目 录

第一章 OptiSYS 快速入门

1. 软件安装

1.1 OptiSYS 软件安装要求：

a) 操作系统

Windows 2000 或 Windows XP。

b) 基本硬件

PC：奔腾 166 处理器以上；

32Mbyte RAM，建议 64Mbyte；

70M 以上硬盘剩余空间；CD-ROM 驱动器；

以太网卡；

Microsoft Windows 支持的彩色显示器、键盘和鼠标。

1.2 主要安装步骤

将数据拷贝到你的 PC 上：

放入安装光盘，双击安装程序 setup.exe，按提示，点击“下一步”，安装程序将所需文件安装到你的 PC 中。

2. 创建工程

2.1 新建 Project

点击“开始》程序》infoteam OpenPCS2004》infoteam OpenPCS2004”或双击桌面图标“infoteam OpenPCS2004”打开编程软件。选择“文件》新建”或点击“新建”图标打开新建对话框，如图 1.2.1-1 所示。

图 1.2.1-1

选择新建 OpenPCS Project，输入相应工程名称，选择好程序路径，建立一个新的工程。

3. 连接设置

3.1 Connections

PC 与 PLC 的编程口为以太网连接。(需要更改设置时用)

PLC 具有初始连接设置，IP 地址为 10.10.70.6。安装光盘内有 OPSconfig 程序,能搜索网上PLC,并且对PLC的内部参数进行设置。打开OptiSYSConfig.exe,点击菜单：文件》搜索网络，搜索到 PLC 后，可以对 PLC 的内部参数作相应修改。具体参数视控制系统需要设定。

PLC 内部参数设定好后，再进行程序的相关设置。

点击编程软件菜单：PLC->connections，弹出图 1.2.2-1 对话框：

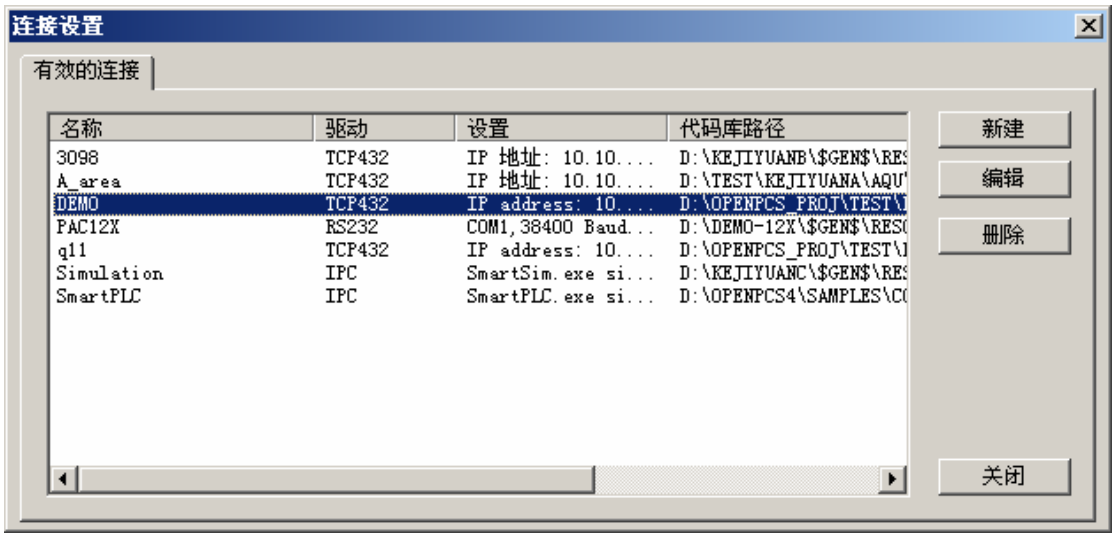


图 1.2.2-1

选择新建（New），弹出图 1.2.2-2 对话框：

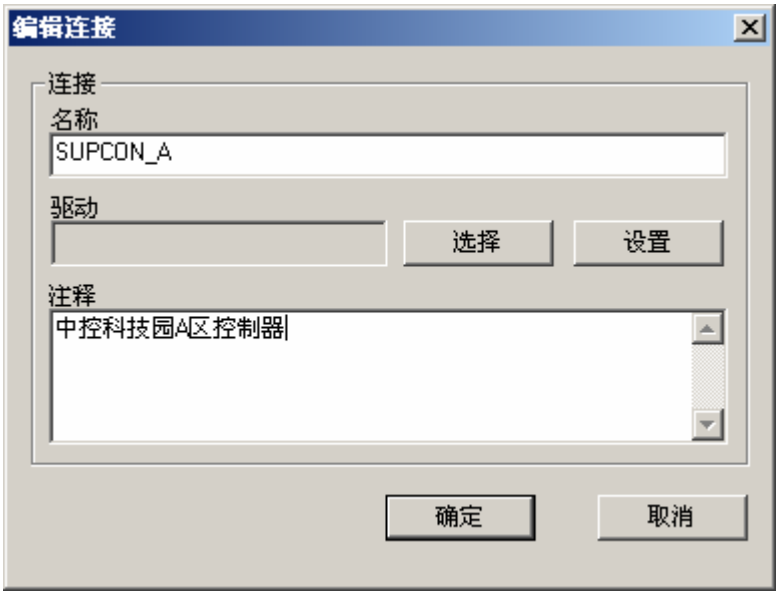


图 1.2.2-2

输入名称(Name)，根据实际连接，选择（Select）相应连接形式，对话框如图 1.2.2-3：

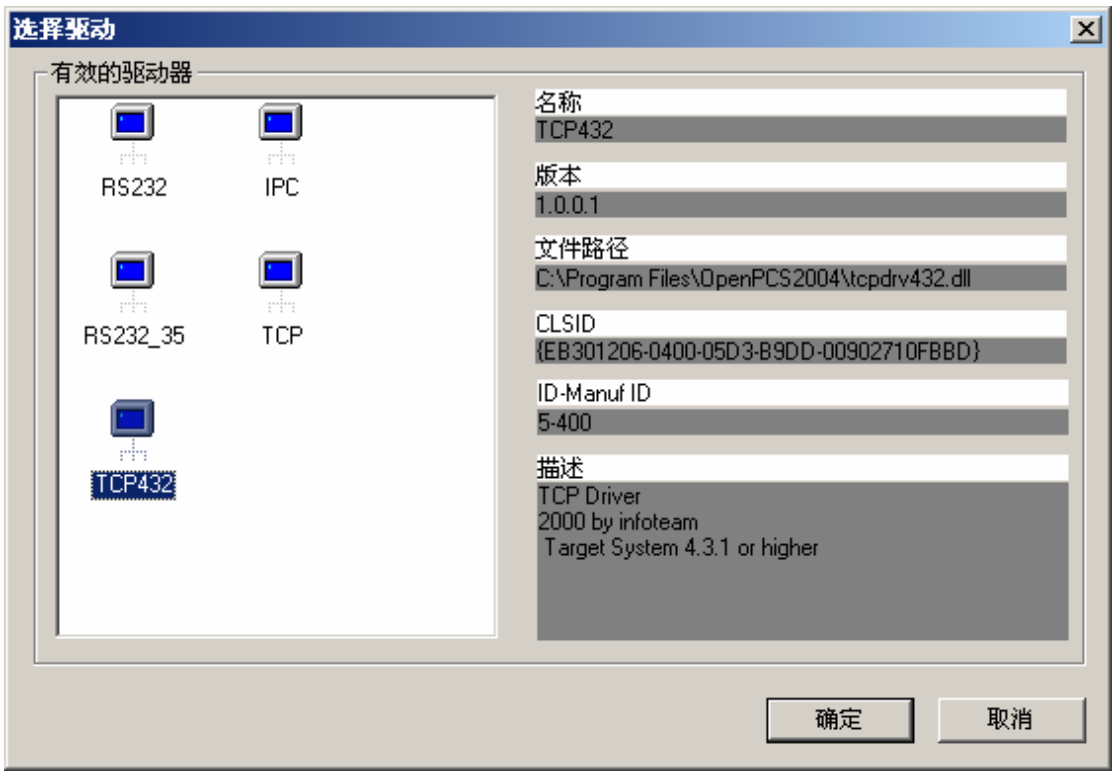


图 1.2.2-3

确定后，设置（Settings）相应连接参数，这里以 TCP432 以太网连接为例，设置好相应 IP 地址及端口号，如图 1.2.2-4 所示：

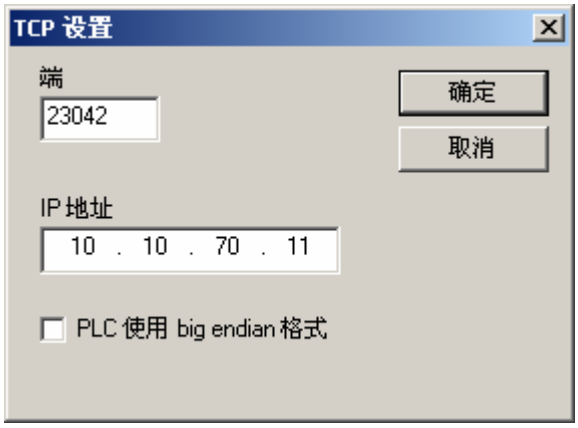


图 1.2.2-4

3.2 Resource

菜单 PLC》Resource Properties，选择 PLC 硬件。

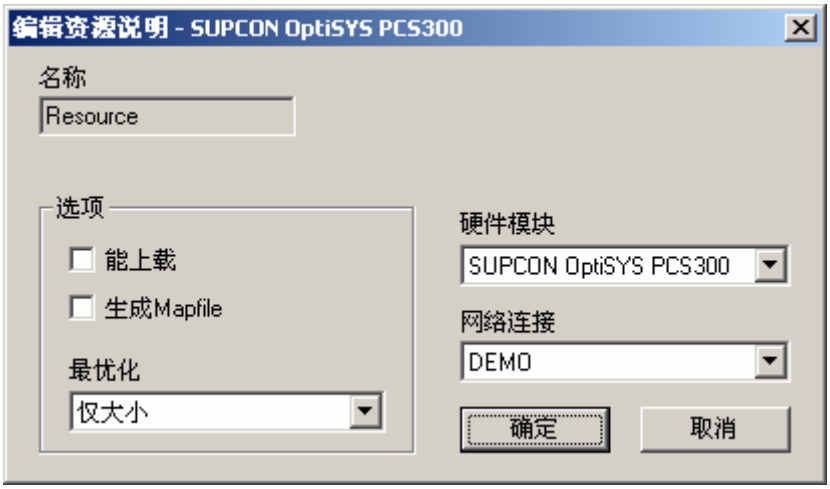


图 1.2.2-5

4. 硬件 I/O 编址及 Memory 地址定义

上述设置完成后，即可对 PLC 进行编程。有关模块输入输出及内部存储器地址定义如下：

4.1 OptiSys 系统变量声名规则

a) 开关量：

开关量一个模块支持8点（或者16点），分别用1个字节中的8个bit（或者1个字的16个bit）对应

i. **开关量输入（DI）**

变量声名方法：’%I’ + ‘addr’ + ’ .’ + ‘bit’

说明：addr 计算方法：（addr=模块地址×16），其中模块地址0～31；

如果模块上点位为第9～16点，则addr相应加1（addr=addr+1）。

bit位计算方法：如果模块上点位为第1至8点，则相应bit为0～7；

如果模块上点位为第9～16点，则bit位为相应点位数减9。

举例：1、声名一个表示模块地址为5的第6个bit位的变量DI_TEST1，

DI_TEST1 at %I80.5 : bool;

其中80 ＝ 16×5

2、声名一个表示模块地址为5、第10个bit位的变量DI_TEST2，

DI_TEST2 at %I81.1 : bool;

其中81 ＝ 16×5+1

ii. **开关量输出（DO）**

变量声名方法：’%Q’ + ‘addr’ + ’ .’ + ‘bit’

说明：addr 计算方法：(addr=模块地址×16) ，其中模块地址0～31；

如果模块上点位为第9～16点，则addr相应加1

（addr=addr+1）。

bit位计算方法：如果模块上点位为第1至8点，则相应bit为0～7；

如果模块上点位为第9～16点，则bit位为相应点位数减9。

举例：1、声名一个表示模块地址为5的第6个bit位的变量DO_TEST1，

DO_TEST1 at %Q80.5 : bool;

其中80 ＝ 16×5

2、声名一个表示模块地址为5的第10个bit位的变量DO_TEST2，

DO_TEST2 at %Q81.1 : bool;

其中 81 ＝ 16×5+1

b) 模拟量：

模拟量一个模块支持8（或16）个字节

i. **模拟量输入（AI）**

变量声名方法：’%I’ + ‘addr’ + ’ .0’

说明：addr 计算方法：(模块地址×16＋变量号×该变量类型长度)

其中模块地址0～31 ，bit位0；

举例:声名4个表示模块地址为5的unsigned int(该变量类型长度 ＝ 2）类型变量

AI_TEST0，AI_TEST1，AI_TEST2，AI_TEST3:

AI_TEST0 at %I80.0 : usint; (80 = 16×5＋0×2)

AI_TEST1 at %I82.0 : usint; (82 = 16×5＋1×2)

AI_TEST2 at %I84.0 : usint; (84 = 16×5＋2×2)

AI_TEST3 at %I86.0 : usint; (86 = 16×5＋3×2)

ii. **模拟量输出（AO）**

变量声名方法：’%Q’ + ‘addr’ + ’ .0’

说明:addr 计算方法:(模块地址×16 ＋变量号×该变量类型长度)

其中模块地址0～31 ，bit位0;

举例：声名4个表示模块地址为5的unsigned int（该变量类型长度＝ 2）类型变量AO_TEST0，AO_TEST1，AO_TEST2，AO_TEST3:

AO_TEST0 at %I80.0 : usint; (80 = 16×5＋0×2)

AO_TEST1 at %I82.0 : usint; (82 = 16×5＋1×2)

AO_TEST2 at %I84.0 : usint; (84 = 16×5＋2×2)

AO_TEST3 at %I86.0 : usint; (86 = 16×5＋3×2)

c) 内部存储器：

最大2048个字节

变量声名方法：’%M’ + ‘addr’ + ’ .0’

说明：addr 计算方法：内部的地址，小于2048

举例：声名4个内部变量VAR_TEST0，VAR_TEST1 VAR_TEST2 ,VAR_TEST3:

VAR_TEST0 at %m0.0 : dword;

VAR_TEST1 at %M10.0 : uint;

VAR_TEST2 at %m20.0 : uint;

VAR_TEST3 at %m30.0 : bool;

5. 程序设计

编程软件支持 5 种编程语言来编写程序，分别是：SFC（Sequential Function Chart），CFC（Continuous Function Chart），ST，IL 和 LD，可以结合各种语言的优缺点，或根据个人编程习惯来选择相应的编程语言。有关语言的使用，可参考其语言帮助。

编程窗口主要有几大部分：工程浏览窗口，代码窗口，输出窗口。如图 1.5-1 所示：

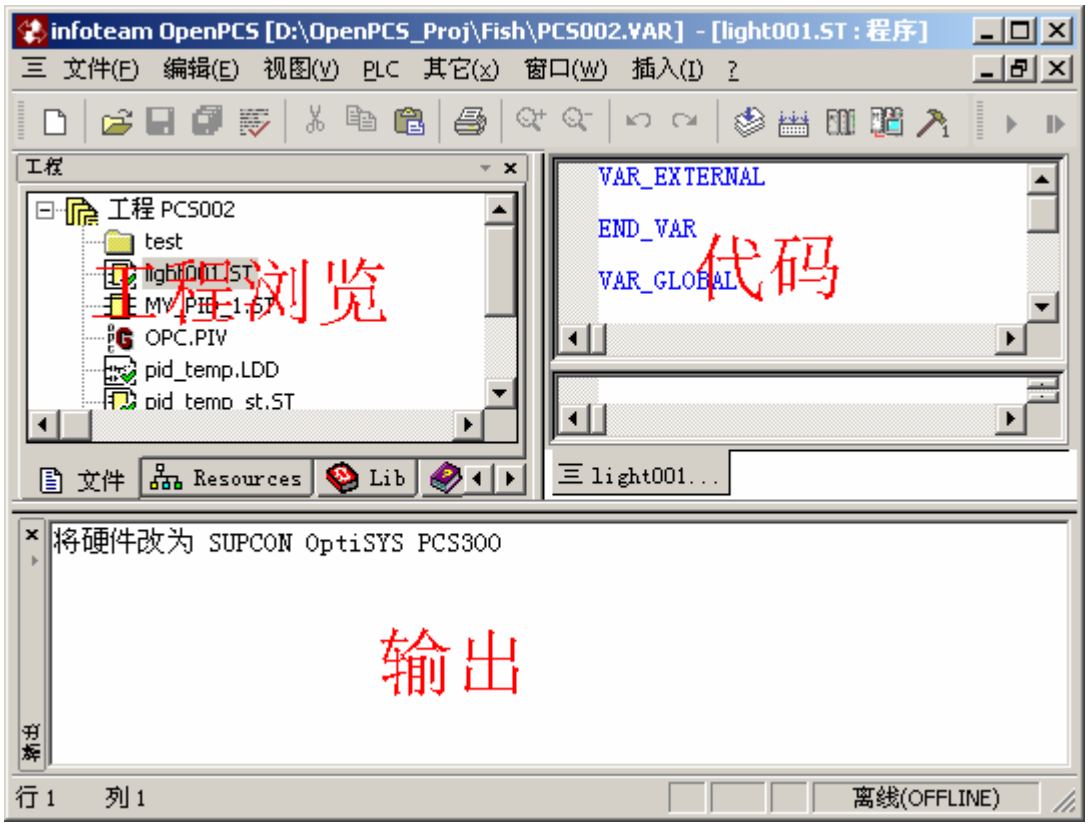


图 1.5 -1

工程浏览窗口分为：Files，Resources，OPC-I/O，lib，help 五页组成，Files 显示的是各个程序文件，可以通过此窗口来打开程序代码；Resources 显示的是各个代码文件的变量，可以此定义某个程序的运行方式，具体操作是打开某一程序属性，选择其运行方式为：cyclic 或 timer 或 interrupt，同时可设定相应运行方式参数。OPC-I/O 显示的是可调用的 OPC。Lib 显示的是可调用的库文件程序。Help 是编程软件帮助。

代码窗口分变量定义和程序两个窗口。所有输入输出、内部地址以及其它自定义变量都必须有变量名称定义。

6. 编译，下载，监控

写完程序，点击 PLC》Build Active Resource/Re build Active Resource/Build All Resources 或者工具栏上相应编译工具图标， 编译程序，输出窗口将显示编译信息。

编译成功，点击 PLC》Online 工者工具栏上相应图标即可下载程序到 PLC，同时会增加变量监控窗口。从浏览栏的 Resources 页中可添加所需监控的变量至监控栏中。在线状态下可操作 PLC 启动或停止。

第二章 OptiSYS 编程软件简介

2.1 ControlX 框架

2.1.1 ControlX 框架：介绍

ControlX 框架容纳了大部分OptiSYS 的工具。ControlX 框架一般看上去跟图2.1.1-1的相似：

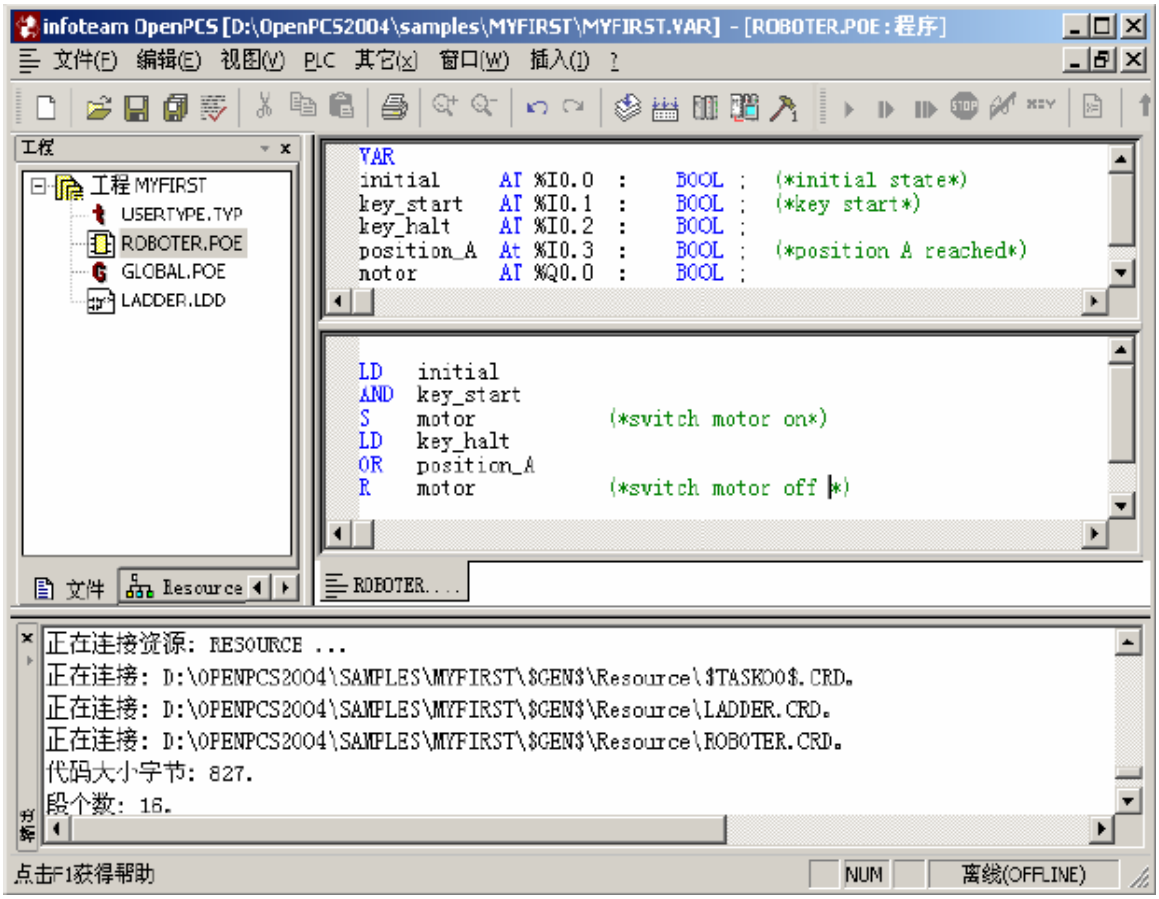


图2.1.1-1

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。ControlX 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ControlX 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OptiSYS 的文件管理。使用浏览器，可将工作组织成文件和工程。从浏览器中，将创建和编辑文件以及编译、下载和监测应用程序。

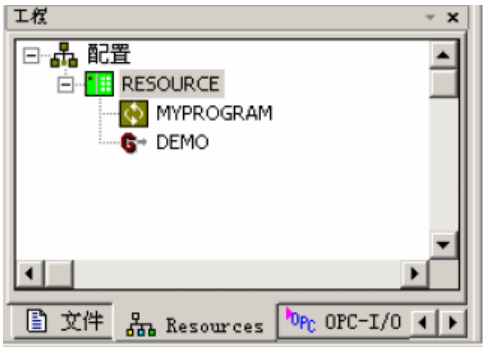


图2.2.1-1

浏览器用户界面由4 个不同的窗口（窗格）组成：

(1) 文件窗格

(2) 资源窗格

(3) OPC-I/O-窗格(可选)

(3) 库窗格

(4) 帮助窗格

2.2.2 浏览器：概况

图2.2.2-1

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。这些文件都是你自己使用OptiSYS 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

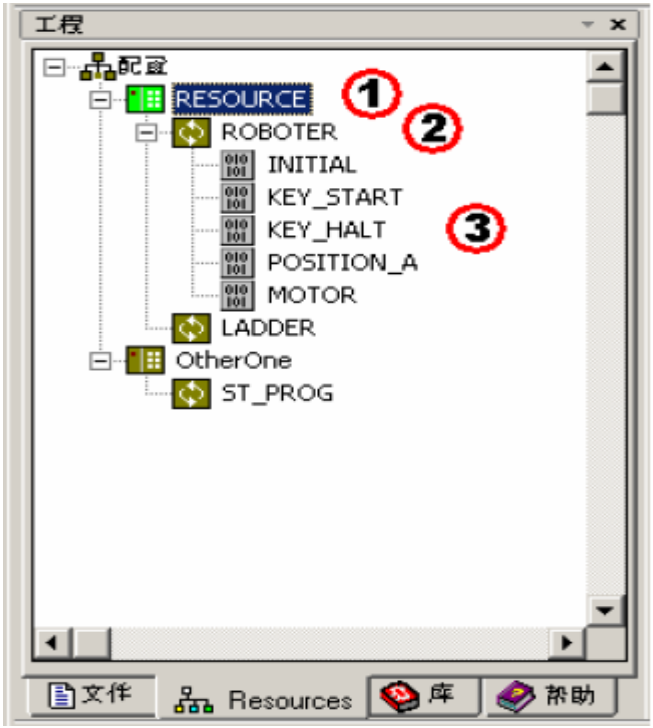


图2.2.2-2

资源窗格包含名为“配置”的实例树。它将显示：你的控制器作为资源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

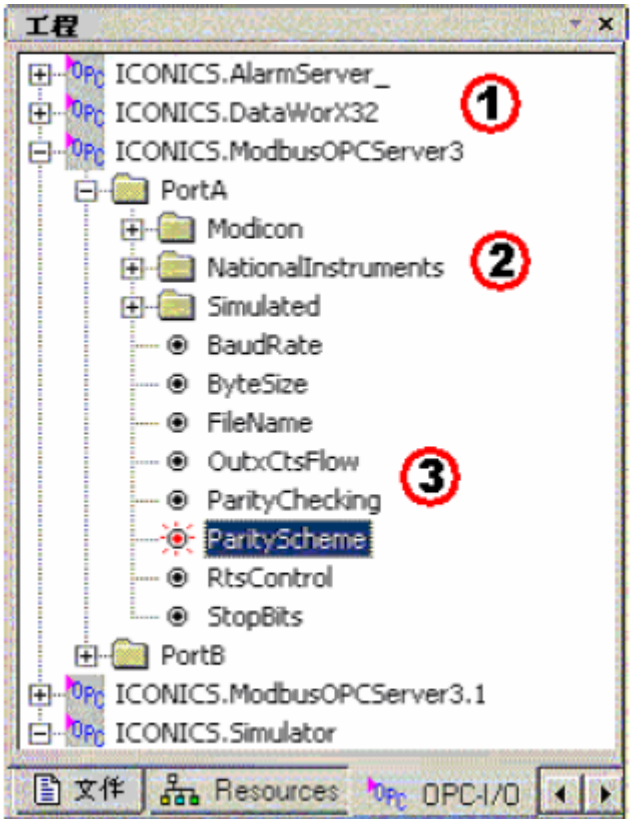


图2.2.2-3

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，你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，你可以通过"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。你可以通过文件→库→安装新的

图2.2.2-4

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。

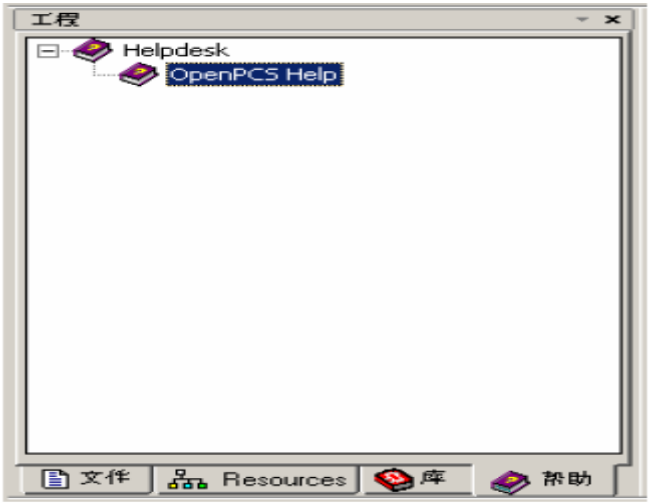


图2.2.2-5

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

如果你已经打开OptiSYS，你可以开始工作了。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）OptiSYS 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，推荐把你的应用程序与OptiSYS 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）一个名字跟你的工程一模一样的子目录会在你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 打开工程

你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，你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”。

2.2.4 文件

2.2.4.1 建立新文件

在ControlX 框架内建立新文件。选择文件→新建可以看到许多选择：

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

在“其它”下面，你可以建立文件去包含资源全局变量，可以使用或者不使用直接硬件地址，也可以建立类型定义文件。

建立完文件后，通过在文件窗格上简单的双击，你可以运行编辑器编辑任何这些文件。

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OptiSYS 所使用的PLC 的属性；还有连接名，也就是关于OptiSYS 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，OptiSYS 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”。

图2.2.5-1

点击“确定”，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，你可以改变下面的属性：

在“硬件模式”下面，选择你使用的控制器的相应配置文件。你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你的目标。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。网络连接项选择为“仿真”，用于与OptiSYS 的PLC仿真器工作。

选中“上载”，将你应用程序的资源打包到目标。如果你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，会在你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OptiSYS 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，请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，首先标记你要创建任务的程序，选择*PLC* 连接到资源。在增加了任务之后，你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，你必须详细指明信息，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OptiSYS 工程，可能会有许多“资源”，要想最好地利用多资源可以查看高级主题部分。然而，为了使OptiSYS 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、在线、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，你不必规定哪个资源去使用这些命令。如果你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 OPC - I/O

2.2.6.1 OPC - I/O: 介绍

使用OPC-I/O 浏览器窗格，你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。

这样在你的PLC 程序中你能把任意一个由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OptiSYS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。

OptiSYS 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编译所有的任务，选择*PLC→*重新生成当前资源。

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，双击你想上线的资源，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，OptiSYS 会自动的给我们提示。只要你愿意，你可以通过使用*PLC→PC→PLC（*下载*)*在任何时候调用一个下载。

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。

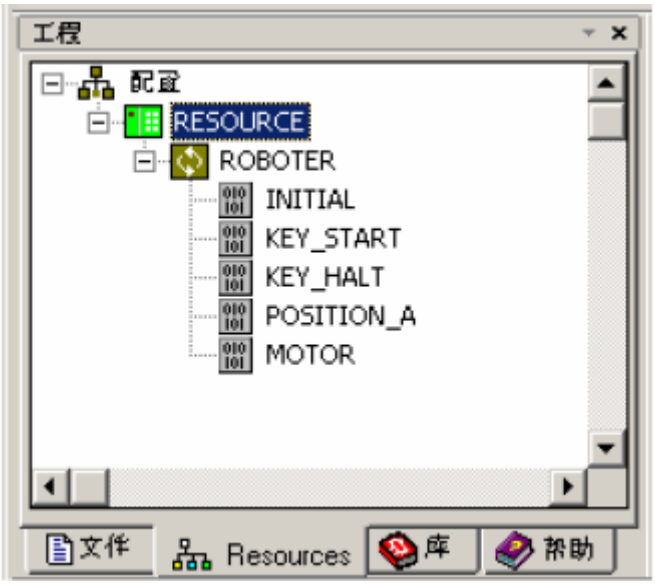


图2.2.8-1

2.2.8.4 启动在线编辑器

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，定位你要监控的实例然后双击它。

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。你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，你可以显示资源的信息。

2.2.8.6 上载

OptiSYS 支持从控制器上上载工程。为了使能这个特性，在编译一个资源时必须选中 “使能上载”选择框。

要从任何的控制器上载，使用PLC→PC<- PLC。

注：PCS-300暂不支持上载功能。

2.2.9 其它浏览器特征

2.2.9.1 资源全局变量

在OptiSYS，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→其它→全局变量或者文件

→新建→其它→直接全局变量。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OptiSYS 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你自己单独的文件。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

OptiSYS 允许增加任何类型的文件到OptiSYS 工程。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。那里你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。

2.2.9.4 浏览器选项

通过其它→浏览器选项，设置浏览器选项：

“ 隐藏文件类型”：如果检查栏是满的，浏览器只显示OptiSYS 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’如果检查栏是满的，所有首字符是‘_’的变量都隐藏在资源树中。

‘ 使能*OPC* 浏览器面板’这个检查栏用来隐藏或显示OPC 浏览器窗格。

2.2.9.5 自定义工具

浏览器在工具栏处(锤子的符号)提供了一个按钮用于启动一个OEM 特制工具。

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

2.2.9.6 交叉参照（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。

图2.2.9-1

在显示的对话框中，你选择的变量的名字和类型位于左上方。在下面是你使用的变量在程序中的所有位置的列表。双击任意一行可以打开相应的文件然后移动光标到各自的行。

如果你选择的变量在许多位置使用，使用右上方的下拉框查找它们。

第三章 硬件组态

为了让 PLC 程序能正常运行，在编写代码之前，要对 PLC 硬件作正确的设置。图 3.2-1 所示为硬件配置程序界面。

图 3.2-1

接下去，我们按一般程序设计步骤来对硬件配置作进一步的说明。

1. 创建控制器

图 3.2.1-1

打开硬件配置程序，首先需要新建一个控制器，点击菜单“文件》新建”，弹出图 3.2.1-1 所示窗口。按提示输入所需创建的 PLC 的 IP 地址，点击“确定”，如图 3.2.1-2 所示，即生成了一个控制器。如果有多个 PLC，可继续同样方法添加。注意，地址相同的控制器组态在同一个网络中是不允许的。

如果 PLC 硬件已准备完善，其处于在线状态，我们可以通过网络搜索在添加一个控制器。右键点击“网络”图标，选择“自定义搜索”，按提示输入要搜索的控制器的 IP 地址，点击“确定”，即生成了一个控制器。如果选择“搜索网络”，程

序会搜索整个网络，出现如图 3.2.1-3 所示进度窗口。程序会列出所在与计算机处于同一网络的在线 PLC。

图 3.2.1-2

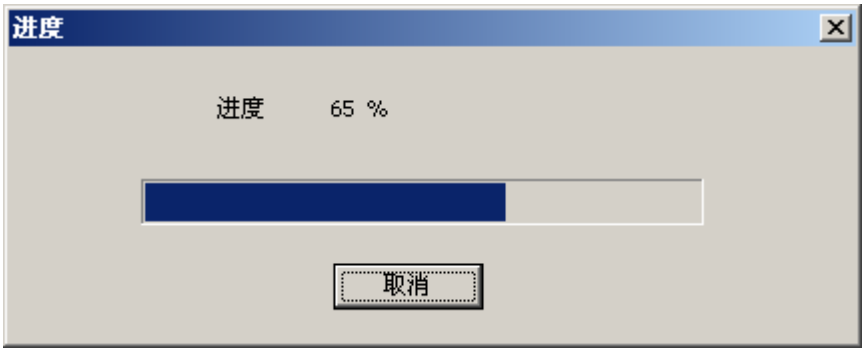


图 3.2.1-3

2. 网络

图 3.2.2-1

点击“网络”页，可看到如图 3.2.2-1 所示网络参数，包括 IP 地址、子网掩码、默认网关、MAC 地址和网络标识。所有参数以灰色显示。选择“更改”，所有参数以可写方式显示，您可以根据您的网络组成方式填写相应的参数。更改完成，点击“应用”，确认更改的参数。如果 PLC 在线，点击“远程操作”栏的“保存”，将参数下载到相应 PLC。点击“远程重启”，PLC 以新的设置参数运行。如果 IP 地址作更改，需重新添加控制器。

3. 时间

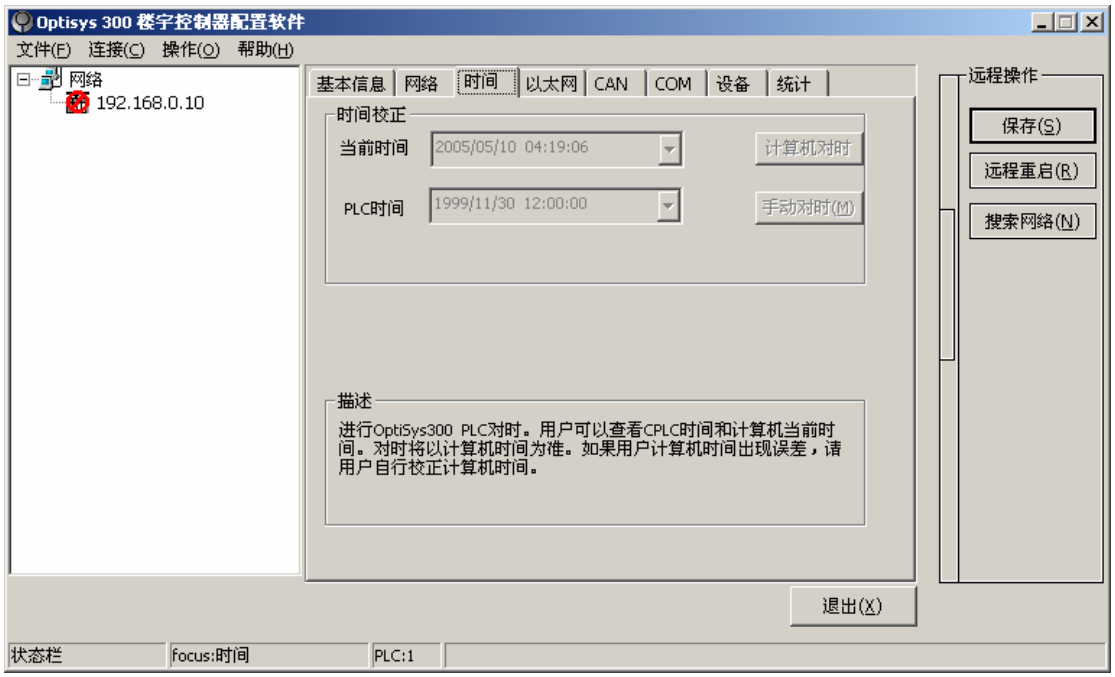


图 3.2.3-1

点击并打开“时间”页面，如图 3.2.3-1 所示。此页面用于在线 PLC 的时间设定，如果 PLC 处于离线状态，将灰色显示。“当前时间”显示的是计算机的当前时间，“PLC 时间”显示 PLC 的当前时间。PLC 时间设定提供两种方式：计算机对时和手动对时。点击“计算机对时”，将 PLC 时间设为与计算机当前同步的时间。点击“手动对时”，弹出图 3.2.3-2 所示时间输入对话框，您可输入指定的时间，点击“确定”，PLC 将以输入的时间开始计时。

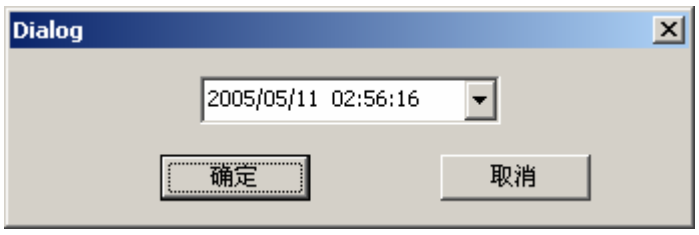


图 3.2.3-2

4. 以太网设置

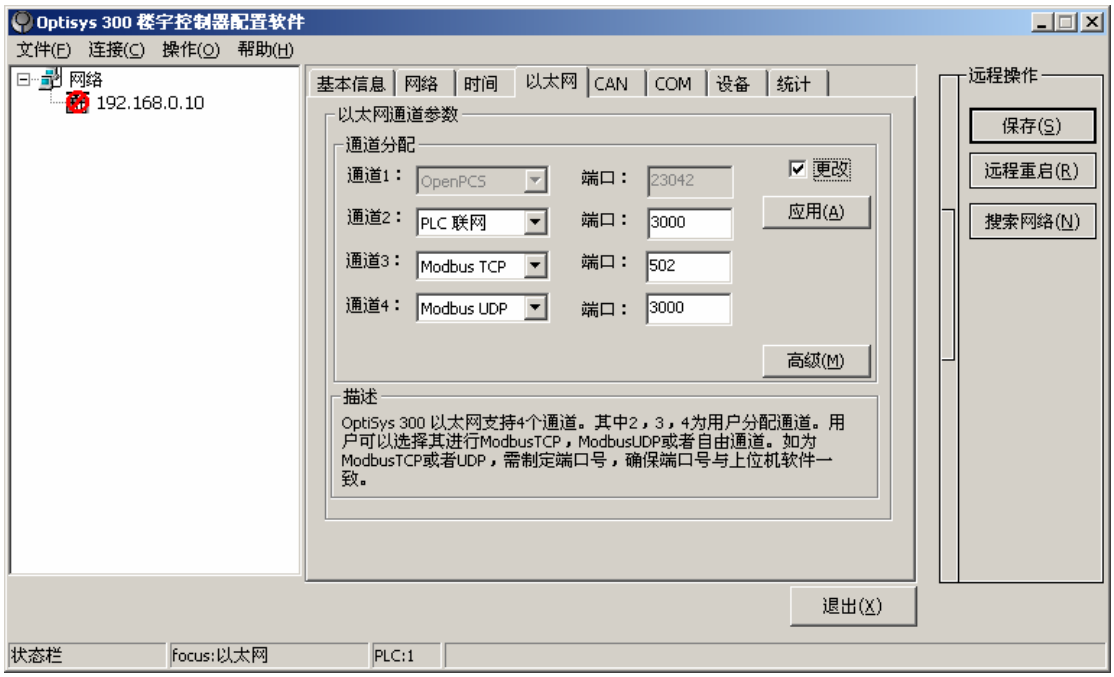


图 3.2.4-1

打开“以太网”页面。Optisys PCS300 以太网支持 4 个通道。其中第一个通道为编程软件编程通道，其端口固定不能修改。第 2、3、4 为用户分配通道。用户可以选择其进行 ModbusTCP、ModbusUDP 或自由通道。如为 ModbusTCP 或 UDP，需指定端口号，确保端口号与上位机软件一致。

5. CAN 设置

图 3.2.5-1

打开 CAN 页面。CAN1 是 CPU 与模块通讯通道，可支持 20K 至 1M 的波特率。CAN2 是 CPU 与其它支持 CAN 协议设备通讯用的通道，同样也支持 20K 至 1M 的波特率。

6. COM 设置

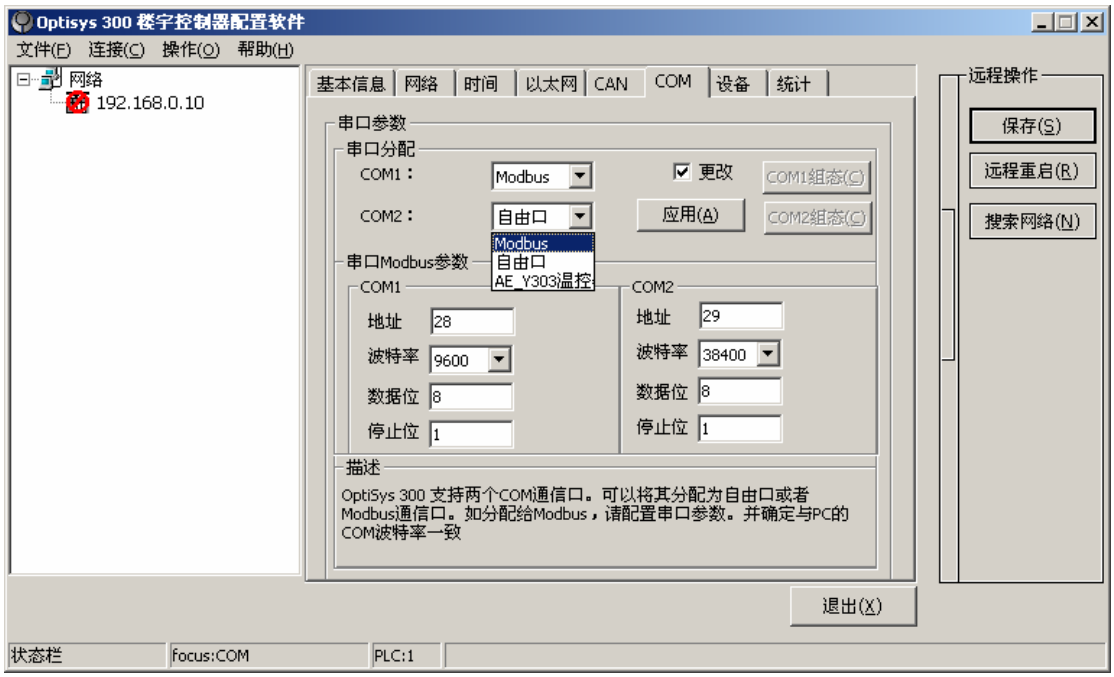


图 3.2.6-1

CPU 带两个独立串行通讯端口。COM1 提供 RS232 通讯信号，COM2 提供 RS485 信号。两个端口支持三种通讯方式：

a) Modbus 通讯：PLC 支持标准的 Modbus 通讯，支持 Modbus 协议的上位机或者控制面版可通过这种通讯方式来访问 PLC 的内部数据；

b) 自由口通讯：PLC 支持串行口的自由编程通讯，具体编程在第八章《通讯》有详细说明；

c) 现场设备组态：这种通讯方式可以将一些常用的设备（如 AEY_303 温控器）以 I/O 组态的方式挂到通讯端口上，通过操作 PLC 内部存储器地址映射来达到对现场设备的操作。

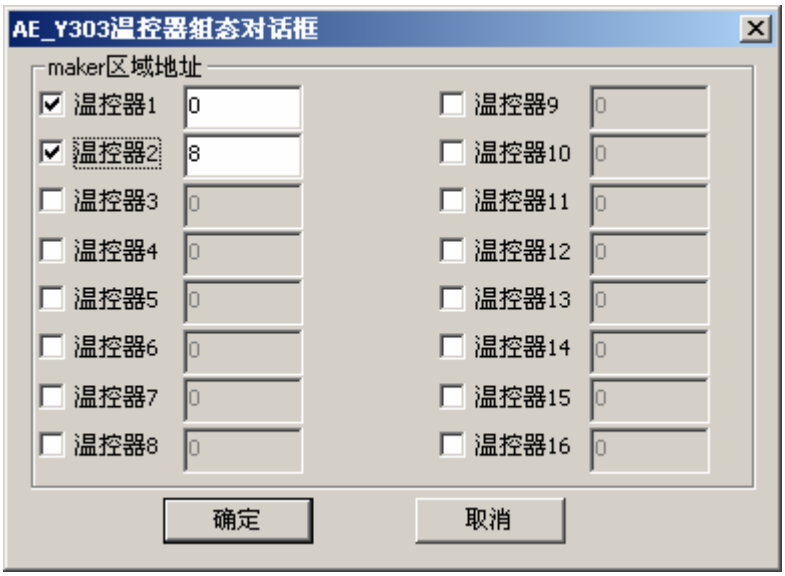


图 3.2.6-2

如图 3.2.6-2 所示，这里组态了两个温控器，其地址分别为 1 和 2，PLC 内部映射地址分别为 m0.0 和 m8.0 起始的存储区。

7. 设备组态

图 3.2.7-1

打开“设备”页面。鼠标右键点击空白表格，选择“添加设备”，弹出如图 3.2.7-2 所示“离线配置”对话框。选择正确的模块地址，选择正确的模块类型，

点击“确定”，设备表格中就添加了一个设备。一个模块显示有地址、类型、产品编号、OpenPCS 变量和 Modbus 地址这五项内容，以方便编程。对于模拟量输入输出模块，在模块添加的同时，需要对其每一通道的功能作相应设置，具体选何种通道类型需根据具体现场情况而定。

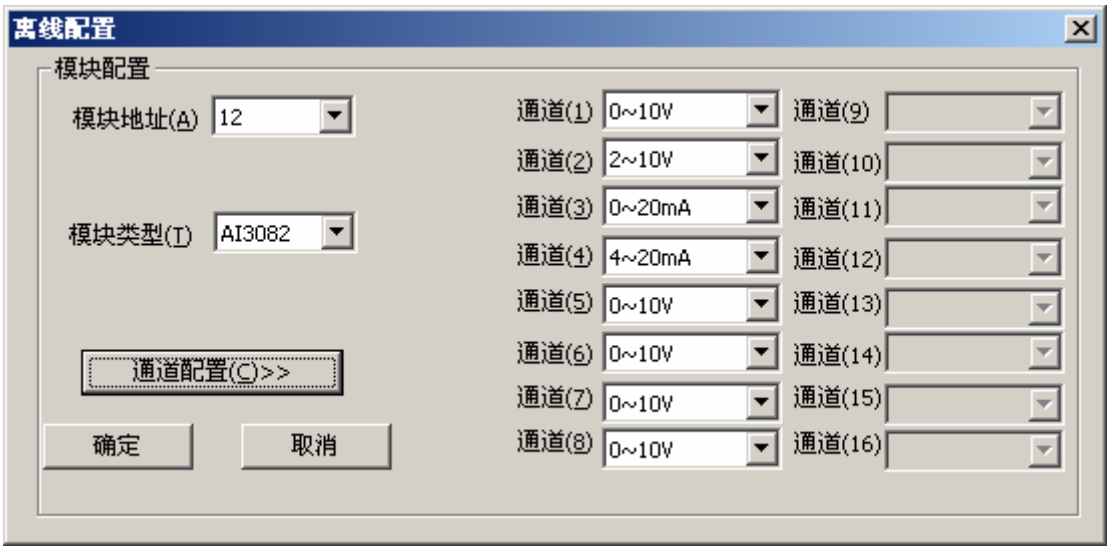


图 3.2.7-2

PLC 还提供在线模块检测功能，点击图 3.2.7-1 所示“上线”，弹出如图 3.2.7-3 所示对话框。PLC 将自动检测其在线设备和设备的 CAN 地址，并且在设备在线浏览器中显示。

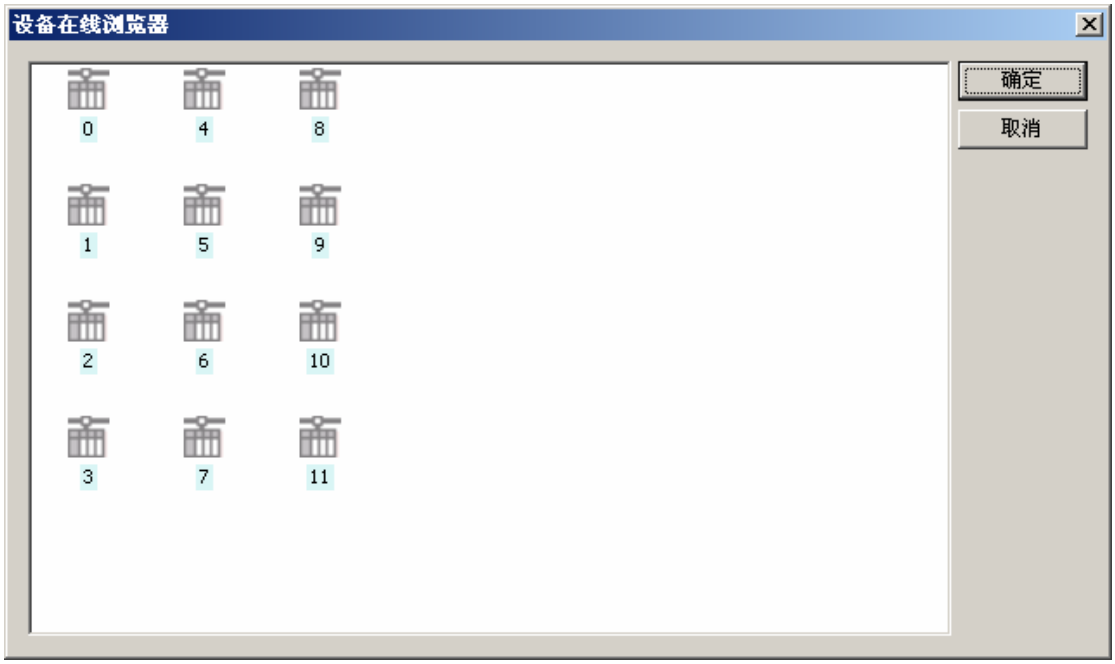


图 3.2.7-3

8. 网络统计信息

图 3.2.8-1

统计页面显示当前 PLC 的各个通道的连接状态，包括 PLC 的 IP 地址、连接速度、数据流量统计、UDP 连接个数和 TCP 连接个数。

所有配置做完后，可点击菜单“文件”—>“保存”，可取相应文件名来保存该控制器的配置文件。在今后的维护应用中，可以打开该文件，获取该控制器硬件配置信息。

第四章 变量定义

1. 声明编辑器：介绍

声明编辑器是在ControlX 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. 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结束(比如，VAR_GLOBAL END_VAR)。

VAR_INPUT：如果一个变量块只需读入一个POU，你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。

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。

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</P>	0 到 4.294.967.295	32
REAL	实型	+/-3，4E+/-38	32
TIME	时间段		4
STRING	字符串		字符长度加 2 字节
BYTE	8 位序列		8
WORD	16 位序列		16
DWORD	32 位序列		32

OptiSYS 不执行下面由 IEC61131-3 定义的数据类型：
DATE: FONT style="font-family:'宋体'; " >日，月，年。
TIME_OF_DAY: FONT style="font-family:'宋体'; " >日期时间。
DATE_AND_TIME: FONT style="font-family:'宋体'; " >日期和时间。

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。OptiSYS

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。

OptiSYS 支持下面表示的变量地址：

I：数字输入

Q：数字输出

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

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你自己定义。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。

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
```

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
```

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
```

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
```

第五章 程序的编写

1. IL 编辑器

1.1 IL 编辑器：介绍

IL 编辑器在ControlX 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：

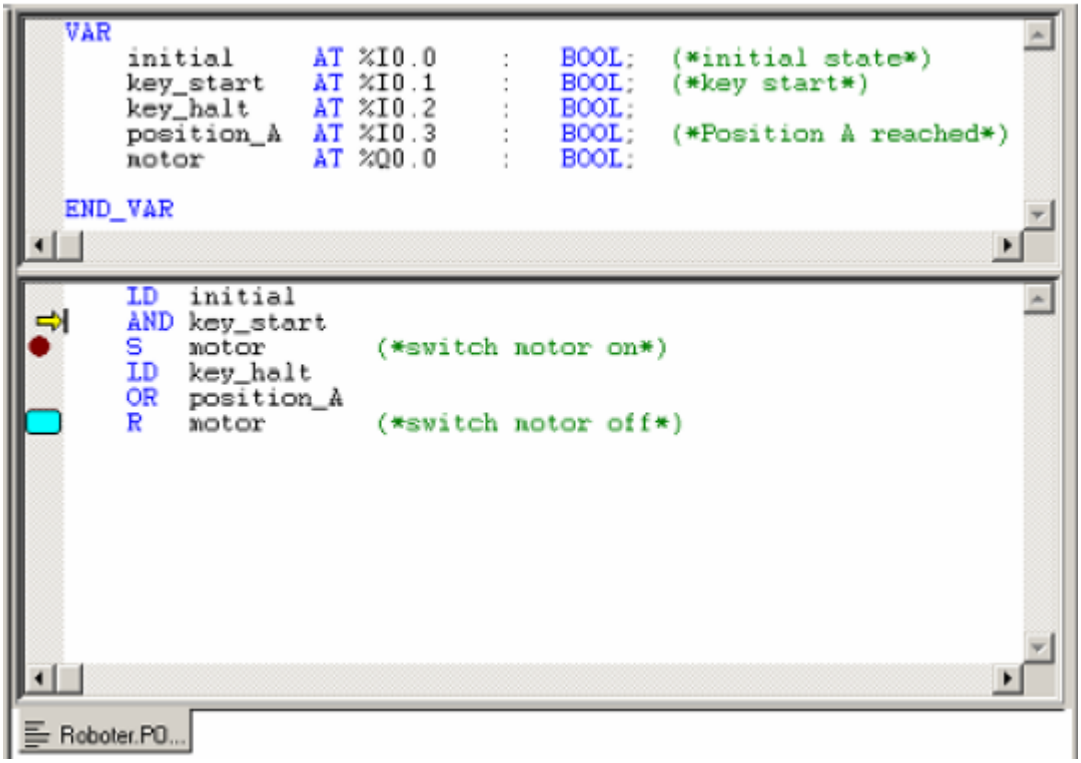


图5.1.1-1

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。

1.2 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1>:] <操作符> <操作数1> [,<操作数2>,<操作数3>,匽[(*<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。

举例定时器：

```
var
    timer1 : ton;
    timer1_run : bool;
    timer1_time : time;
    timer1_Q : bool;
    timer1_ET : time;
end_var
ld timer1_Q
stn timer1_run
cal timer1(in:=timer1_run,pt:=t#2s | timer1_Q:=Q,timer1_ET:=ET)
```

1.3 IL 的指令

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。

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。

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。

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。

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。

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。

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。

这样，这个指令被视为一个指令组成的序列。

跳转仅允许在一个 POU。

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。

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。

CAL

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。

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。

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。

RET

"RET"指令无条件的跳转返回到调用的 POU 上。一若这个 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。

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。

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。

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

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

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

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

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

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。

ADD

输入

In1: ANY_NUM

In2: ANY_NUM

返回

ANY_NUM

两数之和

SUB

输入

In1: ANY_NUM

In2: ANY_NUM

返回

ANY_NUM In1-In2 之差

两数相减

MUL

输入

In1: ANY_NUM 被乘数

In2: ANY_NUM 乘数

返回

ANY_NUM 乘积

两数相乘

DIV

输入

In1: ANY_NUM 被除数

In2: ANY_NUM 除数

返回

ANY_NUM 商

两数相除

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

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

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

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

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

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

1.4 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。

首先确保你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，你可以容易的判断：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。使用这个办法在你检验应用程序的其它部分时还可以密切留意各个变量。

2. ST 编辑器

2.1 ST 编辑器：介绍

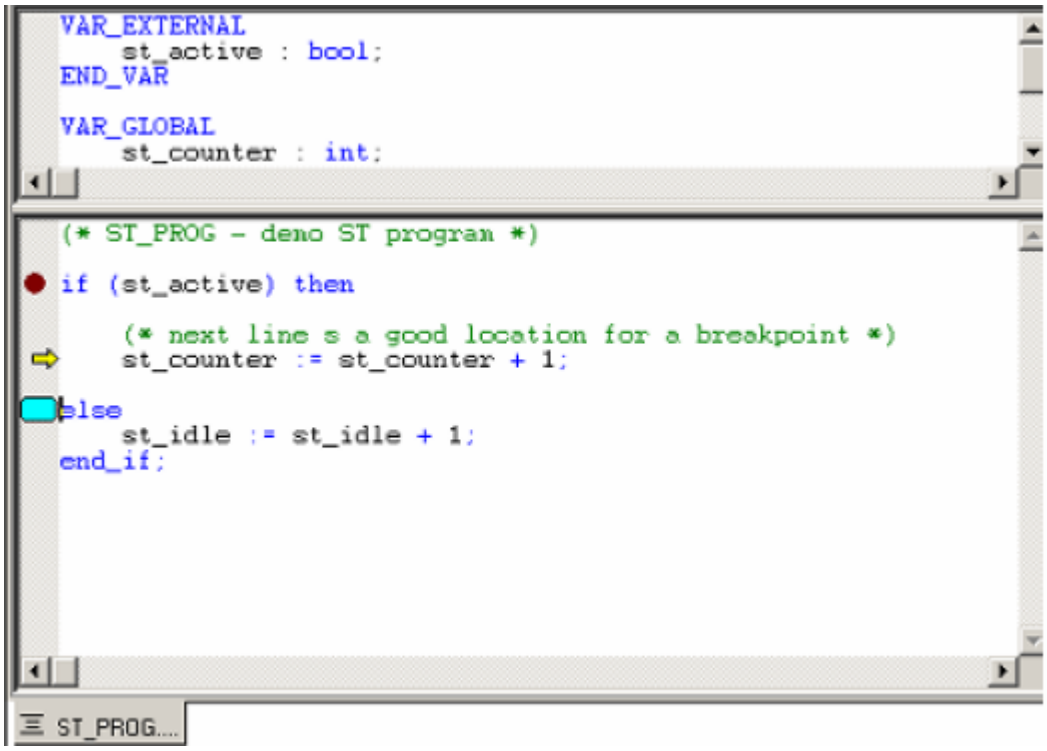


图5.2.2-1

ST 编辑器在ControlX 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。

2.2 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。

指令：

:= (Assignment)

赋值把表达式的结果赋给一个变量。.

举例：

```
VAR
a: INT;
  b: array[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

BY

见 FOR

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”。

DO

见 FOR 和 WHILE

ELSE

见 CASE 和 IF

ELSIF

见 IF

END_CASE

见 CASE

END_FOR

见 FOR

END_IF

见 IF

END_REPEAT

见 REPEAT

END_WHILE

见 WHILE

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。

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

重复第二步

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;
```

OF

见 CASE

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。

RETURN

The 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;
```

UNTIL

见 REPEAT

WHILE

当所给表达式为“true”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 true，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。

2.3 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

2.4 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，因此如果你不用注释，你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*)

注意:注释不可多重使用。

2.5 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。

首先确保你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，你可以容易的判断：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。使用这个办法在你检验应用程序的其它部分时还可以密切留意各个变量。

3. 梯形图编辑器

3.1 梯形图编辑器：介绍

梯形图编辑器在ControlX 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。

3.2 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)

功能块和函数

跳转(控制流的图形元素)

3.3 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。

3.4 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)。

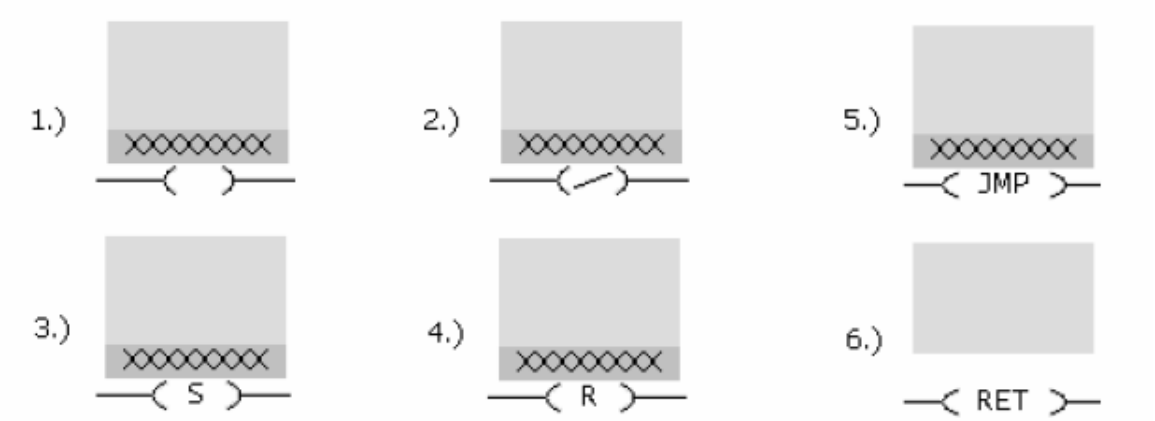
线圈：在网络中线圈把值赋给输出变量。一个线圈把它的左边的连接状态拷

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。

3.5 线圈

输出变量总是位于网络右侧，且和右侧轨线相连。



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”；

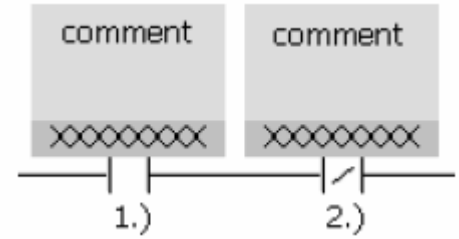
相反，若逻辑连接结果为“0”，将无输出。

(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。

3.6 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。

3.7 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。

3.8 在线梯形图编辑器

首先确保你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。

当你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″-!-″。

4. CFC 编辑器

CFC 编辑器介绍

CFC(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IEC61131 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)。

4.1 使用块工作

要增加块到你的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。点击你要插入新块的图，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。

4.2 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。

OptiSYS 也支持多连接。

4.3 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”。

图5.4.3-1

在名称中，键入对象的名字。它必须是有效的IEC61131-3 变量名。

如果你想让CFC 编辑器为这个空白栏对象声明一个变量，选择IEC611131-3变量。否则你如果选择“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，你可以选择一个物理地址或初始

值。

对于CFC 连接器，你可以选择“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。

4.4 CFC 在线编辑器

首先确保你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。当你在在线模式下启动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″-!-″。

4.5 高级 CFC 主题

a) 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。

b) 打印CFC 图

CFC 编辑器提供打印的几种可能性。使用文件→打印图的当前页，使用文件→全部打印已装载的图的所有页。

c) 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。

d) 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。

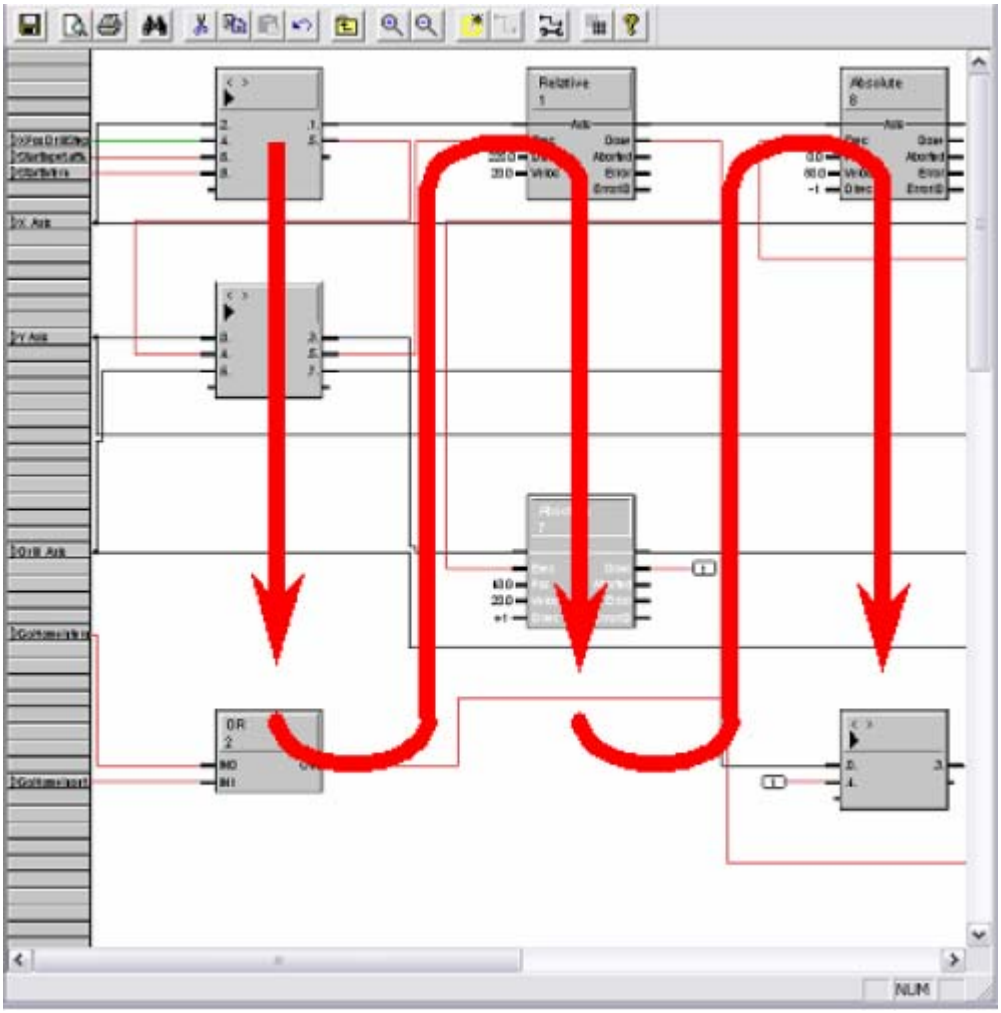


图5.4.5-1

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。

e） 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。你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。

f） 在CFC 中找错误

如果你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。

4.6 复合块

a）复合块：介绍

复合块是构建你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，你可以确定水平方向放置的块的数量。在垂直方向，你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，你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，从别的块上你可以很容易的区分复合块。

b） 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. 在你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。

当你在图上用完空间时，或通过高级分组增加程序的可读性，你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 编辑器将提示你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。

c） 对一个复合块增加一个输入或输出

你可以像别的功能图一样编辑复合块的内容。当你需要提供额外的输入和输出时，你需要改变复合块的接口。你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. 当你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. 双击你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（取决于你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。

如果你点击一个层次上的图标，你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。

5. SFC 编辑器

5.1 SFC：介绍

SFC 编辑器在ControlX 框架之中。在SFC 编辑器的上部，输入POU 的声明。

5.2 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：

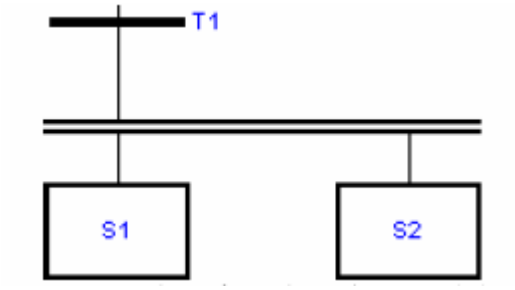
步：每一步包含了许多动作，动作包含代码片段。一个被执行的步，称之为激活。如果激活一个步，包括的动作将被执行。

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。

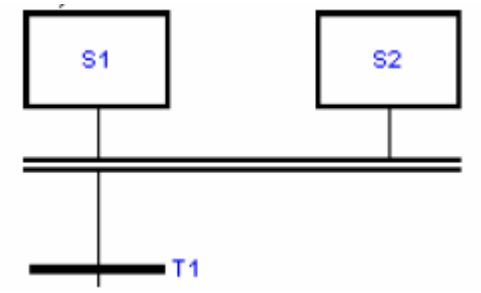
初始化步：在程序的开头激活初始化步。在程序前面执行开始处可以被初始化步标识。

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

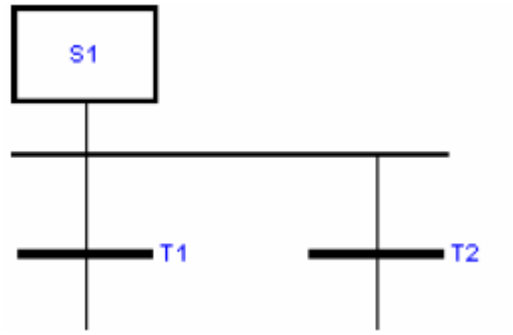
激活先前所有的步。一旦状态转换，解除所有以前的步的激活，并且激活所有紧接着的步。



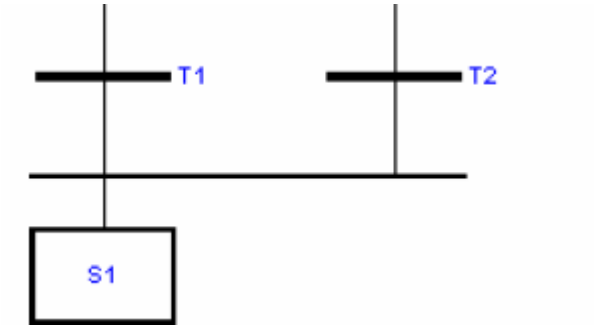
同步顺序：在一个平行链上，一个转换可以同时激活多个步。如果T1 转换的所有先前的步已经处于激活状态，并且转换条件为真，就会激活所有紧接的同步顺序(比如S1， S2)。



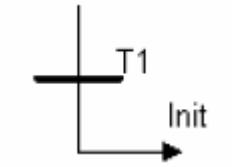
同步顺序的汇集：同步顺序的路径重新聚集在一起。如果所有先前的步处于激活状态(如S1，S2)，在转换T1 为真时，解除先前步的激活并激活紧跟的下一步。



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。

5.3 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IL 程序的入口点是IL 的第一个元素。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，你可以在那个跳转主题处获得更多有关信息。

5.4 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：（*当温度大于70 度时转移关闭*）

```
LD variable of temperature
GT 70
```

5.5 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。

5.6 SFC 在线编辑器

首先确保你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，你可以容易的判断：如果是实例树中的代码，鼠标光标应该是活动的。

当你启动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，你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。

5.7 选择元素

2.9.7.1 标记单个元素

单击左键标记出一个元素。

使用方向键(←,→, ↑, ↓)，可以移动单个元素到邻近的元素。

2.9.7.2 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。

2.9.7.3 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。

5.8 SFC 高级主题

2.9.8.1 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!

2.9.8.2 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OptiSYS 系统中编译时记录下来的。

2.9.8.3 使用不同于IL 的其它语言

默认情况下，SFC 是用指令列表来写步骤和转换过程代码的。用别的语言（梯形图， CFC，ST），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。

记住要在你的SFC 程序的声明部分声明一个功能块实例。

根据你的应用程序的需要，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，这将不仅使你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。

第六章 功能块

1. 定时器和计数器

1.1 OFF 延迟定时器（TOF）

若输入状态为“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。

计时器运行时，输入“IN”变为 0，将不起作用，直至“Q”状态变为 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“1”。若“IN”状态变为 0。“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

1.2 ON 延迟定时器（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。计时器运行后，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”为当前时间值。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”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

1.3 脉冲定时器（TP）

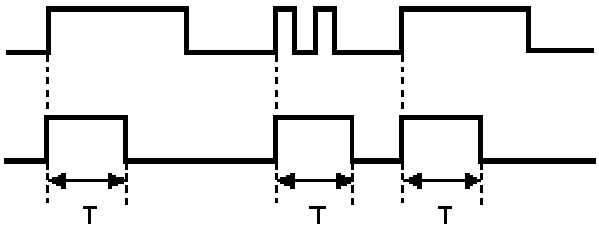
输入 IN 有一个上升沿将启动时间函数“TP”，运行时间由“PT”决定。

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，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

1.4 倒计数（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

1.5 正计数（CTU）

功能块 CTU 以加法计算输入“CD”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 CD 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

PV ：整型计数值上限

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

1.6 双向计数（CTUD

功能块“CTUD”计数上升沿和下降脉冲。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

2. 数值运算

2.1 加（ADD）

输入

In1：ANY_NUM

In2：ANY_NUM

返回

OUT：ANY_NUM

In1+In2 之和

2.2 减（SUB）

输入

In1: ANY_NUM

In2: ANY_NUM

返回

ANY_NUM

In1-In2 之差

2.3 乘（MUL）

输入

In1: ANY_NUM 被乘数

In2: ANY_NUM 乘数

返回

ANY_NUM 乘积

2.4 除（DIV）

输入

In1: ANY_NUM 被除数

In2: ANY_NUM 除数

返回

ANY_NUM 商

2.5 绝对值（ABS）

输入

In: ANY_NUM

返回

ANY_NUM

返回输入的绝对值

2.6 反余弦（ACOS ）

输入

In: REAL

返回

REAL:

返回输入的反余弦值

2.7 反正弦（ASIN）

输入

In: REAL

返回

REAL: 输入反正弦

2.8 反正切（ATAN）

输入

In: REAL

返回

REAL: 输入的反正切值

2.9 余弦（COS）

输入

In: REAL

返回

REAL: 输入的余弦值

2.10 幂（EXP）

输入

In: 实型

返回

实型: e 的 In 次幂

2.11 自然对数（LN）

输入

In: REAL

返回

REAL: 以 e 为底的对数

2.12 对数（LOG）

输入

In: REAL

返回

REAL: 以 10 为底的对数

2.13 模（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。

2.14 取非（NEG）

输入

In: ANY_NUM

返回

ANY_NUM: 输入值的非运算

2.15 正弦（SIN）

输入

In: REAL

返回

REAL: 输入的正弦值

2.16 平方根（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

2.17 取整（TRUNC）

输入

In: REAL

返回

ANY_INT

返回实数的整数部分。

3. 数值传递

Move

输入

In: ANY

输出

Out: ANY

函数"MOVE" 是个赋值运算的函数。

在梯形图中, DIVs 对整形和非整形型数据有效.

函数必须有一个 EN 输入 和一个 ENO 输出控制函数的执行。If "EN" 为 FALSE, 函数不执行且 "ENO" 置为 FALSE. "ENO" 也可用于函数错误状态。这两个参数必须直接或通过子网络和电流轨线相连。

4. 类型转换

编程软件提供 BOOL、BYTE、DINT、DWORD、INT、REAL、SINT、STRING、TIME、UDINT、UINT、USINT、WORD 十三种数据类型的相互转换。

输入

In: ANY

输出

Out: ANY

5. 逻辑功能块

5.1 与（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

5.2 与非（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

5.3 非（NOT）

输入

IN1: ANYBIT 输入

返回

ANYBIT 输入的逻辑非运算

5.4 或（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

5.5 或非（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

5.6 异或（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

5.7 异或非（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。

6. 比较

6.1 等于（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

6.2 大于等于（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

6.3 大于（GT）

输入

IN1: ANY 输入 1

IN2: ANY 输入 2

返回

布尔型 输入 1 大于输入 2 时为真

6.4 小于等于（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

6.5 小于（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

6.6 最大值（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

6.7 最小值（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。

7. 移位操作

7.1 左移（SHL）

输入

IN: ANY_BIT 位 形式

N: UINT 左移位数

返回

ANY_BIT IN, 左移 N 位

最左边的位用零补

7.2 右移（SHR）

输入

IN: ANY_BIT 位 形式

N: UINT 右移位数

返回

ANY_BIT IN, 右移 N 位

最右边的位用零补

7.3 循环左移（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

7.4 循环右移（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.

8. 跳转

8.1 无条件跳转（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。

这样，这个指令被视为一个指令组成的序列。

跳转仅允许在一个 POU

8.2 条件跳转（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。

8.3 条件非跳转（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。

8.4 跳转返回（RET）

"RET"指令无条件的跳转返回到调用的 POU 上。一若这个 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。

8.5 条件返回（RETC）

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。

8.6 条件非返回（RETCN）

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

9. 其它功能块

9.1 置位 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。

9.2 复位 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。

9.3 双稳态输出（SR）

Set1: bool 设置条件

Reset: bool 重设置条件

输出

Q1: bool 双稳态元素的输出状态

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。

9.4 程序调用（CAL/CALC/CALCN）

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。

格式：

[<Label>:]CAL/CALC/CALCN<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。

9.5 字符串串连（CONCAT）

输入

In1: STRING 第一个串

In2: STRING 第二个串

返回

STRING 两字符串串连

9.6 下降沿（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”，并且上升沿指示的时候，输出为 “1”。

9.7 上升沿（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”。

9.8 字符串查找（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。

第七章 编译、下载、监控

1. 编译、下载代码

为了执行应用程序，我们首先需要编译它并且将代码下传到控制器。为了这样做，我们首先必需选择一个激活的资源。到浏览器的 Resource-Pane，用鼠标右击资源。在弹出的相关菜单中点击“Set active”激活资源。现在选择 *PLC Online* 连到 PLC。在输出窗口，你将会看到编译过程。输出的结尾看起来应该如图 7.1-2 这样：

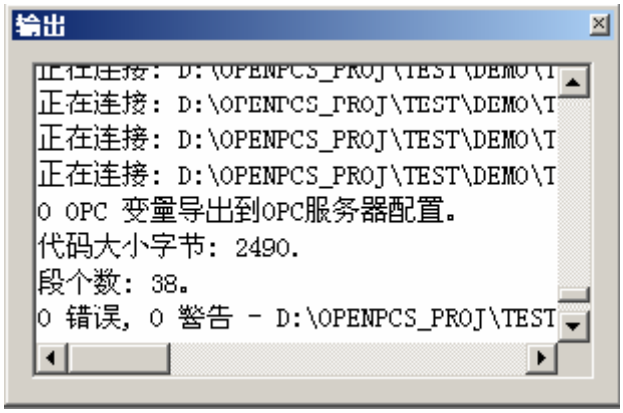


图 7.1-2

编译成功完成之后，OpenPCS 将会检测需要下传到控制器的代码。它会提示你这么做：

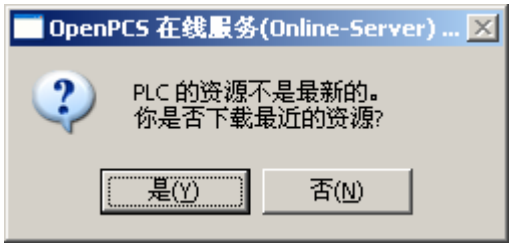


图 7.1-3

点击“是”接受。在代码传送过程你将会看到一个进程条，下载时间受程序代码大小影响。下载完毕时，OpenPCS 自动的打开另一个工具“Test and Commissioning（测试与调试）”。这表明 OpenPCS 处于在线状态。

实例路径	名称	值	类型	地址	强制	注释
TEST_CFC	TIMER1_ET	8s843ms	TIME			
TEST_CFC	TIMER1_PT	10s0ms	TIME			
TEST_CFC	FCT_10_1_TIME_T...	8843	DINT			
TEST_CFC	FCT_10_1_DINT_T...	8843	INT			
TEST_CFC	FCT_10_1_NOT_OUT	TRUE	BOOL			
TEST_CFC	FCT_10_1_ADD_OUT	8843	INT			
TEST_CFC	ADD_INT_OUT_1	8843	INT			
TEST_CFC	ADD_INT_1	0	INT			
TEST_ST	VAR_2	0	REAL	%M500.0		
TEST_ST	VAR_1	0	DWORD	%M500.0		

测试和调试

图 7.1-4

在“测试和调试”中，使用 *PLC Coldstart*（或点击工具条的蓝色箭头）启动执行你的代码。

图 7.1-5

2. 监控代码

现在你的应用程序已经在运行了，回到浏览器在你的工程中找到“Resource”。点击所有小加号打开资源入口下面的所有的树。这将展示“instance tree（实例树）”，它将程序的所有程序和功能块的实例以及程序中使用的变量全部展示出来。

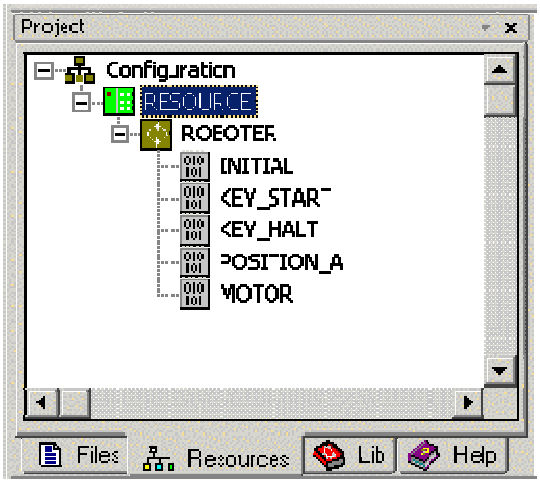


图 7.2-1

双击一些变量的入口（显示 0/1 的灰箱），可以看见相应的变量增加到 Test&Commissioning 的观察列表中。

实例路径	名称	值	类型	地址	强制	注释
TEST_CFC	TIMER1_ET	8s843ms	TIME			
TEST_CFC	TIMER1_PT	10s0ms	TIME			
TEST_CFC	FCT_10_1_TIME_T...	8843	DINT			
TEST_CFC	FCT_10_1_DINT_T...	8843	INT			
TEST_CFC	FCT_10_1_NOT_OUT	TRUE	BOOL			
TEST_CFC	FCT_10_1_ADD_OUT	8843	INT			
TEST_CFC	ADD_INT_OUT_1	8843	INT			
TEST_CFC	ADD_INT_1	0	INT			
TEST_ST	VAR_2	0	REAL	%M500.0		
TEST_ST	VAR_1	0	DWORD	%M500.0		

图 7.2-2

现在回到 Resource-Pane，在资源下面寻找你的程序入口。注意不要和列在 File-Pane 下的资源文件相混淆。双击程序中某一个程序（为避免混乱，关闭所有不在线的资源代码文件）。这将会以在线模式运行 ControlX 编辑器。当你将鼠标在 ControlX 编辑器上移动时会看到与前面不同的光标形状。移动光标到代码中的一个变量上，很短时间后，你会看到一个“tooltip(工具提示)”就像在线值显示的一样：

```
VAR
initial     AT %I0.0    :   BOOL;   (*initial state*)
key_start   AT %I0.1    :   BOOL;   (*key start*)
key_halt    AT %I0.2    :   BOOL;
position_A  AT %I0.3    :   BOOL;   (*Positicn A reached*)
motor       AT %Q0.0    :   BOOL;
END_VAR
```

```
LD   initial
AND  key_start
S    motor          (*switch motor on*)
LD   key_hal
OR   pos
R    motor          (*switch motor off*)
```

BOOL (AT%Q0.0) motor = FALSE

ROBOTER....

图 7.2-3

移动鼠标指向不同的变量，检验它们的值。如果应用程序修改了变量，显示会自动地更新。

如果你需要分析你的代码的逻辑，仅仅值的显示可能是不够的。移动指针到包含所分析代码的那一行，单击鼠标。现在按下 F9 在那一行设置断点，你立马在那一行看到标志断点的一个红点。然后，你会注意到一个黄色的箭头，它用于鉴别当前指令的指针。ControlX 编辑器将会在输出窗口显示“Breakpont reached（到达断点）”。

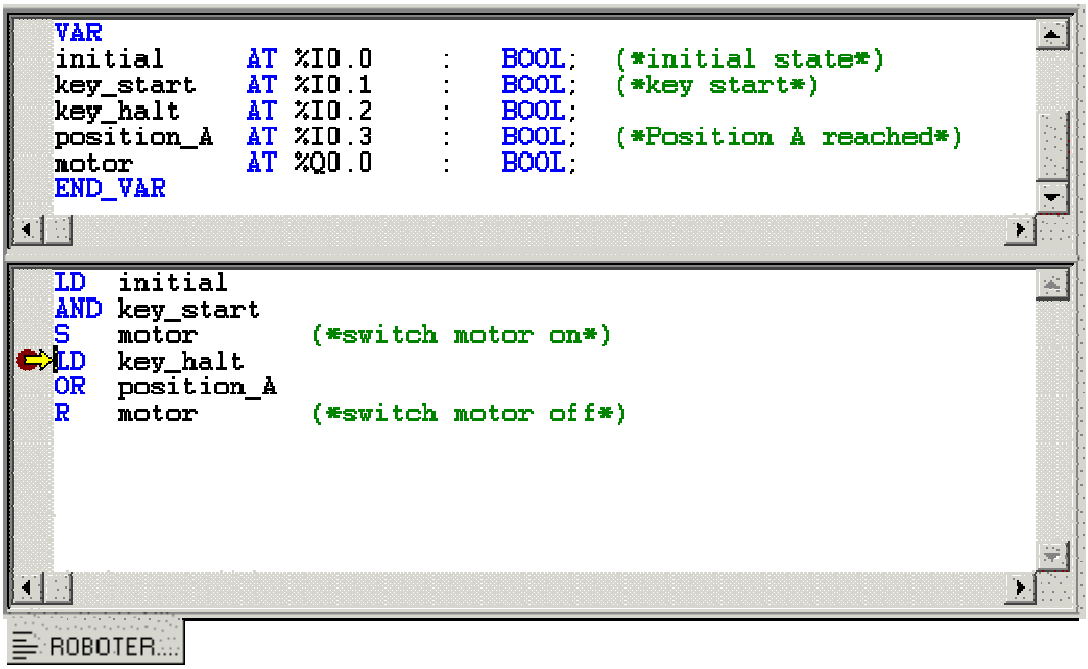


图 7.2-4

即使控制器在断点处已经停止，你依然可以移动鼠标指针去检验变量值。按下 F10 执行单步，或按下 F5 继续执行程序。在包含断点的那一行，再次按下 F9 删除断点。

注意：如果在你设置的断点处程序没有停止，你可能没有正确设置优化设置属性。请确保你的资源设定为“size only”。

3. 编译错误和警告信息

3.1 语法错误

7.3.1.1 S1000

Nested comments are not allowed.

你正使用的是一个与 IEC 61131-3 兼容的版本，在这个版本中不允许嵌套的注释。

7.3.1.2 S1001

Invalid character.

用了一个不支持的字符。参看表 1: 字符集特性

7.3.1.3 S1002

End of file found in comment.

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。

7.3.1.4 S1003

Reserved keyword.

标识符用了保留的关键字。

7.3.1.5 S1004

Invalid value for hour.

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。

7.3.1.6 S1005

Invalid value for minute.

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。

7.3.1.7 S1006

Invalid value for second.

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。

7.3.1.8 S1008

Invalid value for month.

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。

7.3.1.9 S1009

Invalid day range.

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。

7.3.1.10 S1010

Exponent too large.

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。

7.3.1.11 S1011

Incorrect direct address.

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。请查看你的硬件资料，确定分级地址的每个域的最大值。

7.3.1.12 S1012

Invalid day entry.

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。

7.3.1.13 S1013

Invalid hour entry.

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。

7.3.1.14 S1014

Invalid minutes entry.

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.

7.3.1.15 S1015

Invalid seconds entry.

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.

7.3.1.16 S1016

Invalid milliseconds entry.

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.

7.3.1.17 S1017

Direct address too complex.

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。请参考你的用户手册，

确定分级地址的深度。

7.3.1.18 S1018

Integer constant too large/small.

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。

7.3.1.19 S1019

Integer constant too large/small (does not fit into 32 bits).

给定的常数值超过了 DINT/UDINT 类型范围。

7.3.1.20 S1020

Numeric value too large/small.

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。

7.3.1.21 S1021

处理一个浮点型数学函数时出错。

7.3.1.22 S1022

Invalid string constant.

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。

7.3.1.23 S1023

Invalid number (i.e., numerical constant).

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。

7.3.1.24 S1024

Invalid constant.

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。

7.3.1.25 S1025

Invalid direct address.

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。请参看你的硬件资料，确定最大的地址层次数。

7.3.1.26 S1026

Invalid identifier (name, variable, parameter,...)

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。

7.3.1.27 S1027

End of file found in file header.

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，请联系你的生产厂家。

7.3.1.28 S1028

This identifier is too long (> 64 characters).

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。

7.3.1.29 S1029

This word (identifier, constant literal, string, comment) is too long (> 1024 characters).

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。

7.3.1.30 S1030

Too many identifiers.

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。

7.3.1.31 S1031

Unallowed usage of ENO. Just allowed as an identifier for a bool variable in output section.

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。

7.3.1.32 S1032

Unallowed usage of EN. Just allowed as an identifier for a bool variable in input section.

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，如果的值是，则函数或功能块定义的操作不能 EN FALSE 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。

7.3.1.33 S3000

Function block not declared.

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。

7.3.1.34 S3001

Function not present.

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。

7.3.1.35 S3002

Incorrect parameter.

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。

7.3.1.36 S3003

Jump label not present.

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。

7.3.1.37 S3004

Multiple assignment of a variable/name.

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。

7.3.1.38 S3005

This is not a function block instance.

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。

7.3.1.39 S3006

This is not a struct variable or a function block instance.

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。

7.3.1.40 S3007

This is not a FUNCTION-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。

7.3.1.41 S3008

No structure element or block parameter.

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。

7.3.1.42 S3009

No jump label.

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。

7.3.1.43 S3010

Type or function block name expected.

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)。

7.3.1.44 S3011

Identifier is not a variable or type name.

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)。

7.3.1.45 S3012

Variable name or constant expected.

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
```

7.3.1.46 S3014

Numeric data type expected.

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。

7.3.1.47 S3016

Bit data type expected.

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。

7.3.1.48 S3017

Boolean value expected.

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。

7.3.1.49 S3018

Numeric data type expected.

不允许的操作数类型。一个 ANYNUM 数据类型的操作数是需要的。

7.3.1.50 S3019

Operators of type incompatible.

操作数和返回结果的数据类型不兼容。

7.3.1.51 S3020

Operand types incompatible.

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
(* … *)
```

7.3.1.52 S3022

Invalid operand type for this operation.

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。

7.3.1.53 S3023

Invalid operand type for this operation.

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。

7.3.1.54 S3024

Invalid operand type for this operation.

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。

7.3.1.55 S3025

Boolean result required.

返回的结果类型不匹配，返回的应该是一个 BOOL 类型。

7.3.1.56 S3026

Undeclared identifier.

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
(* … *)
```

7.3.1.57 S3028

Comparison not defined for the data type of the current result.

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
(* … *)
```

7.3.1.58 S3030

Comparison not defined for this type.

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
(* … *)
```

7.3.1.59 S3032

Self-referencing (i.e., recursive) declarations are not allowed.

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）。

7.3.1.60 S3033

Operand of type TIME expected.

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
(* … *)
```

7.3.1.61 S3034

String too long for variable.

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。

7.3.1.62 S3035

Unallowed operand type for this function! Numeric operand or operand of date or time type expected.

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
(* … *)
```

7.3.1.63 S3036

Integer constant is out of range.

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
```

7.3.1.64 S3037

The lower bound of the subrange must not be greater than the upper bound.

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。

7.3.1.65 S3038

Initialisation is out of bounds of subrange (Data type is a subrange type).

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。

7.3.1.66 S3039

Index is out of bounds.

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。

7.3.1.67 S3040

Invalid data type. ANY_NUM required.

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
(* … *)
```

7.3.1.68 S3041

Unallowed EN/ENO type. Must be of type bool. Must not be 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。

7.3.1.69 S3042

Missing EN. Use of ENO allowed only in combination with EN.

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。

7.3.1.70 S3044

Data missing. You either need a load or an expression.

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。

7.3.1.71 S3046

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

7.3.1.72 S4000

'AT%': Simultaneous declaration of several direct variables is invalid.

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

7.3.1.73 S4001

Too many variables (identifiers). Maximum is 60 identifiers.

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。

7.3.1.74 S4003

Array too big.

在定义数组时，一个维的元素个数超过了 OptiSYS 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。

7.3.1.75 S4005

Upper bound must be greater or equal than lower bound.

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。

7.3.1.76 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。)] 。

7.3.1.77 S4007

Self-referencing (i.e., recursive) declarations are invalid.

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）。

7.3.1.78 S4008

Too many attributes 'RETAIN' or 'CONSTANT'. You may use only one.

在一个变量声明部分，使用了太多的限定词。

7.3.1.79 S4009

A STRUCTure must contain at least one structure element (variable declaration).

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
```

7.3.1.80 S4010

Simultaneous type declarations are not allowed.

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
```

7.3.1.81 S4011

Valid only in PROGRAMs and there within VAR- und VAR_GLOBAL-Sections.

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。

7.3.1.82 S4012

Valid only in PROGRAMs, FUNCTION_BLOCKs, and in FUNCTIONs.

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。

7.3.1.83 S4013

Valid only in PROGRAMs, FUNCTION_BLOCKs, and in FUNCTIONs.

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。

7.3.1.84 S4014

Valid only in PROGRAMs and in FUNCTION_BLOCKs.

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。

7.3.1.85 S4015

Valid only in PROGRAMs and in FUNCTION_BLOCKs.

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。

7.3.1.86 S4016

Valid only in PROGRAMs and in FUNCTION_BLOCKs

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。

7.3.1.87 S4017

Valid only in PROGRAMs.

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。

7.3.1.88 S4018

Valid only in VAR- and in VAR_GLOBAL-Sections.

在一个不支持“CONSTANT”限定词的变量声明部分，使用了“CONSTANT”限定词。

7.3.1.89 S4019

Valid only in PROGRAMs or in FUNCTION_BLOCKs and there within VAR-, VAR_OUTPUT-, or VAR_GLOBAL-Sections).

在一个不支持“RETAIN”限定词的变量声明部分，使用了“RETAIN”限定词。

7.3.1.90 S4020

Valid only in PROGRAMs or in FUNCTION_BLOCKs and there within VAR_INPUT-Sections with Type 'BOOL' without Initialisation.

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。

7.3.1.91 S4021

Valid only within VAR_INPUT, VAR_OUTPUT, and VAR_IN_OUT-Sections .

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。

7.3.1.92 S4022

Valid only in FUNCTION_BLOCKs or FUNCTIONs and there within VAR..END_VAR-Sections without CONSTANT/RETAIN-Modifiers.

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 OptiSYS 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 OptiSYS 中你看不到这些信息。

7.3.1.93 S4023

Valid only in TYPE..END_TYPE-Sections.

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。

7.3.1.94 S4024

Valid not within VAR_EXTERNAL-Sections.

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
```

7.3.1.95 S4033

Multiple initialisation.

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
```

7.3.1.96 S4034

Invalid POU name.

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。

7.3.1.97 S4035

Invalid type for function.

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。

7.3.1.98 S4036

FUNCTIONs need at least one input parameter 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。

7.3.1.99 S5000

Wrong parameter type.

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。

7.3.1.100 S5001

Array expected. This is not an array.

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
```

7.3.1.101 S5002

This FUNCTION_BLOCK is called by CAL if EN=TRUE. CALC/CALCN are both invalid.

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。

7.3.1.102 S5003

Function block instances may not be '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。

7.3.1.103 S5004

Function blocks instances are invalid in 'FUNCTION'-POUs, STRUCTs, and in ARRAYs.

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。OptiSYS 不支持功能块实例作为 STRUCT 和 ARRAY 类型的一个成员。

7.3.1.104 S5005

Function block instances as function results are not supported.

在 OptiSYS 中不支持功能块实例作为一个函数的返回类型。

7.3.1.105 S5006

Function block instances as parameters are not supported.

功能块的参数类型在 OptiSYS 中不支持。

7.3.1.106 S5008

Expected an integer or an enum. Invalid array index.

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。

7.3.1.107 S5009

Invalid sequence beginning. Current result is empty. Use 'LD' to initialise current result.

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
```

7.3.1.108 S5010

Invalid instruction within a parentheses computation.

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.
```

7.3.1.109 S5011

ARRAYs of function block instances are invalid.

不支持功能块的数组。

7.3.1.110 S5012

Result type and operand type are incompatible.

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
FUNCTION Fun1
ST x (* 65000 不是 INT 数据类型*)
(* … *)
```

7.3.1.111 S5013

Result type and type of the first formal input parameter are incompatible.

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：

```
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
```

7.3.1.112 S5014

Wrong number of parameters.

在调用一个函数或功能块时，发现太多参数。

7.3.1.113 S5015

Invalid type for direct address.

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。

7.3.1.114 S5016

Variable is read-only. Write-access invalid.

企图写一个只能读的变量。

7.3.1.115 S5017

Variable is not a STRUCTure.

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
```

7.3.1.116 S5018

Variable is no array.

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
```

7.3.1.117 S5019

Initialization value and variable type incompatible.

初始化值类型和变量类型不匹配。

例子：

```
VAR
X : INT := 65000;
END_VAR
(* … *)
```

7.3.1.118 S5020

Too many initialisation values.

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
```

7.3.1.119 S5021

Formal parameter incorrectly declared.

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。

7.3.1.120 S5022

Multiple assignments to a parmameter in a call of a function block instance.

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
```

7.3.1.121 S5023

Too much initialisation data.

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
```

7.3.1.122 S5024

Unallowed type for this operation.

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
(* … *)

7.3.1.123 S5025

Unallowed parameter type for this function.

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
LD ‘EXAMPLE’
LEFT 3.0 (* 错误：LEFT 的第二个参数是 ANY_INT 类型*)
ST X
(* … *)

7.3.1.124 S5026

Invalid formal parameter type.

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。

7.3.1.125 S5027

Incompatible operand types.

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
(* … *)

7.3.1.126 S5028

Data type not allowed for this operation.

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
(* … *)

7.3.1.127 S5029

Invalid function block call.

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
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

7.3.1.128 S5030

Variable is write-only. Read-access invalid.

企图读一个只能写的变量。

7.3.1.129 S5031

Bit access allowed only on bit data types.

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
(* … *)

7.3.1.130 S5032

Bit position is greater than the number of bits in the selected variable.

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;
END_VAR
fVar : BOOL := FALSE;
(* 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*)

7.3.1.131 S5033

IN_OUT parameter missing. Please supply every formal IN_OUT parameter with a an actual parameter.

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
```

7.3.1.132 S5034

Invalid IN_OUT parameter. IN_OUT parameters must not be expressions or constants.

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
END_VAR
cal fbInst( InOutParam1 := IntVar1,
VAR_IN_OUT
InOutParam1 : INT;
InOutParam2 : BOOL;
END_VAR
(* 代码*)
END_FUNCTION_BLOCK

PROGRAM main
VAR
fbInst : fb1;
IntVar1 : INT;
```

```
IntVar2 : INT;
(* 代码*)
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
```

7.3.1.133 S5035

Generic data types are not allowed.

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
```

7.3.1.134 S5036

Local types are not allowed in this variable section.

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
```

7.3.1.135 S5037

Too many indices within the braces [....] of an array-access.

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

7.3.1.136 S5038

Directly represented variables are only allowed as parameters in prototypes.

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
PROGRAM main
'&x' is only allowed if x is a direct variable.
END_FUNCTION_BLOCK
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.
```

7.3.1.137 S5039

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。

7.3.1.138 S5040

Too few indices within the braces [....] of an array access.

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
```

7.3.1.139 S5041

Values of type INT24 or REAL48 are invalid in this context.

这个操作不支持这种类型。

7.3.1.140 S5042

Function block instances may not be '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。

7.3.1.141 S6002

No prototype.

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，请参看你的硬件资料。

7.3.1.142 S6004

Recursion (i.e., direct or indirect self-reference) detected.

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)。

7.3.1.143 S6005

Too many types and function blocks. For the maximum number of type definitions please consult your hardware documentation.

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。

3.2 目标代码链接消息

7.3.2.1 L10001

Variable declared twice: <Variable name>.

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：如果你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。

7.3.2.2 L10004

Unresolved external: <Variable name>.

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。

7.3.2.3 L10026

Unsupported address: <AddressDescription>.

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，请参看你的硬件资料。

7.3.2.4 L10027

Invalid hardware description: %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 OptiSYS。如果这个错误还没有解决，请参考硬件文档或者联系你的硬件生产商。

7.3.2.5 L10029

Hardware configuration error.

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 OptiSYS 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。

7.3.2.6 L10030

Invalid type for variable: %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。

7.3.2.7 L10031

Initializations of directly represented variables are not allowed.

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。

7.3.2.8 L10032

Address <AddressDescription> invalid in this context.

Invalid process image description. Please contact your manufacturer.

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。

7.3.2.9 L10033

Attribute RETAIN not supported for directly represented variables.

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。

7.3.2.10 L10034

Attribute CONST not supported for directly represented variables.

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。

7.3.2.11 L10035

Instance limit for function block <FunctionBlockName> reached.

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。请参考你的硬件资料，获得固件功能块实例的最大个数。

7.3.2.12 L10036

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。

7.3.2.13 L10063

打开一个文件%1 时发生一个错误。

7.3.2.14 L10105

Internal error while loading function or DLL: <DLL/Function-Name>.

指定的 DLL 或函数不能被装载。或者你的 OptiSYS 目录中没有包含指定名字的 DLL，或者你的 DLL 是一个无效版本的。请重新安装你的系统或查看你的硬件描述文件。

7.3.2.15 L10106

Native code compiler needed for selected optimization. Please choose another optimization or install a native code compiler.

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。如果你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。适合你硬件的本地代码编译器请询问你的生产商。

7.3.2.16 L12001

Type conflict. Type of external the variable doesn't match with type of the global variable with the same name.

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。

7.3.2.17 L12002

Readable access to this variable is not allowed: <Variable name>.

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。

7.3.2.18 L12003

Writable access to this variable is not allowed: <Variable name>.

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。

7.3.2.19 L12005

内部链接器错误。请联系你的生产厂商。

7.3.2.20 L12006

内存分配故障。没有足够的内存来执行这个操作。

7.3.2.21 L12007

No object information found for task <TaskName>. Please rebuild all.

指定任务的目标文件(<任务名>.crd)没有找到。请重建整个资源。

7.3.2.22 L12008

Interpreter stack overflow in task <TaskName>.

解释调用堆栈溢出。请减少<任务名>的调用层次的深度。

7.3.2.23 L12996

Unknown command: <Command>.

用了一个带有 ITLINK 的未知命令行。

7.3.2.24 L12997

Unkown object kind: <ObjectKindSpecification>.

发现了一个无效的目标文件。请重新编译整个资源。

7.3.2.25 L12998

Invalid object kind. Kind found/requested: <ObjectKind>.

发现了一个无效的目标文件。请重新编译整个资源。

7.3.2.26 L12999

Invalid object version found. Object version found/expected: <ObjectVersion>.

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。

7.3.2.27 L13000

Load of resource global variable information failed.

没有找到带有资源全局信息的目标文件。请重建整个资源。

7.3.2.28 L13001

No object information found for pou <pouname>

没有找到指定 POU 的目标文件(<POU 名>.obj) 。请重建整个资源。

3.3 编译消息

7.3.3.1 C10006

Data type 'REAL' is not supported.

当前硬件不支持“REAL”数据类型。要了解 OptiSYS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看你的硬件资料，得到你的硬件能支持的数据类型列表。

7.3.3.2 C10007

Data type 'DATE' is not supported.

不支持“DATE”数据类型。要了解 OptiSYS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看你的硬件资料，得到你的硬件能支持的数据类型列表。

7.3.3.3 C10008

Data type 'TIME_OF_DAY' is not supported.

不支持“TIME_OF_DAY”数据类型。要了解 OptiSYS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看你的硬件资料，得到你的硬件能支持的数据类型列表。

7.3.3.4 C10009

Data type 'STRING' is not supported.

当前硬件不支持“STRING”数据类型。要了解 OptiSYS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看你的硬件资料，得到你的硬件能支持的数据类型列表。

7.3.3.5 C10010

Data type 'DATE_AND_TIME' is not supported.

不支持“DATE_AND_TIME”数据类型。要了解 OptiSYS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看你的硬件资料，得到你的硬件能支持的数据类型列表。

7.3.3.6 C10012

Data type 'TIME' is not supported.

不支持“TIME”数据类型。要了解 OptiSYS 能支持的数据类型列表请参看 IEC 1131-3 说明。请参看你的硬件资料，得到你的硬件能支持的数据类型列表。

7.3.3.7 C10017

The sections 'VAR_INPUT', 'VAR_OUTPUT' and 'VAR_IN_OUT' are not supported in programs.

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。

7.3.3.8 C10019

Directly represented variables are not allowed in this POU.

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。

7.3.3.9 C10021

Constant must not be negative.

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)。

7.3.3.10 C10024

Constant is out of range.

在给定位置的常数不在指定的数据类型范围之内。

7.3.3.11 C10025

IN/OUT parameters must always be supplied with actual parameters.

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。

7.3.3.12 C10026

Unsupported address.

在给定的位置的地址，当前硬件不支持。请参看你的硬件资料，获得硬件支持的地址列表。

7.3.3.13 C10028

Inout-parameters of type struct are not supported.

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。

7.3.3.14 C10031

RETAIN-variables are not supported by this hardware.

你的硬件不支持 RETAIN 变量。请删除这个属性。请参看你的硬件资料，获得一个能支持的变量列表。

7.3.3.15 C10034

Invalid command for this hardware.

这个硬件不支持在给定位置的命令。请参看你的硬件资料，获得你的硬件不支持的命令

列表。要了解 OptiSYS 不支持的命令列表，请参考 IEC 1131-3 说明。

7.3.3.16 C10035

Operand of type 'UINT' expected.

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
```

7.3.3.17 C10036

Structs and arrays of complex data types are not supported by this hardware.

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。要了解更多有关你的硬件能支持的数据类型，请参看你的硬件资料。

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;
```

7.3.3.18 C10038

Couldn't detect the type of the constant.

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。

7.3.3.19 C10043

Implemention code is not allowed.

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。

7.3.3.20 C10045

Function blocks instances are not allowed in this section.

在不允许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。

7.3.3.21 C10046

'VAR_GLOBAL' is not allowed.

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。

7.3.3.22 C10047

Only 'VAR_GLOBAL' allowed.

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。

7.3.3.23 C10049

String too long.

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。OptiSYS 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中“MODULE”部分的 MaxStringLength 值来限制字符串的最大长度。

7.3.3.24 C10055

This variable can not be initialized.

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。OptiSYS 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。请参看你的硬件资料，确定你的硬件是否能支持变量初始化。

7.3.3.25 C10057

Data type is not supported.

不支持给定位置的数据类型。要了解 OptiSYS 能支持的数据类型列表，请参考 IEC1131-3 说明。要了解你的硬件能支持的数据类型列表，请参看你的硬件资料。

7.3.3.26 C10060

LD/ST of function block instances is not allowed.

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。

7.3.3.27 C10063

打开一个文件时发生错误。

7.3.3.28 C10064

Internal Compiler Error No. %1. Please contact your manufacturer.

发生了一个内部编辑错误。请联系你的生产商。

7.3.3.29 C10067

Struct declarations are not supported.

发现了一个结构声明，但硬件不支持这个声明。OptiSYS 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。请参看你的硬件资料，确定你的硬件是否支持结构声明。

7.3.3.30 C10068

Array declarations are not supported.

发现了一个数组声明，但你的硬件不支持这个声明。OptiSYS 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。请参看你的硬件资料，确定你的硬件是否支持数组声明。

7.3.3.31 C10069

Enumerated data type declarations are not supported.

发现了一个列举数据类型声明，但你的硬件不支持这个声明。OptiSYS 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。请参看你的硬件资料，确定你的硬件是否支持列举数据类型声明。

7.3.3.32 C10075

Invalid array index. It has to range between -32767 and 32767.

一个数组的索引超过了能支持的范围[-32767, 32767]。

7.3.3.33 C10078

Invalid type of a global or directly represented variable.

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。

7.3.3.34 C10083

Only directly represented variables are allowed in this POU.

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。

7.3.3.35 C10084

Global structs are not supported.

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
```

7.3.3.36 C10092

内存分配失败。

7.3.3.37 C10100

Invalid expression for parameter.

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。

7.3.3.38 C10108

Constant of type TIME is out of range.

要了解 OptiSYS 能支持的 TIME 常数的范围，请参考 EC 1131-3 说明。

7.3.3.39 C10109

Invalid data type for this operation. Integer or real type expected.

在给定位置的操作只支持整数和实数类型的操作数。

7.3.3.40 C10110

Nested functions are not supported.

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。

7.3.3.41 C10112

Type conflict.

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。

7.3.3.42 C10113

Operation not supported for this data type.

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。

7.3.3.43 C10114

Parameter expressions are not supported for this operation.

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。

7.3.3.44 C10115

Retain attribute for FB instances forbidden.

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。

7.3.3.45 C11001

Can't determine unambigously the type of constant -> 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。

7.3.3.46 C11007

Function has no input parameter. Is this intended?

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。

3.4 生成文件消息

7.3.4.1 M21004

Unknown command: %1.

使用了一个带有 ITMAKE 的未知的命令行。

第八章 通讯

OptiSYS提供串行通讯自由编程。CPU的串行通讯讯可由用户程序控制，这种操作模式称为自由端口模式。当选择了自由端口模式，梯形图程序可以使用发送指令（SERSEND）和接收指令（SERRCV）来控制通讯操作。在自由端口模式下，通讯协议完全由梯形图程序控制。自由口通讯模式由硬件组态程序组态设定。

发送指令（SERSEND）

输入

ACT

PORT

ADDR

NUM

输出

STATUS

功能块需无条件启动。

ACT的上升沿启动数据发送；PORT指定要参与数据发送的串行口，如PAC313-2，端口分别为1和2；ADDR指定待发送数据的起始地址；NUM指定发送数据的字节数。如PORT是1，ADDR是6，NUM是12，说明要发送的数据为m6.0起始至m17.0的12个字节。一旦给ACT一个上升沿，m6.0至m17.0的12个字节通过串口“1”发送一次。发送成功后，STATUS保持一个PLC扫描周期为TRUE。

接收指令（SERRCV）

输入

ACT

PORT

TIMEOUT

ADDR

MAXNUM

输出

NUM

功能块需无条件启动。

ACT的上升沿启动数据接收；PORT指定要参与数据接收的串行口，如PAC313-2，端口分别为1和2；TIMEOUT指定启动接收开始至接收结束的超时时间（单位：毫秒），如果超时，程序认为该次接收结束；ADDR指定接收数据区的起始地址；MAXNUM指定接收数据区的最大字节数。如PORT是1，TIMEOUT是200，ADDR是20，NUM是32，说明要接收的数据为m20.0起始至m51.0的32个字节。一旦给ACT一个上升沿，PLC启动接收，在200ms内，当PLC接收到数据后，将数据保存在m20.0至m51.0的存储区内。当PLC有数据接收到时，NUM显示当前收到的字节数。

第九章 OPC Server

OPC 服务器

OPC 服务器将从一个数据库中取得所有关于 OPC 标签和通讯信息。要用 OPC，你将必须要有这样一个配置数据库。最简单的办法是把一个叫做 OPCSVR.MDB 的模板导入到你的工程中，这个模板是由 OpenPCS (C:\OpenPCS)提供的。在 OpenPCS 的浏览器中，用“文件增加文件”，双击这个数据库打开它，并且输入你的工程所需要的所有信息。为了这个目的，你将需要安装 Microsoft Access 数据库：

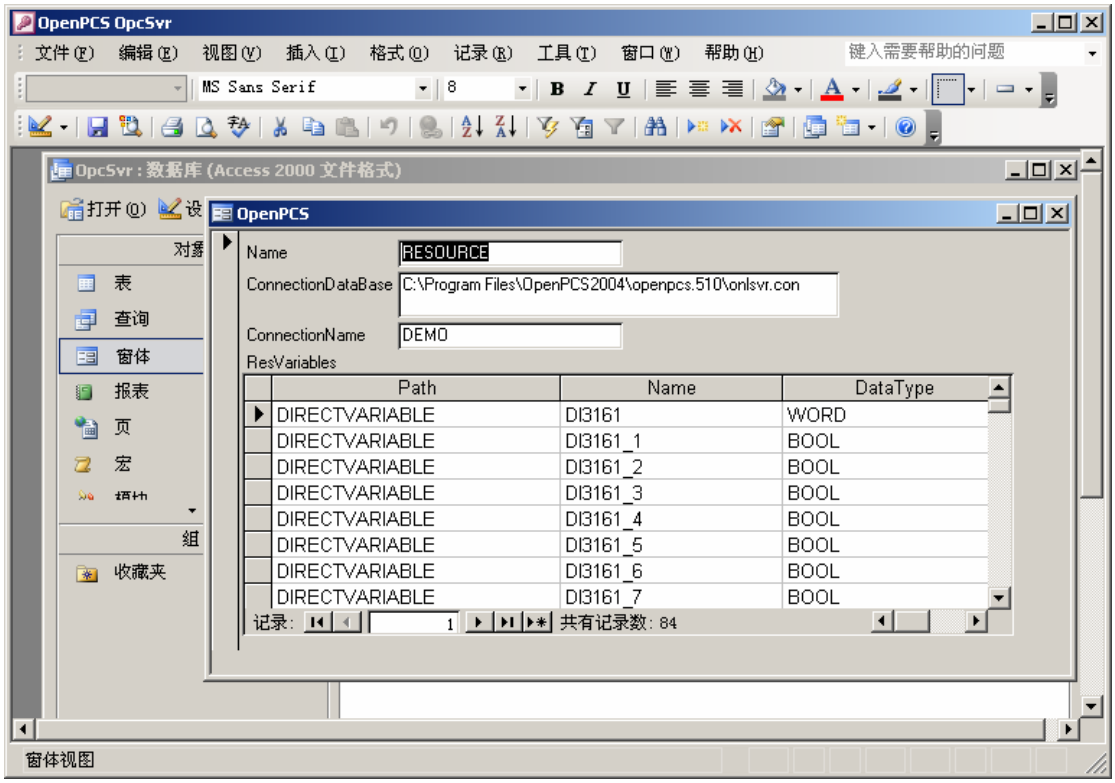


图9.1-1

即使在 OpenPCS 没有运行时，OPC 服务器也需要能够找到这个数据库。因此，这个数据库需要在 Windows 注册表中注册。在 OpenPCS 的浏览器中选择数据库，并且调用“Extras Tools OPC DB Registration”：

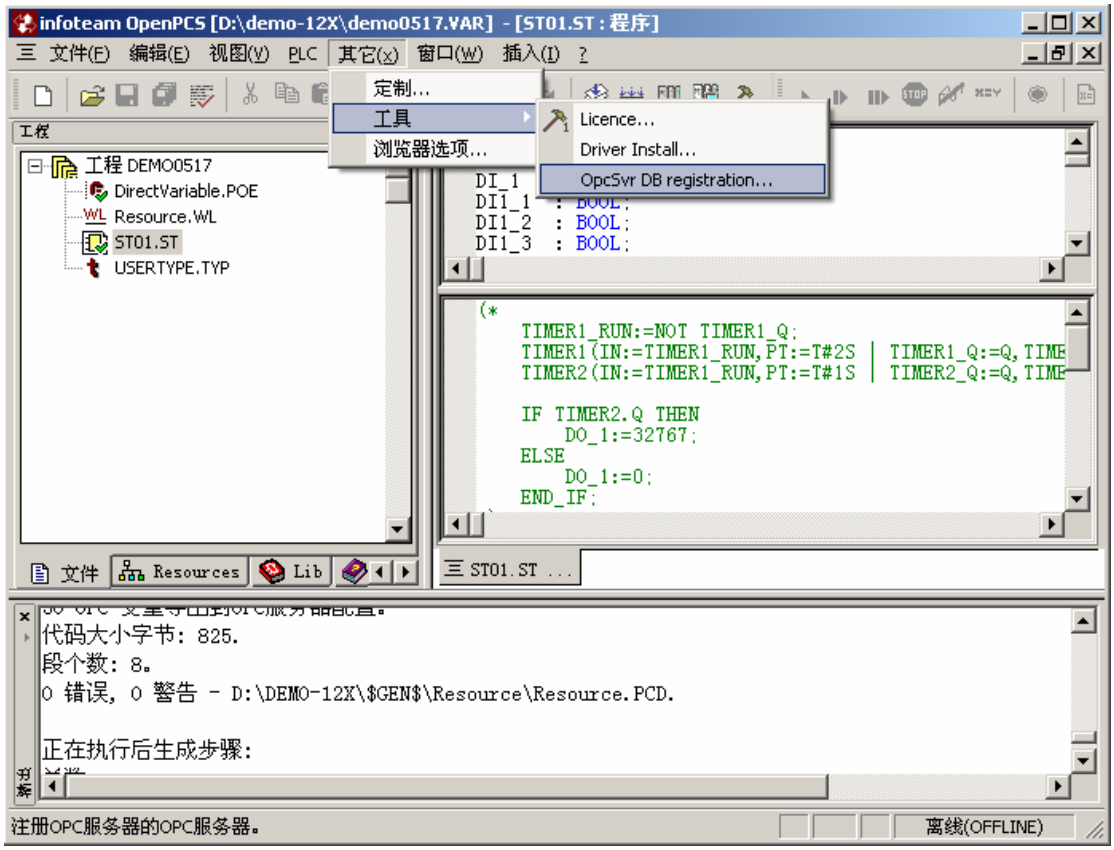


图9.1-2

注意：

(1) 当前的 OPC 服务器不支持多连接。

(2) 注意包括在数据库中的数据（变量名，路径和数据类型）要与你的应用程序相匹配。

(3) 在一个工程中可能有多个数据库，多个工程也可以有一个这样的配置数据库。OPC 服务器总是用由 Extras Tools OPC DB Registration 注册的数据库。这个不取决于哪个工程被打开了。

(4) OpenPCS OPC 服务器支持下面的 OPC 标准：OPC 公用定义和接口 1.0 版本，用户接口标准 2.0 版本，数据自动访问接口标准 1.0 版本， OPC 告警和事件 1.0 版本。