UWinTech Pro控制工程应用软件平台

控制组态手册 V1.0

浙江大学工业自动化国家工程研究中心

杭州优稳自动化系统有限公司

公 司 简 介

浙江大学工业自动化国家工程研究中心是建立在控制理论与控制工程、检测技术与自动化装置国家重点学科基础上的集博士学位授予点、博士后流动站、国家重点学科、国家重点实验室、国家工程研究中心为一体的全面发展的教学、研究群体。其主要任务是：以大型工业过程为背景，开发对国民经济有重大影响的关键性技术，使其形成标准化、系列化、商品化的成套技术和成套设备；开发集散控制系统，现场总线自动化仪表及装置；开发为综合自动化系统服务的工程实时数据库技术、先进控制与过程优化技术及其成套软件；向企业集团和全国大中型企业进行技术转移与扩散。

在孙优贤院士、王文海博士等学术带头人的带领下，研究开发团队长期专注于集散控制系统、可编程控制器、智能仪表、控制工程应用软件平台、大型装备自动化系统的研究开发与产业化；承担国家计委工业自动化高技术产业化重大专项 3 项，具有扎实的科研积累与丰富的技术经验；形成了独具特色的具有自主知识产权的计算机控制系统技术体系，在可靠性设计技术、数据 I/0 技术、实时控制技术、实时数据库技术、软件平台技术等关键核心技术上有 11 项重大创新与技术突破。近 5 年，在工业自动化领域，作为第一、第二完成人获国家科技进步二等奖 1 项，省部级一等奖 3 项，二等奖 1 项。取得软件著作权 11 项，专利 10 项。浙江大学工业自动化国家工程研究中心，总结其在工业自动化领域数十年的理论研究、技术积累与科研成果，研究开发了新一代主控系统——UW500 集散控制系统。

杭州优稳自动化系统有限公司，在浙江大学控制工程国家实验室大楼建立联合技术中心，具有领先的创新意识和丰富的技术资源，负责最新一代控制技术的产业化推广与服务。业务与产品涉及：智能仪表、可编程控制器、集散控制系统、安全控制系统、控制工程应用软件平台的研究开发、生产制造与销售服务。公司信奉“求实、创新、敬业、专业”的价值观；坚持“人才为根本、市场为先导、质量为生命、共赢为目的” 的经营理念；经过科技创新与市场拓展的磨练，形成了一支结构合理、经验丰富、团结奋进、敢于拼搏、勇于创新的企业管理、研究开发和工程服务的人才队伍，研究开发人员 50%以上具有硕士或博士学位。


国家科学技术进步奖
证 书
为表彰国家科学技术进步奖获得者，特颁发此证书。
项目名称：全集成新一代工业自动化系统
奖励等级：二等
获 奖 者：浙江大学
证书号：2006-J-220-2-02-001


浙江省科学技术奖
证 书
为表彰浙江省科学技术奖获得者，特颁发此证书。
项目名称：全集成新一代主控系统
奖励等级：一 等
获 奖 者：浙江大学（第一完成单位）
王文海、孙优贤、张龙强
证书号：0501007-1


浙江省科学技术奖
证 书
为表彰浙江省科学技术奖获得者，特颁发此证书。
项目名称：大型造纸厂生产过程建模、控制与优化
奖励等级：一 等
获 奖 者：浙江大学（第一完成单位）
孙优贤、王文海、皮道映
林长松、杨春节、周金锁
黄锦华、于 玲、熊伯良
王 慧、李 平、张成江
王建新
证书号：0401006-01

版权 ProV1.0-130830

声明

UWin®、UWNTEK 优稳特®的字样及徽标均为杭州优稳自动化系统有限公司的注册商标。

本手册内容与所叙述的系统硬件、软件相符，因为差错难免，我们无法保证完全一致。但我们会对本手册内容进行经常性检查，并对错误或升级部分予以必要更正，欢迎用户提出宝贵意见。

本手册内容与撰写手册时的软件版本（UWinTech Pro 控制工程应用软件平台 ProV1.0）相一致。

本手册内容如有更改，恕不另行通知。

标注

手册中可能会有三种警示读者的方式：

提示 ：向读者提供解决当前问题的另一种方法或系统对某个操作的响应。

注意 ：如果不遵守规定可能会导致系统工作异常。

提示 ：如果不遵守规定可能会导致设备损坏或人身伤害。

适用对象

本手册的阅读对象是使用 UWinTech Pro 控制工程应用软件平台的工程技术人员、系统集成商、控制工程师以及 UW500 集散控制系统等产品的最终用户。掌握本手册的内容能够解决系统工程应用中遇到的大部分问题。

其他帮助

有关 UWinTech Pro 控制工程应用软件平台产品的技术支持、产品培训和订货事宜，请与杭州优稳自动化系统有限公司销售中心及其产品分销商联系。

总　　机：400-007-0089
传　　真：0571-88371967
公司主页：http://www.uwntek.com
邮　　箱：cs@uwntek.com

目录

软件使用说明

1 软件概述

算法编辑器是 UWinTechPro 控制工程应用软件平台的控制组态软件，是编写控制算法的编辑器平台。算法编辑器遵循 IEC61131-3 标准，实现了 IEC61131-3 标准中的 5 种控制语言（FBD、LD、SFC、ST、IL），用户可用这 5 种语言中的一种或几种构建自己的控制算法。

1.1. 软件特点

算法编辑器具有如下特点：

1.1.1. 通用性

算法编辑器提供的概念与编程方法完全符合 IEC61131-3 标准。如果用户对其他控制系统的编程语言有一定的了解，可轻松在算法编辑器编辑环境下编写控制程序。

1.1.2. 易用性

软件界面与多数 Windows 平台应用软件风格相似，用户只要对 Windows 操作系统有一定的了解，即可轻松上手使用。

1.1.3. 方便性

连续仿真和单周期仿真调试功能使用户可以随时对控制算法进行模拟，验证其正确性。

1.1.4. 灵活性

丰富的功能块、函数块以及子程序给予用户充分的自由使用空间，在线帮助查询使用户无师自通。

1.2. 开发流程

1.2.1. 术语

在算法编辑器中所提到的工程、控制工程、程序三个术语说明如下：

- 工程：在 UWinTech Pro 控制工程应用软件平台的工程管理器中建立的工程，它包含实时数据库、监控画面以及控制算法。
- 控制工程：在算法编辑器中建立的工程，它包括若干个控制算法程序。
- 程序：在算法编辑器中具有独立运算周期的控制算法集。

1.2.2. 步骤

利用算法编辑器开发控制工程，必须按照如下步骤进行。用户也可以参考《UWinTech Pro 控制工程应用软件平台实时监控手册》的第二章“系统组态概述”。

1．使用算法编辑器以前，必须用 UWinTech Pro 控制工程应用软件平台的工程管理器新建一个工程（或指定一个已有的工程）并将该工程设为当前工程，此工程必须事先已经用实时数据库编辑器定义了记录点；

2．通过 UWinTech Pro 控制工程应用软件平台的工程管理器打开算法编辑器，选择要进行控制组态的控制站（此后编辑的控制算法将在此控制站上运行）。如果第一次针对该控制

站进行控制算法组态，则算法编辑器会默认建立一个新的控制工程；

3．在算法编辑器中利用五种编程语言（FBD、LD、SFC、ST 和 IL）中的一种或几种编辑控制算法；

4．对编辑好的控制算法工程进行编译，编译输出错误时应进行算法修改；

5．将编译好的控制算法工程进行离线仿真调试，观察输出结果。如果算法错误，则进行修改；

6．仿真调试通过后，下装至控制模件中。

1.3. 注意事项

1．控制算法工程下装至控制模件（控制站）运行以前，必须进行编译并保证编译无误，否则不能下装；

2．在算法编辑器中所建立的子程序与主程序数之和不得大于 512，其中主程序数不得大于 127。单个 FBD 程序中算法块数目不得大于 512 个，整个程序中 FBD 算法块不得大于 16384 个，否则在控制工程编译时会出现出错信息。

2 软件界面

2.1. 界面介绍

算法编辑器软件界面分为菜单栏、工具栏、导航栏、观察窗口、编辑窗口和输出窗口六个部分，如图 2-1 所示。

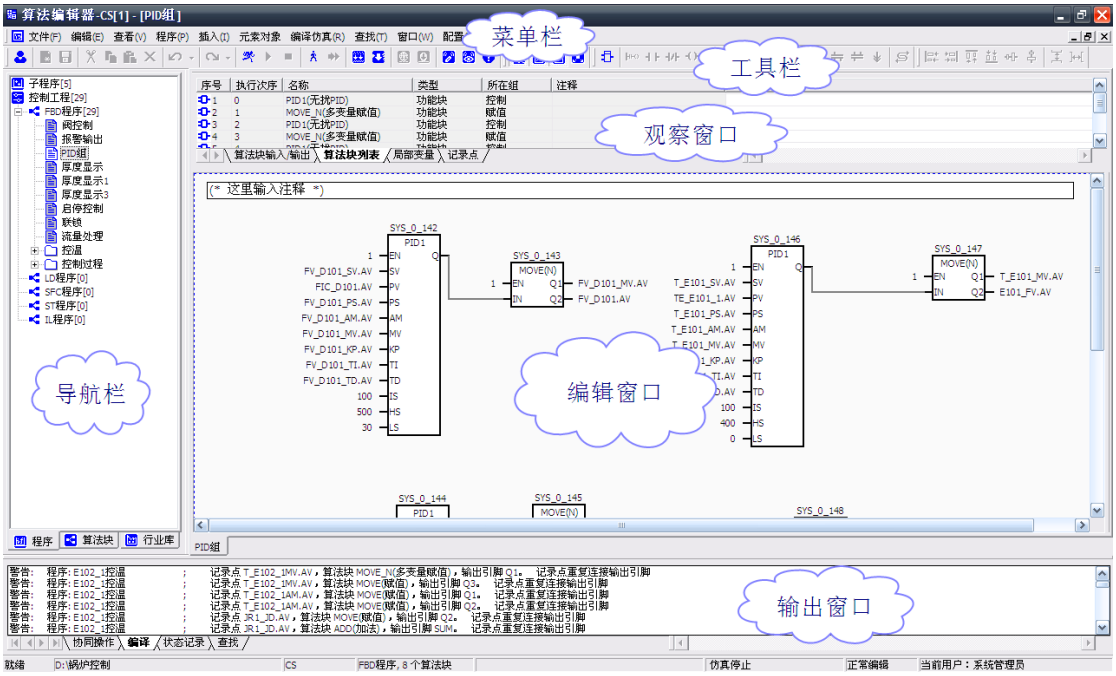


图 2- 1 算法编辑器窗口

2.1.1. 菜单栏

菜单栏中包含了算法编辑器中的大部分功能操作。窗口顶部显示的一行为“菜单栏”，分别表示某一类功能。点击一个菜单项，可以以下拉菜单的形式显示出其中的子菜单项。对菜单项的规定符合常规 Windows 规定。

只有菜单项为蓝色时才能使用，灰色不能使用。

菜单项后面括号中的字母或组合键表示调用该菜单项的快捷方式。

菜单项后面紧跟“…”时，表示将弹出对话框。

2.1.2. 工具栏

工具栏包括了菜单命令的一部分，用于快捷操作。工具栏中的所有按钮均以图形方式表示，当鼠标移到按钮上面时屏幕会浮动显示文本，提示用户该按钮的功能。

和传统的 Windows 应用程序相同，当按钮变为灰色时表示该按钮为无效状态。

2.1.2.1. 主工具条

主工具条（如图 2-2 所示）包括了一些编程的基本操作，如保存、编译、仿真等。

图 2- 2 主工具条

表 2- 1 主工具按钮解释

按钮	用途	使用范围
	用户登录	无用户登录或切换用户登录
	新建程序	程序组或某种编程方法
	保存控制工程	控制工程被改变之后
	剪切操作	编辑状态
	拷贝操作	编辑状态
	粘贴操作	编辑状态
	删除操作	编辑状态
	撤销操作	编辑状态
	恢复操作	编辑状态
	连续仿真	编辑状态
	继续连续仿真	连续仿真状态
	暂停连续仿真	连续仿真状态
	单周期仿真	编辑状态
	运行至下一周期	单周期仿真状态
	编译工程	编辑状态
	全部下装	控制工程编译通过之后
	局部编译	编辑状态
	局部下装	控制工程编译通过之后
	在线编辑	编辑状态

	在线监视	控制工程下装成功之后
	关于	查看状态

2.1.2.2. 外观工具条

外观工具条（如图 2-3 所示）用来设置界面外观，如显示/隐藏窗口、放大、缩小等。

图 2- 3 外观工具条

表 2- 2 外观工具按钮解释

按钮	用途	使用范围
	显示/隐藏导航栏	编辑状态
	显示/隐藏引脚观察窗口	编辑状态
	显示/隐藏信息输出窗口	编辑状态
	全屏显示	编辑状态

2.1.2.3. 插入工具条

插入工具条（如图 2-4 所示）用来在编辑窗口中插入图形元素，如 FBD 算法块、接点、步等。

图 2- 4 插入工具条

表 2- 3 插入工具按钮解释

按钮	用途	使用范围
	插入算法块	FBD、LD
	插入 LD 网络	LD
	插入常开触点	LD
	插入常闭触点	LD
	插入常开线圈	LD
	下插常开触点	LD
	下插常闭触点	LD
	插入步	SFC
	插入转换	SFC
	下插选择分支线	SFC
	下插选择聚合线	SFC

	下插并行分支线	SFC
	下插并行聚合线	SFC
	插入跳转	SFC
	转换元素类型	LD、SFC

2.1.2.4. 排列工具条

排列工具条（如图 2-5 所示）用来在编辑窗口中算法块的排列，如对齐、等间距等。

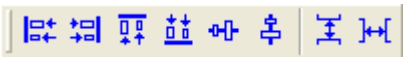


图 2- 5 排列工具条

表 2- 4 排列工具条按钮解释

按钮	用途	使用范围
	图块向左对齐	FBD
	图块向右对齐	FBD
	图块向上对齐	FBD
	图块向下对齐	FBD
	图块水平中心对齐	FBD
	图块垂直中心对齐	FBD
	图块垂直等间距	FBD
	图块水平等间距	FBD

2.1.3. 导航栏

导航栏有三个切换子窗口，分别显示工程中的程序、算法块、行业库等信息。

2.1.3.1. 程序

用于对程序和子程序进行分组、新建、删除、编辑和修改以及属性设置等管理操作。编程语言包括 FBD、LD、SFC、ST 和 IL 五种。

2.1.3.2. 算法块

用于浏览或选择算法编辑器所支持的所有系统功能块和函数。

这里要区别两种不同的算法块，一种叫“函数”，另一种叫做“功能块”。函数算法块只有一个输出，并且没有内部变量，函数的输出与上一周期的运行状态无关，固定的输入总是得到固定的输出。

功能块有内部变量或者该算法块有不止一个输出。有的功能块有内部变量，有内部变量的功能块，内部变量保存着以前周期的运行状态，它的输出不但与当前的输入有关，还与功能块以前的状态有关。即相同的输入不一定能得到确定的输出。

算法块窗口如图 2-6 所示。

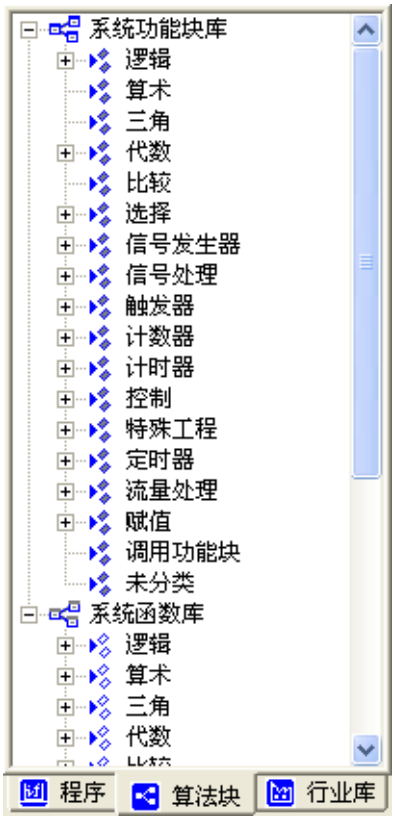


图 2- 6 算法块窗口

在图 2-6 的树形窗口中，算法块被分为两个根节点显示，一个是“系统功能块库”，一个是“系统函数库”，前一个根节点下面全部是功能块，后一个根节点下面全部是函数。功能块和函数下分为若干个组，每个组下面有若干个功能块或函数。

在窗口中点击右键，在右键菜单中选择“合并显示功能块和函数”，则功能块和函数合在一个根节点显示，如图 2-7 所示。

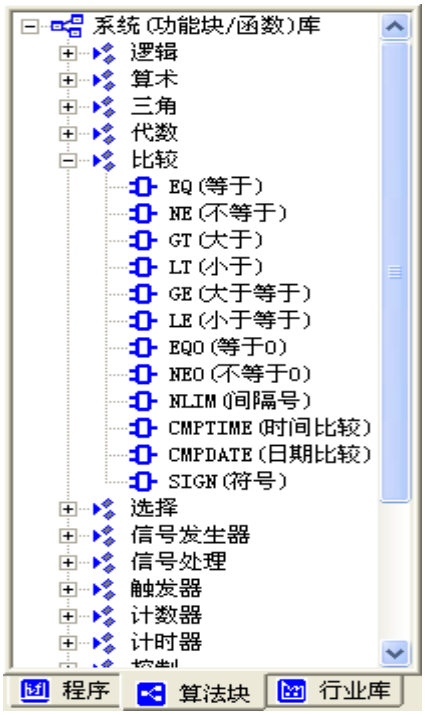


图 2- 7 算法块窗口

图 2-7 中，以不同的图标颜色表示算法块是属于哪种类型(空心图标表示的是函数，实心图标表示是功能块)。比如上图中的“RAND（随机值）”就是函数，而“G01（位振荡）”就是功能块。

2.1.3.3. 行业库

用于浏览或选择算法编辑器所支持的所有行业库功能块。

如图 2-8 所示在此窗口中可以看到所有支持的行业库以及算法块，也可以从其中选择需要的算法块拖入到程序中。

图 2- 8 行业库窗口

2.1.4. 编辑窗口

编辑窗口是程序员进行算法编辑、仿真的主要操作区域，用来对控制算法进行图形化的显示，用户可在其中进行添加、删除、调整图形元素。

2.1.5. 观察窗口

观察窗口用来显示当前进行编辑的程序算法中的图形元素信息以及程序的局部变量。不同的编辑状态观察窗口的显示内容也有所不同。图 2-9 所示为 FBD 编辑状态下的观察窗口。

图 2- 9 观察窗口

表 2- 5 各编辑状态的内容

编辑状态	切换页	显示内容
FBD	算法块输入/输出	指定 FBD 块的输入、输出引脚
	算法块列表	当前 FBD 程序的所有 FBD 块

	局部变量	当前 FBD 程序使用的局部变量
	记录点	当前 FBD 程序使用的记录点信息
	结构体	当前 FBD 程序使用的结构体信息
LD	LD 元素列表	当前 LD 程序的所有触点、线圈和算法块等元素
	局部变量	当前 LD 程序使用的局部变量
	记录点	当前 LD 程序使用的记录点信息
	结构体	当前 LD 程序使用的结构体信息
SFC	SFC 元素列表	当前 SFC 程序的所有步、转换、分支等元素
	局部变量	当前 SFC 程序使用的局部变量
	记录点	当前 SFC 程序使用的记录点信息
ST	局部变量	当前 ST 程序使用的局部变量
IL	局部变量	当前 IL 程序使用的局部变量

2.1.6. 输出窗口

输出窗口包括四个切换窗口，如图 2-12 所示。

1）协同操作：用于显示当前网络状态、算法编辑器的版本信息等；

A）控制信息：显示当前操作站、控制模件中运行工程的工程号以及算法版本号；用于确认控制模件中运行的版本与当前版本是否一致；

B）操作信息：显示当前网络中，运行同一工程的操作站的算法编辑器的版本号，当在网络中的工程操作版本号不一致的时候，可以通过双击来同步操作站间算法的版本，如图 2-10 所示，点击“开始同步”按钮进行站间模块同步操作。

图 2- 10 站间模块同步窗口

在进行算法站间模块同步操作之前，最好先进行数据库的同步操作。当在进行算法站间模块同步时数据库版本号不一致，则在同步的过程中，接收信息方会出现如图 2-11 所示弹出框。此时无论选择“是”按钮还是“否”按钮，数据库都会进行一次重构操作，因此在选择是或否之后，需要再次同步一次数据库。

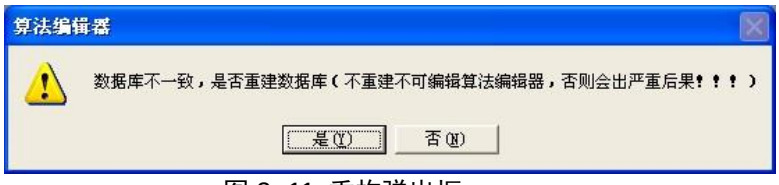


图 2- 11 重构弹出框

2）编译：编译工程后显示的信息，包括编译是否通过、警告信息、错误信息；

3）状态记录：仿真运行状态显示；

4）查找：显示查找结果，包括记录点查找结果、算法块查找结果以及子程序查找结果等；

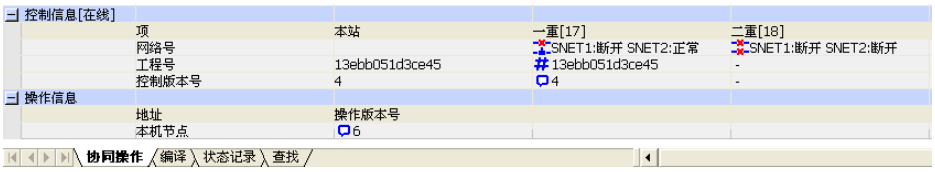


图 2- 12 输出窗口

3 控制工程

3.1. 控制工程

算法编辑器一次只能针对一个控制站进行控制算法的组态。

3.1.1. 保存控制工程

如果工程被修改过，需要保存时，有三种方法可以保存工程：

选择主菜单“工程”下面的子菜单中的“保存控制工程”项；

在工具条上点击“ ”按钮；

在导航栏“程序”子窗口里面的“控制工程”节点上单击右键，在弹出的菜单中单击“保存控制工程”选项，如图 3-1 所示。

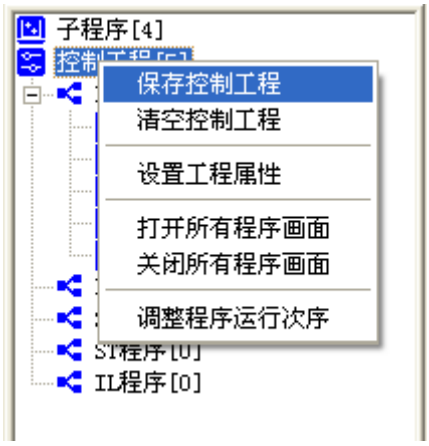


图 3- 1 保存工程

保存后，相应的保存菜单项和工具条按钮失效变灰，说明控制工程文件已经是最新的了。只有再次对工程进行修改后，保存菜单项和工具条上的保存按钮才重新变为有效。

3.1.2. 清空控制工程

当需要删除工程时删除方法有三种：

在主菜单“工程”下面的弹出子菜单上选择“清空控制工程”；

在导航栏上的“程序”子窗口里面的“控制工程”节点上右键单击，在弹出的菜单上选择“清空控制工程”，如图 3-2 所示。

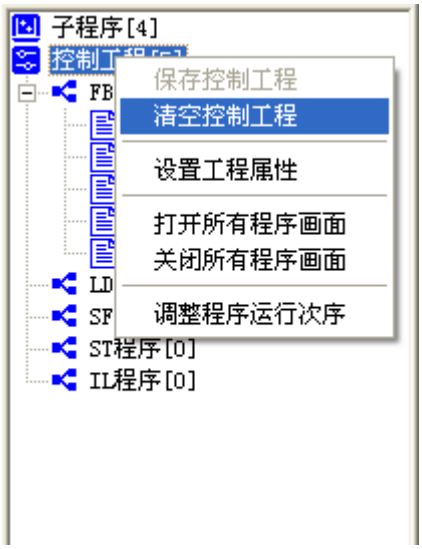


图 3- 2 清空控制工程

选中导航栏“程序”子窗口里面的“控制工程”节点，再按下 Delete 键；

选择菜单项后，将会弹出一个确认是否删除的对话框，如图 3-3 所示。如果选择“是”，则该工程被删除。工程删除后，工程中建立的所有程序（包括子程序）都被删除。

图 3- 3 清空工程确认对话框

记录点不能在算法编辑器中编辑，只能被引用。因此，清空控制工程不会影响记录点。

3.1.3. 控制工程设置

设置工程属性有两个方法：

在主菜单“文件”下面的弹出子菜单上选择“设置工程属性”；

在导航栏上的“程序”子窗口里面的“控制工程”节点上右键单击，在弹出的菜单上选择“设置工程属性”，如图 3-4 所示。

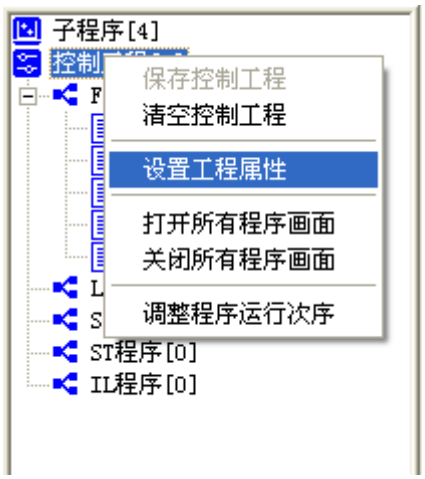


图 3- 4 工程属性设置

在图 3-5 所示的工程属性对话框中，可以对工程的运行基本周期和注释进行设置。

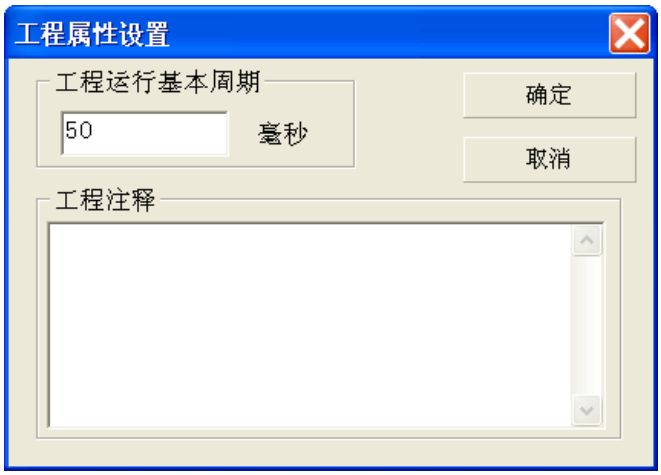


图 3- 5 工程属性对话框

工程运行基本周期的取值范围为 50～10000 毫秒，默认值为 50 毫秒。系统执行控制程序时，并不按工程的基本周期运行，而是按每个程序设置的周期分别独立运行。一般来说，程序周期是工程周期的若干倍。

工程注释是对该控制工程的简单描述，以备工程维护。

4 程序

控制工程是由若干个程序组成，每个程序采用某类控制语言编辑的具有独立运算周期的控制算法集。工程运行时，每个程序按其指定周期依次运行。程序之间是相互独立的，即每个程序的运行不依靠别的程序运行状态。

程序的管理和调度可通过导航栏内的程序窗口进行管理，如图 4-1 所示。

图 4- 1 程序窗口

4.1. 程序分类

在算法编辑器中，根据编程语言的不同，分为 5 大类程序：FBD 程序、LD 程序、SFC 程序、ST 程序和 IL 程序。这 5 种程序对应于 IEC61131-3 标准规定的 5 种控制语言，遵循标准的语法规定。对于每种语言的编辑方法将在下面的章节中加以描述。

如图 4-1 所示，这 5 种类型的程序分别建在“控制工程”节点下面的 5 个不同类型节点下面。而对于子程序，4 种不同类型的子程序全部建在“子程序”这一节点下面。

4.2. 程序分组

每种程序分类下面又可建立多个程序组，程序可直接建在分类节点下，也可建在不同的分组下面。如果程序比较多，为了管理的方便，建议建立程序组。

程序组建在分类节点下面，只能建一层，即程序组下面不能再建程序组。

4.2.1. 新建程序组

新建程序组有两种方法：

1. 选中分类节点（FBD、LD、SFC、ST、IL 任一个），选择主菜单“程序（P）”下面的弹出菜单中的“新建程序组”；

2. 选中分类节点（FBD、LD、SFC、ST、IL 任一个），单击右键菜单，在弹出的菜单中选择“新建程序组”，如图 4-2 所示。

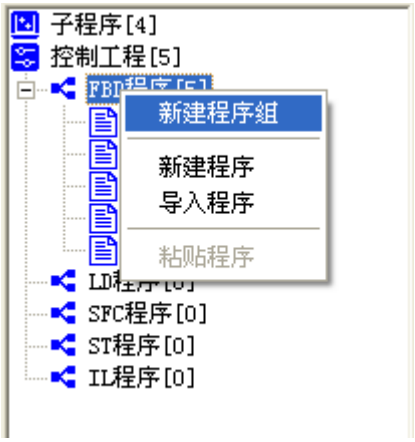


图 4- 2 新建程序组

选择菜单项后，将会弹出一个对话框，如图 4-3 所示，输入程序组名然后点击“确定”按钮即完成程序组的建立。

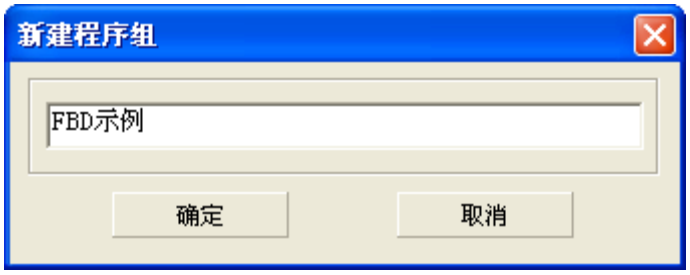


图 4- 3 输入程序组名称

4.2.2. 删除程序组

有三种方法删除程序组：

1. 选中要删除的组节点，选择主菜单“程序（P)”下面的弹出菜单中的“删除组”；

2. 选中要删除的组节点，单击右键，在弹出的菜单中选择“删除程序组”，如图 4-4 所示。

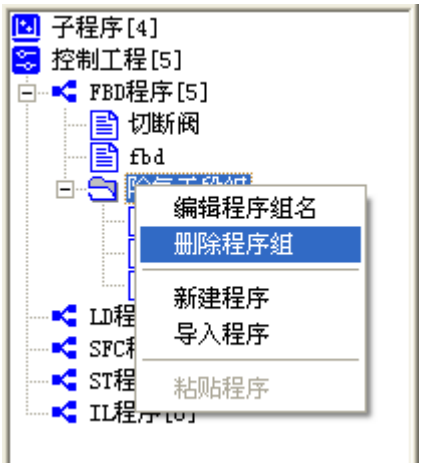


图 4- 4 删除程序组

3. 选中要删除的组节点，按下 Delete 键；

选择删除组后，将弹出一个确认对话框，如图 4-5 所示。点击“是（Y)”后，该组被删除。这里要注意的是，如果删除组，该组下的所有程序也将被删除。

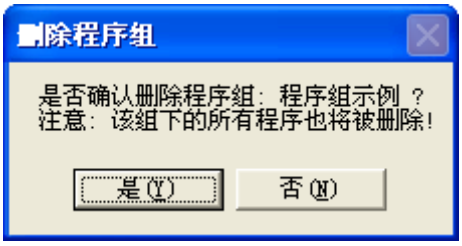


图 4- 5 删除程序确认对话框

4.2.3. 修改程序组

程序组改名的方法：

1. 选中要改名的组节点，选择主菜单“程序（P)”下面的“编辑组名”；

2. 选中要改名的组节点，单击右键，在弹出的菜单中选择“编辑程序组名”；

选择菜单项后，将会弹出一个对话框，如图 4-6 所示，输入程序组名然后点击“确定”按钮即完成程序组名的修改。

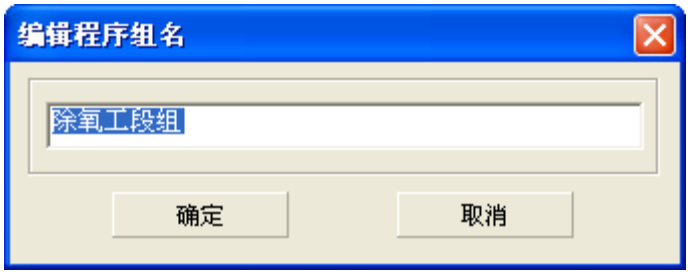


图 4- 6 程序组改名

注　意

同类程序节点下不允许出现重复的程序组名。

4.3. 新建程序

新建程序可以用以下方法之一进行：

1．选中分类节点或程序组节点，选择主菜单“程序（P）”下面的“新建程序”；

2．选中分类节点或程序组节点，点击工具条上的按钮“ ”；

3．选中分类节点或程序组节点，单击右键，在弹出的菜单中选择“新建程序”，如图 4-7 所示。

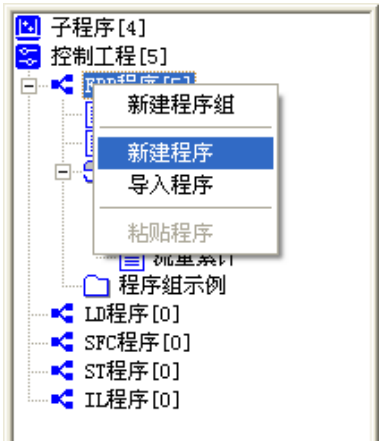


图 4- 7 新建程序

选择菜单项后，将会弹出如图 4-8 所示的对话框。

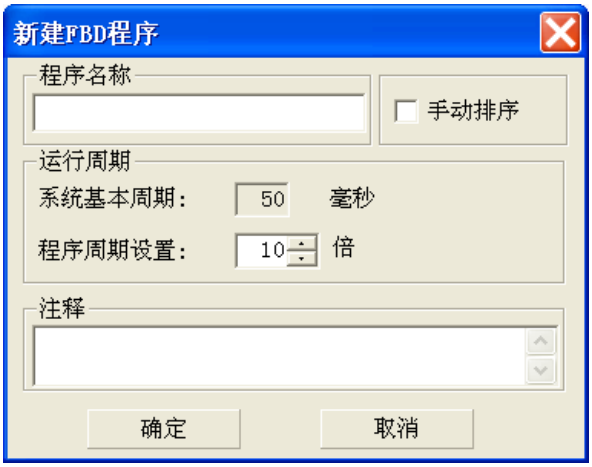


图 4- 8 新建 FBD 程序

程序名称：在“程序名称”对话框中输入程序的名称（注意程序不能与现有的程序名称一样，也不能与系统保留字冲突）。

运行周期：程序周期是系统基本周期的整数倍。系统基本周期在控制工程属性中设置。缺省的程序周期是 500 毫秒（50*10）。

注释：为方便程序的管理，以备维护，最好在注释编辑框中输入程序的简要注释。

按确定后，选中分类节点或程序组节点下就会增加一个程序节点，同时系统自动打开该程序，进入编辑状态。程序的初始编辑窗口为一个空白的程序编辑窗口。

4.4. 删除程序

删除程序的方法：

1. 在导航栏的“程序”子窗口内选中程序节点，选择主菜单“程序（P）”下面的“删除程序”；

2. 在导航栏的“程序”子窗口内选中程序节点，单击右键，在弹出的菜单中选择“删除程序”；

3. 在导航栏的“程序”子窗口内选中程序节点，按下 Delete 键；

这时将弹出一个确认对话框，按“是（Y）”后，程序将被删除，如图 4-9 所示。

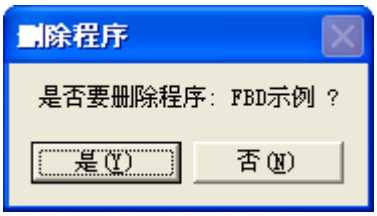


图 4- 9 删除程序确认对话框

4.5. 移动和复制程序

移动和复制程序仅限于在同一类程序内部进行。

移动程序

编辑完一个程序后，如果要把它移到另一个组去，可以用拖放的方法。在导航栏“程序”子窗口内选中程序节点，按住鼠标左键不松手，鼠标拖动到另一个组节点上，然后再松开鼠标，这时，这个程序就移动到另一个组节点下面了。

复制程序

复制程序有三种方法：

1. 选中程序节点，按住 Ctrl 键和鼠标左键不松手，拖动程序节点到同一分类节点或另一个程序组，最后松开鼠标，同时目标组节点下面将会出现另一个程序节点。

2. 在导航栏“程序”子窗口内选中程序节点，在主菜单“程序（P）”中选择“复制程序”，再选中目标组，在菜单中选择“粘贴程序”，则可以复制程序到目标分组。

3. 在导航栏“程序”子窗口内选中程序节点，在节点上单击右键，在弹出的菜单中选择“复制程序”，如图 4-10 所示。

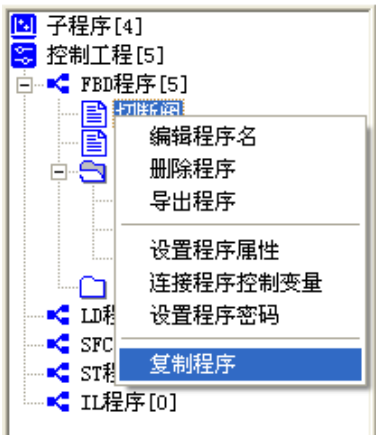


图 4- 10 复制程序

复制程序到程序组中：选中目标组，单击右键，在弹出的菜单中选择“粘贴程序”，如图 4-11 所示。

图 4- 11 粘贴程序

这时目标组下出现一个新程序节点，其程序名为原先程序名称后的数字加 1。

所有复制的程序名称是原程序名称的数字加 1，例如原程序名称是“FBD 示例”，复制以后程序名称变成“FBD 示例 1”，而当原程序名称为“FBD 示例 1”，则复制后程序名称变为“FBD 示例 2”，依此类推。

4.6. 程序改名

程序改名的方法：

1. 在导航栏内选中要改名的程序节点，选择主菜单“程序（P）”下面的“编辑程序名”；

2. 选中要改名的组节点，单击右键，在弹出的菜单中选择“编辑程序名”，如图 4-12 所示；

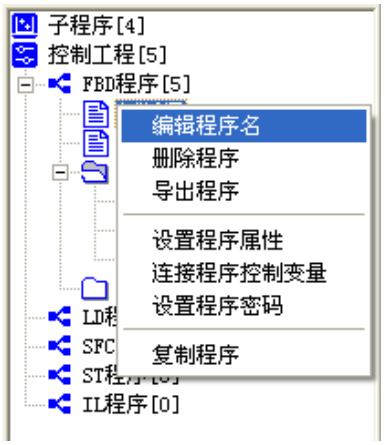


图 4- 12 程序改名

选择菜单项后，将会弹出一个对话框，如图 4-13 所示，输入程序名然后点击“确定”按钮即完成程序名的修改。

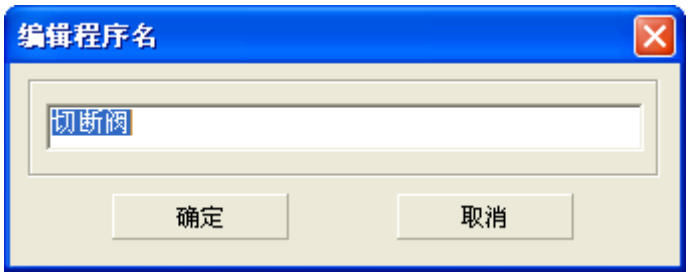


图 4- 13 改名过程

4.7. 程序设置

设置程序属性的方法：

1. 选中导航栏内“程序”子窗口中的程序节点，然后选择菜单上的“设置程序属性”；

2. 选中导航栏内的程序节点，单击右键，在弹出的菜单中选择“设置程序属性”；

选择菜单项后，弹出一个对话框，如图 4-14 所示。在这里可以修改程序的名称、程序的周期、程序的注释。

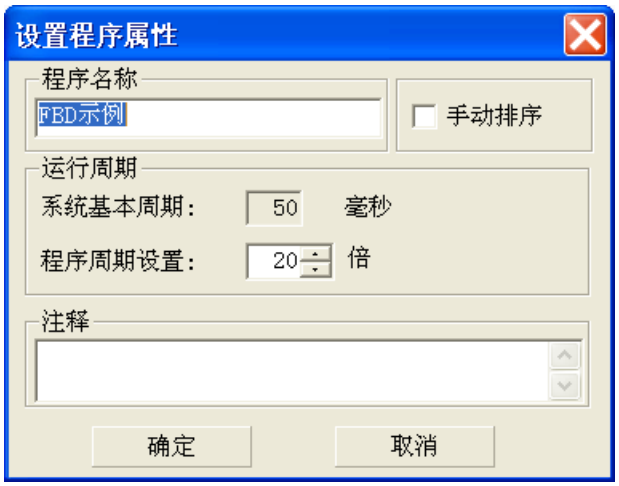


图 4- 14 程序属性设置

在程序属性中，程序运行周期是系统基本运行周期的倍数，系统基本运行周期在工程属

性中设置。例如，如果系统基本运行周期为 100 毫秒，程序周期设置为 12，那么该程序的实际运行周期为 100×12＝1200 毫秒。

4.8. 程序的导入和导出

用户可以将控制工程中的某个程序导出为一个文件，文件后缀名是“*.pou”；也可以从导出的程序文件中导入程序到控制工程中。

程序的导入和导出使不同工程间的算法复制成为可能。比如，编制了一个程序，如果想在另一个工程中编制同样的程序，可以通过导入导出的方法来实现控制方法。

导出和导入的步骤如下：

导出

1．选中导航栏中的程序节点，在该节点上单击鼠标右键，在弹出的菜单中选择“导出程序”，如图 4-15 所示；

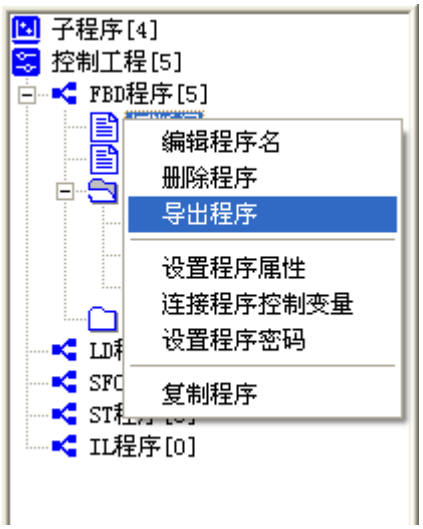


图 4- 15 导出程序

2．弹出一个另存为对话框，如图 4-16 所示，让用户选择导出程序的目标目录；

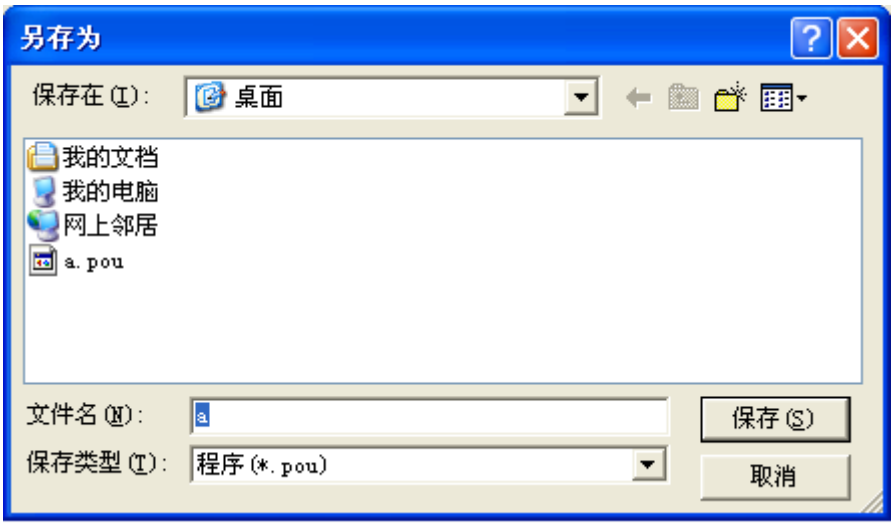


图 4- 16 另存为对话框

3．点击“保存（S）”后，则弹出一个选择对话框，如图 4-17 所示，供用户导出后的程序做是否去掉程序中算法块的数据库连接信息；

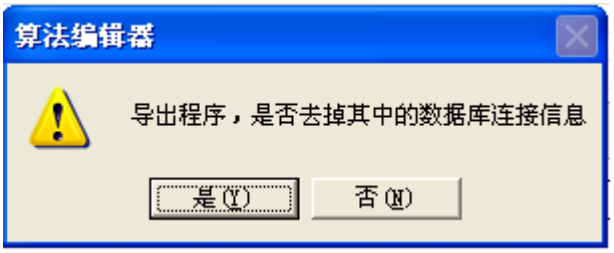


图 4- 17 选择对话框

4．在上图 4-17 中若点击“是（Y)”，则不保留数据库的连接，重新导入时，需要重新进行数据库连接操作。若点击“否（N)”，则导出的程序保留了所有数据库的连接，重新导入时，再次点击“否（N)”，若算法连接的记录点位号在数据库中已建立，不需要重新连接；

5．输入导出后的文件名，点击“保存（S)”按钮，程序就被导出为磁盘文件“FBD 示例. pou”。

导入

1．导入程序时，在导航栏内的分类节点或者程序组节点上单击鼠标右键，在弹出的菜单中选择“导入程序”，如图 4-18 所示；

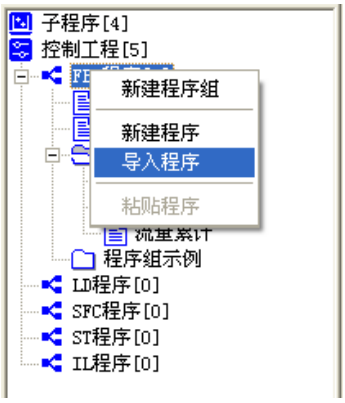


图 4- 18 导入程序

2．在弹出的文件选择对话框中选择要导入的文件，如图 4-19 所示；

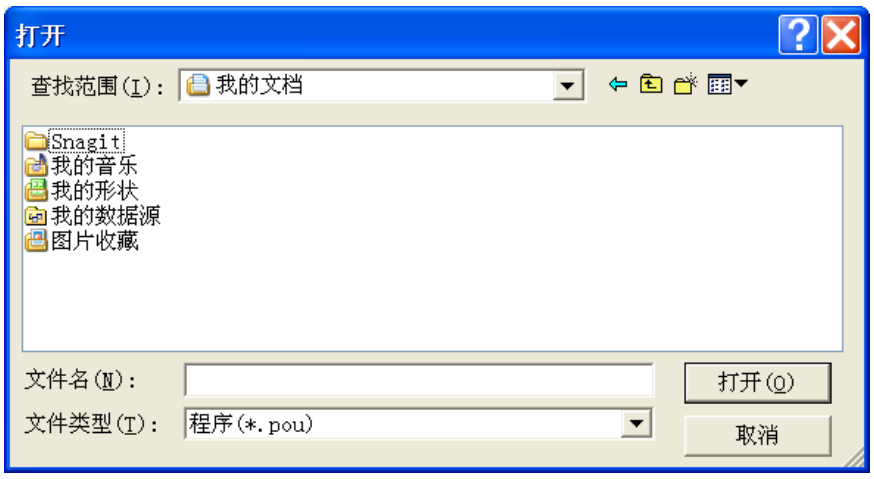


图 4- 19 打开对话框

3．点击打开，弹出如图 4-20 所示的选择对话框；

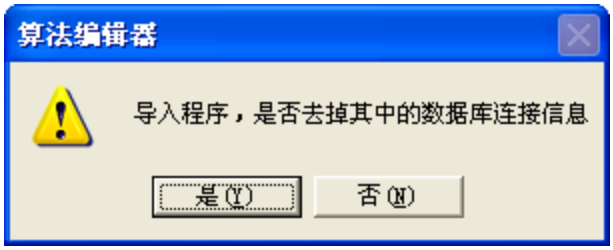


图 4- 20 选择对话框

4．不管上图选择是与否，新导入的程序出现在选中的组节点或分类节点下面，新节点名称就是文件的名称，如果新名称与系统中已有的程序名称相同，则自动在文件名称后面添加数字来以示区别。

注　意

1．SFC 程序不能导出；
2．FBD 程序和 LD 程序导出时，记录点的连接都会丢失。因为不同的工程，记录点数据库是不一样的，所以一个工程中的变量连接在另一个工程中可能就没有意义了。

4.9. 程序的运行次序

缺省状态下，控制工程编译运行时，各个程序的运行次序是其生成时的顺序。根据需要，用户可以调整程序的运行次序，调整运行次序仅对具有相同运算周期的程序才有意义，此时用户可以通过确定的运行次序来了解程序的运行机制，从而在多个程序需要对同一个记录点进行读写操作时，能对该记录点的状态有准确的了解和合理的利用。

调整方法如下：

1. 选择主菜单“工程”下的“调整程序运行次序”菜单项；

2. 在导航栏中的“控制工程”节点上单击鼠标右键，在弹出的菜单中选择“调整程序运行次序”，如图 4-21 所示。

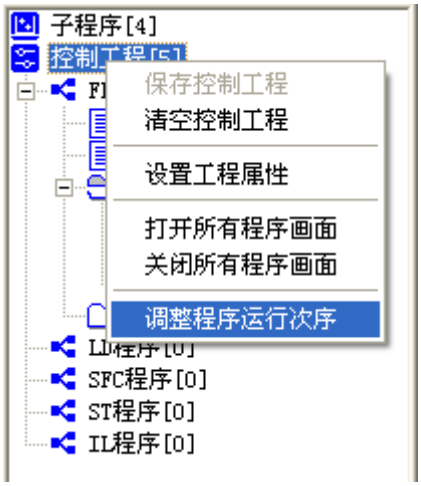


图 4- 21 调整程序运行次序

选择出现如图 4-22 所示的对话框，这个对话框列出了所有的程序（不包括子程序），可以通过点击“上移”按钮和“下移”按钮来调整程序在列表中的排列。修改保存后，程序将按列表中的排列运行。

图 4- 22 调整次序

4.10. 程序的变量连接

程序的变量连接允许工程人员通过设置程序的运行控制变量和运行时间变量来控制程序的运行。

运行控制变量的功能是当该运行控制变量实时值为 FALSE 时，该程序不执行，当为 TRUE 时该程序按照设定的运算周期来执行。

运行时间变量的功能是将当前程序的实际运行时间赋给该变量。

设置方法如下：

1. 在导航栏中，在要进行变量连接的程序节点上单击鼠标右键，在弹出的菜单中选择“连接程序控制变量”，如图 4-23 所示；

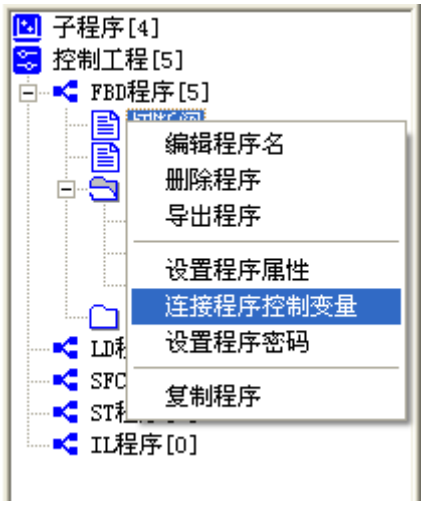


图 4- 23 程序变量连接

2. 选择后，出现如图 4- 24 所示的对话框，通过该对话框用户可以设置当前所选中的程序的运行控制变量和运行时间变量。

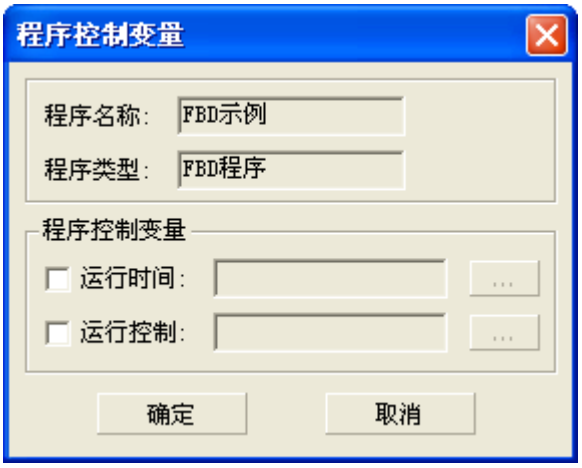


图 4- 24 程序控制变量

对于 SFC 程序，增加了如下几个控制变量，“程序复位”、“禁止转换”、“强制步进”和“操作使能”变量，如图 4-25 所示。

这几个变量的功能如下：

当程序复位所连接的变量为 TRUE 时，该 SFC 程序立即跳转到起始步执行。

当禁止转换所连接的变量为 TRUE 时，该 SFC 程序停留在当前步，不进行转换条件的判断，直到该变量为 FALSE 时。

当强制步进所连接的变量为 TRUE 时，该 SFC 程序忽略当前的转换条件的判断，立即跳转到下一步执行。

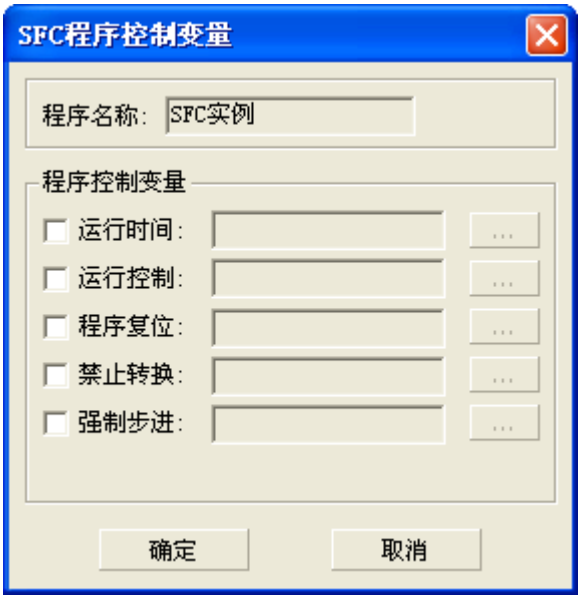


图 4- 25 程序控制变量

4.11. 程序的编辑窗口切换

程序切换有 3 种方法：

1. 在导航栏内双击打开各个程序节点，用户可对处于已打开状态的程序通过单击导航栏内的程序节点实现不同程序之间的切换。

2. 处于已打开状态的程序，在编辑窗口的右下会生成标签，通过点击标签即可实现。标签如图 4-26 所示。

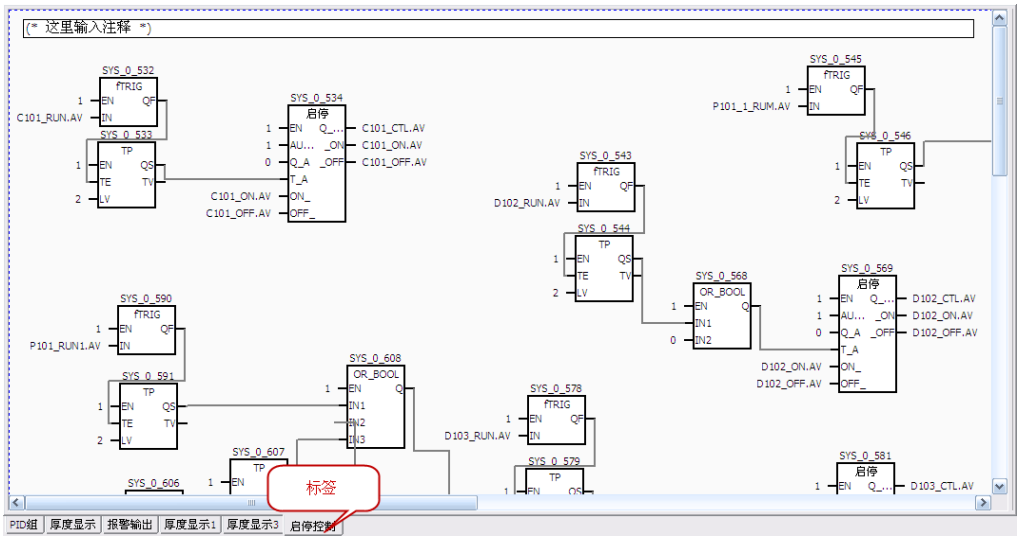


图 4- 26 程序编辑框

3. 在程序编辑窗口中插入按钮，用户通过按钮切换到其他程序（此方法仅限 FBD 程序），具体设置如下：

选定一个程序编辑窗口，在窗口空白处右键选择“插入按钮”，如图 4-27 所示。

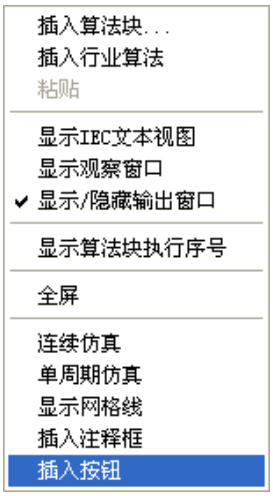


图 4- 27 右键列表

点击插入按钮之后，在程序编辑窗口生成按钮。用户可通过左键拖动按钮，并放置到适当的位置，然后双击按钮弹出选择程序框，如图 4-28 所示。

图 4- 28 选择程序框

用户在图 4-28 中选择所要切换到的程序，选中之后，点击“确定”，在按钮上显示所链接程序的名称，如图 4-29 所示，在仿真运行的时，用户通过点击按钮实现程序切换。

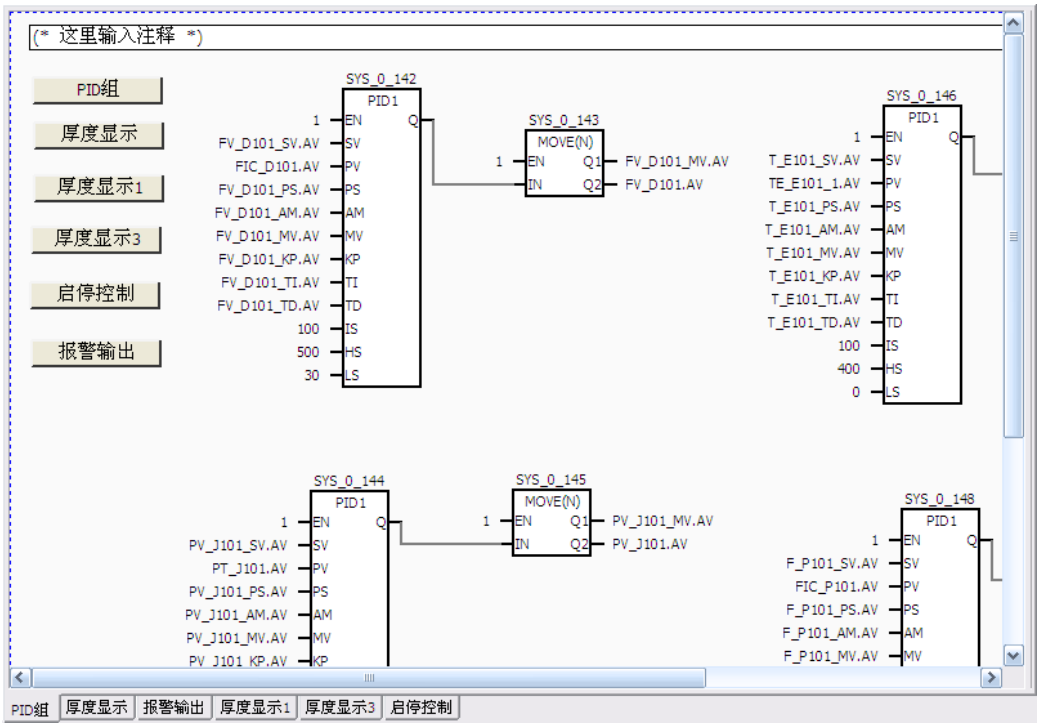


图 4- 29 按钮设定

5 子程序操作

子程序与程序基本上是一样的，但子程序只能被程序通过 CAL 功能块或 CAL 指令调用，如果不被调用，则不参与运行。

子程序的管理主要在导航栏内的“程序”子窗口，如图 5-1 所示。

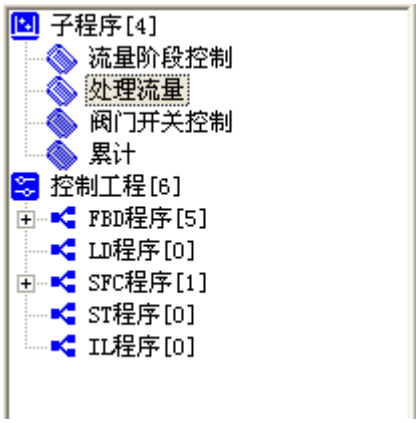


图 5- 1“程序”子窗口

5.1. 子程序的分类

子程序分为 4 大类：FBD 程序、LD 程序、ST 程序、IL 程序。SFC 程序不能被调用，所以不能生成子程序。与程序不同的是，子程序根节点下没有分类节点，根节点下直接建立分组或子程序。

5.2. 子程序组

在子程序根节点下可建立若干程序组，子程序可直接建在子程序根节点下，也可建在不同的分组下面。如果子程序比较多，则用分组的方式比较好。

对子程序组的操作有新建、删除、修改名称。具体操作方法与程序组基本上是一样的，可以通过主菜单或者右键菜单进行。

5.3. 新建子程序

新建子程序可以通过以下方法：

1. 选中子程序根节点或子程序组节点，选择主菜单“程序（P）”下面的“新建程序；
2. 选中子程序根节节点或子程序组节点，点击工具条上的按钮“”；
3. 选中分类节点或程序组节点，单击右键，在弹出的菜单中选择“新建子程序”，如图 5-2 所示。

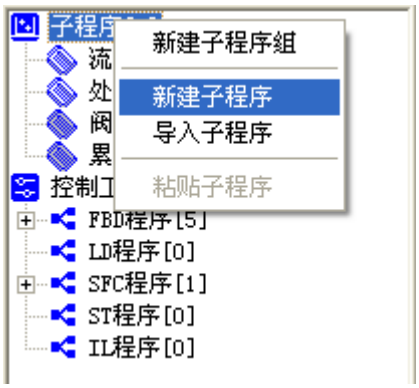


图 5- 2 新建子程序

选择菜单项后，将会弹出如图 5-3 所示的对话框。

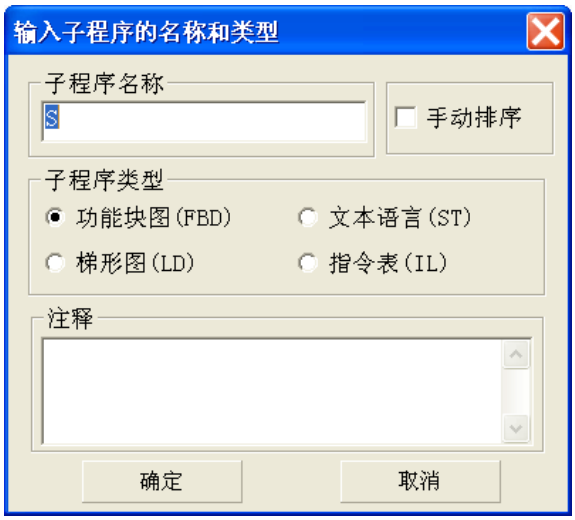


图 5- 3 新建子程序对话框

子程序名称：在“子程序名称”对话框中输入子程序的名称（注意子程序不能与现有的程序名称一样，也不能与系统保留字冲突）。

子程序类型：选择一种子程序的语言类型。

注释：为方便程序的管理，以备维护，最好在注释编辑框中输入程序的简要注释。

按确定后，选中节点下就会增加一个子程序。同时系统新建一个空的子程序编辑窗口。

5.4. 删除子程序

删除子程序的方法：

1. 在导航栏内选中子程序节点，选择主菜单“程序（P）”下面的“删除程序”；

2. 在导航栏内选中子程序节点，单击右键，在弹出的菜单中选择“删除子程序”；

3. 在导航栏的“程序”子窗口内选中程序节点，按下 Delete 键；

这时将弹出一个确认对话框，按“是（Y）”后，程序将被删除，如图 5-4 所示。

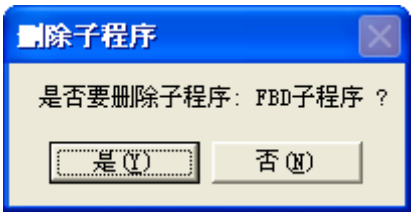


图 5-4 删除子程序对话框

5.5. 移动和复制子程序

参考程序的移动和复制，但子程序只能在子程序根节点下移动和复制。

5.6. 子程序改名

参考程序的改名。

5.7. 子程序设置

选中子程序节点，然后选择菜单上的“设置程序属性…”，或者点击右键，在弹出的菜单上选择“设置子程序属性”，则弹出一个对话框，可以在这里修改子程序的名称、改变程序的注释等。如图 5-5 所示。

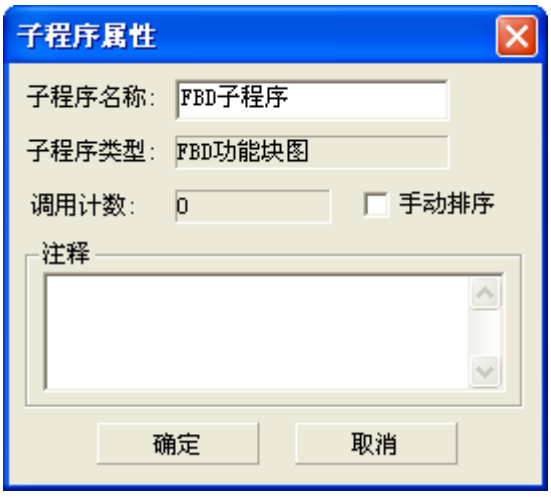


图 5-5 子程序属性设置

5.8. 子程序的导入和导出

用户可以将子程序导出为一个文件，文件后缀名是“*.pou”，也可以从导出的文件中导入子程序。子程序的导入导出方法和程序的导入导出是一样的。

6 编辑器

算法编辑器包括 5 种控制语言的编辑器：FBD 功能块图（Function Block Diagram）、

LD 梯形图（Ladder Diagram）、SFC 顺序控制图（Sequential Function Chart）、ST 结构化文本语言（Structed Text）、IL 指令表语言（Instruction List）。其中 FBD、LD、SFC 这三种是图形化的语言，以图形元素以及它们之间的连接表示控制语句代码和代码执行流程，ST 和 IL 是文本语言。

6.1. FBD 功能块图编辑器

FBD 功能块图编辑器以图形化的块图方式和连接线来组织控制算法，可以较为清晰的显示出控制流程。

FBD 图中最主要的图形元素是算法块和它们之间的连接线。一个算法块表示一种运算。算法块左边的输入引脚表示算法的输入，算法块右边的引脚表示算法的输出。输入/输出引脚可以连接变量，算法块之间的连接线表示中间值的传递。

6.1.1. 外观介绍

FBD 功能块图编辑器的外观如图 6-1 所示，共分为观察窗口和编辑窗口两大部分。

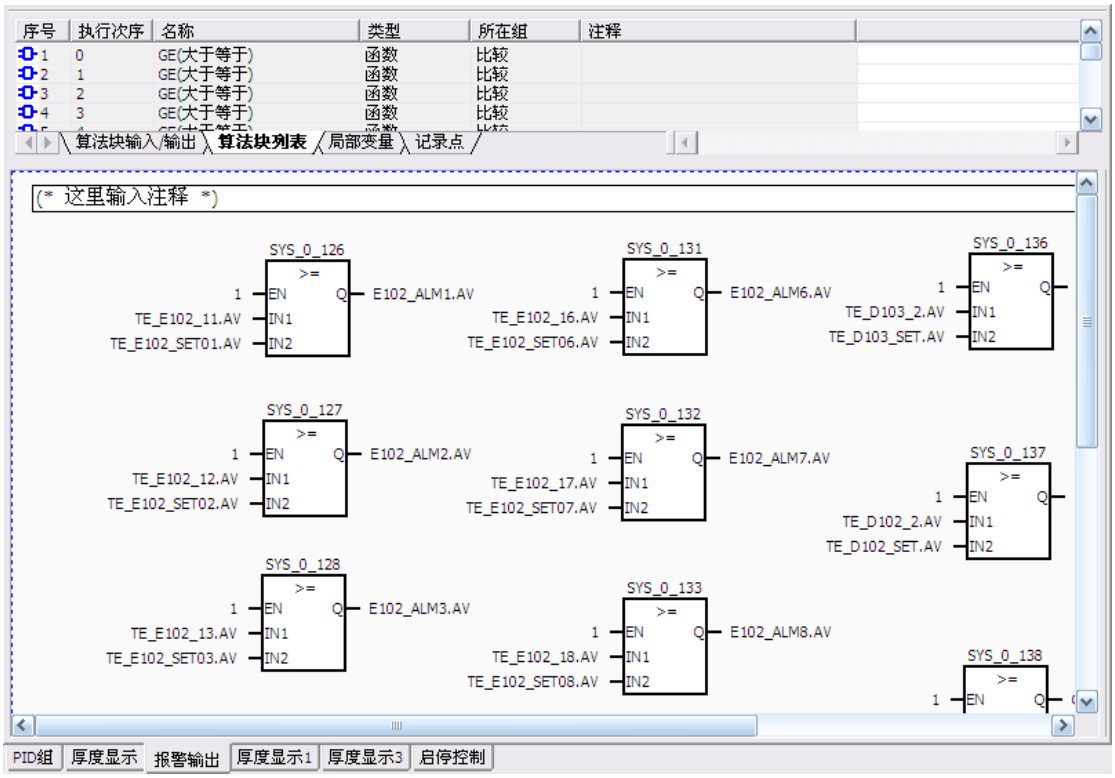


图 6- 1 FBD 编辑器外观图

从图中可以看出， 观察窗口包括 4 个可切换的子窗口：算法块输入/输出窗口、算法块列表窗口、局部变量编辑窗口、记录点信息窗口。

6.1.1.1. 算法块输入/输出窗口

如图 6-2 所示，算法块输入/输出窗口的作用是显示当前编程画面上当前被选中算法块的引脚属性。该窗口仅用于显示，不可修改。

图 6- 2 算法块输入/输出窗口

表 6-1 算法块输入/输出窗口内容

列	解释
第一列	引脚属性(输入/输出)
第二列	输入输出引脚的索引号
第三列	引脚名称
第四列	引脚的数据类型：模拟型、数字型、整数型
第五列	引脚的连接类型
第六列	引脚连接的内容：如果是变量，则显示变量名称；如果是常数，则是常数的初始值；如果为空或连接线，则不显示。
第七列	引脚的当前值，这一列只在仿真时才会显示数值
第八列	引脚的简单注释

当画面上的一个算法块选中时，该窗口就显示出这个算法块中的所有输入/输出引脚。在算法块的引脚上点击时，这个窗口中的相应的行被选中。

6.1.1.2. 算法块列表窗口

如图 6-3 所示，算法块列表窗口的作用是显示当前编程画面内所有算法块的列表。当选中一个算法块时，该窗口中相应的行被选中。双击某一行，可以弹出该行所在 FBD 算法块的属性窗口。

在算法块列表窗口中，当选中某一个算法块时，编辑窗口中对应的算法块将会高亮显示，同样在编辑窗口中选中某个算法块，算法列表窗口中对应的行将被选中，这样可以快速查找所想编辑的算法块.

图 6- 3 算法块列表窗口

表 6-2 算法块列表窗口内容

列	解释
第一列	算法块的索引，每个算法块在该程序中的索引是唯一的
第二列	下一个要执行的算法块的索引：当这个算法块表示的运算执行结束后，接下去要执行的算法块
第三列	算法块的名称

第四列	算法块类型：是函数还是功能块。
第五列	算法块类型分组
第六列	算法块的注释，该注释不是算法块本身的注释，是用户为了表示这个算法块的运算作用而输入。

6.1.1.3. 局部变量窗口

如图 6-4 所示，局部变量窗口的作用是为当前 FBD 程序编辑局部变量。局部变量的定义及具体操作方法请参阅后面的语言说明章节。

图 6- 4 局部变量窗口

表 6-3 局部变量窗口内容

列	解释
第一列	局部变量的索引号
第二列	局部变量名
第三列	局部变量的类型：模拟型、整数型或数字型
第四列	局部变量的初始值
第五列	仿真时局部变量的运算结果
第六列	局部变量是否掉电保持
第七列	对局部变量的注释

6.1.1.4. 记录点信息窗口

如图 6-5 所示，记录点信息窗口用于显示当前 FBD 程序所连接的记录点信息。

图 6- 5 记录点窗口

对记录点窗口各列的解释见表 6-4 所示。

表 6-4 记录点信息窗口内容

列	解释
第一列	序号

第二列	记录点的索引号（例 ID 为[1][12]：[1]为控制站 ID，[12]为数据库记录点 ID）
第三列	记录点名称
第四列	仿真时记录点的运算结果
第五列	对记录点的注释

6.1.1.5. 算法编辑窗口

在算法的观察窗口下面的空白区域即为算法编辑窗口部分。算法的编辑窗口主要用来对算法进行编程，其中，可以通过鼠标右键菜单进行算法组态，如图 6-6 所示。

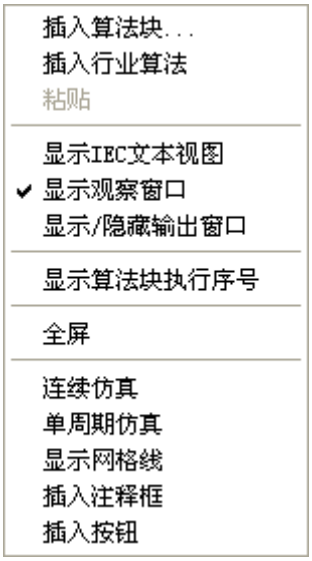


图 6- 6 算法编辑窗口右键菜单

在右键菜单中，可以实现插入算法块及行业库的功能，以及自主选择是否显示 IEC 文本视图、观察窗口、输出窗口、算法块执行序号等，还可以通过右键菜单进行算法的连续仿真与单周期仿真功能。

在右键菜单中选择插入注释框，然后可以在注释框中输入注释信息，可以是对某一段程序的注释。注释框可以自由拖动，且在算法编辑窗口中可以插入多个。

在右键菜单中选择插入按钮，然后选中该按钮，双击后弹出一个选择程序的对话框，任意选择一个程序。该按钮实现的功能是，在算法仿真时，通过点击按钮在程序页之间可以进行跳转。

6.1.2. 算法块的编辑

6.1.2.1. 算法块的外观

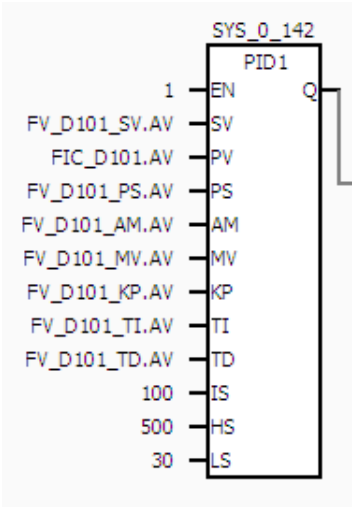


图 6- 7 算法块

如图 6-7 所示，算法块的形状为一个矩形块，左边的引脚表示此算法的输入，右边的引脚表示该算法的输出。

算法块外部上方是此算法块对应的结构体名称，内部上方为此算法块的名称，内部左边每个输入引脚都有文字一一对应，它表示输入引脚的名称（如上图中的 EN、SV 等），内部右边每个输出引脚也有文字一一对应，它表示输出引脚的名称（如图 6-8 中的 Q）。

算法块输入引脚的左边表示该引脚的输入连接变量（或常数），算法块输出引脚的右边表示该引脚的输出连接变量。

所有算法块的第一个输入引脚总是 EN 使能端，只有这个引脚的输入值为 1 时，算法块才运算，否则不运算。

当鼠标移动至算法块时直接显示算法块注释。

6.1.2.2. 算法块的属性设置

将鼠标移到一个算法块上面并双击，会弹出一个对话框，该对话框显示了该算法块的属性，如图 6-8 所示。

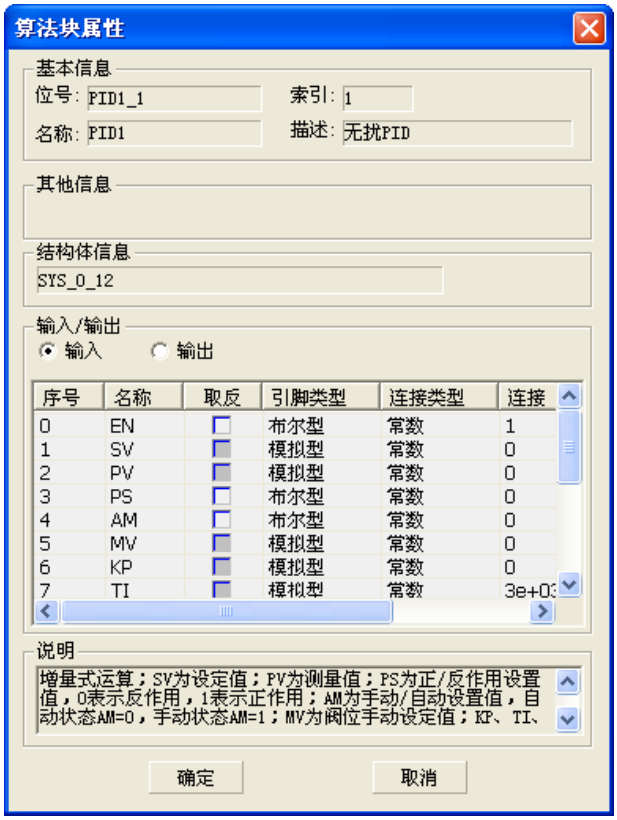


图 6- 8 算法块属性

算法块的基本信息包括位号、名称、索引，以及中文描述。位号是以算法块的名称再加上索引来命名的；名称一般显示的是英文；描述是对算法块的中文解释；索引是指在当前控制程序中算法块的序号，按照算法块被编辑的先后顺序给它分配索引，每个算法块的索引在当前画面中是唯一的。

“结构体信息”是算法块所连接的结构体的名称信息。

“输入/输出”是这个算法块的输入/输出引脚的一些属性列表显示。

“说明”是算法块本身的注释，是这个算法块功能的简要说明。

6.1.2.3. 插入算法块

插入算法块有以下途径：

1. 选择主菜单“插入”下面的“插入算法块”项；

2. 点击工具条上的“”按钮；

3. 在编辑窗口中点击右键，在弹出菜单中选择“插入算法块…”；

4. 直接从导航区的算法块切换页中选择并拖放。

当使用菜单或工具栏按钮时，则弹出如图 6-9 所示。

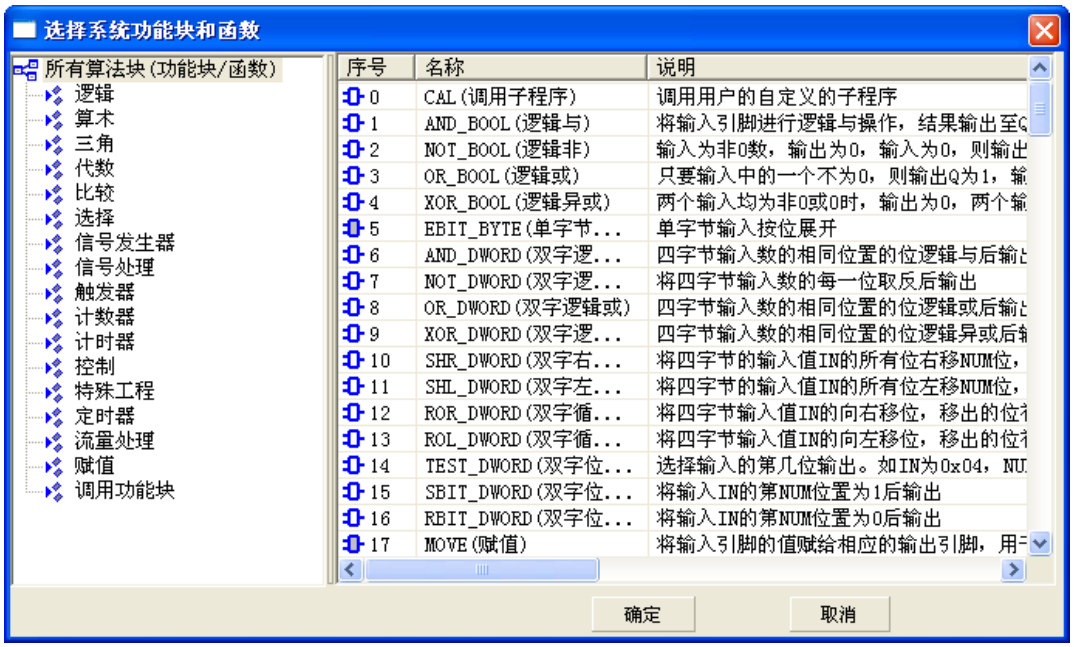


图 6- 9 算法块选择

该对话框罗列出了所有的算法块，包括系统功能块和函数。左边树形子窗口，显示算法块的分组。右边列表子窗口显示左边选中分组中的所有算法块。功能块和函数以不同的图标表示（空心图标表示的是函数，实心图标表示是功能块）。

在右边列表子窗口相应的行上面双击，则此对话框退出。这时可以看到，随着鼠标的移动，画面上出现一个方框，按下鼠标左键，可将选中的算法块放在指定位置（注意：算法块不能重叠放置）。插入算法块时实时数据库自动生成一个结构体与之对应，算法的编辑窗口上并不显示结构体，只在算法块上方显示此结构体名称，对算法块可以做其它正常的编辑。

在算法编辑器中对 AND、OR、MOVE、MOVE(N)、ADD、MUL 这些算法块，显示的时候，在其下方有一个小方块，当鼠标移到这个小方块时，鼠标变为上下箭头，可以通过向上或向下拖动这个小方块，增加这些算法块的输入/输出参数，如图 6-10 所示。

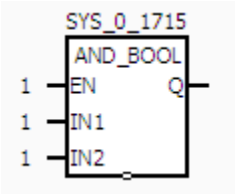


图 6- 10 算法块的操作

6.1.2.4. 复制粘贴算法块

复制算法块有以下途径：

1. 选中算法块，选择主菜单“编辑”下面的“拷贝”项；

2. 选中算法块，在算法块上按下鼠标右键，在弹出的菜单中选择“拷贝算法块”，如图 6-11 所示。

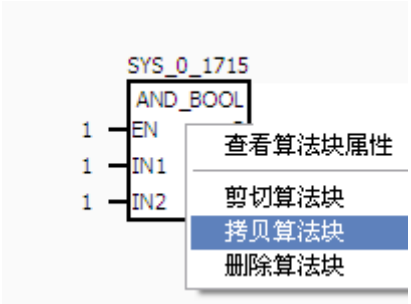


图 6- 11 算法块的复制

3. 选中算法块，用快捷键 Ctrl+C 进行复制。

粘贴算法块有以下途径：

1. 选择主菜单“编辑”下面的“粘贴”项；

2. 在编辑空白的地方点右键，在弹出的菜单中选择“粘贴”，如图 6-12 所示；

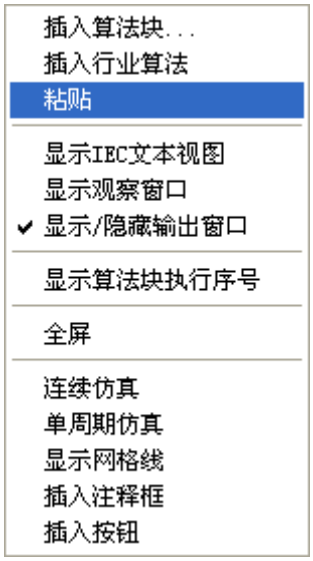


图 6- 12 算法块的粘贴

3. 用快捷键 Ctrl+V 进行粘贴。

算法块的复制一共可以分为以下几种：

1. 多个复制：多个复制就是可以选择一个或一个以上的算法块一起复制，如图 6-13 所示；

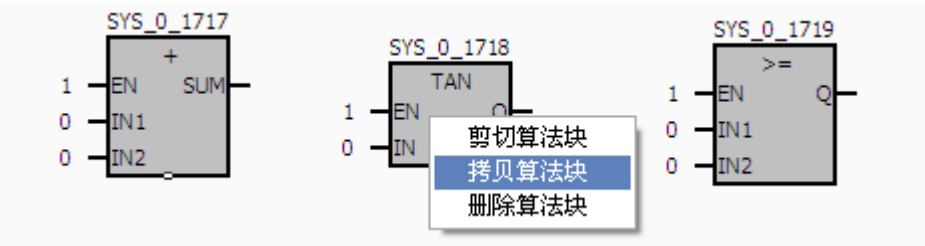


图 6- 13 算法块的多个复制

2. 跨程序复制：跨程序复制允许算法块在页面间的复制，即可以从程序 A 复制到程序 B；

3. 带引脚的复制：一般，引脚连接有 4 种类型，分别是常量、局部变量、数据库记录点、结构体引脚。如果是在当前页面进行复制，那么算法块在复制时连同引脚所连的变量一起复制，包括常量、局部变量、结构体引脚以及数据库记录点；如果是跨程序的复制，那么除局

部变量外，算法块连同引脚一起复制（局部变量局部唯一）；

4. 带连接线的复制：在程序中，多个算法块之间是有连接线的，在复制时，选中连接线，那么算法块连同输入引脚的连接线一起复制，不包括输出引脚的连接线，如图 6-14、图 6-15 所示。

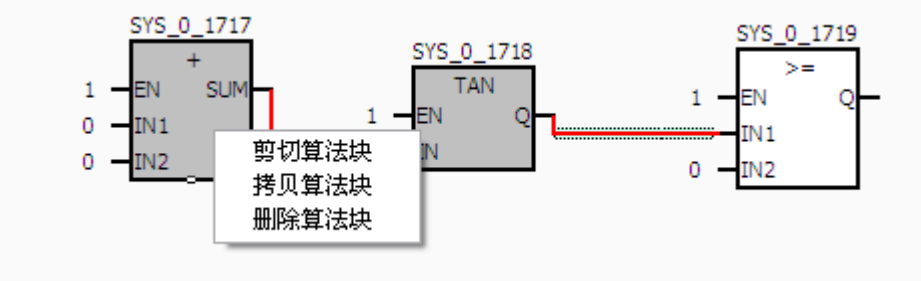


图 6- 14 带连接线的复制（前）

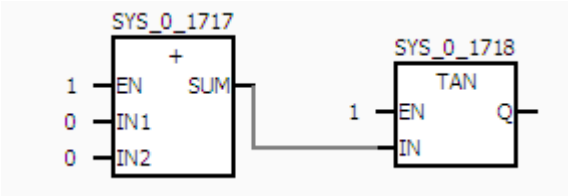


图 6- 15 带连接线的复制（后）

6.1.2.5. 剪切算法块

剪切算法块与复制相同，请参看 6.1.2.4 节。

6.1.2.6. 删除算法块

删除算法块有如下两种方式：

1. 选中算法块，按下键盘上的 Delete 键；

2. 选中算法块，在算法块上按下鼠标右键，在弹出的菜单中选择“删除”，如图 6-16 所示。

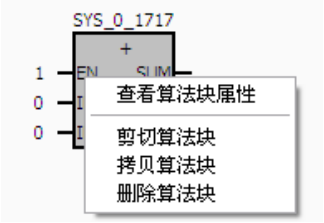


图 6- 16 删除单个算法块

支持多个算法块删除，可以使用左键拉出一个选择框选择多个算法块，只有算法块整体全部在选择框中时才有效；还可以使用 CTRL+左键点击算法块，将算法块加入或剔除选择队列。在选中的算法块上点击右键选择删除，或直接按下 Delete 键。如图 6-17 所示。

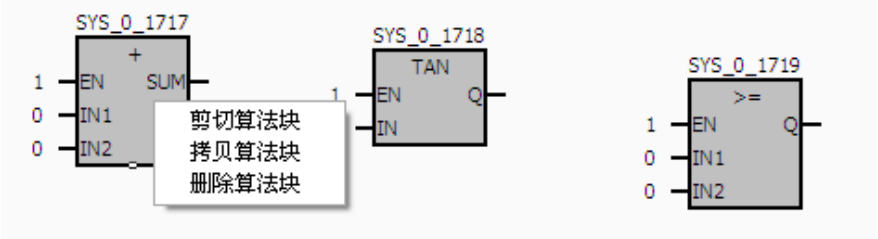


图 6- 17 删除多个算法块

注　意

删除算法块后，连接在算法块上的连接线也随之删除。

6.1.2.7. 算法块引脚的连接

通过算法块引脚与变量的连接，实现变量的输入与输出和值的传递。

算法块的输入引脚有四种连接情况：

1. 连接记录点。引入记录点的值，参与运算；
2. 连接局部变量。局部变量只在当前程序范围有效；
3. 连接常数。即运算时，该算法块的相应输入恒定；
4. 连接线。输入值由线的另一头连接的输出引脚决定；

输出引脚也有四种连接情况：

1. 连接记录点。将算法块的结果输出到该记录点，实际运行时，将值输出到现场；
2. 连接局部变量。局部变量只在当前程序范围内有效，作为运算的中间变量；
3. 连接为空，即什么都不连接，运算时不输出；
4. 连接线。运算输出由连接线引到别的算法块的输入上去；

注　意

对于某些算法块，其引脚可能只允许进行上述连接类型的一个或几个，比如 FCTGEN 算法块的 NUM 引脚只允许连接常数。

- 引脚与变量、常数连接

引脚可以通过如下几种方法建立连接：

➢ 通过引脚属性

在算法块的引脚上双击，会弹出一个对话框，该对话框显示引脚的属性及其连接情况，并且可在这个对话框中设置引脚的连接，如图 6-18 所示。

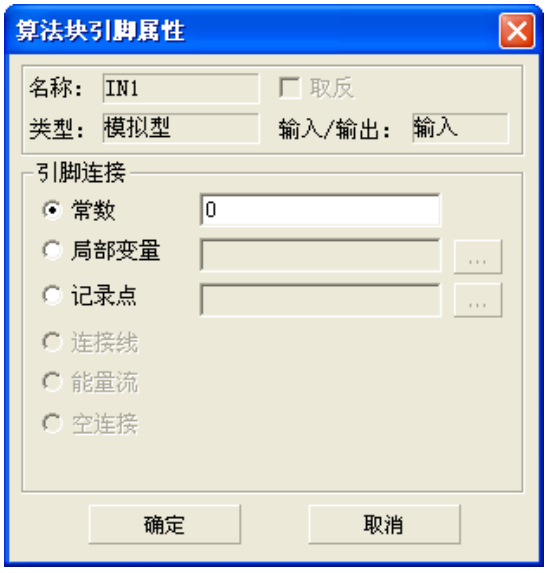


图 6- 18 引脚属性对话框

“引脚连接”表示是一个输入引脚的连接（如果是输出引脚，常数选项会变灰）。每个引脚允许的连接类型由算法编辑器配置文件 dedf.cfg 决定。

如果连接常数，在编辑框中输入常数值。

如果连接局部变量，点击“局部变量”右边的按钮“ … ”，将弹出如图 6-19 所示的对话框，在这里选择要连接的局部变量，不能直接输入。

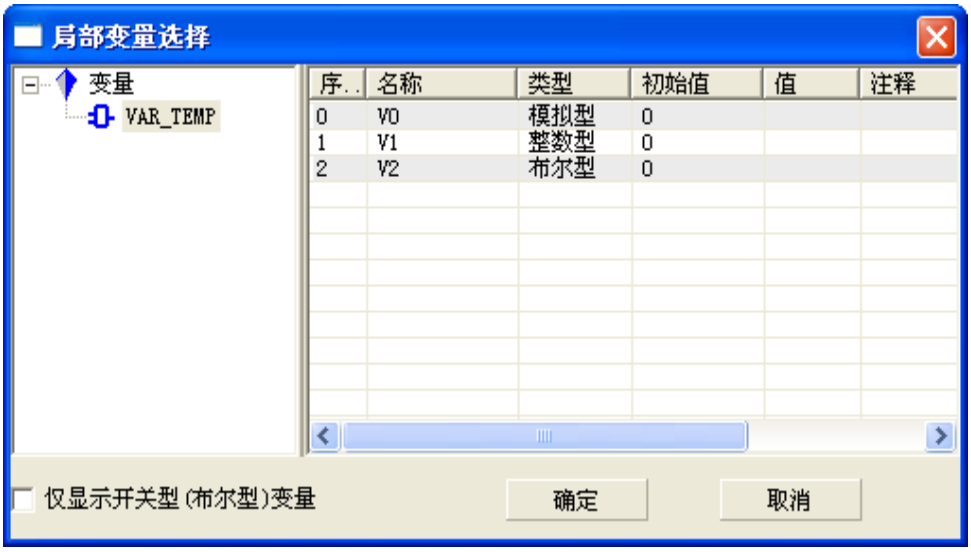


图 6- 19 局部变量选择

如果连接记录点，点击“记录点”右边的按钮“ … ”，将弹出如图 6-20 所示的记录点选择对话框。

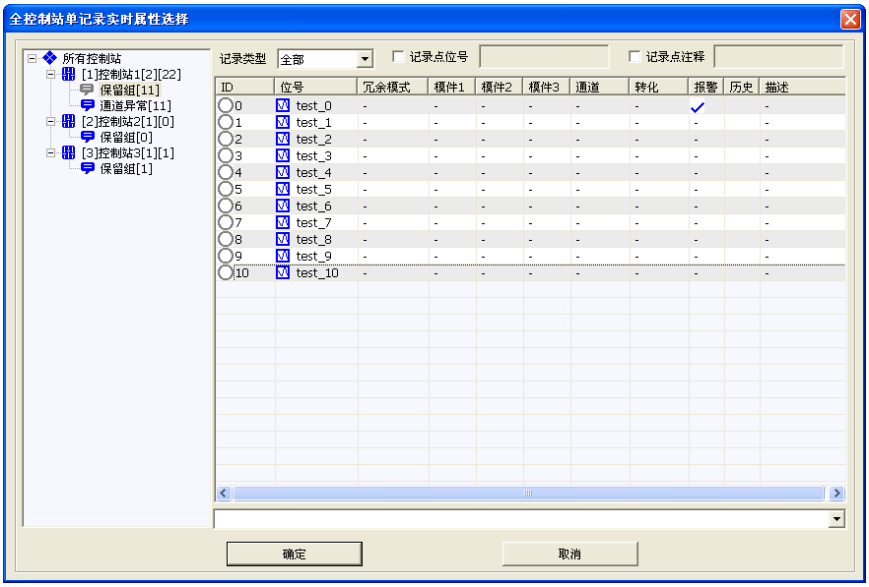


图 6- 20 记录点选择

在这里选择变量。左边是变量所在的控制站和分组，左边分组被选中后，右边显示分组中所有变量。在右边列表窗口中选中变量，然后选择需要连接变量的属性，目前在算法中能够引用的记录点属性包括实时值、记录类型、最大值、最小值、质量戳，然后按“确定”，则相应变量及属性被选择。

- 通过算法块属性

双击算法块，会弹出一个对话框，该对话框显示算法块的所有输入输出引脚，并且可在这个对话框中设置引脚的连接，如图 6-21 所示。

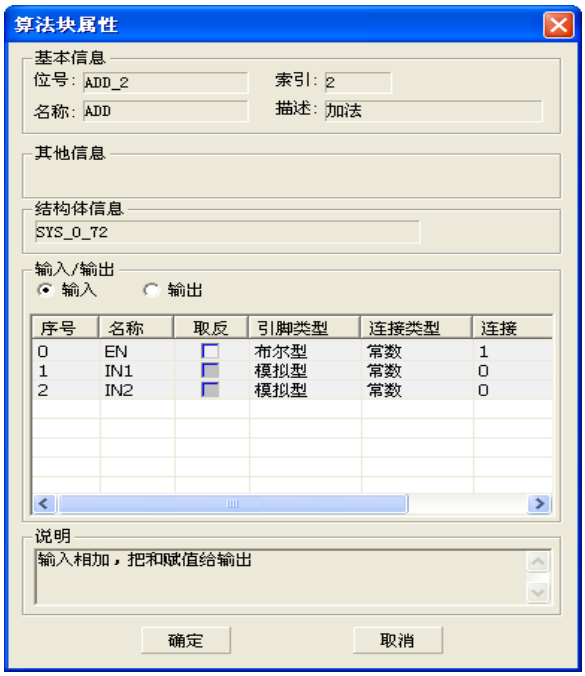


图 6- 21 引脚属性链接

单个输入或输出引脚链接，只需双击要链接的引脚所在行，弹出如图 6-19 所示的对话框，然后再进行链接，具体操作步骤和通过引进属性进行链接一样。

在图 6-23 所示的对话框中可以支持多个输入或输出引脚的链接。在算法块属性对话框中批量选择引脚，可以输入和输出一起选择，只需在输入/输出的页面内进行切换，然后选中引脚即可。然后在选中的引脚上单击鼠标右键，弹出子菜单，可以选择批量连接数据库记录点，如图 6-22 所示。

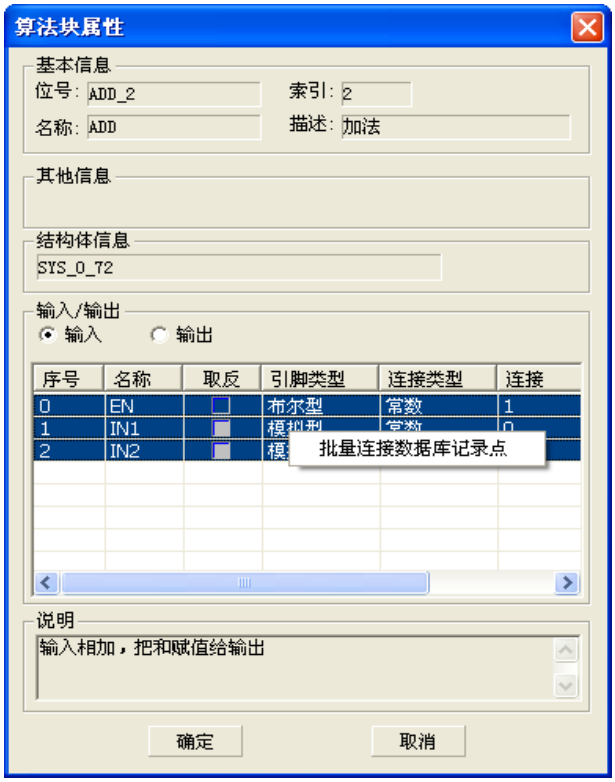


图 6- 22 选择输入/输出引脚

若选择批量连接数据库记录点，则弹出如图 6-23 所示的对话框，在该对话框内选择需

要连接的记录点即可，且在该对话框的左上角显示了需要连接记录点个数以及已选中记录点个数。

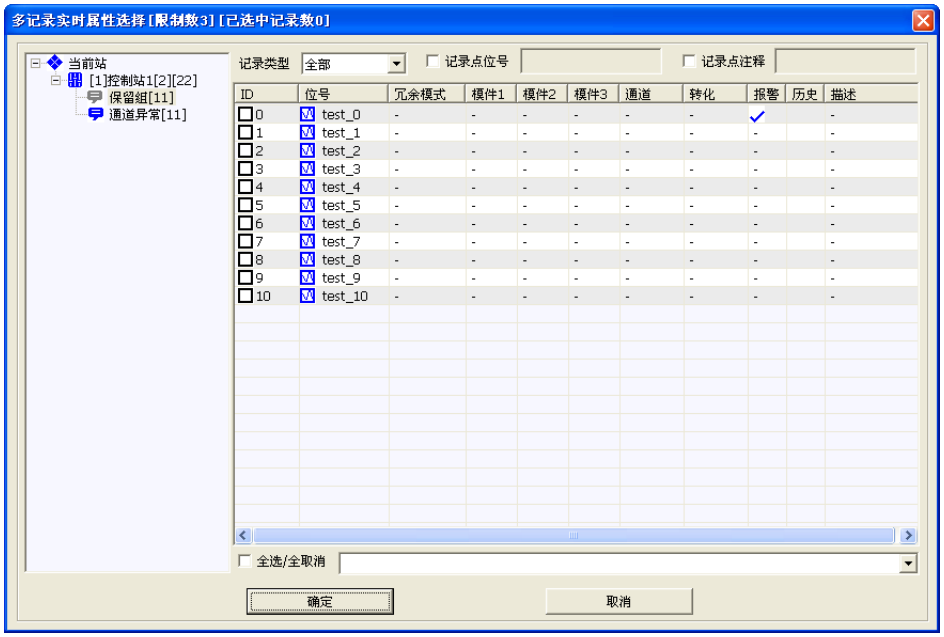


图 6- 23 批量选择记录点对话框

➢ 编辑框选择

在算法块引脚的旁边单击，将会弹出编辑框和一个连接窗口，可以在编辑框中输入变量名称或常数值，也可在连接窗口中双击要连接的变量，如图 6-24 所示。

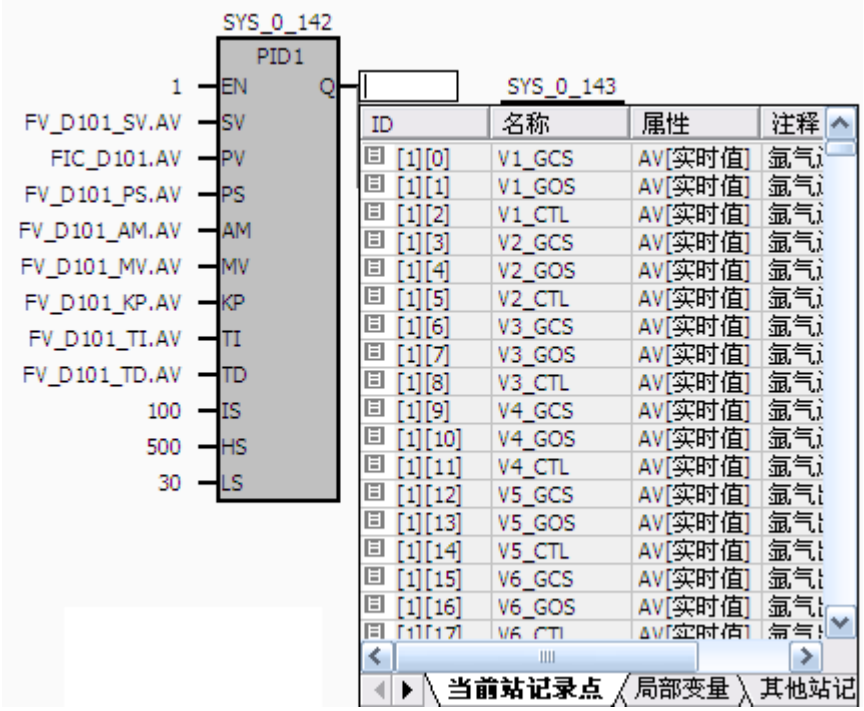


图 6- 24 编辑框和连接窗口

连接窗口中包括三个列表子窗口，一个列表窗口显示本站点的记录点，一个列表窗口显示本程序的局部变量，一个列表窗口显示其他站点的记录点。

在编辑窗口上方的观察窗口中的局部变量列表窗口中选中相应的局部变量，按住鼠标左键，拖动局部变量到算法块引脚上，可以实现局部变量与引脚的连接。

注　意

如果输入引脚已经连接了一根连接线，则无法与变量或常数进行连接；输出引脚连了连接线后，还可以与变量连接，这时的运算输出到变量上，也通过连接线输出到别的算法块的输入上。

- 引脚之间的连接

通过在算法块之间的连接，可以把一个算法块的输出导入到另一个算法块的输入去。在一个引脚上按下鼠标左键不松手，移动鼠标，这时会在引脚与鼠标之间画一道线，拖动这根线到另一个引脚上，这时两个引脚之间自动产生一条连接线，如图 6-25 所示。

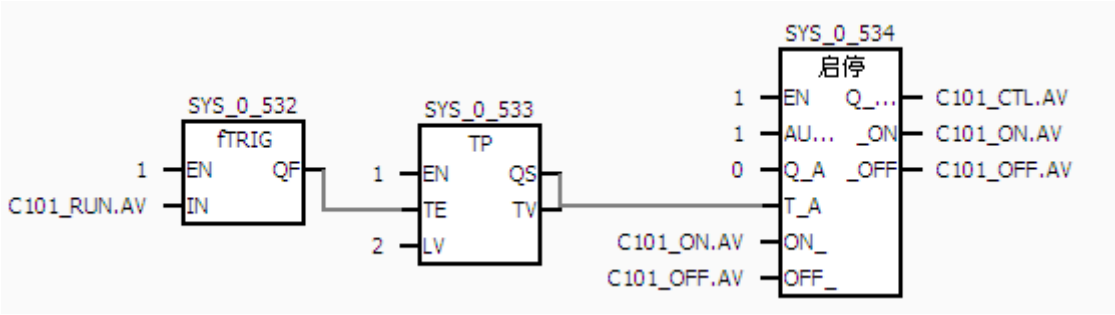


图 6- 25 引脚连接

两个连接的引脚必须一个是输入引脚，另一个是输出引脚。

对于输入引脚而言，如果以前连接的是变量、常数或连接线，连线后，以前的连接将会丢失。输出引脚在连接了变量的情况下，还能与别的引脚用连接线进行连接。

删除连接线有如下三种方式：

- 选中连接线，按下 Delete 键；
- 选中连接线，按下鼠标右键，在弹出菜单中选择“删除连接线”，如图 6-26 所示；

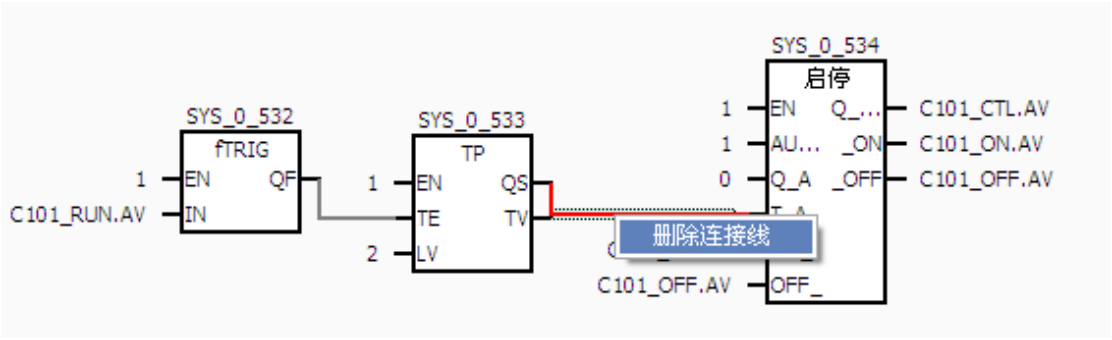


图 6- 26 删除连接线

- 选中算法块输入引脚，按下 Delete 键，连在此引脚上的连线即被删除。

按照以上任何一种方法删除连接线后，输入引脚的连接线将会消失，引脚连接变成常数，其值变为该算法块定义的默认值。

删除连接变量有如下三种方式：

- 选中输入/输出引脚，按下 Delete 键；
- 选中输入/输出引脚，按下鼠标右键，在弹出菜单中选择“删除引脚连接”，如图 6-27 所示。

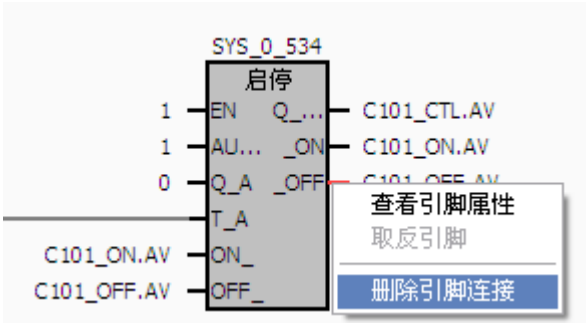


图 6- 27 删除连接变量

➢ 选中算法块引脚连接的变量，按下 Delete 键，连在此引脚上的变量即被删除。

按照以上任何一种方法删除连接变量后，连接的变量丢失，如果此时该输出引脚与其他的输入引脚有连接，则其连线不会丢失；否则该输出引脚的连接为空。

6.1.2.8. 算法块引脚编辑

在算法编辑器中，FBD 算法块以及 LD 算法中的触点、线圈所链接的内容可以直接剪切、复制及黏贴。

现有一个 FBD 算法块，其输入引脚链接了数据库中的某个记录点，鼠标单机选中该记录点，然后剪切或者复制，选中另一个引脚，将内容进行黏贴，然后鼠标点击空白处，或者按下 Enter 键，则引脚内容黏贴成功。

6.1.3. 算法块的执行顺序

系统算法块执行顺序分为手动跟自动排序 2 种。默认的排序方式为自动排序。可以通过设置程序属性对话框选择是自动排序还是手动排序。如图 6- 28 所示，选择了手动排序方式。

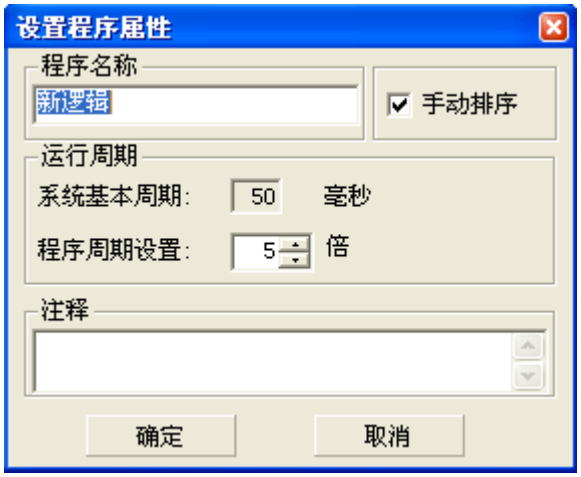


图 6- 28 程序属性设置

- 自动排序模式，系统依据规则自动生成算法块的执行顺序，排序的规则为：

1.新插入的算法块的执行次序排在最后；

2.如果算法块之间有连接关系，则依据数据流向进行追溯，先生成数据源的算法块的执行顺序在前，否则在后；

3.如果算法之间构成回路，则按照前一次排序的结果从回路中最先执行的算法块开始排序；

4.插入，删除一个算法块，添加一条连线等都会自动启动一次自动排序。

5.自动排序尽可能保存原来算法块执行顺序的相对不变性。如图 6-29 所示，这个小部分的执行顺序为 5->6->7->8->9，当有其他算法块移到前面执行时，这部分的执行序号会统

一变大，且保持相对的不变性。因此可以通过增删连接线来加速得到自己想要的执行顺序，而不通过手动排序。

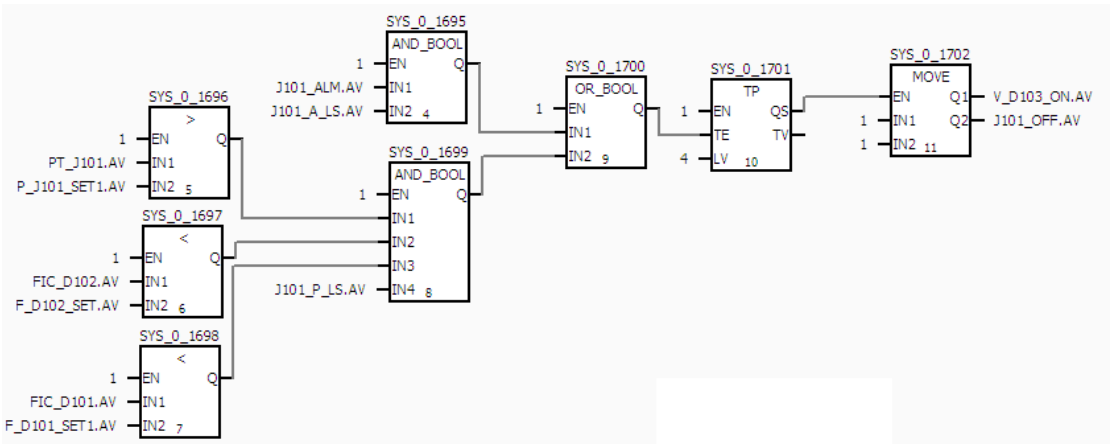


图 6- 29 程序属性设置

- 手动排序用于用户微调操作，由用户指定程序中各算法块的执行顺序。当程序为手动排序状态时，可通过右键点击算法块列表窗口选择算法块是向前移动，向后移动，还是移到最前或最后。如图 6-30 所示。

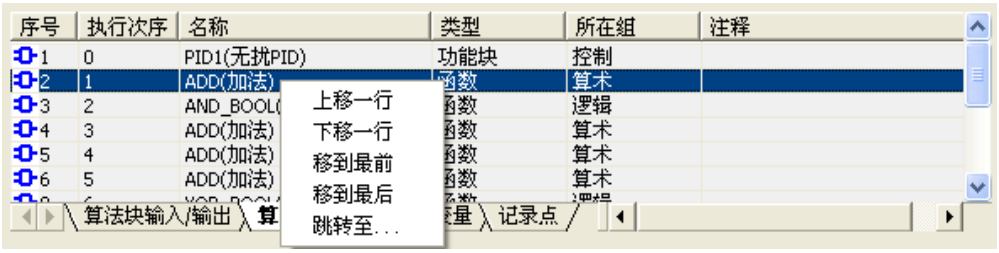


图 6- 30 算法块执行顺序设置

此外还可以通过快捷键的方式，Ctrl+↑等效为上移一行，Ctrl+↓等效为下移一行。同时也可以选择“跳转至...”菜单，在弹出的窗口中输入需要插入到的序号，点击“跳转”直接跳转到所需要的序号。

不管是那种排序方式，新插入的算法块都是最后一个执行。可以通过算法块执行窗口观察算法块的执行顺序，此外还可以通过右击菜单中选择“显示算法块执行序号”，使算法块的执行顺序显示出来。如图 6-31 所示。

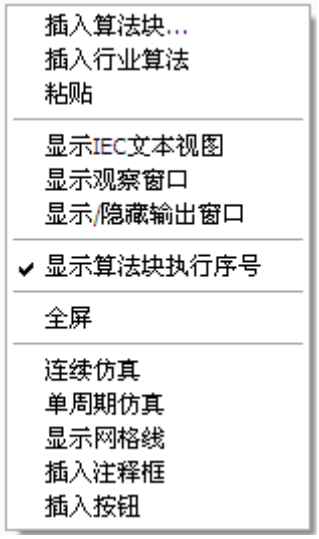


图 6- 31 算法块执行顺序设置

6.2. LD 梯形图编辑器

LD 梯形图编辑器多用于处理逻辑控制。LD 梯形图中有以下几种图形元素：网络标号、网络注释、网络头、触点（又分为常开触点、常闭触点、正跳变触点、负跳变触点）、线圈（又分为常开线圈、常闭线圈、置位线圈、复位线圈、正跳变线圈、负跳变线圈）、算法块。

6.2.1. 外观介绍

LD 梯形图编辑器如图 6-32 所示，梯形图编辑器可以分为观察窗口和编辑窗口两部分，其中观察窗口包括 3 个切换窗口：LD 图形元素列表窗口、局部变量窗口、记录点窗口。

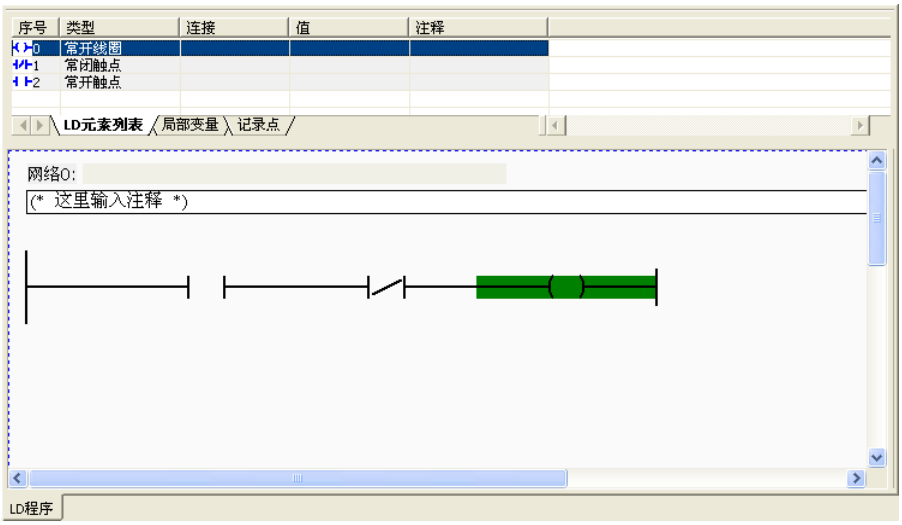


图 6- 32 LD 梯形图编辑器

6.2.1.1. LD 元素列表窗口

LD 元素列表窗口如图 6-33 所示。

图 6- 33 LD 元素列表窗口

对 LD 元素列表窗口各列的解释见表 6-6 所示。

表 6-5 LD 元素列表窗口

列	解释
第一列	元素的序号
第二列	元素类型
第三列	与变量或者算法块的连接。如果元素类型是触点或者线圈，则为连接的变量。如果元素类型是算法块，则为算法块的名称。
第四列	仿真时，连接变量的值。
第五列	对元素的注释

6.2.1.2. 局部变量窗口

参见 FBD 程序的局部变量窗口说明。

6.2.1.3. 记录点窗口

参见 FBD 程序的记录点窗口说明。

6.2.1.4. 编辑窗口

编辑窗口由若干个网络组成，一个网络相当于一个回路，如图 6-34 所示。每个网络由 3 部分组成：标号区、注释区、图形代码区。

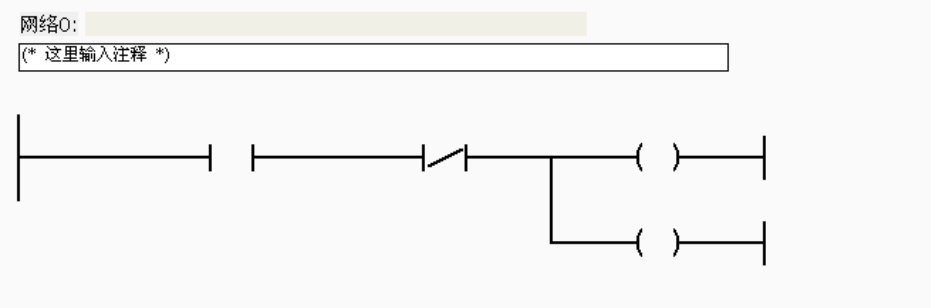


图 6- 34 编辑窗口

网络的最上面一行是标号区，可在这里编辑网络标号，用于标识此网络；

标号区下面是注释区，注释区可以输入对网络的注释。只要点击该区域，就会出现一个编辑框，可在此编辑框中输入文本。

注释区的下方是图形代码区域，一个 LD 网络由若干个触点、线圈或算法块组成，由触点的开关（通断）控制能量流向右流动，最右边一般是一个或几个并联的线圈，前面触点控制的能量流输入到连接的记录点，实现对现场位号的控制。

在 LD 网络中，还可插入 FBD 算法块，算法块的执行与否由其 EN 引脚决定。

与 FBD 功能块图不同的是，LD 中的算法块有个 ENO 引脚，用于能量流的传递，ENO 引脚的值总是等于 EN 引脚。不论算法块的输入引脚类型是数字型、整数型或者模拟型，则该引脚都可以与其左边的元素连接；如果算法块的输出引脚类型是数字型，则该引脚可以与其右边的元素连接。

6.2.2. 网络编辑

6.2.2.1. 插入网络

插入 LD 网络可能通过以下途径：

1. 选择主菜单“插入”下的“插入 LD 网络”；

2. 在画面上单击右键，在弹出的菜单中选择“插入网络”；

3. 点击工具栏中的“插入 LD 网络”按钮。

新网络插入在当前编辑网络的下面，此时缺省建立一个空的标号和注释。

6.2.2.2. 删除网络

删除网络有三种方法：

1. 选中网络最左边的垂直线，选择主菜单“插入”下的“删除 LD 网络”；

2. 选中网络最左边的垂直线，然后在这里单击右键，在弹出的右键菜单中选择“删除网

络”，如图 6-35 所示；

图 6- 35 删除网络

3. 选中网络最左边的垂直线，直接按下 Delete 键，这时将弹出一个确认对话框，按“是（Y）”后，程序将被删除，如图 6-36 所示。

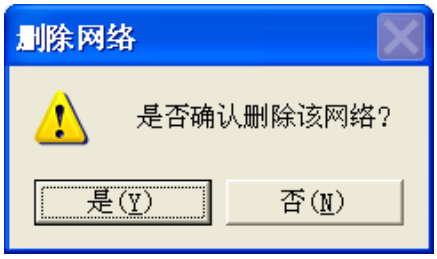


图 6- 36 删除网络确认对话框

注　意

当程序只剩下一个网络时，该网络不能删除。

6.2.3. 触点编辑

6.2.3.1. 触点类型

触点分为 4 种：分别为常开触点、常闭触点、正跳变触点、负跳变触点。如图 6-37 所示。

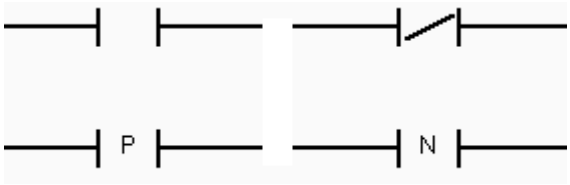


图 6- 37 触点类型

触点作为网络输入信号，一般要指定变量。

6.2.3.2. 插入触点

添加触点时要选中一个元素，然后在这个元素的右边或下边添加新触点。

具体方法如下：

1. 选中一个元素，选择主菜单“插入”下面的“插入常开触点”、“插入常闭触点”、“下插常开触点”、“下插常闭触点”；

2. 选中一个元素，根据需要点击工具条上的按钮“ ”、“ ”、“ ”或“ ”；

3. 选中一个元素，在此元素上单击右键，选择弹出菜单中的“右边插入常开触点”、“右边插入常闭触点”、“下插常开触点”或“下插常闭触点”；

触点加入后，网络将自动调整位置。

注　意

线圈的右边和下边不能插入触点；可以在触点、算法块和垂直线的右边插入触点；垂直线右边插入触点时，只能插入在右边最上边一行。

6.2.3.3. 删除触点

删除触点有如下四种方法：

1. 选中要删除的触点，选择主菜单“编辑”下面的“删除”；

2. 选中要删除的触点，按下工具条上的按钮“”；

3. 选中要删除的触点，在触点上单击右键，在弹出的菜单中选择“删除触点”，如图 6-38 所示；

图 6- 38 删除触点

4. 选中触点，按下 Delete 键；

5. 删除后，网络自动调整位置，被删除的触点的位置由右边或者下边的元素填充。

6.2.3.4. 触点的变量连接

触点连接变量有以下几种方法：

1. 在触点的上方两次单击鼠标左键，这时出现一个编辑框和列表窗口，如图 6-39 所示。

图 6- 39 触点变量连接

在列表框中选择记录点或局部变量，然后双击要连接的变量，编辑框和列表窗口消失，相应的变量名称出现在触点的上方编辑框的位置。

2. 采用拖放的方法，将变量到“拖”到触点上。

在观察窗口中的局部变量窗口中选中相应的局部变量，按住鼠标左键，拖动局部变量到触点上，可以实现局部变量与触点的连接

3. 用触点属性对话框选择，见下一节。

6.2.3.5. 触点的属性

编辑触点的属性可以用以下几种途径：

1. 双击触点，弹出触点属性对话框；

2. 选中触点，选择主菜单“元素对象”下面的“属性…”菜单项；

3. 在触点上单击鼠标右键，选择弹出菜单中的“查看触点属性”，如图 6-40 所示。

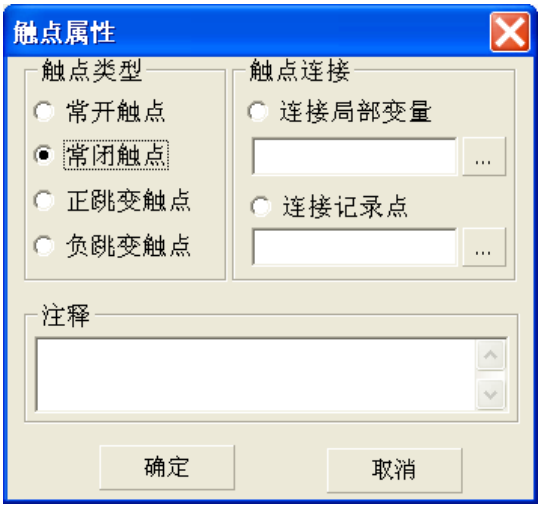


图 6- 40 触点属性

“触点类型”选项用于决定触点的类型：常开、常闭、正跳变或负跳变。

“触点连接”选项则首先确定决定触点状态的是局部变量还是记录点，然后再用按钮“…”选择具体的变量(不能手动输入变量名)。局部变量和记录点的选择对话框如图 6-41、6-42 所示。

图 6- 41 内部变量选择

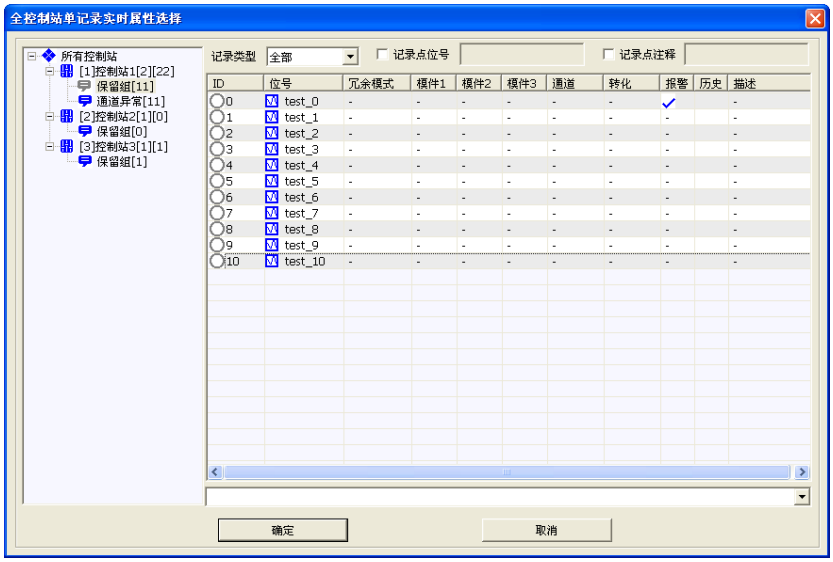


图 6- 42 记录点选择

6.2.3.6. 触点类型的转换

触点有几种类型，可以通过触点属性对话框进行类型转换，也可以通过以下几种方法：

1. 选中触点，选择主菜单“元素对象”下面的“类型转换”；

2. 选中触点，按下工具条上的按钮“ ”；

3. 选中触点，按下 Space 空格键，触点会在几种类型之间循环转换。

6.2.4. 线圈编辑

在 LD 中，共有 6 中线圈：常开线圈、常闭线圈、置位线圈、复位线圈、正跳变线圈、负跳变线圈。如图 6-43 所示。

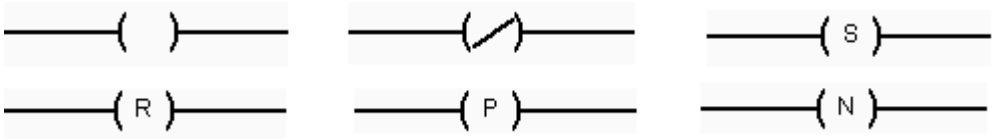


图 6- 43 线圈类型

线圈是网络的输出值，线圈连接变量，运行时，线圈将其输出值赋给变量。

- 常开线圈直接把左链路值赋给变量；
- 常闭线圈把左链路值取反后赋给变量；
- 正跳变线圈当左链路的连接值由 0 变 1 时，向变量赋值 1，其他情况为 0；
- 负跳变线圈当左链路的连接值由 1 变 0 时，向变量赋值 1，其他情况为 0；
- 置位线圈是当左链路为 0 时，向变量赋值 1，其他情况变量不被赋值；
- 复位线圈是当左链路为 1 时，向变量赋值 0，其他情况变量不被赋值。

6.2.4.1. 插入线圈

线圈可以插入的位置：

1. 触点的右边没有元素时，可以插入在这个触点的右边；

2. 算法块右边没有元素时，可以插入在这个算法块的右边；

3. 已经存在的线圈的下方。

线圈的右边不能再插入其他任何元素。插入线圈的具体方法如下：

1. 选中一个元素，选择主菜单“插入”下面的“插入常开线圈”；

2. 选中一个元素，选择工具条上的按钮“”；

3. 选中一个元素，在此元素上单击右键，在弹出的菜单中选择“右边插入线圈”。

6.2.4.2. 删除线圈

删除线圈有如下几种方法：

1. 选中要删除的线圈，选择主菜单“编辑”下面的“删除”；

2. 选中要删除的线圈，按下工具条上的按钮“”；

3. 选中要删除的线圈，在触点上单击右键，在弹出的菜单中选择“删除线圈”；

4. 选中线圈，按下 Delete 键。

6.2.4.3. 线圈的变量连接

线圈连接变量有以下几种方法：

1. 在触点的上方双击鼠标左键，这时出现一个编辑框和列表窗口，如图 6-44 所示；

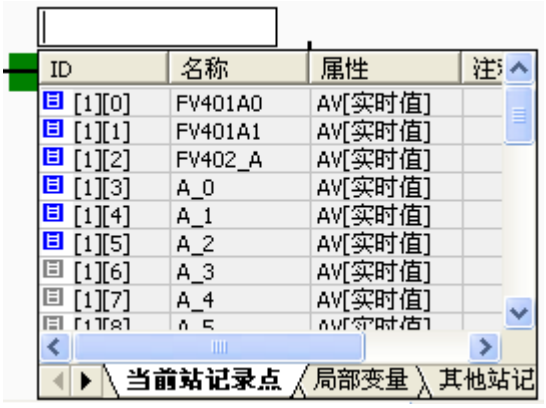


图 6- 44 线圈变量连接

在列表框中选择记录点或局部变量，然后双击要连接的变量，编辑框和列表窗口消失，相应的变量名称出现在线圈上方编辑框的位置。

2. 采用拖放的方法，将变量“拖”到线圈上；

在观察窗口中的局部变量窗口中选中相应的局部变量，按住鼠标左键，拖动局部变量到触点上，可以实现局部变量与触点的连接。

3. 用线圈属性对话框选择，参见下一节。

6.2.4.4. 线圈的属性

编辑线圈的属性可以用以下几种途径：

1. 双击线圈，弹出线圈属性对话框，如图 6-45 所示；

2. 选中线圈，选择主菜单“元素对象”下面的“属性…”菜单项；

3. 选中线圈，在触点上单击鼠标右键，在弹出的菜单中选择“查看线圈属性”。

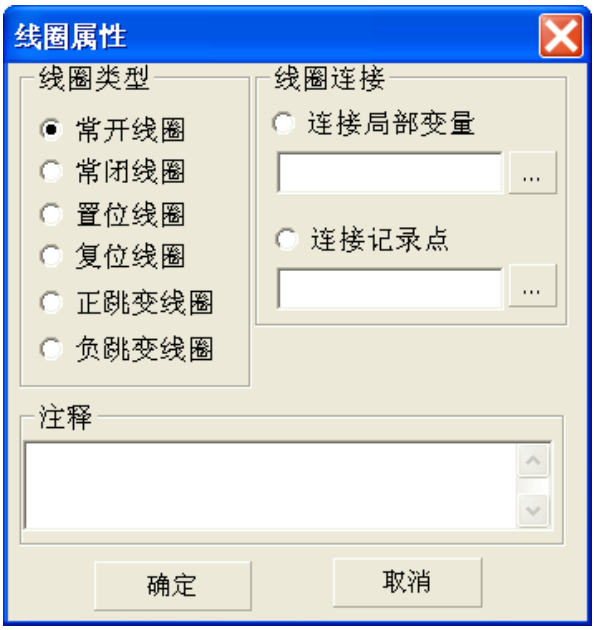


图 6- 45 线圈属性

“线圈类型”选项用于决定线圈的类型：常开、常闭、置位或复位等等。

“线圈连接”选项则首先确定连接线圈的是局部变量还是记录点，然后再用按钮“…”选择具体的变量（不能手动输入变量名）。其操作方法与触点连接类似，在此不再赘述。

6.2.4.5. 线圈类型的转换

线圈有几种类型，可以通过线圈属性对话框进行类型转换，也可以通过以下几种方法：

1. 选中线圈，选择主菜单“元素对象”下面的“类型转换”；
2. 选中线圈，按下工具条上的按钮“”；
3. 选中线圈，按下 Space 空格键，线圈会在几种类型之间循环转换。

注　意

有时如果中文输入法打开的情况下，Space 空格键可能会不起作用。

6.2.5. 算法块的编辑

在编程操作上，LD 中的算法块与 FBD 功能图中的算法块有若干不同的地方：

1．LD 中的算法块的引脚之间不能自由连线，只有相邻的算法块的引脚才可以连接；

2．LD 中的算法块有一个 ENO 引脚，用于传递能量流；

3．LD 中的算法块的位置由程序本身确定，不允许用户调节位置。

6.2.5.1. 插入算法块

插入算法块时要选中一个元素，然后在这个元素的右边或下边添加新算法块。具体方法如下：

1．选中一个元素，选择主菜单“插入”下面的“插入算法块”；

2．选中一个元素，选择工具条上的按钮“”；

3．选中一个元素，在此元素上单击右键，在弹出的菜单中选择“右边插入算法块”；

选择菜单或者工具条按钮后，出现一个算法块选择对话框，在这里选择要插入的算法块后，相应的算法块就插入到刚才选中元素的右边。

6.2.5.2. 删除算法块

删除算法块有如下几种方法：

1．选中要删除的算法块，选择主菜单“编辑”下面的“删除”；

2．选中要删除的算法块，按下工具条上的按钮“  ”；

3．选中要删除的算法块，在算法块上单击右键，在弹出的菜单中选择“删除算法块”；

4．选中算法块，按下 Delete 键。

6.2.5.3. 算法块引脚的连接

LD 网络中算法块引脚的连接与 FBD 图中的基本相同，请参考相关章节。

6.2.5.4. 算法块的属性设置

与 FBD 图中基本相同，请参考相关章节。

6.2.5.5. 算法块能量流的传递

算法块可以通过引脚将左边的能量流传递到右边，不一定非要用缺省的 EN/ENO 引脚，但要求连接的算法块输出引脚类型一定是数字型，如图 6-46 所示。

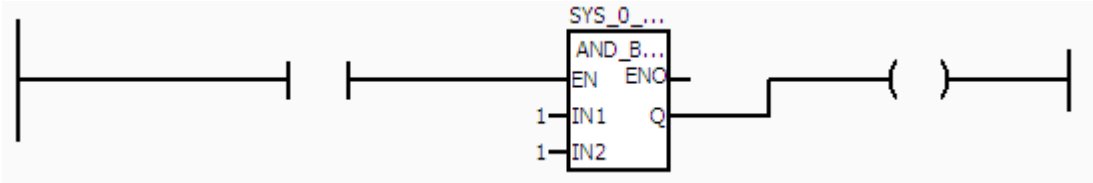


图 6- 46 算法块能量流的传递

6.3. SFC 顺控图编辑器

顺控图程序多用于处理顺序控制。

SFC 中主要有以下几种图形元素：起始步（Initial step）、步（Step）、转换（Transition）、选择分支（Divergence of sequence selection）、选择聚合（Convergence of sequence selection）、并行分支（Divergence of sequence simultaneous）、并行聚合（Convergence of sequence simultaneous）、跳转（Jump）。

6.3.1. 外观介绍

顺控图编辑器如图 6-47 所示。

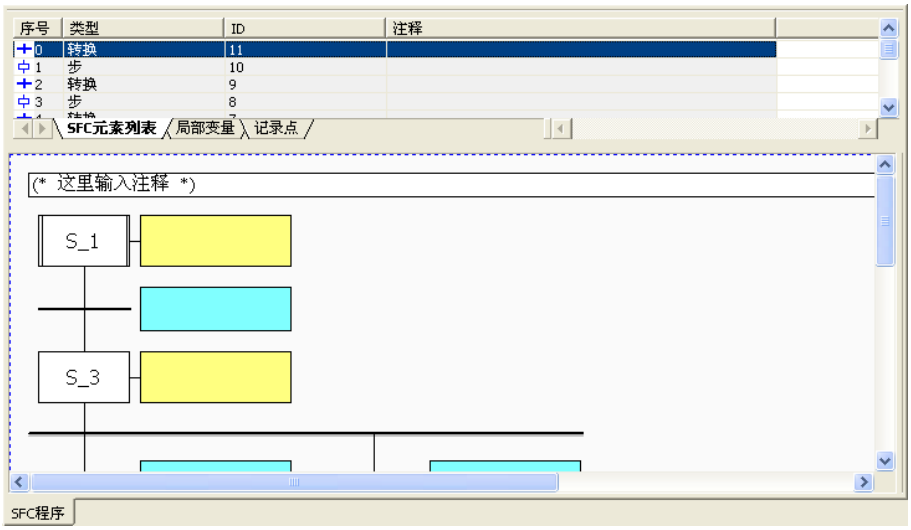


图 6- 47 顺控图编辑器

从图 6- 47 中可以看出，顺控图编辑器可以分为观察窗口和编辑窗口两部分。

6.3.1.1. SFC 图形元素列表窗口

该窗口的作用是显示当前程序编辑窗口中所有的图形元素的列表，如图 6-48 所示。

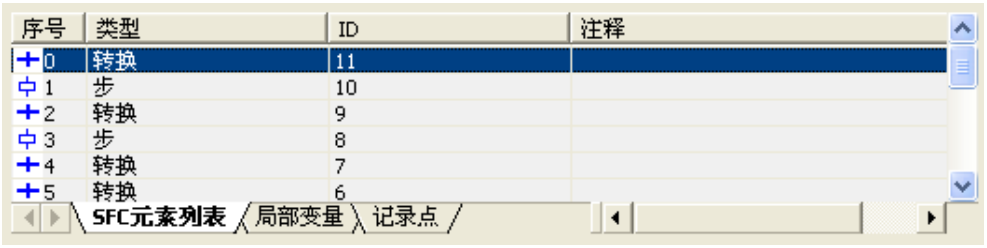


图 6- 48 SFC 图形元素列表窗口

对 SFC 元素列表窗口各列的解释见表 6-7 所示。

表 6-6 SFC 元素列表窗口内容

列	解释
第一列	元素的序号
第二列	元素类型
第三列	元素的 ID 号
第四列	对 SFC 元素的简单注释

6.3.1.2. 局部变量窗口

参见 FBD 程序的局部变量窗口说明。

6.3.1.3. 记录点窗口

参见 FBD 程序的记录点窗口说明。

6.3.1.4. 编辑窗口

编辑窗口由两部分组成：注释区和图形代码区，如图 6-49 所示。

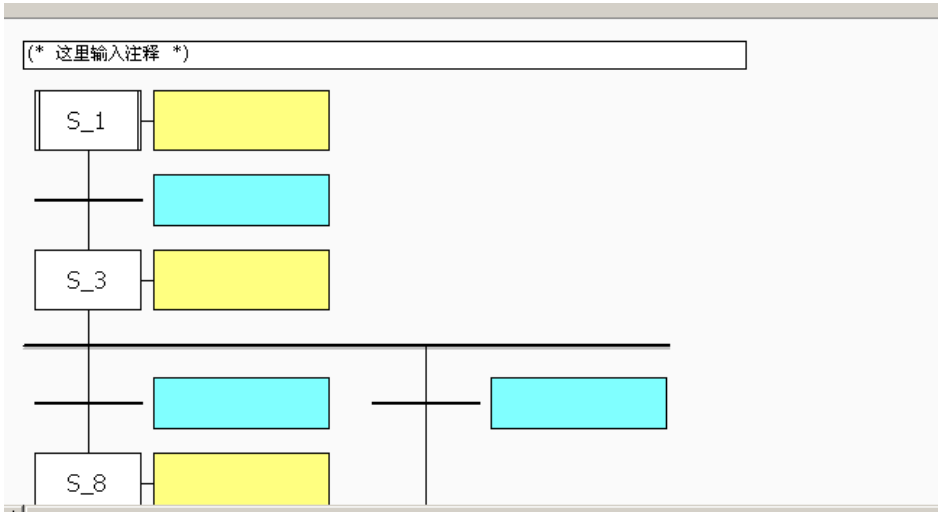


图 6- 49 SFC 编辑窗口

注释区可以输入注释，只要点击该区域，就会出现一个编辑框，可在此编辑框中输出文本。

注释区域的下方是图形代码区域。

6.3.2. 顺控图编辑

SFC 顺控图由步、转换、选择分支、选择聚合、跳转分支、并行分支和并行聚合组成。

步中含有若干个操作；每个操作执行一定动作；动作可以是一个变量，也可以是一个子程序。如果操作动作是一个变量，则变量的赋值由操作中的限定词来决定。如果调用是一个子程序，则子程序的执行由操作中的限定词来决定。如果不同的步中调用相同的子程序或变量名，则该子程序的执行与所有这些步的激活状态和限定词相关，具体运算规则参见顺控图的语法说明部分。

转换中包含表达式。当当前激活步下转换中的表达式条件满足时，激活其下的步，并使其上的步失去激活，开始执行激活步中的操作。

构建规则：

每个顺控图程序只有一个起始步，程序从这里开始执行；

步的下面只能接转换、选择分支线或者选择聚合线；

转换下面只能接步、跳转、并行分支线或并行聚合线；

并行分支线下面只能接步（可以有多个）；

并行聚合线下面只能接转换；

选择分支线下面只能接转换（可以有多个）；

选择聚合线下面只能接步、并行分支线或并行聚合线；

如果程序只执行一遍，最下面的元素必须是步；

- 如果程序需实现跳步或循环，可通过在转换条件下接一个跳转，在这个跳转中设定要跳到的步的名称，从而实现跳步或循环；
- 并行分支线下面有一个并行聚合线与其对应；同样，选择分支线下面也有一个选择聚合线与其对应；

并行分支线/并行聚合线对和选择分支线/选择聚合线对可以嵌套使用，对嵌套的次数没有限制。

6.3.3. 步的编辑

6.3.3.1. 插入步

插入步有以下几种方法：

1. 选择可以下插步的元素（见构建规则），选择主菜单“插入”下面的“插入步”；

2. 选择可以下插步的元素，点击工具条上的按钮“中”；

3. 在可以下插步的元素上单击鼠标右键，在弹出的菜单中选择“下插步（Step）”，如图 6-50 所示。

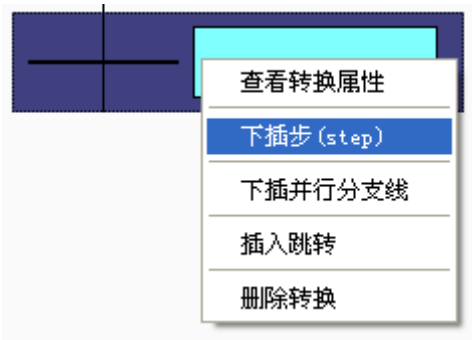


图 6- 50 下插步

6.3.3.2. 删除步

选中步，然后：

1. 选择主菜单“编辑”下面的“删除”；

2. 点击工具条上的按钮“×”；

3. 单击右键，在弹出的菜单上选择“删除步（step）”；

4. 按下 Delete 键。

6.3.3.3. 设置步的属性

在步上面双击，或者在右键菜单中选择“查看步属性”，会弹出如图 6-51 所示的对话框。

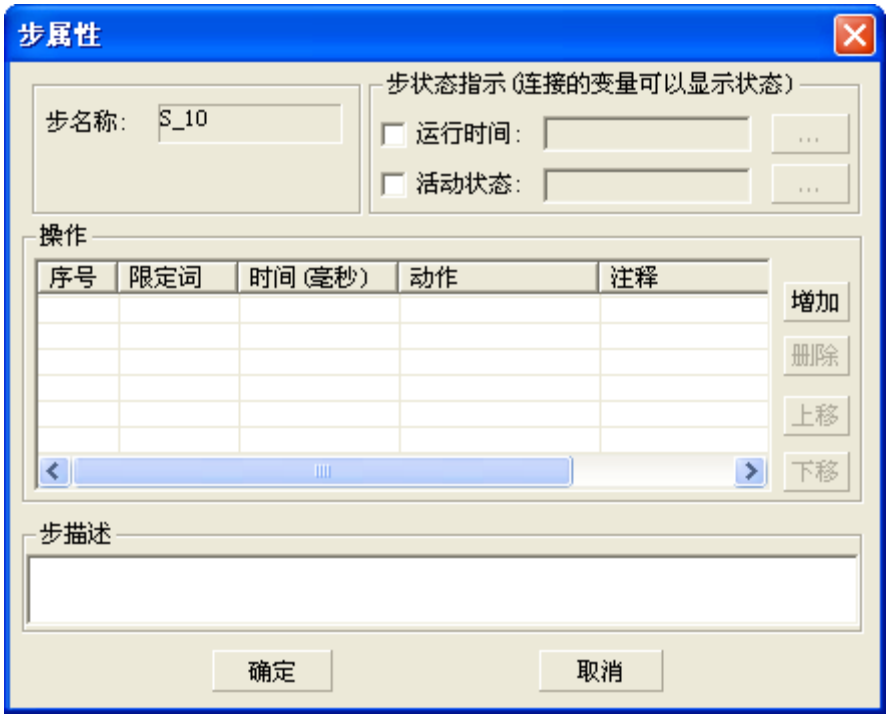


图 6- 51 步属性

图 6-51 对话框中列表控件右边的按钮用来对操作进行增加、删除或移动。

设置操作的限定词：在“限定词”这一列双击，将会弹出一个下拉列表框，可以在这里选择限定词，如图 6-52 所示。

图 6- 52 设置操作的限定词

有的限定词还需要和时间配合使用，在选择了限定词后，双击时间这一列，如果限定词需要与时间配合，可在时间编辑框内输入时间，时间单位是毫秒，要求为 50ms 的整数倍。

另外，在注释这一列双击，会有一个可编辑框，可以在这里输入此操作的注释。

增加一个操作后，要设置该操作的具体执行动作（这里称为调用），调用可以是变量，也可以是子程序。如果是连接的变量，则该变量的值等于该操作的激活状态。如果是子程序，则子程序的执行由该操作的激活状态来决定。在列表控件相应行的调用这一列双击，会弹出一个对话框，对调用进行设置。这个对话框用来选择变量或者子程序，如图 6-53 所示。

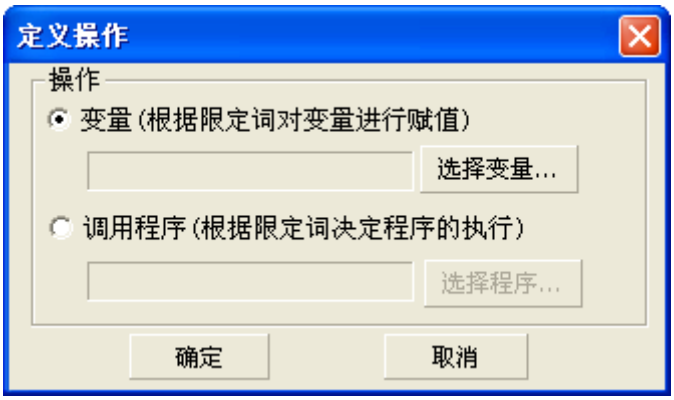


图 6- 53 定义操作对话框

在这个对话框中如果点击“选择变量…”，则弹出记录点选择对话框，选择记录点。如果点击“选择程序…”，则弹出子程序选择对话框，在这里选择要调用的子程序，如图 6-54 所示。

图 6- 54 选择程序

6.3.4. 转换的编辑

6.3.4.1. 插入转换

如果需要插入转换，可以：

1. 选择可以下插转换的元素（见构建规则），选择主菜单“插入”下面的“插入转换”；

2. 选择可以下插转换的元素，点击工具条上的按钮“ ”；

3. 在可以下插回去转换的元素上单击鼠标右键，在弹出的菜单中选择“下插转换(Transition)”，如图 6-55 所示。

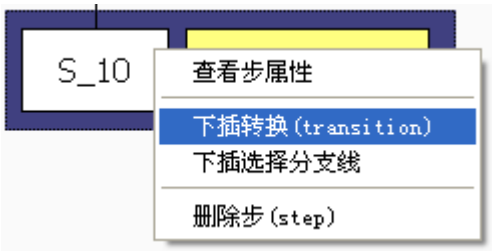


图 6- 55 插入转换

6.3.4.2. 删除转换

转换的删除方法是，先选中转换，然后：

1. 选择主菜单“编辑”下面的“删除”；

2. 点击工具条上的按钮“ ”；

3. 单击右键，在弹出的菜单上选择“删除转换”；

4. 按下 Delete 键。

6.3.4.3. 转换的属性设置

在转换上面双击，或者在右键菜单上选择“查看转换属性”，将弹出如图 6-56 所示的对话框。

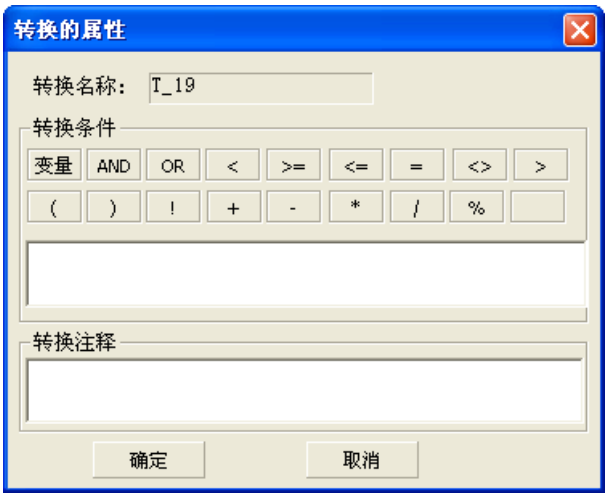


图 6- 56 转换属性

在上图的中间编辑框中输入转换条件，这是一个表达式。编辑框上方的按钮是用来进行自动输入的，只要点击按钮，该按钮代表的操作符就会插入到编辑框的当前光标处。当点击“变量”按钮时，会弹出一个记录点选择对话框，在这里选择记录点后，记录点名称自动插入到编辑框中。

在对话框上按下“确定”后，输入的转换条件显示在转换的右边文本框中。

转换条件表达式的语法可参见《ST 语言》这一章。

6.3.5. 选择分支/聚合线

与步和转换的插入/删除类似，可以通过主菜单、工具条和右键菜单进行插入和删除操作。

注　意

选择分支线的删除只能在它下面的所有元素都被删除之后才行。

6.3.6. 并行分支/聚合线

与步和转换的插入/删除类似，可以通过主菜单、工具条和右键菜单进行插入和删除操作。

注　意

并行分支线的删除只能在它下面接的所有元素都被删除之后才行。

6.4. ST 文本编辑器

ST 文本程序用高级文本语言编写。编写 ST 文本需要对高级语言编程有一定的了解，其语法规则请参考本帮助手册的相关章节。

ST 文本编辑器如图 6-57 所示。

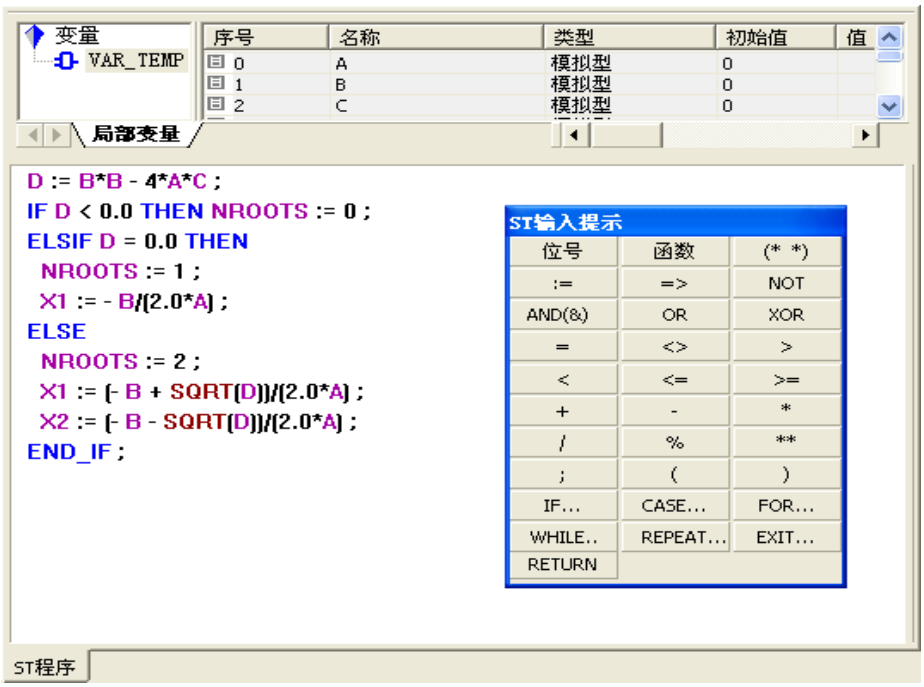


图 6- 57 ST 文本编辑器

6.4.1. 外观介绍

从图 6-57 中可以看出，ST 文本编辑器可以分为观察窗口和编辑窗口两部分。

观察窗口只包括一个子窗口：局部变量编辑窗口。该窗口的作用是编辑当前 ST 程序中使用的局部变量。关于局部变量窗口的描述请参阅 FBD

编辑窗口是一个文本编辑器。用户在这里输入表达式。

每个编辑窗口中都会出现一个输入提示对话框。可以在这里快捷输入。

6.4.2. 文本的输入

ST 语言的语法，请参考本帮助手册中相关章节。

6.4.2.1. 表达式的输入

除了控制语句，每个表达式以一个分号“;”结束。

对于文本中不同的语言符号，编辑器以一定的颜色区分。

6.4.2.2. 输入提示对话框的使用

点击输入提示对话框中的某一按钮，就可以在画面中当前光标所在位置插入相应的符号。

当点击“位号”时，会弹出一个记录点选择对话框，这里选择记录点后，记录点名称自动输入在当前光标所在位置。

当点击“函数”时，会弹出一个功能函数选择对话框，在这里选择功能函数后，函数自动输入在当前光标所在位置。

当点击“IF…”、“CASE…”等按钮时，这些控制语句的基本框架文本自动输入在当前光标所在的位置。

以“(*”、“*)”括号对括起来的文本作为注释，编译时，这些文本将被忽略。

6.4.2.3. 功能块、子程序和行业库的调用

在程序中如果要调用系统功能块，需要在局部变量编辑窗口声明一个类型是“功能块实例”的局部变量，并且选择相应的功能块。该变量的名称可被程序中引用。如果没有声明“系统功能块实例”变量，直接在程序文本中写功能块名称是非法的。例：在变量编辑窗口生成一个“rTRIG”变量，名称是“rTRIG_1”，则在程序文本中引用时，这样写：“rTRIG_1（VarIn, VarOut);”，这样写是不允许的：“rTRIG（VarIn, VarOut);”。

在程序中如果要调用子程序，也需要在局部变量编辑窗口声明类型是“子程序实例”的局部变量，并且选择相应的子程序，这个变量名称可在程序中使用，如果没有声明“子程序实例”变量，直接在程序文本中输入子程序名称是非法的。

在程序中如果要调用行业库，也需要在局部变量编辑窗口声明类型是“行业库实例”的局部变量，并且选择相应的行业库，这个变量名称可在程序中使用，如果没有声明“行业库实例”变量，直接在程序文本中输入行业库名称是非法的。

6.4.3. ST 数组功能使用

6.4.3.1. 功能描述

数组：在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来的一种形式。这些按序排列的同类数据元素的集合称为数组。

在ST语言中，可以使用数组功能的有记录点和结构体。

6.4.3.2. 表现形式

数组的表现形式：a[4]，名[下标]，下标为非负整数。

6.4.3.3. 功能使用

1、记录点中数组功能的使用

对于一组ID连续的记录点，如图6-58所示，记录点可以以ID比它小的记录点位号名 + [ID号的差]来表示，如记录点kai_4等价的表现形式kai_0[4]、kai_3[1]，记录点a 等价于 kai_4[2]。可以将一组连续ID的记录点当成数组来使用。

ID	位号	实时值	冗余模式	模件1	模
29	kai_0	0	-	-	
30	kai_1	0	-	-	
31	kai_2	0	-	-	
32	kai_3	0	-	-	
33	kai_4	0	-	-	
34	kai_5	0	-	-	
35	a	0.00000	-	-	
36	b	0.00000	-	-	

图 6- 58 记录点列表

2、结构体中数组功能的使用

一个结构体就相当于一个数组，结构体的引脚表现形式有两种：1、结构体名.引脚名（STR.Pin_4）2、结构体名[引脚序号]（STR[3]）。

6.4.3.4. 注意要点

数组的使用最容易出现的错误就是下标的使用

1、 下标只能是非负整数，负数和小数是非法的。

2、 下标使用注意禁止超范围使用，记录点的ID范围和结构体引脚序号范围。

6.5. IL 指令表编辑器

IL 指令表是文本形式的指令语言，由一行一行的指令文本构成。

6.5.1. 编辑器外观

IL 指令表编辑器如图 6-59 所示。

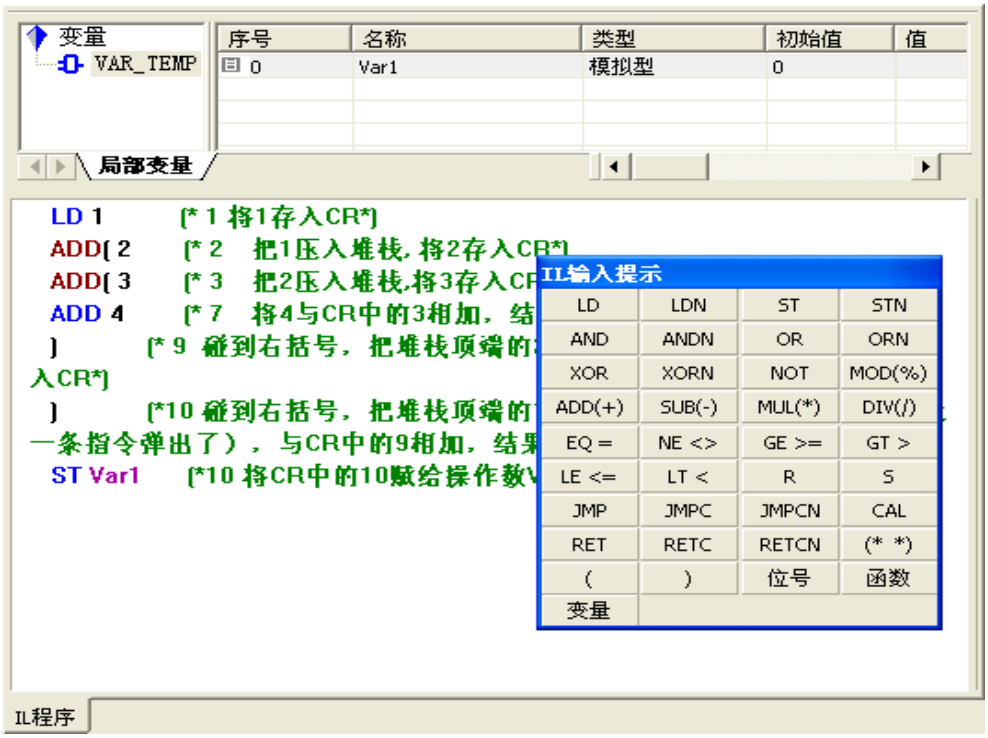


图 6- 59 IL 指令表编辑器

从图 6-59 中可以看出，IL 指令表编辑器可以分为观察窗口和文本编辑窗口两部分。

观察窗口包括一个子窗口：局部变量编辑窗口，该窗口的作用是编辑局部变量。

编辑窗口是一个文本编辑器。用户在这里一行一行的输入指令。每个编辑窗口中都会出现一个输入提示对话框。

6.5.2. 指令的编辑

IL 指令的语法，请参考本帮助手册中相关章节。

6.5.2.1. 指令输入

对于指令中不同的语言符号，编辑器以一定的颜色区分。

每条指令要占用一行。

如果指令要调用系统功能块，则先要在局部变量编辑窗口声明一个类型是“功能块实例”的变量，并且选择相应的功能块。指令中引用时，引用这个变量的名称。而不能直接使用系统功能块的名称。

如果指令要调用子程序，则先要在局部变量编辑窗口声明一个类型是“子程序实例”的变量，并且选择相应的子程序，指令中引用时，引用这个变量的名称，而不能直接使用子程序的名称。

如果指令要调用行业库，则先要在局部变量编辑窗口声明一个类型是“行业库实例”的变量，并且选择相应的行业库，指令中引用时，引用这个变量的名称，而不能直接使用行业库的名称。

以“(*”、“*)”括号对括起来的文本作为注释，编译时，这些文本被忽略。与 ST 文本不同的是，这里的注释文本必须在一行以内，不能分成多行写。

6.5.2.2. 输入提示对话框的使用

用户点击输入提示对话框中的某一按钮，就可以在画面中当前光标所在位置插入相应的符号或操作符。

6.5.2.3. 列表方式编辑指令

在“视图”菜单上选择“显示 IL 列表视图”可以转到 IL 列表视图上进行指令的编辑，在文本编辑视图中的文本自动转换成列表视图中的每一行。IL 列表视图如图 6-60 所示。

图 6- 60 IL 列表视图

在“标号”这一列双击，在文本框中输入行的标号。要注意的是，这里不能输入冒号“:”，转换成指令文本的时候，冒号会被自动加上。

在“操作符”这一列双击，将弹出一个下拉列表窗口，在这里选择操作符。如果选择“函数调用”，然后在别的地方点击，当下拉列表窗口消失的时候，则弹出一个对话框，用来选择系统功能函数。

在“操作数”这一列双击，将弹出一个编辑框，在这里输入指令的操作数。

在“注释”这一列双击，将弹出一个编辑框，在这里输入本条指令的注释。要注意的是，这里不需要输入“(*”和“*)”括号对。转换成指令文本的时候，注释括号会被自动加上。

在“视图”菜单上选择“显示 IL 文本视图”，就会转到 IL 文本视图进行编辑，在列表视图上的输入将会自动转换成一行一行的文本。

7 仿真

仿真是对控制工程现场实际运行的模拟。仿真时，可以用模拟现场类似的信号作为输入，也可以使用特定的信号作为对工程的测试信号，然后得到输出结果进行分析。同时也可以通过修改工程中的记录点和局部变量的值对系统进行局部或整体的测试。

仿真之前必须先编译程序，编译成功后，才能进行仿真。

仿真时必须至少有一个编辑窗口处于打开状态。

7.1. 连续仿真

选择“编译仿真”菜单下面的“连续仿真”项，或者点击工具条上的“”按钮。系统进入连续仿真状态。

如果想停止仿真，选择“编译仿真”菜单下的“停止连续仿真”项，或者点击工具条上的按钮“”，则系统停止仿真。

连续仿真时还可以暂时停止，选择“编译仿真”菜单下的“仿真暂停”，或者点击工具条上的按钮“”，则仿真会暂时停止。

连续仿真暂停后，选择“编译仿真”菜单下的“仿真继续”，或者点击工具条上的按钮“”，则仿真会继续下去。

7.2. 单周期仿真

选择“编译仿真”菜单下面的“单周期仿真”项，或者点击工具条上的按钮“”。系统进入单周期仿真。

单周期仿真时，系统每运行一遍，就会自动停止，要手动再重新继续。选择“编译仿真”菜单或右键菜单的“运行至下一周期”，或者点击工具条上的“”，都可使仿真再继续运行到下一个周期。

想停止仿真时，选择“编译仿真”菜单下的“停止单周期仿真”，或点击工具条上的“”，单周期仿真停止。

7.3. 仿真时变量值的修改

仿真时在以下几个地方双击，可以弹出对话框，修改变量的值：

1. 局部变量窗口相应的行；

2. FBD 图或者 LD 图中算法块的输入引脚；

3. LD 图中的触点。

如果变量类型是模拟型或整数型，则弹出如图 7-1 所示的对话框。

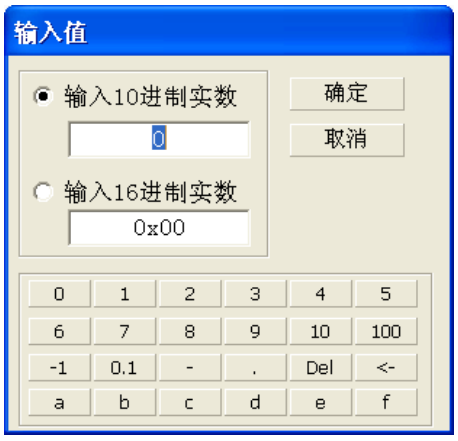


图 7- 1 实数变量输入对话框

如果变量类型是数字型，则弹出如图 7-2 所示的对话框：

图 7- 2 数字型变量输入对话框

7.4. 仿真设置

仿真分为当前任务仿真和整个工程仿真两种模式，当前任务仿真时，只有当前编辑窗口参与运算。整个工程仿真时，工程中所有的程序都参与运算。可以通过选择“编译仿真”菜单下面的“仿真整个工程”和“仿真当前任务”进行设置。

当前任务连续仿真时，可以设置仿真界面刷新周期，选择“编译仿真”菜单下面的“仿真时界面刷新周期”，弹出如图 7-3 对话框，在这里设置仿真界面刷新周期。

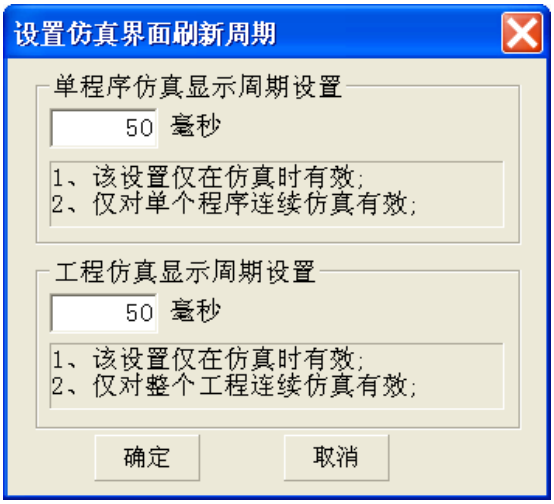


图 7- 3 仿真界面刷新周期设置

8 查找

当一个控制工程比较大的时候，记录点或算法块的使用情况比较复杂，这时就可以用查找功能来定位。

查找可以分为变量连接的查找、算法块使用的查找、子程序调用的查找。

8.1. 记录点的查找

记录点一般使用在以下这几个地方：

FBD 功能块图中算法块的引脚连接；

LD 梯形图中与算法块引脚连接、与触点连接、与线圈连接；

SFC 中作为步里面操作的动作调用；

ST 文本和 IL 文本中作为操作数；

可以选择“查找”菜单条上的“查找变量连接…”，此时弹出了一个对话框，如图 8-1 所示。然后点击右边的按钮“ … ”，弹出记录点选择对话框，如图 8-2 所示。

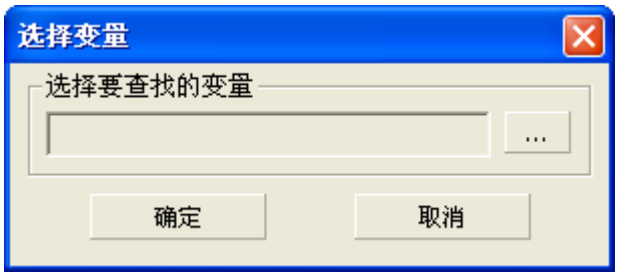


图 8- 1 查找变量

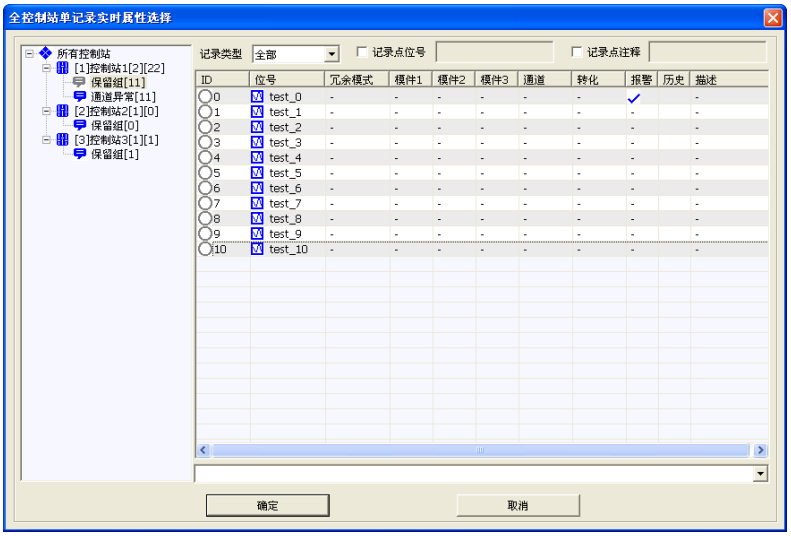


图 8- 2 选择记录点

如果找到变量连接，则在图 8-3 所示的输出窗口中出现以下字样。

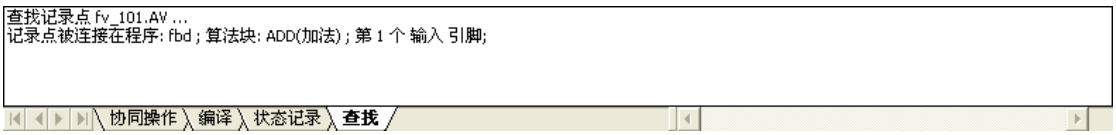


图 8- 3 查找变量输出窗口

在输出窗口中双击，就会打开相对应的程序编辑窗口，并且连接位置被选中。

8.2. 算法块使用情况的查找

算法块可以使用在：FBD 功能块图中、LD 梯形图中、ST 文本程序中、IL 指令表程序中。这里只能对前两种情况使用情况进行查找。

查找时，先转到导航栏内的“算法块”窗口，选中要查找的算法块，再在右键菜单中选择“查找算法块的使用”，如图 8-4 所示。

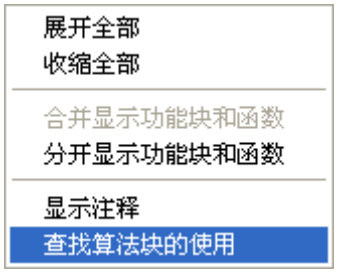


图 8- 4 查找算法块的使用

也可以选择“查找”菜单条上的“查找算法块的使用”，如果此时在导航栏内选择了算法块，则直接对该算法块进行查找，如果没有选中，则弹出了一个对话框，在这里选择要查找的算法块，如图 8-5 所示。

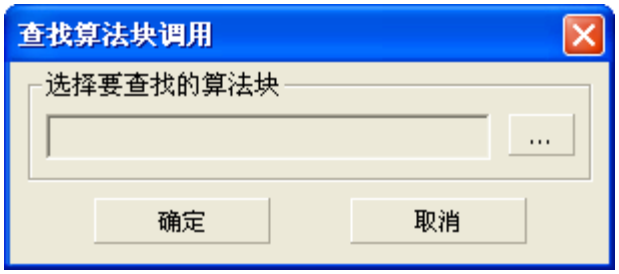


图 8- 5 算法块调用查找对话框

如果算法块被程序使用了，则弹出如图 8-6 所示的输出窗口。

图 8- 6 算法块调用输出窗口

在输出窗口双击某行，就会打开相应的程序编辑窗口。如图 8-6 中，若双击第二行，则打开程序 FBD 示例，且定位到序号 14 的算法块。

8.3. 子程序的使用情况查找

子程序可以使用在：FBD 功能块图、LD 梯形图、SFC 顺控图、ST 文本、IL 指令表。这里可以对前 3 种调用进行查找。

查找时，先转到导航栏内的“程序”窗口，选中要查找的子程序，在右键菜单中选择“查找子程序的调用”，如图 8-7 所示。

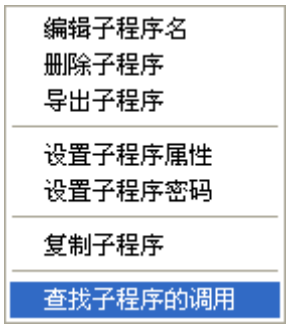


图 8- 7 查找子程序的调用

也可以选择“查找”菜单条上的“查找子程序的调用”，如果此时在导航栏内选择了子程序节点，则直接对该子程序进行查找，如果没有选中，则弹出了一个对话框，在这里选择要查找的子程序，如图 8-8 所示。

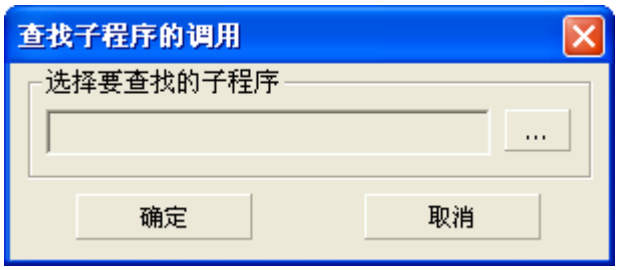


图 8- 8 子程序调用对话框

如果子程序被调用，则在下部的输出窗口中输出如图 8-9 所示的内容。

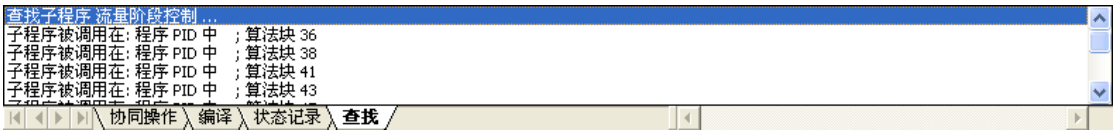


图 8- 9 子程序调用输出窗口

在输出窗口中双击，就会打开相应的程序编辑窗口。

9 算法在线编辑

算法编辑器 UWinIEC 及其编译、仿真运行软件，是支持图形化编程（功能块图 FBD、梯形图 LD、顺控语言 SFC）与文本编程（结构文本 ST、指令表 IL）及多语言混合编程的集成开发环境，支持控制算法的封装、继承、派生、复用，实现控制策略的在线编辑组态与离线/在线调试，提高编程效率，较传统编程模式工作量节约约 80%。

工业自动化现场的需求变化多端，控制策略的在线编辑组态功能可以满足用户在不中断现有系统运行的情况下，进行编辑修改（自动下载）与下载执行。系统提供的基本算法块是系统内不可分割的实现特定功能的算法模块，用户基于基本算法块组态的算法程序或者外部导入程序可以通过定义封装为系统的基本算法块。基本算法块完全封闭式运行，并且独立运行于其他基本算法块和算法程序，采用算法块分页、分组的在线调度与运行监视机制，单个算法块的运行错误不会影响到其他算法块的执行。

控制工程设计开发平台集成逻辑控制、连续控制、顺序控制为一体，支持控制策略的在线组态与在线调试，支持算法的单周期与单步调试；支持基于虚拟控制站的模拟仿真，用于测试控制策略。控制策略的在线组态应以独立运行的算法块为单位进行，而不仅仅是以算法程序页为单位进行增量编译，增量下装组态。以算法程序页为单位进行增量组态一定程度上解决了控制策略升级改造过程中停机重启问题，但无法解决对现有工艺局部小规模升级变动的停机重启问题。因此，研究如何从根本上解决升级控制策略过程中造成工程现场停机重启

具有很重要工程应用价值。控制策略算法在线组态如图 9-1、9-2 所示。

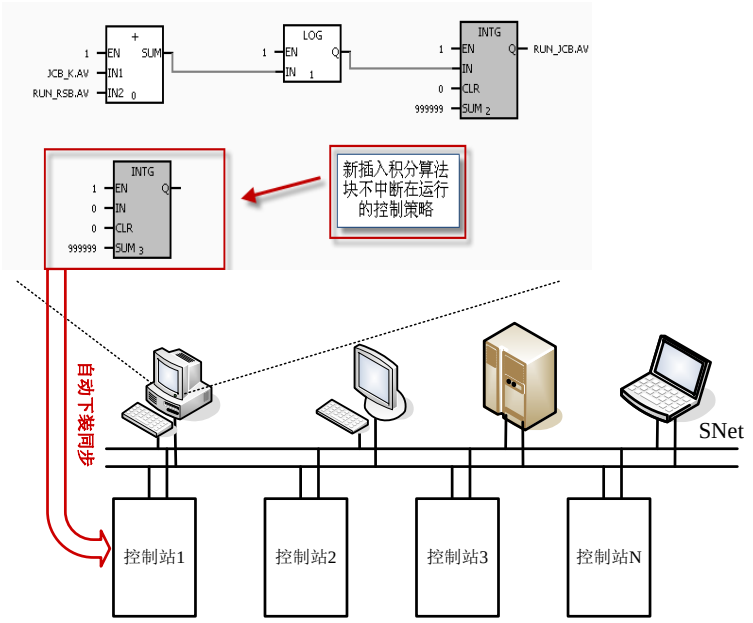


图 9- 1 控制策略算法在线组态功能示意图（一）

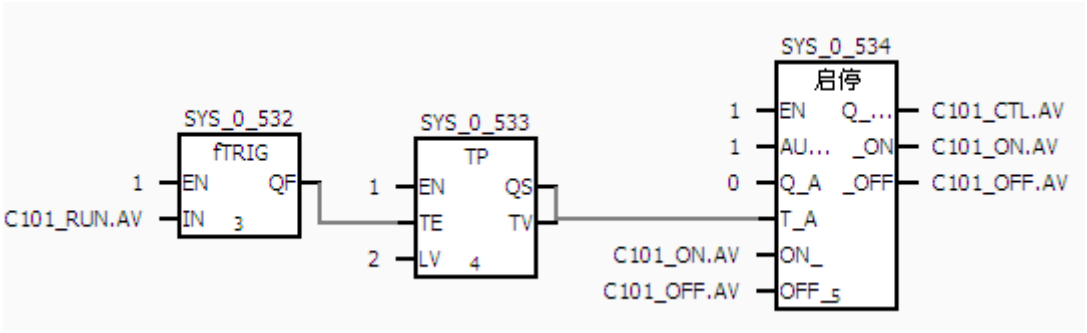


图 9- 2 控制策略算法在线组态功能示意图（二）

在现有的 5 种程序中，FBD 程序和 LD 程序支持算法的在线组态功能。如上所述，用户可以进行在线编辑，但前提是必须先编译工程，编译成功后，再下装算法，且算法的操作站版本号与控制站版本号一致，才能进行在线编辑。点击“ ”按钮，即进入在线编辑状态，之后进行的操作都会自动下装到控制模件中。若用户要取消在线编辑，则再点击“ ”按钮即可。

语言说明

1. FBD 语言

1.1. 概述

术语定义：

基本函数块：在算法编辑器中将由程序内嵌的系统函数块称为基本函数块。

基本功能块：在算法编辑器中将由程序内嵌的系统功能块称为基本功能块。

调用功能块：在算法编辑器中将由 CAL 功能块调用子程序所生成的实例功能块称为调用功能块。

FBD：在算法编辑器中将由基本函数块、基本功能块、调用功能块统称为 FBD，也称为算法块。

局部变量：在算法编辑器中将在每个程序中定义的变量称为局部变量。

记录点：在算法编辑器中将由实时数据库编辑器定义的变量称为记录点。

根据 IEC61131-3，FBD 编辑器将算法块和变量(记录点、内部变量)组成功能块图(FBD)。图形内可以自由放置算法块，并可通过连线、内部变量、记录点等多种方式建立连接关系，如图 10-1 所示。

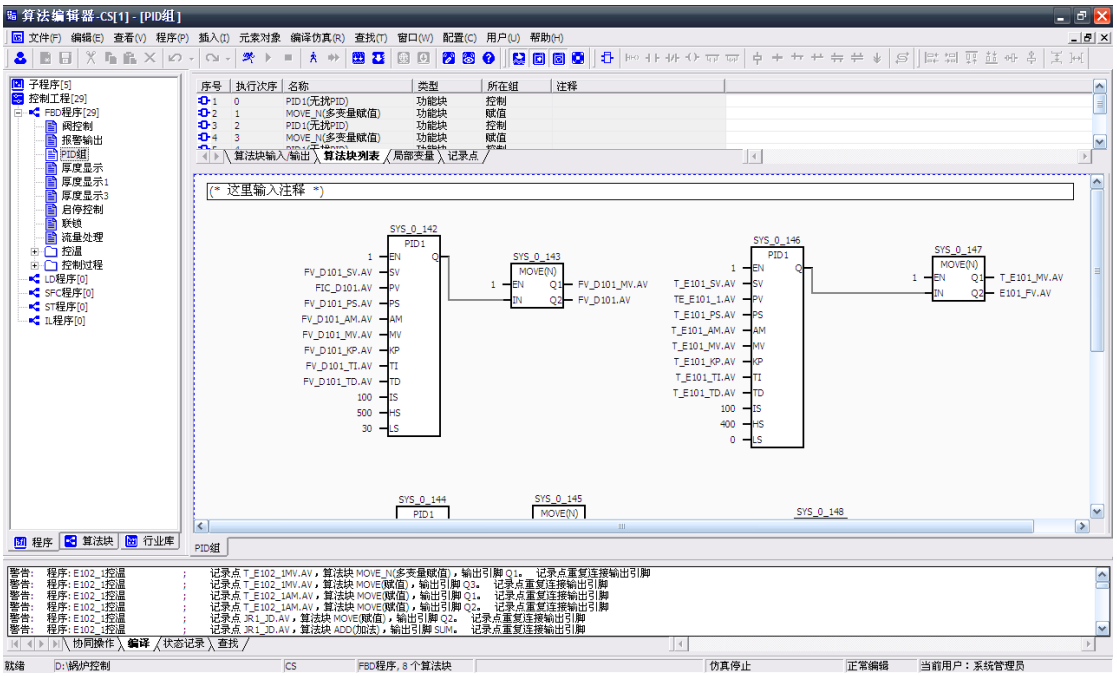


图 10- 1FBD 功能块图

每个算法块都可以用一个 EN 输入进行配置，该输入可以控制该算法块是否执行运算和算法块是否输出。如果当调用某算法块时 EN 值等于 0，则由该算法块定义的算法将不被执行，其输出值也不会写到其所连接的变量。如果当调用算法块的 EN 值等于 1，则由该算法块定义的算法将被执行，其输出值也会被写到其所连接的变量。

算法块用带有输入和输出的图形框来描绘:输入在图形框的左边,输出在图形框的右边。算法块的名称在图形框的中间显示。EN 是每个算法块的缺省的第 0 个输入。

1.2. 算法块

算法编辑器提供了一个系统功能块库和一个系统函数库。函数和功能块的区别在于，函数没有状态变量，其相同的输入值总会产生相同的输出结果。而功能块存在状态变量，相同的输入值不一定产生相同的输出结果。因而在功能块被插入时，必须为其申明一个实例，以保持其内部状态变量，而函数却不需要申明实例，具体申明过程在 FBD 被插入时，算法编辑器已作处理，用户无须关心。

在算法编辑器中所有的子程序都被定义为功能块，也就是说，在算法编辑器中用户通过 CAL 功能块调用子程序时,将会为每个子程序申明一个实例。如果用户在一个程序中利用 CAL 功能块多次调用同一个子程序时，每个被调用的子程序独自拥有自己的一片数据空间，相互不影响。

基本功能块和基本函数块的功能在后续章节中将作详细介绍。

1.3. 连接

连接是算法块之间以及算法块与变量之间的关系。一个算法块输出可以连接多个算法块的输入，同时该算法块的输出也可以连接内部变量或记录点，但该输出不能同时既连接记录点又连接记录点。一个算法块输入可以连接某个算法块的输出或记录点或内部变量，但只允许连接其中一个属性。

在算法编辑器中用户可以对每个算法块的数字型输入引脚设置取反属性，当设置取反时，则 FBD 程序将该引脚的值取反，作为该算法块的输入参数。

连接不能用于闭环的配置，因为闭环连接不能清楚地确定区段循环中的执行次序，闭环必须通过添加变量来解决。通过连接中间变量的方式，我们可以确定了该任务中各 FBD 的运算顺序。

1.4. 执行次序

在 FBD 程序内那些输入只连接变量或常数的算法块，被称为程序起始模块。程序内有多个起始模块时，在程序中最先添加的起始模块被称为启动模块;

在 FBD 中将那些在当前周期输入已被运算过的模块称为可执行模块。所有的起始模块肯定是可执行模块。

程序的执行次序如下:

运算启动模块;

计算在当前周期未被运算的可执行模块;

在该步中当前未被运算的可执行模块的运算顺序，没有严格的限制，根据用户的不同编辑情况，其运算顺序会发生改变;

若所有的算法块均已被运算，则当前周期运算结束，否则转到 4;

扫描所有未被执行的算法块，判断哪些块是可执行模块；返回 2。

举例，如图 10-2 所示。

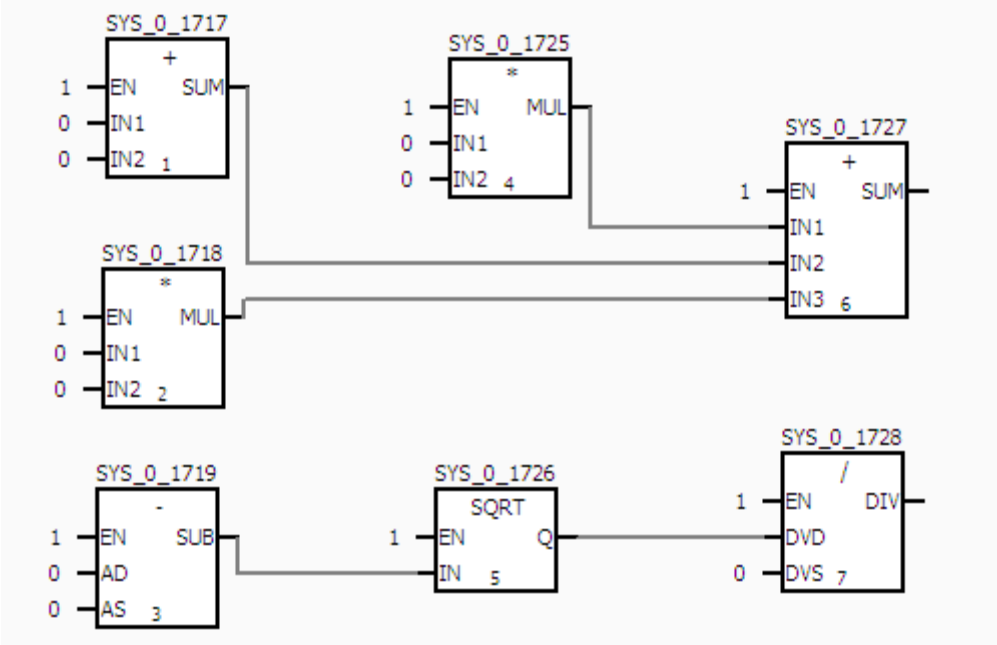


图 10- 2 算法块连接

1.5. 仿真

FBD 程序仿真支持单周期仿真和连续仿真。在仿真前必须先编译，若在程序更改后，未作编译而直接仿真，则程序不做任何操作，用户必须先编译程序才能进行仿真。

假设做好如图 10-3 所示的 FBD 程序。

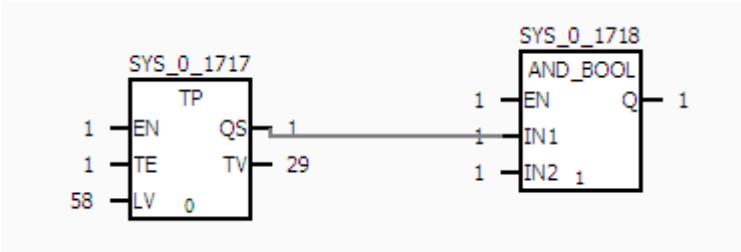


图 10- 3 仿真过程

点击单步仿真按钮，点击 LV 引脚，在弹出的对话框中输入值，则发现 PT 值变为输入值，选择继续下一步仿真，用户可判断输出是否符合该算法。

2. LD 语言

2.1. 概述

根据 IEC61131-3 标准，LD 编辑器将算法块、线圈、触点和变量组成梯形图(LD)。图形内可以自由放置基本元素和注释文本，使用梯形图可以方便的构成顺序和联锁系统。

LD 程序的设计对应于继电器开关的梯级。图形的左边是汇流条，相应于梯级的相线。只有直接或间接与相线有开关量相连的元素在编程期间才会被扫描。

当插入触点或线圈时，程序根据当前选中的触点的位置按照插入串联触点在右侧，并联触点在下方的原则插入。

在每一个 LD 网络中，线圈肯定在最右方，在同一个网络中允许存在多个线圈。

2.2. 触点

触点是 LD 元素，它将状态传送至其右侧的水平链路。这一状态是在其左侧的水平链路中的状态与相关变量的状态进行布尔操作的结果，触点不改变相关变量的值。

2.2.1 常开触点

在常开触点中，如果相关连接变量的状态为 ON 时，左链路的状态复制到右链路，否则的话，右链路的状态为 OFF。

图 10-4 用梯形图和功能块图的方法描述了常开触点。

图 10- 4 常开触点

2.2.2 常闭触点

在常闭触点中，如果相关连接变量的状态为 OFF 时，左链路的状态复制到右链路。否则的话，右链路的状态为 OFF。

常闭触点对应于含两个输入的 AND_BOOL 功能，其中一个输入是反相的，如图 10-5 所示。

图 10- 5 常闭触点

2.2.3 正跳变触点

在正跳变触点中，如果相关连接变量的状态从 OFF 跳变到 ON 时，同时左链路的状态为 ON 的话，则右链路在当前程序周期为 ON，否则的话，右链路的状态为 OFF，如图 10-6 所示。

图 10- 6 正跳变触点

2.2.4 负跳变触点

在负跳变触点中，如果相关连接变量的状态从 ON 跳变到 OFF 时，同时左链路的状态为 ON 的话，则右链路在下一个程序周期为 ON，否则的话，右链路的状态为 OFF，如图 10-7 所示。

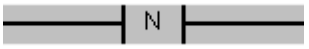


图 10- 7 负跳变触点

2.3. 线圈

线圈是 LD 元素，它将其左侧的水平链路状态传送至其右侧的水平链路，相关变量的状态将保存，线圈通常跟在触点之后。

2.3.1 常开线圈

在常开线圈中左链路的状态复制至相连接的变量和右链路，如图 10-8 所示。

图 10- 8 常开线圈

2.3.2 常闭线圈

在常闭线圈中，左链路的状态复制到右链路，左链路的取反状态复制至相连接的变量，如图 10-9 所示。

图 10- 9 常闭线圈

2.3.3 置位线圈

在置位线圈中，左链路的状态复制到右链路，如果左链路的状态为 ON，则相连接的变量为 ON，否则对连接变量不执行任何动作。相连接变量的状态可以通过复位线圈来复位，如图 10-10 所示。

图 10- 10 置位线圈

2.3.4 复位线圈

在复位线圈中，左链路的状态复制到右链路，如果左链路的状态为 ON，则相连接的变量为 OFF，否则对连接变量不执行任何动作，它可以通过置位线圈来置位，如图 10-11 所示。

图 10- 11 复位线圈

2.3.5 正跳变线圈

在正跳变线圈中，左链路的状态复制至右链路。如果左链路的状态从 OFF 跳变到 ON，则相连接的变量在当前程序周期为 ON，否则状态为 OFF，如图 10-12 所示。

图 10- 12 复位线圈

2.3.6 负跳变线圈

在负跳变线圈中，左链路的状态复制至右链路。如果左链路的状态从 ON 跳变到 OFF，则相连接的变量在当前程序周期为 ON，否则状态为 OFF，如图 10-13 所示。

图 10- 13 复位线圈

2.4. 算法块

在 LD 编辑器中，用户可以插入算法编辑器中的各类算法块，每个插入的算法块均有一个 EN 输入和一个 ENO 输出。如果当调用算法块的 EN 值为 OFF，则由该算法块定义的算法将不被执行，ENO 值自动设置为 OFF。如果当调用算法块 EN 值为 ON 时，则由该算法块定义的算法将被执行，算法执行完成后，ENO 值自动设置为 ON。

在 LD 语言中，算法块的调用与在 FBD 中的调用方式完全相同。

执行次序

在 LD 程序中，执行程序按照从左到右，从上到下的原则进行运算。

2.5. 仿真

LD 程序仿真支持单周期仿真和连续仿真。在仿真前必须先编译，若在程序更改后，未作编译而直接仿真，则程序不做任何操作，用户必须先编译程序才能进行仿真。

假设做好如图 10-14 所示的 LD 程序，

点击连续仿真按钮，测试程序是否按照设定的逻辑进行运算输出。

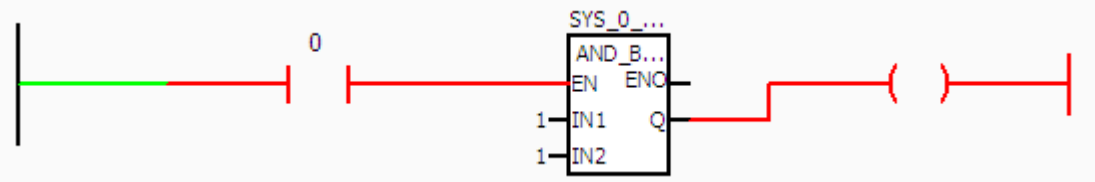


图 10- 14 仿真

3. SFC 语言

3.1. 概述

根据 IEC61131-3 标准，SFC 语言规定将复杂的程序分割为较小的可管理的单元，并描述在这些单元之间的控制流。使用 SFC 语言，可以设计顺序和并行过程。

执行这些单元的时序取决于静态条件和动态条件。

一步接着一步地处理生产过程地行为特性特别适用于 SFC 语言。

SFC 用步和转换条件构成程序段，步中通过定义操作实现对流程的操纵，通过转换实现流程的按顺序前进。

在 SFC 编辑器中，当选中某个元素时，程序在工具栏中指示何种元素可以被插入。

3.2. 步

步是控制流程中相对独立的一组操作的集合。在步中可以定义任意数目的各种类型的操作，通过操作实现对流程的控制。

一个步可以是激活状态或不激活状态。步在紧接在前的转换条件满足时激活，步在紧接在后的转换条件满足时退出激活状态。每个 SFC 程序有一个起始步，该步在第一次执行时默认为激活状态，其余的非起始步默认为不激活状态。

步的上面只能接转换、并行分支或选择聚合。步的下面只能接转换、并行聚合或选择分支。

步有两种类型：起始步和普通步。

起始步在程序刚启动时就处于激活状态，然后程序将按照转换条件的变化按照顺序依次执行，如图 11-1 所示。

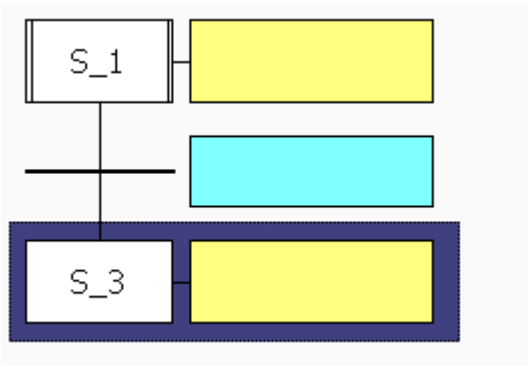


图 11- 1 顺控图

用户可以在步的属性对话框中通过增加或删除按钮来添加或删除该步中所执行的操作序列。

索引：索引是算法编辑器程序中表征某个步的标识。

限定词：限定词指定操作类型。每个限定词的含义将在下面予以介绍。

时间：某些限定词可能需要提供时间参数。

动作：通过动作来选择该操作所执行的子程序或记录点。

注释：注释是该动作的描述语言。

3.3. 转换条件

转换是控制从一个步转移到其他步的条件。

当转换条件满足时，紧接在前的步从激活态变成不激活态。然后紧接在后的步将从不激活态转变成激活态。

只有当所有紧接在前的步的状态都在激活状态时，转换条件才被测试。

转换条件由一个变量或一个表达式来定义。

转换的上面只能接步、选择分支、并行接合；转换的下面只能接步、选择聚合、并行分支、或跳转分支。

选择分支

选择分支提供了在 SFC 程序中实现条件控制的控制流程选择执行的方法。

在选择分支结构中只能有一个分支被激活。

分支跳转的优先级从左到右。

选择分支和选择聚合必须一一对应。

选择分支必须结束于同一选择聚合。

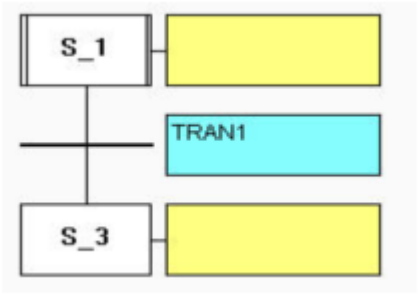
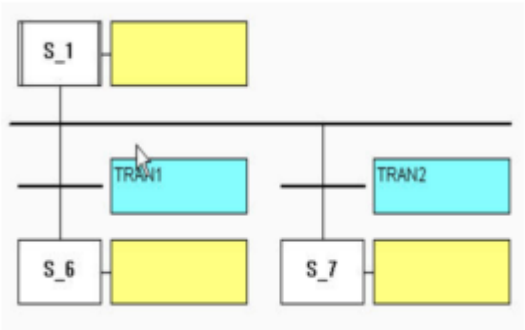
并行分支

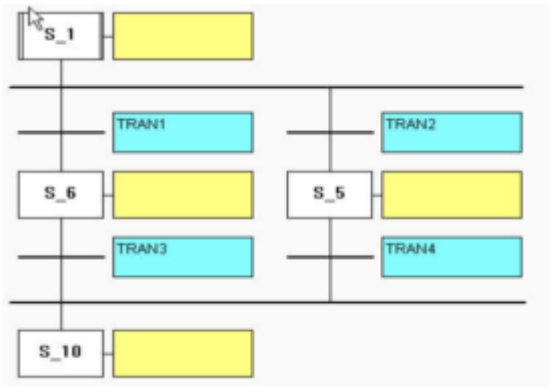
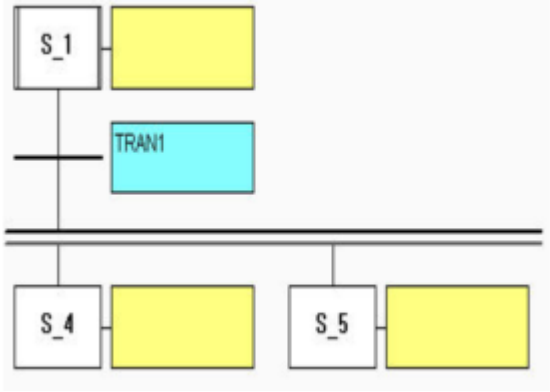
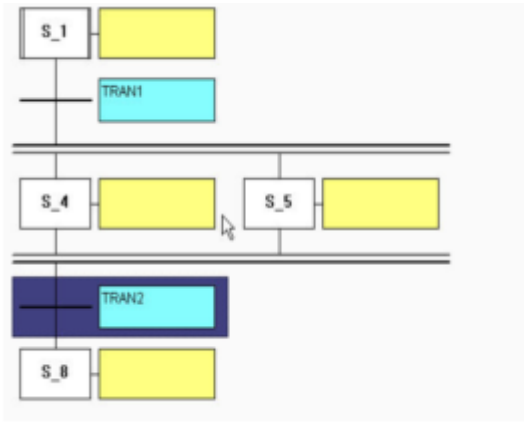
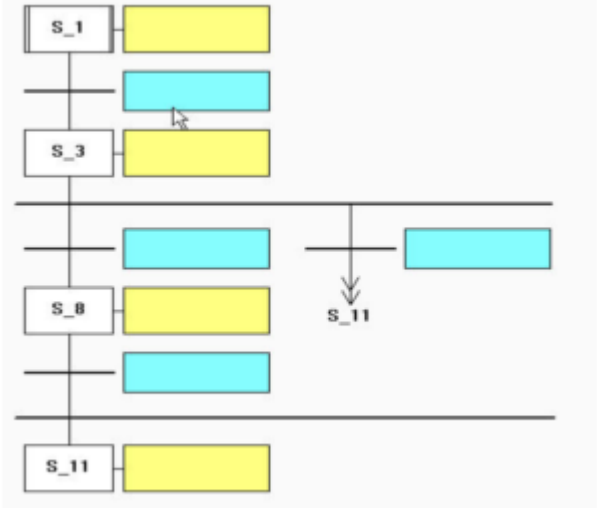
并行分支使流程中的几个子流程同时进行，各分支的执行同时进行，不相互影响，只有当所有的分支的最后一步都激活时，才测试并行聚合紧接的转换的条件是否满足。

并行分支和并行聚合必须一一对应。

表 11-1 列出算法编辑器所支持的流程控制规则：

表 11- 1 流程控制规则

图形对象	名称和解释
	单一顺序： 步的交替—>转换—>串行的步 一旦 TRAN1 所求的值为 TRUE，则解除激活 S_1，同时 S_3 变为活动状态。
	选择分支： 只选择一个顺序，从左到右进行计算； 只要 S_1 处于活动状态，就从左到右对转换进行求值，最先具有 TRUE 的转换停止 S_1 步的执行，并激活其所关联的后继步。

	选择聚合： 组合选择分支。当S_6和S_5中的一个处于活动状态，相应的后继转换条件也变为 TRUE 时激活S_10，同时其本身解除激活。
	并行分支： 同时激活所有被连接的步。 当TRAN1所计算的值为 TRUE 时，解除激活 S_1，而所有通过TRAN1连接的后继步均处于活动状态。这些新激活的步同时运行。
	并行聚合： 并行分支的路径被重新聚集在一起。 当 S_4 和 S_5 处于活动状态以及对应的转换条件 TRAN2 所求的值为 TRUE 时，S_4 和 S_5 步解除激活，并激活S_8。
	跳转分支： 可以通过跳转分支实现跳转到任意指定的步。这里让其跳到 S_11。 顺序的选择与选择分支的路径描述相同。

3.4. 操作

操作是对系统变量进行的操纵的描述。

一个步中可以有 0 个或多个操作。操作有多种类型，操作类型由操作限定词和动作来描述。动作可以是一个记录点，也可以是调用一个子程序。

在步属性窗口内可以编辑操作。

一个操作块包含一个动作连同其执行的条件（称为动作限定词）。

系统对步的激活和解除激活期间所有步的动作块的执行条件进行监视。

在算法编辑器中提供以下几个符合 IEC1131-3 标准的限定词：

N 动作在步的整个激活期间激活，随着步退出激活状态恢复成不激活状态，如图 11-2 所示。

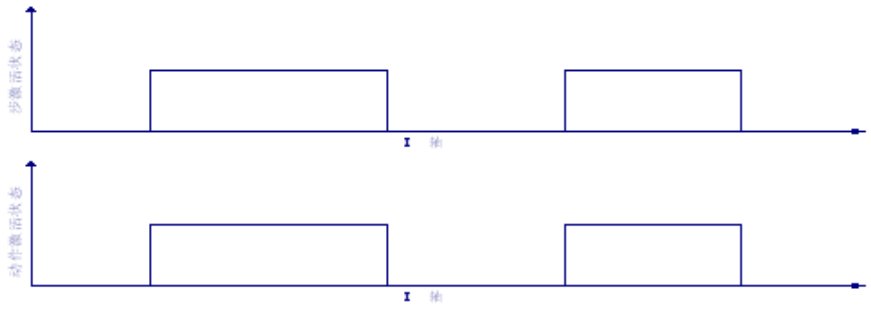


图 11- 2 N 动作状态

S 动作在步激活后将一直保持激活，如图 11-3 所示。

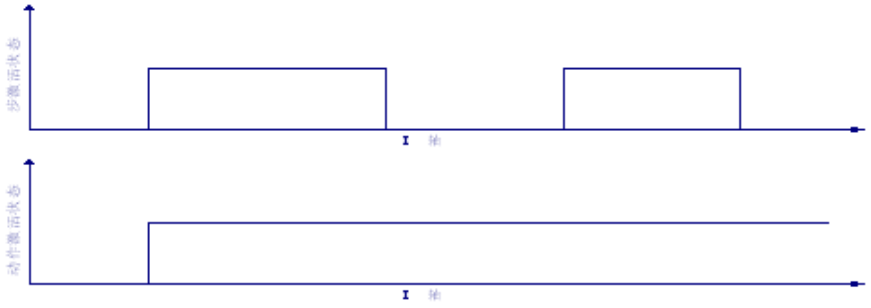


图 11- 3 S 动作状态

R 动作在步激活后将一直保持在不激活状态，如图 11-4 所示。

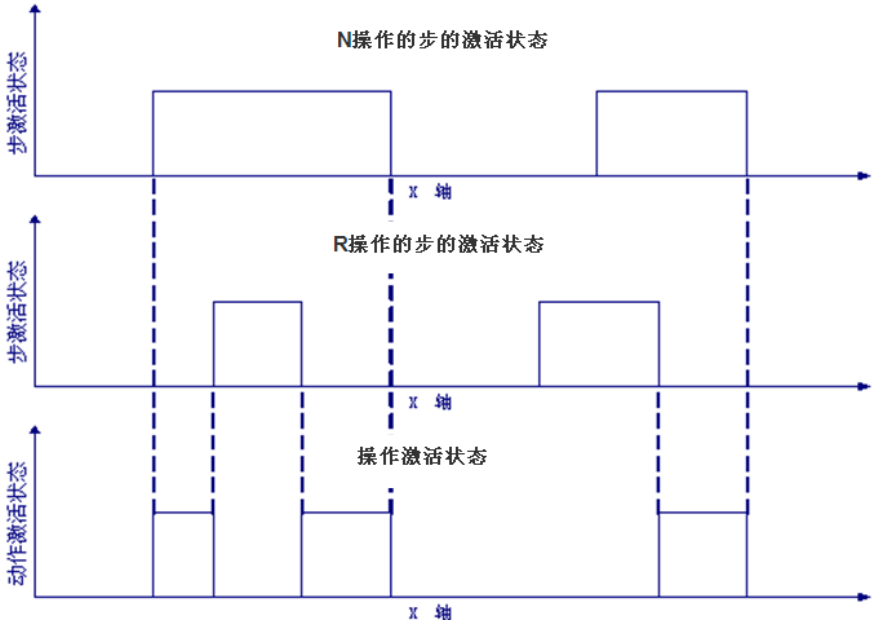


图 11- 4 R 动作状态

L 动作在步激活后在限定的时间内保持激活，超出时间恢复成不激活状态，若在此期间步失去激活，则该动作也变为不激活状态，如图 11-5 所示。

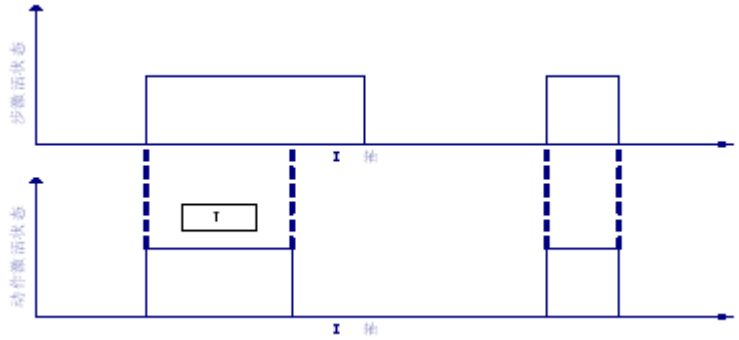


图 11- 5 L 动作状态

D 动作在步激活后经过限定的时间后，变为激活状态，随着步变成不激活状态，操作恢复成不激活，如图 11-6 所示。

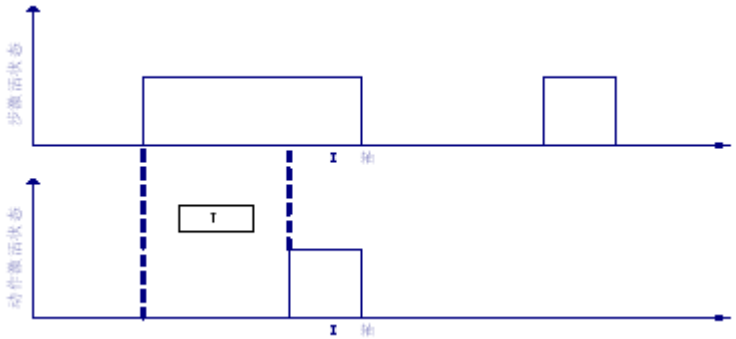


图 11- 6 D 动作状态

P 动作在步激活后只激活一个程序扫描时间，然后恢复成不激活状态，如图 11-7 所示。

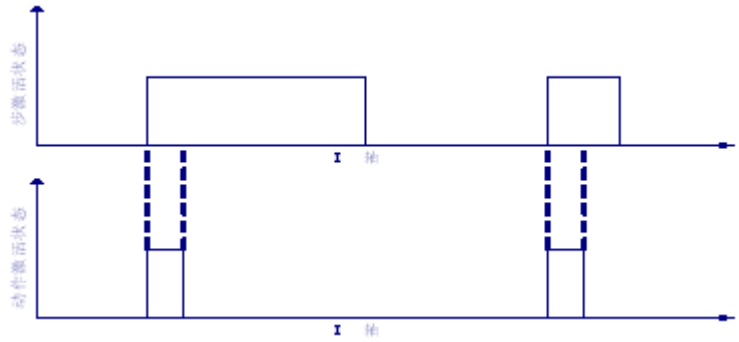


图 11- 7 P 动作状态

DS 动作在步激活后维持限定的时间后，变为激活状态，并一直维持。但 DS 是会被打断的，若时间 T 内，下一步被激活，则直接执行下一步，不会执行 DS 的动作。如图 11-8 所示。

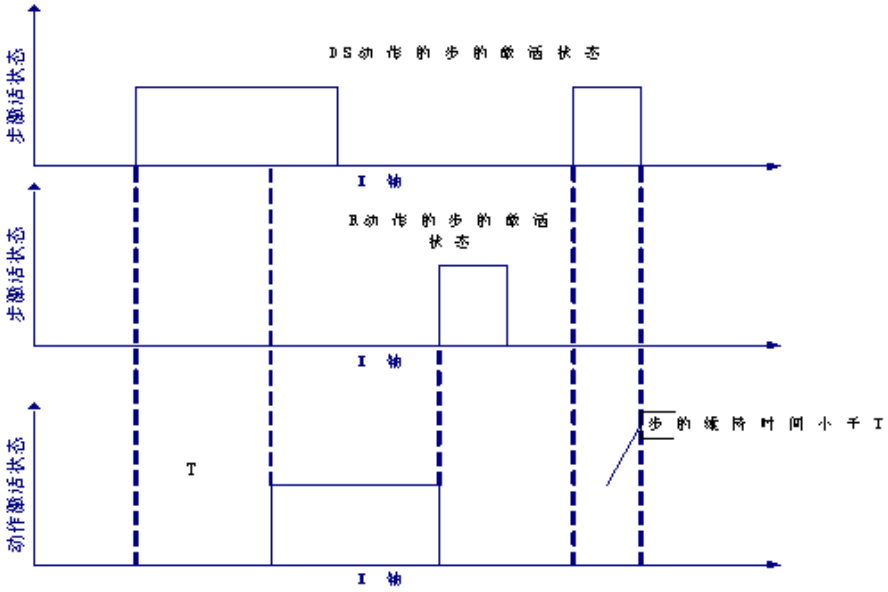


图 11- 8 DS 动作状态

SD 动作，只要时间 T 一到，就执行动作，不管时间 T 内下一步有没有激活。如图 11-9 所示。

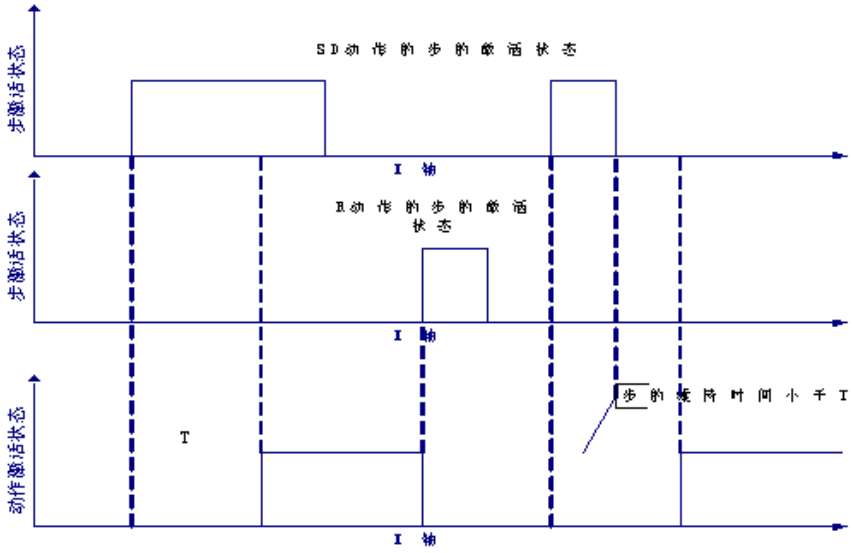


图 11- 9 SD 动作状态

SL 动作在步激活后在限定的时间内保持激活，超出时间恢复成不激活状态，与步的失去激活无关，如图 11-10 所示。

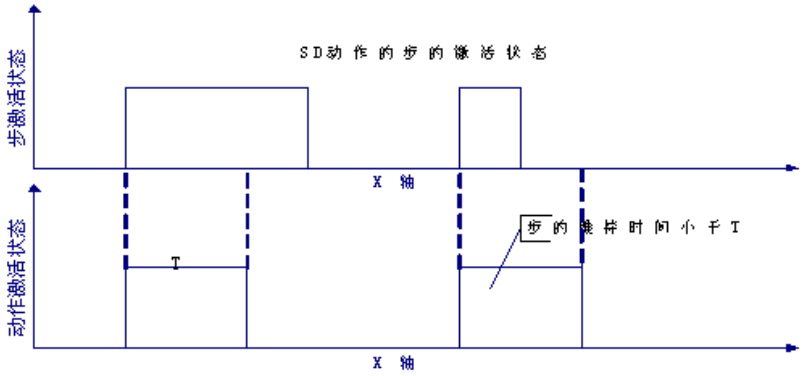


图 11- 10 SL 动作状态

子程序或记录点可以在不同步中被多次调用，从而该子程序的执行或记录点的赋值，由所有这些相关步的激活状态及操作限定词决定。

3.5. 执行顺序

一个 SFC 结构的网络分为一系列步和转换。它们要循环地进行求值和执行。一个步总是处于激活状态或者不激活状态。每一次循环对所有转换的求值，其结果不是 TRUE 就是 FALSE。待处理循环的活动步清单取决于这些步所依赖的转换的计算值。

在一个 SFC 网络中所有指令的执行依照以下算法进行调度。

激活起始步（仅在第一次调用时），否则对其后继转换为 TRUE 的所有当前激活步解除激活，并激活紧接这些转换后的所有步。

检查所有动作控制的 Q 输出，若刚检测到一个 TRUE－FALSE 边沿，则最后一次执行所关联的动作。

执行其动作控制所求值为 TRUE（Q 输出）的所有操作。

对转换进行求值，并从步 1 开始继续循环执行。

在 SFC 网络中没有显性的终点。如果没有后继的转换，则程序不会自动地返回到初始步，SFC 程序将停留在最后一个活动步。

4. ST 语言

4.1. 概述

结构化文本语言 ST 是 IEC61131-3 的文本化语言。ST 被称为高级语言，因为它不采用低级的、面向机器的操作符而是以高度压缩的方式提供大量描述复杂功能性的抽象语句。

4.2. 数据类型

在 ST 语言中，用户可定义模拟型、数字型、整数型三种数据类型，同时也可以定义系统功能块或子程序。具体定义变量可参见《算法编辑器》中《局部变量》这一节。

4.3. 标识符

局部变量或子程序的命名必须满足以下规范：

1. 以英文字母或中文开头；
2. 续以英文字母、数字、下划线或中文；
3. 数据长度不超过 32 字节。

4.4. 关键字

在 ST 语言中，定义了如下关键字，各关键字的功能见表 12-1 所示。

表 12- 1 关键字功能

关键字	描述
CASE……OF……ELSE……END_CASE	CASE 语句
FOR……TO……BY……DO……END_FOR	FOR 语句
IF…THEN…ELSEIF…ELSE…END_IF	IF 语句
REPEAT…UNTIL…END_REPEAT	REPEAT 语句
WHILE…DO…END_WHILE	WHILE 语句
EXIT	退出循环
RETURN	退出程序
：=	赋值
;	空语句
,	参数间隔符
=>	用于功能块调用时输出赋值

4.5. 操作符

ST 中定义了如表 12-2 所示的操作符。

表 12- 2 操作符

操作符	说明	举例和表达式的值	优先级
( )	括号	(2*3)+(4*5)	高
	函数调用	EQ(X，Y)	
**	求幂	3**4	
-	取相反数	-10	
NOT	取反	NOT 0	

*	乘法	10*29	
/	除法	29/10	
%	取余	17%10	
+	加法	1.4+2.5	
-	减法	4.5-2.1-3.	
<，>，<=，>=	比较	10>20	
=	等于	20=30	
<>	不等于	20<>30	
AND	布尔 AND	0 AND 1	
XOR	布尔 XOR	0 XOR1	
OR	布尔 OR	0 OR 1	低

功能块调用不同于函数调用，函数调用是一个表达式，而功能块调用是一个语句，它没有返回值，因此在一个表达式内不允许功能块调用。

4.6. 表达式

表达式为变量、常数、操作符、函数的组合，求值结果为单个值。

表达式的求值按操作符的优先级进行，优先级高的操作符先被处理，相同优先级的按从左到右的顺序执行。

括号不仅能用来定义一个特殊的处理顺序，而且还能提高复杂表达式的可读性。此外，使用括号可以避免错误地假设优先次序。

以下为几个合法的表达式举例：

Q：=A+B*(C+D**E)+SIN(F)；

Q：=SIN(A*F+D)；

4.7. 语句

ST 语言允许的语句如表 12-3 所示。

表 12- 3 语句

序号	语句	功能	例子
1	赋值语句	将右边表达式的值赋给左边的变量	A：=B；
2	函数调用	根据输入参数执行 FUN 函数的功能，并将其输出值作为一个表达式的输出。	A：=FUN（P1，P2）；
3	功能块调用	根据输出参数执行 FB 的功能，其输出值均在其实例所分配的内存区	FB1（IN1，IN2，OUT1，OUT2）；
4	IF	选择语句，以 IF 开头，END_IF 结尾。当所判断的表达式为 TRUE 时，执行 THEN 之后的语句；当所表达式为 FALSE 时，执行 ELSE 之后的语句。	**IF** 表达式 **THEN** 语句块； **ELSEIF** 表达式 **THEN** 语句块； **ELSE** 语句块； **END_IF**

5	CASE	多重选择语句，以 CASE 开头，END_CASE 结尾。当所判断的表达式等于某个选择分支的数值时，执行该选择分支后的语句。若所判断的表达式与所有的选择分支均不符合时，则执行 ELSE 之后的语句。	TW：=FUN(P1,P2); **CASE** 表达式 **OF** 1： 语句块； 2： 语句块； **ELSE** 语句块； **END_CASE**
6	FOR	循环语句，以 FOR 开头，END_FOR 结尾。根据所判断的变量从初始值到终止值，按照 BY 后的表达式值累增，循环执行 DO 后面的语句，直到所判断的变量大于终止值。 一个 FOR 语句包括以下步骤： 1. 初始化变量 2. 如变量在定义的范围之外，则检查结束条件和停止循环的执行； 3. 执行语句块 4. 根据 BY 表达式的值增量控制变量 5. 重复第二步到第 4 步。	**FOR** 变量：＝表达式 **TO** 表达式 **BY** 表达式 **DO** 语句块； **END_FOR**
7	WHILE	循环语句。以 WHLE 开头，END_WHILE 结尾。当所判断的表达式为 TRUE 时，循环执行 DO 后面的语句，直到所判断的表达式为 FALSE 为止。	**WHILE** 表达式 **DO** 语句块； **END_WHILE**
8	REPEAT	循环语句。以 REPEAT 语句开头，END_REPEAT 结尾。首先执行 REPEAT 后的语句，若 UNTIL 后的表达式为 FALSE，则循环执行 REPEAT 后的语句；否则结束循环。	**REPEAT** 语句块； **UNTIL** 表达式 **END_REPEAT**
9	RETURN	脱离当前的 POU 和返回到调用 POU	**IF X<y THEN RETURN;** **END_IF;**
10	EXIT	一个迭代语句的提前结束	**FOR ... TO … BY … DO** **IF…THEN EXIT;** **END_IF;** **…** **END_FOR**

4.8. 函数调用

一个函数调用由功能名和由逗号分割并包括在括号内的参数表组成。既可使用一个形式参数表，也可以使用一个实际参数表。因为一个形式参数表包括每个参数的名称以及其实际值，而形式参数的次序可以是任意的。参数可以省略并会提供一个默认值。实际参数表只包括由逗号分割的实际值。在这种情况下，所有参数必须以其正确的顺序出现，也就是说，在

功能说明中定义的功能参数的顺序。系统功能函数和功能块的说明请参考相应章节《函数库》和《功能块库》。

假设有如图 12-1 所示的用 FBD 语言编辑的控制功能图。

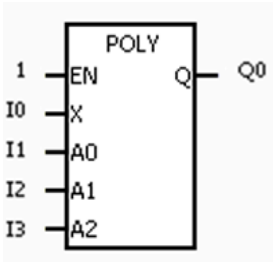


图 12- 1 FBD 功能块图

等价的 ST 语言如下：Q0：＝POLY(I0，I1，I2，I3)；

我们也可以对输入参数进行省略，若省略，则采用该函数声明时的默认值进行处理。

Q0 ：＝POLY(I0,，I2，I3)；这里 A1 的值就采用系统默认值 0 进行计算。每个函数的默认值可参考第九章《函数库》。

4.9. 功能块调用

在 ST 语言中，功能块调用由其实例名和括号中的一个参数表激活。这些参数表包括形式参数和通过“：＝”的实际赋值。参数赋值的先后顺序可以修改。

假设有如图 12-2 所示的用 FBD 编辑的控制功能图：

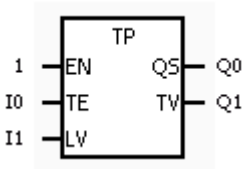


图 12- 2 FBD 功能块图

TP_1 为该 TP 功能块的实例名。

IN 就代表着功能块的实例 TP_1 内部 IN 这个变量的值。

等价的 ST 语言如下：TP_1(LV:=I1, TE:=I0, QS=>Q0, TV=>Q1)。

5. IL 语言

5.1. 概述

指令表 IL 是一种便于使用的类似汇编器的编程语言。

5.2. 指令

IL 是一种面向行的语言。一条指令，是可执行的一项命令，它严格要求由一个行来表述，也允许空白行形式的空指令。

IL 中的一个语句包括表 13-1 所列的这些元素：标号、:、操作符/函数、操作数、注释。

表 13- 1 元素说明

序号	符号	说明
1	标号	跳转标号，为到达该指令；标号和冒号是可选的。
2	:	冒号，作为分界符

2	操作符/函数	IL 操作符或函数名
3	操作数	用于操作符的零个、一个或多个常数或变量或用于功能的输入参数，由逗号分割
4	注释	在（*……*）中的注释是可选的。

在程序执行中，要完成跳转到程序中的其他行，标号是必不可少的。

注释是由一对星号和括号(*　*)进行界定。每个 IL 行只允许有一个注释，注释不能进行嵌套。

在 IL 行中不允许使用分号“；“，分号不能作为一个语句的结束符。

IL 提供一个称为“当前结果“（CR）的累加器，但 CR 并不象实际的硬件累加器那样具有固定数量的存储位。IL 编译程序确保总能提供一个任意存储宽度的虚拟累加器。

5.3. 操作符

本节描述 IL 定义的所有操作符。某些操作符可带修饰符。一个操作符与一个修饰符的组合具有扩展的含义。

修饰符详述：

N　操作数求反

（ 用括号的嵌套级

C　操作符的有条件执行

操作符求反：

在执行该指令之前，将操作数求反：

ANDN　　VAR1

用括号的嵌套级

通过使用括号修饰符，有可能执行 CR 和一个完整序列指令的结果的逻辑运算。当检测到修饰符“(“时，操作符类型以及当前 CR 的数值和数据类型是”保存“的，并将新的数值和类型装入 CR 中；当检测到结束的括号”）”时，则递延的数值和数据类型是可以恢复的，并使用修饰操作符和当前的 CR 值进行运算，运算的结果存储在 CR 中。

求值的顺序如下：

“递沿的 CR“ OP ”当前的 CR“。（OP 为对应的操作符，这里为 AND）。

```
LD      VAR1
AND(    VAR2
OR      VAR3
)
ST      VAR4
```

上述语句相当于：

```
LD      VAR2
OR      VAR3
AND     VAR1
ST      VAR4
```

- 操作符的有条件执行

有些操作符生成 BOOL 数值并存储在 CR 中。如 BOOL 值为真，则执行后继的有条件指令，否则，将处理器的控制给予跟随在有条件指令后的指令。

- 操作符列表如表 13-2 所示。

表 13- 2 操作符列表

操作符	描述
LD LDN	装入操作数（操作数的反值）到 CR
AND ANDN AND(ANDN(	操作数（操作数的反值）和 CR 的布尔 AND（“与“运算）
OR ORN OR(ORN(	操作数（操作数的反值）和 CR 的布尔 OR（“或“运算）
XOR XORN XOR(XORN(	操作数（操作数的反值）和 CR 的布尔 XOR（“异或“运算）
ST STN	将 CR（CR 的反值）存到操作数
S	若 CR=1，则将操作数设置为 1；
R	若 CR=1，则将操作数设置为 0；
)	结束括号级
ADD ADD(	加操作数，结果存入 CR
SUB SUB(	从 CR 减去操作数，结果存入 CR
MUL MUL(	操作数乘以 CR，将结果值写入 CR
DIV DIV(	CR 除以操作数，将结果值写入 CR
GT GT(	CR>操作数，将比较值写入 CR
GE GE(	CR>=操作数，将比较值写入 CR
EQ EQ(	CR=操作数，将比较值写入 CR
NE NE(	CR<>操作数，将比较值写入 CR
LE LE(	CR<=操作数，将比较值写入 CR
LT LT(	CR<操作数，将比较值写入 CR
JMP	无条件跳转到一个跳转标号
JMPC	当条件成立时，跳转到标号，否则执行其后的指令
JMPCN	当条件成立时，执行下面的指令，否则跳转到标号
RET	无条件退出当前程序
RETC	当条件成立时，退出当前程序
RETCN	当条件不成立时，退出当前程序
CAL	无条件调用功能块

5.4. 函数调用

在 IL 语言中，调用一个函数与调用一个操作数基本相同，函数的第一个参数是当前结果（CR）。因此，必须正好在函数调用之前将该值装入 CR 中。用于函数调用的第一个操作数实际上是函数的第二个参数。

一个函数准确地返回一个数值，它存储于 CR 中。

假设有如图 13-1 所示的用 FBD 语言编辑的控制功能图：

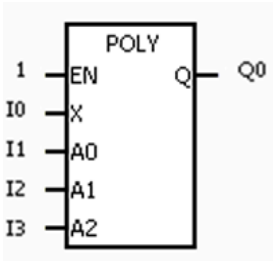


图 13- 1 FBD 功能块图

等价的 IL 语言如下：

LD　　I0

POLY　　I1，I2，I3

ST　　Q0

这里 POLY 为多项式加法，其执行如下功能：Q0：＝POLY(I0，I1，I2，I3)。

注　意

在系统函数以 IL 语言调用时，EN 不能作为输入参数，其第一个输入参数为 FBD 图形左边 EN 引脚下的那个引脚。在函数调用时，其输出参数不能作为参数传递给函数内部。即下面的语法是错误的：

LD　　I0

POLY　　I1，I2，I3，Q0

5.5. 功能块调用

操作符 CAL 可以调用一个功能块或子程序的实例。IEC61131-3 描述 IL 语言中给一个功能块传送参数的三种方法：

1. 使用一个调用，它包括在括号内的实际输入和输出参数的一个列表；
2. 在调用功能块时，装载和保存输入参数；
3. 用输入参数作为操作符“隐性地“调用。

目前算法编辑器只支持前两种调用方法。

假设有如图 13-2 所示的用 FBD 编辑的控制功能图。

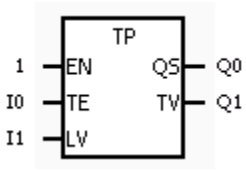


图 13- 2 FBD 功能块图

利用 IL 语言可写成如下指令格式：

CAL TP_1(TE：=I0，LV：=I1，QS=>Q0，TV=>Q2)；

或 LD I0

ST TP_1.TE

LD I1

ST TP_1.LV

CAL TP_1()

LD TP_1.QS

```
ST Q0
LD TP_1.TV
ST Q2
```

注　意

系统功能块的调用，EN 引脚在 IL 语言中同样不能作为输入参数传递，其第一个参数为 EN 引脚下的那个引脚。

5.6. IL 程序举例

下例为一个山地缆车控制系统的简化流程，我们用 IL 语言进行编写，同时给出其等价的 LD 语言。

假设该山地缆车共有三个站，其中一个为山谷站 S1，另一个为山顶站 S3，还有一个为中间站 S2。

程序必须提供以下控制功能：

当缆车客舱到达三个站中的一个站时，传感器 S1，S2，S3 发送 TRUE 信号，计数器 StationStop 存储停靠在这些车站的总次数；

驱动缆车的电机由以下信号控制：

（1） 方向 Direction：前进 TRUE，后退 FALSE；

（2） 启动和停止 StartStop：启动 TRUE，停止 FALSE。

在客舱的内部，用一个开关 DoorOpenSignal 来打开/关闭舱门。当 DoorOpenSignal 等于 TRUE 时，指示“开门“；等于 FALSE 时，指示”关门“。

控制舱门的电机有两个执行器：OpenDoor(开门)和 CloseDoor(关门)。两个执行器均由两种信号中的一种信号的上升沿触发。

一个按钮 MRStart 启动整个控制系统，按钮 MREnd 用于关闭系统。

在缆车停止运行和重新启动之间必须激活一个报告存在危险的信号。

利用 IL 语言编辑的控制程序如下：

```
(*电源接通后的第一次系统运行吗？是：复位停止信号*)
(*由最后的关闭信号激活*)
PouBegin:
CAL     IMRStart(MRStart)
LD      IMRStart.QR             (*第一次调用吗？*)
R       EndSignal               (*是：复位停止信号*)
LD      V1
ST      StationStop.CU
LD      IMRStart.QR             (*第一次调用吗？*)
ST      StationStop.R           (*计数器复位*)
LD      9999
ST      StationStop.PV                  (*计数器累计最大值*)
CAL     StationStop(StationStop.CV=>CtuCount)           (*执行操作*)

(*当客舱到达一个站时，停止客舱，StationStop 计数器加 1*)
```

```
    Arrive:
    LD      S1
    OR      S2
    OR      S3
    (*若现在为 TRUE，则停止，计数器加 1*)
    ST      V1
    R       StartStop           (*停止客舱*)

    (*是否改变方向*)
    LD          S1
    XOR     S3
    JMPC    NoDirChange             (*S1 或 S3 中是否有一个为 TRUE*)

    (*改变客舱方向*)
    LD      Direction
    STN     Direction

    (*开舱门的条件：客舱停止和开门开关被激活*)
    NoDirChange:
    LD      DoorOpenSignal
    ANDN    StartStop
    ST      OpenDoor

    (*在一个站内，用户缆车和舱门的停止信号*)
    LD      MREnd               (*是否激活关闭电源信号*)
    ANDN    StartStop
    S       EndSignal
    JMPC    PouEnd

    (*关闭舱门的开关信号*)
    CloseCabin:
    LD      DoorOpenSignal
    STN     CloseDoor

PouEnd:
    (*在关门信号激活 10 秒后启动客舱*)
    LDN     DoorOpenSignal
    ANDN    StartStop
    ST      DoorTime.IN
    LD      10
    ST      DoorTime.PT
    CAL     DoorTime()
    LD      DoorTime.Q
    S       StartStop
```

6. 子程序

6.1. 概述

用户可以定义以 FBD、LD、ST、IL 语言编辑的子程序，在程序中通过 CAL 功能块或 CAL 指令来实现对子程序的调用。

一个子程序代表由用户定义的输入输出变量组成的框架。内部包含用户定义的程序逻辑。用户可以通过调用功能块或 CAL 指令多次重复调用子程序。

在算法编辑器中所有的子程序都被内部认为是功能块，也就是说用户在调用某个子程序时，实际上对该子程序的实例进行引用，每个被调用的子程序均具有独立的数据地址，它们相互之间不影响。

6.2. 局部变量编辑

在子程序中，用户可以定义三种类型的局部变量；

输入变量 VAR_INPUT；

输出变量 VAR_OUTPUT；

内部变量 VAR_TEMP。

输入变量用作向子程序传递数值，输出变量用于从子程序输出数值。这两类变量被称为形式参数。这些变量在子程序被调用时表现为外部的输入输出引脚。

变量的定义在内部变量观察窗口中定义。引脚序号即为在变量编辑器中的自然顺序。

在子程序中还提供了内部变量。子程序中的内部变量与在程序中定义的内部变量不同。程序中的内部变量唯一对应控制站的一个内存地址，而子程序中的内部变量，在子程序不被其他程序调用时，没有对应实际的控制站地址。当其他程序调用子程序时，每次调用，子程序中的内部变量都会分配一个不同的控制站地址。也就是说，在子程序中的内部变量封装了对控制站内存的申请。每一次子程序实例的创建，都会在控制站为其内部变量申明相同数目的控制站内存，这些内存只被子程序实例所引用，对外部程序不可见。

6.3. 子程序编辑

在算法编辑器中，子程序的编辑与其所选择的程序类型的编辑方式完全相同，用户可以参考相应的程序类型的编辑方式进行编辑。

6.4. 子程序调用

在算法编辑器中，对子程序的调用有四种方式。

6.4.1. FBD、LD 语言中的子程序调用

在 FBD，LD 这两种图形编程语言中，用户可以通过引用调用功能块 CAL 来实现对子程序的调用。

首先选择 CAL 功能块，如图 14-1 所示。

图 14- 1 CAL 功能块

双击该功能块，弹出如图 14-2 所示的窗口。

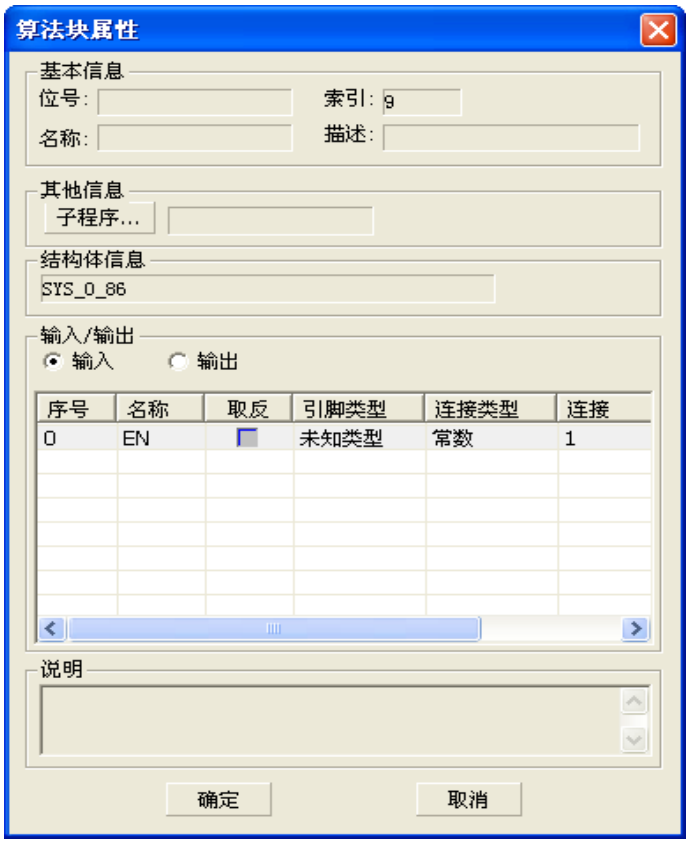


图 14- 2 算法块属性

双击“子程序...”按钮，弹出如图 14-3 所示的对话框。

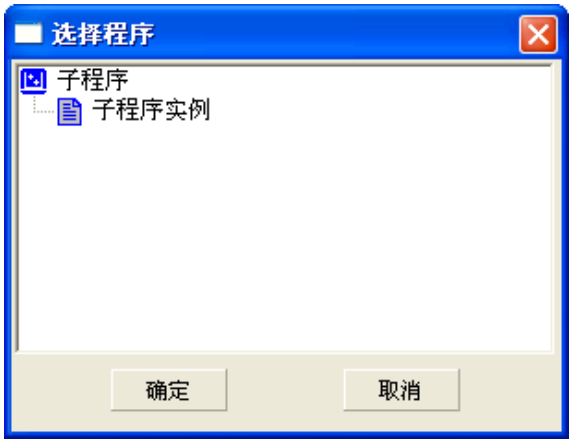


图 14- 3 程序选择对话框

选择要调用的子程序，点击“确定”按钮，即实现了对该子程序的调用。

6.4.2. SFC 语言中子程序的调用

在 SFC 语言中，用户在定义每个步的执行动作时，每个动作可以指定其所调用的子程序。

在步的属性窗口中，点击增加按钮，系统会在左边的列表框中添加一个动作，双击该列表框的某行，系统会弹出如图 14-4 所示的对话框。

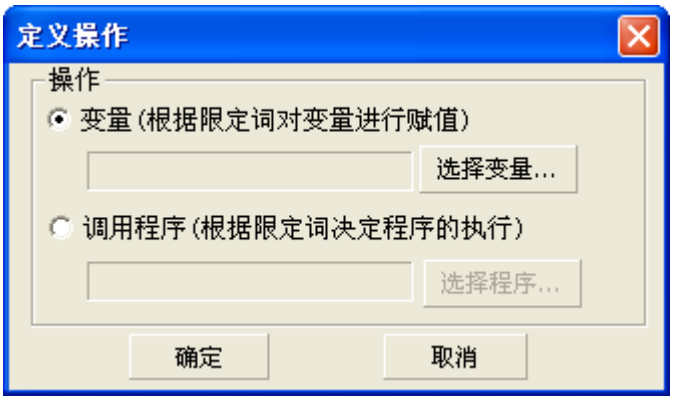


图 14- 4 定义操作对话框

选中“调用程序”，并双击“选择程序”按钮，在弹出的对话框中选择要调用的子程序，如图 14-5 所示。

图 14- 5 程序选择对话框

从而实现了对子程序的调用。

6.4.3. ST 语言的子程序调用

在 ST 文本中，用户可以通过子程序的实例名来调用该子程序。

假设我们定义了如下一个子程序实例变量：SUB_P_1。

则我们可以使用：

SUB_P_1()

来实现调用子程序。

括号内可以根据该子程序中所定义的 VAR_INPUT 或 VAR_OUTPUT 类型的变量，加入一些表达式作为参数传递给该子程序实例。

6.4.4. IL 语言中的子程序调用

在 IL 语言中，用户可以通过 CAL 指令来实现对子程序的调用。

假设我们定义了如下一个子程序实例变量：SUB_P_1。

则我们可以使用：

CAL　　SUB_P_1()

来实现调用子程序。

括号内可以根据该子程序中所定义的 VAR_INPUT 或 VAR_OUTPUT 类型的变量，加入一些表达式作为参数传递给该子程序实例。

行业库

1. 概述

行业库相当于一个子程序的集合，封装了各种自控行业中常用的算法，用户在编程时可以直接在行业库中选择需要的算法，不用自己重新编写类似功能的算法，大大地减少了编程的工作量。

在算法编辑器的导航栏中有“行业库”选项卡，切换到“行业库”选项卡后画面如图 15-1 所示，行业库按行业分类，点开行业库根目录后，下面有以行业分类的分节点，点开一个行业的分节点，如图 15-1 中环保行业所示，是各种封装好的行业库算法。

图 15- 1 行业库

2. 行业库的调用

2.1. 行业库算法在 FBD 程序中的调用

行业库算法和 FBD 功能块的使用相似，在 FBD 程序中调用行业库算法可以直接从导航栏的行业库中选中一个行业库算法拖入 FBD 程序中，或者在 FBD 程序中空白处单击鼠标右键，在弹出的菜单中选择“插入行业算法”，如图 15-2 所示，然后在弹出的画面中选择要插入的算法，如图 15-3 所示，点击“确定”即可成功插入行业库算法。

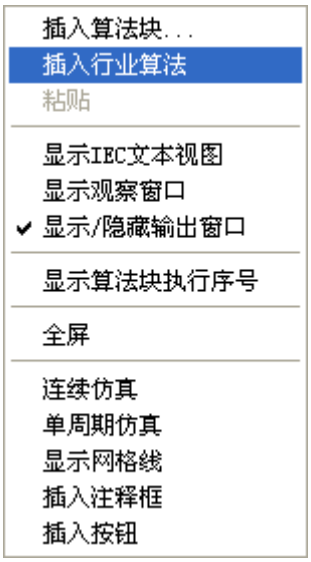


图 15- 2 插入行业算法

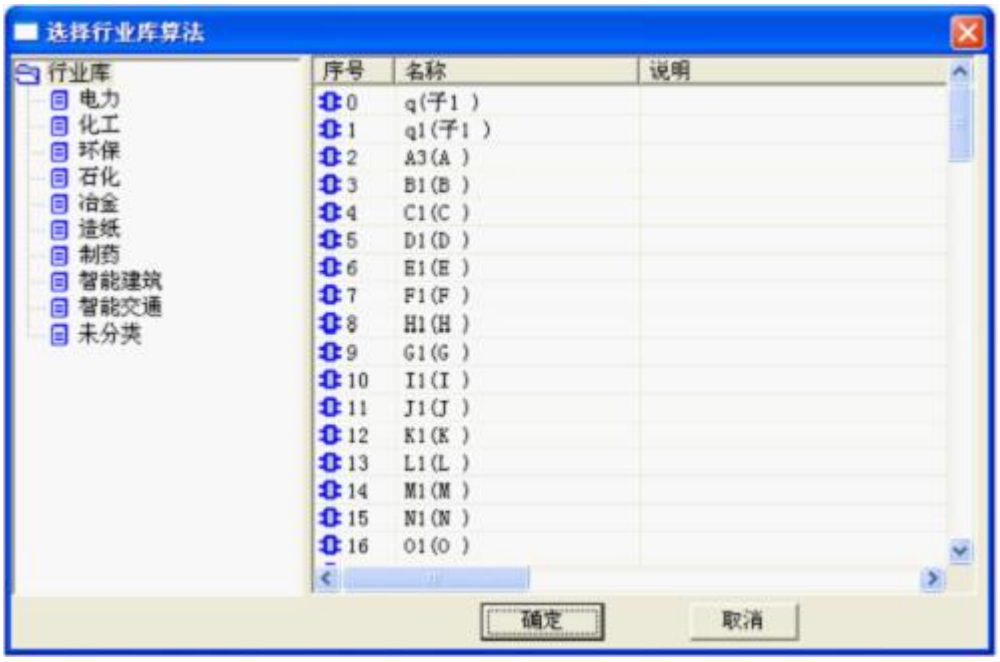


图 15- 3 选择行业算法库算法

行业库算法在 FBD 语言中的其它操作请参考 FBD 算法块的编辑。

2.2. 行业库算法在 LD 程序中的调用

在 LD 程序中调用行业库算法，先选中网络，然后单机鼠标右键，在弹出的菜单中选择“右边插入行业算法”，如图 15-4 所示。

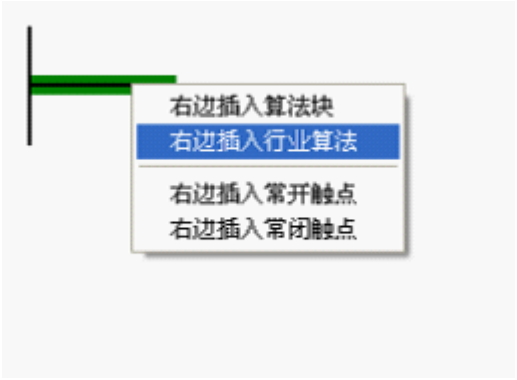


图 15- 4 右边插入行业算法

在弹出如图 15-3 所示窗口中选择要插入的行业库算法，点击“确定”即可成功插入行业库算法。

行业库算法在 LD 程序中的其它操作，请参考 LD 语言中 FBD 算法块的编辑。

2.3. 行业库算法在 ST 和 IL 程序中的调用

在 ST 和 IL 程序中调用行业库，要先添加一个局部变量，然后在局部变量的类型栏里选择行业库实例，如图 15-5 所示。

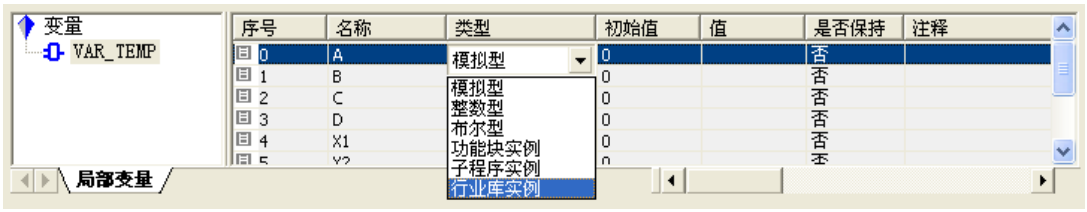


图 15- 5 局部变量类型选择

具体的调用实例参考 FBD 功能块在 ST 语言，IL 语言中的调用。

3. 行业库编辑

3.1. 删除行业

选中要删除的行业，单击鼠标右键，如图 15-6 所示 。

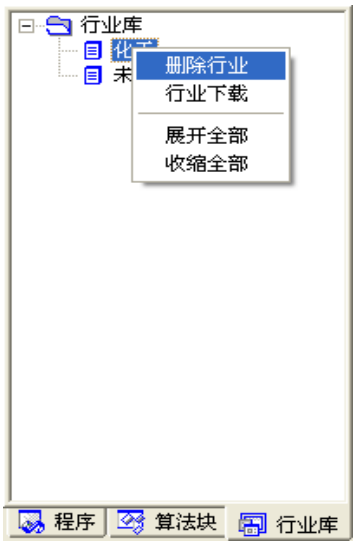


图 15- 6 删除行业

点击“删除行业”，弹出如图 15-7 所示对话框，选择“是（Y）”,就可以把行业删除。

图 15- 7 是否删除行业对话框

3.2. 导入行业

如果要恢复被删除的行业，则在行业库的导航栏中选中行业库根节点，单击鼠标右键，如图 15-8 所示。

图 15- 8 导入行业

点击“导入行业”后，在弹出的对话框里打开 UWinTech Pro 软件安装目录下的 Package 文件夹，如图 15-9 所示，Package 文件夹下有所有行业库中行业的文件。

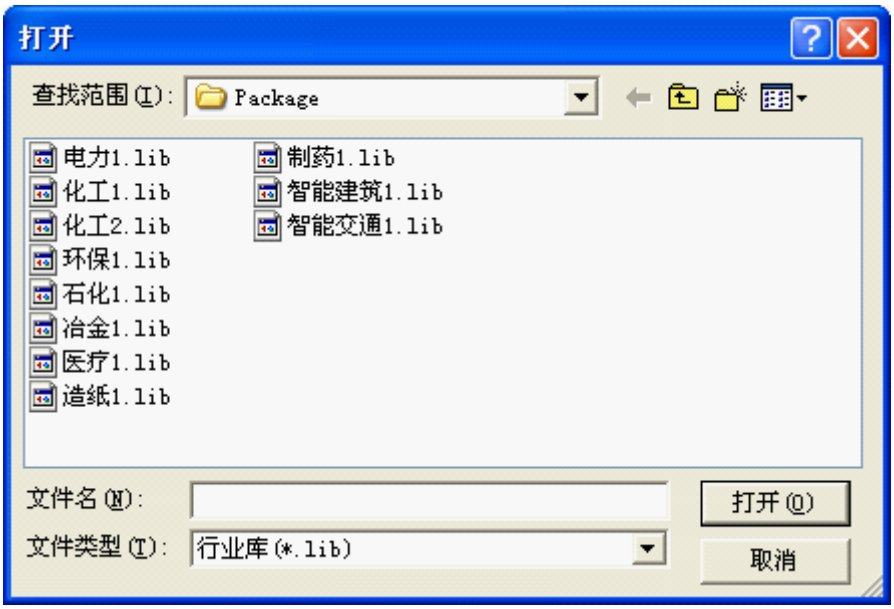


图 15- 9 行业选择画面

在文件夹中选中需要导入的行业，点击“打开（O）”弹出如图 15-10 所示对话框，选择“是（Y）”，重启算法编辑器就可以了。

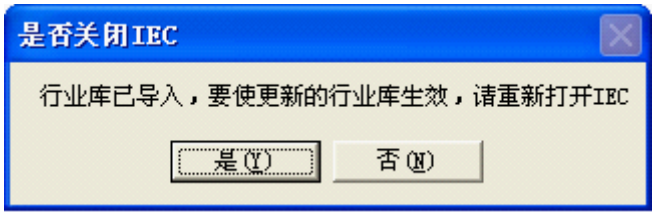


图 15- 10 是否重启算法编辑器对话框

3.3. 行业列表

在行业库的导航栏中选中行业库根节点，单击鼠标右键，点击“行业列表”，如图 15-11 所示，会弹出浏览器显示行业列表，如图 15-12 所示，可以选择下载整个行业库、行业或子行业。

图 15- 11 行业列表

图 15- 12 行业列表浏览器窗口

3.4. 行业下载

在行业库的导航栏中选中行业库根节点，单击鼠标右键，点击“行业下载”，如图 15-13 所示，会弹出浏览器显示行业列表，如图 15-14 所示，输入授权码后才可以下载。授权码也可在线购买。

图 15- 13 行业下载

图 15- 14 行业下载浏览器窗口

注　意

行业列表和行业下载功能需在联网情况下才能使用。

算法库

1. 函数库

1.1. 逻辑

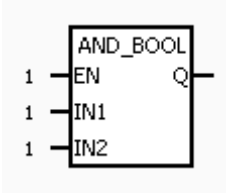
1.1.1. AND_BOOL（逻辑与）

简介：

该模块功能是将两个输入变量的值进行逻辑与运算，输出变量的值为 0 或 1。

该模块认为非 0 即 1。最大允许 12 个输入。

符号：



参数描述：

参数	初始值	含义	真值表			
IN1	1	输入，数字型	0	0	1	1
IN2	1	输入，数字型	0	1	0	1
Q	0	输出，数字型	0	0	0	1

举例：

IN1= 1； IN2= 1； Q=1；

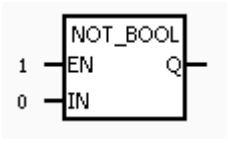
1.1.2. NOT_BOOL（逻辑非）

简介：

该模块将输入变量的值进行逻辑取反运算。输出变量的值为 0 或 1。

该模块认为非 0 即 1。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，数字型
Q	0	输出，数字型

举例：

IN=1； Q=0；

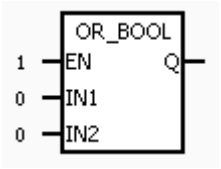
1.1.3. OR_BOOL（逻辑或）

简介：

该模块功能是将两个输入变量的值进行逻辑或运算，输出变量的值为 0 或 1。

该模块认为非 0 即 1。最大允许 12 个输入。

符号：



参数描述：

参数	初始值	含义	真值表			
IN1	0	输入，数字型	0	0	1	1
IN2	0	输入，数字型	0	1	0	1
Q	0	输出，数字型	0	1	1	1

举例：

IN1=0；IN2=1；Q=1；

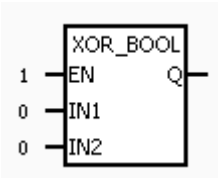
1.1.4. XOR_BOOL（逻辑异或）

简介：

该模块将两个输入变量的值进行逻辑异或运算，相异为 1，相同为 0，输出变量的值为 0 或 1。

该模块认为非 0 即 1。

符号：



参数描述：

参数	初始值	含义	真值表			
IN1	0	输入，数字型	0	0	1	1
IN2	0	输入，数字型	0	1	0	1
Q	0	输出，数字型	0	1	1	0

举例：

IN1=1；IN2=1；Q=0；

1.1.5. AND_DWORD（双字逻辑与）

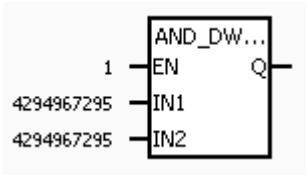
简介：

该模块功能是将两个输入变量的值的各位进行逻辑与运算，输出变量的值为一个无符号

整数。最大允许 12 个输入。

该模块要求所输入的变量均为整数，若不是整数，则对其进行取整(忽略小数) 。

符号：



参数描述

参数	初始值	含义
IN1	4294967295	输入，整数型
IN2	4294967295	输入，整数型
Q	0	输出，整数型

举例：

IN1=2；IN2=5；Q=0；

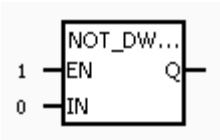
1.1.6. NOT_DWORD（双字逻辑非）

简介：

该模块将输入的各位进行逻辑取反运算。输出变量的值为一个无符号整数。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
Q	0	输出，整数型

举例：

IN=2；Q=0xfffffffd ＝4294967293；

1.1.7. OR_DWORD（双字逻辑或）

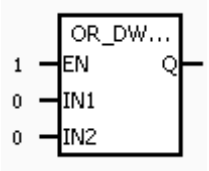
简介：

该模块将两个输入值的各位进行逻辑或运算，输出变量的值为一个无符号整数。

最大允许 12 个输入。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN1	0	输入，整数型
IN2	0	输入，整数型
Q	0	输出，整数型

举例：

IN1=2；IN2=5；Q=7；

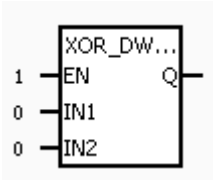
1.1.8. XOR_DWORD（双字逻辑异或）

简介：

该模块将两个输入的各位进行逻辑异或运算，输出变量的值为一个无符号整数。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN1	0	输入，整数型
IN2	0	输入，整数型
Q	0	输出，整数型

举例：

IN1=2；IN2=5；Q=7；

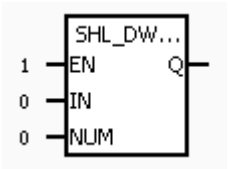
1.1.9. SHL_DWORD（双字左移位）

简介：

该模块将输入变量 IN 的值向左移动 NUM 位，输出变量的值为一个无符号整数，其有效位为 32 位。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
NUM	0	输入，整数型
Q	0	输出，整数型

举例：

IN=2；NUM=5；Q=0x40 ＝ 64；

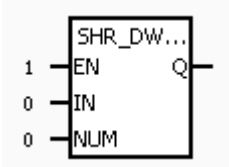
1.1.10. SHR_DWORD（双字右移位）

简介：

该模块将输入变量 IN 的值向右移动 NUM 位，输出变量的值为一个无符号整数，其有效位为 32 位。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
NUM	0	输入，整数型
Q	0	输出，整数型

举例：

IN=2；NUM=5；Q=0；

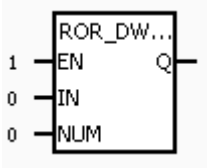
1.1.11. ROR_DWORD（双字循环右移）

简介：

该模块将输入变量 IN 的值向右移动 NUM 位，其被移出的位将被移动到最左端，输出变量的值为一个无符号整数，其有效位为 32 位。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
NUM	0	输入，整数型
Q	0	输出，整数型

举例：

IN=2；NUM=5；Q=0x10000000=268435456；

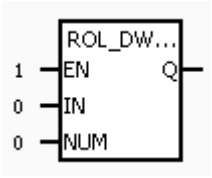
1.1.12. ROL_DWORD（双字循环左移）

简介：

该模块将输入变量 IN 的值向左移动 NUM 位，其被移出的位将被移动到最右端，输出变量的值为一个无符号整数，其有效位为 32 位。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
NUM	0	输入，整数型
Q	0	输出，整数型

举例：

IN=2；NUM=5；Q=0x40 ＝ 64；

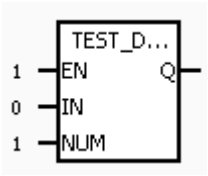
1.1.13. TEST_DWORD（双字位选择）

简介：

该模块选择输入变量 IN 的第 NUM 位的值作为输出，该模块的输出为 0 或 1。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
NUM	1	输入，整数型
Q	0	输出，整数型

举例：

IN=2；NUM=6；Q=0；

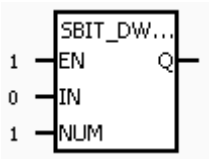
1.1.14. SBIT_DWORD（双字位置位）

简介：

该模块将输入变量 IN 的值的第 NUM 位置为 1，输出变量的值为一个无符号整数，其有效位为 32 位。

该模块要求输入变量必须为整数，若不是整数，则将相应的输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
NUM	1	输入，整数型
Q	0	输出，整数型

举例：

IN=2；NUM=6；Q=0x22 ＝ 34；

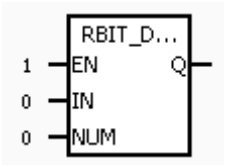
1.1.15. RBIT_DWORD（双字位复位）

简介：

该模块将输入变量 IN 的值的第 NUM 位置为 0，输出变量的值为一个无符号整数，其有效位为 32 位。

该模块要求输入变量必须为整数，若不是整数，则将相应输入变量进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN	0	输入，整数型
NUM	0	输入，整数型
Q	0	输出，整数型

举例：

IN=34；NUM=6；Q=2；

1.2. 算术

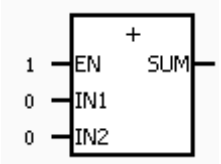
1.2.1. ADD（加法）

简介：

该模块将两个输入变量的值进行加法操作，输出变量的值为单精度浮点。

最大允许 8 个输入。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型

SUM	0.0	输出，模拟型

举例：

IN1=2；IN2=5.4； SUM=7.4；

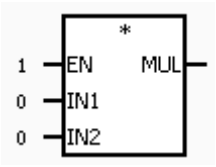
1.2.2. MUL（乘法）

简介：

该模块将输入变量的值进行乘操作，输出变量的值为单精度浮点。

最大允许 8 个输入。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
MUL	0.0	输出，模拟型

举例：

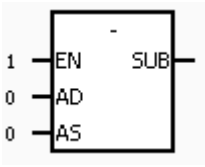
IN1=2；IN2=5.4； MUL=10.8；

1.2.3. SUB（减法）

简介：

该模块将两个输入变量的值进行减法操作，输出变量的值为单精度浮点。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
SUB	0.0	输出，模拟型

举例：

IN1=2；IN2=5.4； SUB=－3.4；

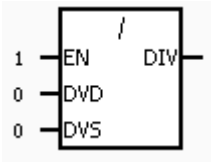
1.2.4. DIV（除法）

简介：

该模块将两个输入变量的值进行除法操作，输出变量的值为单精度浮点。

该模块要求第二个输入 IN2 不等于零，若该值为零，则该模块的输出值保持不变。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
DIV	0.0	输出，模拟型

举例：

DVD=2；DVS=5.4；DIV=0.37；

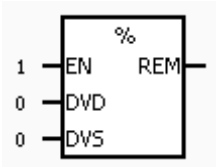
1.2.5. MOD（余数）

简介：

该模块将两个输入变量的值进行求模操作。

该模块要求输入为整数，若不是整数，则对其进行取整(忽略小数)。

符号：



参数描述：

参数	初始值	含义
IN1	0	输入，整数型
IN2	0	输入，整数型
DVS	0	输出，整数型

举例：

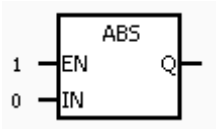
DVD=2；DVS=5； REM=2；

1.2.6. ABS（绝对值）

简介：

该模块将输入变量的值取绝对值赋给输出变量。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

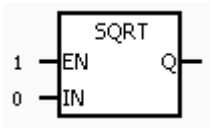
IN=－2.5；Q=2.5；

1.2.7. SQRT（平方根）

简介：

该模块将输入变量的值取平方根赋给输出变量。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

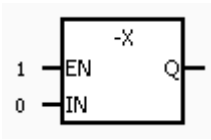
IN=3.24；Q=1.8；

1.2.8. INV（相反数）

简介：

该模块将输入变量的值取取反赋给输出变量。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

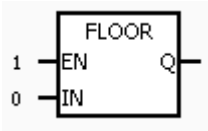
IN=3.24；Q=－3.24；

1.2.9. FLOOR（取整）

简介：

该模块将输入变量的值取整(取不大于输入值的最大整数)赋给输出变量。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0	输出，整数型

举例：

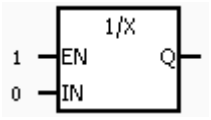
IN=3.24；Q=3；

1.2.10. RECIP（倒数）

简介：

该模块将输入变量的值的倒数赋给输出变量，该模块要求输入变量的值不为零，若输入变量的值为零，则该模块的输出变量的值保持不变。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

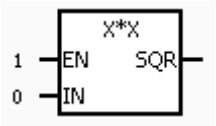
IN=2；Q=0.5；

1.2.11. SQR（平方）

简介：

该模块将输入变量的值的平方赋给输出变量。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
SQR	0.0	输出，模拟型

举例：

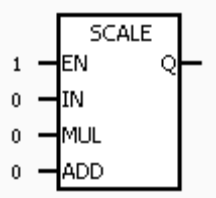
IN=2；SQR=4；

1.2.12. SCALE（收缩平移）

简介：

该模块执行如下运算 Q=IN*MUL+ADD。

符号：



参数描述：

参数	初始值	含义

IN	0.0	输入，模拟型
MUL	0.0	输入，放大系数，模拟型
ADD	0.0	输入，平移值，模拟型
Q	0.0	输出，模拟型

举例：

IN=2；MUL=5；ADD=4；Q=14；

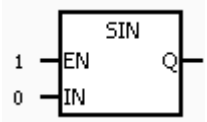
1.3. 三角

1.3.1. SIN（正弦）

简介：

该模块将输入变量的值的正弦赋给输出变量，输入变量的值是以弧度为单位。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型，以弧度为单位
Q	0.0	输出，模拟型，-1～1

举例：

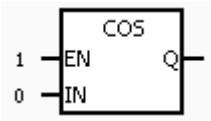
IN=2；Q=0.909；

1.3.2. COS（余弦）

简介：

该模块将输入变量的值的余弦赋给输出变量，输入变量的值是以弧度为单位。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型，以弧度为单位
Q	0.0	输出，模拟型，-1～1

举例：

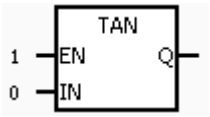
IN=2；Q=－0.416；

1.3.3. TAN（正切）

简介：

该模块将输入变量的值的正切赋给输出变量，输入变量的值是以弧度为单位。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型，以弧度为单位
Q	0.0	输出，模拟型

举例：

IN=2；Q=－2.185；

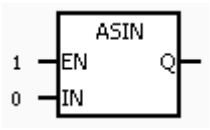
1.3.4. ASIN（反正弦）

简介：

该模块将输入变量的值的反正弦赋给输出变量，输出变量的值是以弧度为单位。

该模块要求输入变量的值在－1～＋1 之间，若不是，则输出变量值保持不变。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型，-1～1
Q	0.0	输出，模拟型，弧度

举例：

IN=0.2；Q=0.201；

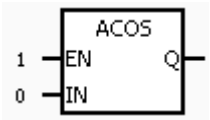
1.3.5. ACOS（反余弦）

简介：

该模块将输入变量的值的反余弦赋给输出变量，输出变量的值是以弧度为单位。

该模块要求输入变量的值在－1～＋1 之间，若不是，则输出变量值保持不变。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型，-1～1
Q	0.0	输出，模拟型，弧度

举例：

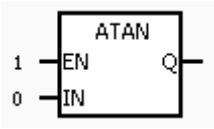
IN=0.2；Q=1.369；

1.3.6. ATAN（反正切）

简介：

该模块将输入变量的值的反正切赋给输出变量，输出变量的值是以弧度为单位。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型，弧度

举例：

IN=0.2；Q=0.197；

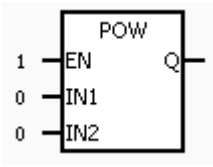
1.4. 代数

1.4.1. POW（求幂）

简介：

该模块将以 IN1 为底，IN2 为指数的值赋给输出变量。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	1.0	输出，模拟型

举例：

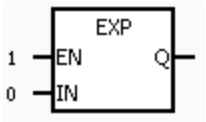
IN1=2；IN2=3.5；Q=11.314；

1.4.2. EXP（指数）

简介：

该模块将输入变量的值的自然指数值赋给输出变量。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IN=0.2；Q=1.221；

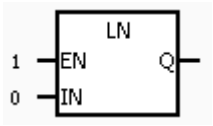
1.4.3. LN（自然对数）

简介：

该模块将输入变量的值的自然对数值赋给输出变量。

该模块要求输入变量值大于零，若不满足，则该模块输出值保持不变。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IN=0.2；Q=Q= -1.609；

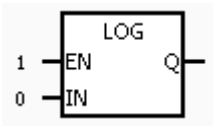
1.4.4. LOG（对数）

简介：

该模块将输入变量的值的以十为底的对数值赋给输出变量。

该模块要求输入变量值大于零，若不满足，则该模块输出值保持不变。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IN=0.2；Q=－0.699；

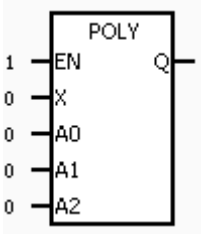
1.4.5. POLY（多项式加法）

简介：

该模块执行如下运算 Q=A0+A1*X+A2*X*X+…；

最大允许 11 项。

符号：



参数描述：

参数	初始值	含义
X	0.0	输入，模拟型
A0	0.0	输入，模拟型
A1	0.0	输入，模拟型
A2	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

X=2；A0=3；A1=4；A2=5；Q=25；

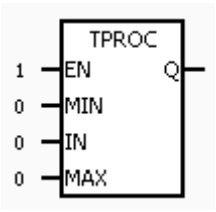
1.4.6. tPROC（转成百分数）

简介：

该模块执行如下运算：Q=(IN-MIN)/(MAX-MIN)×100。

该模块要求输入变量 MAX 值不小于等于 MIN，否则该模块输出值为 0。

符号：



参数描述：

参数	初始值	含义
MIN	0.0	输入，模拟型
IN	0.0	输入，模拟型
MAX	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

MIN=0.1；IN=0.2；MAX=0.3；Q=50；

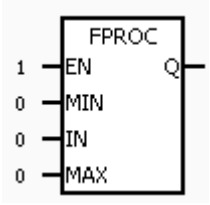
1.4.7. fPROC（转换百分比）

简介：

该模块执行如下运算：Q=MIN+(MAX-MIN)*IN/100。

该模块要求输入变量 IN 的值在 0～100 之间。若小于 0，则输出值等于 MIN；若大于 100，则输出值等于 MAX。

符号：



参数描述：

参数	初始值	含义
MIN	0.0	输入，模拟型
IN	0.0	输入，模拟型
MAX	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

MIN=0.1；IN=50；MAX=0.3；Q=0.2；

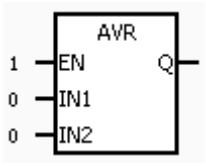
1.4.8. AVR（平均值）

简介：

该模块执行如下运算：Q=(IN1+IN2+…+INn)/n。

最大允许 16 个输入。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IN1=0.1；IN2=0.2；Q=0.15；

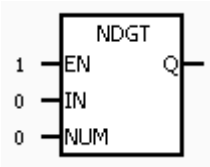
1.4.9. NDGT（四舍五入）

简介：

该模块将输入变量 IN 的值进行四舍五入，四舍五入后保留小数点后 NUM 位。

NUM 是一个非负整数。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
NUM	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IN=163.246；NUM=2；Q=163.25；

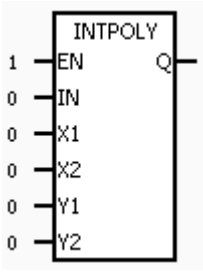
1.4.10. INTPOLY（线性插值）

简介：

该模块执行如下运算：Q=(IN - X1)*(Y2-Y1)/(X2-X1)+Y1。

该模块要求输入变量 X2 值不等于 X1，否则该模块输出值为零。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
X1	0.0	输入，模拟型
X2	0.0	输入，模拟型
Y1	0.0	输入，模拟型
Y2	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IN=2；X1=1；X2=3；Y1=2；Y2=4；Q=3；

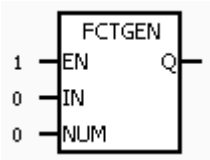
1.4.11. FCTGEN（折线函数）

简介：

最大接十二段折线，根据 x 值递增排序，添加段数参数，坐标以参数形式在算法块内部添加。

NUM 为有效的折线段数，若有 NUM 段折线有效，则有 NUM+1 个坐标点有效。根据输入的 IN 值，输出该坐标点对应的 Y 值。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，X坐标，模拟型
NUM	0.0	输入，模拟型

Q	0.0	输出，在X坐标为IN时Y坐标的值，模拟型
X0~X12	0.0	13个点的X坐标，模拟型
Y0~Y12	0.0	13个点的Y坐标，模拟型

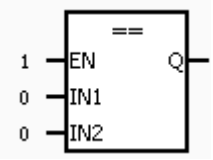
1.5. 比较

1.5.1. EQ（等于）

简介：

该模块执行如下运算，若 IN1 等于 IN2，则输出为 1；否则输出为 0。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0	输出，数字型

举例：

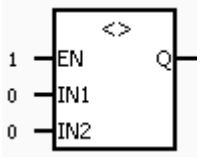
IN1=2；IN2=3.5；Q=0；

1.5.2. NE（不等于）

简介：

该模块执行如下运算，若 IN1 等于 IN2，则输出为 0；否则输出为 1。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0	输出，数字型

举例：

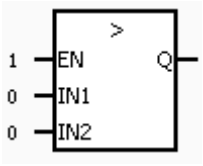
IN1=2；IN2=3.5；Q=1；

1.5.3. GT（大于）

简介：

该模块执行如下运算，若 IN1 大于 IN2，则输出为 1；否则输出为 0。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0	输出，数字型

举例：

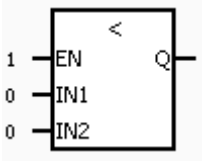
IN1=2；IN2=3.5；Q=0；

1.5.4. LT（小于）

简介：

该模块执行如下运算，若 IN1 小于 IN2，则输出为 1；否则输出为 0。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0	输出，数字型

举例：

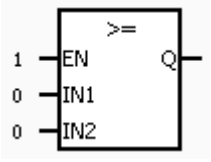
IN1=2；IN2=3.5；Q=1；

1.5.5. GE（大于等于）

简介：

该模块执行如下运算，若 IN1 大于等于 IN2，则输出为 1；否则输出为 0。

符号：



参数描述：

参数	初始值	含义

IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0	输出，数字型

举例：

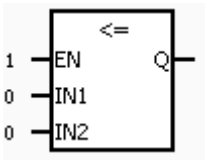
IN1=2；IN2=3.5；Q=0；

1.5.6. LE（小于等于）

简介：

该模块执行如下运算，若 IN1 小于等于 IN2，则输出为 1；否则输出为 0。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0	输出，数字型

举例：

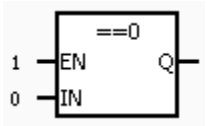
IN1=2；IN2=3.5；Q=1；

1.5.7. EQ0（等于零）

简介：

该模块执行如下运算，若 IN1 等于 0，则输出为 1；否则输出为 0。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0	输出，数字型

举例：

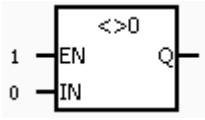
IN=2；Q=0；

1.5.8. NE0（不等于零）

简介：

该模块执行如下运算，若 IN1 不等于 0，则输出为 1；否则输出为 0。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0	输出，数字型

举例：

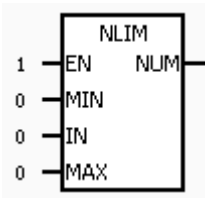
IN=2；Q=1；

1.5.9. NLIM（间隔号）

简介：

该模块判断输入值 INP 处于何种区域。MIN<MAX，若 INP<MIN，则输出为 1；若 INP>MAX，则输出为 3；否则输出为 2。

符号：



参数描述：

参数	初始值	含义
MIN	0.0	输入，模拟型
INP	0.0	输入，模拟型
MAX	0.0	输入，模拟型
Q	0	输出，整数型

举例：

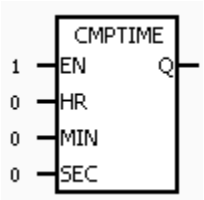
MIN=2.5；IN＝2；MAX=3.5；Q=1；

1.5.10. CMPTIME（时间比较）

简介：

该模块判断系统当前时间值是否等于用户设定输入 HR、MIN、SEC 表示的时间值，若等于输出为 1，否则输出为 0。

符号：



参数描述：

参数	初始值	含义
HR	0.0	输入，模拟型

MIN	0.0	输入，模拟型
SEC	0.0	输入，模拟型
Q	0	输出，数字型

举例：

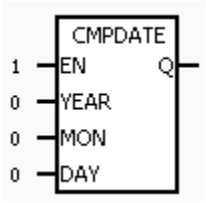
HR=23；MIN=35；SEC=5；Q=0；

1.5.11. CMPDATE（日期比较）

简介：

该模块判断系统当前日期值是否等于用户设定输入 YEAR、MON、DATE 表示的日期值，若等于输出为 1，否则输出为 0。

符号：



参数描述：

参数	初始值	含义
YEAR	0.0	输入，模拟型
MON	0.0	输入，模拟型
DAY	0.0	输入，模拟型
Q	0	输出，数字型

举例：

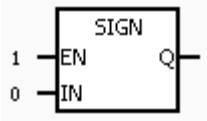
YEAR=2003；MON=11；DAY=5；Q=0；

1.5.12. SIGN（符号）

简介：

该模块执行如下运算，若输入值大于 0，则输出为 1；若输入值小于 0，则输出为－1，否则输出为 0。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q	0	输出，整数型

举例：

IN=2；Q=1；

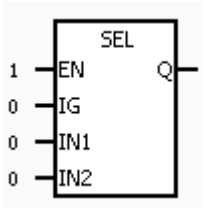
1.6. 选择

1.6.1. SEL（二选一）

简介：

该模块执行如下运算，若 IG 等于 0，则输出 Q=IN1，否则输出 Q=IN2。

符号：



参数描述：

参数	初始值	含义
IG	0	输入，模拟型
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

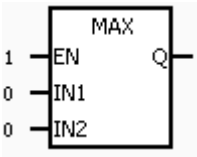
IG=0；IN1=2；IN2=3；Q=2；

1.6.2. MAX（最大值）

简介：

该模块选取输入的最大值作为输出，端子数可变，最多 16 个输入。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

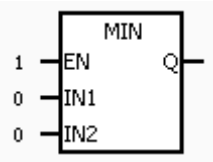
IN1=2；IN2=3；Q=3；

1.6.3. MIN（最小值）

简介：

该模块选取输入的最小值作为输出，端子数可变，最多 16 输入。

符号：



参数描述：

参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

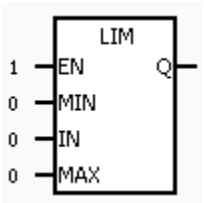
IN1=2；IN2=3；Q=2；

1.6.4. LIM（限值）

简介：

该模块执行如下运算，若输入 IN<MIN，则输出 Q=MIN；若输入 IN>MAX，则输出 Q=MAX；其余情况输入 Q=IN。

符号：



参数描述：

参数	初始值	含义
MIN	0.0	输入，最小值，模拟型
IN	0.0	输入，模拟型
MAX	0.0	输入，最大值，模拟型
Q	0.0	输出，模拟型

举例：

MIN=2.5；IN=2；MAX=5.6；Q=2.5；

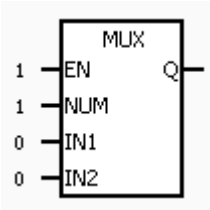
1.6.5. MUX（多选 1）

简介：

该模块执行如下运算，根据 NUM 值选取输入的某个作为输出；端子数可变，最多 16 输入。

该模块要求输入 NUM 必须为 1～16 的整数，否则输出保持上一个值不变。

符号：



参数描述：

参数	初始值	含义
NUM	0	输入，整数型
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

NUM=2；IN1＝2；IN2＝3；IN3=4；IN4=5；IN5=6；IN6=7；IN7=8；Q=3；

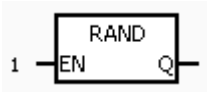
1.7. 信号发生器

1.7.1. RAND（随机值）

简介：

该模块随机产生 0～1 的数作为输出。

符号：



参数描述：

参数	初始值	含义
Q	0.0	输出，模拟型，0～1

举例：

Q=0.564；

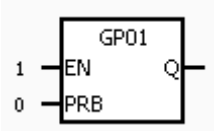
1.7.2. GP01（随机序列）

简介：

该模块以概率 PRB 输出 1，以概率 1－PRB 输出 0；

该模块要求输入 PRB 在 0～1 之间，否则输出为 1。

符号：



参数描述：

参数	初始值	含义
PRB	0.0	输入，模拟型，0～1
Q	1	输出，数字型

举例：

PRB=0.45；Q=1；

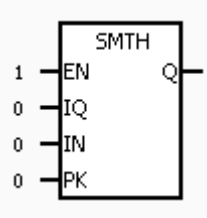
1.8. 信号处理

1.8.1. SMTH（一阶平滑）

简介：

该模块执行如下运算 Q=(1-1/PK)*IQ+IN/PK。

符号：



参数描述：

参数	初始值	含义
IQ	0.0	输入，模拟型
IN	0.0	输入，模拟型
PK	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IQ=2；IN=3；PK=1； Q=3；

1.8.2. VLIM（速度限制）

简介：

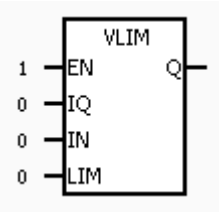
该模块执行如下运算：IF |IN－IQ|>LIM THEN Q=IQ+SIGN(IN－IQ)*LIM；

ELSE Q=IN；

一般情况下，该模块可将输入 IN 为当前输入，IQ 表示上一次采样周期时输入，则该模块的作用是判断输入的变化是否大于 LIM，若大于，则以 IQ+LIM 或 IQ-LIM 代替当前 IN 作为输出。

其中，SIGN 是符号算法块，即通过这个算法块来取一个正/负符号。

符号：



参数描述：

参数	初始值	含义
IQ	0.0	输入，模拟型
IN	0.0	输入，模拟型
LIM	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IQ=0； IN=2；LIM=3；Q=2；

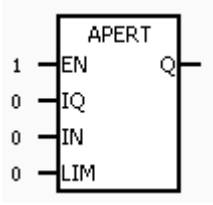
1.8.3. APERT（盲区）

简介：

该模块执行如下运算 IF |IN－IQ| < LIM THEN Q=IQ ELSE Q=IN。

一般情况下，IN 表示当前输入，IQ 为设定值。该模块判断输入的变化是否小于 LIM，若小于则输出等于上次值 IQ，否则输出为当前输入。

符号：



参数描述：

参数	初始值	含义
IQ	0.0	输入，模拟型
IN	0.0	输入，模拟型
LIM	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

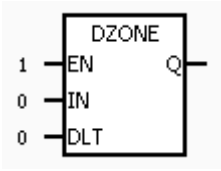
IN=2；IQ=3；LIM=0.5；Q=2；

1.8.4. DZONE（死区）

简介：

该模块执行如下运算：IF |IN|<DLT THEN Q=0 ELSE Q=IN。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
DLT	0.0	输入，模拟型
Q	0.0	输出，模拟型

举例：

IN=2；DLT=0.3；Q=2；

1.8.5. GETTIME（系统时间）

简介：

该模块获取系统当前时间相对于系统指定时刻的值，以毫秒为单位作为输出。

符号：

参数描述：

参数	初始值	含义
Q	0	输出，整数型

举例：

Q=22899540；

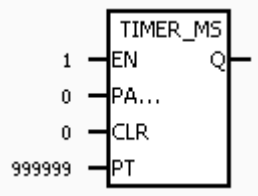
1.9. 定时器

1.9.1. TIMER_MS（毫秒定时器）

简介：

该模块产生从上次启动时的计时值，输出以毫秒为单位。PAUSE 为暂停计时，CLR 为复位端子，PT 为定时器计数限值，超过设定值，则定时器重新计时。

符号：



参数描述：

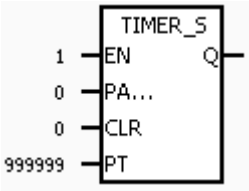
参数	初始值	含义
PAUSE	0	输入，数字型
CLR	0	输入，数字型
PT	999999	输入，整数型
Q	0	输出，整数型

1.9.2. TIMER_S（秒定时器）

简介：

该模块产生从上次启动时的计时值，输出以秒为单位。PAUSE 为暂停计时，CLR 为复位端子，PT 为定时器计数限值，超过设定值，则定时器重新计时。

符号：



参数描述：

参数	初始值	含义

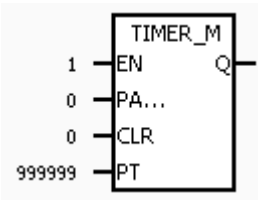
PAUSE	0	输入，数字型
CLR	0	输入，数字型
PT	999999	输入，整数型
Q	0	输出，整数型

1.9.3. TIMER_M（分钟定时器）

简介：

该模块产生从上次启动时的计时值，输出以分钟为单位。PAUSE 为暂停计时，CLR 为复位端子，PT 为定时器计数限值，超过设定值，则定时器重新计时。

符号：



参数描述：

参数	初始值	含义
PAUSE	0	输入，数字型
CLR	0	输入，数字型
PT	999999	输入，整数型
Q	0	输出，整数型

1.10. 流量处理

1.10.1. FLOWC（流量补偿）

简介：

该模块实现对输入流量的补偿，补偿公式如下：

⑴一般气体开方补偿

$$Y = X\sqrt{\frac{K\left(0.10136+P_i\right)}{273.15+T_i}}\quad ,\qquad 其中=K\frac{273.15+T_0}{0.10136+P_0}$$

⑵饱和蒸汽开方补偿

$$Y = X\sqrt{\frac{P_i+0.137411}{P_0+0.137411}}$$

⑶过热蒸汽开方补偿

$$Y = X\sqrt{\frac{K}{\frac{0.4619\left(T_i+273.15\right)}{P_i+0.10136}-9.7+0.0132T_i}}$$

其中
$$K=\frac{0.4619\left(T_0+273.15\right)}{P_0+0.10136}-9.7+0.0132T_0$$

(4)一般气体补偿

$$Y=\frac{K\left(0.10136+P_i\right)X}{273.15+T_i} \qquad \text{其中 } K=\frac{T_0+273.15}{P_0+0.10136}$$

(5)饱和蒸汽补偿

$$Y=\frac{X\left(P_i+0.137411\right)}{P_0+0.137411}$$

(6)过热蒸汽补偿

$$Y=\frac{KX}{0.4619\left(T_i+273.15\right)/\left(P_i+0.10136\right)-9.7+0.0132T_i}$$

其中 $K=0.4619\left(T_0+273.15\right)/\left(P_0+0.10136\right)-9.7+0.0132T_0$

P_i：压力输入，MPa

T_i：温度输入，℃

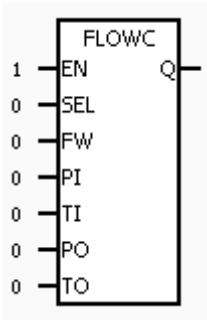
P_0：设计额定工作压力， MPa

T_0：设计额定工作温度，℃

X：流量输入

Y：补偿后流量

符号：



参数描述：

参数	初始值	含义
SEL	0	输入选择端子，为1~6分别对应上面所列的1～6种计算公式，整数型
FW	0.0	输入，流量，模拟型
PI	0.0	输入，压力，模拟型
TI	0.0	输入，温度，模拟型
PO	0.0	输入，额定压力，模拟型
TO	0.0	输入，额定温度，模拟型
Q	0.0	输出，补偿后流量，模拟型

1.10.2. FLOW_A（一般气体开方补偿）

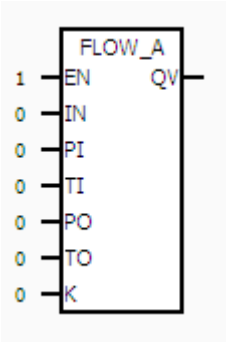
简介：

一般气体开方补偿,适用于差压性质的流量计,如孔板流量计。

该模块的计算公式为：

$$QV = IN * \frac{K}{\sqrt{0.431}} * \sqrt{\frac{273.15+TO}{0.10136+PO} * \frac{0.10136+PI}{273.15+TI}}$$

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，流量，模拟型
PI	0.0	输入，压力，模拟型
TI	0.0	输入，温度，模拟型
PO	0.0	输入，额定压力，模拟型
TO	0.0	输入，额定温度，模拟型
K	0.0	输入，仪表流量转换系数，模拟型
QV	0.0	输出，补偿后流量，模拟型

1.10.3. FLOW_B（一般气体不开方补偿）

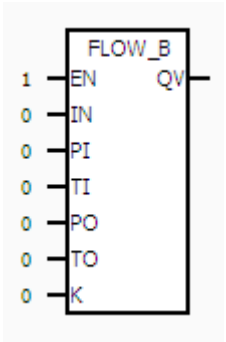
简介：

一般气体不开方补偿, 适用于容积式性质的流量计, 如涡节流量计

该模块的计算公式为：

$$QV = IN * K * \frac{273.15+TO}{0.10136+PO} * \frac{0.10136+PI}{273.15+TI}$$

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，流量，模拟型

PI	0.0	输入，压力，模拟型
TI	0.0	输入，温度，模拟型
PO	0.0	输入，额定压力，模拟型
TO	0.0	输入，额定温度，模拟型
K	0.0	输入，仪表流量转换系数，模拟型
QV	0.0	输出，补偿后流量，模拟型

1.10.4. FLOW_C（饱和蒸汽开方补偿）

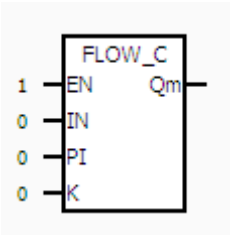
简介：

饱和蒸汽开方补偿, 适用于差压性质的流量计, 如孔板流量计

该模块的计算公式为：

$$Qm = IN * K * \sqrt{\rho} \qquad \rho = f(PI)$$

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，流量，模拟型
PI	0.0	输入，压力，模拟型
K	0.0	输入，仪表流量转换系数，模拟型
Qm	0.0	输出，补偿后流量，模拟型

1.10.5. FLOW_D（饱和蒸汽不开方补偿）

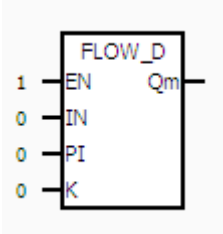
简介：

饱和蒸汽不开方补偿, 适用于容积式性质的流量计, 如涡节流量计该

该模块的计算公式为：

$$Qm = IN * K * \rho \qquad \rho = f(PI)$$

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，流量，模拟型
PI	0.0	输入，压力，模拟型
K	0.0	输入，仪表流量转换系数，模拟型
Qm	0.0	输出，补偿后流量，模拟型

1.10.6. FLOW_E（过热蒸汽开方补偿）

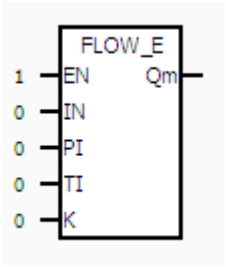
简介：

过热蒸汽开方补偿, 适用于差压性质的流量计, 如孔板流量计

该模块的计算公式为：

$$Qm = IN * K * \sqrt{\rho} \qquad \rho = f(PI, TI)$$

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，流量，模拟型
PI	0.0	输入，压力，模拟型
TI	0.0	输入，温度，模拟型
K	0.0	输入，仪表流量转换系数，模拟型
Qm	0.0	输出，补偿后流量，模拟型

1.10.7. FLOW_F（过热蒸汽不开方补偿）

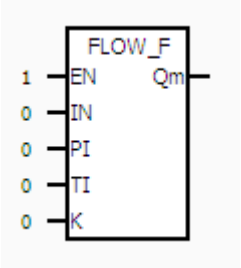
简介：

过热蒸汽不开方补偿, 适用于容积式性质的流量计, 如涡节流量计

该模块的计算公式为：

$$Qm = IN * K * \rho \qquad \rho = f(PI, TI)$$

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，流量，模拟型
PI	0.0	输入，压力，模拟型
TI	0.0	输入，温度，模拟型
K	0.0	输入，仪表流量转换系数，模拟型
Qm	0.0	输出，补偿后流量，模拟型

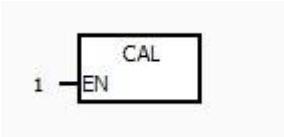
1.11. 调用功能块

1.11.1. CAL 调用功能块

简介：

该模块实现如下功能：双击该模块，在弹出的对话框中选择要调用的子程序，则实现了对子程序的调用。

符号：



2. 功能块库

2.1. 逻辑

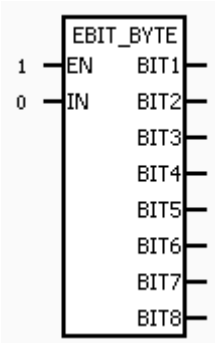
2.1.1. EBIT_BYTE（单字节位展开）

简介：

该模块执行如下运算：将输入值 IN（十进制）的后 8 位的值分别输出到 BIT1~BIT8；

该模块要求输入 IN 的值为整数，否则对其进行取整(忽略小数)。

符号：



描述：

参数	初始值	含义
IN	0	输入，整数型
BIT1	0	输出，数字型、最低位

BIT2	0	输出，数字型
BIT3	0	输出，数字型
BIT4	0	输出，数字型
BIT5	0	输出，数字型
BIT6	0	输出，数字型
BIT7	0	输出，数字型
BIT8	0	输出，数字型、最高位

举例：

IN=0X35；BIT1=1；BIT2=0；BIT3=1；BIT4=0；

BIT5=1；BIT6=1；BIT7=0；BIT8=0；

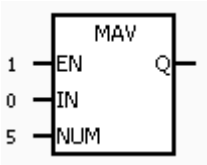
2.2. 代数

2.2.1. MAV（移动平均）

简介：

该模块将输入 IN 的前 NUM 个周期的平均值，NUM 取值范围是 1~1200。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，模拟型
NUM	5	输入，整数型
Q	0.0	输出，模拟型

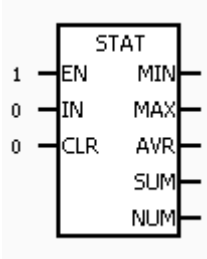
2.2.2. STAT（统计值）

简介：

该模块统计输入 IN 在上次复位之后的最大值，最小值，平均值以及和。

输入 CLR 为复位标志。当 CLR 为 1 时，所有数据复位。

符号：



描述：

参数	初始值	含义

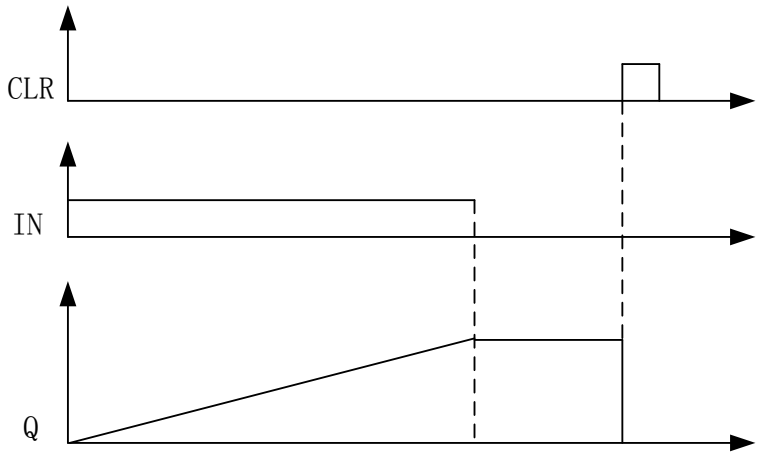
IN	0.0	输入，采样值，模拟型
CLR	0	输入，复位端子，数字型
MIN	0.0	输出，最小值，模拟型
MAX	0.0	输出，最大值，模拟型
AVR	0.0	输出，平均值，模拟型
SUM	0.0	输出，和，模拟型
NUM	0.0	输入，统计输入个数，模拟型

2.2.3. INTG（积分）

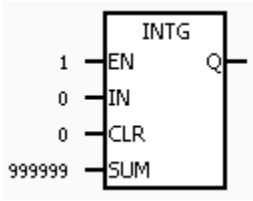
简介：

该模块运算在上次复位之后输入的积分值，输出最大值 SUM 为 999999；当超过该值时，输出从 0 开始积分，输出值是输入值的 1/2，积分输入 IN 以秒为单位。

输入 CLR 为复位标志。当 CLR 为 1 时，所有数据复位。



符号：



描述：

参数	初始值	含义
IN	0.0	输入，采样值，模拟型
CLR	0	输入，复位标志、数字型
SUM	999999.0	输入，模拟型
Q	0	输出，积分值，数字型

2.2.4. DIFF（导数）

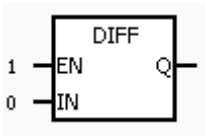
简介：

该模块计算输入的导数。

$$Q = \frac{IN_0 - IN_1}{T}$$

其中 IN_0 为当前输入，IN_1 为上个周期的输入值，T 为程序运行周期。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，采样值，模拟型
Q	0.0	输出，导数，模拟型

2.2.5. DF3（三点导数）

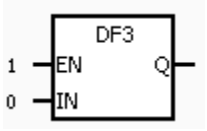
简介：

该模块采用三个采样周期的值来计算导数，其计算公式为：

$$Q = \frac{1.5 \times IN_0 - 2 \times IN_1 + 0.5 \times IN_2}{T}$$

其中 IN_0 为当前输入，IN_1 为上个周期的输入值，IN_2 为上上个周期的输入值，T 为运算周期。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，采样值，模拟型
Q	0.0	输出，导数，模拟型

2.2.6. DDF（二阶导数）

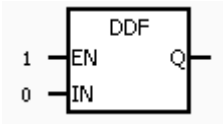
简介：

该模块计算某个点的二阶导数，其计算公式为：

$$Q = \frac{IN_0 - 2 \times IN_1 + IN_2}{T^2}$$

其中 IN_0 为当前输入，IN_1 为上个周期的输入值，IN_2 为上上个周期的输入值，T 为运算周期。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，采样值，模拟型
Q	0.0	输出，导数，模拟型

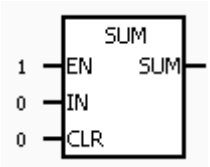
2.2.7. SUM（累加）

简介：

该模块计算输入在上次复位之后的和作为输出

$SUM = (IN_0 + IN_1 + IN_2 + \cdots)$。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，模拟型
CLR	0	输入，复位标志，数字型
SUM	0.0	输出，和，模拟型

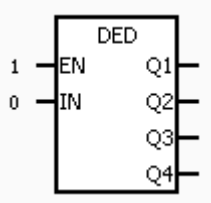
2.3. 选择

2.3.1. DED（纯滞后）

简介：

该模块将输入前三个周期以及当前周期的值作为输出分别赋给 Q1~Q4。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，采样值，模拟型
Q1	0.0	输出，当前周期值，模拟型
Q2	0.0	输出，上一个周期值，模拟型

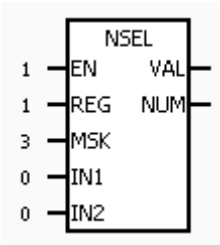
Q3	0.0	输出，上两个周期值，模拟型
Q4	0.0	输出，上三个周期值，模拟型

2.3.2. NSEL（屏蔽选择）

简介：

选择 IN1…INn 中的值赋给 VAL，NUM 输出被选择输入的序号。输入 REG 和 MSK 对选择进行控制。如果输入 REG 等于 0，则输出取输入中的最小值，如相反则取最大值。输入 MSK 的前 16 个位对选择起作用。如果那个位等于 1，则与其序号对应的输入将能参与选择，NUM 为输出 VAL 值所对应的序号，如果那个位为 0 则不参与。如果所有输入都被 MSK 封闭，则输出 NUM 值为 0。

符号：



描述：

参数	初始值	含义
REG	1	输入，控制端子，数字型
MSK	0	输入，选择端子，整数型
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
VAL	0.0	输出，选择结果，模拟型
NUM	0	输出，选中序号，整数型

举例：

REG=0；MSK=7；IN1=2.5；IN2=3.6；IN3=3；IN4=1.2；IN5=4；VAL=2.5；NUM=1；

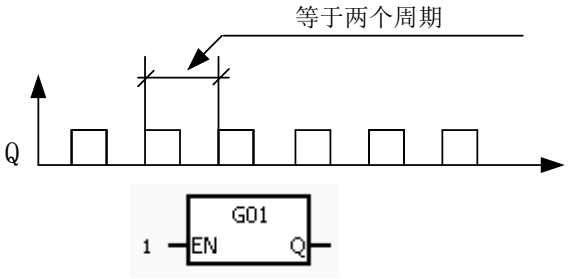
2.4. 信号发生器

2.4.1. G01（位振荡）

简介：

该模块产生周期为 1 的脉冲作为输出，输出值为 0 和 1，周期为 1 秒钟。

符号：



描述：

参数	初始值	含义
Q	0	输出，数字型

2.4.2. G_DWORD（双字循环移位）

简介：

该模块产生一个双字的数，其位被循环置位，其运算初始值为 1，最终产生一个 2 的 31 次方的数。

符号：

描述：

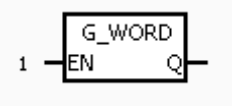
参数	初始值	含义
Q	1	输出，整数型

2.4.3. G_WORD（字循环移位）

简介：

该模块产生一个单字的数，其位被循环置位，其运算初始值为 1，最终产生一个 2 的 15 次方的数。

符号：



描述：

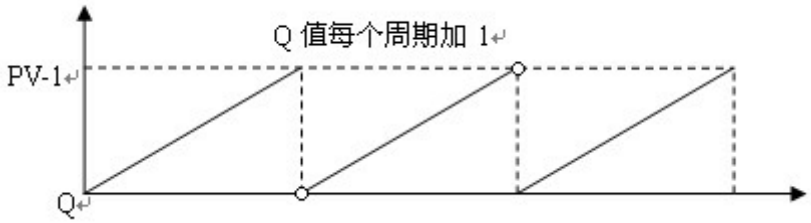
参数	初始值	含义
Q	1	输出，整数型

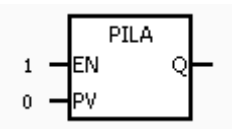
2.4.4. PILA（线性增长）

简介：

该模块产生峰值为 PV，每个运算周期加 1 的锯齿波，PV 值最好不要为 1，因为为 1 的情况下看不出变化，Q 值为从 0 开始累加，当 Q=PV 自动清零进入下个循环。

符号：





描述：

参数	初始值	含义
PV	0.0	输入，峰值，模拟型
Q	0.0	输出，模拟型

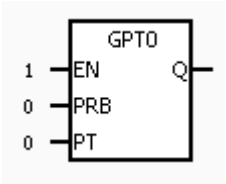
2.4.5. GPT0（0-1 序列）

简介：

该模块输出 0、1 序列。输出 1 的概率由 PRB 给定，PT 为 1 的延续时间。

该模块要求输入 PRB 在 0～1 之间，否则输出维持不变，PT 必须是个大于 0 的数。

符号：



描述：

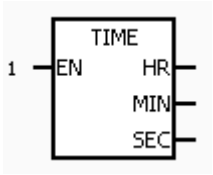
参数	初始值	含义
PRB	0.0	输入、1出现的概率（取值0~1之间），模拟型
PT	0	1出现后的持续时间（单位秒），整数型
Q	0	输出，数字型

2.4.6. TIME（时间）

简介：

该模块获取系统当前时间，以时、分、秒的方式输出。

符号：



描述：

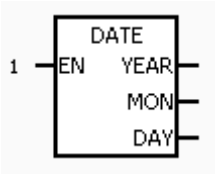
参数	初始值	含义
HR	0.0	输出，时间，模拟型
MIN	0.0	输出，分钟，模拟型
SEC	0.0	输出，秒，模拟型

2.4.7. DATE（日期）

简介：

该模块获取系统当前日期输出，以年、月、日的方式输出。

符号：



描述：

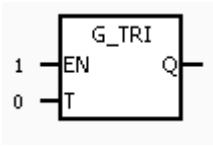
参数	初始值	含义
YEAR	0.0	输出，年，模拟型
MON	0.0	输出，月，模拟型
DAY	0.0	输出，日，模拟型

2.4.8. G_TRI（三角波）

简介：

该模块产生周期为 T 的三角波作为输出。

符号：



描述：

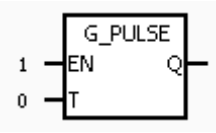
参数	初始值	含义
T	0.0	输入，周期，模拟型
Q	0.0	输出，三角波，模拟型

2.4.9. G_PULSE（方波）

简介：

该模块产生周期为 T 的方波作为输出。T 等于 0 时 Q 输出两个扫描周期的脉冲。

符号：



描述：

参数	初始值	含义
T	0	输入，周期，整数型
Q	0	输出，方波，数字型

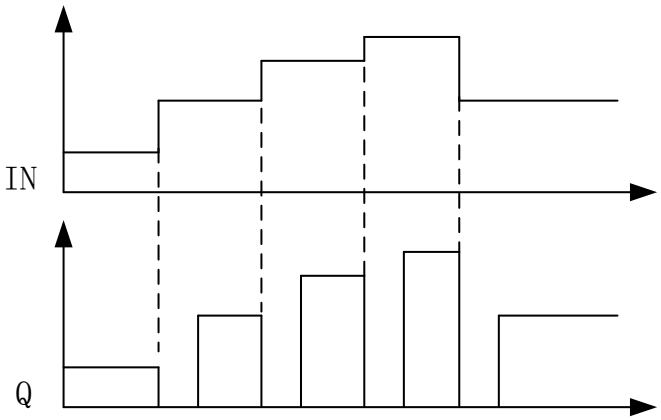
2.5. 信号处理

2.5.1. INS0（插入 0）

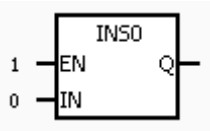
简介：

该模块当输入发生变化瞬间，插入一个周期 0，其余时刻输出等于输入。

符号：



当IN值变化后，Q值下一个周期为零，其余时间Q=IN



描述：

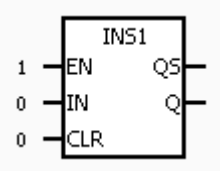
参数	初始值	含义
IN	0.0	输入，模拟型
Q	0.0	输出，模拟型

2.5.2. INS1（插入 1）

简介：

CLR 为 0 时，当输入 IN 发生变化瞬间，输出 QS 插入一个周期 1，输出 Q 等于输入 IN；CLR 从 0 变换为 1 时，该模块的所有输出为 0；CLR 从 1 变换为 0 时，输出 QS 插入一个周期 1，输出 Q 等于输入 IN。

符号：



描述：

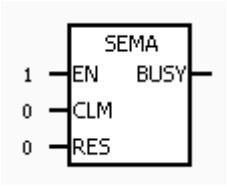
参数	初始值	含义
IN	0.0	输入，模拟型
CLR	0	输入，数字型
QS	0	输出，开关型
Q	0.0	输出，模拟型

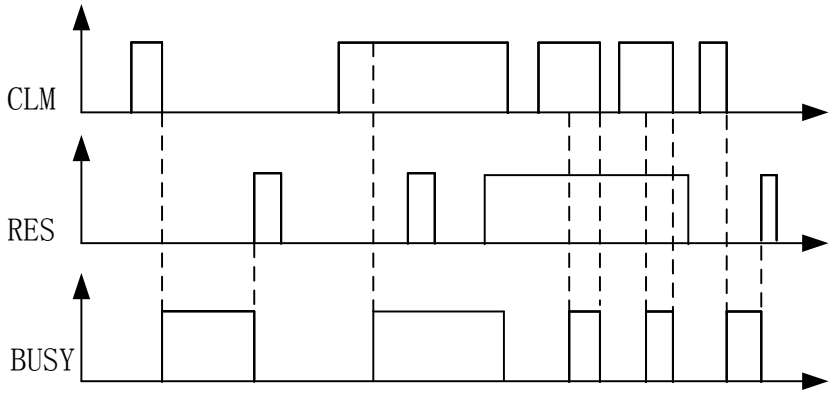
2.5.3. SEMA（信号器）

简介：

该模块实现当输入 RES 非 0 时，输出 BUSY 为 0。当输入 RES 为 0，输入 CLM 为 1 时，输出 BUSY 延迟一个周期为 1，其余时刻输出保持不变。

符号：





当CLM为1时，BUSY延时一个周期置1

描述：

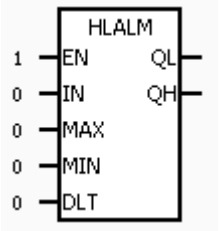
参数	初始值	含义
CLM	0	输入，数字型
RES	0	输入，数字型
BUSY	0	输出，数字型

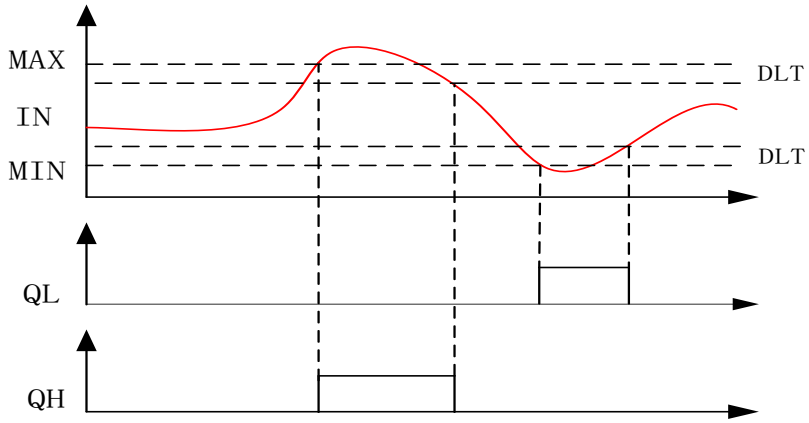
2.5.4. HLALM（高低限报警）

简介：

该模块当超过上限 MAX 时 QH 输出 1，如 IN 不小于 MAX－DLT，这一输出值不发生变化。当低于下限 MIN 时 QL 的值等于 1，当 IN 不大于 MIN＋DLT 时，这一输出值不发生变化。

符号：





描述：

参数	初始值	含义
IN	0.0	输入，模拟型
MAX	0.0	输入，最大值，模拟型
MIN	0.0	输入，最小值，模拟型
DLT	0.0	输入，滞区值，模拟型
QH	0	输出，高限报警输出，数字型
QL	0	输出，低限报警输出，数字型

2.5.5. ALARM（报警）

简介：

该模块根据设定的高高限、高限、低限、低低限，以及滞区值输出当前的报警状态。

输出值Q=1表示当前处于高限报警；

输出值Q=2表示当前处于高高限报警；

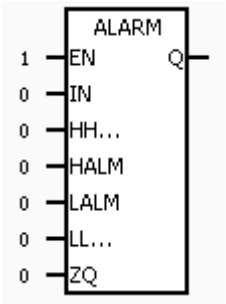
输出值Q=3表示当前处于低限报警；

输出值Q=4表示当前处于低低限报警；

输出值Q=0表示当前输入处于正常状态。

ZQ 是实际值与限值相比，在这个值的范围内变化报警不进行理会，只有超出才报警或者恢复，免得在限值的很小范围内波动时，报警过于频繁。

符号：



描述：

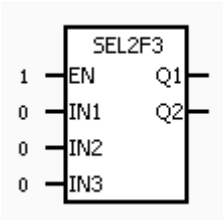
参数	初始值	含义
IN	0.0	输入，模拟型
HHALM	0.0	输入，高高限，模拟型
HALM	0.0	输入，高限，模拟型
LALM	0.0	输入，低限，模拟型
LLALM	0.0	输入，低低限，模拟型
ZQ	0.0	输入，滞区，模拟型
Q	0	输出0、1、2、3、4数值，报警状态，整数型

2.5.6. SEL2F3（三选二）

简介：

该模块实现从三个输入 IN1~IN3 中选择最接近的两个作为输出到 Q1~Q2，是 N1~N3 之间差值最小的两个数。若输入值之间的差值一样时，输出 3 个数中较小的 2 个。

符号：



描述：

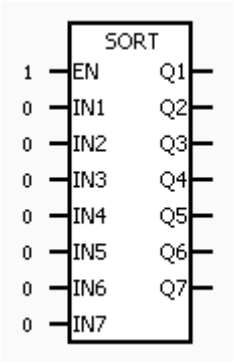
参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
IN3	0.0	输入，模拟型
Q1	0.0	输出，模拟型
Q2	0.0	输出，模拟型

2.5.7. SORT（排序）

简介：

该模块对当前输入 IN1~IN7 按照从小到大的顺序排序作为输出。最大 15 个排序。

符号：



描述：

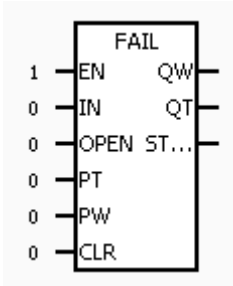
参数	初始值	含义
IN1	0.0	输入，模拟型
IN2	0.0	输入，模拟型
IN3	0.0	输入，模拟型
IN4	0.0	输入，模拟型
IN5	0.0	输入，模拟型
IN6	0.0	输入，模拟型
IN7	0.0	输入，模拟型
Q1	0.0	输出，最小值，模拟型
Q2	0.0	输出，模拟型
Q3	0.0	输出，模拟型
Q4	0.0	输出，模拟型
Q5	0.0	输出，模拟型
Q6	0.0	输出，模拟型
Q7	0.0	输出，最大值，模拟型

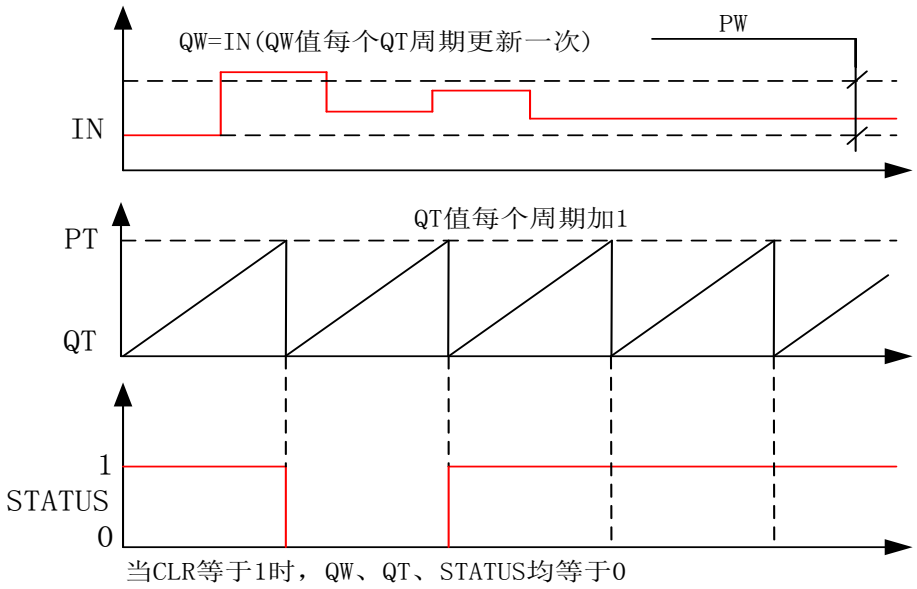
2.5.8. FAIL（输入变化判断）

简介：

该模块判断输入值当输入 OPEN 处于 1 的情况下，输入 IN 在 PT 个时间内的增量的绝对值是否大于 PW，若大于则输出 STATUS 为 0，否则输出 STATUS 为 1。输出 QT 为时间计数，输出 QW 为 QT=PT 时刻的输入 IN 的值。CLR 为复位端子。OPEN 为 0 的情况下为停止状态。

符号：





描述：

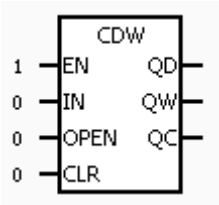
参数	初始值	含义
IN	0.0	输入，采样值，模拟型
OPEN	0	输入，开启状态，数字型
PT	0	输入，计数，整数型
PW	0.0	输入，变化量，模拟型
CLR	0	输入，复位端子，数字型
QW	0.0	输出，PT时刻采样值，模拟型
QT	0	输出，时间计数，整数型
STATUS	0	输出，状态，数字型

2.5.9. CDW（输入增量值）

简介：

该模块用来计算输入 IN 在阀门处于开状态时的采样值的变化量，以绝对值作为输出到 QD。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，采样值，模拟型
OPEN	0	输入，阀门开启状态，数字型
CLR	0	输入，复位端子，数字型
QD	0.0	输出，实际变化量，模拟型
QW	0.0	输出，OPEN从关到开瞬间的输入IN的值，模拟型
QC	0	输出，当前阀门状态的值，数字型

2.5.10. DTIME（转成时间）

简介：

输入的 IN 值转化成日时分秒相加的一个值。

符号：

描述：

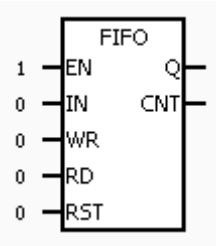
参数	初始值	含义
IN	0	输入，秒，整数型
DAY	0.0	输出，天，模拟型
HOUR	0.0	输出，小时，模拟型
MIN	0.0	输出，分，模拟型
SEC	0.0	输出，秒，模拟型

2.5.11. FIFO（数据序列生成器）

简介：

该算法块生成先入先出的数据序列，最大可以压入 24 个数据序列。当 WR 由低变高（上升沿）时，若 CNT<24，则将 IN 的值同步压入数据序列中，同时数据序列长度 CNT 加 1。当 RD 由低变高（上升沿）时，若 CNT>=1，则数据序列中最先压入的数据组弹出至 Q，同时将 CNT 减 1；否则，输出 Q==0.0。当 RST=1 时清空算法块中的数据序列，同时 Q==0.0、CNT==0.0。

符号：



描述：

参数	初始值	含义
IN	0.0	输入，模拟型
WR	0	输入，数字型
RD	0	输入，数字型
RST	0	输入，数字型
Q	0.0	输出，模拟型
CNT	0.0	输出，模拟型

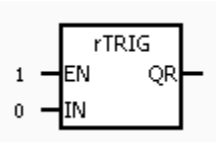
2.6. 触发器

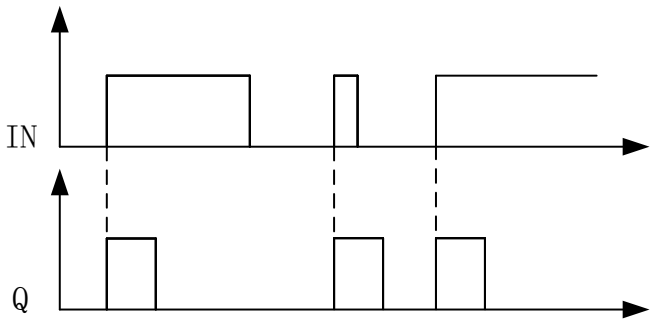
2.6.1. rTRIG（上升沿）

简介：

该模块当输入从 OFF 跳变到 ON 时，输出为 ON，否则输出为 OFF。QR 的变化是 IN 值由 0 到 1 变化时的一个瞬时值。

符号：





Q输出一个周期宽度的脉冲

描述：

参数	初始值	含义
IN	0	输入，数字型
QR	0	输出，数字型

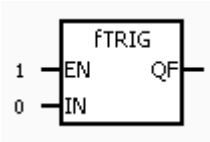
2.6.2. fTRIG（下降沿）

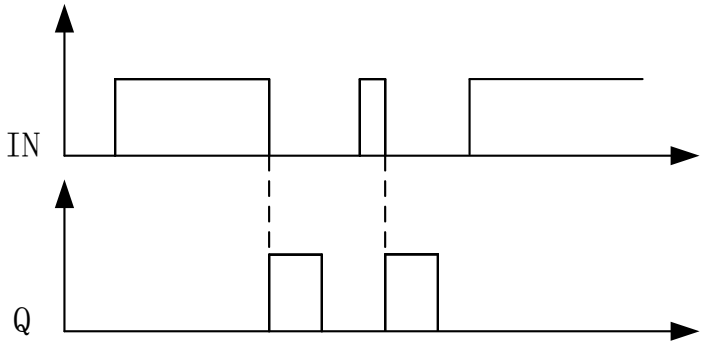
简介：

该模块当输入从 ON 跳变到 OFF 时，输出为 ON，否则输出为 OFF。QF 的变化是 IN

值又 1 到 0 变化时的一个瞬时值。

符号：





Q输出一个周期宽度的脉冲

描述：

参数	初始值	含义
IN	0	输入，数字型
QF	0	输出，数字型

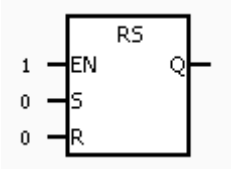
2.6.3. RS（RS 触发器）

简介：

该模块实现如下运算：假设 Q’指上一个运算周期的模块输出值，操作时可参考下表。

S	R	Q
1	0	1
1	1	0
0	0	Q’
0	1	0

符号：



描述：

参数	初始值	含义
S	0	输入，置位端子，数字型
R	0	输入，复位缎子，数字型
Q	0	输出，数字型

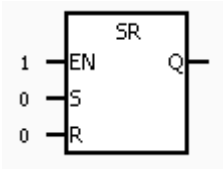
2.6.4. SR（SR 触发器）

简介：

该模块实现如下运算：假设 Q’指上一个运算周期的模块输出值，操作时可参考下表。

S	R	Q
1	0	1
1	1	1
0	0	Q’
0	1	0

符号：



描述：

参数	初始值	含义
S	0	输入，置位端子，数字型
R	0	输入，复位端子，数字型
Q	0	输出，数字型

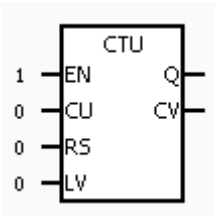
2.7. 计数器

2.7.1. CTU（递增计数器）

简介：

该模块当 CU 从 0 变为 1（上升沿）时，输出 CV+1；当 CV=LV 时，不再计数，输出 Q=1；当复位端子 RS=1 时，输出 CV=0，Q=0，LV 是个计数定值。

符号：



描述：

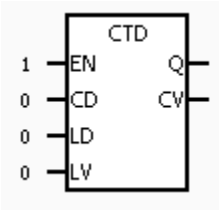
参数	初始值	含义
CU	0	输入，计数脉冲，数字型
RS	0	输入，复位端子，数字型
LV	0	输入，设定计数值，整数型
Q	0	输出，数字型
CV	0	输出，当前计数值，整数型

2.7.2. CTD（递减计数器）

简介：

由复位端子 LD 控制的，操作时首先对算法块进行复位 LD=1，让 CV=LV 再次进行复位 LD=0，此时开始执行递减运算。当 CD 从 0 变为 1（上升沿）时，输出 CV-1。

符号：



描述：

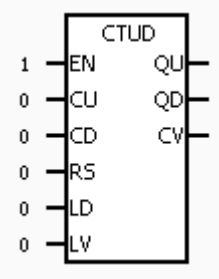
参数	初始值	含义
CD	0	输入，计数脉冲，数字型
LD	0	输入，复位端子，数字型
LV	0	输入，设定计数值，整数型
Q	0	输出，数字型
CV	0	输出，当前计数值，整数型

2.7.3. CTUD（递增递减计数器）

简介：

该模块实现了增计数器、减计数器和的功能。当输入 CU 触发一个上升沿时，输出 CV+1；当输入 CD 触发一个上升沿时，输出 CV-1；当复位端子 RS＝1 时，输出 CV=0；当复位端子 LD=1 时，输出 CV=LV；当输出 CV＝LV 时，输出 QU＝1；当输出 CV=0 时，输出 QD=1。

符号：



描述：

参数	初始值	含义
CU	0	输入，增计数脉冲，数字型
CD	0	输入，减计数脉冲，数字型
RS	0	输入，增复位端子，数字型
LD	0	输入，减复位端子，数字型
LV	0	输入，设定计数值，数字型
QU	0	输出，增计数输出，数字型

QD	0	输出，减计数输出，数字型
CV	0	输出，当前计数值，整数型

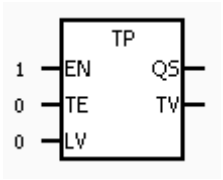
2.8. 计时器

2.8.1. TP（开记忆计时器）

简介：

LV 为定量位，当 TE 的值为 1 时 TV 开始计数，当 TV=LV 时，TV 保持不变，输出 QS 为 0，当 QS=0 时，若输入 TE=0，则 TV=0。

符号：



描述：

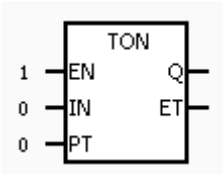
参数	初始值	含义
IN	0	输入，计时控制脉冲，数字型
LV	0	输入，设定计时值，整数型
QS	0	输出，计时输出，数字型
TV	0	输出，当前计时值，整数型

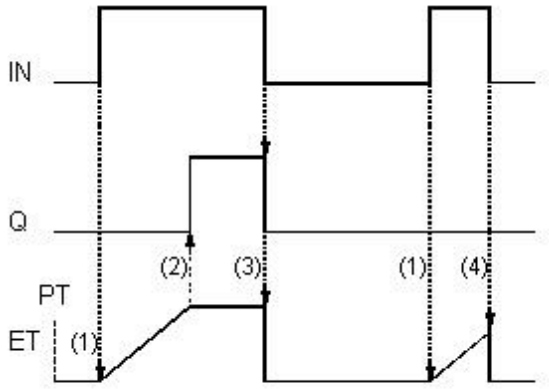
2.8.2. TON（ON 延迟）

简介：

该模块当输入从 0 跳变到 1 时输出 Q 延时 PT 个周期输出 1。若在此期间输入 IN 变化为 OFF，则输出 Q 仍为 0，在输入再次发生 0-1 的跳变时，重新延时输出。

符号：





1. 如果IN为ON，内部时钟 ET 启动（增幅半秒），延时开始。
2. 一旦内部时钟 ET 达到 PT 值，Q变为ON。
3. 如果IN变为OFF，Q变为OFF，ET=0。
4. 如果IN在 ET 达到 PT 值前变为OFF,则 ET=0。

描述：

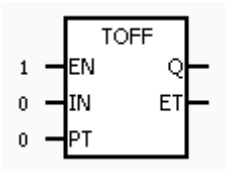
参数	初始值	含义
IN	0	输入，计时控制脉冲，数字型
PT	0	输入，设定计时值，整数型
Q	0	输出，计时输出，数字型
ET	0	输出，当前计时值，整数型

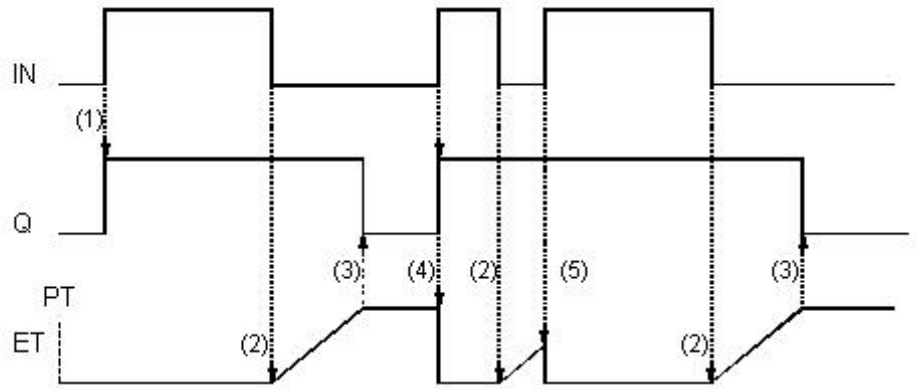
2.8.3. TOFF（OFF 延迟）

简介：

该模块当输入 IN 从 ON 跳变到 OFF 时，输出 Q 延时 PT 个周期输出 OFF。若在此期间输入 IN 变化为 ON，则输出 Q 仍为 ON，在输入再次发生 ON－OFF 的跳变时，重新延时输出。

符号：





1. 如果IN为ON，则Q为ON。
2. 如果IN变为OFF， 内部时钟ET将启动（增幅半秒），延时开始。
3. 当内部时钟ET达到PT值时，Q将变为OFF。
4. 如果IN变为ON，则Q变为ON，且内部时钟ET=0。
5. 如果IN在ET达到PT值之前变为ON，则ET=0。

描述：

参数	初始值	含义
IN	0	输入，计时控制脉冲，数字型
PT	0	输入，设定计时值，整数型
Q	0	输出，计时输出，数字型
ET	0	输出，当前计时值，整数型

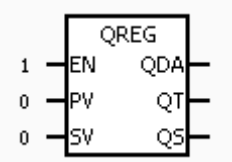
2.9. 控制

2.9.1. QREG（控制质量分析）

简介：

该模块主要实现对控制效果的描述。PV 为测量值，SV 为设定值。QDA 为误差绝对值的最大值，QT 为周期数，QS 为误差累计值。

符号：



描述：

参数	初始值	含义

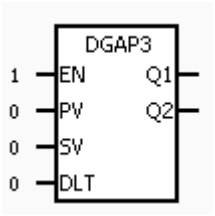
PV	0.0	输入，测量值，模拟型
SV	0.0	输入，设定值，模拟型
QDA	0.0	输出，误差最大值，模拟型
QT	0	输出，周期数，整数型
QS	0.0	输出，误差累计值，模拟型

2.9.2. DGAP3(二位式三状态控制)

简介：

该模块是一种二位式差隙调节器，用于三状态控制应用场合。

符号：



当 PV-SV>DLT(DLT 为正数)，Q2=1、Q1=0。

当 SV-PV>DLT(DLT 为正数)，Q2=0、Q1=1。其余时间 Q1、Q2 均为零。

描述：

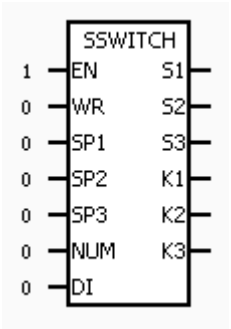
参数	初始值	含义
PV	0.0	输入，测量值，模拟型
SV	0.0	输入，设定值，模拟型
DLT	0.0	输入，滞区值，模拟型
Q1	0	输出，增加开关，数字型
Q2	0	输出，减少开关，数字型

2.9.3. SSWITCH(状态切换)

简介：

该模块实现输出在多种状态下的切换。NUM 决定将输入 DI 的值输出到 Si 和 Ki 中的哪一个。当 WR<=SPi 时，输出 Si=DI，Ki=0；否则，输出 Si=0，Ki=DI。

符号：



描述：

参数	初始值	含义
WR	0.0	输入，测量值，模拟型
SP1	0.0	输入，设定值1，模拟型
SP2	0.0	输入，设定值2，模拟型
SP3	0.0	输入，设定值3，模拟型
NUM	0	输入，选择序号，整数型1～3
DI	0.0	输入，选中输出设定值，模拟型
S1	0.0	输出，关状态1，模拟型；
S2	0.0	输出，关状态2，模拟型；
S3	0.0	输出，关状态3，模拟型；
K1	0.0	输出，开状态1，模拟型；
K2	0.0	输出，开状态2，模拟型；
K3	0.0	输出，开状态3，模拟型；

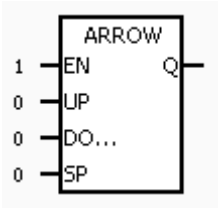
2.9.4. ARROW（上升/下降）

简介：

该模块通过两个控制端子 UP 和 DOWN 来实现输出为 SP 还是 SP 的相反数。

UP	DOWN	SP	Q
1	0	V1	V1
1	1	V1	V1
0	0	V1	0
0	1	V1	-V1

符号：



描述：

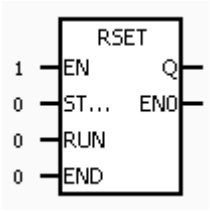
参数	初始值	含义
UP	0	输入，增控制端子，数字型
DWON	0	输入，减控制端子，数字型
SP	0.0	输入，设定值，模拟型
Q	0.0	输出，模拟型

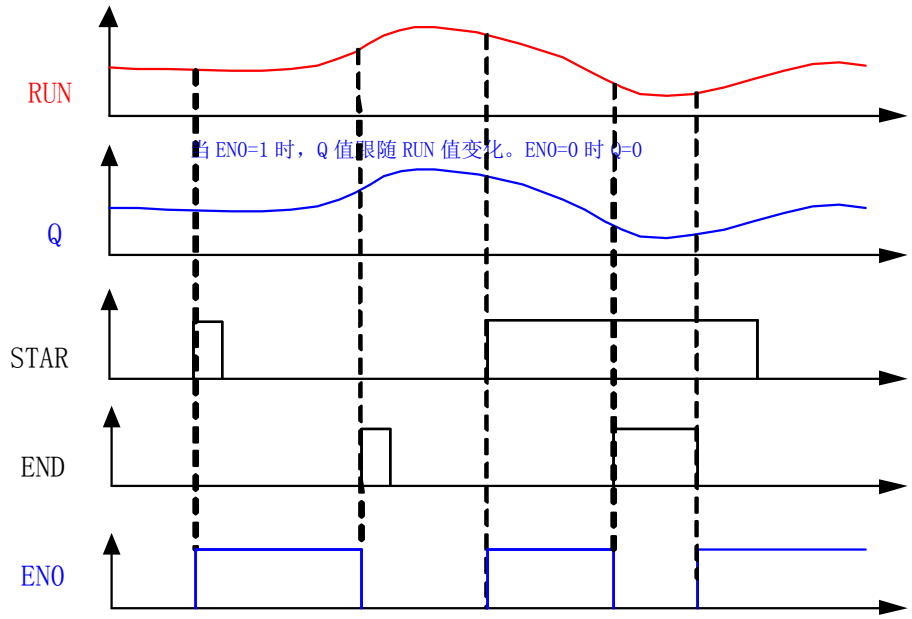
2.9.5. RSET(变频控制)

简介：

该模块实现如下功能：当输入 START 从 OFF 变为 ON 后，输出 Q=RUN；当输入 END=1 后，输出 Q=0。

符号：





描述：

参数	初始值	含义
START	0	输入，启动，数字型
RUN	0.0	输入，设定，模拟型
END	0	输入，停止，数字型
Q	0.0	输出，电流输出，模拟型
ENO	0	输出，启动状态，数字型

2.9.6. PID(无扰 PID)

简介：

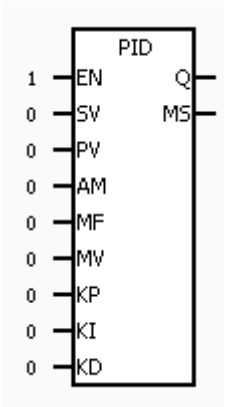
该模块的计算公式为：

$$IN = SV - PV$$

$$Q = KP \times \left(IN_0 + KI \times \int INdt + KD \times \left(IN_0 - IN_1 \right) / T \right);$$

SV 为设定值；PV 为测量值；PS 为正/反作用设置值，0 表示反作用，1 表示正作用；AM 为手动/自动设置值，自动状态 AM=0，手动状态 AM=1；MF 为阀位输出反馈值；MV 为阀位手动设定值；KP、TI、TD 分别为比例系数、积分系数、微分系数；Q 为阀位输出值，取值范围为 0.0~1.0；MS 为阀位输出跟踪值。

符号：



描述：

参数	初始值	含义
SV	0.0	输入，设定值，模拟型
PV	0.0	输入，测量值，模拟型
AM	0	输入，手动/自动设置值，0表示自动，1表示手动，数字型
MF	0.0	输入，阀位输出反馈值，模拟型
MV	0.0	输入，手动设定值，模拟型
KP	0.0	输入，比例系数，模拟型
KI	0.0	输入，积分系数，模拟型
KD	0.0	输入，微分系数，模拟型
IS	999999.9	输入，积分分离设定值，模拟型
DE	0.0	输入，输出死区设定置，模拟型
MH	1.0	输入，输出高限设定，模拟型，0.0～1.0
ML	0.0	输入，输出底限设置，模拟型，0.0～1.0
PS	0	输入，正/反作用设置值，0表示反作用，1表示正作用，数字型
Q	0.0	输出，PID控制输出，模拟型，0.0～1.0
MS	0.0	输出，阀位输出跟踪值，模拟型

（注：考虑到与以前版本兼容保留此算法块）

2.9.7. PID2(无扰 PID)

简介：

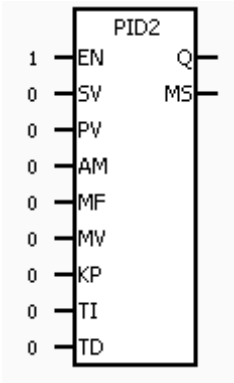
该模块的计算公式为：

$$IN = SV - PV$$

$$Q = KP \times \left(IN_0 + \frac{1}{TI}\int INdt + TD \times \left(IN_0 - IN_1 \right)/T \right);$$

SV 为设定值；PV 为测量值；PS 为正/反作用设置值，0 表示反作用，1 表示正作用；AM 为手动/自动设置值，自动状态 AM=0，手动状态 AM=1；MF 为阀位输出反馈值；MV 为阀位手动设定值；KP、TI、TD 分别为比例系数、积分时间常数、微分时间常数；Q 为阀位输出值，取值范围为 0.0~1.0；MS 为阀位输出跟踪值。

符号：



描述：

参数	初始值	含义
SV	0.0	输入，设定值，模拟型
PV	0.0	输入，测量值，模拟型
AM	0	输入，手动/自动设置值，0表示自动，1表示手动，数字型
MF	0.0	输入，阀位输出反馈值，模拟型
MV	0.0	输入，手动设定值，模拟型
KP	0.0	输入，比例系数，模拟型
TI	0.0	输入，积分时间常数（单位：秒），模拟型
TD	0.0	输入，微分时间常数（单位：秒），模拟型
IS	999999.9	输入，积分分离设定值，模拟型
DE	0.0	输入，输出死区设定置，模拟型
MH	1.0	输入，输出高限设定，模拟型，0.0～1.0
ML	0.0	输入，输出底限设置，模拟型，0.0～1.0
PS	0	输入，正/反作用设置值，0表示反作用，1表示正作用，数字型
Q	0.0	输出，PID控制输出，模拟型，0.0～1.0
MS	0.0	输出，阀位输出跟踪值，模拟型

（注：考虑到与以前版本兼容保留此算法块）

2.9.8. PID1(无扰 PID)

简介：

该模块的计算公式为：

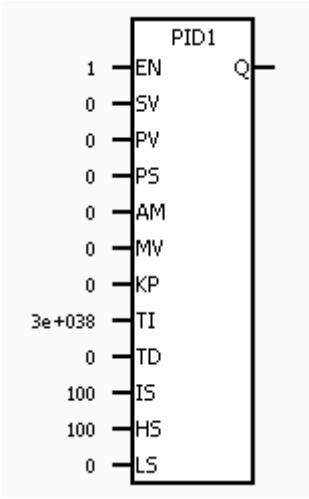
$$Q = KP \times (e + \frac{1}{TI}\int edt + TD\frac{de}{dt})\ ;$$

$$\begin{cases} e = (SV - PV)/(HS - LS) \times 100\%(\text{反作用}) \\ e = (PV - SV)/(HS - LS) \times 100\%(\text{正作用}) \end{cases}$$

SV 为设定值；PV 为测量值；PS 为正/反作用设置值，0 表示反作用，1 表示正作用；AM 为手动/自动设置值，自动状态 AM=0，手动状态 AM=1；MV 为阀位手动设定值；KP、TI、TD 分别为比例系数、积分时间常数、微分时间常数；IS 为积分分离设定值，当 SV 与 PV 的偏差大于等于 IS 时，积分不起作用；HS 为量程上限；LS 为量程下限；Q 为阀位输出值，取值范围为 0.0~100.0。在自动切换成手动时 MV=Q；在手动切换成自动时 SV=PV。

注意：在调节时请设置好测量值 PV 的实际量程上下限 HS、LS；MH、ML 为输出值 Q 的上下限；死区 DE 的默认设置为 0。

符号：



描述：

参数	初始值	含义
SV	0.0	输入，设定值，模拟型
PV	0.0	输入，测量值，模拟型
PS	0	输入，正/反作用设置值，0表示反作用，1表示正作用，数字型
AM	0	输入，手动/自动设置值，0表示自动，1表示手动，数字型
MV	0.0	输入，手动设定值，模拟型
KP	0.0	输入，比例系数，模拟型
TI	3e+38	输入，积分时间常数（单位：秒），模拟型
TD	0.0	输入，微分时间常数（单位：秒），模拟型
IS	100.0	输入，积分分离设定值，模拟型
HS	100.0	输入，量程上限，模拟型
LS	0.0	输入，量程下限，模拟型
DE	0.0	输入，输出死区设定值，模拟型
MH	100.0	输入，输出高限设定，模拟型，0.0～100.0
ML	0.0	输入，输出底限设置，模拟型，0.0～100.0
Q	0.0	输出，PID控制输出，模拟型，0.0～100.0

2.10. 流量处理

2.10.1. IFC97（水蒸气物理计算）

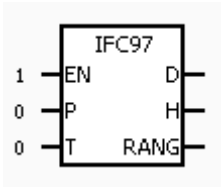
简介：

本算法块按照 IFC97 公式，通过压力，温度，求得密度与比焓。其中 RANG 表示为哪个区间，1 区间表示过冷水区，2 区间表示过热蒸汽区，3 区间表示临界水和临界蒸汽区，4

区间表示饱和区，5 表示过热蒸汽区。

当 RANG=0 表示越界，RANG=3 或 5 时精度可能丢失。建议温度 130℃到 570℃，压力为 0.1MPa 到 22.25MPa。

符号：



参数描述：

参数	初始值	含义
P	0.0	输入，压力（单位：Mpa），模拟型
T	0.0	输入，温度（单位：℃），模拟型
D	0.0	输出，密度，模拟型
H	0.0	输出，比焓，模拟型
RANG	0	输出，整数型

2.10.2. FLOW_SUM（流量累积）

简介：

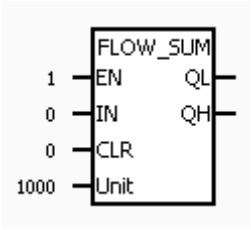
本算法块适用于容积式性质的流量计，如涡节流量计。

流量累积用于记录流量的累积值。IN为流量值，以小时记；CLR为复位端子；Unit为流量累积的高位进位值。

实际流量累积量=QH*Unit+QL。

当 QL 超出 Unit 时，QH 加 1，且 QL=QL-Unit。当 CLR=1 时，输出清零。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
CLR	0	输入，清零，整数型
Unit	0.0	输入，累积单位，模拟型
QL	0.0	输出，模拟型
QH	0	输出，整数型

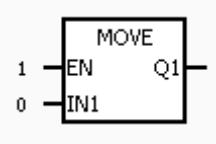
2.11. 赋值

2.11.1. MOVE（赋值）

简介：

该模块将输入变量的值赋给输出变量。最多可 12 对赋值。

符号：



参数描述：

参数	初始值	含义
IN1	0	输入，模拟型或数字型或整数型
Q	0.0	输出，模拟型

举例：

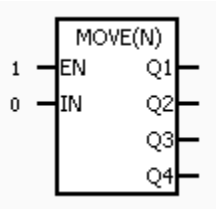
IN=2.5； Q=2.5；

2.11.2. MOVE_N（多变量赋值）

简介：

该模块将输入变量的值赋给多个输出变量。最多赋给 12 个值。

符号：



参数描述：

参数	初始值	含义
IN	0.0	输入，模拟型
Q1～Q4	0.0	输出，模拟型

举例：

IN=2.5； Q1=2.5；Q2=2.5；……；Q6=2.5；

附录 A　系统保留字

系统变量或程序命名除必须按照标识符的命名规则：

以英文字母或中文开头；

续以英文字母、数字、下划线或中文；

还不得与以下保留字冲突。

下表以字母顺序列出算法编辑器编程语言中的所有保留的关键字，它们不得作为用户定义的变量或程序的名称。

A	ABS	ACOS
	ACTION	ADD
	ADDSUM	ADD4
	ALARM	AND
	AND_BOOL	AND_DWORD
	AND4_BOOL	ANDN
	ANY	ANY_BIT
	ANY_DATE	ANY_INT
	ANY_NUM	ANY_REAL
	APERT	ARRAY
	ARROW	ASIN
	AT	ATAN
	AVR	
B	BOOL	BY
	BYTE	BATCH
C	CAL	CALC
	CALCN	CASE
	CD	CDT
	CLK	CONCAT
	CONFIGRUATION	CONSTANT
	COS	CTD
	CTU	CTUD
	CU	CV
	CMPDATE	CMPTIME
	CDW	
D	D	DATE
	DATE_AND_TIME	DELETE
	DINT	DIV

	DO	DS
	DT	DWORD
	DIFF	DF3
	DDF	DED
	DGAP3	DTIME
	DEAL	DZONE
E	ELSE	ELSEIF
	END_ACTION	END_CASE
	END_CONFIGURATION	END_FOR
	END_FUNCTION	END_FUNCTION_BLOCK
	END_IF	END_PROGRAM
	END_REPEAT	END_RESOURCE
	END_STEP	END_STRUCT
	END_TRANSITION	END_TYPE
	END_VAR	END_WHILE
	EN	ENO
	EQ	ET
	EXIT	EXP
	EXPT	EQ0
	EBIT_BYTE	
F	FALSE	F_EDGE
	F_TRIG	FIND
	FOR	FROM
	FUNCTION	FUNCTION_BLOCK
	FLOOR	FPROC
	FCTGEN	FAIL
	FLAG	FTRIG
	FLOWC	
G	GE	GT
	GP01	G01
	G_DWORD	G_WORD
	GTP0	G_TRI
	G_PULSE	GETTIME
I	IF	IN
	INITIAL_STEP	INSERT
	INT	INTERVAL
	INV	INTG

	INTPOLY	INTG2
	INS0	INS1
J	JMP	JMPC
	JMPCN	
L	L	LD
	LDN	LE
	LEFT	LEN
	LIMIT	LINT
	LN	LOG
	LREAL	LT
	LWORD	LIM
H	HLALM	
M	MAX	MID
	MIN	MOD
	MOVE	MUL
	MUX	MOVE_N
	MAV	MOVE_4
N	N	NE
	NEG	NOT
	NOT_BOOL	NOT_DWORD
	NE0	NDGT
	NSEL	NLIM
O	OF	ON
	OR	ORN
	OR_BOOL	OR4_BOOL
	OR_DWORD	
P	P	PRIORITY
	PROGRAM	PT
	PV	POLY
	POW	PID
	PILA	PID2
Q	Q	Q1
	QU	QD
	QREG	
R	R	R1
	R_TRIG	READ_ONLY
	READ_WRITE	REAL

	RELEASE	REPEAT
	REPLACE	RESOURCE
	RET	RETAIN
	RETC	RETCN
	RETURN	RIGHT
	ROL	ROR
	RS	RTC
	R_EDGE	RECIP
	RTRIG	RAND
	RSET	RUNN
	ROR_WORD	ROL_WORD
	ROR_DWORD	ROL_DWORD
	RBIT_DWORD	
S	S	S1
	SD	SEL
	SEMA	SHL
	SHR	SIN
	SINGLE	SINT
	SL	SQRT
	SR	ST
	STEP	STN
	STRING	STRUCT
	SUB	SEL2F3
	SQR	SEL
	SIGN	SMTH
	SHL_DWORD	SHR_DWORD
	SBIT_DWORD	SELPF
	SHR_WORD	SSWITCH
	SHL_WORD	SORT
	STAT	SCALE
T	TAN	TASK
	THEN	TIME
	TIME_OF_DAY	TO
	TOD	TOF
	TON	TP
	TRANSITION	TRUE
	TYPE	TPROC

	TOFF	TEST_DWORD
U	UDINT	UINT
	ULINT	UNTIL
	USINT	
V	VAR	VAR_ACCESS
	VAR_EXTERNAL	VAR_GLOBAL
	VAR_INPUT	VAR_IN_OUT
	VAR_OUTPUT	VLIM
W	WHILE	WITH
	WORD	WEIGHT
X	XOR	XORN
	XOR_DWORD	

附录 B　快捷键表

以下列出算法编辑器中常用的快捷键：

Ctrl + 1 ： 显示导航栏中的“程序”子窗口

Ctrl + 2 ： 显示导航栏中和“算法块”子窗口

Ctrl + 4 ： 显示程序编辑视图

Ctrl + 5 ： 显示程序的IEC文本视图

Ctrl + 6 ： 显示程序的帮助视图（ST和IL）

Ctrl + 7 ： 显示/隐藏观察窗口

Ctrl + 8 ： 显示/隐藏输出窗口

Ctrl + 9 ： 显示/隐藏状态栏

Ctrl + 0 ： 显示/隐藏导航栏

Ctrl + F ： 全屏幕

Ctrl + N ： 新建程序

Ctrl + S ： 保存工程

Ctrl + I ： FBD和LD图中弹出算法块选择对话框，用以插入算法块

Ctrl + O ： 弹出系统设置属性表

Ctrl + Z ： 撤销操作

Ctrl + Y ： 恢复操作

Ctrl + X ： 剪切

Ctrl + C ： 拷贝

Ctrl + V ： 粘贴

Ctrl + Left ： FBD图中移动前一个算法块

Ctrl + Right ： FBD图中移动到后一个算法块

Ctrl + PgDn ： 在打开的程序画面之间切换

Ctrl + F6 ： 在打开的程序画面之间切换

Ctrl + 上箭头 ： 算法块执行顺序上移 （当算法块的执行顺序为手动排序且选中算法块列表时，该快捷键有效）

Ctrl + 下箭头 ： 算法块执行顺序下移（同上）

Ctrl + A ： 全选

F1 ： 弹出帮助

F2 ： 开始/停止连续仿真

F3 ： 暂停连续仿真

F4 ： 重新开始连续仿真

F5 ： 开始/停止单周期仿真

F6 ： 单周期仿真运行至下一步

F9 ： 编译

F10： 查找变量引用

F11： 查找算法块的使用

F12： 查找子程序的调用

附录 C　错误代码表

以下为编译时在编译信息栏中出现的错误代码说明：

1－变量定义有误；
2－非法的语句起始符；
3－表达式含有功能块；
4－功能块不存在此引脚；
5－不存在合法的语句结束符；
6－不合法语句；
7－功能块调用参数超过设定；
8－不存在此数据类型；
9－功能块类型被重复定义；
10－算法块调用时，不允许输入类型的引脚赋值给其他变量；
11－SFC中步名被重复定义；
12－SFC中该步未被定义；
13－SFC中转换条件被重复定义；
14－SFC中转换条件未被定义；
15－SFC中动作名被重复定义；
16－SFC中动作名未被定义；
17－程序名被重复定义；
18－程序名未被定义；
19－任务名被重复定义；
20－任务名未被定义；
22－实时数据库变量名未被定义；
23－变量名被重复定义；
24－变量名未被定义；
25－时间类型变量声明不合法；
26－只有功能块实例才能被调用；
27－非法的标识符；
28－IL语言中标号未被定义；
29－IL语言中标号被重复定义；
30－控制算法数据区超过系统设定值8192点。

附录 D　保持算法块

以下是所有 FBD 算法块中保持算法块的说明：

组名	算法块名称	中文名	备注
代数	STAT	统计值	保存累计值以及输入个数
代数	INTG	积分	保存积分值
代数	ADDSUM	累加	保存累加结果
信号发生器	G_DWORD	双字循环移位	保存相位
信号发生器	G_WORD	字循环移位	保存相位
触发器	RS	RS 触发器	保存历史输出
触发器	SR	SR 触发器	保存历史输出
计数器	CTUD	递增递减计数器	保存历史输出
计时器	TP	开记忆计时器	保存计时输出
计时器	TON	ON 延时	保存当前计时值
计时器	TOFF	OFF 延时	保存当前计时值
定时器	TIMER_MS	50 毫秒定时器	保存时间计数
定时器	TIMER_S	秒定时器	保存时间计数
定时器	TIMER_M	分钟定时器	保存时间计数
流量处理	FLOW_SUM	流量累计	保存流量累计值

附录 E　数据类型转换

以下是 IEC 算法程序支持的数据类型及类型转换关系：

英文名	中文名	取值范围	BOOL	float	DWORD	WORD	BYTE	double	作用域	
									IEC	RTM.APU
BOOL	布尔型(开关量)	0, 1 (UINT32.D0)	—	0.0, 1.0	0, 1	0, 1	0, 1	0.0, 1.0	√	√
float	浮点型(模拟量)	±1.18E-38~±3.4E+38 (7 位有效数字)	fabs(x)>0.000001 为 1，否则为 0	—	UINT32(x)	UINT16(x)	UINT8(x)	double(x)	√	√
INT	整数型	-2147483648~ 2147483647	x=0 为 0，　x != 0 为 1	float(x)	UINT32(x)	UINT16(x)	UINT8(x)	double(x)	√	√
DWORD	无符号整数型	0~4294967295	x=0 为 0，　x != 0 为 1	float(x)	—	UINT16(x)	UINT8(x)	double(x)	√	√
WORD	无符号短整数型	0~65535	x=0 为 0，　x != 0 为 1	float(x)	=x	—	UINT8(x)	double(x)	√	√
BYTE	单字节整数型	0~255	x=0 为 0，　x != 0 为 1	float(x)	=x	=x	—	double(x)	√	√
double	双精度浮点型	为了保证计算精度，算法中的浮点运算均采用 double 数据类型	fabs(x)>0.000001 为 1，否则为 0	float(x)	UINT32(x)	UINT16(x)	UINT8(x)	—	√	√
DATE& TIME	日期+时间	1970-1-1 12:00:00 至 3000-12-31(DWORD)	—	—	—	—	—	—	√	√

[1] x表示为数据类型转换的源数据；

[2]算法块输入端数据类型与所连接的数据类型不一致，则以算法块输入端数据类型为准，先进行数据类型转换（参考上表的转换规则），再参与运算；

[3]算法块输入端连接常数时，则以算法块输入端数据类型为准；

[4]作用域：IEC 指上位机 IEC 算法相关模块(包括编辑、编译、仿真运行等)；RTM.APU 指控制模件的算法运行模块；

质量管理

质量方针：提供优质稳定的自动化产品，满足行业用户的自动化需求

始终以满足用户广泛需求为新产品开发宗旨。收集用户关于产品功能、技术和性能价格方面的要求，并热诚欢迎您的直接参与。对所有新产品开发的思路，公司召集相关专业人员共同研究探讨。

设计与测试同步化。为确保产品质量，在产品开发初期，测试部会考察整个项目设计方案。在整个产品开发过程中，进行一系列的严格、专业的测试，直到所有的测试全部通过后，才能进行试生产。

质量承诺：追求高品质生产，保证用户满意

高品质生产。新开发的产品只有在经过设计过程中的所有质量检查和生产前的全面测试后，才能投入生产，并在制造过程中，进行更加严格的质量测试。

富有弹性的生产能力可保证按时供货。公司采用一体化的采购与生产体系，合理配置资源，提高生产的灵活性与有效性，无论用户的需求量大小，都可以保证按时供货。

总体质量控制。在生产过程中使用总体质量控制程序，从装配到系统集成，每个产品都要单独接受检测，并进一步使用静态和动态预烧测试。

技术支持：丰厚的专业知识，提供行业自动化解决方案

无偿的技术支持。无偿获取自动化产业界的最新技术动态和最新产品介绍，无偿的产品软件升级服务，无偿提供解决方案设计与电话技术支持。

卓越的OEM/ODM能力。集长期工控系统设计和生产经验，我们有能力满足您特殊的应用要求，协助实现您的个性化设计理念。并给出最适合您应用需求的解决方案。

对用户询问的快速反馈。对与您的技术问题，保证在24小时内给予回答。

迅捷的供货能力。所有产品都拥有适量库存，包括系统升级需要的各种组件和附件。

<table>
<tr>
<td>     </td>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>EMC等可靠性测试实验仪器</td>
<td>重大工程仿真试验平台</td>
<td>生产测试车间</td>
</tr>
</table>

UWNTEK 杭州优稳自动化系统有限公司

技术中心：浙江大学控制工程国家实验室大楼（310027）

技术支持：400-007-0089

传真：0571-88371967

http://www.uwntek.com

email:bd@uwntek.com;