infoteam
software AG

用户手册

OpenPCS Version 7.0

内容

1 使用 OpenPCS

1.1 安装

OpenPCS 以 CD 的形式交付。CD 会自动运行一个安装窗口，在这里您可以选择想安装的软件。如果自动运行没有被激活或者不起作用，请运行子目录 SOFTWARE\ OPENPCS\ <*语言*>\ 下的 SETUP.EXE。

在安装的最后，会提问是否安装硬件驱动。如果您的 PLC 中有的话，输入硬件驱动的路径，否则点击 退出 。您在获得 PLC 的驱动的同时，也得到了一个 OpenPCS 的 许可证号码。要添加一个许可证号码，请参看许可证编辑器。

如果没有硬件驱动器或许可证，OpenPCS 将正常工作，但是使用 模拟 时会有所限制。

注意： Windows XP 不支持安装到替代驱动器。

1.2 硬件和软件要求

OpenPCS 要求 PC 至少有以下配置：

- Pentium II，1 GHz

- 512 MB RAM

- 180 MB 可用硬盘空间

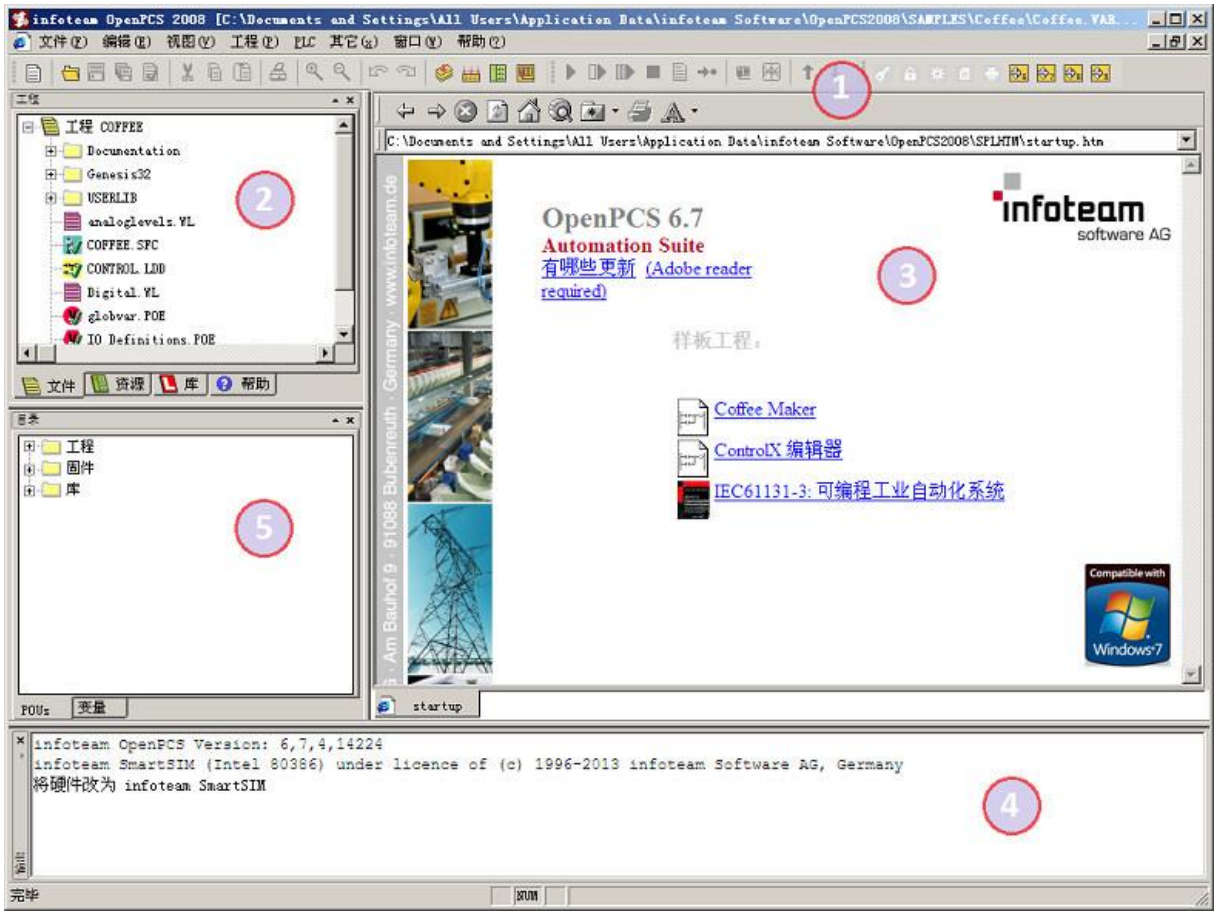
- CD-ROM 及 1024×768 分辨率

- Windows 2003，Windows XP SP2，或 32 位的 Windows Vista

为了支持特定的 PLC，还应该有更多的必要条件（例如更多的内存空间），还需要额外的硬件或软件（例如接口卡、电缆等）。如果有任何疑问，请查看您的 PLC 手册。

1.3 启动 OpenPCS

启动 Windows，在开始菜单中选择 ***开始*** -> ***程序***->***OpenPCS***，这将打开 OpenPCS-框架。如果工程是由 OpenPCS 版本 7.0 以前生成的，在打开工程时 OpenPCS 会询问是否将工程文件转换为 UTF-8 格式。



屏幕分成 5 个区域：

1. 最顶端是菜单和工具栏
2. 工程浏览器
3. 编辑窗口
4. 输出窗口
5. 目录窗口

启动时默认显示最近打开过的工程。

在编辑窗口中会显示随产品提供的实例工程，因此可能与本图不同。

1.4 OpenPCS 例子

OpenPCS 提供了多个例子工程。第一次运行 OpenPCS 会显示带有一栏例子工程的启动屏幕。点击其中一个，就可以打开它。

Coffee	模拟一个咖啡机。在下面几页会做详细解释。
ControlX	示范 IL 语言，ST 语言，顺序功能图 SFC 及梯形图 LD
BookExam	《IEC61131-3：工业自动化系统编程》一书中的例子，作者 Karl-Heinz John and Michael Tiegelkamp

如果没有显示这个界面，您可以在您的 OpenPCS 目录里的例子文件夹里寻找。

注意：

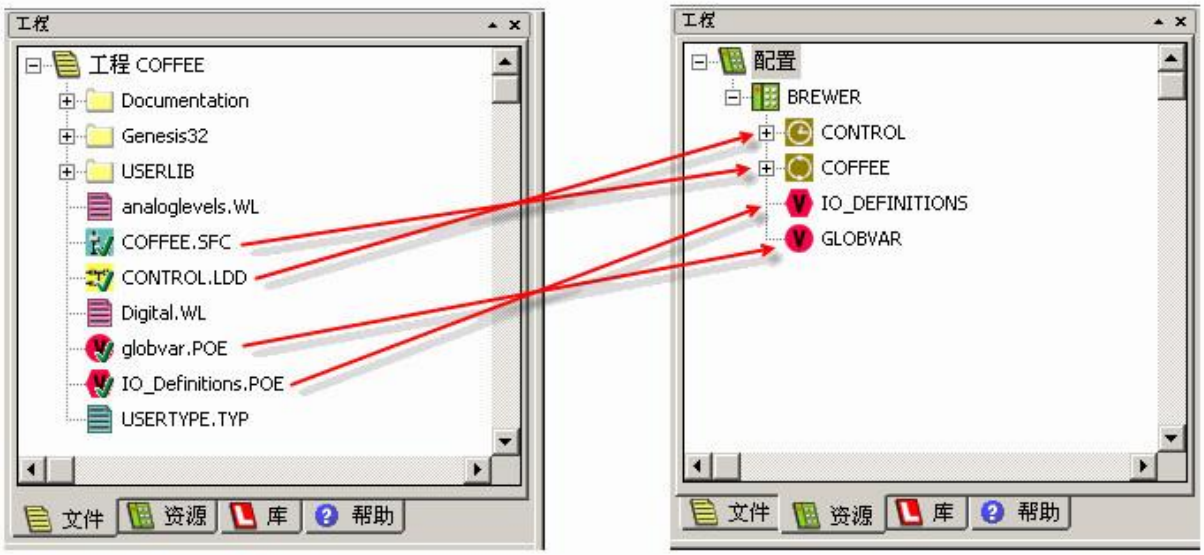
1. 每个例子工程打开时，它的有关描述会自动显示。
2. 您看到的例子中可能被改变了，这是 OEM 通过 Brand Labelling OpenPCS（OpenPCS 商标标记）独立完成的。

1.5 导览

1.5.1 导览：简介

这个导览将引导用户逐步进入 OpenPCS 编程系统。

Coffee 实例随 OpenPCS 提供，位于 OpenPCS SAMPLES 目录下。这个实例模拟了一个咖啡机并对它的工作流程进行控制。这两个步骤各自又分为两个任务。模拟机器的工作由一个名为 coffee.sfc 的顺序功能图（sequential function chart, SFC）来完成。在这里我们使用能够反映咖啡机物理性质的时间线。梯形图 control.ldd 用来控制任务。同时我们需要两个声明文件 globvar.poe 和 IO_Definitions.pos 来完成两个任务之间的交互作用。在工程浏览器的资源窗格中可以看到任务的分割。在这里 BREWER 资源包含了前面提到的两个任务以及它们的定义。展开树结构就能够看到它们各自的变量。下图显示了文件与资源之间的关系。

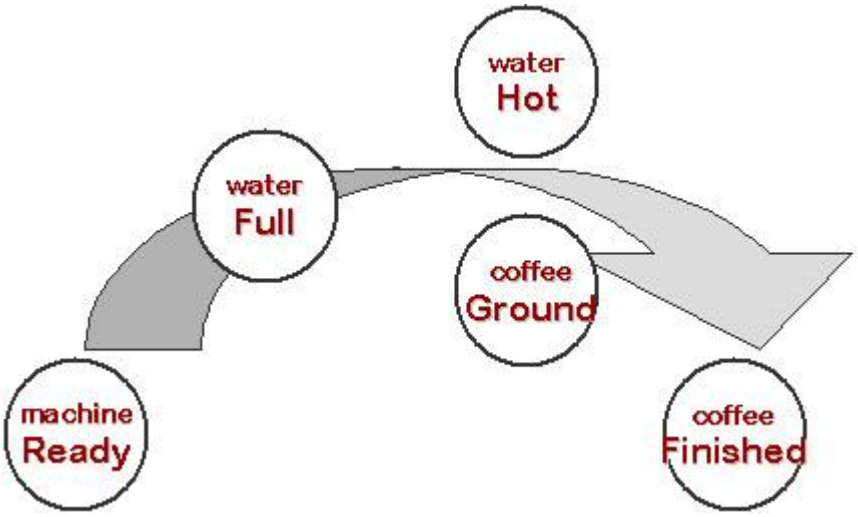


要把新文件添加到资源，用户需要在工程浏览器中右键点击该文件，然后选择 连接到激活的资源 。如果文件被成功连接到资源，那么它的图标中会出现一个绿色的勾。

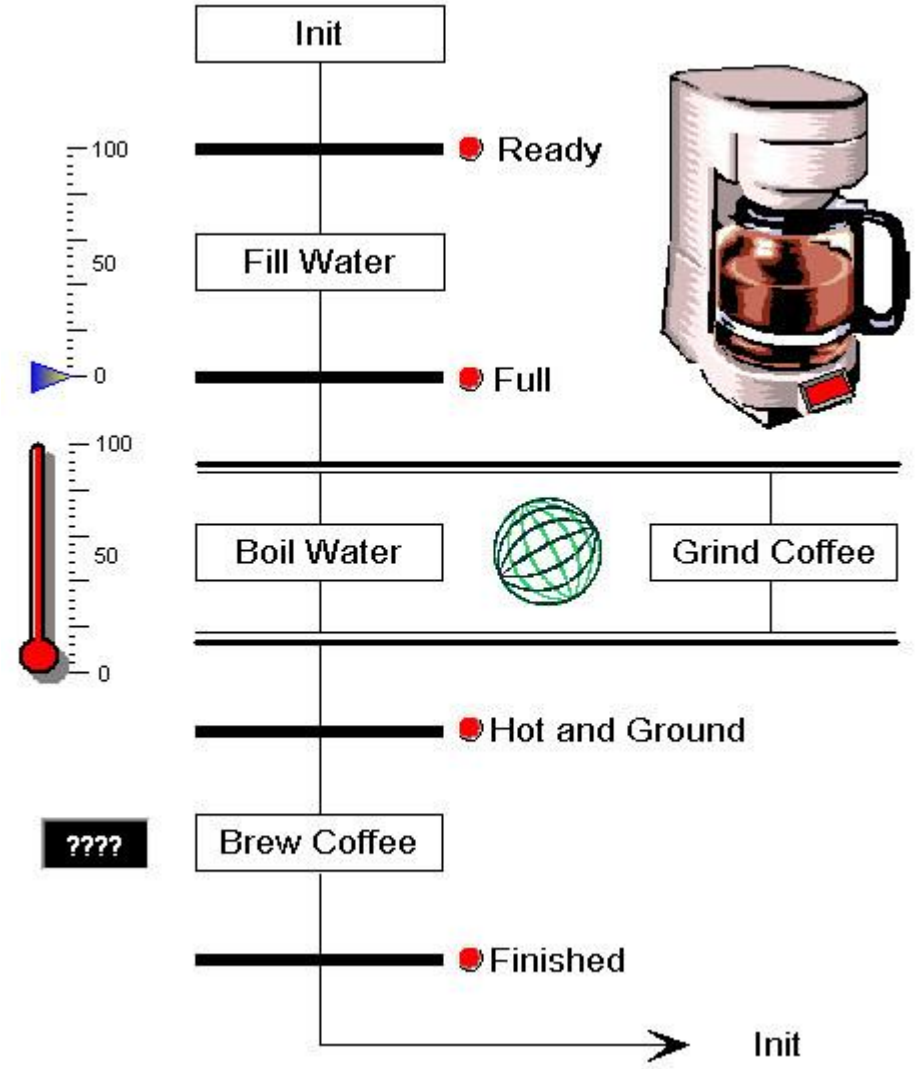
我们的导览分为两节。第一节为读者介绍实例程序并解释所用到的常规功能。第二节会讲解怎样用 OpenPCS 对程序进行编译、执行和监控。

1.5.2 编程实例

一台咖啡机可以用 5 个状态来模拟。一个强制初始状态机器就绪（machine ready），以及水已加满（water full），水已热（water hot），咖啡已磨好（coffee ground）和咖啡完成（coffee finished），如下图所示。对 coffee.sfc 的编程就是基于这些状态。正如图中所示，研磨咖啡和水的加热可以同时进行。



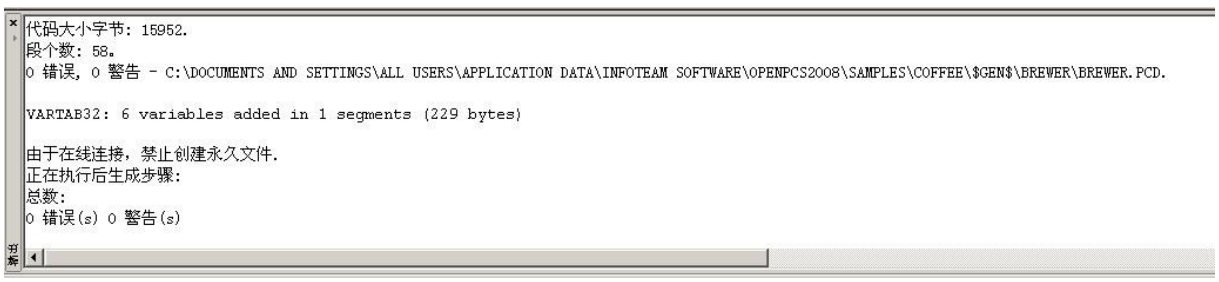
用 IconicsWorkX 可以形象地描绘出这台咖啡机。我们描绘了模拟咖啡机的 coffee.sfc 功能图，和表示水温和水位的显示图。水已热和咖啡已磨好两个状态分别由 Hot 和 Ground 来表示。



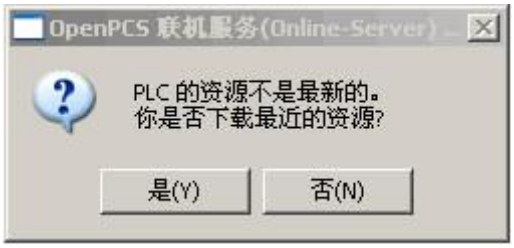
1.5.3 执行代码

请打开实例工程 coffee.var。

要执行这个应用程序，我们需要编译它，并把代码传输给控制器。要为控制器生成代码，选择菜单栏中的 PLC-生成当前资源。在输出窗口中，可以看到编译进程。最后的输出与下图类似：

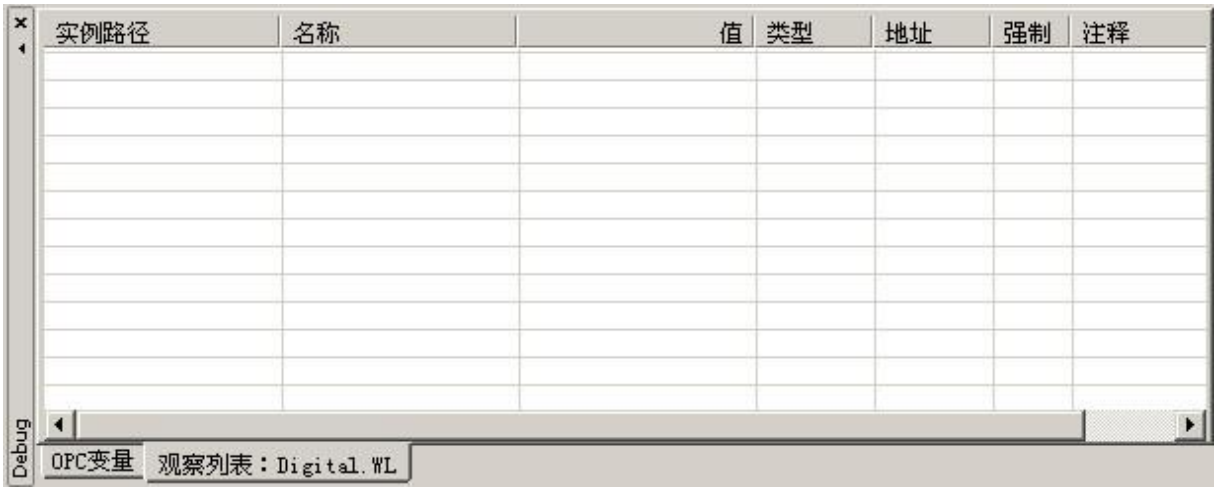


编译成功之后，代码必须要传输给控制器。现在选择 ***PLC->联机*** 来连接资源。OpenPCS 会探测到，你的应用程序要被下载，并会征求你的允许。

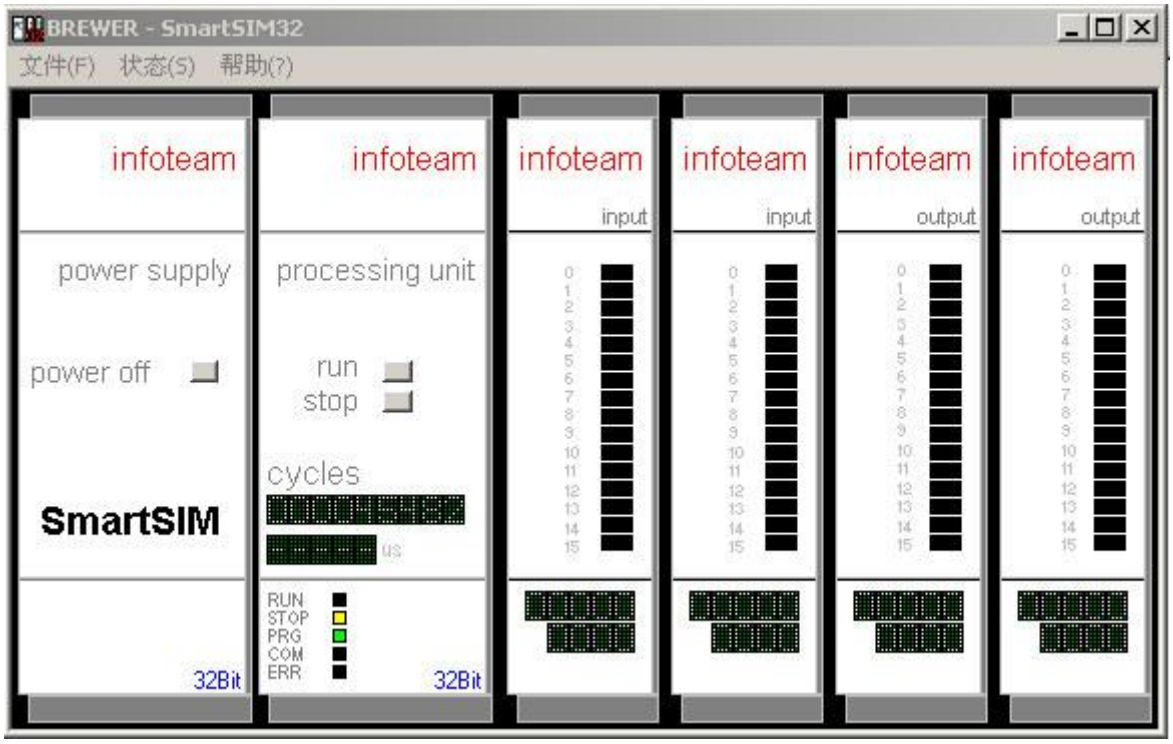


如果出现问题，OpenPC 会在同一个窗口中显示错误信息。OpenPCS 手册中概括了所有的错误并提供了可能的解决方法。

点击 是 来同意下载。在传输代码的过程中，你会看到一个进程条，但是在这个小实例中，这个过程很快就完成了。下载完成后，你会看到 OpenPCS 自动打开它的另一个工具， Debug 。这证明 OpenPCS 已联机：

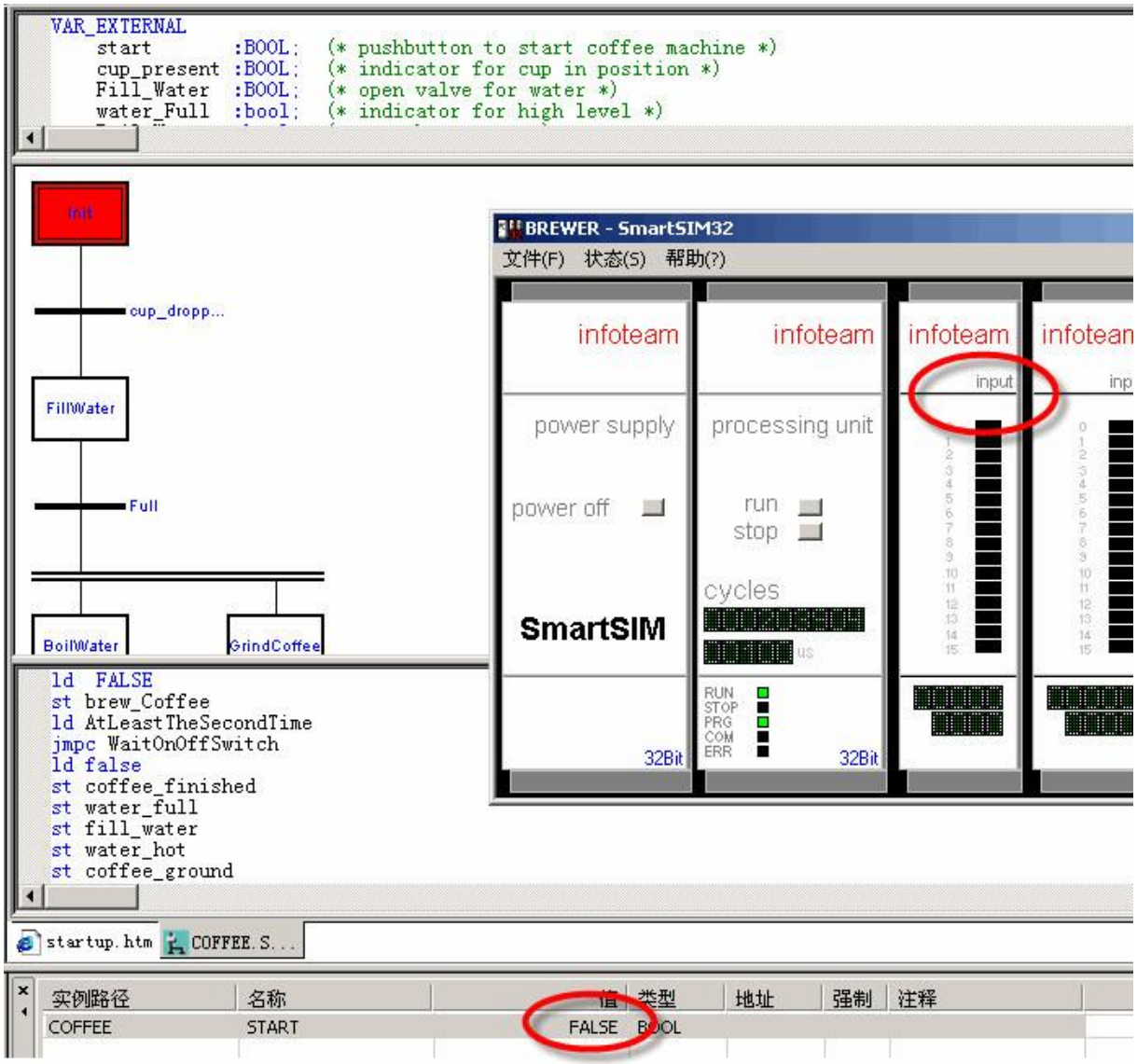


在本简介中，我们没有使用真正的硬件的控制器。代替它的是随 OpenPCS 提供的 Windows 模拟工具，叫做 SmartSIM。OpenPCS 在下载程序的时候会自动打开它。

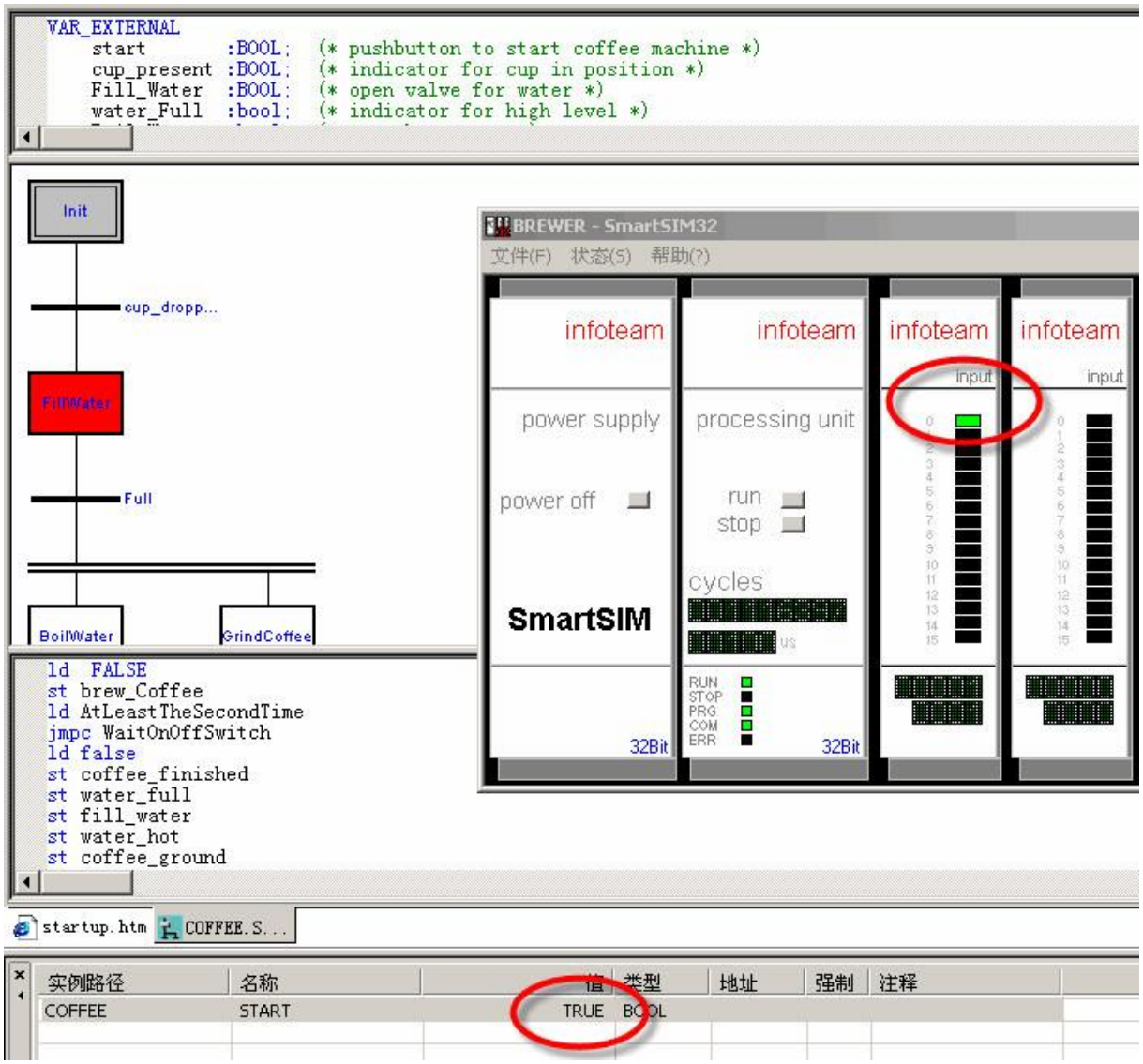


用菜单中的 ***PLC->冷启动***（或点击工具栏中的红色箭头）来执行代码。

在开始执行后屏幕显示如下。咖啡机被初始化，初始步标记为红色。



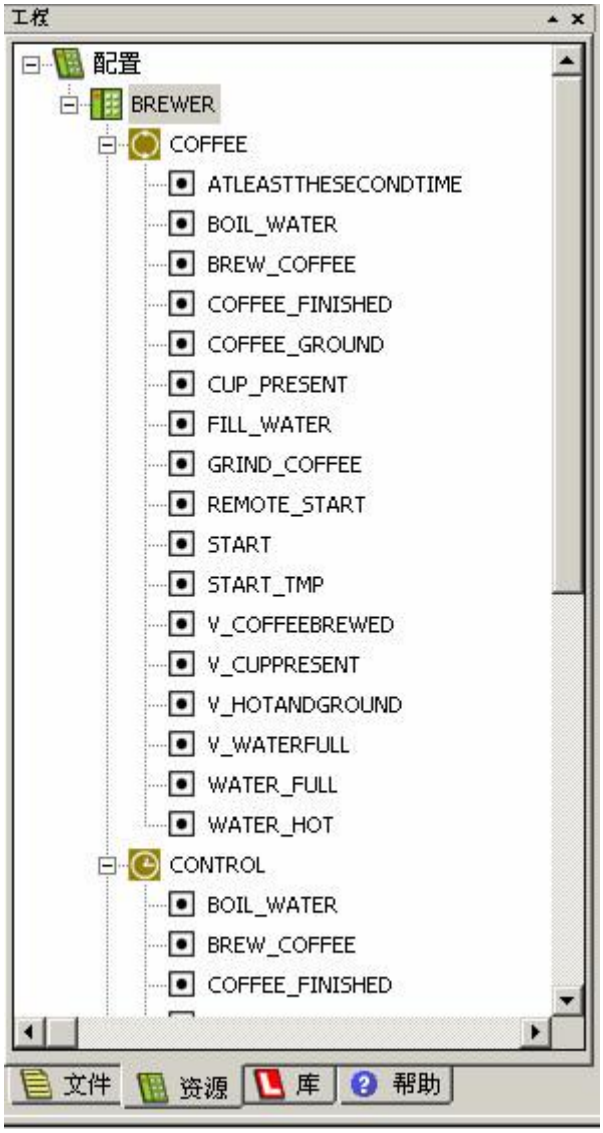
在 SmartSIM 中激活第一个输入钮（input）。在屏幕下方的观察列表中可以看到，变量 start 的值变为真（TRUE），机器开始工作。另外，在顺序功能图中， Fill_Water 步被激活并标记为红色。



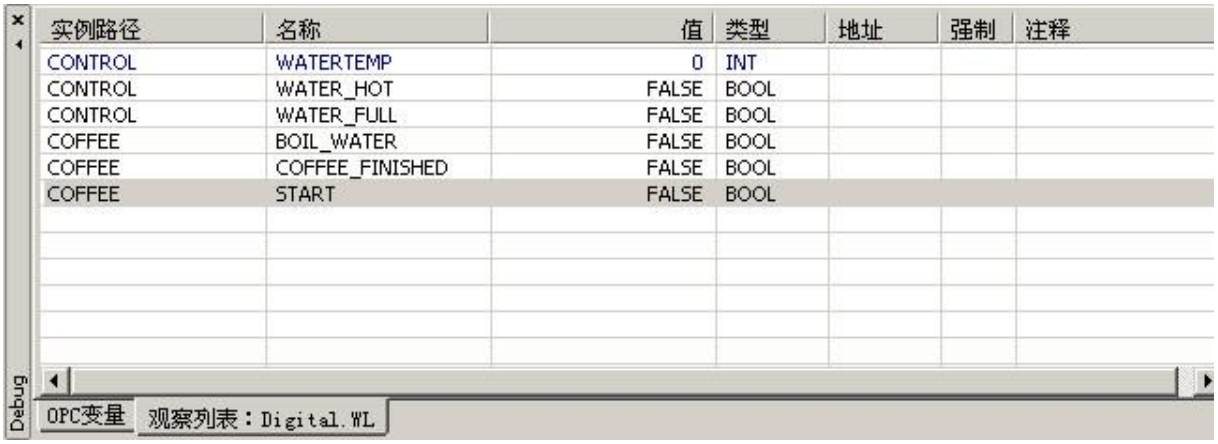
在第一杯咖啡完成之前，其它的咖啡无法进行烹煮。

1.5.4 监控代码

现在你的应用程序已经在运行了，回到左上角的工程窗口并打开资源栏。在这个栏中可以看到所有的资源以及它们的任务和变量。点击所有的小加号打开资源条目下的整个树结构。这将展示 实例树 ，它将所有程序和功能块的实例，以及程序中使用的变量都显示出来：



双击一些变量条目（由中间有一个圆点的小方块表示），可以看到对应的变量被添加到测试与调试的观察列表中。



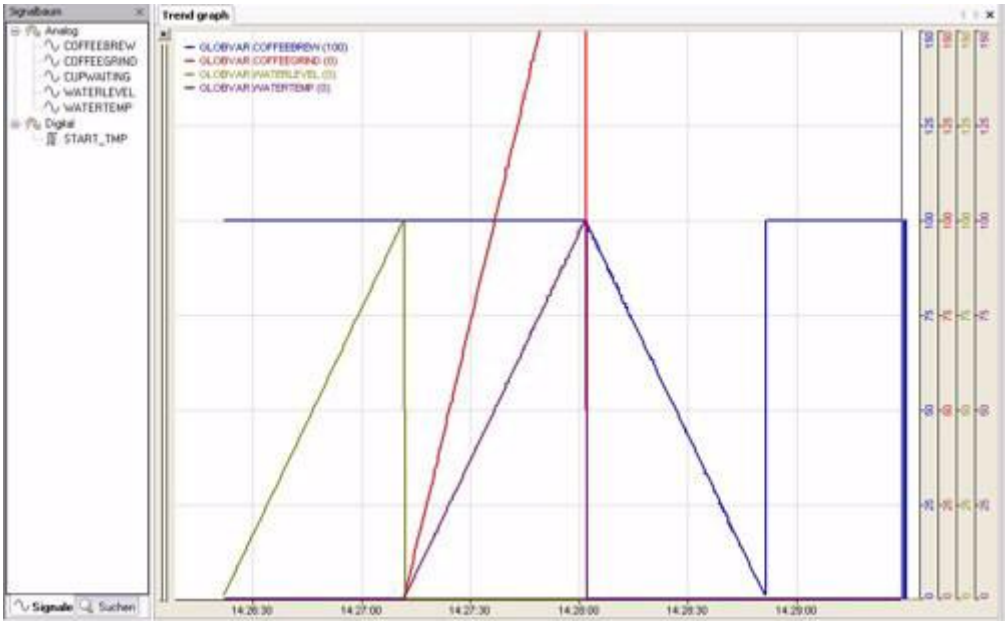
回到 SmartSIM，改变输入来看看观察列表有什么变化。

OpenPCS 支持 在线编辑 ，更多信息请参考用户手册中的在线编辑章节。

注意：如果在你设置的断点处 SmartSIM 没有停止，你可能没有正确设置优化设置。请确保你的资源设定为仅大小 。

1.5.5 Control Data Analyzer

控制数据分析器给编程人员观察变量通过时间变化的可能性。分析器可以通过**观看->控制数据分析器**打开。这个选项在整个上线过程中有效。如下截图显示了煮咖啡的全过程。先加水(黄线)，然后加热(红线)。同时磨咖啡豆。当两个步骤都停止，开始煮咖啡。



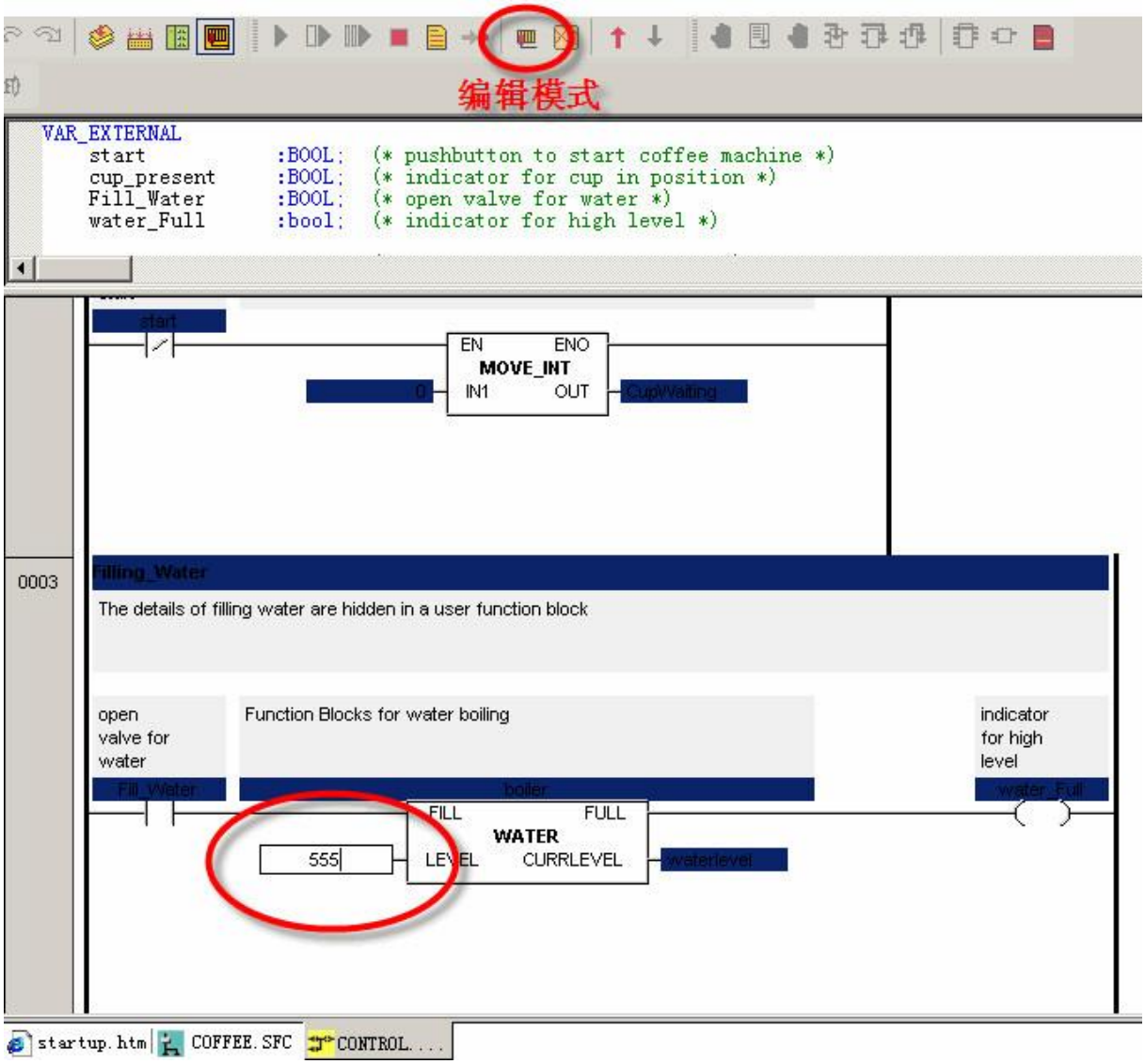
1.5.6 在线编辑

在线编辑（或者在线变更）是一种特性，它使得程序的变化能够直接被应用于 PLC，而不需要重新运行。要使用在线编辑，必须先执行以下几步。

程序必须通过编译并在 PLC 上运行。资源在编辑窗口中被打开。

编辑器可以通过 ***PLC->观察/编辑*** 或者工具栏里对应的按钮，在观察模式（绿色符号）和编辑模式（红色符号）之间切换。 

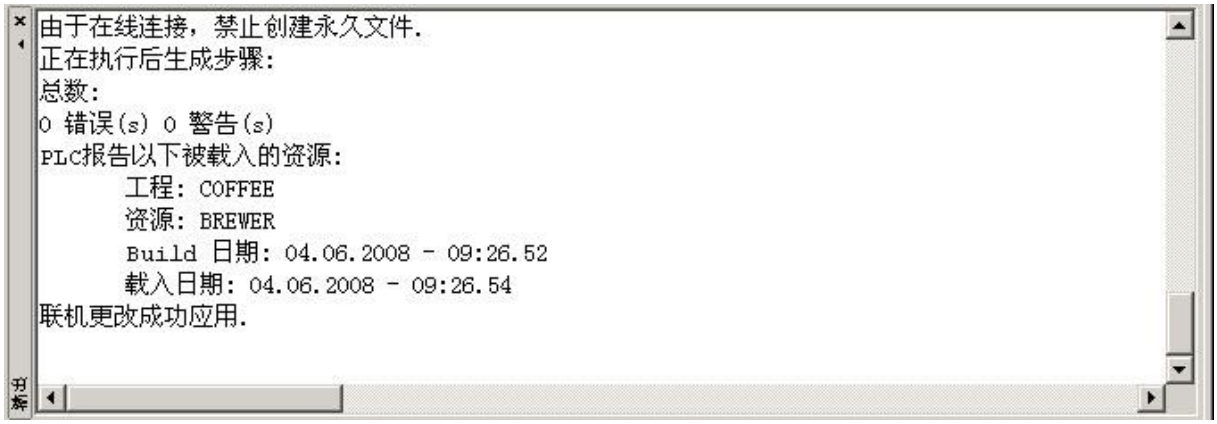
进行了修改之后再次通过 ***PLC->观察/编辑*** 来关掉编辑模式。下图显示的变化将期望水位从 100 改为 555。



OpenPCS 会显示对话框来确定和应用这些改变。

如果改变被接受，OpenPCS 会在不中断循环的情况下，重新编译必要的单元，并把它们重新下载到 PLC。这些变化会在下次循环中产生作用。

升级完成后，OpenPCS 会在输出窗口中显示一条信息。



注意：这些改变对 PLC 并不是持久的，因此你需要一个可选的目标系统特性，叫做保存系统。

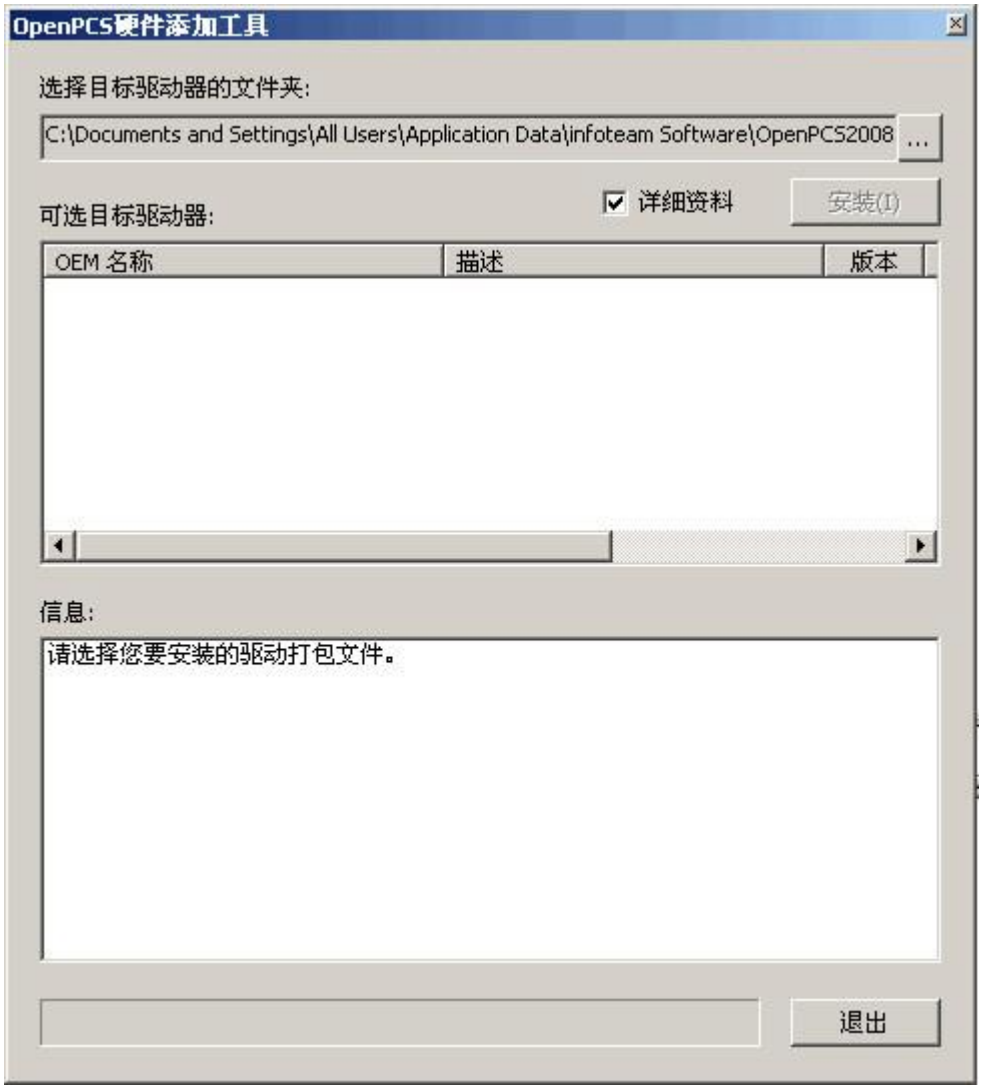
1.6 更多

1.6.1 添加硬件支持

为了其硬件/控制器的安装支持，所有自动化网络成员公司都在提供目标驱动(*.cab files)。在一次安装中，可以为不同的控制器安装不同制造商的不同的目标驱动。

要安装一个目标驱动，选择***其他->工具->安装驱动...***

首先，为目标驱动指明路径（例如对于 CD-ROM，路径为 X:\ ），然后选择您想要安装的驱动，之后点 安装 。



目标驱动可以包括硬件定义，通信驱动，帮助文件，类库和（对于工程，资源，程序文件等的）模板等等。请参看目标驱动提供的的信息，或者是你的 PLC 制造商提供的信息.

1.6.2 模版

OpenPCS 2008 支持文件和项目模版，以减少为特殊任务创建解决方案时的耗费。用于特定 PLC 的经过优化的模版由 PLC 的生产商作为目标驱动器的一部分提供。模版可以应用于资源、任务、声明、工程以及程序文件等。

注意：由一个生产商提供的模版可能不适用于其他生产商的硬件。

1.6.3 导入/导出 XML

OpenPCS 为 IEC 61131-3 提供 PLCOpen 标准 XML-导入/导出。XML-导入/导出能够把整个工程导出为一个单独的 XML 文件，或者将一个 XML 文件导入为工程。

选择 ***工程->XML-导出*** 来导出一个项目。在对话框中选择保存 XML 文件的文件夹。该 XML 文件的名字时不可选的，它的名字应该和工程的名字相同。以下类型的文件可以被导出：

1. ST
2. IL
3. SFC
4. 全局变量的声明文件
5. 直接全局变量的声明文件

以下类型的文件目前还无法导出：

1. 梯形图
2. FBD
3. CFC
4. 包含用户自定义数据类型的文件
5. OPC 变量文件
6. 子目录
7. 子目录中的文件
8. 非 OpenPCS 的文件（例如 PDF、DOC、ZIP 等）

选择***工程->XML-导入***来导入一个文件。选择要导入的 XML 文件以及创建新工程的目录。

如果想要备份您的工程，我们不推荐使用 XML 导出功能，因为 OpenPCS 有自己的备份功能。选择选择***工程->备份*** 来生产一个备份。选择在哪里保存备份文件（.BAK）。您可以选择选择选择***工程->恢复*** ，并选择.BAK 文件来恢复一个工程。

OpenPCS 备份功能可以保存工程中的所有文件。

注意：用早于 5.2.0 版本的 OpenPCS 生产的 SFC 文件必须用新的 OpenPCS 重新保存，因为从 5.2.0 版本起 OpenPCS 开始为 SFC 文件使用 XML 文件格式。

1.6.4 关于这个手册

这个手册包含 4 个主要的章节：

第一章您已经阅读了，它简单介绍了 OpenPCS 最普遍的特征。如果您先前还没用过 OpenPCS，一定要阅读这一章。

第二章详细介绍所有的 OpenPCS 工具。一开始先是介绍各个不同的编辑器，然后是编译器和不可见的工具。阅读这章可以通过工具的功能获得一般认识。

第三章深入介绍一些高级主题，大部分影响到不止一个工具。阅读这一章可以获得背景信息以及深入理解如何最好地使用 OpenPCS。

第四章是参考，所有的关键字、函数、编译信息、警告和许多款项按照字母顺序组织在这里。使用这一章可以快速地查找信息。

索引将帮助您找到所有的信息。用户手册□□不管是打印出来的还是电子 PDF 格式，它们的内容跟在线帮助是一样的。因此如果您在用户手册里找不到您所需的信息，可以尝试使用帮助文件的搜索功能。

1.6.5 更多信息

这个用户手册只适用于 OpenPCS 软件，不是培训指南。如果您需要更多信息，我们推荐查阅安装的使用手册以及：

1. 《工业自动化系统编程（Programming Industrial Automation Systems）》，作者 Karl-Heinz John 和 Michael Tiegelkamp，德文版 ISBN 3-540-66445-9，英文版 ISBN 3-540-67752-6 以及中文版。

2. OpenPCS 的生产商 infoteam Software，可以提供关于 OpenPCS、IEC61131-3、运动控制（Motion Control）、实时编程（Real-Time programming）的现场培训课程和相关期刊。联系：info@infoteam.de 以咨询价格和货源提供。

打印日期：2008 年 7 月 1 日

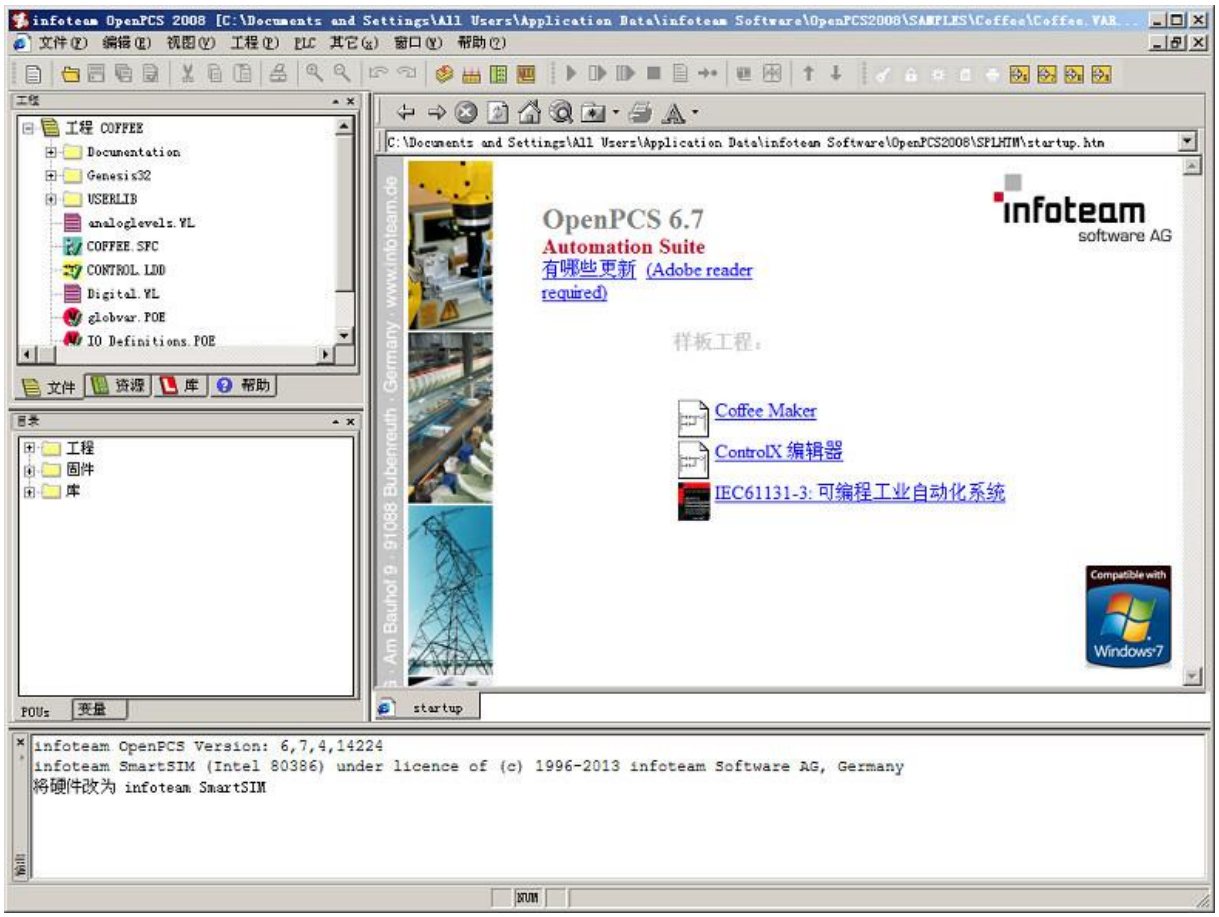
更多阅读及下载，请登录公司主页 http://www.infoteam.de。

2 OpenPCS 工具

2.1 OpenPCS 框架

2.1.1 OpenPCS 框架：介绍

OpenPCS 框架容纳了大部分 OpenPCS 的工具。OpenPCS 框架一般看上去跟下面的相似：



工程显示在左边的工程浏览器中，编辑器窗格定位在中间。大部分编辑器使用分屏技术，声明放在上层窗格，指令编辑在下层窗格。对于所有的编程语言，声明看上去都是一样的，指令却变化很大。OpenPCS 框架可以同时处理许多文件。诊断信息显示在底部的输出窗口。

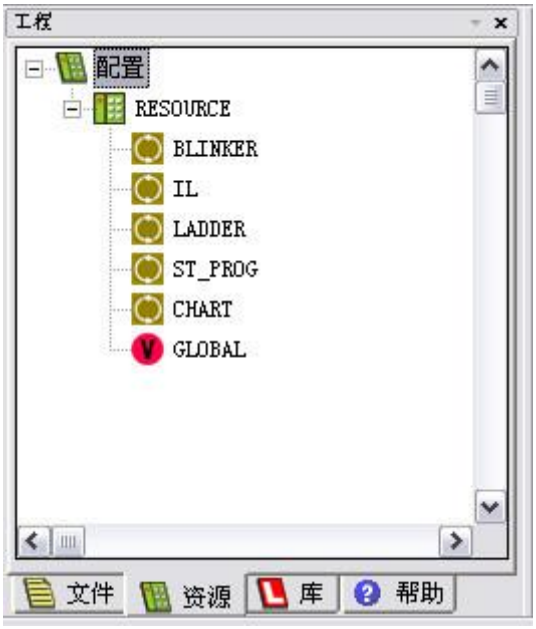
2.1.2 输出窗口

输出窗口位于 OpenPCS 框架的底部，用于显示诊断消息。

2.2 浏览器

2.2.1 浏览器：介绍

工程浏览器用于 OpenPCS 的文件管理。使用浏览器，您将您的工作组织成文件和工程。从浏览器中，您将创建和编辑文件以及编译、下载和监测应用程序。

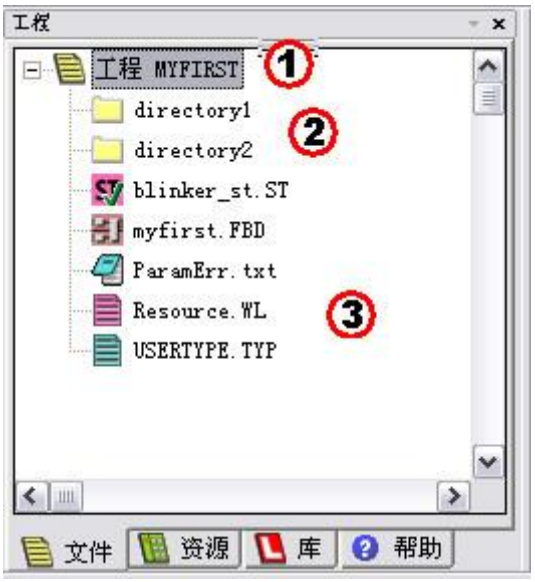


浏览器用户界面由以下不同的窗口（窗格）组成：

1. 文件窗格
2. 资源窗格
3. OPC - I/O-窗格(可选)
4. 库窗格
5. 帮助窗格

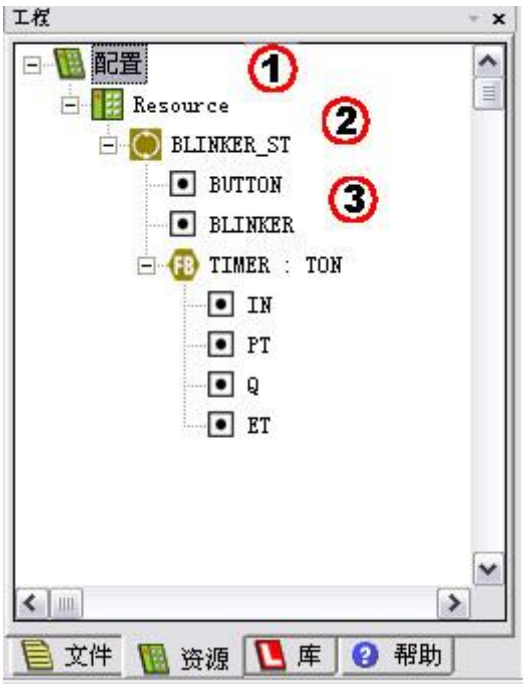
2.2.2 浏览器总览

2.2.2.1 文件窗格



文件窗格包含一个所有资源文件的目录树，这些资源文件收集在当前的工程下(1)。这些文件都是您自己使用 OpenPCS 的一种编辑器或者其它应用程序编写的。当前工程路径的所有目录(2)和文件(3)在这里都显示出来。

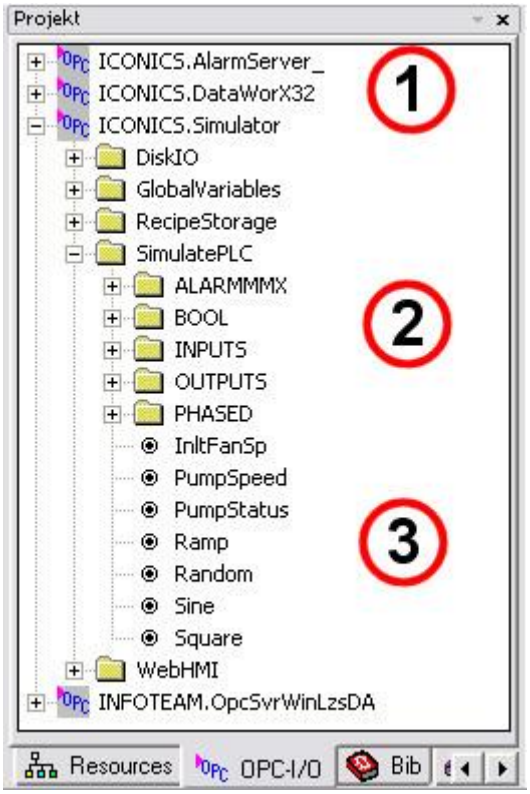
2.2.2.2 资源窗格



资源窗格包含名为 配置 的实例树。它将显示：您的控制器作为资源 (1)、运行在这些控制器上的任务(2)、函数和功能块实例以及在这些控制器中定义的所有变量(3)。激活资源用绿色按钮显示。

在实例树中，只有文件和定义在文件窗格的工程的连接：任务对应 PROGRAM 类型的 POU，全局变量对应全局声明文件等等。

OPC-I/O-窗格



OPC-I/O 窗格包含本地有用的 OPC-DA 地址空间树。

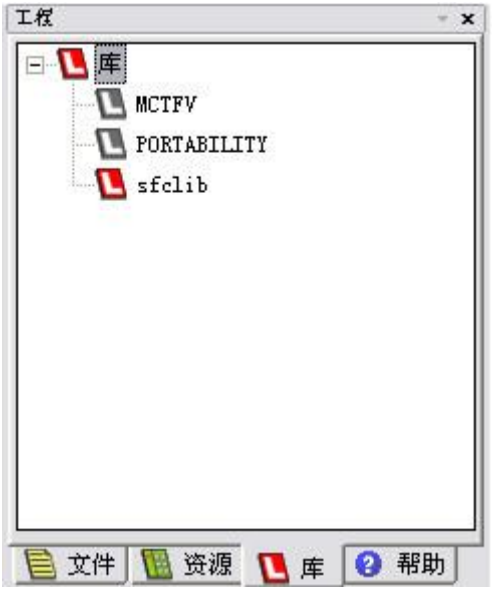
在根部（1），它列出了所有当前注册在运行 PC 上的 OPC-DA 服务器。在根部以下，您能发现好几层内嵌的 OPC 文件夹（2），由被选择的 OPC-DA 服务器的地址空间构成。最后，在这个树的末端，有 OPC-Tags（3）表示 OPC-DA 服务器的 I/O 值。

既然使用 OPC-I/O 窗格只对支持 OPC-I/O 的目标有意义，您可以通过 ***其它->浏览器选项*** 对话框来打开或者关闭这个窗格的显示。

注意：目前不支持非本地 OPC 服务器。Infoteam 的 OPC 服务器（infoteam.PadtOpcSvrDA）在这个列表中不可用。

2.2.2.3 库窗格

库窗格包含一个树，这个树是工程中所有已安装的库的目录。您可以通过***文件 -> 库 ->安装新的***□来安装新库。



选择*文件 ->库 -> 在当前工程中使用*可以使用工程中的一个库。当前正在使用的库会用一个红色符号来显示。

2.2.2.4 帮助窗格

帮助窗格包含帮助主题。

2.2.3 工程

2.2.3.1 创建新工程

如果您已经打开 OpenPCS，您可以开始工作了。第一步是新工程的创建。选择*文件 ->工程 ->新建* 或者按下工具条中相应的按钮。

请注意：

1. OpenPCS 工程的名称不能包含空(空格)字符或者特殊字符。另外，为了容易更新，推荐把您的应用程序与 OpenPCS 分开存储。举个例子，C:\PROJECTS 就是一个用于存储工程的不错的目录。

2. 一个名字跟您的工程一模一样的子目录会在您键入的位置自动创建。这个目录包含工程中的所有文件。

2.2.3.2 检查工程一致性

如果您的工程有任何问题，OpenPCS 为您提供了***工程->检查*** 对话框。它自动检查工程文件中的不一致性，并且告诉您有哪些不一致。您可以打开它们。工程一致性状态窗口中有 4 个按钮：

1. **修复**：修复一个 POU 名称的不一致
2. **打开**：打开在窗口中选中的单个文件
3. **打开所有**：打开窗口中的所有文件
4. **关闭**：关闭工程一致性状态窗口

造成工程文件不一致性的主要原因，一方面是旧 POE 文件（即 POE 比 ST 文件旧），另一方面是缺少 POE 文件（关闭窗口，重新选择***工程->检查*** 可以重新检查此类不一致）。

打开、人工修改语法并保存可能可以修复它们。

注意：这个工具主要用于检查不一致性而不是检查语法错误。

2.2.3.3 打开工程

您可以通过三种方式打开一个工程：

在 文件 菜单中：在 最近打开的工程 列表中，您要找的文件可能包含在这里。

通过工具条：点击按钮 ***打开工程***

通过菜单：在主菜单中点击 ***文件*** ->***工程*** ->***打开...***。

在对话框或者在文件夹选择所要的工程，工程文件有后缀名 .var 。

2.2.3.4 导入/导出

用 ***文件*** ->***工程*** ->***导出*** 可以导出您的工程文件到一个 PLC 标准工程，同样导入使用 ***文件*** ->***工程*** ->***导入***.

注意： 当前版本的 OpenPCS 仅支持导出/导入 POE, ST, SFC 和 MAK 文件以及 ENV 文件夹.

2.2.3.5 在工程中查找

当前工程中嵌入的所有文件，都可以通过***工程->在文件中查找*** 来进行查找。查找结果在输出窗口中列出。

双击其中一个结果，就可以在编辑器中打开打开文档的相应位置。

2.2.3.6 更新工程信息

选择***工程->更新工程信息***可以刷新工程的信息并重新写入工程。也就是说，工程原型被重新读入，库被刷新。

2.2.4 文件

2.2.4.1 建立新文件

在 OpenPCS 框架内用 OpenPCS 建立新文件。选择***文件->新建*** 可以看到许多选择：

POU 用于程序，功能块和函数，以及 IEC61131-3 定义的基本代码块。对于每一个基本代码块，您可以在 OpenPCS 的几种编程语言中选择定义，要尽可能的恰当。

声明用于建立全局资源，直接全局，类型和 OPC 变量声明文件。

资源用于建立新的资源。

在**工程**中有样板工程包含示例配置。

在**其他**中，可以建立文件夹，GWX 文件和观察列表。

注意；可能包含其他样板（或结构），根据不同 OEM 生产商。

2.2.4.2 文件操作

子菜单 **文件->文件** 中可以进行下列操作

1. 移动文件到另一文件夹
2. 复制文件
3. 重命名一个文件
4. 从另一工程或位置导入 一个文件
5. 导出文件到另一个工程或位置

注意： 以上操作针对在浏览器中选中的文件.

2.2.5 资源和任务

2.2.5.1 资源：介绍

通常，一个资源就等同于一个 PLC 或一个微控制器。一个资源定义包括用于鉴别的**名字**，**硬件描述**，也就是 OpenPCS 所使用的 PLC 的属性；还有**连接名**，也就是关于 OpenPCS 和控制系统之间通信类型的信息。

一个资源保留了运行于控制系统的一个任务列表。

2.2.5.2 建立资源

每当创建一个新工程，OpenPCS 就定义一个资源。如果要建立额外的资源，点击***文件 ->新建...***，在下面的对话框中进入 其它 ，并选择 资源 。

点击 OK，一个新的资源就会出现在资源窗格中。

2.2.5.3 编辑资源

要编辑一个资源，右键点击它，在弹出的相关菜单中选择 属性 。

这会打开一个对话框，您可以改变下面的属性：

在 硬件模式 下面，选择您使用的控制器的相应配置文件。您所使用的控制器的制造商应该提供这个配置文件。要使用 Windows 仿真 SmartSIM，使用 SmartSIM 。

在 网络连接 下面，选择通信连接去连接您的目标。使用 ***PLC->连接***定义新的连接或者查看或者修改定义的连接属性。网络连接预选择为 仿真 ，用于与 OpenPCS 的 PLC 仿真器工作。

选中 上载 ，将您应用程序的资源打包到目标。如果您想在调试的末尾保存工程到控制器以便其他服务人员在后面使用，这样做是有帮助的。

生成映射文件 ：生成代码之后，会在您寻找连接器信息的地方生成 3 个文本文件。这些文件将被保存在资源目录下，分别名为 Pcedata.txt， PceVars.txt 和 PceSegs.txt 。一些别的 OpenPCS 的功能（GetVarAddr）需要这个功能去开启，因此在没有一个好的理由之前，最好不要禁用它。

对于最优化设置的描述，看高级主题的最优化设置。

2.2.5.4 增加任务

通常，一个任务就是程序加上如何执行这个程序的信息。任务的定义包括名称、任务执行的信息和在这个任务中需要执行的 PROGRAM 类型的 POU。

要增加一个任务，首先标记您要创建任务的程序，选择 ***PLC->连接到资源***。在增加了任务之后，您可以通过在资源窗格内双击它来改变任务说明。

注意任务的名称取决于程序的名称，并且是不能改变的。要完成任务定义，您必须详细指明信息，这个任务是如何被执行的：循环、定时器控制还是中断控制，任务类型，优先权和定时器控制这个任务的执行以及与其它任务的合作。要完成这个，右键单击鼠标选择 属性 。更多的信息看多任务。

2.2.5.5 激活资源

对于每一个 OpenPCS 工程，可能会有许多 资源 ，要想最好地利用多资源可以看高级主题部分。然而，为了使 OpenPCS 更加用户友好化和更容易的使用，任何时候总是正好有一个激活的资源。在浏览器中，它会显示为一个绿色的图标。

许多用户命令－如编译、上线、下载等等，隐含地在使用 激活的资源 。因此即使在一个工程中有许多资源，您不必规定哪个资源去使用这些命令。如果您想使用一个不同于当前激活资源的资源，在这个资源上右击鼠标然后从弹出的相关菜单中选择 设置激活 。

2.2.6 IPC - I/O

2.2.6.1 OPC - I/O: 介绍

使用 OPC-I/O 浏览器窗格，您可以浏览本地可用的 OPC-DA 地址空间树并且利用它来指派任何一个这个树的末端项给一个在任务编辑器中定义的全局变量。

这样在您的 PLC 程序中您能把任意一个由 OPC-DA 服务器提供的变量当作一个全局输入或者输出变量来使用。

2.2.6.2 关于 OPC

OPC 的意思是过程控制中对象的链接和嵌入，是一系列由 OPC 组织创立的标准规格。

目前 OpenPCS 支持 ***Data Access Specification (OPC-DA) Version 2.0***。
需要更多关于 OPC 的信息，请咨询 OPC 组织的网页 http://www.opcfoundation.org

2.2.7 编译器

2.2.7.1 组建激活的资源

可以仅仅组建自上一次修改后发生变化的部分资源。通过 ***PLC->生成当前资源***来调用。

OpenPCS 自动组建上线时所需要的任何东西，但不时地去重新编译是一个好习惯，这样程序可以尽可能早地去检测错误。

2.2.7.2 重新组建激活的资源

要在一个激活的资源里重建所有的任务，选择 ***PLC->重新生成当前资源。***这会编译资源中所有的任务。

2.2.7.3 重新组建所有的资源

像重新组建激活的资源一样，不过会重建所有的（激活的和非激活的）资源。

2.2.8 在线

2.2.8.1 上线

要进入在线模式，双击您想上线的资源，选择 ***PLC->在线***，或者按下工具条中的 在线 按钮。重复这个可以再次下线。

2.2.8.2 下载

在任何需要下载的时候，OpenPCS 会自动的给我们提示。只要您愿意，您可以通过使用 ***PLC-> PC->PLC (下载)***在任何时候调用一个下载。

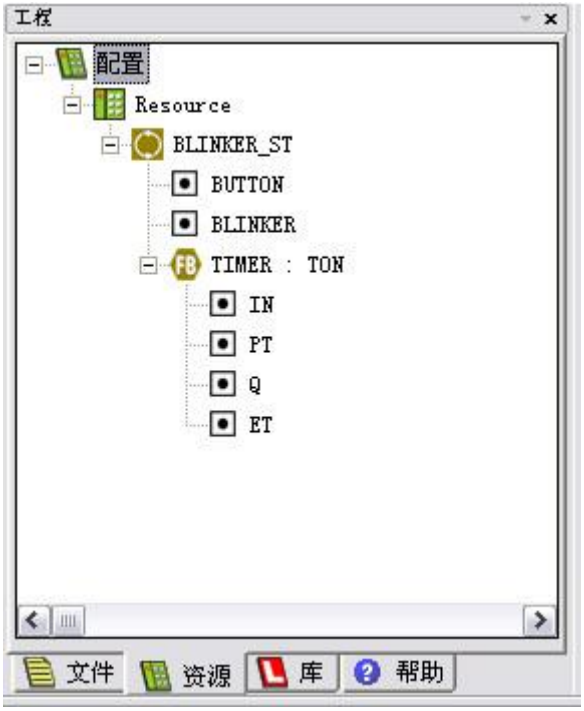
只要当前工程已经在资源里，您可以实时下载工程（无须中断 PLC）。但这个下载不是永久的。这大体上对应在线编辑的功能。

如果您的 PLC 支持保存系统的功能，我们可以提供永久下载工程而无须中断 PLC 的能力。

注意：通常的下载模式（中断 PLC）是永久的。

2.2.8.3 观察变量

将变量加入**测试与调试**的观察列表，打开应用程序的资源树然后双击任何一个变量。



2.2.8.4 启动在线编辑器

上线会在在线模式下打开所有的 POU。要在线模式下启动应用程序的代码块的编辑器，打开资源树，定位您要监控的实例然后双击它。

OpenPCS 支持 在线编辑 ，请参看用户手册中的 在线编辑 。

注意：

为了避免混乱，强烈推荐在打开在线编辑器之前关闭所有的不在线的编辑部分。

不要把代码的实例 (位于浏览器的 配置 下)和功能块的源代码(位于浏览器的 工程文件 下面)相混淆。

2.2.8.5 硬件信息

这个菜单只在在线模式下有效。您可以得到所使用硬件的信息。标记激活的资源然后选择菜单项 ***PLC ->PLC 信息***。

资源信息

这个菜单只在在线模式下有效。工程名、资源名、版本号(内部自建并分配到具体的编译)将被显示。

通过标记资源然后选择菜单项 ***PLC ->资源信息***，您可以显示资源的信息。

2.2.8.6 上载

OpenPCS 支持从控制器上上载工程至 PC。因此在更新 PLC 时工程源代码并不是必须的，您可以上载该工程。

为了使能这个特性，在编译和下载一个资源时必须在资源属性中选中 能上载 选择框，如下图所示：

上载工程时，要确定资源的属性如上所示设置好，并且没有连接到任何 PLC，然后使用 ***PLC***-> ***PC***<= ***PLC***。

现在您需要选择与 PLC 的连接，以上载工程。

之后您会被询问保存工程的路径（请确保该工程尚不存在）：

被上载的工程会自动打开。

2.2.8.7 清除

只有在在线模式下可以选择菜单 **PLC->清除** 或点击工具栏的相应按钮来从 PLC 清除整个程序。

系统确切的反应取决于 您的 PLC 的 OEM。若想了解更多，请向其询问□

2.2.9 其他浏览器特征

2.2.9.1 资源全局变量

在 OpenPCS，有两种全局资源变量：

全局变量：这些变量没有硬件地址，比如，用于中间结果。

直接全局变量：这些变量有直接硬件地址和 IO 声明。它们代表硬件的接口。

要建立一个有全局变量的新文件，选择***文件*** -> ***新建*** ->***声明***->***全局***或者***文件*** -> ***新建*** -> ***声明*** ->***直接全局***。编辑这些文件，然后与要使用它们的资源相链接。

2.2.9.2 类型定义

在默认情况下对每个 OpenPCS 工程，用户定义数据类型(usertype.typ)都存在一个文件中。如果需要自己的数据类型，编辑这个文件或创建您自己单独的文件。为了任何的资源都能使用那些数据类型，将文件加到各自的资源中。

2.2.9.3 增加文件

OpenPCS 允许增加任何类型的文件到 OpenPCS 工程。使用***文件*** ->***导入***□然后选择您要加入的文件。除了使用 OpenPCS 编辑器(IL，LD，FBD，ST，CFC，SFC)编写的文件外，可以导入象资源一样的类型定义和类型声明文件。此外，在一个工程里注册文件是行得通的，甚至它们是由其它程序创建的，比如：Microsoft Word，Microsoft Excel，Microsoft Project，AutoCAD。

在弹出的菜单中选择想要的文件类型，然后打开相应的目录。那里您可以选择想要拷贝的文件。按住鼠标左键和 Shift 或 Ctrl 键，可以多重选择。这个文件将会被拷贝到浏览器的当前目录中并且可以通过双击它进行编辑它。

2.2.9.4 浏览器选项

通过***其它->选项***，设置浏览器选项：

常规：

仅显示 OpenPCS 文件类型 ：如果检查栏被标记，浏览器只显示 OpenPCS 文件类型。除了列在附加文件类型 的文件类型外，所有其它的文件是隐藏的。

不显示 SFC 变量 如果检查栏被标记，所有首字符是 _ 的变量都隐藏在资源树中。

不显示注册标签 如果检查栏被标记，浏览器面板上的标题只有图标而无标题。

其他设置：

在 OPC 中显示浏览器面板 这个检查栏用来隐藏或显示 OPC 浏览器窗格

编译时升级和激活 OPC 标签数据库 这个检查栏用于自动生成 OPC 数据库。

在 OpenPCS 框架中打开源 GrafWorX 文件 这个检查栏用于允许在 OpenPCS 框架中打开 GWX 文件。

保存观察列表 这个复选框用作激活下线时自动保存观察变量列表功能。

编译器：

编译器输出级别 通过这个下拉菜单可以选择编译器输出错误的级别（0 为最详细）。

警告当可用的分配给资源的 PLC 内存小于给定的百分比 这个复选框可以被用作激活给当前资源的 PLC 内存不足时的警告。可用内存空间的最小值可用百分比设定。

注意：这个选项需要硬件支持。

警告当可用最大区块大小小于给定的百分比 这个复选框可以被用作激活当前资源过大区块警告。可用内存空间的最小值可用百分比设定。

2.2.9.5 CFC/FBD 选项

通过***其他->选项->CFC 编辑器***可以设置 CFC 和 FBD 编辑器的外观选项。更改后的效果只有当新打开 CFC 或 FBD 文件后才能显现出来。

块宽度百分比：改变一个功能块的宽度。100 相应于默认宽度。宽度值可以在 1 至 1000 间设置。

边缘宽度百分比：改变边缘的宽度。100 相应于默认宽度。宽度值可以在 1 至 1000 间设置。

2.2.9.6 设置字体和颜色

在 **其他->字体/颜色** 对话框中，可以设置 IL, ST, SFC, 梯形图 和 FBD 文本编辑器的外观.

注意：当前 OpenPCS 版本仅提供设置字体类型和大小

2.2.9.7 自定义工具

浏览器在工具栏处提供了一个按钮用于启动一个 OEM 特制工具。

OpenPCS 的 OEM 可以通过这个按钮配置 OpenPCS 启动任何特殊的工具。默认为启动许可证编辑器。

2.2.9.8 从工程中排除

子目录可以被允许排除在工程之外。工程中被排除的子目录会被忽略。您将无法在被排除的目录内导航。这个功能可以用于排除一个"subversion"目录或者一个文档目录。

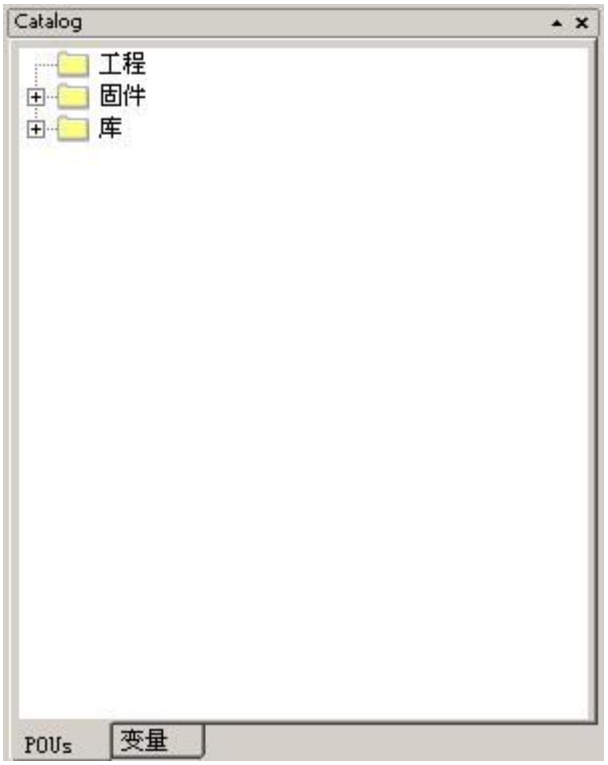
将一个目录从工程中排除，选择在对应目录的右键菜单选择"从工程中排除"。

2.3 目录

2.3.1 目录

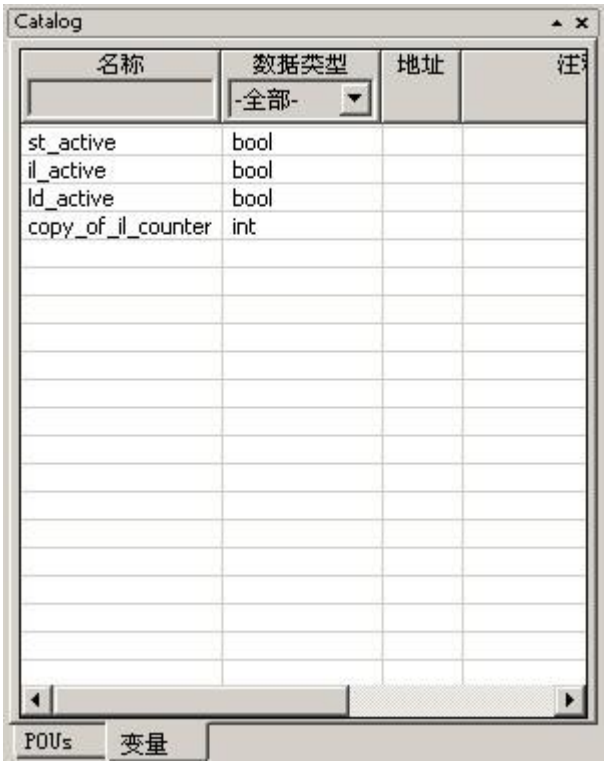
目录是一个在您的程序中插入功能块的工具。您可以在项目浏览器下面找到它，如果没有，请在菜单***视图->目录***中选择。有了目录，您就可以通过鼠标拖拽在您的程序中加入功能块了。

通过使用目录，您不必再通过指定名称或用菜单来添加功能块。

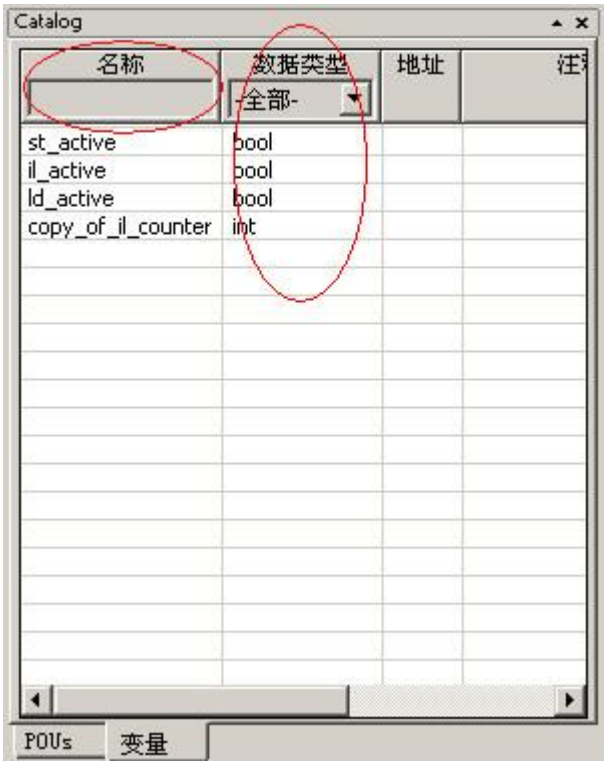


2.3.2 变量目录

变量目录是目录的一部分，所有的全局变量被显示在变量格中。您可以看到变量的名字，数据类型，地址，注释（如果有）和它们的作用域。 此时只有 CFC 编辑器支持使用的标记。



变量格使您可以通过拖曳鼠标在您的程序中加入或者过滤全局变量。如果使用拖曳，键盘输入的步骤将会被省略掉。您也可以过滤变量名、数据类型及作用域以观察现有哪些变量和它们的数据类型。目前只有 CFC 编辑器支持使用标记（flag）



变量格子使您能够通过拖放鼠标在您的程序中添加和过滤全局变量。当使用鼠标拖放时，键盘输入的步骤就可以省略了。而过滤全局变量则意味着您可以过滤全局变量的名称、数据类型和作用域而无须翻阅它们的定义。

仅仅输入变量名您就可以看到所有适合您输入的变量。您也可以使用星号（比如在变量名框中键入 *A* 您就会得到所有名称中含有 A 的变量）和组合过滤（先输入名称再改变其数据类型）。

如果您创建新的全局变量，它们会在存储于全局变量文件之中后自动的显示出来。在变量格子中使用鼠标右键单击并选择刷新可以更新变量格子。

2.4 声明编辑器

2.4.1 声明编辑器：介绍

声明编辑器是在 OpenPCS 框架之中的。这里键入由 IEC1131 定义的声明。

IEC61131-3 要求所有数据对象必须当作变量被声明。提供一组不同的声明部分定义不同域的范围。IEC61131-3 带有一组预定义的数据类型，即所谓的基本数据类型。还有一些是用户定义的，即所谓的派生数据类型，包括结构、数组和枚举。

对于大部分变量，由编译器分配存储，没有任何程序员的参与。对于输入、输出、标志和更多的潜在的变量类型，程序员可以使用直接表示变量具体分配一个内存位置。

声明输入在由 IEC61131-3 定义的文本框内。

2.4.2 声明部分

变量在不同部分进行声明，即所谓的声明块。一个声明块起始于一个关键字并且以 END_VAR 结束(比如，VAR_GLOBAL□END_VAR)。

VAR_INPUT：如果一个变量块只需读入一个 POU，您必须声明这个变量为输入变量。因此不允许在这个 POU 修改这个变量。一个输入变量可以用于函数和功能块的参数传递。

VAR_IN_OUT：一个输入输出变量通过功能块在相同名字下存取。在参数传递期间通过功能块调用，变量获得一个传递变量和它的内存的地址参考(指针)。因为写操作对输入输出变量的内容有直接的影响，不允许使用写保护类型用于作为输入变量或具有 CONSTANT 属性的变量的传递变量。

VAR_OUTPUT：输出变量在功能块中声明，用于返回值。调用 POU 可以访问它们。

VAR_GLOBAL：如果一个变量在一个 POU 中必须有效并且所在的功能块被这个 POU 调用，这个变量在 POU 程序中应该声明为全局变量。要在所有功能块中使用这个变量，必须把这个变量声明为外部变量（VAR_EXTERNAL）。

VAR_EXTERNAL：如果一个全局变量将在一个功能块中使用，在这个功能块里必须把这个变量声明为外部变量。

VAR：一个本地变量只在已声明的 POU 内部有效。本地变量的声明可以通过属性 RETAIN ，或 CONSTANT ，或地址来补充。

TYPE：关键字 TYPE 用于用户定义数据类型的声明，这个数据类型是在 POU 类型 程序 和 功能块 的本地范围内的使用，或者是类型定义的全局范围内使用。

根据 POU 的不同类型相应的变量可以使用：

一个程序类型的 POU 可以使用 TYPE，Local，Global 和 External。

一个功能块类型的 POU 可以包含 Type，Input，Output，In_Out，Local 和 External

一个函数类型的 POU 可以使用 Type，Input 和 Local。

CONSTANT 可以作为关键字(比如，VAR_GLIBAL CONSTANT)的修饰符去声明这个部分的所有变量声明，这些变量不会被应用程序所修改。如果这样一个变量在上下文中会被改变，编译器会发布一个警告。

RETAIN 可以作为关键字(比如，VAR_RETAIN)的修改量声明这个部分的所有变量为保持变量，也就是说这些变量在热启动或温启动中不被重新初始化。如果目标系统支持非失忆内存，在掉电情况下，可以保持变量的值。

OPC 允许用户标示一个专有的已经位于 OpenPCS 的编辑器的变量声明部分的变量，使其成为变量表的一部分。 变量限定 OPC 可以在以下的声明区域中被使用：

1. VAR （局部变量）
2. VAR GLOBAL （资源和任务变量）
3. VAR INPUT, VAR OUTPUT, VAR IN_OUT （输入和输出变量）
4. VAR EXTERNAL （外部变量）

这里这些变量可以是**程序**，**功能块**或**函数**的一部分。以下的变量数据类型可以被支持：

USINT，UINT，UDINT，SINT，INT，DIN，BOOL，BYTE，WORD，DWORD，REAL，LREAL，STRUCT，ARRAY

功能块的实例声明不被允许，如：InstanceFB1 ：FB1;

变量限定 OPC 可以和其他限定器 CONSTANT 和 RETAIN 一起混合使用，这样就允许了类似如下的声明：

VAR CONSTANT OPC

Var1 ：INT;

END VAR

下列句法可以被支持：

1. CONSTANT OPC
2. OPC CONSTANT
3. RETAIN OPC
4. OPC RETAIN

因为 RETAIN 和 CONSTANT 不能混用， 所以， 类似于 CONSTANT RETAIN OPC 这样的顺序是不允许的。

变量表在运行系统中提供了符号的信息，它在 OPC 服务器 SmartLINK 中被用于提供 OPC 标签。

2.4.3 声明行的结构

一个声明行有下面的形式，可选部分写在方括号内[ ]，表达式写在尖括号内< >：

<变量名> [AT <地址]>]: <类型>[:=<初始值>]: [(*<注释>*)]

首先给出变量名，紧跟一个冒号，冒号后面是类型，最后通过属性 AT 引入硬件地址。如果变量在一开始需要有确定的值，它跟在 := 后面。一行总是以分号结束。行可以加注释，放在 (* 和 *) 之间。

举例：

Exgpvariable1 AT%I0.0:BOOL;(*地址%I0.0 的 BOOL 型变量*)

Expvariable2：BOOL:=TRUE;(*初始值为 TRUE 的 BOOL 型变量*)

一个例外是没有变量名的直接地址。（这些变量通过地址来引用）：

AT<地址>:<类型>[:=<初始值>];[(*<注释>*)]

在这种情况下变量名是省略的，因此地址声明是必须的。

举例：

AT%I0.0:BOOL (*地址%I0.0 是一个 BOOL 类型数据)

为了清晰，最好避免寻址的第二种方式，因为变量的意义跟变量名有很大关系。如果其他人要阅读或编辑这个 POU，这一点是很重要的。

一些例子：

无初始值的变量：InterMedSum：INT;

带初始值的变量：Pieces：INT := 5;

无名无初始值的直接表示的变量：AT %Q0.0：BOOL;

有名无初始值的直接表示的变量：Valve AT %Q0.2：BOOL;

功能块例子：Counter1：CTU;

注解：

1. 初始值只可以当作字符来处理。在声明阶段，使用其它变量初始化变量是不可能的。
2. 变量名的有效长度是 64 个字符。
3. 无法在进程映像中初始化变量。

2.4.4 基本数据类型

关键字	名字	范围	位数
BOOL	布尔型	0(FALSE),1(TRUE)	1 或 8
SINT	短整型	-128 到+127	8
INT	整型	-32768 到+32767	16
DINT	双整型	-2.147.483.648 到 +2.147.483.647	32
UINT	无符号整型	0 到 65535	16
UDINT	无符号双整型	0 到 4.294.967.295	32
REAL	实型	+/-3.4E+/-38	32
LREAL	长实型	+/-1.8E+/-308	
TIME	时间段		4
DATE	日，月，年	0001-01-01 至 11795222-01-20	32

TIME_OF_DAY	时间点	00:00:00:000 至 23:59:59:999	32
DATE_AND_TIME	日期和时间		64
STRING	字符串		字符长度加 2 字节
WSTRING	双字符字符串		
BYTE	8 位序列		8
WORD	16 位序列		16
DWORD	32 位序列		32

请同时参看 常数。

2.4.5 直接表示的变量

直接表示的变量是指那些被映射到一定输入、输出或程序员具体指定内存地址的变量。它使用关键字 AT 来声明，地址由百分符%开始的字符串指定。

举例：直接表示的变量

使用和不使用符号名的直接表示的变量声明

```
PROGRAM poe4

VAR

AT%I0.0:BOOL;

  Eingang_1 AT%I0.1:BOOL;
Ergebnis:BOOL;

END_VAR

LD Eingang_1

AND%I0.0

ST Erg bnis

END_PROGRAM
```

对于直接表示的变量，强烈推荐使用符号名，因为这样对于不同的地址容易重复编写。这个声明也不能保存。

注意：直接表示的变量只可能在类型为 program 的 POU 中定义。OpenPCS 不支持映射到物理 PLC 地址（通过使用 AT%）的 ARRAY 和 STRUCT 类型的变量。

如果直接表示的变量声明为程序 POU 的全局变量，它们可以用为一个调用功能块的外部变量。另一个可选的办法是将变量作为 VAR_IN_OUT 参数传到功能块。不管是对 PCS 输出，标志和通信文件 RD、SD，这个办法都是可行的。

OpenPCS 支持下面表示的变量地址：

I：数字输入

Q：数字输出

M：标志

哪一个地址有效取决于工程的硬件。

作为 大小 ，可以使用下面的符号：

X，或没有：(位)，大小=1 位；例如：%IX0.0 或 %I0.0

B (字节)，大小=8 位；例如：%IB0.0

W (字)：大小=16 位；例如：%QW0.0

D (双字)：大小=32 位；例如：%ID0.4

L (长字)：大小=64 位；例如：%IL0.0

2.4.6 派生数据类型

派生数据类型由控制器的制造商定义或者是您自己定义。这些新的数据类型使用关键字 TYPE□END_TYPE 定义，以基本数据类型为基础。定义完之后，使用它们就像使用预定义或基本数据类型一样。

在下面的例子代码，定义一个新的数据类型去代表 压力 值。

```
TYPE
Pressure:INT;
END_TYPE
VAR
PreValvePressure:Pressure;
END_VAR
```

不同的数据类型可以组合成一个派生数据类型。数组和结构也可以集合进来。下面的例子定义了一个结构 A。这个结构本身包含另一个结构 B 和一个长度为 5 的整形数组。B 中又派生出三个新的数据类型：字符串 Stationname 和双精数 Value1，Value2

```
TYPE
A :
STRUCT
B :
STRUCT
 Stationname : STRING
 Value1 : DOUBLE
 Value2 : DOUBLE
END STRUCT
Arr_5_INT:ARRAY [1..5] OF INT;
END_STRUCT
END_TYPE
VAR
 Data1: A;
END_VAR
```

2.4.7 数组类型的声明

数组包含同一数据类型的多个元素。使用关键字 ARRAY 定义数组。数组的每一个元素是基本数据类型变量。

举例：数组数据类型

类型 Arr1 包含 5 个 INT 型的元素。

```
PROGRAM feld
TYPE
Arr_5_INT:ARRAY[1..5] OF INT;
END_TYPE
```

```
VAR
Arr1:Arr_5_INT;
END_VAR
END_PROGRAM
```

2.4.8 结构数据类型的声明

一个结构包含多个相同或不同数据类型的元素。使用关键字 STRUCT 定义一个结构。一个结构的各个元素称为那个结构的成员，通过写出结构再跟上一个点号和成员名来访问它们。

举例：结构数据类型

```
PROGRAM struktur
TYPE
  RobotArm:
 STRUCT
 Angle_1:REAL;
 Angle_2:REAL;
 Grip:BOOL;
 Length:INT;
  END_STRUCT;
END_TYPE
VAR
 Robot1:RobotArm;
 Robot2:RobotArm;
END_VAR
LD Robot.Grip
.
.
```

END_PROGRAM

2.4.9 枚举类型的声明

一个枚举类型的变量可以使用任何一个固定列表中的值。列在枚举类型声明的合法值通过逗号分隔。初始值跟在小括号) 后面；如果没有给出初始值，默认为第一个值。

举例：枚举类型

数据类型 TrafficLight 可以是 red， yellow 或 green 。 yellow 是默认值。

```
TYPE TrafficLight:
(red,
yellow,
green):= yellow;
END_TYPE
VAR
MainRoad: TrafficLight;
CrossRoad: TrafficLight;
StopCar: BOOL;
END_VAR
```

在 POU 的指令部分，可以使用已定义的枚举值：

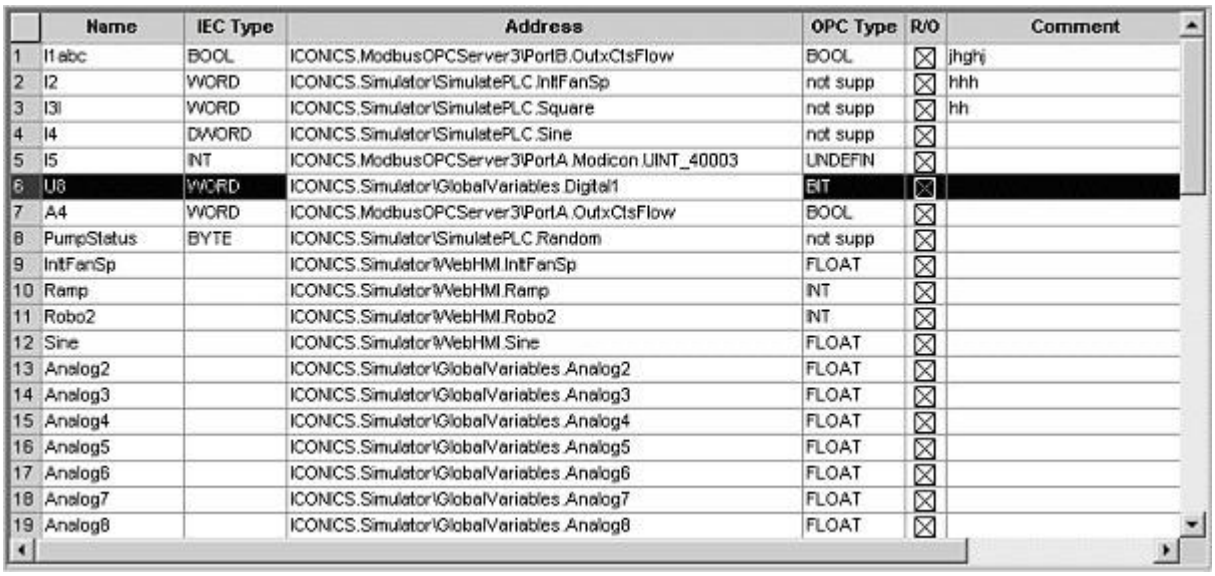
举例：IL

```
LD MainRoad
EQ red
ST StopCar
```

2.5 赋值编辑器

2.5.1 赋值编辑器：介绍

赋值编辑器位于 OpenPCS 框架，它作为一个文档编辑窗口来显示。



	Name	IEC Type	Address	OPC Type	R/O	Comment
1	I1 abc	BOOL	ICONICS.ModbusOPCServer3\PortB.OutxCtsFlow	BOOL	☒	jhghj
2	I2	WORD	ICONICS.Simulator\SimulatePLC.IntFanSp	not supp	☒	hhh
3	I3l	WORD	ICONICS.Simulator\SimulatePLC.Square	not supp	☒	hh
4	I4	DWORD	ICONICS.Simulator\SimulatePLC.Sine	not supp	☒	
5	I5	INT	ICONICS.ModbusOPCServer3\PortA.Modicon.UINT_40003	UNDEFIN	☒	
6	U8	WORD	ICONICS.Simulator\GlobalVariables.Digital1	BIT	☒	
7	A4	WORD	ICONICS.ModbusOPCServer3\PortA.OutxCtsFlow	BOOL	☒	
8	PumpStatus	BYTE	ICONICS.Simulator\SimulatePLC.Random	not supp	☒	
9	IntFanSp		ICONICS.Simulator\WebHMI.IntFanSp	FLOAT	☒	
10	Ramp		ICONICS.Simulator\WebHMI.Ramp	INT	☒	
11	Robo2		ICONICS.Simulator\WebHMI.Robo2	INT	☒	
12	Sine		ICONICS.Simulator\WebHMI.Sine	FLOAT	☒	
13	Analog2		ICONICS.Simulator\GlobalVariables.Analog2	FLOAT	☒	
14	Analog3		ICONICS.Simulator\GlobalVariables.Analog3	FLOAT	☒	
15	Analog4		ICONICS.Simulator\GlobalVariables.Analog4	FLOAT	☒	
16	Analog5		ICONICS.Simulator\GlobalVariables.Analog5	FLOAT	☒	
17	Analog6		ICONICS.Simulator\GlobalVariables.Analog6	FLOAT	☒	
18	Analog7		ICONICS.Simulator\GlobalVariables.Analog7	FLOAT	☒	
19	Analog8		ICONICS.Simulator\GlobalVariables.Analog8	FLOAT	☒	

它用来指派全局变量给 I/O 端口，比如一个标记在 OPC-DA 服务器上的值。

这个编辑器是一个栅格的形式，每行代表一个变量任务。列的含义描述如下：

Name 指派的 IEC 变量的有效名称

IEC-Type 指派的 IEC 变量的类型

Address 指派的外部数据项的地址。 这个形式取决于指派的数据项类型. 在 OPC 情况下, 这列将包含全部合格的 OPC 标签名称。

OPC Type 指派的 OPC 的特殊的标签类型

R/O 当检查时，显示变量应该声明为只读。当不检查时， 变量应该声明为读写。

Comment 任务中可选的文本内容。 它仅存在任务文档中，并且不被编译器和其他工具所使用。

您可以在 地址 区域中手动加入 OPC 标签的名称，或者利用 OPC-I/O 浏览器窗格来浏览地址空间并且指派标记，这个标记用菜单项的编辑 -> 添加标记或编辑 -> 指派标记来指派。

Edit / Add tag （或双击）：为了声明当前在浏览器窗格中选择的 OPC 标签符，在任务编辑器中插入一个新的声明行。如果这个 OPC 标签已经在任务编辑器中定义了，那这个定义行就作为当前行。

Edit / Assign tag：从浏览器窗格中指派一个标记符到任务编辑器的当前选择行。当且只有当前一行恰好被选中时运行。

利用这些功能中的一个，指派标记的 OPC 类型被自动确定，并且被赋给 OPC 类型 的列。在默认情况下，IEC 变量名使用 OPC 标签名中关键成分，并且可以按用户需要而改变。变量的 IEC 类型必须要用户手动指定。

2.6 IL 编辑器

2.6.1 IL 编辑器：介绍

IL 编辑器在 OpenPCS 框架之中。在 IL 编辑器的上部，输入 POU 的声明，在下面的窗格，输入 IL 指令：

IL 编辑器支持书签(在编辑一个文件时，标志感兴趣的位置方便查找)和断点。

2.6.2 指令表的结构

一个 IL 行有下面的形式，可选部分写在方括号内[ ]，表达式写在尖括号内<>：

[<标签 I>:] <操作符> <操作数 1> [,<操作数 2>,<操作数 3>,□] [(*<注释>*)]

如果行代表一个跳转目标，一开始则是一个标签。后面是操作符和操作数，由逗号分隔开。注释在 (* 和 *) 之间。

举例：

Start: LD a (*a 装载到寄存器*)

ADD b (*寄存器加 b*)

ST c (*存储结果到变量 c*)

通过分别使用操作 CAL 和 CALC 调用功能块实例；操作数是实例名，跟着是圆括号和其内的参数。

[<Label>:]CAL/CALC<Instance name>(

[<Input1>:=<Value1>,<Input2>:=<Value>,□]

|

[<Variable>:=<Output1>,<Variable2>:=<Output2>,□]

)

参数传递包括两个部分。在第一部分，分别通过对 INPUT-和 IN_OUT 变量设值将参数传到功能块。没有获得值的变量，它们的值分别为最后一次调用的值和初始值。第一部分被 | 所分隔，指定输出参数。

2.6.3 IL 的指令

要看 IL 支持的所有指令列表，请看参考部分，指令表关键字。

2.6.4 在线 IL 编辑器

为了调试和监控用 IL 写的代码，可使用在线模式的 IL 编辑器。

首先确保您已经打开资源窗格中代码的实例，而不是文件窗格中类型的代码。通过移动鼠标光标到代码处，您可以容易的判断：如果是实例树中的代码，鼠标光标应该是活动的。

主要有三种办法调试和监控 IL 代码：

1. 使用断点停止执行程序，单步执行代码。使用这个办法可以理解、跟踪和发现应用程序中的逻辑问题。

2. 移动鼠标光标到一个变量上，可以看到一个很小的 工具箱 ，这个工具箱显示变量名、类型和变量值。这个值是不断更新的。在停止或没有停止执行的情况下，在断点处或者在单步执行时，使用这个办法，可以快速检验一个域内不同变量的当前值。

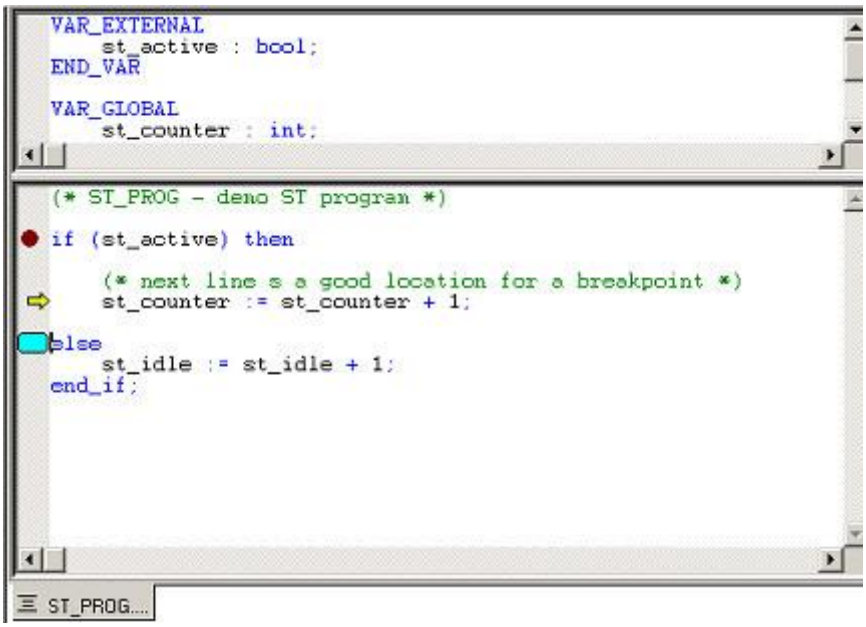
3. 使用测试和调试的观察列表监控一整套变量，这些变量可能来自应用程序的各个部分。使用这个办法在您检验应用程序的其它部分时还可以密切留意各个变量。

OpenPCS 支持在线编辑，更多信息请参阅用户手册中的在线编辑。

2.7 ST 编辑器

2.7.1 ST 编辑器：介绍

ST 编辑器在 OpenPCS 框架之中。在 ST 编辑器的上部，输入 POU 的声明，在下面的窗格，输入 ST 指令：



ST 编辑器支持书签(在编辑一个文件时，标志感兴趣的位置方便查找)和断点。

2.7.2 ST 中的指令

ST 中的代码是一系列的 ST 指令。指令以分号结束。

换行不是很重要，也就是说，一个行可以有几条指令，一条指令也可以占用几个行。

要看 ST 支持的所有指令，请看参考部分，结构文本关键字。

2.7.3 ST 的表达式

ST 中的操作数为：

字符变量，比如：14， abc ，t#3d_5h

变量，比如：Var1，Var[2,3]

函数调用，比如：Max(a,b)

因为操作是 ST 语言的一部分，表达式必须通过 ST 元素的帮助来构建。操作符需要操作数去构建表达式。

圆括号	( )
函数调用	
求幂	**
取负	-
求补	NOT
乘	*
除	/
求模	MOD
加	+
减	-
比较	<, >, <=, >=
相等	=
不等	<>
与	&, AND
或非	XOR
或	OR

2.7.4 ST 中的注释

像所有现代的编程语言一样，ST 支持注释。注释是包括在 (* 和 *) 之间，比如：

(*注释是有用的*)

编译器在生成可执行代码时会忽略注释，因此如果您不用注释，您的程序执行速度跟有注释是一样的。注释可以扩展到多个行，比如：

(*这个注释

很长，需要

几

行

*)

2.7.5 ST 在线编辑器

要调试和监控 ST 的代码，使用 ST 编辑器在线模式。

首先确保您已经打开资源窗格中代码的实例，而不是文件窗格中的类型代码。通过移动鼠标光标到代码处，您可以容易的判断：如果是实例树中的代码，鼠标光标应该是活动的。

主要有三种办法调试和监控 ST 代码：

1. 使用断点停止执行程序，单步执行代码。使用这个办法可以理解、跟踪和发现应用程序中的逻辑问题。

2. 移动鼠标光标到一个变量上，可以看到一个很小的 工具箱 ，这个工具箱显示变量名、类型和变量值。这个值是不断更新的。在停止或没有停止执行的情况下，在断点处或者在单步执行时，使用这个办法，可以快速检验一个域内不同变量的当前值。

3. 使用测试和调试的观察列表监控一整套变量，这些变量可能来自应用程序的各个部分。使用这个办法在您检验应用程序的其它部分时还可以密切留意各个变量。

OpenPCS 支持在线编辑，更多信息请参阅用户手册中的在线编辑。

2.7.6 结构和结构元素的提示条

现在您可以在 ST 编辑器的任何层查看结构体的信息。

```
control.speed := 15;
```

```
control : my_struct
STRUCT
   is_running : BOOL;
   speed : DINT;
END_STRUCT
```

如果编辑器是在 编辑 模式下，那么结构和它的首层成员会和它们的数据类型一起显示出来。在 在线 模式下，可分解成员的值会显示在后面。

```
control.speed := 15;
```

```
my_struct control
STRUCT
   BOOL is_running = FALSE
   DINT speed = 15
END_STRUCT
```

2.7.7 自动完成/自动声明

在输入变量时，如果该变量未声明同时 CTRL+SPACE(Return 在梯形图编辑器中)被按下，声明变量窗口会显示。

如果同样名称的变量已经存在，没有变化。

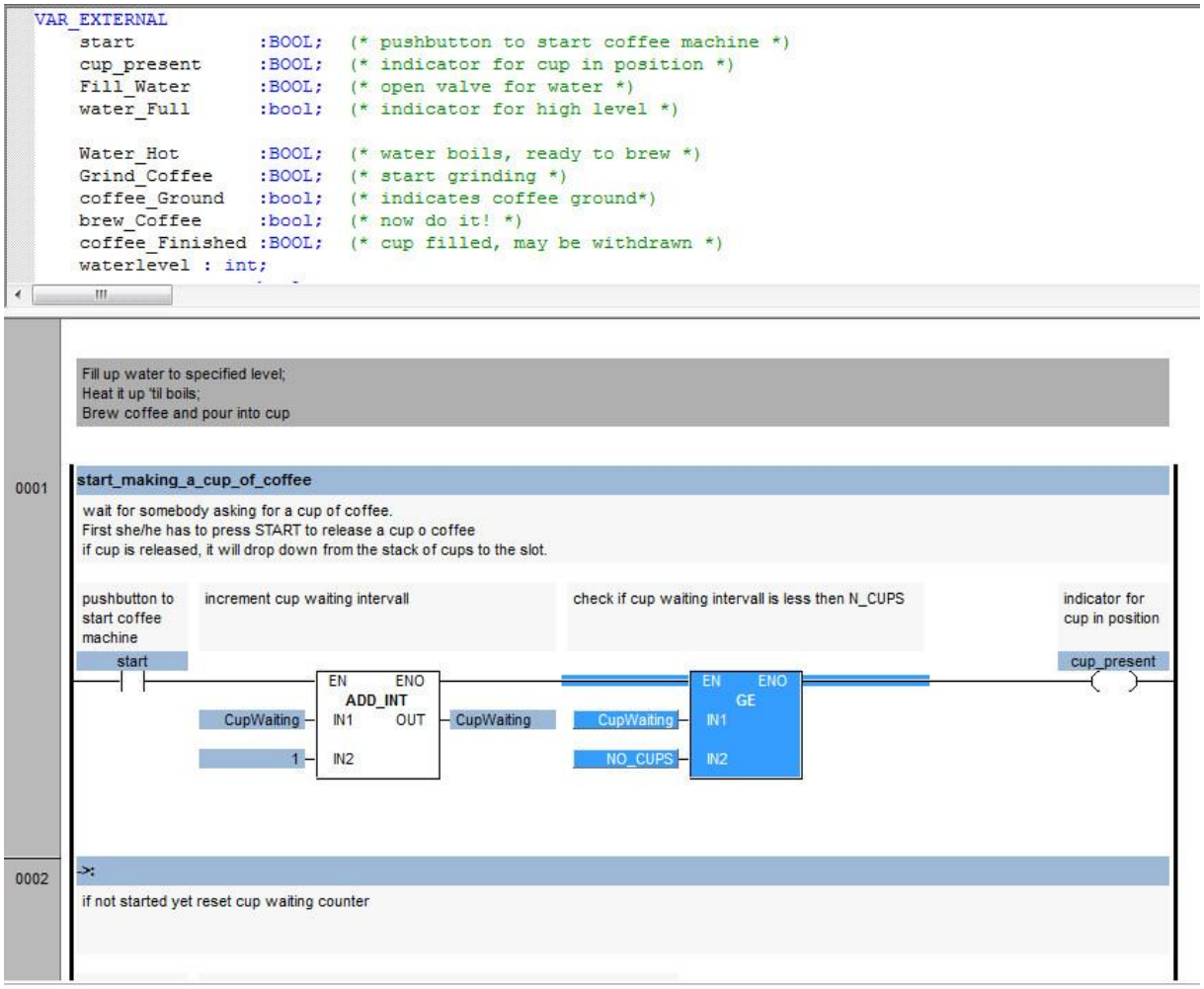
在输入变量时，如果输入了已声明变量的前一部分，该变量会被插入到指定位置。

如果同时有多个变量可选，将会显示一个列表，用户可用 UP 或 DOWN 键来选择。按下回车键当前选定的变量会被插入指定位置。按下 ESCAPE 键列表隐藏并返回正常编辑模式。

2.8 梯形图编辑器

2.8.1 梯形图编辑器：介绍

梯形图编辑器在 OpenPCS 框架之中。在梯形图编辑器的上部，输入 POU 的声明，在下面的窗格，输入梯形图指令。



2.8.2 梯形图逻辑：介绍

梯形逻辑的基本原理是能量流自上而下流过网络。一般地，梯形逻辑限制在布尔信号(1=真，0=假)的处理上。

一个网络是限制在梯形图编辑器的左侧和右侧间所谓的边界连接区中。边界连接区的左侧有逻辑值 1(当前值)。这些连接将引导能量在元素(变量)中流通，是否流通至右侧或者被隔离将取决于其逻辑状态。处理的结果依赖于元素的排列以及它们连接的方式(AND=串联，OR=并联)。

网络中包含的图形对象：

连接(平行或垂直线和结点)

触点，线圈，控制继电器

功能块和函数

跳转(控制流的图形元素)

2.8.3 网络

梯形图编辑器的指令部分划分为几个所谓子网络，便于图形的结构化。

一个网络包括：网络标号，网络注释和 网络图形。

网络标号：每个网络可成为另一个网络的跳转目标，它会自动被分配为一个字符标识符或无符号的十进制整数。缺省时，网络将分配数字。向网络插入一个新的网络时，所有网络的数字将会自动更新。网络标号数字化后，就可以根据文本编程语言的行号方便地找到某一特定的网络。

网络注释：梯形图中网络注释描述在矩形区域。键入注释时，双击这个矩形区域即可。注释总是置于网络标号之下的位置。注意第一个网络附加的包含了一个梯形图注释，它位于网络标号和网络注释之上。

网络图形：网络图形由图形对象组成，可以是图形符号或连接。图形符号之间有连接可以传输数据，它能在输入端处理数据并把处理过的数据传到输出端。注意连接可交叉。

2.8.4 操作符

在梯形图中，操作符标明图形对象的接点、线圈和跳转。

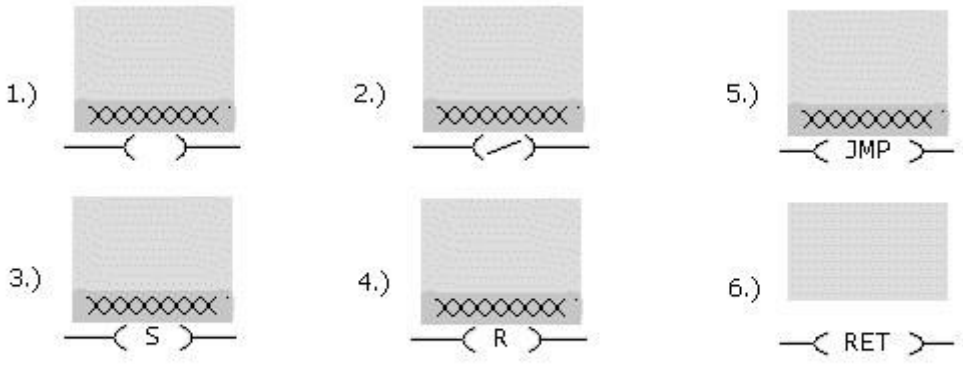
触点：一个触点把输入值和指定的变量值连接起来。连接的类型取决于接点的类型。结果将会传到右侧，那里有触发器或中断器(变量的布尔值将不变化)。

线圈：在网络中线圈把值赋给输出变量。一个线圈把它的左边的连接状态拷贝到右边的连接，而不会发生变化。线圈还具有保持状态功能，或把它的左边连接状态赋值给一个布尔变量。

跳转：跳转操纵程序的流程。它可根据指定的顺序直接调用特定的网络。当遇到跳转操作符时，控制流在不同网络中继续执行。因此，跳转对于 LD 基本原则(即网络总是自上而下处理的) 而言是个例外。

2.8.5 线圈

输出变量总是位于网络右侧，且和右侧轨线相连。



1. 逻辑连接的结果直接赋值给输出变量。

2. 输出变量被赋值为逻辑连接结果的非。

3. 输出变量被逻辑连接结果置位：若逻辑连接结果为 1 ，输出变量将置 1 ；相反，若逻辑连接结果为 0 ，将无输出。

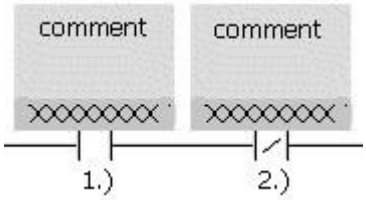
4. 输出变量被逻辑连接结果复位：若逻辑连接结果为 1 ，输出变量将置 0 ；相反，若逻辑连接结果为 0 ，将无输出。

5. 跳转操纵程序的控制流程。使用跳转，仅执行指定的网络。跳转可由一个二进制组合结果有条件或无条件的执行，比如，可以无条件的执行。跳转的目标必须是在一个网络的开头，可由网络标号指定。

6. 返回跳转可在当前 POU 下停止程序的执行，在 POU 被调用点上继续执行。返回跳转应由一个二进制组合结果设为有条件的或无条件的。

2.8.6 触点

布尔输入变量有两种接点符号：



左边变量的接点符号必须为 1 ，使得相应的布尔连接为真，若变量和物理地址相连，状态 1 对应于一个释放的中断器或一个按下的触发器。右边变量的接点符号必须为 0 ，使得相应的布尔连接为真，若变量和物理地址相连，状态 0 相应于一个按下的中断器或释放的触发器。

2.8.7 控制继电器

控制继电器相当于插入到线圈前的触点。例如，它可在手动执行时设置断点。在每个线圈前总应有一个控制继电器。

插入->控制继电器：使用这个命令插入一个对于逻辑符号可选的控制继电器。

2.8.8 功能块和函数

为在网络中插入功能块或者函数，单击一个连接并且使用***插入->功能块*** 或者 **插入-> 函数** 在网络中插入。您可以从可用功能块 / 函数中选择想要的块或者函数。只有预定义的函数才能被使用。

注意：

一个功能块只能被加到一个网络中，如果满足下面的标准：

1. 块的第一个输入参数必须是布尔类型，并且必须有个名称是 EN 。如果这个参数在网络中被设 为 FALSE，相应的块将不启动。

2. 块的第一个输出参数必须是布尔类型，并且必须有个名称是 ENO 。这个参数必须被设为 TRUE ，如果这个块工作正确并且没有错误的话。

2.8.9 在线梯形图编辑器

首先确保您已经打开资源窗格中代码的实例，而不是文件窗格中的类型代码。

当您启动梯形图编辑器的在线模式时，它会自动的尽可能展示触点、线圈、函数和功能块输入和输出的实时值。

如果在线编辑器不能从运行时系统中得到一个变量值，那它将显示 -!- 。

目前版本的梯形图编辑器还不支持大于 4 bytes 的变量类型在在线编辑器中显示它们的值（字符串，数组，结构等）。要查看它们，请使用测试与调试。

OpenPCS 支持在线编辑，更多信息请参阅用户手册中的在线编辑。

2.8.10 检查变量

梯形图编辑器包含了一个检查注释的工具，如果一个程序的语法发生了变化，它会在注释中作上标记。OpenPCS 在注释中加上 [检查!] 来表示这个注释可能有错误。您可以自己决定这些注释是否正确。

这样做的原因是因为在使用梯形图编辑器时，很有可能用一个触点来替换一个函数（块），或者反之。这样就会改变程序的语法，因此函数（块）或变量上方的注释可能变成不正确的。

为了演示这个功能，请参看下图。选择一个您想要用一个触点变量替换的函数，右键点击它并选择插入变量。

替换完成以后，原来函数上方的注释发生了变化，它被写入了 [检查!] 。

主要原因就是，程序的语法发生了变化，但注释没变。这是一个提示，来检查注释是否依然正确。

2.8.11 自动完成/自动声明

在输入变量时，如果该变量未声明同时 CTRL+SPACE(Return 在梯形图编辑器中)被按下，声明变量窗口会显示。

如果同样名称的变量已经存在，没有变化。

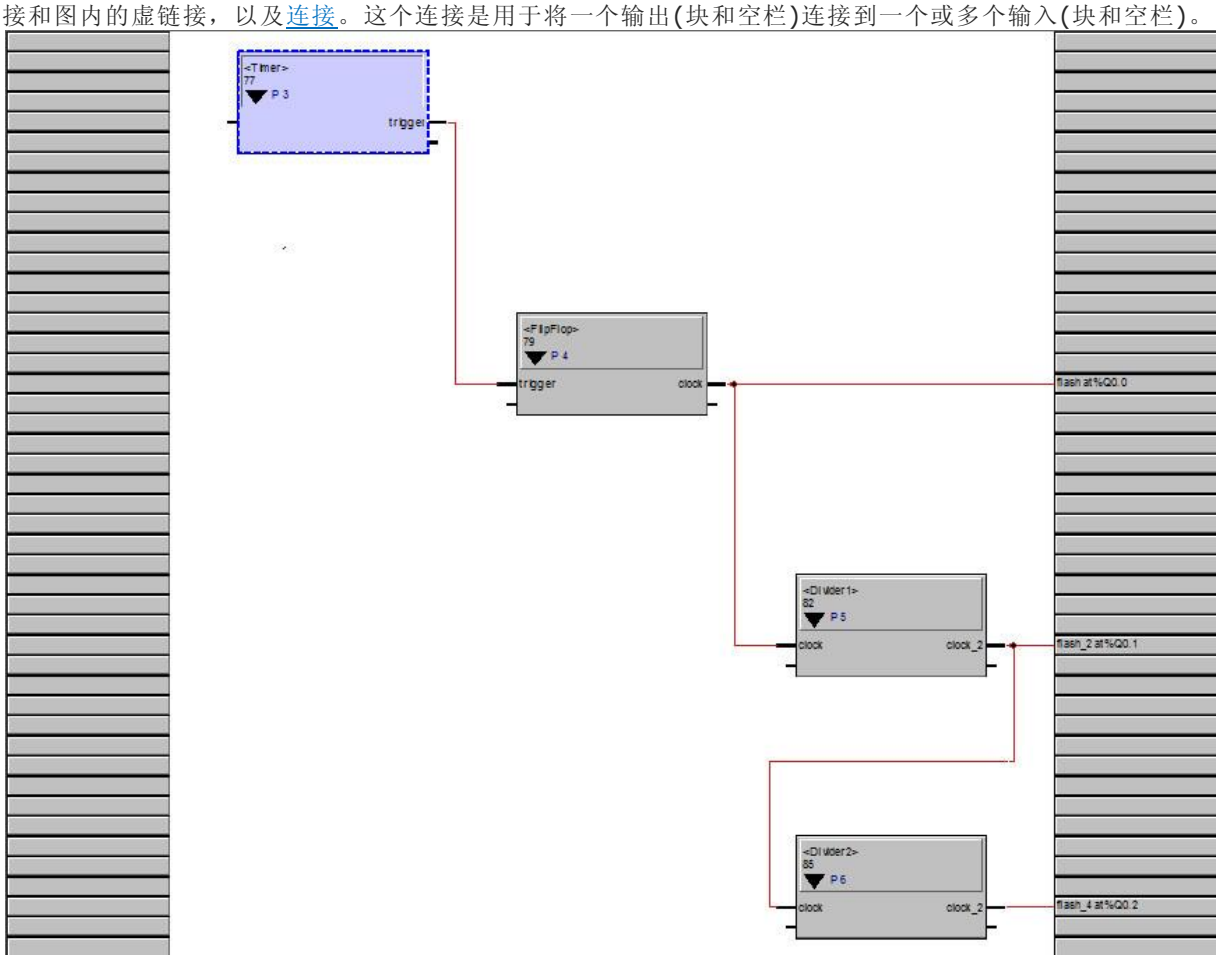
在输入变量时，如果输入了已声明变量的前一部分，该变量会被插入到指定位置。

如果同时有多个变量可选，将会显示一个列表，用户可用 UP 或 DOWN 键来选择。按下回车键当前选定的变量会被插入指定位置。按下 ESCAPE 键列表隐藏并返回正常编辑模式。

2.9 CFC 编辑器

CFC 编辑器介绍

OpenCFC(R)编辑器(连续功能图形编辑器)是一个用来创建自动化程序的工程工具。CFC 功能图的主要元素是块 (固件块、用户定义块、复合块)，它可以自由地安排在图的空白栏 (左和右)上，用于提供与 IEC1131 变量的连接和图内的虚链接，以及连接。这个连接是用于将一个输出(块和空栏)连接到一个或多个输入(块和空栏)。



2.9.1 使用块工作

要增加块到您的 CFC 图，使用 插入->块，可以选择固件或用户定义块，使用 插入->文本块 可以插入文本块，或使用 插入->复合块 插入复合块。点击您要插入新块的图，鼠标光标会发生变化。

要重新组织块，选择块然后拖放到新位置即可。当增加一个新块或移动一个已存在的块时，CFC 编辑器会适当的移动其它块来给它们腾出位置。

要删除块，选择它们然后按下 DEL 键。

鼠标双击一个块可以给这个块一个别名。

2.9.2 连接

要连接两个对象，首先选择输出的对象(块的输出，或左边空白栏的条目)，然后选择输入(功能块的输入，或空栏右边的选项)，然后按下插入->连接。

OpenPCS 也支持多连接 。

2.9.3 空白栏

空白栏用于将 CFC 图的逻辑与同一 CFC 图的其它部分或与应用程序的其它部分或者是需要控制的过程连接起来。

要配置空白栏的任何一个元素，右击鼠标然后从弹出的相关菜单中选择 属性 。

在名称中，键入对象的名字。它必须是有效的 IEC61131-3 变量名。

如果您想让 CFC 编辑器为这个空白栏对象声明一个变量，选择 IEC611131-3 变量。否则您如果选择 CFC 连接器 ，对象只是虚拟的，并且所有的信息立即传到所连接的输出。这样可能在运行时间和内存占用上会比较经济，但是它不允许在线监控。

对于 IEC61131-3 变量，从下拉式列表框中选择声明部分。这个选择取决于块的类型和空栏的类型。对于一些类型的变量，您可以选择一个物理地址或初始值。

对于 CFC 连接器，您可以选择 复合块连接器 ，也就是一个复合块到外部的连接，或选择 (连接到)内部连接器 ，也就是在右边空白栏虚接一个入口到左边白空栏。 内部连接器 和 连接到内部连接器 是相似的，但对于前一个来说只对于一个在右边空白栏(内部连接器定义在这里)有效，而后一个只对于一个在左边空白栏(这里可能使用内部的连接器)有效。

2.9.4 CFC 在线编辑器

首先确保您已经打开资源窗格中代码的实例，而不是文件窗格中的类型代码。当您在在线模式下启动 CFC 编辑器时，它会自动开始显示块，连接和空白栏条目的实时值。如果在线编辑器不能从运行时系统中得到一个变量值，它将显示 -!- 。

OpenPCS 支持在线编辑，更多信息请参阅用户手册中的在线编辑。

2.9.5 高级主题 CFC

2.9.5.1 文本块

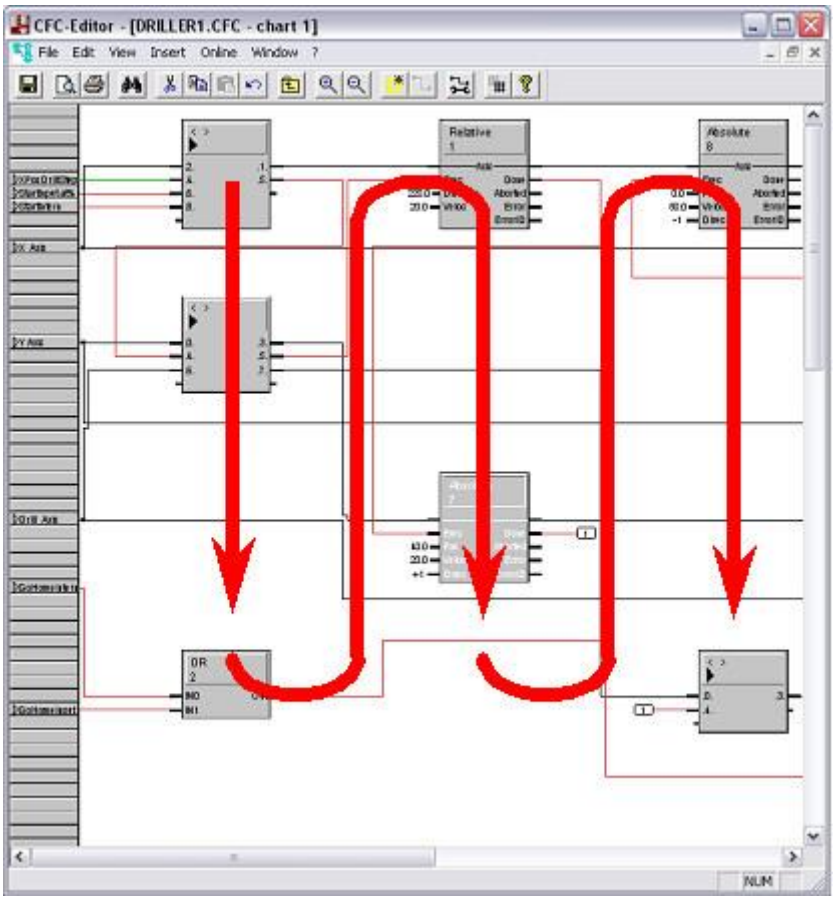
使用***插入->文本*** 块在 CFC 图上插入文本块。一个文本块只用于文档目的，对于被执行的代码，它没有增加任何东西。

2.9.5.2 使用常数作为输入

要使用一个常数作为一个块的输入，选择输入(或空白栏入口)，右击鼠标，选择 属性 然后在表单 默认值 上的编辑域 值 中键入常数值。

2.9.5.3 执行顺序

在一个图中块的安排和执行顺序直接相关：块从顶到底从第一列执行，然后第二列从顶到底执行，以此类推，要修改执行顺序，就需要重新安排块。



复合块在它所在的位置作为一个整体来执行。复合块内部的执行遵循相同的原则。这跟现代编程语言的子程序非常相似。

2.9.5.4 多连接

CFC 编辑器支持一个输出和多个输入之间的连接。要建立多连接，首先要建立一个期望输出和一个输入之间的连接。现在，选中下个输入并且单击输出。这时这个在第一步建立的连接和输出都被选中。选择***插入->连接*** 来建立这个输出和这两个输入之间的多连接。您可以用同样方法加更多的输入。

要从多连接中删除一个输入，选中这个输入，单击删除按钮。只有这个输入和输出之间的连接被删除。

2.9.5.5 功能块的替换

CFC 编辑器支持用另一类型的功能块替换厂商和用户定义的块。 选定要替换的功能块，在菜单中选择编辑-> 替换功能块。一个类似插入->功能块 的对话框会弹出，用户可以从一个固件的和用户定义的功能块类型的列表中选择期望的功能块的新类型。

除此之外，用户可以选择选项 自动替换当前企划中所有实例 ，此选项被选中，当前 CFC 中所有与所选功能块类型相同的功能块，即使没有被选定，也会全部自动替换为新类型。

选择了一个新的功能块类型之后，弹出另一个对话框，使用户在替换之后重组旧块类型的连接器，形成新块类型的连接。表格左栏显示旧块类型的连接器，以及连接器（1*）的类型(VAR_INPUT/VAR_OUTPUT)。表格右栏显示合适新块类型的连接器列表。用户可以匹配相应的连接器到旧块类型连接器。注意，每个新功能块的连接器只能被匹配一次。

若一个连接器无法或不应重新连接，请选择选项 不自动重新连接 。

电击 确定 之后 CFC 编辑器替换所有旧类型的功能块为新类型，并根据连接器匹配表重新接线。

(*1): VAR_IN_OUT 连接器会在连接器列表中出现两次：一次为 VAR_INPUT& ， 另一次为 VAR_OUTPUT&. & 标记连接器实际上表示的一个 VAR_IN_OUT 参数

2.9.5.6 在 CFC 中找错误

如果您双击输出窗口 框中的不同错误消息，CFC 编辑器会定位出相应的错误位置。

2.9.5.7 功能块特定帮助

您可以得到功能块特定的帮助。右键点击想要获得帮助的功能块，在菜单中选择 显示说明文档 。如果 OpenPCS 找不到相关文档，您会得到一个提示。如果找到一个，它会显示出来。如果找到了多个文档，您会被提示选择其中之一来查看。

2.9.5.8 可扩展输入

以下 CFC（和 FBD）函数可以扩展。也就是说我们可以添加一个或多个第一个输入的复制作为输入。

AND, ANDN, OR, ORN, XOR, XORN, MUL, ADD, MUX, MIN, MAX, CONCAT

要添加输入，只需选中以上函数之一然后使用右键菜单中的 追加输入 。如果您想删除追加的输入，请选中该输入然后使用右键菜单中的 删除输入 项。

2.9.5.9 可对输入取反的函数

对于下列所有逻辑 CFC（FBD）函数，您都可以对它们的布尔型输入取反：

AND, ANDN, OR, ORN, XOR, XORN, NOT

要对输入取反，请选择该输入并使用右键菜单中的 输入取反 。在这个输入上会显示一个圈表示取反。

再次使用右键菜单中的 输入取反 项将取消取反操作。

2.9.5.10 CFC 连接的语法检查

在对任何 CFC 连接器的值或 IEC 标识符进行直接编辑后，用户输入的值都会被进行语法检查：如果一个不符合连接器数据类型的常量被输入，那么您会得到一条类似的消息：

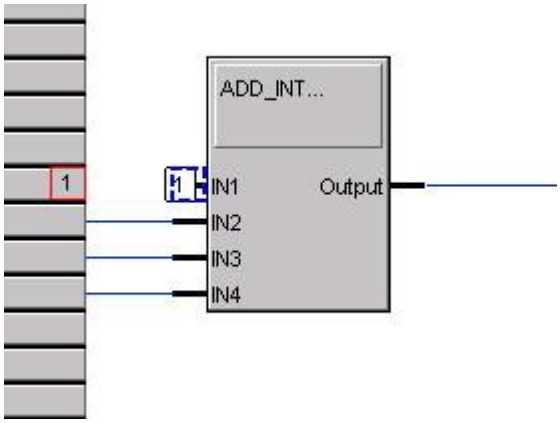
语法错误：无效数据类型对于常量 XXX

但该值还是仍旧被接受。

2.9.5.11 连接旗标

为了减少可视的连接线的数量，我们可以隐藏单个的连接。在右键菜单中选择 强制连接旗标 。

对单个连接使用连接旗标。



隐藏起来的连接会和图一起被保存并在打开时还原。

连接旗标也被使用，当连接的两个连接器处于不同的页。这些旗标在程序中是不可见的，但如果打印注释和旗标选项（见打印选项）被选中，这些旗标也会被打印。

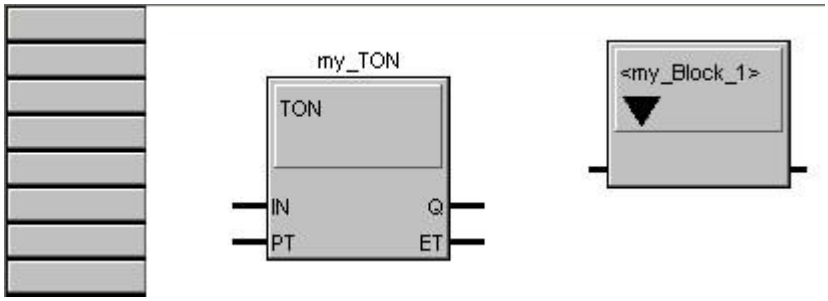
旗标会根据页码以十六进制格式编码。

2.9.5.12 复制有输入的功能块

如果至少有一个功能块已被选中，那么右键菜单中会出现一个新的项： 副本 。选择这一项会将选中的功能块复制到内部的剪切板中，并把编辑器设置为复制模式□□鼠标箭头和插入符同粘贴模式：无论在何处点击或是按下空格键，都会插入一个该功能块的副本，并且所有的输入连接也会被复制。编辑器会一直处于复制模式，直到点击鼠标右键，或按下 ESC 键，或点到了 无法粘贴 的区域，这样您就可以插入多个副本。

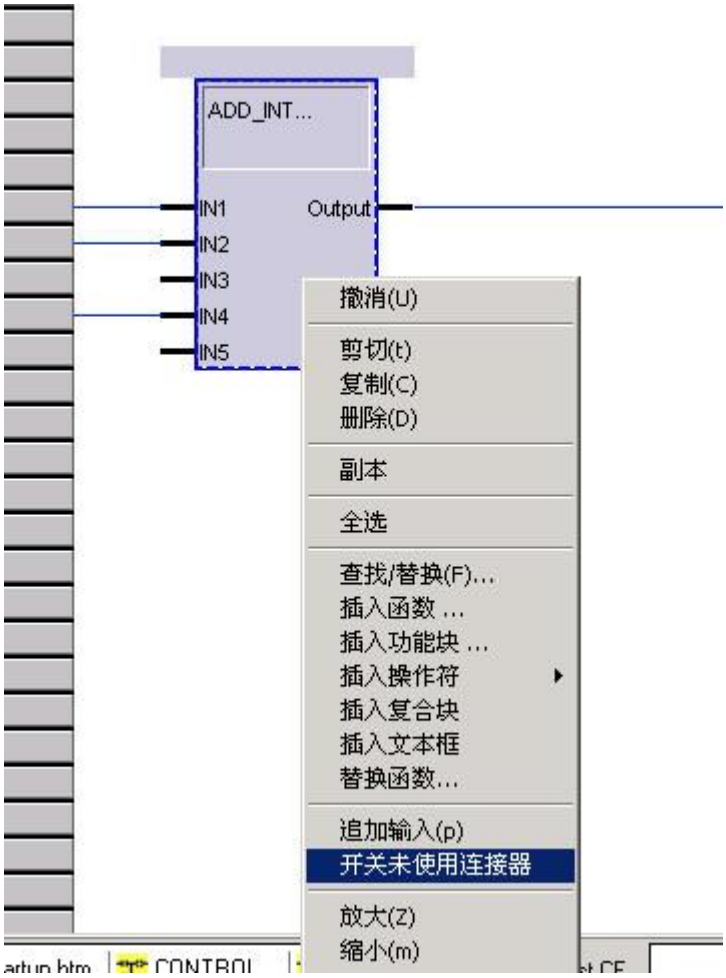
2.9.5.13 别名

用户可以个功能块取别名以便标记并迅速找到特殊的功能块。函数和功能块的别名显示在块的上方，并可以直接编辑。复合块的别名则显示在块中，也可以直接编辑。

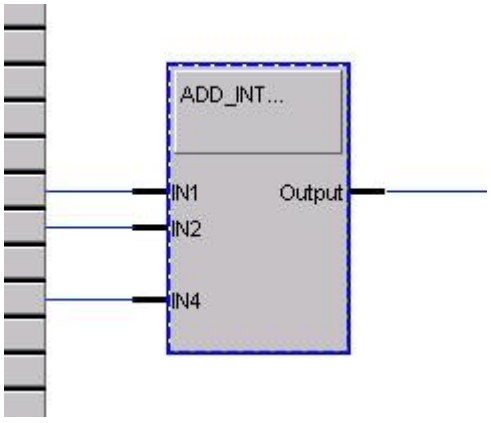


2.9.5.14 隐藏未使用的连接器

为了使图看上去更清晰一些，我们可以使用右键菜单中的 开关未使用连接器 项，这样可以隐藏或显示所有未使用的连接器。未使用的连接器即没有连接或值的连接器。



未使用的连接器不会被显示出来：



如果连接器被隐藏起来了，那么

1. 我们无法通过查找来找到它们
2. 无论是用鼠标还是键盘，我们都无法对它们进行定位。

3. 但我们可以通过双击编译或语法的错误和警告来找到它们。

2.9.5.15 全局标识

每个对象（块或连接器）会被指定一个独一无二的全局标识。块的标识会被显示在块的名称的下方。全局标识也可以在提示条中显示。

2.9.5.16 CFC 与 FBD 编辑器的键盘操作

2.9.5.16.1 键盘使用基础

用键盘进行移动时，一个小的插入符会标识出用户目前的输入位置在哪里。

CFC 和 FBD 编辑器可以同时使用鼠标和键盘来操作。光标并不随着插入符而移动。光标的形状不会自动根据插入符的状态而改变。当然光标的状态会随着光标位置的移动而变化而不是插入符的位置。

2.9.5.16.2 插入符和选择

当前的选择在插入符所在位置。例外或特殊情况有：

1. 如果插入符移动到了一个空的网格单元，那么选择被取消（没有选中任何元素）。
2. 当建立一个连接时，需要把区分当前选择和插入符位置，因此需要按下 shift 键的同时移动插入符。放开 shift 键后，选中的元素中会加上当前插入符所在位置（相当于左键单击该元素）。只有有效的选择状态才会被执行。
3. 在移动时按住 ctrl 键可以完成多重选择。（包含了独立的块的多重选择不被允许。）

2.9.5.16.3 插入符的显示

即使插入符所在位置的元素已经被选中，插入符也是可见的。

1. 在特殊情况下，插入符表现为不同的形式。
2. 即使使用鼠标进行选择，插入符仍然可见。
3. 插入符无法被关闭。
4. 插入符不会被打印出来。

2.9.5.16.4 插入符的定位

插入符

1. 定位在鼠标左键或右键点击标出的点。
2. 通常遵照鼠标的选择。

2.9.5.16.5 特定操作时插入符的位置

即使是在同时使用鼠标和键盘时，也要保证插入符的有效位置。以下的操作会移除有效插入符所在位置的元素，对于这些操作时插入符的位置我们进行如下定义：

1. 通过鼠标选择：通常插入符的位置依据鼠标的选择和自动功能而定

2. 移除/ 剪切一个块：插入符会扩展到整个该块所在的网格单元。

3. 移除/ 剪切一组功能块：插入符会选中改组块所在的网格左上角的单元。

4. 移除/ 剪切一个块的输入：插入符会跳转到该输入上方的输入。如果上方没有其它输入，则插入符会扩展到整个块。

5. 移除/ 剪切一个网络：插入符会跳转到该网络上方的网络。如果上方没有其它网络，则插入符会跳转到下方的可能的网络。

6. 移除/ 剪切一组网络：插入符会跳转到最上面的一个的网络上方的网络。如果上方没有其它网络，则插入符会跳转到下方的第一个网络。

7. 减少网络的行数：插入符会跳转到上方的网格行，网格列不变。插入符会指向第一个网格单元，即使其中本身包含了一个块。

8. 全选 之后插入符的位置：进行 全选 操作之后，插入符跳转到图的最左上角的网格单元。该图也会向上滚动以便显示插入符。在内部其实调用了与使用<ctrl>+<home>时同样的方法。

2.9.5.16.6 插入符的自动定位

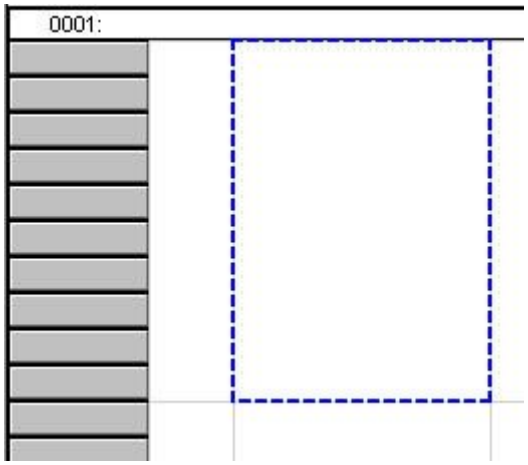
1. 载入一个文件后，插入符被定位在左上角的网格单元。插入符的位置不会和图一起保存。

2. 进入一个复合块中后，插入符定位在左上角的网格单元。

3. 在进行取消/ 重复的操作时，插入符位置根据操作来定。因此在取消/ 重复之前插入符的位置会被保存并根据网络号和位置（行，列）重置。如果网络或者所指的网格单元不再存在，那么插入符会跳转到上方紧接着的网络或单元。

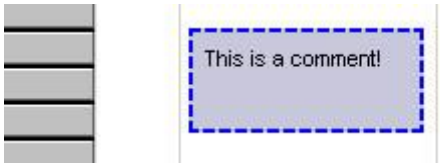
下面列出了使用不同的 CFC/FBD 元素时默认的插入符位置。在位置之间的移动会在下节介绍（插入符的移动）。

空网格单元中的插入符



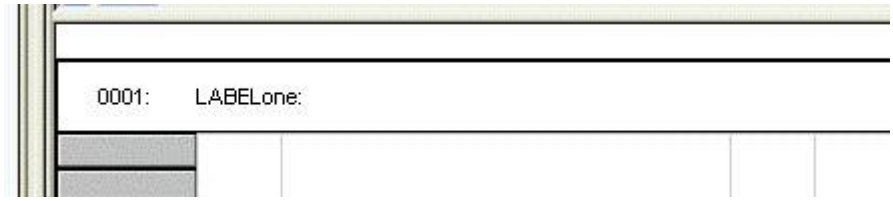
在空的网格单元中，插入符会取整个单元的大小和位置。

插入符和注释



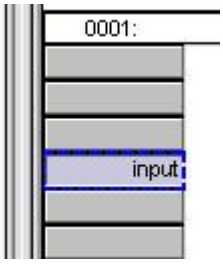
在有注释的网格单元中，插入符会取选中的注释的大小和位置。

网络标题处的插入符



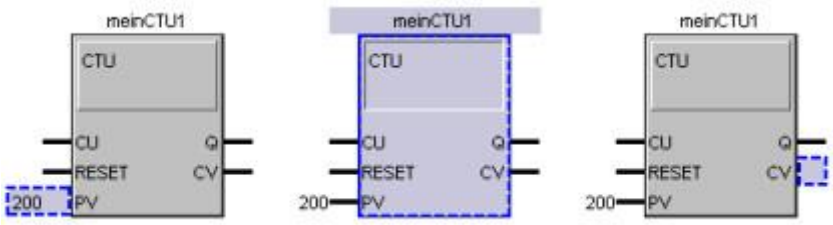
在网络标题中，插入符会根据选中网络的不同来取该网络标题行的大小和位置。

空栏连接器处的插入符



在空栏连接器处，插入符会取选中的空栏连接器的位置和大小。

包含功能块的网格单元中的插入符



插入符会环绕一个块或者一个连接器。插入符的大小取决于选中的连接器或块。这些元素的名字都不会被包含在插入符内。

2.9.5.16.7 插入符的移动

以下说明怎样在 CFC/FBD 图中移动插入符。

在空栏处移动

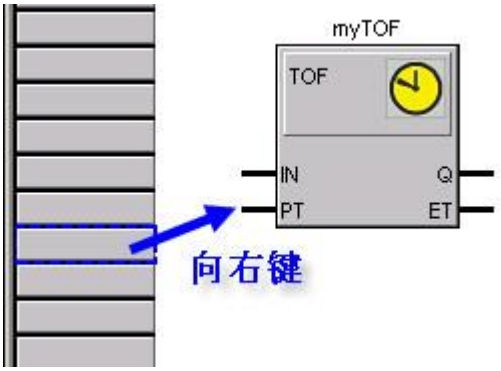
在空栏处，您可以通过向上或向下键来跳转到上方或下方的空栏元素。

在（FBD）网络和网络标题之间移动

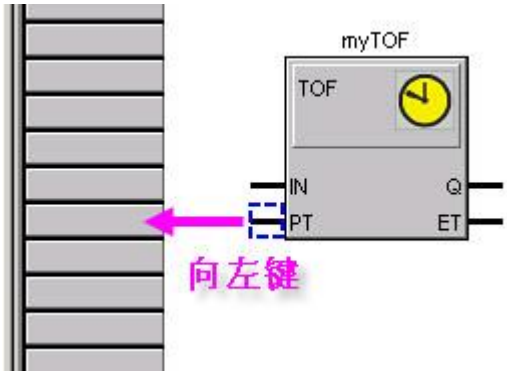
1. 如果插入符位于较上或较下的空栏连接器，可以使用向上或向下键来跳转到上方或下方的网络（见下图）。
2. 如果插入符位于一个网络中较靠上的行的网格单元或元素，您可以用向上键移动到上方网络中。
3. 如果插入符位于一个网络中较靠下的行的网格单元或元素，则可以用向下键跳转到下方的网络的标题行。
4. 如果插入符位于一个网络的标题行，您可以用向上键移动到上方网络左下角的网格单元（或网格元素或连接器）。
5. 如果插入符位于一个网络的标题行，您可以通过向下键跳转到该网络的网格单元（或网格元素或连接器）。用向左或向右键可以跳转到该网络的左空栏或右空栏的连接器。

空栏和块之间的跳转

当插入符位于左边空栏或右边空栏时，使用向左或向右键，可以时插入符跳转到和空栏连接器相邻的网格单元或单元中的元素。连接通路级的空栏连接器总是被指定给该连接通路上方的网格单元。如果网格单元中有块，那么插入符跳转到离得最近的连接器。



如果插入符位于空栏旁边的网格单元中或块连接器上，那么它会跳转到离得最近的空栏连接器。



输入/输出处的向上和向下

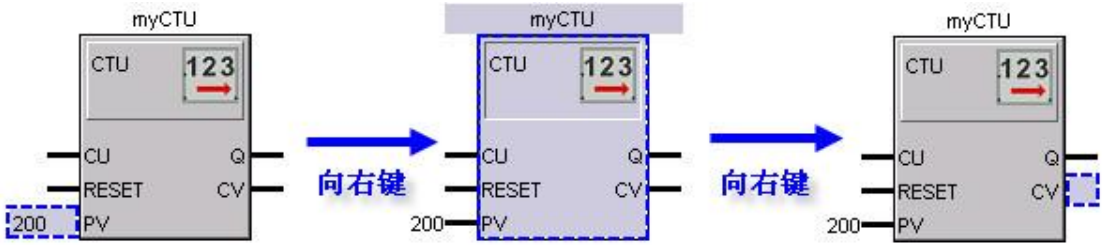
使用向上或向下可以使插入符在功能块的输入和输出之间移动。

如果插入符位于最下面的输入或输出，则使用向下键会使它跳转到下方的网格单元或下个网络的标题栏。

输入和输出处的向左和向右

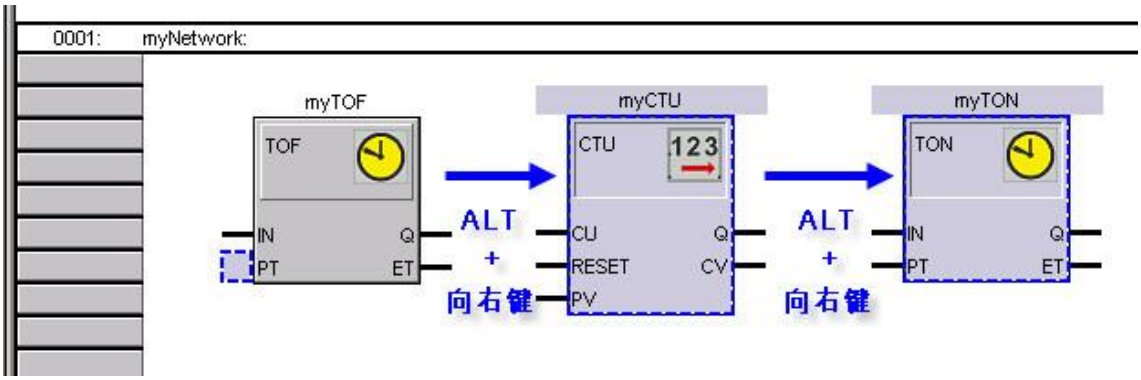
使用向左或向右键可以在输入/输出以及块本身之间移动。

下面我们来观察插入符如何在同一个块的输入和输出之间移动。为此，上一个插入符连接器的行和列会被暂时保存起来，因此我们可以看到以下的情况：



当移动到块体上的时候，插入符连接器的行号没有改变，它会在下次按下向右键时被使用。同样的情况也会被使用在插入符连接器的列号，我们会在下面讨论。

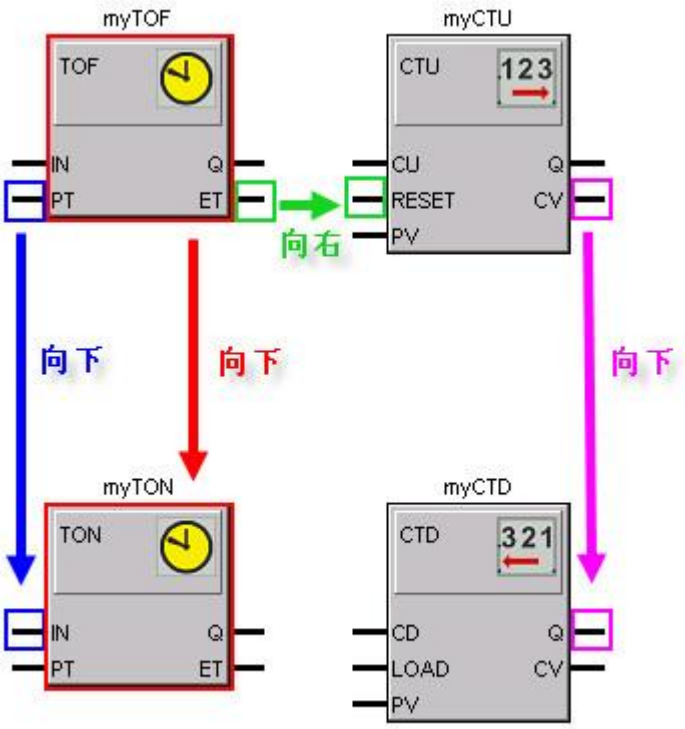
如果想更快地在有功能块的网格单元之间移动，您可以使用 ALT+向上/ 向下/ 向左/ 向右来之间跳转到块体。



在网格单元之间移动

我们来看如何在有功能块的网格单元之间移动。在移动到一个空的单元或只有一个注释的单元时，插入符会定位于该注释或整个网格元素，而不考虑起始位置。

在有功能块的网格单元之间移动时，基础规则仍是前面所提到的对插入符连接器的行号和列号进行缓存的规则。



如果没有与当前连接器的行或列相匹配的连接器（如 JMPC），那么插入符会移动到块体。

沿着连接移动

插入符能够从输出连接器跳转到所有和该连接器相连的输入连接器。您可以从任何一个输入连接器跳转到所有和它相连的输出连接器，反之亦然。

注意：下一个输出连接器为从时间上来讲下一个和该输出相连的输入连接器。

对于这些动作，我们有以下的右键菜单项：

跳转到数据源：跳转到数据源

跳转到下一个数据目的地：跳转到下一个数据槽

跳转到上一个数据目的地：跳转到上一个数据槽

2.9.5.16.8 插入符的快速移动

Home 和 End

Home 和 End **只针对网格本身**（空栏不包括在内），它们将插入符定位在当前行的最左边或最右边的网格。

Ctrl+Home 和 Ctrl+End

Ctrl+Home 和 Ctrl+End **只针对网格本身**（空栏不包括在内），它们将插入符定位在网格的左上角或右下角。也就是说，在 FBD 中 Ctrl+Home 跳转到第一个网络的左上角，Ctrl+End 跳转到最后一个网络的右下角。

Page Up/Down

使用 Page Up/Down 时，可见的部分总是和一个网格单元的上边对齐。一般会根据可见网格单元的个数来进行滚动。

自动向下滚动

在移动时，可见部分的滚动要保证插入符可见（插入符和编辑器边缘保持一定的距离）。

撤销选择

使用 ESC 键会撤销当前选择，但不改变插入符的位置。

选择多个元素

使用 Ctrl+向左/向右/向上/向下键可以选择多个元素。但是，仍然只有一致和有效的选择才被允许。（例如：块和边界线连接器不能同时被选中）

注意：使用插入符时，无法进行矩形选择（拖拽框选）。

2.9.5.16.9 在插入符位置的直接编辑

如果插入符定位在一个可直接编辑的元素上，该元素会被选中，一旦用户开始输入文字数字符号，它就会被在直接编辑模式下打开。

这里会产生一个关于插入符和选择的歧义：如果已经有另一个可直接编辑的元素被选中，那么直接编辑模式会打开插入符当前所在的元素。

2.9.5.16.10 通过键盘操作插入功能块

用键盘来插入功能块有以下步骤：

1. 用快捷键调出选择功能块对话框。

2. 选择要插入的功能块类型。

3. 关闭选择功能块对话框，插入模式被自动激活。

4. 插入符必须移动到要插入的位置，以便插入功能块。插入符只允许在网格单元之间移动。插入符会入 空网格单元中的插入符 中描述的显示出来（即使该单元中已经有功能块）。

5. 如果插入符定位在不允许插入功能块的位置，那么它会根据情况改变它的图形。

6. 如果选择了有效的插入位置，那么用空格键就可以插入块，并且插入符定位在块体上。

7. 如果选择了无效的位置并按下空格键，那么一个事件会被发送到自动化套件，说明插入操作不成功。插入操作会被终端，并显示标准插入符。

2.9.5.16.11 用键盘移动和复制功能块以及空栏连接器

1. 功能块可以用 Ctrl+Shift+向上/向下/向左/向右键来移动。一旦放开 Ctrl+Shift 键，在当前位置的插入操作就完成了（等同于用鼠标移动块或空栏连接器时放开左键）。在无效位置时插入符的图形和插入功能块时的相同。

2. 空栏连接器可以用 Ctrl+Shift+向上/向下键来移动。放开 Ctrl+Shift 键，在当前位置的插入操作就完成了（等同于用鼠标移动块或空栏连接器时放开左键）。在无效位置时插入符的图形和插入功能块时的相同。

3. 块和空栏连接器的复制可以通过复制和粘贴来完成，因此你只能在网格单元之间移动。

2.9.5.16.12 用键盘插入连接

为了用键盘插入一个连接，两个可组合的元素（块连接器和/或空栏连接器）需要用插入符标出。之后一个新的连接就可以用快捷键或菜单项 插入->连接 来插入了。

更快捷的方法：选中两个或多个可以被连接的连接器后放开 Shift 键，连接就会自动被加入。

2.9.5.16.13 移动插入符的键盘组合

Alt+方向键： 块之间的快速移动

Ctrl+方向键： 多项选择（例如连接器或功能块）

Alt+Ctrl+方向键： 功能块的快速多项选择

Shift+方向键： 放开后选中

Shift+Alt+方向键：快速移动时，放开后选中

Ctrl+Shift+方向键：移动功能块或空栏连接器

2.9.6 复合块

2.9.6.1 复合块：介绍

复合块是构建您的应用程序的一种方法。

CFC 编辑器的工作区域限制在一页的宽度。通过选择页的大小，您可以确定水平方向放置的块的数量。在垂直方向，您可以无限添加功能图。

虽然实际上功能图的长度没有限制，但打开浏览太长的图时会觉的图很松散。复合块是优化应用结构的有效的方法，在一个复合块中隐藏了几个逻辑上相关的块。

复合块中块与块之间的信号对外是不可见的。仅那些进入或离开复合块的信号才是可见的。

在屏幕上，双击复合块来查看它的内容。使用***视图 -> 水平向上***或用工具条去获得复合块被激活的位置。

复合块可以嵌套，比如在一个复合块内，您能够定义或使用别的复合块。复合块的内容能够被编辑也可以增加或删除块以及重新连结或删除连结。

在屏幕上，复合块的最后一个输入和输出的端子比别的端子都短，因此，从别的块上您可以很容易的区分复合块。

2.9.6.2 创建复合块

要创建一个新的、空的复合块。

1. 选择 插入-> 复合块...
2. 鼠标光标发生改变
3. 在您要插入新的复合块的地方点击鼠标。

现在首先双击它，填充复合块，并且像别的功能图一样编辑它们。或者首先给复合块增加输入和输出，然后使用已提供的输入和输出编辑它的内容。

当您在图上用完空间时，或通过高级分组增加程序的可读性，您可以将一些已连接的块转化为复合块。

1. 选中块
2. 选择 ***插入-> 复合块...***
3. CFC 编辑器将提示您是否将块转化为复合块。
4. 选中的块会被删除代之于复合块。这些块之间所有的信息随块一起删除，和别的块之间的信号将被保存或转化为复合块的接口信号。

注解：

目前还不支持一组块转化为复合块的过程的恢复。

2.9.6.3 对一个复合块增加一个输入或输出

您可以像别的功能图一样编辑复合块的内容。当您需要提供额外的输入和输出时，您需要改变复合块的接口。您能够从复合块的周围（由上向下）或在复合块内（由下往上）做这些事情。

由上向下：

1. 任何复合块的最后一个端子都比别的短。这总是最后一个端子，一个在左边作为输入，一个在右边作为输出。
2. 连线最后一个输入和输出
3. 当您使用最后一个端子时，它将显示全部的长度，另一个稍短的端子将被增加到最后。

由下往上：

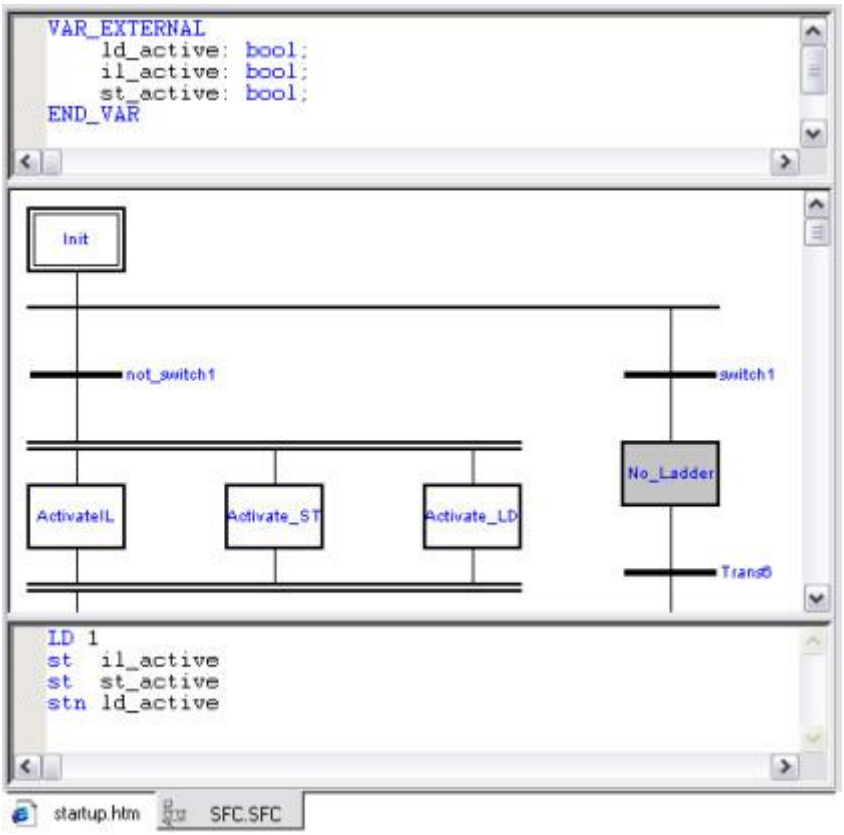
1. 双击您要增加端子的复合块
2. 在复合块中连接一个块，从左边的空栏到右边的空栏（取决于您是否要创建一个输入或输出）。
3. 在端子上点击右键打开属性对话框。
4. 标记出选项 CFC 连接器 和 复合块连接器 为它命名并单击 OK 关闭此窗口。

如果您点击一个层次上的图标，您可以看到另一个稍短的，未被使用的端子已增加到复合块上。

2.10 SFC 编辑器

2.10.1 SFC：介绍

SFC 编辑器在 OpenPCS 框架之中，分为三个部分。在 SFC 编辑器的上部，输入 POU 的声明。在 SFC 编辑器的中部可以编辑图，在下部可以编辑元件的代码。



注意： 代码语言是在新建程序时一次性选择。当前版本支持 IL 和 ST.

2.10.2 连续功能图的元素

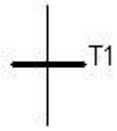
SFC 平面图是对公式化的工艺过程控制流的一个工具，它通过状态的变换来定义。每一个状态的转移都结合了特定的情况。

SFC 提供了下列语言元素：

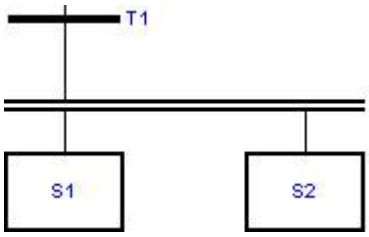
步：每一步包含了许多动作，动作包含代码片段。一个被执行的步，称之为激活。如果激活一个步，包括的动作将被执行。

一个步可以被以下激活：以前转换的切换、一个跳转元素、设置初始标识。

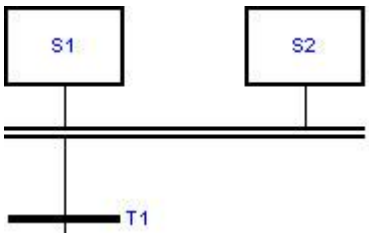
初始化步：在程序的开头激活初始化步。在程序前面执行开始处可以被初始化步标识。



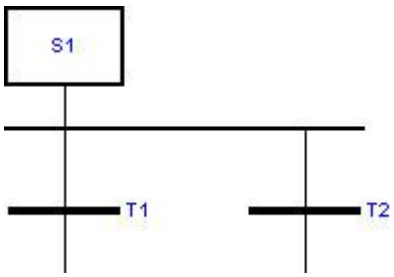
转换：通过转换来控制程序流。如果转换状态是真，一个转换将动作，同时激活先前所有的步。一旦状态转换，解除所有以前的步的激活，并且激活所有紧接着的步。



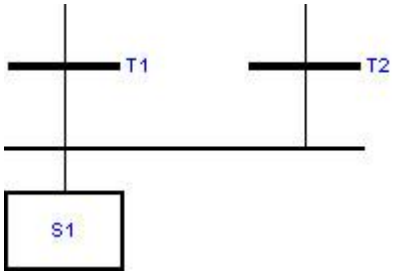
同步顺序：在一个平行链上，一个转换可以同时激活多个步。如果 T1 转换的所有先前的步已经处于激活状态，并且转换条件为真，就会激活所有紧接的同步顺序(比如 S1，S2)。



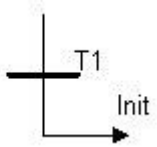
同步顺序的汇集：同步顺序的路径重新聚集在一起。如果所有先前的步处于激活状态(如 S1，S2)，在转换 T1 为真时，解除先前步的激活并激活紧跟的下一步。



分叉路径：顺序步的选择，如果先前步（S1）处于激活状态，从左到右对转换（T1，T2）进行求值，最先具有值 TRUE 的转换停止先前步的执行，并激活相关联的后继步。



顺序的汇集：结果选择分歧链被转换为一个状态。如果其中一个状态为真，解除它前面的步的激活状态并激活它的后继步。



跳转：在另一位置继续程序流，如果先前的状态（T1）为真，跳转到程序的转移处。

2.10.3 步和初始化步

当且仅当处于激活状态时，循环执行步的代码。原则上可以这么说，代码被一个循环所包围，即如果还没有发生转换就处于这个循环，如果激活了下一个转换则跳出这个循环进入下一个步。如果一个状态被激活，它的代码至少执行一次，初始步在程序开始时总是处于激活状态。标准的，进入程序的入口点是图中的第一个元素（初始步）。通过激活属性窗口里的控制箱 初始化步（initial step），每一个步都能转换成初始步。

步的名称必须符合下面的语法：步名必须以字母（ a - z 或 A - Z ）开头，第二个字符可以是字母或者数字（ 0 - 9 ）或者下划线（ _ ）。正确的步名： Step1 和 S_1 。无效的步名： _Step1 、 1Step 和 Heater off 。

步名最长不能超过 31 个字符。您可以在那个跳转主题处获得更多有关信息。

2.10.4 转换

转换负责把以前被激活的步的状态转到下面的状态。转换以布尔变量真的形式描述了可能的转换（转换条件）。

必须写出转换的代码，以便于在代码结尾的当前结果是布尔型的。当且仅当累加器的值为真时，转换才发生。通过当地定义的变量来实现和别的 SFC 元素通信。

例（IL）；（*当温度大于 70 度时转移关闭*）

```
LD variable of temperature

GT 70

例 (ST)； (*当温度大于 70 度时转移关闭*)

 IF variable_of_temperature > 70 THEN

  result := true;

 ELSE

  result := false;

 END_IF

 trans1 := result;
```

注意:

转换代码的最后一行应当始终装载一个逻辑值(在 IL)中， 或赋值变量为期望结果（在 ST 中），这些变量的命名相当于相对转换（比如；trans1）。

2.10.5 跳转

跳转是 SFC 程序中用来控制程序流的元素。到现在为止所介绍的元素，步的激活总是直接从上到下。对循环的程序或相似的操作，激活以前的步是有必要的，跳转则提供了这种功能。

跳转之前的元素是转换，跳转的目标总是一个步。通过给定跳转状态的名称来确定跳转的目标。如果一个步是跳转的目标，其名称必是唯一的。如果跳转目标没有或不止一个有效的，相应的错误信息会在语法检查过程中生成。

为了保证 SFC 图的连续性，跳转的插入仅作为分叉路径的最后一个元素。

2.10.6 SFC 在线编辑器

首先确保您已经打开资源窗格中代码的实例，而不是文件窗格中的类型代码。通过移动鼠标光标到代码处，您可以容易的判断：如果是实例树中的代码，鼠标光标应该是活动的。

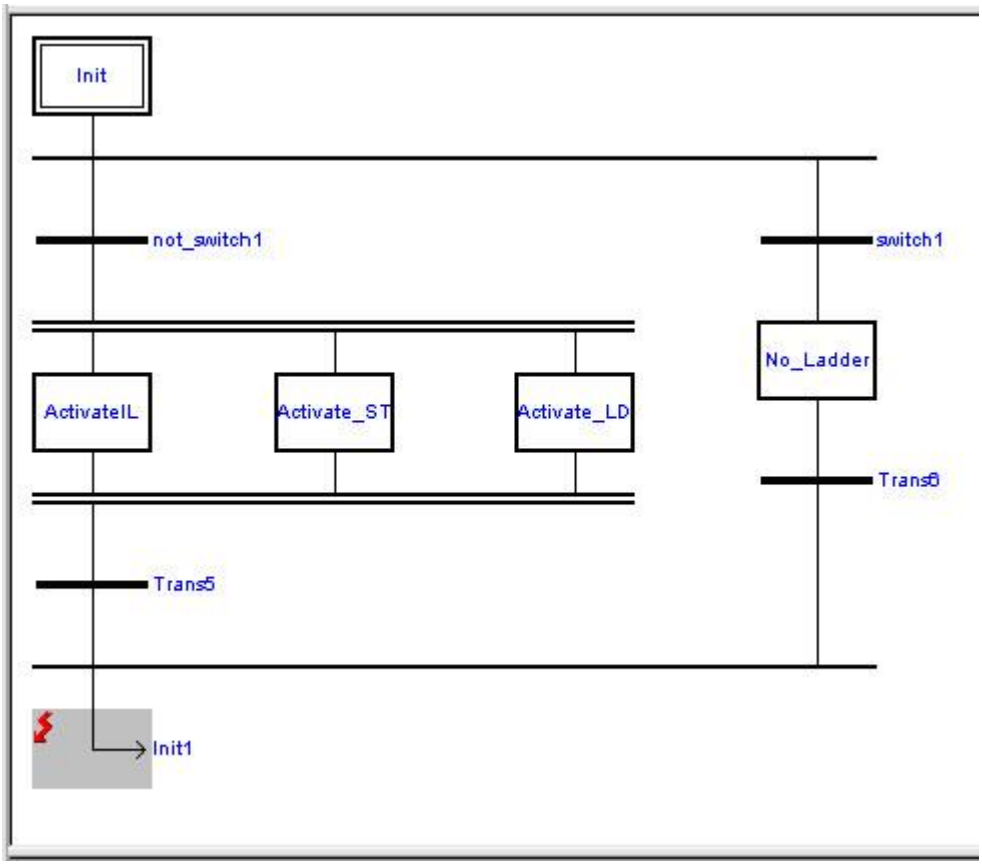
当您启动 SFC 编辑器的在线模式时，它会自动显示状态信息。小红框会显示在所有激活的步骤中。这个信息会以尽可能快的频率更新。然而，目标控制器的频率可能会太快以至于不能够显示所有中间代码的状态。当在线的时候，您能够总观步骤内容和转换过程，但是不能看到状态信息。如果步骤内容或转换过程变的足够复杂，以至于需要调试的话，强烈推荐把它移到一个独立的功能块中。

OpenPCS 支持在线编辑，更多信息请参阅用户手册中的在线编辑。

2.10.7 一般错误

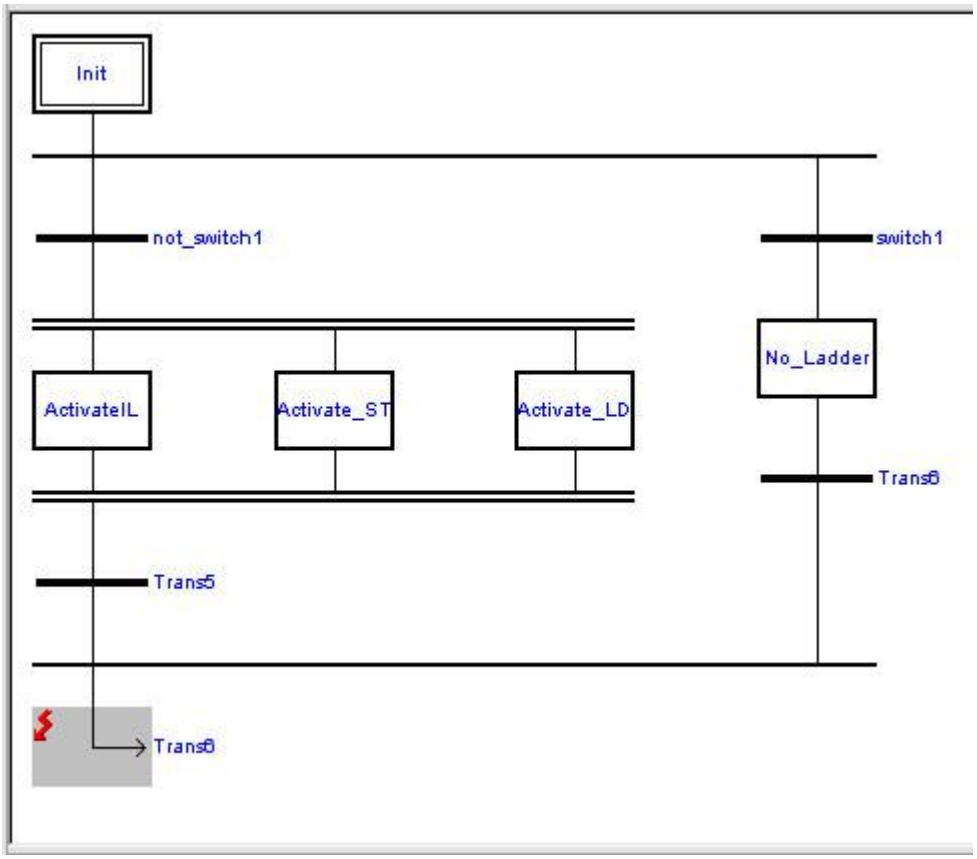
只要工程被保存，在 SFC 编辑器的视图中的错误由一个红色的闪电符号标识 。最普遍的错误会被通过 ControlX 样例的外观在 OpenPCS 中表示出来。

1. 跳转目标非法



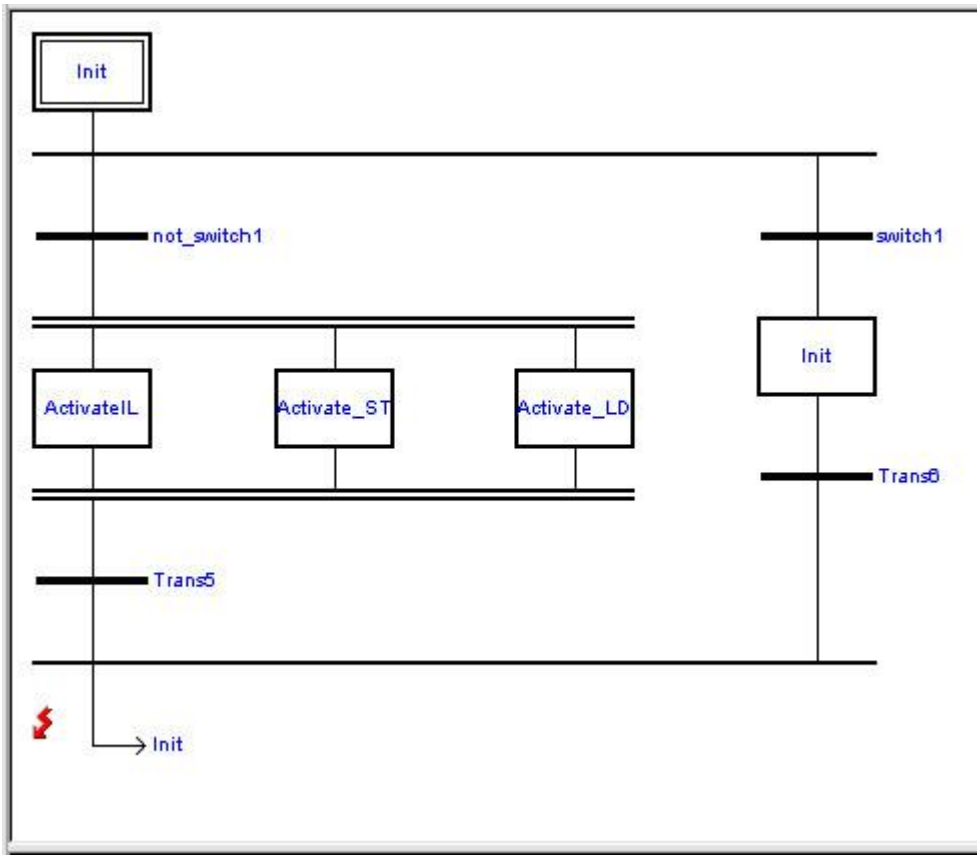
Init1 被定义为一个在此工程中并不存在的目标。在提示窗口，OpenPCS 会提示：

错误： Init1 不是一个合法的跳转目标 （称为 Init1 的步不存在）。



只有步才是合法的目标，因此若一个转换被定义为目标， 同样的消息将会被提示。

2. 非唯一的目标



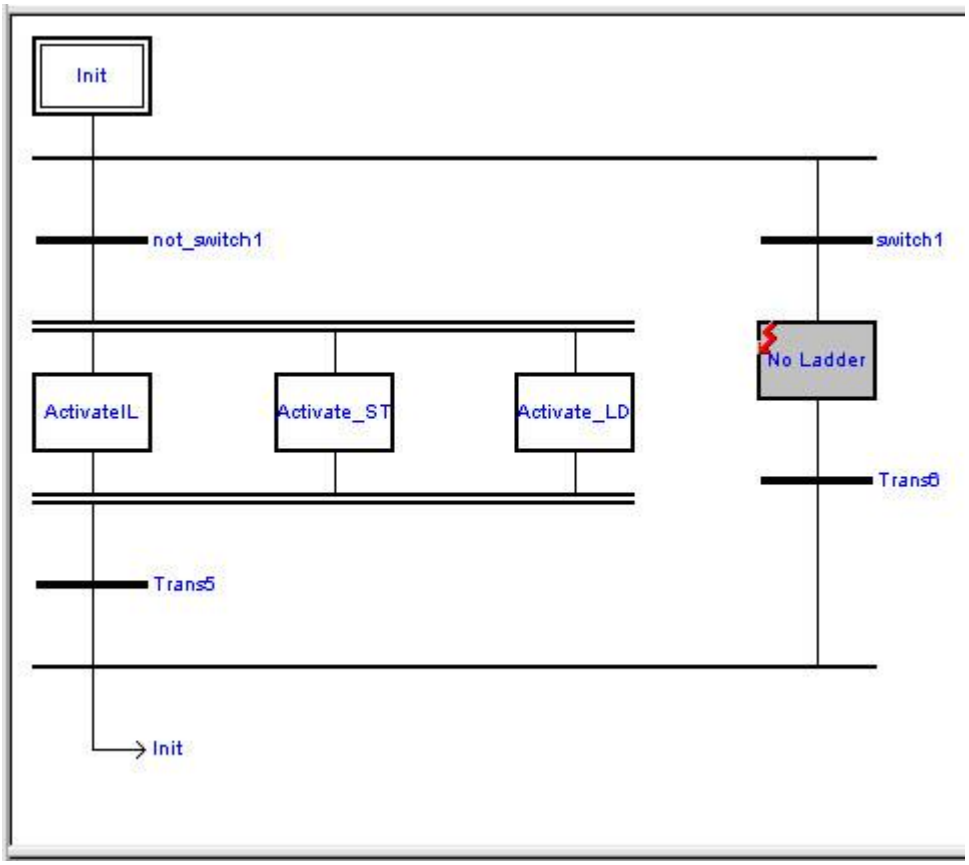
有两个步被命名为 Init。因为步被选择为目标，OpenPCS 无法判断应该使用哪个步，由此以下消息将会被输出：

错误： 跳转目标 Init 不明　下列两个步有同样的名称：

☐\CONTROLX\CHART.SFC(10,100,4)：对象：步 Init

☐\CONTROLX\CHART.SFC(10,100,5)：对象：步 Init

3. 非法的名称



在步名称中不允许有空格。

错误：对象 No Ladder 的名称中含有非法字符。

2.10.8 择元素

2.10.8.1 标记单个元素

单击左键标记出一个元素，其代码将显示在下面的文本框中。

使用方向键(, , ,->)，可以移动单个元素到邻近的元素。

2.10.8.2 区域标记

使用鼠标或键盘首先标出一个元素

按住 shift 键，用鼠标单击另一个元素，选择一整块图形。

或按住 Ctrl 键，用鼠标点击别的元素加入到已选择的元素中。

在这个版本中，不能用键盘来区域标记。

2.10.8.3 标记几个元素

为了在 SFC 图中几个元素上执行一个函数，所有相应的元素必须被标出。

使用鼠标或键盘首先标出一个元素

按住 Shift 键，用鼠标单击另一个元素，选择一整块图形。

或按住 Ctrl 键，用鼠标点击别的元素加入到已选择的元素中。

在这个版本中，不能用键盘来区域标记。

2.10.9 高级主题 SFC

2.10.9.1 例外处理

在执行一个 SFC 程序时，可能会碰到需要一个特殊转换执行逻辑的情况。带有附加标准元素（转换，跳转，步骤）的 例外处理 模式是可能的，但是会减少程序的清晰度。

为了解决这个问题，SFC 编辑器提供了 IL 指令宏，有目的地使步骤激活或不激活。下面是一些有效的的指令：

```
@ACTIVATE_STEP(StepName)    /* 激活一个步骤 */
@DEACTIVATE_STEP(StepName)  /* 不激活一个步骤 */
@DEACTIVATE_ALL_STEPS()  /*不激活所有的步骤*/
```

这些处理内部执行控制，因此指定的步骤将在附加的下一个循环中（不）激活。

注意**:**

如果上面的命令用在 IL 代码中，不确定的或不可执行的网络可能会出现！

注意:

当前版本不支持在 ST 编辑器中的上述命令。.

2.10.9.2 找错误位置

编辑->Goto IL Line: 在 SFC 平台上，根据产生的 POE 文件中的错误行号，用这个命令找到需要的代码片段。这个错误行号是在 OpenPCS 系统中编译时记录下来的。

2.10.9.3 使用不同于 IL/ST 的其它语言

SFC 是用 IL 或 ST 表来写步骤和转换过程代码的。用别的语言（梯形图，CFC，FBD），把代码写到一个功能块（或函数，如果可行的话）中，并且从步骤或转换过程中调用一个功能块实例。

记住要在您的 SFC 程序的声明部分声明一个功能块实例。

根据您的应用程序的需要，可以在不同的步骤和转换过程中重新使用一个这样的功能块实例，或不同的功能块。对于更复杂的代码，这将不仅使您的应用程序结构更清晰，而且能减少内存的消耗和增加可调试性。

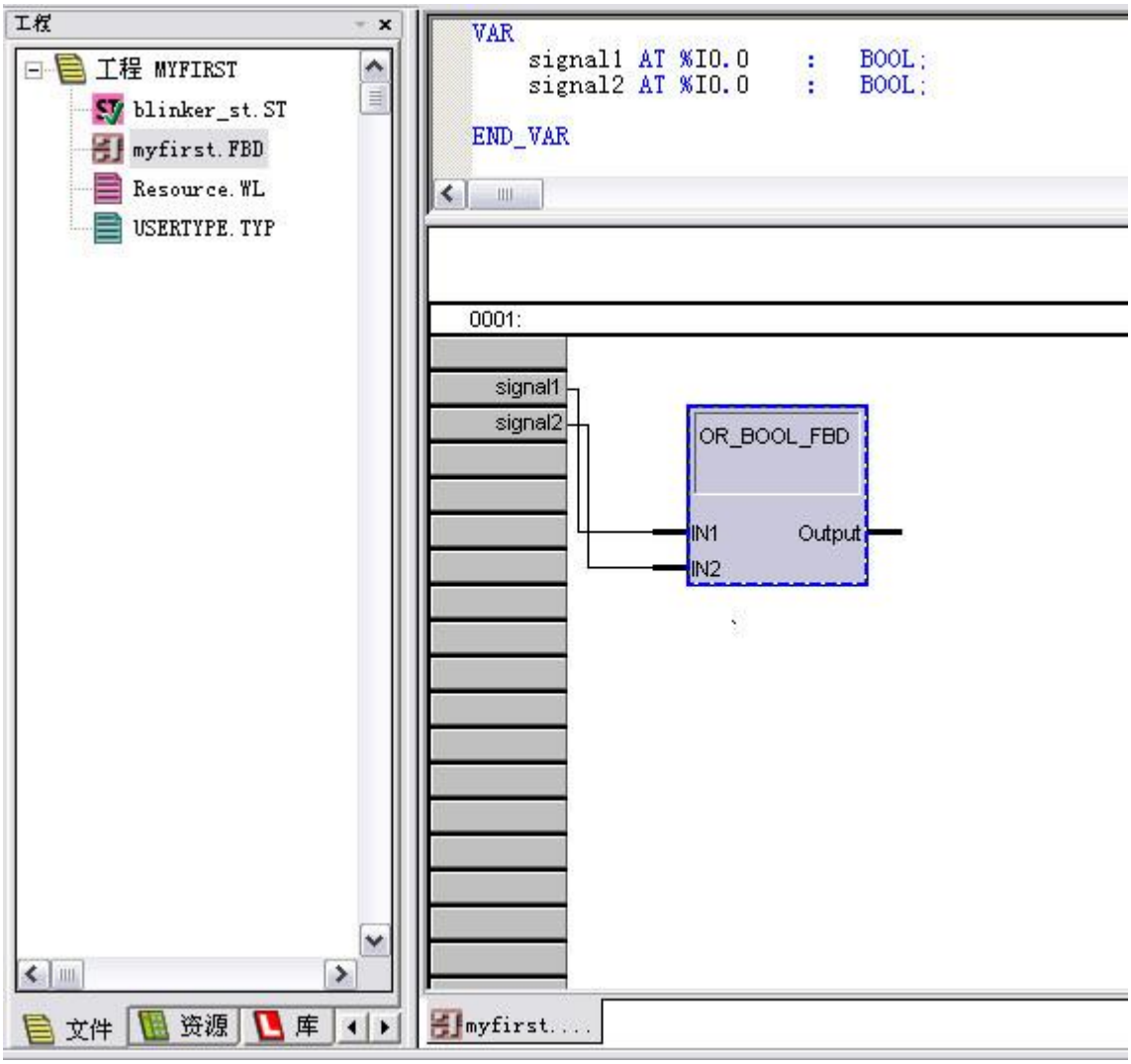
2.11 FBD 编辑器

2.11.1 FBD 编辑器介绍

FBD 编辑器（功能框图编辑器）一种工程工具，用于图形化创建自动化程序。

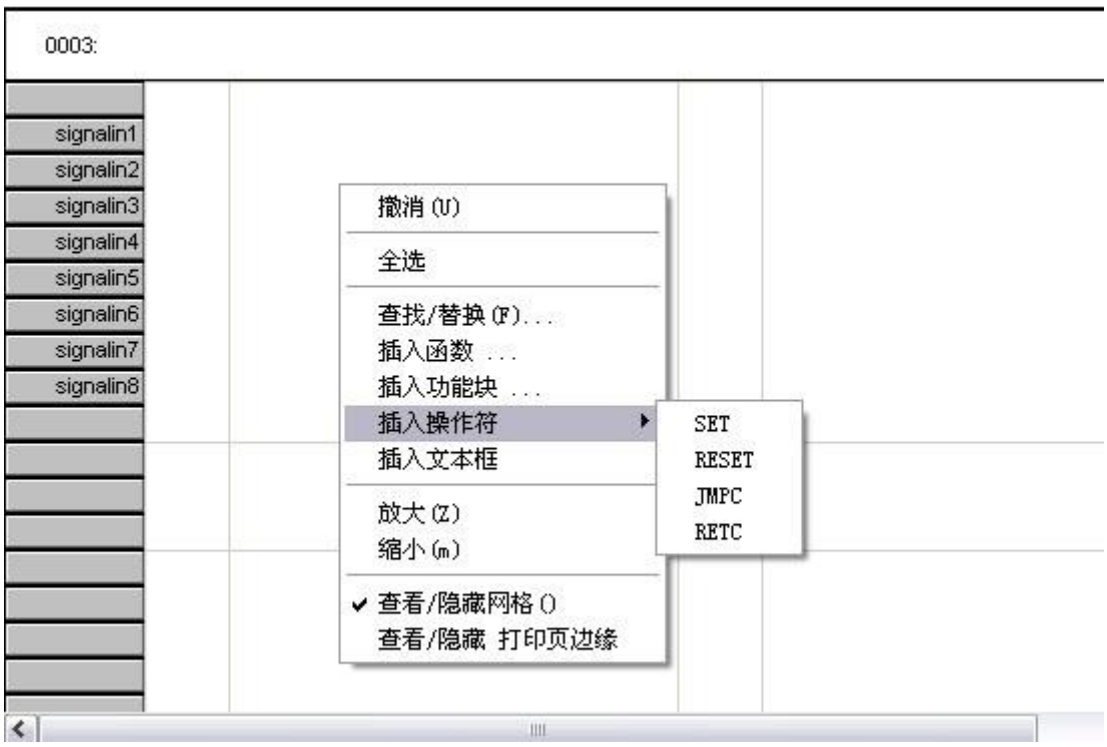
FBD 编辑器在 OpenPCS 框架中。在 FBD 编辑器的上半部分，输入 POU 的声明。

FBD 图的主要元素为块（函数，功能块，操作符等）和空栏（左边及右边）。块可以任意放置在图中，空栏提供了与 IEC61131 变量和连接的链接。连接用于将一个输出（块或空栏）同一个或多个输入（块或空栏）连接起来。



2.11.2 使用块工作

要增加块到您的 FBD 图，使用 插入◊块，可以选择固件或用户定义块，使用 插入◊功能块或用户定义功能块，或者插入->文本块。也可以在指令窗格中右击鼠标然后从快捷菜单中选择 插入函数 插入功能块 插入文本块 或 插入操作符 。



鼠标光标会发生变化，这时电击要插入块的地方。

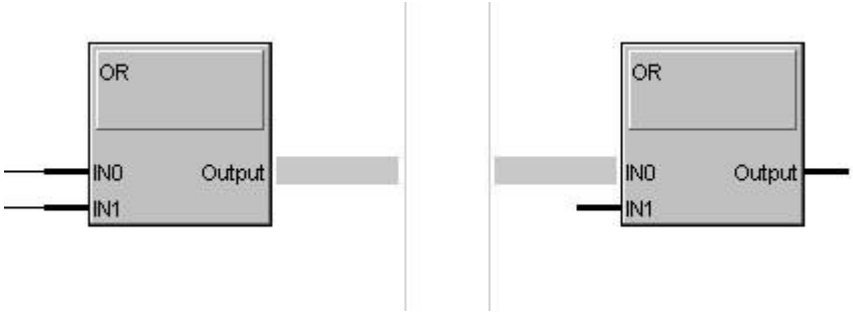
要重新组织块，选择块然后拖放到新位置即可。当增加一个新块或移动一个已存在的块时，FBD 编辑器会适当的移动其它块来给它们腾出位置。

要删除块，选择它们然后按下 DEL 键。

2.11.3 连接 FBD

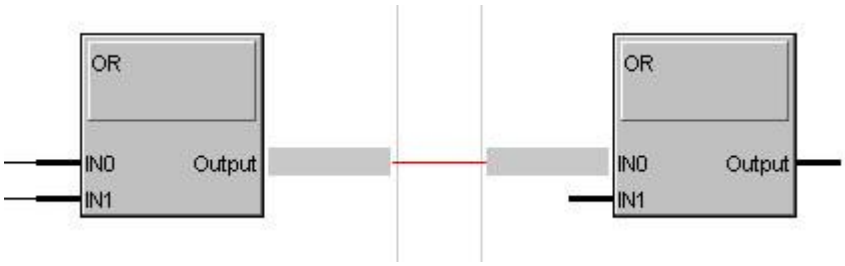
为了连接变量，功能，功能块等，仅仅左击空白条或者功能标识上的节点。通过连接 插入 菜单中的 连接 选项或者利用快捷键[CTRL] [B]来突出空白条或者节点。新的连接选中后将是红色的，没有选中的是黑色的。

1. 左击



2. 实施连接

3. 查看连接

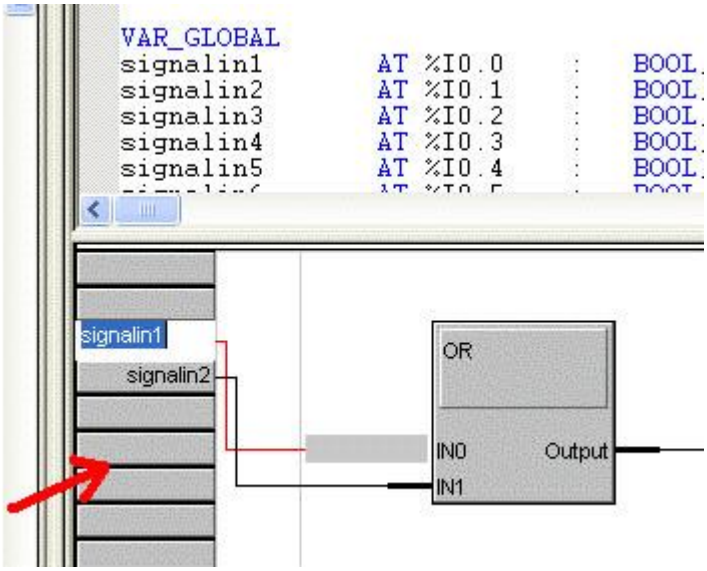


选择一个连接仅仅左击其中一个末端（空白条或者节点）。选中的连接可以直接通过[DEL]删除或者右击空白条或者节点来选择 删除 。

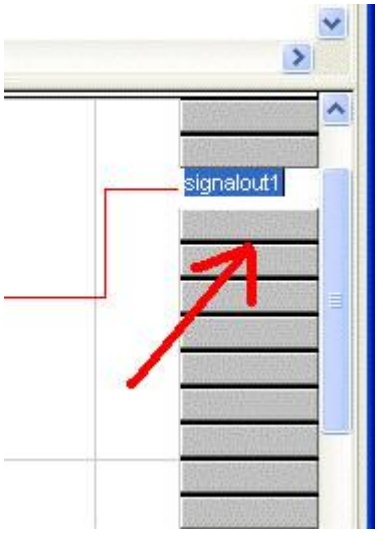
2.11.4 边际条

左右的边际条用来连接 FBD 包含的逻辑单元和输入（左）输出（右）变量。只需双击边际条然后输入变量名称。

左边际条；



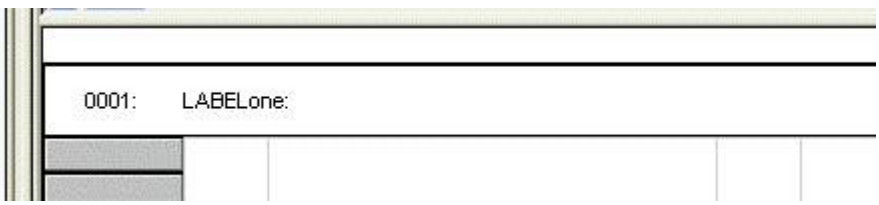
右边际条



2.11.5 高级

2.11.5.1 使用网络工作

可以采用网络来组织您的工程。每个网络自动得到一个四位的阿拉伯数字作为标识显示在网络的顶端。单击或双击那个区域允许您给那个网络重新标柱。

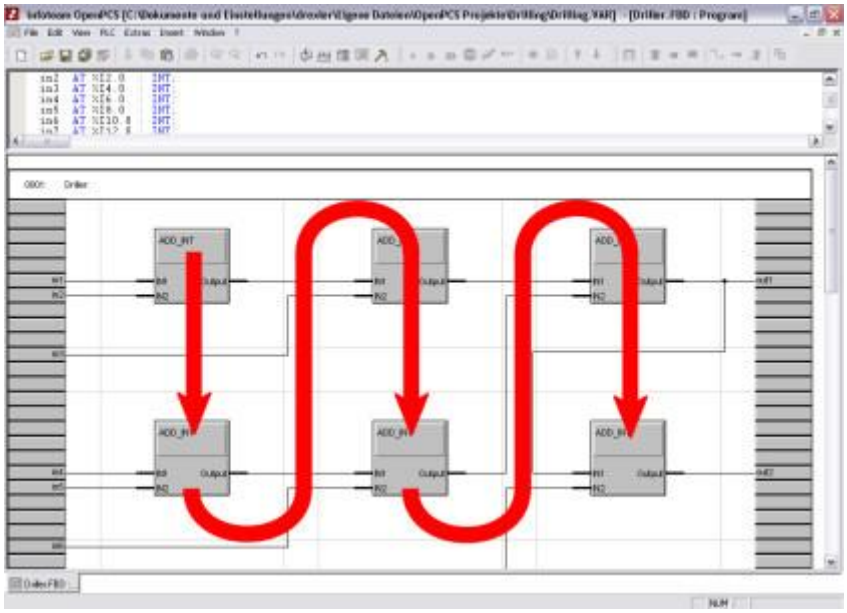


单击鼠标右键，出现一个选择菜单，允许您增加一个新的网络，或者跳到下一个或前面一个网络。

2.11.5.2 执行顺序

网络由上至下依次执行。

在每个网络中块的排列直接关系到它们的执行顺序。块的执行顺序是第一栏由上至下，然后第二栏由上而下，依次类推。要修改执行顺序必须重新排列块。



功能块的替换

FBD 编辑器支持用另一类型的功能块替换厂商和用户定义的块。 选定要替换的功能块，在菜单中选择编辑-> 替换功能块。一个类似插入->功能块 的对话框会弹出，用户可以从一个固件的和用户定义的功能块类型的列表中选择期望的功能块的新类型。

除此之外，用户可以选择选项 自动替换当前企划中所有实例 ，此选项被选中，当前 FBD 中所有与所选功能块类型相同的功能块，即使没有被选定，也会全部自动替换为新类型。

选择了一个新的功能块类型之后，弹出另一个对话框，使用户在替换之后重组旧块类型的连接器，形成新块类型的连接。表格左栏显示旧块类型的连接器，以及连接器（1*）的类型(VAR_INPUT/VAR_OUTPUT)。表格右栏显示合适新块类型的连接器列表。用户可以匹配相应的连接器到旧块类型连接器。注意，每个新功能块的连接器只能被匹配一次。

若一个连接器无法或不应重新连接，请选择选项 不自动重新连接 。

电击 确定 之后 FBD 编辑器替换所有旧类型的功能块为新类型，并根据连接器匹配表重新接线。

(*1): VAR_IN_OUT 连接器会在连接器列表中出现两次：一次为 VAR_INPUT& ， 另一次为 VAR_OUTPUT&. & 标记连接器实际上表示的一个 VAR_IN_OUT 参数

2.11.5.3 在 FBD 中找错误

FBD 编辑器会导引您到一个错误附近，双击鼠标相应的错误信息会显示在 输出窗口 中。

2.11.5.4 FBD 在线编辑器

首先确定您在资源窗格中打开了相应编码实例，而不是在文件窗格中。

用在线模式打开 FBD 编辑器，功能块，连接，以及边际条上的值会自动显示。

如果在线编辑器无法从运行系统中取得一个变量的值，在该位置会显示 -!- .

2.11.5.5 CFC 和 FBD 编辑器的键盘操作

请参看 OpenPCS 工具->CFC 编辑器->高级主题 CFC->CFC 与 FBD 编辑器的键盘操作。

或由此开始阅读：键盘使用基础

2.12 测试与调试

2.12.1 测试与调试：介绍

测试与调试是维护 OpenPCS 所有在线操作的工具。用 T+C（测试与调试）工具可以监视变量的值，启动和停止您的控制器，在运行应用程序时改变在线块。

2.12.2 启动和停止

测试与调试支持三种不同的方式启动应用程序： 冷启动 将会重新设置所有变量的初始值， 热启动 不会重新设置任何变量，然而 温启动 将只会重新初始化没有声明为 RETAIN.的变量。

2.12.3 监视变量

在测试一个程序时，知道变量的值或哪个变量值发生错误是很重要的。因此，我们有可能要监视变量。

转到资源面板，打开您想要监视的变量的任务分支。双击您要监视的变量，变量就会显示在 测试与调试 窗口中，在这个窗口中还显示了实例路径、类型、值和状态。当程序在 PLC 上执行时，这些变量会不断的更新。如果 OpenPCS 不能从运行时系统中得到一个变量值（举例来说，这个变量在当前运行程序中不是有效的），那么将显示 -!- 。

要从列表中删除变量也有三种可能的方式。单击鼠标左键以标记这个变量，然后在工具栏上单击相应的符号，或用 del 键，或在 编辑 菜单中选择 **删除变量** 条目。

双击一个数组变量将打开一个对话框，您应该输入您要查看的索引。多维数组的索引必须用逗号分开。

2.12.4 设置变量

在测试情况下，要改变控制程序的行为，您可以给变量设置特殊的值。在 T+C 工具中标记这个变量，并且选择菜单条目 ***PLC->设置变量*** ，或者在 T+C 工具中直接单击这个变量。输入新的值，点击 设置 按钮接受这个新值。也可以参看强制变量这部分。

2.12.5 强置变量

除了监视和设置变量外，OpenPCS 还支持强置变量。如果强置了一个变量，则这个变量值在每一次循环结束时（写输出之前）都会重新设置为指定的值。强置是由设置变量对话框中的三个标签按钮（ 设置 ， 能强置 ， 不能强置 ）控制的。

在测试与调试窗口中的 强置 列，OpenPCS 将会显示一个变量在当前是否能够被强置。按 OK 按钮执行的动作取决于三个按钮（设置，能强置，不能强置）中选中了哪一个按钮。

如果变量当前没有被强置，则 设置 将给变量设置一个指定值。如果变量被应用程序修改，这也只能是在非常短的时间内有效。 使能强置 将强置变量为指定值。也就是在每一次循环结束时将变量设置特定值。 不能强置 没有这个功能。

如果一个变量被 强置 了，则 设置 不能强置这个变量为特定值，并且只能设置这个变量为特定值一次。 使能强置 将继续强置这个变量，但是是用当前的值强置这个变量， 不能强置 将不能设置这个变量，但是仅仅是不能强置这个变量而以。

请注意：

1. 强置仅仅是在每一次循环结束时重新设置变量。在一个循环过程中的修改是有可能的，并且是不能阻止的。
2. 对于直接表示的变量(AT %☐)其强制不受限制。
3. 从观察列表中删除一个变量会自动取消对改变量的强置。

2.12.6 使用观察列表工作

变量的测试& 调试列表可以保存到观察列表文件。当在线时允许在不同观察列表之间切换。

在工程的根目录中总有一个名称为<*name of your resource*>.*WL* 的缺省观察列表。
当在线时，观察列表是通过主菜单命令 ***SPS*** -> ***保存观察列表到*** 来保存的。

这个保存的观察列表将出现在浏览器文件框中。保存后，所有接下来的变量列表的修改都将被保存在观察列表中。

要恢复一个不同的观察列表，仅仅需要在浏览器中双击打开，或者在浏览器中选观察列表后选择***文件->打开***。

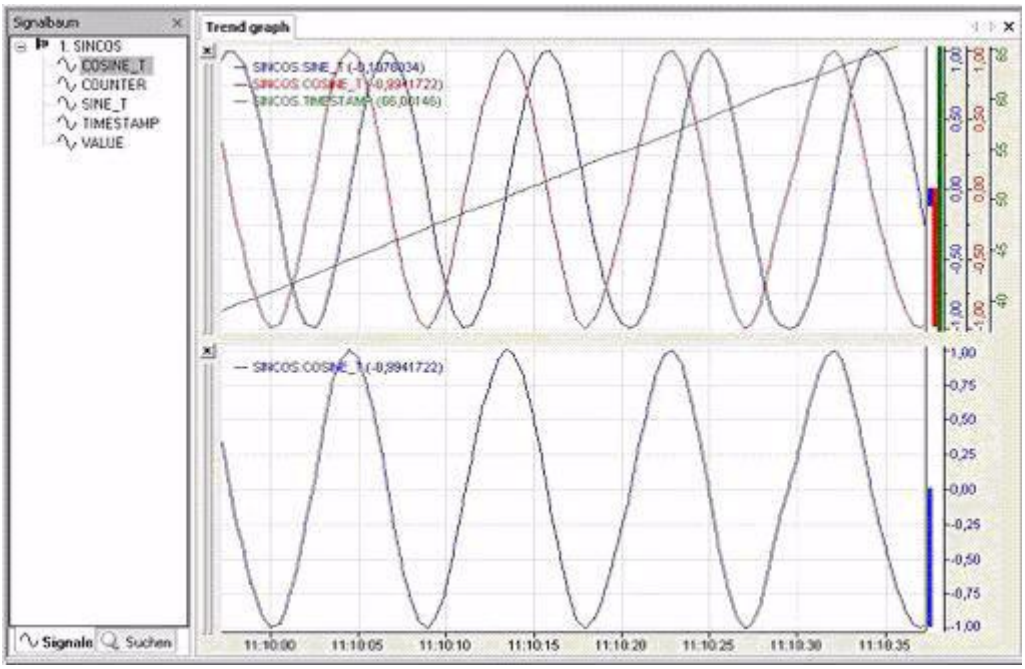
可以通过选择***文件->新建/其它/观察列表***来创建一个空的观察列表。

2.13 控制数据分析器

2.13.1 控制数据分析器

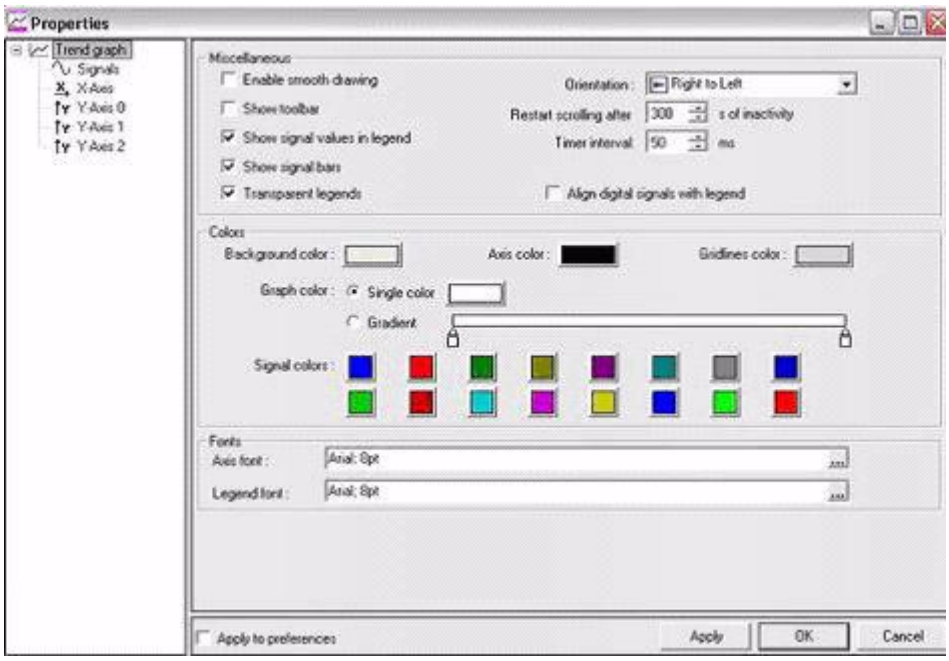
控制数据分析器提供给用户一个绘制 OPC 变量的发展的机会。

控制数据分析器只有在 OpenPCS 在线时才可用并可以通过 ***查看 -> 控制数据分析器*** 来启动。控制数据分析器将会在编辑器区域中显示。



OPC 变量在左侧栏中可见，称为信号树。信号可以根据类型（模拟或数字）或它们定义所在的文件分组。一个新的绘制可以通过双击变量开始。用户可以通过拖曳变量到图中来绘制它的变化。每个图都有其自己唯一的颜色和 Y 轴。

用户可通过右击绘图来打开属性。



用户可以编辑一个绘图和分析器的总体外观的颜色和字体。在信号中用户可以赋予图偏差值并区别设计。轴可以在相应部分设置。

2.13.2 示波器

示波器是一个直角坐标系显示随着时间的信号。标记支持的测量任务。三种模式和触发器支持显示控制。

信号是添加的信号树。触发事件显示在 x 轴中间。每一次的数据是相对于该事件。 Y 轴为中心，为 0。

该工具栏可以切换三种模式：Run，Stop 和 OneShot：

- Run：到达的数据不断检查，以定义的触发器。如果触发事件发生，信号图表以它为中心。
- Stop：触发立即停止，从而触发发生事件无关紧要。视图切换到导航。
- OneShot：一旦触发事件被识别，模式切换到停止模式。

注意：可通过上下文菜单激活工具栏。

2.13.3 触发器

当前触发可通过标题的上下文菜单，触发按钮或示波器属性对话框进入。

有两种不同的触发器：固定和延迟。固定只是一个简单的触发，延迟由三个简单的触发器组成：A，B 和 Reset。延迟仅会为真，如果 B 在 A 后 Reset 前发生。

一个简单的触发器为真根据规定的条件和模式。条件是值和所连接的信号边沿类型。

该值只可能是整数　　信号的数据会被取整。

信号的边缘，可上升，下降，交替（顶点），或 上升或下降 。

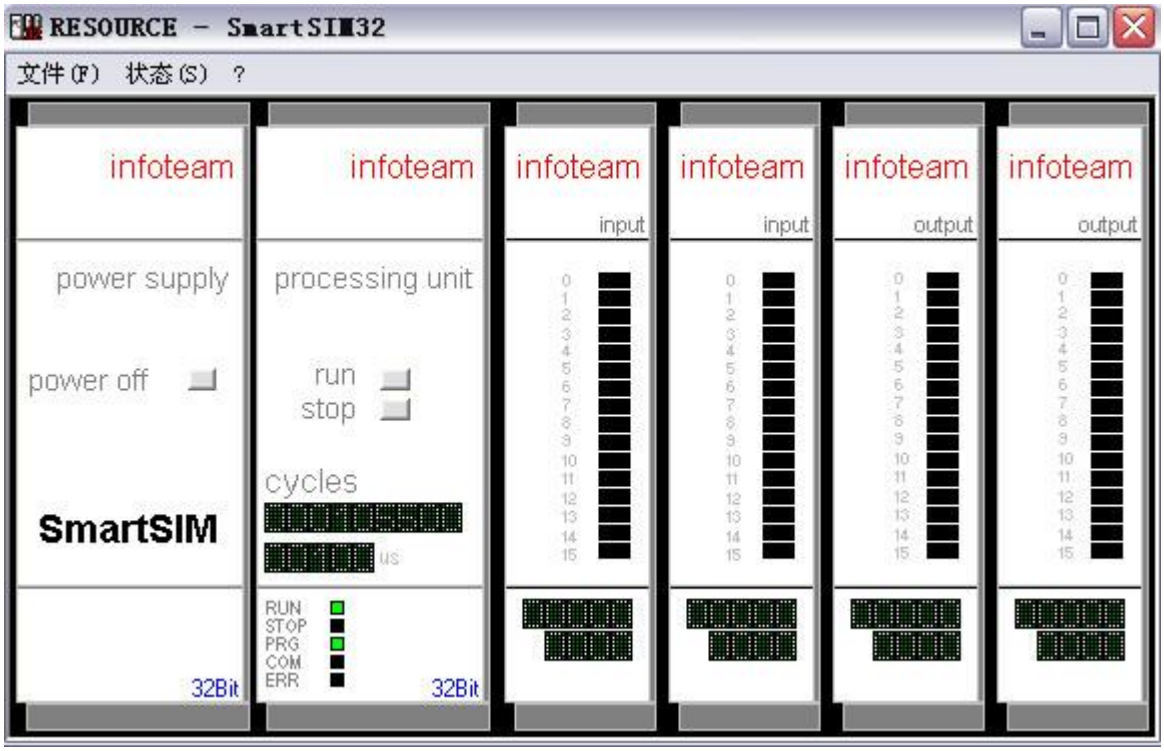
根据不同的模式下，触发被视为：

- 正常：触发器为真，如果所有符合条件的匹配。
- 自动：触发器为真，如果所有符合条件的匹配或一个定义的数据样本数被不匹配条件地处理。
- 自动（级别）：触发器为真，如果所有条件匹配。如果条件不匹配的数据抽取的样本数，该值将自动设置为对信号幅度的一半，流继续在自动模式

2.14 SmartSIM

2.14.1 概述 SmartSIM

为了测试程序，需要一个 PLC。您可以使用和 OpenPCS 一同购买的实际控制系统。为了区分不同 OpenPCS 版本的手册，我们仅在这里使用 SmartSIM32。



SmartSIM32 能够被单独运行，但是一般情况下，只要您在线连接一个资源，这个资源使用连接 仿真 时，OpenPCS 就会自动启动 SmartSIM32。

注意：

为避免 SmartSIM 运行程序存贮在磁盘上，在启动 SmartSIM 时按住 CTRL 和 SHIFT 键。

中断任务

SmartSim 同时也支持中断任务的执行。通过按键盘键可以激发中断任务。因此用户只需选择一个键盘键并将其定义为中断键。在 资源窗格中 选择任务属性，确定 任务类型 设置为 中断 ，并命名中断为一个键盘键.

注意:

1. 允许设置为中断键的符号包括 + , - 和 0x1F to 0x7F 域内的 ascii 代码.
2. 中断任务激发时，SmartSim 要求被定焦.

2.15 OPC 服务器

2.15.1 关于 OPC 服务器

Infoteam SmartLink OPC 服务器是 OPC 客户端和一个或多个基于 IEC61131-3 运行系统的控制器直接的通路。它能够免费从 www.infoteam.de 上获得。同时还提供了一个连接到 infoteam SmartSim 的演示。

2.15.2 远程 OPC 服务器

若要配置一个通过 DCOM 从 OPC 客户端到 OPC 服务器 SmartPLCDA 的远程访问，请遵循以下步骤。

1. 检查必须的前提需求：

   - 操作系统：WindowsXP SP2
   - 密码必须是空的或者 admin
   - 您必须在本地机上有系统管理员的权限以更改 DCOM 的设置。

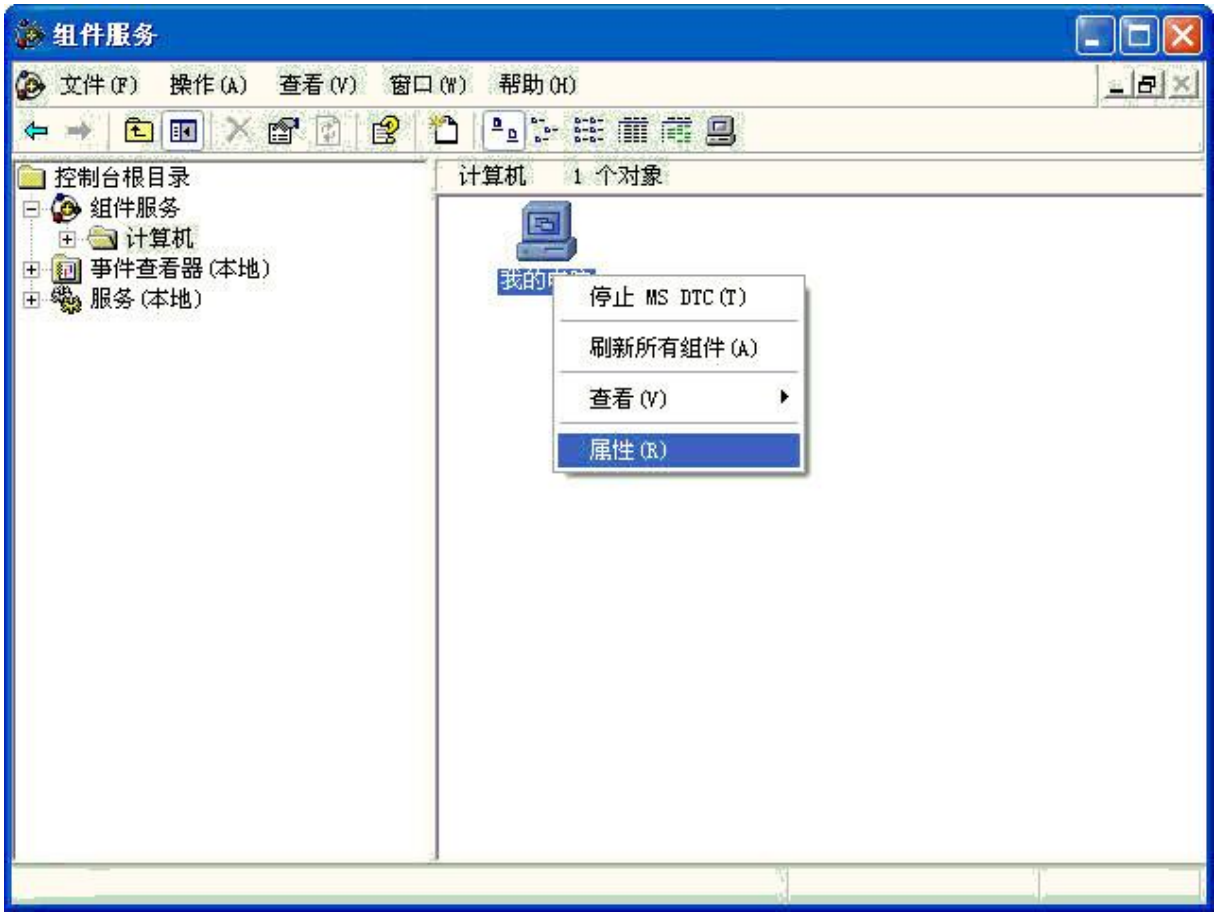
   DCOM 必须在硬件.ini 文件中被启用

   [ENABLE_OPC_PANE]

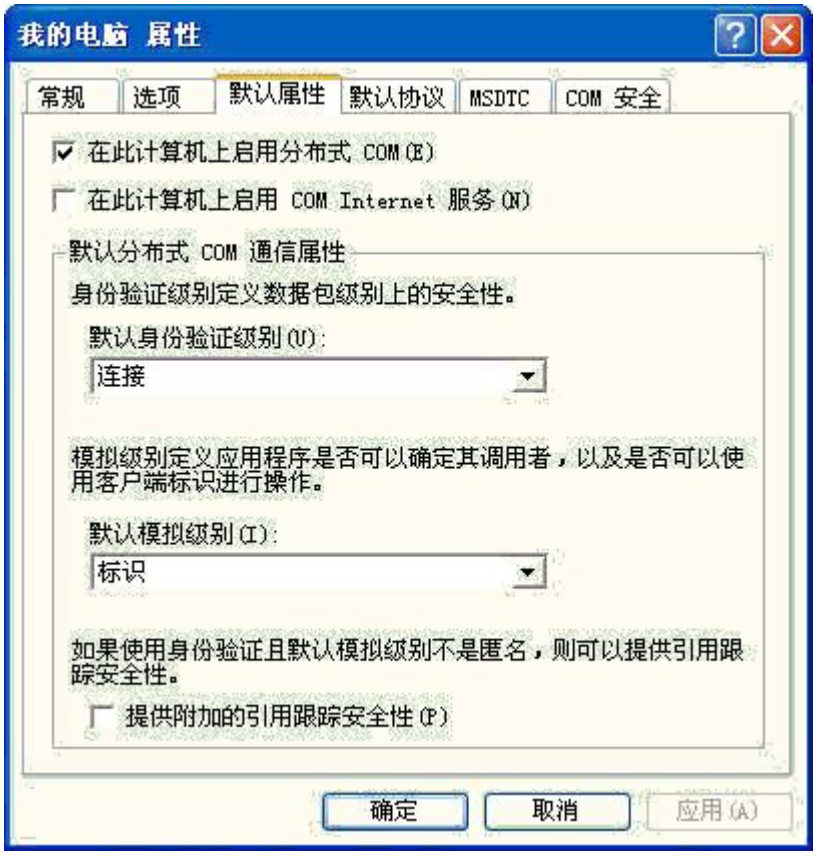
   DCOM=1

2. 设置 DCOM 通讯

   - 选择***开始->运行***并输入 dcomcnfg ，这将会打开 WindowsXP 的服务管理器组件。
   - 前往***控制台根目录->组件服务->计算机***，单击***我的计算机***并选择***属性***。

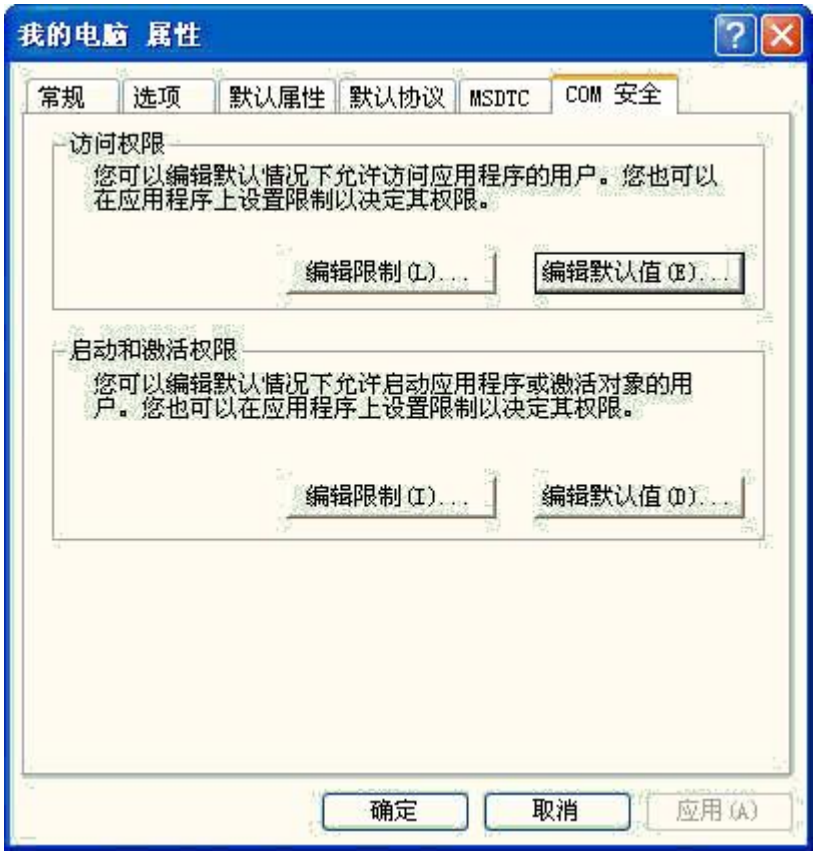


- 择默认属性签，然后选择下面的图中的选项：



3. 设置 DCOM 的权限（默认权限必须更改以建立通讯）

   - 选择 ***COM 安全***签
   - 选择***访问权限***中的***编辑默认值***



- 确认在访问权限窗口中至少包括下列项目：

  i. Everyone

  ii. Interactive

  iii. Network

  iv. System

  v. Domain Administrator

  vi. Domain Users

如果没有这些项目，则须点击***添加***按钮，然后点击***高级***和***立即查找***，就会列出所有的组和用户（若没有，须确定对象类型***组***被选定）。在下一个窗口中选择组并点击***确定***。这样，这些组就被添加了。

- 对于 domain users 请按上面的指导，但在所在地中指定域名。
- 重复上面的步骤以运行以激活所有用户组的权限。最后关闭我的电脑属性以结束更改设置。
- 必须重启电脑以使更高生效。

2.16 在线服务器

2.16.1 在线服务器：概况

在线服务器是所有 OpenPCS 工具和您的控制器之间的通路。它在后台自动运行，发送所有应用程序和控制器之间的数据和命令。当在线服务器运行时，在系统平台中会显示一个小图标  。单击这个图标会终止服务器运行，然而应该尽量避免这样做。

2.16.2 在线服务器设置

2.16.2.1 在线连接：介绍

连接 是潜在的连接到您的控制器的符号名。每个资源都确切地配置用一个 连接 ，并且每个 连接 都绑定给一个 驱动 ，给定一个参数设置给这个驱动。OpenPCS 带有几个驱动，但是默认的只有的一个连接，它绑定给 IPC 驱动。

默认下，下面驱动可用：

IPC -内部程序通讯，用于连接 SmartSIM

TCP -用于 TCP / IP 连接到比 4.3.1 版本旧的目标系统

TCP_432 -用于 TCP / IP 连接到 4.3.1 或者更新版本的目标系统

RS232 -用于串行连接到 4.0 或者更新版本的目标系统

RS232_35 -用于串行连接到比 4.0 版本旧 的目标系统

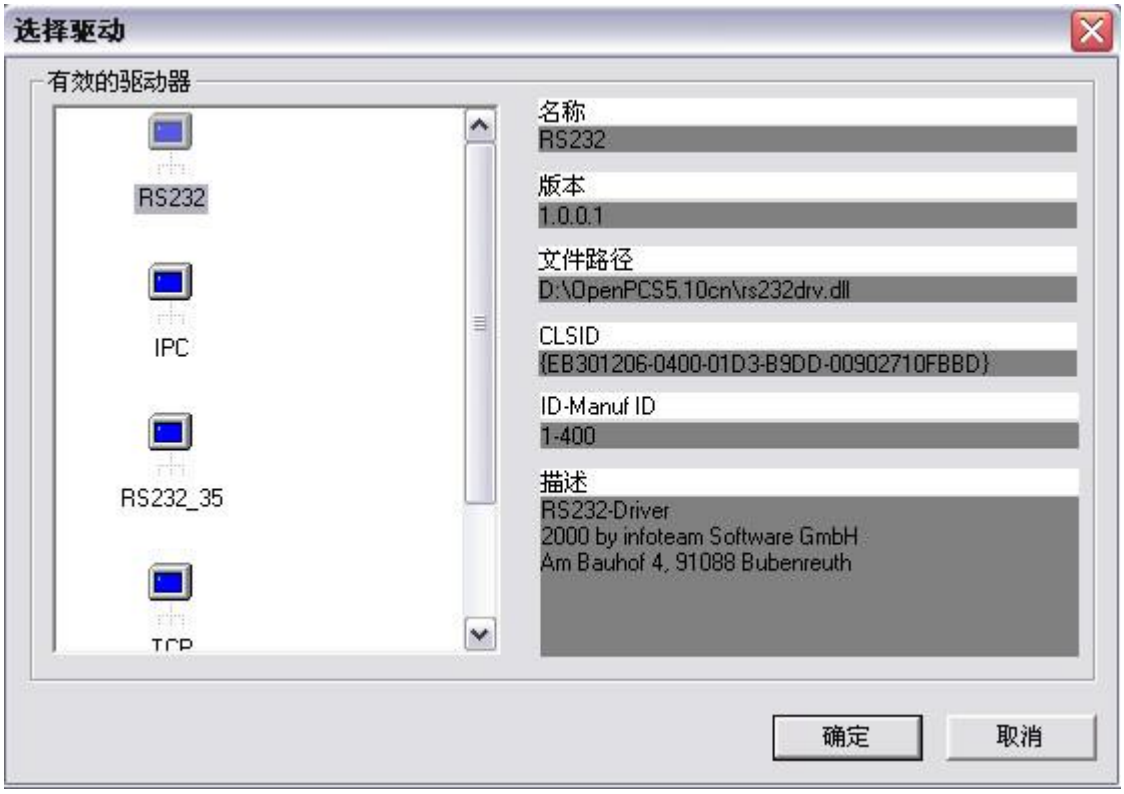
2.16.2.2 创建新连接

选择 ***PLC***->*连接* □

连接设置窗口显示时，点击**新建**按钮。

连接属性对话框打开：

现在输入新建的连接名。注意连接名不能有空格，用下划线(_)代替空格。点击选择按钮选择一个驱动，打开显示有驱动设置的驱动选择窗口：



点击需要的驱动后单击 **OK** 按钮。连接属性窗口会被再一次激活。点击**设置**按钮，打开驱动设置对话框：

您能够单个地修改设置。当您设置好后点击 **OK**，否则点击**取消**。如果您要放置一个注释时，您可以在连接属性窗口做这个。最后，点击连接属性窗口中的 **OK** 按钮。

2.16.2.3 删除一个连接

在有效的连接列表中点击连接名选择一个连接，然后点击 **删除** 按钮，这个选中的连接就被删除了，并且从列表中消失了。

2.16.2.4 编辑连接属性

在有效的连接列表中点击连接名，它就被选中了，然后单击**属性**按钮，连接属性对话框会出现。

继续按照创建新连接部分介绍继续进行。

2.16.2.5 选择连接

在浏览器中右键单击激活的资源（用绿色标记的）并在关联菜单中选择 属性... ，这个 编辑资源说明 对话框被打开了：

要选择当前的连接，点击编辑资源说明窗口中 **网络连接** 右边的按钮.

2.17 硬件驱动

2.17.1 硬件驱动：概述

利用 AddDriver 工具（AddDrvr.exe），您可以自动安装驱动包到 OpenPCS 中，这个由您的原始设备制造商提供。这些驱动包都是定义好内容的 MS-Cabinet 文件，这些都由您的硬件制造商连同 OpenPCS 一起提供给您，或者通过互联网提供。

如果没有命令行参数时执行 AddDrvr.exe，(比如，从菜单中***其它*** ->***工具*** -> ***安装驱动...***)，那么可以在对话框中选择想要的驱动包。

2.18 编译器

2.18.1 编译器：概况

编译器是一个工具，它转换在不同编辑器中编写的可执行的代码的应用程序资源，这个可执行代码是通用代码或本地代码。

无论什么时候，需要重新编译您的应用程序时，通过浏览器编译器会自动启动，并且编译器可以手动从浏览器中调用。一般的，没有必要从命令行中调用编译器。从命令行中运行编译器，需要从 infoteam 公司得到单独的许可。

2.18.2 指令列表编译器

2.18.2.1 编译器命令行

ILC [-s | -c | -p | -i | -l] <Poe file names> -v <Varfilename> -d <Devicename> -r <ResourceName> { -o <OutputDir> } {-g | -m } {-x} { -b } {-y} { -w<OutputLevel> }

ILC command <Poe file names> -v <Varfilename> -d <Devicename> -r <ResourceName> options

命令和选项是在斜线(/)或破折号(-)后面，并且不区分大小写。每次调用多个命令是允许的。

命令：

-s：对命令后面的文件执行 IL 语法检查。

-c：编译命令后面的所有文件。

-p：对命令后面的文件列出的所有 POU 创建原型。

-I：对命令后面的文件列出的所有 POU 创建包含文件。

-l：对命令后面的文件列出的所有 POU 创建一个其次的文件。

如果在一个调用 ILC.EXE 中使用了多个命令，则每一个单个命令应用于命令后面的所有文件。

选项：

-o：如果单个资源被创建，用这个选项，用户可以指定一个输出地址。目标路径名如果-o 后面跟一个目录名，则目标就存储在这个指定的目录中。

-g： 输入文件(*.poe 文件)没有被定位资源全局变量的。

-m：输入文件(在命令后面指定的文件) 能够定位资源全局变量。

-x： 跳转目标文件。

-f： 禁止最终的错误信息。

-b：用短文件名。ILC 将超过八个字符的长 POUs 文件名截短为八个字符的文件名。

-w：用这个标记，用户能指定输出信息的输出标准。这个标记后面必须跟一个表示输出标准的整数。下面的值定义了 输出标准 ：

0：打印所有有用信息，比如错误，告警和消息信息。

2：禁止告警信息。

4：禁止消息信息 (例如： 编译 c:\test\test.poe)。

6：禁止告警和消息信息。

请注意错误信息总是被打印出来，并且不能禁止。

-y：写一个能够在文本文件中编辑的初始数据段。这个命令是用来获得<hardware>.ini 文件的固件 POUs 的初始数据段。

2.18.3 连接器

2.18.3.1 链接器命令行

ITLink [-r | -t] <source file names> -v <Varfilename> -m <makefile name> {-g <resource global object filename>} {-s <packed sources file> } { -o <OutputDir> } {-x}

命令和选项是在斜线(/)或破折号(-)后面，并且不区分大小写。每次调用多个命令是不允许的。

命令：

-r：把命令后面的文件列表中指定的文件链接到一个资源目标 (*.pcd 文件)。

-t：把命令后面的文件列表中指定的文件链接到一个任务目标 (*.crd 文件)。

在这两种情况下，链接器需要工程文件路径和生成文件路径。工程文件路径必须由 -v 开始，且生成文件路径也必须由 -m 开始。

选项：

-o：如果创建单个资源，则需指定一个输出目录或目标路径名。如果-o 后面跟一个目录名，则目标就存储在这个指定的目录中。

-g：输入文件(这个文件在命令后面指定) 包含了关于资源全局变量的对象信息。

-s：选项后面指定的文件包含了资源的打包源码。这个文件的内容会插入到资源全局段表中。这个选项必须和命令 -r 联合使用时才有效。

-x： 跳转目标文件。

-w：用这个标记，用户能指定输出信息的输出标准。这个标记后面必须跟一个表示输出标准的整数。下面的值定义了 输出标准 ：

0：打印所有变量信息，即错误，告警和消息信息。

2：禁止告警信息。

4：禁止消息信息 (例如： 编译 c:\test\test.poe)。

6：禁止告警和消息信息。

请注意错误信息总是被打印出来，并且不能禁止。

2.18.4 生成器

2.18.4.1 生成命令行

ITMake [-p | (-m <makefilename> | -u <ArchiveFileName> -v) | -y] <Varfilename> { -o <OutputDir> | <Outputfilename> } {-i } { -s } {-w<OutputLevel>} {-b}

命令和选项是在斜线(/)或破折号(-)后面，并且不区分大小写。每次调用多个命令是允许的。

命令：

-p：在指定工程中创建所有资源。这个命令必须后面至少跟一个工程文件路径 (VAR-file-path)。

-m：创建指定的资源。这个命令后面必须跟一个资源文件列表，并且指定符 -v 后面要跟工程文件路径。要创建的资源必须是变量文件名中指定的工程的一部分。

-u：解压缩文件。这个命令后面至少要跟一个要解压缩的文件名。

-y：在工程中创建所有 POUs 的初始数据段，并把它写入一个文本文件。这个命令是用来获得 <hardware>.ini 文件的固件 POUs 的初始数据段。

选项：

-o：用这个选项，用户可以指定一个输出地址，或目标路径名，如果单个资源被创建。如果-o 后面跟一个目录名，则目标就存储在这个指定的目录中。

-e：这个选项是保留给将来执行。它能够使用户指定输出消息的目标路径。如果指定了一个错误文件，则用户的输出就会打印到这个文件中，并且代替标准输出。如果-e 跟了一个目录名，则为每个编译或链接实体创建的错误文件就存储在这个目录中。

-I：增加的创建，即只把修改的文件和被这个修改影响到的文件重新编译和链接。

-s：用短文件名。ITMake 将把超过八个字符的长 POUs 文件名截短为八个字符的文件名。

-w：用这个标记，用户能指定输出信息的输出标准。这个标记后面必须跟一个表示输出标准的整数。下面的值定义了 输出标准 ：

0：打印所有变量信息，即错误，告警和消息信息。

2：禁止告警信息。

4：禁止消息信息 (例如： 编译 c:\test\test.poe)。

6：禁止告警和消息信息。

请注意错误信息总是被打印出来，并且不能禁止。

-b：用短文件名。如果指定了这个选项，若一个 POU 的名字超过八个字符，则编译器用这个 POU 名字的前八个字符作为它的文件名。

2.19 许可证编辑器

2.19.1 许可证编辑器：概况

许可编辑器在 OpenPCS 启动时自动调用，也可稍后在 OpenPCS 的浏览器的帮助菜单中选择 许可 启动它：

在上面输入您公司的信息，并在下面的文本框中最多输入九个序列号和九个许可代码。

注意:

1. 若您拥有一个有效许可证，使用该许可证。
2. 要了解已安装的许可信息，点击 信息 按钮。

2.19.2 无许可证时的使用

没有许可证，OpenPCS 仍然可以完全使用，但是受限于 Simulation。

3 高级主题

3.1 运行时

3.1.1 多任务

多任务很大程度上与目标系统相关。这里介绍的是 OpenPCS 所实现的标准行为，但针对具体的目标设备可能有所不同。如果有疑问，与您的控制器供应商联系。

OpenPCS 支持 3 种由 IEC61131-3 定义的任务类型：

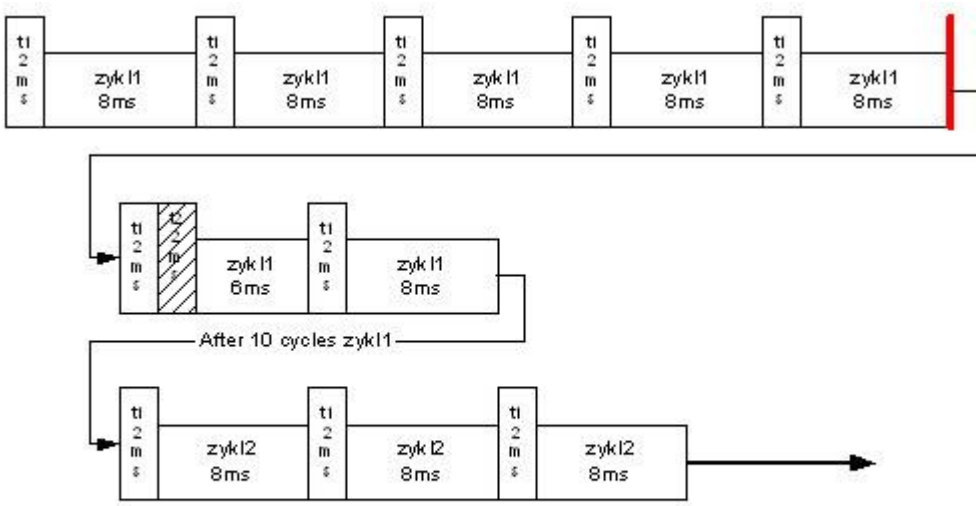
当没有记时器或中断任务即将运行的时候，将执行循环任务。任务属性栏指定的优先级将被解释为循环交替。比如优先权=3 时，每 3 个循环执行一次任务。在多循环任务中，OpenPCS 没有定义特殊的执行次序。

在任务属性中指定一个 N，定时器任务将每隔 N 毫秒被执行。

当与中断任务相连的中断发生时，中断任务被执行。由于在仿真中缺少中断资源，SmartSIM 仿真器不支持中断任务。 (请参考： 如何激活)。

作为一个例子，在下面的表中一个资源有四个任务。

任务	时间	优先权
zyk11	需 40ms	1
zyk12	需 20ms	10
timer1	每 10ms	1
	需 2ms	
timer2	每 50ms	2
	需 2ms	



3.1.2 中断

中断任务立刻被处理一旦终端发生。OpenPCS 里共有三种预定义的中断能够被 infoteam 的 SmartPLC 和 SmartPLC/OPC 支持。详情请参考 OEM 的文档。

中断类型：

- STARTUP：PLC 启动时引起。（冷启动/暖启动/热启动）
- STOP：PLC 停止时引起。
- ERROR： 错误发生时引起。

这些任务执行一次。以获得特定的状态信息或对特定状态进行操作，您须使用固件功能块例如 Resume 和 ETRC。

3.1.3 优化设置

OpenPCS 支持 速度 、 大小 和 正常 三种模式的优化设置。 大小 模式将只产生通用代码，不产生本地代码。本地代码编译器将不被调用。 正常 模式将通用代码和本地代码混合起来编译，他们两者都支持优化调试。如果本地代码支持 直接调用 ，将不能使用这种模式。如果应用的任何部分不能用本地代码来编译， 速度 模式将产生一个错误的信息。如果本地代码编译器支持 直接调用 ，可以使用这种模式。

3.1.4 多资源

OpenPCS 支持多资源。要使用多资源，在您的工程中创建一个新资源。记住源代码（在浏览器的 工程文件下）能够在不同的资源上使用。为了在线运行、下载或编译，应提前将激活资源。

如果有很多而不是少量的资源（20，50，或 200）需要同时更新、下载和执行，OpenPCS 提供了 在线批量 选项。和您的 OEM 或 infoteam 公司联系，此功能将可用。

3.1.5 变量地址

在一些应用中，给出变量名，确定运行时变量的地址是必要的。一些人机界面和网络接口需要用到这些地址。

OpenPCS 能够下载一些符号信息以支持这个功能。编译选项 Mapfile 需要用到，然后使用功能块 GetVarFlatAddress 得到地址的信息。

因为一个应用可以包含多个符号变量，所以仅一个经过过滤的子集被下载到控制器。默认情况下，仅下载全局变量，但这个过滤对 OEM 的更改是开放的，比如通过变量的名字来过滤。

3.1.6 性能

有一些明显的和不很明显的因素影响了您的程序的性能。

明显的因素包括您的平台的性能、I/O 和网络、程序的大小，这些运行时以行或字节测量的代码。如果用本地代码编译器，它可提高程序的性能。

不明显的因素包括以下：

1. 应用的**任务结构**：由于任务切换的额外开销，多任务一般会降低程序的性能。从快速的、循环任务中去掉代码，将其移到不经常执行的时间任务中，或仅当中断任务需要时，执行它们，可以很明显地提高程序的执行效率。

2. **本地代码**执行一般较快，而任务切换比较慢。因此即使本地代码有效，也有理由使用通用代码(优化 大小)执行循环任务，而仅在定时器和中断任务中使用本地代码来最大限度提高性能同时减小优化时间。

3. 当所有的代码编译成目标使用指令表作为通用中间代码时，**不同编辑器**产生的代码将不同。有些程序使用某种语言是最好的选择，但对其它程序而言则可能完全不同。仔细地评价不同的编辑器和语言然后对不同的程序选择您最喜欢的。

3.1.7 调整循环任务的顺序

正如在资源窗口中可以看到的，编译器利用资源制作文档的任务顺序来将任务连接到资源中。运行时系统使用这个顺序来执行循环任务。在资源窗口中加入了两个右键菜单项。用 向上移动 和 向下移动 就可以调整任务的顺序。在任务属性中设置的优先级对循环任务有同样的意义。

3.2 本地代码编译器

3.2.1 本地代码

OpenPCS 支持通用代码和本地代码。

本地代码是可选的，对大多数平台适用。通用代码是为不同平台间移植、快速任务转换和小的存储之间而优化设计的。本地代码则是为特殊平台而提高程序的执行速度而优化设计的。对程序级别而言，编程者能够通过资源或任务级别的优化设置选择使用的代码。

OpenPCS 的一些调试特性仅对通用代码有用，所以如果您想使用两者之一，确保将优化设置为 大小 模式（包括断点和在线显示梯形图以及 POU 编辑器、在线改变）。

OpenPCS 运行时系统通常允许在每个通用代码指令后触发任务开关，也就是说即使在一个小循环中也会频繁地触发。而在本地代码这。如果没有直接调用被使用，在每一个调用处运行时系统将检查任务开关。对直接调用，在每个任务完成后，运行时系统将检查任务开关。

考虑到不同的处理器、运行时系统和本地代码则的执行、编译运行时系统的 C 编译器以及程序的结构等的加速因素，本地代码的执行速度一般快于通用代码。如果不使用直接调用，由于调用协议转换，调用本地代码的功能块比调用通用代码的功能块要慢。因此，如果您的程序使用大量的小功能块，加速因素可能会不明显。对一些复杂指令（比如 sine 或 real-divide）通用代码开销较小，所以本地代码产生的加速因素也可能很小。

执行通用代码时，运行时系统中的通用代码虚拟机将做一些检查。在本地代码中，检查将增加运行时系统的负担和速度的提高，因此检查被省略。我们强烈地推荐，在使用本地代码前先使用通用代码仔细地调试您的程序。如

果您需要一个带检查功能的本地代码编译器，请与 infoteam 公司联系。本地代码编译器的检查功能不支持数组的溢出和字符串长度检查功能。

直接调用

一些本地代码编译器执行 直接调用 特性，这将避免功能块转换之间的开销。对指令 CAL 和 RET，直接调用一个子例程，然后在汇编语言中各自返回。这将提高性能，但降低了任务转换的响应。如果您使用单任务，性能提高很明显，如果是多任务，则会出现抖动现象。

直接调用仅在速度优化设置中使用。

3.2.2 本地代码中异常的处理

除非注明，本地代码编译器创建的代码由于性能的原因将不检查异常，诸如被零除或非法的数组索引。因此我们推荐首先使用优化设置 大小 模式仔细地调试您的程序，然后转到本地代码。

3.2.3 不认识的指令

infoteam 公司生成全部的本地代码编译器以便于它们能够编译所有的 UCODE 指令，由于一些原因，有时一些本地代码编译器不能编译一些 UCODE 指令。

这种情况发生时，本地代码编译器将报告此情况并为不包含此代码的 POU 创建本地代码。双击输出窗口中的汇报信息可使您定位到那行。

如果优化设置为 速度 模式，这一汇报信息将是个错误的信息。要么屏蔽掉那条指令要么将优化设置为 默认 模式。

在优化设置为 默认 模式时，这个 POU 在 UCODE 中执行，如果可能，其余的在本地代码中执行。

3.2.4 段范围

OpenPCS 中所有的段大小被限制在 64k 字节。本地代码比 UCODE 大，这将导致一些程序仅能运 UCODE 而不能运行本地代码。一些本地代码编译器具有特殊 段范围 ，这将允许段超过 64 k 字节。

3.2.5 NCC Intel 保护模式

NCC86 将检查非法的数组索引和被零除的异常。

NCC86 支持段范围特性。

NCC86 支持直接调用特性。

3.2.6 NCC Infineon C16x(大模式)

NCC167H 支持直接调用特性。

3.2.7 Motorola 68k

NCC68k 支持段范围特性。

NCC68k 支持直接调用特性。

3.2.8 NCC Hitachi H8/300H

NCC Hitachi H8/300H 检查无效数组索引被零除的异常

3.2.9 NCC Motorola DSP563xx

这个 NCC 仅对 OpenPCS 客户定制版本有效。

3.2.10 NCC Intel 实模式

这个 NCC 仅对 OpenPCS 客户定制版本有效。

3.2.11 NCC Motorola PowerPC

这个 NCC 将检查无效数组和被零除的异常。

这个 NCC 没有 直接调用 特性.

3.2.12 NCC ARM 模式

NCC ARM 将会检查非法的数组索引和除零异常。

NCC ARM 实现了展开分段的特征。

NCC ARM 实现了直接调用特征。

3.2.13 NCC ARM THUMB 模式

这种 NCC 仅仅在 OpenPCS 定制版本中才有。

3.3 参考文件

3.3.1 交叉参考

也可见交叉参考（每个变量） 和 CFC 交叉参考。

给您的工程创建交叉参考列表，右键单击激活的资源并从相关菜单中选择 交叉参考列表 ，将显示交叉参考的预览。交叉参考可以被查看和在线操纵，或者打印。

3.3.2 交叉参考(每个变量)

也可参看文档中的交叉参照。

打开资源窗格，鼠标右击一个变量，在弹出的相关菜单中选择交叉参照。

3.3.3 打印 IEC61131 配置

为了得到您的资源和任务配置的打印文本，在上下文菜单中选择浏览器资源表，在其中选择 打印配置 。

3.3.4 CFC 交叉参考

CFC 交叉参考在调试和对 CFC 图的理解时很有帮助的。

OpenPCS 标准交叉参考的使用受限于 CFC 编程者，交叉参考中列出的符号名字是由 CFC 编辑器自动产生的而对编程者没有意义。

为创建 CFC 交叉参考，选择 文件->交叉参考 ，或打印图表也可以看到交叉参考。文件中存储的交叉参考是不易读的，但更适合于用第三方工具（象：grep,awk）的自动邮寄处理。

CFC 交叉参考列在表单中

源：名字[图]

目标 1：名字[图]

目标 2：名字[图]

此处

右边空白栏是源的名字，也就是说设计一个信号离开一个复合块。

左边空白栏是目标的名字，也就是说设计一个信号进入一个复合块。

名字是由 CFC 编辑器为此信号自动产生的变量名。在测试和调试工具中查找此信号以监视信号的值。

图表是复合块名字的路径。在 CFC 编辑器中通过打开一个指定顺序的子复合块，或通过打印图表内容定位，都可以找到这个路径。入口由资源/目的分类。如果您需要其它分类顺序，入口指向存储过的文件。

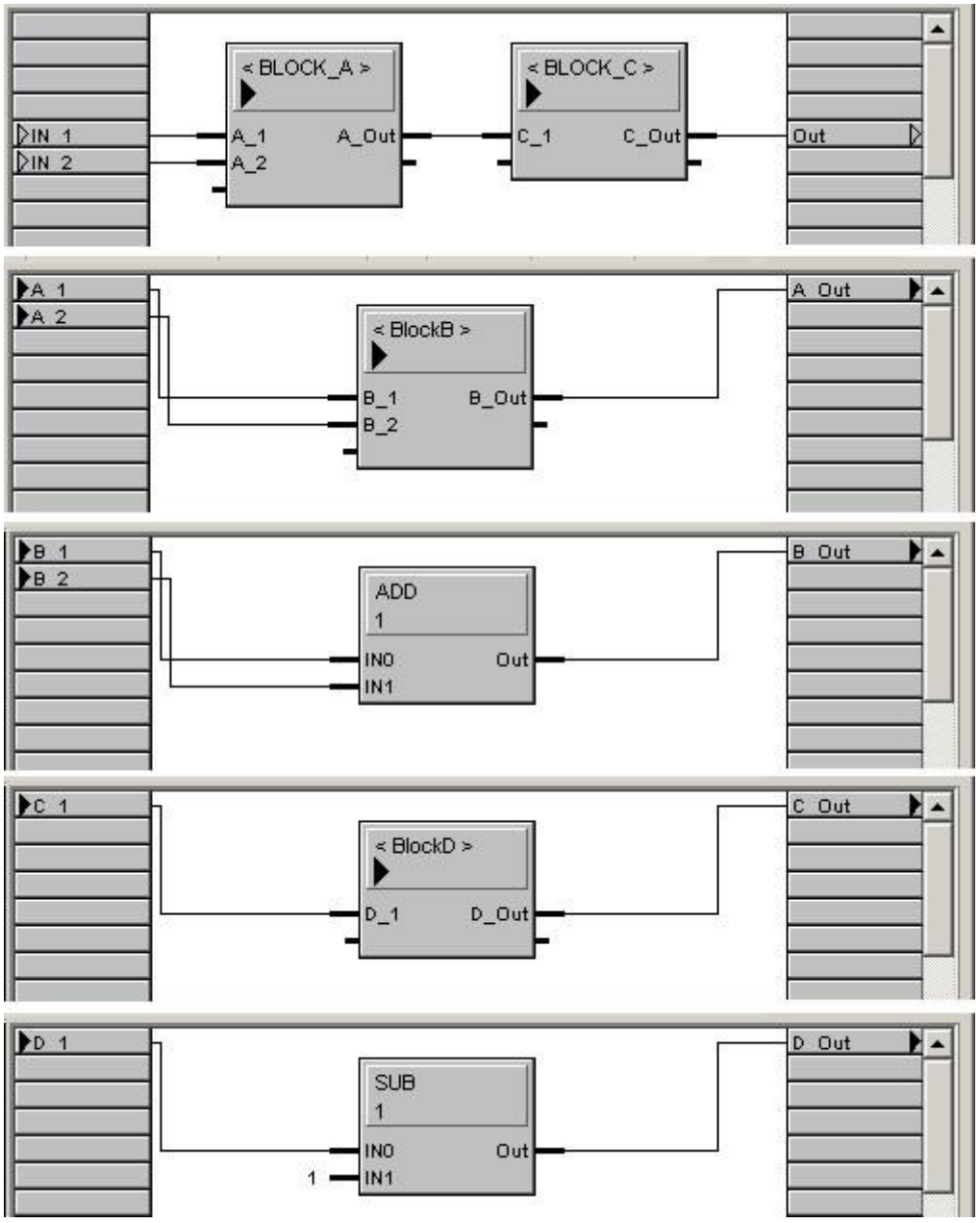
入口由源/目标来分类，如果需要其它分类顺序，请参考文件的存储。

CFC 交叉参考例子

我们用一个小例子来说明 CFC 交叉参考。

建立一个小的 CFC 程序，用两个块(ADD 和 SUB)，向一个输入变量添加 23，然后从结果中减去一：

现在将 ADD 块移动到复合块 A 且将 SUB 块移到复合块 C。打开块 A 并移动 ADD 到一个新的块 B。打开块 C 并移动 SUB 块到一个新的复合块 D。为所有空白栏入口输入合理的名称。如果您打开所有块，结果会像下面：



对这个小例子，CFC 交叉参数的输出将看起来像：

nameA_3: FCT_10_10_10_1_ADD_Out [SAMPLE.chart 1.BlockA.BlockB]

nameD_1: FCT_10_30_10_1_SUB.IN0 [SAMPLE.chart 1.BlockC.BlockD]

nameB_3: FCT_10_10_10_1_ADD_Out [SAMPLE.chart 1.BlockA.BlockB]

nameD_1: FCT_10_30_10_1_SUB.IN0 [SAMPLE.chart 1.BlockC.BlockD]

对此，下面问题很容易回答：

看 ADD 块：此输出信号去哪里？查找名为 name-3 的输出信号的名字。查看交叉参考以发现它在 chart1.Block C.Block D 块中连接到 nameD-1。

看 SUB 块：输入信号来自哪里？查找输入信号 name D_1 的名字，在交叉参数中定位 nameD_1，并发现它来自 nameA_3。因为清单以源名分类，通过用某种编辑器打开文件来查找比通过查看打印的交叉参数来查找要容易。

如何在线监测进入 SUB 块中的信号呢？在空白栏（nameD_1）中找到 SUB 块输入的名字，在交叉参数中定位并读取与变量(FCT_10_30_10_1_SUB.INO)有关的 IEC1131 变量的名字。在浏览器事例树中发现此变量并双击它以将其添加到观察清单。

3.3.5 打印选项

所有 OpenPCS 工具均支持表单的打印，将自动地使用当前激活的表单。要改变激活的打印表单，选择***其它->工具->打印选项***。这时可选择一个要打印的表单。

附加选项：

1. **打印封面**：封面由当前打印表单和一个位于工程目录的 rich text 格式文件"cover.rtf□组成。如果无法找到"cover.rtf□，只有打印表单会被打印。

2. **打印注释**：CFC 的注释将会被打印在附加的页上（相应的表后）（只适用于 CFC）。

3. **打印注释和旗标**：CFC 注释和逐条列记的存在的连接旗标，包括名称，全局标识，页码和网格位置将被打印在附加页上（相应的表后）（只适用于 CFC）。

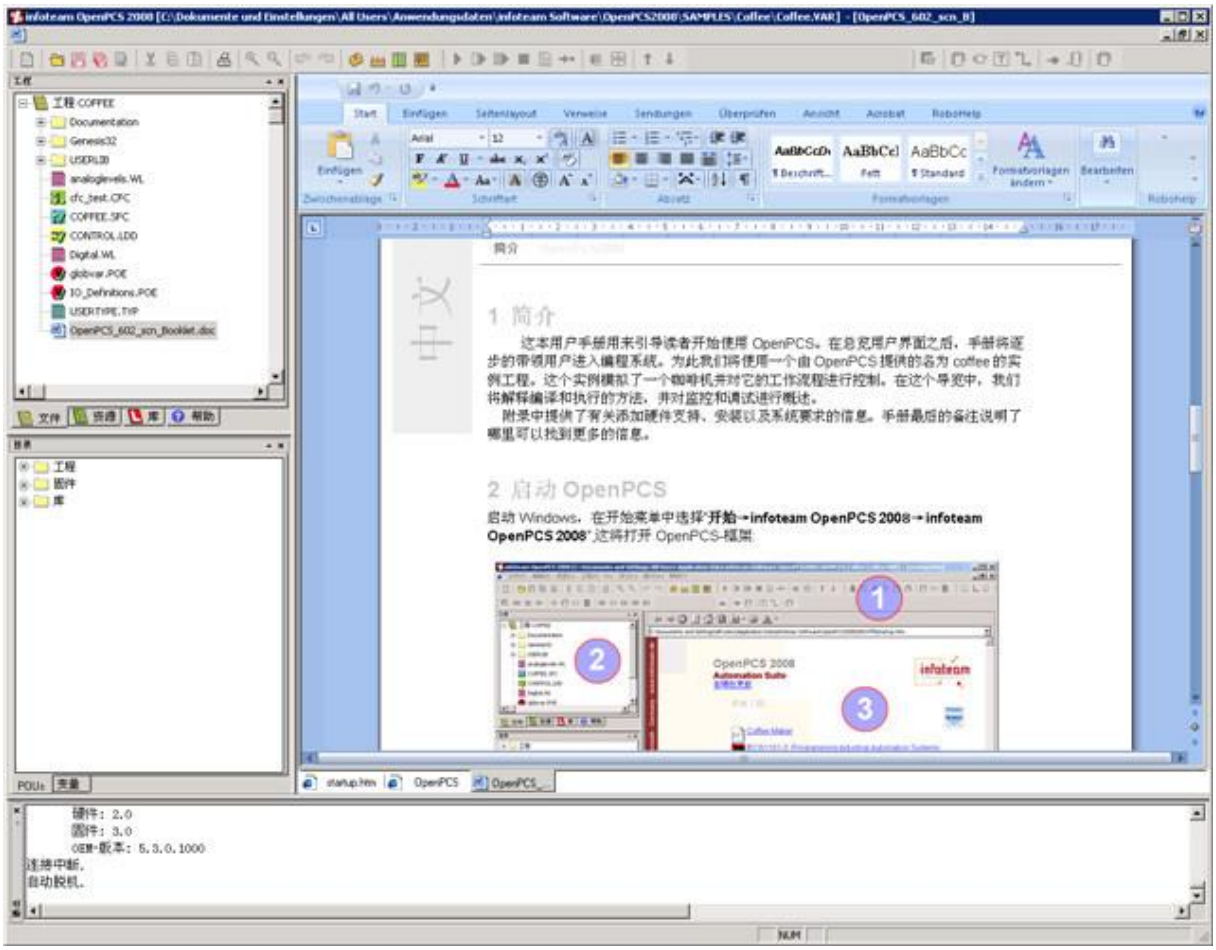
4. **打印前一页的注释**：额外的页打印当前表前页的注释（只适用于 CFC）。

注意：打印表单的纵横比须与打印机选项对应（特别是当打印表单含有图片时）。而尺寸是独立的。

3.3.6 活动文档服务器

OpenPCS 提供了一个活动文档服务器界面，也就是说，OpenPCS 支持所有注册了的活动文档，可以打开并编辑它们。

当打开这样一个文件时，文档在 OpenPCS 的编辑窗口中被打开，入下图所示：



注意：能够被 OpenPCS 编辑的文件类型可能随系统设置和安装的应用程序不同而有所变化。

警告：活动文档服务器不是 OpenPCS 的一部分，如果活动文档服务器不稳定，那么可能会引起 OpenPCS 的性能不稳定。

3.4 库

3.4.1 库：总览

库是函数和功能块的结合，它们对不同的 OpenPCS 工程可以重复使用。

库的工作涉及到以下几个步骤：首先是创建库，很可能实在其它过程中。如果使用者和创建者不是同一人可以通过软盘、CD-ROM 或 Internet 分发给他们并使库可用。用户安装这些库，也就是将库传给自己的 PC。在 OpenPCS 工程中使用库，这个库必须增加到当前工程中，这将使库的内容可用。

库能从工程中去除。如果需要一个相同的库可但具有不同的功能，这种方法是必须的。

可以从 PC 中卸载库。在不同的 PC 上使用库，这是必须的，因为许可证条件需要库首先被去除。下面的章节将给出如何创建您自己的库。

3.4.2 创建库

创建库和创建普通 OpenPCS 工程的过程一样。当您完成库工程中的 POUs(函数或功能块)时，执行语法检查。

如果从供应商那里得到库，您不必创建库，仅安装就行了。

举例：

使用*文件* -> *工程* -> ***新建...***创建一个新工程 MyLib。

使用*文件* -> *新建* -> *功能块* -> ***IL*** 创建一个功能块 det_edge（用于边沿检测）。功能块的完成如下：

```
VAR_INPUT
 input  :  BOOL ;

END_VAR

VAR_OUTPUT

 output :   BOOL ;

END_VAR

VAR

 tempvar :  BOOL ;

END_VAR

LD  input

ANDN  tempvar

ST  output

LD  input

ST  tempvar
```

使用*文件->语法检查*调用语法检查。

3.4.3 安装一个库

在使用库之前需要将它安装在您的 PC 上。利用*文件 ->库 ->安装新的**□。

使用 浏览 按钮，查找描述工程的.VAR 文件。如果库是您创建的，它将保存在创建库工程时所指定的目录下。如果从磁盘上得到库，它将在以 A:\ 开头的目录下。在安装过程中，库工程将拷贝到<windows>\openpcs.500\Lib 的一个子目录中。

举例：

在浏览器中通过*文件 ->工程 ->新建...*创建一个新工程。命名新工程为 TEST

选择*文件->库->安装新的...*

点击浏览按钮查找先前创建的 MyLib一工程，然后点击 OK

3.4.4 给工程增加一个库

安装结束后，所有库需要的文件将出现在您的电脑上。但是库中的函数和功能块在工程中不能自动可用。您必须通过*文件 -> 库 -> 在当前工程中使用*.添加库到工程中。

举例：

在库面板中标记出库 MyLib 然后选择*文件->库->在当前工程中使用*创建一个新的程序型的 POU，名字是 main。选择*插入->功能块...*看您的库函数。为了使用功能块 DET_EDGE，程序如下：

```
VAR
 sig1 AT %I0.0 : BOOL ;
 anEdge : DET_EDGE;
 count : UINT ;
END_VAR

CAL  anEdge (
```

```
 input := sig1
 |
 :=output
)

LDN anEdge.output
JMPC ende

LD  count
ADD 1
ST   count

ende:
```

编译这段程序，将它添加到您所选择的资源中，并且执行它。改变输入值%i0.0,观察变量数目的增量。

3.4.5 卸载库

如果您想删除安装在 PC 上的库，先确认该库不会再使用，选择***文件*** ->***库*** ->***卸载***。在弹出的对话框中选择想要卸载的库，单击 OK 完成。

举例：

在库面板中给库 MyLib 做个标记。

选择***文件***->***库***->***卸载***。在弹出的对话框中选择<Windows>\ openpcs.500\MyLib。

单击 OK ， MyLib 将不再是可用的库。

3.5 CANopen

3.5.1 CANopen：介绍

本手册根据 IEC61131-3 的标准描述了 PLC 程序中 CANopen 设备的集成。这个 CANopen 的集成允许使用网络变量通过预定义的功能块直接访问 CANopen 参数和函数，这需要 PLC 和 CANopen 的接口。

根据 IEC61131-3 的标准 CANopen 对 PLC 程序的服务在 CiA（CAN in Automation e.V.）草图标准 405 中。这些标准是 CANopen 函数的基础。

使用网络变量

网络变量是 CANopen 网络系统中最简单的数据交换方式。在 PLC 中，程序对网络变量的访问方式和对 PLC 内部以及本地变量的访问方式相同。从 PLC 编程的角度来说，一个变量被分配为 PLC 设备的本地输入或网络扩展设备的输入并不重要。网络变量的使用仅需要 CANopen 的基础知识。总之，对单个的 CANopen 设备，一个 CANopen 配置工具和 EDS 文件可用性在将网络变量整合进 PLC 时是必需的。

使用 CANopen 功能块

CANopen 功能块能够直接访问 CANopen 设备，这样提高了对目标系统应用的灵活性。而且，使用这些功能块不需要额外的 CANopen 配置工具和 EDS 文件。然而，使用 CANopen 功能块假定使用者对 CANopen 及其服务具有详细的知识。

3.5.2 CANopen 网络变量

许多控制设备能够交换数据，通过 CANopen 网络特性或者将附加的 CANopen I/O 模块连接到 CAN 总线来扩展系统。数据交换总是通过 PLC 程序级的网络变量进行。根据 IEC61131-3 的标准，这些变量被声明为 VAR_EXTERNAL 并标记为 外部控制单元 。PLC 本身保持这些变量的拷贝，而网络层则负责实际值的分配并使实际值和网络化的 CANopen 设备的值相同步。

从 CANopen 的角度来看，PLC 提供了 标准 I/O 模块，这些输入和输出对标准连接器的使用者不可用，但以网络变量的形式映射到过程映像。具有网络化输入和输出的 PLC 的出现可以 PLC 程序所使用的网络变量改变其数量和大小。这意味着运行不同的程序，相同的 PLC 对 CANopen 网络是不同的。为了支持这种灵活性，PLC 采用了动态目标字典，一种像数据库中使用的管理变量的通信和映射参数的结构。标准 I/O 模块通常具有一个静态目标字典。

PLC 网络变量按照 CiA 草稿标准 405 存储在目标字典的 A000h ☐ AFFFh 位置。

下面的术语对分散 I/O 扩展模块的解释很重要。

CANopen I/O 模块：

CANopen I/O 模块代表了提供这些资源的设备，诸如访问网络的输入和输出通道。这个模块从网络管理角度可以看成为一个从站。

映射：

映射描述了输入和输出变量对相应字节和位的连接解释。

CANopen 配置器：

CANopen 配置器或配置工具是一个特殊的软件工具，它用来设计和管理 CANopen 网络以及输入和输出变量的网络参数的配置。而且，CANopen 配置器用来连接 PLC 程序的网络变量和相应的 CANopen I/O 模块的输入和输出。CANopen 配置器是一个独立的软件工具且不包括在 *OpenPCS* IEC61131 编程系统的发货内容中。我们推荐 Vector Informatik 公司的程序 ProCANopen 。

EDS 文件：

EDS 文件（电子数据表）由 CANopen 设备的制造商提供。这个文件描述了 I/O，工厂默认的映射和网络通信设置以及用户可以更改的参数。

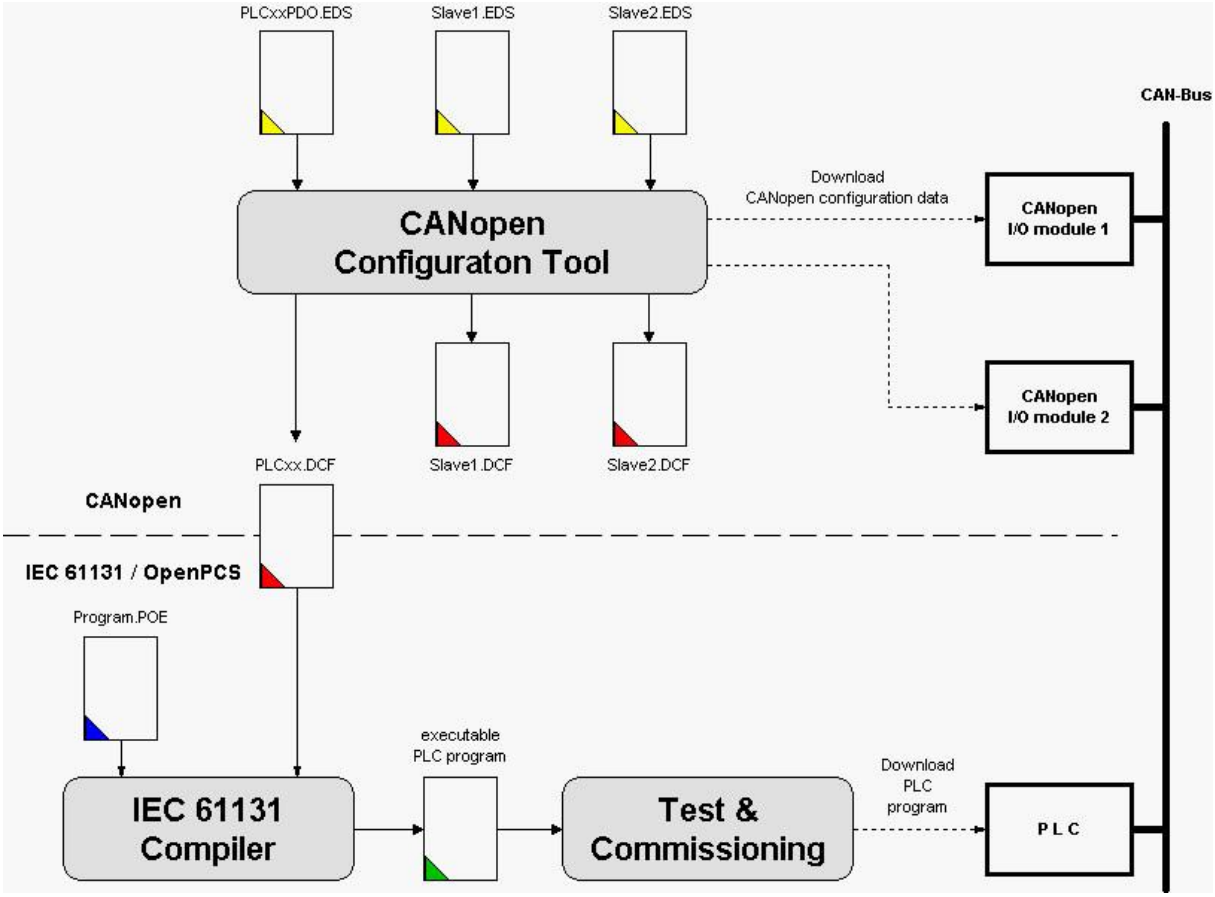
DCF 文件：

DCF 文件（设备配置文件）由 CANopen 配置器在配置的过程创建。CANopen 配置器使用 EDS 文件作为一个模板并通过用户加入识别符和映射的参数增加了有效的入口。

为了将分散 I/O 分配给 PLC，PLC 程序中激活变量被连接到 CANopen I/O 模块的输入和输出。一般来说，这需要 CANopen 配置器。与标准 CANopen I/O 设备相比，PLC 没有描述了输入和输出的数量和类型的有效 EDS 文件。PLC 上，输入和输出由用户定义，通过特殊的应用程序，输入和输出的数量和类型通过 CANopen 网络是可以访问的。由于这种原因，仅有通用的 EDS 文件对 PLC 有效，这个 PLC 定义了支持动态目标的控制单元。

3.5.3 配置过程

在 I/O 配置过程中，EDS 文件是最重要的一项。CANopen 配置器读取 EDS 文件，然后允许从用户应用中访问 CANopen I/O 模块提供的资源。在 I/O 配置过程中，用户定义了输入和输出以及相应 I/O 的字和位的信息。在配置过程中，CANopen 配置器为每一个节点创建一个 DCF 文件。如果用户需要改变供应商提供的默认参数，这种人工配置是需要的。使用一个定义了的并可调节节点数的算法来计算和分配标准参数。关于这个过程的更多信息在可应用系统或 CANopen I/O 模块的硬件手册中找到。



与静态输入输出 I/O 相比，PLC 显示了动态目标的的特性，也就是相应的 PLC 程序中定义的网络变量。由于制造商不知道运行时创建的目标，所以 EDS 文件中没有相应的目标路径。因此当连接动态目标时，需要一个配置器。配置过程的结果存储在 DCF 文件中。IEC61131 程序系统利用配置器创建的 DCF 文件来分配声明，为 VAR_EXTERNAL 网络变量以及为网络层创建必要的控制信息。

根据用户的需要，利用 CANopen 配置器的帮助来定义改变过程数据的网络参数。比如，这些参数是传送模式（同步的，异步的），分配的标识符，或映射。配置器还允许用户创建连接系统，这个系统对 PLC 和 CANopen I/O 模块是基本的。对于单个 I/O 模块将符号名字分配给过程数据(通常是输入和输出)。这将使得 PLC 程序在以后容易地访问网络变量。

对 PLC 函数，DCF 文件的功能就是作为 CANopen 配置器和 OpenPCS 编程环境的接口。这个配置文件需要连接到相应的硬件。这为控制单元提供了和 CANopen I/O 模块交换数据的必须的网络信息。

3.5.4 在 OpenPCS 中插入一个 DCF-文件

如果您的硬件支持 CANOpen，您可以通过编辑资源对话框在您的 OpenPCS 工程中插入一个 DCF-文件；

如果这个输入区域在您的对话框中不存在，请联系您的硬件设备制造商。

3.5.5 CANopen 网络变量的声明

PLC 程序中的网络变量用关键字 **VAR_EXTERNAL ... END_VAR** 声明。这样，它们被标记为 *outside of the progam* 和 *outside of the PLC* 。因此，网络变量的声明和本地变量的声明相同。

```
VAR_EXTERNAL

NetVar1 : BYTE ;
NetVar2 : UINT ;

END_VAR
```

与自由变量编辑器中的 VAR 部分相似，VAR_EXTERNAL 部分必须用人工加入。当使用语法控制的变量辑器时，网络变量必须在外部定义。自由变量和语法控制的变量之间的开关使用 POU 编辑器里的菜单〈其它-> 变量编辑器〉来完成。

DCF 文件中为过程数据定义的符号变量必须以**网络变量名字**来使用。变量名字是联系 IEC61131 PLC 和 CANopen 之间的桥梁。

与 IEC61131 和 CANopen 都兼容的数据类型必须被选为**网络变量类型**。

根据 IEC61131 的标准，在 PLC 程序中声明变量时必须与网络变量（逻辑，算术）相匹配。IEC61131 和 CANopen 之间没有明确的安排但是是有效的。

注解：

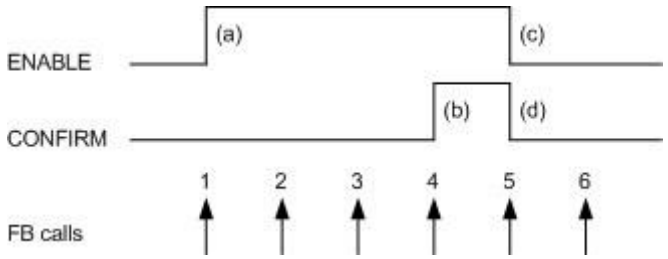
IEC61131 和 CANopen 之间没有明显的可用方案。由于这种原因，PLC 程序中使用的 IEC61131 数据类型必须根据相应变量的用法来选择。

IEC61131 和 CANopen 之间的数据类型分配：

IEC61131	*CANopen*	*Usage*	*Data Size (Bit)*
BOOL	Boolean	逻辑	1
BYTE	Unsigned 8	逻辑	8
USINT	Unsigned 8	算术 (无符号)	8
SINT	Integer8	算术 (有符号)	8
WORD	Unsigned 16	逻辑	16
UINT	Unsigned 16	算术 (无符号)	16
INT	Integer16	算术 (有符号)	16
DWORD	Unsigned 32	逻辑	32
3.5.5.1.1 UDINT	Unsigned 32	算术 (无符号)	32
DINT	Integer32	算术 (有符号)	32
REAL	Float	算术	32

同步

大部分符合 IEC61131-3 的 CANopen 功能块与 PLC 程序异步地完成。CANopen 和 PLC 之间的同步由元素信号 ENABLE 和 CONFIRM 执行。



在 CANopen 和 PLC 程序之间处理同步

一个 CANopen 和 PLC 程序之间的异步执行按下面步骤来完成：

PLC 初始化完所有的输入变量后，将 ENABLE 设置为 TRUE 同时给出 CANopen 元件被调用的命令（call1#1）。在输入 ENABLE 处 CANopen 元件识别正跳变信号，接纳所有的输入信号然后开始相应的 CANopen 设备（步（a））。最后组件返回到 PLC 程序，在此，初始化的 CANopen 服务在后台处理。

到 CANopen 服务完全处理完时，PLC 程序将多次调用功能块。为了在后台处理 CANopen 服务，在此过程中，ENABLE 必须设置为 TRUE（call2，call3）。

当成功完成 CANopen 服务时，功能块将他的输出 CONFIRM 设置为 TRUE。这个信号通知 PLC 程序 CANopen 服务已完成，同时也示意任何附加的输出变量被设为有效值（例如，从步（b）,call4）。

PLC 程序提供将 ENABLE 设置为 FALSE 对功能块提供 CANopen 服务以结束的证明。同时 PLC 程序示意 CANopen 以接收到功能块传递的输出变量（步（c）,call5）。最后一步，功能块将其输出 CONFIRM 设置为 FALSE，以便于系统返回（步(d)）。

注意：

作为一个规则，网络层允许 CANopen 服务的执行，这个执行与 PLC 程序不同时。输入 ENABLE 设置为 TRUE（步 1），访问网络层被锁住，阻止别的功能块访问。此锁住状态一直到功能块被再次调用（步 4），再次调用是当服务结束时，通过设置 ENABLE 为 FALSE 来完成的（功能块的输出 CONFIRM 设置为 TRUE）。对别的 CANopen 功能块的中间调用将导致错误输出 TRANSFER_BUSY。

当前 CANopen 服务成功地结束时，输出 CONFIRM 由 FALSE 变为 TRUE。可能的错误显示在错误输出 ERROR 和 ERRORINFO。因此为了估计已发生的错误，需要 PLC 程序监视错误值和输出 CONFIRM 信号。

当 ENABLE 设置为 FALSE 时调用功能块，将导致在后台被激活 CANopen 服务中断，并导致功能块内部复位。同时，输出 CONFIRM 被设置为 FALSE 而错误输出 ERROR 和 ERRORINFO 被设置为 NO_ERROR。

3.5.6 CANopen 常数

为了描绘网络层的内部错误状态，CiA 标准草稿 405 定义了具体的数据类型 CiA405_CANOPEN_KERNEL_ERROR 。PLC 本地网络层出现的错误状态被总结在下面。这些错误代码被不同的功能块作为输出参数 ERROR 使用。

数据类型 CiA405_CANOPEN_KERNEL_ERROR 的常数

常数 错误代码

16#0000 (= 00 dec) NO_ERROR

16#0001 (= 01 dec) OTHER_ERROR

16#0002 (= 02 dec) DATA_OVERFLOW

16#0003 (= 03 dec) TIME_OUT

16#0010 (= 16 dec) CAN_BUS_OFF

16#0011 (= 17 dec) CAN_ERROR_PASSIVE

16#0021 (= 33 dec) GENERIC_ERROR

16#0022 (= 34 dec) FUNCTION_NOT_AVAIALBE

16#0023 (= 35 dec) NO_MASTER_MODE

16#0024 (= 36 dec) INVALID_DEVICE

16#0025 (= 37 dec) TRANSFER_BUSY

16#0030 (= 48 dec) NO_SDO_CHANNEL_AVAIALBE

16#0031 (= 49 dec) SDO_BUSY

16#0032 (= 50 dec) SDO_INITIALIZE_ERROR

16#0033 (= 51 dec) SDO_LENGTH_ERROR

16#0040 (= 64 dec) NO_VALID_DATA_AVAIALBE

16#0041 (= 65 dec) COBID_ALREADY_REGISTERED

16#0042 (= 66 dec) NO_FREE_COBID_TABLE_ENTRY

16#0043 (= 67 dec) NO_SUCH_COBID_REGISTERED

16#0044 (= 68 dec) NO_FREE_RECEIVE_CHANNEL

16#0045 (= 69 dec) DATA_LENGTH_ZERO_NOT_ALLOWED

为了描绘 CANopen 设备的状态，CiA 标准草稿 405 定义了具体的数据类型 CIA405_STATE。

状态值 UNKNOWN 和 NOT_AVAIL 是现存标准的扩展。别的常数值与 CiA Draft Standard 301 的定义相适应。

CIA405_STATE 数据类型常数

常数 状态值

16#0000 INIT

16#0001 RESET_COMM

16#0002 RESET_APP

16#0003 PRE_OPERATIONAL

16#0004 STOPPED

16#0005 OPERATIONAL

16#0006 UNKNOWN

16#0007 NOT_AVAIL

为了描绘 CANopen 设备的转换状态，CiA 标准草稿定义了具体的数据类型 CIA405_TRANSITION_STATE。常数值与 CiA Draft Standard 301 的定义相匹配。

CIA405_TRANSITION_STATE 数据类型常量

常数 状态值

16#0000 START_REMOTE_NODE

16#0001 STOP_REMOTE_NODE

16#0002 ENTER_PRE_OPERATIONAL

16#0003 RESET_NODE

16#0004 RESET_COMMUNICATION

另外，CiA 标准草稿 405 定义了数据类型 CIA405_SDO_ERROR 和 EMCY_ERR_CODE 以及 EMCY_ERR_REGISTER。这些数据类型代表了别的节点产生的错误信息。

数据类型 CIA405_SDO_ERROR 用作功能块 SDO 的参数 ERRORINFO 同时传递 SDO 功能块的外部代码通信参数。普通的外部代码在 CiA 标准草稿 301 中定义但能被 CANopen 的制造商扩展来使用。

数据类型 EMCY_ERR_CODE 以及 EMCY_ERR_REGISTER 被用作功能块 CAN_RECV_EMCY 和 CAN_RECV_EMCY_DEV 相应的参数。它们包括节点的紧急错误信息。通常的紧急错误在 CiA 草稿标准 301 中定义但能被 CANopen 的制造商扩展来使用。

3.6 IEC61131-3

3.6.1 IEC61131-3 详细介绍

3.6.1.1 字符串文字

字符常数是 包围的。特殊的字符通过使用$符号嵌入在字符串中。如下表所示：

预先确定的字符串	含义
$	省略号
$$	符号$本身
$L or $l	换行
$N or $n	新行
$P or $p	换页
$R or $r	回车
$T or $t	列表

例子

字符常数	含义和长度
A	单个字符 A，长度是 1

	空格字符，长度是 1
	没有字符，长度是 1
RL	回车，换行，长度是 2
$0D$0A	回车，换页，长度是 2

3.6.1.2 最大字符串长度

每个字符串在限定最大字符串长度之内。默认的最大字符串长度为 32 个字符。该值可以单独更改，只需在关键字 STRING.后的圆括号中定义最大字长。

最大字符串的长度可以定义在 0 到 251 之间。不过在其他硬件条件下可能会不同。

例:

```
TYPE
 name: STRING(15) := John Q. Public ;    (*最大字符串长度 15*)
 address: STRING(50) := Main Street 1, 12345 Springfield, ??? ; (*最大字符串长度 50*)
END_TYPE


VAR
 user: name;     (*最大字符串长度 15*)
 id: string(8) := 12345678 ;    (*最大字符串长度 8*)
 phone : STRING;     (*最大字符串长度 32*)
END_VAR
```

3.6.1.3 常数

在字符常数中，允许使用下划线增加可读性。这种下划线对常数值没有影响。一些数据类型的字符常数需要特殊的前缀。

常数类型	例子			含义
INT	-13			整数 -13
	45165 +125	or	45_165	整数 45165 整数 125

REAL	-13.12 123.45 0.123 -1.23E-3	实数 -13.12 实数 123.45 实数 0.123 实数-0.00123
Dual number	2#0111_1110 or 126	126
Octal number	8#123 or 83	83
Hexadecimal number	16#123 or 291	291
BOOL	0 and 1 TRUE and FALSE	布尔型 TRUE 和 FALSE 值
STRING	ABC	字符串 ABC
TIME	T#12.3ms TIME#12.3ms	or 时间长度，12.3 毫秒
	T#12h34m T#12h_34m	or 时间长度，12 小时 34 分钟
	T#-4m	负的时间长度，4 分钟
DATE	DATE#1995-12-24 D#1995-12-24	or 日期，1995 年 12 月 24 日
TIME_OF_DAY	TOD#12:05:14.56 TIME_OF_DAY#12:05:14.56	or 当天时间，下午 12 点 05 分 14.56 秒
DATE_AND_TIME	DT#1995-12-24-12:05:14.56 DATE_AND_TIME#1995-12-24-12:05:14.56	or 日期和时间，1995 年 12 月 24 日下午 12 点 05 分 14.56 秒

数据类型 TIME，DATE 和 DATE_AND_TIME 的文字常量使用关键字加上符号 # 。关键字以长（举例：DATE_AND_TIME）或短（举例：DT）的形式来写。

注释：OpenPCS 当前版本不支持 DATE,TIME_OF_DAY 和 DATE_AND_TIME 三种常数类型。参看 基本数据类型。

3.6.1.4 单个位访问

OpenPCS 中，BYTE 和 WORD 变量的单个位能够通过写位号被访问，这个位号在变量名后用一点分开。

举例：

```
PROGRAM Only_1_Bit

VAR
Bitpattern1 : BYTE := 210101010;

  Bitpattern2 AT %IW0.0 : WORD;

END_VAR

LD Bitpattern2.15 (* Copy bit 15 *)

ST Bitpattern1.0 (* into bit 0 *) .

END_PROGRAM
```

请注意这种格式不一定对所有硬件平台的所有数据类型适用。

传递输出参数

IEC61131 定义了两种传递参数的方式。作为 IEC61131 的扩展，OpenPCS 提供了一种直接传递输出参数的方式。您能够在 CAL 指令内使用垂直线 | 而不使用逗号来传递输出参数，同时在赋值号的左边给出实际参数。

举例

```
CAL SR_Instance_1(SET1 := On,

RESET := Off

|
Result := Q1)
```

3.6.1.5 嵌套注释

注释不可嵌套。

3.6.1.6 块类型：程序，函数，功能块

IEC61131 定义的 OpenPCS 中的**程序**，有以下特殊的属性：

只有程序允许声明映射物理地址的变量。程序允许调用函数和其他功能块的实例。

IEC61131 定义的**功能块**，有以下特殊的属性：

它可以有一个、一个以上或者没有输入值。可以有一个、一个以上或者没有输出值。一个功能块能够创建多个实例，每个实例都会保存一个功能块的所有数据的拷贝（输入、输出、中间数据）；不能调用功能块，只有实例可以被调用。功能块有一个 存储器 ，即在多次调用之间所有数据（输入、输出、局部）不会丢失。调用功能块时，

不必提供所有输入数据；那些没有提供的数据只是被先前的调用保留了数值（或者是以前没有调用的默认值）。功能块能够调用功能和其他功能块的实例。

IEC61131 定义的**函数**，有以下特殊的属性：

有一个或者更多的输入值（但是不允许没有输入值）。有且仅有一个的输出值（可以是一个结构）。函数在调用之间，没有记忆，对于相同的输入值始终返回相同的输出值。每次调用函数时，必须提供所有的输入值。函数可以将局部变量用于中间结果，但是当函数结束时，这些中间结果的值就会丢失。函数可以调用别的函数，但是不能调用函数块实例。

3.6.2 IEC61131-3 一致性声明

3.6.2.1 一致性声明

下面的表格与 IEC61131-3/EN61131-3 标准中的有相同的编码方式。这个 OpenPCS 版本不支持的特性没有被列出来。IEC61131-3 中一些表不包括特性，因此，没有列出的表号并不意味着没有特性。为了理解这个特性，请参考 IEC61131-3。

这个 OpenPCS 版本下面的语言特点遵循 IEC61131-3 的标准：

3.6.2.2 表 1：字符设置特点

编号	描述	Yes	No
1	Required character set	x	
2	Lower case	x	
3a	Number sign (#)	x	
	Or		
3b	Pound sign (£)		x
4a	Dollar sign ($)	x	
	Or		
4b	Currency sign		x
5a	Vertical bar (|)		x
	Or		

5b	Exclamation mark (!)	x	
	Subscript delimiters:		
6a	brackets [ ]	x	
	Or		
6b	parentheses ( )		x

3.6.2.3 表 2: 标识符特点

编号	描述	Yes	No
1	Upper case and numbers	x	
2	Upper and lower case, numbers, embedded underlines	x	
3a	Upper and lower case, numbers, leading or embedded underlines	x	

3.6.2.4 表 3：注释特性

编号	描述	Yes	No
1	Comments	x	

3.6.2.5 表 4：数字文字

编号	描述	Yes	No
1	Integer literals	x	
2	Real literals	x	
3	Real literals with exponents	x	
4	Base 2 literals	x	
5	Base 8 literals	x	
6	Base 16 literals	x	
7	Boolean zero and one	x	
8	Boolean FALSE and TRUE	x	

3.6.2.6 表 5：字符串文字特点

编号	描述	Yes	No

1	Empty string (length zero)	x	
2	String of length one containing the single character A	x	
3	String of length one containing the space character	x	
4	String of length one containing the single quote character	x	
5	String of length two containing CR und LF	x	
6	String of length five which would print as $1.00	x	

3.6.2.7 表 6：在字符串中两个字符合并

编号	描述	Yes	No
2	Dollar sign ($$)	x	
3	Single quote ($)	x	
4	Line feed ($L or $l)	x	
5	New line ($N or $n)	x	
6	New page ($P or $p)	x	
7	Carriage return ($R or $r)	x	
8	Tab ($T or $t)	x	

3.6.2.8 表 7：文字特性

编号	描述	Yes	No
	Duration literals without underlines:	x	
1a	Short prefix	x	
1b	Long prefix	x	
	Duration literal with underlines	x	
2a	Short prefix	x	
2b	Long prefix	x	

3.6.2.9 表 8：天的日期和时间文字

编号	描述	Yes	No
1	Date literals (long prefix: DATE#)		x
2	Date literals (short prefix: D#)		x
3	Time of day literals (long prefix: TIME_OF_DAY#)		x
4	Time of day literals (short prefix: TOD#)		x
5	Date and time literals (long prefix: DATE_AND_TIME#)		x

6	Date and time literals (short prefix: DT#)		x

3.6.2.10 表 10:基本数据类型

序号	关键字	数据类型	Yes	No
1	BOOL	布尔型	x	
2	SINT	短整形	x	
3	INT	整形	x	
4	DINT	双整形	x	
5	LINT	长整型		x
6	USINT	无符号短整形	x	
7	UINT	无符号整形	x	
8	UDINT	无符号双整形	x	
9	ULINT	无符号长整形		x
10	REAL	实数	x	
11	LREAL	长实数		x
12	TIME	持续时间	x	
13	DATE	日期 (唯一)		x
14	TIME_OF_DAY or TOD	时间 (唯一)		x
15	DATE_AND_ TIME or TD	日期和时间		x
16	STRING	长变量字符串	x	
17	BYTE	8 位	x	
18	WORD	16 位	x	
19	DWORD	32 位	x	

20	LWORD	64 位		x

3.6.2.11 表 12：数据类型声明特点

编号	描述	Yes	No
1	Direct derivation from elementary types	x	
2	Enumerated data types		x
3	Subrange data types		x
4	Array data types	x	
5	Structured data types	x	

3.6.2.12 表 13：默认初始值

描述	初始值	Yes	No
BOOL, SINT, INT DINT, LINT,	0	x	
USINT, UINT, UDINT, ULINT	0	x	
BYTE, WORD, DWORD, LWORD	0	x	
REAL, LREAL	0.0	x	
TIME	T#0s	x	
DATE	D#0001-01-01		x
TIME_OF_DAY	TOD#00:00:00		x
DATE_AND_TIME	DT#0001-01-01-00:00:00		x
STRING	(the empty string)	x	

3.6.2.13 表 14：数据类型初始值声明特征

编号	描述	Yes	No
1	Initialization of directly derived types	x	
2	Initialization of enumerated data types		x
3	Initialization of subrange data types		x
4	Initialization of array data types		x
5	Initialization of structured data types	x	
6	Initialization of derived structured data types	x	

3.6.2.14 表 15： 直接表示变量的区域和前缀大小

编号	描述	Yes	No
1	I: Input location	x	

2	Q: Output location	x	
3	M: Marker location	x	
4	X: (Single) bit size	x	
5	None: (Single) bit size	x	
6	B: Byte (8 bits) size	x	
7	W: Word (16 bits) size	x	
8	D: Double word (32 bits) size	x	
9	L: Long word (64 bits) size		x

3.6.2.15 表 16：变量声明的变量关键字

关键字	Yes	No
VAR	x	
VAR_INPUT	x	
VAR_OUTPUT	x	
VAR_IN_OUT	x	
VAR_EXTERNAL	x	
VAR_GLOBAL	x	
VAR_ACCESS		x
RETAIN	x	
CONSTANT	x	
AT	x	

3.6.2.16 表 17：变量类型分配特性

编号	描述	Yes	No
1	Declaration of directly represented, non-retentive variables	x	
2	Declaration of directly represented, retentive variables	x	
3	Declaration of locations of symbolic variables	x	
4	Array location assignment		x
5	Automatic memory allocation of symbolic variables	x	
6	Array declaration	x	
7	Retentive array declaration	x	
8	Declaration of structured variables	x	

3.6.2.17 表 18：变量初始值分配特性

编号	描述	Yes	No

1	Initialization of directly represented, non-retentive variables		x
2	Initialization of directly represented, retentive variables		x
3	Location and initial value assignment to symbolic variables		x
4	Array location assignment and initialization		x
5	Initialization of symbolic variables	x	
6	Array initialization	x	
7	Retentive array declaration and initialization	x	
8	Initialization of structured variables	x	
9	Initialization of constants	x	

3.6.2.18 表 19：布尔信号的求反图形

编号	描述	Yes	No
1	Negated input	x	
2	Negated output		x

3.6.2.19 表 20：用 EN 输入和 ENO 输出

编号	描述	Yes	No
1	Use of EN and ENO		x
2	Use of EN and ENO		x
3	FBD without EN and ENO	x	

3.6.2.20 表 21：类型和过载函数

编号	描述	Yes	No
1	Overloaded functions (non type-dependent)	x	
2	Typed functions	x	

3.6.2.21 表 22：类型转化函数特点

编号	描述	Yes	No
1	*_TO_**	x	
2	TRUNC	x	
3	BCD_TO_**		x

4	*_TO_BCD		x

注释：

如果您正使用 TIME 值，那只有 TIME_TO_DINT, TIME_TO_REAL 和 DINT TO_TIME 被执行。其他的 TIME 类函数只在梯形图编辑器中有效。

对于第一行，（*）为输入变量数据类型，（**）为输出变量数据类型。支持以下数据类型：

BOOL

BYTE

DINT

DWORD

INT

REAL

SINT

STRING

TIME

UDINT

UINT

USINT

WORD

3.6.2.22 表 23：一个数字变量的标准函数

编号	描述	Yes	No
1	ABS	X	
2	SQRT	x	
3	LN	x	
4	LOG	x	
5	EXP	x	
6	SIN	x	
7	COS	x	
8	TAN	x	
9	ASIN	x	

10	ACOS	x	
11	ATAN	x	

3.6.2.23 表 24：数学标准函数

编号	名字	符号	Yes	No
12	ADD	+	X	
13	MUL	*	X	
14	SUB	-	X	
15	DIV	/	X	
16	MOD		X	
17	EXPT	**	X	
18n	MOVE			x
18s	TAN	:=	X	

3.6.2.24 表 25：标准位转换函数

编号	名字	Yes	No
1	SHL	X	
2	SHR	X	
3	ROR	X	
4	ROL	X	

3.6.2.25 表 26：标准按位布尔函数

编号	名字	Yes	No
1	AND	x	
2	OR	x	
3	XOR	x	
4	NOT	x	

3.6.2.26 表 27：标准选择函数

编号	名字	Yes	No

1	SEL		x
2a	MAX	x	
2b	MIN	x	
3	LIMIT	x	
4	MUX		x

3.6.2.27 表 28：标准比较函数

编号	名字	Yes	No
5	GT	x	
6	GE	x	
7	EQ	x	
8	LE	x	
9	LT	x	
10	NE	x	

3.6.2.28 表 29：标准字符串函数

编号	名字	Yes	No
1	LEN	x	
2	LEFT	x	
3	RIGHT	x	
4	MID		x
5	CONCAT	x	
6	INSERT		x
7	DELETE		x
8	REPLACE		x
9	FIND	x	

3.6.2.29 表 30：时间数据类型的函数

编号	名字	操作	Yes	No
1	ADD	TIME + TIME = TIME	x	
2		TOD + TIME = TOD		x

3		DAT + TIME = DAT		x
4	SUB	TIME - TIME = TIME	x	
5		DATE - DATE = TIME		x
6		TOD - TIME = TOD		x
7		TOD - TOD = TIME		x
8		DAT - TIME = DAT		x
9		DAT - DAT = TIME	x	
10	MUL	TIME * ANY_NUM = TIME		
11	DIV	TIME / ANY_NUM = TIME		
12	CONCAT	DATE TOD = DAT		
		Type conversion functions		
13		DATE_AND_TIME_TO_TIME_OF_DAY		
14		DATE_AND_TIME_TO_DATE		

3.6.2.30 表 31：枚举数据类型的函数

编号	名字	Yes	No
1	SEL		x
2	MUX		x
3	EQ		x
4	NE		x

3.6.2.31 表 33：功能块声明特点

编号	描述	Yes	No
1	RETAIN qualifier on internal variables	x	
2	RETAIN qualifier on output variables	x	
3	RETAIN qualifier on internal function blocks		x

4a	Input/output declaration (textual)	x	
4b	Input/output declaration (graphical)		x
5a	Function block instance name as input (textual)		x
5b	Function block instance name as input (graphical)		x
6a	Function block instance name as input/output (textual)		x
6b	Function block instance name as input/output (graphical)		x
7a	Function block instance name as external variable (textual)		x
7b	Function block instance name as external variable (graphical)		x
	Textual declaration of		
8a	- rising edge inputs	x	
8b	- falling edge inputs	x	
	Graphical declaration of		
9a	- rising edge inputs		x
9b	- falling edge inputs		x

3.6.2.32 表 34：标准双稳态的功能块

编号	名字	Yes	No
1	SR	x	
2	RS	x	
3	SEMA		x

3.6.2.33 表 35：标准边界检测功能块

编号	名字	Yes	No
1	R_TRIG	x	
2	F_TRIG	x	

3.6.2.34 表 36：标准计数功能块

编号	名字	Yes	No
1	R_TRIG	x	

2	F_TRIG	x	

3.6.2.35 表 37：标准定时器功能块

编号	名字	Yes	No
1	TP(Pulse)	x	
2a	TON (on-delay)	x	
2b	T---0 (on-delay)		x
3a	TOF (off-delay)	x	
3b	0---T (off-delay)		x
4	RTC (real-time clock)	x	

3.6.2.36 表 39：程序声明特性

编号	描述	Yes	No
1	RETAIN qualifier on internal variables	x	
2	RETAIN qualifier on output variables		x
3	RETAIN qualifier on internal function blocks		x
4a	Input/output declaration (textual)		x
4b	Input/output declaration (graphical)		x
5a	Function block instance name as input (textual)		x
5b	Function block instance name as input (graphical)		x
6a	Function block instance name as in-put/output (textual)		x
6b	Function block instance name as in-put/output (graphical)		x
7a	Function block instance name as external variable (textual)		x
7b	Function block instance name as external variable (graphical)		x

	Textual declaration of		
8a	- rising edge inputs		x
8b	- falling edge inputs		x
	Graphical declaration of		
9a	- rising edge inputs		x
9b	- falling edge inputs		x
10	Formal input and output parameters		x
11	Declaration of directly represented, non-retentive variables	x	
12	Declaration of directly represented, retentive variables	x	
13	Declaration of locations of symbolic variables	x	
14	Array location assignment		x
15	Initialization of directly represented, non-retentive variables		x
16	Initialization of directly represented, retentive variables		x
17	Location and initial value assignment to symbolic variables		x
18	Array location assignment and initialization		x
19	Use of directly represented variables	x	
20	VAR_GLOBAL .. END_VAR Declaration within a PROGRAM		
21	VAR_ACCESS .. END_VAR Declaration within a PROGRAM		

3.6.2.37 表 40：步骤特性

编号	描述	Yes	No

1	Step graphical	x	
	Initial step graphical	x	
2	Step textual		x
	Initial Step textual		x
3a	Step flag general form		x
3b	Step flag - direct connection of boolean variable		x
4	Step elapsed time		x

3.6.2.38 表 41：转换和转换条件

编号	描述	Yes	No
1	Transition condition using ST language		x
2	Transition condition using LD language		x
3	Transition condition using FBD language		x
4	Use of connector		x
4a	Transition condition using LD language		x
4b	Transition condition using FBD language		x
5	Textual transition in ST	x	
6	Textual transition in IL	x	
7	Transition name	x	
7a	Transition condition using LD language		x
7b	Transition condition using FBD language		x
7c	Transition condition using IL language	x	
7d	Transition condition using ST language	x	

3.6.2.39 表 42：动作声明

编号	描述	Yes	No
1	boolean variable as action		x
2l	graphical declaration in LD language		x
2s	inclusion of SFC elements in action		x
2f	graphical declaration in FBD language		x
3s	textual declaration in ST language		x
3i	graphical declaration in IL language	x	

3.6.2.40 表 43：步骤/动作联合

编号	描述	Yes	No
1	action block		x
2	concatenated action blocks		x
3	textual step body	x	
4	action block d field		x

3.6.2.41 表 44：动作块特点

编号	描述	Yes	No
1	qualifier as per 2.6.4.4		x
2	action name	x	
3	boolean indicator variables		x
4	IL language		x
5	ST language		x
6	LD language		x
7	FBD language		x
8	action blocks in ladder diagrams		x
9	action block in function block diagrams		x

3.6.2.42 表 45：动作限定符

编号	描述	Yes	No
1	none	x	
2	N (non-stored)	x	
3	R (overriding reset)		x
4	S (set stored)		x
5	L (time limited)		x
6	D (time delayed)		x
7	P (pulse)		x
8	SD (stored and time delayed)		x
9	DS (delayed and stored)		x
10	SL (stored and time limited)		x

3.6.2.43 表 46：顺序进展

编号	描述	Yes	No
1	single sequence	x	
2a	divergence of sequence selection (left-to-right)	x	
2b	divergence of sequence selection (with priorities)		x
2c	divergence of sequence selection (with mutual exclusion)		x
3	Convergence of sequence evolution	x	
4	simultanouse sequences divergence	x	
5	simultanouse sequences convergence	x	
5a	sequence skip (left-to-right)	x	
5b	sequence skip (with priorities)		x
5c	sequence skip (with mutual exclusion)		x
6a	sequence loop (left-to-right)		x
6b	sequence loop (with priorities)		x
6c	sequence loop (with mutual exclusion)		x
7	directional arrows		x

3.6.2.44 表 52：操作指令列表 (IL)

编号	操作命令	改造者	Yes	No
1	LD	N	x	
2	ST	N	x	
3	S		x	
	R		x	
4	AND	N,(	x	
5	&	N,(	x	
6	OR	N,(	x	
7	XOR	N,(	x	
8	ADD	(	x	
9	SUB	(	x	
10	MUL	(	x	
11	DIV	(	x	
12	GT	(	x	
13	GE	(	x	
14	EQ	(	x	

15	NE	(	x	
16	LE	(	x	
17	LT	(	x	
18	JMP	C, N	x	
19	CAL	C, N	x	
20	RET	C, N	x	
21	)		x	

3.6.2.45 表 53：IL 语言的功能块特点

编号	描述	Yes	No
1	CAL with input list	x	
2	CAL with load/store of inputs	x	
3	Use of input operators		x

3.6.2.46 表 55：ST 语言操作

编号	描述	Yes	No
1	Parenthisization	x	
2	Function evaluation	x	
3	Exponentiation	x	
4	Negation	x	
5	Complement	x	
6	Multiply	x	
7	Divide	x	
8	Modulo	x	
9	Add	x	
10	Subtract	x	
11	Comparision	x	
12	Equality	x	
13	Inequality	x	
14	Boolean AND	x	
15	Boolean AND	x	
16	Boolean Exclusive XOR	x	
17	Boolean OR	x	

3.6.2.47 表 56：ST 语言综述

编号	描述	Yes	No
1	Assignment	x	
2	Function block invocation and FB output usage	x	
3	RETURN		x
4	IF	x	
5	CASE	x	
6	FOR	x	
7	WHILE	x	
8	REPEAT	x	
9	EXIT	x	
10	Empty Statement	x	

3.6.2.48 表 57：线程和块表示

编号	描述	Yes	No
1	Horizontal lines:		x
2	ISO/IEC 646 minus character	x	
	graphic or semigraphic		
4	Vertical lines:	x	
3	ISO/IEC 646 vertical line character		x
4	graphic or semigraphic	x	
	Horizontal/vertical connection:		
5	ISO/IEC 646 plus character		x
6	graphic or semigraphic	x	
	Line crossing without connection:		
7	ISO/IEC 646 characters		x
8	graphic or semigraphic	x	
	Connected and non-connecte corners:		
9	ISO/IEC 646 characters		x
10	graphic or semigraphic	x	
	Blocks with connecting lines	x	
11	ISO/IEC 646 characters		x
12	graphic or semigraphic	x	
	Connectors using ISO/IEC 646 chararcters:		

13	Connector, Continuation of a connected line		x
14	graphic or semigraphic	x	

3.6.2.49 表 58：图形执行控制元素

编号	描述	Yes	No
	Unconditional Jump		
1	FBD language	x	
2	LD language	x	
3	Conditional Jump (FBD language)	x	
4	Conditional Jump (LD language)	x	
	Conditional Return		
5	LD language	x	
6	FBD language	x	
	Unconditional Return		
7	from Function	x	
	from Function Block	x	
8	Alternative Representation in LD language	x	

3.6.2.50 表 59：功率轨道

编号	描述	Yes	No
1	Left power rail	x	
2	Right power rail	x	

3.6.2.51 表 60：链接基础

编号	描述	Yes	No
1	Horizontal link	x	
2	vertical link with attached horizontal links	x	

3.6.2.52 表 61：触点

编号	描述	Yes	No
	Normally open contact		

1		x	
2		x	
	Normally closed contact		
3		x	
4		x	
	Positive tranisition-sensing contact		
5			x
6			x
	Negative transition-sensing contact		
7			x
8			x

3.6.2.53 表 62：线圈

编号	描述	Yes	No
1	Coil	x	
2	Negated Coil	x	
3	SET (latch) coil	x	
4	RESET (unlatch) coil	x	
5	Retentive (Memory) coil		x
6	SET retentive (Memory) coil		x
7	RESET retentive (Memory) coil		x
8	Positive transition-sensing coil		x
9	Negative transition-sensing coil		x

3.6.2.54 表 63：保留名

数据类型不能用作文件名或变量名。以下的名字也不能用于变量和/或文件：

D

L

N

P

Q

3.6.2.55 FBD 语言要素

[注意：这一章是空的，因为在 IEC1131-3 中相应的章节中没有列作出它的任何特性以供参考]。

3.6.2.56 附录 D.1：决定执行的参数

条款	参数	值
1.5.1	Error handling procedures	see next chapter
2.1.1	National characters used	see table 1 above
2.1.2	Maximum length identifiers	256
2.1.5	Maximum comment length	>512
2.2.3.1	Range of values of duration	+/- 24,85 days
2.3.1	Range of values for variables of type TIME	+/- 24,85 days
	Precision of representation of seconds in type	
	TIME_OF_DAY and DATE_AND_TIME	-
2.3.3	Maximum	
	- number of array subscripts - array size - number of structure elements - structure size - number of variables per declaration	6 < 4KB per POU < 8KB per POU
2.3.3.1	Maximum number of enumerated values	< 64 KB per POU
2.3.3.2	Default maximum length of STRING variables	32
	Maximum permissible length of STRING variables	253 [see note 1]
2.4.1.1	Maximum number of hierarchical levels Logical or physical mapping	5
2.4.1.2	Maximum number of subscripts Maximum number of subscript values Maximum number of levels of structures	- - >512
2.4.2	Initialization of system inputs	The value of the system inputs correponds to their physical values
2.4.3	Maximum number of variables per declaration	< 8 KB per POU
2.5	Information to determine execution times of program organisation units	No
2.5.1.1	Method of function representation	Textual
2.5.1.3	Maximum number of function specifications	limited only by available memory
2.5.1.5	Maximum number of inputs of extensible functions	IL: 2, LD/FBD: unlimited

2.5.1.5.1	Effects of type conversions on accuracy	truncated
2.5.1.5.2	Accuracy of functions of one variable Implementation of arithmetic functions	Currently not supported
2.5.2	Maximum number of function blocks and instantiations	ca. 8000
2.5.2.3.3	PVmin, PVmax of counters	minimum/maximum value of respective data type
2.5.3	Program size limitations	limited only by available memory
2.6	Timing and postability effects of execution control elements	-
2.6.2	Precision of step elapsed time Maximum number of steps per SFC	-
2.6.3	Maximum number of transitions per SFC and per step	-
2.6.4	Action control mechanism	-
2.6.4.2	Maximum number of action blocks per step	-
2.6.5	Graphic indication of step state Transition clearing time Maximum width of diverge/converge constructs	-
2.7.1	Content of RESOURCE libraries	-
2.7.2	Maximum number of tasks Task interval resolution Pre-emptive or non-pre-emptive scheduling	-
3.3.1	Maximum length of expressions Partial evaluation of Boolean expressions	unlimited no
3.3.2	Maximum length of statements	unlimited
3.3.2.3	Maximum number of CASE selections	unlimited
4.1.1	Graphic/semigraphic representation Restrictions on network topology	graphic
4.1.3	Evaluation order of feedback loops	-

注释 1： OpenPCS 高级的配置的，因此这个参数很可能要依赖于硬件。如果有疑问，请参考您的硬件描述文档资料。

3.6.2.57 表 E.1(标准化的)：错误情况

条款	参数	值
2.3.3.1	Value of a variable exceeds the specified subrange	Syntax error; overflow can be scanned during run time

2.4.2	Length of initialisation list doesn t match the number of array entries	Syntax error
2.5.1.5.1	Type conversion errors	Syntax error
2.5.1.5.2	Numerical result exceeds range for data type	Can be monitored
	Division by zero	Can be monitored
2.5.1.5.4	Mixed input data types to a selection function	Syntax
	Selector (K) out of range for MUX function	-
2.5.1.5.5	Invalid character position specified	-
	Result exceeds maximum string length	
2.5.1.5.6	Result exceeds range for data type	Restriction to maximum value (see 2.2.3.1)
2.6.2	Zero or more than one initial step in the SFC network	-
	User program attempts to modify step state or time	
2.6.2.5	Simultaneously true, non-prioritized transitions in a selection divergence	-
2.6.3	Side effects in evaluation of transition condition	-
2.6.4.5	Action control contention error	-
2.6.5	Unsafe or Unreachable SFC	-
2.7.1	Data type conflict in VAR_ACCESS	-
2.7.2	Tasks require too many processor resources	-
	Execution deadline not met	
	Other task scheduling conflicts	
3.2.2	Numerical result exceeds range for data type	Scan via functions
3.3.1	Division by zero	Syntax error
	Invalid data type for operation	can be monitored
3.3.2.1	Return from function without value assigned	-
3.3.2.4	Iteration fails to terminate	-
4.1.1	Same identifier as connector label and element name	-
4.1.4	Uninitialised feedback variable	-
4.1.5	Numerical result exceeds range for data type	-
	Division by 0	

3.7 在线特性

3.7.1 断点

在文本语言 ST 和 IL 中，OpenPCS 支持断点。本地代码不支持断点功能，因此设置优化为 大小 模式。由于硬件的限制，断点可能不支持所有的目标系统。断点不能被保存，因此，在开始下载的新应用以前，应设置新的断点。

如果一个断点到达 OpenPCS 应用的任何一个任务，所有任务的执行立即被终止。当单步、连续到下一个断点时，这个断点没有被定义并且如果同时执行别的任务，断点将离开目标。因此单步执行时，我们推荐单任务执行。

用断点和单步执行终止控制器时，将使您的控制器和应用的安全防范失效，因此确保采取适当的措施来避免这种损坏。

3.7.2 在线编辑

在线编辑（或者在线变更）是一种特性，它使得程序的变化能够直接被应用于 PLC，而不需要重新运行。

进行在线编辑的步骤：

- 在在线模式中，通过 ***PLC->监视/编辑***（或使用工具栏上的监视/编辑）切换一个编辑器到编辑模式
- 在此编辑器中修改声明和代码
- 通过使用监视/编辑切换回监视模式
- 现在你将会被提示更新目标。选择 是 以存储任何修改，重新编译应用并将你的修改下载到目标而无须停止程序
- 选择 否 以放弃在线编辑和所有改动（不会有改动被保存到文件中）

变更诸如添加变量，变量重命名或者改变变量的数据类型会导致变量重设为初始值。因为一次重启并不是必需的，那些没有被变更影响到的程序部分的变量值保持它们当前的值。（也就是说它们不会被重设为初始值）。如不重启，任何对初始值的改变不会导致显著的结果。

不支持的特性：增加/减少任务，改变任务属性，改变资源属性，改变任务顺序。

小心：如果目标系统不支持保存系统，变更不是永久的。系统可以通过 ***PLC -> 保存系统*** 保存，如果变更需要永久的保存在控制器上。更多信息清参照保存系统。

注意 1：严格地说，函数也是 POU。但因为它们是无状态的，所以在线编辑不用考虑函数。

注意 2：在线编辑从 OpenPCS 版本 5.3.0 开始被支持。它需要目标系统版本 5.2.2 或以上。

注意 3：此外其他配置规则定义如下：

运行系统支持无中断下载 & OnlineLinking = 1	保留数据，可在线编辑
运行系统支持无中断下载 & OnlineLinking = 0	数据设为初始值，可在线编辑
运行系统不支持无中断下载 & OnlineLinking = 0	无法在线编辑，直到重新编译及下载数据才会被应用
运行系统不支持无中断下载 & OnlineLinking = 1	无法在线编辑，直到重新编译及下载数据才会被应用

提示：在线链接可在硬件配置文件（.ini 文件）中设置。

3.7.3 保存系统

PLC->存储系统.....将整个系统永久的写入控制器。如果在在线编辑时做出更改，则这一步是必须的。

保存系统是目标系统的可选特性。

3.7.4 错误日志

一个详尽的错误日志可以从控制器通过 ***PLC->上传错误日志***来上传。上传的文件将会被命名为 yymmdd_hhmmsssErrorlog.txt 并保存在当前工程目录下。

保存错误日志是可选的目标系统特性。

4 参考列表

4.1 关键字(按类型)

4.1.1 IEC61131 标准功能块

OpenPCS 实现以下 IEC61131-3 功能块：

CTD

CTU

CTUD

F_TRIG

R_TRIG

RS

SR

TOF

TON

TP

4.1.2 IEC61131-3 标准函数

OpenPCS 实现以下 IEC61131-3 函数：

ABS

ACOS

AND

ASIN

ATAN

CONCAT

COS

DELETE

EQ

EXP

FIND

GE

GT

INSERT

LE

LEFT

LEN

LIMIT

LN

LOG

LT

MAX

MID

MIN

MOD

NE

NEG

OR

REAL_TO_*

RIGHT

ROL

ROR

SHL

SIN

SHR

SQRT

TAN

TIME_TO_*

TRUNC

XOR

4.1.3 IEC61131-3 运算

OpenPCS 实现以下 IEC61131-3 运算:

ADD

ADD (time)

DIV

DIV (time)

MUL

MUL (time)

SUB

SUB (time)

4.1.4 OpenPCS 函数和功能块

除了 IEC1131-3， OpenPCS 还提供以下函数和功能块：

GetTaskInfo

GetTime

GetTimeCS

GetVarData

GetVarFlatAddress

4.1.5 数据类型

IEC61131-3 定义了以下基本数据类型：

BOOL

BYTE

DATE_AND_TIME

DATE

DINT

DWORD

INT

REAL

SINT

STRING

TIME_OF_DAY

TIME

UDINT

UINT

WORD

除了 IEC1131-3，OpenPCS 还定义以下数据类型：

VARINFO

POINTER

4.1.6 关键字声明

END_TYPE

END_VAR

RETAIN

TYPE

VAR_GLOBAL

VAR_IN_OUT

VAR_INPUT

VAR_OUTPUT

VAR_EXTERNAL

VAR

4.1.7 指令表指令

程序逻辑指令：

) (Right-paranthesis-operator)

CAL Instance name

CALC Instance name

CALCN Instance name

JMP Label

JMPC Label

JMPCN Label

RET

RETC

RETCN

布尔操作与指令：

NOT

AND

ANDN

OR

ORN

XOR

XORN

S BOOL

R BOOL

数学操作：

ADD

SUB

MUL

DIV

运行/保存指令：

LD ANY

LDN ANY_BIT

ST ANY

STN ANY_BIT

逻辑操作符：

GT

GE

EQ

NE

LE

LT

4.1.8 结构化文本关键字

OpenPCS 在结构化文本编程语言中使用以下关键字：

:= (Assignment)

BY

CASE

DO

ELSE

ELSIF

END_CASE

END_FOR

END_IF

END_REPEAT

END_WHILE

EXIT

FOR

IF

OF

REPEAT

RETURN

TO

UNTIL

WHILE

4.1.9 CANopen

CAN_RECV_EMCY

CAN_RECV_EMCY_DEV

CAN_NMT

CAN_GET_STATE

CAN_SDO_WRITE8

CAN_SDO_READ8

CAN_GET_CANOPEN_KERNEL_STATE

CAN_GET_LOCAL_NODE_ID

CAN_REGISTER_COBID

CAN_PDO_READ8

CAN_PDO_WRITE8

CAN_SDO_READ_STR

CAN_SDO_WRITE_STR

CAN_WRITE_EMCY

CAN_RECV_BOOTUP_DEV

CAN_RECV_BOOTUP

CAN_ENABLE_CYCLIC_SYNC

CAN_SEND_SYNC

4.1.10 Others

ACTION

ANY

ANY_BIT

ANY_DATE

ANY_INT

ANY_NUM

ANY_REAL

CD

CDT

CLK

CONFIGURATION

CU

CV

D(DATE)

D(Action Qualifier)

DS

DT

END_ACTION

END_CONFIGURATION

END_RESOURCE

END_STEP

END_STRUCT

END_TRANSITION

ET

EXPT

FROM

SEMA

Single

SL

STEP

Task

TOD

Transition

ULINT

USINT

VAR_ACCESS

WITH

4.2 关键字(按字序)

4.2.1) (右括号操作符)

右括号操作符执行一条指令，这条指令被左括号修饰符所延缓。

例子:

```
LD a
OR( b (* 指令 OR 的执行被延缓 *)
AND c
) (* 现在执行指令 OR *)
OR( d
AND e
)
ST f
```

注解：

这是一条语言指令表中的指令。

由 IEC61131-3 定义

4.2.2 *_TO_BOOL

0 被转换为 false，其它全部为 true。

字符串转换为布尔型和实型转换为布尔型在相关章节里有描述。

4.2.3 *_TO_STRING

输入

原数据类型*

返回值

转换后的数据类型字符串

此功能块将类型*的第一个值转换为字符串类型的相同值。

以下数据类型可以被转换：

BOOL

true -> true

false -> false

DINT, INT 和 SINT

见下面 REAL

BYTE, DWORD, WORD, USINT, UINT 和 UDINT

位掩码被之间转换为字符串

例如： 001101 -> 001101

REAL

为了转换为字符串，使用了函数 Sprintf(str, %#g , value);

例如：

0.0 -> 0.000000

123.45678 -> 123.456

-12.345678 -> -12.3456

12345678.9 -> 1.23457e+007

0.000000123 -> 1.23000e-007

4.2.4 赋值

赋值把表达式的结果赋给一个变量。.

举例：

```
VAR
  a: INT;
  b: arrey[0..5] OF INT;
  c: REAL;
  e: INT;
END_VAR
a := 5; (* 赋 a 为 5 *)
b[1]:= a*2; e := a; (*两个赋值 *)
e:= REAL_TO_INT( c );
(* 函数调用赋值*)
```

赋值指令将评估表达式右边的值并把结果赋给左边.

注解:

这个关键字仅 ST 语言中有用

这是由 IEC61131-3 定义

4.2.5 ABS

输入

In: ANY_NUM

返回

ANY_NUM

返回输入的绝对值

4.2.6 ACOS

输入

In: REAL

返回

REAL:

返回输入的反余弦值

4.2.7 ACTION

这个关键字由 IEC61131-3 定义，用于程序设计语言 SFC 的文本表示。OpenPCS 不支持 SFC 的文本表示，所以您将不能输入这个关键字，当打印 SFC 时您将看到这个。

4.2.8 ADD

输入

In1: ANY_NUM

In2: ANY_NUM

返回

ANY_NUM

两数之和。 见表 E.1: 错误条件 结果溢出

注意:

标准: IEC61131-3 定义的运算

4.2.9 ADD (time)

输入

In1: TIME 时间持续值

In2: TIME

返回

TIME

时间值之和

注意:

标准: IEC61131-3 定义的运算

4.2.10 AND

输入

IN1: ANY_BIT 输入 1

IN2: ANY_BIT 输入 2

返回

ANY_BIT 逻辑的, 输入 1 和输入 2 的位与

注解

标准: IEC61131-3 定义的函数

4.2.11 ANDN

输入

IN1: ANY_BIT 输入 1

IN2: ANY_BIT 输入 2

返回

ANY_BIT 逻辑的,输入 1 和输入 2 非的与运算

注解:

标准: IEC61131-3 定义的函数

4.2.12 ANY

ANY 是个由 IEC61131-3 定义的 通用 数据类型。不允许使用这个数据类型来声明变量。不管这个数据类型在什么地方使用，都被认为是 ANY_BIT, ANY_DATE, ANY_INT, ANY_REAL 中的一个。

4.2.13 ANY_BIT

ANY_BIT 是个由 IEC61131-3 定义的 通用 数据类型。不允许使用这个数据类型来声明变量。不管这个数据类型在什么地方使用，都被认为是 BOOL, BYTE, WORD, DWORD, LWORD 中的一个。

4.2.14 ANY_DATE

ANY_DATE 是个由 IEC61131-3 定义的 通用 数据类型。不允许使用这个数据类型来声明变量。不管这个数据类型在什么地方使用，都被认为是 DATE, DATE_AND_TIME, TIME_OF_DAY 中的一个。

4.2.15 ANY_INT

ANY_INT 是个由 IEC61131-3 定义的 通用 数据类型。不允许使用这个数据类型来声明变量。不管这个数据类型在什么地方使用，都被认为是 SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT 中的一个。

4.2.16 ANY_NUM

ANY_NUM 是个由 IEC61131-3 定义的 通用 数据类型。不允许使用这个数据类型来声明变量。不管这个数据类型在什么地方使用，都被认为是 ANY_INT, ANY_REAL 中的一个。

4.2.17 ANY_REAL

ANY_REAL 是个由 IEC61131-3 定义的 通用 数据类型。不允许使用这个数据类型来声明变量。不管这个数据类型在什么地方使用，都被认为是 REAL, LREAL.中的一个。

4.2.18 数组

ARRAY 是声明数组元素的关键字,见 导出数据类型

例子：

以下是含五个整数的数组声明并且给定初始值:

```
VAR
 x1: ARRAY[0..4] of INT := [1,2,3,4,5];
END_VAR
```

300 个布尔数的三维数组 ：

```
VAR
 x2: ARRAY[0..4, 15..20, 1..10] of BOOL;
END_VAR
```

一百个结构体的数组:

```
TYPE
 x3: STRUCT
      member1: BOOL;
      member2: INT;
 END_STRUCT;
END_TYPE
VAR
  x4 : ARRAY[1..10,1..10] of x3;
```

END_VAR

多维数组的声明

声明多维数组时，定义一个初始值列表的列表，每个维数分别定义在方括号中。在声明中给出的第一个维数与最外层的方括号相对应。

```
VAR
 x2: ARRAY[0..4, 1..2] of INT := [[1,2], [3,4], [5,6], [7,8], [9,10]];
   x3:          ARRAY[0..1,          0..2,          0..3]          of          INT          :=
   [[[1,2,3,4],[5,6,7,8],[9,10,11,12]],[[13,14,15,16],[17,18,19,20],[21,22,23,24]]];
END_VAR
```

注解：

由于性能的原因 OpenPCS 使用 16 位整数表示数组下标。数组申明不应超过 16 位地址限制，否则导致无定义行为。

4.2.19 ASIN

输入

In: REAL

返回

REAL: 输入反正弦

4.2.20 AT

AT 是一个定义变量地址的关键字，OpenPCS 为这个给定的变量分配地址.

第一个输入位:

```
VAR x1 at %ix0.0: bool; END_VAR
```

输出字开始于第二个输出字节:

VAR x2 at %qw1.0: word; END_VAR

4.2.21 ATAN

输入

In: REAL

返回

REAL: 输入的反正切值

4.2.22 BOOL

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.23 BOOL_TO_*

输入

原数据类型布尔型

返回值

转换后的数据类型*

此功能块将布尔型的第一个值转换为*类型的相同值。

以下数据类型可以被转换:

DINT, INT 和 SINT

BYTE, DWORD, WORD, USINT, UINT 和 UDINT

true -> 1

false -> 0

REAL

true -> 1.0

false -> 0.0

STRING

true -> true

false -> false

4.2.24 BY

见 FOR

4.2.25 BYTE

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.26 CAL

程序将在名字作为操作数传递功能块中继续执行。无条件的调用仅作为顺序结果，而不允许括号操作。

注解:

这是 IL 语言中使用的关键字

这是 IEC61131-3 定义的

也见 EN.

4.2.27 CALC

若 CR 值保持为 TRUE, 功能块作为操作数被调用。若值为 0 , 不调用。程序执行跳转指令后的指令。

注解:

这是 IL 语言用关键字

这是 IEC61131-3 定义的

4.2.28 CALCN

若 CR 值保持为 FALSE, 指定的功能块作为操作数被调用。若 CR 为 1 ,不调用. 程序执行跳转指令后的指令。

注解:

这是 IL 语言用关键字

这是 IEC61131-3 定义的

4.2.29 CAN_ENABLE_CYCLIC_SYNC

允许或阻止循环 SYNC 消息的功能块。

输入

SYNC_MODE: BOOL 允许生成循环 SYNC 消息

SYNC_TIME: TIME 两个 SYNC 消息之间的时间。0 在每次 PLC 循环后生成 1 SYNC

ENABLE: BOOL 输入使能端

NETNUMBER: USINT 网络号

输出

ERROR: WORD 与数据类型 CIA405_CANOPEN_KERNEL_ERROR 一致的错误代码

CONFIRM: BOOL 功能块信号服务完成的输出

功能块 CAN_ENABLE_CYCLIC_SYNC 用于允许信号 SYNC 消息。如果 SYNC_MODE 设置为 TRUE，那么当前一个 SYNC 传递了一个 SYNC_TIME 时，本功能块会在一个 PLC 循环后生成一个 SYNC 消息。

这个功能块只在 有 CANopen Master 的 PLC 模式中的控制单元中。

4.2.30 CAN_GET_CANOPEN_KERNEL_STATE

用于对本地 PLC 的 CANopen 内核状态查询的功能块.

输入

ENABLE 功能块作用使能端

输出

CONFIRM 功能块信号服务完成后的输出

STATE: CIA405_CANOPEN_KERNEL_ERROR 型的状态或错误代码

功能块 CAN_GET_CANOPEN_KERNEL_STATE 用于对本地 PLC 的 CANopen 内核进行状态查询。

4.2.31 CAN_GET_LOCAL_NODE_ID

查询本地节点地址的功能块。.

输入

ENABLE 使能或锁定功能块的输入
输出

DEVICE: PLC 本地节点地址
CONFIRM 功能块信号服务完成后的输出

功能块 CAN_GET_LOCAL_NODE_ID 查询本地节点地址。控制单元的节点地址会影响不同功能块 (有或没有 CANopen 标准的 PLC) 的有效性。

4.2.32 CAN_GET_STATE

查询不同设备的节点状态查询的函数元件。.

输入

DEVICE: 查询的节点地址 (1-127 或 0 本地节点)

ENABLE 功能块作用使能端

输出

STATE: 对应于数据类型 CIA405_STATE 的节点地址

CONFIRM 功能块信号服务完成后的输出

功能块 CAN_GET_STATE 查询指定设备的节点状态。 状态查询基于 Heartbeat 或 Lifeguarding 监测. 输出状态的返回值具有以下含义:

UNKNOWN: 特定地址的 CANopen 设备不支持 Heartbeat 和 Lifeguarding, 因此监测不到状态。在一个无 CANopen 主站的 PLC，或支持状态传输的网络中没有可用的 CANopen 控制的 PLC，或 PLC 有问题处于停止状态 (PLC 程序停止)，都会报告状态情况.

NOT_AVAIL: 指定地址的 CANopen 设备不回答 Heartbeat 或 Lifeguarding 查询，因此对系统不再有效。

其他: 除了状态值 UNKNOWN 和 NOT_AVAIL 以外， 返回值必须和 CiA 标准草稿 30 定义的一致。

带 DEVICE = 0 的功能块调用发送本地 PLC 的本地节点状态.

4.2.33 CAN_NMT

发送 NMT 消息的功能块。

输入

DEVICE: 控制的节点地址 (1-127 或 0 对于所有的节点)

STATE: 相应于 CIA405_TRANSITION_STATE 数据类型的节点状态

ENABLE 功能块使能或锁定的输入

输出

ERROR: 相应于 CIA405_CANOPEN_KERNEL_ERROR 数据类型的错误代码

CONFIRM 功能块信号服务完成后的输出

功能块 CAN_NMT 控制一个节点状态 (DEVICE = 1□127) 或若 DEVICE = 0 网络所有节点.

功能块只在带 CANopen 主站模式下的 PLC 控制单元有效。

4.2.34 CAN_PDO_READ8

通过网络层读取 PDO 和 CAN 第二层消息的功能块。

输入

COBIB: UINT 要读取的消息的 COBID（CAN-Identifier）

ENABLE: BOOL 输入使能端

NETNUMBER: USINT 网络号

输出

CONFIRM: BOOL 功能块信号服务完成的输出

ERROR: WORD 与数据类型 CIA405_CANOPEN_KERNEL_ERROR 一致的错误代码

ERRORINFO: DWORD 关于错误的附加信息

DATA0 - DATA7: BYTE 接收到的消息的数据字节

DATALENGTH: USINT 接收到的消息的长度

功能块 CAN_PDO_READ8 通过网络层读取 PDO 和 CAN 第二层消息。消息必须事先通过 CAN_REGISTER_COBID 进行了注册。接收缓存值保存最近的消息，该消息会从网络层的接收缓存中删除。

在一次成功的读取操作之后，CONFIRM 被设为 TRUE，DATA0 至 DATA7 包含了接收到的消息的单个字节。DATALENGTH 报告有效字节数。空消息也是有效的。

如果指定 COBID 没有消息，那么 CONFIRM 被设定为 FALSE。如果发生了错误，那么 ERROR 也被设置。

4.2.35 CAN_PDO_WRITE8

通过网络层发送 PDO 和 CAN 第二层消息的功能块。

输入

COBIB: UINT 要读取的消息的 COBID（CAN-Identifier）

ENABLE: BOOL 输入使能端

DATA0 - DATA7: BYTE 接收到的消息的数据字节

DATALENGTH: USINT 接收到的消息的长度

NETNUMBER: USINT 网络号

输出

CONFIRM: BOOL 功能块信号服务完成的输出

ERROR: WORD 与数据类型 CIA405_CANOPEN_KERNEL_ERROR 一致的错误代码

ERRORINFO: DWORD 关于错误的附加信息

功能块 CAN_PDO_WRITE8 通过网络层发送 PDO 和 CAN 第二层消息。DATA0 至 DATA7 包含了消息的单个字节。DATALENGTH 报告有效字节数。

如果消息被正确的保存在 CANopen kernel 的发送缓存中，那么 CONFIRM 被设为 TRUE。但是这并不说明消息已被发送。

4.2.36 CAN_RECV_BOOTUP

从网络层的接收缓存读取任意节点的启动消息的功能块。

输入

ENABLE: BOOL 输入使能端

NETNUMBER: USINT 网络号

输出

ERROR: WORD 与数据类型 CIA405_CANOPEN_KERNEL_ERROR 一致的错误代码

CONFIRM: BOOL 功能块信号服务完成的输出

功能块 CAN_RECV_BOOTUP_DEV 用于从网络层的接收缓存读取任意节点的启动消息。如果一个消息被成功接收，CONFIRM 被设置为 TRUE，否则缓存中不包含任何节点的消息。此功能块总是返回最旧的消息（先进先出规则） 并删除缓存中的读取消息。

这个功能块只在 有 CANopen Master 的 PLC 模式中的控制单元中。

4.2.37 CAN_RECV_BOOTUP_DEV

从网络层的接收缓存读取指定节点的启动消息的功能块。

输入

DEVICE: USINT 用于诊断的附加信息

ENABLE: BOOL 输入使能端

NETNUMBER: USINT 网络号

输出

ERROR: WORD 与数据类型 CIA405_CANOPEN_KERNEL_ERROR 一致的错误代码

CONFIRM: BOOL 功能块信号服务完成的输出

功能块 CAN_RECV_BOOTUP_DEV 从网络层的接收缓存读取指定节点的启动消息。如果一个消息被成功接收，CONFIRM 被设置为 TRUE，否则缓存中不包含任何节点的消息。一个消息被读取后将从缓存中删除。

这个功能块只在 有 CANopen Master 的 PLC 模式中的控制单元中。

4.2.38 CAN_RECV_EMCY

功能块用于从网络层接收缓冲器读出一个节点的紧急信息。

输入

ENABLE:bool 功能快使能或锁定输入

输出

DEVICE : 收到突发信息的节点 (1-127) 地址
EMCY_ERR_CODE
EMCY_ERR_REGISTER
EMCY_ERR_FIELD1 - EMCY_ERR_FIELD5
Emergency CiA 标准草稿 301 的标准错误信息
ERROR CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码
CONFIRM 功能块信号服务完成后的输出

CAN_RECV_EMCY 功能块用于从网络层缓冲器读出一个节点的紧急信息。若功能块返回输出 CONFIRM 设置为 TRUE, 输出 DEVICE 报告收到信息的节点地址。EMCY_ERR 包括相应于 CiA 草稿 301 的节点突发错误信息。然而，若 CONFIRM 设置为 TURE, 则网络层缓冲器不包含任何突发事件信息。

功能块总是返回第一个进入接受缓冲器 (= oldest message)的突发信息，信息是按进入顺序擦除，因此每一个突发信息只能被 PLC 程序读一遍。功能块 CAN_RECV_EMCY_DEV 和 CAN_RECV_EMCY 访问同一个接受缓冲器.

此功能块只在 带 CANopen 主站的 PLC 控制模式下控制单元有效。

4.2.39 CAN_RECV_EMCY_DEV

功能块用于从网络层接收缓冲器读出一个指定节点的紧急信息。.

输入

DEVICE 接受紧急信息的节点 (1-127)地址

ENABLE 功能块使能或锁定的输入

输出

EMCY_ERR_CODE

EMCY_ERR_REGISTER

EMCY_ERR_FIELD1 - EMCY_ERR_FIELD5

Emergency CiA 标准草稿 301 的标准错误信息

ERROR CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码

CONFIRM 功能块信号服务完成后的输出

CAN_RECV_EMCY_DEV 功能块用于从网络层缓冲器读出指定节点的紧急信息。若功能块返回输出 CONFIRM 设置为 TRUE, 元素 EMCY_ERR 根据 CiA Draft Standard 301 维护紧急错误信息。然而，若 CONFIRM 设置为 FALSE, 则网络层缓冲器不包含任何紧急事件信息.

功能块总是返回第一个进入接受缓冲器 (= oldest message)的紧急信息，信息是按进入顺序擦除，因此每一个紧急信息只能被 PLC 程序读一遍.功能块 CAN_RECV_EMCY_DEV 和 CAN_RECV_EMCY 访问同一个接受缓冲器.

此功能块只在 带 CANopen 主站的 PLC 控制模式下控制单元有效。

4.2.40 CAN_RESISTER_COBID

通过网络层注册或清除对 PDO 和 CAN 第二层的消息的接收的功能块。

输入

COBIB: UINT 要接收或清除的消息的 COBID（CAN-Identifier）

REGISTER: BOOL TRUE：注册 COBID；FALSE：从注册登记中删除 COBID

NETNUMBER: USINT 网络号

输出

CONFIRM: BOOL 功能块信号服务完成的输出

ERROR: WORD 与数据类型 CIA405_CANOPEN_KERNEL_ERROR 一致的错误代码

功能块 CAN_REGISTER_COBID 通过网络层注册 PDO 和 CAN 第二层的消息的接收或根据输入参数 REGISTER 清除它的注册。网络层中的消息及所有注册信息将通过 COBID=0 进行删除。

消息必须在网络层中进行注册后，才能被象 CAN_PDO_READ8 这样的功能块读取。

4.2.41 CAN_SDO_READ8

功能块通过一个 SDO 传递来读出节点对象入口。

输入

DEVICE 读节点地址 (1-127 or 0 为本地 OD)

INDEX 需读入口的索引数字

SUBINDEX 需读入口的子索引数字

ENABLE 输入使能端

输出

DATA0 - DATA7 需读入口的数据字节

DATALENGTH 需读入口的长度

ERROR: CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码

ERRORINFO CIA405_SDO_ERROR 型的 SDO 异常通讯代码

CONFIRM 功能块信号服务完成后的输出

功能块 CAN_SDO_READ8 通过 SDO 转换读出当前节点对象入口。SDO 转换一般在后台完成.

依据功能块的输出 CONFIRM 置为 TRUE，就按 DATA0 到 DATA7 逐个读出入口对象路径地址。输出 DATALENGTH 记录有效的数据字节个数(从 DATA0 开始).

任一时刻网络层仅允许一个 SDO 转换器通过 PLC 程序。SDO 转换器置 ENABLE 为 TRUE,就启动了，SDO 通道被锁定，以防止其他组件使用.一直保持直到数据转换结束 后，通过 SDO 功能块 ENABLE 置为 FALSE，才解除锁定状态.

带 DEVICE = 0 的功能块调用将产生一个 PLC 本地对象字典的入口。 因此可读出本地对象字典的值。

4.2.42 CAN_SDO_READ_STR

通过一个 SDO 传递从对象字典中读取字符串的功能块。

输入

DEVICE: USINT 读节点地址 (1-127 or 0 为本地 OD)

INDEX: WORD 需读入口的索引数字

SUBINDEX: BYTE 需读入口的子索引数字

SDOTYPE: SUBINDEX SDO 传递的类型。标准为 SDO_TYPE_AUTO_BEST_CASE

RXDATA: STRING 存储需读字符的字符串变量

MAXLENGTH: INT 需读字符数的上限

ENABLE: BOOL 输入使能端

NETNUMBER: USINT 网络号

输出

RXDATA: STRING 存储需读字符的字符串变量

RXLENGTH: INT 需读字符串的长度

ERROR: WORD CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码

ERRORINFO: DWORD CIA405_SDO_ERROR 型的 SDO 异常通讯代码

CONFIRM: BOOL 功能块信号服务完成后的输出

功能块 CAN_SDO_READ_STR 通过一个 SDO 传递从对象字典中读取字符串。

所有 SDO 转换都在后台完成。同步可以通过 ENABLE 和 CONFIRM 来处理。

如果 CONFIRM 为 TRUE，那么读取的字符串保存在 RXDATA 中，长度保存在 RXLENGTH 中。

4.2.43 CAN_SDO_WRITE8

功能块通过传递一个 SDO 来写节点入口..

输入

ENABLE: BOOL 功能块输入使能端

DEVICE: USINT 需写节点的地址(1-127 或 0 对本地对象字典)

INDEX: WORD 需写入口的索引数字

SUBINDEX: BYTE 需写入口的子索引数字

DATA0 - DATA7: BYTE 需写入口的数据字节

DATALENGTH: USINT 需写入口的长度

NETNUMBER: USINT 网络号

输出

CONFIRM: BOOL 功能块信号服务完成后的输出

ERROR: WORD CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码

ERRORINFO: DWORD CIA405_SDO_ERROR 型的 SDO 异常通讯代码

功能块 CAN_SDO_WRITE8 通过 SDO 转换写当前节点对象入口. SDO 转换一般在后台完成.

需写对象入口的各个字节传入 DATA0 到 DATA7. 输入 DATALENGTH 指定有效数据字节个数 (从 DATA0 开始).

任一时刻网络层仅允许一个 SDO 转换器通过 PLC 程序. SDO 转换器置 ENABLE 为 TRUE,就启动了，SDO 通道被锁定，以防止其他组件使用.一直保持直到数据转换结束后，通过 SDO 功能块 ENABLE 置为 FALSE ，才解除锁定状态.

带 DEVICE = 0 的功能块调用将产生一个 PLC 本地对象字典的入口。 因此值可写到本地对象字典.

4.2.44 CAN_SDO_WRITE_STR

通过 SDO 传递向一个节点的对象字典中写入字符串的功能块。

输入

DEVICE: USINT 读节点地址 (1-127 or 0 为本地 OD)

INDEX: WORD 需读入口的索引数字

SUBINDEX: BYTE 需读入口的子索引数字

SDOTYPE: SUBINDEX SDO 传递的类型。标准为 SDO_TYPE_AUTO_BEST_CASE

TXDATA: STRING 存储需写字符的字符串变量

TXLENGTH: INT 需写字符数的上限

ENABLE: BOOL 输入使能端

NETNUMBER: USINT 网络号

输出

TXDATA: STRING 存储需读字符的字符串变量

ERROR: WORD CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码

ERRORINFO: DWORD CIA405_SDO_ERROR 型的 SDO 异常通讯代码

CONFIRM: BOOL 功能块信号服务完成后的输出

功能块 CAN_SDO_READ_STR 通过 SDO 传递向一个节点的对象字典中写入字符串。所有 SDO 传递都是在后台执行的。可以通过 ENABLE 和 CONFIRM 来对同步进行处理。

由于网络层同一时间只支持一个 SDO 传递，因此在此功能块使用这个 SDO 通道时，该通道会对其它功能块封闭。

4.2.45 CAN_SEND_SYNC

启用或关闭循环 SYNC 消息的功能块。

输入

ENABLE: BOOL 输入使能端

NETNUMBER: USINT 网络号

输出

ERROR: WORD CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码、

CONFIRM: BOOL 功能块信号服务完成后的输出

如果 ENABLE 为真，那么功能块 CAN_SEND_SYNC 发送一个 SYNC 消息。

此功能块只在 带 CANopen 主站的 PLC 控制模式下控制单元有效。

4.2.46 CAN_WRITE_EMCY

通过网络层发送应用程序特定的紧急事件消息的功能块。

输入

EMCY_ERR_CODE: WORD

EMCY_ERR_REGISTER: BYTE

EMCY_ERR_FIELD1 - EMCY_ERR_FIELD5: BYTE

符合 CiA 草拟标准 301 的紧急错误信息

EMCY_ADD_INFO: WORD 为诊断提供的附加信息

EANBLE: BOOL　输入使能端

NETNUMBER: USINT　网络号

输出

ERROR: WORD　CIA405_CANOPEN_KERNEL_ERROR 类型的错误代码、

CONFIRM: BOOL　功能块信号服务完成后的输出

功能块 CAN_WRITE_EMCY 用于通过网络层发送应用程序特定的紧急事件消息。根据 IEC61131-3，各项 EMCY_ERR 成员中须保存相关的紧急错误信息。

如果消息被保存在 CANopen 缓存中，那么 CONFIRM 被设置为真。但是，该功能块并不说明消息是否被成功发送。

4.2.47　CASE

尽管 IF 指令可以是嵌套的，但是每次检查一个条件使用 IF 看起来会很复杂。CASE 指令可以使用一个指令检查多个值。CASE 指令的表达式是 INT 类型，只有与这个的 INT 值相应的指令才被执行。之后，执行 END_CASE 后面的第一条指令。

如果 IF 表达式不适合任一种 case 值，则执行 ELSE 后面的第一条指令。这条指令是可选的。

```
CASE expression OF
  case_value1: { instructions; }
  case_value2: { instructions; }
...
case_valueN: { instructions; }
[ ELSE instructions; ]
END_CASE;
```

例如:

```
    VAR
number : INT:= 10;

      amount : INT :=2;

      END_VAR
      CASE number OF

       10: amount := amount +1;

       11: amount := amount -1;

    ELSE
amount := number;

    END_CASE;
```

在这个例子中， number 的值将被确定，如果它是 10， amount 将增加，如果它等于 11， amount 将减少。其它情况 amount 将等于 number 。

注解:

这是仅 ST 语言中使用的关键字

这是 IEC61131-3 定义的

4.2.48 CD

这是个标准功能块(CTD)的一个形式参数的名称，并且定义为一个关键字。

4.2.49 CDT

这是个标准功能块(RTC)的一个形式参数的名称，并且定义为一个关键字。

4.2.50 CLK

这是个标准功能块(R_TRIG)的一个形式参数的名称，并且定义为一个关键字。

4.2.51 CONCAT

输入

In1: STRING 第一个字符串

In2: STRING 第二个字符串

返回

STRING 两字符串串连

字符串 IN1 和 IN2 串连起来，成为新字符串，且装载到工作寄存器。字符串 IN1 和 IN2 从左到右顺序按升序连接起来。

增加输入管脚的功能对此功能块有效。

4.2.52 Configuration

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字，

4.2.53 CONSTANT

CONSTANT 是申明值不被程序代码改变的变量的关键字。如果您试图写这个变量：VAR CONSTANT x1 : INT := 15; END_VAR ， OpenPCS 编译器将会给出一个错误码:

见 声明部分

4.2.54 COS

输入

In: REAL

返回

REAL: 输入的余弦值

4.2.55 CR

CR 是 Current Result 的缩写, 在 IEC61131-3 编程语言是个虚拟累加器。

4.2.56 CTD

功能块 CTD 以减法计算输入 CD 的脉冲数。初始化时，计数器置 0。

若 LOAD 置 1，PV 值将为计数器初值。输入 CD 的每个上升沿将使计数器减 1。

输出 CV 为计数器当前值。若计数器值为正，输出 Q 将置 0；若计数器值为 0 或负，输出 Q 置 1。

输入

CD: 布尔型 计数脉冲，上升沿

LOAD: 布尔型 设置条件

PV: 整型　初始值

输出

Q: 布尔型 信号（计数值是否到 0）

CV: 整型　计数值

注解

标准: IEC61131-3 定义的功能块。

CTU

功能块 CTU 以加法计算输入 CU 的脉冲数，初始化时，计数器置 0。

若 RESET 收到值 1，计数值将重置。

输入 CU 的每个上升沿使计数器加 1。

输出 CV 为当前计数值。若计数值小于 PV 上限值，输出 Q 将为布尔 0 ，若大于或等于 PV 值， Q 置 1。

输入

CU：布尔型 计数器脉冲

RESET：布尔型 重启计数器

PV：整型 计数上限

输出

Q： 布尔型 计数器是否到达上限值

CV：整型 当前计数器值

注解

标准: IEC61131-3 定义的功能块

4.2.57 CTUD

功能块 CTUD 计数上升沿和下降沿脉冲。初始化时，计数器置 0 。输入 CD 的每个上升沿；将使计数器加 1；而 CD 的每个下降沿，计数器减 1。

若 LOAD 为 1 ， PV 值将预置到计数器。

若 RESET 值为 1，计数器将重置；若计数值少于 PV 预置值，输出 Q 将显示布尔值 0 ；若计数值到达或超过预置值，输出 Q 置 1。若计数值为正，输出 QD 将为布尔值 0 。若计数值为 0 或负，输出 QD 将置 1 。

输入

CU: 布尔型 计算上升、下降脉冲

CD: 布尔型 计算上升、下降脉冲

RESET: 布尔型 重置条件

LOAD: 布尔型 下载条件

PV: 整型 下载值

输出

QU: 布尔型 信号（计数值是否达到 PV 值）

QD: 布尔型 信号（计数状态是否为 0 ）

CV: 整型 计数状态

注解

标准: IEC61131-3 定义的功能块

4.2.58 CU

这是个标准功能块(CTU)的一个形式参数的名称，并且定义为一个关键字。

4.2.59 CV

这是个标准功能块(CTD)的一个形式参数的名称，并且定义为一个关键字。

4.2.60 D

当指定一个文字常量的数据类型时，D 可以用作 DATE 的缩写。作为数据类型，在 OpenPCS 中不执行 DATE，您不能在 OpenPCS 中使用这个关键字。

4.2.61 D(动作识别符)

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.62 DATE

见 基本数据类型

注解:

标准： 这是 IEC61131-3 定义的数据类型

注意： 当前 OpenPCS 中不使用这个数据类型

4.2.63 DATE_AND_TIME

见 基本数据类型

注解:

标准: 这是 IEC61131-3 定义的数据类型

注意： 当前 OpenPCS 中不使用这个数据类型

4.2.64 DELETE

输入

IN1: STRING 部分需要删除的基本字符串

L: ANY_INT (根据支持) 需山初部分子字符串的长度。L < 0 无效。

P: ANY_INT (根据支持) 子字符串起始位置。P < 0 无效。

返回

STRING 缩短的字符串。IN1 参数无效。

DELETE 函数在给定的字符串 IN1 中从 P 位置开始删除长度为 L 的子字符串。

注解

标准: 这是 IEC61131-3 定义的函数

4.2.65 DINT

见 基本数据类型

注解:

标准: 这是 IEC61131-3 定义的数据类型

4.2.66 DIV

输入

In1: ANY_NUM 被除数

In2: ANY_NUM 除数

返回

ANY_NUM 商

两数相除。若除数为 0，出现错误，见 表 E1：错误条件

注解:

标准: 这是 IEC61131-3 定义的运算

4.2.67 DIV (time)

输入

In1: TIME 时间持续值

In2: ANY_NUM 除数

返回

TIME 除后时间值

时间值相除。

注解:

标准: 这是 IEC61131-3 定义的运算。

4.2.68 DO

见 FOR 和 WHILE

4.2.69 DS

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.70 DT

当指定一个文字常量的数据类型时，DT 可以用作 DATE_AND_TIME 的缩写。作为数据类型，在 OpenPCS 中不执行 DATE_AND_TIME，您不能在 OpenPCS 中使用这个关键字。

4.2.71 DWORD

见 基本数据类型

注解:

标准: 这是 IEC61131-3 定义的数据类型

4.2.72 ELSE

见 CASE 和 IF

4.2.73 ELSIF

见 IF

4.2.74 EN

功能块可能有一个布尔型的输入变量 EN。如果是这种情况，那么当且仅当这个实例的输入变量 EN 的值为 TURE 时，这个功能块的实例执行启动 。

也见 CAL 和 ENO.

注解：

1. . EN 是 Enable 的缩写。
2. . 如果输入和 / 或者输出变量由 CAL 指令被指定为相同的状态，即使由于 EN＝FALSE，CAL 没有被激活，这些任务也照样执行。
3. . EN 的缺省值是 TRUE。

4.2.75 END_ACTION

这个关键字由 IEC61131-3 定义，用于程序设计语言 SFC 的文本表示。OpenPCS 不支持 SFC 的文本表示，所以您将不能输入这个关键字，当打印 SFC 时您将看到这个。

4.2.76 END_CASE

见 CASE

4.2.77 END_CONFIGURATION

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字，

4.2.78 END_FOR

见 FOR

4.2.79 END_FUNCTION

见 Function.

4.2.80 END_FUNCTION_BLOCK

见 Function Block.

4.2.81 END_IF

见 IF

4.2.82 END_PROGRAM

见 PROGRAM

4.2.83 END_REPEAT

见 REPEAT

4.2.84 END_RESOURCE

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字，

4.2.85 END_STEP

这个关键字由 IEC61131-3 定义，用于程序设计语言 SFC 的文本表示。OpenPCS 不支持 SFC 的文本表示，所以您将不能输入这个关键字，当打印 SFC 时您将看到这个。

4.2.86 END_STRUCT

查看 STRUCT.

4.2.87 END_TRANSITION

这个关键字由 IEC61131-3 定义，用于程序设计语言 SFC 的文本表示。OpenPCS 不支持 SFC 的文本表示，所以您将不能输入这个关键字，当打印 SFC 时您将看到这个。

4.2.88 END_TYPE

见 声明部分

注解：

这是 POU 声明部分的关键字

这由 IEC61131-3 定义

4.2.89 END_VAR

见 声明部分

注解:

这是 POU 声明部分的关键字

这由 IEC61131-3 定义

4.2.90 END_WHILE

见 WHILE

1. **ENO**

功能块可能有一个布尔型的输出变量 ENO。通常定义 TRUE 为执行正确信号，FALSE 是执行错误信号。通常这个 ENO 跟另一个功能块的输入 EN 连起来。

注解:

ENO 是 Enable Output 的缩写

4.2.91 EQ

输入

IN1: ANY 输入 1

IN2: ANY 输入 2

返回

布尔型 输入 1 等于 输入 2 时为真

注解

标准: 这是 IEC61131-3 定义的函数

4.2.92 ET

这是个标准功能块(TOF)的一个形式参数的名称，并且定义为一个关键字。

4.2.93 ETRC

一般来说，一个事件任务只被执行一次。因为某些事件的反应会比一个时钟循环长，所有有必要重启当前的任务。为了实现这个可以使用 ETRC(Event Task Run Control)。它会延长它自己的事件任务另一个时钟循环。此外，功能块还在它的输出口提供诸如循环计数或从第一次调用 ETRC 实例的经历时间。利用这些信息，那些会变成无限循环的错误的反应可以被处理：

输入：

IN：BOOL TRUE: 事件任务应启动另一个循环

FALSE: 事件任务无需再启动。这个功能块被调用只是为了获得输出信息

输出：

Q：BOOL TRUE: 事件任务将会执行多一个循环

FALSE: 事件任务将会在此循环结束后中止

EVC：USINT 事件代码(EVC)描述了内部的原因为何事件任务会被调用

ERT：TIME 经历时间(ERT)返回了从第一次当前事件任务启动以来的时间

CCV：UDINT 循环计数表示了事件任务执行的次数

ERROR：USINT ETRC 执行的返回值

0: 成功执行

1:无法执行因为功能在任务外被调用(无效的调用)

功能块的事件代码：

0	被叫任务未知
1	执行了冷启动
2	执行了暖启动
3	执行了热启动
4	执行了单循环启动
5	PLC 被硬件的 RUN/STOP 开关停止
6	PLC 被软件的停止
7	经历一个单循环后 PLC 进入 STOP 状态
8	PLC 执行时的一般错误
9	除数为 0
10	无效的数组索引访问
11	执行固件功能块时发生错误

4.2.94 EXIT

接到循环指令前，任何循环将跳出程序控制。EXIT 指令跳到循环内部之后的第一条指令处。

例如：

```
VAR
start: INT :=0;
summe: INT :=0;
ende : INT := 10;
END_VAR
FOR Start := 1 TO Ende BY 2 DO
```

```
Summe := Summe + 1;
IF Summe > 4 THEN
EXIT;
END_IF;
END_FOR;
(* 在这里继续 *)
```

如果 Summe 大于 4, 跳出 FOR 循环.

注解:

这仅是 ST 语言使用的关键字

这是 IEC61131-3 定义的

4.2.95 EXP

输入

In: 实型

返回

实型: e 的 In 次幂

4.2.96 EXPT

输入：

In1 : ANY_REAL

In2 : ANY_NUM

返回：：

ANY_REAL: In1 ** In2

4.2.97 F_EDGE

F_EDGE 用于说明布尔型输入的下降沿探测函数。这将引出一个 F_TRIG 类型的功能块的内在定义。

例子:

```
FUNCTION_BLOCK AND_EDGE

VAR_INPUT
        X : BOOL R_EDGE;
        Y : BOOL F_EDGE;
END_VAR

VAR_OUTPUT
        Z : BOOL ;
END_VAR

        Z := X AND Y ; (* ST language example *)

END_FUNCTION_BLOCK
```

4.2.98 F_TRIG

输入

CLK: 布尔型 操作输入，仅对下降沿检测

输出

Q: 布尔型 操作输出，仅在 CLK 有下降沿时置位。

功能块 F_TRIG 检测输入操作 CLK 的状态。通过输出 Q 来检测处理循环中状态由 1 变为 0 的变化。在一个处理周期内，即检测到 CLK ，并且下降沿指示的时候，输出为 1 。

注解

标准: 这是 IEC61131-3 定义的功能块

4.2.99 FALSE

BOOL.类型的常数

4.2.100 FBD

FBD 是 Function Block Diagram 的缩写,是 IEC61131-3 编程语言的一种。

4.2.101 FBD 语言要素

[注意：这一章是空的，因为在 IEC1131-3 中相应的章节中没有列作出它的任何特性以供参考]。

4.2.102 FIND

在另一个字符串中查找一个字符串。

输入

IN1: 字符串 待寻找的基本字符串序列; 通过工作寄存器使这个字符串有效

IN2: 字符串 要从 IN1 基本字符串中查找的字符串.

返回

INT 第一个出现的位置

从 IN1 基本字符串中查找特殊字符序列。若找到，这个序列的第一个字符位置进入工作寄存器，否则， 0 进入工作寄存器。若找到不止一个，则首先找到的字符位置进入工作寄存器。

在程序 search 调用 FIND 函数

```
PROGRAM search
VAR
Basic_Text : STRING := □StartupCondition□ ;
Search_Text : STRING := □Switch□ ;
Position : INT;
END_VAR
LD Basic_Text
FIND Search_Text
ST Position (* Position: 4 *)
END_PROGRAM
```

注解

标准: 这是定义的函数 IEC61131-3

4.2.103 FOR

使用 FOR 循环，循环控制变量将被设置为一个特定的起始值，然后增加（或减少）该变量，当循环遇到给定的值时终止。

语法:

```
FOR assignment TO Endvalue BY Increment DO
    Instructions;
END_FOR;
```

例如：

```
VAR
Field : arrey[1..5] OF INT :=[2,14,8,12,5];
```

```
  Index : INT;

  MaxIndex : INT :=5;

Maximum : INT :=0;

 END_VAR
 FOR Index :=1 TO MaxIndex BY 1 DO

  IF Field[Index] > Maximum THEN

Maximum := Field[Index];

   END_IF;
 END_FOR;
```

控制变量的索引 Index 初始化为 1，每执行一个循环后，索引的增量为 1。当最大索引值为 5 时，终止循环。

注解: BY 部分是可选项，可以省略。1 是默认的增量。

执行 FOR 循环：

初始化控制变量

如果有必要，则检查结束标准和停止循环的执行

执行语句块

增加控制变量

转到第二步

注解:

这是仅 ST 语言用关键字

这是 IEC61131-3 定义的

4.2.104 FROM

查看 转换.

4.2.105 Function

IEC61131-3 定义三种块类型: PROGRAM, FUNCTION 和 FUNCTION BLOCK。更多见 高级主题 下的 块类型。

函数返回值赋值给变量，且与函数有相同的名和类型。例如

```
FUNCTION MyFun : INT

□

MyFun := 999;

END_FUNCTION
```

注解:

1. IEC61131 以 END_FUNCTION 或 RETURN 当前的结果作为函数返回。OpenPCS 忽略这个值仅使用赋给的函数名的值。.
2. 关键字 FUNCTION 和 END_FUNCTION 在 OpenPCS 一般不可见，他们在编辑器内部维护。
3. 函数返回类型(INT 在以上例子) 由与您指定函数名相同的对话框中选择，一般在底部。默认类型是 BOOL。
4. 您也可以通过在输入区域添加数据类型名称，从而加入用户定义的数据类型（(STRUCT s, ARRAY s, 等)。

4.2.106 FUNCTION BLOCK

IEC61131-3 定义三种块类型: PROGRAM, FUNCTION 和 FUNCTION BLOCK。更多见 高级主题 下的 块类型。

关键字 FUNCTION 和 END_FUNCTION 在 OpenPCS 一般不可见，他们在编辑器内部维护。

4.2.107 GE

输入

IN1: ANY 输入 1

IN2: ANY 输入 2

返回

布尔型 输入 1 大于或等于 输入 2 时为真

注解

标准: 定义的函数 IEC61131-3

4.2.108 GetDateStruct

输入

IN: Date 须转换的日期

输入输出

DATESTRUCT_INOUT：DateStruct 日期结构体

如：

```
DateStruct: STRUCT
 YEAR：UINT;
 MONTH：USINT;
 DAY：USINT;
END_STRUCT
```

GETDATESTRUCT 将一个给定的日期转换为结构体日期，以单独整型变量分别表示日，月，年。

4.2.109 GETSYSTEMDATEANDTIME

输入

EN: BOOL

输出

ENO: BOOL

ODT: DATE_AND_TIME

功能块 GetSystemDateAndTime 返回 ODT 中真实的系统时间。

注意：

标准化：这个功能块并未在 IEC61131-3 中定义。

4.2.110 GetTaskInfo

输出：

Count: 双字型; (*执行的任务循环数 *)

LastCT: 时间型; (*最后周期需要的时间*)

AverageCT: 时间型; (*执行的平均时间*)

MinCT: 时间型; (*执行的最小时间*)

MaxCT: 时间型; (*执行的最大时间*)

State: 双字型; (*未用*)

GetTaskInfo 返回 当前任务最后循环的执行时间信息。这个功能块无输入参数。

4.2.111 GetTime

输入

IN1: TIME 先前的时间

返回

TIME: 上电后消耗的时间, 减去 IN1

GETTIME 将恢复从控制器最后一次打开以来所消耗的时间，减少作为输入提供的时间值。这可以用来很容易的测量时间间隔。

例如 Stop Watch

```
PROGRAM StopW
VAR
 begin, result : TIME;
END_VAR
start:
 LD t#0ms
 GETTIME
 ST begin
  ...

stop:
 LD begin
 GETTIME
 ST result
END_PROGRAM
```

4.2.112 GetTimeCS

获得当前系统时间

输入

IN1: TIME 以前时间

返回

TIME: 上电后消耗的时间,减去 IN1

GETTIME 将恢复从控制器最后一次打开以来最后的系统控制点所消耗的时间，最小时间值作为输入被提供。这可很容易地用来测量时间跨距。对比于 GETTIME，GETTIMECS 将在相同循环内调用多个时间时返回相同的值。

例如 Stop Watch

```
PROGRAM StopW
VAR
 begin, result : TIME;
END_VAR
...
start:
 LD t#0ms
 GETTIMECS
 ST begin
  ...

stop:
 LD begin
 GETTIMECS
 ST result
END_PROGRAM
```

4.2.113 GetVarData

输入输出

VarName: 字符串 变量名
输出

Q: 布尔型 VarInfo 有效时为真
VarData: VarInfo 变量信息

被指定为输入的变量定位于内存地址空间，返回变量信息. 若变量不能地址, Q 返回 FALSE.

请注意:

OpenPCS 通过名字定位变量, 生成一个 MAP 文件 (资源选项)

VARINFO 的定义, 见 关键字 下 VARINFO。

4.2.114 GetVarFlatAddress

输入

VarName: STRING 变量名
输出

Q: bool VarInfo 有效时为 TRUE
Address: DWORD 特定变量的内存地址
这个输入变量位于内存地址空间，并且返回它的位置地址。如果这个变量不能被赋予地址，那么 Q 返回 FALSE。

注解：

对于 OpenPCS 来说，要能通过名字定位变量，必须生成一个 MAP 文件（在资源选项中），返回的内存地址不必保存和使用，除了当前执行循环。

4.2.115 GT

输入

IN1: ANY 输入 1

IN2: ANY 输入 2

返回

布尔型 输入 1 大于或等于 输入 2 时为真

注解

标准: IEC61131-3 定义的函数。

4.2.116 IF

IF 指令有以下句法：

IF 表达式 **THEN** 块
{ **ELSIF** 表达式 **THEN** 块}

[**ELSE** 块]

END_IF;

如果 IF 后面的表达式值为 ture ，则执行 THEN 后面的指令。如果 IF 后面的表达式值为 false ，将执行 ELSE 后面的指令或者检查 ELSEIF 条件句。无论如何将在 END_IF 后面继续实行下一个命令。

下面的 IF 指令将计算两个数的最大值：

```
IF a>b THEN
maximum := a;

ELSE
maximum := b;

END_IF;
```

IF 指令可以是嵌套的，即 THEN 部分或 ELSE 部分可以包括其他 IF 指令。

例如:

下面的程序将再次计算两个数的最大值，但是如果最大值是 a 且 a 大于 10，则它将减 1：

```
VAR
a: INT :=12;

  b: INT :=5;
```

```
 maximum: INT;

END_VAR
IF a>b THEN

 maximum :=a;

IF (a>10) THEN

 a:=a-1;

ELSE

   a:=a+1;
END_IF;
ELSE
maximum                                                                    :=b;
END_IF;
```

注解:

这是仅 ST 语言用关键字

这是 IEC61131-3 定义的

4.2.117 IL

IL 是 Instruction List 缩写, IEC61131-3 的一种编程语言.

4.2.118 IN

这是个标准功能块(TOF)的一个形式参数的名称，并且定义为一个关键字。

4.2.119 INITIAL_STEP

这个关键字由 IEC61131-3 定义，用于程序设计语言 SFC 的文本表示。OpenPCS 不支持 SFC 的文本表示，所以您将不能输入这个关键字，当打印 SFC 时您将看到这个。

4.2.120 INSERT

输入

IN1: STRING　字符串

IN2: STRING　待插入的字符串

P: ANY_INT (根据支持)　起始位置。P < 0 和 P > LEN(IN1) 无效

返回

STRING　组合的字符串。 IN1 无效参数

INSERT 函数将字符串 IN2 插入字符串 IN1。拼接后的字符串由 IN1 前 P 个字符，完整的 IN2 和剩下的 IN1 组成。

注解

标准: 这是 IEC61131-3 定义的函数

4.2.121 INT

参见 标准数据类型

注解:

标准: 这是 IEC61131-3 定义的数据类型

4.2.122 Interval

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字，

4.2.123 JMP

程序在指定的跳转目标继续执行。跳转目标必须是个由标号唯一确定顺序开始。

跳转仅允许在一个 POU.

注解:

这是 IL 语言用关键字

这是 IEC61131-3 定义的

4.2.124 JMPC

若 CR 值为 TRUE, 程序跳转到指定的目标继续执行. 若值为 0 , 不跳转。程序在跳转指令后继续执行。

注解:

这是 IL 语言用关键字

这是 IEC61131-3 定义的

4.2.125 JMPCN

若 CR 值为 FALSE, 程序跳转到指定的目标继续执行. 若值为 1 , 不跳转。程序在跳转指令后继续执行。

注解:

这是 IL 语言用关键字

这是 IEC61131-3 定义的

4.2.126 L

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.127 LD

操作数赋值，并装载当前结果值.数据重新存到 CR。操作数是不改变的。操作数类型决定了连续的操作数的类型。

注解:

这是 IL 语言用关键字

这是 IEC61131-3 定义的

4.2.128 LD (梯形图)

LD 是 Ladder Diagram 的缩写, IEC61131-3 的一种编程语言。

4.2.129 LDN

操作数被赋值,当前结果的值是负数。而且操作数是不可改变的。操作数类型决定连续的操作数可可允许的数据类型.

注解:

这是 IL 语言用关键字

这是 IEC61131-3 定义的

4.2.130 LE

输入

IN1: ANY 输入 1

IN2: ANY 输入 2

返回

布尔型 输入 1 小于或等于 输入 2 时为真

注解

标准: 定义的函数 IEC61131-3

4.2.131 LEFT

输入

IN: 字符串型 字符串

L: ANY_INT(根据支持) 要得到的字符数

返回

STRING IN 左边 L 个字符，IN 如果参数无效

LEFT 函数获得当前字符串的左边 L 个字符，并放入到工作寄存器。输入操作数 L 定义输入字符个数。

4.2.132 LEN

输入

In: STRING 字符串

返回

INT IN 的长度

函数 LEN 计算字符串长度 (输入操作数类型为 STRING) 并把长度值以 INT 型数放入工作寄存器中。

4.2.133 LIMIT

输入

MN: Any_Num 下限

IN: Any_Num 测试值

MX: Any_Num 上限

返回

Any_Num 其中一个 输入值 , 见 描述

MN 和 MX 值定义了上下限值。函数比较 IN 值与 MN 和 MX 大小。若 IN 在两限值之间, 它将装载到工作寄存器。若 IN 小于 MN , 输出 MN 值.若 IN 大于 MX ,输出 MX 值。

注解

标准: IEC61131-3 定义的函数

4.2.134 LINT

这是一个基本数据类型名称，IEC61131-3 定义，但 OpenPCS 不支持。查看一致性声明中的表 10。

4.2.135 LN

输入

In: REAL

返回

REAL: 以 e 为底的对数

4.2.136 LOG

输入

In: REAL

返回

REAL: 以 10 为底的对数

4.2.137 Lreal

这是个基本数据类型的名称，由 IEC61131-3 定义，但 OpenPCS 不支持。查看一致性声明中的表 10。

4.2.138 LT

输入

IN1: ANY 输入 1

IN2: ANY 输入 2

返回

布尔型 输入 1 小于 输入 2 时为真

注解

标准: IEC61131-3 定义的函数

4.2.139 Lword

这是个基本数据类型的名称，由 IEC61131-3 定义，但 OpenPCS 不支持。查看一致性声明中的表 10。

4.2.140 MAX

输入

In1: Any_Num 输入值 1

In2: Any_Num 输入值 2

□

InN: Any_Num 输入值 N

返回

Any_Num 所有输入值的最大值

The MAX 函数确定哪个输入操作数有最大值 选中的操作数载入工作寄存器中.

注解

标准: IEC61131-3 定义的函数

4.2.141 MID

输入

IN: STRING 字符串

L: ANY_INT (根据支持) 得到字符串的长度。 L < 0 无效

P: ANY_INT (根据支持) 起始位置。P <= 0 和 P > LEN(IN) 无效

返回

STRING IN 中从第 P 个字符开始的 L 个字。IN 如果参数无效

MID 函数将中间一部分当前调入的字符串输入寄存器。输入 P 定义了首个输入的字符。L 定义输入字符串的长度。

注解

标准: 这是 IEC61131-3 定义的函数

4.2.142 MIN

输入

In1: Any_Num 输入 Value1

In2: Any_Num 输入 Value2

□

InN: Any_Num 输入 ValueN

返回

Any_Num 所有输入的最小值

MIN 函数确定哪个输入有最小的值。并把最小值装载到工作寄存器中.

注解

标准: IEC61131-3 定义的函数

4.2.143 MOD

输入

In1: ANY_INT

In2: ANY_INT

返回

ANY_INT

第一个输入被第二个输入除，MOD 返回当前结果的余数.

4.2.144 MOVE

输入

In: ANY

输出

Out: ANY

函数 MOVE 是个赋值运算的函数。

4.2.145 MUL

输入

In1: ANY_NUM 被乘数

In2: ANY_NUM 乘数

返回

ANY_NUM 乘积

两数相乘. 见 表 E.1: 错误条件 溢出结果.

注解:

标准: IEC61131-3 定义的运算

4.2.146 MUL (时间)

输入

In1: TIME 时间值

In2: ANY_NUM 被乘数

返回

TIME 时间值的乘法

时间值的乘法

注解:

标准: IEC61131-3 定义的运算

4.2.147 N (动作识别符)

这是一个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，所以您不必使用在 OpenPCS 中这个这个关键字。

4.2.148 NCC

NCC 是 native code compiler 首字母的缩写。

4.2.149 NE

输入

IN1: ANY 输入 1

IN2: ANY 输入 2

返回

布尔型 输入 1 不等于输入 2 时为真

注解

标准: IEC61131-3 定义的函数

4.2.150 NEG

输入

In: ANY_NUM

返回

ANY_NUM: 输入值的非运算

4.2.151 NOT

输入

IN1: ANYBIT 输入

返回

ANYBIT 输入的逻辑非运算

注解

标准: IEC61131-3 定义的函数

4.2.152 OF

见 CASE

4.2.153 On

查看 资源.

4.2.154 OPC

变量限定 OPC 允许用户标示一个已经在 OpenPCS 声明编辑器中存在的专有变量成为变量表中的一部分。

请参照声明区域 。

4.2.155 OR

输入

IN1: ANY_BIT 输入 1

IN2: ANY_BIT 输入 2

返回

ANY_BIT 逻辑的, 输入 1 和输入 2 的或运算

注解

标准: IEC61131-3 定义的函数

4.2.156 ORN

输入

IN1: ANY_BIT 输入 1

IN2: ANY_BIT 输入 2

返回

ANY_BIT 逻辑的， 输入 1 和输入 2 非的或运算

注解:

标准: IEC61131-3 定义的函数

4.2.157 P

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.158 POINTER

数据类型 pointer 是由 OpenPCS 所定义的作为 IEC61131-3 的补充。借助这种数据类型使得通过不同大小的数组调用函数或功能块成为可能。一个 pointer 必须象下面这样定义：

```
VAR
 IntVar: INT;
pInt: POINTER;
END_VAR
```

为了访问一个变量的地址，地址操作符 & 必须位于变量名的前面。

例如在 IL 中： LD&IntVar

在 ST 中： pInt := &IntVar;

4.2.159 POU

POU 是 Program Organisation Unit 缩写，意味用 IEC61131-3 一种编程语言来写的程序、函数、功能块。

4.2.160 Priority

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字，

4.2.161 PROGRAM

IEC61131-3 定义三种块类型: PROGRAM, FUNCTION 和 FUNCTION BLOCK。见 高级主题 下的 块类型 获得更多信息。

关键字 PROGRAM 和 END_PROGRAM 在 OpenPCS 中是不可见的，这一点在编辑器内含。

4.2.162 PT

这是个标准功能块(TOF)的一个形式参数的名称，并且定义为一个关键字。

4.2.163 PV

这是个标准功能块(CTD)的一个形式参数的名称，并且定义为一个关键字。

4.2.164 Q

这是个标准功能块(CTD)的一个形式参数的名称，并且定义为一个关键字。

4.2.165 Q1

这是个标准功能块的一个形式参数的名称，并且定义为一个关键字。

4.2.166 QD

这是个标准功能块(CTUD)的一个形式参数的名称，并且定义为一个关键字。

4.2.167 QU

这是个标准功能块(CTUD)的一个形式参数的名称，并且定义为一个关键字。

4.2.168 R

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.169 R(eset)

若 CR 值为 1 ，则重置操作数。如果这种预处理未碰到，操作数不变。CR 是不可以改变的。

注解:

IL 语言用关键字

IEC61131-3 定义的

4.2.170 R_EDGE

R_EDGE 用于说明布尔型输入的上升沿探测函数。这将引出一个 R_TRIG 类型的功能块的内在定义。

例子:

```
FUNCTION_BLOCK AND_EDGE

VAR_INPUT
        X : BOOL R_EDGE;
```

```
        Y : BOOL F_EDGE;
END_VAR

VAR_OUTPUT
        Z : BOOL ;
END_VAR

        Z := X AND Y ; (* ST language example *)

END_FUNCTION_BLOCK
```

4.2.171 R_TRIG

输入

CLK: 布尔型 输入的上升沿有效

输出

Q: 布尔型 输出; 指示 CLK 上升沿

功能块 R_TRIG 检测输入操作 CLK 的状态变化。若在处理循环中状态由 0 变被检测并通过输出 Q 为 1 来示意。仅当 CLK 状态在一个周期内的变化被检测到且产生了一个上升沿时，输出状态为 1 。

注解

标准: IEC61131-3 定义的功能块

4.2.172 R1

这是个标准功能块的形式参数的名称，并且定义为一个关键字。

4.2.173 READ_ONLY

这个关键字由 IEC61131-3 定义，用于访问路径的定义。OpenPCS 不支持访问路径，所以您不能在 OpenPCS 中使用这个关键字。

4.2.174 READ_WRITE

这个关键字由 IEC61131-3 定义，用于访问路径的定义。OpenPCS 不支持访问路径，所以您不能在 OpenPCS 中使用这个关键字。

4.2.175 REAL

见 基本数据类型

提示：

浮点数的（内部）二进制表示与硬件相关。有时甚至无法正确存储。所以比较数值的相等可能会出错。因此更好的办法是采用相对或绝对容错的区间值。

使用浮点数有时会丢失精度。比如浮点型是 32 位宽。24 位用来存储尾数，剩下的用来存储指数。16777216 是 2^24。所以无法表示 16777216+1。如果把浮点型用在诸如 x=x+1.0 的计数器里，计数器一旦到达 16777216 以后会停留在这个值上。

注解：

标准：IEC61131-3 定义的数据类型

4.2.176 REAL_TO_*

输入

实型

返回

转换的数据类型 *

把实型型数转换为另外一种数据类型 * 。

以下数据类型可以被转换：

BOOL

在±1,175494351e-38 之间的值被转换为 false，其它均为 true。

如：

1.1 -> true

-22.33 -> false

DINT, INT 和 SINT

四舍五入，因此小于 x.5 的值取整为较小的数，否则就取下一个较大的数。

例如：

0.3 -> 0

-0.6 -> -1

-1.5 -> -2

BYTE, DWORD, WORD, USINT, UNIT 和 UDINT

同正数的整形转换。

负值被转换为新的大小，位组合被处理为正数。

例如：

-1.6 -> 254 (USINT), 65534(UINT), 4294967294(UDINT)

(sint -2 的位组合为 1111 1110，它被转换为 254)

33.3 -> 33

STRING

为了转换为 string，使用了函数 Sprintf(str, %#g , value);

例如：

0.0 -> 0.000000

123.45678 -> 123.456

-12.345678 -> -12.3456

12345678.9 -> 1.23457e+007

0.000000123 -> 1.23000e-007

4.2.177 Release

这是个标准功能块(SEMA)的一个形式参数的名称，并且定义为一个关键字。

4.2.178 REPEAT

与其他循环类型相比，REPEAT 在执行循环后检查循环表达式。语法：

```
REPEAT
instructions;
UNTIL expression

END_REPEAT;
```

因此 REPEAT 循环至少要执行一次。

例如：

```
VAR
i : INT := -1;

END_VAR
REPEAT
i:=i-1;
UNTIL                    i                    <                    0
END_REPEAT;
(* now, i = -2 *)
```

尽管 i 在程序开始的时候满足循环条件，但 REPEAT 循环将执行一次。

注解：

ST 语言用关键字

IEC61131-3 定义的

4.2.179 REPLACE

输入

IN1: STRING 基本字符串，其中一部分需要被代替

IN2: STRING 新字符串

L: ANY_INT (根据支持) IN1 需要去掉的子字符串的长度。L < 0 无效

P: ANY_INT (根据支持) 插入字符串的起始位置。 P < 0 和 P > LEN(IN1)无效

返回

STRING 新组成的字符串。 IN1 如参数无效

REPLACE 函数用字符串 IN2 替换一个始于 P 长度为 L 的 IN1 的子字符串。

注解

标准: 这是 IEC61131-3 定义的函数

4.2.180 Resource

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字，

4.2.181 RESUME

Resume 功能块能够恢复任务的运行，当该任务被中断任务中断以进行错误处理。

输出：

Q: BOOL TRUE 如果成功

4.2.182 RET

RET 指令无条件的跳转返回到调用的 POU 上。－若这个 POU 是程序 POU,就返回到系统程序。 跳转返回时,调用的 POU 在中断点处继续.延时操作将被执行.

注解:

IL 语言用关键字

IEC61131-3 定义的

4.2.183 RETAIN

RETAIN 是声明变量的保留关键字, 在 VAR, VAR_GLOBAL 后是可选的。对保留的执行有赖于您的控制器。见声明部分

4.2.184 RETC

有条件返回

指令不带任何操作数.

若 CR 值为 1 , 则执行返回跳转到调用的 POU 上- 例如.若调用的 POU 是 program 类型，则返回到系统程序上. 若 CR 值为 0 , 则不跳转返回. 程序在跳转指令后的指令继续执行.

注解:

IL 语言用关键字

IEC61131-3 定义的

4.2.185 RETCN

有条件返回

指令不带任何操作数.

CR 布尔值决定是否跳转返回.

若 CR 值为 0 ,则执行返回跳转到调用的 POU 上- 例如.若调用 POU 是 program 类型，则返回到系统程序上. 若 CR 值为 1 , 则不跳转返回. 程序在跳转指令后的指令继续执行.

注解：

IL 语言用关键字

IEC61131-3 定义的

4.2.186 RETURN

RETURN 指令将跳离当前的 POU, 返回到调用当前 POU 处。注意使用函数时,函数值 (函数名变量)应该赋值. 若功能块的输出未赋给本地值,将输出预先定义的数据类型值.

例如:

```
IF a<b THEN

   RETURN;
 END_IF;
```

注解:

ST 语言用关键字

IEC61131-3 定义的

4.2.187 RIGHT

输入

IN: STRING 字符串

L: ANY_INT(根据支持) 要得到的字符数

返回

STRING 字符串 L 最右边的字符，IN 如果参数无效

RIGHT 函数把当前字符串的右边部分字节装载到工作寄存器. 输入操作数 L 定义了要进入的字符个数.

4.2.188 ROL

输入

IN: ANY_BIT 位 形式

N: UINT 左移的位数

返回

ANY_BIT IN, 循环左移 N 位

左移出的位循环放到右边

注解

标准: IEC61131-3 定义的函数

4.2.189 ROR

输入

IN: ANY_BIT 位 形式

N: UINT 右移位数

返回

ANY_BIT IN, 循环右移 N 位

右移出的位循环放到左边.

注解

标准: IEC61131-3 定义的函数

4.2.190 RS

输入

Set: bool　设置条件

Reset1: bool 重置条件

输出

Q1: bool　双稳态元件的输出状态

功能块 RS 动态切换一个数据元素（输出 Q1）为布尔值 1 或 0。 0 和 1 之间切换是根据布尔输入操作 SET 和 RESET1

处理前，输出 Q1 初始化为 0 ，若 SET 值为 1 ，功能块作用，输出 Q1 将置 1。此后， SET 改变不改变输出 Q1 。输入 RESET1 为 1 时总是置 Q1 为 0 。如，它可以重置输出。

若两个输入都为值 1 ，重置将起主导作用。也即 Q1 将重置。

注解

标准: IEC61131-3 定义的功能块

4.2.191 RTC

若 EN = 1，RTC 功能块将输出 CDT 设置为输入 PDT，否则 CDT 为非法。

输入：

IN: BOOL

PDT: DATE_AND_TIME 当前日期和时间

输出：

Q: BOOL　IN 的复制

CDT: DATE_AND_TIME　当前日期和时间，当 IN=1 时有效

注意：

标准化：这个功能块由 IEC61131-3 定义。

4.2.192 S(动作识别符)

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.193 S(et)

若 CR 值为 1，操作数置位，为 0 时操作数不变。 CR 是不改变的。

注解:

IL 语言用关键字

IEC61131-3 定义的

4.2.194 S1

这是个标准功能块的一个形式参数的名称，并且定义为一个关键字。

4.2.195 SD

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.196 SEL

这是个标准功能块的名称，由 IEC61131-3 定义，但 OpenPCS 不提供。查看一致性声明中的表 31 in。

4.2.197 SEMA

这是个标准功能块的名称，又 IEC61131-3 定义，但 OpenPCS 不提供，查看一致性声明的表 34 in。

4.2.198 SETSYSTEMDATEANDTIME

输入：

EN: BOOL

IDT: DATE_AND_TIME

输出：

ENO: BOOL

功能块 SetSystemDateAndTime 设置在 IDT 中的实际系统时间。

注意：

标准化：这个功能块并未在 IEC61131-3 中定义。

4.2.199 SFC

SFC 是 Sequential Function Chart 的缩写，是 IEC61131-3 一种编程语言。

4.2.200 SHL

输入

IN: ANY_BIT 位 形式

N: UINT 左移位数

返回

ANY_BIT IN, 左移 N 位

最右边的位用零补

注解

标准: IEC61131-3 定义的函数

4.2.201 SHR

输入

IN: ANY_BIT 位 形式

N: UINT 右移位数

返回

ANY_BIT IN, 右移 N 位

最左边的位用零补

注解

标准: IEC61131-3 定义的函数

4.2.202 SIN

输入

In: REAL

返回

REAL: 输入的正弦值

4.2.203 Single

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字

4.2.204 SINT

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.205 SL

这是个动作识别符，查看一致性声明中的表 45。由于 OpenPCS 只支持类型 N 的动作，您在 OpenPCS 中不必使用这个关键字。

4.2.206 SQRT

输入

In: REAL

返回

REAL: 输入的平方根值

SQRT 计算输入的平方根值

4.2.207 SR

输入

Set1: bool 设置条件

Reset: bool 重置条件

输出

Q1: bool 双稳态元件的输出状态

功能块 SR 静态转换一个数据元素（输出 Q1）为布尔值 1 或 0。

0 和 1 之间转换是根据布尔输入操作 SET1 和 RESET

处理开始时，输出 Q1 初始化为 0 ，若 SET1 值为 1 ，功能块作用，输出 Q1 将置 1。此后， SET1 改变不再改变输出 Q1 。输入 RESET 为 1 时总是置 Q1 为 0 ，如它可以复位。

若两个输入都为值 1 ，置位将起主导作用，也即 Q1 将置位。

注解

标准: IEC61131-3 定义的功能块

4.2.208 ST

CR 寄存器的值赋给操作数。这将覆盖操作数的值。操作数的类型必须和寄存器的数据元素的类型匹配。这个操作不改变 CR 寄存器。CR 寄存器的类型由赋值给变量的第一个值的数据类型决定。之后仅当数据类型相匹配时赋值.一个赋值后可以接另一个赋值。

注解:

IL 语言用关键字

IEC61131-3 定义的

4.2.209 ST (Structured Text)

ST 是 Structured Text 的缩写，是 IEC61131-3 的一种编程语言

4.2.210 STEP

这个关键字由 IEC61131-3 定义，用于程序设计语言 SFC 的文本表示。OpenPCS 不支持 SFC 的文本表示，所以您将不能输入这个关键字，当打印 SFC 时您将看到这个。

4.2.211 STN

CR 寄存器的负数赋给操作数。这将重写操作数的值。操作数的类型必须和寄存器的数据元素的类型匹配。这个操作不改变 CR 寄存器。一个 STN 必须跟在另一个 ST 或 STN 指令之后.

注解:

IL 语言用关键字

IEC61131-3 定义的

4.2.212 STRING

也见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.213 STRING_TO_*

输入

原数据类型 string

返回值

转换后的数据类型 *

该功能块将字符串类型的第一个值转换为*类型的相同值。

以下数据类型可以被转换:

BOOL

字符串 1 和 true 被转换为 true，其它为 false。

DINT, INT 和 SINT

字符串被从左向右读取，直至非法字符或字符串结束。

例如:

-1 -> -1

213hallo -> 213

23.5 ->23

BYTE, DWORD, WORD, USINT, UINT 和 UDINT

同正数的整形转换。

负值被转换为新的大小，位组合被处理为正数。

例如：

-1.6 -> 254 (USINT), 65534(UINT), 4294967294(UDINT)

(sint -2 的位组合为 1111 1110，它被转换为 254)

33.3 -> 33

REAL

同以上的转换。e 标识符有效。

例如：

-123.456 -> -123.456

0.23 -> 0.23

-1.2e-2 -> -0.012

4.2.214 STRUCT

STRUCT 是结构体数据类型的关键字, 见派生数据类型

一个变量包含两个成员：

VAR x1: STRUCT x2: INT; x3: BOOL; END_STRUCT; END_VAR

一个用户定义的类型：

TYPE x4: STRUCT x5: REAL; x6 : BOOL; END_TYPE

VAR x7: x4; END_VAR

4.2.215 SUB

输入

In1: ANY_NUM

In2: ANY_NUM

返回

ANY_NUM In1-In2 之差

两数相减.

注解:

标准: IEC61131-3 定义的运算

4.2.216 SUB (时间)

输入

In1: TIME 时间值

In2: TIME

返回

TIME 两个时间值之差

时间的差值

注解:

标准: IEC61131-3 定义的运算

TAN

输入

In: REAL

返回

REAL: 输入正切

4.2.217 Task

这个关键字由 IEC61131-3 定义，用于配置、资源和任务的文本定义。在 OpenPCS 中，这些通过属性对话框来定义和配置。您只有在打印这个配置定义时才能在 OpenPCS 中看到这个关键字，

4.2.218 THEN

见 IF

4.2.219 TIME

见 基本数据类型

怎样创建 TIME 常量也见常数

注解:

标准: IEC61131-3 定义的数据类型

4.2.220 TIME_OF_DAY

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

注意： 当前 OpenPCS 中不使用这个数据类型

4.2.221 TIME_TO_*

输入

In: TIME

返回

改变数据类型 *

TIME 值转换为别的数据类型 * ,可以是 e:

BOOL

BYTE

DINT

DWORD

INT

REAL

SINT

STRING

UDINT

UINT

USINT

WORD

注解:

1. 标准: IEC61131-3 定义的函数
2. 除了 TIME_TO_DINT，所有这列出的函数都只在梯形图编辑器中有效。

4.2.222 TO

See FOR

4.2.223 TOD

TOD 可以在指定一个文字常量的数据类型时作为 TIME_OF_DAY 的缩写。作为数据类型，TIME_OF_DAY 在 OpenPCS 中不执行，您将不能在 OpenPCS 中使用这个关键字。，

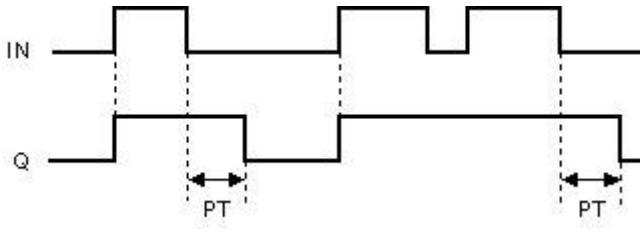
4.2.224 TOF

若输入状态 IN 为 1 ，它将无延时的转到输出 Q 。若有下降沿来临，一个计时函数将启动，延时时间有 PT 决定。

等到计时溢出时，操作数 Q 变为状态 0。若 PT 值在启动后改变，它会在下一个上升沿 IN 来临后才作用。

ET 包含当前时间值。若时间到， ET 将保持其值，直到 IN 值为 0 。若 IN 状态变为 1。 ET 值也变为 0。

若输入 IN 关，输出 Q 在经过指定的一段延时后也将关闭。



输入

IN: 开始条件

PT: time 初始时间值

输出

Q: bool 计时器状态

ET: time 当前时间值

注解

标准: IEC61131-3 定义的功能块

4.2.225 TON

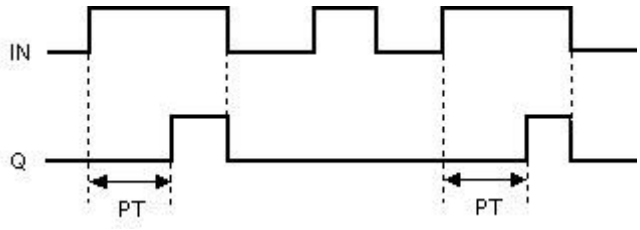
输入 IN 有一个上升沿将启动定时器 TON ， 延时时间为 PT 指定。

计时器运行时，输出 Q 置值 0 ，若时间到， Q 状态变为 1 ，且保持其值直到 IN 变为 0 。

若计时器运行后 PT 值改变，它仅当产生下一个上升沿 IN 时有效。

输出 ET 为当前定时器值。若时间到， ET 保持其值，保持时间和输入 IN 值为 1 一样长。若 IN 变为 0， ET 值将为 0。

若输入 IN 为开，输出 Q 将经过指定的延时时间后开。



输入

IN: 开始条件

PT: time 初始时间值

输出

Q: bool 计时器状态

ET: time 当前时间值

注解

标准: IEC61131-3 定义的功能块

4.2.226 TP

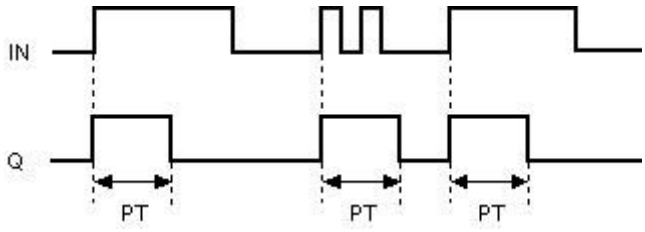
输入 IN 有一个上升沿将启动时间函数 TP ，运行时间由 PT 决定。

计时运行时，输出 Q 置值为 1。输入 IN 的任何变化将无效。

若启动后， PT 值改变将到下一个上升沿 IN 产生发生作用。

输出 ET 为当前时间值。若时间到 IN 值为 1 后， ET 将保持其值。

任何上升（下降）沿产生而计时器不运行，在输出端 Q 将产生一个指令时间脉冲。



输入

IN: 开始条件

PT: time 初始时间值

输出

Q: bool 计时器状态

ET: time 当前时间值

注解

标准: IEC61131-3 定义的功能块

4.2.227 Transition

这个关键字由 IEC61131-3 定义，用于程序设计语言 SFC 的文本表示。OpenPCS 不支持 SFC 的文本表示，所以您不能使用这个关键字。当打印 SFC 时您将看到这个。

4.2.228 TRUE

BOOL 类型的常数。

4.2.229 TRUNC

输入

In: REAL

返回

ANY_INT

返回实数的整数部分.

注解:

标准: IEC61131-3 定义的函数

4.2.230 TYPE

见 声明部分 和 派生数据类型

注解:

1. Pous 声明部分用关键字
2. IEC61131-3 定义的
3. TYPE .. END_TYPE 不应该在 VAR..END_VAR 块中嵌套使用，但可于申明部分的顶层，或是工程级的类型申明文件中。

4.2.231 UDINT

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.232 UINT

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.233 ULINT

这是个基本数据类型的名称，由 IEC61131-3 定义，但 OpenPCS 不支持。查看一致性声明中的表 10。

4.2.234 UNTIL

见 REPEAT

4.2.235 USINT

查看 基本数据类型

注意：：

标准： IEC61131-3 定义的数据类型。

4.2.236 VAR

见 声明部分

注解:

仅 POUs 申明部分用关键字

IEC61131-3 定义的数据类型

4.2.237 VAR_ACCESS

这个关键字由 IEC61131-3 定义，用于访问路径的定义。OpenPCS 不支持访问路径，所以您不能在 OpenPCS 中使用这个关键字。

4.2.238 VAR_INPUT

见 声明部分

注解:

仅 POUs 申明部分用关键字

IEC61131-3 定义的

4.2.239 VAR_OUTPUT

见 声明部分

注解:

仅 POUs 申明部分用关键字

IEC61131-3 定义的

4.2.240 VAR_IN_OUT

见 声明部分

注解:

仅 POUs 申明部分用关键字

IEC61131-3 定义的

4.2.241 VAR_GLOBAL

见 声明部分

注解:

仅 POUs 申明部分用关键字

IEC61131-3 定义的

4.2.242 VAR_EXTERNAL

见 声明部分

注解:

仅 POUs 声明部分用关键字

IEC61131-3 定义的

4.2.243 VARINFO

VARINFO 定义为

VARINFO : Struct

TYP : UINT;

SIZE : UINT;

PROG : UINT;

SEG : UINT;

OFFSET:UINT;

BIT: UINT;

SCOPE: UINT;

end_struct;

4.2.244 WHILE

当所给表达式为 ture 时，WHILE 循环执行循环体。

语法:

```
WHILE expression DO

    instructions;
END_WHILE;
```

在进入循环前判断 **WHILE** 后面的表达式。如果它为 ture ，执行循环体。当表达式的值是 false 时，终止循环。

例如

```
    VAR
i : INT := 3;

     END_VAR
     WHILE i > 0 DO

       i:=i-1;

   END_WHILE;
```

最初 i 等于 3，3 大于 0，所以 WHILE 后面的表达式为 ture，执行循环体。将 i 的值减小到 2，2 仍然大于 0，再次执行循环体。多次执行之后，循环体 i 减少到 0。在下一次判断中，WHILE 后面的表达式为 false，此后不在执行循环体。

注解:

仅 ST 语言用关键字

IEC61131-3 定义的

4.2.245 WITH

这个关键由 IEC61131-3 定义，用于配置的文本定义。在 OpenPCS 中，这些都使用属性对话框来定义和配置。您只有在打印一个配置定义时才能在 OpenPCS 中看到这个关键字。

4.2.246 WORD

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.247 WSTRING

见 基本数据类型

注解:

标准: IEC61131-3 定义的数据类型

4.2.248 XOR

输入

IN1: ANY_BIT 输入 1

IN2: ANY_BIT 输入 2

返回

ANY_BIT 逻辑的, 输入 1 和 输入 2 的异或值

注解

标准: IEC61131-3 定义的函数

4.2.249 XORN

输入

IN1: ANY_BIT Input 1

IN2: ANY_BIT Input 2

返回

ANY_BIT 逻辑的，将 Input1 和 Input 2 的反进行位运算。

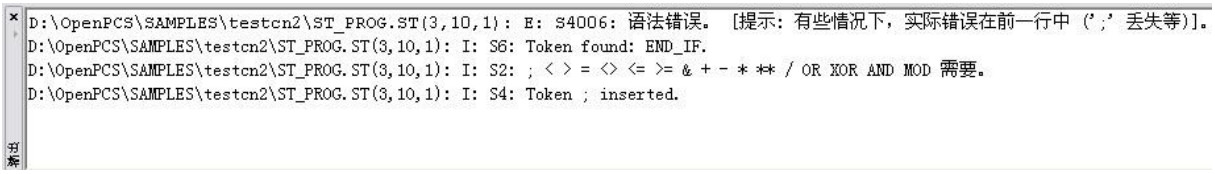
注解：

标准：这个函数是由 IEC61131-3 定义的。

4.3 错误和警告

4.3.1 怎么读错误信息

在输出窗口您可以发现编译器的所有出错信息。



每个错误信息行符合下面格式：

1. 导致错误信息的文件名，包括源代码的路径。
2. 三个数，第一个数指出错误发生的部分(2 用于 声明 ， 3 用于 指令)，第二个指出行，最后一个指出列（以前提起过）。
3. 大写字母给出信息类型：

   字母 表示

   I 信息

   E 错误

   W 警告

   F 致命错误
4. 错误数代号可以帮助您在文档中找到错误的详细描述。

5. 错误的简单描述。

4.3.2 常规错误

4.3.2.1 G10001

警告 G10001：文件"文件名"不一致，请勿用。

文件不一致。可能的原因时文件名和包含该文件的 POU 名不一致。这通常是由在 OpenPCS 外重命名文件而引起的。POU 应该用 OpenPCS 的功能 ***文件->重命名***来重新命名。

4.3.3 语法错误

4.3.3.1 S1000

嵌套的注释是不允许的。

您正使用的是一个与 IEC 61131-3 兼容的版本，在这个版本中不允许嵌套的注释。

4.3.3.2 S1001

无效的字符。

用了一个不支持的字符。参看表 1: 字符集特性

4.3.3.3 S1002

注释中找到文件结束。

一个打开的注释关闭之前到了文件末尾。请在调用语法检查之前关闭注释。

4.3.3.4 S1003

保留的关键字。

标识符用了保留的关键字。

4.3.3.5 S1004

无效的小时值。

一个 TIME_OF_DAY 或 DATE_AND_TIME 的小时单位的数值必须是在[0, 23] 范围的整数。

4.3.3.6 S1005

无效的分钟值。

一个 TIME_OF_DAY 或 DATE_AND_TIME 的分钟单位的数值必须是在[0, 59] 范围的整数。

4.3.3.7 S1006

无效的秒值。

一个 TIME_OF_DAY 或 DATE_AND_TIME 的秒单位的数值必须是在[0, 60] 范围的一个固定的整数。

4.3.3.8 S1008

无效的月范围。

一个 TIME_OF_DAY 或 DATE_AND_TIME 月的单位的数值必须是在[1, 12] 范围的整数。

4.3.3.9 S1009

无效的数据范围。

一个 TIME_OF_DAY 或 DATE_AND_TIME 中月的天数数值必须是在[0, 31] 范围的整数。例如，如果一个月的天数少于 31，则这个月的最大天数是天数说明的最大有效值。

4.3.3.10 S1010

指数太大。

实数的指数值必须是在[-37, 38]的一个整数，长实数的指数值必须是在[-307, 308]的一个整数。

4.3.3.11 S1011

不正确的直接地址。

一个直接表示变量的分级地址的位置数字值是硬件相关的整数，但是不能超过 4294967295。请查看您的硬件资料，确定分级地址的每个域的最大值。

4.3.3.12 S1012

无效的日期输入。

一个 TIME 的天数单位的数值必须是在[0, 255] 范围的一个固定点数字。

4.3.3.13 S1013

无效的小时输入。

一个 TIME 的小时的数值必须是在［0, 24） 范围的一个固定整数，这个小时数不是一个重要的连续时间变量，反之，这个小时数可以溢出。

例如：

T#25h_15m 是允许的。

T#1d_25h_15m 是不允许的，正确的表示法为：T#2d_1h_15m。

4.3.3.14 S1014

无效的分钟输入。

一个 TIME 的分钟单位的数值必须是在［0, 60） 范围的一个固定整数，这个分钟数不是一个重要的连续时间变量，反之，这个分钟数可以溢出。

例如：

T#75m 是允许的。

T#5h_75m 是不允许的，正确的表示法为：T#6h_15m.

4.3.3.15 S1015

无效的秒输入。

一个 TIME 的秒单位的数值必须是在［0, 60］ 范围的一个固定整数，这个秒数不是一个重要的连续时间变量，反之，这个秒数可以溢出。

例如：

T#75m 是允许的。

T#5m_75s 是不允许的，正确的表示法为：T#6h_15s.

4.3.3.16 S1016

无效的毫秒输入。

一个 TIME 的微秒单位的数值必须是在［0, 1000）范围的一个固定整数，这个微秒数不是一个重要的连续时间变量，反之，如果这个微秒数是唯一的一个 TIME 单元，则这个微秒数可以溢出。

例如：

T#1200s 是允许的

T#1s_1200ms 不允许的，正确的表示法为：T#2s_200ms.

4.3.3.17 S1017

直接地址太复杂。

一个立即数的分级地址域的个数随硬件而定的，但不能超过 8。请参考您的用户手册，确定分级地址的深度。

4.3.3.18 S1018

整型常数太大/小。

一个常数的值必须在它的数据类型所能表示的范围之内。一个整型常数类型依赖于指定的常数变量类型，但是不能超过 LINT/ULINT (8 字节整数/无符号整数)常数所能表示的范围。

4.3.3.19 S1019

整型常数太大/小 (不能适合 32 位)。

给定的常数值超过了 DINT/UDINT 类型范围。

4.3.3.20 S1020

数字值太大/小。

给定的常数值必须在它的数据类型所能表示的范围之内。有符号整型常数依赖于指定的常数类型，但不能超过 LINT (8 字节整数)常量范围。

4.3.3.21 S1021

处理一个浮点型数学函数时出错。

4.3.3.22 S1022

无效的字符串常数。

给定的字符串常量包含一个无效字符。一个字符串是零序列或用单引号( )括起来的多字符序列。有效字符是除 $ 之外的任何能打印的字符。由美元符和跟在它后面的两个十六进制数组成的三个字符联合体，将解释为像显示在字符串文字特性中的十六进制表示的 8 位字符代码那样。

而且，由美元符开头的两个字符当出现在字符串中时，将由两个字符联合字符串表格来解释。

4.3.3.23 S1023

无效的号码 (即，数字常数)。

给定的数字常数包含一个无效的字符。参看数字文字 关于数字文字有效性的例子。

4.3.3.24 S1024

无效的常数。

给定的常数包含无效字符。

表示有效常数的列表，参看表 53：IL 语言的功能块特性。

4.3.3.25 S1025

无效的直接地址。

一个立即表示数的变量包含无效字符。

一个变量的直接表示法是由百分号%，一个位置前缀，一个可选的大小前缀和一个或多个被周期性(.)分开的整数组成。

生产厂商会详细说明在输入或输出寄存器中，直接表示变量和物理或逻辑地址之间的通信。当一个直接的表示扩展成由几个周期性分离的附加的整数域时，它将解释为一个分层的物理或逻辑地址，最左边的域部分表示层次最高，最右边的域部分表示层次最低。例如，变量%IW2.5.7.1 表示在可编程控制器上第二个 I/O 总线 的第五个架 上的第七个模块的第一 通道 （字）。直接表示的变量只能用在程序中。最大的地址层次数依赖于硬件，但不能超过 8。请参看您的硬件资料，确定最大的地址层次数。

4.3.3.26 S1026

无效的指示符 (名称, 变量, 参数,...)。

一个标识符包含一个或多个无效字符。

标识符是由字母，数字和下划线组成的或由下划线开头的。字母可以是大写或小写的。由多个下划线起始的或有多重嵌套的下划线是不允许的。

标识符不允许包含空格。

4.3.3.27 S1027

文件头中找到文件结束。

读文件头时出错。用一个文本编辑器打开这个文件，并删除在 PROGRAM，FUNCTION 或 FUNCTION_BLOCK 关键字前面的所有行，就能修改这个错误。如果这个错误经常发生，请联系您的生产厂家。

4.3.3.28 S1028

标识符过长 (> 64 字符).

一个标识符的长度超过了能支持的最大长度。标识符的最大长度是 64 个字符。

4.3.3.29 S1029

这个字 (标识符，常数，字符串，注释) 太长(> 1024 字符)。

一个记号（标识符，常量，字符串，注释）超过了 1024 个字符。这里最大只支持 1024 个字符。

4.3.3.30 S1030

太多指示符。

标识符的个数超过了最大值，能支持的最大标识符个数是 65535。

4.3.3.31 S1031

不允许的 EN 用法。只允许在输入部分为布尔变量作为指示符。

一个带有 EN 名字的变量在错误的变量部分定义了或定义错了数据类型。

这个 EN （enable）名字是保留给布尔型输入变量的。

当函数或功能块被调用时，如果 EN 的值是 FALSE，则函数或功能块定义的操作不能执行。如果布尔型输出参数 ENO 也被定义了，则 ENO 的值会重设为 FALSE。

当函数或功能块被调用时，如果 EN 的值是 TRUE，则函数或功能块定义的操作能执行。如果布尔型输出参数 ENO 也被定义了，则这些操作包括给布尔型输出参数 ENO 设定一个布尔值。

4.3.3.32 S1032

不允许的 ENO 用法。只允许在输出部分为布尔变量作为指示符。

一个带有 ENO 名的变量在错误的变量部分定义了或定义错了数据类型。

这个 ENO （Enable Out）名字是保留给布尔型输出变量的。 ENO 变量需要布尔型输入变量 EN 。

当函数或功能块被调用时，如果 EN 的值是 FALSE，则函数或功能块定义的操作不能执行，且输出参数 ENO 的值会重置为 FALSE。

当函数或功能块被调用时，如果 EN 的值是 TRUE，则函数或功能块定义的操作能执行。这些操作包括给布尔型输出参数 ENO 设定一个布尔值。

4.3.3.33 S3000

功能块未声明。

一个调用发现一个未知的功能块实例。

一个功能块实例在使用前必须声明。

提示：

确定需要的功能块实例已经在变量声明部分声明了。

确定需要的功能块实例的名字拼写正确。

4.3.3.34 S3001

函数未表示。

一个调用发现一个未知的函数。

一个函数在使用前必须声明。在函数使用之前，必须在声明部分或原型中指定函数的参数。

提示：

确定包含这个函数的声明或原型的文件属于这个工程或者这个函数是固件的一部分。

确定函数的名字拼写正确。

4.3.3.35 S3002

不正确的参数。

在功能块的形参列表中没有找到所需要的参数。

提示：

确定参数的名字拼写正确。

确定所定义的功能块的参数列表包含指定任务的参数名。

4.3.3.36 S3003

跳转标签未表示。

一个 JMP 指令发现一个未知的标号。

在程序指令部分，要用的标号必须已经定义了。

提示：

确定在同一程序单元中，标号已经定义了。

确定标号的名字拼写正确。

4.3.3.37 S3004

一个变量/名称的多个分配。

给定的标识符被定义了多次。

提示：

确定在同一程序单元中，这个标识符没有被定义两次以上。

确定这个标识符在用户自定义类型和全局类型声明中没被使用，或者没有作为一个函数、功能块和程序的名字。

4.3.3.38 S3005

这不是一个功能块实例。

发现一个带有调用状态名称的变量，但不是一个功能块中实例。

提示：

确定这个标识符拼写正确。
确定这个要调用的功能块实例已经被定义了，并且也在这个程序单元范围内或在全局范围内。

4.3.3.39 S3006

这是一个变量或功能块实例。

在访问一个结构成员或功能块变量时，指定的变量标识符不是一个功能块或不是一个结构。

提示：

确定这个标识符拼写正确。
确定给定的主变量名是一个结构或一个功能块名。

4.3.3.40 S3007

这不是一个函数-POU。

要作为一个函数名的标识符已经被定义，但被定义的不是一个函数名。

提示：

确定这个标识符拼写正确。

确定这个标识符名是一个函数名，而不是一个功能块名。

确定在指定的位置，所需要的是一个函数而不是一个功能块实例。

4.3.3.41 S3008

无结构元素或块参数。

在访问结构成员或功能块变量时，指定的成员变量标识符不是一个功能块或结构实例的参数。

提示：

确定这个标识符拼写正确。

确定这个功能块或结构实例是正确的。

如果要访问的变量是一个功能块实例，必须确定这个功能块有一个所给定的标识符名的参数。

如果要访问的变量是一个结构实例，必须确定这个结构有一个所给定的标识符名的成员。

4.3.3.42 S3009

无跳转标签。

在给定的位置找到用在 JMP/JMPC/JMPCN 指令中的标识符，但不是一个标号名。

提示：

确定这个标识符拼写正确。
确定用在 JMP/JMPC/JMPCN 指令后面的标识符是一个标号名。

4.3.3.43 S3010

类型不存在。

需要一个类型或功能块名，在当前范围内找到的这个标识符即不是一个类型名也不是一个功能块实例名。

提示：

检查这个名字是否拼写正确。

确定这个标识符不是一个变量名(例如：一个功能块名)。

4.3.3.44 S3011

不正确的变量/名称。

需要一个变量或者功能块实例。在当前范围中发现一个标示符，但既不是变量也不是功能块实例。

提示：

检查这个名字是否拼写正确。

确定这个标识符不是一个类型名(例如：一个功能块名)。

4.3.3.45 S3012

需要变量名或常数。

这个错误发生在,一个不是变量名或数字常数的标识符用在了一个需要变量名或常数的地方。

例子：：

```
TYPE
Colours : (red, yellow, blue) := red;
END_TYPE
VAR
Colour : Colours := Colours; (* 错误：需要一个数字常数， EnumType 是一个类型名 *)
END_VAR

LD Colours  (* 错误：需要常数或常量， EnumType 是一个类型名 *)
ST Colour
```

4.3.3.46 S3014

需要数组的数据类型。

操作指令和操作数类型不兼容。一个 ANYNUM 数据类型的操作数是需要的。

4.3.3.47 S3016

需要位数据类型。

操作指令和操作数类型不兼容。一个 ANYBIT 数据类型的操作数是需要的。

4.3.3.48 S3017

需要逻辑值。

操作指令和操作数类型不兼容。一个 BOOL 数据类型的操作数是需要的。

4.3.3.49 S3018

需要数组的数据类型。

不允许的操作数类型。一个 ANYNUM 数据类型的操作数是需要的。

4.3.3.50 S3019

操作数类型不兼容。

操作数和返回结果的数据类型不兼容。

4.3.3.51 S3020

操作数类型不兼容。

这个错误发生在，一个不允许的时间和日期数据类型的联合使用作为一个 SUB 操作的输入参数。这个操作允许的输入和输出联合使用的数据类型请参看在 IEC 1131-3 一致性声明中的表 30-时间数据类型函数。

例子：

```
VAR
    TimeVar : TIME;
    DateVar : DATE;
END_VAR

 LD DateVar
SUB TimeVar (* 错误：SUB 不是定义给这个输入参数的联合体 *)
```

ST DateVar

(*...*)

4.3.3.52 S3022

无效的操作数类型对于此操作。

在指定的位置，这个操作的操作数类型是无效的。所需要的操作数是一个 TIME 或 ANYNUM 类型。

4.3.3.53 S3023

无效的操作数类型对于此操作。

在指定的位置，这个操作的操作数类型是无效的。所需要的操作数是一个 TIME，TIME_OF_DAY，DATE_AND_TIME 或 ANYNUM 类型。

4.3.3.54 S3024

无效的操作数类型对于此操作。

在指定的位置，这个操作的操作数类型是无效的。所需要的操作数是一个 ANYBIT 类型。

4.3.3.55 S3025

需要逻辑结果。

返回的结果类型不匹配，返回的应该是一个 BOOL 类型。

4.3.3.56 S3026

未声明的指示符。

这个错误发生在在给定位置的标识符在编辑 POU 的有效范围内没有定义。

例子：

```
TYPE
  Colours : (red, yellow, blue) := red;
END_TYPE
```

```
VAR
  Colour : Colours := green; (* 错误：green 没有被当作一个数字常数定义 *)
END_VAR

LD IntVar (* 错误：IntVar 变量没有被声明 *)
ADD 5
ST IntVar
(*...*)
```

4.3.3.57 S3028

当前结果的数据类型的比较未定义。

在给定位置的比较关系没有定义当前返回类型，即实际参数类型与第一个形参类型不匹配，要了解更多的信息，请参看在 IEC 1131-3 一致性声明中的表 28-标准比较函数。

例子：

```
TYPE
    Day_of_Week : STRUCT
        Name : String;
        DayNo : INT(1..7);
    END_STRUCT;
END_TYPE
VAR
    DayVar1 : Day_of_Week;
    DayVar2 : Day_of_Week;
    BoolVar : BOOL;
END_VAR
(*...*)
LD DayVar1
```

GT DayVar2 (* 错误：结构变量的比较关系是不允许的 *)

ST boolVar

(*...*)

4.3.3.58 S3030

这个类型的比较未定义

在给定位置的操作数类型不允许用来作比较。即实际参数类型与形参类型不匹配。要了解更多的信息，请参看在 IEC 1131-3 一致性声明中的表 28-标准比较函数。

例子：

```
TYPE
    Day_of_Week : STRUCT
        Name : String;
        DayNo : INT(1..7);
    END_STRUCT;
END_TYPE
VAR
    DayVar1 : Day_of_Week;
    DayVar2 : Day_of_Week;
    BoolVar : BOOL;
END_VAR
(*...*)
LD DayVar1
GT DayVar2 (* 错误：结构变量的比较关系是不允许的 *)
ST boolVar
(*...*)
```

4.3.3.59 S3032

自参照(即递归) 声明不允许。

发现递归调用。一个函数不能直接或间接调用它自己（例如，调用另外一个函数，但这个函数已经调用了它本身）。

功能块和程序不能直接或间接的声明它们自己（例如，调用另外一个功能块实例，但这个功能块已经声明了它自己的一个实例）。

4.3.3.60 S3033

需要 TIME 类型的操作数。

需要一个 TIME 类型的常数或变量，但在给定位置的操作数是一个别的类型。

例子：

```
VAR

StartTime : TIME_OF_DAY;

StopTime : TIME_OF_DAY;

RunTime : TIME := T#10s;

END_VAR

(*...*)

 LD StartTime
ADD 10000 (* 错误：操作数必须是一个 TIME 类型 *)

ST StopTime

(*...*)

LD StartTime

ADD RunTime (* 正确 *)

ST Stop Time

(*...*)
```

4.3.3.61 S3034

变量的字符串太长。

一个字符串文字指定给了一个字符串变量，但这个字符串文字不适合字符串变量。例如，字符串的长度超过了分配给字符串变量的长度。

4.3.3.62 S3035

此函数不允许的操作数类型! 需要数字或日期或时间操作数。

在给定位置的操作没有定义给当前的返回类型（即第一个实际参数）。

例子：

```
VAR
  BitMake: WORD;
END_VAR
(*...*)
LD BitMask (* 错误：操作数必须是 TIME, ANY_DATE 或 ANY_NUM 类型 *)
SUB 3
ST BitMask
(*...*)
```

4.3.3.63 S3036

整型常数超出范围。

在给定位置的整数常数超过了指定的数据类型的范围。

例子：

```
VAR
        Range1 : UINT(-1..1000); (* 错误：符号不配。UINT 类型的值一定不能是负数 *)
Range2 : INT(-1..36000); (* 错误：溢出。上限值超过了 INT 类型的最大有效值 *)
END_VAR
```

4.3.3.64 S3037

下限范围必须大于上限。

在指定子区域上限值时，上界值小于它的下限值。指定一个整数类型的子区域时，区域内的取值界于或包括上下限，上限值必须大于下限值。

4.3.3.65 S3038

初始化超出子范围的编界(数据类型为子范围)。

一个子类型的变量初始化时，超过了这个子类型的范围。声明一个子类型时，这个类型的每一个元素取值只能界于指定的上下界之间，包括上下界。

4.3.3.66 S3039

索引超出边界。

访问一个数组类型变量时，索引值超过了指定类型或变量的范围。

4.3.3.67 S3040

无效的数据类型。需要 ANY_NUM。

在给定位置的操作没有定义当前返回类型（即第一个实际参数）。

例子：

```
VAR
  BitMake: WORD;
END_VAR
(*...*)
LD BitMask (* 错误：操作数必须是 TIME, ANY_DATE 或 ANY_NUM 类型 *)
NEG
ST BitMask
(*...*)
```

4.3.3.68 S3041

不允许为 EN/ENO 类型。必须为布尔类型。不得为 RETAIN。

一个带有 EN 名的输入变量或带有 ENO 名的输出变量被声明成了一个不允许的类型或带有 RETAIN 标识符。EN 标识符是保留给布尔类型的输入变量的。 ENO 标识符是保留给布尔类型的输出变量的。这个变量一定不能带有 RETAIN 标识符。

4.3.3.69 S3042

丢失 EN。ENO 仅允许与 EN 结合使用。

一个带有 ENO 名的输出变量已被定义，但没有找到带有 EN 名的输入变量。这个 ENO 输出变量只能和 EN 输入变量联合使用。

4.3.3.70 S3044

数据丢失。你需要装载或表达式。

当前返回结果没有定义。在当前位置之前必须要一个 LC 指令或者必须要一个表达式。这个错误发生在象语法错误 S 5010 那样的结果错误。请把这个指令移到圆括弧之外。

4.3.3.71 S3046

类型名称不能被用作实例名称。 %1 已经定义为类型!

一个类型名或 POU 名在声明部分被当作一个变量名使用了。在工程层次上，POU 和类型定义在整个工程范围内是知道的，并且它们的名字不能当作局部变量的标识符使用。

例子：

```
FUNCTION Power

(* 功能块声明 *)

(* 指令 *)

END_FUNCTION


PROGRAM main

VAR

    Power : REAL; (* 错误：Power 不允许当作一个变量名作用，因为它已被当作一个函数名使用了 *)

(*...*)

END_VAR
```

(* 代码 *)

END_PROGRAM

4.3.3.72 S3047

数函参数必须按照函数原型中定义的顺序指定。颠倒的参数顺序会导致错误的代码，即使参数名被正确的指定。

当一个功能块在 ST 中被调用，ST 编译器直接把给定的调用参数列表翻译成 IL 代码，因为它不了解功能块的声明。因此，指定的顺序须与功能块声明的输入输出顺序相符。

例如：

```
FUNCTION_BLOCK Example
VAR_INPUT
        In1 : int;
        In2 : int;
END_VAR
FUNCITON_BLOCK_END

Program:
VAR
        Instance : Example;
        Local1 : int;
        Local2 : int;
END_VAR

(* 正确：参数顺序与声明的顺序相符 *)
Example(In1 := Local1, In2 := Local2);

(* 错误：参数顺序与声明的顺序不符 *)
```

```
Example(In2 := Local2, In1 := Local1);
```

4.3.3.73 S3048

可能的字符串截断已被指派。

发生这个警告是因为要赋值的目标字符串的总长度小于源字符串。这个检查在编译时完成基于两个字符串声明的大小。

例如：

```
VAR
        strDestination   : string[10];
        strSource        : string[40];
END_VAR

strDestination := strSource;
```

4.3.3.74 S3049

ST 语法错误 (双击以获得 ST 错误)

编译 ST 时监测到 POU 包含一个语法错误。双击这个错误可以跳入错误源头并显示语法错误信息。

4.3.3.75 S3050

数组范围不匹配

警告如果对两个不同区间但大小和类型相同的数组进行赋值。编译器允许对相同大小和类型的数组进行赋值。例如：对两个整型数组范围 [0□9]和[1□10]进行赋值是允许的，不过会引起这个警告。

4.3.3.76 S4000

"AT%": 同时声明多个直接变量是无效的。

标识符列表在本地变量声明中已经使用了。直接表示的变量只能分配给一个标识符。

例如：

下面的定义是不允许的：

```
VAR

dirVar1, dirVar2, dirVar3 : at%I0.0;

END_VAR
```

4.3.3.77 S4001

太多变量(指示符)。最大为 60 个指示符。

在标识符变量声明列表中，标识符太多了。支持的最大的数目是 60。

4.3.3.78 S4003

数组太大。

在定义数组时，一个维的元素个数超过了 OpenPCS 能支持的最大数。 这个最大元素数依赖于能支持的索引范围。

4.3.3.79 S4005

上限必须大于或等于下限。

在指定位置的数组声明中，同一维的上限值小于它的下限值。同一维的上限值必须大于或等于指定的下限值。

4.3.3.80 S4006

语法错误[提示：在某些情况下，真正的错误发生在前一行 (漏了一个 ； 等。)]。

4.3.3.81 S4007

自参照(即递归) 声明无效。

发现递归调用。一个函数不能直接或间接的递归调用它本身（例如，调用另外一个函数，但这个函数在调用层次上已调用了它本身）。功能块和程序不能直接或间接声明它们自己的实例（例如，调用另外一个功能块的实例，但这个功能块在调用层次上已经调用了它本身）。

4.3.3.82 S4008

太多"RETAIN" 或 "CONSTANT"属性。你可以只使用一个。

在一个变量声明部分，使用了太多的限定词。

4.3.3.83 S4009

一个结构(STRUCTure)必须包含至少一个结构元素(变量声明)。

定义了一个空的结构，这是不允许的。一个结构必须至少包含变量元素。

例如：

不允许：

```
TYPE
  Mystruct : struct end_struct;
END_TYPE
```

允许：

```
TYPE
  Mystruct : STRUCT
    M1 : int;
  END_STRUCT
END_TYPE
```

4.3.3.84 S4010

你不能同时声明多个类型。

在指定声明类型的地方包含一个标识符列表，这是不允许。

请声明另外的新类型。

例如：

不允许：

```
TYPE
```

MyInt1, MyInt2, MyInt3 : int;

END_TYPE

允许：

TYPE

MyInt1 : int;

MyInt2 : int;

MyInt3 : int;

END_TYPE

4.3.3.85 S4011

只在程序中 VAR- 和 VAR_GLOBAL-段有效.

不支持在 POU 或变量声明部分声明的直接表示的变量。在 PROGRAMs 的 VAR-或 VAR_GLOBAL 的变量声明部分只支持本地变量。

4.3.3.86 S4012

仅在程序, 功能块和函数中有效。

在一个单元中，发现一个变量声明(VAR <声明> END_VAR)部分不支持。在程序，函数和功能块中，允许声明局部变量。

4.3.3.87 S4013

仅在程序, 功能块和函数中有效。

在一个不支持输入变量的 POU 中，发现一个输入变量声明(VAR_INPUT <声明> END_VAR)部分。

4.3.3.88 S4014

仅在程序和功能块中有效。

在一个不支持输入输出变量的 POU 中，发现一个输入输出变量声明(VAR_IN_OUT <声明> END_VAR)部分。

4.3.3.89 S4015

仅在程序和功能块中有效。

在一个不支持输出变量的 POU 中，发现一个输出变量声明(VAR_ OUTPUT <声明> END_VAR)部分。

4.3.3.90 S4016

仅在程序和功能块中有效。

在一个不支持外部（external）变量的 POU 中，发现一个外部（external）变量声明（VAR_ EXTERNAL <声明> END_VAR）部分。在程序和功能块中支持外部变量声明。

4.3.3.91 S4017

仅在程序中有效。

在一个不支持全局（global）变量的 POU 中，发现一个全局（global）变量声明（VAR_ GLOBAL<声明> END_VAR）部分。全局变量只允许在程序中声明。

4.3.3.92 S4018

CONSTANT 仅在 VAR 和 VAR_GLOBAL 区域中有效。

在一个不支持 CONSTANT 限定词的变量声明部分，使用了 CONSTANT 限定词。

4.3.3.93 S4019

RETAIN 仅在程序或者功能块的 VAR-，VAR_OUTPUT-， 或者 VAR_GLOBAL-区域中有效。

在一个不支持 RETAIN 限定词的变量声明部分，使用了 RETAIN 限定词。

4.3.3.94 S4020

仅在程序或者功能块的 VAR_INPUT 区域中的未初始化的 BOOL 变量有效。

在一个不支持边缘限定词的 POU 或变量声明部分中，使用了边缘限定词。

4.3.3.95 S4021

仅在 VAR_INPUT, VAR_OUTPUT, 和 VAR_IN_OUT-部分有效。

在一个不支持 ADDRESS 限定词的 POU 或变量声明部分中，使用了 ADDRESS 限定词。

4.3.3.96 S4022

仅在功能块或函数中并在 VAR..END_VAR 之间且没有 CONSTANT/RETAIN 修饰符情况下有效。

在一个不支持 ATTRIBUTES 限定词的 POU 或变量声明部分中，使用了 ATTRIBUTES 限定词。这个限定词只允许在没有 CONSTANT 或 RETAIN 限定词的函数和功能块中的变量部分（VARSections）。

注意：

关键字 ATTRIBUTES 只在 OpenPCS 的定制版本中才被支持，用来在 IEC61131-3 的扩展中定义变量的附加属性。在标准 OpenPCS 中您看不到这些信息。

4.3.3.97 S4023

仅在 TYPE..END_TYPE-部分有效。

在一个不支持结构的变量声明部分，发现了一个结构声明。结构声明只在 TYPE 声明部分被支持。

4.3.3.98 S4024

在 VAR_EXTERNAL 区域中无效。

在一个 EXTERNAL 变量声明部分，一个变量带了一个初始值。这是不允许的。请在各自的 GLOBAL 变量声明部分指定它们的初始值。

例如：

```
VAR_EXTERNAL
   A : INT := 5; // not allowed
END_VAR

VAR_EXTERNAL
   A : INT;
END_VAR
```

VAR_GLOBAL

A : INT := 5

END_VAR

4.3.3.99 S4033

多个初始化。

一个结构成员变量初始化多次。这个错误发生在当一个外部结构和一个结构的每一个元素同时初始化的时候。

例如：

下面的初始化是不允许的：

TYPE

StructType : Struct

Member1 : int := 5;

Member2 : bool;

END_STRUCT := (Member1 := 4, Member2 := true);

END_TYPE

用下面的一个初始化代替：

TYPE

StructType : Struct

Member1 : int ;

Member2 : bool;

END_STRUCT := (Member1 := 4, Member2 := true);

END_TYPE

或

TYPE

```
StructType : Struct
        Member1 : int := 5;
        Member2 : bool := true;
END_STRUCT;
END_TYPE
```

4.3.3.100 S4034

无效的 POU 名称。

这个错误发生在当使用了一个关键值作为一个 POU 的名字或没有名字被定义时。

4.3.3.101 S4035

无效的函数类型。

函数类型必须是预先定义的类型或是已确定的类型。这个错误一般发生在使用了一个保留关键字（一个 IEC61131-3 专用的字符串或者数字）作为一个函数类型或者还没有定义函数类型。

4.3.3.102 S4036

函数至少需要一个输入参数 VAR_INPUT。

定义了一个没有输入参数的函数。根据 IEC61131-3 标准，一个函数必须带一个输入参数。

4.3.3.103 S5000

错误的参数类型。

一个函数或功能块实例的实际参数类型和已指定的形参类型不匹配。

4.3.3.104 S5001

需要数组。这不是一个数组。

索引访问一个变量，但这个变量不是数组。

例如：

```
PROGRAM

  VAR
x : INT;

y : INT;

END_VAR



LD x[3] (* 如果这个变量不是数组这是不允许的*)

ST y



END_VAR
```

4.3.3.105 S5002

这个功能块被 CAL 调用当 EN=TRUE。CALC/CALCN 都无效。

一个带有 EN 输入参数的功能块实例被 CALC/CALCN 调用，这是不允许的，要用 CAL 指令代替。一个带有 EN 输入参数的功能块代码被调用，如果这个参数值是 TRUE。

4.3.3.106 S5003

功能块实例不能为 "CONSTANT"。

在带有 CONSTANT 属性的变量部分，定义了一个功能块实例，这是不允许的。请删除这个属性或把实例声明移到另一个变量部分，这个变量部分不带 CONSTANT 属性。

4.3.3.107 S5004

功能块实例在"FUNCTION"-POUs, 结构, 和数组中无效。

一个功能块实例被定义在一个函数部分里或被当作一个 STRUCT 或 ARRAY 类型。IEC61131-3 不允许在函数中声明功能块实例。OpenPcs 不支持功能块实例作为 STRUCT 和 ARRAY 类型的一个成员。

4.3.3.108 S5005

功能块实例作为函数结果是不支持的。

在 OpenPCS 中不支持功能块实例作为一个函数的返回类型。

4.3.3.109 S5006

功能块实例作为参数是不支持的。

功能块的参数类型在 OpenPCS 中不支持。

4.3.3.110 S5008

需要整型或枚举型。无效的数组索引。

作为索引变量访问的一个索引，这个变量或常数类型是无效的。一个索引必须是 INT 类型或枚举类型。

4.3.3.111 S5009

无效的序列头。当前的结果为未定义。使用"LD"以初始化当前结果。

这个错误发生在顺序使用当前返回结果的指令中。第一个指令通常是一个装载指令。这个错误也可能发生在当前返回结果用在一个 CAL,JMP 或标签指令之后的第一条指令中。

例子：

```
PROGRAM main
VAR
  Switch : BOOL;
(*...*)
END_VAR
ST Switch (* 错误：当前返回结果没有定义 *)

LD Switch
EQ TRUE
JMPC NextStep

LD TRUE
JMP End (* 在执行 JMP 指令之后，前面指令装载的值可能会被丢失 *)
NextStep:
```

```
LD FALSE


END:
ST Switch (* 错误：在一个标签之后，当前返回结果没有定义 *)
(* 代码 *)
END_PROGRAM
```

4.3.3.112 S5010

无效的指令在括号计算中。

在给定位置的指令不允许在圆括弧之间。请替换这条指令或把它移到圆括弧外面。

例子：

```
FUNCTION_BLOCK Count
VAR_INPUT
    StartValue : DINT;
    FReset : BOOL;
END_VAR
VAR_OUTPUT
    CurrentCountValue : DINT;
END_VAR
VAR
    CountValue : DINT;
END_VAR
LD fReset
EQ TRUE
JMPCN Continue
```

```
LD StarValue
ST CountValue

Continue:
LD CountValue
ADD 1
ST CountValue
ST CurrentCountValue
END_FUNCTION_BLOCK

PROGRAM main
VAR
	Counter : Count;
	StartValue : DINT;
	Result : DINT;
(*...*)
END_VAR
LD 5
ADD (StartValue
ST Counter.StartValue
EQ 1000
ST Counter.fReset
CAL Counter (* 错误：CAL 指令不允许在圆括弧之间 *)
LD Counter.CurrentCounter (* 错误：装载指令不允许在圆括弧之间 *)
)
ST Result
END_PROGRAM.
```

4.3.3.113 S5011

功能块实例的数组无效..

不支持功能块的数组。

4.3.3.114 S5012

结果类型和操作数类型不兼容。

前面操作的返回类型和存储这个返回结果的变量类型不匹配。

例子：

```
VAR
X : INT;
END_VAR
LD 65000
ST x  (* 65000 不是 INT 数据类型 *)
(*...*)
```

4.3.3.115 S5013

数据类型不兼容。

在调用一个函数或功能块时，前面操作的返回类型和第一个输入参数类型不匹配。

例子：

```
FUNCTION Fun1
VAR
InVar : INT;
END_VAR
(* 代码 *)
END_FUNCTION
```

```
PROGRAM main
VAR
X : DINT;
END_VAR
LD x
ADD 1000
Fun1 (* 错误：前面操作的返回类型是 DINT 类型，而 Fun1 的第一个输入参数是 INT 类型 *)
ST x
(*...*)
END_PROGRAM
```

4.3.3.116 S5014

错误的参数号。

在调用一个函数或功能块时，发现太多参数。

4.3.3.117 S5015

无效的直接地址类型。

一个局部变量被声明成不支持的数据类型。只支持 ANY_NUM 或 ANY_BIT 数据类型的局部变量。

4.3.3.118 S5016

变量未只读。写访问无效。

企图写一个只能读的变量。

4.3.3.119 S5017

变量不是结构。

对一个结构初始化时，赋值的变量不是一个结构类型。

例子：

```
VAR
A : INT := (m1 := 5, m2 := TRUE); (* 不允许 *)
END_VAR
```

4.3.3.120 S5018

变量不是数组

企图把一个已初始化的数组分配给一个不是数组类型的变量。

例子：

```
VAR
A : INT := [4]; (*不允许*)
END_VAR
```

4.3.3.121 S5019

初始值和变量类型不兼容。

初始化值类型和变量类型不匹配。

例子：

```
VAR
X : INT := 65000;
END_VAR
(*...*)
```

4.3.3.122 S5020

太多初始值。

初始化一个数组类型或变量时，提供的元素个数超过了数组定义的元素个数。

例子：

```
VAR
A : ARRAY [1..5] OF INT := [1, 2, 3, 4, 5, 6]; (* 初始化值太多了，数组只有 5 个元素 *)
END_VAR
```

4.3.3.123 S5021

形式参数正确地声明。

需要一个输出参数名，在当前范围内找到的这个标识符不是一个输出参数名。

提示：

检查这个名字是否拼写正确。

确定这个标识符不是一个输入或输出参数。

4.3.3.124 S5022

多个初始化。

这个错误发生在调用一个功能块实例时，一个参数被初始化两次。

例如：

```
FUNCTION_BLOCK Fb1
VAR_INPUT
InParam1 : int;
InParam2 : int;
InParam3 : bool;
END_VAR
```

```
(* 代码 *)
END_FUNCTION_BLOCK

PROGRAM main
VAR
fbInst : fb1;
END_VAR
(* 代码 *)
cal fbInst( InParam1 := 1,
InParam1 := 2,
InParam3 := true
)
(* 代码 *)
END_PROGRAM
```

4.3.3.125 S5023

太多初始数据。

这个错误发生在一个外部结构初始化时，这个结构或实例的一个成员被初始化两次。

例如：

```
TYPE
        StructType : STRUCT
                Member1 : int;
                Member2 : int;
                Member3 : bool;
        END_STRUCT;
END_TYPE
VAR
```

```
    StructVar : StructType := (Member1 := 1, Member1 := 2, Member3 := FALSE);
END_VAR
```

4.3.3.126 S5024

此操作不允许的类型。

在给定位置的操作没有定义当前返回结果的类型，即实际参数类型和第一个形参类型不匹配。

例子：

```
VAR
  X :REAL;
END_VAR
LD 1 (* 常数 1 能被修改成任何的整数或位类型 *)
LN (* 错误：LN 只定义给 ANY_REAL 类型 *)
ST X
(*...*)
```

4.3.3.127 S5025

此函数不允许的参数类型。

在给定位置，实际参数类型与任何允许的参数类型不匹配。

例子：

```
VAR
X : STRING;
END_VAR
```

4.3.3.128 S5026

无效的形式参数类型。

需要一个输入或输出参数名，在当前范围内找到的这个标识符即不是一个输入参数名也不是一个输出参数名。

提示：

检查这个名字是否拼写正确。

确定这个标识符不是一个输出参数。

4.3.3.129 S5027

不兼容的操作数类型。

在给定位置的操作的操作数必须相匹配，即它们必须有相同的数据类型或有一个参数是常数，这个常数有可能被修改成别的操作数类型。

例子：

```
VAR
  X :REAL;
END_VAR
LD 1 (*常数 1 能被修改成任何的整数或位类型 *)
MAX X (* 错误：X 是一个 REAL 类型 *)
ST X
(*...*)
```

4.3.3.130 S5028

无效的数据类型对于这个操作。

这个错误发生在给定位置的操作，不允许提供的实际参数类型。

例子：

```
VAR
StringVar : STRING;
```

END_VAR

LD 1

CONCAT EXAMPLE (* **错误**：CONCAT 操作希望一个 STRING 操作数作为第一个输入参数 *)

ST StringVar

(*...*)

4.3.3.131 S5029

无效的功能块调用。

这个错误发生在调用一个功能块实例时，这个实例是调用的功能块或程序的一个输入参数。

例子:

```
FUNCTION_BLOCK Fb1
VAR_INPUT
    InParam1 : int;
    InParam2 : int;
    InParam3 : bool;
END_VAR
(* 代码 *)
END_FUNCTION_BLOCK


FUNCTION_BLOCK Fb2
VAR_INPUT
fbInstInput : Fb1;
(* 别的输入定义 *)
END_VAR
VAR
(* 局部变量定义 *)
```

```
END_VAR

(* 代码 *)

cal fbInstInput( InParam1 := 1,

                 InParam2 := 2,

                 InParam3 := true

                 )

(* 代码 *)

END_PROGRAM
```

4.3.3.132 S5030

变量未只写。读访问无效。

企图读一个只能写的变量。

4.3.3.133 S5031

位访问仅适用于位数据类型。

这个错误发生在一个变量中选择了位，但这个变量不是位数据类型或 BOOL 类型。

例子：

```
VAR

DintVar : DINT;

BoolVar : BOOL;

END_VAR

LD DintVar.4 (* 错误：位选择只允许在除 BOOL 类型之外的 ANY_BIT 类型变量中 *)

ST BoolVar

(*...*)
```

4.3.3.134 S5032

位的位置大于所选变量的位数。

这个错误发生在所选择的变量中，选择的位的位置大于最大有用位的位数。一个位选择的变量可选的位数依赖于变量的数据类型。这个位的位置从最小有用位的 0 位开始计数，到最大有用位的 n-1 位，这个 n 是数据类型位数。

例如：

```
VAR
    wVar : WORD := 5;
    fVar : BOOL := FALSE;
END_VAR
(* 代码 *)
LD wVar.16 (* 这个选择的变量是一个字类型的变量。也就是，它有 16 位，位置从 0 到 15。*)
ST fVar
(* 代码 *)
```

4.3.3.135 S5033

IN_OUT 参数丢失。请提供每个形式的IN_OUT 参数一个时间参数。

这个错误发生，一个功能块至少有一个 IN_OUT 参数在调用各自的功能块实例时没有提供实际的参数。 在每一个功能块实例调用时，涉及的 IN_OUT 参数必须提供一个实际的参数。

例如：

```
FUNCTION_BLOCK Fb1
VAR_IN_OUT
    InOutParam1 : INT;
    InOutParam2 : BOOL;
END_VAR
(* 代码 *)
```

```
END_FUNCTION_BLOCK


PROGRAM main
VAR
        fbInst : fb1;
        IntVar1 : INT;
        IntVar2 : INT;
END_VAR
(* 代码 *)
cal fbInst() (* 错误的：没有提供 FB1 的 IN_OUT 变量的实际参数 *)
cal fbInst( InOutParam1 := IntVar1 ) (* 错误的：给第二个 IN_OUT 参数的实际参数漏掉了*)
cal fbInst ( InOutParam1 := IntVar1,
                InOutParam2 := IntVar2
) (* 正确的：FB1 的每一个形参都提供了一个实际参数*)
(* 代码 *)
END_PROGRAM
```

4.3.3.136 S5034

无效的输入输出参数。输入输出参数不得未表达式或常数。

这个错误发生，提供给 IN_OUT 参数的是一个表达式或常数。这是不允许的，因为 IN_OUT 参数是参考变量
.

例如：

```
FUNCTION_BLOCK Fb1
VAR_IN_OUT
      InOutParam1 : INT;
```

```
        InOutParam2 : BOOL;
END_VAR
(* 代码 *)
END_FUNCTION_BLOCK


PROGRAM main
VAR
        fbInst : fb1;
        IntVar1 : INT;
        IntVar2 : INT;
END_VAR
(* 代码 *)
cal fbInst( InOutParam1 := IntVar1,
                InOutParam2 := 5
) (* 错误：给第二个 IN_OUT 参数的实际参数是一个常数 *)
cal fbInst( InOutParam1 := IntVar1,
                InOutParam2 := (IntVar1
                    ADD IntVar2)
) (* 错误：给第二个 IN_OUT 参数的实际参数是一个表达式 *)
cal fbInst ( InOutParam1 := IntVar1,
                 InOutParam2 := IntVar2
) (* 正确：提供给 FB1 的两个 IN_OUT 参数是变量*)
(* 代码 *)
END_PROGRAM
```

4.3.3.137 S5035

通用数据类型不允许。

这个错误发生在一个 ANY 数据类型被一个变量或参数声明中。 只允许在函数过载和标准函数类型转化或生产商提供的函数中使用类的（generic）数据类型。

例子：

```
FUNCTION IntegerToString : STRING
VAR_INPUT
InVar : ANY_INT; (* 错误：用户自定义的函数不能过载 *)
END_VAR
(* 代码 *)
END_FUNCTION
```

4.3.3.138 S5036

在此变量部分局部类型是不允许的。

这个错误发生在一个全局或外部变量部分或在一个参数声明部分定义了用户自定义的类型。全局和外部变量象参数一样都必须有一个预定义的类型或全局类型。全局类型依赖于硬件的类型，这些类型是由固件或工程全局由自定义的类型提供的。

例子：

```
PROGRAM main
TYPE
    StructType : STRUCT
        Member1 : BOOL;
        Member2 : STRING;
    END_STRUCT;
(* 别的类型定义 *)
END_TYPE
VAR_GLOBAL
  GlobVar : StructType; (* 这是不允许的，因为在别的 POU 中不知道 StructType *)
```

```
(*别的全局变量定义*)
END_VAR

VAR
(* 局部变量定义*)
END_VAR
(* 代码 *)
END_PROGRAM

FUNCTION_BLOCK Fb1
TYPE
    StructType：STRUCT
        Member1：BOOL;
        Member2：STRING;
    END_STRUCT;
END_TYPE
VAR_EXTERNAL
        GlobVar：StructType; (* 这是不允许的，因为在别的 POU 中不知道 StructType *)
(* 别的外部定义 *)
END_VAR
VAR_INPUT
    InVar：StructType; (* 这是不允许的，因为在别的 POU 中不知道 StructType *)
(* 别的输入定义 *)
 END_VAR
(* 代码 *)
END_FUNCTION_BLOCK
```

4.3.3.139 S5037

在数组访问括号[....] 中的索引太多。

这个错误发生在，访问数组元素时提供的维数超过了数组定义的数据类型的维数。

例子：

```
PROGRAM main
TYPE
    ArrayType : Array[1..5, 1..20] of INT;
(* 别的类型定义 *)
END_TYPE
VAR
    ArrayVar : ArrayType;
    IntVar : INT;
    (* 别的变量定义 *)
END_VAR
LD ArrayVar[1, 2, 3] (* 错误：ArrayType 类型变量只有 2 维 *)
ST IntVar

(* 代码 *)
END_PROGRAM
```

4.3.3.140 S5038

地址变量在原型中仅允许作为参数。

一个直接表示的变量被定义在 POU 的 VAR_INPUT，VAR_OUTPUT 或 VAR_IN_OUT 部分，这是不允许的。直接表示的变量也不允许在函数和功能块中定义。程序中不支持 VAR_INPUT，VAR_OUTPUT 和 VAR_IN_OUT 变量。

如果想要在一个功能块中访问一个直接表示的变量，程序的 VAR_GLOBAL 部分需声明这个变量的一个代表符号，并在功能块的 VAR_EXTERNAL 变量部分用这个代表符号声明。 函数不允许访问直接表示的变量。

例子：

```
FUNCTION_BLOCK SetOutput
VAR_EXTERNAL
OutputLocation : BOOL;
END_VAR
VAR_INPUT
Value : BOOL;
END_VAR
LD Value
ST OutputLocation
END_FUNCTION_BLOCK


PROGRAM main
VAR_GLOBAL
OutputLocation AT%Q0.0 : BOOL;
END_VAR
VAR
Switch : SetOutput;
CurrentValue : BOOL;
END_VAR
LD CurrentValue
NOT
CAL Switch(Value := CurrentValue)
END_PROGRAM.
```

4.3.3.141 S5039

无效的操作数类型对于此操作。

在&算子之前的标识符不是一个直接表示的变量名。

提示：

检查这个名字是否拼写正确。

确定这个变量是一个直接表示的变量。

4.3.3.142 S5040

在数组访问括号[....] 中的索引太少。

这个错误发生在，访问数组元素时的维数少于数组定义的数据类型的维数。

例子：

```
PROGRAM main
TYPE
    ArrayType : Array[1..5, 1..10, 1..20] of INT;
    (* 别的类型定义 *)
END_TYPE
VAR
    ArrayVar : ArrayType;
    IntVar : INT;
    (* 别的变量定义 *)
END_VAR
LD ArrayVar[1, 2] (* 错误：ArrayType 类型变量有 3 维 *)
ST IntVar

(* 代码 *)
END_PROGRAM
```

4.3.3.143 S5041

在此上下文中类型 INT24 或 REAL48 的值有效。

这个操作不支持这种类型。

4.3.3.144 S5042

功能块实例不能为 "RETAIN"。

一个功能块实例被定义在带有 RETAIN 属性的变量部分里，这是不支持的。请删除这个属性或把实例声明移到另外一个不带有 RETAIN 属性的变量部分中。

S5043

变量, 常数和参数载声明初始值中是不允许的。请使用文字或枚举值。

在声明部分，变量、常量或参数不能用来初始化值。

4.3.3.145 S6002

无原型。致命的系统错误

一个未知类型的类型名被用在一个变量声明部分或一个函数调用中。

提示：

确定带有这个名字的类型、函数或功能块在当前工程范围内已定义了。

确定这个类型、函数或功能块名拼写正确。

重新编译整个工程。

如果上面的提示没有消除这个问题，请参看您的硬件资料。

4.3.3.146 S6004

递归 (即，直接或间接自参照) 被检测到。

发现递归。一个函数不能直接或间接递归调用它本身(即调用别外一个函数，但这个函数在调用层次上又调用了它本身)。功能块和程序不能直接或间接的声明它们自己的实例(即调用别外一个功能块实例时，这个功能块在调用层次上又声明了它本身的一个实例)。

4.3.3.147 S6005

太多类型和功能块。请询问制造商。

这个错误发生在，在一个 POU 的调用层次上使用了太多的函数或功能块。要知道能支持的最大数目的类型、函数和功能块，请参看表 D.1□□执行依赖的参数。

4.3.4 连接错误

4.3.4.1 L10001

变量声明两次: %1.

指定名字的变量被声明了两次。

提示：

如果这个变量在一个程序 POU 中声明了，请检查是否有一个相同名字的资源全局变量被声明了。

如果这个变量是一个资源全局变量，请在资源的一个程序 POU 中检查是否有一个相同名字的全局变量。

如果上面的一种情况是真，改变其中一个变量的名字或把程序 POU 中的这个变量移到 VAR_EXTERNAL 变量声明部分。注意：如果您把这个变量移到外部变量部分，则每次访问外部变量时要访问相同名字的资源全局变量。

4.3.4.2 L10004

未解决的外部: %1。

没有找到指定名字的全局变量，或者没有找到指定名字的功能块类型。

提示：

确定这个变量名是否拼写正确。

如果这个变量不是一个功能块实例，则请确定有相同名字的变量是否正在调用的程序中的 VAR_GLOBAL 变量声明部分或在资源全局变量声明文件中被声明了。

如果这个变量是一个功能块实例，则请确定这个功能块是否被成功的编译了，即存在一个这个功能块的目标文件。

4.3.4.3 L10026

这个硬件不支持地址<地址描述> 。

提示：

检查地址是否拼写正确。

检查地址描述的语法是否正确。地址描述的语法是依赖于硬件的，但必须是一个以百分号%开头的，后面跟着位置前缀，一个前缀大小，一个或多个无符号整数，这些整数是周期(.)分开的字符串形式。前缀大小不一可以为空。要了解硬件的有效地址和前缀大小，请参看您的硬件资料。

4.3.4.4 L10027

无效的硬件描述: %1.

没有找到带有名字<硬件名>的硬件描述文件。

提示：

检查资源说明是否包含一个有效的硬件模块名。要了解更多有关设置资源说明的信息，请参考 工程浏览器 这一章中的编辑一个资源部分的内容。

重新设置 OpenPcs。如果这个错误还没有解决，请参考硬件文档或者联系您的硬件生产商。

4.3.4.5 L10029

硬件配置错误。

在获取硬件信息时出错了。请检查硬件配置文件是否正确或在 OpenPCS 目录里指定的固件 DLL 是否安装。

注意：

这个文件只能由生产商修改。

4.3.4.6 L10030

无效的变量类型: %1。

发现一个复合类型(array, struct, string)的直接表示变量。硬件是不支持这个变量的。

4.3.4.7 L10031

直接表示变量的初始化不允许。

发现初始化了一个直接表示的变量。硬件是不支持这个操作的。请册除这个初始值。

4.3.4.8 L10032

地址 %1 在此处无效。

指定的地址是一个有效地址，但在当前位置(Task,POU, Resource, Configuration)中是不允许的。

4.3.4.9 L10033

属性 RETAIN 不支持直接表示的变量。

发现了一个带有 RETAIN 属性的直接表示变量，这个变量硬件是不支持的。请把这个变量声明移到别一个变量声明部分或把这个属性删掉。

4.3.4.10 L10034

属性 CONST 不支持直接表示的变量。

发现了一个带有 CONST 属性的直接表示的变量，这个变量硬件是不支持的。请把这个变量声明移到别一个变量声明部分或把这个属性删掉。

4.3.4.11 L10035

功能块的实例限制 %1 已达到。

超过了指定的功能块实例的最大个数。一个固件功能块实例的最大个数是由硬件决定的，这个最大数由硬件生产商通过设置或改变硬件描述文件中的功能块说明部分的 MaxInstances 句柄的值。请参考您的硬件资料，获得固件功能块实例的最大个数。

4.3.4.12 L10036

无效的处理映像描述。请联系你的制造商。

在硬件配置文件中，处理映像的描述是无效的。请检查输入、输出和生成部分的大小是否正确，以及所有的入口是否在同一个单元中。它们也应该用位或字节来指定大小。

注意：

这个文件只能由生产商修改。

4.3.4.13 L10063

打开一个文件%1 时发生一个错误。

4.3.4.14 L10105

不能装载函数或DLL：%1.

指定的 DLL 或函数不能被装载。或者您的 OpenPCS 目录中没有包含指定名字的 DLL，或者您的 DLL 是一个无效版本的。请重新安装您的系统或查看您的硬件描述文件。

4.3.4.15 L10106

本地代码编译器需要所选的优化。请选择优化和安装本地代码编译器。

只可以激活速度优化选项，但对这个硬件没有定义本地代码编译器。如果安装了一个本地代码编译器，则只有最优的速度是有效的。如果您没有本地代码编译器，请在编辑资源说明对话框中选择另一个最佳的。适合您硬件的本地代码编译器请询问您的生产商。

4.3.4.16 L12001

类型冲突。外部变量类型与同名的全局变量类型不匹配。

发现了一个与外部变量有相同名字的全局变量，但是全局变量和外部变量的类型是不相同的。

提示：

确定这个外部变量名拼写是否正确。

确定这个外部变量类型拼写是否正确。

确定这个全局变量是否是所需的变量。

改变外部变量或全局变量的类型。

4.3.4.17 L12002

对此变量的可读访问是不允许的: %1.

企图读一个只能写的变量。

提示：

确定这个指定的变量名拼写是否正确。

指定的变量是一个输出局部变量。读访问一个输出局部变量是不允许的。

4.3.4.18 L12003

这个变量的可写访问是不允许的: %1。

企图写一个只能读的变量。

提示：

确定这个指定的变量名是否拼写正确。

指定的变量是一个常数。写一个常数变量是不允许的。检查这个 CONSTANT 属性是否能从变量中删除掉。

指定的变量是一个输入局部变量。写访问一个输入局部变量是不允许的。

4.3.4.19 L12005

内部链接器错误。请联系您的生产厂商。

4.3.4.20 L12006

内存分配故障。没有足够的内存来执行这个操作。

4.3.4.21 L12007

无对象信息被找到在任务 %1。请重新生成全部。

指定任务的目标文件(<任务名>.crd)没有找到。请重建整个资源。

4.3.4.22 L12008

解释器栈溢出在任务 %1。

解释调用堆栈溢出。请减少<任务名>的调用层次的深度。

4.3.4.23 L12064

Error exporting OPC variables to OPC server configuration. Error code: %1.

An OPC variable is erroneous. Please use a proper one.

一个错误的 OPC 变量，请用正确的。

4.3.4.24 L12065

Error initializing ConfOPC.DLL. Please contact your manufacturer.

DLL 不能初始化。请咨询硬件制造商。

4.3.4.25 L12066

存储器定位失败。没有足够内存以执行操作。

直接变量须被转移到合适对齐的地址，以避免在某些以 2 或 4 对齐的控制器潜在的错误表现。如果对齐为 2，所有大小为 WORD(W)或 DWORD(D)的变量须被转移到偶数的地址。如果对齐为 4，所有大小为 WORD(W)的变量须被转移到偶数的地址，大小为 DWORD(D)的变量须被转移到 4 的倍数的地址。

4.3.4.26 L12996

不认识的命令: %1.

用了一个带有 ITLINK 的未知命令行。

4.3.4.27 L12997

不认识的对象类型: %1。

发现了一个无效的目标文件。请重新编译整个资源。

4.3.4.28 L12998

无效的对象类型。类型被发现/需要: %1

发现了一个无效的目标文件。请重新编译整个资源。

4.3.4.29 L12999

无效的对象版本被发现。类型被发现/需要: %1

目标文件版本和编译目标版本不相同。目标文件已被创建为一个不同的编译版本。请重新编译整个资源。

4.3.4.30 L13000

全局变量资源信息装载失败。

没有找到带有资源全局信息的目标文件。请重建整个资源。

4.3.4.31 L13001

无对象信息被找到在 pou %1.

没有找到指定 POU 的目标文件(<POU 名>.obj) 。请重建整个资源。

4.3.4.32 L14009

资源大小超出 PLC 内存。

资源大小超过 PLC 内存限制。该计算可能跟实际值有出入。

4.3.4.33 L14010

资源大小到达警戒值。 %1

资源的大小到达了设定的警告限度。在浏览器选项里设置该大小。

4.3.4.34 L15001

未定义的任务类型被使用或者没有为任务 %1 定义任务类型。

检查任务类型属性中的配置参数。您可以咨询硬件制造商。

4.3.4.35 L20012

Persistency file creation disabled due to online linking

如果您的硬件使用在线连接，建立永久文件被禁止。

4.3.5 编译错误

4.3.5.1 C10006

数据类型 "REAL" 不支持。

当前硬件不支持 REAL 数据类型。要了解 OpenPcs 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看您的硬件资料，得到您的硬件能支持的数据类型列表。

4.3.5.2 C10007

数据类型"DATE"不支持。

不支持 DATE 数据类型。要了解 OpenPcs 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看您的硬件资料，得到您的硬件能支持的数据类型列表。

4.3.5.3 C10008

数据类型 "TIME_OF_DAY" 不支持。

不支持 TIME_OF_DAY 数据类型。要了解 OpenPcs 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看您的硬件资料，得到您的硬件能支持的数据类型列表。

4.3.5.4 C10009

数据类型 "STRING" 不支持。

当前硬件不支持 STRING 数据类型。要了解 OpenPCS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看您的硬件资料，得到您的硬件能支持的数据类型列表。

4.3.5.5 C10010

数据类型"DT"不支持。

不支持 DATE_AND_TIME 数据类型。要了解 OpenPCS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看您的硬件资料，得到您的硬件能支持的数据类型列表。

4.3.5.6 C10012

数据类型 "TIME" 不支持。

不支持 TIME 数据类型。要了解 OpenPcs 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看您的硬件资料，得到您的硬件能支持的数据类型列表。

4.3.5.7 C10017

"VAR_INPUT", "VAR_OUTPUT" 和 "VAR_IN_OUT" 部分在程序中不支持。

不支持程序中的 VAR_INPUT, VAR_OUTPUT 和 VAR_IN_OUT 部分。要了解更多有关能支持的变量类型的信息请参看 IEC 1131-3 说明。

4.3.5.8 C10019

直接地表示变量在这个 POU 中是不允许的。

任一的一个 POU 是一个函数、功能块或是一个带有全局符号定义的文件。直接表示的变量不允许在函数或功能块中使用。如果要从一个功能块中访问直接表示的变量，请在程序的 VAR_GLOBAL 变量定义部分声明一个带有符号名的变量，并在功能块的 VAR_EXTERNAL 变量定义部分用这个符号名。函数不允许访问直接表示的变量。

直接表示的资源全局变量必须在一个专门的文件中声明。

4.3.5.9 C10020

这个变量/参数不允许位的访问

变量或参数必须为 ANY BIT 类型。

4.3.5.10 C10021

常数不得为负。

在需要无符号操作数的地方发现一个负数。请改变这个常数值或变量类型(如果可能的话)。

4.3.5.11 C10024

常数超出范围。

在给定位置的常数不在指定的数据类型范围之内。

4.3.5.12 C10025

IN/OUT 参数必须提供一个实际参数。

在一个功能块中定义一个输入输出参数，在一个实例的 CAL 指令中没有提供实际的参数。输入输出参数是参考的，并且必须提供一个实际的参数。

4.3.5.13 C10026

不支持的地址。

在给定的位置的地址，当前硬件不支持。请参看您的硬件资料，获得硬件支持的地址列表。

4.3.5.14 C10028

"STRUCT"类型的输入输出参数不支持。

结构的输入输出参数不支持。请定义一个这种类型的输入参数和输出参数。

4.3.5.15 *C10030*

硬件配置文件项"对齐"必须大于0。

硬件配置文件里，**对齐**的值设成了 0。请至少设成 1。

4.3.5.16 C10031

RETAIN-变量不被硬件所支持。

您的硬件不支持 RETAIN 变量。请删除这个属性。请参看您的硬件资料，获得一个能支持的变量列表。

4.3.5.17 C10034

无效的命令 (对于这个硬件)。

这个硬件不支持在给定位置的命令。请参看您的硬件资料，获得您的硬件不支持的命令列表。要了解 OpenPcs 不支持的命令列表，请参考 IEC 1131-3 说明。

4.3.5.18 C10035

操作数/参数必须是类型 "UINT".

一个函数调用（操作）中，需要一个实际的 UINT 类型的参数，但这个实际参数不是这种类型。

例子：

```
VAR
    StringVariable : STRING;
    Length : INT := 32;
END_VAR
LD EXAMPLE
LEFT length (* 错误：这个参数必须是 UINT 类型 *)
ST StringVariable
```

4.3.5.19 C10036

复杂数据类型的结构和数组不被此硬件支持。

定义了一个结构类型的数组，或一个数组类型的数组，或带有一个结构成员的结构，或带有一个数组成员的结构。硬件是不支持这种定义的。要了解更多有关您的硬件能支持的数据类型，请参看您的硬件资料。

例子：

```
TYPE
DayOfWeek : STRUCT
    Name : STRING;
    DayNumber : UINT;
END_STRUCT;

DayDescriptions : ARRAY[1..100] OF DayOfWeek; (* 错误：星期的天数是一个复合数据类型。硬件不支持复合类型的数组。*)

Presence : STRUCT
        Name : STRING;
        OursPerDay : ARRAY[1..31] OF UINT; (* Error: ARRAY 是一个复合类型。硬件不支持复合类型的结构 *)
END_STRUCT;
```

4.3.5.20 C10038

不能检测常数类型。

不能确定一个常数类型。请初始化这个常数所需要的类型的变量，并用这个变量代替这个常数。

4.3.5.21 C10043

实现代码是不允许的

在一个文件的资源全局变量声明部分发现执行代码，这是不允许的。请在另一个 POU 中声明需要的变量作为外部变量，并把代码移到文件中。

4.3.5.22 C10045

功能块不得被声明。

在不允许声明功能块实例的部分中发现了功能块声明。请把这个声明移到一个能支持功能块实例声明的部分中。

4.3.5.23 C10046

不允许"VAR_GLOBAL"。

在此区类型不被支持的程序组织单元中有一个 VAR_GLOBAL 区。请改变此区类型或从不支持全局变量的文件中移除变量声明。

根据 IEC 61131-3，VAR_GLOBAL 区仅在 PROGRAM 中支持。然而硬件制造商可能将全局变量的声明限制在资源全局变量文件中。也就是说全局变量仅在包含全局变量声明的指定文件中运行。

4.3.5.24 C10047

只有"VAR_GLOBAL"允许。

在不是 VAR_GLOBAL 变量声明部分发现声明了一个文件的全局资源变量，这是不允许的。 请改变这部分的类型或把这个变量声明移到另一个能支持这种声明的文件中。

4.3.5.25 C10049

字符串太长。

一个字符串定义时指定了长度，但这个长度超过了硬件能支持的最大长度。OpenPcs 能支持的最大长度请参考 IEC 1131-3 说明。无论如何，硬件生产商能通过修改硬件描述文件中 MODULE 部分的 MaxStringLength 值来限制字符串的最大长度。

4.3.5.26 C10055

变量不能初始化。

发现了一个初始化了的直接表示变量或者硬件不支持变量初始化。OpenPcs 不支持直接表示的变量初始化。生产商能改变硬件描述文件中的 MODULE 部分的 InitVariables 句柄的值为 0 来禁止符号变量的初始化。请参看您的硬件资料，确定您的硬件是否能支持变量初始化。

4.3.5.27 C10057

数据类型不支持。

不支持给定位置的数据类型。要了解 OpenPcs 能支持的数据类型列表，请参考 IEC 1131-3 说明。要了解您的硬件能支持的数据类型列表，请参看您的硬件资料。

4.3.5.28 C10060

LD/ST 的功能块实例是不允许的。

发现了功能块实例的一个 LD 或 ST 指令作为一个操作数，这是不允许的。

4.3.5.29 C10063

打开一个文件时发生错误。

4.3.5.30 C10064

内部编译错误号 %1 。

发生了一个内部编辑错误。请联系您的生产商。

4.3.5.31 C10067

结构声明不支持。

发现了一个结构声明，但硬件不支持这个声明。OpenPcs 能支持结构声明。无论如何，硬件生产商能设置配件描述文件中的 MODULE 部分的 StructAllowed 句柄的值为 0 来禁止结构声明。请参看您的硬件资料，确定您的硬件是否支持结构声明。

4.3.5.32 C10068

数组声明不被支持。

发现了一个数组声明，但您的硬件不支持这个声明。OpenPcs 能支持数组声明。无论如何，硬件生产商能设置配件描述文件中的 MODULE 部分的 ArrayAllowed 句柄的值为 0 来禁止数组声明。请参看您的硬件资料，确定您的硬件是否支持数组声明。

4.3.5.33 C10069

枚举型数据类型不支持。

发现了一个列举数据类型声明，但您的硬件不支持这个声明。OpenPcs 能支持列举数据类型声明。无论如何，硬件生产商能设置配件描述文件中的 MODULE 部分的 EnumAllowed 句柄的值为 0 来禁止列举数据类型声明。请参看您的硬件资料，确定您的硬件是否支持列举数据类型声明。

4.3.5.34 C10075

太多数组元素.数组大小不能超过最大片段大小.

一个数组的索引超过了能支持的范围[-32767, 32767]。

4.3.5.35 C10076

数组下界超过允许的最小值 (维数%1).

数组下界超过支持范围[-32767, 32767]。Dimension #表示错误的数组范围。

4.3.5.36 C10077

数组上界超过允许的最大值 (维数%1).

数组上界超过支持范围[-32767, 32767]。Dimension #表示错误的数组范围。

4.3.5.37 C10078

无效的全局或直接变量类型。

定义了一个直接表示的复合变量或一个用户自定义的数据类型。这个定义是不支持的。全局变量的结构类型也不支持。

4.3.5.38 C10083

只有直接变量允许在这个 POU 中。

资源全局变量被分成两种类型的文件　　只包含符号变量的文件和只包含直接表示的变量的文件。在这些文件中，符号变量和直接表示的变量一定不能混在一起。要了解更多有关怎样声明这种类型的变量，请参考 工程浏览器这一章中的增加[direct]全局变量这一部分内容。

4.3.5.39 C10084

全局结构不支持。

请在一个局部变量部分声明这个变量且使用输入输出参数，这些参数将被一个函数或功能块改变它们的值。所需要的结构类型声明必须在工程层次上声明。

例子：

```
(* 下面的结构必须被声明为一个工程全局类型 *)
TYPE
    DayOfWeek : STRUCT
        Name : STRING;
        DayNumber : UINT;
    END_STRUCT;
END_TYPE

FUNCTION_BLOCK AdjustDayName
VAR_INPUT
  DayIn : DayOfWeek;
END_VAR
VAR_OUTPUT
  DayOut : DayOfWeek;
END_VAR
LD DayIn
ST DayOut

LD DayIn.DayNumber
EQ 1
LD MONDAY
ST DayOut.Name

LD DayIn.DayNumber
EQ 2
LD TUESDAY
ST DayOut.Name
```

```
(*...*)
END_FUNCTION_BLOCK

PROGRAM main
VAR
    Day : DayOfWeek;
    DayNumber : UINT;
END_VAR

(*...*)
LD DayNumber
ST Day.DayNumber
CAL AdjustDayName(DayIn := Day | Day := DayOut)

(*...*)
END_PROGRAM
```

4.3.5.40 C10092

内存分配失败。

4.3.5.41 C10093

数据段存储器分配失败。

太多数据(例如变量)导致程序或功能块无法装进 64KB 的区块，该错误发生。区块的大小被限定为 64KB。

备注：

如果该错误发生，您可以尝试重新组织程序或功能块，试将部分变量放进其他功能块(功能块可以被用作数据容器)或使用全局资源。

4.3.5.42 C10094

初始数据段存储器分配失败。

太多数据(例如变量)导致程序或功能块无法装进 64KB 的区块，该错误发生。区块的大小被限定为 64KB。

备注：

如果该错误发生，您可以尝试重新组织程序或功能块，试将部分变量放进其他功能块(功能块可以被用作数据容器)或使用全局资源。

4.3.5.43 C10095

代码段存储器分配失败。

当程序代码(UCode/Native Code)无法装进 64KB 的区块，该错误发生。区块的大小被限定为 64KB。

备注：

如果该错误发生，您可以尝试重新组织程序(例如将部分代码放进功能块)，这样程序的大小可以减少的 64KB 以下。

4.3.5.44 C10096

数据段大小达到警戒值。 %1

对应数据段的大小到达了设定的警告限度。在浏览器选项里设置该大小。

4.3.5.45 C10097

初始数据段大小达到警戒值。 %1

对应初始数据段的大小到达了设定的警告限度。在浏览器选项里设置该大小。

4.3.5.46 C10098

代码段大小达到警戒值。 %1

对应代码段的大小到达了设定的警告限度。在浏览器选项里设置该大小。

4.3.5.47 C10100

无效的参数表达式

在调用一个函数或功能块实例时，一个无效的表达式被现当作一个实际参数。

4.3.5.48 C10108

时间(TIME)类型常数超出范围。

要了解 OpenPcs 能支持的 TIME 常数的范围，请参考 EC 1131-3 说明。

4.3.5.49 C10109

无效的数据类型对于这个操作。需要整型或实数类型。

在给定位置的操作只支持整数和实数类型的操作数。

4.3.5.50 C10110

嵌套的函数是不支持的。

在调用一个函数或功能块实例时，发现一个函数调用作为一个实际参数，这是不允许的。请用一个变量保存这个函数返回的结果并用这个变量作为调用这个 POU 的实际参数。

4.3.5.51 C10112

类型冲突。

当前返回的结果类型与所需要的数据类型不匹配，或者实际参数类型与各自的形参类型不匹配。

4.3.5.52 C10113

操作不支持此数据类型。

在给定位置的操作不允许这种操作数的数据类型。要了解更多有关这个操作允许的数据类型请参考 IEC 61131-3 和 IEC 1131-3 说明。

4.3.5.53 C10114

参数表达式不支持此操作。

一个表达式被用作一个实际参数，这是不支持的。请存储这个表达式的结果到一个变量里，并且通过这个变量来调用函数或功能块。

4.3.5.54 C10115

RETAIN 属性对于 FB 实例被禁止。

不支持带有 RETAIN 属性的功能块。请删除这个属性或把这个实例移出这个变量定义部分。

4.3.5.55 C11001

不能明确地确定常数类型 -> take %1.

不能清楚地确定一个数字常数的类型。在这种情况下，通常假定能支持的最大数据类型(ANY_INT, ANY_REAL, ANY_BIT)作为所需要的数据类型。

4.3.5.56 C11007

参数未找到: %1.

发现一个函数调用一个不带任何参数的函数。这是故意的吗？函数不包含内部状态信息，并且只能提供输入参数。返回结果一般是用输入参数计算出来的。因为这个原因，一个不带输入参数的函数通常是没有意义的。请检查这个调用的函数是否有意义。

4.3.6 生成错误

4.3.6.1 M21004

Unknown command: %1.

使用了一个带有 ITMAKE 的未知的命令行。

4.4 热键

4.4.1 常规热键

4.4.1.1.1.1 文件菜单

CTRL+N:	新建
CTRL+F4:	关闭
CTRL+S:	保存
ALT+F10:	语法检查
CTRL+P:	打印

4.4.1.1.1.2 编辑菜单

CTRL+Z:	撤消
CTRL+Y:	重复
CTRL+X/SHIFT+DEL:	剪切
CTRL+C/CTRL+INS:	复制
CTRL+V/SHIFT+INS:	粘贴
DEL:	删除
F4:	下一个错误
SHIFT+F4:	前一个错误
CTRL+F:	查找
CTRL+H:	替换
CTRL+G:	到 IL 行 (SFC)
CTRL+A:	全选
ALT+RETURN:	属性

4.4.1.1.1.3 PLC 菜单

F7:	生成当前资源
CTRL+F7:	重新生成当前资源
F9:	拨动断点
F5:	到
F11:	进入
F10:	下一步
SHIFT+F11:	跳出
ALT+ENTER:	资源属性

4.4.1.1.1.4 窗口菜单

F6:	下一个窗格
ALT+1:	工程
ALT+2:	测试和调试
ALT+3:	文档
ALT+4:	输出

4.4.1.1.1.5 插入-> 变量子菜单

ALT+SHIFT+V:	所有变量
ALT+SHIFT+I:	输入变量
ALT+SHIFT+O:	输出变量
ALT+SHIFT+N:	输入/输出 变量
ALT+SHIFT+L:	局部变量
ALT+SHIFT+G:	全局变量
ALT+SHIFT+E:	外部变量
ALT+SHIFT+F:	FB-实例变量

4.4.1.1.1.6 其他

CTRL+U: 大写

CTRL+SHIFT+U: 小写

4.4.2 编辑器决定的热键

4.4.2.1.1.1 IL/ST 编辑器

CTRL+ALT+F: 插入函数

CTRL+ALT+B: 插入功能块

4.4.2.1.1.2 LADDER 编辑器

F12: 插入网络

CTRL+ALT+F: 插入函数

CTRL+ALT+B: 插入功能块

SHIFT+RETURN: 在注释中插入行

4.4.2.1.1.3 SFC 编辑器

CTLR+ALT+S: 插入步/转移

CTLR+ALT+L: 插入步/左转移

CTLR+ALT+R: 插入步/右转移

CTLR+ALT+J: 插入跳转

CTLR+ALT+B: 插入功能块

CTLR+ALT+F: 插入函数

4.4.2.1.1.4 CFC/FBD 编辑器

CTRL+B: 插入连接

CTRL+SHIFT+V: 在线模式下边际条中在变量名称和变量值之间转换

5 目录

联系人

Supportline

Tel: +49 (0) 9131 / 78 00 - 99
Fax: +49 (0) 9131 / 78 00 - 50
supportline@infoteam.de

infoteam Software AG
Am Bauhof 9
D-91088 Bubenreuth