LOGIC LAB

用户手册

Revision 4.2 - February 2015

AXEL

SOFTWARE SOLUTIONS FOR THE INDUSTRIAL AUTOMATION

LogicLab User Manual

Revision 4.2 - 2015-02-10

Published by Axel S.r.l.

Via del Cannino, 3

21020 Crosio della Valle (VA)

目录

1. 介绍

1.1 本文档中使用约定

文本类型	描述
命令，关键	命令或键盘快捷方式键的名称。
码	源代码的文本。
[上下文菜单]	工具栏图标和上下文菜单的声音。
[上下文菜单]	上下文菜单的声音，没有任何图标。
菜单>项目	对于菜单项的层次结构，使用了"**>**"符号。录音文件>打开项目相当于"文件菜单下打开项目项目"。
菜单>项目	与上述相同，包括在工具栏上显示的图标。
（见段落）（见第四章）	链接到本指南中的相关主题。
术语	重要的名词或概念。

2. 概述

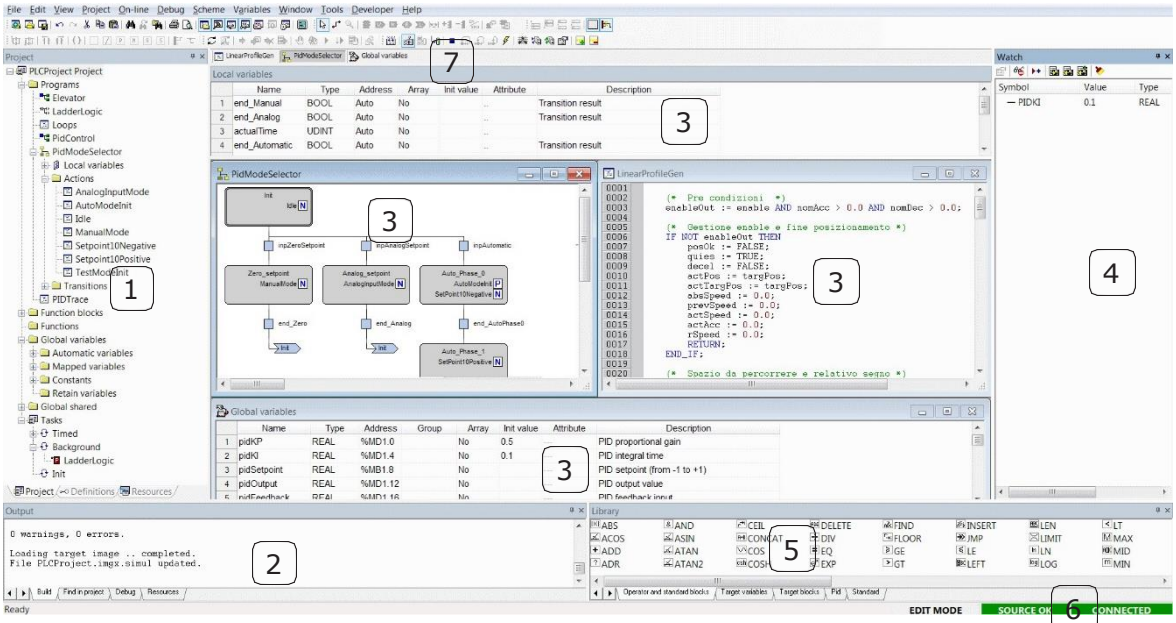
LogicLab是一种IEC61131-3集成开发环境支持标准中定义的五种编程语言。

为了支持用户开发应用程序所涉及的所有活动，LogicLab功能包括：

- 文本源代码编辑器的指令列表（简称为IL）和结构化文本（简称为ST）编程语言（见章节6）；
- 图形源代码编辑器的梯形图（简称为LD），功能块图（简称为FBD）和顺序流程图（简称为SFC）编程语言（见章节6）；
- 编译器：LogicLab根据IEC标准将IEC61131-3应用程序直接编译成对应平台的机器代码，避免了对运行时解释执行的需要，从而尽可能快地执行程序（见章节7）；
- 通信系统：将应用程序下载到目标环境（见章节8）；
- 多种调试工具：易于使用的监视窗口以及强大的示波器，可以直接在目标环境中采样快速变化的数据，确保数据准确可靠且易于分析和导出（见章节9）。

2.1 工作台

下图显示了LogicLab工作区的视图，其中包括许多常用组件。

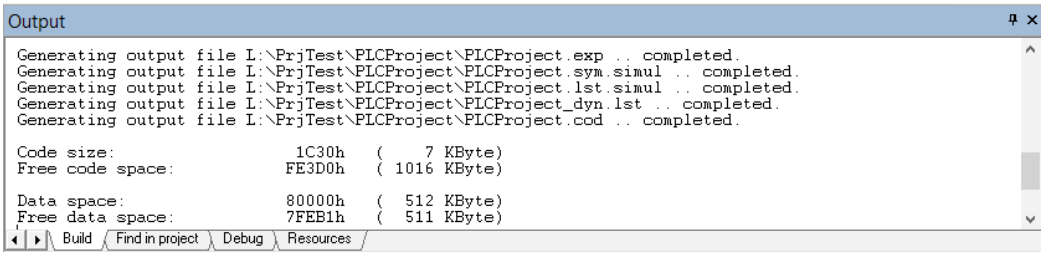


1. 工作区窗口 2. 输出窗口 3. 代码编辑区 4. 监视窗口 5. 库窗口 6. 状态栏 7. 文件标签栏

下面的段落将解释说明这些组件。

2.1.1 输出窗口

输出窗口是LogicLab打印其输出消息的地方，该窗口包含四个选项卡：Build，Find in project，Debug，Resource。



Build

Build面板显示下列活动的输出：

- 打开项目;
- 编译项目;
- 下载代码到目标。

Find in Project

此面板显示在项目活动中查找的结果。

Debug

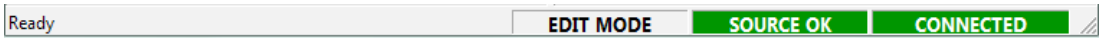
Debug面板显示高级调试活动（例如：断点调试）的信息。根据所使用目标设备上的接口，LogicLab可以通过这个输出窗口上打印每一个PLC运行时错误（例如：除零错误），由此定位错误发生的确切位置。

Resource

资源面板显示LogicLab正在连接的目标设备的相关消息。

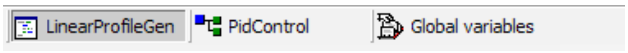
2.1.2 状态栏

状态栏的左边框用于显示应用程序的状态，有边框的动画控件用于显示其通信的状态。



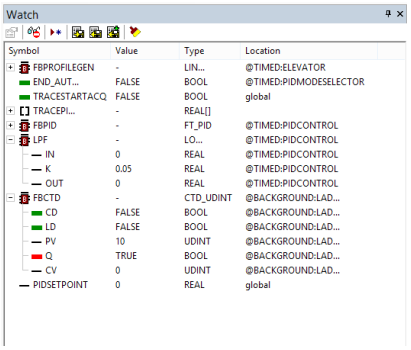
2.1.3 文件栏

文件标签栏列出了所有当前在LogicLab打开，并正在编辑的文档。



2.1.4 监视窗口

监视窗口是LogicLab提供了许多调试工具之一。除此之外，值得一提是示波器调试工具，具有触发器和现场调试模式（见9.2章节）。

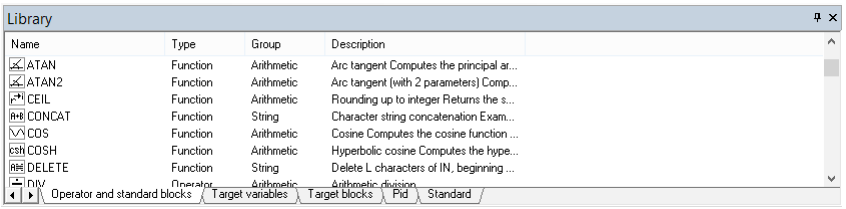


2.1.5 库窗口

库窗口中包含了一组不同类别的窗口，下面将说明这些不同的类别的窗口。
您可以通过单击鼠标右键来选择显示模式。
在*[view List]*模式中，每个元素由其名称和图标表示。 而是在*[View detail]*模式中显示一个表格，其中每一行都与其中一个元素相对应，并包含库的类型（运算符/函数）和每个元素的描述。
如果右键单击此面板的其中一个元素，然后单击对话框中的*[Object properties]*，则会出现一个窗口，其中包含有关所选元素的更多详细信息（输入和输出支持的类型，输入和输出引脚的名称等）。
在*[View Folder]*模式下，每个元素都被分组到它所属的文件夹中。
这些元素按照一定的逻辑被分组到不同的文件夹中，以方便使用和查找。

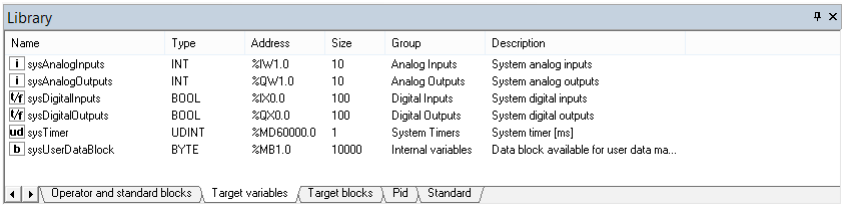
2.1.5.1 运算符与标准功能块

此面板列出了基本的语言元素，例如IEC 61131-3标准定义的运算符和函数。



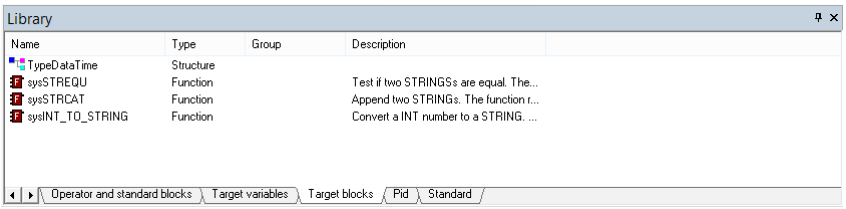
2.1.5.2 系统变量

此面板列出了所有的系统变量，也被称为目标变量，它们是固件和PLC应用程序代码之间的接口。



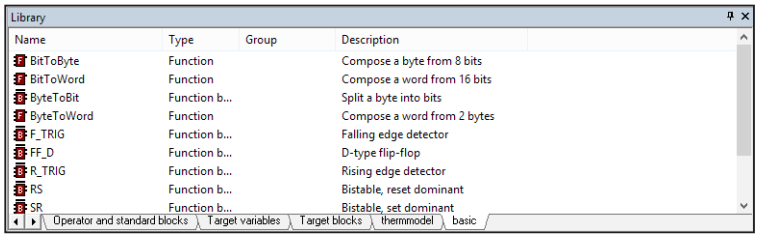
2.1.5.3 用户库

此面板列出了特定目标设备上可用的所有的系统功能和功能块。



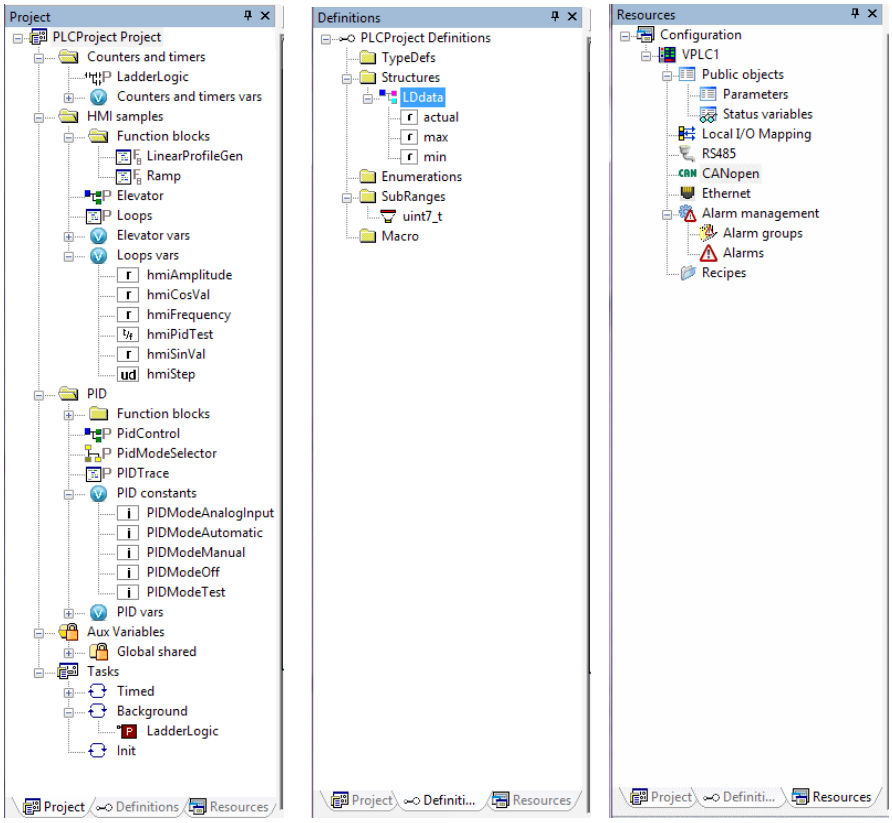
2.1.5.4 增加库面板

在前述段落中描述的面板通常是在“库”窗口中默认自带的。同时，当前LogicLab工程中包含的每个库都可以添加到此窗口中。例如下面图片来自于一个LogicLab工程，其中包含两个库：basic.pll和thermmodel.pll（详细内容参见4.7章节）。



2.1.6 工作区窗口

工作区窗口包括三个不同的面板，如下面图片所示。



2.1.6.1 工程

“工程”面板中包含了一组文件夹：

- 任务：这个项目列出了系统任务以及分配给每个任务的程序（参见5.3章节）

2.1.6.2 定义

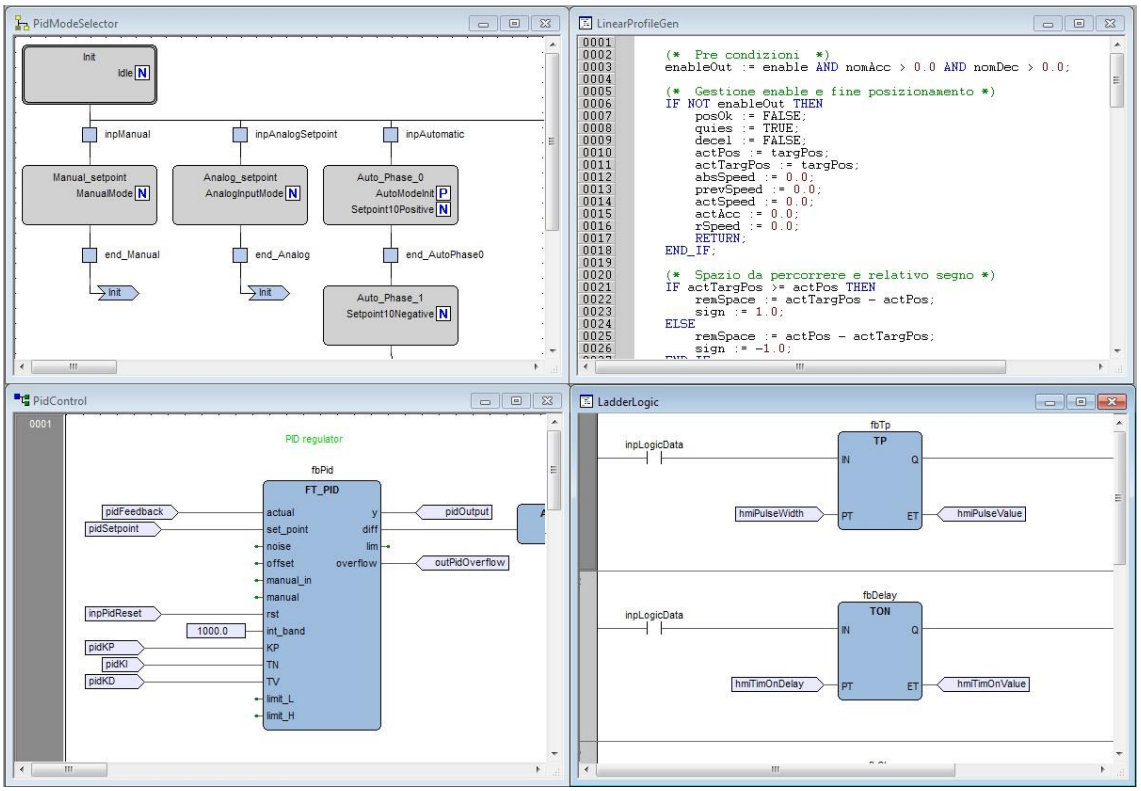
该“定义”面板包含了所有用户定义的数据类型，比如结构体或枚举类型。

2.1.6.3 资源

“资源”面板的内容取决于LogicLab与之接口的目标设备：它可能包括配置元素，模式，向导等。

2.1.7 源代码编辑区

LogicLab编程环境中包含一组编辑器，用于管理、编辑和打印以IEC 61131-3标准定义的5种编程语言中的任何一种编写的源文件（参见章节6）。



全局和局部变量是通过特定的表格来进行定义和编辑的

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	hmiElevatorOn	BOOL	%MX1.78		No		---	Positioning enable
2	hmiElevatorStanding	BOOL	%MX1.83		No		---	If TRUE, elevator is not moving
3	hmiPidTest	BOOL	%MX1.1319		No		---	Starts execution of PID test
4	traceStartAcq	BOOL	Auto		No		---	Start of test PID acquisition
5	traceTrigger	BOOL	%MX1.1286		No		---	
6	hmiPIDMode	INT	%MW1.32		No		---	
7	tracePIDLen	UINT	%MW1.1284		No		---	

3. 使用环境

本章主要介绍如何设定LogicLab中的用户接口，使得IDE以最适合您的特定开发过程的方式呈现。

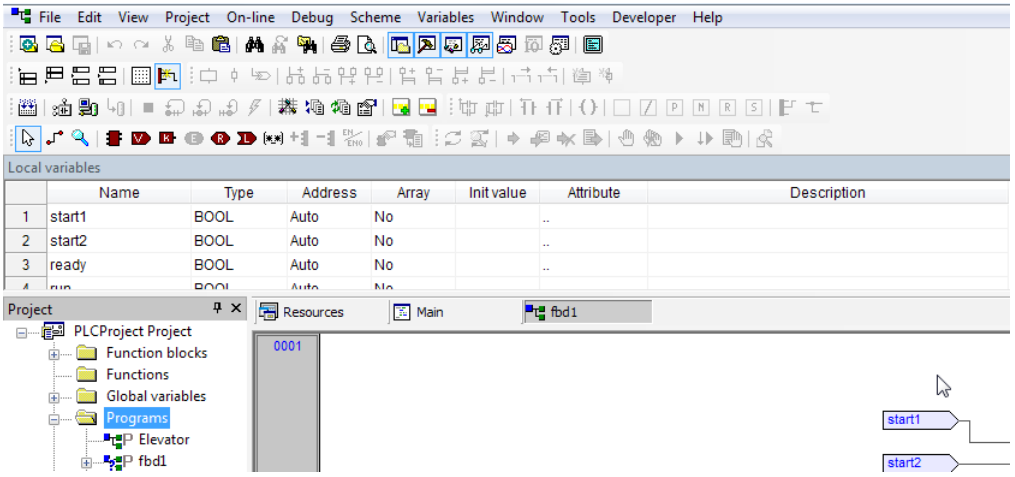
3.1 布局定制

LogicLab的工作空间的布局可以自由定制，以满足您的需求。LogicLab将会在应用程序退出时保存布局配置，以便在不同的工作窗口之间保持您设定的选项。

3.2 工具栏

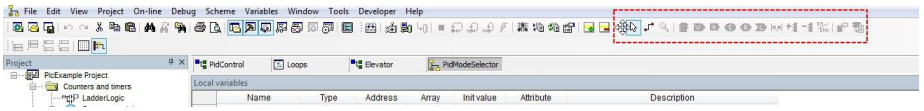
3.2.1 显示/隐藏工具栏

为了显示（或隐藏）工具栏，请打开 *View>Toolbars* 菜单，然后选择所需的工具栏（例如，FBD栏），此工具将被显示（或隐藏）。

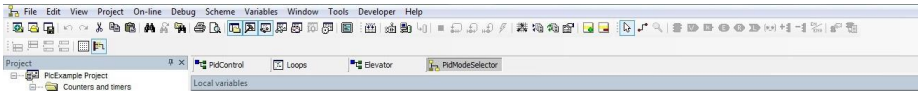


3.2.2 移动工具栏

您可以通过单击其左边框移动工具栏，然后将其拖放到目标位置。



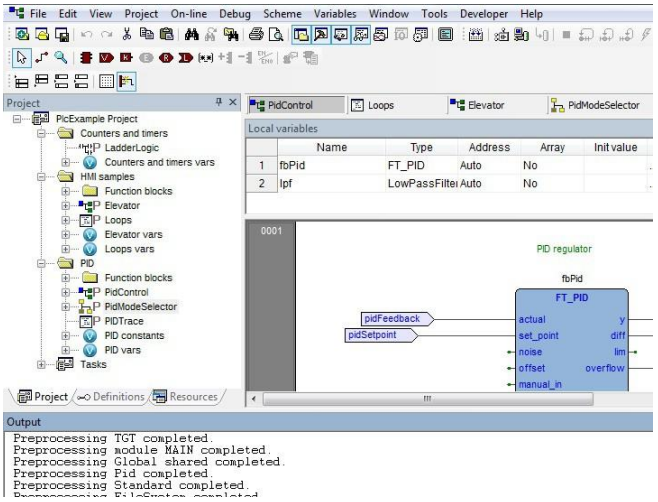
工具栏将被固定到新的位置。



3.3 停靠窗口

3.3.1 显示/隐藏工具窗口

*View>Tool Window*菜单允许您显示（或隐藏）工具窗口（例如“输出”窗口）。选中的工具窗口将被显示（或隐藏）。

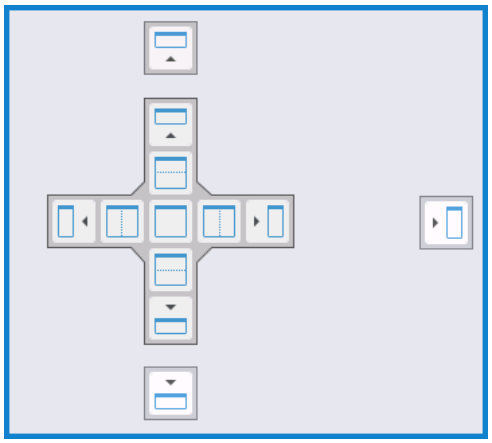


3.3.2 浮动工具窗口

您可以移动停靠在LogicLab其默认位置的任何窗口，通过拖动它到你想要的任何位置。只需双击该窗口的标题栏，即可将窗口移回最近的停靠位置。

3.3.3 停靠工具窗口

当您将窗口拖动到另一个位置时，LogicLab会显示了一个方向指南，以帮助您轻松地重新停靠窗口。拖动窗口时，通过移动鼠标光标根据方向指南将其放在新的位置。



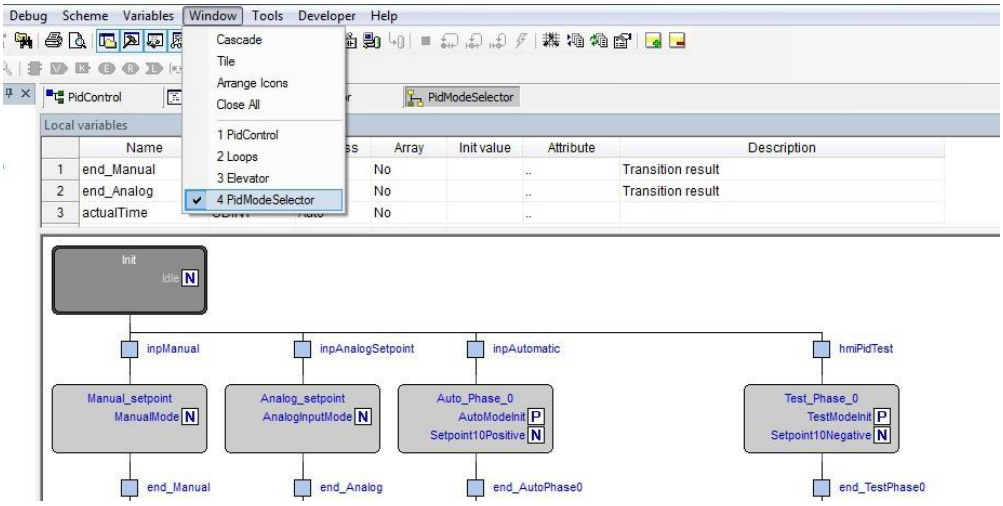
工具窗口可以固定在LogicLab中的框架内或者框架的一侧。

3.3.4 自动隐藏工具窗口

通过在窗口右上角的图钉按钮，您可以将窗口切换到自动隐藏模式或常规对接模式。

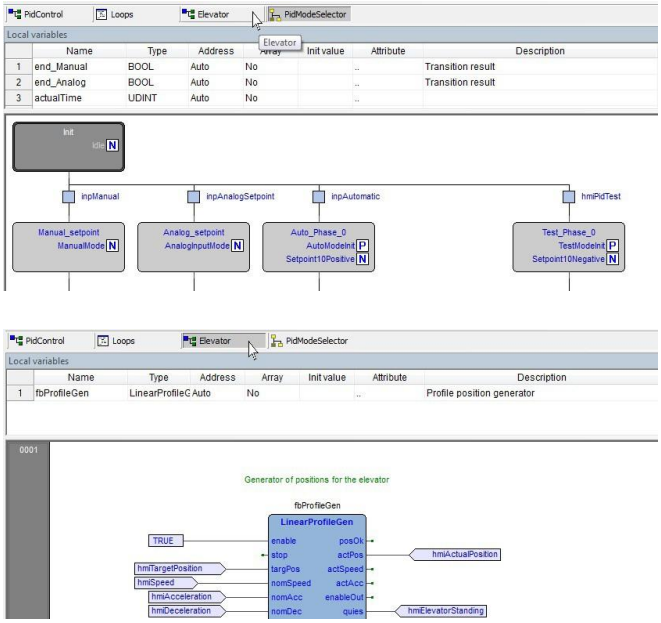
3.4 窗口导航

LogicLab允许打开许多源代码编辑器，这样可能会使工作区变得比较混乱。
但是，您可以通过文件栏和窗口菜单轻松在这些窗口之间导航。



3.4.1 文件栏

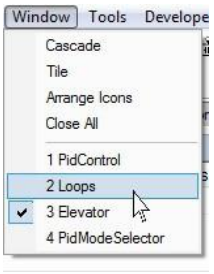
只需单击相应的文件名称，文档栏就可以在当前打开的编辑器之间进行切换。



您可以在*View>Toolbars>Document bar*菜单中根据文档的名称选择显示或隐藏文件栏。

3.4.2 窗口菜单

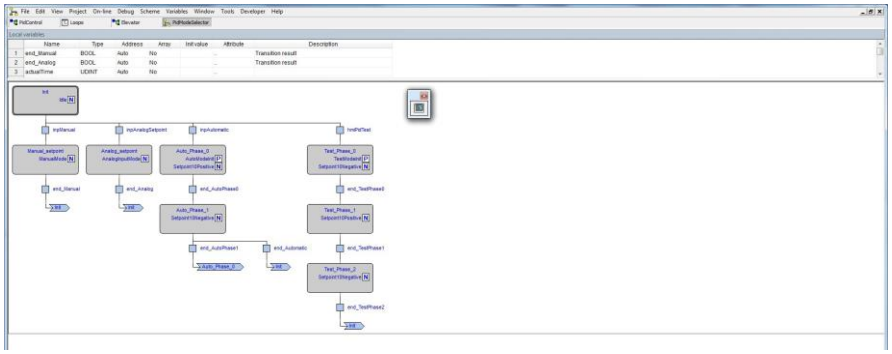
窗口菜单是文档栏的替代选项：它列出了所有当前打开的编辑器，并允许它们之间进行切换。



此外，该菜单还提供了一些命令来自动一些基本任务，例如关闭所有窗口。

3.5 全屏模式

为了方便您的应用程序的编码，你可能需要打开全屏幕模式。在全屏模式下，源代码编辑器延伸到整个工作区域，使得编辑代码的工作变得更容易，特别是当涉及图形化编程语言时（即，LD，FBD和SFC）。



您可以使用 *View>Full screen* 打开或者全屏幕模式。

3.6 环境选项

如果点*File>Options…*,将会出现一个对标签的对话框，您可以自由项LogicLab的某些选项。

3.6.1 常规选项

3.6.1.1 保存选项

自动保存：如果自动保存复选框被选中，LogicLab将会定期保存整个项目。您可以通过“自动保存时间间隔”文本框中输入分钟数来指定此自动保存动作的执行间隔。

副本数量上限：如果设置大于0，则表示可以压缩并存储在PreviousVersions文件夹中项目的副本的最大数量。

3.6.1.2 通讯

如果启用，上次使用的端口将被设置为默认端口。

3.6.1.3 提示

如果启用，当用户将光标放在编辑器中的符号上时，将出现小的信息框。

3.6.1.4 工具窗口

您可以指定用于工具窗口的字体系列和大小。

3.6.1.5 输出窗口

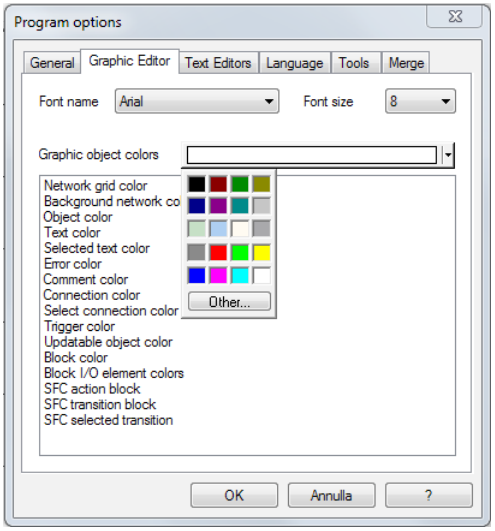
您可以指定用于输出窗口的字体系列和大小。

重置栏的位置： 在IDE中停靠栏的布局将被重置为默认的位置和尺寸。必须重新启动LogicLab生效。

3.6.2 图形编辑器

您可以在此面板上编辑LD，FBD和SFC源代码编辑器的属性。您可以指图形编辑器使用的字体系列和大小。

您还可以修改图形对象的颜色。



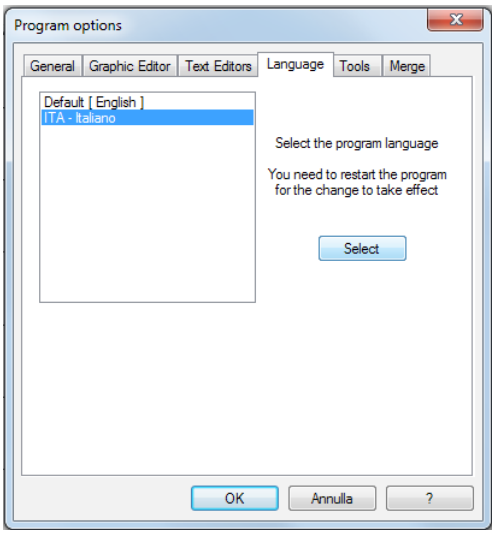
3.6.3 文本编辑器

您可以为代码编辑器和变量编辑器指定字体的系列和大小。

3.6.4 语言

您可以选择从这个面板中显示的列表下选择一个合适的语言环境。

选择新语言后，按下选择按钮，并点击OK确认。重新启动LogicLab后设定才会生效。

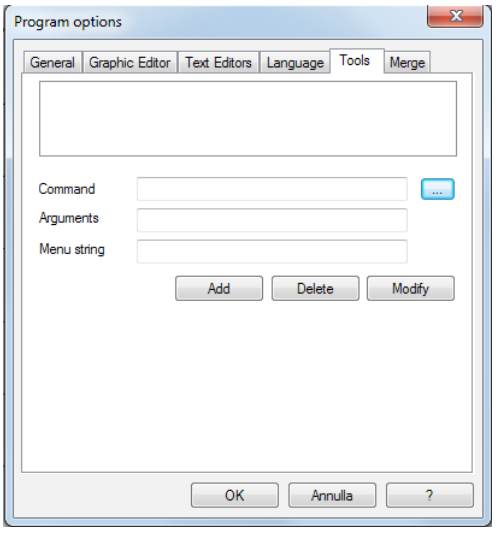


3.6.5 工具

您最多可以在“工具”菜单中添加16个命令。这些命令可以与您的操作系统上运行的任何程序相关联。

。您也可以为您添加到工具菜单中的任何命令指定参数。以下过程演示了如何将工具添加到工具菜单。

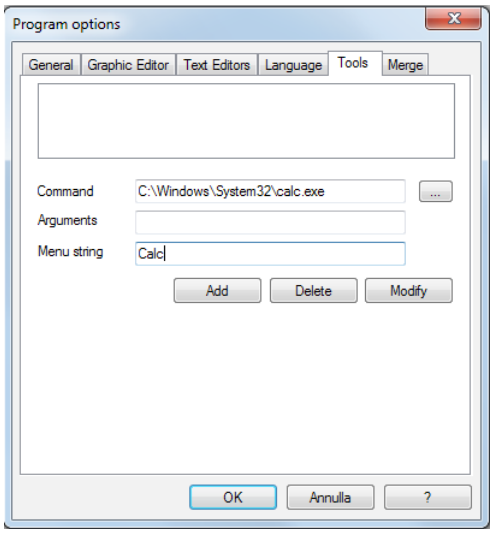
1) 在命令文本框中输入可执行文件的完整路径。此外，您也可以通过从Windows资源管理器中通过点击浏览按钮打开选择其指定的文件名。



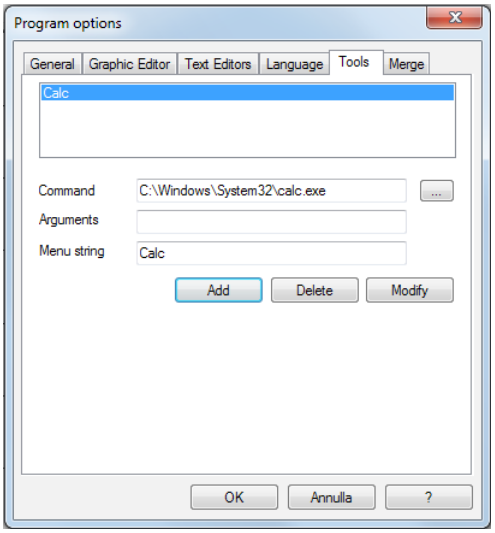
2) 在“参数”文本框中，输入要传递给在步骤1中提到的可执行命令的参数（如果有），他们必须用空格分隔。
3) 在“菜单说明”文本框中输入您想要添加的工具名称。这是将显示在“工具”菜单中的字符串。
4) 按“添加”可将新的命令加入菜单中。
5) 点击“OK”确认，或“Cancel”退出。

例如，假设我们想要将Windows计算器添加到“工具”菜单中

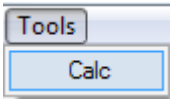
- 如图所示，填写对话框中的字段。



- 点击添加。你为新工具指定的名称将显示在面板的顶部的列表框中。

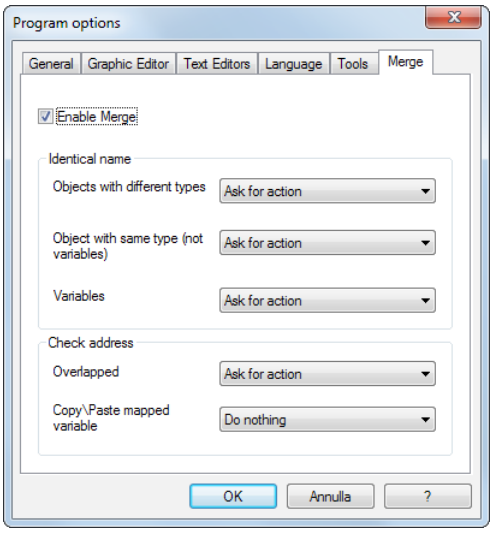


接下来就可以点击*Tools>Calc*来使用您添加的计算器工具了。



3.6.6 合并功能

您可以在这个界面中设置合并功能的属性（更多详细信息，请参见4.8.3.2章节）。



4. 管理项目

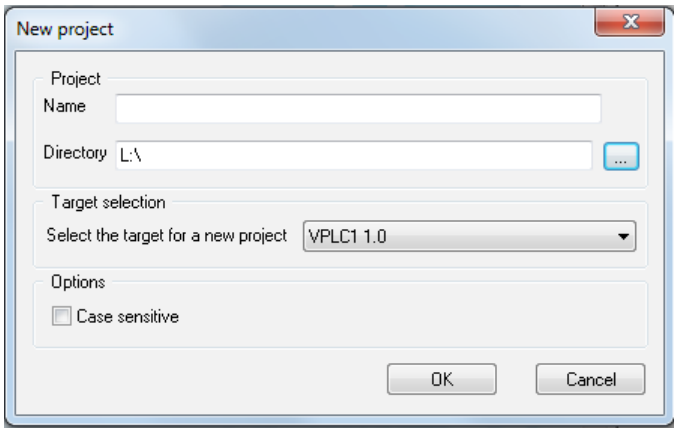
本章重点介绍LogicLab项目。

一个项目与PLC应用程序相对应，并且包括在目标设备上运行该应用程序所必须的所有必要元素，包括其源代码、链接库、以及有关目标设备的信息等等。

以下段落将说明如何创建一个LogicLab项目。

4.1 创建一个新项目

开始一个新项目，请在LogicLab的主窗口中单击 *文件>新项目* 。



您需要在“名称”文本框中输入新项目的名称。您输入的项目名称也将是LogicLab项目文件的文件夹的名称，其中将包含项目需要的文件。在“路径”文本框指定该项目文件夹的默认位置。

目标选择 指定您将运行该项目的目标设备。

最后，你可以通过激活相关的选项使项目区分大小写。需要注意的是，默认情况下在IEC 61131-3标准中，这个选项是不被激活的，也就是说当您选择创建一个区分大小写的项目，它将是不符合标准的。

当您确认创建一个新的项目，并已提供所需全部信息后，LogicLab将创建项目目录和所有项目文件；随后，该项目被打开。

您可以选择创建的项目“目标设备”的列表取决于LogicLab中配置的目标设备的目录。

如果没有您期望的目标设备，可能是您运行了错误的LogicLab安装可执行文件或者您要安装一个负责更新目标设备目录的软件补丁。在这两种情况下，您应该联系您的供应商的支持。

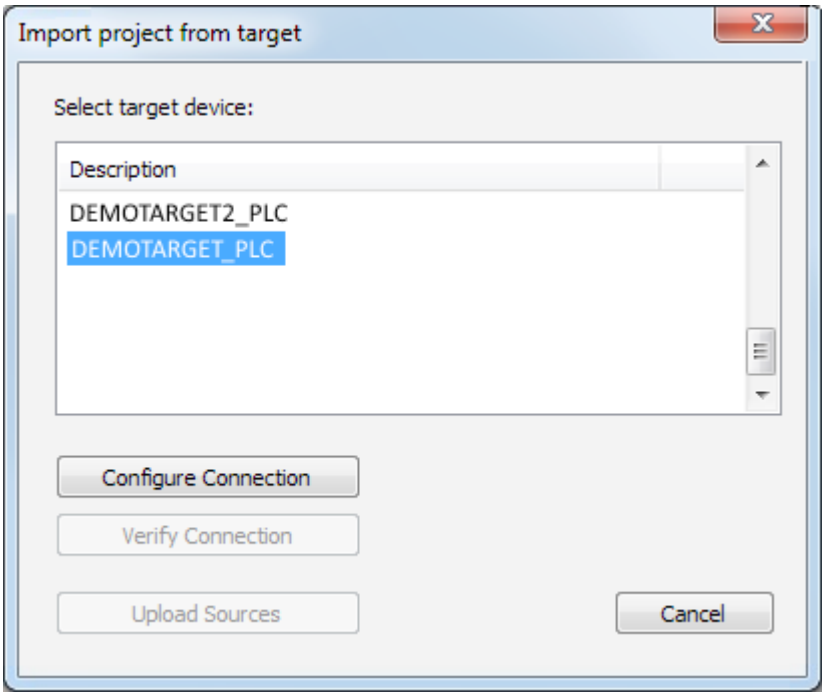
4.2 从目标设备中导出项目

根据与您连接的目标设备的不同，您也许可以从目标设备中上载其本身正在运行的LogicLab项目。

为了从目标设备上上载LogicLab项目，请按照下列步骤操作：

1

1) 在LogicLab主窗口菜单中点击文件>导出，这将打开目标列表对话框。



2) 从显示列表从中选择您要上再项目的目标设备。
3) ***配置连接*** 中配置正确的通讯参数（更多细节见第8.1节）。
4) 您可以通过“***验证连接***”按钮测试与目标设备的连接。LogicLab将尝试建立连接目标设备并报告测试结果。
5) 如果连接可用，请单击“***上载项目***”按钮确认操作。当应用程序上传成功完成，该项目将自动打开，可以进行编辑。

4.3 保存项目

4.3.1 变更项目保存

当你对项目进行了任何更改时（例如添加一个新的程序组织单位），您需要对该项目的更改进行保存。至，您可以选择相应的保存选项 *文件>保存项目* ，来保存此项目的更改。

4.3.2 项目另存为

您也可以使用 *文件>另存为* 选项来重命名项目，更改其保存格式或修改您想要保存文件的位置。

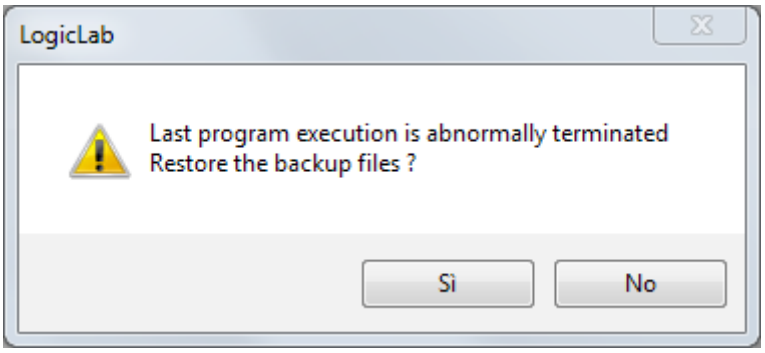
LogicLab要求您选择新的目标位置（必须是空目录），然后保存项目的副本到该位置，并打开并编辑这个新的项目文件。

4.3.3 自动保存

LogicLab拥有自动保存功能，它可以定期保存您的项目。

自动保存 将数据保存在名为“Backup”的独立的文件夹中，该文件夹存储在项目文件夹同级目录下。

自动保存 可在LogicLab意外退出时保存您的项目。当再次启动LogicLab时，如果存在“Backup”文件夹，则您可选择恢复项目的最后一个有效的备份文件。



当您正确地关闭LogicLab时，备份文件夹及其内容将被删除。您可以指定两次保存之间的间隔时间（以分钟为单位）。

默认情况下，自动保存的运行间隔为1分钟（更多详细信息，请参加 3.6 节）。

4.3.4 制作备份

LogicLab具有对该项目的历史版本进行备份的功能。

当您显式的保存项目时，LogicLab会将项目之前的版本保存在文件夹“PreviousVersions”中，该文件夹存储在项目文件夹的同一级目录下;

您可以设置保存在您的PC上备份文件的上限。默认情况下为10，如果要禁用此功能，可设置为0（更多详细信息，请参加 3.6 节）。

4.4 管理现有项目

4.4.1 打开现有项目

要打开现有项目，请点击LogicLab的主窗口中 *文件\打开项目* 的，或在 ***欢迎页面*** （未打开任何项目）。这将会弹出一个对话框，您可以加载包含项目的目录，并选择相对工程文件。

4.4.2 编辑项目

为了编辑项目中元素，你可以通过浏览工作区栏的项目选项卡中工程树找到该元素，然后通过双击其名称来打开此元素。

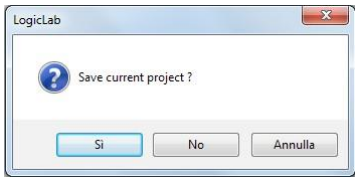
通过双击需要修改对象的名称，将会打开该对象类型的编辑器：例如，当你双击一个项目POU的名称时，将显示相应的源代码编辑器；如果你双击一个全局变量名，将会显示变量编辑器。

需要注意的是，当处于以下模式时， LogicLab项目将处于无法编辑的状态：

- 当前处于调试模式。
- 这是一个包含库的对象（而您可以修改您从库导入的对象）。
- 该项目以只读模式（查看项目）打开。

4.4.3 关闭项目

您可以通过显式关闭项目或者退出LogicLab来终止工作窗口。在这两种情况下，当有尚未保存到文件时，LogicLab都会要求您在保存和丢弃此项目之间选择。



如果要关闭项目，请选择 *文件>关闭项目*；LogicLab将重新显示“欢迎”页面，以便您可以迅速开始新的项目。

4.5 项目发布

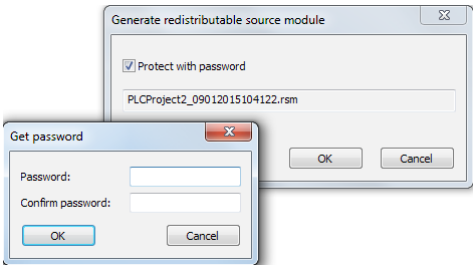
当你需要与其他开发者共享一个项目时，可以向他/她发送项目文件的副本或者LogicLab生成的可发布的源模块（RSM）。

在前一种情况下，你要共享的文件的数量取决于项目文件的格式：

- PLC单项目文件（文件扩展名为.ppjs）：项目文件本身包含运行应用程序所需的全部信息（假设接收项目您的开发者具有可用的目标设备），包括所有的源代码模块。因此您仅需要共享.ppjs文件。
- PLC多个项目文件（文件扩展名为.ppjx或.ppj）：项目文件只包含构成项目的源代码模块的连接，这些链接作为单个文件存储在项目目录中。您需要共享整个目录。
- 完整的XML PLC项目文件（文件扩展名为.plcprj）：项目文件完全是用XML语言生成的。项目文件中包含的信息和应用与扩展名.ppjs的项目是相同的。

或者，你也可以使用 *项目>生成可再分发的源模块* 生成可再分发的源模块（RSM）。

LogicLab将会通知您RSM文件的名称，并让你选择是否使用密码保护文件。如果您选择保护文件，则LogicLab要求你输入密码。



RSM格式文件的优点是：

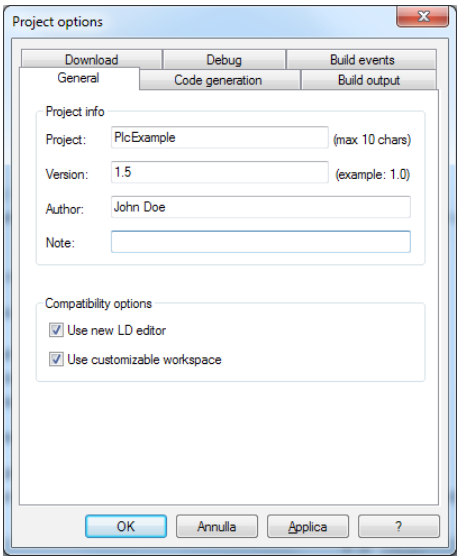
- 源代码以二进制格式编码，因此除LogicLab的第三方软件无法读取源代码，从而使得通过网络传输更加安全；
- 可以用一个密码进行保护，LogicLab在打开文件时需要输入密码验证；
- 作为二进制文件，相比其他项目文件尺寸大大减小。

4.6 项目选项

您可以在 *项目>选项...* 中编辑一些重要的项目属性。

4.6.1 项目信息

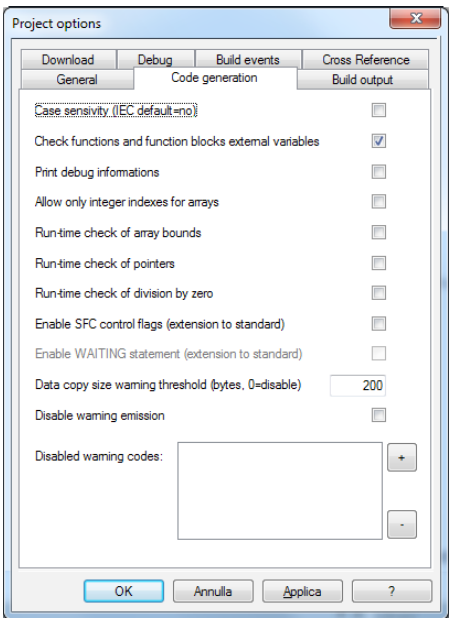
您可以在这里设置与项目相关的一些基本属性，例如其应用程序名称和版本。



- ***使用新的LD编辑器***：新梯形图编辑器更容易使用，帮助您在使用LD语言时会更快和更高效。需要注意的是，这个选项是被默认勾选的。
- ***使用可自定义的工作空间***：允许你管理你的项目树，使得您的工更高效。需要注意的是，这个选项是被默认勾选的。

4.6.2 生成代码

在这里，您可以编辑有关代码生成的一些属性。

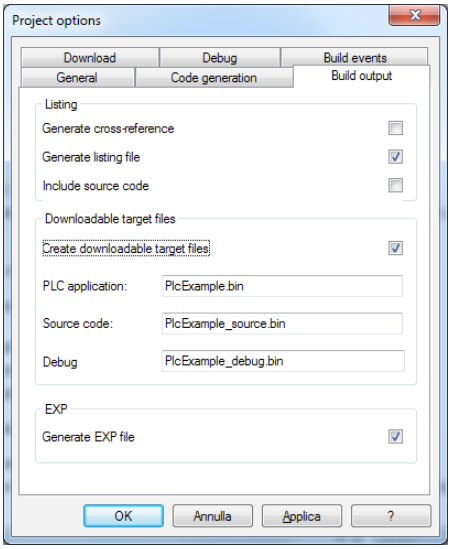


- ***区分大小写***：您可以选中此选项将项目设置为区分大小。注意，这个选项是默认不勾选的。

- ***检查功能和功能块外部变量***：如果禁用此选项，则所有的功能和功能块都可以访问全局变量，而无需将其声明为外部变量。需要注意的是，默认情况下，根据IEC 61131-3标准启用该选项。
- ***打印调试信息***：在输出窗口中打印一些重要的调试信息。
- ***仅允许使用整数索引作为数组的索引***：如果选中此选项，则不能将BYTE，WORD或DWORD用作数组索引。
- ***数组越界的运行时检查***：如果选中此选项，则会添加一些检查代码以验证在运行时对数组索引在运行时不会超出范围。可以根据目标设备设置此选项。
- ***通过零运行时分的检查***：如果选中此选项，则会添加一些检查代码以验证在运行时对数组执行零除。可以根据目标设备设置此选项。
- ***指针的运行时检查***：如果选中此选项，将在使用指针之前检查其有效性，并且在目标设备上调用用户自定义的函数 checkptr 。可以根据目标设备设置此选项。
- ***启用SFC控制标志（扩展标准）***：如果选中此选项，则启用SFC POU的HOLD和RESET标志。
- ***启用WAITING声明（扩展标准）***：如果选中此选项，则ST语言中的WAITING构造将作为IEC61131-3扩展名添加（更多详细信息，请参见11.7.3节）。
- ***数据复制大小警告阈值（单位：BYTE，0 = 禁止）***：当复制数组或结构体时，如果其尺寸超过规定的阈值时，则会发出警告通知的PLC的性能可能会受到影响。如果阈值设置为0，则不会发出警告。
- ***禁用输出警告***：如果选中此选项，则不会将任何警告打印在输出窗口上。
- ***禁用指定警告***：如果选中此选项，则某些指定的的警告不会打印到输出窗口上。

4.6.3 编译输出

在这里，您可以编辑编译生成的输出文件的一些重要属性。



属性清单

- ***生成清单文件***：如果这个选项被选中，编译器将生成一个命名为 projectname.lst 的列表文件。
- ***包含源代码*** （仅在选中“生成清单文件”时有效）：如果这个选项被选中，则源代码将在第一文件中以可见的形式插入。否则，将源代码被隐藏。

可下载的目标文件

- ***创建可下载的目标文件***：如果这个选项被选中，则编译器将生成可下载的目标设备的二进制文件。您可以指定自定义文件名或使用默认的文件名。

请注意，仅接受有效的Windows文件名称！

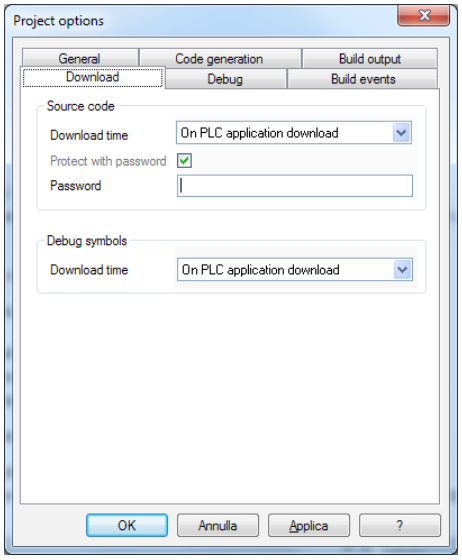
- ***PLC应用程序*** （仅在选中“创建可下载的目标文件”时有效）：该字段指定PLC应用程序二进制文件的名称。默认情况下为 porjectname.bin
- ***源代码***（仅在选中“创建可下载的目标文件”时有效）：该字段指定源代码的二进制文件的名称。默认情况下porjectname._source.bin
- ***调试***（仅在选中“创建可下载的目标文件”时有效）：该字段指定调试符号二进制文件的名称。默认情况下projectname._debug.bin

生成EXP文件

- ***生成EXP文件***：如果这个选项被选中，编译器将生成一个名为projectname.exp的EXP文件

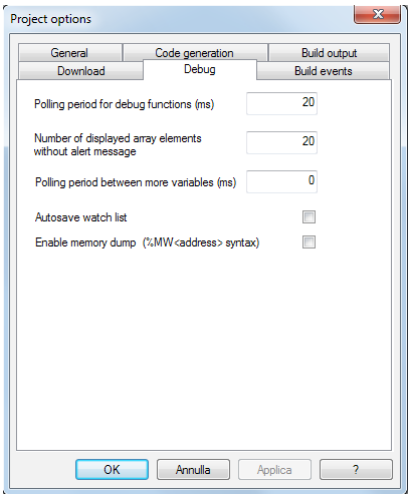
4.6.4 下载

在这里，您可以编辑一些有关下载行为的重要属性。（更多详细信息，请参见8.3.1节）



4.6.5 调试

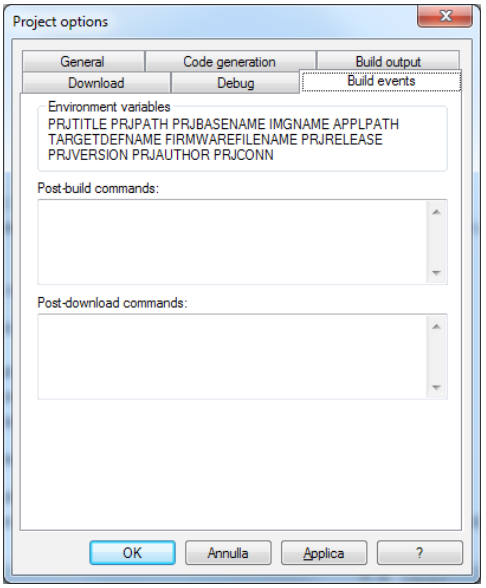
在这里，您可以编辑一些有关调试行为的重要属性。



- ***调试功能的轮询周期*** （ms）：设定对功能状态的有效采样周期。
- ***可监视的最大数组元素的数量***：指定监视窗口中可以监视的最大数组元素的数量，超过此上限将会在输出窗口中发出警告。
- ***多变量之间的轮询周期*** （ms）：设置对两个变量进行采样之间的休眠时间。
- ***自动保存监视列表***：如果项目关闭将自动保存监视列表状态到文件中（更多详细信息，请参见9.1.7节）。

4.6.6 生成事件

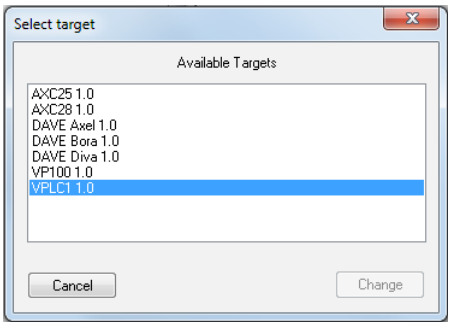
在这里，您可以添加编译前和编译完成后运行的命令。您还可以使用在窗口顶部列出的环境变量。



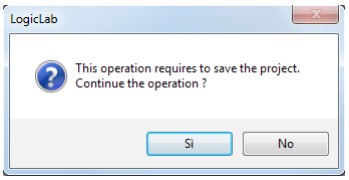
4.7 选择目标设备

您可能需要在目标设备上移植一个PLC应用程序，该设备不同于您最初为其编写代码的设备。请按照下面的说明修改LogicLab项目设置以适应新的目标设备。

1) 点击LogicLab主窗口中的 *项目>选择目标设备* 菜单。将会弹出以下对话框。



2) 在下拉框中列出的目标设备中，选择您的目标设备。
3) 单击修改按钮以确认您的选择，或者点击取消按钮中止操作。
4) 如果确认修改，LogicLab将会弹出下面的对话框。



点击是按钮完成转换，点击否按钮退出修改。

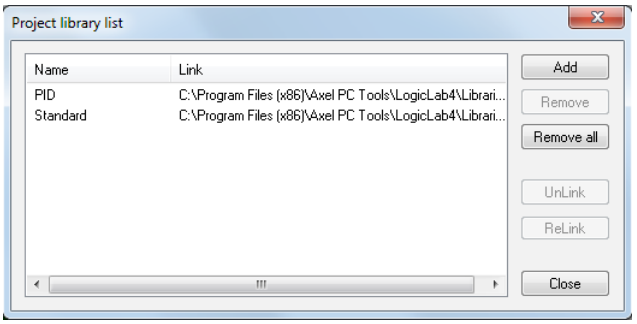
如果点击是，LogicLab将更新项目设置为新的目标设备。

此外，LogicLab还会在项目目录的子目录下制作项目文件的副本，以方便您在任何意外的情况下可以通过手动将项目文件替换为备份文件来进行版本回退操作（利用Windows资源管理器）。

4.8 库管理

库可以方便的在LogicLab项目之间共享资源，其通常存放扩展名为.pll的专用源文件中。

4.8.1 库管理器



库管理器列出了LogicLab项目中当前包含的所有的库。您可以在此进行添加库或删除库的操作。

如果想要访问库管理器，请单击 *项目>库管理器* 。

4.8.1.1 添加一个库

以下过程介绍如何在LogicLab项目中添加一个库，使得可在该项目中使用所有此库中的对象。

添加库意味着将库的.pll文件的引用添加到当前项目中，并创建该库的本地副本。需要注意的是，您只能进行导入的操作而无法编辑库中的元素。

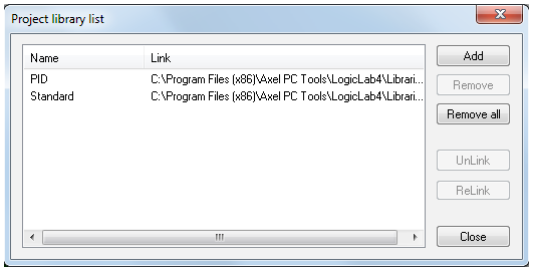
如果您想要复制或移动包括一个或多个库的项目，请确保改项目对这些库的引用在新的位置仍然有效。

1) 点击 *项目>库管理器* ，将弹出 ***库管理*** 对话框。
2) 点击添加按钮，在弹出的资源管理器对话框中你选择你要打开的库的.pll文件。
3) 找到.pll文件后，请双击它或通过按“打开”按钮将其打开。库的名称及其绝对路径将显示在白框列表的底部新行中。
4) 对于所有要包括的库重复步骤1，2和3。
5) 当你完成了添加库的操作后，请按OK键确认，或按Cancel键退出。

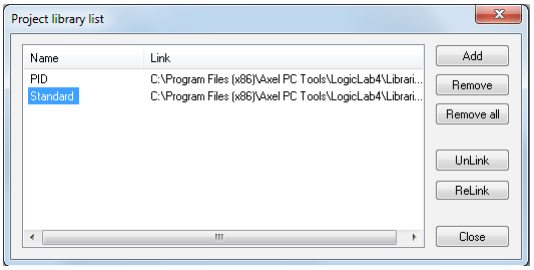
4.8.1.2 删除库

以下过程介绍如何从LogicLab项目删除已包含的库。请注意，删除库操作并不意味着删除库本身，而只是删除项目对库的引用。

1) 点击LogicLab主窗口的 *项目>库管理器* 菜单，将弹出 ***库管理*** 对话框。



单击选择您想删除库的名称。删除按钮将处于可用状态。



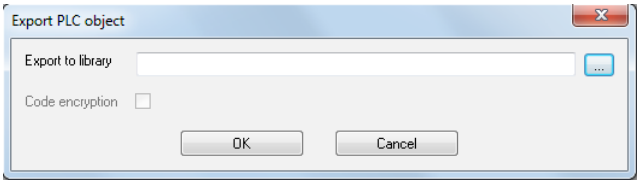
2) 单击删除按钮，选定库的引用将会从此列表中删除。
3) 对要删除的所有库重复上述操作。此外，如果你想删除所有的库，可以按 ***全部删除*** 按钮。
4) 当您完成删除库的操作后，请按OK键确认，或者按Cancel键取消操作。

4.8.2 导出库文件

您可以在当前打开的项目中导出对象到库，以提供给其他项目使用。下面的过程演示如何将对象导出到库。

1) 在工作区栏的项目选项卡中的树状结构中找到您需要导出的对象，然后单击该对象的名称。

2) 点击 *项目>导出对象到库* 。将弹出下面的对话框。



3) 指定生成的.pll文件的目标位置。您可以选择以下方式：
   - 在白色文本框中输入的完整路径;
   - 点击“浏览”按钮，以打开资源管理器对话框，选择您希望保存.pll文件的目标地址。
4) 您可以任意选择加密要导出的POU的源代码，以保护您的知识产权。
5) 点击“OK”确认操作，否则按“Cancel”退出。

如果在此过程的步骤3中您输入的是一个不存在的.pll文件名称，则LogicLab将创建文件，从而建立一个新的库。

4.8.2.1 撤消导出库

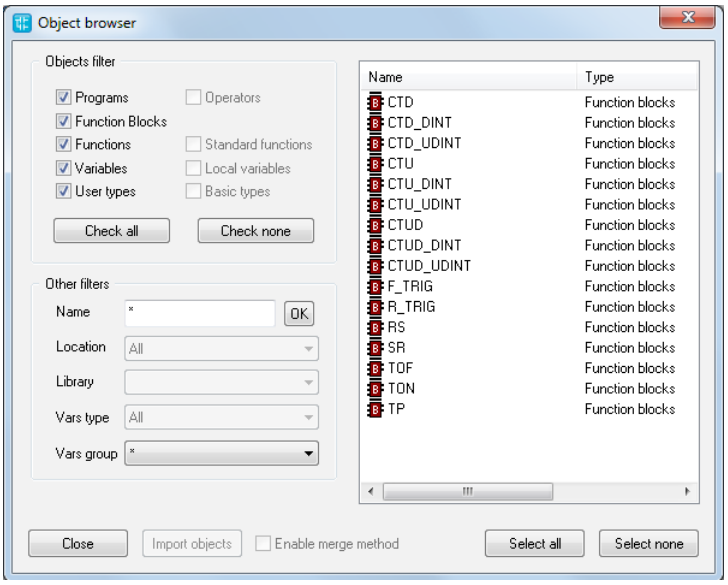
此时已经无法撤销导出库的操作，如果此时您发现库中包含了一个错误的对象，只能重新创建并导出到另一个库，在此库去除您不需要的对象。

4.8.3 从库或其他来源导入

您可以从库中导入对象，以便在当前项目中使用它。当您从库中导入的对象时，该对象的本地副本将失去对原始库的引用，并且仅属于当前项目。因此，不同于包括在库中的对象，您可以编辑导入的对象。

有从库中获得POU的有两种方式。以下步骤显示了如何从库中导入对象。

1) 点击 *项目>从库中导入对象* ，将弹出资源管理器对话框，选择您需要打开的库的.pll文件。
2) 找到.pll文件后，双击它或按“打开”按钮打开它的。将弹出资源管理器对话框。该对话框中每个选项卡包含一对象列表，其类型与选项卡的标题一致。



3) 选择要导入的对象的类型的选项卡。您还可以使用过滤器对每个选项卡中的对象进行简单的查询。但是请注意，名称过滤器只适用于库。要使用它，请选择一个选项卡，然后输入所需的对象的名称，甚至可以使用 * 通配符来进行查找。

4) 选择所有您要导入的对象，然后按 **导入对象** 按钮。

5) 完成导入的对象后，点击“OK”确定操作点击“Cancel”以关闭库浏览器。

4.8.3.1 撤消导入对象

当您在LogicLab项目导入一个对象时，实际上使用的只是该对象的本地副本。因此，你只需要删除本地对象即可撤销导入对象的操作。

4.8.3.2 合并功能

当您在LogicLab项目中导入对象或插入复制的映射变量时，你可能会遇到地址重叠或重复命名的警告。通过设置相应的环境选项（详细信息，请参见 3.6 节），你可以选择LogicLab在遇到这些问题时应该处理的行为。

可能的操作有：

		询问	自动	从库中导入	不处理
命名行为	如果不同的类型	X	X		X
	如果类型相同，但不是变量	X	X	X	
	如果两个都是变量	X	X	X	
地址行为	如果地址重叠	X	X	X	
	复制/粘贴映射变量		X		X

- ***询问***（默认）：用户必须决定每次需要采取的措施。
- ***自动***：LogicLab会自动生成一个有效的名称或地址，将其分配给导入的对象。

- ***从库中导入***：名称或地址取自导入的对象。
- ***不处理***：项目中对象的名称或地址将不做任何处理。

导入的对象后，LogicLab将在项目文件夹中生成详细的的日志文件信息。

4.8.4 更新库

如果您编辑链接的库文件，则可以在不关闭LogicLab的情况下刷新其项目中的内容。

1) 点击 *项目>刷新所有库* 。
2) 如果该文件是正确的，则LogicLab将会更新链接库的内容，并打印在输出窗口一条成功的消息，否则，不会对现有链接库进行任何更改。

5. 管理项目要素

本章主要介绍如何处理组成项目的要素：
程序组织单元（简称为POU），任务，派生数据类型和变量。

5.1 程序组织单元

POU是程序，功能或功能块类型的程序组织单元。

本段介绍了如何将新的POU添加到项目中，如何编辑以及最终删除它们。

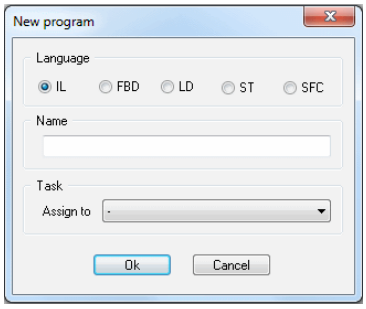
5.1.1 创建一个新的程序组织单元

首先在选择菜单中添加一个指定类型的POU。

项目>新的对象>新的程序

请注意，子菜单中的项目可能会根据您想要创建的POU的类型而变化。

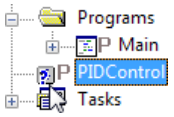
LogicLab将弹出一个对话框，需要您在文本框中给POU命名并选择合适的编程语言。



点击“OK”按钮确认操作。

另外，您也可以从上下文菜单中选择一个文件夹或项目的根元素创建一个新的POU（详细信息，请参见5.7.4节）。

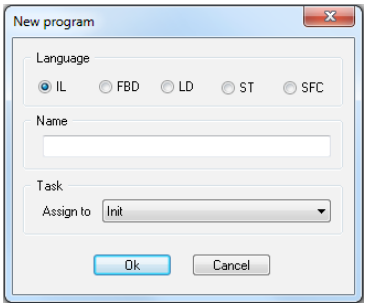
创建新程序后，新程序图标的下方将出现警告图标（问号）。



此警报图标表示该程序尚未与任务关联。请参阅第5.3.1节，将程序分配给所需的任务。

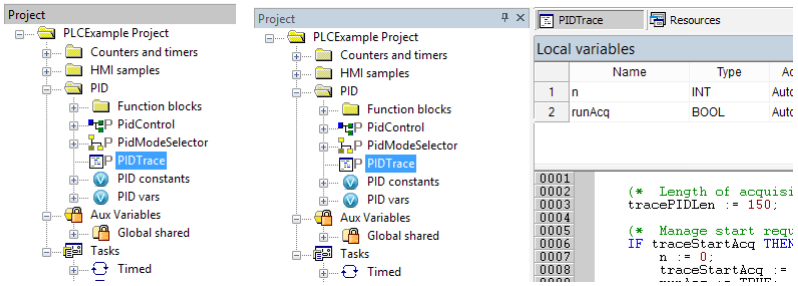
5.1.1.1创建程序时指定任务

您可以在LogicLab中创建新的程序时，将程序分配到任务：在“新建程序”窗口中的“任务”下拉列表中选择要关联该程序的任务。



5.1.2 编辑POU

首先从项目树中双击将它打开，在项目编辑器中即可修改POU的源代码。



更改POU的名称：

从项目树中选择一个POU，菜单中选择 *项目>PLC对象属性* 。
请注意，菜单中选项可以根据所选择的POU的类型而改变。

复制一个POU：

从项目树中选择一个POU，菜单中选择 *项目>复制对象* 。
请注意，菜单中选项可以根据所选择的POU的类型而改变。

输入新复制的POU的名称并确认操作。

删除POU

从项目树中选择一个POU，菜单中选择 *项目>删除对象* 。

请注意，菜单中选项可以根据所选择的POU的类型而改变。

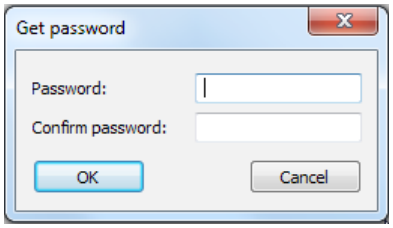
确认删除POU操作。

5.1.3 源代码的加密/解密

LogicLab可以对POU加密并用密码保护它们，从而隐藏POU的源代码。

加密POU：
从项目树中选择一个POU，然后在上下文菜单中选择 *[加密]* 选项

输入密码并确认操作。



LogicLab将在工程树中显示一个特殊的标记，该标记覆盖指定的POU图标，以此来通知用户POU已加密。

解密POU：

从项目树中选择一个POU，然后在上下文菜单中选择 *[解密]* 选项

加密所有POU：

选择项目树中的根节点，然后在上下文菜单中选择 *[加密所有对象]* 选项。

则所有POU将使用相同的密码进行加密。

解密所有POU：

选择项目树中的根节点，然后在上下文菜单中选择 *[解密所有对象]* 选项。

5.2 变量

LogicLab中有两种类型的变量：全局变量和局部变量。

本段将说明如何在项目中添加、编辑、并最终删除全局和局部变量。

5.2.1 全程变量

全局变量可以在项目中的任何模块中被看到和引用。

5.2.1.1全局变量类别

全局变量被定义在项目树的制定文件夹中，成为全局变量组。这些变量根据其性质可分类为：

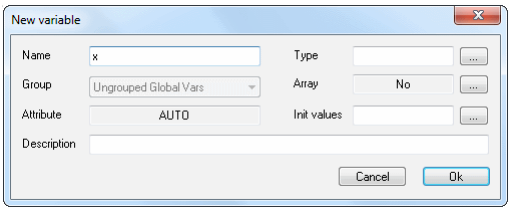
- 自动变量：编译器自动会将它们分配到目标设备内存中的适当位置。
- 映射型变量：它们在目标设备逻辑寻址系统中有一个指定的地址，应由开发人员指定。
- 常量：声明为具有CONSTANT属性，常量不能被写入。
- 保持型变量：他们它们具有RETAIN属性; 它们的值存储在目标设备的掉电保持区域。

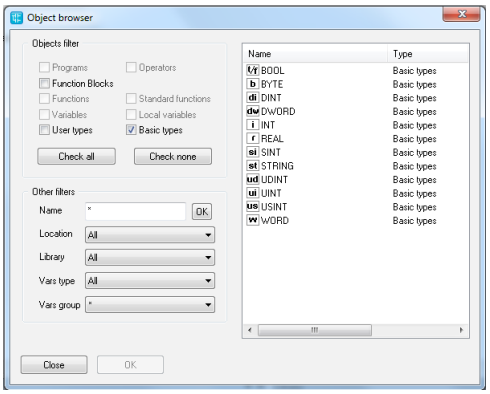
5.2.1.2 创建一个新的全局变量

1) 为了创建一个新的全局变量，您需要先定义一个全局变量组，然后从工程树中选择它，在菜单中选择 *项目>新的对象>新变量* 选项

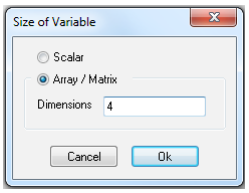
2) LogicLab将弹出一个对话框，您必须输入变量的名称（请记住，不能使用某些字符，比如‘?’，‘.’，‘/’等：变量名必须是有效IEC 61131-3标识符）。

3) 通过键入变量类型或点击“浏览”从LogicLab显示的列表中选择指定变量的类型。

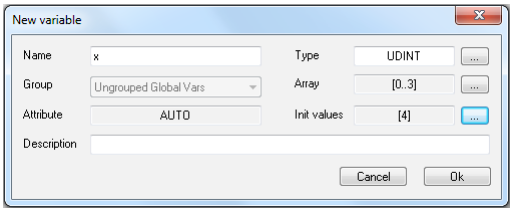


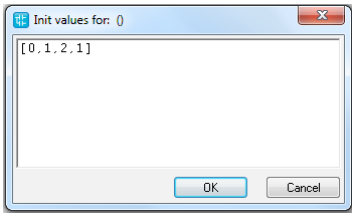


如果要声明一个数组，则必须指定其大小。



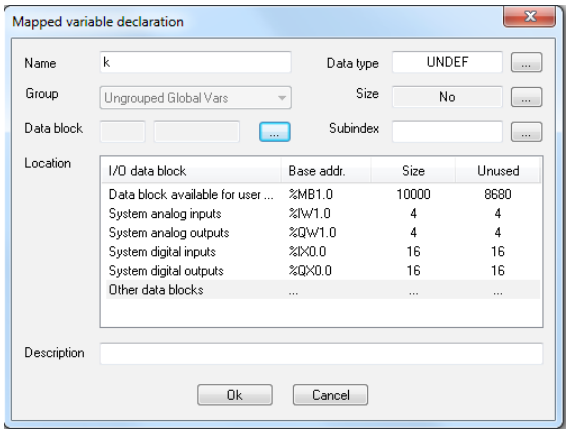
4) 您可以设定变量或者数组中单个元素的初始值。

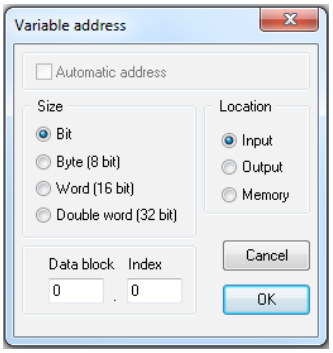




如果你创建一个新的映射变量，则需要在定义变量的时候指定变量的地址，您可以执行下列操作之一：

- 点击按钮，打开地址编辑器，然后输入所需的值。

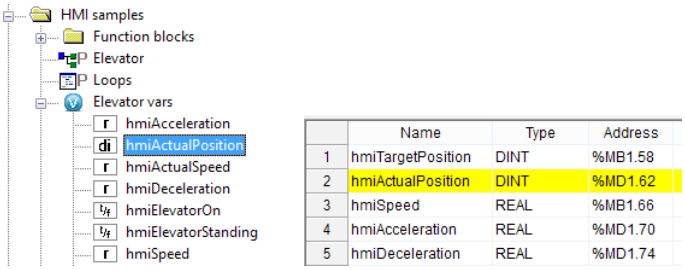




- 从LogicLab显示的列表中选择您希望使用的存储区域，该工具将会自动计算该区域的第一个空闲存储位置的地址。

5.2.1.3 编辑一个全局变量

要编辑现有的全局变量的定义，请从项目树中双击它或从它所属的文件夹中将其打开。则您可在全局变量编辑器中修改它的定义。



更改变量的名称：

从项目树中选择要重命名的变量，然后在菜单中选择 *项目>显示PLC对象属性* 选项。

复制一个变量：

从项目树中选择要复制的变量，然后在菜单中选择 *项目>复制变量* 选项。

输入新复制的变量的名称并确认。

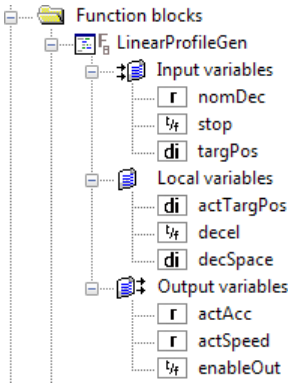
5.2.1.4 删除全局变量

从项目树中选择要删除的变量，然后在菜单中选择 *项目>删除变量* 选项，并确认删除该变量的操作。

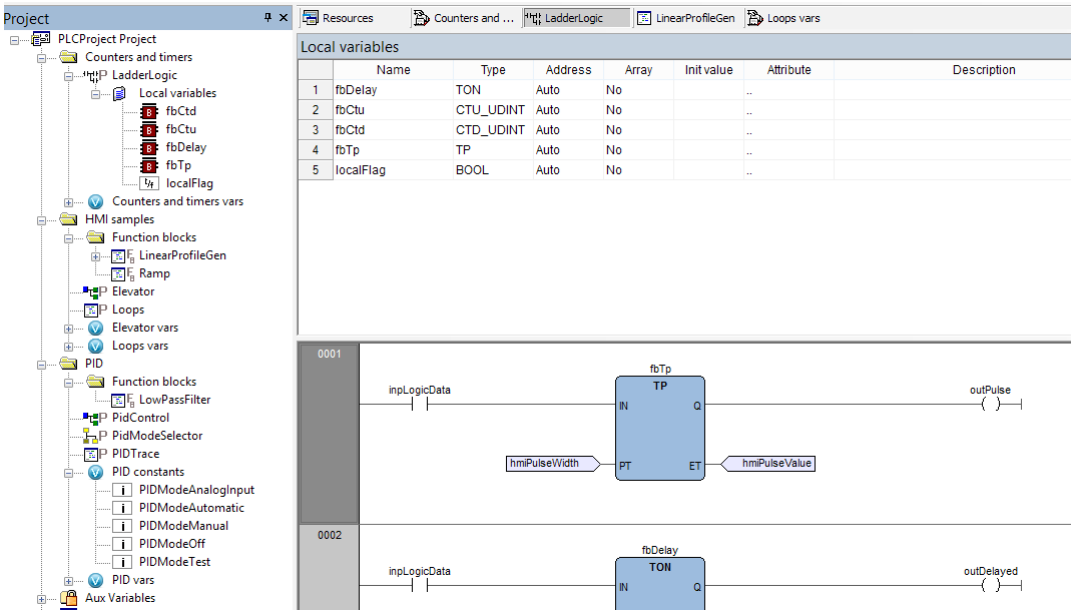
5.2.2 局部变量

局部变量在POU（程序、功能或功能块）中声明，模块本身是唯一可以引用和访问它们的项目元素。

局部变量在其所属的POU的项目树中列出（仅当POU打开并进行编辑时），根据其类别进一步分类（例如，作为输入或输出变量）。



打开对应的程序组织单元，使用局部变量编辑器即可创建，编辑和删除局部变量。更改或添加局部变量后，需要保存该项目以更新项目树的POU分支结构，包括应用到局部变量的更改。



有关详细信息，请参考本手册相应的部分（请参见6.6.1.2节）。

5.2.3 创建多个变量

LogicLab允许您一次创建多个变量。

打开POU进行编辑，然后选择菜单中选择 *变量>创建多个变量* 。

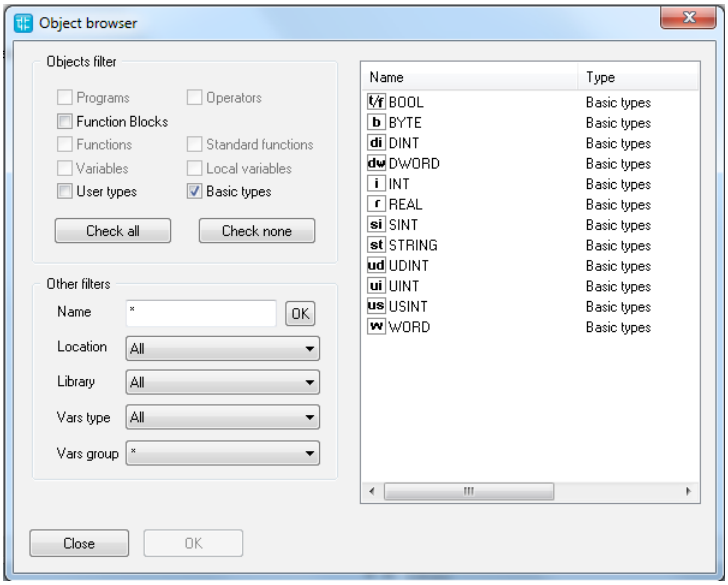
LogicLab将向弹出一个对话框，您必须指定前缀和后缀来命名新变量。

1) 选择变量的类型。
2) 插入要创建的变量数，指定开始索引编号、结束索引编号和步长值。您可以在对话框底部看到生成的变量名的示例。

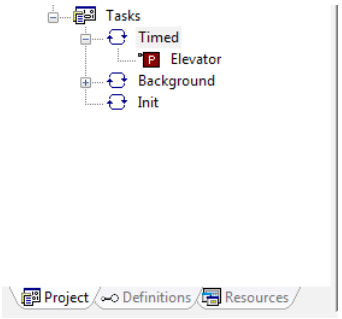
5.3 任务

5.3.1 关联一个程序到任务

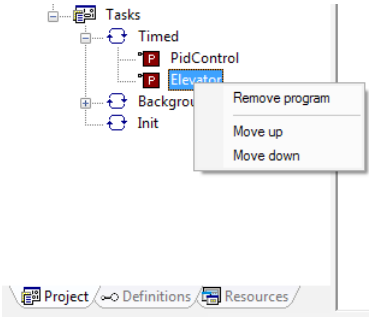
1) 从项目树中选择要在其中添加的程序的任务，然后从上下文菜单中选择 *[添加程序]* 选项。
2) 从显示的列表中选择要由任务执行的程序，并确认您的选择。



3) 如项目树中所示，程序已分配给任务。



请注意，您可以为任务分配多个程序。从上下文菜单中，你可以进行排序，并最终删除对任务的程序分配。

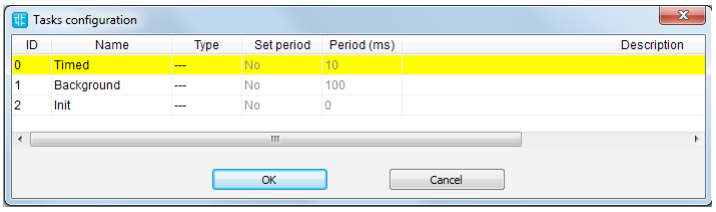


5.3.2 任务配置

根据所连接的目标设备的不同，对PLC任务的设置也有些不同。

从项目树中选择任务元素，然后选择上下文菜单中选择 *[任务属性]* 选项。

在任务配置窗口中，您可以编辑任务执行周期。



5.4 派生数据类型

通过 *工作区* 窗口的的 *定义* 允许您定义派生数据类型。派生数据类型是一个复杂的分类，可定义为一种或多种数据类型，其由基本数据类型组成。

用户可以灵活地创建具有高级属性的数据类型，其使用场景远远超出了基本的原始数据类型。

5.4.1 类型定义

下面段落显示了如何管理类型定义。

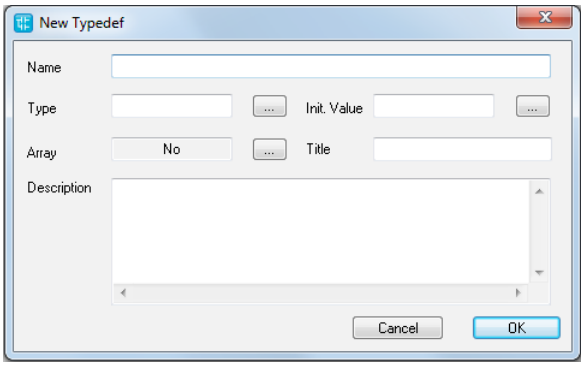
LogicLab可以管理的类型定义包括结构体、枚举和子范围。

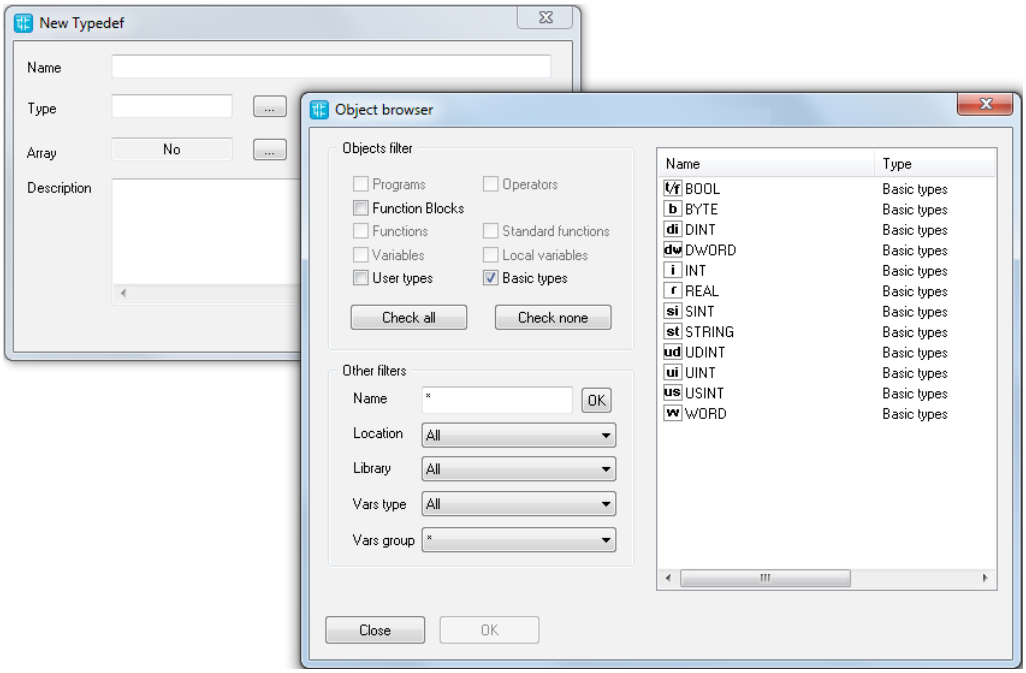
为了创建、编辑或删除这些数据类型，请使用 *工作区* 窗口的的 *定义* 选项。

5.4.1.1 创建一个新的数据类型

为了创建一个新的数据类型，请在 *定义* 选项列表中选择 *类型定义* 文件夹项，然后在上下文菜单中选择 *新类型定义* 选项。

LogicLab将向弹出一个对话框，您必须指定新数据类型的名称并选择要定义的类型：

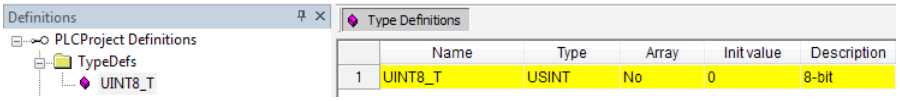




（如果你想为数组类型定义别名，则应选择数组的大小）并可输入您的描述信息（可选），最后确认操作。

5.4.1.2 编辑类型定义

为编辑现有的类型定义，你必须从 ***定义*** 选项列表中双击打开它的编辑器，并在其中修改它的定义。



5.4.1.3 删除类型定义

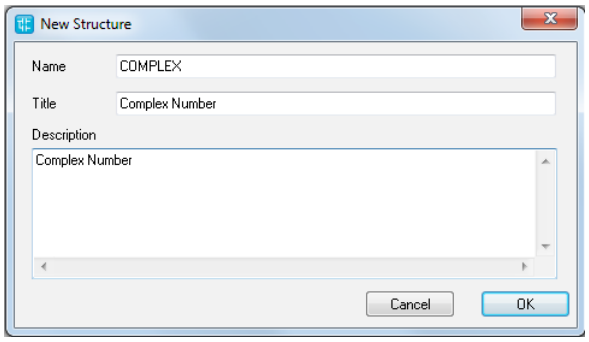
为删除现有的类型定义，你必须从 ***定义*** 选项列表选择它然后在上下文菜单中选择 *[删除]* 选项

5.4.2 结构体

下面的段落显示了如何管理结构体数据类型。

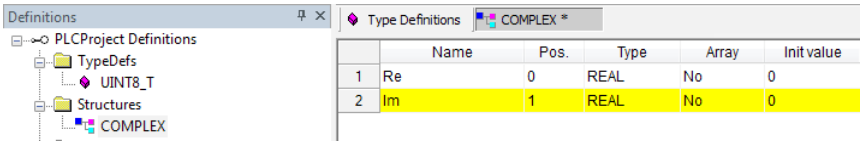
5.4.2.1 创建一个新的结构体

为了创建一个结构体数据类型，请在 ***定义*** 选线中选择 ***结构体*** 文件夹项，然后选择上下文菜单中的 *[新建结构体]* 选项。
LogicLab将弹出一个对话框，您必须指定新的结构体的名称以及其描述信息，然后确认该操作。



5.4.2.2 编辑结构体

为了编辑现有的结构体数据类型，请从 ***定义*** 选项列表中双击将其打开编辑器，在其中您可以修改它的定义和结构体中的元素。



5.4.2.3 删除结构体

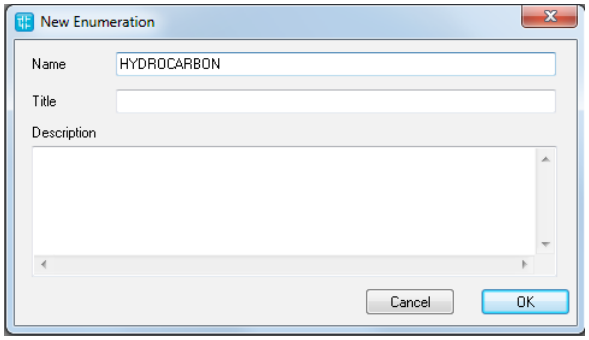
为了删除现有的结构体数据类型，请从 ***定义*** 选项列表中择它并在上下文菜单中选择 *[删除]* 选项。

5.4.3 枚举

下面的段落显示了如何管理枚举数据类型。

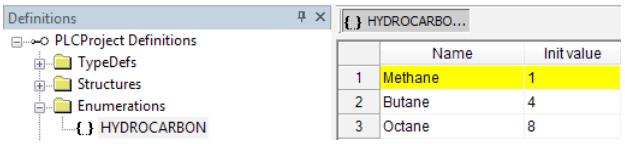
5.4.3.1 创建一个新的枚举

为了创建一个枚举数据类型，请在 ***定义*** 选线中选择 ***枚举*** 文件夹项，然后选择上下文菜单中的 *[新建枚举]* 选项。
LogicLab将弹出一个对话框，您必须指定新的枚举的名称以及其描述信息，然后确认该操作。



5.4.3.2 编辑枚举

为了编辑现有的枚举数据类型，请从 ***定义*** 选项列表中双击将其打开编辑器，在其中您可以修改它的定义和元素的初始值。



5.4.3.3 删除枚举

为了删除现有的枚举数据类型，请从 ***定义*** 选项列表中择它并在上下文菜单中选择 *[删除]* 选项。

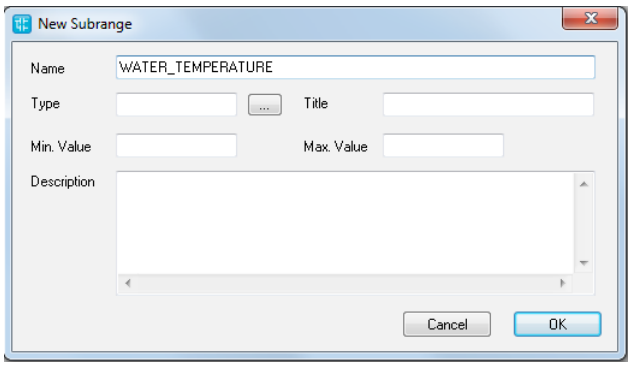
5.4.4 子范围

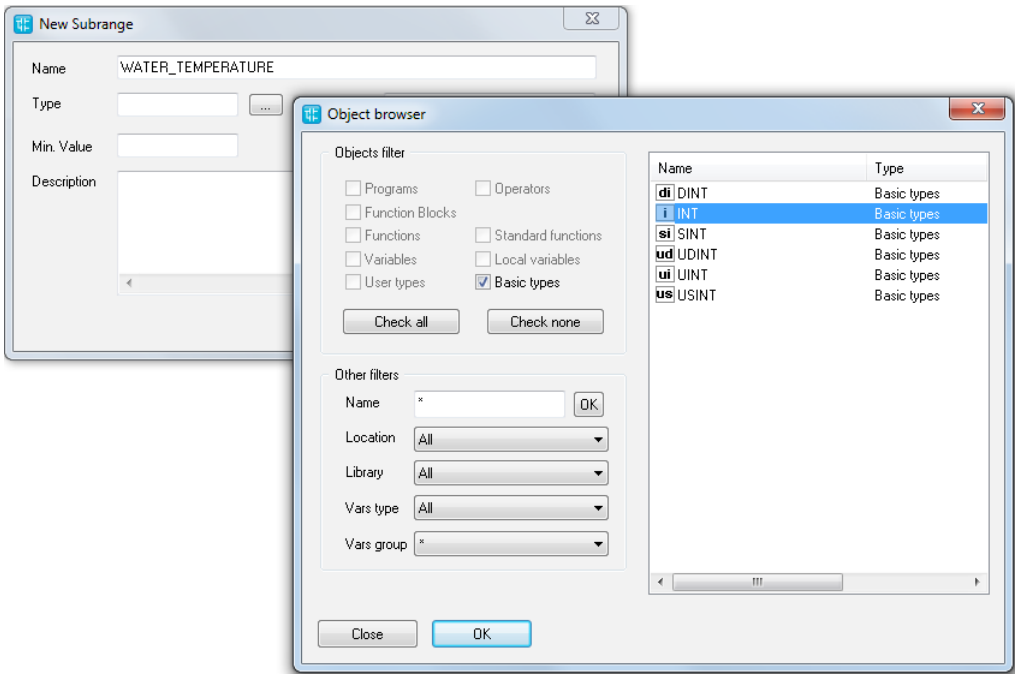
下面的段落显示了如何管理子范围数据类型。

5.4.4.1 创建一个新的子范围

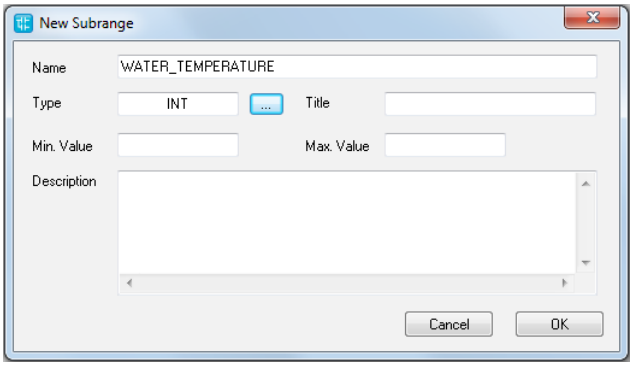
为了创建一个子范围数据类型，请在 ***定义*** 选线中选择 ***子范围*** 文件夹项，然后选择上下文菜单中的 *[新子范围]* 选项。

LogicLab将弹出一个对话框，您必须在其中指定新的子范围的名称并选择其基本类型。



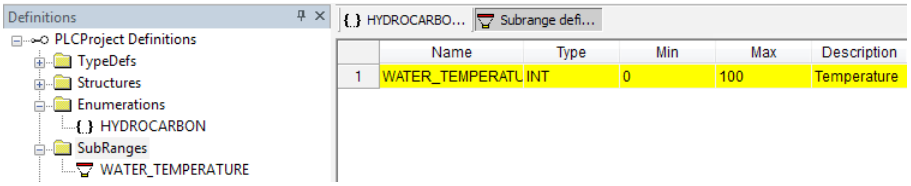


输入子范围的最小值和最大值以及您的描述信息（可选），并确认操作。



5.4.4.2 编辑子范围

为了编辑现有的子范围数据类型，请从 ***定义*** 选项列表中双击将其打开编辑器，在其中您可以修改它的定义。



5.4.4.3 删除子范围

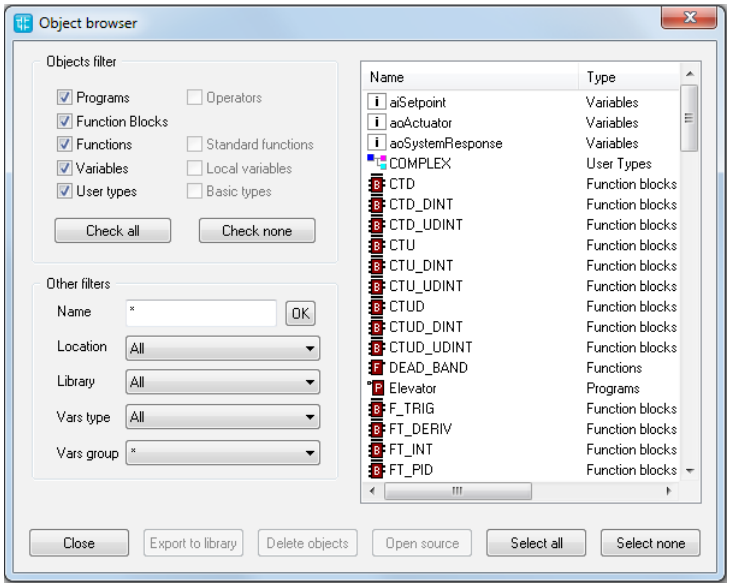
为了删除现有的子范围数据类型，请从 ***定义*** 选项列表中择它并在上下文菜单中选择 *[删除]* 选项

5.5 浏览项目

随着应用的复杂项目可能会越来越大，因此LogicLab提供了两种工具来搜索项目中的对象：***对象浏览器*** 中的 ***在项目功能查找*** 功能。

5.5.1 对象浏览器

LogicLab提供一个用于浏览项目的对象的实用工具：***对象浏览器***。



此工具取决于上下文环境，这意味可以选择的对象类型以及在不同的上下文环境中对象可用的操作是不同的。

可以通过以下这三个主要方式打开 ***对象浏览器***:

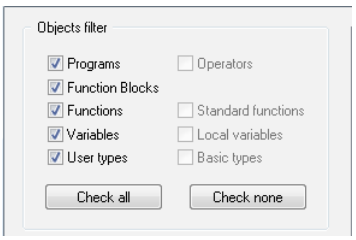
- ***浏览器*** 模式。
- ***导入对象*** 模式。
- ***选择对象*** 模式。

在三种模式下，使用对象浏览器的方式是相同，接下来将详细介绍。

5.5.1.1 常用功能和对象浏览器的使用

本节将介绍 ***对象浏览器*** 的功能和用法，这些功能和用法在三种模式下都是相同的。

对象过滤器



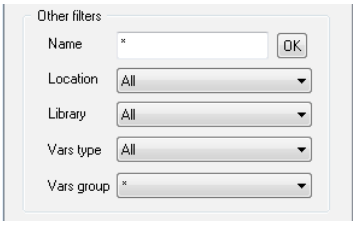
这是 ***对象浏览器*** 的主要功能。用户可以在此检索到可用的对象单元。

在此示例子中，选择了***程序***、***功能***、***功能块***，因此该类型的对象将通过过滤器显示在对象列表中。用户可以选择 ***变量*** 和 ***用户类型*** 的对象，但该类型的对象当前未显示在对象列表中。

用户无法选择过滤 ***运算符***、***标准函数***、***局部变量*** 和 ***基本类型*** （由于上下文的原因），从而无法检索浏览。

用户还可以单击 ***检查全部*** 选项从而一次选择所有可用的对象，或可以点击 ***不检查*** 选项从而一次取消选择的全部对象。

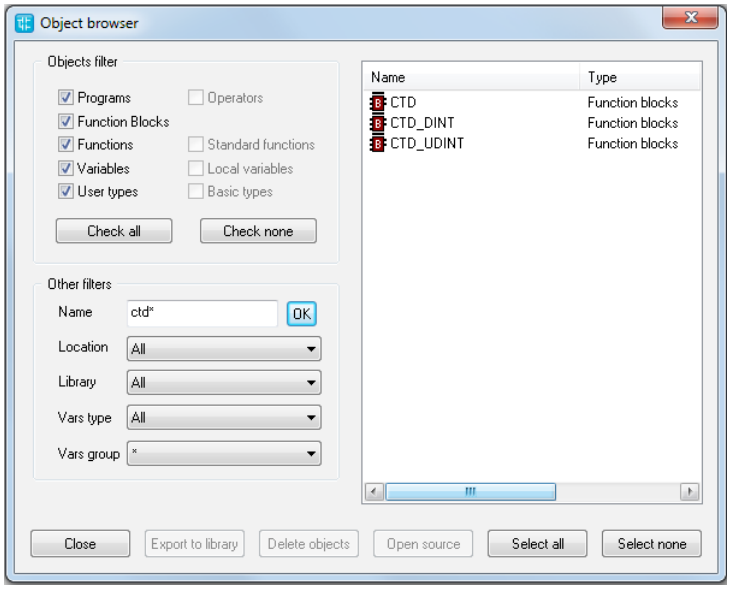
其它过滤器



选定的对象可以按名称，符号位置，具体的库和变量类型进行过滤。

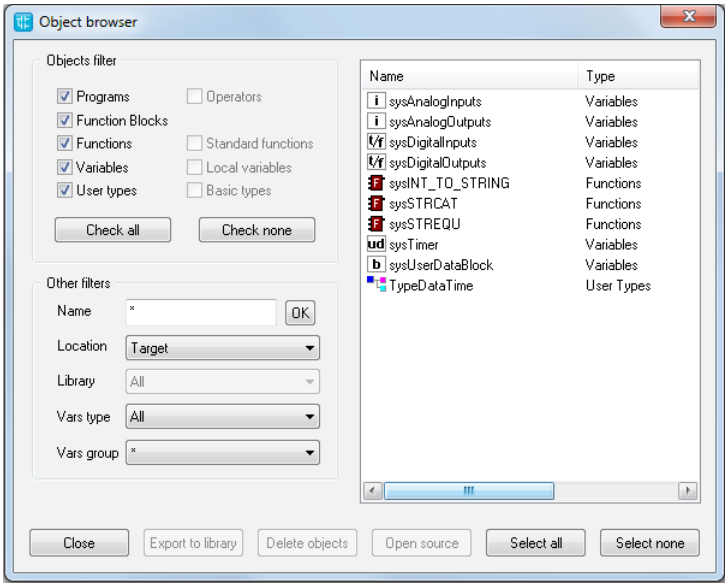
添加过滤器进行设置后立即生效。

名称	
功能	根据对象名称进行过滤。
可设定值	所有字符串。
使用	输入一个字符串将显示其名称与该字符串匹配的对象。如果您希望显示所有包含此字符串的对象名称，请在 ***名称*** 文本框中使用 * 通配符。如果要禁用此过滤器，请键入 * 符号。点击编辑框中 ***输入*** 按钮，或者编辑框旁边的 ***确认*** 按钮以应用此过滤器。
适用于	所有对象类型。

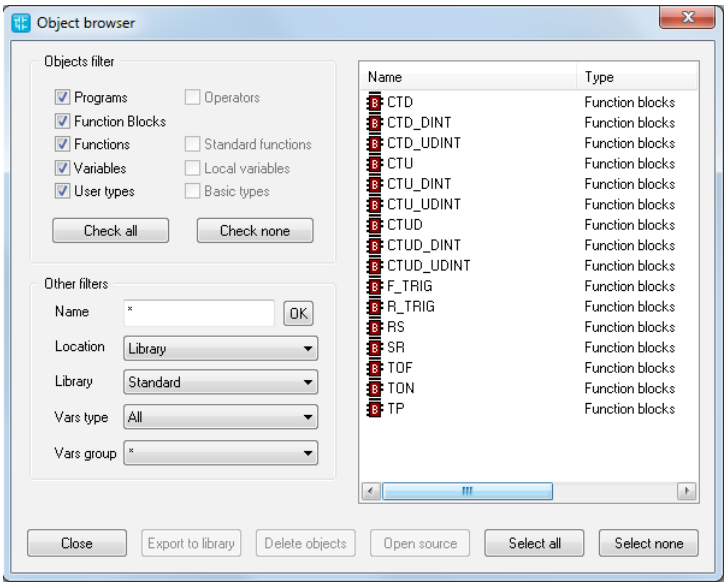


符号位置	
功能	根据对象的位置过滤对象。
可设定值	所有、项目、目标、库、辅助资源。

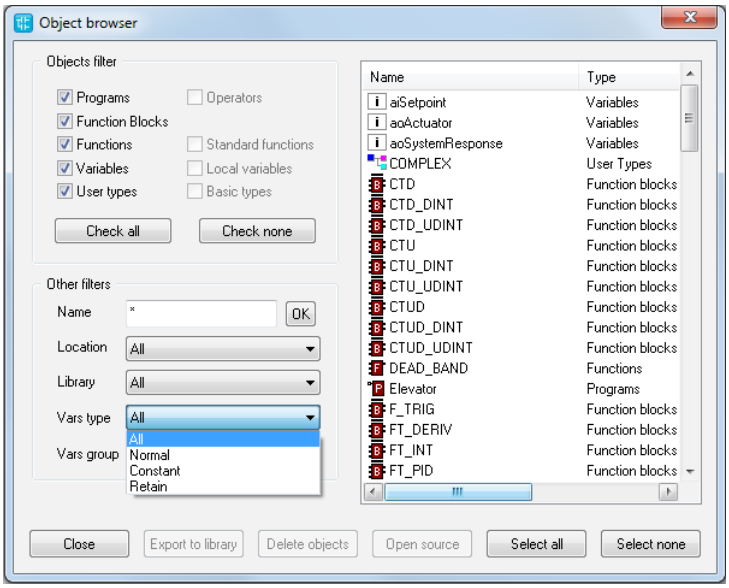
使用	所有 = 禁止该过滤器 项目 = 在LogicLab项目声明对象 目标 = 固件对象 库 = 库中包含的对象。此时，请使用如下所述的库过滤器。 辅助资源 = 仅显示辅助资源。
适用于	所有对象类型。



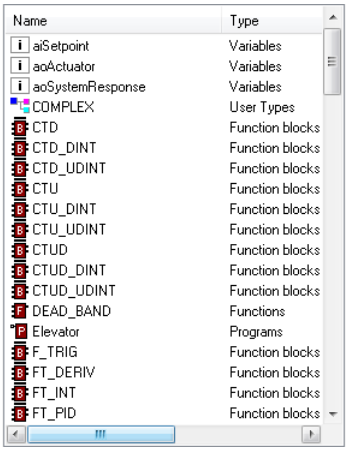
库	
功能	可对库中包含的对象的查询。只有当 ***符号位置*** 过滤器设置为 ***库*** 时，此控件才可被使用。
可设定值	全部、库名称1、库名称2 .....
使用	全部 = 显示包含在库中所有对象。 库名称N = 仅显示名为librarynameN库中包含的对象。
适用于	所有对象类型。



变量类型	
功能	根据它们的类型来过滤器全局变量和系统变量（也称为固件变量）。
可设定的值	全部、普通、常量、保持型
使用	全部 = 显示所有的全局和系统变量。 普通 = 只显示普通的全局变量。 常量 = 只显示常数。 保持型 = 只显示保持型变量。
适用于	变量。



对象列表



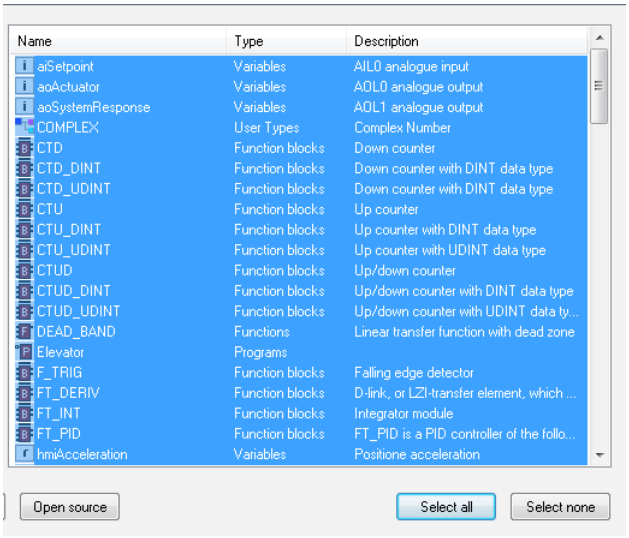
对象列表 显示了所有过滤的对象。通过单击列标题，可以按升序或降序对列表进行排序。因此，可以按名称、类型或描述说明进行排序。

用户可以通过双击某个对象执行默认相关联的操作（这个动作和执行 ***确认***、***导入对象*** 或 ***打开资源*** 等动作是类似的）。

可选择多个对象，此时，***全选*** 、***不选*** 按钮将会可见。

点击 ***全选*** 按钮选择所有对象。点击 ***不选*** 按钮则取消选择所有对象。

在列表中需要至少选择一个对象。



调整

可以调整窗口的大小，光标移致对话框的边界可调整窗口的大小。当重新打开后，***对象浏览器*** 对话框的大小将保持上一次的设定值。

5.5.1.2 对象浏览器的使用

为了使用对象浏览器查看项目的元素，请选择菜单中 *项目>对象浏览器* 选项。 。

适用对象

在这种模式下，您可以列出以下类型的对象：

- 程序
- 功能块
- 功能
- 变量
- 用户类型

这些项目可以在 ***对象过滤器*** 部分中选中或取消选中，以显示或隐藏列表中所选择的类型的对象。

在此处无法查看其他类型的对象（比如：运算符，标准函数，局部变量，基本类型等），因此它们在这里是不可选的。

具体操作

该模式下允许的操作包括：

打开资源（双击对象即默认打开资源）	
功能	打开由对象的编辑器，并显示相关的源代码。
使用	如果对象是程序、功能或功能块，则此按钮将会打开相关的源代码编辑器。 如果对象是一个变量，则此按钮将会打开变量编辑器。 选择要打开其编辑器的对象，然后单击 ***打开资源*** 按钮即可。

导出到库	
功能	导出对象到库。
使用	选择要导出的对象，然后点击 ***导出到库*** 按钮。

删除对象	
功能	允许您删除对象。
使用	选择要删除的对象，然后点击 ***删除对象*** 按钮。

多选功能

此模式下允许多选，***全选*** 和 ***不选*** 按钮可见。

5.5.1.3 使用对象浏览器导入

对象浏览器还可以将所需的外部库中的对象导入到项目中。

使用对象浏览器将外部库导入到项目中，请选择菜单中 *项目>从库导入对象* 选项。

适用对象

在这种模式下，你可以列出这些类型的对象：

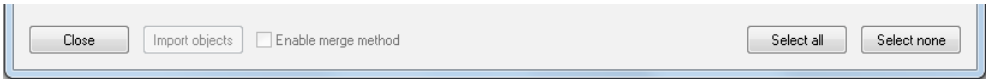
- 程序
- 功能块
- 功能

- 变量
- 用户类型

这些对象可以在 ***对象过滤器*** 中选中或是取消选中，以显示或隐藏在列表中所选择类型的对象。

无法导入其他类型的对象（运算符，标准函数，局部变量，基本类型），因此它们是无法被选中的。

具体操作



导入对象 是此模式下唯一支持的操作。通过点击 ***导入对象*** 按钮或双击列表中的一个对象，可以导入所选的对象。

多选择

此模式下允许多选，***全选*** 和 ***不选*** 按钮可见。

5.5.1.4 使用对象浏览器进行对象选择

选择一个的PLC对象后，可以通过对象浏览器对话框进行很多操作。比如，可以使用对象浏览器来选择要添加到任务的程序、选择变量的类型、选择项目中一个对象等操作。

适用对象

可选的对象取决于上下文环境，例如，在任务操作的程序分配过程中，唯一可用的对象是程序对象。

默认情况下，不是所有的对象都可被选中操作。

具体操作

在这种模式下，可以通过双击列表中或单击 ***确定*** 按钮来选择单个对象，然后会自动关闭对话框。

多选择

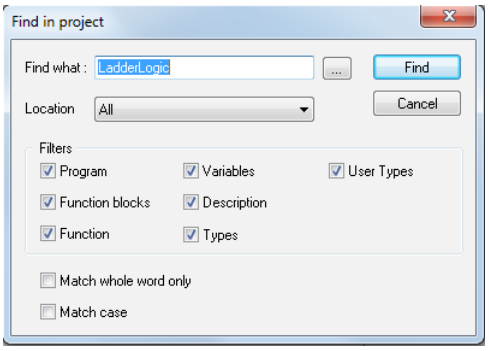
此模式下不允许多选，***全选*** 和 ***不选*** 按钮不可见。

5.5.2 项目中搜索

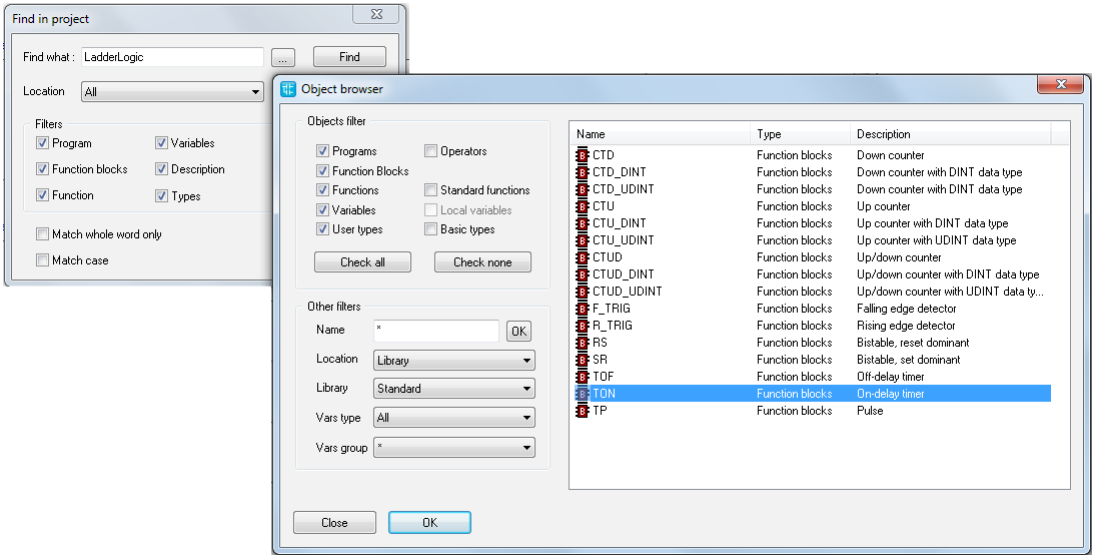
项目中搜索 命令将检索项目中指定字符串的所有实例

为了使用此功能，请在菜单中选择 *编辑>项目中搜索* 选项。

LogicLab将会弹出下面对话框：

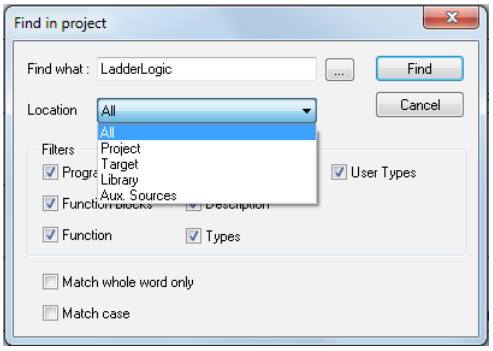


1) 在查找内容文本框中，输入要搜索对象的名称。



也可以通过单击文本框右侧的 ***浏览*** 按钮，然后从所有现有项目的列表中选择对象的名称。

2) 选择在 ***位置*** 组合框中列出的值之一，选择要查找对象的位置。

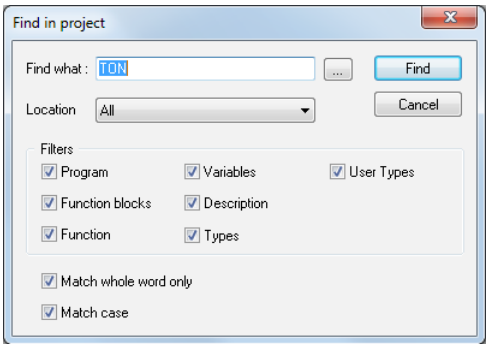


3) ***过滤器*** 选项中包含7个复选框，选中对应的复选框即表示在其所代表的的范围内进行字符串搜索。

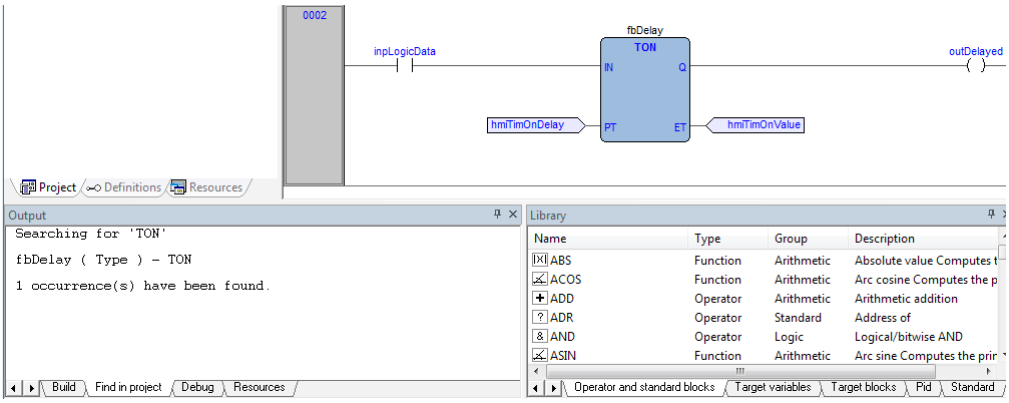
4) 只有当您希望将字符串进行全字符匹配时，勾选 ***全字匹配*** 。

5) 如果你希望搜索区分大小写，请勾选 ***区分大小写*** 。

6) 点击 ***查找*** 开始搜索，否则单击 ***取消*** 放弃操作。



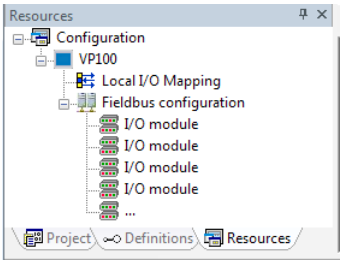
结果将打印在 ***输出窗口*** 的 ***项目中搜索*** 选项卡下。



5.6 LogicLab拓展

LogicLab的 *工作区窗口* 包含的 ***资源面板*** 其内容完全取决于与IDE连接的目标设备。

如果资源面板可见，您可以通过其访问一些目标设备相关的附加功能（如：配置元素、模式、向导等）。

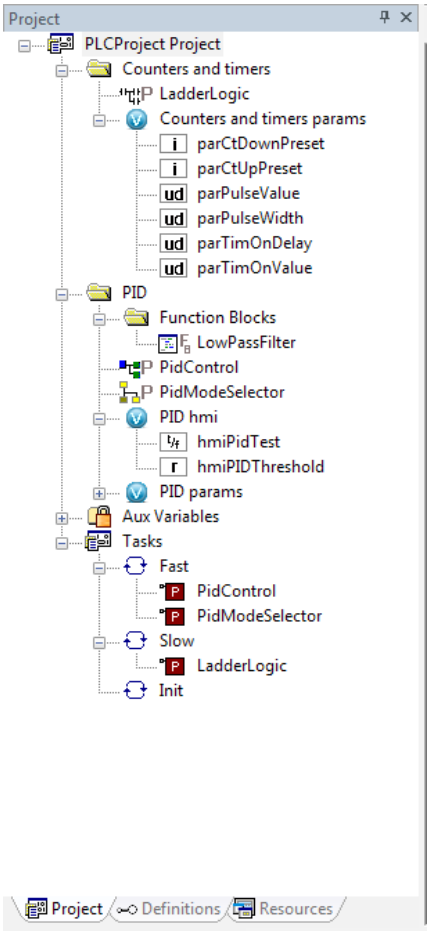


有关这些功能的信息可以在单独的文档中找到：有关详情信息，请咨询您的硬件供应商。

5.7 自定义工作空间

自定义工作空间功能使您根据您的需求组织项目树，以提高项目的管理的效率。

自定义工作区的操作不会影响各个单元的之间的逻辑：创建和编辑这些单元不会对PLC上的代码有任何影响。



5.7.1 启用自定义工作区到现有项目

要启用这个功能，请在 *项目>选项* 选择此功能，启用此功能后请重新加载项目（详细信息，请参见4.6.4节）

默认情况下，此功能是根据目标设备不同来决定是否启用。

5.7.2 工作区迁移

每当切换此功能时，LogicLab都会尝试以下逻辑来重新排序工作区，以保持用户自定义的设置：

默认（旧）工作空间转换为自定义（新）工作空间

默认自带的逻辑单元（例如功能块文件夹）将被动态转换成自定义设置中的名称。默认自带的全局变量名称（例映射变量）将被动态转换为自定义设置中的名称。不属于任何组的全局变量将被分为一个叫 ***未分组全局变量*** 的新组中。

自定义（新）工作空间转换为默认（旧）工作空间

所有的自定义单位将被删除，所有POU和全局变量将被还原为默认设定（例如：功能块文件夹，映射变量）。

5.7.3 自定工作区基本单位

在新的自定义的工作空间，您可以使用两种不同的主要逻辑单元进行工作：

- ***文件夹***：这是一个可选的逻辑单元，可以包含的POU、文件夹（文件夹支持嵌套）和全局变量组。

- ***全局变量组***：此逻辑单元只能包含全局变量。为了创建一个全局变量，您需要在自定义工作空间先创建一个全局变量组。

5.7.4 定制用户工作空间

通过以下对工作控制的各种操作，您可以更好得组织项目。

创建文件夹

如果你需要创建一个文件夹，请选择项目树根节点或者一个现有的文件夹，然后选择上下文菜单中的 *[添加>新文件夹]* 选项。

此操作增加了一个新的自定义文件夹（默认命名为 ***新文件夹*** ），您可以对它重命名。

创建一个全局变量组

如果你需要创建一个全局变量组，请选择项目树根节点或者一个现有的文件夹，然后选择上下文菜单中的 *[添加>新的全局变量组]* 选项。

此操作增加了一个新的全局变量组（默认命名为 ***新全局变量组*** ），您可以对它重命名。

重命名（文件夹或全局变量组）

如果你需要重命名一个文件夹或全局变量组，请选择它，然后选择上下文菜单中的 *[重命名]* 选项。

此操作使得您可以对此单位重新命名。

删除（文件夹或全局变量组）

如果你需要删除一个文件夹或全局变量组，请选择它，然后选择上下文菜单中 *[删除]* 选项。

如果该单位包含子元素，系统将提示您以下三种选项：

1) 删除所有子元素（这可能会影响PLC工程）。
2) 不删除子元素，他们会在项目上向上移动。
3) 取消操作，什么都不做。

导出所有元素到库

为了导出一个全局变量组或文件夹中的所有元素，请选择它，然后选择上下文菜单中 *[将所有子项导出到库]* 选项。

通过此操作可以将所选项目的所有子元素递归导出到一个库（更多详细信息，参见4.8.2节）

移动单元

您只需要将需要移动的单元拖放到项目树的其他位置即可重新安排项目工作区。如果移动父项，则按照原始结构自动所有子项。

您也可以从项目树中（单选）和变量表格中（单选或多选）移动变量。（有关变量的编辑器的更多信息，请参见 6.6节 ）。

5.7.5 工作空间元素的限制

工作区的一些元件是固定的，不能进行自定义。它们是由LogicLab自动生成，并不允许进行任何特殊的自定义操作。

根项目元素

您不能移动、重命名和删除此元素。它可以包含自定义单元为子元素。

POU子元素

这些元素是根据它们属于POU的结构生成的。您不能从项目树中移动、重命名和直接删除这些元素。有关的POU的更多详细信息（请参见5.1节）。

SFC子元素

这些元素同样遵循上述规定，特别地，对SFC子节点的动作和转换元素同样不允许重命名或删除操作。有关的SFC的更多详细信息（请参见6.5节）.

辅助变量元素

您不能移动、重命名和删除此元素及其子元素。它们由LogicLab自动生成。

任务元素

您不能移动、重命名和删除这些元素。它们由LogicLab自动生成。有关SFC语言的详细信息（请参见5.3节）。

6. 编辑源代码

PLC编辑器

LogicLab包含五种源代码编辑器，它支持IEC 61131-3标准的五种编程语言：指令表（IL），结构化文本（ST），梯形图（LD），功能块图（FBD），和顺序功能图（ SFC）。

此外，LogicLab具有网格状编辑器，支持用户自定义变量。

所有的编辑器（图形和文本编辑器）均支持工具提示。通过启用它们（请参阅3.6.1.3节）用户在其上移动鼠标LogicLab将显示一些有关的符号信息。

本章重点介绍这些编辑器。

6.1 指令表（IL）编辑器

```
0001     MUL sysIq
0002     SHR 16#04
0003     ADD addIqSq
0004
0005     MUL sysIq
0006     SHR 16#04
0007     ADD addIqSq
```

IL编辑器允许您使用IL（即指令表）编写和修改的POU。

IL（即指令表）是符合IEC标准的语言之一。

6.1.1 编辑功能

IL编辑器具有大多数在Windows平台上运行的编译器所共有的功能，即：

- 文本选择。
- *编辑>剪切*
- *编辑>复制*
- *编辑>粘贴*
- *编辑>替换*
- 拖放选中的文本

6.1.2 参考PLC对象

您如果需要在IL代码中添加到您现有PLC对象的引用，则有两个选择：

- 您可以直接输入PLC对象的名称。
- 您可以将其拖动到一个合适的位置。例如，可以从 ***工作区*** 窗口获取全局变量，可以从 ***库*** 窗口中拖动标准运算符和功能/功能块到目标代码位置，局部变量可以从 ***本地变量编辑器*** 中来选择。

6.1.3 自动错误定位

IL编辑器还会自动显示编译器错误的位置。双击在输出栏中对应的错误行，即可以定位到具体的编译错误位置上。

6.1.4 书签

您可以设置书签以标记源文件中经常访问的地方。设置标签后，您可以使用键盘命令将光标移至该书签处。当您不再需要时，您可以删除书签。

6.1.4.1 设置书签

将插入点移动到要设置书签的行，然后按 ***Ctrl+ F2***。该行的边缘处被淡蓝色圆圈标记。

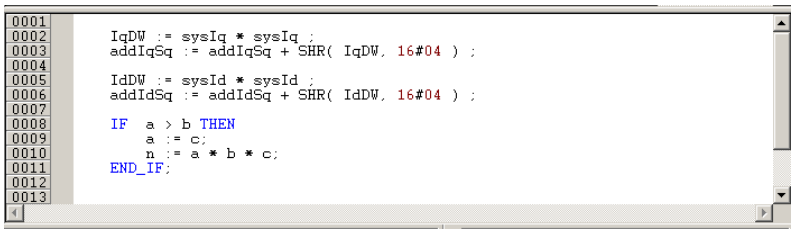
6.1.4.2 跳跃到书签

反复按F2，直到达到目标行。

6.1.4.3 删除书签

将光标移动到在包含书签行上的任意位置，然后按下 ***Ctrl+ F2***。

6.2 结构化文本（ST）编辑器



ST编辑器允许您使用ST（即结构化文本）编写和修改的POU。

ST（即结构化文本）是符合IEC标准的语言之一。

6.2.1 创建和编辑ST对象

请参阅创建和编辑的POU（请参见5.1.2和 5.1.2节）。

6.2.2 编辑功能

ST编辑器具有大多数在Windows平台上运行的编译器所共有的功能，即：

- 文本选择
- *编辑>剪切*
- *编辑>复制*
- *编辑>粘贴*
- *编辑>替换*

- 拖放选中的文本

6.2.3 参考PLC对象

您如果需要在ST代码中添加到您现有PLC对象的引用，则有两个选择：

- 您可以直接输入PLC对象的名称。
- 您可以将其拖动到一个合适的位置。例如，可以从 ***工作区*** 窗口获取全局变量，可以从 ***库*** 窗口中拖动标准运算符和功能/功能块到目标代码位置，局部变量可以从 ***本地变量编辑器*** 中来选择。

6.2.4 自动错误位置

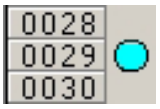
ST编辑器还会自动显示编译器错误的位置。双击在输出栏中对应的错误行，即可以定位到具体的编译错误位置上。

6.2.5 书签

您可以设置书签，以在源文件中经常访问的线条标示。一旦书本 - 标志设置，你可以使用键盘命令移动到它。您可以删除书签，当你不再需要它。

6.2.5.1 设置书签

将插入点移动到要设置书签的行，然后按 ***Ctrl+ F2***。该行的边缘处被淡蓝色圆圈标记。



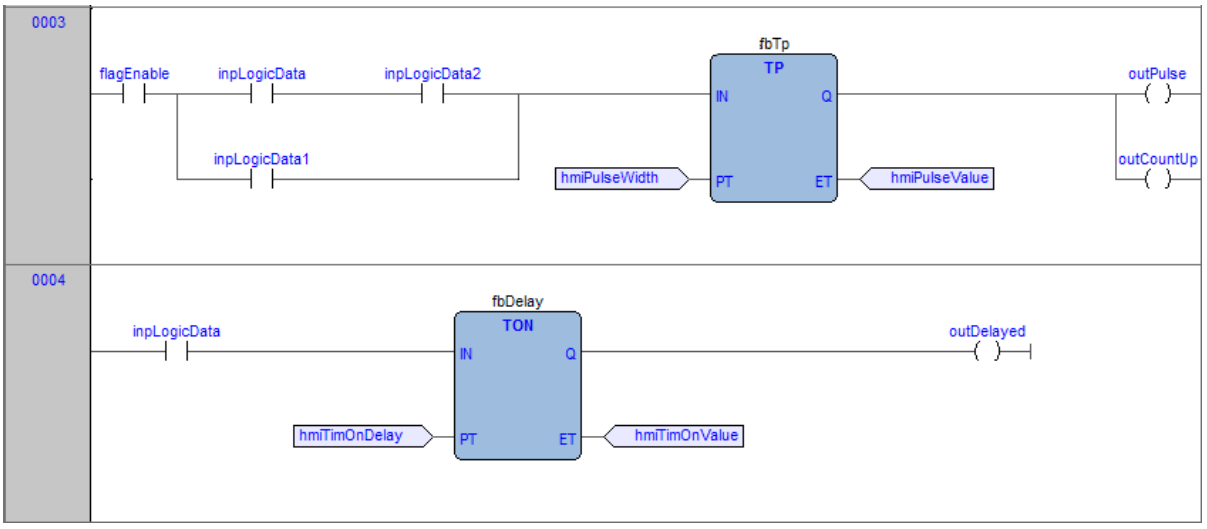
6.2.5.2 跳跃到书签

反复按F2，直到达到目标行。

6.2.5.3 删除书签

将光标移动到在包含书签行上的任意位置，然后按下 ***Ctrl+ F2***。

6.3 梯形图（LD）编辑器



LD编辑器允许您使用LD（即梯形图）编写和修改的POU。

LD（即梯形图）是符合IEC标准的语言之一

6.3.1 创建新的LD

请参阅创建和编辑的POU（请参见5.1.2和 5.1.2节）。

6.3.2 添加/删除网络

由LD编码的POU由一系列的网络组成。网络被定义为互连图形元素的最大集合。每个网络的上限和下限由两条直线确定，而每个网络的左侧由包含网络编号的灰度凸起界定。

每个LD网络上，根据LD语言标准左右两侧表示其电源导轨。

在新的LD网络上，一条水平线连接两个电源导轨。这就是所谓的"电源链接"。在此链路中，必须放置所有的LD元件（触点、线圈和功能块）。

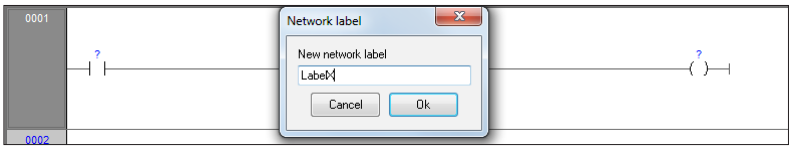
您可以在网络中执行以下操作：

- 如果要添加一个新的空白网络，请单击 *方案>网络>新建*，或者网络工具栏中的等效的按钮。
- 如果要将标签分配给选定的网络，请点击 *方案>网络>标签*，这样将会跳转到标记的网络。
- 如果要显示有助您对齐对象的背景网络，请单击 *视图>栅格*
- 如果要添加备注，请单击 *方案>对象>新备注*

6.3.3 网络标签

您可以通过跳转语句来改变网络的顺序执行，该语句将程序的网络控制权转移到网络的标号。要将标签分配给网络，请双击左侧带有网络编号的的灰色凸起按钮。

将会弹出一个对话框出现，在那里您可以输入要与所选网络关联的标签。

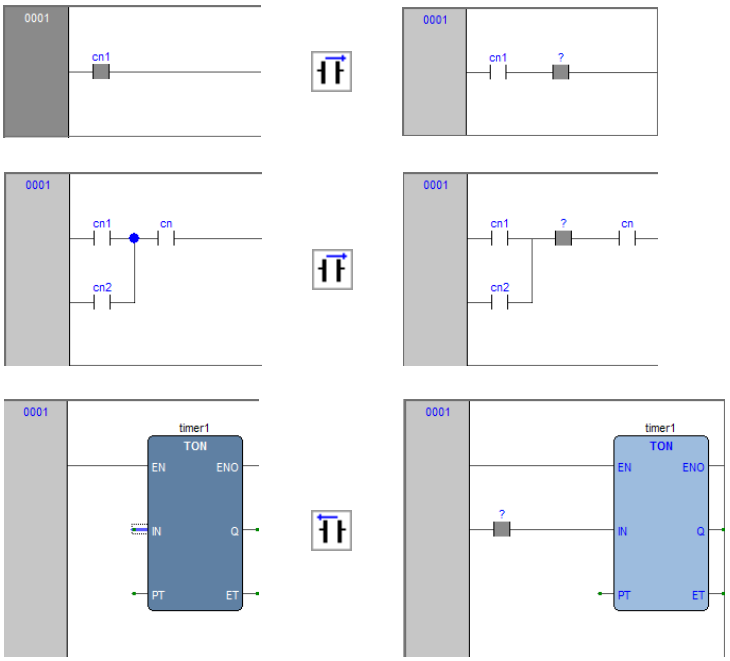


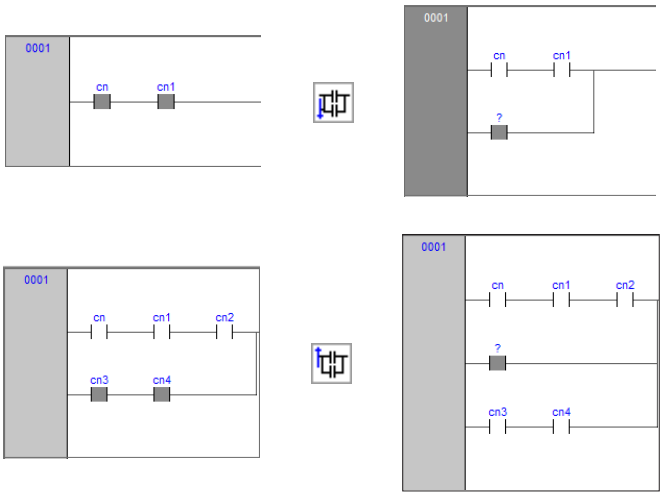
如果按 ***确定*** 键，则标签将打印在所选网络的左上角。

6.3.4 插入连接单元

要在网络上插入新的连接单元，请使用以下选项之一：

- 选择将要用作插入的触点、块、块针或连接点，通过使用 ***方案>对象>新连接单元*** ，在连接类型（串行或并行）和位置（当前所选对象之前或之后）之间插入新的连接单元。 对于串行插入，根据插入之前选择的元素，新触点将插入到所选触点/块的左侧或右侧，或插入所选连接的中间。 对于并行插入，可以在执行插入之前选择几个触点，新插入的连接单元将插入到所选连接单元组的上方或下方。



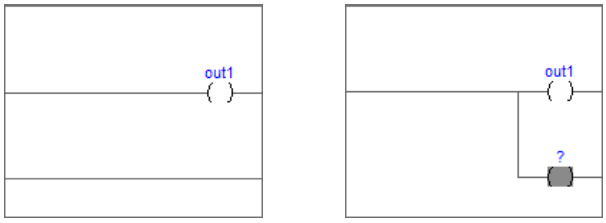


- 将布尔变量拖到对象上的所需位置。 例如，可以从 *工作区* 窗口中获取全局变量，也可以从本地变量编辑器中选择本地变量。此外，通过拖放插入的连接单元将始终串联插入在目标对象之后。

6.3.5 插入线圈

如果要在网络上插入新的线圈，请使用以下选项之一：

- 点击 *方案>对象>新建>线圈* 。新建的线圈将被插入并链接到右侧电源导轨。如果网络中存在其他线圈、返回或跳转，则新的线圈将与现有的线圈并行添加。



- 在网络上的现有的输出（线圈、返回、跳转）上拖动一个布尔类型的变量到网络上。例如，**可以从 *工作区* 窗口中获取**全局变量，**也可以从本地变量编辑器中选择**局部变量。

6.3.6 插入块

要在网络上插入块，请使用以下选项之一：

- 选择一个连接点、连接或块，然后单击 *方案>对象>新建>块* ，将会弹出一个对话框，列出该项目的所有对象，然后您需要从列表中选择一项。。
- 将选定的对象（从 *工作区* 窗口， *库* 窗口或 ***本地变量编辑器*** ）拖放到所需的连接上。

如果对象具有至少一个BOOL输入和一个BOOL输出引脚，则它们将连接到电源链路（以后可以使用提供的命令添加EN / ENO引脚）；否则，将自动添加EN / ENO引脚。

操作符、功能和功能块只能插入到主电源链路或分支的电源链路的LD网络中（因此不能将其插入触点的并联线路中）。同时，也不能在块的并行线路中创建触点。

如果一个块具有BOOL输入引脚，则可以在其之前创建另一个触点和块的逻辑子网。 否则，您只能将变量，常量或表达式（尽管仍然可以连接到BOOL引脚）连接到非BOOL输入引脚。

6.3.7 编辑线圈和触点性能

可以通过以下操作之一来更改触点的类型（正常，否定，正，负极）或线圈（正常，否定，置位，复位，正，负极）：

- 双击元素（触点或线圈）。
- 选择元素，然后按 ***确认*** 键。
- 选择元素，弹出菜单中选择 *[属性]* 。

此时会弹出有一个对话框、从呈现的列表中选择所需元素类型，然后按 ***确认*** 。

否则，选择所需的触点或线圈，可使用LD工具栏中提供的六个的按钮或 ***方案*** 菜单中的六个命令来更改其类型。

6.3.8 编辑网络

LD编辑器具有Windows平台上运行的大多数图形应用程序所共有的功能，即：

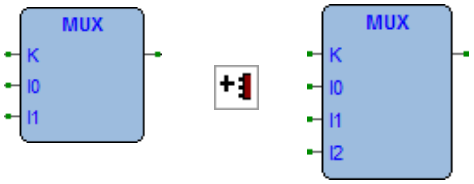
- 选择一个块。
- 通过在每个连接单元上选择Ctrl+Left按钮选择相邻的连接单元； 如果选择择跨不同并行分支，则会在选择中添加更过的连接单元。
- *编辑>剪切* ， *编辑>复制* ， *编辑>粘贴* 操作对于单个块和多个块是一样的。

- ***拖放*** 所选对象或组，将其移到当前的网络的内部或外部。

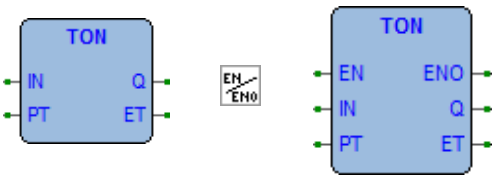
添加，移动，删除或复制/粘贴对象将自动重新规划网络对象的布局，因此，无法手动“绘制”连接线或在不对其连接到网络的情况下进行自由放置。

6.3.9 改变块的属性

- 点击 *方案>增增加引脚数* ，以增加某些运算符和功能的输入引脚数。



- 点击 *方案>启用EN / ENO引脚* ，以显示使能输入和输出引脚。



仅当所选择的块具有至少一个BOOL类型的输入和一个BOOL类型的输出时，才能删除EN/ENO引脚；否则，EN/ENO引脚将在创建块时自动添加，并且无法删除（使能EN/ENO引脚的命令将被禁用）。

如果一个块具有多个BOOL类型的输出引脚，则可以选择其中一个引脚作为块的输出信号与电源连接：选择目标输出引脚，然后单击 方案>设定输出引脚 。

点击 方案>对象>实例名称 以改变功能块实例的名称。

6.3.10 获取信息上的块

您可以通过执行以下操作，获取到您添加到LD编辑区的块的信息：

- 点击 *方案>对象>打开源代码* ，以打开块的源代码。
- 点击菜单中 *方案>对象属性* ，以查看所选择的块的特性和输入/输出管脚。

6.3.11 自动错误检索

LD编辑器还可以自动显示编译器错误的位置。要找到发生编译器错误的块，请在 ***输出*** 栏中双击相应的错误行。

6.3.12 插入变量

要将变量连接到模块的输入或输出引脚，请使用以下选项之一：

- 选择一个块的引脚，然后单击菜单中的 *方案>对象>新建>变量* 命令；然后双击新的变量对象（或按确认）并输入变量名称。
- 将选定的变量（从 ***工作区*** 窗口、 ***库*** 窗口或 ***本地变量编辑器*** 中）拖动到目标块的引脚上。

6.3.13 插入常量

为一个数值常数连接到块的输入管脚，选择销并点击 *方案>对象>新建>恒* 菜单命令；然后双击恒新转播JECT（或按Enter键），然后输入数值恒定值。

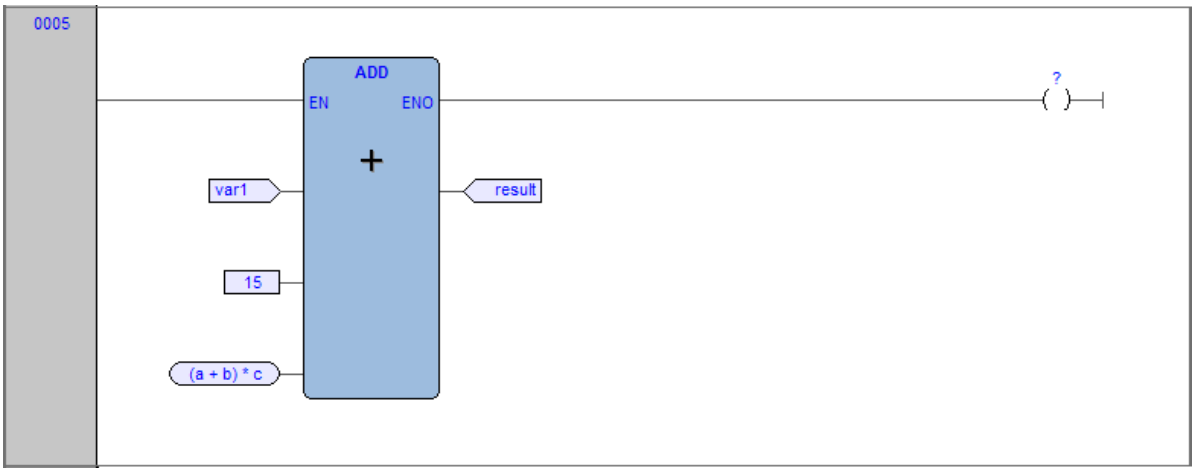
6.3.14 插入表达

如果要将常量连接到块的输入引脚，请选择该引脚，然后单击菜单 *方案>对象>新建>常量* 命令； 然后双击新新的常数对象（或按确认）并输入数字常数值。

(a+b)*c

TO_INT(n)

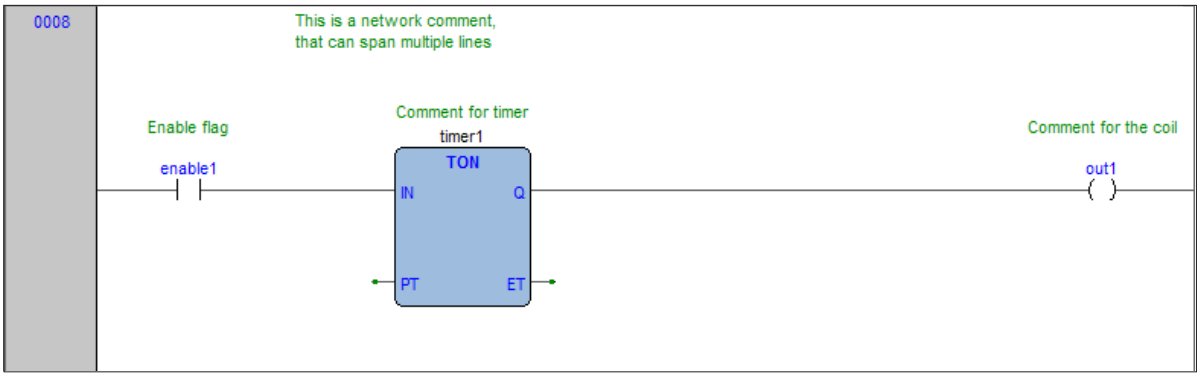
ADR(x)



6.3.15 注释

可以插入以下两种类型的注释：

- ***网络*** 注释：通过单击左侧或网格内部的标题（但不选择任何对象）来激活网络，然后单击菜单中 *方案>对象>新建>注释* 命令。 网络注释将显示在网络的顶部，如果需要，可以展开以显示注释的所有文本行。
- ***对象*** 注释：通过 *视图>显示对象注释* 中的相应菜单命令激活它们；在任何触点、功能块或线圈上方，将首先显示相关PLC变量的描述（如果存在），但是使用注释命令，您可以对其进行修改以输入特定对象注释，该注释将覆盖PLC变量描述。

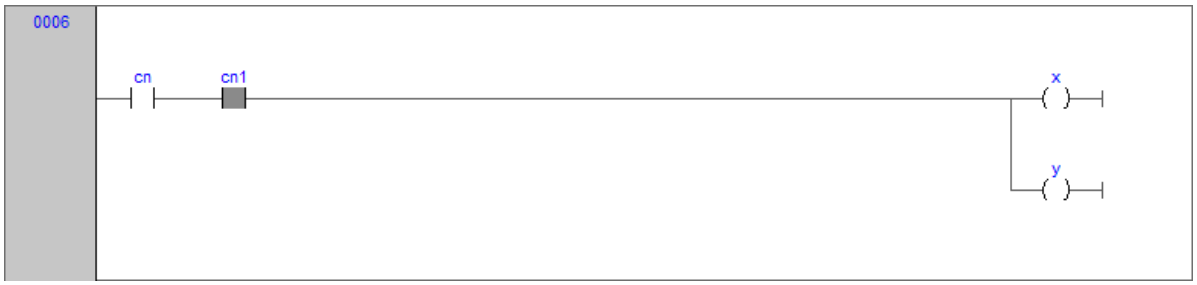


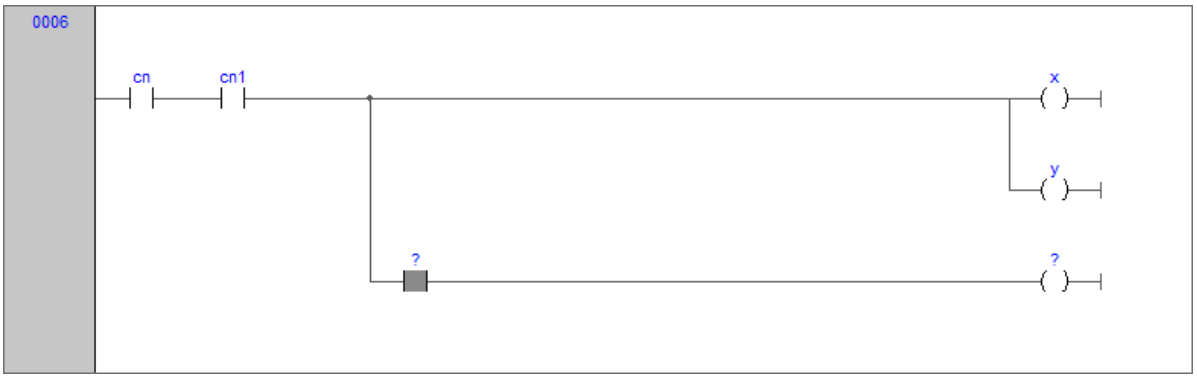
6.3.16 分支

主网络可以创建分支以创建子网络，子网络可以进一步添加分支。选择您要添加分支位置的对象，然后点击菜单中 *方案>对象>新建>分支* 命令。

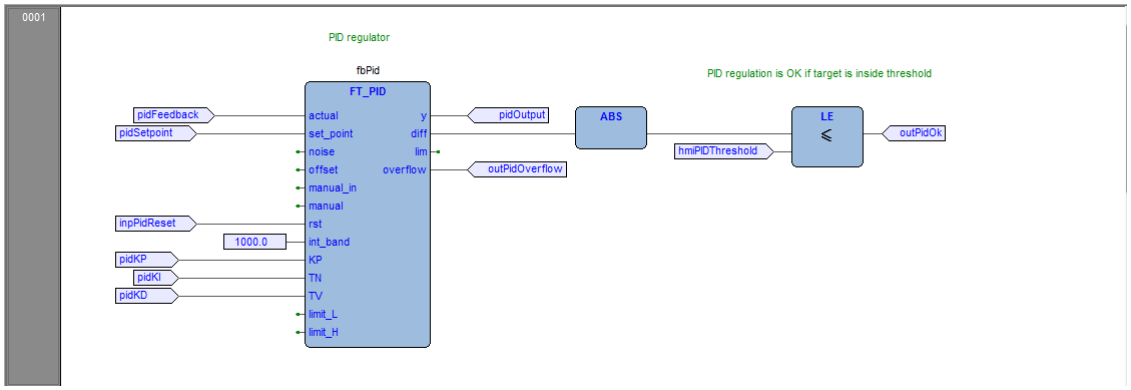
新分支的开始标记为电源线中的一个比较大的节点；删除分支上的所有对象都会删除分支本身。

在分支上选择一个对象有效地选择分支，例如，在分支上选择一个触点，然后单击 *方案>对象>新建>线圈*，则会在分支上添加线圈，而不是在主电源线上添加线圈。





6.4 功能块图（FBD）编辑器



FBD编辑器允许您使用FBD（即功能块图）编写和修改的POU。

FBD（即功能块图）是符合IEC标准的语言之一。

6.4.1 创建新的FBD

请参阅创建和编辑的POU部分（请参见5.1.1和 5.1.2）。

6.4.2 添加/删除网络

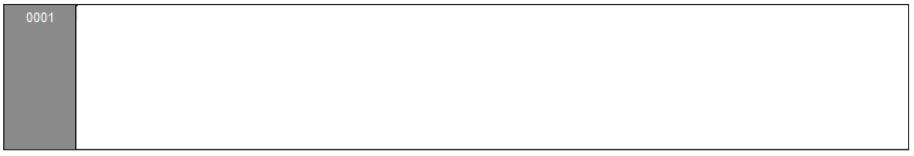
FBD编码的每个POU都由一系列网络组成。 网络被定义为互连图形元素的最大集合。 每个网络的上限和下限由两条直线固定，而每个网络的左侧由包含网络编号的灰色凸起按钮界定。

您可以在网络中执行以下操作：

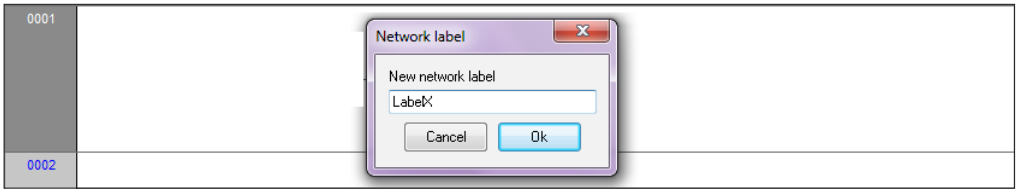
- 如果要添加一个新的空白网络，请单击 *方案\网络\新建*，或者网络工具栏中的等效的按钮
- 如果要将标签分配给选定的网络，请点击 *方案\网络\标签*，这样将会跳转到标记的网络。
- 如果要显示有助您对齐对象的背景网络，请单击 *视图\栅格*
- 如果要添加备注，请单击 *方案\对象\新备注*

6.4.3 网络标签

您可以通过跳转语句来改变网络的顺序执行，该语句将程序的网络控制权转移到网络的标号。要将标签分配给网络，请双击左侧带有网络编号的的灰色凸起按钮。



将会弹出一个对话框出现，在那里您可以输入要与所选网络关联的标签。



如果按 ***确定*** 键，则标签将打印在所选网络的左上角。

6.4.4 插入和连接块

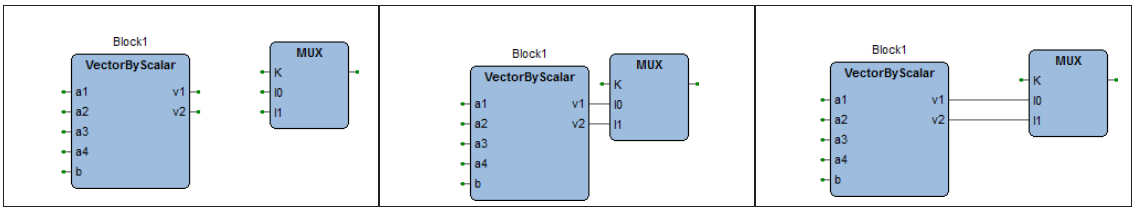
本段将说明了如何建立一个FBD网络。

通过应用以下选项，可以将块添加到空白网络：

- 点击 *方案\对象\新建\功能块* 将会在弹出的对话框中列出该项目的所有对象，然后从列表中选择一项。如果该块是一个常量、返回语句或跳转语句，则可以直接按FBD工具栏上的相关的按钮。
- 将选定的对象拖到合适的位置。例如，可以从 ***工作区*** 窗口中获取全局变量，也可以从 ***库*** 窗口中拖动标准运算符和函数，同样也可以从 ***局部变量编辑器*** 中选择局部变量。

重复上述步骤，直到添加了构成网络的所有块。

- 点击 *编辑>连接方式*，或直接按键盘上的空格键。单击源引脚，然后将鼠标拖动到目标引脚：FBD编辑器将在两者之间绘制逻辑线。
- 如果要连接两个引脚一一对应的模块，可以通过单以 *方案>自动连接*，启用自动连接模式。然后将两个块向彼此靠近拖动，以使相应的引脚重合，FBD编辑器会自动绘制逻辑线。



如果删除一个块，但是不会自动删除其连接，它们的连接将变为无效并重新绘制为红色。

请点击 *方案>删除无效连接* 。

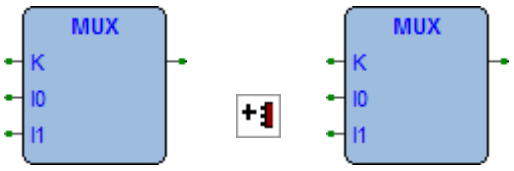
6.4.5 编辑网络

FBD编辑器具有Windows平台上运行的大多数图形应用程序所共有的功能，即：

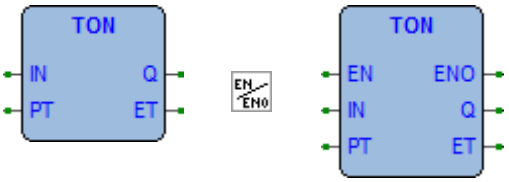
- 选择一个块。
- 通过按Ctrl+Left按钮来选择一组相邻块，可以将其绘制成一个包含选择的块的代码框架。
- *编辑>剪切* ，*编辑>复制* ，*编辑>粘贴* 操作对于单个块和多个块是一样的。
- ***拖放***

6.4.6 修改块的属性

- 点击*方案>增量引脚数* ，以增加某些运算符和功能的输入引脚数。



- 点击*方案>启用EN/ENO引脚* ，显示使能输入和输出引脚。



- 点击 *方案>对象>实例名称* ，或者单击*方案>对象属性* ，以更改功能块实例的名称。

6.4.7 获取信息上的块

您始终可以通过选择FBD代码中的块，通过执行以下操作来获得其有关信息：

- 点击*方案>对象>打开源代码* ，以打开一个块的源代码。
- 点击*方案>对象属性* ，以查看所选块的属性和输入/输出引脚。

6.4.8 自动错误检索

FBD编辑器还可以自动显示编译器错误的位置。要找到发生编译器错误的块，请在 ***输出*** 栏中双击相应的错误行。

6.5 顺序功能图（SFC）编辑器

SFC编辑器允许您使用SFC（即顺序功能图）编写和修改的POU。

SFC（即顺序功能图）是符合IEC标准的语言之一。

6.5.1 创建新的SFC

请参阅创建和编辑的POU部分（请参见5.1.1和 5.1.2）。

6.5.2 插入一个新的SFC元素

- 点击 *方案>对象>新建>步* 。
- 点击 *方案>对象>新建>转换* 。
- 点击 *方案>对象>新建>跳转* 。

在任一情况下，鼠标指针变为：

 对应于步；

 对应于转换；

 对应于跳转。

6.5.3 连接SFC元素

请按照以下步骤来连接SFC块：

- 点击 *编辑>连接方式* ，或直接按键盘上的空格键。单击源引脚，然后将鼠标拖动到目标引脚：SFC编辑器将在两者之间绘制逻辑线。
- 您也可以通过单以 *方案>自动连接* ，启用自动连接模式。然后将两个块向彼此靠近拖动，以使相应的引脚重合，SFC编辑器会自动绘制逻辑线。

6.5.4 指定动作到步

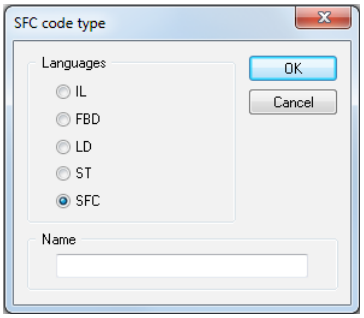
这一段解释具体的实时操作以及如何将指定动作分配给步。

6.5.4.1 写一个行动守则

要开始执行操作之前，您需要打开一个编辑器。你可以通过以下步骤实现：

- 点击 *方案>代码对象>新动作* 。
- 在 ***工作区*** 窗口中右键点击POU的名称 *[新动作]*

在任一情况下，LogicLab都会显示下面所示的一个对话框。



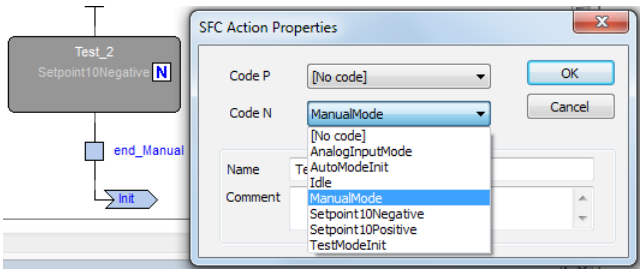
选择一种编程语言，然后在对话框底部的输入新动作的名称。 然后按 ***确定***，或点击 ***取消*** 退出。

如果按 ***确定*** ，则LogicLab会自动打开与您在上一个对话框中选择的编程语言对应的编辑器，在此您可以输入新动作的代码。

请注意，在新的动作中是不允许声明新的局部变量的，因为正在编辑的模块是原始SFC模块的组成部分，原始SFC模块中是可以声明局部变量。局部变量的范围扩展到构成SFC模块的所有动作和转换。

6.5.4.2 指定动作到步

当您编写完成代码后，双击您要为其分配新动作的步， 将弹出以下对话框。

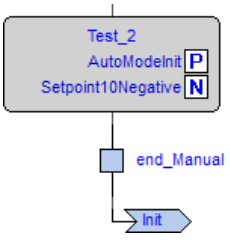


从 ***代码N*** 框中显示的列表中，选择在步骤处于活动状态时要执行的动作的名称。

您还可以从 ***代码P***（脉冲）框中显示的列表下选择每次激活该步骤后要执行的动作的名称（即无论步骤保持活动的循环数，每次激活该动作仅执行一次）， 通过按 ***确认*** 分配。

在SFC模式中，对步骤分配的动作由步骤块上的字母表示：

- 右上角用字母N表示执行的N方式的动作；
- 右下角用字母P表示执行的P方式的动作。

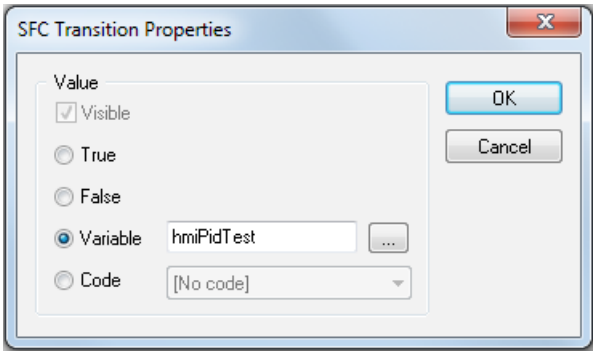


如果以后需要编辑操作的源代码，则只需双击这些字母（P/N）即可。 或者您可以在 ***工作区*** 窗口的 ***动作*** 文件夹中双击动作的名称。

6.5.5 指定常量/变量作为转换的条件

如语言参考的相关部分所述，可以通过常量、变量或一段代码来指定转换条件。本段说明了如何使用前两种方式，而下一段讨论了条件代码。

首先，双击要为其分配条件的转换。将弹出如下对话框。



如果您需要始终转换条件为真，则选择 ***真*** ；如果希望PLC程序继续执行前面的块，则选择 ***假*** 。

如果选择 ***变量*** ，则转换将取决于布尔类型变量的值。单击相应的项目符号，可以在其右侧的文本框中指定变量的名称。

同时，您还可以使用对象浏览器，可以通过按下下面显示的 ***浏览*** 按钮来调用该对象浏览器。

单击 ***确定*** 以确认或 ***取消*** 而不应用更改直接退出。

6.5.6 将条件代码分配给转换

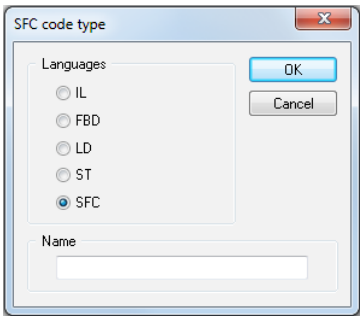
本段说明如何通过一段代码指定条件，以及如何将其分配给转换。

6.5.6.1编写条件代码

按照以下步骤之一，首先打开编辑器：

- 点击 *方案>代码对象>新转换* 。
- 在 ***工作区*** 窗口上右键点击 SFC POU的名称 *[新转换]* 。

在任一情况下，LogicLab都会显示一个对话框，如下图所示。



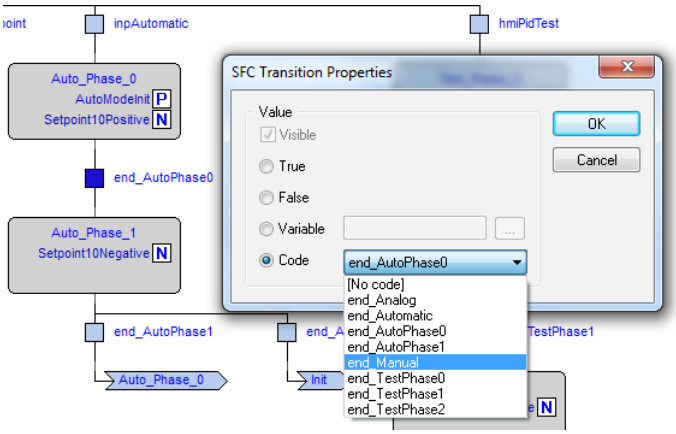
请注意，可以使用除SFC之外的任何语言编写条件。选择一种语言，然后在对话框底部的文本框中输入新条件的名称。 然后按 ***确认*** ，或单击 ***取消*** 退出。

如果按 ***确定*** ，则LogicLab会自动打开与在上一个对话框中选择的编程语言对应的编辑器，在此您可以输入新动作的代码。

请注意，在新的条件中是不允许声明新的局部变量的，因为正在编辑的模块是原始SFC模块的组成部分，原始SFC模块中是可以声明局部变量的。局部变量的范围扩展到构成SFC模块的所有动作和转换。

6.5.6.2将条件分配给转换

完成编写代码后，双击要为其分配新条件的转换。将弹出以下对话框。



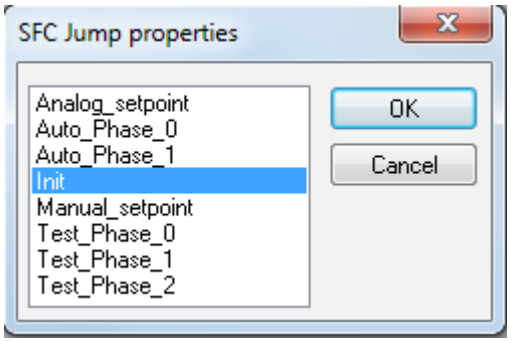
选择要分配给此步的条件的名称。 然后点击 ***确认*** 。

如果以后需要编辑此条件的源代码，则可以在 ***工作区*** 窗口的 ***转换*** 文件夹中双击转换的名称。

6.5.7 指定跳转目标

指定跳转的目标步，请在 ***图表*** 区域中双击跳转块。

将弹出如下所示的对话框，其中列出了所有现有步的名称。 选择目标步，然后按 ***确定*** 进行确认或按 ***取消*** 退出。



6.5.8 编辑SFC网络

SFC编辑器具有Windows平台上运行的大多数图形应用程序所共有的功能，即：

- 选择一个块。
- 通过按Ctrl+Left按钮来选择一组相邻块。
- *编辑>剪切*，*编辑>复制*，*编辑>粘贴* 操作对于单个块和多个块是一样的。
- ***拖放***

6.6变量编辑

LogicLab包含一个用于全局变量和局部变量的图形编辑器，提供了用于声明和编辑变量的用户友好界面：该工具负责将这些编辑器的内容转换为语法正确的IEC 61131-3源代码。

例如，以下图所示的全局变量编辑器中的内容。

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	pidKP	REAL	%MD1.0	PID	No			PID proportional gain
2	pidKI	REAL	%MD1.4	PID	No			PID integral time
3	pidSetpoint	REAL	%MB1.8	PID	No			PID setpoint (from -1 to +1)
4	pidOutput	REAL	%MD1.12	PID	No			PID output value

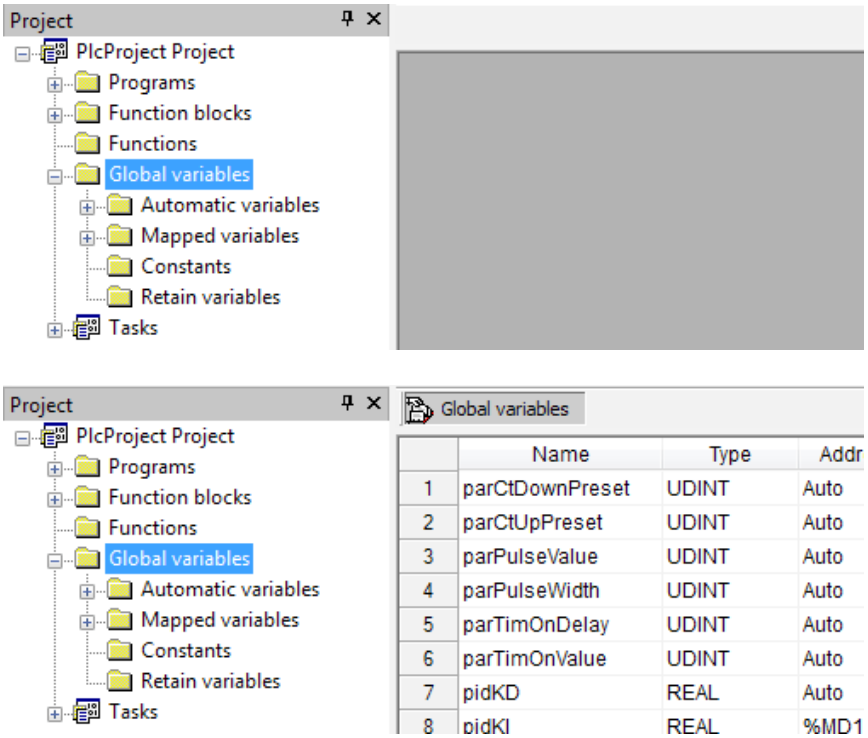
相应的源代码如下所示：

```
VAR_GLOBAL
    gA : BOOL := TRUE;
    gB : ARRAY[ 0..4 ] OF REAL;
    gC AT %MD60.20 : REAL := 1.0;
    END_VAR
    VAR_GLOBAL CONSTANT
    gD : INT := -74;
    END_VAR
```

6.6.1 打开一个变量编辑

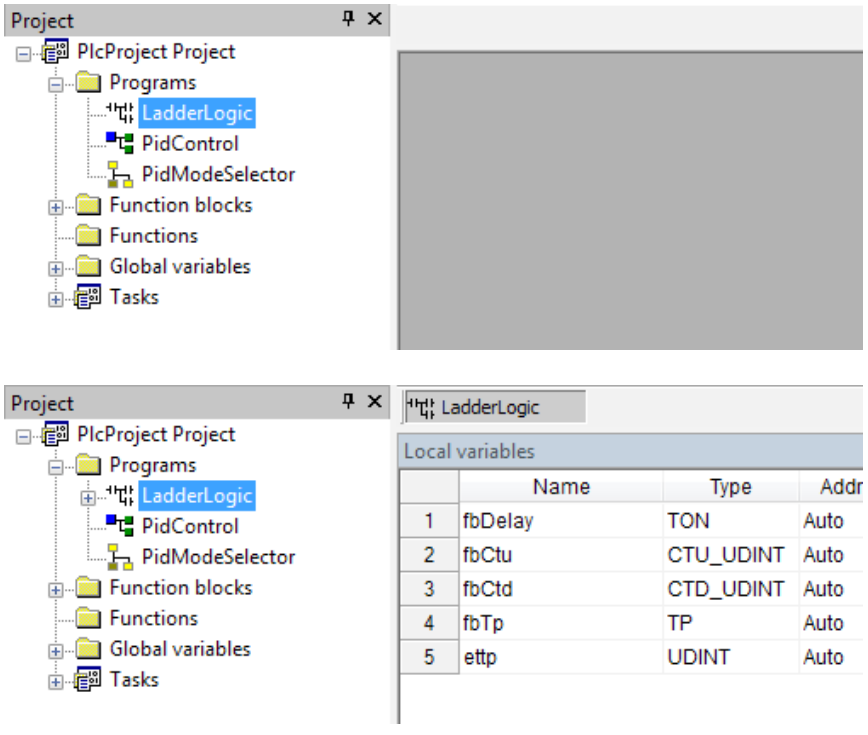
6.6.1.1 打开全局变量编辑器

为了打开全局变量编辑器，在项目树中双击 ***全局变量*** 。



6.6.1.2 打开本地变量编辑

要打开局部变量编辑器，只需打开要你希望编辑的程序组织单元的本地变量页。



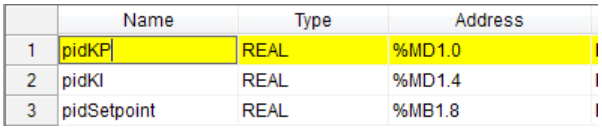
6.6.2 创建一个新变量

为了创建一个新的变量，您可以点击 *变量>插入* 。

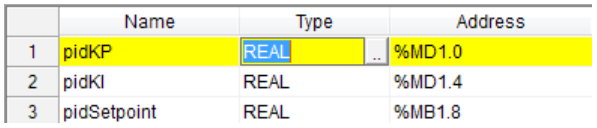
6.6.3 编辑变量

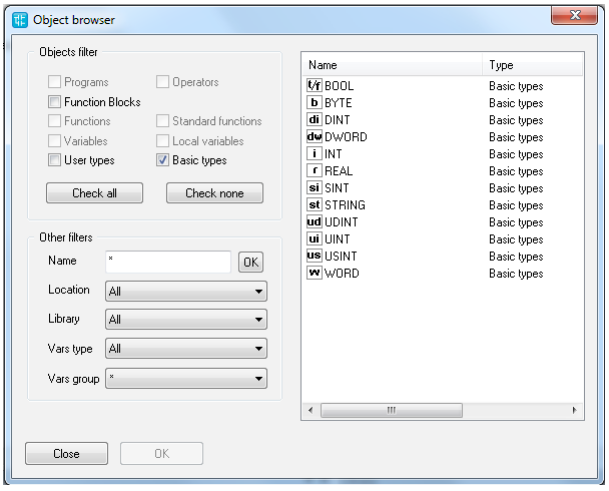
请按照以下过程在变量编辑器中编辑变量的声明（以下所有步骤都是可选的，通常在编辑变量时将跳过其中的大多数步骤）：

1) 在相应的单元格中编辑变量的名称。

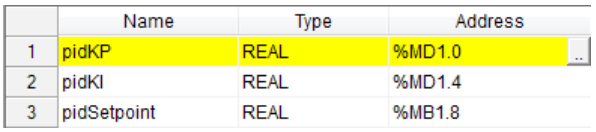


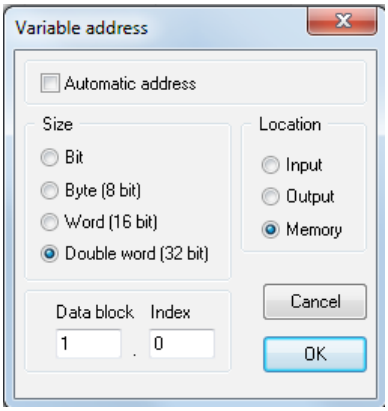
2) 通过编辑相应单元格中的类型名称或单击该单元格中的按钮来更改变量类型，然后从弹出的列表中选择所需的类型。



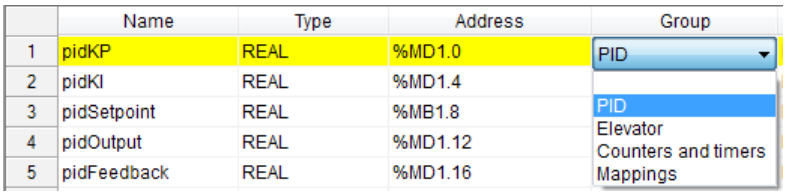


3) 通过单击相应单元格中的按钮并在出现的窗口中输入所需的信息来编辑变量的地址。请注意，对于全局变量，此操作可能会更改变量在项目树中的位置。

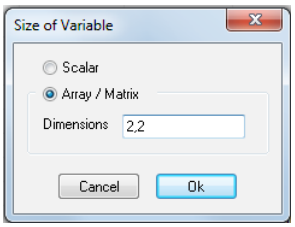




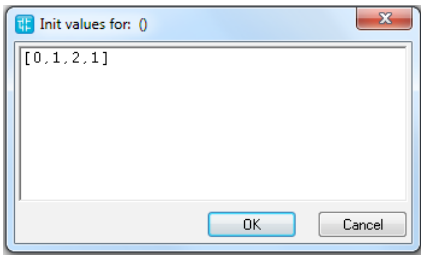
4) 对于全局变量，可以通过在单击相应单元格时打开的列表中进行选择，将变量分配给组。此操作将更改变量在项目树中的位置。



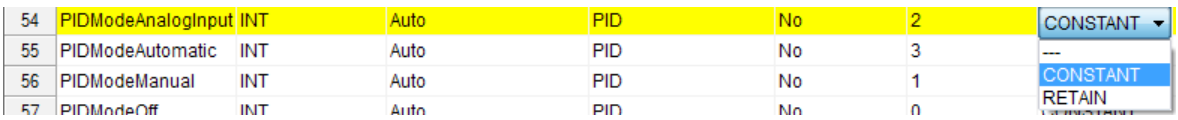
5) 选择一个变量是否为数组；如果是，请编辑变量的大小。



6) 编辑变量的初始值：单击相应单元格中的按钮，然后在弹出的窗口中输入值。



7) 通过从单击相应单元格时打开的列表中选择变量，给变量分配属性（例如，*常数* 或 *保持型*）。



8) 在相应的单元格中键入变量的描述。请注意，对于全局变量，此操作可能会更改变量在项目树中的位置。

	No	2	CONSTANT	Indicates PID analog input reference mode
	No	3	CONSTANT	Indicates PID automatic reference mode
	No	1	CONSTANT	Indicates PID manual reference mode
	No	0	CONSTANT	Indicates PID reference mode disabled

9) 保存项目以保留对变量声明所做的更改。

6.6.4 删除变量

为了删除一个或多个变量，请在编辑器中选择它们：您可以使用CTRL或SHIFT键选择多个元素。

	Name	Type	Address
1	pidKP	REAL	%MD1.0
2	pidKI	REAL	%MD1.4
3	pidSetpoint	REAL	%MB1.8
4	pidOutput	REAL	%MD1.12
5	pidFeedback	REAL	%MD1.16
6	pidKD	REAL	%MD1.20
7	hmiPIDSetpoint	REAL	%MB1.24

然后单击 *变量>删除* 。

请注意，您不能单独删除IEC61131-3功能的输出变量。

6.6.5 排序变量

您可以在编辑器中对变量进行排序，单击要用作排序字段的列标题。

	Name	Type	Ac
1	valueFilt	REAL	%M
2	tau	REAL	%M
3	valueRef	REAL	%M
4	period	REAL	%M
5	klnilncr	INT	Auto
6	ki	UINT	Auto
7	incr	INT	Auto
8	freeRunCounter	DINT	Auto

	Name	Type	Ac
1	freeRunCounter	DINT	Auto
2	incr	INT	Auto
3	ki	UINT	Auto
4	klnilncr	INT	Auto
5	period	REAL	%M
6	tau	REAL	%M
7	valueFilt	REAL	%M
8	valueRef	REAL	%M

6.6.6 复制变量

变量编辑器使您可以快速复制和粘贴元素。

您可以使用键盘快捷键菜单中 *编辑>复制* 和 *编辑>粘贴* 。

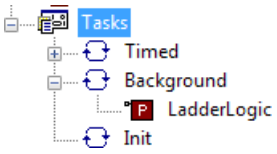
注意： 复制映射的变量可能会出现地址重叠的问题。 LogicLab可以自动将新的空闲地址分配给新粘贴的变量，并解决重叠问题。 为了启用此功能，请参阅第3.6和4.8.3.2段以获取更多详细信息。

7. 编译

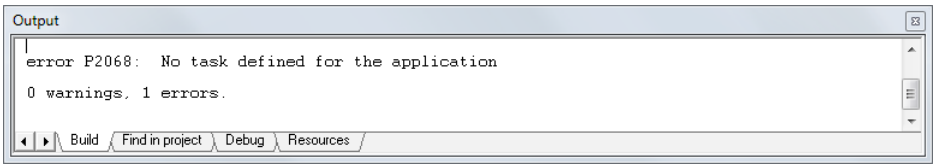
编译过程包括获取PLC源代码并将其自动转换为二进制代码，这些代码将在目标设备上的处理器执行。

7.1 编译项目

在开始实际编译之前，请确保至少已将一个程序分配给一个任务。



当此先决条件不成立时，编译将中止并显示相应的错误消息。

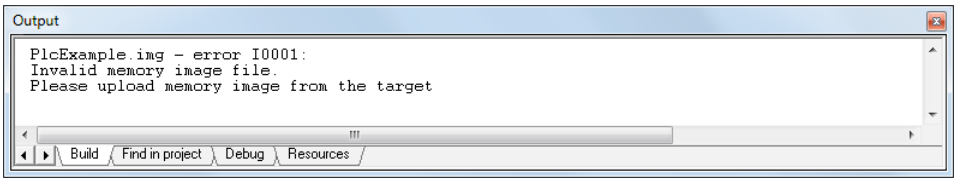


为了开始编译，请点击 *项目>编译* 。

需要注意的是LogicLab开始编译前会自动保存项目的所有变更。

7.1.1 加载镜像文件

在执行实际编译之前，编译器需要加载映像文件（img文件），该文件包含目标设备的内存映射。 如果在开始编译时已连接目标，则编译器将直接在目标上查找映像文件。否则，它将从工作文件夹中加载镜像文件的本地副本。 如果目标设备已断开连接，并且没有映像文件的本地副本，则无法进行编译：需要连接到正在运行的目标设备。



7.2 编译器输出

如果完成了上一步，则编译器将执行实际的编译，然后在“输出”窗口中打印报告。 报告的最后一个字符串具有以下格式：

```
m warnings, n errors
```

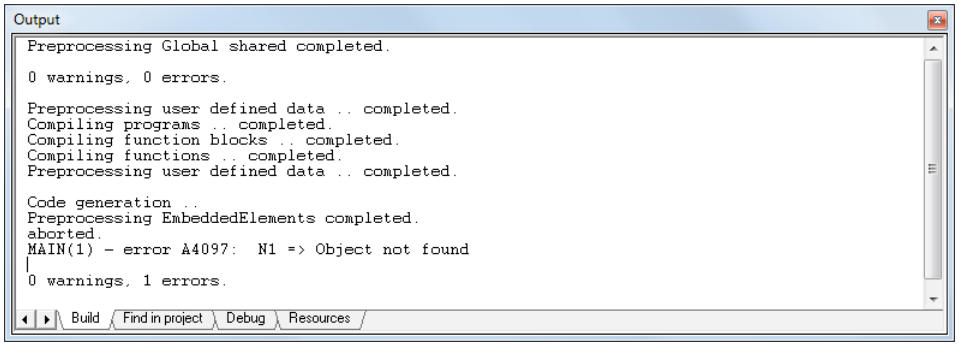
它表示编译的结果。

条件	描述
n > 0	编译器错误。 PLC代码包含一个或多个严重错误，编译器无法解决。

条件	描述
n = 0时， m > 0	发出警告。 PLC代码包含一个或多个小错误，编译器会自动发现并解决这些小错误。 但是，系统会通知您PLC程序可能以与预期不同的方式运行：建议您通过编辑和重新编译应用程序来消除这些警告，直到没有警告消息发出为止。
n = m = 0	PLC代码完全正确，编译完成。你应该总是以0警告，0失误的进行编译。

7.2.1 编译器错误

当您的应用程序编译过程中包含一个或多个错误时，对于这些每个错误，都会在 *输出* 窗口中打印一些有用的信息。

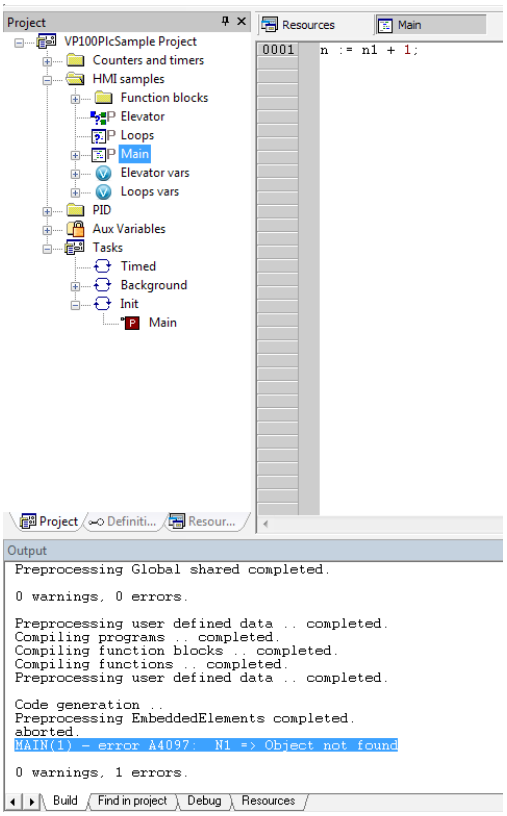


如您所见，这些信息包括：

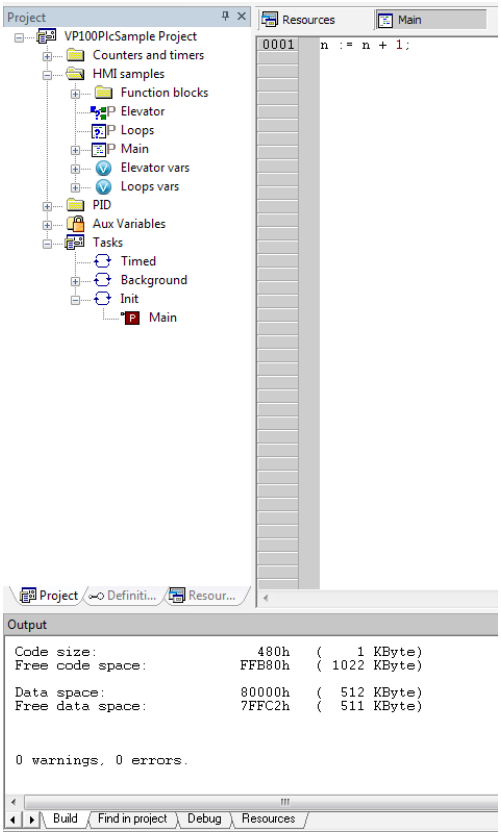
- 受错误影响的程序组织单元的名称;
- 导致错误的源代码行的编号;
- 是一个致命的错误（*错误*），还是一个编译器可以解决的错误（*警告*）;
- 错误代码;
- 错误描述。

请参考编译器错误引用相应的部分。

如果您在 *输出* 栏中双击该错误消息，LogicLab将打开源代码并突出包含错误的行。



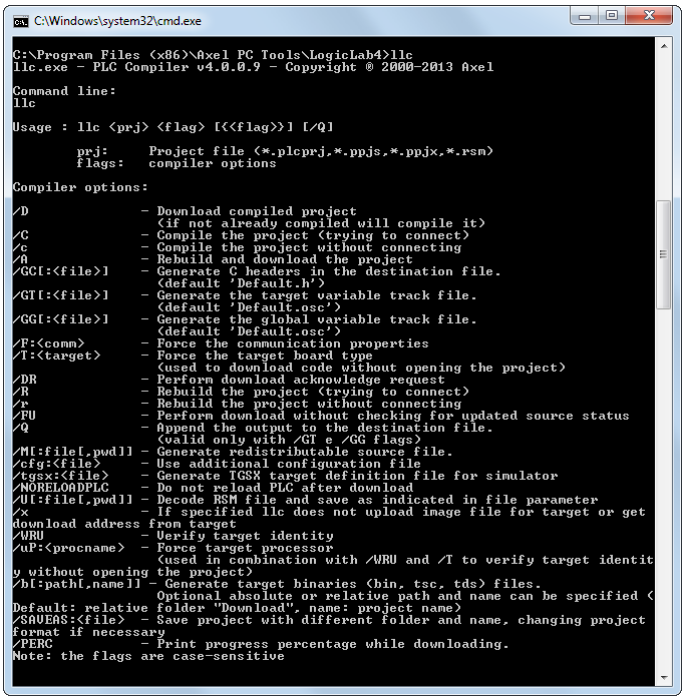
然后，您可以解决问题后进行重新编译。



7.3 命令行编译

LogicLab的编译器可以独立于IDE使用：在LogicLab的目录中，您可以找到可执行文件命令行编译器，可以使用许多选项来调用（例如，以批处理文件形式）。

为了获得有关此命令行工具的语法和选项的信息，只需启动不带参数的可执行文件即可。



8. 启动应用程序

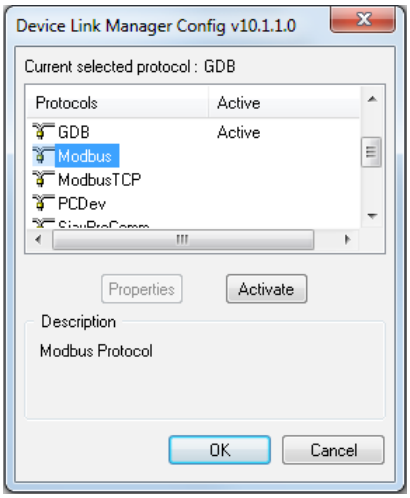
为了下载和调试应用程序，您必须与目标设备建立连接。本章重点介绍连接到目标和下载应用程序所需的操作，而LogicLab的各种调试工具应单独撰写一章（请参阅第9章）。

8.1 建立通信

为了建立与目标设备的连接，请确保物理链路已建立（所有电缆均已插入，网络已正确配置等）。

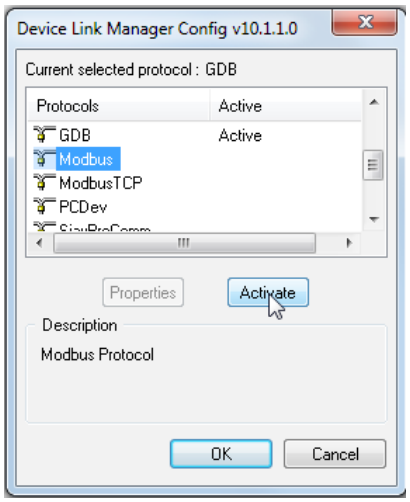
请遵循以下过程来建立并建立与目标设备的连接：

1) 点击LogicLab主窗口的 *联机>设置通信...* 选项。将会弹出以下对话框。

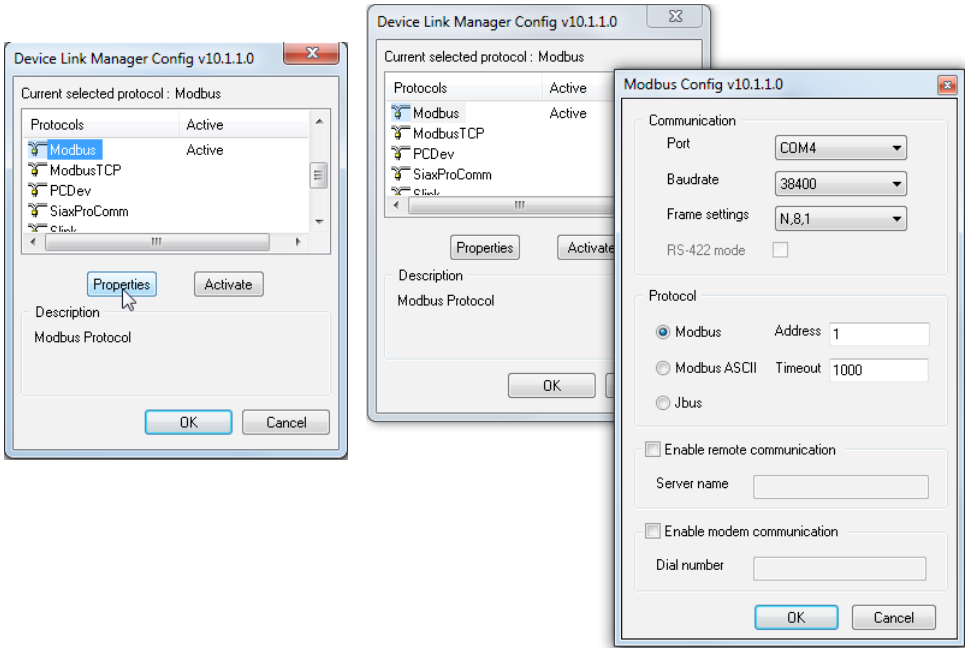


您可以选择的通信协议列表中的元素取决于您在PC上运行的安装可执行文件（如果缺少希望在列表中显示的协议，请与硬件提供商联系）。

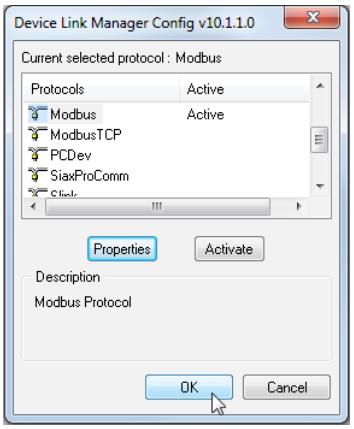
2) 选择适当的协议并，并激活它。



3) 填写所有协议的设置（例如地址或通信超时，其中通信超时表示LogicLab在显示通信错误消息之前必须等待目标响应的时间）。



4) 将您所做的更改应用到通信设置。

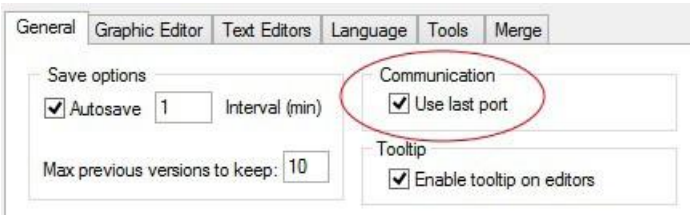


现在，您可以通过点击 *上线>连接* 选项来建立通信。

8.1.1 保存最后使用的通信端口

当使用串行端口（COM端口）连接到目标设备时，通常需要对所有设备使用相同的端口（许多现代PC仅具有一个COM端口）。 您可以保存上次使用的COM端口，并让LogicLab使用该端口来覆盖项目设置：与其他开发人员共享项目时，此功能非常有用，后者可以使用其他COM端口连接到目标设备。

为了保存您的COM端口设置，请在 *文件\选项...* 菜单中启用 *使用最后一个端口* 选项。

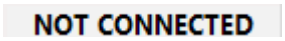


8.2 在线状态

8.2.1 连接状态

通信 状态显示在状态栏右边框旁边的小框中。

如果尚未尝试连接到目标，则通信状态将设置为 *未连接* 。



当您尝试连接到目标设备时，通信状态将变为以下状态之一：

- 错误：无法建立通信。您应该同时检查物理连接和通信设置。

- 连接：已建立通信连接。

8.2.2 应用程序状态

通信状态旁边是另一个小方框，该方框提供有关目标设备上当前正在执行的应用程序状态的信息。

当连接状态为“已连接”时，应用程序状态将变为以下状态之一：

- `No code`：目标设备上没有正在执行的应用程序。

- `Diff. code`：目标设备上当前正在执行的应用程序与IDE中当前打开的应用程序不同；此外，没有与正在运行的应用程序一致的调试信息：因此，监视窗口或示波器中显示的值不具备参考价值，并且无法激活调试模式。

- `Diff. code, Symbols OK`：当前在目标设备上执行的应用程序与IDE中当前打开的应用程序不同；

但是，可以使用一些与正在运行的应用程序一致的调试信息（例如，因为该应用程序先前已从同一台PC下载到目标设备）：监视窗口或示波器中显示的值是可靠的，但是调试 模式仍无法激活。

- Source OK：当前在目标设备上执行的应用程序与IDE中当前打开的应用程序相同：可以激活调试模式。

8.3 下载应用程序

必须将已编译的PLC应用程序下载到目标设备，以使的处理器处理执行它。本段将说明如何使得PLC代码发送到目标设备。请注意，只有在目标设备连接到运行LogicLab的PC时，LogicLab才能将代码下载到目标设备。有关详细信息，请参见相关部分。

要下载该应用程序，请点击 *在线>下载代码* 。

LogicLab将检查项目是否有未保存的更改。在这种情况下，它将自动开始应用程序的编译。 最终将二进制代码发送到目标设备，然后在传输结束时对其进行自动重置。 现在，您发送的代码实际上已由目标设备上的处理器执行。

8.3.1 控制源代码下载

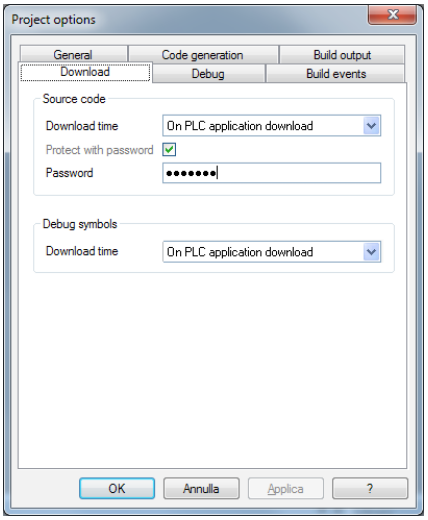
是否将应用程序的源代码与二进制代码一起下载，取决于您所连接的目标设备：某些设备将应用程序源代码托管在其存储中，以允许开发人员在后面的某个时刻中将项目的源代码上传到设备中。

在这种情况下，您可以控制源代码下载过程的某些方面，如以下各段所述。

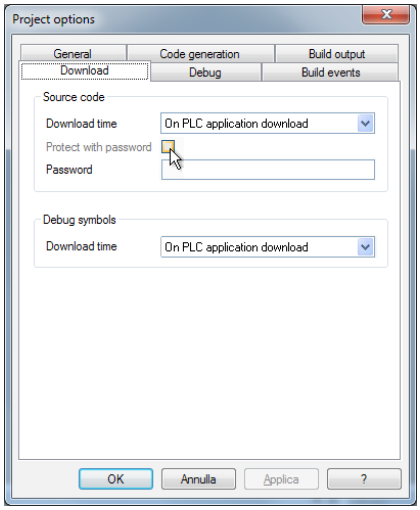
8.3.1.1 保护源代码密码

您可能希望使用密码保护下载到目标设备的源代码，使得LogicLab只有输入正确的密码，才能打开上传的项目。

单击 *项目>选项...* 菜单，然后设置密码。

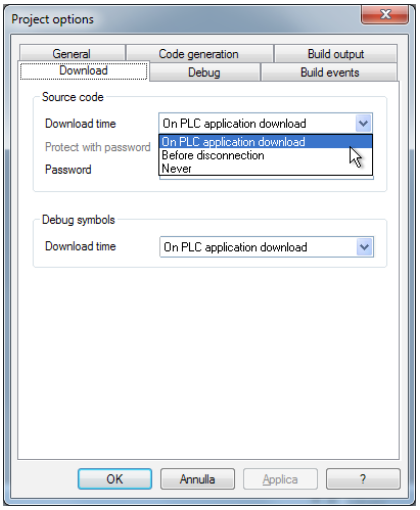


您也可以选择禁用密码。



8.3.1.2 源代码和调试符号的下载时间

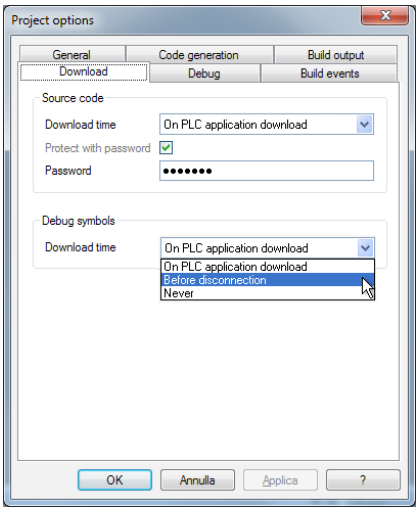
在下面的选择菜单中，您可以设置源代码下载时间。



选择：

- ***在下载PLC的应用程序时***：源代码将与PLC应用程序一起下载到目标设备中。
- ***断开连接之前***：源代码将在与目标设备断开连接之前下载。
- ***从不***：源代码将永远不会下载到目标设备。

可以使用以下具有相同选项的选择菜单来设置调试符号的下载时间。



8.4 模拟

根据与您连接的目标设备的不同，您也许可以使用LogicLab的集成仿真环境SimuLab对PLC应用程序的执行进行仿真。

为了开始仿真，只需单击 *调试>模拟模式* 。

请参阅SimuLab的手册以获取有关如何控制仿真的信息。

8.5 控制PLC执行

可以使用项目栏中的相关功能或通过在线菜单中显示的命令来控制PLC应用程序的执行。

8.5.1 暂停

您可以通过单 *在线>暂停* 停止PLC执行

8.5.2 冷启动

PLC应用程序将被重新启动，可保持型变量和非保持型变量都将被复位。

您可以通过单击 *在线>冷启动* 以重新冷启动PLC。

8.5.3 暖启动

PLC应用程序将被重新启动，并且非保持型变量将被复位。

您可以通过点击 *在线>暖启动* 以重新暖启动PLC。

8.5.4 热启动

PLC应用程序将被重新启动，不会复位任何类型的变量。

您可以通过点击 *在线>热启动* 以重新热启动PLC。

8.5.5 重启目标设备

您可以通过点击 *在线>重新启动目标* 以重新启动目标设备。

9. 调试

LogicLab提供了几种调试工具，可帮助开发人员检查应用程序的行为是否符合预期。

所有这些调试工具都允许开发人员在PLC应用程序运行时观察所选变量的值。

LogicLab调试工具可以分为两类：

- 异步调试器：向目标设备发出连续查询，读取开发人员选择的变量值。异步调试工具管理器（在PC上运行）以及可能负责响应这些查询的任务（在目标设备上）都独立于PLC应用程序运行。

  因此，就PLC应用程序的执行而言，不能保证同时对两个不同变量的值进行采样（采样时可能已经发生了一个或多个循环）。由于同样的原因，单个变量的值的变化也不连续，尤其是当它快速变化d的时候。

- 同步调试器。它们需要在PLC代码中定义触发器。当每次处理器到达触发器时，它们都会同时刷新已分配的所有变量，因为在刷新所有变量的值之前，无法再执行任何指令。由此，同步调试器消除了影响异步调试器的限制。

本章介绍了如何调试使用异步和同步异步的工具调试您的应用程序。

9.1 监视窗口

监视 窗口允许您监视一组变量的当前值。作为异步调试工具，*监视* 窗口不能保证值的实时同步。

因此，在 *监视* 窗口中读取变量的值时，请注意它当前显示的值可能来自于相应任务的不同执行周期。

监视 窗口包含项目添加到其中的每个变量。*监视* 窗口中显示的信息包括变量的名称、值、类型及其在PLC应用程序中的位置。

Symbol	Value	Type	Location
HMIPIDTEST	FALSE	BOOL	global
HMIPIDTHRESHOLD	0.2	REAL	global
PARCTDOWNPRESET	100	INT	global
BASETIME	0	UDINT	@FAST:PIDMODESELECTOR

9.1.1 开启和关闭监视窗口

如果要打开或者关闭监视窗口中，请单击 *视图>工具窗口>监视窗口* 。

关闭 *监视* 窗口意味着仅隐藏它，而不是对其进行重置。实际上，如果关闭 *监视* 窗口，然后再次打开它，您将看到它仍然包含您添加到其中的所有变量。

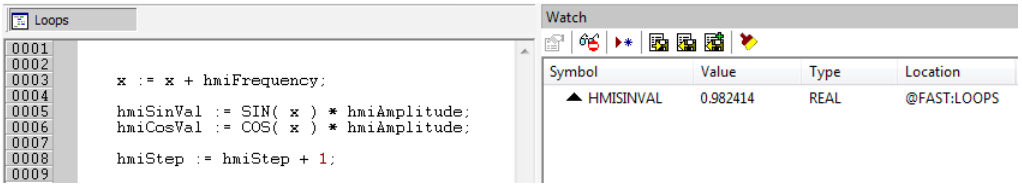
9.1.2 添加变量到监视窗口

要监视变量，您需要将它添加到监视列表下。

需要注意的是，不同于 *触发* 窗口和 *图形触发* 窗口中，无论变量声明在何处，您都可以将此变量添加到 *监视* 窗口。

9.1.2.1 从文本编辑器源代码添加变量

请按照以下过程变量从文本（即IL或ST）的源代码编辑器向监视窗口中添加变量：选择一个变量，通过双击，然后将其拖动到监视窗口。

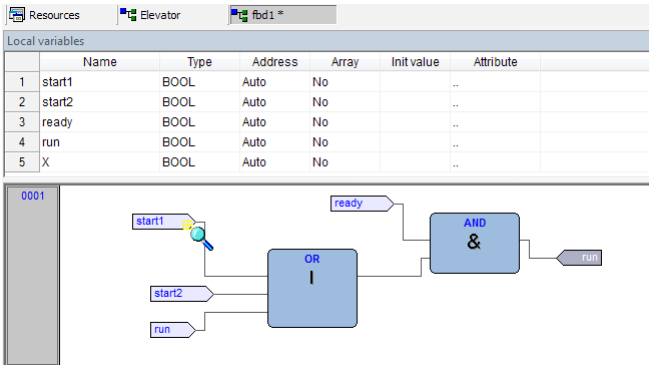


同样的步骤适用于所有需要监视的变量。

9.1.2.2 从图形源代码编辑器添加变量

请按照以下过程变量从图形（即LD、FBD或者SFC）的源代码编辑器向监视窗口中添加变量：

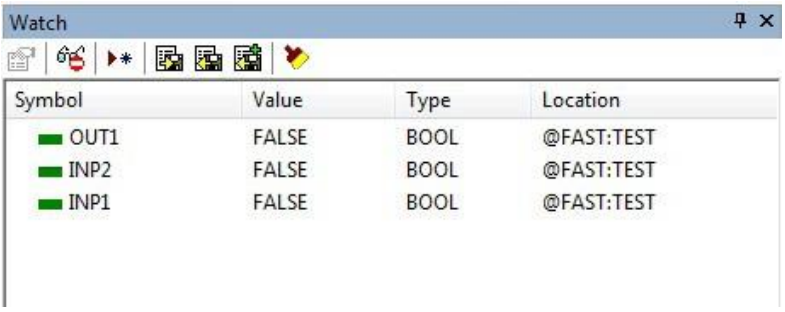
1) 点击 *编辑>监视模式* 。
2) 单击您希望在 *监视* 窗口中显示的变量的块。



将弹出一个对话框，将列出所有当前存在的调试窗口实例，并询问您哪个是您刚刚单击的对象。

为了在 *监视* 窗口中显示变量，请选择 *监视* ，然后按 *确认* 。

此时，变量的名称、值以及位置将会显示在 *监视* 窗口的新行中。

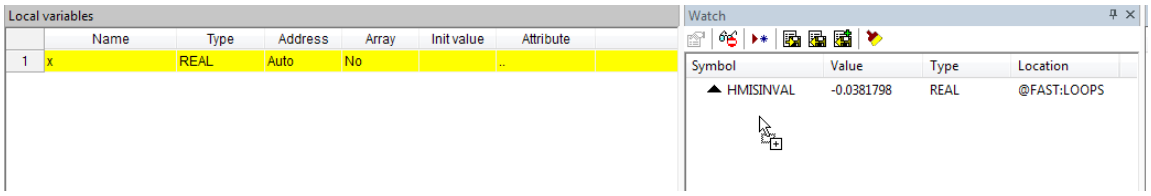


同样的步骤适用于您需要监视的所有变量。

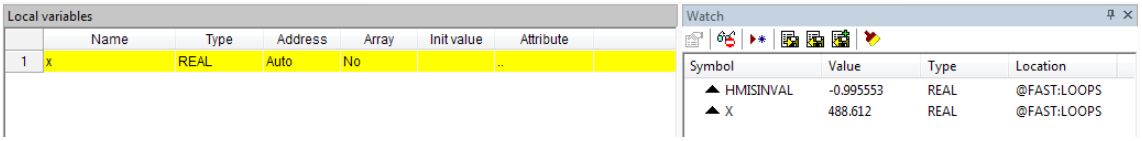
将所有要监视的变量添加到 监视 窗口，点击 *编辑>插入/移动模式* 鼠标光标将变成其原有的形状。

9.1.2.3 从变量编辑中添加变量

为了从变量编辑器中将变量添加到 *监视* 窗口，您可以在变量编辑器中选择变量所处的行，然后将其拖放到 *监视* 窗口中。

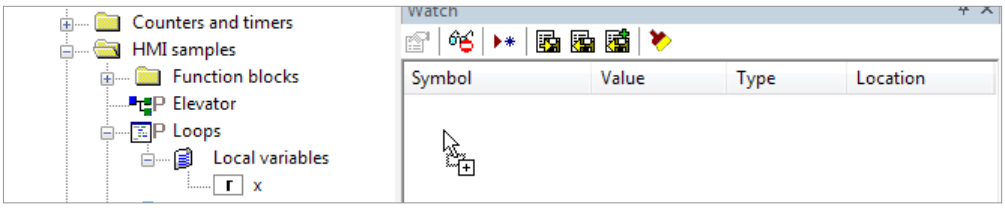


或按F8键。

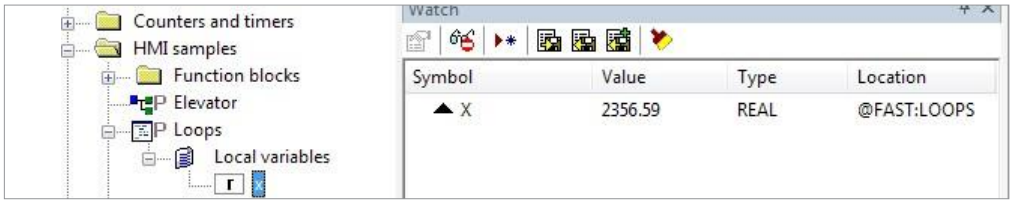


9.1.2.4 从项目树中添加变量

为了从工程树中将变量添加到 *监视* 窗口，可以在项目树中选择它，然后将其拖放到 *监视* 窗口中

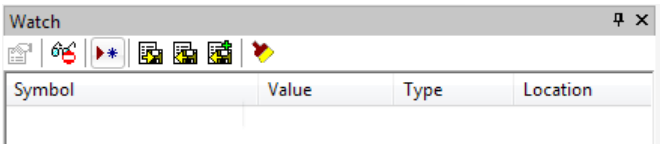


或按F8键。

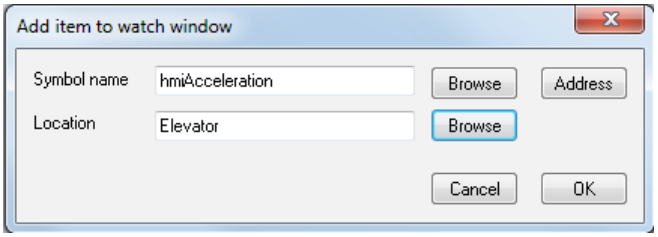


9.1.2.5 从监视窗口工具栏添加变量

您还可以单击 *监视* 窗口内部工具栏上的相应项目，以便向其中添加变量。

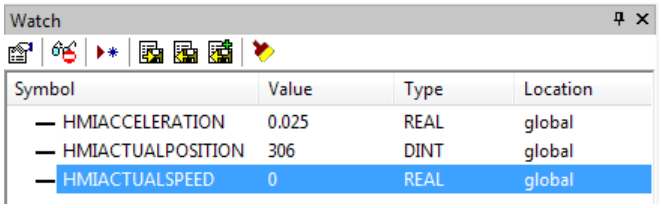


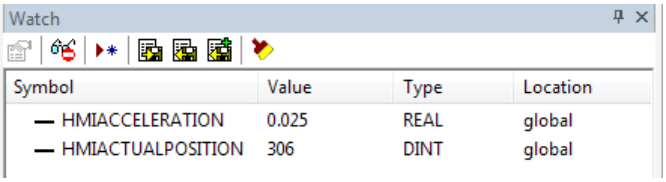
您应输入（或通过浏览项目符号进行选择）变量的名称及其位置（声明位置）。



9.1.3 删除变量

如果您不想在 *监视* 窗口中再显示一个变量，请单击其名称以选择它，然后按键盘上的Del键。

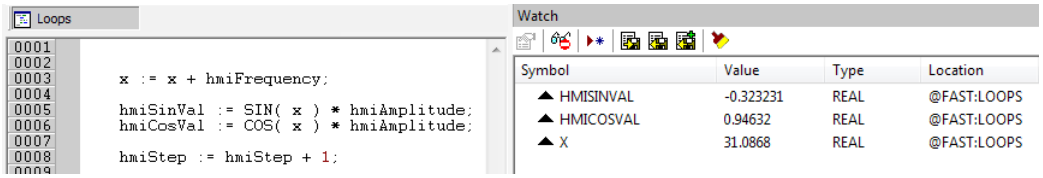




9.1.4 刷新值

9.1.4.1 普通操作

让我们看看下面的例子。

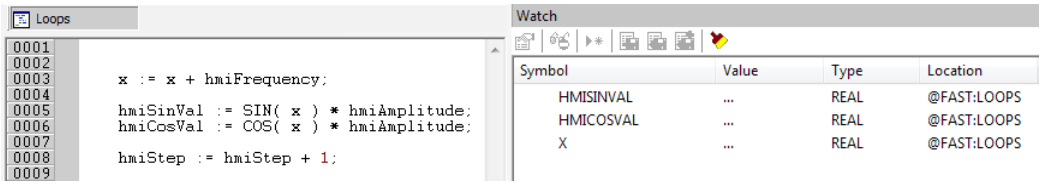


监视窗口管理器会定期从内存中读取变量的值。

但是，此操作是异步执行的，也就是说，优先级较高的任务可能会在读取某些变量时修改某些变量的值。 因此，在刷新过程结束时，窗口中显示的值可能引用自PLC代码的不同执行周期。

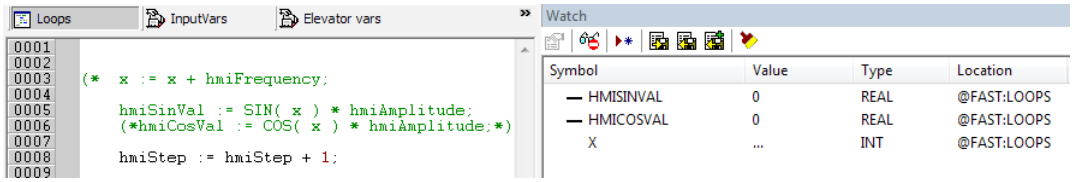
9.1.4.2 目标设备断开连接

如果目标设备断开连接时，监视窗口的 *值* 将包含以下三个。

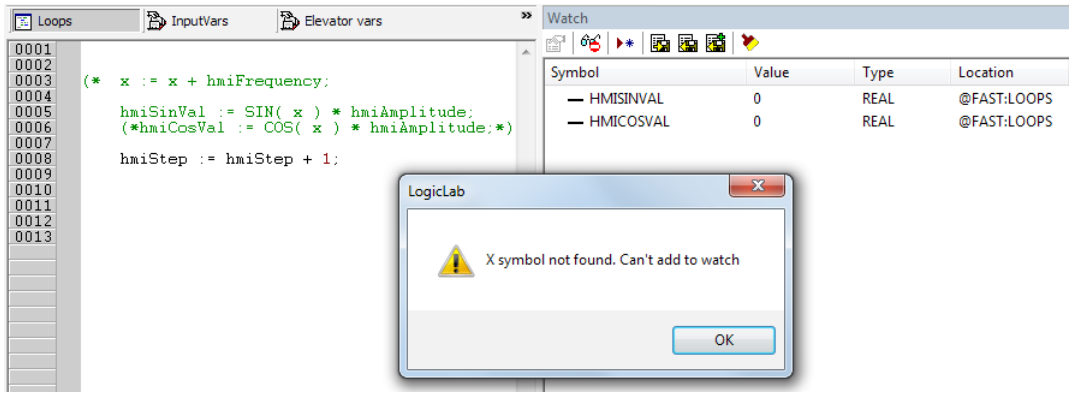


9.1.4.3 对象未找到

如果PLC代码更改，并且LogicLab无法在 *监视* 窗口中检索到对象的存储位置，监视窗口的 *值* 将包含以下三个。



如果您尝试将尚未分配的符号添加到监视窗口，LogicLab将提示以下错误信息。

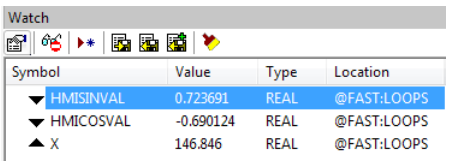


9.1.5 更改数据的格式

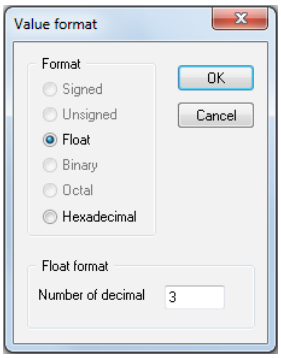
将变量添加到 *监视* 窗口时，LogicLab会自动识别其类型（无符号整数，有符号整数，浮点数，十六进制），并始终显示其值。此外，如果变量是浮点数，则LogicLab会为其分配默认数量的十进制数字。

但是，您可能需要将变量以其他格式打印输出。

要增加LogicLab自动分配的格式以外的其他格式，请按工具栏中的 *格式值* 按钮。



选择格式并确认您的选择。

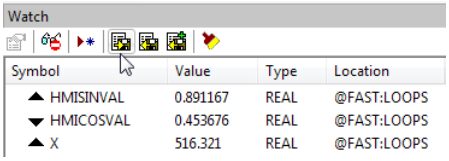


9.1.6 使用监视列表

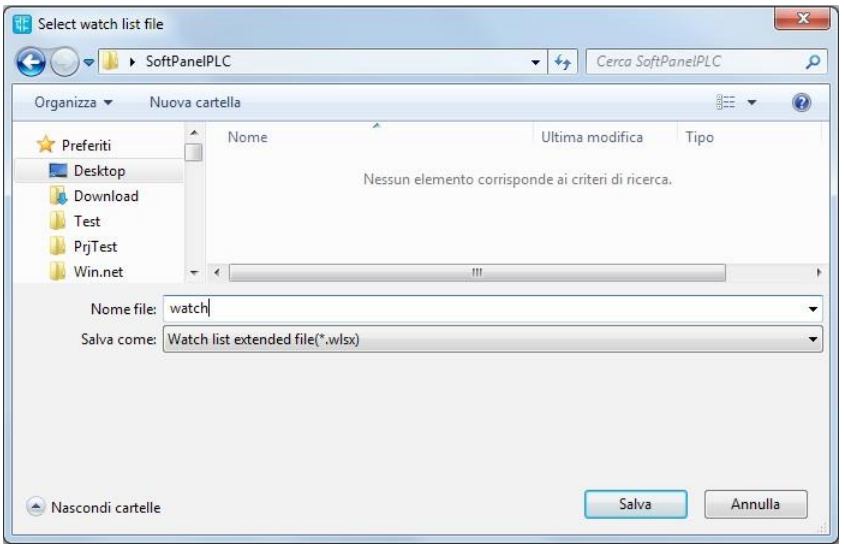
您可以将 *监视* 窗口中的所有监视对象的集合保存到文件，以便在后续的工作汇中可以轻松地调用调试工具中的设定。

请按照一下步骤保存监视裂变：

1) 点击 *监视* 窗口工具栏中相应项目。

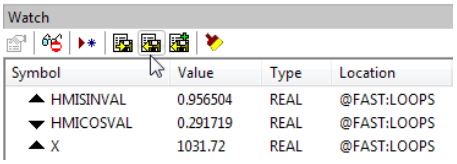


2) 输入文件名，并选择其在文件系统中的存储位置。

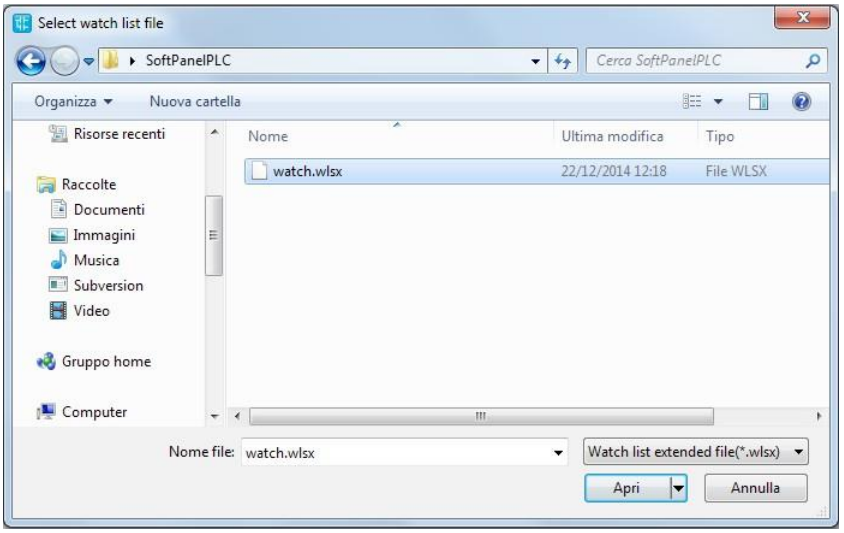


您可以按照以下步骤从文件中加载一个监视列表，并移除当前打开的监视列表：

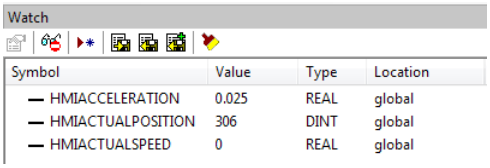
1) 点击监视窗口工具栏中相应的图标。



2) 浏览文件系统并选择监视列表文件。

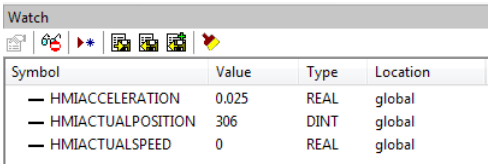


在监视列表中变量集合将被添加到监视窗口。

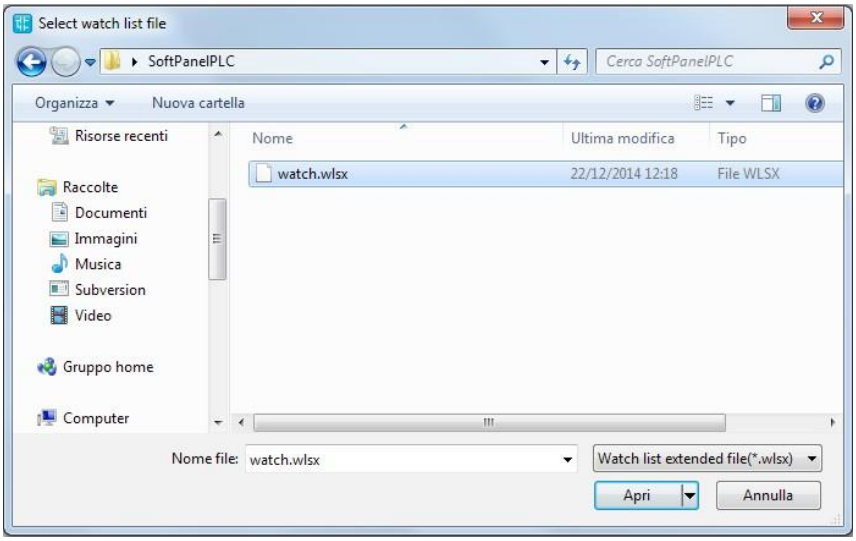


您可以按照以下步骤从文件中加载的监视列表，并将其追加到打开的监视列表中：

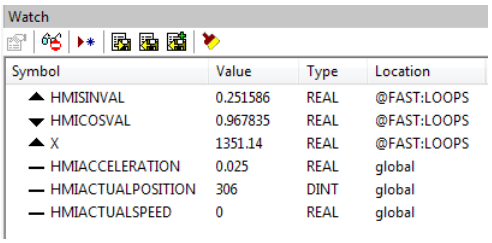
1) 点击监视窗口工具栏中相应的图标。



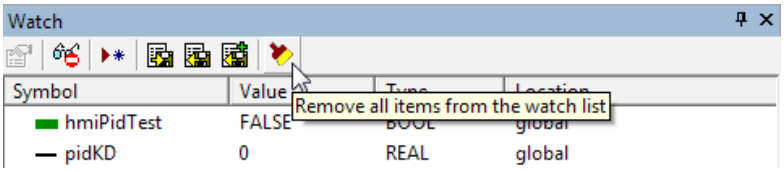
2) 浏览文件系统并选择监视列表文件。



监视列表中的变量集合将被添加到监视窗口。



您可以通过点击以下图标来清除当前打开的监视列表：



9.1.7 自动保存监视列表

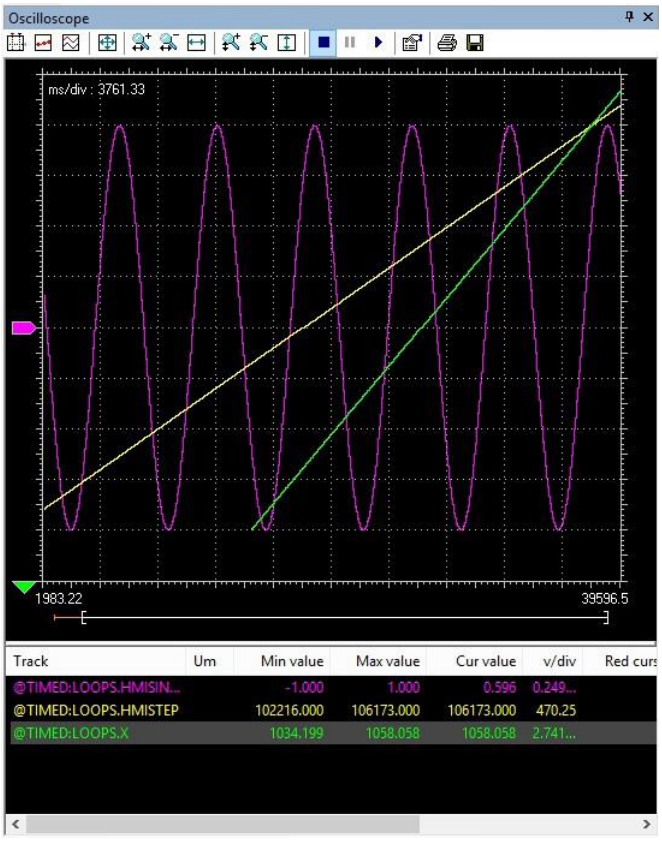
通过在项目选项对话框中选择关联的选项（有关更多信息，请参见第4.6.5节），监视列表将在项目结束时自动保存。

重新打开项目时，保存的监视列表将在第一个连接上自动加载（没有附加选项）到目标。

9.2 示波器

示波器允许您绘制一组变量值的变化。 示波器是一种异步工具，不能保证变量值的实时同步。

打开示波器窗口将显示在LogicLab右边框旁。这是用于访问示波器提供的调试功能的界面。 示波器由三个元素组成，如下图所示。



工具栏可以让你更好地控制示波器。本章后面的章节将详细为你介绍每个控件的功能。

图表区域包含以下几项：

- 绘图区域：包含变量曲线的区域。
- 垂直光标：标识两个不同的垂直线的光标。列出与这些垂直线相交的每个变量的值。
- 滚动条：如果x轴的比例过大而无法在绘图区域显示的所有采集样本，滚动条可以让您沿水平轴来回滑动。

示波器的下部的表格中的行分别表示每个变量的值。

9.2.1 开启和关闭示波器

如果要打开或者关闭示波器，请点击 *视图>工具窗口>示波器* 。

关闭示波器只不过是将其隐藏而不是复位它。实际上，如果在关闭示波器后再次打开它，您将重新看到添加所有变量的曲线图。

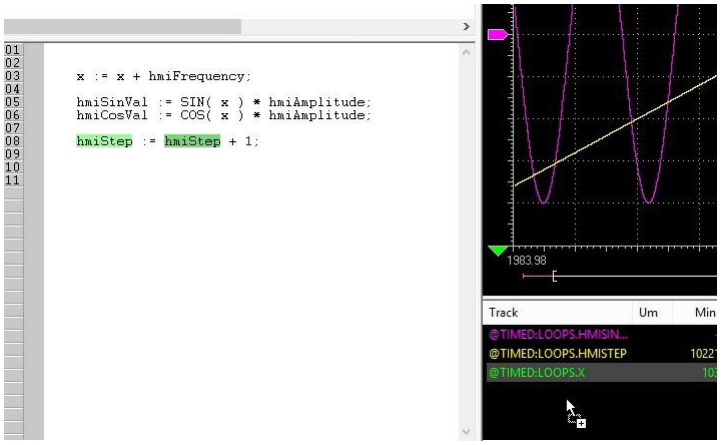
9.2.2 添加对象到示波器

为了绘制一个变量值的变化，你需要把它添加到示波器中。

请注意，与 *触发* 窗口和 *图形触发* 窗口不同，无论变量对象声明在何处，都可以将其添加到示波器中。

9.2.2.1 文本编辑器源代码添加变量

请按照以下步骤从文本（即IL或ST）源代码编辑器添加变量到示波器中：双击目标变量，然后将其拖到 *示波器* 窗口中。

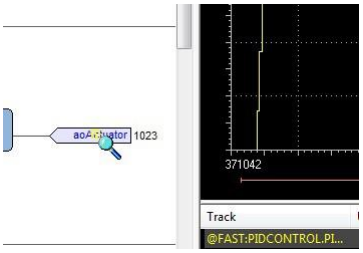


同样的步骤适用于所有要监视的变量。

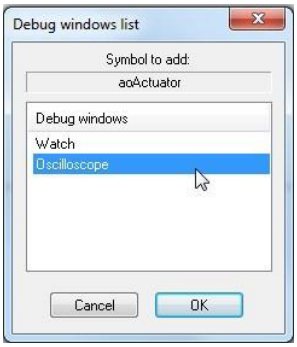
9.2.2.2 从图形源代码编辑器添加变量

请按照以下过程从图形（即LD、FBD或SFC）源代码编辑器向示波器添加变量：：

1) 点击 *编辑>监视模式* 。
2) 单击您希望在 *监视* 窗口中显示的变量的块。



3) 将弹出一个对话框，将列出所有当前存在的调试窗口实例，并询问您哪个是您刚刚单击的对象。



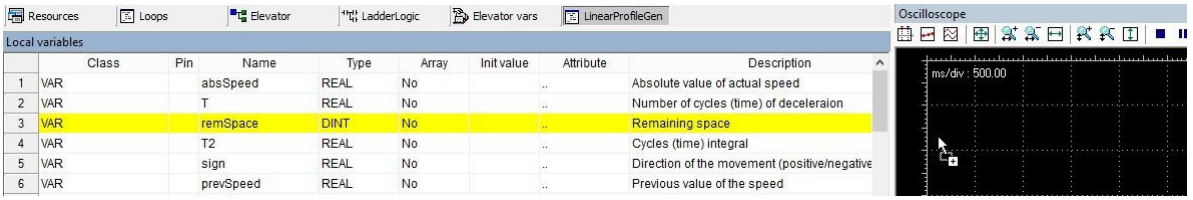
选择 *示波器* 时，然后按 *确定* 。变量的名称将显示在 *追踪* 列表中。

同样的步骤适用于您需要监视的所有变量。

将所有要监视的变量添加到 *示波器* 窗口，点击 *编辑>插入/移动模式* 鼠标光标将变成其原有的形状。

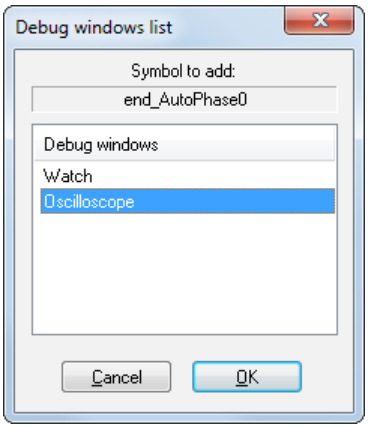
9.2.2.3 添加变量从变量编辑

为了将一个变量添加到 *示波器* 窗口，您可以在变量编辑器中选择变量所处的行，然后将其拖放到 *示波器* 窗口中



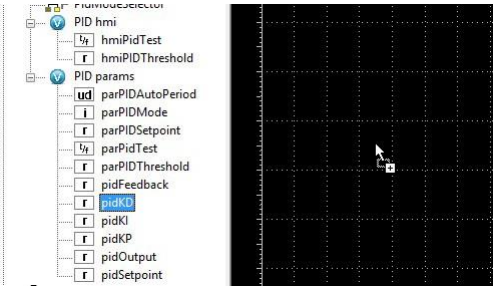
	Class	Pin	Name	Type	Array	Init value	Attribute	Description
1	VAR		absSpeed	REAL	No		..	Absolute value of actual speed
2	VAR		T	REAL	No		..	Number of cycles (time) of deceleraion
3	VAR		remSpace	DINT	No		..	Remaining space
4	VAR		T2	REAL	No		..	Cycles (time) integral
5	VAR		sign	REAL	No		..	Direction of the movement (positive/negative
6	VAR		prevSpeed	REAL	No		..	Previous value of the speed

或按F10键，从弹出的调试窗口列表中选择 *示波器*。



9.2.2.4 从项目树中添加变量

为了从工程树中将变量添加到 *示波器* 窗口，可以在项目树中选择它，然后将其拖放到 *示波器* 窗口中



或按F10键，从弹出的调试窗口列表中选择 *示波器* 。

9.2.3 删除变量

如果您不想在 *示波器* 窗口中再显示一个变量，请单击其名称以选择它，然后按键盘上的Del键。

9.2.4 变量抽样

9.2.4.1 普通操作

示波器管理器会定期的从内存中读取变量的值。但是，此操作是异步执行的，也就是说，优先级较高的任务可能会在读取某些变量时修改某些变量的值。因此，在采样过程结束时，与x轴的相同值关联的数据实际上可能引用自PLC代码的不同执行周期。

9.2.4.2 目标设备断开连接

如果断开目标设备的连接，则所拖动变量的曲线将被冻结，直到恢复通讯。

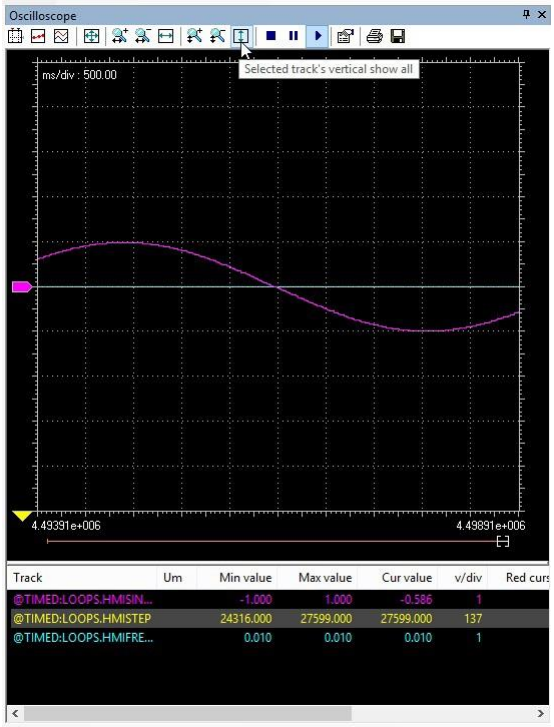
9.2.5 控制数据的采集和显示

示波器包括一个带有多个命令的工具栏，可用于控制采集过程和数据显示方式。

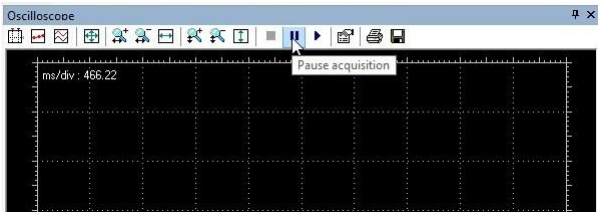
本段重点介绍这些命令。请注意，如果未将任何变量添加到示波器，则工具栏中的所有命令都将被禁用。

9.2.5.1 开始和停止数据采集

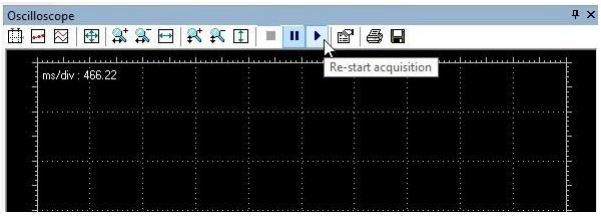
当您添加一个变量到示波器后，数据采集立即开始。



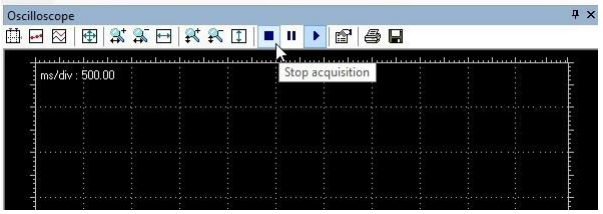
但是，您可以通过点击 *暂停采集* 按钮来暂停数据采集过程。



曲线冻结（虽然数据采集过程仍在后台运行），直到您单击 *重新开始采集* 。



您可以点击 *停止采集* 按钮来停止采集数据的过程。

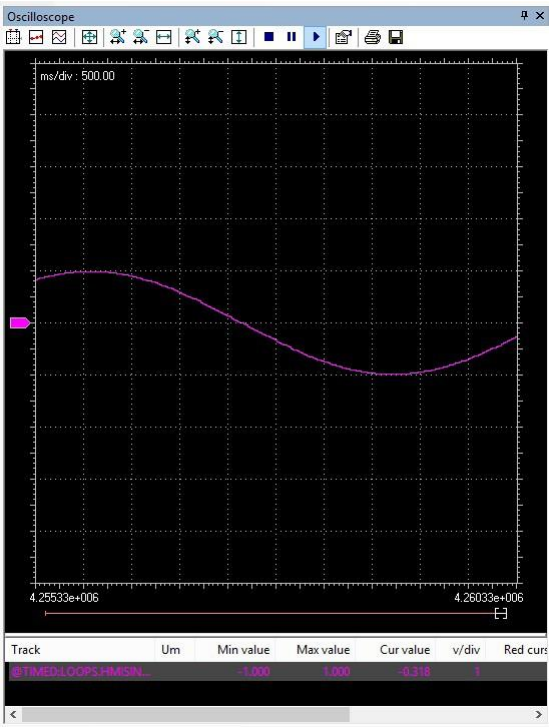


在这种情况下，当您单击 *重新开始采集* 时，将从头开始绘制变量值的变化。

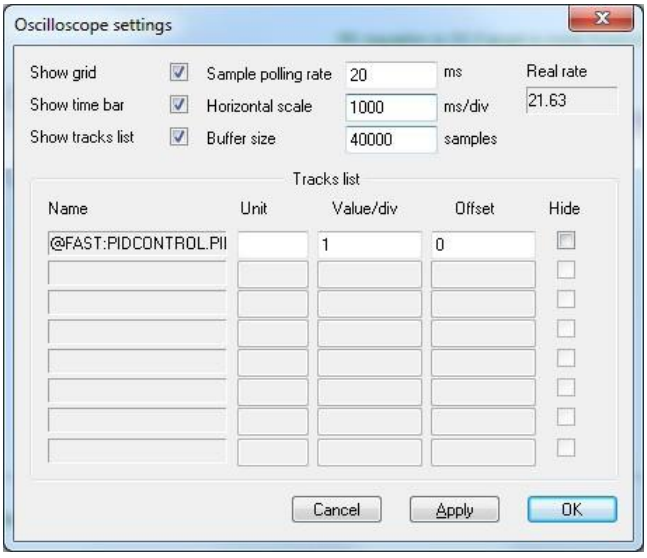
9.2.5.2 设置坐标轴的刻度

当您打开示波器时，LogicLab将默认比例应用于轴。 但是，如果要设置其他比例，可以按照以下步骤操作：

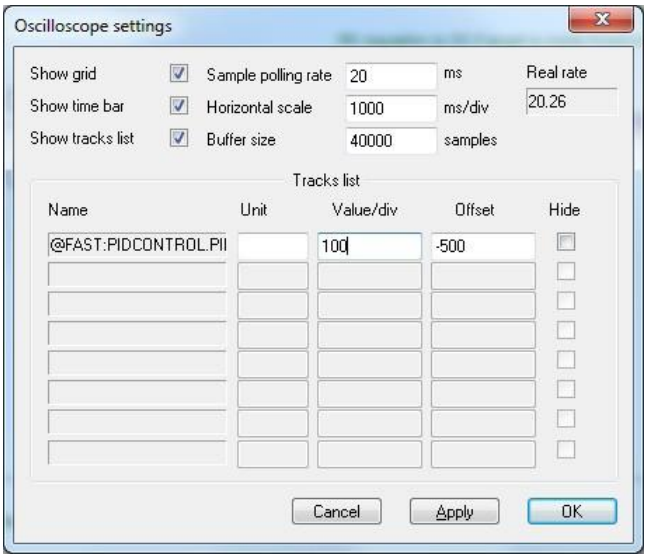
1) 通过点击工具栏上的相应的对象来打开图形属性。



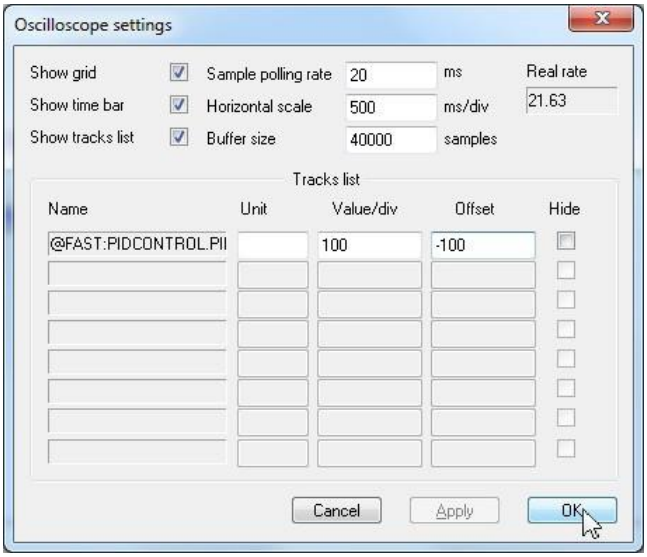
2) 设置水平轴的比例，这将应用到所有曲线的坐标系。

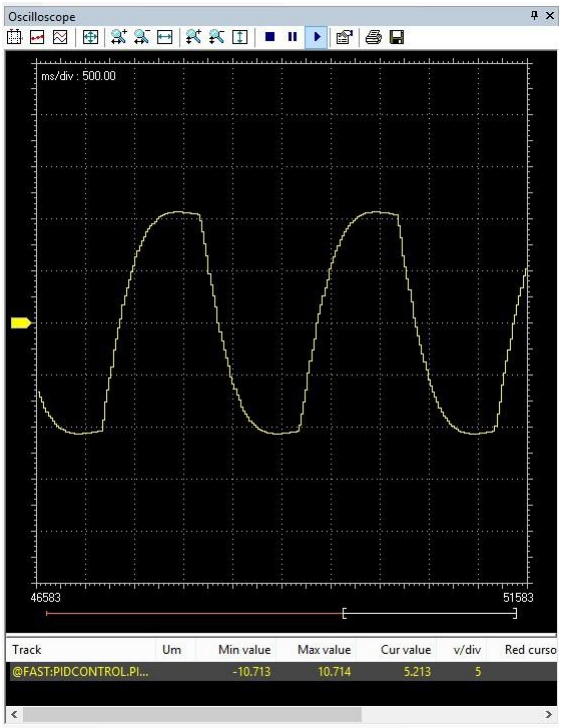


3) 对于每个变量，您可以为垂直坐标指定不同比例。

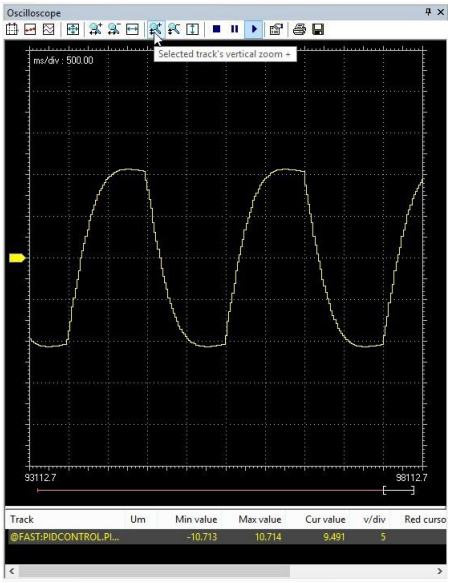


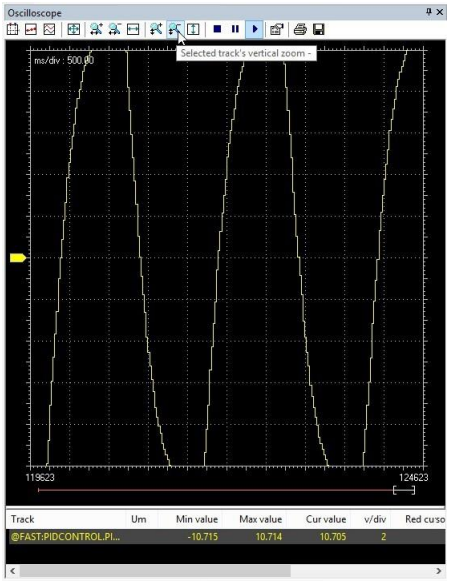
4) 确认您的设置，图表将自动适应为新的设置。



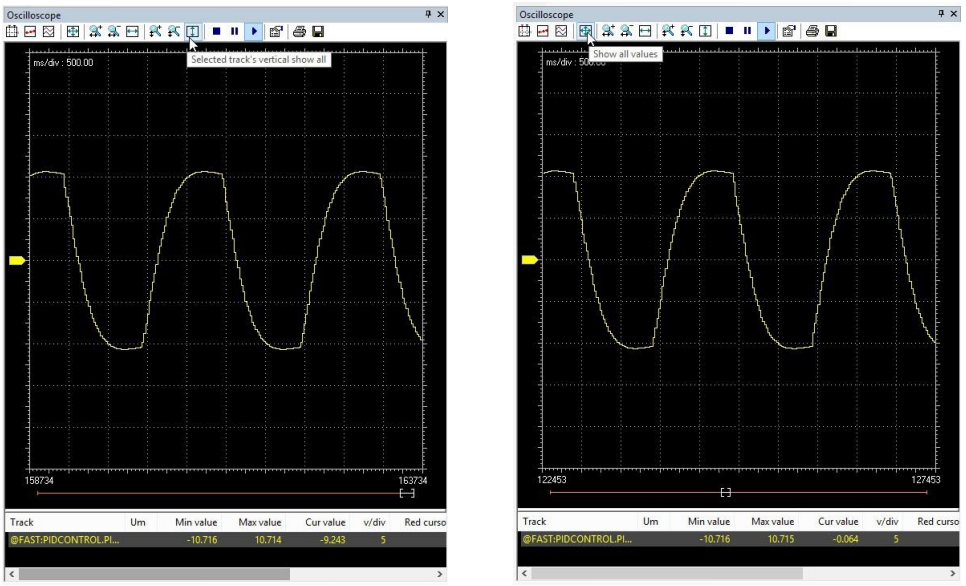


您还可以相对于水平轴和垂直轴进行放大或缩小。



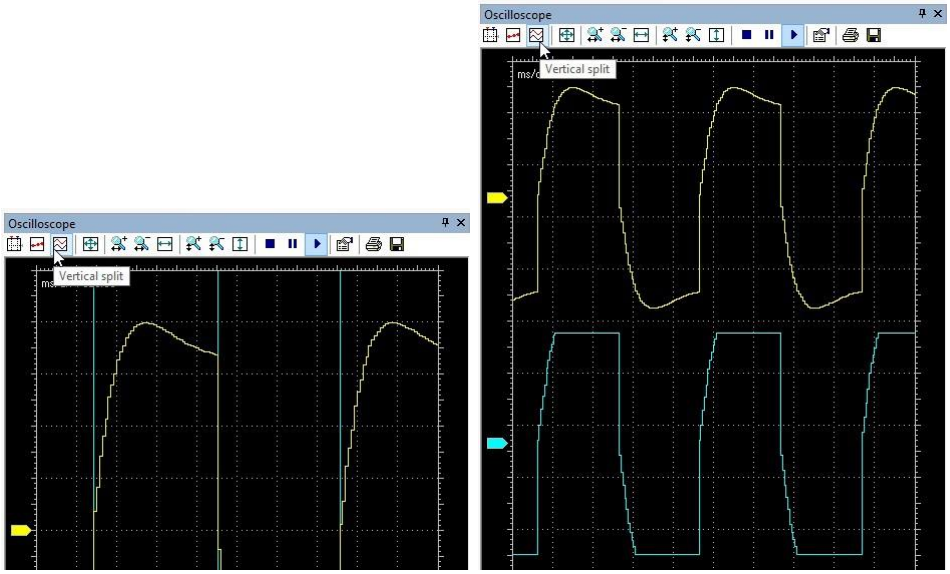


最后，你可以通过点击工具栏的相应按钮，快速地调整水平轴和垂直轴的比例，以使得可见的坐标系中包含所有需要的变量曲线。



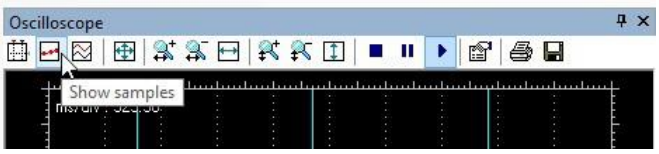
9.2.5.3 垂直拆分

当您正在观察看两个或多个变量的变化时，您可能希望将其拆分到各自坐标系下。您可以在 *示波器* 工具栏中点击 *垂直拆分* 对象即可。

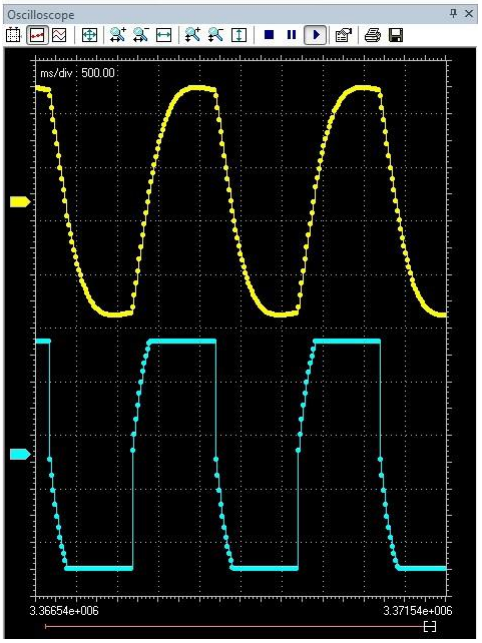


9.2.5.4 查看样本

如果单击 *示波器* 工具栏中的 *显示样本* 项，则该工具将突出显示在数据采集期间检测到的单个值 。



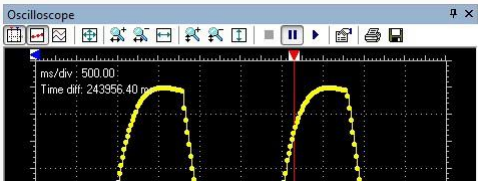
您可以再次点击此选项，示波器将返回到默认的视图模式。



9.2.5.5 采集变量方法

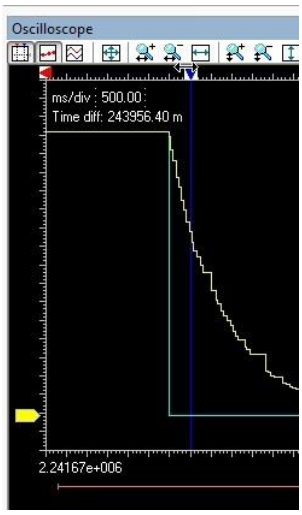
示波器包括两个测量按钮，可以利用这些 *测量按钮* 对图表进行一些测量。如果需要显示或隐藏它们，请单击示波器工具栏中的 *显示测量按钮* 。

如果要测量两个事件之间的时间间隔，只需在图形中将一个时间条移动到与第一个事件相对应的点，将另一个时间条移动至与第二个事件相对应的点。

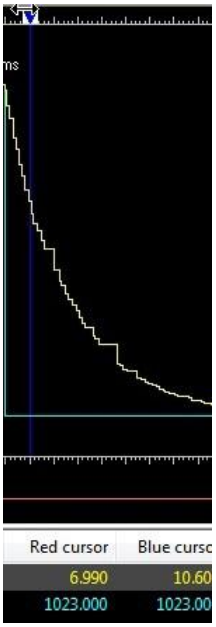


图表的左上角显示了两个时间条之间的时间间隔。

您还可以使用测量条在特定时刻读取示波器中所有变量的值：在图形中将时间条移动到与您要观察的瞬间相对应的点即可。



此时，在图表下方的表格中，您可以读取特定时刻所有变量的值。



9.2.5.6 示波器设置

您可以通过单击工具栏中的 *图形属性* 选项来进一步自定义示波器的外观。

在弹出的窗口中，您可以选择是否显示 *背景网格*， *时间滑动条* 和 *追踪列表* 。

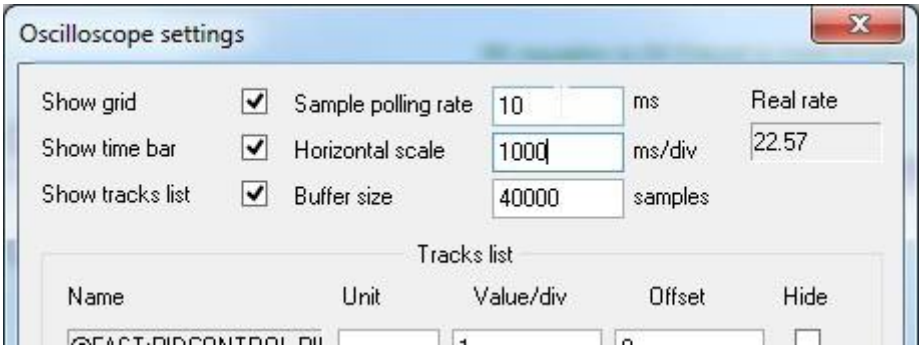
9.2.6 修改轮询速率

LogicLab周期性向目标设备发送查询，以读取要在示波器中绘制的数据。

可以通过以下步骤来配置轮询速率：

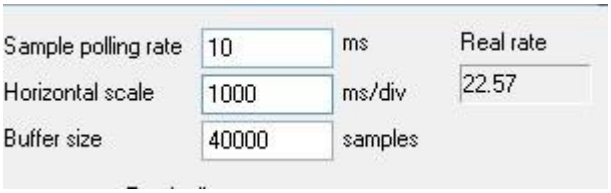
1) 点击工具栏上的 *图形属性* 项。

2) 在弹出的窗口中的编辑 *采样轮询速率*。



3) 确认您的设置。

请注意，实际速率取决于目标设备的性能（特别是取决于其通信任务的性能） 您可以在 *示波器设置* 窗口中读取实际速率。



9.2.7 保存和打印图表

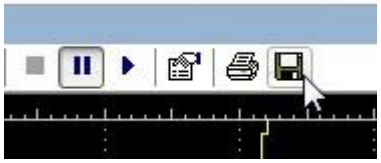
LogicLab允许你将数据保存到文件或通过打印示波器中绘制的数据视图。

9.2.7.1 将数据保存到文件

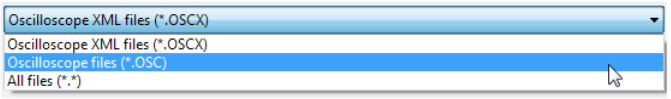
您可以将示波器采集的样本保存到文件中，以便使用其他工具进一步分析数据。

1) 您可能需要先停止采集，然后再将数据保存到文件。

2) 点击 *示波器* 工具栏中的 *将采集数据保存到文件*。



3) 选择您希望的输出文件格式：其中OSC是简单的纯文本文件，其中包含每个样本的时间和值；OSCX是XML文件，其中包含更完整的信息，可以使用与LogicLab分开提供的另一种工具进行进一步分析。

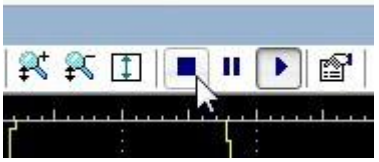


4) 选择一个文件名和目标目录，最后确认操作。

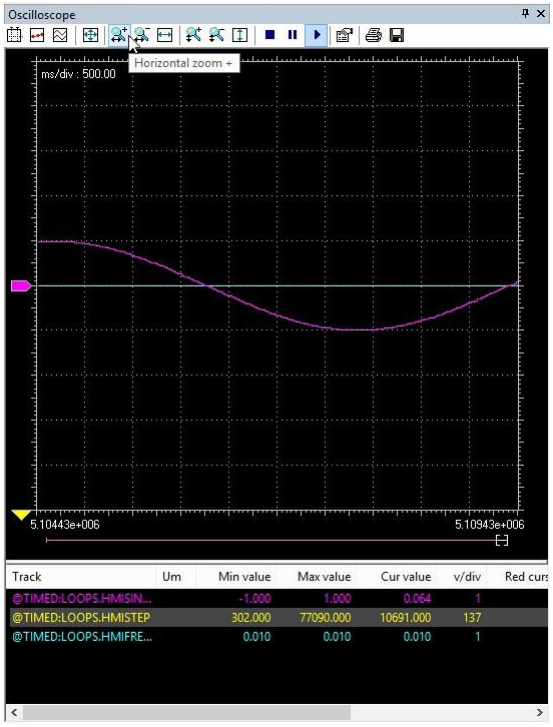
9.2.7.2 打印图表

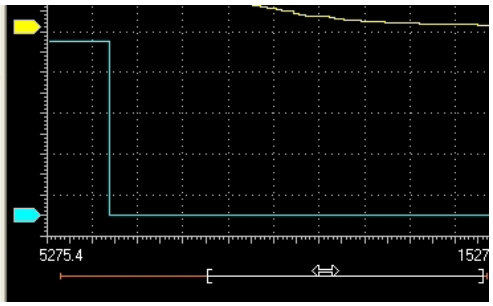
请按照如下步骤打印在示波器中绘制的数据的视图：

1) 首先需要停止或者暂停数据采集。



2) 移动时间滑块并调整缩放比例，以便在视图中包含要打印的曲线。





3) 点击 *打印图表* 选线。

9.3 编辑和调试模式

虽然 *监视* 窗口和 *示波器* 窗口均未使用源代码，但所有其他调试器都会使用到：当调试模式打开时，将禁止对源代码进行更改，并且调试工具将变为活动状态。

当至少满足以下条件之一时，LogicLab会自动启用调试模式：

- 至少正确设置一个断点。
- 至少正确设置了一个触发器（图形或文本中）。
- 实时调试模式已打开。

当不满足上述所有条件时，调试模式将自动关闭，并且LogicLab进入编辑模式。

状态栏显示调试模式是否处于活动状态。

需要注意的是，如果此时的连接状态不是 *已连接* ，您将不能进入调试模式。

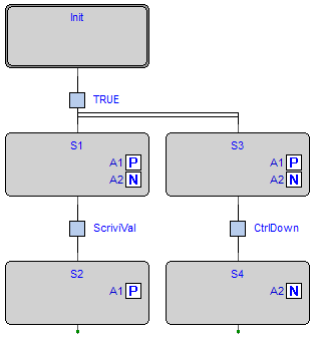
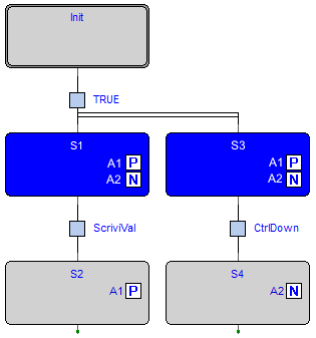
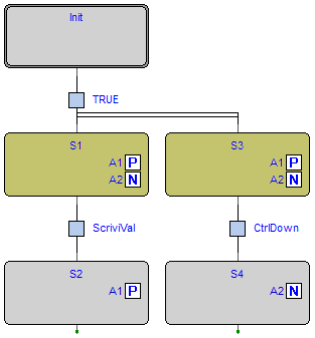
9.4 实时调试

LogicLab可以显示任何以IEC 61131-3编程语言编写的程序组织单位（POU）当前执行状态随时间的变化。

如果要打开或者关闭实时调试模式，您可以点击 *调试>实时调试模式* 。

9.4.1 SFC调试

如语言参考的相关部分所述，SFC POU由一系列步构成，每个步在任何给定时刻都是活动或不活动的。一旦启动此SFC专用调试工具将以动画的形式来通过突出显示SFC代码中活动步。

调试关	调试开	保持当前调试状态
		

在左列中，显示了*SFC*网络的一部分，调试动画处于关闭状态。。

如中间列网络中显示，当实时调试模式处于活动状态时，中间列的图片显示步 *S1* 和 *S3* 当前处于活动状态，而 *Init* 、*S2* 和 *S4* 未处于活动状态。

在右列中，显示了SFC网络的一部分以及当前处于活动保留状态的步 *S1* 和 *S3* 。在SFC语境中，当它们是非活动状态的根节点的分支时，可能会出现这种状态。

需要注意的是，SFC调试管理器会定期测试所有步骤的状态，并且不允许用户编辑采样周期。因此，可能会出现一个步处于激活状态的时间段太短而无法显示的情况。

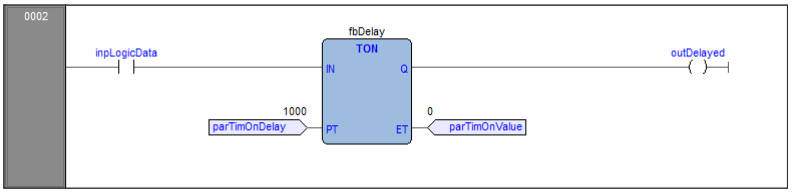
某个步没有被调试显示不代表未执行该步，这可能由于采样率太慢而无法检测到执行。

9.4.1.1调试操作和条件

如SFC语言参考中所述，可以将步分配给动作，并且可以将转换与条件代码关联。动作和条件可以用任何IEC 61131-3语言进行编码。通用调试工具可在每个动作/条件中使用，如同独立的POU一样。

9.4.2 LD调试

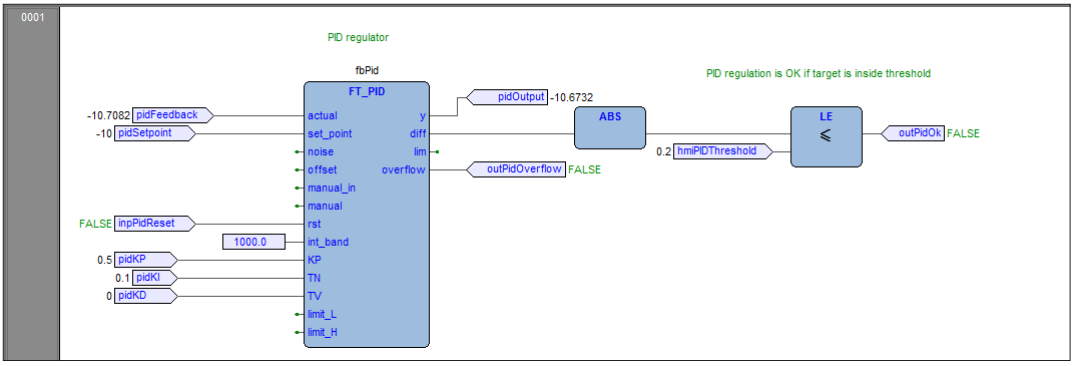
在实时调试模式下，通过突出显示值为真的触点和线圈来调试显示梯形图代码。



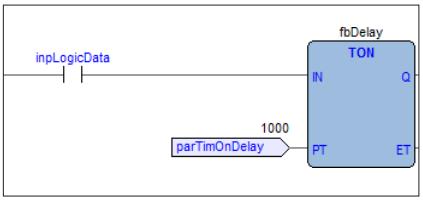
需要注意的是，LD调试管理器会定期测试所有元素的状态。可能发生这样的情况，触点为真的时间太短了以至于无法在调试界面中显示。调试状态下为真的元素不意味着它实际的值没有出现过值为真的情况（可能是采样率太慢没有采集到）。

9.4.3 FBD动画

在实时调试模式下，LogicLab可以在图形源代码编辑器中显示所有可见的变量的值。



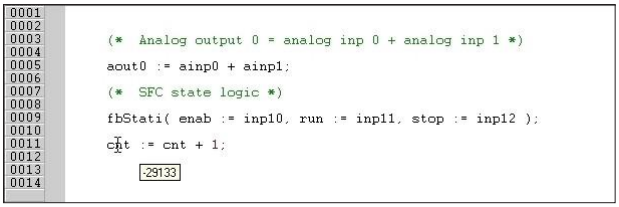
这适用于FBD和LD编程语言。



再次注意的是，这个工具是异步的。

9.4.4 IL和ST调试

实时调试模式也适用于文本源代码编辑器（IL和ST中的一种）。您可以利用鼠标悬停在它上面快速观看变量的值。



9.5 触发器

9.5.1 触发窗口

触发窗口 工具允许您选择一组变量，并在一个特殊的弹出窗口中同步更新它们。

9.5.1.1 打开触发器窗口的先决条件

无需特殊编译

LogicLab调试工具工作在运行时上。因此，与其他编程语言（如C++），并不需要告知编译器是否支持触发窗口：给定PLC代码，编译器的输出是唯一的，且调试和发布版本之间没有区别。

内存可用性

触发窗口应用占用应用程序代码中的固定长度的一段。显然，为了启动一个触发窗口，必须要有足够可用内存，否则将出现错误消息提示。

与图形触发窗口不兼容

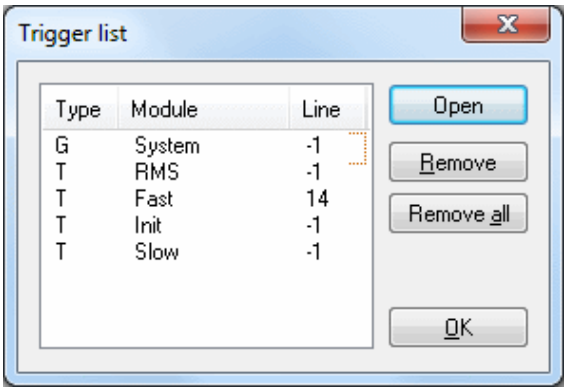
图形触发窗口占用应用程序代码段的所有可用空间。 因此，一旦启动了此调试工具，就无法添加任何触发器窗口，并且如果您尝试启动新窗口，则会出现错误消息。 一旦图形触发窗口最终关闭，触发窗口才将再次可用。

请注意，启动图形触发窗口之前打开所有触发窗口仍然可以正常工作，但是不允许您添加新的触发窗口。

9.5.1.2 触发窗口工具栏

触发窗口图标 *调试* 工具栏的一部分，仅在LogicLab处于调试模式下才可用。

按键	命令	描述
	*设置/删除触发*	为了真正启动触发器窗口，请选择要要在PLC代码中插入的触发点，然后点击此按钮。同样的步骤适用于触发器窗口的删除：为了确定地关闭调试窗口，请单击一次插入触发点的指令或块，然后再次按此按钮。
	*图文轨迹*	此按钮操作与上述 *设置/删除* 触发完全相同，不同之处在于它会打开一个图形触发器窗口。同样，它也可以用于删除图形触发窗口。 快捷键：按 *Shift + F9* 等效于单击 *设置/删除* 触发按钮。
	*删除所有触发器*	按下此键将同时删除所有现有的触发窗口和图形触发窗口。 快捷键：按 *Ctrl + Shift + F9* 等效于单击此按钮。
	*触发器列表*	该键将打开一个对话框，列出所有现有的触发器窗口。快捷键：按 *Ctrl + I* 等效于单击此按钮。

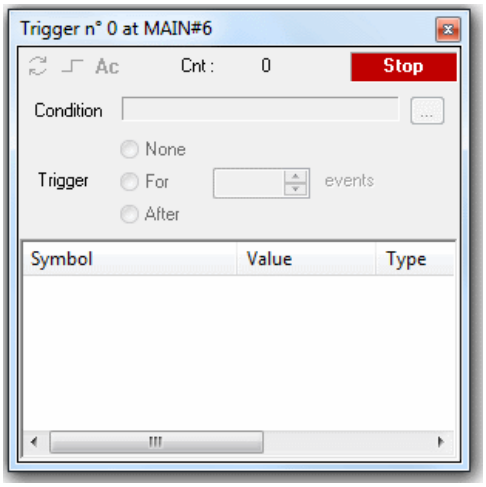


每个记录均指图形或文本的触发窗口。 下表简要说明了每个字段的含义。

领域	描述
类型	***T:*** 触发器窗口。 ***G:*** 图形触发器窗口。
模组	添加触发器的程序，函数或功能块的名称。 如果模块是功能块，则此字段包含其名称，而不是实际放置触发器的实例的名称。
行	对于文本语言（IL，ST）指示触发器所在的行。 对于其他语言，该值始终为-1。

9.5.1.1 触发器窗口界面

设置触发器后将会弹出一个界面：这是访问触发器窗口提供的调试功能的接口。 它由三个元素组成，如下所示。



标题栏

弹出窗口的 *标题* 栏显示有关触发器位置的信息，当处理器到达时，将触发 *变量* 窗口的刷新。

标题 栏中的文本具有以下格式：

Trigger n° X at ModuleName#Location

x	触发标识符
模块名	添加触发器的程序、函数或功能块的名称。

位置	通过模块名来制定触发器的确切位置。
	如果模块名在IL中，则位置具有以下格式：
	`N1`
	否则，如果模块名在FBD中，就变成：
	`N2 $ BT: BID`
	定位：
	`N1` = 指令行号
	`N2` = 网络模块编号
	`BT` = 块类型（操作数、功能、功能块等）
	`BID` = 块标识符

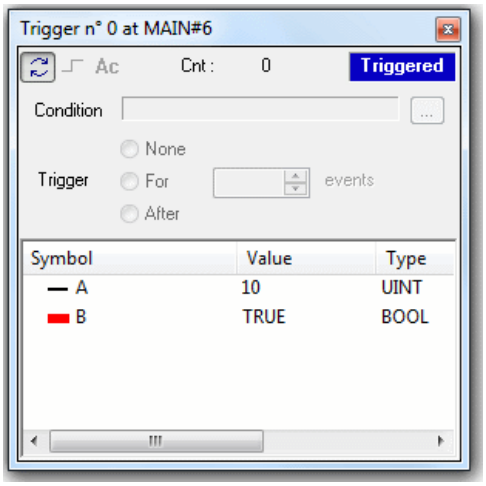
控制部分

该对话框使用户可以更好地控制触发器窗口的刷新，以获取有关范围内代码的更多信息。 在 *触发* 窗口控件部分中，将详细介绍每个控件的功能（请参阅第9.5.2.11节）。

只有将至少一个变量拖到调试窗口中之后，才能访问除Ac（*累加器显示* 按钮）以外的所有控件。

变量部分

调试 窗口的下半部分是一个表格，其中包含您拖入的每个变量的一行。每行具有四个字段：在最后一次刷新期间从内存中读取的变量的名称、值、类型及其位置（@task：ModuleName） 。



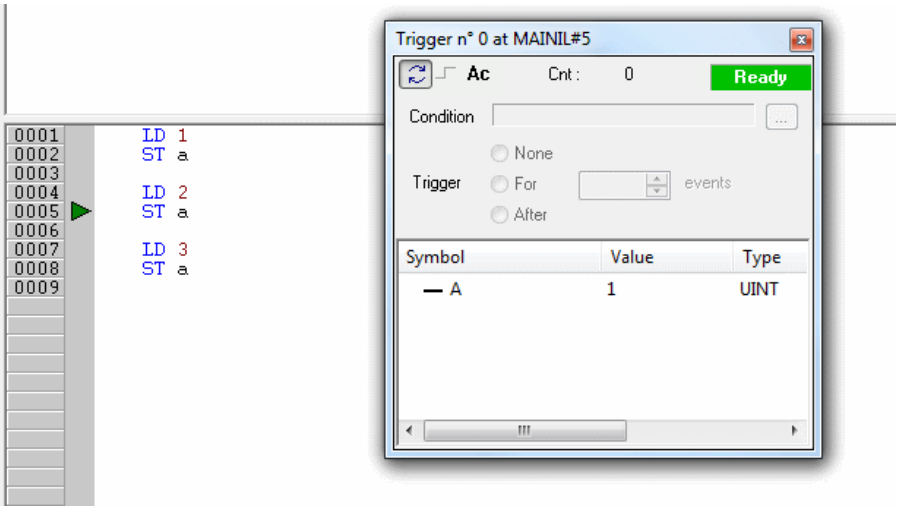
9.5.1.2 触发器窗口：拖放信息

要监视变量，您需要将其复制到 *调试* 窗口的下部。

此部分是一个表格，其中包含您拖入的每个变量的一行。您只能将放置相对触发器的模块本地变量、全局变量或参数拖到触发器窗口中。 您不能拖动在另一个程序、函数或功能块中声明的变量。

9.5.1.3 刷新值

我们将通过下面的例子来说明 *刷新值* 。



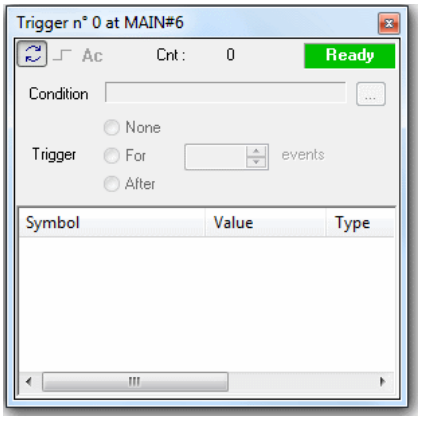
每次处理器执行绿色箭头标记的指令时。 当然，您可以设置控件仅在触发器满足您定义的更多限制条件时刷新此变量。

请注意，在标记的指令（在这种情况下：第5行的指令）之前也就是前一条指令（第4行的指令）执行之后，从内存中读取 *符号列表* 中变量的值。

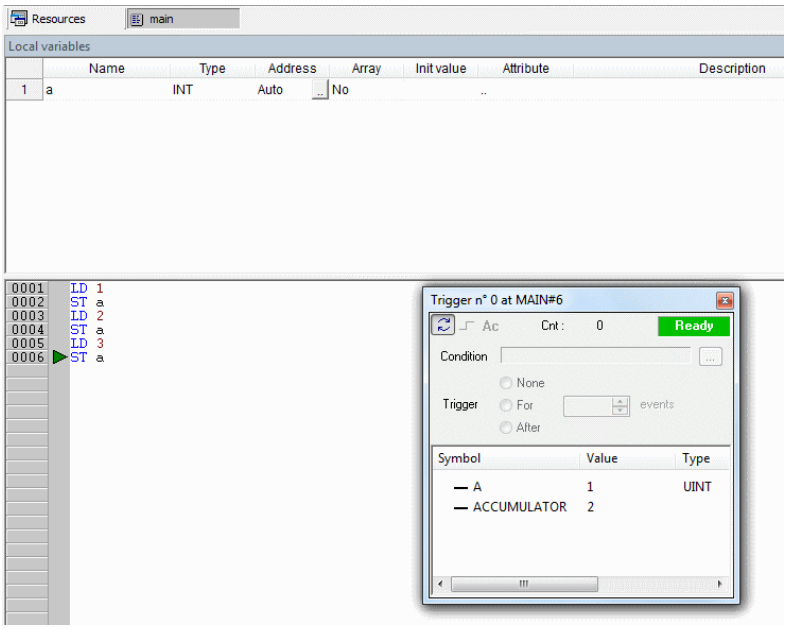
因此，在上面的示例中，当从内存中读取新的a值并将其显示在触发窗口中时，第二ST语句尚未执行。因此，第二个ST a的结果为1。

9.5.1.4 触发窗口控件

本段将介绍触发器窗口控件，它使您可以更好地使用调试工具，以获取有关范围内代码的更多信息。
无论在何种模块类型（IL或FBD）中插入触发器，触发器窗口控件的作用与行为都会是明确一致的。
此外，只有将至少一个变量拖到 *变量* 窗口中，才能访问除 *累加器显示* 以外的所有控件。
用户可以通过调试窗口的灰色上半部分访问窗口控件。



按键	命令	描述
	*启动/停止*	此控件用于启动触发会话。 如果系统正在触发，则可以单击此按钮强制停止。 否则，达到条件时会话将自动停止。 此时，您可以按此按钮开始另一个触发会话。
	*单步执行*	此控件用于执行单步触发器。 仅当没有活动的触发会话并且未选择 *无* 时才启用。 考虑指定条件，单步触发完成后，触发会话将自动停止。
	*累加器显示*	该控件将 *累加器* 添加到已经拖入触发窗口的变量列表中。在变量表的底部添加了新行，其中在 *符号列* 中包含字符串 *累加器* ，累加器的值和类型在 *值* 列中未指定，而 *位置* 设定为 *全局* ，如下图所示。



为了从表中删除累加器，请在 *符号列* 中单击其名称，然后按Del键。

如果在ST语句之前插入了触发器，此控件将非常有用，因为它使您可以知道在当前执行任务期间在目标变量中写入了什么值。 通过将触发器移到绿色箭头所标记的指令之后，也可以得到相同的结果。

触发计数器

窗口中此只读控件将计算调试窗口管理器被触发的次数。

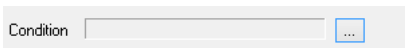
每次启动新的触发会话时，窗口管理器都会自动重置此计数器。

触发状态

此只读控件向用户显示 *调试* 窗口的状态。它可能出现以下值。

Ready	当前任务执行期间未触发。
Triggered	在当前任务执行期间发生了触发动作。
Stop	系统未触发。触发尚未开始或已被用户停止或已达到停止条件。
Error	与目标设备的通信中断，无法确定触发窗口的状态。

用户定义的条件



如果使用此控件定义一个条件，则每次调用此窗口管理器且用户定义的条件为真时，都会刷新 *调试* 窗口中的值。

输入条件后，控件将显示其简化表达式。

计数器

这些控件允许用户在触发计数器上定义条件。

触发窗口可以处于以下三种状态之一。

- ***无***：没有启动计数器，因此未在触发器上指定任何条件。
- ***设定***：假设您给计数器限制值N，则每次触发调试窗口时，窗口管理器都会在计数器的当前值上加1并刷新其变量的值。 但是，当计数器等于N时，窗口将停止刷新值，并变为停止状态。
- ***之后***：假设您给计数器限制值N，则窗口管理器将重置计数器，并在每次触发时将其当前值加1。 窗口保持在 *就绪* 状态，并且直到计数器达到N时才更新其变量的值。

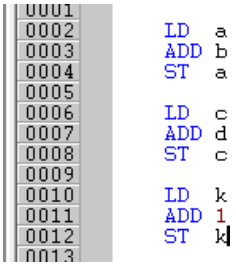
9.5.2 触发窗口调试

9.5.2.1 介绍

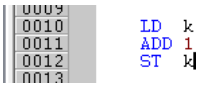
触发窗口工具允许用户选择一组变量，并在弹出窗口中同步显示和更新它们的值。与 *监视* 窗口不同，每次触发时，触发窗口都会同时刷新它们包含的所有变量。

9.5.2.2 在IL语句中打开触发窗口

假设您有一个IL模块，其中还包含如下代码。



如果您想在执行ST k指令之前就知道b，d和k的值。为此，将光标移到第12行。



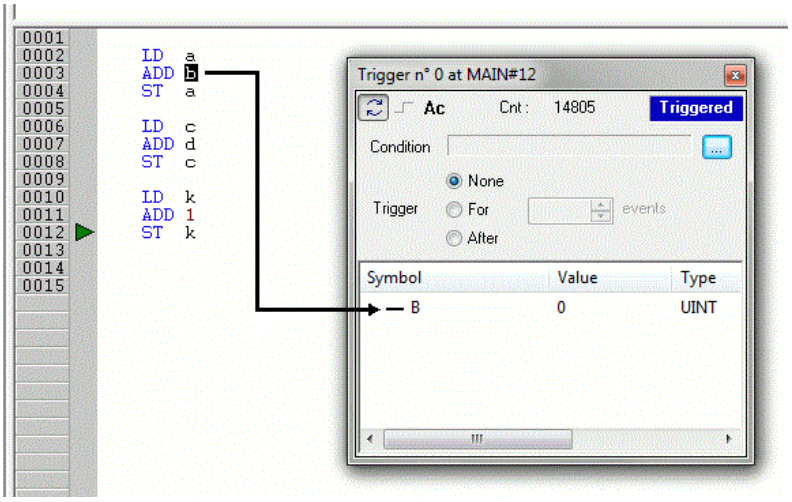
然后您可以点击 *调试>设置/删除触发*

在这两种情况下，行号旁边都会出现一个绿色箭头，并弹出相关的触发窗口。

并非所有的IL指令都支持触发器。例如，不允许在包含JMP语句的行的开头放置触发器。

9.5.2.3 从IL语句向触发窗口添加变量

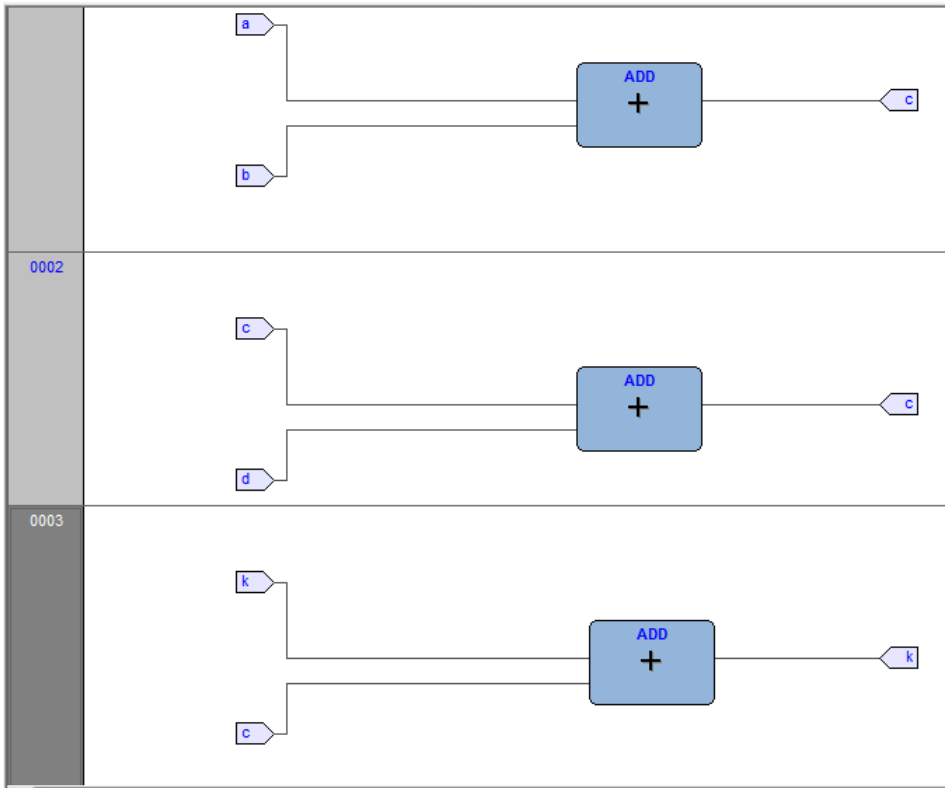
为了观察变量的值，您需要将其添加到触发窗口。为此，在IL语句中通过双击选择一个变量，然后将其拖动到 *变量* 窗口中，该窗口位于弹出窗口中下方。变量的名称将显示在 *符号* 列表中。



同样的过程适用于您需要需要监视的变量。

9.5.2.4 在FBD语句打开触发窗口

假设您有一个FBD模块，其中还包含如下代码。



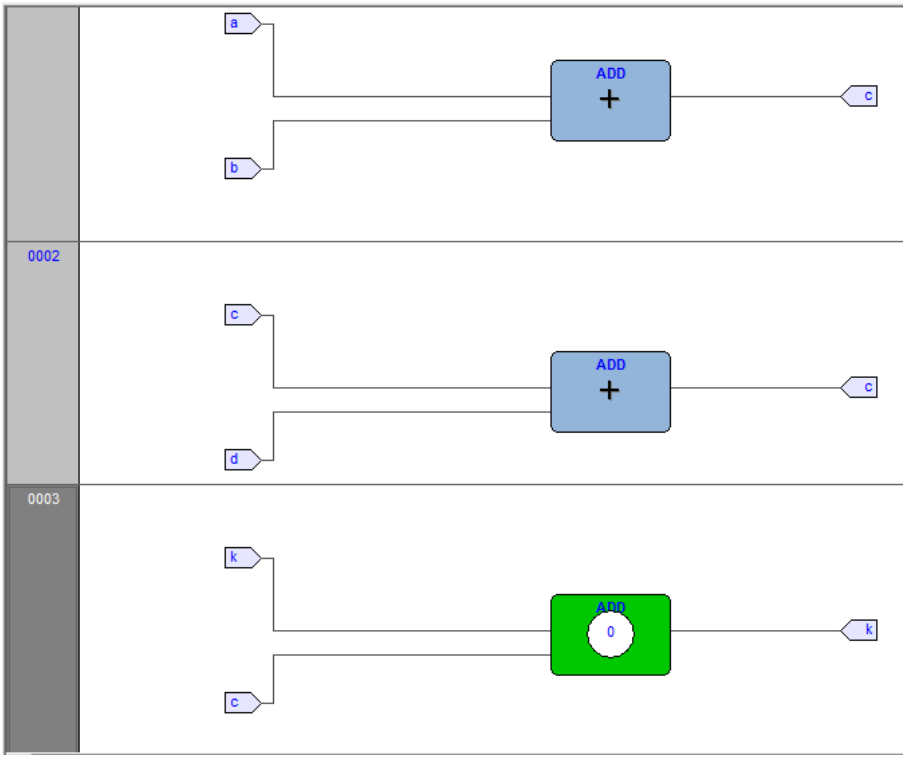
我们还假设您想在执行ST k语句之前知道c、d和k的值。

前提是您绝不能在表示变量的块中放置触发器，例如必须选择所选变量之前的第一个可用块。

在上述示例中，必须将光标移至网络3，然后单击ADD块。

您可以点击 *调试>设置/删除触发*

在这两种情况下，所选块的颜色变为绿色，在块的中间出现一个带有数字的白色圆圈，并弹出相关的触发窗口。



在预处理FBD源代码时，编译器将其转换为IL指令。网络3中的ADD指令扩展为：

```
LD k
ADD 1
ST k
```

将触发器添加到FBD块时，实际上是将触发器放置在其IL等效代码的第一条语句之前。

9.5.2.5 从FBD语句向触发窗口添加变量

为了观察变量的值，您需要将其添加到触发窗口。让我们假设您想在下面的图中检查FBD代码的变量k的值。

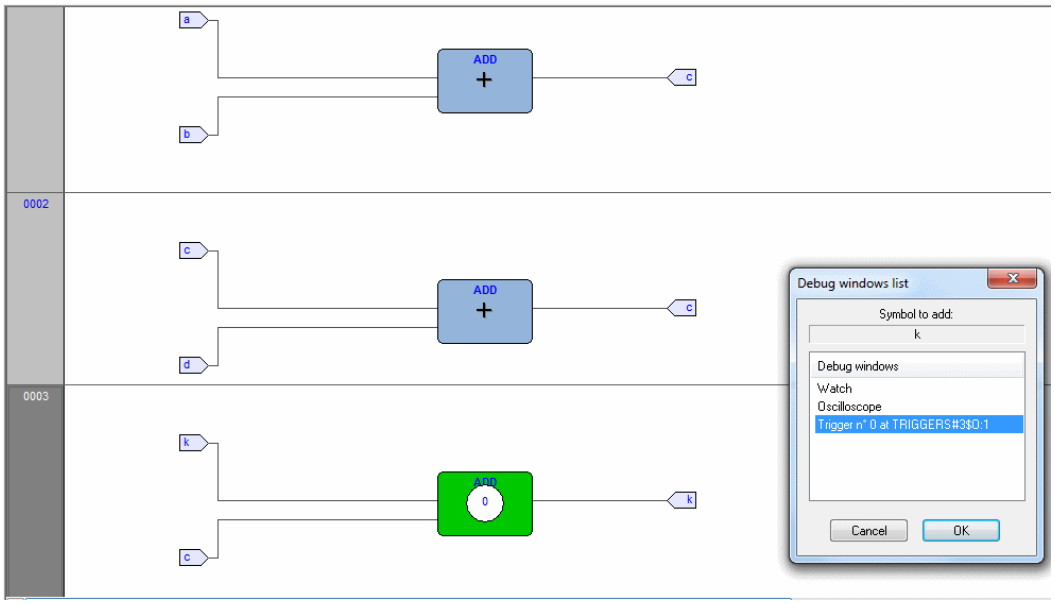
为此，请点击 *编辑>观看模式*

光标会变成如右图所示。

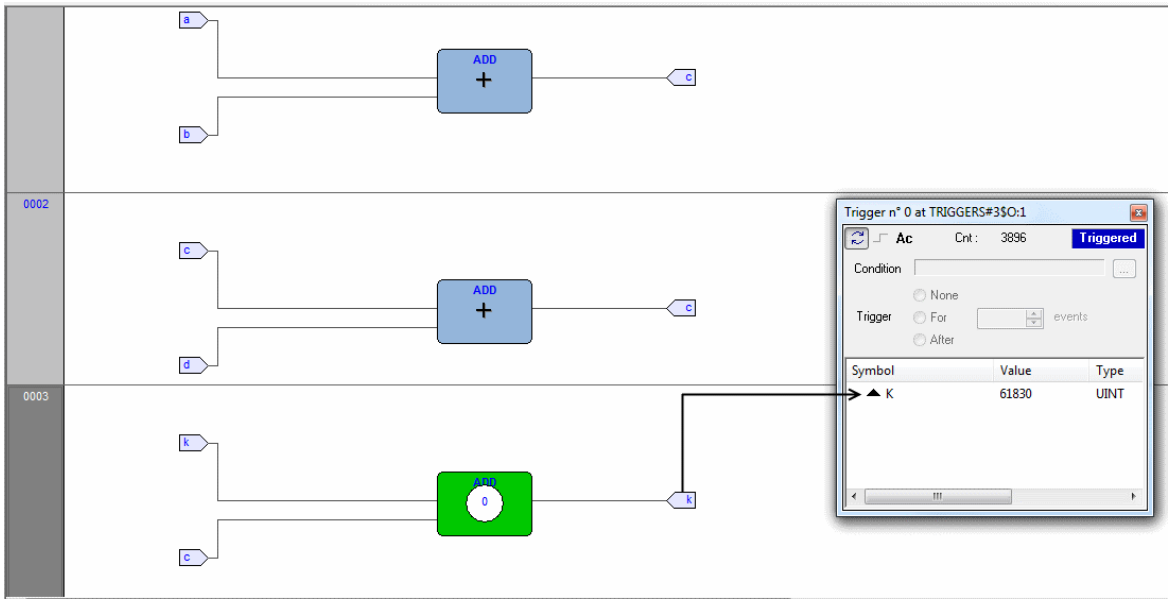
现在，您可以单击代表希望在触发窗口中显示的变量的块。

在我们正在考虑的示例中，单击按钮块。

此时将会出现一个对话框，列出所有当前的调试窗口实例，并询问您哪个是您刚刚单击的对象。



为了在触发窗口中显示变量k，请在 *调试窗口* 列中选择其引用，然后按OK。现在，变量名称将显示在 *符号* 列中。

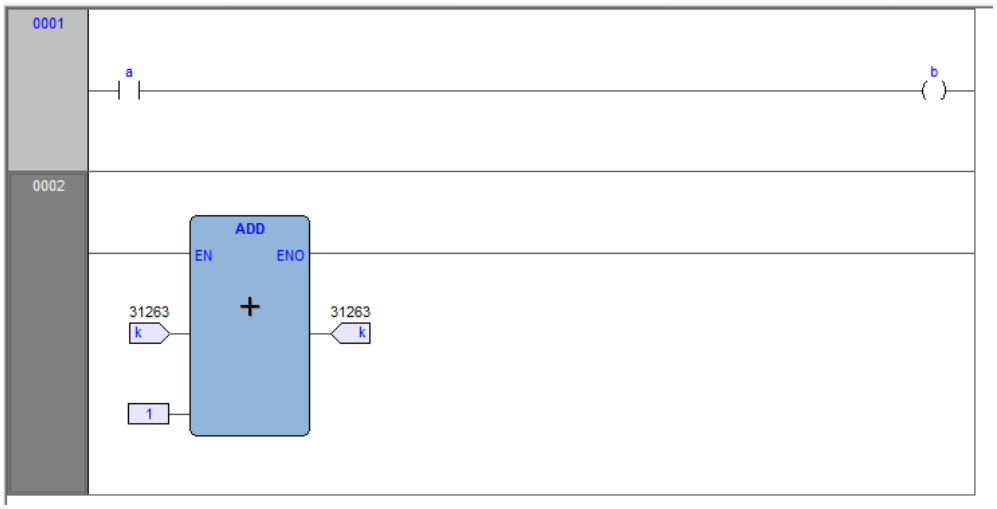


相同的过程适用于您要监视的所有变量。

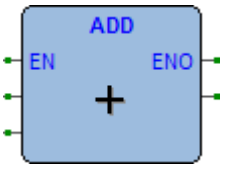
将要观察的所有变量添加到 *图形* 监视窗口后，可以单击 *编辑>插入/移动模式* ，以使光标恢复其原始形状。

9.5.2.6 在LD语句中打开触发窗口

假设您有一个LD模块，其中还包含如下代码。



您可以将触发器放置在如下所示的块上。

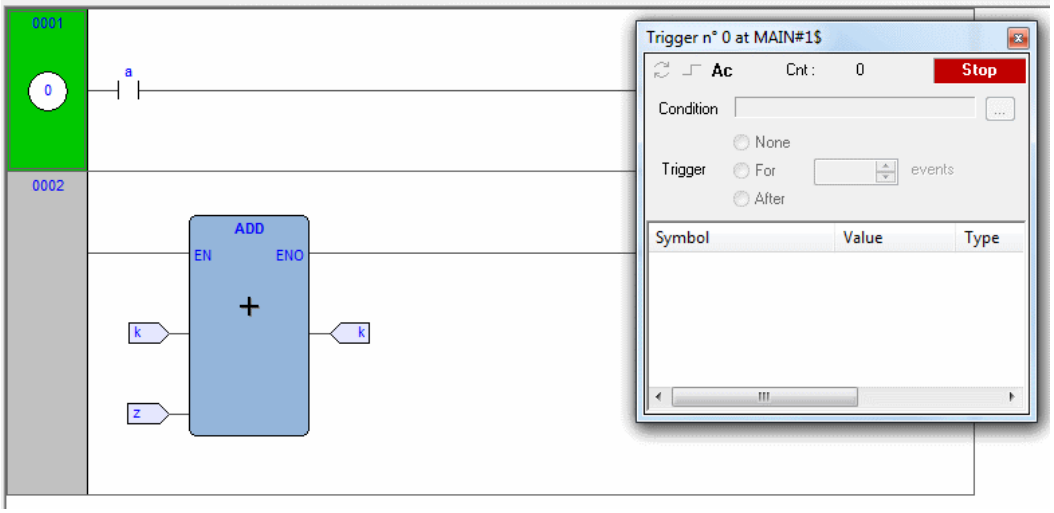


在这种情况下，同样的规则适用于FBD模块触点上插入一个触发器

或者线圈

在这种情况下，请遵循SE说明。假设您希望每次处理器到达网络编号1时都知道一些变量的值。首先，您必须单击组成网络号1的项目之一。现在，您可以单击 *调试>添加/删除文本触发器*

在这两种情况下，包含网络号的灰色凸起按钮都将变为绿色，并且在按钮的中间会出现一个带有内部触发器编号的白色圆圈，同时会弹出相关的触发器窗口。



与LogicLab支持的其他语言不同，LD不允许您将触发器插入单个触点或线圈，因为它只能选择整个网络。 因此，每次处理器到达所选网络的起点时，触发器窗口中的变量都会刷新。

9.5.2.7 从LD语句向触发窗口添加变量

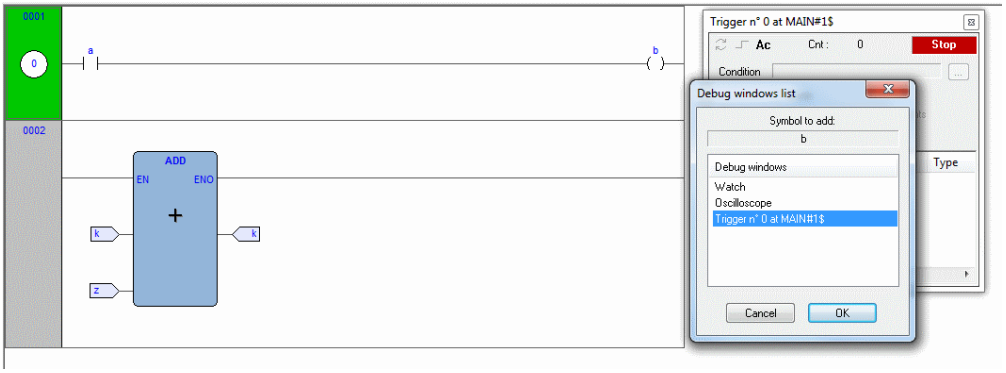
为了观察变量的值，您需要将其添加到触发窗口。假设您要检查下面图所示的LD代码中的变量b的值。

为此，请点击 *编辑>观看模式*

光标会变成如右图所示。

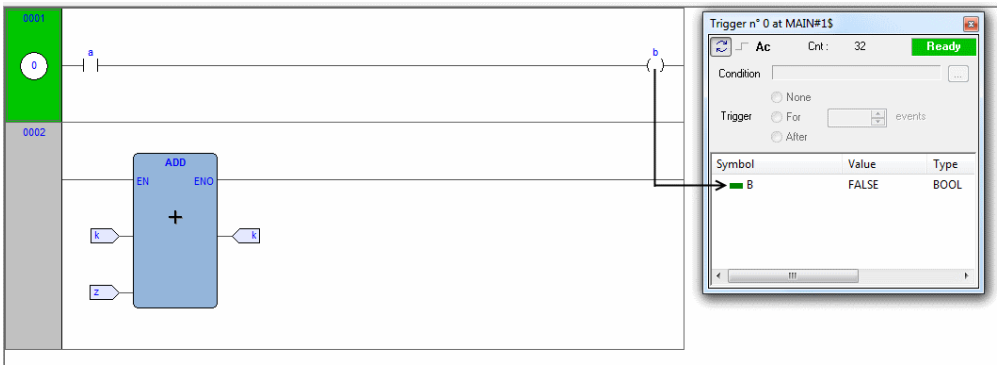
现在，可以单击您希望在触发窗口中显示的变量的项目。

LogicLab将会弹出一个对话框，列出所有当前调试窗口实例，并询问您哪个是您刚刚单击的对象。



为了在触发窗口中显示变量B，请在 *调试窗口* 列中选择其引用，然后点击 *确定* 。

现在，变量名称将显示在 *符号列表* 中。



相同的过程适用于您要检查的所有变量。

将要观察的所有变量添加到 *图形* 监视窗口后，可以单击 *编辑>插入/移动模式* ，以使光标恢复其原始形状。

9.5.2.8 在ST语句打开触发窗口

假设您有一个ST模块，其中还包含如下代码。

```
0001
0002    a := b * b;
0003    c := c + SHR( a, 16#04 );
0004
0005    d := e * e;
0006    f := f + SHR( d, 16#04 );
0007
```

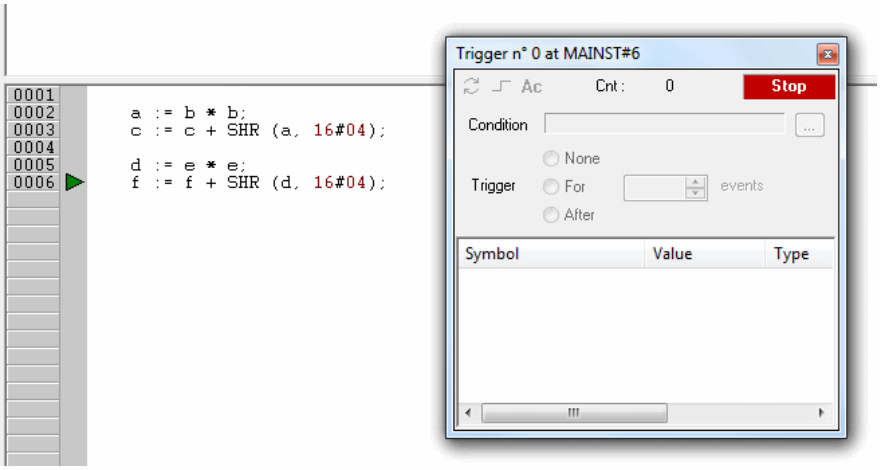
假设您想在执行下面的指令之前知道e、d和f的值

```
F: = F + SHR（d，16＃04）
```

为此，将光标移到第6行。

接着您可以点击 *调试>添加/删除文本触发器*

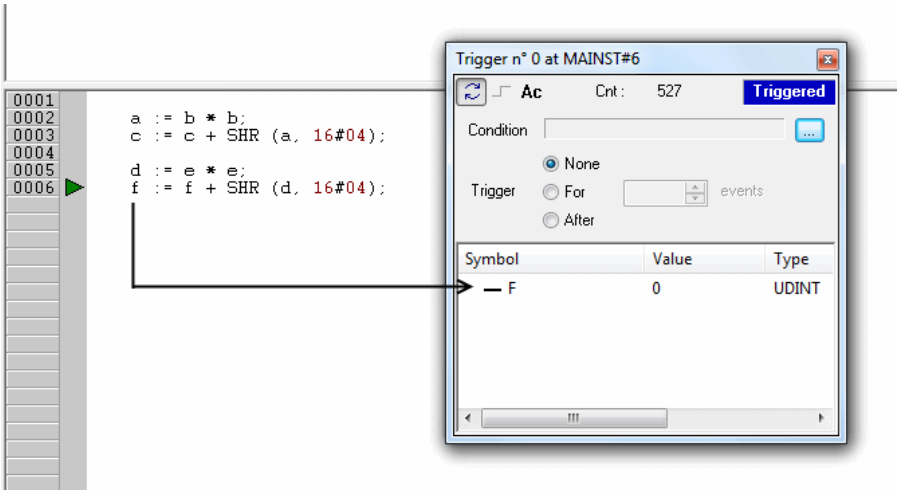
在这两种情况下，行号旁边都会出现一个绿色箭头，并弹出相关的触发窗口。



并非所有的ST指令支持触发器。例如，无法在包含终止符（例如END_IF、END_FOR、END_WHILE等）的行上放置触发器。

9.5.2.9 从ST语句向触发窗口添加变量

为了观察变量的值，您需要将其添加到触发窗口。为此，在ST语句中通过双击选择一个变量，然后将其拖动到 *变量* 窗口中，该窗口位于弹出窗口中下方。变量的名称将显示在 *符号* 列表中。



同样的过程适用于所有您要监视的变量。

9.5.2.10 从触发窗口中删除变量

如果您不想在触发窗口中再显示一个变量，请单击一次它的名称来选择它，然后点击Del键。

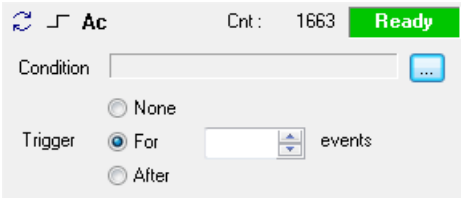
9.5.2.11 触发窗口控件的使用

本段将介绍触发器窗口控件，它使您可以更好地使用调试工具，以获取有关范围内代码的更多信息。触发窗口控件的主要目的是让您定义更多的限制条件，以便在处理器到达触发位置并且满足这些条件时刷新 *变量* 窗口中的变量。 如果不使用控件，则每次处理器到达相对触发器时，变量也会刷新。

启用控件

设置触发器时， *控制窗口* 中的所有元素均显示为禁用状态。

实际上，除非将至少一个变量拖到 *调试窗口* 中，否则您无法访问除 *累加器* 显示之外的任何控件。发生这种情况时，触发将自动开始，并且 *控件窗口* 将作如下更改。



可以通过相应的按钮开始/停止触发。

固定刷新的次数

如果要在第一次触发窗口时刷新值，请选择 *无* ，然后按单步按钮，或者选择 *设定* ，并将计数器设置为1。

如果要在触发窗口第X次时刷新值，请将计数器设置为X，然后选择 *设定* 。

如果要在触发窗口第Y次后刷新值，请将计数器设置为Y，然后选择 *之后* 。

当（重新）开始触发时，触发条件设置将变为原始设置。

查看累加器

如 *刷新值* 部分所述（请参见第9.5.1.5节），当您在指令行上插入触发器时，可以确定在执行指令本身之前，每次处理器到达该位置时，相对调试窗口中的变量都将更新。

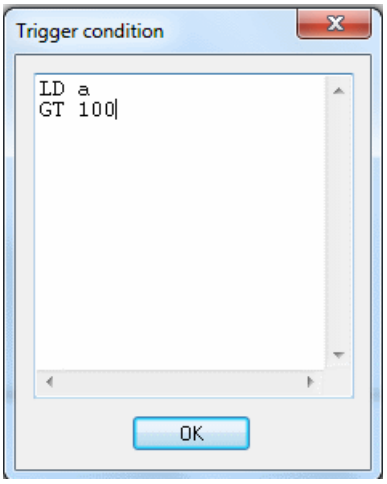
在某些情况下，例如当触发器放在ST语句之前时，了解累加器的值可能会很有用。这使您可以预测在更新触发窗口中的所有变量之后将要执行的指令的结果。要将累加器添加到触发窗口，请单击累加器显示按钮。

定义条件

通过此控件，用户可以设置触发条件。默认情况下，此条件设置为 *真* ，并且每次触发窗口管理器时都会刷新调试窗口中的值。

如果要限制刷新机制，则可以通过单击适当的按钮来指定条件。

此时，将会弹出一个文本窗口，您可以在其中编写IL代码来设置条件。



完成条件代码的编写后，请单击 *确定* 按钮进行安装，或按 *取消* 按钮退出操作。如果选择加载，则每次触发窗口管理器并且用户定义的条件为 *真* 时，调试窗口中的值都会刷新。

此时，条件的简化表达式将会出现在控件中。

如要对其进行修改，请再次按上述按钮。

此时将出现文本窗口，其中包含您最初编写的文本，您现在可以对其进行编辑。

要完全删除用户定义的条件，请在文本窗口中删除整个IL代码，然后单击 *确定* 按钮。

执行条件代码后，累加器必须为布尔类型（TRUE或FALSE），否则会发生编译器错误。

条件代码中只能使用全局变量和拖入的变量。即如果尚未将其拖放到调试窗口中，则最初在其上插入触发器的模块本地的所有变量均不在范围内，且在条件窗口中不能声明任何新变量。

9.5.2.12 关闭触发窗口并移除触发器

下面介绍了使用触发窗口完成调试会话时可以执行的操作。 您可以选择以下选项。

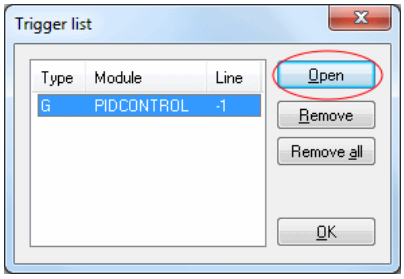
- 关闭触发窗口
- 删除触发器
- 删除所有的触发器

请注意，上面列出的操作产生不同的效果。

关闭触发窗口

如果已经通过触发器窗口完成了对一组变量的监视，则可能要关闭 *调试* 窗口，而不必删除触发器。如果单击右上角的按钮，则仅隐藏界面窗口，而窗口管理器和相关触发器均仍然保持工作。

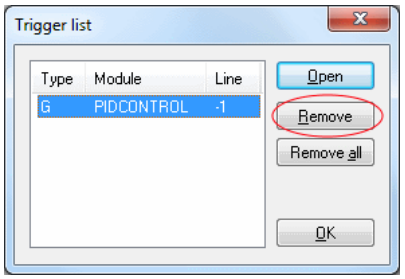
实际上，如果以后要使用先前隐藏的触发窗口恢复调试，则只需打开 *触发器列表* 窗口，选择引用该触发窗口的记录，然后单击 *打开* 按钮。



此时会弹出一个界面窗口，其中包含变量的值，并且触发计数器已更新，就好像尚未关闭一样。

删除触发器

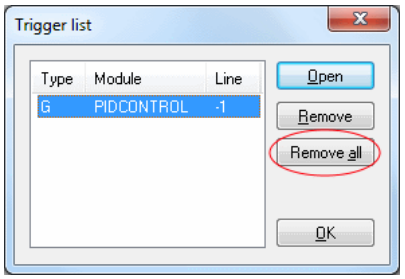
如果选择此选项，则将窗口管理器及其触发器的代码完全删除。为此，只需打开 *触发器列表* 窗口，选择要消除的与触发器窗口相关的记录，然后单击 *删除* 按钮。



或者，您可以将光标移动到该行（如果位于IL或ST模块中），或单击放置触发器的块（如果位于FBD或LD模块中）。请按 *调试* 工具栏中的 *设置/删除触发器* 按钮。

删除所有触发器

此外，您也可以同时删除所有现有的触发器，不管选择哪条记录，只要单击 *删除所有* 按钮即可。



9.6 图形触发器

9.6.1 图形触发窗口

图形触发窗口工具允许您选择一组变量，并对其进行同步采样，并在弹出的窗口中显示其曲线。

每次处理器到达您放置触发器的位置（即，如果为IL，ST则为指令，如果为FBD，LD则为块）时，都会对拖入变量进行采样。

9.6.1.1 打开图形触发窗口的前提条件

无需特殊编译

LogicLab调试工具工作在运行时上。因此，与其他编程语言（如C++），并不需要告知编译器是否支持图形触发窗口：给定PLC代码，编译器的输出是唯一的，且调试和发布版本之间没有区别。

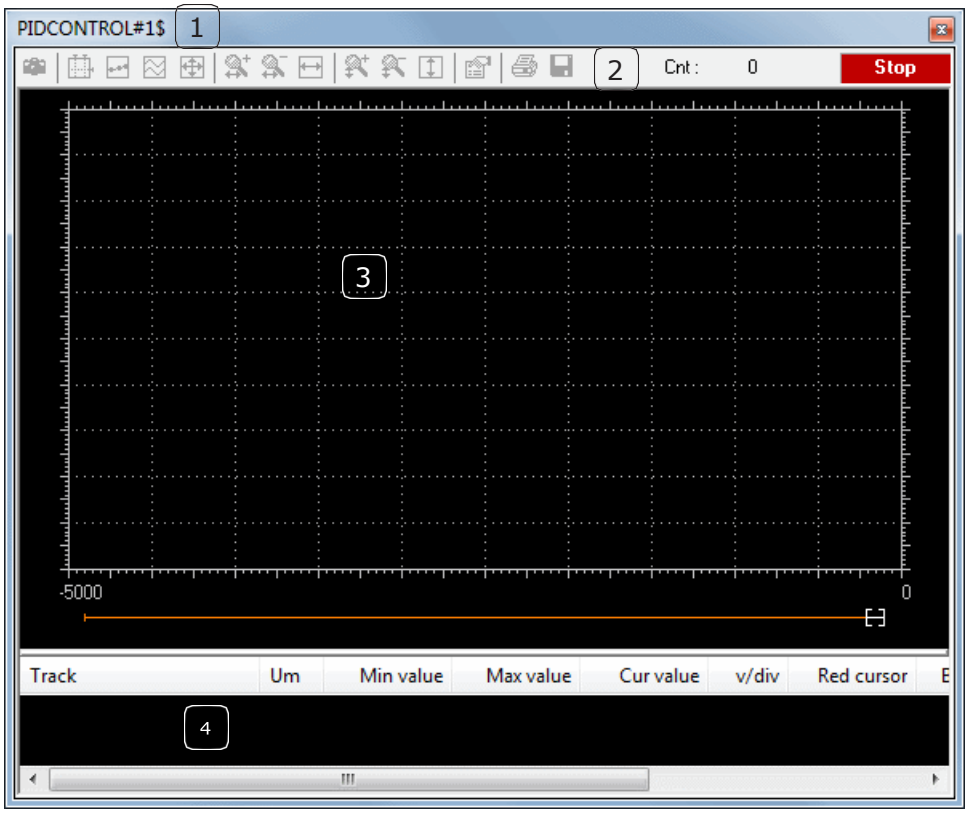
内存可用性

图形触发窗口将会占用应用程序代码扇区中的所有可用内存空间。

显然，为了启动触发窗口，必须有足够的可用内存，否则会出现错误消息。

9.6.1.2 图形触发窗口界面

设置图形触发器后将会出现一个弹出窗口，称为 *界面* 窗口。这是访问图形触发窗口提供的调试功能的主界面。它由如下几个元素组成：



***1.** 标题栏* ***2.** 控制栏* ***3.** 图表区域* ***4.** 变量窗口*

标题栏

弹出窗口顶部的标题栏显示有关触发器位置的信息，该信息会导致对 *变量* 窗口中列出的变量进行采样。标题中的文本具有以下格式：

```
ModuleName#Location
```

模块名	添加触发器的程序、函数或功能块的名称。
位置	通过模块名来制定触发器的确切位置。 如果模块名在IL、ST中，则位置具有以下格式： N1 否则，如果模块名在FBD、LD中，就变成： N2 $ BT：BID 定位： N1 = 指令行号 N2 = 网络模块编号 BT = 块类型（操作数、功能、功能块等） BID = 块标识符

控件栏

该对话框使您可以更好地控制图形触发窗口的工作。图形触发窗口控件部分中提供了每个控件功能的详细说明（请参见第9.6.1.5节）。

图表区

图表区域包含以下六项：

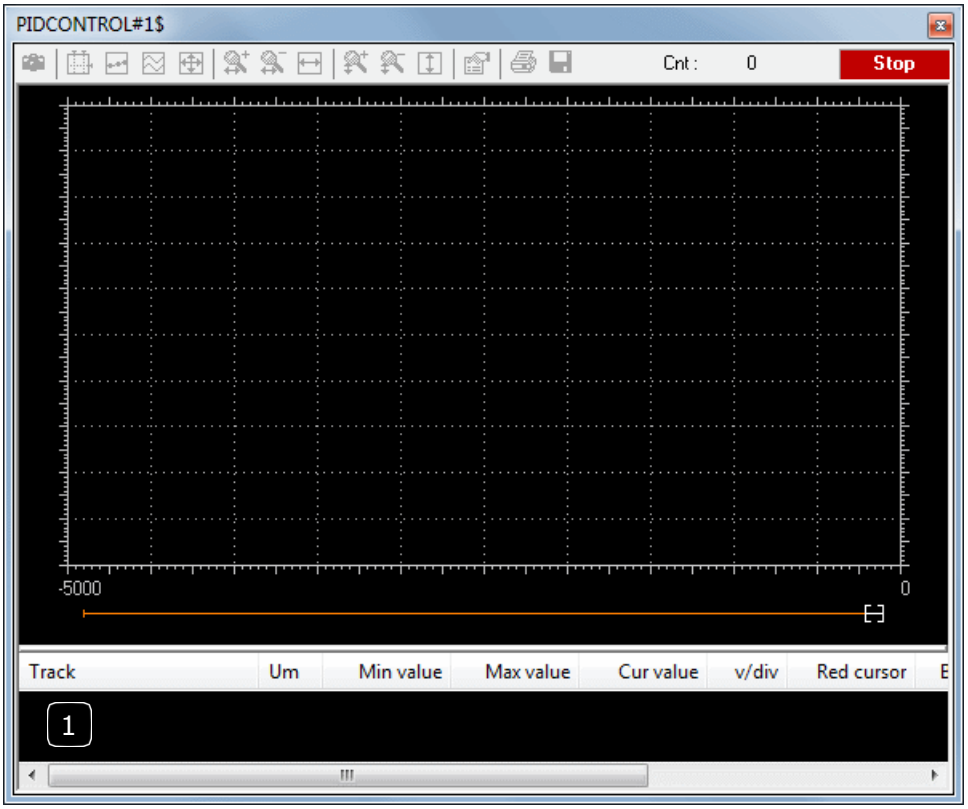
1) 绘图：包含所拖入变量的曲线的实际绘图的区域。
2) 样品采集：通过图形触发窗口管理器采集的样本数量。
3) 水平光标：标识一条水平线的光标。在此行的交点处的每个变量的值在列水平光标中报告。
4) 蓝色光标：标识垂直线的光标。与该行相交处的每个变量的值都在左侧光标列中报告。
5) 红色光标：与蓝色光标相同。
6) 滚动条：如果x轴的比例太大而无法在 *绘图* 区域中显示所有样本，则滚动条可让您沿水平轴来回滑动。

变量窗口

调试 窗口的下半部分是一个表，其中的行包含您拖入的每个变量。每一行都有多个字段，在 *拖放信息* 部分中对此进行了详细说明。

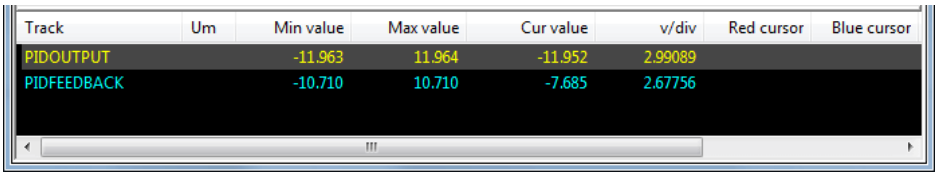
9.6.1.3 图形触发窗口：拖放信息

要监视变量，您需要将其复制到 *调试* 窗口的下方。



1. *变量窗口*

调试 窗口的下方是一个表格，其中包含您拖入的每个变量的一行。每一行都有多个字段，如下图所示。



对象	描述
名称	变量的名称。
单位	测量单位。
最小值	在记录设定的最低值。
最大值	在记录设定的最大值。
当前值	该变量的当前值。
v/Div	y轴的单位（即垂直轴上两个刻度之间的间隔）代表多少工程单位。
蓝色光标	变量与蓝色光标标识的线相交处的值。
红色光标	变量与红色光标标识的线相交处的值。
水平调整光标	与水平光标标识的线相交处的变量值。

请注意，您只能将放置相对触发器的模块本地变量，全局变量或参数拖到图形触发器窗口中。您不能拖动在另一个程序，函数或功能块中声明的变量。

9.6.1.4 变量抽样

让我们看看下面的例子。

每次触发窗口管理器时，即每次处理器执行由绿色箭头标记的指令时，都会对变量的值进行采样。但是，您可以设置控件，以便在触发器还满足您定义的其他限制条件时对变量进行采样。

列 *跟踪* 中变量的值在标记的指令之前和前一条指令之后将会立即从内存中读取。

9.6.1.5 图形触发窗口控件

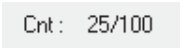
本段讨论 *图形* 触发窗口的控件。通过控件，您可以详细指定何时应使用LogicLab采样添加到 *变量* 窗口的变量。

无论在何种模块类型（IL，ST，FBD或LD）中插入触发器，触发器窗口控件的作用与行为都会是明确一致的。

通过调试窗口的控件栏，用户可以访问窗口控件。

按键	命令	描述
	*启动图形跟踪*	当您按下该按钮时，就可以开始采集。如果是正在运行采集过程中释放此按钮，则将停止采样过程，并重置到目前为止已采集的所有数据。
	*开启/关闭游标*	只要按下此按钮，就可以看到两个光标（红色光标，蓝色光标）并沿其轴移动。如果要同时隐藏所有光标，请释放此按钮。
	*显示样本*	此控件用于证明在每个样本上触发变量的确切位置。
	*拆分变量*	按下此控件时，将y轴分为与拖动的变量一样多的段，以便将每个变量的图绘制在单独的区域中。
	*显示所有值*	它用于在图形窗口中显示当前记录集中选定变量的所有采样值。
	*水平放大和缩小*	放大是一种使 *图表* 区域中的曲线在屏幕上显得更大的操作，以便可以查看更多细节。 缩小是使曲线在屏幕上显得较小的操作，以便可以完整查看。 水平缩放仅作用于水平轴。
	*横向显示所有*	此控件用于水平居中记录集样本。 因此，第一个样本将放置在图形窗口的左边缘，最后一个样本将放置在图形窗口的右边缘。
	*垂直放大和缩小*	垂直缩放仅作用于垂直轴。
	*垂直显示全部*	此控件用于垂直居中记录集样本。 因此，最大值样本将放置在顶部边缘附近，而低值样本将放置在图形窗口的底部边缘。
	*图形触发窗口属性*	按下此按钮将显示一个选项卡对话框，您可以使用该对话框设置影响图形触发窗口操作的常规用户选项。 由于可以设置的选项很多，因此将利用专门的一节来说明。
	*打印图表*	按此按钮可同时打印 *图表* 区域和 *变量* 窗口。
	*保存图表*	按下此按钮保存图表。

触发器计数



此只读控件以以下格式显示两个数字：X / Y。

X 表示自从加载图形触发器以来，调试窗口管理器已被触发多少次。

Y 代表图形窗口在停止数据采集和绘制曲线之前必须采集的样本数。

触发器状态

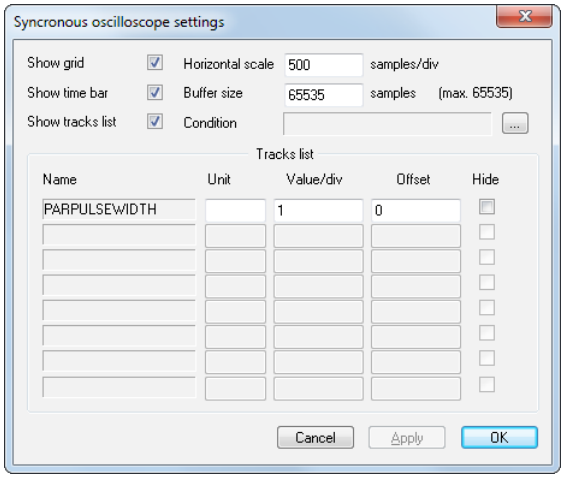
此只读控件向您显示 *调试* 窗口的状态。它可能是以下值。

Ready	没有采样，因为在当前任务执行期间未发生触发事件。
Triggered	在当前任务执行过程中触发发生时收集到了样本。
Stop	触发计数器指示已收集了满足用户请求或内存限制的大量样本，因此停止了获取过程。
Error	与目标设备的通信中断，无法确定触发窗口的状态。

9.6.1.6 图形触发窗口选项

为了打开 *选项* 选项卡，必须单击 *控制* 栏中的 *属性* 按钮。执行此操作时，将出现以下对话框。

一般

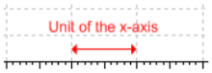


控制

控制	描述
显示网格	勾选此控件将在 图表 区域背景中显示网格。
显示时间栏	只有选中此选项，图表区域底部的滚动条才会可用。

控制	描述
显示追踪列表	只要选中此选项，就会显示 *变量* 窗口，否则 *图表* 区域将扩展到图形触发窗口的底部。

值

控制	描述
水平刻度 	x轴单位的样本数。以x轴为单位，空间是指背景网格的两条垂直线之间。
缓冲区大小	要获取的样本数。当打开选项卡时，将要监视的所有变量拖入后，您可以在此字段中读取默认数字，该数字代表可以为每个变量收集的最大样本数。因此，您可以输入一个小于或等于默认数字的数字。

追踪

此选项卡允许您定义每个变量曲线的图形属性。
要选择一个变量，请在 *追踪列表* 中单击其名称。

控制	描述
单元	度量单位，打印在 *变量* 窗口的表中。
Value/Div	y轴的单位值。y轴单位是指背景网格的两条水平线之间的间隔。
隐藏	选中此标志可隐藏图形上的选定轨迹。

点击 *应用* 使更改生效，或按 *确定* 以应用更改并关闭 *选项* 选项卡。

用户定义条件

如果使用此控件定义条件，则直到满足该条件后才开始采样过程。请注意，与触发窗口不同，一旦开始数据采集，则每次触发窗口管理器时都将采样，无论用户条件是否为真。

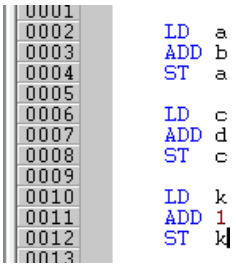
输入条件后，控件将显示其简化表达式。

9.6.2 图形触发窗口调试

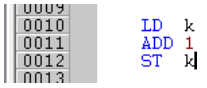
图形触发窗口工具允许您选择一组变量，并让他们同步采样，并显示在一个特殊的弹出窗口的曲线。

9.6.2.1 在IL语句中打开图形触发窗口

假设您有一个IL模块，其中还包含如下代码。

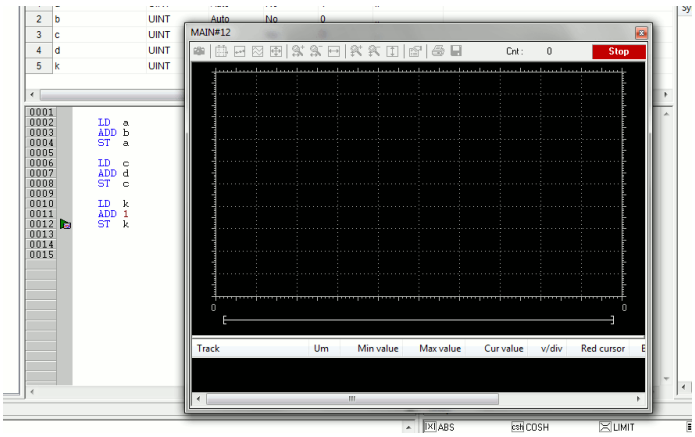


如果您想在执行ST k指令之前就知道b，d和k的值。为此，将光标移到第12行。



然后点击 *调试>添加/删除图形触发*

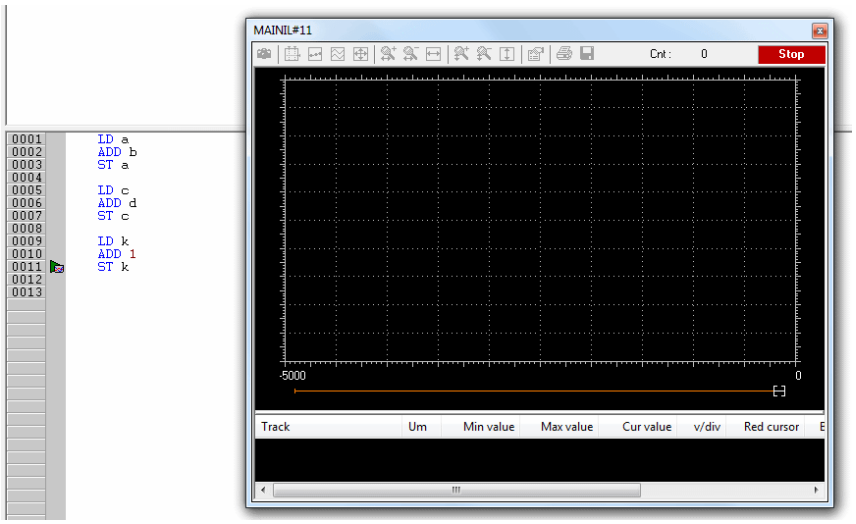
行号旁边将会出现一个绿色箭头，并弹出相关的触发窗口。



并非所有的IL指令都支持触发器。例如，不允许在包含JMP语句的行的开头放置触发器。

9.6.2.2 从IL语句向图形触发窗口添加变量

为了获得绘制的变量曲线，您需要将其添加到图形触发器窗口。为此，通过双击选择一个变量，然后将其拖动到 *变量* 窗口中，变量将会显示在 *跟踪列表* 中。

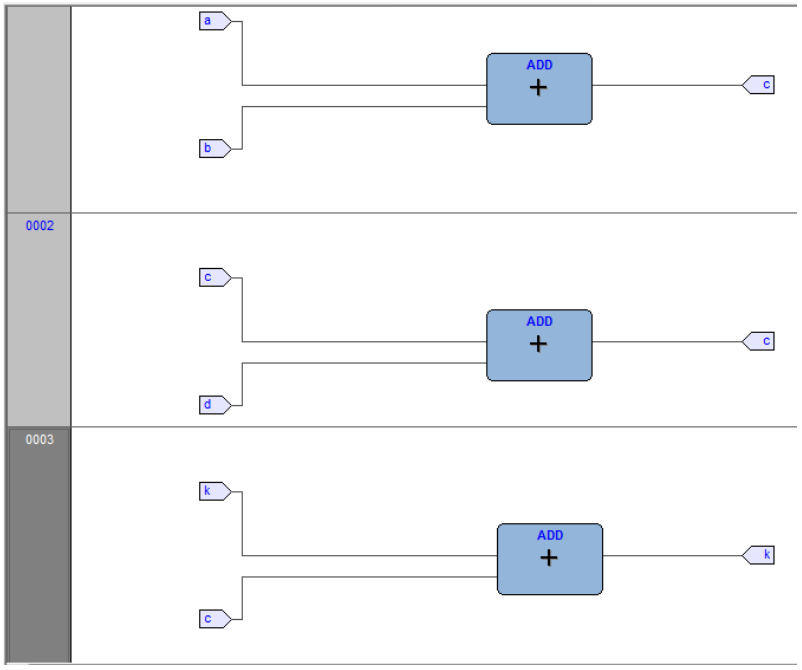


同样的过程适用于您需要监视的变量。

将第一个变量放入图形轨迹后，将自动弹出 *图形属性* 窗口，并允许用户设置采样和可视化属性。

9.6.2.3 在FBD语句打开图形触发窗口

假设您有一个FBD模块，其中还包含如下代码。



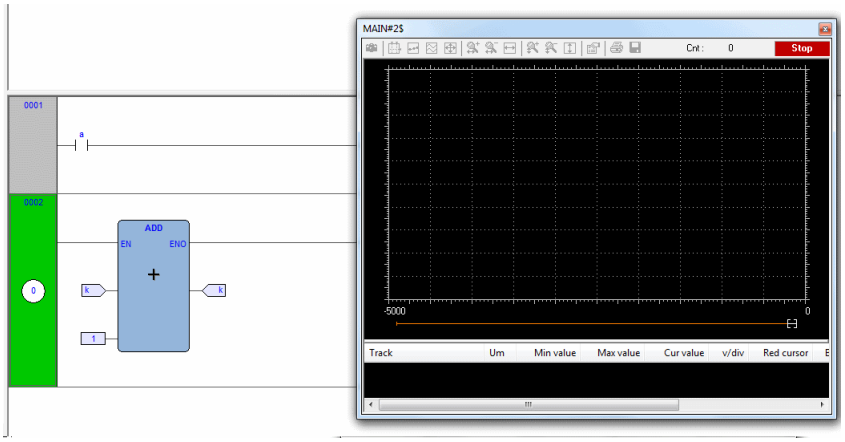
我们还假设您想在执行ST k语句之前知道c、d和k的值。

前提是您绝不能在表示变量的块中放置触发器，例如必须选择所选变量之前的第一个可用块。

在上述示例中，必须将光标移至网络3，然后单击ADD块。

现在点击 *调试>添加/删除图形触发*

这将导致所选块的颜色变为绿色，该块的中间内部出现带有触发器ID号的白色圆圈，并弹出相关的触发器窗口。



在预处理FBD源代码时，编译器将其转换为IL指令。网络3中的ADD指令扩展为：

```
LD k
ADD 1
ST k
```

将触发器添加到FBD块时，实际上是将触发器放置在其IL等效代码的第一条语句之前。

9.6.2.4 从FBD语句向图形触发窗口添加变量

为了观察变量的变化曲线，您需要将其添加到触发窗口。让我们假设您想在下面的图中检查FBD代码的变量k的值。

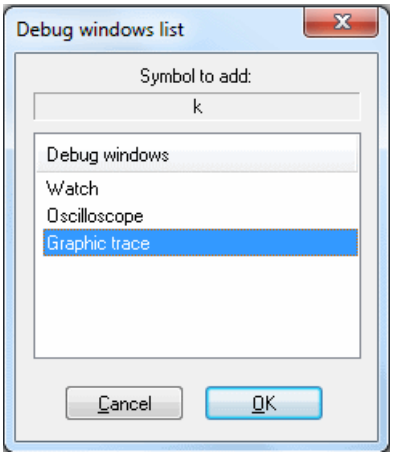
为此，请点击 *编辑>观看模式*

光标会变成如右图所示。

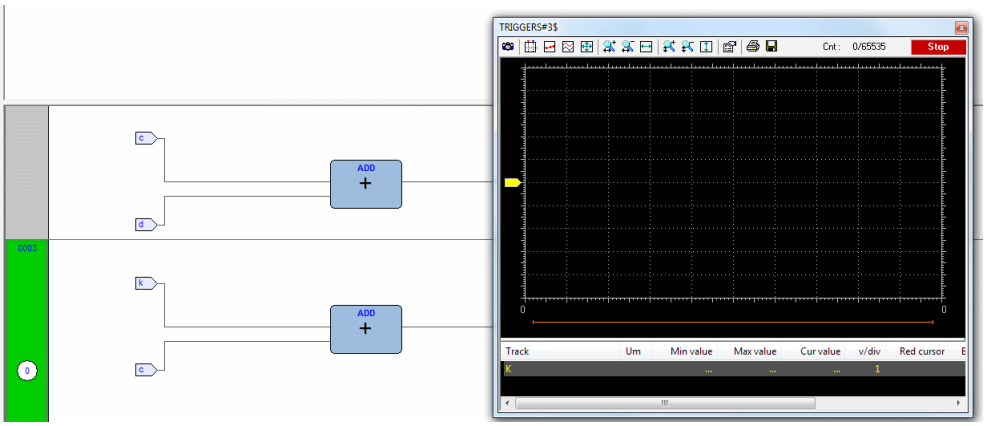
现在，您可以单击代表希望在图形触发窗口中显示的变量的块。

在我们正在考虑的示例中，单击按钮块。

此时将会出现一个对话框，列出所有当前的调试窗口实例，并询问您哪个是您刚刚单击的对象。



为了绘制变量k的曲线，请在 *调试窗口* 列中选择 *图形跟踪* ，然后按OK，变量的名称将显示在 *跟踪列表* 中。



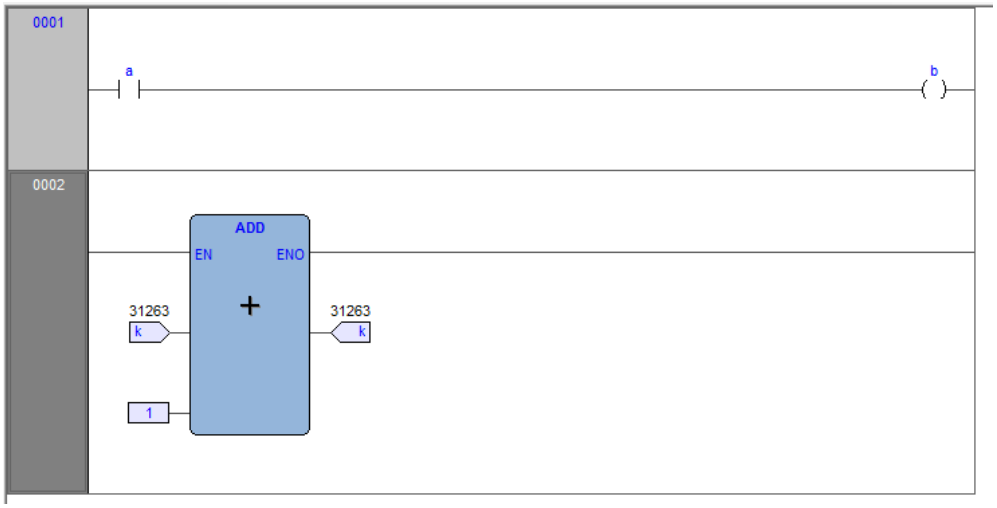
相同的过程适用于您要监视的所有变量。

将要观察的所有变量添加到 *图形* 监视窗口后，可以单击 *编辑>插入/移动模式* ，以使光标恢复其原始形状。

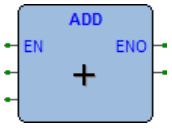
将首个变量放入图形轨迹后，将自动显示 *图形属性* 窗口，并允许用户设置采样和可视化属性。

9.6.2.5 在LD语句中打开触发窗口

假设您有一个LD模块，其中还包含如下代码。



您可以将触发器放置在如下所示的块上。



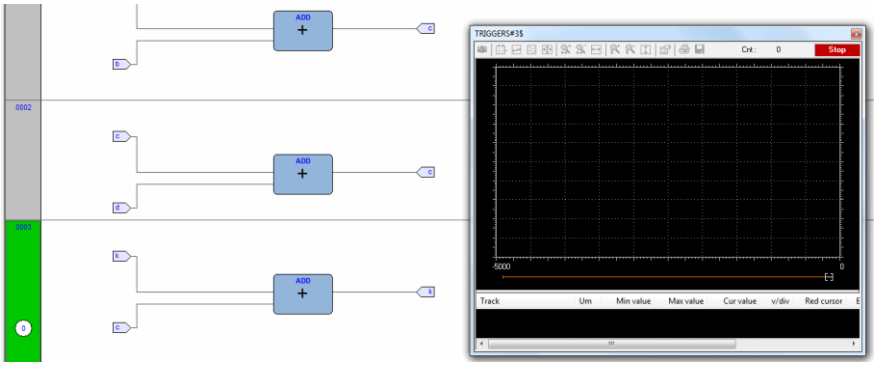
在这种情况下，同样的规则适用于FBD模块触点上插入一个触发器

或者线圈

在这种情况下，请按照说明进行操作。 假设您希望每次处理器到达网络编号1时都知道一些变量的值。

您必须单击组成网络号1的项目之一，然后点击 *调试>添加/删除图形触发*

这将导致包含网络号的灰色凸起按钮都将变为绿色，并且在按钮的中间会出现一个带有内部触发器编号的白色圆圈，同时会弹出相关的图形触发窗口。



请注意，与LogicLab支持的其他语言不同，LD不允许您在单个触点或线圈之前插入触发器，因为它只能选择整个网络。
因此，每次处理器到达所选网络的起点时，都会在 *图形* 触发窗口中对变量进行采样。

9.6.2.6 从LD语句向触发窗口添加变量

为了观察变量的变化曲线，您需要将其添加到图形触发窗口。假设您要检查下面图所示的LD代码中的变量b的值。

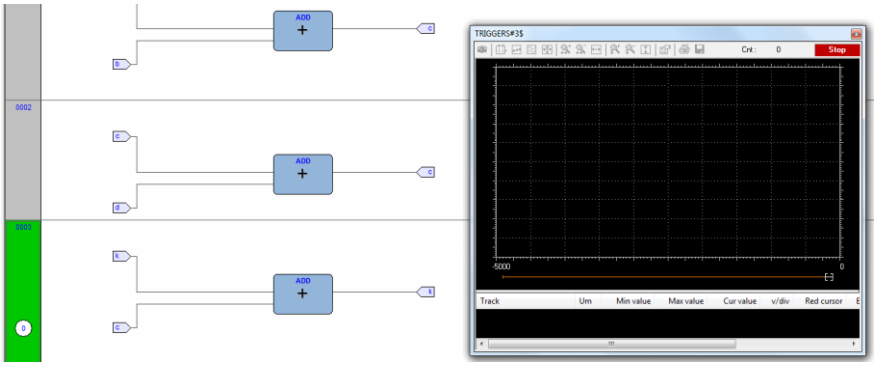
为此，请点击 *编辑>观看模式*

光标会变成如右图所示。

现在，可以单击您希望在图形触发窗口中显示的变量的项目。

LogicLab将会弹出一个对话框，列出所有当前调试窗口实例，并询问您哪个是您刚刚单击的对象。

为了绘制变量b的曲线，请在 *调试窗口* 列中选择 *图形跟踪* ，然后按OK，变量的名称将显示在 *跟踪列表* 中。



相同的过程适用于您要检查的所有变量。

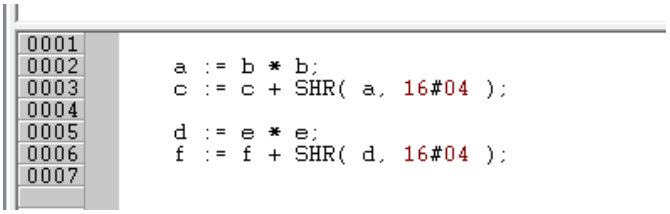
将要观察的所有变量添加到 *图形* 监视窗口后，可以单击 *编辑>插入/移动模式* ，

以使光标恢复其原始形状。

将首个变量放入图形轨迹后，将自动显示 *图形属性* 窗口，并允许用户设置采样和可视化属性。

9.6.2.7 在ST语句打开图形触发窗口

假设您有一个ST模块，其中还包含如下代码。



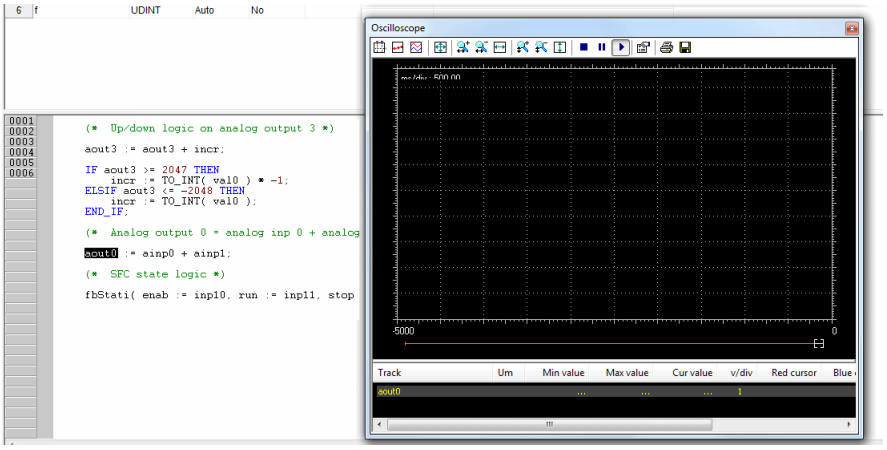
假设您想在执行下面的指令之前知道e、d和f的值

```
F: = F + SHR（d，16＃04）
```

为此，将光标移到第6行。

然后点击 *调试>添加/ 删除图形触发*

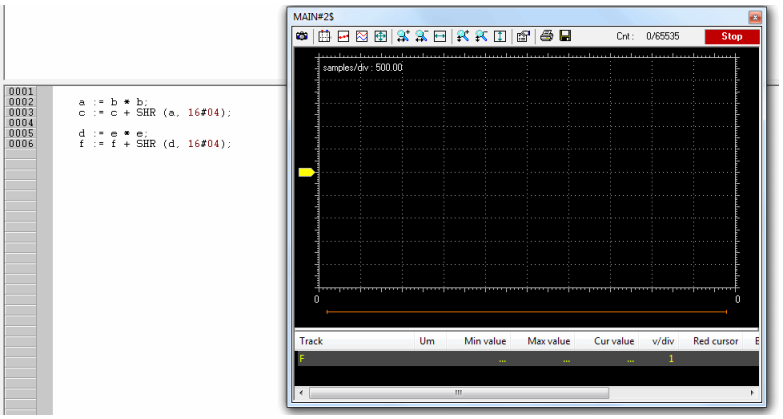
行号旁边都会出现一个绿色箭头，并弹出图形触发窗口。。



并非所有的ST指令支持触发器。例如，无法在包含终止符（例如END_IF、END_FOR、END_WHILE等）的行上放置触发器。

9.6.2.8 从ST语句向图形触发窗口添加变量

为了获得绘制的变量曲线，您需要将其添加到图形触发器窗口。为此，通过双击选择一个变量，然后将其拖动到 *变量* 窗口中，变量将会显示在 *跟踪列表* 中。



相同的过程适用于您要检查的所有变量。

将首个变量放入图形轨迹后，将自动显示 *图形属性* 窗口，并允许用户设置采样和可视化属性。

9.6.2.9 从触发窗口中删除变量

如果您想在图形触发窗口中删除一个变量，请单击一次它的名称来选择它，然后点击Del键。

9.6.2.10 图形触发窗口控件的使用

本段将介绍图形触发器窗口控件，它使您可以更好地使用调试工具，以获取有关范围内代码的更多信息。

启用控件

设置触发器时，将启用控制栏中的所有元素。您可以通过单击 *开始图形轨迹获取* 按钮来开始数据采集。

如果您定义了当前为假的用户条件，则即使您按了相应的按钮，也不会开始数据获取。

相反，一旦条件变为真，数据采集开始并继续，直到释放 *开始图形跟踪采集* 按钮，无论条件是否仍然为真。

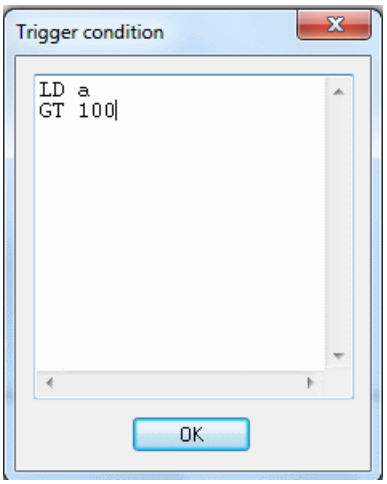
如果在获取所有必需的样本之前松开 *开始图形跟踪获取* 按钮，则获取过程将停止并且所有收集的数据都将丢失。

定义条件

该控件使用户可以设置何时开始采集的条件。 默认情况下，此条件设置为 真 ，并且在您按下 *启用/禁用* 采集按钮后立即开始采集。从那时起，每次触发时都会对 *调试* 窗口中的变量值进行采样。

为了指定条件，请打开 *选项* 对话框的 *条件* 选项卡，然后按相关按钮。

此时，将会弹出一个文本窗口，您可以在其中编写IL代码来设置条件。



完成条件代码的编写后，请单击 *确定* 按钮进行安装，或按 *取消* 按钮退出操作。 在按下 *开始图形轨迹获取* 按钮并且用户定义的条件为真之前，样品不会开始收集。此时，条件的简化表达式将会出现在控件中。

如要对其进行修改，请再次按上述按钮。

此时将出现文本窗口，其中包含您最初编写的文本，您现在可以对其进行编辑。

要完全删除用户定义的条件，请在文本窗口中删除整个IL代码，然后单击 *确定* 按钮。

执行条件代码后，累加器必须为布尔类型（TRUE或FALSE），否则会发生编译器错误。

条件代码中只能使用全局变量和拖入的变量。即如果尚未将其拖放到调试窗口中，则最初在其上插入触发器的模块本地的所有变量均不在范围内，且在条件窗口中不能声明任何新变量

设置坐标轴比例

- x轴

采集完成后，LogicLab会绘制调整x轴的拖动变量的曲线，以使所有数据适合 *图表* 窗口。 如果要应用其他比例，请打开 *图形属性* 对话框的 *常规* 选项卡，在水平比例编辑框中键入一个数字，然后单击 *应用* 进行确认。

- y轴

您可以通过 *图形属性* 对话框的 *轨迹* 列表选项卡更改每个变量的图的比例。否则，如果不需要精确指定比例，则可以使用 *放大* 和 *缩小* 控件。

9.6.2.11 关闭图形触发窗口并移除触发器

在与图形触发窗口调试会话结束时，您可以在下列选项中进行选择：

- 关闭图形触发窗口
- 删除触发器
- 删除所有的触发器

关闭图形触发窗口

如果已通过 *图形触发器窗口* 完成了对一组变量变化曲线的绘制，则可能要关闭 *调试* 窗口，而不必删除触发器。 如果单击右上角的按钮，则仅隐藏界面窗口，而窗口管理器和相关触发器均仍然保持工作。

实际上，如果以后要恢复先前隐藏的 *图形触发器窗口* ，请执行以下操作：

- 打开 *触发器列表* 窗口;
- 选择记录（类型为G）;
- 单击 *打开* 按钮。

此时会弹出一个界面窗口，并且触发计数器已更新，就好像尚未关闭一样。

删除触发器

如果选择此选项，则将窗口管理器及其触发器的代码完全删除。为此：

- 打开 *触发器列表* 窗口;
- 选择记录（类型为G）;
- 单击 *删除* 按钮。

或者，您可以将光标移动到该行（如果位于IL或ST模块中），或单击放置触发器的块（如果位于FBD或LD模块中）。请按 *调试* 工具栏中的 *设置/删除触发器* 按钮。

删除所有触发器

此外，您也可以同时删除所有现有的触发器，不管选择哪条记录，只要单击 *删除所有* 按钮即可。

10. LogicLab参考

10.1 菜单参考

在下表中，您可以看到所有LogicLab命令的列表。但是，由于LogicLab具有多文档界面（MDI），因此您可能会发现一些禁用的命令，甚至某些不可用的菜单，具体取决于当前您当前使用的文档类型。

10.1.1 文件菜单

命令	图标	快捷键	描述
新项目			创建一个新的LogicLab项目。
打开项目			打开现有LogicLab项目。
从目标导入项目			从目标设备中导入源工程。
查看项目(只读)			只读模式打开现有LogicLab项目。
保存项目			保存当前打开的项目。
项目另存为			将当前打开的项目另存为新的名称，位置和扩展名。
关闭项目			关闭打开的项目。
新的文本文件			打开一个空白的新的文本文件。
打开文件		*Ctrl+O*	打开现有文件，无论其扩展名是什么。 该文件显示在文本编辑器中。 无论如何，如果打开项目文件，则实际上是打开它所引用的LogicLab项目。
保存		*Ctrl+S*	保存当前活动窗口的代码。
关闭			关闭当前活动窗口的代码。
选项			打开LogicLab选项对话框。
打印		*Ctrl+P*	打印当前活动窗口的文件。
打印预览			创建当前活动窗口的文档预览，准备打印。
打印项目			打印组成项目的所有文件。
打印机设置			打开打印机设置对话框。
..最近..			列出最近打开的一组项目文件。
退出			关闭LogicLab。

10.1.2 编辑菜单

命令	图标	快捷键	描述
撤销		*Ctrl+Z*	撤销在文件中作出最后的动作。
重做		*Ctrl+Y*	恢复被撤消的最后一个动作。
剪切		*Ctrl+X*	从活动文档中删除所选项目，并将其存储在系统缓冲区中。
复制		*Ctrl+C*	将所选项目复制到系统缓冲区。
粘贴		*Ctrl+V*	在活动文档中粘贴系统缓冲区的内容。
删除		*Del*	删除选定的项目。
删除行		*Ctrl+E*	删除整段源代码行。
查找项目		*Ctrl + Shift + F*	打开在项目中查找对话框。
书签...			
添加/切换		*Ctrl+F2*	在某一行添加书签。 如果已经定义了书签，则将其删除。
下一个		*F2*	跳转到下一个定义的书签
上一个		*Shift+F2*	跳转到上一个定义的书签
移除所有			删除所有定义的书签
跳转到行		*Ctrl+G*	使您可以快速移至源代码编辑器中的目标行。
查找		*Ctrl+F*	要求您键入一个字符串，并从游标的当前位置在活动文档中搜索它的第一个实例。
查找下一个		*F3*	在查找命令的结果中逐个搜寻。
替换		*Ctrl+H*	允许您自动将一个字符串的一个或所有实例替换为另一字符串。
插入/移动模式			在用于插入或移动块的编辑模式之间切换。
连接模式			此模式下允许您绘制逻辑线以连接引脚。
观看模式			此模式下允许您将变量添加到任何调试工具。

10.1.3 查看菜单

命令	图标	快捷键	描述
工具栏			
主工具栏			显示或隐藏 *主要* 工具栏。
状态栏			显示或隐藏 *状态* 工具栏。
调试酒栏		Ctrl+B	显示或隐藏 *调试* 工具栏。
FBD酒栏		Ctrl+d	显示或隐藏 *FBD* 工具栏。
LD栏		Ctrl+A	显示或隐藏 *LD* 工具栏。
SFC栏		Ctrl+Q	显示或隐藏 *SFC* 工具栏。
项目栏		Ctrl+J	显示或隐藏 *项目* 工具栏。
网络		Ctrl+N	显示或隐藏 *网络* 工具栏。
文档栏		Ctrl+M	显示或隐藏 *文档 工具栏*。
工具窗口			
工作区		Ctrl+W	显示或隐藏 *工作区* 窗口（也称为 *项目* 窗口）。
库		Ctrl+L	显示或隐藏 *库* 窗口。
输出		CTRL+R	显示或隐藏 *输出* 窗口。
示波器		Ctrl+K	显示或隐藏 *示波器* 窗口。
监视窗口		Ctrl+T	显示或隐藏 *监视*窗口。
强制 I/O			显示或隐藏 *强制I/O* 工具栏。
PLC运行时状态			显示或隐藏 *PLC运行时状态* 窗口。
交叉参考窗口			尚未实现。
全屏		Ctrl+U	展开当前活动的文档窗口以填满整个屏幕。（Esc键退出该模式）。
网格			显示或隐藏在图形源代码编辑器的背景的虚线格。
显示对象的注释			显示或隐藏单个对象的注释，而不仅限于网络。 （仅适用于LD编辑器）。

10.1.4 项目菜单

命令	图标	快捷键	描述
新对象			
新程序			创建一个新的程序。弹出对话框以指定新程序性属性。
新功能块			创建一个新的功能块。弹出对话框以指定新功能块属性。
新功能			创建一个新的功能块。弹出对话框以指定新的功能性属性。
新变量			
自动创建			创建一个新的自动变量。弹出对话框以指定新的变量属性。
映射变量			创建一个新的映射变量。弹出对话框以指定新的变量属性。
常量			创建一个新的常数。弹出对话框以指定新的常量属性。
保持性变量			创建一个新的保持型变量。弹出对话框以指定新的变量属性。
复制对象			在当前 *工作区* 中选择复制对象。
粘贴对象			粘贴先前复制的对象。
重复对象			复制工作区中当前选定的对象，并要求您键入副本的名称。
删除对象			删除当前选择的对象。
查看PLC对象属性		*Alt+Enter*	显示当前所选对象的属性和描述。
对象浏览器			打开 *对象* 浏览器，使您可以在对象之间导航。
编译		*F7*	启动LogicLab编译器。
重新编译所有		*Ctrl + Alt + F7*	重新编译该项目。
生成可再发行源模块			生成一个RSM文件。
从库中导入对象			让您从库中导入LogicLab对象。
导出对象到库			使您可以将LogicLab对象导出到库。
库管理			打开 *库* 管理。
刷新所有的库			重新加载链接到项目的所有库文件。
宏			
新宏			创建一个新的宏。 提示对话框以指定新的宏属性。

命令	图标	快捷键	描述
复制宏			复制选定的宏，创建一个新的宏。
删除宏			删除选定的宏。
属性			显示选定的宏的属性。
选择目标...			让您选择一个新的目标项目。
刷新当前目标			使您可以将目标文件更新为相同版本的目标。
选项...			打开项目选项对话框。

10.1.5 在线菜单

命令	图标	快捷键	描述
建立通讯...			使您可以设置与目标的连接属性。
连接			LogicLab尝试建立与目标的连接。
下载代码		*F5*	LogicLab检查自上次编译以来是否已应用任何更改，如果是，则对项目进行编译，然后将源代码下载到目标。
下载选项			设置下载到目标的源代码的属性。
强制上传镜像文件			如果连接了目标设备，则可以上传镜像文件。
强制上传调试符号			如果已连接目标设备，则可以上传调试符号文件。
停止			停止PLC的运行。
冷启动			重新启动PLC执行，保持型和非保持型变量都将被重置。
暖启动			重新启动PLC执行，非保持型变量将被重置。
热启动			重新启动PLC执行，无需对任何变量进行重置。
重启目标设备			重启目标设备。
再次阅读所有日志			从目标重新加载所有远程日志。

10.1.6 调试菜单

命令	图标	快捷键	描述
仿真模式			打开/关闭集成的仿真环境。
启动/停止监视值			开始或停止（切换）对监视窗口中添加的符号的监视。
添加符号到监视窗口		*F8*	添加一个符号到监视窗口。
插入新对象到监视窗口		*Shift+F8*	插入一个新的对象到窗口。
添加符号到调试窗口		*F10*	添加一个符号到调试窗口。
插入新对象到调试窗口		*Shift+F10*	插入一个新的对象到调试窗口。
实时调试模式			如果正在运行调试模式，则启动或停止（切换）实时调试模式。
添加/删除文本触发器		*F9*	添加/删除文本触发器。
添加/删除图形触发器		*Shift+F9*	添加/删除图形触发器。
删除所有触发器		*Ctrl + Shift + F9*	删除所有活动的触发器。
触发器清单		*Ctrl+I*	列出所有活动的触发器。
运行			遇到断点后重新启动程序。
添加/删除断点		*F12*	添加或删除断点。
删除所有断点			删除所有活动断点。
断点列表			列出所有活动的断点。
改变当前执行个体			更改当前功能块实例（实时调试模式）。

10.1.7 FBD图标菜单

命令	图标	快捷键	描述
网络			
新的			
顶部			在当前代码的顶部添加一个空白网络。
底部			在当前代码的底部添加一个空白网络。
目标之前			在当前代码所选网络之前添加空白网络。
目标之后			在当前代码所选网络之后添加空白网络。
标签			为选定的网络分配标签，以便可以将其指示为跳转指令的目标。
对象			
新			
功能			打开该对象浏览器，选择一个 *功能* 插入到到当前代码中。
功能块		*Shift+B*	打开该对象浏览器，选择一个 *功能块* 插入到到当前代码中。
变量		*Shift+V*	打开该对象浏览器，选择一个 *变量* 插入到到当前代码中。
常量		*Shift+K*	打开该对象浏览器，选择一个 *常量* 插入到到当前代码中
返回		*Shift+R*	添加一个返回语句到选定的网络中。
跳转到标签		*Shift+J*	添加一个跳转语句到选定的网络中。
运算符			打开该对象浏览器，选择一个 *运算符* 插入到到当前代码中。
注释		*Shift+M*	在所选的网络中添加注释。
实例名称			打开该对象浏览器，选择一个 *实例* 插入到到当前代码中。
打开编辑器			打开用于创建所选对象的编辑器，并显示相关的源代码： - 如果对象是程序，函数或功能块，则此命令将打开其源代码； - 如果对象是变量或参数，则此命令将打开相应的变量编辑器； - 如果对象是标准功能或运算符，则此命令没有功能。
自动连接			切换自动连接模式，以便在两个块足够近时自动连接它们。
删除无效的连接		*Ctrl+M*	删除所有无效连接，在当前代码中用红线表示。

命令	图标	快捷键	描述
增加引脚		*Ctrl+'+'*	增加附加的引脚到所选模块，以增加标准引脚。
减少引脚		*Ctrl+'-'*	删除由 增加引脚 命令添加的引脚。
使能EN / ENO引脚			将启用输入/启用输出引脚添加到所选模块。 仅当enable 输入信号为true时，才会执行实现所选块的代码。使能输出信号只是重复使能输入的值，允许您级联使用。
对象属性			显示所选块的一些属性。

10.1.8 LD图标菜单

命令	图标	快捷键	描述
网络			
新			
顶部			在当前代码的顶部添加一个空白网络。
底部			在当前代码的底部添加一个空白网络。
目标之前			在当前代码所选网络之前添加空白网络。
目标之后			在当前代码所选网络之后添加空白网络。
标签			为选定的网络分配标签，以便可以将其指示为跳转指令的目标。
对象			
新			
并联前		*Shift+P*	在所选网络之前并联一个节点到所选网络中。
并联后			在所选网络之后并联一个节点到所选网络中。
串联前			在所选网络之前串联一个节点到所选网络中。
串联后		*Shift+C*	在所选网络之后串联一个节点到所选网络中。
线圈		*Shift+O*	增加了一个线圈到所选择的网络。
块		*Shift+B*	打开该对象浏览器，选择一个 块 插入到到当前代码中。
常量		*Shift+K*	打开该对象浏览器，选择一个 常量 插入到到当前代码中。
返回		*Shift+R*	添加一个返回语句到选定的网络中。
跳转		*Shift+J*	添加一个跳转语句到选定的网络中。
变量		*Shift+V*	打开该对象浏览器，选择一个 变量 插入到到当前代码中。
表达式		*Shift+E*	增加了一个 表达式 到所选择的网络。
新建分支			在当前位置后创建新的分支。
注释		*Shift+M*	在所选的网络中添加注释。
实例名称			打开该对象浏览器，选择一个 *实例* 插入到到当前代码中。

命令	图标	快捷键	描述
打开编辑器			打开用于创建所选对象的编辑器，并显示相关的源代码： - 如果对象是程序，函数或功能块，则此命令将打开其源代码； - 如果对象是变量或参数，则此命令将打开相应的变量编辑器； - 如果对象是标准功能或运算符，则此命令没有功能。
触点打开		*O*	将所选触点改变为打开的触点。
触点关闭		*C*	将所选触点改变为关闭的触点。
触点取正		*P*	将所选触点改变为正的触点。
触点取反		*N*	将所选触点改变为取反的触点。
置位线圈		*S*	将所选线圈改变为置位线圈。
复位线圈		*R*	将所选线圈改变为复位线圈。
增加引脚		*Ctrl+'+'*	增加附加的引脚到所选模块，以增加标准引脚。
减少引脚		*Ctrl+'-'*	删除由 增加引脚 命令添加的引脚。
使能EN / ENO引脚			将启用输入/启用输出引脚添加到所选模块。 仅当enable 输入信号为true时，才会执行实现所选块的代码。使能输出信号只是重复使能输入的值，允许您级联使用。
设置输出线			将选定的引脚设置为模块的输出线。
对象属性			显示所选块的一些属性。

10.1.9 SFC图标菜单

命令	图标	快捷键	描述
对象			
新			
步			将新的 *步* 添加到选定的网络。
转换			将新的 *转换* 添加到选定的网络。
跳转			将新的 *跳转* 添加到选定的网络。
修改			
添加一个发散转换分支引脚			向选定的转换添加一个发散分支引脚。
移除一个发散转换分支引脚			删除选定的转换的发散分支引脚。
添加一个合并转换分支引脚			将合并的引脚添加到选定的转换。
移除一个合并转换分支引脚			移除会选定的转换上合并的引脚。
添加一个引脚到同步发散转换分支			将同步发散的引脚添加到选定的转换。
移除一个引脚到同步发散转换分支			移除选定的转换上同步发散的引脚。
添加一个引脚到同步合并转换分支			将同步合并的引脚添加到选定的转换。
移除一个引脚到同步合并转换分支			移除选定的转换上同步合并的引脚。
在最右端号前添加空格			在最右端号前添加空格。
删除最右边端号前的空格			删除最右边端号前的空格。
代码对象			
新动作			在当前代码中添加的一个动作。
新的转换代码			在当前代码中添加的一个转换。
自动连接			切换自动连接模式，以便在两个块足够接近时自动连接。
删除无效的连接		Ctrl+M	删除所有无效连接，在当前代码中用红线表示。

10.1.10 变量菜单

命令	图标	快捷键	描述
加			
自动			创建一个新的自动变量。弹出对话框以指定新的变量属性。
映射变量		*Ctrl + Shift + M*	创建一个新的映射变量。弹出对话框以指定新的变量属性。
常量			创建一个新的常量。弹出对话框以指定新的常量属性。
保持型变量			创建一个新的保持型变量。弹出对话框以指定新的变量属性。
插入		*Ctrl + Shift + ins*	添加一个新行到网格在当前编辑器中。
删除		*Del*	删除当前编辑器中的所选行中的变量。
创建多个变量			可以让你一次创建多个变量。
变量组			打开一个对话框，它可以让你创建或删除变量组。

10.1.11 窗口菜单

命令	图标	快捷键	描述
层叠			层叠替换所有打开的文档，除了标题使其完全层叠打开。
平铺			根据当前打开的文档数，可编程逻辑控制器编辑器区域被拆分为具有相同尺寸的框架。每一块自动分配给每一个文档。
排列图标			在 *PLC编辑器* 区域的左下角替换最小化文档的图标。
关闭所有文件			关闭所有打开的文档。

10.1.12 帮助菜单

命令	图标	快捷键	描述
索引			列出所有帮助关键字并打开相关主题。
上下文		*F1*	上下文相关帮助。打开与当前活动窗口相关的主题。
关于...			版本信息。

10.2 工具栏参考

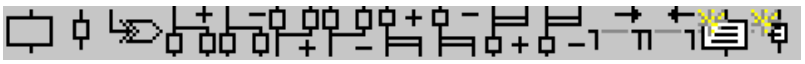
在下表中，您可以看到LogicLab所有工具栏的列表。无论当前处于打开的项目文件如何，构成每个工具栏的按钮是始终相同。但是，此时点击一些和项目文件有逻辑关系的功能可能不会产生任何效果。

10.2.1 主工具栏

10.2.2 FBD工具栏

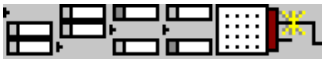
10.2.3 LD工具栏

10.2.4 SFC工具栏



10.2.5 项目工具条

10.2.6 网络工具栏



10.2.7 调试工具栏

11. 语言参考

所有LogicLab语言均符合IEC 61131-3标准。

- 公共元素
- 指令表（IL）
- 功能块图（FBD）
- 梯形图（LD）
- 结构化文本（ST）
- 顺序功能图（SFC）

此外，LogicLab实现了一些扩展：

- 指针
- 宏

11.1 公共要素

文本和图形元素是IEC 61131-3标准规定的所有可编程控制器编程语言所共有的。

注意： 大多数公共元素（变量、结构化元素、功能块定义等）的定义和编辑由LogicLab通过指定的编辑器、窗体和表格进行管理。
LogicLab不允许直接编辑与上述公共元素相关的源代码。
以下段落是一种语言规范。要正确管理公共元素，请参阅LogicLab用户指南。

11.1.1 基本元素

11.1.1.1 字符集

图形语言的文本文档和文本元素是使用标准的ascii字符集编写的。

11.1.1.2 注释

用户注释的开头和结尾分别由特殊字符组合 “（*” 和 “*）” 分隔。在程序中的任何地方都允许注释，并且它们在本标准中所定义的任何语言中都没有语法或语义意义。

使用嵌套的注释，例如，（*（*嵌套*）*），被处理为错误。

11.1.2 基本数据类型

LogicLab提供了许多基本数据类型(即预先定义的数据类型)，均符合IEC 61131-3标准。

基本数据类型、每个数据类型的关键字、每个数据元素的位数和每个基本数据类型的值范围在下表中进行了描述。

关键词	数据类型	位	范围
BOOL	布尔	见说明	0 - 1
SINT	短整型	8	-128 - 127
USINT	无符号短整型	8	0 - 255
INT	整数	16	-32768 - 32767

关键词	数据类型	位	范围
UINT	无符号整数	16	0 - 65536
DINT	长整数	32	-2^{31} - $2^{31}-1$
UDINT	无符号长整型	32	0 - 2^{32}
BYTE	长度为8的比特串	8	-
WORD	长度为16的比特串	16	-
DWORD	长度为32的比特串	32	-
REAL	实数	32	-3.40E+38 - +3.40E+38
STRING	字符串	-	-

注意： BOOL数据类型的实际长度取决于目标设备的处理器，例如对于具有对位寻址的设备，它的长度为1位。

11.1.3 派生数据类型

导出的数据类型可以使用 TYPE ... END_TYPE 结构进行声明。除了基本数据类型之外，它们还可以用于变量声明。
单元素变量和多元素变量的元素都被认为是派生数据类型，可以在任何地方使用父类型的变量。

11.1.3.1 类型定义

Typedefs的目的是为现有类型分配一个名称。除了名称之外，Typedefs和它的父类型之间没有任何区别。
Typedefs可以使用以下语法声明：

```
TYPE
<enumerated data type name> : <parent type name>;
END_TYPE
```

例如，将名称LONGWORD映射到IEC61131-3标准类型DWORD：

```
TYPE
longword : DWORD;
END_TYPE
```

11.1.3.2 枚举数据类型

枚举数据类型声明指定该类型的任何数据元素的值只能是关联的标识符列表中给定的值之一。 枚举列表定义了一组有序的枚举值，从列表的第一个标识符开始，到最后一个标识符为止。

枚举数据类型可以使用以下语法声明：

```
TYPE
<enumerated data type name> : ( <enumeration list> );
END_TYPE
```

例如，考虑以下两个枚举数据类型的声明。请注意，如果在枚举列表中没有为标识符提供显式值，则其值等于分配给前一个标识符的值加一。

```
TYPE
enum1: (
      val1, (* the value of val1 is 0 *)
      val2, (* the value of val2 is 1 *)
      val3 (* the value of val3 is 2 *)
);
enum2: (
      k := -11,
      i := 0,
      j, (* the value of j is ( i + 1 ) = 1 *)
      l := 5
);
END_TYPE
```

不同的枚举数据类型可以对枚举值使用相同的标识符。但是为了在特定上下文中使用时被唯一标识，枚举文字可以由包含其关联数据类型名称和 # 符号的前缀来指定。

11.1.3.3 数组

数组声明指定将该类型的任何数据元素的值限制在指定的上限和下限之间并包括它们。

数组可以使用以下语法声明:

```
TYPE
<subrange name> : <parent type name> ( <lower limit>..<upper limit>
);
END_TYPE
```

请参考下面的例子:

```
TYPE
int_0_to_100 : INT (0..100);
END_TYPE
```

11.1.3.4 结构体

结构声明指定该类型的数据元素应包含指定类型的子元素，可以通过指定的名称访问这些子元素。

结构体可以用下面的语法声明:

```
TYPE
<structured type name> : STRUCT
      <declaration of structurestructure elements>
END_STRUCT;
END_TYPE
```

请参考下面的例子:

```
TYPE
structure1: STRUCT
      elem1 :  USINT;
      elem2 :  USINT;
      elem3: INT;
```

```
        elem3: REAL;
    END_STRUCT;
    END_TYPE
```

11.1.4 常量

11.1.4.1 数字常量

在各种可编程控制器中，数据可以由数字常量表示。
有两类数字常量:整数常量和浮点常量。数字常量被定义为十进制数字或基数。十进制数字用传统的十进制记数法表示。

真正的文字是由小数点来区分的。指数表示10的整数次幂，需要将前面的数字乘以10才能得到所表示的值。十进制文字及其指数可以包含前面的符号(+或-)。

整数字面值也可以用基数2、8或16表示。以16为基数的基数采用十进制记数法，使用由字母A到F组成的扩展数字集，分别具有十进制10到15的传统意义。基于数字不包含任何前导符号(+或-)。

布尔数据由关键字FALSE或TRUE表示。数值文字特征和例子如下表所示。

特征描述	例子
整型常量	-12 0 123 986
浮点常量	-12.0 0.0 0.4560
带指数的浮点常量	-1.34E-12 or -1.34e-12 1.0E + 6 1.0E or + 6 1.234E6 or 1.234E6
2进制常量	2＃11111111（十进制为256） 2＃11100000（十进制为240）
8进制常量	8＃377（十进制为256） 8＃340（十进制为240）
16进制常量	16个＃FF or 16个＃FF（十进制为256）16＃E0 or 16＃E0（十进制为240）
布尔类型FALSE和TRUE	FALSE TRUE

11.1.4.2 字符串常量

字符串文字是由单个或多个字符组成的序列，以单引号字符(')作为前缀和结尾。
美元符号($)后面两个十六进制双位数的三进制组合应解释为八位字符代码的十六进制表示。

例	说明
''	空字符串（长度为零）
'A'	长度为1的字符串，包含单个字符A
' '	长度为1的字符串，包含空格字符
'$''	长度为1的字符串，包含单引号字符

例	说明
'" '	长度为1的字符串，包含双引号字符
'RL'	长度为2的字符串，包含CR和LF字符
'$0A'	长度为1的字符串，包含LF字符

如下面的表中，当它们出现在字符串美元符号开始的两个字符的组合应解释。

组合	打印时解释
$$	美元符号
$'	单引号
$L或$l	换行
$N或$N	新行
$P或$P	换页
$R或$R	回车
$T或$T	选项

11.1.5 变量

11.1.5.1 前言

变量提供了用户可通过改变其内容来改变的数据对象的手段，例如可编程控制器的输入、输出或存储器相关的数据。 变量必须声明为基本类型之一， 变量可以用符号表示，也可以用直接表示数据元素与可编程控制器的输入、输出或存储器结构中的物理或逻辑位置的关联的方式表示。

每个程序组织单位（POU，即每个程序，功能或功能块）在其开始处至少包含一个声明部分，该声明部分由一个或多个结构元素组成，这些结构元素指定组织单元中使用的变量的类型（如有必要，还包括物理或逻辑位置）。此声明部分的文本形式为关键字区段中定义的关键字VAR，VAR_INPUT或VAR_OUTPUT之一，在VAR的情况下，其后为零或一次出现限定词RETAIN、NON_RETAIN或限定词CONSTANT，并且在如果是VAR_INPUT或VAR_OUTPUT，则为零或一次出现限定符RETAIN或NON_RETAIN，然后是一个或多个用分号分隔并以关键字END_VAR终止声明。声明还可以指定声明变量的初始化，当可编程控制器支持用户声明变量的初始值时。

11.1.5.2 结构单元

变量的声明必须在以下程序结构元素中执行:

```
  KEYWORD [RETAIN] [CONSTANT]
    Declaration 1
    Declaration 2
  ...
    Declaration N
END_VAR
```

11.1.5.3 关键词和范围

关键词	变量的使用
VAR	组织单元内部使用。
VAR_INPUT	外部输入。
VAR_OUTPUT	通过组织单位提供给外部使用。
VAR_IN_OUT	由于外部输入，可在组织单元内部进行修改。
VAR_EXTERNAL	经由VAR_GLOBAL配置提供，可在组织单元内进行修改。
VAR_GLOBAL	全局变量声明。

结构单元中包含声明的范围（有效性范围）仅是此程序组织单元（POU）的本地范围。也就是说，声明的变量如果希望被其他的程序组织单元访问，只有通过显式的参数传递或者使用这些单元的输入或输出声明的变量才可被访问。但是对于已经被定义为全局变量，程序可以在任何情况下访问此类变量，也可以通过功能块的VAR_EXTERNAL声明访问此类变量。在VAR_EXTERNAL中声明的变量类型必须与在VAR_GLOBAL块中声明的类型一致。

要在不使用任何关键字的情况下访问所有类型的POU的变量，必须在项目选项的代码生成选项卡中启用此选项（请参见第4.6.2节）。

如果出现以下情况，则出现错误：

- 任何程序组织单位试图修改已用CONSTANT限定符声明的变量的值；
- 在包含元素内包含的任何元素的VAR_EXTERNAL声明（不包含CONSTANT限定词）中，使用在配置元素或程序组织单元（“包含元素”）中声明为VAR_GLOBAL CONSTANT的变量。

11.1.5.4 限定符

预选赛	描述
CONST	属性CONST表示结构元素中的变量是常量，即它们有一个常量值，一旦PLC被引导，这个常量值就不能被修改。
RETAIN	属性RETAIN表示结构元素中的变量是保留的，即即使在重置或关闭目标设备之后，它们仍然保持其值。

11.1.5.5 单元素变量和数组

单元素变量表示基本类型之一或派生数据类型之一的单个数据元素。

数组是相同数据类型的数据元素的集合；为了访问数组的单个元素，必须使用方括号内的下标（或索引）。下标可以是整数或单元素变量。

为了轻松表示数据矩阵，数组可以是多维的；在这种情况下，需要一个复合下标，每个维度一个索引，以逗号分隔。 数组定义中允许的最大维数为3。

11.1.5.5 声明语法

变量必须在结构元素中声明，使用以下语法：

```
VarName1 : Typename1 [ := InitialVal1 ];
VarName2 AT Location2 : Typename2 [ := InitialVal2 ];
VarName3 : ARRAY [ 0..N ] OF Typename3;
```

其中：

关键词	描述
VarNameX	变量标识符，由长度为1或1以上的字母数字字符组成的字符串。它用于变量的符号表示。
TypenameX	变量的数据类型，从基本数据类型中选择。
InitialValX	复位目标后，变量将重置为初始值。
LocationX	详细说明见下一个章节。
N	最后一个元素的索引，长度为N+1的数组

11.1.5.6 位置

变量可以用符号表示，即通过它们的标识符来访问，也可以用一种直接表示数据元素与可编程控制器的输入、输出或内存结构中的物理或逻辑位置的关联的方式来改变。

单一元素变量的直接表示由百分号“%”、位置前缀和大小前缀以及一个或两个由句点(.)分隔的无符号整数连接而成的特殊符号提供。

```
%location.size.index.index
```

1) 位置

位置前缀可以是下列之一：

位置前缀	描述
I	输入区
Q	输出区
M	存储区

2) 尺寸

大小前缀可以是下列之一：

大小前缀	描述
X	Bit（1位）
B	BYTE（8 bits）
W	WORD（16 bits）

大小前缀	描述
D	DWORD（32 bits）

3) index.index

这个无符号整数序列由点分隔，指定变量在位置前缀指定的区域中的实际位置。

例：

地址	描述
%MW4.6	从存储数据块4的第7个元素的第一个字节开始的一个WORD类型的数据。
%IX0.4	输入区数据块0的第5个元素的第一个字节的第一位。

注意，绝对位置取决于数据块元素的大小，而不取决于大小前缀。实际上，%MW4.6和%MD4.6从内存中的同一字节开始，但是前者指向的区域比后者短16位。

高级用法：如果索引仅包含一个整数（无点），则它将丢失对数据块的任何引用，并且直接指向内存中具有索引值作为其绝对地址的字节。

地址	描述
%MW4.6	从内存中数据块4的第7个元素的第一个字节开始的一个WORD类型的数据。
%MW4	从内存区第4字节开始的一个WORD类型的数据。

例：

```
VAR [RETAIN] [CONSTANT]
XQuote : DINT; Enabling : BOOL := FALSE;
TorqueCurrent AT %MW4.32 : INT;
Counters : ARRAY [ 0 .. 9 ] OF UINT;
Limits: ARRAY [0..3, 0..9]
END_VAR
```

- 变量 XQuote 的长度为32位，由LogicLab编译器自动分配。
- 目标重置后，变量启用被初始化为FALSE。
- 在目标设备的存储区中分配了变量TorqueCurrent，它从数据块4的第33个元素的第一个字节开始占用16位。
- 变量 Counters 是10个无符号整数类型的独立变量的数组。

11.1.5.7 LOGICLAB变量声明

无论您使用哪种PLC语言，LogicLab都提供了本地变量编辑器，全局变量编辑器和参数编辑器，您可以忽略上面的语法而使用一种表格化的操作接口来声明各种变量。

11.1.6 程序组织单元

程序组织单元包括功能、功能块和程序。程序组织单元可以由制造商交付，也可以由用户通过标准定义的方式由用户编程。

也就是说，程序组织单元不能被另一个相同类型的程序组织单元所调用。

11.1.6.1 功能

对于可编程控制器编程语言，将功能定义为一种程序组织单位（POU），当执行该程序时，它会产生一个数据元素，该数据元素被视为功能。

功能不包含内部状态信息，即调用具有相同参数的函数（输入变量VAR_INPUT和输入输出变量VAR_IN_OUT）始终产生相同的值（输出变量VAR_OUTPUT，输入输出变量VAR_IN_OUT和函数结果）。

声明语法

功能的声明如下：

```
FUNCTION FunctionName : RetDataType
VAR_INPUT
     declaration of input variables (see the relevant section)
END_VAR
VAR
     declaration of local variables (see the relevant section)
END_VAR
      Function body
END_FUNCTION
```

关键词	描述
FunctionName	所声明的函数的名称。
RetDataType	该功能返回值的数据类型。
FunctionBody	指定对输入变量执行的操作，以便根据功能的语义将值赋给与功能同名的变量，该变量表示函数的结果。它可以在LogicLab支持的任何语言中使用。

在LogicLab声明功能

无论您使用哪种PLC语言，您可以忽略上面的语法, LogicLab提供了一个友好的界面方便您使用。

11.1.6.2 功能块

对于可编程控制器编程语言，功能块是另一种程序组织单位（POU），执行时会产生一个或多个值。可以创建功能块的多个命名实例（副本）。每个实例都有一个关联的标识符（实例名称），以及一个包含其输入，输出和内部变量的数据结构。

从功能块的一次执行到下一次执行，该数据结构的所有输出变量值和必要的内部变量都将保留。因此，具有相同参数（输入变量）的功能块的调用并不总是产生相同的输出值。

在功能块的实例外部只能访问输入和输出变量，即，功能块的内部变量对功能块的用户不可见。为了执行其操作，功能块需要由另一个POU调用。

调用取决于调用功能块的模块的特定语言。

功能块实例的范围对于实例化该功能块的程序组织单元而言是本地的。

声明语法

功能块的声明如下：

```
FUNCTION_BLOCK FunctionBlockName
  VAR_INPUT
      declaration of input variables (see the relevant section)
  END_VAR
  VAR_OUTPUT
      declaration of output variables
  END_VAR
  VAR_EXTERNAL
      declaration of external variables
  END_VAR
  VAR
      declaration of local variables
  END_VAR
      Function block body
END_FUNCTION_BLOCK
```

关键词	描述
FunctionBlockName	要声明的功能块的名称（注意：模板的名称，而不是其实例的名称）。
VAR_EXTERNAL .. END_VAR	只有在VAR_EXTERNAL结构元素中列出了全局变量，功能块才能访问全局变量。可以通过FB修改通过VAR_EXTERNAL构造传递到FB中的变量。
Function block body	规定要对输入变量执行的操作，根据功能块的语义和内部变量的值将值分配给输出变量。可以在LogicLab支持的任何语言中使用。

在LogicLab声明功能

无论您使用哪种PLC语言，您可以忽略上面的语法, LogicLab提供了一个友好的界面方便您使用功能块。

11.1.6.3 程序

程序在IEC 61131-1中定义为“所有编程语言元素和构造的逻辑组合，是通过可编程控制器系统控制设备或过程中信号与数据处理所必需的”。

声明语法

程序的声明如下：

```
PROGRAM < program name>
     Declaration of variables (see the relevant section)
     Program body
END_PROGRAM
```

关键词	描述
Program Name	要声明的程序的名称。
Program body	指定要执行的操作以及处理数据的程序逻辑，可以在LogicLab支持的任何语言中使用。

在LogicLab中编写程序

无论您使用哪种PLC语言，您可以忽略上面的语法, LogicLab提供了一个友好的界面方便您编写程序。

11.1.7 IEC 61131-3标准功能

本段是LogicLab中可用的所有IEC 61131-3标准功能的参考，以及其他一些功能（可以被视为LogicLab对标准的扩展）。

这些功能是整套编程语言所共有的，因此可以在任何可编程组织单位（POU）中使用。

在本段中指定为可扩展（外部的）的功能允许具有可变数量的输入。

类型转换功能

根据IEC 61131-3标准，类型转换函数的格式应为* _TO _ **，其中“ *”为输入变量的类型，而“ **”为输出变量的类型（例如INT_TO_REAL）。

LogicLab提供了一组更为方便的重载类型转换函数，从而减轻了开发人员指定输入变量类型的麻烦。

TO_BOOL	
描述	转换为BOOL类型（布尔值）
操作数	1
输入数据类型	任何数值类型
输出数据类型	BOOL
示例	out := TO_BOOL(0); (* out = FALSE *) out := TO_BOOL(1); (* out = TRUE *) out := TO_BOOL(1000); (* out = TRUE *)

TO_SINT	
描述	转换为SINT类型（8位有符号整数）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	SINT
示例	out := TO_SINT(-1); (* out = -1 *) out := TO_SINT(16#100); (* out = 0 *)

TO_USINT	
描述	转换为USINT类型（8位无符号整数）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	USINT
示例	out := TO_USINT(-1); (* out = 255 *) out := TO_USINT(16#100); (* out = 0 *)

TO_INT	
描述	转化为INT类型（16位有符号整数）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	INT
示例	out := TO_INT(-1000.0); (* out = -1000 *) out := TO_INT(16#8000); (* out = -32768 *)

TO_UINT	
描述	转换为UINT类型（16位无符号整数）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	UINT
示例	out := TO_UINT(1000.0); (* out = 1000 *) out := TO_UINT(16#8000); (* out = 32768 *)

TO_DINT	
描述	转换为DINT类型（32位带符号整数）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	DINT
示例	out := TO_DINT(10.0); (* out = 10 *) out := TO_DINT(16#FFFFFFFF); (* out = -1 *)

TO_UDINT	
描述	转换为UDINT类型（32位无符号整数）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	UDINT
示例	out := TO_UDINT(10.0); (* out = 10 *) out := TO_UDINT(16#FFFFFFFF); (* out = 4294967295 *)

TO_BYTE	
描述	转换为BYTE类型（8比特串）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	BYTE
示例	out := TO_BYTE(-1); (* out = 16#FF *) out := TO_BYTE(16#100); (* out = 16#00 *)

TO_WORD	
描述	转换为WORD类型（16比特串）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	WORD
示例	out := TO_WORD(1000.0); (* out = 16#03E8 *) out := TO_WORD(-32768); (* out = 16#8000 *)

TO_DWORD	
描述	转换到DWORD类型（32比特串）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	DWORD
示例	out := TO_DWORD(10.0); (* out = 16#0000000A *) out := TO_DWORD(-1); (* out = 16#FFFFFFFF *)

TO_REAL	
描述	转换到REAL类型（32位浮点）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	REAL
示例	out := TO_REAL(-1000); (* out = -1000.0 *) out := TO_REAL(16#8000); (* out = -32768.0 *)

TO_LREAL	
描述	转换到LREAL类型（64位浮点）
操作数	1
输入数据类型	任何数值类型或STRING类型
输出数据类型	LREAL
示例	out := TO_LREAL(-1000); (* out = -1000.0 *) out := TO_LREAL(16#8000); (* out = -32768.0 *)

数值函数

以下功能的可用性取决于目标设备。 有关详细信息，请咨询您的硬件供应商。

ABS	
描述	绝对值，计算的输入的绝对值
操作数	1
输入数据类型	任何数值类型
输出数据类型	与输入类型相同
示例	OUT := ABS(-5);(* OUT = 5 *) OUT := ABS(-1.618);(* OUT = 1.618 *) OUT := ABS(3.141592);(* OUT = 3.141592 *)

SQRT	
描述	平方根，计算的输入的平方根
操作数	1
输入数据类型	LREAL（如果有），否则为REAL
输出数据类型	LREAL（如果有），否则为REAL
示例	OUT := SQRT(4.0); (* OUT = 2.0 *)

LN	
描述	自然对数，计算具有的输入以e为底的对数
操作数	1
输入数据类型	LREAL（如果有），否则为REAL
输出数据类型	LREAL（如果有），否则为REAL
示例	OUT := LN(2.718281); (* OUT = 1.0 *)

LOG	
描述	常用对数，计算具有的输入以10为底的对数
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT := LOG(100.0);(* OUT = 2.0 *)

EXP	
描述	自然指数，计算的输入的指数函数
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = EXP（1.0）；（* OUT≈2.718281 *）

罪	
描述	正弦，计算的输入正弦函数，结果以弧度表示
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = SIN（0.0）；（* OUT = 0.0 *） OUT: = SIN（2.5 * 3.141592）；（* OUT≈1.0 *）

COS	
描述	余弦，计算的输入余弦函数，结果以弧度表示
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = COS（0.0）；（* OUT = 1.0 *） OUT: = COS（-3.141592）；（* OUT≈-1.0 *）

TAN	
描述	正切，计算的输入正切函数，结果以弧度表示
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = TAN（0.0）；（* OUT = 0.0 *） OUT: = TAN（3.141592 / 4.0）；（* OUT≈1.0 *）

ASIN	
描述	反正弦，计算的输入的反正弦，结果以弧度表示
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = ASIN（0.0）；（* OUT = 0.0 *） OUT: = ASIN（1.0）；（* OUT = PI / 2 *）

ACOS	
描述	反余弦值，计算的输入的反余弦，结果以弧度表示
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = ACOS（1.0）; （* OUT = 0.0 *） OUT: = ACOS（-1.0）; （* OUT = PI *）

ATAN	
描述	反正切，计算输入的反正切，结果以弧度表示
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = ATAN（0.0）; （* OUT = 0.0 *） OUT: = ATAN（1.0）; （* OUT = PI / 4 *）

ADD	
描述	算术加法，计算两个输入的和。
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入类型
示例	OUT: = ADD（20，40）; （* OUT = 60 *）

MUL	
描述	算术乘法，计算两个输入相乘的积。
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入类型
示例	OUT: = MUL（10，10）; （* OUT = 100 *）

SUB	
描述	算术减法，从输入＃0减去输入＃1
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入类型
示例	OUT: = SUB（10，3）; （* OUT = 7 *）

DIV	
描述	算术除法，将输入#0除以输入#1
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入类型
示例	OUT: = DIV (20, 2) ; (* OUT = 10 *)

MOD	
描述	求余数，将输入＃0除以输入＃1，并返回余数
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入类型
示例	OUT: = MOD (10, 3) ; (* OUT = 1 *)

POW	
描述	指数运算。
操作数	2
输入数据类型	LREAL（如果有），否则为REAL; LREAL（如果有），否则为REAL;
输出数据类型	LREAL（如果有），否则为REAL
示例	OUT: = POW (2.0, 3.0) ; (* OUT = 8.0 *) OUT: = POW (-1.0, 5.0) ; (* OUT = -1.0 *)

ATAN2*	
描述	反正切（有两个参数），计算y/x的主弧正切；结果以弧度表示
操作数	2
输入数据类型	LREAL（如果有），否则为REAL; LREAL（如果有），否则为REAL
输出数据类型	LREAL（如果有），否则为REAL
示例	OUT: = ATAN2 (0.0, 1.0) ; (* OUT = 0.0 *) OUT: = ATAN2 (1.0, 1.0) ; (* OUT = PI / 4 *) OUT: = ATAN2 (-1.0, -1.0) ; (* OUT = (-3/4) * PI *) OUT: = ATAN2 (1.0, 0.0) ; (* OUT = PI / 2 *)

SINH*	
描述	双曲正弦。计算的输入的双曲正弦函数
操作数	1
输入数据类型	LREAL（如果有），否则为REAL
输出数据类型	LREAL（如果有），否则为REAL
示例	OUT: = SINH (0.0) ; (* OUT = 0.0 *)

COSH*	
描述	双曲余弦值，计算的输入的双曲余弦函数
操作数	1
输入数据类型	LREAL（如果有），否则为REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = COSH（0.0）; （* OUT = 1.0 *）

TANH*	
描述	双曲正切值，计算的输入的双曲正切函数
操作数	1
输入数据类型	LREAL（如果有），否则为REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = TANH（0.0）; （* OUT = 0.0 *）

CEIL*	
描述	舍入到整数，返回大于或等于输入的最小整数
操作数	1
输入数据类型	LREAL（如果有），否则为REAL
输出数据类型	LREAL（如果有），否则为REAL
示例	OUT: = CEIL（1.95）; （* OUT = 2.0 *） OUT: = CEIL（-1.27）; （* OUT = -1.0 *）

FLOOR*	
描述	四舍五入到整数,返回小于或等于输入的最大整数
操作数	1
输入数据类型	LREAL（如果有），否则REAL
输出数据类型	LREAL（如果有），否则REAL
示例	OUT: = FLOOR（1.95）; （* OUT = 1.0 *） OUT: = FLOOR（-1.27）; （* OUT = -2.0 *）

*：作为扩展IEC 61131-3标准所提供的功能。

位操作功能

SHL	
描述	输入＃0左移输入＃1的bit，右边填充为零。
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入#0类型
示例	OUT: = SHL（IN: = 16＃1000CAFE，16）; （* OUT = 16＃CAFE0000 *）

SHR	
描述	输入＃0右移输入＃1的bit，左侧填充为零
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入＃0类型
示例	OUT: = SHR（IN: = 16＃1000CAFE，24）； （* OUT = 16＃00000010 *）

ROL	
描述	输入＃0左移的输入＃1的bit，每个bit循环移动。
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入＃0类型
示例	OUT: = ROL（IN: = 16＃1000CAFE，4）； （* OUT = 16＃000CAFE1 *）

ROR	
描述	输入＃0右移的输入＃1的bit，每个bit循环移动。
操作数	2
输入数据类型	任何数值类型
输出数据类型	同输入＃0类型
示例	OUT: = ROR（IN: = 16＃1000CAFE，16）； （* OUT = 16＃CAFE1000 *）

AND	
描述	逻辑与，如果两个输入＃0和输入＃1是BOOL，则按逻辑与， 否则按位与运算。
操作数	2
输入数据类型	除了字符串类型
输出数据类型	同输入类型
示例	OUT := TRUE AND FALSE; (* OUT = FALSE *) OUT := 16#1234 AND 16#5678; (* OUT = 16#1230 *)

OR	
描述	逻辑或，如果两个输入＃0和输入＃1是BOOL，则按逻辑 或，否则按位或晕眩。
操作数	2
输入数据类型	除了字符串类型
输出数据类型	同输入类型
示例	OUT := TRUE OR FALSE; (* OUT = FALSE *) OUT := 16#1234 OR 16#5678;(* OUT = 16#567C *)

XOR	
描述	逻辑异或，如果两个输入＃0和输入＃1是BOOL，则按逻辑异或，否则按位异或。
操作数	2
输入数据类型	除了字符串类型
输出数据类型	同输入类型
示例	OUT := TRUE OR FALSE; (* OUT = TRUE *) OUT := 16#1234 OR 16#5678; (* OUT = 16#444C *)

NOT	
描述	逻辑非，如果输入是BOOL，则按逻辑非，否则按位取反。
操作数	1
输入数据类型	除了字符串类型
输出数据类型	同输入类型
示例	OUT: = NOT FALSE; (* OUT = TRUE *) OUT: = NOT 16#1234; (* OUT = 16#EDCB *)

选择功能

SEL	
描述	二元选择功能
操作数	3
输入数据类型	BOOL，任何类型，任何类型
输出数据类型	同选定的输入类型
示例	OUT := SEL(G := FALSE, IN0 := X, IN1 := 5); (* OUT = X *)

MAX	
描述	最大值选择功能
操作数	2，可扩展
输入数据类型	任何数值类型，任何数值类型，..，任何数值型
输出数据类型	同最大值输入类型
示例	OUT: = MAX（-8，120，-1000）; （* OUT = 120 *）

MIN	
描述	最小值选择功能
操作数	2，可扩展
输入数据类型	任何数值类型，任何数值类型，..，任何数值型
输出数据类型	同最小值输入类型
示例	OUT: = MIN（-8，120，-1000）; （* OUT = -1000 *）

LIMIT	
描述	限制输入#0大于或等于输入#1，并且等于或小于输入#2。
操作数	3
输入数据类型	任何数值类型，任何数值类型，任何数值型
输出数据类型	同输入类型
示例	OUT: = LIMIT（IN: = 4，MN: = 0，MX: = 5）；（* OUT = 4 *） OUT: = LIMIT（IN: = 88，MN: = 0，MX: = 5）；（* OUT = 5 *） OUT: = LIMIT（IN: = -1，MN: = 0，MX: = 5）；（* OUT = 0 *）

MUX	
描述	多路复用器。根据输入K选择N个输入之一。
操作数	3，可扩展
输入数据类型	任何数值类型，任何数值类型，...，任何数值类型
输出数据类型	同选定的输入类型
示例	OUT: = MUX（0，A，B，C）；（* OUT = A *）

比较功能

如果目标设备支持此功能，则还可以使用比较函数来比较字符串。

GT	
描述	大于，如果输入#0 > 输入#1则返回TRUE，否则返回FALSE。
操作数	2
输入数据类型	除了BOOL类型
输出数据类型	BOOL类型
示例	OUT: = GT（0，20）；（* OUT = FALSE *） OUT: = GT（'AZ'，'ABC'）；（* OUT = TRUE *）

通用电器	
描述	大于或等于，如果输入#0 >= 输入#1则返回TRUE，否则返回FALSE。
操作数	2
输入数据类型	除了BOOL类型
输出数据类型	BOOL类型
示例	OUT: GE =（20，20）；（* OUT = TRUE *） OUT: = GE（'AZ'，'ABC'）；（* OUT = FALSE *）

EQ	
描述	等于，如果输入#0 = 输入#1则返回TRUE，否则返回FALSE。
操作数	2
输入数据类型	任何类型
输出数据类型	BOOL类型
示例	OUT: = EQ（TRUE，FALSE）；　　（* OUT = FALSE *） OUT: = EQ（ 'AZ'， 'ABC'）；（* OUT = FALSE *）

LT	
描述	少于，如果输入#0 < 输入#1返回TRUE，否则返回FALSE。
操作数	2
输入数据类型	除了BOOL类型
输出数据类型	BOOL类型
示例	OUT: = LT（0，20）；（* OUT = TRUE *） OUT: = LT（ 'AZ'， 'ABC'）；（* OUT = FALSE *）

LE	
描述	小于或等于，如果输入#0 <= 输入#1返回TRUE，否则返回FALSE。
操作数	2
输入数据类型	除了BOOL类型
输出数据类型	BOOL类型
示例	OUT: = LE（20，20）；（* OUT = TRUE *） OUT: = LE（ 'AZ'， 'ABC'）；（* OUT = FALSE *）

NE	
描述	不等于，如果输入#0 != 输入#1返回TRUE，否则返回FALSE。
操作数	2
输入数据类型	任何类型
输出数据类型	BOOL类型
示例	OUT: = NE（TRUE，FALSE）；（* OUT = TRUE *） OUT: = NE（ 'AZ'， 'ABC'）；（* OUT = TRUE *）

字符串函数

以下功能的可用性取决于目标设备。有关详细信息，请咨询您的硬件供应商。

CONCAT	
描述	字符串拼接功能
操作数	2
输入数据类型	STRING类型
输出数据类型	STRING类型
示例	`OUT: = CONCAT ( 'AB', 'CD') ; (* OUT = 'ABCD' *)`

DELETE	
描述	删除IN从第P个字符位置开始的L个字符。
操作数	3
输入数据类型	STRING类型，UINT类型，UINT类型
输出数据类型	STRING类型
示例	`OUT: = DELETE (IN: = 'ABXYC', L: = 2, P: = 3) ; (* OUT = 'ABC' *)`

FIND	
描述	找出IN2在IN1中第一次出现的开始的字符位置。如果未找到IN2的出现，则OUT := 0
操作数	2
输入数据类型	STRING类型，STRING类型
输出数据类型	UINT类型
示例	`OUT: = FIND (IN1: = 'ABCBC', IN2: = 'BC') ; (* OUT = 2 *)`

INSERT	
描述	在第P个字符位置后将IN2插入IN1
操作数	3
输入数据类型	STRING类型，STRING类型，UINT类型
输出数据类型	STRING类型
示例	`OUT: = INSERT (IN1: = 'ABC', IN2: = 'XY', P: = 2) ; (* OUT = 'ABXYC' *)`

LEFT	
描述	取IN最左边边的L个字符
操作数	2
输入数据类型	STRING类型，UINT类型
输出数据类型	STRING类型
示例	`OUT: = LEFT (IN: = 'ASTR', L: = 3) ; (* OUT = 'AST' *)`

MID	
描述	取IN中从第P个开始的L个字符
操作数	3
输入数据类型	STRING类型，UINT类型，UINT类型
输出数据类型	串
示例	OUT: = MID（IN: = 'ASTR', L: = 2, P: = 2）; （* OUT = 'ST' *）

REPLACE	
描述	从第P个字符位置开始，用IN2替换IN1的L个字符
操作数	4
输入数据类型	STRING类型，STRING类型，UINT类型，UINT类型
输出数据类型	串
示例	OUT: = REPLACE（IN1: = 'ABCDE', IN2: = 'X', L: = 2, P: = 3）; （* OUT = 'ABXE' *）

RIGHT	
描述	取IN最右边的L个字符
操作数	2
输入数据类型	STRING类型，UINT类型
输出数据类型	STRING类型
示例	OUT: = RIGHT（IN: = 'ASTR', L: = 3）; （* OUT = 'STR' *）

TO_STRING	
描述	转换到字符串类型
操作数	1
输入数据类型	任何数值类型
输出数据类型	STRING类型
示例	str := TO_STRING(10.0); (* str = '10,0' *) str := TO_STRING(-1); (* str = '-1' *)

TO_STRINGFORMAT	
描述	转换为字符串，使用格式说明符
操作数	2
输入数据类型	任何数值类型或STRING类型
输出数据类型	STRING类型
示例	str := TO_STRINGFORMAT(10, '%04d'); (* str = '0010'*)

11.2 指令列表(IL)

本节定义IL(指令列表)语言的语义。

11.2.1 语法和语义

11.2.1.1 IL指令的语法

IL代码由一系列指令组成。每条指令从一个新行开始，包含一个带有可选修饰符的操作符，如果需要特殊的操作符，还包含一个或多个以逗号分隔的操作数。操作数可以是文字或者变量的任何表示。

指令的前面可以有一个标识标签，后面跟一个冒号(:)。可以在指令之间插入空行。

例子

让我们来解析一小段代码:

```
START:
    LD %IX1 (* Push button *)
    ANDN %MX5.4 (* Not inhibited *)
    ST %QX2 (* Fan out *)
```

构成每条指令的要素如下:

标签	操作符 (+修改器)	操作数	注释
START:	LD	% IX1	(*按钮*)
	ANDN	% MX5.4	(*不被禁止*)
	ST	% QX2	(*展开*)

IL指令语义

- 累加器

  累加器是指包含当前计算结果的值的寄存器。

- 运算符

  除非另有说明，否则运算符的语义为

  ```
  accumulator := accumulator OP operand
  ```

  也就是说，累加器的值被运算运算符对累加器本身的当前值应用运算符产生的结果所代替。例如，“AND %IX1”指令被解释为

  ```
  accumulator := accumulator AND %IX1
  ```

  如果累加器的当前值大于输入字10的值，则指令“GT %IW10”的布尔结果为TRUE，否则，指令“GT %IW10”的布尔结果为FALSE:

  ```
  accumulator := accumulator GT %IW10
  ```

- 修饰符

  修饰词“N”表示操作数的按位取反。

  左括号修饰符“（”表示运算符的评估必须推迟到遇到右括号运算符“）”为止。 括号中的指令序列的形式如下所示，请参阅该指令

  ```
  accumulator := accumulator AND (%MX1.3 OR %MX1.4)
  ```

修饰语“C”表示仅当当前评估结果的值为布尔值1时才可以执行关联的指令（如果将运算符与“ N”修饰语结合使用则为布尔值0）。

11.2.2 标准的运算符

下面列出了标准操作符及其允许的修饰符和操作数。

操作符	修饰符	支持的操作数类型： Acc_type, Op_type	语义
LD	N	任何类型	将累加器设置为操作数。
ST	N	任何类型	将累加器存储到操作数位置。
S		BOOL类型,BOOL类型	如果累加器为真，则将操作数设置为真。
R		BOOL类型,BOOL类型	如果累加器为真，则将操作数设置为假。
AND	N, (	除了REAl类型，除了REAL类型	逻辑或位和
OR	N, (	除了REAL类型，除了REAL类型	逻辑或位或
XOR	N, (	除了REAL类型，除了REAL类型	逻辑或位XOR
NOT		除了REAL类型	逻辑或位不
ADD	(	除了BOOL类型	除了
SUB	(	除了BOOL类型	减法
MUL	(	除了BOOL类型	乘法
DIV	(	除了BOOL类型	除法
MOD	(	除了BOOL类型	模块划分
GT	(	除了BOOL 类型	比较:
GE	(	除了BOOL类型	比较:=
EQ	(	除了BOOL类型	比较:=
NE	(	除了BOOL类型	比较:
LE	(	除了BOOL类型	比较:
LT	(	除了BOOL类型	比较:
JMP	C, N	标签	跳到标签
CAL	C, N	FB实例名	调用功能块
RET	C, N		从被调用的程序、函数或函数块返回。
)			计算延迟操作。

11.2.3 调用功能和功能块

11.2.3.1 调用功能

功能(如相关部分中定义的)是通过将功能名放在运算符字段中来调用的。此调用采用以下形式:

```
LD 1
MUX 5, var0, -6.5, 3.14
ST vRES
```

注意，第一个参数不包含在输入列表中，但累加器用作功能的第一个参数。如果需要，在操作数字段中提供其他参数（以第二个参数开头），按声明顺序用逗号分隔。例如，上表中的运算符MUX采用5个操作数，其中的第一个操作数加载到累加器中，而其余4个参数则在功能名之后按顺序输入。

以下规则适用于功能调用

1) 对VAR_INPUT参数的赋值可以为空、常量或变量。
2) 功能的执行在到达RET指令或功能的物理末端时结束。当这种情况发生时，功能的输出变量被复制到累加器中。

调用功能块

功能块(如相关部分中定义的)可以通过CAL操作符有条件地和无条件地调用。此调用采用以下形式:

```
LD A
ADD 5
ST INST5.IN1
LD 3.141592
ST INST5.IN2
CAL INST5
LD INST5.OUT1
ST vRES
LD INST5.OUT2
ST vVALID
```

这种调用方法相当于带有参数列表的CAL，它只包含一个具有FB实例名称的变量。
通过对操作数执行以下形式的ST / LD操作，将输入参数传递给FB实例并读取输出参数:

```
FBInstanceName.IO_var
```

说明

关键字	描述
FBInstanceName	要调用的实例的名称。
IO_var	要写入/读取的输入或输出变量。

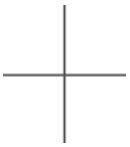
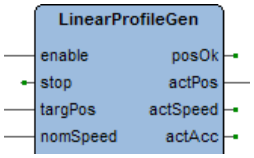
11.3 功能块图(FBD)

本节定义FBD(功能块图)语言的语义。

11.3.1 线和块的表示

图形语言元素是使用图形或半图形元素绘制的，如下表所示。

数据的存储或与数据元素的关联不能与连接器的使用相关联；因此，为避免歧义，不能给连接器任何标识符。

功能	例子
连接线	
有交叉的连接线路	
带有连接线和未连接管脚的块	

11.3.2 网络中的数据流

网络被定义为互连图形元素的最大集合。可以在右边用冒号（:）分隔的网络标签与每个网络或网络组关联。网络的范围及其标签对于网络所在的程序组织单位（POU）是本地的。

图形语言用于通过一个或多个表示控制计划的网络来表示概念量的流。即，在功能框图（FBD）的情况下，通常使用“信号流”，类似于信号处理系统中各元件之间的信号流。FBD语言中的信号流从功能或功能块的输出（右侧）到已连接的功能或功能块的输入（左侧）。

11.3.3 计算网络

11.3.3.1 网络的计算顺序

网络及其元素的计算顺序不一定与标签或显示网络的顺序相同。当程序组织单位（POU）的主体由一个或多个网络组成时，上述主体内的网络计算顺序遵守以下规则：

1) 在所有输入的状态都被求值之前，网络的任何元素都不会被求值。
2) 在对网络元素的所有输出值的状态进行评估之前，网络元素的评估是不完整的。
3) 如描述FBD编辑器时所述，网络号会自动分配给每个网络。在程序组织单位（POU）中，根据网络编号的顺序对网络进行评估：除非通过执行控制元素另外指定，否则网络N在网络N+1之前进行计算。

11.3.3.2 元素组合

FBD语言的元素必须通过信号流线相互连接。

块的输出不得连接在一起，特别是不允许使用LD语言的“wire -OR”结构，需要使用显式的布尔“OR”块。

反馈

当功能或功能块的输出用作网络中位于其之前的功能或功能块的输入时，网络中存在反馈路径。相关的变量称为反馈变量。

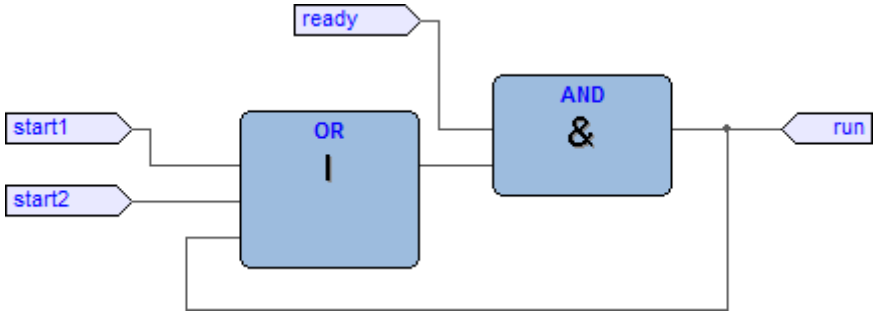
可以遵循以下规则来利用反馈路径：

1) 反馈变量必须初始化，并且在网络的第一次计算期间将使用初始值。查看 *全局变量* 编辑器，局部变量编辑器或参数编辑器，以了解如何初始化各个项目。

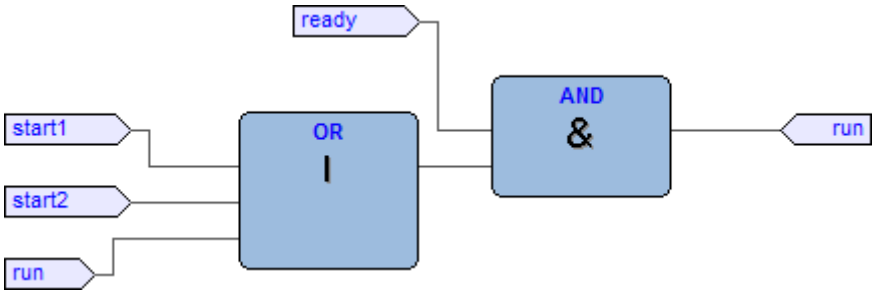
2) 一旦对具有反馈变量作为输出的元素进行了计算，就将使用反馈变量的新值，直到对元素进行下一次计算为止。

例如，在下面的示例中，布尔变量RUN是反馈变量。

显式循环



隐式循环



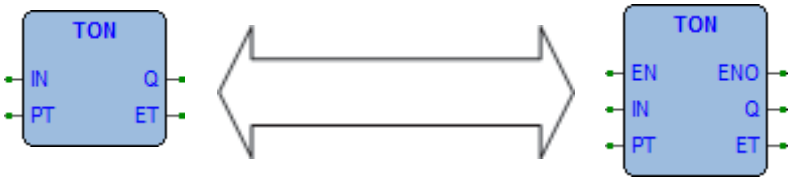
11.3.4 执行控制元素

11.3.4.1 EN / ENO引脚

根据标准，LogicLab软件中可对块附加的布尔EN（启用）输入和ENO（启用输出）引脚。

EN	ENO
VAR_INPUT EN: BOOL:= 1;END_VAR	VAR_OUTPUT ENO: BOOL;END_VAR

请参阅“*块的修改属性*”一节以了解如何将这些引脚添加到块中。



使用这些变量时，将根据以下规则控制由块定义的操作的执行：

3) 如果在调用该块时EN的值为FALSE，则不会执行功能体定义的操作，并且可编程控制器系统会将ENO的值重置为FALSE。

4) 否则，由可编程控制器系统将ENO的值设置为TRUE，并执行由块体定义的操作。

11.3.4.2 跳转

跳转由以双箭头终止的布尔信号线表示。跳转条件的信号线来源于布尔变量或功能或功能块的布尔输出。当信号线的布尔值为TRUE时，将程序控制转移到指定的网络标签。因此，无条件跳转是条件跳转的特例。

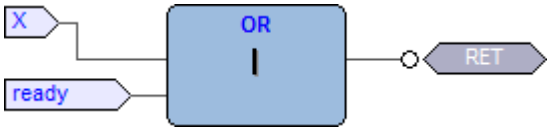
跳转的目标是程序组织单元中发生跳转的网络标签。

符号/例子	解释
	无条件转移
	条件转移

符号/例子	解释
	示例:跳转条件网络

11.3.4.3 有条件的返回

- 如下表所示，使用RETURN构造函数和功能块的条件返回。当布尔输入为TRUE时，程序执行将转移回调用实体，而当布尔输入为FALSE时以常规方式继续执行。
- 函数或功能块的物理端提供无条件返回。

符号/例子	解释
	有条件的回报
	例:返回条件网络

11.4 梯形图(LD)

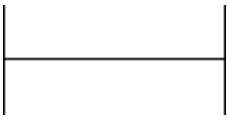
本节定义LD(梯形图)语言的语义。

11.4.1 电源轨道

LD网络在左侧由称为左电源轨的垂直线定界，在右侧由称为右电源轨的垂直线定界。正确的电源轨在LogicLab实现中可能是明确的，并且始终会显示出来。

两个电源轨始终与一条水平线连接，该水平线称为信号链路。所有LD元件都应放置并连接到信号链路上。

描述	象征符号
左侧电源轨(附水平连接线)	
右侧电源轨(附水平连接线)	

描述	象征符号
电源轨线之间由信号链路连接	

11.4.2 链接元素和状态

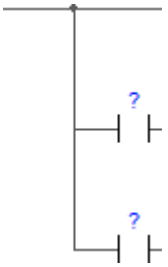
链接元素可以是水平或垂直的。链接元素的状态应表示为“ ON”或“ OFF”，分别对应于文字布尔值1或0。链路状态的术语应与电源术语同义。

以下属性适用于链接元素：

- 始终将左导轨的状态视为开启状态，没有为右导轨定义状态。
- 水平链接元素由水平线表示。水平链接元素将元素的状态从其紧邻的左侧传递到其紧邻的右侧的元素。
- 垂直连接元件由一条垂直线组成，该垂直线在每一侧与一个或多个水平连接元件相交。 垂直链接的状态表示水平链接左侧的ON状态的包含性OR，即，垂直链接的状态为：

  OFF：如果左侧所有附加水平链接的状态为OFF，则为OFF；

  ON：如果左侧的一个或多个附加水平链接的状态为ON。
- 垂直链接的状态将复制到其右侧所有附加的水平链接。
- 垂直链接的状态不会复制到左侧任何已连接的水平链接。

描述	象征符号
垂直链接和附加的水平链接	

11.4.3 触点

触点是一种元素，该元素赋予右侧的水平链接状态一个状态，该状态等于左侧水平链接状态的布尔与，并且具有关联的布尔输入，输出或存储变量的适当功能 。

出现不会修改关联的布尔变量的值。下表列出了标准的出现符号。

名称	描述	象征符号
常开触点	如果关联的布尔变量的状态为ON，则将左侧链接的状态复制到右侧链接。 否则，右链接的状态为OFF。	
常闭触点	如果关联的布尔变量的状态为OFF，则将左侧链接的状态复制到右侧链接。 否则，右链接的状态为OFF。	
正向传递触点	当在左侧链接的状态为ON的同时感测到相关变量从OFF到ON的过渡时，从当前计算到下一次计算，右侧链接的状态为ON。右链接的状态在所有其他时间均应为OFF。	
反向传递触点	当在左侧链接的状态为ON的同时感测到相关变量从ON到OFF的转换时，从当前计算到下一次计算，右侧链接的状态为ON。右链接的状态在所有其他时间均应为OFF。	

11.4.4 线圈

线圈将其左侧的链接状态复制到右侧的链接，而无需进行修改，并将状态的适当函数或左侧链接的转移存储到关联的布尔变量中。

下表中显示了标准线圈符号。

名称	描述	象征符号
线圈	左侧链接的状态将复制到关联的布尔变量。	
否定线圈	左链接状态的反函数复制到关联的布尔变量，即，如果左链接的状态为OFF，则关联变量的状态为ON，反之亦然。	
置位(子锁)线圈	当左链接处于ON状态时，关联的布尔变量设置为ON状态，并保持设置状态，直到被RESET线圈复位为止。	

名称	描述	象征符号
重置(解锁)线圈	当左链接处于ON状态时，关联的Boolean变量将重置为OFF状态，并保持重置状态，直到由SET线圈进行设置为止。	-(R)-
正向传递线圈	当检测到左链接从OFF到ON的转换时，从该元素的一个评估到下一个评估，关联的布尔变量的状态为ON。	-(P)-
反向传递线圈	当检测到左链接从ON到OFF的过渡时，从该元素的一个评估到下一个评估，关联的布尔变量的状态为ON。	-(N)-

11.4.5 运算符、功能和功能块

LD语言中功能和功能块的表示形式与FBD中使用的类似。每个块上至少应显示一个布尔输入和一个布尔输出，以允许信号流通过块，如下图所示。

11.5 结构化文本(ST)

本节定义了ST(结构化文本)语言的语义。

11.5.1 表达式

表达式是一种构造，当对其求值时，将产生一个值，该值对应于基本数据类型表中列出的数据类型之一。 LogicLab并未对表达式的最大长度设置任何约束。

表达式由运算符和操作数组成。

11.5.1.1 操作数

操作数可以是文字、变量、函数调用或另一个表达式。

11.5.1.2 运算符

打开运算符表以查看ST支持的所有运算符的列表。对表达式的求值包括按运算符优先级规则定义的顺序将运算符应用于操作数。

11.5.1.3 运算符优先级规则

运算符在运算符表中指定不同的优先级。表达式中首先运算具有最高优先级的运算符，然后是下一个较低优先级的运算符，依此类推，直到计算完成。优先级相同的运算符按照从左到右的表达式中的说明进行应用。

例如，如果A，B，C和D的类型分别为INT，值1、2、3和4，则：

```
A+B-C*ABS(D)
```

结果为-9，不同的是：

```
(A+c)*ABS(D)
```

结果为0。
当一个运算符有两个操作数时，最左边的操作数将首先求值。 例如，在表达式中

```
SIN(A)*COS(B)
```

先求SIN(A)表达式的值，然后求COS(B)表达式的值，最后求乘积的值。

如标准中所定义的那样，功能作为表达式的元素被调用，表达式由功能名后跟带圆括号的参数列表组成。

11.5.1.4 ST语言的运算符

操作	符号	优先级
括号	(<表达式>)	最高
功能求值	<帧> (< arglist >)	.
取反	- NOT	. .
求幂	* *	.
乘除运算与模块划分	* / MOD	. . .
加减运算	+ -	. .
比较运算	<, >, <=, >=	.
对等运算	= <>	. .
布尔和	AND	.
布尔异或	XOR	.
布尔或	OR	最低

11.5.2 ST语句声明

所有声明均符合以下规则：

- 以分号结尾；
- 与IL不同，回车符或换行符与空格符相同；
- LogicLab对语句的最大长度不设置任何约束。

ST语句根据其语义可以分为几类。

11.5.2.1 赋值语义

赋值语句通过计算表达式的结果替换单个或多个元素变量的当前值。

赋值语句还用于分配功能返回的值，方法是将功能名放在赋值操作符左侧的功能声明体中。功能返回的值是对这种赋值的最近一次计算的结果。

语法

赋值语句由左侧的变量引用组成，后跟赋值运算符“：=”，后跟要求值的表达式。 例如，该语句

```
A := B;
```

如果两个变量均为INT类型，则将用变量B的当前值赋值给变量A。

例子

```
a := b;
```

赋值

```
pCV := pCV + 1;
```

赋值

```
c := SIN(x);
```

带有功能调用的赋值

```
FUNCTION SIMPLE_FUN: REAL
variables declaration
...
function body
...
SIMPLE_FUN:= a * b - c;
END_FUNCTION
```

将输出值赋给一个功能

11.5.2.2 功能和功能块语句

- 功能被调用为表达式的元素，表达式由功能名和圆括号中的参数组成。每个参数可以是文字、变量或任意复杂的表达式。
- 功能块由一个语句调用，该语句由功能块实例的名称和一个圆括号括起来的参数列表组成。支持使用形式参数列表和参数分配进行调用。
- RETURN：功能和功能块控制语句由调用功能块和在功能或功能块的运算结束之前将控制返回给调用实体。RETURN语句可提早退出功能或功能块（例如，作为IF语句求值的结果）。

语法

1) 功能:

```
dst_var := function_name(arg1, arg2, …),argN);
```

2) 功能块与形参列表:

```
instance_name(var_in1 := arg1,
                  var_in2 := arg2,
                   …,
                  var_inN := argN);
```

3) 带有参数赋值的功能块:

```
instance_name。var_in1 := __arg1;
…
instance_name。var_inN :=
argN;instance_name ();
```

4) 功能和功能块控制语句:

```
RETURN;
```

例子

```
CMD_TMR(IN := %IX5,

PT := 300);
```

具有实参列表的FB调用:

```
IN := %IX5 ;

PT := 300 ;

CMD_TMR() ;
```

参数赋值的FB调用:

```
a := CMD_TMR.Q;
```

FB输出用法:

```
RETURN;
```

从函数或函数块中提前退出

11.5.2.3 选择语句

选择语句包括IF和CASE语句。选择语句根据指定的条件选择一个(或一组)语句执行。

- IF: IF语句指定只有在关联的布尔表达式的计算结果为TRUE时才执行一组语句。 如果条件为假，则不执行任何语句，或者执行紧随ELSE关键字（如果关联的布尔条件为true，则为ELSIF关键字）的语句组。
- CASE：CASE语句由一个表达式组成，该表达式的计算结果为DINT类型的变量（ *选择器* ），以及一个语句组列表，每个组用一个或多个整数或整数范围（如果适用）标记。它指定要执行的第一组语句（其中一个范围包含选择器的计算值）。如果选择器的值在任何情况下都没有出现，则执行关键字ELSE后面的语句序列（如果它存在于CASE语句中）。否则，将不执行任何语句序列。

LogicLab没有对CASE语句的最大 *选择器* 数量做任何约束。

语法

请注意，方括号包含可选代码，而括号则包含代码的可重复部分。

1) IF:

```
IF expression1 THEN
stat_list
[ { ELSIF expression2 THEN
stat_list } ]
ELSE
stat_list
END_IF ;
```

2) CASE:

```
CASE expression1 OF
intv [ {, intv } ] :
stat_list
{ intv [ {, intv } ] :
stat_list }
[ ELSE
stat_list ]
END_CASE ;
intv being either a constant or an interval: a or a..b
```

例子

IF语句:

```
IF d 0.0 THEN
nRoots := 0 ;
ELSIF d = 0.0 THEN
nRoots := 1 ;
x1 := -b / (2.0 * a) ;
ELSE
nRoots := 2 ;
x1 := (-b + SQRT(d)) / (2.0 * a) ;
x2 := (-b - SQRT(d)) / (2.0 * a) ;
END_IF ;
```

CASE语句:

```
CASE tw OF
1, 5:
display := oven_temp ;
2:
display := motor_speed ;
3:
display := gross_tare;
4, 6..10:
display := status(tw - 4) ;
```

```
 ELSE
display := 0;
tw_error := 1;
END_CASE ;
```

11.5.2.4 循环语句

循环语句指定重复执行关联语句组。如果可以预先确定循环次数，则使用FOR语句。否则，将使用WHILE或REPEAT语句。

- FOR：FOR语句指示重复执行直到END_FOR关键字为止，同时将执行次数分配给FOR循环控制变量。控制变量，初始值和最终值是相同整数类型（例如SINT，INT或DINT）的表达式，并且不能由任何重复的语句更改。FOR语句以表达式的值确定的增量将控制变量从初始值向上或向下递增到最终值。该值默认为1。在每次迭代的开始都对终止条件进行检测，因此，如果初始值超过最终值，则不执行语句序列。
- WHILE：WHILE语句使直到END_WHILE关键字的语句序列重复执行，直到关联的布尔表达式为false。如果表达式最初为假，则该语句组根本不会执行。
- REPEAT：REPEAT语句将会重复执行直到UNTIL关键字（至少执行一次），或者直到关联的布尔条件为TRUE。
- EXIT：EXIT语句用于在满足终止条件之前终止循环。 当EXIT语句位于嵌套的循环结构中时，退出动作将从EXIT所在的最内层循环开始，也就是说，控制权传递到EXIT语句之后的第一个循环终止符（END_FOR，END_WHILE或END_REPEAT）之后的下一个语句 。

注意：WHILE和REPEAT语句不能用于实现进程间同步，例如，作为具有外部确定的终止条件的"等待循环"。 为此必须使用定义的SFC元素。

语法

注意，方括号包含可选代码，而大括号包含代码的可重复部分。

1) FOR:

```
FOR control_var := init_val TO end_val [ BY increm_val ] DO

stat_list

END_FOR;
```

2) WHILE:

```
WHILE expression DO

stat_list

END_WHILE;
```

3) REPEAT:

```
REPEAT
stat_list
UNTIL expression
END_REPEAT;
```

例子

FOR语句:

```
  j:= 101;
  FOR i:= 1 TO 100 BY 2 DO
  IF arrvals[i] = 57, THEN
   j := i;
   EXIT ;
   END_IF;
  END_FOR;
```

WHILE语句:

```
  j:= 1;
  WHILE j <= 100 AND arrvals[i] <> 57 DO
  J := j + 2;
  END_WHILE;
```

REPEAT语句:

```
j:= -1;
REPEAT
  j:= j + 2;
  UNTIL j = 101, AND rrvals[i] = 57
  END_REPEAT;
```

11.6 程序功能图(SFC)

本节定义了顺序功能图（SFC）元素，以构造PLC程序组织单位（POU）的内部组织，标准中定义的SFC语言目的是执行顺序控制功能。本节中的定义源自IEC 848，并进行了必要的更改，以将表示形式从标准文档转换为PLC程序组织单元的一组执行控制元素。

由于SFC元素需要存储状态信息，因此可以使用这些元素构成的程序组织单元只有功能块和程序。

如果将程序组织单位的任何部分划分为SFC元素，则将整个程序组织单位划分为SFC元素。如果没有为程序组织单位提供SFC分区，则整个程序组织单位都将被视为在调用实体的控制下执行的单个操作。

SFC元素

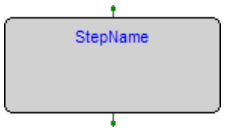
SFC元素提供了一种将PLC程序组织单元划分为由有向链接互连的一组步和转换的方法。与每个步相关联的是一组动作，并且与每个转换都关联一个转换条件。

11.6.1 步

11.6.1.1 定义

步表示这样一种情况，即程序组织单元(POU)相对于其输入和输出的行为遵循一组由步的关联定义的规则。步可能是活动的或非活动的。在任何给定时刻，程序组织单元的状态由一组活动步及其内部和输出变量的值定义。

步由包含标识符形式的步名称的块图形表示。进入步的定向链接可以用附在步顶部的垂直线来图形化地表示。该步之外的定向链接可以用附加到该步底部的垂直线表示。

图形表示	描述
	步 （带有直接链接的图形表示）

LogicLab对每个SFC的最大步数没有做任何限制。

步状态

可以用布尔变量*** _ x的逻辑值表示步状态（步的活动或不活动状态），其中***是步名称。布尔变量在相应步处于活动状态时的值为TRUE，而在其无效时为FALSE。步名称和步状态的使用范围仅限于步的程序组织单元本地。

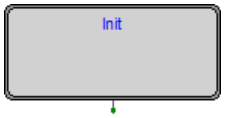
表示	描述
`Step Name_x`	步标志 = TRUE，Name_x步处于激活时 = FALSE，Name_x步处于不激活时

用户不能直接写步状态。

11.6.1.2 初始步

程序组织单元的初始状态由其内部变量和输出变量的初始值，以及其初始步组（即最初处于活动状态的步）表示。每个SFC网络或其文本等效文件都只有一个初始步。可以用双线画出边界的图形作为初始步，如下所示。对于系统初始化，普通步的默认初始状态为FALSE，初始步的默认初始状态为TRUE。

LogicLab无法编译一个不包含初始步的SFC网络。

图形表示	描述
	初始步骤 (具有直接链接的图形表示)

11.6.1.3 动作

一个动作可以是:

- IL语言编写的指令集;
- FBD语言编写的网络集;
- LD语言编写的梯形图集;
- ST语言编写文本代码集;
- 本节定义的顺序功能图SFC编写的网络集。

每个步可以关联零个或多个动作。动作是通过下表中列出的文本结构元素来声明的。

结构化元素	描述
STEP StepName : (* Step body *) END_STEP	步(文本形式)
INITIAL_STEP StepName : (* Step body *) END_STEP	初始步(文本形式)

对于每一个步至少有一个关联的动作，这种结构化元素将存在于lsc文件中。

11.6.1.4 动作限定符

与步关联的动作执行的时间取决于其动作限定符。

LogicLab实现以下动作限定符。

限定符	描述	意义
N	不存储(零限定符)	只要步保持活动状态，就会执行动作。
P	脉冲	每个步激活后只执行一次动作，而不考虑步保持活动的周期数。

如果某个步的关联动作为零，则将其视为具有WAIT功能，即等待后续转移条件变为真。

11.6.1.5 跳转

SFC数据流直接链接仅向下流动。因此，如果要从较低的位置返回较高的步，则不能从后者到前者绘制逻辑连线。存在一种称为跳转的特殊类型的块，它使您可以实现这种过渡。

跳转块在逻辑上等效于步，因为它们与转换不同。跳转的唯一作用是激活上一步的步状态，并激活它指向的步的状态。

表示	描述
	跳 (逻辑链接到目标步)

11.6.2 转换

11.6.2.1 定义

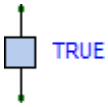
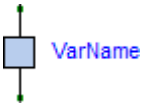
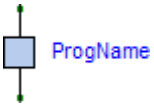
转换表示控制从转换之前的一个或多个步转到相应定向链接的一个或多个后续步的条件。转换由垂直定向链接上的小灰色正方形表示。

需要遵循的变换方向是从先前步的底部到后续步的顶部。

11.6.2.2 转换状态

每个转换都有一个关联的转换条件，该条件是对单个布尔表达式求值的结果。始终为true的转换条件用关键字TRUE表示，而始终为false的转换条件用关键字FALSE表示。

可以通过以下方式之一将转换条件与转换相关联：

图形表示	描述
	通过将对应的布尔常量{TRUE, FALSE}放置在垂直链接的旁边。
	通过声明一个布尔变量，其值决定是否转换。
	通过用LogicLab支持的任何语言(SFC除外)编写一段代码，这种代码的计算结果决定了转换条件。

转换名称作用的范围是该转换所在的程序组织单位（POU）的局部范围内。

11.6.3 过渡规则说明

介绍

SFC网络的初始状态的特征在于初始步，该初始步在初始化包含该网络的程序或功能块时处于活动状态。

当一个或多个转移被清除时，步的活动状态的将会沿着定向链路移动。

当通过定向链接到相应转换符号的所有前面的步均处于活动状态时，将使能转换。启用转换且关联的转换条件为true时，将清除转换。

清除转换会导致通过定向链接连接到相应过渡符号的所有紧接在前的步被停用（或“重置”），然后激活紧随其后的所有步。

在SFC元素的连接中，始终保持了交替步/转换和转化/步，即:

- 两个步从来没有直接联系;它们总是被一个转换分开;
- 两个转变从来没有直接联系在一起;他们之间总是隔着步。

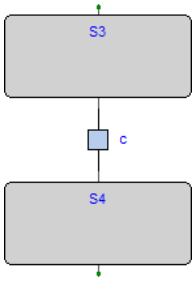
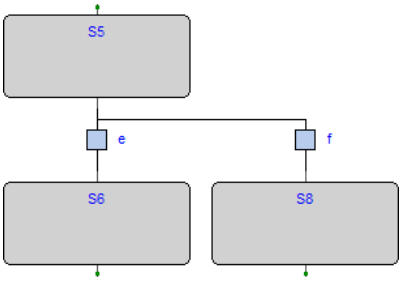
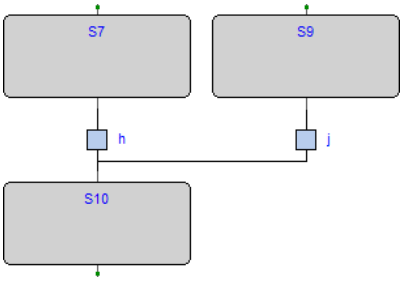
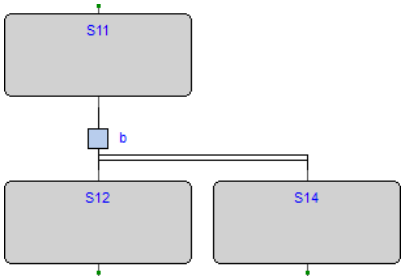
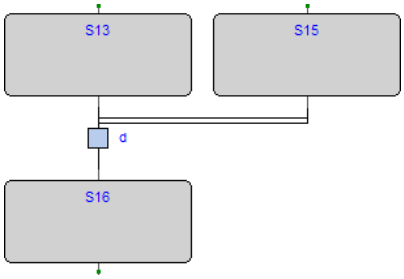
当清除一个转移导致同时激活几个步时，这些步所属的序列称为同步序列。

在它们同时激活之后，这些序列中的每一个的进化都变得独立。为了强调这种结构的特殊性，同时序列的发散和收敛用一条双水平线表示。

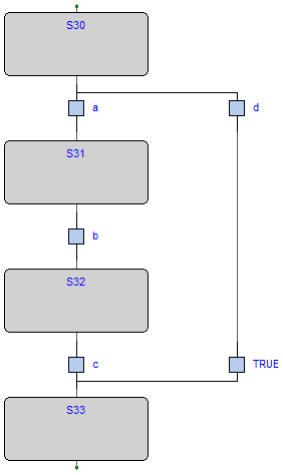
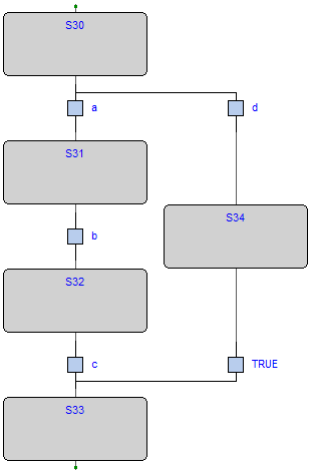
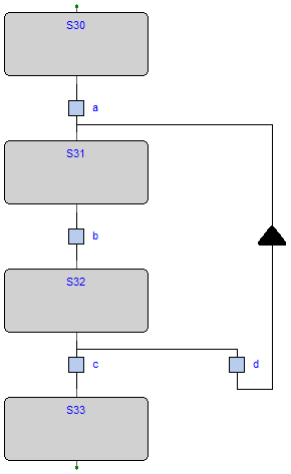
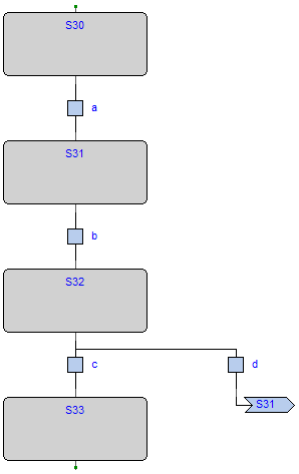
从理论上讲，转换的清除时间只是一个预期的时间，但是它永远不能为零。实际上，清除时间将由PLC实施施加：在特定PLC实施的时序约束和序列演变表中定义的优先级约束内，可以同时清除多个转换。出于同样的原因，步活动的持续时间也永远不会被视为零。在激活步的影响已经在声明该步的整个程序组织单元中传播之前，不得对活动步骤的后继转移条件进行测试。

序列运行说明

该表定义了允许的步和转换组合的语法和语义。

示例	规则
	正常的转换 当且仅当步S3处于激活状态并且转变条件c为真时，才发生从步S3到步S4的过渡。
	发散的转换 当且仅当S5有效且转换条件e为TRUE时，才从S5过渡到S6；仅当S5有效且f为TRUE且e为FALSE时，才从S5过渡至S8。
	收敛的转换 仅当S7有效且转换条件h为TRUE时，才从S7过渡 到S10；仅当S9有效且j为TRUE时，才从S9过渡到S10。
	同步发散转换 仅当S11处于活动状态并且与公共转换关联的转换条件b为TRUE时，才会从S11过渡到S12，S14...。于此同时激活S12，S14后，每个序列的动作独立进行。
	同步收敛转换 仅当以上所有步并连接到双水平线且所有步均处于活动状态且与公共转换相关联的转换条件d为TRUE时，才会从S13，S15，...过渡到S16。

例子

错误的示例	正确的示例	说明
		预期行为：如果a为FALSE而d为TRUE，则会发生从S30到S33的过渡。 最左边一栏中的方案是无效的，因为条件d和TRUE是直接链接的。
		预期行为：如果c为FALSE且d为TRUE，则会发生从S32到S31的过渡。 最左边一栏中的方案是无效的，因为直接链接仅向下流动。 可以通过跳转块执行向上转换。

11.6.4 SFC控制标志

LogicLab为SFC程序或功能块提供了一些控制标志。

要启用此功能，请参阅第4.6.2段。

这些标志是:

- `<POU name>_HOLD_SFC (type BOOL);`
- `<POU name>_RESET_SFC (type BOOL).`

其中<POU name>表示SFC POU(程序或函数块)的名称。

例如，如果SFC POU被命名为Main，那么控制标志将被命名为Main_HOLD_ SFC和Main_RESET_SFC。

每个SFC动作都可以使用另外两个动作，这些动作也包含在SFC POU中。
例如，如果上述程序Main包含一个名为Execute的SFC操作，则此操作的控件标志将为Main_Execute_HOLD_SFC和Main_Execute_RESET_SFC。

这些标记的功能将在下一段落中详细描述。

11.6.2.3 保持标志位

以下是<POU name>_HOLD_SFC标志的主要特征:

- 默认值为FALSE;
- 当设置为TRUE时，所引用的SFC块（与<POU name>相同的名称）将保持当前状态（保持），并且不执行任何代码;
- 当标志重新设置为FALSE时，将通过<POU name> _HOLD_SFC := TRUE从设置为保持的点恢复SFC块执行。

11.6.2.4 重置标志位

以下是<POU name>_RESET_SFC标志的主要特征:

- 默认值为FALSE;
- 当设置为TRUE时，被引用的SFC块（与<POU name>相同的名称）将返回初始状态，即初始动作的执行状态。
- 这是一个自动重置标志，这意味着如果将其设置为TRUE，则执行重置操作后，他自己的状态将变为FALSE。 因此，不必将<POU name> _RESET_SFC值重新设置为FALSE。

11.6.2.5 标志位的可见性

<POU name> _HOLD_SFC和<POU name> _RESET_SFC标志是从代码编译器自动生成的，它们属于所引用的POU的局部变量。

LogicLab不会在POU的变量列表中显示此标志。 它们是隐藏的，但是无论如何它们都可以在代码中的任何地方使用。

11.6.5 从其他程序中检查SFC POU

为了管理来自其他程序的SFC POU， LogicLab提供了以下功能:

- 编译器自动生成<POU name>_RESET_SFC和<POU name> _HOLD_SF标志。
- 如果SFC POU是一个功能块，则用户可以将其声明为VAR_INPUT属性的BOOL类型的变量，且可以使用作为SFC POU的控制标志的名称。
- 如果SFC POU是一个程序，用户可以使用VAR_GLOBAL属性的BOOL类型的变量，且可以使用作为SFC POU控制标志的名称。
- 在以上两种情况下，LogicLab编译器都将使用在VAR_INPUT或VAR_GLOBAL变量中声明的变量，而不是自动生成的变量（因此将不会生成在列表中）。

使用这些技术，用户可以使用SFC POU的INPUT变量从其他POU管理SFC POU的工作状态。

例子

```
FUNCTION_BLOCK test
    VAR_INPUT
    …
    test_RESET_SFC : BOOL; (* Control flag explicitly declared *)
    END_VAR
…
END_FUNCTION_BLOCK
PROGRAM Main
    VAR
    …
    block : test; (* SFC block instance *)
    END_VAR
    …
    (* Reset SFC block state *)
    block.test_RESET_SFC := TRUE;
    …
END_PROGRAM
```

11.6.5.1 SFC宏库

LogicLab向用户提供了一个名为SFCControl.pll的库，以允许通过命令而不是变量设置来管理SFC状态。

该库由仅可用于ST语言的宏组成。

11.6.5.2 控件标志的使用示例

下面是一些使用控制标志的例子，假设SFC POU被命名为Main:

- 保持(冻结):

```
Main_HOLD_SFC: = TRUE;
```

- 从保持状态下重新启动:

```
Main_HOLD_SFC: = FALSE;
```

- 在保持状态下重新启动且将SFC块置为初始状态：:

```
Main_RESET_SFC := TRUE;
Main_HOLD_SFC := FALSE;
```

- 重置为初始状态并立即重启SFC块：:

```
Main_RESET_SFC := TRUE; (* automatic reset from compiler *).
```

11.7 LOGICLAB语言扩展

LogicLab对IEC 61131-3标准进行了一些扩展，以进一步丰富语言并适应不同的编码风格。

11.7.1 宏

LogicLab实现宏的方式与C语言预处理器实现宏的方式相同。

宏可以使用以下语法定义:

```
MACRO <macro name>
PAR_MACRO
      <parameter list>
END_PAR
<macro body>
END_MACRO
```

请注意，参数列表最终可能为空，从而区分了不带参数的类对象宏和带参数的类函数宏。

下面是宏定义的一个具体示例，该示例占用两个字节并组成一个16位字：

```
MACRO MAKEWORD
PAR_MACRO
lobyte;
hibyte;
END_PAR
{ CODE:ST }
lobyte + SHL(TO_UINT( hibyte ),  8)
END_MACRO
```

只要宏名称出现在源代码中，它就会被宏主体替换（与实际的参数列表一样，在函数类宏的情况下）。例如，给定宏MAKEWORD的定义和以下结构化文本代码片段：

```
w:= MAKEWORD(b1, b2);
```

宏预处理程序将其扩展为

```
w:= b1 + SHL(TO_UINT(b2),  8);
```

11.7.2 指针

指针是一种特殊类型的变量，它充当对另一个变量（指向变量）的引用。实际上，指针的值是指向变量的地址；为了访问存储在指向的地址处的数据，可以取消对指针的引用。

指针声明与变量声明中使用的语法相同，其中类型名是指向变量的类型名，其后带有@符号：

```
VAR
<pointer name> : @<pointed variable type name>;
END_VAR
```

例如，指向REAL变量的指针的声明应如下所示：

```
VAR
px: @REAL;
END_VAR
```

可以为一个指针分配另一个指针或一个地址。 特殊运算符ADR可用于检索变量的地址。

```
px := py; (* px and py are pointers to REAL (that is, variables of type
@REAL) *)
px := ADR( x ) (* x is a variable of type REAL *)
px := ?x (* ? is an alternative notation for ADR *)
```

@运算符用于取消引用指针，从而访问指向的变量。

```
px := ADR( x );
@px := 3.141592; (* the approximate value of pi is assigned to x *)
pn := ADR( n );
n := @pn + 1; (* n is incremented by 1 *)
```

注意，不小心使用指针有潜在的危险：确实，指针可以指向任意位置，这可能会导致不良后果。

11.7.3 WAITING声明

LogicLab实现了一个WAITING语句，该语句可以在ST代码中使用，如下例所示：

```
...
WAITING <condition> DO
            <code to be executed waiting for condition becomes true>
END_WAITING;
...
```

在不验证条件之前，将执行代码（不是在循环周期中执行，而是在每次执行中返回给调用方）。

仅在启用了关联的项目选项的情况下才能使用WAITING语句（有关更多详细信息，请参见第4.6.2节）。

12. 错误的参考

12.1 编译时错误消息

错误代码	简短的描述	说明
A4097	对象未找到	指示的对象（变量或功能块）在应用程序中没有进行定义。
A4098	数据类型不支持	指示数据类型所要求的大小（位数）得不到目标系统的支持。
A4099	自动变量空间耗尽	所有本地变量所要求的总分配空间超过了目标系统上的可用空间。
A4100	保持型变量空间耗尽	所有本地保持变量所要求的总分配空间超过了目标系统上的可用空间。
A4101	位变量空间耗尽	所有本地位（布尔）变量所要求的总分配空间超过了目标系统上的可用空间。
A4102	运算符++在数据块上无效	所指示的变量与在相对数据块中不可用的索引相关联。
A4103	数据块未找到	所指示的变量与一个在目标系统中不存在（没定义）的数据块相关联。
A4104	代码空间耗尽	用于POU代码（程序，功能和功能块）的总空间超过目标系统的可用空间。
A4105	无效位偏移	所指示的变量与在相对数据块中不可用的位索引相关联。
A4106	要求图片变量	取代为错误代码。
A4107	目标功能未找到	所指示的功能在目标系统中不可用。
A4108	基础对象未找到	所指示的实例涉及到一种没有定义的功能块定义。
A4109	基础对象类型无效	所指示的变量与不被定义的数据类型（包括功能块定义）相关联。
A4110	数据类型无效	用于变量定义的数据类型不存在。
A4111	操作类型无效	操作类型不允许当前的操作。
A4112	功能块共享全局数据，并被更多的任务所使用	该指示的功能块由多个任务调用，但使用带有过程映像的全局变量。由于这个原因，编译器无法为功能块的每个实例引用正确的图像变量。
A4113	临时变量分配错误	内部编译器错误。
A4114	功能不支持将数组用作输入变量	
A4115	输入到功能的参数过多	
A4116	增量编译失败，执行重建命令	
A4117	可用数据空间10%以下	
A4118	可用保持性数据空间10%以下	

错误代码	简短的描述	说明
A4119	可用自由位数据10%以下	
A4120	超过数据块空间的变量	
A4121	元素未找到	
A4123	访问私有成员无效	
A4129	不是结构体类型	
A4130	不是一个功能块实例	
A4131	不兼容的外部声明	
A4133	不是一个变量	
A4134	索引超过数组大小	
A4135	索引数据类型无效	
A4136	缺少索引（es）	
A4137	所需的功能块实例	
A4138	所需的简单变量	
A4139	过多的索引	
A4140	不是一个结构实例	
A4141	不是一个数组	
A4143	不是一个指针	
A4144	不允许多级间接寻址	
A4145	待实施	
A4146	位数据类型不允许	
A4147	无法计算变量偏移	
A4148	复杂的变量不能过程映像	
A4149	在功能块不能使用过程图像直接表示的变量（未实现）	
A4150	功能块实例不允许	
A4151	结构不允许	
A4152	16位的变量必须与16位边界对齐	
A4153	32位的变量必须与32位边界对齐	
A4154	临时字符串变量分配错误。指令将被拆分。	
A4155	Ext/aux自动变量空间耗尽	
A4156	枚举值不明确，需要<enum>＃前缀	
B0001	数据块未找到	所指的变量与一个在目标系统中不存在（没有定义）的数据块相关联。
B0002	创建文件错误	所指的文件由于一个系统文件错误或一个丢失的源文件而不能被创建。
C0001	解析器未初始化	内部编译器错误。
C0002	无效指令	该字节在当前语法环境下无效

错误代码	简短的描述	说明
C0003	无效的文件规范	内部编译错误。
C0004	不能打开文件	由于文件系统错误或缺少源文件，无法打开指定的文件。
C0005	解析器表错误	内部编译错误。
C0006	解析器不指定	内部编译错误。
C0007	文件意意外关闭	指示的文件被截断或语法不完整。
C0009	保留关键字	所指示的单词不能用于声明目的，因为它是该语言的关键字。
C0010	无效的元素	所指示的单词对于语言语法来说不是有效的。
C0011	由用户中止	
C0032	宏调用中的参数太多	
C0033	宏调用中的参数数目无效	
C0034	嵌套了太多的宏调用	
C4097	无效的变量类型	不允许使用指定的数据类型。
C4098	无效的位置前缀	指定变量的地址字符串不正确，'%'丢失。
C4099	无效的位置规范	所指示的变量的地址字符串不正确，数据访问类型指示不是'I','Q'或'M'。
C4100	无效的位置类型	所指示变量的地址字符串不正确，数据类型指示不是“X”、“B”、“W”、“D”、“R”或“L”。
C4101	无效的位置索引说明	指定变量的地址字符串不正确，索引不正确。
C4102	重复的变量名	所指示的变量的名称已经用于其他项目对象。
C4103	此处只允许数字0	编译器只使用基于数组零索引的
C4104	无效的数组维度	数组的维数没有以正确的方式表示(例如:包含无效字符、负数等)。
C4105	常数没有初始化	每个常数都需要有一个初始值。
C4106	无效的字符串的大小	
C4107	初始化超出字符串大小	
C4108	无效的重复初始化	
C4109	初始化的数据类型无效	
C4353	复制标签	所指示的标签已在当前POU（程序，功能或功能块）中定义。
C4354	常量不被承认	所指示的操作不允许使用常量(通常是存储或分配操作)。
C4355	未定义的显式常量地址	
C4356	超出下标的最大数目	
C4358	无效的数组基础	
C4359	无效的操作数	
C4609	无效的二进制常数	带有2#前缀的常量值必须只包含二进制数字(0或1)。

错误代码	简短的描述	说明
C4610	无效的八进制常数	前缀为8#的常量值必须只包含八进制数数字(0到7之间)。
C4611	无效的十六进制常数	带有16#前缀的常量值必须只包含十六进制数字(在0和9之间，在A和F之间)。
C4612	无效的十进制常数	十进制常数必须只包含0到9之间的数字、前导符号+或-、十进制分隔符'。或指数指标'e'或'e'。
C4613	无效的时间常数	一个带t#前缀的常量值必须包含一个十进制的时间指示，并且在“ms”、“s”或“m”之间有一个时间单位。
C4614	无效的常量字符串	
C4864	重复的函数名	所指示的函数名已经用于另一个应用程序对象。
C4865	无效的函数类型	指定函数返回的数据类型不正确。
C5120	重复的程序名	所指示的程序名已经用于另一个应用程序对象。
C5376	重复函数块名	所指示的函数块名称已经用于另一个应用程序对象。
C5632	无效的编译指示	
C5633	无效的编译指示值	
C5889	重复宏名称	
C5890	复制宏参数名	
C6144	无效的资源定义:两个或更多的任务具有相同的ID	
C16385	无效的初始化值	
C16386	无效的初始化定义	
C16387	无效的数组分隔符(括号)	
C16388	无效的初始化值	
C16389	数组初始化值为空	
C16390	无效的重复初始化值	
C16391	没有实现	
C16392	缺少数组分隔符(方括号)	
C16393	缺少逗号	
C16394	没有实现	
C16395	无效(不完全)字符串	
D12289	不能分配数据库	参数数据库所需的内存空间超过了目标系统上可用的空间。如果可能，请删除未使用的参数的记录，菜单等。
D12290	不能分配数据库记录	参数数据库所需的内存空间超过了目标系统上可用的空间。如果可能，请删除未使用的参数的记录，菜单等。
D12291	数据库变量未找到	内部编译错误。
D12292	无效的表达式或表达式语法错误	指示结果的数据库表达式不正确，包含语法错误或无效的运算符。

错误代码	简短的描述	说明
D12293	表达式中的参数引用无效	具有指定结果的数据库表达式包含一个参数(作为操作数)，该参数与表达式所引用的参数不同。该表达式只能使用PLC变量（包括与参数关联的变量）和当前交换的参数值。 例如：pDELTA = DELTA / pRATIO + pOFFSET是正确的，因为交换的参数是DELTA，并且它是表达式中使用的唯一参数值。 表达式：pDELTA = DELTA / pRATIO + OFFSET不正确，因为表达式中使用的参数OFFSET当前未交换。
D12294	递归表达式	具有指定结果的数据库表达式通过使用包含当前表达式结果的操作数调用自身。
D12295	表达式中的未解析的变量	具有指定结果的数据库表达式使用的操作数在整个PLC项目中没有定义。
D12296	无法解析的表达结果	内部编译错误。
D12297	表达式的结果类型无效	表达式结果的参数的数据类型无效（例如枚举）或未定义。
D12298	表达式中的操作数无效	具有指定结果的数据库表达式使用无效的操作数。
D12299	表达式的变量类型无效	作为表达式结果的变量的数据类型无效（例如枚举）或未定义。
D12300	汇编程序错误	内部编译错误。
D12301	不能分配数据库代码	表达式所需的代码空间已用完。从参数的数据库中删除一些表达式是必要的。
D12302	表达式操作无效	具有指定结果的数据库表达式使用无效的操作数。
F1025	无效的网络	所指示的FBD或LD网络包含一个连接错误(错误通常由红色连接表示)。
F1026	未连接的引脚	所指示的块(操作符、功能、接触或线圈)有一个未连接的引脚。
F1027	无效连接(不完整，超过一个来源等)	内部编译错误。
F1028	每个块有多个网络	所指示的网络包含更多的块网络和它们之间没有连接的变量。
F1029	模糊的网络连接顺序	编译器无法找到确定块执行顺序的明确方法。
F1030	临时变量分配错误	内部编译错误。
F1031	网络不一致	所指示的网络没有输入或输出变量。
F1032	连接到电源轨的对象无效	
F1033	无效使用引脚取反（ADR操作符不允许输入取反）	
F1034	无效使用引脚取反（SIZEOF运算符不允许输入取反）	

错误代码	简短的描述	说明
G0001	无效的操作数	对于指定的操作数或功能，操作数的数量不正确。
G0002	变量未定义	该变量尚未在本地或全局上下文中定义。
G0003	标签没有定义	在当前的POU（程序，功能或功能块）中未定义为JMP操作数指示的标签。
G0004	函数块未定义	所指示的实例涉及未在整个项目中定义的功能块。
G0005	对未定义的对象的引用	所指示的实例涉及未在整个项目中定义的对象。
G0006	常量不被承认	所指示的操作不允许使用常量(通常是存储或分配操作)。
G0007	代码缓冲区溢出	用于POU(程序、函数和函数块)的代码的总大小超出了目标系统上可用的空间。
G0008	变量访问无效	不允许访问指定的变量。尝试写入只读变量或读取只写的变量。
G0009	未找到程序	所指示的程序在当前项目中不存在。
G0010	已分配给任务的程序	所指示的程序已分配给目标系统的多个任务。
G0011	不能分配代码缓冲区	PC上没有足够的内存来创建目标系统代码的映像。
G0012	函数没有定义	所指示的功能在当前项目中不存在。
G0013	功能块的循环声明	所指示的函数块直接或通过其他函数调用自身。
G0014	不兼容的外部声明	当前函数块的外部变量声明与它引用的全局变量定义(同名的变量)不匹配。通常是类型不匹配的情况。
G0015	累加器扩展	
G0016	外部变量未找到	外部变量不引用项目的任何全局变量(例如:没有同名的全局变量)。
G0017	程序未分配给任务	所指示的程序没有分配给目标系统中的任务。
G0018	任务未在资源中找到	指示的任务没有在目标系统中定义。
G0019	应用程序中未定义任务	目标系统没有任务定义。目标定义文件(*. tar)丢失或不完整。请联系目标设备供应商。
G0020	远程数据仅允许用于PROGRAM中的加载/存储操作	不允许对函数块进行大内存访问，只允许对程序(错误代码仅对具有近/远数据访问的目标系统有效)。
G0021	无效的处理器类型	目标定义文件（* .TAR）中指示的处理器不正确或编译器不支持。
G0022	带有进程映像变量的函数块不能用于事件任务	
G0023	进程映像变量不能用于事件任务	

错误代码	简短的描述	说明
G0024	累加器未定义	
G0025	无效的指数	
G0026	索引只允许常数	
G0027	对寄存器地址的非法引用	
G0028	可用代码空间少于10%	
G0029	索引超过数组大小	
G0030	假设索引为0，以标量形式访问数组	
G0031	与变量大小不匹配的索引数	
G0032	不支持多维变量	
G0033	无效的数据类型	
G0034	无效的操作类型	
G0035	汇编程序错误	
G0036	由用户中止	
G0037	元素没有定义	
G0038	循环声明结构	
G0039	循环声明类型	
G0040	Typedef未定义	
G0041	超出typedef的尺寸	
G0042	无法分配编译器内部数据	
G0043	代码生成器内部错误	
G0044	不支持REAL类型数据	
G0045	不支持LREAL类型数据	
G0046	不支持LONG类型数据	
G0047	操作未执行	
G0048	无效的操作符	
G0049	无效的操作符值	
G0050	括号不完整	
G0051	数据类型转换	
G0052	待实施	
G0053	无效的索引数据类型	
G0054	无条件否定	
G0055	不允许对布尔值进行操作	
G0056	非布尔操作数的取反	
G0057	需要布尔操作数	
G0058	不允许使用浮点参数	
G0059	操作数的扩展	
G0060	除零	

错误代码	简短的描述	说明
G0061	非法的比较	
G0062	功能块必须实例化	
G0063	不允许字符串操作数	
G0064	不允许在指针上操作	
G0065	目标空间太小，无法存储当前结果	
G0066	不能在多个任务中使用包含带有过程映像的外部变量的功能块	
G0067	无法加载显式常量的地址	
G0068	将实值写入整数变量	
G0069	无法在函数中使用复杂变量。 未实现	
G0070	有符号/无符号不匹配	
G0071	转换数据类型不匹配，可能会丢失数据	
G0072	布尔值到整数的隐式类型转换	
G0073	布尔值到实数的隐式转换	
G0074	整型布尔型的隐式类型转换	
G0075	整型到布尔型的隐式类型转换	
G0076	实数到布尔值的隐式转换	
G0077	实数到整数的隐式类型转换	
G0078	算术运算需要数字运算数	
G0079	按位逻辑操作需要位字符串/整数运算数	
G0080	比较运算需要基本运算(即非用户定义)操作数	
G0081	不能取bit变量的地址	
G0082	将有符号值写入无符号变量	
G0083	将无符号值写入有符号变量	
G0084	从单精度到双精度的隐式转换	
G0085	从双精度到单精度的隐式转换	
G0086	函数参数扩展	

错误代码	简短的描述	说明
G0087	无法强制转换相同类型数据	
G0088	功能参数编号错误	
G0089	未找到目标功能	
G0090	递归类型声明	
G0091	初始值错误。 有符号/无符号不匹配	
G0092	错误的初始值。转换数据类型不匹配，可能会丢失数据	
G0093	字符串将被截断	
G0094	初始化值类型不匹配	
G0095	初始值不正确	
G0096	未找到初始值对象	
G0097	无效的指针赋值	
G0513	无效的操作符	不允许使用此操作符。
G0514	操作符没有实现	当前目标系统不支持此操作符。
G0515	不支持REAL类型数据	正在使用的目标系统不支持浮点数据类型。
G0516	目标空间可能太小，无法存储当前结果	存储/分配操作的可使得目标的数据类型小于累加器之一。操作中数据可能会丢失。例如，如果累加器包含340并且目标操作数为SINT类型，则赋值操作将会造成数据丢失。可以使用适当的类型转换功能（TO_SINT，TO_INT，TO_DINT等）来消除警告消息。
G0517	不支持LONG类型数据	正在使用的目标系统不支持LONG数据类型。
G0518	累加器扩展	存储/分配操作的可使得目标的数据类型大于累加器之一。扩展操作已由编译器自动执行。请使用适当的类型转换功能（TO_SINT，TO_INT，TO_DINT等）来消除此警告消息。
G0519	汇编程序错误	内部编译错误。
G0520	只允许对布尔值求反	用于某些IL运算符（LDN，STN和AND等）的'N'修饰符不能与布尔型以外的其他运算符一起使用。
G0521	布尔类型允许的操作	当累加器的类型不是BOOL时，不能使用IL操作符 (通常为'S'或'R')。
G0522	指令有固定的结果	所指示的操作的结果是固定的(例如，某个数乘以0再与运算FALSE)。
G0523	指令是NOP	所示的运算对累加器的值没有影响（例如，某个数乘以1再与运算TRUE）。

错误代码	简短的描述	说明
G0524	不完整的括号	在指定的代码块中，开括号的数量与闭括号的数量不匹配。
G0525	不允许对布尔值进行的操作	指定的操作不能在布尔值(例如算术操作)上执行。
G0526	无法对LONG类型值执行模数运算	当前目标系统不允许对LONG数据类型进行模块化操作。
G0527	除0	所述除法运算的分母值为0。
G0528	无条件的否定	指定的操作（JMP或RET）具有否定修饰符' N'，而没有条件评估修饰符' C'。 使用JMPCN代替JMPN或RETCN代替RETN。
G0529	初值未定义	内部编译错误。
G0530	无效的初始值	变量的初始值不正确。
G0531	无效的累加器类型	累加器的数据类型不适用于指定的操作（例如带有REAL累加器的MUX运算符）。
G0532	代码生成器内部错误	内部编译错误。
G0533	无效的操作符值	当前使用的值不适用于指定的操作（例如，用于SHL的常数大于32）。
G0534	累加器未定义	执行该操作时不会事先将值加载到累加器中。
G0535	无效的指数	对于数组维度，在指定的表达式中使用的常量索引值太大。请参见数组声明。
G0536	只允许常量索引	编译器不支持将变量用作指示数组的索引。 此错误通常是由布尔（位）数组发出的。
G0537	不允许对布尔常量进行索引	编译器不支持将变量用作指示数组的索引。 此错误通常是由布尔（位）数组发出的。
G0538	程序不允许返回	程序块中不允许使用RET运算符。
G0539	必须实例化功能块	不能使用CAL指令直接调用功能块。 使用前必须实例化它。 必须是具有与功能块相对应的数据类型的变量。
G0540	不允许对REAL类型进行操作	指定的操作无法在REAL数据类型上执行。 这种指令是逻辑和按位运算。
G0541	累加器转换	此警告表示累加器的数据类型已由编译器自动转换。 通常在算术运算中使用的累加器和操作数具有不同的数据类型时执行此运算。
G0542	实际累加器必须重新加载	一些特定目标带有软件浮点运算可能要求在每个存储操作之前必须有一个新的加载操作或算术序列。
G0543	实际累加器未存储	一些特定目标带有软件浮点运算可能要求当浮点堆栈已加载时，应在算术序列末尾将其释放。

错误代码	简短的描述	说明
G0544	不支持LREAL类型数据	编译器不支持LREAL数据类型。
G0769	无效的操作符	不允许使用此操作符。
G0770	操作未实现	当前目标系统不支持此操作符。
G0771	汇编程序错误	内部编译错误。
G0772	不支持LREAL类型数据	编译器不支持LREAL数据类型。
G0773	不支持LONG类型数据	编译器不支持LINT数据类型。
G0774	非布尔类型参数的取反	否定修饰符'N'不能用于非布尔数据类型值的运算中。
G0775	不允许对布尔值进行的操作	指定的操作不能在布尔值(例如算术操作)上执行。
G0776	累加器扩展	存储/分配操作的可使得目标的数据类型大于累加器之一。扩展操作已由编译器自动执行。请使用适当的类型转换功能（TO_SINT，TO_INT，TO_DINT等）来消除此警告消息。
G0777	累加器未定义	执行该操作时不会事先将值加载到累加器中。
G0778	目标空间可能太小，无法存储当前结果	存储/分配操作的可使得目标的数据类型小于累加器之一。操作中数据可能会丢失。例如，如果累加器包含340并且目标操作数为SINT类型，则赋值操作将会造成数据丢失。可以使用适当的类型转换功能（TO_SINT，TO_INT，TO_DINT等）来消除警告消息。
G0779	除0	所述除法运算的分母值为0。
G0780	只允许在REAL类型参数上操作	指定的操作无法在REAL数据类型上执行。 这种指令是逻辑和按位运算。
G0781	非法的比较	在非类型数据之间执行比较操作。
G0782	无条件的否定	指定的操作（JMP或RET）具有否定修饰符' N'，而没有条件评估修饰符' C'。 使用JMPCN代替JMPN或RETCN代替RETN。
G0783	需要布尔类型参数	当累加器的类型不是BOOL时，不能使用IL操作符 (通常为'S'或'R')。
G0784	操作数的扩展	操作数的数据类型已扩展为累加器的数据类型。 然后执行该操作。 只要操作数数据类型小于累加器数据类型，就进行操作数扩展。
G0785	不支持浮点型累加器	累加器具有REAL数据类型，并且不允许进行指示的操作（通常为MUX操作）。
G0786	不支持布尔型累加器	累加器具有布尔数据类型，并且不允许进行指示的操作（例如MUX运算符）。

错误代码	简短的描述	说明
G0787	无符号类型和有符号类型数据的比较	比较操作是使用有符号和无符号数据类型的操作符来执行的。可能会出现不希望的或无法控制的结果。
G0788	非法的转换	内部编译错误。
G0789	转换可能导致数据丢失或损坏	错误代码未使用。
G0790	非法对REAL类型参数取反	错误代码未使用。
G0791	将REAL类型数据写入INT型变量或参数	传递给功能的参数为REAL类型，而不是函数输入变量定义所需的INT数据类型。
G0792	将INT类型数据写入REAL型变量或参数	传递给功能的参数为INT类型，而不是函数输入变量定义所需的REAL数据类型。
G0793	将有符号值写入无符号变量或参数	赋值操作是对无符号数据类型变量执行的，但是累加器数据类型具有带符号数据类型。 可能会出现不希望的结果。
G0794	将无符号值写入有符号变量或参数	赋值操作是对无符号数据类型变量执行的，但是累加器数据类型具有带符号数据类型。 可能会出现不希望的结果。
G0795	不完整的括号	在指定的代码块中，开括号的数量与闭括号的数量不匹配。
G0796	扩展参数时出错	内部编译错误。
G0797	无效的指数	对于数组维度，在指定的表达式中使用的常量索引值太大。请参见数组声明。
G0798	使用布尔类型数据索引访问数组的元素	指示的数组访问不正确，因为使用的索引变量是布尔数据类型。
G0799	程序不允许返回	不允许在程序块中使用RET操作符。
G0800	需要布尔类型累加器	所指示的SEL操作符要求累加器具有布尔数据类型。
G0801	操作符类型不匹配	由MUX和SEL运算符执行的选择应在具有同类数据类型的元素之间完成。
G0802	必须进行功能块块实例化	不能使用CAL指令直接调用功能块。 使用前必须实例化它。必须是具有与功能块相对应的数据类型的变量。
G1537	使用布尔类型数据索引访问数组的元素	
G1538	不支持布尔类型累加器	
G1539	不支持浮点类型累加器	
G1540	扩展操作数时出错	
G1541	将有符号值写入无符号变量	
G1542	将无符号值写入有符号变量	
G1543	将REAL值写入INT变量	

错误代码	简短的描述	说明
G1544	将INT值写入REAL变量	
G1545	将字符串转换为数字	
G1546	将数字转换为字符串	
G1547	FPU栈满	
G1548	FPU栈空	
G1549	FPU栈大小错误	
G1550	通过功能非法访问变量	
G1551	对可通过功能访问的变量地址的非法引用	
G1552	无效的功能访问	
G1553	具有相同句柄的两个变量	
G1554	通过功能访问的变量的索引无效	
G1555	无效指令与非空FPU堆栈	
G1556	类型为string的功能返回需要存储到变量中	
G8193	未知的数据类型	
G8194	类型定义超出数组尺寸	
G8195	数据类型的循环定义	
G8196	不支持双指针	
G8197	没有列举的元素	
G8199	无效或未定义的初始化常数	
G10241	变量的初始值设定项太多	
G10242	变量的初始值设定项太少	
G10243	没有初始化值的常量	
P2048	无法打开参数文件	无法打开参数的源文件（扩展名为PPC），因为该文件丢失或被电脑的文件系统锁定。
P2049	没有创建符号表文件	由于磁盘写保护或磁盘空间不足，无法写入符号分配文件（扩展名为SYM）。
P2050	不能创建参数文件	由于磁盘写保护或磁盘空间不足，无法写入参数文件（扩展名为PAR）。
P2051	无法创建目录	无法创建新项目的目录。当存在磁盘写保护或为该项目指示的新目录在现有磁盘目录的深一层以上时，就会出现此问题。 编译器仅从现有目录创建一个新的目录级别（一个带有项目名称的目录级别）。

错误代码	简短的描述	说明
P2052	无法打开源项目	用于创建新项目的源项目不存在、不完整或被文件系统锁定。
P2053	保存项目错误	由于磁盘写保护、目标目录不存在或文件系统锁定，因此无法保存新项目。
P2054	通用文件错误	在文件操作期间发生了一个非特定的错误。
P2055	无法复制文件	由于缺少源文件、磁盘写保护或目标文件已存在且受保护，无法复制指定的文件。
P2056	不能保存文件	由于磁盘写保护或目标文件已存在且受保护，无法保存指定的文件。
P2057	对象已经存在于项目中	指定的对象(变量、函数、函数块或程序)包含在最后加载的库中，但是在当前项目中已经有另一个具有相同名称的对象。
P2058	无法打开库文件	指定的库文件不存在或由于文件系统锁定而无法打开。
P2059	未创建列表文件	
P2060	无法创建PLC应用程序二进制文件	
P2061	无法打开模板项目	
P2062	不支持处理器	
P2063	可用代码空间少于10%	
P2064	可用数据空间不到10%	
P2065	可用保持型数据空间不到10%	
P2066	空闲位数据少于10%	
P2067	资源中找不到任务	
P2068	资源中未定义任务	
P2069	项目采用旧的PPJ格式。 它将以实际的PPJX格式保存	
P2070	无法打开辅助源文件	
P2071	不能读取文件	
P2072	应用程序名称超过10个字符:只有前10个字符将被下载到目标中	
P2073	可下载的源代码文件不受密码保护	
P2074	未创建可下载的PLC应用程序二进制文件	
P2075	Ext/aux数据少于10%	
P2076	该库的项目专用副本丢失了，并已替换为该库的新副本（来自原始路径）	

错误代码	简短的描述	说明
P2077	不能装载库!此库的项目私有副本丢失，且到库的原始路径无效:库已被删除	
P2078	未创建PLC变量导出文件	
P2079	没有创建调试符号打包文件(用于下载到目标设备)	
P2080	没有创建源代码打包文件(用于下载到目标设备)	
P2081	无效的任务定义	
P2083	无效或不连贯的任务周期	
P2084	已损坏的库链接	
S1281	一般的ST语法错误	
S1282	嵌套的表达式太多	
S1283	缺少循环循环退出语句	
S1284	缺少END_IF	
S1285	无效的ST声明	
S1286	无效的分配	
S1287	缺失;	
S1288	无效的表达式	
S1289	无效的表达式或缺少DO	
S1290	缺少END_WHILE	
S1291	缺少END_FOR	
S1292	缺少END_REPEAT	
S1293	无效的表达式或缺少THEN	
S1294	无效的表达式或缺少TO	
S1295	无效的表达式或缺少BY	
S1296	无效语句或缺少UNTIL	
S1297	无效的赋值，预期为:=	
S1298	无效的地址表达式	
S1299	无效的表达式大小	
S1300	函数返回值被忽略	
S1301	无效的参数传递	
S1302	函数参数未定义	
S1303	无用的表达式	
S1304	不完整的括号	
S1305	未知功能	
S1306	无效的功能参数规范	
S1307	功能参数不存在	

错误代码	简短的描述	说明
S1308	不允许多重赋值(按照IEC 61131-3标准)	
S1309	ST预处理器缓冲区溢出	
S1310	非功能块实例的功能块调用	
S1311	缺少END_WAITING	
S1312	语法错误	
S1537	通用SFC错误	
S1538	初始步骤缺失	
S1539	输出连接丢失	
S1540	输出引脚必须连接到一个转换	
S1541	转换的每个输出引脚都必须连接到一个步/跳转块	
S1542	预计需要一个转换	
S1543	预计需要一个步或者跳转	
S1544	找不到关联程序代码	
S1545	找不到条件码	
S1546	未知类型的转换	
S1547	无效的目标位置	
S1548	重复动作。同一SFC的动作不能在一个以上的步中使用	
T8193	通讯超时	与目标系统的通信失败，因为系统本身没有应答。更常见的原因是电缆连接错误、通信设置中的目标地址无效、通信参数设置无效(如波特率)、目标系统故障。
T8194	不兼容的目标版本	错误代码未使用。
T8195	无效的代码文件	目标系统镜像文件(带有IMG扩展名)无效或损坏。尝试使用“通讯上传镜像文件”菜单选项上传并创建镜像文件的新版本。
T8196	无效数据块索引	镜像文件(带有IMG扩展名)包含一个数据块，该数据块的索引大于目标系统支持的最大索引。尝试使用“通讯上传镜像文件”菜单选项上传并创建镜像文件的新版本。如果问题持续存在，请与目标系统供应商联系。
T8197	目标信息地址无效	内部编译错误。
T8198	Flash擦除失败	目标系统无法完成Flash擦除程序。联系目标系统供应商了解详细信息。
T8199	代码写入失败	目标系统无法完成Flash编程程序。联系目标系统供应商了解详细信息。

错误代码	简短的描述	说明
T8200	通信设备不可用	编译器尝试与目标系统通信，但是通信通道不可用。 如果问题仍然存在，并且还有其他应用程序与目标系统通信，请在其他应用程序上停用通信，然后重试。
T8201	无效的功能索引	内部编译错误。
T8202	无效的数据库信息地址	目标系统的参数数据库存储区的地址不正确或无效。尝试使用“通讯上传镜像文件”菜单选项上传并创建镜像文件的新版本。
T8203	无效的目标信息	
T8204	需要重建工程	
T8205	无效的任务	
T8206	应用层通信协议错误:PLC运行时无法解析接收到的命令	
T8207	未实现完成	
T8209	目标设备上没有存放源文件的空间	
T8210	从目标设备上传源代码时出错	
T8211	目标上没有调试符号的空间	
T8212	内存读取错误	
T8213	内存写入错误	
T8214	目标设备上没有足够的空间用于存储PLC应用程序二进制文件	