中控·SUPCON

AdvanTrol-Pro V2.70 软件

图形编程使用手册

浙江中控技术股份有限公司

目　录

图形编程

图形编程软件是 SUPCON 集散控制系统软件的重要组成部分之一，基于 Windows 操作系统设计，充分利用 Windows 操作系统的优点，具有良好的用户界面。

图形编程概述

图形编程软件用图形方式描述控制过程，使控制过程组态变的更简单，也使控制工程师可以专注于控制方案。

图形编程性能特点

图形编程集成了 LD 编辑器、FBD 编辑器、SFC 编辑器、ST 编辑器、数据类型编辑器、变量编辑器、DFB 编辑器。采用工程化的文档管理方法，提供了一个功能强大的实现程序重用和结构化的工具 。

编程指南

图形编程软件的编程包括 LD 语言编程、FBD 语言编程、SFC 语言编程和 ST 语言编程。FBD 编辑器、LD 编辑器、SFC 编辑器和 ST 编辑器作为其最重要的编辑器，与变量编辑器、数据类型编辑器、DFB 编辑器等共同构成了一个强大的编辑环境。

图形编程使用指南

使用指南介绍了软件的运行环境和工作界面，并介绍了用户如何利用菜单功能、图形化的功能模块及其他一些工具，顺利地进行 LD、FBD、SFC 及 ST 语言的编程。

图形编程模块库

模块即为图形编程软件的功能模块，图形编程软件提供了近 200 个基本模块供用户选用，分为 IEC 模块以及非 IEC 标准模块两大类。另外，用户还可以使用自行设计的自定义模块。

1 图形编程概述

图形编程软件，作为集成的图形编程工具，是针对集散控制系统所开发的全中文界面的 DCS 组态与控制工具，是 SUPCON 系列 DCS 的控制方案组态工具，依据 IEC61131-3 标准，为用户提供高效的组态环境，与系统组态软件联合完成对系统的组态，是 SUPCON 集散控制系统软件的重要组成部分之一。图形编程软件基于 Windows 操作系统设计，充分利用 Windows 系统的优点，具有良好的用户界面。

图形编程软件的组态通过图形用户接口进行，只要求用户有基本的 Windows 操作基础。

图形编程提供灵活的在线调试功能，用户可以观测程序的详细运行情况。

图形编程提供了详细的在线帮助，上下文关联的联机帮助使用简单的按鼠标或 F1 键为组态中的每种情况提供支持。

2 图形编程性能特点

图形编程集成了 LD 编辑器、FBD 编辑器、SFC 编辑器、ST 编辑器、数据类型编辑器、变量编辑器及 DFB 编辑器。

图形编程的所有编辑器使用通用的标准 File、Windows、Help 等菜单。灵活地自动切换不同编辑器的特殊菜单和工具条。

图形编程在图形方式下组态十分容易。在各编辑器中，目标（功能块、线圈、触点、步、转换等）之间的连接在连接过程中进行语法检查。不同数据类型间的链路在编辑时就被禁止。图形编程提供注释、目标对齐等功能改进图形程序的外观。

图形编程采用工程化的文档管理方法。通过导入导出功能，用户可以在工程间重用代码和数据。

图形编程 DFB 提供了一个功能强大的实现程序重用和结构化的工具。

图形编程的特点可简单归纳如下：

1. 使用 Windows 的友好图形界面，使用鼠标也可以使用键盘进行编辑操作，工具条上所有功能都有文字提示；
2. 编辑环境通过工程文件管理多个图形文件，用户容易操作；
3. 组态元素放置灵活，自动格线对齐，触点、线圈、功能块和变量等可用文本进行注释；
4. 图形绘制采用矢量方式，具备块剪切、拷贝、粘贴、删除等功能，达到事半功倍的效果；
5. 具备对前次操作步骤的撤消和恢复功能，提高了组态效率；
6. 智能连线处理，模块引脚接近时自动连接；
7. 连线时动态检查数据类型，数据类型不一致拒绝连接；
8. 强大的查找和替换功能，可在当前程序段也可在当前整个工程中查找变量、常数、位号及模块，并进行标记，用户只需用鼠标点击相应的信息就可以直接跳到所要查找的位置。替换功能亦然，可在当前程序段或当前整个工程中逐个替换或全部替换所选择的变量、常数、位号 及模块；
9. 提供缩放功能，使用户更清晰地查看页面或按照缩小的比例看到页面中更多的内容；
10. 系统为用户管理定义的位号和变量，用户不用关心具体物理内存；
11. 在每个编辑器中可以使用系统已定义的基本功能模块（EFB）和用户自己定义的功能模块（DFB）。每个编辑环境中内嵌自定义模块（DFB）编辑器。极大地提高了程序的重用性，减少编程工作量；
12. 用户可以用 EFB 和 DFB 再组成新的 DFB。具有无限的功能扩展性。方便用户做二次开发；
13. 用户可以使用工程的导入导出功能重用功能模块；
14. 用户可通过数据类型编辑器生成自定义的数据类型；
15. 功能块编辑器（FBD）、梯形图编辑器（LD）及顺控图编辑器(SFC)集成在一起，可相互嵌套调用，具有无限的功能扩展性；
16. 提供在线调试功能；

17. 强大的在线帮助功能。

3 编程指南

图形编程软件的编程包括 LD 语言编程、FBD 语言编程、SFC 语言编程和 ST 语言编程。编程流程包括工程的创建、段落的创建、区段的创建、程序段的编辑、工程的编译及链接等几个过程。

在图形编程软件中，FBD 编辑器、LD 编辑器作为最重要的编辑器，与变量编辑器、数据类型编辑器、DFB 编辑器等共同构成了一个强大的编辑环境。

用户可进行在线调试，可以将外部现成的有用 LD、FBD 程序导入进工程中，也可以将本工程中比较实用、或能用于其他工程的各种文件通过导出操作，提供给其他工程利用，充分代码重用。

3.1 综述

图形编程的编程语言包括功能块图（FBD）、梯形图（LD）、顺控图（SFC）及 ST 语言。支持国际标准 IEC61131-3 数据类型子集。用户可以使用数据类型编辑器生成自己的数据类型。

图形编程的每一个工程对应一个控制站。工程可包含多个段落。每个段落只能选用一种编辑器。按 IEC61131-3 标准，FBD 编程语言的基本元素是功能块；LD 编程语言的基本元素除了功能块外还包括触点和线圈；SFC 编程语言的基本元素是转换、步和跳转；ST 编程语言除了可使用基本的 ST 语法外，还可调用系统函数。在工程中可以分别指定不同段落的执行周期和执行次序。

图形编程提供以下编辑器：

- FBD 编辑器
- LD 编辑器
- SFC 编辑器
- ST 编辑器

在生成段落时，用户可以指定生成的段落的类型。段落的类型指定了使用何种编辑器。

除与编程语言有关的编辑器外，还有：

- 数据类型编辑器
- 变量编辑器

3.2 工程管理

以下介绍：

- 工程
- 段落
- 区段
- 变量
- 注释文本
- 调试文本

3.2.1 工程

图形编程用一个工程（Project）描述一个控制站的所有程序。工程包含一个或多个段落（Section）。每个工程唯一对应一个控制站，工程必须指定其对应的控制站地址。图形编程通过工程管理多个段落文件，在工程文件中保存配置信息。

3.2.2 段落

段落是通常意义上的一个文档，是组成工程的基本单位。

新建段落时必须指定段落的编辑类型和程序类型。

按编辑类型可将段落分类为：

- FBD 段落
- LD 段落
- SFC 段落
- ST 段落

按程序类型分可将段落分类为：

- 程序段落
- 模块段落

选择编辑类型相当于选择何种编辑器进行编程。选择程序类型相当于选择是生成一个可执行的程序或是进入 DFB 编辑器生成 DFB 模块。

3.2.3 段落管理

选择**工程**菜单中的**段落管理**进入段落管理对话框。

图 3-1 段落管理对话框

可以通过**新建**按钮新建一个段落，效果与**文件**菜单中的**新建程序段**命令一样。

图 3-2 新建程序段

选择一个段落，然后可以通过**打开**按钮打开段落。

选择**文件**菜单中的**打开程序段**命令。弹出打开段落对话框，选择需打开的段落按打开按钮也可以打开段落。

图 3-3 打开段落

直接在工程栏中双击相应的段落名，也可以打开段落。

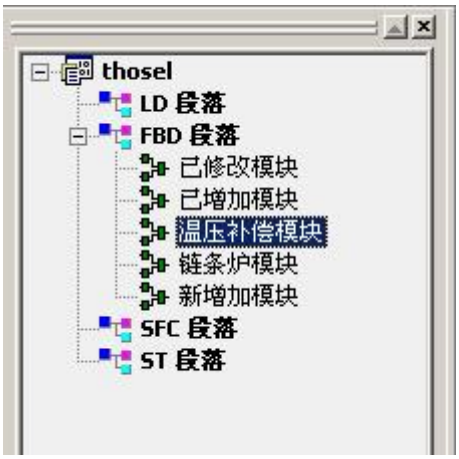

图 3-4 工程栏中打开段落

在段落管理对话框中选择想删除的段落，按删除按钮可以删除段落。

在工程栏中，选择想删除的段落，按鼠标右健，弹出浮动菜单，选择删除段落，也可以删除段落。

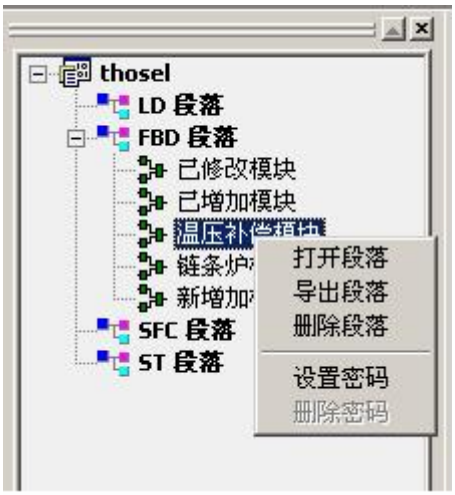

图 3-5 后键菜单

在段落管理对话框中选择想导出的段落，按**导出**按钮可以导出段落到文件。

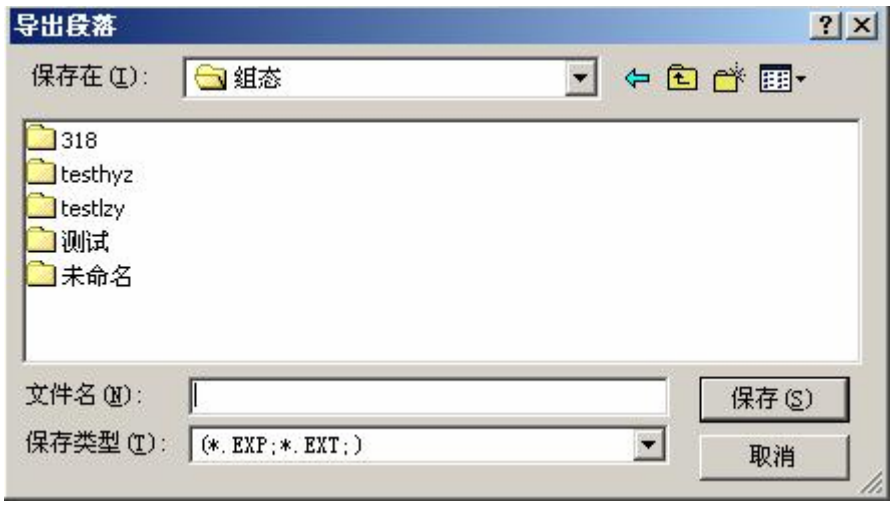

图 3-6 导出段落

导出文件名必须为*.exp。

在工程栏中，选择想导出的段落，按鼠标右健，弹出浮动菜单，选择导出段落，也可以导出段落。

在段落管理对话框中按**导入**按钮可以从文件中导入段落到工程。

当在 **段落管理** 对话框中选择一个或多个段落导出时，用户要指定导出段落存放的文件名。图形编程先检查所有的段落，如段落中包含未被选择的 DFB，则图形编程自动追加这些 DFB 段落。然后检查所有段落中包含的变量的数据类型，若发现其中的数据类型是由自定义数据类型派生而来，图形编程将自动追加这些自定义数据类型。导入时，选择已生成的导出文件，工程中将添加所包含的数据类型、段落。当导入时，发现段落名冲突，将提示用户是替换或保留或用新名导入。

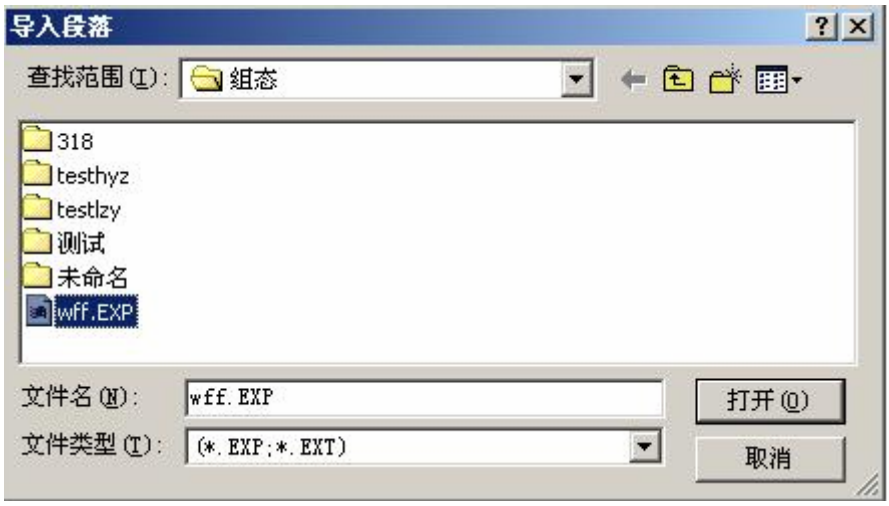

图 3-7 导入段落

在段落管理对话框中按**修改**按钮可以修改段落名。

图 3-8 修改段落名称

3.2.4 任务管理

当有多个程序时，程序段落的执行周期和执行次序会影响程序的运行结果。在工程菜单中选择任务管理弹出任务管理对话框用于设置执行周期和执行次序。

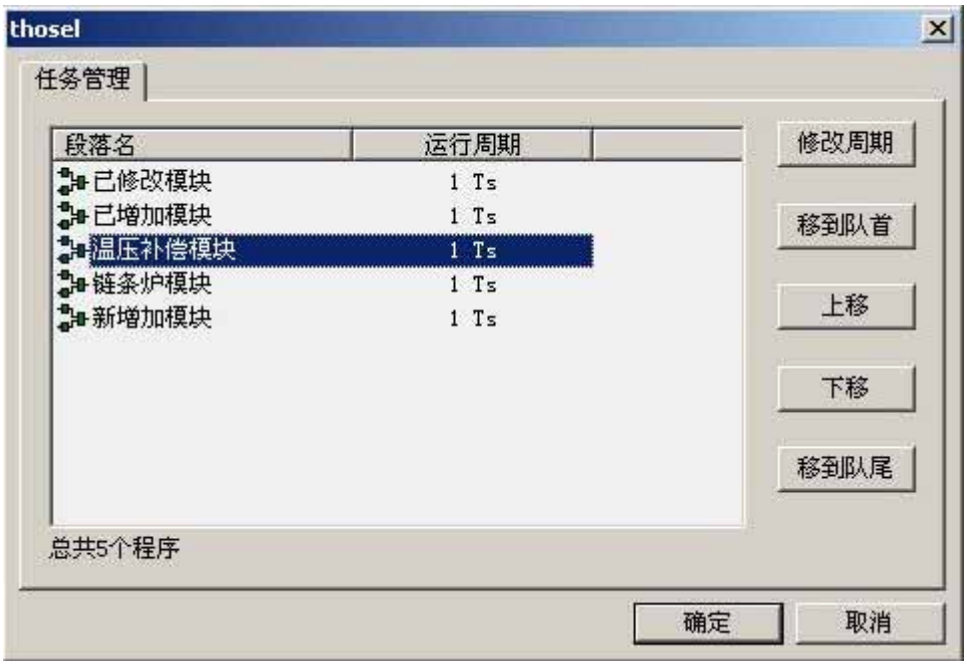

图 3-9 任务管理对话框

图形编程以系统组态软件中设置的控制周期为 1Ts，即：如果在系统组态软件的组态过程中设置了控制周期为 0.1s，则 1Ts = 0.1s.

用户还可通过操作“ 移到队首 ”、“ 上移 ”、“ 下移 ”、“ 移到队尾 ”等操作设置同一运行周期各程序运行的优先级，即排在队列靠前的同一运行周期程序比排在队列靠后的程序优先执行。不同运行周期的程序之间的优先级无法比较。

3.2.5 区段

区段指在同一段落中有数据信号相连的元素的总和。一个段落可以包含一个或多个区段（SFC 段落只有一个区段）。

在区段内 EFB 或 DFB 的执行次序是由区段中间的数据流来决定的。在 FBD 区段内那些输入只连接变量或位号或常数的、在 LD 区段输入只连接变量或位号或常数或左汇流条的、SFC 区段中的起始步，被称为区段的**起始模块**。区段内有多个起始模块时，在图形区域中位置最上的模块称为**启动模块**。区段的执行就从启动模块开始，按数据流要求逐步进行。

同一段落内区段间的执行次序就依据区段的启动模块在图形区域中位置来决定。启动模块在上的先执行。

3.2.6 数据类型

类 型	关键字	字节数	表 示 范 围
布尔型	BOOL	1	0 或 1
字节	BYTE	1	0 ~ 255
字	WORD	2	0 ~ 65535
双字	DWORD	4	0 ~ 4294967295
整型	INT	2	-32768 ~ +32767
无符号整型	UINT	2	0 ~ 65535
长整型	LONG	4	-2147483648 ~ 2147483647
无符号长整型	ULONG	4	0 ~ 4294967295
半浮点型	SFLOAT	2	-7.9998 ~ +7.9998
浮点型	FLOAT	4	±1.175490351E-38 - 3.402823466E+38
累积型	structAccum	8	

3.2.7 累积类型

在程序中可以直接定义累积类型变量。累积类型是系统提供的一种结构类型，即 structAccum。该结构的定义如下：

```
struct structAccum
{
   sfloat remainder;     //小数部分
   long accum;           //整数部分
   int reserved;         //保留部分，禁止使用
}
```

由定义可以看出，累积类型变量含有三个成员：remainder、accum、reserved。其中保留

部分禁止用户使用，remainder 表示累积的小数部分（小于 1），accum 表示累积的整数部分。当小数部分超过表示范围[0－1）时，自动向整数部分进位。

在 AI 结构中还存在着一种模拟量累积量，模拟量累积量在工程中大量运用，它由两部分构成：

sum1	sum0

其中，sum1 占 32 位，是长整形；sum0 占 16 位，是 sfloat 型，是无符号 12 位定点小数，整数部分占 4 位。当模拟量累积超过 sum0 所能表示的范围（0－15.999），自动向高位的 sum1 进位。

例如，accum1 为模拟量累积类型，accum2 为累积类型。

当 accum1=12.123443 时，accum1.sum0=12.123443，accum1.sum1=0。

accum2=12.123443 时，accum2.remainder=0.123443，accum2.accum=12。

当 accum1=34.457638 时，accum1.sum0=2.457638，accum.sum11=2，因为 sum1 从第 17 位开始，所以 sum1=2,实际指的是 2×16=32。

accum2=34.457638 时，accum2.remainder=0.457638，accum2.accum=34。

3.2.8 数据类型存储方式

在计算机中，所有数据都由二进制表示：

BOOL：占 1 字节，零表示 FALSE，非零表示 TRUE；

BYTE：1 字节，占 8 位。

WORD：两字节，占 16 位，无符号；

DWORD：四字节，占 32 位，无符号；

INT：两字节，占 16 位，最高位是符号位：0 表示正数，1 表示负数；

UINT：两字节，占 16 位，无符号；

LONG：四字节，占 32 位，最高位是符号位：0 表示正数，1 表示负数；

ULONG：四字节，占 32 位，无符号；

SFLOAT：两字节，占 16 位，用定点法表示。在定点表示法中，二进制小数点位置通常是固定不变的。小数点可以固定在数值位之前，也可以固定在数值位后面。前者称为定点小数表示法，后者叫做定点整数表示法。SFLOAT 定点数 N 的一般表示形式为：

符号位	整数位	尾数

其中，符号位占一位：0 为正数，1 为负数；整数位占三位；尾数占十二位。

FLOAT：四字节，占 32 位，用符点法表示。在采用符点表示的二进制数中，小数点位置是浮动的,不固定的。通常任何一个二进制都可以写成：

$$N=2^P\times S$$

式中，S 为二进制数 N 的尾数，代表了 N 的实际有效值；P 为 N 的阶码，可以决定小数点的具体位置。因此，任何一个符点数 N 都由阶码和尾数两部分组成。阶码部分包括阶符和阶码，尾数部分有数符和尾数组成。其形式为：

阶符	阶码	数符	尾数

其中，阶符占一位，阶符=0 表示阶码为正，阶符=1 表示阶码为负；阶码为七位；数符占一位，数符=0 表示该数为正数，数符=1 表示该数为负数；尾数为二十三位。

3.2.9 变量

变量包括用于在段落中间、段落之间的指定名称的数据以及操作站和控制站进行数据交换的位号。

变量按组织形式分为：

- 基本变量
- 复合变量

基本变量的数据类型是基本数据类型。复合变量的数据类型为复合数据类型。复合数据类型通过数据类型编辑器生成，通过基本数据类型和已生成的复合数据类型组合而成。

变量按作用关系分为：

- 全局变量

全局变量指在段落之间共享的变量。在工程中声明全局变量后，在所有段落都可以访问。全局变量一经声明，就被分配一个固定的控制站地址，放在系统数据区中，能够将当前数据保持到下一个控制周期。

- 私有变量

在程序中可以声明私有变量。私有变量一经声明，就被分配一个固定的控制站地址，放在系统数据区中，能够将当前数据保持到下一个控制周期。私有变量与全局变量的不同在于私有变量只有声明的段落能够存取，其他段落对此变量不可见。变量的作用范围就被限制在当前段落中。变量封装有利于用户编程。

- 输入变量与输出变量

在 DFB 段落（创建时程序属性被指定为模块的段落）中，可以声明输入变量和输出变量。通过指定输入变量和输出变量，就设定了 DFB 的外部接口。对输入变量和输出变量，系统不分配控制站内存。

在 DFB 段落中也可以声明私有变量。但此段落中的私有变量只有在 DFB 被使用时才被分配控制站地址。DFB 每被使用一次，私有变量就被创建一次。DFB 相当于一个类，此私有变量相当于成员。类可以创建多个实例，而每个实例的成员地址都不同。DFB 段落内的私有变量与 DFB 外部接口无关，在 DFB 外部不可见，主要用来存放需要将当前状态保持到下一个控制周期的数据。

用包含私有变量的 FFB 构建的 DFB 时，新的 DFB 将继承包含的 FFB 的私有变量。在 DFB 段落中，私有变量在保证 DFB 重用性的同时还满足了与时序相关的控制的要求。

系统中还包含了一种对用户不可见的热备份变量。热备份变量用于当 DCS 系统配置为双主控卡热冗余时，在双主控卡间定时的同步数据。如积分模块，就包含了一个热备份变量，用来同步积分的当前值。这样，在 冗余切换时，保证了无扰动切换。通过这种热冗余方式，充分保证了系统的安全性。

跟时间相关具有累积作用的 EFB 模块都包含了热备份变量。

用包含热备份变量的 FFB 构建新的 DFB 时，新的 DFB 将继承包含的 DFB 的热备份变量。

3.2.10 注释文本

注释文本在程序中增加标注信息，用于增加程序的可读性。文本目标的大小取决于文本的长度。根据文本字体大小，目标的大小可以通过在垂直以及在水平方向上更多的网格来进

行扩展。注释文本不占控制站内存。注释文本的字体和颜色都可以设置。

3.2.11 调试文本

调试文本是在联机状态下显示变量或位号在控制站中的实际值的文本信息。用户通过调试文本可以操纵控制站数据。方便调试程序和监视系统运行。调试文本不占控制站内存。

图 3-10 属性

3.2.12 查找与替换

用于进行用户所需信息（如模块、文字、变量名等）的检索。用户可自行填写，也可按照不同类型、数据源通过浏览进行检索。

按 CTRL+F 或用编辑菜单的查找命令进入查找对话框。

图 3-11 查找对话框

使用查找功能可以查找变量和模块。选择当前段则在当前打开的段落中搜索目标，选择当前工程将搜索整个工程。按查找按钮，鼠标将跳到搜索到的下一个目标并将其选中。按标记按钮则在信息栏中显示所有搜索到的目标。用鼠标双击目标信息，则打开目标所在得段落，并跳到目标位置。

按 CTRL+H 或用编辑菜单的替换命令进入替换对话框。

图 3-12 替换对话框

按查找按钮，鼠标将跳到搜索到的下一个目标并将其选中。然后按替换，则搜索目标将被替换成替换目标。如果按全部替换按钮，则所有符合搜索条件的目标都被替换。

3.3 功能块图（FBD）语言

根据 IEC1131-3，FBD 编辑器将基本的功能/功能块（EFB）和信号（变量、位号）组成功能块图（FBD）。EFB 和变量可以加注释。图形内可以自由放置基本元素和文本。部分 EFB 的输入可以扩展方便使用。

在 FBD 编辑器中，窗口的背景是逻辑坐标网格。当正在创建时，功能块在该网格的光栅中对准。如果功能块发生与别的功能块重叠的情况，将会出现错误信息并且功能块不创建。当实际参数在功能块输入/输出创建时，他们可能与别的目标重叠但不会破坏区段画面的界限。如果有一链路作为与别的功能块的连接，则该连接要进行检查。如果这是一个未经许可的连接，该链路将不生成。

除这些目标之外，还可以将注释文本和调试文本放入 FBD 段落中。该文本目标的大小取决于文本的长度。根据文本大小，目标的大小可以通过在垂直以及在水平方向上更多的网格来进行扩展。

3.3.1 FBD 编辑器

根据 IEC61131-3，FBD 编辑器将基本的功能/功能块（EFB）和信号（变量、位号）组成功能块图（FBD）。EFB 和变量可以加注释 ，图形内可以自由放置基本元素和文本，部分 EFB 的输入可以扩展方便使用。

图形编程提供了部分预定义的 EFB 模块库，包含了近 200 个基本模块，并且库中的模块被组织成不同的组。 模块库包括：

- IEC 模块库

包括在 IEC61131-3 中定义的功能块。包括：算术运算、比较运算、逻辑运算、转换、选择、触发器、计数器、定时器等类型。

- 辅助模块库

包括控制模块、通讯辅助函数、累积函数、输入处理、辅助计算、电量转换等。

- 自定义模块库

用户自定义模块。

- 附加库

包括特殊模块、锅炉模块、造气模块、DEH 模块、智能通讯卡模块等。

在 FBD 编程中，用户可以使用基本功能块和自定义功能块。

用户用 DFB 编辑器生成的自定义模块被放在自定义模块库中。DFB 加入模块库中后，就可以被各种语言编辑器使用。

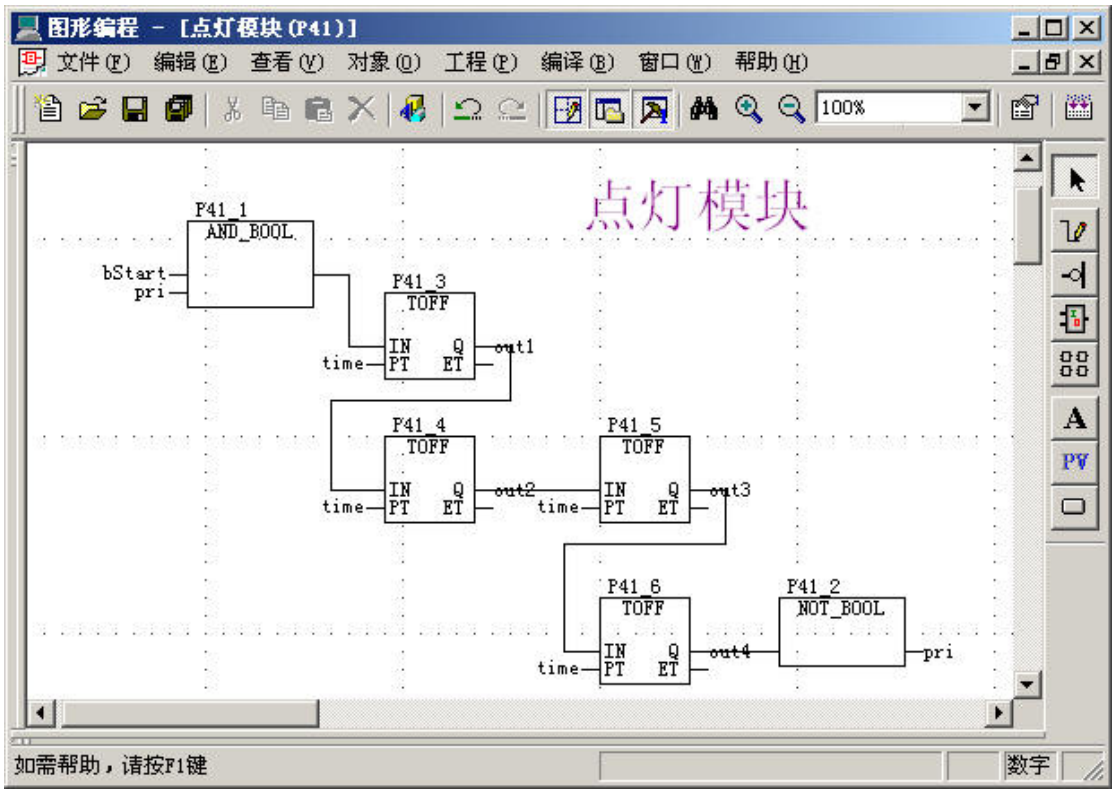

图 3-13 FBD 编辑器

3.3.2 FFB（功能和功能块）

FFB 是基本功能块（EFB）和自定义功能块（DFB）的统称。功能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。功能块的名称在图形框的中间显示。功能块的实例名在图形框上显示。在同一工程内，模块的实例名是唯一的。

所有功能块都可以用一个 EN 输入和一个 ENO 输出进行配置。

如果当调用功能块时 EN 值等于 0，则该功能块将不被执行，ENO 值自动设置成 0；如果 EN 值等于 1，则该功能块将被执行，执行完后，ENO 值自动设置成 1。

在不需要 EN 的时候，可以隐藏 EN 和 ENO 引脚。以下分别是显示有 EN、ENO 口与隐藏了 EN 和 ENO 口的模块示意图：

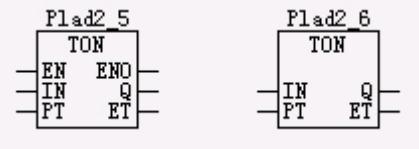

图 3-14 模块示意图

FFB 的输入/输出关系不受 EN 和 ENO 的影响。

EFB 和 DFB 都可以包含私有变量和热备份变量。在编辑 DFB 时，用户可以添加私有变量。在引用时，FFB 的私有变量和热备份变量对用户透明。

3.3.3 实际参数

在程序运行中，取值过程的值和其他数据是通过实际参数向 FFB 传送的。实际参数包含变量、位号、常数。

实际参数的数据类型必须与相连接的引脚类型一致。

3.3.4 功能块库

图形编程提供了几个功能块库，IEC 模块库、辅助模块库、自定义模块库、附加库等。

IEC 模块库包含以下几类模块：

- 算术运算，如 ADD、SUB、MUL、DIV
- 比较运算，如 GE、LT、NE
- 转换运算，如 INT_TO_LONG
- 数学函数，如 SIN、EXP、LOG
- 逻辑运算，如 AND、OR、XOR、
- 选择运算，如 MUX、LIM
- 计数器、定时器、触发器，如 TON

辅助模块库包含以下几类模块：

- 累积函数，如 ACCUM_TO_SUM
- 通讯辅助函数，如 GETBIT
- 控制模块，如 BSC、CSC
- 辅助计算，如 BCS_TO_BIN
- 输入处理，如 ACCELERATE_MV
- 文本代码模块，如 TEXTCODE
- 电量转换，如 ACMETER12
- 信号选择模块，如 AVE_1IN3_SFLOAT
- 浮点处理模块，如 F_DEADBND

附加库包含以下几类模块：

- 特殊模块，如 D_INVERSE_COUNT
- 锅炉模块，如 COAL_AND_WIND
- 造气模块，如 CtoG_SC001_POINT
- DEH 模块，如 SERVOCON
- 智能通讯卡模块，如 GW_GETBOOL

具体模块库中所包含的模块以及各个模块的使用请参见《图形编程模块使用手册》

择不同类型的主控卡（或通讯卡），附加库中可见的模块有所不同。比如，只有选择了 FW248 卡，智能通讯卡模块才可见。

3.3.5 链接

链接是功能块之间的连接。一个功能块输出可以连接多个功能块的输入，要连接的输入/输出必须要有相应的数据类型。链接允许与其他目标重叠，链接不能循环配置。循环必须通过实际参数来解决。

在 LD 段落中当触点靠近左汇流条时，自动生成链路。触点与触点、触点与线圈靠近时，也将自动生成链路。

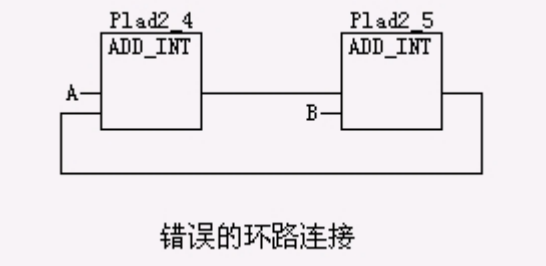

图 3-15 错误的环路连接

解决方法：

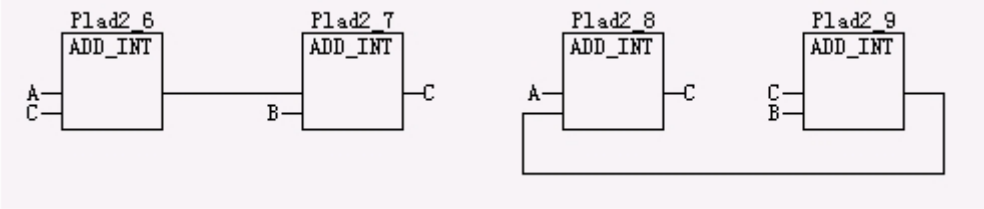

图 3-16 正确的环路连接

3.3.6 执行次序

在 FBD 区段内那些输入只连接变量或位号或常数的模块，被称为区段的**起始模块**。

区段内有多个起始模块时，在图形区域中位置最上的模块称为**启动模块**。

区段的执行从启动模块开始。

FBD 区段内的执行次序由区段内的数据流决定。

FBD 段落中区段间的执行次序由区段的启动模块在段落图形中的位置决定。执行次序由上到下。

下图说明了功能块图的执行次序：

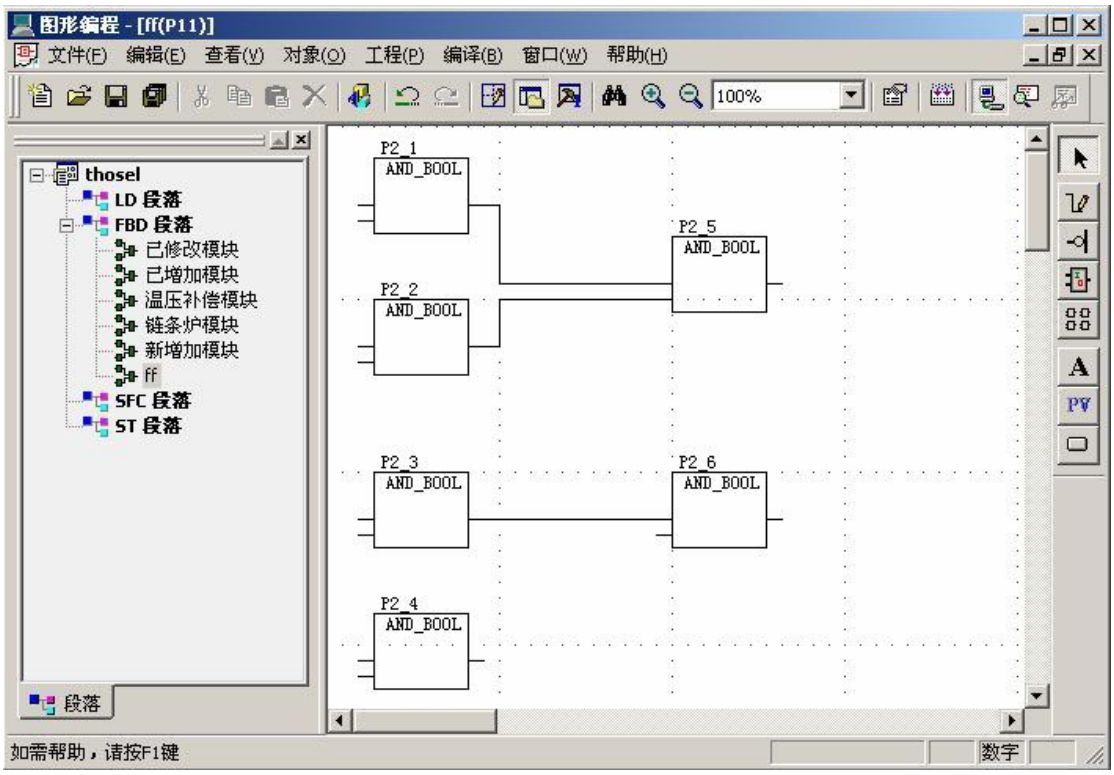

图 3-17 功能块图执行次序

3.3.7 FBD 语言编程

FBD 语言的编程分为创建新工程、创建新程序段、程序段的编程、工程的编译、连接等几个步骤：

- 创建一个新工程（project）

在图形编程软件编程环境中，点击“ 文件 ”菜单项，在弹出的菜单条中，选择“ 新建工程 ”，弹出一个“ 新建工程 ”对话框：

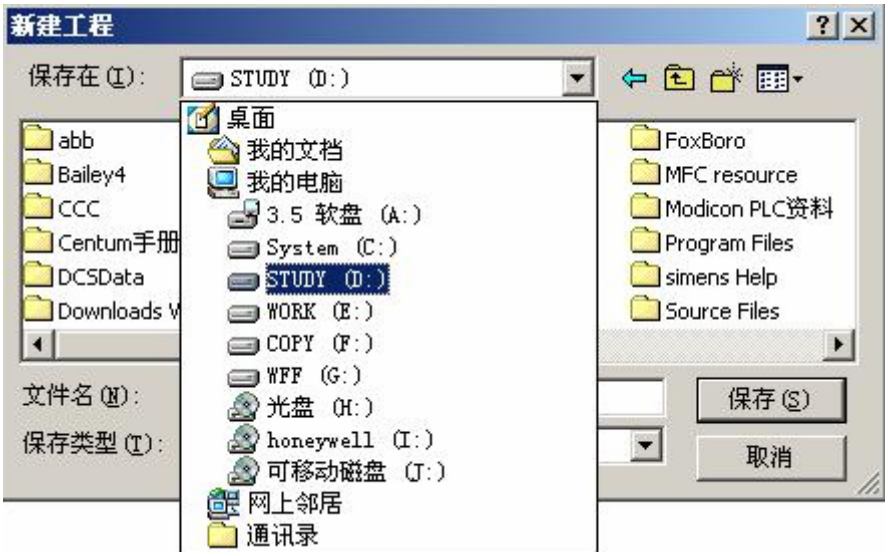

图 3-18 新建工程

用户可通过浏览，选择工程存放的位置，在“ 文件名 ”框内填上工程名（工程文件名以.prj 为后缀名），单击“ 保存 ”按钮，一个新工程就创建成功了。单击“ 取消 ”按钮，即放弃当

前工程的创建工作。

在图形编程软件编辑环境中，可同时创建多个工程，这些工程将同时显示在工作空间(Workspace)中。

■ 创建一个新程序段

创建成功一个新工程后，还需要创建一个或多个程序段。

在图形编程编辑环境中，点击“ 文件 ”菜单项，在弹出的菜单条中，选择“ 新建程序段 ”，弹出一个 “ 新建程序段 ” 对话框：

图 3-19 新建程序段

在对话框的 “ 程序类型 ” 框中选择 “ 功能块图 ”，即 FBD 语言类型；

接下来，用户可在程序和模块两种 “ 段类型 ” 之中选择其一：

- 程序——该类型的程序段可独立运行，程序段内可包括一个或多个模块；
- 模块——该类型的程序段相当于一般高级语言的子程序，需要别的程序调用方可发挥作用，不能独立运行。

用户在根据实际需要选择好段类型后，在“ 段名 ”框内填入程序段的名字，单击“ 确定 ”按钮，即成功创建一个新的程序段。单击 “ 取消 ” 按钮，即放弃当前程序段的创建工作。

如果用户已经创建或打开多个工程，则在创建一个新程序段时，要注意先将相应的工程激活（即确定先点击了该工程名），然后再创建，以免新建于其他的工程下。

■ 程序段的编程

FBD 编辑器用于将基本功能/功能块（EFB）和信号（变量）排列成功能块图。FBD 语言的编程类似于 LD 语言的编程，相比较而言，FBD 的编程不需要让逻辑行从母线以接点输入开始，同样可自由运用大量的功能模块等工具，遵循编程原则进行。

FBD 编程语言用来将程序段构造成一些基本功能和基本功能块、导出功能块和用户自定义功能块，可通过实际参数或链接进行连接。

- 基本元素

功能和功能块（联结时就变成了逻辑单元）

- 编程原则

a. 变量必须先声明再使用；

b. 输入输出类型必须一致；

c. 功能块和变量可以有注释；

d. 不允许通过链路构成环路。

- 编程技巧

对于比较复杂或较大的 FBD 编程，宜先将程序分为简单的程序段，然后再逐段编程。

■ 编译、连接

用户编程后，将工程存盘，即可调用编译命令，对工程进行编译；反复调试，直至编译正确；然后，调用“设置相关控制站地址”命令，弹出对话框：

图 3-20 控制站地址设置

用户在对话框中填入下位机的主控卡地址，单击“确定”；随后，单击“连接”菜单项或按钮，即可完成与下位机的连接。

完成上述步骤后，调用“显示状态”功能，即可观察到程序实时运行的效果。

3.4 梯形图（LD）语言

根据 IEC1131-3，LD 编辑器将基本的功能/功能块（EFB）、线圈、触点和信号（变量、位号）组成梯形图（LD）。图形内可以自由放置基本元素和注释文本。部分 EFB 的输入可以扩展方便使用。

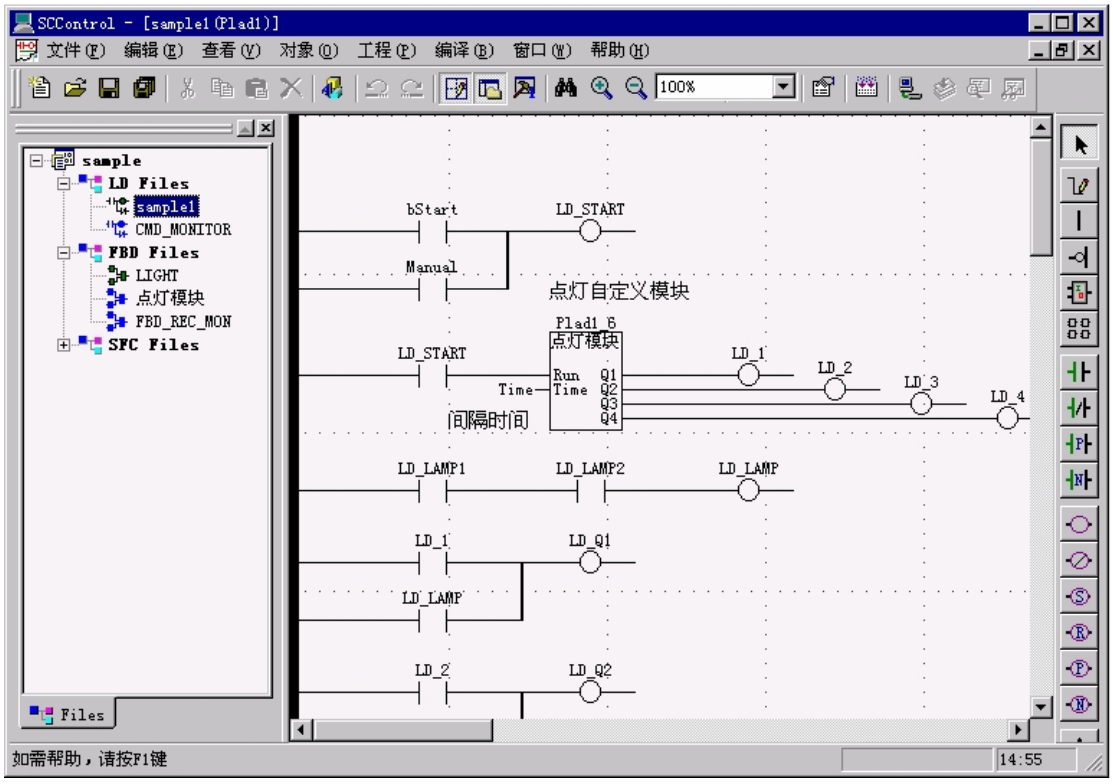

图 3-21 LD 编辑器

LD 段落的设计对应于继电器开关的梯级（rung）。图形的左边是汇流条，相应于梯级的相线（L）。只有直接或间接与相线有开关量相连的元素在编程期间被“扫描”。右汇流条缺省不画出。但可以认为所有的线圈和 FFB 开关量输出都连接到右汇流条上，从而建立电流回路。

在 LD 编程中，用户可以使用基本功能块和自定义功能块、线圈和触点。

用户用 DFB 编辑器生成的自定义模块被放在自定义模块库中。DFB 加入模块库中后，就可以被各种语言编辑器使用。

在 LD 编辑器中，窗口的背景是逻辑坐标网格。当正在创建时，功能块或线圈触点在该网格的光栅中对准。如果发生与别的功能块重叠的情况，将会出现错误信息并且功能块不能创建。当实际参数在功能块输入/输出创建时，他们可能与别的目标重叠但不会破坏区段画面的界限。如果有一链路作为与别的功能块的连接，则该连接要进行检查。如果这是一个未经许可的连接，该链路将不生成。当触点靠近左汇流条时，自动生成链路。触点与触点、触点与线圈靠近时，也会自动生成链路。

除这些目标之外，也可以将注释文本和调试文本放入 LD 段落中。该文本目标的大小取决于文本的长度。根据文本大小，目标的大小可以通过在垂直以及在水平方向上更多的网格来进行扩展。文本不占用控制站内存。

下面将从以下几个方面分别作出介绍：

- LD 编辑器
- 触点
- 线圈
- FFB 模块
- 链接
- 执行次序
- 导入和导出
- 工程文件管理
- LD 编程

3.4.1 LD 编辑器

根据 IEC61131-3，LD 编辑器将基本的功能/功能块（EFB）、线圈、触点和信号（变量、位号）组成梯形图（LD）。图形内可以自由放置基本元素和注释文本。

LD 段落的设计对应于继电器开关的梯级（rung）。图形的左边是汇流条，相应于梯级的相线（L）。只有直接或间接与相线有开关量相连的元素在编程期间被“扫描”。右汇流条缺省不画出。但可以认为所有的线圈和 FFB 开关量输出都接到右汇流条上，从而建立电流回路。

在 LD 编程中，用户可以使用基本功能块和自定义功能块、线圈和触点。

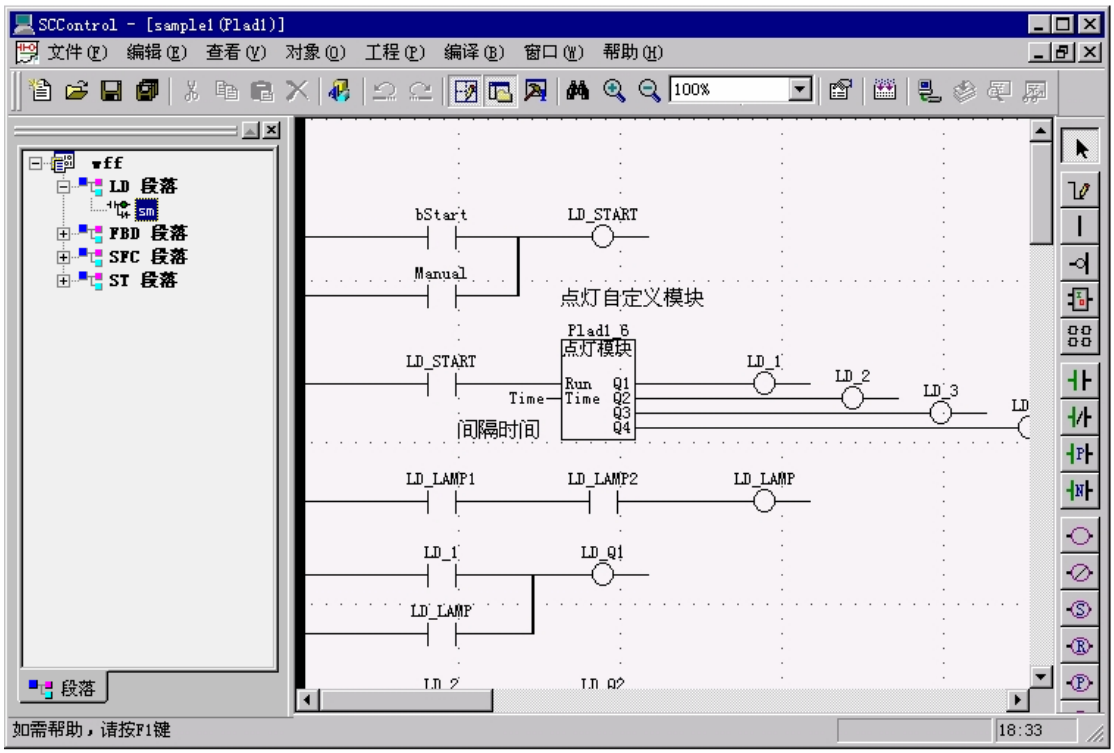

图 3-22 LD 编辑器

3.4.2 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。

- 常开触点
- 常闭触点
- 正跳变触点
- 负跳变触点

常开触点

在常开触点中，如果和触点相关的变量(IN1)为 ON 时，左链路的状态复制到右链路；否则右链路的状态为 OFF。

下图用梯形图和功能块图的方法描述了常开触点：

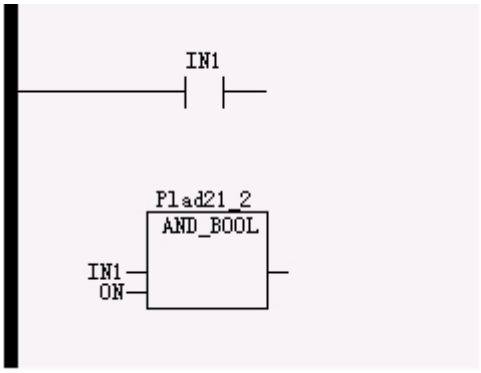

图 3-23 常开触点梯形图和功能块图

下图描述了常开触点在用垂直连接线连接后的原理示意：

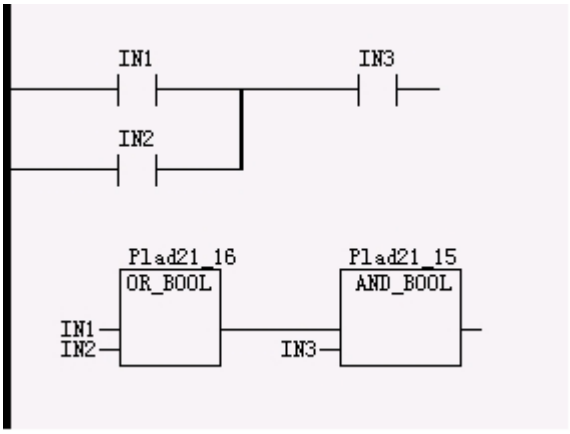

图 3-24 连接后的常开触点

常闭触点

在常闭触点中，如果和触点相关变量(IN1)的状态为 OFF 时，左链路的状态复制到右链路；否则右链路的状态为 OFF。

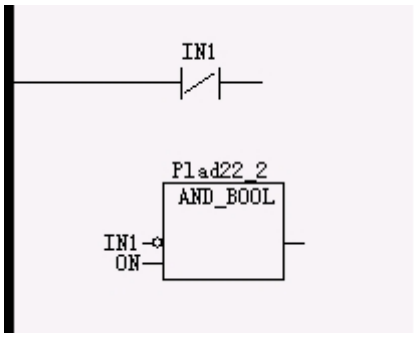

图 3-25 常闭触点

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。

正跳变触点

在正跳变触点中，如果和触点相关 BOOL 变量(IN1)的状态从 OFF 跳变为 ON 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；否则右链路的状态为 OFF。

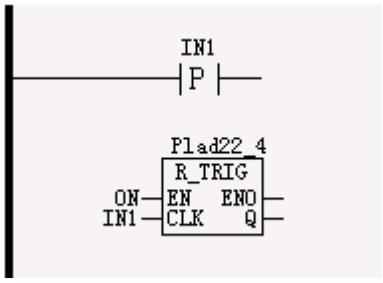

图 3-26 正跳变触点

负跳变触点

在负跳变触点中，如果和触点相关 BOOL 变量(IN1)的状态从 ON 跳变为 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；否则右链路的状态为 OFF。

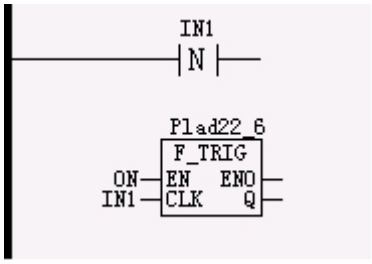

图 3-27 负跳变触点

3.4.3 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。

- 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

常开线圈

在线圈中，左链路的状态复制到相关的布尔变量和右链路。线圈通常跟在触点之后，但线圈之后也可以接触点单元。 常开线圈对应于 MOVE 功能。

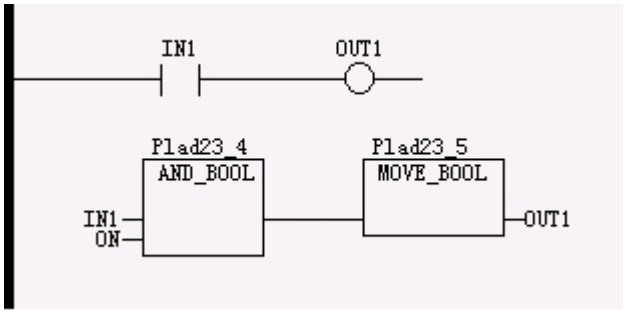

图 3-28 常开线圈

常闭线圈

在取反线圈中，左链路的状态复制到右链路。左链路的取反状态复制至相关的布尔变量(OUT1)。如果左链路为 OFF，则右链路将为 OFF，而相关变量将为 ON。

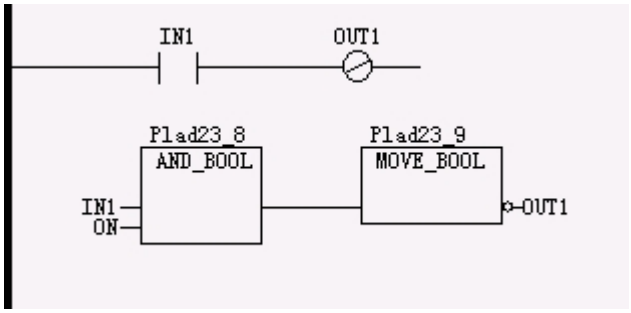

图 3-29 常闭线圈

取反线圈对应于带有反相输出的 MOVE 功能

置位线圈

在置位线圈中，左链路的状态复制至右链路。如果左链路为 ON，则相关的布尔变量(OUT1)置为 ON；否则相关的布尔变量保持不变直至程序或人工修改其值。相关布尔变量能够借助复位线圈复位。置位线圈对应于输入固定为 ON 的 MOVE 功能。

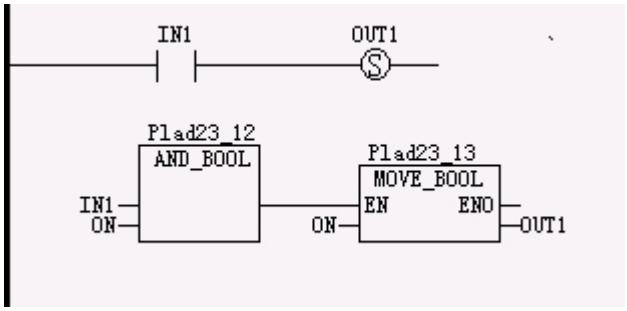

图 3-30 置位线圈

复位线圈

在复位线圈中，左链路的状态复制至右链路。如果左链路为 ON，则相关的布尔变量(OUT1)置为 OFF；否则保持不变直至程序或人工修改其值。相关布尔变量能够借助置位线圈置位。复位线圈对应于输入固定为 OFF 的 MOVE 功能。

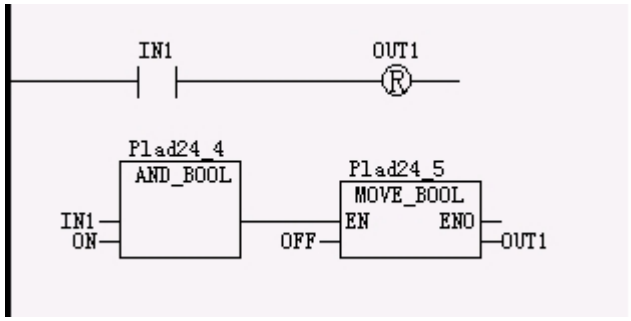

图 3-31 复位线圈

正跳变线圈

在正跳变线圈中，左链路的状态复制至右链路。如果左链路从 OFF 跳变为 ON，则相关的布尔变量(OUT1)将在下一个程序周期内为 ON。正跳变线圈对应于 R_TRIG 功能块。

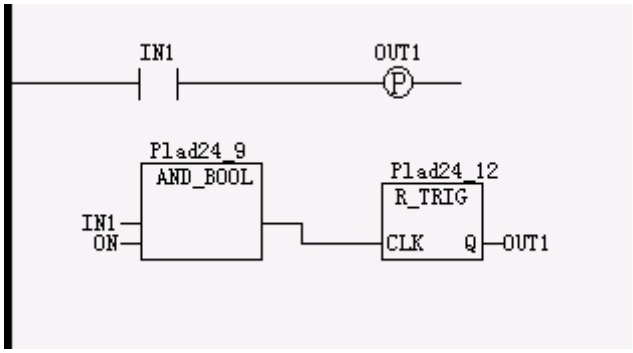

图 3-32 正跳变线圈

负跳变线圈

在负跳变线圈中，左链路的状态复制至右链路。如果左链路从 ON 跳变为 OFF，则相关的布尔变量(OUT1)将在下一个程序周期内为 ON。负跳变线圈对应于 F_TRIG 功能块。

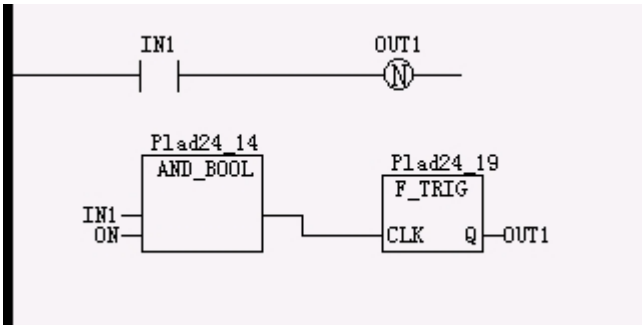

图 3-33 负跳变线圈

3.4.4 链接

链接是功能块之间的连接。一个功能块输出可以连接多个功能块的输入，要连接的输入/输出必须要有相应的数据类型。链接允许与其他目标重叠，链接不能循环配置。循环必须通过实际参数来解决。

在 LD 段落中当触点靠近左汇流条时，自动生成链路。触点与触点、触点与线圈靠近时，也将自动生成链路。

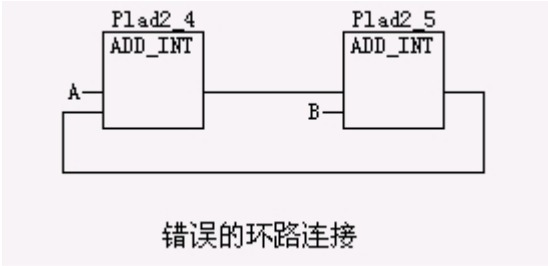

图 3-34 错误的环路连接

解决方法：

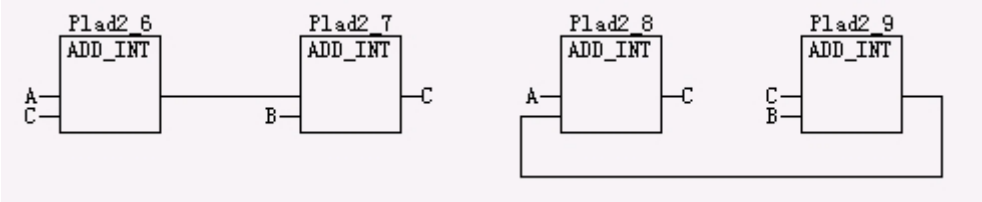

图 3-35 正确的连接

3.4.5 执行次序

在 LD 区段输入只连接变量或位号或常数或左汇流条的被称为区段的**起始模块**。

区段内有多个起始模块时，在图形区域中位置最上的模块称为**启动模块**。

LD 区段从启动模块开始执行。

LD 区段内的执行次序由区段内的数据流决定。

LD 段落中区段间的执行次序由区段的启动模块在段落图形中的位置决定。执行次序由上到下。如下图所示：

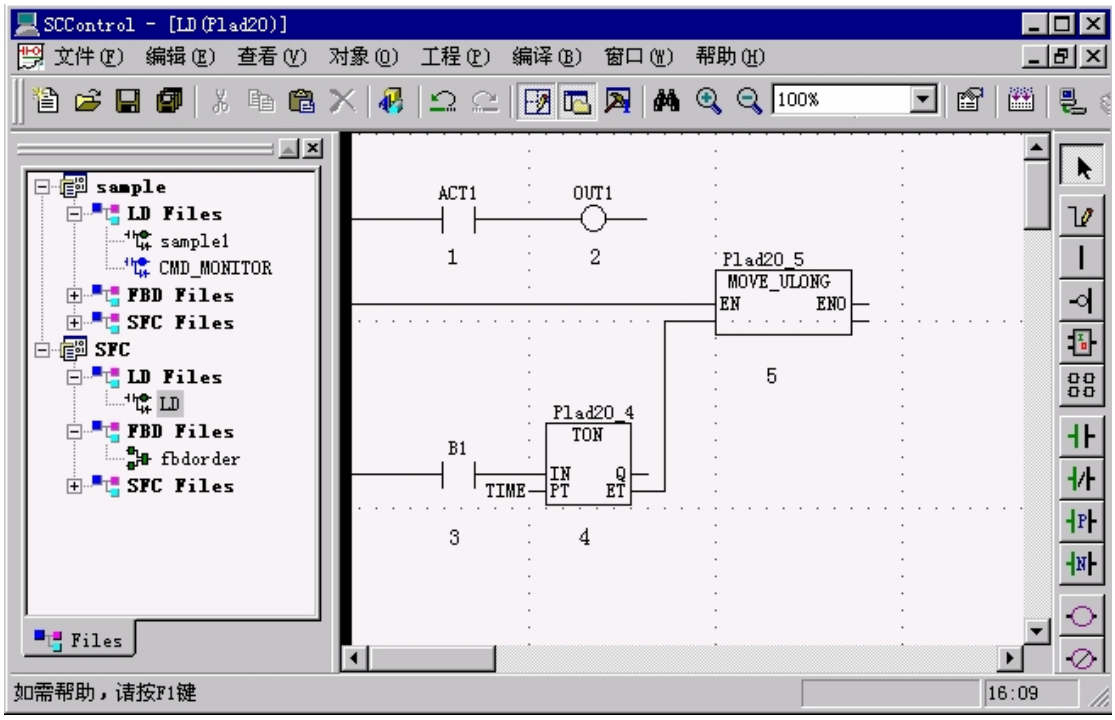

图 3-36 执行次序

3.4.6 LD 语言编程

LD 语言的编程分为创建新工程、创建新程序段、程序段的编程、工程的编译、连接等几个步骤：

■ 创建一个新工程（project）

在 SCControl 软件编程环境中，点击“ 文件 ”菜单项，在弹出的菜单条中，选择“ 新建工程 ”，弹出一个“ 新建工程 ”对话框：

图 3-37 新建工程

用户可通过浏览，选择工程存放的位置，在“ 文件名 ”框内填上工程名（工程文件名以.prj 为后缀名），单击“ 保存 ”按钮，一个新工程就创建成功了。单击“ 取消 ”按钮，即放弃当前工程的创建工作。

在 SCControl 软件编辑环境中，可同时创建多个工程，这些工程将同时显示在工作空间(Workspace)中。

■ 创建一个新程序段

创建成功一个新工程后，还需要创建一个或多个程序段。

在 SCControl 编辑环境中，点击“ 文件 ”菜单项，在弹出的菜单条中，选择“ 新建程序段 ”，弹出一个“ 新建程序段 ”对话框：

图 3-38 新建程序段

在对话框的“ 程序类型 ”框中选择“ 梯形图 ”，即 LD 语言类型；

接下来，用户可在程序和模块两种“ 段类型 ”之中选择其一：

- 程序——该类型的程序段可独立运行，程序段内可包括一个或多个模块；
- 模块——该类型的程序段相当于一般高级语言的子程序，需要别的程序调用方可发挥作用，不能独立运行。

用户在根据实际需要选择好段类型后，在“ 段名 ”框内填入程序段的名字，单击“ 确定 ”按钮，即成功创建一个新的程序段。单击“ 取消 ”按钮，即放弃当前程序段的创建工作。

如果用户已经创建或打开多个工程，则在创建一个新程序段时，要注意先将相应的工程激活（即确定先点击了该工程名），然后再创建，以免新建于其他的工程下。

■ 程序段的编程

LD 编辑器用于将基本功能/功能块（EFB）、触点、线圈和信号（变量）整理成梯形图。作为一种图形编程工具，SCControl 可被用户运用软件提供的大量功能模块、线圈、触点等，通过选取、拖动、放置、移动等操作，以及利用基本功能块创建自定义功能块等轻松地完成一系列编程工作。

- 基本元素

  功能和功能块、触点、线圈

- 编程原则

  a. 变量必须先说明再使用；

  b. 梯形图的每一逻辑行必须从左边母线以接点输入开始；

  c. 接点的使用次数不受限制；

  d. EFB、触点、线圈和变量可以有注释；

  e. 生成链路时，允许与其它链路和目标重叠、交叉。

- 编程技巧

  对于比较复杂或较大的 LD 编程，宜先将程序分为简单的程序段，然后再逐段编程。

■ 编译、连接

用户编程后，将工程存盘，即可调用编译命令，对工程进行编译；反复调试，直至编译正确；然后，调用“设置相关控制站地址”命令，弹出对话框：

图 3-39 设置控制站地址

用户在对话框中填入下位机的主控卡地址，单击“确定”；随后，单击“连接”菜单项或按钮，即可完成与下位机的连接。

完成上述步骤后，调用“显示状态”功能，即可观察到程序实时运行的效果。

3.5 顺控图 SFC

SFC 编辑器实现的元素和结构

步 Step

步是一段控制程序，用来执行规定的动作(Action)。动作是一组赋值语句，可以给常规控制模块的参数赋值。步有 3 种类型，起始步、常规步、结束步。每个 SFC 模块有一个起始步和一个结束步。常规步可以有多个。起始步在 SFC 模块启动时执行，结束步在 SFC 模块正常运行结束或放弃执行时执行，以保证 SFC 模块结束在固定状态。

转换 Transition

转换是一段条件判断程序，用来实现步间的运行转换。转换中包含条件。当条件判断满足，转换的输出值为 TRUE，转换之前的步执行结束，然后执行转换之后的步。

链路 Link

用来连接步和转换。

文本 Text

Text 是注释文本，用来使 SFC 模块增强可读性。文本可以放在绘图区中任何地方。

顺序结构

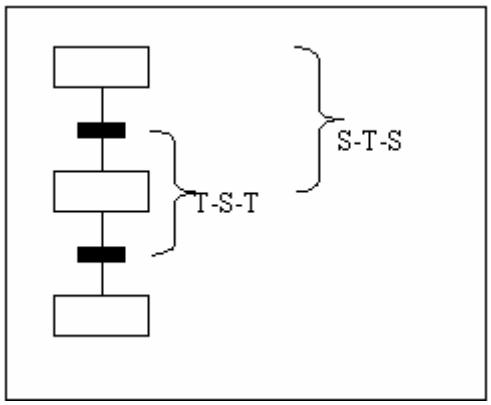

图 3-40 顺序结构

顺序结构用来实现**步**的顺序执行。顺序结构按结构划分可归结为 S-T-S 型和 T-S-T 型。S-T-S 型结构两头都是**步**，单独一个**步**也是 S-T-S 型结构。T-S-T 型结构两头都是**转换**，单独

一个转换也是 T-S-T 型结构。

在顺序结构中，某步激活时，它下面的转换开始判断内部的条件是否全满足，如果没有全满足，不断执行此步，如果内部的条件全满足则转到下一步继续执行。

并联结构

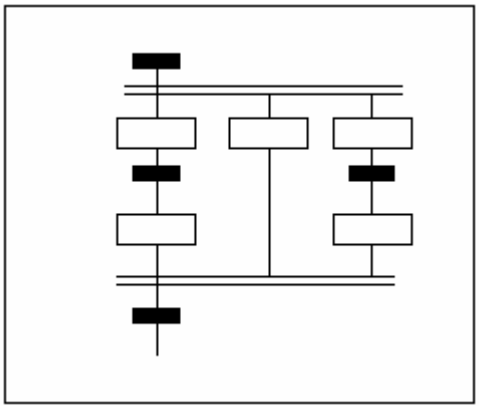

图 3-41 并联结构

并联结构用来实现几个顺序支路的并联执行。适用于几个操作需要同时进行的场合。并联结构以图形化表示的并联分支（双横线）开始，结束于以双横线表示的并联接合。在并联结构中，每个顺序分支必须是 S-T-S 结构。并联结构上面只有一个公共转换是允许的。当此公共转换为真时，各顺序支路并行启动，此后，各顺序支路彼此独立处理。当所有的顺序支路都处理完，才判断并联接合后的转换。

选择结构

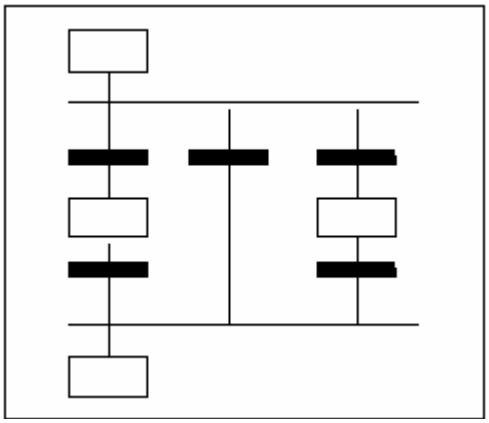

图 3-42 选择结构

选择结构实现在几个顺序支路中任意选择一个执行的功能。使用在当某个操作完成后，可根据当前工业过程的状态来选择一个操作的场合。选择结构以图形化表示的单横线选择分支开始，结束于以单横线表示的选择接合。在选择结构中，每个分支必须是 T-S-T 结构。如果同时有几个分支的条件都满足，则系统按从左到右的优先级选择一路执行。从同一个选择分支引出的顺序支路必须交汇到同一个选择接合上，或通过跳转跳出。

跳转

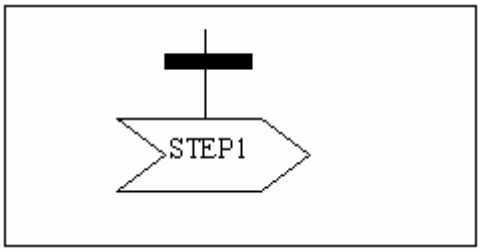

图 3-43 跳转

跳转允许程序从不同的区域继续运行。跳转有两类结构，顺序跳转和循环。

顺序跳转实现在选择分支中在条件满足时，跳过几步继续运行。如下图所示：执行完 S1 后，可以在条件满足的情况下，跳过 S2，直接执行到 S3。

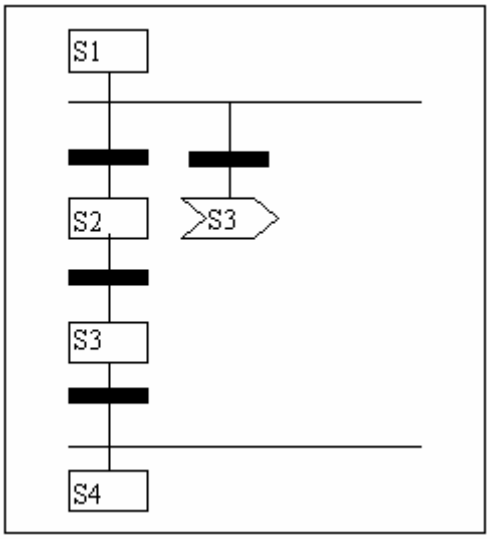

图 3-44 顺序跳转

循环实现重复操作。

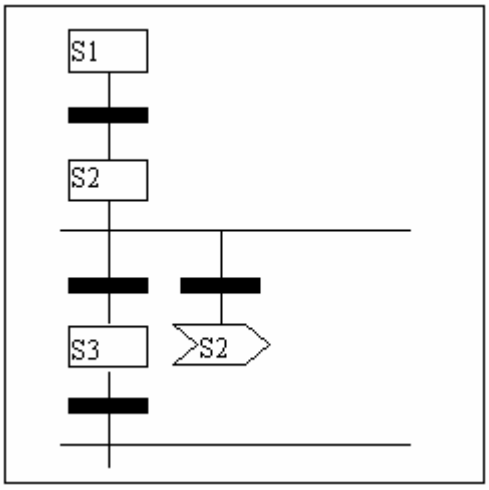

图 3-45 循环

如上图所示：执行完 S1 后执行 S2，执行完 S2 后，只要条件满足，就可以不执行接下去的 S3，而再一次执行了 S2。如此可重复执行多次 S2，直到重复的条件不再满足，方开始执行下面的 S3。

跳转的图形是一个箭头，内有跳转到的步名称。

3.5.1 概述

SFC 编辑器实现的元素和结构包括了：

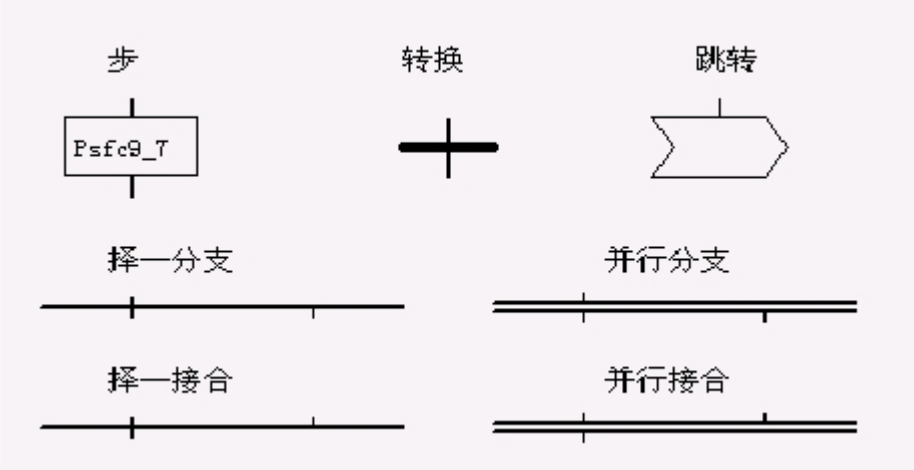

图 3-46 SFC 编辑器元件和结构

3.5.2 步（step）

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量的各种类型的操作（Action），以此来实现对流程的控制。

步在激活时才执行相应的操作，它只有在紧接它上面的转换条件满足时才被激活，步中的操作全都执行完毕之后如果紧接在它后面的转换条件满足才退出激活状态。

步的上面只能接转换、并行分支或择一接合；步的下面只能接转换、并行接合或择一分支。

步有三种类型：起始步、普通步、终止步，如下图所示

图 3-47 步的三种类型

在一幅 SFC 图中，起始步和终止步必须有且各只有一个。SFC 的执行从起始步开始到终止步结束。

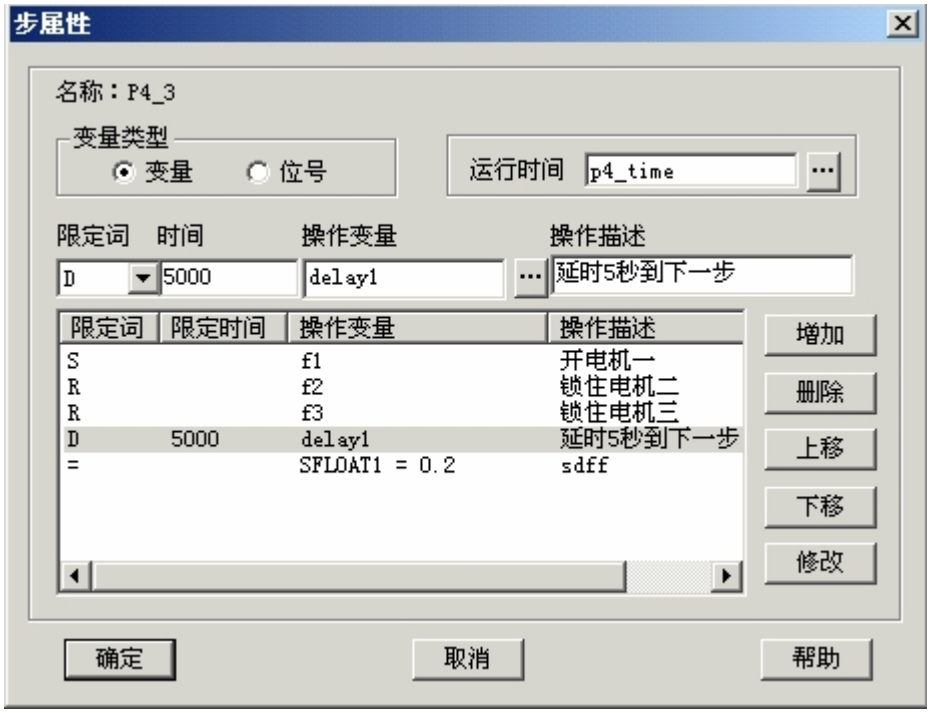

图 3-48 步属性

- 运行时间

指定将步的激活时间赋给 ULONG 类型的变量用于显示。

- 变量类型

指定选择变量时使用系统位号浏览还是使用变量浏览。

- 限定词

指定当前操作的类型

- 时间

指定当前操作的限定时间

- 操作变量

用于指定操作

- 操作描述

用于对当前操作添加注释。

通过增加删除按钮在当前步中加入或删除操作。

选择操作后可以通过上移和下移更改操作的执行次序。

通过修改按钮可以修改当前操作。

3.5.3 转换（Transition）

转换用来指明将控制从一个步转移到其它步的条件。

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都在激活状态时，转换的条件才被测试。

转换条件由一个布尔变量或布尔表达式定义。

转换的上面只能接步、择一分支、并行接合。转换的下面只能接步、择一接合、并行分支、跳转。

3.5.4 跳转（Jump）

跳转允许程序从不同的步继续执行。根据跳转对象的不同，可以构成顺序跳转和顺序环路，不能在不同的并行区域间跳转。

下图为顺序跳转。顺序跳转主要用于故障处理。

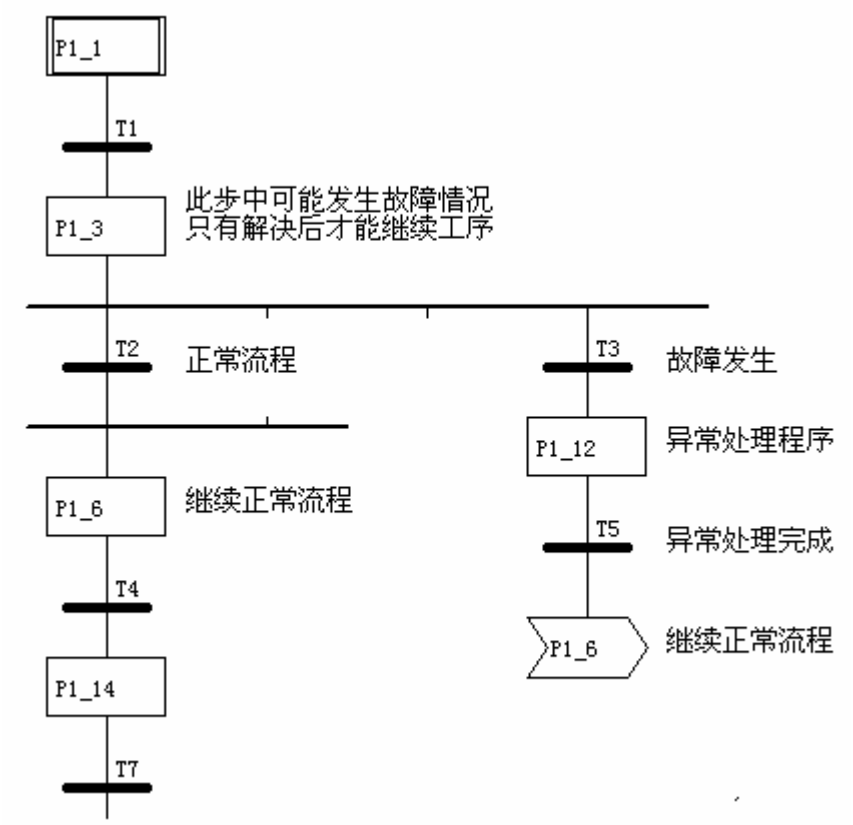

图 3-49 顺序跳转

下图为顺序环路：

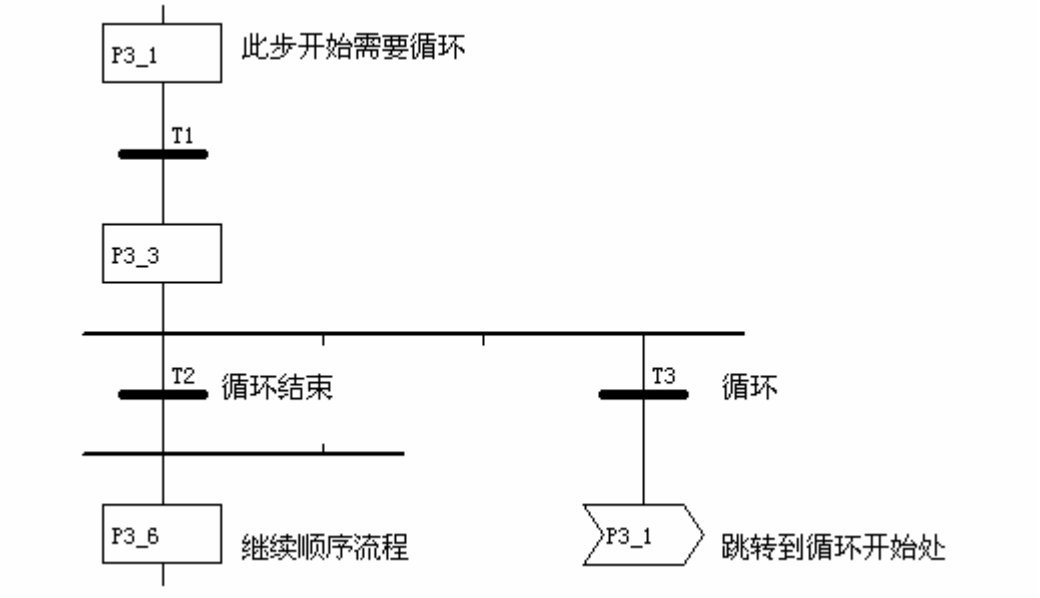

图 3-50 顺序环路

3.5.5 择一分支（Alternative Branch）

择一分支提供了在 SFC 程序中实现条件控制的控制流程选择执行的方法。

在择一分支结构内只能有一个分支被激活。

分支跳转的优先级从左到右（如果几个分支的判断变量同时满足，那么执行最左边分支）。

择一分支和择一接合必须一一对应。

分支必须结束于同一择一接合或者结束于跳转。

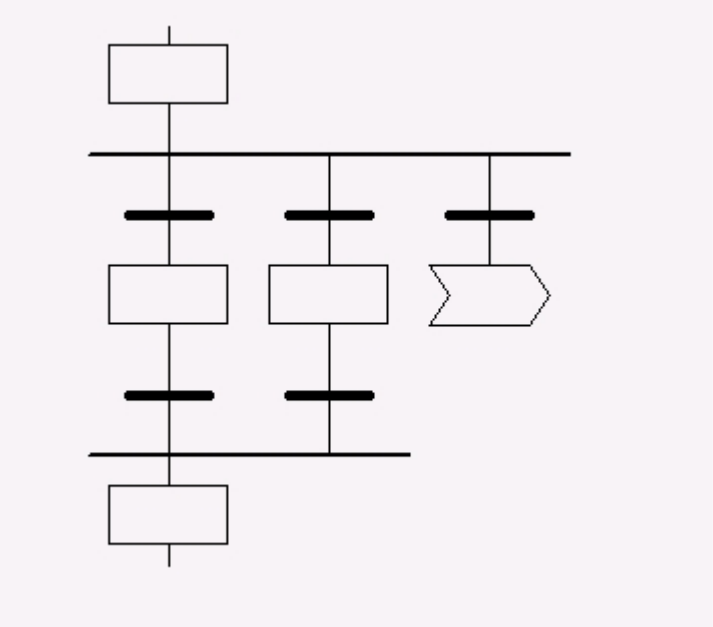

图 3-51 择一分支

3.5.6 并行分支

并行分支使流程中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，才测试并行接合紧接的转换的条件是否满足。

并行分支和并行接合必须一一对应。

在并行结构内部的跳转不能跳到并行结构的外部。

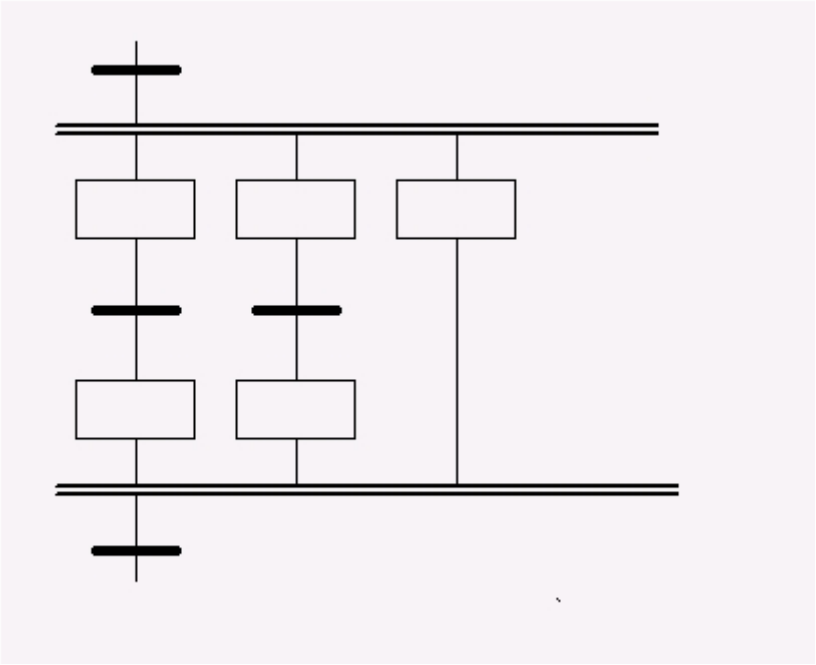

图 3-52 并行分支

3.5.7 操作

操作是对系统信号（变量、位号）进行操纵的描述。

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来描述。操作可以是一个布尔变量（操作变量），也可以是一个赋值表达式。

在步属性窗口内可以编辑操作。

图形编程软件 SFC 编辑器提供以下符合 IEC1131-3 标准的操作限定词：

1) N 操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
2) S 操作在步激活后将一直保持激活
3) R 操作在步激活后将一直保持在不激活状态
4) L 操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
5) D 操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
6) P 操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
7) DS 操作在步激活后经过限定的时间后，变为激活状态，并一直维持激活状态

在以上类型的操作中，操作变量只能定义为布尔量，其中使用 L 、D、DS 操作限定词时必须指定限定时间。**限定时间的单位是 ms**。

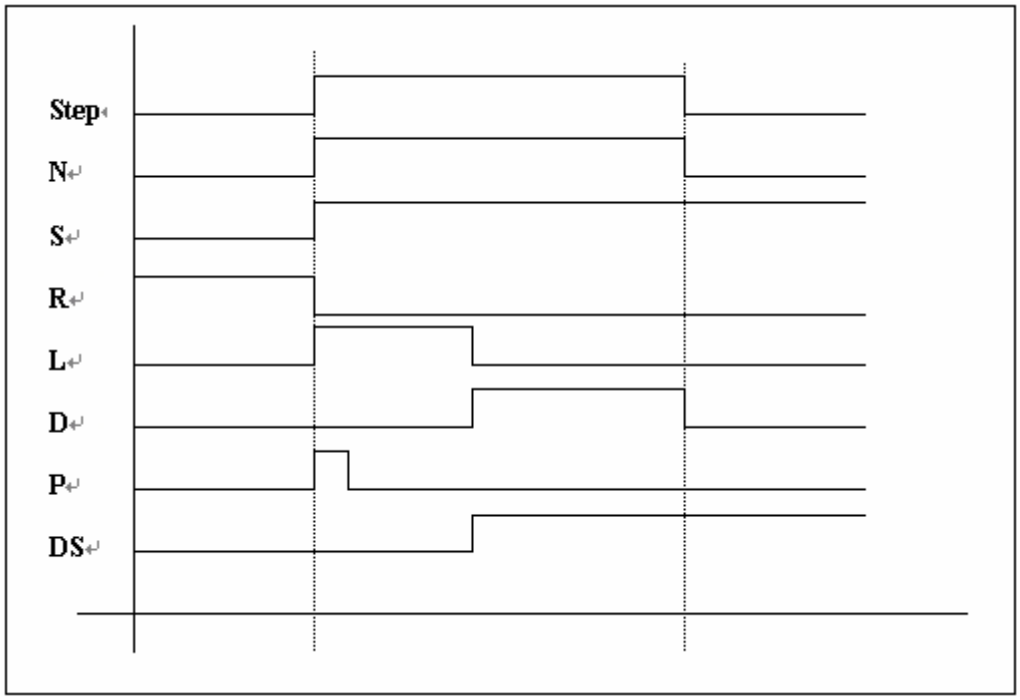

图 3-53 操作

图形编程软件 SFC 编辑器提供了扩展的操作限定词：

= 赋值操作限定词。表示在步的整个激活期间赋值操作一直进行直到步退出激活状态恢复成不激活状态。

在使用赋值操作限定词时，必须指定赋值表达式才可以增加操作，如需复杂表达式可以双击该操作编辑 ST 语言程序。

SFC 控制变量

可以设置以下控制变量来控制 SFC 程序的运行：

- 运行控制
- 复位
- 禁止转换
- 强制步进
- 操作使能

运行控制变量为 ON 时，SFC 程序正常执行。为 OFF 时，所有其它控制变量都无效，SFC 程序停止运行。程序必须指定一个运行控制变量。

复位变量状态为 ON 时，SFC 程序起始步被设置为激活步，其它步都强制变为不激活状态，顺控程序从头开始重新执行。当运行变量为 OFF 时，复位变量无效。

禁止转换变量为 ON 时，当前激活步将一直保持执行而不管紧接的转换条件是否满足，转换条件测试将不进行。禁止转换变量受运行变量和复位变量的影响。

强制步进变量为 ON 时，当前激活步不管转换条件是否满足，都变为不激活状态，按顺序的下一步变为激活状态。强制步进变量受以上所有变量的影响。

操作使能变量操作：操作使能变量为 ON 时，步中的操作才被执行。

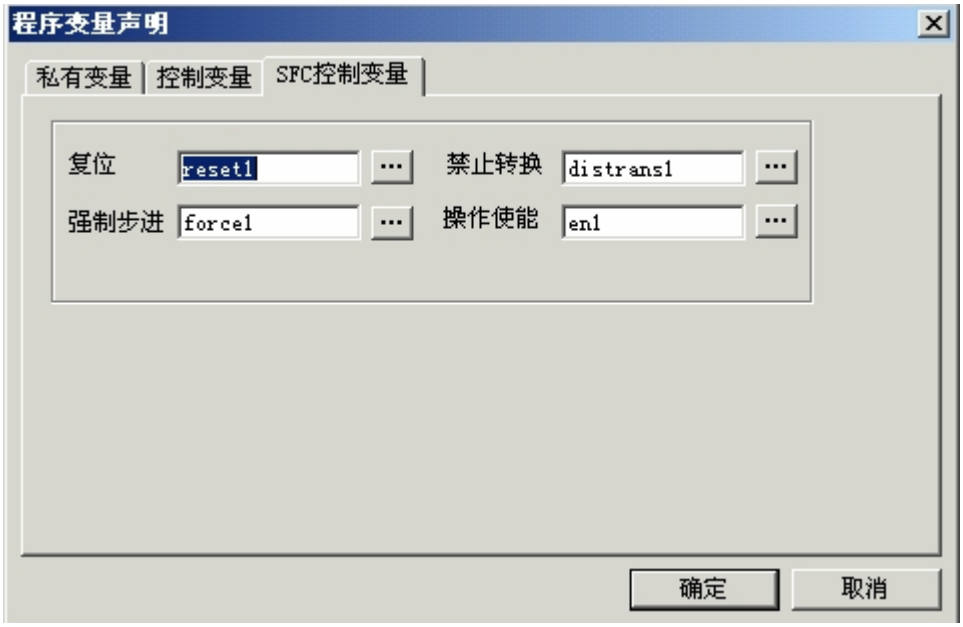

图 3-54 程序变量声明

3.6 ST 语言

ST 语言在图形编程中和其它图形编程语言组合使用。实现了 IEC1131-3 标准的一个子集。

使用方法

在工程中加入 ST 语言段落。

可以在梯形图和功能块图中插入文本代码模块。在模块中用 ST 语言编程。

可以在顺控图中的步的操作中使用 = 操作限定词，然后可以用 ST 语言编程。

在顺控图的转换条件中可以使用 ST 语言的逻辑表达式来指定条件。

在 SFC 编辑器中，当指定转换条件时可以使用 ST 语言逻辑表达式。

在操作中用 = 操作符可以使用 ST 的语句，双击该条操作即可弹出编辑界面。

注意：使用循环语句会明显增加编译时间，为获得更快的编译速度建议尽可能不使用 WHILE 语句并且少使 REPEAT/FOR 语句。

3.6.1 ST 语言语法

1) ST 语言语法

ST 语言在图形编程软件中和其它图形编程语言组合使用。实现了 IEC61131-3 标准的一个子集。支持多种数据类型，支持函数、结构和数组，可以操作各种系统变量。

表达式

表达式为变量、操作符、常量、函数的组合，求值结果为单个值。

表达式的求值按运算符的优先级进行，优先级高的运算符先被处理。相同优先级的运算符按从左到右的顺序执行。

以下为几个合法的表达式举例：

A + B * (C-3) + FUNC1(2,D)

B1

FUNC1()

使用方法

变量、函数等标识符的命名必须满足下列条件：

✓ 以英文字母开头；

由英文字母、数字或下划线组成；

✓ 字符长度最多为 24 个字符。

标识符包括变量、函数、功能块、常数。

关键字

关键字	描述
CASE…OF…ELSE…END_CASE	CASE 语句
BOOL WORD DWORD INT LONG UINT ULONG SFLOAT FLOAT	数据类型
EXIT	终止循环
FALSE	逻辑假
FOR…TO…BY…DO…END_FOR	FOR 语句
FUNCTION…END_FUNCTION	函数定义
FUNCTION_BLOCK END_FUNCTION_BLOCK	功能块定义
IF…THEN…ELSEIF…ELSE…END_IF	IF 语句
ON	逻辑真
OFF	逻辑假
REPEAT…UNTIL…END_REPEAT	REPEAT 语句
RETURN	函数返回
TRUE	逻辑真
VAR…END_VAR VAR_INPUT…END_VAR VAR_OUTPUT…END_VAR	变量定义
WHILE…DO…END_WHILE	WHILE 语句

运算符

按运算优先级从高到低有：

运算符	描述	类型	优先级
()	表达式运算		9
.	取结构成员		8
[]	取数组成员		8
-	单目负		7
NOT	取反	逻辑运算	7
* (MUL)	乘	算术运算	6
/ (DIV)	除	算术运算	6
MOD	取余	算术运算	6
+ (ADD)	加	算术运算	5
- (SUB)	减	算术运算	5
>	大于	比较运算	4
>=	大于等于	比较运算	4
<=	小于等于	比较运算	4

<	小于	比较运算	4
=	等于	比较运算	4
<>	不等于	比较运算	4
AND	与	逻辑运算	3
XOR	异或	逻辑运算	2
OR	或	逻辑运算	1

语句

以下为允许的语句列表。

No.	语句	例子
1	赋值语句	A = B; A = B + 1;
2	函数调用、功能块调用	A = FUNC(P1,P2); FB1(IN1,OUT1,OUT2);
3	RETURN	A = FUNC(P1,P2); RETURN A;

4	IF	IF A > 0 THEN
		B = 1;
		ELSEIF A > -5 THEN
		B = 2;
		ELSE
		B = 3;
		END_IF;

5	CASE	TW = FUNC1(); CASE TW OF 1 : I = 1; 2 : I = 2; ELSE I = 3; END_CASE;

6	FOR	J = 10;
		FOR I = 1 TO 100 BY 2 DO
		IF B1 THEN
		J = 1;
		EXIT;
		END_IF;
		END_FOR;

7	WHILE	J = 1; WHILE J <= 100 AND B1 DO J = J + 2; END_WHILE;
8	REPEAT	J = 1; REPEAT J = J + 2; UNTIL J = 101 OR B1 END_REPEAT;

9	EXIT	J = 1;
		WHILE J <= 100 AND B1 DO
		J = J + 2;
		IF J>= 50 THEN
		EXIT;

		END_IF; END_WHILE;
10	EMPTY	FOR I = 1 TO 100 BY 2 DO ; END_FOR;

赋值语句

赋值语句将“=”右边表达式的值赋给左边的变量。

函数调用语句

函数和功能块的调用包括函数名或功能块名加小括号对，括号内为参数，参数间由逗号隔开。

函数的调用规则：

ret = Func(in1,in2)；（*作为表达式返回值*）

Func(in1,in2)；（*作为子程序处理*）

功能块的调用规则：

调用功能块时要严格按照输入输出顺序，先输入输入参数，再输入输出参数，参数顺序按照定义时的顺序。

输出参数必须是变量： FuncBlock(in1,in2,out1,out2)。

2) ST 语言函数和功能块

函数定义

只有一个输出（变量类型可以自行确定），根据输入可以唯一确定输出。

功能块定义

有多个输出，或输出不但和当前输入有关还和上次内部状态有关。

除系统内部的函数和功能块外，用户可以自定义函数和功能块。自定义的函数和功能块可以在工程内 LD/FBD 段落中调用。

函数的调用规则

ret = Func(in1,in2); （*作为表达式返回值*）

Func(in1,in2); （*作为子程序处理*）

功能块的调用规则

调用功能块时要严格按照输入输出顺序，先输入输入参数，再输入输出参数，参数顺序按照定义时的顺序。

输出参数必须是变量。

FuncBlock(in1,in2,out1,out2);

使用限制

ST 的 FUNCTION 模块

可以调用其它 ST FUNCTION,允许嵌套

可以调用标准函数

ST 的 FUNCTION_BLOCK 模块

可以调用 ST FUNCTION

可以调用其它 ST FUNCTION_BLOCK,但是所调用的 FUNCTION_BLOCK 不允许嵌套

可以调用标准函数

TEXTCODE 模块

可以调用 ST FUNCTION

可以调用 ST FUNCTION_BLOCK

可以调用标准函数

可以调用各种编程语言生成的功能块

编译速度

由于当前 SCControl 所使用的 ARM 编译器（SDT251）的缘故，使用循环语句，尤其是其中的 WHILE 语句，会明显增加编译时间，为获得更快的编译速度建议尽可能不使用 WHILE 语句以及少使用 REPEAT/FOR 语句。

3) FUNCTION

```
FUNCTION FUNC3 : BOOL
VAR_INPUT
        IN1:BOOL;
END_VAR
VAR
        TEMP1 : BOOL;
END_VAR
    FUNC3 := DoSomething();
END_FUNCTION
```

说明：

VAR_INPUT/END_VAR 用于说明功能块的输入变量；

VAR/END_VAR 用于说明功能块内部的临时变量（临时变量存储在系统堆栈中，不能维持状态到下一周期）；

各类变量声明的次序不能颠倒；

调用时要严格按照声明的先后次序。

4) FUNCTION_BLOCK

```
FUNCTION_BLOCK FB3
VAR_INPUT
        IN1:BOOL;
END_VAR

VAR_OUTPUT
        OUT1 : BOOL;
END_VAR

VAR
        TEMP1 : BOOL;
END_VAR

DoSomething();

END_FUNCTION_BLOCK
```

说明：

VAR_INPUT/END_VAR 用于说明功能块的输入变量；

VAR_OUTPUT/END_VAR 用于说明功能块的输出变量；

VAR/END_VAR 用于说明功能块内部的临时变量（临时变量存储在系统堆栈中，不能维持状态到下一周期）；

各类变量声明的次序不能颠倒；

不能对输出变量进行自操作；

功能块调用时要严格按照声明的先后次序。

5) ST 语言程序实例

赋值语句

赋值语句将“=”右边表达式的值赋给左边的变量。

```
A = B ;
A = B + C;
A = B * C + D;
A = B AND C AND D OR E;
A = AND_DWORD ( B,C);
```

IF 语句

```
IF （A AND (B > C) OR (E > F + 1) ） THEN
        AA = BB;
ELSEIF ( B > G) THEN
            AA = CC;
ELSE
    AA = DD;
END_IF;
```

IF 语句规定了一组语句在规定的逻辑表达式为 TRUE 时执行。当逻辑表达式为 FALSE 时，这些语句不被执行，或在 ELSE(ELSEIF)中规定的另一组语句被执行。

CASE 语句

```
CASE  A  OF
1:
    AA = BB;
2:
    AA = CC;
3:
    AA = DD;
ELSE
     AA = EE;
END_CASE;
```

CASE 语句规定了整数类型的选择项，以及选择项在不同的值时的几组语句组。当选择项等于某个规定的值时，相应的语句组被执行，当没有规定的值符合时在 ELSE 中的语句组将被执行（在 CASE 语句中定义了 ELSE 分支）。

FOR 语句

```
FOR I = 1 TO 100 BY 2 DO
        DOSOMETHING();
END_FOR;
```

在以上 FOR 语句中，I 为控制变量，1 为初始值，100 为终止值，2 为步进值。在 FOR 语句中控制变量的初始值、终止值、步进值必须是相同的整型。步进值缺省为 1。终止条件

的判断一开始就进行，当初始值大于终止值时，规定的语句组一次都不会执行。

WHILE 语句

WHILE 条件 DO

语句组

END_WHILE;

条件的判断一开始就进行，如条件一开始就变 FALSE 时，规定的语句组一次都不会执行。

REPEAT 语句

REPEAT

语句组

UNTIL 终止条件

END_REPEAT;

终止条件的判断在语句组执行一次后才进行，所以规定的语句组至少会执行一次。当终止条件成 TRUE 时，循环被终止。

3.6.2 ST 可调用函数列表

• 算术运算
• 比较函数
• 转换函数
• 逻辑运算
• 数学函数
• 选择函数
• 通讯辅助函数
• 系统时间函数
• 特殊函数
• 其他函数
• 输入处理函数
• 智能通讯卡处理函数

1) 算术运算

- 加法函数

  这些函数的功能是将输入值相加，并将结果赋给输出值。函数类型如下：

  FLOAT ADD_FLOAT(FLOAT a, FLOAT b)
  INT ADD_INT(INT a, INT b)
  LONG ADD_LONG(LONG a, LONG b)
  UINT ADD_UINT(UINT a, UINT b)
  ULONG ADD_ULONG(ULONG a, ULONG b)
  SFLOAT ADD_SFLAOAT(SFLOAT a,SFLOAT b)

- 平均函数

  这些函数的功能是求输入值的平均值，并将结果赋给输出值。函数类型如下：

  FLOAT AVE_FLOAT(FLOAT a, FLOAT b)
  INT AVE_INT(INT a, INT b)
  LONG AVE_LONG(LONG a, LONG b)
  UINT AVE_UINT(UINT a, UINT b)
  ULONG AVE_ULONG(ULONG a, ULONG b)
  SFLOAT AVE_SFLOAT(SFLOAT a,SFLOAT b)

  对于 AVE_FLOAT 函数，两个输入之和不能超出浮点的量程。

- 除法函数

  这组函数的功能是将输入值相除，并将结果赋给输出值。函数类型如下：

  FLOAT DIV_FLOAT(FLOAT a, FLOAT b)
  INT DIV_INT(INT a, INT b)
  LONG DIV_LONG(LONG a, LONG b)
  UINT DIV_UINT(UINT a, UINT b)
  ULONG DIV_ULONG(ULONG a, ULONG b)
  SFLOAT DIV_SFLOAT(SFLOAT a,SFLOAT b)

- 赋值函数

  该组函数的功能是将输入赋给输出值。函数类型如下所示：

  BOOL MOVE_BOOL(BOOL a);
  BYTE MOVE_BYTE(BYTE A);
  DWORD MOVE_DWORD(DWORD a);
  FLOAT MOVE_FLOAT(FLOAT a);
  INT MOVE_INT(INT a);
  LONG MOVE_LONG(LONG a);
  UINT MOVE_UINT(UINT a);
  ULONG MOVE_ULONG(ULONG a);
  WORD MOVE_WORD(WORD a);
  SFLOAT MOVE_SFLOAT(SFLOAT a);

- 乘法函数

  该组函数的功能是将输入值相乘，并将结果赋给输出值。函数的类型如下所示：

  FLOAT MUL_FLOAT(FLOAT a, FLOAT b);
  INT MUL_INT(INT a, INT b);
  LONG MUL_LONG(LONG a, LONG b);
  UINT MUL_UINT(UINT a, UINT b);

ULONG MUL_ULONG(ULONG a, ULONG b);
SFLOAT MUL_SFLOAT(SFLOAT a,SFLOAT b);

- 减法函数

该组函数的功能是将输入值相减，并将结果赋给输出值。函数类型如下所示：

FLOAT SUB_FLOAT(FLOAT a, FLOAT b);
INT SUB_INT(INT a, INT b);
LONG SUB_LONG(LONG a, LONG b);
UINT SUB_UINT(UINT a, UINT b);
ULONG SUB_ULONG(ULONG a, ULONG b);
SFLOAT SUB_SFLOAT(SFLOAT a,SFLOAT b)';

2) 比较函数

- 等于比较

该组函数功能是检查第一个输入值是否等于第二个输入值，若是，则输出值为 ON，否则为 OFF。函数类型如下所示：

BOOL EQ_BOOL(BOOL a, BOOL b);
BOOL EQ_BYTE(BYTE A,BYTE B);
BOOL EQ_DWORD(DWORD a, DWORD b);
BOOL EQ_FLOAT(FLOAT a, FLOAT b);
BOOL EQ_INT(INT a, INT b);
BOOL EQ_LONG(LONG a, LONG b);
BOOL EQ_UINT(UINT a, UINT b);
BOOL EQ_ULONG(ULONG a, ULONG b);
BOOL EQ_WORD(WORD a, WORD b);
BOOL EQ_SFLOAT(SFLOAT a,SFLOAT b);

- 大于等于比较

该组函数功能是检查第一个输入值是否大于等于第二个输入值，若是，则输出值为 ON，否则为 OFF。函数类型如下所示：

BOOL GE_FLOAT(FLOAT a, FLOAT b);
BOOL GE_INT(INT a, INT b);
BOOL GE_LONG(LONG a, LONG b);
BOOL GE_UINT(UINT a, UINT b);
BOOL GE_ULONG(ULONG a, ULONG b);
BOOL GE_SFLOAT(SFLOAT a,SFLOAT b);

- 大于比较

该组函数功能是检查第一个输入值是否大于第二个输入值，若是，则输出值为 ON，否则为 OFF。函数类型如下所示：

BOOL GT_FLOAT(FLOAT a, FLOAT b);
BOOL GT_INT(INT a, INT b);
BOOL GT_LONG(LONG a, LONG b);
BOOL GT_UINT(UINT a, UINT b);
BOOL GT_ULONG(ULONG a, ULONG b);
BOOL GT_SFLOAT(SFLOAT a,SFLOAT b);

- 小于等于比较

该组函数功能是检查第一个输入值是否小于等于第二个输入值，若是，则输出值为 ON，

否则为 OFF。函数类型如下所示：

```
BOOL        LE_FLOAT(FLOAT a, FLOAT b);
BOOL        LE_INT(INT a, INT b);
BOOL        LE_LONG(LONG a, LONG b);
BOOL        LE_UINT(UINT a, UINT b);
BOOL        LE_ULONG(ULONG a, ULONG b);
BOOL       LE_SFLOAT(SFLOAT a,SFLOAT b);
```

- 小于比较

该组函数功能是检查第一个输入值是否小于第二个输入值，若是，则输出值为 ON，否则为 OFF。函数类型如下所示：

```
BOOL        LT_FLOAT(FLOAT a, FLOAT b);
BOOL        LT_INT(INT a, INT b);
BOOL        LT_LONG(LONG a, LONG b);
BOOL        LT_UINT(UINT a, UINT b);
BOOL        LT_ULONG(ULONG a, ULONG b);
BOOL       LT_SFLOAT(SFLOAT a,SFLOAT b);
```

- 不等比较

该函数的功能是对两个输入值进行比较，若输入值不等，则输出值为 ON，否则为 OFF。N 和 ENO 能作为附加参数加以设置。 函数类型如下所示：

```
BOOL        NE_BOOL(BOOL a, BOOL b);
BOOL        NE_BYTE(BYTE A,BYTE B);
BOOL        NE_DWORD(DWORD a, DWORD b);
BOOL        NE_FLOAT(FLOAT a, FLOAT b);
BOOL        NE_INT(INT a, INT b);
BOOL        NE_LONG(LONG a, LONG b);
BOOL        NE_UINT(UINT a, UINT b);
BOOL        NE_ULONG(ULONG a, ULONG b);
BOOL        NE_WORD(WORD a, WORD b);
BOOL       NE_SFLOAT(SFLOAT a,SFLOAT b);
```

3) 转换函数

- LONG DWORD_TO_LONG(DWORD a);

该函数功能是将 DWORD 型的输入值转化为 LONG 型数据类型。

- INT FLOAT_TO_INT(FLOAT a);

该函数功能是将 FLOAT 型的输入值转化为 INT 型数据类型。

- FLOAT INT_TO_FLOAT(INT a);

该函数功能是将 INT 型的输入值转化为 FLOAT 型数据类型。

- LONG INT_TO_LONG(INT a);

该函数功能是将 INT 型的输入值转化为 LONG 型数据类型。

- UINT INT_TO_UINT(INT a);

该函数功能是将 INT 型的输入值转化为 UINT 型数据类型。

- WORD INT_TO_WORD(INT a);

该函数功能是将 INT 型的输入值转化为 WORD 型数据类型。

- DWORD LONG_TO_DWORD(LONG a);

该函数功能是将 LONG 型的输入值转化为 DWORD 型数据类型。

- FLOAT LONG_TO_FLOAT(LONG a);
  该函数功能是将 LONG 型的输入值转化为 FLOAT 型数据类型。
- INT LONG_TO_INT(LONG a);
  该函数功能是将 LONG 型的输入值转化为 INT 型数据类型。
- ULONG LONG_TO_ULONG(LONG a);
  该函数功能是将 LONG 型的输入值转化为 ULONG 型数据类型。
- INT UINT_TO_INT(UINT a);
  该函数功能是将 UINT 型的输入值转化为 INT 型数据类型。
- ULONG UINT_TO_ULONG(UINT a);
  该函数功能是将 UINT 型的输入值转化为 ULONG 型数据类型。
- WORD UINT_TO_WORD(UINT a);
  该函数功能是将 UINT 型的输入值转化为 WORD 型数据类型。
- DWORD ULONG_TO_DWORD(ULONG a);
  该函数功能是将 ULONG 型的输入值转化为 DWORD 型数据类型。
- LONG ULONG_TO_LONG(ULONG a);
  该函数功能是将 ULONG 型的输入值转化为 LONG 型数据类型。
- UINT ULONG_TO_UINT(ULONG a);
  该函数功能是将 ULONG 型的输入值转化为 UINT 型数据类型。
- BYTE WORD_TO_BYTE(WORD A);
  该函数功能是将 WORD 型的输入值转化为 BYTE 型数据类型。
- WORD BYTE_TO_WORD(BYTE A);
  该函数功能是将 BYTE 型的输入值转化为 WORD 型数据类型。
- INT WORD_TO_INT(WORD a);
  该函数功能是将 WORD 型的输入值转化为 INT 型数据类型。
- UINT WORD_TO_UINT(WORD a);
  函数功能是将 WORD 型的输入值转化为 UINT 型数据类型。
- FLOAT SFLOAT_TO_FLOAT(SFLOAT A);
  函数功能是将 SFLOAT 型的输入值转化为 FLOAT 型数据类型。
- SFLOAT FLOAT_TO_SFLOAT(FLOAT A);
  函数功能是将 FLOAT 型的输入值转化为 SFLOAT 型数据类型。
- INT SFLOAT_TO_INT(SFLOAT A);
  函数功能是将 SFLOAT 型的输入值转化为 INT 型数据类型。
- SFLOAT INT_TO_SFLOAT(INT A);
  函数功能是将 INT 型的输入值转化为 SFLOAT 型数据类型。
- INT DEC_TO_BCD(BYTE dec);
  函数功能：同十进制转换为 BCD 码模块，具体见 FBD 说明书。

4) 逻辑运算

- **逻辑与函数**

该组函数的功能是将输入值进行该逻辑与操作，并将结果赋给输出值。函数类型如下所示：

```
BOOL      AND_BOOL(BOOL a, BOOL a);
BYTE      AND_BYTE(BYTE A,BYTE B);
DWORD     AND_DWORD(DWORD a, DWORD b);
```

WORD AND_WORD(WORD a, WORD b);

- **逻辑取反函数**

该组函数的功能是将输入值进行逻辑取反操作，并将结果赋给输出值。函数类型如下所示：

BOOL NOT_BOOL(BOOL a);
BYTE NOT_BYTE(BYTE A);
DWORD NOT_DWORD(DWORD a);
WORD NOT_WORD(WORD a);

- **逻辑或函数**

该函数的功能是将输入值进行逻辑或操作，并将结果赋给输出值。函数类型如下所示：

BOOL OR_BOOL(BOOL a, BOOL b);
BYTE OR_BYTE(BYTE A,BYTE B);
DWORD OR_DWORD(DWORD a, DWORD b);
WORD OR_WORD(WORD a, WORD b);

- **循环左移函数**

该组函数功能是将输入值 IN 进行循环左移，并将结果赋给输出值。 函数类型如下所示：

DWORD ROL_DWORD(DWORD a, UINT b);
WORD ROL_WORD(WORD a, UINT b);

- **循环右移函数**

该组函数功能是将输入值 IN 进行循环右移，并将结果赋给输出值。函数类型如下所示：

DWORD ROR_DWORD(DWORD a, UINT b);
WORD ROR_WORD(WORD a, UINT b);

- **逻辑左移函数**

该组函数功能是将输入值 IN 进行左移（从右边填零），并将结果赋给输出值 OUT。函数类型如下所示：

DWORD SHL_DWORD(DWORD a, UINT b);
WORD SHL_WORD(WORD a, UINT b);

- **逻辑右移函数**

该组函数功能是将输入值 IN 进行右移（从左边填零），并将结果赋给输出值 OUT。函数类型如下所示：

DWORD SHR_DWORD(DWORD a, UINT b);
WORD SHR_WORD(WORD a, UINT b);

- **逻辑异或函数**

该组函数的功能是将输入值进行逻辑异或操作，并将结果赋给输出值。函数类型如下所示：

BOOL XOR_BOOL(BOOL a, BOOL b);
BYTE XOR_BYTE(BYTE A,BYTE B);
DWORD XOR_DWORD(DWORD a, DWORD b);
WORD XOR_WORD(WORD a, WORD b);

5) 数学函数

- FLOAT ABS_FLOAT(FLOAT a);

INT ABS_INT(INT a);
LONG ABS_LONG(LONG a);
这三个函数的功能是计算输入值的绝对值并将结果赋给输出值。

- FLOAT ACOS(FLOAT a);
  该函数功能是计算输入值的反余弦值，并将结果以弧度的形式赋给输出值。

- FLOAT ASIN(FLOAT a);
  该函数功能是计算输入值的反正弦值，并将结果以弧度的形式赋给输出值。

- FLOAT ATAN(FLOAT a);
  该函数的功能是计算输入值的反正切值，并将结果以弧度的形式赋给输出值。

- FLOAT ATAN2(FLOAT a, FLOAT b);
  该函数的功能是计算坐标（x,y)对应的反正切值，并将结果以弧度的形式赋给输出值。

- FLOAT COS(FLOAT a);
  该函数功能是计算输入值的余弦值，并将结果赋给输出值。

- FLOAT COSH(FLOAT a);
  该函数功能是计算输入值的工程余弦值，并将结果赋给输出值。输入值必须是弧度形式。

- FLOAT EXP(FLOAT a);
  该函数功能是计算以 e 为底，输入值 IN 为指数的幂级数，并将结果赋给输出值。

- FLOAT LN(FLOAT a);
  该函数功能是计算输入值自然对数，并将结果赋给输出值。

- FLOAT LOG(FLOAT a);
  该函数功能是计算以 10 为底的对数，并将结果赋给输出值。

- FLOAT POW(FLOAT a, FLOAT b);
  该函数功能是计算 y 为指数，x 为底的幂级数，并将结果赋给输出值 OUT。

- FLOAT SIN(FLOAT a);
  该函数功能是计算输入值 IN 的正弦值，并将结果赋给输出值 OUT。

- FLOAT SINH(FLOAT a);
  该函数功能是计算输入值 IN 的工程正弦值,并将结果赋给输出值。

- FLOAT SQRT_FLOAT(FLOAT a);
  SFLOAT SQRT_SFLOAT(SFLOAT a);
  这两个函数功能是计算输入值 IN 的平方根，并将结果赋给输出值 OUT。 对于 SQRT_SFLOLAT 函数，其输入值在 0 到 1 之间。

- FLOAT TAN(FLOAT a);
  该函数功能是计算输入值 IN 的正切值，并将结果赋给输出值 OUT。

- FLOAT TANH(FLOAT a);
  函数功能是计算输入值 IN 的工程正切值，并将结果赋给输出值 OUT。

6) 选择函数

- 限幅函数

该组函数的功能是限幅，即当输入大于上限值时输出上限值，当输入小于下限值时输出下限值，否则输出输入值。函数类型如下所示：

```
FLOAT     LIM_FLOAT(FLOAT max, FLOAT a, FLOAT min);
INT       LIM_INT(INT max, INT a, INT min);
LONG      LIM_LONG(LONG max, LONG a, LONG min);
UINT      LIM_UINT(UINT max, UINT a, UINT min);
ULONG     LIM_ULONG(ULONG max, ULONG a, ULONG min);
SFLOAT    LIM_SFLOAT(SFLOAT max,SFLOAT a,SFLOAT min);
```

- 最大值函数

该组函数的功能是将输入值中的最大值赋给输出值。函数类型如下所示：

```
FLOAT     MAX_FLOAT(FLOAT a, FLOAT b);
INT       MAX_INT(INT a, INT b);
LONG      MAX_LONG(LONG a, LONG b);
UINT      MAX_UINT(UINT a, UINT b);
ULONG     MAX_ULONG(ULONG a, ULONG b);
SFLOAT    MAX_SFLOAT(SFLOAT a,SFLOAT b);
```

- 最小值函数

该组函数的功能是将输入值中的最小值赋给输出值。函数类型如下所示：

```
INT       MIN_INT(INT a, INT b);
LONG      MIN_LONG(LONG a, LONG b);
UINT      MIN_UINT(UINT a, UINT b);
ULONG     MIN_ULONG(ULONG a, ULONG b);
SFLOAT    MIN_SFLOAT(SFLOAT a,SFLOAT b);
```

- 选择函数

该组函数的功能是当 SW=OFF 时，将输入值 IN1 赋给输出值；当 SW=ON 时，将输入值 IN2 赋给输出值。函数类型如下所示：

```
BOOL      SEL_BOOL(BOOL sw, BOOL a, BOOL b);
DWORD     SEL_DWORD(BOOL sw, DWORD a, DWORD b);
FLOAT     SEL_FLOAT(BOOL sw, FLOAT a, FLOAT b);
INT       SEL_INT(BOOL sw, INT a, INT b);
LONG      SEL_LONG(BOOL sw, LONG a, LONG b);
UINT      SEL_UINT(BOOL sw, UINT a, UINT b);
ULONG     SEL_ULONG(BOOL sw, ULONG a, ULONG b);
WORD      SEL_WORD(BOOL sw, WORD a, WORD b);
SFLOAT    SEL_SFLOAT(BOOL sw,SFLOAT a,SFLOAT b);
```

- BOOL SEL_1IN3(BOOL pv1,BOOL pv2,BOOL pv3);

函数功能：同三选一开关信号选择模块。

- BOOL SEL_1IN5(BOOL pv1,BOOL pv2,BOOL pv3,BOOL pv4,BOOL pv5);

  函数功能：同五选一开关信号选择模块。

7) 通讯辅助函数

- BOOL GETBIT(DWORD num, UINT serial);

函数功能：从输入的 DWORD 型变量 num 中取出指定位的标号，如果该标号为 1，则输出为 ON，如果为 0，则输出为 OFF。

serial 取值范围是 0～31，如果 serial 大于 31，那么输出 OFF。

- FLOAT GETFLOAT(DWORD num);

  函数功能：将输入 DWORD 型变量用 FLOAT 型来解释输出。

- INT GETINT(DWORD num, UINT serial);

函数功能：从 32 位的 DWORD 型输入变量 num 的指定位置取出 16 位的 INT 型变量，当 SERIAL＝0 时，取低 16 位；当 SERIAL≠0 时，取高 16 位。

- DWORD GETMSG(UINT nStation, UINT serial);

函数功能：该模块根据输入的控制站序号 nStation 和消息序号 serial，直接从接收缓冲区中读出指定控制站指定位置的信息。

- UINT GETUINT(DWORD num, UINT serial);

函数功能：从 32 位的 DWORD 型输入变量 num 的指定位置取出 16 位的 UINT 型变量，当 serial＝0 时，取低 16 位；当 serial≠0 时，取高 16 位。

- WORD GETWORD(DWORD num, UINT serial);

函数功能：从 32 位的 DWORD 型输入变量 num 的指定位置取出 16 位的 WORD 型变量，当 serial＝0 时，取低 16 位；当 serial≠0 时，取高 16 位。

- SFLOAT GETSFLOAT(DWORD num,UINT serial);

函数功能：该模块的功能是从输入的 32 位 DWORD 型值的指定位置取 16 位的 SFLOAT 型值， 当 serial=0，取低 16 位；当 serial≠0，取高 16 位。

- void SENDMSG(UINT size);

函数功能：该模块根据输入的消息个数，使能控制站共享信息广播，广播所指定的消息个数到 SCnetII 网络，其他控制站通过 SCnetII 网络收到该控制发送的信息。具体功能见 SENGMSG 功能块。

- DWORD SETBIT(DWORD num, BOOL val, UINT serial);

  函数功能：设置输入 num 的 serial 位的值，当 val 为 OFF 时，该位为 0，反之则为 1。serial 取值范围是 0～31，当 serial>31 时，返回 0。

- DWORD SETFLOAT(FLOAT val);

函数功能：将输入 val 安 DWORD 型来解释输出。

- DWORD SETINT(DWORD num, INT val, UINT serial);

函数功能：当 serial＝0，将输入 num 的低 16 位设置为 val，当 serial≠0，将输入 num 的高 16 位设置为 val。

- DWORD SETUINT(DWORD num, UINT val, UINT serial);

函数功能：当 serial＝0，将输入 num 的低 16 位设置为 val，当 serial≠0，将输入 num 的高 16 位设置为 val。

- DWORD SETWORD(DWORD num, WORD val, UINT serial);

函数功能：当 serial＝0，将输入 num 的低 16 位设置为 val，当 serial＝1，将输入 num 的高 16 位设置为 val。

- DWORD SETSFLOAT(DWORD num,SFLOAT val,UINT serial);

函数功能：当 serial＝0，将输入 num 的低 16 位设置为 val，当 serial＝1，将输入 num 的高 16 位设置为 val。

8) 系统时间函数

- INT CENTURY();

函数功能：得到当前年份的前两个数字，与 YEAR()函数得到的值组成当前年份。

- INT YEAR();

函数功能：得到当前年份后两个数字，与 CENTURY()函数得到的值组成当前年份。

- INT MONTH();

函数功能：得到当前月份。

- INT DAY();

函数功能：得到当前日期。

- INT HOUR();

函数功能：得到当前时钟值。

- INT MINUTE();

函数功能：得到当前的分钟值。

- INT SECOND();

函数功能：得到当前的秒数。

9) 特殊函数

- BOOL LIMITOR(INT IN1,INT IN2,INT IN3,INT IN4,INT IN5,INT IN6,INT IN7,INT IN8,INT S9,INT S10);

函数功能：用来监视 8 个以下的数字量输入，并根据 S9 和 S10 设置的状态产生一个 bool 量输出。具体见 FBD 说明书。

- SFLOAT NEWSQRT(SFLOAT IN,SFLOAT KP,SFLOAT DIS);
  函数功能：用来计算半浮点输入 IN 的平方根，输出= KP×SQRT（IN）＋DIS 。

10) 其他函数

- BOOL ISSTANDBY();
  函数功能：取得主控卡冗余状态，返回为 ON 表示处于备用状态，OFF 则为工作状态。

11) 输入处理函数

- WORD GETPATFLAG(INT N);
  函数功能：得到第 N 通道 PAT 卡工作标志，具体见 FBD 说明书。

- WORD GETPATSTATE(INT N);
  函数功能：取得第 N 通道 PAT 卡状态，具体见 FBD 说明书。

- SFLOAT GETPATPV(INT N);
  函数功能：取第 N 通道的 PAT 卡 PV 值。

- FLOAT SATENTHA(FLOAT P);
  函数功能：计算压力范围是 0.1MPa-16.0MPa 的饱和蒸汽的焓值。输入的压力单位是 KPa。具体见 FBD 说明书。

- SFLOAT SATSTEAM(float press,float press0,SFLOAT Flow0);
  函数功能：饱和蒸汽的温压补偿，具体可见 FBD 说明书。

- SFLOAT SATSTEAM_DP(float PRESS,float DENSITY0,SFLOAT FLOW0);
  函数功能：饱和蒸汽的温压补偿，具体可见 FBD 说明书。

- SFLOAT SATSTEAM_EX(BOOL SIGNALSEL,float PRESS,float DENSITY0,SFLOAT SIGNAL);
  函数功能：饱和蒸汽的温压补偿模块，具体见 FBD 说明书。

- SFLOAT COMPENSATE(SFLOAT SteamFlow0,FLOAT SteamTemperture,FLOAT SteamPress,FLOAT DesignV);
  函数功能：过热蒸汽的温压补偿模块，具体可参见 FBD 说明书。

- SFLOAT FXY(SFLOAT x,INT n);
  函数功能：折线表处理函数，x 是输入值，n 是指采用第几个折线表进行插值。

- SFLOAT GET_FXY_X(INT num,INT n);
  函数功能：得到第 n 个折线表第 num 个 x 的值。

- SFLOAT GET_FXY_Y(INT num,INT n);
  函数功能：得到第 n 个折线表第 num 个 y 的值。

- void SET_FXY_X(SFLOAT x,INT num,INT n);
  函数功能：设置第 n 个折线表第 num 个 x 的值。

- void SET_FXY_Y(SFLOAT x,INT num,INT n);
  函数功能：设置第 n 个折线表第 num 个 y 的值。

- SFLOAT LINECPS(SFLOAT IN,SFLOAT KP,SFLOAT DIS);
  函数功能：输出= IN×KP＋DIS。

- void SETPATCON(WORD CON,INT N);
  函数功能：设置第 N 个 PAT 通道的特殊标志字节。

12) 智能通讯卡处理函数

- ULONG GW_DEFBUF(UINT Size)
  功能：定义命令的发送缓冲区
  输入：
  Size: UINT（缓冲区大小，最大不超过 1024）
  输出： ULONG（缓冲区指针）
- void GW_WRITEBUF(ULONG MsgBuf, UINT Serial, BYTE Char)
  功能：向缓冲区写入一个元素
  输入：
  MsgBuf: ULONG (缓冲区指针)
  Serial: UINT (写入位置)
  Char: BYTE (写入数据)
  输出：
  无, (注：如果 Serial 越限,则放弃操作)
- BYTE GW_READBUF(ULONG MsgBuf, UINT Serial)
  功能：从缓冲区读入一个元素
  输入：
  MsgBuf: ULONG (缓冲区指针)
  Serial: UINT (写入位置)
  输出：
  BYTE (返回数组内指定位置数据,如果 Serial 越限,则返回 0)
- BYTE GW_ISCMDFINSHED(BYTE CmdId)
  功能：判断命令是否执行完毕
  输入：
  CmdId: BYTE (命令号 0-255)
  输出： BYTE
  0: 正在执行
  1: 正常结束
  2: 异常结束
- UINT GW_GETRCVLEN(BYTE CmdId)
  功能：取得接收长度
  输入：
  CmdId: BYTE (命令号 0-255)

输出： UINT(返回接收到信息的长度)

- ULONG GW_GETRCVMSG(BYTE CmdId)

  功能： 拷贝接收到的数据到指定缓冲区

  输入：

  CmdId: BYTE (命令号 0-255)

  输出： ULONG (接收数据区指针)

- WORD GW_CRC16(ULONG MsgBuf, WORD Len)

  功能： 对缓冲区中指定长度元素进行 CRC 检验

  输入：

  MsgBuf: ULONG (缓冲区指针)

  Len: WORD(指定长度)

  输出： WORD (16 位的 CRC 校验)

- void GW_STARTCMD(BYTE CmdId)

  功能：手动启动命令执行

  输入：

  CmdId: BYTE (命令号 0-255)

  输出：无

- BYTE GW_FIRSTRUN(BYTE Any)

  功能： 判断是否首次运行

  输入：

  Any ： BYTE 任何数

  输出： BYTE (0:非首次运行， 1：首次运行)

- WORD GW_CRC16x(ULONG MsgBuf,WORD Len);

  功能 :对缓冲区中指定长度元素进行 CRC 检验(CCITT 标准 CRC16 校验)。MsgBuf 为缓冲区指针，Len 为指定长度。函数返回 16 位 CRC 校验值。

  输入：

  MsgBuf：ULONG （缓冲区指针）

  Len： WORD（指定长度）

  输出： WORD（16 位的 CRC 校验）

3.6.3 导入和导出

图形编程中的自定义数据类型、全局变量、段落都可以导入和导出。通过导入和导出，用户可以方便的重用先前的信息，在工程间传递组态。

图形编程的智能导入导出给予用户极大的方便。

导入导出时先使用 **工程** 菜单中的 **工程管理** 命令弹出工程管理对话框。

当在 **数据类型管理** 对话框选择一个或多个自定义数据类型导出时，用户要指定导出数据存放的文件名。图形编程检查所有的数据类型，若发现所选中的数据类型是由未选中的自定义数据类型派生而来，图形编程将自动追加这些更底层的自定义数据类型。导入时，选择已生成的导出文件，工程中将添加所包含的数据类型。

当在 **全局变量** 对话框选择一个或多个变量导出时，用户要指定导出数据存放的文件

名。导出的变量以 ASCII 文本格式存放，易于用户操作。导入时，选择已生成的导出文件，工程中将添加所包含的变量。

当在 **段落管理** 对话框中选择一个或多个段落导出时，用户要指定导出段落存放的文件名。图形编程先检查所有的段落，如段落中包含未被选择的 DFB，则图形编程自动追加这些 DFB 段落。然后检查所有段落中包含的变量的数据类型，若发现其中的数据类型是由自定义数据类型派生而来，图形编程将自动追加这些自定义数据类型。导入时，选择已生成的导出文件，工程中将添加所包含的数据类型、段落。当导入时，发现段落名冲突，将提示用户是替换或保留或用新名导入。

3.6.4 数据类型编辑器

图形编程内置数据类型编辑器，用户通过数据类型编辑器生成自定义的数据类型。定义新数据类型时，可以使用基本数据类型和已在工程中生成的自定义数据类型。

图形编程内置数据类型编辑器，用户可以用数据类型编辑器生成自己的数据类型，并可以在任何编辑变量类型的地方使用。

用**工程**菜单中的**工程管理**命令打开工程管理对话框。再选择**数据类型管理**进入数据类型编辑器。

图 3-55 数据类型编辑器

按添加则弹出添加复合类型对话框，用户可以选择数组或是结构：

图 3-56 添加复合类型

选择一个数组类型按编辑按钮弹出数组声明对话框。

图 3-57 数组声明

选择一个结构类型按编辑按钮弹出结构声明对话框。

图 3-58 结构声明

通过导入/导出功能，用户可以将数据类型保存到一个文件中，其他功能从此文件中导入就可以使用这些定义的数据类型。

系统内已预定义了部分数据类型，这些数据类型只读，用户无法修改删除。

3.6.5 变量编辑器

用户通过变量编辑器定义变量。

定义全局变量、私有变量、输入变量、输出变量的界面统一。可以定义变量名、变量数据类型和注释。

3.6.6 DFB 编辑器

DFB 编辑器用于建立自定义功能块（DFB）。从编程角度看，DFB 可以理解为子程序。

DFB 可以用 FBD 编辑器、LD 编辑器生成，也可以被 FBD、LD、SFC 调用。

图形编程 DFB 用于建立 DFB（Derived Function Blocks 自定义功能块）用 FBD、LD 都可以设计 DFB。

从编程角度上看，一个 DFB 就相当于一个子程序。

一个 DFB 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。用户可以多次重用 DFB。

图 3-59 选择模块

在使用方法上看，DFB 与 EFB 没有不同。

在 DFB 内部可以引用一个或多个 EFB 和其他 DFB。DFB 不能自身循环嵌套。

在图形编程中，当新建或打开模块段落时，DFB 编辑器自动启动。

DFB 变量

DFB 内有 4 类变量：

- 私有变量
- 输入变量
- 输出变量
- 热备份变量

输入变量用作向 DFB 传递数值，输出变量用作从 DFB 传出数值。这两类变量被称为形式参数。这些变量在 DFB 被使用时表现为外部的输入输出引脚。

形式参数的名字、数据类型、引脚序号在变量编辑器中定义。引脚序号即为在变量编辑器中的自然顺序。

在图形编程 DFB 中，输入/输出变量最大可各定义 32 个。

在 DFB 被使用时，实际参数替代形式参数参与 DFB 内部的算法逻辑。

热备份变量对用户不可见，当 DFB 引用了包含热备份变量的 FFB 时，DFB 自动继承了 FFB 的热备份变量。

图形编程 DFB 提供了私有变量。在 DFB 中的私有变量与程序段落中的私有变量不同。程序中的每一个私有变量唯一对应控制站的一个内存地址。而 DFB 中的私有变量在 DFB 不被其他程序引用时，没有对应实际的控制站地址。当其他程序引用 DFB 时，每次引用时，DFB 中的私有变量都被分配一个不同的控制站地址。换句话说，DFB 中的私有变量封装了对控制站内存的申请。每一次 DFB 实例的创建，DFB 都按照私有变量的要求申明了相同数目的控制站内存，这些内存只被 DFB 的实例所引用，对外部程序不可见。图形编程 DFB 对私有变量的支持，极大地提高了 DFB 的灵活性和重用性。

用包含私有变量的 DFB 构建新的 DFB 时，新的 DFB 将继承包含的 DFB 的私有变量。

3.6.7 系统资源

1) 系统资源

- 定时器

100 毫秒定时器 UINT timerms[256]

秒定时器 UINT timers[256]

分定时器 UINT timerm[256]

应用举例

按 START 按钮 20 秒后做某事

```
IF  NOT  START  THEN
  timers[20] = 0;
END_IF;
IF timers[20] >= 20 THEN
  dosomething();
END_IF;
```

- 自定义控制模块

单回路控制模块数据块

g_bsc[128]

串级控制模块数据块

g_csc[128]

注意：g_bsc 和 g_csc 共用系统内存，不能同时使用同一个下标。

- 控制站间共享数据区

DWORD g_msg[128]

控制站共享此数据区用于站间交换数据

注意：各控制站不能共用同一个数组下标。

应用举例

从 2 号控制站（地址为 2，3） 发送数据到其它控制站，使用 6 号数组下标

```
g_msg[6] = 0;
g_msg[6] = setbit ( g_msg[6] , BOOL1 , 0); (*  用第 0 位发送开关量 1 *)
g_msg[6] = setbit ( g_msg[6] , BOOL2 , 1); (*  用第 1 位发送开关量 2 *)
g_msg[6] = setsfloat ( g_msg[6] , SFLOAT1 , 1);  (*  用高 16 位发送模拟量 1 *)
sendmsg ( 7);  (*  使用的最大下标为 6 *)
```

从其它控制站读取 2 号控制站发送的数据(使用了 6 号数组下标)

```
DWORD dw;
dw = getmsg ( 2 , 6);  (* 读取从 2 号控制站来的数据，使用 6 号数组下标  *)
BOOL1 = getbit ( dw , 0); (*  从第 0 位读开关量 1 *)
BOOL2 = getbit ( dw, 1); (*  从第 1 位读开关量 2 *)
SFLOAT1 = getsfloat ( dw,1); (*  从高 16 位读模拟量 1 *)
```

- 冷热启动及下载组态标志

BOOL g_bHotStartup

热启动标志，热启动时为 ON，需要编程清 0；

BOOL g_bColdStartup

冷启动标志，冷启动时为 ON，需要编程清 0；

g_bDownUsrPrgFlag

下载用户程序标志，下载了用户程序之后为 ON，需要编程清 0；

g_bDownCfgFlag

下载组态标志，下载了组态（包括硬件组态、用户程序组态）后为 ON，需要编程清 0。

- 全局热冗余数组

LONG g_sfcbaks[32]

全局热冗余数组。

2) 质量说明

质量码的使用方法。

表 3-1 模入位号数据成员 Flag（位号质量码）定义（支持型号为 FW243X、XP243X、FW247 的控制器）

质量码的位		说明	位格式	相关数值
D2=0	D0	上限报警	1——报警 0——不报警	1
	D1	下限报警	1——报警 0——不报警	2
D2=1	D0	上上限报警	1——报警 0——不报警	5
	D1	下下限报警	1——报警 0——不报警	6
D3				
D4				
D5				
D6				
D7				
D8		信号可疑	1——信号可疑 0——信号不可疑	256
D9				512
D10		手动输入	1——手动输入 0——自动输入	1024
D11		信号故障	1——信号故障 0——信号无故障	2048
D12		速度报警	1——速度报警 0——速度不报警	4096
D13				
D14				
D15				

表 3-2 模入位号数据成员 Flag（位号质量码）定义（支持型号为 SP243X、FW243、 SP243、XP243、SP244、FW244 等的控制器）

质量码的位		说明	位格式	相关数值
D2=0	D0	上限报警	1——报警	1
			0——不报警	
	D1	下限报警	1——报警	2
			0——不报警	
D2=1	D0	上上限报警	1——报警	5
			0——不报警	
	D1	下下限报警	1——报警	6
			0——不报警	
D3		偏差报警	1——报警	8
			0——不报警	
D4				
D5				
D6				
D7				
D8		模块故障	1——故障，伪信号	256
			0——无故障	
D9		速度报警	1——报警	512
			0——不报警	
D10		信号手动 / 自动输入	1-自动（测量信号）	1024
			0-手动（设定信号）	
D11		超量程报警*	1——报警	2048
			0——不报警	
D12		时限报警	1——报警	4096
			0——不报警	
D13				
D14				
D15				

★:若测量点与变送器之间的线路短路或断路，导致信号超量程，这又可称为开路报警。如输入为III型电流信号， 量程为（ 4～20）mA，若输入线路短路，则为 0mA，那通过质量码的 D11 位就可以判断出来。

例如，如果你需要判断某个模入信号是否处于上上限报警，首先将这个模入信号的成员 Flag 与成员 HH 的相关数值 5（详见上表）进行“ 与 ”操作，即屏蔽无关位，只留下 FLAG 中的相关位（D2D0)；然后将“ 与 ”操作完成后的数值与该相关数值 5 比较判断是否相等。如果相等，则表明此信号处于上上限报警状态；否则此信号没有处于上上限报警状态。原则上上限报警，上限一定报警；同理下下限报警，下限也一定报警。

4 图形编程软件使用指南

图形编程软件的使用是十分方便的。以下将分别介绍软件的运行环境、编辑界面、菜单项功能、程序类型说明、变量类型说明、工程设计、文件结构以及在线调试等方面的使用。

- 运行环境
- 编辑界面
- 菜单项功能
- 程序类型说明
- 变量类型说明
- 工程设计
- 文件结构
- 在线调试

4.1 图形编程的运行环境

图形编程要求运行在 P Ⅱ 300 、64M 内存以上的 PC 机上。要求 Win9x 或 WinNT 或 Windows2000 上。如果需要在线调试，则需与 SUPCON DCS 的主控卡相连。

4.2 编程界面介绍

图形编程软件的编程界面友好，在一个图形化的界面上提供了大量模块化的功能块，用户通过对模块、触点、线圈等的选取、放置、移动、连接等操作，即可轻松完成编程工作，较一般的高级语言编程要更简单、更方便、更可视化。

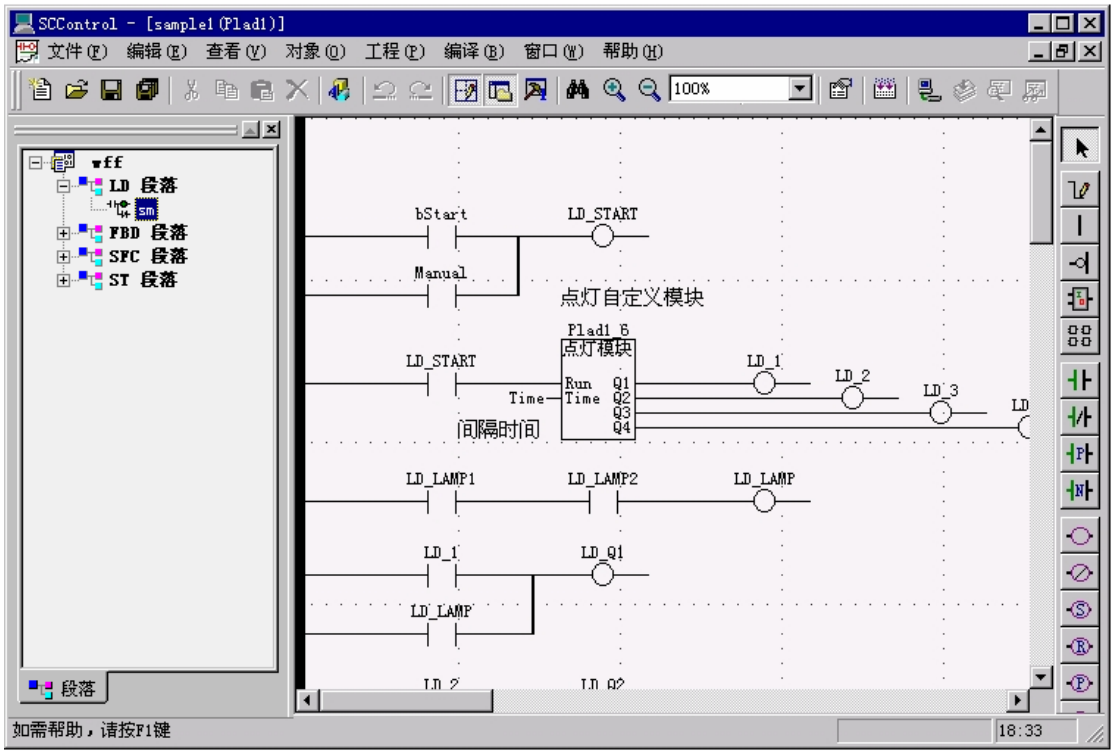

图 4-1 编辑界面

上图是软件的一个编程界面，包括了菜单栏、工具栏、状态栏、工程栏、信息栏以及程序的编辑区。

- 菜单栏

图 4-2 菜单栏

菜单栏集成了软件的大多数功能，有效利用菜单栏可完成一般程序的编辑工作。

- 工具栏

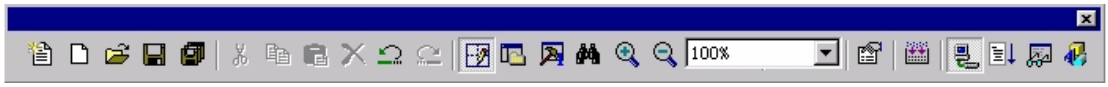

图 4-3 工具栏

工具栏将菜单栏中许多重要的功能图标化，简化了操作。工具栏中的图标和菜单项相连。它们的出现依赖于菜单项的状态：如果菜单项为灰色，对应的工具栏图标也为灰色；如果菜单项被激活，则对应的工具栏图标也处于激活状态。

- 梯形图(LD)组态元素

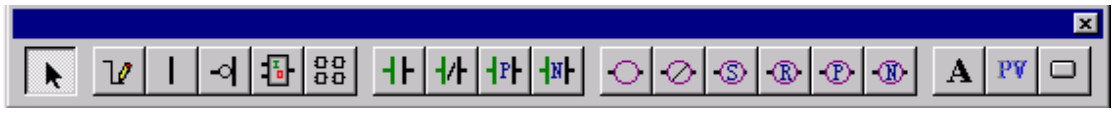

图 4-4 梯形图组态元素

- 顺控图(SFC)组态元素

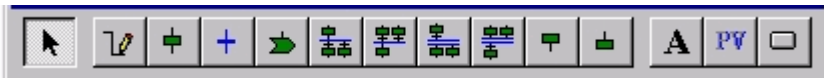

图 4-5 SFC 组态元素

- 功能块图(FBD)组态元素

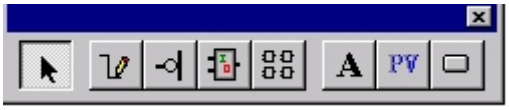

图 4-6 FBD 组态元素

- 工程栏

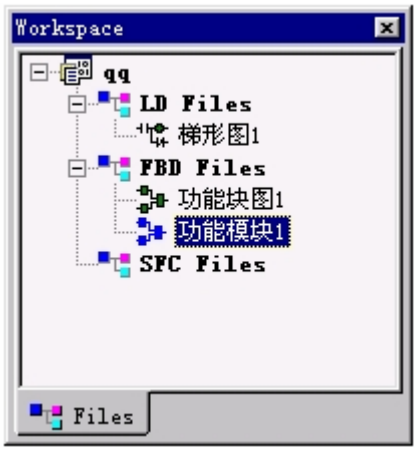

图 4-7 即为一个工程栏。

在 Workspace 中列出了工程中的所有程序段，图中的 LD Files 指的是梯形图类型的程序段；FBD files 指的是功能块图类型的程序段；SFC Files 指的是顺控图类型的程序段。

图 4-7 工程栏

- 编程区

下图所示的程序编辑区是用来进行程序与模块设计、编辑的。窗口的背景是逻辑坐标网格。

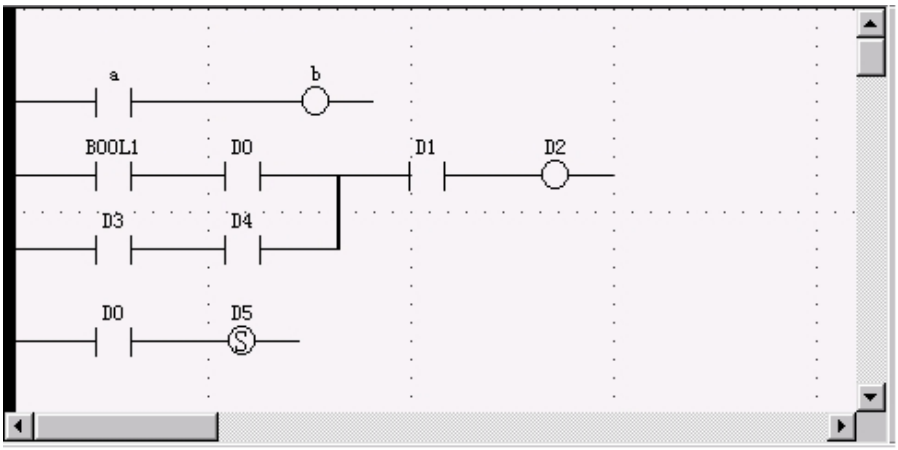

图 4-8 编程区

- 信息栏

信息栏用于编译及工程打开成功与否的信息。

成功打开一个工程或创建一个新工程后，编译信息输出栏输出“ 打开工程成功 ”信息（如下图）；否则输出“ 打开文件失败 ”信息；

图 4-9 信息栏

成功编译好一个程序段或工程后，编译信息输出栏输出“ 编译成功 ”信息（如下图）；

图 4-10 编译成功信息

如果程序段或工程没有完成或存在错误，则编译不成功，编译信息输出栏输出编译不

成功的原因（如下图）。用户必须修改程序，反复调试，直到编译信息输出栏输出“编译成功”信息方能进行下一步的操作。

图 4-11 编译错误信息

此外，信息栏还显示编译过程中的错误信息，鼠标双击错误信息则自动跳转到相应的出错的作图元素上，方便改错。

- 状态栏

图 4-12 状态栏

状态栏位于界面的最下方，显示编辑状态，如提示用户在遇到疑难时，可按 F1 键，获取帮助信息等。

4.3 菜单功能项介绍

菜单项功能简介

菜单项集成了几乎所有图形编程软件中提供的功能，运用好菜单项，就可以完成整个工程的编辑。图形编程软件的菜单项分为“ 文件 ”、“ 编辑 ”等数个类别，以下将分别加以介绍：

文件	编辑
查看	对象
工程	编译
窗口	帮助

1. 文件菜单功能介绍

在文件菜单条上，设置了很多功能。利用、或者说有效地运用这些功能，将大大有助于图形编程软件的编辑。

新建工程

用于建立新的图形编程软件的工程文件。相关内容可参见 FBD 语言编程中的“ 创建一个新工程 ”，或 LD 语言编程中的“ 创建一个新工程 ”。在图形编程软件中，用户可在同一编辑环境中同时新建多个工程。

打开工程

用于打开以往所建立、保存的图形编程工程文件。当对已存在的工程进行编辑操作时，选取文件/**打开工程**命令或者直接用鼠标左键单击工具栏中的打开按钮，屏幕上就会出现**打开工程**对话框。在图形编程软件中，用户可在同一编辑环境中同时打开多个工程。

关闭工程

用于关闭一个正在编辑，或已编辑好的图形编程工程文件。

保存工程

当完成整个工程（包括程序段等）的编辑后，可单击工具栏上的保存图标或选取**文件**菜单中的**保存工程**菜单项，如果所要保存的文件是一个尚未赋予文件名的新文件，屏幕上就会出现保存为对话框，让用户键入文件名。

如果是对一个已存在的旧工程文件进行编辑后保存该文件，则不会出现对话框，而是自动直接以原工程文件名保存，编辑后的工程内容将覆盖原有工程文件内容。

工程另存为

用于将原工程文件保存为其他文件类型的文件，或保存为同一文件类型、其他文件名的文件。

新建程序段

用于建立一个新的程序段。相关内容可参见 LD 语言编程中的“ 创建一个新程序段 ”，或 FBD 语言编程中的“ 创建一个新程序段 ” 的相关内容。

打开程序段

用于打开一个已经建立并存在的程序段。

保存程序段

用于将一个编辑好或正在编辑的程序段进行存盘操作。

打印

在保存好工程或程序段文件后，要打印所建立的文件内容时，可单击工具栏中的“ 打印 ” 图标按钮或选择文件/打印命令，执行打印时，会出现如图“ 打印 ” 对话框：

图 4-13 打印

- 打印机

画面所显示的打印机名称为当初安装 Windows 95 时所指定的机型，可在下拉式列表框中改变或添加。

- 打印范围

用于选择设定要打印的部分。可选择：

全部： 打印整份文件。

页数： 可在文字框内指定要打印的页次。

选定范围： 打印文件中用户所选定的部分。

- 打印到文件

此复选框是将要打印的文件内容存入一个文件中，而不是由打印机输出。当要把该文件拿到另外一台只有 DOS 系统的计算机上打印时，用户可在 DOS 提示符下键入命令：

TYPE 文件名>PRN

将其直接打印出来，不需进入 Windows 环境。

该项功能与硬件有关，也就是说，用于打印的文件内部不仅仅包含了打印的内容，而且还包含有打印机的配置信息，例如打印机型号、纸张类型等等。所以在使用该功能时，一定要保证相应的两台计算机的打印机、打印设置等等都完全一致。

- 份数

可选定要打印的份数。

- 属性

单击“ 属性 ”按钮时，会出现如下图所示对话框：

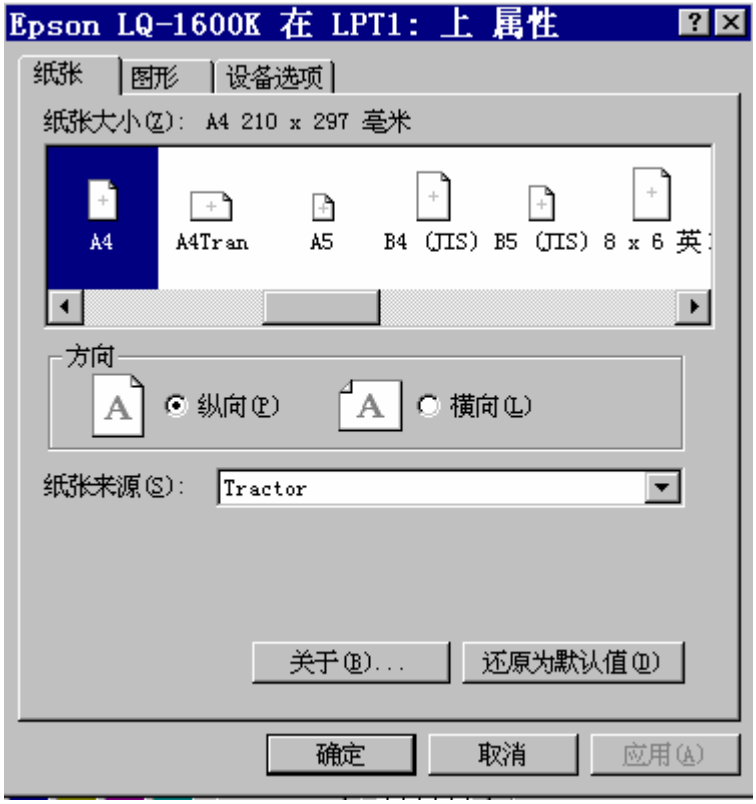

图 4-14 属性

其中包括纸张、图形及设备选项三部分，用户可以根据需要分别设定。

在正式打印文件之前，SUPCON 流程图制作软件提供了预先浏览文件打印状况的功能：**文件/打印预览**，可避免因打印状况不当，导致不必要的浪费。

打印预览

用于在正式打印之前，预先观察实际打印后的效果。详细情况可参考打印功能。

打印设置

用于在打印之前，根据用户自己的实际需要，在“打印设置”对话框中，选取所需的打印设置条件，以获取尽可能好的打印效果。详细情况可参考打印功能。

退出

退出图形编程软件的工程编辑状态，此时将关闭所有正在编辑的程序或模块文件。

2. 编辑菜单功能介绍

撤消

该功能用于取消上一次的操作。图形编程软件提供了多次撤消功能。

恢复

用于执行了撤消操作后，使编辑状态恢复到撤消操作前的状态。

剪切

用于将程序编辑区中用户指定区域的内容复制到**剪贴板**内，同时删除该区域里的内容。

所谓**剪贴板**，是指操作系统内部的一块内存。使用剪贴板，可以在几个模块文件之间传递功能块、线圈、触点、文字或它们之间的任意组合，还可以在几个程序之间进行同样的传递。

该功能的操作过程如下：

1) 选取要剪切的内容

用鼠标选出需要剪切的内容。

2) 剪切

鼠标左键单击**剪切**按钮（或者单击**编辑/剪切**命令），所选定区域中的图像就会复制到剪贴板上去，同时删掉了该区域中的内容。

此时，就可以根据需要，将“剪贴板”中的内容复制到其他的图形编程文件中去。

复制

用于将程序编辑区中选定区域复制到剪贴板内，除了不删除选定区域的内容外，其余功能和用法与“剪切”功能相同，具体使用请参见“剪切功能”的详细说明。

粘贴

用于将剪贴板中的最新内容（即最近一次剪切或复制的内容）复制到指定编辑区内。用鼠标左键单击工具栏的**粘贴**图标，剪贴板中的最新内容将被粘贴在画面的左上角。可以看到粘贴图标有被选中标志，此时，可以按住鼠标左键将其拖到需要的位置（选取工具必须在粘贴内容区域内），此为推荐操作；如果在移动粘贴内容之前进行了其它操作（诸如选取），则粘贴图标的被选中标志消失，若再想移动粘贴内容，必须重新选中，有可能将其下面的内容一起选中，带来不必要的麻烦。

该功能必须在执行**剪切**或**复制**之后使用。在未执行**剪切**或**拷贝**之前，由于剪贴板中无内容，所以图标呈“灰色”，表示该功能无效。

该功能可连续执行多次，在两次操作之间，用户可以使用左右滚动条和上下滚动条调整程序编辑区在实际程序编辑区中的位置。

删除

用于删除一个选定的对象。

查找

用于进行用户所需信息（如模块、文字、变量名等）的检索。用户可自行填写，也可按照不同类型、数据源通过浏览进行检索。可参见“查找与替换”。

替换

用于将当前工程或段落中的内容用用户需要的其他内容替代。参见“查找与替换”。

全选

用于将编辑区中的所有功能块、触点、连接线、线圈、文字等选中。

3. 对象功能介绍

图形编程软件提供了丰富的对象功能，可方便用户从菜单上选取功能块、各种触点、线圈、文本等对象，以下只是对象功能的一部分，其他内容可参看“ 跳转功能 ”等的相关介绍。

选择

对象操作时的缺省操作，用于选取编辑区内的模块，连接线，线圈等等对象。

连接线

用于模块与模块、模块与线圈、模块与母线、模块与触点以及其他对象之间连接的水平线条。

垂直连接线

用于连接来自几个输入与一个输出的垂直线条。输入可以来自模块、母线、线圈等对象，输入可以为多个；输出可连向模块、线圈等对象，只能为一个。垂直连接线的功能相当于一个“ 或 ”模块。

最近选择的功能块

用于在编辑区内放置一个最近选取过的功能模块。

取反

用于模块的输入管脚对输入信号取反，及用于模块的输出管脚将模块的输出值取反输出。

注释文本

在程序编辑区中起注释作用的文字说明。

功能块选择

用于选取各种不同类型（例如算术运算、选择运算模块等）、不同作用（例如“ 加 ”模块、“ 逻辑右移 ”模块等）以及不同数据类型（例如 BOOL、WORD 型“ 或 ”模块，FLOAT、INT、LONG、SFLOAT、UINT、ULONG 型的“ EQ ”模块等）的功能模块。用户可在如下图所示的对话框中任意选取：

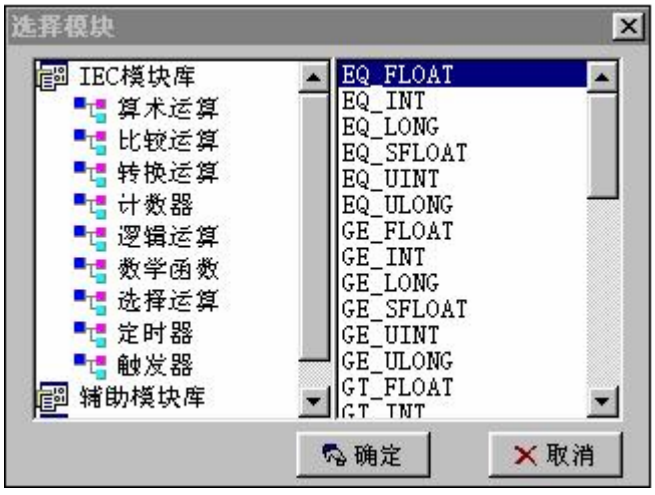

图 4-15 功能块选择

变量定义

用于声明变量，变量必须在声明之后使用。

4. 工程功能介绍

控制站地址

用于指定工程对应的控制站地址。

数据类型编辑器

用于编辑自定义数据类型。

变量编辑器

用于定义全局变量。

段落管理

用于管理工程中的程序段，可以打开、删除、导入、导出程序段。

任务管理

该功能可用于设置工程中各程序的运行周期。在一个工程中往往有多个程序，它们的重要性或在工程中运行时需调用的周期大小也不同，为在工程的分时处理中，使各个程序得到合理的执行时间，避免各程序执行时的冲突，优化处理过程，用户可运用该功能，根据实际需要自行安排各个程序的运行周期。

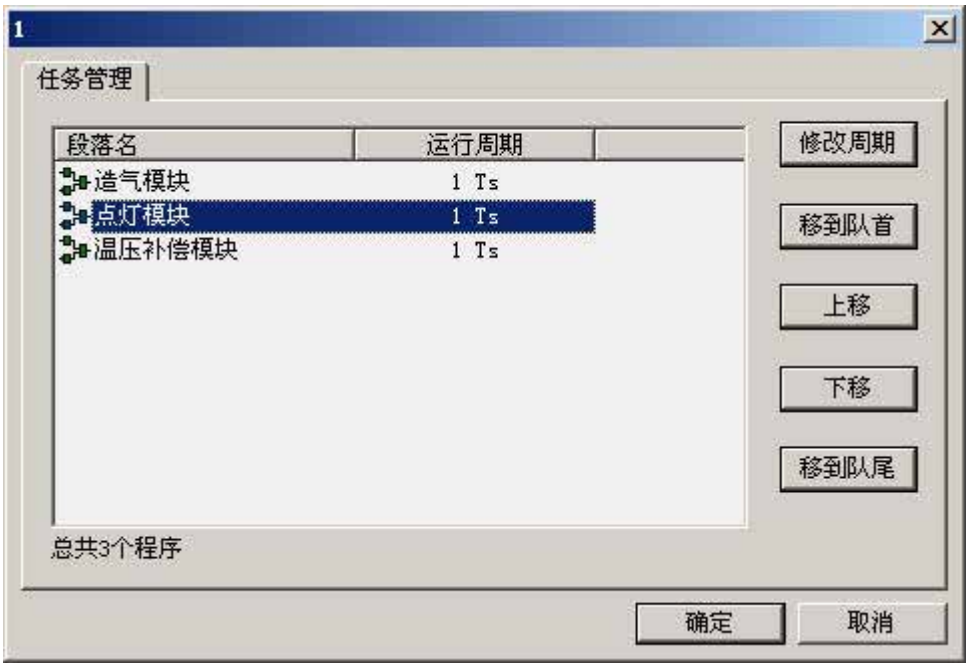

图 4-16 任务管理

上图为设置运行周期的对话框，用户可点击对话框右侧的“ 修改周期 ”按钮，自行修改运行周期。

图形编程以系统组态软件中设置的控制周期为 1Ts，即：如果在系统组态软件的组态过程中设置了控制周期为 0.1s，则 1Ts = 0.1s。

用户还可通过操作“ 移到队首 ”、“ 上移 ”、“ 下移 ”、“ 移到队尾 ”等操作设置同一运行周期各程序运行的优先级，即排在队列靠前的同一运行周期程序比排在队列靠后的程序优先执行。不同运行周期的程序之间的优先级无法比较。

5. 窗口

层叠

将所有已打开的窗口以层叠的方式排列，原本打开的窗口同时出现在当前编辑区中。如

下图所示：

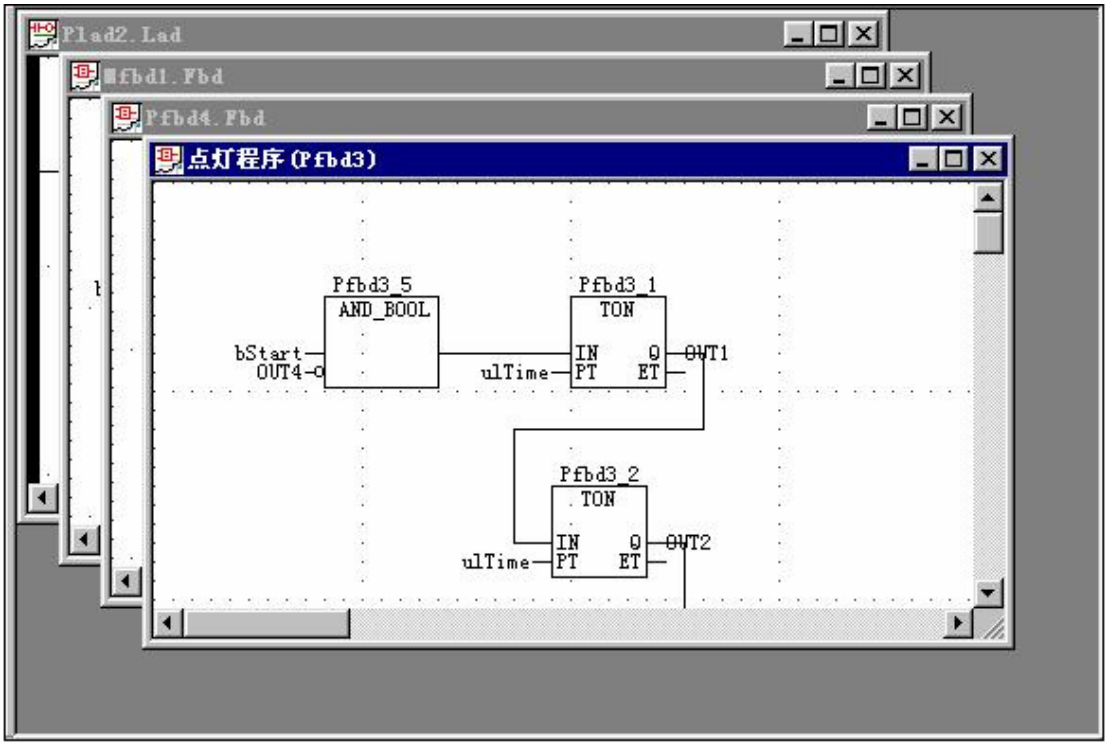

图 4-17 层叠

平铺

将所有已打开的窗口以平铺的方式排列显示于整个编辑区中。如下图所示：

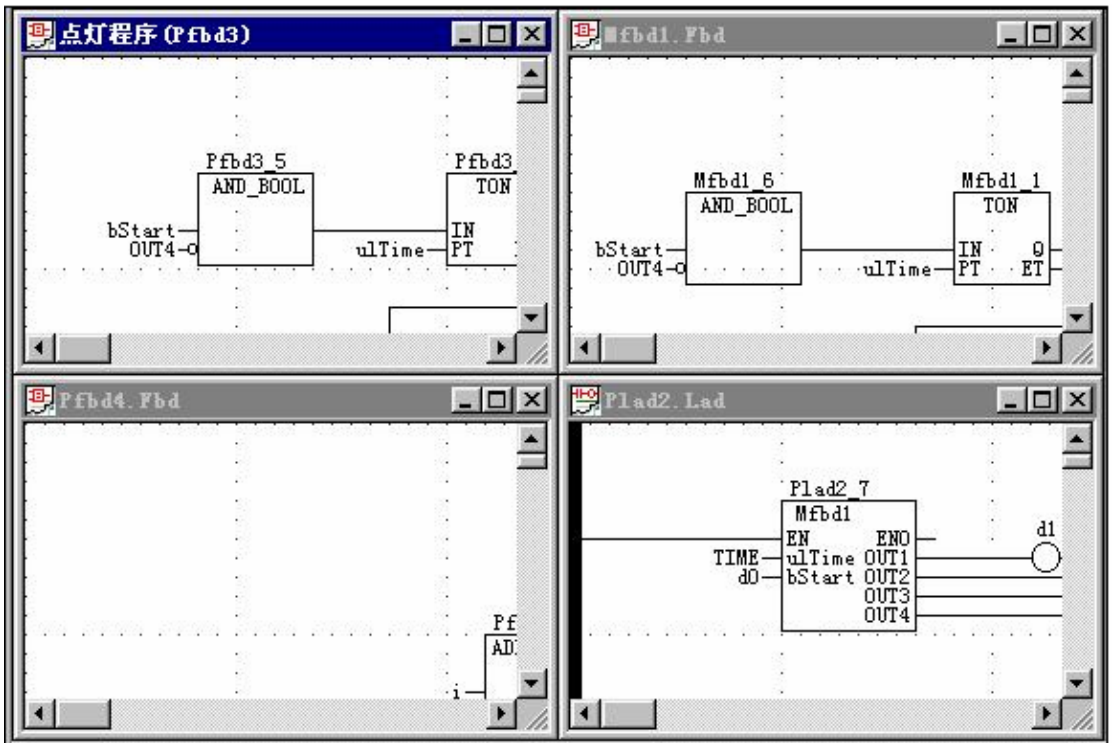

图 4-18 平铺

关闭所有窗口

将所有已经打开的窗口全部关闭。

6. 帮助菜单功能介绍

帮助主题

提供在线帮助。在使用图形编程软件进行 LD、FBD 语言编程或其它操作的过程中产生疑问或遇到困难，需要帮助时，可以单击该菜单，将得到软件大量详实的相关帮助信息。

关于图形编程

提供软件的版权、版本号等一些相关信息。

7. 编译功能介绍

编译工程

用于将由程序、模块等组成的工程或单独的程序进行编译，以产生机器可识别的目标代码。

优化编译

在编译中重新分配控制站内存。

联机

用于工程与下位机的连接。

调试

用于观察工程执行时的效果。

变量调试窗口

用于在工程运行时，通过手工调节程序变量的取值（见下图），来调试工程或程序的正确性。

图 4-19 变量调试窗口

4.4 变量类型说明

图形编程软件支持多种变量类型，编程软件采用的是强类型检查的编译方式，不同类型的相互之间不存在隐式类型转换。系统支持的类型如表 4-1 所示。

表 4-1 图形编程软件支持的系统类型

类型	描述
BOOL	1 字节类型
BYTE	1 字节类型

INT	2 字节类型
UINT	2 字节类型
WORD	2 字节类型
SFLOAT	2 字节类型
LONG	4 字节类型
ULONG	4 字节类型
DWORD	4 字节类型
FLOAT	4 字节类型
structAccum	8 字节类型
structAI	AI 类型
structDI	DI 类型
structPAT	PAT 类型
structCSC	CSC 类型
structBSC	BSC 类型

另外图形编程软件还支持自定义数组和结构类型。其中自定义结构支持任意深度的嵌套。

4.5 工程设计

设计的主要步骤：

- **步骤 1** 启动图形编程软件
- 从 Windows 启动组态软件 Sckey，打开目标文件后，从自定义图形组态启动图形编程软件。
- 或直接从 Windows 启动图形编程软件。

- **步骤 2** 新建或打开工程
- 从 **文件** 菜单选择 **新建工程** 。在新建对话框中键入工程名。
- 从 **文件** 菜单选择 **打开工程** 。在打开对话框中选择需编辑的工程文件。
- 从 **文件** 菜单选择 **最近打开文件**。

- **步骤 3** 编程
- 从 **文件** 菜单选择 **新建段落** 。在对话框中键入段落名，选择段落属性。
- 使用工具条或 **对象** 菜单命令生成程序。
- 在 **工程** 菜单中选择 **设置任务** 命令设定程序的执行周期和执行次序。
- 在 **工程** 菜单中选择 **设置相关控制站地址** 设定相关联的控制站地址。

- **步骤 4** 保存编译
- 从 **文件** 菜单选择 **保存工程** 或 **工程另存为**。
- 从 **编译** 菜单选择 **编译工程** 或 **全部重新编译**。

- **步骤 5** 下载测试

- 从 **编译** 菜单选择 **连接** 再选择 **下载。**
- 从 **编译** 菜单选择 **调试**。
- **步骤 6** 优化和断开
- 在 程序修改和下载多次以后，应该进行内存分配优化，选择 **编译** 菜单的 **全部重新编译** 将重新分配内存资源优化编译。
- 断开连接只需再次选择 **连接** 命令。

- **步骤 7** 文档工作
- 在工程调试完毕，应该生成一套完整的文件。

4.6 文件结构

编译通过之后观察生成的文件目录结构如下：

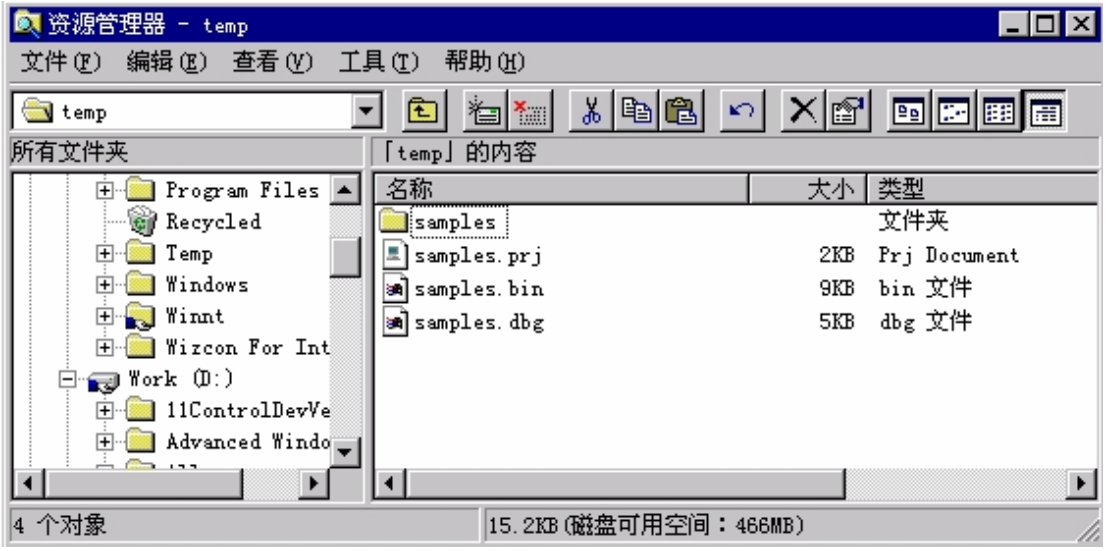

图 4-20 文件目录结构

如上例，samples.prj 为实际的工程文件。

Samples.dbg 包含了工程对控制站的内存映象，每个变量在控制站的地址保存在此文件中。删除此文件将导致工程对控制站内存映象重新建立。

Samples.bin 为生成的工程代码。此代码下载到控制站就执行相应的控制方案。

Samples 子目录包含了工程的段落文件。

4.7 在线调试

编程器与控制站连接后，可以下载程序，也可以进入在线调试（即联机调试）。进入联机调试时先校验上位机工程与实际控制站程序，不一致则报警。编程器中的当前程序与控制站实际连接，程序中的开关量和开关链路将根据实际数据显示通断状态。在程序中的调试文本（PV）将显示其实际值。用户可以通过 PV 设置控制站的数据。

- **连接**

只有当连接处于工作时，才能使用下载和调试命令。使用 **编译** 菜单上的 **连接** 命令或单击工具条上 **连接** 按钮就进入连接状态。这时，**下载** 命令和 **调试** 命令从不可用状态变

为可用状态。

- **调试**

按 **调试** 按钮进入调试状态。在调试状态下，可以以动画形式观察布尔变量在变化，也能从调试文本操纵模拟数据。如图：

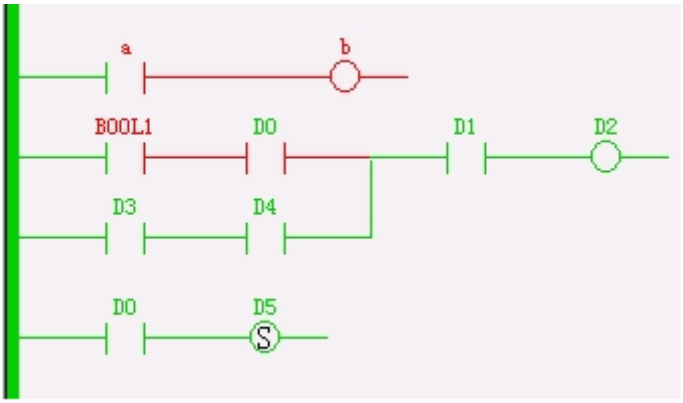

图 4-21 调试状态

变量和链路的布尔数值为 ON 时 动画显示为绿色，OFF 显示为红色。

按 **变量调试窗口** 按钮可进入调试变量对话框，在对话框内用户可以键入需调试变量的值，变量值将实时刷新。

图 4-22 变量调试窗口

变量调试窗口关闭后，修改的变量的值仍然有效。

- **运行时间**

在调试状态下，打开运行时间窗口，可以观察每个程序在每次执行花费的时间 。

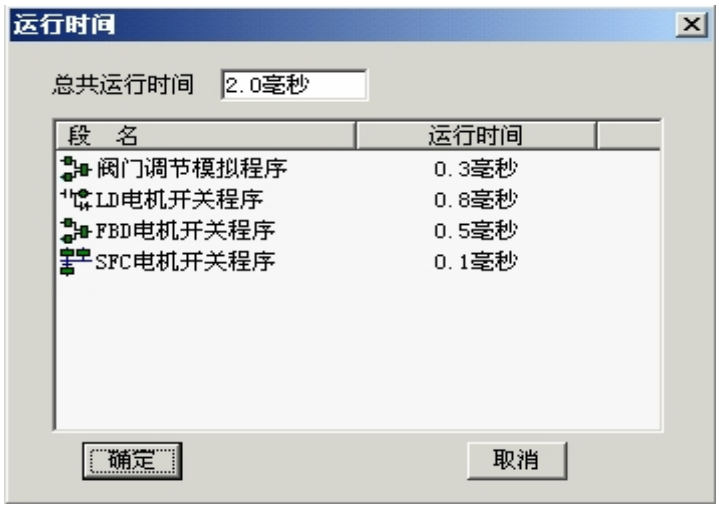

图 4-23 运行时间

4.8 密码保护

- 设置密码
- 删除密码
- 使用注意

1. 设置密码

密码可以有字符、数字或者下划线组成；左键选中已打开的段落，点击右键出现如下图所示的密码设置菜单，选择设置密码菜单，即可进行密码设置。

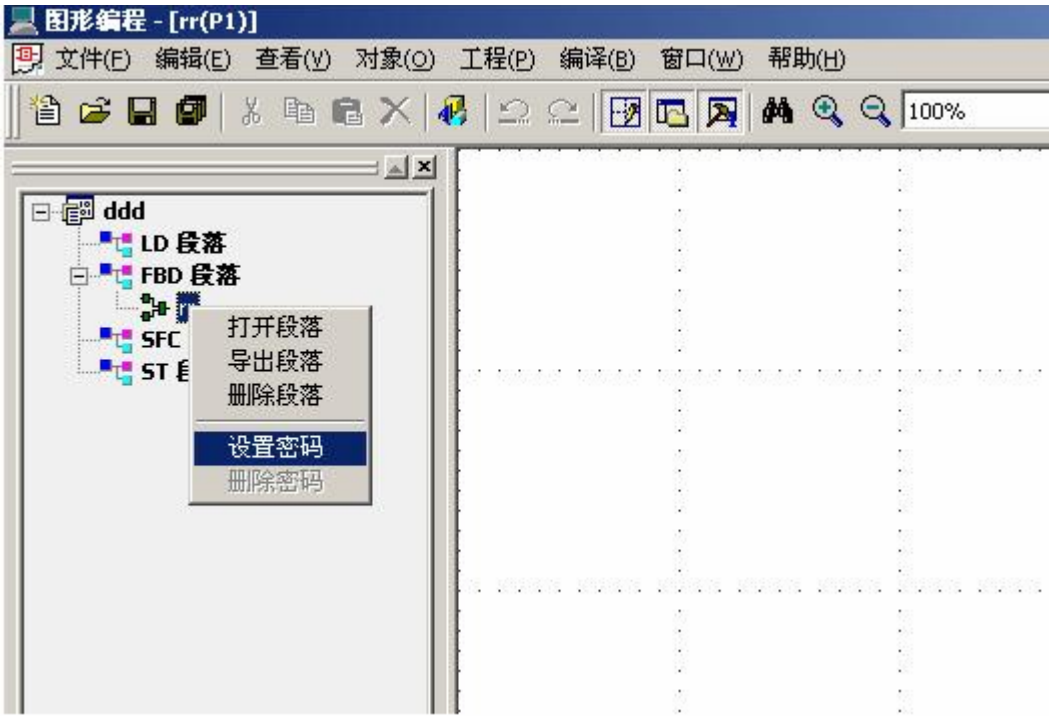

图 4-24 密码设置

段落必须在其编辑界面打开后，才能设置密码或删除密码的。

有密码保护段落的编辑界面一旦关闭，段落就处于密码保护状态，用户不能进行设置密码或删除密码操作。只有当编辑界面打开后，才可以进行设置密码和删除密码的操作。

2. 删除密码

段落在其编辑界面打开后，如果有密码保护就可以进行删除密码操作。删除密码时，将有确认对话框出现，如下图密码确认对话框。确定删除密码后，密码从工程中删除。

图 4-25 删除密码

3. 使用注意

工程文件的版本

工程文件中的某一段落设置密码后，以前的 SCControl 版本无法打开有密码保护段落的工程文件。

段落导入和导出的

当前导出的段落，如果有密码保护，则导出的段落始终存在密码保护。在导入的时候，其编辑界面默认处于关闭状态，密码跟导出前段落的一样。

段落的更名操作

如果当前有密码保护的段落的编辑界面处于打开状态，则修改段落名后，相应修改后的段落有密码保护,并且处于打开状态 如果当前有密码保护的段落的编辑界面处于关闭状态，则修改段落名后，相应修改后的段落有密码保护，并且处于关闭状态

输出栏中的操作

当段落处于密码保护状态时，且当前段落的编辑界面处于关闭状态，则当双击输出栏中的对象处于此段落时，将出现密码输入框。正确输入密码一次后，则段落的编辑界面处于打开状态，以后双击输出栏中的对象处于此段落时，就不出现密码输入框。

工程文件的操作

当前有密码保护段落的工程文件，如果对该工程文件选择另存为操作，则相应的段落还是处于密码保护中。

5 图形编程模块库

图形编程包含以下几个模块库：

IEC 模块库、辅助模块库、自定义模块库、附加库等。

具体模块使用说明请参见《图形编程模块使用手册》

6资料版本说明

表 6-1 版本升级更改一览表

资料版本号	更改说明
图形化编程使用手册（V1.1）	使用软件版本说明：AdvanTrol-Pro V2.50
图形化编程使用手册（V2.0）	使用软件版本说明：AdvanTrol-Pro V2.65
图形化编程使用手册（V2.1）	使用软件版本说明：AdvanTrol-Pro V2.65 修改 3.6.7 节中“ 质量码的使用方法 ”
图形化编程使用手册（V2.2）	使用软件版本说明：AdvanTrol-Pro V2.70