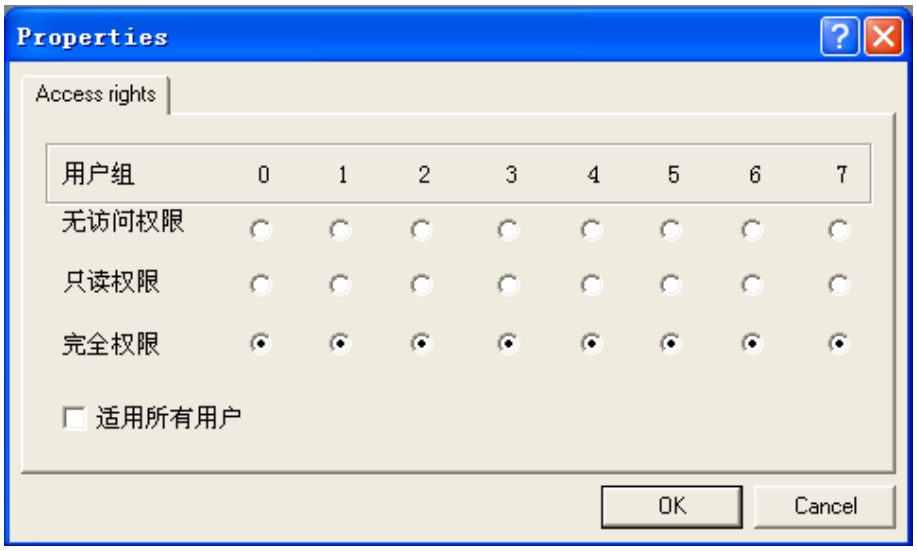


图 7-5-17 用户操作权限设定

下面是一个简单的用户组命令的应用例子。

打开“工程”/“对象”/“属性”，设置用户组权限，如图 7-5-18 所示。

设置用户组 0 和 1 的访问权限为完全权限。

设置用户组 2 和 3 的访问权限为只读权限。

设置用户组 4、5、6 和 7 的用户权限为无访问权限。

设定好用户组权限后，分别对各用户组设置相应口令。

图 7-5-18 设定用户组权限属性

这里简单设定用户组口令，如图 7-5-19 所示。

先对“Level 0”进行口令设置，在“口令”空栏处填入“0”，在“口令确认”空栏处填入“0”，点击“确认”，则“Level0”口令设置完毕。

同样，分别对 Level1、Level2 和 Level3 设置口令。用户组与口令的对应关系设置如表 7-5-1 所示。

由于用户组 4、5、6 和 7 的用户权限为无访问权限，所以无需设置其口令。

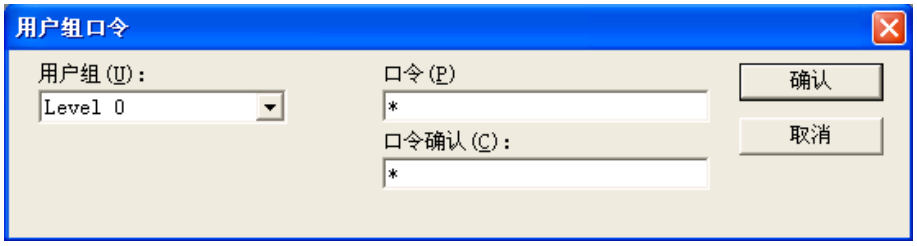


图 7-5-19 设定用户组权限口令

表 7-5-1 用户组与口令

用户组	口令
Level 0	0
Level 1	1
Level 2	2
Level 3	3

分别设置好不同用户组的口令后，在下一次打开程序时，会提示要求输入用户组口令。对于不同用户组，因其设置的口令不同，可以不同的身份进入程序，而且享有不同的操作权限。现以“Level0”的用户组进入，输入口令如图 7-5-20 所示。

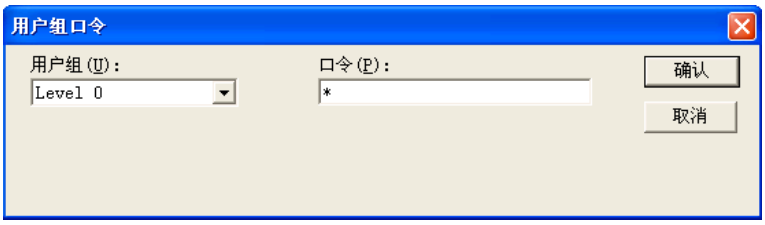


图 7-5-20 输入用户组权限口令

如果输入的口令与用户组 0 所设置的口令不同时，会出现如图 7-5-21 所示的提示。

图 7-5-21 输入用户组权限口令错误

如果输入的口令与用户组“Level0”所设置的口令一致，则会以用户组“Level0”的身份进入。对于用户组 Level4、Level5、Level6、Level7 和 Level8 的成员，由于无访问权限，则会弹出如图 7-5-22 所示的对话框。

图 7-5-22 用户组无访问权限

7.6 工程选项设置

“工程选项”有两种打开方式。一种是点击“工程”下拉菜单中的“选项”，另一种是双击“资源”选项卡中的“工程选项”，如图 7-6-1 所示。

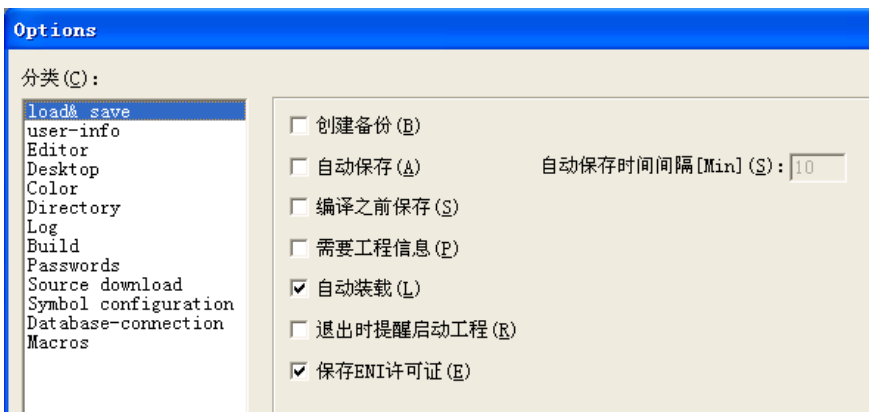


图 7-6-1 工程选项设置

打开“工程选项”设置对话框，该选项被分为几大类。每个选项对应右边的不同设置。这些选项包括：

- 下载与保存 Load&Save
- 用户信息 User Information
- 编辑器 Editor
- 窗口 Desktop
- 颜色 Color
- 目录 Directories
- 日志 Log
- 编译 Build
- 口令 Passwords
- 源代码下载 Source download
- 符号配置 Symbol configuration
- 数据库连接 Database-connection
- 宏 Macro

在上述选项中，PLC 不支持“源代码下载 Source download”、“符号配置 Symbol configuration”、“数据库连接 Database-connection”和“宏 Macro”，这里不作介绍。

下面分别对“下载与保存 Load&Save”、“用户信息 User Information”、“编辑器 Editor”、“窗口 Desktop”、“颜色 Color”、“目录 Directories”、“日志 Log”、“编译 build”和“口令 passwords”等进行介绍，其余均采用默认值。

7.6.1 下载与保存

在“工程选项”下，选择“Load&Save”选项，在窗口右侧显示其设置选项，如图 7-6-1 所示。下载与保存“Load&Save”的具体选项设置可以分为如下几类情况。

- 选中“创建备份”，系统会在保存工程的时候以相同路径建立一个扩展名为“.bak”的备份文件。通过这种方式可以保存最后一次保存之前的工程文件。
- 选中“自动保存”，那么工作时，工程会经常不断按照设定的“自动保存时间间隔”，以扩展名“.asd”保存临时文件。该文件会在程序正常退出时被删除，系统默认的保存路径为 D:\Hollysys\PowerPro\Projects。如果系统非正常退出（例如电源故障），文件就会被保留下来。当再次打开程序时，出现如图 7-6-2 所示的消息框，可以选择是打开自动保存文件还是打开原始文件。

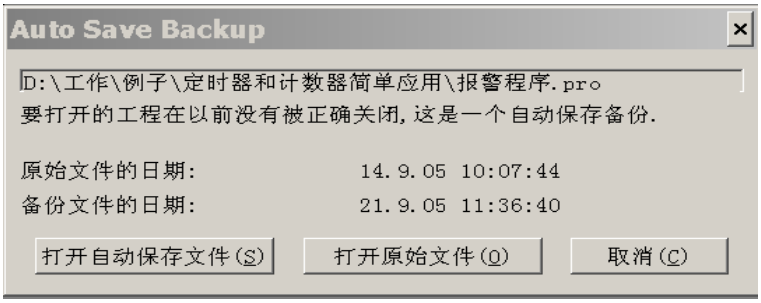


图 7-6-2 自动恢复提示

- 选中“编译前保存”，每次执行编译命令前，自动保存“*.asd”临时文件。
- 选中“需要工程信息”，那么当保存一个新工程或用新名字另存成一个工程时，工程信息对话框会自动被调用，相当于自动执行“工程”/“项目信息”命令。项目信息包含工程的标题、目录、作者和版本等信息，作为工程文件的一部分，可打印。
- 选中“自动装载”，软件在运行时会自动载入上一次最后打开的工程。
- 选中“退出时提醒启动工程”，如果工程已经改变和下载，但是自从上次下载启动工程之后没有创建新的启动工程，那么在用户退出前将会弹出如图 7-6-3 所示的对话框：“自上次下载后，没有创建启动工程，是否退出？”。这里对启动工程加以定义。所谓“启动工程”，即保存在 PLC 的 flash 中，上电后运行的用户程序。
- 选中“保存 ENI 许可证”，则会保存 ENI 许可证。

图 7-6-3 “退出时提醒启动工程”对话框

7.6.2 用户信息

在“工程选项”下，选择“user-info”选项，在窗口右侧显示其设置选项。用户信息的具体内容设置如图 7-6-4 所示。在相应的用户名、姓名缩写和公司等空格处填入相应的用户信息内容。

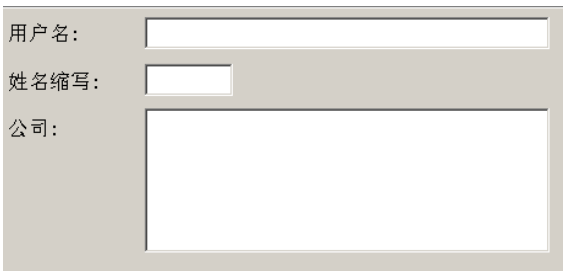


图 7-6-4 用户信息

7.6.3 编辑器

在“工程选项”下，选择“Editor”选项，在窗口右侧显示其设置选项。编辑器的设定包括自动声明、自动格式化、表元素、声明为表、标记、位值等选项，如图 7-6-5 所示。

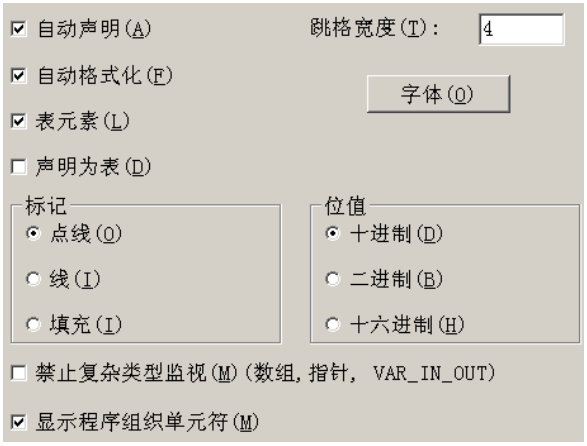


图 7-6-5 编辑器选项

- 选中“自动声明”，那么在用户编辑的程序中一旦出现了没有定义的新变量，系统会自动弹出对此变量定义的窗口，提示定义变量。
- 选中“自动格式化”，系统在编辑器中将会执行自动格式，即当编辑完成一行后，这一行会自动进行格式转化，将小写的操作数用大写表示，插入空格使各列均匀分开。
- 选中“声明为表”，声明编辑器将以表格的形式出现，如图 7-6-6 所示。用户可以采用填表的方式定义变量。这张表是卡片索引形式的对话框，共有六张变量卡和一张信息卡。变量卡分别是 VAR 本地、VAR_INPUT 输入、VAR_OUTPUT 输出、VAR_IN_OUTPUT 输入/输出、CONSTANT 常数和 RETAIN 保留型变量等。对于每一个变量，需要设置变量的名称、地址、类型、初始值和注释等字段。INFO 信息卡自动显示 POU 类型及名称。

图 7-6-6 表格声明

- “跳格宽度”设定在编辑器中空格的宽度，缺省的设置是 4 个字符宽。字符的宽度取决于选择的字体。
- “字体”按钮设定工作区域中所有窗口的字体。大字体放大了打印输出的大小，对每一种编辑器都适用。
- “标记”设定在图形编辑器中当前选择是由虚线矩形表示，还是由实线矩形表示，或是由填充矩形表示。
- “位值”设定在线模式下变量（类型 BYTE、WORD 和 DWORD 等）显示的格式是十进制[D]数，还是二进制[B]数，或是十六进制[H]数。
- ✧ 举例

十进制[D]	二进制[B]	十六进制[H]
a=53	a=2#0000 0000 0011 0101	a=16#0035
b=57	b=2#0000 0000 0011 1001	b=16#0039

- “禁止复杂类型监视（数组，指针，VAR_IN_OUT）”PLC 不支持此功能。

- 选中“显示程序组织单元符”选项，则在程序编辑过程中，如果调用相应的功能块，会自动添加“PowerPro/Library”路径下的 bmp 文件，丰富功能块内容，增加软件的友好性。

7.6.4 窗口

在“工程选项”下，选择“Desktop”选项，在窗口右侧显示其设置选项，如图 7-6-7 所示。

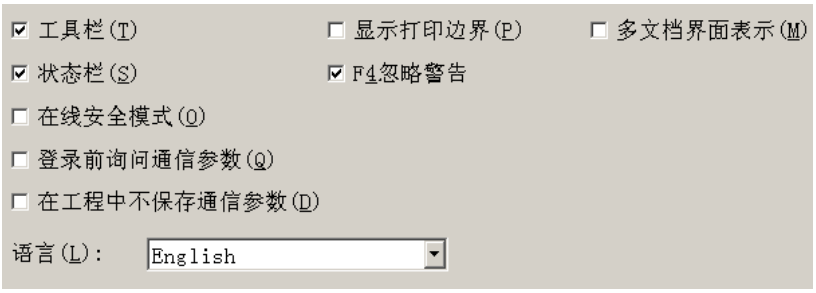


图 7-6-7 窗口选项

- 选中“工具栏”，在主窗口中出现工具栏，显示菜单的常用命令快捷按钮。
- 选中“状态栏”，在主窗口的最下部出现状态栏。
- 选中“在线安全模式”，当执行在线命令时，例如“运行”、“停止”、“复位”或“断点”等在线命令，会弹出确认对话框，再次确认操作。这样可以进一步提高安全性，避免误操作。
- 选中“登录前询问通信参数”，则会在登录时提示通讯参数对话框，方便校准通讯参数，例如参数是否设置及其设置的正确性。这就无须再打开“在线”菜单来设置“通讯参数”。否则，则不具备此项功能。
- 在“语言”选项中，只能选择菜单和对话框的显示语言，默认为 English。
- 选中“显示打印边界”，在每一个编辑窗口，将每页打印纸可打印区域用红色虚线框出。这个区域的大小取决于打印机属性（纸张、布局）和“Content”区域的大小（文件/打印机设置），如图 7-6-8 所示。
- 选中“F4 忽略警告”，当程序编译后，在消息窗口使用快捷键 F4 快速跳转时，会略过出现警告（warning）的行，只跳到出现错误（error）的行。

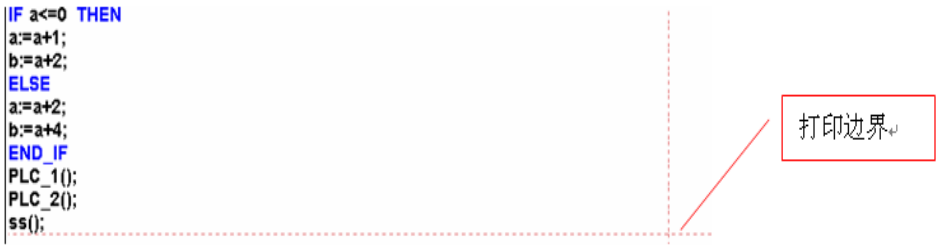


图 7-6-8 可打印区域显示

7.6.5 颜色

在“工程选项”下，选择“Colors”选项，在窗口右侧显示其设置选项，如图 7-6-9 所示。

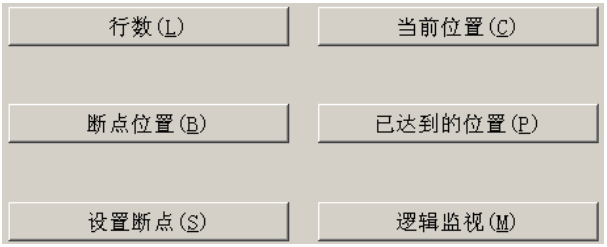


图 7-6-9 颜色选项

根据用户的需求和习惯，可对系统的某些特定显示颜色进行重新设置。采用颜色标识，主要针对在线调试的情况，可以直观方便地观察程序当前运行的情况。点击对应选项按钮，打开颜色对话框，按照特定需求设定相应的颜色。一般情况采用默认值。

- “行数”：缺省浅灰色，程序编辑器中网络号或行号的背景颜色。
- “当前位置”：缺省红色，在线运行时，遇到断点停止的网络号或行号的背景颜色。
- “断点位置”：缺省深灰色，可设置断点的网络号或行号的背景颜色。
- “已到达的位置”：缺省绿色，显示流控制时，已执行的网络号或行号的背景颜色。
- “设置断点”：缺省浅蓝色，设置了断点的网络号或行号的背景颜色。
- “逻辑监视”：缺省深蓝色，在线模式下，数字逻辑 TRUE 的颜色。

7.6.6 目录

在“工程选项”下，选择“Directory”选项，在窗口右侧显示其设置选项，如图 7-6-10 所示。在“工程”栏内可以设置工程的库、编译文件、配置文件和文件视图等所在的目录。在“目标”栏内显示“配置文件”所在目录，对于“库”默认为无路径，此项是系统自动生成的，不可修改。PowerPro 软件安装时自动生成的文件目录会显示在“公用”栏内的库、编译文件、上传文件、配置文件和文件视图等的目录中。

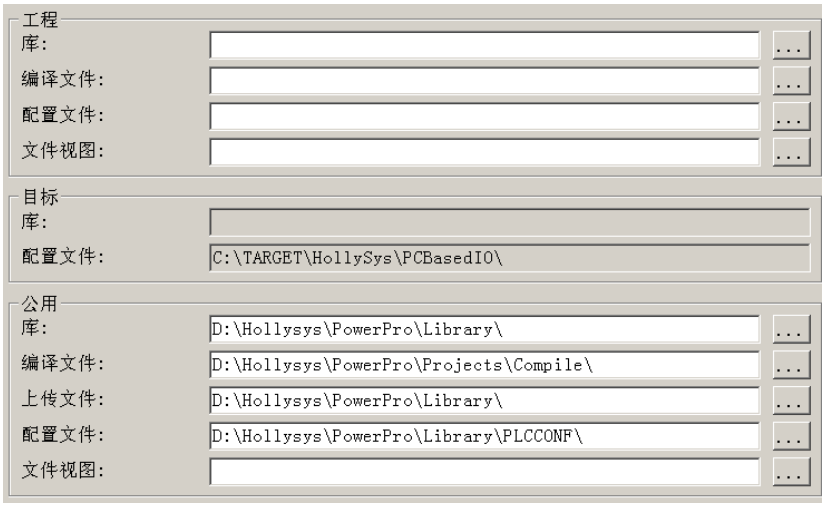


图 7-6-10 目录

7.6.7 日志

所谓“日志”表示工程按一定的年月日顺序记录用户所作的一系列动作。其中记录内容包括登录、运行、初始化、输入值、退出、清除缓存区和无法登录等。在“工程选项”下，选择“Log”选项，在窗口右侧显示其设置选项，如图 7-6-11 所示。

图 7-6-11 日志选项

- 选中“工程日志目录”，可以修改工程日志的保存路径，其默认的路径为D:\Hollysys\PowerPro。
- “最大工程日志”设定日志窗口中最多显示的在线会话的个数。
- 选中“激活日志”，在工程中启用日志记录功能，显示日志列表，如图 7-6-12 所示。在日志中分别记录了登录、运行、初始化、输入值、退出和清除缓存区等一系列用户动作、状态变化或内部动作。

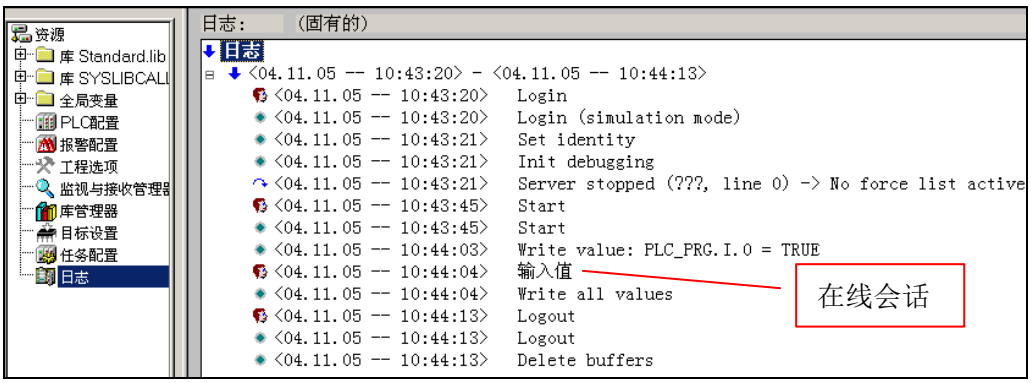


图 7-6-12 日志列表

- 在“选择”栏中，选定每个在线会话所记录的内容，包括用户动作、状态变化、内部动作和例外等。
- “日志”窗口在“离线”和“在线”两种模式下均可使用。图 7-6-12 所示为一个在离线仿真模式下产生的日志列表。

7.6.8 编译

在“工程选项”下，选择“Build”选项，在窗口右侧显示其设置选项，如图 7-6-13 所示。

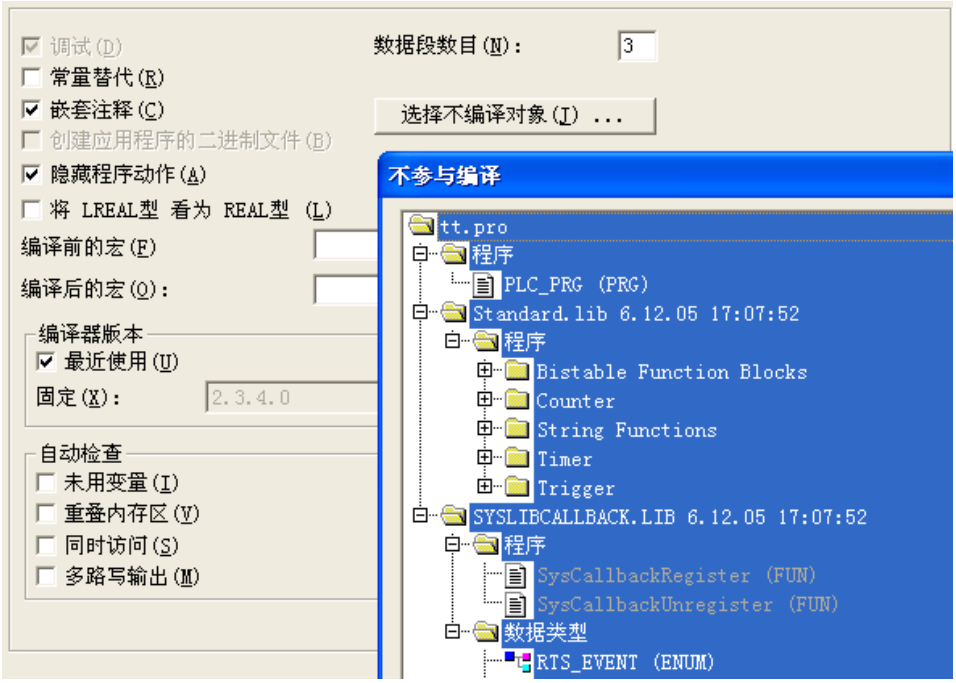


图 7-6-13 编译

- “调试”：当此项被选中时，附加的调试代码会增加代码段长度。当此项未被选中时，会加快工程的速度，减少代码段长度。另外，只有选中该项，登录后才可以运行“断点”和“单步”等调试功能。
- “常量代替”：每个常量的值是直接下载的，在“在线模式”下，常量的值为绿色，强制值、输入值和监视常量不再有效。如果此选项无效，则该常量值会通过变量访问下载到存储区。
- “嵌套注释”：允许使用嵌套的注释语句。

✧ 举例：嵌套注释

```
(*
a:=inst.out;   (*检查*)
b:=b+1;
*)
```

- “数据段数目”：用于设置编译的数据段的个数。
- “编辑器版本”：如果选中“最近使用”，会给出最近使用的版本号。
- “自动检查”：

选中“未用变量”，在编译时会自动检查未使用的变量。
选中“重叠内存区”，在编译时会自动检查重叠内存区。
选中“同时访问”，在编译时会自动检查同时访问。
选中“多路写输出”，在编译时会自动检查多路写输出。

“自动检查”与“工程”菜单栏里的“查看”具有同样的功能。唯一不同的是，“工程/查看”只有在编译后，可以通过“查看”选择要查看的内容，例如未用变量等，而且只能一一查看。而自动检查则可以选择多项，在编译时一起实现自动检查。

图 7-6-14 所示为当上述四项均选中，在编译时，消息窗口自动生成的检查情况。

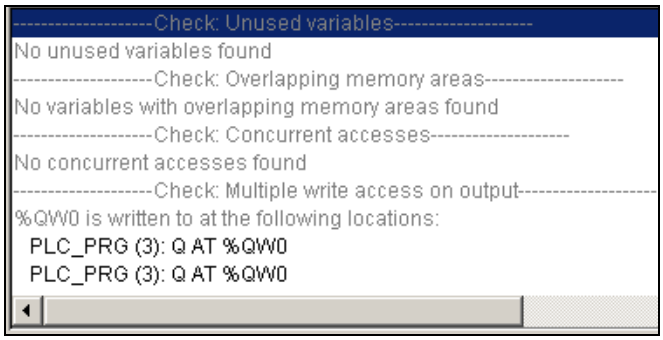


图 7-6-14 编译消息

7.6.9 口令

如果程序需要加密，则需要对该程序设置相应的口令。

在“工程选项”下，选择“Passwords”选项，在窗口右侧显示其设置选项，即可设置相应的口令。口令代码完全根据个人需求自行设置。如果保密性强，建议多设置几位口令代码。在图 7-6-15 所示的例子中，设置了六位代码的口令，下面介绍其使用方法。

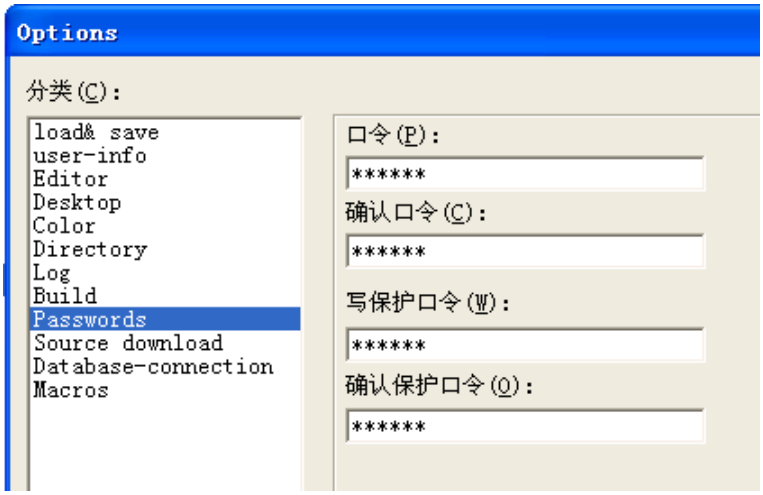


图 7-6-15 口令设置

在对程序进行口令设置且保存后，在下一次打开该程序时，系统会提示要求输入口令。输入相应的口令，点击“确认”，如图 7-6-16 所示。

图 7-6-16 输入口令

点击“确认”后，窗口会弹出如图 7-6-17 所示的对话框，要求输入写保护口令。如果所输入的口令正确，则程序被打开，而且可以进行编辑。

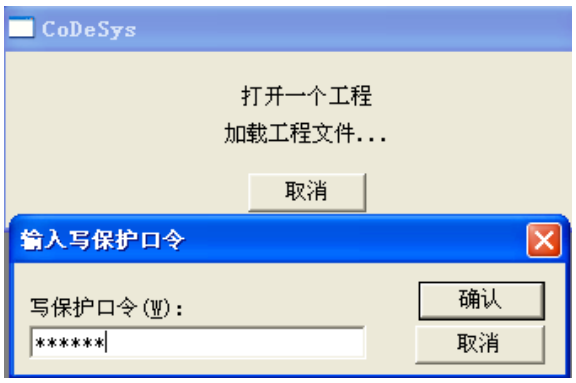


图 7-6-17 输入写保护口令

对于已经加密的程序，如果现在不需要加密，则只需打开“工程选项”\“Passwords”，将以前设置的所有密码删除并存盘。当再次打开该程序时，密码保护功能失效。

第8章　编译与调试

当程序编写完毕后，要对程序进行编译。当编译通过后，才能将程序下载到 PLC 中。本章主要对 PowerPro 软件的编译与下载过程进行介绍。

8.1 编译

PowerPro 软件的“工程”菜单提供了“编译”和“全部编译”两种编译命令，用于检查程序有无语法错误，如图 8-1-1 所示。

- 编译：仅对程序的变化部分进行编译，并更新到原有的目标文件中。
- 全部编译：与“编译”命令不同，可以完全重新编译整个工程。
- 清空：清除上次编译和下载的信息。
- 载入在线修改信息：PLC 不支持此项功能。

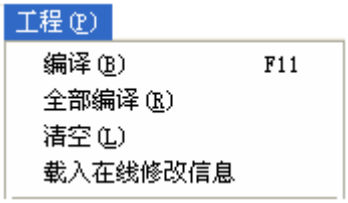


图 8-1-1 工程菜单

没有语法错误的程序才能生成可执行的目标文件。编译的结果会显示在消息窗口中，如图 8-1-2 所示。

常见编译错误信息参见

图 8-1-2 编译信息显示

8.2 显示参考数据

PowerPro 软件的“工程”菜单提供显示一些参考数据的命令，这些命令只有在编译通过后才有效。

8.2.1 查看调用树

“工程”/“查看调用树”命令可以在一个新窗口中显示当前对象调用程序、函数和功能块的树型结构，直观地指出当前 POU 与工程中其它 POU 的先后调用关系，如图 8-2-1 所示。工程必须通过编译，该命令才有效。

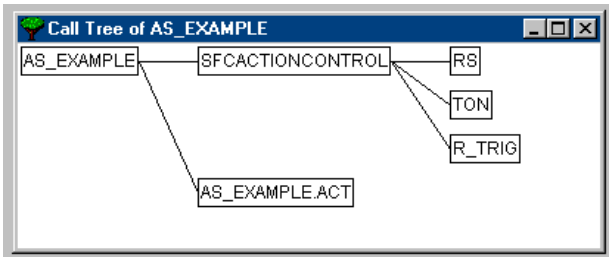


图 8-2-1 查看调用树

8.2.2 查看交叉引用列表

“工程”/“查看交叉引用列表”命令可以显示并查看所有应用程序点，如图 8-2-2 所示。所谓“应用程序点”是指某一个变量、地址或程序在全部工程中的位置。工程必须通过编译，该命令才有效。

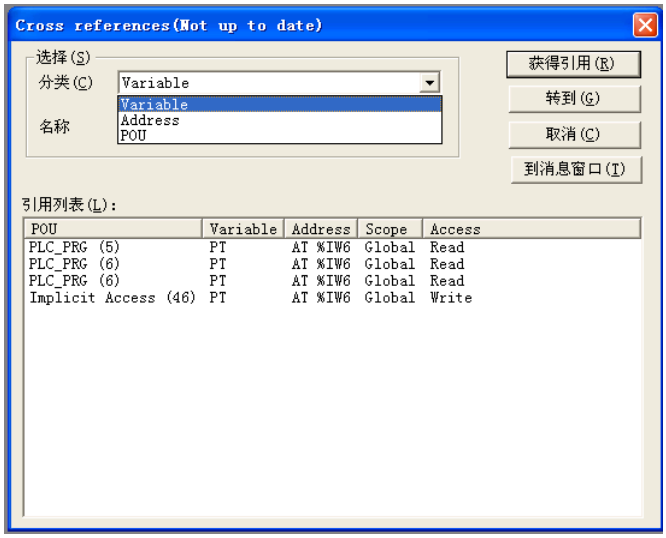


图 8-2-2 查看交叉引用列表

首先需要选择变量（Variable）、地址（Address）或程序（POU）的类别，然后输入名称，最后点击“获得引用”按钮，在“引用”栏中就得到相应的应用程序点列表，列出该点的信息。例如，是只读还是可写，是局部变量还是全局变量，变量是否被连到硬件地址上等信息。

选中交叉参考列表中的一行，按“转到”按钮，或者直接双击此行，程序会在编辑器中显示出相应的点。通过这种方式，可以任意跳到所需的应用程序点处，而无须进行费时的搜索。还可以使用“到消息窗口”按钮，使交叉参考列表直接显示在消息窗口中。鼠标双击跳到相应的程序位置，搜索起来更加方便。

8.2.3 查看

“查看”菜单，仅适用于“仿真模式”下，如图 8-2-3 所示。

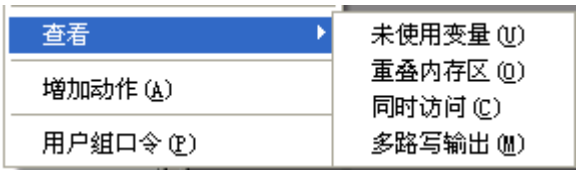


图 8-2-3 查看菜单

➢ “查看”/“未使用变量”

编写算法时，常常会删除某个变量，或重新命名某个变量，此时原变量的声明不会被自动删除，仍保留在声明编辑窗口中。这样，工程里就可能存在只有声明却从未被使用过的变量，占用内存空间，也不利于变量的管理。

“工程”/“查看”/“未使用变量”命令用来检查工程中是否存在只有声明却没有被使用的变量。工程必须通过编译，该命令才有效。检查结束后，消息窗口出现所有未使用变量的列表，如图 8-2-4 所示。

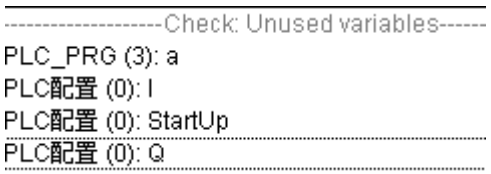


图 8-2-4 查看未使用变量

➢ “查看”/“重叠内存区”

选择此项，工程自动查看重叠内存区。如果没有重叠内存区，则会在消息窗口显示如下提示：No variables with overlapping memory area found（没有重叠内存区）。

➢ “查看”/“同时访问”

选择此项，工程自动查看有无同时访问，如果没有同时访问，则会在消息窗口显示如下提示：No concurrent accesses found（没有同时访问情况）。

➢ “查看”/“多路写输出”

选择此项，工程自动查看多路写输出情况，如果没有检查多路写输出情况，则会在消息窗口显示如下提示：No outputs found which are written to at more than one location（没有多路写输出）。

8.3 下载

8.3.1 设备安装与连接

➢ 设备安装

首先根据实际工程的需要，选择合适的 CPU 模块和扩展模块。然后根据现场情况确定模块的安装方式，并初步确定 PLC 的工作方式。最后规划并制定合理的接线方案，将现场的传感器或执行器连接到 PLC 模块的接线端子上。

➢ 连接电缆

根据所选 CPU 模块的型号和类型，连接电源线，如图 8-3-1 所示。电源线接好之后，先不要接通电源。在检查所有电缆连接无误后，再接通系统电源，并确认 CPU 模块面板上的 RUN 指示灯点亮，并显示正常，以保证 PLC 可靠运行。注意，当电源线连接好之后，应该把端子盖扣好，以免造成不必要的人身伤害或设备损坏。

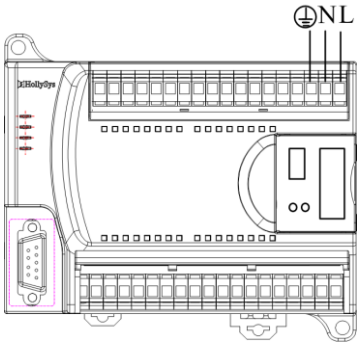


图 8-3-1 连接电源线

➢ 建立 PC 通讯

通过配套的编程电缆，将 CPU 模块连接到个人计算机（PC）的 RS232 串行通信接口，建立数据传递通道，如图 8-3-2 所示。由于 CPU 模块的 RS232 串行通讯接口是非隔离的，所以编程电缆的连接应该在 PLC 上电之前进行。

注意，CPU 模块 LM3108 和 LM3109 有 2 个串口，通过左边的 PORT1 串口将程序下载到 PLC 中。

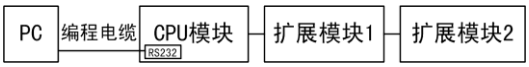


图 8-3-2 连接编程电缆

8.3.2 建立通信连接

将编译生成的目标文件下载到 CPU 模块中去，需要配置并选定通讯线路，建立本地计算机与目标模块之间的通信连接。实现步骤如下所述。点击“在线”菜单中的“通讯参数”，弹出“Communication Parameters”通讯参数对话框，如图 8-3-3 所示。

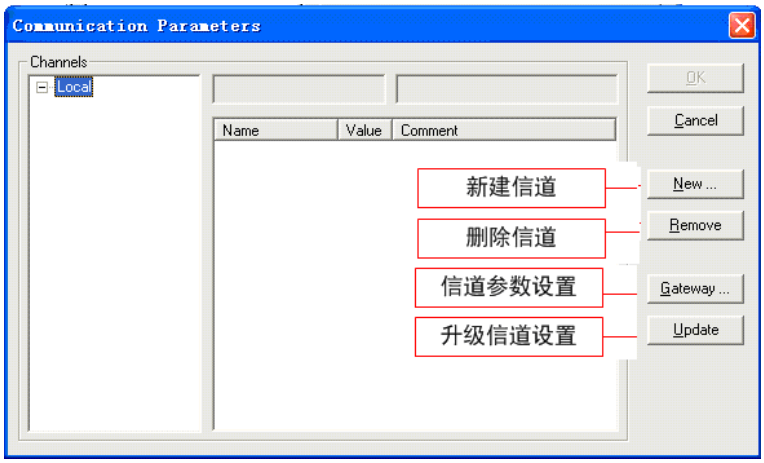


图 8-3-3 通讯参数对话框

点击“Gateway”进入信道参数设置，确认“Connection”通道参数设置为“Local”，点击“OK”按钮，如图 8-3-4 所示。

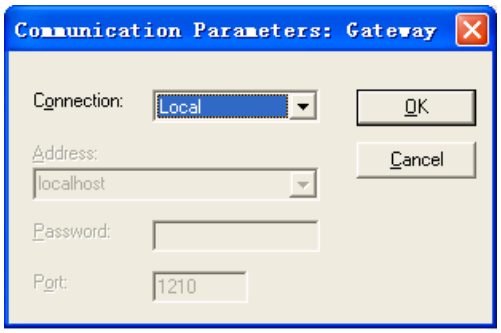


图 8-3-4 通道参数设置

确认后返回“Communication Parameters”通讯参数对话框，点击“New”按钮添加新信道，出现如图 8-3-5 所示的对话框。其中，信道名称默认为“Local_”，通信协议使用缺省的 RS232 协议，点击“OK”按钮返回“Communication Parameters”通讯参数对话框。

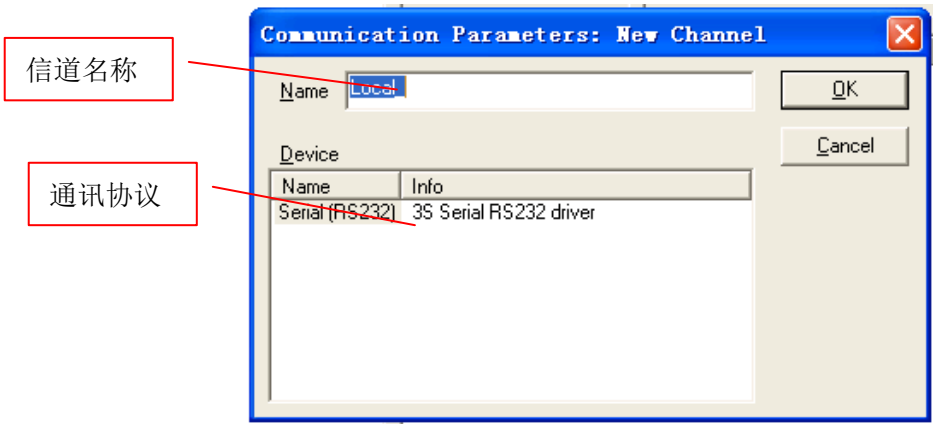


图 8-3-5 添加新信道

改变通讯速率。连续双击“Baudrate”中“Value”的对应值，使其变为“38400”，如图 8-3-6 所示，点击“OK”按钮确认。这样就建立了本地计算机与 PLC 的 CPU 模块之间的通信连接。

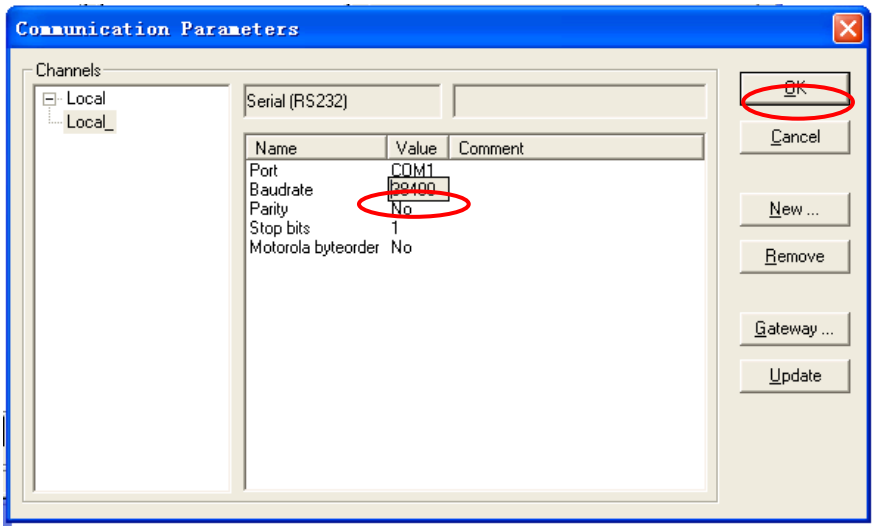


图 8-3-6 通讯速率设置

8.3.3 程序下载

➢ 下载程序

点击“在线”菜单中“登录”，可以实现程序的下载。编译通过的目标文件在进行下载时，会将全部的目标文件下载到模块中。同时将模块复位，所有变量返回到初始状态。在“在线”菜单中选定“登录”，建立本地计算机与 CPU 模块的连接，并出现系统提示下载信息，如图 8-3-7 所示。

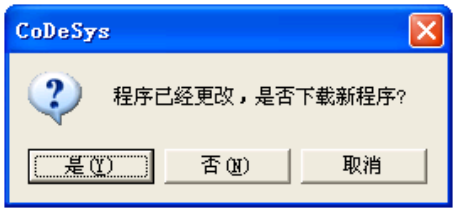


图 8-3-7 下载提示信息

当下载的 PLC 程序与 PLC 内部不符的时候，就会弹出该对话框。

如果要更改程序，选择“是”按钮，将新程序下载到 CPU 模块中去。如果不想下载到 PLC 中，只想建立连接，则选择否。

点击“是”按钮，下载到 PLC 后，出现如图 8-3-8 所示的创建启动工程提示信息。启动工程是指下载至 Flash 中的程序。PLC 为了保证在断电重新上电后，程序不丢失，会在 Flash 中重新创建程序，称为启动程序。

在图 8-3-8 中，点击“是”按钮，确保 PLC 断电后再上电时，运行此下载工程。下载结束。

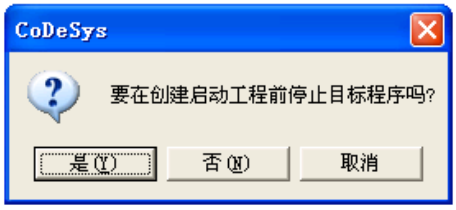


图 8-3-8 创建启动工程提示信息

➢ 不下载而进入在线监控状态

若不希望下载程序，仅想在线监控 PLC 内的数值，可以在系统提示下载信息时选择“否”，此时对工程的修改不会被下载到模块中，但可以监控模块中原有程序的运行情况。注意，如果在线监控之后仍然需要下载程序，可以选择“在线”下拉菜单中的“下载”来实现下载。

➢ 登录、下载、创建启动工程的区别

为了解释清楚这几个名词，需要了解一下 PLC 程序的下载过程。PowerPro 需要将编译好的程序下载到 PLC，首先需要将 PowerPro 与 PLC 之间建立一个连接。“登录”命令的作用就是将 PLC 和 PowerPro 建立一个连接。建立完连接后，PowerPro 会自动判断程序是否改变。若程序没有改变，则自动转入在线监控状态，通过 PowerPro 就可以在线监控当前 PLC 的状态。若程序发生了改变，则会提示 “是否要下载新的程序”，选择“否”，则不下载程序，转入在线监控状态；这时若想下载程序，可以通过在线菜单中的“下载”命令下载程序。也就是说，“下载”命令只有在登录以后才有效。若选择“是”，则会把修改后的程序下载指 PLC 的 CPU 内。但是因为 CPU 断电后，数据全部清除，所以为了保证程序在 PLC 重新上电后依然存在，在下载到 CPU 后，还需要将程序保存到 FLASH 内，这个过程叫创建启动工程。在下载至 CPU 完成后，PowerPro 还会弹出一个对话框，是否在创建启动工程是停止目标文件，无论选择是还是否，都会创建启动工程。若选择“取消”，则不会创建启动工程，这样在断电重新上电后，刚下载的程序不会保存，而是之前保存的原始程序。此时，可以点击创建启动工程命令，重新往 FLASH 中创建程序。

8.4 调试

系统所支持的调试命令都在“在线”菜单下，并且在调试状态下可以使用，如图 8-4-1 所示。在调试状态下，系统用不同的默认颜色来代表不同的状态和操作，例如逻辑真（蓝）、逻辑假(黑)、断点（浅蓝）、流控制（绿色）等等，这些颜色可以在“工程”/“选项”/“Color”中设定。了解各种颜色的含义，有利于程序的调试和监视。

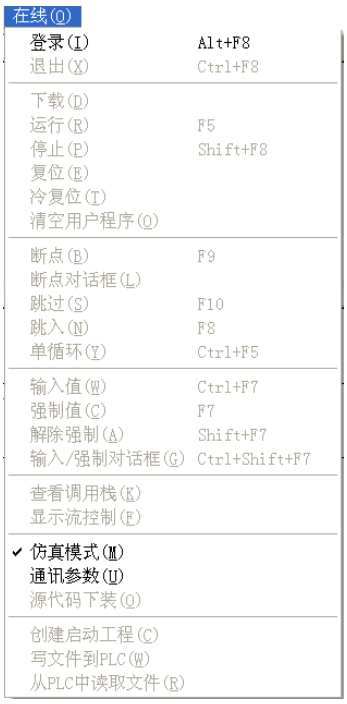


图 8-4-1 在线菜单

8.4.1 进入调试状态

执行“在线”/“登录”命令进入调试状态。将程序下载到 PLC 的 CPU 硬件模块中，称为在线调试状态。如果没有连接硬件模块，在本地计算机模拟运行用户程序，称为仿真模式。“在线”/“仿真模式”被选中（出现“√”），登录时便会进入仿真模式。

在程序中，如果调用了与 RTS 上硬件相关的功能块，则不可以使用仿真模式。例如，系统事件不可以使用仿真模式。与 RTS 上硬件相关的功能块包括定制库中的外部扩展功能块和外部功能块，均不可使用仿真模式。

另外，设定自由口参数后，如果想要恢复原编程系统的下载与调试功能，需要将 RUN/STOP 开关拨到 STOP 位置，才可以进行编程系统登录。

8.4.2 退出调试状态

执行“在线”/“退出”命令退出调试状态，进入编程状态。

8.4.3 运行程序

执行“在线”/“运行”命令，启动下载到模块中的程序，或者启动处于仿真模式下的程序。一般需要运行程序的情况有以下几种：

- “在线”/“登录”之后。
- 用户程序使用“停止”命令终止之后。
- 用户程序设置了断点之后。
- 执行一个“单循环”之后。

8.4.4 停止程序

执行“在线”/“停止”命令，暂停程序在模块或在仿真模式下的运行，保存当前变量值。此时程序仍处于调试状态下，使用“在线”/“运行”命令可重新启动程序，从上次停止的地方开始继续运行。

8.4.5 复位

如果已经明确地定义了变量的初始值，“复位”命令会把当前变量的值设置为初始值，保留型变量的值保持当前值。按“F5”键，程序会重新按照复位后的初始值运行。

- “冷复位”命令重置所有的变量为初始值，包括保留型变量。只有常量在“冷复位”后保持原来的值不变。按“F5”键，程序会重新按照冷复位后的初始值运行。
- “清空用户程序”命令重置所有的变量为初始值，包括保留型变量和常量，并删除模块中的用户程序，模块返回到初始状态。

以上命令都会出现对话框，需要进一步确认复位操作，如图 8-4-2 所示。

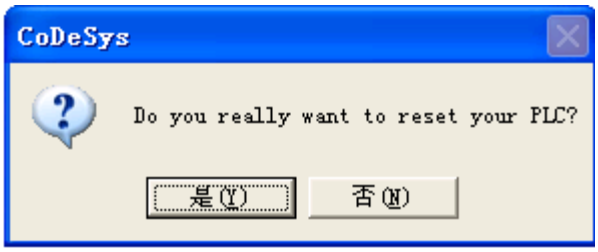


图 8-4-2 复位确认对话框

8.4.6 断点

断点是指可以在程序中设置进程停止的地方。通过设置断点，可以在程序的具体地点观察当前的变量值，便于分段调试程序。

➢ 设置断点

执行“在线”/“断点”命令，可在当前位置设置或删除一个断点。断点的设置位置取决于活动窗口中程序的语言类型。黑色背景的行号（网络号）表示该行可以设置断点，而灰色背景的行号（网络号）表示该行不可设置断点，如图 8-4-3 所示。

图 8-4-3 断点设置（1）

在 IL 和 ST 中，断点设置在当前光标所在的行。在 FBD 和 LD 中，断点设置在当前被选定的网络中。在 SFC 中，断点设置在当前选定的步上。

可以直接用鼠标点击黑色背景的行号（网络号）设置或删除断点。如果该行被设置为断点，对应的行号（网络号）会呈现浅蓝色背景。

当程序运行到达所设置的断点处终止时，相应的行号（网络号）以红色背景来显示。使用“在线”/“运行”、“跳进”或“跳出”命令，可以继续运行程序。

例如，“登录”后在 0004 行设置一个断点，如图 8-4-4 所示。

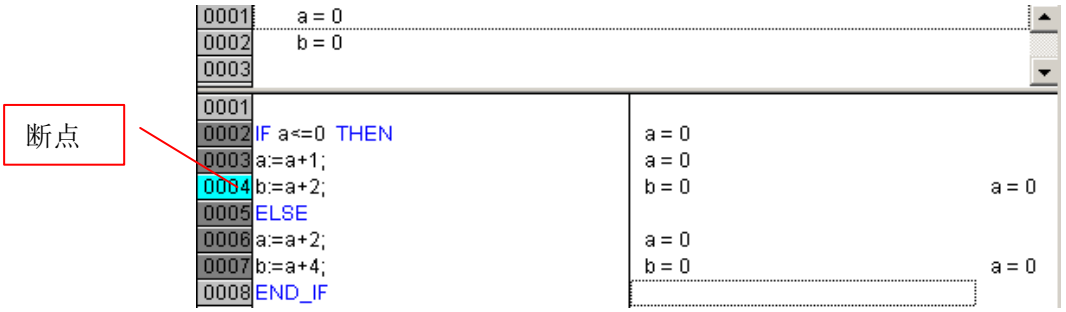


图 8-4-4 断点设置（2）

运行程序，执行到 0004 行断点处时，程序中断，变量 b 保持初始值不变，而上一行的变量 a 被重新赋值。如图 8-4-5 所示。

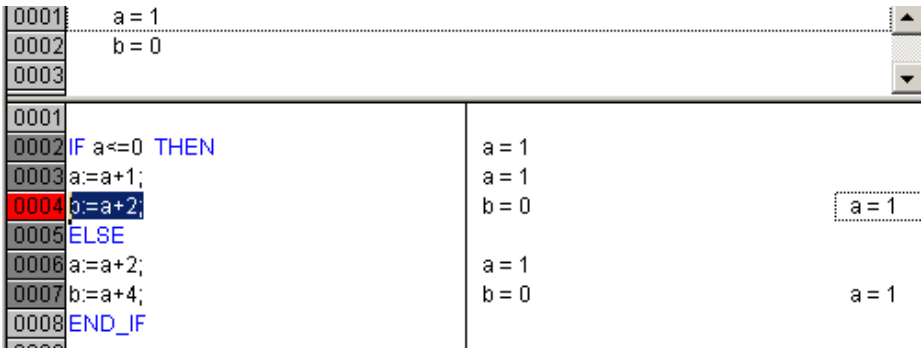


图 8-4-5 断点设置（3）

➢ 编辑断点

执行“在线”/“断点对话框”命令，弹出如图 8-4-6 所示的断点对话框，可以显示和编辑整个工程的断点。

设置断点：在“程序组织单元”项中选择一个需要设置断点的程序，在“位置”项中选择设置断点的行号（网络号），点击“添加”，则断点被添加到列表中。

删除断点：选中断点，点击“删除”，则删除该断点。

查看断点：选中断点，点击“转到”，则立即跳转到编辑器中断点设置的位置。

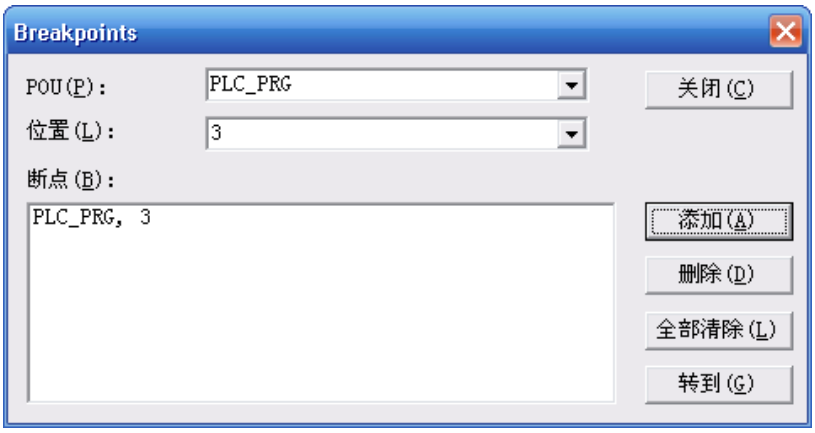


图8-4-6 设置断点对话框

当“在线”退出之后，所设置的断点被删除。注意，断点的个数应当限制在 100 以内，如果超出范围，则会出现如图 8-4-7 所示的提示。

图8-4-7 断点设置（2）

8.4.7 单步

利用单步可以逐步检查程序逻辑的正确性。在对应的活动窗口中，对于不同的编程语言，单步的含义有所不同。

- 在 IL 中，执行程序直到下一个 CAL、LD 或者 JMP 命令。
- 在 ST 中，执行下一个指令。
- 在 LD 和 FBD 中，执行下一个节。
- 在 SFC 中，继续执行动作，直到下一步。

执行“在线”/“跳出”或“跳进”命令实现单步执行。当遇到断点时，程序执行当前语句后停止。当遇到功能块或函数时，“跳出”命令会跳过功能块或函数执行下一条语句，而“跳进”命令则跳入功能块或函数的内部单步执行。

8.4.8 单循环

执行“在线”/“单循环”命令使程序运行完一个周期后就停止运行。从本质上来说，“单循环”命令等同于“运行”命令，都可以使用户程序在线运行。只不过“单循环”命令让程序运行完一个周期后就自动停止，而“运行”命令让程序循环运行，直到执行“停止”命令为止。

8.4.9 变量输入值

- 设定新值

在调试状态下，用鼠标直接双击声明编辑器中的变量，弹出写变量对话框，如图 8-4-8 所示。在“新值”中填入变量的新值，点击“确认”按钮，当前变量的后面即出现淡蓝色文本标示的新值。对于布尔型变量，双击后直接在当前值的后面用淡蓝色文本标出 TRUE/FALSE 的状态切换，不使用写变量对话框输入。

- 激活新值

执行“在线”/“输入值”命令，或使用快捷键“Ctrl+F7”，变量的新值才能被激活，即输入到模块中。允许对在多个变量写入新值后执行“输入值”命令，将多个新值同时输入模块。

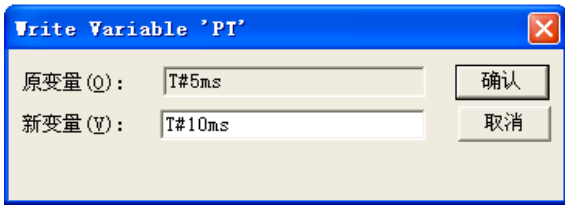


图8-4-8 写变量

8.4.10 变量强制值

➢ 强制值

强制写变量的方法与“输入值”命令一样，先输入新值，然后执行“在线”/“强制值”命令，或者使用快捷键“F7”写入强制值。

被强制的变量，在程序的每个循环之后都被写入强制值，直到执行“解除强制”命令后为止。对于“输入值”命令，变量只被写一次，而且允许变量被其它程序赋值。

当开关量的输入为物理点输入时，例如 I0.0，必须使用强制值，但在仿真模式下可以使用输入值。允许多个变量写入新值后执行“强制值”命令。

变量被强制时，其值用红色文本表示，如图 8-4-9 所示。

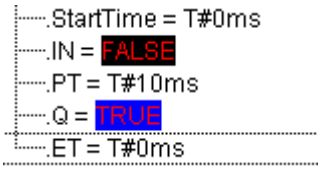
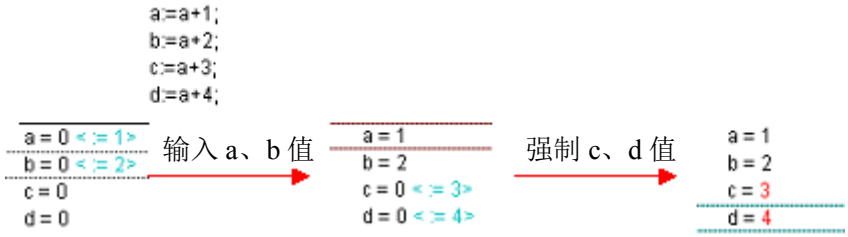


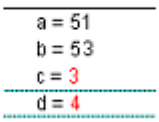
图8-4-9 强制值

通过下面这个例子，可以清楚地看出“输入值”和“强制值”的差别。

程序主体：



运行后显示：



“输入值”命令只改变了变量 a、b 的当前值，程序一旦运行起来，变量 a、b 将会按照程序的设定自行累加，每执行一次，程序重新赋值一次。而“强制值”命令使变量 c、d 在每次程序执行时都被赋予强制值，所以一直保持 3、4 不变。

➢ 解除强制

执行“在线”/“解除强制”命令，终止对变量的强制赋值命令。强制赋值被解除后，变量值恢复黑色。

➢ 输入/强制对话框

执行“在线”/“输入/强制对话框”命令，则会弹出如图 8-4-10 所示的对话框，其中包括监视列表选项卡和强制列表选项卡。

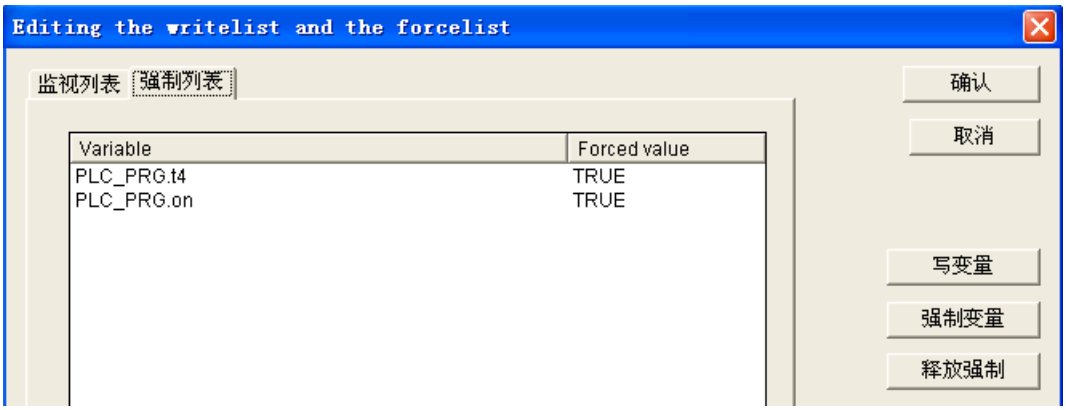


图8-4-10 输入/强制对话框

8.4.11 查看调用栈

执行“在线”/“查看调用栈”命令，则会在某个断点处，程序停止执行，显示调用堆栈里的当前程序的列表，如图 8-4-11 所示。

列表中第一个程序是系统默认的程序，或者是在“任务配置”中设定的第一个被调用的程序。列表中最后一个程序是当前被执行的程序。

选定了一个程序后，点击“跳转到”按钮，则立即跳到该程序的执行窗口，显示正在被处理的行或段。

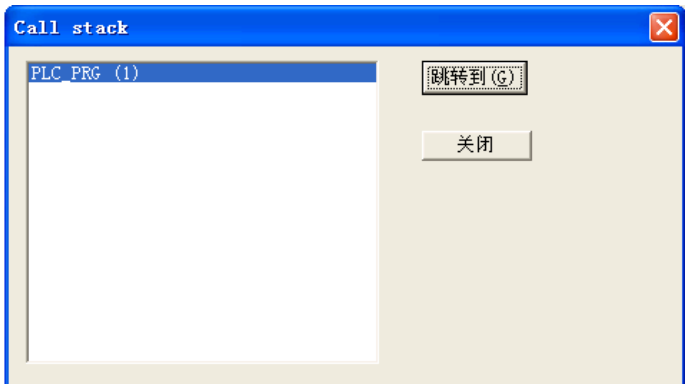


图8-4-11 堆栈调用

8.4.12 显示流控制

如果选择了流控制，“在线”/“显示流控制”菜单命令前会有一个“√”号出现。此后，在当前循环中，被执行的每一行或每一个网络都会被做上标记，运行的行和行号（或网络号）以绿色显示。在“单循环”模式下，用该命令可非常直观地看到程序的当前运行流程。

图 8-4-12 所示为一个简单的小程序，定义初值为 0 的整型变量 a。当 a<=0 时累加 1，当 a>0 时累加 2。

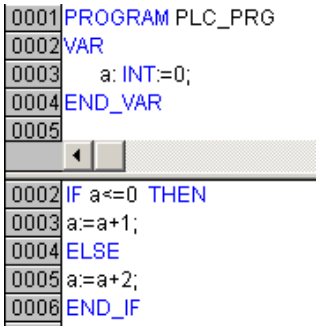


图8-4-12 程序示例（1）

“登录”后选中“显示流控制”命令，然后选择“单循环”方式。由于 a 的初值为 0，所以在第一次运行时，a 满足条件<=0，累加 1，当前 a 值变为 1。“显示流控制”命令将本次循环执行的行（2 行和 3 行）用绿色标示，如图 8-4-13 所示。

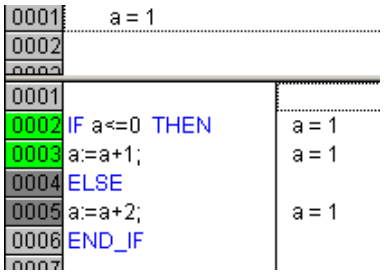


图8-4-13 程序示例（2）

再次选择“单循环”，此时由于 a 的值已经变为 1，满足程序中“ELSE”的条件，累加 2，当前值变为 3。“显示流控制”命令将本次循环执行的行（2 行和 5 行）用绿色标示，如图 8-4-14 所示。

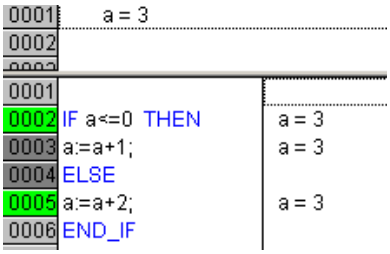


图8-4-14 程序示例（3）

8.4.13 监视与接收管理器

在对象组织器的“资源”选项卡中，可以打开监视与接收管理器的窗口，如图 8-4-15 所示。在调试过程中，可以在监视与接收管理器的窗口中集中监控工程中各程序的变量。监视与接收管理器可以将数值预置入某个变量，然后把它们作为一组发送到模块。同样，当前模块的值可以读入和存储在监视与接收管理器中。

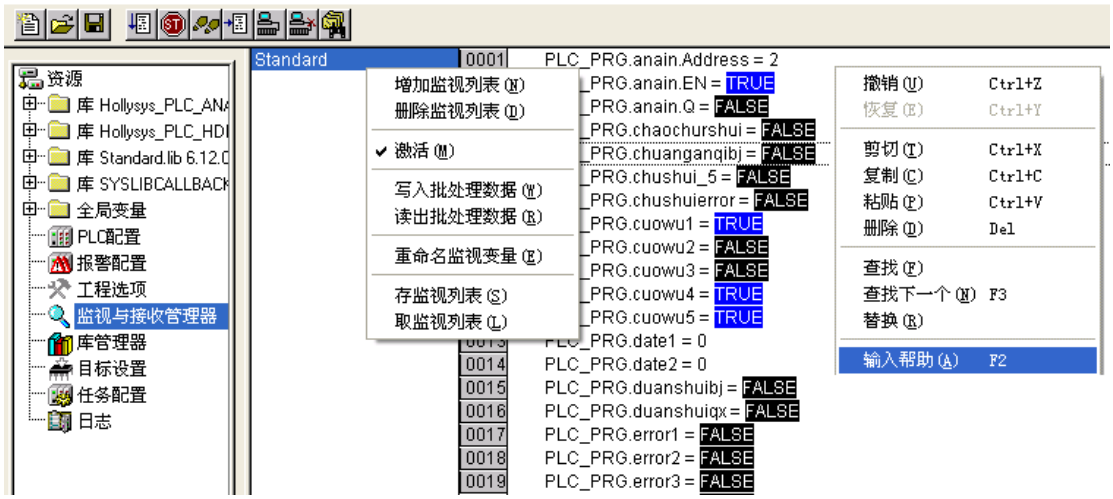


图8-4-15 监视与接收管理器

➢ 创建监视列表

在监视与接收管理器的列表区（左区）点击鼠标右键，从弹出的菜单中选择“增加监视列表”，并为列表输入合适的名称，点击“确认”。也可通过执行“插入”/“增加监视列表”命令创建监视列表。

➢ 修改监视列表名称

执行“重命名监视变量”，可以改变监视列表的名称。

➢ 选择监控变量

选择“输入帮助”，或者利用编辑菜单，列出工程中已定义了的所有变量，用鼠标选择即可将变量加入到监视列表中。

➢ 保存监视列表

执行“存监视列表”，可以保存监视列表，扩展名为“*.wtc”。

➢ 导入监视列表

执行“取监视列表”，可以重新载入已经保存的监视列表。

➢ 激活监视列表

使“激活”之前出现符号“√”，激活监视列表，才可以监控列表中的变量。创建监视列表的各项功能仅在未激活前有效。

➢ 修改变量值

执行“写入批处理数据”，可以将变量的值改写为设置的变量值。

➢ 读取变量值

执行“读出批处理数据”，可以读取变量的当前值。

第9章 IEC 编程基础

PowerPro 软件遵循国际电工技术委员会（International Electrotechnical Commission，缩写为 IEC）的 IEC61131-3 标准。之前在 7.4 章节，讲述了 LD 语言编程的规范，本章主要介绍 FBD、IL、ST、SFC 及 CFC 等其他 IEC 标准编程语言。

9.1 功能块图 FBD

FBD 是功能块图（Function Block Diagram）的简称。FBD 是一种图形化的编程语言，与 LD 的结构类似。FBD 由一系列“节”组成，每“节”由一系列方块组成。每“节”完成一段相对独立的运算，这些运算可以包括逻辑表达式、算术表达式、功能块、连线、输入、输出、跳转和返回等，如图 9-1-1 所示。

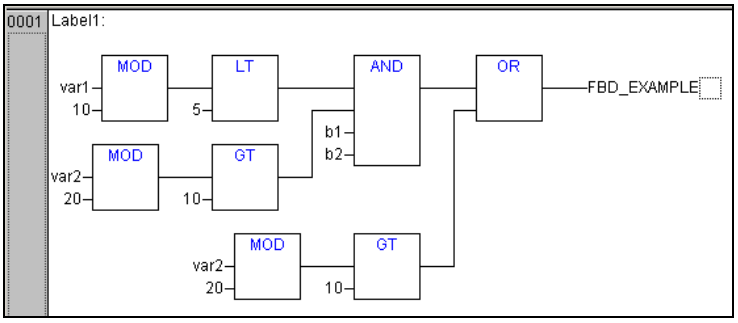
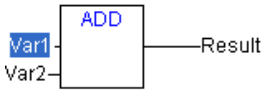


图 9-1-1 功能块图语言

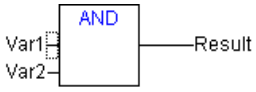
9.1.1 光标位置

通过虚线矩形框确认当前的光标位置。下面介绍可能的光标位置，便于在使用“插入”菜单下的各项命令时识别当前位置。

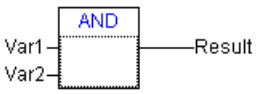
- 文本（光标位置 1）：



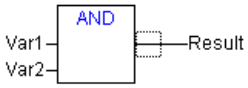
- 输入（光标位置 2）：



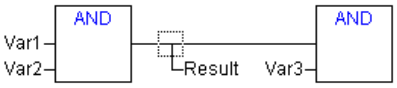
- 操作符、函数或功能块（光标位置 3）：



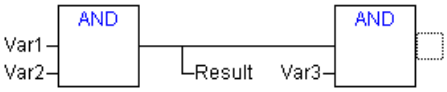
- 输出（光标位置 4，后面紧跟着赋值或跳转）：



- 赋值、跳转或返回的线段交叉点（光标位置 5）：



- 节的末尾输出端（光标位置 6）：



- 赋值前面的交叉线（光标位置 7）：

9.1.2 操作说明

- 添加“输入”

快捷菜单：  。在当前光标位置插入一个函数或功能块的输入端。

对于某些运算符，输入的数量是变化的，有时需要扩展运算符的输入。例如，ADD 可以是两个数相加，也可以是更多的数相加。选中输入（光标位置 2），插入的新输入成为功能块的第一个输入。如果要插入一个位于末端位置的输入，必须选中功能块本身（光标位置 3）。插入的输入缺省值为文本“???”。点击选中文本，改变成所需要的常量或变量。对此可以使用输入辅助。

在需要的时候，增加功能块的输入可以大大简化程序，如图 9-1-2 所示。

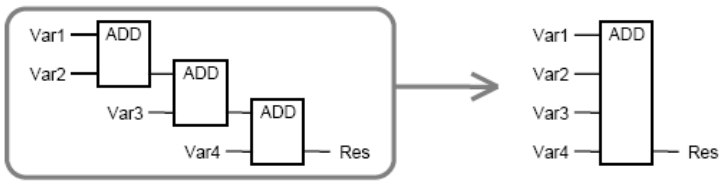


图 9-1-2 扩展输入端

如何快速切换输入端？位于运算符右边的分支与函数的第一个输入端相连，但此时需要改为与第二个输入端相连。可以选中第一个输入执行命令“编辑”/“剪切”。然后选中第二个输入执行命令“编辑”/“粘贴”。这样，此分支连接到第二个输入端。

- 添加“输出”

快捷菜单：  。在当前光标位置插入一个功能块的输出端。

对于某些功能块，输出的数量可能是变化的。此命令可以扩展功能块的输出。

➢ 插入“运算符”

快捷菜单：  。在当前位置插入一个运算符。

如果输入被选中（光标位置 2），运算符被插入到输入的前面。此运算符的第一个输入连接到选中输入的左边分支，新运算符的输出连接到选中的输入。

如果输出被选中（光标位置 4），运算符被插入到输出的后面。此运算符的第一个输入连接到选中的输出，新运算符的输出连接到原来连接的分支上。

如果运算符、函数或功能块被选中（光标位置 3），旧的元素被新的运算符代替。分支的连接与没有被替换之前的情形相同。如果旧元素有比新运算符分支更多，那么多余的分支将被删除。

如果一个跳转或返回被选中，那么运算符会插入到跳转或返回之前。运算符的第一个输入与选中的元素的左边的分支相连，运算符的输出连接到选中元素的右边的分支。

如果“节”的最后一个光标位置被选中（光标位置 6），那么运算符会被插入到最后一个元素之后，运算符的第一个输入连接到选中位置的左边分支。

被插入的运算符缺省总是 AND。选中关键字，可以把 AND 转换成其它的运算符。也可以借助“提示输入”从运算符类型列表中选择所要的运算符。新运算符的输入端将被自动连接到前面分支上。所有没有连接的输入端都标以“???”，可以将其删除或改变成所要的常量或变量。

➢ 添加“赋值”

快捷菜单：  。插入一个赋值符号，将运算符的结果赋值输出。

根据当前选中位置的不同，插入的位置也有所不同。具体地说，赋值符号插入到输入（光标位置 2）的前面，在输出（光标位置 4）的后面，在交叉线（光标位置 5）的前面，在节的末尾（光标位置 6）的后面。

为了给一个已存在的赋值插入一个附加赋值，使用“输出”命令。

➢ 设置“跳转”

快捷菜单：  。设置一个跳转。如果条件为真，则跳转到指定位置。

根据选中的位置，插入到输入的前面（光标位置 2）、输出的后面（光标位置 4）、交叉线的前面（光标位置 5）或节末尾（光标位置 6）的后面。

对于插入的跳转，可以被赋给它的标签代替。

➢ 插入“返回”

快捷菜单：  。插入一个返回。

在当前 POU 被其它 POU 调用，且返回条件为真时，返回到调用它的 POU。

根据选中的位置，插入到选中的输入的前面（光标位置 2）、选中的输出的后面（光标位置 4）、选中的交叉线前面（光标位置 5）或“节”的末尾（光标位置 6）。

➢ “反向”操作

快捷菜单：  。反向输入、输出、跳转或返回指令。

反向的图标是一个连接的小圆。如果光标选中输入（光标位置 2），那么输入被反向。

如果选中输出（光标位置 4），那么输出被反向。如果选中跳转或返回，那么跳转或返回的输入端被反向。再次执行反向命令，则取消反向。

- “置位/复位”操作

快捷菜单：  。定义输出为置位输出或复位输出。

置位输出显示为 S，复位输出显示为 R，如图 9-1-3 所示。如果输出 TRUE 值，则置位端设为 TRUE 并一直保持此值。如果输出 FALSE，则复位端设为 FALSE 并一直保持此值。此命令多次交替执行，输出在置位、复位和正常值之间交替。

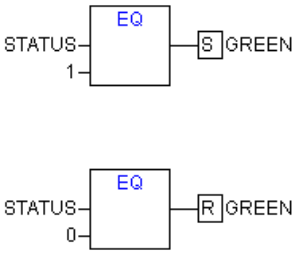


图 9-1-3 置位与复位

FBD 语言的简单应用示例如图 9-1-4 所示。本程序可以产生“1s 断 2s 通”的脉冲信号。

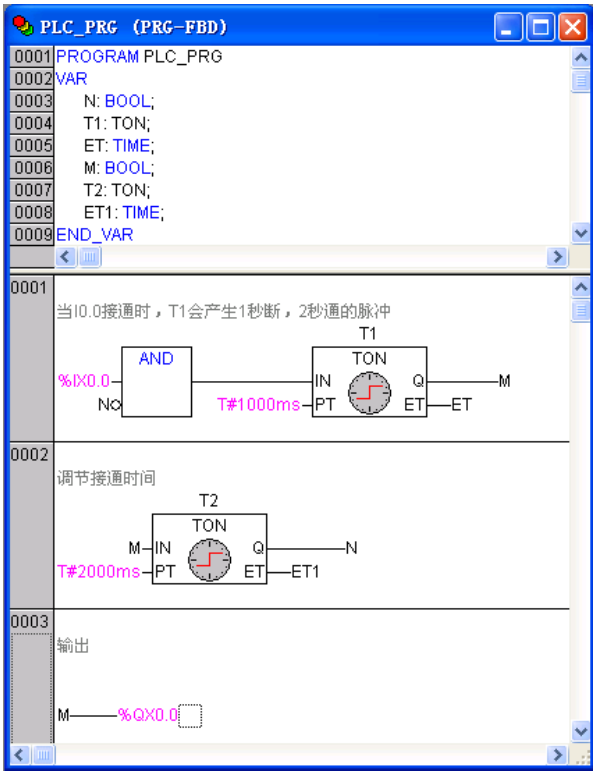


图 9-1-4 FBD 语言示例

9.2 指令列表 IL

指令列表（IL）是一种汇编语言风格的编程语言，程序不易阅读，但执行速度最快。IL

语言包含一系列的指令，每条指令占据一行，包含一个运算符和一个或多个用逗号隔开的操作数。操作数之间用逗号分隔。每行开始可以有标签，标签后要有冒号。每行结束可以有注释，注释用“（*　*）”括起来。每行指令之间可以插入空行。IL 编辑器是一种文本编辑器，具有常见 Windows 文本编辑器的功能，点击菜单栏或鼠标右键可以进行编辑。

9.2.1 操作说明

打开“插入”菜单，选择所要插入的内容，如图 9-2-1 所示。

图 9-2-1 “插入”菜单

- “插入”/“操作符”

在编程区任何位置点击此命令，从弹出窗口中选择所需的操作符，例如“ABS”。

- “插入”/“操作数”

在编程区任何位置点击此命令，从弹出窗口中选择所需的操作数，点击“确认”。

- “插入”/“函数”

在编程区任何位置点击此命令，从弹出窗口中选择所需的函数，点击“确认”。

- “插入”/“功能块”

在编程区任何位置点击此命令，从弹出窗口中选择所需的功能块，点击“确认”。

9.2.2 程序举例

✧ 举例

下面是一个用 IL 语言实现的简单运算程序。

声明：

```
VAR
    A：REAL；
    B：REAL；
    C：REAL；
END_VAR
```

程序：

```
LD      10    （*将数字 10 赋予当前值*）
ADD     A     （*当前值与变量 A 加运算后结果存入当前值*）
GE      B     （*当前值与变量 B 进行大于等于比较*）
JMPC Next1    （*上一表达式结果为真，则跳转至标志 Next1 处*）
```

```
LD      A    （*将变量 A 的值赋予当前值*）
ADD     B    （*当前值与变量 B 加运算后结果存入当前值*）
ST      C    （*将当前值赋予变量 C*）
JMP Next2    （*无条件跳转至标志 Next2 处*）
Next1:       （*标志*）
LD      A    （*将变量 A 的值赋予当前值*）
SUB     B    （*当前值与变量 B 减运算后结果存入当前值*）
ST      C    （*将当前值赋予变量 C*）
Next2:       （*标志*）
```

IL 语言支持两种修饰符 C 和 N。C 表示条件执行，只有当前一个表达式的值为真（TRUE）时，指令才被执行。N 与 JMP、CAL 和 RET 连用时表示条件非执行，只有当前一个表达式的值为假（FALSE）时，指令才被执行；其它情况下，表示操作数取负。表 5-4-1 列出了 IL 的所有运算符以及它们的修饰符和相应意义。

表9-2-1 IL语言运算符列表

运算符	修饰符	意　　义
LD	N	将操作数赋予当前值
ST	N	将当前值赋予操作数
S		如果当前结果是 TRUE，把布尔型操作数置为 TRUE
R		如果当前结果是 TRUE，把布尔型操作数置为 FALSE
AND	N、(	位逻辑运算符 AND
OR	N、(	位逻辑运算符 OR
XOR	N、(	位逻辑运算符 XOR
ADD	(	加
SUB	(	减
MUL	(	乘
DIV	(	除
GT	(	>大于判断
GE	(	>=大于等于判断
EQ	(	=等于判断
NE	(	<>不等于判断
LE	(	<=小于等于判断
LT	(	<小于判断
JMP	C、N	转移到标签
CAL	C、N	调用其它 POU
RET	C、N	退出 POU 并且返回到调用处
)		计算延迟操作

✧ 举例

以下是利用修饰符进行 IL 编程的例子：

```
LD      TRUE    (*将 TRUE 赋予当前值*)
```

```
ANDN    BOOL1    (*BOOL1 变量取反后与当前值进行与运算*)
JMPC    mark     (*如果结果是真，那么跳转到标签“mark”处*)
LDN     BOOL2    (*将 BOOL2 取反后赋予当前值*)
ST      ERG      (*将当前值存入 ERG*)
Mark:
LD      BOOL2    (*将 BOOL2 赋予当前值  *)
ST      ERG      (*将当前值存入 ERG*)
```

如果在运算符之后插入括号，那么括号里的值可以看成是一个操作数。例如：

```
LD    2
MUL   2
ADD   3
ST    ERG
```

运行后 ERG 的值是 7。但是，如果加入一对圆括号：

```
LD    2
MUL   (2
ADD   3
)
ST    ERG
```

运行后 ERG 的值是 10，MUL 运算符只有到“）”才执行，等同于执行 MUL 5。

✧ 举例

IL 语言简单应用示例如图 9-2-2 所示。本程序产生“1s 断 2s 通”的脉冲信号

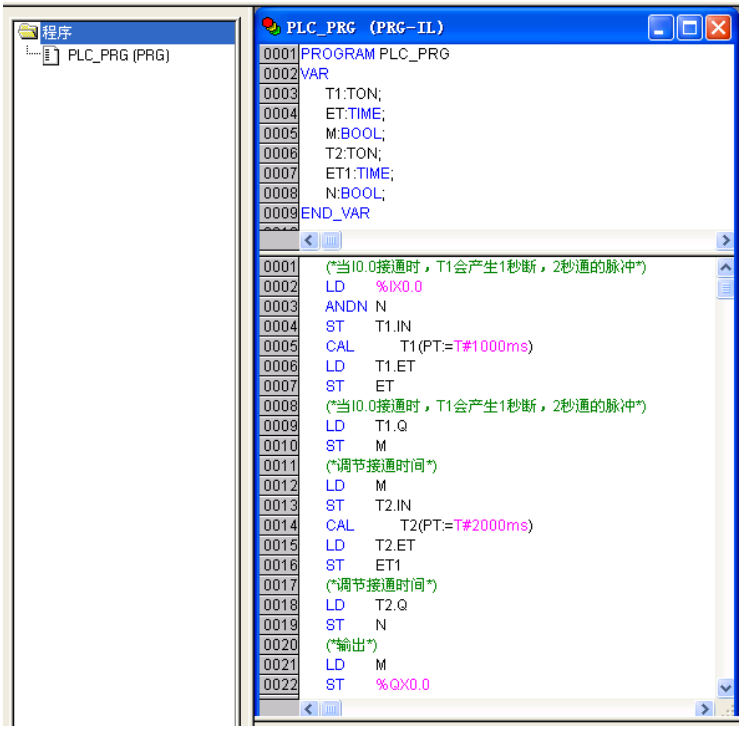


图 9-2-2 IL 语言示例

9.3 结构化文本 ST

结构化文本（ST）可以执行选择语句（IF…THEN…ELSE）和循环语句（WHILE…DO），类似于 PASCAL 和 BASIC 等高级语言。

9.3.1 ST 表达式

ST 语言中的表达式由运算符和操作数组成。操作数可以是常量、变量、函数调用或另一个表达式。表达式的计算通过执行具有不同优先级的运算符完成。有最高优先级的运算符先被执行，然后依次执行下一个优先级的运算符，直到所有的运算符被处理完。有相同优先级的运算符按从左到右的顺序执行。ST 语言的运算符如表 9-3-1 所示。

表9-3-1 ST语言运算符

运算	符号	优先级
放入圆括号	(表达式)	最高优先级
函数调用	函数名（参数列表）	
求幂	EXPT	
求负	-	
求补	NOT	
乘积	*	
除	/	
取模	MOD	
加	+	
减	-	
比较	<、>、<=、>=	
相等	=	
不等	<>	
逻辑与	AND	
逻辑异或	XOR	
逻辑或	OR	最弱优先级

9.3.2 ST 指令

ST 语言的指令如表 9-3-2 所示。

表9-3-2 ST语言指令

指令类型	例子
赋值	A:=B; CV := CV + 1; C:=SIN(X);
调用功能块并且赋初值	TP(IN:= %IX0.5, PT:=t#30);
使用功能块输出	A:=TP.Q;
返回	RETURN;
IF	D:=B*B;

	IF D<0.0 THEN
	C:=A;
	ELSE IF D=0.0 THEN
	C:=B;
	ELSE
	C:=D;
	END_IF;

CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;

FOR循环	J:=101;
	FOR I:=1 TO 100 BY 2 DO
	IF ARR[I] = 70 THEN
	J:=I;
	EXIT;
	END_IF;
	END_FOR;

WHILE循环	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE
REPEAT循环	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
退出程序	EXIT;
空指令	;

➢ 赋值

执行赋值操作时，等号左边是操作数（变量或地址），右边是被赋予的表达式的值。例如：Var1 := Var2 * 10;

➢ 调用功能块

通过写入功能块实例的名字，并且在随后的圆括号中赋给参数值来调用功能块。

✧ 举例

变量声明：

TPInst:TP;

VarBOOL1: BOOL;

VarBOOL2: BOOL;

程序：

TPInst(IN:= VarBOOL1,PT:= T#5s);（*参数 IN 和 PT 设定时钟脉冲的触发信号和高电平的长度*）

VarBOOL2:=TPInst.Q;（*输出脉冲值 Q 赋给变量 VarBOOL2*）

- 返回指令

返回指令可以根据条件退出 POU。

- IF 指令

使用 IF 指令可以检查条件，根据条件执行相应的指令。

语法：

IF <逻辑表达式> THEN
<IF 指令>
{ELSIF <逻辑表达式 1> THEN
<ELSE IF 指令 1>
ELSIF <逻辑表达式 n> THEN
<ELSE IF 指令 n>
ELSE
<ELSE 指令>}
END_IF;

其中{}的部分可选。

如果<逻辑表达式>返回 TRUE，那么只有<IF 指令>被执行，其它的指令不被执行。同样，从<逻辑表达式 1>开始，相继执行逻辑表达式，直到其中一个表达式返回 TRUE 为止，返回 TRUE 的逻辑表达式对应的指令被执行。

如果没有逻辑表达式生成 TRUE，那么只有<ELSE 指令>被执行。

举例

```
IF temp<17
THEN heating_on := TRUE;
ELSE heating_on := FALSE;
END_IF;
```

这里，如果温度低于 17 度，打开加热器，反之则保持关闭状态。

CASE 指令

使用 CASE 指令，可以在结构中用一个相同的条件变量表示几个条件指令。

语法：

```
CASE <Var1> OF
<Value1>: <指令 1>
<Value2>: <指令 2>
<Value3, Value4, Value5>: <指令 3>
<Value6 .. Value10>: <指令 4>
...
<Value n>: <指令 n>
ELSE <ELSE 指令>
```

END_CASE;

CASE 指令根据下面的模型来执行：
如果变量<Var1>有值<值 i>，那么<指令 i>被执行。
如果变量<Var1>没有任何指定的值，那么<ELSE 指令>被执行。
如果变量的几个值都需要执行相同的指令，那么可以把几个值相继写在一起，并且用逗号分开。这样，就会有相同的执行指令。
如果对于变量的一个范围需要执行相同的指令，可以写入初值和终值，中间用两个点分开。这样，条件就会有相同的执行。
举例
CASE INT1 OF
1, 5: BOOL1 := TRUE;
BOOL3 := FALSE;
2: BOOL2 := FALSE;
BOOL3 := TRUE;
10..20: BOOL1 := TRUE;
BOOL3:= TRUE;
ELSE
BOOL1 := NOT BOOL1;
BOOL2 := BOOL1 OR BOOL2;
END_CASE;

FOR 循环
使用 FOR 循环，可以编写循环过程。
语法：
INT_Var :INT;
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE>
{BY <Step Size>} DO
<Instructions>
END_FOR;
其中{}的部分可选。
只要计数<INT_Var>不大于<END_VAULE>，指令就会被执行。指令执行之前，首先检查这个条件，如果<INIT_VALUE>大于<END_VALUE>，指令就永远不会被执行。
当指令被执行时，<INT_Var>总是增加步长<Step Size>。步长可以是任意的整数值。如果不写步长，缺省值是 1。当<INT_Var>大于<END_VALUE>时，循环结束。
举例
FOR Counter:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
假定 Var1 的缺省值是 1，那么循环结束后，Var1 的值为 32。
注意，<END_VALUE>一定不能等于计数变量<INT_VAR>的极限值。如果计数变量的类型是 SINT（范围从-128-127），<END_VALUE>是 127，则会进入死循环。

WHILE 循环

WHILE 循环用起来很象 FOR 循环，不同的是，结束条件可以是任意的逻辑表达式。指

定一个条件，当条件满足的时候，循环被执行。

语法：

WHILE <逻辑表达式>

<指令>

END_WHILE;

只要<逻辑表达式>的值返回 TRUE，<指令>就会被重复执行。如果在第一次计算时，<逻辑表达式>的值已经是 FALSE，那么指令永远不会被执行。如果<逻辑表达式>的值永远不会是 FALSE，那么<指令>被无休止的执行，产生一个相对时间延迟。

举例

WHILE counter<>0 DO

Var1 := Var1*2;

Counter := Counter-1;

END_WHILE

在一定意义上，WHILE 和 REPEAT 循环比 FOR 循环功能更强大。因为不需要在执行循环之前计算循环次数。因此，在有些情况下，用这两种循环就可以了。然而，如果清楚知道循环次数，那么 FOR 循环更好。

REPEAT 循环

REPEAT 循环不同于 WHILE 循环，因为只有在指令执行以后才检查中断条件。无论结束条件怎样，循环至少执行一次。

语法：

REPEAT

<指令>

UNTIL <逻辑表达式>

END_REPEAT;

直到<逻辑表达式>的值返回 TRUE，<指令>才停止执行。如果在第一次计算时，<逻辑表达式>产生 TRUE，那么<指令>只被执行一次，如果<逻辑表达式>不会产生 TRUE，那么<指令>将无休止的循环，导致相对时延。

举例

REPEAT

Var1 := Var1*2;

Counter := Counter-1;

UNTIL

Counter=0

END_REPEAT;

EXIT 指令

如果 EXIT 指令出现在 FOR、WHILE、REPEAT 循环中，那么不管中断条件如何，EXIT 出现时循环终止。

举例

下面是一个用 ST 语言实现简单运算的小程序：
变量声明：

```
PROGRAM    PLC
VAR
A:BOOL;
B:BOOL;
C:INT;
END_VAR
```

程序：

```
IF A=TRUE THEN
C:=20;
ELSE IF B=TRUE THEN
C:=30;
ELSE C:=50;
END_IF
```

简洁、易读的语言 ST
结构化文本这个名字就已经表明它是用于结构化编程的。也就是说，ST 语言为特定的经常使用的结构化编程提供了预先确定的结构。

举例
比较下面两段分别用 IL 和 ST 语言实现 2 的乘幂的循环的程序代码。
用 IL 语言：

```
Loop:
LD Counter
NE 0
NOT
JMPC END_LOOP
LD Var1
MUL 2
ST Var1
LD Counter
SUB 1
ST Counter
JMP Loop
End_LOOP:
LD Var1
ST ERG
```

用 ST 语言：

```
WHILE counter<>0 DO
Var1:=Var1*2;
Counter:=counter-1;
```

```
END_WHILE
Erg:=Var1;
```

可以看出，用 ST 语言实现的循环不仅简洁，而且易于阅读。

9.4 顺序功能图 SFC

SFC 是顺序功能图（Sequential Function Chart）的简称，是一种图形化的编程语言，用来描述程序中不同动作的时间顺序。SFC 由一系列的步和转移组成，如图 9-4-1 所示。步定义动作，转移控制顺序。

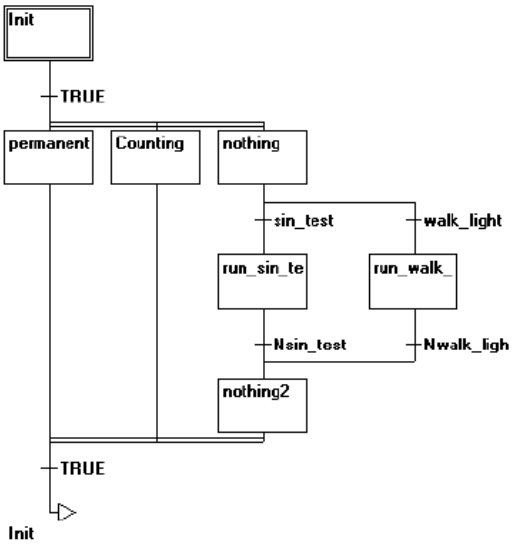


图 9-4-1 顺序功能图语言

9.4.1 基本概念

➢ 单步

SFC 语言包含一系列的步，这些步通过有向的连接彼此联系。步分为简化步和 IEC 步。

简化步包含一个动作和一个显示步动作的标志。如果步包含动作，则在步的右上角出现一个小三角形。

IEC步包含一个或多个动作和逻辑变量。新插入的步是否为IEC步，取决于是否选择了“高级”/“使用 IEC 步（I）”命令。必须添加特殊的 SFC 库 Iecsfc.lib 才能够使用 IEC 步。

➢ 动作

动作是使用其它语言实现的一系列指令，可以是用 IL 或 ST 语言实现的指令语句，也可以是用 LD、FBD 或 SFC 实现的网络。

对于简化步，动作总是和步直接相关。用鼠标双击动作所属的步，进行编辑。

对于 IEC 步，在对象组织器中选中 SFC 程序，点击右键，选择“添加动作”命令创建新动作。可以赋给 IEC 步多个动作，同时这些动作也可以被多个步重复使用。

IEC 步的动作显示在步的右边框中。左边字段包含可能有时间常量的限定符，右边字段包含动作名即逻辑变量名。

IEC 的动作分散在步中。在它们所属的 POU 中，这些动作可以被重复使用。使用“高

级”/“关联动作”命令添加动作，使用“高级”/“清除动作或转移”命令删除已添加的动作。

包含两个动作的 IEC 步举例，如图 9-4-2 所示。

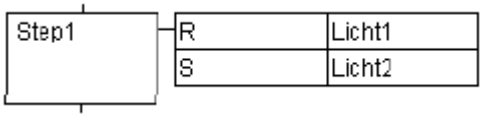


图 9-4-2 顺序功能图语言举例

➢ 入口动作和出口动作

“添加入口动作”和“添加出口动作”命令可以在步中加入入口动作和出口动作。

入口动作只有当步在活动状态时立即执行一次。出口动作只在步不活动之前执行一次。有入口动作的步在左下角有“E”标志进入，有出口动作的步右下角有“X”标志退出。可以用任意语言实现入口和出口动作，鼠标双击步的相应角，即可编辑入口或者出口动作。

入口和出口动作步的举例，如图 9-4-3 所示。

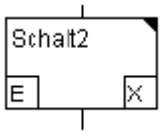


图 9-4-3 入口和出口动作步的举例

➢ 转换/转换条件

步之间的切换就是转换。只有当步的转换条件为真时，步的转换才进行。即前步的动作执行完后，如果有出口动作则执行一次出口动作，后步如果有入口动作则执行一次后步的入口动作，然后按照控制周期执行该活动步的所有动作。

转换条件可以是逻辑变量、逻辑地址、逻辑常量或者是由其它语言编程实现的逻辑。

➢ 活动步

调用 POU 之后，初始化步（双边的步）的动作首先执行。动作被执行的步称为活动步。

每次循环执行活动步的动作。在线模式下，活动步以蓝色显示。为了更容易地跟随过程，在线模式下，所有的活动动作象活动步一样以蓝色显示。每次循环以后做一次检查，查看哪个动作是活动的。在一个控制周期中，执行活动步的所有动作。之后，如果下步的转换条件是 TRUE，下步将变成活动步，在下个周期执行。

➢ 限定符

把 IEC 步和动作关联，需要使用限定符。IEC 步的使用如图 9-4-4 所示，限定符对应的含义如表 9-4-1 所示。

图 9-4-4 使用 IEC 步

表9-4-1 SFC限定符

限定符	名称	说明	备注

N	非存储	**普通型**：动作随步活动，当步开始激活时，动作开始执行；当步退出时，动作停止执行。	
S	设置存储	**置位型**：当步开始激活时，动作开始执行；当步退出时，动作还将继续执行。一直等到有 R 限定符给它复位之后，动作才将停止执行。在动作没有被 R 停止之前，任何限定符也不能使得动作停止（包括 P 限定符）。	
R	重置	**复位型**：当步开始激活时，动作停止执行。	
L	时限	**限时结束型**：动作在一定时间被激活。最长时间是步活动时间。当步开始激活时，动作开始执行；同时开始计时，当时间到达设定值后，动作将停止执行。如果时间还没有到达设定值时，步的转移条件已经满足，此时，动作也将停止执行。	要加时间常数**（设定值）**
D	延时	**延时开始型**：当步开始激活时，开始计时，当时间到达设定值后，动作将开始执行；当步退出时，动作停止执行。如果时间还没有到达设定值时，步的转移条件已经满足，在这种情况下，动作将不可能被执行。	要加时间常数**（设定值）**
P	脉冲	**脉冲型**：当步开始激活时，动作开始执行，并且动作只被执行一次，之后动作将停止执行。	
SD	存储和延时	**延时绝对开始置位型**：当步开始激活时，开始计时，当时间到达设定值后，动作将开始执行；当步退出时，动作还将执行，一直等到有 R 限定符给它复位之后，动作才将停止执行。在动作没有被 R 停止之前，任何限定符也不能使得动作停止（包括 P 限定符）。如果时间还没有到达设定值时，步的转移条件已经满足，在这种情况下，计时还在进行，一直到时间达到设定值后，动作将被执行，此时不管当前步还是不是原来的步了。	要加时间常数**（设定值）**
DS	延时和存储	**延时开始置位型**：当步开始激活时，开始计时，当时间到达设定值后，动作将开始执行；当步退出时，动作还将执行，一直等到有 R 限定符给它复位之后，动作才将停止执行。在动作没有被 R 停止之前，任何限定符也不能使得动作停止（包括 P 限定符）。如果时间还没有到达设定值时，步的转移条件已经满足，在这种情况下，动作将不可能被执行。	要加时间常数**（设定值）**
SL	存储和时限	**绝对限时结束型**：在一定时间内动作是活动的。当步开始激活时，动作开始执行；同时开始计时，当时间到达设定值后，动作将停止执行。如果时间还没有到达设定值时，步的转移条件已经满足，此时，动作还将继续执行，直到当时间到达设定值后动作才会停止执行。限制符 L、D、SD、DS 和 SL 需要 TIME 常量格式的时间值。	要加时间常数**（设定值）**

➢ SFC 中的隐含变量

在 SFC 中可以隐含声明的变量。每步的标志都存储了步的状态。

对于简化步，步标志为<步名>。对于 IEC 步，步标志为<步名>.x。

当步是活动状态的时候，步标志的值为 TRUE。当步是非活动状态的时候，步标志的值为 FALSE。

对于 IEC 步，可以使用变量<动作名>.x 询问动作是否是活动的。对于 IEC 步，可以使用

隐含变量<步名>.t 用来询问步的活动时间。

- SFC 标志

在 SFC 中，步的活动状态时间取决于它的属性状态时间，有时会设置一些特殊标志。同样，可以设置变量来控制顺序功能图中的程序流。要使用这些标志，必须在全局或局部变量中作为输入或者输出变量声明这些标志。

SFCEnableLimit：BOOL 型变量，变量值是 TRUE 时，步的超时将会注册在 SFCWError 中，忽略其它的超时。

SFCInit：BOOL 型变量，变量值是 TRUE 时，顺序功能图返回到初始化步，同时其它的 SFC 标志重置。只要变量有 TRUE 值，初始化步保持活跃，但不被执行。只有当 SFCInit 被重新设置为 FALSE，功能块才恢复正常执行。

SFCQuitError：BOOL 型的变量，变量值是 TRUE 时，SFC 图停止执行。在变量 SFCError

中，重置一个可能的超时。当变量重新获得 FALSE 时，在活动步中所有以前的时间重置。

SFCPause：BOOL 型变量，变量值是 TRUE 时，SFC 图停止执行。

SFCError：当超时发生在 SFC 图中时，设置这个逻辑变量。如果一个超时发生而变量 SFCError 不重置，则不会注册其它超时。

SFCTrans：BOOL 型变量。当转换执行时，变量值为 TRUE。

SFCErrorStep：STRING 型变量，存储一个导致超时发生的步名。

SFCErrorPOU：STRING 型变量，包含一个发生了超时的块名。

SFCCurrentStep：STRING 型变量，步名存储在活动变量中，不受时间监视。一旦在同时顺序下，步存储在右外侧的分支里。

9.4.2 操作说明

- 选中元素

被选中的元素周围用点线矩形圈住。点击鼠标左键，或用键盘的上下键可以选中元素。按住 SHIFT 键，点击鼠标左键可以选中一组元素。注意，删除步时，必须同时选中此步前转移或后转移，一起删除。

- “插入”/“前步转移”：  ，快捷键<Ctrl>+<T>

执行此命令，在选中元素前面插入步和转移。

- “插入”/“后步转移”：  ，快捷键<Ctrl>+<E>

执行此命令，在选中元素后面插入步和转移。

- “插入”/“右选择分支”：  ，快捷键<Ctrl>+<A>

选中元素必须以转移开始和结束，执行此命令，新插入的分支作为选中元素的右分支，新分支包括一个转移。

- “插入”/“左选择分支”： 

选中元素必须以转移开始和结束，新插入的分支作为选中元素的左分支，新分支包括一个转移。

- “插入”/“右并行分支”：  ，快捷键<Ctrl>+<L>

选中元素必须以步开始和结束，执行此命令，新插入的分支作为选中元素的右分支，新分支包括一个步。如果此并行分支需从其它元素跳转过来，那么此并行分支还要加标签。

- “插入”/“左并行分支”： 

选中元素必须以步开始和结束，执行此命令，新插入的分支作为选中元素的左分支，新分支包括一个步。如果此并行分支需从其它元素跳转过来，那么此并行分支还要加标签。

- “插入”/“跳转”：  ，快捷键<Ctrl>+<U>

选中元素所在的分支必须是选择分支，执行此命令，在此分支末尾插入跳转。新跳转旁边有文本字符串 Step，可以改为要跳转到的步的名字或并行分支的跳转标签。

- “插入”/“转移跳转”： 

选中元素所在的分支必须是并行分支，执行此命令，在此分支末尾插入转移和跳转。新跳转旁边有文本字符串 Step，可以改为要跳转到的步的名字或并行分支的跳转标签。

- “插入”/“添加入口动作”

入口动作只在步被激活时执行一次，入口动作可以选择一种语言编写。带入口动作的步在左下脚有个“E”。

- “插入”/“添加出口动作”

出口动作只在步被解除激活前执行一次，出口动作可以选择一种语言编写。带出口动作的步在右下脚有个“X”。

- “高级”“右侧并行粘贴”

选中元素必须是以步开始和结束，执行此命令，剪贴板的内容作为选中元素的右分支。剪贴板的内容也必须以步开始和结束。

- “高级”/“并行分支增加标签”

选中元素必须是并行分支的前转移，执行此命令，此并行分支的上方出现标签。标签缺省的是“Parallel”加“序列号”，可以编辑，如图 9-4-5 所示，改为“abc”，删除标签名，就可以删除跳转标签。

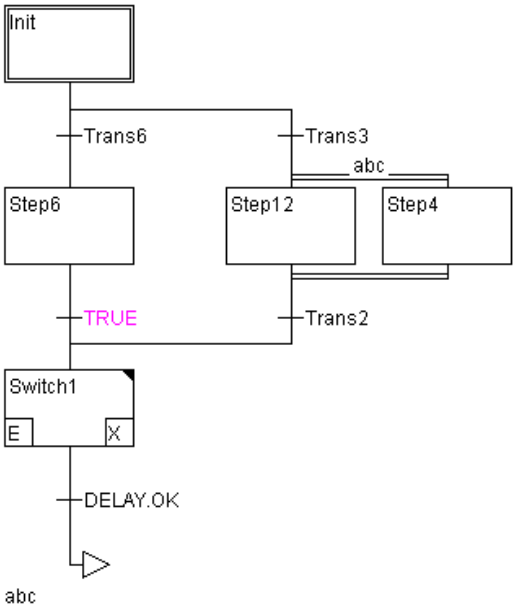


图 9-4-5 并行分支增加标签

➢ “高级”/“后面粘贴”

选中元素，执行此命令，剪贴板的内容粘贴到选中元素的第一个步或转移后面。

注意，普通的粘贴（“编辑”“粘贴”）会粘贴到选中元素的前面。

➢ “高级”/“放大动作/转移”：快捷键<Alt>+<Enter>

执行此命令，弹出编辑器，查看动作或转移条件。如果动作或转移条件是空的，执行此命令，可以在弹出窗口中选择一种语言，进入相应编辑器，编写内容。

注意，编辑器中的转移条件优先于直接在转移标记中编写的条件。如图 9-4-6 所示，如果 i>100，那么转移条件是 FALSE，尽管转移标记中写的是 TRUE。

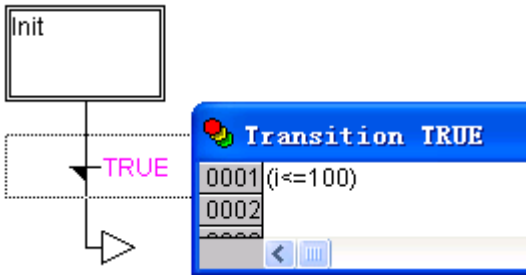


图 9-4-6 转移条件有优先级

➢ “高级”/“清除动作/转移”

执行此命令，清除动作或转移。

➢ 选择分支

在 SFC 中，两个以上的分支可定义为选择分支。每个选择分支必须开始和结束于转换。选择分支包含并行分支和其它选择分支。选择分支开始于水平行（选择开始）结束于水平行（选择结束）或者结束于跳转。

如果选择起始行之前的步是活动的，那么每个选择分支的第一个转换从左到右执行。从左边开始，转换条件是 TRUE 则第一个转换被打开，随后的步被激活。

➢ 并行分支

在 SFC 中，两个以上的分支可以定义成并行分支。每个并行分支开始和结束于一个步。并行分支开始于一个双行（并行开始）结束于一个双行（并行结束）或者是一个跳转。

如果前步的并行开始行是活动的，并且这一步的转换条件是 TRUE，那么所有并行分支的第一步都成为活动的。这些分支彼此是平行处理的。当前面的步都是活动的并且这一步之前的转换条件是 TRUE，则并行结束行之后的步被激活。

➢ 跳转

跳转实际上是连接到有跳转符号的步的操作。当不允许产生彼此交叉的连接时就需要跳转。

图 9-4-7 所示是一个 SFC 语言典型功能的举例。

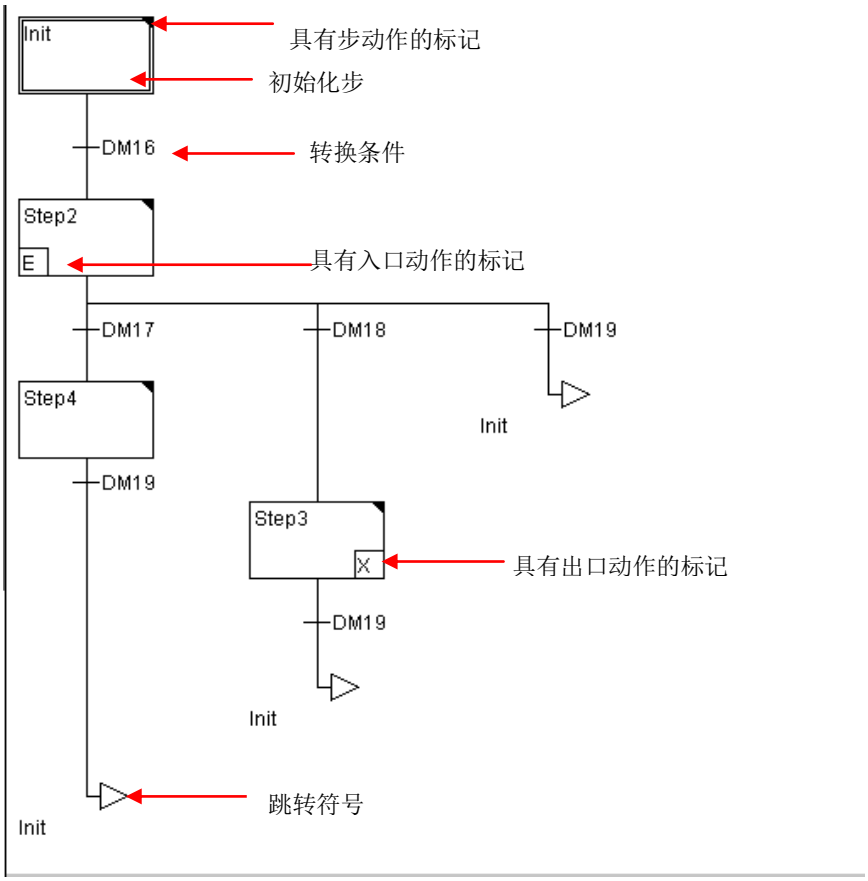


图 9-4-7 SFC 功能图解说

9.5 连续功能图 CFC

9.5.1 CFC 编辑器

CFC 是连续功能块图（Continuous Function Chart）的简称。CFC 是一种图形化的编程语言。CFC 基于 FBD 语言，但没有网络限制，摆放元素更加灵活。

CFC 编辑器是一种图形编辑器，如图 9-5-1 所示，通过菜单栏或鼠标右键可以进行编辑。元素可以摆放在编程区任意位置。用鼠标拖拽在元素之间连线，当元素移动位置时，编辑器会自动调整连线长度。当元素之间空间不够时，连线会变为红色，一旦空间够用，红线会变为普通连线。

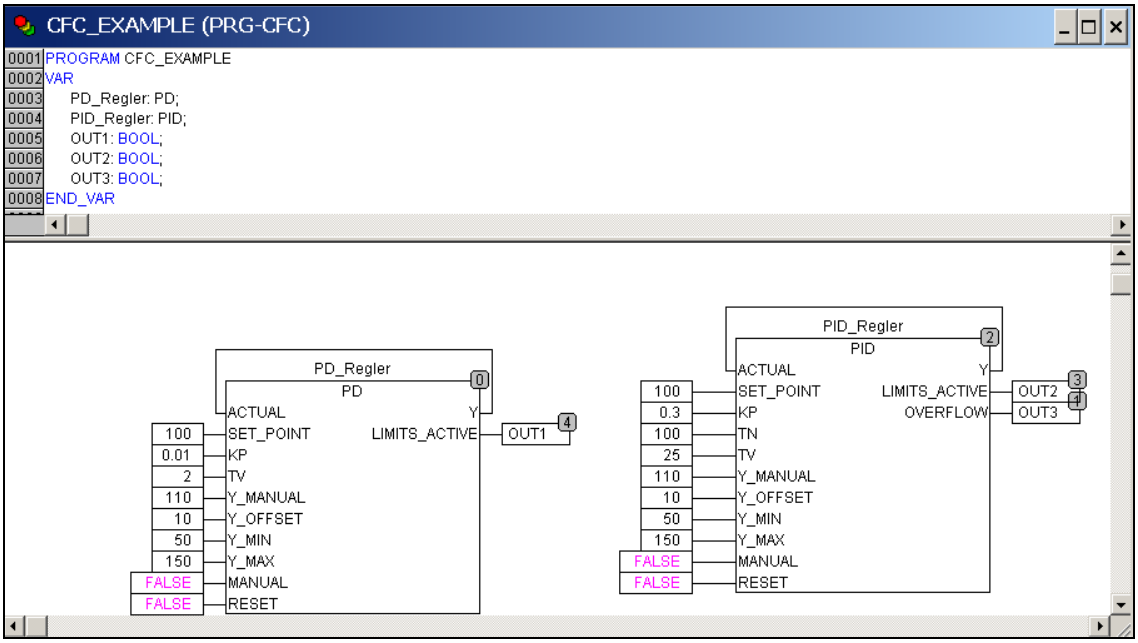
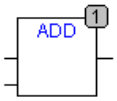
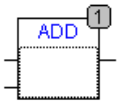
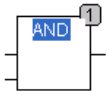
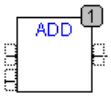
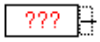
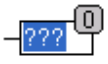
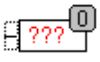
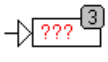
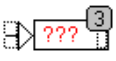
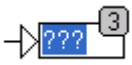
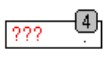
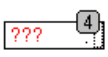
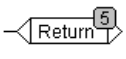
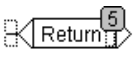
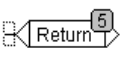


图 9-5-1 CFC 编辑器

9.5.2 操作说明

CFC 的元素包括块、输入、输出、跳转、标记、返回和注释等。其中块分为操作符、函数、功能块和程序四种形式。

➢ 光标位置

名称	图形元素	选中元素	文本域	输入引脚和输出引脚
光标位置	编程工具元素	光标处于各元素的的中继线	光标处于蓝色区域及注释的光标所在位置上	光标处于元素的输入引脚和输出引脚上
块				
输入				
输出				
跳转				
标记				——
返回			——	
注释				——

➢ 选中元素

在元素中继线处点击鼠标左键，可以选中元素。

如果想同时选几个元素，按住<Shift>键并选中单个元素。也可以用鼠标左键在编辑器中画矩形区域选中其中几个元素。”高级”/“全选”选中所有元素。

➢ 移动元素

当光标在位置 a 时，或按住<Shift>键同时选中几个元素时，按住鼠标左键可以在编辑器中移动元素，到合适的位置后释放左键。如果释放位置处已有其它元素或超出编辑区，被移动元素会跳回原位置，移动失败。

➢ 连线

一个元素的输入引脚只能连一个输出引脚（本元素的输出引脚或其它元素的输出引脚），而一个元素的输出引脚可以连几个输入引脚（本元素的输入引脚或其它元素的输入引脚）。如图 9-5-2 所示，有三种方式在 E1（a）和 E2（ADD）之间连线。在连线时，编辑器会检查双方的数据类型是否匹配。如果不匹配，光标会变为“禁止”样式，连线失败。

把鼠标放在 E1 的输出引脚上，按下左键，拖拽到 E2 的输入引脚上，释放左键。

把鼠标放在 E2 的输入引脚上，按下左键，拖拽到 E1 的输出引脚上，释放左键。

按下左键不放，移动 E1 或 E2，使它们的输入引脚和输出引脚接触。

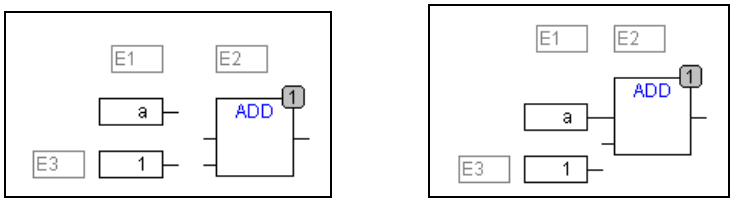


图 9-5-2 连线

➢ 删除连线

如图 9-5-2 所示，有三种方式删除 E1（a）和 E2（ADD）之间的连线。选中 E1 的输出引脚，按下<Delete>键或“编辑”“删除”，如果 E1 的输出引脚有几条线，则会同时删除。选中 E2 的输入引脚，按下<Delete>键或“编辑”“删除”。选中 E2 的输入引脚，用鼠标左键拖拽连线到编辑器空白位置，释放。

➢ “插入”/“块”：  ，快捷键<Ctrl>+<B>

执行此命令，可以插入操作符、函数、功能块和程序。新插入的块随鼠标移动，移动到合适的位置点击鼠标左键，插入成功。起初插入的是 AND，可以把 AND 改为其他名称。或者选中 AND，按功能键 F2，从帮助窗口中选择名称。

➢ “插入”/“输入”：  ，快捷键<Ctrl>+<I>

执行此命令，可以插入输入。新插入的输入随鼠标移动，移动到合适的位置点击鼠标左键，插入成功。文本域“???”要输入变量或常量。也可按功能键 F2，从帮助窗口中选择。

➢ “插入”/“输出”： 

执行此命令，可以插入输出。新插入的输出随鼠标移动，移动到合适的位置点击鼠标左键，插入成功。文本域“???”要输入变量。也可以按功能键 F2，从帮助窗口中选择。

➢ “插入”/“跳转”：  ，快捷键<Ctrl>+<J>

执行此命令，可以插入跳转。新插入的跳转随鼠标移动，移动到合适的位置点击鼠标左键，插入成功。文本域“???”要输入要跳转到的跳转标记。

➢ “插入”/“标记”：  ，快捷键<Ctrl>+<L>

执行此命令，可以插入标记。新插入的标记随鼠标移动，移动到合适的位置点击鼠标左键，插入成功。文本域“???”要输入跳转标记。在线模式时，编辑器自动在 POU 末尾自动插入标记“RETURN”。

➢ “插入”/“返回”：  ，快捷键<Ctrl>+<R>

执行此命令，可以插入返回。新插入的返回随鼠标移动，移动到合适的位置点击鼠标左键，插入成功。注意，此处的“返回”和在线模式时的 POU 末尾的标记“RETURN”不同，标记“RETURN”是为了 POU 结束本周期执行之前，自动执行下周期。

➢ “插入”/“注释”：  ，快捷键<Ctrl>+<K>

执行此命令，可以插入注释，用<Ctrl>+<Enter>换行。新插入的注释随鼠标移动，移动到合适的位置点击鼠标左键，插入成功。

➢ “高级”/“顺序”/“显示”

执行此命令，切换执行顺序的显示与否。缺省的是显示（命令前有“√”）。

➢ “高级”/“顺序”/“拓扑”

执行此命令，执行顺序根据拓扑结构排列，也就是所有元素从上到下，从左到右排列。注意，执行顺序显示在元素的右上脚。

➢ “高级”/“顺序”/“根据数据流排列”

执行此命令，执行顺序根据数据流排列，而不是根据元素所在的位置排列。

CFC 应用举例如图 9-5-3 所示。

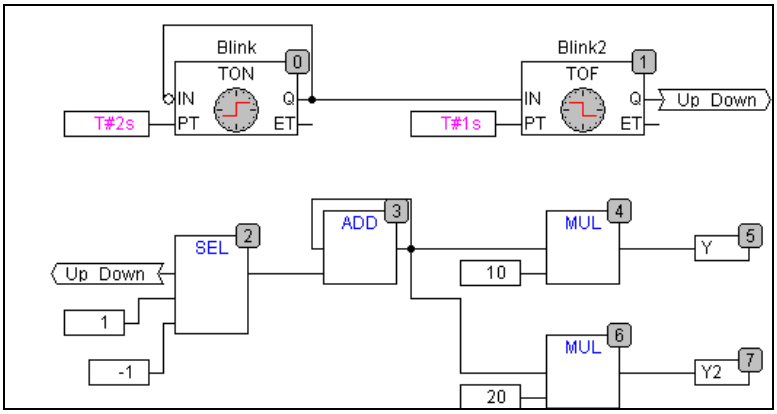


图 9-5-3 连续功能图语言示例

第10章 特殊功能

10.1 Modbus 通讯

LM 系列 PLC 可以通过 Modbus 协议与触摸屏或组态软件等第三方设备进行通讯。默认状态下，LM 系列 PLC 作为 Modbus RTU 从站。

LM 系列 PLC 的 RS232 端口和 RS485 端口均支持 Modbus RTU 从站协议。

10.1.1 Modbus 概述

Modbus 协议是主从站通讯协议，用异步串行口完成通讯，物理层采用 RS485 或 RS232。传输速率可以达到 115kbps，理论上可接（寻址）一台主站和至多 247 台从站。受线路和设备的限制，最多可接一台主站和 32 台从站。

Modbus 协议的某些特性是固定的，如帧格式、帧顺序、通讯错误和异常情况的处理以及所执行的功能等，都不能随便改动。其他特性属于用户可选的，如传输介质、波特率、字符奇偶校验、停止位的个数等等，传输模式为 RTU。用户所选择的参数对于各个站必须一致，在系统运行时不能改变。

10.1.2 Modbus 通讯功能

LM 系列 PLC 所支持的 Modbus RTU 通讯功能码如表 10-1-1 所示。

表 10-1-1 Modbus 功能码

功能码	名称	作用（对主站而言）
01	读取开出状态	取得一组开关量输出的当前状态
02	读取开入状态	取得一组开关量输入的当前状态
03	读取模出状态	取得一组模拟量输出的当前状态
04	读取模入状态	取得一组模拟量输入的当前状态
05	强制单路开出	强制设定某个开关量输出的值
06	强制单路模出	强制设定某个模拟量输出的值
15	强制多路开出	强制设定从站几个开关量输出的值
16	强制多路模出	强制设定从站几个模拟量输出的值

Modbus RTU 协议能访问的数据区包括：

输入区（I）、输出区（Q）、中间区（M）

这三个数据区，均可通过 BOOL 型或 WORD 型数据访问。这些数据区与 Modbus 协议地址映射关系，如表 10-1-2 所示：

表 10-1-2　LM 系列 PLC 数据区 Modbus 地址映射关系

数据区		类型	地址范围	Modbus 地址	映射公式	Modbus 数据类型
I	%IX	BOOL	%IX0.0~%IX511.7	0~4095	IXm.n：m*8+n	1x
区	%IW	WORD	%IW0~%IW510	0~255	IWm：m/2	3x
Q	%QX	BOOL	%QX0.0~%QX511.7	0~4095	QXm.n：m*8+n	0x
区	%QW	WORD	%QW0~%QW510	0~255	QWm：m/2	4x
M	%MX	BOOL	%MX0.0~%MX7816.7	3000~65535	MXm.n：m*8+n+3000	0x
区	%MW	WORD	%MW0~%MW8190	3000~7095	MWm：m/2+3000	4x

部分 HMI 数据地址从 1 开始，若使用 Modbus RTU 协议与 PLC 通讯，在填入数据地址时，需要在映射地址公式基础上加 1。如：%MX100.0，其地址应为 100*8+0+3000+1=3801。此类 HMI 如：Eview、MCGS、Weinview 等触摸屏和组态王、三维力控等组态软件。但有的 HMI 数据地址则无需在映射地址公式基础上加 1，如：Hitech 等。

注意：

1、 M 区大小为 8K，其地址按 BOOL 类型访问，从%MX0.0~%MX8191.7，但是根据 Modbus 协议，其地址最大范围为 65535，所以通过 Modbus 协议访问 M 区的开关量，最大只能访问到%MX7816.7。

2、 表中所列的 I 区和 Q 区范围为最大的可能范围，具体情况需根据实际配置进行计算。对于不存在的 I 区和 Q 区数据无法进行通讯。

10.1.3　Modbus 通讯举例

LM 系列 PLC 的 RS232 和 RS485 均可独立配置为 Modbus RTU 从站协议。在通讯时，PLC 端需要设置从站地址及通讯参数。LM 系列 PLC 默认的从站地址为 51，通讯参数为 38400,n ,8, 1。

设置 PLC 从站地址，需使用 SET_LOCAL_ADDRESS 指令设置 RS232 口的从站地址，对 LM3108/9-D02 及后续版本使用 SET_LOCAL_ADDRESS_RS485 指令设置 RS485 口的从站地址。设置通讯参数，RS232 串口使用 Reset_COMM_PRMT 指令，RS485 串口使用 Reset_COMM2_PRMT 指令。具体指令使用方法，请参见《指令手册》。

图 10-1-1 为设置从站地址为 5，RS232 串口波特率为 9600，数据位为 8，停止位为 1，校验方式为无校验的设置方法。

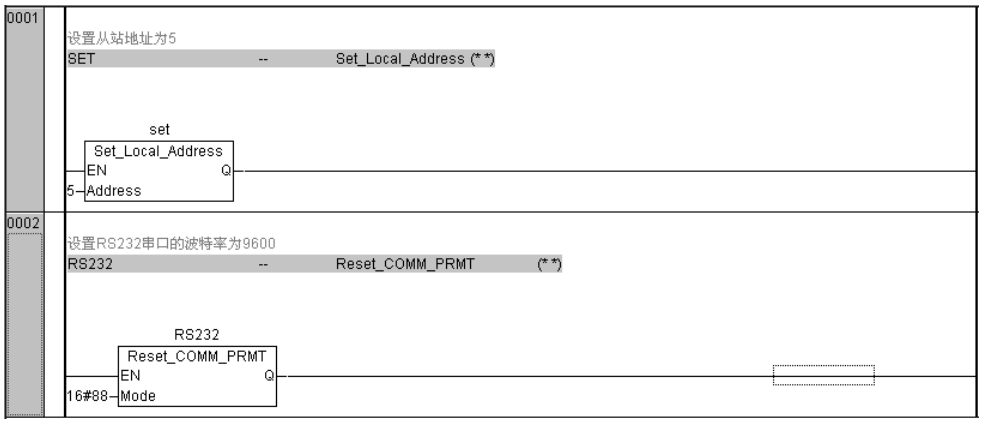


图 10-1-1

10.2 中断

10.2.1 中断概述

中断指当 PLC 接到外界硬件或内部发来的特殊信号时，马上停止原来的工作，转去处理这一事件，在处理完了以后，PLC 又回到原来的工作继续工作的过程。产生中断的信号称为中断源。

LM 系列 PLC 可处理多种中断源产生的中断信号。各种中断源是有优先级区别的，当两个中断同时到来的时候，优先级高的中断先被执行。

表 10-2-1 是 LM 系列 PLC 能处理的中断源。

表 10-2-1 中断事件

名称	描述
Start	当程序开始时调用
Stop	当程序停止时调用
Debug_loop	当调试循环运行时被调用
Taskcode not called	不调用任务代码
Fast External 0 interupt	快速外部中断 0 产生时调用
Fast External 1 interupt	快速外部中断 1 产生时调用
Fast External 2 interupt	快速外部中断 2 产生时调用
Fast External 3 interupt	快速外部中断 3 产生时调用
HD_TC7 interupt	定时器 T7 中断产生时调用
HD_TC2 interupt	高速计数器 T2 产生中断时调用
HD_TC3 interupt	高速计数器 T3 产生中断时调用
HD_TC4 interupt	高速计数器 T4 产生中断时调用
HD_RTC_ALM0 interupt	实时时钟报警 0 产生中断时调用
PTO_0 Finished interupt	QX1.1 高速脉冲输出结束中断时调用
PTO_1 Finished interupt	QX0.3 高速脉冲输出结束中断时调用

10.2.2 中断使用举例

使用中断，需要完成两项工作。其一，需要编写一个中断服务程序。所谓中断服务程序是指在产生中断后，PLC 所要执行的程序。编写中断服务程序的过程就是编写子程序的过程，请参见 7.4.7 章节。其二，完成中断服务程序后，需要进行任务配置，配置所产生的中断及相应的中断服务程序。在“资源”选项卡中，双击“任务配置”，右侧弹出任务配置窗口，点击窗口中“任务配置”下的“System events”，右边窗口显示所有可用的系统事件，如图 10-2-1 所示。在右边窗口中选择需要使用的系统事件，即在系统事件的前面方框里打勾，然后在该系统事件后面的“called POU”处，填入事件触发时所需要调用的中断服务程序。

下面举例说明中断的使用方法。

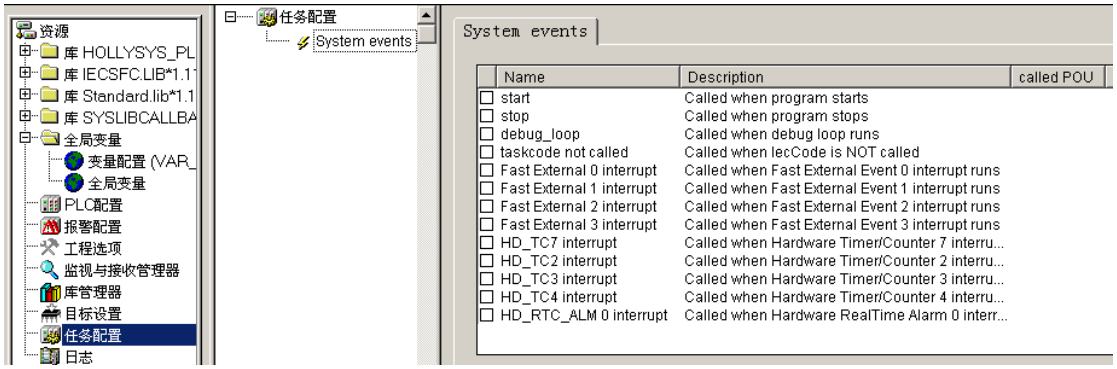


图 10-2-1 中断调用对话框

✧ 举例

✓ 要求

不受 PLC 扫描周期的影响，实现下列功能：

每当 I0.6 到达一个上升沿，PLC 立刻响应该脉冲，并产生一个中断，%MW100 中的值增加 10。

每当 I0.7 到达一个上升沿，PLC 立刻响应该脉冲，并产生一个中断，%MW102 中的值增加 15。

✓ 程序分析

按上述要求，硬件选用 CPU 模块 LM3106，软件需要使用以下指令：

Fast_ExINT_E（快速外部中断）

程序分为如下 3 个部分：

主程序—定义 CPU 模块 LM3106 的快速外部中断模式。

INT3PRO—I0.6 脉冲到达执行的中断程序，%MW100 中的值增加 10。

INT2PRO—I0.7 脉冲到达执行的中断程序，%MW102 中的值增加 15。

✓ 程序编制

主程序：

首先要将程序所用到的库 Hollysys_PLC_Ex_ExINT.lib 添加到库管理器。按照要求，I0.6 和 I0.7 每到达一个上升沿，就产生一个中断，I1.0 不使用，则 Fast_ExINT_E 功能块的 Mode=16#50，主程序的变量定义和梯形图如图 10-2-2 所示。

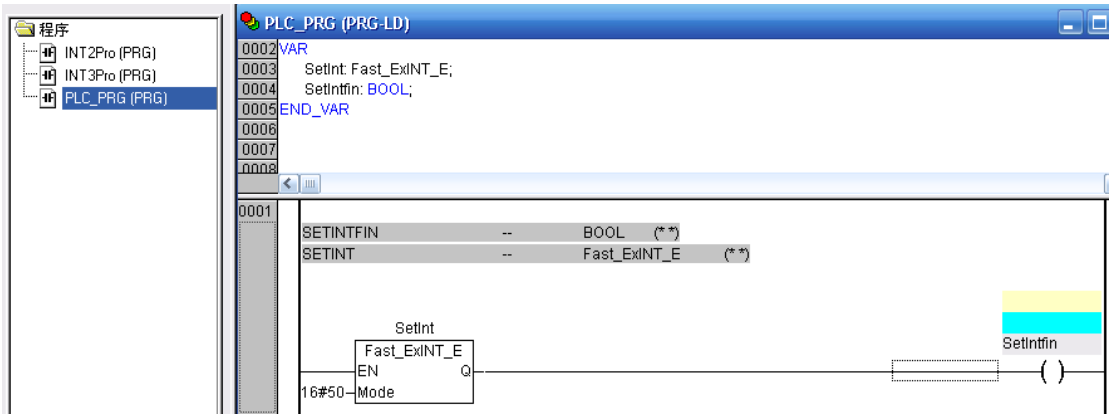


图 10-2-2 主程序的变量定义和梯形图

由于使用了快速外部中断 3（I0.6）和快速外部中断 2（I0.7），所以在如图 10-2-3 所示的位置前打勾。

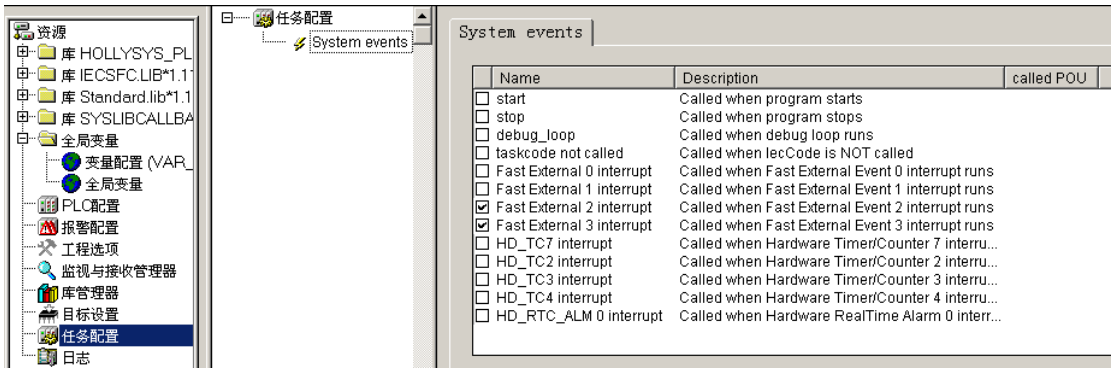


图 10-2-3 快速外部中断选择

在 Fast External 2 interrupt 和 Fast External 3 interrupt 后面分别创建一个子程序 INT2Pro 和 INT3Pro，分别点击 Create POU，则 2 个中断程序被创建成功，如图 10-2-4 所示。

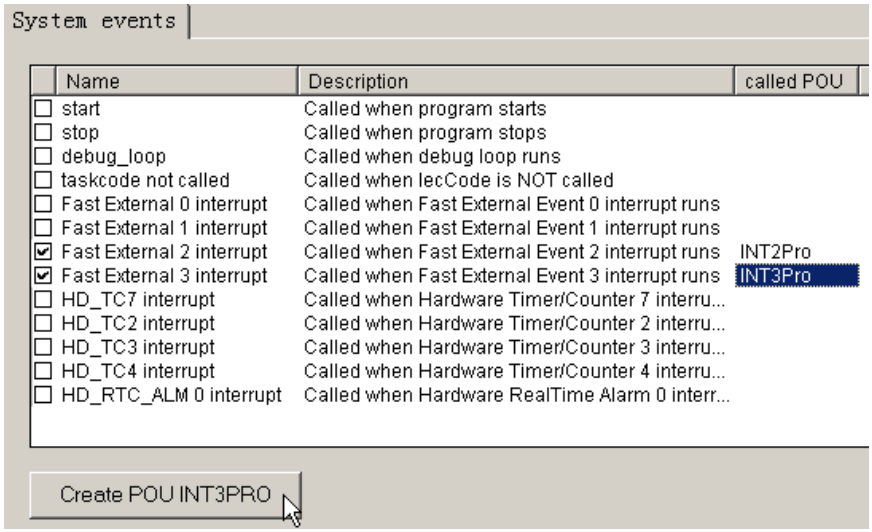


图 10-2-4 创建中断程序

所创建的中断程序默认为 ST 语言，可以选择转换为 LD 语言。转换前需要编译通过，如图 10-2-5 所示。

图 10-2-5 将 ST 转换为 LD

INT3Pro 中断程序梯形图如图 10-2-6 所示。

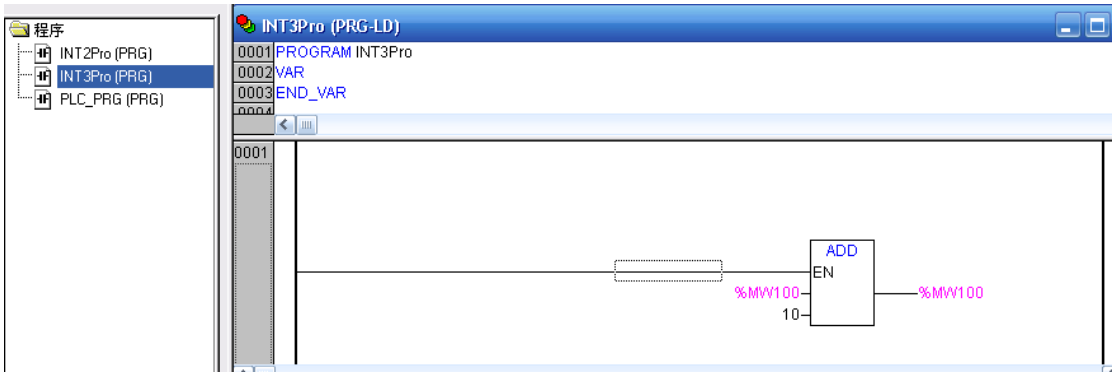


图 10-2-6 INT3Pro 中断程序梯形图

INT2Pro 中断程序梯形图如图 10-2-7 所示。

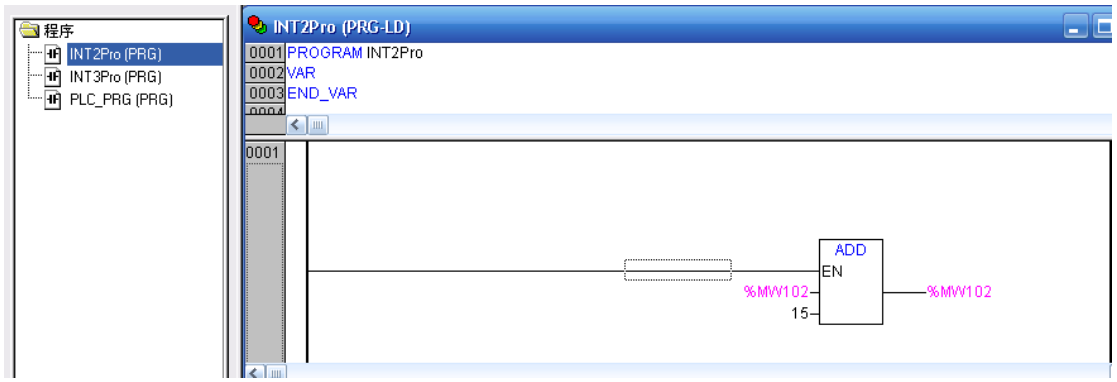


图 10-2-7　INT2Pro 中断程序梯形图

然后就可以在 INT2Pro 和 INT3Pro 中编写梯形图程序。当然也可以不进行语言的转化，使用 ST 直接编写。当系统事件触发时，调用相应的程序。

注意，系统事件不支持仿真模式，只有在程序编译通过且登录运行后才会响应该事件。当一次进行多个任务配置时，要求先进行全部编译之后，再保存文件。

10.3 模拟量功能使用

10.3.1　模拟量模块寻址

LM 系列 PLC 在使用模拟量模块时，首先需要知道该模块所占用的地址。无论是模拟量输入还是模拟量输出模块，都需要占用 PLC 的输入区或输出区。在 PLC 配置时，配置了模拟量模块，则系统自动给该模块配置了数据地址。

以 LM3310 为例。如图 10-3-1 所示，在 LM3107 模块后配置四通道模拟量输入模块 LM3310，则系统自动给 LM3310 分配了%IW2、%IW4、%IW6、%IW8 四个字的地址，每个字代表一个通道的采集数值。

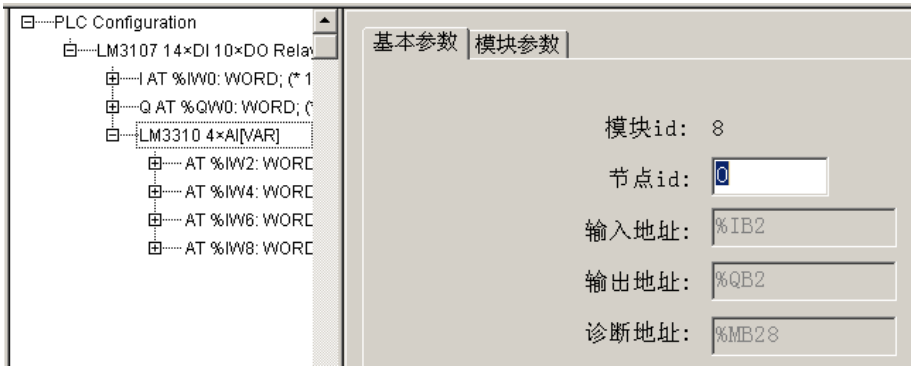


图 10-3-1 模拟量输入模块配置表

对于模拟量输出模块，则系统会自动分配 Q 区。诸如%QW2，若要输出模拟量，则需给%QW2 赋值。

在程序中，可以直接使用这些地址表示该通道的数值。通道数值与实际量程的关系，如下表所示。

表 10-3-1 模拟量模块的量程及范围

模块	量程	范围
LM3107E	输入：0~10V（电压）、0~20mA（电流）	0~10000
	输出：0~10V（电压）、0~20mA（电流）	0~4095
LM3310/A/B	输入：0~10V（电压）、0~20mA（电流）、4~20mA（电流）	0~65535
LM3320	输出：0~10V（电压）、0~20mA（电流）	0~4095
LM3313	输入：-10~10V (电压)、-20mA~20mA（电流）	-32000~32000
LM3330	输入：0~10V（电压）、0~20mA（电流）、4~20mA（电流）	0~65535
	输出：0~10V（电压）、0~20mA（电流）	0~4095

10.3.2 模拟量模块使用

如果 PLC 配置中使用了模拟量输入扩展模块，则程序中需调用模拟量输入扩展指令 ANALOG_IN。该指令所需填入的参数 Address，与该模拟量输入模块的节点 id 一致。如图 10-3-1 所示的 LM3310，则在程序中，需填入的 Address 值为 0。

如果需要使用多个模拟量输入模块，则需要配置多个 Analog_IN 指令，每个指令对应的 Address 输入值应与对应模块的节点 id 号保持一致。

如果 PLC 配置中使用了模拟量输出扩展模块，则程序中需调用模拟量输出扩展指令 ANALOG_OUT。该指令所需填入的参数 Address，与该模拟量输出模块的节点 id 一致。

如果需要使用多个模拟量输出模块，则需要配置多个 Analog_OUT 指令，每个指令对应的 Address 输入值应与对应模块的节点 id 号保持一致。

具体指令的使用方法，请参见《指令手册》。

在 PLC 配置模拟量模块后，还需对模块进行参数设置。只有设置正确的模块，才能正确的使用。参数设置主要包括滤波设置、各通道量程设置、使能设置等。具体设置参数请参见《硬件手册》。关于模拟量模块配置，也可以参见 7.3.2 章节。

注意：

1、 对于 LM3107E 模块，在使用到模拟量处理的时候，无需添加 ANALOG_IN 和 ANALOG_OUT 指令。

2、 对于 LM3330 模块，同时使用模拟量输入和模拟量输出功能时，需添加 ANALOG_IN 和 ANALOG_OUT 指令。

10.3.3 模拟量模块使用举例

以 LM3330 模块为例，说明模拟量的使用。LM3330 模块为 4 通道模拟量输入、1 通道模拟量输出模块。使用 LM3330 的第一个通道采集压力变送器的数值，其量程为 4~20mA，而输出通道用于控制阀门开度，量程为 0~10V。CPU 选用 LM3107 模块。

PLC 配置如图 10-3-2 所示，模块节点 id 为 0。

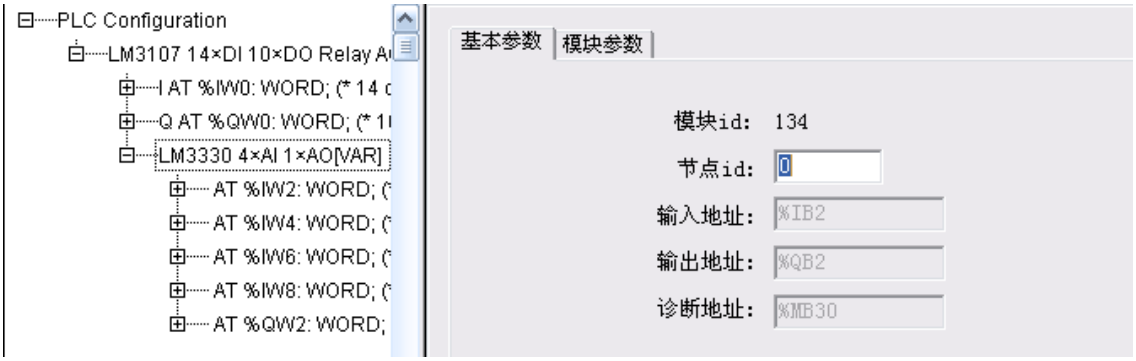


图 10-3-2 LM3330 配置

从配置中可以知道，LM3330 的输入通道所占的地址为%IW2、%IW4、%IW6、%IW8，输出通道占用地址为%QW2。配置完成后，需要对 LM3330 模块进行参数设置。为保证采集的数据比较稳定，设置模块滤波参数为 16，如图 10-3-3 所示。

图 10-3-3 设置滤波参数

设置完滤波参数后，还需要设置输入通道和输出通道的量程，如图 10-3-4 和 10-3-5 所示。将第一个输入通道%IW2 的量程设置为 4~20mA，输出通道%QW2 的量程设置为 0~10V。

图 10-3-4　设置输入通道的量程

图 10-3-5　设置输出通道的量程

设置完成后，开始进入编程。首先需要在程序中添加 ANALOG_IN 和 ANALOG_OUT 指令，参数 Address 均为 0。在使用这两个指令之前，首先需要添加相应的库指令。这两个指令均在 Hollysys_PLC_analog.lib 库中。关于库的操作，请参见 7.4.4 章节内容。

模拟量输入通道的量程为 4~20mA，参见表 10-3-1 可知，其对应的数值为 0~65535，即%IW2 的范围为 0~65535，0 表示采集到的电流信号为 4mA，而 65535 表示采集到的电流信号为 20mA。

模拟量输出通道的量程为 0~10V，参见表 10-3-1 可知，其对应的数值为 0~4095，即%QW2 的范围为 0~4095，0 表示输出 0V 电压，而 4095 表示输出 10V 电压 。

程序如图 10-3-6 所示。第一节和第二节添加了模拟量处理指令。第三节为模拟量输入和输出的读取和设置。这里，还用到了另外一个指令 H_E，这个指令的作用是完成工程量的转换，将%IW2 读取到的数值转换为实际的物理量，程序中将转换后的物理量放入 pressure 变量。%IW2 的范围为 0~65535，数据类型为整型，而 pressure 的范围为 4~20 mA，数据类型为 REAL 类型。关于该指令的用法，请参见《指令手册》。

在输出处理时，只是简单地将一个变量赋值为%QW2，当%QW2 的范围在 0~4095 之间时，LM3330 模块输出 0~10V 的电压。

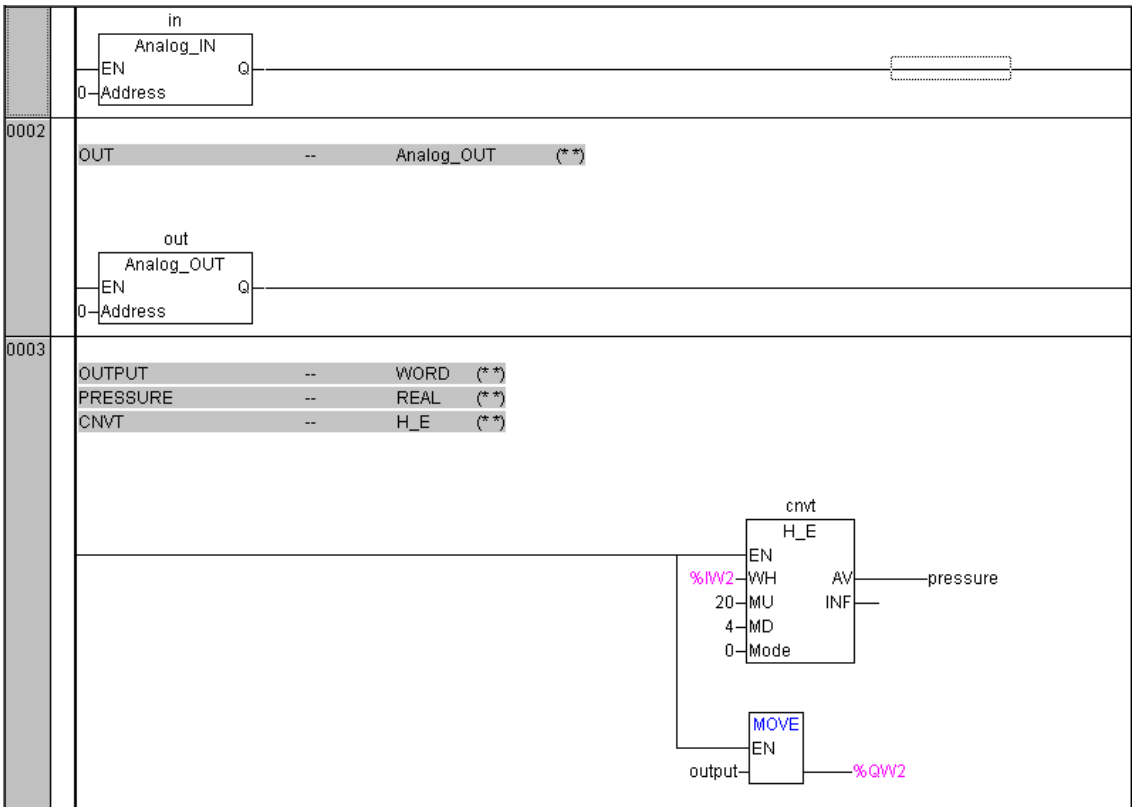


图 10-3-6　程序

10.4 DP 通讯

10.4.1　DP 通讯设置

LM 系列 PLC 提供 Profibus-DP 通讯处理。LM3401 为 DP 通讯模块，提供 DP 从站通讯功能。

在完成 PLC 配置后，需要对 LM3401 模块进行参数设置，主要设置参数为输入区和输出区大小。LM3401 模块参数设置如图 10-4-1 所示，其中 Value 值可选 0-64。在图 10-4-1 中，Value 值为 64。有关 LM3401 模块的具体技术规格请参见《硬件手册》。

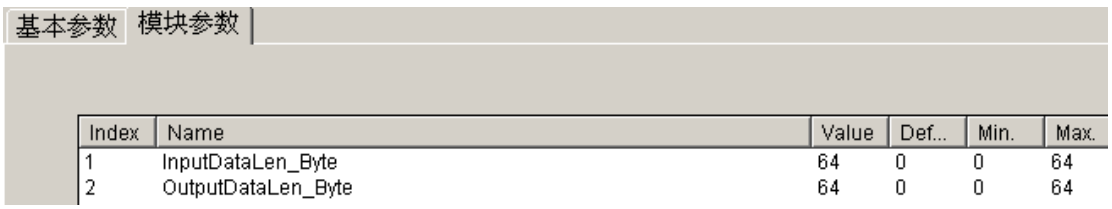


图 10-4-1　LM3401 模块参数设置

与模拟量使用方法类似，在使用 DP 通讯时，也需要添加一个指令 DP_Slave。DP_Slave 指令的程序，如图 10-4-2 所示。其中 Address 处的输入值 0 应与 PLC 配置表中 LM3401 的节点 id 一致。

当 EN 置位时，对 DP 模块进行扫描。当 EN 复位时，不对 DP 模块进行扫描。

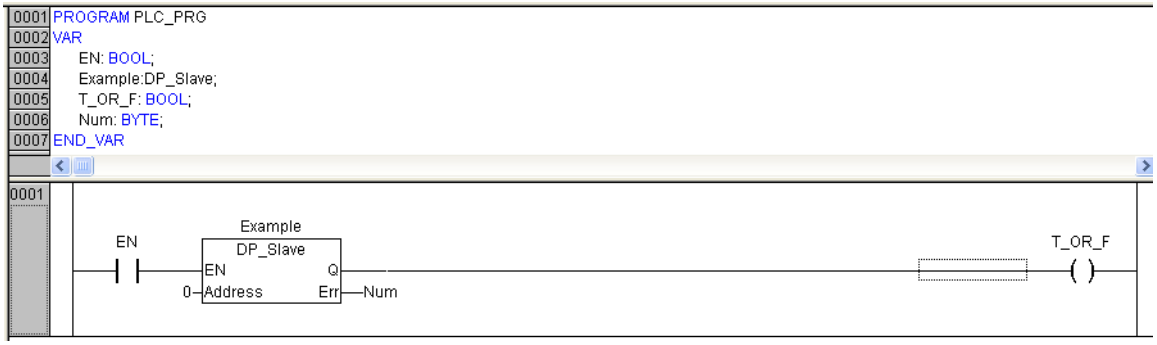


图 10-4-2 DP_Slave 功能块

配置了 LM3401 后，系统自动分配了输入区和输出区，如图 10-4-3 所示。输入区从%IW2 开始的 64 个字节，输出区从%QW2 开始的 64 个字节。DP 主站设备与 LM 系列 PLC 通讯就是与这些输入区和输出区进行通讯，完成数据交换。输入区用于存放主站发送过来的数据，而若 LM 系列 PLC 要发送数据给主站，需要将数据放置于 LM3401 的输出区。

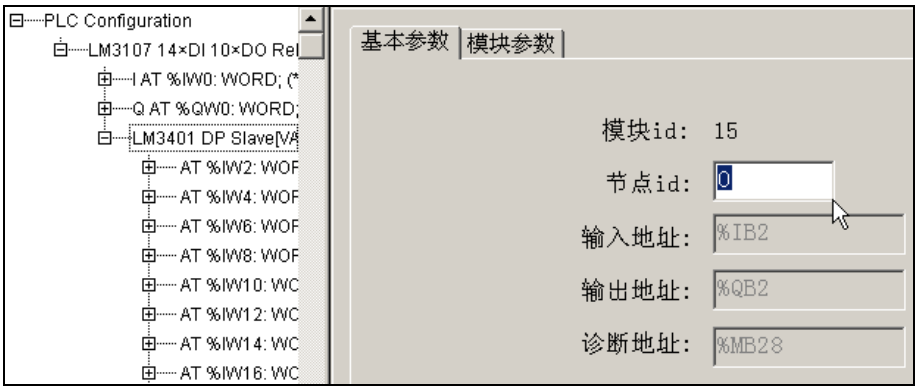


图 10-4-3 DP_Slave 配置表

10.4.2 DP 通讯举例

下面作一个简单的 Profibus-DP 功能块的应用实例。

程序要求：PLC 通过 DP 从站模块向 DP 主站模块发送 8 个字节的数据，同时从 DP 主站模块接收 8 个字节的数据。

变量定义：

```
PROGRAM    PLC_PRG
VAR
    EN: BOOL;
    Example: DP_Slave;
    T_OR_F: BOOL;
    SendDataA: WORD;    PLC 向 DP 主站发送的数据 A
    SendDataB: WORD;    PLC 向 DP 主站发送的数据 B
    SendDataC: WORD;    PLC 向 DP 主站发送的数据 C
    SendDataD: WORD;    PLC 向 DP 主站发送的数据 D
```

RecDataA: WORD;　DP 主站向 PLC 发送的数据 A
RecDataB: WORD;　DP 主站向 PLC 发送的数据 B
RecDataC: WORD;　DP 主站向 PLC 发送的数据 C
RecDataD: WORD;　DP 主站向 PLC 发送的数据 D
END_VAR

软件配置：
配置 DP 从站模块 LM3401，如图 10-4-4 所示。
InputDataLen_Byte 为 DP 主站向 PLC 发送的数据长度，输入接收的字节数 8。
OutputDataLen_Byte 为 PLC 向 DP 主站发送的数据长度，输出发送的字节数 8。

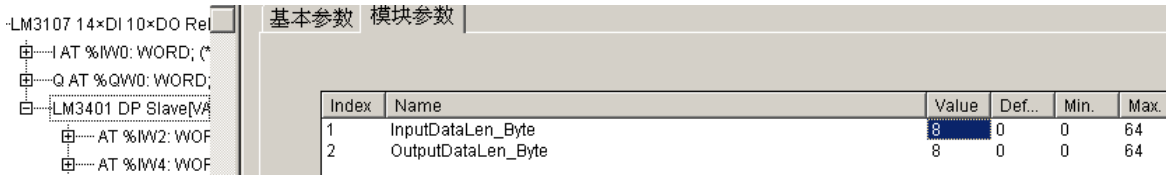


图 10-4-4　LM3401 模块参数设置

DP_Slave 模块中的 Address 与图 10-4-5 所示的节点号一致，DP 主站向 PLC 发送的数据 A、B、C、D 分别存放在图 10-4-5 所示的%IW2、%IW4、%IW6、%IW8 之中。

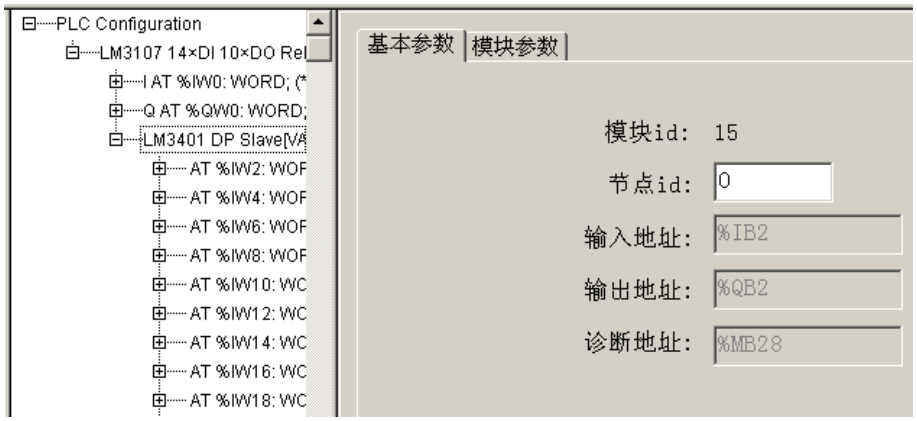


图 10-4-5 LM3401 配置表

PLC 向 DP 主站发送的数据 A、B、C、D 分别存放在图 10-4-6 所示的%QW2、%QW4、%QW6、%QW8 之中。

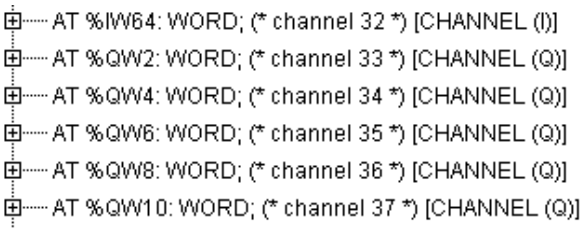


图 10-4-6　LM3401 通道

配置 DP 主站的接收和发送区，DP 从站的 QW 区数据会自动传送到 DP 主站的接收区，DP 主站的发送区数据会自动传送到 DP 从站的 IW 区。梯形图如图 10-4-7 所示。

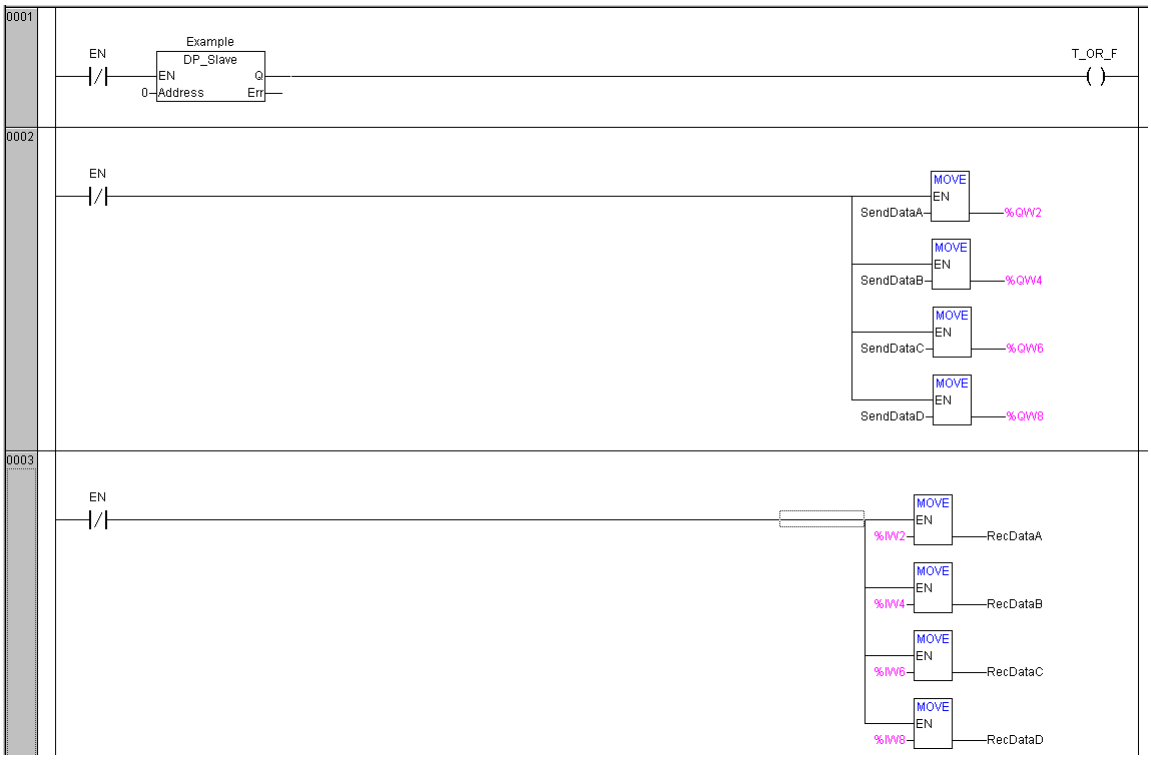


图 10-4-7　LM3401 梯形图

系统上电后，程序将 SendDataA、SendDataB、SendDataC、SendDataD 中的数据复制到%QW2、%QW4、%QW6、%QW8 中，%QW2、%QW4、%QW6、%QW8 中的数据会自动发送到 DP 主站的接收区。

DP 主站发送的数据自动存放在%IW2、%IW4、%IW6、%IW8 中，程序将%IW2、%IW4、%IW6、%IW8 中的数据分别复制到 RecDataA、RecDataB、RecDataC、RecDataD 中。

在主站配置中，其配置地址根据 DP 模块的地址顺序计算，假如按位计算，第一个位的地址为 1，诸如%IX2.0 或%QX2.0，第二个位地址为 2，诸如%IX2.1 或%QX2.1。按字节计算，第一个字节地址为 1，诸如%IB2 或%QB2，第二个字节地址为 2，诸如%IB3 或%QB3，以此类推。

10.5 以太网通讯

10.5.1　以太网通讯设置

LM 系列 PLC 提供以太网通讯处理。LM3403 为以太网通讯模块，提供 Modbus TCP 从站通讯功能。

在完成 PLC 配置后，需要对 LM3403 模块进行参数设置，主要设置参数为输入区和输出区大小。LM3403 模块参数设置如图 10-5-1 所示。在图 10-5-1 中，分别配置 IP 地址 IP_Address、子网掩码 Subnet_Mask、网关 Gateway_Adress、输入区大小 WriteDataLen_Byte 和输出区大小 ReadDataLen_Byte、MAC 地址 MAC_Address 等参数。有关 LM3401 模块的具体技术规格请参见《硬件手册》。

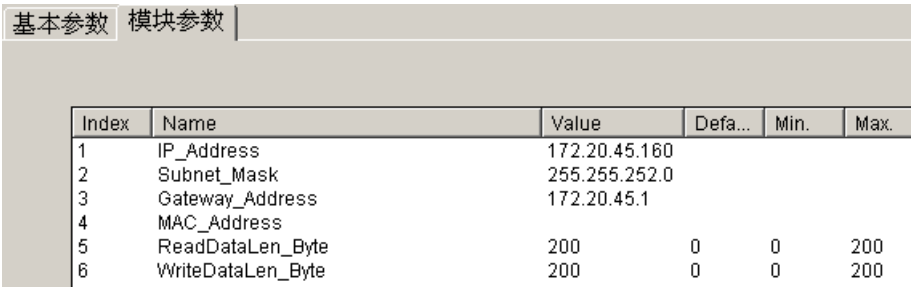


图 10-5-1　LM3403 模块参数设置

与模拟量使用方法类似，在使用以太网通讯时，也需要添加一个指令 EtherNet_TCP。EtherNet_TCP 指令的程序如图 10-5-2 所示。其中 Address 处的输入值 0 应与 PLC 配置表中 LM3403 的节点 id 一致。

当 EN 置位时，对以太网模块进行扫描。当 EN 复位时，不对以太网模块进行扫描。

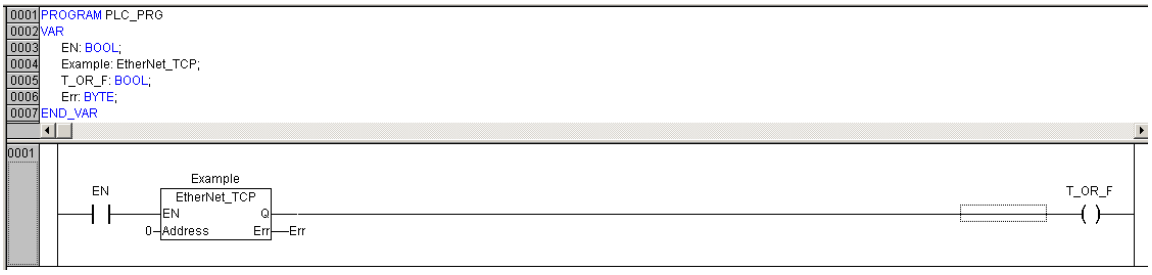


图 10-5-2 EtherNet_TCP 梯形图

配置了 LM3403 后，系统自动分配了输入区和输出区，如图 10-5-3 所示。输入区从%IW2 开始的 200 字节，输出区从%QW2 开始的 200 个字节。Modbus TCP 主站设备与 LM 系列 PLC 通讯，就是与这些输入区和输出区进行通讯，完成数据交换。输入区用于存放主站发送过来的数据，而若 LM 系列 PLC 要发送数据给主站，需要将数据放置于 LM3403 的输出区。

图 10-5-3　EtherNet_TCP 配置表

10.5.2 以太网通讯举例

本例中，PC 机（MODBUS/TCP 主站）通过以太网模块向 PLC 输入区发送 2 个字的数据，从 PLC 输出区接收 2 个字的数据，同时 PC 机按位操作向 PLC 输入区发送 1 个位，从 PLC 输出区接收 1 个位。

变量定义

```
PROGRAM PLC_PRG
VAR
    EN: BOOL;
    Example: EtherNet_TCP;
T_OR_F: BOOL;
SendDataA: WORD;   (*PLC 向 MODBUS/TCP 主站发送的数据 A*)
SendDataB: WORD;   (*PLC 向 MODBUS/TCP 主站发送的数据 B*)
SendBitC: BOOL;    (*PLC 向 MODBUS/TCP 主站发送的位 C*)
RecDataA: WORD;    (*MODBUS/TCP 主站向 PLC 发送的数据 A*)
RecDataB: WORD;    (*MODBUS/TCP 主站向 PLC 发送的数据 B*)
RecBitC: BOOL;     (*MODBUS/TCP 主站向 PLC 发送的位 C*)
END_VAR
```

软件配置

- 配置LM3403以太网模块如图10-5-4。

图 10-5-4 LM3403 配置

- ✓ IP_Address 为本以太网模块 IP 地址(必须与 PC 机在同一网段,且无冲突的 IP 地址)。
- ✓ Subnet_Mask 为子网掩码，与 PC 机的子网掩码一致。
- ✓ GateWay_Addres 为网关地址。
- ✓ MAC_Address 不填。
- ✓ ReadDataLen_Byte 为 PC 机向 PLC 发送的数据长度，输入接收的字节数 8（必须大于实际用到的长度，且最大 200）。
- ✓ WriteDataLen_Byte 为 PLC 向 PC 机发送的数据长度，输出发送的字节数 8（必须大于实际用到的长度，且最大 200）。

- 以太网功能块中的Address与图10-5-5所示的节点id一致，PC机向PLC发送的数据A、B分别存放在下图所示的%IW4、%IW6，位C存放在%IX8.0。

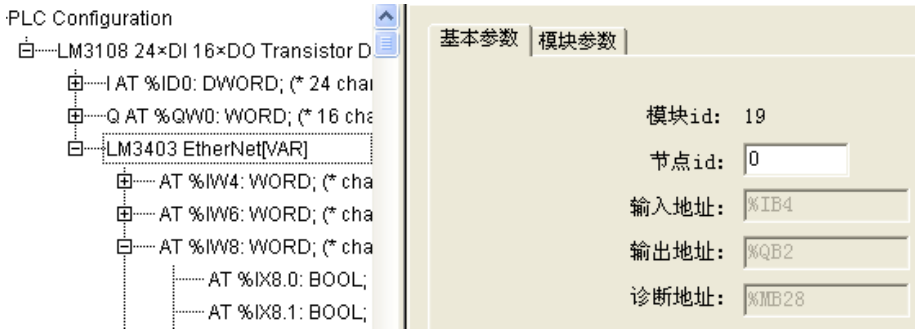


图 10-5-5

➢ PLC向PC机发送的数据A、B分别存放在图10-5-6所示的%QW2、%QW4之中，位C存放在%QX6.0。

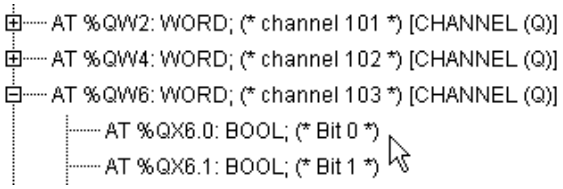


图 10-5-6 输出区地址

程序如 10-5-7 所示。在主站配置中，其配置地址根据以太网模块的地址顺序计算，假如按位计算，第一个位的地址为 1，诸如%IX2.0 或%QX2.0，第二个位地址为 2，诸如%IX2.1 或%QX2.1。按字节计算，第一个字节地址为 1，诸如%IB2 或%QB2，第二个字节地址为 2，诸如%IB3 或%QB3，以此类推。

图 10-5-7 程序

注意：在以太网的主站配置中，I 区对应的位地址类型为 0x，字地址类型为 4x，Q 区对应位地址类型为 1x，子地址类型为 3x。

第11章 视图

视图（Visualization）是 PowerPro 软件的一个组件，能够以图形的方式显示工程变量及其变化规律，用于实现控制过程的可视化。因此，视图是 PLC 的人机界面（Human Machine Interface，HMI）。

PowerPro 软件的编程系统带有一个集成的视图编辑器。在开发控制系统应用程序的过程中，PowerPro 软件允许用户开发视图对象来观察和操作 PLC 的数据，而无需使用其它的开发工具。

11.1 创建视图文件

启动 PowerPro 软件，创建工程 project1.pro。在对象组织器中，点击“视图”选项卡，如图 11-1-1 所示。

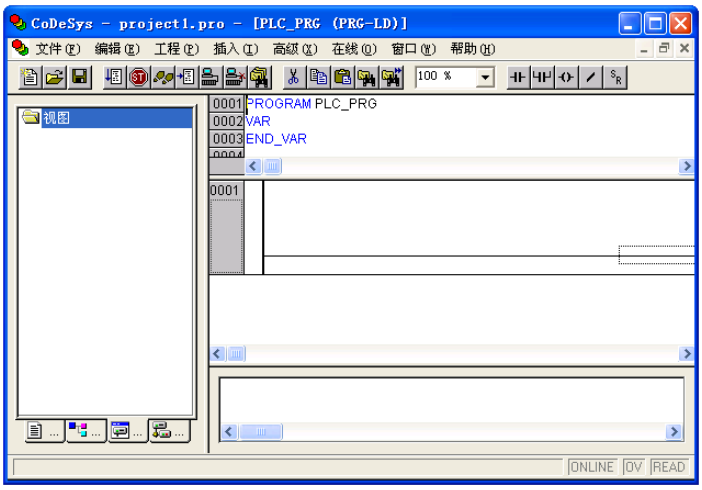


图 11-1-1 视图选项卡

在“视图”选项卡中点击鼠标右键，选择“添加”按钮，如图 11-1-2 所示。

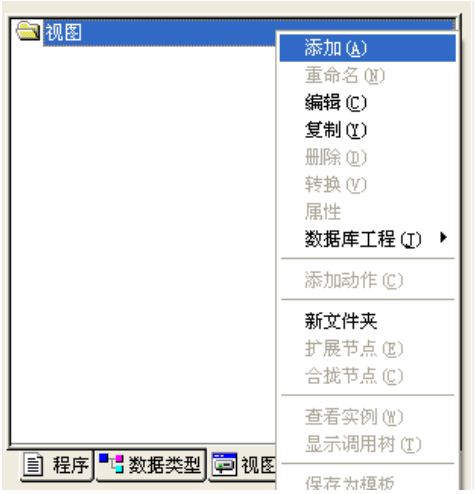


图 11-1-2 添加视图

弹出视图名称对话框，如图 11-1-3 所示。

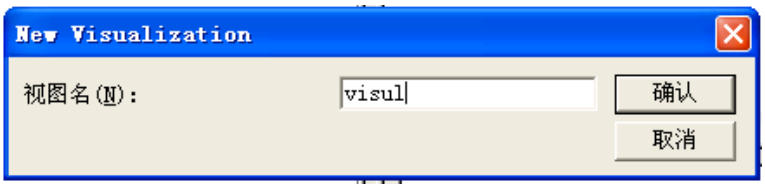


图 11-1-3 视图名称

在“可视名”中输入视图的名称，例如 visu1，点击“确认”按钮，则在视图选项卡中创建了一个名为 visu1 的视图，右侧的工作区域为视图编辑区域，如图 11-1-4 所示。

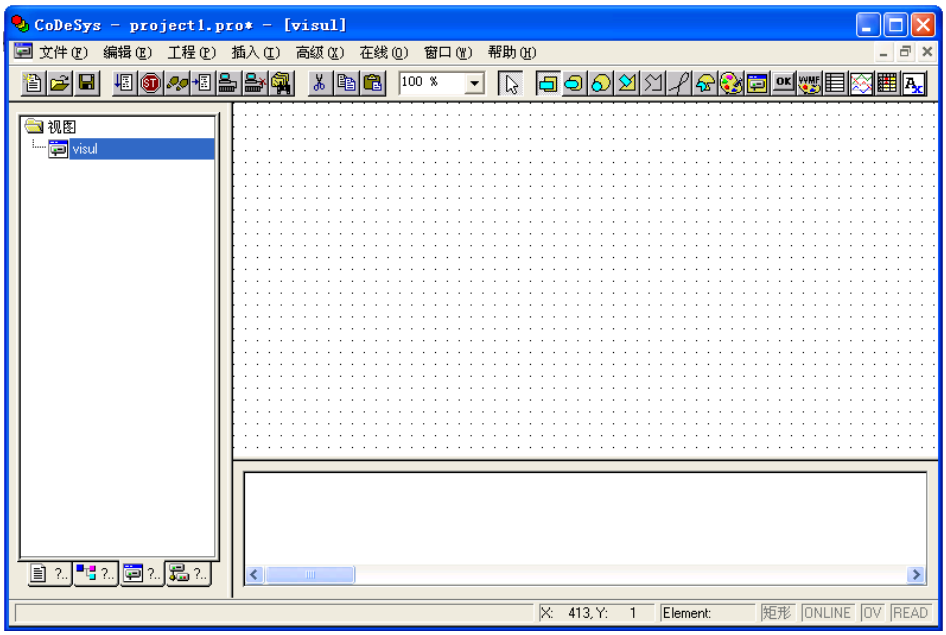


图 11-1-4 创建视图 visu1

11.2 视图编辑工具

编辑视图的工具位于视图编辑区域顶端的标题栏内，包括下拉菜单和快捷工具按钮。

点击标题栏的“插入”菜单，出现编辑视图的“插入”下拉菜单，如图 11-2-1 所示，可以选择不同的插入对象来添加所需的视图。这些视图对象是视图文件的基本组成单元。

图 11-2-1 视图“插入”菜单

点击标题栏的“高级”菜单，出现编辑视图的“高级”下拉菜单，如图 11-2-2 所示，可以对视图对象的属性进行配置。在视图编辑区域点击鼠标右键也会弹出此菜单。

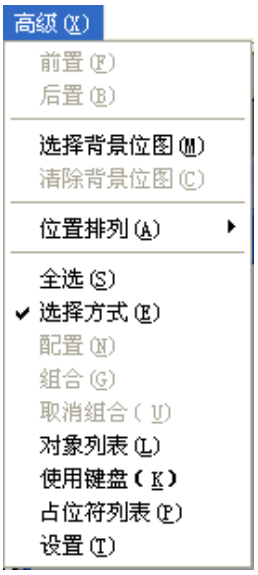


图 11-2-2 视图“高级”菜单

编辑视图的快捷工具按钮位于视图编辑区域顶端标题栏的工具栏中，如图 11-2-3 所示，与“插入”菜单完全相同。编辑视图的工具共有 15 种，其功能如表 11-2-1 所示。

图 11-2-3 视图快捷工具按钮菜单

表 11-2-1 编辑视图工具

名称	快捷工具	功能描述
矩形		在当前视图中插入矩形。
圆角矩形		在当前视图中插入圆角矩形。
椭圆		在当前视图中插入圆或椭圆。
多边形		在当前视图中插入多边形。
折线		在当前视图中插入直线或折线。
曲线		在当前视图中插入贝塞尔曲线。
Pie 图		在当前视图中插入饼图（圆或椭圆的楔形图）。
位图		在当前视图中插入位图文件。
视图		在当前视图中插入已经创建的其它视图。
按钮		在当前视图中插入按钮。
WMF 文件		在当前视图中插入 WMF 文件（Windows Metafile，即 Windows 图元文件格式）。
表		在当前视图中插入表格。
趋势图		在当前视图中插入趋势图。
报警表		在当前视图中插入报警表格。
ActiveX 控件		在当前视图中插入 ActiveX 控件。

11.3 视图编辑方法

11.3.1 绘制视图

在编辑视图时，将鼠标移到视图编辑工具按钮处，会看到相关工具的自动提示。

➢ 绘制矩形、圆角矩形、圆或椭圆等规则视图

在需要的工具按钮位置点击鼠标左键，选中所需的视图，然后将鼠标移动到视图编辑区域中需要绘制所选视图的位置。按下鼠标左键并拖动到所需视图的尺寸，抬起鼠标左键即可

完成所选视图的绘制。

- 绘制多边形、折线等不规则视图

在需要的工具按钮位置点击鼠标左键，选中所需的视图，然后将鼠标移动到视图编辑区域中需要绘制所选视图的位置。在不规则视图的各个顶点处依次点击鼠标左键，可以画出不规则视图的各条边，在最后的顶点处双击鼠标左键，即可完成不规则视图的绘制。

- 绘制曲线

一条曲线由初始点、中间点和结束点等三个点来确定。依次点击鼠标左键三次即可完成一条曲线的绘制。点击鼠标左键确定曲线的初始点、中间点和结束点，以确定曲线的长度和弧度。鼠标左键第三次点击后，可以通过移动鼠标来改变曲线结束点的位置。双击鼠标左键结束曲线绘制过程。

- 复制视图

使用“编辑”/“复制”命令，或者<Ctrl>+<C>复合键，可以复制所选择的一个或多个视图对象。复制视图的另外一种方法是，选择要复制的视图对象，按下<Ctrl>键的同时点击该视图对象，则会在原来的视图对象上产生复制的视图对象。

- 视图中的状态栏

在视图中，鼠标指针的 X、Y 位置显示在状态栏上。状态栏上的位置值总是相对于视图的左上角。当鼠标位于视图对象上或者正在编辑视图对象时，状态栏上显示该视图对象的顺序号码。如果选择了要插入的视图对象，状态栏也会显示该视图对象的名称。

11.3.2 布置视图

在绘制视图的过程中，需要对视图进行修改和布置。

- 选择

用鼠标左键点击视图对象可以将其选中。

可以按下<Tab>键，选择视图对象中的第一个视图对象，再次按下<Tab>键则选择下一个视图对象。如果同时按下<Tab>和<Shift>键,可以按照视图对象中相反的顺序选择。

在选择一个视图对象后按下<Shift>键，同时点击相应的视图对象，可以选择多个视图对象。也可以按住鼠标左键不放，在要选择的视图对象上拉一个窗口，同样可选择多个对象。

- 全选

标题栏“高级”菜单中的“全选”菜单可以将当前视图中的所有视图对象全部选中。

- 选择方式

如果标题栏“高级”菜单的“选择方式”菜单前有“√”，或者快捷工具栏中表示鼠标状态的快捷工具按钮被按下时，表示此时处于“选择”状态，可以选择视图对象。否则处于绘图状态。

- 选择方式和插入方式的切换

视图对象插入后，自动回到选择方式。如果要用同样方式再插入对象，可以再次选择相应的菜单命令。

切换的另外一种方法是在选择方式下，同时按下<Ctrl>键和鼠标右键，可以在选择方式和插入方式之间快速转化。

- 拖动

点击鼠标左键选择视图对象。在所选的视图对象上按下鼠标左键或按下方向键，可以拖动一个或多个视图对象。

- 修改

用鼠标点击对象或者按下<Tab>键，可以选择一个视图对象。在所选择视图对象的周围有一些小的黑色矩形。通过点击这些小的黑色矩形，按住鼠标左键，可以改变视图对象的大小，控制视图对象的轮廓。

选择一个视图对象后，同时显示旋转点。旋转点是一个中间带有白色十字的黑色圆圈，可按一定的角度使对象绕这点旋转。可以按下鼠标左键拖动旋转点。

对于多边形，使用同样的方法拖动每个顶角。鼠标拖动时，按下<Ctrl>键，可以插入一个顶角。按下<Shift>+<Ctrl>键，可以删除一个顶角。

在视图编辑区，点击鼠标右键，会弹出如图 11-3-1 所示的菜单。

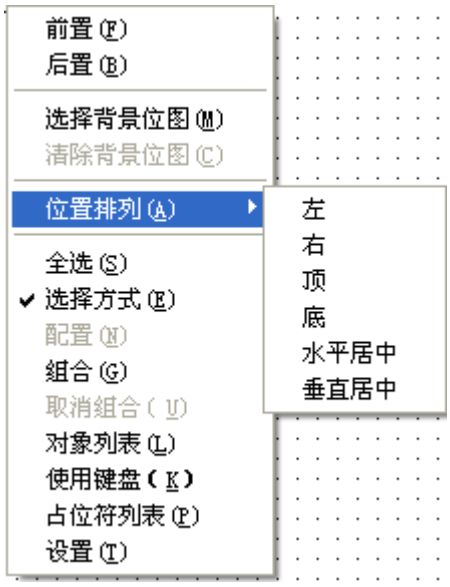


图 11-3-1 视图右键菜单

下面将依次介绍菜单中的选项。

➢ 前置

将选择的视图对象置于最上一层。选择一个视图对象，点击标题栏“高级”菜单中的“前置”菜单，可以将该视图对象放到其它视图对象的前面。

当视图对象以不同对象号显示时，视图对象实际上相当于以不同的图层来显示。将几个视图置于同一位置处时，会发现视图中对象号最大的置于最上层。如需使某一视图置于最上层时，可用此命令使该视图前置，且对象号变为当前最大值。

➢ 后置

将选择的视图对象置于最下一层。选择一个视图对象，点击标题栏 “高级”菜单中的“后置”菜单，可以将该视图对象放到其它视图对象的后面。

与“前置”同理，可以使某一视图对象置于最底层，且将视图中对象号变为 0。

➢ 选择背景位图

从相应的路径选择合适的背景位图，选择“打开”，在视图中即可呈现所选择的背景位图。

➢ 清除背景位图

在背景位图设置好后，如果背景位图不满足需求，选择“清除背景位图”可对其进行清除。

➢ 位置排列

选择多个视图对象，点击标题栏“高级”菜单中的“位置”菜单，可以对该多个视图对象的位置进行排列。

“位置”菜单有以下几个选项：

“左”表示被选中的视图对象以最左侧视图对象的左边沿为基准对齐排列，即左对齐。

“右”表示被选中的视图对象以最右侧视图对象的右边沿为基准对齐排列，即右对齐。

“顶”表示被选中的视图对象以最顶侧视图对象的顶边沿为基准对齐排列，即顶对齐。

“底”表示被选中的视图对象以最底侧视图对象的底边沿为基准对齐排列，即底对齐。

“水平居中”表示每个视图对象以所有视图对象的平均水平中心为基准对齐排列。

“垂直居中”表示每个视图对象以所有视图对象的平均垂直中心为基准对齐排列。

- 配置

对于不同的视图对象，其配置中的分类选项不同。对于“配置”，一定要在视图对象被选中的情况下，才显示该对象的一些配置，否则，该项为灰体。

- 组合

选择多个视图对象，点击标题栏“高级”菜单中的“组合”菜单，可以将多个视图对象组合成一个视图对象。组合后视图对象的行为与一个视图对象的行为相同。

- 取消组合

选择一个组合的视图对象，点击标题栏“高级”菜单中的“取消组合”菜单，可以将该组合的视图对象分解为多个单独的视图对象。

另外，视图编辑区右键菜单中的内容除了以上介绍的选项外，还包括对象列表、使用键盘、占位符列表和视图设置，这四个选项将在下面进行介绍。

11.3.3 对象列表

点击标题栏“高级”菜单中的“对象列表”菜单，打开视图对象列表对话框，如图 11-3-2 所示，其中包括视图对象的号码、类型和位置等信息。点击该对话框右侧的工具按钮，可以对其进行编辑。在绘图区点击鼠标右键，选中“对象列表”，也会弹出此对象列表。

- “确认”按钮：当对对象列表内容操作完毕后，点击“确认”按钮，关闭对话框。
- “最前”按钮：把选择的视图对象放在最上层，此时对象号最大。
- “最后”按钮：把选择的对象放在最底层，此时对象号最小。
- “向前一步”按钮：把选择的视图对象向上移动一层，此时对象号增大一号。
- “向后一步”按钮：把选择的视图对象向下移动一层，此时对象号减小一号。
- “删除”按钮：删除选择的对象。
- “取消”按钮：取消上一次操作
- “重做”按钮：恢复上一次操作。
- “编辑”按钮：对视图对象进行编辑操作。

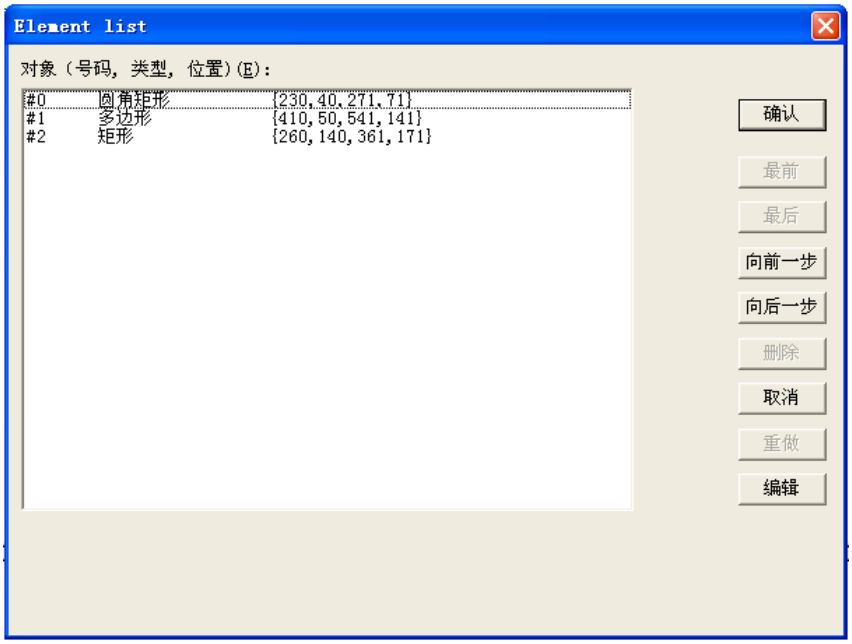


图 11-3-2 视图对象列表对话框

11.3.4 使用键盘

可以通过选择“Shift”、“Ctrl”及“Key”下拉菜单里的可选按键来实现动作的运行。

通过“Expression”表达式来描述要实现的动作或显示运行动作后的结果。使用键盘对于视图的每个物体可独自设置，因此，在不同视图中相同的按键设置可以起到不同作用。

✧ 举例

下面的表 11-3-1 和表 11-3-2 为 VIS_1 和 VIS_2 两个视图的不同按键设置。

表 11-3-1 VIS_1 键盘设置

Shift	Ctrl	Action	Key	Expression
x		Toggle	A	PLC_PRG.automatic
	x	Zoom	Z	VIS_2

表 11-3-2 VIS_2 键盘设置

Shift	Ctrl	Action	Key	Expression
x		Exec	E	INTERN LANGUAGE DEUTSCH
	x	Zoom	Z	VIS_1

如果在“在线模式”下，且当前视图为 VIS_1，按住“Shift+A”复合键，则 PLC_PRG 中的 automatic 变量会触发“toggle”，另外，如果按住“Ctrl+Z”复合键，则会自动跳到视图 VIS_2。

对于“Action”的各选项，请参见表 11-3-3。

表 11-3-3 键盘动作

Action	含义
Toggle	切换变量
Tap true	键控变量置为真
Tap false	键控变量置为假
Zoom	缩放视图
Exec	执行程序
Text	文本显示的文本输入变量

可以通过选择按键配合“shift”或“ctrl”组成复合键，触发相应的动作。

11.3.5 占位符列表

占位符列表如图 11-3-3 所示。

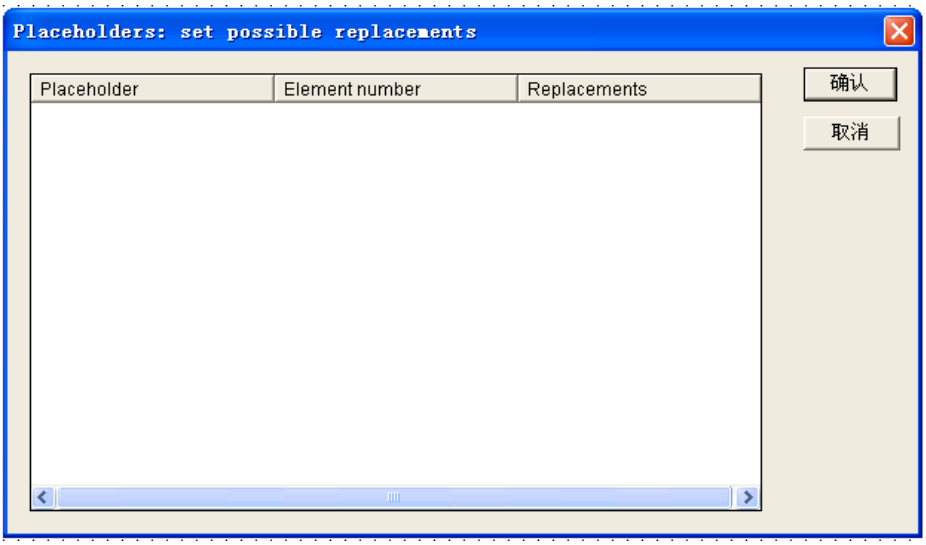


图 11-3-3 占位符列表

- Placeholder：列出用于配置所有视图对象的占位符。
- Element number：显示包含占位符的对象号。
- Replacements：可以输入一些字符串。例如文本、变量或表达式等。

11.3.6 视图设置

视图设置如图 11-3-4 所示。

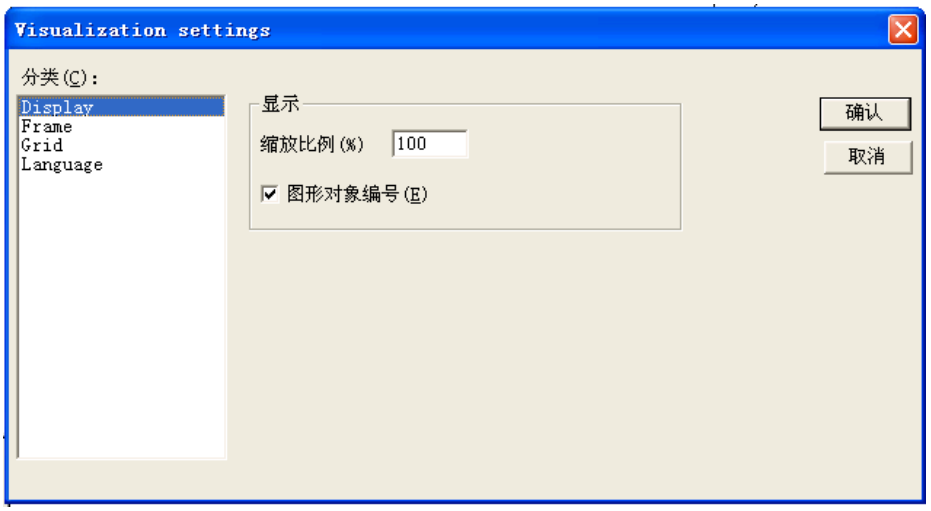


图 11-3-4 视图设置

- Display：填入相应扩大或缩小比例改变屏幕大小，可以选择是否在视图中显示对象号，如图 11-3-4 所示。
- Frame：框架设置，如图 11-3-5 所示。

选择“自动滚动”，在画某个对象或移动某个对象时达框架边界时，视图窗口会跟随该对象自动移动。

选择“包含背景视图”，背景视图将会符合窗口框架大小，否则只考虑对象。

选择“在线方式最佳匹配”，视图以最佳效果进行运行模拟。

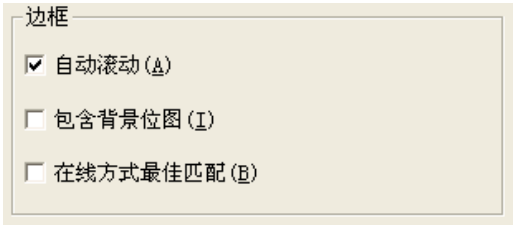


图 11-3-5 框架设置

- Grid：网格设置，如图 11-3-6 所示。

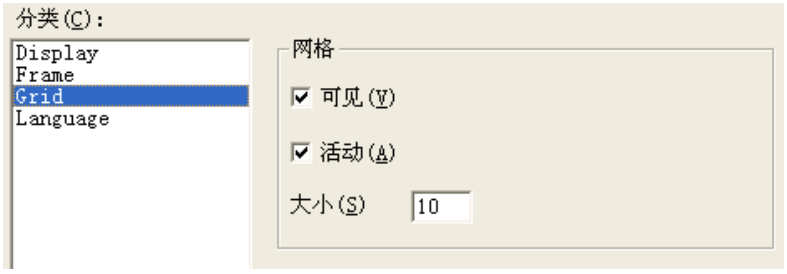


图 11-3-6 网格设置

选择网格“可见”，则会在窗口中呈现网格，可以通过改变网格大小调整网格间距，最小值为 10。

选择“活动”，则会以网格为单位进行移动，否则可实现在窗口任意空间移动。

- Language：PLC 不支持此项功能。

11.4 视图属性配置

11.4.1 属性配置方法

在编辑视图的过程中，选中某一个视图对象并点击鼠标右键，或点击标题栏的“高级”菜单，则会弹出“高级”下拉菜单，并且激活了其中的“配置”菜单，如图 11-4-1 所示。对于“配置”菜单，一定要在视图对象被选中的情况下才被激活，否则该项为灰体。

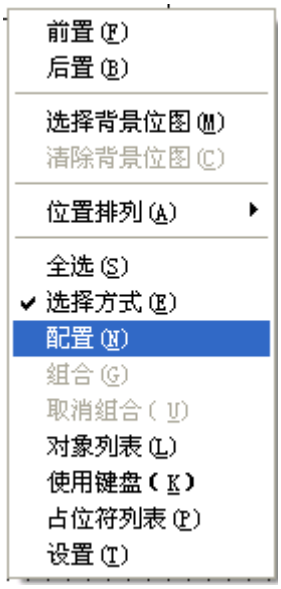


图 11-4-1 视图“高级”/“配置”菜单

点击此“配置”菜单，弹出视图对象属性配置对话框，如图 11-4-2 所示，则可以对所选中的视图对象的属性进行配置。用鼠标左键双击视图对象则可以快速打开此对话框。

图 11-4-2 视图对象属性配置对话框

11.4.2 视图对象的属性

对于不同的视图对象，其属性配置中的分类选项不同，属性配置对话框右侧的对应参数内容也不相同。可以通过设置不同的参数变量来定义该视图对象，从而改变视图对象的属性。表 11-4-1 给出了视图属性与视图对象的对应关系。其中符号“√”表示有对应关系。

表 11-4-1 属性分类与视图对象的对应关系

视图属性		视图对象														
属性分类	属性名称	矩形	圆角矩形	椭圆	多边形	折线	曲线	饼图	位图	视图	按钮	图元文件	表格	趋势图	报警表	控件
Shape	形状	√	√	√	√	√	√									
Text	文本	√	√	√	√	√	√	√	√	√	√	√				
Text variables	文本变量	√	√	√	√	√	√	√	√	√	√	√				
Line width	线宽	√	√	√	√	√	√	√	√	√		√				
Colors	颜色	√	√	√	√	√	√	√						√		
Color variables	颜色变量	√	√	√	√	√	√	√	√	√		√				
Motion absolute	绝对移动	√	√	√	√	√	√	√	√	√		√				
Motion relative	相对移动	√	√	√					√	√		√				
Variables	变量	√	√	√	√	√	√	√	√	√	√	√				
Input	输入	√	√	√	√	√	√	√	√	√	√	√				
Text for tooltip	提示文本	√	√	√	√	√	√	√	√	√	√	√	√	√	√	
Security	安全	√	√	√	√	√	√	√	√	√	√	√	√	√	√	
Programmability	编程	√	√	√	√	√	√	√	√	√	√	√				
Angle	角度							√								
Bitmap	位图属性								√		√					
Visualization	视图属性									√						
Group	组框架属性											√				
Table	表格属性												√			
Columns	列												√		√	
Rows	行												√			
Selection	选择												√			
Trend	趋势图属性													√		
Alarm table	报警表属性														√	
Settings for sorting	排序设置														√	
Settings for alarm table	报警表设置														√	
Control	控件类型															√
Method calls	方法调用															√
Display	显示															√

11.5 视图静态属性

所谓视图的静态属性是指描述图形对象基本形状的参数，主要包括形状、文本、线条宽度、颜色等。

11.5.1 形状

形状 Shape 属性用来定义视图对象的形状。对于规则视图，形状 Shape 属性可以选择矩形、圆角矩形、椭圆或直线等。对于不规则视图，形状 Shape 属性可以选择多边形、折线或曲线等，如图 11-5-1 所示。形状 Shape 属性的改变只在所确定的范围内进行。

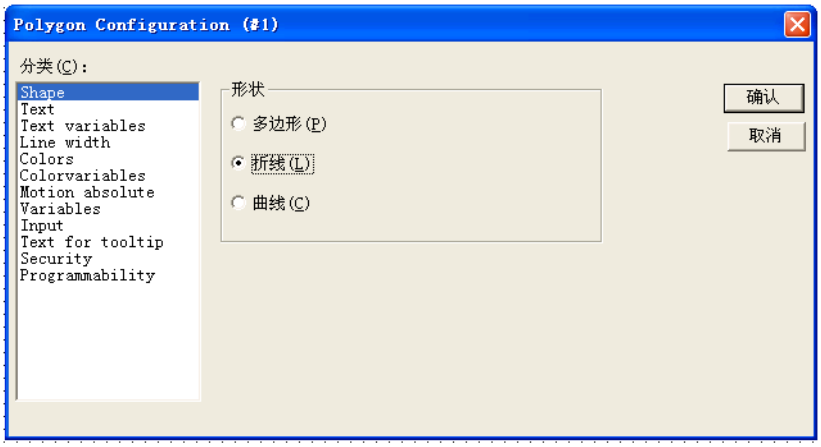


图 11-5-1 不规则视图对象属性配置对话框

11.5.2 文本

在视图对象中可以添加文本，用文本 Text 属性来设置，如图 11-5-2 所示。

➢ 内容

在“内容”文本框中输入文本，按<Ctrl>+<Enter>组合键换行。

➢ 水平

在“水平”选项中设置文本在视图对象中的左、中、右位置。

➢ 垂直

在“垂直”选项中设置文本在视图对象中的上、中、下位置。

点击“字体”或“标准字体”按钮，可以设置文本的字体。

图 11-5-2 文本设置

在标准 C 库中，常用的占位符有%s（字符格式）、%f（浮点型格式）、%d（整型格式）和%x（16 进制整型）等，其含义如表 11-5-1 所示。

表 11-5-1 占位符与含义

占位符	含义
% a	星期缩写
% A	星期全写
% b	月缩写
% B	月全称
% c	日期时间
% d	一月中的天数（01-31）
% H	24 小时格式（00-23）
% I	12 小时格式（01-12）
% j	一年中的天数（001-366）
%m	月（01-12）
%M	分钟（00-59）
%p	12 小时 A.M/P.M 格式
%S	秒（00-59）
%U	一年的第几个星期(00-53)，周日为星期第一天
%w	星期（0-6，周日是 0）
%W	一年的第几个星期(00-53)，周一为星期第一天
%x	日期
%X	时间
%y	不含世纪的年（00-99）
%Y	含世纪的年
%z	时区名，如：中国标准时间
%Z	
%%	百分号

✧ 举例

如果在文本内容中填入%2.5f mm，则在程序运行时会显示 32.8889 mm。

✧ 举例

在视图中设置如图 11-5-3 所示的格式。程序运行结果如图 11-5-4 所示。显示当前时间格式：中国标准时间-年-月-日 时：分：秒

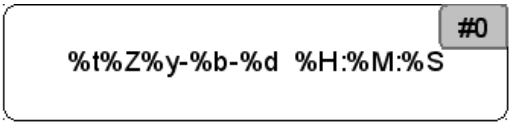


图 11-5-3 时间占位符应用

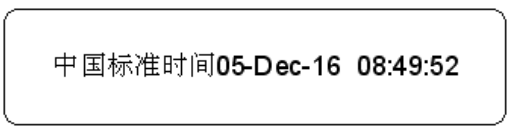


图 11-5-4 程序运行结果

11.5.3 线宽

线宽 Line width 属性用来定义视图对象线条的宽度。图 11-5-5 所示为线宽设置对话框，共有 5 种线宽可以直接选择。选中“高级”选项可以输入线宽。在“线宽变量”中可以输入控制线宽的工程变量，使线宽产生动画效果。输入工程变量时可以使用 F2 功能键。在线模式下，动态线宽属性会覆盖静态线宽属性。

图 11-5-5 线宽设置

11.5.4 颜色

颜色 Colors 属性用来定义视图对象的颜色和报警的颜色。图 11-5-6 所示为视图对象颜色设置对话框，可以分别设置视图对象的内部颜色和边框颜色。选中“无填充色”或“无边框色”可以创建透明的图形对象。视图对象的报警颜色可以用填充颜色或边框颜色来显示。在线模式下，动态颜色属性会覆盖静态颜色属性。

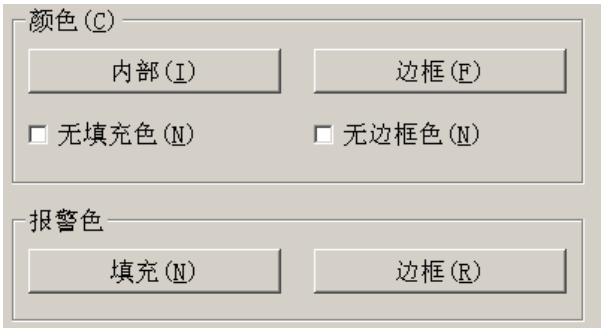


图 11-5-6 视图颜色设置

11.5.5 工具提示文本

工具提示文本 Text for tooltip 属性用来定义视图对象的工具提示文本。图 11-5-7 所示为视图对象工具提示文本设置对话框，可以在“内容”文本框中输入表示视图对象的文本。离线或在线模式下，当鼠标移动到该图形对象上时，会显示所输入的文本。在“内容”文本框中输入文本，可以按<Ctrl>+<Enter>组合键换行。

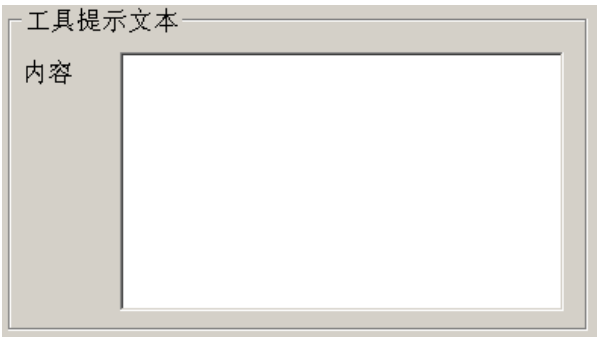


图 11-5-7 工具提示文本

11.5.6 安全属性

安全 Security 属性用来定义视图对象的访问权限。图 11-5-8 所示为视图对象安全设置对话框，可以定义用户对于视图对象的访问权限。用户组有 8 组，访问权限分为“无访问权限”、“只读权限”和“完全控制权限”等 3 种。在线模式下，具有不同的访问权限的用户组可以实现的操作不同。

- 具有“无访问权限”的用户组，不具有访问视图对象的权限，视图对象不可见。
- 具有“只读权限”的用户组，视图对象可见，但不可操作，即无输入操作权限。
- 具有“完全控制权限”的用户组，可以对视图对象进行完全控制。

如果需要将此安全权限的配置用于其他视图对象中，选中“适用所有视图元素”即可。

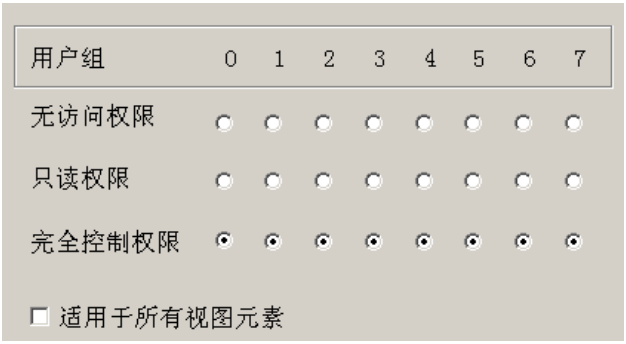


图 11-5-8 操作权限设置

11.5.7 位图属性

在视图中可以插入位图。位图 Bitmap 属性用来设置插入到视图中位图的参数。图 11-5-9 所示为位图设置对话框。

- 位图

点击“位图”文本框后的按钮可以选择所需要的位图文件。

选中“背景透明”，可以创建透明的图形对象。

- 框架

在“框架”选项中可以设置位图框架的属性。

“各项异性”表示在改变位图的尺寸时，位图的高度和宽度可按任意比例进行拉伸。

“各项同性”表示在改变位图的尺寸时，位图的高度和宽度比例保持不变。

“固定”表示在改变位图的尺寸时，位图本身的高度和宽度大小固定不变。只是可根据需求，改变位图的显示大小。

在选中“固定”选项前提下，再选中“绘图”表示显示位图对象的框架，否则，不显示位图对象的框架；如果再选中“剪切”表示当位图对象的框架小于位图时，只显示框架内的位图，而不显示框架外的位图。如果想显示整个位图大小，可拉伸框架的大小。

“颜色”和“报警色”可以分别设置位图框架的颜色和报警色。

图 11-5-9 位图设置

11.5.8 视图属性

在视图中可以插入其它视图。视图 Visualization 属性用来设置插入到视图中其它视图的参数。图 11-5-10 所示为视图设置对话框。

- 视图

点击“视图”文本框后的按钮可以选择所需要的视图文件。

点击“占位符”按钮可以设置视图的替换占位符。

- 框架

在“框架”选项中可以设置视图框架的属性。

“各项异性”表示在改变视图的尺寸时，视图的高度和宽度可按任意比例进行拉伸。

“各项同性”表示在改变视图的尺寸时，视图的高度和宽度比例保持不变。

“固定”表示在改变视图的尺寸时，视图的本身的高度和宽度大小是固定不变。可根据需求，改变视图的显示大小。

在选中“固定”选项前提下，再选中“绘图”表示显示视图对象的框架，否则，不显示视图对象的框架；如果再选中“剪切”表示当视图对象的框架小于视图时，只显示框架内的视图，而不显示框架外的视图。如果想显示整个视图大小，可拉伸框架的大小。

“颜色”和“报警色”可以分别设置视图框架的颜色和报警色。

图 11-5-10 视图设置

11.5.9 组框架属性

在视图中可以插入 WMF 图元文件格式的图形文件。WMF 图元文件的组框架 Group 属性用来设置插入到视图中 WMF 图元文件的参数。图 11-5-11 所示为 WMF 图元文件组框架 Group 设置对话框。

在“框架”选项中可以设置视图框架的属性。

选中“绘图”表示显示视图对象的框架。

选中“各向同性的”表示在改变视图的尺寸时，视图的高度和宽度比例保持不变。

选中“剪切”表示当视图对象的框架小于视图时，只显示框架内的视图，而不显示框架外的视图。

“颜色”和“报警颜色”可以分别设置视图框架的颜色和报警色。

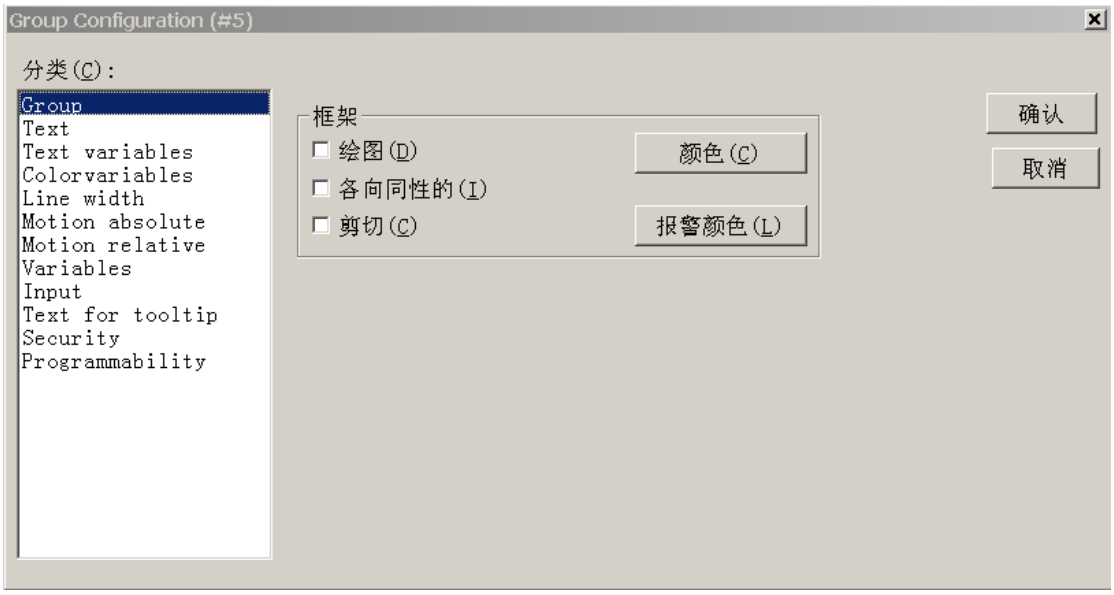


图 11-5-11 图元文件组框架属性设置

11.5.10 角度

角度 Angle 属性用来定义饼图对象的角度。双击饼图对象，弹出饼图对象属性配置对话框，如图 11-5-12 所示。点击角度 Angle 属性，在“开始角度”和“结束角度”文本框中分别输入饼图对象的开始角度和结束角度，则将会以顺时针方向画出所需要的饼图。选中“只进行段显示”，则饼图对象不显示夹角，只显示弧段。

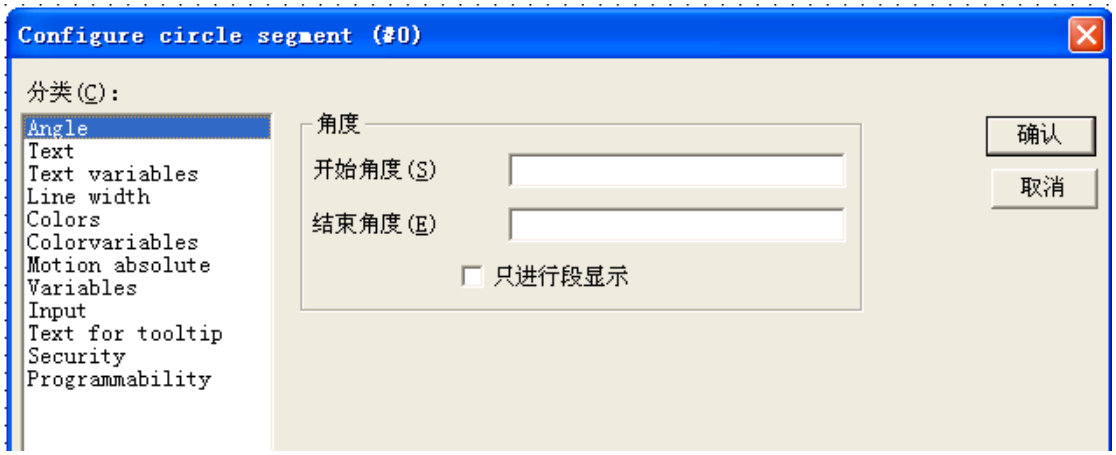


图 11-5-12 饼图对象属性配置对话框

✧ 举例

声明变量

```
PROGRAM PLC_PRG
VAR
    angle_start: REAL := 0;
    angle_end: REAL := 90;
END_VAR
```

视图运行结果如图 11-5-13 所示。

图 11-5-13 饼图示例

11.6 视图编程

11.6.1 编程属性

视图对象的属性不仅可以进行静态参数的设置或常规工程变量的定义，而且可以进行结构变量的定义，这些结构变量专门用于视图对象的编程。为此，在库 SysLibVisu.lib 中定义了结构体 VisualObjectType，其成员分量可以用来定义绝大多数图形对象的属性。为了避免图形对象属性的多重定义，常规工程变量的值将覆盖结构变量的值，而且二者都覆盖静态属性的定义。

为了使用结构变量来配置图形对象的属性，可以打开图形对象属性配置对话框，选择编程属性 Programmability，如图 11-6-1 所示。选中“对象名字”选择框，在其后的文本框中输入结构变量的名称，则会自动声明 VisualObjectType 结构体类型的结构变量，该结构体 VisualObjectType 在库 SysLibVisu.lib 中已经被定义。这样声明的结构变量是全局变量，其声明是不可见的隐含声明，前提是必须在库管理器中添加 SysLibVisu.lib 库。

重新编译工程后，新定义的结构变量将出现在工程中。

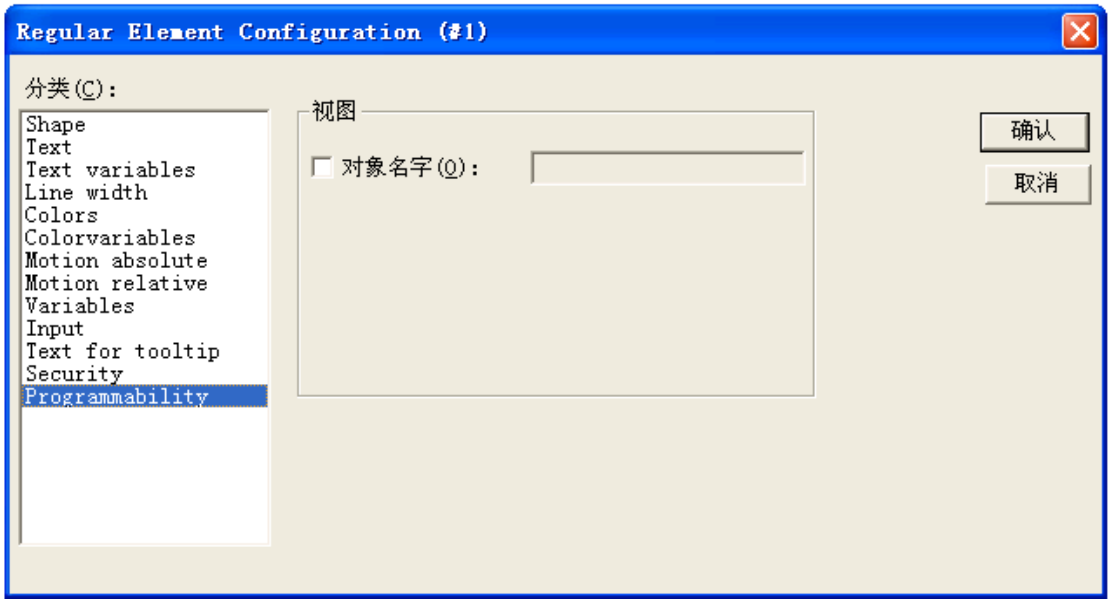


图 11-6-1 编程属性的设置

11.6.2 视图库

视图库 SysLibVisu.lib 中结构体 VisualObjectType 的定义如下所示。

```
TYPE VisualObjectType
STRUCT
    (* Absolute movement *)
    nXOffset:INT;
    nYOffset:INT;
    nScale:INT;
    nAngle:INT;
    (* Variables *)
    bInvisible:BOOL;
    stTextDisplay:STRING;
    bToggleColor:BOOL;
    bInputDisabled: BOOL;
    stTooltipDisplay:STRING;
    (* Text and font *)
    dwTextFlags:DWORD;
    dwTextColor:DWORD;
```

```
    nFontHeight:INT;
    dwFontFlags:DWORD;
    stFontName:STRING;
    (* Line *)
    nLineWidth:INT;
    (* Color *)
    dwFillColor: DWORD;
    dwFillColorAlarm: DWORD;
    dwFrameColor: DWORD;
    dwFrameColorAlarm: DWORD;
    dwFillFlags: DWORD;
    dwFrameFlags: DWORD;
    (* Relative movement *)
    nLeft:INT;
    nTop:INT;
    nRight:INT;
    nBottom:INT;
END_STRUCT
END_TYPE
```

结构体 VisualObjectType 中的成员分量的数据类型及功能如表 11-6-1 所示。成员分量的前缀字母表示数据类型，均为小写字母，其含义分别为：

n 表示 INT，dw 表示 DWORD，b 表示 BOOL，st 表示 STRING。

表 11-6-1 成员分量的数据类型及功能

成员分量 及数据类型	功能	示例（结构变量为vis1）	对应的图形对象属性
nXOffset : INT;	X方向位移	vis1.nXOffset:=val1; （沿 X 方向移动到 X=val1 处）	Motionabsolute: X-Offset
nYOffset : INT;	Y方向位移	vis1.nYOffset:=val2; （沿 Y 方向移动到 Y=val2 处）	Motionabsolute: Y-Offset
nScale : INT;	改变大小	vis1.nScale:=plc_prg.scale_var; （取值 plc_prg.scale_var）	Motion absolute: Scale
nAngle : INT;	旋转角度	vis1.anglevar:=15; （顺时针旋转 15 度）	Motion absolute: angle
bInvisible : BOOL;	可见/不可见	vis1.visible:=TRUE; （图形对象可见）	Color: No color inside+ No frame color Colorvariables: Fillcolor+Framecolor
stTextDisplay: STRING;	显示文本	vis1.TextDisplay:='ON / OFF'; （显示文本ON / OFF）	Text: Content
bToggleColor : BOOL;	颜色在 TRUE 与 FALSE 之间切换	vis1.bToggleColor:=alarm_var; （当变量 alarm_var 为 TRUE 时，分别显示 dwFillColorAlarm 和 dwFrameColorAlarm 定义的颜色）	Input: Toggle variable Variables: Change color

bInputDisabled : BOOL;	取值 FALSE，忽略配置 Input	vis1.bInputDisabled:=FALSE; （此图形对象不能输入值）	Variables: Input Disable
stTooltipDisplay : STRING;	工具提示文本	vis1.stTooltipDisplay:='Switch for '; （显示工具提示文本Switch for）	Text for Tooltip: Content:
dwTextFlags : DWORD;	文本位置：: 1 左对齐 2 右对齐 4 水平居中 8 顶对齐 10 底对齐 20 垂直居中 （取值可以相加）	vis1.dwTextFlags:=24; （文本居中，4+20）	Text: Horizontal and Vertical options Textvariables: Textflags
dwTextColor : DWORD;	文本颜色（颜色的定义见表后示例）	vis1.dwTextColor :=16#00FF0000; （蓝色）	Textvariables: Textcolor
nFontHeight : INT;	字体高度，取值范围 10-96 像素	vis1.nFontHeight:=16; （字体高度为 16 像素）	Textvariables: Font heigth
dwFontFlags : DWORD;	字体： 1 斜体字 2 粗体字 4 下划线 8 取消 （取值可以相加）	vis1.dwFontFlags:=10; （显示粗体字并取消，2+8）	Textvariables: Fontflags
stFontName : STRING;	改变字体	vis1.stFontName:='Arial'; （使用 Arial 字体）	Textvariables: Fontname
nLineWidth : INT;	框架的线宽（像素）	vis1.nLWidth:=3; （框架线宽为 3 像素）	Line width
dwFillColor : DWORD;	填充颜色（颜色的定义见表后示例）	vis1.dwFillColor":=16#00FF0000; （填充蓝色）	Color: Color Inside Colorvariables: Fill color
dwFillColorAlarm : DWORD;	当 bToggleColor 为 TRUE 时，填充颜色（颜色的定义见表后示例）	vis1.dwFillColorAlarm:=16#00808080; （当 bToggleColor 为 TRUE 时，填充灰色）	Color: Alarm color Inside Colorvariables: Fill color alarm
dwFrameColor : DWORD;	框架颜色（颜色的定义见表后示例）	vis1.dwFrameColor:=16#00FF0000; （框架为蓝色）	Color: Color Frame Colorvariables: Frame color
dwFrameColorAlarm : DWORD;	当 bFrameColor 为 TRUE 时，填充颜色（颜色的定义见表后示例）	vis1.dwFrameColorAlarm:=16#00808080; （当 bFrameColor 为 TRUE 时，填充灰色）	Color: Alarm color Frame Colorvariables: Frame color alarm
dwFillFlags: DWORD;	显示颜色： 0 显示颜色 >0 忽略颜色设置	vis1.dwFillFlags:=1; （显示颜色）	Color: No color inside + No frame color Colorvariables: Fillflags
dwFrameFlags : DWORD;	显示框架： 0 全显示 1 虚线(---) 2 点( ) 3 点划线(_._._) 4 双点划线(_.._..) 8 不显示	vis1.FrameFlags:=1; （框架显示虚线）	Colorvariables: Frameflags

颜色取值定义：vis1.dwFillColor := 16#00FF00FF;

颜色取值以八位 16 进制数 16#00FF00FF 表示，包括蓝、绿和红等三种基本颜色。每种颜色以两位（0-255）表示，上述三种颜色依次占据后六位。在符号 16#后面的两位 00 必须保留，以满足 DWORD 数据类型的语法规则。对于每种颜色，可以取值 0-255，如下所示：

16#	00	FF	00	FF
关键字	保留位	蓝色	绿色	红色

举例（颜色取值示例）

```
PROGRAM PLC_PRG
VAR
    n:INT:=0;
    bMod:BOOL:=TRUE;
END_VAR
(* Blinking element *)
n:=n+1;
bMod:= (n MOD 20) > 10;
IF bMod THEN
    blinker.nFillColor := 16#00808080; (*  灰色  *)
ELSE
    blinker.nFillColor := 16#00FF0000; (*  蓝色  *)
END_IF
```

11.7 视图动态属性

所谓视图的动态属性是指描述图形对象动画效果的参数，即用工程变量来控制图形对象产生动画效果。

11.7.1 文本变量

文本变量 Text variables 用来定义文本的动态属性。图 11-7-1 所示为文本变量设置对话框，在文本框中输入相应的变量，即可实现动画效果。输入变量时可以使用 F2 功能键。

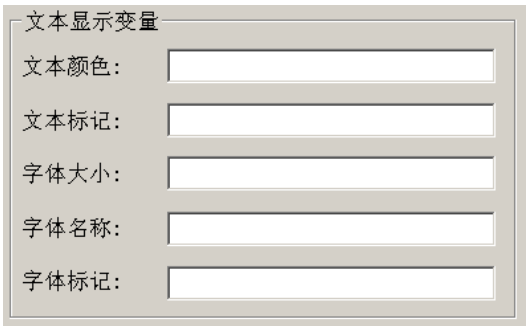


图 11-7-1 文本变量设置

11.7.2 颜色变量

颜色变量 Color variables 用来定义颜色的动态属性。图 11-7-2 所示为颜色变量设置对话框，在文本框中输入相应的变量，即可实现动画效果。输入变量时可以使用 F2 功能键。

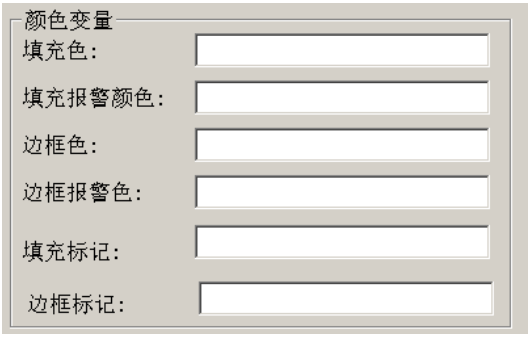


图 11-7-2 颜色变量设置

11.7.3 绝对运动

绝对运动 Motionabsolute 用来定义绝对运动的动态属性。图 11-7-3 所示为绝对运动设置对话框，在文本框中输入相应的变量，即可实现动画效果。输入变量时可以使用 F2 功能键。

- X 位移：设置变量的水平移动一定距离的变量。
- Y 位移：设置变量的上下垂直移动一定距离的变量。
- 比例：设置比例系数，改变视图大小。如果要将新视图对象变为原视图对象的 1 倍，则比例取值为 1000。如果要将新视图对象变为原视图对象的 2 倍，则比例取值为 2000。
- 角度：视图对象的旋转角度。正值表示顺时针旋转，负值表示逆时针旋转。对于矩形，相当于视图对象以基点为原点相对平移一定角度，而不进行旋转。对于除矩形外的视图对象，例如多边形和椭圆等，都是以基点为原点进行整体旋转，也就是说，多边形等视图对象的每一点都在旋转。

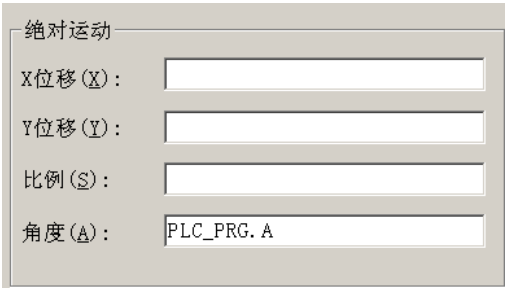


图 11-7-3 绝对运动设置

11.7.4 相对运动

相对运动 Motion relative 用来定义相对运动的动态属性。图 11-7-4 所示为相对运动设置对话框，在文本框中输入相应的变量，即可实现动画效果。输入变量时可以使用 F2 功能键。对于坐标轴，X 方向“右”为正，Y 方向“上”为正，即变量取值为正时，视图对象的边缘向正向移动一定距离，否则，向负向移动。

- 左边缘：视图对象左边缘移动一定距离。
- 上边缘：视图对象上边缘移动一定距离。
- 右边缘：视图对象右边缘移动一定距离。
- 下边缘：视图对象下边缘移动一定距离。

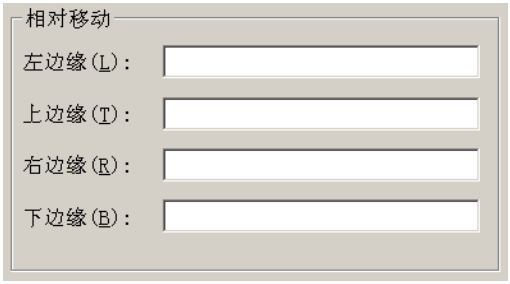


图 11-7-4 相对运动设置

11.7.5 变量

变量 Variables 用来定义图形对象的其它动态属性。图 11-7-5 所示为变量设置对话框，在文本框中输入相应的变量，即可实现动画效果。输入变量时可以使用 F2 功能键。

- 在“不可视变量”中输入逻辑变量，其值决定图形对象的可视属性。如果变量值为 FALSE，则图形对象是可见的。如果变量值为 TRUE，则图形对象是不可见的。
- 在“改变颜色变量”中输入逻辑变量，其值决定图形对象的颜色。如果变量值为 FALSE，则图形对象显示缺省颜色。如果变量值为 TRUE，则图形对象显示报警颜色。
- 在“文本显示”字段可输入显示值。如果已经在“文本”的“内容”中输入了“%s”，则“在线模式”时，“文本显示”的值会代替“%s”显示在文本区域。

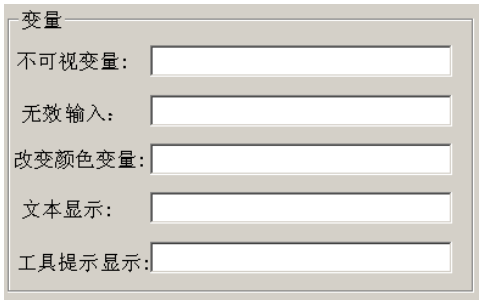


图 11-7-5 变量设置

11.7.6 输入

输入 Input 用来定义图形对象的输入动态属性。图 11-7-6 所示为输入设置对话框，在文本框中输入相应的变量，即可实现动画效果。输入变量时可以使用 F2 功能键。

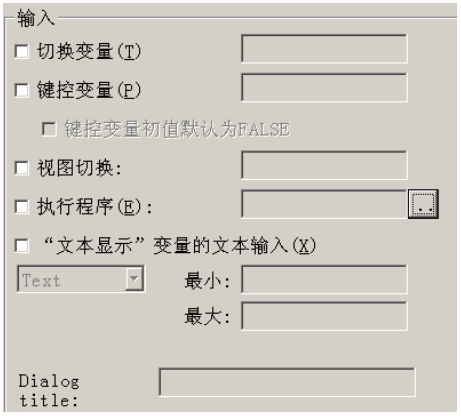


图 11-7-6 输入设置

- 切换变量：选择“切换变量”，在线模式时，在对象上每次点击都会改变位于输入字段的变量值。鼠标每次点击，逻辑变量的值都会从 TRUE 变成 FALSE，或者相反。
- 键控变量：选项“键控变量”允许在线模式下改变逻辑变量的值。这个逻辑变量位于输入字段。把鼠标光标放在对象上，按下鼠标键不要松开，例如变量值从 TRUE 改为 FALSE。当释放鼠标时，变量的值返回到初始值 TRUE。
- 视图切换：选择字段“视图切换”，允许在随后字段里输入同一工程的视图对象名。在线模式时，在视图窗口中，使用鼠标点击改变到已输入的对象。如果输入 STRING 类型的程序变量，例如是 PLC_PRG.xxx，而不是一个视图对象，那么这个变量可以用来定义视图对象名，例如 visu1。鼠标点击时，系统将显示这个视图对象，例如 xxx := visu1。
- 执行程序：选择“执行程序”，在输入字段中输入任意的可执行程序。在线模式下，鼠标点击对象，执行该可执行程序。
- “文本显示”变量的文本输入：选择“‘文本显示’变量的文本输入”，在线模式下，通过视图对象分配给变量一个值。点击对象产生一个编辑框架，给变量键入新值，按下<Enter>键接受这个值。

11.8 表格

在编辑视图时，可以在视图中插入表格，如图 11-8-1 所示。

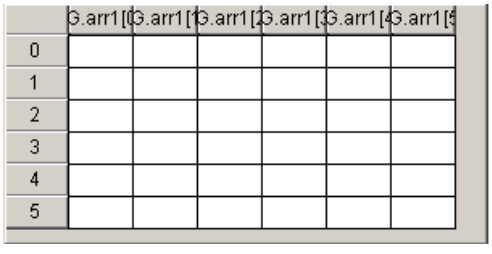


图 11-8-1 插入表格

双击表格对象，弹出表格对象属性配置对话框，如图 11-8-2 所示，可以设置表格 Table、列 Columns、行 Rows、选择 Selection、工具提示文本 Text for tooltip 和安全 Security 等属性。输入变量时可以使用 F2 功能键。

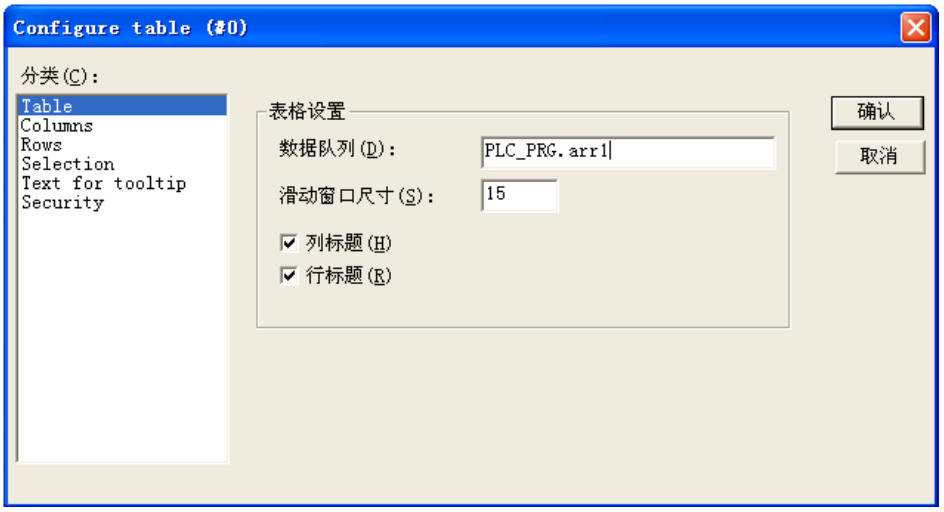


图 11-8-2 表格配置对话框

11.9 趋势图

在编辑视图时，可以在视图中插入趋势图，如图 11-9-1 所示。

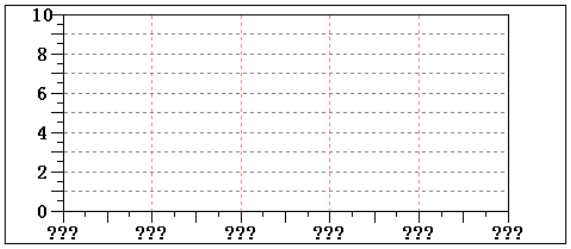


图 11-9-1 插入趋势图

双击趋势图对象，弹出趋势图对象属性配置对话框，如图 11-9-2 所示，可以设置趋势图 Trend、颜色 Colors、工具提示文本 Text for tooltip 和安全 Security 等属性。输入变量时可以使用 F2 功能键。

图 11-9-2 趋势图配置对话框

11.10报警表

在编辑视图时，可以在视图中插入报警表，如图 11-10-1 所示。

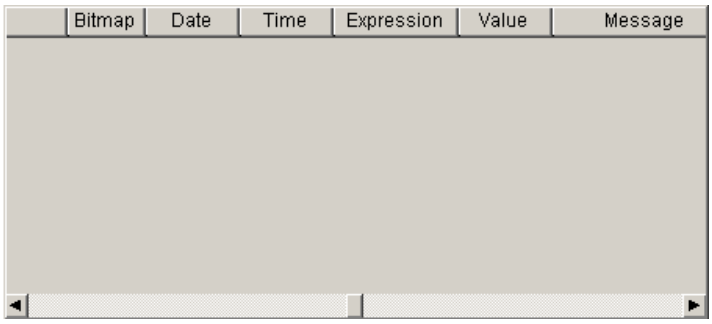


图 11-10-1 插入报警表

双击报警表对象，弹出报警表对象属性配置对话框，如图 11-10-2 所示，可以设置报警表 Alarm table、列 Columns、排序 Settings for sorting、选择 Selection settings、工具提示文本 Text for tooltip 和安全 Security 等属性。输入变量时可以使用 F2 功能键。

图 11-10-2 报警表配置对话框

11.11ActiveX 控件

在编辑视图时，可以在视图中插入 ActiveX 控件，如图 11-11-1 所示。

图 11-11-1 插入 ActiveX 控件

双击 ActiveX 控件对象，弹出 ActiveX 控件对象属性配置对话框，如图 11-11-2 所示，可以设置控件 Control、方法调用 Method calls 和显示 Display 等属性。输入变量时可以使用 F2 功能键。

图 11-11-2 ActiveX 控件配置对话框

11.12视图举例

下面以一个小程序为例来介绍视图的应用。此程序可以实现两个信号灯的交替闪烁。程序编写如图 11-12-1 所示。

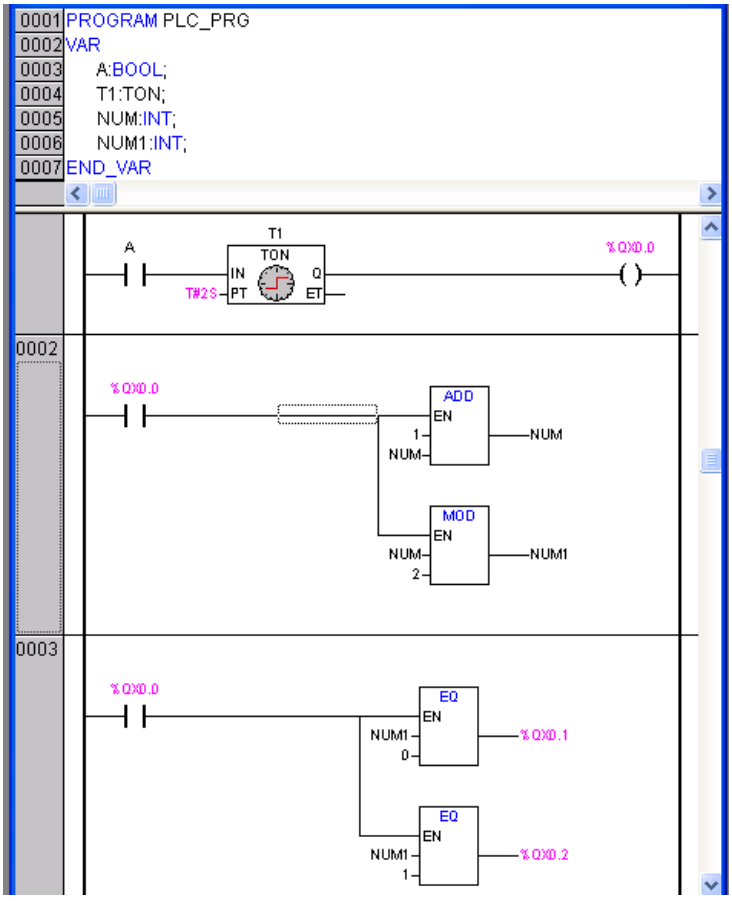


图 11-12-1 视图程序编写

为了能够进一步看清楚信号灯的闪烁情况，可以定义视图对象以不同颜色来显示信号灯的亮、灭情况。用绿色代表灯亮，红色代表灯灭。

首先创建视图 A。在视图 A 中，用两个圆分别表示两个信号灯，用两个矩形分别表示两个信号灯的名称“Q.1”和“Q.2”，如图 11-12-2 所示。

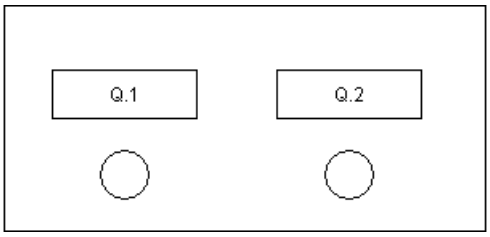


图 11-12-2 视图绘制

选中 Q.1 信号灯，并双击，则会出现“Regular Element Configuration”对话框。选择“Colors”设置“颜色”栏中的“内部”颜色为绿色，单击“确定”，如图 11-12-3 所示。

同样，设置“报警色”栏中的“填充”颜色为红色。Q.2 信号灯的颜色设置方式相同。

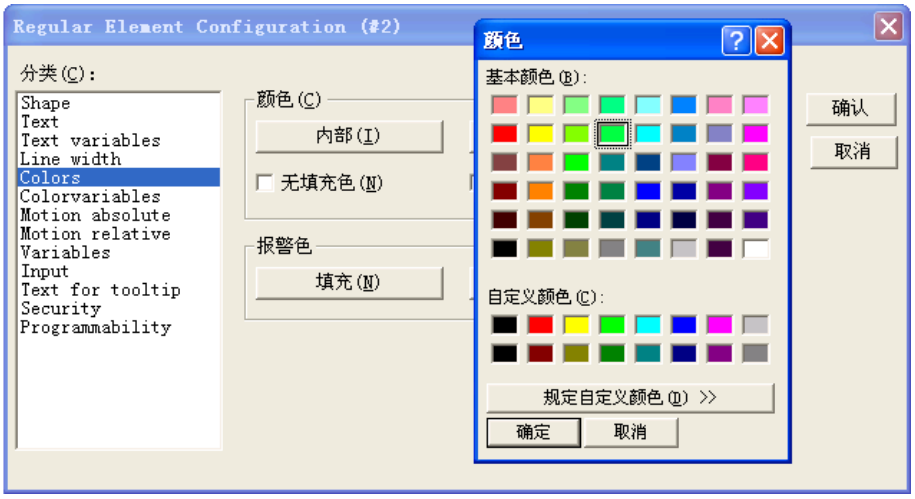


图 11-12-3 颜色设置

在“分类”栏中，选择“Variables”，在“改变颜色变量”选项中，填入信号灯“Q.1”，单击“确认”，如图 11-12-4 所示。同理，在“Q.2”的“改变颜色”项中填入“Q.2”。

图 11-12-4 变量设置

至此，本程序的视图已经配置完毕。程序的运行情况如图 11-12-5 所示。Q.1 和 Q.2 两信号灯实现红、绿颜色的交替闪烁。

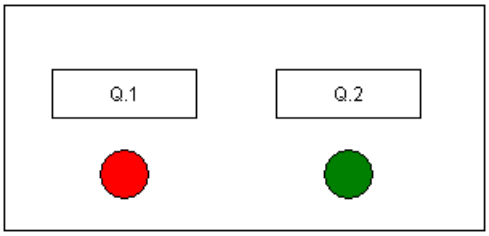


图 11-12-5 程序运行情况

附录

模块存储空间

下表所列举版本的模块，其程序存储空间为 28KB，在工程目标配置时，需选择“HOLLiAS-LEC G3 CPU”。

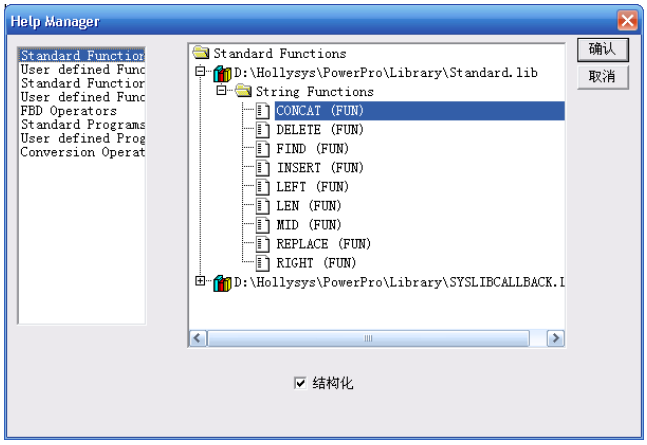
模块型号	模块版本
LM3104	A01 、A02 、A03
LM3105	A01 、A02、 A03
LM3106	A01 、A05 、B06 、B07、 B08
LM3106A	A01、 A02、 A03 、A04
LM3107	A01 、A05 、B06、 B07 、B08

除此之外，其余版本模块的程序存储空间为 120KB，在目标配置时，选择“HOLLiAS-LEC G3 CPU Extend”。

PowerPro 输入提示功能

对于不熟悉标准编程语言的用户，PowerPro 软件提供了有效的输入提示功能，帮助用户完成标准函数、功能块、运算符、操作数以及变量等的输入。使用主菜单中的“编辑”/“提示输入”命令，或者使用快捷功能键 F2，可以打开帮助管理器窗口，如图附录-1 所示。根据当前位置和选中对象的不同，可以列出不同的输入选项。

Standard Functions：标准函数，来自函数库中的函数；
User defined Functions：用户自定义函数，由用户编写生成的函数；
Standard Function Blocks：标准功能块，来自函数库中的功能块；
User defined Function Blocks：用户自定义功能块，由用户编写生成的功能块；
FBD Operators：PowerPro 支持的标准 IEC 运算符；
Standard Programs：标准程序；
User defined Programs：用户自定义程序，由用户编写生成的程序；
Conversion Operators：转换运算符，用于不同类型变量之间的数据类型转换。



图附录-1 输入提示帮助

PowerPro 键盘命令

一般功能	
在程序的声明区和指令区之间移动	<F6>
在对象管理器、对象和消息窗口之间移动	<Alt>+<F6>
移动到下一个已经打开的编辑器窗口	<Ctrl>+<F6>
移动到上一个已经打开的编辑器窗口	<Ctrl>+<Shift>+<F6>
弹出右键菜单	<Shift>+<F10>
变量声明的快捷方式	<Ctrl>+<Enter>
从消息窗口返回到编辑器的初始位置	<Enter>
打开和关闭多层变量	<Enter>
打开或关闭文件夹	<Enter>
在对象管理器和库管理器的选项卡之间转换	<Arrow Keys>
移动到下一个字段	<Tab>
帮助	<F1>
一般命令	
"文件" /"打开"	<Ctrl>+<O>
"文件" /"保存"	<Ctrl>+<S>
"文件" /"打印"	<Ctrl>+<P>
"文件" /"退出"	<Alt>+<F4>
"编辑" /"恢复"	<Ctrl>+<Z>
"编辑" /"重复"	<Ctrl>+<Y>
"编辑" /"剪切"	<Ctrl>+<X>
"编辑" /"复制"	<Ctrl>+<C>
"编辑" /"粘贴"	<Ctrl>+<V>
"编辑" /"删除"	<Del>
"编辑" /"查找下一个"	<F3>
"编辑" /"替换"	<Ctrl>+<H>
"编辑" /"提示输入"	<F2>
"编辑" /"自动定义"	<Shift>+<F2>
"编辑" /"后错误"	<F4>
"编辑" /"前错误"	<Shift>+<F4>
"工程" /"全部编译"	<F11>
"工程" /"删除对象"	<Del>
"工程" /"添加"	<Ins>
"工程" /"重命名"	<Spacebar>
"工程" /"打开"	<Enter>
"在线" /"登录"	<Alt>+<F8>
"在线" /"退出"	<Ctrl>+<F8>
"在线" /"运行"	<F5>

"在线" /"停止"	<Shift>+<F8>
"在线" /"断点"	<F9>
"在线" /"跳出"	<F10>
"在线" /"跳进"	<F8>
"在线" /"单循环"	<Ctrl>+<F5>
"在线"/"输入值"	<Ctrl>+<F7>
"在线" /"强制值"	<F7>
"在线" /"解除强制"	<Shift>+<F7>
"在线" /"输入/强制对话框"	<Ctrl>+<Shift>+<F7>
"窗口" /"信息"	<Shift>+<Esc>
LD编辑器命令	
“插入”/后节"	<Shift>+<T>
“插入”/ "触点"	<Ctrl>+<K>
“插入”/ "并联触点"	<Ctrl>+<R>
“插入”/ "功能块"	<Ctrl>+<B>
“插入”/ "输出"	<Ctrl>+<L>
"其它" /"下方并联"	<Ctrl>+<U>
"其它" /"反向"	<Ctrl>+<N>
FBD编辑器命令	
“插入”/ "后节"	<Shift>+<T>
“插入”/ "输入"	<Ctrl>+<U>
“插入”/ "运算符"	<Ctrl>+<B>
“插入”/ "赋值"	<Ctrl>+<A>
“插入”/ "跳转"	<Ctrl>+<L>
“插入”/ "返回"	<Ctrl>+<R>
"其它" /"反向"	<Ctrl>+<N>
"其它" /"放大"	<Alt>+<Enter>
SFC编辑器命令	
“插入”/ "前步转移"	<Ctrl>+<T>
“插入”/ "后步转移"	<Ctrl>+<E>
“插入”/ "右选择分支"	<Ctrl>+<A>
“插入”/ "右并行分支"	<Ctrl>+<L>
“插入”/ "跳转"	<Ctrl>+<U>
"其它" /"编辑动作/转移"	<Alt>+<Enter>

和利时集团
地址：北京经济技术开发区地盛中路2号院（100176）
电话：010-5898 1588
传真：010-5898 1558
产品咨询热线：4008-111-999
技术支持邮箱：PLC@hollysys.com
主页：www.hollysys.com