www.3s-software.com

CoDeSys V3 基础编程入门

马立新　　康存锋

目　　录

1. 概述和基本概念

CoDeSys 是一种与设备（硬件）无关的可编程控制器（PLC）编程系统。**CoDeSys** 不仅完全支持所有符合 **IEC 61131- 3** 标准的编程语言，同时还支持 **C** 语言等高级编程语言。与 **CoDeSys** 实时运行系统（Runtime System）结合后，可以在一个工程（项目）中对多个控制器(设备)进行统一配置和编程。

使用 **CoDeSys** 编程时，请留意下列基本概念：

▪ **面向对象的编程：**

CoDeSys V 3.4 中，在编程元素、编程特性、工程结构、版本管理等各个方面都体现了面向对象进行编程的重要思想。可以通过联合、实例化来实现在一个工程中的**多设备编程**和**多应用编程**。因此，开发者可以在同一个设备上运行多个应用程序、可以对应用进行拷贝、可以在一个工程中混合配置参数型的和可编程型的硬件（系统）。

▪ **基于组件的编程系统结构：**

在用户界面中（例如编辑器和菜单等）可以使用的功能，是由在配置文件中定义的所使用的组件（插件）来决定。组件又分为系统组件和可选组件，其中系统组件是必需的基本组件。除了德国 **3S** 软件公司提供的这些组件之外，用户还可以使用 **CoDeSys** 自动化开发平台工具包（**CoDeSys Automation Platform Toolkit**）来创建**自定义的组件**。

▪ **版本管理：**

在 **CoDeSys** 中可以同时安装一个组件的多个版本，并且可以组合使用这些版本，编译器也可以安装和使用多个版本；而且无需更新整个版本就可以新增独立的功能。

▪ **工程（项目）的组织方式也同样采用了面向对象的方法：**

在 **CoDeSys** 工程中，包含了由各种编程对象组成的 **PLC** 程序对象，还包含了在目标系统（硬件设备）上运行 **PLC** 程序时需要的“资源”对象。

由上所述，在一个工程中有两类主要的对象：

（1）编程对象（POUs):

编程对象 **POU** 包括程序、函数、功能块、方法、接口、动作、数据类型定义等。

在“**POU** 窗口”中管理的编程对象，在整个工程范围内都有效，且可以被工程中所有的“应用”通过任务配置来调用，即**实例化**。在“设备窗口”中管理的编程对象（即针对特定应用的编程对象），只能被本应用来使用，或被本应用的“子应用”实例化后使用。

（2）资源对象（设备树）:

资源对象包括设备对象、应用、任务配置、配方管理等。资源对象只能在设备窗口中进行管理，即只能在设备树中进行管理。在设备树中添加对象后，需要按一定的“规则”与被控设备进行映射。对象（如库和 **GVL** 等）在工程中的有效范围，会依据设备树中应用和设备对象的层级关系而定，一般来说，一个应用中的对象对其“子应用”也有效，可以被使用。

▪ 由集成的编译器生成代码，并使用机器码以便加快执行时间。

▪ 与控制器设备之间的数据传输：在 **CoDeSys** 与目标设备之间，通过 **Gateway** 组件和实时运行系统（Runtime System）进行数据的传输。提供了完善的在线功能对设备程序进行实时监控。

2. 创建并运行一个工程

以下部分介绍了如何创建一个包含 **PLC** 程序的简单工程，以及如何通过 **Gateway Server** 将这个程序加载到 **PLC**（目标设备）硬件上，运行并监控此程序。**CoDeSys** 安装程序中缺省提供了用于该示例工程的 **PLC** 实时运行系统。

示例程序用结构化文本（**ST**）语言编写，包含一段程序：PLC_PRG，和一个功能块：FB1；PLC_PRG 中包含一个计数器变量 ivar，并调用功能块 FB1；FB1 从 PLC_PRG 中得到输入值“in”，在这个输入值上加“2”，并将结果输出到 out，由 PLC_PRG 读 out。

（**注意**：下列关于用户界面的默认配置说明由当前安装的软件版本提供）

启动 CoDeSys 并创建一个工程

（1）启动 CoDeSys

从开始菜单选择

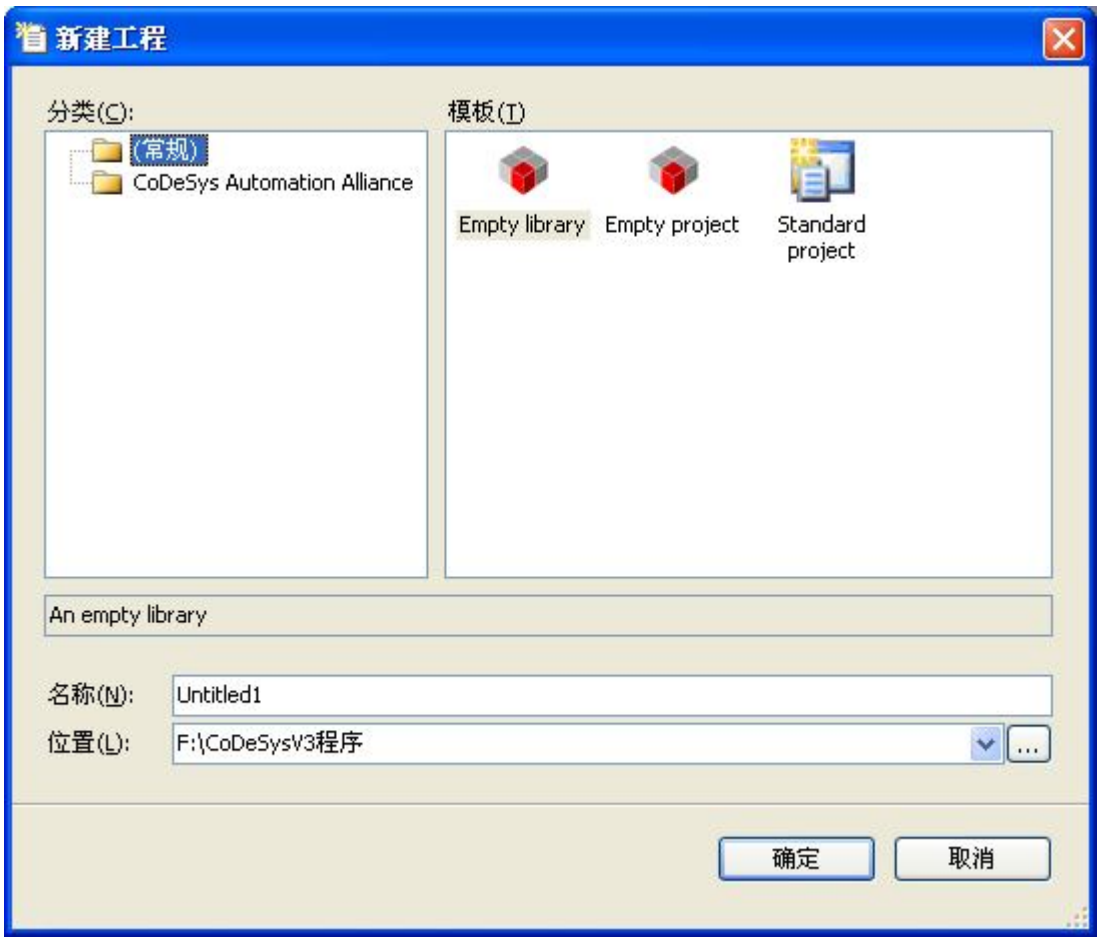
程序 > 3S CoDeSys > CoDeSys > CoDeSys V 3.4

或者双击桌面上的图标 启动 **CoDeSys**。（之后，系统要求用户选择配置文件，点击“继续”后，打开 **CoDeSys** 的用户界面）

系统将自动按照预定义配置文件启动，所以事先不需要选择配置文件。

（2）创建工程

在**文件**菜单中选择**新建工程**命令，用来创建一个新的工程。



在**新建工程**对话框模板区选择**标准工程**，并为工程文件输入**名称**及**本地**路径，点击“**确定**”后向导对话框打开，如下图所示：

设备选择“**CoDeSys** SP Win V3 (3S-Smart Software Solutions GmbH)”，PLC_PRG 的编程语言选择“结构化文本(**ST**)”。点击“**确定**”保存配置。

工程名显示在 **CoDeSys** 用户界面的标题栏上，同时作为 **POUs** 视窗和设备视窗中的一个根节点符号。

POU 视窗包含工程设置。

设备视窗显示一个 **CoDeSys SP Win V3** 类型设备树，显示“设备 (**CoDeSys SP Win V3**)”及其包含的应用。

其中，应用包括：

PLC_PRG：程序由结构化文本编辑；

任务配置：用来定义控制 PLC_PRG 程序的“MainTask”；

库管理器：包含“I/O Standard.library”及“Standard.library”；“I/O Standard.library”用来进行 I/O 配置；“Standard.library”提供所有符合 IEC 61131- 3 标准的所有函数和功能块，作为 **IEC** 编程系统的标准 **POUs**。

（此时，“设备(**CoDeSys SP Win V3**)”节点下的“Plc Logic”节点无实际意义，仅是一个符号节点，指示该设备是“可编程”的。）

选择设备名符号节点，点击空格键，进入编辑状态，输入其他名称替换“Device”，可以对设备名重新命名。

编写 PLC 程序

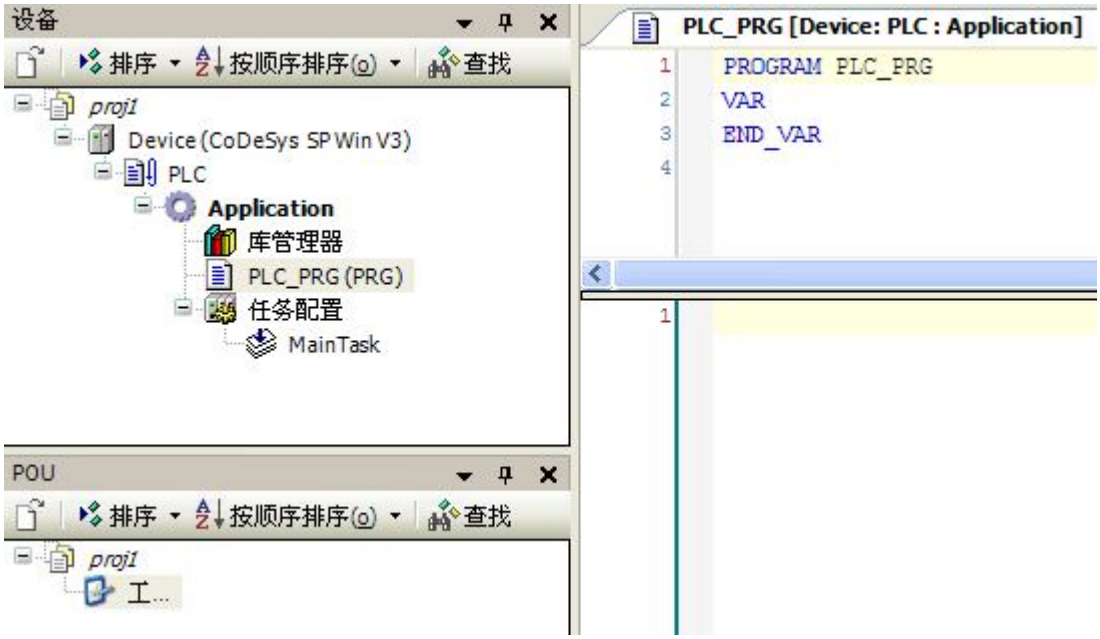
（1）在 PLC_PRG 中声明变量

在设备视窗中，缺省 **POU** 为“PLC_PRG”，双击设备树中“PLC_PRG”，自动在 **CoDeSys** 用户界面中部的 **ST 语言编辑器**中打开。

ST 语言编辑器包含声明部分（上部），实现部分（下部），由一个可调的分割线分开。

声明部分包括：显示在左侧边框中的行号、**POU** 类型和名称(如“PROGRAM PLC_PRG”)，以及在关键字“VAR”和“END_VAR”之间的变量声明。

实现部分是空，仅显示行号 1：



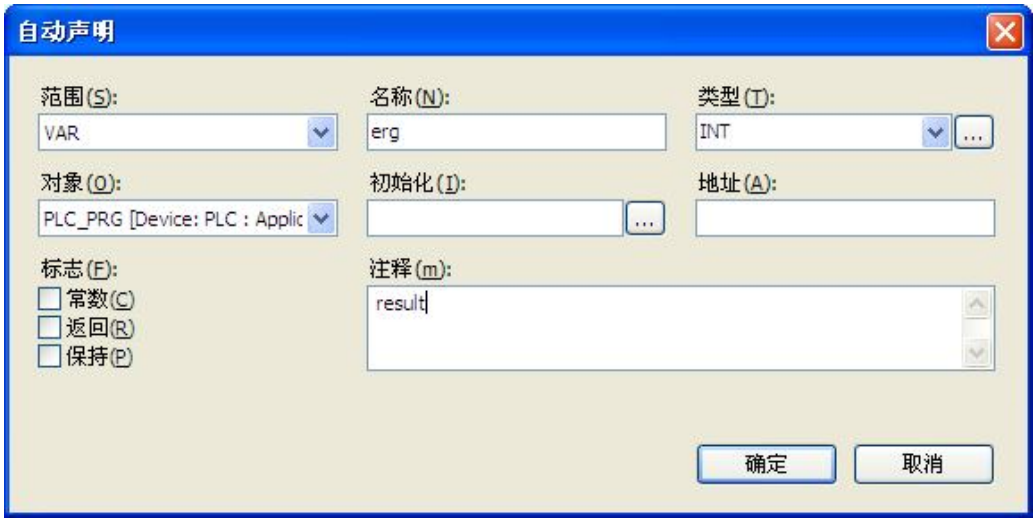
在编辑器的声明部分，将光标移到 VAR 后，点击回车。插入新的空行，声明 INT 类型变量“ivar”，INT 类型变量“erg”，FB1 类型变量“fbinst”

```
PROGRAM PLC_PRG
  VAR
        Ivar：  INT；
        Fbinst：FB1；
        Erg：  INT；
  END_VAR
```

另外，也可以使用自动声明功能，在编辑器执行部分直接输入指令，详见第(4) 部分。

(2) 在 PLC_PRG 的实现部分输入程序代码

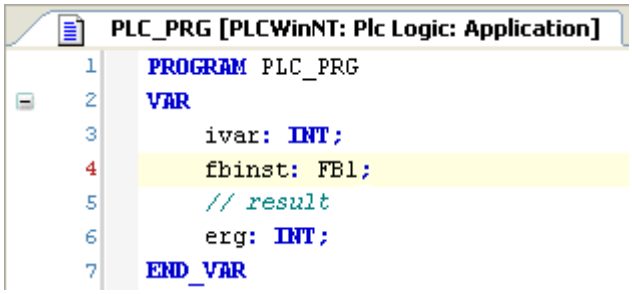
```
ivar := ivar+1;              // counter
fbinst (in:=11, out=>erg);    // call function block of type FB1,
                                  // with input parameter "in"
                                  // output is written to "erg"
```



代替第(3)步，第(4)步介绍可以使用自动声明功能：在程序的实现部分输入指令，点击回车，如果在新一行中有未声明的变量，则系统打开自动声明对话框，在此可以进行声明设置。

变量名称、范围及当前 **POU**（对象）将自动添加。根据(3)中声明描述的输入需要的类型和初始值。注意：如果在自动声明对话框中定义注释内容，不需要插入“// ”，而是以在声明部分以 XML 格式描述，以便以后作为文档使用。

点击“确定”关闭对话框。在 **POU** 的声明部分，erg 变量声明之前插入其注释。



(3) 创建编程 POU(用 ST 语言编写功能块 FB1)

我们可以再创建一个功能块：**FB1**，其功能是在输入变量“in”上加“2”，将结果输出到“out”。

从工程菜单中选择“**添加对象**”命令。

在“添加对象”对话框左侧选择“**POU**”，输入 **POU** 名称：**FB1**，在类型选项中激活“**功能块**”选项。

方法**实现语言**选择“结构化语言(**ST**)”。

点击“**打开**”按钮确认对象设置。

用于新功能块 **FB1** 的编辑窗口打开。与 PLC_PRG 的变量声明一样，在此对以下变量声明：

```
FUNCTION_BLOCK FB1
  VAR_INPUT
    in:INT;
  END_VAR
```

```
VAR_OUTPUT
  out:INT;
END_VAR
VAR
  ivar:INT:=2;
END_VAR
```

在编辑器实现部分输入以下内容：

```
out:= in+ivar ;
```

为在 PLC 上运行和控制程序，定义资源对象

（1）启动 Gateway Server 和 PLC

启动 **Gateway Server**：

Gateway Server 作为服务程序在系统启动时自动启动，请确认在系统托盘处是否有指示 **Gateway Server** 运行的图标  。如果图标为  ，则表明 **Gateway Server** 当前未启动。

（这个图标是 **Gateway Sys Tray** 程序的一部分，**Gateway Sys Tray** 用来控制和监视 **Gateway Server**，图标右键菜单提供“启动”和“停止”命令，因此允许用户手动停止或者重启程序。**Gateway Inspector** 功能目前尚未实现。使用菜单中的“退出 **Gateway Control**”命令，可以退出 **Gateway Sys Tray** 程序。Windows 启动时， **Gateway Sys Tray** 程序会自动启动，也可以通过程序菜单手动启动。）

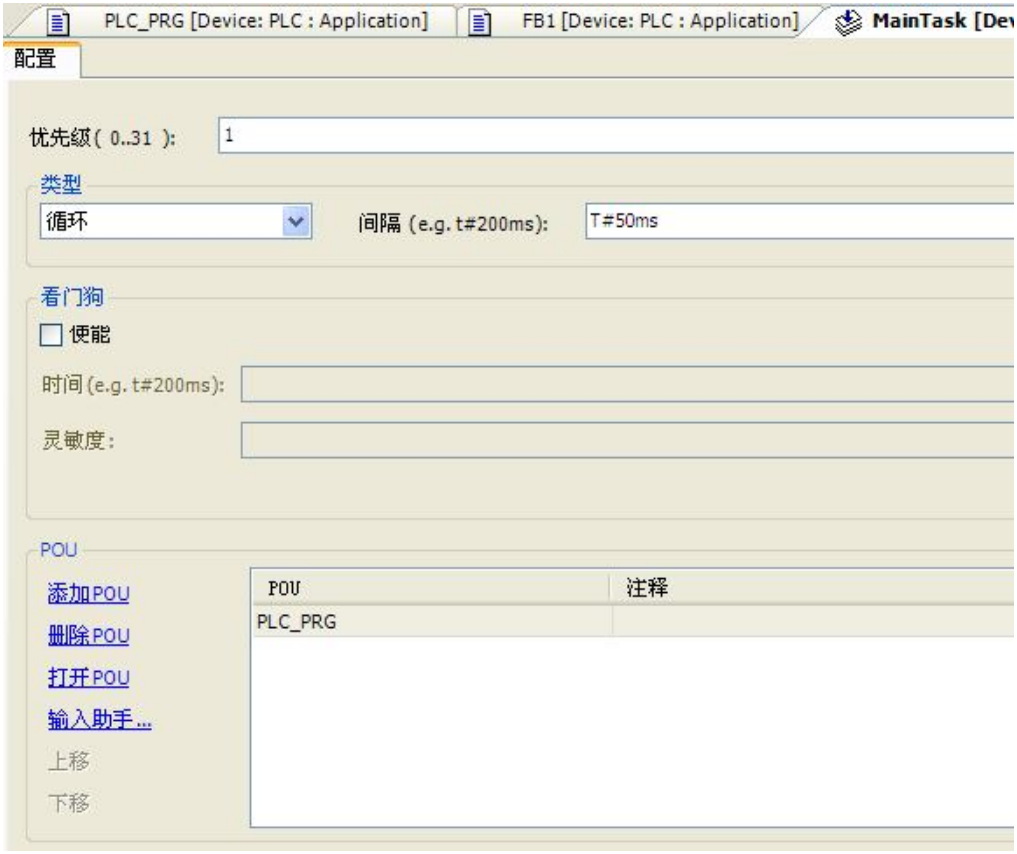
启动 **PLC**：

系统启动时，**PLC(CoDeSys SP Win V3)**作为服务程序在系统启动时自动启动。其图标会显示在系统托盘中，  代表“停止”状态，  代表“运行”状态。若系统许可，**PLC** 服务程序将在系统启动时自动启动。否则需要手动点击图标右键菜单中的“启动 **PLC**”命令启动服务。

（这个图标是程序 **CoDeSys SP Sys Tray** 的一部分，**CoDeSys SP Sys Tray** 用来控制和监视 **CoDeSys SP server**，图标右键菜单提供了“启动”和“停止”命令，因此允许用户手动停止或者重启程序。使用菜单中“退出 **PLC Control**”命令，则退出 **CoDeSys SP Sys Tray** 程序。Windows 启动时， **CoDeSys SP Sys Tray** 程序会自动启动，也可以通过程序菜单手动启动。）

（2）激活“应用”

点击 **Standard project** 设备视窗中的 MainTask，打开包含任务设置的编辑器视图，则如下图所示：



在设备视窗中，“**Application**”显示为黑色字体，表明该应用处于激活状态。这样，所有与 **PLC** 通讯相关的命令和动作都关联于该应用。在设备视窗中选择应用，从右键菜单中选择 “设置当前的应用” 命令将激活此应用。

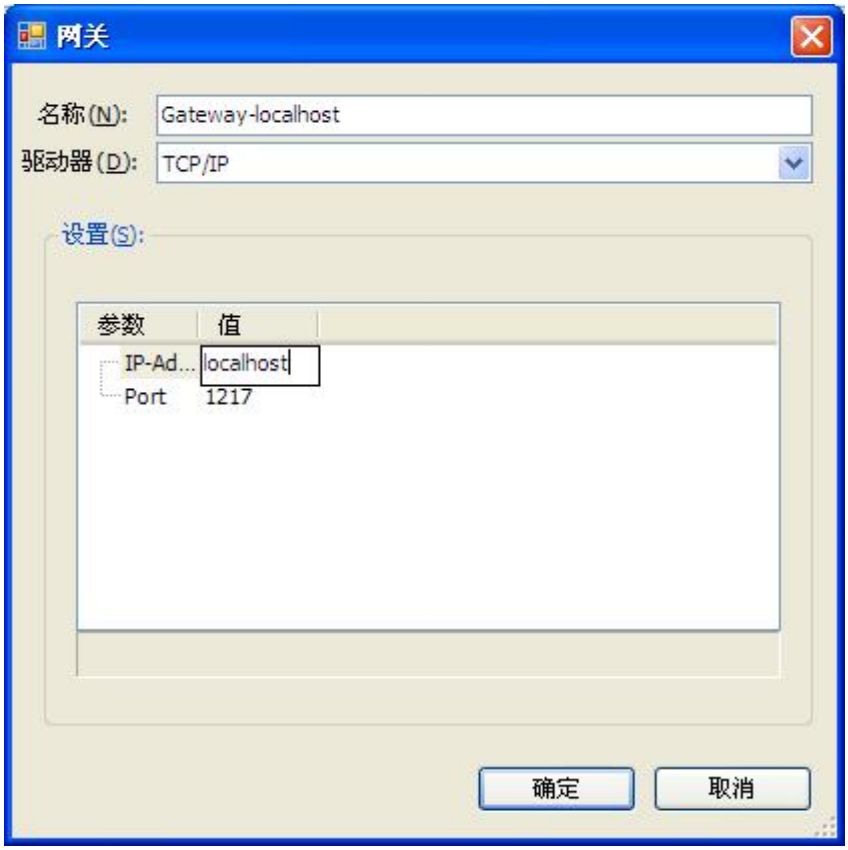
(3) 设置通讯参数

在设备视窗中双击 **Device (CoDeSys SP Win V3)**，打开通讯设置子对话框。根据以下步骤，可以在此处安装 **PLC (目标设备)**和编程系统的连接。连接将被输入到“给控制器选择网络路径”的下一行。

如果第一次使用 **CoDeSys** V 3.4 进行通讯设置，您首先需要设定本地 **Gateway Server**。

（如果之前已经设定过服务程序，则其设置如下图“现在定义……” 中通讯设置对话框所示。这样，可省略该步，并继续定义目标通讯通道，也可参见下面的“现在定义…”部分。）

该服务程序由 **CoDeSys** 安装程序提供。点击“**添加网关**”按钮打开 **Gateway** 对话框：



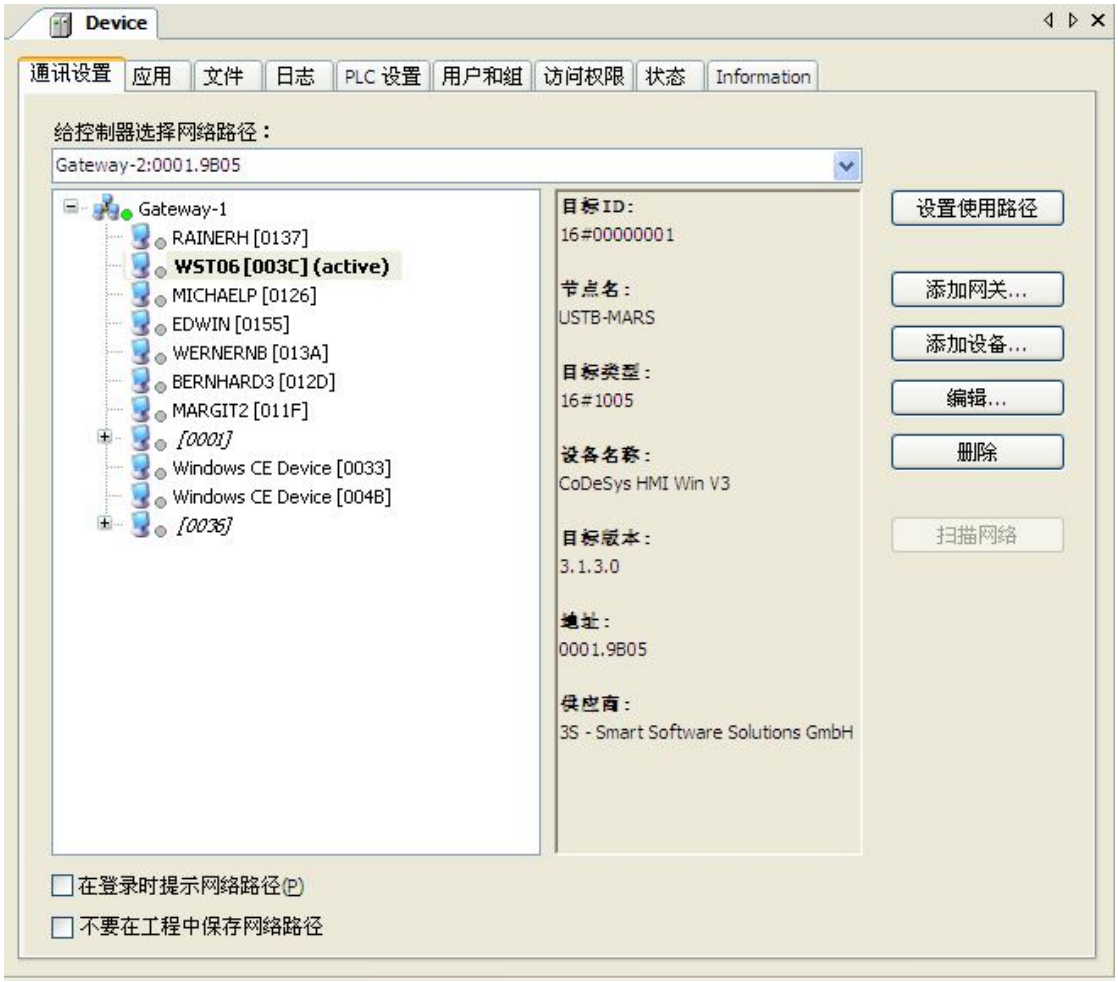
为网关输入名称，驱动器选择“**TCP/IP**”，**IP** 地址设置为“**local host**”（鼠标双击相应列，打开编辑框），端口不用设置，点击“确定”关闭对话框。

网关被添加到通讯对话框左侧，网关名称被添加到“给控制器选择网络路径”选项列表。当网关正常运行时，**Gateway** 前面会显示绿色圆点，红色则表示异常。

现在确认通过网关设置与目标设备的通道是否被连接：

点击“**扫描网络**”按钮，可以在本地网络上搜寻可用的设备。

至少可以查找到与 **CoDeSys** 同时安装的 **PLC**：该 **PLC** 在 **Gateway** 下**缩进显示**：在下图中的类似“**WST06 [003C]**”的位置会显示您的计算机名称和地址。若未找到 **PLC**，请检查其是否正在运行，请参见第（6）步。



选择 **PLC**（设备）项，点击“**设置使用路径**”。

该操作会激活通讯通道设置，也就是所有与通讯相关的操作将与该通道关联。当工程中设置多个通讯通道时，请注意这点。

点击“**确定**”关闭通讯对话框并应用设置。

在 PLC 中运行并查看应用

(1) 编译并将应用程序下载到 PLC

如果只检查“激活”应用程序中的语法错误，点击“编译应用”命令（右键菜单，编译菜单）或者使用快捷键 **<F11>**。注意：这种情况下并不产生代码。信息窗口中会显示消息、警告和错误信息，默认的信息窗口在用户界面的底部。即使并未进行过语法检查，仍可以登录到 **PLC**。（首先要确定 **PLC** 正在运行，系统任务栏的图标是高亮的）。
选择在线菜单的“登录到应用”命令。如果已经按第（8）步中的描述进行了通讯设置，以下信息框会出现（否则您会被询问更正通讯设置）：

“在目标设备中没有应用程序，是否想要创建和继续进行下载？”

点击“是”启动应用的编译和下载。信息窗口显示编译信息。如果工程正确创建，不会出现编译错误，此时应用被启动，请参见第（10）步。

（2）启动和监控应用程序

正如前面步骤所描述的，创建“standard project”，下载至“**应用**”（已编译无错），设备“**CoDeSys SP Win V3**”的应用被启动 ：

在 PLC 上启动应用程序：

点击在线菜单中的“启动”命令，程序开始运行，在用户界面的底部状态栏会显示一个绿色的“运行”标志。

监控应用程序

使用以下三种方法，可以监控应用程序中的变量：

1. 包含已定义监控列表的监控窗口；
2. 写入和强制变量；
3. 特殊 **POU** 的在线视图；

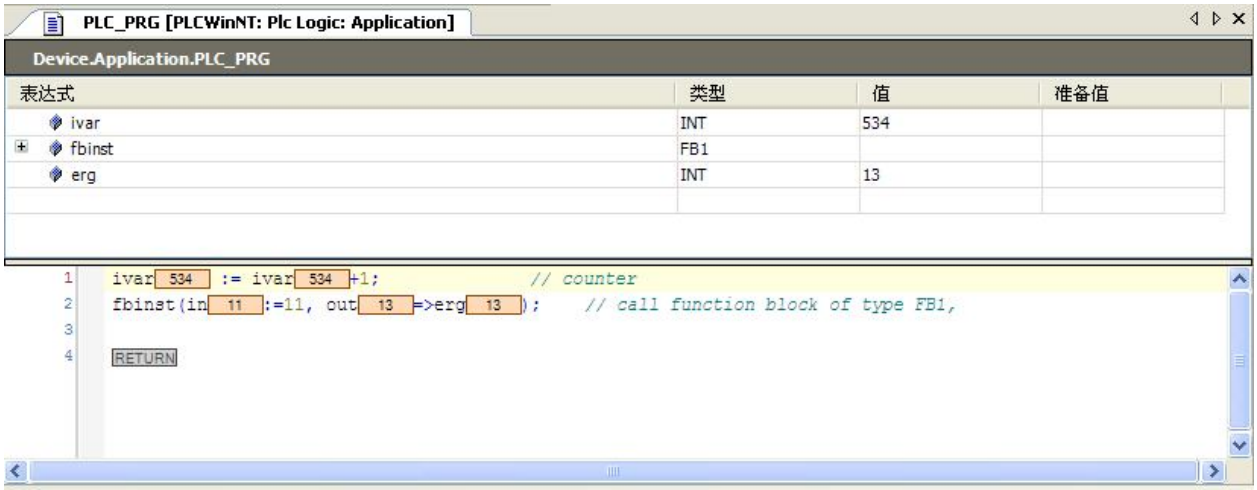
（1）．打开程序的示例窗口

POU 的示例视图提供了该示例的所有监控表达式，并在声明部分以表格显示，若激活“在线监控”功能，在实现部分也同样会显示。

双击在设备窗口里的执行程序“PLC_PRG”，或选择该项右键菜单点击 “**编辑对象**”命令，打开在线视图，出现如下对话框，显示所有 PLC_PRG 的示例(该例中仅一个) ：

这里，可以选择 **POU** 以在线模式或是离线模式进行查看。在当前示例中，默认设置为在线模式，点击**确定**关闭对话框。

PLC_PRG 在线视图被打开(如下图)：上半部分显示对应于 PLC_PRG 实现部分视图的程序主体，由每一个变量后的内部监控窗口显示实际的值。上半部分显示 PLC_PRG 的监控表达式，如：在 PLC 的**“应用”**中的各自变量当前值。



（2）. 写入和强制变量

可以通过写入或强制的方式将某一**“预设值”“准备值”**赋给变量 ivar，意思是表示在下一周期开始，变量 ivar 将显示为该值。在**预值栏**的区域双击鼠标左键激活输入框，输入所需整数

的值，单击（点击）回车或者该区域外部，关闭此输入框。 执行命令“**写入值**”或者“**强制值**”命令，**将写入值**或强制值下装到 **PLC**。在**准备值**的窗口中可以立刻看到结果。

（3）. 使用监控视窗

监控窗口可用于配置特定的监控表达式，例如：调试目的。

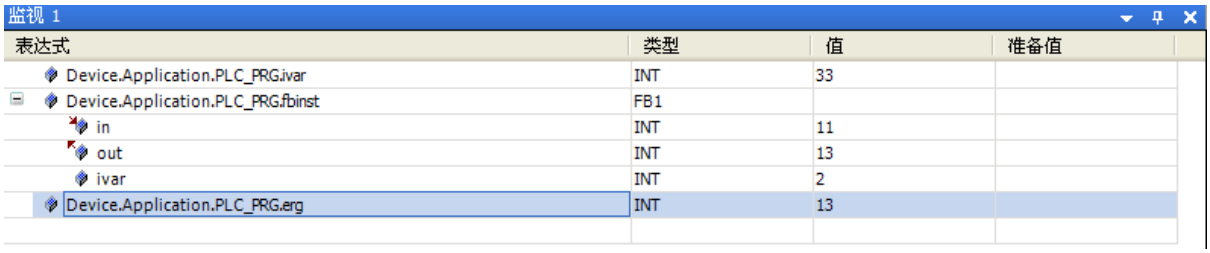
打开**视图**菜单，选择**监控 -> 监控 1** 命令，打开监控窗口。

鼠标点击**表达式的**第一行，打开编辑框，输入要监控的变量 ivar 完整路径：“ PLCWinNT. Application .PLC_PRG. Ivar ”。

建议通过输入助手按钮实现该功能，点击回车，关闭编辑框，变量类型将自动添加到表格。

该操作对其他变量同样适用。下图显示的监控列表只包含 PLC_PRG 的表达式，也可以在工程中创建任意一组变量。注意示例变量中：例如，对于 **FB1** 的示例变量，输入表达式“MyPlc . Application . PLC_PRG . fbinst”就可以了。将自动加入特殊变量，点击加号，可以打开与其相关联的行：在“值”列显示当前变量的值：

如果尚未完成，选择在线菜单上的“**启动**”命令。**PLC** 上将启动应用程序，当前值显示在“值”对应的列中：



此处也可以对变量进行写入和强制值（同 2 中的介绍）。

点击在线菜单中的“**退出**”命令，与 **PLC** 断开连接

调试程序

(1) 设置断点并单步执行程序

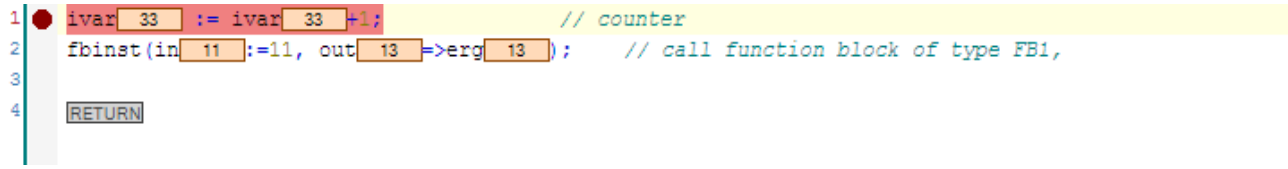
在线模式下，可以设置断点，定义程序执行的暂停位置。

程序到达一个断点时，可以逐步执行程序。在每个停止位置，都可以在监控视图中查看变量的当前值。

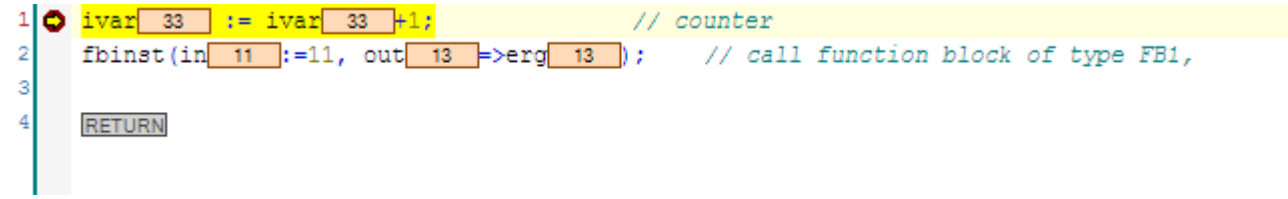
请尝试以下步骤：

选择 PLC_PRG 的第一行，按功能键**<F9>**或者从在线菜单中点击“**切换断点**”命令，断点显示在相应位置。

如果应用程序当前处于“停止”状态，如下图所示：



如果应用程序当前正在运行，它将在新的断点处停止：



使用功能键**<F8>**或者从在线菜单中选择“**跳入**”命令，进入到功能块示例中。按功能键**<F10>**键或者从在线菜单中选择“跳出”命令，可以跳出功能块。**PLC** 中每个变量的当前值都会显示出来。

通过视图菜单中的“断点”命令打开断点对话框查看。此处，除可以查看和编辑断点的当前设置外，还可以输入新的断点。

注意：在退出时，断点位置会被保存，并由浅红色圆点指示。

3. CoDeSys V3 – 快速入门

使用 CoDeSys V3 来编写一个控制车库门的程序

在下面的描述中，我们将向您展示：使用 CoDeSys V3 编写一个简单的自动化工程是如此的容易。

首先，我们先让您熟悉一下这个真实的工程任务。为了让您更快地掌握它，我们将这个工程分为多个容易实现的单元。我们将和您一起：一步一步地编写这个应用程序，您也可以在这个教程的最后来下载整个工程。一般来说，按照正常的学习速度，您可在 30 到 60 分钟内就可完成这个工程项目。

祝您好运！

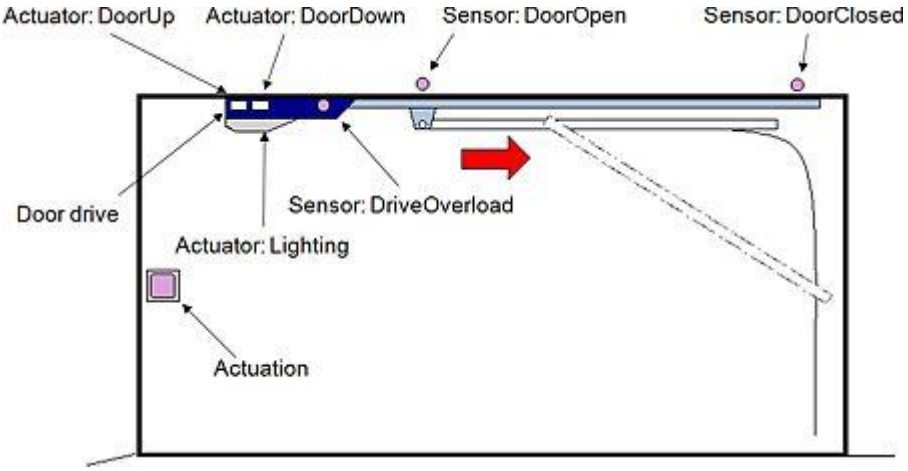
同时，我们非常荣幸能得到您的反馈信息。如果您有任何疑问或建议，请直接联系我们：codesys@126.com

友情提示：此应用程序仅能做示例之用，禁止用于工业现场或实际应用。

CoDeSys V3 – 快速学习 1/6

工程任务：

使用 CoDeSys V3 SoftPLC 来控制一个标准的车库门。



任务描述：

如果用户按下控制按钮（驱动），车库门根据当前的状态，执行由开门到关门，或者由关门到开门的动作。在此过程中，控制器激活相应的执行部件（升门/ 降门），直到传感器（门开 / 门闭 / 门过载）报告门已达到了最终位置，或有错误发生，终止了门的开闭过程。再次按下控制按钮，用户可以随时手动停止车库门。在车库门开启或者关闭的过程中，车库内顶灯会自动打开，经过一定的时间间隔后，顶灯会自动关闭。任何错误的发生，比如车库门过载，或者车库门开或关的时间过长，当前动作立即中断，顶灯开始闪烁，直到再次按下按钮。

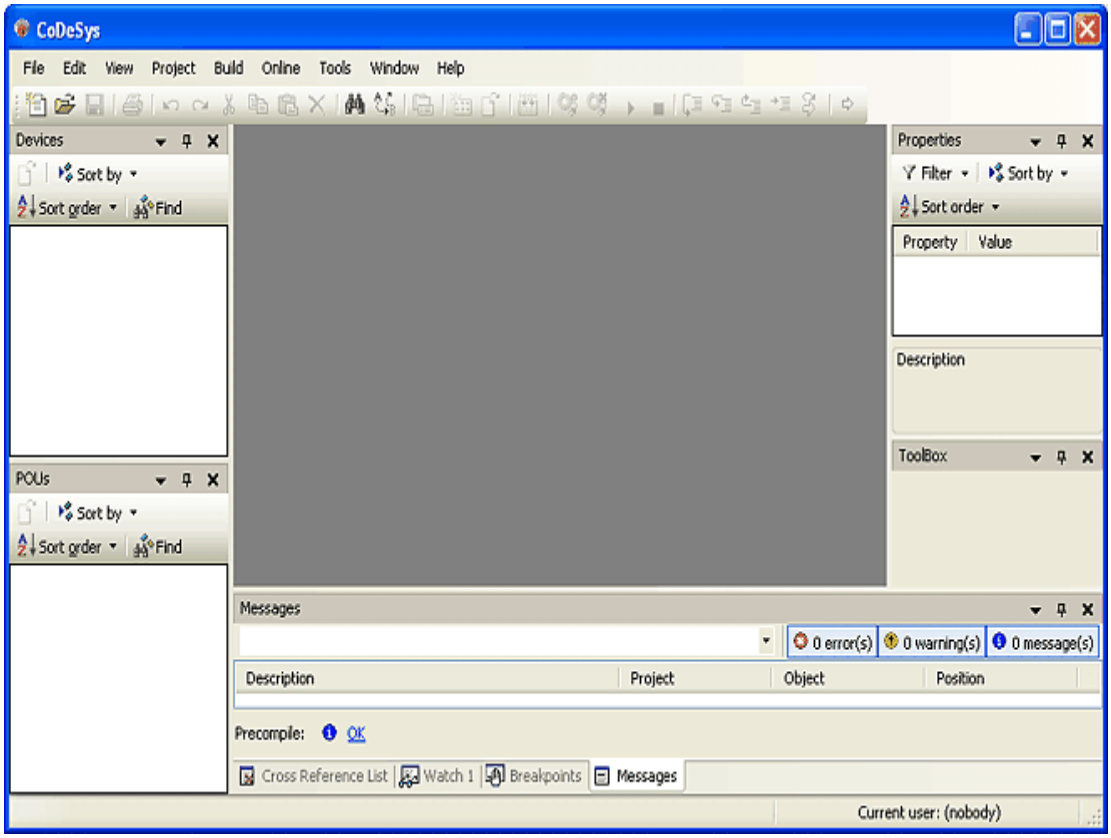
CoDeSys V3 – 快速学习 2/6

工程配置：

让我们开始学习。双击 CoDeSys V3 的图标。

请注意：不同的安装版本，图标有不同的标题。

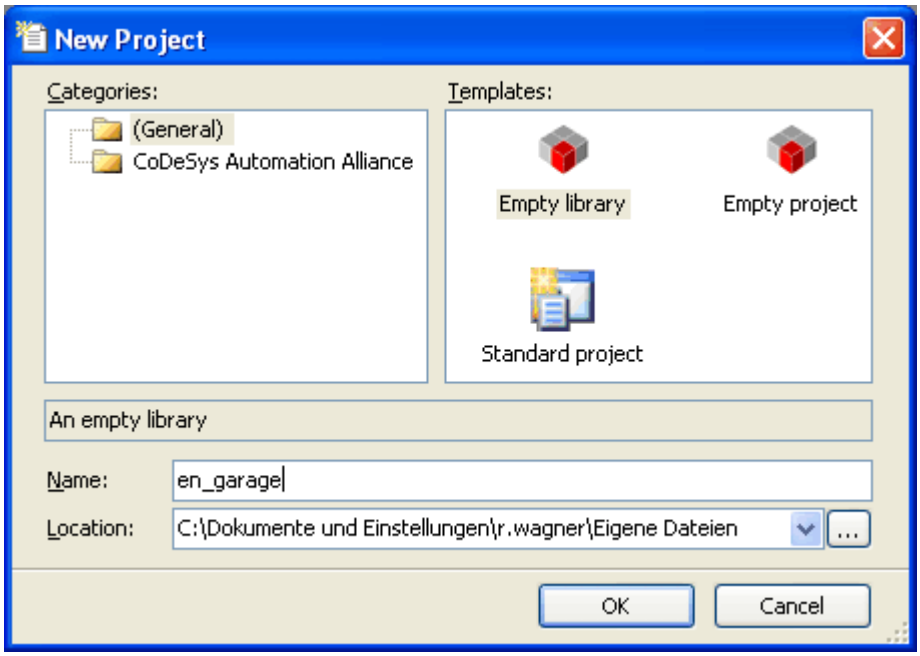
CoDeSys V3 是基于.net 平台，当 CoDeSys V3 启动的时候，这个安装版本的所有需要的的组件都要进行初始化。在此期间，您可以关注一下窗口中的进度条。



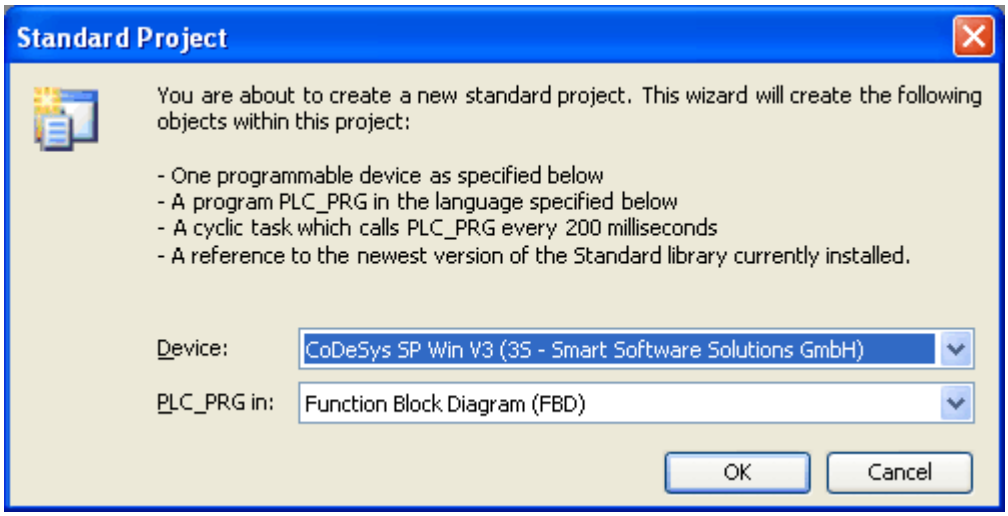
我们可以看到，屏幕左边是空的项目树窗口，由设备标签（device tab）和程序组织单元标签（POU tab）组成。由于这个例子中，我们不需要任何独立于设备的 POUs，所以我们可以暂时忽略程序组织单元标签。在右手边，我们可以看到一个属性窗口和一个工具箱窗口。属性窗口暂时不需要，可以关闭。位于屏幕底部的是消息窗口，将会显示一些有用的信息，主要是编译报错信息。

因为 CoDeSys V3 的窗口布局可以按照用户的特定需要来布置，所以可以和上面描述的布局不同。当你打开 CoDeSys 窗口时，可能是两个独立的窗口（Devices，POUs）就像上面的图片一样出现在屏幕的左手边，而不像上面描述的是单一的对象树和两个标签页。这两个独立的窗口可以分别放置、缩放和关闭。在菜单'视图'下，我们可以打开所有的窗口。通过打开菜单“工具 / 定制化”，然后点击“恢复到默认状态”，可以恢复到 CoDeSys V3 刚打开时的默认布局。

点击菜单“文件/ 新建工程”或者图标  我们可以打开一个工程向导。

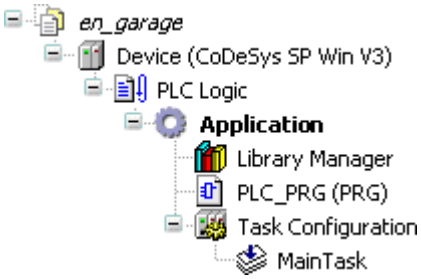


在这个对话框中，我们可以选择一个标准的工程，为其命名，并决定存储位置。

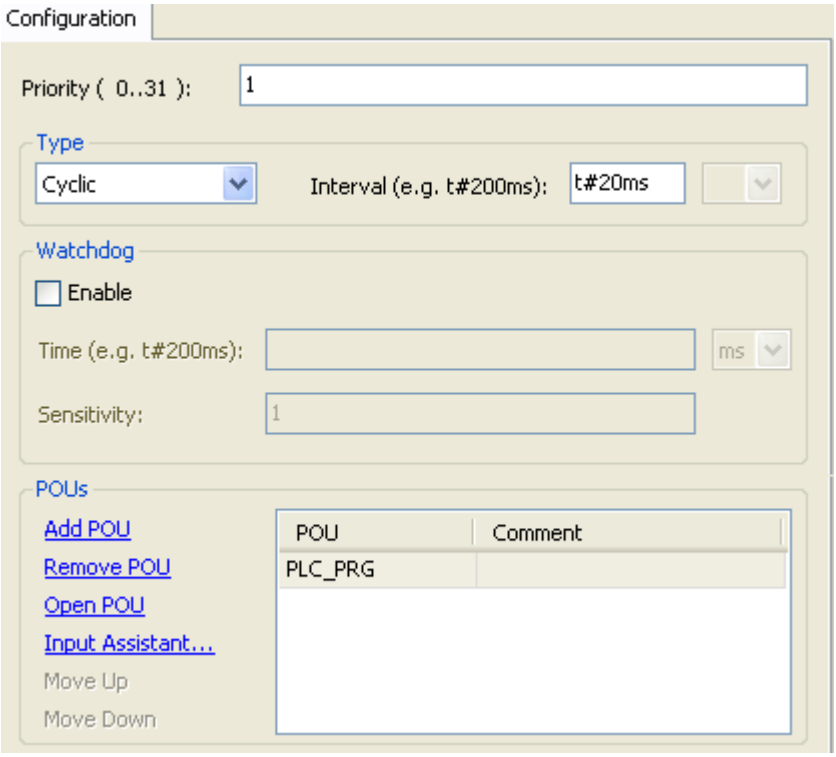


为了能够生成这个标准工程，我们必须告诉 **CoDeSys**，我们打算使用什么样的目标设备和选择哪种 IEC 61131- 3 的编程语言来编写我们的主要任务。在我们的车库应用程序中，我们将使用功能模块图语言（FBD）。我们将选择软 PLC “**CoDeSys SP Win V3**”来作为目标设备，**CoDeSys SP Win V3** 可以安装后自动启动。我们或许需要检查一下，软 PLC 的版本是否和编程系统的版本是对应的。一般来说，这是由系统自动来做的。Windows 任务栏中的图标  告诉我们软 PLC 已经激活。如果软 PLC 的图标是灰色的（  ），这是由于演示模式（DEMO）下的运行系统的运行时间是 2 个小时，运行 2 个小时后自动停止了。这时，您只需用鼠标右键点击软 PLC 的图标，然后点击“启动 PLC”即可。

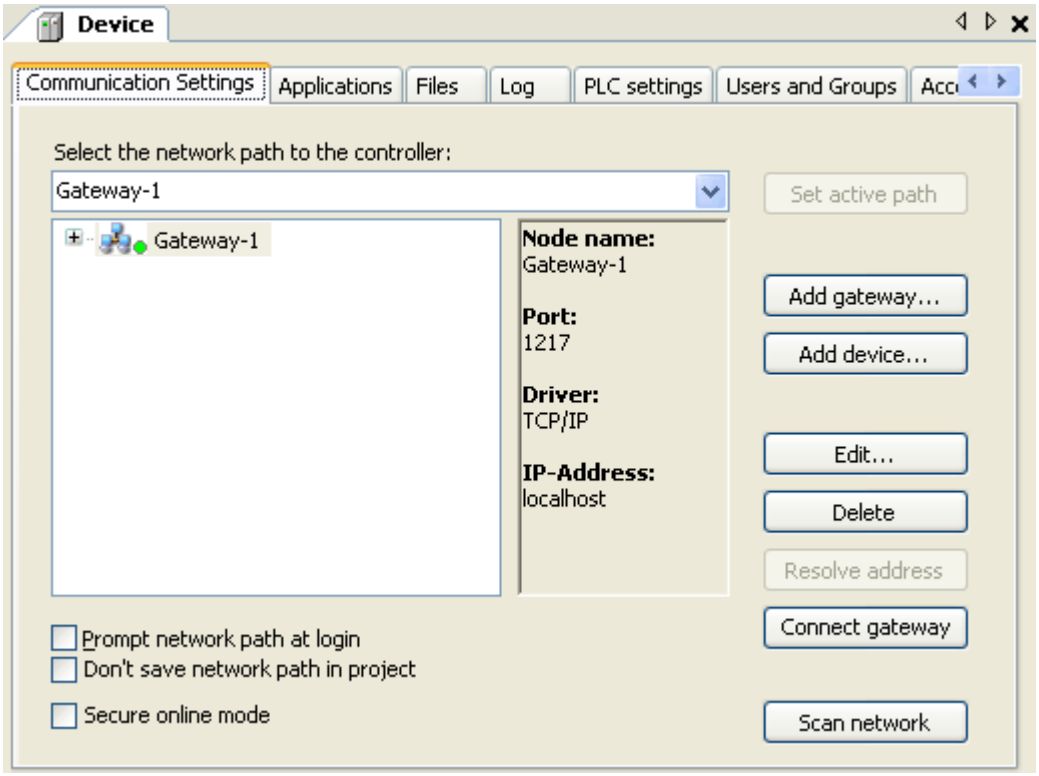
现在向导创建的对象树的基本配置如下：



这棵对象树包含一个带有 **MainTask** 的任务配置，这个任务配置包含在主程序 POU PLC_PRG 中。通过双击自动创建的 **MainTask**，我们可以在打开的窗口中设置间隔的扫描时间，来满足我们的需求。



为了把我们电脑上的软 PLC 与编程系统 **CoDeSys** V3 连接起来，我们现在必须双击对象树中的设备图标 

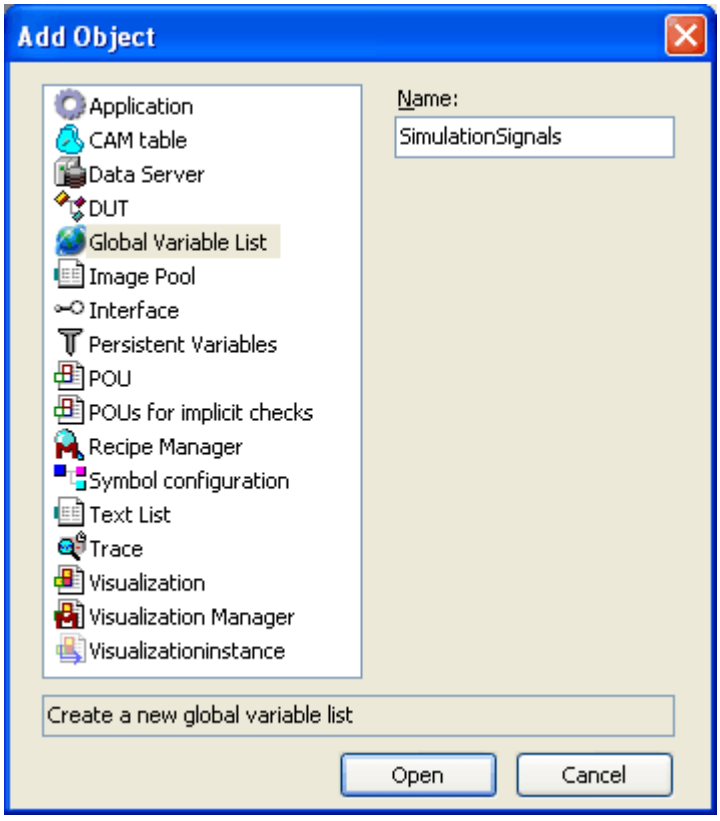


如果“**Gateway-1**”没有自动出现在设备窗口中，请单击按钮“**添加网关**”。现在我们选择左边的网关服务器（**gateway sever**），点击右边的“扫描网络”，运行在我们 PC 上的软 PLC 就出现在网关服务器的目录树下。现在我们选择这个条目，点击按钮“设置使用路径”。被选的设备例如运行在我们的 PC 上的软 PLC 现在会以粗体显示，并已经连接到编程系统。现在工程配置已经成功设置，我们可以关闭设备窗口了。

CoDeSys V3 – 快速学习 3/6

变量声明

在我们的例子中，我们计划在全局变量列表中声明输入和输出变量。如果我们右键点击应用程序的图标 ，选择“添加对象”，我们就可以增加不同的对象到我们的应用程序中。我们将选择“全局变量列表”并命名为“仿真信号”。



需要声明输入变量是 DoorOpen，DoorClosed，DoorOverloaded（用作传感器）和 Actuation（用作控制按钮）。需要声明的输出变量是 DoorUp 和 DoorDown 作为驱动，Lighting 作为灯的驱动。

SimulationSignals [Device: PLC Logic: Application]

```
1  VAR_GLOBAL
2      //Inputs
3      DoorOpen, DoorClosed, DriveOverload: BOOL;
4      Actuation: BOOL;
5      //Outputs
6      DoorUp, DoorDown, Lighting: BOOL;
7  END_VAR
```

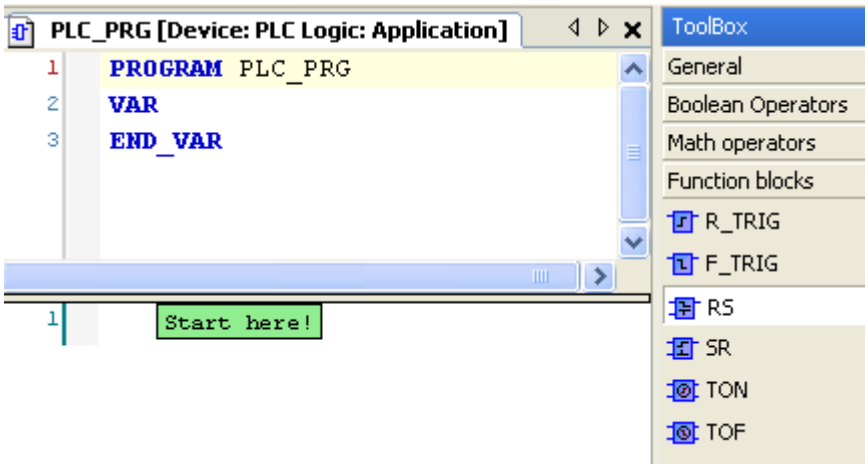
所有的 I/O 变量都是布尔类型变量。

CoDeSys V3 - 快速学习 4/6

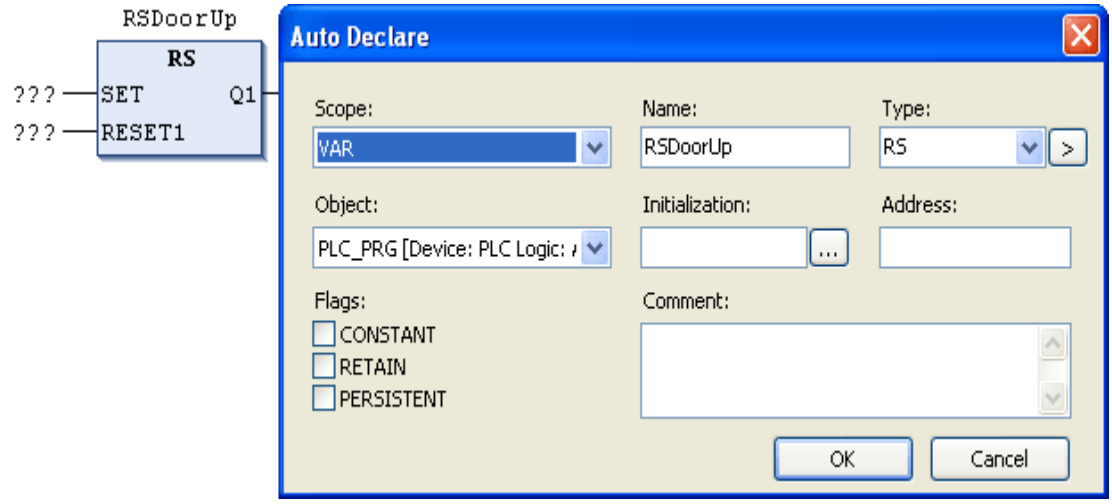
车库门控制的编程

编写应用程序的第一件事情就是双击设备树中的主程序单元 POU“PLC-PRG”来打开主程序编辑器。

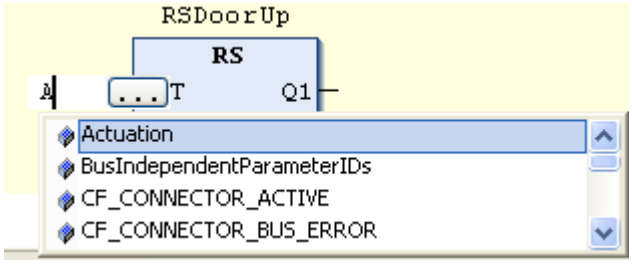
首先，我们将编写打开车库门这一部分的应用程序，我们将使用一个变量功能块来实现它。



我们可以从工具箱窗口中选择 RS 功能块（在功能块的目录中）将其拖到 POU 编辑器中的指定位置（“Start here”）。该功能块初始化后，分配到一块内存区域，在功能块的上部有“???”的地方书写该实例的名字。写完名字后，我们只要一按回车键，变量自动声明的对话框就会自动打开。实例的名字和变量类型（如这个功能块 RS）就已经输入好了。通过点击 OK 按钮，声明就会传到 POU PLC_PRG 的本地变量声明区（在 FBD 编辑器的上部）。

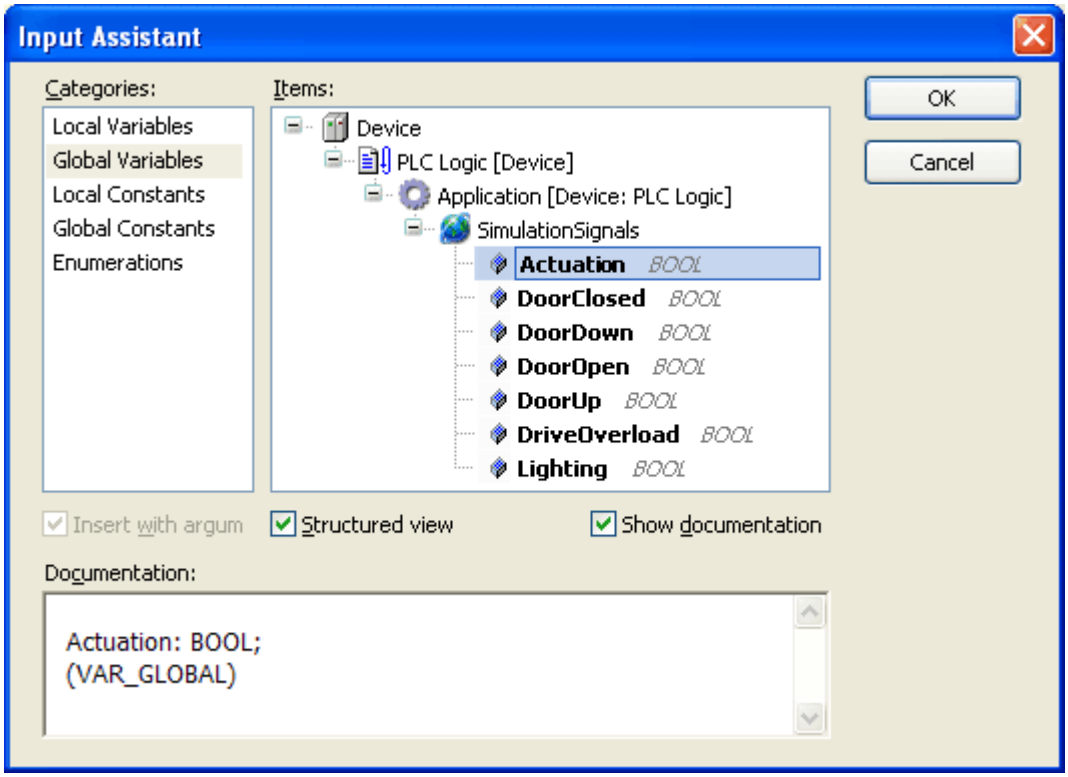


通过使用 RS 功能块，只要设置功能块的输入端发送所需的条件比如来自控制按钮的上升沿，执行部件 DoorUp 就能很容易闭锁。在设置输入端输入控制按钮变量 Actuation。在输入端的三个问号上输入变量的第一个字母时，**CoDeSys** 的智能输入就列出了所有可用的变量，变量 Actuation 就可以从列表中选择一个回车输入。

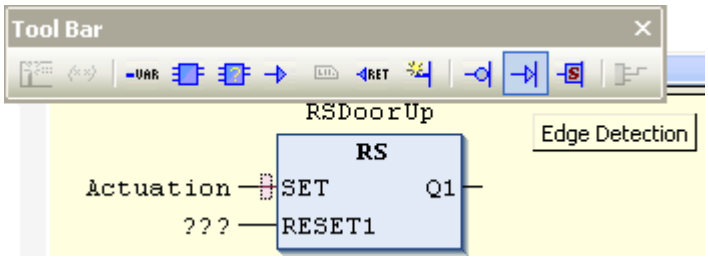


在标记了输入后，通过按 F2 在线帮助键可以实现同样的效果。和前面相同，我们仅仅需要

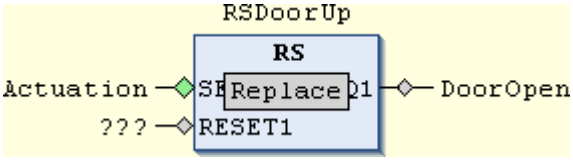
从全局变量的目录中选取我们想要的变量。



当控制按钮没有被按时，布尔变量 Actuation 的值为假，按下去时，变量值为真。这就意味着这个变量是一个上升沿。为了确保输入的是一个上升沿，我们可以使用 FBD 编辑器集成的沿检测功能。通过点击输入前的标记，激活工具条，点击工具条中的沿信号。图标  表示为上升沿，再点击一次变成下降沿。

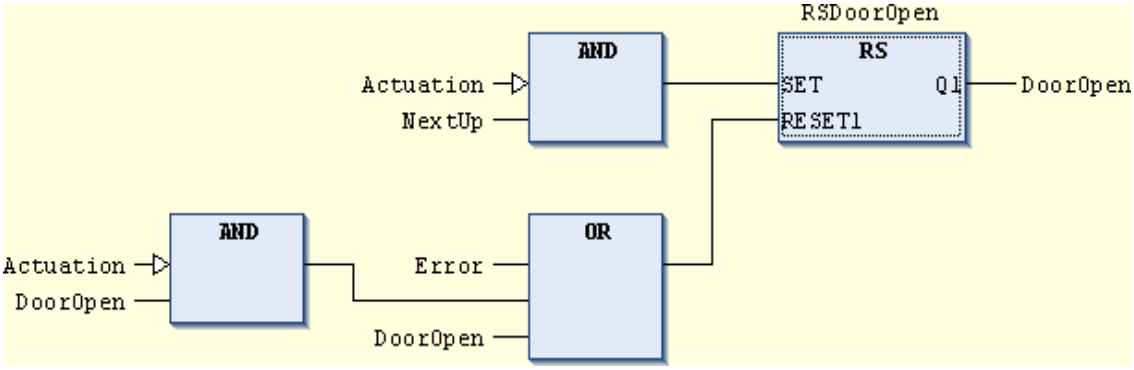


为了确保控制按钮按下时车库门不是每次都会打开，我们必须引入一个附加变量，用来表示门正在朝上移动，或者在朝下移动：仅当 NextUp 为真时，门可以被打开。这个新的布尔变量，就像功能块实例一样，使用自动声明对话框来声明，和先前的变量 Actuation 是与（and）的关系，输入这个功能块。可从右侧工具箱/布尔操作符的列表中，点击操作符 AND（2 输入）拖到功能块的输入设置（Replace 方框）位置即可实现。控制按钮变量 Actuation 自动移动到 AND 操作符的第一个输入端。功能块的 I/O 前的钻石符显示了可能的插入点。如果你把操作符放在绿色钻石附上，则操作符就插入到这个地方。下一步，我们将连接变量 DoorOpen 到 RS 块的输出 Q1 上。鼠标右键点击输出端，选择命令“插入输出”。在“???”的地方，输入变量。

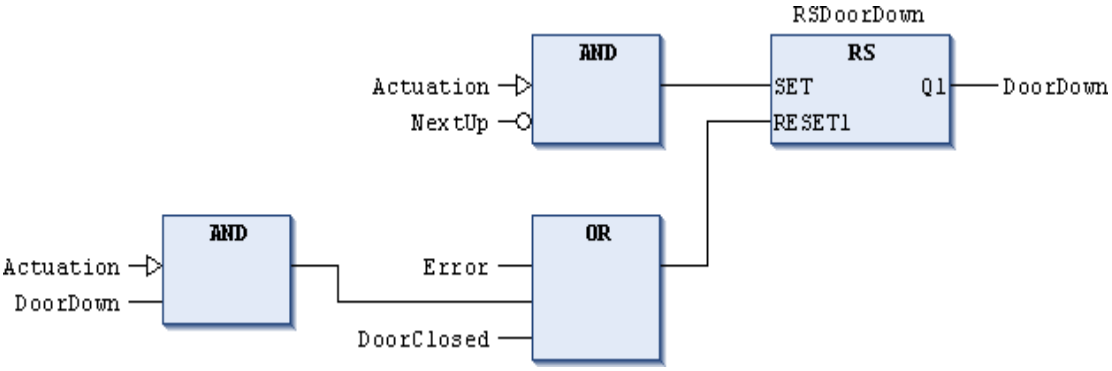


由于有三种可能的原因停止车库门打开的过程，所以我们放置一个 OR 操作符（3 输入）在 RS 功能块的 RESET1 输入端。三种可能的原因被连在 OR 操作符的输入端：DoorOpen 传感器报告车库门已经到达它的最高位置。这个变量已经被声明，可以直接连接到 OR 操作符的输入端。用户按下控制按钮的同时，执行部件 DoorUp 被激活。通过插入一个 AND 操作符（2 输入），连接控制按钮变量 Actuation（带有沿检测），变量 DoorUp 连接到操作符的输入端，来查询这种情形。由于驱动过载，或者过长时间的 Door-up 或者 Door-down 都会引起错误。这个错误需要用到第二个附加变量，并且在自动声明的对话框中声明为一个布尔变量。

现在我们的第一个网络完成如下：

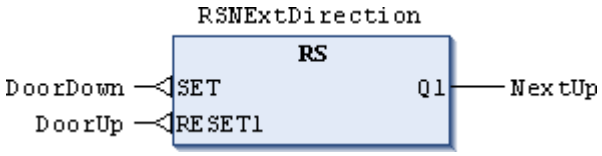


第二个网络将负责关闭车库门，因此几乎和第一个网络是相同的。我们可以通过点击编辑器的左侧选取并且复制粘贴第一个网络，或者也可以在选取第一个网络后，使用快捷键“Ctrl+C” 和“Ctrl+V ”。这样，第二个网络现在就出现在第一个网络中。当然，我们现在需要将负责开门的变量替换为负责关门的变量：



同样重要的是，负责辨识车库门上移，还是下移的变量必须取反。这需要单击 NextUp 后边的横线，然后工具栏中的 从插入到横线上。第三个网络用来设置附加变量，来负责辨识门是在上移还是下移，使用变量 NextUp。通过从工具箱目录‘常规’拖拽新网络插入并且放置在现存网络的下方。首先，我们可以将熟悉的 RS 功能块插入到这个网络当中。通过在功能块的上方输入实例名字“RS Next Direction”来初始化该功能块，我们将变量 NextUp 放置在输出端。因为智能输入功能的支持，只要输入首字母，我们可以得到一个备选变量的

列表，供我们选择。先前的动作可以推测出后来可能发生的动作：例如，如果执行部件 DoorOpen，负责打开车库门的部件，发出一个下降沿（逻辑值由真到假），我们就知道了车库门打开过程结束。因此，下一个动作将是车库门的关闭。可以将一个带有下降沿检测的变量 DoorDown 来激活这个沿检测，并且将这个变量插入到这个 RS 功能块的输入设置端（通过双击工具条中沿检测符号  可以变为下降沿检测符号）。同样的做法可以应用在 DoorUp 上，将该变量插入到 RESET1 输入端。



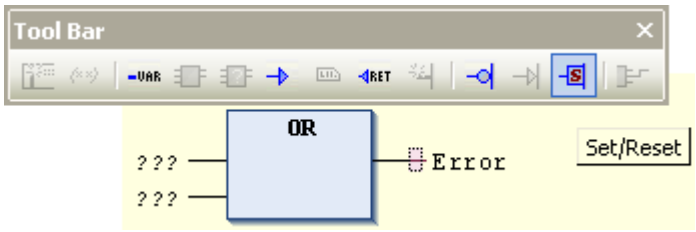
所有需要控制车库门的步骤都已经完成。现在仍缺少的就是灯光的控制和对车库门错误监控的控制。

CoDeSys V3 - 快速学习 5/6

关于错误监控和控制灯光的编程

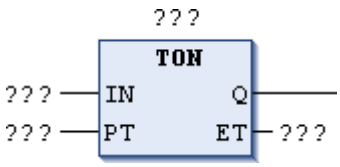
当我们编程第一个网络时，我们引入了变量 Error，用来报告错误和车库门的停止。这个变量可以如下编程：

当传感器变量 DriveOverload 设置为真的时候，或者开关车库门的时间超过了 20 秒的时候，就会有错误发生。为了重现上述的情景，我们需要在一个新网络中插入一个 OR 的操作符。这个 OR 的操作符的输出端设置为变量 Error。可以通过鼠标单击输出端，并且点击工具条中的“Set / Reset’”按钮，直到一个 S 出现在输出管脚上。这样的话，连在管脚上的变量就可以通过点击“Set / Reset”按钮来复位。



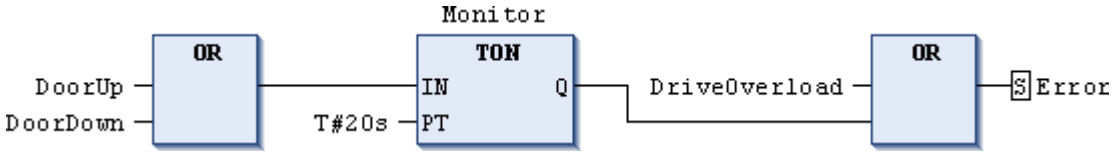
该功能块的第一个输入将分配为传感器变量 DriveOverload。

为了能够检测开关门的最大时间跨度是否被超过了，我们需要一个时间监控机制。恰好我们从工具箱引入了功能块 TON（Timer On Delay 延时通），并且将它初始化命名为 Monitor。

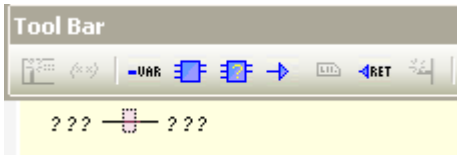


一个布尔为“真”的量连接到该模块的输入端，在过了一定预置时间（PT）后，布尔量“真”就被写入到 Q 输出端。但是，如果在预置的时间段之内，输入端 IN 转换到“假”，那么输出仍旧保持“假”。

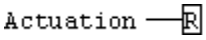
一旦我们的车库门开始移动，例如开或者关（执行变量 DoorUp / DoorDown 代表这两个动作），TON 功能块就开始工作。在正常的操作中，两个传感器变量 DoorOpen 或者 DoorClosed 中的一个将在 10-20 秒内结束门的运动。执行部件变量设成“假”，定时器模块的输出也保持“假”，没有错误报告。如果整个流程占用的时间超过了 20 秒，TON 模块的输出 Q 就会变为“真”，并且报错。TON 功能块的第二个输出 ET（消逝时间），是用来报告从定时器开始工作所消逝的时间。由于在我们的实例中，我们不需要这个输出，我们可以通过鼠标左键单击输出管脚选中，按 Del 键将这个管脚删除即可。既然时间监控机制应用于两个执行部件变量，我们将在这两个执行部件变量和 TON 功能块中间插入一个 OR 操作符。



输入预设的时间量的时候，必须按照 IEC 61131- 3 的格式：“T#”需要作为时间规范的前缀。用户可以通过按控制按钮 Actuation 来复位错误。为了实现我们的应用，我们必须创建一个新网络。一种方式是从工具箱的的常规菜单来拖出实现，另一种是在最后一个网络左边的内容菜单，点击右键，选择“插入网络（下方）”。一旦我们创建了新网络，我们现在就能输入一个新任务到这个网络里。有三种方式来输入这个新任务：从工具箱的常规菜单中拖出，鼠标右键打开智能选择菜单，或者使用快捷键“**Ctrl+A**”。



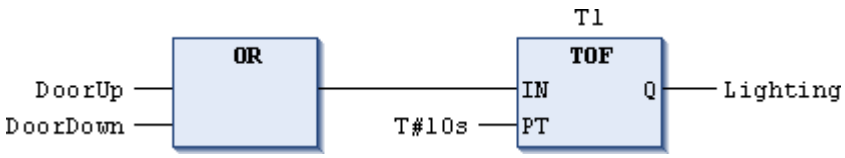
变量 Actuation 写到任务的输入端，它的值分配给变量 Error。但是在这个案例中，变量 Error 必须可以重置，所以我们必须在输出端设置“Reset”R (从工具条中拖出，或者使用鼠标右键，或者使用组合键 “**Ctrl+M**”)。



应用程序的最后一个网络是负责开灯的。如同以上所写，当车库门打开或者关闭的同时，车库门的灯会自动打开，持续一段时间后关闭，比如 10 秒钟。这就意味着，我们需要另外一个定时功能块。因为我们在一定的时间段之后，我们需要关闭布尔信号，因此我们需要功能块 TOF（Timer Off Delay 延时断）。把 TOF 功能块拖入网络后，我们必须把该功能块实例化，命名为“T1”。在我们的例子当中，ET 管脚仍然没有用，可以直接删除了。

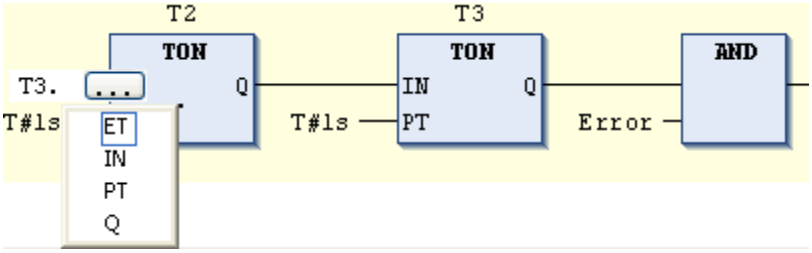
在 TOF 功能块的输出端赋值变量 lighting，将上述的时延 10s 按照 IEC 61131- 3 的格式填写到输入端 PT。输入端 IN 可插入 OR 运算符，可以通过工具箱拖拽，也可用鼠标右键选取，也可使用组合键“Ctrl+B”。执行部件变量 DoorUp 和 DoorDown 直接与 OR 运算符的两个输入端相连。第二个输出端 ET 代表已经消逝的时间，在本例子当中没有什么作用，可以鼠标单击选中，直接按 Del 键把这个管脚删除。

我们现在也完成了车库门驱动无故障操作的灯光控制。

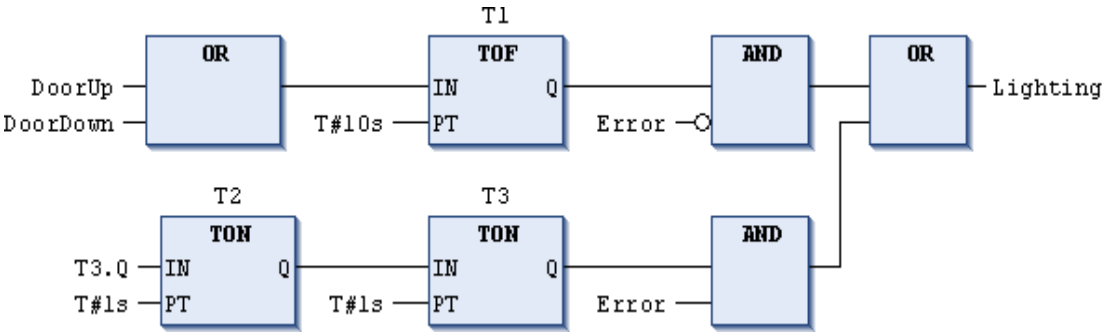


为了能在同样的网络里实现我们的错误情景，我们必须在变量 Lighting 之前放置 OR 操作符。这个 OR 操作符使得为现存网络增加一个新线程的可能性，这个网络主要负责错误发生的处理。另外一个线程仅在无故障操作的时候需要。所以，我们现在需要在 OR 操作符的每一个输入端连接一个 AND 操作符。第一个 AND 可以插入到现存的现场中，正常的操作下，控制灯光之前。第二个 AND 的输入分配给一个取反的变量 Error。结果就是，如果没有错误发生，整个线程将仅仅影响输出灯光控制。

在我们的负责错误的步骤中的 AND 操作符的第二个输入也分配给变量 Error，但是这次不用取反符号。结果是，如果错误确实发生，整个过程将仅仅输出灯光控制。在负责错误的线程中的 AND 操作符的第一个输入需要灯光闪烁，所以我们串接了两个 TON 功能块。左边 TON 功能块的输出连接着右边 TON 功能块的输入，右边功能块的输出依次连接着我们的 AND 操作符的第一个输入。两个 TON 的预定时为 1s，ET 管脚删除。当然，我们必须实例化两个模块为 T2 和 T3。左边 TON 的输入管脚 IN 是右边 TON 的输出管脚的值。要实现这种方法，可以在左边的第一个输入管脚处输入右边 TON 功能块的实例名 T3.，**CoDeSys** 就知道了我们现在想使用功能块 TON 数据结构中的一个变量，智能输入窗口就会列出 T3 所有的变量供我们选择。



在正常工作或者错误操作下的，控制灯管的网络已经完成。这也意味着我们成功的编写了我们的应用程序

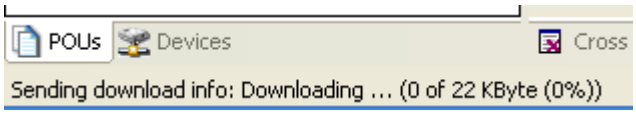


CoDeSys V3 - 快速学习 6/6

测试应用程序

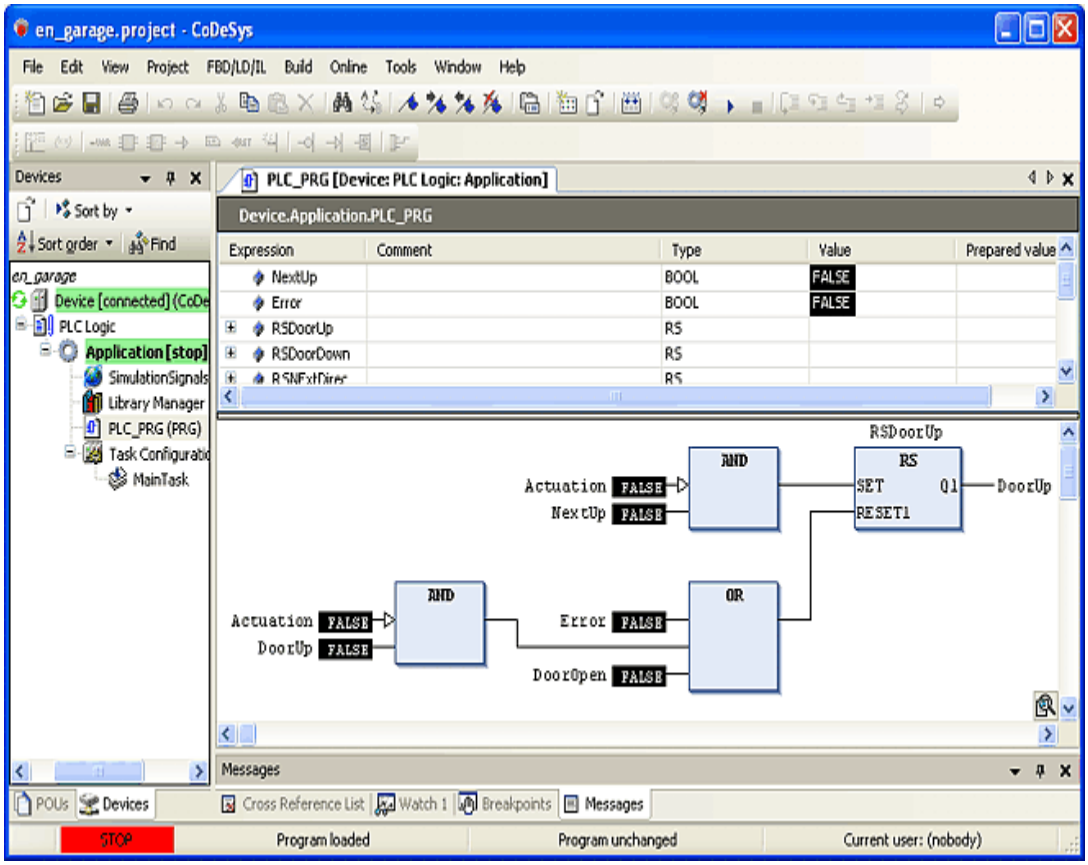
现在我们完成了整个车库门应用程序的编写，我们需要做些测试来确保每个功能都是正确的。但是我们没有一个真正的带有车库门驱动以及所有传感器和执行部件，我们只能人工仿真这个应用程序。**CoDeSys** 提供了一整套完善的、综合的调试功能来帮助我们测试应用程序。

首先，我们登录控制器，下载应用程序。这个过程中，**CoDeSys** 使用集成的编译器，将图形 FBD 代码翻译成真正的机器代码，这些可以运行在多种不同的 CPU 和操作系统之上。而示例中运行在 PC 上的软 PLC 是我们在编程之前就在工程配置当中配置好的（参见单元 2 中工程配置）。当然，我们可以将我们的应用程序运行在任何可以使用 **CoDeSys** 编程的设备和控制器上。我们可以从菜单“在线”选取命令“登录应用程序 [名称] ”。**CoDeSys** 开始检查应用程序是否已经运行在目标设备（软 PLC）上了；如果没有应用程序在运行，那么会跳出一个对话框，“设备上没有应用程序，你想创建并且下载应用程序吗？”如果你确认了这个对话框，**CoDeSys** 就编译这个应用程序并且下载到目标设备上。在编程窗口的底部的左手边进度条会显示程序代码的编译下载状态。

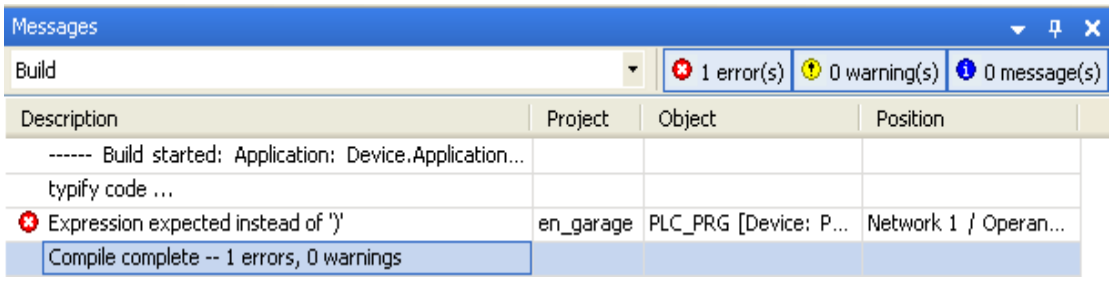


只要下载顺利地结束了，下列状况和事项就会发生：

- 我们创建的应用程序的 FBD 编辑器的代码部分从编辑模式变为了在线模式，所有变量的当前值都会显示出来。
- 所有编辑器的变量声明部分都变为了在线模式。现在我们不能再编辑变量声明部分，但是所有的变量值都可以显示，也可以被改变（这种功能可以用来测试我们的应用程序）。
- 在左手边的设备树当中，被编程的设备前边出现一个绿色的符号，而在我们的例子中，显示为“Device [connected] (CoDeSys SP WinV3)”，显示对设备的连接已经建立。整个字段变为绿色的高亮显示。
- 设备下面的已经下载成功的应用程序也是绿色高亮显示。“应用”字样后边的[停止]显示应用程序还没有开始启动。
- 整个开发环境的底部的状态条中红色“STOP”也是显示应用程序没有启动，同时显示“程序下载”。



如同上面显示，建立一个连接仍旧不行，还需要检查通信设置，工程配置的最后提到了如何进行配置。如果我们在应用程序代码中有一些语法错误，以至于不能编译通过时，**CoDeSys** 将发布以下消息“存在编译错误，你打算不下载此程序就直接登录吗？”通过点击“否”按钮来关闭消息窗口。我们试着来纠正这些错误。**CoDeSys** 检测到的所有的错误和这些错误的位置都会列表在消息窗口当中（通常在我们屏幕的最低位置）。现在最好按照错误列表的提示一步一步地去浏览错误。如果我们按“F4”键，**CoDeSys** 就从错误消息跳到应用程序代码中出错的地方。

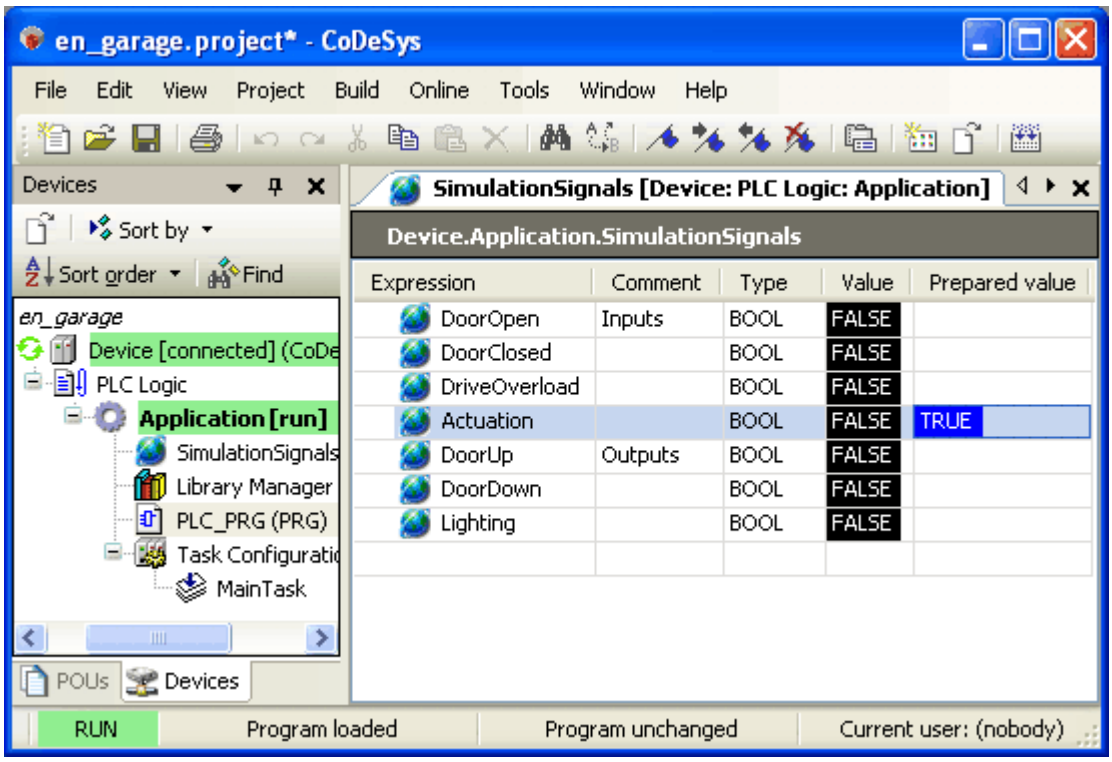


我们可以把所有发生的错误代码检查一遍并且校正过来。例如典型的错误，FBD 功能块的输入管脚仍然开放着，没有赋给输入值。一旦我们已经纠正了这个检测到的错误，我们则可以通过按 F4 键跳到下一个有错误的地方。纠正了检测到的错误之后，再按 F11 键来重新测试应用程序。这样，应用程序就已经编译好了，但是本地代码并没有下载到控制器当中。如果没有其它错误被发现的话，消息窗口将出现以下信息：

“编译完成 –0 个错误，0 个警告”。

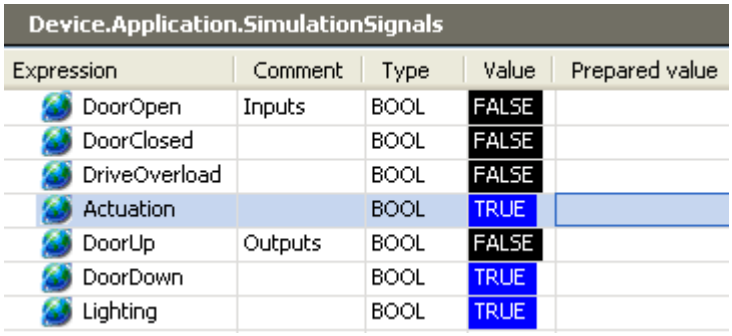
通过点击菜单栏“在线”中的“登录到 Application[Name]”选项，或者点击工具栏中  按钮来再次登入控制器。然后，按“F5”键，或者点击菜单“在线/启动 Application [Device：PLC Logic]”。现在设备树中 Application 后面的字显示为[运行]，屏幕底部的状态栏中红色“停止”变为绿色“运行”。在应用程序代码中，例如在 POU PLC_PRG 中，所有的布尔变量设置为假。只有最后一个网络当中的错误监控的闪烁信号每秒都从真变到假。为了能够测试应用程序的功能，我们必须在全局变量的声明表中改变全局变量：驱动、DoorOpen、DoorClosed、以及 DriveOverload 变量。这些的实现可以是在我们的变量表“仿真信号”中直接操作我们的程序。

通过点击栏目**“准备值”**，我们可以改变布尔变量的值。



注意：

这样依次将单一值写入到控制器中，并不是最好的方式。最好是能在一个控制循环内，同时将多个值写入到控制器当中，这样可以避免机器设备或者流程的损坏。这就是为什么：为了将“真”值写入到 Actuation 中，必须先点击命令“在线 / 写入值”或者使用快捷键“Ctrl+F7”。一旦变量 Actuation 被写入值，车库门开始关闭，因为在第一次下载后，应用程序所有的变量值都被设置为“假”或者“0”（如果没有声明其它初始化的值）。



Device.Application.SimulationSignals

Expression	Comment	Type	Value	Prepared value
DoorOpen	Inputs	BOOL	FALSE	
DoorClosed		BOOL	FALSE	
DriveOverload		BOOL	FALSE	
Actuation		BOOL	TRUE	
DoorUp	Outputs	BOOL	FALSE	
DoorDown		BOOL	TRUE	
Lighting		BOOL	TRUE	

由于车库门正在关闭，DoorDown 的值被应用程序设置为“真”。我们的灯光及对应的变量为 Lighting. 正如我们所愿，当车库门在运行当中，灯光就会打开。为了能够跟随可编程的每一个单步，我们现在手动复位 Actuation 为“假”，然后仿真传感器变量 DoorClosed 被设置为“真”（然后回到“假”）。执行部件变量 DoorDown 被程序设置为“假”，门关闭。10 秒钟后，灯光自动关闭，变量 Lighting 设置为“假”。通过再次按控制钮，例如，设置变量 Actuation 为“真”，我们可以测试开门的整个步骤。涉及的变量就是开门所需变量 DoorUp 和 DoorOpen。

如果我们通过仿真传感器变量 DoorClosed 或者 DoorOpen，一直没有对程序的开始或者结束进行操作，在程序开始后，经过预置的 20 秒后，应用程序就可以辨识错误。结果就是，执行部件 DoorDown 或者 DoorUp 被设置为“假”，灯光开始闪烁，变量 Lighting 从“真”变到“假”。再次按动控制按钮（Actuation 为真），错误被通知，灯光停止闪烁，我们已经回到了我们的初始状态。错误 DrivOverload 可以通过手动设置为“真”同时门开或者关来进行测试。车库门再次停止，错误显示灯光开始闪烁。

下载整个应用程序的 CoDeSys V3 的工程文件

CoDeSys V3 工程文件可以使用 CoDeSys V3 来打开，点击菜单命令“文件/ 工程文件存档 / 解压存档”。

提示：请注意：

我们的示例程序可以在 CoDeSys V3.3（或者更高版本）当中执行。

对于 **MS Internet Explorer** 的用户：

下载后请对文档重新命名：可以将文件的扩展名改回到*.projective.

对于 **Mozilla Firefox** 的用户：

请不要在浏览器内打开文件，选择“目标另存为……”。

我们准备了两个不同版本的工程文件：

1. **原始版：**就是快速学习的例子中所引入的，同时您也可以从 YouTube 视频网站下载。
2. **带有集成的车库仿真版本。**

除了车库门控制程序的代码，这个工程中还包括了仿真模块。集成的可视化模块使它变得更容易测试。

小提示：在可视化编程环境中，你可以通过点击轮子来移动汽车。

我们认为仿真和可视化是非常基础性的，它为您的创造力留下了必要的空间。

祝您好运！