1. CoDeSys简要介绍 1-1
1.1 CoDeSys是什么 1-1
1.2 CoDeSys功能一览 1-1
2. CoDeSys组成 2-1
2.1 工程组件 2-1
2.1.1 程序 2-5
2.2 语言 2-8
2.2.1 指令表 2-8
2.2.2 结构化文本 2-10
2.2.3 顺序功能图（SFC） 2-15
2.2.4 功能模块图 2-20
2.2.5 连续功能图表编辑器 2-20
2.2.6 梯形图 2-20
2.3 调试、联机功能 2-22
2.4 标准化 2-23
3. 我们来编写一个小程序 3-1
3.1 控制一个交通灯信号单元 3-1
3.2 可视化交通信号单元 3-10
4. 各个单独的组件 4-1
4.1 主窗口 4-1
4.2 工程选项 4-3
4.3 管理工程 4-20
4.3.1 ’工程’’数据库连接’ 4-46
4.4 管理工程中的对象 4-52
4.5 编辑功能 4-59
4.6 联机功能 4-65
4.7 设置窗口 4-78
4.8 帮助 4-79
5. CoDeSys中的编辑器 5-1
5.1 关于所有的编辑器 5-1
5.2 声明编辑器 5-2
5.3 声明编辑器中的预处理pragma指令 5-11
5.4 文本编辑器 5-18
5.4.1 指令表编辑器 5-21
5.4.2 结构化文本编辑器 5-22
5.5 图形化编辑器 5-22
5.5.1 功能模块图编辑器 5-26
5.5.2 梯形图 5-30
5.5.3 顺序功能图表编辑器 5-36
5.5.4 连续功能图表编辑器 5-44
6. 资源 6-1
6.1 全局变量、变量配置、文件框架 6-1
6.1.1 全局变量 6-1
6.1.2 变量配置 6-5

6.1.3 文档框架 6-6
6.2 报警配置 6-7
6.3 库管理器 6-15
6.4 日志记录 6-16
6.5 任务配置 6-18
6.6 监控和配方管理器 6-24
6.7 工作空间 6-26
6.8 对象系统设置 6-26
6.9 PLC 配置 6-27
6.9.1 综述 6-27
6.9.2 PLC配置中的工作 6-28
6.9.3 PLC配置中的一般设置 6-29
6.9.4 定制特定的参数的对话框 6-30
6.9.5 I/O模块配置 6-30
6.9.6 通道配置 6-33
6.9.7 Profibus模块的配置 6-33
6.9.8 CAN 模块的配置 6-41
6.9.9 Can驱动器的配置 6-45
6.9.10 在线模式中的PLC配置 6-48
6.9.11 来自PLC的硬件扫描/状态/诊断信息 6-49
6.10 采样追踪 6-49
6.10.1 综述和配置 6-49
6.10.2 采样追踪的显示 6-50
6.10.3 保存采样追踪 6-52
6.10.4 外部采样追踪配置 6-52
6.11 参数管理器 6-53
6.11.1 参数管理器的编辑 6-54
6.11.2 参数列表的导出、导入 6-58
6.11.3 在线模式下的参数管理 6-59
6.12 PLC浏览器 6-59
6.13 工具 6-62
7. ENI 7-1
8. DDE接口 8-1
8.1 CoDeSys程序设计系统的DDE接口 8-1
8.2 DDE网关服务器的DDE信息 8-2
9. CoDeSys的许可证管理器 9-1
10. 附录 10-1
A:IEC操作符和额外的标准扩展功能块 10-1
10.1 算术操作符 10-1
10.2 位串操作符 10-4
10.3 移位操作符 10-6
10.4 选择操作符 10-8
10.5 比较操作符 10-11
10.6 地址操作符 10-13

10.7 调用操作符 10-14
10.8 类型变化 10-14
10.9 数字操作符 10-20
10.10 初始化操作符 10-23
附录B：CoDeSys中的操作数 10-24
10.11 常量 10-24
10.12 变量 10-26
10.13 地址 10-28
10.14 功能 10-29
附录C：CoDeSys中的数据类型 10-30
10.15 标准数据类型 10-30
10.16 已定义的数据类型 10-32
附录D：CoDeSys程序库 10-37
10.17 standard.lab标准库 10-37
10.17.1 字符串功能 10-37
10.17.2 双稳功能程序 10-41
10.17.3 触发器 10-43
10.17.4 记数器 10-44
10.17.5 定时器 10-46
10.18 Util.lib库 10-48
10.18.1 BCD 转换 10-49
10.18.2 位/字节功能 10-49
10.18.3 10.18.3 数学辅助功能 10-50
10.18.4 控制器 10-51
10.18.5 信号生成 10-53
10.18.6 功能操作器 10-54
10.18.7 模拟值的处理 10-56
10.19 AnalyzationNew.lib库 10-57
10.20 CoDeSys系统程序库 10-57
附录E：操作符及程序库模块总结 10-58
10.21 CoDeSys中的操作符: 10-58
10.22 Standard.lib库的元素: 10-61
10.23 Util.lib库的元素: 10-62
附录F：命令行/命令文件 10-62
10.24 命令行相关命令 10-62
10.25 命令文件（cmdfile）命令 10-63
附录G：导入Siemens产品数据 10-70
10.26 导入一个SEQ符号文件 10-70
10.27 将S5 转换成IEC 61131-3 10-71
10.28 导入一个S5 工程文件 10-74
附录I 应用键盘 10-74
10.29 应用键盘 10-74
10.30 组合键 10-75
CoDeSys中的关键字 10-78

CoDeSys 文件 10-81
附录K 关于编译错误和警告 10-83
10.31 警告 10-83
10.32 错误 10-87
索引： I

1. CoDeSys简要介绍

1.1 CoDeSys是什么

CoDeSys是可编程逻辑控制PLC的完整开发环境（CoDeSys是Controlled Developement System的缩写），在PLC程序员编程时,CoDeSys为强大的IEC语言提供了一个简单的方法，系统的 编辑器和调试器的功能是建立在高级编程语言的基础上（如Visual C++）。

1.2 CoDeSys功能一览

如何构建一个工程?

一个 工程放在以工程命名的文件中，新工程中创建的第一个程序组织单元 POU自动命名为 PLC_PRG，程序从这里开始执行（相当于C程序中的主函数），从这一点能够访问其它的POUs（程序, 功能块 和 功能）。

一旦定义了一个任务配置，就不必创建程序PLC_PRG 了。在 任务配置章节将讲到更多的内容。

在工程中有不同的对象：POUs、数据类型、可视化和资源。

对象管理器中包含了工程中的所有对象。

如何建立一个工程?

首先，为了保证在工程文件中使用的地址的正确性，应该配置 PLC。

然后创建解决问题所需要的 POUs。

当程序编写完成时，可以编译这个工程并去除所有的错误。此刻可以用你所喜欢的编程语言来编写这些POUs。

如何测试自己的工程?

一旦排除了所有的错误，激活 **仿真**模式，登录入仿真的PLC并在PLC中“加载”工程文件，此时处于联机模式。

现在打开一个 PLC 配置的窗口测试工程的时序正确性，手动为此输入输入变量，观察输出变量是不是所期望的。你也可以观察 POU 中的局部变量的序列值，在监视和接收管理器中可以配置你希望检查的值的数据记录。

调试

你可以在程序出错的地方设置 断点。当程序运行后停在断点处时，你可以及时检查在这个点处所有变量的值，通过一步一步(单步) 执行,你可以检查程序的逻辑正确性。

附加的联机功能

更多的调试功能：

你可以设置程序变量并输入输出某些值。

你可以通过流程控制来检查那些程序行已完成运行。

日志文件按照时间的顺序记录了联机模式下的操作以及用户行为和内部进程的情况。

如果在目标设置中激发了采样追踪，那么它允许你在一个较长的过程中来追踪和显示变量值的真实变化过程。

PLC 浏览器是目标系统的一个特殊功能，它能够用来向 PLC 请求某些信息。

当 工程完成了建立和测试后，它也能够装载到硬件中并进行测试。和 仿真模式 式下有相同的联机功能。

附加的CoDeSys功能

整个 工程可以在任何时候 文档化或 导出到一个文本文件中。

为了通讯的目的，CoDeSys 有一个符号接口和一个动态数据交换（DDE）接口。网关服务器和 OPC 服务器和动态数据交换服务器是 CoDeSys 的标准安装软件包的组件。

使用恰当的目标设置能够把相同的 CoDeSys 工程加载到不同的目标系统中，可以通过目标文件来加载这些目标设置。

通过当前的目标设置来激活网络全局变量和参数管理器。可以在控制器网络中交换数据。

通过 ENI 服务器，用工程接口能够访问任何我们所期望的源代码管理程序。ENI 服务器是个独立运行的程序，CoDeSys 的程序组织单元和编译文件可以以文档方式存于数据库中，它们能够被 ENI 服务器的客户端访问到。这允许在一个工程文件的工作过程中进行多用户操作，它为 CoDeSys 和其它工具提供了一个公用数据缓冲池而且它使版本管理成为可能。

工具：这个功能性也是与目标有关的，它允许启动 CoDeSys 工程中的特殊目标执行文件，除此之外还可以定义要被装载到控制器中的这些文件，它与外部工具的联系可以在目标文件中预定义或者插入到工程资源树中。

CoDeSys 的可视化可以处理象网页可视化或目标可视化，这样可以通过因特网或者 PLC 监视器的运行来展示可视化。

2. CoDeSys组成

2.1 工程组件

工程

一个工程包含了 PLC 程序中的所有对象，工程存储在以工程命名的文件中，工程中包含下列对象：POU，数据类型，可视化，资源和库

POU（程序组织单元）

功能、功能块、程序是程序组织单元，它们能够通过动作来增补，每一个程序组织单元都包含一个定义部分和主体部分，主体部分可以用 IEC 的语言来编写，这些语言包括指令列表，结构化文本，顺序功能图，功能模块图，梯形图或连续功能图表。

CoDeSys支持所有IEC标准的POU，如果你想在你的 工程文件中使用这些POU，你必须在你的工程文件中包含标准库文件standard.lib。

POU 可以调用其它的 POU，但递归调用是不允许的。

功能

一个功能是一个 POU，它正确地产生一个数据元素（可以包含若干元素，比如，字段或者结构体）在处理过程中，可以用文本化语言中的表达式中的一个操作数来调用它。

在声明一个功能的时候，一定要给它一个类型，这就是说，在功能名后面加上一个冒号然后跟一个数据类型。

一个正确的功能声明可以参考下面的例子：

```
FUNCTION Fct: INT
```

另外，必须分配给功能一个结果，即把功能名作为一个输出变量

功能的声明从关键字FUNCTION开始。推荐的声明方式。

下例是在指令列表（IL）中的一个功能，它声明了三个输入变量：

前两个变量的相乘然后除以第三个变量。功能返回此操作的结果。

声明部分：

```
FUNCTION Fct: INT
VAR_INPUT
PAR1:INT;
PAR2:INT;
PAR3:INT;
END_VAR
```

程序部分：

```
LD PAR1
MUL PAR2
DIV PAR3
ST Fct
```

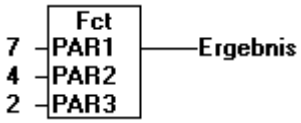
在结构文本中功能的调用可以作为表达式中的一个操作数。

功能不会有任何内部条件，这就是说，调用带有相同的输入变量功能将会返回相同的输出结果。

下面是调用功能的例子：

```
in IL:
```

```
LD 7
Fct 2,4
ST  Ergebnis
in ST:
Ergebnis := Fct(7, 2, 4);
in FUP:
```



功能不会保持内部状态，对于不包含全局变量和地址的功能，每次在它被调用的过程中，给它传递相同的输入变量，它将返回相同的值 。

注意：如果一个局部变量在一个功能中被声明为 RETAIN，这也没有任何影响，为此变量将不会写到保留区。

如果你定义了一个功能名为 CheckBounds，你可以用它来检验工程中的溢出的范围，定义地功能名是它的仅有的标识符，更详细的内容请参考 10.1 章节中关于数学运算符中的 DIV。

如果在工程中你定义了 CheckDivByte, CheckDivWord, CheckDivDWord 和 CheckDivReal 功能, 如果你用了除法算式，你可以用它们来检查除数的值，可以避免分母为零。

如果你定义了功能名CheckRangeSigned 和 CheckRangeUnsigned,超出变量定义范围的数据类型被截取。

所有这些检验功能名作为特定的用途保留下来。更多的知识请参考 Defined Datatype，Array。

在 SFC 中，一个功能的调用只能发生在一个单步操作或变换之内。

功能块

一个功能块是一个程序组织单元，在程序中提供一个或多个值，与 功能相反，一个功能块没有返回值。

功能块的声明用关键字FUNCTION_BLOCK开始.推荐的声明方式。

可以创建功能块的复制或 实例。

调用功能块是通过功能块实例实现的。

下面是一个在指令列表中功能块的例子，在指令列表中功能块中包含两个输入变量和两个输出变量，一个输出的是两个输入变量的乘积，另一个是两个输入变量的是否相等的比较。

声明部分：

```
FUNCTION_BLOCK FUB
VAR_INPUT
PAR1:INT;
PAR2:INT;
END_VAR
VAR_OUTPUT
MELERG:INT;
VERGL:BOOL;
END_VAR
```

在 IL 的执行部分：

```
LD PAR1
MUL PAR2
ST MULERG
LD PAR1
EQ PAR2
ST VERGL
```

功能块实例

可以创建 功能块的复制或实例。

每一个实例都有它自己的标识符，并且一个数据结构体中包含它的输入输出和内部变量，实例可以象变量一样被声明为局部变量或全局变量，然而功能块的名称表示标识符的类型。

推荐的声明方式。

例如名为 INSTANCE 功能块 PUB 实例：

fubInstance ： FUB;

功能块通常是通过上述的实例来调用的。

从此功能块实例的外部只能访问它的输入输出变量，不能访问它的内部变量调用。

下面是一个访问输入变量的例子

The function block FB has an input variable in1 of the type INT.

```
PROGRAM prog
VAR
inst1:fb;
END_VAR
    LD 17
ST  inst1.in1
CAL inst1
    END_PROGRAM
```

功能块 FB 有一个整型的输入变量 in1

功能块和程序的声明部分能够包含实例的声明，实例的声明不能包含在 功能之中。

访问功能块实例仅限于它被声明的 POU中，除非它被声明为全局变量。

注意：在一个功能块处理完后的所有值将保存下来，直到下一个功能模块调用。所以，功能块调用相同的输入值往往不返回相同的输出值。

注意: 如果存在至少一个功能块变量是保留变量，整个实例将被存储在保留区。

调用一个功能块

通过建立一个 功能块的实例并且用下面的语法来规定要求的变量，可以从其它的POU访问这个功能块的输入和输出变量。

〈实例名〉.〈变量名〉

在调用时为变量赋值：

如果你喜欢在调用功能块的时候再设置输入或输出变量，你可以用指令列表和结构化文本语言。通过在功能块实例名后面的括号中为变量赋值来进行（对输入变量的赋值就象在声明位置的 变量初始化一样，使用“：=”来分配变量的值。）

如果在 ST 或 IL POU 的执行窗口中使用选项 With arguments，并通过输入帮助（〈F2〉）来插入实例，它将根据这个句式显示所有的变量，但不必为这些变量赋值。

例如：

FBINST 是一个功能块类型的局部变量，它包含了输入变量 xx 和输出变量 yy。当 FBINST 是通过输入帮助插入到了 ST 程序中，将显示如下的调用：FBINST1(xx:= , yy=〉)。

在调用输入输出变量时：

请注意:功能块的输入输出变量作为指针来处理。因此在调用一个功能块时，常量是不能赋予 VAR_IN_OUT 并且从外部没有读和写的权限。

例如：

在 ST 模式下调用 fubo 功能块的一个 VAR_IN_OUT 变量 inout1：

```
VAR
```

```
     fuboinst:fubo;
     iVar1:int;
    END_VAR
    iVar1:=2;
    fuboinst(iInout1:=var1);
```

下列在这种情况下下列语句示不允许的

```
    fuboinst(iInout1:=2);  或 fuboinst.iInout1:=2;
```

下面举例说明调用功能块 FUB：

关于功能块 FUB，参照上述‘功能块’部分

```
    Deklarationsteil:
    FUNCTION_BLOCK FUB
    VAR_INPUT
    PAR1:INT;
PAR2:INT;
    END_VAR
    VAR_OUTPUT
    MELERG:INT;
VERGL:BOOL;
    END_VAR
    Implementationsteil in AWL:
    LD PAR1
MUL PAR2
ST MULERG
    LD PAR1
EQ PAR2
ST VERGL
```

乘法运算的结果被存储在变量 ERG 中，比较的结果存储在 QUAD 中，FUB 的实例被声明为 INSTANCE

下面是功能块的实例在指令列表中调用的例子

IL 中调用 FUB:

声明部分:

```
    PROGRAM AWLaufruf
    VAR
    QUAD :     BOOL;
INSTANZ :   FUB;
ERG:        INT:=0;
    END_VAR
```

执行部分:

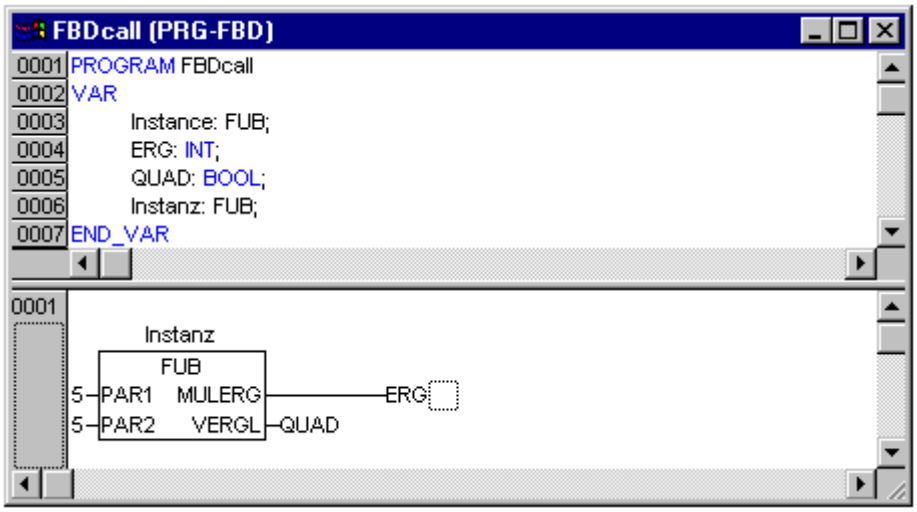
```
    CAL    INSTANZ(PAR1:=5;PAR2:=5);
    LD     INSTANZ.VERGL
ST    QUAD
    LD     INSTANZ.MULERG
ST    ERG
```

下面是功能块的实例在结构化文本中调用的例子（声明部分与指令列表部分相同）

```
    PROGRAM STaufruf
```

INSTANZ(PAR1:=5;PAR2:=5);　　bzw.　INSTANZ;
QUAD:=INSTANZ.VERGL;
ERG:=INSTANZ.MULERG;

下面是功能块的实例在功能块图中调用的例子（声明部分与指令列表部分相同）

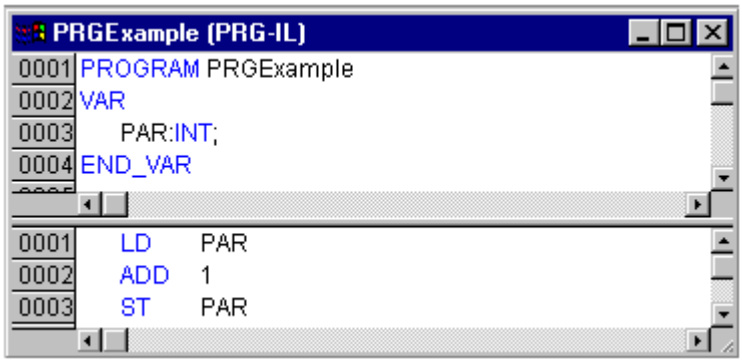


在顺序功能图中功能块的调用只能一步一步进行。

2.1.1 程序

一个程序是一个 POU，它在操作过程中返回几个值，程序在工程文件中是全局的。程序的所有值将保留到下一个程序开始运行。

下面是 程序的一个例子：



程序可以被调用，在一个 功能中调用程序是不允许的，同时也不存在程序的实例。

如果一个 POU 调用一个程序，并且如果程序的值发生了变化，那么这些变化将保留到下一次程序的调用时。即使是其它的 POU 内部调用了它。

这和调用功能块不同，那里只有给定的功能块实例中的特定的值才会变化。

当相同的实例被调用时，这些变化才会发挥重要的作用。推荐的声明方式。

程序的声明开始于关键字 PROGRAM 结束于 END_PROGRAM

如果你喜欢在调用程序的时候再设置输入或输出变量，你可以用文本语言如指令列表和结构化文本来做这些。在功能块的实例名后面的括号中为变量赋值（对输入变量的赋值就象在声明位置的 变量初始化一样，使用“：=”来分配变量的值。）

在结构化文本或者指令列表程序组织单元的执行窗口中，如果程序是通过带 With arguments 选项的输入帮助插入的，根据这个语法，程序和它的所有变量将自动的显示出来。但是你不必给这些变量赋值。

下面是程序调用的例子：

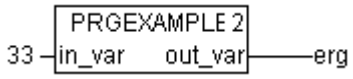
IL 中:

```
CAL PRGexample2
LD PRGexample2.out_var
ST  ERG
CAL PRGexample2(in_var:=33, out_var=>erg )
```

ST 中:

```
PRGexample2;
Erg := PRGexample2.out_var;
PRGexample2(in_var:=33, out_var=>erg );
```

FBD 中:



PLC_PRG调用顺序的例子：

请参照本页之上的程序 PRGexample

```
LD 0
ST  PRGexample.PAR (*Default setting for PAR is 0*)
CAL IL call (*ERG in IL call results in 1*)
CAL ST call (*ERG in ST call results in 2*)
CAL FBD call (*ERG in FBD call results in 3*)
```

如果程序 PRGexample 中的变量 PAR 在初始化时被主程序赋予 0 值，随后用上面命名的程序调用一个接一个的调用。那么程序中 ERG 的结果会有 1，2 和 3，如果改变了调用的顺序，那么给出的结果变量的值也会相应的跟着变化。

PLC_PRG

PLC_PRG 是一个特殊的预定义的 POU，每一个工程文件中必须包含一个这样的特殊的程序。实际上这个 POU 在每个控制循环中只调用一次。

在一个新工程文件创建之后，将首次使用“工程”“添加对象”命令，在 POU 的对话框的缺省项目是一个名为 PLC_PRG 的程序类型的 POU。你不能更改这些默认的设置。

如果定义了 任务，那么工程中可以不包含PLC_PRG，因为在这种情况下，程序的时序依赖于任务的分配。

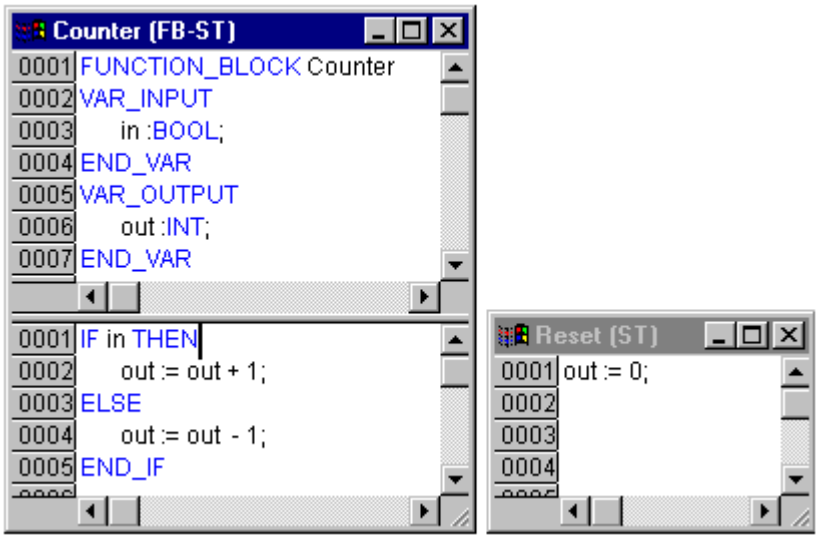
注意：不要删除或者重命名程序组织单元PLC_PRG(假如你没有使用 任务配置)PLC_PRG是一个单任务程序中的主程序。

动作

动作能够被定义并分配给 功能块和 程序，动作代表了一个另外的执行，它可以用用其它的语言进行创建，每一个动作都有一个名称。

每一个动作都是和功能块或者程序中的数据一起工作的，动作使用和标准执行相同的输入/输出变量和局部变量。

下面是一个功能块的动作的例子：



在上面的例子中，调用一个功能块计数器增加或减少输出变量 out 的值，它依赖输入变量 in 的值，调用功能块的复位来设置输出变量为零，相同的变量 out 写到了两个例子中。

调用一个动作：

调用一个动作是通过<程序名>.<动作名>或<功能块实例名>.<动作名>，注意在FBD中的注释（看下例）！如果需要在自己的模块中调用这个动作，只需要在文本编辑器和图形界面中使用动作的名称来构成功能块的调用而不必需要 实例的信息。

下面是一个从其他的程序组织单元调用上述动作的例子：

声明:

```
PROGRAM PLC_PRG
VAR
 Inst : Counter;
END_VAR
```

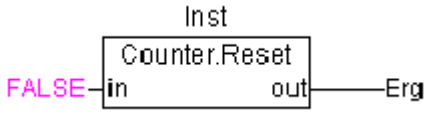
采用 IL 编程方式,用另一个 POU 调用 'Reset':

```
CAL Inst.Reset(In := FALSE)
LD Inst.out
ST  ERG
```

采用 ST 编程方式,用另一个 POU 调用 'Reset':

```
Inst.Reset(In := FALSE);
Erg := Inst.out;
```

采用 FBD 编程方式,用另一个 POU 调用 'Reset':



注意：动作在顺序功能图中发挥重要的作用，参照 顺序功能图，IEC标准只认可顺序功能图中的动作，除此之外都不认可。

资源

你需要用资源来配置和组织你的工程文件和追踪变量的值

- 工程文件或网络中使用的 全局变量。
- 添加库文件到工程文件中的 库管理器
- 记录在线期间工作的 日志文件
- 在工程中为报警处理进行 报警配置
- 配置可编程控制器的 PLC配置资源

•通过任务来引导创建程序的 任务配置
•显示变量值和添加默认变量值的 监控和配方管理器
•选择 目标设置和必要时的确定的目标系统的最终配置
•作为工程选项的 工作空间
根据在 CoDeSys 中作出的目标系统和目标设置，在你的工程中也要用的到下列资源。
•用于变量图形显示的采样追踪
•用于在同一个网络中与其它控制器交换数据的变量管理器
•作为控制监视的 PLC 浏览器
•工具箱?可用性依赖对象系统?用于在 CoDeSys 内部调用它外部的工具程序

库

你可以在你的工程文件中包含一系列的库文件，你可以象使用用户定义的变量一样使用库文件的程序组织单元，数据类型，和全局变量，库文件中的标准库文件和 util.lib 是标准的部件并且你经常使用它。

更多的知识请参照 “库管理器”

数据类型

参照标准的数据类型，用户可以定义自己的数据类型，可以建立结构体枚举类型和引用。

详见 “数据类型”

可视化

CoDeSys 提供了可视化，因此你可以显示工程的变量，通过可视化的帮助你可以在离线的情况下绘制几何元素，在联机模式下能够响应特定变量的值从而改变他们的形式，颜色和文本输出。

可视化的界面可以用作带 CoDeSys 的 HMI 的 PLC 纯操作接口，或者作为一个网页可视化或通过因特网与 PLC 直接连接的对象可视化。

详见 “CoDeSys可视化”用户手册

2.2 语言

CoDeSys 支持 IEC _61131 所描述的所有语言
文本化的语言：
•指令表
•结构文本
图形化的语言：
•顺序功能流程图
•功能模块图
•梯形图
还可采用基于功能模块图的 连续功能编辑器（CFC）.

2.2.1 指令表

指令表中包含一系列的指令，依赖于操作的类型,每一条指令在一个新行开始并且包含运算符号和一个或多个用逗号隔开的操作数。

在一个指令前面，还可以有一个标号，后缀一个冒号。

注释部分在一行的最后，指令与指令之间可以插入空行。

例如：

```
LD 17
ST lint (* Kommentar *)
```

```
GE 5
JMPC next
LD idword
EQ istruct.sdword
STN test
next:
```

请参考：

指令表中的修饰符和操作符

指令表中的修饰符和操作符

在指令列表中将用到下面的操作符和修饰符：

修饰符：

•C	与操作符 JMP, CAL, RET 连用:	当前面的表达式处理的结果为 TRUE 时，才执行此指令。
•N	与操作符 JMPC, CALC, RETC 连用:	当前面的表达式处理的结果为 FALSE 时，才执行此指令。
•N	用于其它情况:	取操作数的反（不包括累加器）。

下面是操作符和它们可能的修饰符以及相关的意思：

操作符	修饰符	含意
LD	N	使当前的值等于操作数
ST	N	在操作数的位置保存当前值
S		当当前的值为 TRUE 时，把布尔型操作数置为 TRUE
R		当当前的值为 TRUE 时，把布尔型操作数置为 FALSE
AND	N,(	位逻辑运算符号“与”
OR	N,(	位逻辑运算符号“或”
XOR	N,(	位逻辑运算符号“异或”
ADD	(	加法
SUB	(	减法
MUL	(	乘法
DIV	(	除法
GT	(	>
GE	(	>=
EQ	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	CN	跳转到标号
CAL	CN	调用程序功能块
RET	CN	离开 POU 并返回到调用的地方
)		执行延时操作

单击这里可以得到所有 IEC操作符的列表。

使用修饰符编写的程序的例子：

LD	TRUE	(*把 TRUE 加载到累加器中*)
ANDN	BOOL1	(*执行 AND 和 BOOL1 变量的反之“与”*)
JMPC	mark	(*当上面的结果为 TRUE 时，跳转到标号“mark”处*)
LDN	BOOL2	(*保存 BOOL2 的反 *)
ST	ERG	(*把 BOOL2 保存在 ERG*)
标号:		
LD	BOOL2	(*保存 BOOL2 的值 *)
ST	ERG	(*把 BOOL2 保存在 ERG*)

在 IL 中也可以在操作之后放一个圆括号。圆括号内的值被认为是一个操作数。

例如：

```
LD  2
MUL 2
ADD 3
Erg
```

这里 Erg 的值为 7，但是如果加一个圆括号：

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

Erg 的结果是 10，当到达")"时操作 MUL 才开始计算；此时对操作数计算 MUL 5。

2.2.2 结构化文本

结构化文本中包含一系列的指令，这些用高级语言编写的指令能够被执行（例如 IF……THEN……ELSE）或者在循环（WHILE..DO）。

例如：

```
IF value < 7 THEN
WHILE value < 8 DO
value:=value+1;
END_WHILE;
END_IF;
```

参照：

表达式
表达式的计算
对操作数赋值
在ST中调用功能块
RETURN 指令
IF 指令
CASE 指令
FOR 循环
WHILE 循环

REPEAT 循环
EXIT 指令

表达式

表达式是一个在运算后返回一个值的结构。
表达式由运算符和操作数组成，操作数可以是常量、变量、功能调用或其它表达式。

表达式的计算

依照一定的规则来处理运算符号可以计算出表达式的值，约束力最高的运算符首先参加运算，然后是约束力稍高的运算符，直到所有的运算符都被处理为止。

运算符号“=”号的处理是从左到右的顺序。

下面是结构文本中运算符号约束力的级别排列

操作	符合	约束力
放入圆括号	(表达式)	最强的约束力
功能调用	Function name (parameter list)	
求幂	EXPT	
取反	- NOT	
乘法 除法 取模	* / MOD	
加法 减法	+ -	
比较	<,>,<=,>=	
等于 不等于	= <>	
布尔运算与	AND	
布尔运算异或	XOR	
布尔运算或	OR	最弱的约束力

下面这些是结构化文本中的其它指令，和例子一起安排在一个表中。

指令类型	例子
赋值	A:=B; CV := CV + 1; C:=SIN(X);
调用一个功能块并使用功能块输出	CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
RETURN	RETURN;

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

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

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

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

对操作数赋值

“=”号左边是一个操作数（变量，地址），它的右边是赋予它的表达式的值

例如：

Var1 ： =Var2*10

在运算结束后，变量 Var1 就得到了 Var2 的 10 倍值。

在结构化文本中调用功能块

通过写 功能块的实例名和随后在括号中给参数分配值来调用一个 功能块 ，在下面的例子中，通过给两个参数IN和PT赋值来调用一个定时器，然后结果变量Q的值赋予变量A

结果变量，就象在指令表中，被表示为功能块名称后跟一个小点和变量的名字。

CMD_TMR(IN := %IX5, PT := 300)；

A:=CMD_TMR.Q

RETURN 指令

返回指令可以用来按照条件离开一个 POU（程序组织单元）。

IF 指令

IF 指令可以检验一个条件，根据这个条件，执行指令。

语法：

IF <Boolean_expression1> THEN
<IF_instructions>
{ELSIF <Boolean_expression2> THEN
<ELSIF_instructions1>
ELSIF <Boolean_expression n> THEN
<ELSIF_instructions n-1>
ELSE
<ELSE_instructions>}
END_IF;

在{}中的部分是可选的。

如果布尔运算表达式<Boolean expression>返回 TRUE，只有 if 指令部分执行，其它部分不执行。

否则，布尔运算表达式从<Boolean expression 2>开始，一个接一个的计算，直到某个布尔表达式返回为 TRUE，然后，在这个布尔运算表达式 2 之后，ELSE 或 ELSE IF 之前的部分被计算。

如果没有任何一个布尔运算表达式返回 TRUE，那么只计算 ELSE 下的指令

例如：

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

这里当温度降到 17 度以下时加热开始，否则保持关闭状态。

CASE 指令

使用 CASE 指令，可以在一个结构中，用同一个条件变量组合多个有条件的指令。

句式：

CASE <Var1> OF
<Value1>: <Instruction 1>
<Value2>: <Instruction 2>
<Value3, Value4, Value5>: <Instruction 3>
<Value6 .. Value10>: <Instruction 4>
...
<Value n>: <Instruction n>
ELSE <ELSE instruction>
END_CASE;

•CASE 指令根据下面的模式来处理

•如果变量 Var1 有值 Value1，那么执行指令 Instruction1。

•如果变量 Var1 不是所指明的值，那么执行 ELSE Instruction。

•如果有多个变量值要执行同一个指令，那么这些条件执行一个公共指令

•如果对于一个变量在一个值的范围内执行同一个指令，那么在初始值和最后值之间用两个句点隔开，所以你可以规定公共条件。

例如：

CASE INT1 OF
1, 5: BOOL1 := TRUE;
BOOL3 := FALSE;
2: BOOL2 := FALSE;
BOOL3 := TRUE;

```
10..20: BOOL1 := TRUE;
BOOL3:= TRUE;
ELSE
BOOL1 := NOT BOOL1;
BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

FOR 循环

通过 FOR 循环程序可以编写重复执行的过程。

句式：

```
INT_Var :INT;
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO
<Instructions>
END_FOR;
```

{}内的部分是可选的。

只要计数器 INT_Var 不大于 END_VALUE，指令 Instructions 就一直执行，在执行 Instructions 之前首先检查计数器的值，如果 INIT_VALUE 比 END_VALUE 大的话 Instructions 将不在执行。

当 Instructions 执行后，INT_Var 通常要增加一个 Step size，Step size 可以是任何整型值，如果没有 Step size，它将设置为 1，当 INT_Var 大到一定值时，循环结束。

例如：

```
FOR Counter:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

我们假设 Var1 的默认值是 1，那么在循环结束后它将得到值 32

注意：END_VALUE 一定不要大于等于与计数器 INT_VAR 的极限值，例如：如果变量计数器是一个 SINT 类型并且 END_VALUE 为 127，那么这将是一个死循环。

WHILE 循环

WHILE 循环可以象 FOR 循环那样使用，不同之处在与 WHILE 循环的退出条件可以是任何布尔型表达式，当条件满足时，就会执行循环。

句式：

```
WHILE <Boolean expression>
<Instructions>
END_WHILE;
```

只要 Boolean_expression 返回 TRUE，那么就重复执行 Instructions 如果 Boolean_expression 在首次计算出 FALSE，那么指令将不再执行，如果 Boolean_expression 从不出现 FALSE，Instructions 将没完没了的重复执行。

注意：程序员必须保证不出现死循环，这可以通过改变循环中指令部分的条件来实现，例如：可以通过计数器增加或减少。

例如：

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

对于 WHILE 和 REPEAT 个循环在循环之前不必知道循环的次数，从这个意义上来说，这两种循环要比 FOR 要强大一些。因此在这种情况下，可以用这两种循环。如果循环数比较明确，那么 FOR 循环因为没有死循环

而更好一点。

REPEAT 循环

REPEAT 循环和 WHILE 循环的不同之处在于它的中断条件是在循环执行之后才被检查，这就是说，循环至少要执行一次，不管中断是什么条件

句式：

```
REPEAT
<Instructions>
UNTIL <Boolean expression>
END_REPEAT;
```

Instructions 一直执行到 Boolean expression 返回 TRUE

如果 Boolean expression 第一次就赋予真值，Instructions 只执行一次，否则 Instructions 将重复执行将会导致时间延迟。

注意：程序员可以通过改变循环中指令部分的条件来保证没有死循环出现，例如：可以通过计数器增加或减少。

例如：

```
REPEAT
Var1 := Var1*2
Counter := Counter-1;
UNTIL
Counter=0
END_REPEAT;
```

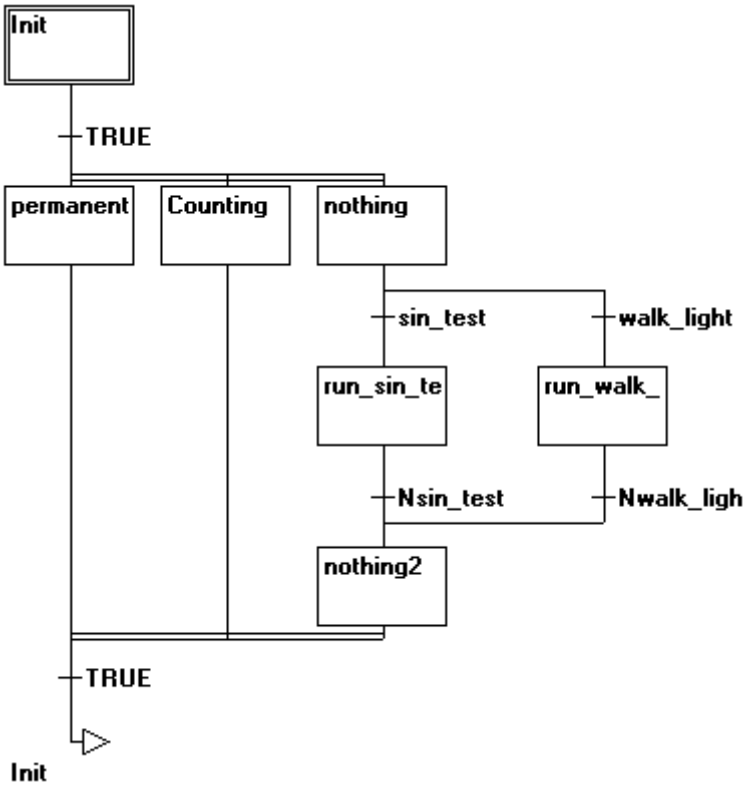
EXIT 指令

如果在 FOR WHILE 或 REPEAT 循环中有 EXIT 指令，那么内循环就结束，不管中断是什么条件。

2.2.3顺序功能图（SFC）

顺序功能图是基于图形化的语言，用它可以描述一个程序中不同动作的先后顺序。因为这些动作分配给单步元素，通过变迁元素来控制处理的顺序。

下面是一个顺序功能图的例子：



关于顺序功能图编辑器的更多知识请参照 5.4.4 章节

参照：

步

动作

进入和退出动作

转换/转换条件

激活步

IEC 步

限定词

顺序功能图种的隐含变量

SFC 标志符

可选分支

平行分支

跳转

在联机模式下可参考编辑和行为信息：

顺序功能图编辑器

顺序功能图联机模式

步

用顺序功能图编写的程序组织单元包含了一系列的步，这些步之间是通过定向连接（转换条件）实现的。

有两种类型的步：

•简单类型：每步包括一个 动作 和一个标记，这个标记用来表示此步是否激活。如果单步动作正在执行，那么在步的右上角方向会出现一个小三角形。

•IEC 类型：每步包含一个标记和一个或多个赋值的动作或布尔变量。相关的动作出现在步的右边。

动作

一个动作可以包含一系列的指令表或结构化文本指令，功能模块图或 梯形图许多的网络，或者又包含另外顺序功能图。

在 简单步 中，动作经常是和步连接在一起的，为了能编辑一个动作，在步上双击鼠标或选择此步再选择菜单命令'扩展''快速动作/转换'。另外，每一个步中允许一个输入或输出动作。

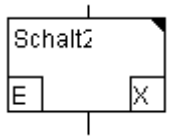
IEC步的动作是附加在顺序功能图-程序组织单元内的对象管理器中，通过双击或者在它的编辑器中按Enter键可以加载它。也可以通过"'工程''添加动作'来创建一个新的动作。你可以为一个IEC步分配最多九个动作。

进入和退出动作

可以额外的为一个步添加一个进入和退出的动作，在一个步激活后，一个进入动作只能执行一次。退出动作只在步失效之前执行一次。

进入动作用左下角一个"E"来表示，退出动作用右下角的"X"表示。

下面是一个带有进入和退出动作的步的例子：



转换/转换条件

在步和步之间有所谓的转换。

转换条件的值必须是TRUE或FALSE，因而它可以是一个布尔变量、布尔地址或布尔常量。在结构化文本句式（例如（I<=100）AND b）或者在任何一种期望的语言（参照'附加''快速动作/转换'）中，它也能包括一系列有布尔结果的指令。转换中不能包括程序、功能块或赋值。

注意：除了转换外，也能用渐进模式跳到下一步，查看 SFCtip 和 SFCtipmode

激活步

在调用顺序功能图的POU后，初始化步的 动作（被一个双边线包围）将首先执行。动作正在执行的步，状态是激活的，在联机模式下，激活的步显示为蓝色。

在一个控制循环中激活步的所有动作都将执行。所以，当激活步之后的 转换条件是TRUE时，它之后的步被激活。当前激活的步将在下个循环中再执行。

注意：如果激活的步包含一个输出动作，譬如它下面转换条件是 TRUE，那么它只能在下个循环过程中执行。

IEC 步

在顺序功能图中可以使用标准的 IEC 步。

为了能使用 IEC 步，你必须在你的工程文件中联接 Iesfc.lib 库文件。

一个IEC步中不能分配超过九个 动作 ，IEC的动作不象简单步那样固定地作为输入或输出到某个步的动作，而是和步分开存储并且能够在一个程序组织单元中重复使用多次。因此，它们必须用命令'扩展连接动作'和单个步联系在一起。

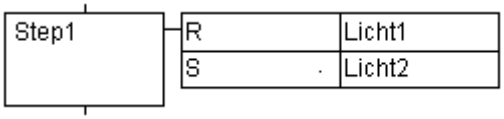
除了动作，布尔变量也能分配给步。

能够使用所谓的 限定词来控制激活和未激活的动作和布尔变量。时间延迟是可能的，如果一个动作依然激活这，而下一个步已经开始处理了。通过限定词S（设置），可以取得并发的过程。

随着每一个顺序功能模块的调用，相关联的布尔变量被设置或复位，也就是说，随着每一次调用，这个值将在 TRUE 到 FALSE 之间来回变化。

IEC 步的关联动作在步右边的两长方形中表示，左边的区域包含了限定词，可能带有时间常量，右边的区域包含了动作名和各自的布尔变量名。

下面是一个带有两个动作的 IEC 步：



为了处理的方便，联机模式下的所有激活动作象激活步一样都显示为蓝色，在一个循环之后检查一次哪个动作是激活的。

注意：如果一个动作已经失去激活了。它会再执行一次，这就是说，每一个动作至少被执行两次。

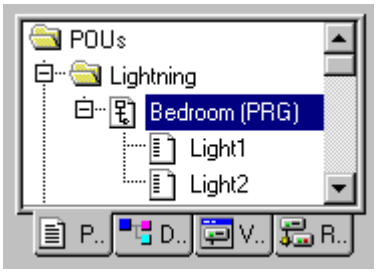
在首次调用一个未激活的动作时，激活的动作将按字母表的顺序执行。

一个新插入的步是不是 IEC 步，取决于命令菜单 ‘’扩展’’使用 IEC 步’是否被选中。

在对象管理器中，动作都直接存放在各自的SFC POU中，新的动作可以通过’工程’’添加动作’.来创建。

要使用 IEC 步，你必须在工程文件中包含特殊的 SFC 库文件 lecsfc.lib

在对象管理器中带有动作的SFC POU



限定词

为了关联 动作和 IEC步，用到下面的限定词。

N	非存储	动作和步一起激活
R	复位	动作是未激活的
S	设置	动作被激活再复位前保持激活状态
L	时间限制	动作激活一段时间，最大和步激活时间一致
D	时间延迟	如果步仍然激活，动作在一定时间后激活，然后只要步是激活的，它就保持激活。
P	脉冲	如果步激活，动作只执行一次。
SD	存储和时间延迟	在一定时间之后动作激活并保持激活状态到下一个复位开始。
DS	延迟和保持	只要步仍然激活并且保持到下一个复位开始，那么在一定时间后动作被激活激
SL	保持和时间限制	动作激活并保持一段时间

限定词 L、 D、 SD、 DS 和 SL 需要一个 TIME 常量格式的时间值。

注意：当一个动作失去激活时，它会再执行一次。这就是说每个动作至少执行两次。

顺序功能图种的隐含变量

在 SFC 中使用一些隐含声明的变量。

每一个步都有一个标记，标记中存储着步的状态。对于 IEC步来说，步的标记（激活或未激活）被称为<StepName>.x或者对一个 简单的步来说称为<StepName>，当关联的步激活的时候这个布尔变量值为TRUE，反之则值为FALSE。它能够用在SFC模块中的每一个 IEC动作和转换中。

可以通过查询<ActionName>.x 来查询一个 IEC 步是否激活。

隐含变量<StepName>.t 能够用来查询步激活的时间。

隐含变量也能够被其它程序访问，例如，boolvar1:=sfc1.step1.x; 这里 step1.x 是隐含布尔变量，它

代表了程序组织单元 sfc1 中的 IEC 步 step1 的状态。

SFC标志符

SFC 程序组织单元标志符用来控制操作，它在工程运行期间隐含的创建，为了能读这些标志符，你必须定义合适的全局变量或局部变量。例如，如果在一个 SFC 程序组织单元中一个步激活的时间超过了它定义的属性，那么就会设置一个标志符，通过用一个“SFCError”变量可以访问到这个标志符（此时 SFCError 得到真值）。

可以定义下列标志符变量：

SFCEnableLimit:这个变量的类型是布尔型，当它的值为 TRUE 时，这一步的超时将会注册进 SFCError，其它的超时将被忽略。

SFCInit:当这个布尔变量值为 TRUE 时，顺序功能图复位到初始状态，其它的 SFC 标志符也会被复位。初始步保持激活，直到变量值为 TRUE 时，才开始执行。只有当 SFCInit 被重新设置为 FALSE 时，模块才能正常工作。

SFCReset:这个布尔变量，与 SFCInit 很相似，不同之处在于，进一步的处理发生在步的初始化之后，因而，例如 SFCReset 标志符可以在初始化步中被复位到 FALSE。

注意:从 2.3.7.0 版编译器开始，SFCReset 可以用于复位与 IEC 步相关联的布尔型动作。

SFCQuitError:当这个布尔变量得到 TRUE 时，SFC 的执行将会停止，因此，在变量 SFCError 中一个可能超时将复位，当这个变量呈现 FALSE 时，激活步中的所有时间都会复位，先决条件是在 SFC 中已经定义过登记任何超时设定的标志符 SFCError。

SFCPause:当这个布尔变量值为 TRUE 时，SFC 图表的执行就会停止。

SFCError:当在 SFC 图表中有超时时这个变量得到 TRUE。如果在这个之后还有其它的超时发生，除非是这个变量已经复位，否则，这些状态将不会登记。如果你想使用其它的时间控制标志符(SFCErrorStep,SFCErrorPOU, SFCQuitError,SFCErrorAnalyzation)的前提条件是定义 SFCError

SFCTrans:当一个转换被驱动时，这个布尔变量得到真值。

SFCErrorStep:这个变量是一个字符变量，如果 SFCError 中登记了一个超时，这个变量将存储这个超时步的名字。前提条件是在 SFC 中已定义了登记任何超时的标志符 SFCError。

SFCErrorPOU:这个字符变量包含了发生超时的模块名字。前提条件是在 SFC 中已定义了登记任何超时的标志符 SFCError。

SFCCurrentStep:这个字符变量存储了那个被激活步的名字，它与的时间监控无关。在仿真的情况下，此步存储在外部适当的分支种。如果一个超时发生其它的将不再登记，而且 SFCError 也不会复位。

SFCErrorAnalyzationTable:这是数组型变量，它提供一个转换表达式的分析结果，表达式中对转换中的 FALSE 和上一步起时有影响的每个一元素。要把下列信息写进 ExpressionResult 结构中写进：名称，地址，注释，当前值。

最大可以容纳 16 个元素，因此，数组的范围从（0-15）。

ExpressionResult 结构和隐含使用的分析模块都是由 AnalyzationNew.lib 库文件提供的，分析模块也能够被其它的不用 SFC 编写的程序组织单元显式使用。

上一步的超时登记是分析一个转换表达式的先决条件，因此在这里必须有一个时间监视的应用，而且，SFCError 必须在声明窗口被定义。

SFCTip, SFCTipMode:这个布尔变量允许 SFC 的渐进模式。当用在 SFCTipMode=TRUE 来切换它时。如果 SFCTip 设置值为 TRUE 时它只可能跳到下一个步，只要 SFCTipMode 是设置为 FALSE 时，它可能跳过转换。

注意:对于扫描的状况和分步运动时间 隐藏变量还是可用的。

可选分支

在SFC中可以定义两个或两个以上的可选择分支。每一个分支的开始和结束必须带有一个 转换。可选分支可以包含平行的分支和其它的选择分支，一个可选分支开始于一个水平线并终止于一个 水平线（选择结束）或是一个跳跃。

如果在可选分支开始行前面的步是激活的，每一个可选分支的首次变换将从左到右被计算，最先的转换将从左边转换条件为 TRUE 的开始，然后下面的步被激活。

平行分支

在 SFC 中可以定义两个或两个以上的分支为平行分支。每一个平行分支在开始和结束处必须有一个步。平行分支可以包含可选择的分支或其它平行分支，一个平行分支开始于一个双划线，结束于一个双划线或者一个跳跃，它能提供一个跳跃标识。

如果平行分支的先前步是激活的，并且这个步之后的 转换条件值是TRUE时，那么平行分支的第一步激活。这些分支彼此平行处理。当所有平行步激活并且这些步之后转换条件为TRUE时，那么平行分之末端线后的那一步被激活

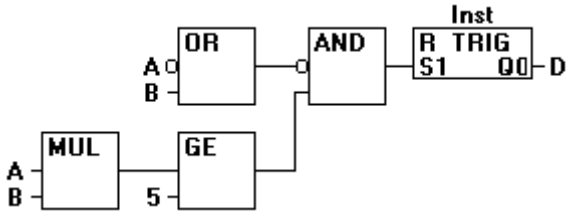
跳转

跳转是对在跳转符号下面指明的 步名的一个连接。当在不允许创建导致向上或互相交叉联系的时候，必须使用跳转。

2.2.4 功能模块图

功能模块图是一种基于图形的编程语言，它用一串网络来工作，每一个网络包含一个算术或逻辑表达式、功能块的调用、跳转或返回指令的结构。

下面是功能模块图中一个网络的例子



关于功能模块编辑器的更多的知识请参照 5.4.1 章节中的讲述。

2.2.5 连续功能图表编辑器

连续功能图表编辑器不象功能模块图表那样操作，但是可以自由放置元素，它允许使用反馈。

关于连续功能图的更多信息请参照 5.4.4 章节

连续功能图编辑器中网络的例子：

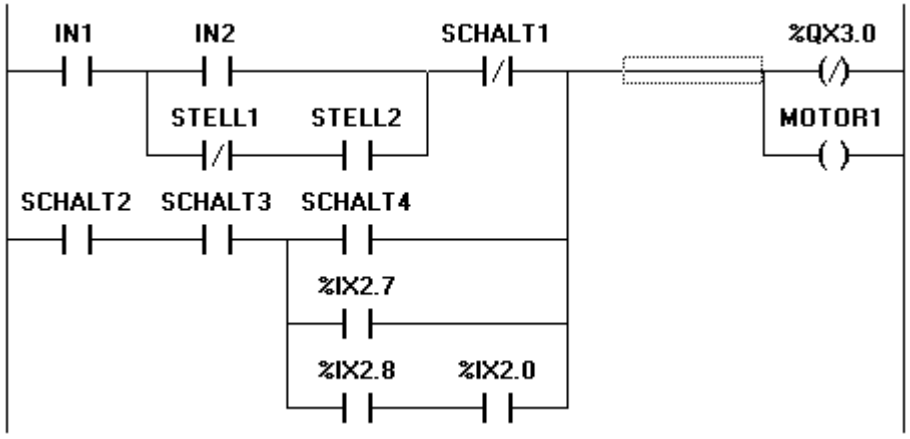
2.2.6 梯形图

梯形图也是一种基于图形化的编程语言，它接近于电子电路的结构，一方面，梯形图很适合构建逻辑开关，另一方面，它也能创建象 FBD 中的网络图，所以梯形图在控制调用其它程序组织单元的时候是很有用的。

梯形图包含了一系列的网络，左右两边各有一个垂直的电流线，网络图仅限制于左右两母线之间的范围内，在中间是由线圈触点和连接线组成的电路图。

每一个网络包含左边的一系列触点，这些触点根据布尔变量值的 TRUE 和 FALSE 来传递从左到右的开和关的状态。每一个触点是一个布尔变量，如变量值为 TRUE，电路从左到右通过连接线就连通。否则右边接收到“关“的值

下面是一个梯形图的例子，它由线圈和触点组成



触点

在梯形图中的每一个网络图的左边都有触点（触点是用两个平行线||来表示），它用来表示电路的“开”“关”状态。

这些状态与布尔变量 TRUE 和 FALSE 相一致。布尔变量属于每一个触点。如果变量值为 TRUE，那么状态可以通过连接线从左边传到右边。否则，右边接收到的是“断开”。

触点可以并联使用，其中的一个并联分支必须传递“开”状态时，并联分支才能传递“开”。或者触点串联连接，此时，触点必须传递“开”状态时，最后的触点才传递“开”，这些与串并联连接的电路一致。

线圈

在梯形图网络图的右边有一些所谓的线圈，它们用（）表示并且只能通过水平线来连接。线圈传递从左到右的连接状态，并且复制状态到布尔变量中，可以描述入口线的状态为“开”（对应布尔变量的 TRUE）或者“关闭”状态（对应布尔变量的 FALSE）。

触点和线圈也可以取否定值（在上例中的触头 SWITCH1 和线圈%QX3.0 是取否定值）。如果触点取否定值（在触头符号中用“/”来表示），然后把它复制否定后的值到相应的布尔型变量中。如果一个触点取否定值，仅且相应的布尔变量取到 FALSE 时，电路才能连通。

梯形图中的功能模块

可以在网络图中添加功能模块和程序，但它们必须具有布尔型值的输入和输出并且可以象触点那样用在梯形图的左边。

梯形图中的功能块

在 接点和 线圈上你也可以在网络图中输入 功能块 和程序，它们必须有布尔变量的输入和输出并且可以被用在和触点一样的地方，它在LD网络图左边。

置位/复位线圈

线圈也可以被定义设置或重置线圈。以“S”做为线圈被设置的标志：(S))。它从不在布尔变量上写入TRUE。也就是说，如果一个变量被设为TRUE，它便被保留了下来。

以“R”做为线圈重置的标志：(R))。它从不在布尔变量上写入 FALSE：如果一个变量被设为 FALSE，它便被保留了下来。

LD和FBD

当应用LD是你很可能想用为了连接其他POUs的接触开关。令一方面，你可以用 线圈把结果插入一个可以在其他地方使用的全局变量你还可以在你的LD网络图中插入可执行命令。为此你导入一个POU和EN输入。

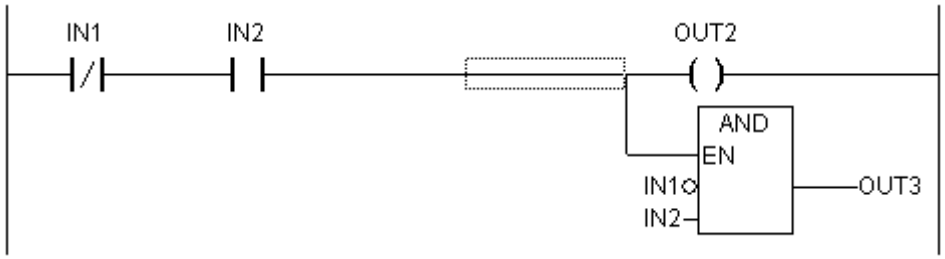
这样的POUs是完全正常的有附加输入并以EN为标志的操作数、功能、程序或者 功能块。EN输入总是BOOL类型并且它的意思是：当EN存在真值时，EN输入的POU求值。

EN POU 平行连接到线圈上。EN 输入连接到接点和线圈中间的连接线。输入 ON 信息在线上传送，这个 POU

将被求值。

从 EN POU 开始，你可以创造和 FBD 相似的网络图

EN POU的LD 网络图的例子



2.3 调试、联机功能

采样追踪

采样追踪允许你追踪变量的连续变化的值，它依赖于所谓的触发事件，触发事件是先前定义的布尔变量（触发变量）的上升沿或下降沿。CoDeSys 允许对 20 个变量进行追踪，每一个变量可以追踪 500 个值。

调试

CoDeSys 的调试功能可以让你很容易的找到错误。

为了调试，运行‘工程’‘选项’命令并且在 生成选项对话框中选择动态调试。

断点

断点是程序处理过程中停止的位置，因而它可以在程序中的特定位置观察变量值的变化。

断点可以在编辑器中设置，在文本编辑器中断点在行的编号处设置,在连续功能图中和梯形图中是在网络编号处设置,在 CFC 中是在程序组织单元处设置,在 SFC 中是在步处设置，在功能模块图的实例中不能设置断点。

注意: CoDeSys SP 全 32 位运行系统只要程序在断点处停止运行，它将取消与之相连任务的看门狗功能.

单步

单步是:

在指令表中:执行程序直到运行 CAL LD 和 JMP 命令。

在结构化文本中：执行下一条指令。

在功能模块图 梯形图中： 执行下一步网络。

在顺序功能图中：继续目前的 动作 直到下一个 步开始。

通过一步一步的运行你可以检查程序中的逻辑错误。

单循环

如果选择了单循环，每一个循环结束，执行也就结束。

联机模式下改变值

在操作过程中，变量可以设置为一个特定的值（写入新值）或者在每一个循环之后重新定义为特定的值（强制新值）。在联机模式下可以通过双击变量的值来改变它的值，布尔变量从TRUE变为FALSE或从FALSE变为TRUE。对于每一种类型的变量都可以打开写入变量对话框。在这里可以编辑变量的真实值。

监视

在联机模式下，所有的显示变量从控制器中读出并及时的显示。你可以在定义和程序编辑器中找到这些显示。你也可以在观察和接收器中读出变量的当前值并且可以看到它们。如果要监视功能模块的实例中的变量,相应的实例块必须已经打开。

在监视 VAR_IN_OUT 变量时，不引用的值将输出。

在监视指针时，指针和不引用的值都将在声明部分输出。在程序部分，只有指针输出：

+ --pointervar = '<'pointervalue'>'

在不引用值中的 POINTER 也相应的显示。在行上双击或在交叉上单击，显示或是展开或是收缩。

在执行部分，显示指针的值。对于不引用，将显示不引用的值。

监视数组元素：数组元素除了由常量指出的之外，还有由变量指出的：

anarray[1] = 5

anarray[i] = 1

如果索引中包含有表达式(例如， [i+j] or [i+1])，元素不能显示出来。

请注意：如果已经达到了被监视变量的最大编号，对于随后的变量不是显示当前的值，而是显示字符串“监视的变量太多”。

仿真

在模拟过程中，创建的 PLC 程序不在实际的 PLC 中运行，而是 CoDeSys 系统中的计算器中运行。所有的联机功能都是可用的。它允许你在无需 PLC 硬件的情况下检测逻辑的正确性。

注意：外部库文件的 POU 是不能运行在模拟的模式的。

日志

日志记录着用户的操作、内部进程、状态变换和联机模式处理过程中发生的意外的情形。它用来监视和跟踪错误。

2.4 标准化

IEC61131-3 是一种国际标准化的 PLC 编程语言。

在 CoDeSys 中提供的可编程语言符合标准化的要求。

根据这个标准，一个程序包含以下元素：

结构（请看 数据类型）

程序组织单元（POU）

全局变量

通用的语言的元素在标识符、地址、类型、注释和常量部分中已经讲述。

CoDeSys程序的处理是从一个特殊的程序组织单元（POU）PLC_PRG开始的，程序组织单元PLC_PRG能调用其它的程序组织单元。

3. 我们来编写一个小程序

3.1 控制一个交通灯信号单元

现在让我们编写一个简单的控制交通信号单元的小程序，用它来控制十字路口的交通信号灯，红/绿信号灯交替作用，为了避免事故，我们插入黄色或黄/红变换状态，后者比前者时间要长。

在这个例子中,将使用标准的IEC1131-3标准化语言来显示时间依赖程序是怎样工作的,怎样能在CoDeSys帮助下编辑其它的标准化语言，怎样在熟悉CoDeSys仿真的情况下很容易的来连接它们。

创建POU

打开CoDeSys并选择'文件''新建'

在出现的对话框中，最先的 POU已经给予默认的名字 PLC_PRG，保持这个名字不变，这个POU定义为一个程序，每个工程文件都需要一个这样名字的程序。在这个例子中，我们选择连续功能图编辑器作为这个POU的编程语言。

现在创建三个对象，用命令“'工程''工程添加'或者用快捷菜单（在 对象管理器中按右键)。在 顺序功能图中创建一个名为SEQUENCE的程序，在 功能模块图中创建一个名为TRAFFICSIGNAL的 功能块，在 指令表中创建一个名为WAIT的功能块。

交通信号的用途

在 POU TRAFFICSIGNAL 中，我们将给交通灯分配各自的信息状态。例如我们要保证红灯在红和黄/红状态应该变红和黄灯在黄和黄/红状态变黄等等。

WAIT的用途

在 WAIT 中，我们将编写一个简单的计时器，它的输入端将接收状态的毫秒值，当时间段完成时，它的输出端将产生 TRUE 值。

SEQUENCE的用途

在 SEQUENCE 中，所有的状态都组合在这里。因此，灯可以在我们期望的时间段内点亮。

PLC_PRG的用途

在 PLC_PRG 中，输入启动信号连接到交通灯同时颜色指令作为灯的输出。

交通信号模拟

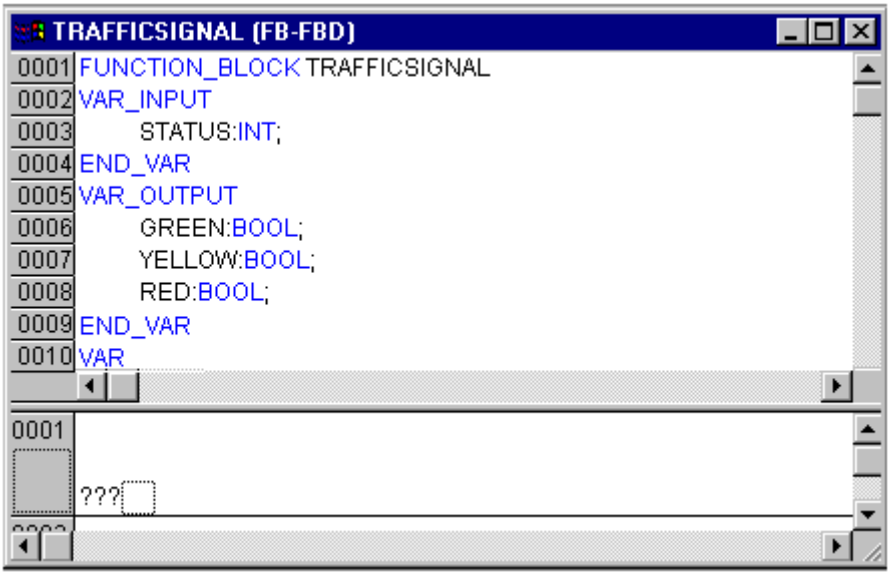
现在模拟程序中测试这个程序。编译（“工程”“生成”）并加载（“联机”“登录”）。 通过“联机”“运行”来启动程序，然后设置变量 ON 为 TRUE 。例如，在 CFC 编辑器中的输入框的条目“ON”上双击，变量将设置为〈TRUE〉。然后按<Ctrl><F7>或命令“联机”“写入值”来设置值。现在 ABLAUF 中的变量 START（在程序的开始阶段我们手动设置为 TRUE）从变量 ON 上获得到值。这用来运行交通灯循环。PLC_PRG 转换为一个监视窗口，在声明编辑器中的加号上双击，变量将顺序显示，你能看到变量的各自的值

“交通信号”声明

让我们先看一下 POU TRAFFICSIGNAL，在声明编辑器中定义输入变量 STATUS 为整型变量（在关键字 VAR_INPUT 和 VAR_OUTPUT 之间)。STATUS 有四个可能的状态，它们是交通信号状态中绿、红、黄/红、红中的一种。

对应的交通信号有三个输出，分别是红、黄和绿。你应该声明这些变量，功能块交通信号的声明部分应该如下：

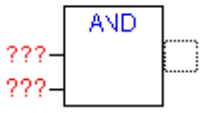
功能块TRAFFICSIGNAL声明部分



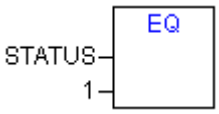
“交通信号”主体部分

由输入STATUS的值我们来决定输出变量的值，先进入POU的主体，单击第一个网络图左边的区域（编号为0001 的灰色区域）。现在选中了第一个网状，选择菜单项’插入’’框’.

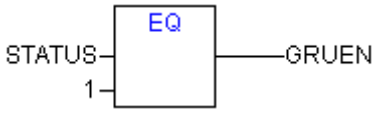
在第一个网状插入一个具有两个输入端和操作符号 AND 的方框中：



单击文本 AND，显示选中状态并改变值为 EQ，然后分别选择两个输入端（三个问号的标记）。并赋予它们值“STATUS”和“1”。



在方框EQ的后面上单击，选中EQ操作的输出端，选择’插入’’赋值’来改变三个？？？标记为GREEN，就会创建一个具有下面这个结构的网状图。



STATUS 和 1 相比较，结果赋予 GREEN，如果预先设置的状态值为 1，那么网络的状态将转换到 GREEN。

对于另外的 TRAFFICSIGNAL 颜色，我们需要两个以上的网状图。创建第一个执行命令“插入”“网络(后)”插入一个象上述的 EQ 方框,然后选择输出端并用“插入”“框”同样插入一个方框，用 OR 代替 AND，选中 OR_方框第一个输出端用命令 “插入”“赋值”把它赋予“GELB”，选中 OR_方框第二个输入端并用鼠标单击三个？？？标记的旁边水平线，出现一个带点的矩形框。用“插入”“框”再添加一个 EQ_方框。

最后我们得到如下的网络图：

为了能在一个操作数附近插入另一个操作数。你必须选择你想要添家到方框的输入端。

然后使用命令’插入’’框’.，否则你会创建与第一个一样的网状图

现在我们的 POU 已经创建结束，TRAFFICSIGNAL，根据输入变量 STATUS 的值控制我们想要的灯的颜色。

连接标准库standard.lib

为了使用定时器WAIT，我们需要POU中的一个 标准库文件。通过“窗口”“库管理器”打开 库管理器，选择 ’插入’ ’附加库’,出现打开文件的对话框，从库文件的列表中选择standard.lib库文件。

“WAIT”声明部分

现在让我们转向 POU WAIT ，这个程序组织单元用来作为一个计时器来决定每一个 TRAFFICSIGNAL 状态的时间长短。我们的 POU 接收一个 TIME 类型的时间变量作为输入变量，作为输出变量的 OK，当期望的时间段结束时，它应该为 TRUE。我们在末尾部分格式为“：=FALSE”来设置它为 FALSE。

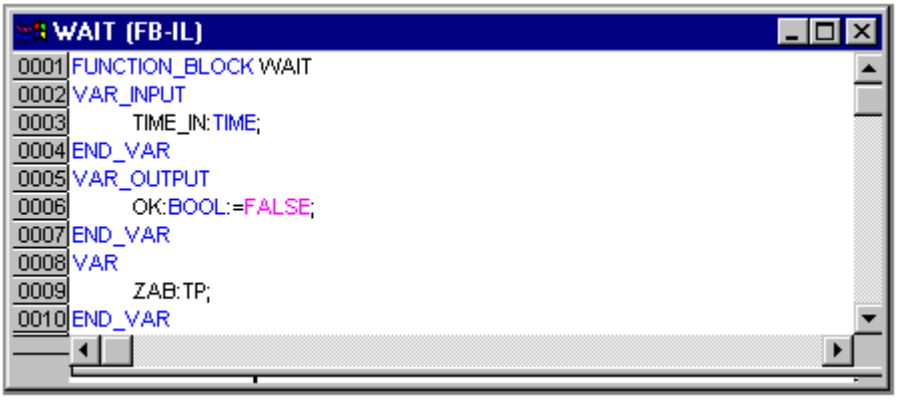
我们还需要一个 POU TP 时钟发生器。它有两个输入端（IN PT），两个输出端（Q ET） TP 做以下的工作：

如果IN是FALSE，那么ET是 0 并且Q是FALSE。只要IN为TRUE，输出端ET以毫秒开始计算时间值，当ET达到了PT的值 ，就不在计时。只要ET的值比PT小，Q就会保持TRUE。当ET的值达到PT值时。Q产生FALSE。请查看标准库中关于所有的POU的简要介绍。

为了在 POU WAIT 使用 POU TP。我们必须从 TP 中创建一个本地的实例，我们定义一个局部变量 ZAB（消逝的时间）类型为 TP（在关键字 VAR END_VAR 之间）

WAIT 的定义部分如下：

功能块 WAIT, 声明部分



“WAIT”主体部分

为了创建期望的计时器，POU 的主体部分必须编辑为如下：

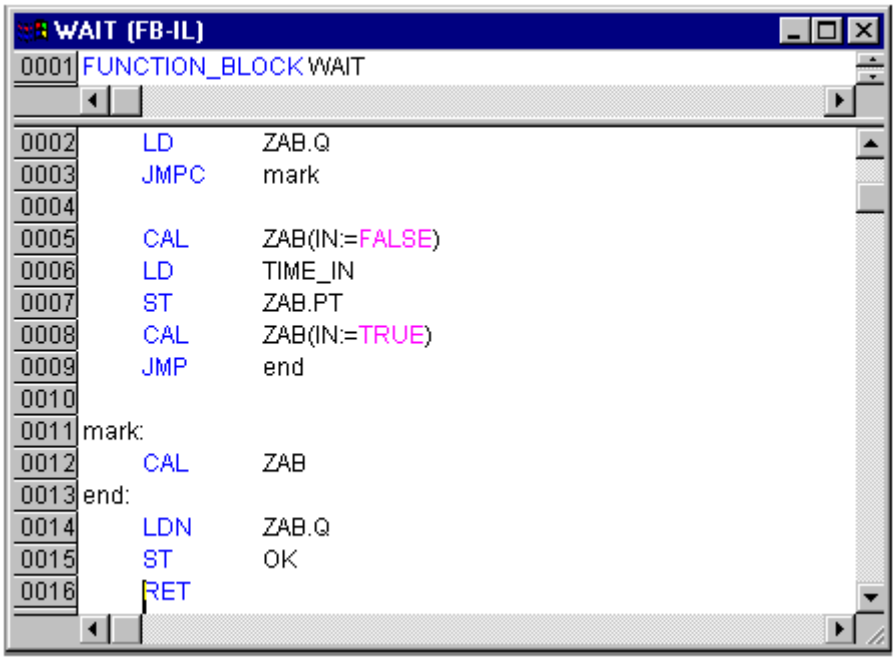
功能块 WAIT, 主体部分

首先检查 Q 的值是否设置为 TRUE（即使已经开始计数），在这样情况下，我们不改变 ZAB 的值。但是我们调用 ZAB 模块不需要输入（为了检查时间段是否已经结束）

否则我们设置变量IN值为FALSE，这样ET为 0，Q为FALSE。所有的值都设置为期望的初始状态。现在我们从变量TIME中分配必须的时间给PT，并调用ZAB， IN：=TRUE。在 功能模块ZAB中变量ET开始计算直到它达到TIME的时间值，随后Q被设置为FALSE。

Q 的否定值在每次 WAIT 执行后存储在 OK 变量中，只要 Q 是 FALSE，OK 就产生 TRUE。

计时器在这个点上结束，下面是在程序 PLC_PRG. 中组合两个功能模块WAIT和SEQUENCE的使用。

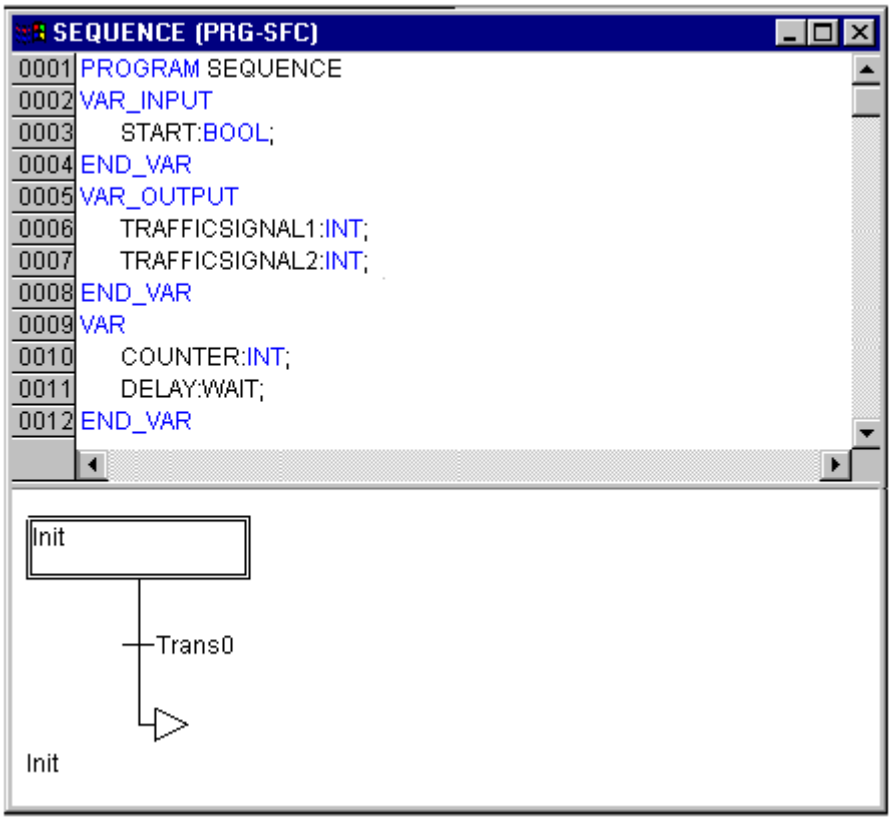


“SEQUENCE”第一扩展部分

首先我们声明我们需要的变量，它们是：布尔型输入变量 START。两个整型变量 TRAFFICSIGNAL1 和 TRAFFICSIGNAL2 和一个 WAIT 类型的变量 DELAY。

SEQUENCE 的程序如下：

程序 SEQUENCE, 第一展开部分,声明部分



创建一个顺序功能图

在 SFC 中一个 POU 的开始图表经常包含一个动作“Init”和一个伴随转变“Trans0”和返回 Init 的跳转。我们详细讲述一下：

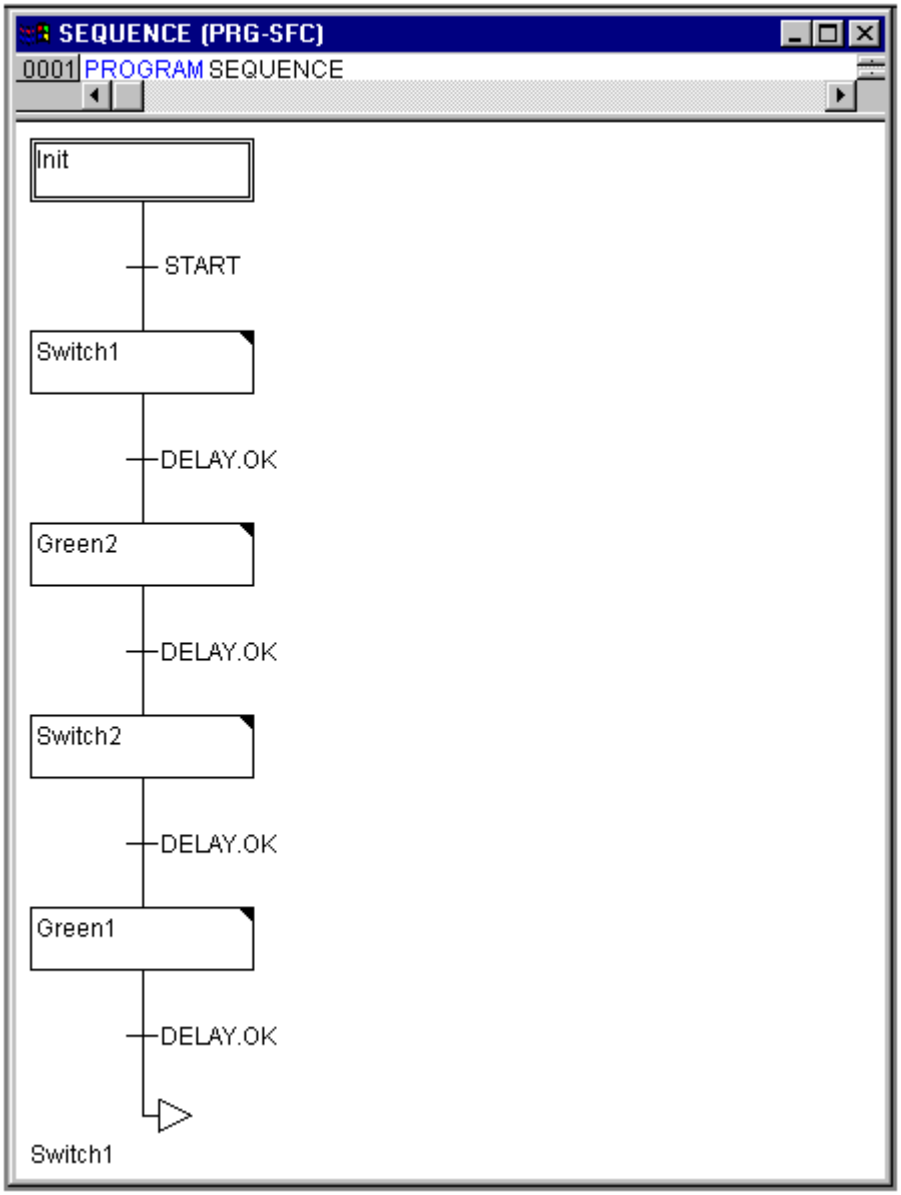
在我们编写各个动作和变换之前，我们先决定一下图表的结构。我们需要为每个 TRAFFICSIGNAL 状态分配一个步，选中标志 Transo 并选择“插入”“步转换(后)”来插入步，重复这个动作来插入三个步。

如果在步或转变名字上单击，你可以改变它。命名 Init 之后的第一个转变为 START，其他的转变“DELAY.ok”.

当 START 的值为 TRUE 并且其它所有开关通过 OK 中的 DELAY 都输出 TRUE 时,第一个变换开关接通，例如,当设定的时间段结束。

从上到下的步依次命名为 Switch1, Green2, Switch2, Green1。只有初始化过程保留它的名字，“Switch”应当包括一个黄色的状态，在 Green1，TRAFFICSIGNAL1 将变为绿色，在 Green2，TRAFFICSIGNAL2 将变为绿灯。最后在开关 Switch1 后返回到初始化的值。如果你一切做的都正确，图表应该是如下的情形：

程序SEQUENCE, 第一展开部分, 指令部分

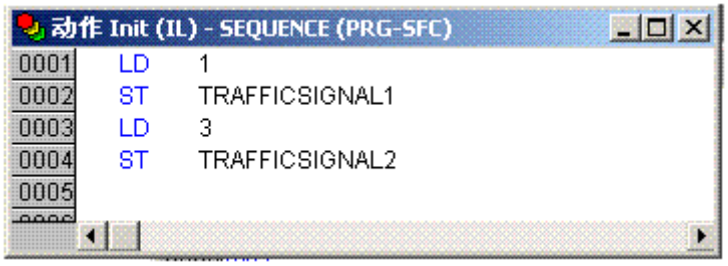


在我们已经完成了编程所需的各个步，如果在一个步上双击，那么就得到打开一个新动作的对话框，在这里，我们使用的是指令表语言。

动作和转换条件

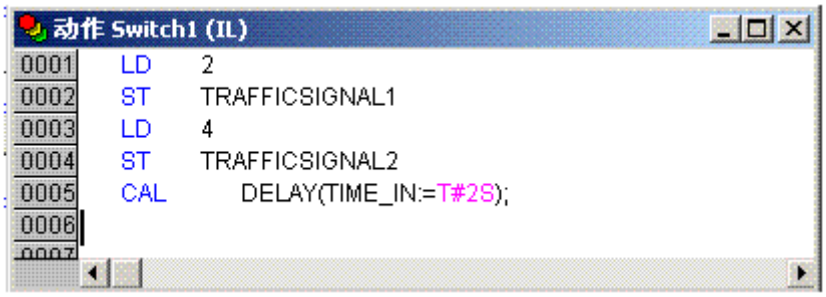
在Init步的 动作中，变量被初始化。TRAFFICSIGNAL1 的STATUS应该是 1（GREEN），TRAFFICSIGNAL2 的状态应该是 3（RED）。初始化如下：

动作 Init



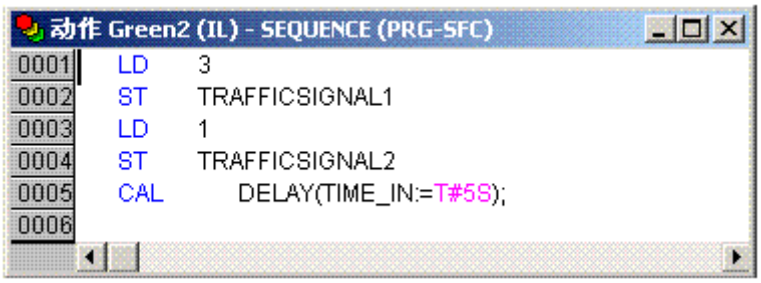
Switch1 改变 TRAFFICSIGNAL 的值为 2（yellow），并且改变 TRAFFICSIGNAL2 的值为 4（yellow-red），另外，设置了一个 2000 毫秒的时间延迟，动作如下：

动作 Switch1



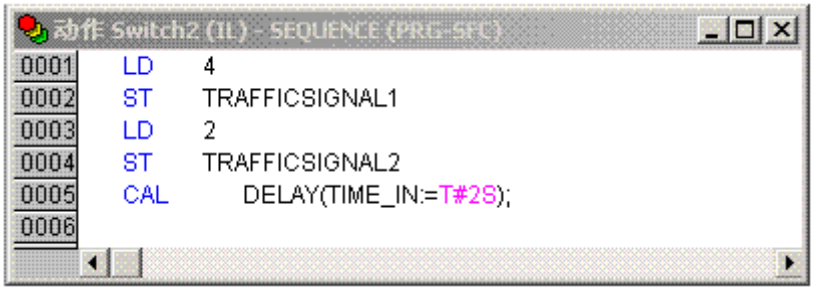
当 Green2 的 TRAFFICSIGNAL1 是红灯时（STATUS：=3）时，TRAFFICSIGNAL2 是绿色（STATUS：=1），并且延迟时间为 5000 毫秒。

动作 Green2



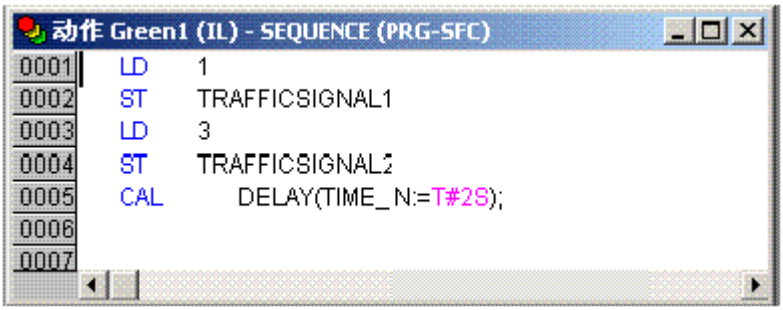
在 Switch2，TRAFFICSIGNAL1 的 STATUS 变为 4（yellow-red），TRAFFICSIGNAL2 的状态变为 2（yellow），并设置了 2000 毫秒的时间延迟。

动作 Switch2



Green1，TRAFFICSIGNAL1 是绿色灯（STATUS:=1），TRAFFICSIGNAL2 是红灯（STATUS:=3），并且时间延迟为 5000 毫秒。

动作 Green1



程序的展开部分已经完成，现在你可以在模拟模式来测试 POU ABLAUF 了。通过菜单“工程”“生成” 编译工程，在信息窗口中应该得到 0 个错误和 0 个警告。现在检查“联机”“仿真”是否激活，用命令“联机”“登录”进入仿真模式,用“联机“运行”来启动程序，通过在 ABLAUF 的入口处双击来打开 ABLAUF POU,程序现在开始启动了，不过要想运行它，变量 START 必须是 TRUE。随后我们要在 POU 中手动设置它，在声明部分中 START 被设置为 FALSE 的行上，执行一次双击，这将给它赋以 TRUE，现在选择命令“联机”“写入值”来设置它的值。在顺序功能图中 START 将以蓝色显示,当前激活步中的正在处理的步将标记为蓝色。

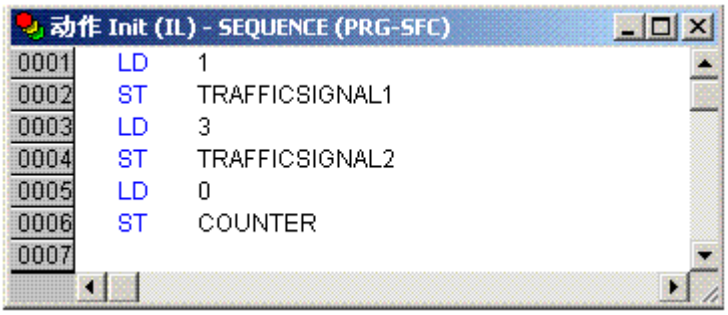
当你完成了这些中间的测试，通过使用命令“联机” “退出”来退出模拟模式,继续编写程序。

“SEQUENCE”第二扩展部分

为了保证我们的图表有至少一个可选择的分支，并且我们能够在晚上的时间关闭我们的交通灯，我们现在在程序中编写一个计数器，在一定数的 TRAFFICSIGNAL 循环之后，关闭交通灯。

首先我们需要一个新的整型变量 COUNTER，在 SEQUENCE 的声明部分定义这个变量，并初始化使它为零。

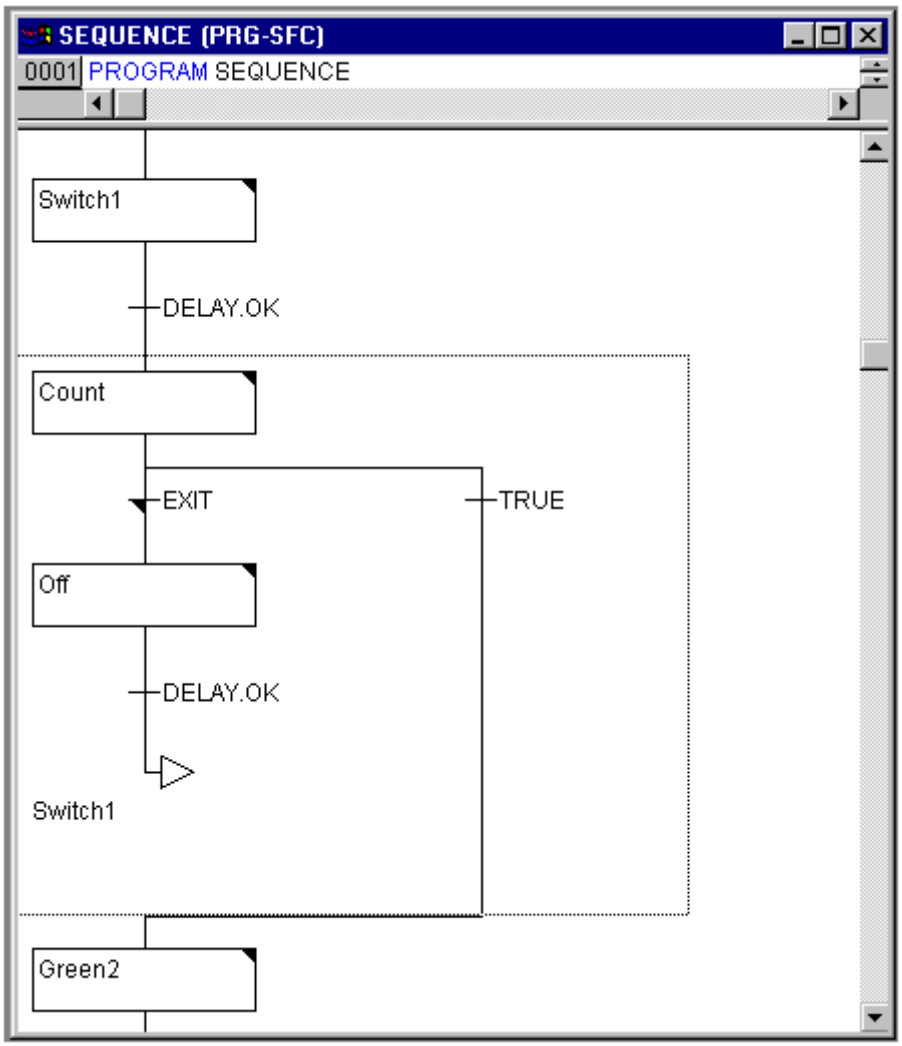
Action Init, Second Version



现在选择Switch1 后的 转变条件并插入一个步和一个转变，选择结果变换并在它的左边插入一个可供选择的分支，在左边的变换条件之后插入一个步和一个转变。在新转变条件之后，在Switch1 之后插入一个跳转。

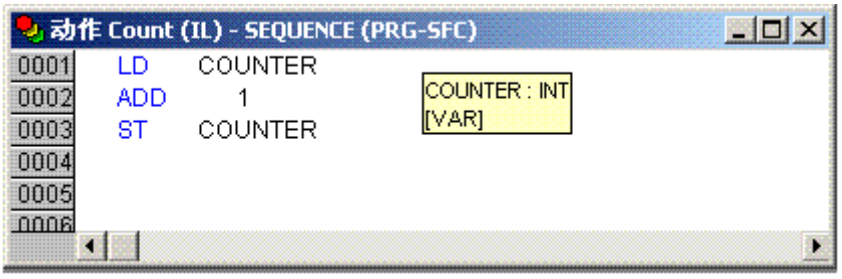
对新部分命名如下：上面两个新步命名为“Count”，下面的叫“Off”，转变名为 EXIT、TRUE 和 DELAY.OK 。新部分应该象下面用虚线框标注的部分一样。

Program SEQUENCE, Second Expansion Level, Instruction Part



现在两个新 动作和一个新 变换条件被应用，在步Count处变量COUNTER增加 1

Action Count

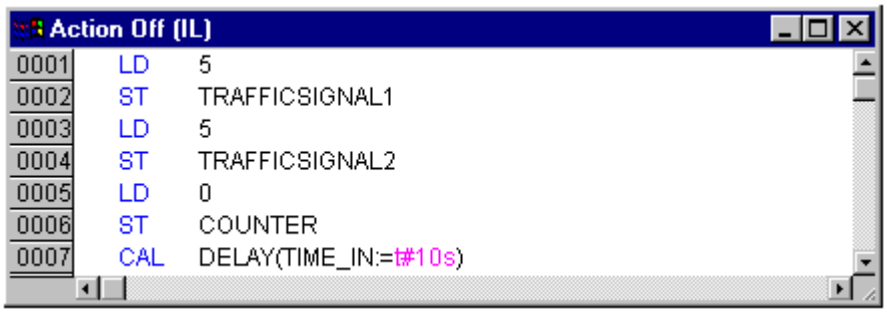


EXIT 变换检查计数器是否大于一个特定的值。例如：

Transition EXIT

在关闭状态时，两个灯都设置为 5（off）（或者每个数字都不等于 1，2，3 或 4）,COUNTER 复位为 0，并设置一个 10 秒的延迟。

Action Off



结果

在我们假定情况下，在 7 个交通信号循环之后进入夜晚,10 秒后，交通信号熄灭。随着白天的到来，交通灯又再次点亮，整个过程不断的重复。如果你喜欢，你可以在我们创建 POU PLC_PRG 之前，在模拟模式下来做当前版本的程序的测试。

PLC_PRG

我们已经在模块 SEQUENCE 中为两套交通灯定义和关联了各个阶段的时间序列，但是，我们看到的交通灯系统是一个总线系统的一个模块，例如 CAN BUS。我们必须在模块 PLC_PRG 中可利用输入和输出变量，我们希望用 ON 开关上打开交通灯系统，并且为 SEQUENCE 的各个步分配给 6 个灯相应的“信号命令”，现在为这 6 个输入和 1 个输出变量定义布尔类型，在编辑器中编写程序之前，为它们分配值，同时分配相应的 IEC 地址。

下面步是定义 Phases 类型变量 LIGHT1 和 LIGHT2。

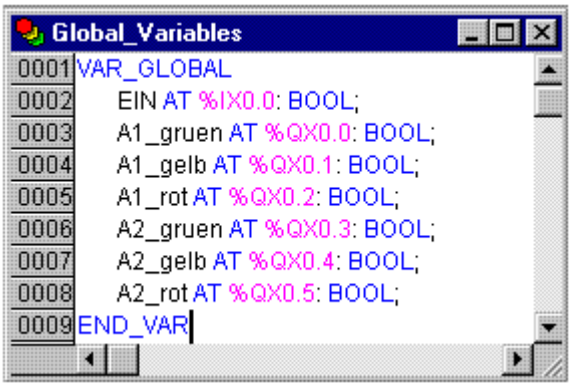
LIGHT1 和LIGHT2 的声明

这些为模块 SEQUENCE 的各个步传递 6 个灯的布尔型值给上述 6 个输出变量。但是，我们不定义在 PLC_PRG 模块中已经见到过，但在资源中代替全局变量的输出变量。布尔型输入变量 IN。用来在模块 SEQUENCE 中设置变量 START 的值为真值。也可以用同样的方法设置。ON 也会分配一个 IEC 地址。

选择资源标签并打开全局变量列表。

定义如下部分：

用于CAN配置的输入/输出变量声明



AT 跟在变量名后面， %号符号开始是 IEC 地址。I 代表输入，Q 代表输出。B（在例子中使用）代表字节。

各个模块的字节数用 0.0（0.1，0.2 等等）表示。在这个例子中，我们不需要配置控制器，因为它依赖于你计算机中的可利用的目标包，更多的信息请查看 PLC 配置部分。

我们现在要完成模块 PLC_PRG。

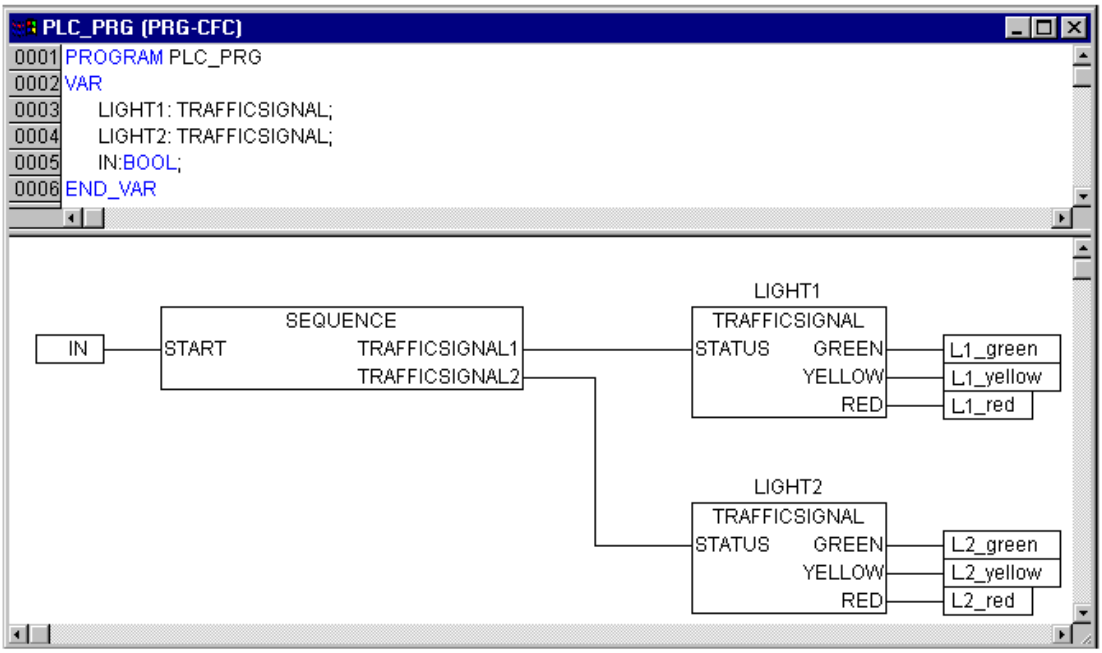
进入编辑窗口，选择 连续功能图表编辑器，从菜单栏下面的工具栏中可以找到一个CFC的工具符号（查看 连续功能图表编辑器）。

在编辑窗口单击鼠标右键，选择 Box，单击文本 AND，改为“SEQUENCE”，这样就带来 SEQUENCE 所有已经定义的输入输出变量。插入两个命名为 PHASES 的功能模块，用已定义好的变量 LIGHT1 和 LIGHT2 替代模块的???标记。现在设置一个输入类型的元素和六个输出变量依次命名为 L1_green、 L1_yellow、L1_red、L2_green、 L2_yellow、L2_red。

程序中的所有元素都已经在这里，通过在输入/输出上面的短线上单击并按住一直拖拉鼠标到期望的元素上面可以连接这些输入和输出变量。

程序显示如下：

PLC_PRG, 连续功能图下的PLC_PRG和声明



交通信号模拟

现在模拟程序中测试这个程序。编译（“工程”“生成”）并加载（“联机”“登录”）。 通过“联机”“运行”来启动程序，然后设置变量 ON 为 TRUE 。例如，在 CFC 编辑器中的输入框的条目“ON”上双击，变量将设置为〈TRUE〉。然后按<Ctrl><F7>或命令“联机”“写入值”来设置值。现在 ABLAUF 中的变量 START（在程序的开始阶段我们手动设置为 TRUE）从变量 ON 上获得到值。这用来运行交通灯循环。PLC_PRG 转换为一个监视窗口，在声明编辑器中的加号上双击，变量将顺序显示，你能看到变量的各自的值

3.2 可视化交通信号单元

可视化交通信号单元

通过 CoDeSys 的可视化。你可以很容易的把程序中的变量加以应用。我们现在为四个交通灯单元规划两个信号灯和一个 ON_按钮。这四个交通灯单元将显示开关的过程。

创建一个新的可视化

为了创建一个新的可视化，必须在对象管理器中选择可视化的范围，首先单击可视化的窗口的左边下部的Visualization选项卡的，如果你选择命令’工程’’对象添加’，将会打开一个对话框。

在这里任意输入一个名字，按 OK 按键确认。随后打开一个窗口，在这里你可以设置你的新的可视化内容。

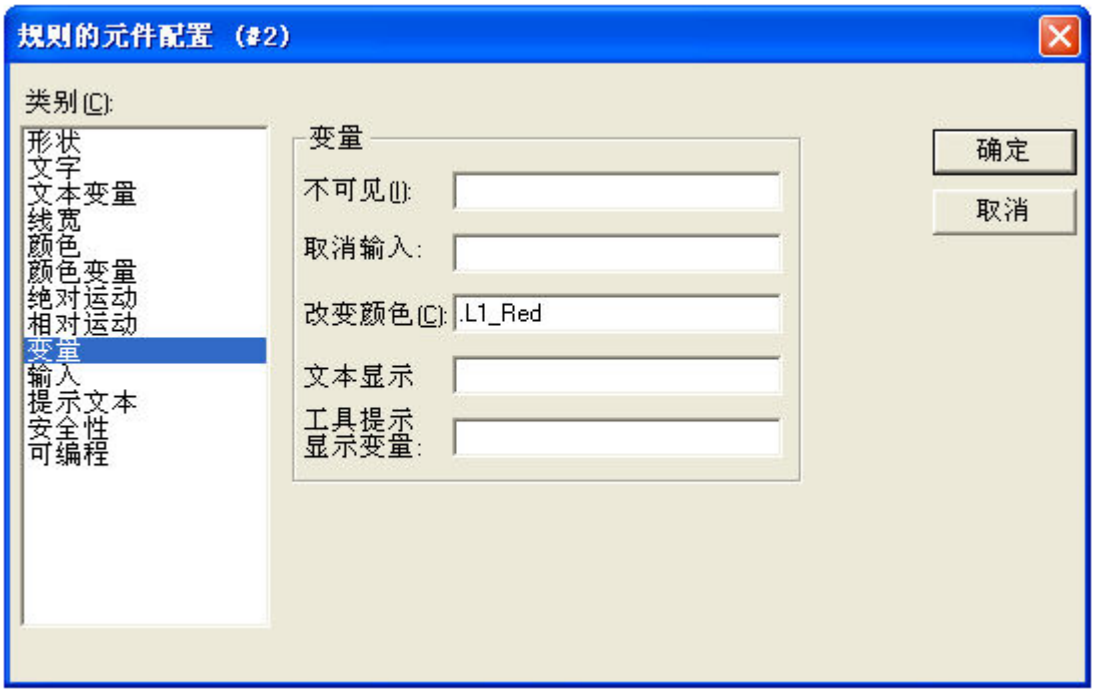
在可视化中插入元件

为了交通信号的可视化，按照如下来处理：

通过命令“插入”“椭圆”画一个 2cm 的圆,在编辑区域中按鼠标左键拖动到合适的尺寸。

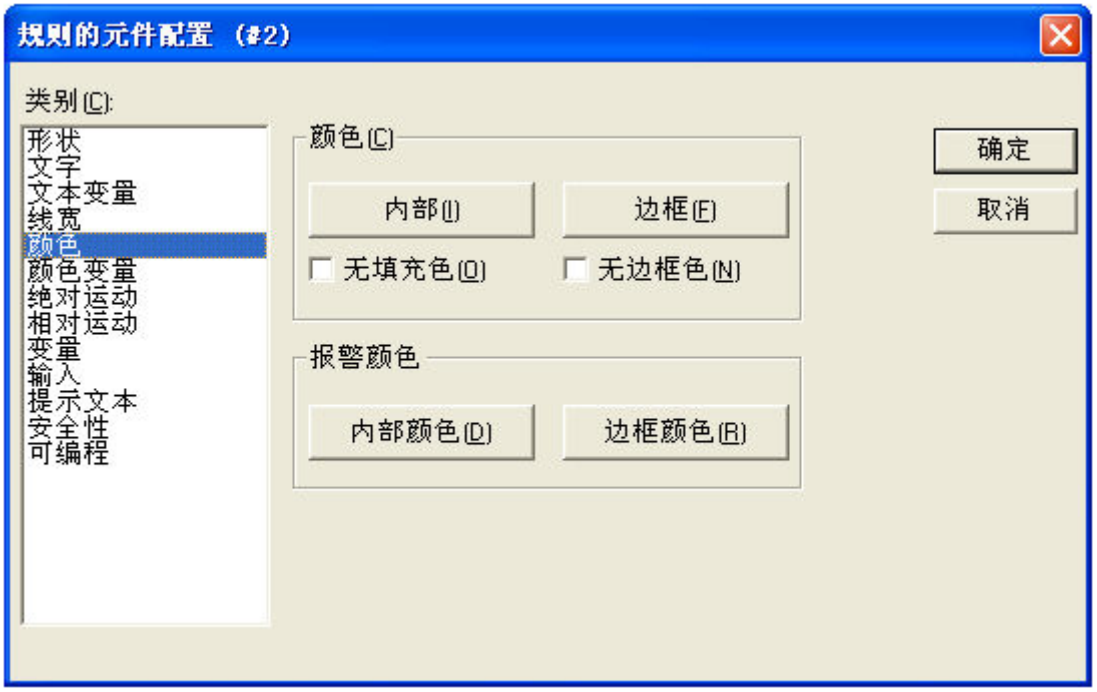
双击圆，打开编辑可视化元素的对话框。

在类别中选择变量，在区域中改变颜色中输入变量名.L1_red 或"L1_red"。当全局变量 L1_red 为 TRUE（真）时颜色会发生改变，变量名前面的小点表示了它是一个全局变量，但它不是强制的。



然后选择颜色，在 Inside 按键上单击，选择一个中性颜色例如黑色。

现在在警告颜色区域上单击 Inside 按钮，选择比较接近红灯的红色。



完成的圆将全为黑色，当 TARFFICSIGNAL1 中的变量 RED 为 TRUE 时，它的颜色就会变为红色，我们为第一个 TRAFFICSIGNAL 创建了第一个灯。

其它的交通灯

现在输入命令'编辑''复制'然后使用命令'编辑''粘贴'两次，将得到与上面第一个灯一样尺寸的两个圆。也可以按住鼠标左键拖动它，在这个例子中，它的理想位置是编辑窗口的左边的垂直列。在两个圆中的任意一个上双击，打开配置对话框，为下列变量输入相应的颜色。

中间的圆：L1_yellow

最下面的圆：L1_green

现在在‘颜色’选项中为圆设置，在‘报警颜色’区域中选择相应的颜色（黄色 或 绿色）

交通信号灯的例子

现在执行命令“插入”一个“长方形”,和插入圆一样，它包围着三个圆,为矩形选择尽可能一个中性的颜色，并执行命令“附加” “置于后面”,圆又重新显示出来。

如果模拟模式没有打开，可以通过命令'联机''仿真'激活它。

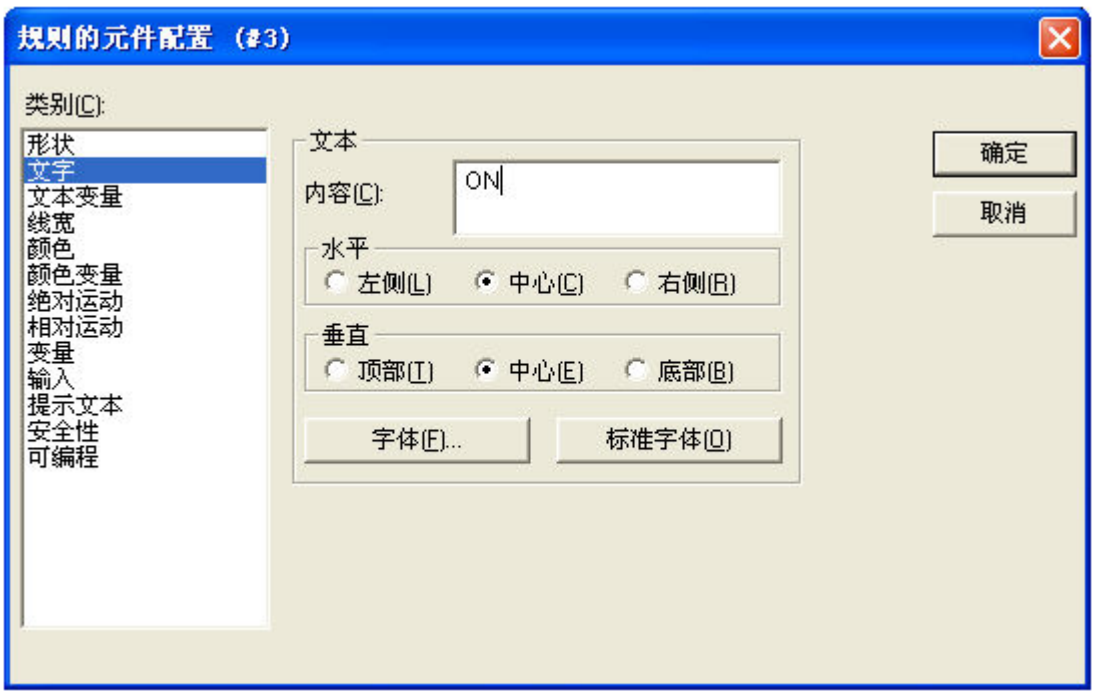
如果通过命令'联机''登录'和'联机''运行'已开始模拟，那么你会观察到第一个交通信号灯的颜色变化。

第二个交通信号灯

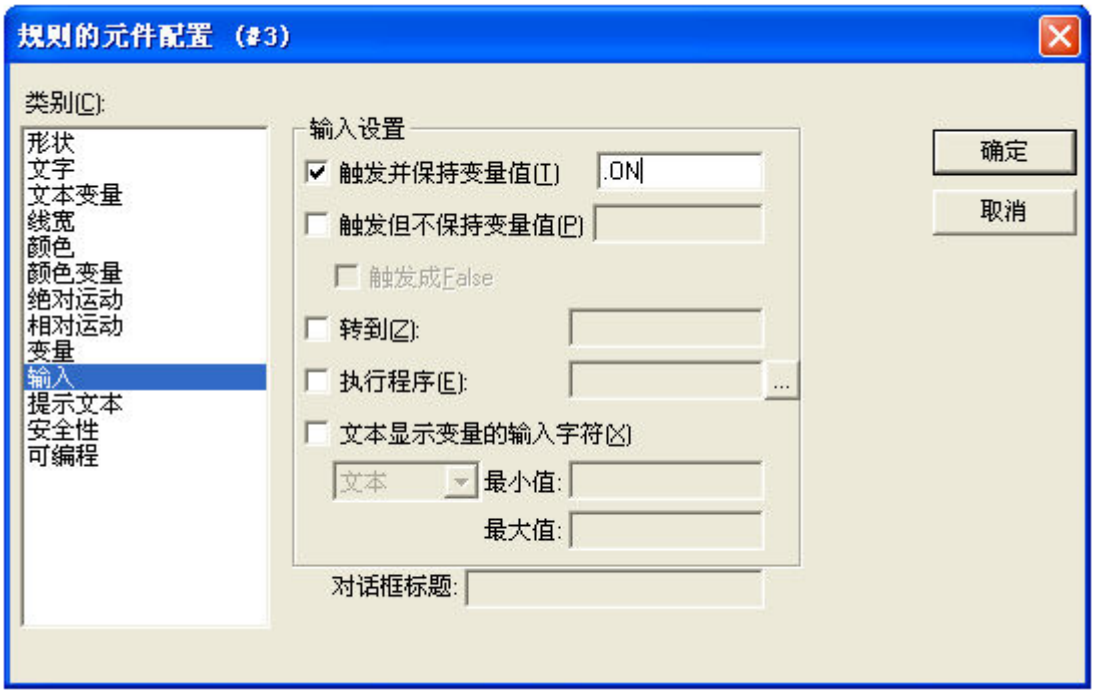
最简单的方法是拷贝第一个信号灯的全部元素来创建第二个信号灯，选择第一个灯的全部元素并用“编辑”“复制”命令来复制它们。然后把 TARFFICSIGNAL1 改为 TARFFICSIGNAL2,第二个交通灯的可视化也已经完成。

ON开关

插入一个矩形并给它选择一个颜色,在‘变量’类别中的'改变颜色'区域中输入.ON。在文本中的'内容'区域输入“ON”



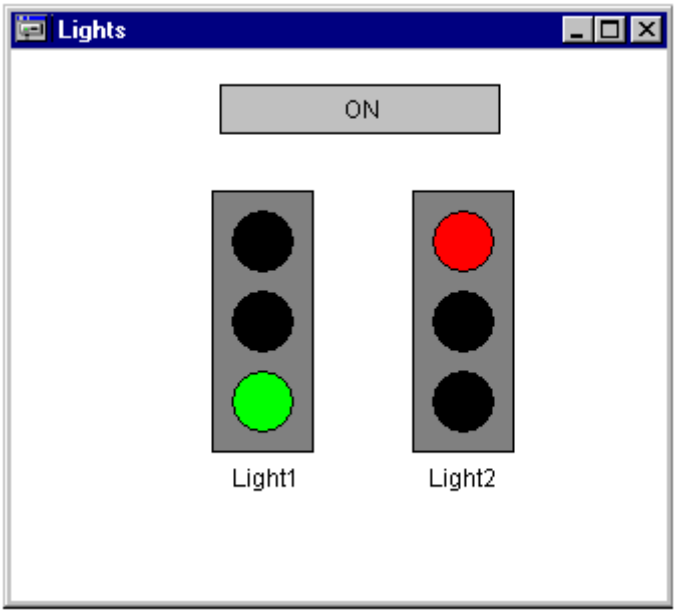
为了给变量 ON 设置 TRUE，用鼠标单击开关。激活“Input”选项中“触发并保持变量值”，并在这里输入“.ON”，变量键值的意思是当鼠标在可视化元素上单击时，变量.ON 设置为 TRUE，当鼠标释放以后它的值又复位为 FALSE（我们在这里为四个交通灯程序设置了一个简单的开关 ON）。



可视化中的字体

为了完成可视化，在交通信号的下面插入两个矩形。

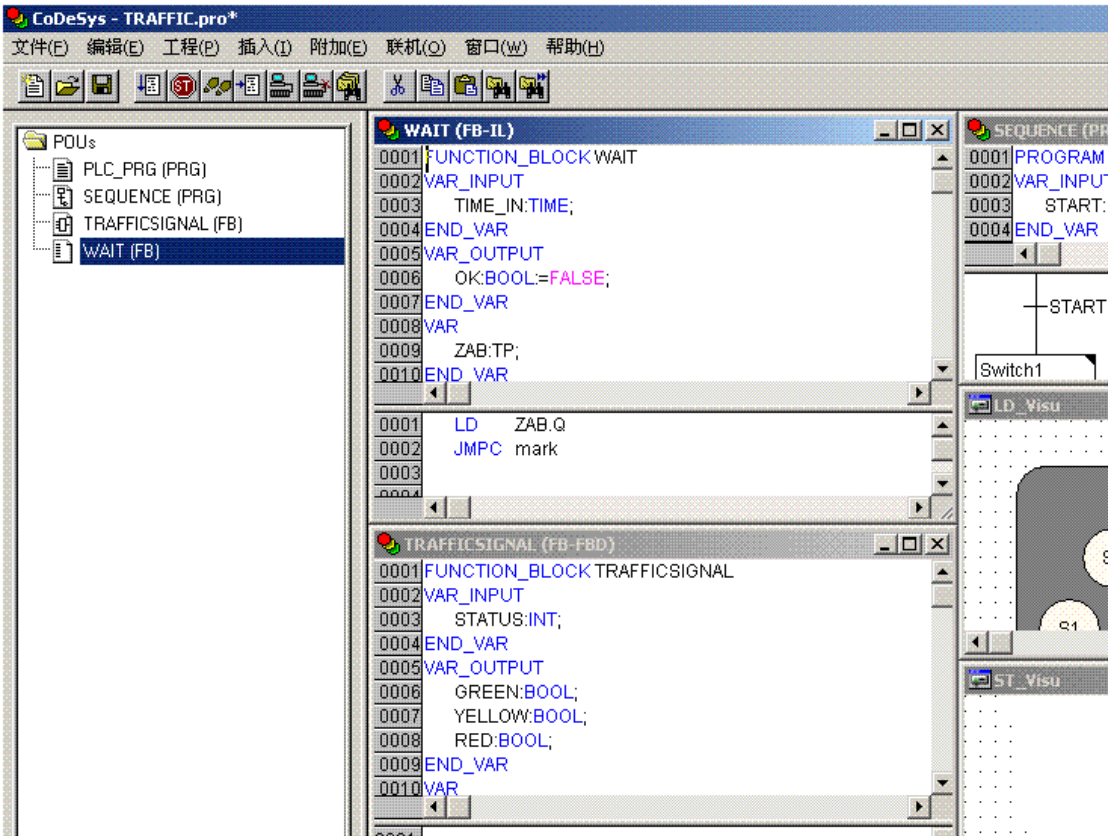
在可视化对话框，在‘颜色’类别中将‘边框’颜色设置为白色。在‘文本’类别中‘内容’区域输入 Light1 或 Light2，可视化界面如下：


Lights
ON
Light1
Light2

4. 各个单独的组件

4.1 主窗口

主窗口的组成



CoDeSys 的主窗口从上到下包括以下组件

菜单

工具栏（可选择）：快速选择菜单的命令的按钮

带程序组织单元、数据类型、可视化和资源选项卡的 对象管理器

在对象管理器和CoDeSys工作空间之间的垂直 屏幕分隔器

用于包含编辑器窗口的 工作空间

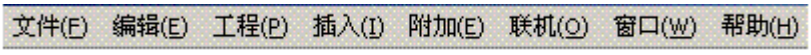
信息窗口（可选择）

状态栏（可选）；显示当前工程文件的状态的信息

以上内容也可参看：上下文菜单

主菜单

菜单条位于主窗口的最上边，它包含了所有的菜单命令。



工具栏

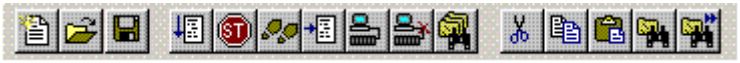
通过用鼠标在符号上单击，就可以更迅速的选择一个菜单命令，所选用的符号能自动的出现于活动窗口。

当鼠标键按在工具按钮上单击，然后释放后，命令才执行。

如果用鼠标指针在工具按钮上停留几秒钟，在工具条中就显示按钮的信息。

为了能看到工具栏中每个按钮的描述，在帮助编辑器中选择你想要的信息，单击你感兴趣的工具栏符号。

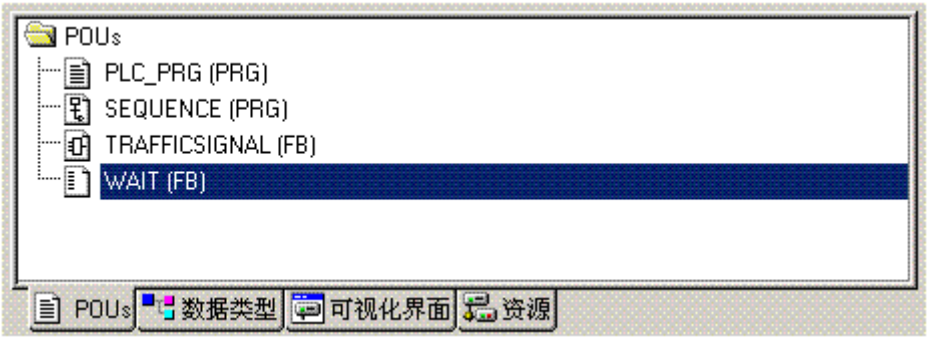
工具栏的显示是可选择（参看“工程”“选项”中“桌面”部分的说明）



对象管理器

对象管理器通常位于CoDeSys的左边,在底部有四个选项卡它们是  **POUs，**  **数据类型，**  **可视化界面**和  资源 。单击或使用键盘的左或右箭头能在相应的选项卡之间变换。

在管理对象章节中你会学到在 对象管理器中怎样使用对象。



屏幕分割器

屏幕分割器是指两个非重叠窗口之间的边界。在CoDeSys中，在主窗口的 工作区和 对象管理器之间有一个屏幕分割器。在界面（声明部分）和程序组织单元的执行（指令部分）以及工作区和 信息窗口之间都有一个屏幕分割器

你可以用鼠标指针来移动屏幕分割器，按住鼠标的左键来拖动到合适的位置。

在窗口尺寸发生变化时，应确保屏幕分割器的绝对位置。如果屏幕分割器好象不见了，那么只要放大你的窗口就会重新出现。

工作区

工作区位于CoDeSys中主窗口的右边，在这个区域可以打开对象的所有编辑器和 库文件管理器。当前的对象名显示在标题栏中，在它后面的括号中显示POU中一个缩略的POU类型和当前使用的编程语言。

在编辑器章节将详细讲述 编辑器的功能。

在“窗口”菜单下，可以得到所有 窗口管理命令。

信息窗口

消息窗口位于主窗口的 工作区的 屏幕分割器的下面。

它包含了所有来自先前的编译、检查或比较的信息。搜索结果和交叉引用列表也能从这里输出。

如果在消息窗口的消息上双击鼠标或者是按回车键，编辑器将打开这个对象，对象的相应的行被选中。用命令’编辑’’下一个错误’ 和“’编辑’’前一个错误’ ，能迅速的在错误消息之间跳转。

状态栏

CoDeSys 的主窗口的窗口框架底部的状态栏给出了当前工程文件和菜单命令的信息。

如果一个项目激活，那么它的名称将在状态栏的右边以黑色字迹显示，否则以灰色字迹显示。

当你在联机模式时， Online 以黑色字迹显示状态栏中，在离线模式下，它以灰色字迹显示。

在联机模式下，可以从状态栏中看到是否在 仿真模式（SIM）、程序是否在运行（RUNS）、是否设置了 断

点（BP）或是否有 强制赋值（FORCE）。

在 文本编辑器中，可以显示鼠标指针的行和列的位置，（例如， 行:5,列.:11）。联机模式下‘OV’在状态栏中显示为黑色，按〈Ins〉键在覆盖和插入之间转换。

如果鼠标指针是可视化的,当前鼠标指针的 X 和 Y 位置将以与屏幕左上角相关的象素表示,如果鼠标指针是在元素上,或元素正在处理中,那么将会显示出它的数值， 如果插入了对象,那么它也会显示出来（例如：矩形）。

如果你已经选择了一个菜单命令但是还没有执行它,那么在状态栏中会出现一个简短的描述。

状态栏的显示是可选择的（参看“工程”“选项”中“桌面”部分的说明）。

上下文菜单

快捷方式：<Shift>+<F10>

可以用鼠标右键来代替使用菜单栏执行命令，菜单包含了对象或活动编辑器常用的命令，根据激活窗口的不同出现不同的内容菜单。

4.2 工程选项

工程->选项

使用这个命令，可以打开设置选项的对话框。这些选项被分为不同的类别。可以使用鼠标点击，或者是使用箭头键，选择位于对话框左侧的你所要求的选项，或者对位于对话框右侧的选项进行更改。

在资源标签中的组件’工作空间’中，可以找到已经设定的当前工程文件选项信息。

在主窗口中，可以看到已经设定的相关信息。除非特殊指定，这些信息都存储在文件“CoDeSys.ini”中，并在下一次启动 CoDeSys 软件的时候，重新恢复。

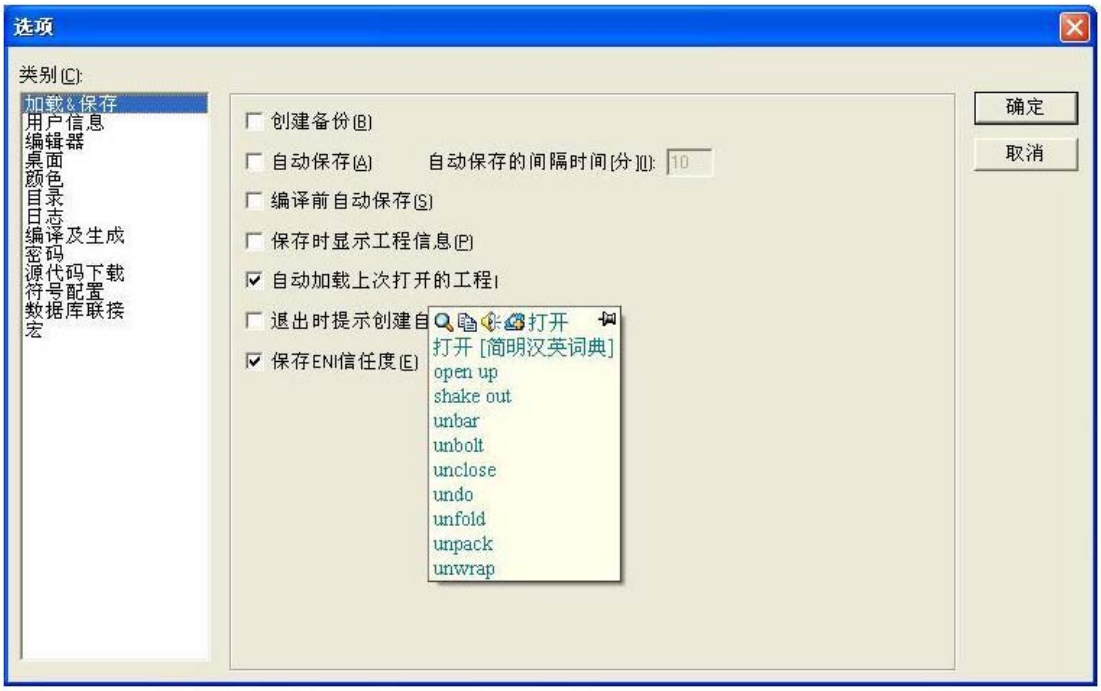
可以对以下的类别进行配置：

	stored in CoDeSys	stored in project
加载和保存	x	
用户信息	x	
编辑器	x	
桌面	x	
颜色	x	
目录	Cat. Common	Cat. Project
日志	x	
编译及生成		
密码		
源代码下载	x	
符号配置	x	
数据库连接		x
宏		x

’加载与保存’选项

在选项对话框中选择了这个选项将出现下面的对话框：

加载与保存选项对话框



当激活一个选项时，选项前面会出现一个√。

创建备份：CoDeSys 在每一次保存文件的时候，都创建一个扩展名".bak"的备份文件，和下面自动保存创建的扩展文件名*.asd 不同，这个扩展名为".bak"的文件在关闭工程文件以后一直保存着，因此你可以恢复最近保存的工程文件的版本。

自动保存：根据设定的时间间隔（自动保存间隔），工程文件将保存到扩展名为".asd"的一个临时文件中，这个文件在程序正常退出之后删除。任何情况下，CoDeSys 没有正常的关闭（例如，因为电源故障），那么文件将不会删除。当你再次打开文件的时候会出现下列信息：

此时，你可以决定是否想打开原始文件或自动保存文件。

编辑前自动保存：在每次编译之前，工程文件将自动保存，在这个过程中创建一个扩展名为".asd"的文件，这个文件的作用象上面在选项“自动保存”中讲述的一样。

询问工程信息：当保存一个新的工程文件或另存为一个新文件名字的时候，系统自动访问工程文件的信息，可以通过命令“工程” “工程信息”对文件信息可视化并处理它。

自动加载：在下次 CoDeSys 启动的时候自动加载最近打开的工程文件，可以在命令行中输入工程文件，以便在启动 CoDeSys 软件时，加载相应的文件。

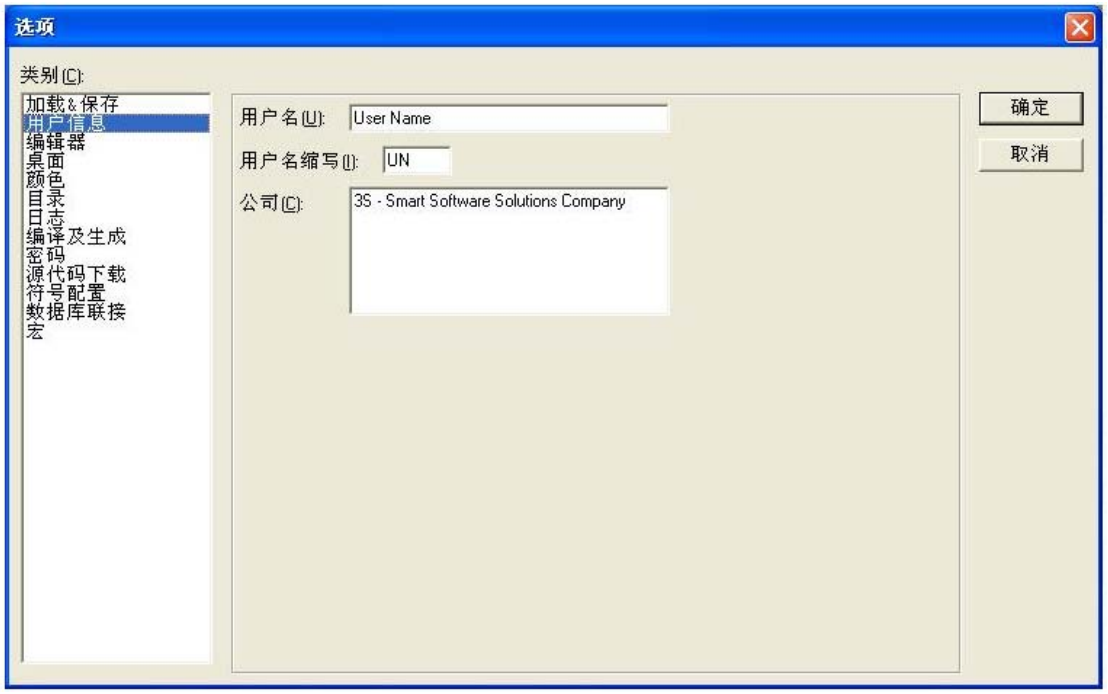
导入命令提醒：如果工程文件已经被修改并且没有创建新的导入文件，那么在用户退出工程之前，系统弹出一个对话框，提醒用户 “上次下载后没有创建导入工程文件。要退出吗？”

保存 ENI 信用度：用户名和密码，因为它们它们被插入到 ENI 数据库的登录对话框中，并与工程文件一起保存。

’用户信息’选项

如果在 选项对话框中选择了这个类别，就会出现如下的对话框：

用户信息选项对话框



在这个对话框中，用户名、名字的大写字母缩写和他的工作的公司都属于用户信息，每个条目都可以进行修改，这些设置将会应用到任何在本地计算机上由 CoDeSys 创建的工程文件中。

’编辑’选项

如果在 选项对话框中选择了这个类别，就会看到以下的对话框：

可以为编辑器进行下列的设置：

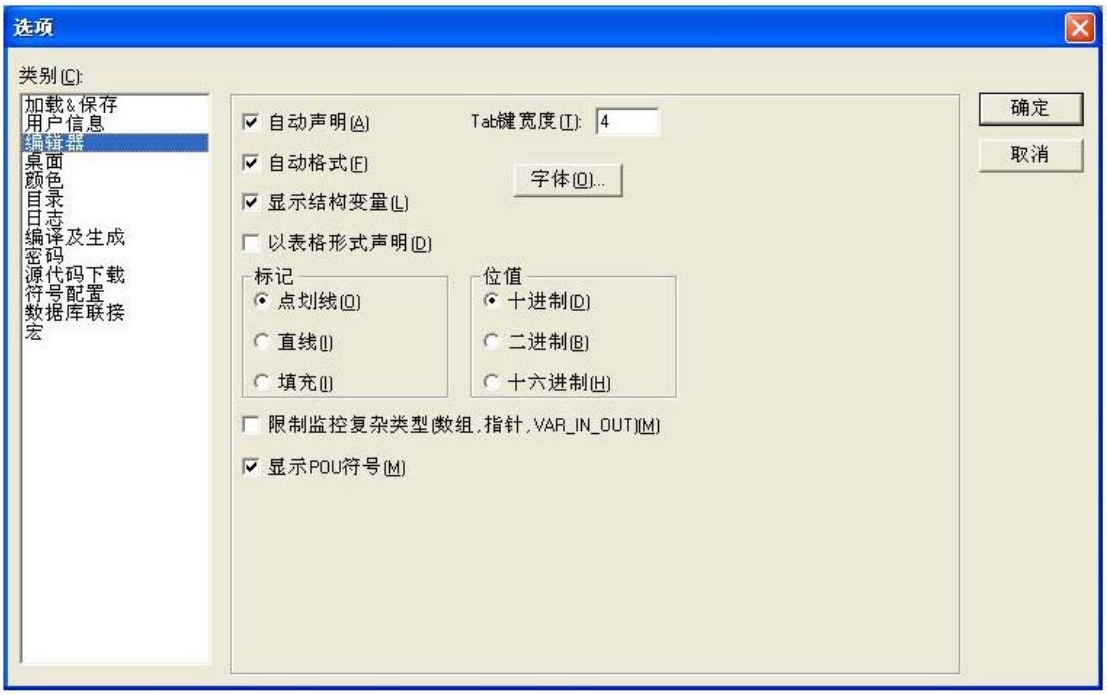
自动声明：如果激活这个选项，在输入一个没有声明的变量后，所有的编辑器中出现声明所需变量的对话框。

自动格式：如果激活这个选项，那么CoDeSys软件将自动对 指令表语言和 声明编辑器进行格式化处理。当你完成一行时，产生下列的格式：

1. 以小写字母写的操作数以大写字母显示；
2. 制表符插入到列中。

显示结构变量：如果激活这个选项，系统可以提供智能化的输入帮助功能。如果在一个应该插入工程符号的地方插入一个小圆点，系统就会自动打开一个选择列表。列表中提供了工程用到的所有全局变量。如果你插入了功能模块实例的名称，那么你会得到实例功能模块的所有输入和输出的列表。在编辑器中、在观察和接收管理器中、在可视化中和在采样追踪中都可以使用这个智能化的功能。

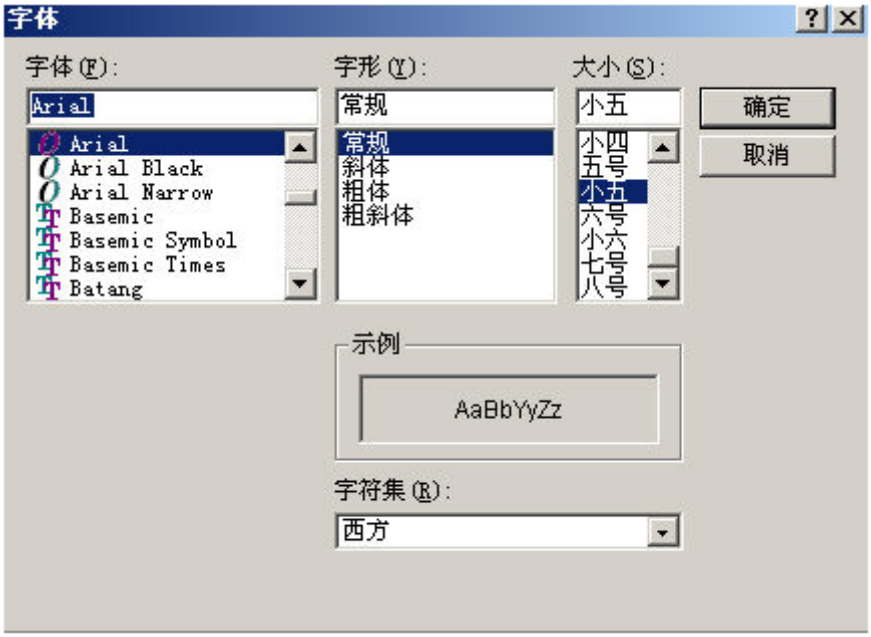
编辑器选项对话框



以表格形式声明:如果激活这个选项，可以在表格中编辑变量而代替使用通常的声明编辑器，这个表格象一个纸牌盒那样存储，在这个纸牌盒中，你可以找到用于输入变量、输出变量、局部变量和输入输出变量的各种符号。对于每一个变量，都有编辑区进行名称，地址，类型，大写和注释等内容的编辑。

Tab 键宽度：在编辑器中按 Tab 键后跳过的字符数，默认的设置是四个字符，字符的宽度依赖于所选择的字体。

字体:在对话框中编辑类别中字体按钮上单击，可以在所有的 CoDeSys 编辑器中选择字体，对所有的制图操作来说字体的大小是基本的单元，选择一个大字体尺寸能放大字体。在输入命令之后，出现选择字体、字型和字号的字体对话框。



’桌面’选项

如果在选项对话框中选择了这个类别，将会出现如下的对话框：

当激活一个选项，在它的旁边出现一个 √。

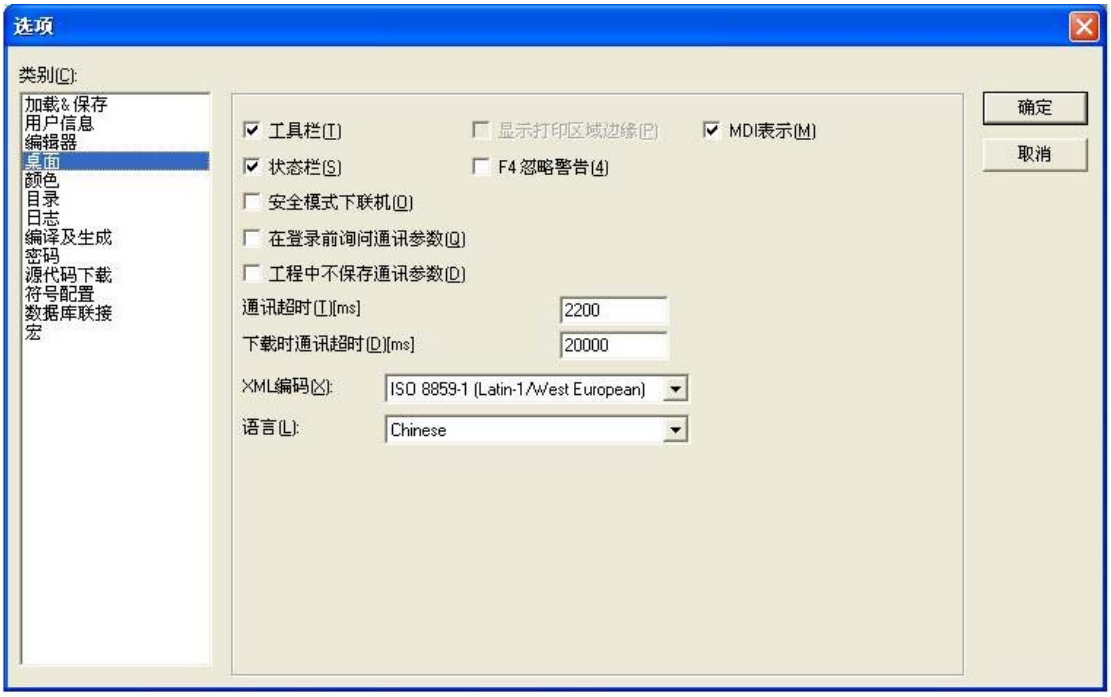
工具条：在菜单栏下面的快速选择菜单命令的工具栏的按钮变为可视。

状态条:在 CoDeSys 主窗口中底部边界的状态栏变为可视。

安全模式下的联机：在联机模式下，有以下命令：’运行’，’停止’，’复位’’设置断点’，’单循环’，’写入新值’,’强制新值’ 和 ’强制赋值’，并出现一个带有确认请求命令是否确实执行的对话框。如果目标系统支持，当你想从设计系统加载一个实际工程到PLC中，并且这个实际工程已经存在时，就可得到一个扩展的对话框。这个对话框既可显示已有工程的工程信息，又可显示当前要加载的工程的信息。当在PLC中已经有一个工程，再创建一个导入工程时，也要用到这些有关工程的信息。

这个选项和工程一起保存。

桌面选项对话框



在登录前查询通讯参数：只要执行了命令“联机”“登陆”，将首先打开通讯参量对话框，为进入联机模式，必须首先通过 OK 选择这个对话。

工程中不保存通讯参量：通讯参量的设置对话（“联机”“通讯参数”）将不会和工程一起保存。

显示空白区域边缘：在每个编辑窗口，当前设置打印范围的界限用红色虚线标记。它们的尺寸依赖于打印机的特性（纸张大小，方向）和打印版面的“内容”区域的大小。（“菜单” “文本设置”）。

F4 忽略警告：在编译之后，当在一个消息窗口按下 F4 时，就会跳到有错误信息“警告信息被忽略”的行中。

MDI 表示：这个选项默认情况下是激活的，因而能同时打开几个窗口。如果选项是无效的（单文档模式）只有一个窗口能打开并全屏显示。例外情况：一个程序的动作和程序自身能在单文档界面模式下并排显示。

语言：在这里定义菜单和对话框文本中显示的语言。

请注意：在 windows98 下不能进行语言选择。

’颜色’选项

如果在 选项对话框中选择了这个类别，将会出现如下的对话框：

颜色选项对话框

可以对 CoDeSys 中的默认的颜色设置进行编辑。你可以选择为行号码（默认预设置：浅灰色）、为断点位置（黑灰色）、为设置断点（浅蓝色）、为当前断点（红色）、到达位置（绿色）来改变的颜色设置，或者为布尔变量值的监视（蓝色）而改变颜色设置。

如果你已经选择了一个显示的按钮，将会打开选择颜色的对话框。

'目录'选项

如果在 选项对话框中选择了这个类别，将会出现如下的对话框：

目录选项对话框

可以在工程和公共区域为 CoDeSys 输入目录（例如编译文件是图和列表文件，而不是符号文件。后者将存储在工程目录），以便进行库文件和控制器配置文件的查询，或者查找存储编译和源上传文件。如果你点击一个区域后面的...按钮，将打开目录选择对话框，对库和配置文件，可以输入多个路径，彼此之间用分号间隔开。

请注意：通过使用前缀. ，库文件路径可以基于工程文件路径上输入。例如输入".\libs"。如果当前工程是在'C:\programs\projects\libs'中,库文件将会从'C:\programs\projects\libs'中查找。可以参看“插入”“添加库文件”

请注意：不要在目录路径中使用空格和除了"_"之外的特殊字符。

工程区域里的信息和工程一起保存，在公共区域的信息写到编程系统的 ini 文件中并应用到所有的工程。

目标区域显示了在目标系统中设置的库文件和配置文件的目录。例如，这些区域是不能编辑的，但是条目能被选择和复制（右鼠标键内容菜单）。

CoDeSys 通常首先在“工程”中输入的目录中查找，然后在“目标系统”（在目标文件中定义），最后在“公共”区域查找。如果找到两个同名文件，将使用在目录中首先找到的文件。

日志选项

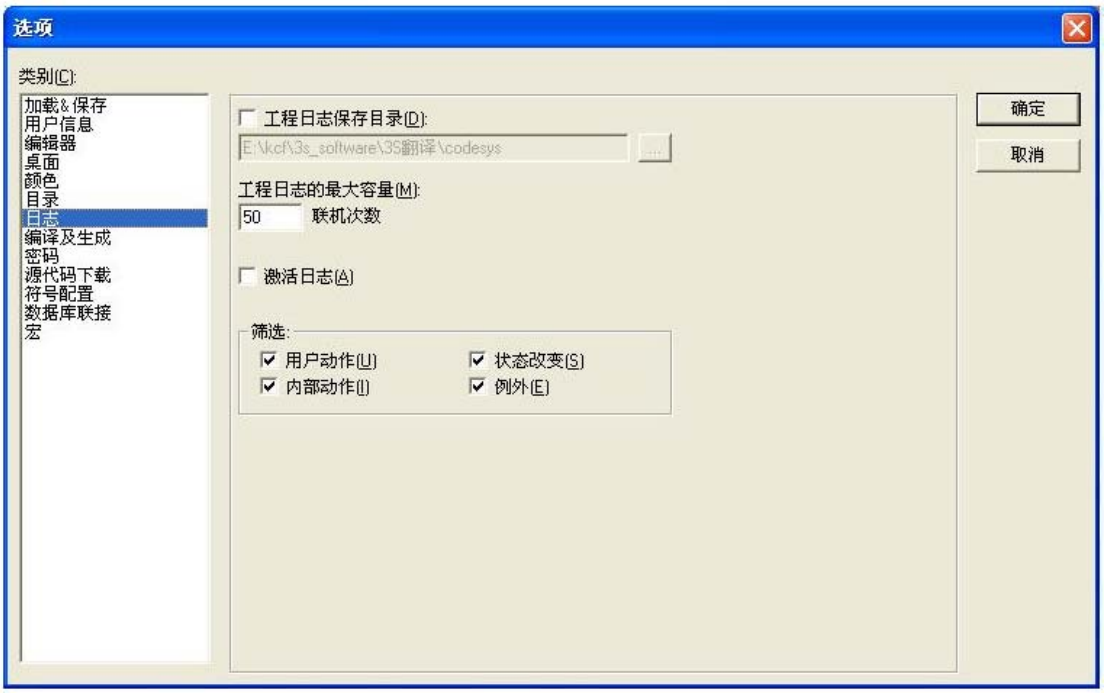
如果在选项对话框中选择了这个类别，将会看到如下的对话框：

在这个对话中，配置工程的日志文件。在联机模式过程中它记录所有用户动作和内部过程。

如果一个现有的工程没有日志文件，将会打开一个对话框，让你注意到一个日志现在正在建立，它将接收在下个登录过程中首次输入。

当保存工程时，日志文件在工程目录中自动存储为二进制文件。如果你喜欢存放在一个不同的目标目录，可以激活选项　工程日志目录:在编辑区域输入适当的路径。为达到这个目的使用按钮  来访问“选择目录”对话框。

日志选项对话框



日志文件自动分配工程的名字再加上扩展名‘.log’，联机阶段的最大次数由最大工程日志大小决定。如果在记录过程中超过了这个数字，系统将删除旧的条目为新条目让出空间。

通过选项中的激活日志区域可以控制日志功能的开启或关闭。

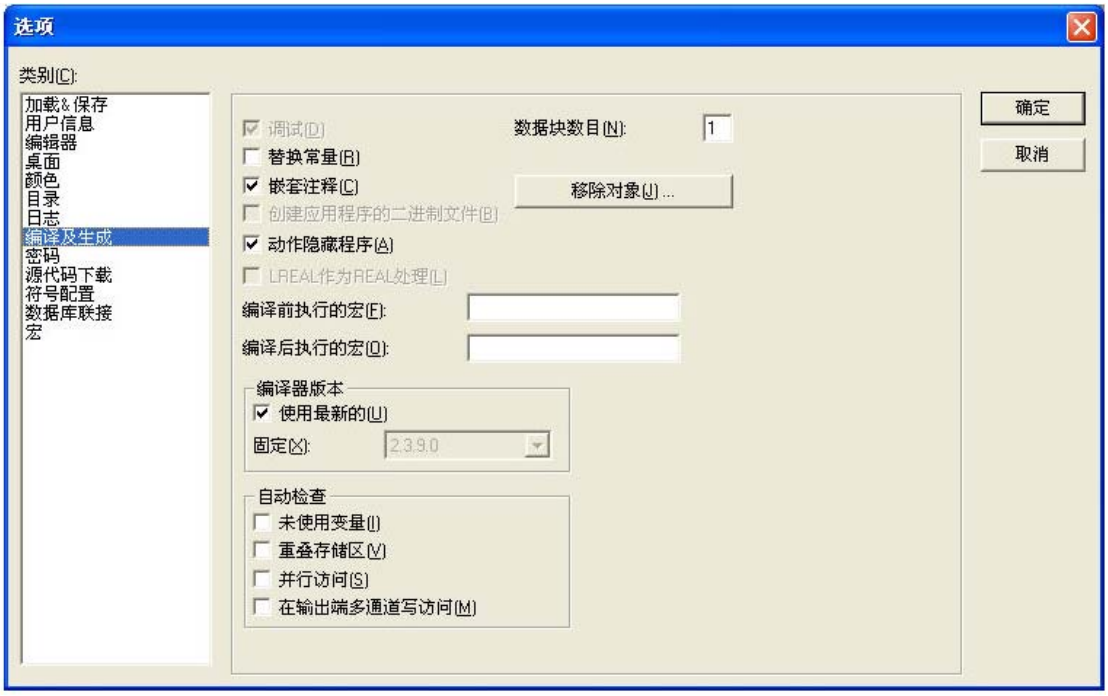
可以在Filter区域选择记录哪些动作：用户动作、内部动作、状态改变、例外，只有在这里选中的属于目录的动作才会出现在日志窗口并记录到日志文件中（请参照 日志来了解关于目录的更多的知识）。

可以通过命令’窗口’’日志’来打开日志窗口。

’编译及生成’选项

如果选择 选项中的’生成‘选项，将显示下面的对话框。

编译生成对话框



调试：取决于用户在目标描述中是否激活此项。如果它被激活，将会创建附加的代码，这些代码相当大，

为了充分使用 CoDeSys 提供的调试功能（例如，断点）需要使用这些代码。当你关闭这个选项，工程处理将加快并且代码的大小减小，这个选项和工程一起保存。

替换常量：每一个常量都直接加载，在联机模式下常量显示为绿色。如果这个选项是未激活的，不能对常量进行 强制, 写 和 监控 。通过变量访问把这个值加载到存储区（这实际上允许写变量值，但意味较长的处理时间）

Nested comments（嵌套的注释）：注释可以放在其它的注释里面。例如：

```
(*
a:=inst.out; (* to be checked *)
b:=b+1;
*)
```

这里注释从第一个括号开始，没有在"checked"后的括号结束，而是在最后一个括号结束。

创建应用程序的二进制文件：在编译的过程中在工程目录中创建一个文件名：<project_name>.bin二进制代码文件（导入工程）。通过命令'联机' '创建引导工程'在控制器上建立导入工程。

注意：如果一个已有的工程是打开的，这个工程是由先前版本的 CoDeSys 创建的，选项将会失效。但是先前有效的层次（局部变量在全局变量之前在程序之前在局部动作之前）能够保持。

数据块数目：这里可以为 PLC 中的工程数据定义应该分配多少内存段。即使是新变量加入，这个空间也能保证使联机交换成为可能。如果在编译过程中，得到信息“超出全局数据内存……”，输入一个较高数字，此时这个局部程序变量会当作全局变量来处理。

移除对象：这个按钮打开对话从结构中排除：在工程组件的树形图中选择那些在编译过程中不被考虑的 POU，激活选项排除，在这里排除的 POU 在选择树上将会显示为绿色。如果你只想显示在程序中当前使用的那些 POU，按按钮排除不使用的。

编译器版本:这里说明将要用到的编译器版本, CoDeSys 在 V2.3.3 后的版本除了包含目前的编译器版本，还包含了先前的版本。如果你想在任何情况用当前的版本编译工程，激活选项最近使用。如果工程需要用一个特殊的版本编译，通过在装置的选择列表中定义它。

自动检查:为了在每个工程的编译时得到语义上的正确性，可以激活下列选项：

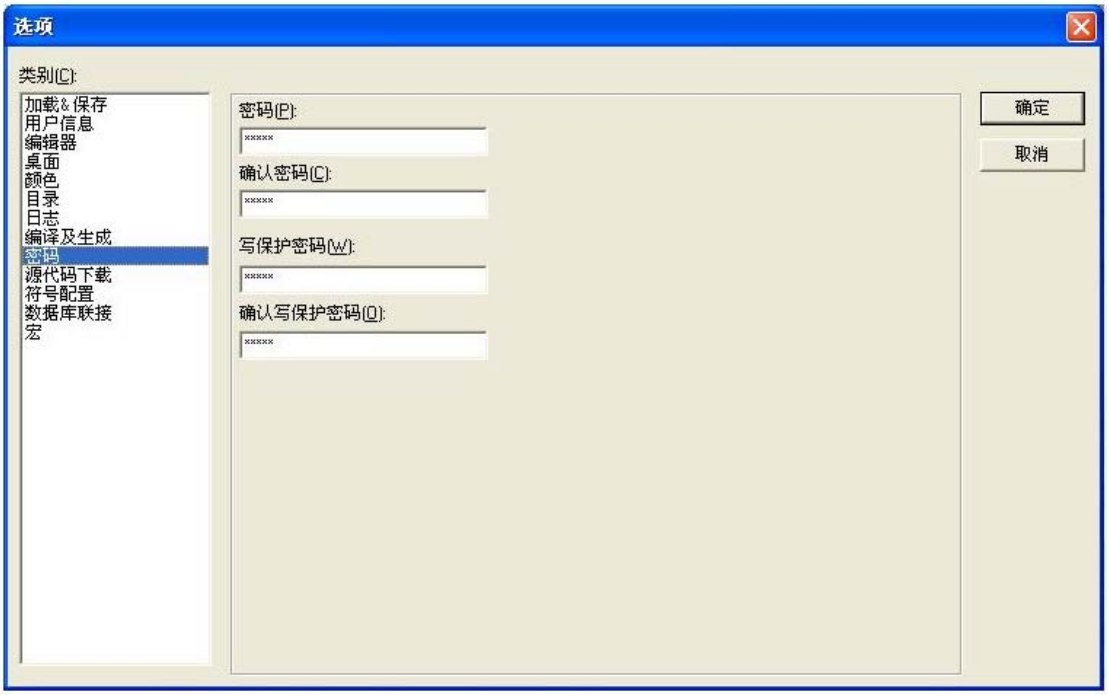
- 未用的变量
- 重叠存储区
- 并行访问
- 在输出端多通道写访问

结果将会在消息窗口显示，这些检查也能通过菜单“工程”的子菜单“检查”的命令激活。

密码选项

如果在 选项对话框中选择了这个类别，将会出现如下的对话框：

密码选项对话框



为了保护你的文件不受非授权的访问，CoDeSys 提供了密码选项来保护文件的打开和更改。

在密码区域输入你期望的密码。对每个敲入的字符在区域中出现一个*。在确认密码区域中必须重复输入相同的字符，按 OK 键关闭对话框。如果出现消息“密码和确认密码不相符”，那么在两个条目中的一个中输入有错误,在这种情况下在重复在两个条目输入直到关闭时不出现消息为止。

如果你现在保存文件，然后重新打开它，你会看到一个要求你输入密码的对话框，只有在正确输入密码的情况下才能打开工程，否则，CoDeSys 报告“输入密码不正确”。

随着打开文件，你也可以用密码来保护你的文件不被修改。你必须在写保护密码区域设置一个密码并在下面的区域确认。

当打开一个文件时，如果 CoDeSys 要求输入写保护密码，如果按退出按钮，可以不需要密码打开一个写保护的工程。现在你可以编译工程，加载到 PLC 中，模拟，等等，但是不能改变它。

当然记住两个密码是很重要的，但是，如果忘记了一个密码，可以联系你的 PLC 制造商。

密码和工程一起存储。

为了创建不同访问权限你可以定义用户组和 用户组密码。

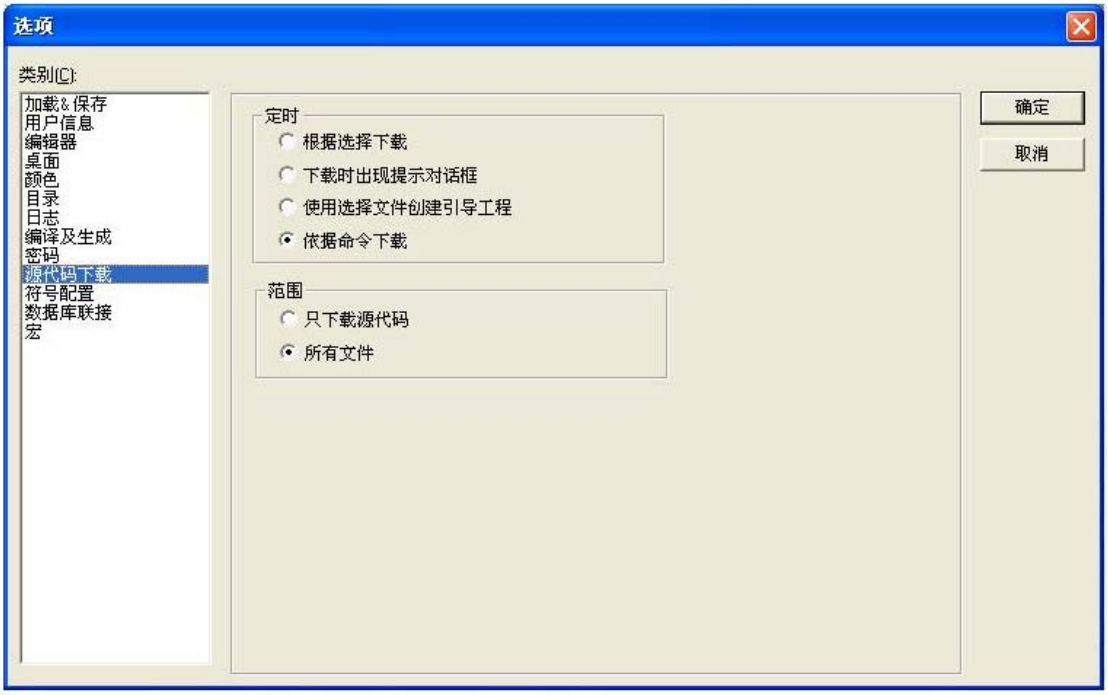
另外可以通过加密的方式保护一个工程，例如如果不正确输入密码，此库不可使用。

注意:设置保存在工程中。

’源代码下载’选项

如果在选项对话框中选择了这个类别，将会出现如下的对话框：

源代码下载选项对话框



可以选择定时或者多大范围，把工程文件加载到控制器系统。选项只下载源代码只包含 CoDeSys 文件扩展*.pro 的文件，选项全部文件也包含相关库文件、可视化位图、配置文件等等。

选项根据选择下载允许在使用命令“联机” ’下载’时自动加载被选择的文件范围到控制器系统中。

选项根据选择文件创建引导工程允许在使用命令“联机” “创建引导工程”时自动加载被选择的文件范围到控制器系统。

选项下载出现提示对话框提供一个对话，当命令“联机” ’下载’是给定，将出现“你想写源代码到控制器系统吗？”。按是按钮将会自动加载被选择的文件范围到控制器系统，或者你可以选择No来放弃。

若使用选项需求，则必须通过命令’联机’’源代码下载’加载被选择的文件范围到控制器系统中。

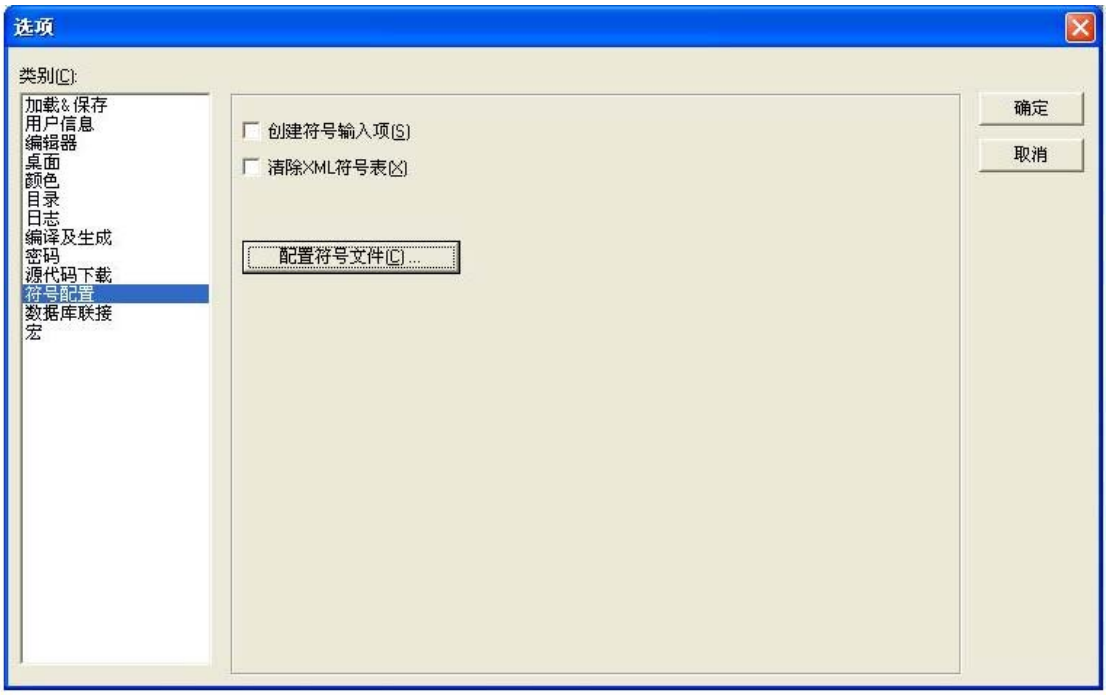
在控制器系统存储的工程可以通过使用命令 ’文件’ ’打开’从PLC中重新打开，文件在处理过程中解包。

注意:设置保存在 CoDeSys 中。

“符号配置”选项

这里提到的对话框是用于配置符号文件。这会在工程目录中创建文本文件<工程名称>.sym，二进制文件<project name>.sdb（依赖于在用的网关版本），符号文件是为通过符号接口和控制器进行数据交换和用作那个目的，例如，网关 DDE 服务器。

符号配置选项对话框



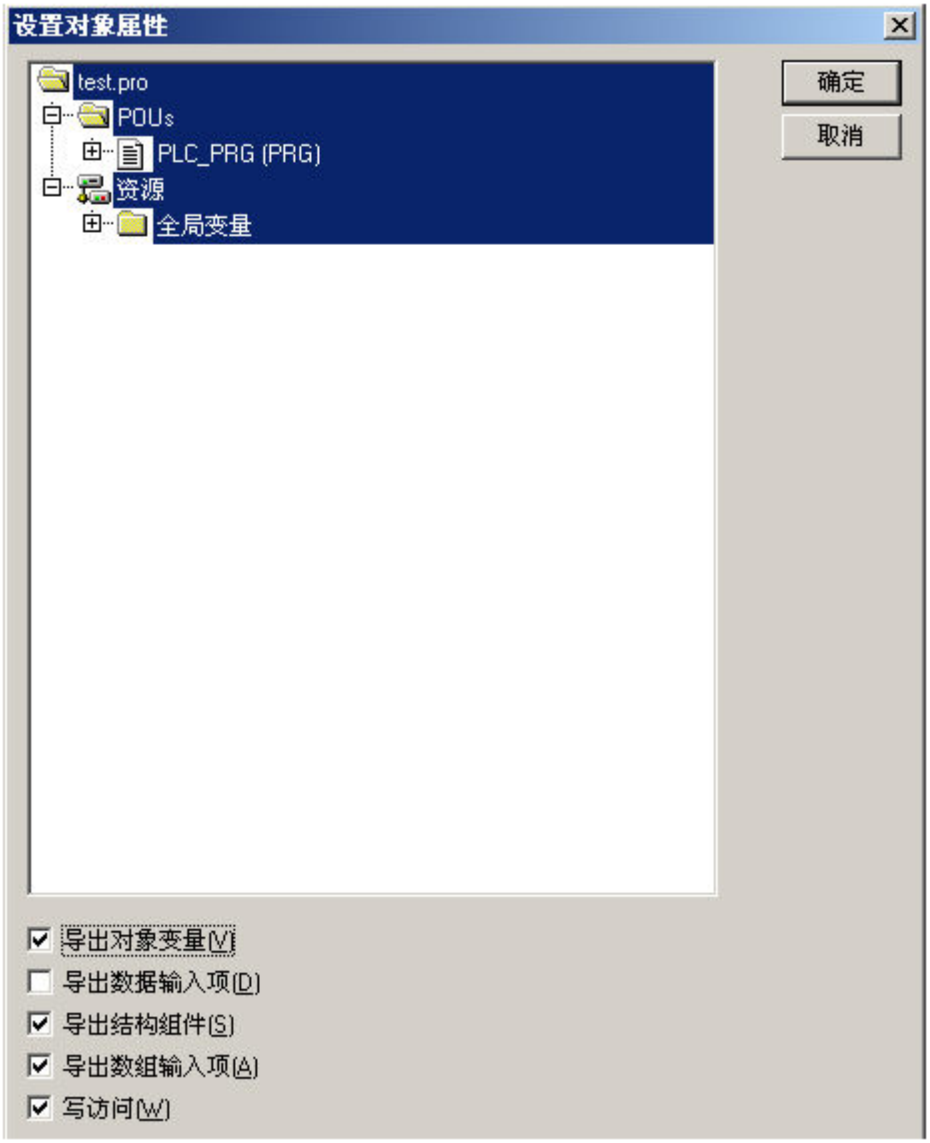
如果激活选项创建符号条目，那么工程变量的符号条目在工程的每个编译时在符号文件中自动创建。

如果激活附加的选项创建 XML 符号表格，那么在工程目录中也创建一个包含符号信息的 XML 文件，它被命名为<project name>.SYM_XML。

当配置符号条目时请注意以下信息：

•如果在目标设置中激活了选项'INI-file 符号配置'，那么符号配置将会从 codesys.in 中读出或在这里定义的其它的 ini 文件中读出（此时，在 CoDeSys 中不能编辑对话“设置对象属性”）

•如果选项“INI-file 符号配置”没有激活，将会产生与“设置对象属性”对话设置一样的符号条目，使用按钮设置符号文件能完成这些：



使用树状结构的选择编辑器来选择工程 POUs 和通过在相应的小框单击鼠标来设置对话框下面的期望的选项，选中的选项是激活的，可以设置下列选项：

导出对象变量:选择对象的变量在符号文件中导出。

下面的选项只有导出对象变量激活才起作用：

导出数据输入项:为对象的结构和数组访问全局变量创建的条目。

导出结构组件：为每个对象的结构变量组件单独的创建的一个条目。

导出数组输入项：为每个对象的数组的变量组件单独的创建的一个条目。

写访问：通过 OPC 服务器可能会改变对象的变量。

一旦完成当前选择的 POU 设置，其它的 POU 也能够在关闭对话之前被选中，并且打开一个选项配置，这能完成任意多 POU 的选择，一个接一个，当通过 OK 关闭对话框时，所有的配置从对话框打开开始应用。

注意:设置保存在工程中。

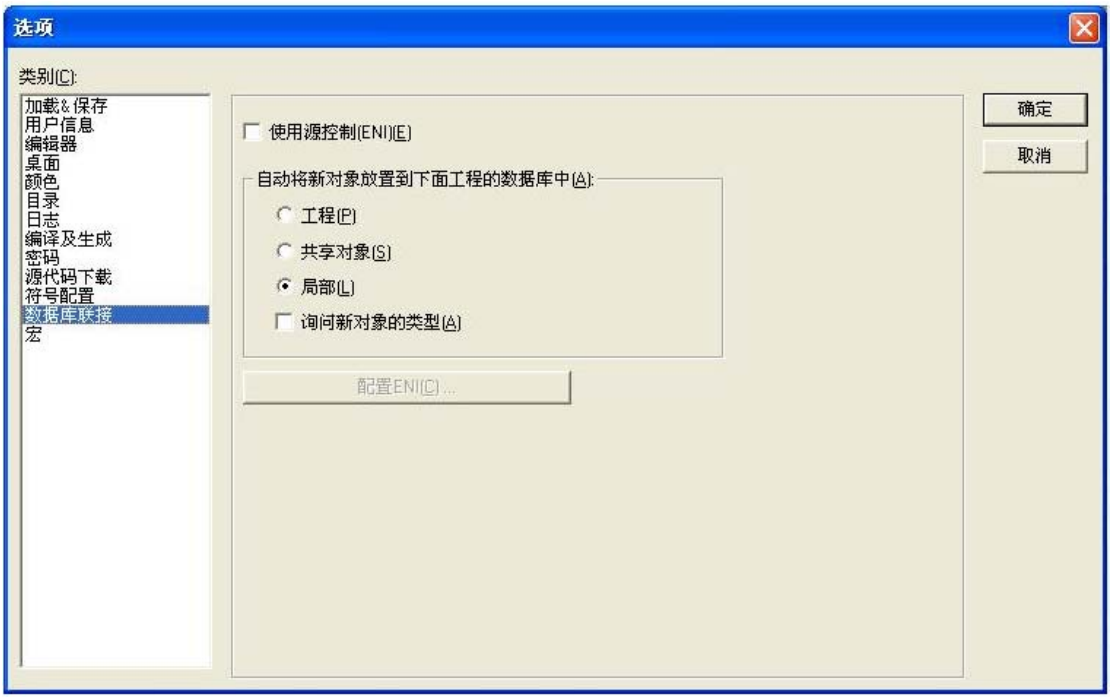
‘数据库连接’

这个对话可以用于定义是否在工程数据库中管理工程和相应地配置 ENI 接口。

使用源控制(ENI):用户如果想通过ENI服务器访问工程数据库来管理数据库中所有的或单个选择的POUs，激活这个选项，但前提条件是：ENI服务器和数据库必须已经安装并且必须在数据库中注册了一个用户。参照 ENI-Server文档 或 ‘The CoDeSys ’ENI’ 章节。

如果选项是激活的，那么数据库就能用来处理工程的POUs。一些数据库功能就象选项对话中定义那样自动运行。并且在菜单’工程’’数据基本连接’ 中能得到准确调用功能的命令，除此之外将会在对话属性中插入标号‘数据基本连接’。可以给POU分配一个特殊的数据库类别。

数据库连接选项对话框



用下面的数据库连接新对象：

这里建立一个默认：如果插入到工程中一个新对象，那么它将自动赋值给在这里定义的对象类别，这个赋值将会在对象属性对话中显示并且以后可以在这里修改，可能的赋值有：

工程：POU将存储在 ENI配置/工程对象对话中‘工程’区域定义的那个数据库文件夹中。

共享对象：POU将存储在 ENI配置/共享对象对话中‘工程’区域定义的那个数据库文件夹。

局部：POU 将不在 ENI 数据库中管理，但是只在本地工程中存储。

除了‘工程’ 和‘共享对象’这里还有针对这种直到工程编译时才创建的对象的第三类别数据库‘编辑文档’ ，这个类别与当前设置无关。

询问新对象类型:如果这个选项激活，那么无论何时加入到工程中一个新对象，对话’对象’’属性’都会打开。这里你可以为POU选择上面提到的三个对象类别，这样标准设置会被重写。

ENI 配置:这个按钮打开三个中的第一个 ENI 配置对话：

在 ENI 数据库中管理的每个工程对象，能赋给下列数据库类别中的一种：‘工程’，‘共享对象’ 或‘编辑文档’。这些类别可以使用各自的对话来定义它要存储的数据库文件夹。

ENI配置对话框 / 工程对象

ENI配置对话框 / 共享对象

ENI会话对话框 / 编辑文档

注意：每个对象和工程一起本地保存

如果是做第一个配置，对话会一个接一个打开，在这种情况下出现一个 Wizard 会引导你并且在第一个对话中输入的设置会自动的复制给其它的设置，如果你需要不同变量值，只需要仅仅修改它们。

如果你想修改一个存在的配置，那么三个对话会组合到一个窗口（三个标签）。

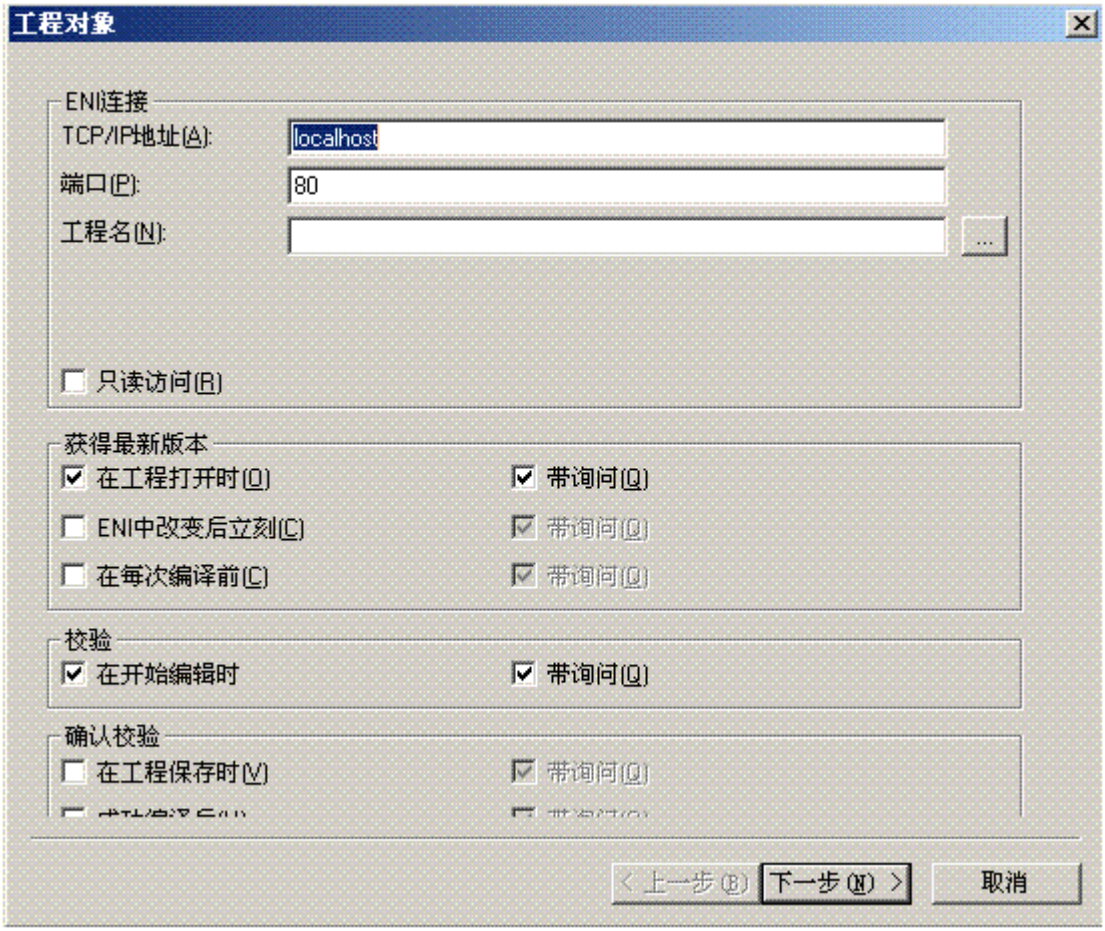
如果你以前还没有成功登录数据库，那么 登录对话框会自动打开。

注意:设置保存在工程中。

关于工程数据的工程对象和共享对象的选项

这些菜单是工程数据库选项('工程' '选项' '工程源控制')中的一部分.在这里你定义'工程对象' 和'共享工程'访问参数.两个菜单包括相同的条目.第三个菜单是访问数据库 编译文件的配置的变量.

工程源控制选项中的菜单'工程对象'



ENI 连接

TCP/IP 地址:	ENI服务器在计算机中运行的地址.
端口:	默认值:80;必须放置在 ENI 服务器的配置参数中
工程名称:	数据库文件夹被存储的名称.点击一个已经存在数据库工程的文件夹,进入'工程名称' 编辑区域.如果你没有登入ENI服务器直到你试着打开文件夹,那样你将要首先进入 登录菜单,必须输入'使用者姓名' 和'密码' 定义你的ENI使用帐号去访问三个数据库.
只读	如果此选项被激活,在定义的数据库文件夹上只能进行读取访问.

获得最新的版本

'获得最新版本' 数据库功能,菜单'工程' '数据库连接' 复制目前版本的POUs从定义的数据库文件夹到当前打开的工程,本地的变量将要被覆盖.若发现在数据库中有域工程不同的版本,则在被确定的时间到来是对于所有对象这将自动运行.选择选项来设置一个检查标记:

在工程打开时	CoDeSys 中工程打开的同时
ENI 变化瞬时	在数据库中一个新的 POU 版本被检查出来时(例如,通过其他的使用者),POU 将马上更新当前的工程并且适当的信息将出现
在任何编译前	CoDeSys 中任何汇编程序之前

校验确认

数据库工程'隔离' 为 POU 将要以'工作中'为标志并锁定其他的用户直到再用'登录' 或'撤掉校验确认'解除锁定.如果选项在开始编辑时及被激活则一个对象将被自动隔离直到你开始编辑它.如果对象已经被其他用户隔离,提示信息将出现.

校验

数据库功能'登记' 为一个新的对象版本将被创建在数据库中,老的版本将被取代.

你可以激活一个或全部以下两个选项来定义自动登记时间:

在工程保存时	工程保存时及起作用
在成功完成后	工程没有错误的完成后及起作用

'获得最新版本', '隔离' 和 '登记' 中每一个选项将被激活时将被询问.这样你就能在打开菜单时确认是去除它还是使用它

'共享对象' 菜单和'工程对象'菜单一样.设置申请所有的对象被分配到数据库'共享对象'中.

如果你做一个首要配置,配置菜单将会一个接一个的出现引导你完成(用点击下一步).第一个菜单中的设置将自动的传个 下一个.所以只要编辑必要的修改就可以了

取消将关闭当前没有保存的修改过的对话框.你将返回主菜单'选项' '工程源控制'.

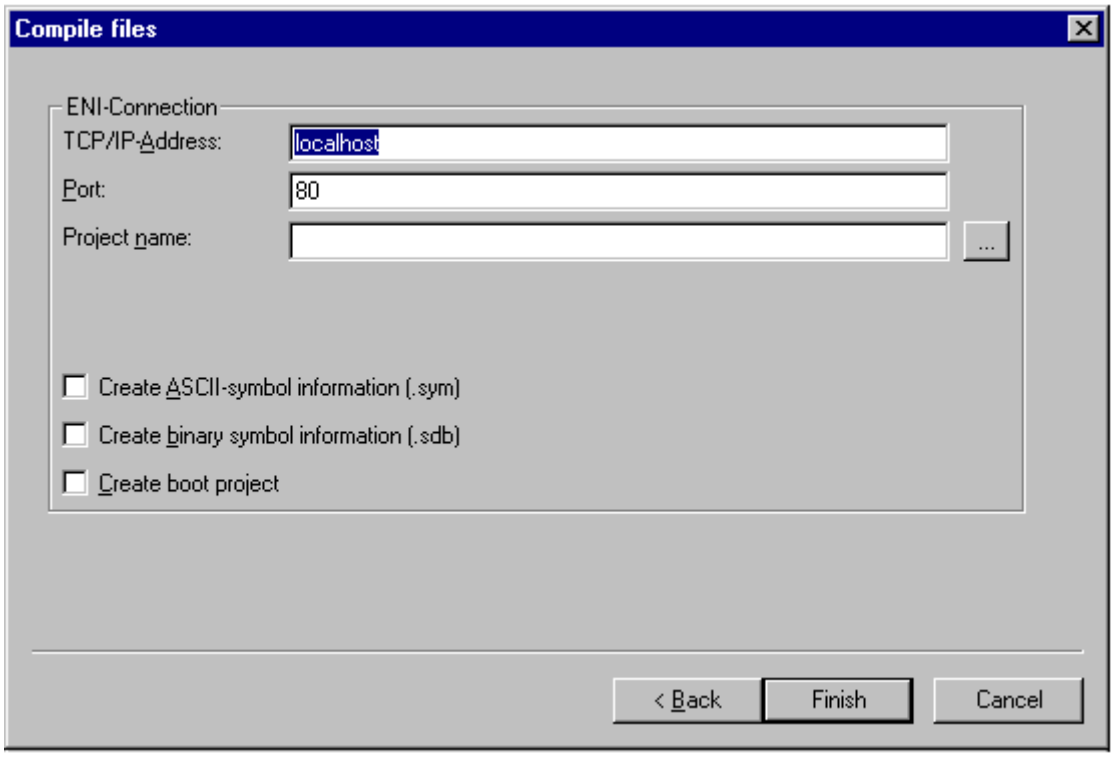
如果一个已有设置被更改,新的更改点击 OK 保存.然后菜单关闭返回 '选项' '工程源控制'.

注意: 设置将被储存在工程中

关于工程数据库编辑文件的选项

这个菜单是工程数据库选项中的一部分('工程' '选项' '工程源控制'). 这里你定义对象以什么种类'Compile files'运用的数据库中.

工程源控制中的菜单'编辑文件'



在TCP/IP地址输入域,端口,工程名称被用菜单 工程对象/共享对象来描述

创建 ASCII 符号信息(.sym)	如果选项被激活,符号文*.sym (文本格式)和*.sdb (二进制格式) 讲被创建,这个文件将被自动写入到数据库中.符号文

创建二进制符号信息(.sdb)	件像定义工程选项中'符号配置'一样被创建.
创建导入工程	如果选项被激活,一个导入工程将被创建,这个文件将被自动写入到数据库中.

如果你做一个首要配置,配置菜单将会一个接一个的出现引导你完成(用点击下一步).第一个菜单中的设置将自动的传个　下一个.所以只要编辑必要的修改就可以了

取消将关闭当前没有保存的修改过的对话框.你将返回主菜单'选项'　'工程源控制'.

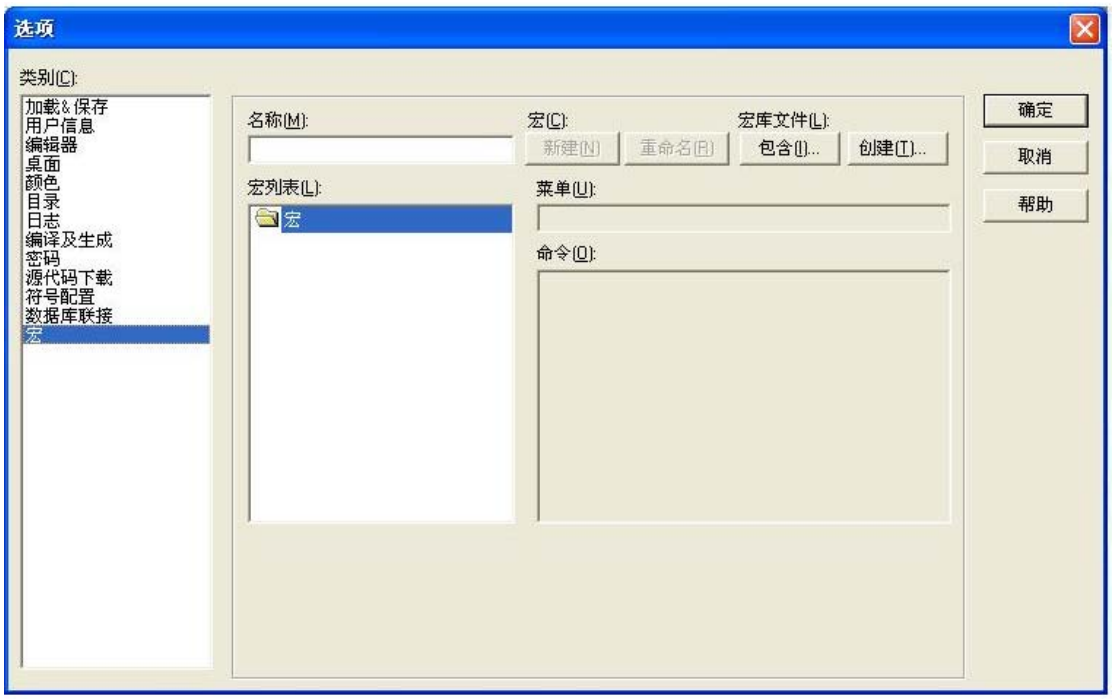
如果一个已有设置被更改,新的更改点击 OK 保存.然后菜单关闭返回 '选项'　'工程源控制'.

注意: 设置将被储存在工程中

宏选项

如果在 选项 对话框中选择了这个类别，将会出现如下的对话框：

宏选项对话框



在这个对话里，可以使用CoDeSys批处理的命令来定义宏，然后能在'编辑'　'宏'菜单中调用它。

通过下面的步骤来定义一个新宏：

•在 Name 输入区域，为要创建的宏输入一个名字。在按了新建按钮后，这个名字转移到了宏列表区域并且在这里标记为被选中。宏列表以树状结构表示，本地定义的宏按顺序排列，如果宏库是完整的,那么库名字将列出来并且通过鼠标在那些条目附近的加号/减号符号点击可以打开和关闭一个库文件的列表。

•目录区域是用来定义菜单条目，宏出现在'编辑'　'宏'菜单中。为了能使用一个单独的字母作为快捷键，字母必须在它之前加 '&'。例如：名字"Ma&cro 1"产生菜单项"Macro 1"。

•在编辑区域命令，可以定义和/或编辑命令来组成最近创建的或已选择的宏。可以使用所有 CoDeSys 的批处理命令和所有的关键字，通过按 Help 按钮可以得到一个列表，按<Ctrl><Enter>开始一个新命令行，通过按鼠标右键菜单可以得到带公共文本编辑器功能的内容菜单。属于集合的命令组件可以通过使用引用标记来分组。

•如果你想创建更多的宏，重复执行 1-3 步，按 OK 按钮关闭对话。

如果你想删除一个宏，在宏列表中选择它并按<Del>按键。

如果你想重命名一个宏，在宏列表中选择它，在编辑区域 'Name' 中插入一个新名字然后在按 Rename 按钮。

为编辑一个已有的宏，在宏列表中选择它并编辑 '目录' 和/或 '命令'区域，当按 OK 按钮时，更改保存。

只有通过按 OK 按钮关闭现有的对话所有宏的描述将保存到工程中。

在“编辑” “宏”菜单中的宏菜单条目现在按它们定义的顺序显示，只有选中菜单才能选中宏。

宏库:

宏可以保存在外部宏库文件中，这些库文件可以被包含在其它工程中。

创建一个包含当前打开的工程的宏库：按创建按钮，得到合并工程对话，这里列出了所有的宏变量，选择期望的条目并按 OK 确认，选择对话将关闭同时保存宏库打开，在这里为新库输入一个名字和路径，按保存按钮保存，库被命名为<library name>.mac 并且对话将关闭。

???在当前打开的工程中包含一个宏库<library name>.mac ：按包括按钮，打开对话打开宏库，它显示带扩展名*.mac 的文件，选择期望的库并按 Open 按钮，对话将关闭并且库被添加到了宏列表的树中。

提示：工程中的宏也能导出('工程' '导出')

4.3 管理工程

这里讲述菜单项目“文件”和“工程”下包含的命令。

'文件' '新建'

符号： 

通过这个命令可以创建一个名为“Untitled”的空白工程，在保存的时候必须修改这个名字。

'文件' '从模板中新建'

使用这个命令打开一个CoDesys工程作为“模板”工程，打开工程文件对话框弹出，选中的工程以“Unknown”名字打开。

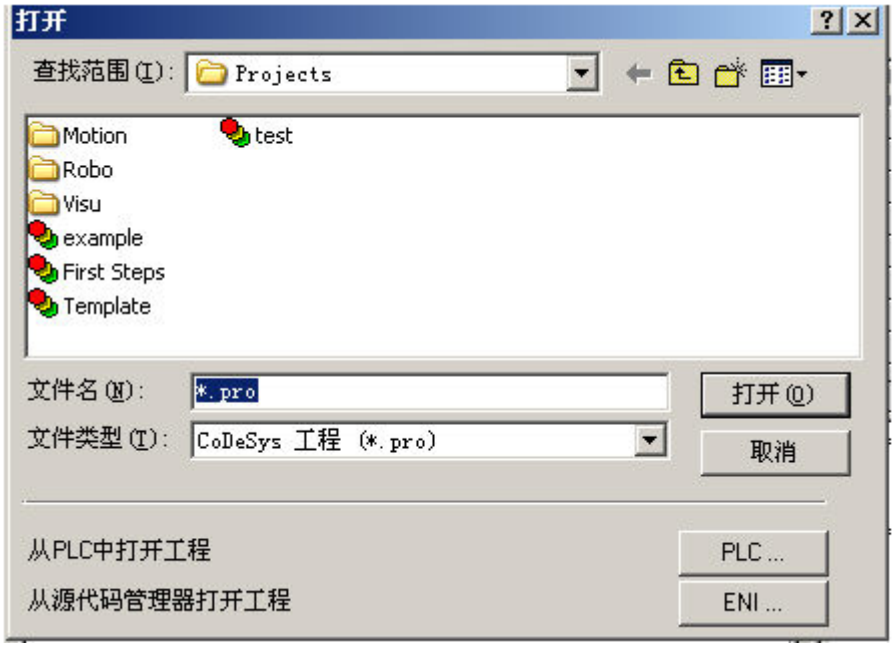
'文件' '打开'

符号： 

使用这个命令可以打开一个现存的工程，如果一个工程已经打开和修改了，那么 CoDeSys 会询问是否要保存这个工程。

打开文件的对话框出现，必须选择一个带扩展名“*.pro”工程文件或一个带扩展名“*.lib”库文件 ，这个文件必须已经存在。通过命令“打开”是不能创建一个工程。

文件打开对话框



从 PLC 中打开一个工程

在从 PLC 中打开工程文件时，按“PLC”键，就可以从 PLC 中上传一个工程文件。下一步，出现一个通讯变量对话框（请参看“联机”“信息参数”菜单），用以没有进行 PLC 连接时的传输变量设置。一旦联机成功，系统可以检测到具有相同命名的工程文件存在于你的计算机硬盘目录中。当从控制器加载工程的对话框出现时，你可以决定是否用控制器使用的工程替代本地文件（这个顺序与“联机”“加载源代码”的顺序是相反的。工程源文件保存在控制器中。不要与“创建引导工程”相混淆）。

注意：请注意，在任何情况下必须给工程一个新名字，当你把它从本地目录加载到 PLC 时，否则它是没有名称的。如果目标系统支持，输入到工程信息中的'Title'将会预定义为新工程文件名。在这个情况下从 PLC 中调用工程时，保存文件的对话框将会打开。新文件名是自动添加的并且可以确认或修改。

如果还没有加载工程到 PLC，会得到一个错误信息。

（参照“工程” “选项”“类别” ’源下载’）

从源代码管理器中打开一个工程（ENI 数据库）

为打开一个存储在 ENI 工程数据库的工程，激活选项在源代码管理器上打开工程，前提条件是你必须能访问运行数据库的 ENI 服务器，按 ENI...按钮，得到一个对话，在这里可以连接到数据库类别“工程对象”的服务器上。

输入正确的 访问数据 （TCP/IP地址，端口，用户名，密码，只读等）和数据库文件夹（工程名），就可以从数据库文件夹中得到对象，按NEXT按钮，对话会关闭，另一个对话打开，这里你必须为数据库类别'共享工程'输入入访问数据，按完成按钮，对话框关闭并且在CoDeSys对象管理器中自动显示重新得到定义文件夹的对象，然后打开 工程选项对话来设置期望的变量。

最近打开的文件

最近打开的文件在文件菜单中在命令“文件”“退出”的下边列出。如果你选择它们中的一个，相应的工程将打开。

如果为工程定义了 用户组或 密码，会出现一个要求输入密码的对话框。

’文件’ ’关闭’

用这个命令可以关闭当前打开的工程，如果工程被修改了，CoDeSys 会询问是否保存这些变化。

如果要保存的工程名字为“Untitled”,必须赋予它一个名字（参照 ’文件’ ’另存为’）

’文件’ ’保存’

符号：  快捷方式 <Ctrl>+<S>

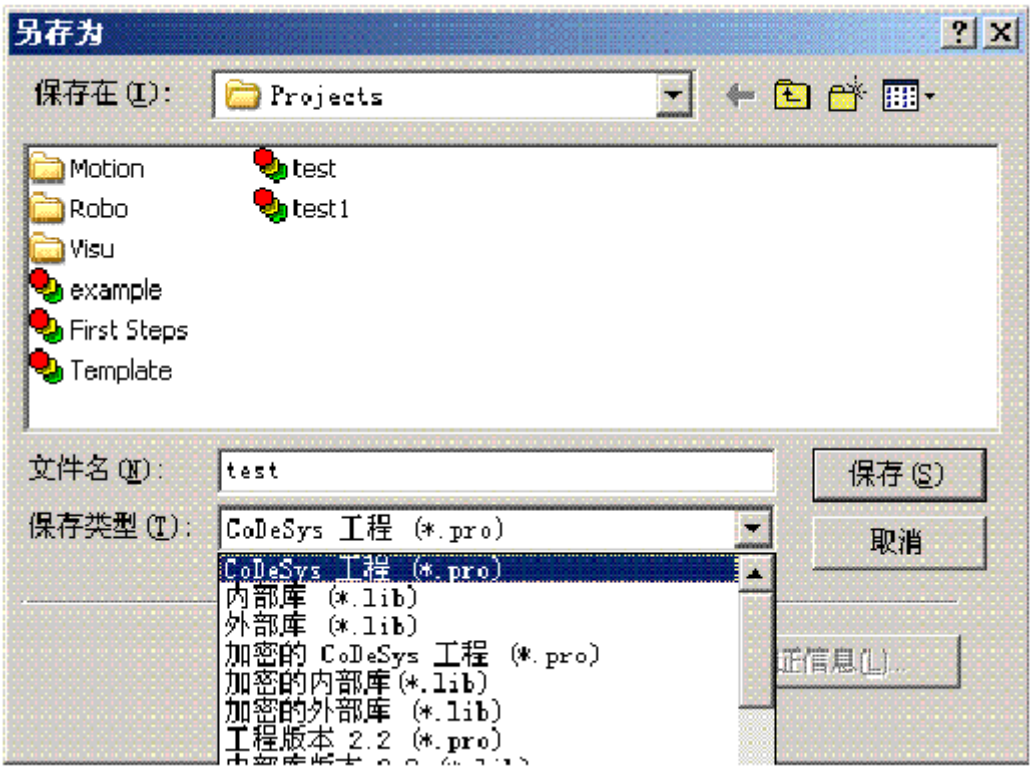
用这个命令可以保存工程中的任何变化,如果要保存的工程名字为“Untitled”， 你必须给它一个名字。

’文件’’另存为’

用这个命令当前工程可以保存为另外的文件或存为一个库文件，这不会改变原工程文件。

在命令选择之后，出现保存对话框，选择一个存在的文件名或输入一个新文件名，并选择合适的文件类型。

“另存为”对话’



如果工程文件以一个新文件名保存，还应选择文件类型 CoDeSys Project (*.pro)。

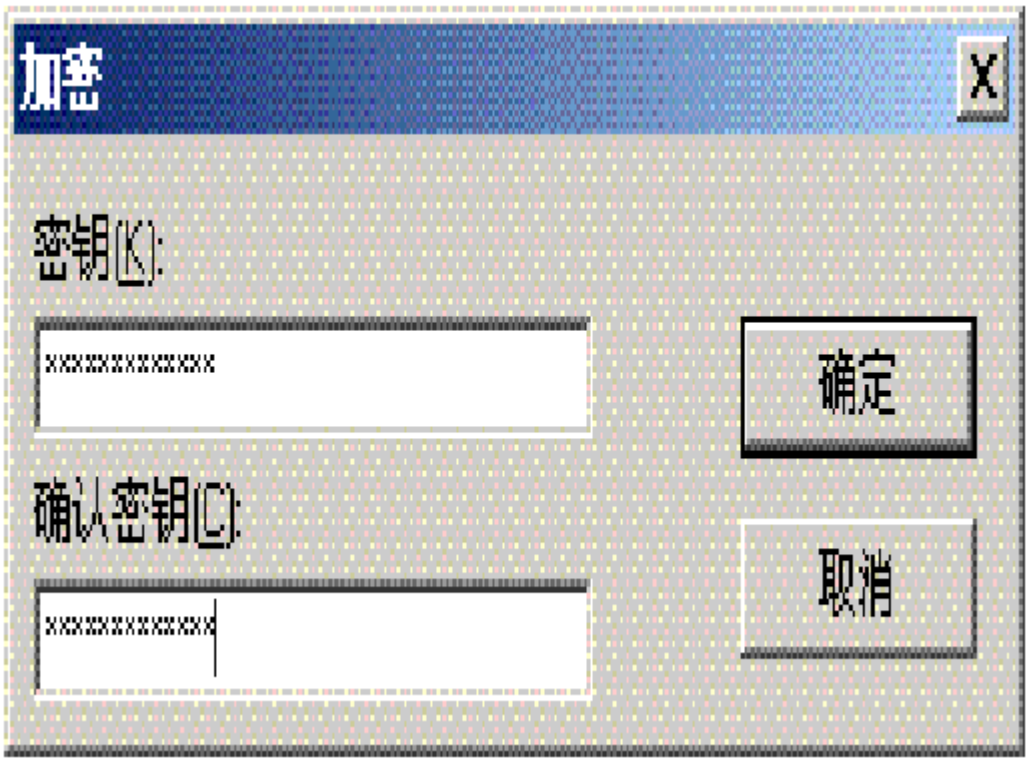
如果你选择了文件类型 Project Version 1.5 (*.pro), 2.0 (*.pro), 2.1 (*.pro) or 2.2 (*.pro)，当前的工程保存为先前版本 1.5, 2.0, 2.1 or 2.2 创建的，版本 2.3 的特殊的数据可能因此丢失，然而，工程可以在版本 1.5, 2.0, 2.1 or2.2 上运行。

为了在其它工程中使用它，也可以把当前工程保存为一个库文件，如果你已经在 CoDeSys 中编写了 POUs，选择文件类型自带库 (*.lib)。

如果你想运行程序，或以其他语言（例如 C 语言）集成 POUs，就可以把文件类型选择为扩展库 (*.lib)。这意味着其它文件也可以使用库文件名保存，但是扩展名必须是“*.h”。这个库文件的构造是一个 C 语言头文件。它保留对所有程序组织单元（POU）、数据类型和全局变量的声明。如果使用外部库，在仿真模式时，可以对 CoDeSys 中程序组织单元（POU）执行写操作。在 C 语言中编写的执行程序将会与硬件一起工作。

如果将工程保存成加密的工程或库文件，请选择加密 CoDeSys 工程(*.pro)、加密内部库(*.lib)或加密外部库(*.lib)，在这种情况下，将弹出‘加密’对话框，你可以定义和确认访问密码。设置密码后如果没有密码，您将无法打开它。

工程加密对话框

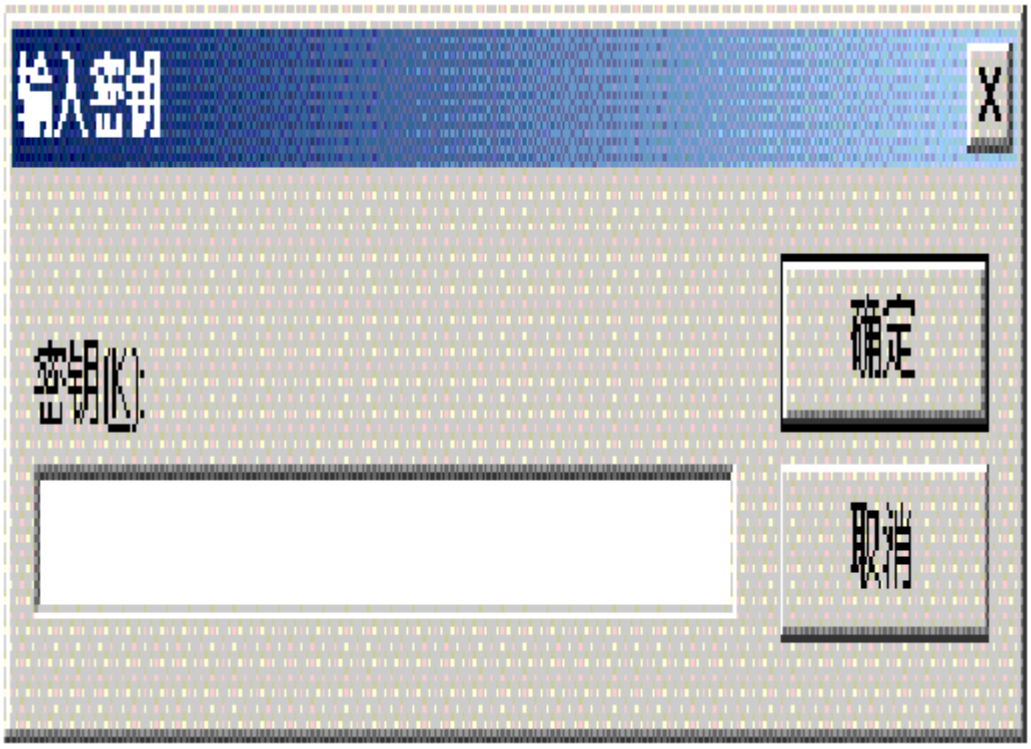


加密功能增强了工程的保护性，到目前为止，只能借助 写保护密码访问工程，这种方法将一直沿用，注意如果不输入库密码，那么不能将一个库插入到工程中。

定义的密码将一直保存在工程中，如果要修改它，请再次使用“另存为”对话框进行设置。

如果打开加密工程或插入加密库到工程中，将显示输入密码对话框。

输入密码对话框



带许可证的库文件：

如果你想保存工程为一个特许库文件，可以在对话“编辑许可证信息”中添加适当的特许信息，通过按

按钮编辑许可证信息打开对话。详细讲述参照“CoDeSys 中的许可证管理”部分。

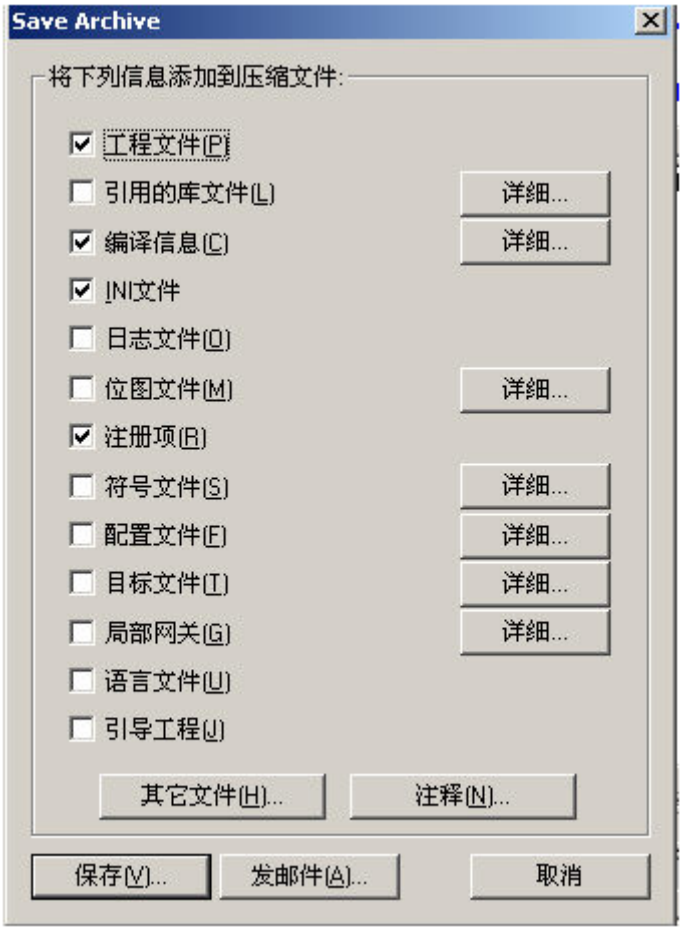
在做完所有的设置后，按是，当前工程将会保存到指定的文件中，如果新文件名已经存在，那么会询问你是否想覆盖这个文件。

当工程文件存为一个库文件时，要对整个工程文件进行编译。编译中如果出现错误，将会被告知创建库文件的工程必须是正确的，工程不能存为库文件。

’文件’’保存/发送压缩文件’

这个命令用来建立和创建一个工程存档文件，CoDeSys 工程使用的和引用的所有的文件能被压到一个压缩包中，压缩文件能在 email 中存储或直接发送。如果你想传送一套工程的相关文件，这是非常有用的。

当命令执行时，打开保存存档文件对话框：



这里可以定义把那些文件类别加入到压缩档案文件中：通过激活/取消相应的检查框选择或取消选定一个类别，在检查框上单击鼠标或在类别名上双击鼠标。如果一个类别标记了☑，这个类别的所有文件将会添加到压缩文件中，如果它标记为☐，将没有文件加入。按动相应的按钮，可以进行同一个类别中的单个文件的选择。

打开文件选择对话框.

在这个对话中选择/取消选择需要的文件：用按钮选择所有将选中文件列表中的所有文件，用不选择将取消所有的选中 。单个的文件可以通过在选择框中点击鼠标来选择/取消选择，也可以在列表条目上双击鼠标或当列表条目被标记时通过按空格键来选择。

用 Save 保存新设置来关闭对话框。

在主对话中类别的选择框，不是所有的文件都被选择，将显示一个灰色的背景色 。

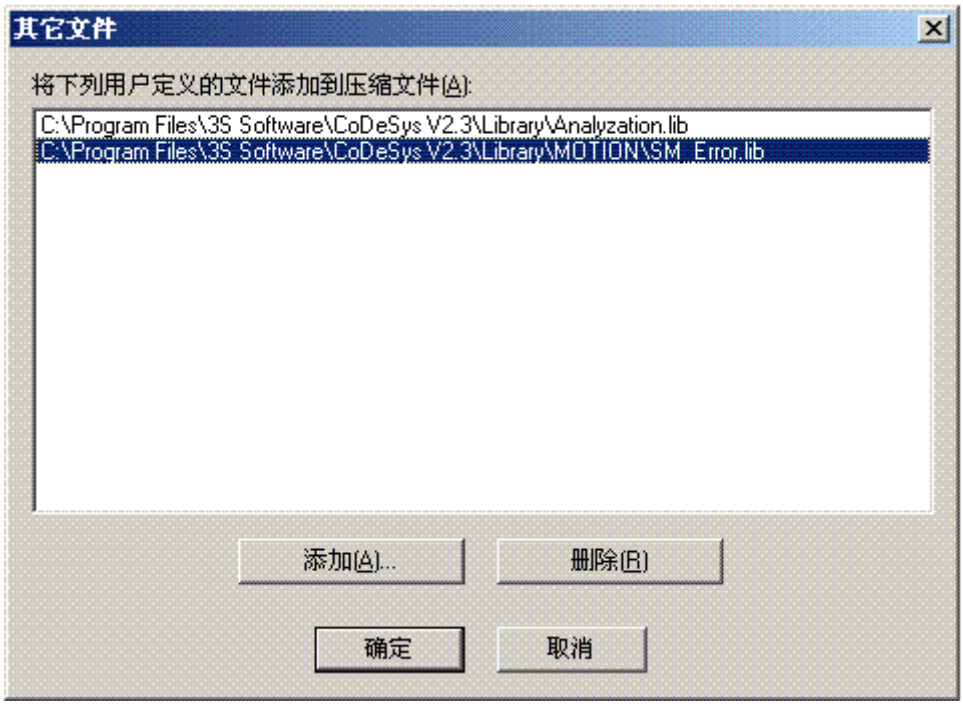
下面的文件类别是常用到的，表中右边的列显示了能添加到压缩文件中的文件：

Category	Files
工程文件	工程文件名.pro(CoDeSys 工程文件)
引用库	*.lib,*.obj,*.hex(库和可能的相应的对西那嘎和 16 进制文件)
编译信息	*.ci(编译信息) *.ri(下载/引用信息) <temp>.*(临时编译和下载文件) 也用于仿真
INI 文件	Codesys.ini
日志	*.log(工程日志文件)
位图文件	*.bmp(用于工程 POU 和可视化的位图)
注册输入项	Registry.reg (输入项用于自动化联盟,网关和 SPS;下面的路径被打包: HKEY_LOCAL_MACHINE\SOFTWARE\3S-Smart Software Solutions HKEY_LOCAL_MACHINE\SOFTWARE\AutomationAlliance″

符号文件	*.sdb, *.sym (符号信息)
配置文件	用于配置 PLC（配置文件，设备文件，图标等。）：例如 *.cfg,*.con,*.eds,*.dib,*.ico...
目标文件	*.trg(所有已安装目标的二进制格式的目标文件) *.txt(可用的已安装目标的文本格式的目标文件)
网关文件	Gateway.exe,GatewayDDE.exe,GClient.dll,GDrvBase.dll,GDrvStd.dll,GHandle.dll,GSymbol.dll, 其他在网关目录下可用的 DLL 文件
语言文件	language files used for visualizations (*.vis, *.xml)
引导工程	引导文件分别为<工程名>.prg, <工程名>.chk,目标系统特定的引导文件.

如要把其它文件添加到压缩文件中，可以按动按钮其他文件。此时将打开对话框“其他文件”。在这里你可以建立一个需要压缩的文件列表。

为档案压缩文件中添加 其它文件对话框。



按动“添加”按钮，可以打开一个用于打开文件的标准对话框。在这个对话框中可以浏览文件；选择一个文件并打开；把文件添加到‘Other files’对话框中。重复以上操作就可以添加每个你想添加的文件。按动删除按钮，可以从文件列表中删除文件。当选择的文件列表建好后，按“是”关闭对话框。

按动“注释”按钮，将打开一个文本编辑器。在这里可以输入文本，把一个 Readme 文件添加到档案文件。按“OK”关闭对话框。在创建档案文件过程中会添加一个 readme.txt 文件。输入的附加信息将包含文件创建期和 CoDeSys 的版本信息。

如果所有期望的选择已经成功，在主对话中可以分别按动：

•保存... 为创建和保存档案文件：保存文件的标准对话将会打开并且你可以输入档案文件要保存的路径，每个档案文件默认名字为<projectname>.zip，用 Save 来开始建构它，在创建中显示当前进程状态并且在消息窗口列出并发的步。

•邮件… 为创建一个临时的档案文件和自动产生一个包含了档案文件附件的空的电子邮件，这个特点仅

仅在 MAPI（消息应用程序接口)正确安装在系统时才起作用，否则产生一个错误信息。在设置电子邮件过程中显示进程的状态和在消息窗口列出步的动作，临时的档案文件在动作完成时自动移除。

•取消　取消动作；将不创建档案文件。

注意：在一个不同系统解压档案文件，它可能必须改变文件路径！

'文件' '打印'

快捷方式：<Ctrl>+<P>

用这个命令打印活动窗口的内容。

在选中命令后，接着出现打印对话框，选择期望的选项或配置打印机然后按 OK，就可以打印活动窗口，可以打印出编辑器中的所有颜色。

打印设置对话框

可以选定文件打印的份数，以及把打印内容输出到一个文件。

还可以通过“属性”按钮，设置打印机。

使用 打印设置，你可以设置打印输出方式。

在打印过程中，对话框显示打印完成的页数，如果关闭此对话框，在下一页打印完成后打印中止。

使用命令 '工程' '文档' 可以把整个工程文档化。

如果想为工程创建一个文档框架，在这里可以保存关于工程中所用变量的注释，那么打开一个全局变量列表和使用命令“附加”“创建文档框架文件”。

'文件' '打印设置'

用这个命令可以决定打印页的版面，出现下面的对话框：

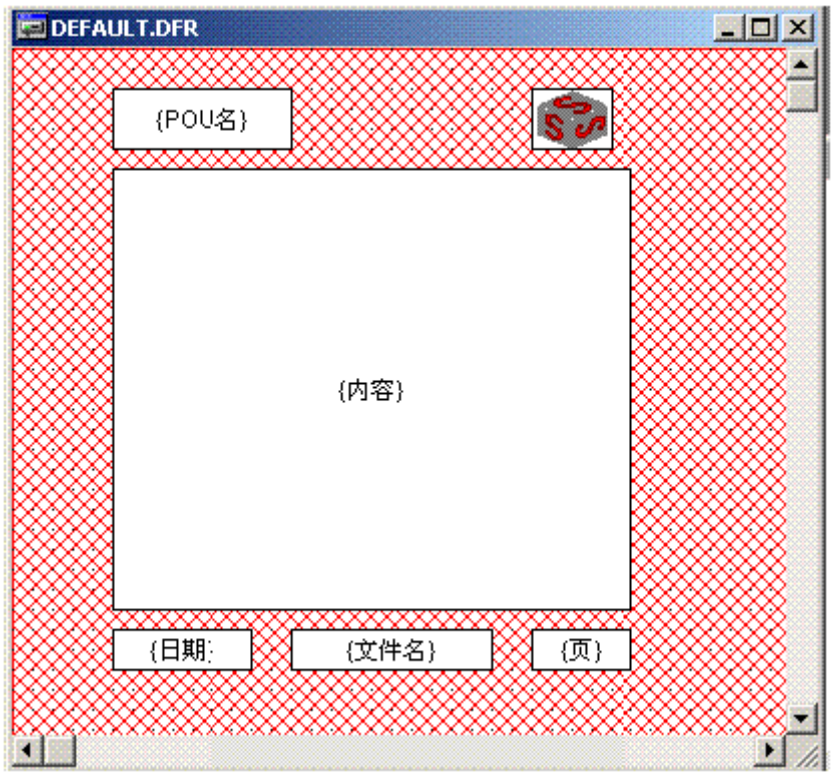
页面设置对话框

在文件区域，输入要保存的版面中带扩展名".dfr"的文件的名字，默认的目的文件设置是文件DEFAULT.DFR。

如果你想改变目前的版面，用按钮浏览在目录树中浏览寻找期望的文件，

也可以选择一个项目或者一个子项目为一个新页面。使用打印机设置按钮来打开打印机设置页面。

页面占位符设置



用菜单项目“插入”“占位符”和并发的选择在五个占位符(Page、 POU name、 File name、 Date and Content)，通过在版面上按鼠标左键拖动一个矩形插入一个所谓的占位符到版面，在打印输出它们被替换为下面：

命令	占位符	作用
页面	{页面}	当前页码出现在打印输出中
POU 名	{ POU 名}	显示当前 POU 名
文件名	{文件名}	显示工程名

日期	{日期}	显示当前日期
内容	{内容}	显示 POU 内容

另外，用“插入” “位图”命令可以在页面中插入一个位图（如公司的标志等），在选择图形之后，在版面上使用鼠标可以在这里画一个矩形，也能插入其它的可视化组件（参照可视化）。

如果模板改变了，在窗口关闭时，CoDeSys 会询问是否保存这些变化。

’文件’ ’退出’

快捷方式：<Alt>+<F4>

用这个命令你可以从 CoDeSys 中退出。

如果一个工程已经打开，那么它会象 ’文件’ ’保存’中讲述的那样关闭。

’工程’ ’编译生成’

快捷方式：<F11>

使用“’工程’ ’编译生成’ ”来编译工程，编译过程基本上是补充增加式的，当保存工程时，最后一次编译所需的信息保存在 ci-file 中。只有改变的 POUs 才重新编译，如果首先执行命令“工程” “清空”，能得到一个非增量编译。

对于联机修改的目标系统，所有装入控制器的 POUs，在编译时，在对象管理器被标上一个蓝色的箭头。

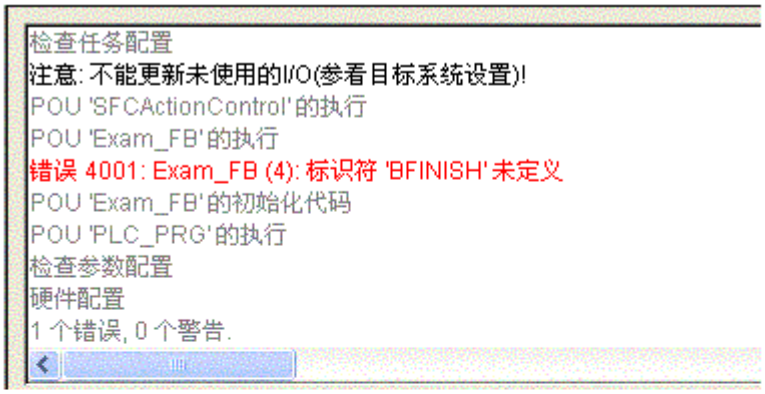
如果控制器通过 ’联机’ ’登录’登录，随着“’工程’ ’编译生成’”编译过程自动执行。

参看此处 图表，它表明了工程－编译生成、工程－下载、在线修改和登录到目标系统之间的关系。

在编译过程中，消息对话框显示了编译的进程、在编译过程中可能发生任何错误和警告和在使用中的 POU 的索引和内存空间（数字和百分比）的信息。错误和警告用数字标记，使用 F1 可以得到当前选择的错误的更多信息。

参照所有变量 出错信息和警告的列表。

在信息窗口中的错误报警和编译信息:



如果在加载 & 保存 类别中的选项对话中选择了选项编辑前保存，工程在编译之前先保存。

注意：交叉引用创建在编译过程中并和编译信息一起保存。为了在菜单“工程”“检查”中能使用命令’显示调入树’、’显示交叉参考’ 和命令’未使用变量’、’并行访问’和’多重输出’，工程必须在改变后重新建构。

’工程’’全部重新编译生成’

用“’工程’’全部重新编译生成’”，不象增量编译(’工程’’编译生成’)那样，工程将彻底重新编译，和命令’全部清除’作用相同。

参看此处 图表，它表明了工程－编译生成、工程－下载、在线修改和登录到目标系统之间的关系。

’工程’’全部清除’

用这个命令，删除来自上次下载和来自上次编译的所有信息。

在命令选择后，出现一个对话框，报告没有新下载的登录将不再有效。这里，命令既可以取消也可以确认。

注意:在执行"全部清除"命令后,,对 PLC 工程进行在线修改只有在下述的情况下可以进行,即在执行全部清除以前已经重新命名了工程目录中的*.ri 文件(保存上次下载信息)或已经将此文件保存到工程目录以外.(参看 ’加载下载信息’), 在登录前先加载它.

参看此处 图表，它表明了工程－编译生成、工程－下载、在线修改和登录到目标系统之间的关系。

’工程’’加载下载信息’

使用这个命令能卸载工程的下载信息，如果它和工程不在同一个目录中存储，在选择命令后，会打开标准对话框“文件打开”。

下载信息在下载时自动保存，它依赖于目标系统，潜在地在每次离线时在文件中创建一个导入工程，文件命名为<工程名><目标标识符>.ri 并放置到工程目录中。每次工程重新打开时这个文件重新加载，在登录时它用来检查那个 POU 的代码已经改变，在联机交换过程中只有这些 POUs 随后加载到 PLC 中。

注意:在执行"全部清除"命令后,，附属于当前工程的*.ri 文件将从工程目录中自动删除.只有在执行全部清除前重新命名此文件或将文件保存到其它地方,然后在登录前通过此命令加载才能实现在线修改.

参看此处 图表，它表明了工程－编译生成、工程－下载、在线修改和登录到目标系统之间的关系。

’工程’’翻译成其他语言’

这个菜单项目是用来把当前的工程文件转化为其它语言，通过读从工程中产生的转化文件中来执行和在文本编辑器的帮助之下，工程可以显示或真实转化为其它的语言版本。

菜单的子菜单：

创建翻译文件

翻译工程

显示翻译的工程

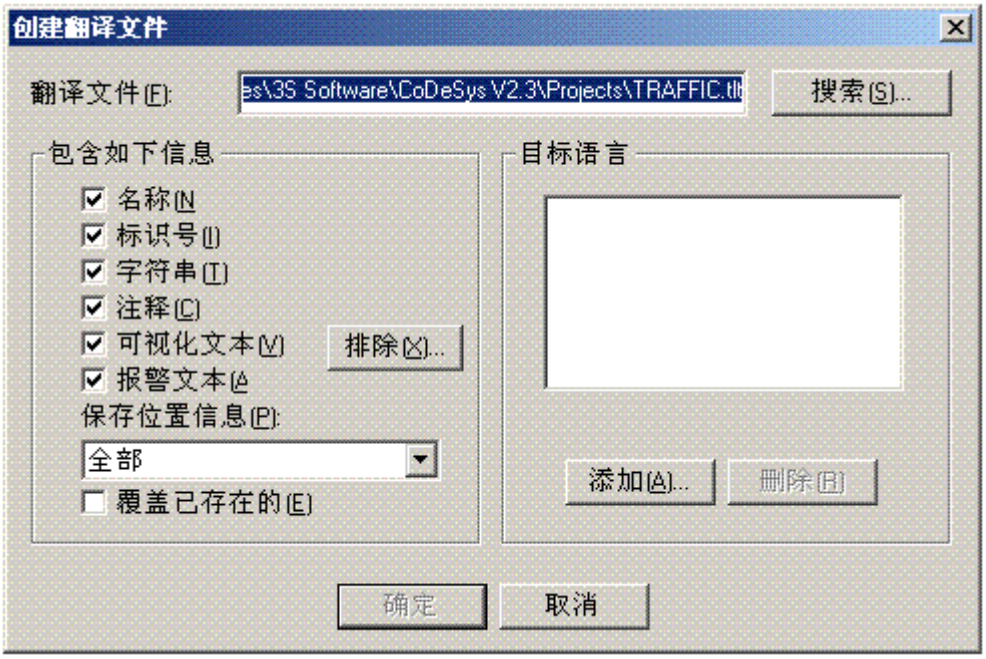
触发翻译

参照: ’翻译文件的编辑’

创建翻译文件

在“工程”“翻译成其他语言”菜单中的这个命令引导“创建翻译文件”对话。

创建翻译文件对话框



在转化文件区域，输入一个文件要存放的位置的路径，默认的扩展名是*.tlt；这是一个文本文件。如果你想在 EXCEL 或 WORD 上操作，你也能使用扩展名*.txt，也是推荐的，因为在这种情况下是用表格形式组织数据。

如果想处理的翻译文件已经存在，给出文件的路径或使用 Search 按钮取得标准 windows 文件选择对话。

下面的来自工程中的信息可以有选择的传递到正在修改或创建的翻译文件中，因此在翻译中用到它们：名字，比如，在对象管理器中的标题'POUs')，标识符、串、注释、可视化文本、报警文本 。另外，也能翻译这些工程组件的位置信息 。

如果相应的选项选中，当前工程的信息会导出语言符号到一个新创建的翻译文件中或添加到一个已经存在的翻译文件中。如果各自的选项没有选中，相关类别的信息，不论是它来自那个工程，将会从翻译文件中删除。

在可视化元素里的“Text” 和“Tooltip-Text”在这里也被认为是可视化文本。

注意：对于可视觉化文本，为了能转移到翻译文件中，必须在可视化元素配置对话中必须指明它们放置在两个"#"符号之间，这些文本在使用命令“工程” '翻译成其他语言'时也不翻译！在联机模式下并且相应的语言输入到“附加”“设置”对话时可视化的语言才改变。

位置信息：详细的文件路径、POU 和排列语言符号的位置对翻译很有用，可以选择以下三个选项：

无:	步产生位置信息
首次出现	元素首次出现的位置被添加到翻译文件中
所有	指定所有出现的元素的位置

如果要编辑的翻译文件是以前创建的，它比当前选择的文件中包含了更多的位置信息，不论它是从哪个工程中产生的，那么它将会相应地被删节或删除。

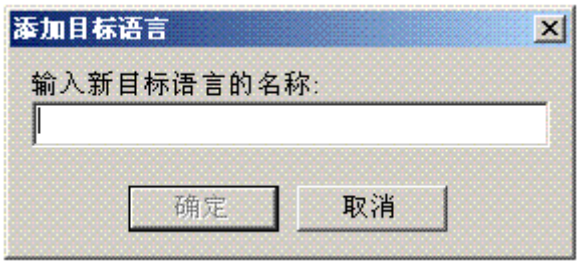
注意：每个元素（语言符号）将会产生最大的 64 个位置详述，即使用户已经在“创建翻译文件”对话中的“位置信息”下选择“所有”

目标语言：这个列表包含了翻译文件中包含的所有语言的标识符和那些要添加到“创建翻译文件”对话的语言。

扩展按钮打开“扩展库”对话，在这里能选择在工程中包含的库，它的标识符信息不转移到翻译文件中。为完成这个，在左边在表扩展库用鼠标选中相应的条目使用 Add 按钮放置到右边的扩展库表中，同样地，放在这里的条目也可以使用删除按钮删除。按'确定'按钮关闭对话框。

从翻译文件中排除库文件对话框

添加按钮打开 “添加目标语言”对话框：



在编辑区域必须输入语言标识符；在开始或结束不能包含空格或变音字母(ä, ö, ü)，按 OK 关闭“添加目标语言”对话，新目标语言出现在语言列表中。

按删除按钮，从列表中移除一个选择的条目。

为了产生一个翻译文件，可以通过 OK 来确认“创建翻译文件”对话。

如果存在一个相同名字的翻译文件，系统提示回答 Yes 或 No 的确认信息：

“指定的翻译文件已经存在，现在将改变和创建一个现有文件的备份，你想继续吗？”按 No，对“创建转化文件”对话不返回任何动作，如果按 Yes，会在同一目录中创建现有翻译文件的复制文件，文件名为“Backup_of_<translation file>.xlt”，相应的翻译文件将会依据输入的选项被修改。

当一个翻译文件产生时下面的也产生：

•对每个新目标语言，每个要显示语言符号产生一个占位符("##TODO")。(参照’翻译文件的编辑’怎样在翻译文件上工作)。

•如果一个现有的翻译文件在处理，出现在翻译文件中的语言的文件条目，但不在目标语言列表中，会被删除，不论是从哪个工程中产生的。

编辑翻译文件

翻译文件必须打开并且保存为文本文件.记号是## 键. 文件中的##TODO 占位符可被变量翻译替换. 对于一种语言符号,一个以类型标识符为始末的段落将被产生例如##NAME_ITEM and ##END_NAME_ITEM 包括对象建立时的名称部分(COMMENT_ITEM 表明注释部分,IDENTIFIER_ITEM 为标识符,STRING_ITEM 为行 VISUALTEXT_ITEM 为可见文本).

参照一个用于 POUs 工程的翻译文件段落. ST_Visu. 目标语言将为英语(美国)和法语.这个例子中工程元素的位子信息将被翻译为一下部分:

翻译前:

##NAME_ITEM
[D:\CoDeSys\projects\Bspdt_22.pro::ST_Visualisierung::0]
ST_Visualisierung
##English :: ##TODO
##French :: ##TODO
##END_NAME_ITEM

翻译后:

The ##TODOs have been replaced by the English resp. French word for 'Visualisierung':

##NAME_ITEM
[D:\CoDeSys\projects\Bspdt_22.pro::ST_Visualisierung::0]
ST_Visualisierung
##English :: ST_Visualization
##French :: ST_Visu
##END_NAME_ITEM

请检查翻译标识符,关于标准和注释的保存变量名称的正确性.例如:对于一个注释(##COMMENT_ITEM) 在翻译文件里被描绘为"(* Kommentar 1)" ,"##TODO" 后的"##English" 必须被"(* comment 1 *)"取代. 对于(##STRING_ITEM) 被描绘为"zeichenfolge1" , "##TODO" 必须被"string1"取代.

注意: 下列翻译文件如果不是高级用户不要被修改:语言块,标记块,位置信息,最初文本

翻译工程（到其他语言）

在菜单“工程”“翻译成其他语言”中的这个命令打开 “翻译成其他语言”对话框。

如果使用了适当的 翻译文件 可以把当前的工程翻译成另外的语言。

注意：如果你想在最初创建的语言中保存工程的版本，在一个不同的名字下保存工程副本优先于翻译，翻译过程不能够完成。在另外语言中只仅仅显示的工程（在这个显示版本中不能编辑）

在翻译文件区域，提供要翻译的文件路径，通过按搜索按钮可以访问标准 windows 文件选择对话。

目标语言区域包含一个输入到翻译文件的语言标识符列表，从这个列表中可以选择期望的目标语言。

按 OK 开始把当前工程在指定的翻译文件的帮助下翻译成选择的目标语言。在翻译过程中，显示一个进度对话和错误信息，在翻译之后，对话框和工程中所有打开编辑窗口关闭

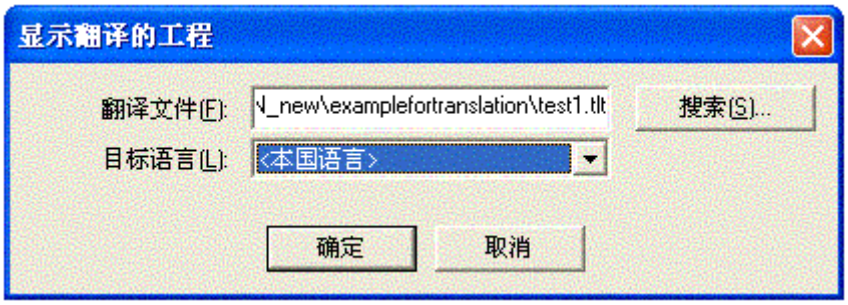
按‘取消’按钮不修改当前工程而关闭对话框，如果翻译文件包含错误的条目，在按 OK 按钮之后显示一个错误消息，错误消息给出文件路径和错误行，例如：“[C:\Programs\CoDeSys\projects\test.tlt (78)]; 翻译期望的文本”。

'显示翻译工程'

如果这里有工程用到的翻译文件，你能在这里显示定义的语言版本中的一个，不需要覆盖工程的原始语言版本。(关于这个可能性可以与一个工程的“real”翻译相比较，你可以用命令“翻译工程”来翻译工程，这意味着要创建一个新的版本的工程!)。

在菜单'工程' '翻译成其他语言'中的命令“翻译此工程”打开对话“翻译成其他语言”。

显示翻译工程对话框



在区域翻译文件插入你想要使用的翻译文件的路径，通过按浏览按钮打开文件时能得到标准对话的帮助。

在区域目标语言有一个选择列表，它除了“<原始语言>”条目外还提供当前设定的翻译文件定义的语言标识符，原始语言是通常和工程一起保存的语言（它只能被“工程”“翻译”修改。）。选择可用语言中的一个并按 OK 确认，这里工程将以选择的语言显示，但是不能编辑它！

如果你想退回来在原始语言中查看工程，使用命令“查看翻译工程”。

’触发语言翻译’

如果你已经通过命令“显示翻译的工程”在其它语言中显示了工程，通过使用’工程’’翻译成其他语言’菜单的命令“触发翻译”能在这个语言版本和（可编辑的）原始版本之间切换。

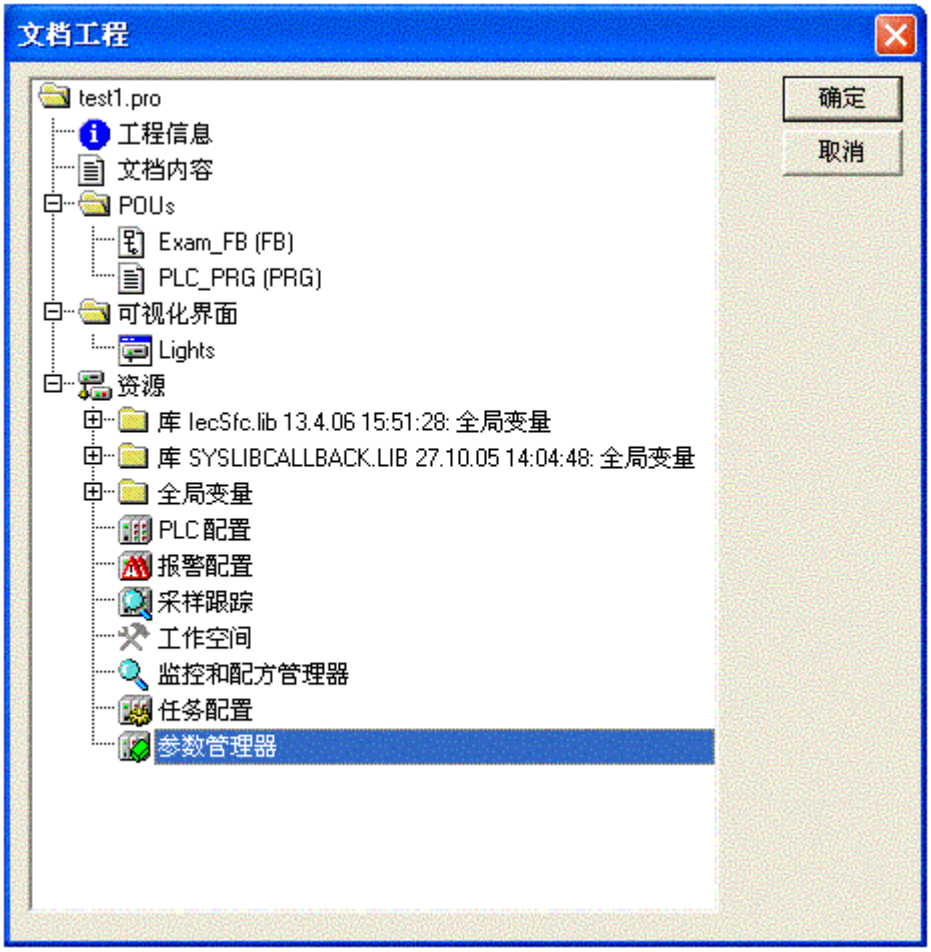
’工程’’文档’

这个命令能打印整个工程的文档，一个完整的文档包括以下组件：

- 程序组织单元（POU）
- 文档的内容
- 数据类型
- 可视化
- 资源、全局变量、变量配置、采样追踪、PLC配置、任务配置、查看和配方管理器。
- 程序组织单元和数据类型的 调用树
- 交叉参考列表

对最后两个条目，工程编译时必须没有错误。

工程文档对话框



只有在对话框中以高亮度蓝色表示的那些区域才能打印输出。

在第一行选择的工程的名字就能选择整个工程。

如果，另一方面，你只想选择一个对象，那么在相应的对象上点击或用方向键移动带点的矩形框到期望的对象上。对象在它们的符号附近有加号的是组合对象包含了其它的对象，在加号上单击对象展开，在减号上点击它有关闭。当你选择一个组合对象，那么所有相关的对象也被选择。通过按<Shift>键你能选择一组的对象，通过按<Ctrl>键你能选择几个不连续的对象。

一旦做好了选择，按OK。打印对话框出现。你能用 ’文件’ ’打印机设置’决定要打印的页的版面。

’工程’ ’导出’

工程能 导入和导出，这允许在不同的IEC编程系统之间交换程序。

在指令表、结构化文本和顺序功能图表（IEC 1131-3 功能组件格式）中的程序组织单元有一个标准化的交换格式，在梯形图和功能模块图中的程序组织单元和其它对象，CoDeSys 有它自己的归档格式

选择的对象写到了一个 ASCII 文件。

POUs、数据类型、可视化和资源能被导出。另外，库管理器中的条目也就是库的连接信息也能导出（不是库文件）。

重要：如果在图形编辑器注释包含一个引用标记(’)，重新导入一个导出的 FBD 或 LD POU 会导致错误，因为这个被解释为字符串的开始。

在对话框窗口做出了选择（和用 ’工程’ ’文档’有相同的情形），此时能决定导出选择的部分到一个文件还是导入到多个文件中，一个对象一个文件，切换选项One file for each object开或关，然后按OK按钮。出现保存文件的对话框，输入一个带扩展名“.exp”的文件名字，为对象导出文件分别存放于目录中，它将以文件名<对象名.exp>保存在这里。

’工程’’导入’

在结果对话框中打开选择期望的 导出文件对话。

数据导入到当前的工程，如果在同一的工程中存在一个与对象相同的名字，那么出现一个对话框询问“你想替换它吗”：如果你回答 Yes，那么在工程中的对象被导入文件中的对象替换。如果你回答 No，那么接收新对象的名字，且补充一个下划线和一个数字(″_0″, ″_1″, ..)。

如果导入的信息同一个库链接，库将被加载和附加到库管理器中的列表的最后。如果库已经加载到了工程，它不会被重新加载。然而，如果导出文件正在被导入，它显示库中不同的存储时间，库文件名在库管理器中用一个″*″标记（例如，standard.lib*30.3.99 11:30:14），和加载一个工程相似，如果不能找到库，那么出现一个信息对话：“不能找到库{<路径>\}<名称> <日期> <时间>”，与当一个工程被加载时一样。

导入的信息在 信息窗口提示。

’工程’’导入西门子程序及变量’

在子菜单“导入西门子工程”中，能找到导入 POUs 和 Siemens-STEP5 and STEP7 文件中的变量的命令。

可用到下面的命令：

- “导入一个SEQ符号文件”
- “导入一个S5 文件”

参看附录G：了解关于 导入西门子的更详细信息。

’工程’’比较’

对于工程中的所有单元，不同处用颜色改变来标记。对于当前打开的工程，可以接受比较工程的结果（这种接受是单向的，从比较工程到当前工程）。

为了接受某个单元的改变，请使用‘接受单独改变’指令。

这个命令用来比较两个工程或比较一个工程的目前的版本和最后保存的版本。

总结：

定义：	当前工程：	当前使用的工程.
	引用的工程:	与当前工程比较的工程.
	比较模式:	工程会在’工程’’比较’执行后被显示出来.
	单元:	参与比较的最小单元。可以是一条直线（声明编辑器，ST 编辑器，IL 编辑器）， 可以是一个网络（FBD 编辑器，LD 编辑器）或是一个元素/POU（CFC 编辑器，SFC 编辑器）.

在比较模式中当前的工程和引用的工程将会出现在一个双向的窗口，发现差别，POUs 的名字被用颜色标记，在编辑器中 POUs 的内容和 POUs 以面对面的方式显示。在比较模式下结果和呈现的方式依赖于：1.为比较运行激发了什么过滤条件，这影响在比较过程中对空白和注释的考虑 2.在行或网络或组件内修改是否被认为插入一个完全新的 POU。

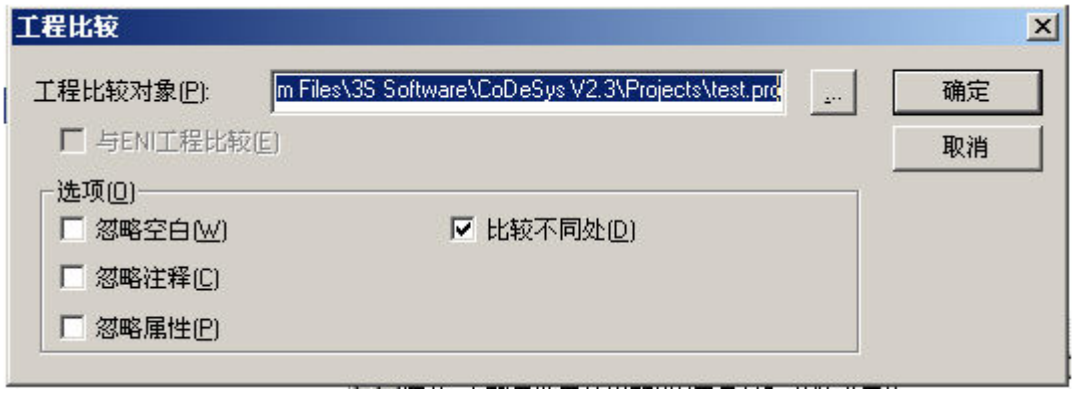
请注意：在比较模式（查看状态栏：COMPARE）工程不能被编辑！

参看:

- 执行比较
- 比较结果的表示方法
- 工作在比较模式下

执行比较

在执行命令“工程”“比较”后，打开对话框工程比较 ：



在工程比较对象处插入 引用工程的路径， 如果你想使用标准对话来打开一个工程，按按钮 ，如果你插入当前工程的名字，工程的当前版本将会和它最后保存的版本比较。

如果工程在 ENI 数据库里在源控制下，那么本地版本可以和在数据库中找到的当前版本比较。要这么做需激活选项 ENI 工程比较 。

可以激活比较中的下列选项：

忽视空白:检测到空白的地方视为没有区别，也就是忽略空白。

忽视注释:对注释视为无区别

忽视属性:对对象的属性视为无区别

比较不同处:如果在一个 POU 内的一个行、一个网络或一个元件已经被修改，在比较模式中它将在双向的窗口显示并直接与其它工程（标记为红色，查看下面）的版本相对。如果选项是未激活的，相应的行将会作为“删除”显示在引用工程中，作为“插入”显示在当前的工程中（蓝/绿，查看下面）。这意味着它不会显示为直接地与其它工程中的相同的行对立。

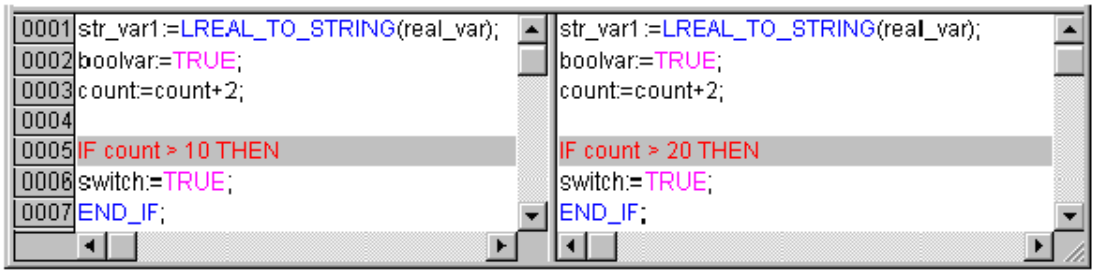
例如：

在当前工程中（左边）0005 行已经被修改。

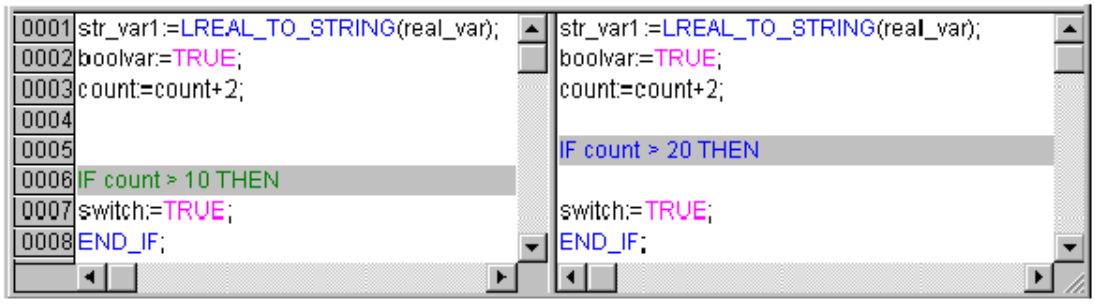
结果图

Example for "Oppose differences"

Option 'Oppose differences' activated:



Option 'Oppose differences' not activated:



当按 OK 关闭对话“工程比较”时，将根据设置执行比较。

参看:

- 比较结果的表示方式
- 在比较模式下工作

比较结果的请求

首先将会打开标题为“工程比较”工程的结构树，来显示比较的结果，这里你能选择特殊的 POUs 来查看详细的区别。

1. 比较模式中工程的概览

在工程比较之后，打开一个双向的窗口，它显示了在比较模式下的工程。在标题栏中显示工程的路径：“工程比较<当前工程路径>-<参考工程路径>”。当前的工程是在窗口的左半部分表示，引用的工程在右边表示。每个结构树在最上面显示了工程的名字，它符合对象管理器的结构。

不同的 POUs 在结构树中通过阴影、特殊的颜色和附加的文本来标识：

红:单元已经被修改；它在两部分窗口中用红颜色字母显示

蓝:单元只在引用的工程中可用；在当前工程的预览结构中相应的位置插入一个间隙。

绿:单元只在当前工程中用到；在当前工程的预览结构中相应的位置插入一个间隙。

黑:在没有发现区别的地方用到。

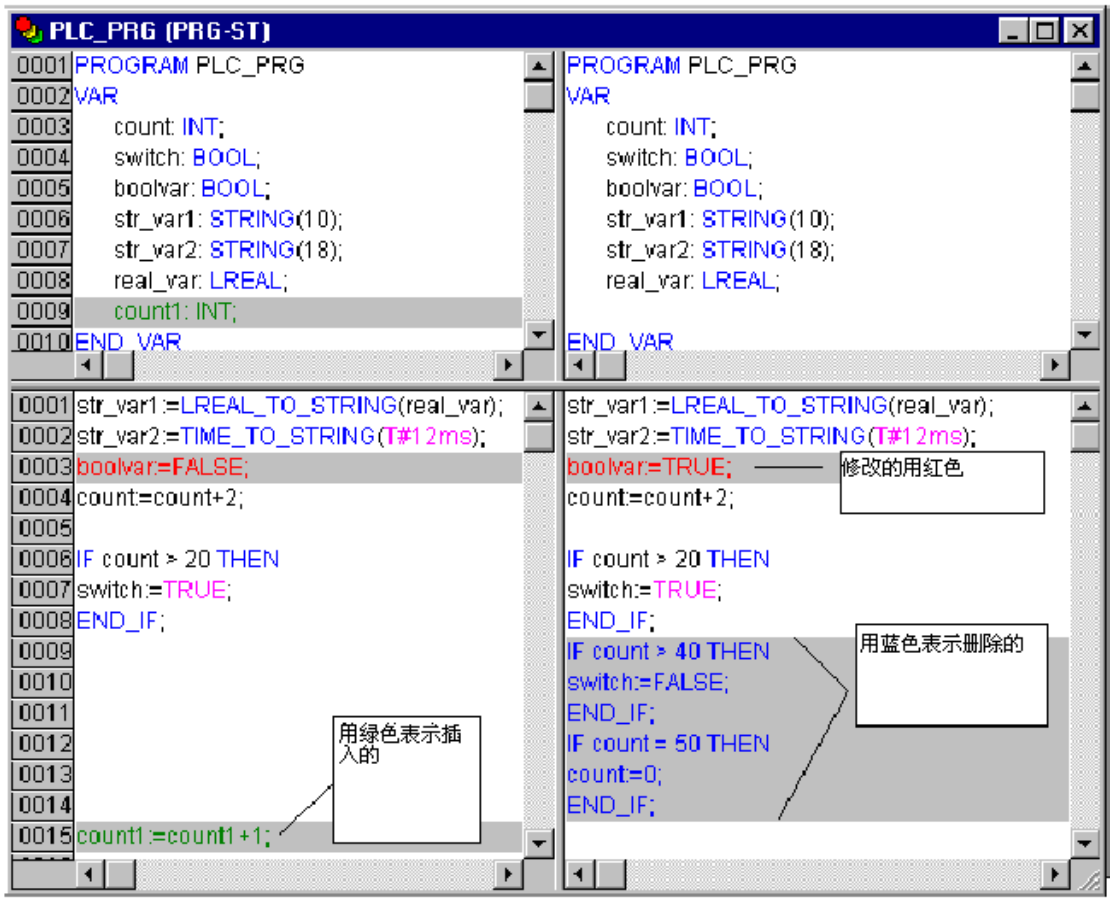
“(属性差别)”：如果检测到了 POU 的属性差别，在工程结构树里这个文本附加到 POU 文件名后。

“(进入正确的变化)”：如果检测到了 POU 的访问权限的差别，在工程结构树里这个文本附加到 POU 文件名后。

2.比较模式下 POU 的内容

•如果它是一个文本或图形编辑器 POU，将会在一个双向窗口中打开。引用工程的内容（右边）与当前工程的内容(左边)是相对应的,在比较过程中最小的单元是一行(declaration editor, ST, IL)、一个网络 (FBD, LD)或一个元素(CFC, SFC)，使用与上面工程概览描述的一样的颜色。

例如，在比较模式下的POU：



••如果它不是编辑器 POU，而是任务配置、目标设置等编辑器，可以在工程结构上各个行上双击鼠标，打开当前工程和引用工程的 POU 版本。此时不显示那些工程的 POU 详细的区别。

参看:

- 在比较模式下工作

在比较模式下工作

把鼠标光标放在双向窗口中的表示不同内容的某一行上，就会出现菜单“附加”。根据工程浏览或者 POU 的不同，菜单可以提供以下指令的选择：

’下一个不同处’

’前一个不同处’

’接受改变’

’接受改变项目’

’接受属性’

’接受访问权限’

’附加’’下一个不同处’

快捷方式：〈F7〉

这个命令在比较模式下是可用的（查看上面’工程’’比较’）。

光标跳到下一个单元，在这里指明了一个差别（在工程概览中的行，在 POU 的行/网络/元件）。

’附加’’前一个不同处’

快捷方式 ：〈Shift〉〈F7〉

这个命令在比较模式下是可用的（查看上面’工程’’比较’）。

光标跳到上一个单元，在这里指明了一个差别（在工程概览中的行，在 POU 的行/网络/元件）。

’附加’’接受改变’

快捷方式 ：〈Space〉

这个命令在比较模式下是可用的（参看上面的’工程’’比较’）。

对于所有的有相同的类别,不同标记的相连接的单元，参考工程的版本被当前的工程接收。相应单元(带颜色)将显示在窗口的左侧。如果它是一个标记为红色的单元， 在当前工程中的接收标记为黄色。

使用’接受改变项目’来接受特定单元的修改.

注意：对于接受工程之间的不同处或访问权限属性,只能是当前工程接受参考工程的.

’附加’’接受改变项目’

快捷方式 ： 〈Ctrl〉〈Spacebar〉

这个命令在比较模式下是可用的（查看上面’工程’’比较’）。

只有光标当前位于单个单元（行，网络，元件）上，才接受当前的版本，相应的单元(带颜色的)将显示在窗口的左边.

对于一个 POU,如果在树型结构中标记为红色,则说明它的内容修改了;如果接受改变,则在当前工程中的POU 标记成黄色.

’附加’’接受改变属性’

这个命令在比较模式下是可用的（查看上面’工程’’比较’）。

执行此命令,光标位置处的 POU 的对象属性将被设置成参考工程的。

’附加’’接受访问权限’

这个命令在比较模式下的工程概览中才有用：（参看上面的 ’工程’’比较’）。

执行此命令,光标位置处的 POU 的访问权限将被设置成参考工程的。

’ 工程’’合并’

使用这个命令，可以进行各种对象（POUs、数据类型、可视化和资源）的合并，还可以把其它工程文件导入到你工程文件内。

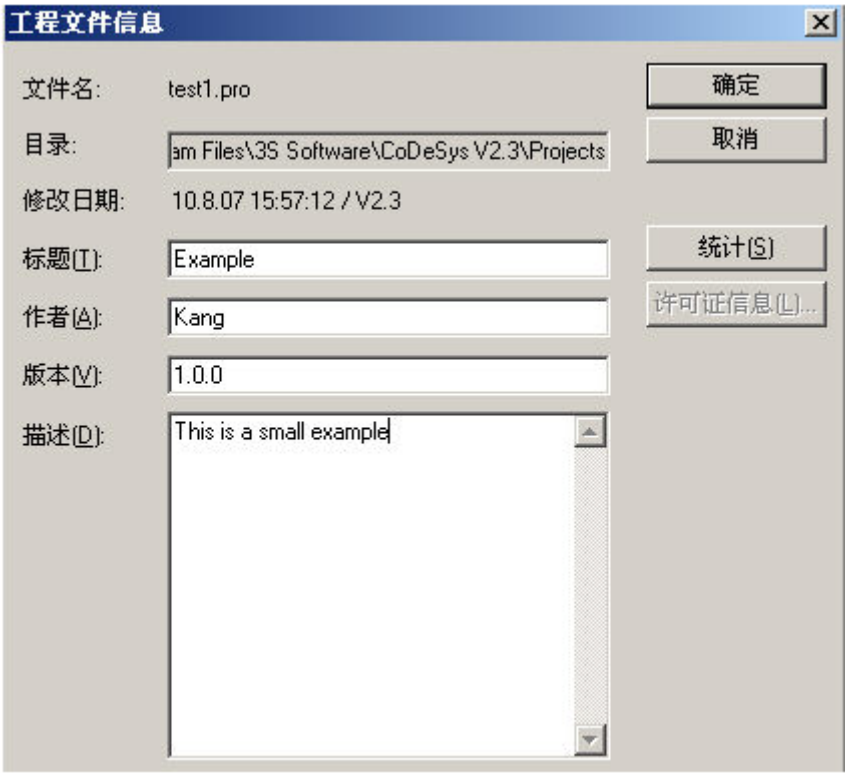
当执行这个命令时，首先出现打开文件的标准对话框。在这个对话框中选择一个文件后，又出现一个对话框。在这个对话框里，你可以象“’工程’’文档’中讲述的那样选择期望的对象。

如果在工程文件中存在与对象相同的名称，则在新对象的名称后附加下划线和数字(如"_1", "_2"等) 。

’工程’’工程信息’

在这个菜单条目下可以存储工程的信息，当给出命令后，打开下面的对话框：

工程信息对话框



显示下列各项信息：

•文件名

•目录路径

•最近改变的时间（改变日期）

•另外，你可以添加下列的信息：

•工程的标题：

请注意：如果目标系统支持，当通过命令“文件”“从 PLC 中打开工程”加载工程时（在这种情况会打开一个保存文件的对话），这个标题自动作为工程文件名。

•作者的名字，

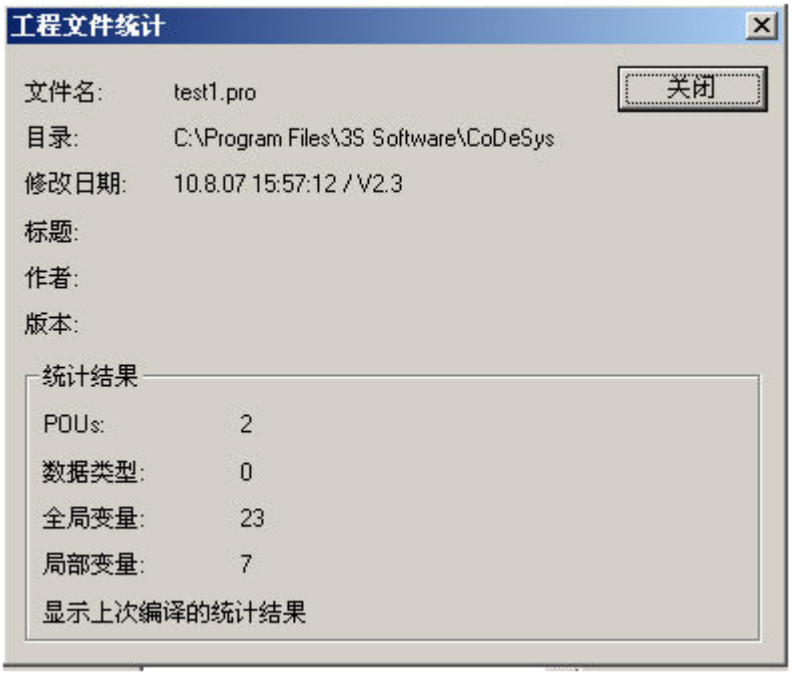
•版本号

•工程的描述。

这个信息是可选的，当你按按钮 Statistics 时你可以接收工程的统计信息。

它包含了信息比如与它们在上次编译被追踪一样的 POUs 数量、数据类型和局部和全局变量。

工程统计结果



当通过命令’文件’’另存为...’处理已经存储的带有许可信息的CoDeSys工程时，按钮许可证信息可用。在这种情况下按钮打开对话“Edit Licensing Information”，在这里你可以修改或删除许可（查看第 9 章“CoDeSys中的许可证管理器”）。

如果你选择了选项对话框中的类别加载 & 保存中的选项询问工程信息，当保存一个新工程，或在一个新名字下保存一个工程时，自动的调用工程信息对话。

’工程’’全局搜索’

使用这个命令，可以搜索 POUs 中的文本、数据类型或在全局变量的对象里的位置。

当输入命令后，会打开一个对话，在这里，可以象在’工程’’文档’描述的那样来选择期望的对象。

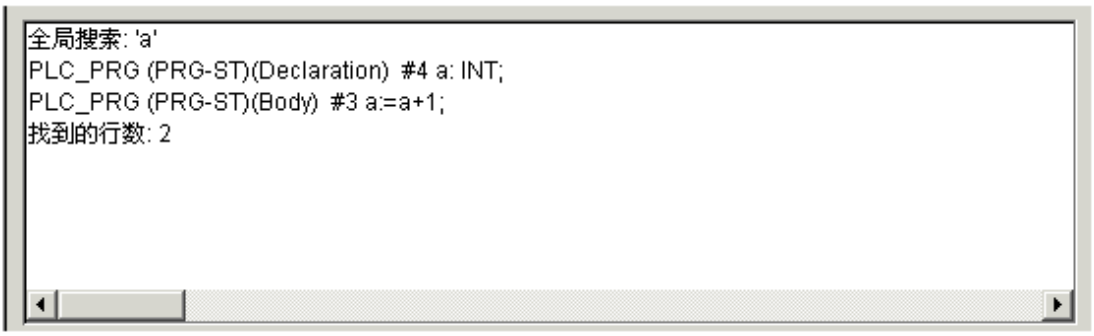
按 OK 键确认了选择后，会带开搜索的标准对话，当通过使用在菜单栏中的符号  来调用“全局搜索”时会迅速出现这个对话；在工程中的所有搜索范围内自动执行搜索。可以通过搜索区域的组合框来选择最新输入的搜索字符串，如果在一个对象中找到一个文本字符串，被加载到相应的编辑器或加载到库管理器和字符串被找到地方的位置将被显示。找到的文本的显示，与命令“编辑”“搜索”的搜索和搜索下一个功能的动作相似。

如果你选择了按钮信息窗口，在被选择的对象中的所有一系列的符号被搜索的地方在消息窗口将会一行一行的列出，然后，将会显示找到的位置的数量。

如果消息窗口没有打开，每个找到的位置显示下列各项：

- 对象名
- POU 的声明或执行部分的位置
- 行和网络编号
- 文本编辑器中的全部行
- 在图形编辑器中完整的文本元素

信息窗口中显示搜索结果.



如果在信息窗口中在一个行上双击鼠标或按<Enter>键，打开加载对象的编辑器，对象中的相关行被标记，可以使用功能键<F4> 或 <Shift>+<F4>在显示行之间进行切换。

'工程' '全局替换'

使用这个命令，可以进行POUs中文本、数据类型或全局变量的对象位置的搜索，并用其它的文本替换这个文本。执行这个命令和使用命令“'工程' '全局搜索'或 '编辑' '替换'效果相同。库文件不能进行选择，也不能在信息窗口中显示。

搜索结果在信息窗口显示。

'工程' '检查'

这个命令用来检查工程的语义上的正确性，这个命令将考虑最近编译的状态，在测试执行以前，工程必须是没有错误的；否则菜单项目是灰色的。

打开一个列出下列命令的子菜单：

•未使用的变量

•内存重叠区域

•并行访问

•在输出端多通道写访问

结果在 信息窗口显示。

请注意：在工程选项类别'编译生成'中，你能定义它们以便在工程编译时自动做这些检查。

未使用变量

在菜单'工程' '检查'中这个功能用来搜索在程序中已经定义的但还没有使用的变量，它们通过POU名和行表示，例如：PLC_PRG (4) - var1。不检查库中的变量。

搜索结果显示在信息窗口中。

内存重叠区域

在菜单 '工程' '检查'中的这个功能可以检查通过"AT"声明的变量在特殊内存区域中的分配是否有重叠发生。例如，当分配一个变量“var1 AT %QB21: INT”和 “var2 AT %QD5: DWORD”时发生重叠，因为它们都使用 21 字节。检查结果如下：

%QB21 被下列变量引用：

PLC_PRG (3): var1 AT %QB21

PLC_PRG (7): var2 AT %QD5

检查结果显示在信息窗口中。

在输出端多通道写访问

菜单'工程' '检查'中这个功能用来搜索内存区域中单个工程在多个地方进行写访问。检查结果如下：

%QB24 被写到下列地址：

PLC_PRG (3): %QB24

PLC_PRG.POU1 (8): %QB24

搜索结果显示在信息窗口中。

并行访问

菜单’工程’’检查’中这个功能用来搜索被多个任务引用的IEC地址的内存区域.写和读访问是没有区别的。例如：

%MB28 在下列任务中引用：

Task1 - PLC_PRG (6): %MB28 [read-only access]

Task2 - POU1.ACTION (1) %MB28 [write access]

结果显示在信息窗口中。

用户组

在 CoDeSys 中，能建立八个对 POUs、数据类型、可视化和资源有不同访问权限用户组，能建立访问单个对象或全部的对象权限的用户组，只有特定的用户组才能打开工程文件，可以通过密码来识别这样的用户组。

用户组从 0 到 7 编号，0 组拥有管理员权限，例如，只有 0 组的成员才能决定所有的组和/或对象的密码和访问权限。

当一个新工程开始，所有的密码初始化为空，直到为 0 组设置了密码，输入工程的成员自动被认为是 0 组的成员。

在工程被加载时，如果存在一个用户组 0 的密码，当工程打开时需要所有的组的密码，会出现下面的对话框：

用户组密码对话框.

在对话框中的左边的组合框用户组中，输入所属的组并在右边输入相关的密码，按 OK，如果密码与存储的密码不相符合，会出现一个消息：“密码不正确。”

只有当你输入正确的密码时才能打开工程。

用命令’用户组密码’能赋予密码，用命令’工程’’访问权限’能对单个对象或对所有的对象定义权限。

’工程’’用户组密码’

使用这个命令你可以打开对话框为 用户组设置密码，这个命令只能由用户组 0 组的成员执行，当给出命令时，出现下面的对话框：

输入密码对话框.

在左边用户组下拉选项中选择用户组，在密码区域为用户组输入期望的密码，对每个输入的字符在区域

中显示一个*号。你必须在确认密码区域中重复输入相同的密码，在密码输入后按'确定'关闭对话框，如果你得到消息：“密码和确认密码不一致”，那么这两个输入项不一致，在这种情况下重新输入它们。

如果必要，通过再次调用命令为下一个组分配一个密码。

注意：如果任何用户组都没有设置密码，那么任何用户组都可以打开此工程文件。

使用命令'工程''访问权限'对单个对象或全部对象来分配权限。

4.3.1 '工程''数据库连接'

'工程''数据库连接'

在工程选项对话的类别'工程源控制'中激活选项“使用源控制 (ENI)”，这个菜单项目可用，在子菜单里能找到下面的处理对象和当前连接的 ENI数据库中工程的命令

登录（用户登录到ENI服务器中）

如果在对象管理器中一个对象被标记并且执行了命令数据库连接（按鼠标右键弹出上下文菜单），可用到下面的命令来执行数据库动作。如果用户以前没有登录到ENI服务器，那么自动打开 '数据库登入'对话，直到登录成功选择的命令才执行：

定义

获得最新版本

校验确认

校验

取消校验确认

显示不同处

显示版本历史记录

如果激活了菜单”工程”中的命令“数据库连接”，附加的菜单条目可用，它涉及到了工程中的所有对象：

多重定义

得到全部最新的版本

多重校验确认

多重校验

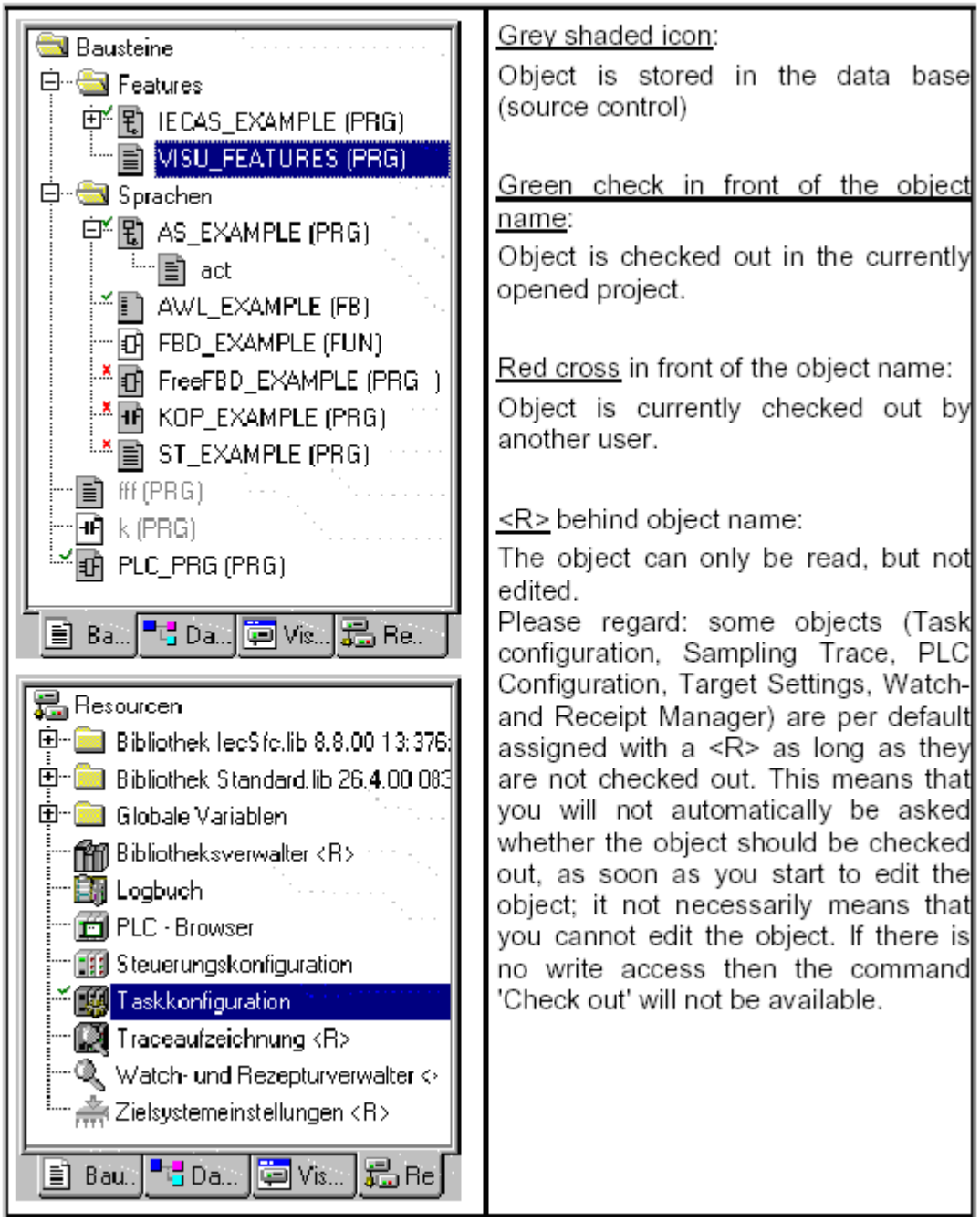
取消多重校验

工程版本历史记录

标签版本

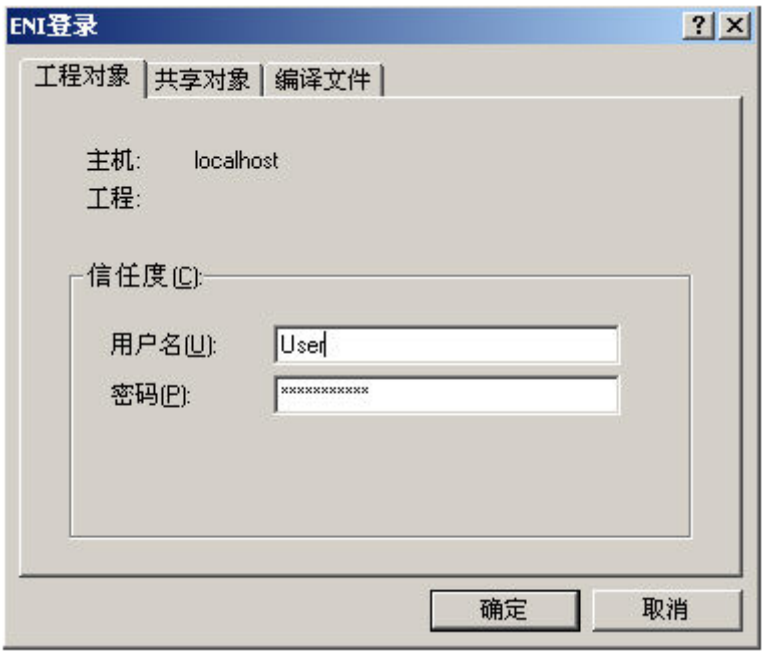
添加共享对象

更新状态



登录

使用这个命令打开对话“登录”，在这里通过 ENI 服务器能为 ENI 数据库输入访问数据，访问数据也必须在 ENI 服务器中定义(ENI 管理，用户管理）和依赖当前使用的数据库，也必须在数据库的用户管理中定义，在命令执行后，首先会打开类别“工程对象”的登录对话。



显示了下面的选项：

数据库:工程对象

主机：ENI 服务器运行的计算机的地址（必须和在工程选项中“工程源控制”中的条目区域“TCP/IP 地址”相匹配）。

工程:数据库工程的名字（必须和在工程选项中“工程源控制/种类工程对象”中的条目区域“工程名称”相匹配）。

许可：

•输入用户名和密码。

•激活工程的选项使用自定义，当前 CoDeSys 工程和数据库涉及当前类别中的对象进行通讯，上面输入的访问数据将自动使用。

•按 OK 键来确认设置。对话关闭的同时’共享对象’登录对话打开。以在’工程对象’讲述的相同的方式输入访问数据并按 OK 确认。在第三个为打开类别’编辑文件’ 的登录对话中输入访问数据。

•在象上面成功登录之前只要你试图访问数据库，登录对话将一直打开。

注意：如果你想保存把访问数据和工程一起保存，激活工程选项中类别 ’加载 & 保存’中的选项“保存ENI信用度”。

定义

命令：“工程” “数据库连接”“定义”

可以定义当前在对象管理器中标记的对象应该保存在数据库中还是保存在工程中，对话打开，你可以选择其中的一个 数据库类别“工程” 或“共享对象”或类别“本地的”。

数据库中管理的对象的图标在对象管理器中显示为灰黑色。

获得最新版本

命令：“工程” “数据库连接”“获得最新版本”

在对象管理器中标记的对象的当前的版本从数据库中复制并覆盖本地的版本，与 隔离 动作相反，对于其它用户数据库中将不锁定对象。

读出

命令：“工程” “数据库连接” “读出确认”

在对象管理器中标记的对象将从数据库中被选中并对其它用户锁定。

当执行命令时，用户得到对话框“读出对象”，注释保存到数据库中对象的版本历史记录中。

在按 OK 关闭对话以后，在本地工程的对象管理器中选中的对象将标记一个绿色的对号，对于其它的用户将出现标记一个红色的叉号，他们不能编辑这些对象。

写入

命令：“工程” “数据库连接”“写入”

在对象管理器中标记的对象将会登记到数据库中，因而在数据库中会创建对象的新版本，旧版本也会一直保留。

当执行命令时用户得到对话框“写入对象”，注释保存到在数据库中对象的版本历史记录中。

在按 OK 关闭对话后，在对象管理器中的对象名附近的绿色的对号将会消失。

取消读出

命令：“工程” “数据库连接” “取消读出”

使用这个命令来取消对象管理器中当前标记的对象的选中，因而也包括本地产生的对象的修改也会被取消，不出现对话框。在数据库中保留对象的未改变的最新版本并且它可以被其它用户访问，在对象管理器中在对象名附近的红色的叉号将消失。

显示不同处

命令：“工程” “数据库连接”“显示不同处”

在CoDeSys中用户当前打开的对象在一分为二的窗口中显示，本地用户正编辑的本地版本与在数据库中最新（当前）版本相对比，版本之间的差别将象在工程比较中讲述的那样标记出来（查看’工程’’对比’）。

显示版本历史记录

命令：“工程” “数据库连接” “显示版本历史记录”

对于当前在对象管理器中标记的对象打开对话“版本历史记录 <object name>”，在数据库中已登记的或 标注的对象的所有版本都被列出来。

给出了下列信息：

版本:为在数据库中一个接一个登记的对象的版本编的特殊的编号，标注的版本没有版本号但被一个标注图标标记。

用户: 执行登记或标注动作的用户的名字。

日期:动作的时间和日期

动作:已经执行的动作的类型，能的类型：“创建”(对象首次已经登记到数据库中)，登记（除了首次登记的对象之外的所有对象）和“标签〈label〉”（分配给对象的这个版本一个标签)。

按钮：

关闭:关闭对话

显示:表中当前标记的版本在CoDeSys的窗口中打开,标题栏显示“ENI:〈数据库中的工程名称〉/〈object name〉”。

对话:对话“历史版本资料”打开：

File（文件)（工程的名字和数据库中的对象)、版本（查看上面)、日期（查看上面)、用户、注释（当对象已经登记或已经标注时注释插入)。使用按钮下一个或上一个来在对话“版本历史记录 ..”中的下一个或前一个条目的详细窗口之间跳转。

获得最新版本:在表中标记的版本会被加载到CoDeSys中并覆盖当前的版本。

差别:如果在表中只有对象的一个版本被标记，这个命令将使这个版本和最新的数据库版本进行比较，如果标记了两个版本，那些版本将进行比较，差别象在 工程比较中那样显示在一分为二的双向窗口中。

复位版本:在表中标记的版本将被设置为最新版本，已经选中的所有的版本将会被删除！这对恢复一个对象的早期的状态很有用。

只标签:如果这个选项激活，对象的用标注标记的版本在表中显示。

选择框在选项'只标签'下：这里能找到为当前工程的对象执行任何数据库动作的所有用户的名字，如果你想得到涉及到所有用户或一个特定用户的版本历史记录，选择“All”或其中的一个名字。

多重定义

命令“工程” “数据库连接” “多重定义”

如果你想把位于一个单列的几个对象分配给一个特定的 数据库类别，可以使用这个命令。象命令’定义’讲述的那样打开对话框“工程”。选择期望的类别并按“OK”键关闭对话框。随后打开对话框“ENI选择”。对话框中列出了所有的工程文件中的POUs（例如：如果你选择了类别 “共享对象”，选择窗口只提供带有资源标签的的POUs)。POUs以树形结构显示，与在对象管理器中的POUs相同，选择期望的POUs并按“OK”键确认。

获得所有最新版本

命令 “工程” “数据库连接” “获得所有最新版本”

当前打开工程的每个对象的最新版本，在源控制下保留，将会被数据库调用。考虑以下方面：

•如果附加的对象已经存储在数据库工程文件夹中，那些对象将添加到CoDeSys中的本地工程中。

•如果对象在数据库中已经删除，那些对象将会在本地工程中删除，但它们自动分配到 类别“加载”中。

•类别“Shared Objects”中的对象如果已经在本地工程中可用，他的最新的版本只能被调用，了解更多的信息查看命令’获得所有最新版本’。

多次读出

命令 “工程”“数据库连接” “多次读出”

可以选择多个对象，“ENI选择”对话打开，列出工程的所有的POUs，选择那些应选中的POUs并按OK确认。了解更多的信息参看命令 ’隔离’。

多次写入

命令“工程” “数据库连接”“多次写入”

可以同时登记多个对象，对话框“ENI选择”打开，列出工程的所有的POUs，选择那些应登记的POUs并按

OK确认。了解更多的信息参看命令’登记’。

取消多次读出

命令“工程” “数据库连接” “取消多次读出”

可以同时取消选中的动作，对话框’ENI选择’打开，列出了工程的所有的POUs，选择那些你想取消选中的POUs并按OK确认。更多的信息请查看命令’撤销校验’。

工程变量历史记录

命令 “工程” “数据库连接” “工程变量历史记录”

如果选择的数据库系统支持那个功能性，就能使用这个命令来查看当前打开的工程的版本历史记录。

对话“历史记录<数据库工程名称>”将打开，它以时间顺序显示了为工程中的特殊的对象执行的动作(创建、登记、标注)，对象的整个数量显示在变量历史记录后面。对话可以象命令 ’显示变量历史记录’那样来处理，但要注意下面的事情：

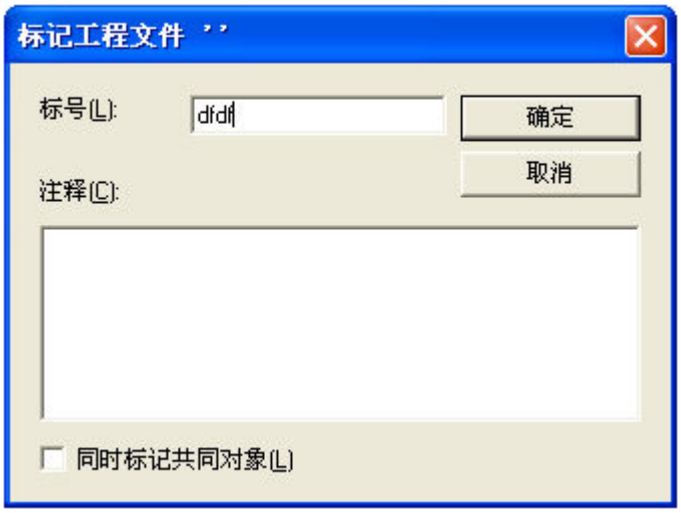
•命令“复位变量”只能对单个对象使用。

•命令“获得最新版本”是当前被标记的对象的版本的所有对象将会调用到本地工程中！在 CoDeSys 中的对象将会被旧版本中的对象覆盖。但：本地对象，在旧版本中还不是工程的一部分，将不会从本地工程中移除！

版本标签

命令“工程” “数据库连接” “版本标签”

这个命令用来在工程的每个对象的当前版本上加一个"标签"，这个工程版本能在稍后被调用。对话框’标识 <数据库工程名称>’打开，插入一个标签（例如，"强制变量"）和可选的一个注释，当按’确定’后，对话框关闭,标签和动作"带标签的<label name>"会出现在版本历史记录的表中，单个对象的历史记录和工程的历史记录一样。工程标签的版本没有版本号，但是在列“变量”中标记为一个标签图标，如果在版本历史记录对话中激活选项“只显示有标签的”，将只列出标注的版本。



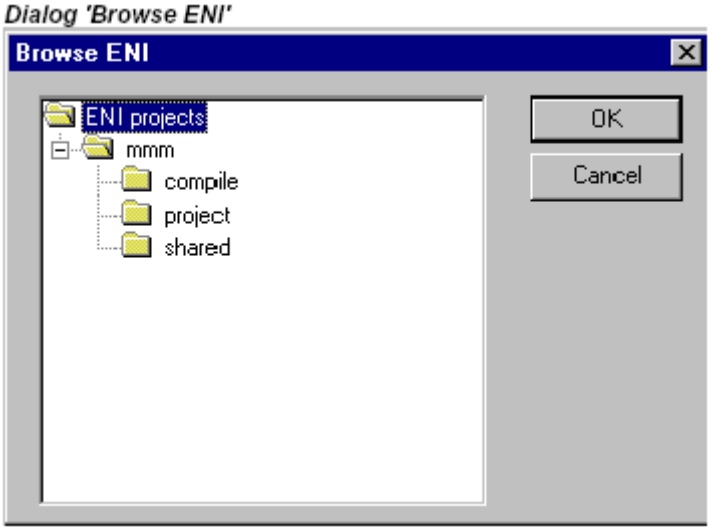
添加共享对象

命令 “工程” “数据库连接”“添加共享对象”

使用这个命令来添加新的 数据库类别“共享对象”中的对象到CoDeSys中本地打开的工程中。对于类别“工程对象”的对象，这是没必要的，因为命令“获得(所有)最新版本”自动调用在数据库工程文件夹中的所有对象，但对于类别“共享对”的对象在这种情况下只有那些在本地工程中可用的对象才被调用。

执行命令“添加共享对象”来打开对话“浏览 ENI”，窗口右边部分的列表显示了在左边部分的列表中当前被选择的数据库文件夹中用到的所有对象，选择期望的对象并按 OK 或在条目上双击来插入对象到当前打开的 CoDeSys 工程中。

Dialog 'Browse ENI'



更新状态

命令 “工程”“数据库连接”“更新状态”

使用这个命令来更新在对象管理器中的显示，因此你能看到工程的关于源控制的对象的当前的状态。

4.4 管理工程中的对象

管理工程中的对象

在以下的章节中，读者可以了解到如何进行对象的管理，以及在对象轨迹（文件夹、调用树、参考表等）上可以获得帮助等信息。

参照:

对象

文件夹

’新建文件夹’

’展开节点’ ’收缩节点’

’工程’ ’对象删除’

’工程’ ’对象添加’

’工程’ ’对象重命名’

’工程’ ’对象转换’

’工程’ ’对象复制’

’工程’ ’对象打开’

’工程’ ’对象访问权限’

’工程’ ’数据库连接’

’工程’ ’对象属性’

’工程’ ’添加动作’

’工程’ ’打开实例’

’工程’ ’显示调入树’

’工程’ ’显示交叉参考’

对象

POUs、数据类型、可视化和资源全局变量、变量配置、采样追踪、PLC配置、任务配置和观察和接收管理器等，都可以被定义为“对象”。其中部分包括用于构造工程的被插入的文件夹。工程的所有对象都处于 对

象管理器之中。

鼠标光标在对象管理器中的一个 POU（包括程序、功能和功能块）上停留几秒，就会在工具条中出现有关 POU 类型的提示。对于全局变量工具条上将显示关键字(VAR_GLOBAL, VAR_CONFIG)。

在一个对象类型内，可以通过拖拉进行对象（和文件夹，参看 '文件夹'）的移动。选择对象并按住鼠标左键，可以把对象移动到期望的地点。如果移动中出现导致名称冲突现象，新引进的元素将会通过附加一个连续的数字来进行标识（例如，“Object_1”）。

文件夹

为了进行大工程的管理，应该在文件夹中对 POUs、数据类型、可视化和全局变量进行分组。

可以创建任意多级别的文件夹。如果在一个收缩的文件夹  前面有一个加号，说明这个文件夹中包含对象和附加的文件夹。在加号上点击，就可以在打开文件夹的同时显示子对象。在减号（替代了加号）上点击，文件夹又关闭了。在文本菜单中也能找到相同的功能的命令''展开节点' '收缩节点''。

通过拖拉，可以移动对象，连同在它们对象类型内部的文件夹。方法是选择对象并按住鼠标左键，把它拖放到期望的位置。

通过命令 '新建文件夹'，可以创建更多的文件夹。

注意：文件夹对程序没有影响，只为了清晰的表达工程。

对象管理器中的文件夹

'新建文件夹'

使用这个命令，可以以一个结构化对象的方式插入一个新 文件夹。如果已经选择了一个文件夹，就可在其下面创建新文件夹。否则就在同级创建新文件夹。如果选择了一个动作，新文件夹将创建到与POU同级的动作POU中。

对象管理器的文本菜单包含了这个命令，选择一个对象或对象类型时，按鼠标右键或<Shift>+<F10>出现这个命令。

最新插入的文件夹最初有名称“新建文件夹”，应遵守下面的文件夹命名的约定：

•在层次中同级的文件夹必须有唯一性，在不同层次的文件夹可以具有相同的名字。

•对象在相同层次时文件夹中不能有相同的名字。

•如果在相同的层次上已经存在一个名为“新建文件夹”的文件夹，文件夹名字后面自动附加连续的数字（例如，"新建文件夹 1"），不能重命名为一个正在使用的文件夹名字。

'展开节点' '收缩节点'

用这个命令展开所选对象下包含的子对象,使它们可见。

使用收缩命令从属的对象将不再显示。

通过鼠标双击或按<Enter>键可以打开或关闭 文件夹 。

对象管理器的内容菜单中包含这个命令，当一个对象或对象类型被选中时并且你按了鼠标右键或<Shift>+<F10>时，这个命令出现。

'工程' '对象删除'

快捷方式 ： <Delete>

当前选中的对象（POU、数据类型、可视化或全局变量），或带从属对象的文件夹将从对象管理器中删除从而从工程中删除，对象的删除可以通过命令“编辑” “取消”来恢复。

删除的对象可以通过命令“编辑” “取消”来恢复。

如果对象的编辑器窗口是打开的，它自动关闭。

使用命令 '编辑' '剪切' 来删除，对象被放置于剪贴板上。

'工程' '添加对象'

快捷方式 ： <Insert>

用这个命令可以创建一个新对象，对象（POU、数据类型、可视化或全局变量）的类型依赖于在对象管理器中被选择的选项，“数据类型”、“功能”、 “功能块”或“程序”类型的对象能使用一个 模板来创建。

在出现的对话框中输入新 POU 的名字，对象的名字可以是没有使用过的名字。

注意下面的限制：

•POU 的名字中不能包含空格

•一个 POU 不能和另一个 POU 或数据类型具有相同的名字。为了避免可视化改变带来的问题,也不应与可视化对象具有相同名字.

•数据类型不能与其它数据类型或 POU 有相同的名字。

•全局变量列表不能与其它的全局变量列表有相同的名字。

••在同一 POU 中动作和其它的动作有相同的名字。

••可视化不能与其它的可视化有相同的名字。为了避免可视化改变带来的问题,也不应与 POU 具有相同名字.

在其它情况下，同一命名是允许的，例如动作属于不同的 POUs 可以有相同的名字，可视化可以和 POU 同名。

POU，POU 类型（程序，功能或功能模块）和编程的语言必须被选择，“程序”是 POU 的类型的默认值，POU 的语言的默认值是最近创建的 POU 的语言，如果创建了一个功能类型的 POU，期望的数据类型必须输入到返回类型文本输入区域，这里所有的基本的和定义的数据类型（数组，结构体，枚举类，别名）是允许的，允许使用输入帮助（按<F2>）。

添加对象对话框

只有在按上面的命名规则没有冲突之后才能按 OK，在对象管理器中创建了新对象并且输入窗口出现。

使用命令“编辑”“插入”，不出现对话，在剪贴板上的对象插入。如果插入的对象名字和命名规则冲突（查看上面），附加带一个下划线字符的一系列数字来唯一标识。（例如，″Rightturnsig_1″）。

在一个ENI数据库中工程是在源控制下（依赖于在工程选项对话中’工程源控制’的设置），自动询问你想在那个数据库类别中处理新对象，在这种情况下打开“属性”对话框，在这里你能将一个对象分配到 数据库对象类别中。

’另存为模板’

“数据类型”、“功能”、 “功能块”或“程序”类型的对象能保存为模板，在对象管理器中选择对象并在上下文菜单（鼠标右键）选择命令“另存为模板”。相同类型的 新对象自动得到最近创建的对象模板的声明部分。

’工程’ ’重命名对象’

快捷方式 ：〈Spacebar〉

为当前选择的对象或 文件夹起一个新名字，对象的名字可以是还没有使用过的。

如果对象的编辑窗口打开，当对象名字改变时它的标题也跟着自动改变。

重命名POU对话框

’工程’ ’对象转换’

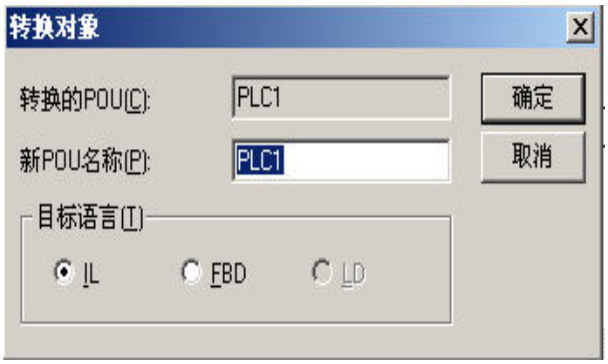
这个命令只能在 POU 中使用，可以从 SFC、 ST、FBD、 LD 和 IL 语言转换成 IL, FBD, 和 LD 三种语言中的一种。

工程必须被编译，选择你想转换到的语言并给 POU 起一个新名字，POU 的名字可以是还没有使用过的，然后按 OK，新 POU 添加到了 POU 列表中。

在转换过程中处理的类型与应用于编译的类型相一致。

注意：动作不能转换。

转换POU对话框



注意下列可能性:在FBD编辑器中创建的POU，能使用命令’附加’ ’查看’ 且不需要任何转换就能在KOP编辑器中显示和编辑.

’工程’ ’对象复制’

使用这个命令可以复制一个对象并存为一个新名字，在结果对话框中输入新对象的名字，对象的名字可以是没有使用过的名字。

另一方面，使用命令“编辑”“复制’，将对象复制到剪贴板上,此时不出现对话框，。

对象复制对话框

’工程’ ’打开对象’

快捷方式 ： 〈Enter〉

使用命令把在 对象管理器中选择的对象加载到各自的编辑器中，这个对象的窗口已经打开，将会放置在显著的位置并能编辑。

这里有两种打开对象的其它方法：

•在期望的对象上双击

•如果在对象管理器中敲入对象名字的首字母，则打开一个对话框,在这里显示了以这个字母为开头的所有对象，动作用符号〈POU 名称〉.〈动作名称〉列出。在对象选择框中的对象按字母顺序列出的，POU 的动作经常位于这个 POU 下面，选择期望的对象并按按钮 Open 把对象加载到它的编辑窗口中。在对象上以分级方式放置的所有的文件夹将会展开，资源中的全局变量也支持这个选项.

打开对象对话框

’工程’ ’对象属性’

在对象管理器中,显示当前选中对象的“属性”对话框 。

执行命令’工程’ ’对象访问权限’也将打开此对话框。

依据对象和工程的设置,附加的页面可用于定义对象的属性:

全局变量列表:

在全局变量列表中,显示列表中的参数和与网络变量数据交换的参数,可以在此修改它们.如果在对象管理器中,选择"全局变量"中的某一项来 创建一个新的全局变量列表并执行"添加对象",也将打开此对话框.

可视化:

在’可视化界面’下,可以定义可视化对象的属性,设置方式如下:

作为网页可视化界面和目标可视化界面: 如果在目标系统中设置了网页可视化和目标可视化,在这里你将界面设置成网页可视化界面和目标可视化界面.

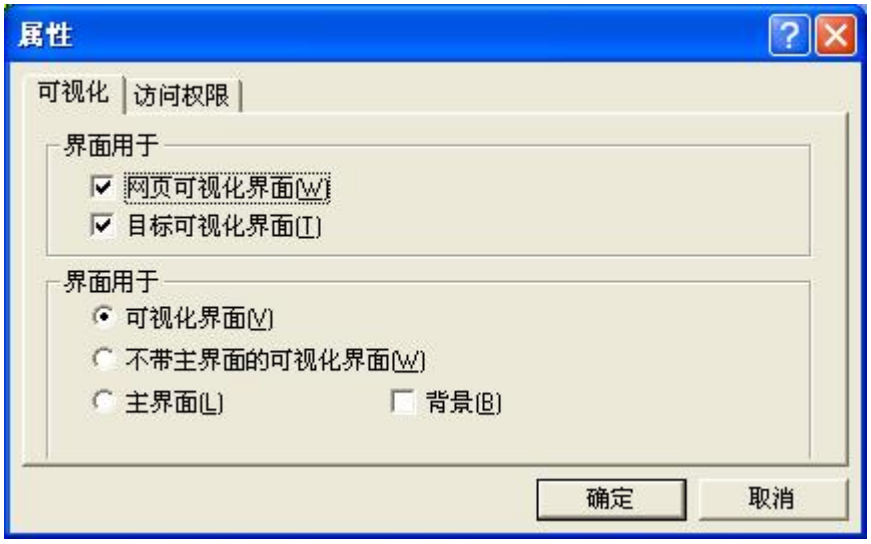
作为: 下面是涉及’主界面’应用的设置

可视化界面: 此界面作为一般的可视化界面.

可视化界面不带主界面：如果在工程中定义了主界面,此界面将不使用主界面.

主界面：此界面将作为主界面.在默认方式下,主界面总是在一个可视化界面的前面,除非将它设置成背景.

可视化界面属性对话框



数据库连接

•如果工程连接到ENI数据库（参看 ’工程’ ’选项’ ’数据库连接’），那么在属性对话框中将出现“数据库连接”页面，这里你能显示和修改来自数据库给此对象的赋值 。更多信息参看 ’什么是ENI?’。

’工程’ ’对象访问权限’

打开给不同 用户组分配访问权限的对话框：

访问权限对话框

用户组 0 的成员现在为每个用户组分配单个的访问权限，有三个可能的设置：

•不可访问:用户组的成员不能打开对象。

•只读访问:用户组的成员只能查看对象,但不能修改它们。

•完全访问:用户组的成员能够打开并修改对象。

如果选中’应用到全部组件’,工程中的所有POU，数据类型，可视化和资源的设置都将采用在 对象管理器中当前选中对象的设置.

如果给用户组设置了密码，打开工程时将要求输入密码。

请注意:在设置访问权限时,要考虑涉及到可视化元件（可视化，安全性）操作的可能性。

’工程’ ’添加动作’

在 对象管理器中这个命令用来给所选POU添加一个动作，在出现的对话框中输入动作的名字和选择编程的语言。

在对象管理器中新动作置于块的下面，在块的前面出现一个加号，在加号上点击出现动作对象并且在块的前面出现一个减号，重新在减号上点击动作消失,加号又出现。这也能在上下文菜单命令 ’展开节点’ ’收缩节点’中完成。

’工程’ ’打开实例’

用这个命令打开并显示一个在对象管理器中选中的功能块的实例;另外,在对象管理器中双击功能块上将打开一个选择对话框，在这个对话框中列出了功能块的实例和执行代码，在这里选择期望的实例或执行代码并按确认,所选项将显示在窗口中。

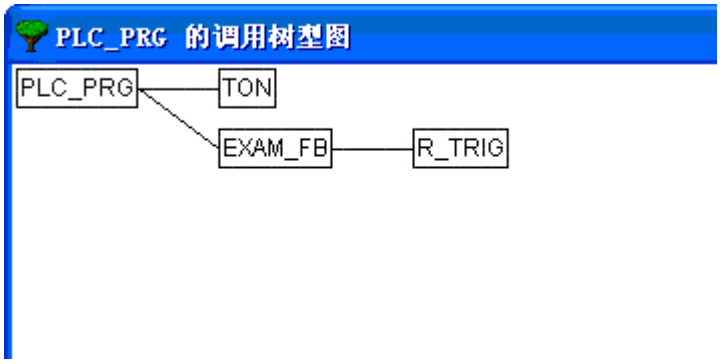
请注意：如果你想查看实例，你首先要登录!（工程经过编译没有错误并通过命令’联机’ ’登录’加载到PLC中）。

打开实例对话框

’工程’ ’显示调用树’

用这个命令你能打开一个显示了在 对象管理器中选中的对象的调用树的窗口，工程必须编译（参看’Rebuild all’），调用树包含了POU的调用和引用的数据类型。

显示调用树的例子



’工程’ ’显示交叉参考’

使用这个命令，打开一个对话框，此对话框可以列出所有变量、地址或POU的输出点。执行这个命令前，工程必须先编译（请参看’工程’ ’编译生成’）。

首先选择类别“变量”,“地址”,或 “POU”,然后输入期望的元素的名字(或 F2)，在名字中输入一个″*″可以得到相应类别的所有元素 。

如果在上次编译后,工程作了修改,那么在对话框的标题中显示’未更新’.在这种情况下,如果不重新编译,那么在交叉参考列表中将不包含修改部分.

按’获得参考’按钮，得到所有应用位置的列表，和 POU 相连的是行号或网络编号、变量名和绑定的地址，范围空间显示了变量是局部的还是全局的；访问列显示了变量在当前位置上是能读还是能写，列的宽度根据条目的长度自动调整。

当选择了交叉引用列表中的一行并按’转到’按钮或在行上双击，包含此行的 POU 在编辑器中显示，通过这种方式你不需要耗时的搜索就能跳转到所有的应用位置。

为是使处理过程简化，使用发送到消息窗口按钮把当前的交叉引用列表移到信息窗口，从 信息窗口中切换到相应的POU中。

对话框和显示交叉参考的例子

4.5 编辑功能

在所有的编辑器中你能使用下面的命令和在对象管理器中使用部分命令，命令位于菜单“编辑”下面和鼠标右键打开的上下文菜单中。

如果计算机上安装了 IntelliPoint 软件，CoDeSys 支持微软智能鼠标的所有功能。所有的编辑器具有缩放功能：按<Ctrl>键并滚动鼠标轮来放大，想缩小的话，按<Ctrl>键并向后滚动鼠标轮。

参照:

’编辑’ ’撤消’
’编辑’ ’重复’
’编辑’ ’剪切’
’编辑’ ’复制’
’编辑’ ’粘贴’
’编辑’ ’删除〞
’编辑’ ’查找’
’编辑’ ’查找下一个’
’编辑’ ’替换’
’编辑’ ’输入助手’
非结构化显示
结构化显示
’编辑’’自动声明
’编辑’ ’下一个错误’〞
’编辑’ ’前一个错误’〞

’编辑’ ’宏’

’编辑’ ’撤消’

快捷方式 ： <Ctrl>+<Z>

这个命令用来取消在当前打开的编辑器窗口中或在对象管理器中最近执行的动作；重复使用取消动作，所有的动作后退到窗口刚打开时的动作。这个命令适用于编辑器中对 POU，数据类型，可视化和全局变量所做的操作,和在对象管理器中所做的操作。

用’编辑’ ’重复’你可以恢复一个已经撤消的动作。

注意：命令撤消 和 重复适用于当前窗口。每个窗口带有自己的动作列表。如果你想在几个窗口中撤消动作，那么必须激活相应的窗口。当在对象管理器中撤消和重复时,焦点必须位于这里。

’编辑’ ’重复’

快捷方式 ：<Ctrl>+<Y>

在当前打开的编辑器窗口或在 对象管理器中使用这个命令时，能恢复已经撤消的动作(’编辑’ ’撤消’)。

每当你先前执行了命令“撤消”，你可以执行命令“重复”。

注意：命令撤消 和 重复适用于当前窗口。每个窗口带有自己的动作列表。如果你想在几个窗口中撤消动作，那么必须激活相应的窗口。当在对象管理器中撤消和重复时,焦点必须位于这里。

’编辑’ ’剪切’

符号：  **快捷方式：<Ctrl>+<X>** or **<Shift>+<Delete>**)

执行这个命令，可以把当前的选择对象从编辑器转移到剪贴板中。即选择的对象从编辑器中移除。

此命令也可以在 对象管理器 中应用，不是所有的对象都能剪切，如 PLC配置。

并不是所有的编辑器都支持剪切命令，它在某些编辑器中应用时会受到限制。

选择的形式依赖各自的编辑器：

在文本编辑器 IL, ST和 变量声明中，选择的是一列字符。

在 FBD和 LD编辑器中，选择的是由虚线框包围的网络、带输入的框、框和操作数。

在 SFC编辑器中选择的是虚线框包围的一系列步。

使用命令 ’编辑’ ’粘贴’来粘贴剪贴板的内容，在SFC编辑器中你也能使用命令 ’附加’ ’插入并行分支(右侧)’”或’附加’ ’粘贴在下面’来做这些。

使用命令’编辑’ ’复制’复制选择到剪贴板上而不删除它。

为了不改变剪贴板的内容而移除选中的区域，使用命令’编辑’ ’删除’来完成。

’编辑’ ’复制’

符号：  快捷方式：<Ctrl>+<C>

执行这个命令，可以将当前的选择从编辑器复制到剪贴板上，它不改变编辑器窗口的内容。

这个命令同样应用于被选择的对象，不是所有的对象都被复制。例如，PLC 配置。

不是所有的编辑器支持复制，它在某些编辑器中应用时会受到限制。

对于选择的类型与使用命令“编辑” “剪切”具有相同的规则。

选择的形式依赖于各自的编辑器：

在文本编辑器(IL, ST和 变量声明)中选择是一列字符。

在 FBD和 LD编辑器中，选择的是由虚线框包围的网络、带输入的框、框和操作数。

在 SFC编辑器中选择的是虚线框包围的一系列 步。

使用命令 ’编辑’ ’粘贴’来粘贴剪贴板的内容，在SFC编辑器中你也能使用命令 ’附加’ ’插入并行分支(右侧)’”或’附加’ ’粘贴在下面’来做这些。

使用命令“编辑”“复制”复制选择到剪贴板上而不删除它。

为了删除一个选中区域同时把它放到剪贴板上，使用命令'编辑''剪切'来完成。

'编辑''粘贴'

符号：  快捷方式：<Ctrl>+<V>

在编辑器窗口中把剪贴板上的内容粘贴到当前位置上，在 图形化编辑器中当插入产生一个正确的结构时命令才执行。

可以使用 对象管理器来从剪贴板上粘贴内容。

不是所有的编辑器都支持粘贴，它在某些编辑器中应用时会受到限制。

根据编辑器的类型不同，当前位置也有不同的定义。

在文本编辑器中（IL, ST，声明）当前位置是光标闪烁（一个垂直的直线）的地方，可以单击鼠标把光标定位到这里。

在 FBD和 LD编辑器 中，当前的位置是在网络编号区域中带虚线框的的第一个网络，剪贴板的内容插入到这个网络的前面，如果复制一部分结构，它将插入到选中元素的前面。

In the SFC editor the current position is determined the selection which is surrounded by a dotted rectangle. Depending upon the selection and the contents of the clipboard, these contents are inserted either in front of the selection or into a new branch (parallel or alternative) to the left of the selection.

在 SFC编辑器 中，当前位置是由虚线框包围处决定，依赖于选择和剪贴板的内容，这些内容或者插入到选择的前面或者作为一个分支（平行或可选分支）插入到选择位置的左边。

在SFC中可以使用命令'附加''插入平行分支 (权利)' 或'附加''粘贴'来插入剪贴板的内容。

用命令'编辑''复制'来复制选择到剪贴板上而不删除它的内容。

用命令'编辑''删除'来删除一个选中的区域而不改变剪贴板内容。

'编辑''删除″

快捷方式: <Del>

从 编辑窗口中删除选中的区域，这不改变剪贴板中的内容。

对于选择的类型，与使用命令“编辑”“剪切”的规则相同。

选择的形式依赖于各自的编辑器：

在文本编辑器中（IL, ST，声明）选择的是一列字符。

在 FBD和 LD 编辑器中，选择的是在网络编号区域内的虚线框包围的许多网络。

在 SFC编辑器中选择的是虚线框包围的一系列 步。

在 库管理器中选择是当前选中的库名。

使用命令'编辑''剪切'来删除一个选中的区域同时把它的内容放到剪贴板上。

'编辑''查找'

符号： 

用这个命令你可以在当前的编辑器窗口中搜索文本段。查找对话框一直打开，直到按取消键关闭。

在区域查找内容中输入你要查找的内容。

另外，你能决定要查找的文本是否完全匹配，或区分大小写，和从当前光标位置开始是向上搜索还是向下搜索。

按钮查找下一个开始在选中的位置上继续按选择的方向搜索。

如果找到文本信息，它被加亮显示，如果文本没有找到，出现消息说明未发现信息。搜索能连续地被重复执行，直到编辑器窗口中内容的开始或最后。在 CFC 编辑器中将会考虑元素的集合顺序，搜索将从窗口的左上角开始到右上角结束，请注意 FBD 中 POU 是从右到左处理的。

查找对话框

’编辑’ ’查找下一个’

符号 ：  快捷方式 ：〈F3〉

用这个命令与最后执行的’编辑’ ’查找’使用相同的参数执行搜索。

请注意在 FBD 中，POU 的搜索顺序是从右到左！

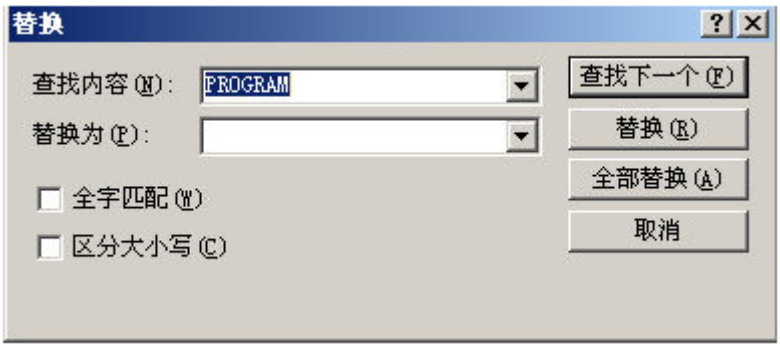
’编辑’ ’替换’

用这个命令与使用命令 ’编辑’ ’查找’一样查找一特定的信息，用其它的信息来替代它，在你选择了命令后查找和替换的对话框出现，直到按取消 或 关闭按钮才关闭这个对话框。

在编辑器中你标记过的字符自动作为查找内容，也可手工输入搜索字符串，按‘替换’将会用在区域替换为给出的字符串替换当前的，用‘查找下一个’找到下一个符合的字符。请注意，在 FBD 中，POU 是从右到左处理的。

按钮全部替换，将找到的字符全部替换，在处理过程最后报告替换了多少处。

查找和替换对话框



’编辑’ ’输入助手’

快捷方式 ： 〈F2〉

在编辑器窗口中使用这个命令将打开一个变量输入对话框，在左边列出了可选的输入类别，在右边列出了此类别下的条目，按‘确认’将选择插入到当前光标位置。

提供的类别依赖于当前光标在编辑器窗口中的位置。例如，那些可以在此处输入的（如，变量、操作数、POU、转换等）。

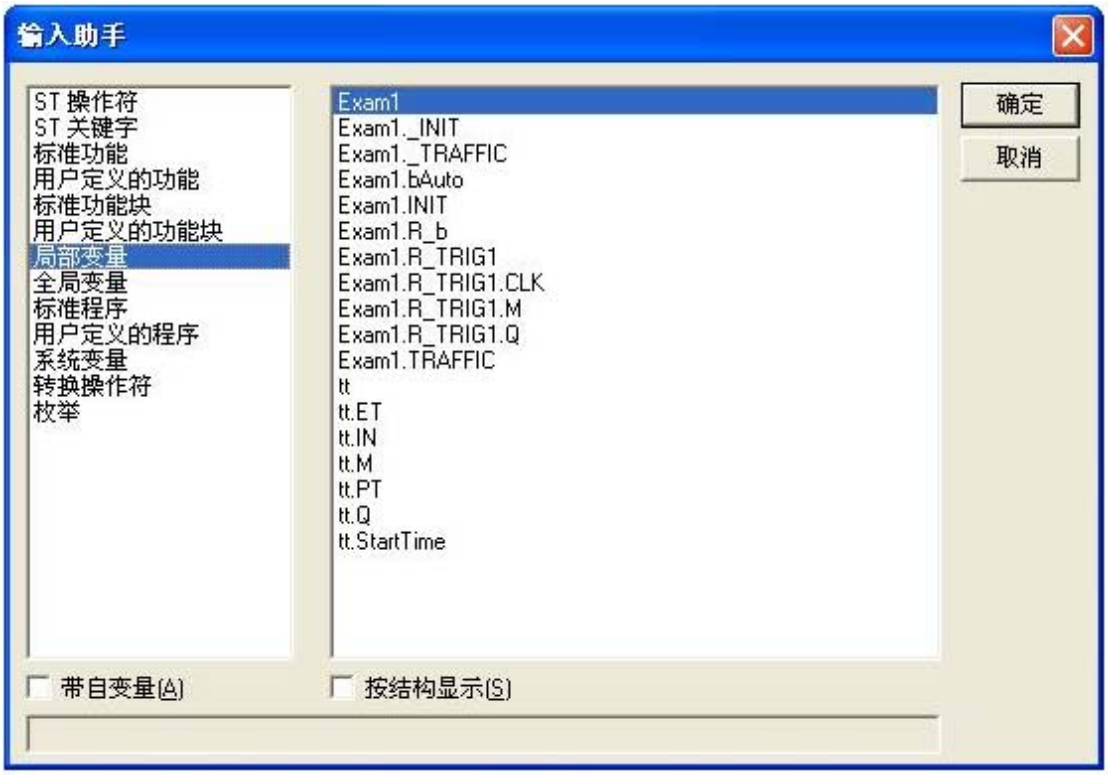
如果选项’带自变量’激活，当被选择元素插入时，元素所带变量也将插入，例如：功能块 ful 被选中，它定义了输入变量 var_in ，则在当前位置插入 ful(var_in:=)；功能 func1 的插入使用 var1 和 var2 作为传递变量 ： func1(var1,var2)。

按 结构化还是按 非结构化来显示变量可以通过激活选项’结构化显示’来切换。

注意：也能使用 智能功能来插入标识符.

非结构化显示

非结构化显示变量的输入助手对话框.



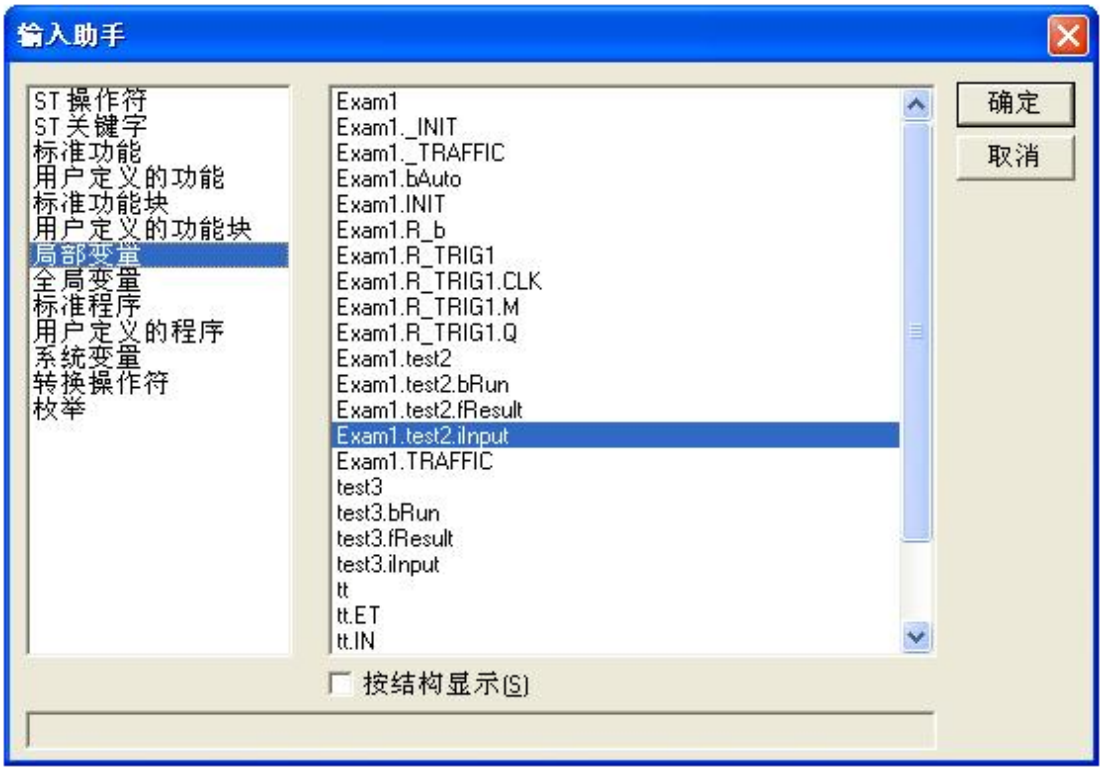
在每个类别中 POU、变量或数据类型只是简单地按字母顺序列出。

在不同的地方（比如，在监控列表中），需要多阶段变量，在那种情况下，输入帮助对话框显示所有 POU 列表和以点方式显示全局变量列表，在每个 POU 名字后有一个点。如果双击或按'确定'选中一个 POU，将打开它所包含的变量列表，如果 POU 中存在实例和数据类型，可以按级打开各自的变量列表，按'确定'选中的变量。

通过激活 按结构显示能切换到结构化显示。

结构化显示

结构化显示对话框



如果选中结构化显示，POU、变量或数据类型将按层次排列。对标准程序、标准功能、标准功能模块、用户定义的程序、用户定义的功能、用户定义的功能模块、全局变量、局部变量、定义的数据类型、监控变量也能按层次排列。可视化界面和它的层次显示与对象管理器中的显示相同；如果引用了库中的元素，这些元素按字母表的顺序插入在相应类别的最上面,并且相关层次显示与在库管理器中一样。

功能模块中被定义为局部或全局变量的输入输出变量在实例名下的“局部变量” 或 “全局变量” 里列出，选择实例名（例如，Inst_TP）按’确定’。

如果功能模块的实例被选中，可以选中选项’带自变量’。在文本语言 ST,IL 以及任务配置过程中，将插入功能模块的实例名和输入参数。

例如，如果选中 Inst (DeklarationInst: TON;) ，将插入：

Inst(IN:= ,PT:=)

如果选项没有被选中，只有实例名插入。在图形化语言编辑器或监控窗口中，通常只插入实例名。

结构组件的显示与功能模块实例类似。

对于枚举变量，单个的枚举值在枚举类型下面下列出。顺序是：库的枚举变量、数据类型的枚举变量、POU 的局部枚举变量。

通常的规则是包含子对象的行是不可选的（除了实例，参看上面），只能对自己的层次进行扩展显示或合拢，这与多阶段变量名使用一样。

如果在监控和配方管理器中或在采样跟踪配置对话框中选择跟踪变量时使用输入助手，有多种选择方式，按住<Shift>键时，你能选择一个范围的变量；当按住<Ctrl>键时，你能选择许多单个的变量，选中的变量被标记。在范围选择过程中不包含被选中的无效变量行，这些行将不会包含在选择中，当单个选择选好后，这些行不能被标记。

在监控窗口和采样跟踪配置中能够从输入助手对话框中选择结构，数组或实例。双击鼠标将展开和合拢所选元素的层次，在这些情况下只能按’确定’进行选择。

然后，选中的变量按行插入在监控窗口中，也就是说每个选中的变量独占一行。对于采样跟踪变量，每个变量在采样跟踪变量列表中独占一行.

允许的采样跟踪变量最多20个,如果选中的变量超过了20,会出现一个错误的信息“只允许20个变量”，

其余的选中变量不再插入列表中。

通过取消选项'按结构显示'来切换到 非结构化显示。

注意：一些条目只有在编译之后在输入助手对话框中才更新。

'编辑''变量声明'

快捷方式 ： 〈Shift〉+〈F2〉

这个命令打开一个 变量的声明对话框，使用菜单'工程''选项''编辑''变量声明'或当在声明编辑器中使用一个新的未定义的变量时，这个对话框会自动打开。

'编辑''下一个错误'″

快捷方式 ： 〈F4〉

工程编译出错后，这个命令用于显示下一个错误.执行此命令后显示包含错误的编辑器窗口并且标记出错误的位置，同时在 信息窗口显示相应的错误信息。

'编辑''前一个错误'″

快捷方式 ：〈Shift〉+〈F4〉

工程编译出错后，这个命令用于显示前一个错误.执行此命令后显示包含错误的编辑器窗口并且标记出错误的位置，同时在 信息窗口显示相应的错误信息。

'编辑''宏'

此菜单项将列出所有在工程中定义的宏(更多的信息请参看'工程''选项''宏'.当选中一个可执行的宏后,将打开对话框“执行宏”，对话框中显示宏的名字和当前执行的命令行，可以按'取消'按钮来取消宏的处理，当前命令的处理过程将无条件终止，在信息窗口中显示一条消息并且在联机操作过程中的日志中记录：“〈Macro(宏)〉: 执行被用户终止”。

可以联机或离线执行宏指令,但是只有在相应模式下允许的命令可以执行.

4.6 联机功能

菜单项“联机”下面包含了在联机时用到的命令，某些命令的执行依赖于当前激活的编辑器。

联机命令在登录之后可用，下面的章节将进行详细讲述。

“联机修改”功能能够对控制器上运行的程序做在线修改，参看'联机''登录'。

参看此处 图表，它表明了工程－编译生成、工程－下载、在线修改和登录到目标系统之间的关系。

'联机''登录'

符号：  快捷方式 ： 〈Alt〉+〈F8〉

这个命令把编程系统和PLC结合起来（或启动 仿真程序）并使程序进入联机模式。

如果当前的工程自从打开或自从上次修改后还没有编译，将先进行编译。如果在编译过程中有错误，CoDeSys 不会进入联机模式。

如果当前的工程在上次下载后在控制器上做了修改，并且上一次下载信息还没有用命令“工程”“全部清除”删除，那么在执行登录命令之后弹出一个对话框：“程序已修改，是否下载新程序吗？”，按'是'按钮后，工程中修改的部分将加载到控制器中(相关信息参看此处 图表，它表明了工程－生成、工程－下载、在线修改和登录到目标系统之间的关系。)，若选择'否'，程序不发生任何变化加载到控制器中，按'取消'取消命令，〈全部加载〉将把整个工程重新加载到控制器中。

扩展的登录对话框

如果选中'工程'-'选项'-'桌面'中的“安全模式下联机”,并且目标系统支持这个功能,按'详细'按钮后会扩展这个对话框：它将显示在 CoDeSys 中当前打开的工程的工程信息和当前加载到控制器中的工程的工程信息。

注意:联机登录对话框打开的样式依赖于哪个按钮设置为默认按钮。

注意:修改任务配置或 PLC 配置，或插入一个库以及执行命令“工程” “全部清除”（参看下面）后将不能进行联机修改。

注意:联机修改将不对变量进行重新初始化.当联机修改执行后保留变量一直保持它们的值，当工程重新下载后重新初始化它们(参看"联机""下载")。

在成功登录后，所有的联机功能都能使用（如果在“工程”“选项”中的类别'编译生成'中相应的设置都已经设置），将可以监控 变量声明中所有变量的当前值。

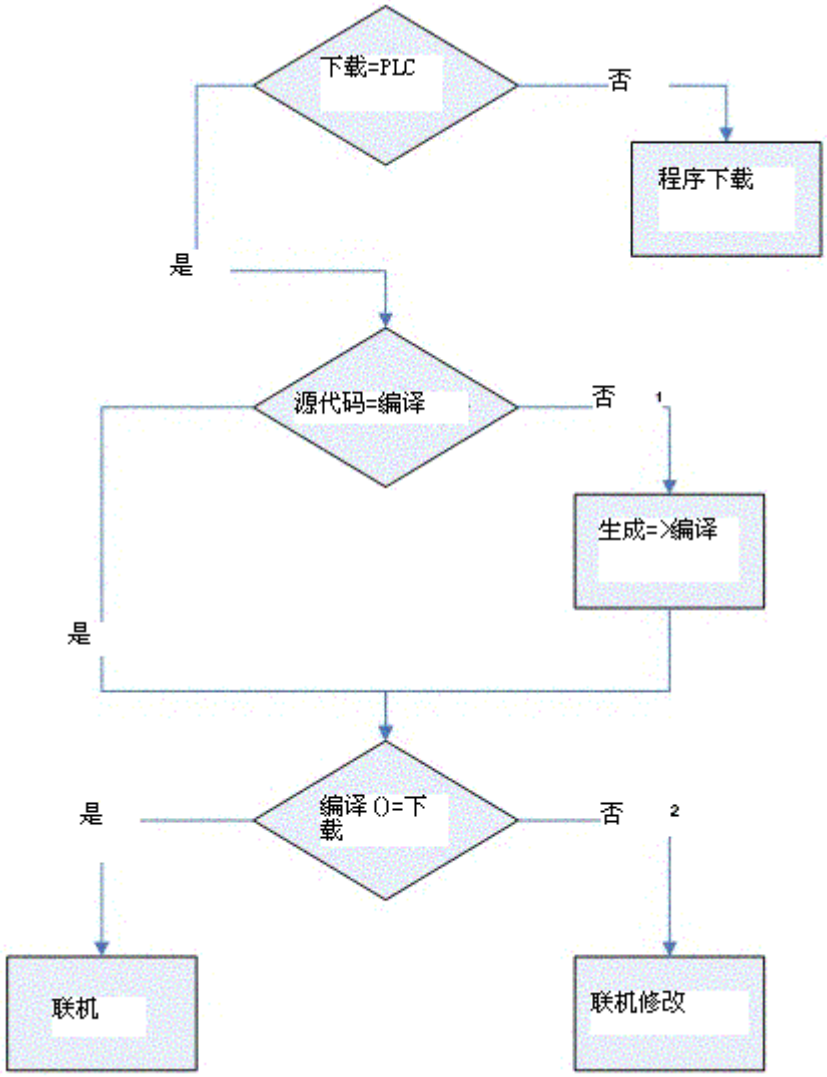
使用命令'联机' '退出'来退出联机模式。

登录、生成、下载和在线修改之间的关系

下图表明了登录、生成（编译）、下载和在在线修改之间的关系:

使用的术语如下:

源代码	当前 CoDeSys 工程(*.pro 文件, 本地 PC)
编译	上次生成过程中的编译信息将用于补充编译.(*.ci-文件, 本地 PC)
下载	最后一次加载到 PLC 中的信息(*.ri-文件,本地 PC)
PLC	在 PLC 中可获得的工程(*.prg 文件, 目标系统)



在线修改说明

在修改任务配置或 PLC 配置、插入一个库文件、执行'工程'＇全部清除'后，在线修改功能无效.(参看下面说明).

如果删除了上次下载（或在线修改）时创建的工程下载信息文件(文件〈工程名〉<目标标识号>.ri)，例如：执行命令＇工程'＇全部清除'，那么在线修改功能不再起作用，除非在另外一处保存了 ri-文件或重命名了它，这样你仍然可以得到上次下载的信息或通过'工程'＇加载下载信息'获得. 相关内容可参看下面＇工程的在线修改....'.

在线修改不重新初始化变量，因而在线修改不考虑初始值的修改!

执行在线修改后保持变量保持原值，工程重新下载也一样(参看下面，＇联机'＇下载').

当一个工程在多个 PLC 上运行时在线修改程序:

如果在两个同样的 PLC1 和 PLC2（相同的目标系统）上运行一个工程 proj.pro， 为了确保使用在线修改功能使得工程的更新在两个控制器上都能实现，方法如下：

(1) 在 PLC1 中加载并启动工程，保存 PLC1 的下载信息

1. 建立 CoDeSys 工程 proj.pro to 与控制器 PLC1 的连接（联机/通讯参数)，将工程 proj.pro 加载到 PLC1（联机/登录,下载). 在工程目录中将创建包含下载信息的文件 proj00000001.ri.

2. 重命名文件 proj00000001.ri，如为了清楚起见改为 proj00000001_PLC1.ri.改名保存这个文件是非常必要的，因为再下载 proj.pro 后新的下载信息将覆盖原来的信息，这样 PLC1 的下载信息将丢失。

3. 启动 PLC1 上的工程然后退出（'联机''运行', '联机''退出'）.

(2) 在 PLC2 中加载并启动工程,保存 PLC2 的下载信息:

1. 与 PLC2 建立连接(使用与 PLC1 相同的目标系统，并将 proj.pro 下载到 PLC2. 因此在工程目录中将再次创建包含下载信息的文件 proj00000001.ri.

2.为了清楚起见将新文件 proj00000001.ri 改名为 proj00000001_PLC2.ri.

3. 启动 PLC2 上的工程然后退出（'联机''运行', '联机''退出'）.

(3) 在 CoDeSys 中线修改工程:

在 CoDeSys 中修改 proj.pro，然后使用在线修改功能传送到两个 PLC 中.

(4) 对 PLC1 中的程序进行在线修改, 为 PLC1 重新保存下载信息:

1. 在登录时 CoDeSys 查找文件 proj00000001.ri,因此为了对 PLC1 做在线修改,你必须首先恢复在 PLC1 中的下载信息文件，它现在保存在文件 proj00000001_PLC1.ri.

有两种方式:

(a) 将 proj00000001_PLC1.ri 改回 proj00000001.ri. 这样在登录到 PLC1 时，相应的下载信息可自动获得，CoDeSys 将询问你是否做在线修改.

(b)在登录前使用命令'工程''加载下载信息'加载文件 proj00000001_PLC1.ri。在这种情况下你不需要给 ri-文件改名.

(5) 对 PLC2 中的程序进行在线修改, 为 PLC2 重新保存下载信息:

方法同上。只是下载信息文件在 proj00000001_PLC2.ri 中 .

如果系统报告

错误:

“选中的控制器配置与目标系统中的不匹配……”

确保在目标系统设置（资源）中输入的目标系统与在命令'联机''通讯参数'中输入的变量匹配。

错误:

“通讯错误，执行退出”

检查控制器是否工作，检查在“联机”“通讯参数”中输入的变量是否与控制器中的变量匹配。特别是，应该检查是否输入了正确的端口；控制器和编程系统的波特率是否匹配；如果使用了网关服务器，检查通道设置是否正确。

错误:

“程序已经被修改！要加载新程序吗？”

在编辑器中打开的工程和在PLC（或程序正在运行的 仿真模式）中的当前工程不兼容，因而不能进行监控和 调试，你既可以选择‘否’来退出，然后打开所需的工程，也可以选择‘是’，在PLC中加载当前的工程。

信息:

“程序已经改变，加载改变的部分吗？（联机改变）”。

工程正在控制器上运行，目标系统支持联机修改并且工程在控制器上随着最新下载或最新联机改变而改变，你可以决定这些变化是否要加载或命令是否要取消，也可以选择全部加载按钮加载这个编译过的代码。

'联机''退出'

符号 ：  快捷方式 ：<Strg>+<F8>

和PLC的连接中断或 仿真模式程序终止，转换到离线模式。

使用命令'联机''登录'可以转换到联机模式。

'联机''下载'

这个命令把编译过的工程加载到 PLC 中。

如果你使用 C 代码产生，在下载之前调用创建下载文件的 C 编译器，如果不是 C 代码产生的，在编译过

程中创建下载文件。

下载的信息存储在文件<工程名>0000000ar.ri之中，在 联机改变过程中用来比较当前程序和最近加载到控制器中的程序，因而只有改变的程序部分才重新加载，用命令'工程''清除'可以删除这个文件。

关于在多个 PLC 上做联机修改，参看:'联机' 登录'. 注意：在联机修改时*.ri-file 也将被更新.

在离线模式下，依据创建引导工程时的目标系统设置，可以重新生成*.ri 文件。

在下载后，只有永久变量保持它们的值（不重新初始化）。

参看此处 图表，它表明了工程－编译生成、工程－下载、在线修改和登录到目标系统之间的关系。

'联机''运行'

符号 ：  快捷方式 ：<F5>

这个命令启动在PLC中或 仿真模式中的程序。

在命令'联机''下载'或在PLC中的用户程序被命令'联机''停止'终止后，或用户程序停在一个断点处，或当执行了命令'联机''单循环' 后，这个命令开始执行。

'联机''停止'

符号 ：  快捷方式 ：<Shift>+<F8>

在两个循环周期内,终止PLC中或 仿真模式下的程序执行。

使用命令'联机''运行'来继续程序运行。

'联机''复位'

这个命令把除了保持变量(VAR RETAIN)之外的所有变量(包括 VAR PERSISTENT 变量)的当前值复位成初始化值，如果变量设置了初始化值，这个命令使变量恢复到初始化值，所有的其它变量赋予系统默认值(如整型变量默认值是 0)。作为一个预防措施，在覆盖所有的变量之前 CoDeSys 会询问你来确认你的操作，在电源故障或关闭了控制器情况下，那么程序运行时将复位系统。

使用命令'联机''运行'重新启动程序。

请参看'联机''复位 (初始状态)'和“'联机''复位(冷)'。

'联机''复位 (冷)'

这个命令使除了永久变量之外的所有变量也包括保持变量 复位到它们的初始化值。这种情况发生在加载到PLC中之前已经下载的程序启动时（冷启动），只有永久变量在复位之前保留它们的值。

相关的信息查看'联机''复位' 和'联机''复位 初始状态' 和变量重新初始化概述 - 保持变量。

'联机''复位(初始状态)'

这个命令把所有的变量包括（保持变量和永久变量）复位到它们的初始化值并在控制器上删除用户程序，控制器恢复到它的最初状态。

相关请查看'联机''复位' 和'联机''冷复位'和变量重新初始化概述 - 保持变量。

'联机''设置断点'

符号 ：  快捷方式 ：<F9>

这个命令用来在活动窗口的当前位置设置一个 断点，如果在当前位置已经设置了断点，将取消这个断点。

设置断点的位置依赖于在活动窗口中用什么语言编写 POU。

在 文本编辑器(IL, ST)，如果要在某行设置断点（可以设置断点的行的号码区是深灰色），断点设置在光标定位的地方，你也可以文本编辑器中的行号码区单击设置或删除一个断点。

在 FBD和 LD中，断点设置在当前选中的网络上。在FBD或LD编辑器中也可以在网络号码区单击设置和删除一个断点。

在 SFC中，断点设置在当前选中的 步上，在SFC中你也能使用<Shift>加双击来设置或删除一个断点。

如果一个断点已经设置，行号区、网络号码区或步的背景色将会改变成高亮蓝色。

程序运行到断点，会自动暂停，相应的区域显示为红色背景色，为了继续运行程序，使用命令'联机''运行'、'联机''单步进入'、或'联机''单步跳过'。

也可以用断点对话框来设置和删除断点。

'联机''断点对话框'

执行这个命令将打开整个工程的 断点编辑对话框，对话框也显示当前设置的所有断点。

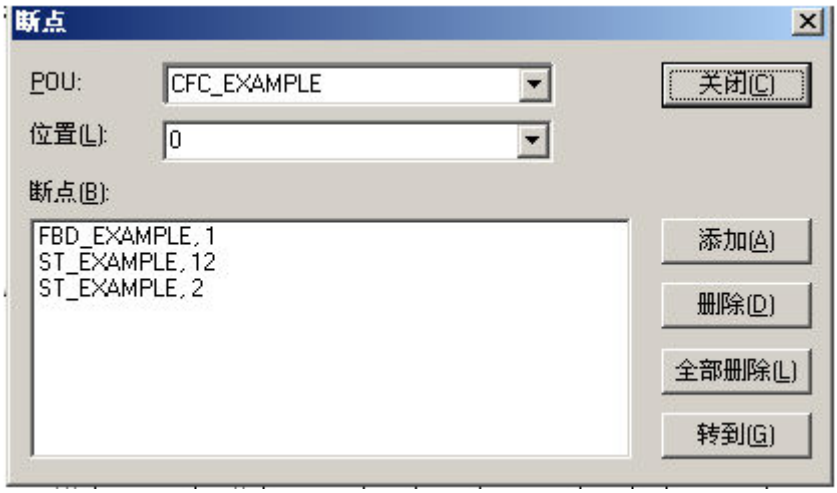
为了设置一个断点，在 POU 组合框中选择一个 POU，在断点位置组合框中选择相应的行或网络，然后按添加按钮，断点将添加到列表中。

为了删除一个断点，在断点列表中点击选择要删除的断点并按删除按钮。

全部删除按钮用来删除所有的断点。

为了在编辑器中定位到断点设置的位置，在断点列表中点击选择断点并按转到按钮。

断点编辑对话框



可以使用命令'联机''设置断点'来设置或删除断点。

'联机''单步跳过'

符号 ：  快捷方式 ：〈F10〉

使用这个命令，可以 单步执行程序，如果程序在调用一个POU，程序在调用的POU执行后暂停，在SFC中将执行一个完整的 动作。

如果当前的指令是调用一个功能或功能模块，那么功能或功能块将全部执行完。如果要运行到调用的功能或功能块的第一条指令，使用命令'联机''单步进入'。如果到达了最后一条指令，程序将会返回到POU中的下一条指令继续运行。

'联机''单步进入'

使用这个命令时进行 单步执行，程序暂停在调用的POU的第一条指令。

如果必要，可以转变到一个打开的 POU 中。

如果当前的位置是调用一个功能或功能块，那么光标停在调用的 POU 的第一条指令上。

在其它情况下，此命令和'联机''单步跳过'功能一样。

'联机''单循环'

快捷方式 ： 〈Ctrl〉+〈F5〉

这个命令执行 一个PLC循环并在这个循环后停止。此命令可以连续使用。

当执行命令'联机''运行'后，单循环结束。

'联机''写入新值'

快捷方式 ：<Ctrl>+<F7>

用这个命令可以在循环开始时写入用户为变量设置的新值。(参看命令 ’联机’ ’强制新值’为变量设置永久值)。

只要单元素变量在监控窗口中可见，就可以改变的它们的值。

在命令“写入新值”执行之前。要为变量设置新值。

对于非布尔型变量，双击变量声明的行或选中变量所在行后按<回车>键，弹出对话框“写变量 <x>”，可在此对话框中输入变量的新值。

写变量对话框

对布尔型变量，通过双击变量声明的行来设置新值（它只在 TRUE 和 FALSE 之间切换，而没有其它值)；不出现对话框。

要写入的值以青绿色显示在变量旧值后面的一个括号中。例，a=0 <:=34>。

提示：在 FBD 和 LD 编辑器中值在变量名后面以青绿色显示但不带括号。

要写入新值的变量的数目不受限制。

可以用同样的方式纠正和删除要输入的变量新值。

要写入的值在写入、删除或通过命令“强制新值”转移到强制列表之前,先存储在写入列表 (监控列表)中。

有两种方法进入写入新值对话框：

•在菜单“联机“ ”写入新值“

•点击对话框“编辑写入列表和强制列表“中的按钮”写入新值“。

当执行“写入新值“命令时，在写入列表中的变量新值将在循环开始时写到控制器中，然后从写入列表中删除。(如果执行了命令”强制新值“，要写入新值的变量也将从写入列表中删除，并转移到强制列表中!)

注意:在顺序功能图语言中,当变换是一个组合表达式时,不能对组合中的单个值用“写入新值“来改变,这是因为监控的是表达式的值,而不是单个变量的值(例如,“a AND b”,如果两个变量都是’真’时显示TRUE)。

另外，在 FBD 中，例如如果一个表达式用做一个功能模块的输入，只有表达式的第一个变量才被监控，因而“写入新值”命令只能对这个变量写入新值。

’联机’ ’强制新值’

快捷方式 ：<F7>

使用这个命令，可以对一个或多个变量进行永久性赋值。(参看命令 ’联机’ ’写入新值’，在循环开始为变量设置一次新值)。

赋值在运行系统中进行，在循环的开始处和循环的结束处新值生效。

一个循环的时序为：1.读入输入量 2. 强制赋值 3. 处理代码, 4. 强制赋值 5.写输出。

此项功能一直保持有效，直到被用户取消（用户使用命令“联机”“解除强制”）或退出编程系统。

为了设置新值，首先要创建一个写入表(参看“联机”“写入新值”的描述)。写入表中包含的变量在监控中被标记出来。当执行命令“联机”“强制新值”后，写入表转换为强制写入表。如果当前已经激活了一个强制写入表，那么系统根据要求更新。写入表被清空，同时新值用红色表示（红色表示是强制值)。下次执行“强制新值”时将强制写入表中的修改部分写入到程序中。

注意：强制写入表的修改通过使用命令下一个“强制新值”来传递程序中。

注意：在写入表中包含的变量写入新值之前，当写入表存在的同时创建强制写入表为写入表中的变量强

制新值。

为变量强制新值的方法有两种：

1、在菜单“联机“中的命令”强制新值“

2、使用对话框 “编辑写入值列表和强制值列表“中的按钮”强制新值“

注意:在顺序功能图语言中,当变换是一个组合表达式时,不能对组合中的单个值用“强制新值“来改变,这是因为监控的是表达式的值,而不是单个变量的值(例如,“a AND b”,如果两个变量都是’真’时显示TRUE)。

另外，在 FBD 中，例如如果一个表达式用做一个功能模块的输入，只有表达式的第一个变量才被监控，因而“强制新值”命令只能对这个变量强制新值。

’联机’ ’解除强制’

快捷方式 ：<Shift>+<F7>

这个命令用于解除控制器中变量的强迫赋值。变量值采用正常的方式进行更改。

被强制赋值的变量能通过红色显示在监控器中进行识别。可以删除整个强制列表，也可以根据需要选择要解除强制的变量。

删除整个强制列表，对所有的变量进行强制释放，可以选择下面方式中的一个：

•菜单“联机”中的命令“解除强制”。

•使用对话框’编辑写入值列表和强制值列表’中的按钮“解除强制”。

•在对话框“移动写入/强制值列表”中使用命令“解除强制”来删除整个强制列表，如果你选择命令“解除强制”的同时 写入列表存在将打开这个对话。

•对单个变量解除强制,首先用下面的方法来标记这些变量,标记的变量后面将出现提示“<解除强制>”。

•在一个非布尔型变量所在行上双击鼠标，打开对话 ’写入值 <x>’。选中<解除此变量的强制>，然后按“确定”。

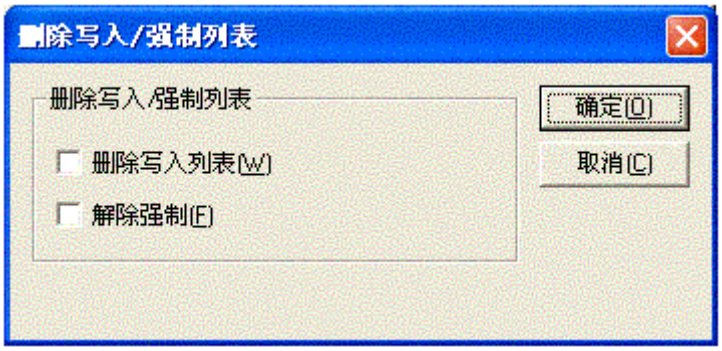
•在布尔变量声明的行上双击后在行的末端显示<解除强制>。

•使用菜单“联机”″写入/强制对话框″命令打开写入/强制对话框,删除列’强制值’编辑字段中的值。

设置成″<解除强制>″的变量将在声明窗口显示，执行命令“强制新值”将把修改的强制列表传递到程序中。

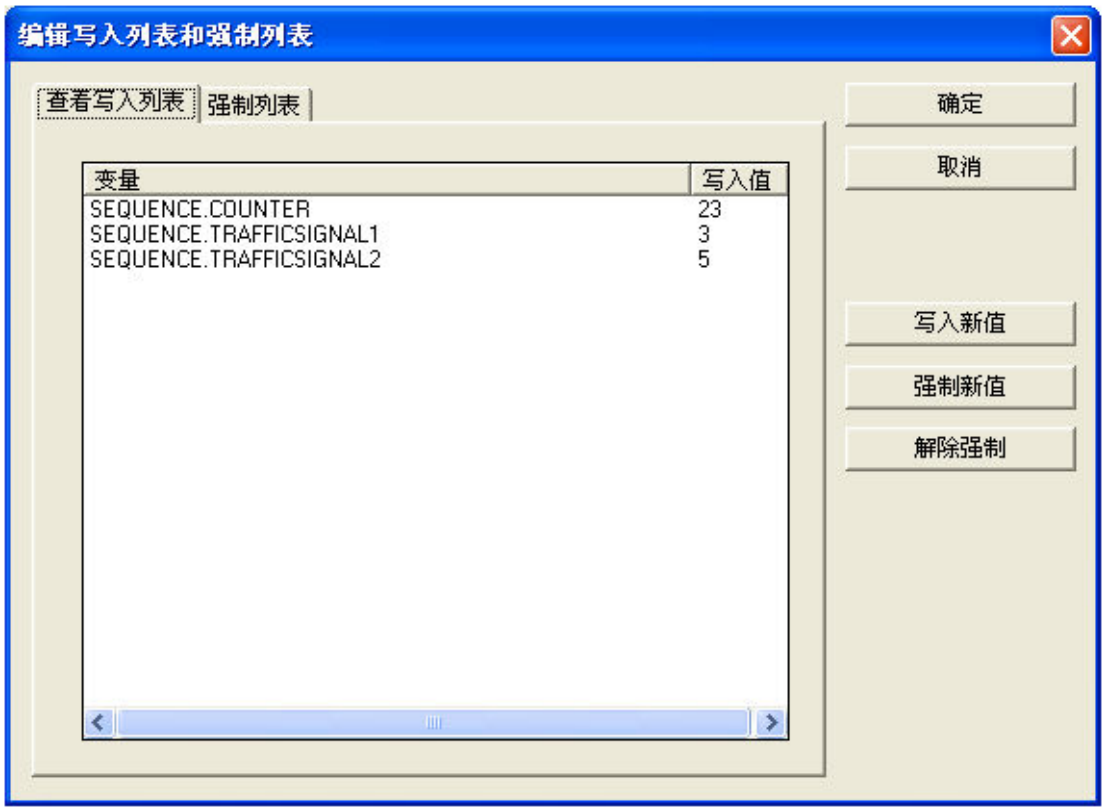
在执行命令“解除强制”时，如果当前写入列表（查看 ’联机’ ’写入新值’）是非空的，对话框“删除写入/强制值列表”将会打开，用户可以决定是释放强制还是想删除写列表或把两个列表都删除。

删除写入/强制值列表对话框



’联机’ ’写入/强制对话框’

这个命令打开一个对话框，在这个对话中显示了选项卡写入列表和强制列表，在表中列出了写入或强制的变量的名字和数值。



通过命令’联机’’写入新值’，可以把变量加进写入列表中。通过命令’联机’’强制新值’，变量被转移到强制列表中。单击某变量的列“新值”或“强制值”来编辑新值。如果输入值的类型与定义的不一致，将显示一个错误消息。如果删除一个数值，条目也将从写入表中删除。或者变量被强制中止，并不需要退出指令而关闭对话框。

下面的命令，与联机菜单中的一样，可以通过按钮来使用它们：

强制新值：当前写入列表中的全部变量将转入到强制列表，控制器中变量被强制赋值。所有标记<解除强制>的变量都不再被强制。然后关闭对话框。

写入新值：当前写入列表中的变量新值被写入到控制器中。然后关闭对话框。

解除强制：在强制列表中的所有变量将被删除，或如果写入列表存在，将打开“删除写入/强制列表”对话框，由用户根据需要做选择是解除强制、删除写入列表还是两者都选。单击确定后，两个对话框同时关闭。

’联机’’显示调用堆栈’

当 仿真模式在一个 断点停止时，可以使用这个命令，出现一个POU调用栈的列表的对话框。

调用堆栈例子

最先的POU通常是 PLC_PRG，因为这是执行的开始。

最后的 POU 通常是正在执行的 POU。

在选择了 POU 并按了‘转到’按钮，选中的 POU 加载到编辑器中来显示,显示的是正在处理的行或网络。

’联机’’显示流程控制’

依据目标系统的设置，用户可以激活或不激活流程控制功能。如果激活了它，在菜单项目的前面出现一个对勾。PLC 循环中正在执行的每一行或每一个网络将会被标记。

正在运行的行的行编号区域或网络编号区域将显示为绿色。一个附加的区域将添加到 IL 编辑器中，在这里显示当前累加器中的内容。在功能模块图和梯形图编辑器中，不传递布尔型值的连线上都将插入一个框。在框中显示变量值。当连线上传输的布尔变量值为 TRUE 时，连接线将显示成蓝色，这可以监控信息/数据的流向。

注意：

1、使用流程控制后将增加程序的运行时间。这可能会造成以时间为周期的程序出现超时错误。

2、在激活的断点处不显示流程控制。

3、如果定义了与任务有关的看门狗，当激活流程控制时将关闭此功能。

’联机’’仿真模式’

如果选择了仿真模式，在菜单的前面出现一个对勾。

在仿真模式中，用户程序运行在 PC 机的 windows 操作系统下，这个模式用来测试工程。PC 和仿真模式的通讯使用的是 Windows 的消息机制。

如果程序不处于仿真模式，程序将运行在 PLC 上，PC 和 PLC 的通讯通常采用串口。

这个标记的状态和工程一起存储。

请注意：外部库的 POU 将不能运行在仿真模式下。

’联机’’通讯参数’

当本地 PC 和实时系统之间的通讯运行在系统中的网关服务器中时（如果使用了 OPC 或 DDE 服务器，在配置中必须输入相同的通讯变量），使用配置通讯参数对话框来设置通讯参数。

参看下面的项目：

网关系统的原理

本地PC的通讯参数对话框

设置期望的网关服务器和通道

为本地网关建立新的通道

本地PC上通讯参数对话框显示的内容

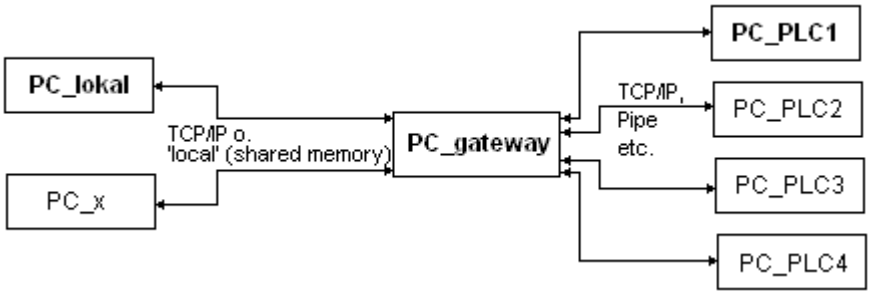
网关系统的原理

网关系统的原理:

在我们解释对话的操作之前先来解释一下网关系统的原理：

网关服务器可以允许本地 PC 和一个或多个实时系统之间进行通讯，网关服务器可以和实时系统一起运行在本地 PC 上。如果我们要处理在其它计算机上运行的网关服务器时，我们必须保证它已经运行。本地安装的网关服务器，当登录到目标实时系统时，它自动启动。你可以通过在任务栏右下的 CoDeSys 符号的出现来识别它是否启动。在符号上单击鼠标右键，可以得到 Info 和 Finish 菜单项。

下面是一个网关系统图：



PC_local 是本地计算机，PC_x 是另外的计算机，PC_gateway 是网关服务器，PC_PLC1 到 PC_PLC4 是运行实时系统的计算机。图表显示了分离的模块，实际上，网关服务器可以和实时系统一起安装在本地计算机上。

重要：和网关服务器之间的通讯只能建立在 TCP/IP 上，要保证你的计算机的配置的正确性。

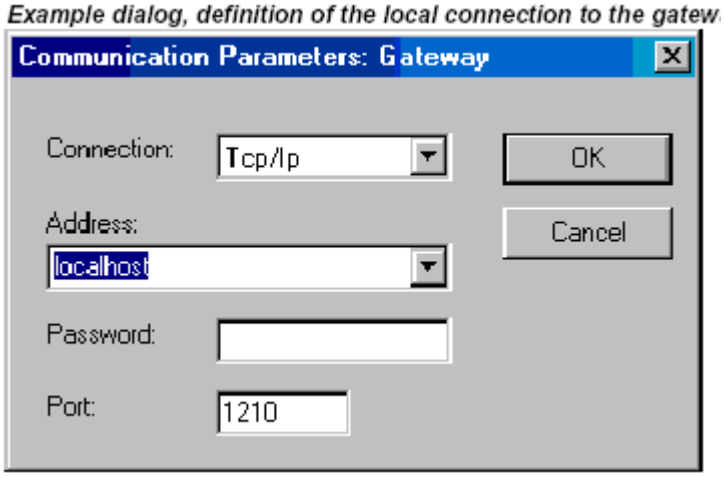
从网关到不同的实时系统计算机可以运行在不同的协议上(TCP/IP、Pipe 等等)。

建立期望的网关服务器和通道

1. 在通讯变量对话中建立期望的网关服务器和信道。

为了定义与期望的网关服务器之间的连接，按网关按钮来打开对话“网关信息参数”。

Example dialog, definition of the local connection to the gatew



在这里你可以输入或编辑下面的部分：

•从你计算机到正在运行的网关服务器计算机之间的连接类型，如果网关服务器在本地计算机上运行，连接通过共享内存或通过 TCP/IP 都是可以的；如果需要连接到不同的计算机上，只能使用 TCP/IP 协议。

•正在运行的网关服务器的地址：IP 地址或正确的符号名比如 localhost。在初始化，标准的’localhost’作为计算机名字（地址），可以访问本地安装的网关。名字’localhost’与 IP 地址 172.0.0.1 在大部分情况下相同，在某些情况下必须在地址区域中直接输入名字。如果你想访问其它计算机上的网关服务器，你必须用它的名字或 IP 地址来替换’localhost’。

•为选中的网关服务器设置密码，如果它在一个远程计算机上。如果它不正确的输入，或没有输入完整，

出现一个错误消息。注意这个连接：你可以通过以下步为本地安装的网关服务器设置密码：在工具栏右下部分的网关符号上单击鼠标右键并选择“改变密码”。出现一个改变或输入密码的对话。如果你访问本地网关服务器不要求输入密码。

•网关服务器正运行的计算机的端口，作为一个规则已经给出了选中的网关服务器端口。

按 OK 键关闭对话，相应的条目（计算机地址）出现在在“通信参数”对话的顶部信道区域中，在它下面是网关服务器可用的信道。

2.在选中的网关服务器上建立期望的信道

用鼠标在条目上单击选择其中的一个信道，相应的变量随后显示在表中。如果没有能和选中的网关地址建立连接，可能是因为还没有启动它或地址不正确，在地址之后的括号中显示‘没有连接’并出现一个消息‘不能找到网关的设置’。

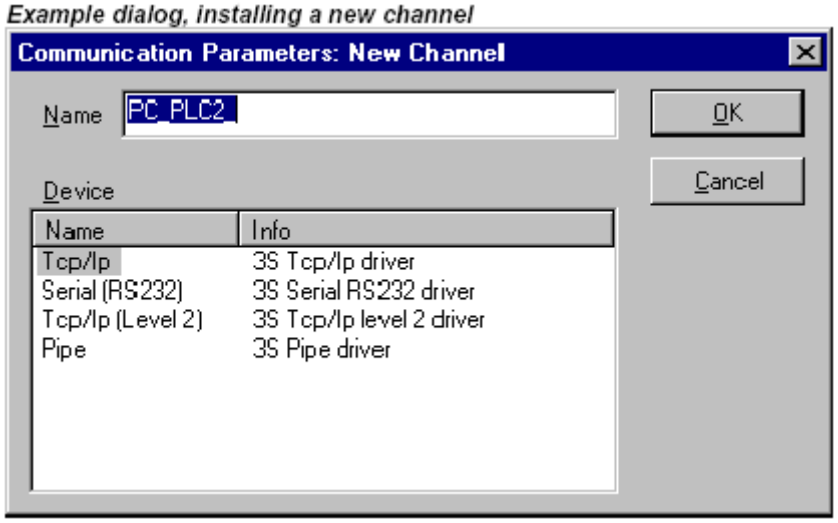
一旦建立了期望的信道，按 OK 关闭对话。设置和工程一起保存起来。

为本地网关联机新的通道

可以为当前连接的网关服务器设置一个新的信道，这样可以从服务器上建立更多的连接，例如，和一个控制器的连接。可用的选择依赖于计算机上安装的设备驱动程序的数目的选择。

在通讯变量对话中，按新建按钮，打开对话 通信参数: 新通道。

Example dialog, installing a new channel



•Name 输入区域自动包含最近输入的信道的名字，如果没有定义信道，将会提供当前的网关名字，后跟一个下划线字符，例如，“localhost_”，可以在这里编辑信道的名字，信道的名字不必有一个唯一的名字，但是推荐使用唯一的名字。

•在 Device 下面的表中列出了网关计算机上用到的设备驱动程序，在名字列中，用鼠标单击来选择一个可用的驱动程序，相应的注释，如果有的话，将出现在信息列中。

如果按 OK 键关闭了“...新信道”对话，新定义的信道出现在“通信参数”对话中，在信道中作为在减号下面的最下位置一个新条目，它只保存在本地工程中。你可以编辑列（查看下面的提示），按 OK 来确认输入的参数，然后退出“通信参数”对话。

为了使网关服务器 xy 能识别新输入的网关信道和它的变量，同时也为了使其它的计算机能访问这个网关，必须先登录到实时系统中，再打开“联机” “通信参数”对话时，新信道出现在信道树形结构中，不但在它的先前位置，而且网关服务器 xy 的名字或地址下面，这表明了它已经存在于网络中。

除了本地计算机外，也可以在其它计算机上打开通讯变量对话，选择网关 xy 并使用它的新信道。

如果登录时出现通讯错误，可能是接口不能打开（例如，串行连接的端口 COM1），可能是因为端口被其它设备使用了，也可能是控制器没有运行。

网关服务器已经识别的信道的变量在配置对话中将不能编辑，变量区域显示为灰色，当它未激活时，可以删除这个连接。

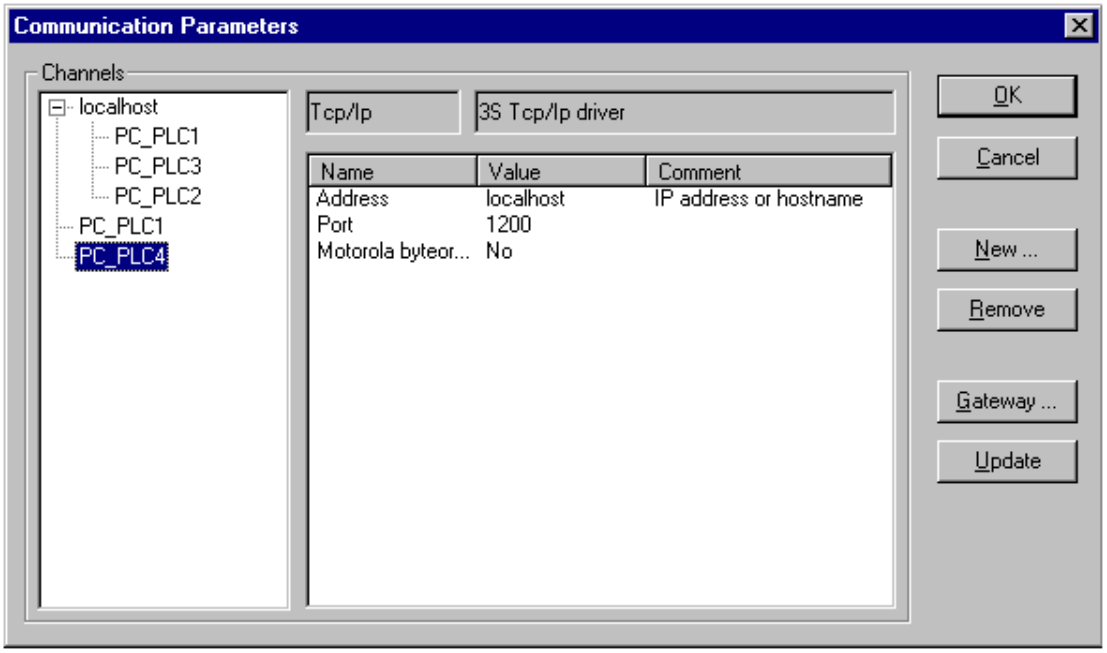
重要：删除一个信道是不可逆的，当按按钮移动时，它就被删除了。

如何在本地PC表示通信参数菜单

这个对话用来 选择一个与PLC通讯的网关服务器，而且可以为安装在本地计算机上的网关服务器 设置新的信道，这些信道能够被网络上的其它计算机访问。

使用按钮更新可以在任何时候调用当前的设置。

如果根据例子’网关系统原理’已经配置了通讯变量，将出现下面的对话：



标题 Channels 下列出了两个类别的连接：

一方面显示出的安装在当前连接的网关服务器上的所有连接称为“本地主机”。这个网关的地址或名字位于在减号后面的最上位置，在我们的例子中运行在本地计算机上。在通常情况下“本地主机”对应于本地计算机的 IP 地址 127.0.0.1。它的下面，是网关信道设置到的(PC_PLC1 to 3)实时计算机的三个地址，它们可能在本地计算机或其它的没有连接到网关服务器的计算机上设置过。

第二个类别的信道包括所有连接到从本地计算机上设置的网关的连接，它们在减号下面直接创建了分支PC_PLC1 和PC_PLC4。这些地址不需要在网关上识别。对于例子中的PC_PLC4，配置变量存储在本地 工程中但下次登录到实时系统时，首先被网关识别。对于PC_PLC1 这个已经出现，相关的网关地址已经作为一个子分支出现在了信道树形结构中。

在对话的中间部分可以看到左边选中的信道和 Name、Value 和 Comment 下相关变量的说明。

在通信参数菜单中编辑参数的Tips

只能编辑值列下的文本区域。

用鼠标选择一个文本区域，通过双击或按空格键进入编辑模式，文本输入完成时按<Enter>确认。

你可以使用<Tabulator>或<Shift> + <Tabulator>来跳转到下一个或先前切换或编辑中。

为编辑数字值，可以用方向键或 Page Up/Down 键来分别地改变值一个或十个单位。鼠标双击也能增加一个单位，为数字值安装了输入检查：<Ctrl> + <Home> 或 <Ctrl> + <End>可以为问题中的变量类型输入最低或最高值可能值。

快速检查连接网关的不成功连接

如果连接到选中的网关计算机不成功，应该有一个下面的检查（在信道区域中网关服务器地址之后 通讯变量对话中出现消息“没有连接” ）：

•网关服务器已经启动了吗？（工具栏的低部右边部分出现三色符号）

•在通讯变量对话中输入的 IP 地址是不是网关运行的计算机的？（使用“ping”来检查）

•TCP/IP 连接工作吗？错误可能在于 TCP/IP。

’联机’’源代码下载’

这个命令将工程的源代码加载到控制器系统中，不要与工程编译时产生的代码相混淆！在’工程’’选项’’代码源下载 ’对话框中设置下载选项（时间，大小）。

’联机’’创建引导工程’

使用这个命令，在控制器上为已编译的工程创建引导工程，当重新启动时可以自动加载它。引导工程的存储依赖于目标系统而不同。例如，在 386 系统上创建三个文件：default.prg 包含工程代码，default.chk 包含代码检验，default.sts 包含了重新启动之后的控制器状态（启动/停止）。

在离线模式下，也可以对编译没有错误的工程使用命令“联机”“创建引导工程”，在这种情况下，在工程目录中创建下面的文件：保存引导工程代码的<工程名>.prg 和包含检验的<工程名>.chk，这些文件可以重命名，然后复制到 PLC 中。

依据目标系统的设置，在离线模式下创建引导工程时，可能会创建一个新的 *.ri-文件 ，如果这个文件已经存在，依据目标系统设置会出现一个对话框。

注意：如果激活了工程-选项-源代码下载- 使用选择文件创建引导工程 ，在使用命令“联机”“创建引导工程”时，选中的资源自动加载到控制器中。

’联机’’文件写入PLC’

这个命令可以把任何期望的文件加载到控制器中，在打开的对话框“在控制器中写入文件”中可以选择期望的文件。

在使用“打开”按钮关闭对话框后，文件加载到了控制器中并用同一名字保存，在加载过程中有一个进度提示。

使用命令’联机’’从PLC中读取文件’可以取回先前加载到控制器中的文件。

’联机’’从PLC中读取文件’

用这个命令将打开对话框“从控制器中读取文件”，用于读取回先前使用命令’联机’’文件写入PLC’加载到控制器中的文件。在文件名下面，提供了期望文件的名字，在选择窗口中输入它加载的目录，按“保存”按钮关闭对话。

4.7 设置窗口

窗口设置

在“窗口”菜单下,能找到管理窗口的所有命令，有自动设置窗口命令、有打开 库管理器的命令、有切换打开窗口的命令，在菜单的最后你会找到一个按它们打开时的顺序打开的所有窗口的一个列表，在相关的条目上单击鼠标能切换到期望的窗口，在活动窗口的附近出现一个√。

参照:

’窗口’’水平平铺’

’窗口’’垂直平铺’

’窗口’’层叠窗口’

’窗口’’最小化排列’

’窗口’’关闭所有’

’窗口’’信息’

’窗口’’水平平铺’

用这个命令你能在工作区水平的排列所有的窗口使它们不重叠并且占据整个工作区间。

’窗口’’垂直平铺’

用这个命令你能在工作区垂直的排列所有的窗口使它们不重叠并且占据整个工作区间。

'窗口''层叠窗口'

使用这个命令你能在工作区间中以层叠的方式排列所有的窗口，一个窗口跟着一个窗口。

'窗口''最小化排列'

使用这个命令你能在工作区间排列所有的最小化窗口使它们在工作区间的底端排成一列。

'窗口''关闭所有窗口'

使用这个命令能关闭工作区间的所有窗口。

'窗口''信息'

快捷方式 ： 〈Shift〉+〈Esc〉

用这个命令你能打开和关闭带有来自最近编译，校核，或比较过程的消息的 信息窗口

如果消息窗口是打开的，在命令的附近会出现一个 √。

4.8 帮助

'帮助''内容' 和 '搜索'

使用‘帮助’菜单下的‘内容’或‘搜索’，打开帮助主题窗口。此窗口中的文件也可以借助 HTML 帮助查看器（Internet Explorer V4.1 或更高版本）。

内容选项卡显示帮助手册的目录。通过双击或点击‘+’/‘-’可以打开或关闭书。在窗口右侧显示所选目录的帮助信息。具有不同颜色和下划线的文本表明此处有超级链接，即查看更多相关内容。在这些文字上单击将打开链接或图片。例如，点击在这页下面的‘帮助主题窗口’，将显示帮助主题窗。点击‘上下文帮助’将打开相应的帮助信息页。

在索引选项卡上，可以寻找特定词的帮助信息。在搜索选项卡上，在所有帮助信息中查找搜索词。

参看: 上下文关联帮助.

帮助主题窗口

上下文关联帮助

快捷方式 ： 〈F1〉

为了打开 在线帮助，在活动窗口中、在一个对话框中，或菜单命令上按〈F1〉键。

你也可以选中文本（例如：关键字或一个标准功能），然后按‘F1’,则显示相关的帮助。

5. CoDeSys中的编辑器

5.1 关于所有的编辑器

编辑器的组件

POU 的所有编辑器由声明部分和主体部分组成，主体可以由其它的文本或图形编辑器组成；声明部分通常是文本编辑器。主体和声明部分通过能拖动的屏幕分割器来分开，通过用鼠标点击它并朝上或朝下移动它到希望的位置。

打印范围

如果在工程选项中的对话“工作区域”中的选项“显示打印范围”被选中，编辑器内容打印时用到的水平和垂直的页边空白用红色虚线显示。打印机的属性和打印版面的尺寸在菜单’文件’’打印机设置’中选择。如果没有打印机设置或打印版面输入，应用默认设置 (Default.DFR 和默认打印机)。

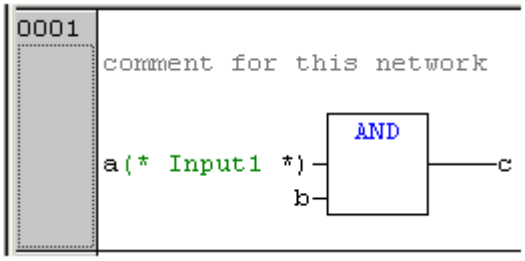
注意：当放大因数选择 100%时，打印页边空白原样显示。

注释

用户注释必须封套入特殊的符号序列中（“*”和“*”）。例如，(*这是一个注释*)。

在所有的文本编辑器中，在任何期望的位置、在所有的声明、IL和 ST语言和在自定义数据类型中都允许使用注释。如果工程使用一个模板来打印输出，在变量声明过程中输入的注释出现在每个变量后的基于文本编程组件中。

在 FBD和 LD图形化编辑器中，可以为每个网络输入注释。



搜索你想添加注释的网络并激活“插入”“信息”，如果在菜单 ’附加’ ’选项’中相应的选项激活，在梯形图编辑器中可以为每个特殊的触点和线圈添加附加的 注释，在 CFC中有特殊注释的POU可以按自己的意愿放置。

在 SFC中，你能在编辑步属性对话中输入关于步的注释。

在 ’工程’ ’选项’ ’建立选项’中的合适的选项激活，允许使用嵌套注释。

在联机模式，如果鼠标在变量上停留，在内容提示中将显示变量和地址的注释。

切换到POU

快捷方式 ：<Alt>+<Enter>

用这个命令选中的 POU 加载到它的编辑器中，如果光标放置在文本编辑器中的 POU 名字上或如果 POU 框在图形编辑器中被选中，在内容菜单(<F2>)或在“附加”菜单中可用到这个命令。

如果你从库中处理一个 POU，那么库管理器被调用，相应的 POU 显示。

’附加’ ’打开实例’

这个命令与“’工程’ ’打开实例’ 相同。

如果光标放置在文本编辑器中的功能模块的名字上或如果功能模块框在图形编辑器中被选中，在内容菜

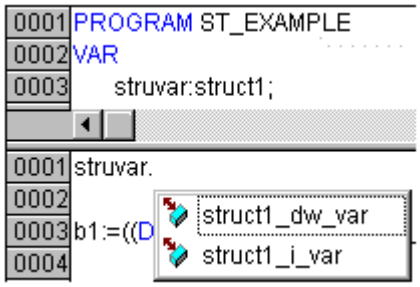
单(<F2>)或在'附加'菜单中可用到这个命令。

智能功能

如果选项列 表组件在工程选项对话中为类别“编辑器”激活，智能功能将在所有的编辑器中、在监控和配方管理器中、在可视化和采样跟踪中可用：

•如果插入一个圆点“.”来代替一个标识符，出现一个选择框，列出了工程中的所有本地和全局变量，在这里可以选择这些元素中的一个并按“返回”来插入到圆点之后，你也能通过双击在列表条目上来插入元素。

•如果在圆点后输入了一个 功能模块实例或一个 结构化变量，选择框列出了相应功能模块的所有输入和输出变量 或列出结构化组件，在这里可以能选择期望的元素并按“Return”或双击来输入它。



脱机状态下的标识符

在脱机状态且所有的编辑器被如下申请：如果指针被放置在可编辑的标识符上，变量被按数据类型划分（如 VAR_GLOBAL），变量特征（如 RETAIN），地址和注释将被显示

5.2 声明编辑器

声明编辑器用来声明 POU 变量和全局变量、声明数据类型，它能使用通常的窗口功能，如果安装相应的驱动程序还能使用智能鼠标的功能。

在改写模式下，“OV”在状态栏上显示为黑色，通过<Ins>键可以在插入和改写模式之间切换。

句式颜色支持变量的声明。

在内容菜单（鼠标右键或<Ctrl>+<F10>）中有最重要的命令。

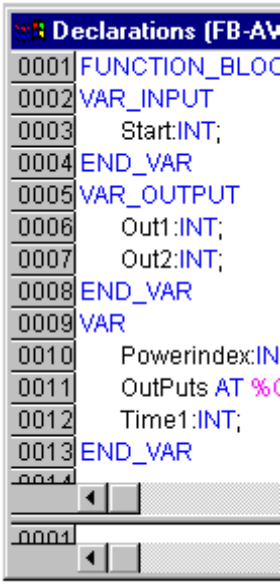
声明部分

数据类型	下限	上限	位数	前缀	注释
BOOL	FALSE	TRUE	1 Bit	x	
				b	保留
BYTE			8 Bit	by	位串，不用于算术运算
WORD			16 Bit	w	位串，不用于算术运算
DWORD			32 Bit	dw	位串，不用于算术运算
LWORD			64 Bit	lw	位串，不用于算术运算
SINT	-128	127	8 Bit	si	
USINT	0	255	8 Bit	usi	
INT	-32.768	32.767	16 Bit	i	
UINT	0	65.535	16 Bit	ui	
DINT	-2.147.483.648	2.147.483.647	32 Bit	di	
UDINT	0	4.294.967.295	32 Bit	udi	
LINT	-263	263 - 1	64 Bit	li	
ULINT	0	264 - 1	64 Bit	uli	
REAL			32 Bit	r	
LREAL			64 Bit	lr	
STRING				s	
TIME				tim	
TIME_OF_DAY				tod	
DATETIME				dt	
DATE				date	
ENUM			16 Bit	e	

只有在这个POU中的所有将要使用的变量才在POU的声明部分中声明，这些变量包括：输入变量、输出变量、输入/输出变量、局部变量、添加的变量和常量。声明格式是基于IEC61131-3标准。

关于使用模板创建全局变量、数据类型、功能、功能模块或程序类型的对象的可能性，查看4.3章节“File’ New from template”。

下面是在CoDeSys编辑器中正确声明变量的例子：



推荐的标识符命名方式

在变量声明(变量名称)，用户定义的数据类型和创建POUs（功能,功能块,程序)和可视化界面时定义标识符.为了保证标识符的唯一性,你可以使用下面的标识符命名方法..

(1) 变量名称

在应用程序和库文件通常采用匈牙利符号法:

对于每个变量,主名称的命名应该直观,简短.每个词的第一个字母应该大写,其它的小写.(例如: FileSize). 如果需要,可以创建其它语言的翻译文件.

对应于变量的数据类型,可以在主名称前增加前缀,以表明变量的类型,最好用小写字母.

为了区别于 BYTE 和易于 IEC 编程人员的理解(参看赋地址%IX0.0),强烈推荐在 BOOL 变量前使用 x 作为前缀.

例如:

```
bySubIndex: BYTE;
sFileName: STRING;
udiCounter: UDINT;
```

在嵌套声明中,按声明顺序增加前缀:

类型	下限	上限	存储空间	前缀	注释
POINTER				p	
ARRAY				a	

例如:

```
pabyTelegramData: POINTER TO ARRAY [0..7] OF BYTE;
```

功能块实例和用户定义的数据类型的变量的前缀可以分别使用 FB 和数据类型名(例如: sdo).

例如:

```
cansdoReceivedTelegram: CAN_SDOTelegram;
TYPE CAN_SDOTelegram :      (* 前缀: sdo *)
STRUCT
    wIndex:WORD;
    bySubIndex:BYTE;
    byLen:BYTE;
    aby: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE
```

对于局部常量(c) 开头以 c 为前缀, 后跟下划线(_),然后是类型前缀和变量名.

例如:

```
VAR CONSTANT
    c_uiSyncID: UINT := 16#80;
END_VAR
```

对于全局变量(g)和全局常量(gc) 附加的前缀+'_'被附加到库库前缀:

例如:

```
VAR_GLOBAL
    CAN_g_iTest: INT;
END_VAR
VAR_GLOBAL CONSTANT
    CAN_gc_dwExample: DWORD;
END_VAR
```

(2) 用户定义的数据类型 (DUT)

每个结构数据类型的名称组成方式是: 库名前缀(如: CAN), 下划线'_'和结构的恰当的描述(如:

SDOTelegram). 对于这个结构所使用的变量,相关的前缀应该直接放在冒号后.

例如:

```
TYPE CAN_SDOTelegram :      (* 前缀: sdo *)
STRUCT
    wIndex:WORD;
    bySubIndex:BYTE;
    byLen:BYTE;
    abyData: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE
```

枚举 以库名前缀开始(如: CAL), 后跟下划线'_'和大写字母的标识符.

注意:在 CoDeSys 老版本中,枚举值大于 16#7FFF 会发生错误,因为它们不能自动转换成 INT 型数值. 由于这个原因枚举(ENUM)应该使用正确的 INT 型数值定义.

例如:

```
TYPE CAL_Day :(
CAL_MONDAY,
CAL_TUESDAY,
CAL_WEDNESDAY,
CAL_THIRSDAY,
CAL_FRIDAY,
CAL_SATURDAY,
CAL_SUNDAY);
```

声明:

```
eToday: CAL_Day;
```

(3) 功能,功能块,程序(POU)

功能,功能块和程序的名称组成是库前缀(Example: CAN), 下划线'_'和 POU 的恰当描述(如: SendTelegram). 与变量声明一样,POU 名的第一个词的第一个字母应该大写,其它是小写.建议 POU 名由动词和名词组成.

例如:

```
FUNCTION_BLOCK CAN_SendTelegram (* 前缀: canst *)
```

在声明部分,为 POU 注释一个简短的描述. 同时为所有的输入和输出加注释. 在功能块下,对于创建的实例,相关的前缀应该直接放在名称后面.

动作可以没有前缀; 因为他只是内部调用, 即由 POU 本身调用, 以 prv_开头.

由于要与 CoDeSys 老版本兼容,每一个功能至少要有一个参数. 外部功能不能使用结构作为返回值.

(4) 可视化界面的名称

注意: 你必须避免可视化界面的名称不能与工程中的 POU 具有相同的名字. 否则在可视化界面之间转换时会出现错误.

输入变量

在关键字 VAR_INPUT 和 END_VAR 之间，所有定义的变量作为 POU 的输入变量，在调用位置，可以随着调用赋予变量值。

例如：

```
VAR_INPUT
in1:INT (* 1. Inputvariable*)
END_VAR
```

输出变量

在关键字 VAR_OUTPUT 和 END_VAR 之间，定义的所有变量作为 POU 的输出变量，也就是说调用后值返回到 POU。

例如：

```
VAR_OUTPUT
out1:INT; (* 1. Outputvariable*)
END_VAR
```

输入输出变量

在关键字 VAR_IN_OUT 和 END_VAR 之间，定义的所有变量作为 POU 的输入和输出变量。

使用这个变量，传递变量的值被改变。那意味着变量的输入值不能为常量。

正是这个原因，所以不能通过<functionblockinstance><in/outputvariable>来从外部直接地读或写功能模块的输入和输出变量。

例如：

```
VAR_IN_OUT
inout1:INT; (* 1. 输入输出变量 *)
END_VAR
```

局部变量

在关键字 VAR 和 END_VAR 之间定义了 POU 的所有局部变量，它们没有外部的连接，换句话说，它们不能从外部写入。

例如：

```
VAR
loc1:INT; (* 1. 局部变量*)
END_VAR
```

保持变量

Remanent 变量能在程序运行期间始终保持它们的值，这些变量包括保留变量和永久变量。

保留变量用关键字RETAIN来识别，这些变量保持它们的值即使是在控制器的非正常关闭时和正常的关闭和其中的一个控制器或在命令 ’联机’ ’复位’时。当程序重新运行时，存储的值将进行进一步的处理。一个具体的例子是生产线上的饼形计数器在电源故障后重新开始计数。

所有其它的变量从新初始化，不是用它们的初始化值或标准初始化的值。

与永久变量相反，保留变量在程序的一个新的下载时重新初始化。

永久变量通过关键字PERSISTENT来识别。不象保留变量，这些变量在一个重新下载或在执行命令“联机”’复位 (初始值)’或“Online”’复位 (冷)’之后保留它们的值，但在关闭和控制器开不保留它们的值，因为它们不保存在保留区。如果永久变量需要在控制器的非正常关闭后保留它们的值，那么它们必须另外定义为保留变量。永久保留变量的一个具体例子是一个操作定时器在电源故障之后重新定时。

例如：

```
VAR RETAIN
rem1:INT; (* 1. 保持变量*)
END_VAR
```

注意：

1.如果一个局部变量定义为保留变量，变量将保存在保留区（象一个全局保留变量）

2.如果在功能模块中的一个局部变量定义为保留变量，功能模块的整个实例将会保存在保留区（POU 的所有数据），因而只有定义的保留变量才处理为保留变量。

3.如果在功能中的局部变量定义为保留变量，这不起任何作用，变量将不保存在保留区内！如果一个局

部变量在功能中定义为永久变量，这也不起任何作用。

常量

常量用关键字 CONSTANT 来识别，它们能定义为局部或全局变量。

句式：

VAR CONSTANT

<Identifier>:<Type> := <initialization>;

END_VAR

例如：

VAR CONSTANT

con1:INT:=12; (* 1. 常量*)

END_VAR

外部变量

要导入到 POU 的全局变量用关键字 EXTERNAL 来指定，它们也出现在联机模式中声明部分的观察窗口中。

如果外部变量声明与全局变量声明在各方面不匹配，出现下面的消息：“变量的声明和全局变量的声明不匹配”。

如果全局变量不存在，出现下面的错误消息“不能识别的全局变量”

例如：

VAR EXTERNAL

var_ext1:INT:=12; (* 外部变量 *)

END_VAR

关键字

关键字在所有编辑器中书写为大写字母，关键字不能用作变量。例如关键字：VAR、 VAR_CONSTANT、 IF、 NOT、 INT。查看 CoDeSys中的有效关键字目录.

声明变量

变量的声明采用下面的句式：

<Identifier> {AT<Address>}:<Type> {:=<initialization>};

{}中的部分是可选择的。

关于标识符，是一个变量的名字，需要注意的是它不能包含空格或变元音字符，它不能重复定义并且不能与关键字相同。不区分大小写，VAR1、 Var1 and var1 是相同的变量。在标识符中的下划线是有意义的，例如，A_BCD 和 AB_CD 在解释的时候认为是不同的变量。不允许在标识符前面或在标识符中有多个连续的下划线。标识符的长度是不限制的。

变量的所有声明和数据类型元素能包含初始化值。它们通过“：=”来操作。对于基本类型的变量它们的初始化值是常量，所有的声明的默认的初始化值是 0。

例如：

var1:INT:=12; (* 整型变量,初始值是 12*)

如果你想让变量直接使用明确的地址，那么必须用关键字 AT 来定义变量。

为了快速输入声明，使用快捷方式模式。

在 功能块你能指定变量不完整的地址描述，为了在本地 实例中使用这样的变量，在 变量配置中必须使用它的条目。

也可以 自动声明。

AT声明

如果你想让变量直接使用明确的地址，那么必须用关键字 AT 来定义变量。

这样做的好处是你能给地址赋予一个恰当的名字，输入和输出信号的任何改变只要在一个地方修改就可

以了。

要求输入的变量不能被写操作访问。

例如：

counter_heat7 AT %QX0.0: BOOL;

lightcabinetimpulse AT %IX7.2: BOOL;

download AT %MX2.2: BOOL;

注意：如果布尔变量被赋予了字节，字，或 DWORD 地址，它们只占一个字节而不是在偏移后的第一个位。

‘插入’‘声明关键字’

你能用这个命令来打开一个在POU的声明部分用到的关键字的列表，在选中 关键字后并且确认了选择，关键字插入在当前光标位置。

当你打开 输入助手(<F2>)并选择了’声明’类别时，你也能查看关键字列表。

‘插入’‘类型’

用这个命令你能接收一个为变量的声明的可能类型，当你访问输入帮助(<F2>)后你也能接收列表。

类型分为下列类别：

标准类型 布尔，字节，等等。

结构体，枚举类型，等等。

实例声明的标准功能模块

实例声明的已定义的功能模块

CoDeSys支持IEC1131-3 的所有 标准类型：

语句颜色

在所有的编辑器中你能在变量的应用和声明中使用可视化的帮助。可以避免错误，或很快查出错误，因为文本用颜色显示。

注释解释指令，迅速显示；关键字不会偶然拼错，等等。

将会用到下面强调的颜色：

蓝色　关键字

绿色　文本编辑器中的注释

粉红色 特殊常量（例如，TRUE/FALSE, T#3s, %IX0.0）

红色　输入错误（例如，无效时间常量，关键字，小写……）

黑色　变量，常量 ……

快捷模式

CoDeSys的 声明编辑器允许你使用快捷模式。

当你使用<Ctrl><Enter>结束一行时这个模式激活。

支持下面的捷径：

所有的标识符一直到最近的标识符将成为变量声明标识符

声明的类型由行的最近的标识符决定。在本书中，使用到以下快捷：

B or BOOL	声明为	BOOL
I or INT	声明为	INT
R or REAL	声明为	REAL
S or string	声明为	STRING

如果通过这些规则还没有建立类型，类型是 BOOL 并且最近的标识符将不会用作一个类型。（例 1.）

每个常量，依赖声明时的类型，将会变成一个初值或字符串。（例 2 和 3）

在分号后面的文本成为一个注释（例 4.）

所有在行中的其它字符被忽略（例如，例 5.的感叹号）

例如：

快捷方式	声明
A	A: BOOL;
A B I 2	A, B: INT := 2;
ST S 2; A string	ST:STRING(2); (* 字符串 *)
X %MD12 R 5 Real Number	X AT %MD12: REAL := 5.0;(* 实数 *)
B !	B: BOOL;

自动声明

如果在选项对话框的编辑器类别中选中 自动声明选项，在没有定义的变量输入后在所有编辑器中出现一个对话框，在对话框帮助下，现在可以声明变量。

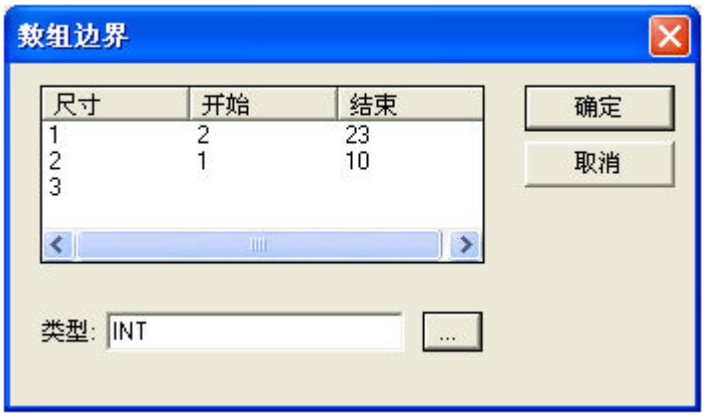
变量声明对话框

在 CLASS 组合框包含有局部变量(VAR)、输入变量(VAR_INPUT)、输出变量(VAR_OUTPUT)、输入/输出变量(VAR_INOUT)或全局变量(VAR_GLOBAL)，选择你要处理的类型。

通过常量、保持变量、永久变量选项，你能决定要处理常量还是保持变量。在编辑器中输入的变量名添加到了’名称’区域，BOOL 已位于’类型’区域。  用来打开输入助手对话框,你可以选择所需的数据类型。

数组的声明:

如果选择了数组作为变量的类型，输入数组边界的对话框出现：



通过用鼠标在相应的区域单击打开编辑区域可以在’起始’和’结束’下输入数组边界，数组的数据类型输入在’类型’区域。为了做这些，用按钮  来调用输入助手对话框。

按 OK 按钮来退出数组边界对话框，在 IEC 格式中的变量声明建立在对话中的 Type 区域中的条目。例如，

整型数组 ARRAY [1..5, 1..3]，在区域'初始值'中，可以输入声明变量的初始化值。如果这是一个数组或结构体，通过按钮  或打开其它类型变量的输入帮助对话，你将可以打开一个特殊的初始化对话框。

在数组初始化对话中，出现一列数组元素；在“：=”前的空白处单击打开一个编辑区域来输入元素的初始化值。

为结构体和单个元素初始化的对话显示在一个树状结构中，类型和默认的初始化值出现在变量后面的括号中；每个后面跟着“：=”。在“：=”后面的区域单击打开一个编辑区域，在这里可以输入期望的初始化值。如果元素是数组，数组中单个区域的显示可以通过鼠标在加号上单击来扩展开并且可以用初始化值来编辑这些区域。

按'确定'键退出初始化对话框后，数组或结构体的初始化值出现在在 IEC 格式中的声明对话框中的'初始值'区域中。

例如：x:=5,field:=2,3,struct2:=(a:=2,b:=3)

在'地址'区域中，可以绑定声明到 IEC 地址的变量。

如果需要，可以输入一个注释，注释可以通过使用组合键<Ctrl> + <Enter>来形成换行格式。

按'确定'键，声明对话关闭并且根据 IEC 的句法变量输入到相应声明编辑器中。

注意：变量声明的对话框，也可以通过命令 '编辑' '声明变量'来得到。在联机模式下，如果光标停留在一个变量上，用<Shift><F2>可以打开自动声明的窗口，在这个窗口中，当前变量的相关设置显示在这里。

声明编辑器中行号

在脱机模式下，在一个特定的行号码上单击将标记整个文本行。

在联机模式下，若此行显示的是结构变量，那么在行号上单击将会展开或隐藏结构变量中的变量。

按表格形式声明

在选项对话框如果声明表格选项激活，声明编辑器象一个表格，就象一个卡片索引框，你能选择各自变量类型的选择卡并编辑变量。

每个变量有下面的条目区域。

名字： 变量的输入标识符。

地址： 如果必须，变量的输入地址（AT声明）

类型： 变量的输入类型。（当例示为功能模块时，输入功能模块）

初始化: 输入变量的值（与赋值符号":= "相同）

注释： 在这里输入一个注释。

声明编辑器的两个显示类型之间可以切换，在联机模式下，显示没有区别。

选择命令'插入' '新声明'来编辑一个新变量。

声明的表格形式

‘插入’ ‘新声明’

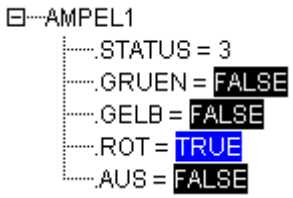
使用这个命令,将在声明编辑器的 声明表格中加入一个新变量，如果当前的光标位置在表格中区域，新变量将会粘贴在行的前面；否则，新变量粘贴在表格的最后。而且，通过使用右箭头或在表格最后的区域中使用TAB键，也能在表格的最后粘贴一个新声明。

你会得到一个变量名字位于'名称'区域、“Bool”值位于'类型'区域，作为它的默认设置。可以改变这些值为自己期望的变量完整定义所需的值、名字和类型。

联机模式中的声明编辑器

在联机模式中，声明编辑器变为一个监视窗口。在每个行中变量后面跟着等号(=)和它的值，如果在这个点上的变量是未定义的，会出现三个问号？？？。对于功能模块，只有打开的实例（命令：“工程”“打开实例”）才显示变量的值。

在每个多元素变量附近有一个加号，按<Enter>键或在变量上双击之后，打开变量。如例子交通信号结构将打开。



当打开一个变量，它的所有变量在它后面列出，减号出现在变量附近。如果你双击或按<Enter>键，变量将会关闭，加号将重新出现。

在单变量上按<Enter>键或双击将打开一个对话框来写变量，这里可以改变变量的当前值，在布尔变量的情况下，不出现对话框；这些变量被触发。

新值显示在变量之后，在突出的括号中颜色为青绿色并保持不变，如果执行了命令'联机''写入新值'，所有的变量被放于选中的列表中并显示为黑色。

如果使用了命令'联机''强制新值'，所有的变量将被设置为选中的值，直到执行了'强制赋值'命令。在这种情况下，强制值的颜色改变为红色。

5.3 声明编辑器中的预处理pragma指令

编程指令影响编译和预编译过程中变量的属性，在 声明编辑器的编程行中或在它自己的行中可以使用辅助的文本。

编程指令写在{}中，不区分大小写，如{ <Instruction text> }。

如果编译器不能解释指令文本，这个程序作为一个注释处理并读取一次，出现一个警告：“编译器忽略指令‘<Instruction text>’！”。

依赖于程序的类型和内容，Pragma 在它所在的行上操作,或在 pragma 结束前的所有行上操作，或执行了相同的带有不同参数的 pragma 前的所有行上操作,或到达文件末尾。这里的文件是声明部分,执行部分,全局变量列表和类型声明.

左括号应跟在变量名后面,左和右括号应在同一行中。

在 CoDeSys 中可用到下面的 Pragma：

Pragma{flag}用来初始化，监控，创建符号

Pragma{bitaccess...}用来访问位

Pragma{parameter..}, {template...}, {instance...}用来创建变量管理器中的条目

用于库声明部分显示控制的Pragmas.

初始化，追踪，符号创建，位存取的预处理Pragma指令

Pragma {flag [<flags>] [off|on]}

<flags>可以是下面标记的组合：

noinit:　　变量将不初始化

nowatch:　　将不再监视变量

noread:　　变量导出到一个非读权限的符号文件中

nowrite:　　变量导出到一个非读权限的符号文件

noread,　　变量将不导出到符号文件中

nowrite:

{flag on}开始 Pragma 对所有随后的变量声明操作直到{flag off}结束，或直到被其它{flag <flags> on} pragma 覆盖。

没有{flag on}和{flag off}时，Pragma 只对当前变量声明起作用（变量声明被下一个分号关闭）。

使用 pragma {flag}的例子：

变量的初始化和监视：

变量 a 将不初始化和被监视，变量 b 将不初始化：

```
VAR
a : INT {flag noinit, nowatch};
b : INT {flag noinit };
END_VAR
VAR
{flag noinit, nowatch on}
a : INT;
{flag noinit on}
b : INT;
{flag off}
END_VAR
```

两个变量都不初始化：

```
{flag noinit on}
VAR
a : INT;
b : INT;
END_VAR
{flag off}
VAR
{flag noinit on}
a : INT;
b : INT;
{flag off}
END_VAR
```

把变量添加到符号文件：

使用“noread” 和“nowrite”，在 POU 中有读和/或写权限，来为选中变量提供严格的访问权限。变量的默认的访问权限和变量定义的 POU 中的设置一样。如果一个变量没有读写权限，它不会导出到符号文件。

例如：

```
VAR
```

```
a : INT {flag noread};
b : INT {flag noread, nowrite};
END_VAR
VAR
{ flag noread on}
a : INT;
{ flag noread, nowrite on}
b : INT;
{flag off}
END_VAR
```

变量 a 和 b 都不导出到符号文件：

```
{ flag noread, nowrite on }
VAR
a : INT;
b : INT;
END_VAR
{flag off}
VAR
{ flag noread, nowrite on}
a : INT;
b : INT;
{flag off}
END_VAR
```

pragma 指令对后面的变量声明做附加操作..

例如：（所有使用的 POU 将会导出读和写权限）

```
a : afb;
...
FUNCTION_BLOCK afB
VAR
b : bfb {flag nowrite};
c : INT;
END_VAR
...
FUNCTION_BLOCK bfB
VAR
d : INT {flag noread};
e : INT {flag nowrite};
END_VAR
```

“a.b.d”：将不会被导出

“a.b.e”：导出后只具有只读权限

“a.c”：导出后具有读写权限

Pragma {bitaccess...} 用于位访问

这个 pragma 可以用来正确显示，在全局常量、输入帮助、智能功能变量在声明窗口监视的帮助之下进行

位访问操作的变量。当在特殊的 POU 的声明窗口中监视这个变量时，使用的全局常量在各自结构体变量下面。

注意: 必须激活工程->选项->生成-> ’替换常量’ ！

Pragma 必须插入到结构体的声明中的一个单独的行。这个行不能用分号终止。

句式：

{bitaccess <Global Constant> <Bitnumber> ’<注释>’}

<Global Constant> : 全局常量的名称, 它必须在全局变量列表中定义.

<Bitnumber> : 全局常量数值,在全局变量列表中定义.

例如,在变量中给位赋地址

Pragma {link} 用于在代码产生过程中联接 POU

通常情况下,在工程中不被调用的 POU(程序,功能,功能块)或数据类型定义(DUT),在代码产生中不被联接.

但是如果一个通过库包含在工程中的功能,即使不被应用程序直接使用(例如,检查操作),那么在下载到运行系统后也应该可以使用.由于这个原因,为了与 POU 联接,你可以在 POU 或 DUT 声明部分的任一位置添加 {link} pragma 指令.

参数管理入口的Pragma指令

Pragma 指令可以插入到变量声明中为了在变量管理器中处理的变量列表中为这些变量创建条目。它依赖于目标系统，变量管理器在 CoDeSys 编程系统中是否可用。它必须在目标设置的类别网络功能中激活。

句式：

Pragma 指令写在{}号中，不区分大小写：{<Instruction text>}。如果它包含在标准变量的声明中，它必须在声明的分号之前设置。

在界面 VAR_CONFIG 中使用的 Pragma 在每个单独的行上设置并且不用分号结束!

关键字定义由空格字符隔开，其它的包括在方括号中。

<name>:在变量管理器中列出的变量的名字。如果列表不存在，它将会被创建。

<key>:属性的名字，例如，在变量列表中的列标题；例如，"Name"、 "Value"、"Accesslevel" 等等；它依赖于变量列表类型的定义，键可以在 pragma 中定义。

<value>:用<key>定义的属性的值。

请注意：pragma 指令在焦点改变后有效，在预编译之后有效。例如，错误的输入直到工程被编译才显示出错消息。

1. 在参数列表中类型’变量’的输入项

(a) 来自程序和全局变量列表的声明部分:

句式：{parameter list=<name> [<key>=<value> <key>=<value> ...further keys] }

例如：

```
VAR
bvar:INT{parameter list=parlist1 [name=bvar1 value=102 index=16#1200
subindex=16#1 ] };
END_VAR
```

(b) 借助 VAR_CONFIG 接口中的一个声明:

通过放置一个 pragma 在 VAR_CONFIG 窗口中,可以在变量列表中类型’Variables’中为变量创建一个条目。（变量配置的独立性，也可以在 VAR_CONFIG 界面来完成）：

句式：{parameter list=<name> path=<path> [<key>=<value> <key>=<value> ...further keys] }

例如：对于变量 var_x 在参数列表"varlist1"创建一个输入项, 符号名称是 xvar".

```
VAR_CONFIG
{parameter list=varlist1 path=PLC_PRG.act1.var_x [ name=xvar ] }
```

END_VAR

2. 借助于功能块和结构,在参数列表中类型'模板'的输入项

在类型'Template'的变量列表中能使用 Pragmas 来创建条目，Pragmas 在功能模块和结构体的变量声明中。

句式：{template list=<name> [<key>=<value> <key>=<value> ...further keys] }

例如：变量 strvar 是结构体"stru1"的一个元素，它应该输入到类型'Template'的变量列表"templ1"中；条目的符号名字是“struvar1”, 访问级别为 "low"：

```
TYPE stru :
STRUCT
ivar:INT;
strvar:STRING{template list=vorl1 [member=struvar1 accesslevel=low] };
END_STRUCT
END_TYPE
```

3. 在参数列表中类型'实例'的输入项 (用于数组,功能块或结构变量)

(a) 来自程序和全局变量列表的声明部分:

在程序中或在全局变量列表中定义数组、功能模块或结构体变量时可以直接创建类型'Instance'的变量列表：

句式：{instance list=<name> template=<template> baseindex=<index>
basesubindex=<subindex> [<key>=<value> <key>=<value> ...further keys] }

注意：不要给关键字"Member"定义值；这个列使用数组索引值自动填充。

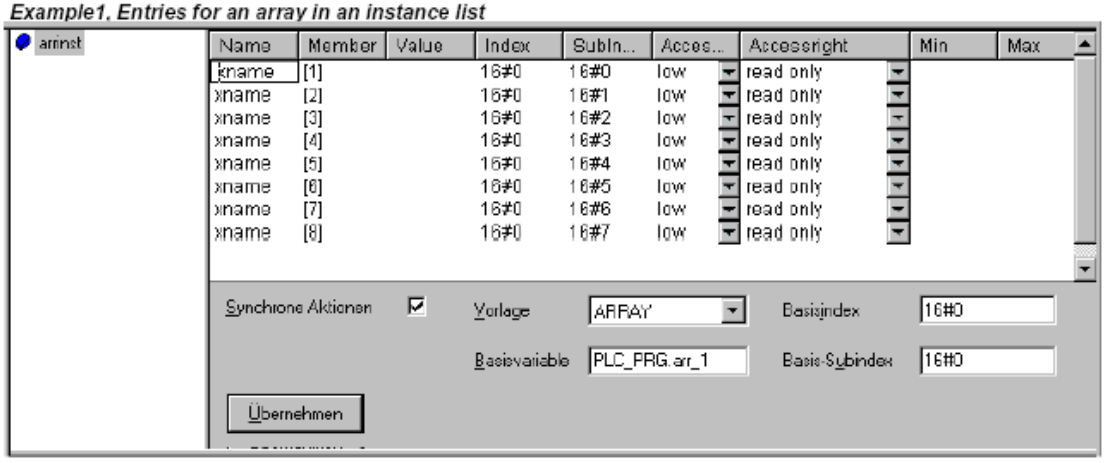
例如：

例子 1；

为了得到类型'Instance'的变量列表"arrinst"中的条目接下来我们定义一个数组变量"arr_1"；在这个列表中所有的数组元素得到一个符号名"xname"（能在变量管理器中编辑）和分索引从零开始计数每个条目加 1。

```
arr_1: ARRAY [1..8] OF INT{instance list=arrinst template=ARRAY baseindex=16#0
basesubindex=16#0 [name=xname ] };
```

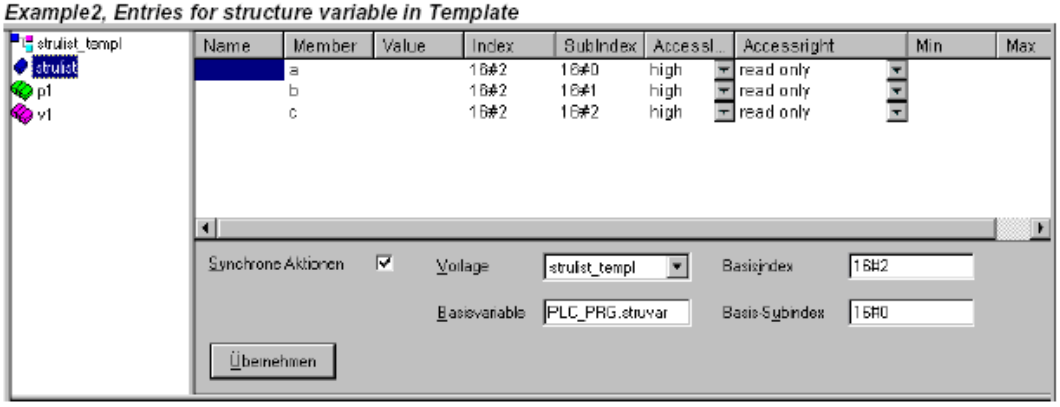
Example1. Entries for an array in an instance list

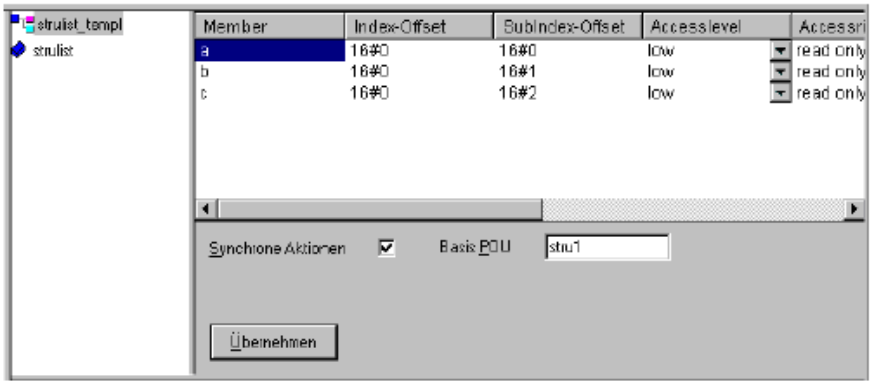


例子 2：

为了在类型'Instance'的变量列表中得到条目

Example2, Entries for structure variable in Template





，象下面讲述的一样，我们定义一个类型为"stru1"的结构体变量（包含变量 a，b，c）。结构体变量是基于模板"strulist_temp"；列表得到变量 a，b，c 的条目，没有分配符号名，访问级别实质为高并且每个通过模板定义的索引值将会增加 2。确定变量管理器中的 pragma 中定义的模板是可用的：

struvar:stru1{instance list=strulist template=strulist_templ baseindex=16#2 basesubindex=16#0 [accesslevel=high] }；

(b) 借助 VAR_CONFIG 接口中的声明

通过在 VAR_CONFIG 窗口（变量的独立性配置也可以在这个界面中定义）中的一个 pragma 你可以直接地在类型'Instance'的变量列表中创建条目。

确定变量管理器中的 pragma 中定义的模板是可用的：

句 式 ： {instance list=<name> path=<path> template=<template> baseindex=<index> basesubindex=<subindex>[<key>=<value> <key>=<value> ...further keys] }

<path>:变量的实例路径；例如，"PLC_PRG.fb1"，fb1 是一个功能模块。

在 VAR_CONFIG 窗口中，下列条目将为在基于模板"fb1_templ"的实例列表"varinst1"中的功能模块"fb1"的所有变量创建条目。对每个条目，模板预先定义的索引将按照 2（基索引）增加来偏移，分索引偏移不会被修改。每个条目得到一个符号名"fb1var"，符号名字可以在变量管理器中编辑。

```
VAR_CONFIG
{instance list=varinst1 path=PLC_PRG.fb1 template=fb1_templ baseindex=16#2
basesubindex=16#0 [ name=fb1var ]}
END_VAR
```

控制库声明部分显示的语法

在 CoDeSys 中创建一个库时，你可以定义部分声明可视窗口和管理库中不可视的布拉格，当库被包括在一个工程中时。库中执行部分的显示将不受影响。

因而，注解或任何可视的声明可以被用户隐藏。语法【私有库】和【公共库】每一个影响一些线的剩余部分，后来的线，只要他们没有被其他部分过多的写入。

语法：【公共库】后续的测试将显示在管理库中。

【私人库】后续的测试将不被显示。

例子：注意在 CoDeSys 中被声明的部分库下。注解"(* this is for all *)" 在已经包括一个工程库的管理库中会被显示。注解"(* but this is not for all*)" 将不会显示出来。本地变量和 in2 也不会被显示出来：

```
{library public}(* this is for all *) {library private} (* this is not for all *)
{library public}
FUNCTION afun : BOOL
VAR_INPUT
in: BOOL;
END_VAR
{library private}
VAR
local: BOOL;
END_VAR
{library public}
VAR_INPUT
in2: BOOL;
{library private}
in3: BOOL;
{library public}
END_VAR
```

非永久数据类型的语法

正常情况下下列将被显示：尽管在功能块中仅有一个本地变量或一个结构是持久的，在运行时间系统中的持久信息（persist.dat）中一个自动存储所有成分的实例被使用。为了保春你能使用布拉格的地方。

在功能块的分离结构声明中，它只影响那些被注明"persistent"的功能块的分离结构，他们将会被登陆到稳定信息当中。

例如：

如果一个如下功能块的实例被注明是稳定的，只局部可变且 fblevel3 将要被写入到稳定信息中。除了布拉格{nonpersistent}

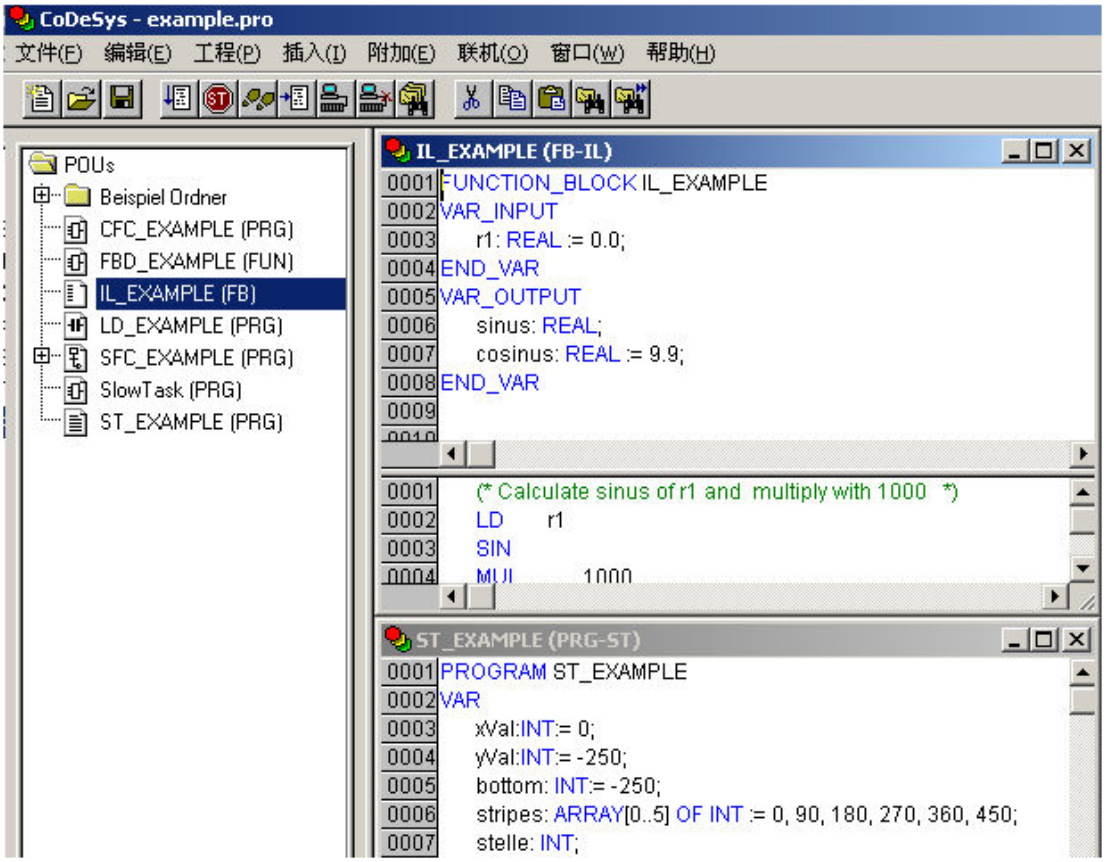
所有功能块的变化将被储存到那里。

```
FUNCTION_BLOCK FB_Level_2
{nonpersistent}
VAR_INPUT
bvar_in : BOOL;
END_VAR
VAR_OUTPUT
bvar_out : BOOL;
END_VAR
VAR
ivar2 : INT;
END_VAR
VAR PERSISTENT
local : INT := 33;
```

```
fblevel3 : FB_Level_3;
END_VAR
```

5.4 文本编辑器

文本编辑器为 CoDeSys 的执行部分提供通用的窗口文本编辑器功能。
句式颜色支持文本编辑器中的执行部分。



在改写模式下 状态栏 显示一个黑色的OV，你可以通过<Ins>键来在改写模式和插入模式之间切换
在内容菜单（鼠标右键或<Ctrl>+<F10>）中包含有重要的命令。
文本编辑器使用下面的菜单命令：
参照:
’插入’ ’算子’
’插入’ ’操作数’
’插入’ ’功能’
’插入’ ’功能块’
调用带有输出参数的POUs
联机模式下的文本编辑器
’扩展’ ’监视选项’
断点位置
如何设置断点
删除断点
断点处的状态
在文本编辑器中行号码
指令表编辑器

结构化文本编辑器

'插入''操作符'在文本编辑器中

使用这个命令，当前语言中用到的所有运算符将显示在对话框中。如果选中一个运算符按 OK 列表关闭，然后高亮度显示的运算符将插入到当前光标位置。(象在输入帮助中一样操作)。

'插入''操作数'在文本编辑器中

使用这个命令将显示在对话框中的所有变量，你可以从全局变量、局部变量、系统变量中选择要显示那个变量的列表。

如果选中了其中的一个操作数，按'确定'后对话框关闭，高亮度显示的操作数将会插入到当前的指针位置（和输入助手的操作一样)。

'插入''功能'在文本编辑器中

使用这个命令在对话框中将显示所有的 功能,你选择是显示用户定义的列表显示还是标准功能列表显示。

如果选中了一个功能，按 OK 键关闭对话框，高亮显示的功能将插入到当前指针位置。

如果在对话框中选中了'带形参'选项，必须的输入和输出变量也会插入进来。

'插入''功能块'在文本编辑器中

使用这个命令在对话框中显示所有的 功能块,你可以选择显示用户定义的列表还是标准功能模块的列表。

如果选中一个功能模块，按'确定'将关闭对话框，高亮度显示的功能模块将插入到当前指针位置。

如果在对话框中选择了'带形参'选项，功能模块的必要的输入和输出变量也将插入进来，但是你不能强制给这些变量赋值。

在文本编辑器中调用带有输出参数的POUs

在文本语言 IL 和 ST 中，可以直接为被调用 POU 的输出变量赋值 。

例如：

Afbinst 的输出变量 out1 赋予变量 a .

```
IL: CAL afbinst(in1:=1, out1=>a)
ST: afbinst(in1:=1, out1=>a);
```

如果通过输入帮助(<F2>)用选项“With arguments”在 ST 或 IL POU 的执行窗口中插入一个 POU，它将自动和所有的变量以这个句法格式显示出来，但不能强制这些变量赋值。

联机模式下的文本编辑器

在编辑器中的联机功能是用来设置断点和单个步处理（步)，和监视功能结合起来，用户就拥有了标准语言调试器的调试功能。

在联机模式，文本编辑器窗口垂直分为两部分，在窗口左边的部分是标准程序文本，在右边是显示的变量，它的值在各自的行上被改变。

显示与声明部分的显示一样。当 PLC 运行时，将会显示各自变量的当前值。当监视表达式或位地址变量时要注意以下：在监视表达式时，通常显示整个表达式的值。例如，a 和 b 显示为蓝色或如果 a 和 b 的值是 TRUE 显示“：=TRUE”。对于位地址变量，位值通常被监视（例如，a.3 显示为蓝色或如果 a 有值 4)。

如果你用鼠标在变量上停留一会，变量的类型、地址、注释将会在工具提示中显示出来。

'附加''监控选项'

用这个命令可以配置监视窗口。在文本编辑器中，在监视过程中窗口被分成两部分，程序加载在左边部分，在右边部分中，相应程序行中的变量被监视。

可以指定监视窗口的宽度和在一行中两个变量之间的距离，声明距离为 1 对应选中一行字体的高度。

在文本编辑器中断点位置

在 CoDeSys 中多个 IL 行组合为一个 C 代码行，因此断点不能在每一行中设置。断点的位置包括在程序中变量的值改变的地方或程序流程图中断的地方的所有位置。（例外：功能调用，如果必须，功能中的断点必须设置在这里。）

在 IL 中提供了下面的断点位置：

•在 POU 开始的地方

•在每个 LD、LDN 处

•在 JMP、JMPC、JMPCN 处

•在每个标签处

•在每个 CAL、CALC、CALCN 处

•在每个 RET、RETC、RETCN 处

•在 POU 结束的地方

结构化文本提供下面的断点位置：

•在每个赋值处

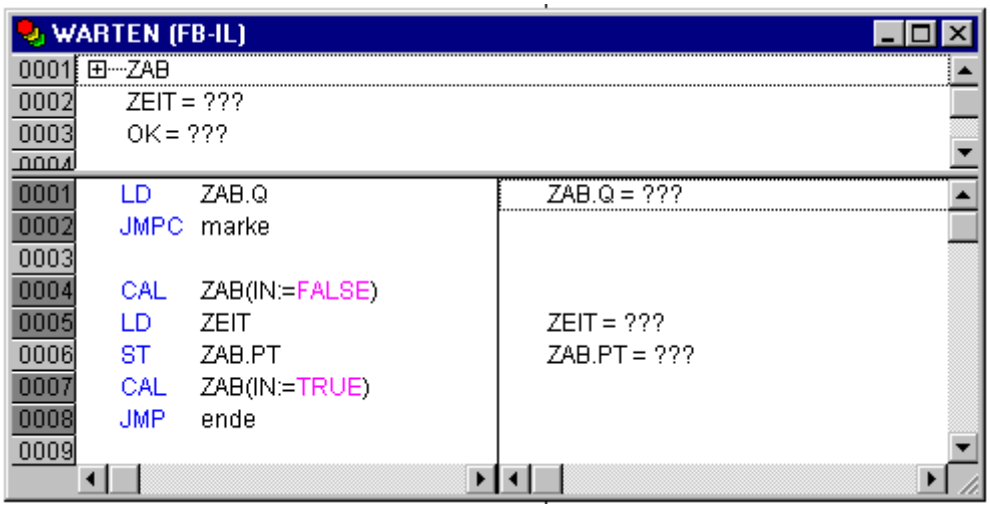
•在每个 RETURN 和 EXIT 指令处

•在计算条件的行处(WHILE, IF, REPEAT)

•在 POU 结束的地方

在行号码区域处，用在工程选项中设置的颜色来标记断点位置。

可能断点位置的IL编辑器（暗色数字部分）



如何设置断点

为设置一个断点，在你要设置断点的行的行号码区域单击，如果选中的区域是断点的位置，行号码区域的颜色将从黑灰色转变为淡蓝色并且断点将在 PLC 中激活。

删除断点

相应地，为删除一个断点，在有断点的行的行号码区域单击，断点将删除。

设置和删除断点也可以通过菜单命令('联机' '删除断点')、通过功能键<F9>或通过在工具栏中的符号来

选择。

断点处的状态

在 PLC 中到达断点时，屏幕上会显示断点和相应的行，行的行号码区域将显示为红色，用户程序在 PLC 中停止。

如果程序到达断点，处理过程可以通过 '联机' '运行' 命令来继续。另外，用 '联机' '跳过' 或 '进入' 命令，你可以使程序在下一个断点位置运行。如果你所在处的指令是CAL或如果在行中直到下一个断点位置处有一个功能调用，你可以使用命令“跳过”越过功能调用。用“进入”，将进入打开的POU分支。

在文本编辑器中行号码

文本编辑器的行号码给出了 POU 的执行部分的每个文本行的号码。

在脱机模式下，在一个特定的行号码上单击将会标记整个文本行。

在联机模式下，行号码的背景颜色指示了每个行的断点的状态。

颜色的标准设置有：

•黑灰色：这个行是断点的可能的位置。

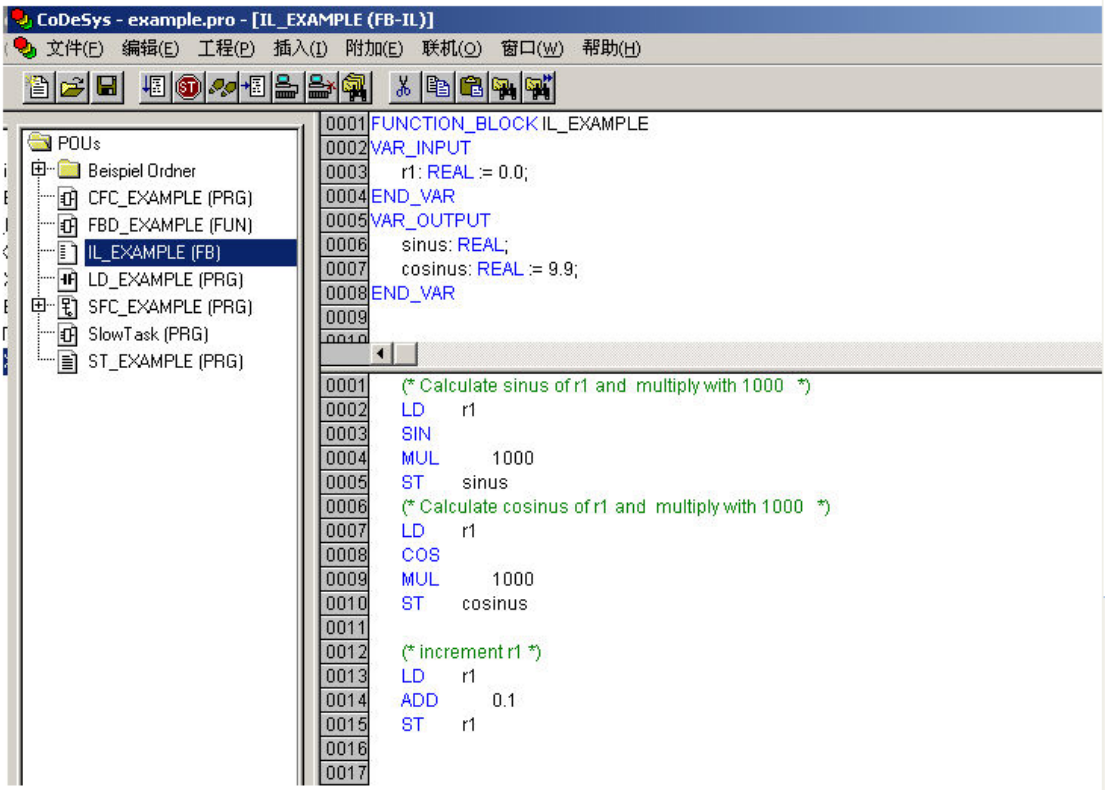
•淡蓝色：断点已经在这个行中设置。

•红色： 程序到达了这个断点。

在联机模式下，单击鼠标将会改变这个行中的断点状态。

5.4.1 指令表编辑器

下面是在 CoDeSys 中相应的编辑器 IL 中编写 POU 的窗口。



POU 的所有编辑器有声明和主体部分，它们被屏幕分割器分割开。

指令列表编辑器是具有文本编辑器窗口功能的文本编辑器，在内容菜单（鼠标右键或<Ctrl>+<F10>）中可以找到最重要的命令，多行 POU 调用也是可以的：

例如：

CAL CTU_inst(

```
CU:=%IX10,
PV:=(
LD A
ADD 5
)
)
```

关于语言的信息，查看 2.2.1 章节，指令列表(IL)。

另见:

联机模式下的IL

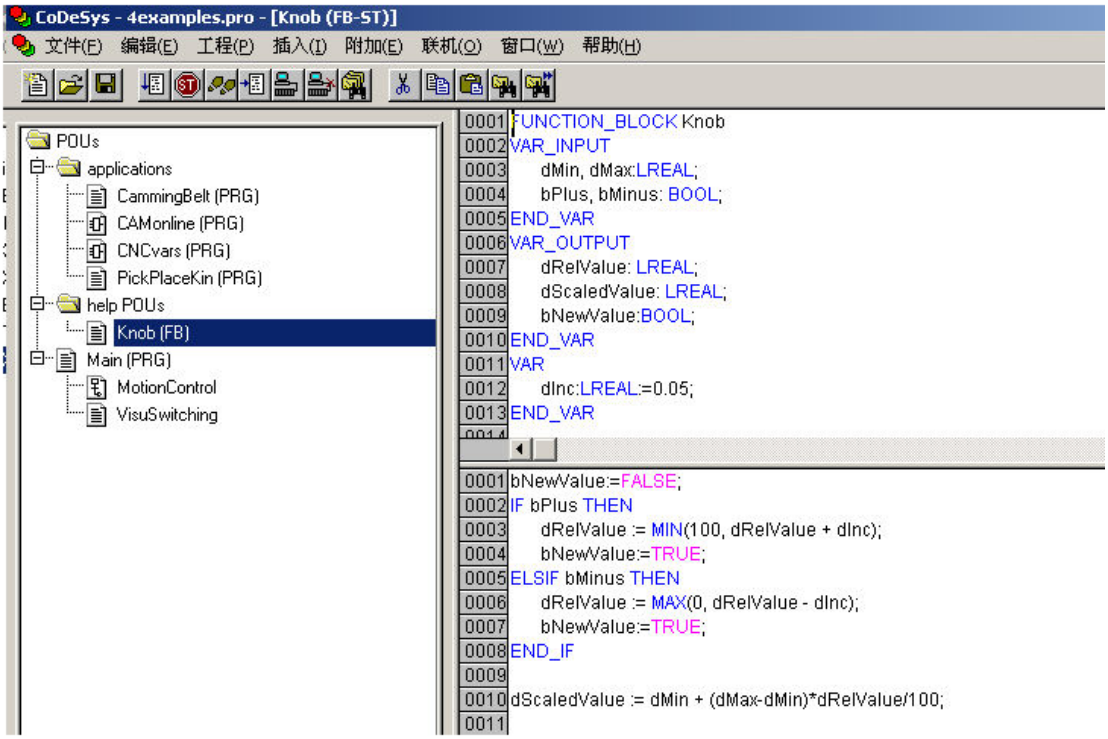
联机模式下的IL

用命令’联机’’溢出控制’，附加的域插入到IL编辑器中左边部分的每一个行中。

在联机模式下关于IL的更多信息，参照’联机模式下的文本编辑器’

5.4.2 结构化文本编辑器

下面是在 CoDeSys 中相应的编辑器 ST 中编写 POU 的窗口。



POU 的所有编辑器有声明和主体部分。它们被屏幕分割器分割开。

结构化文本编辑器是具有文本编辑器窗口功能的文本编辑器，在内容菜单（鼠标右键或<Ctrl>+<F10>）中可以找到最重要的命令。

关于语言的信息，查看 2.2.2 章节，结构化文本。

5.5 图形化编辑器

图形化导向的语言编辑器、顺序功能图 SFC、梯形图和功能模块图和自由图形化功能模块图有许多共同点。在下面的段中将总结这些特点；关于 LD、FBD、CFC 和顺序功能图表语言 SFC 在分开的部分中讲述，句式色彩支持图形化编辑器的执行部分。

参考:

缩放
网络
标志
网络注释,’扩展’ ’选项’
’插入’ ’网络（后面）’ 或者’插入’ ″网络（前面)″
联机模式下的网络编辑器
功能块图编辑器
梯形图编辑器
顺序功能图编辑器
连续功能图编辑器

缩放

在语言 SFC、LD、FBD、CFC 和在可视化中的对象如 POU、动作、转换等可是通过缩放功能来放大或缩小尺寸，执行部分的窗口内容中的所有元素都将受到影响，声明部分保持不变。

在标准形式中，每个对象以缩放级别 100%显示，缩放级别在工程中存储为对象的属性。

工程文档的打印预览通常显示为 100%。

缩放级别可以在工具栏中通过选择列表来设置，它的值可以从 25% 到 400%之间选择，在 10%和 500%之间的值可以通过手工输入。

如果光标停留在在图形化语言创建的对象上或停留在一个可视化对象上，就可以选择缩放级别。

在联机模式，每个对象根据设置的缩放级别来显示，联机功能没有任何限制。

当使用智能鼠标时，通过按<CTRL>键并同时向前或向后旋转鼠标轮能够放大/缩小对象。

网络

在 LD 和 FBD 编辑器中，程序由一系列的网络组成，每个网络都由左边的网络编号指示，并且还包括逻辑或算术表达式的结构、程序、功能或功能模块调用、跳转或返回指令。

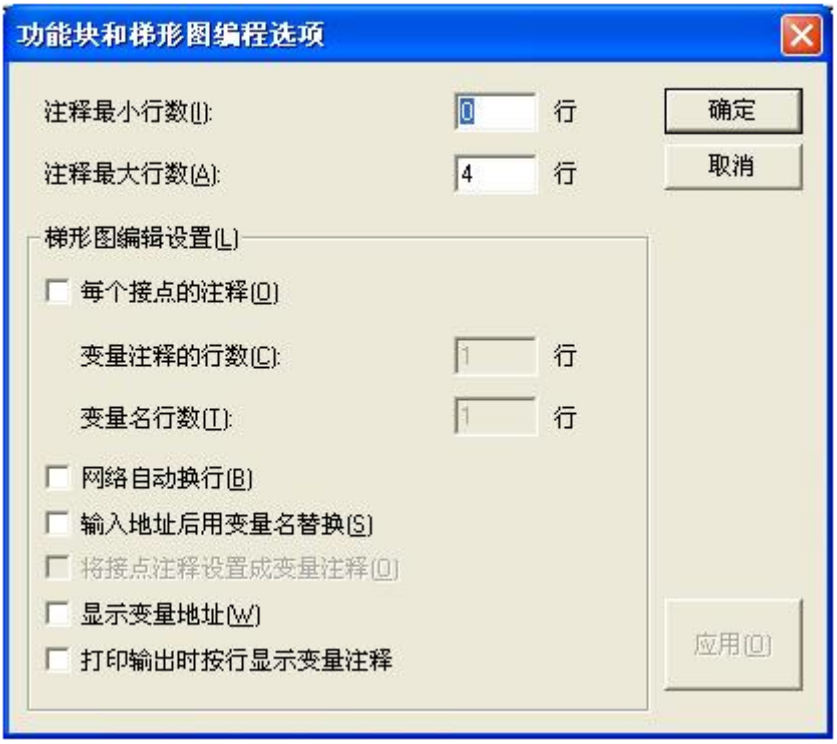
标签

每个网络都有一个标签，标签可以为空，通过单击网络编号旁的网络的第一行来编辑标签，然后在冒号后面输入一个标签。

注释，网络中的行断点，’附加’ ’选项’

每个网络都可以添加一个多行的注释，在通过命令“附加”“选项”打开的对话“功能块和梯形图选项”中，可以设置关于注释的设置。

功能块和梯形图选项

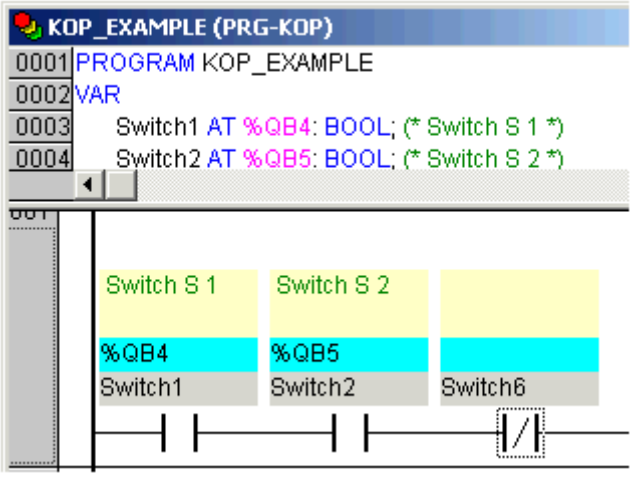


在最大注释尺寸区域中，可以输入注释行的最大的数字（默认值为 4），在最小注释尺寸区域中可以输入通常要保留的行数字。如果，输入了 2，那么，在每个网络的开始处在标签行后面都有两个空行，默认的值为 0，这样做的优点是使更多的行显示在屏幕区域。

如果最小注释数字比 0 大，为了输入一个注释只需在注释行行单击然后输入注释，否则，就需要首先选择要输入注释的网络，并使用命令“插入”“注释”来插入注释行。和程序文本相比，注释显示为灰色。

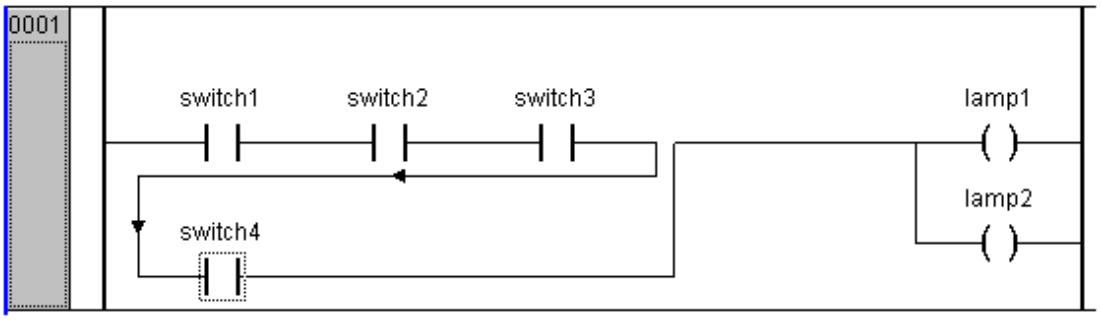
在梯形图编辑器中可以为每个接点和线圈添加注释，只要激活选项注释每个接点并插入到编辑区域变量注释行输入保留用来显示注释的行的编号，当完成这个设置时，注释区域将显示在编辑器中每个触点和线圈的上面。

如果激活选项注释每个接点，在梯形图编辑器中可以定义哪个行编号用于接点或线圈的变量名，这用来显示由多行组成的长名字。



在梯形图编辑器中，通过激活选项网络自动换行，在网络的长度超出给定的窗口尺寸并且一些元素看不见的情况下可以自动换行。

自动换行例子

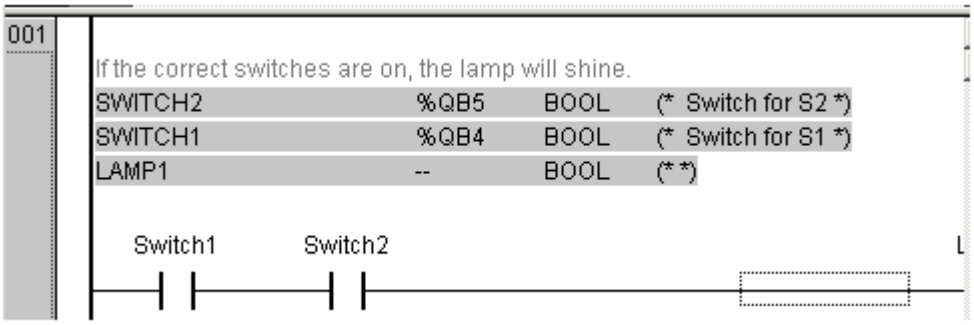


输入地址后用符号替换：(只适用于梯形图编辑器)：如果此项激活,那么在接点或线圈处输入地址后,如果地址已经分配给了一个变量,地址将被变量名替换,否则保留地址.

接点注释设置成变量注释： 如果此项激活，将在接点或线圈注释区内显示变量声明时编写的注释.注释可以再编辑(参看上面的图片），为达到这个目的,必须激活'每个接点的注释'选项.注意:即使变量在声明区没有注释,原有注释也会自动被变量声明区中的注释替换!

显示符号地址：(只适用于梯形图编辑器)：如果此项激活同时接点或线圈的变量与一个地址对应,那么将显示在变量名上面.(参看上面的图片).

打印输出时按行显示变量注释：如果此项激活,每个网络中用到的变量各占一行,被显示在网络上面,显示的内容:名称,地址,数据类型和注释.这和变量声明一样. 此项功能在工程打印输出时很有用. 例如:



应用选项:

确定：按此按钮将当前设置用于当前 POU 并关闭对话框.

应用：按此按钮将设置用于整个工程. 按此按钮后将打开一个提示框来确认是否修改设置.

'插入''网络（后面）' 或者'插入'"网络（前面）

快捷方式：<Shift>+<T>（网络后面）

为了在 FBD 或梯形图编辑器中插入一个新网络，选择命令“插入”“网络（后面）”或'插入'"网络（前面)"，取决于你想在当前网络之前或之后插入一个新网络。当前的网络可以通过在网络编号上单击来改变，可以通过在编号下面的点划矩形框来识别它，使用<Shift key>和单击鼠标可以选中从当前到单击的网络的整个区域，

联机模式下的网络编辑器

在 FBD 和 LD 编辑器中只能为网络设置断点，设置断点的网络的编号区域显示为蓝色。处理过程在断点设置的网络附近停止，在这种情况下，网络编号区域显示为红色，通过单步处理可以在网络之间跳转。

在进入和退出网络 POU（程序组织单元）时，所有的值都被监视。

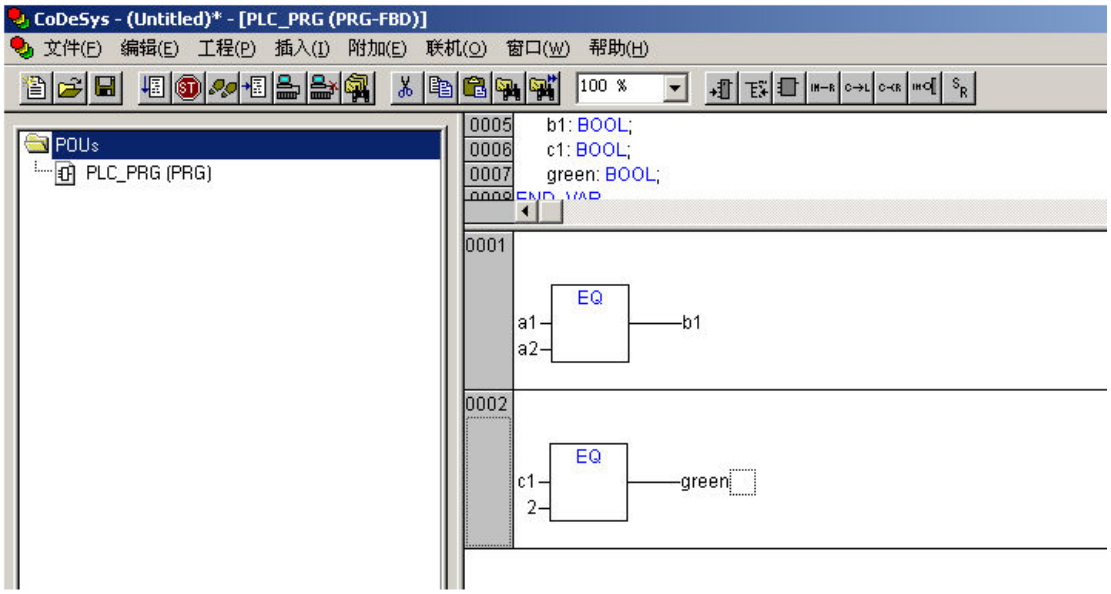
当监视表达式或位地址变量时，应注意以下方面：在表达式中，例如，a AND b，作为变换条件或功能模块输入，整个表达式的值通常显示（如果 a AND b 的值为 TRUE 时， a AND b 显示为蓝色或：=TRUE)。对于位地址变量，寻址的位值通常要被监视（例如，当 a 有值 4 时，a.3 显示为蓝色或带:=TRUE）

’联机’ ’流程控制’命令启动流程控制，使用流程控制，可以查看当前网络连接线上传递的值。如果连接线没有传递布尔型的值，值将显示在一个特定的输入区域。未使用的变量的监视区域（例如，在功能SEL中）显示灰色的阴影，如果连接线传递布尔型的值，当它们传递TRUE时，将显示为蓝色。所以，在PLC运行时可以观察流程的信息。

如果用指针在变量上停留一会，在工具提示中将显示变量的类型、地址和注释。

5.5.1 功能模块图编辑器

下面是在 CoDeSys 中相应的编辑器 FBD 中编写一个 POU 的窗口



功能模块图编辑器是一个图形化的编辑器，每个网络包含了显示、逻辑或算术表达式、功能模块的调用、功能、跳转、返回指令。在内容菜单（按鼠标右键或<Ctrl>+<F10>）中可以找最重要的命令。

可以在 FUP-和 KOP 之间切换 FUP-POU 的显示，离线模式和联机模式相同。

参照:

FBD的当前位置

如何设置FBD的当前位置

’插入’ ’赋值’

’插入’ ’跳转’

’插入’ ’返回’

’插入’ ’框’

’插入’ ’输入’

’插入’ ’输出’

’扩展’ ’否定’

’扩展’ ’设置/复位’

’扩展’ ’预览’

’扩展’ ″快速’

’扩展’ ’打开实例’

’扩展’ ’选项’

剪切，复制，粘贴和删除

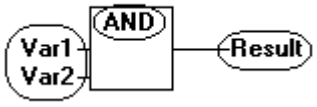
联机模式下的功能块图

FBD的当前位置

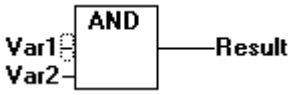
每个文本是一个可能的光标位置，选中的文本以蓝色背景表示并且当前可以被修改。

通过点矩形框也能识别当前光标位置，下面是光标所有可能位置的例子：

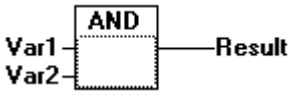
1）每个文本区域（加黑色框的可能指针位置）：



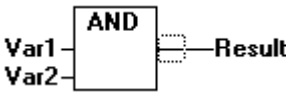
2）每个输入：



3）每个操作数、功能或功能模块：

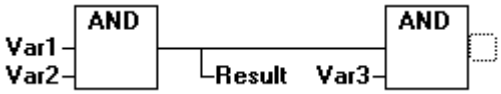


4）输出，如果随后有赋值或跳转：

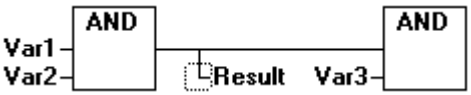


5）行在赋值、跳转、返回指令上发生交叉：

6）在最外面对象的网络图的右边（“最后的指针位置”，与选择网络的指针位置相同）：



7）在赋值附近的线性交叉：



如何设置FBD的当前位置

通过单击鼠标或使用键盘可以在特定的位置放置光标，在任何时候，使用箭头键可以在选择的方向上跳到最近的光标位置。通过这种方式可以访问所有的光标位置包括文本区域。如果最近的光标位置被选中， 就可以使用<up>或 <down>方向键来选择先前的网络或随后的网络的最近光标位置。

空的网络只包含三个问号“？？？”，在它们之后单击，最近的光标位置被选中。

’插入’ ’赋值’

符号 ：  快捷方式：<Ctrl>+<A>

这个命令插入一个赋值。

插入赋值依赖于选中的位置（参照’FBD的当前位置’），在选中的输入端附近（光标位置 2）、在选中的输出端（光标位置 4）之后或在网络的末端（光标位置 6）可以直接插入赋值。

对于一个插入的赋值随着选中输入的文本“？？？”被选中，赋值可以被将要赋值的变量替代，也可以

使用输入帮助。

使用命令“插入”“输出”给当前存在的赋值插入一个附加的赋值。

’插入’ ’跳转’

符号 ： 快捷方式：<Ctrl>+<L>

这个命令插入一个跳转。

插入依赖于选中的位置（参照’FBD的当前位置’），在选中的输入端（光标位置 2）附近、在选中的输出端（光标位置 4）后或在网络的末端（光标位置 6）可以直接插入跳转。

对于插入的跳转，随着选中输入的文本“？？？”选中跳转，它可以被要赋予它的标签替代。

’插入’ ’返回’

符号 ： 快捷方式：<Ctrl>+<R>

这个命令插入一个返回指令。

插入返回指令依赖于选中的位置（参照’FBD的当前位置’），在选中的输入端附近（光标位置 2）、在选中的输出端（光标位置 4）之后或在网络的末端（光标位置 6）可以直接插入返回指令。

’插入’ ’框’

符号 ：  快捷方式：<Ctrl>+<B>

用这个命令能插入运算符，功能，功能模块和程序。首先，它通常插入一个“AND”运算符，这可以通过选择和类型文本的覆盖来改变到其它操作符，功能，每个功能模块和每个程序中。

通过使用输入帮助(<F2>)可以选择期望的 POU。如果新选择块中有其它输入的最小数字，这些将绑定。如果新块中有输入的较小数字，最近的输入将被删除。

在功能和功能模块中，显示输入和输出的正式名字。

在功能模块中的方框上存在一个可编辑的实例区域。通过改变类型文本来调用一个没名的功能模块，将显示带有两个输入和给定的类型的运算符方框。如果实例区域被选中，可以通过<F2>和变量选择的类别获得输入帮助。

最新的 POU 插入到选中的位置：

如果选中一个输入端（光标位置 2），POU 插入到这个输入端的附近。这个 POU 的第一个输入端连接到选中输入端左边的分支。新 POU 的输出连接到了选中的输入端。

如果选中了一个输出端（光标位置 4），那么 POU 插入到输出端之后。POU 的第一个输出端与选中的输出端连接。新 POU 的输出端与选中的输出端连接的分支相连接。

如果 POU、功能或功能模块被选中（光标位置 3），旧的 POU 将被新的 POU 替代。

反之将会与它们被替代之前的相同方式相连接。

如果旧元素的输入端比新元素多，不能连接的分支将被删除，相同的分支为输出端保持真值。

如果选中了跳转或返回指令，POU 将会插入到这个跳转或返回指令之前。POU 的第一个输入端与选中元素的左边的分支相连接。POU 的输出端与选中的元素的右边的分支相连接。

如果选中了网络的最近光标位置（光标位置 6），POU 将会插入到最后元素的后面。POU 的第一个输入端与选中位置的左边的分支相连接。

不能连接的 POU 的所有输入端出现文本“？？？”。这个文本必须改变为期望的常量或变量。

插入的 POU 的右边分支，分支将会分配给第一个 POU 输出端。否则输出端保持不赋值。

’插入’ ’输入’

符号 ：  快捷方式：<Ctrl>+<U>

这个命令插入一个输入端运算符。随着运算符，输入端的数目可以不同（例如，ADD 可以后 2 个或多个输入端。）

为了通过输入来扩展这样一个运算符号，你必须选择输入端，在它的附近插入附加输入端（光标位置 1）；或者你必须选择运算符本身（光标位置 3），如果要插入一个低输入端（参照'FBD的当前位置'）。插入的输入端分配了文本“？？？”。这必须改变为期望的常量或变量。你也可以使用输入帮助来做这些。

'插入''输出'

符号 ： 

这个命令用来添加一个附加的赋值到已存在的赋值中。

'附加''取反'

符号 ：  快捷方式：<Ctrl>+<N>

用这个命令可以对输入、输出、跳转或返回指令进行否定操作，否定的符号是在连接处的一个小圆圈。

如果选中了一个输入（光标位置 2）（查看'FBD的当前位置'），随后这个输入将被否定。

如果选中了一个输出端（光标位置 4），那么这个输出端将被否定。

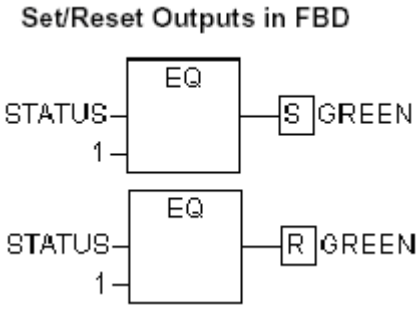
如果一个跳转或返回被标记，那么跳转或返回将被否定。

否定可以通过重新否定来取消。

'附加''置位/复位'

符号 ： 

用这个命令可以象设置或复位那样来定义输出，设置的输出用[S]表示，复位输出端用[R]表示。



如果属于它的栅格返回了 TRUE，输出端设置为 TRUE，输出将保持这个值，即使栅格跳回了 FALSE。

如果属于它的栅格返回了 FALSE，输出端设置为 FALSE。输出现在保持这个值，即使栅格跳回了 FALSE。

随着重复执行这个命令，输出将在设置、复位和正常输出之间改变。

'附加'预览'

这个命令为在 FBD 编辑器中创建的 POU 选择显示在梯形图或者 FBD 编辑器中，离线模式和联机模式功能相同。

'附加''打开实例'

这个命令与“'工程''打开实例' 相同。

如果光标放置在文本编辑器中的功能模块的名字上或如果功能模块框在图形编辑器中被选中，在内容菜单(<F2>)或在'附加'菜单中可用到这个命令。

剪切，复制，粘贴和删除

在菜单项目“编辑”下可以用到“剪切”,“复制”“粘贴”和“删除”

如果选中了一个交叉线（光标位置 5），那么位于交叉行下面的赋值、跳转或返回将会被剪切、删除或复制。

如果选中了一个 POU（光标位置 3），选中的对象自身将被剪切、删除或复制，同时也包括在输入端的所有独立的分支和第一种情况外的分支。

否则，在光标位置附近的所有分支将会被剪切、删除或复制。

在复制或剪切之后，删除或剪切的部分位于剪贴板上，可以随意粘贴它。

首先选中粘贴点，有效的粘贴点包括输入和输出端。

如果 POU 被加载到了剪贴板上（这种情况下所有连接的分支除了第一条外都一块位于剪贴板上），第一个输入连接到粘贴点之前的分支上。

否则，位于粘贴点附近的所有分支将被剪贴板中的内容替代。

最后粘贴的元素将连接到粘贴点附近的分支上。

注意：下面的问题可以通过剪切和粘贴来解决：一个新的操作数插入到网络的中间，分支位于操作数的右边，现在将和第一个输入相连接，但应该和第二个输入相连接。你可以选择第一个输入端并执行命令“编辑”“剪切”，这样，你可以选择第二个输入端并执行命令“编辑”“粘贴”，分支就连接到了第二个输入端。

联机模式下的功能块图

在功能模块图，断点只能设置到网络上。如果已经在网络上设置了断点，网络编号区域将显示为蓝色，处理过程将停止在设置断点的附近。在这种情况下，网络编号区域将显示为红色。使用 单步，你可以从一个网络跳到另一个网络。

显示每个变量的当前的值，例外：如果功能模块的输入是一个表达式，只有表达式中第一个变量才被监视。

在变量上双击将打开写变量的对话框，在这里可以改变当前变量的值。如果是布尔变量的情况，不出现对话框，这些变量被选中。

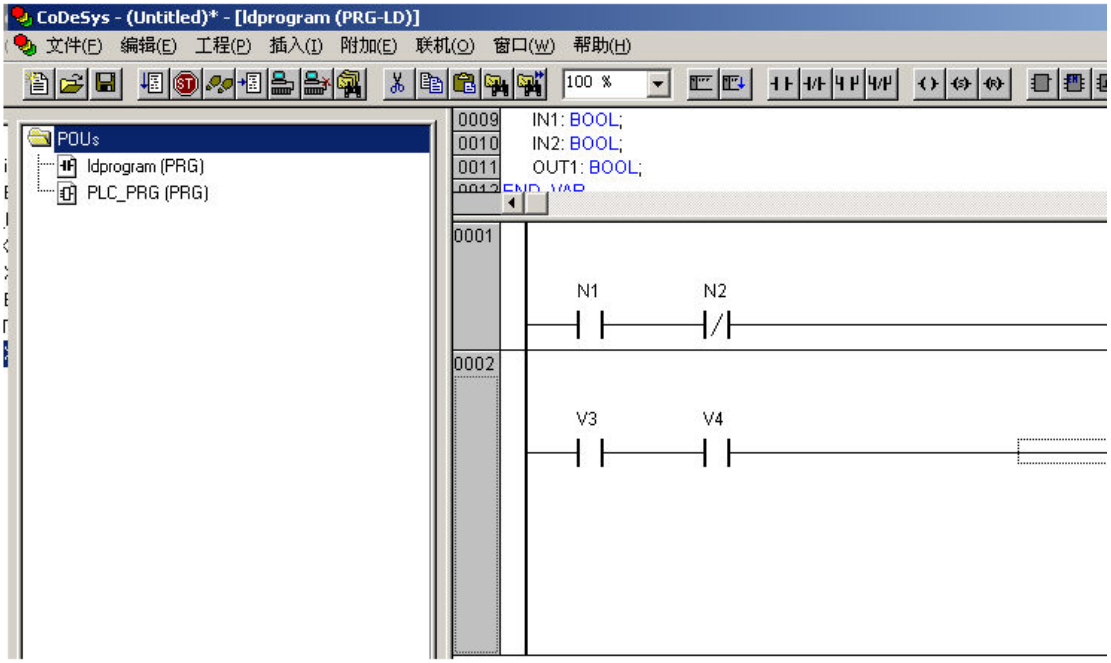
如果使用了命令“’联机’’写入新值’，新值将变为红色并保持不变。所有的变量将放入选择列表中，然后重新显示为黑色。

用命令 ’联机’’流程控制’来开始流向控制，你可以查看当前网络上正在传递的值。如果连接线上没有传递布尔值，值将显示在一个特殊的插入区域。如果线上传递的是布尔型的值，当它们传递TRUE值，以暗蓝色显示。通过这个方法，你可以在PLC运行时知道流程的信息。

如果鼠标在变量上停留一会，变量的类型、地址和注释将显示在工具提示中。

5.5.2 梯形图

下面是在 CoDeSys 编辑器 LD 中编写 POU 的窗口



POU 的所有编辑器包括声明部分和主体部分，它们被屏幕分割器隔开。

梯形图编辑器是一个图形化的编辑器，在内容菜单（按鼠标右键或<Ctrl>+<F10>）中包含有最重要的命令。

关于元素的信息，参照 2.2.6 章节， 梯形图。

参照:

LD编辑器的当前位置

移动原理或托拽名称

’插入’ ’连接’

’插入’ ’并联’

’插入’ ’LD的功能块’

’插入’’线圈’

以EN输入的POUs

’插入’ ’ LD中EN的框’

’插入’ ’在LD中插入块’

’插入’ ’跳转’

’插入’ ’返回’

’扩展’ ’向后粘贴’

’扩展’ ’向下粘贴’

’扩展’ ’向上粘贴’

’扩展’ ’否定’

’扩展’ ’设置/复位’

’扩展’ ″快速’

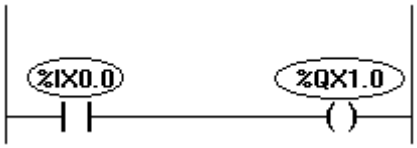
’扩展’ ’打开实例’

联机模式下的梯形图

LD编辑器的当前位置

下面的位置是可能的光标位置，在功能模块和程序访问中可以作为触点的位置、具有EN输入的POU和和在功能模块图一样处理的其它的POU 。编辑这个网络部分可以在 5.4.2 章节找到详细的信息，FBD编辑器。

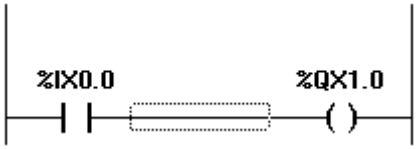
1. 每个文本区域（在黑色框中是光标位置）



2. 触点或功能模块

3. 线圈

4. 触点和线圈之间的连接线

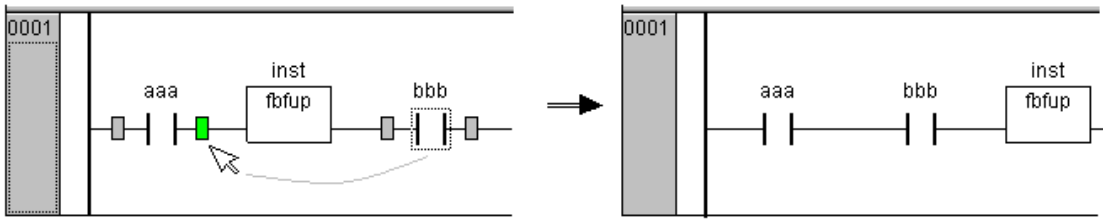


梯形图以特定方式使用下面菜单命令：

元件的移动

在 LD 的 POU 中可以通过拖放来移动元素到一个不同的位置。

为了移动元素，选择期望的元素（触点、线圈、功能模块）并按住鼠标键拖动它离开当前的位置，在 POU 的所有将要移动到的所有可能位置的网络内的元素，将会通过灰色填充的矩形来指示，移动元素到这些位置中的一个并释放鼠标键，元素将插入到新位置。



’插入’ ’网络(前)’

符号：

这个命令在梯形图中插入一个 网络。如果已经存在网络，新插入的将在当前网络前面。

’插入’ ’网络（后)’

符号：

这个命令在梯形图中插入一个 网络。如果已经存在网络，新插入的将在当前网络后面。

’插入’ ’常开接点’，注释

符号： 快捷方式：<Ctrl>+<0>

在LD编辑器中使用这个命令来在网络图中标记的位置插入一个 接点。

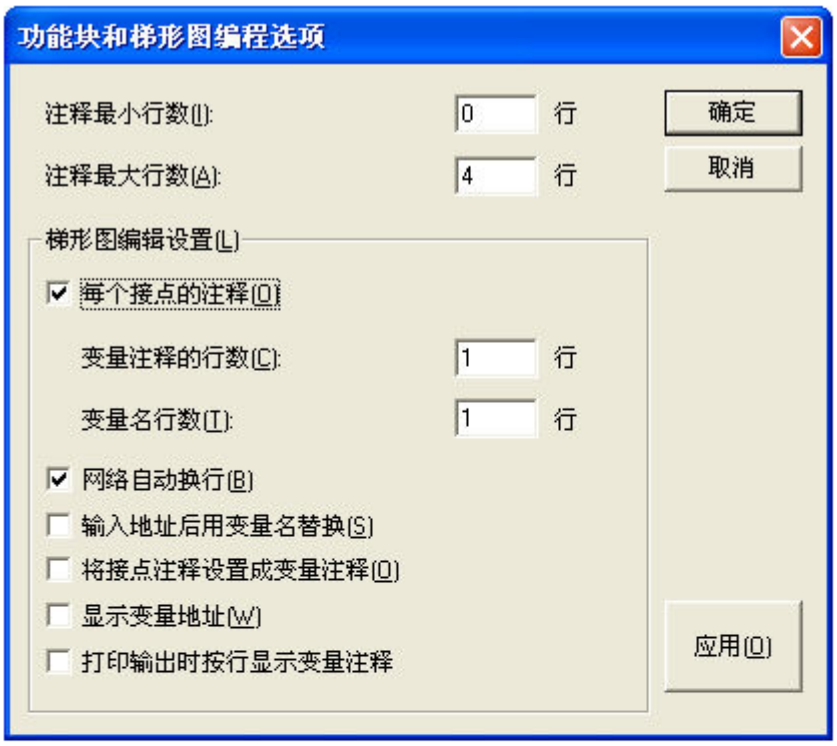
如果 标记的位置是一个 线圈或在触点和线圈之间连接线，新触点将会连续地连接到先前的触点。

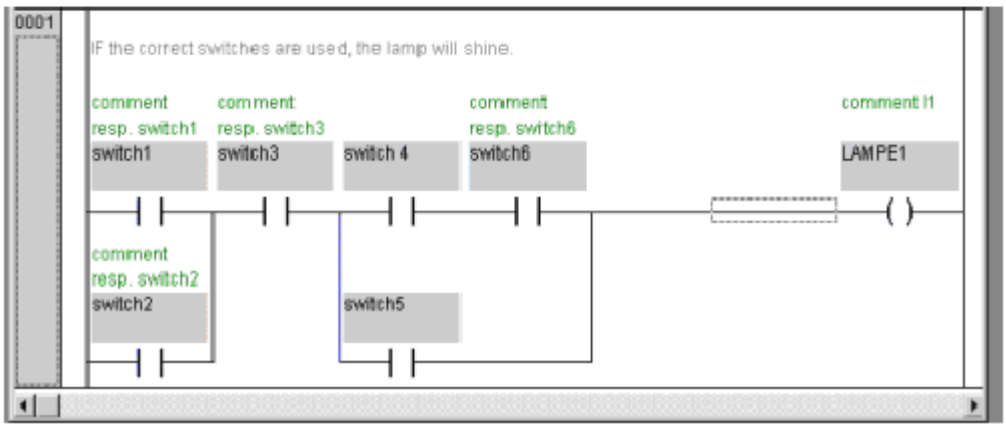
触点预设文本为“？？？”，你可以在这个文本上单击并改变它为期望的变量或期望的常量，你也可以

使用 输入助手，也能激活对话 ’ 功能块和梯形图选项’ (’ 附加’ ’ 选项’) 中的选项注释每个连接并在变量注释下画线来为变量名保留一定数量的行，如果使用长变量名，这可能对保持网络简略是有用的。

可以在梯形图选项中激活选项’ 网络自动换行’ 。

下面是在 FBD 或梯形图网络中的选项对话和结果显示的例子：





’插入’ ’常闭接点’

符号： 快捷键： 〈Ctrl〉 + 〈G〉

这个命令插入一个 常开接点。像 ’ 插入’ ’ 接点’ 和 ’ 附加’ ’ 取反’ 一样，也可插入一个常闭接点

’插入’ ’常开并联接点’

符号 ： 快捷方式 ：〈Ctrl〉+〈R〉

在LD编辑器中使用这个命令来在网络中标记位置处插入一个平行的 接点。

如果 标记的地方是一个 线圈或接头和线圈之间的连接，新的接头将与整个先前接头连接平行进行连接。

接点预设为文本“？ ？ ？ ”，可以单击文本并改变它为期望的变量或期望的常量，你也可以使用输入助手来做这些。

’插入’ ’常闭并联接点’

符号:  快捷键: 〈Ctrl〉 + 〈0〉

这个命令插入一个 常闭并联接点。像 ’ 插入’ ’ 常开并联接点’ 和 ’ 附加’ ’ 取反’ 一样，也可插入一个常闭并联接点

’插入’’线圈’

符号 ：  快捷方式 ：〈Ctrl〉+〈L〉

在编辑器LD中使用这个命令来插入一个与先前线圈平行的 线圈。

如果 标记的位置是线圈和 接点之间的连接，新线圈将插入到最后，如果标记的位置是线圈，新线圈将直接插入到它的上面。

线圈的默认文本是“？？？”。可以在文本上单击并改变它为期望的变量。也可以使用输入帮助来做这些。

也可以为线圈添加一个单独的注释，详细的讲述参照“插入” “接点”。

’插入’ ’置位线圈’

符号:  关键字: 〈Ctrl〉 + 〈I〉

这个命令插入一个 置位线圈。像 ’插入’ ’线圈’和 ’ 附加’ ’置位/复位’一样，也可插入一个置位线圈.

’插入’ ’复位线圈’

符号: 

这个命令插入一个 复位线圈。像 ’插入’ ’复位’和 ’ 附加’ ’设置/复位’一样，也可插入一个复位线圈.

’插入’ ’功能块’

快捷方式 ：  〈Ctrl〉+〈B〉

用这个命令来插入一个运算符、功能模块、功能或作为 POU 的程序。在接点和线圈之间的连接，或触头与触头之间的连接，必须标记出来。新 POU 在起先有名称 AND，可以改变它为其它的名称，也能使用输入帮助来做这些。标准的和自定义的 POU 都是可用的。

POU 的第一个输入放置在输入连接上，第一个输出在输出连接上，这些变量必须定义为 BOOL 类型。POU 的所有其它的输入和输出被文本“？？？”填充，这些优先的条目能改变为其它的常量、变量或地址。

也可以使用输入助手来做这些。

带EN输入的POUs

如果你想使用你的LD网络作为一个PLC来调用其它的POU，必须结合一个带EN输入的POU，这样的POU与 线圈平行连接，在这个POU之后你可以展开网络，就象在功能模块图中那样。在菜单项目“插入”“插入块”下，在EN POU的地方能找到插入命令。

操作符、功能模块、程序或带 EN 输入的功能与在功能模块图中的相应 POU 以相同的方式执行操作，除了它们的执行在 EN 输入上控制。这个输入附加到线圈和触点之间连接的地方，如果这个连接携带信息″On″，将会计算 POU 的值。

如果创建的 POU 带有 EN 输入，这个 POU 可以用来创建网络。这意味着来自通常操作符、功能和功能模块的数据能流进 EN POU 并且 EN POU 也能传送数据到这样的通用的 POU 中。

如果，你想在LD编辑器中编写一个网络，就象在 FBD中那样，你只需在新网络中首先插入一个EN操作符。随后，从这个POU开始，能象在FBD编辑器中那样继续从你的网络创建。

’插入’ ’ LD中EN的框’

符号： 

用这个命令可以把一个功能模块、操作符、功能或带 EN 输入的程序插入到 LD 网络中。

标记的位置是触点和线圈（光标位置 4）或线圈（光标位置 3）之间的连接。新 POU 平行插入到在线圈下面，它包含最初的名称“AND”。你也可以改变这个名称为其它的名字，也可以使用输入帮助来做这些。

’插入’’在LD中插入块’

使用这个命令你可以插入附加的元素到已经插入元素的 POU（也包括带 EN 输入的 POU）中，在这个菜单项目下的命令就象在功能模块图中相应命令一样可以在同一光标位置执行。

用输入可以添加一个新的输入到 POU 中。

用输出可以添加一个新输出到 POU 中。

用 POU，插入一个新 POU，过程与“插入“” POU“讲述的一样。

用分配你能给变量插入一个值。首先，显示为三个问号“？？？”，可以用期望的变量来编辑和替换，也可以使用输入帮助来做这些。

’插入’’上升沿触发器’

符号： 

这个命令时插入一个在引入信号中检测上升沿(FALSE -> TRUE)的 R_TRIG功能块。像 ’插入’’功能块’一样，它也能用于插入任何一个可用功能块。

插入\下降沿检测触发器

符号： 

这个命令时插入一个在引入信号中检测下降沿(FALSE -> TRUE)的 F_TRIG功能块。像’插入’’功能块’ 一样，它也能用于插入任何一个可用功能块。

’插入’’计时器 (TON)’

符号： 

这个命令是插入一个 TON计时器功能块。它负责接通延迟（延迟引入信号的通过），像 ’插入’’功能块’一样，它也能被用作插入TON模块。

’插入’’跳转’

用这个命令能插入一个平行的跳转在选中的LD编辑器中，在平行分支中，在先前 线圈的末端。如果引入行传递值"On"，跳转将跳转到指示标签处执行，

标记的位置必须是在触点和线圈之间的连接。

跳转用文本“？？？”表示，可以单击这个文本并改为期望的标签。

’插入’’返回’

在 LD 编辑器中，可以使用命令来在先前线圈的末端的平行部分插入一个返回指令。如果输入线上传递值“On”，在这个网络中 POU 的处理过程中断。

标记的位置必须是在触点和线圈之间的连接部分。

’附加’’粘贴在后面’

在编辑器中使用这个命令来粘贴剪贴板中的内容作为在标记的位置下面的连续接点，这个命令只有在剪贴板中的内容和标记的位置是接点的网络组成时才起作用。

’附加’’粘贴在下面’

快捷方式 ： <Ctrl>+<U>

在 LD 编辑器中使用这个命令来插入剪贴板的内容作为在标记位置之下的行接点，这个命令只有在剪贴板的内容和标记位置是接点的网络组成时才有效。

'附加''粘贴在上面'

在编辑器中使用这个命令来插入剪贴板的内容作为标记位置下面的平行接点，这个命令只有在剪贴板的内容和标记的位置是接点的网络组成时才有效。

'附加''取反'

符号 ：  快捷方式 ：<Strg>+<N>

使用这个命令来对一个触点、线圈、跳转或返回指令，或在当前 光标位置的EN POU的输入或输出进行否定操作。

在 线圈的圆括号或 接点的直线之间，出现一个斜线((/) or |/|)。如果这些是跳转，返回指令，或 EN POUs 的输入或输出，在连接处出现一个小圆圈，和在 FBD编辑器中的一样。

线圈现在在各自的布尔型变量中写入输入连接的否定值，如果各自的布尔型变量传递值 FALSE，否定触头切换输入的状态为输出。

如果标记了一个跳转或一个返回，这个跳转的输入或返回将被否定。

可以对否定进行否定来取消否定。

'附加''置位/复位'

符合： 

如果在 线圈上执行这个命令，会接收一个 设定线圈。这个线圈在各自布尔变量从不覆盖值TRUE。这意味着一旦为变量设置了值TRUE，它将一直保持TRUE。设置线圈在线圈符号中用“S”表示。

如果你再次执行这个命令，会得到一个复位线圈。这个线圈在各自的布尔变量中从不覆盖值 FALSE。这就是说，一旦为这个变量设置了值 FALSE，它将一直保持 FALSE，复位线圈在线圈符号中用"R"表示。

如果你重复执行这个命令，线圈将在设置和复位和常态之间改变。

联机模式下的梯形图

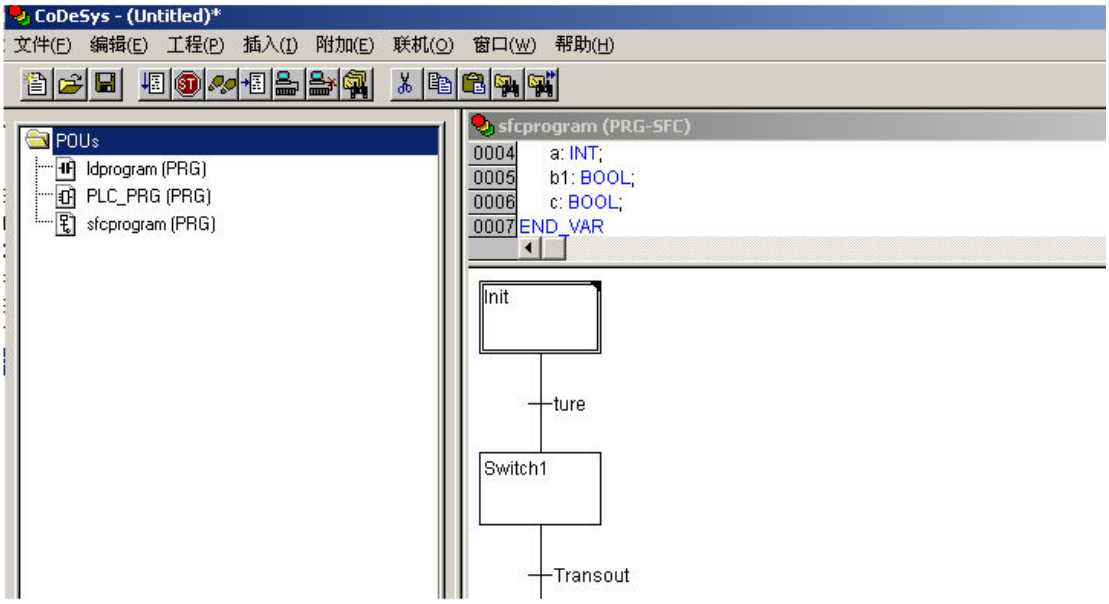
在联机模式下，在梯形图中的在“On”状态的触点和线圈以蓝色表示，同样地，所有传递“On”的线也涂上蓝色，在 功能模块的输入和输出处，相应变量的值显示出来。

断点只能在网络上设置，通过使用步，可以从网络跳到另一个网络。

如果你把鼠标在变量上停留一会。变量的类型、地址、注释将在工具提示中显示出来。

5.5.3 顺序功能图表编辑器

下面是在 CoDeSys 中相应的编辑器 SFC 中编写 POU 的窗口：



POU 的所有编辑器包含一个声明部分和主体部分，它们被屏幕分割器隔开。

顺序功能图表编辑器是一个图形化的编辑器，在内容菜单（按鼠标右键或<Ctrl><F10>）中能找到最重要的命令。工具提示显示缩放模式中全名或步的表达式、转换、跳转、跳转标签、标识符或相关的动作。离线模式和联机模式的显示一样。

详细信息参照 2.2.3 章节 ’顺序功能图’。

顺序功能图表编辑器必须与 SFC 的特殊编辑器相一致，下面的菜单项目将会常用到。

参照:

在SFC中标记块

’插入’ ’转换的步（前)’

’插入’ ’转换的步（后)’

’插入’ ’选择分支(右)’

’插入’ ’选择分支(左)’

’插入’ ″平行分支(右)″

’插入’ ’并行分支(左)’

’插入’ ’J跳转’

’插入’ ’转换跳转’

’插入’ ’添加进入动作’

’插入’ ’添加退出动作’

’扩展’ ’粘贴并行分支(右)’

’扩展’ ’增加并行分支标志’

删除标志

’扩展’ ’向后粘贴’

’扩展’ ’快速移动/转换’

’扩展’ ″清除动作/转换″

’扩展’ ’步的特性’

’扩展’ ’时间总览’

’扩展’ ’选项’

’扩展’ ’关联动作’

’扩展’ ’使用IEC-Steps’

联机模式下的顺序功能图

在SFC中标记块

标记的块是用虚线框包围的一组 SFC 元素。

通过移动鼠标到元素上并按鼠标左键，或使用方向键，可以选择一个元素（步、转换或跳转）。为了标记一组中多个元素，在标记的块上按<Shift>，并选择组的左下角或右下角的元素，这样就选中了包括这些元素的最小的组。

请注意，步只能和它之前或随后的转换一起 删除！

'插入''步-转换（前)'

符号 ：  快捷方式 ：<Ctrl>+<T>

这个命令在 SFC 编辑器中的标记块的附近插入一个后跟转换的步。

'插入''步-转换(后)'

符号 ：  快捷方式 ：<Ctrl>+<E>

这个命令用来在 SFC 编辑器中的标记块附近第一个转换后的转换后插入一个

删除步和转换条件

步只能和前面或随后的转换一起被删除，移动选择框到步和转换周围并选择命令“编辑”“删除”或按<Del>键。

'插入''选择分支(右)'

符号 ：  快捷方式 ：<Ctrl>+<A>

这个命令插入一个可选分支在 SFC 编辑器中作为标记块的右分支。

标记块必须开始和结束于一个变换，新的分支组成一个转换。

'插入''选择分支(左)'

符号 ： 

这个命令插入一个可选分支在 SFC 编辑器中作为标记块的左分支。

这个标记块必须开始和结束于转换，新分支组成一个转换。

'插入'"平行分支(右)"

符号 ：  快捷方式 ：<Ctrl>+<L>

这个命令在 SFC 编辑器中插入一个平行分支作为标记块的右分支。

标记块必须开始和结束于步，新分支作成一个步。要使已创建的平行分支允许跳转，必须提供一个跳转标签。

'插入''并行分支(左)'

符号 ： 

这个命令在 SFC 编辑器中插入一个平行分支作为标记块的左分支。

标记块必须开始和结束于步，新分支作成一个步，要使已创建的平行分支允许跳转，必须提供一个跳转标签。参照'附加''增加并行分支标志'

'插入''跳转'

在 SFC 编辑器中用这个命令在标记块所属的分支的末端插入一个跳转，这个分支必须是可选分支。

在插入的跳转中文本字符串“Step”可以被选中和被步名字或要跳转到平行分支的跳转标签替代。

’插入’ ’转换跳转’

符号 ： 

在 SFC 编辑器中使用这个命令来插入一个转换，在选择分支的末端后跟一个跳转，这个分支必须是选择分支。

在插入的跳转中插入的文本字符串“Step”能够被选中并且能被步的名字或要跳转到的平行分支的跳转标签替代。

’插入’ ’添加进入动作’

使用这个命令可以为步添加一个 进入动作。在步 激活之后，进入动作只执行一次，进入动作可以以你选择的语言执行。

带进入动作的步在左下角用"E"表示。

’插入’ ’增加退出动作’

用这个命令可以为 步添加一个 退出动作，在步失效之前，退出动作只执行一次，退出动作可以以选择的语言执行。

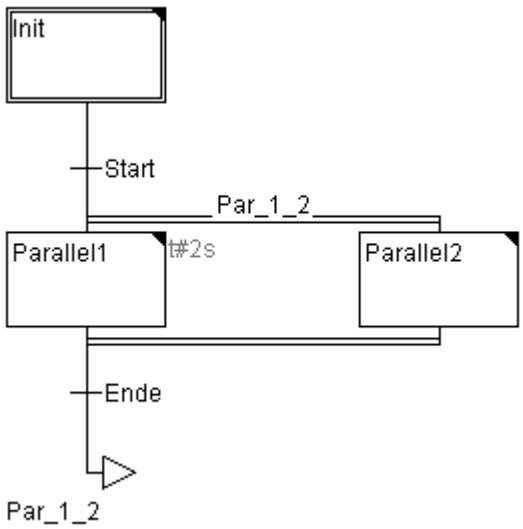
带退出动作的步在右下角用"X"表示。

’附加’ ’粘贴并行分支(右)’

这个命令粘贴剪贴板的内容作为标记块的右 分支，标记块必须开始和结束于一个步，剪贴板的内容同样也必须是开始和结束于步的SFC块。

’附加’ ’增加并行分支标志’

为了提供一个新的带跳转标签的插入平行分支，跳转发生在平行分支必须被标记并且命令“增加并行分支标志”被执行。在那个点，平行分支将会给定一个标准名字包含“并行”和一个附加序列号，它可以按照标识符名字的规则来编辑。



删除标志

通过删除标签名可以删除标签。

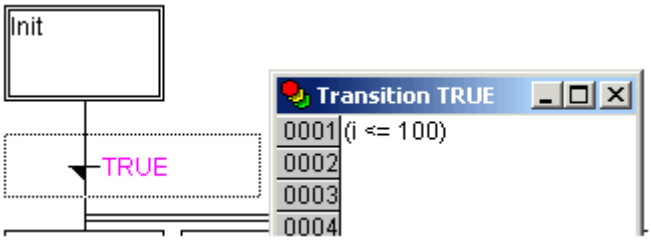
’附加’ ’粘贴在后面’

这个命令粘贴剪贴板上在第一个步之后或标记块的第一个转换后 SFC 块，（正常在标记块附近复制粘贴它。）根据语言标准，如果 SFC 结构是正确的，这个将开始执行。

’附加’’添加移动/转换’

快捷方式 ：<Alt>+<Enter>

标记块的第一个步的 动作或标记块的第一个转换的转换主体被加载到各自书写它们语言的编辑器中，如果动作或转换主体是空的，那么书写它们的语言必须被选中。



’附加’″清除动作/转换″

使用这个命令你能删除标记块的第一个步的 动作或第一个转换的转换主体的动作。

如果在一个步中，只执行了动作、进入动作或只执行退出动作，那么相同的将会通过命令被删除，否则出现一个对话框，你可以选择那个或那些动作要删除。

如果光标位于IEC步的动作上，那么只有这个连接被删除，如果 IEC步和相关的 动作被选中，相关连接被删除，有多个动作的IEC步，将出现一个选择对话框。

’附加’’步的特性’

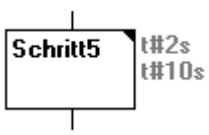
用这个命令能打开一个对话框，在对话框中能编辑标记步的属性。

可以利用在步属性对话框中的三个不同条目，在最小时间下，你可以输入这个步的处理过程占用的最小时间长度。在最大时间，你可以输入这个步的处理过程占用的最大时间长度。

条目是时间类型，因此应该使用时间常量（例如，T#3s）或时间类型的变量。

在注释下，你可以为步插入一个注释，使用命令“附加” “选项”打开“顺序功能图选项”对话，在SFC 编辑器中能决定显示步的注释还是步的时间设置。在右边，步的旁边，注释或时间设置将出现。

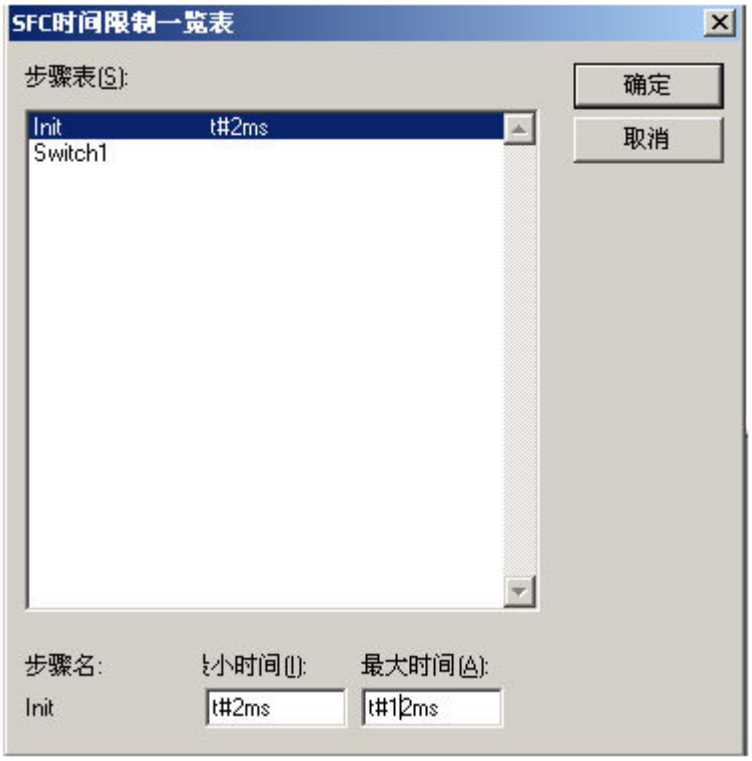
如果超过了最大时间，用户可以查询设置的 SFC标记。



例子中显示了一个在联机模式下至少执行 2 秒最多执行 10 秒的步，除了这来两个时间外，还显示了步已经激活了多长时间。

’附加’ ’时间总览’

用这个命令你能打开一个窗口，在这个窗口中能编辑 SFC 步的时间设置：



SFC POU的所有步都显示出来，如果已经为步 输入一个时间边界，时间界限显示在步的右边（首先是时间下限，然后是时间上限）。你可以编辑这些时间界限。在期望的步上单击，步的名字显示在窗口的下面，在最小时间或最大时间区域里输入期望的时间界限，如果按OK关闭对话，所有的改动将被保存。

在例子中，步 2 和 6 有时间边界，Shift1 持续最少 2 秒，最多 10 秒。Shift2 持续最少 7 秒，最多八秒。

’附加’ ’选项’

用这个命令打开一个对话框，在对话框中能为 SFC POU 设置不同的选项。

在SFC选项对话框中能设置五个条目，在步高下，输入在SFC中一个SFC步占多少个行高，在这里是标准设置是 4。在步宽下，可以输入一个步占多少个列，标准设置是 6。也可以预设显示步。可以有三种选择的可能性：你可以选择不显示，或显示注释，或显示时间限制。最后两个条目与在 ’扩展’ ’不的特性’显示的方式相同。

’附加’’关联动作’

用这个命令 动作和布尔变量可以连接到 IEC步。

在IEC步右边连接一个附加的分开框，对于动作的连接，在左边的区域预设有限定词“N”并且在右边区域有“Action”的名字，可以通过 输入助手来改变它们。

一个 IEC 步最多能连接 9 个步。

使用命令“工程”“添加动作”在对象管理器中创建 IEC 步的新动作，

’附加’’使用IEC步’

符号 ： 

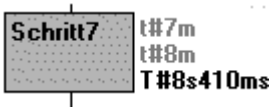
如果激活这个命令（在菜单项目的附近有一个对号表示），IEC步将替代简单的步插入到步 转换和 平行分支上。

如果这个选项打开，当你创建一个新 SFC POU 时初始化步设置为 IEC 步。

这个选项保存在文件“CoDeSys.ini”中并且当 CoDeSys 重新启动时恢复。

联机模式的顺序功能图表

使用联机模式下的顺序功能图编辑器，当前 激活的步显示为蓝色。如果在命令下 ’扩展’ ’选项’已经设置了它，那么步右边描述时间管理。在下和上边界已经设置的第三个时间指示器将出现，从这里能读出步已经激活了多长时间。



在上面的图片中描述的步已经激活了 8 秒和 410 毫秒，步在退出之前至少还要激活 7 分钟。

用命令“联机”“托拽断点”，将会在步上设置一个断点，或在一个动作的位置处用使用的语言来设置，处理过程在这个步之前停止或在程序中动作位置之前停止，断点设置所在步或程序的位置标记为亮蓝色。

如果在平行分支的数个步都激活，激活的步显示为红色，它的动作将处理下一个。

如果已经使用了 IEC步，在联机模式所有激活动作将显示为蓝色。

使用命令“联机”“跳过”，它将跳到动作正在执行的下个步中。

如果当前位置是：

•在 POU 的线性处理中的步或在 POU 的最右边分支中的步，从 SFC POU 的执行将返回到调用处，如果 POU 是主程序，下个循环开始。

•在平行分支中除了最右边的步，执行跳转到下一个平行分支的活动步。

•最后的断点位于 3S 动作之内，执行跳转到 SFC 的调用处。

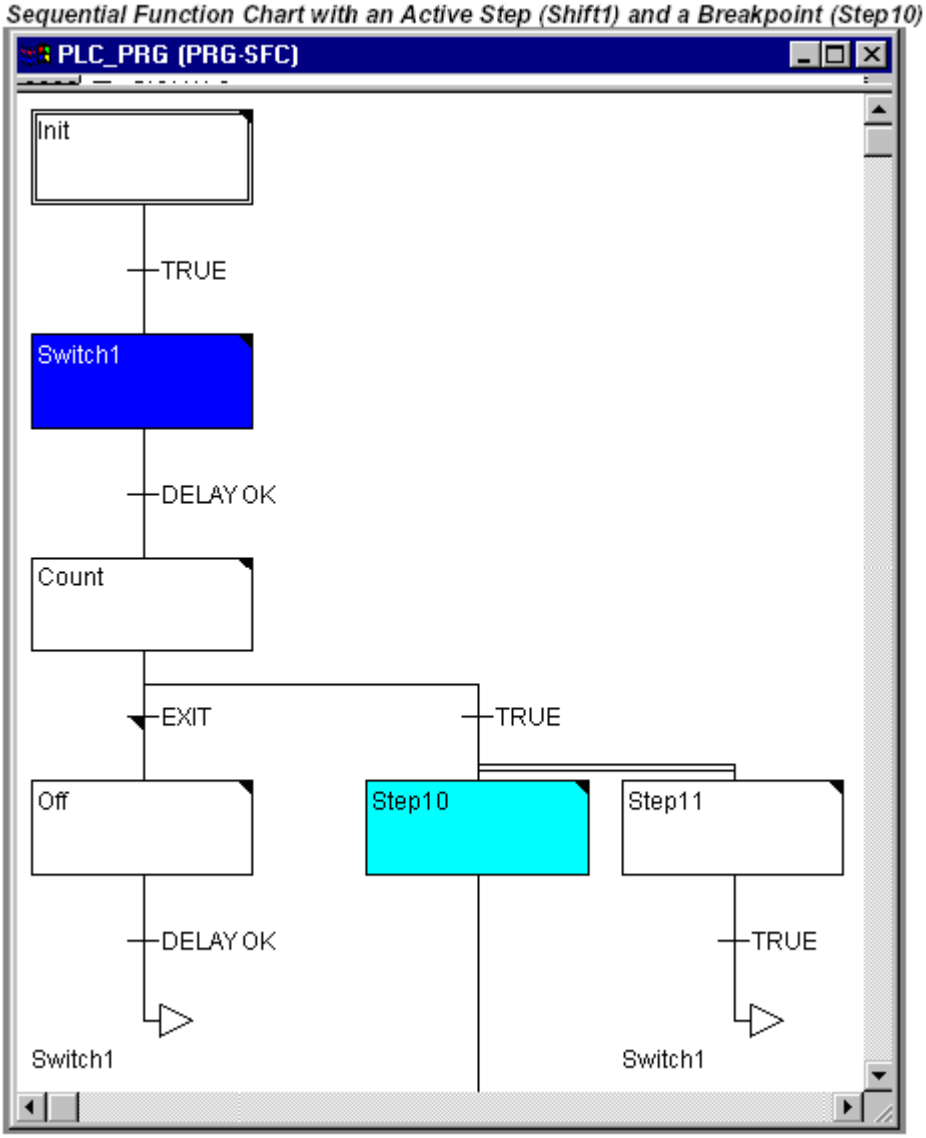
•最后的断点位于 IEC 动作内，执行跳转到 SFC 的调用处

•最后的断点位于输入或输出动作之内，执行跳转到下一个活动的步中。

使用“联机”“进入”，包括能进入动作中，如果输入、输出或 IEC 动作要被跳入，必须在这里设置一个断点。在动作内，相应编辑器的所有调试功能对用户都可用。

如果光标在声明编辑器中的变量上停留一会，变量的类型、地址和变量的注释将显示在工具提示中。

Sequential Function Chart with an Active Step (Shift1) and a Breakpoint (Step10)



请注意：如果你重命名了一个步并在步是活动的时候执行了一个联机改变，那么程序将会停止在未定义状态

在序列中的元素的处理顺序

1. 首先，在序列中使用的 IEC 动作中的所有动作控制块标示符将复位（然而，IEC 动作的标示符在动作内部调用）

2. 所有步按顺序测试（从上到下和从左到右）

3. 对于所有的步，以下部分按序列中设定的顺序来完成：

-如果可用，逝去的时间复制到相应的步的变量中。

-如果可用，任何超时都被检测并且 SFC 错误标示符按要求提供。

对非 IEC 步，相应动作开始执行。

4. 在序列中用到的 IEC 动作按字母顺序执行，通过动作列表有两种途径来完成这个动作，第一种是，所有的在当前循环中未激活的 IEC 动作开始执行。第二种途径是，在当前循环中所有激活的 IEC 动作执行。

5. 转变开始进行：如果在当前循环中的步是激活的并且下面的转变条件返回 TRUE（如果最小激活时间已经消逝），下面的步是激活的。

关于动作的执行部分应该注意以下部分：

动作可以在一个循环中执行多次因为它关联了重复序列。（例如，一个 SFC 可以有两个 IEC 动作 A 和 B，

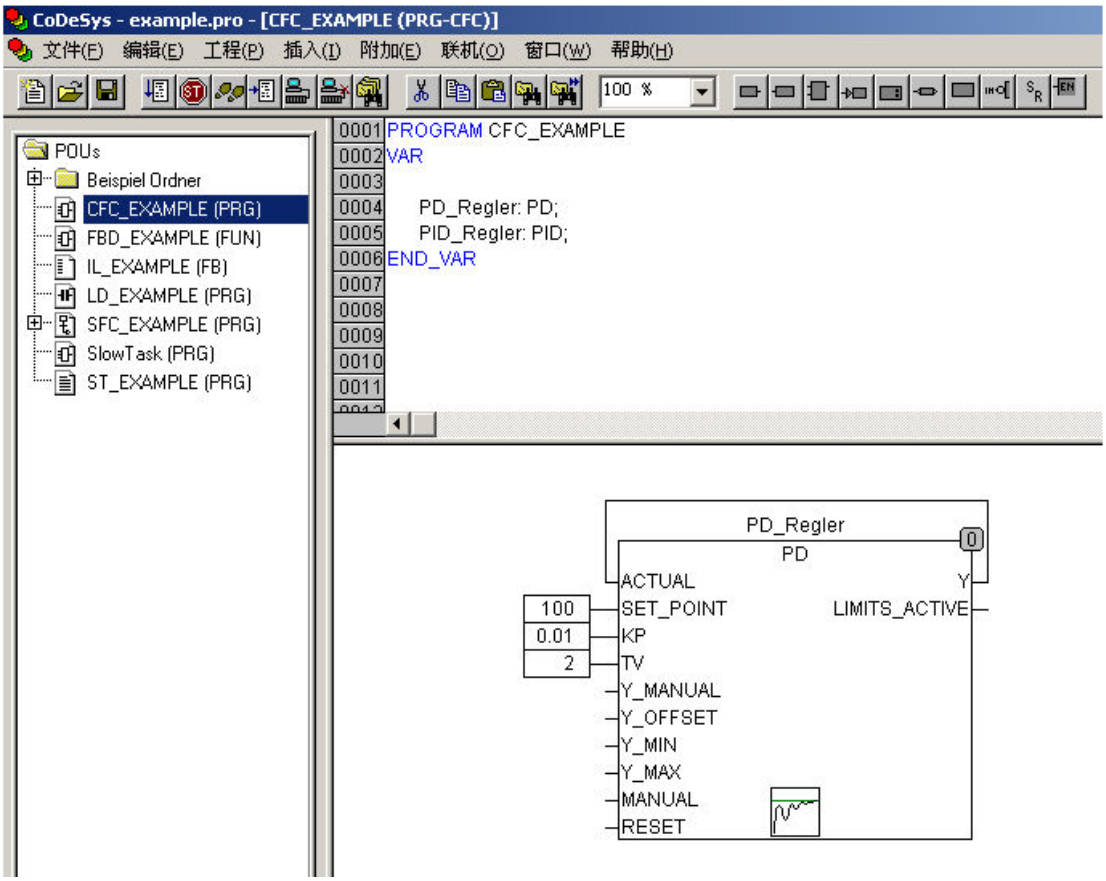
它们都在 SFC 中执行，并且都可以调用 IEC 动作 C，那么 IEC 动作 A 和 B 都可以在相同的循环中激活，而且在这两个动作中 IEC 动作 C 也能激活，C 将被调用两次）。

如果相同的 IEC 动作在 SFC 中不同的级别中同时使用，由于上述处理序列的原因将导致不可预料的结果。因而，在这种情况下将回出现出错信息。它也可能出现在其它旧版本的 CoDeSys 工程创建过程中。

注意：在转换条件中监视表达式（例如，A AND B）时，显示的是整个表达式的值

5.5.4 连续功能图表编辑器

它看起来象使用 连续功能图表编辑器产生的块：



连续功能图表编辑器中没有使用捕捉栅格，因此元素可以任意放置。连续处理列表的元素包括框、输入、输出、跳转、标签、返回和注释。这些元素的输入和输出可以通过用鼠标拖动连接来连接起来，连接线自动画出。最短的可能的连接线要考虑到现有的连接，当元素移动时连接线自动调整，如果连接线因为缺乏空间不能画出，在输入和相关的输出之间出现一个红线，这个红线只有当空间充足时才转化为连接线。

与通常的 功能模块图编辑器相反的连续功能图表的一个优点是反馈路径可以直接插入。

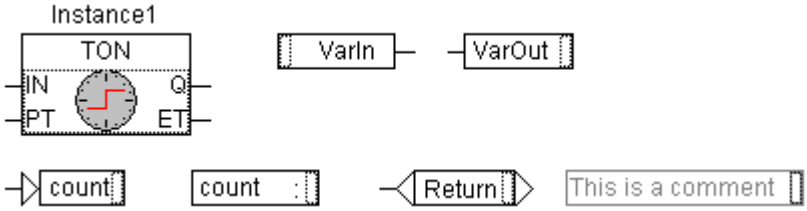
在内容菜单中可以找到最重要的命令。

CFC的当前位置

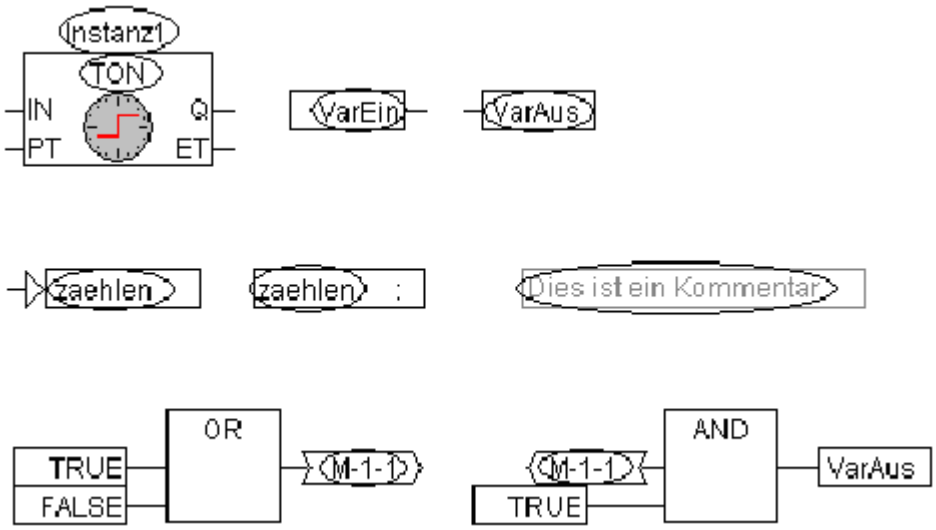
每个文本都是光标可能的位置，选中的文本渐变为蓝色并且可以被修改。

在其它的情况下当前鼠标的位置通过虚线矩形框来显示，下面是光标可能位置的例子：

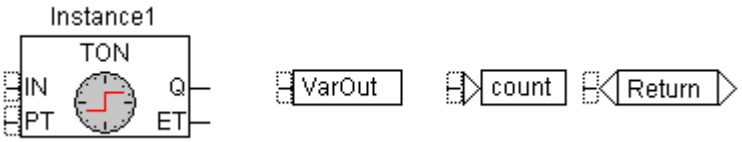
1. 框、输入、输出、跳转、标签、返回和注释这些元素的中继线。



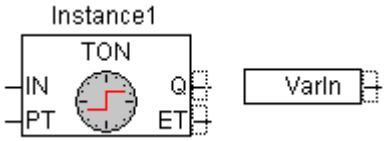
2. 与连接标记的文本区域一样框、输入、输出、跳转、标签、返回和注释这些元素的文本区域。



3. 框、输入、输出、跳转、标签、返回这些元素的输入端



4. 元素框和输入的输出端：



'插入' '框'

符号 ：  快捷方式 ：<Ctrl>+<B>

通常插入一个“AND”操作符，可以通过选中操作符并用其它其它的操作符、功能、功能模块和程序覆盖文本来修改它，输入帮助提供从支持块的列表中选出期望的块的帮助。如果新块，有输入的其它的最小数字，这些将附加上。如果新块有输入的较小最高号码，最后的输入将被删除。

'插入' '输入'

符号 ：  快捷方式 ：<Ctrl> + <E>

这个命令用来插入一个输入，出现的文本“???”可以被选中并被变量或常量替代，输入帮助也可以在这里使用。

'插入' '输出'

符号 ：  快捷方式 ：<Ctrl>+<A>

这个命令用来插入一个输出，可以选中出现的文本"???"并且可以用变量替代，在这里也可以使用输入帮助，与这个输出的输入相关的值分配给这个变量。

'插入''跳转'

符号 ：  快捷方式 ：<Ctrl>+<J>

这个命令用来插入一个跳转，可以选中出现的文本“???”并用程序要跳转的跳转标签来替代它。

使用命令“插入”“标志”插入跳转标签。

'插入''标签'

符号 ：  快捷方式 ：<Ctrl>+<L>

这个命令用来插入一个标签，可以选中出现的文本“???”并用跳转标签来替代它，在联机模式下自动插入一个标记 POU 的末端的跳转标签。

用命令“插入”“跳转”可以插入跳转。

'插入''返回'

符号 ：  快捷方式 ：<Ctrl> + <R>

这个命令插入一个返回命令，注意在联机模式下带 RETURN 名字的跳转标签自动插入到第一列和编辑器最后元素的后面，在分支中，它自动跳转到执行将离开 POU 之前的地方。

'插入''注释'

符号 ：  快捷方式 ：<Ctrl> + <K>

这个命令用来插入注释。

用<Ctrl> + <Enter>在注释内获得一个新行。

'插入''输入框'

快捷方式 ：<Ctrl> + <U>

这个命令用来在方框处插入一个输入。不同的操作符有不同的输入端数（例如，ADD 可以有两个或多个输入端）。

必须选中方框才能为这个操作符增加输入端的数目，每次添加一个输入端。

' 插入''输入针' '插入''输出针'

符号 ： 

为了编辑而打开宏时，这些命令是可用的，它们用来插入宏的输入或输出或输入和输出。通过它们的显示和有没有位置索引来区分 POU 的输入和输出端。

'附加''取反'

符号 ：  快捷方式 ：<Ctrl> + <N>

这个命令用来对输入、输出、跳转、返回命令取反，在连接上用一个十子交叉表示否定的符号。

元素块、输出、跳转或返回当它们被选中时，它们的输入否定。

元素块的输出或输入当它被选中时被否定（光标位置 4）。

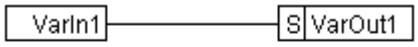
通过再次否定可以取消原来的否定。

’附加’’置位/复位’

符号 ：  快捷方式 ：〈Ctrl〉+〈T〉

这个命令只能用来选择输出元素的输入端。

设置的符号是 S，复位的符号是 R。



如果 VarIn1 传递 TRUE，VarOut1 设置为 TRUE，VarOut1 保持这个值，即使当 VarIn1 跳回到 FALSE。

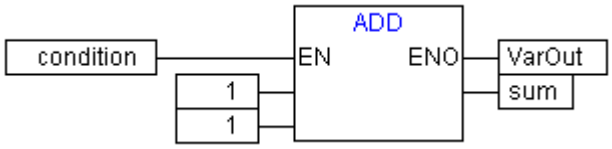
如果 VarIn2 传递 TRUE，VarOut2 设置为 FALSE。VarOut2 保持这个值，即使当 VarIn1 跳回到 FALSE。

重复激活这个命令可以使输出在设置、复位和正常状态之间转换。

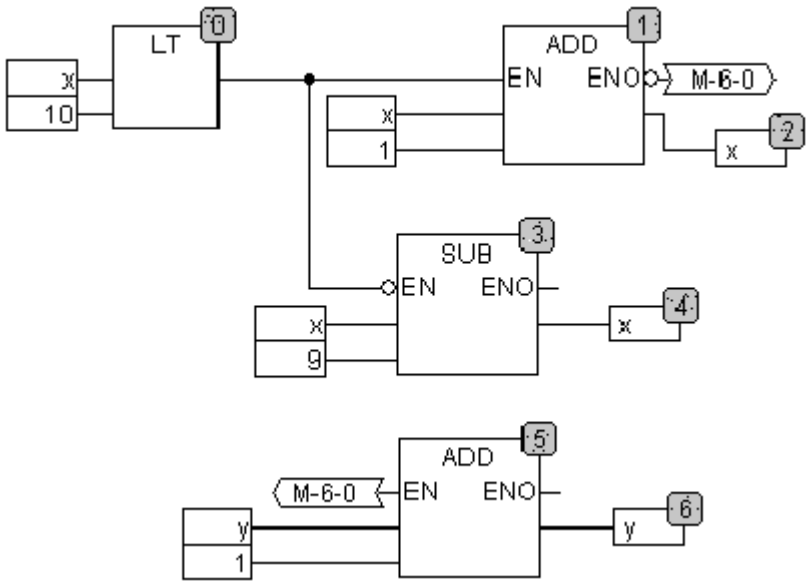
’附加’’EN/ENO’

符号 ：  快捷方式 ：〈Ctrl〉+〈0〉

这个命令用来给选中的块添加一个附加的布尔型使能输入 EN 端（使能输入端）和一个附加的布尔型使能输出 ENO（使能输出端）。



在这个例子中，当布尔变量“Bedingung”（状态）为 TRUE 时，ADD 才执行，在 ADD 执行后 VarOut 设置为 TRUE。当“Bedingung”（状态）为 FALSE 并且 VarOut 保持 FALSE 值时 ADD 将不会执行。下面这个例子显示了 ENO 值是怎样为随后的块工作的：



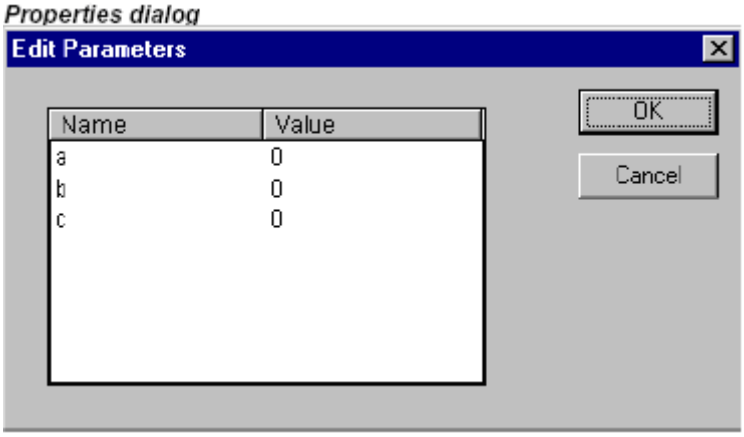
x 应该初始化为 1 和 y 初始化为 0，在块的右角上的数字表明了执行的顺序。

x 将会每次增加 1 直到值为 10，块的输出 LT（0）传递值 FALSE 并且 SUB（3）和 ADD（5）将会被执行，x 的值设置回 1y 的值每次增加 1，只要 x 的值比 10 小，LT（0）就重复执行，y 记录 x 在值 1 到 10 之间通过的次数。

’附加’’属性 ’

在连续功能编辑器中不显示 功能和 功能模块中的常量输入参数(VAR_INPUT CONSTANT)，当你选中块的主体并且使用命令“附加”“属性”或者在主干上双击时“特性编辑器”对话框打开，在这里显示和修改它们的值：

Properties dialog



常量输入参数(VAR_INPUT CONSTANT)的值是可以改变的，在这里你必须在列值中标记参数值。单击鼠标或按空格键可以编辑参数值，按<Enter>键确认或者按<Escape>键取消。按’确认’存储所有的改变。

选择元件

在元素的主体上单击来选中它。

为标记多个元素，按<Shift>键并一个接一个单击需要选择的元素，或者按鼠标左键并拖动鼠标到要标记的元素上。

命令“附加”“选择所有”一次能标记所有的元素。

移动原理

一个或多个选中的元素可以通过方向键来移动就如按住<Shift>键一样，也可以通过按住鼠标左键拖动来移动元素。通过释放鼠标左键放置这些元素直到它们不覆盖其它的元素或者超出编辑器的边界。否则，标记的元素跳回到它的初始位置并出现一个警告声音。

复制原理

一个或多个选中的元素可以使用命令“编辑”“复制”来复制并且可以用命令“编辑”“粘贴”来插入。

创建连接

元素的输入端可以精确的连接到另一个元素的输出端，一个元素的输出端可以连接到其它元素的输入端。

元素 E2 的输入端和元素 E1 的输出端的连接有多种可能性。

把鼠标置于元素 E1 的输出端，单击鼠标左键，按下鼠标左键并拖动鼠标指针到 E2 的输入端，然后释放鼠标键，在用鼠标拖动的过程中产生从元素 E1 的输出端到鼠标指针的连接，放置鼠标到元素 E2 的输入端，单击鼠标左键，按下鼠标左键并拖动鼠标指针到元素 E1 的输出端并释放鼠标左键。

创建连接

在元素 E1 的输出端和元素 E2 的输入端之间的连接可以很容易的改变为元素 E1 的输出端和元素 E3 的输入端之间的连接，在 E2 的输入端上单击，按下鼠标左键，移动鼠标指针到 E3 的输入端并释放。

产出连接

删除 E1 的输出端和元素 E2 的输入端之间的连接有多种方法：

选中元素 E1 的输出端并按<Delete>键或执行命令“编辑”“删除”，如果 E1 的输出端连接到了多个输入端，那么将会同时删除多个连接。

选中元素 E2 的输入端并按<Delete>键或执行命令“编辑”“删除”。

用鼠标选中元素 E2 的输入端，按下鼠标左键并拖动连接从 E2 的输入端离开，当鼠标左键在屏幕的自由区域中释放时，连接删除。

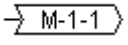
’附加’’连接标记’

连接也可以用一个连接器（连接标记）表示来替代连接线，在这里唯一的名字连接器添加到输出和相关的输入端。

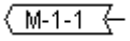
在两个元素之间已经存在的连接现在用连接器表示，连接线的输出端被标记并且菜单中的“扩展”“连接标记”被选中，下面图表显示一个连接在菜单点选择前后的连接情况。

程序给出一个唯一的标准化的名字，开始于 M，连接器的名字存储为一个输出变量，它可以被更改，既可以在输入端更改也可以在输出端更改。

连接器的名字是和连接的输出端的属性相关的并且和它一起保存。

1.在输出端编辑连接器： 

如果在连接器中的文本被替代，在输入端的所有相关的连接器都会采用新的连接器名字。一种情况不会采用，选择的名字已经属于另外的连接器，因为违背了连接器名字的唯一性。

2.在输入端编辑连接器： 

如果连接器中的文本被替换，在其它的 POU 中相应的连接器也会被替换。在标记连接的输出端（光标位置 4）并且选择菜单点“扩展” “连接标记”后，连接器表示中的连接可以转换为正常的连接。

改变连接

元素 E1 的输出和元素 E2 的输入之间的连接可以被方便的改为元素 E1 的输出和元素 E3 的输入。把鼠标选中 E2 的输入，然后按住左键移动鼠标到 E3 的输出位置松开左键。

删除连接

有很多方法来移除元素 E1 的输出和元素 E2 的输入：

选择元素 E1 的输出然后按 <Delete> 键或者执行命令’编辑’’删除’。如果 E1 输出连接多个输入，删除情况相同。

选择元素 E2 的输入然后按<Delete> 键或者执命令’编辑’’删除’。

用鼠标选择 E2 的输入，按住鼠标左键从 E2 处拖出，在空白处松开鼠标 E2 就被移除

插入 输入/输出 ″不工作状态″

如果选中一个元素的输入或输出端，相应的输入或输出可以直接插入，编辑器区域中充满了在键盘上输入的字符串。

执行次序

元素块、输出、跳转、返回和标签每个都具有一个编号表明了它们执行的顺序，在这个连续的顺序中，每个元素在运行时被计算。

当元素的编号根据拓扑顺序自动给出（从左到右和从上到下），如果顺序已经改变了，新元素将接收它的拓扑继承者的编号，并且所有较高的编号增加 1，

当它移动时元素的编号保持不变。

顺序影响结果，在一定情况下必须改变顺序。

如果显示了顺序，相应连续执行编号在元素的右上角显示。

参照:

’附加’ ’次序’ ’显示’

’附加’ ’次序’ ’次序拓扑’

’附加’ ’次序’ ’向上移动一格’

’附加’ ’次序’ ’向下移动一个’

’附加’ ’次序’ ’移动到首端’

’附加’ ’次序’ ’移动到末端’

’附加’ ’次序’ ’依照数据溢出的所有次序’

’附加’ ’显示排序’

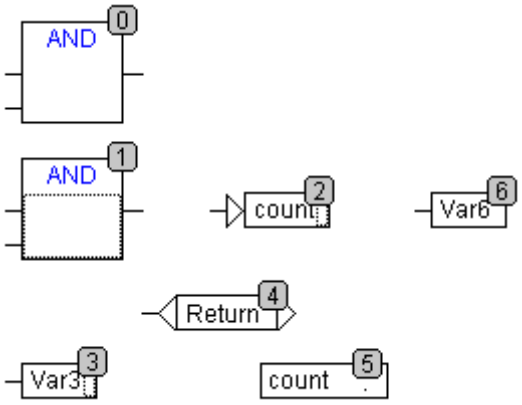
这个命令用来切换执行顺序显示的开和关闭。默认的设置是显示（在菜单点的附近用一个 ü 识别）。

在块、输出、跳转、返回和标签这些元素执行编号的相关顺序出现在元素的右上角。

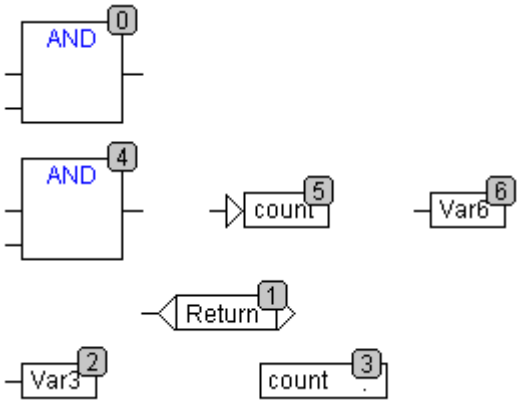
’附加’ ’排序’ ’拓扑排序’

当执行从左到右并且从上到下执行时，元素按拓扑顺序处理。编号从左到右增加并且元素从上到下拓扑安排。连接是不相关的，只是元素的位置是重要的。

当使用命令“附加”“排序”“拓扑排序”时，所有选中的元素以拓扑结构安排。在选择中的所有元素在这个处理过程中从连续处理列表中去掉。元素然后各自地从右下到左上输入到保留的连续处理过程列表中。每个标记的元素在它的拓扑继承者之前输入到列表中，例如，当编辑器中的所有元素都按拓扑顺序进行了排序，在拓扑顺序中在元素之前插入的将在元素之后执行。可以用下面这个例子阐明。



编号为 1、2、3 的元素被选中。如果命令“Order topologically”被选中元素将首先从连续处理列表中除去。Var3，跳转和 AND 操作符然后一个接一个插入，Var3 放置在标签之前并且接收编号 2，跳转然后被排序并首先接收编号 4 但是随后在 AND 插入之后变为 5，新的执行顺序出现：



当引入一个新产生的块时，它将在顺序处理列表中默认放置它的拓扑继承者的附近。

’附加’’排序’’向上移动一步’

除了在连续处理列表中开始处的元素外，用这个命令选中的所有元素将在处理列表中向前移动一个位置编号。

’附加’’排序’’向下移动一步’

除了在连续处理列表中末端处的元素外，用这个命令选中的所有元素将在处理列表中向后移动一个位置编号。

’附加’’排序’’移动到首前端’

使用这个命令所有选中的元素将会移动到连续处理列表的附近，在选中元素的组内顺序保持不变，未选择元素的组内的顺序也保持相同。

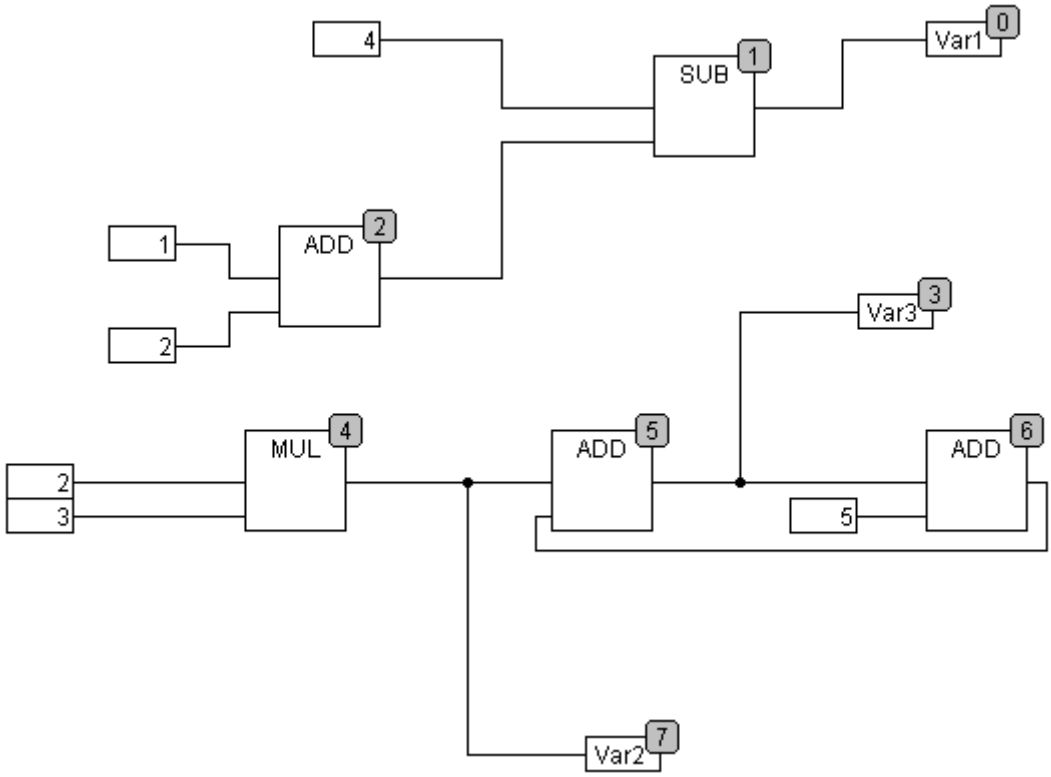
’附加’’排序’’移动到末端’

使用这个命令所有选中元素将会移动到连续处理列表的末端，选中元素的组内顺序保持不变。未选中元素的组内的顺序也保持不变。

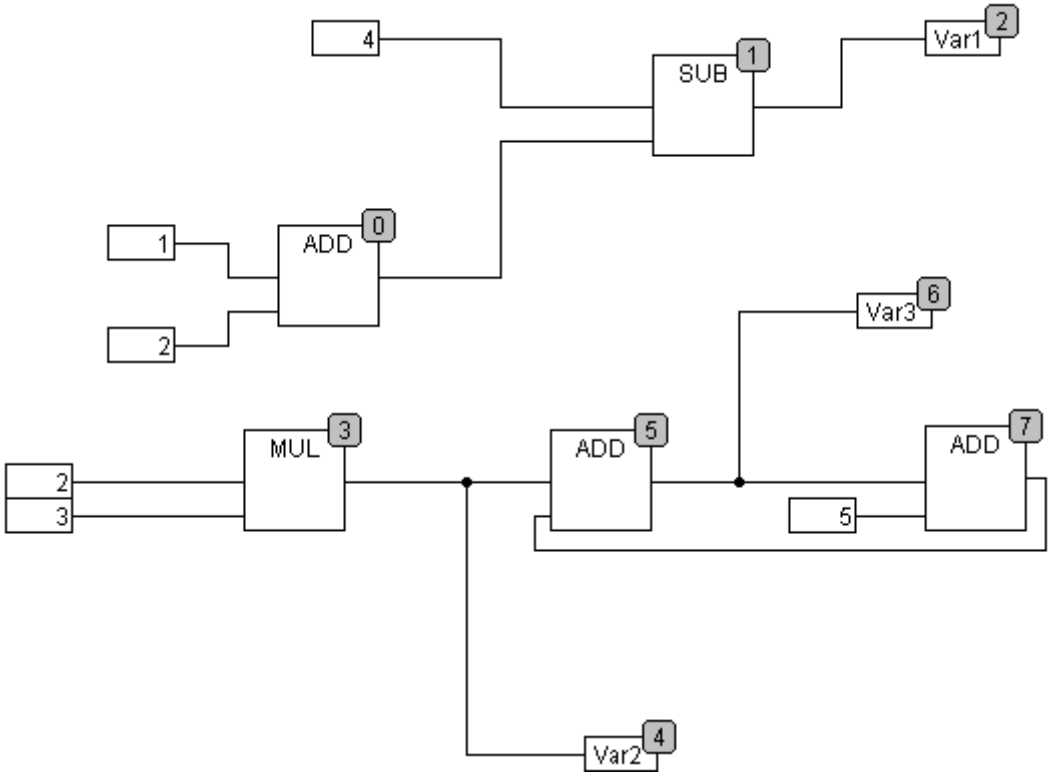
’附加’’排序’’依照数据流排序’

这个命令影响所有的元素。执行顺序由元素的数据流向决定而不是它们的位置决定。

下面的图表显示已经按拓扑结构排列的元素。



下面是命令选择后的元素排列：



当选中这个命令后，元素将首先按拓扑顺序排列。创建一个新的处理列表。基于已经知道的输入值，计算机将计算哪个未编号的元素下一步处理。在上面“network”块 AND，例如，当在它的输入端（1 和 2）的值都已经确定下迅速处理。在块 ADD 的结果处理后，块 SUB 才执行。

反馈路径最后插入。

’附加’ ’创建宏’

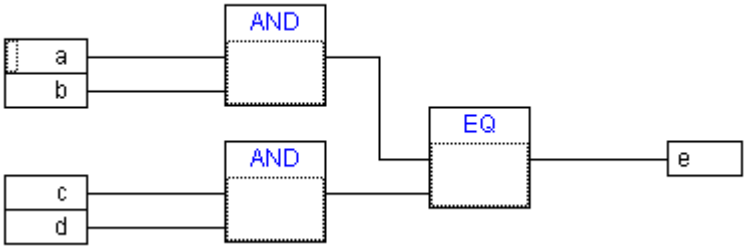
符号： 

用这个命令，同时选中的多个 POU 可以组合到一个块中，这个块可以称为一个宏。宏只能通过复制/粘贴来复制，每个复制的宏成为一个名字可以独立选择的单独的宏。宏不能引用。被宏的创建删除的所有连接在宏上产生输入或输出针脚。输入端的连接产生输入针脚。默认的名字出现在针的旁边形式为 In<n>。输出端的连接，出现 Out<n>。影响有连接标记先于宏的创建的连接，保留连接标记在宏的针脚上。

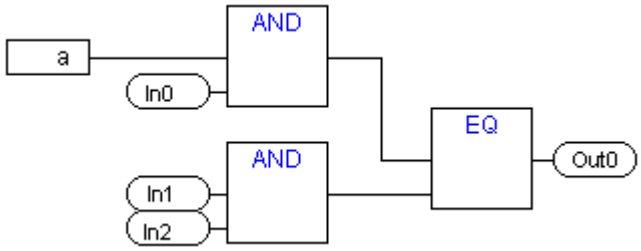
首先，宏有默认的名字“MACRO”，可以在宏使用的名字区域中来改变它。如果宏被编辑，宏的名字跟在 POU 名字后面将显示在编辑窗口的标题栏中。

例如：

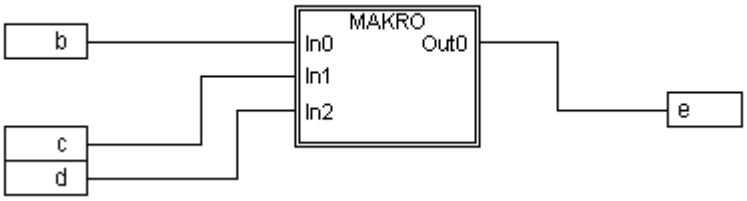
选择



宏：



在编辑器中：



’附加’ ’编辑宏’

符号： 

通过这个命令，或在 宏的主干上双击，在相关的POU中的编辑窗口中宏被打开，宏的名字跟在POU名字的后面显示在标题栏中。

为宏的输入和输出产生的针脚方框在创建过程中可以作为普通 POU 的输入和输出端处理。它们也能移动，删除，添加等等，它们的不同之处在于它们怎样显示和有没有位置索引。对于添加你可以使用按钮  （输

入）或按钮  （输出），这些按钮在菜单栏上是可用的，针框有圆角，在针框中的文本匹配在宏中显示的针的名字。

在宏方框中的针脚的顺序遵循宏中的元素的执行顺序，低顺序索引在高顺序的前面，高针脚在低针脚的前面。

在宏内的处理顺序是封闭的，换句话说宏在主 POU 中的宏的位置处作为一个块来处理，管理执行顺序的命令只在宏能起作用。

’附加’ ’扩展宏’

使用这个命令，选中的 宏重新展开并且它里面包含的元素在宏的位置处插入到POU中。与宏的针脚的连接重新显示就象连接到元素的输入或输出端一样。如果在宏方框的位置处因为缺乏空间而不能展开宏，宏将放置到右下直到空间可用。

注意：如果工程在版本 2.1 下保存，宏将都同样地展开。所有的宏在转化为其它的语言前也都展开。

’附加’ ’后退一级宏’，’附加’ ’后退所有宏’

符号： 

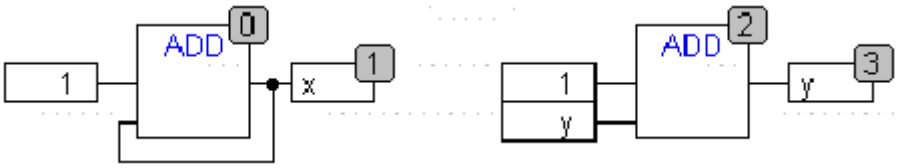
只要宏打开，这些命令在工具栏中也可用。如果 宏彼此互相嵌入，可以转换到下一个高的或最高的显示级别。

在CFC中的反馈路径

反馈路径只能直接显示在连续功能图编辑器中，不能在通常的功能模块图编辑器中显示。应该注意块的输出经常携带一个内部中间变量。中间变量结果的数据类型，对于操作数，从输入的最大数据类型。

常量的数据类型从最小可能数据类型，常量’1’采用数据类型 SINT。如果有一个带加号的反馈和常量’1’被执行，第一个输入给出数据类型 SINT 和因为反馈第二个是未定义的。因而内部中间变量只分配给输出变量。

下面的图表显示了一个带反馈的加法和带变量的加法。变量 x 和 y 在这里是 INT 型。



两种加法的区别是：

变量 y 可以初始化为一个不等于零的值但这不是左边加法的中间变量的情况。

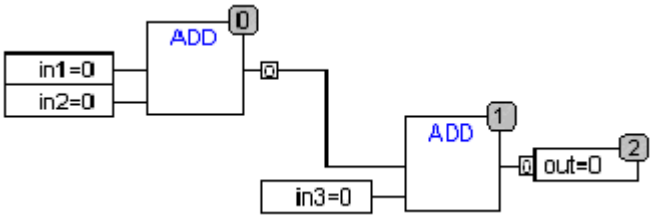
左边加法的中间变量有数据类型 SINT，在右边的中间变量有数据类型 INT。在调用 129th 后，变量 x 和 y 有不同的值，变量 x，尽管是 INT 型，包含值 127 因为中间变量已经溢出。另一方面，变量 y 包含值 129。

联机模式下的CFC

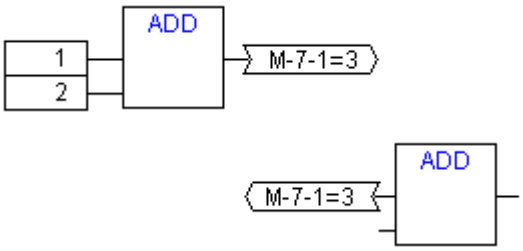
监视：

输入和输出的值显示在输入或输出方框中。常量不被监视。对于非布尔型变量，方框展开来适应值的显示。对于布尔型连接如果值为 TRUE，变量名和连接显示为蓝色，否则它们保持黑色。

在 TRUE 状态时，内部布尔型连接在联机模式下也显示为蓝色。内部的非布尔型连接的值在一个带圆角的小框中在连接的输出针脚上显示。



宏的针脚和输入或输出方框的监视一样：



带连接标记的非布尔型连接在连接标记内显示它们的值。对于布尔型连接，如果线中传递值 TRUE，线和标记名显示为蓝色，否则显示为黑色。

流程控制：

当流程控制为开时，横越的连接标记为工程选项中选择的颜色。

断点：

断点在所有的有处理顺序索引的元素上设置。程序的处理各自元素的执行之前停止并且

返回标签：

在联机模式下，名字为“RETURN”的跳转标签在第一列中和编辑器中最后元素之后自动产生，这个标签标记 POU 的末端，当执行前的分级离开 POU 时跳转到 POU ，在宏中没有 RETURN 标记插入。

分级：

当使用 ’跳过’时，将通常跳转到下一个较高级别索引的元素处，如果当前的元素是一个宏或POU，那么它的执行分支当’跳入’有效时，如果一个“跳过”从这里执行，将跳转到宏的顺序索引跟随的元素处。

切换到POU

快捷方式 ：<Alt>+<Enter>

用这个命令选中的 POU 加载到它的编辑器中，如果光标放置在文本编辑器中的 POU 名字上或如果 POU 框在图形编辑器中被选中，在内容菜单(<F2>)或在“附加”菜单中可用到这个命令。

如果你从库中处理一个 POU，那么库管理器被调用，相应的 POU 显示。

6. 资源

你需要用资源来配置和组织你的工程文件和追踪变量的值
•工程文件或网络中使用的 全局变量。
•添加库文件到工程文件中的 库管理器
•记录在线期间工作的 日志文件
•在工程中为报警处理进行 报警配置
•配置可编程控制器的 PLC配置资源
•通过任务来引导创建程序的 任务配置
•显示变量值和添加默认变量值的 监控和配方管理器
•选择 目标设置和必要时的确定的目标系统的最终配置
•作为工程选项的 工作空间
根据在 CoDeSys 中作出的目标系统和目标设置，在你的工程中也要用的到下列资源。
•用于变量图形显示的采样追踪
•用于在同一个网络中与其它控制器交换数据的变量管理器
•作为控制监视的 PLC 浏览器
•工具箱?可用性依赖对象系统?用于在 CoDeSys 内部调用它外部的工具程序

6.1 全局变量、变量配置、文件框架

’在全局变量’ 中的对象

在对象管理器的资源登记卡中，在全局变量文件夹内有二类对象（对象默认值的名字在圆括号里）。
•?全局变量列表
•?变量配置
在这些对象中定义的所有变量在整个工程中被公认。
如果全局变量文件夹未打开（文件夹前面有个加号），可以在该行双击或<回车>来打开它。
选择相应的对象，用’对象添加’指令先打开以前定义的全局变量的窗口。这个编辑器跟声明编辑器有同样的工作方式。

多个变量列表

必须在不同的对象中定义 全局变量，全局网络变量(VAR_GLOBAL)，全局网络变量(VAR_GLOBAL，目标平台专用)和 变量配置 (VAR_CONFIG)。
如果你已经声明了许多全局变量并且想更好地构造全局变量列表，那么你可以创建更多的变量列表。
在对象管理器中，选 Global Variables 文件夹或一个已有的全局变量的 对象。然后执行’工程’’对象添加’指令。给出现在对话框中的对象一个相应的名字。有了这个名字，将会用关键字 VAR_GLOBAL 来创建一个新的对象。如果你希望对某个对象进行变量配置，你可以把相应的关键字变成 VAR_CONFIG。

6.1.1 全局变量

什么是全局变量

贯穿整个工程众所周知的“标准的”变量，常数或剩余的变量可以被声明为全局变量，也可以是其他的网络用户交换数据的 网络变量。

注意：在一个工程中你可以定义一个跟全局变量重名的局部变量。在这种情况下，在 POU 的范围内使用局部变量。

不允许两个全局变量有同样的名字。例如，如果你在 PLC 配置中定义了一个变量“var1”，同时这个变量也存在于全局变量列表中，那么你将得到编译错误。

参考:

网络变量

创建全局变量列表

编辑全局变量和网络变量列表d

编辑剩余的全局变量列表

全局常量

网络变量

注意：使用网络变量必须得到目标系统的支持，必须在 目标平台设定（范畴网络功能类）中激活它。

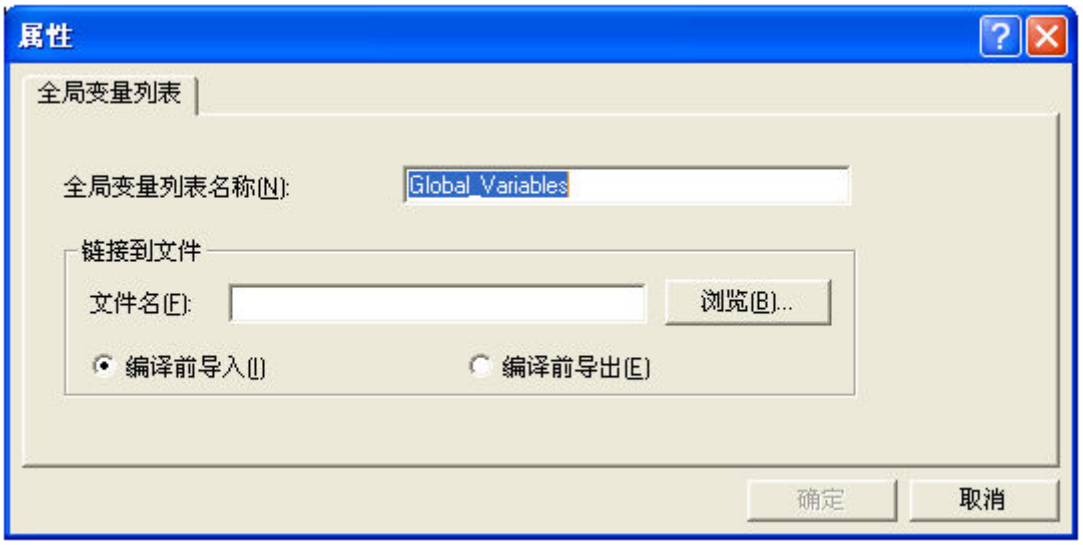
通过自动的数据交换（与通过参数管理器的非自动的数据交换比较），在CoDeSys相互兼容的控制器网络内的几个控制器系统上，可以更新网络变量的值。这不需要控制器特定的功能，但是，网络用户须为工程中的网络变量使用完全一样的声明列表和相称的转移配置。为了使前项操作成为可能，建议在创建列表时对每个控制器应用不要用手工输入变量的声明，而是用一个独立的文件来加载（见’创建一个全局变量列表’）。

创建全局变量列表

为了创建一个全局变量列表，打开对象管理器中的’资源’ ，选择条目’全局变量’，或者选一个已经存在的列表。接着，选择指令’工程’’对象’’添加’来打开全局变量列表对话框。

如果在对象管理器中标明了一个存在的全局变量列表，那么也可以用指令’工程’’对象’’配置’打开它的对话框。这个列表的配置显示如下：

创建一个新的全局变量列表的对话框



全局变量列表名称：插入列表名称

链接文件：

文件名：如果有一个输出文件(*.exp)或者是一个 DCF 文件，这些文件包含了想要的变量，那么可以对这个文件建立链接。为了做到这些，要在文件名域中插入文件的路径。点击浏览按钮将得到标准的对话框’选择文本文件’。当读入它们的时候，DCF 文件被转换为 ICE 格式。

如果希望在每一个工程编译之前从外部文件读取变量列表，那么就激活编译前输入选项。如果希望在每一个工程编译之前把变量列表写到外部文件，那么就激活编译前输出选项。

如果点击 OK 来关闭’全局变量列表’对话框，那么就创建了这个新对象。全局变量列表在对象管理器中通

过标志 来标识。你可以用'工程' '对象' '道具'指令重新打开'全局变量列表'配置对话框。

网络变量的配置

如果在目标平台设定中激活选项'支持网络变量'，那么就可采用按钮<添加网络>。点击这个按钮，对话框将会加长如上所示。如果没有激活这个选项，那么按钮<添加网络>不起作用。

连接 <n> (<网络类型>):在对话框的下部，可以建立多达四个网络连接的配置，一个配置设置为网络内的特定变量列表定义数据交换的参数。为了在网络内按要求交换数据，必使相同变量列表有一致的配置以便跟其他的网络用户相匹配。

如果尚未配置，点击'添加网络'按钮，就可得到一张在 UDP 网络的情况下，题伪'连接 1 (UDP)'的表单。每按一次'添加网络'按钮，便可得到一张'连接 n'表单（n 是表单序号）。

网络类型：从列表中选择希望的网络类型。以上列表是由目标系统条目定义的。例如，"CAN" 是 CAN 网络的缩写，或者，"UDP" 是"UDP"传输系统的缩写。

设置：这个按钮用下列配置参数打开<网络类型>的设置对话框。

UDP:

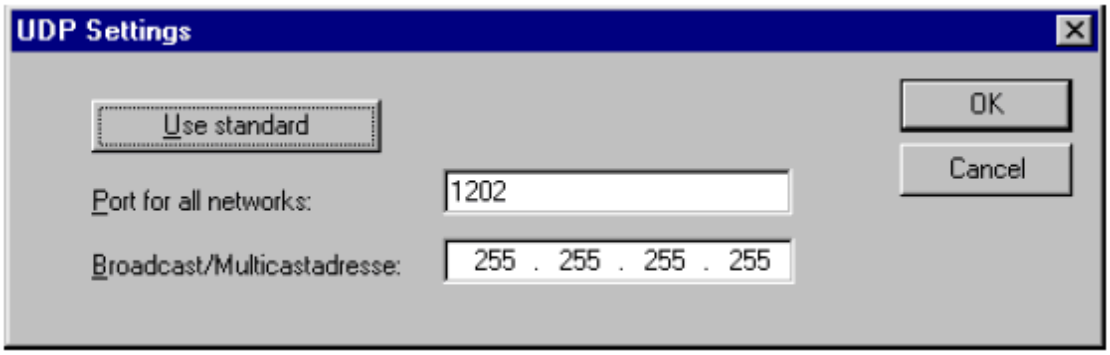
使用标准： 如果点击这个按钮，那么端口(port)1202 定义为和其他的网络用户进行数据交换的端口，广播/多路传输地址被设置到"255 . 255 . 255 . 255" 上，这意味着，数据交换将在所有的网络用户之间进行。

端口：写入要求的端口号（按照标准）。确保网络里的其他的节点定义相同的端口！如果在工程中定义了一个以上的 UDP 连接，那么根据在这里的输入将自动修改所有的配置集中的端口号。

广播/多路传输地址:如果要用标准设置，在这里写入代表子网的地址范围的地址，（例如，如果你想与 IP 地址 197 . 200 . 100 . x 的所有的节点通讯，那么其他地址应为"197.200.100.255"）。

对于 Win32 系统，广播/多路传送地址必须和 PC 配置的 TCP/IP 的子网掩码相匹配。

'UDP 设置'对话框



CAN 控制器索引：通过它传递变量。

在配置变量的传输特性时，下列选项可能被激活或者失效：

变量包:以网络许可的大小大包传输（报文），需要组装变量。如该选项无效，那么就为每个变量打包。

变量电报号码: 第一个信息包的标识号，默认值为 1，后续信息包以升序编号。

包含检验: 对每个被发送的信息包都要加上校验和接受检查校验。以便确认发送器和接收器的变量定义是同一个。不会接收校验和不一致的信息包，并将给出否定应答。

使用公认传输:接收器将应答每一个信息。发送器在一个周期内不能得到至少一个应答，马上就产生错误信息。

读取：读取列表中的变量；如果这个选项无效，通过网络发送的变量将被忽略。

启动时的请求：如果本地节点是个"读"节点（激活选项"读取"），那么一旦重新启动本地节点，就马上向所有的"写"节点请求实际的变量值然后"写"节点发送这些值,它不同于通常触发通信的其它条件(时间，事件）前提条件：在写节点配置中，必须激活选项"回复启动请求"！(参看下述内容)。

写入：写变量，应用下列选项：

回复启动请求：如果本地节点是“写”节点（“写”选项已激活）那么要回答一个“读”节点在启动时的每一个请求（选项导入请求，见上述内容）发送的读入节点要求得到答复，那意味着，此时，即使任何一个其他定义的传输引起物（时间，或者事件）不能被传送，实际变量的值也将被传送。

全局传送：在一个间隔后的指定的间隔时间内写变量。(时间表示法 T#70ms)。

变换传送:只有当变量的值变化时才写变量。在最小域内可以设置在两次传输之间的最小时间间隔。

事件传送：一旦插入在可执行变量编程 TRUE 之后，马上就写列表中变量。

在对象管理器中，全局网络变量列表通过标志 来标识。

注意：如果一个网络变量用于多个任务，对传送的时间有下列要求：当访问每个任务时，要测试决定哪个参数要用于变量值的传输（在’全局变量列表’对话框里配置）。变量值的传送与否依赖于规定的时间间隔是否已过去。在每个传送开始时，变量的时间间隔计数器将复位到零。

发送总是由起作用的控制器的运行系统承担。于是不必为数据交换提供专门的控制功能。

编辑全局变量和网络变量列表

全局变量的编辑器与声明编辑器类似。但是，注意，你不能在这个编辑器中编辑连接外部变量列表映象的列表！外部变量列表仅仅能在外部被编辑，并且，他们将在每个次打开和编译工程时被读取。

语句：

```
VAR_GLOBAL
(* Variables declarations *)
END_VAR
```

如果目标系统允许，可以只使用网络变量。也可以在这语句中定义网络变量。

例子：用链接一个输出文件*.exp 创建的一个网络变量列表，命名为 NETWORKVARIABLES_UDP 用加载一个输出文件*.exp 创建的一个网络变量列表，命名为 Network_Vars_UDP 的网络变量列表的例子：

Network_Vars_UDP D:\NetworkGlobalVars UDP.EXP

```
0001 VAR_GLOBAL CONSTANT
0002     MAX_NetVarItems_UDP  : INT := 0;
0003     MAX_NetVarPDO_Rx_UDP  : INT := 0;
0004     MAX_NetVarPDO_Tx_UDP  : INT := 0;
0005     MAX_NetVarOD_UDP     : INT := 0;
0006 END_VAR
0007 VAR_GLOBAL
0008     pNetVarItems_UDP       : ARRAY[0..MAX_NetVarItems_UDP] OF NetVarDataItem_UDP;
0009     pNetVarPDO_Rx_UDP        : ARRAY[0..MAX_NetVarPDO_Rx_UDP] OF NetVarPDO_Rx_UDP;
0010     pNetVarPDO_Tx_UDP        : ARRAY[0..MAX_NetVarPDO_Tx_UDP] OF NetVarPDO_Tx_UDP;
0011     pNetVarOD_UDP          : ARRAY[0..MAX_NetVarOD_UDP] OF NetVarSDO_UDP;
0012 END_VAR
```

编辑剩余的全局变量列表

如果运行系统支持剩余全局变量列表，那么就可以处理剩余变量。剩余全局变量有两种类型：

保存变量：在CoDeSys中，运行时间系统(off/on)或者’联机’ ’复位’不受控制的停机后，保存变量保持不变。 运行系统(停止, 开始)或者’联机’ ’冷启动’或者 下载受控制的停机后，变量保持不变。

剩余的变量被赋给关键字RETAIN，或者PERSISTENT。

剩余变量的语句定义。

语句：

```
VAR_GLOBAL RETAIN
(* Variables declarations *)
END_VAR
```

VAR_GLOBAL PERSISTENT
(* Variables declarations *)
END_VAR
Network variables (target specific) are also defined using this syntax.
用这个语句也可以定义 网络变量（指定目标）。

全局常量

全局常量的关键字是 CONSTANT。
语句:

```
VAR_GLOBAL CONSTANT
(* Variables declarations *)
END_VAR
```

6.1.2 变量配置

如果把变量定义放在关键字 VAR 和 END_VAR 之间，在功能块中，为不完全定义的输入输出指定地址是可能的。不完全定义的地址用星号识别。
例如:

```
FUNCTION_BLOCK locio
VAR
loci AT %I*: BOOL := TRUE;
loco AT %Q*: BOOL;
END_VAR
```

这里定义了两个局部 I/O 变量，一个是局部输入 (%I*)另一个是局部输出 (%Q*)。
如果你想在对象管理器资源登录卡中为变量配置设定局部 I/Os，通常可以应用对象 Variable_Configuration。对象可以重新命名，并且，为变量配置可以创建其它对象。
变量配置编辑器就象声明编辑器那样工作。
局部 I/O 配置变量必须位于关键字 VAR_CONFIG 和 END_VAR 之间。
这样变量的名字由完整的实例路径构成，在路径中用句点把各个 POUs 和实例名互相分开。在功能块中，声明必须包括一个这样的地址，它的类（输入/输出）对应于不完全指定地址(%I*, %Q*)。数据类型也必须和功能块中的声明一致。
因为实例不存在而实例路径无效的配置变量被指示出错。另一方面，如果实例变量无配置也要报告出错。为了收到全部要配置的变量的列表，可以使用'插入'菜单里的子菜单"所有实例路径"。
变量配置的例子
假设在程序中给出功能块有下列定义：

```
PROGRAM PLC_PRG
VAR
Hugo: locio;
Otto: locio;
END_VAR
```

那么正确的变量配置如下：

```
VAR_CONFIG
PLC_PRG. Hugo.loci AT %IX1.0 : BOOL;
PLC_PRG. Hugo.loco AT %QX0.0 : BOOL;
PLC_PRG. Otto.loci AT %IX1.0 : BOOL;
```

PLC_PRG.Otto.loco AT %QX0.3 : BOOL;
END_VAR

变量配置的例子

Assume that the following definition for a function block is given in a program:
PROGRAM PLC_PRG
VAR
Hugo: locio;
Otto: locio;
END_VAR

Then a corrected variable configuration would look this way:
VAR_CONFIG
PLC_PRG.Hugo.loci AT %IX1.0 : BOOL;
PLC_PRG.Hugo.loco AT %QX0.0 : BOOL;
PLC_PRG.Otto.loci AT %IX1.0 : BOOL;
PLC_PRG.Otto.loco AT %QX0.3 : BOOL;
END_VAR

参考:
插入’’所有实例路径’

’插入’’所有实例路径’

用这个指令，可以产生VAR_CONFIG END_VAR块，这个块包含了该工程中的全部有效的实例路径。为了包含已经存在的地址，不需要再入现有的声明。如果工程已被编译(’工程’’重建所有’)，可以在变量配置的窗口中找到子菜单。

6.1.3 文档框架

如果一个工程要接受多个文档，也许它们附有德语和英语注释，或者，如果想为几个使用相同的变量名字的类似的工程提供文档，那么，通过用 ’扩展’’生成文本框架文件’指令创建文档，可以为自己节省很多工作。

创建的文件可以被载入所希望的文本编辑器，可以被编辑。该文件从 DOCUFILE 行开始。接着，是工程变量的列表对每个变量分配了行：VAR 行表示一个新变量的开始；下一行是变量名；最后是空行，你可以用变量的注释来替换这行。你可以简单地删除任何对文档无用的变量。如果愿意，可以为工程创建几个文档框架。

为了使用文档框架，给出'扩展''连接文本文件'指令。现在如果向这个工程提供文档，或者打印工程其他部分，那么在程序文本中，可以把文本框架（docuframe）中产生的注释插入到所有的变量中。这注释仅仅在打印输出中出现！

'附加''生成文本框架文件''

用这指令来创建文档框架。无论什么时候只要你从全局变量中选择一个对象，这个指令都在你的控制之下。

打开一个用新名字保存文件对话框。在 name file 域中，已经输入了扩展名*.txt。选择想要的文件名。现在列出工程所有变量的文本文件已经创建完毕。

'附加''连接文本文件''

用这个指令可以选择文档框架。

打开一个用于打开文件的对话框。选择想要的文档框架，点击 OK。现在如果向整个工程提供文档，或者打印工程的某些部分，那么在程序文本中，可以把文本框架中产生的注释插入到所有的变量中。这注释仅仅在打印输出中出现！

用'附加''生成文本框架文件'指令创建一个文档框架。

6.2 报警配置

报警配置综述

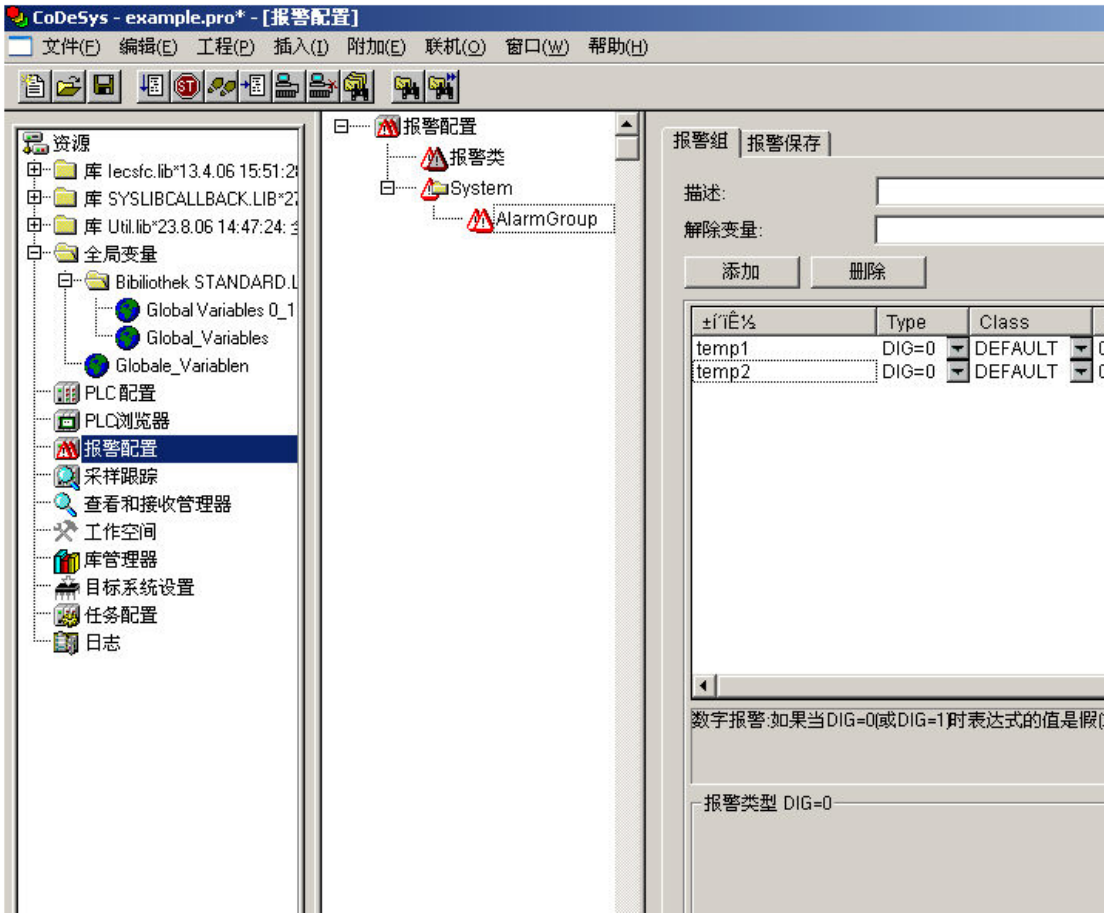
集成在CoDeSys中的报警系统可以检测关键的过程状态，记录下他们，并且借助可视元件的帮助，用户可以看见他们。在CoDeSys中或者PLC中可以做报警处理。对于在PLC上的报警处理，请看目标平台可视化（'Visualization'）设置类目。在资源中，可以利用条目'报警配置'来配置报警系统。

在这里定义报警类（Alarm classes）和报警组（Alarm groups）。报警类用于对警报的分类，这表示，它把某参数分配给报警。报警组用于具体配置一个或几个报警（对它们分配某个类和更多的参数）。类对于构造有效的报警是很有用的。在配置树中，用户通过在报头 ''System'下边插入适当的条目来定义不同的报警组。

在CoDeSys可视化中，为了报警的可视化，可以用报警表”Alarm table”。使用报警表，用户可以监视和应答警报。

如果报警事件（Alarm Events）的历史记录，要写到日志文件上，那么必须定义此文件，并且对每个警报组必须定义 存储特性。

当你在资源表中打开报警配置（'Alarm configuration'）的时候，打开带有双分窗口的对话框'Alarm configuration'，它相关的操作模式跟 PLC 配置或任务配置相似。在左边窗口显示配置树，在右边窗口打开相应的配置对话框。



通过用鼠标单击条目 '报警配置'前的加号，打开当前可以利用的配置树。如果你打算创建新配置，这树将仅仅显示条目 '报警分类'和 '系统'。

参考:

警报的一般信息，术语

报警类

报警组

存储报警

'附加' '设置' '文本框架'

警报的一般信息，术语

在 CoDeSys 里，警报系统的使用遵守下列有关报警的通用描述和定义：

•?报警：通常报警被当作特殊的状态（表达值）。

•?优先级：报警优先级，也称为“严重性”，描写报警状态的重要程度。最高的优先级是“0”，最低的有效的优先级是“255”。

•?报警状态： 为报警控制配置的表达式/变量能有下列状态：NORM（没有报警），INTO （警报刚来到），ACK （警报已经来了，用户已经应答了），OUTOF（报警状态已经终止了，警报“已经过去”，但是还没被应答!）

•? 子状态：报警状况有极限值(Lo, Hi)和超极限值（LoLo，HiHi）。 例子：一个表达式的值上升，首先

通过 Hi-limit，这样引起 Hi-alarm 的到来。在警报得到用户应答前，如果值继续上升并且超 HiHi-limit， 那么 Hi-alarm 将自动被应答，并且，只有 HiHi-alarm 留在报警列表中（那是应用警报管理器的内部列表）。在这种情况下，Histate 被称为子状态。

• 报警应答：报警的主要的目的是把报警的状态通知用户。在这样做的过程中，必须确认用户已经注意到这个信息（参看在报警类配置中分配给报警的可能的动作）。为了把报警从警报列表中除掉，用户必须应答报警。

•??报警事件： 警报事件一定不要和警报状况混淆。警报状况可以在更长的时间内有效，报警事件仅描述瞬时出现的变化，例如，从正常状态变为报警状态。在 CoDeSys 中事件的三种类型的报警配置和相应的报警状态用相同的名字(INTO, ACK, OUTOF)。

在 CoDeSys 中支持下列特征：

•?对单一报警和报警组撤销报警的产生

•?通过定义警报组和优先级来选择要显示的报警

•?存储在警报表中所有警报事件

•?在 CoDeSys 可视化中的报警表（'Alarm table'）元素的可视化

参考:

警报的一般信息，术语

报警类

报警组

存储报警

'扩展' '设置' '文本框架'

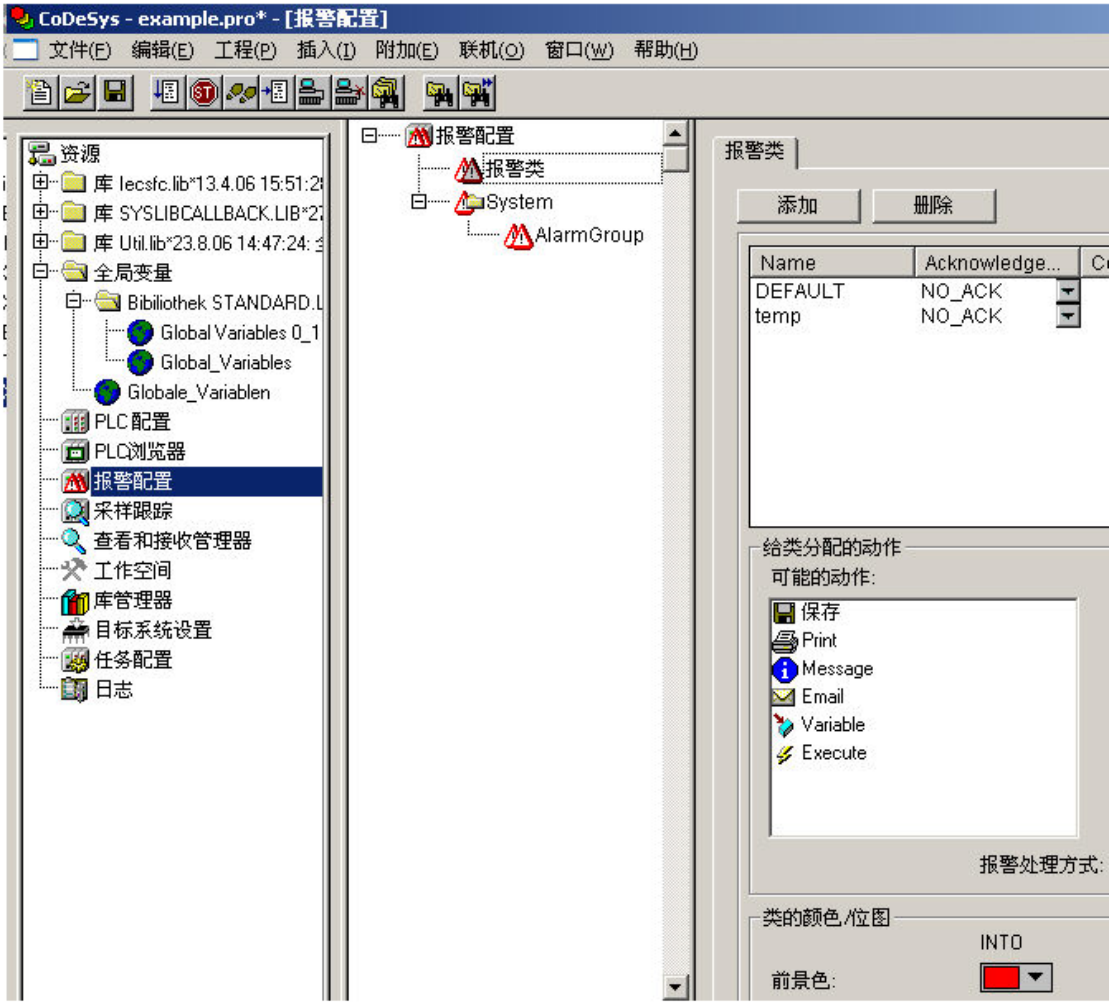
报警类

报警类用于某警报准则的一般描述，例如如何处理应答（用户确认警报）。一旦某种 报警状态被察觉之后，应该马上自动决定执行哪些动作，

报警表的可视化要使用什么样的颜色和位图。在 报警配置全局性地定义报警类，在配置 报警组时，报警类用作基础配置。

报警类的配置：

在警报配置树中选择条目'报警类'。打开配置对话框'报警类'：



点击按钮 Add 创建新的警报类。随即在上边的窗口插入一行，原来在'Acknowledgement' 栏里只有条目“NOACK” （不应答）。在 Name 栏相应的域中，为警报类定义一个名字（通过鼠标点击域来打开编辑框），如果必要，在 Acknowledgement 栏中修改应答类型。

可采用下列应答：

NO_ACK：不要求用户应答报警

ACK_INTO： （状态"INTO"，报警发生）必须由用户确认出现的报警状况。

ACK_OUT OF：（状态“OUTOF”，报警终止）必须由用户确认终止报警。

ACK_ALL：必须由用户 确认报警的出现和终止。

另外，你可以输入注释（Comment）。追加的报警类条目将被添加在列表的末尾。

用 Delete 按钮从列表中删除当前选择的条目。

对类分配动作:

在上边窗口定义的每个报警类可以分配给一列动作，报警事件发生后，应该马上执行这些动作。

在可能的动作（Possible actions）列表中选择一个动作，点击按钮">"把它分配给动作（Assigned actions）域。这个域将最终包括分配给此警报类的动作。通过按钮“>>”你可以一下子加进全部动作。通过按钮"<" 和 "<<"可以从已经完成的现有的选择中除去一个或者所有的动作。对于在'Assigned actions' 列表中标明的动作，通过"..."可以打开相应的对话框，以便定义想要的 e-mail 设置，打印设置，过程变量可执行程序及信息正文。

支持下列动作类型(可能的动作)：

动作	描述	在相应的对话框中要做的设置:

存储：	报警事件将保存在内部以便分发到 日志文件中。请注意：在这种情况下必须在 警报组的配置中定义记录文件!	在 报警存储对话框内报警组定义中进行设置。
打印：	把信息正文发送给打印机。	Printer：选择一个在本地系统上定义的打印机；Outputtext ： 应该打印的信息正文（见下述内容）
消息:	在当前的报警 可视化中，打开消息窗，显示定义的文本。	Message：在消息窗中要显示的消息正文
E-Mail:	发送包含规定消息的电子邮件。	From：发送者的电子邮件地址；To：接收者的电子邮件地址；Subject：任何主题；Mesage：消息正文（见下述内容）；Server：电子邮件服务器的名称
变量：	CoDeSys 程序的变量将得到警报状态和消息正文串。	Variable：变量名称：通过输入帮助(<F2>)，可以选择工程变量：布尔变量将指示报警状态 NORM =0 和 INTO=1，一个整型变量指示不报警状态 NORM =0, INTO =1, ACK =2, OUTOF =4；一个字符串变量得到在域中定义的消息正文;Mesage(见下述内容)
执行：	一旦报警事件发生后，马上就开始执行可执行文件。	Executable ： 要 执 行 的 文 件 名 称 （ 例 如 notepad.exe)，可以使用“...”按钮得到标准的对话框来选择文件；Parameter：调用执行文件所需的参数

消息正文的定义：

对于动作类型'信息'， '打印'， 'Email'或'变量'，可以定义信息正文，在报警事件发生时要输出信息正文。

可以用<Ctrl>+<Enter>在'信息'Email'或'变量'正文定义中插入行分隔符。

在定义报警消息的时候，可以使用下列占位符：

MESSAGE	使用在 报警组的配置中为特别的警报定义的消息正文。
DATE	日警报状态到达(INTO)的日期。
TIME	警报进入的时间。
EXPRESSION	引起警报的表达式（在警报组中定义）
PRIORITY	警报的优先级（对警报组定义）
VALUE	表达式的当前值（见上述内容）
TYPE	报警类型（在报警组中定义）
CLASS	报警类（在警报组中定义）
TARGETVALUE	报警类型目标值 DEV+ 和 DEV-（在警报组中定义）
DEADBAND	报警容差（在警报组中定义）
ALLDEFAULT	输出警报的任何信息，如同在日志文件（历史记录）中行条目描述信息。

定义警报消息的例子：

为定义消息框，在消息窗中输入下列内容：

此外，当在报警组中定义报警时，在'Message' 栏中输入"Temperature critical !"：
最后警报消息的输出如下：

注意： 如果消息正文包括在*.vis-file或 翻译文件*.tlt内，在工程语言改变的情况下，消息正文也将受到影响。但是：在这种情况下一象提交到可视化的正文一样，正文必须放在两个"#"之间（例如，在上面的例子中："#Temperature critical !#"和"TIME /EXPRESSION: MESSAGE #current#: VALUE"，以便将此正文输入到象ALARMTEXT_ITEMs那样的翻译文件）

在 报警组的配置中要定义'Save'动作的日志文件（见 6.3.4 节）。

对动作的警报事件：

.对每一个动作，要定义在哪个 报警事件下启动它。

激活要求的事件：

INTO 产生报警。Status = INTO。

ACK 用户应答已经完成。 Status = ACK。

OUTOF 报警状态结束。 Status = OUT OF。

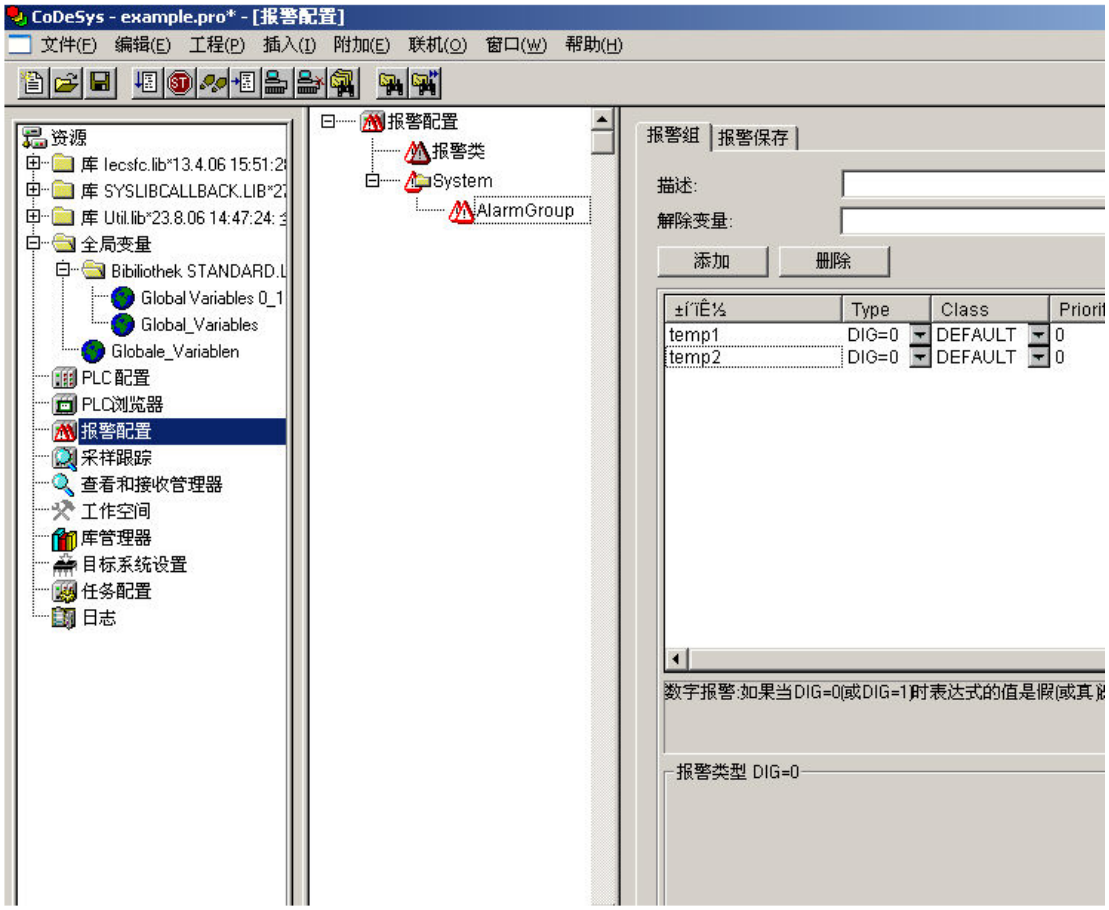
类的颜色 / 位图<类名>

每个警报类都可以得到自己的颜色和位图，在可视化元件 警报表中，他们将被用于区别报警。为可能的事件INTO, ACK 和 OUTOF（见上述内容）选择前景颜色和背景颜色。在颜色符号上单击鼠标，马上就打开选择颜色的标准对话框。为了选择位图，在灰色矩形框上的点击鼠标将打开选择文件的标准对话框。

报警组

报警组用于组织可用的警报。每个报警明确的被分配到一个恰当的报警组并且由这个警报组来管理。组里的所有报警逗能被分配一个公共的去激励变量（Deactivation variable）和若干报警存储用的公共参数。注意，即使是单个报警，也必须在报警组内配置。

通过 fFolder 元素定义报警组的层次结构。在配置树中选择警报组时，将自动显示对话框 Alarm group：



在 Description 域中，你可以输入警报组名称。

可以定义一个布尔工程变量作为去激励变量（Deactivation variable)。在这个变量的上升沿处，关闭组内的所有警报，在下降沿处它重新激活报警。

通过按钮 Add，可以把报警添加到报警组。在表格窗口插入新行，并且设置下列参数：

Expression：输入工程变量或者报警设计的表达式(例如， "a + b")。为了正确输入，推荐使用输入帮助<F2> 和智能功能

Type：可以使用如下所列的报警类型。对每种类型注意在该表外作出的相应注释和定义。

DIG=0　数字警报，一旦表达式得到 FALSE 后马上就报警。

DIG=1　数字警报，一旦表达式得到 TRUE 后马上就报警。

LOLO　模拟警报，表达式的值降到报警类型 LOLO 定义的值以下后马上就报警。你可以定义一个容差死区（Deadband)。只要表达值在容差的范围内，即使 LOLO 值降到界限以下，报警也不被激活。

LO　　与 LOLO 相应

HIHI.　模拟警报，表达式的值超过报警类型定义的值后马上报警。你可以定义容差死区（Deadband)。只要表达值在容差的范围内，即使 HIHI 值超过界限，报警也不被激活。

HI　　与 HIHI 相应

DEV-　目标值的偏差；一旦表达式的值降到报警类型 DEV-定义的值加百分比偏差以下就立即报警。百分比偏差＝目标值*（用％表示的偏差）／100。

DEV+　目标值的偏差；一旦表达式的值超过警报类型 DEV+加百分比偏差定义的值。百分比偏差＝目标值*（用％表示的偏差）／100。

ROC　　单位时间变动率；表达式的值严重偏离前面的值后马上报警。激活警报的极限值是通过值的每秒变化（变动率），每分钟变化，或每小时变化来定义。

分类：选择要求的 警报类。选择列表将提供所有的类，这些类是在工程的最后保存之前在报警类配置中定义的。

优先级：可以定义优先级的范围 0-152。0 是最高的优先级。优先级将影响报警表范围内的报警排序。

消息：在报警的情况下，为消息框定义正文。消息框由用户用 OK 来确认，但是，这 OK 不会自动应答警报！为了确认（应答）报警，必须访问报警表。通过可视化元素报警表或者通过进入该表的报警日期，确认（应答）报警是可行的。此日期可以从选择性创建的日志文件里读取。

钝化：输入一个工程变量，在该变量的上升沿，关闭所有创建的警报。

保存报警

在 警报类配置对话框中，如果把一个存储动作分配给此类，对每个警报组，都可以定义存储报警事件的文件。

在配置树中选择警报组，打开对话框表 ‘’Alarm saving’：

可以作下列定义：

文件路径：文件所在的目录路径，这个文件是在文件名称中定义的；通过按钮"..."，你可以得到选择目录的标准对话框。

文件名：保存警报事件的文件名称（例如，"alarmlog"）。文件是自动创建的，它在这里得到定义的名称加附属的数字及扩展名".alm"．。数字表示日志文件的版本号。第一个文件的数子是"0"；往后的文件根据定义的文件变动事件创建的文件－被编号为 1，2 等。（如，"alarmlog0.alm", "alarmlog1.alm)。

文件改变事件： 在这里定义引起创建新的报警存储文件的事件。可能的事件有：

Never（无），Hour（一小时后），Week(一周后），Month（一月后），TriggerVariable（在 TriggerVariable 域中定义的变量的上升沿），Number of record（超过了在 Number of records 中定义的文件记录数）

Triggervariable 和 Number of records：文件变动事件见上述内容。

删除旧文件：文件创建后的天数，过了这几天后除了最新的日志文件，删除所有的报警日志文件。

日志文件（历史）包含下列条目：

(参看栏目类型及两个报警的典型输入)

Date/Time in DWORD	Date	Time	Event	Expression	Alarm type	Limit	Tolerance	current value	class	Prio
1046963332	6.3.03	16:08:52	INTO	PLC_PRG.b	LO	-30	5	-31	Alarm_high	0
1046963333	6.3.03	16:08:53	ACK	PLC_PRG.n	HIHI	35			Warnng	9

在日志文件中可以看到的例子：

1046963332,6.3.03 16:08:52,INTO,PLC_PRG.ivar5,HIHI,,,, 9.00,a_class2,0,

1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar4,ROC,2,,, 6.00,a_class2,2,

1046963333,6.3.03 16:08:53,INTO,PLC_PRG.ivar3,DEV-,,,, -6.00,a_class2,5,

1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar2,LOLO,-35,,3, -47.00,warning,10,warning: low temperature !

1046963334,6.3.03 16:08:54,INTO,PLC_PRG.ivar1,HI,20,,5, 47.00,a_class1,2,temperature to high ! Acknowledge !

’附加’菜单：设置

在报警配置中，用指令’扩展’ ’设置’打开对话框报警配置设置：

Categorie Date/Time：

在这里设置 日志文件中报警表达格式。根据下列语句定义格式。破折号和冒号要放在引号内。

日期： dd’-’MM’-’yyyy -> 如， ″12.Jan-1993″

时间： hh’:’mm’:’ss -> 如， ″11:10:34″

Language：

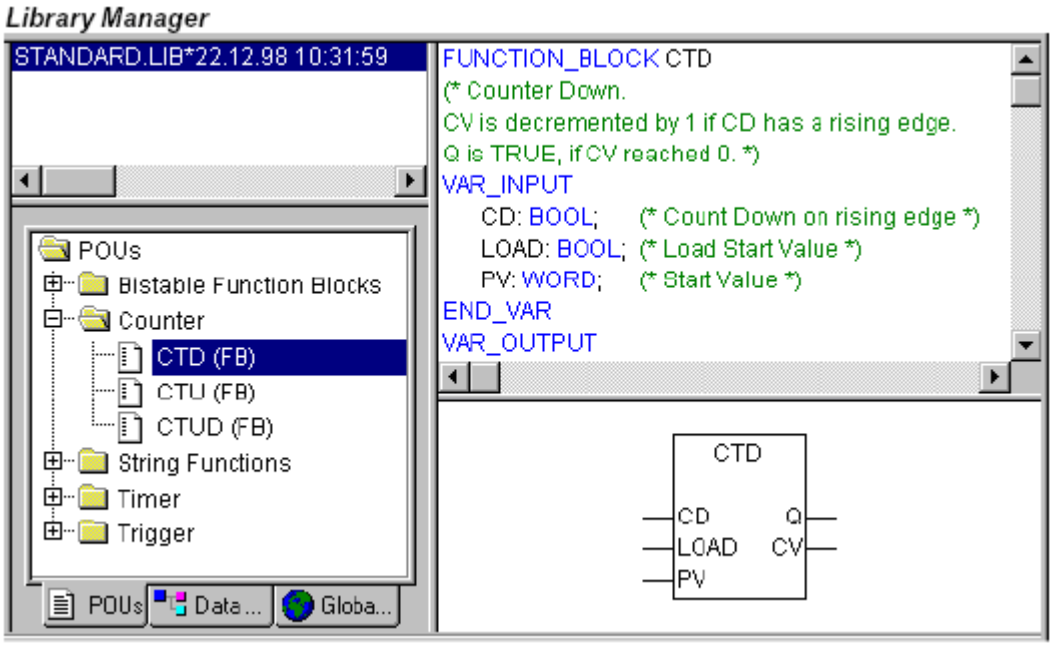
在CoDeSys中选择当改变 语言时应该使用的语言文件。注意，为了整个目的，语言文件必须包括报警配置的文字串的翻译。联系上下文，见下列描述：

- 可视化，设置它的语言，见CoDeSys可视化用户手册
- 把工程翻译成其它语言，见 4.3 节

6.3 库管理器

库管理器显示连接当前工程的所有库。库的 POUs 数据类型和全局变量像用户定义的 POUs 数据类型和全局变量那样用相同的方式被使用。

用指令’窗口’ ’库管理器’打开库管理器。在工程中贮存了它包含的各库的信息，并且在 ‘’扩展库信息’对话框中可看到。打开对话框，选择在库管理器中相应的库的名称，并且执行指令’附加’ ’特性’。



参照:

使用库管理器

标准库

自定义的库

’插入’ ’扩展库’

删除库

使用库管理器

屏幕划分器把库管理器窗口分为 3－4 部分。附属于工程的库列在区域的左上方。

在这个区域的下部，根据选择的登录卡，列出了上部区域选定库的 POUs，数据类型，可视化或者全局变量。

双击行或者按<Enter>键来打开和关闭文件夹。在关闭的文件夹前有一个加号，在打开的文件夹前有一个减号。

如果点击鼠标或者用箭头键来选择 POU，那么 POU 的声明将在出现库管理器的右上方；在右下方是图形显示器，以黑箱的形式来输入和输出。

数据类型和全局变量的声明显示在库管理器的右边。

标准库

带有"standard.lib"的库总是可以使用的。它包含了 IEC61131-3 要求的全部功能和功能块作为 IEC 编程系统的标准的 POUs，标准和操作符之间的区别是操作符通过编程系统来隐含识别，而标准的 POUs 必须连接到工程中(standard.lib)。

这些 POUs 的代码作为 C-库而存在，是 CoDeSys 的一个元件。

自定义的库

如果一个工程用它的实体编译并且没有错误，那么可以用'文件'菜单里的指令'另存为' 把它保存在库中。工程本身保持不变。将产生一个附加的文件，这文件有默认的扩展名".lib"。然后，这个库就像标准库一样可以被使用和访问。

为了达到在其它工程中使用该工程的 POUs 的目的，把该工程存储为内部库 *.lib。然后，使用库管理器可以把这个库插入到其它工程中。

如果你用其它的 编程语言（如C语言）来实现POUs，并且想让他们进入库，那么用数据类型外部库 *.lib 来存储此工程。你可以得到库文件和另一个带有扩展名"*.h"的文件。这个文件的结构象C头文件一样并且包含所有POUs，数据类型和全局变量的声明，这些声明被库采用。如果在工程中使用外部库，那么在仿真方式下将执行POUs程序，这执行是用CoDeSys写的；但是在目标方式下，将被处理用C写的程序。

如果你想给库添加许可证信息，那么点击按钮编辑许可证信息...并且在对话框'编辑许可证信息'里插入相应的设置。参见在在CoDeSys中'文件' '另存为...'和 许可证管理的相应的描述

'插入' '添加库'

用这个指令，你可以把附加的库插入到工程里。

当执行这个指令时，出现打开文件的对话框。选择你想要的带扩展名"*.lib"的库，点击 OK 关闭对话框。现在该库列出在库管理器，你可以象使用用户定义的对象那样使用这些库对象。

一旦包含需要许可证的库，但是没有发现有效的许可证，马上就得到这样的消息：库仅仅在仿真模式中可用或当前的设置目标得不到许可证。那时你可以忽略这消息或者启动关于许可证的相应动作。在编译(工程''建立').期间，无效的许可证将产生错误。在这种情况下双击错误消息或按<F4>键将打开对话框 ‘’许可证信息'，在这里你可以启动相应的动作（见关于'许可证管理器'相关文档）。

移除库

用指令'编辑' '删除'可以从工程和库管理器中删去库。

'附加' '特性'

这个指令将打开对话框'自带库（扩展库）信息'。对内部库你可以发现所有的数据，当库在CoDeSys中已经创建完成时，这些数据已经插入到 工程信息中了（包括可使用的许可证信息）。对外部库，将显示库的名称和路径。

6.4 日志记录

日志按时间顺序保存在线期间发生的动作。为达到这个目的，建立了一个二进制日志文件(*.log)。然后，在外部日志中，用户可以贮存来自相应工程日志记录的摘录。

日志窗口在脱机或在线的模式中打开，于是可以用作直接在线监视器。

参照:

'窗口' '日志'

菜单日志

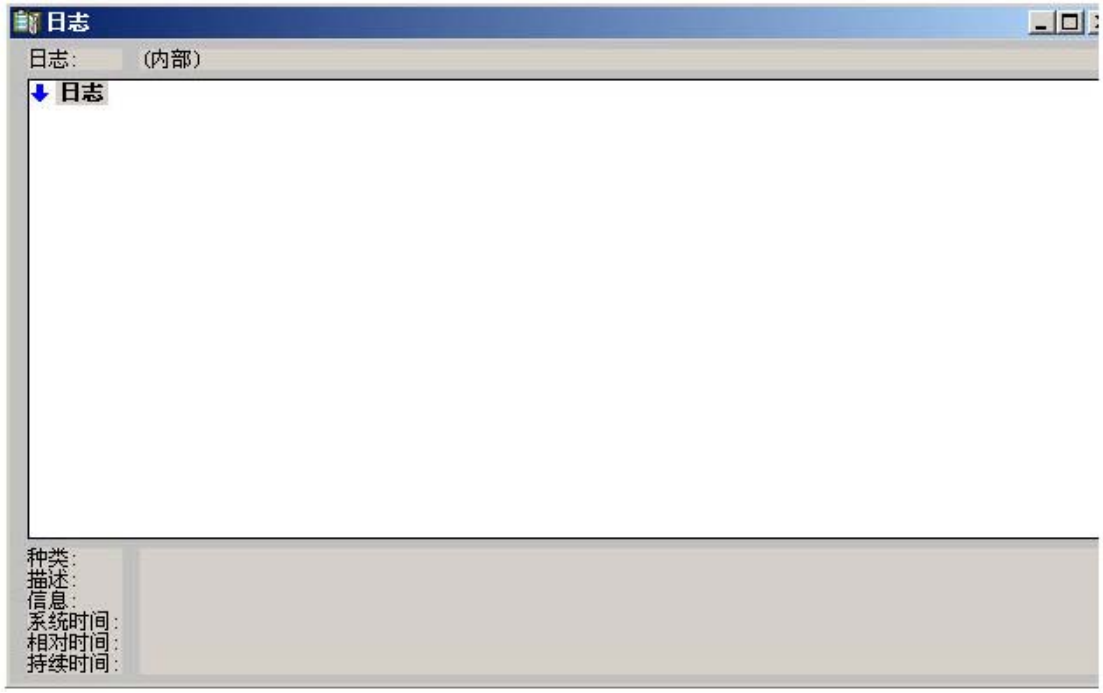
贮存文件日志记录

'窗口' '日志'

为了打开日志，选择子菜单'窗口' '日志'或者在资源表中选择条目'日志'。

在日志窗口中，当前显示的日志文件名出现在 log 后；如果这是当前工程的日志，将显示单词"(Internal)"。

在日志记录窗口中显示记录的条目。最新条目总是出现在底端。只显示这样的动作：属于在菜单'工程' '选项' '日志'的'过滤'域中被激活的类别的动作。



关于当前选择条目的可用信息显示在日志窗口的下面：

种类：特定的日志条目属于的类别。可能是下列四种类别：

•?用户动作：用户已经完成了一个在线动作（一般的来自在线菜单）。

•?内部动作：在在线层中已经执行了一个内部动作（例如，删除缓冲器或开始调试） 。

•?状态变化：运行系统的状态已经变化（例如，如果达到断点那么就从运行变为中断）。

•?异常：发生一个异常，例如通信出错。

描述：动作类型。用户动作跟他们相应的菜单指令有相同的名称；其它的动作是用英语描述并且跟相应的 OnlineXXX()功能有相同的名称。

信息：这个域包含一个在动作期间产生的错误描述。如果没有错误产生，这个域是空的。

系统时间：动作开始的系统时间，到最近的秒。

连接时间：从在线活动开始测量的时间，到最近的毫秒。

持续时间：用毫秒表示的动作延续时间。

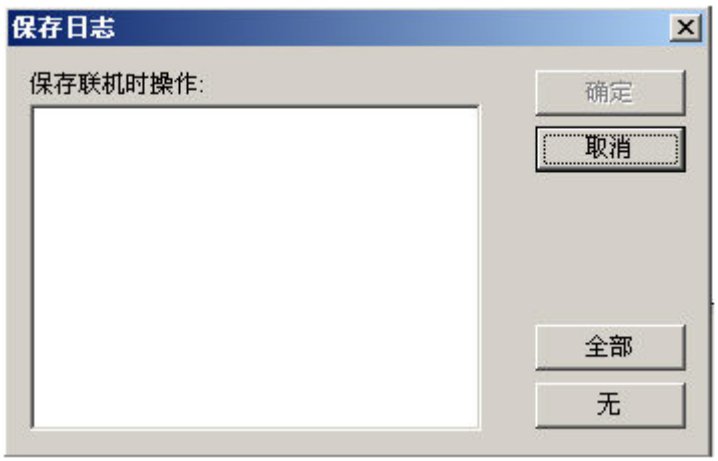
菜单日志

当日志窗口有输入活动时，在菜单条中出现菜单选项 Log 而不是'扩展' 和 '选项'里。

菜单包括下列子菜单：

Load…用标准的文件对话框可以加载和显示外部日志文件*.log。此指令不会改写存在于工程里的日志。如果日志窗口关闭并且稍后再打开，或者启动新的在线活动，那么这个载入的版本将再一次被工程日志取代。

Save…如果工程日志记录正在显示，那么只能用这个子菜单。它允许工程日志记录的摘录贮存在外部文件里。因此，将显示下列对话框，在这里可以选择要贮存的在线活动：



成功的选择后，打开贮存文件的标准对话框('保存日志')。

显示工程日志 如果当前显示外部日志，那么只能选择这个指令。它把显示切换到工程日志。

贮存文件日志记录

不管日志是否贮存在外部文件（见上述内容），工程日志自动贮存在名为<projectname>.log 的二进制文件。如果在'工程' '选项' '日志'对话框里没有明确的给出不同路径，那么文件保存在存储工程的目录里。

可以在'工程' '选项' '日志'对话框里输入在线活动的最大数。如果在记录期间超过了这个数，那么就删除最早的那个活动以便为最新的腾出空间。

6.5 任务配置

使用任务管理除了声明特定的 PLC_PRG 程序外，还可以控制工程的处理。

一个任务是一个在IEC程序处理中的时间单位。用名称，优先级和确定触发任务启动的条件的类型来 定义它。这条件可以通过时间（周期的，随机的）或者通过内部或外部的触发任务的事件来定义，例如，全局工程变量的上升沿或者控制器的中断事件。

对每个任务，可以设定一串由任务启动的程序。如果在当前周期内执行此任务，那么这些程序会在一个周期的长度内受到处理。

优先权和条件的结合将决定任务执行的时序。

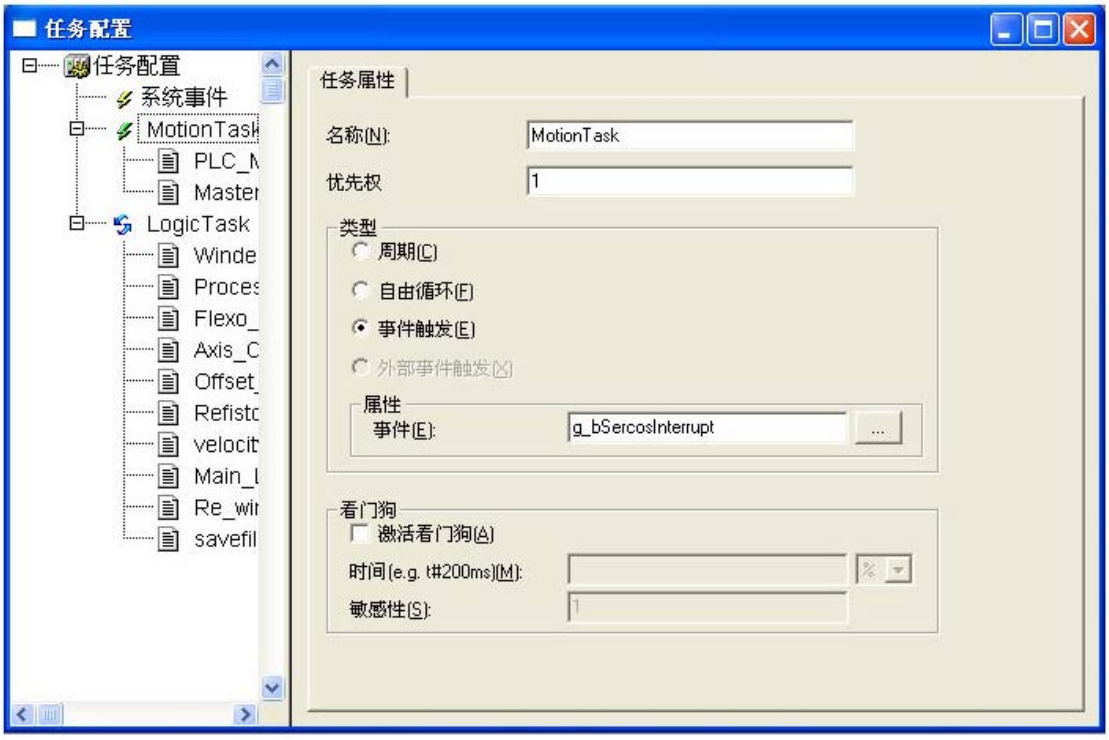
每个任务可以直接 启用或停用。

对每个任务，可以配置一个 监视器（时间控制）。

在 在线模式中，可以用图表监视任务的处理。

另外，可以把系统条件（例如，开始，停止，重启）直接与工程 POU 的执行连接起来。

在对象管理器的资源表中可以找到任务配置对象 。任务编辑器打开在两分窗口中。



在窗口的左边，用配置树描述各任务。在最顶端，你会发现'任务配置'条目。下面有'系统事件'条目和各个用任务名描述的条目，在每个任务条目下，插入指定的程序调用。每一行前有一个图标。

在窗口的右边，显示一个属于当前标明的配置树条目的对话框，你可以配置任务属性(Task properties)，程序调用(Program call)，定义系统事件(System events)的连接。在配置对话框中哪个选项可用依赖于目标系统。他们通过在目标文件中引用的描述文件定义。如果用户特定定义扩展了标准描述，那么那些扩展会显示在窗口右边的一个附加表项'参数'中。

注意：在几个任务中不要用相同的字符串功能(见 standard.lib)，因为这可能由重写引起的程序错误。

任务配置的工作状态

•在上下文菜单中可找出最重要的指令（鼠标右键）。

•任务配置标题是"任务配置"，如果在标题前有加号，那么顺序列表是关闭的。通过在列表上双击鼠标或者点击<Enter>，可以打开列表。出现减号。通过再次双击，可以再次关闭列表。对每个任务，有一个附属的调用程序列表。用同样的方式可打开和关闭这个列表。

•用 '插入' '插入任务' 指令，可以插入一个任务。

•用 '插入' '附加任务' 指令，可以在配置树底端插入一个任务。

•用 '插入' '添加调用程序'，可以把一个程序调用分配给在配置树中选择的实际任务。

•进而，对于配置树中的每个表项，相应的配置对话框出现在窗口的右边。选项可被激活 / 关闭。对编辑域可以输入。取决于配置树中选择的表项，有定义任务属性'任务属性'（见 ' 插入任务'）的对话框，定义程序调用'程序调用'（见 '添加调用程序'）或者系统事件 系统事件表格的对话框。 一旦把此窗口区设置给配置树，配置树立即接受在对话框中作出的设置。

•任务名称或程序名称也可以在配置树中编辑。为此，在名称上点击鼠标或者选择此表项按<Space> 键来打开编辑框。

•可以用箭头键选择配置树中的上一个或下一个表项。

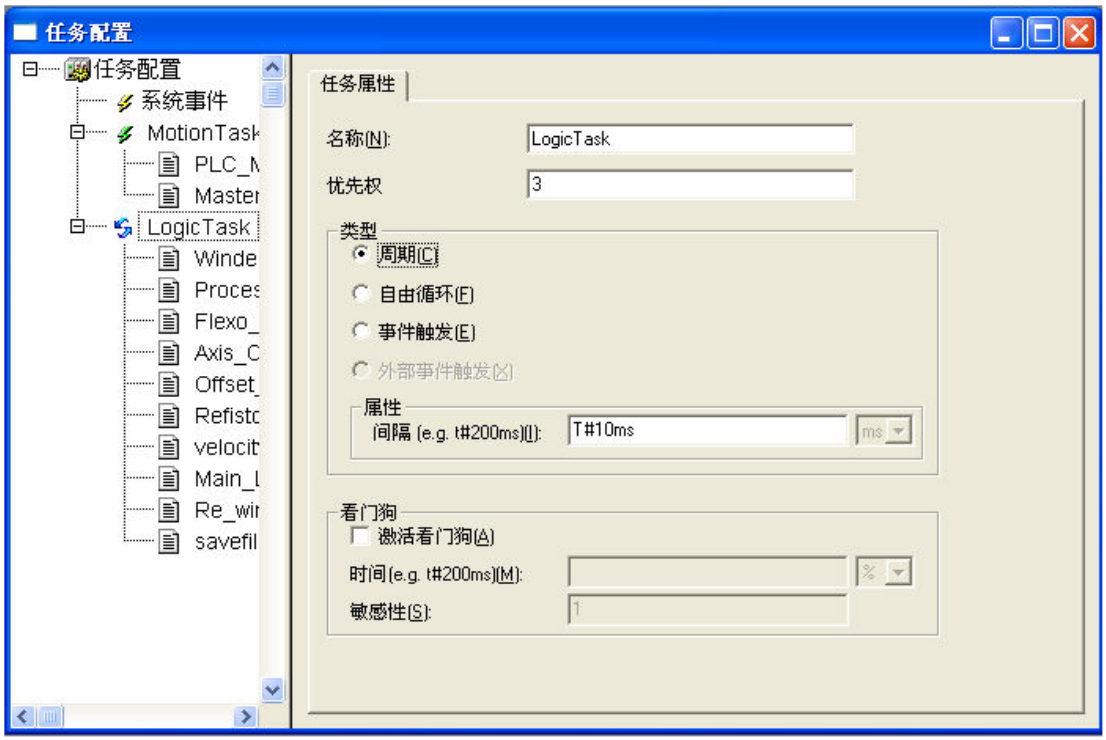
•通过点击任务名或程序名，或者通过按<Space bar>，可以在名称周围设置一个编辑控制框。然后可在任务编辑器中直接改变名称。

'插入' '插入任务' 或 '插入' '扩展任务'

用这个指令，可以把新任务插入到任务配置。每个表项由符号和任务名称构成。

如果选择了一个任务或表项'系统事件'，那么你可以用指令'插入任务'，将一个新的任务插到被选的任务之后。如果选择输入'任务配置'，那么可以用'扩展任务'，把一个新的任务插入到现有的列表的表项底端。由目标系统定义任务的最大数量。注意，可能已经把某个任务号分配给了PLC配置模块（在 cfg-file中定义）

在插入一个任务时，将打开设置任务属性的对话框：



插入要求的属性：

名称： 任务名称，在配置树中用这个名称描述任务；在选择表项上单击鼠标或按<Space>键后，可以编辑名称。

优先权(0-31)：(0 和 31 之间的数字；0 是最高优先权， 31 是最低优先权)。

类型：

循环（ ）： 根据在域'Interval'中的给出的时间定义，周期处理此任务（见下述内容）。

自动循环（ ）：程序启动后，马上就处理此任务，并且在一次运行结束时，自动重启动一个连续循环。没有定义循环时间。

触发事件（ ）：一旦在 Event 域中定义的变量达到上升沿马上启动此任务。

触发附加事件（ ）：一旦在 Event 域中定义的系统事件发生后，马上启动此任务。在选择列表中支持和提供哪些事件依赖于目标系统。(不要和系统事件混淆)。

属性：

间隔(用于 '循环' 或'自动循环'类型)：任务重启动的时间周期。你只要输入一个数字，然后在编辑域后的选择框中选择想要的单位(毫秒[ms]或者微秒[μs])。重画窗口后，就会用 TIME 格式中马上显示输入(例如"t#200ms")但是你也可以直接用 TIME 格式输入值。用[ms]的输入总是显示为纯粹的数字(例如，"300")。

单个 (用于 '触发事件' 或 '触发扩展事件'类型)：是一个其上升沿被察觉后，马上触发任务启动的全局变量。用按钮...或者输入帮助键<F2>得到一个包含所有可用的全局变量的列表。

如果在上述两个域中均没有输入，那么任务间隔将取决于运行系统(见运行文件)；例如，在这种情况下，对 CoDeSys SP NT V2.2 和更高版本使用 10 ms 间隔。

监视器：

激活监视器： 当激活这个选项时(☑)，一旦处理所用时间超过在'时间'域中定义的时间，任务马上以出错状态停止。

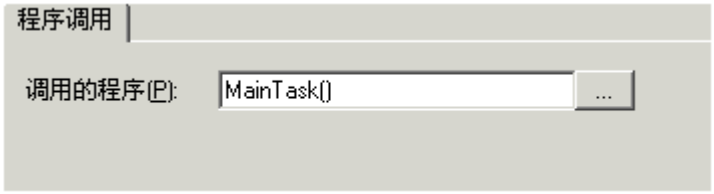
时间 (e.g.: t#200ms)：监视时间；在这一段时间期满后，将激活 watchdog，除非任务尚未中止。

灵敏度：允许不发错误的监视时间超时次数。

'插入''添加调用程序'或 '插入''扩展程序调入'

在任务配置中，用这些指令可以打开向任务输出程序调用的对话框。在任务配置树中每个表项都由符号(📄)和程序名称组成。

用'插入程序调入'，在被选择程序调用前，插入一个新的程序调用，用'扩展程序调入'，将这个程序调用 l 添加到现有的列表或程序调用的底端。



在域'程序调入'中指定该工程中的有效程序名称或用Select按钮打开输入帮助（导入向导） 以选择一个有效的程序名。程序名称也可以在配置树中修改。为此选择表项输入按<Space>键或者点击鼠标打开编辑域。如果选择的程序需要输入变量，那么用通常的形式和被声明的类型输入这些变量（例如，prg(invar:=17)）。

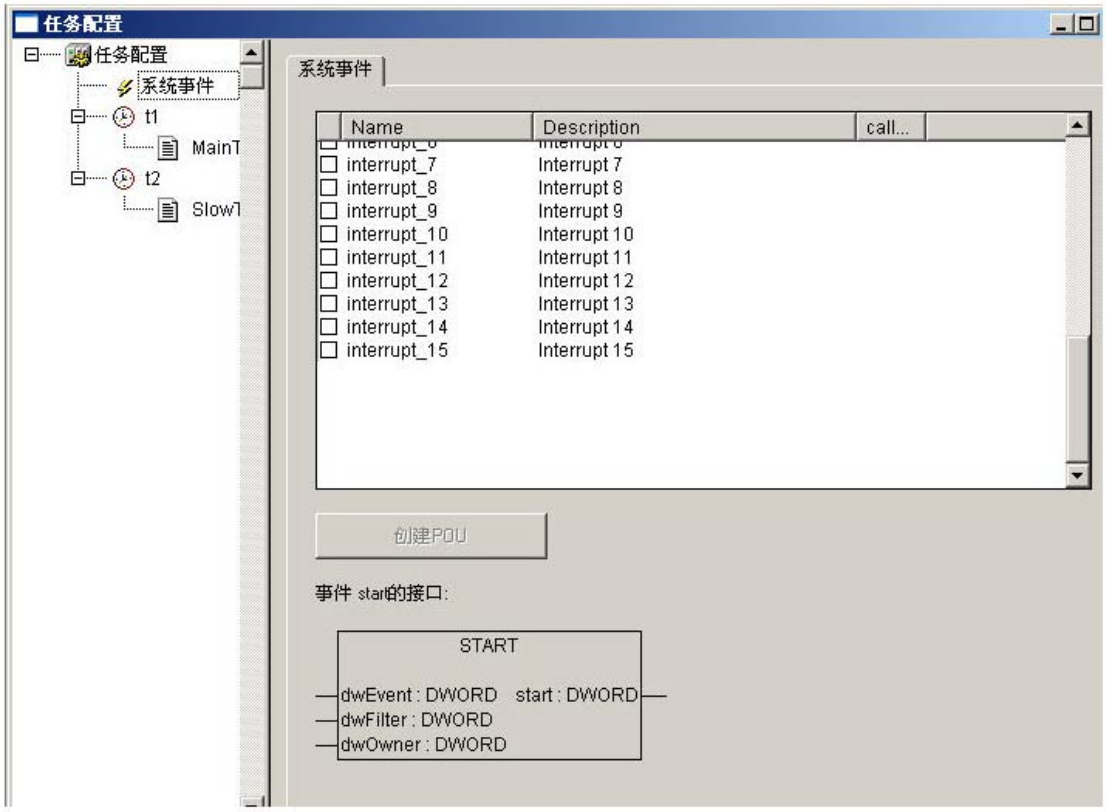
在线模式中，程序调用的处理将根据任务编辑器中的次序（从上到下）进行。

注意: 在几个任务中不要使用相同的字符串功能（见 标准库元素），因为，在这种情况下任务处理期间可能对变量产生冲击。

系统事件

用系统事件而不是任务也能调用工程的 POU。可用的系统事件是目标系统特定的（在目标文件中定义）。目标的标准事件列表可以通过用户特定事件来扩展。例如，可能的事件有：停止，开始，在线，改变。

在任务配置编辑器中把系统事件分配给 POUs。用对话框'事件'，在任务配置树中表项 □" 系统事件"后，马上就打开此对话框：



对每个事件的描述：名称 和 描述按目标文件中的定义显示，在 called POU 栏中，可以输入，事件产生后马上要调用的工程 POU 名。

为此，使用帮助键(<F2>)或者手工输入已经存在的 POU 的名称（例如，"PLC_PRG" 或 "PRG.ACT1"），或者插入尚未存在的 POU 的名称。为了创建这个 POU，按 Create POU 按钮。于是，在对象管理器中插入此 POU。事件需要的输入输出参数自动定义在 POU 声明中。在分配表下，用图形显示当前选择的事件和要求的参数。

如果你实际上希望用事件调用 POU，在分配表(☐)激活此表项。用鼠标点击控制框完成激活/关闭 。

正在处理哪个任务？

•?任务的执行应用下列规则

•?执行满足条件的任务，即，如果其指定的时间到期或在其条件（事件）变量成县上升沿。

•?如果几个任务都具备有效的条件，那么将执行最高优先权的任务。I

•?如果几个任务都具备有效的条件和同等的优先权，那么首先执行等待时间最长的任务。

•?系统调入的处理将根据任务编辑器中的顺序（从上到下）完成程序调用处理。

联机模式下的任务配置

在线模式中，每个任务的状态和通过的周期数将在配置树中显示。在图表中监视时间流动，其前提是：在工程中包含库 SysTaskInfo.lib 和 SysTime.lib，以提供内部计算任务次数的功能。一旦支持任务监视的目标设置后，就立即自动包含了这些库。

在结构树中的显示任务状态：

在线模式中，任务的当前状态和已经通过处理的周期数显示在配置树中任务表项末端的括号里。这个更新的间隔跟通常 PLC 值的监视一样。可能的状态有：

Idle	自最后一次更新以来，还没有启动过；尤其是用于事件任务
Running	从最后一次更新以来至少启动过一次
Stop	停止
Stop on BP	因为达到任务中的断点而停止

Stop on Error 错误，例如，被零除，页面错误等

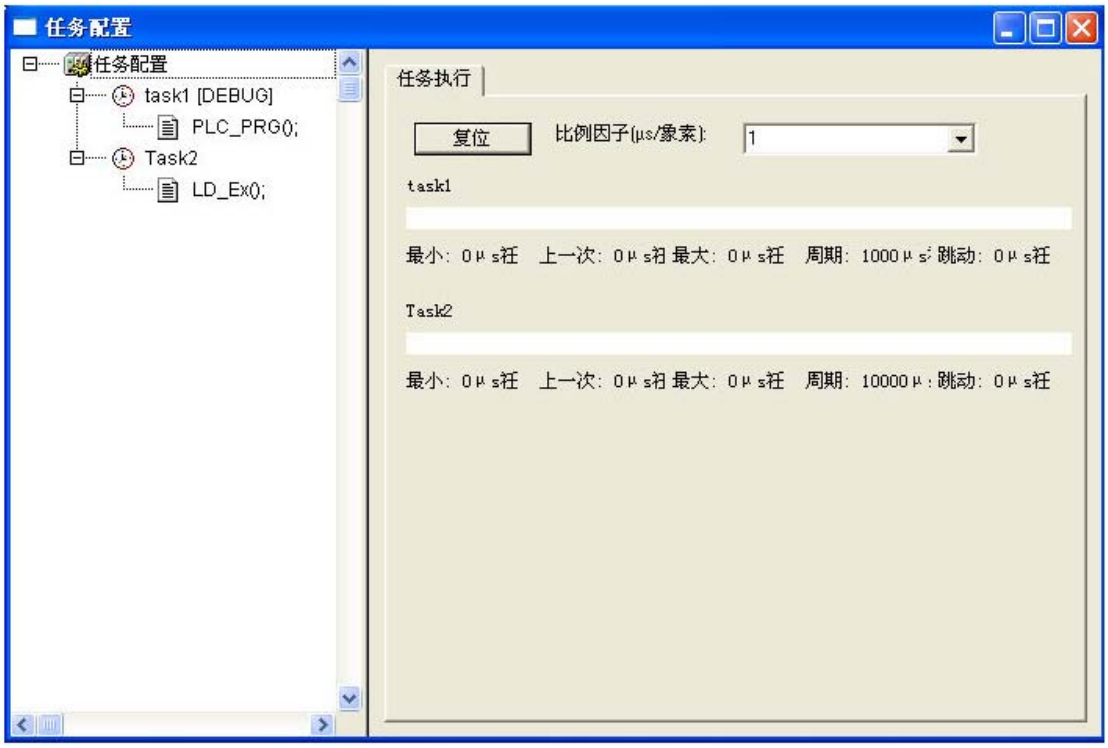
Stop 已经超过周期时间了

Watchdog

在’遇错停止’ 或 ’停止监控’状态下，任务项将显示红色。

任务的时间流显示

如果在配置树中选择’任务配置’表项，那么将在窗口右边用条形图显示任务的使用情况：



对每个任务，都显示一个条形图。条的长度表示了周期的长度。在条的下面，下列测量值用用条的相应的标记来图解。

Min: 测得的最小运行时间 μs

Akt: 上一次测得的运行时间 μs

Max: 测得的最大运行时间 μs

Cycle: 周期的总长 μs

Jitter: 测得的最大抖动量精确到 μs

复位:按钮可以用来把 Min., Max. 和 Jitter 的值归零。

图表的比例（微秒 / 像素）可以通过在 Scaling [μs/Pixel]的选择列表的帮助来调节。

在上下文菜单和’Extras’菜单中的其它在线功能：

’附加’ ’设置调试任务’

用这个指令，在 任务配置中，在线模式下设置一个调试任务。出现文本[DEBUG]设置任务后。

然而，仅仅对于这个任务具有调试能力。换言之，只有由此设置的任务运行的程序，才能停到断点。

’附加’ ’启用/禁用任务’

用这个指令，可以使在任务配置中当前标出的任务启用或停用。在程序处理期间，不考虑停用的程序。在配置树中，它用灰色的表项表示。

’附加’ ’调用堆栈’

在 任务配置中这个指令可以在扩展菜单中采用。如果在调试期间程序停在断点，它可以用来显示相应的

POU的调用栈（调入堆栈）。为了这个目的，必须在任务配置树中选择 调试任务。打开窗口’调入任务中的堆栈〈task name〉’。得到POU的名称和断点位置(例如， ″prog_x (2)″ for line 2 of POU prog_x)。下面以向后的顺序显示全部的调用栈。如果按按钮’转到’，视点将跳到在调用栈中当前标出的POU中那个位置。

6.6 监控和配方管理器

在监控和配方管理器的帮助下，可以观察所选变量的值。监控和配方管理器还可以用确定的值预置变量并且以组的形式把他们传送给PLC (’写入配方’)。用同样的方式，当前PLC的值可以读入和储存在监控和配方管理器里(’读取配方’)。这些功能对于设置和输入控制参数是有用的。

在监控和配方管理器的左栏显示所有 建立的监控表。点击鼠标或箭头键来选择这些列表。在监控和配方管理器的右边，显示在任何给定的时间都可应用的变量。

为了使用监控和配方管理器工作，打开对象管理器中的″资源″选项卡中打开监控和配方管理器。

脱机方式下的监视和收据管理器

在脱机方式下，用’插入’’新建监控列表’在监视和收据管理器中创建几个监视列表。

为了输入被监视的变量，可以用输入帮助召唤一个包含所有变量的列表，或者根据下列表示法用键盘输入变量：

〈POUName〉.〈Variable Name〉

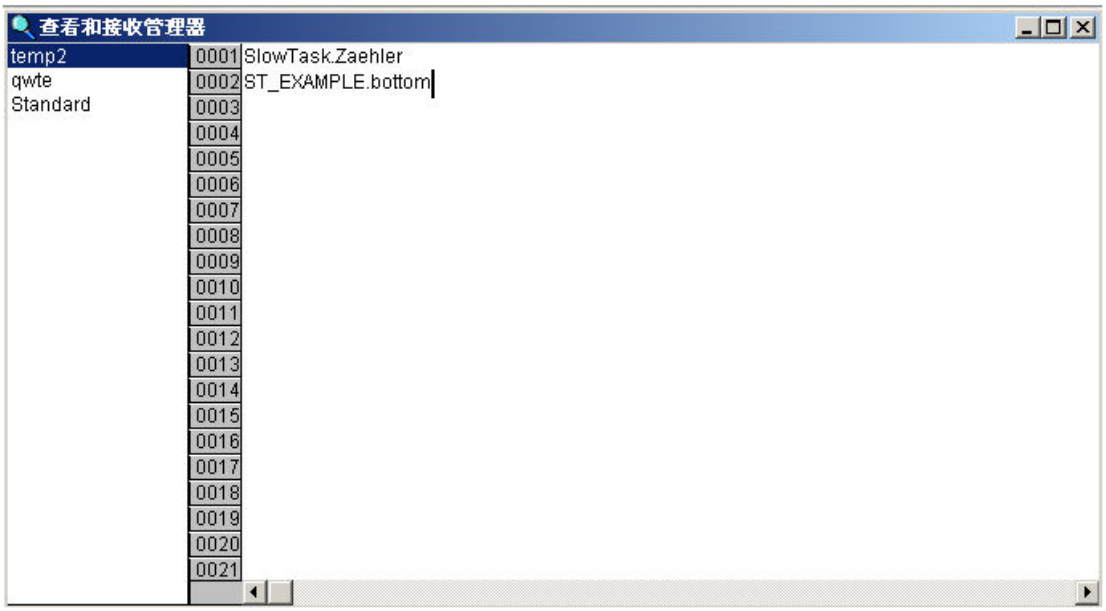
采用全局变量时则可省去 POU Name。从点开始。再次，变量名称可包含多层。可直接输入地址。

多层变量的例子：

PLC_PRG.Instance1.Instance2.Structure.Componentname

全局变量的例子：

.global1.component1



在监视列表中的变量可以用常数值预置。那意味着在线模式中可以用 ’附加’’写入配方’指令把这些值写入变量。必须用:=来分配变量的常数值：

例如：

PLC_PRG.TIMER:=50

在例子中，PLC_PRG.COUNTER 变量用值 6 预置

’插入’’新建监控列表’

在脱机方式中用这个指令，可以把新的监视列表插入到监控和配方管理器。在出现的对话框中输入要求的监视列表名。

’附加’ ’删除监控列表’

用这个指令可以在监控和配方管理器中改变监控列表名称。在出现的对话框中输入监视列表新名称。

’附加’ ’保存监控列表’

用这个指令可以保存 监控列表。它打开保存文件的对话框。文件名用监控列表名来事先调整并且得到扩展名″*.wtc″。

用’附加’ ’读取监控列表’可再次加载已保存的监控列表。

’附加’ ’读取监控列表’

用这个指令可以重新加载已保存的监控列表。它打开一个为打开文件的对话框。用扩展名″*.wtc″选择想要的文件。在出现的对话框中，可以给监控列表一个新的名称。预制不带扩展名的文件名。

用’附加’ ’保存监控列表’，可以保存监控列表。

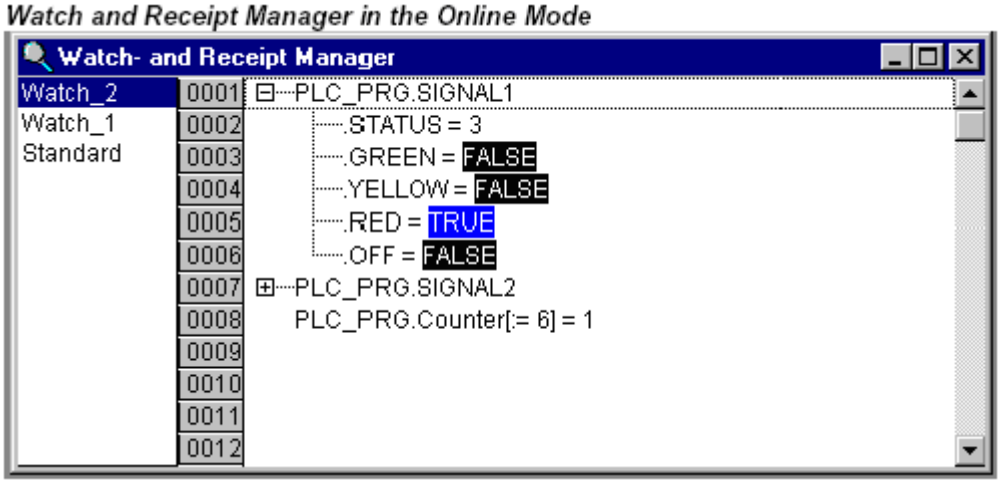
在线模式下的监控和配方管理器

在线模式中，显示输入变量的值。

结构体的值（数组，结构或者功能块实例）用加号在标识符前面作记号。用鼠标点击加号或者按<Enter>，打开或关闭变量。如果在监控列表中选择一个功能块变量，那么就把相关的上下文菜单扩展到包含’缩放’ 和’打开实例’。

为了输入新的变量，用 ’附加’ ’激活监控’指令关闭显示。输入变量后，可以用相同的指令重新激活值的显示。

Watch and Receipt Manager in the Online Mode



在脱机方式下，可以用常数值预置变量（通过在变量后输入:= <value>）。在线方式下，用 ’附加’ ’写入配方’指令可以把这些值写入变量。

用’附加’ ’读取配方’指令，可以用变量现有的值替换变量的预设值。

注意：只能加载那些在监控和配方管理器中选择的监控列表值！

’附加’ ’监控激活’

在 监控和配方管理器在线方式下， 用这个指令打开或者关闭显示。如果显示激活，记号(?)将出现在此菜单项前。

为了输入新的变量或预置一个值（见脱机方式），通过此指令关掉显示。输入变量后，可以用相同的指令激活值的显示。

’附加’ ’写入配方’

在监控和配方管理器的在线方式下，用这个指令可以把预设值写入变量（见脱机方式）。

注意：只能加载那些在监控和配方管理器中选择的监控列表值！

'附加' '读取配方'

在监控和配方管理器的在线方式下，用这个指令可以用变量现有值替换变量的预设值（见 脱机方式）。

例如：

PLC_PRG.Counter [:= <present value>] = <present value>

注意: 只能加载那些在监视和收据管理器中选择的监视列表值!

强制值

在监视和收据管理器还可以'强制值' and '写入值'。如果点击某个变量值，那么就打开对话框，在这里可以输入新的变量值。改变的变量在监视和收据管理器中以红色显示。

6.7 工作空间

资源表项中的这个对象提供当前设置的工程选项的一个映象（见 4.2 节，工程选项)。如果打开它，可以得到已知的类别的选项'对话框。

6.8 对象系统设置

"对象设置"是一个资源对象。在这里要定义工程要使用什么样的对象系统和怎么配置它。如果用'工程''新建'开始一个新工程，那么打开一个目标设置对话框，意味着为目标预先定义一个配置。

可用的目标列表依赖于哪个目标支撑软件包 TSP（对象支持包)安装在电脑中。这些描述平台的特定的基本配置和定义，在CoDeSys 目标设置对话框中用户可以修改。

注意: 如果无有效的 TSP，那么在目标系统选择框只能设置为'None'。这将自动切换到仿真模式不可能设定配置。

参照:

对象支撑软件包

对象设置对话框

对象支撑软件包

启动前，通过安装程序插入对象的帮助来安装目标支撑软件包(TSP)，安装程序是 CoDeSys-Setup 的一部分。

目标支撑软件包(TSP)包含对于用 CoDeSys 建立的程序控制一个标准平台所必需的所有文件和配置信息。必须配置的有：代码生成器，内存安排，PLC 功能，I/O 模块。另外，必须链接库，网关驱动程序，用于错误消息的 ini-files 和 PLC 浏览器。TSP 的核心组成部分是一个以上的目标文件。目标文件管理那些配置目标系统所必须的附加文件。几个目标文件可以共享这些附加的文件。

目标文件的默认扩展名是*.trg，格式是二进制的。若干附加定义连接到在目标文件中的条目上，它们决定用户是否能看见和编辑在 CoDeSys 对话框中设置。

在 TSP 安装期间，每个目标的目标文件被放入一个 0 单独的目录中并且注册了它的路径。根据 Info 文件 *.tnf 的信息，把相联系的文件拷贝导计算上。目标目录的名称跟目标的名称一样。推荐把目标专用的文件储存到用生产商的名字命名的目录中。

当启动 CoDeSys 时，读入用 TSP 安装的文件。用工程保存在 CoDeSys 对话框中完成的目标设置。

注意： 如果使用新的目标文件或者如果改变现有目标文件，必须重新启动 CoDeSys 来读更新的版本。

对象设置对话框

当创建新的工程时，对话框对象设置自动打开。用选择对象管理器资源登陆中选择子菜单'对象设置'也能打开此对话框。

选择在配置中提供的目标配置中的一个。

如果没安装目标之称软件包,只能选择'None'，这意味着在仿真模式下工作。如果选择了一个已安装了的

配置，那么它依赖于目标文件中的条目，在 CoDeSys 对话框中可以定制它的配置。如果选择了一个在计算机上没有有效的许可证的配置，CoDeSys 要求选择另一个目标。

如果选择了一个在相应的目标文件中有条目"隐藏设置"的配置，那么只能看见配置的名称。否则，有五个可用的对话框来修改给出的配置：

目标平台（Target Platform）

内存安排（Memory Layout）

常规配置（General）

网络功能（Networkfunctionality）

可视化（Visualization）

注意 !: 要知道，对预先确定的目标配置的每次修改可以引起目标性能和特性的重大变化！

如果你希望把目标设置恢复导目标文件给出的标准配置，请按<Default>。

6.9 PLC 配置

6.9.1 综述

在对象管理器中，  PLC配置是登录卡**Resources**中的一个对象。你必须用PLC配置编辑器描述建立工程的硬件，输入输出的数量和位置对于程序的实现尤其重要。用这个描述，**CoDeSys**检验程序中使用的IEC地址是否真正存在于硬件内。

配置编辑器，工作的基础是配置文件（*.cfg；见下述关于版本兼容性的注释）和设备文件（例如，*.gsd, *.eds）。这些文件贮存在目标文件里定义的目录中（见，目标设置），并且当在**CoDeSys**中打开工程时，读取这些文件。你可以随时向这个目录添加文件。

配置文件*.cfg描述基本的配置，它是在配置编辑器中制定的，并且它定义了在哪个范围内用户可以定制这个配置。

注意：一旦基础的配置文件(*.cfg)被修改，必须在CoDeSys中重做配置！

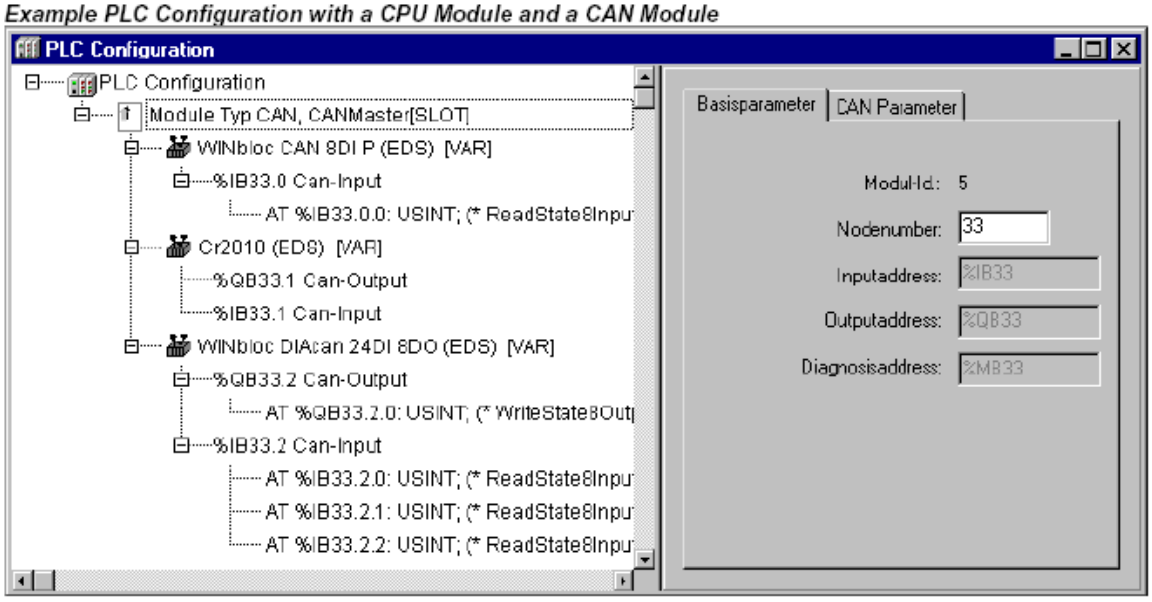
注释：关于版本的兼容性：在CoDeSys V2.2中，为PLC配置执行新的格式。从此基本配置文件的版本必须用扩展名***.cfg**。而以前版本的CoDeSys要求配置文件有扩展名***.con。**但是：在目标文件中你可以决定使用“旧的”配置器，即使当用V2.2或者更高版本打开旧的工程时。这避免了必须创建新的配置文件，继续可以使用*.con文件。如果这个选项没在目标文件中设置，那么你可以把旧的PLC配置（这配置贮存在工程中）转换成新的格式，以提供相应的新的*.cfg-文件。在’Extras’ ’Convert’中可见到更多的详细资料。

CoDeSys配置编辑器不仅允许配置I/O模块而且可以配置CAN 和 Profibus模块。

如果目标系统支持，可以从PLC中获取如下信息：1. 在PLC配置中直接可使用的实际的硬件配置的一瞥，2. 以CoDeSys消息显示的诊断消息，3. 在PLC配置对话框中显示的状态信息。

用户最终定制后，把配置的二进制映象发送到PLC：

Example PLC Configuration with a CPU Module and a CAN Module



PLC配置在编辑器中以树形结构显示并且用菜单指令和对话框来编辑。配置包括输入和／或输出元素及管理元素，管理元素本身还有子元素（例如，CAN-bus或者有8个输入的数字输入卡）。

对输入输出，可以分配符号名来访问。输入或者输出的IEC地址放在符号名的后面。

6.9.2 PLC配置中的工作

配置编辑器被分为两部分。左边窗口显示配置树。树的结构和组成部分基本上来源于配置文件中建立的定义（标准配置），但是它可以由用户在CoDeSys PLC 配置中修改。在右边窗口显示当前可用的配置对话框。

窗口的右部默认都是可视的，但是可以通过子菜单'Extras' 'Properties'来消隐。

在配置树的顶部是带有名称的根模块条目，这个名称已在配置文件中定义。

下面是分层缩进的其它的配置元素：不同类型的模块（CAN, Profibus, I/O），通道或者位通道。

元素选择

为了选择元素，在相应的元素上点击鼠标，或者用箭头键，把点线矩形框移动到想要的元素上。

以加号开始的元素是结构元素它包含子元素。选择元素双击加号或者按<Enter>键，以打开元素。你可以用相同的方式关闭打开的元素（元素前面是减号）。

插入元素，'Insert' 'Insert element', 'Insert' 'Append subelement'

根据在配置文件中的定义和采用的设备文件（在打开工程时已读入，在配置树中已自动安置了基本构成元素。如果选择了这些元素中的一个，只要在配置文件中定义允许，并且所需的设备文件有效，就可以增加下层元素：

'Insert' 'Insert element'：在当前在配置树中标明的元素前，可以选择并且插入元素。

'Insert' 'Append subelement'： 在结构树中当前表明的元素下选择和插入子元素，该子元素插在最后位置。

在上下文菜单（鼠标右键或<Ctrl>+<F10>键）中可找到最重要的指令。

请注意：如果目标系统支持，为了在CoDeSys PLC配置中插入模块，可以使用现有的硬件的搜索。

替代／转换元素，'Extras' 'Replace element''

依靠配置文件的定义，当前选择的元素可以被其它元素替代。用同样方法可以转换通道，这些通道是用象输入输出元素一样的方式建立的。。

符号名

在配置文件中可以定义模块和通道的符号名。它放在配置编辑器中，各个元素IEC地址的‘AT’前。在配

置文件中还定义符号名在配置编辑器中是否可编辑或者插入。为输入符号名，在配置树中选择要求的模块或者通道并且在IEC地址前用鼠标点击'AT'打开文本域。双击符号名后，用同样的方式你可以编辑现有的符号名。注意，分配符号名要与有效变量声明相对应！

6.9.3 PLC配置中的一般设置

在配置树的顶部选择条目 'PLC configuration'（根模块）。于是，'Settings'对话框显示在窗口的右部。可以激活下列选项：

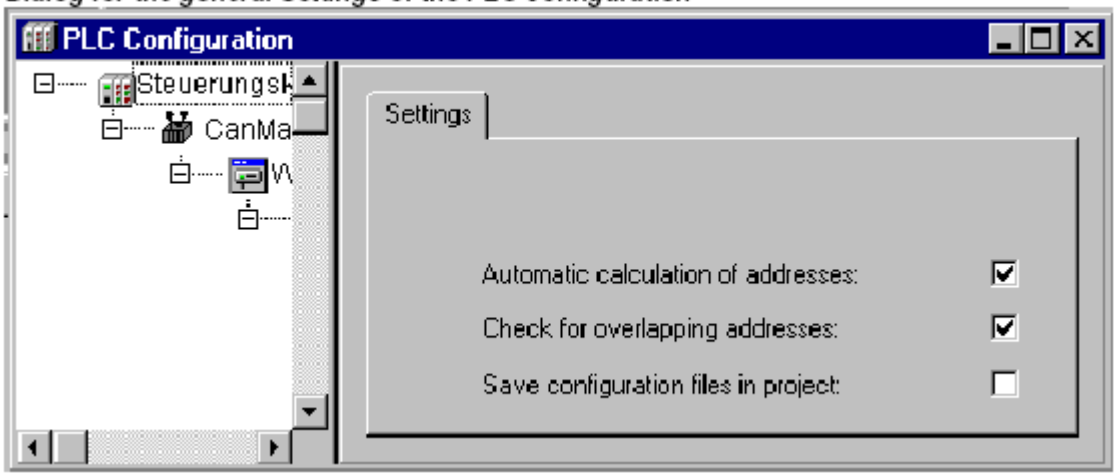
计算地址：每一个最近自动插入的模块被分配一个地址，这一地址产生于前面插入模块的地址加上该地址的大小。如果模块从配置中取消，则随后模块的地址会自动调整。当执行指令'Extras' 'Compute addresses'时，从被选节点（模块）开始的所有地址将被重新计算。

检查重复的地址：在编译时，检查工程的重复地址并显示相应的消息。

在工程中保存配置文件：包含在配置文件*.cfg和当前PLC配置支持的设备描述文件中的信息将被保存在工程中。

于是（如果配置文件没有定义标准配置，那么标准配置应当重新载入!），像用户建立的，配置将被保持在工程中，即使在重新打开工程时没有发现该配置文件。记住，如果没有激活该选项，那么，在这种情况下将失去全部工程特定的配置。通过用工程保存配置信息后，在目标改变时也能保持这些信息。但是，注意在这种情况下，新目标可能带来自己的要额外关照的配置文件。

Dialog for the general Settings of the PLC configuration



在配置文件中定义了PLC配置中偏址，地址的全局模式（直线地址/Id地址）。

重新计算模块地址，'Extras' 'Compute addresses'

如果在PLC配置编辑器的'Settings'对话框中激活选项"Calculate addresses"，那么指令Extras' 'Compute addresses'将重新计算模块地址。从配置树中当前选择的一个模块开始，所有模块的地址将重新计算。

添加配置文件：

在菜单'Extras'中使用这个指令来为工程的配置文件添加另一个文件。在工程选项，类别'Directories'，输入域'Configuration files'指定的目录路径中可以找到这些文件。

打开对话框**Select configuration file**，在这里可以为CAN- (*.eds,*. dcf)，Profibus- (gsd*.*)，配置文件(*.cfg)或者所有的文件(*.*)设置过滤器。选完想要的文件后，要检查一下，是否在配置文件定义的目录中找到了此文件。在这种情况下，将出现相应的消息并且不能添加文件。如果选择cfg-file，在这种情况下会打开一个对话框，在这里能得到要做什么的消息。

如果可以添加文件，所有工程定义的当前配置目录都出现在对话框**Select configuration directory**的列表中。选择点击Ok完成确认选择后，对话框将关闭，文件在PLC配置中马上可用。

返回标准配置，'Extras' 'Standard configuration'

指令'Extras' 'Standardconfiguration'可以用来恢复最初的PLC配置，这配置是通过配置文件*.cfg定义并且保存在工程中。

注意：在配置文件*.cfg中，可以定义每次在重新打开工程时要恢复标准配置。在这种情况下，所有用户完成的配置的修改将会丢失。

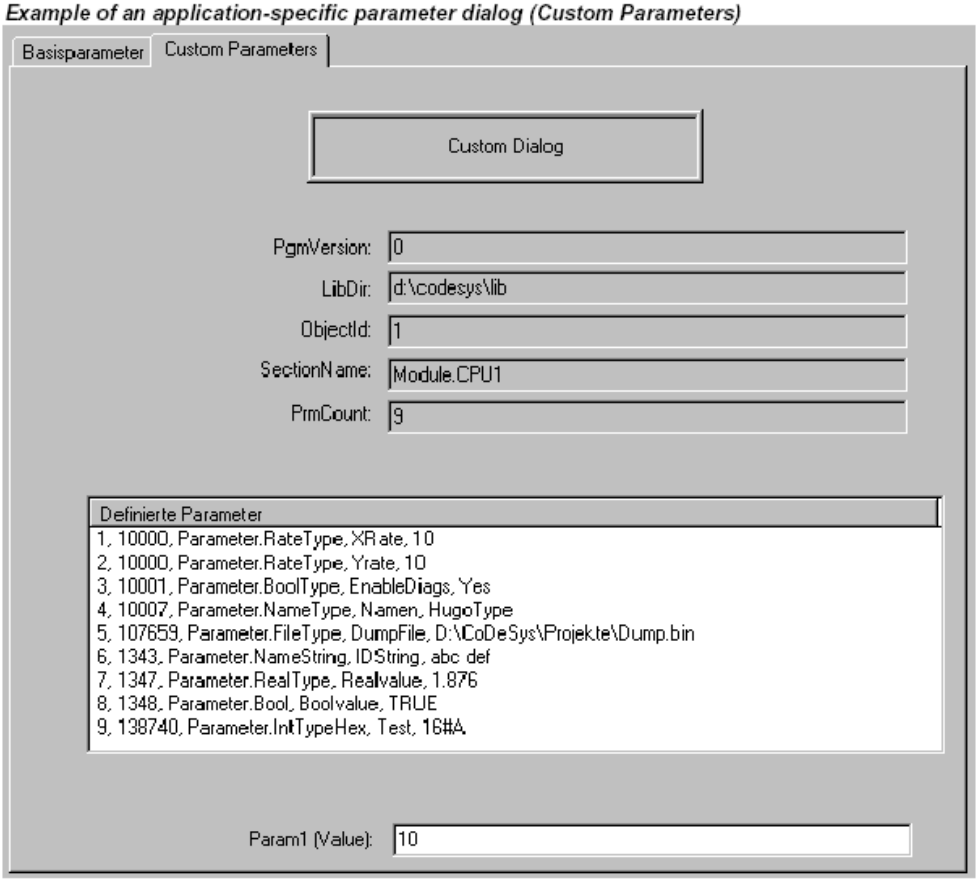
旧 PLC 配置的转变，'Extras' 'Convert'

如果打开一个包含用比V2.2更早的CoDeSys版本创建的PLC配置的工程，可以用菜单'Extras'中的这个指令。如果全部所需的配置文件有效，指令'Convert'将把现有的配置变为实际的PLC配置的格式。它打开一个对话框，询问要把配置转换成新格式吗?注意：不能撤销！你可以选择Yes 或 No。如果用Yes关闭对话框，配置编辑器将被关闭。重新打开，将会看新格式的配置。应当知道，转换后，旧的格式再也不能恢复！

6.9.4 定制特定的参数的对话框

通过应用专用的DLL对话框，可以扩展配置器的参数化能力。这个'Hook'-DLL一定在含有配置文件的目录中，并且可通过一个配置文件的入口把它链接到一个模块或者通道上。如果这样，那么，标准对话框'Module parameters'将被在DLL中定义的对话框替代。

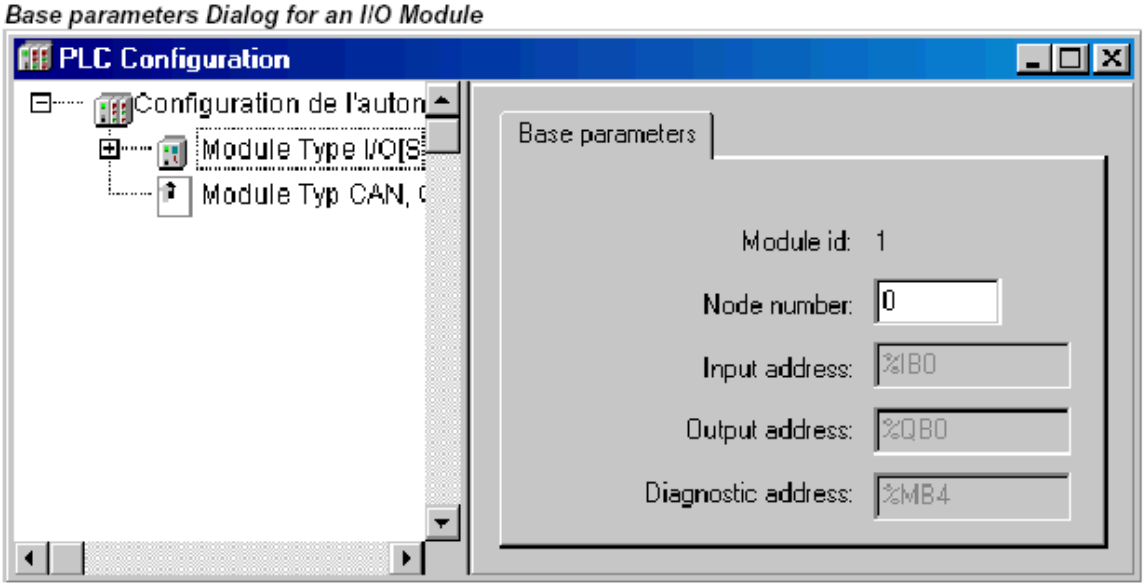
Example of an application-specific parameter dialog (Custom Parameters)



6.9.5 I/O模块配置

I/O模块的基础参数

Base parameters Dialog for an I/O Module



如果在配置树中选择I/O模块，那么显示带下列表项对话框'Base parameters'：

Module id: 模块id是全部配置内模块的唯一标志符。它由配置文件定义并且在配置编辑器中不可编辑。

Node number: 节点号由配置文件中此表项来定义，如果没有此表项，那么由配置结构中模块的位置来确定。

Input address, Output address, Diagnostic address: 分别用于输入输出地址和诊断数据存储地址。这些地址归属于模块。哪些地址已经被预订，哪些地址模式有效，地址能否在这里编辑。

Load module description: 如果这个选项不激活，那么在工程下载时将不考虑此模块。默认此选项时激活的，并且在配置文件*.cfg中定义配置对话框是否可见和可编辑。

PLC配置中的诊断：

模块诊断地址必须是标记地址。对正常的I/O模块如何处理诊断取决于专门的硬件配置。对总线系统，像CAN 或者rofibus DP，其诊断工作如下描述：从给出的诊断地址向前，储存了关于*GetBusState*结构的各种信息，*GetBusState*是制造商交付的相应库的一部分(e.g. 由3S - Smart Software Solutions提供的BusDiag.lib)。每当IEC任务已经读写了模块的处理数据的时候，要求所有的总线模块在每个周期中填写诊断结构。一旦总线系统里至少有一个模块生产错误，就能用*DiagGetState*功能块来读这个具体的诊断信息，*DiagGetState*功能块是上面提到的库的一部分。这功能只对总线主站有效，总线主站已经在CoDeSys PLC配置内设定了！

功能块**DiagGetState**的输入输出参数如下表所示。在**CoDeSys**工程中要为特定的总线模块定义一个此功能块的实例以便读取诊断信息：

DiagGetState的输入变量：

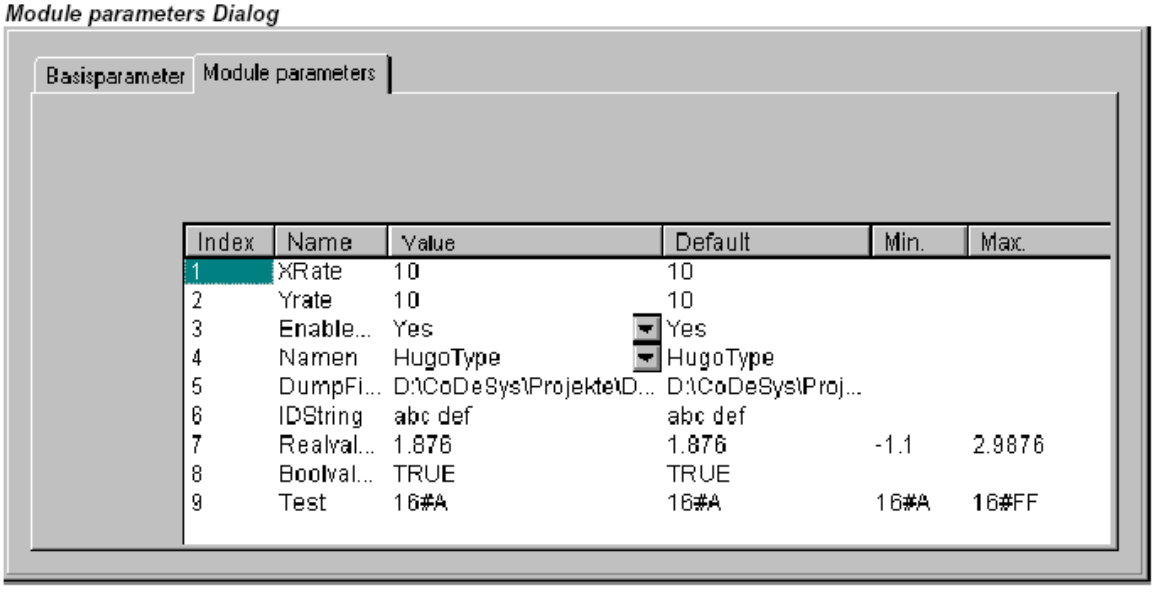
ENABLE:BOOL;	在ENABLE上升沿处，此功能块开始工作
DRIVERNAME:POINTER TO STRING;	总线驱动器名（名字地址）。要向它发送诊断请求，如果此值为0，那么向所有存在的驱动器传送诊断请求。
DEVICENUMBER:INT;	由驱动器管理的总线标识号。例如，Hilscher公司的驱动器可以处理5块卡，标识号从0开始
BUSMEMBERID:DWORD;	总线模块的总线/驱动器唯一专用标识号（例如，对于CANopen卡，它是节点ID，对于PB-DP卡，它是站地址）

DiagGetState的输出变量：

READY:BOOL;	TRUE：诊断请求的工作已经终止
STATE:INT;	如果READY＝TRUE，那么STATE得到下列定义此功能块实际状态的值。 -1：输入参数无效 （NDSTATE_INVALID_INPUTPARAM:INT） 0：功能块不工作 (NDSTATE_NOTENABLED:INT;) 1：功能块正在读诊断信息 (NDSTATE_GETDIAG_INFO:INT;) 2：诊断信息有效 (NDSTATE_DIAGINFO_AVAILABLE:INT;) 3：诊断信息无效 (NDSTATE_DIAGINFO_NOTAVAILABLE:INT;)
EXTENDEDINGFO:ARRAY[0..129] OF BYTE;	多达100个字节的制造商特定总线诊断数据，为每条总线保留1个字节，其中0-2位描述下列信息 0位：在PLC配置中存在总线。 位1：在总线系统中总线模块有效 位2：总线模块报告出错

I/O模块的模块参数/客户参数

Module parameters Dialog



在这个对话框中，显示设备文件给出的参数。只有’value’栏是可编辑的。

Index: 索引是连续的数字，它给模块参数编号。

Name: 参数名称

Value: 参数的值，可编辑

开始显示默认值。数值可以直接或者通过符号名名称设置。如果配置文件中的此条目没有设置为’Read Only’，那么它们是可编辑的。编辑方法：点击编辑域，选择滚动列表中的条目。如果值是文件名，可以通过

双击打开对话框'Open file'，浏览其他文件。

Default：参数默认值

Min：参数最小值（只有在不使用符号名的时候有效）

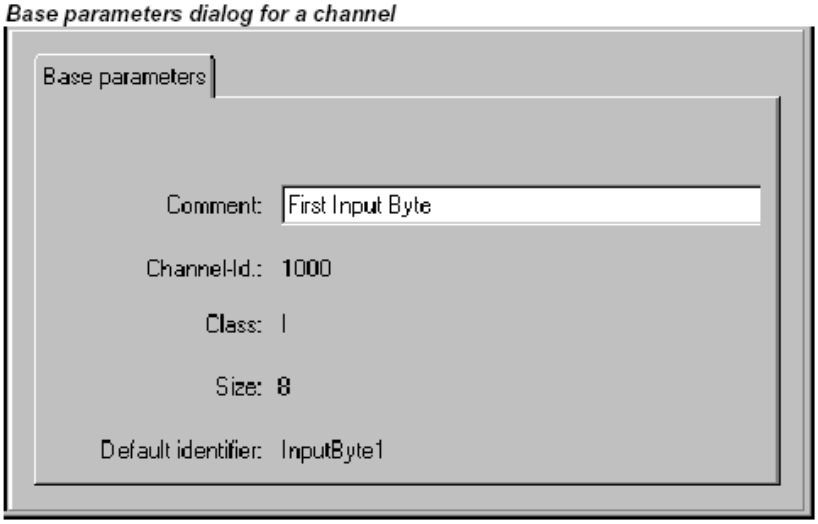
Max：参数最大值（只有在不使用符号名的时候有效）

tooltip可以给出关于当前标明的参数的其它信息。它是用户特别的对话框而不是模块参数对话框。事实上，由一个在配置文件中模块定义处的（Hook-DLL）入口来链接这样的对话框。

6.9.6 通道配置

通道基础参数

Base parameters dialog for a channel



ChannelId：通道的全局唯一标识号

Class：定义通道用作输入(I)，输出（Q），或者输入和输出（I&Q），或者可转换。如果通道是可转换的，那么这可以通过指令'Extras' 'Replace element'来完成。

Size：通道规模（用字节数表示）

Default identifier：通道的符号名

通道的名称在配置文件中定义。只有在它父模块的定义允许下，才能在配置树中编辑通道的名称。

Comment：关于通道的附加信息

在这个编辑域，可以插入或修改注释。

Address：如果在配置文件中通过一个条目来激活它，那么这个编辑域才可以使用。为通道插入要求的地址。

通道参数

相应于模块参数对话框，通道参数对话框用来显示和修改通道参数：**Index，Name，Value，Default，Min，Max**。这对话框可以用用户特定的对话框'Custom Parameters'代替。

位通道

当在配置文件中用条目CreateBitChannels=TRUE定义通道时，自动插入**位通道**。

6.9.7 Profibus模块的配置

CoDeSys支持相应于profibus DP标准的硬件配置。在profibus系统中，你会发现主从模块。每个从模块由它的主模块提供参数集，并且向主模块供应主模块要求的数据。

PROFIBUS DP系统由一个或者几个主模块和他们的从模块组成。首先必须配置模块，这样才可能在总线上交换数据。在总线系统初始化时，每个主模块确定由配置分配给它的从模块的参数。在运行的总线系统中，主模块向从模块发送数据或向从模块请求数据。

在CoDeSys中主从模块的配置基于硬件制造商绑到它们上的gsd文件。为此目的，要考虑所有储存在配置文件目录中的gsd文件。gsd文件描述的模块可以插入到配置树中，并且可以在那里编辑他们的参数。

在主模块下可以插入一个或者几个从模块。

如果在配置树中选择DP主模块，那么在配置的右部下列对话框可以使用：基本参数，DP参数，总线参数，模块参数。

如果选择了插在DP主模块下的DP从模块，那么可以使用下列对话框（根据配置文件中的定义）：基本参数，DP参数，输入／输出，用户参数，组，模块参数。根据在配置文件中的设置，对话框"DP Parameter"可以有另一个名称。

如果DP从模块被插入在主模块层上 ，那么为了配置可以使用下列对话框：基本参数，DP参数，输入／输出，模块参数。

DP主模块的基本参数

DP主模块的基本参数对话框和其它的模块一致：**模块ID，Node number，输入，输出和诊断地址**（见'I/O模块基本参数'）。

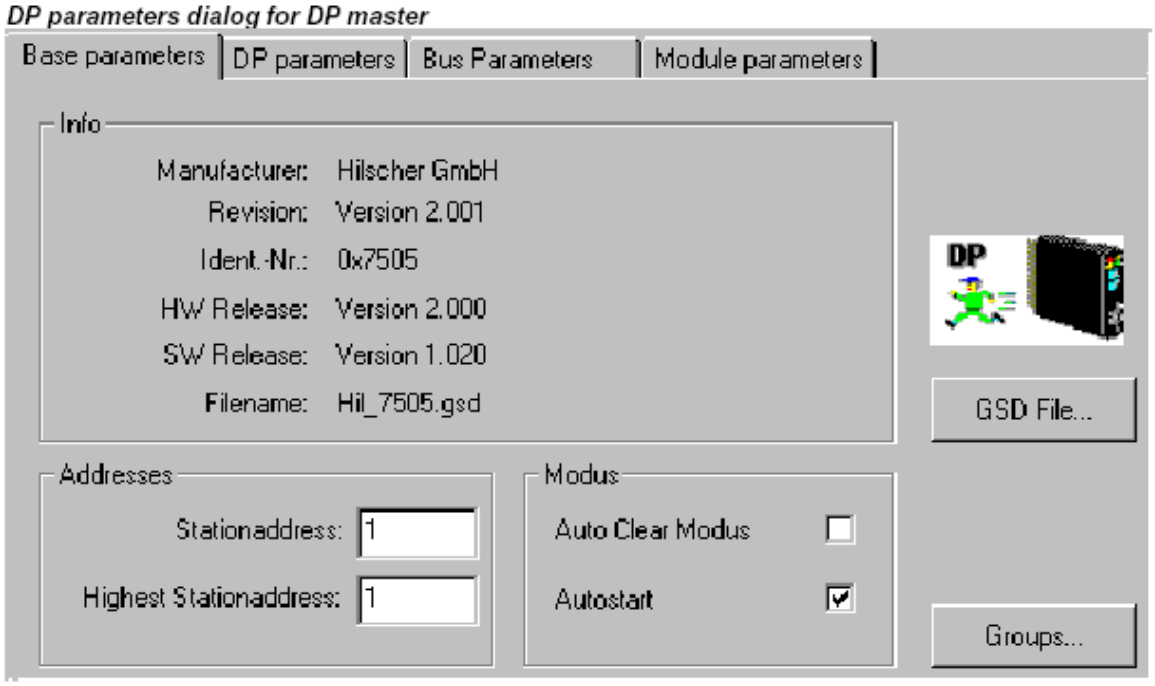
不自动修改地址：此选项只有在配置文件中定义了才有效。如果激活它，那么在地址重算时不考虑此模块。

DP主模块的模块参数

DP主模块的模块参数对话框和其它的模块一致：在配置文件中除了DP和总线参数外，分配给主模块的参数都在此对话框中显示并可编辑其参数值。（见6.6.5节' I/O 模块的模块参数'）。

DP主模块的DP参数

这对话框显示从DP主模块设备文件中提取的下列参数（对话框可以有不同的名称，这名称是在配置文件中定义的）：



info　　Manufacturer（生产商），Revision（GSD修订本），IdentNum（ID标识号），HW Release(硬件版本)和SW Release （软件版本），Filename （GSD-文件名）

Module name 可以在这位置编辑设置模块名。

Addresses Station address（站地址）：可容许的范围从0到126。向每个在总线线路上新插入的设备自动提供下一个高地址（注意：地址126是默认的DP从模块地址）。可以手动输入地址；但要检测地址的重叠。

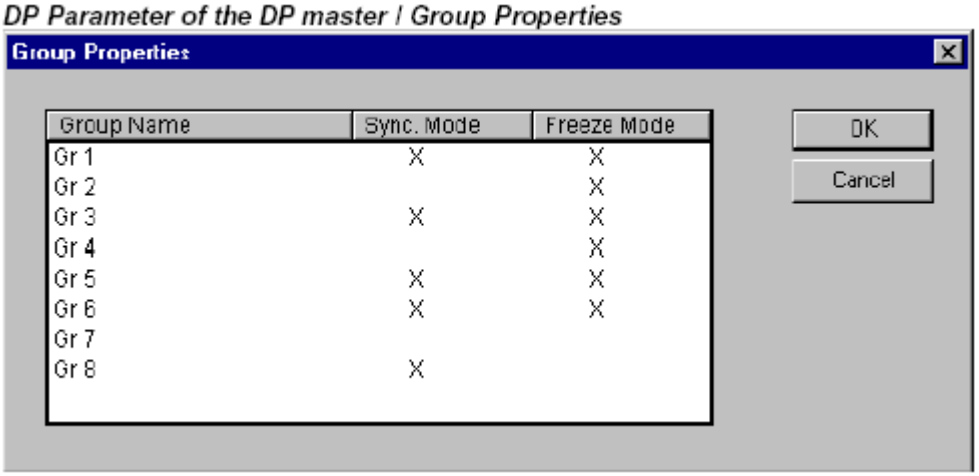
Higest address（最高位地址）：显示在总线上分配的最高站地址(HSA)。为了使GAP范围变窄，可以输入较低的地址（换言之，搜索新有效设备在此内执行）。

附属于一个设备的GSD文件可以被打开并且可以用**GSD File**按钮来检查。

Groups按钮打开组属性（'Group properties'）对话框。组属性附属于分配给主模块的从模块。最多可以设立八组。对每一组，确定它是否用冻结（**Freeze**）方式和/或同步（**Sync**）方式进行操作。通过把从模块（见'Properties of the DP slave' 'Group assignment'）分配给各种组，用全局控制指令可以与主模块的数据交换同步进行。用Freeze指令，主模块指示一个从模块或者一个组，在瞬时状态，冻结输入并且在随后的数据交换中传送这些数据。用Sync指令只是各从模块在下一个同步指令下同步切换到输出，在第一个同步命令下从主模块接收到的所有数据。

为切换一个组的Freeze 和 Sync选项开/关，请在表格的适当的位置点击鼠标左键，来安放或删去“X”，或者用鼠标右键通过上下文菜单激活或关闭此选项。另外，可以在这里编辑组的名称。

DP Parameter of the DP master / Group Properties

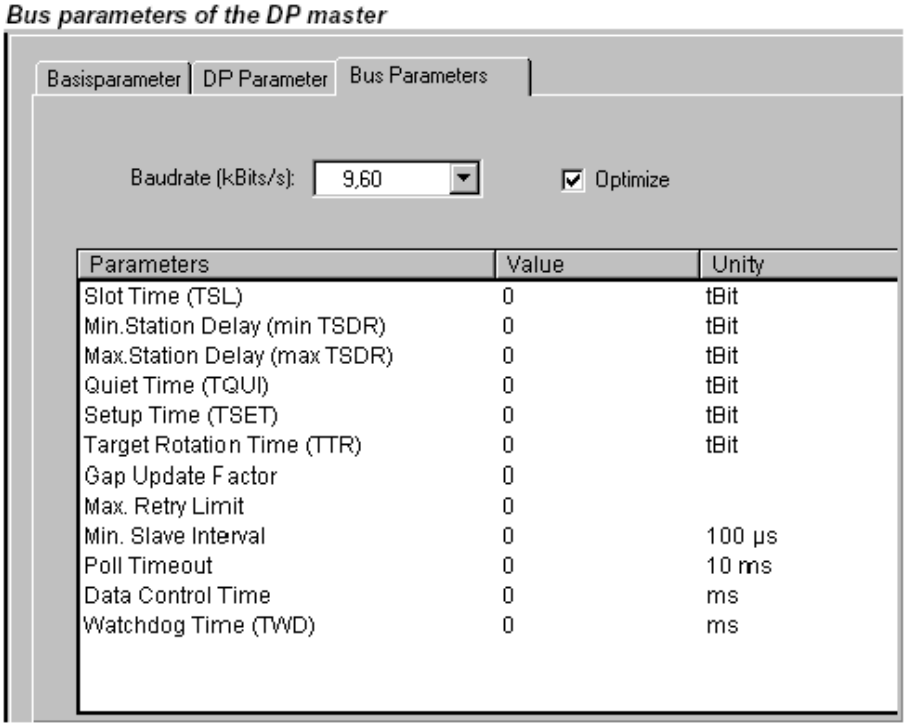


DP主模块的总线参数

总线参数描述通信的时间安排。如果激活选项**Optimize**，那么将根据用户设置的**波特率（Baudrate）**和GSD文件给出的设置自动计算参数值。

注意：自动计算的值只是近似值！

Bus parameters of the DP master



所有参数也可以手工编辑。

Baudrate（波特率） 此表项已经提供了在GSD文件中有效的选择值，但是，只可以输入所有从模块支持的传输速率。

Optimize（最优化） 如果激活这选项，用GSD文件中的规范来最优化在'Bus parameters'对话框中制作的表项；如果关闭此选项，那么就可以编辑这些值。

注意：自动计算的值仅是粗略的近似值。

Slot Time（时间槽） 主模块等待的最大时间：从发送请求的消息至接收到从模块回答消息的第一个字符。

Min.Station Delay min TSDR（单位：tbit）（最小站延迟）：在总线上的一个站可以答复的最小反应时间(min. 11 tBit)

Max.Station Delay max TSDR（单位：tbit）（最大站延迟）：从模块必须回答的最大的时间间隔。

Quiet Time TQUI（单位：tbit）（空闲时间）：必须在把NRZ（不归零）信号转换成其它编码期间里考虑空闲时间（转发器的转换时间）

Target Rotation Time TTR（单位：tbit）（目标循环时间）：设定令牌周期时间；设计主模块快接收到令牌的时间间隔。来自于总线上所有主模块的令牌停顿时间总和。

Gap Update Factor 更新因子G ：为另一个新插入的有效站点找到主模块GAP（从它自己的总线地址到下一个有效站点地址的地址范围）的总线周期数。

Max. Retry Limit （最大重复请求数） 当没有从从模块收到有效答复的时候，主模块试图重复请求的最大数量。

Min. Slave Interval 两个总线周期之间的时间，在这段时间里从模块可以处理来自主模块的请求（时间基100us）。在这里输入的值必须与从模块GSD文件的相关规范进行核对。

Poll Timeout （轮询的最大时间）在这时间后请求者（Class2DP主模块）必须检索主模块间通信的大富（时间基1 ms）。

Data Control Time（数据控制时间） 主模块对其从模块报告它的状态的时间。同时，主模块监控在这期间是否与从模块发生过至少一次数据交换，并且更新Data_Transfer_List。

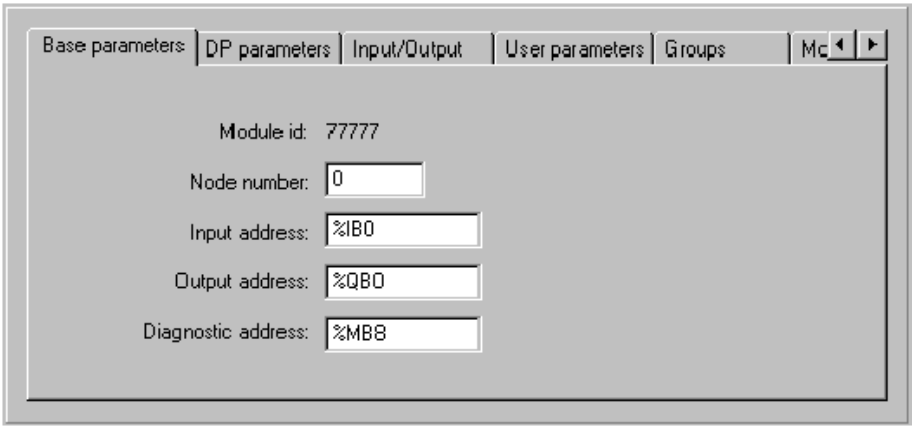
Watchdog Time （监视器时间） 访问监视器的时间值。当前不支持设置(固定设置成400 ms)。

DP slave的基本参数

DP-Slaves基本参数对话框对应的其它模块的基本参数：Module id， Node number， Input address，Output address 和 Diagnostic address。

不自动修改地址：此选项只有在配置文件中定义了才有效。如果激活它，那么在地址重算时不考虑此模块。

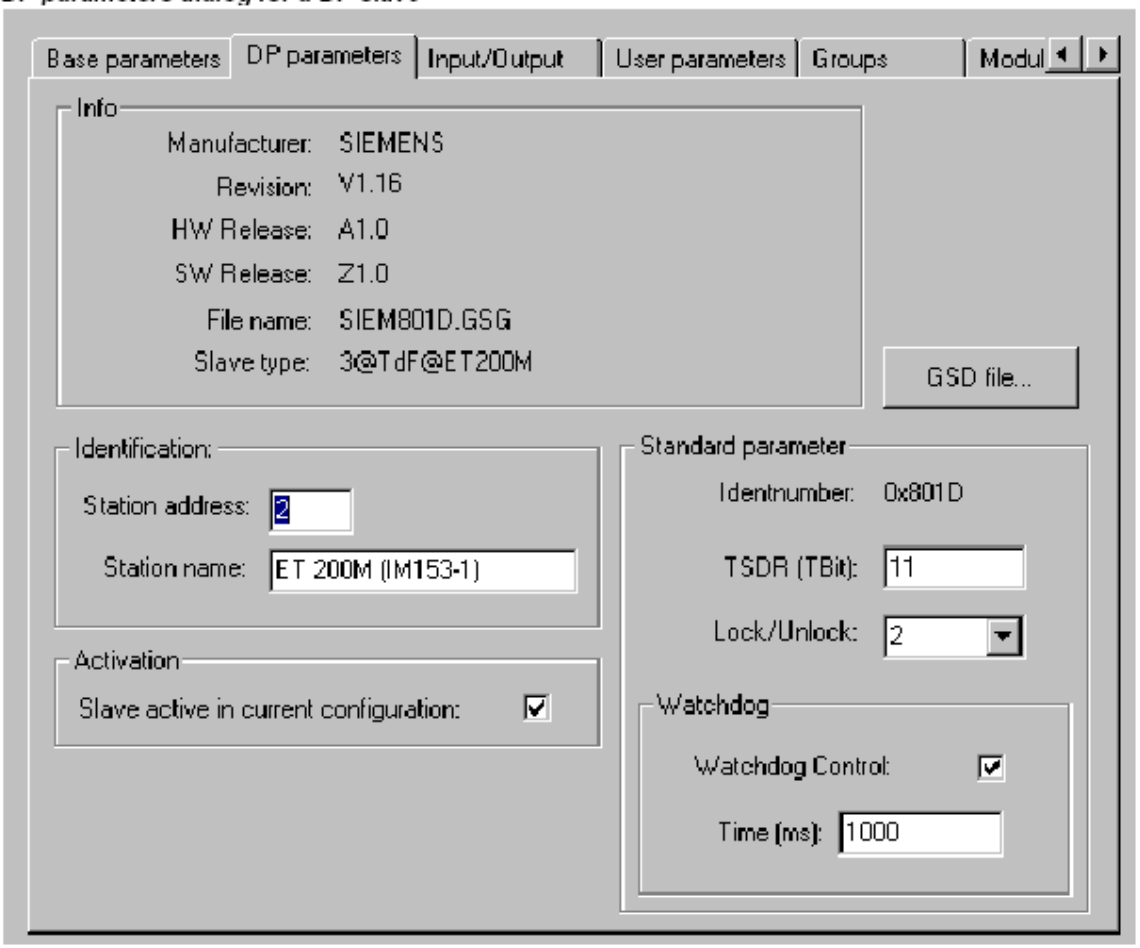
Base parameter dialog for a DP slave



DP slave的DP参数

这对话框显示从 DP slave的设备文件中取出的下列参数（对话框可以有不同的名称，这名称是在配置文件中定义的）：

DP parameters dialog for a DP slave



info（信息），Manufactureer（生产商），Revision（GSD修订本），HW Release（硬件版本）和SW Release(软件版本)，Filename（GSD－文件名），Slavetype（从模块类型）

Standard parameter（标准参数）

Identnumber：PNO分配给此设备类型的唯一的ID号。在DP slave和相应的GSD 文件之间允许明确关联。

TSDR (Tbit*)：站点延迟响应时间，反应时间，允许从模块对主模块作出响应的最早时间。(最少. 11 TBit) * Tbit：是在PROFIBUS上传送一位的时间单位，是传输速率的倒数；例如，在12MBaud下1 TBit=1/12.000.000 Bit/sec=83ns

Lock/Unlock：从模块被锁到定其他的主模块或者解除锁定：

0：可以写最小的TSDR和从模块的特定参数 ；

1：从模块从主模块释放

2：接受从模块锁定到主模块所有的参数

3：从模块从主模块释放

Identification **Station address （站地址）**：(见 'DP主模块属性'), **Station name**（站名，与设备名一致，可编辑）

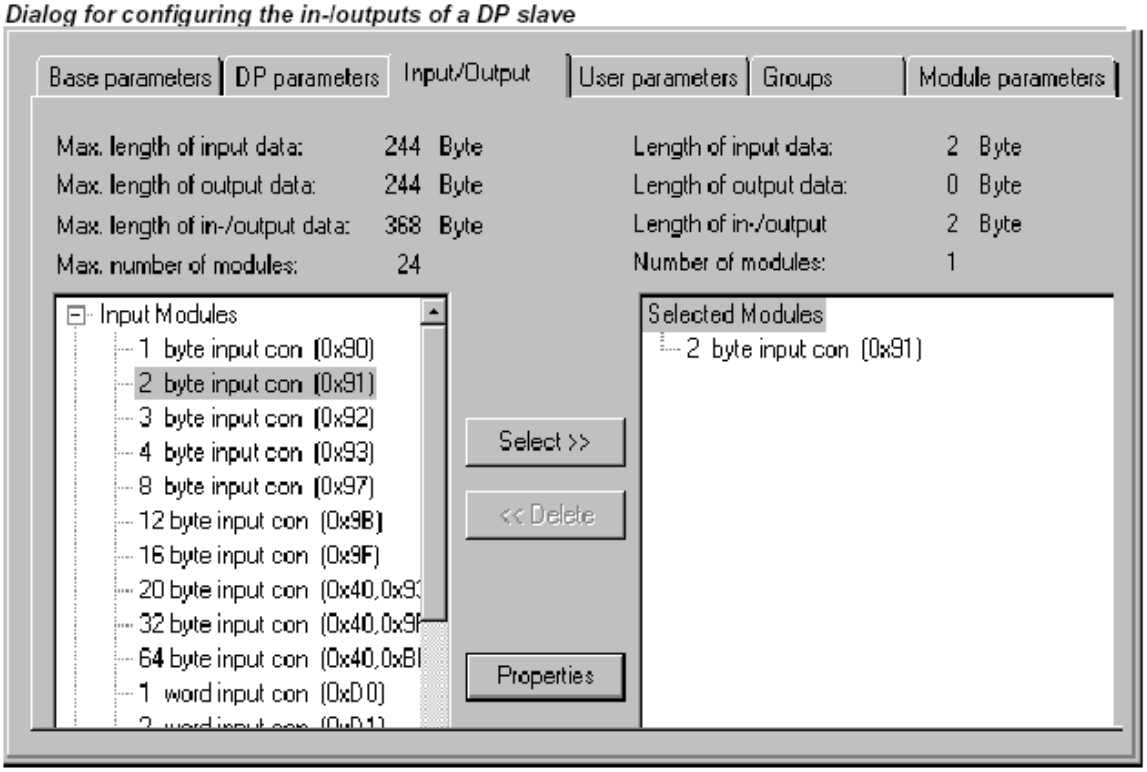
Activation 在当前配置中激活/关闭从模块。如果没有选择激活,配置数据将在下载时被传送到耦合处，在总线上不发生数据交换。

Watchdog 如果把**Watchdog Control**设定为激活，那么应用输入的监视器时间(访问监视器的时间基10 ms)。如果从模块在这段时间内没有被主模块访问，那么它恢复到初始状态。

你可以通过**GSD-File**按钮检查相应的GSD文件。

DP slave的输入／输出

Dialog for configuring the in-/outputs of a DP slave



完成DP slave的配置的方式取决于它是“标准化的”还是“非标准化的”，“固定的”从模块。

对标准的从模块的选择象以下描述的那样来做：

在列表中，在对话框的左边的列表中选要求的输入或输出模块，点击按钮**Select**把它放入右边的窗口中。在窗口中的错误输入可以通过按钮**Delete**纠正。插入的模块将立即显示在配置树中。如果选择他们，那么相

应的对话框Profibus Modul将可以用于显示输入，输出和模块的诊断地址。如果选择一个此模块已插入的通道，将打开对话框Profibus Channel，显示通道地址。对这两个对话框的任意一个，可以在配置文件中通过设置定义不同的名称。

因为在GSD-file中指定的最大数据长度（最大输入数据长度，最大输出数据长度，最大输入/输出数据长度）和模块的最大的数量（Max number of modules）必须遵守可以用两个列表显示这些的信息。左边的块显示设备的最大可能值，右边的块显示从总结配置中选择产生的值。如果超过最大值，发出错误信息。

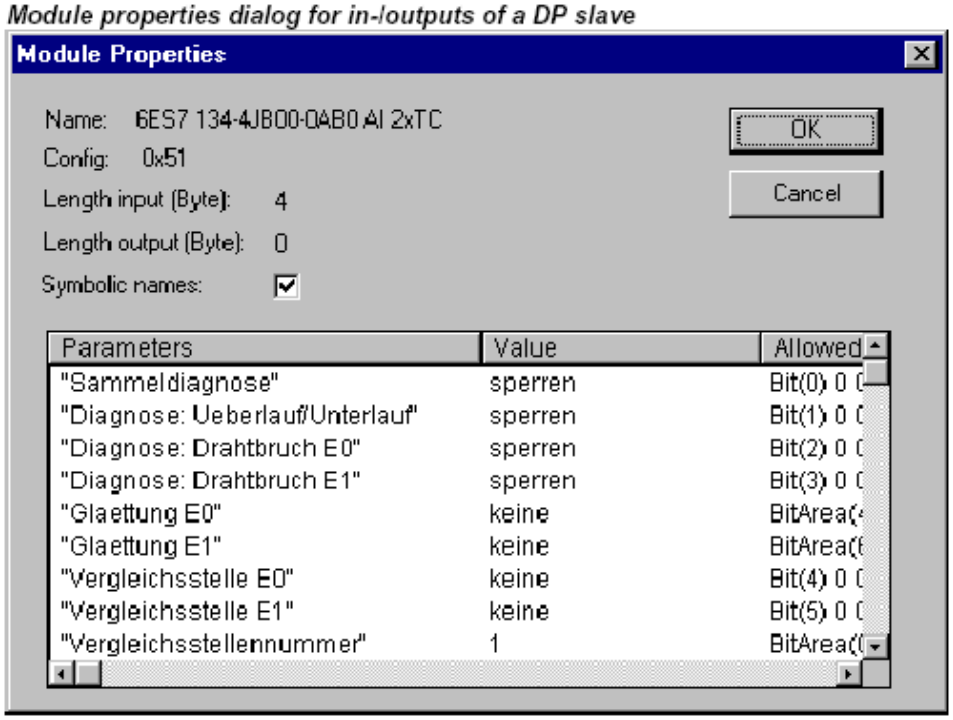
对话框左边窗口列出所有的在从模块 GSD-file中有效的输入输出模块，同时在右边的窗口包含为设备选择的当前的输入输出配置。

如果这是一个标准化从模块（备有各种的I/O模块的设备），那么选择的过程如下：在左边列表，通过点击鼠标来选择想要的输入或输出模块，使用>>按钮复制到右边窗口。错误的输入可以通过在右边窗口选择不想要的模块点击Delete按钮来纠正。

这种选择对非标准的从模块来说是不可能。这将直接强制关闭右边窗口中的输入输出显示。不想要的模块可以通过选择使用Delete来取消。

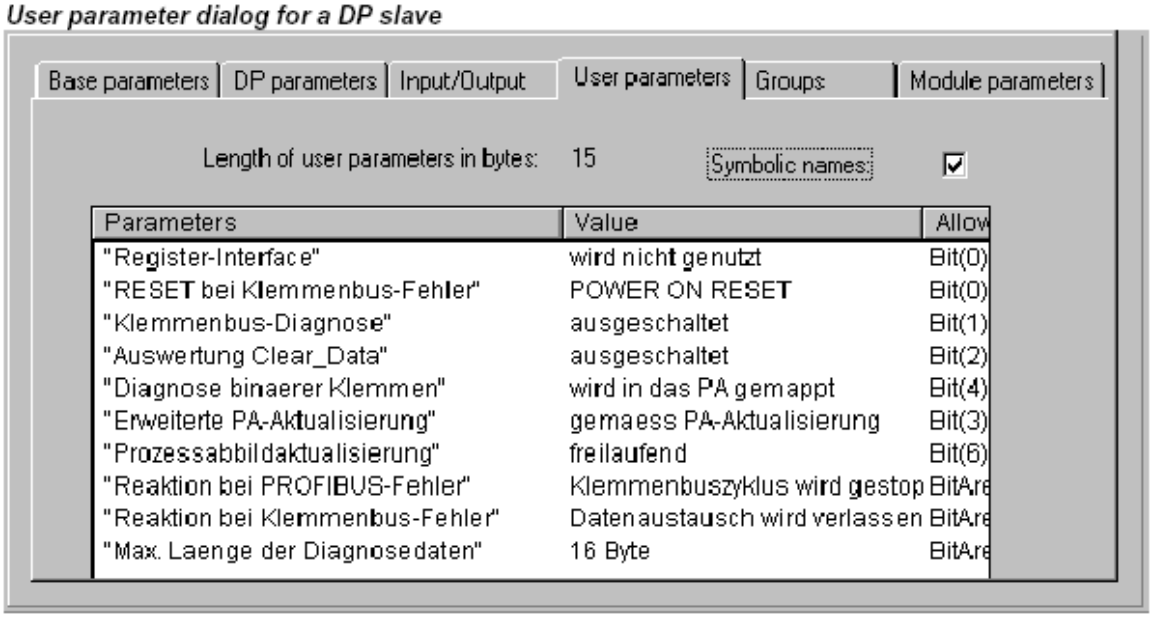
Properties按钮为在左或右列选择当前的输入输出模块打开'Module properties'对话框。它显示Name，Config（根据PROFIBUS标准描述模块编码）Length，input和output。如果GSD文件里的模块描述除了标准设定外还包括特定的参数，那么在这里列出了它们的值和值的范围。如果激活Symbolic names选项，可以使用符号名。

Module properties dialog for in-/outputs of a DP slave



DP slave的用户参数

User parameter dialog for a DP slave

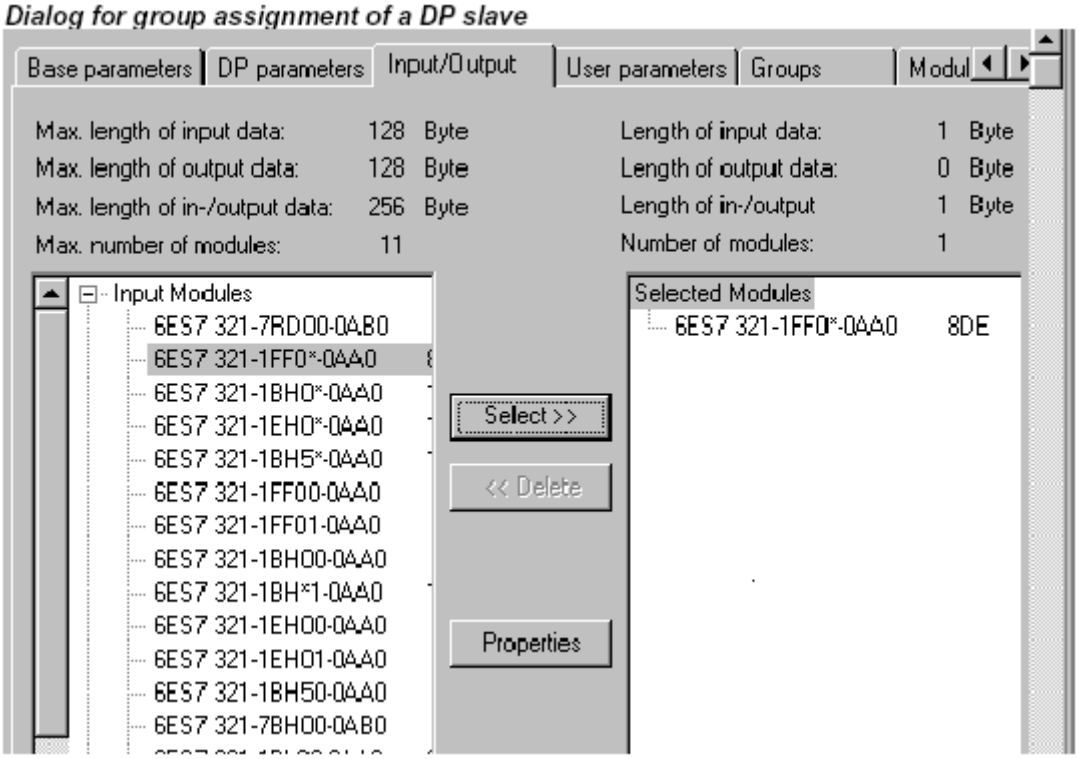


这里列出了定义在GSD-file中的DP slave的各种扩展参数。Parameters栏显示参数的名称。输入**Value**栏的参数值可以通过双击或者鼠标右键来改变。另外，**Value range**值范围是指定的。

如果参数符号名在GSD-file中也被指定，那么可以激活**Symbolic names**选项，以致可以用这些名称显示它们的值。还在表格上给出**Length of user parameters（用户参数长度）**信息。

DP slave 的组分配

Dialog for group assignment of a DP slave



这对话框是用来把从模块分配给八个可能的组中的一个或多个。另一方面，普遍可用的组特性(**Sync. Mode** 和 / 或 **Freeze Mode**)在主模块特性的配置里被定义(见 'Properties of the DP master, Group properties')。这对话框也可以通过**Global Group Properties**按钮来得到。

分配的从模块的组用加号来标明。通过在**Group Membership**栏里选择组的名称并且用鼠标右键点击'Add slave to group' 或'Remove slave from group'，或者用鼠标再次点击组名的左边来完成将从模块分配给一个组或

把从模块从一个组中删去。
从模块设备只能分配给那些支持它的特性组。有关的每个从模块特性（sync.模式/Freeze模式）显示在表格上面。检查驱动支持的模式。

DP slave 的模块参数

DP slave的模数参数对话框与其它的模块参数一致（见6.6.5节）。除在配置文件中的DP和用户参数外，分配给slave的参数在这里显示，在标准情况下可以编辑它们的值。

在 Profibus 从模块操作中的 DP Slave 特性

如果以从模块的模式运行Profibus，那么从模块设备被插入到配置树的主模块层中.在下列对话框中完成配置（见上述各节所讲）：

基本参数
DP参数
模块参数
输入 / 输出

6.9.8 CAN 模块的配置

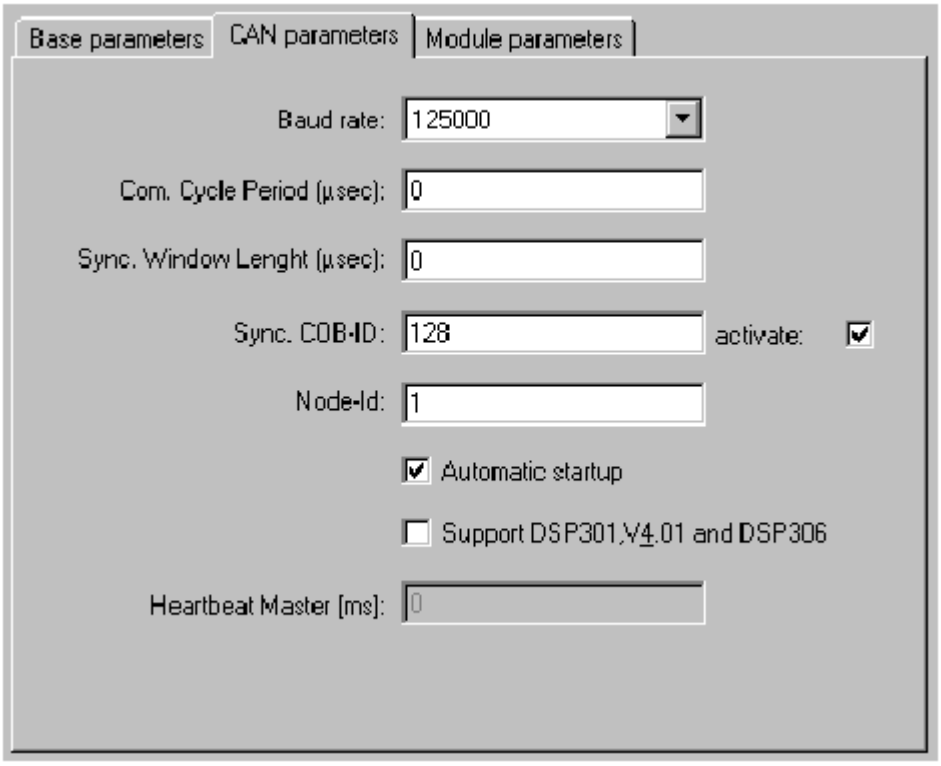
CoDeSys支持符合CANopen Draft Standard 301硬件配置。其配置是与硬件相关的。所有EDS（电子数据单）代表各个贮存在配置文件目录中的DCF（设备配置文件），在CoDeSys启动时，它们可以在配置中被集成，编辑和显示。在EDS文件中描述了CAN模块的配置选择，如果增减一个DCF文件描述的模块，只能修改IEC地址，因为在CAN配置中已经完成了该模块的配置。

CAN 主模块的基本参数

关于Modul-Id, input-/output addresses，Diagnostic address的信息见6.6.5节，”I/O模块的基本参数”。

CAN 主模块的CAN参数

CAN Parameter Dialog for a CAN-Master



CAN总线的发送特性可以在插入模块后直接设置或者用指令’Extras’ ‘Properties’来调用。

使用选择选项，设置发送所需的**Baud rate（波特率）**。

在PDO’s（处理数据对象），区分同步和异步发送方式是不同的（见PDO properties）。同步消息是在由**Communication Cycle Period**通信周期给出的间隔微秒中，用唯一的同步号. **COB-ID**（通信对象ID)来发送的。在同步消息进入时间窗口(用微秒表示的同步窗口长度)后，直接发送同步的PDO。如果公共循环周期和同步窗长度域为0，那么将不发送同步消息。

activate: 只有在激活选项的时候，同步信息才在主模块和从模块之间传送。

Node-Id：用于唯一地识别CAN模块并且模块本身在1和127之间的设置号一致。Id必须以十进制数字输入（不要和’Node number’节点号混淆！）

如果激活了**Automatic start** 选项，那么在进行下载时和控制器启动时，CAN总线自动初始化和启动。如果没激活这个选项，在工程中必启动CAN总线。

如果激活了选项**Support DSP301,V3.01**和**DSP306**，那么将支持标准CAN从模块和有关标准DSP301 V3.01和DSP306的某些扩展。例如，可以调节律动测试（Heartbeat）的幅值（HeartbeatMaster[MS]:）。Heartbeats是一种选择防护机制，和节点防护的功能相比，它可以由主模块和从模块执行。通常的配置是由主模块向从模块发送heartbeats。

CAN主模块的模块参数

CAN 主模块的模块参数对话框和其它的模块一样（见6.6.5节）：在配置文件中分配给主模块的参数在这里显示和编辑。

CAN模块的基本参数

关于**Module id, Input/Output addresses, Diagnostic address** 的信息，见6.6.5节，’I/O模块的基础参数’。

对于Output和Input addresses，输入在工程中可寻址的PDO’s的IEC地址，从模块的角度确定输入输出的方向

必须对CAN模块的**diagnostic address**给标记地址。它像CAN主模块描述的那样工作。

CAN模块的CAN参数和CAN主模块的参数不同，它不起主控作用（总线的全局监控）。

General：

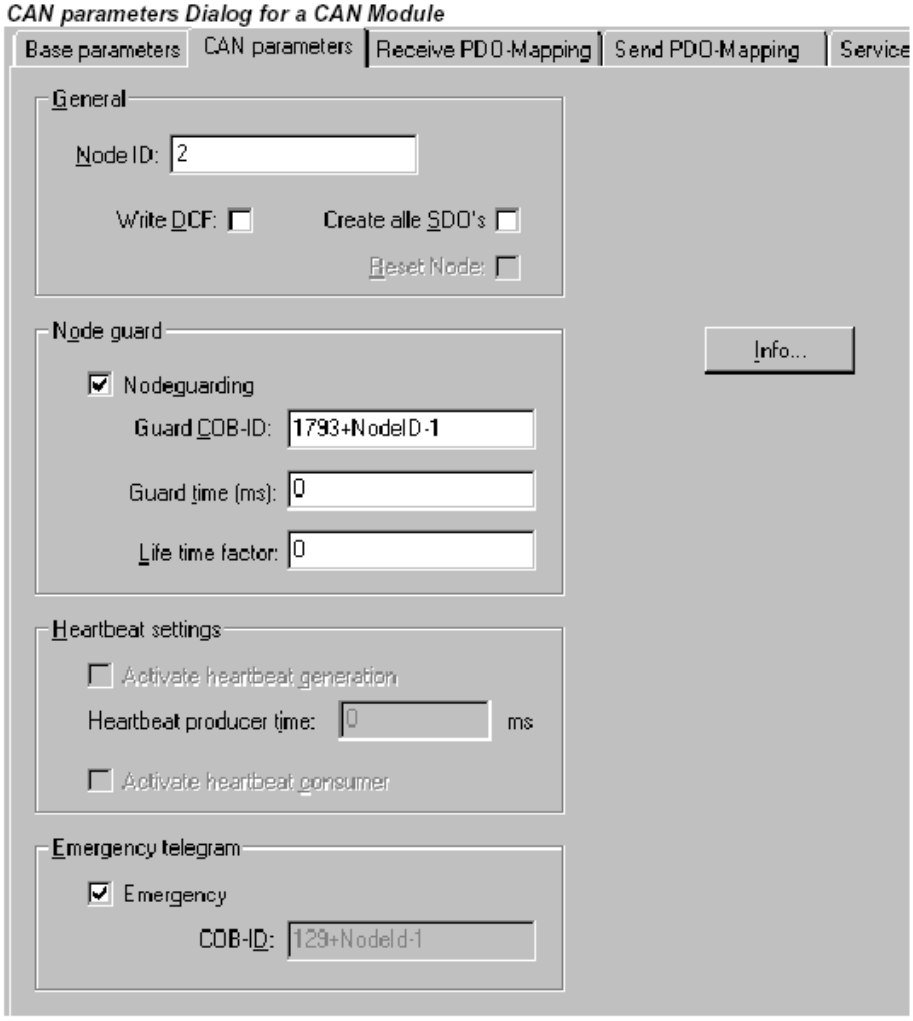
Node-Id 用于唯一地识别CAN模块，在1和127之间设置适当的数字。Id必须以十进制数字输入。

如果激活**writeDCF**，在把一个EDS文件插入到为编译文件定义的目录中后就创建出一个DCF文件，编译文件名由FDS文件名后附加上节点ID构成。

如果激活选项**Create all SDO’s**，那么对所有的对象将创建SDO’s（不仅对那些修改过的对象）。

如果激活选项**Reset node**，那么在下载配置前将从模块复位。

CAN parameters Dialog for a CAN Module



节点保护

如果激活选项**Nodeguarding**，根据**保护时间**（毫秒）设置的间隔，把消息发送到模块。如果此模块没有以给定的**Guard COB-ID**(通信对西那嘎标识号)发送一个消息，它将收到“过时”状态。一旦过时的数目达到**Life Time Factor**后，模块将马上收到"not OK"状态。这个状态将被储备在诊断地址中。如果没有定义Guard Time 和 Life Time Factor，将不产生模块监控。

律动设置：（替代节点保护）

如果激活选项**Activate Heartbeat Producer**，此模块将根据在**Heartbeat Producer Time**中定义的间隔毫秒发送率动。

如果激活选项**Activate Heartbeat Consumer**，此模块将侦听主模块发出的律动。一旦不再收到律动后，此模块将马上切断I/Os。

紧急报文：

当有内在错误时，模块用唯一的**COB-Id**发送**紧急**消息。这些消息储存在诊断地址中。

来自相应模块制造商的EDS或DCF文件的 “FileInfo”和“DeviceInfo”可用Info钮打开。

为标准的从模块选择CAN模块

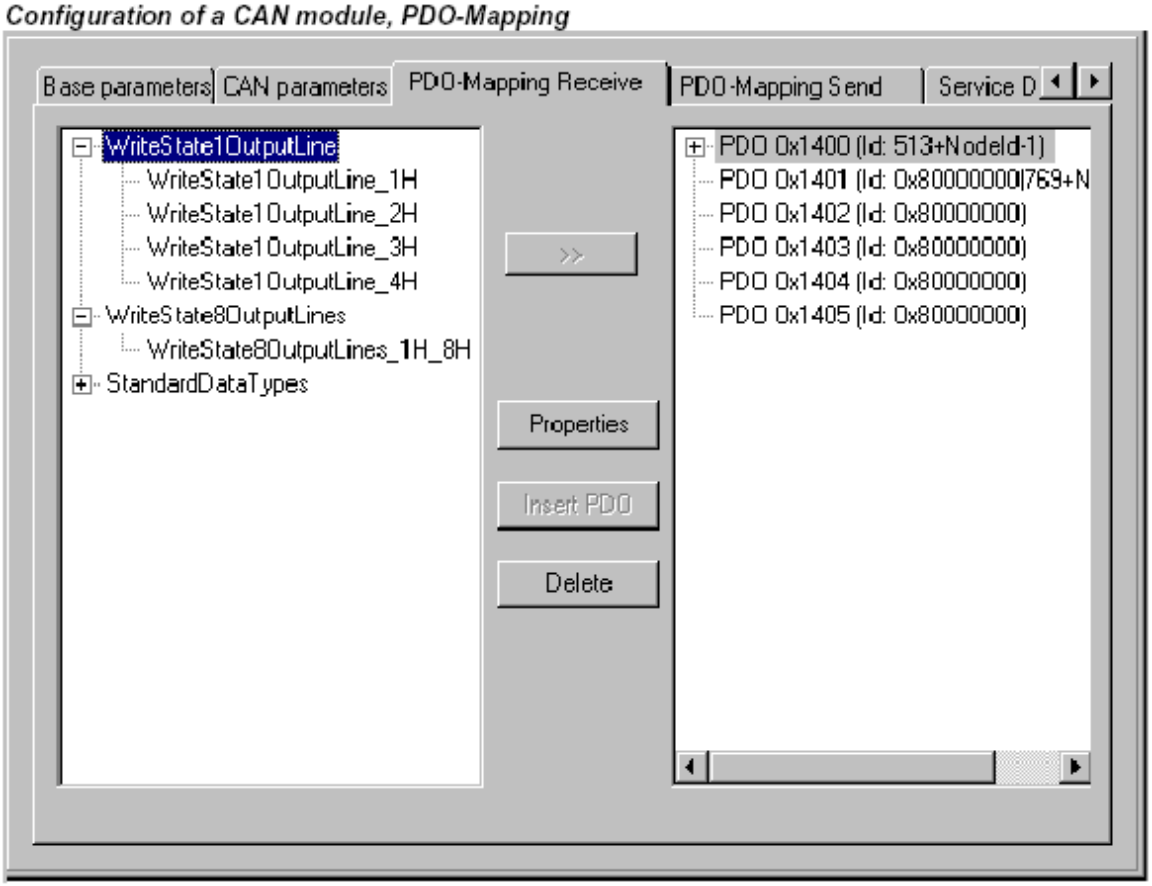
在左栏（**Available modules**）你可发现所有可采用的从模块。标明想要的模块并且用按钮**Add** 和 **Remove** 在右栏(**Selected Modules**)创建一个选择。其PDO-和SDO选择将自动更新。

CAN模块的PDO（数据处理对象）映射

在配置对话框中，CAN模块选项**Receive PDO mapping**和**Send PDO mapping**允许改变在EDS文件中描述的模块映射。

在EDS文件中所有的可映射对象位于左边，并且可以用“>>”按钮把它们添加到右边PDO's中 （处理数据对象），或者用以在PDO中创建空的空间再次取消（用**Remove**按钮）。可插入StandardDataTypes。

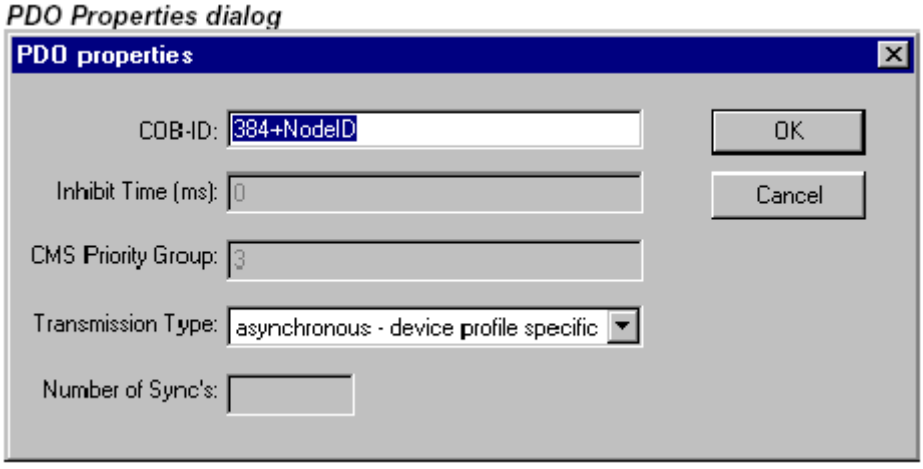
Configuration of a CAN module, PDO-Mapping



按钮**Insert Element**用来创建更多的PDO's并且给他们添加适当的对象。在插入的PDO上面完成对IEC地址的输入或输出的分配。当离开对话框的时候，在控制系统配置中建立的设置将变得可见。可以为各个对象可以提供符号名。

可以用**Properties**编辑PDO's的标准设置属性。

PDO Properties dialog



每个PDO消息需要唯一的**COB-Id**（通信对象标识符）。

对于模块不支持的选项或者其值不能改变，那么该域显示在灰色框里并且不能编辑。

Inhibit Time是来自这个PDO的两个消息之间的最小时间。这用来防止在PDO的值改变时频繁发送它。

在CAN发送期间，不能改变**CMS Priority Group**（优先级组）描述了PDOs的相关重要性。显示的值从0到7，

0是最高值。

Transmission Type为这模块提供了可能的发送方式选择：

acyclic - synchronous：同步、非周期性发送PDO。

cyclic - synchronous: 同步周期发送PDO，**同步号**给出了同步消息的数量，这些数字存在于PDO的两个传输之间。

synchronous - RTR only：在每个同步消息之后，PDO将被更新但不发送。只有在有明确请求（远程发送请求）时，才发送PDO。

asynchronous - RTR only：只有在有明确请求（远程发送请求）时，才更新和传输发送PDO。

asynchronous - device profile specific and asynchronous - manufacturer specific：只有在特定的事件产生时，才发送PDO。

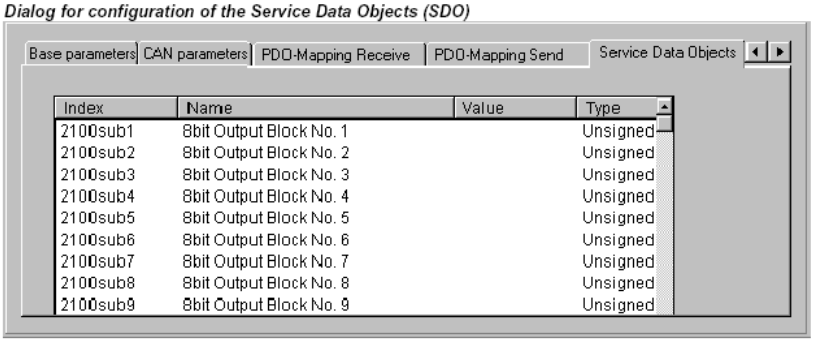
Number of Syncs: 如果设置了周期发送，在这里输入同步消息的数量（见在CAN参数对话框中的'Com. Cycle period'），在两次发送PDO之间发送它。

Event-Time: 如果设置了相应的发送类型，在这里输入两次发送之间的间隔（毫秒）。

服务数据对象 SDO

在这里有一个在EDS或DCF文件中所有对象的列表，这些对象在索引0x2000 到 0x9FFF之间的区域并且表明可写。

Dialog for configuration of the Service Data Objects (SDO)



对每个对象显示的属性有**Index（索引）, Name（名）, Value（值）, Type（类型）**和 **Default（默认值）**。值可以改变。标明一个值并且按<Space bar>。做完后，变化用<Enter>确定新的值或者用<Escape>键拒绝新的值。

在CAN bus初始化时候，设置值以SDO(Service Data Object)的形式被传送到CAN modules。

注意：在CANopen和IEC-61131之间所有矛盾的数据类型在**CoDeSys** 中将被下一个更大的IEC-61131数据类型替代。

6.9.9 Can驱动器的配置

用CoDeSys可编程的PLC在CAN网络中可以用于CANopen从模块(CANopen-Node, 以下称的"CanDevice")。

为了这个目的，在CoDeSys PLC配置器中可以配置PLC并可把此配置保存在EDS-file里。EDS-file (驱动文件)稍后可用于任一CANopen主模块的配置中。

用来在CoDeSys PLC Configurator中建立CanDevice的前提条件：

1. 下列库

3S_CanDrv.lib

3S_CanOpenManager.lib

3S_CanOpenDevice.lib

必须包括在CoDeSys工程中。需要他们作为一个CANopen设备运行PLC。

2. 在的基础配置文件中（*.cfg），必须插入CanDevice的相应条目。只有这样才能在PLC配置编辑器中，可在下列三个配置对话框中添加和参数化子元素‘CanDevice’，：

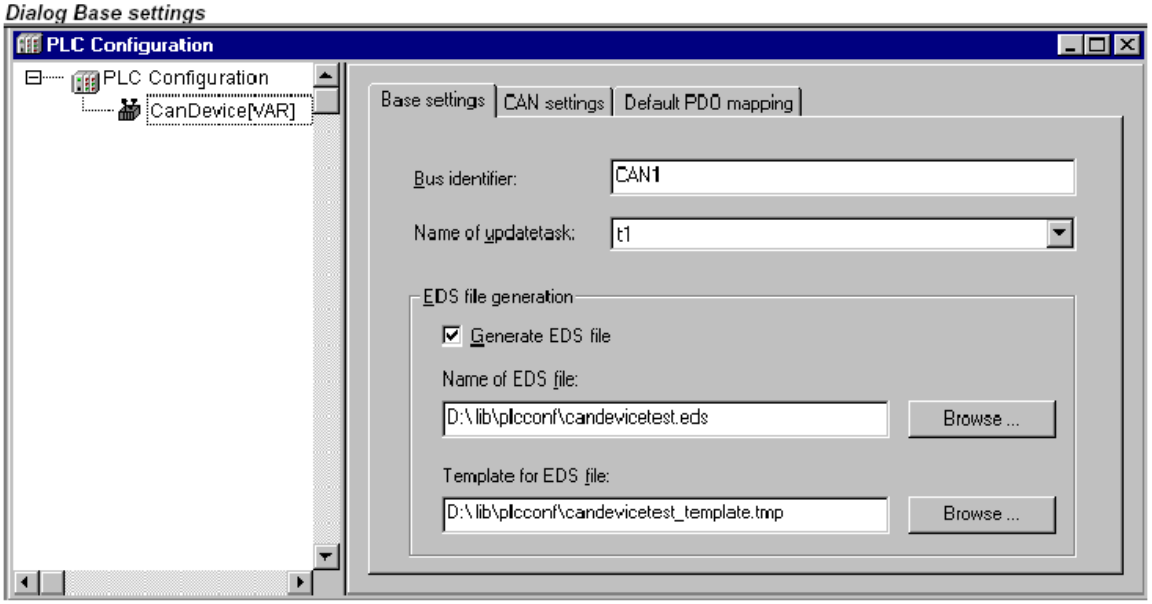
Base settings

CAN settings

Default PDO mapping

CanDevice的基本设置

Dialog Base settings



Bus identifier: 当前不能用。

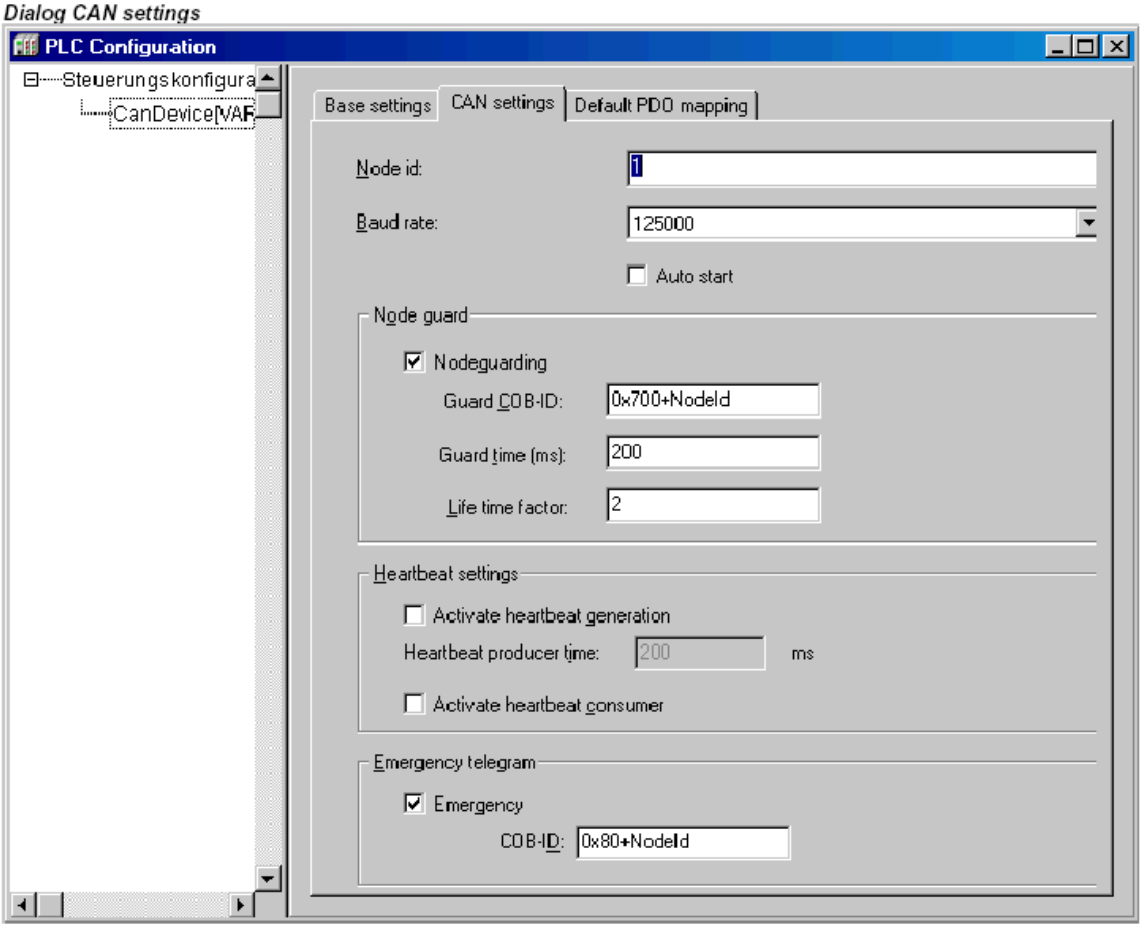
Name of updatetask: 调用CanDevice任务名，一个选择列表将提供此工程中可以采用的所有任务。

EDS file generation: 如果要在当前配置设定中产生一个设备文件以便能够稍后在任意主模块的配置中使用CanDevice，那么就要激活这个选项。在**Name of EDS file**域中输入此文件路径和名称。可以选择性地用手工创建一个模板文件（EDS文件模板），在配置对话框中用完成的设置来补充这个文件。例如，可以创建一个包含某些EDS文件条目的文本文件，把它保存。"EDS_template.txt"并且把这个模板的路径输入到当前对话框中。然后，如果从当前工程产生EDS文件"device_xy.eds"，那么把从工程产生的条目与这个模板合并并且存入"device_xy.eds"。（对模板文件不要使用扩展名".eds"！）如果用当前工程制作创建已由模板定义的条目，那么模板定义不会被改写。

为了输入文件路径，可以使用标准对话框。浏览文件，可以用按钮**Browse**打开它....

CanDevice的CAN设置

Dialog CAN settings



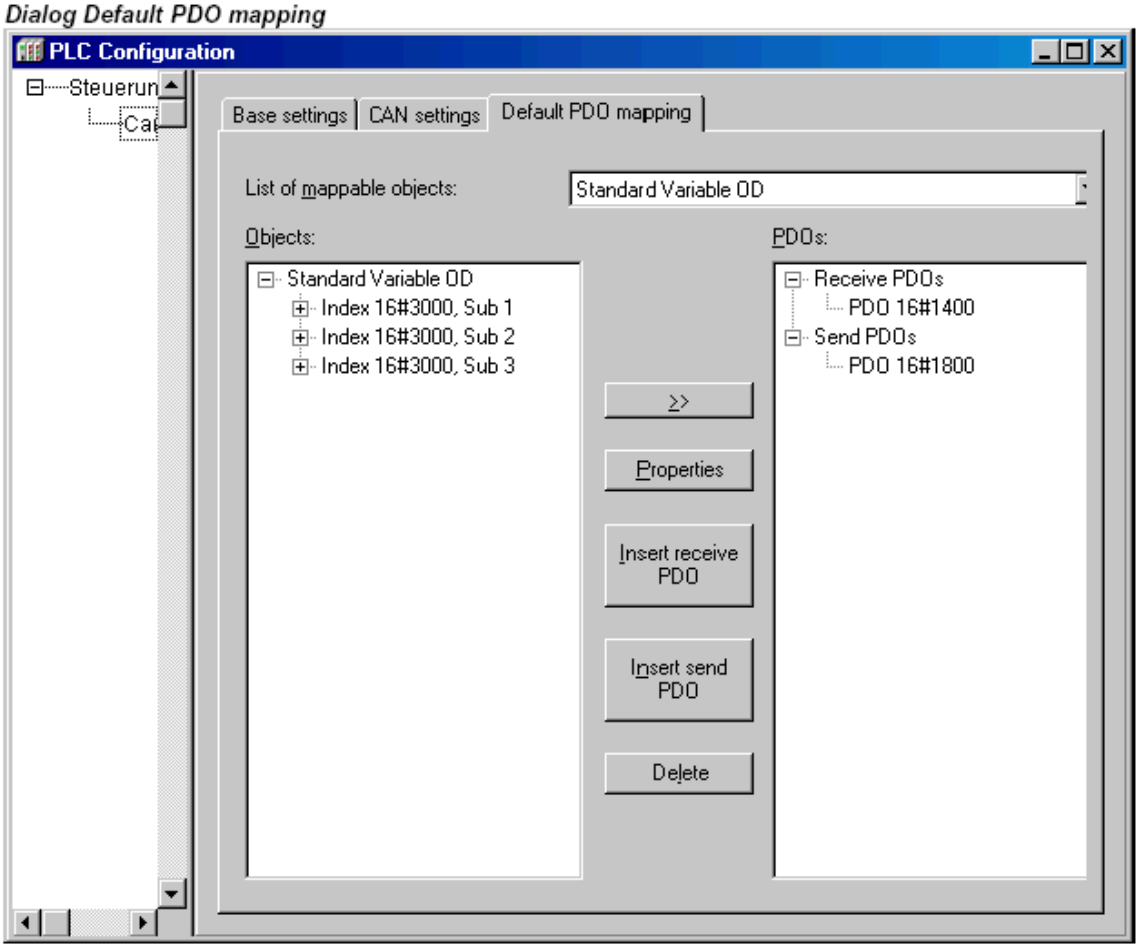
可以设置Node id 和 Baud rate。node id是节点号，由主模块用它来在CANopen网络中寻址设备。

可以配置Nodeguarding, Heartbeat和Emergency Telegram功能。

请参看CAN模块和主模块配置的相应描述。Heartbeat当前不被支持。

CanDevice默认的PDO映射

Dialog Default PDO mapping



在这个对话框中，局部参数管理器的输入可以分配给由CanDevice发送／接收的PDOs。为了PDO映射，然后在任一个集成了CanDevice的master配置中，可以采用PDO映射。

在参数管理器列表中，参数项通过索引／分索引和工程变量连接起来。

注意:索引的分索引0－这暗示了不止是一个分索引－用于隐含存储分索引号。因为这个原因不要在参数管理器中使用分索引。同样注意，在参数管理器中，某个索引参数必须用升序输入(分索引 1,2,3...）。

List of mapable objects：从选择列表中选出CAnDevice要产生PDO的那些变量参数列表。如果目标系统支持，'Mapping'类型的参数列表可以在参数管理器中创建，它包括尤其要用于CANDevice的PDO映射的过程变量。在这种情况下，在mapping对话框中只是提供这些参数列表。否则，提供所有可用的'Variables' 和 'Instance'类型的参数列表。

根据选择的参数列表，**Objects**将出现在左边窗口。通过按钮**Insert receive PDO**，在右边窗口可以建立要求的PDO配置(**PDO's**)。 **Insert send PDO**在对应的列表组成元素下插入'Receive PDOs' 和 'Send PDOs'。为了把左窗口的一个对象分配给发送或者接收PDOs，在左窗口标明对象，在右窗口标明PDO，接着按>>，于是在右边窗口中的PDO下插入对象。在为其他的CAN模块的PDO配置使用的对话框中可以定义PDO的**属性**。

通过使用按钮**Delete**，在右边窗口中当前标明的PDO将从配置中取消。

例子：

目标：在CanDevice的第一个Receive PDO (COB-Id = 512 + NodeId)上，要接收变量PLC_PRG.a。于是，在参数管理器的变量列表中必须把索引／分索引分配给变量PLC_PRG.a 。只有在'Network functionality'类目标设置中激活参数管理器，并且在那里定义了有效的索引和分索引范围，才能打开参数管理器。

现在在CanDevice 的'Default PDO-Mapping'对话框中，各参数列表的索引／分索引输入可以分配给Receive PDO 。

6.9.10 在线模式中的PLC配置

在在线模式中，PLC配置显示PLC的输入输出状态。如果一个布尔输入或输出有值TRUE，配置树中输入行头部的小方框将变蓝，非布尔值将被加到输入行的末端(例如 ″=12″)。

布尔输入可以通过单击鼠标交替切换。对其它输入时，在输入行的头部，单击鼠标就会打开对话框，框内的值可以修改。用OK键关闭对话框后，就会在PLC中马上设置已修改的值。

另外， 还考虑目标系统对在线诊断的特殊可能性。

6.9.11 来自PLC的硬件扫描/状态/诊断信息

如果目标系统和实际配置文件(*.cfg)支持，可以从PLC得到当前相连硬件的结构，状态和诊断结果信息，并把它们显示在CoDeSys的PLC配置中：

扫描模块配置

如果目标系统和实际配置文件(*.cfg)支持，就可以对在PLC配置树中选择的当前模块采用上下文菜单中的指令**Scan module configuration。**

这个指令只是在离线的模式中采用。如果激活它， PLC上某个模块的实际硬件配置将被扫描并且自动插入到CoDeSys PLC Configuration的设备树中。于是可以简单地在CoDeSys中映射现有的模块配置。

加载模块状态

如果目标系统和实际配置文件(*.cfg)支持，就可以对PLC配置树中选择的当前模块采用上下文菜单中的指令**Load module state。**

这个指令只是在在线模式中采用。如果激活它，模块的实际状态将从PLC读取并且在配置树中用特殊的颜色显示：

Black： 模块存在并且参数正确。

Blue： 模块存在但是参数不正确。

Red： 没找到模块。

状态显示的更新在每次下载时将自动完成。

显示诊断信息

如果目标系统和实际配置文件(*.cfg)支持，就可以对PLC配置树中选择的当前模块采用上下文菜单中的指令**Show diagnosis messages**。这个指令只是在在线模式中采用。如果激活它，来自PLC的模块实际诊断消息将在CoDeSys窗口中显示。

6.10 采样追踪

6.10.1 综述和配置

如果在目标设置**target settings**（目录 'General'）中激活了采样追踪，那么在CoDeSys 资源中可以采用Sample tracing对象。采样追踪可以用来追踪已被跟踪了一段时间的变量的的值的行踪。这些值被写入了环形缓冲器(**trace buffer**)。如果储存器已满，那么就会重写最早的值。可以同时追踪多达20个变量。每个变量可以最多追踪500个值。

因为在PLC内追踪缓冲器的规模有一定值，在很多或很广的变量DWORD的事件中，只能追踪少于500个值。

例如：如果追踪10 WORD变量并且如果在PLC中储存器有5000字节长，那么对每个变量，可以追踪250个值。

为了能够执行一个追踪，在对象管理器， 在资源登录卡中为一个  **Sampling Trace**打开此对象。建立和加载相应的追踪配置，定义被追踪的变量。（见'Extras''Trace Configuration' 和 'Selection of the Variables to be Displayed'）。

在PLC中创建了配置和启动了追踪后，那么就会追踪变量值。用'Read Trace'，最后追踪的值将被读出并显示除图形曲线。

在工程格式(*.trc) 或在XML 格式 (*.mon) 中，可以保存和重新载入Trace（变量值和配置）。通过

*.tcf-file，仅仅储存和重新载入配置。

在工程中可以采用各种追踪。他们被列在窗口的右上角选择列表('Trace')中。可以选择其中的一个用于当前的追踪配置。

注意：如果任务配置用于控制程序，那么追踪功能归属于调试任务。I

选择要显示的变量

窗口右边的组合框定义追踪配置中的最终变量，窗口的右边显示追踪曲线。如果从列表选择一个变量，那么在读除追踪缓冲器后，变量将用相应的颜色显示(Var 0 绿色，等)。在曲线已经被显示时，便可选择变量。

在追踪窗口中最多可以同时观察八个变量。

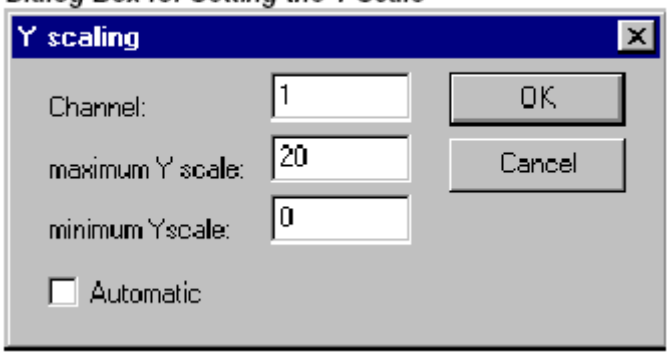
'Extras' 'Y Scaling'

用这个指令在追踪显示中可以改变曲线预设的Y比例。双击曲线，将得到对话框'Y-scaling'。

只要激活选项**Automatic**，便将使用默认的比例，这比例取决于使用的变量的类型。为了改变比例，关闭选项'Automatic'并且输入相关曲线的通道号(**Channel**)和y轴上新的最大比例(**maximum y scale**)和最小比例(**minimum y scale**)。

最早的通道号和比例位时预制的。

Dialog Box for Setting the Y Scale



'Extra' 'Start Trace'

符号: 

用这个指令，追踪配置传送到PLC，并且，追踪采样在PLC中启动。

'Extra' 'Read Trace'

符号: 

用这个指令，从PLC读当前的追踪缓冲器，并且，显示被选变量的值。

用这个菜单的一些指令，把追踪配置和追踪值存入文件或从文件加载追踪配置和追踪值，从控制器把追踪加载到工程，或者某个追踪设置成工程要用的追踪。

注意：考虑用菜单' Extras' ' Save Trace' 指令来保存和重载跟踪的另一种方式

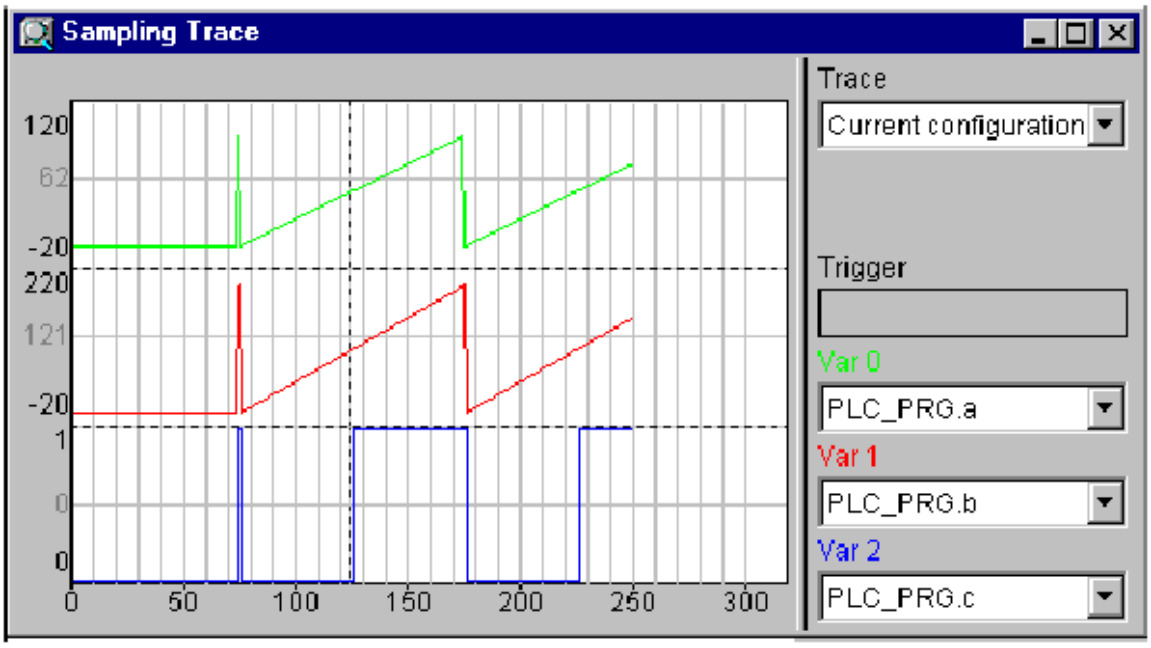
'Extra' 'Stop Trace'

符号: 

这个指令在PLC中停止采样追踪。

6.10.2 采样追踪的显示

Sampling Trace of Different Variables



如果加载了一个追踪缓冲器，那么将读出和显示所有要显示的变量的值。如果没有设置扫描频率，那么将在X轴上写上追踪值的连续数字。追踪窗口的状态指示器（第一行）显示追踪缓冲器是否装满，追踪什么时候完成。

如果指定了扫描频率的值，那么x轴将规定追踪值的时间。把这个时间分配给“最早的”追踪值。例如，显示最后25秒的值。

把值以相应的数据类型写在Y轴上。一句使最低和最高位都能安插在显示区的方法设置比例。例如，Var0的最低值是6，最高值是100，其比例设置在左边。

如果满足触发条件，那么在触发条件出现前后的分界面处显示一条垂直的点线。

在改变工程或者离开系统前一致保存已被读出的存储器。

'Extras' 'Cursor Mode'

在监视区设置光标的最简单的方法就是用鼠标左键点击那里。那样就会出现光标并且可用鼠标移动它。在监视窗口顶部，显示当前光标的x位置。在相邻的'Var 0', 'Var 1', ..., 'Var n'域中，显示各自变量的值。

另一种方式是指令'Extras' 'Cursor mode'。用这个指令两条垂直的直线将出现在采样追踪中。开始时，它们重迭在一起。用箭头键可以左右移动其中的一条直线。按<Ctrl>+<left> 或 <Ctrl>+<right>，可以使移动速度增加10倍。

如果再按<Shift>键，可以移动第二条直线，显示与第一条线间的差。

'Save to file'

用这个指令，在文件中以XML格式保存追踪配置和追踪值。为了这个目的，打开保存文件的标准对话框。将自动使用文件扩展名*.mon.

用指令'Load from file'可以重载*.mon-文件。

'Load from file'

用这个指令可以把XML格式文件（*.mon）中的追踪配置和追踪值加载到工程中。它打开一个打开文件的对话框你可以浏览带扩展名*.mon的文件。加载的追踪将被显示并且被添加到配置对话框域'Trace'中的选择列表。如果要把它设置为当前工程追踪配置，那么使用指令'Set as project configuration'。

用指令'Save to file'可以建立一个*.mon-文件。

注意：考虑通过使用菜单'Extras' 'Save trace values'的指令保存追踪的一种替代方法。

'Extra' 'Read Trace'

符号: 

用这个指令从PLC 读现存的追踪缓冲器，并且显示被选变量的值。

'Set as project configuration'

用这个指令，可以把有效的追踪列表（追踪窗口中域'Trace'）中当前选择的追踪配置设置为此工程内的现行配置。

选择列表除了当前使用的（在顶端）外还提供了通过指令'Load from file'从*.mon-文件载入工程的所有追踪。

'Load from controller'

用这个指令，当前在控制器上使用的追踪配置和追踪值，可以载入CoDeSys工程。它将被显示在追踪窗口，并且可设置为现行的工程追踪配置。

6.10.3 保存采样追踪

用这个菜单的指令把追踪配置和追踪值保存到文件中，从文件把它们重载到工程中。此外，可以把追踪保存到ASCII形式的文件中。

注意: 考虑通过使用菜单'Extras' 'External Trace Configurations'的指令储存和重载追踪的一种替代方法(XML 格式, *.mon-文件)!

'Save Values'

用这个指令可以保存采样追踪的值和配置数据。打开保存文件的对话框。文件名接受扩展名"***.trc**"。

注意: 在这里保存追踪的值和追踪配置，然而，在配置对话框中**Save trace**仅仅涉及配置数据。

保存的采样追踪可以用'Extras' 'Load Trace'再次载入。

'Load Values'

用这个指令，可以重新载入保存的采样追踪的值和配置数据）。打开一个打开文件的对话框。用"*.trc"扩展名选择想要的文件。

用'Extras' 'Save Values'可保存采样追踪。

'Extras' 'Stretch'

符号: 

用这个指令伸展开（Zoom）显示的采样追踪的值。用水平的画面调整条设置开始的位置。随着一次次的重复伸展，显示在窗口中的追踪段的规模将不断收缩。

这个指令与'Extras' 'Compress'相对。

6.10.4 外部采样追踪配置

'Extras' 'Show grid'

用这个指令你可以在绘图窗口切换栅格的开和关。当打开栅格时，记号( )将出现在相邻的菜单项里面。

'Extras' 'Compress'

符号: 

用这个指令压缩显示采样追踪的值；即，使用这个指令后可以在更大的时间框架内看追踪变量的变化。可以多次执行此指令。

这指令与'Extras' 'Stretch'相对。

'Extras' 'Cursor Mode'

在监视区设置光标的最简单的方法就是用鼠标左键点击那里。随之出现光标并且可用鼠标移动它。在监

视窗口顶部，显示当前光标的x位置。在相邻的'Var 0', 'Var 1', ..., 'Var n'域中，显示各自变量的值。

另一种方法是指令'Extras' 'Cursor mode'。用这指令两条垂直的直线将出现在采样追踪中。开始时它们重迭在一起。用箭头键可以左右移动其中的一条直线。按<Ctrl>+<left> 或 <Ctrl>+<right>，可以使移动的速度增加10倍。

如果再按<Shift>键，可以移动第二条直线，显示与第一条线之间的差。

'Extras' 'Multi Channel'

用这指令可以在采样追踪的单一通道和多重通道的采样追踪显示之间改变。多通道显示时在菜单项前有记号( )。

已经事先设置了多通道显示。显示窗口被分为八个显示曲线。对每条曲线，最大值和最小值显示在边缘上。

在单一通道显示，所有的曲线用相同的比例因子显示并且迭加在一起。当显示曲线紊乱时，它很有用。

'Trace in ASCII-File'

用这指令，可以把采样追踪保存在ASCII-文件中。它打开一个打开文件的对话框。文件名接受扩展名"*.txt"。值按照下列模式存放在文件里：

```
BODAS Trace
D:\BODAS\PROJECTS\TRAFFICSIGNAL.PRO
Cycle PLC_PRG.COUNTER PLC_PRG.LIGHT1
0 2 1
1 2 1
2 2 1
.....
```

如果在追踪配置中没有设置扫描频率，那么第一栏是周期；那意味着每一周期在任一给定时间只记录一个值。另一方面，这一项是以毫秒数为单位存储变量值的时间点。

后面多栏保存追踪变量的相应的值。在任一给定时间，用空格把值彼此分开。

依据顺序，在第三行显示一个接一个的附属变量名，其顺序是(PLC_PRG.COUNTER, PLC_PRG.LIGHT1)。

6.11 参数管理器

参数管理器是 CoDeSys 编程系统的一个目标系统专用组件，并且必须在目标设置中激活(见 6.12 节)。

参数管理器用来建立 CoDeSys IEC-程序的变量,常量参数或者特定系统参数在一个网络中的所有 CoDeSys 兼容系统都可以访问它们以实现数据交换，一段是通过现场总线。为此目的，在编辑器中可创建参数列表和从运行系统的下载和加载参数列表.

注意: 也可以通过程序用在变量声明中的条目来创建和提供参数列表。

什么是参数？：

在本上下文中的参数是：

CoDeSys IEC 工程的过程变量

独立的过程参数

目标系统预先定义的系统专用参数

功能块实例或结构体变量，数组

每个参数由某个属性（例“默认值”，“访问权”），特别是可为读写数据寻址的唯一访问链（“索引”，“子索引”，“名称”）来识别。通过通信服务完成数据交换，并且不必要知道任何变量的地址或者提供任何额外的函数。因此，使用参数管理器的功能是使用网络变量的一种办法。

什么参数列表？：

参数列表被用来组织参数并且可以用工程来保存可以加载到由相应 IEC-program 控制的本地目标系统。对参数的每种类型，有相应的参数列表类型。

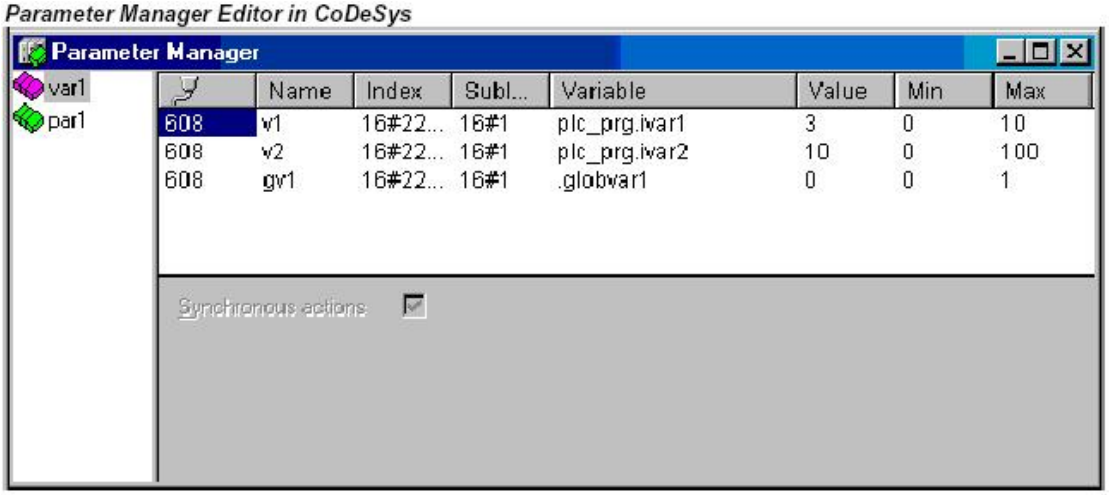
在参数列表中，每个参数项用一行描述。列表的每一栏代表一种参数属性。除了某一套标准属性外，还可以使用生产商规定的属性来描述参数管理器中的参数。

在参数管理器中哪些属性（栏）可见且可编辑，在参数列表中用哪种方法来安排属性，由目标特定描述文件中的定义决定。如果描述文件丢失，那么将显示全部标准属性，各自属性显示默认值。

除了工程变量和工程常量的列表外，参数管理器还能处理系统参数列表。系统参数是由目标系统给出的参数。进而你可以在参数管理器中为功能块实例或者结构体变量创建列表，这些列表也可以用基于用户定义的模板来创建。

由于事实上数据储存独立于 IEC 程序，例如，参数列表可以用来保存'组态表'，即使程序被其它版本的替代，它也会被保留。而且可以用不同的组态表供给一个正在运行的 PLC，而不需要重新下载程序。

Parameter Manager Editor in CoDeSys



注意：在启动某个工程时，是否要考虑参数管理器，取决于目标系统。

激活参数管理器

在 CoDeSys 资源中打开目标系统设置，选择网络功能：

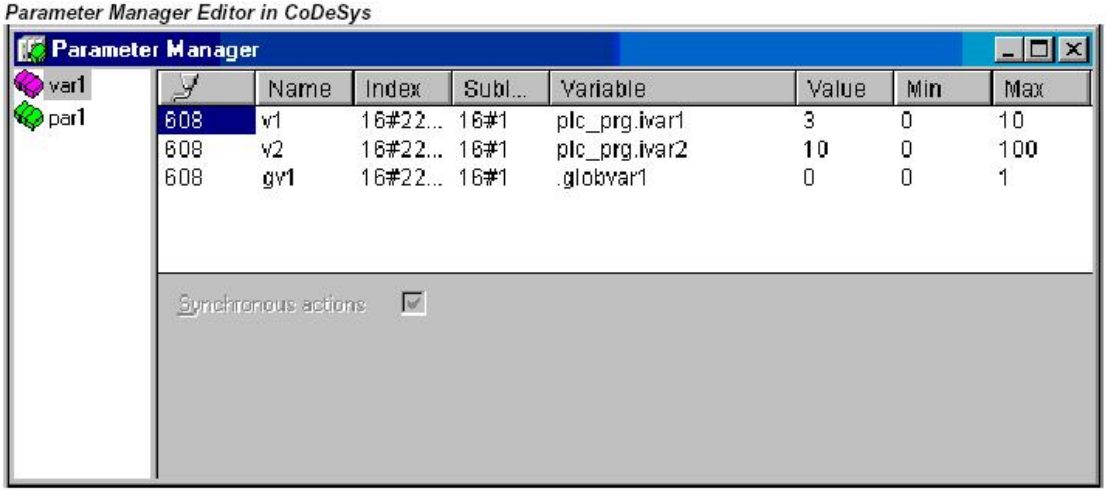
激活选项'支持参数管理器',在'参数索引范围，'变量索引范围和目标系统支持的'映射索引范围（对 CAN 设备 PDOs 的映射)中输入有效的索引范围和分索引范围。

6.11.1 参数管理器的编辑

在资源表中选择对象’Parameter-Manager’。编辑器窗口将打开，在那里可以建立，编辑和储存参数列表，并且在线模式中也可以把它们加载到目标系统和监视当前参数值。

注意： 为了在 CoDeSys 中使参数管理器的功能有效，在 Target Settings 中必须激活，选项’ Support Parameter Manager’ ，并且必须定义相应的索引范围！

Parameter Manager Editor in CoDeSys



编辑器窗口是分成两部分。左边用于导航，显示一列当前载入 Parameter Manager 的所有参数列表的列。右边包含表格编辑器，各栏用属性的名称作为标题。

在导航窗中，可以插入，删除，重新排列或重命名不同类型（变量，常量参数，模板实例，系统参数）的参数列表。

在表格编辑器中， 用参数输入填充列表。每个列表类型将显示各自选择属性栏，它们可被编辑或仅仅可见。这或者由目标特定描述文件定义或者接受默认设置。

可以在 navigation 窗口和表格编辑器之间通过按<F6>来回跳动。

在线模式中可以把已建立的列表载入当前连接的目标系统。也可以用参数管理器功能访问他们必须与其它系统交换数据。而且，在参数管理器窗口中，可以监视参数的当前值。如果当前在线连接没有建立，那么只能建立本地参数列表并且用工程保存。

参数列表的类型

参数管理器可以处理下列参数列表类型：

变量： 这类型的参数列表里各项表示工程的过程变量。

参数： 这类型的参数列表里的各项表示其值与过程无关的参数。

系统参数： 这类型的参数列表里的各项表示与过程无关的，由目标系统确定的特殊常量参数。系统参数列表不能删除或重命名。

模板： 模板不包含为数据交换而直接访问的参数项。事实上，它的各项为功能块或结构体元素提供了“基本属性配置”。因而可在’实例’类的参数列表中使用模板。

实例： 这类型的参数列表里的各项表示是功能块或结构体类型变量的参数项，这意味着它是实例或结构体变量。为了方便地输入参数，可以使用已经在参数管理器中建立地模板。

映射： 只有目的系统支持下才能采用此，这列表类型它地各项代表在 CAN-Device 的 PDO 映射中要使用地过程变量。因此，映射列表基本上是变量列表，但是，它们工作在各自的索引/分索引范围内。这范围必须在目标设置的网络功能类目中定义！为此只有在 PLC 配置中配置的 CANDevice 才会用于’映射索引范围’。否则，在 PDO 映射对话中采用变量或实例列表的各项。

根据 XML 格式的描述文件定义的属性，在参数管理编辑器中显示各个列表类型。如果这个文件丢失，使用默认的设置。

实例和模板

“实例”参数列表…

它处理表示功能块，结构体变量或数组的参数项。功能块或结构体的实例列表都基于模板，模板在也是要在参数管理器中为各个功能块或结构体来定义的。数组的实例列表参数管理器中不能用在参数管理器中制作的模板，但是可以直接参照工程中使用的数组。

“模板”参数列表...

它不包含为数据交换而直接访问的参数。事实上，它为表示功能块或结构体元素的参数项输入定义索引和分索引的偏移量和定的属性。然后在实例参数列表中使用模板（见上述内容），从而为功能块或结构体的实例变量提供简单的方式建立参数项。

制作模板参数列表:

在与基本 POU 相邻的编辑域中输入要建立参数模板的功能块或结构体的名称。用输入帮助，可以浏览工程中可用 POUs。按'应用'进入参数列表编辑器中已选择的 POU 的元素。然后编辑属性域并且关闭此列表使它可在实例列表中使用。

在上下文菜单或在'附加'菜单中的指令'插入丢失项'将根据当前基本 POU 的版本更新表项。在删除几行后或改变基本-POU 后往往需要用此指令。

为了对数组建立实例参数列表，在不必在参数管理器中建立模板。模板 ARRAY 可映射隐含使用。

建立实例参数列表：

从表格下的选择列表编辑模板。此列表提供了当前可用于参数管理器中的功能块或结构体的所有模块，如果你想直接参照在工程中被使用的数组，也可选用你要的数组。按'应用'，把预先确定的元素插入参数列表。

为你想建立的参数项，在编辑域'基本变量'正确输入工程变量（其烈性必须使所选模板描述的功能块或结构体或数组的类型）。

对此实例输入'基本索引号' 和 '基本子索引号' 。通过增加在模板中为每个元素定义的索引和分索引（就数组来说其值是 0）自动标出各元素的索引和分索引。他们将自动填写各自的属性域。例如：如果为一个元素输入基索引“3”，在模板中为它定义索引偏移值为“3000”，此元素将设置为索引 3003。

在上下文菜单或'附加'菜单中的指令'插入丢失项'将更新表项。在删除表项后或修改模板后，需要使用此指令。

例子：

建立一个带有输入和输出变量 a,b,c 的功能块 fubo。在 In PLC_PRG 中定义下列功能块的实例: inst1_fubo:fubo; inst2_fubo:fubo.

为了为变量 inst1_fubo.a，inst1_fubo.b，inst1_fubo.c 和 inst2_fubo.a，inst2_fubo.b，inst2_fubo.

建立参数列表,打开参数管理器。插入模板参数列表,把它命名为和"fubo_template"名称。定义Base-POU："fubo"。按 Apply 定义并且为成分 a，b，c，定义属性：te。输入索引偏移量：a: 16#1, b: 16#2, c: 16#3. 分索引偏移量,例. a: 16#2, b: 16#3, c: 16#4。

然后关闭此模板，插入实例类型的参数列表。选择模板"fubo_template"。插入基本变量"inst1_fubo"。定义一个的基索引(例 16#2300)和基分索引(例 30)(必须考虑到在目标设置的网络功能类目中定义的范围!)按 Apply 显示索引，通过增加基偏移量和模块定义的偏移量，算出的元素 a,b,c 索引是：16#2301， 16#2302，16#2303；分索引是:16#23，16#33， 16#43。

插入列表

快捷键： Ins

在参数管理器中，为插入新的参数列表，在'插入'或上下文的菜单中使用指令'插入列表...'和 '插入新列表'。当输入焦点在空的导航窗口或在导航树中有的条目上时可以采用此指令。

打开'插入列表'对话框：

为新的参数列表插入名称（在列表类型范围内一定是唯一的），选择下列列表类型中的一个：

变量	过程变量表项
参数	常量参数表项，常量参数值与过程保持独立
模板	为功能块或结构体的元素设置的属性的模板
实例	功能块或组织的类型变量的表项，基于相应的模板（见上述内容）
映射	用于 CAN Device 的 PDO mapping 中的过程变量条目只有在目标系统支持下才能采用这种类型！
系统参数	参数值和与过程无关并且由目标系统定义的参数表项

用 OK 确定设置和关闭对话框后，新的参数列表将出现在导航窗口中，由图标表明列表类型。在表格编辑器由此栏中标题显示相应属性。栏的选择和次序由目标规定的描述文件定义，否则使用默认的设置。为每个要求的参数输入一行，来编辑表格（见 6.11.4，编辑参数列表）。

重命名列表

在导航窗口中当前标明的参数列表－可以用’附加’或上下文菜单中指令’重命名列表’重命名。当在列表名称上双击鼠标时，打开编辑域。

剪切 / 复制 /粘贴列表

快捷键： 〈Strg〉 + 〈X〉, 〈Strg〉 + 〈C〉, 〈Strg〉 + 〈V〉,

指令’剪切’（编辑菜单）或’剪切列表’（上下文菜单）把当前标明的列表从导航窗口移动到暂存缓冲器因而，用’粘贴’指令可以把它重新插入到导航树中的任何其它位置。在重新插入之前要需要标明插入其前的表项。

指令’复制’’复制列表’也用暂存缓冲器，但是原先的导航树表项将被保持，并且通过’粘贴’添加复制。

删除列表

快捷键： 〈Del〉

在导航窗口当前选择的列表将由指令’删除’（编辑菜单）’删除列表’（’附加’或上下文菜单)删除。

注意：在线模式中，这个指令将删除运行系统中相应的列表。

编辑参数列表

显示哪个栏（属性）：

当前标明的参数列表(导航窗内)将按目标指定的描述文件中的定义或默认设置示在表格窗口中。

这意味着每个特定参数的属性值将在各行中以列表类型的规定次序和所选的栏目进行显示。

当光标指到列表栏标题条的任何一个域的时候，可以通过上下文菜单中 deactivating / activating 来使该栏淡出和淡入。

当光标放在栏标题域时，为了修改此栏，可以移动栏标题域之间的分割线或者使用在上下文菜单中的一个指令。指令 Standard column width 为所有的栏设置标准宽度，并让所有的栏在窗口中可见。指令 Maximize

width 改变当前焦点栏的宽度以致使每个项得到充分的显示。
编辑参数列表表项的指令：
下列编辑参数列表的指令在上下文菜单和菜单'插入'或 '附加'中可用：
插入/删除行：

插入新行 或 新行	在光标当前所在位置前插入新的输入（行）。
当前行后插入 或 在当前行后插入新行 或 快 捷 键:<Ctrl><Enter>	在光标当前所在位置后插入新的输入（行）。
删 除 行 或 快 捷 键 : <Shift>+<Del>	删除光标当前所在位置的行。
剪切, 复制, 粘贴行	这些指令用来移动（剪切 / 粘贴）或者复制（复制 / 粘贴）选择的行。

编辑属性值：
如果为一个参数项插入新的行，那么属性域将被自动写入默认值。见 6.11.3，' 参数列表和属性'，为了输入或编辑属性值，点击相应的域。如果属性可编辑那么就会打开编辑域。在要输入 CoDeSys 工程元素的域中。可以采用输入帮助（<F2>）
按<Enter> 关闭输入。
用箭头键跳到另一个域。
按 <Del> 删除当前编辑域的表项。
为了在‘十进制’和‘十六进制’之间交替切换输入格式，可使用在'Extras'菜单中的指令'Format Dec/Hex'。
为了把输入焦点置入（或退出）导航窗口可按 <F6>。
选项：
在编辑窗口中在表格下面，可以激活下列选项（可用性依赖于列表类型）：
与程序一起下载： 在登录时，列表将自动下载到控制器。
同步动作： 当前没有此功能

参数列表的排序

在参数列表内表项的次序可以按属性值的上升或下降的次序来排列。在脱机方式和在线方式中都可做次项工作。
在所要求的属性栏标题的域上双击鼠标。于是，表格行将重新排列，并且在属性栏标题的域中用箭头，显示当前的排列（向上的箭头 = 升序排列，向下的箭头，降序排列）。

6.11.2 参数列表的导出、导入

'附加' '导出'

'附加'菜单的指令'导出'可以用于从参数管理器列表导出到 XML 文件.例如，使用在 CoDeSys 参数管理器中的导出功能把这个文件到处到到另一个工程。打开保存文件的标准对话框，事先设置文件的扩展名 *.prm 。把在参数管理器中所有有效的列表写到导出文件。
用一般工程导出功能('工程' 导出')可以输出参数管理器的内容。

'附加' '导入'

'Extras'菜单的指令'Import'可用来导入描述参数列表的 XML-file。例如，此文件在 CoDeSys 参数管理器中使用输出功能来建立。

如果导入文件包含在参数管理中已经使用过的列表，那么打开对话框，询问用户是否要重写已有的列表。

6.11.3 在线模式下的参数管理

编辑器和控制单元之间的列表传输

在线模式中，在编辑器中建立的参数列表可以下载到运行系统或从运行时间系统上传。而且，可以把单个参数值写入运行系统。

注意：在自动登录时，选项'与工程一起加载'被激活所有参数列表的下载将被自动完成。

在线方式中在参数管理器中显示追加栏的当前值：

为了处理编辑器和控制器之间的列表传输，在菜单'附加'中可用下列指令。

删除列表	在导航窗口中当前标明的列表将从 PLC 运行系统中删除。
写列表	这个指令打开对话框'复制对象'，在这里可以选择希望下载到运行系统的有效列表。按'确认'后，下载将马上完成。
读列表	参数类型的所有列表将从运行系统读入并且载入参数管理器。
写入新值	在'数值'栏定义的所有的值将吸入运行系统内的参数列表。为写单个值，双击栏中相关域得到对话框'写入新值'。
写入缺省值	在'缺省'栏中定义的值将写入运行系统中的参数列表。
接受数值	从运行时间系统读当前值并且把它上传到'数值'栏 。

指令'格式 Dec/Hex'可用于在‘十进制’ 和‘十六进制’的输入格式之间交替切换。

在启动工程中的参数列表

在建立工程启动时是否要考虑参数列表,取决于目标系统.

6.12 PLC 浏览器

PLC-浏览器是一个基于文本的控制监视器（终端）。以一个输入项输入来自于控制器的特定的信息请求命令，字符串发送给控制器。在浏览器的一个结果窗口显示返回的响应串。这功能为诊断和调试按钮服务。

设置号的目标系统可用指令由 CoDeSys 的标准集加控制器生产商可能的扩展集组成。他们在 ini 文件中被管理，从而在运行系统中被执行。

请参考：

关于 PLC 浏览器操作的一般讨论

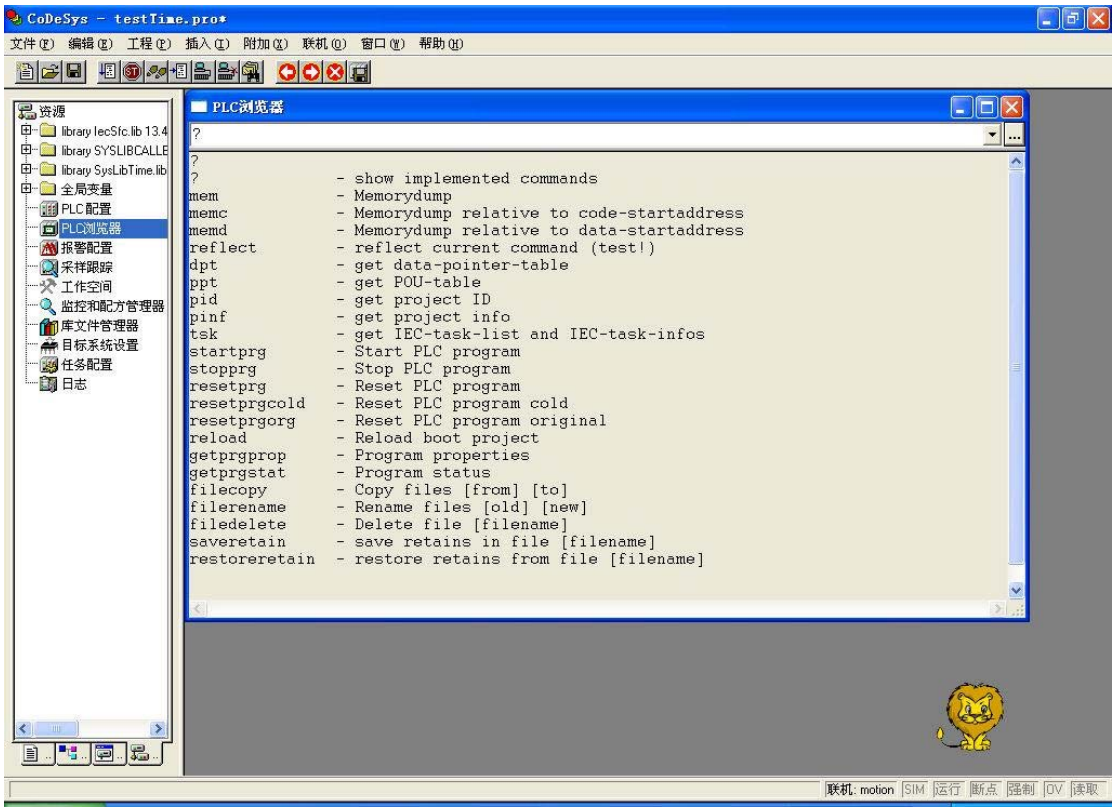
在 PLC 浏览器中的命令输入

在 PLC 浏览器输入命令时使用宏指令

PLC 浏览器的更多选项

关于 PLC 浏览器操作的一般讨论

在资源中选择 PLC-浏览器。只要在当前目标设置(网络功能类目中)激活它，就可以采用它。



浏览器由指令输入行和结果 / 显示窗口组成。

在选择框中，输入行显示自工程启动以来所有输入的指令列表输入历史记录。它们是可以再次选择直到关闭工程。只有那些不同于现有的指令才被添加到此列表中。

用<Enter>把输入的指令发送到控制器。如果不是在线连接，那么在结果窗口用与发送到控制器的同样的方式显示指令，否则，在那里显示来自于控制器的响应。如果把新的指令发送到控制器，那么会删除结果窗口的内容。

可以以指令串 的形式输入指令，也可能使用宏指令。

在 PLC 浏览器中的命令输入

基本上 PLC-浏览器可以采用在运行时间系统中使的 3 S 标准指令硬代码。它和直接内存操作，工程输出，状态功能及运行监视等功能有关。它们在浏览器的 ini-file 中描述，浏览器的 ini-file 是 Target Support Package 的部分。这些标准指令可以进一步的补充专门指令，例如，自我诊断功能或控制应用的其它状态信息。指令列表的扩展必须在运行系统的客户接口中或者通过在浏览器 ini 文件中补充条目来进行。

当打开工程时，基于在浏览器 ini 文件中的条目产生 PLC 浏览器可用的指令列表。在对话框”Insert standard command″中使用用...键或者使用<F2>，可以象输入帮助那样访问指令列表。也可以用指令’Insert’’Standard commands’来得到指令列表。指令可以用手工输入命令行，或者通过在相应的表项上双击鼠标从列表选择指令。

一般指令语句：

<KEYWORD><LEER><KEYWORD-DEPENDEND PARAMETERS>

Keyword 是指令。可扩招那的指令参数在输入帮助窗口相关的工具提示中描述。

在输出数据窗口重复已经被发送的指令，控制器的响应出现在它下面。

例如，用指令“pid”向控制器请求工程的 Id。

在命令行输入：

pid.....

在结果窗口输出：

pid
Project-ID: 16#0025CFDA

为每个标准指令用“？”〈BLANK〉〈KEYWORD〉提供帮助文本。在 ini 文件中给出了类似的定义。

下列指令完全集成在运行系统中，并且其相应的条目包含在 ini 文件中以便提供输入帮助，工具提示和帮助：

指令	描述
?	运行系统提供可用的指令列表。此列表与目标系统描述文件的状态无关。
mem	内存区域的十六进制映像 语句 1：mem 〈start address〉〈end address〉 语句 2：mem 〈start address〉-〈end address〉 可以用十进制，十六进制或作为一个宏来输入地址。
memc	与控制器中代码的起始地址相关的十六进制影响 ；象 mem 一样，数据被添加到代码区。
memd	与控制器中数据库地址相关的十六进制映像；象 mem，数据被添加到代码区。
reflect	为试验目的，反映当前命令行
dpt	读数据指针表
ppt	读 POU 表格
pid	读工程 Id
pinf	读工程信息
tsk	显示包含任务信息 IEC-task 列表
startprg	启动 PLC 程序
stopprg	停止 PLC 程序
resetprg	复位 PLC 程序。仅仅初始化非保留的数据。
resetprgcold	冷复位 PLC 程序。也要初始化保留的数据。
resetprgorg	复位最初的 PLC 程序。删除当前应用程序及所有数据（包括保留数据和持久数据）。
reload	重新加载引导工程
getprgprop	程序特性
getprgstat	程序状态
filedir	文件指令"dir"
filecopy	复制文件[从] [到]
filerename	重命名文件[旧名称] [新名称]
filedelete	删除文件[文件名]
saveretain	保存保留变量
restoreretain	加载保留变量
setpwd	在控制器上设置密码 语句： setpwd 〈password〉 其中 level＝0（默认值）在从偏程系

统登陆时才有效 level＝1 对所有应用均有效

delpwd　　删除控制器密码

注意：输入的指令次序的第一个单词被认为是关键字，如果在关键词后有 “<SPACE>?”（例如，“mem ?”），那么就会搜索 ini 文件以查找对此关键字的帮助部分。如果找到了，不　　向控制器发送什么，只是在输出数据窗口中显示。

如果控制器没有识别出指令输入的第一个词（关键字），那么没有找到关键词的响应,在结果窗口中显示未找到关键词。

在 PLC 浏览器输入命令时使用宏指令

如果在命令行输入与宏结合的指令，在指令发送到控制器前就把宏展开。在结果窗口的响应以同样的展开形式显示。

输入语句：<KEYWORD><macro>

<KEYWORD> is 指令。

<macro>是宏，宏有：

%P<NAME>　如果 NAME 是 POU-名，那么把此表达式展开程到<POU-Index>，否则没有变更

%V<NAME>　如果 NAME 是变量名，那么把此表达式展开成#<INDEX>:<OFFSET>，否则没有变更（由控制器解释把这种表示法#<INDEX>:<OFFSET>解释成内存地址）

%T<NAME>　如果 NAME 是变量名，那么把此表达式展开成<VARIABLENTYP>，否则没有变更

%S<NAME>　如果 NAME 是变量名，那么把此表达式展开成<SIZEOF(VAR)>，否则没有变更

如果换码符\（反斜杠）放在前方，那么符号%就被忽视。如果换码符写成\\，那么只是传送此符号。

例如：

在命令行输入：（variable .testit 的内存映像？）

mem %V.testit

在结果窗口输出：

mem #4:52

03BAAA24 00 00 00 00 CD CD CD CD ....ÍÍÍÍ

PLC 浏览器的更多选项

在’附加’ 菜单或在 PLC 浏览器工具栏中，有下列指令用来处理命令输入或历史记录列表：

用’向前一个历史记录’  和 ’向后一个历史记录’  ，你可以向后和向前滚动来查询已完成的结果。历史记录一直继续到你离开工程。

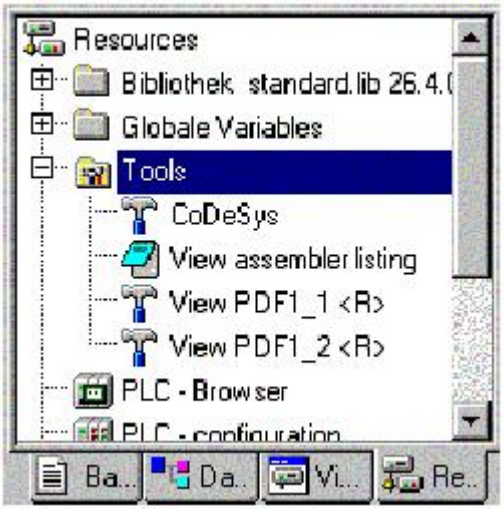
用’取消命令’  ，可以中断已经开始的查询。

用’保存历史记录’  ，可以把已完成的查询结果保存到外部文本文件中。出现对话框’另存为’，在这里你可以输入带有扩展名".bhl"的文件名（浏览器历史记录列表）。指令’打印上一次命令’(Print last command)打开打印用标准对话框。可以打印当前查询和在信息窗内的输出数据。

6.13 工具

如果开启了当前设定的目标系统的工具功能，那么在资源中可以使用对象'工具'。它显示外部工具的可执行文件的所有可用快捷键（连接），为了从 CoDeSys 内调用这些外部程序，双击快捷键激活它。由目标文件确定允许使用哪些快捷键。依赖这定义，用户可以在'工具'文件夹中添加或删除新的快捷键。

例如，对象管理器中的工具文件夹如下面所示：



在这个例子中安装了四个工具快捷键。一个用于启动另一个 CoDeSys 编程系统，一个用于打开文本编辑器中的汇编程序列表，其它两个用于打开 PDF 文件。用"<R>"标记的快捷键在 CoDeSys 中不能修改。快捷键包含对编辑器的连接，例如，notepad.exe，或者对某个 PDF 文件的连接，以致，在此表项上双击将打开记事本窗口显示汇编程序列表，或者打开 Acrobat Reader 显示 PDF 文件。

另外，激活快捷键后，你可以马上定义要下载到 PLC 的某个文件。

可用工具快捷键的特性（对象特性）

在管理器的资源来中用鼠标点击条目'工具'前的加号就会打开，可用的快捷键的列表。如果你刚开始创建一个新的工程，那么只能看见那些在目标文件中定义或固定条目的快捷键。如果工具文件夹已被修改，你可以发现用户在 CoDeSys 中添加的其它快捷键。

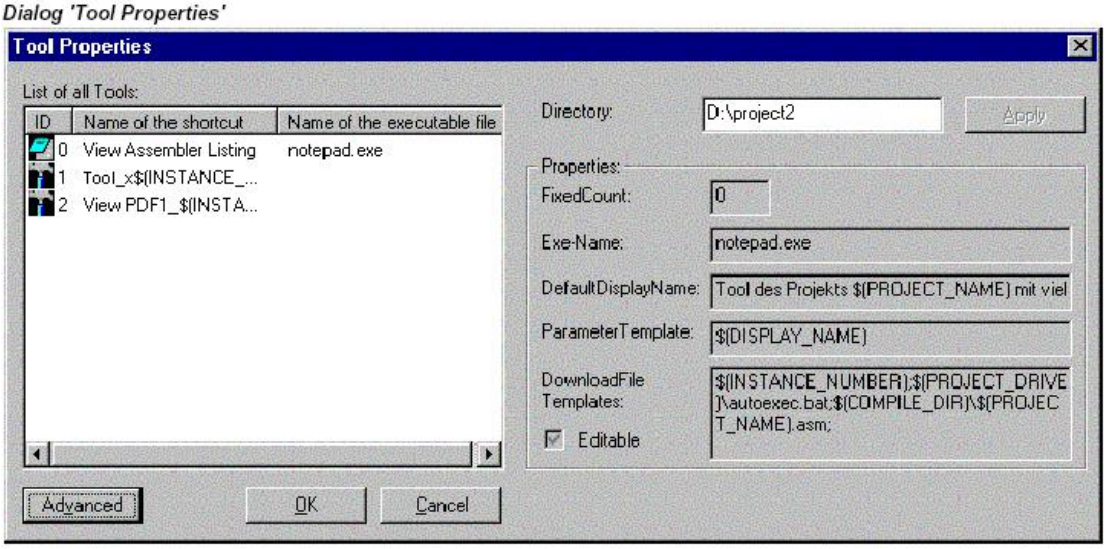
你可以查看工具的全局特性（对'工具'中所有列出的快捷键均有效）及单个快捷键的特性。

1. 工具属性:

如果在资源中标明'工具'，在你可以在上下文菜单或在菜单'工程' '对象'中发现指令'对象属性'，该指令将打开对话框'工具属性'。

在这个对话框里列出了可以用于当前设置目标的全部工具快捷键的表。表上显示了下列参数：工具 Id(唯一的标识号)，Name of the shortcut（快捷键名称用来引用对象管理器中的快捷键），Name of the executable file（工具的可执行文件名称）。按钮 Advanced(高级)用来扩展了对话框或者关闭对话框的扩展。

Dialog 'Tool Properties'



扩展的对话显示快捷键在目标文件中被定义全局特性。在可用编辑域内，Directory(目录)用来定义要使用的可执行文件。点按钮 Apply(应用)后，立即存储文件路径而不关闭对话框，马上保存路径。

工具的特性:

FixedCount　工具快捷键的数量，它是不可改变的被自动的插入到工具文件夹中。只有在输入"0"的时候，用户才能根据需要建立很多快捷键。
注意:对于由目标文定义为“fix”的快捷键 ，在工具文件夹中可能使用的数量被预先确定，并且此属性 不能被 CoDeSys 用户修改（在对象管理器中由"<R>"辨识）

Exe-Name:　工具可执行文件的文件名或完整路径，也可以。输入指向 exe-file 的注册路径："[registry path].<registry entry in this path pointing to an exe-file>"。如果没有 entry ，在"Parameter Template"中给出的文件的扩展文件通过窗口自动产生相应工具的可执行文件的启动。
例子： "C:\programme\notapad.exe", "345.pdf"

DefaultDisplayName: 用来表示在对象管理器中的工具的名称。或许，使用模板$(INSTANCE NUMBER)（见下述'Parameter Template'）。

Parameter Template:　确定由工具打开的文件的模板。下列由相应专门字符连接模板可以使用：
$(PROJECT_NAME)当前打开的工程名
(没有扩展*.pro 的文件名).
$(PROJECT_PATH) 工程文件所在的目录路径（无驱动器指示）
$(PROJECT_DRIVE) 当前被打开工程的驱动器
$(COMPILE_DIR) 工程的编译目录（包含指示的驱动器）
$(TOOL_EXE_NAME) 工具的执行文件名
$(DISPLAY_NAME) 用在工具文件夹中得当前快捷键名
$(INSTANCE_NUMBER)快捷键号码（实例号码运行号码，从 1 开始）
$(CODESYS_EXE_DIR) CoDeSys 执行文件所在目录的路径（包含驱动器指示）
在快捷键特性，在对话框中可看见的模板的转化
(见下述内容)
例子：
"$(PROJECT_NAME)_$(INSTANCE_NUMBER).cfg" ???打开<当前工程名>.cfg,<

快捷键号码>.cfg.

DownloadFile Templates: 在下载期间要复制到 PLC 文件，文件路径或模板，。如果激活选项 Editable ，在快捷键的特性对话框中，这些列出的文件可编辑。如果输入一个没用路径的文件名，将在 CoDeSys 执行文件的目录中寻找此文件。

例子：

″a.up;$(PROJECT_NAME).pro;$(INSTANCE_NUMBER).upp″ ???在下一次下载时 a.up, <当前 CoDeSys 工程>.pro and <快捷键号码>.upp 复制到 PLC。

2. 快捷键特性:

在对象管理器'工具'中标明快捷键项，并且在上下文菜单中或在菜单'工程' '对象'中选择指令'对象属性'。打开对话框'快捷键属性(Shortcut Properties)'，它包含下列项目：

Command 调用工具；执行文件和在'参数(Parameter)' 中命名的文件的路径（由参数模块预定义，见上述内容）

例如： C:\programs\notepad.exe D:\listings\textfile.txt

Parameter 由工具调用的文件的路径。如果已经激活选项'可编辑(Editable)'，它在目标文件中被定义并且可以在这里编辑（见下述内容）。

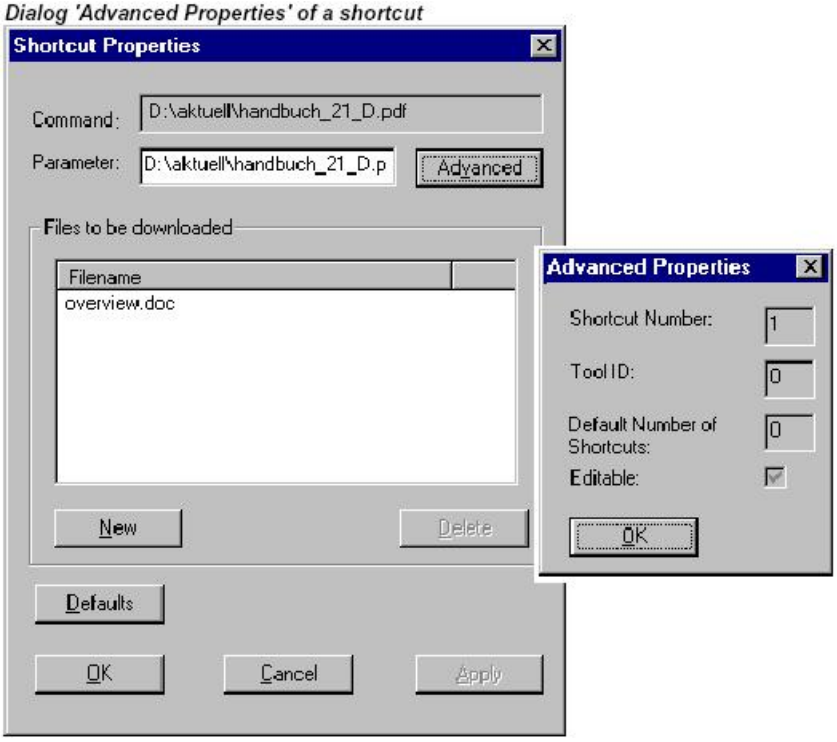
Files to be downloaded 首先，你在这里可找到文件名 Filename，文件名由目标文件定义并且在工具属性中描述(DownloadFileTemplate, 见上述内容)。如果在外延对话框（见下述内容）中激活选项'Editable'，那么可以修改列表。为此目的，点按钮 New 打开对话框'Filename'，在这里可以输入 另一个文件或文件路径。如果输入没用路径文件名，将在 CoDeSys 执行文件的目录中寻找文件。按钮 Delete 将取消当前标明的列表项。

按钮 Standard(标准) 把对话框的条目复位到目标文件定义的默认值。

按钮 Apply(应用)保存完成的设置而不关闭属性对话框。

按钮 Advanced(高级) 扩展对话框如下所示：

Dialog 'Advanced Properties' of a shortcut



Shortcut Number:	运行号码，从 1 开始。当前工具的新的快捷键，每个将得到下一个高号。如果后来取消一个快捷键，保留下的快捷键的号码保持不变。用模板$(INSTANCE_NUMBER)可以把快捷键插入其它定义中。(例如，. 见上述，'Parameter Template')。
Tool ID:	工具的唯一的识别号；在目标文件中定义。
Default Number of Shortcuts:	一个工具的快捷键(实例)的号码。与在目标文件中定义的"FixedCount"相符。见上述，Tool Properties。.
Editable:	如果激活这选项，可以编辑域'Parameter' 或要下载的文件列表。

按钮 OK(确认)完成设置并且关闭属性对话框。

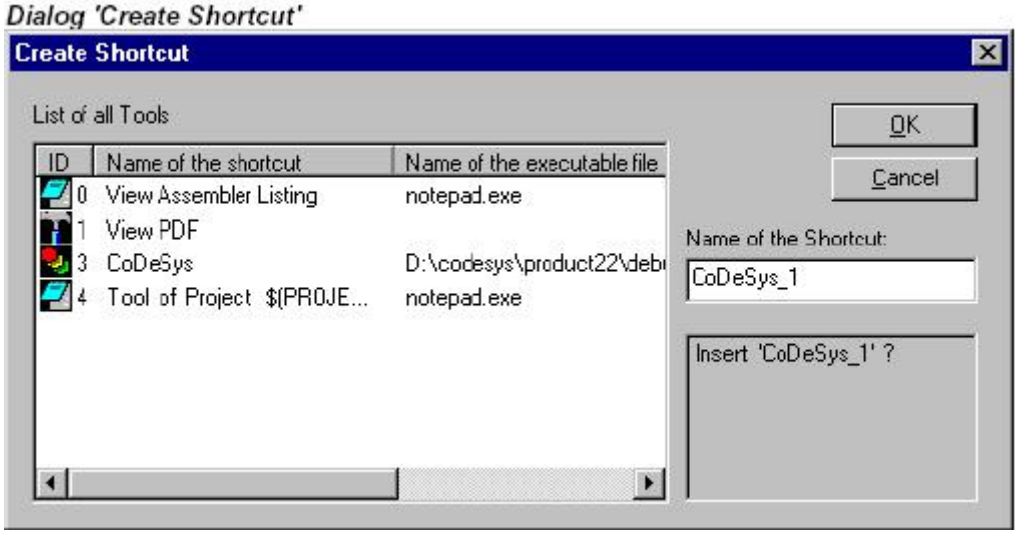
创建新的工具快捷键

在对象管理器的资源中选择'工具'条目或者一个快捷键条目，在上下文菜单或'工程' '对象'菜单中选择指令'添加对象'来打开对话框'Create Shortcut(创建快捷键'，见下述内容。

此表格列出所有创建新快捷键（连接）的工具。根据在目标文件中的定义，显示下列参数：工具 ID，Nmae of the shortcut（快捷键的默认名称）和 Name fo executable file（可执行文件的名称）。

为了为提供的工具中的一个创建另外的快捷键,在'ID'栏点击鼠标选择这个工具。于是可以在编辑域Name of the shortcut 修改快捷键的默认名称并且用 OK 确认。如果输入一个还没使用的名称，那么这是唯一要做的工作。

Dialog 'Create Shortcut'



按'OK(确认)'关闭对话框，在资源中插入新的快捷键，用快捷键名称和快捷键要代表这一个新快捷键，此快捷键号比至今工具实例和使用的最高号大。

在名称域的下面，显示有关用户输入的相应的提示。

删除工具快捷键

通过上下文菜单（鼠标右键）或'工程' '对象'菜单中的指令'删除'来完成删除快捷键。只有在对当前标明的快捷键没有定义快捷键的固定号码，才能用这个指令。如果删除快捷键，保留下的快捷键的号码不会改变。

执行工具快捷键

在资源树的这个条目上双击鼠标执行快捷键，或者在'工程' '对象'菜单或上下文菜单（鼠标右键）中通过指令'打开对象'来执行快捷键。

如果在快捷键特性（参数）中定义的文件执行失败，那么将出现相应的错误信息。如果没有发现参数文件，那么将执行工具的可执行文件，并且打开对话框，询问是否要创建文件。

如果在定义的路径中没有发现工具的可执行文件或者没有明确路径，那么将打开选择文件的对话框，并

且要求用户输入可执行文件的路径。当用 OK 关闭对话框时，将保存路径，并且在其它 CoDeSys 工程中也可以使用此工具路径。

保存工具快捷键

当保存 CoDeSys 工程时，在资源中保存'工具'文件夹的状态和设置。

注意： 如果通过'另存为(Save as)'用新名称保存工程，那你必须考虑是否使用在参数文件和要下载的文件定义的模板$(PROJECT_NAME)，如果在旧工程中已读为工具添加了快捷键，那么在新工程中对应于新的工程名必须手工重命名此文件。与用快捷键的固定号码定义的工具相比，当前工程名将自动取代此模板！

关于工具的常见问答

在资源中我为什么不能得到'工具'条目 ？

只有在当前设定的目标系统的目标文件中定义了工具才能使用，'工具'功能。

哪些工具可用快捷键，哪些快捷键可以添加到 CoDeSys 工程中?

在对象管理器的资源树中双击加号，打开工具文件夹。你便可看见哪些工具已经和当前工程连接。如果你刚建立一个新的工程，还没有使用工具表，那么仅仅显示在目标文件中被定义为不可改变的条目。否则你可以看见一个已经抓们修改过的工具列表。为了检查是否可扩展此列表，选择指令'添加对象'。你将得到一个可以提供创建快捷键的所有工具的对话框

有效的工具有哪些全局特性?

在对象管理器中标明'工具'条目，并且从上下文的菜单（鼠标右键）选择指令'对象属性'。点'高级(Advanced)'按钮处延现的对话框。可以看见可用工具的列表和相应的参数。用鼠标点击 ID 符号来选择其中的一个工具，例如，如果激活快捷键，那么在域'FixedCount'中显示工具的快捷键允许的号码，或者显示下载到 PLC 的文件。可以以模板形式显示文件名或文件路径，对于每个快捷键，都要用模板解释：

有效的快捷键有哪些特性?

在对象管理器中标明'Tools'下面的条目中的一个，并且从上下文菜单的菜单（鼠标右键）选择指令'对象属性(Object Properties)'。点'高级(Advanced)'按钮得到被选快捷键的参数。其中一部分对应于上述描述的全局工具特性。如果目标文件中定义允许，可以编辑这些参数。

怎样为工具创建快捷键?

在对象管理器中标明'工具'条目，并且从上下文的菜单（鼠标右键）选择指令'添加对象'。你可以看见一列可用的工具，但是它们的号码没有达到快捷键的最大号码。选择一个工具并且按 OK。此工具将被插入对象管理器中工具文件夹。如果想再次插入它，那么首先必须输入不同的工具名，这意味着把这个新的条目标志为相同的工具。例如，你可以给工具 Toolxy 的实例命名为"Toolxy_1", "Toolxy_2" 等。

怎样修改工具参数?

为了修改快捷键的参数（工具连接的实例），在对象管理器中标出快捷键并且从上下文的菜单中选择指令'对象属性' 。在属性对话框中是否可以编辑参数取决于目标文件中工具的预定义。（看扩展对话框中是否激活了选项'可编辑(Editable)'）。按钮'标准(Standard)'把所有编辑的数值复位到默认值。

怎样执行工具快捷键?

在对象管理器快捷键条目上双击或者当条目在对象管理器中被标出时在上下文菜单或在菜单'工程'中选择指令'打开对象' 。

7. ENI

什么是 ENI

ENI（工程接口）允许外部数据库连接到 CodeSys 编程系统中。从而，创建自动化工程时所需要的外部数据就可以存储，外部数据库的使用保证了数据的连续性，这样就能被多个用户、工程以及程序共享。同时通过以下条款也扩展了 CODeSys 的功能：

CodeSys 版本及其他相关资源的控制（共享对象）：如果在数据库以外检查一个项目，那么当该项目被更改并再次检查后，关于该项目的新的版本信息将在数据库中被创建，而旧的版本仍然会保留在数据库中，并可以被调用打开。对于每个工程以及整个项目，版本的历史记录都会被记录为日志，并且不同版本还可以检查不同的地方。

多用户操作：最新版本的项目，比如项目中的一个工程，可以被一组用户访问。此时，被其中一个用户访问、修改过的项目将会被标记以“被更改”，其他用户均不再能够编辑。这样，多个用户就能同时在相同的项目中工作，而不会有默认覆盖最新版本的风险。

外部工具可访问：B 在 CoDeSys 软件以外的工具中，只要提供了 ENI 功能，就可以访问通用的数据库。该外部工具（例如，可视化，ECAD 系统等等）需要在 CodeSys 中创建自身调用时所需的数据库。

ENI 由客户机和服务器两部分组成。如果需要多用户操作，可以通过远程计算机登陆数据库。CoDeSys 编程系统是相对独立的 ENI 服务器的一个客户端。其处理过程相当于一个要求访问数据库的请求。(请参考关于 ENI 服务器的文档)。

当前ENI支持的数据库系统为’Visual SourceSafe 6.0’, ’MKS Source Intergrity’, ’PVCS Version Manager’ V7.5 或更高版本，还有本地的文件系统。对象们可以储存在不同的’文件夹’。(数据库目录具有不同的访问权限)。可以选中一个对象来进行编辑，与此同时此对象相对于其他用户进入锁定状态。数据库可以调用一个对象的最新版本。更进一步说用户可以在本地工程中存储任何对象，就像没有来源控制的工程。

参照：

对ENI项目的数据库进行操作的预处理

在CoDeSys中运行ENI项目的数据库。

工程数据库中的项目种类

使用ENI工程数据库的前提

注意：关于 ENI 服务器的安装和使用工具，由 3S - Smart Software Solutions GmbH 公司提供，请进入在线帮助，那里您可以找到快速链接向导。考虑到使用 ENI 浏览器，可能会允许在当前使用的数据库系统中执行独立的数据库操作。

如果用户想在 CODESYS 系统中通过使用 ENI 的外部数据库来管理工程项目，必须完成以下一些预先操作：

CoDeSys 和 ENI 服务器之间的通讯需要进行 TCP/IP 配置，因为 ENI 服务器使用 HTTP 协议。

ENI 服务器必须在本地或远程的计算机上安装和启动，而服务器中安装的标准数据库驱动器在运行时必须具有授权认证，只有在本地文件系统中的驱动器可以在没有授权的 ENI 服务器中使用。

ENI 服务器的服务工具（ENI 控件）所链接的数据库必须配置正确，在安装时用户可以根据安装的提示自动进行安装，但也可以在以后的 ENI 控件中更改设置。

必须安装创建基于支持 ENI 驱动器的工程数据库，且最好安装是在 ENI 服务器运行的同一台计算机上，作为选择，可以使用基于默认驱动器的本地文件系统。

数据库的管理员以及其他用户（客户端），访问数据库时必须以特定的权限事先注册为具有权限的用户。如果用户想使用另一个数据库系统，该规定就成为一个必要的可视安全资源。在用户配置前请注意阅读相应的文件以获取必要的信息。

在当前的 CoDeSys 项目中，必须激活 ENI 接口。(在 CoDeSys 的选项'工程'\ '选项' '工程数据库'对话框中）也可以切换到 ENI 中定义的其他用户，比如，为了定义更为详细的信息可以切换进入数据库的管理员权限，但通常，如果用户仅仅想要通过 ENI 服务器登陆数据库时，只要正常登陆就足够了。

在当前的CoDeSys项目中，同数据库的链接必须配置正确。这些操作可在'工程' '选项' '工程源控制对话框中完成。.

当前工程下的用户必须以正确的用户名和密码才能登陆 ENI 服务器，这些可以在登陆对话框中完成，也可以在外部打开，使用命令'工程' \'数据库连接' \'登录'配置中，该对话框将会在用户没有登陆而想进入数据库时自动弹出。

参照：

什么是ENI?

在CoDeSys中运行ENI项目的数据库。

工程数据库中的项目种类。

在CoDeSys中运行ENI项目的数据库

工程数据库命令（得到最新的版本，可检查版本历史、标记文本等）用来管理在ENI工程数据库库内的工程项目，只要在当前的CoDeSys项目中设置正确，一旦链接到数据库中并且被激活后就是可以获取的。具体可参见 对ENI项目的数据库进行操作的预处理 。指令就可以在子菜单“数据库连接” 的上下级菜单或者指向当前在目标组织中标记的目标“项目”菜单中任意使用。

当前的目标 数据库目录任务 可以在对象属性中显示和更改，数据库目录的属性（通讯参数、访问权限、登录/离开操作）都可以在工程数据库的选项框中更改（('工程' '选项' '工程源控制')。

参照：

什么是ENI?

对ENI项目的数据库进行操作的预处理

工程数据库中的项目种类

工程数据库中的项目种类

在一个 Codesys 工程的资源选项中有四个项目种类分类:

ENI在工程的数据库管理种分出三个类（ENI项目种类)：工程对象（Project objects）,共享对象（Shared objects）,编译文件（Compile files）

“本地”分类：如果不是存储在服务器中，项目将被分配其中，这表明该项目将作为不具有来源控制的项目被处理。

因而，在编程系统中，一个CoDesys项目可以被分派到工程对象（Project objects）,共享对象（Shared objects）或者本地项目之中。,编译文件（Compile files）选项不作为工程中的可选项目。当工程被创建时项目会被自动指定到分类的相应项目中去，因为这些操作都在 '工程源控制 “工程\数据库连接”或“多用途定义”中事先定义了。

每个 ENI 项目分类都会在”项目资源控制” 的设定选项中单独予以组态，这是工程选项的一部份（工程\选项：'Project' 'Options')。这就意味着每个分类自身的数据通过数据库（包括目录、端口、访问权限、用户访问数据等）得到定义，并且根据不同的操作实现最新版本调用、登陆、校验等操作。这些设置可以在属于该种类下的所有项目中有效，用户必须分别登陆进数据库的每个目录中去，这样就不用通过每个登陆对话框完成设定。

用户最好在数据库中为每个项目种类创建一个独立的文件夹，但是也可以将所有项目储存在同一个文件夹中（“种类”只是项目的属性选项，而不能作为文件夹的属性)。

参考下面三个 ENI 对象种类:

工程对象: 包含工程的详细资源信息的对象，比如：为多用户操作共享的工程单元（POUs)；指令“起用所有新版本”将会从数据库自动调用所有该种类的项目到本地工程中，即使这些本地工程当前并

不存在；

共享对象：　　不包含详细资源信息的工程对象，比如：在多个工程中都会用到的工程库（POU libraries）；
注意：指令“起用所有新版本”将仅仅在从数据库中拷贝该种类的项目到已经在工程中存在的本地项目中！

编译文件：　　由 CoDeSys 为当前项目创建的编译信息（比如符号文件），以及其他程序可能需要的信息。
例如：外部的可视化文件可能即在工程变量中需要，也在分配的地址中需要，而后者直到整个项目完成编译才知道。

作为选择的，所有 CoDeSys 工程的项目 都可以在项目资源控制选项之外被设定为“本地”的种类，这表明这些项目仅作为工程中的项目被存储而不具有来源控制。

参照：

什么是ENI？

对ENI项目的数据库进行操作的预处理

在CoDeSys中运行ENI项目的数据库

8. DDE接口

DDE 和CoDeSys的通讯

CoDeSys 有一个 DDE（动态数据交换）接口可以用来读取数据。CoDeSys 应用这个接口提供一些其它的应用，如用 DDE 接口实现变量和 IEC 地址内容的控制。

如果以符号工作的 DDE 网关服务器打开，CoDeSys 就不需要从 PLC 中读取变量和用 DDE 接口把它们传送给应用程序。

注意：不能通过 DDE 服务器来读取直接地址！因此用户需要在 CoDeSys 下定义变量，用此变量分配给需要的地址(AT)。

注意：DDE 接口已经完成在 Windows NT 4.0 系统下的 Word 97 和 Excel 97 测试。如果 DDE 通讯失败是由于电脑上安装的其它额外版本，3S 公司将不负任何责任。

8.1 CoDeSys 程序设计系统的 DDE 接口

激活DDE接口

当PLC（或 模拟模式）登陆时DDE接口将被激活。

数据连接的普通方法

访问 DDE 可以分为三个部分：

1. 程序的名称 （这里：CoDeSys）
2. 文件名称
3. 要读取的变量名称

程序名称: CoDeSys

文件名称: 工程编译路径 (c:\example\example.pro).

变量名称: 在监视和收到管理器中出现的变量名称 .

可以读取什么样的变量？

可以读取所有的变量和地址。变量或地址应该在监视和收到管理器的应用格式下输入。

例如:

```
%IX1.4.1      (* 读取输入 1.4.1*)
PLC_PRG.TEST (* 读取 POU PLC_PRG 中的变量 TEST*)
.GlobVar1     (* 读取全局变量 GlobVar1  *)
```

用WORD来连接变量

为了在微软 WORD 下通过 DDE 接口得到 POU PLC_PRG 下变量 TEST 的当前值，必须插入一个 WORD 类型的区域('插入'"区域")。然后当你用鼠标右键点击此区域和选择命令"连接区域代码"时，你可以改变选择文本的区域功能。下面是一个例子：

{ DDEAUTO CODESYS "C:\CODESYS\PROJECT\IFMBSP.PRO" "PLC_PRG.TEST" }

再一次在此区域上点击鼠标右键，然后点击"更新区域"，则期望的变量内容出现在文本框内。

用EXCEL来连接变量

下面在用户分配一个变量给一个单元前必须进入微软的 EXCEL。

=CODESYS|'C:\CODESYS\PROJECT\IFMBSP.PRO'!PLC_PRG.TEST'

当用户点击'编辑' "连接"时，连接的结果如下：

类型: CODESYS
源文件: C:\CODESYS\PROJECT\IFMBSP.PRO
元素: PLC_PRG.TEST

用Intouch来访问变量

应用 CODESYS 名称和 DDE 主题名称连接用户的 DDE 数据库名称<数据库名称>。C:\CODESYS\PROJECT\IFMBSP.PRO.

此时用户可以用数据库名称<数据库名称>和 DDE 变量类型来连接。输入变量名称作为项目名称。(例如., PLC_PRG.TEST)

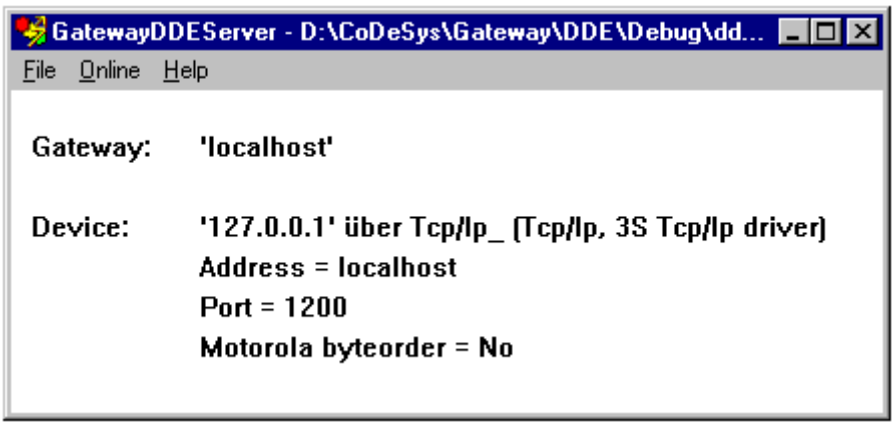
8.2 DDE网关服务器的DDE信息

DDE网关服务器的操作

为了实现工程和其它客户端或PLC的通讯，在CoDeSys下需要创建一个符合，DDE网关服务器可以应用这个符号。(参考 '工程' '选项' '符号配置'). 它可以像EXCEL一样提供DDE接口应用。这样就可以传送一个PLC的变量值给一个应用程序。例如，可以用来监视。

DDE 网关服务器开始时打开一个窗口，在此可以配置开始需要的通讯参数。一个已经存在的配置文件可以被调用或其参数可以被更新。

DDE网关服务器的开始对话框



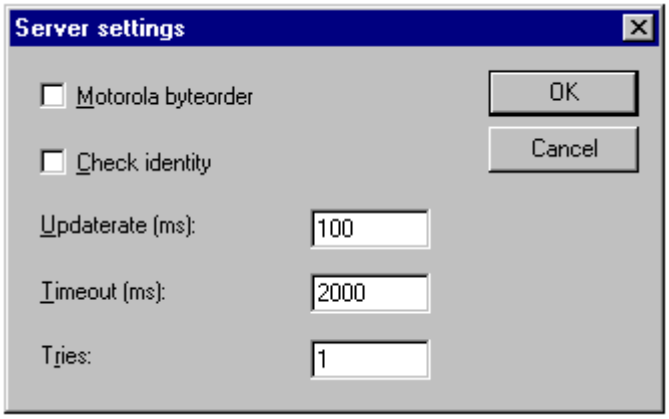
应用'文件' '打开'命令可以调用一个存储了参数配置信息的文件。此时将弹出一个标准的对话框来选择要打开的文件，扩展名为".cfg"的文件时可用的。如果选中一个配置文件，其配置参数和定义的目标设备将显示出来。

如果选项'文件' '自动加载'被激活，则 DDE 网关服务器将自动打开上一次关闭服务器时的配置信息。

如果没有预先确定配置文件或选择自动加载而开启服务器，则将显示配置窗口'网关:'和'设备:'。然后用户需要设置新的配置信息。

'文件' '设置'命令打开'服务器配置'对话框，在此可以设置下面的参数：

配置DDE网关服务器的对话框



摩托罗拉位命令	已经应用摩托罗拉位命令
检查特性	将检查符号所给的工程 ID 是否和 PLC 中存储的一致。
更新速率 [ms]	从 PLC 中读取所有符号变量的时间间隔。
超时时间[ms]	和用户驱动器连接的超时时间。
尝试	传递数据块时连接驱动器的尝试次数。(不是所有的驱动器都支持 !)

设置网关连接，’联机’ ’参数’命令打开’通讯参数’对话框。它和用户在CoDeSys下通过 ’联机’ ’Kommunikationsparameter’命令得到的对话框一样。在这里进行的设置必须要和CoDeSys工程下的设置相一致。

可以通过命令’文件’ ’保存’来存储当前的 DDE 网关服务器配置信息为一个文件。将打开一个用来存储的标准对话框，存储文件的默认扩展名是*.cfg。

为了使网关进入激活模式，可通过’联机’ ’登陆’命令来登陆。(在状态条下的网关符号将随后被释放。)在登陆时，期望的连接将被建立，且可以访问可用的符号。这些必须在先前的 CoDeSys 工程下创建好。

用’联机’ 退出’来退出登陆。

数据连接的普通方法

访问 DDE 可以分为三个部分：

1. 程序的名称
2. 文件名称
3. 要读取的变量名称

程序名称: DDE 网关服务器

文件名称: 要读取的变量所属工程的名称 (例如. example.pro).

变量名称: 在监视和收到管理器中出现的变量名称 .

可以读取什么样的变量?

可以读取所有的变量。变量应该在监视和收到管理器的应用格式下输入。注意不能直接读取地址!

例如:

```
PLC_PRG.TEST    (* 读取 POU PLC_PRG 中的变量 TEST*)
.GlobVar1       (* 读取全局变量 GlobVar1 *)
```

用WORD连接变量

在打开 WORD 前启动 DDE 网关服务器。

为了在微软 WORD 下通过 DDE 接口得到 POU PLC_PRG 下变量 TEST 的当前值，必须插入一个 WORD 类型的区域(’插入’ "区域")。然后当你用鼠标右键点击此区域和选择命令"连接区域代码"时，你可以改变选择文本的

区域功能。下面是一个例子：

{ DDEAUTO GATEWAYDDESERVER "BSP.PRO" "PLC_PRG.TEST" }

再一次在此区域上点击鼠标右键，然后点击"更新区域"，则期望的变量内容出现在文本框内。

用EXCEL连接变量

在打开 EXCEL 前启动 DDE 网关服务器。

在你把变量分配给一个单元前，下面的代码必须输入到微软的 EXCEL 中。

=GATEWAYDDESERVER|<Dateiname>!<Variablenname>

Beispiel:

=GATEWAYDDESERVER|'bsp.pro'!'PLC_PRG.TEST'

当你点击‘编辑’、‘连接’时，此连接的结果是：

类型: CODESYS

源文件: C:\CODESYS\PROJECT\IFMBSP.PRO

成分: PLC_PRG.TEST

{ DDEAUTO GATEWAYDDESERVER "BSP.PRO" "PLC_PRG.TEST" }

DDE网关服务器的命令行选项

如果用命令行启动 DDE 网关服务器，则可以附加下列选项：

/n	在开始时对话框信息不自动出现。		
/s	显示对话框窗口	/s=h	无
		/s=i	最小（图标）
		/s=m	最大
		/s=n	普通
/c	自动登陆配置文件	/c=<配置文件>	
/o	以选中的配置来进行联机运行（自动登陆或通过"/c="来定义）		

例如:

命令行如下:

GATEWAYDDE /s=i /c="D:\DDE\conf_1.cfg"

DDE 网关服务器将被启动，对话框窗口将像一个图标一样出现，保存在文件 conf_1.cfg 中的配置将被装载。

9. CoDeSys的许可证管理器

许可证管理器

在计算机上,3S 的许可证管理器用于管理 3S 模块的许可证,以及提供相应许可证信息文件的模块的许可证.在CoDeSys中,创建一个工程,然后通过带有许可证的库文件提供给用户.在安装任一需要许可证的3S模块后,许可证管理器将自动被安装.

也可参看:

3S 许可证管理器文档

在 CoDeSys 中创建一个有许可证的库文件

在 CoDeSys 中创建一个有许可证的库文件

众所周知 CoDeSys 工程可以保存成一个库文件. 如果想要创建一个有许可证的库文件,需要添加相应的许可证信息. 为了生成库文件,执行命令 “文件”另存为... 选择数据类型 ’内部库’或 ’外部库’, 然后按编辑许可证信息....

在编辑许可证信息对话框中按下面描述输入信息. 许可证信息将添加到工程信息中. 然后当库文件包含到工程后，库管理器中的库对象属性对话框中检查许可证信息.

编辑许可证信息对话框

一般信息:

名称: 输入在 3S 许可证管理器中显示的库模块名称,必须输入名称.

供应商-ID: 制造商的标识号,它取决于制造商的许可证管理工具.

许可证不受限制的方式: 如果这个模块在演示模式下运行,激活这个选项,即不需要许可证 ID 号.输入″演示许可证″到期的天数. 许用天数按 10 递增,如(10,20,30...). 如果模块无时间限制,选择’无限制’(可在列表中选择).

目标: 输入许可证有效的目标系统的目标 ID. 可以分号分开输入多个 ID 或一个范围. 如: ″12;15-19;21″; 输入的有效 ID 号是 12,15,16,17,18,19,21.

联系方式:

借助电话获得许可证: / 借助邮件获得许可证: 输入许可证提供者的电话或电子邮件.必须输入.

可选信息:

在右边窗口,可以按照左边窗口输入相关文字,如:描述,制造商,供应商,价格信息.

注意:

1. 可以通过密码方式加密带有许可证信息的库文件.如果采用无密码方式保存工程,将会弹出一个信息窗口.

2. 3S 库的许可证信息与库文件一同保存,只要库文件包含到工程中,库就自动注册到计算机中.但是不是 3S 提供的模块的许可证信息被保存在单独的文件中,它与 XML 格式兼容, 可由 3S 许可证管理器读取.

也可参看 3S 许可证管理文档.

10. 附录

A:IEC操作符和额外的标准扩展功能块

CoDeSys 支持所有 IEC 符号，与标准功能相对应，这些符号在整个项目中服从内部组织在一起。除了 IEC 符号，CoDeSys 还支持

如下符号，它们没有列入标准中，这些符号像 POU 中的功能一样。

注意:浮点变量的运算结果依赖预当前使用的对象系统 ！

算术操作符

位串操作符

移位操作符

选择操作符

比较操作符

地址操作符

调用操作符

类型变换

数字操作符

10.1 算术操作符

ADD

CoDeSys IEC 操作符:

相加变量的类型: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL 和 LREAL.

两个时间量也可以相加得出另一个时间量。

(例如： t#45s + t#50s = t#1m35s)

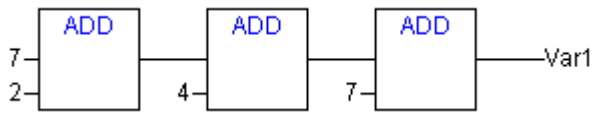
IL 例子:

```
LD  7
ADD 2,4,7
ST  Var1
```

ST 例子:

```
var1 := 7+2+4+7;
```

FBD 例子:



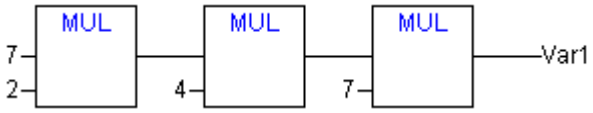
MUL

CoDeSys IEC 操作符:

相乘变量的类型: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL 和 LREAL.

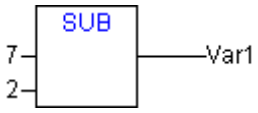
IL 例子:

LD 7
MUL 2,4,7
ST Var1
ST 例子:
var1 := 7*2*4*7;
FBD 例子:



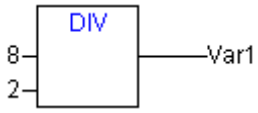
SUB

CoDeSys IEC 操作符:
相减变量的类型: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL 和 LREAL.
一个时间变量与另一个时间变量相减得出第三个时间类型变量。注意负时间值无意义。
IL 例子:
LD 7
SUB 2
ST Var1
ST 例子:
var1 := 7-2;
FBD 例子:



DIV

CoDeSys IEC 操作符:
相除变量的类型: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL 和 LREAL.
IL 例子:
LD 8
DIV 2
ST Var1 (* 结果是 4 *)
ST 例子:
var1 := 8/2;
FBD 例子:



注意: 若将项目中 CheckDivByte, CheckDivWord, CheckDivDWord 和 CheckDivReal 的名字的功能加以定义, 如果使用 DIV 除数的值，便可以用它们来检查, 例如，为了避免被 0 除. 上述功能必须具有上述的列名字.
注意: 请注意, 不同的目标系统因被 0 除而有不同的表现 !
请参考下例中 CheckDivReal 功能的应用:
Example for the implementation of the function CheckDivReal:

```
    FUNCTION CheckDivReal : REAL
VAR_INPUT
 divisor:REAL;
END_VAR
    IF divisor = 0 THEN
 CheckDivReal:=1;
ELSE
 CheckDivReal:=divisor;
END_IF;
```

DIV 符使用 CheckDivReal 的输出功能作除数. 在下面例子中的程序里，这样可以避免除 0, 除数(d)被设为 0 — 1. 所有除后的结果是 799.

```
    PROGRAM PLC_PRG
VAR
 erg:REAL;
 v1:REAL:=799;
 d:REAL;
END_VAR
    erg:= v1 / d;
```

注意: 用 Check.Lib 库提供的 CheckDiv-functions 为解决类似的例子!在使用库模块前，检查它们是否按你的意思运行, 或者像项目中的 PUU 一样直接运行相应的功能.

MOD

CoDeSys IEC 操作符:

模块分割的两个变量类型: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT. 此功能的结果则是此项分割的余数，其结果将是整个数字.

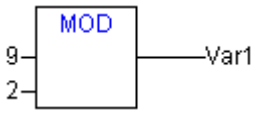
IL 例子:

```
LD  9
MOD 2
ST  Var1 (* Result is 1 *)
```

ST 例子:

```
var1 := 9 MOD 2;
```

FBD 例子:



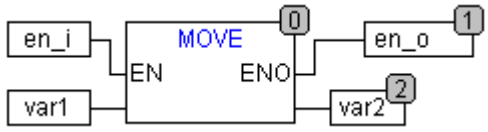
MOVE

CoDeSys IEC 操作符:

一个适当的类型从一个变量分配到另一个变量中.由于 Move 在图表编辑 LD,CFC 的框盒子里, 此处(未加锁的) EN/ENO 功能 也可以用于变量分配中，在 FBD 编辑器里这个则是不可能的。

CFC 例子带 EN/ENO 功能:

只有当 en_i 是 TRUE, var1 将赋值给 var2.



IL 例子:

```
LD ivar1
MOVE
ST ivar2 (* Result: ivar2 gets assigned value of ivar1 *)
```

(! 你会得到相同的结果:

```
LD ivar1
ST ivar2 )
```

ST 例子:

```
ivar2 := MOVE(ivar1);
```

(! 你会得到相同的结果: ivar2 := ivar1;)

INDEXOF

CoDeSys 操作符:

此功能未列入 IEC61131-3 标准.执行这项功能用来找到 POU 中的内部索引。

ST 例子:

```
var1 := INDEXOF(POU2);
```

SIZEOF

CoDeSys 操作符:

此功能未列入 IEC61131-3 标准.执行此功能块可以判断所给变量要求达到的字符数。

IL 例子:

```
arr1:ARRAY[0..4] OF INT;
Var1 INT
LD arr1
SIZEOF
ST Var1 (* 结果是 10 *)
```

ST 例子:

```
var1 := SIZEOF(arr1);
```

10.2 位串操作符

AND

CoDeSys IEC 操作符:

位操作符的 AND。其类型应该属于 BOOL, BYTE, WORD 或 DWORD.

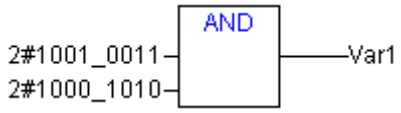
IL 例子:

```
Var1 BYTE
LD 2#1001_0011
AND 2#1000_1010
ST Var1 (* 结果是 2#1000_0010 *)
```

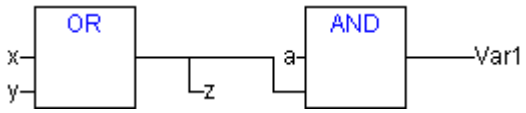
ST 例子:

```
var1 := 2#1001_0011 AND 2#1000_1010
```

FBD 例子:



注意: 若你把一个程序插入 SFC，如图所示



如果用 68xxx- 或 C-语言 生成程序,请注意：在AND符号模块的第二个输入变量值分配给变量Z时不执行。此因SFC中的优化处理过程，以免输入变量出现FALSE值。

OR

CoDeSys IEC 操作符:

位操作符的OR。其类型应该属于 BOOL, BYTE, WORD 或 DWORD.

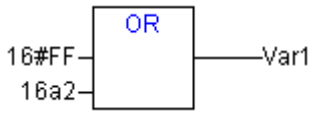
IL 例子:

```
var1 :BYTE;
LD 2#1001_0011
OR 2#1000_1010
ST  var1 (* 结果是 2#1001_1011 *)
```

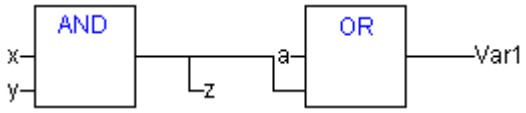
ST 例子:

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

FBD 例子:



注意：若你把一个程序插入SFC，如图所示



如果用 68xxx- 或 C-语言 生成程序,请注意：在AND符号模块的第二个输入变量值分配给变量Z时不执行。此因SFC中的优化处理过程，以免输入变量出现FALSE值。

XOR

CoDeSys IEC 操作符:

位操作符的XOR。其类型应该属于 BOOL, BYTE, WORD 或 DWORD.

注意：把XOR的作用行为视为其延伸形式, 也即如果出现2次以上的输入. 输入符号会被成对检查，所得的数据结果也会再次被成对检查核对。 (这与标准相一致,但也许并不是用户所期望的).

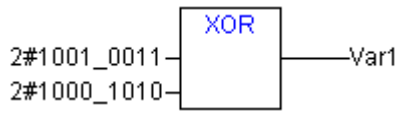
IL 例子:

```
Var1 :BYTE;
LD 2#1001_0011
XOR 2#1000_1010
ST  Var1 (* 结果是 2#0001_1001 *)
```

ST 例子:

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

FBD 例子:



NOT

CoDeSys IEC 操作符:

位操作符的 NOT。其类型应该属于 BOOL, BYTE, WORD 或 DWORD.

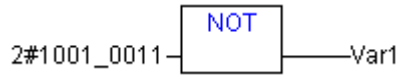
IL 例子:

```
Var1 :BYTE;
LD 2#1001_0011
NOT
ST  Var1 (* 结果是 2#0110_1100 *)
```

ST 例子:

```
Var1 := NOT 2#1001_0011
```

FBD:



10.3 移位操作符

SHL

CoDeSys IEC 操作符: 操作数的位左移 : erg:= SHL (in, n)

in 向左位移 n 个位. 如果 n > 大于数据类型的宽度, BYTE, WORD 和 DWORD 将用 0 来填补. 但是如果使用有符号的数据类型, 如 e.g. INT, 那么此种情况下会执行一次运算位移, 也即它会由最上边的位值占位.

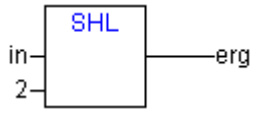
注意: 请注意位的数量, 它是用来作算术计算的, 然而却是以输入变量的数据类型的形式出现的。若输入变量是个常数则被视为最小数据类型。输入变量的数据类型对算术计算毫无影响。

注意: 见如下例子中十六进制位符号,根据输入变量数据类型的不同(BYTE 或 WORD),会得到 erg_byte 和 erg_word 的不同结果，尽管此时的 in_byte 和 in_word 输入变量的值相同。

ST 例子:

```
PROGRAM shl_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
erg_byte:=SHL(in_byte,n); (* 结果是 16#14 *)
erg_word:=SHL(in_word,n); (* 结果是 16#01141 *)
```

FBD 例子:



IL 例子:

```
LD  16#45
SHL 2
ST  erg_byte
```

SHR

CoDeSys IEC 操作符: 操作数的位右移 : erg:= SHR (in, n)

in 向右位移 n 个位. 如果 n > 大于数据类型的宽度, BYTE, WORD 和 DWORD 将用 0 来填补. 但是如果使用有符号的数据类型, 如 e.g. INT, 那么此种情况下会执行一次运算位移, 也即它会由最上边的位值占位.

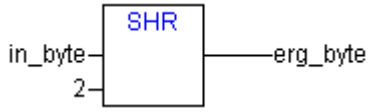
注意: 请注意位的数量, 它是用来作算术计算的, 然而却是以输入变量的数据类型的形式出现的。若输入变量是个常数则被视为最小数据类型。输入变量的数据类型对算术计算毫无影响。

见如下例子中十六进制位符号，根据输入变量数据类型的不同(BYTE 或 WORD)，会发现算术运算的结果。

ST 例子:

```
PROGRAM shr_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
    erg_byte:=SHR(in_byte,n); (* 结果是 11 *)
    erg_word:=SHR(in_word,n); (* 结果是 0011 *)
```

FBD 例子:



IL 例子:

```
LD  16#45
SHR 2
ST  erg_byt
```

ROL

CoDeSys IEC 操作符: 操作数的位循环左移: erg:= ROL (in, n)

erg, in 和 n 应该为 BYTE, WORD 或 DWORD 类型. in 将向左方向移动一个位，移动 n 次，而距离左边最远的位符，将被再次从右边插入。

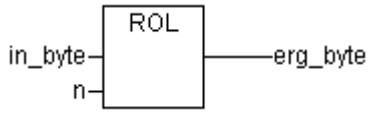
注意: 请注意位的数量, 它是用来作算术计算的, 然而却是以输入变量的数据类型的形式出现的。若输入变量是个常数则被视为最小数据类型。输入变量的数据类型对算术计算毫无影响。

见如下例子中十六进制位符号，根据输入变量数据类型的不同（BYTE 或 WORD)，会得到 erg_byte 和 erg_word 的不同结果，尽管此时的 in_byte 和 in_word 输入变量的值相同。

ST 例子:

```
PROGRAM rol_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
    erg_byte:=ROL(in_byte,n); (* 结果是 16#15 *)
    erg_word:=ROL(in_word,n); (* 结果是 16#0114 *)
```

FBD 例子:



IL 例子:

```
LD  16#45
ROL 2
ST  erg_byte
```

ROR

CoDeSys IEC 操作符: 操作数的位循环右移: erg = ROR (in, n)

erg, in 和 n 应该为 BYTE, WORD 或 DWORD 类型. in 将向右方向移动一个位，移动 n 次，而距离右边最远的位符，将被再次从左边插入。

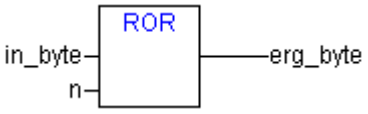
注意: 请注意位的数量, 它是用来作算术计算的, 然而却是以输入变量的数据类型的形式出现的。若输入变量是个常数则被视为最小数据类型。输入变量的数据类型对算术计算毫无影响。

见如下例子中十六进制位符号，根据输入变量数据类型的不同（BYTE 或 WORD），会得到 erg_byte 和 erg_word 的不同结果，尽管此时的 in_byte 和 in_word 输入变量的值相同。

ST 例子:

```
PROGRAM ror_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
    erg_byte:=ROR(in_byte,n); (* 结果是 16#51 *)
    erg_word:=ROR(in_word,n); (* 结果是 16#4011 *)
```

FBD 例子:



IL 例子:

```
LD 16#45
ROR 2
ST erg_byte
```

10.4 选择操作符

所有选择操作也都可以使用变量来操作。为了简明，我们只选择如下方案，其中的操作符为常数。

参照:

SEL

MAX

MIN

LIMIT

MUX

SEL

CoDeSys IEC 操作符: 位选择.

OUT := SEL(G, IN0, IN1) 也就是说:

OUT := IN0 if G=FALSE;

OUT := IN1 if G=TRUE.

IN0, IN1 和 OUT 可为任何变量类型, G 必须是 BOOL 型. 选择结果如果 G 是 FALSE 则为 IN0,如果 G 为 TRUE，结果则为 IN1。

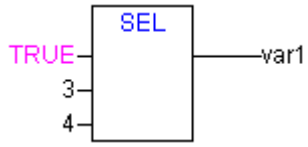
IL 例子:

```
LD  TRUE
SEL 3,4    (* IN0 = 3, IN1 =4 *)
ST  Var1   (* 结果是 4 *)
LD  FALSE
SEL 3,4
ST  Var1   (* 结果是 3 *)
```

ST 例子:

```
Var1:=SEL(TRUE,3,4); (* 结果是 4 *)
```

FBD 例子:



注意: 如果 IN1 或 IN2 前出现符号而 IN0 是 TRUE，则此符号不会被处理。

MAX

CoDeSys IEC 操作符: 最大作用. 返回两个位当中的最大值。

OUT := MAX(IN0, IN1)

IN0, IN1 和 OUT 可以是任何变量数据类型。

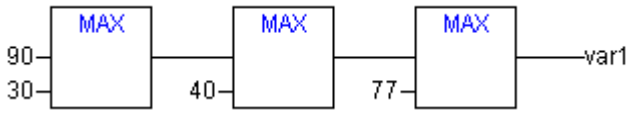
IL 例子:

```
LD 90
MAX 30
MAY 40
MAX 77
ST Var1 (* 结果是 90 *)
```

ST 例子:

```
Var1:=MAX(30,40); (* 结果是 40 *)
Var1:=MAX(40,MAX(90,30)); (* 结果是 90 *)
```

FBD 例子:



MIN

CoDeSys IEC 操作符: 最小作用. 返回两个位当中的最小值。

OUT := MIN(IN0, IN1)

IN0, IN1 和 OUT 可以是任何变量数据类型。

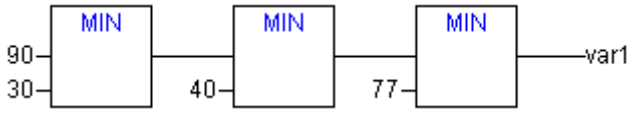
IL 例子:

```
LD 90
MIN 30
MIN 40
MIN 77
ST Var1 (* 结果是 30 *)
```

ST 例子:

```
Var1:=MIN(90,30); (* 结果是 30 *);
Var1:=MIN(MIN(90,30),40); (* 结果是 30 *);
```

FBD 例子:



LIMIT

CoDeSys IEC 操作符: 限制

OUT := LIMIT(Min, IN, Max) 也就是:

OUT := MIN (MAX (IN, Min), Max)

结果中 Max 为上限,Min 为下限。如果 IN 值超过上限 MAX,LIMIT 将会返回 Max. 如果 IN 低于 Min, 其结果仍为 Min.

IN 和 OUT 可以是任何变量数据类型。

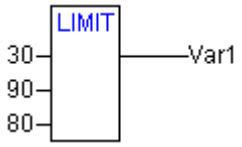
IL 例子:

```
LD 90
LIMIT 30,80
ST Var1 (* 结果是 80 *)
```

ST 例子:

```
Var1:=LIMIT(30,90,80); (* 结果是 80 *);
```

FBD 例子:



MUX

CoDeSys IEC 操作符: 多路器

OUT := MUX(K, IN0,...,INn) 也就是:

OUT := INK.

IN0, ...,INn 和 OUT 可以是任何变量类型。 K 必须是 BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT 或 UDINT.在一组数中 MUX 选择 Kth 值。

Example in IL:

```
LD 0
MUX 30,40,50,60,70,80
ST Var1 (* 结果是 30 *)
```

Example in ST:

Var1:=MUX(0,30,40,50,60,70,80); (* 结果是 30 *);

注意：如果一个输入符前出现了符号而不是 INK ，为了节省时间，此符号将不被处理。只有在模拟模式下的有的符号才会全部被处理。

10.5 比较操作符

GT

CoDeSys IEC 操作符：大于

当第一个操作数值大于第二个时，布尔算子将会返回 TRUE 值，操作数可以是 BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME 和 STRING.

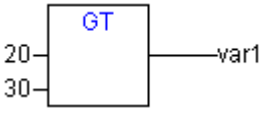
IL 例子：

```
LD 20
GT 30
ST Var1 (* 结果是 FALSE *)
```

ST 例子：

```
VAR1 := 20 > 30 > 40 > 50 > 60 > 70;
```

FBD 例子：



LT

CoDeSys IEC 操作符：小于

当第一个操作数值小于第二个时，布尔算子将会返回 TRUE 值，操作数可以是 BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME 和 STRING.

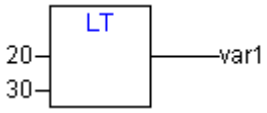
IL 例子：

```
LD 20
LT 30
ST Var1 (* 结果是 TRUE *)
```

ST 例子：

```
VAR1 := 20 < 30;
```

FBD 例子：



LE

CoDeSys IEC 操作符：小于或等于

当第一个操作数值小于或等于第二个时，布尔算子将会返回 TRUE 值，操作数可以是 BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME 和 STRING.

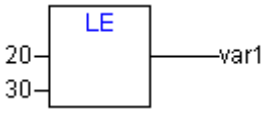
IL 例子：

```
LD 20
LE 30
ST Var1 (* 结果是 TRUE *)
```

ST 例子：

VAR1 := 20 <= 30;

FBD 例子:



GE

CoDeSys IEC 操作符: 大于或等于

当第一个操作数值大于或等于第二个时，布尔算子将会返回 TRUE 值，操作数可以是 BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME 和 STRING.

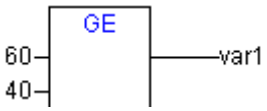
IL 例子:

```
LD 60
GE 40
ST Var1 (* 结果是 TRUE *)
```

ST 例子:

VAR1 := 60 >= 40;

FBD 例子:



EQ

CoDeSys IEC 操作符: 等于

当第一个操作数值等于第二个时，布尔算子将会返回 TRUE 值，操作数可以是 BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME 和 STRING.

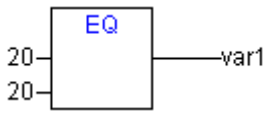
IL 例子:

```
LD 40
EQ 40
ST Var1 (* 结果是 TRUE *)
```

ST 例子:

VAR1 := 40 = 40;

FBD 例子:



NE

CoDeSys IEC 操作符: 不等于

当第一个操作数值不等于第二个时,布尔算子将会返回 TRUE 值,操作数可以是 BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME 和 STRING.

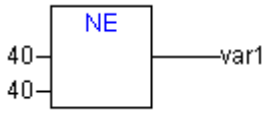
IL 例子:

```
LD 40
NE 40
ST Var1 (* 结果是 FALSE *)
```

ST 例子:

VAR1 := 40 <> 40;

FBD 例子:



10.6 地址操作符

ADR

CoDeSys 操作符: 地址功能未被列入 IEC61131-3 标准。

ADR 返回到 DWORD 中其自变数的地址。此地址会被传到加工功能处作为指针进行处理，或被分配到工程中作为指针。

```
dwVar:=ADR(bVAR);
```

IL 例子:

```
LD bVar
ADR
ST dwVar
man_fun1
```

注意: 在在线更改之后，还可以对某些地址方面的数据加以更改。

ADRINST

CoDeSys 操作符: 地址功能未被列入 IEC61131-3 标准。

功能块程序实例中的 ADRINST 把 DWORD 中程序的地址归位。此地址会被传到加工功能处作为指针进行处理，或被分配到工程中作为指针。

ST 例子 (带有功能块的实例):

```
dvar:=ADRINST();    (* 实例中的地址被写到变量 dvar 中 *)
fun(a:=ADRINST()); (* 实例中的地址作为功能块 fun 的输入变量 *)
```

IL 例子:

```
ADRINST
ST dvar

ADRINST
fun
```

BITADR

CoDeSys 操作符: 地址功能未被列入 IEC61131-3 标准。

BITADR 将 DWORD 中片断内的位偏移较正归位。注意偏移值对应地址目标设定中选项固定器是否被激活而定。

```
VAR
 var1 AT %IX2.3:BOOL;
 bitoffset: DWORD;
END_VAR
```

ST 例子:

```
bitoffset:=BITADR(var1); (* Result if byte addressing=TRUE: 19, if byte addressing=FALSE: 35 *)
```

IL 例子:

```
LD Var1
BITADR
ST Var2
```

注意: 在在线更改之后，还可以对某些地址方面的数据加以更改。

内容操作符

CoDeSys IEC 操作符: 指示器可以采用在操作符“^”后增加内容符的方法废弃。

ST 例子:

```
pt:POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2:=pt^;
```

注意: 在在线更改之后，还可以对某些地址方面的数据加以更改。

10.7 调用操作符

CAL

CoDeSys IEC 操作符: 呼叫功能程序模块或程序。

用 IL 中的 CAL 来调出功能模块程序。把用作输入变量的变量放在功能模块程序名称之后的括号内。

例子:

从功能程序模块中调取 Inst 程序时，输入变量的 Par1 和 Par2 分别为 0 和 TRUE 。

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

10.8 类型变化

类型变换功能块

禁止暗中把“较大”变为“较小”类型。(例如从 INT 转变城 BYTE，或从 DINT 变成 WORD)。若必须改变则需要特殊的类型变换符，不过一般可以把基本类型相互转换。

语法:

<elem.Typ1>_TO_<elem.Typ2>

请注意，在 TO_STRING 变换符处，字符串生成为左对齐形式。如果定义为长度不等，则在右侧截断。

参考:

BOOL_TO 变换
TO_BOOL 变换
积分数字类型间的转换
REAL_TO-/ LREAL_TO 转换
TIME_TO/TIME_OF_DAY 转换
DATE_TO/DT_TO 转换
STRING_TO 转换
TRUNC

BOOL_TO 转换

类型 BOOL 转换为其它类型。

操作数为 TRUE，数字类型结果则为 1，操作数为 FALSE 时，则为 0。就字符串 STRING 类型而言，其结果

为 TRUE 或 FALSE。

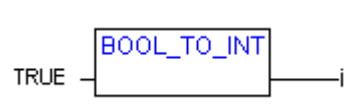
IL 例子:

```
LD TRUE                          (*结果是 1 *)
BOOL_TO_INT
ST i
LD TRUE                          (*结果是 'TRUE' *)
BOOL_TO_STRING
ST str
LD TRUE                          (*结果是 T#1ms *)
BOOL_TO_TIME
ST t
LD TRUE                          (*结果是 TOD#00:00:00.001 *)
BOOL_TO_TOD
ST
LD FALSE                         (*结果是 D#1970-01-01 *)
BOOL_TO_DATE
ST dat
LD TRUE                          (*结果是 DT#1970-01-01-00:00:01 *)
BOOL_TO_DT
ST dandt
```

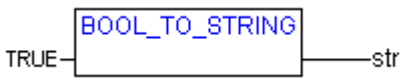
ST 例子:

```
i:=BOOL_TO_INT(TRUE);            (* 结果是 1 *)
str:=BOOL_TO_STRING(TRUE);       (* 结果是 "TRUE" *)
t:=BOOL_TO_TIME(TRUE);           (* 结果是 T#1ms *)
tof:=BOOL_TO_TOD(TRUE);          (* 结果是 TOD#00:00:00.001 *)
dat:=BOOL_TO_DATE(FALSE);        (* 结果是 D#1970 *)
dandt:=BOOL_TO_DT(TRUE);         (* 结果是
                                 DT#1970-01-01-00:00:01 *)
```

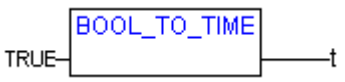
FUP 例子:



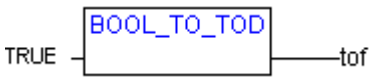
(*结果是 1 *)



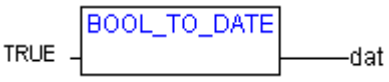
(*结果是 'TRUE' *)



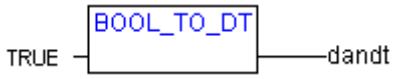
(*结果是 T#1ms *)



(*结果是 TOD#00:00:00.001 *)



(*结果是 D#1970-01-01 *)



(*结果是
DT#1970-01-01-00:00:01 *)

TO_BOOL 转换

从另一变量类型变换为 BOOL:

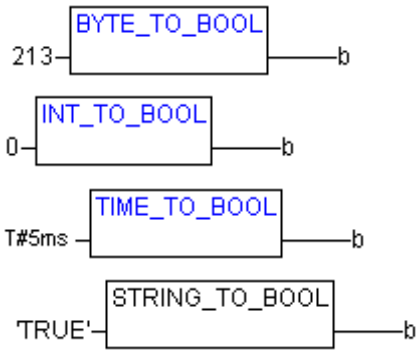
当操作数不为 0 时，其结果为 TRUE ，当操作数为 0 时，则其结果为 FALSE。

当操作数为“TRUE”时，STRING 类型变量的结果是正确的，否则其结果则为 FALSE（错误）。

IL 例子:

```
LD 213                          (*结果是 TRUE *)
BYTE_TO_BOOL
ST b
LD 0                            (*结果是 FALSE *)
INT_TO_BOOL
ST b
LD T#5ms                        (*结果是 TRUE *)
TIME_TO_BOOL
ST b
LD 'TRUE'                       (*结果是 TRUE *)
STRING_TO_BOOL
ST b
```

FUP 例子:



(*结果是 TRUE *)

(*结果是 FALSE *)

(*结果是 TRUE *)

(*结果是 TRUE *)

St 例子:

```
b := BYTE_TO_BOOL(2#11010101);    (* 结果是 TRUE *)
b := INT_TO_BOOL(0);              (* 结果是 FALSE *)
b := TIME_TO_BOOL(T#5ms);         (* 结果是 TRUE *)
b := STRING_TO_BOOL('TRUE');      (* 结果是 TRUE *)
```

积分数字类型间的转换

从一个积分类型变换为另一个数字类型：

类型变换从大到小，会出现数据丢失的状况。如果要变换的数字超出一定范围，数字的首批字节会被忽略掉。

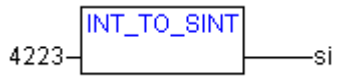
ST 例子:

```
si := INT_TO_SINT(4223); (* 结果是 127 *)
```

如果把整数 4223 存为一个 SINT 变量，则显示为 127（十六进制中显示 16#107f）。

IL 例子:

LD 2
INT_TO_REAL
MUL
FBD 例子:



REAL_TO-/ LREAL_TO 转换

将 REAL 或 LREAL 变量类型变为另一种类型:

数值会被四舍五入到最近的总数，然后变换成一个新的可变类型。其中例外有如下可变类型：STRING, BOOL, REAL 和 LREAL。

Please regard at a conversion to type STRING that the total number of digits is limited to 16. If the (L)REAL-number has more digits, then the sixteenth will be rounded. If the length of the STRING is defined to short, it will be cut beginning from the right end.

类型变换从大到小，会出现数据丢失的状况。

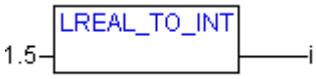
ST 例子:

```
i := REAL_TO_INT(1.5); (* 结果是 2 *)
j := REAL_TO_INT(1.4); (* 结果是 1 *)
i := REAL_TO_INT(-1.5); (* 结果是 -2 *)
j := REAL_TO_INT(-1.4); (* 结果是 -1 *)
```

IL 例子:

```
LD 2.7
REAL_TO_INT
GE %MW8
```

FBD 例子:



TIME_TO/TIME_OF_DAY 转换

把可变类型 TIME 或 TIME_OF_DAY 变为其它类型:

时间会被以毫秒为单位存储在 DWORD 内部 (如对于变量 TIME_OF_DAY 而言，开始的时间为 12:00 A.M.). 这个值将被转变。

类型变换从大到小，会出现数据丢失的状况。

而对于 STRING 类型变量而言，其结果则是一个时间常数。

IL 例子:

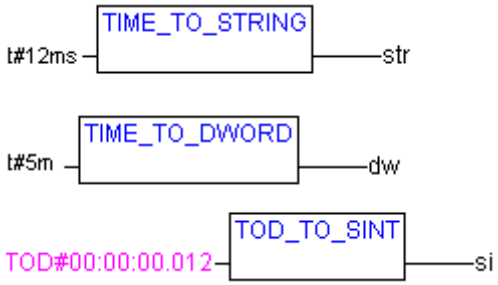
```
LD T#12ms                   (*结果是 'T#12ms' *)
 TIME_TO_STRING
ST str
LD T#300000ms               (*结果是 300000 *)
TIME_TO_DWORD
ST dw
LD TOD#00:00:00.012         (*结果是 12 *)
TOD_TO_SINT
```

```
    ST si
```

ST 例子:

```
    str :=TIME_TO_STRING(T#12ms);          (* 结果是 T#12ms *)
    dw:=TIME_TO_DWORD(T#5m);               (* 结果是 300000 *)
    si:=TOD_TO_SINT(TOD#00:00:00.012);     (* 结果是 12 *)
```

FBD 例子:



DATE_TO/DT_TO 转换

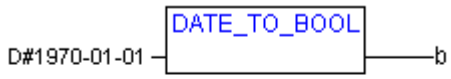
变量类型 DATE 或 DATE_AND_TIME 变为另一种类型:
1970 年 1 月起，日期将会被以秒为单位存储在 DWORD 内部。 这个值可以用来转换。
类型变换从大到小，会出现数据丢失的状况。
而对于 STRING 类型变量而言，其结果则是一个时间常数。
IL 类型:

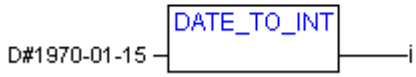
```
    LD D#1970-01-01            (* 结果是 FALSE *)
    DATE_TO_BOOL
    ST b
    LD D#1970-01-15            (* 结果是 29952 *)
    DATE_TO_INT
    ST i
    LD DT#1970-01-15-05:05:05 (* 结果是 129 *)
    DT_TO_BYTE
    ST byt
    LD DT#1998-02-13-14:20     (* 结果是 'DT#1998-02-13-14:20' *)
    DT_TO STRING
    ST str
```

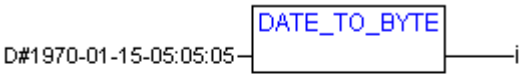
ST 类型:

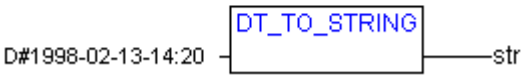
```
    b :=DATE_TO_BOOL(D#1970-01-01);            (* 结果是 FALSE *)
    i :=DATE_TO_INT(D#1970-01-15);             (* 结果是 29952 *)
    byt :=DT_TO_BYTE(DT#1970-01-15-05:05:05); (* 结果是 129 *)
    str:=DT_TO_STRING(DT#1998-02-13-14:20);    (* 结果是
                                               'DT#1998-02-13-14:20'
                                               *)
```

FBD 类型:









STRING_TO 转换

将变量类型 STRING 变为另一种类型:

STRING 类型变量的操作数必须含有一个在目标变量类型中有效的值，否则其结果为 0。

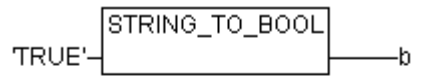
IL 例子:

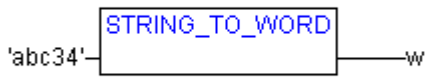
```
LD 'TRUE'              (* 结果是 TRUE *)
STRING_TO_BOOL
ST b
LD 'abc34'             (* 结果是 0 *)
STRING_TO_WORD
ST w
LD 't#127ms'           (* 结果是 T#127ms *)
STRING_TO_TIME
ST t
```

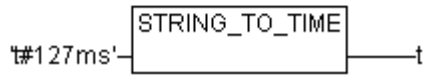
ST 例子:

```
b :=STRING_TO_BOOL('TRUE');       (* 结果是 TRUE *)
w :=STRING_TO_WORD('abc34');      (* 结果是 0 *)
t :=STRING_TO_TIME('T#127ms');    (* 结果是 T#127ms *)
```

FBD 例子:







TRUNC

CoDeSys IEC 操作符: REAL 变为 INT. 整个值的数字部分将会被用到。

类型变换从大到小，会出现数据丢失的状况。

IL 例子:

```
LD 2.7
TRUNC
```

GE %MW8
ST 例子:
i:=TRUNC(1.9); (* 结果是 1 *)
i:=TRUNC(-1.4); (* 结果是 -1 *).

10.9 数字操作符

ABS

CoDeSys IEC 操作符: 返回一个数字的绝对值。 ABS(-2) 等于 2。

可以将下列的输出输入变量进行类型合并:

IN	OUT
INT	INT, REAL, WORD, DWORD, DINT
REAL	REAL
BYTE	INT, REAL, BYTE, WORD, DWORD, DINT
WORD	INT, REAL, WORD, DWORD, DINT
DWORD	REAL, DWORD, DINT
SINT	REAL
USINT	REAL
UINT	INT, REAL, WORD, DWORD, DINT, UDINT, UINT
DINT	REAL, DWORD, DINT
UDINT	REAL, DWORD, DINT, UDINT

IL 例子:

```
LD  -2
ABS
ST  i  (* 结果是 2 *)
```

ST 例子:

```
i:=ABS(-2);
```

FBD 例子:

SQRT

CoDeSys IEC 操作符: 返回一个数字的平方根。

IN 可以为类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT. OUT 必须是类型 REAL.

IL 例子:

```
LD  16
SQRT
ST  q (* 结果是 4 *)
```

ST 例子:

```
q:=SQRT(16);
```

FBD 例子:

LN

CoDeSys IEC 操作符：返回一个数字的自燃对数。

IN 可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT. OUT 必须是类型 REAL.

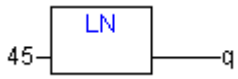
IL 例子：

```
LD  45
LN
ST  q   (* 结果是 3.80666 *)
```

ST 例子：

```
q:=LN(45);
```

FBD 例子：



LOG

CoDeSys IEC 操作符：返回一个根为 10 的数字的对数。

IN 可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT 必须是类型 REAL.

IL 例子：

```
LD  314.5
LOG
ST  q   (* 结果是 2.49762 *)
```

ST 例子：

```
q:=LOG(314.5);
```

FBD 例子：

EXP

CoDeSys IEC 操作符：指数功能的还原。

IN 可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT 必须是类型 REAL.

IL 例子：

```
LD  2
EXP
ST  q (* 结果是 7.389056099 *)
```

ST 例子：

```
q:=EXP(2);
```

FBD 例子：

SIN

CoDeSys IEC 操作符: 返回一个数字的正弦值。

输入值 IN 以拱分为单位进行计算。它可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT. OUT 必须是类型 REAL.

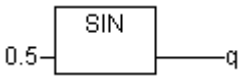
IL 例子:

```
LD  0.5
SIN
ST  q  (* 结果是 0.479426 *)
```

ST 例子:

```
q:=SIN(0.5);
```

FBD 例子:



TAN

CoDeSys IEC 操作符: 返回一个数字的正切值。

输入值 IN 以拱分为单位进行计算。它可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT. OUT 必须是类型 REAL.

IL 例子:

```
LD  0.5
TAN
ST  q  (* 结果是 0.546302 *)
```

ST 例子:

```
q:=TAN(0.5);
```

FBD 例子:

ASIN

CoDeSys IEC 操作符: 返回数字的反正弦值。（正弦的反运算）

IN 可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT 必须是类型 REAL.

IL 例子:

```
LD  0.5
ASIN
ST  q  (* 结果是 0.523599 *)
```

ST 例子:

```
q:=ASIN(0.5);
```

FBD 例子:

ACOS

CoDeSys IEC 操作符: 返回一个数字的反余弦值。（余弦的反运算）

值以拱分为单位进行计算。

IN 可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT 必须是类型 REAL.

IL 例子:

LD 0.5
ACOS
ST q (* 结果是 1.0472 *)
ST 例子:
q:=ACOS(0.5);
FBD 例子:

ATAN

CoDeSys IEC 操作符: 返回一个数字的反正切。 (正切的反运算)

IN can be type 可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT. OUT 结果是以拱分为单位进行计算而且必须为类型 REAL.

EIL 例子:
LD 0.5
ATAN
ST q (* 结果是 0.463648 *)
ST 例子:
q:=ATAN(0.5);
FBD 例子:

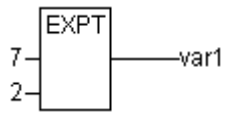
EXPT

CoDeSys IEC 操作符: 两个变量幂:

OUT = IN1IN2.

IN1 和 IN2 可以是类型 BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, OUT 必须是类型 REAL.

IL 例子:
LD 7
EXPT 2
ST var1 (* 结果是 49 *)
ST 例子:
var1 := EXPT(7,2);
FBD 例子:



10.10 初始化操作符

INI 操作符

INI 操作符可以用于初始化由 POU 功能程序提供的保留变量。

此操作符必须分配给一个布尔变量。

语法: <bool-Variable> := INI(<FB-instance, TRUE|FALSE)

如果第二个参数设定为 TRUE, 所有 FB 功能块程序中的保留变量将被初始化。

ST 例子:
fbinst 是功能块 fb 的一个实例, 在此一个保留变量 retvar 被定义。
在 POU 中的声明:

```
fbinst:fb;
b:bool;
```

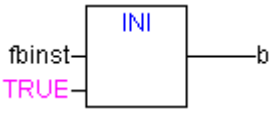
执行部分:

```
b := INI(fbinst, TRUE);
ivar:=fbinst.retvar  (* => retvar 被初始化 *)
```

IL 中调用例子:

```
LD fbinst
INI TRUE
ST b
```

FUP 中调用例子:



附录B：CoDeSys中的操作数

在codesys常量中，变量，地址，可能还有 功能调用都显示为操作数。

10.11 常量

BOOL 常量

布尔常量指逻辑值 TRUE 和 FALSE.

TIME 常量

TIME 常量可在 CoDeSys 中显示。他们一般被用于运行标准库中的定时器。
一个 TIME 常量总是由一个“t″或“T”《或“时间”的大小写格式〉和一个“#”符号组成。
随即显示时间，包括日（d 表示）时（h 表示）分（m 表示）秒（s 表示）和毫秒（ms 表示）。请注意实践相必须根据度按此顺序依次列出《d-h-m-s-ms》。
但是不要求将全部时间项一一列出。
例如：ST 分配中正确的 TIME 常量的表达方式：

```
TIME1 := T#14ms;
TIME1 := T#100S12ms;  (*最高成分允许超过其界限*)
TIME1 := t#12h34m15s;
```

下面的是错误的:

```
TIME1 := t#5m68s; (*在较低的成分中界限被超过的情况*)
TIME1 := 15ms;   (*T#丢失*)
TIME1          := (*时间顺序错误*)
t#4ms13d;
```

DATE 常量

此常量可用于键入日期。日期常量开头显示为"d","D","DATE"或"date"后加“#”符号。随后便可以以年-月-日的格式键入日期。

例如:

DATE#1996-05-06

d#1972-03-29

参照:

时间数据类型

(再见 10，15 张数据类型，时间数据类型。)

TIME_OF_DAY 常量

使用常量的这个类型存储一天的时间。开头显示为"tod#","TOD#","TIME_OF_DAY#"或"time_of_day#".

接下来是时间格式时-分-秒。可以秒的位置。

可以键入真正的数字，或键入一秒的小数部分。

例如:

TIME_OF_DAY#15:36:30.123

tod#00:00:00

参照:

时间数据类型

(再见 10，15 张数据类型，时间数据类型。)

DATE_AND_TIME 常量

日期常量和时间也可以合并以形成 DATE_AND_TIME 常量。

日期和时间常量以"dt#","DT#",DATE_AND_TIME#或 date_and_time#开始。日期与时间之间要有一个连字符。

例如:

DATE_AND_TIME#1996-05-06-15:36:30

dt#1972-03-29-00:00:00

参照:

时间数据类型

(再见 10，15 张数据类型，时间数据类型。)

数字常量

数值将以二进位，八进制，十进制和十六进制出现。

若基数不为十进制，就需基数后的常量前加写“#”，

十六进制中 10-15 的值总是用字母 A--F 来表示。

例如:

14 (十进制数)

2#1001_0011 (二进制数)

8#67 (八进制数)

16#A (十六进制数)

此数值可以是如下的变量类型如 B-/TE,

不允许将“大”的变量类型变为“小”的变量类型。

REAL/LREAL 常量

REAL 和 LREAL 常量可以以十进制小数的形式出现并用成方的形式代替。用美国标准格式中的十进制的点来表示。

例如:

7.4 代替 7,4

1.64e+009 代替 1,64e+009

STRING 常量

一个字串是一串字；字串常量前后都加单引号。也可以键入空格和特殊字（如 umlants).他们也会像其它字一样进行相应处理。

在资的顺序上,美元符号(“￥”后跟两个十六进制的数字被读为十六进制中的 8 个字节码。再者,“￥”符后两字合并被读为如下含义：

$$　　美元符

$'　　单引号

$L or $l 移行

$N or $n 换行

$P or $p 移页

$R or $r 断行

$T or $t 制表

例如:

'w1W　?'

' Abby and Craig '

':-)'

打印文字

基本上，在使用 IEC 常熟市会用到最小的数据类型。如果一定要用另一数据类型，无需明示用常量，既可以借助排版字获得。为此常量则由一个定义类型的前缀提供。

写成: <Type>#<Literal>

<Type> 标明的需要的数据类型；可用的有： BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. 类型必须上标。

<Literal> 标明的是常量。键入的数据必须与<type>下的数据类型相契合。

例如:

var1:=DINT#34;

如果常量不能在数据不丢失的情况下变为目标类型，便会生成一个错误信息。

10.12 变量

变量

在 POU 中可以在局部声明一个变量,也可以在全局列表中声明变量.

注意: 局部和全局定义的变量不同名，可直接使用本地名。不能重名,若出现同名便会出现编辑错误。变量标示符可不含空格或特殊字,只能声明一次,不能与任何关键字相同。不识别大小写。即 VAR1,Var1,和 var1 相同。

下划线字在提示符中可以识别（例：A_BCD 和 AB_CD 为两个不同标示符。）一行中可由一个下划线字。标示符的长度和其有意义部分却是无限的。

只要在公式中允许的类型，现有变量可以随处可用。

可用“输入助手”获取可用的现有变量。

参照:

系统标记

获取变量数组，结构和POUs.

变量中的地址符

系统标记

系统标记案中宣告的变量在每个 PLC 上都不同。为了找出系统中的系统标记，请用命令"插入","Oporand","输入助手"对话框弹出，选择系统变量项。

获取变量数组，结构和POUS

两位数组可以用下列句法获取。:

<Fieldname>[Index1, Index2]

结构变量可用下列句法获取:

<Structurename>.<Variablenname>

功能模块程序和程序变量可用下列句法获得:

<Functionblockname>.<Variablename>

变量中的地址符

在原数变量中，各字节皆可获取。为此，地址索引符附在变量上，中间用圆点分开。自付索引可由任何常数提供。标定指数以 0 为基数。

注意: 直接变量中的位获取不允许使用。

例如:

```
a : INT;
b : BOOL;
...
a.2 := b;
```

a 变量的第三位将设定为变量 b 的值。

如指标大于变量的位宽，将出现下列错误信息。指标"h"如超出变量的有效范围。

定位址可用如下变量类型：SINT,...DWORD.

若变量类型不允许，如下错误信息发现："无效数据类型 type"以标定指数。

一次位取得不一定都定向为一个 VAR_IN_OUT 变量。

通过公用数据进行位取:

如果已宣告一个供用常量，则定义其位标示，便可使用此常量进行位取。

注意: 任务选项"重置常数"（列表）必须激活。

见下列例子中变量上的位取得情况何一个结构变量的位获取情况：

两例中公共变量以公示:

V 变量使位获取能够得以定义:

```
VAR_GLOBAL CONSTANT
enable:int:=2;
END_VAR
```

例如 1, 态数变量上的位获取:

POU 中的描述:

```
VAR
xxx:int;
END_VAR
```

位获取:

```
xxx.enable:=true;    -> 变量 XXX 中第三，第二位将设为 TRUE。
```

例如 2, 态数结构部分的位获取:

结构 strul 的描述:

```
TYPE strul :
```

```
STRUCT
bvar:BOOL;
rvar:REAL;
wvar:WORD;
{bitaccess enable 42 'Start drive'}
END_STRUCT
END_TYPE
```

POU 中的描述:

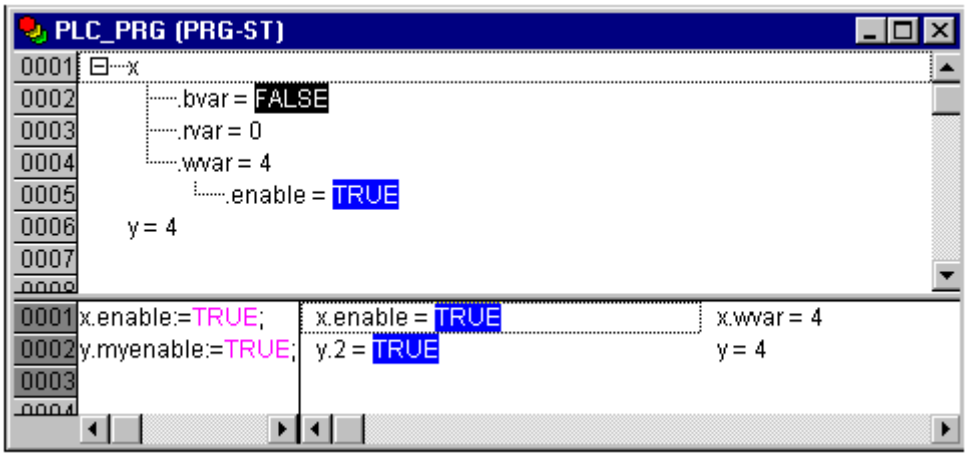
```
VAR
x:strul;
END_VAR
```

位获取:

```
x.enable:=true;
```

这将把变量 X 中的 TRUE 设定为 42 位。由于 bvar 有 8 位，war 有 32 位。为获取将在变量 wvar 的第一位上完成，其结果值是 4。

注意：如借助公共常数在结构变量完成为获得过程，一个变量应该争取地在输入助手里显示出来。在描述窗口的监视和“智能感应功能”处，请使用例子中的 **pragma {bitaccess}** 位获取。然后在窗口进行监视期间，各结构变量以外共用常数都会被显示出来：



10.13 地址

地址，注意

注意：在线修改可能改变地址内容。在 地址上应用 指针时请注意这点。

地址

通过特殊字的排列直接显示各个存储器的位置。这些排列是 "%"号,一个范围前缀，一大小前缀和一个或多个由空格分开的自然数的串联。

支持下列范围前缀:

I	输入
Q	输出
M	存储器位置

支持如下大小前缀:

X 单位

None 单位

B 字节 (8 位)

W 字(16 位)

D 双字 (32 位)

例如:

%QX7.5 和 %Q7.5 输出位 7.5

%IW215 输入字 215

%QB7 输出字节 7

%MD48 存储器中第 48 处存储位置的双字

%IW2.5.7.1 依据 PLC 配置

ivar AT %IW0 ：例如一个变量的声明包含地址的分配。

WORD;

程序中现有 PLC 配置决定地址是否有效。

注意:如没有说明不使用单位地址，布尔值便可用字节来分配，如，改变 varbool1 AT %QW0 会影响到 QX0.0－QX0.7 的值。

注意：在线修改可能改变地址内容。 在地址上应用 指针时请注意这点。

参照:

地址操作符

参见附录 A 章：IEC 符及附加的功能、地址符规范。

存储器地址

可用任何可支持的大小码来连通存储器。

例：地址%MD48 会处理存储器中(48*4=192)字节数 192，193，194 和 195。连通字，字节，甚至位都可用同样方式：地址%MX50 可以连通第 5 个字的第一位(位一般以字的方式保存)。

注意：在线修改可能改变地址内容。 在地址上应用 指针时请注意这点。

参照:

地址操作符

参见附录 A 章：IEC 符及附加的功能、地址符规范。

10.14 功能

功能

在 ST 中，功能调用也以操作数的方式出现。

例如:

```
Result := Fct(7) + 3;
```

TIME()-功能

TIME()功能返回自系统启动后所经历的时间(以毫秒为单位)。

数据类型为 TIME 型。

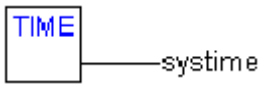
IL 例子:

```
TIME
ST systime  (* 结果为: T#35m11s342ms *)
```

ST 例子:

systime:=TIME();

FUP 例子:



附录C：CoDeSys中的数据类型

10.15 标准数据类型

数据类型

用户编程时可以使用标准和自定义数据类型。每个标识符与一种数据类型匹配。数据类型决定了存储内存空间的大小以及它所存储的值的类型。

布尔变量(BOOL)

布尔类型变量的取值是 TRUE(真) 和 FALSE(假). 它保留 8 位的存储空间.

参照: BOOL 常量

参看章节 10，11, CoDeSys 中的操作数 , BOOL 常量

整数数据类型

BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, 和 UDINT 都是整型数据类型。每个不同的数据类型包含一系列不同的值。下表列出了各整型数据类型的范围界限：

类型	下限	上限	存储器空间
BYTE	0	255	8 位
WORD	0	65535	16 位
DWORD	0	4294967295	32 位
SINT:	-128	127	8 位
USINT:	0	255	8 位
INT:	-32768	32767	16 位
UINT:	0	65535	16 位
DINT:	-2147483648	2147483647	32 位
UDINT:	0	4294967295	32 位

当大的数据类型转换成小的数据类型时，有可能导致信息丢失。

参照: 数字常数

也可参看 10，11 章，CoDeSyS 中的操作数。

REAL / LREAL

REAL 和 LREAL 就是所谓的浮点型类型。他们指代有理数。其中 REAL 占 32 位存储空间，LREAL 则占 64。

REAL 存储范围：1.175494351e-38F ～ 3.402823466e+38F

LREAL 存储范围：2.2250738585072014e-308 ～ 1.7976931348623158e+308

也可参见 10.11 章，REAL-/LREAL 常数

参照:REAL-/LREAL常数

字符串

字符串类型变量可包含任何一串字符。声明时变量的大小就决定为变量保留多大的存储空间。此涉及字符串的字符数并可放入圆括号和方括号内。如果未给出变量的大小规格，默认大小为 80 个字符。

基本上，在 CoDeSys 中，字符串的长度没有限制，但是字符串的功能仅可以处理 1-255 个字符！

35 个字符串声明的例子：

str:STRING(35):='This is a String';

参照也可参见 10.11 章，CoDeSys 中的操作数，

时间 日期类型

数据类型 TIME,TIME_OF_DAY(缩写 TOD),DATE and DATE_AND_TIME(缩写 DT)像 DWORD 一样由内部进行处理。

TIME 和 TOD 中时间单位设定为毫秒，在 TOD 中的时间从上午 12 点开始。

在 DATE 和 DT 中，时间单位设定为秒，起始日期为 1970 年 1 月 1 日上午 12 点。

见如下时间数据格式，它们被用于为时间常数分配值：

TIME 常数:

总是由首字母"t"或"T"("time"或"TIME")和一个数字符"#"组成。随之便是包含日(显示为"d")，时(显示为"h")，分(显示为"m")，秒(显示为"s")，和毫秒(显示为"ms")的确切的时间声明。请注意，时间项必须按时间顺序排列(d 在 h 前，h 在 m 前，m 在 s 前，s 在 ms 前)，但无须包含所有时间部分。最大值：49 天 17 时 2 分 47 秒 295 毫秒(4194967295 毫秒)。

ST 分配中正确的时间常数的例子:

```
TIME1 := T#14ms;
TIME1          := (*最高部分可以超过限制*)
T#100S12ms;
TIME1          :=
t#12h34m15s;
```

下面则是错误的:

```
TIME1 := t#5m68s; (*较低部分超过界限*)
TIME1 := 15ms;    (*T#数据丢失*)
TIME1 := t#4ms13d;(*项目顺序的错误*)
```

DATE 常数:

一个日期常数以"d"，"D"，"DATE"或"date"开始，后接"#"号。然后便可按年月日的格式键入日期。可能出现的值：1970-00-00 至 2106-02-06。

例如:

DATE#1996-05-06

d#1972-03-29

TIME_OF_DAY 常数，用来存储一天的时间：

以"tod#","TOD#","TIME_OF_DAY#"或"time_of_day#"开始，接着是以时分秒格式出现的时间。

秒可作为真正的数字键入，或以秒的分数形式键入。可能出现的值为：00：00：00～23：59：59.999。

例如:

TIME_OF_DAY#15:36:30.123

tod#00:00:00

DATE_AND_TIME 常数，日期与时间的合并：

以"dt#","DT#","DATE_AND_TIME"或"date_and_time"起始.日期与时间之间用连字符连接。可能出现的

值：1970-00-00-00:00 至 2016-02-06-06:28:15
例如:
DATE_AND_TIME#1996-05-06-15:36:30
dt#1972-03-29-00:00:00

10.16 已定义的数据类型

数组

1，2，3 维符号组(数组)为所支持的基本数据类型。数组在 POU 的声明部分和全局变量列表中都可加以定

义。最多的 9 维可用数组嵌套而成。

语法:

<Field_Name>:ARRAY [<ll1>..<ul1>,<ll2>..<ul2>] OF <elem. 类型>.

ll1,ll2,ll3 可识别符号组的下限，ul1,ul2,ul3 识别上限。限值须为整数，还必须在 DINT 值域内。

例如:

```
Card_game: ARRAY [1..13, 1..4] OF INT;
```

初始化数组:

数组完整初始化例子:

```
arr1 : ARRAY [1..5] OF INT := 1,2,3,4,5;
arr2 : ARRAY [1..2,3..4] OF INT := 1,3(7); (*  1,7,7,7 的缩写 *)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := 2(0),4(4),2,3;
       (*  0,0,4,4,4,4,2,3  的缩写*)
```

结构中数组初始化例子:

```
TYPE STRUCT1
STRUCT
p1:int;
p2:int;
p3:dword;
END_STRUCT
ARRAY[1..3] OF STRUCT1:= (p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299),
(p1:=14,p2:=5,p3:=112);
```

数组部分初始化例子:

```
arr1 : ARRAY [1..10] OF INT := 1,2;
```

无数值配置的元素，初始化数值默认为基本类型的初始值。因此上述例子中，数组[6]到数组[10]则初始化为 0。

访问数组元素:

数组元素用 2 维数组访问，语法如下:

<Field_Name>[Index1,Index2]

例如:

```
Card_game [9,2]
```

注意: 若工程中使用 CheckBounds定义功能，可以核对工程中域溢出情况。(见 2.1 章，CoDeSys的祥述，以及"工程元素"和"功能"的说明)

CheckBounds 功能

若在工程中使用CheckBounds来定义功能，便可自动检查 数组中域溢出错误。功能的名称被修复，并且

只能有这一个名称。

CheckBounds 功能的例子:

```
FUNCTION CheckBounds : DINT
VAR_INPUT
 index, lower, upper: DINT;
END_VAR
    IF  index < lower THEN
 CheckBounds := lower;
ELSIF  index > upper THEN
 CheckBounds := upper;
ELSE  CheckBounds := index;
END_IF
```

如下检测 CheckBounds 功能的示范程序超出了一个定义了的数组的范围。CheckBounds 功能可使 TRUE 值加以重定向，不是定向到 A[10]区，而是定向到上述仍有效的域边界 A[7]区。使用 CheckBounds 功能，数组边界以外的参照符能加以纠错。

CheckBounds 功能的测试程序:

```
PROGRAM PLC_PRG
VAR
 a: ARRAY[0..7] OF BOOL;
 b: INT:=10;
END_VAR
    a[b]:=TRUE;
```

注意: 由 Check.Lib 库提供的 CheckBounds 功能只是一个样板方案。在使用库包前，请检查此功能是否按要求运行，否则要直接作为方案中的一个 POU 来实现一个相应的功能。

指针

变量或功能块 地址在程序运行时被存储在指针中。

声明一个指针的语法如下:

<标识符>: POINTER TO <数据类型/功能块>;

指针可指向任何数据类型或功能块，甚至可用于自定义类型。地址操作符 ADR 的功能是分配变量或功能块的地址到指针。

在指针标志符后通过添加一个内容操作符"^"可废除指针。

注意: 指针以字节为单位计算！可以通过使用说明 p＝p+SIZEOF(p^)像在 C_Compiler 一样进行计算。

例如:

```
pt:POINTER TO INT;
var_int1:INT := 5;
var_int2:INT;
pt := ADR(var_int1);
var_int2:= pt^; (* var_int2 is now 5 *)
```

运行期间检查指针访问的功能

为了检测指针访问情况，可以生成检测功能，在每次访问前自动调用。为此，每项功能都必须直接或通过功能库放入方案中。支持如下功能:

检查指针功能: 检测目前存储在指针处的地址是否在有效的存储器域内。

检查指针排列功能: 由存储器校正检测扩展的检查指针的功能性。

这些功能必须用上述名称。它们可以还原被废止的指针的地址，这样最多也被接受为第一次输入的参数。

(如下例中的 dwAddress)。

见下例中的检查指针排列功能，其中的输入参数经检测功能处理。(参数的名称为例子)。检查指针功能必须外观相同，除了指针访问时，没有间隔尺寸参数：

```
FUNCTION CheckPointerAligned : DWORD    (* 功能数据类型(还原值)必须和在当前设定的目
                                           标系统中所使用的指针相同；例：使用 32 位指针
                                           的系统类型 DWORD，16 位指针系统类型 WORD *)
VAR_INPUT
 dwAddress : DWORD;                     (* 指针的目标地址；数据类型必须与当前系统中
                                           设定使用的指针相同，见上：还原值 *)
 iSize : DINT;                          (* 指针访问的尺寸，数据类型必须与整数兼容，
                                           并且须包含在所存的指针地址中最大数据类型的
                                           尺寸范围内*)
 iGran : DINT;                          (* 不在检查指针功能中应用:
                                           访问粒度，如”2”，如果 INT 在已知地址中被用
                                           作最小的无结构数据类型，那数据类型必须与整数
                                           兼容 *)
 bWrite: BOOL;                          (*访问类型: 读或写; TRUE=读方式访问; 数据类
                                           型必须为布尔型*)
END_VAR
```

如果方案中有一个检查指针功能和一个检查指针排列功能，便会调用检查指针排列功能。

枚举

列举为自定义数据类型，由字符串参数的数字组成。这些常数被作为列举值。

列举值在所有的计划中都可被识别，即使它们在 POU 中已被声明。 最好在注册卡 数据类型下的对象管理器中把列举功能生成为目标。它们以 TYPE 为关键字开始，以 END_TYPE 结束。

语法:

```
TYPE <标识符>:(<Enum_0> ,<Enum_1>, ...,<Enum_n>);
END_TYPE
```

TYPE<Identifier>变量可以接纳一个列举值，并将用第一个将其初始化。这些值与所有数字兼容，即你可以像操作 INT 一样执行此类操作。可以把一个数字 X 配置到变量中。如果列举值未被初始化，计数则会从 0 开始。在进行初始化时，要确认原始值不断增大。在运行过程中数字的有效性会被检查。

例如:

```
TYPE TRAFFIC_SIGNAL: (Red, Yellow, Green:=10); (*对每个颜色的初始化值分别为 红色是 0, 黄色
是 1, 绿色是 10 *)
END_TYPE
TRAFFIC_SIGNAL1 : TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0; (* 交通灯信号值为红*)
FOR i:= Red TO Green DO
 i := i + 1;
END_FOR;
```

同一 POU 中所使用的一个或所有的列举中，同一个列举值不能重复使用两次。

例如:

TRAFFIC_SIGNAL: (red, yellow, green);
COLOR: (blue, white, red);
Error: red may not be used for both TRAFFIC_SIGNAL and COLOR.

结构

在注册中，数据类型项下的对象管理器中，结构被生成为目标。它们以 TYPE 和 STRUCT 开始，以 END_TYPE 和 END_STRUCT 结束。

结构声明语法如下：

```
TYPE <结构体名称>:
STRUCT
 <变量 1 的声明>
 .
 .
 <变量 n 的声明>
END_STRUCT
END_TYPE
```

<结构体名称>为一种在整个工程中都可被识别并像标准数据类型使用的类型。

允许联锁结构。唯一的限制即变量不可放在地址中。(不允许 AT 声明！)

名为 Polygonline 的结构定义例子:

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

结构初始化例子：

Poly_1:polygonline := (Start:=3,3, Point1 =5,2, Point2:=7,3, Point3:=8,5, Point4:=5,7, End := 3,5);

不可能出现变量是初始化。见 Arrays 项下的结构数组初始化例子。

结构元素的访问:

用如下语法可访问结构的元素。:

<Structure_Name>.<成分名称>

如上述 polyonline 示例中，可通过 Poly_1.start 来访问此结构的元素。

参照符

可用自定义参照符给变量，常数或功能块生成可选的名称。

在注册卡中，数据类型项下的 ObjectOrganizer 中把参照符生成为目标。它们以关键 TYPE 开始，以 END_TYPE 结束。

语法:

```
TYPE <Identifier>: <Assignment term>;
END_TYPE
```

例如:

```
TYPE message:STRING[50];
END_TYPE;
```

附属域类型

所谓附属域即是一个其数值域为基本类型之子集的类型。尽管其声明可中数据类型注册中进行，但是变量还可以字节用一个附属域类型直接声明。

‘数据类型’注册中声明用的语法：

TYPE <名称> : <Inttype> (<ug>..<og>) END_TYPE;

<名称> 必须是个有效的 IEC 标识符。

<Inttype> 是下面数据类型当中的一个 SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD).

<ug> 是常数，必须与基本类型兼容。它还为域类型设置下边界。下端边界本身包含在本域内。

<og> 为常数，必须与基本类型兼容。它还为域类型设置上边界，此上边界本身包含在这个基本类型中。

例如:

```
TYPE
 SubInt : INT (-4095..4095);
END_TYPE
```

用附属域类型直接声明变量:

```
VAR
 i : INT (-4095..4095);
 ui : UINT (0..10000);
END_VAR
```

如果一个常数被分配到一个附属域类型中(在声明中或在执行中)，此时变能产生一个错误信息。

为了检查运行时域边界有效执行，必须引用检查行符号或检查列符号功能。由此，边界失效会被用恰当的方法和手段捕捉到(例，值可减少或设一个错误标记)。一旦一个变量被写为由一个有符号或无符号类型构成的附属域。它们便被自动调用。

例如:

如果一个变量属于一个有符号的附属域类型（如 i，见上)，检查行符号功能就被调用；为了把一个值修正为容许范围，程序例子如下：

```
FUNCTION CheckRangeSigned : DINT
VAR_INPUT
 value, lower, upper: DINT;
END_VAR
IF (value < lower) THEN
 CheckRangeSigned := lower;
ELSIF(value > upper) THEN
 CheckRangeSigned := upper;
ELSE
 CheckRangeSigned := value;
END_IF
```

在自动调用功能时，功能名称检查列符号是固定的，它是界面的规范：DINT 类型的返回值和三个参数。调用时，此功能参数如下：

- 值： 分配到域类型的值
- 下限： 域的下边界
- 上限： 域的上边界
- 返回值：实际分配到域类型的值

下例中一次分配 i:=10*y 自动生成如下结果：

```
i := CheckRangeSigned(10*y, -4095, 4095);
```

即使例子中 y 值为 1000，分配后，i 值仍只有 4095。

同样适用于检查行无符号功能：功能名称和界面必须正确。

```
FUNCTION CheckRangeUnsigned : UDINT
VAR_INPUT
 value, lower, upper: UDINT;
END_VAR
```

重要提示：若 CheckRangeSigned 或 CheckRangeUnsigned 皆无，运行时附属域类型将不会检测任何类型，i 变量将接受任何时间介于 32768 和 32767 之间的任何值。

注意：若如上所述，无 CheckRangeSigned 或 CheckRangeUnsigned 功能，如果 FOR 循环指令中使用附属域类型，就会出现无限的循环指令。如果在 FOR 循环指令的域比附属域类型的域一样或更大时，这种情况便会发生。

注意：Check.Lib 库所提供的 CheckRangeSigned 功能只是一个解决方案的样例！在使用程序库模块之前，请检查该功能是否按工程中要求的那样正在运行，或者像工程中的一个 POU 一样直接执行一个适当的 CheckRange-功能。

例如：

```
VAR
 ui : UINT (0..10000);
END_VAR

FOR ui:=0 TO 10000 DO
...
END_FOR
```

FOR 循环指令永远不会结束，因为 ui 不可能大于 10000。在定义 FOR 循环指令的增加值时还要注意 CheckRange-功能的定义。

附录D：CoDeSys程序库

10.17 standard.lab标准库

10.17.1 字符串功能

注意：字符串功能不具有’线程安全’：当使用任务时,字符串功能只能用于一个任务中,如果同一功能用于不同任务,有被覆盖的危险.

See also:

LEN
LEFT
RIGHT
MID
CONCAT
INSERT
DELETE
REPLACE
FIND

LEN

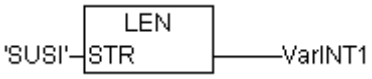
返回字符串长度。

输入 STR 便是类型 STRING，功能的返回值是 INT 类型。

IL 例子:

```
LD   'SUSI'
LEN
T   VarINT1 (* 结果是 4 *)
```

FBD 例子:



ST 例子:

```
VarSTRING1 := LEN ('SUSI');
```

注意: 字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

LEFT

Left 返回，已知字符串为初始化字符串。

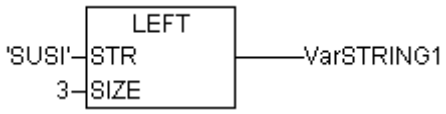
输入 STR 即类型 STRING，SIZE 为 INT 类型，此功能的返回值为 STRING 类型。

LEFT(STR,SIZE)意思是：在字符串 STR 从右边取第一个 SIZE 字节。

IL 例子:

```
LD   'SUSI'
LEFT 3
ST   VarSTRING1 (* 结果是 'SUS' *)
```

FBD 例子:



ST 例子:

```
VarSTRING1 := LEFT ('SUSI',3);
```

注意: 字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

RIGHT

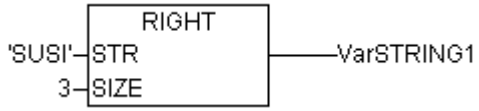
Right 返回，已知字符串为初始化字符串。

RIGHT(STR,SIZE)意思是：从字符串 STR 的右边取第一个 SIZE 字节。

IL 例子:

```
LD    'SUSI'
RIGHT 3
ST    VarSTRING1 (* 结果是 'USI' *)
```

FBD 例子:



ST 例子:

```
VarSTRING1 := RIGHT ('SUSI',3);
```

注意: 字符串功能并非"thread safe": 使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

MID

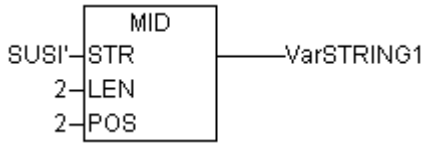
MID 从一个字符串中返回一部分字符串。

输入 STR 为类型 STRING，而 LEN 和 POS 为类型 INT，功能的返回值为类型 STRING

IL 例子:

```
LD  'SUSI'
MID 2,2
ST  VarSTRING1 (* 结果是 'US' *)
```

FBD 例子:



ST 例子:

```
VarSTRING1 := MID ('SUSI',2,2);
```

注意: 字符串功能并非"thread safe": 使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

CONCAT

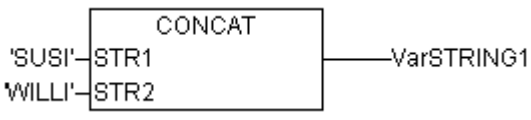
两个字符串的连接(合并)

输入变量 STR1 和 STR2 以及功能的返回值都是类型 STRING。

IL 例子:

```
LD     'SUSI'
CONCAT 'WILLI'
ST     VarSTRING1 (* 结果是 'SUSIWILLI' *)
```

FBD 例子:



ST 例子:

```
VarSTRING1 := CONCAT ('SUSI','WILLI');
```

注意: 连接 CONTACT 功能如果嵌套三层以上便失灵。

注意: 字符串功能并非"thread safe": 使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，

便有被覆盖的危险。

INSERT

INSERT 可以在定义的点把一个字符串插入到另一个字符串中。

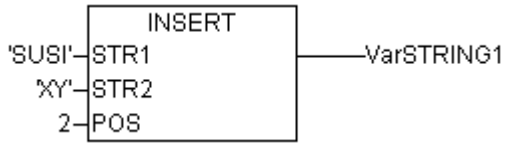
输入变量 STR1 和 STR2 为 STRING 类型，POS 是 INT 类型，功能返回值是 STRING 类型。

INSERT(STR1,STR2,POS)意思是：在 POS 之后将 STR2 插入到 STR1 中。

IL 例子:

```
LD    'SUSI'
INSERT 'XY',2
ST    VarSTRING1 (* 结果是 'SUXYSI' *)
```

FBD 例子:



ST 例子:

```
VarSTRING1 := INSERT ('SUSI','XY',2);
```

注意: 字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

DELETE

DELETE 可以在定义的位置把一个较大的字符串的一部分删除掉。

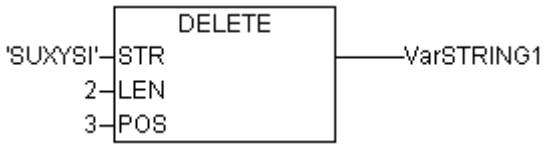
输入变量 STR 为 STRING 类型，LEN 和 POS 为 INT 类型，功能的返回值是 STRING 类型。

DELETE(STR,L,P)意思是：从以 P 位置开始的字节处将 L 字节删除。

IL 例子:

```
LD    'SUXYSI'
DELETE 2,3
ST    Var1 (* 结果是 'SUSI' *)
```

FBD 例子:



ST 例子:

```
Var1 := DELETE ('SUXYSI',2,3);
```

注意: 字符串功能并非“thread safe”：使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

REPLACE

REPLACE 可用第三个字符串从较大的字符串中替换掉部分字符串。

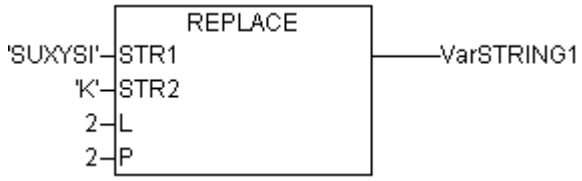
输入变量 STR1 和 STR2 为 STRING 类型，LEN 和 POS 为 INT 类型。

REPLACE(STR1,STR2,L,P)意思是：从以 P 位置开始的 STR2 把 STR1 从 L 字节处替换。

IL 例子:

```
LD      'SUXYSI'
REPLACE 'K',2,2
ST      VarSTRING1 (* 结果是 'SKYSI' *)
```

FBD 例子:



ST 例子:

```
VarSTRING1 := REPLACE ('SUXYSI','K',2,2);
```

注意: 字符串功能并非"thread safe": 使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

FIND

FIND 在一个字符串中查找到某部分字符串。

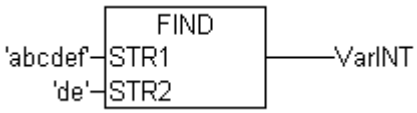
输入变量 STR1 和 STR2 都是 STRING 类型，功能的返回值是 STRING 类型。

FING(STR1,STR2)意思是：查找 STR1 第一次出现在 STR2 处时，后者第一个字节的位置。

IL 例子:

```
LD   'abcdef'
FIND 'de'
ST   VarINT1 (* 结果是 '4' *)
```

FBD 例子:



ST 例子:

```
arINT1 := FIND ('abcdef','de');
```

注意: 字符串功能并非"thread safe": 使用任务时，字符串公民只可用在单任务中。如果不同任务中使用相同的功能，便有被覆盖的危险。

10.17.2 双稳功能程序

SR

重置双稳定功能模块

Q1 = RS (SET, RESET1) 的意思是:

Q1 = NOT RESET1 AND (Q1 OR SET)

输入变量 SET 和 RESET1 以及输出变量 Q1 都是 BOOL 类型。

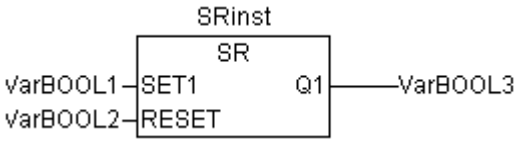
声明如下所示:

```
RSInst : RS ;
```

IL 例子:

```
CAL RSInst(SET:= VarBOOL1,RESET1:=VarBOOL2)
LD RSInst.Q1
ST VarBOOL3
```

FBD 例子:



ST 例子:

```
RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2 );
VarBOOL3 := RSInst.Q1 ;
```

RS

重置双稳定功能模块

Q1 = RS (SET, RESET1) 的意思是:

Q1 = NOT RESET1 AND (Q1 OR SET)

输入变量 SET 和 RESET1 以及输出变量 Q1 都是 BOOL 类型。

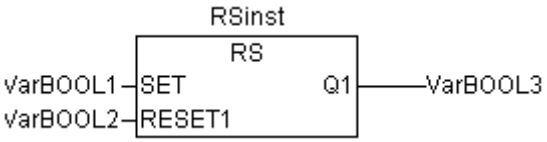
声明如下所示:

```
RSInst : RS ;
```

IL 例子:

```
CAL RSInst(SET:= VarBOOL1,RESET1:=VarBOOL2)
LD RSInst.Q1
ST VarBOOL3
```

FBD 例子:



ST 例子:

```
RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2 );
VarBOOL3 := RSInst.Q1 ;
```

SEMA

软件信号量(可断续的)

BUSY = SEMA(CLAIM, RELEASE) 的意思是:

```
BUSY := X;
IF CLAIM THEN X:=TRUE;
ELSE IF RELEASE THEN BUSY := FALSE; X:= FALSE;
END_IF
```

X 为内部布尔变量，初始化时是 FALSE。输入变量 CLAIM 和 RELEASE 以及输出变量 BUSY 都是 BOOL 变量。

如果调用 SEMA 时，BUSY 为 TURE，即 SEMA 已经被分配了值(SEMA 被调用时，CLAIM＝TRUE)。若 BUSY 是 FALSE，不调用 SEMA 或已被启动(RELEASE=TRUE)。

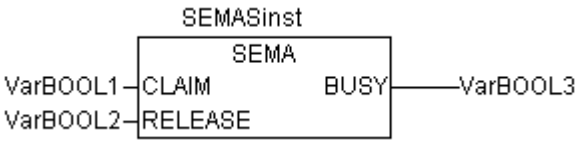
声明如下所示:

```
SEMAInst : SEMA ;
```

IL 例子:

```
CAL SEMAInst(CLAIM:=VarBOOL1,RELEASE:=VarBOOL2)
LD SEMAInst.BUSY
ST VarBOOL3
```

FBD 例子:



ST 例子:

```
SEMAInst(CLAIM:= VarBOOL1 , RELEASE:=VarBOOL2 );
VarBOOL3 := SEMAInst.BUSY;
```

10.17.3 触发器

R_TRIG

R_TRIG 功能块触发一个上升的边界。

```
FUNCTION_BLOCK R_TRIG
VAR_INPUT
 CLK : BOOL;
END_VAR
VAR_OUTPUT
 Q : BOOL;
END_VAR
VAR
 M : BOOL := FALSE;
END_VAR
 Q := CLK AND NOT M;
 M := CLK;
```

只要输入变量 CLK 为 FALSE，输出 Q 和帮助变量 M 依然是 FALSE。一旦 CLK 返回 TRUE，Q 首先返回 TRUE，那么 M 就会被设置为 TRUE。此意味着，没调用一次此功能，Q 便会返回 FALSE，直到 CLK 边界已下降，随即边界再次上升。

声明如下所示:

```
RTRIGInst : R_TRIG ;
```

IL 例子:

```
CAL RTRIGInst(CLK := VarBOOL1)
LD  RTRIGInst.Q
ST  VarBOOL2
```

FBD 例子:

ST 例子:

```
RTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := RTRIGInst.Q;
```

F_TRIG

F_TRIG 功能块触发一个下降的边界。

```
FUNCTION_BLOCK F_TRIG
VAR_INPUT
```

```
 CLK: BOOL;
END_VAR
VAR_OUTPUT
 Q: BOOL;
END_VAR
VAR
 M: BOOL := FALSE;
END_VAR
 Q := NOT CLK AND NOT M;
 M := NOT CLK;
```

只要输入变量 CLK 返回 TRUE,输出 Q 和帮助变量 M 依然是 FALSE。一旦 CLK 返回 FALSE,Q 首先返回 TRUE，那么 M 就会被设置为 TRUE。此意味着，没调用一次此功能，Q 便会返回 FALSE，直到 CLK 边界已上升，随即边界再次下降。

声明如下所示:

```
FTRIGInst : F_TRIG ;
```

IL 例子:

```
CAL FTRIGInst(CLK := VarBOOL1)
LD  FTRIGInst.Q
ST  VarBOOL2
```

FBD 例子:

ST 例子:

```
FTRIGInst(CLK:= VarBOOL1);
VarBOOL2 := FTRIGInst.Q;
```

10.17.4 记数器

CTU

功能模块增量器:

输入变量 CU 和 RESET 以及输出变量 Q 都是 BOOL 类型，输入变量 PV 和输出变量 CV 是 WORD 类型。

计数器变量 CV 在 RESET 为 TURE 时被初始化为 0，如果 CU 从 FALSE 到 TURE 为上升边界，CV 将被增加 1，Q 在 CV 大于或等于 PV 上限时，会返回 TRUE。

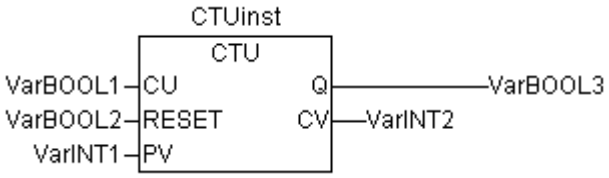
声明如下所示:

```
CTUInst : CTU ;
```

IL 例子:

```
CAL  CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV := VarINT1)
LD   CTUInst.Q
ST   VarBOOL3
LD   CTUInst.CV
ST   VarINT2
```

FBD 例子:



ST 例子:

```
CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTUInst.Q ;
VarINT2 := CTUInst.CV;
```

CTD

功能模块减数器:

输入变量 CD 和 LOAD 以及输出变量 Q 都是 BOOL 类型，输入变量 PV 和输出变量 CV 为 WORD 类型。

当 LOAD_为 TRUE 时，如果 CV 大于 0(即，它不致让值低于 0).，CV 则会下降 1。当 CVis 等于 0 时 Q 返回 TRUE。

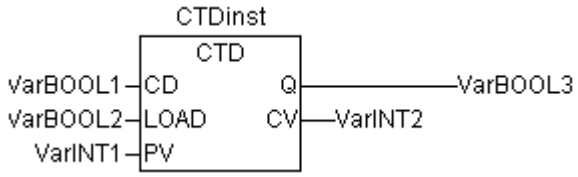
声明如下所示:

```
CTDInst : CTD ;
```

IL 例子:

```
CAL  CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV := VarINT1)
LD   CTDInst.Q
ST   VarBOOL3
LD   CTDInst.CV
ST   VarINT2
```

FBD 例子:



ST 例子:

```
CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1);
VarBOOL3 := CTDInst.Q ;
VarINT2 := CTDInst.CV;
```

CTUD

功能模块增量器/减量器

输入变量 CU，CD，RESET，LOAD 以及输出变量 QU 和 QD 都是 BOOL 类型，PV 和 CV 是 WORD 类型。

若 RESET 有效，计数变量 CV 以 0 值进行初始化。若 LOAD 有效，CV 则以 PV 进行初始化。

若 CU 从 FALSE 到 TRUE 边界下降，则 CV 上 1。如果 CD 工 FALSE 到 TRUE 为上升边界，如果这不会使数值降到 0 以下，CV 则会降 1。

CV 大于等于 PV 时，QU 返回 TRUE。CV 等于 0 时，QD 返回 TRUE。

声明如下所示:

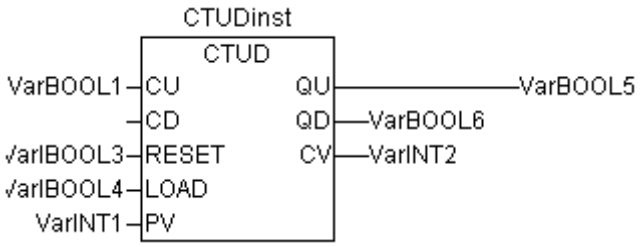
```
CTUDInst : CUTD ;
```

IL 例子:

```
CAL  CTUDInst(CU:=VarBOOL2, RESET:=VarBOOL3,
LOAD:=VarBOOL4, PV:=VarINT1)
```

```
        LD    CTUDInst.Q
        ST    VarBOOL5
        LD    CTUDInst.QD
        ST    VarBOOL5
        LD    CTUInst.CV
        ST    VarINT2
```

FBD 例子:



ST 例子:

```
        CTUDInst(CU := VarBOOL1, CU:= VarBOOL2, RESET :=
VarBOOL3, LOAD:=VarBOOL4 , PV:= VarINT1);
        VarBOOL5 := CTUDInst.QU ;
        VarBOOL6 := CTUDInst.QD ;
        VarINT2 := CTUDInst.CV;
```

10.17.5 定时器

TP

功能块定时器是一个触发器。TP（IN，PT，Q，ET）意思是:

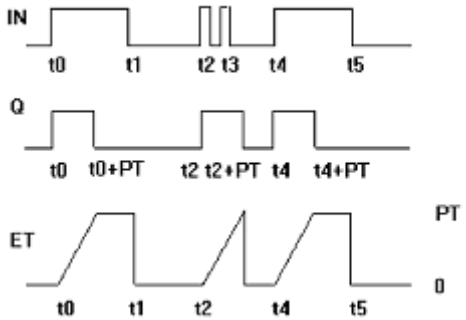
IN 和 PT 分别是类型为 BOOL 和 TIME 的输入变量。Q 和 ET 分别是类型为 BOOL 和 TIME 的输出变量。若 IN 是 FALSE，Q 是 FALSE，ET 为 0。

在 IN 变成 TRUE 时，ET 中的时间便开始以毫秒计数，直到其值等于 PT。然后便保持恒定。

当 IN 为 TRUE 并且 ET 小于或等于 PT 时，Q 为 TRUE,。否则 Q 是 FALSE。

Q 返回一个 PT 中已知的时间段信号。

定时器时间顺序的图表显示:

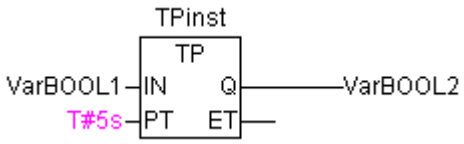


声明如下所示:

```
        TPInst : TP ;
        IL 例子:
        CAL TPInst(IN := VarBOOL1, PT := T#5s)
        LD  TPInst.Q
```

```
        ST   VarBOOL2
```

FBD 例子:



ST 例子:

```
        TPInst(IN := VarBOOL1, PT:= T#5s);
        VarBOOL2 :=TPInst.Q;
```

TON

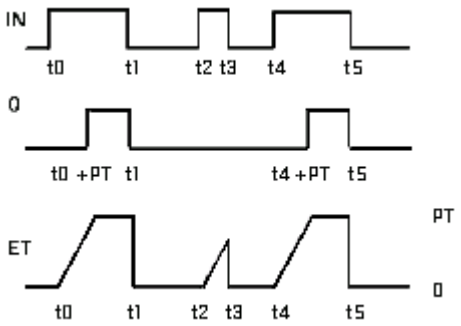
功能块 Timer On Delay 实现开启延迟。

TON(IN, PT, Q, ET)意思是：

IN 和 PT 分别是类型为 BOOL 和 TIME 的输入变量。Q 和 ET 分别是类型为 BOOL 和 TIME 的输出变量。若 IN 是 FALSE，Q 则是 FALSE，ET 则为 0。

IN 一旦变成 TRUE 时，ET 中的时间便开始以毫秒计数，直到其值等于 PT。然后便保持恒定。当 IN 为 TURE 并且 ET 等于 PT 时，Q 为 TRUE。否则 Q 为 FALSE。这样，当 PT 所示的以毫秒为单位的时间耗尽时，Q 呈现一个上升的边界。

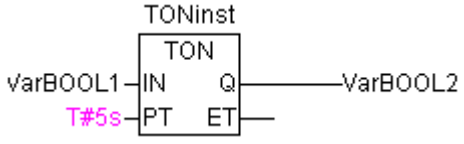
TON 时间行为图表如图：



声明如下所示:

```
        TONInst : TON ;
        IL 例子:
        CAL TONInst(IN := VarBOOL1, PT := T#5s)
        LD  TONInst.Q
        ST  VarBOOL2
```

FBD 例子:



ST 例子:

```
    TONInst(IN := VarBOOL1, PT:= T#5s);
```

TOF

功能块 TOF 实现关闭延迟。

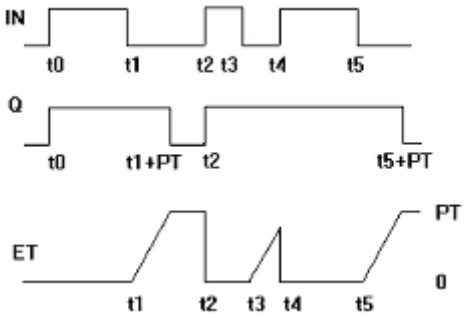
TOF(IN, PT, Q, ET)意思是：

IN 和 PT 分别是类型为 BOOL 和 TIME 的输入变量。Q 和 ET 分别是类型为 BOOL 和 TIME 的输出变量。若 IN

是 TRUE，则输出变量分别为 TRUE 和 0。

一旦 IN 变为 FALSE，ET 中时间便开始以毫秒为单位进行计时，直到值等于 PT 为止。然后便保持恒定。当 IN 为 FALSE 并且 ET 等于 PT 时，Q 为 FALSE。否则 Q 为 TRUE。这样，当 PT 以毫秒为单位所示的时间耗尽时，Q 的边界下降。

TOF 时间行为图表如图：



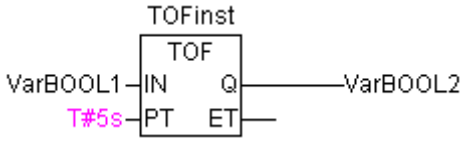
声明如下所示：

```
TOFInst : TOF ;
```

IL 例子：

```
CAL TOFInst(IN := VarBOOL1, PT := T#5s)
LD TOFInst.Q
ST VarBOOL2
```

FBD 例子：

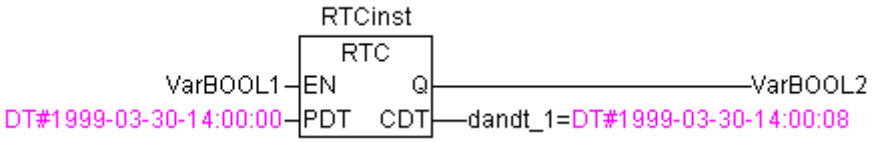


ST 例子：

```
TOFInst(IN := VarBOOL1, PT:= T#5s);
VarBOOL2 :=TOFInst.Q;
```

RTC

定时开始，功能块 Runtime Clock 返回当前时间和日期。



RTC(EN, PDT, Q, CDT)意思是：

EN 和 PDT 是类型为 TIME 输入变量，Q 和 CDT 分别是类型为 BOOL 和 DATE_AND_TIME 输出变量。若 EN 为 FALSE 时，输出变量 Q 和 CDT 则分别时 FALSE 和 DT#1970-01-01-00：00：00。一旦 EN 变为 FALSE，PDT 时间已定，并以秒为单位计时。

只要 EN 变成 TRUE，CDT 则复原(见上例图示)。每当 EN 重设成 FALSE，则 CDT 又重设为初始值 DT#1970-01-01-00：00：00。请注意，PDT 的时间只能被上升边界设定。

10.18 Util.lib库

此库包含可以用于 BCD 转换，位/字节功能，数字辅助功能等各种块的附加集，以作模拟值处理控制器，信号生成器和功能操控器。

由于其中一些功能和功能块含有 REAL 变量,还有一个名为 UTIL_NO_REAL 的附加库,其中不含有这些 POU。

10.18.1 BCD 转换

BCD 格式中一个字节含有 0-99 之间的整数。每个十进位的空间占用 4 个位。10 个十进位的空间被存在 4-7 位之间。这样，BCD 格式则与十六进制表达方式是相似的。唯一不同的是 BCD 格式中只存 0-99 的数字，而十六进制字节是从 0-FF。

示例:整数 51 应转换位 BCD 格式。5 在二进制中为 0101,1 在二进制中为 0001,这样 BCD 字节便为 01010001,此对应的值为$51=81。

参照:

BCD_TO_INT

此功能把 BCD 格式的一个字节转换成 INT 值:

此功能的输入值为 BYTE 类型，输出为 INT 类型。

字节应转换的地方，如不是 BCD 格式，其输出结果为 1。

ST 例子:

```
i:=BCD_TO_INT(73); (* 结果是 49 *)
k:=BCD_TO_INT(151); (* 结果是 97 *)
l:=BCD_TO_INT(15); (* 输出 -1, 因为它不是 BCD 格式 *)
```

INT_TO_BCD_

此功能把 INTEGER 值转换成 BCD 格式 的一个字节:

此功能的输入值为 INT 类型，输出为 BYTE 类型。

数字 255 出现在 INTEGER 应该转换，然而却不能转换为字节的地方。

ST 例子:

```
i:=INT_TO_BCD(49); (* 结果是 73 *)
k:=BCD_TO_INT(97); (* 结果是 151 *)
l:=BCD_TO_INT(100); (* 错误! 输出: 255 *)
```

10.18.2 位/字节功能

EXTRACT

这功能的输入是一个 DWORD X，以及一个 BYTE N。此输出为一个 BOOL 值，其中含有输入 X 的第 N 个位的内容，由此，此功能开始从零位计数。

ST 例子:

```
FLAG:=EXTRACT(X:=81, N:=4); (* 结果: TRUE, 因为 81 的二进制数为 1010001, 所以第四位是 1 *)
FLAG:=EXTRACT(X:=33, N:=0); (* 结果: TRUE, 因为 33 的二进制数为 100001, 所以第零位是 1 *)
```

PACK

此功能可以回送 8 个输入位 B0，B1，…，B7，从类型 BOOL 作为一个 BYTE。功能块 UNPACK 与此功能紧密相关。

PUTBIT

此项功能的输入含有一个 DWORD X，一个 BYTE N 和一个布尔值 B。PUTBIT 把值 B 上从 X 设到第 N 位，由此从零位计数。

ST 例子:

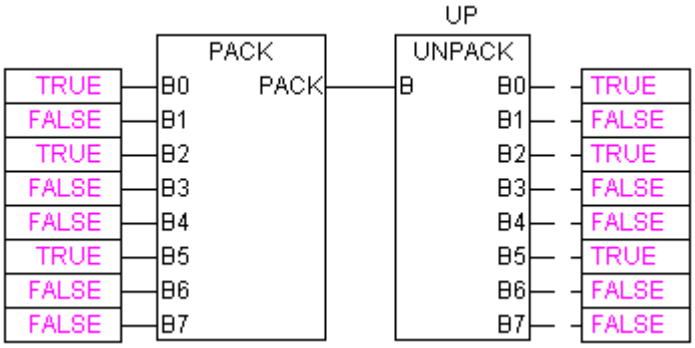
```
A:=38;      (* 二进制数为 100110 *)
B:=PUTBIT(A,4,TRUE); (* 结果: 54 = 2#110110 *)
```

C:=PUTBIT(A,1,FALSE); (* 结果: 36 = 2#100100 *)

UNPACK

UNPACK 把输入变量 B 从类型 BYTE 转换为类型 BOOL 的 8 个输出变量 B0，…，B7，这是与 PACK 相对的。

FBD 例子: 输出:



10.18.3　10.18.3 数学辅助功能

微分

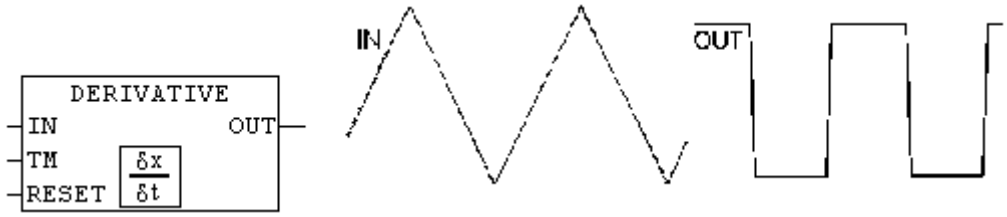
微分功能块。

功能值是使用 IN 以 REAL 类型变量表达的。TM 包含在 DWORD 中以毫秒为单位已通过的时间，类型 BOOL 的 RESET 输入可使此功能块通过释放值为 TRUE 而使其重新开始。

输出 OUT 的类型为 REAL。

为了取得最佳结果，派生约计使用最后的 4 个值，以便尽量减少输入参数不精确所产生的错误。

FBD 下的功能块:



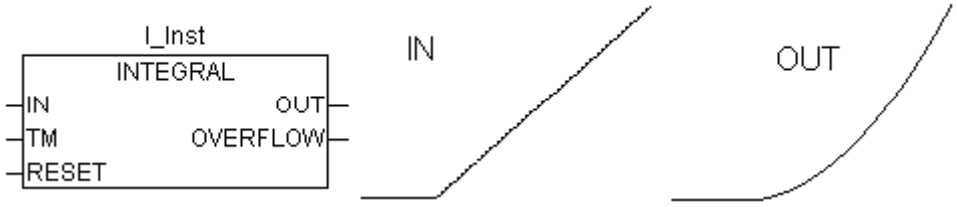
积分

此功能块大致确定功能的积分。

在派生的模拟方式中，功能值的方式是使用 IN 的 REAL 变量。

TM 含有在 DWORD 中以毫秒为单位所记录的时间并且类型为 BOOL 的输入 RESET 可以使功能块以 TRUE 值开始重新启动。输出 OUT 类型为 REAL。积分为两个步进功能的约数。这些平均值显示为约定的积分数。

FBD 下的功能块:　例如: 线性功能的综合:



LIN_TRAFO

此功能块(util.lib)转换一个 REAL 值，此致在已定义有上下值限的值域内为一个相应的 REAL 值，此值则在另一个已定义的有上下值限的值域内。下面的等式便是此项转换的基础:

(IN - IN_MIN) : (IN_MAX - IN) = (OUT - OUT_MIN) : (OUT_MAX - OUT)

输入变量:

变量	数据类型	描述
IN	REAL	输入值
IN_MIN	REAL	输入值域的下限
IN_MAX	REAL	输入值域的上限
OUT_MIN	REAL	输出值域的下限
OUT_MAX	REAL	输出值域的上限

输出变量:

变量	数据类型	描述
OUT	REAL	输出值
ERROR	BOOL	产生错误：TRUE，如IN_MIN=IN_MAX,或如IN不在输入值域内。

应用例子:

温度传感器提供电压值(输入 IN)。将此温度值转换为摄氏度值(输出值 OUT)。输入（伏特）值域已由 IN_MIN＝0 和 IN_MAX＝10 定义的上下限。其输出(摄氏度)值域则由 OUT_MIN=-20 和 OUT_MAX=40 为上下限。这样，输入为 5 伏的温度其摄氏度的结果为 10 度。

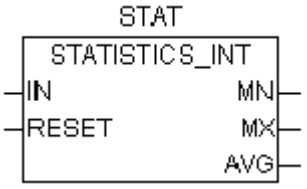
STATISTICS_INT

此功能块计算部分标准的统计学值。

输入值 IN 为 INT 类型。当布尔输入变量 RESET 为 TRUE，所有的值都会被重新初始化。

输出 MN 包含 IN 中的最小值和最大值的 MX。AVG 描述的是平均值，那也是所期望的 IN 值，三个输出变量都是 INT 类型。

FBD 下的功能块:



STATISTICS_REAL

此功能块 STATISTICS 相呼应，除了其输入变量 IN，像 MN，MX，AVG 都是 REAL 类型。

VARIANCE

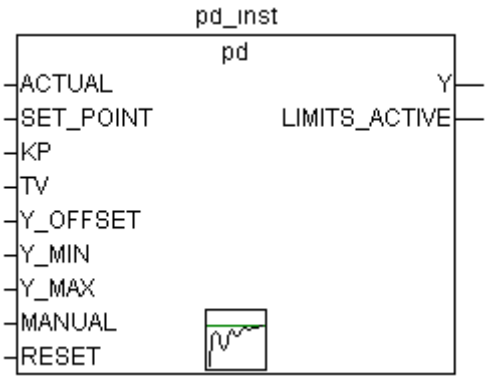
VARIANCE 计算已键入值的变异。

输入变量 IN 为 REAL 类型，RESET 为 BOOL 类型，输出 OUT 也是 REAL 类型。此块计算的输入值的变异。VARIANCE 可以用 RESET＝TRUE 重新设置。标准偏差可以用 VARIANCE 的平方根的方法很容易地被计算出来。

10.18.4 控制器

PD

PD 控制器功能块:



ACTUAL（当前值）和 SET_POINT(期望或额定值)以及 KP，均衡系数，都是 REAL 类型的输入值。TV 为 DWORD 类型，含有以秒为单位的(如 0.5 秒代表 500 毫秒)派生活动时间。Y_OFFSET，Y_MIN 和 Y_MAX 为 REAL 类型，被用于已指定域内操纵量的转换。类型为 BOOL 类型，被用于重置控制器。

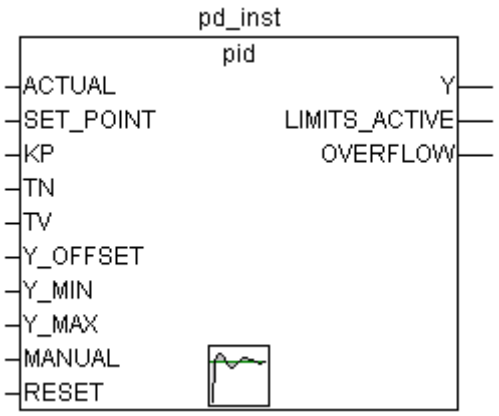
Y 也被限定在所允许的 Y_MIN 和 Y_MAX 的范围之间。如果 Y 超出此范围，作为布尔输出变量的 LIMIT_ACTIVE 便成为 TRUE。如果想让操纵量无限制，Y_MIN 和 Y_MAX 设为零。

若 MANUAL 为 TRUE，那么调节器被暂停使用，即 Y 不被更改（被控制器），如果 MANUAL 变为 FALSE，此时则会重新初始化控制器。

P 控制器可通过设置 TV，很容易使其生成一个固定的值零。

PID

PID 控制器功能块:



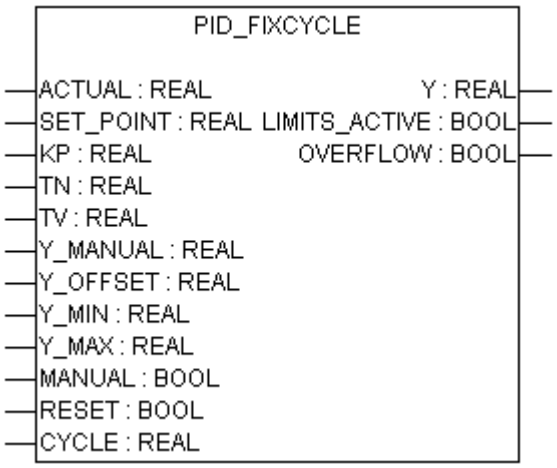
此功能不像 PD 控制器功能，它含有一个更深层次的 DWORD 输入变量 TN，以秒为单位(如 0.5 秒代表 500 毫秒)调整时间。输出操纵量（Y）还是 REAL 类型，并不像 PD 控制器那样含有一个附件的积分部分：

Y = KP x (D + 1/TN ?edt + TV dD/dt) + Y_OFFSET

PID 控制器可以很方便地通过将 TV 设置成零转为 PI 控制器。因为有了附加的积分部分，所以控制器参数输入出现错误就会造成溢出，条件是错误的积分变得过大。因此，为了安全起见，便会把布尔输出的变量 OVERFLOW 调用出来。此时其值应该是 TRUE。同时，控制器暂停，只有重新初始化时才会被再次激活。

PID_FIXCYCLE

PID_FIXCYCLE 控制器功能块:



此功能块与 PID 控制器相应，除了周期不是有内部功能自动测量，而是被输入变量 CYCLE（以秒为单位）进行设定。

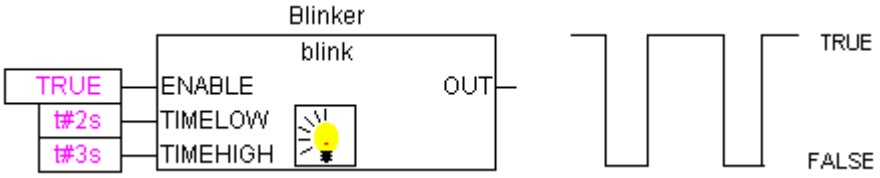
10.18.5 信号生成

BLINK

BLINK 功能块可生成脉冲信号。其输入变量由布尔类型的 ENABLE 以及 TIMELOW 和 TIMEHIGH 时间类型组成的。输出变量 OUT 类型为 BOOL。

如果 ENABLE 被设为 TRUE，BLINK 便开始生效，将时间输出变量 TIMEHIGH 设为 TRUE，然后将时间 TIMELOW 的变量值设为 FALSE。

CFC 例子：



FREQ_MEASURE

此功能块用于测量（平均）布尔输入信号的频率。可以详细列出应当平均为多少个时间段。一个时间段便为两个输入信号的两个边界的上升之间的时间。

输入变量:

变量	数据类型	描述
IN	BOOL	输入信号
PERIODS	INT	时段数量，即，边界上升的间隔时间，应当计算输入信号的平均频率。可能的值为1-10。
RESET	BOOL	把所有参数重置为0

输出变量:

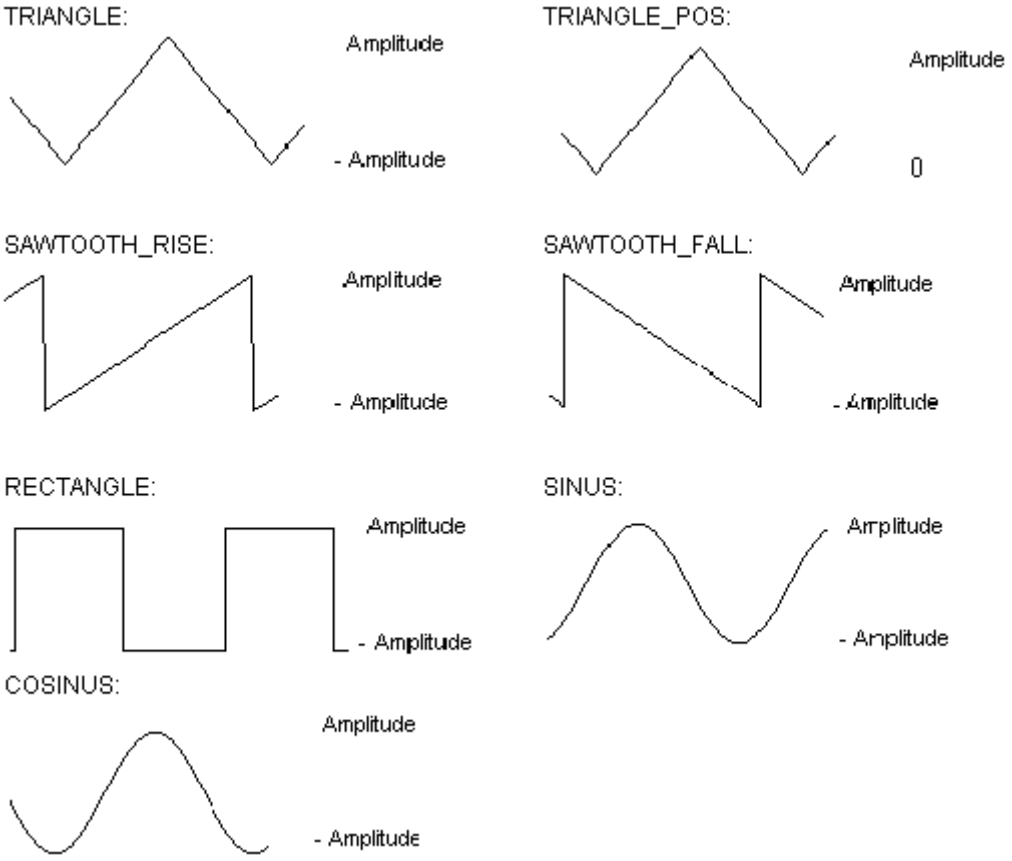
变量	数据类型	描述
OUT	REAL	得到频率[赫兹]
VALID	BOOL	首次测量之前为FALSE，或者如果时间长度>3*OUT（显示输入变量出了问题）

GEN

功能生成器生成典型的周期功能:

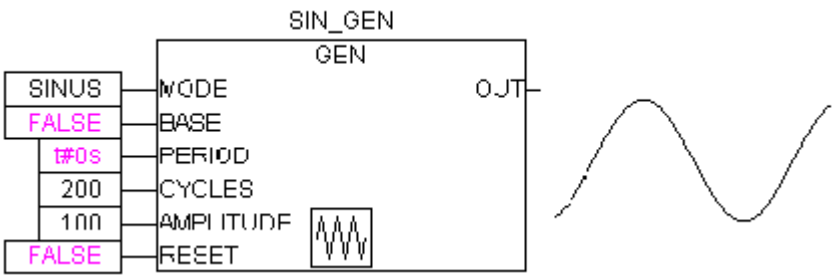
输入变量为一个由 MODE 组成，其中包括预定的计数类型 GEN_MODE，BOOL 类型的 BASE， TIME 类型的 PERIOD，组成部分中已有两个 INT 型的值 CYCLE 和 AMPLITUDE 以及布尔 RESET 输入变量。

MODE 说明所生成功能利用列举值 TRIANGLE 和 TRIANGLE_POS 传送两个三角形功能。SAWTOOTH_RISE 为一个上升功能，而 SAWTOOTH_FALL 为一个下降的锯齿，RECTANGLE 为一个矩形信号，SINE 和 COSINE 分别为正弦和余弦:



BASE 定义周期是否真的与所定义的时间（BASE＝TRUE）有关，或者它是如何与 PERIOD 或 CYCLE 定义相应的周期时间。AMPLITUDE 从细节方式定义所生成的功能的周期变量的最大绝对值。此功能生成器在 RESET＝TRUE 时就会再次被设为 0。

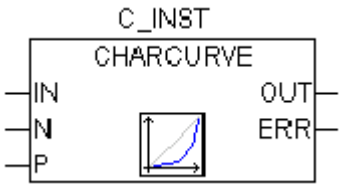
FBD 例子:



10.18.6 功能操作器

CHARCURVE

此项功能块用于在线性功能中一一表现值:



INT 类型中的 IN 要用值作操作源进行操作。BYTE N 指定点数，此点定义表现功能。此特征线随后便用类型 POINT 中的 P 在一个 ARRAY P[0……10]中生成，其中的 POINT 为一个基于 2 个 INT 值（X 和 Y）两个值的一个结构。输出的组成部分有 INT 中的 OUT。操作的值以及 BYTE ERR 此项主要在必须时显示错误。

ARRAY 中的点 P[0]…P[N-1]必须根据其 X 值进行分类，否则，ERR 便会接收到值 1。如果输入的 IN 不在 P[0].X 和 P[N-1].X，则 ERR＝2 并且含有相应限值 P[0].Y 或 P[N-1].Y。如果 N 不在所允许的 2－11 之间的值域内，那么 ERR＝4。

ST 例子:

首先，ARRAY P 必须在页首被定义:

```
VAR
 ...
 CHARACTERISTIC_LINE:CHARCURVE;
 KL:ARRAY[0..10] OF POINT:=(X:=0,Y:=0),
(X:=250,Y:=50),
(X:=500,Y:=150),(X:=750,Y:=400),7((X:=1000,Y:=1000));
 COUNTER:INT;
 ...
END_VAR
```

再提供一个值一直增加的情况下 CHARCURVE 的例子:

```
COUNTER:=COUNTER+10;
 CHARACTERISTIC_LINE(IN:=COUNTER,N:=5,P:=KL);
```

结果跟踪显示说明其结果:

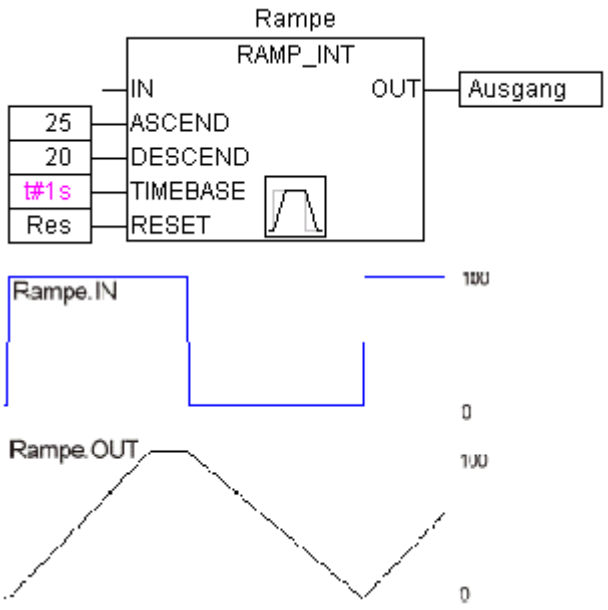
RAMP_INT

输入变量由三个 INT 值的 OUT 变量组成：IN，功能输入以及 ASCEND 和 DESCEND，每个间隔之间的最大增或减量，它又是被 TIME 类型中的 TIMEBASE 来定义的。将 RESET 设为 TRUE 会导致 RAMP_INT 被重新初始化。

INT 类型中的 OUT 输出变量中含有上升和下降的有限功能值。

把 TIMEBASE 设为 t#0s，ASCEND 和 DESCEND 虽与时间间隔无关，但是却能保持不变。

CFC 例子:



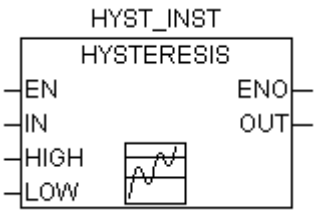
RAMP_REAL

RAMP_REAL 与 RAMP_INT 中的功能一样，不同的是，其中的输入变量为 IN，ASCEND， DESCEND 和输出变量 OUT 皆为 REAL 类型。

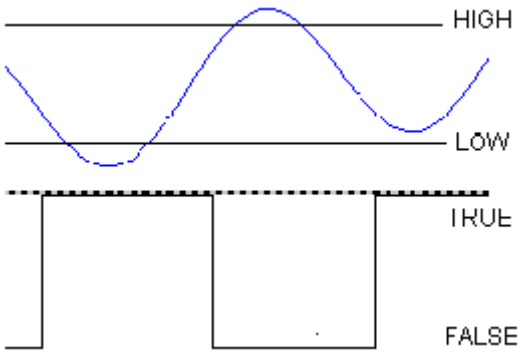
10.18.7 模拟值的处理

HYSTERESIS

此功能块的输入变量包括三个 INT 值：IN，HIGH 和 LOW。输出变量 OUT 为 BOOL 类型。



如果 IN 值低于 LOW 的限值，OUT 则变成 TRUE。若 IN 超过了上限 HIGH 的值，便会显示 FALSE 的结果。
说明示例：

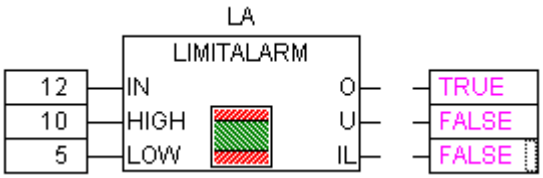


限制警告

此项功能块说明输入变量的值是否在设定范围内，如果不在范围内便超出了限制范围。输入值 IN，HIGH 和 LOW 都是 INT 类型，而输出变量 O，U 和 IL 则都是 BOOL 类型。

如果上限 HIGH 被 IN 超过，O 便变为 TRUE，而如果 IN 低于 LOW，U 就会编程 TRUE。如果 IN 介于 LOW 和 HIGH 之间 IL 则为 TRUE。

FBD 例子：结果：



10.19 AnalyzationNew.lib库

AnalyzationNew.lib

此程序库提供表达式的分析模块。如果组成表达式为 FALSE，其组成部分可以进行评估，并加注到这个结果中。在 SFC-Editor 中 SFCErrorAnalyzationTable 标记，自动使用此功能对运算过程中的表达式进行分析。

分析示例：

b OR NOT(y < x) OR NOT (NOT d AND e)

功能：

下面的变量用于所有模块：

InputExpr：BOOL，要进行分析的表达式。

DoAnalyze：BOOL，TRUE 开始分析。

ExpResult：BOOL，表达式当前值。

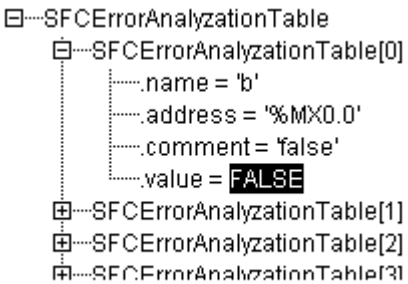
分析结果输出不同：

AnalyzeExpression 复原字符串中表达式的成分，则将其加注在整个 FALSE 中。AppendErrorString 功能用于此目的，用“|”分开输入字符串的特定成分。

OutString：STRING，分析结果，表达式相关成分的顺序（如：y<x|d）

AnalyseExpressionTable 写出表达式的成分，并加注到总值 FALSE 之中。每个成分都由 structureExpressionResult 加注如下信息：姓名，地址，备注，（现）值。

例如：



AnalyseExpressionCombined 将 AnalyzeExpression 和 AnalyseExpressionTable 的功能性合并。

10.20 CoDeSys系统程序库

注：程序中使用和支持哪些程序库取决于目前所使用的目标程序。

附录E：操作符及程序库模块总结

下表为操作符总结，它们这些操作符分别都存在于 CODeSys 的 Standard.lib 和 Util.lib 程序库中。

注意‘IL 符’栏：只有使用此符号的行才被显示。一个必要条件是：第一组必要的操作数已被顺利载入前一行（如，LD in）

‘Mod.IL’栏显示 IL 中有可能出现的修饰符:

C　前行表达式为 TRUE 的结果的修饰符。

N　对于 JMPC，CALC，RETC：如前行的结果为 FALSE，则执行此命令。

N　否则：否定操作数（非累积器）

(　括弧中操作符：只有括号出现后操作才开始执行。

请从结合到每个 CoDeSys 程序库中的相应有关附录里获取详细的使用信息。

10.21 CoDeSys中的操作符:

在 ST 中	在 AWL 中	Mod. AWL	描述
,			字符串分隔符
..			Array 值域大小
[ ]			
:			声明中操作数与类型间的分隔符。
;			说明终止。
^			指示符废止。
	LD var1	N	缓冲器中变量 1 值的载入。
:=	ST var1	N	将实际结果存入变量 1。
	S boolvar		当实际结果为 TRUE，准确设置操作数布尔变量为 TRUE。
	R boolvar		当实际结果为 FALSE,准确设置操作数布尔变量为 TRUE。

在 ST 中	在 AWL 中	Mod. AWL	描述
	JMP label	CN	直接跳至标签。
<Program name>	CAL prog1	CN	调用程序 prog1。
<Instance name>	CAL inst1	CN	调用功能程序 inst1。

<Fctname>(vx, vy,..)	<Fctname> vx, vy	CN	调用功能 fctname 和 transmit 变量 vx，vy。
RETURN	RET	CN	Leave POU and go back to caller
	(		括号后的值作操作数处理，括号前的运算在括号内的表达式之前不执行。
	)		现执行已被设置运算
AND	AND	N,(	位 AND
OR	OR	N,(	位 OR
XOR	XOR	N,(	位专用 OR
NOT	NOT		位 NOT
+	ADD	(	加法
-	SUB	(	减法
*	MUL	(	乘法
/	DIV	(	除法
>	GT	(	大于
>=	GE	(	大于等于
=	EQ	(	等于
<>	NE	(	不等于
<=	LE	(	小于等于
<	LT	(	小于
MOD(in)	MOD		模相除
INDEXOF(in)	INDEXOF		POU in1 的内部指数
SIZEOF(in)	SIZEOF		已知 in 数据类型所需字节数
SHL(K,in)	SHL		操作符 in 位左移 k
SHR(K,in)	SHR		操作符 in 位右移 k
ROL(K,in)	ROL		操作符 in 位旋转 k 到左
ROR(K,in)	ROR		操作符 in 位旋转 k 到右
SEL(G,in0,in1)	SEL		in0（G 为 FALSE）和 in1（G 为 TRUE）两个操作数之间的两进制选择。
MAX(in0,in1)	MAX		2 个值中较大的一个的复原。
MIN(in0,in1)	MIN		in0 和 in1 两值之中较小一个的复原。

LIMIT(MIN,in,Max)	LIMIT		限制值域（in 重置为 MIN，或 MAX，以免超出范围）。
在 ST 中	在 AWL 中	Mod. AWL	描述
MUX(K,in0,...in_n)	MUX		一组值中（in 至 In_n）选择第 k 个值。
ADR(in)	ADR		DWORD 中操作数的地址。
ADRINST()	ADRINST()		调用操作符即功能块操作符的地址。
BITADR(in)	BITADR		DWORD 中操作数的位平衡。
BOOL_TO_<type>(in)	BOOL_TO_<type>		布尔操作数的类型转换。
<type>_TO_BOOL(in)	<type>_TO_BOOL		类型转换成 BOOL。
INT_TO_<type>(in)	INT_TO_<type>		INT 操作数类型转换位另一基本类型。
REAL_TO_<type>(in)	REAL_TO_<type>		REAL 操作数类型转换位另一基本类型。
LREAL_TO_<type>(in)	LREAL_TO_<type>		LREAL 操作数类型转换位另一基本类型。
TIME_TO_<type>(in)	TIME_TO_<type>		TIME 操作数类型转换位另一基本类型。
TOD_TO_<type>(in)	TOD_TO__<type>		TOD 操作数类型转换位另一基本类型。
DATE_TO_<type>(in)	DATE_TO_<type>		DATE 操作数类型转换位另一基本类型。
DT_TO_<type>(in)	DT_TO_<type>		DT 操作数类型转换位另一基本类型。
STRING_TO_<type>(in)	STRING_TO_<type>		字符串操作数类型转换位另一基本类型，in 必须含所求类型的有效值。
TRUNC(in)	TRUNC		从 REAL 转换为 INT。
ABS(in)	ABS		操作数 in 的绝对值。
SQRT(in)	SQRT		操作数 in 的平方根。
LN(in)	LN		操作数 in 的自然对数。
LOG(in)	LOG		操作数 in 的以 10 为底的对数。
EXP(in)	EXP		操作数 in 的幂方。
SIN(in)	SIN		操作数 in 的正弦。
COS(in)	COS		操作数 in 的余弦。

TAN(in)	TAN	操作数 in 的正切。
ASIN(in)	ASIN	操作数 in 的反正弦。
ACOS(in)	ACOS	操作数 in 的反余弦。
ATAN(in)	ATAN	操作数 in 的反正切。
EXPT(in,expt)	EXPT expt	操作数 in with expt 的幂乘方。

10.22 Standard.lib库的元素:

在 ST 中	在 AWL 中	描述
LEN(in)	LEN	操作数 in 的字符串长度
LEFT(str,size)	LEFT	已知字符串 str 大小的左边第一个字符串。
RIGHT(str,size)	RIGHT	已知字符串 str 大小的右边第一个字符串。
MID(str,size,pos)	MID	POS 处已知大小的部分 str 字符串。
CONCAT('str1','str2')	CONCAT 'str2'	两个后继字符串的合并。
INSERT('str1','str2',pos)	INSERT 'str2',p	在 pos 处把字符串 str1 插入字符串 str2
DELETE('str1',len,pos)	DELETE len,pos	删除部分字符串（长度 len），启动位置为 str1 的 pos 处。
REPLACE('str1','str2',len,pos)	REPLACE 'str2',len,pos	替换长度字符串为 str2，起始位置为 str1 的 pos 处。
FIND('str1','str2')	FIND 'str2'	在 str1 中寻找部分字符串 str2。
SR	SR	双稳态的 FB 设置为显性。
RS	RS	双稳态的 FB 重设。
SEMA	SEMA	FB: 臂板软件（可中断）
R_TRIG	R_TRIG	FB: 上升边界被监测。
F_TRIG	F_TRIG	FB: 下降边界被监测。
CTU	CTU	FB: 升值计数。
CTD	CTD	FB: 下降计数。
CTUD	CTUD	FB: 上下计数。
TP	TP	FB: 触发
TON	TON	FB: On-Delay 时钟。
TOF	TOF	FB: Off-Delay 时钟。

RTC	RTC	FB：现实时钟。

10.23 Util.lib库的元素：

BCD_TO_INT	字节转换：BCD 转为 INT 格式。
INT_TO_BCD	字节转换：INT 转为 BCD 格式。
EXTRACT(in,n)	DWORD in 的第 n 位以 BOOL 类型还原。
PACK	多达 8 位被压入一个字节。
PUTBIT	DWORD in 一个位被设位一定的值。
UNPACK	字节被作为单个的位被复原。
DERIVATIVE	局部派生。
INTEGRAL	整数。
LIN_TRAFO	REAL 值的转换。
STATISTICS_INT	INT 格式的最小，最大和平均值。
STATISTICS_REAL	REAL 格式的最小，最大和平均值。
VARIANCE	变异。
PD	PD 控制器。
PID	PID 控制器。
BLINK	脉冲信号。
FREQ_MEASURE	布尔输入信号的测量频率。
GEN	周期性功能。
CHARCURVE	线性功能
RAMP_INT	限制供应功能（INT）减弱的上升。
RAMP_REAL	限制供应功能（REAL）减弱的上升。
HYSTERESIS	滞后作用。
LIMITALARM	注意输入值是否超出定义范围的限制。

附录F：命令行/命令文件

10.24 命令行相关命令

当 CoDeSys 起动后，你可以在命令行里添加命令，程序执行过程中该命令仍会显示。这些命令都要用一个“/”开头。字母大小写均可。随后命令就会自左至右连续地执行。

/online(联机)	启动后，CoDeSys 会立即对当前项目联机。
/run(运行)	登录后 CoDeSys 启动应用程序。 仅在显示/online 时有效。
/show…(显示…)	可在 CoDeSys 提示框里设定。
/show hide（隐藏）	提示框不显示内容。任务菜单里也不显示。
/show icon（显示图标）	窗口显示最小化。
/show max（最大显示）	窗口显示最大化。
/show normal（正常显示）	窗口关闭前一直以同样状态显示。
/out(outfile)(读写)	所有信息均在提示框里显示，同时写入文件。
/noinfo(无信息)	CoDeSys 启动后无信息显示。
/userlever(group)用户（组）设定	定义用户（组），如/userlevel0 即指用户组 0。
/password（口令）	直接输入用户口令，如“/password abc”
/openfromplc(开放的程序系统)	当前已连接的任务系统项目被调出运行。
/visudownload(可视下载)	在 CoDeSys HMI 运行后，如果与当前连接的任务系统不匹配，会提示下载程序。(有对话框，可选 YES（是）或 NO（否）关闭）
/notargetchange（不可改变任务）	A任务系统的改变必须通过一个文件才能实现。见 10.25 章，命令“任务...”
/cmd(cmdfile)命令（文件）	启动后，命令（cmd文件）开始执行。

注意命令行的句子结构：

“<CoDeSys 可执行文件的路径>”“<项目路径>”/命令 1/命令 2/…

例如：

若想在 ampel.pro 运行后，窗口不显示。命令文件 command.cmd 里的命令即开始执行。注意用双引号把路径引起来。“D:\dir1\CoDeSys”“C:\project\ampel.pro”/showhide/cmd command.cmd。

10.25 命令文件（cmdfile）命令

命令文件里（cmdfile）可用的命令，可见以面的列表。你可以通过一个 命令行（见上）访问命令文件。命令行会在信息窗口里里予以提示，它会在提示文件里给出，除非命令前加上前缀“@”。

所有分号“；”后的符号会被忽略（注释）。所有含有表格的参数必须用引号标记上。Umlauts 只有在命令文件是用ＡＮＳＩ码创建时才使用。命令参数里可以用关键词。下面你即可看到一些与相关命令有关的关键词列表。

控制后续命令的命令：

onerror continue	即使出错时后续命令也继续执行。
onerror break	只要检测到出错，后续命令即不再执行。

联机菜单命令：

online login	运行程序时登录（‘登录’）
online logout	退出（‘联机’‘退出’）
online run	起动应用程序（‘联机’‘运行’）
online stop	终止应用程序（‘联机’‘停止’）
online bootproject	创建一个引导 工程,此命令可在离线和联机模式下应用!（参看命令描述 ’联机’’创建引导工程’!）
online sourcodedownload	把项目的源代码下载到 PLC 里（‘联机’‘源代码下载’）
online sim	开通模拟模式‘联机’‘仿真’
online sim off	终止模拟状态‘联机’‘仿真’

文件菜单命令：

file new	创建一个新项目（‘文件’‘新建’）
file open(projectfile)	项目<projectfile>开始运行(‘文件’‘打开’）
可选项：	
/readpwd:<readpassword>	在可读保护开启时，输入口令才不会出现要求输入口令的对话窗口。
/writepwd<writepassword>	在可读保护开启时，输入完整的口令才不会出现要求输入口令的对放大窗口。
file close	当前项目被关闭。‘文件’‘关闭’
file save	当前项目被存储。’文件’’保存‘
file saveas<projectfile>	当前项目以<projectfile 项目文件>文件名存储。
可选项<type><version>	(‘文件’‘另存为’） 缺省值：在当前 CoDeSys 版本，项目被存储为<projectfile>.pro。如果你想把项目存储为内外部库或者早期版本，可分别加下边的命令： 可在<type>中加入： “internallib”存储为内部库： “externallib”存储为外部库： “pro”存储为早期版本： 在<version>中输入有效的条目：15，20，21，22（指产品的版本 1.5,2.0,2.1,2.2） 例如：“文件另存为 lib_xy internallib22”->在当前 CoDeSys 版本 V2.2 中创建的项目“project xy.pro”被另存为“lib_xy.lib”。
file saveas<projectfile>	当前项目以<projectfile 项目文件>文件名存储。 (‘文件’‘另存为’）
file	定义一个文件的结构（’文件’打印机设置’），也可以定义打印选项’按对象分页’或‘按子对象分页’；这些设定对文件的打印有影响。（项目或 文件见下面）
printersetup<filename>.dfr	
可选项：	
pageperobject	

pagepersubject	
file archive<filename>.zip	项目会以一个压缩文档存储，文件名是给定的。（‘文件‘保存/发送压缩文件’）
file quit	CoDeSys 关闭。（‘文件’退出‘）

项目菜单命令：

project build	对已运行的项目进行编译（‘工程’’生成‘）
project rebuild or project compile	可对运行的项目进行充分的编译（‘工程’‘重新生成’）
project clean	当前项目的编译或联机变动信息被删除。（’工程’‘清除工程’）
project check	对已运行的项目进行检查（’工程’’全部检查）
project compile	当前项目通过输入“全部重建“命令进行编译。（’工程’’全部重新生成’）
project check	对当前项目进行检查（‘工程’‘检查全部’）
project build	生成当前项目。（’工程’生成’）
project import<file1><file2>…<fileN>	用‘工程’ ‘导入’ 命令把文件 1 到 N 输入到当前项目。注意：这时可以用通配符。如命令“project import C:\project*.exp”就会将所有扩展名为*.exp 的文件输入到创建的目录 C:\projects。
project export<expfile>	以文件<expfile>输出当前项目。（‘工程’’导出’）
project expmul	当前项目的每个任务独立输出，文件名为每个任务的名称。
project documentation	整个项目由默认的打印机输出。（’工程’’文档’，可见上面‘文件打印设定’）

信息文件控制命令：

out open<msgfile>	以信息文件方式打开<mmsgfile>。新的信息聚即被填加。
out close	关闭当前打开的信息文件。
out clear	当前打开的所有信息文件被删除。

信息控制命令：

echo on	显示命令行。
echo off	不显示命令行。
echo<text>	提示框里显示<text>

控制文件输入、输出、复制时单个任务更新的命令：

replace yesall	全部更新（所有‘询问’命令均被忽略，无对话窗出现）。
replace noall	均不更新。(所有‘询问’命令均被忽略，无对话窗出现）
replace query	设定一个‘询问’命令后，即使有‘全部替换’或‘全部不替换’命令，对话窗口也会开启。

CoDeSys 对话默认参数设定命令：

query on	显示对话窗口并需要用户输入。
query off ok	所有对话均响应为好象用户单击‘是’按钮。
query off no	所有对话均响应为好象用户单击‘否’按钮。
query off cancel	所有对话均响应为好象用户单击‘取消’按钮。

命令文件作为子程序调用的命令

call<parameter1>…<parameter10>	命令文件将作为子程序调用。最多可有 10 个参数。在被调用的文件里，参数可设定 $0~$10。
call<parameter1>…<parameter10>	命令文件作为子程序调用。最多可有 10 个参数。在被调用的子程序里，参数可设定 $0~$10。

CoDeSys 使用的目录设定（－>项目选项对话，分类有‘directories 目录’，子目录‘general’）:如果几个目录都用下面中的一个设定，他们必须用分号加空格分开，同时整个目录行用双引号引上。如，两个路径：

Dir lib ODQ\codesys\Libraries\Standard DQ\codestys\Libraries\NetVaro

dir lib<libdir>	设定<libdir>为库文件夹。
dir compile<compiledir>	设定<compiledir>为编译文件夹。
dir config<configdir>	设定<configdir>为配置文件夹
dir upload<upload>	设定<uploaddir>为加载文件夹

延迟编译进程：

delay 5000	延迟 5 秒

监控接收管理器控制：

watchlist load<file>	加载监控文件保存为<file>,同时打开相应的窗口。(‘附加’‘加载监控文件’)
watchlist save<file>	保存当前的监控清单为<file>(‘附加’’’保存监控文件’)
watchlist set<text>	监控清单可以主动设置（与监控接收管理器窗口的左边列表选择相对应)。
watchlist read	更新监控变量的值（‘’附加’’‘读取数据处方’）
watchlist write	用在监控清单中建立的值填入监控变量。(‘附加’‘写入数据处方’

库链接：

library add<library	file1><library 填加指定的库文件到当前打开项目

file2>…<library fileN>	的库列表。如果文件路径是相关的，项目里的库目录占用一个根路径。
library delete<library1><library2>…<libraryN>	从当前打开的项目库列表里删除指定的库。

复制任务：

object project copy<source file><source path><target path>	从源项目文件里复制指定路径的任务，到已打开项目的目标路径。 如果源路径是一个对象的名字，则被复制。 如果是一个文件夹，文件夹里的所有对象均被复制。这种情况下，源文件夹里的结构也被复制。如果目标路径还不存在，会创建一个。

特殊任务的只读访问：

object setreadonly<TRUE | FALSE><object type><object name>	设定某一任务的只读访问；定义任务类型，假定任务类型 pou,dut,gvl,vis 也是任务的名称。 可能的任务类型有：pou,dut(数据类型)，gvl(共用变量列表)，vis(可视)，cnc(数控任务)，liblist(库)，targetsettings,toolinstanceobject(特指工具举例)，toolmanagerobject(工具树里的所有举例)，customplcconfig(plc 配置)，projectinfo(项目信息)，tastconfig(任务配置)，trace,watchentrylist(/监控接收管理器)，alarmconfig(报警配置)。 如：“object setreadonly true pou plc_prg”可设定只读访问 PLC_PRG。

输入通信参数（网关、设备）

gateway local	设定本地计算机上的网关为当前网关。
gateway tcpip<Address><Port>	把指定的远程计算机网关设定为当前网关。 <地址>远程计算机的 TCP/IP 地址或主机名。 <端口>远程网关的 TCP/IP 端口。 重要提示：只有那些没有设有口令的网关可联通！
device GUID<guid>	设定指定的 GUID 为当前设备。 GUID 必须有下面的格式： {01234567-0123-0123-0123-0123456789ABC} 弯括号和连字符必须出现在指定位置。
device instance<instance name>	设定当前设备的名字为指定名字。
device parameter<id><value>	赋以指定的值，它由设备用指定的 ID 解译成参数。

系统访问：

system<command>	完成指定的操作系统命令。

选择任务系统：

target<id>	为当前项目设定任务平台。如果 CoDeSys 起动时显示

	命令行选项“/notargetchange”（见 10.24 章），只能通过该命令设定一个任务。

在 ENI 数据库中管理项目的有关命令：

下面是命令占位符描述中使用的命令：

<category>:可用“工程”“共享”“编译”代替，这要视相关的数据库类型是项目任务，共享任务，编译任务再而定。

<POUname>:与 CoDeSys 中使用的任务名称相应的任务名称。

<objecttype>:可用快捷方式代替，它可作为扩展名填加到数据库中任务的 POU 名称，同时它反映任务的类型（在任务类型列表里定义，见 ENI 管理中’Object types’部分。）

例如：任务“GLOBAL_1.GVL”的 POU 名称为“GLOBAL_1”，任务类型是“GVL”（共用变量表）。

<comment>可用注释文本代替（用单引号括起来），它用特殊的方式存储在版本历史记录里。

链接到 ENI 服务器的任务数据库配置命令：

Eni on Eni off	激活或不激活“(ENI)用户源程序控制”选项。(对话框‘工程’‘选项’‘工程源控制’)
Eni project readonly on/off	在数据库种类‘工程对象’中激活或不激活只读选项。（对话框‘工程’‘选项’‘工程对象’）
Eni shared readonly on/off	在数据库种类‘共享对象’中激活或不激活只读选项。（对话框‘工程对象’‘选项’‘共享对象’）
Eni set local<POUname>	任务指定为种类‘本地’，即它不被存储到项目数据库中。(对话框‘工程’‘对象’‘属性’‘数据库连接’)
Eni set shared<POUname>	赋值为共享任务‘共享对象’.(对话框‘工程’‘对象’‘属性’‘数据库连接’)
Eni set project<POUname>	赋值为项目任务‘工程对象’(对话框‘工程’‘对象’‘properties 属性’‘数据库连接’).
Eni<category>server <TCP/IP_Address> <Port><Projectname> <Username><Password>	为‘工程’类型配置到 ENI 服务器的连接。(对话框‘项目’‘选项’‘工程数据库’)，例如： Eni project server localhost 80 batchtest\project enibatsh batch 即指： (TCP/IP 地址为当地主机，端口＝80，项目名称＝batshtest\project, 用户名＝EniBatsh, 用户名＝Batch.)
Eni compile sym on/off	激活或不激活 ‘编译文件’类型中的任务选项‘创建 ASCII 码信息（.sys）’ （对话框有编译文件中的‘工程’‘选项’‘项目源程序控制’‘ENI 设定’）
Eni compile sdb on/off	激活或不激活 ‘编译文件’类型中的任务选项‘创建二进制制码信息(.sys)’。(对话框有编译文件中的‘项目’‘选项’‘项目源程序控制’‘ENI 设定’)
Eni compile prg on/off	激活或不激活 ‘编译文件’类型中的任务选项‘创建引导项目（.sys）’。(对话框有编译文件中的‘项

目’‘选项’‘项目源程序控制’‘ENI 设定’）

用数据库工作的‘项目’‘数据库链接’菜单命令：

Eni set<category>	任务按指定的数据库类型赋值（‘定义’）
‘eni set<category>set: <Objecttype>: <POUname>	用空格分开的列示任务按指定的数据库类型赋值（‘多重定义’）
<Objecttype>: <POUname>	例如：“eni set project pou:as_fub:st_prg”->任务(pou)as_fub 和 st_pub 按项目任务的类型赋值。
eni<category>getall	从数据库中调入指定种类所有任务的最新版本。（“获得所有最新版本”）。
‘eni set<category>get: <Objecttype>: <POUname> <Objecttype>: <POUname>	从数据库中调入指定种类的任务，他们用空格键分开列示。（“multiple define 综合定义）。（调入最新版本”）。例如：“eni project get pou:as_fub gvl:global_1”->从数据库中调入 pou as_fub.pou 和通用变量列表 global_1.gvl。
Eni<category> checkoutall ‘<comment>’	在数据库验证所有指定种类的任务。定义的备注在版本历史记录中以验证方式存储。
Eni<category>checkout ‘<comment>’ <Objecttype>: <POUname> <Objecttype>: <POUname>	数据库中单独列示的所有任务（任务类型：POU 名称）均被验证，他们用空格键分开列示。对每一个特定的任务而言，定义的备注在版本验证历史记录中以验证方式存储。例如：“eni project checkout”for working on xy’’ pou:as_fub gvl:global_1”-> pou as_fub 和通用变量列表 global_1 会被验证，同时注释“for working on xy”即用这种方式存储。
Eni<category>checkinall ‘<comment>’	项目数据库中所有源程序控制的项目中的任务均被检查。定义的注释以这种方式存储。
Eni<category>checkin ‘<comment>’ <Objecttype>: <POUname> <Objecttype>: <POUname>	数据库中单独列示的所有任务（任务类型：POU 名称）均被检查，他们用空格键分开列示。对每一个特定的任务而言，定义的备注在版本检查历史记录中以验证方式存储。（见上述验证部分）

命令参数关键字：

下列关键字加上“$”可以在命令参数中使用：

$PROJECT_NAME$	当前 CoDeSys 项目的名称（文件名没有扩展名“.pro”,如“project_2.pro”）
$PROJECT_PATH$	当前 CoDeSys 项目文件目录的路径。（没有驱动器的提示，尾部也没有反斜杠，如“projects\sub1”）.
$PROJECT_DRIVE$	当前 CoDeSys 项目文件所在的驱动器。（结尾没有反斜杠，如“D：”）

$COMPILE_DIR$	编译当前 CoDeSys 项目文件（有驱动器的提示，尾部没有反斜杠，如“D:\codesys\compile”）
EXE_DIR	Codesys.exe 可执行文件所在的目录。(有驱动器的提示，尾部没有反斜杠，如“D:\codesys”

命令文件举例：

下述的命令文件会打开项目文件 ampel.pro，接着会运行一个存储为 w.wtc 的监控列表，然后起动一个应用程序，约 1 秒钟后开始写入变量值到监控列表 watch.wtc(将被保存)中，最后关闭项目。

File open c:\projects\codesys_test\ampel.pro(打开该文件)

Qury off ok（关闭对话）

Watchlist load c:\work\w.wtc（运行监控列表文件）

Online login（登录）

Online run（登录后运行）

Delay 1000（延迟）

Watchlist read（读入监控列表）

Watchlist save $project_drive$\$project_path$\w_update.wtc（保存）

Online logout（退出）

File close（关闭）

这个命令文件会打开项目文件 ampel.pro，接着会运行一个已有 w.wtc 的监控列表，它起动一个应用程序,约 1 秒钟后开始写入变量值到监控列表 w_update.wtc,它将被保存到目录 c:\projects\codesys_test 中，最后再关闭项目。

下述所示则表示一个命令行被调入：

"<path of codesys.exe>" /cmd "<path of cmd file>"

附录 G：导入 Siemens 产品数据

在’工程’’导入 Siemens 产品数据’子菜单中，可以使用相应的命令从 Siemens STEP5 文件中导入 POUs 和变量. 命令"导入一个 SEQ 符号文件" 将导入 STEP5 符号文件中的全局变量. 在执行命令 ’导入一个 S5 工程文件’之前执行此命令,当导入 POU 时创建可读的符号名 这两个命令用于导入 STEP5 程序文件中的 POU. 当命令执行完后,POUs 将插入到当前打开的 CoDeSys 工程. 你来决定导入的 POU 的编程语言,是继续使用 STEP5 IL 语言,还是转换成 IEC 语言.

我们建议将工程导入到一个空的 CoDeSys 工程中,当然，你必须确定已经插入 standard.lib，否则不可能导入计数器和定时器功能块.

参看:

导入一个 SEQ 符号文件

导入一个 S5 工程文件

将 S5 转换成 IEC 61131-3

10.26 导入一个SEQ符号文件

SEQ 格式是 STEP5 工程中符号文件的通用格式.符号的分配可以从 SEQ 符号文件(*.seq)中读取.每个符号分配的组成包含一个 S5 程序元件(输入,输出,内存位置等)的绝对地址，一个相应的符号标识符及关于符号的注释.SEQ 文件是文本文件，文件中的每行都包含一个类型分配.行中的每个字段用制表符分开.每行只能有一个用分号开头的注释.

在 SEQ 文件中的符号分配将转换成基于 IEC61131-3 标准的全局变量声明.符号的名称,地址和注释(如果有)将在处理过程中被转移.地址要适合于 IEC 61131-3 标准 (%, 等).如果在 S5 符号名称中包含了不允许作

为 IEC 标识符的字符,这个名称将被修改. 无效的字符用下划线替换.这个可能在一行中会出现多个下划线,每行的第二个将由字符替换(如, "0"). 在转换过程如果修改了符号名称,那么在改变后原先的名称被添加到注释中.SEQ注释将直接转换成IEC注释. 可能会创建多个全局变量数据块. 每个数据快至少有64K文本组成.

描述的 SEQ 格式用于 Siemens STEP5-PG 软件,多数版本的 Siemens STEP7-300/400,和 DELTALOGIC 的 ACCON-PG. 由STEP7版本3.x或更高版本创建的STEP7-SEQ文件支持这种格式. 由STEP7版本2.x导出的SEQ文件不被支持. 这种格式不使用分隔符(制表符), 而是使用固定长度的符号名称,如有必要用空白填补.

首先在标准的 Windows 对话框中选择 SEQ 文件. 然后执行导入,导入完成后编译全局变量列表. 当STEP5/7 标识符转换成 IEC61131-3 相兼容的标识符时,在这个处理中可能会产生错误. 例如, STEP5 标识符"A!"和 "A?"将被转换成 IEC 标识符 "A_",这种情况下,出现下面的信息, "标识符 A_声明多次".修改其中的一个变量.

在绝对没有其它原因的情况下,你不需要对全局变量列表做修改.如果识别的地址 Siemens PLC 中有效,在你的控制器中无效,即使在编译中有上千个错误信息,也保留它们,当按顺序导入 POU 时需要准确的地址.

如果要导入的工程中已经包含了全局变量 x 的声明且带地址(如:"%MX4.0"),而且在 SEQ 文件中同样包含相同地址变量的定义,这在 IEC61131-3 中是允许的.因此没有错误信息显示,但程序中不同的 POU 中使用同一个地址.为避免这个问题,最好使用空的工程或工程中未使用这个绝对地址.

SEQ导入后,STEP5/7 POU可以被导入.也可以在PLC配置中添加输入和输出点. 这些不用于STEP5/7导入,但当重新编译生成工程时,将检查地址,可能会有错误.

10.27 将S5 转换成IEC 61131-3

当导入STEP5时,选择一个IEC语言作为目标语言,工程中POU并不是都能转换成IEC61131-3.如果S5 POU中包含的代码不能转换成 IEC61131-3,将产生错误信息,不能转换的代码作为注释添加到 IEC POU 中.因此这部分代码需要重新编写.只在特定的 S5 CPU 中起作用的系统命令也不能转换到 IEC.尽管 STEP5 有很大不同,但点击按钮后'STEP5 内核命令表'可以转换成 IEC 代码.

可以转换成IEC61131-3的内核命令表包含了可以转换成STEP5编程系统中的LD或FBD的所有命令,和在STEP5-FB(程序块)允许的所有命令.另外,在 IL 或 FB(功能块)中允许的 STEP5 命令且可以转换成 IEC 的命令是那些可以在 S5 CPU 中得到的命令(如,绝对和相对跳转,移位命令等)

转换的例外或限制与在STEP5中执行的复位定时器有关,而不是在IEC61131-3中.

单独可转换的命令:

U, UN, O, ON, S, R, = 带下面的操作数: I (输入), O (输出), M (内存位置), S (S 内存位置), D (数据块中的数据)

U, UN, O, ON 带下面的操作数: T (定时器), C (计数器)

S, R 带下面的操作数: C

SU, RU, P, PN 带下面的操作数: E, A, M, D

O, O(, U(,)

L, T 带下面的操作数范围: E, A, M, D, T, C, P (外设) 和字节: B (字节), W (字), D (双字), L (高字节), R (低字节)

L 带下面的常量格式: DH, KB, KF, KH, KM, KT, KZ, KY, KG, KC

SI, SE, SA 带下面的操作数: T

ZV, ZR 带下面的操作数: C

+, -, X, : 带下面的操作数: F (固定点数), G (浮点数)

+, - 带下面的操作数: D (32 位固定点数)

!=, ><, >, <, >=, <= 带下面的操作数: F, D, G

ADD 带下面的操作数: BF, KF, DH

SPA, SPB 带下面的操作数: PB, FB (带各种参数类型), SB

A, AX 带下面的操作数: DB, DX
BE, BEA, BEB
BLD, NOP, ***
UW, OW, XOW
KEW, KZW, KZD
SLW, SRW, SLD, RRD, RLD
SPA=, SPB=
SPZ=, SPN=, SPP=, SPM=
TAK
D, I
以上是大多数操作数命令
不可转换的命令
U, UN, O, ON, S, R, = 带下面的位操作数: 定时器和计数器位 (T0.0, C0.0)
L, T 带下列操作数范围: Q (扩展外设)
LC 带下面的操作数: T, C
SV, SS, R, FR 带下面的操作数: T
FR 带下面的操作数: C
用于启动,复位和释放定时器的操作数命令
带操作数的所有命令来自 BA, BB, BS, BT (操作系统数据).
SPA, SPB 带下面的操作数: OB (只与确定的 S5 和 OB 一起用)
BA, BAB 带下面的操作数: FX
E, EX 带下面的操作数: DB, DX
STP, STS, STW
DEF, DED, DUF, DUD
SVW, SVD
SPO=, SPS=, SPR
AS, AF, AFS, AFF, BAS, BAF
ENT
SES, SEF
B 带下面的操作数: DW, MW, BS
LIR, TIR, LDI, TDI, TNW, TXB, TXW
MAS, MAB, MSA, MSB, MBA, MBS
MBR, ABR
LRW, LRD, TRW, TRD
TSG
LB, TB, LW, TW 带下面的操作数: GB, GW, GD, CB, CW, CD
ACR, TSC
BI
SIM, LIM

如果检查不能转换的命令,将会发现他们是特殊的命令,只能在特定的CPU中使用.不能转换成IEC的标准命令是:加载BCD定时器或计数器值(LC T, LC C),定时器类型SV和SS, 以及复位定时器.

数据块

转换到POU中的STEP5数据块只有头文件而没有代码.如果数据块用于变量是很方便的,如果用于试图实

现象 STEP5 程序中的数据块实例是很不方便的.

当从 STEP5 导入时的问题

采用下列方法可以手动改进 STEP5 的导入.

1. 字变量中的时间值

在 STEP5 中,在内存区域或数据块中的每个字地址可以使用时间值,但在 IEC61131-3 中不允许.时间变量或常量与字地址不兼容.当从 STEP5 中导入时这会产生错误的命令序列.如果打开数据块并为地址选择时间格式(KT),可以避免这个错误发生.换句话说,只有改变 STEP5 程序时才会发生.显示的错误信息是"不兼容的类型:不能转换 WORD 到 TIME"或"不兼容的类型:不能转换 TIME 到 WORD".然后必须修改 WORD 变量的声明,转变成 TIME 变量.

2. 访问数据块失败

在 IEC61131-3 中没有数据块,也不可能完整的在 IEC 中重新创建它们.在 STEP5 中它们被用于标准的变量区(如同内存区一样),也是数组(B DW),指针(B MW100 A DB 0),联合(DB 中的字节,字或双字)形式.如果是结构化的,STEP5 转换只转换 DB 访问.当试图访问 DB 时,必须知道打开了哪个 DB(A DB)

必须注意:当一个 DB 操作在同一 POU 开始位置时,或包含的 DB 号和 POU 作为参数时.如果在第一次 DB 访问前未找到 DB,POU 不能被转换.提示信息是'插入 DB 时不能打开数据块'.在转换的 POU 中,当最新转换的 POU 编译时,假如未定义的变量"ErrorDW0",那么产生错误信息.需要用正确的 DB 来替换访问的变量,如用"DB10.DW0"替换"ErrorDW0",其它选项将丢弃转换的 POU,在 STEP5 的开头插入一个 DB.

访问数据字(数据字节等)总是先打开数据块.如果有必要,在导入前在 POU 的开头部分插入一个正确的 DB 命令来修改 POU.否则,转换后需要修改转换的 POU.

如果有多个 A BD 操作需要跳过,转换会产生一个错误,即产生的代码访问错误的 DB.

3. 与数据块访问有关的概念

在 STEP5 中,有选项通过代码块打开一个数据块的索引版本来创建相似的实例.可以按下面的代码顺序实现:

```
L KF +5
T MW 44
B MW 44
A DB 0
```

在这个序列的结尾处(一般说来,将打开在内存位置字%MW44 处找到的 DB 号)打开 DB5,这种类型的访问在转换中不被认可,即在转换后要做下列修改:

首先,所有的 DB 被导入作为 DB 实例,如 DB5 和 DB6.它们导入为你希望的 IL,LD,或 FBD POU.POU 中没有代码,而是包含局部变量的定义的头文件.可以从这些 POU 中创建类型实例.创建用户定义类型(如,DBType),插入局部变量和转换的 DB 作为组件.然后在全局变量列表中创建这个类型的全局实例:

```
VAR_GLOBAL
 DB5, DB6 : DBType;
END_VAR
```

现在从工程中删除转换的 DB.

然后需要创建 DB 索引版本的备份,以便在相应的 POU 中为 DBType 设置 VAR_INPUT 参数.对于这个实例,在 POU 中的数据访问改变了,在打开时,必须包含一个 DB 实例作为实际的参数.

4.所谓具有 STEP5 访问接口的集成 S5 功能块有一个特殊的功能,它们功能的实现不能被写入 STEP5, 它受特殊的机械结构保护.这种 POU 一般是固化的,只能作为一个接口导入.这种类型 POU 的执行部分是空的.在被转换需要重新编写它们.

5.也有不带接口的固化 OB,它的代码在 805xx 汇编程序中(例如),而不是在 STEP5 中.这会影响列出的 PID 作为 OB251,OB251 通过选择的单独数据块获得它的参数和局部变量. 与数据块相对应的 PID 调节器,或使用调节器访问数据块的 POU 不能转换成 IEC.在转换过程中为数据块或那些 POU 创建的 IEC 代码在没有 PID 调节器

的情况下是无效的.每个程序的功能可以查看 CPU 的编程手册.

6.有时用于配置 S5 CPU 和其它汇编的配置数据块(如 DB1 [S5-95U], DX0 和 DX2)将转换成无用的 IEC POU. 大多数这种类型的数据的含义可以在 CPU 的编程手册中找到.对于其他的,必须使用 S5 编程系统来评价配置的 DB.配置对于通信,模拟量处理,多处理有影响.因此,在非 Siemens SPS 上使用这些 POU 是没有用的.

在导入完成后,将显示错误.然后你需要纠正错误.错误的位置按如下注释方式标记出来:

(*警告! 不能转换的 STEP5/7 代码将按注释显示:*)

然后下面是不能转换的代码,它们也按注释方式显示.

最后,必须检查地址.在导入时创建原始的 Siemens 地址. 这些地址的格式如下:

Bits: Byte-Offset.Bit-Nummer

Non-Bits:Byte-Offset

在序列中的字地址也可以重叠.(简单的说,在地址中的数字是字节的偏移量)这意味着%MW32 和 %MW33 有一个重叠的字节,即%MB33 (只在 Siemens SPS 上). 对于你的 SPS, %MW32 和%MW33 出现重叠是不允许的.

你的 PLC 可能是多层次的.例如,非位的变量有互锁层次("%MW10.0.0"是字),你可以通过改变地址使它们适应你的 PLC,或完全不考虑它们.必须小心.在原始的 Siemens 程序中,在相同的内存位置上字的访问和位或字节访问是一样的.在导入 CoDeSys 时,这种类型的访问只对数据块的编译是正确的.在这种情况下,CoDeSys 将为 DB 中的字创建字变量.那么当访问 DB y 中的 x 字时不会有问题.因此可以访问 x 字的高字节或低字节,双字或位,并将它们编译到复杂的表达式中.由于在标准的访问方式(如字访问)下不允许这么做,所以在带有内存地址,输入或输出的情况下也不允许.如果将%MX33.3 和 %MB33 或 %MW32 或 %MD30 编写在一块,你必须自己转换它们.通过 CoDeSys 导入生成的 IEC 程序将不能正确运行.

为了检查访问,打开包含所有输入,输出和内存区域的交叉参考列表是很必要的,手动删除交叉访问.

10.28 导入一个S5 工程文件

POUs 可以读取 Siemens S5 程序文件 (*.s5d). 使用的代码是可以在 S5 SPS 中运行的 MC5 代码. 一般说来, MC5 代码与编程人员熟悉的 STEP5 指令表语言 (不带符号名)一致. S5D 中包含来自 STEP5 指令表的行注释. 由于 S5D 文件中只包含不带符号名的绝对地址,而 CoDeSys 在当前工程中搜索符号名,如果没有找到,保留绝对地址不改变,因此如果你认为符号名有用,那么在导入 S5 文件前先导入 SEQ 文件.

你首先在标准 Windows 对话框中选择 S5D 文件. 此时弹出另一个对话框,它包含了 POU 列表,你可以它们中选择.最好全选.你也可以选择 POU 使用 STEP5 IL 语言或转换成 IL, LD 或 FBD.

符号名将尽可能多的替换绝对地址名称.如果在导入过程中 CoDeSys 遇到指令 "U M12.0", 它将在内存位置 M12.0 查找全局变量 符合描述的第一个声明将被使用,指令被导入并将"U M12.0"替换成 "U-Name" (对于内存位置的标识符名称是 M12.0).

有时,在导入和代码转换过程中需要附加的变量.这些附加变量被声明为全局变量.例如, R_TRIG 实例需要再现边沿触发输入(例如, 在 S5 计数器中).

附录I 应用键盘

10.29 应用键盘

如果你想只应用键盘来运行 CoDeSys，你将发现必须应用一些在菜单中无法找到的命令。

功能键 <F6> 允许用户向前或向后打开 POU 的声明和指令部分。

<Alt>+<F6> 组合键允许用户从一个打开的对象移动到对象管理器，如果消息窗口打开的话也可以转移到那。如果搜索栏打开着，这个组合键将允许用户从对象管理器转换到搜索栏。

按<Ctrl>+<F6> 移动到下一个打开的编辑窗口。按 <Ctrl>+<Shift>+<F6> 返回上一级菜单.

按<Tab> 可以在对话框的输入区域和按钮间移动。

方向键允许用户在库管理器的注册卡间和对象管理器的对象间进行移动。

所有的其它操作可以通过菜单命令或命令后的快捷键来实现。上下文菜单针对选择的对象或激活的编辑器包含了最常用的命令，<Shift>+<F10>组合键可以用来打开它。

10.30 组合键

下面是所有组合键和功能键的概述:

一般的功能	
在 POU 的声明部分和命令部分切换	<F6>
在对象管理器，对象和消息窗口间切换	<Alt>+<F6>
上下文菜单	<Shift>+<F10>
声明的快捷方式	<Ctrl>+<Enter>
从消息窗口的一个消息切回到编辑区域的最初位置。	<Enter>
切换到下一个打开的编辑窗口	<Ctrl>+<F6>
切换到先前打开的编辑窗口	<Ctrl>+<Shift>+<F6>
打开和关闭多层的变量	<Enter>
打开和关闭文件夹	<Enter>
转换在对象管理器或库管理器中的注册记录卡。	<Arrow keys>
在对话框中移到下一个区域	<Tab>
关于上下文的帮助	<F1>
一般的命令	
’文件’ ’保存’	<Ctrl>+<S>
’文件’ ’打印’	<Ctrl>+<P>
’文件’ ’退出’	<Alt>+<F4>
’工程’ ’检查’	<Ctrl>+<F11>
’工程’ ’生成’	<Shift>+<F11>
’工程’ ’全部重新生成’	<F11>
’工程’ ’删除对象’	<Del>
’工程’ ’增加对象’	<Ins>
’工程’ ’重命名对象’	<Spacebar>
’工程’ ’打开对象’	<Enter>
’编辑’ ’撤消’	<Ctrl>+<Z>
’编辑’ ’重复’	<Ctrl>+<Y>
’编辑’ ’剪切’	<Ctrl>+<X> or <Shift>+<Del>

’编辑’ ’复制’	<Ctrl>+<C>
’编辑’ ’粘贴’	<Ctrl>+<V>
’编辑’ ’删除’	<Del>
’编辑’ ’查找下一个’	<F3>
’编辑’ ’输入助手’	<F2>
’编辑’ ’自动声明’	<Shift>+<F2>
’编辑’ ’下一个错误’	<F4>
’编辑’ ’前一个错误’	<Shift>+<F4>
’联机’ ’登陆’	<Alt><F8>
’联机’ ’退出’	<Ctrl>+<F8>
’联机’ ’运行’	<F5>
’联机’ ’设置断点’	<F9>
’联机’ ’单步跳过’	<F10>
’联机’ ’单步进入’	<F8>
’联机’ ’单个循环’	<Ctrl>+<F5>
’联机’ ’设置新值’	<Ctrl>+<F7>
’联机’ ’强制新值’	<F7>
’联机’ ’解除强制’	<Shift>+<F7>
’联机’ ’’设置/强制对话框’	<Shift>+<F7>
’窗口’ ’信息’	<Shift>+<Esc>

FBD 编辑器命令

’插入’ ’网络 (插入在当前行后)’	<Shift>+<T>
’插入’ ’赋值’	<Ctrl>+<A>
’插入’ ’跳转’	<Ctrl>+<L>
’插入’ ’返回’	<Ctrl>+<R>
’插入’ ’操作符’	<Ctrl>+<O>
’插入’ ’功能’	<Ctrl>+<F>
’插入’ ’功能块’	<Ctrl>+<B>
’插入’ ’输入’	<Ctrl>+<I>
’附加’ ’取反’	<Ctrl>+<N>
’附加’ ’转换’	<Alt>+<Enter>

CFC 编辑器命令

’插入’ ’POU’	<Ctrl>+<B>
’插入’ ’输入’	<Ctrl>+<E>
’插入’ ’输出’	<Ctrl>+<A>

’插入’ ’跳转’	<Ctrl>+<G>
’插入’ ’标号’	<Ctrl>+<L>
’插入’ ’返回’	<Ctrl>+<R>
’插入’ ’注释’	<Ctrl>+<K>
’插入’ ’POU 输入’	<Ctrl>+<U>
’附加’ ’取反’	<Ctrl>+<N>
’附加’ ’置位/复位 ’	<Ctrl>+<T>
’附加’ ’联接标记’	<Ctrl>+<M>
’附加’ ’EN/ENO’	<Ctrl>+<E>
’附加’ ’转换’	<Alt>+<Enter>

LD 编辑器命令

’插入’ ’网络 (插入在当前行后)’	<Shift>+<T>
’插入’ ’常开接点’	<Ctrl>+<K>
’插入’ ’关联常开接点’	<Ctrl>+<R>
’插入’ ’功能块’	<Ctrl>+<B>
’插入’ ’线圈’	<Ctrl>+<L>
’附加’ ’粘贴在下面’	<Ctrl>+<U>
’附加’ ’取反’	<Ctrl>+<N>
’附加’ ’转换’	<Alt>+<Enter>

SFC 编辑器命令

’插入’ ’步－转换 (插入在当前行前)’	<Ctrl>+<T>
’插入’ ’步－转换 (插入在当前行后)’	<Ctrl>+<E>
’插入’ ’选择分支 (插入在右侧)’	<Ctrl>+<A>
’插入’ ’选择分支 (插入在左侧)’	<Ctrl>+<L>
’插入’ ’跳转’	<Ctrl>+<U>
’附加’ ’缩放动作/转换’	<Alt>+<Enter>
从 SFC 总图返回编辑器	<Enter>

关于 PLC- 任务配置的操作

打开和关闭结构成员	<Enter>
在名称附加放置一个编辑控制框	<Spacebar>
’附加’ ’登陆编辑’	<Enter>

关于参数管理编辑器的操作

连接导航窗口和列表编辑器	<F6>

在列表编辑框下删除一行	<Ctrl>+<Del> <Shift>+<Del>
在列表编辑框下删除一个区域	<Del>

CoDeSys中的关键字

下面的字符串保留为CoDeSys中的 关键字，它们不可以作为POUs或变量的标识符来使用:

ABS
ACOS
ACTION（仅在输出格式下应用）
ADD
ADR
ADRINST
AND
ANDN
ARRAY
ASIN
AT
ATAN
BITADR
BOOL
BY
BYTE
CAL
CALC
CALCN
CASE
CONSTANT
COS
DATE
DINT
DIV
DO
DT
DWORD
ELSE
ELSIF
END_ACTION（仅在输出格式下应用）
END_CASE
END_FOR
END_FUNCTION（仅在输出格式下应用）

END_FUNCTION_BLOCK（仅在输出格式下应用）
END_IF
END_PROGRAM（仅在输出格式下应用）
END_REPEAT
END_STRUCT
END_TYPE
END_VAR
END_WHILE
EQ
EXIT
EXP
EXPT
FALSE
FOR
FUNCTION
FUNCTION_BLOCK
GE
GT
IF
INDEXOF
INI
INT
JMP
JMPC
JMPCN
LD
LDN
LE
LINT
LN
LOG
LREAL
LT
LWORD
MAX
MIN
MOD
MOVE
MUL
MUX
NE
NOT
OF
OR

ORN
PERSISTENT
POINTER
PROGRAM
R
READ_ONLY
READ_WRITE
REAL
REPEAT
RET
RETAIN
RETC
RETCN
RETURN
ROL
ROR
S
SEL
SHL
SHR
SIN
SINT
SIZEOF
SQRT
ST
STN
STRING
STRUCT
SUB
TAN
THEN
TIME
TO
TOD
TRUE
TRUNC
TYPE
UDINT
UINT
ULINT
UNTIL
USINT
VAR
VAR_ACCESS（仅在非常特定的情况下应用，取决于硬件）

VAR_CONFIG
VAR_CONSTANT
VAR_EXTERNAL
VAR_GLOBAL
VAR_IN_OUT
VAR_INPUT
VAR_OUTPUT
WHILE
WORD
WSTRING (IEC 数据类型, 在 CoDeSys 下不支持)
XOR
XORN
另外，输入助手列出的所有转换操作符都被认为是关键字。

CoDeSys 文件

CoDeSys 可以创建下列类型的文件:

文件扩展名	例子	描述	格式	路径(默认)
*.pro	project01.pro	CoDeSys 工程文件	二进制	工程所在的路径
*.ci	project01<number>.ci	工程最后一次生成（编译）信息->增加编译可能；只有在工程保存时创建。 编码：目标-ID 编码	二进制	工程所在的路径
*.eci	project01<number>.eci	外部编译信息；eci 格式下 ci 文件的子集；通过访问动态链接库来读取。 编码：目标-ID 编码	PE	工程所在的路径
*.cic	project01<number>.cic	工程从目标的信息最后一次编译（编辑） -> 增加编译可能；只有在工程保存时创建。 编码：目标-ID 编码	二进制	工程所在的路径
*.cit	project01<number>.cit	临时的 *.ci-文件；在目标转换时创建，在工程下一次保存时转换到 ci-文件 编码：目标-ID 编码	二进制	工程所在的路径
*.ri	project01<number>.ri	最后一次的下载信息，对于联机改变非常重要；在每次下载时创建。 编码：目标-ID 编码	二进制	工程所在的路径
*.exp	project01.exp, PLC_PRG.exp	输出文件	输出格	工程所在的路径

		('工程' '输出')	式 （文本)	
*.tlt *.txt	project01.tlt project01.txt	转化文件(在'工程' '转化成另一种语言'下定义)	文本	
*.sym	project01.sym	符号文件	文本	工程所在的路径
*.sdb	project01.sdb	符号文件	二进制	工程所在的路经
*.sym_xml	project01.sym_xml	符号文件	XML	工程所在的路径
*.asd	project01.asd	保存文件 (临时的, '自动保存', '编译前自动保存')	二进制	工程所在的路径
*.asl	lib01.asl	像 CoDeSys 下的工程一样保存一个打开的库(临时的, '自动保存', '编译前自动保存')	二 进制	库或工程所在的路径
*.bak	project01.bak	为工程做备份文件(永久的, '创建备份')	二进制	工程所在的路径
*.prg *.bin	default.prg project01.prg	创建引导工程, 文件名称由目标系统决定	二进制	目标系统 (联机模式下创建) 工程所在的路径(模拟模式下创建)
*.chk	default.chk project01.chk	检查引导工程的代码	二进制	目标系统 (联机模式下创建) 工程所在的路径(模拟模式下创建)
*.ini	codesys.ini	不同 CoDeSys 设置的初始化文件。	文本	同 codesys.exe
*.dfr	default.dfr project01.dfr	结构文件 (打印机设置)	二进制	同 codesys.exe
*.asm	code386.asm	创建工程代码时的汇编列表。	文本	编译所在路径
*.lst	project01.lst	创建工程代码时的汇编列表。	文本	编译所在路径
*.bpl	project01.bpl	调试文件 (断点信息)	文本	编译所在路径
*.st	PLC_PRG.st	调试文件, 固有的 ST-代码	文本	编译所在路径
*.map	project01.map	地图文件; 存储器机构和变量位置信息	文本	编译所在路径
*.hex *.h86	project01.hex (Output) resp. standard.hex (Lib)	英特尔或摩托罗拉的十六进制, 英特尔的 h86; 编译外部库的输出或输入。	英特尔或摩托罗拉的十六进制文件	编译所在路径 或库所在路径
*.trd	projectxy0.trd	趋向日志 (如果文件已满而且	文本	工程所在的路径

		另一个文件必须被创建，园点前的编号加一。）		
*.log	projectxy.log	日志文件（日志）	二进制	工程所在的路径
*.wtc	projx_watch1.wtc	监控列表（监视和解决方法管理器）	文本	用户定义路径
*.alm	alarmlog0.alm	报警日志文件		用户定义路径或控制器的下载路径
*.zip	projectxy.zip	工程存档文件；属于工程的压缩文件，’文件’’保存/邮件存档’		用户定义路径
*.trc	project01_tr1.trc	记录描述	二进制	同 codesys.exe
*.mon	project01_tr1.mon	记录描述	XML	同 codesys.exe
*.tcf	project01_tr1.tcf	记录描述	二进制	同 codesys.exe

附录K 关于编译错误和警告

10.31 警告

如果在工程 编译过程中出现错误，将在 信息窗口出现提示。警告也同时显示出来。在此窗口按 <F4>跳到消息的下一行，与此相关的POU将会打开。在错误和警告之前有一个唯一的编号。在消息窗口选择了一个消息行，按 <F1>打开相应的在线帮助窗口。

1100

未知库中的功能名‘<name>’

使用一个外部库。请检查在 hex 文件中定义的所用功能是否也定义在 lib 文件内。

1101

″ 不能识别的符号‘<symbol>’″

代码生成器期望有一个名称为<symbol>的 POU。它没有在目标项目中定义。并使用这个名称定义了一个功能/程序。

1102

″对符号‘<symbol>’的无效接口″

代码发生器期望有一个名为<symbol>的功能和恰好是一个标量的输入，或一个名为<symbol>和无输入或输出的程序。

1103

″在代码地址‘<address>’的常数‘<name>’改写一个 16K 页的边界!″

超过 16K 页边界的一串常数。系统不能处理这种边界。这取决于运行时的系统，通过目标文件中的一个登录项是否能解决这个问题。请与 PLC 制造商联系。

1200

″在任务调用的名为’<name>’的 POU 中的存取变量内容在参数列表中未获更新″

在任务配置中调用的任务块中所使用的变量将不会列在交叉参考列表中。

1300

"文件找不到'<name>'名"

指示全局变量的文件不存在。请检查其路径是否正确。

1301

"分析库未找到。分析代码无法生成"

系统使用了分析功能，但相应的库.lib 没找到。将它加入到库管理程序内。

1302

"新添加了外部引用功能，因此不能使用在线变更功能"

由于最后装载，你链接的一个库它所包含的功能尚未在运行时系统中引用。为此，你应装载整个项目。

1400

"未知的编译指令'<name>'，编译器忽略"

这个附注不受编译程序支持。受支持的命令见关键字的附注。

1401

"名为'<name>'的结构未定义任何元素"

此结构未定义任何元素，但是系统仍然会分配 1B 的内存容量。

1410

"功能中定义本地变量时'RETAIN'和'PERSISTENT'关键字无效"

系统将按照一般内部变量处理。

1411

"变量配置中定义的变量'<name>'未在任何任务调用中得到更新"

没有在任何调用中引用变量的高水平的例子。因此其不能在过程镜象中被拷贝。

例如:

变量配置:

```
VAR_CONFIG
 plc_prg.aprg.ainst.in AT %IB0 : INT;
END_VAR
```

plc_prg:

```
index := INDEXOF(aprg);
```

程序 aprg 已经引用但是没有调用. 因此 plc_prg.aprg.ainst.in 永远得不到 %IB0 的实际值。

1412

"编译指令{pragma name}出现未知代号"

用户使用的编译指令包含有在该场合不能正确使用的指令。详情请参看 CoDeSys 在线帮助。

1413

"'<Name>'在名字列表中的索引非法，此索引将被忽略"

在编译指令中使用了不存在的参数列表。请参看参数管理器获取当前列表。

1414

名为'<name>'的编译指令定义太多组成部分

编译指令包含的定义（指括号里面的内容）超过了相应数组、功能块或者结构所能容纳的元素个数。

1500

"表达式未包含赋值。此语句无代码产生"

程序未使用该表达式的运算结果，因此该语句无代码产生。

1501

"字符串常量的内容被改写"

在 POU 中定义的字符串常量的内容不允许更改，因为系统忽略其大小检查。

1502

"POU 与其定义的某个变量重名。导致 POU 不能被正确调用"

POU 与其定义的某个变量重名。

例如:

```
PROGRAM a
...
VAR_GLOBAL
a: INT;
END_VAR
...

a; (* 没有 POU a 被调用，但是变量已经加载 *)
```

1503

"名为'<name>'的 POU 无输出，系统设定其输出为'TRUE'"

连接 FBD 或 KOP 的 POU 无输出。 系统自动将其设定为 TRUE。

1504

"?lt;name>?(?lt;number>?: 逻辑表达式的此分支将不会被执行，因为整个表达式的值已可确定"

逻辑表达式的此分支将不会被执行，因为整个表达式的值已可确定

例如:

```
IF a AND funct(TRUE) THEN ....
```

如果 a 是 FALSE 则 funct 不被调用。

1505

"'<name>'分支有歧义，可能不会被执行"

POU 的第一个输入为 FALSE，由于这个原因旁边的第二个输入将不被执行。

1506

"变量与某个本地动作同名，将导致该动作不能被正确调用"

重命名变量或行为。

1507

"创建的实例与某个功能名同名，将导致该实例不能被正确调用"

在 ST 编程语言中，如果调用与某功能同名的实例，系统将执行同名功能而不是实例。

1550

"名为'<name>'的 POU 在一个执行网络中被多重调用，可能导致不可预料的结果"

检查，对 POU 的多重调用是否是必要的。通过多重的调用可能导致不可预料的后果。

1600

"不明确打开 DB （可能产生错误的代码)."

没有表明 POU 打开的最初西门子程序。

1700

"输入框未联接上"

在 CFC 中有输入框未连接上，将不能产生执行代码。

1750

"步 '<名称>': 设定的时间下限超过时间上限"
请使用该步的' 步属性' 对话框进行时间的正确设定。

1751

"'<Name>'变量的使用警告：该变量使用于固定代码中并且影响步的执行顺序"
推荐对该变量重命名以使其具有唯一的标识符，从而避免负面影响。

1800

"非法的监控表达式'<Name>'"
可视化元素中包含了无法监控的表达式，检查变量名称和占位符的替换是否无误。

1801

"表达式不能作为输入变量使用"
可视化组件的输入框使用了组合表达式，请使用单个变量替换之。

1802

"名为'<Name>'的位图文件未找到"

1803

"网页可视化和目标平台可视化不支持打印操作"

1804

"目标平台不支持'<name>'型字体"

1805

"要使用可视化组件实现 PLC 趋势数据的存储应先进行相关的设定"

1806

"应对'目标设定'中的'PLC 报警处理'选项进行设定"

1807

"<name> (<number>): 针对可视目标没有警告消息窗口"

1850

"映射在%IB<number>位置的输入变量在任务'<name>'中使用，但在另外任务中更新"

1851

"映射在%IQ<number>位置的输出入变量在任务'<name>'中使用，但在另外任务中更新"

1852

"CanOpenMaster 功能块在任务'<name>'事件中未能实现循环调用！若要实现其循环调用，请使用'模块参数'对话框正确设置运行参数并更新任务!"

1853

"PDO(index: '<number>')可能在任务'<name>'事件中未能实现周期性更新"

1900

"POU'<name>'（主程序）在库中找不到；当工程作为一个程序库使用时，主入口程序不可用" 当项目作为库使用时，不能利用“启动 POU”

1901

"入口变量和变量配置表未存入相应库文件中!"
存取变量和变量配置没有保存在库内。

1902

"'<name>'库不支持当前硬件平台类型!"

库的对象文件是为其他设备而生成的。

1903

"'<name>'库是非法库；该库文件不符合实际目标平台要求的格式"

文件没有实际目标所需要的格式。

1904

" '<name>'常量覆盖了链接库的一个同名常量"

如果用户在工程中定义了与链接库同名的常量，库中该常量将被覆盖。

1970

"参数管理器：'<Name>'列，'<Name>'行的'<Name>'值不能被引入!"

请检查数据引入文件*.prm 的格式是否符合当前参数管理器的设置。

1980

"对全局变量'<Name>'同时读、写可能导致数据丢失"

1990

" '<name>'参数未能在参数设置中获的可用的内存地址映射"

请在资源选项卡中的 参数设置窗口对其进行正确设置。

2500

"任务 '<任务名称>': 对循环任务没有指定循环时间"

10.32 错误

3100

"用户程序代码太大。系统支持代码最大容量为'<number>'字节(<number>K)"

超过最大程序规模。减小项目规模。

3101

"用户程序使用数据太多，最大容量为'<number>'字节(<number>K)"

超出内存容量范围。请减少应用程序的数据使用量。

3110

"'<Name>'库出现错误"

该库文件*.hex 不是 INTEL 16 进制格式。

3111

" '<Name>'库太大。系统支持最大容量 64K"

.hex 文件超过设定的最大容量。

3112

"库中存在不可重复定位指令"

该库代码无法进行链接。

3113

"库代码覆盖功能表格."

代码和功能表的范围重叠。

3114

″库使用了多个内存段.″

.hex 文件中的表和代码利用不止一个字段。

3115

″不能把常量赋值给 VAR_IN_OUT 型变量，数据类型不匹配.″

由于数据设置在“near”(近的),但是串常数设置在“huge”（大的）或“far”（远的），因而串常数的内部指针格式不能转换成 VAR_IN_OUT 的内部指针格式。如有可能，改变这些目标设置。

3116

″功能表格覆盖库代码或段分界线。″

3117

″<Name> (<Zahl>):表达式过于复杂，无法用寄存器装载处理″

3120

″当前代码段超出 64K.″

当前生成的代码大于 64K。最终建立太多的初始化代码。

3121

″POU 太大.″

POU 的大小应限制在 64K 以下。

3122

″功能或结构的初始化代码太大，超出 64K 上限″

用于一个功能或一个结构化 POU 的初始化代码不应超过 64K。

3123

″数据段太大：段'<Number>%s', 大小<size> 字节(最大为 <number>字节)″

3124

″字符串常量太大: '<number>' 字节 (最大 253 字节)″

所给出的常量必须缩短字符数

3130

″用户堆栈太小：需要'<number>' 双字容量的堆栈，只能提供'<number>'双字容量的堆栈.″

POU 调用的嵌套深度太大。在目标设置中输入一个更大的堆栈尺寸，或编译没有任选项“debug”(可在对话框“project”(项目)“(options)”任选项“build”(建立)中设置)的建立项目。

3131

″用户堆栈太小：需要'<number>' 双字容量的堆栈，只能提供'<number>'双字容量的堆栈.″

请联系 PLC 制造商。

3132

″系统堆栈太小：需要'<number>' 双字容量的堆栈，只能提供'<number>'双字容量的堆栈.″

请联系 PLC 制造商。

3150

″功能 '<名称>'的参数 <编号>: 不能作为 C－函数的一个字符串参数来传递 IEC－功能的结果.″

使用一个中间变量，将 IEC 功能的结果赋值给该中间变量。

3160

″不能打开'<name>'库文件.″

工程通过库管理器包含了'<name>'库，但是在指定的路径未能找到该库文件。

3161

″库文件'<name>'缺少代码段″

一个程序库的一个.obj 文件至少应包含一个 C 功能。将一个亚功能插入到.lib 文件内，在.lib 文件内尚未定义这个亚功能。

3162

″无法解决库 '<名字>'涉及到的(符号 '<名称>', 类'<名称>', 类型'<名称>')″

.obj 文件包含一个对其他符号不能分辨的引用。请检查 C 编译程序的设置。

3163

″'<name>'库包含未知的引用类型″

.obj 文件包含一个引用类型，它不能被代码发生器所分辨。请检查 C 编译程序的设置。

3200

″'<name>'：布尔表达式过于复杂，请使用中间变量简化之″

目标系统的暂时存储器容量对表达式的规模来说显得不够。将表达式分成几个部分表达式，为此可使用对几个中间变量进行赋值。

3201

″<名称> (<网络>): 一个网络必须不能导致代码超过 512 字节。″

不能分辨内部跳转。启动选项“利用 16 位跳转补偿”中的 68K 目标设置。

3202

″功能嵌套调用导致堆栈溢出″

使用一嵌套的功能调用 CONCAT(x,f(i))。这会导致数据丢失。将调用分成两个表达式。

3203

″表达式过于复杂（无法使用寄存器进行处理）.″

将赋值分成几个表达式。

3204

″程序跳转超出范围 32k 字节″

跳转范围应不超过 32767 字节。

3205

″内部错误：包含太多常量字符串″

在一个 POU 中，最多能使用 3000 个常量字符串。

3206

″功能块程序代码溢出″

功能块程序代码上限为 32Kb。

3207

″数组优化失败″

计算索引时功能被调用导致最优化数组存取失败。

3208

″转化未成功″

使用的转换功能，不是为实际代码发生器而实现的。

3209

″操作未成功″

使用的操作符，不是为这种数据类型和实际代码发生器而实现的。MIN(字符串 1，字符串 2)

3210

"'<Name>'功能未找到"
调用的功能不能用于项目内。

3211

"字符串使用超出范围"
字符串变量一个表达式中可最多使用 10 次。

3212

"在 POU <POU 名称>错误的库命令 "

3250

"8 位控制器不支持 REAL 类型"
目标当前未得到支持。

3251

" 8 位控制器不支持日期类型。"
目标当前未得到支持。

3252

"堆栈溢出<number>字节"
当前目标系统不支持。

3253

"找不到此 hex 文件：'<Name>'"
目标当前未得到支持。

3254

"调用的外部库功能不能被解析."
目标当前未得到支持。

3255

"8 位控制器不支持指针."
运行于 8 位系统上的程序应避免使用指针.

3260

"功能'<名称>' 拥有太多的自变量: 增加目标系统自变量堆的大小."

3400

"读取输入变量时发生错误"
.exp 文件包含一个正确的存取变量段。

3401

"装载变量配置时发生错误"
.exp 文件包含一个正确的配置变量段。

3402

"装载全局变量时发生错误"
.exp 文件包含一个正确的全局变量段。

3403

"目标<name>不能被装载"
.exp 文件中的对象<name>段不正确。

3404

"装载任务配置时发生错误"

.exp 文件中的任务配置段不正确。

3405

"装载 PLC 配置时发生错误"

.exp 文件中的 PLC 配置段不正确。

3406

"SFC 程序组织单元存在同名步"。第二步将不能载入。"

.exp 文件中的 SFC POU 段包含有相同名的两个步。在输出文件中重新命名其中的一步。

3407

"当前步的上一步未找到"

.exp 文件中丢失步<name>。

3408

"当前步的后续步未找到"

.exp 文件中丢失步<name>。

3409

" '<' name>'步转换条件不正确 "

.exp 文件中，丢失转换名，它需要步<name>作为先行步。

3410

" 对应转换条件'<name>'的步不正确"

.exp 文件中，丢失一个步，它需要转换名<name>作为先行条件。

3411

" '<name>'步与初始步的联系丢失"

.exp 文件中，丢失步<name>与初始步之间的连接。

3412

"宏'<name>'不能被载入"

3413

"CAM 文件载入时发生错误."

3414

"CNC 文件载入时发生错误"

3415

"报警配置载入时发生错误"

3450

"'<PDO-name>':COB-ID 丢失"

为此模块，点出 PLC 配置对话框中的“properties”（属性），并输入用于 PDD<PDD 名>的一个 COB ID。

3451

"EDS 文件载入错误：硬件配置引用了此文件，但是找不到该文件"

CAN 配置所需的设备文件不在正确的目录内。检查“project”（项目）“options”（选项）“目录”中的配置文件的目录设置。

3452

"'<Name>'模块不能被创建!"

<Name>'模块的设备文件不兼容当前的设置，应根据 CoDeSys 的相关设置对其进行修改。

3453

"'<Name>'通道不能被创建!"

'<Name>'通道的设备文件不兼容当前的设置，应根据 CoDeSys 的相关设置对其进行修改。

3454

"'<name>'地址指向的内存已使用!"

在对话框 PLC 配置的“setting”（设置）中，已启动选项“check for overlapping addresses”（检查重叠地址），并已检测到一个重叠。注意，区域检查是基本尺寸，它与模块的数据类型有关。在配置文件中，通过登录项“size”（尺寸），给出区域尺寸。

3455

"载入 GSD 文件出错：硬件配置中要引用此文件，但是找不到该文件!"

Profibu 配置所需的设备文件不在正确的目录内。检查“project”（项目）“options”(选项)“目录”中的配置文件的目录设置。

3456

"总线设备驱动'<name>'无法被创建!"

用于模块<name>的设备文件不适应于当前的配置。自建立配置以来已作了修改，或者它已损坏。

3457

"模块解析错误!"

请检查这个模块的数据文件。

3458

"PLC 配置无法完成，请检查配置文件."

3459

"不支持所选波特率."

3460

3S_CanDrv.lib 库版本号错误。

3461

"3S_CanOpenMaster.lib 库版本号错误."

3462

"3S_CanOpenDevice.lib 库版本号错误."

3463

"3S_CanOpenManager.lib 库版本号错误."

3464

"3S_CanOpenNetVar.lib 库版本号错误."

3465

"CAN 通讯从模块的副索引应连续标号"

3466

"CAN 总线变量：PLC 配置中未发现 CAN 通讯控制器"

3468

"CAN 总线驱动：任务配置中未实现任务更新."

3469

"CANOpen 通讯主模块不能被调用，请手动分配其任务."

3470

″任务更新常数名称非法″

3500

″'<name>'变量未在变量列表中声明″

将用于变量的说明插入到全局变量表内，该表包含“variable_configuration”（变量—配置）。

3501

″在变量列表中声明的'<name>'变量未分配相应的内存地址.″

将这个变量的一个地址分配给全局变量表，该表包含“variable_configuration”（变量—配置）。

3502

″变量配置列表中的变量'<name>'数据类型声明有误″

在包含“variable_configuration”（变量—配置）。的全局变量表中，变量以不同于 POU 中的数据类型说明。

3503

″变量配置列表中的变量'<name>'数据类型有误″

在包含“variable_configuration”（变量—配置）。的全局变量表中，变量以不同于 POU 中的数据类型说明。

3504

″变量配置中不支持初始赋值″

“variable_configuration”（变量—配置）的一个变量是以地址和初始值说明的。但是，一个初始值只能为没有地址分配的输入变量而定义的。

3505

″'<name>'路径不是合法的实例路径名 ″

“variable_configuration”（变量—配置）包括一个存在的变量。

3506

″存取路径未指定″

在“access variables” （存取变量）的全局变量表中，一个变量的存取路径是不正确的。正确的应该是:《标示符》:《存取路径》:《类型》《存取方式》

3507

″存取变量对应的内存使用不合法″

3550

″标识符'<name>'重复定义″

使用一个相同的名来定义两个任务。重新命名其中的一个。

3551

″任务'<name>'中至少应包含一个程序调用″

插入一个程序调用或删除任务。

3552

″任务'<name>'使用的事件变量'<name>'未定义″

在任务属性的“single”（单）区域对话框中设置的一个事件变量，在项目未经全局说明。利用其他变量或全局定义该变量。

3553

″任务'<name>'中的事件变量'<name>'必须是布尔类型变量″

在任务属性的“single”（单）区域对话框中，使用一个类型 BOOL 的变量作为事件变量。

3554

"任务入口 POU 必须是程序或者全局功能块实例"

在该字段中，输入“program call”（程序调用）一个功能或一个不定义的 POU。输入一个有效的程序名。

3555

"任务入口 POU 包含非法参数"

在该字段中，所用的”append program call”（附加程序）参数不符合程序 POU 的说明。

3556

"当前目标系统不支持此任务"

3557

"任务数超出最大范围"

3558

"任务'<name>'的优先权超出合法范围>'"

3559

"任务'<name>'：当前目标系统不支持任务间隔调用模式"

3560

"任务'<name>'：当前目标系统不支持任务 freewheeling 调用模式"

3561

"任务'<name>'：当前目标系统不支持任务触发调用模式"

3562

"任务'<name>'：当前目标系统不支持任务外部触发调用模式"

3563

"任务'<name>'的调用间隔设置超出合法范围"

3564

" 当前目标系统不支持任务'<name>'所设置的外部触发事件'<name>'"

3565

"定义的事件触发任务数超出最大范围"

3566

"定义的间隔运行任务数超出最大范围"

3567

"定义的 freewheeling 任务数超出最大范围"

3568

"定义的外部事件触发任务数超出最大范围"

3569

"设定为系统事件触发的 POU 未定义"

3570

"多个任务设定的优先权相同"

3571

"工程未包含 SysLibCallback.lib 库，不能生成系统事件."

3572

″任务'<name>'的看门狗时间间隔设定超出合法范围″

3573

″看门狗时间设定超出合法范围″

3574

″事件变量对应的内存地址不能在事件中重复使用″

3575

″任务'<名称>': 循环时间应该是 '<数字>'的整倍数 .″

3600

″隐含变量未找到!″

使用命令”rebuild all”（重建所有)。如依然得到出错消息，请于 PLC 制造商联系。

3601

″'<name>'变量名称是系统保留用名，请更改″

给出的变量在项目中说明，虽然它是保持用于代码发生器。重新命名该变量。

3610

″ 不支持'<name>'″

给出的属性不被编译系统的当前版本支持。

3611

″提供的编译目录名'<name>'非法″

用于编译文件的“project” （项目）—“options” （选项）—“directories”（目录）中给出的目录是无效的。

3612

″超出系统所能处理最大 POU 数量，编译被中止.″

在项目中所用的 POU 和数据类型太多。在“target setting/memory layout”（目标设置/存储器布局）中，修改 POU 的最大数。

3613

″工程编译取消″

编译过程被用户删除。

3614

″工程必须包含一个名为 PLC_PRG 的主程序或者进行任务配置″

建立一个类型”program”（程序）的初始 POU，或建立一个任务配置。

3615

″<名称> (主要的程序) 必须有程序类型″

在项目中所用的一个初始 POU 不是” program”（程序）类型。

3616

″程序不能在外部库中执行″

应作为外部库保存的项目包含一个程序。当使用该库时，不能使用这个程序。

3617

"内存不足"

增加你的计算机的虚拟存储器的容量。

3618

"当前代码生成器不支持位存取!"

当前设置的目标系统的代码发生器，不支持对变量的位存取。

3619

"库文件'<name>'与其对应的目标文件版本不合!"

3620

"PLC_PRG 不能出现在库文件中"

3621

"无法对编译文件'<name>'进行写操作"

3622

"无法创建符号文件'<name>'"

3623

"无法对引导工程文件'<name>'进行写操作"

3624

"目标设置 <目标设置 1>=<设置值> 与 <目标设置 2>=<设置值> 不兼容"

在目标设置对话框内检查并纠正这些设置(资源标签)。如果这些设置不可见且不可在那里编辑，请联系 PLC 制造商。

3700

" 工程中存在与库中同名的 POU"

一个 POU 名已用于项目内，这一名早已用于一个库 POU。重新命名 POU。

3701

" 目标名不能与 POU 同名"

使用命令“project”（项目）--“rename object”(重新命名对象)，重新命名“object organizer（对象组织程序）”中的 POU 或在说明窗内改变 POU 的名。在那里，POU 名必须放置在临近的关键字 PROGRAM，FUNCTION 或 FUNCTIONBLOCK 中的一个。

3702

"标识符声明过量"

最大为 100 个标示符可输入到一个变量说明内。

3703

"标识符'<name>'重复定义 "

要注意，在 POU 的说明部分中，只允许一个标示符有给定的名。

3704

"数据存在递归调用"

3705

"PLC_PRG 中不允许使用 VAR_IN_OUT 型变量"

3706

″标识符 ’常量’ 只允许在 ’VAR’, ’VAR_INPUT’, ’VAR_EXTERNAL’ 和’VAR_GLOBAL’ 中应用。″

3720

″AT 关键字应紧接内存地址″

在关键字 AT 后面加一个有效地址，或修改关键字。

3721

″只有变量以及全局变量能定位于某个内存地址″

将说明放置在一个 VAR 或 VAR-GLOBAL 说明区。

3722

″只有布尔型变量允许使用位地址″

修改地址或修改分配地址的变量类型。

3726

″常量不能存储于指定的内存中″

这种类型的变量不能放置在给定的地址上。

3727

″该地址不允许定义数组″

3728

″非法的内存地址″

3729

″内存地址’<name>’中存储的数据类型非法 ″

3740

″非法类型 ″

在一个变量说明中，使用无效的数据类型。

3741

″非法的数据类型″

使用一个关键字或一个操作符以代替一个有效类型标示符。

3742

″应指定枚举值″

在枚举类型定义中，在打开的括号后，或在括号之间的逗号后面，丢失一个标示符。

3743

″枚举应使用整型数字进行初始化″

枚举只能以类型 INT 的数初始化。

3744

″枚举常量名已被定义过″

检查一下，你是否遵循了以下的枚举值定义规则：

.在一个枚举定义中，所有的值都应是唯一的。

.在所有的全局枚举定义中，所有的值均应是唯一的。

.在所有的局部枚举定义中，所有的值均应是唯一的。

3745

″子区域中只允许使用整型数据类型!″

子范围类型只能在整数数据类型上定义。

3746

″子区域'<name>'与基本数据类型不兼容″

子范围类型的范围极限设定中，有一个极限设定超出其基本类型的有效范围

3747

″字符串'<name>'长度未知″

没有一个有效的常数用于串长的定义。

3748

″最多只能定义 3 维数组″

在一个数据的定义中，给出允许的维数即大于三维。若可应用的话，使用“ARRAY OF ARRAY”（数组的数组）。

3749

″下限'<name>'未定义″

使用一个为定义的常数来定义一个子范围或数组类型的下限。

3750

″上限'<name>'未定义″

使用一个为定义的常数来定义一个子范围或数组类型的上限。

3751

″字符串长度非法″

3752

″数组嵌套使用最大不能超过 9 维″

3760

″初始值错误″

使用一个与类型定义相当的初始值。为了更改说明，你可使用变量的说明对话框。(SHIFT F2 或 EDIT(编辑）“autodeclare”(自动说明))。

3761

″ VAR_IN_OUT 型变量不允许赋初值.″

在 VAR_IN_OUT 变量的说明部分，去除初始化。

3780

″期待'VAR', 'VAR_INPUT', 'VAR_OUTPUT' 或 'VAR_IN_OUT'等类型变量″

POU 名后的第一行必须包含这些关键字中的一个关键字。

3781

″期待 END_VAR 标识符″

在说明窗内的给出行的起始位置输入一个 END—VAR 的有效标示符。

3782

″意外的结束″

在说明性编辑器中：在说明部分结束处加上关键字 END—VAR。在编译部分的文本编译器中：加上一个指令，它中止最后的指令顺序。

3783

″期待 END_STRUCT 标识符″

应确实弄清楚，类型说明已正确的终止。

3784

″当前目标系统不支持设定的此属性″

3800

″全局变量占用大量内存，请在工程选项增加内存使用量.″

增加在对话框”project” （项目）--“options” （选项）--“build”（建立）的设定中所给出的程序段数。

3801

″'<name>'变量太大″

变量使用大于 1 个数据段的类型。段尺寸是一个目标特定的参数并可在目标设定/存储器布局中修改。如果你在当前的目标设定中找不到它，请与你的 PLC 制造商联系。

3802

″保留内存溢出.″

可供保变量用的内存空间已耗尽。在目标设定存储器布局中可设定目标专用的内保存区域尺寸。如果你在对话框中找不到设定字段，请与你的 PLC 制造商联系。

3803

″全局数据存储内存溢出.″

可供全局变量用的内存空间已耗尽，在目标设定/存储器布局中可设定目标专用的内保存区域尺寸。如果你在对话框中找不到设定的字段。请与你的 PLC 供应商联系。

3820

″功能中不允许使用 VAR_OUTPUT 和 VAR_IN_OUT 型变量″

在一个功能中，可以定义无输出或输入-输出变量。

3821

″功能至少应有一个输入接口″

为这种功能至少附加一个输入参数。

3840

″未知的全局变量'<name>'!″

在 POU 中，使用一个 VAR_EXTERNAL 变量，对它未说明其全局变量。

3841

″'<name>'声明与全局声明不匹配!″

在 VAR_EXTERNAL 变量说明中给出的类型与全局说明中的类型不匹配。

3850

″在包装的'<名称>'结构体内部对未包装'<名称>'的结构体的声明是不允许的!″

3900

″标识符的多重强调″

在标示符名下除去多重下划线。

3901

″地址表达式至多允许包含 4 个数字段″

有一种表达式允许直接赋值给一个地址，它有不止四层的数值域。

3902

″关键字必须大写″

对关键字应使用大写字母，或启动选项，“project” （项目）—“options”--（选项）中的“autoformat”（自动格式化）。

3903

"非法的 duration 型常量"

常数的计数法不符合 IEC61131-3 格式。

3904

"duration 型常量溢出"

用于时间常数值不能以内部格式表示。可表示的最大值为 t#49d17h2m47s295ms

3905

"非法的日期常量"

常数的计数法不符合 IEC61131-3 格式。

3906

"非法的时间常量"

常数的计数法不符合 IEC61131-3 格式。

3907

"非法的日期和时间常量"

常数的计数法不符合 IEC61131-3 格式。

3908

"非法的字符串常量"

串常数包含一个无效的字符。

4000

"期望标识符"

在这个位置输入一个有效的标示符。

4001

"没有声明变量'<名称>'"

说明变量是局部或全局。

4010

"类型搭配错误：不能转换'<名称>'为'<名称>'."

检查一下，操作符期望的是何种数据类型，并改变引起错误的变量类型，或选择其他变量。

4011

"在'<名称>'中参数类型搭配错误：不能转换'<名称>'为'<名称>'."

实际参数的数据类型不能自动转换成格式化参数的类型。使用某种类型转换方式或使用其他变量类型。

4012

"在'<名称>'中参数类型搭配错误：不能转换'<名称>'为'<名称>'."

将一个无效类型《类型 2》变量分配给输入变量'<name>'。将变量或常数以一个类型《类型 1》变量代之，或采用一种类型转换。相应的，一个带有类型前缀的常数。

4013

"在'<名称>'中输出类型搭配错误：不能转换'<名称>'为'<名称>'."

将一个无效类型《类型 2》变量分配给输入变量'<name>'。将变量或常数以一个类型《类型 1》变量代之，或采用一种类型转换。相应的，一个带有类型前缀的常数。

4014

"无夸张类型：不能转换'<名称>'为'<名称>'"

常数类型与前缀类型不兼容。

4015

″对于直接的位存储数据类型’<名称>’不合法″

直接位编址只允许用于整数和“位串”数据类型。你在位存取<var1>.<bit>中，使用 RIAL/LREAL 类型的变量 var1,或一个常数。

4016

″对于’<名称>’类型的变量的位索引’<编号>’超出范围″

你正式试图存取的一位，它不是为变量的数据类型而规定的。

4017

″对于’REAL’，’MOD’没有定义″

操作符 MOD 只可用于整数和位串数据类型。

4020

″具有写权的变量或直接地址需要’ST’,’STN’,’S’,’R’″

使用一个具有写存取的变量取代第一个操作数。

4021

″变量’<名称>’不具有写权″

使用一个具有写存取的变量取代该变量。

4022

″期望操作数″

在注释后面加上一个操作数。

4023

″在’＋’或’－’后需要数字″

输入一个数字。

4024

″在’<名称>’前需要<操作数 0>或<操作数 1>’或……″

在命名的位置处，输入一个有效的操作数。

4025

″在’<名称>’前需要’:=’或’=>’″

在命名的位置处，输入二种操作符中的一种。

4026

″‘BITADR’需要一个位地址或关于位地址的一个变量″

使用一个有效的位地址。

4027

″需要一个整数符号或常数符号″

输入一个整数或一个有效常数的标示符。

4028

″‘INI’操作数需要功能块实例或数据单元类型实例″

检查 INI 操作符所使用的变量的数据类型。

4029

″不允许对同一个功能进行嵌套.″

在不是可重入的目标系统和在仿方式时，一个功能调用并不包含作为参数的自身调用。

4030

″‘ADR’操作数不允许用表达式或常量″

使用一个变量或一个直接地址取代常数或表达式。

4031

″位操作不允许用’ADR’！请用‘BITADR’来代替.″

使用 BITADR。请注：BITADR 功能不返回一个物理的内存地址。

4032

″对于’<名称>’来说，’<数字>’操作数太少。至少需要’<数字>’个″

检查一下，命名的操作符需要多少操作数，并添加缺少的操作数。

4033

″对于’<名称>’来说，’<数字>’操作数太多。至少需要’<数字>’个″

检查一下，命名的操作符需要多少操作数，并除去多余的操作数。

4034

″用 0 来除″

你正在一个常数表达式中以 0 作为除数。如果你要引起一个运行时的错误，使用—如可以应用的话—一个有值为 0 的变量。

4035

″如果’代替常量’被激活，’VAR’必须在’VAR CONSTANT’中应用″

使用直接值的一个常数地址存取是不可能的。如可应用的话，取消”project”（项目）---“options”（选项）---“build”（建立）

4040

″标签’<名称>’没有定义″

使用名（标记名）定义一个标记，或将名（标记名）改变成一个定义的标记名。

4041

″标签’<名称>’进行了多重定义″

在 POU 中，标记<name>是重复定义的。重新命名标记或从重复定义中除去一个名。

4042

″在序列中没有比’<数字>’更多的标签可以被允许″

跳转标记数限制在“<Anzahl>”。插入一个哑指令。

4043

″标签格式错误。一个标签在冒号后必须定义一个随意的名字.″

标记名是无效的，或在表达式中丢失了冒号。4050

4050

″没有定义 POU’%S’″

利用命令”project”（项目）--“Add Object”（附加对象）来定义有名<name>的一个 POU，或将 <name> 改变成一个已定义的 POU 名。

4051

″‘%S’没有功能″

替代<name>，使用一个在项目或在库中定义的功能名。

4052

″’<名称>’必须是 FB’<名称>’声明的一个实例″

使用一个在项目中定义的数据类型<name>的实例，或将（实例名）类型改为<name>。

4053

″’<名称>’没有有效的盒子或操作符″

使用一个 POU 名或一个在项目定义的操作符来取代<name>。

4054

″所给参数没有有效的 POU 名称″

给出的参数不是一个有效的 POU 名。

4060

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’需要写权作为输入″

具有写存取的变量必须移交给 VAR_IN_OUT 参数，这是因为，一个 VAR_IN_OUT 可在 POU 内修改。

4061

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’必须被用.″

VAR_IN_OUT 参数必须有移交的具有写存取的变量，这是因为，一个 VAR_IN_OUT 可在 POU 中修改。

4062

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’没有外部接口.″

VAR_IN_OUT 参数只能在 POU 内写入或读取，这是因为它们是由引用转交的。

4063

″’<名称>’的‘VAR_IN_OUT‘参数’<名称>’不准应用位地址.″

一个位地址不是一个有效的物理地址。转交一个变量或一个直接的非位地址。

4064

″在本地调用时‘VAR_IN_OUT‘必须复写!″

在局部动作调用中，删除用于 VAR_IN_OUT 变量的参数集。

4070

″POU 包含太复杂的表达式″

通过将表达式分成几个表达式来降代嵌套深度。为此，可使用中间变量。

4071

″网络太复杂″

将网络分成几个网络。

4072

″在 FB 类型’<名称>’和实例’<名称>’的动作标识符应用不一致.″

4100

″‘^‘需要指针类型″

你正在试图将一个不作为指示符说明的变量非关联化。

4110

″[<索引>]需要数组变量″

[<index>]用于一个变量，它不是作为有 ARRAY OF 的一个数组加以说明的。

4111

″一个数组的索引表达式必须是‘INT‘类型″

使用正确类型的表达式，或类型转换。

4112

″数组有太多索引″

检查索引数（1.2 或.3），这些索引是说明数组的，除去多余的索引。

4113

"数组有太少索引"

检查索引数（1.2 或.3），这些索引是说明数组的，添加缺少的索引。

4114

"一个常量索引超出数组范围"

确实弄清楚，所用的索引是在数组界限之内。

4120

"‘.‘需要变量结构体"

句点左边的标识符必须是一个类型 STRUCT 或 FUNCTION_BLOCK 的变量，或是名为一个 FUNCTION 或一个 PROGRAM 的变量。

4121

"’<名称>’不是’<对象名称>’的成分"

成分<name>不包括在对象<object name>的定义内。

4122

"’<名称>’不是调用的功能块的输入变量"

检查一下，被调用的功能块的输入变量，并将’<name>’改变成这些变量中的一个变量。

4200

"需要 LD"

在 IL 编辑器中，在跳转符号后面，至少应插入一个 LD 指令。

4201

"需要 IL 操作符"

每个 IL 指令必须用一个操作符或一个跳转符开始。

4202

"括号内文本意外结束"

在文本后插入一个关闭括号。

4203

"’<名称>’在括号内不允许"

操作符<name>在一个 IL 括号表达式中是无效的。（无效的是：JMP,RET,CAL,LDN,LD,TIME）

4204

"对于结束括号缺少相应的开始括号"

插入一个打开括号，或除去这个关闭括号。

4205

"在‘）‘后不允许逗号"

在关闭括号后面除去逗号。

4206

"括号内不允许有标签"

移去跳转符号，使其在括号外面。

4207

"‘N‘修正符需要操作数类型’BOOL‘’BYTE‘’WORD‘或’DWORD‘"

N 修改符需要可执行布尔求反的数据类型。

4208

″条件操作符需要‘BOOL‘类型″
确实弄清楚，表达式给出布尔结果，或应用类型转换。

4209

″在此不允许功能名″
使用一个变量或一个常数来替换这个功能。

4210

″’CAL’, ’CALC’ 和 ’CALN’需要功能块实例作为操作数″
说明一个你要调用的功能块的实例。

4211

″在 IL 程序中只允许在行的末尾进行注释″
将注释移到行的结束处，或移到附加行。

4212

″条件声明前的累加器错误″
未定义这个累加器。如果一个指令是领先的，它未提交一个结果。

4213

″‘S‘和’R‘需要’BOOL‘操作数″
在这个位置使用一个布尔变量。

4250

″在 POU 结尾需要另一个‘ST‘声明″
该行不是以一个有效的 ST 指令开始。

4251

″在功能’<名称>’内有太多参数″
给出的参数多于功能定义中所说明的参数。

4252

″在功能’<名称>’内有太少参数″
给出的参数少于功能定义中所说明的参数。

4253

″‘IF‘或’ELSIF‘需要’BOOL‘表达式作为条件″
确实弄清楚，IF 或 ELSIF 的条件是一个布尔表达式。

4254

″’ WHILE‘需要’BOOL‘表达式作为条件″
确实弄清楚，WHILE 后的条件是一个布尔表达式。

4255

″’ UNTIL‘需要’BOOL‘表达式作为条件″
确实弄清楚，UNTIL 后的条件是一个布尔表达式。

4256

″‘NOT’ 需要’BOOL‘操作数″
确实弄清楚，NOT 后的条件是一个布尔表达式。

4257

″‘FOR‘声明的变量必须是’INT‘类型″
确实弄清楚，计数器变量是一个整数或位串数据类型。（例如：DINT，DWORD）

4258

″‘FOR‘声明的表达式没有具有可写权的变量″

使用一个有写存取的变量来替换计数器变量。

4259

″‘FOR‘声明的开始值没有具有可写权的变量″

“FOR”指令中的起始值必须与计数器变量类型相兼容。

4260

″‘FOR‘声明的结束值没有具有可写权的变量″

“FOR”指令中的结束值必须与计数器变量类型相兼容。

4261

″‘FOR‘声明的增加值没有具有可写权的变量″

FOR”指令中的起始值必须与计数器变量类型相兼容。

4262

″循环外需要退出″

只能在“FOR”,“WHILE”或“UNTIL”指令内使用“EXIT”。

4263

″需要数字，‘ELSE‘或‘END_CASE‘″

在一个“CASE”表达式内，你只能使用一个数或一个“ELSE”指令，或结束指令“END_CASE”。

4264

″‘CASE‘需要整数类型选择器″

确实弄清楚，选择器是一个整数或位串数据类型的（例如 DINT，DWORD）。

4265

″‘，‘后需要编号″

在 CASE 选择器的枚举时，在逗号后面必须插入一个其他选择器。

4266

″至少需要一个声明″

插入一个指令，至少一个分号。

4267

″功能块的调用需要功能块实例″

功能块的调用中的标识符没有实例。说明所需要的功能块的一个实例，或利用一个早已定义的实例。

4268

″需要表达式″

在这里插入一个表达式。

4269

″在'ELSE'分支后需要有'END_CASE'″

使用一个“END_CASE”来终止“ELSE”部分后面的“CASE”指令。

4270

″'CASE' 常量'<名称>' 已经被应用″

一个“CASE”选择器在一个“CASE”指令内只能使用一次。

4271

″范围的下限值比上限值大.″

修改选择器的区域边界，使其下部边界不大于上部边界。

4272

″在调用'<名称>'的地方<位置>需要参数'<名称>'!″

你可以这样的方式编辑一个功能调用，使它还包含参数名，而不仅包含参数值。但是，无论如何，参数的位置（顺序）必须与功能定义中相同。

4273

″部分'CASE'范围'<范围>'在'范围'<范围>'中已经被应用″

确实弄清楚，用于 CASE 指令中的选择器区域不出现重叠。

4274

″在'CASE'声明时出现多重的 'ELSE'分支″

一个 CASE 指令不能包含多于一个 ELSE 指令。

4300

″跳转需要'BOOL'作为输入类型″

确实弄清楚，相应于 RETURN 指令的跳转输入是一个布尔表达式。

4301

″POU '<名称>' need exactly 需要正确的 <数字> 输出″

输入数不对应于在 POU 定义中给出的 VAR_INPUT 和 VAR_IN_OUT 变量数。

4302

″POU '<名称>' 需要正确的输出 %d ″.

输出数不对应于在 POU 定义中给出的 VAR_OUTPUT 变量数。

4303

″'<名称>' 没有操作符″

使用一个有效的操作符来替换<name>。

4320

″一个触点的开关信号必须是布尔类型的表达式″

用于一个接点的开关信号必须是一个布尔表达式。

4321

″线圈的输出变量必须是布尔类型。″

一个线圈的输出变量必须是类型 BOOL。

4330

″功能块 '<名称>'的输入'EN'处期望一个表达式 ″

将一个输入或一个表达式分配给 POU“<name> ”的输入 EN。

4331

″功能块 '<名称>'的输入'<编号>'处期望一个表达式 ″

没有分配操作符 POU 的输入<number>。

4332

″ 功能块 '<名称>'的输入'<名称>'处期望一个表达式″

POU 的输入是类型 VAR_IN_OUT 而且没有分配。

4333

″跳转处期望一个标识符″

给出的跳转标记不是一个有效的标识符。

4334

"跳转的输入处期望一个表达式"

将一个布尔表达式分配给 RETURN 指令的输入。如这是 TRUE，则将执行这个跳转。

4335

"返回的输入处期望一个表达式"

将一个布尔表达式分配给 RETURN 指令的输入。如这是 TRUE，则将执行着个跳转。

4336

"输出的输入处期望一个表达式"

将一个合适的表达式分配给输出框。

4337

"期望输入的标识符"

在输入框中插入一个有效的表达式或标识符。

4338

" '<名称>'功能块没有输入"

对操作符 POU<name>输入分配有效的表达式。

4339

"输出时的类型搭配错误: 不能转换 '<名称>' 到 '<名称>'.

输出框中的表达式类型与应分配给它的表达式类型不兼容。

4340

"跳转需要 'BOOL' 作为输入类型"

确实弄清楚，跳转是一个布尔表达式。

4341

"返回需要一个布尔输入"

确实弄清楚，用于 RETURN 指令的输入是一个布尔表达式。

4342

"在功能块 '<名称>'的输入'EN'处需要一个表达式"

将一个有效的布尔表达式分配给框的 EN 输入。

4343

"常量的值和声明的不一致"

框<name>的输入<name>作为 VAR_INPUT CONSTANT 说明。但是，在对话框“编辑参数”中，已将一个类型不兼容的表确定式发配给这个 POU 框。

4344

"'S' 和'R' 需要'BOOL' 操作数"

在相应的“复位”指令的设定后面，插入一个有效的布尔表达式。

4345

" '<名称>'的无效的参数类型'<名称>' : 不能转换 '<类型>' 到'<类型>'."

分配给 POU 框<name> 的输入<name>的一个表达式。类型是不兼容。

4346

"不允许一个常量作为输出"

你只能将一个输出分配给一个变量或一个有写存取的直接地址。

4347

"'VAR_IN_OUT' 参数需要可写的输入变量"

只有写存取的变量才可移交给 VAR_IN_OUT 参数，这是因为这些变量可在 POU 内修改。

4348

"无效的程序名称 '<名称>'. 具有同样名称的一个变量已经存在."

4349

"在 POU <名称>中的输入或输出已经被删除: 检查功能块的所有连接.只有当 CFC 编辑后此错误消息才会消失。"

4350

"不能从外部访问一个 SFC-行为动作!"

SFC 动作只能在它们被定义的 SFC POU 内才能调用。

4351

"步的名称没有标识符: '<名称>'"

重新命令步名，或选择一个有效的标识符作为步名。

4352

"有效的步名称后有额外的字符:'<名称>'"

在步名中除去无效的字符。

4353

"步名称被复写: '<名称>'"

重新命名一个步名。

4354

"跳转到没有定义的步: '<名称>'"

选择一个现有的步名作为相应的跳转的目标。插入一个有名的步，“<name>”

4355

"一个转换必须没有任何的边缘效应(分配, FB-调用等.)"

一次转换必须是一个布尔表达式。

4356

"跳转时没有有效的步名称: '<名称>' "

使用一个有效的标识符作为跳转目标（标记）。

4357

"没有找到 IEC-库"

检查一下，库 iecsfc.lib 是否插入在库管理程序内，以及在”project”（项目）——“options”（选项）——“paths”(路径)中规定的库路径是否正确。

4358

"行为动作没有声明: '<名称>'"

确实弄清楚，在对象组织程序内，IEC 步的动作是在 SFC POU 下面插入的，而且在编辑器中，动作名插入在限定符右边的框内。

4359

"无效的限定词: '<名称>'"

在动作名左边的框内，输入一个用于 IEC 动作的限定符。

4360

" 限定词'<名称>'后期望时间常数"

紧邻动作名左侧的框，在一个限定符后输入一个时间常数。

4361

"'<名称>' 不是行为动作的名称"

紧邻限定线右侧的框，输入一个动作名，或在项目中定义的一个变量名。

4362

"行为动作中没有布尔表达式的应用: '<名称>'"

插入一个布尔变量或一个有效的动作名。

4363

"IEC-步名称已经被变量应用: '<名称>'"

请重新命名步或变量。

4364

"一个转换必须是布尔量表达式"

转换表达式的结果必须是类型 BOOL。

4365

"限定词'<名称>'后期望有时间常数"

打开步”<name>的对话框“步属性”，并输入一个有效的时间变量或时间常数。

4366

"平行分支的标签没有有效的标识符: '<名称>'"

在三角形符号旁输入一个有效的标识符，它标记跳转符。

4367

" '<名称>' 标签已经用过"

早已有一个跳转符或一个步使用过这个名。请相应的重新命名。

4368

"'<名称>'行为在多重的一系列步中应用，也就是一个包含其它的。"

动作<name>用于 POU 以及 POU 的一个或几个动作。

4369

"一个转换需要一个严密的网络"

对于一个转换使用了几个 FBD 相应的 LD 网络。请减少到一个网络。

4370

"正确的 IL-转换后有额外的行"

在转换结束处除去不必要的线路。

4371

"有效的表达式后有无效的字符: '<名称>"

在转换结束处除去不必要的字符。

4372

"'<名称>'步: 时间限制需要'TIME'类型"

4373

" 只在 SFC-POUs 下允许 IEC-行为"

4374

"期望步代替'<名称>'转换 "

4375

"期望转换代替'<name>'步"

4376

"'<名称>'转换后期望步"

4377

" '<名称>'步后期望转换"

4400

POU '<名称>'输入/ 转换 有错误。例如没有完成."

POU 不能完全转换到 IEC61131-3，相应的丢失一个操作数。

4401

"S5 时间常量 <名称> 秒数太大 (最大. 9990s)."

在累加器内没有有效的 BCD 编码时间。

4402

"只有 I/O 可进行直接访问."

确实弄清楚，你只存取定义为输入或输出的变量。

4403

"STEP5/7 结构体无效或不能转换成 IEC 61131-3."

有些 STEP5/7 命令不能转换到 IEC61131-3，相应的丢失一个操作数。

4404

"STEP5/7 操作数无效或不能转换成 IEC 61131-3."

有些 STEP5/7 操作数不能转换到 IEC61131-3，相应的丢失一个操作数。

4405

"重置一个 STEP5/7 定时器不能被转换成 IEC 61131-3."

相应的 IEC 定时器没有复位输入。

4406

"STEP5/7 计数器常量超出范围 (最大是 999)."

累加器中没有有效的 BCD 编码的计数器常数。

4407

"STEP5 结构体不能转换成 IEC 61131-3."

有些 STEP5/7 指令不能转换成 IEC61131-3，例如 DUF。

4408

"定时器 或 计数器 的位访问不能转换成 IEC 61131-3."

专用的定时器/计数器命令不能转换成 IEC61131-3。

4409

"ACCU1 或 ACCU2 的内容没有定义, 不能转换成 IEC 61131-3."

与两个累加器相连接的一个命令是不能转换的，这是因为未规定累加器的值。

4410

"没有在工程中调用 POU ."

输入被调用的 POU。

4411

"全局变量表出现错误."

请检查 SEQ 文件。

4412

″内部错误编号 11″

请与 PLC 制造商联系。

4413

″在数据块中格式化行时出错″

应输入的代码中有一个错误的日期。

4414

″FB/FX 名称丢失.″

在需始的 S5D 文件中，丢失一个（扩展的）POU 的符号名。

4415

″不允许块结束后的说明.″

不能输入一个受保护的 POU。

4416

″命令无效″

S5/S7 命令不能被取消。

4417

″注释没有封闭″

使用“*）”来关闭注释。

4418

″FB/FX-名称太长（最多 8 个字符)″

一个（扩展的）POU 的符号名太长。

4419

″期望对行″″(* 名称: 〈FB/FX-名称〉 *)″″格式化 ″

校正相应的行。

4420

″FB/FX 参数名称丢失″

检查 POU。

4421

″FB/FX 参数类型无效″

检查 POU。

4422

″FB/FX 参数类型丢失″

检查 POU。

4423

″ FB/FX 调用参数无效。″

检查 POU 的接口。

4424

″警告: 调用 FB/FX 时 丢失或参数错误或有'0'参数。″

被调用的 POU 尚未输入，或不正确，或无参数（在最后一种场合，你可忽略出错消息)。

4425

″标签定义丢失″

未定义跳转的目标（标记）。

4426

"POU 没有有效的 STEP 5 块名称, 例如 PB10"

修改 POU 名。

4427

"时间类型没有声明"

在全局变量表中加入一个定时器的说明。

4428

"超出打开 STEP5 支架的最大数量。"

不允许使用多于七个的开括号。

4429

"正式参数的名称错误。"

参数名不能超过四个字符。

4430

"参数的正式类型不是 IEC 可改变的。"

在 IEC61131-3 中，定时器，计数器和 POU 不能转换为形式参数。

4431

"调用 STEP5 STL 时其 'VAR_OUTPUT' 有太多的参数。"

一个 POU 不能包含多于 16 个形式参数作为输出。

4432

"不允许标签带表达式。"

在 IEC61131-3 中，跳转标记不能插入在任何需要的位置。

4434

"太多标签。"

一个 POU 不能包含多于 100 个标记。

4435

"在转移 / 调用后, 必须启动一个新的表达式"

跳转或调用后，必须后随一个“load”（装入）命令 LD。

4436

"结果位没有定义, 不能转换成 IEC 61131-3."

VKE 使用的命令不能转换，这是因为，VKE 的值是未知的。

4437

"指令和操作数的类型不一致"

一个值命令用于一个字操作数（或反过来也如此）。

4438

"数据块没有打开 (在 DB 之前插入 C 指令)"

插入一个 ADB。

4500

"变量或地址未被承认"

在项目内未说明监视变量。通过按 F2 按钮，你得到列出说明变量的输入辅助。

4501

″有效监视表达式后的额外特性。″

除去多于的记号。

4520

″程序错误： 在'<名称>'前应该有标记!″

附注不正确。检查一下，”<name>”是否是一个有效的标记。

4521

″程序错误: 不期望的成分 '<名称>'!″

检查一下，附注是否正确编写。

4522

″标记超出程序的要求!″

丢失断开的附注，添加一个“flag off”（除去标记）指令。

4523

″程序{<程序名称>} 不允许的界面类型 '<名称>'″

4550

″索引超出定义范围 : 变量 OD ″<编号>, 行<行号>.″

保证索引是在目标设定/网络功能度中所规定的区域内。

4551

″分索引超出定义范围 : 变量 OD ″<编号>, 行<行号>.″

保证子索引是在目标设定/网络功能度中所规定的区域内。

4552

″索引超出定义范围 : 参数 OD ″<编号>, 行<行号>.″

保证索引是在目标设定/网络功能度中所规定的区域内。

4553

″分索引超出定义范围 : 参数 OD ″<编号>, 行<行号>.″

保证子索引是在目标设定/网络功能度中所规定的区域内。

4554

″变量名称错误: 变量 OD <编号>, 行<行号>.″

保证在字段”variable”（变量）中的一个有效项目变量。使用相应于全局变量(variable name) 的语法 POU 名（变量名)。

4555

″登陆表为空, 输入不能随意: 参数 OD <编号>, 行<行号>″

你必须在这个字段做出一个登录项。

4556

″登陆表为空, 输入不能随意: 变量 OD <编号>, 行<行号>″

你必须在这个字段做出一个登录项。

4557

″必须的参数内存太大″

4558

″必须的变量内存太大″

4560

″错误的值: 路径'<名称>', 纵列'<名称>', 行'<行号>'″

4561

″纵列没有定义: '<名称>'″

4562

″索引/分索引 已经存在: 路径'<名称>', 行'<行号>'″

4563

″标识符'<名称>' 已经存在: 路径'<名称>', 行'<行号>'″

4564

″索引'<名称>' 超出范围: 路径'<名称>', 行'<行号>' ″

在目标设置定义范围内输入一个索引。

4565

″分索引'<名称>' 超出范围: 路径'<名称>', 行'<行号>' ″

在目标设置定义范围内输入一个分索引。

4566

″参数管理器输入时发生错误″

用户应该输入一个在参数管理器中包含错误信息的输出文件。检查*.exp 文件。

4600

″网络变量: '<名称>' 表达式不是布尔类型。″

4601

″网络变量 '<名称>': 发现网络变量交换没有循环或无限循环。″

4602

″'<网络变量名称表>': 对象用 UDP 端口'<端口号>' 代替 '<端口号>'″

在指定的网络变量表 设置中，一个端口号已经应用，它和在全局变量文件夹中找到的第一个网络变量表应用的端口号不同。当心所有的网络变量表应用同一个端口！

4620

工程中发现没有使用的变量。请参考'工程' '检查' 未使用变量 命令的描述。

4621

变量分配内存空间时发生交迭。请参考'工程' '检查' '重叠内存范围'命令的描述。

4622

IEC地址分配的同一个内存空间涉及到多个任务。请参考'工程' '检查' '同时访问'命令的描述。

4623

工程在多于一个地方同时获得对同一个内存的写入权。请参考 '工程' '检查'' '输出量的多重访问'命令的描述。

4650

″轴组'<名称>': 任务'<名称>' 不存在.″

4651

″轴组'<名称>': 没有设置循环时间(dwCycle).″

4652

″驱动器'<名称>': wDriveID 在此轴组中已经存在.″

4670

″CNC 程序'<名称>': 没有找到全局变量 '<名称>' .″

4671

″CNC 程序'<名称>': 变量'<名称>' 具有矛盾的类型.″

4685

″凸轮'<名称>':未知的凸轮表类型.″

4686

″凸轮'<名称>': 凸轮上的点超出了数据类型范围.″

4700

″'<编号>' ('<名称>'): 监视表达式 '<名称>' 不是数字变量.″

4701

″'<名称>' ('<编号>'): 监视表达式 '<名称>' 不是布尔类型.″

4702

″'<名称>' ('<编号>'): 监视表达式 '<名称>' 不是字符串类型.″

4703

″'<名称>' ('<编号>'): 监视表达式'<名称>'错误″

可视化窗口包含了错误的变量。

4704

″'<名称>'('<编号>'): 监视表'<名称>'初始化值错误.″

检查所用的表。

4705

″'<名称>' ('<编号>'): 报警表分配的报警组无效.″

在报警表配置对话框中输入一个有效的报警组。(报警种类表)

4900

″类型转换错误″

当前选择的代码转换器不支持你所用的类型转换。

4901

″内部错误：数组存取溢出!″

数组范围超过 32 位变量，减少数组索引范围。

5100

″<名称> (<Zahl>): 表达式太复杂。没有寄存器可用。″

对于可用寄存器来说指定的表达式太复杂。请尝试用中间变量来减少表达式的复杂度。

索引：

符号
‘插入’ ‘类型’..... 135
‘插入’ ‘声明关键字’..... 135
‘插入’ ‘新声明’..... 138
‘数据库连接’..... 58
’插入’’输入针’ ’插入’’输出针’. 174
’附加’’下一个不同处’..... 85
’工程’’合并’..... 86
“SEQUENCE”第二扩展部分..... 36
“SEQUENCE”第一扩展部分..... 32
“WAIT”声明部分..... 31
“WAIT”主体部分..... 32
“符号配置”选项..... 56
“交通信号”声明..... 30
“交通信号”主体部分..... 30

A

ABS..... 280
ACOS..... 283
ADD..... 261
ADR..... 273
ADRINST..... 273
AND..... 264
ASIN..... 282
ATAN..... 283
AT 声明 134

B

BCD_TO_INT..... 310
BITADR..... 274
BLINK..... 314
BOOL 常量..... 284
BOOL_TO 转换..... 275

C

CAL..... 274
CAN 主模块的 CAN 参数..... 226
CAN 主模块的基本参数..... 226
CanDevice 的 CAN 设置 231
CanDevice 的基本设置 230
CanDevice 默认的 PDO 映射 232
CAN 模块的 PDO（数据处理对象）映射. 228
CAN 模块的基本参数..... 227
CAN 主模块的模块参数..... 226
CASE..... 18
CFC 的当前位置 173
CHARCURVE..... 316
CheckBounds..... 293
CONCAT..... 300
CTD..... 305
CTU..... 305
CTUD..... 306

D

DATE 常量..... 285
DATE_AND_TIME 常量..... 285
DATE_TO/DT_TO 转换..... 278
DDE 和 CoDeSys 的通讯..... 256
DDE 网关服务器的操作..... 257
DDE 网关服务器的命令行选项..... 259
DELETE..... 301
DIV..... 262
DP slave 的 DP 参数..... 222
DP slave 的模块参数..... 225
DP slave 的输入 / 输出..... 223
DP slave 的用户参数..... 224
DP slave 的组分配..... 225
DP 主模块的 DP 参数..... 219
DP 主模块的基本参数..... 219
DP 主模块的模块参数..... 219
DP 主模块的总线参数..... 220

E

EQ272
EXIT..... 21
EXP..... 281
EXPT..... 283

符号

’Extra’’Read Trace’..... 235, 237
’Extra’’Start Trace’..... 235
’Extra’’Stop Trace’..... 236

E

EXTRACT..... 310

符号

'Extras' 'Compress'..... 238
'Extras' 'Cursor Mode'..... 236, 238
'Extras' 'Multi Channel'..... 238
'Extras' 'Show grid'..... 237
'Extras' 'Stretch'..... 237
'Extras' 'Y Scaling'..... 235

F

F_TRIG..... 304
FBD 的当前位置 155
FIND..... 301
FOR..... 19
FREQ_MEASURE..... 314

G

GE272
GEN..... 315
GT271

H

HYSTERESIS..... 317

I

I/O 模块的基础参数 215
I/O 模块的模块参数/客户参数 217
IEC 步..... 23
IF 指令..... 18
INDEXOF..... 264
INI 操作符..... 284
INSERT..... 300
INT_TO_BCD_..... 310

L

LD 编辑器的当前位置 159
LD 和 FBD..... 27
LE272
LEFT..... 299
LEN..... 298
LIMIT..... 270
LIN_TRAFO..... 311
LN281

符号

'Load from controller'..... 237
'Load from file'..... 237
'Load Values'..... 237

L

LOG..... 281
LT271

M

MAX..... 269
MID..... 300
MIN..... 270
MOD..... 263
MOVE..... 263
MUL..... 261
MUX..... 271

N

NE273
NOT..... 266

O

ON 开关 41
OR265

P

PACK..... 310
PD312
PID..... 313
PID_FIXCYCLE..... 313
PLC_PRG..... 38
PLC_PRG 的用途 30
PLC 浏览器的更多选项..... 245, 248
PLC 配置中的诊断：..... 216
POU（程序组织单元） 6
PUTBIT..... 310

R

R_TRIG..... 303
RAMP_INT..... 316
RAMP_REAL..... 317
REAL / LREAL..... 291
REAL/LREAL 常量..... 286
REAL_TO-/ LREAL_TO 转换..... 277
REPEAT..... 20
REPLACE..... 301
RETURN 指令..... 18

RIGHT..... 299
ROL..... 267
ROR..... 268
RS302
RTC..... 309

符号

'Save to file'..... 236
'Save Values'..... 237

S

SEL..... 269
SEMA..... 303
SEQUENCE 的用途 30

符号

'Set as project configuration'.... 237

S

SFC..... 24
SHL..... 266
SHR..... 267
SIN..... 282
SIZEOF..... 264
SQRT..... 281
SR302
STATISTICS_INT..... 312
STATISTICS_REAL..... 312
STRING 常量..... 286
STRING_TO 转换..... 279
SUB..... 262

T

TAN..... 282
TIME 常量..... 284
TIME_OF_DAY 常量..... 285
TIME_TO/TIME_OF_DAY 转换..... 278
TO_BOOL 转换..... 276
TOF..... 308
TON..... 308
TP307

符号

'Trace in ASCII-File'..... 238

T

TRUNC..... 280

U

UNPACK..... 310

V

VARIANCE..... 312

W

WAIT 的用途 30
WHILE..... 20

X

XOR..... 265

A

按表格形式声明..... 137

B

版本标签..... 97

符号

'帮助' '内容' 和 '搜索'..... 127

B

保持变量..... 133
保存报警..... 197
保存工具快捷键..... 253
报警类..... 192
报警配置综述..... 191
报警组..... 195
比较结果的请求..... 83

符号

'编辑' '查找'..... 109
'编辑' '查找下一个'..... 109
'编辑' '撤消'..... 107
'编辑' '复制'..... 108
'编辑' '宏'..... 112
'编辑' '剪切'..... 107
'编辑' '前一个错误'"..... 112
'编辑' '删除"..... 108
'编辑' '输入助手'..... 110
'编辑' '替换'..... 109
'编辑' '下一个错误'"..... 112

'编辑' '粘贴' 108
'编辑' '重复' 107
'编辑''变量声明' 112

B

编辑参数列表 243
编辑翻译文件 77
编辑器的组件 127
编辑器和控制单元之间的列表传输 244
编辑剩余的全局变量列表 188
编辑许可证信息对话框 260

符号

'编辑'选项 48
'编译及生成'选项 53

B

变量 287
变量中的地址符 287
标签 151
标准库 199
表达式 16
并行访问 90
布尔变量(BOOL) 290

C

采样追踪 27
菜单日志 201
参数管理入口的 Pragma 指令 141
参数列表的类型 241
参数列表的排序 244
参照符 296

符号

'插入' ' LD 中 EN 的框' 163
'插入' "平行分支(右)" 166
'插入' '标签' 174
'插入' '并行分支(左)' 166
'插入' '步-转换 (前)' 166
'插入' '步-转换(后)' 166
'插入' '操作符'在文本编辑器中 146
'插入' '操作数'在文本编辑器中 147
'插入' '插入任务' 或 '插入' '扩展任务' 204
'插入' '常闭并联接点' 162
'插入' '常闭接点' 161
'插入' '常开并联接点' 162
'插入' '常开接点', 注释 160
'插入' '返回' 156, 164, 174
'插入' '复位线圈' 162
'插入' '赋值' 155
'插入' '功能' 在文本编辑器中 147
'插入' '功能块' 162
'插入' '功能块' 在文本编辑器中 147
'插入' '计时器 (TON)' 163
'插入' '框' 156, 173
'插入' '上升沿触发器' 163
'插入' '输出' 157, 174
'插入' '输入' 157, 174

C H

插入 输入/输出 "不工作状态" 178

符号

'插入' '输入框' 174
'插入' '所有实例路径' 189
'插入' '添加调用程序'或 '插入' '扩展程序调入' 205
'插入' '添加进入动作' 167
'插入' '添加库' 200
'插入' '跳转' 156, 163, 167, 174
'插入' '网络 (后)' 160
'插入' '网络 (前)' 160
'插入' '新建监控列表' 209
'插入' '选择分支(右)' 166
'插入' '选择分支(左)' 166
'插入' '在 LD 中插入块' 163
'插入' '增加退出动作' 167
'插入' ''置位线圈' 162
'插入' '注释' 174
'插入' '转换跳转' 167

C H

插入列表 242
插入\下降沿检测触发器 163

符号

'插入''线圈' 162

C H

常量 133
撤销多重校验确认 96
程序 11
初始化，追踪，符号创建，位存取的预处理 Pragma 指令 139
触点 26

符号

'触发语言翻译' 79
'窗口' '层叠窗口' 127
'窗口' '垂直平铺' 126
'窗口' '关闭所有窗口' 127
'窗口' '日志' 200
'窗口' '水平平铺' 126
'窗口' '信息' 127
'窗口' '最小化排列' 127

C H

窗口设置 126
创建 POU 29
创建翻译文件 75
创建连接 177
创建全局变量列表 185
创建新的工具快捷键 252
创建一个顺序功能图 33
创建一个新的可视化 40

C

存储器地址 290

D

打印范围 127
打印文字 286
带 EN 输入的 POUs 162
单步 28
单循环 28
登录 92
地址 289
第二个交通信号灯 41
调试 5, 28
调用一个功能块 9
动作 12, 22
动作和转换条件 34
断点 28
断点处的状态 149
对操作数赋值 18
对象 98
对象管理器 44
对象设置对话框 211
对象支撑软件包 211
多个变量列表 184
多重定义 96
多重校验 96
多重校验确认 96

F

翻译工程（到其他语言） 78
返回标准配置 214
仿真 29
非结构化显示 110
非永久数据类型的语法 144
服务数据对象 SDO 229
符号名 213

符号

'附加' '显示排序' 178
'附加' "清除动作/转换" 168
'附加' 'EN/ENO' 175
'附加' '保存监控列表' 209
'附加' '编辑宏' 181
'附加' '步的特性' 168
'附加' '创建宏' 181
'附加' '打开实例' 128, 158
'附加' '导出' 244
'附加' '导入' 244
'附加' '调用堆栈' 208
'附加' '读取监控列表' 209
'附加' '读取配方' 210
'附加' '关联动作' 170
'附加' '后退一级宏', '附加' '后退所有宏' 182
'附加' '监控激活' 210
'附加' '监控选项' 147
'附加' '接受访问权限' 86
'附加' '接受改变' 86
'附加' '接受改变项目' 86
'附加' '接受改变属性' 86
'附加' '扩展宏' 182
'附加' '连接标记' 177

'附加' '连接文本文件''. 190
'附加' '排序' '拓扑排序'. 178
'附加' '排序' '向上移动一步'. 179
'附加' '排序' '向下移动一步'. 179
'附加' '排序' '依照数据流排序'. . . . 179
'附加' '排序' '移动到末端'. 179
'附加' '排序' '移动到首前端'. 179
'附加' '启用/禁用任务'. 208
'附加' '前一个不同处'. 85
'附加' '取反'. 157, 164, 175
'附加' '删除监控列表'. 209
'附加' '设置调试任务'. 207
'附加' '生成文本框架文件''. 190
'附加' '时间总览'. 169
'附加' '使用 IEC 步'. 170
'附加' '特性'. 200
'附加' '添加移动/转换'. 168
'附加' '写入配方'. 210
'附加' '选项'. 169
'附加' '增加并行分支标志'. 167
'附加' '粘贴并行分支(右)'. 167
'附加' '粘贴在后面'. 164, 168
'附加' '粘贴在上面'. 164
'附加' '粘贴在下面'. 164
'附加' '置位/复位'. 157, 164, 175
'附加' '属性 '. 176
'附加'菜单：设置. 198

F

附加的 CoDeSys 功能. 6
附加的联机功能. 5

符号

'附加'预览'. 158

F

附属域类型. 296
复制原理. 177

G

改变连接. 178
更新状态. 98

符号

'工程' '比较'. 81
'工程' '编译生成'. 73
'工程' '打开对象'. 102
'工程' '打开实例'. 105
'工程' '导出'. 80
'工程' '导入'. 81
'工程' '导入西门子程序及变量'. 81
'工程' '对象访问权限'. 104
'工程' '对象复制'. 102
'工程' '对象删除'. 100
'工程' '对象属性'. 103
'工程' '对象转换'. 102
'工程' '翻译成其他语言'. 75
'工程' '工程信息'. 86
'工程' '加载下载信息'. 74
'工程' '检查'. 89
'工程' '全部清除'. 74
'工程' '全部重新编译生成'. 74
'工程' '全局搜索'. 88
'工程' '全局替换'. 89
'工程' '数据库连接'. 91
'工程' '添加动作'. 104
'工程' '添加对象'. 100
'工程' '文档'. 79
'工程' '显示调用树'. 105
'工程' '显示交叉参考'. 105
'工程' '用户组密码'. 90
'工程' '重命名对象'. 101

G

工程->选项. 45
工程变量历史记录. 96
工程数据库中的项目种类. 255
工具栏. 44
工作区. 45
功能. 6, 290
功能块. 7
功能块实例. 8
关键字. 134
关于工程数据的工程对象和共享对象的选项. 60
关于工程数据库编辑文件的选项. 61
关于工具的常见问答. 253
管理工程中的对象. 98

H

宏选项..... 62
获得所有最新版本..... 96
获得最新版本..... 93
获取变量数组，结构和 POUS 287

J

积分..... 311
积分数字类型间的转换..... 277
激活 DDE 接口..... 256
激活步..... 22
激活参数管理器..... 239
加载模块状态..... 234

符号

'加载与保存'选项..... 46

J

监视..... 28
剪切，复制，粘贴和删除..... 158
建立期望的网关服务器和通道..... 122
交通信号的用途..... 29
交通信号灯的例子..... 41
交通信号模拟..... 30, 39
节点保护..... 227
结构..... 295
结构化显示..... 111
结果..... 38
紧急报文：..... 228
进入和退出动作..... 22
警报的一般信息，术语..... 192
旧 PLC 配置的转变..... 214
局部变量..... 133

K

可视化..... 13
可视化交通信号单元..... 39
可视化中的字体..... 42
可选分支..... 25
可以读取什么样的变量?..... 259
可以读取什么样的变量？..... 257
可用工具快捷键的特性（对象特性）.. 249
控制库声明部分显示的语法..... 144
库13
快捷模式..... 135
快速检查连接网关的不成功连接..... 125

L

类型变换功能块..... 275
连接标准库 standard.lib..... 31

符号

'联机' '创建引导工程'..... 126
'联机' '从 PLC 中读取文件'..... 126
'联机' '单步进入'..... 118
'联机' '单步跳过'..... 117
'联机' '单循环'..... 118
'联机' '登录'..... 112
'联机' '断点对话框'..... 117
'联机' '仿真模式'..... 122
'联机' '复位'..... 116
'联机' '复位 (冷)'..... 116
'联机' '复位(初始状态)'..... 117
'联机' '解除强制'..... 119
'联机' '强制新值'..... 119
'联机' '设置断点'..... 117
'联机' '停止'..... 116
'联机' '通讯参数'..... 122
'联机' '退出'..... 116
'联机' '文件写入 PLC'..... 126
'联机' '下载'..... 116
'联机' '显示调用堆栈'..... 121
'联机' '显示流程控制'..... 121
'联机' '写入/强制对话框'..... 120
'联机' '写入新值'..... 118
'联机' '源代码下载'..... 126
'联机' '运行'..... 116

L

联机模式的顺序功能图表..... 170
联机模式下的 CFC..... 182
联机模式下的 IL 150
联机模式下的功能块图..... 158
联机模式下的任务配置..... 206
联机模式下的梯形图..... 164
联机模式下的网络编辑器..... 153
联机模式下的文本编辑器..... 147
联机模式下改变值..... 28
联机模式中的声明编辑器..... 138

符号

'另存为模板' 101

L

律动设置：（替代节点保护） 228

M

枚举 295
密码选项 54
命令占位符描述中使用的命令： 328

符号

'目录'选项 51

N

内存重叠区域 89
内容操作符 274

P

平行分支 25
屏幕分割器 44

Q

其它的交通灯 41
强制值 210
切换到 POU 128, 183
取消校验确认 94
全局常量 188

R

任务配置的工作状态 203
日志 29
日志选项 52
如果系统报告 115
如何测试自己的工程 5
如何构建一个工程 5
如何建立一个工程 5
如何设置 FBD 的当前位置 155
如何设置断点 148
如何在本地 PC 表示通信参数菜单 124

S

扫描模块配置 234

S H

删除标志 168
删除步和转换条件 166
删除断点 148
删除工具快捷键 252
删除连接 178
删除列表 243
上下文菜单 45
上下文关联帮助 127
声明编辑器中行号 137
声明变量 134
什么参数列表？： 239
什么是 ENI 254
什么是参数？： 239
什么是全局变量 184
时间 日期类型 291
实例和模板 241
使用 ENI 工程数据库的前提 254
使用库管理器 199
输出变量 132
输入变量 132
输入输出变量 133
数据类型 13, 290
数据连接的普通方法 256, 259
数字常量 285
数组 292
顺序功能图种的隐含变量 24

S

缩放 151

T

梯形图中的功能块 27
添加共享对象 97
添加配置文件： 214
跳转 25
通道参数 218
通道基础参数 217
推荐的标识符命名方式 130
脱机方式下的监视和收据管理器 208
脱机状态下的标识符 129

W

外部变量 134
网关系统的原理 122
网络 151
网络变量 185

微分 311
为本地网关联机新的通道 123
为标准的从模块选择 CAN 模块 228
未使用变量 89

符号

'文件' '保存' 65
'文件' '保存/发送压缩文件' 68
'文件' '从模板中新建' 64
'文件' '打开' 64
'文件' '打印' 71
'文件' '打印设置' 71
'文件' '关闭' 65
'文件' '另存为' 65
'文件' '退出' 73
'文件' '新建' 64

W

文件夹 99

X

系统标记 287
系统事件 205
显示版本历史记录 94
显示不同处 94

符号

'显示翻译工程' 78

X

显示诊断信息 234
线圈 27
限定词 23
限制警告 317
校验 94

符号

'新建文件夹' 99

X

信息窗口 45
修饰符和操作符 14
许可证管理器 260
选择要显示的变量 235
选择元件 176

符号

'颜色'选项 50

Y

移除库 200
移动原理 176
用 EXCEL 来连接变量 257
用 EXCEL 连接变量 259
用 Intouch 来访问变量 257
用 WORD 来连接变量 257
用 WORD 连接变量 259

符号

'用户信息'选项 47

Y

用户组 90
语句颜色 135
元件的移动 160
元素选择 213

符号

'源代码下载'选项 55

Z

在 CFC 中的反馈路径 182
在 CoDeSys 中运行 ENI 项目的数据库 .. 255
在 PLC 浏览器输入命令时使用宏指令 .. 248
在 PLC 浏览器中的命令输入 246
在 Profibus 从模块操作中的 DP Slave 特性 225
在 SFC 中标记块 166
在比较模式下工作 85
在结构化文本中调用功能块 18
在可视化中插入元件 40
在启动工程中的参数列表 245

符号

'在全局变量' 中的对象 184

Z

在输出端多通道写访问 89
在通信参数菜单中编辑参数的 Tips.... 125
在文本编辑器中调用带有输出参数的 POUs 147
在文本编辑器中断点位置 148

在文本编辑器中行号码 149
在线模式下的监控和配方管理器 209

符号

'展开节点' '收缩节点' 99

Z H

整数数据类型 290
正在处理哪个任务? 206
执行比较 82
执行次序 178
执行工具快捷键 252
指针 294
智能功能 128
置位/复位线圈 27
重命名列表 243
重新计算模块地址 214
主菜单 44
主窗口的组成 43
注释, 网络中的行断点, '附加' '选项' 151
贮存文件日志记录 202
转换/转换条件 22
状态栏 45

符号

'桌面'选项 49

Z

资源 13
字符串 291
自定义的库 199
自动声明 136