人机界面 **HMI**

本篇包括的主题内容：

第十二章 可视化

12.1 概述

可视化是项目变量的图形化表示，联机模式时允许通过鼠标和键盘输入到 PLC 程序中。TwinCAT 可视化编辑器是编程系统的一部分，提供的图形元素可以按照需要进行布局，并且可以与项目变量进行连接。因此，联机模式时这些图形元素的外观将随对应的变量值变化。

简单示例：为了表示一个填充值大小，该值在 PLC 程序中计算，可以绘制一个棒图并将其与对应的项目变量进行连接，这样，棒图的长度和颜色将显示当前填充值的大小。添加一个文本框可以使文本显示当前值，添加一个按钮则可以启动和停止该程序。

单个可视化元素以及所有可视化对象的属性将在相应的配置对话框和对象属性对话框中进行定义。可以在此处通过激活选项设置基本参数，并且通过输入项目变量定义动态参数。

另外，还可以通过设置结构变量使元素具备可编程能力的属性。

在配置对话框中使用占位符可以提高效率，当你想多次使用相同的可视化对象，而其配置又不相同时尤为明显。

编程系统创建的可视化在很多情况下可以作为唯一的用户接口，用于联机模式时控制和监视关联的 PLC 程序。为此，必须为程序提供独立的输入，激活可视化元素。为实现该目的，在可视化配置期间，可以使用特殊的输入特性，并可以为不同的可视化选择定义特殊的快捷键。

使用 TwinCAT 创建的可视化可以在以后使用不同的方法进行运用：

可以使用 TwinCAT HMI 进行制作，即专用的运行系统 - 用于 PLC 计算机的全屏模式可视化操作。

也可以作为 Web-可视化，即通过 Internet 进行调用和操作。（用于远程维护之目的）

另外，还可以实现目标系统的可视化，直接在 PLC 上运行。

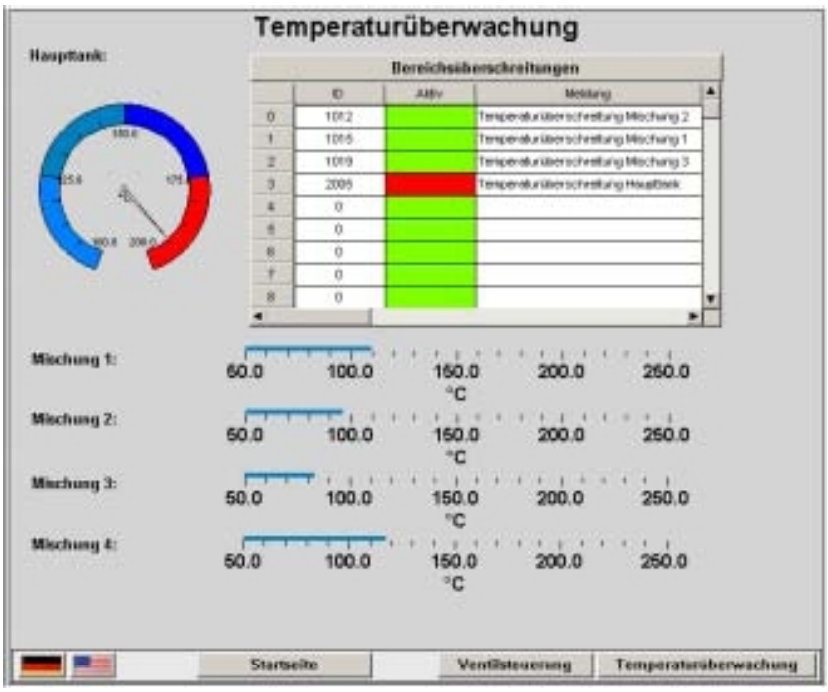


图 12-1 可视化示例

12.2 新建可视化

可视化对象是 TwinCAT 对象，并在对象管理器的“可视化”（'Visualization' ）属性页中管理。它包括可视化元素的布局和操作对象属性。一个或几个可视化对象可以在 TwinCAT 项目中创建，并且彼此进行链接。

为了在对象管理器中创建可视化对象，必须在对象管理器中选择  可视化属性页。

使用“项目”（'Project'）、“添加对象”（'Object Add'） 命令，可以创建一个新的可视化对象。打开“新可视化”（'New visualization' ）对话框，你可以在此处输入新可视化的名称。一旦输入了一个有效的名称，即该名称没有被使用过，并且不包含特殊字符，则可以使用 OK 按钮关闭该对话框。一个窗口将被打开，你可以在此编辑这个新的可视化对象。

当一个可视化对象在对象管理器中被标识时，通过命令“项目” （'Project' ）“对象”（'Object'）“属性”（ 'Properties' ），打开属性对话框，可以在此设置对象的 Web 服务器或目标可视化以及布局相关的使用方法。

请注意：如果你想使用隐含变量 CurrentVisu (类型 STRING) 用于寻址当前打开的可视化对象，则必须使可视化对象的名称全部为大写字符（示例： "PLC_VISU"）。（有关隐含变量信息参见相关章节）

12.3 可视化元素 - 插入

可视化元素是图形元素，用于构成可视化对象。TwinCAT 提供的可视化元素在菜单中显示。每个元素都有一个独立的配置。

你可以将各种几何形状，如位图、WMF 文件、按钮、各种专用元素和已有的可视化对象插入到你的可视化对象中。请注意：尽量为项目中的选项定义可视化文件的专用目录。

指向“插入”('Insert')菜单项，并自由选择下面的命令：  “矩形”('Rectangle')、  “圆角矩形”('Rounded Rectangle')、  “椭圆”('Ellipse')、  “多边形”('Polygon')、  “多边线”('Polyline')、  “曲线”('Curve')、  “饼图”('Pie')、  “位图”('Bitmap')、  “可视化”('Visualization')、  “按钮”('Button')、  “表格”('Table')、  “ActiveX-控件”('ActiveX-Element')、  “仪表”('Meter')、  “棒图”('Bar Display')、  “历史趋势”('Histogram')、  “报警表”('Alarm table')、  “趋势图”('Trend')、  “WMF 文件”('WMF file')。选择的命令前将出现一个检查框。你也可以使用工具条。选中的元素将以按下的按钮状态表示 （示例  ）。

如果你现在使用鼠标进入编辑器窗口，你将看到鼠标指针的形状与对应的符号相同（示例  ）。在你所需要的位置点击，并以此点作为你选择元素的起始点，按压鼠标左键并移动鼠标指针，直到达到所期望的尺寸时为止。

如果你想创建一个多边形或线段，首先用鼠标点击多边形的第一个角的位置（线段也是如此）。作为该线段的起始点，然后点击其它所需要的点。在最后一个点双击鼠标，多边形将闭合；并将绘出全部已完成的各个线段。如果你想创建一条曲线（贝塞尔曲线）(Bezier curves)，可以使用鼠标点击确定其初始点和其它两个点，以便确定绘制矩形。在第三次点击鼠标时将绘出一个圆弧。然后，你可以移动鼠标改变该圆弧的结束点位置，并双击鼠标结束此过程；或再点击鼠标添加其它的圆弧。

另外，请注意状态条中模式的选择和插入变化。同时，也应注意尽量使用占位符和布局特性。

12.3.1 “插入”（'Insert' ）“矩形”（'Rectangle'）

符号： 

使用该命令，你可以将一个矩形元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

12.3.2 “插入”（'Insert' ）“圆角矩形”（'Rounded Rectangle'）

符号： 

使用该命令，你可以将一个圆角矩形元素插入到你当前的可视化中。(用法，参件可视化元素

（Visualization Elements），插入（Insert））

12.3.3 “插入”（’Insert' ）“椭圆”（'Ellipse'）

符号： 

使用该命令，你可以将一个椭圆元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

12.3.4 “插入”（’Insert' ）“多边形”（'Polygon'）

符号： 

使用该命令，你可以将一个多边形元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

12.3.5 “插入”（’Insert' ）“多边线”（'Polyline'）

符号： 

使用该命令，你可以将一个多边线元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

12.3.6 “插入”（’Insert' ）“曲线”（'Curve'）

符号： 

使用该命令，你可以将一个曲线元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

12.3.7 “插入”（’Insert' ）“饼图”（'Pie'）

符号： 

使用该命令，你可以将一个饼图元素插入到你当前的可视化中。

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。你会看到一个椭圆元素，其中包含一条标识0°位置半径的线段。只要保持鼠标左键按下，你可以通过移动鼠标立即改变该元素的大小和位置。此时，有一个小黑色矩型框附加在该元素上，它指示围绕该元素的一个虚拟矩形的边角。

为了定义饼图的起始点和结束点的角度，可以通过点击鼠标选择该圆弧中半径线段上的结束点。只要你保持按压鼠标左键，并移动光标，将显示两个小矩形，它指示这两个角度位置。至此，它们可以被分别进行选择和移动。如果你想通过变量动态地定义该角度值，则可以打开配置对话框，并在“角度”（’Angle’）分类中输入所期望的变量名即可。

随后，你可以通过点击中心点，该点光标显示为对角十字箭头，并按压鼠标键移动（或使用箭头键）改变该元素的大小和形状。或者，你可以选择和移动该元素外侧的小矩形指示角。为了将元素移动到其它位置， 点击元素内部显示为垂直交叉的光标，并移动该光标。

12.3.8 “插入”（'Insert' ）“位图”（'Bitmap'）

符号： 

使用该命令，你可以将一个位图元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。用于打开文件的对话框将被打开。一旦你已经选择了所期望的位图，它将被插入到该区域中。你可以在配置对话框中进行定义，所链接的位图文件是进行储存，还是作为元素进行插入。

12.3.9 “插入”（'Insert' ）“可视化”（'Visualization'）

符号： 

使用该命令，你可以将一个可视化元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。并打开已有可视化的选择表。在你选择了所期望的可视化之后，它将被插入到所定义的区域中。

插入的可视化也将被命名为“实例”（"instance"）。

12.3.10 “插入”（'Insert' ）“按钮”（'Button'）

符号： 

使用该命令，你可以将一个按钮元素插入到你当前的可视化中。(用法，参件可视化元素（Visualization Elements），插入（Insert））

按压鼠标左键的同时拖拉该元素至所期望的尺寸大小。

如果有一个切换变量链接至该按钮，它将显示该变量的状态，即：按钮是否被按压。相反，该变量可以通过按压该按钮进行切换。

12.3.11 “插入”（'Insert' ）“WMF 文件”（'WMF file'）

符号： 

该命令用于插入一个 Windows 媒介文件。并显示打开文件的标准对话框，你可以在此选择一个文件（扩展名 *.wmf）。当用 OK 关闭该对话框之后，该文件将被作为元素插入到可视化中。请注意，如果文件没有链接将不会被保存，这与插入一个位图时的情况相同，但媒介文件元素可以形成一个组插入。

12.3.12 “插入”（’Insert' ）“表格”（'Table'）

符号： 

使用该命令可以插入一个表格元素到你当前的可视化中。它对于显示当前数组元素值是非常有用的。

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。在显示该元素之前，将打开“配置表格”对话框（”Configure table"）。此时你将发现，除标准的工具条提示（Tooltip）和安全（Security）分类之外，又增加了‘表格’（'Table'）、‘列’（'Columns'）、‘行’（'Rows' ）和‘选择’（'Selection'）分类，并在此定义该表格的内容和外观。

12.3.13 “插入”（’Insert' ）“ActiveX 控件”（'ActiveX-Element'）

符号： 

使用该命令可以插入一个 ActiveX 控件到你当前的可视化中。并可以作为以后在 TwinCAT HMI 的 32 位 Windows 系统和目标可视化系统中使用。按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。并将标题为 “Control” 的矩形插入。

可以通过命令‘附加’（’Extras’）配置（’Configure’）选择 ActiveX 控件、配置调用方法以及在该元素上双击打开‘ActiveX 控件’对话框。

12.3.14 “插入”（’Insert' ）“仪表”（'Meter'）

符号： 

使用该命令可以插入一个仪表元素到你当前的可视化中。它提供一个带有刻度的圆弧扇区和一个指针元素。

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。在显示该元素之前，将打开“配置仪表”对话框（”Configure Meter"）。在这里，你可以定义各种有关元素显示的参数，并在确认对话框实际插入该元素之前对配置进行预览检验。

12.3.15 “插入”（’Insert' ）“棒图显示”（'Bar Display'）

符号： 

使用该命令可以插入一个棒图显示元素到你当前的可视化中。它用于显示分配变量的值，并由水平刻度的棒图长度表示。

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。在显示该元素之前，将打开“配置棒图显示”对话框（”Configure Bar Display"）。在这里，你可以定义各种有关元素显示的参数，并在确认对话框实际插入该元素之前对配置进行预览检验。

12.3.16 “插入”（’Insert' ）“历史趋势”（'Histogram'）

符号： 

使用该命令可以插入一个历史趋势元素到你当前的可视化中。它用于数组元素的可视化，并通过并行排列的棒图长度指示其值。

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。在显示该元素之前，将打开“配置历史趋势”对话框（”Configure Histogram"）。在这里，你可以定义各种有关元素显示的参数，并在确认对话框实际插入该元素之前对配置进行预览检验。

12.3.17 “插入”（’Insert' ）“报警表”（'Alarm table'）

符号： 

使用该命令可以插入一个报警表元素到你当前的可视化中。

按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。在显示该元素之前，将打开“配置报警表”对话框（”Configure Alarm table"）。在这里，你将发现除标准工具条（Tooltip）、安全特性（Security）分类之外，还有‘报警表’（'Alarmtable'）、‘排序设置’（'Settings for sorting'）、‘列栏目’（'Columns' ）和‘报警表设置’（’Settings for alarmtable'）等分类，你可以在此定义表的外观和内容。

报警表可用于显示报警，即项目中已定义的报警配置。

12.3.18 “插入”（’Insert' ）“趋势”（'Trend'）

符号： 

使用该命令可以插入一个趋势元素到你当前的可视化中。按压鼠标左键的同时，将形成的区域改变为所需要的尺寸。配置（轴、变量、历史）(axes, variables, history) 在‘趋势’配置对话框中完成。

趋势元素，也可称之为示波器元素，用于显示一段时间内的变量值。它在客户端的文件中保存数据，并以图形显示。只要有值发生变化，将在文件中生成一个新的条目，显示日期/时间和新的值。
趋势图元素是透明绘制的。因此，你可以指定任何所希望的背景（位图，颜色）。

12.4 可视化元素布局

12.4.1 选择可视化元素

缺省状态时选择模式是有效的。为了选择一个元素，可以使用鼠标点击该元素。你也可以通过按压制表符<Tab>键选择元素列表中的第一个元素，每按压一次都向下一个元素跳转。如果你在按压换挡键<Shift>的同时按压制表符<Tab>键，则按照元素列表中的次序反向跳转。

为了选择叠放在一起的一个元素，首先，使用鼠标点击选择最上层的元素。然后，按压<Ctrl>+<Shift>键的同时再次点击鼠标，直到到达所在层的元素。

为了标记多个元素，按压并保持<Shift>键，同时一个接着一个点击相应元素；或者，按压鼠标左键的同时，拖拉一个窗口覆盖所需要选择的元素即可。

为了选择所有的元素，使用‘附加’（'Extras' ）‘选择全部’（'Select All' ）命令。

如果你位于元素列表（通过‘附加’（**'Extras'**）‘元素列表’（**'Element list'**）），可以通过列表选择相关的可视化元素。

12.4.2 切换选择和插入模式

插入一个可视化元素之后，系统自动返回到选择模式。如果你想使用相同的方法插入其它元素，可以再次从菜单中选择相应的命令或工具条中的符号。 

你也可以通过按压<Ctrl>键的同时点击鼠标右键，快速在选择模式和插入模式之间进行切换。

在插入模式，鼠标指针将显示相应的符号，其名称也将在状态条中用黑色表示。

12.4.2.1 ‘附加’（'Extras'）‘选择’（'Select'）

该命令是选择模式的开/关切换命令。它也可以使用符号  完成，或者，按压<Ctrl>键的同时，点击鼠标右键即可。

12.4.2.2 ‘附加’（'Extras' ）‘选择全部’（'Select All'）

该命令允许你在当前的可视化对象中选择所有的可视化元素。

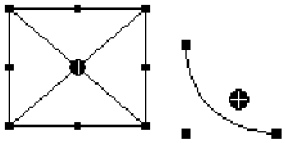
12.4.3 拷贝可视化元素

一个或多个选择元素可以使用‘编辑’（’Edit’）‘拷贝’（’Copy’）、<Ctrl>+<C> 组合键，或相应的拷贝符号，以及‘编辑’（'Edit'）‘粘贴’（'Paste'）命令插入。

另一种可能性是选择该组元素，并在按压<Ctrl>键的同时再次点击这些元素中的一个。如果你现在还保持着按压鼠标左键，则可以从原始的拷贝中分离该元素。

12.4.4 修改可视化元素

你可以通过鼠标点击元素或按压<Tab>键选择一个已插入的元素。该元素的各个角将显示黑色小方形，（椭圆时则是外围矩形角）。多边形、线段或曲线，以及其它方形元素的中心点和角边沿的情况例外。



对于选中的元素，将同时显示**调整点（turning point）**（平衡点（balance point））。你可以设置运动/角度使该元素围绕该点旋转。该调整点显示为一个带白色十字的小黑色圆（ ）。你可以通过按压鼠标左键拖拽该调整点。

按压鼠标左键的同时，点击该元素的任何一个黑色点即可改变其大小，控制调整为新的外形。

当选择多边形（*polygon*）时，你可以使用相同的方法拖拽各个角。当操作时，如果你按压<Ctrl>键，将在该角处插入另外一个角，插入的角也可以被鼠标移动。通过按压<Shift>+<Ctrl>键，则可以删除一个角。

12.4.5 拖拽可视化元素

可以通过按压鼠标左键或箭头键拖拽一个或多个元素。

12.4.6 组合元素

通过选择所有需要的元素并执行命令‘附加’（’Extras’）‘组合’（’Group’），元素可以被组合。组合元素与单个元素的特性相同：

组合元素拥有一个集合框：当拖拽该框时，所有的元素都将被拉伸或压缩；只有组合元素才能被移动到其它位置。

组合元素拥有集合属性：输入只影响组合元素而不是单个元素。因此，该元素组也拥有一个集合配置对话框(‘组合’（'group'）分类)。‘改变颜色’（'Change color'）属性则不能配置一个组合元素！

为了重新定义组合元素中的一个元素，该组合元素必须通过命令‘附加’（’Extras’）‘解除组合’（’Ungroup’）拆开。此时，该组合元素的配置将丢失。

注： 使用 TwinCAT 2.9 版保存项目，组合的可视化元素将被自动保留；即组合元素将在可视化中作为单个元素显示。

12.4.6.1 ‘附加’（'Extras'）‘前向放置’（'Send to Front'）

使用该命令，将把选择的可视化元素前向放置。

12.4.6.2 ‘附加’（'Extras'）‘后向放置’（'Send to Back'）

使用该命令，将把选择的可视化元素后向放置。

12.4.6 3 ‘附加’（'Extras'）‘对齐’（'Align'）

使用该命令，将对齐所选择的可视化元素。

有如下的对齐选项方法：

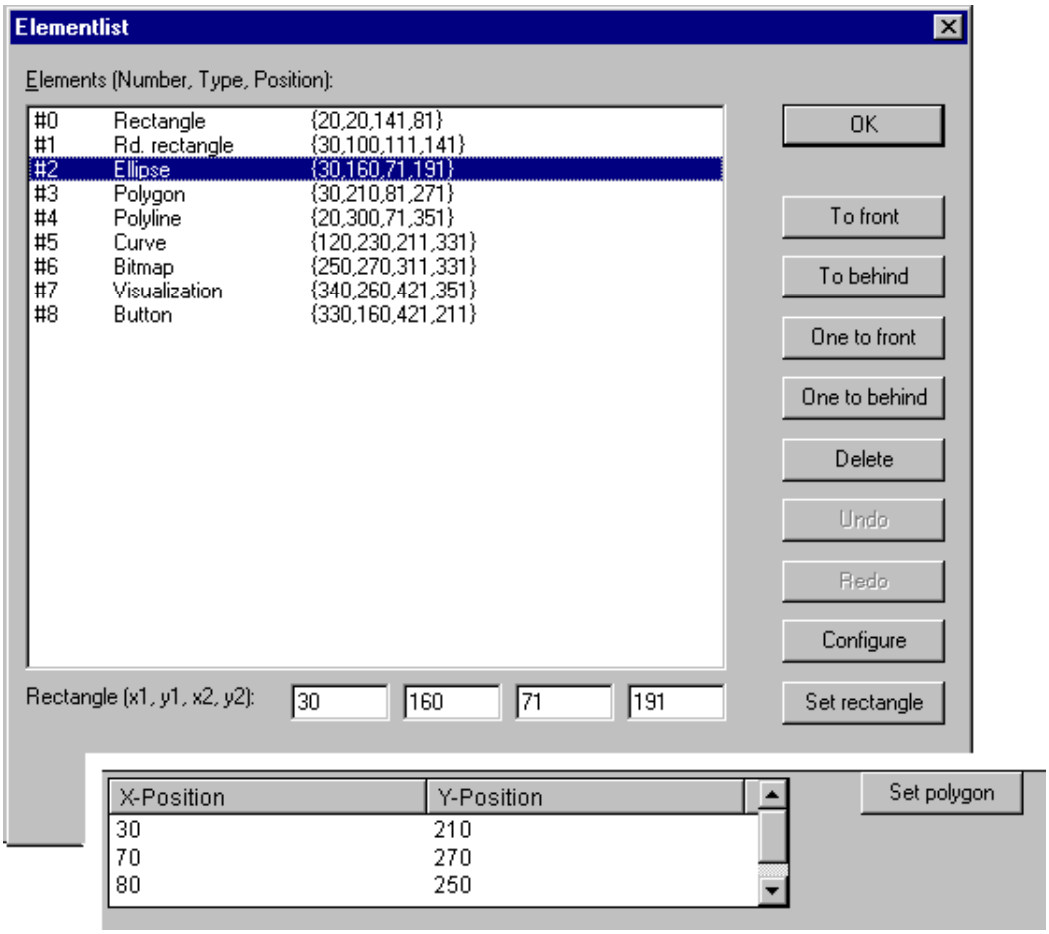
- **左（Left）：**各个元素的左侧将与最左侧的元素对齐。
- **右（Right）/顶（Top）/ 底（Bottom）**的对齐规则与上面相同。

- **水平对中（Horizontal Center）**：各个元素将与所有元素的平均水平中心对齐。
- **垂直对中（Vertical Center）**：各个元素将与所有元素的平均垂直中心对齐。

12.4.6.4 “附加”（'Extras'）“元素列表”（'Elementlist'）

该命令打开一个包含所有可视化元素列表的对话框，其中包括元素的编号（**number**）、类型（**type**）和位置（**position**）。位置由元素所在的左上角（x1,y1）和右下角（x2,y2）的 x 和 y 表示。

元素列表对话框



当选择了一个或多个项目时，可视化中的相应元素均被标识用于可视化控制，如有必要，可以滚动可视化部分，从而显示可视化所包含的元素。

使用前向布局（**To front**）按钮，可以将所选择的可视化元素向前布局。使用后向布局（**To behind**）按钮可以将它们向后布局。

取决于当前所选择的元素，你可以在元素列表下部找到跟随的组合编辑区域，并在此处修改元素的大小和位置：

- 如果当前选择的是矩形、圆角矩形、椭圆、位图、可视化、按钮或 WMF 文件，则在文本字段 **"Rectangle (x1, y1, x2, y2)"** 处有四个编辑区，显示实际的 x/y 位置并可以对其进行修改。

- 如果当前选择的是线段、多边形或曲线，则使用表格显示用于标记该元素外形的小黑点实际 X-位置（**X-Position**）和 Y-位置（**Y-Position**）。一旦它们被选中，这些值即可在此进行修改。

为了设置可视化元素列表中修改的位置值，可分别按压按钮设置矩形（**Set rectangle**）（情形 1）和设置多边形（**Set polygon**）（情形 2）。

使用删除（**Delete**）按钮可删除选中的可视化元素。

使用取消（Undo）和恢复（Redo）按钮则完成取消或恢复已作出的修改，其作用与使用命令‘编辑’（'Edit'）‘取消’（'Undo'）和‘编辑’（'Edit'）‘恢复’（'Redo'）完全相同。在该对话框中，你可以观察正在发生的变化。

点击 **OK** 关闭该对话框并对作出的改变进行确认。

使用配置（**Configure**）则可以操作元素的配置对话框。

12.4.7 可视化中的状态条

如果可视化拥有焦点，则使用像素表示的、相对于映像左上角的当前鼠标光标 **X** 和 **Y** 位置（**position**）显示于状态条中。如果鼠标点定位于一个元素（**Element**）之上，或者该元素正被处理，则显示该元素的编号。如果你已经选择了一个准备插入的元素，则该元素也将被显示（示例，矩形（**Rectangle**））。

12.5 配置可视化

12.5.1 概述

如果你正在操作一个可视化，你可以通过‘附加’（’Extras’）菜单进入配置对话框，对一个可视化元素或一个作为整体的可视化对象进行配置。另外，某些设置还可以通过对象管理器中可视化对象的属性对话框完成。

请注意项目选项，可以对可视化文件（**visualization files**）定义单独的目录。

12.5.2 配置可视化元素

通过 'Extras'(附加) 'Configure' (配置)命令，可以在打开的配置对话框中设置一个元素或对象的选项属性或动态属性（通过插入项目变量实现）。除此之外，还可以通过结构变量成员对属性进行编程，该结构变量成员可以对每个可视化元素进行定义。

联机模式时，分析顺序按照下列方式进行：

- 动态赋值，即：通过项目变量，将覆盖同一属性的固定参数。

- 如果一个元素的属性在使用一个“普通”的项目变量进行定义的同时，又使用了结构变量进行了定义，则联机模式时，系统将优先考虑项目变量的值。

请注意尽量使用占位符、特殊输入的可能性，该特性对于 TwinCAT HMI、目标可视化或万维网可视化而言都是非常有用的；例如，可视化服务仅用于 PLC 程序的用户接口 (参见：配置对话框 INTERN 分类 'Input' （输入），键盘用法)。

12.5.3 可视化占位符

配置对话框中任何一个输入变量或文本的位置，均可以使用一个占位符进行替代。该特性使得可视化对象如果在程序中不是直接使用时变得更加智能化，占位符创建之后，可以在其它的可视化对象中作为“实例”插入。当配置这样一个实例时，占位符可以替代变量名或文本(参见可视化插入配置，并可以在此找到使用占位符的示例)。

包含在两个美元符号 ($) 中的任何字符串都是有效的占位符 (例如：$variable1$，variable$x$)。 对于每一个占位符而言，“编组值”可以在“占位符列表”对话框中作为特定的输入进行定义(通过'Extras'（附加）'List of placeholders'（占位符列表）调用)。当配置可视化对象的实例时，可以使用这些值中的任何一个替代该占位符。占位符列表只有在替代实例时才能使用。

使用占位符概念的应用示例：

借助于相同的可视化使显示功能块实例变得非常轻松。例如，配置可视化 visu 时，想观察功能块的变量，可以在每个变量条目的开始处放置一个占位符 FUB (例如：FUB.a)。如果将来使用这个 visu 的实例时 (通过插入 visu 到另一个可视化或通过调用 'Zoom to vis.'(缩放到可视化))，则在配置占位符的实例时，FUB 可以使用功能块实例名替代，从而实现对其进行观察。

下面显示的是具体的示例：

在项目中定义一个如下声明的功能块：

```
FUNCTION_BLOCK fu
VAR_INPUT
 changecol : BOOL; (* 改变可视化颜色 *`)
END_VAR
```

在 MAIN 中定义两个 ‘fu’ 的实例：

```
inst1_fu : fu;
inst2_fu : fu;
```

创建可视化对象 'visu'。插入一个元素并打开配置对话框，在 'Variables'（变量）分类的 'Change color' （改变颜色）字段中输入："FUB.changecol"。打开 'Input' （输入）分类，并在 'Tap Variable' （变量按钮）字段中输入 "FUB.**changecol**"。打开 'Text' （文本）分类并输入"FUB - **change color** "。

创建另外一个可视化对象 'visu1'。

在 ‘visu1’ 中两次插入可视化 'visu' (两个 ‘visu’ 参考)。

标记第一个 ‘visu’ 参考，并打开配置对话框的 'Visualization'（可视化）分类。按压 'Placeholder'（占位符）按钮，以便显示占位符列表。并在此使用 ‘MAIN.inst_1’ 替代 'FUB' 条目。

现在，标记第二个 ‘visu’ 参考，并采取前面相同的方法，使用 ‘MAIN.inst_2’ 替代 'FUB' 字段。

至此，联机模式时，已完成配置的两个 ‘fu’ 实例变量值可以在相应的 ‘visu’ 实例中进行观察。

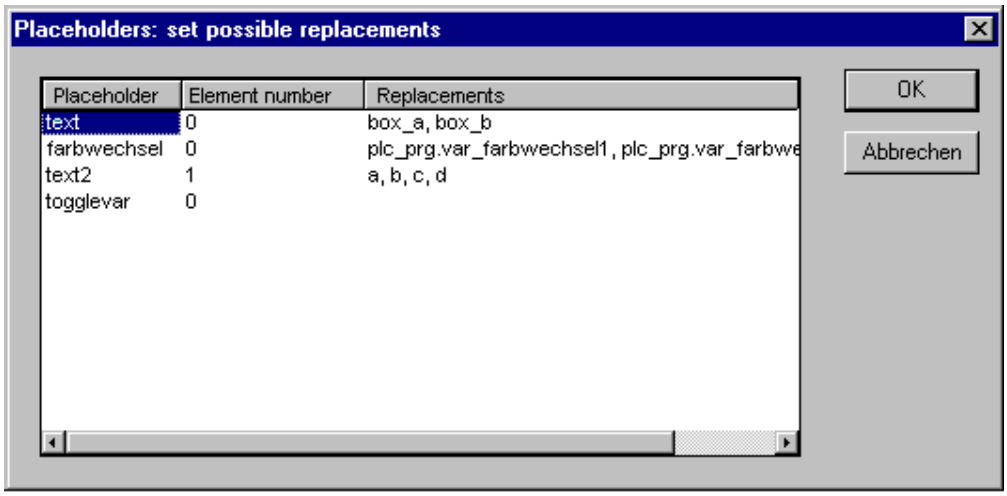
当然，占位符 FUB 可以在 ‘visu’ 配置框中任何输入变量或文本的地方使用。

12.5.3.1 'Extras'（附加） 'List of Placeholders...'（占位符列表）

该列表在 TwinCAT 中的两个地方使用：占位符管理和占位符配置：

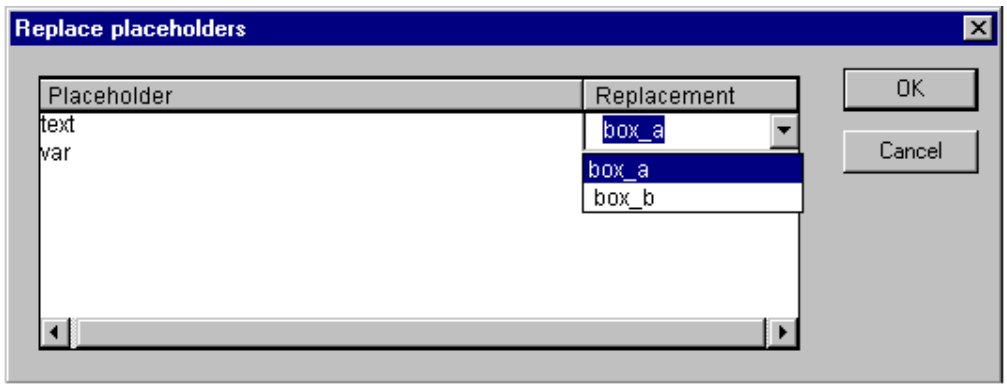
首先，在配置可视化对象时使用该列表，并在以后作为实例插入到其它可视化中。为此，在配置对话框中，你可以使用占位符替代附加的变量和字符串。可以通过 ‘Extras’ （附加）菜单或上下文菜单中的 'List of Placeholders' （占位符列表）命令打开 'Placeholders' （占位符）对话框。该列表中有三列项目：

可以替代占位符输入的占位符列表：



- Placeholder（占位符）列将列举所有的占位符，即当前可视化对象配置中所使用的占位符。Element number（元素编号）列则显示包含占位符的元素。在 Replacements（替代）列，你可以输入一个或几个字符串 (文本、变量、表达式)，即你想在以后配置可视化对象实例期间，替代一个占位符时所需要使用的元素。所选择的元素必须用逗号分隔。如果没有指定或指定了一个不可替代的字符串，则在以后的可视化参考配置期间可以使用任何期望的文本替代这个占位符。
- 此后，在配置上面提到的可视化对象的实例时，你可以使用这个占位符列表，即：当这个对象已经插入到另外一个可视化之后(作为一个'reference'（参考）)。通过命令 'Insert'（插入） 'Visualization'（可视化）完成。为此，按照下列步骤打开对话框：选择插入的可视化，执行命令 'Extras'（附加）'Configure'（配置），并按压'Visualization'（可视化）分类中的'Placeholders' （占位符）按钮。此时，对话框仅包含两列：

可视化实例中替代占位符的占位符列表



Placeholder（占位符）列 – 和上面描述的一样 – 将显示原始的可视化对象所定义的所有占位符。如果还定义了其他的替代选择项，将在 'Replacement' （替代）列中列举出可以使用的项目。选择这些条目中的一个，并替代当前实例中的占位符。如果没有预定义任何替代项，则可以手动输入表达式或变量。为此，在 'Replacement' （替代）列字段点击鼠标可以打开一个编辑字段。

12.5.3.2 'Extras'（附加）'Configure'（配置）

使用该命令，将打开'Configure element' （配置元素）对话框，以便对所选择的可视化元素进行配置(参见选择可视化元素)。当双击该元素时，也可以打开该对话框。

在对话框的左侧选择一个类别(可以使用的类别取决于元素类型)，并在右侧填写所需要的信息。可以通过激活对应的选项实现。通过插入有效的变量名，其值应定义元素的属性。

注意: 还可以使用对一组元素进行配置的配置对话框。其用法取决于"element"（元素）组的有效设置。如果你想对该组中的某个元素进行配置，你应拆解该组。

注意: 如果你已经通过“静态”设置定义了一个元素的属性，同时又通过变量对同一属性进行了动态链接，则联机模式时，变量将覆盖静态值(示例："Alarm color Inside"（报警内部颜色）可以在'Color'（颜色）分类静态定义，也可以通过变量在'Colorvariables' （颜色变量）分类中动态定义)。如果该设置在通过“普通”项目变量进行控制的同时，还通过结构变量进行控制，则结构变量值也将覆盖“普通”项目变量值。

请注意: Meter（仪表），Bar Display（棒图显示）和 Histogram（历史趋势）必须在此之前进行重组！

在元素配置区域，即变量作用区，允许的输入条目如下：

- 变量名，可以使用输入助手。
- 由组件通道、带常数索引通道的字段、变量和直接地址组成的表达式。
- 操作数和常数，可以是上述表达式的组合。
- 替代变量名或文本字符串的占位符。

允许的表达式示例：

x + y

100 * PLC_PRG.a

TRUE

NOT PLC_PRG.b

9 * sin(x + 100) + cos(y+100)

不允许功能调用。非法表达式将在登录时导致出现错误信息 („Invalid Watch expression..." （非法监视表达式）)。非法表达式示例：fun(88)，a := 9，RETURN。

在配置对话框中有两种方式写入全局变量：".globvar" 和 “globvar" 是相同的。但是，带点的形式 (它也在观察和接收管理器中使用) 不允许在表达式集合中使用。

请注意使用占位符的可能性。

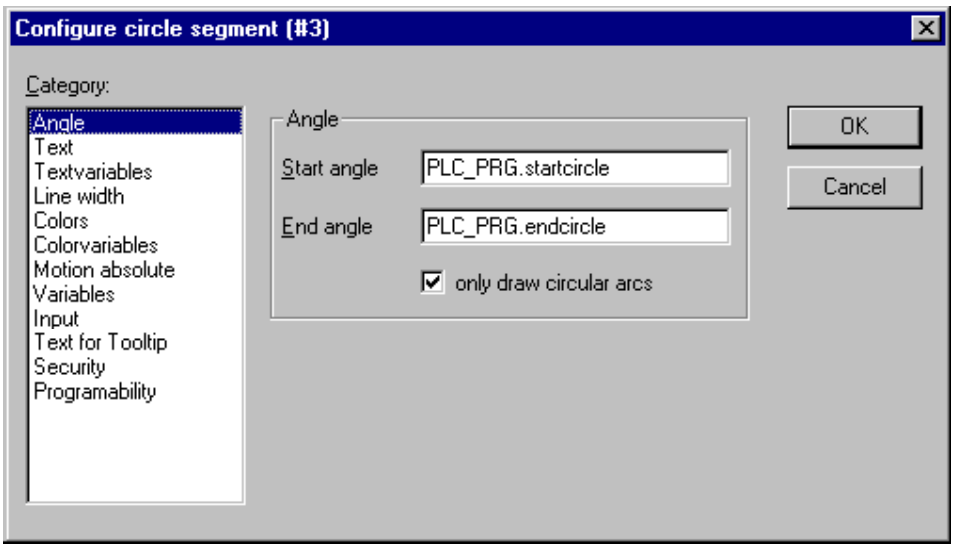
12.5.4 Angle（角度）

在配置对话框的 **Angle** （角度）分类 'Configure Pie'（配置饼图）中，你可以使用度分别输入值或变量定义元素扇区的起始角和终止角。该扇区将以顺时针的方向从起始角位置到终止角位置绘制。

示例：

输入起始角："90"，终止角："180"

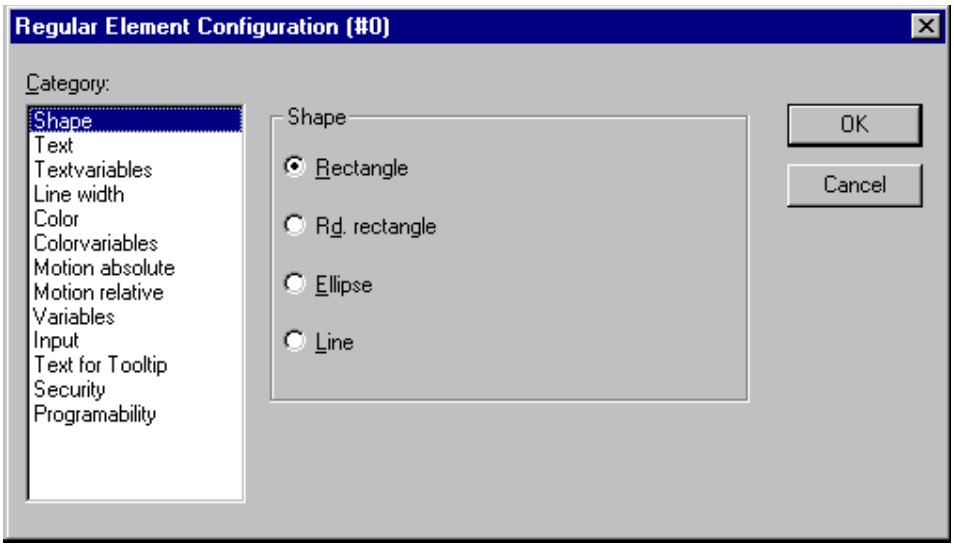
配置饼图对话框



12.5.5 Shape（形状）

在可视化元素配置对话框中，你可以在 Shape（形状）分类中选择 Rectangle（矩形）、Rounded Rectangle（圆角矩形）、Line（直线）和 Ellipse（椭圆）；以及分别表示 Polygon（多边形）中的 Line（直线）和 Curve（曲线）。其形状将改变已设置的尺寸大小。

可视化元素的配置对话框 (Shape （形状）分类)



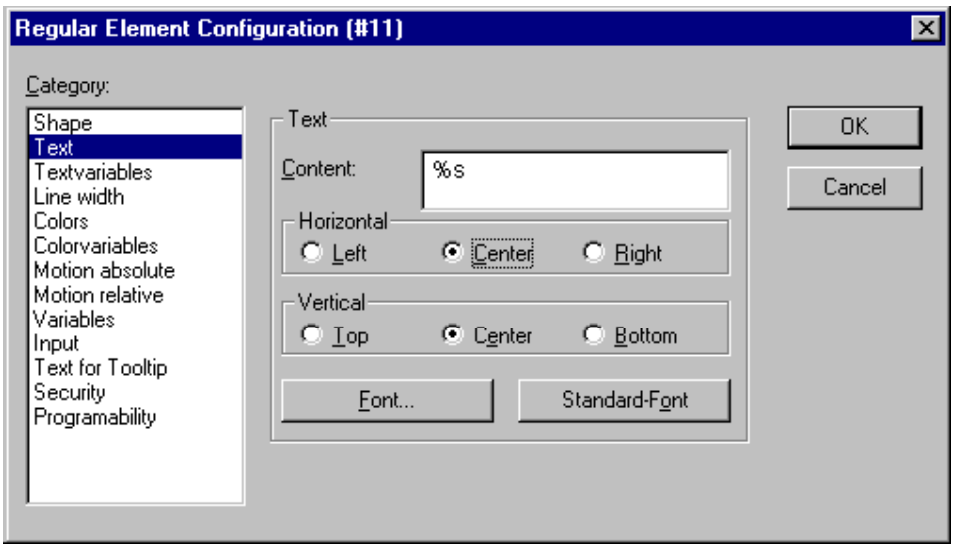
12.5.6 Text（文本）

在可视化元素配置对话框中，你可以在元素的 Text（文本）分类中指定文本。可以直接输入或/和通过变量定义，以便确定文本字符串。也可以使用占位符。同时，字体和对齐方式也在此设置为缺省值。

注意: 一旦文本参数由附加的动态文本提供，即通过系统或结构变量 (参见下面，'Text variables'（文本变量）和'Programmability'（编程）)，则当前打开的对话框中定义的静态设置将被覆盖！

在一个元素属性有多重定义的情况下，应考虑联机模式时一个值将被另一个值所覆盖的特定处理顺序。

可视化元素的配置对话框 (Text（文本）分类)



在内容字段中输入文本。使用组合键 <Ctrl>+<Enter>，你可以插入回车行，使用 <Ctrl>+<Tab>， 可禁止制表符。除了可以输入纯文本串之外，你还可以使用下面的格式顺序：

- 如果你在文本中包含 "%s"，则联机模式时，该位置将被 Variables （变量）分类中的 Textdisplay（文本显示）字段中的变量值所替代。除 "s" 之外，你还可以使用与标准 C-库函数 'sprintf' 中相同的其它格式化字符串。参见下面的字符串和句法示例：

字符	句法 / 输出形式
d,i	十进制数
o	无符号八进制数 (无引导符零)
x	无符号十六进制数 (无引导符 0x)
u	无符号十进制数
c	单个字符
s	字符串
f	REAL-值；语法： %|<对齐格式><最小宽度><精度>|f 精度定义了逗号之后小数点的个数 (缺省值：6)。示例参见下面。

注意: 如果你想显示一个百分号 % 并与上面提到的格式化字符串进行组合；你必须输入"%%"。例如：输入"Rate in %%: %s" ，则联机模式时将显示 "Rate in %: 12" (假如文本显示变量的当前值为 "12")。

变量值将在联机模式时进行相应的显示。你可以输入任何相应的 IEC-格式化字符串，并与使用的变量类型相对应。

注意: 系统不检验格式化字符串类型是否与变量类型匹配，该变量在 'Text Output' （文本输出）字段中进行定义！

示例：

'Content'（内容）字段输入：Fill level %2.5f mm

'Textdisplay'（文本显示）字段输入：(变量类型为 REAL)，例如：plc_prg.fvar1

-> 联机模式输出，例如：Fill level 32.8999 mm

- 如果你输入 "%t"，后跟某些特定的占位符，则该位置将在联机模式时由系统时间占据。占位符定义了显示格式，参见下表。

注意: 在 'Content' （内容）字段中，不要在 %t 之前插入任何其它字符 (反之则允许，例如："%s"，参见上面)

%a	星期缩写
%A	星期全称
%b	月份缩写
%B	月份全称
%c	与本地相对应的日期和时间

%d	十进制表示的月份中的天数 (01 – 31)
%H	24-小时格式的小时数 (00 – 23)
%I	12-小时格式的小时数 (01 – 12)
%j	十进制表示的年份中的天数 (001 – 366)
%m	十进制表示的月份数 (01 – 12)
%M	十进制表示的分钟数 (00 – 59)
%p	当地时间 A.M./P.M. 指示的 12-小时时间
%S	十进制表示的秒数 (00 – 59)
%U	十进制表示的年份中的星期数，星期天为一个星期中的第一天 (00 – 53)
%w	十进制表示的星期数 (0 – 6；星期天为 0)
%W	十进制表示的年份中的星期数，星期一为一个星期中的第一天 (00 – 53)
%x	当地表示的日期
%X	当地表示的时间
%y	无世纪表示的年份，十进制数 (00 – 99)
%Y	有世纪表示的年份，十进制数
%z, %Z	时区名或缩写；如果时区未知则为空
%%	百分号

示例：

%t%a %b %d.%m.%y %H:%M:%S

联机模式时显示：Wed Aug 28.08.02 16:32:45

占位符之间，你可以插入任何文本字符串：

%tToday is %d.%m.%y

->联机模式时显示：Today is 28.08.02

注意: 如果一个文本字符串想转换到一个转换文件中，该文件将在联机模式时使用，并使其切换到其它国家语言，则必须在开始和结束处添加限定符 #。
示例："#Pump 1#" 或者是 "#Pump# 1"
示例中的第二种情况用于文本 Pump 被多次引用时使用(Pump 1，Pump 2，等)，以防止转换中多次出现该文本串。

- 如果你在文本中包含"%<PREFIX>"，则可以使用某个字符串替代 "PREFIX"，即作为动态文本使用的标识符。该前缀将与一个 ID 号同时使用，即配置对话框的'Variables'（变量）分类字段'Textdisplay'（显示文本）。该组合对应参考某个文本，包含在一个 xml-文件中，并列举了所有可能的动态文本。因此，系统运行时，标记为当前 ID-前缀组合的文本将被显示。有关更加详细的信息参见语言分类中的'Settings'（设置）描述。

配置后的文本将在联机模式下，以预定义的对齐方式出现在元素内：水平方向 居左，居中或居右，垂直方向居上，居中或居下。

如果你使用 Font(字体)按钮，将出现字体选择对话框。选择所需要的字体并用对话框的 OK 按钮确认。如果你使用 Standard-Font（标准字体）按钮，系统将以项目选项('Project'（项目）'Options'（选项） 'Editor'（编辑器）)中所选择的字体进行设置。如果字体在那里发生了变化，则该字体将作用于所有的元素，但使用 Font（字体）按钮选择后的元素除外。

12.5.7 Textvariables（文本变量）

在可视化元素配置对话框的 Textvariables（文本变量）分类中，你可以指定一个变量，该变量将动态设置'Text'（文本）分类中定义的字符串颜色和字体。最好使用输入助手(<F2>)输入变量名。

你也可以使用 VisualObjectType 类型的结构成员设置文本属性。为此，请参见'Programability'（编程）分类的描述；在那里你可以找到特殊结构成员可能有的值以及它们的效果。

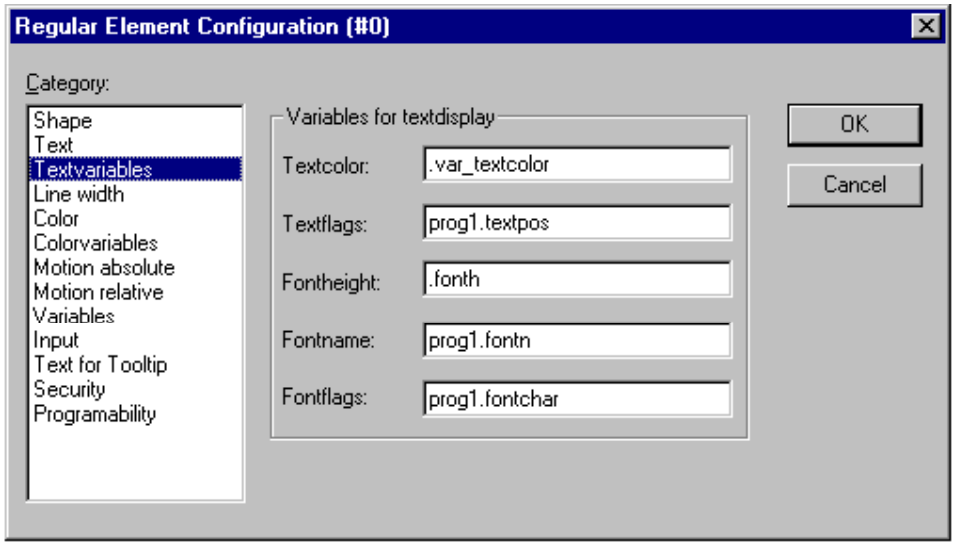
注意: 如果对'Text'（文本）分类中相应的静态属性进行了定义，则该属性值将被动态参数值覆盖。

在多重定义元素属性的情况下，应考虑值被另一个值所覆盖时的特定处理顺序。

对话框参数：

参数:	意义:	项目变量输入示例:	程序变量用法示例:	VisualObjectType 结构对应的成员:
Textcolor:	文本颜色	"plc_prg.var_textcolor"	var_textcolor= 16#FF00FF 洋红	dwTextColor
Textflags:	对齐方式(居右，居左，居中...)	"plc_prg.textpos"	textpos:=2 文本居右	dwTextFlags
Fontheight:	像素表示的字体	".fonth"	fonth:=16; 字体高度 16 像素	ntFontHeight
Fontname:	字体名称	"vis1.fontn"	fontn:=arial; Arial 字体	stFontName
Fontflags:	字体显示(粗体，下划线，斜体...)	"plc_prg.fontchar"	fontchar:=2 粗体显示文本	dwFontFlags

可视化元素配置对话框 (Textvariables（文本变量）分类)

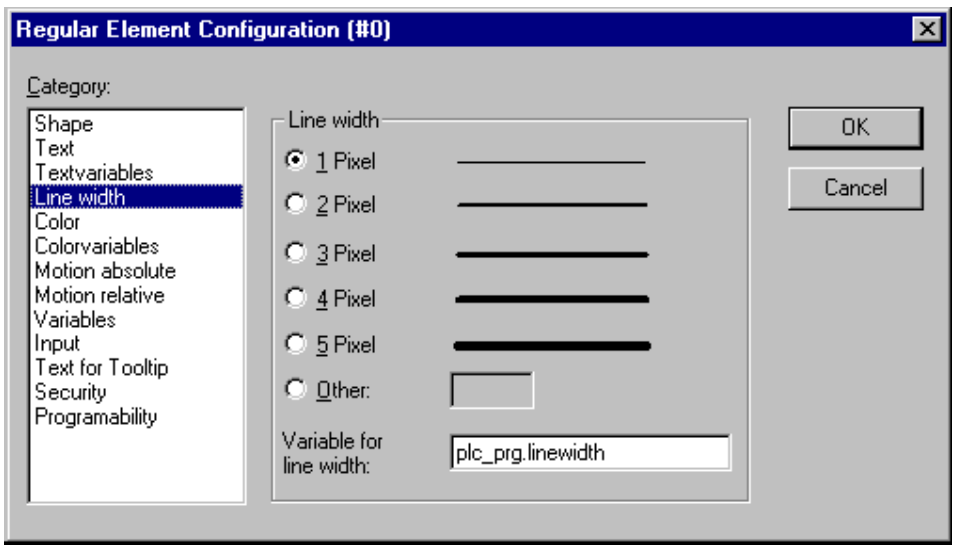


12.5.8 Line width（线型宽度）

在可视化元素配置对话框中，你可以对元素选择线型宽度。作为预定义的选项，你可以发现像素宽度为 1 到 5 的设置，另外，其它值可以手动输入(Other:（其它）)，或插入一个项目变量(Variable for line width:（线型宽度变量）)。对于后一种情况，可以使用输入助手(<F2>)。

注意: 只要该参数被动态定义，即：通过结构变量定义(参见下面'Programmability'（编程）分类)，静态设置将在联机模式下被覆盖。

可视化元素配置对话框 (Line width（线型宽度）分类)



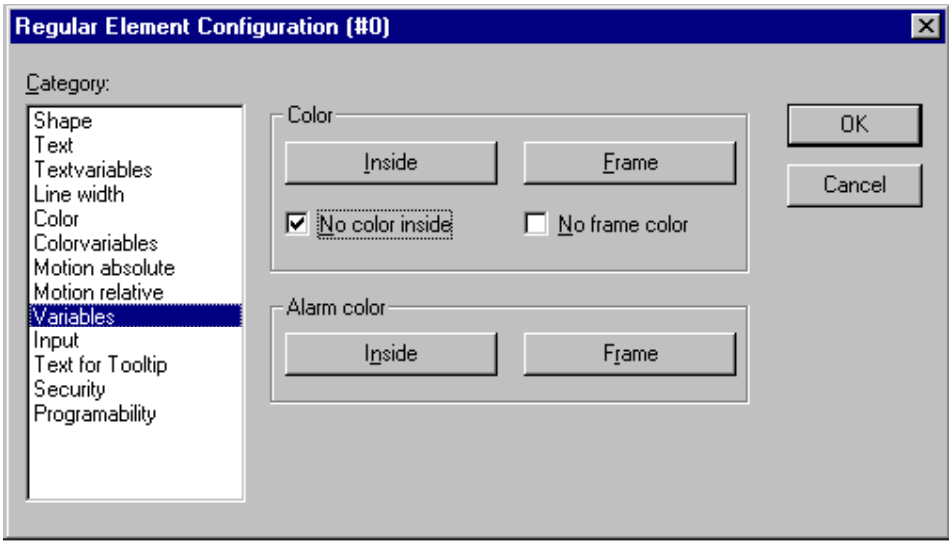
12.5.9 Colors（颜色）

在可视元素配置对话框的'Color' （颜色）分类中，你可以为元素的内部区域和框架选择原始颜色和报警颜色。如果选择内部无颜色和无框架颜色，则可以创建透明的元素。

注意: 只要参数被一个变量作了附加的动态定义，联机模式时静态设置将被覆盖。

在多重定义元素属性的情况下，应考虑值被另一个值所覆盖时的特定处理顺序。

可视化元素配置对话框 (Color （颜色）分类)



现在，如果你在 Variables（变量）分类的 Change Color（改变颜色）字段中输入一个布尔变量，只要变量值为 FALSE，则元素将以设置的 Color（颜色）显示。如果变量值为 TRUE，则元素以 Alarm Color（报警颜色）显示。

注意: 如果 PLC 处于联机模式，改变颜色功能才变为有效。

如果你想改变框架的颜色，则应按压 Frame （框架）按钮，而不是 Inside （内部）按钮。不管是哪种情况，对话框将打开所选择的颜色。

在这里可以从原始颜色和用户定义的颜色中选择所希望的颜色。通过按压 Define Colors（定义颜色），你可以改变用户定义的颜色。

12.5.10 Color Variables（颜色变量）

在这里，你可以输入项目变量(例如：PLC_PRG.color_inside)，以确定联机模式时的指定属性： 这些属性定义借助于结构 VisualObjectType 成员也可以进行编程。为此，请参见可视化元素的 "Programmability"（编程）描述。并在那里找到所有可能值的列表和它们的效果。

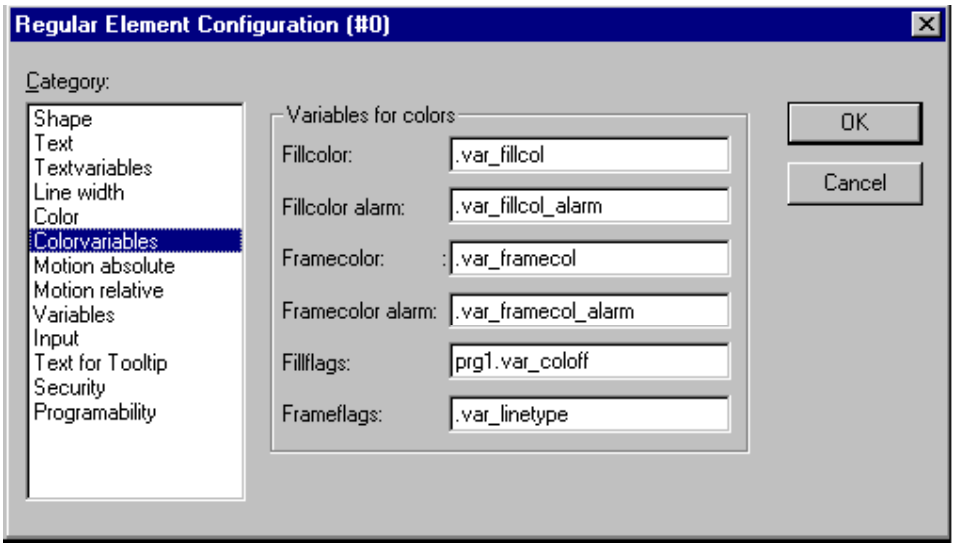
注意: 输入到 Color Variables （颜色变量）对话框中的变量，联机模式时，将由'Color'（颜色）分类以及结构变量赋予的值覆盖静态值。

在多重定义元素属性的情况下，应考虑值被另一个值所覆盖时的特定处理顺序。

对话框参数：

参数：	**描述：**	**输入示例：**	**程序使用变量示例：**	**结构变量 VisualObjectType 对应成员：**
Fillcolor:	填充颜色	"plc_prg.var_fillcol"	var_var_fillcol:=16#FF00FF→ 粉红色填充	dwFillColor
Fillcolor alarm:	如果'Change color' （改变颜色）变量为TRUE，则填充颜色有效	"plc_prg.var_fillcol_a"	var_fillcol_a:=16#FF00FF→ 报警填充粉红颜色	dwFillColorAlarm
Framecolor:	框架颜色	"plc_prg.var_framecol"	var_framecol:=16#FF00FF→ 框架粉红颜色	dwFrameColor
Framecolor alarm:	如果'Change color' （改变颜色）变量为TRUE，则框架颜色有效	"plc_prg.var_framecol"	var_framecol:=16#FF00FF →报警框架粉红颜色	dwFrameColorAlarm
Fillflags:	当前内部配置的颜色有效(FALSE) 否则失效(TRUE)	"plc_prg.var_col_off"	var_col_off:=1 → 填充颜色定义的颜色无效，框架颜色保持有效	dwFillFlags
Frameflags:	显示框架(实线，点线等)	"plc_prg.var_linetype"	var_linetype:=2; → 框架将显示为点线	dwFrameFlags

可视化元素配置对话框 (Colorvariables（颜色变量）分类)



12.5.11 Motion absolute（绝对运动）

在可视化元素配置对话框的 Motion absolute （绝对运动）分类中， X- 或 Y-Offset（偏移）字段可以进行输入。这些变量可以在 X 或 Y 方向上移动元素，并取决于相应的变量值。Scale（比例）字段变量将改变元素的线性度大小到其当前值。该值用于比例因子，并被隐含地除以 1000， 因此，不需使用 REAL-变量获得缩小的元素。该比例值始终参考平衡点。

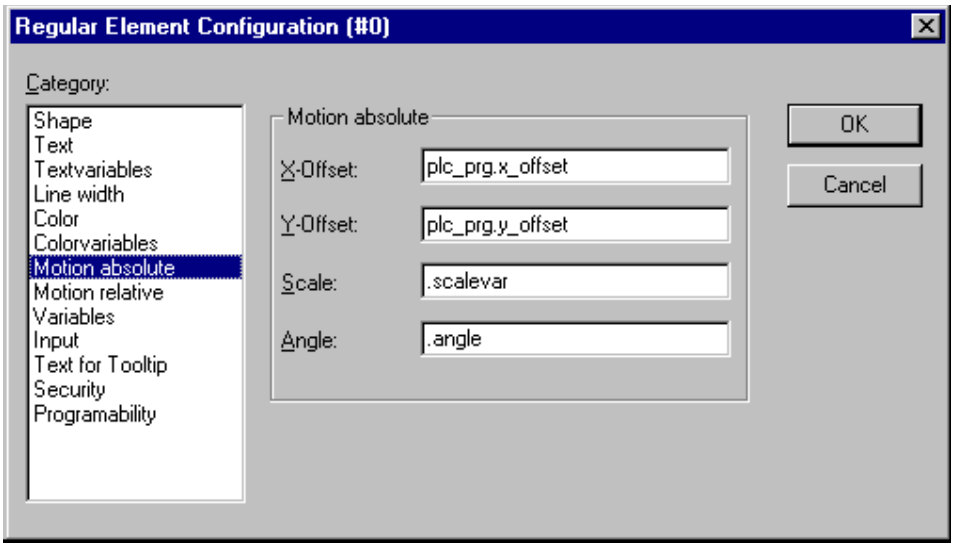
Angle（角度）字段变量将导致元素沿着调整点旋转，并取决于相应的变量值。(正值 = 算数正值 = 顺时钟方向)。该值用度表示。对于多边形而言，每个点都旋转；换句话说，多边形旋转。对于所有其它的元素，元素旋转，同样，上边沿始终保持在上方。

当点击一个元素时，调整点将出现，并由一个小黑圆及白色的十字符表示 ( )。你可以通过按压鼠标左键拖拉该调整点。

注意: 联机模式时，'Motion absolute'（绝对运动）对话框中设置的变量将被另一个同属性定义的结构成员值所覆盖('Programability'（编程）)。

在多重定义元素属性的情况下，应考虑值被另一个值所覆盖时的特定处理顺序。

可视化元素配置对话框 (Motion Absolute（绝对运动）分类)



12.5.12 Motion relative（相对运动）

在可视化元素配置对话框的 Motion Relative （相对运动）分类中，你可以对元素的每个边沿分配变量。并取决于变量值，元素相应的边沿将被移动。输入变量到字段中的最简便方法是使用 Input Assistant（输入助手） (<F2>)。

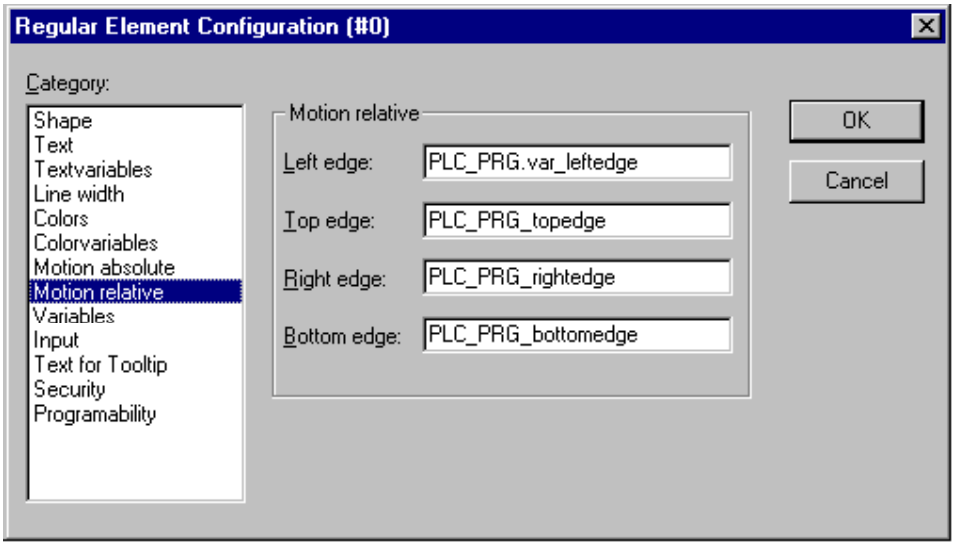
四个字段对应元素的四个边。边角的基点位置始终为零。在相应的栏目中，变量的新值将以像素为单位将边界移动到该值附近。因此，输入的变量应该是 INT 变量。

注意: 正值使水平边向下移动，或者，垂直边向上移动！

注意: 联机模式时，'Motion Relative'（相对运动）对话框中设置的变量将被另一个同属性定义的结构成员值所覆盖('Programability'（编程）)。

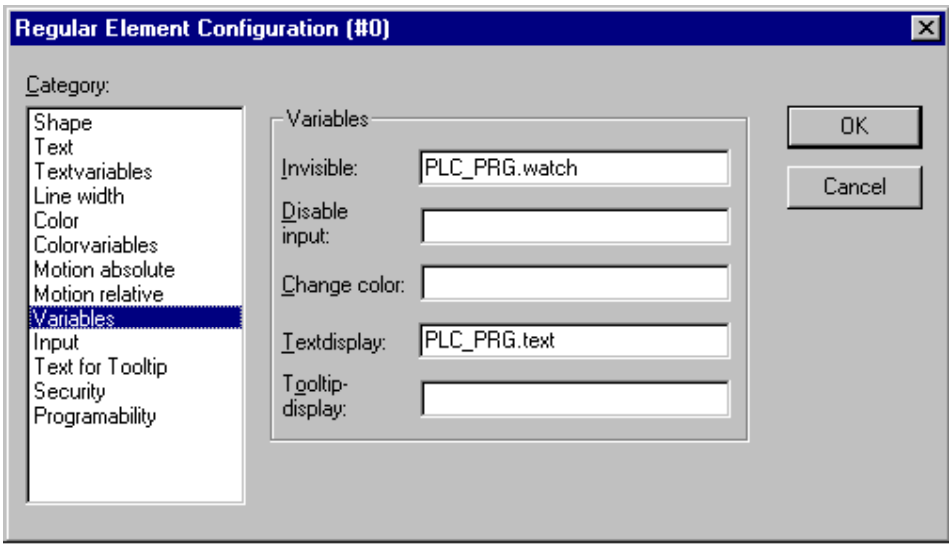
在多重定义元素属性的情况下，应考虑值被另一个值所覆盖时的特定处理顺序。

可视化元素配置对话框 (Motion Relative（相对运动）分类)



12.5.13 Variables（变量）

可视化元素配置对话框 (Variables （变量）分类)



在可视化元素配置对话框的 Variable（变量）分类中，你可以输入变量描述可视化元素的状态。输入变量到字段中的最简便方法是使用 Input Assistant（输入助手）。

注意: 联机模式时，'Variable'（变量）对话框中设置的变量将被另一个同属性定义的结构成员值所覆盖('Programability'（编程）)。

在多重定义元素属性的情况下，应考虑值被另一个值所覆盖时的特定处理顺序。

你可以在 Invisible（不可见）和 Change color（改变颜色）字段中输入布尔变量。该字段中的值决定了它们的行为。如果 Invisible（不可见）字段中的变量值为 FALSE，则可视化元素将可见。如果变量值为 TRUE，则可视化元素将不可见。

Disable input（输入禁止）：如果输入变量值为 TRUE，则'Input'（输入）分类中的所有设置将被忽略。

Change color（改变颜色）：如果该字段中定义的变量值为 FALSE，则可视化元素将以其缺省值显示。如果变量值为 TRUE，该元素将以其 alarm color（报警颜色）显示。

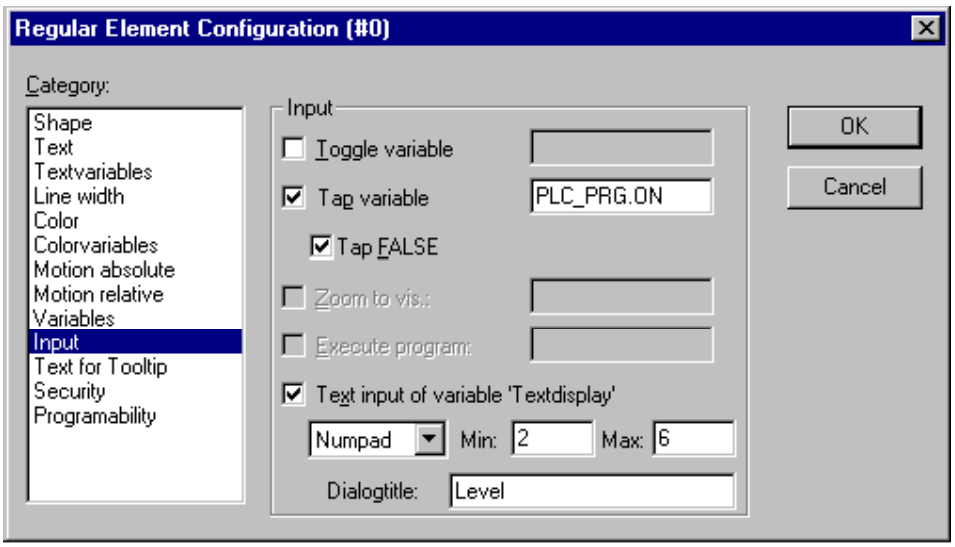
Textdisplay（文本显示）：

- 如果你已经在 **Text** （文本）分类的内容字段中插入了一个 "%s" ，或者在文本字串中插入了一个 "%s"，则在'Textdisplay'（文本显示）中定义的变量值将在联机模式时显示。"%s" 将被该值所取代。
- 如果你已经在 **Text** （文本）分类的内容字段中插入了一个包含"%<PREFIX>" 的符号 ("PREFIX" 必须是一个字符顺序)，则'Textdisplay'（文本显示）中输入的变量值将作为一个 ID，与前缀结合后作为一个文本的参考，并在一个 XML-文件中描述。该文本将在可视化对象的联机模式时替代 "%<PREFIX>" 显示。因此，动态修改文本显示成为可能。参见语言'Settings'（设置）分类中的详细信息描述。
- 如果你想在联机模式时使用键盘编辑该变量的值，可以通过 Input（输入）分类中的 'Text input of variable'（变量输入文本） 'Textdisplay' （文本显示）实现。

Tooltip-display（工具条显示）：在这里输入一个类型为 STRING 的变量，联机模式时，该值将在元素的工具条中显示。

12.5.14 Input（输入）

可视化元素配置对话框 (Input （输入）分类)



Toggle variable（翻转变量）：如果该选项有效，联机模式时，当你每次用鼠标点击可视化元素时，input（输入）字段中的变量值被翻转。你可以使用 input assistance（输入助手）<F2>输入

数据。布尔变量的值将在鼠标每次点击之后从 TRUE 变为 FALSE，当鼠标再次点击时则变回值 TRUE，依此类推。

Tap Variable（变量塞）：如果该选项有效，联机模式时，你可以在 TRUE 和 FALSE 之间切换 input（输入）字段中的布尔变量值。将鼠标光标放置于该元素上，按压鼠标键并保持压下。如果 FALSE 变量塞选项有效，只要鼠标被按下，该值即被设置为 FALSE，否则，其值被设置为 TRUE。一旦你释放鼠标键，该变量值就返回到其初始值。

Zoom to Vis...（缩放到可视化）：如果该选项有效，你可以在编辑字段中输入同一个项目中的可视化对象名，联机模式时，如果鼠标点击该元素，系统将跳转切换到该可视化中。此时，始终是目标可视化的第一个窗口将在当前可视化关闭前被打开。

系统允许输入以下内容：

- 当前项目的可视化对象名 (参见 Object Organizer（对象管理器）)
- 如果可视化参考中包括将要跳转的占位符，调用时，该占位符可以被变量名或文本直接替代。为此，请参照以下句法：
  <Visuname>(<Placeholder1>:=<Text1>, <Placeholder2>:=<Text2>,..., <Placeholder n>:=<Textn>)。
  可视化编译期间将对其进行检查，检查文本是否与占位符列表中所定义的替代值匹配，如果不匹配，将输出一个报警。

  示例：
  调用可视化 visu1，其中，visu1 中所使用的占位符 var_ref1 和 var_ref2 将分别被变量 PLC_PRG.var1 和 PROG.var1 所替代：

  visu1(var_ref1:=PLC_PRG.var1, var_ref2:=PROG.var1)
- 如果输入替代可视化对象的程序变量类型是 STRING (例如： PLC_PRG.xxx)，则该变量可以用于定义可视化对象名(例如：visu1)，当鼠标点击事件发生时，系统将切换到所定义的可视化名中(例如：xxx:= visu1)。
- 如果你在 Zoom to vis.（缩放到可视化）字段中发送„ZOOMTOCALLER"（缩放到调用位置）命令，联机模式时，当你点击该元素时，系统将反向跳转到调用的可视化中，当然，只有在已经配置这种调用位置的前提下才能发生。

注意: 隐含变量 CurrentVisu (类型 STRING，用于(系统)隐含变量)描述了当前打开的可视化对象名。例如，它可以在应用中用于控制哪个可视化已被打开，或当前打开的是哪个可视化。但是，只有当可视化对象名是由大写字符定义时才有效 (参见'Create a visualization object'（创建一个可视化对象）)。示例：CurrentVisu:='PLC_VISU'；

Execute program（执行程序）：如果该选项有效，你可以在该输入字段中输入 ASSIGN- 或特殊的 "INTERN"-命令，联机模式时，只要你用鼠标点击该元素，这些命令将被执行。按压按钮 "..."，可以进入程序对话框，并在此选择所希望的命令(Add（添加）)、根据需要安排它们的顺序(之前，之后)。

注意: 如果可视化是系统操作的唯一接口，该特性尤其重要(纯操作版)。参见命令描述：TwinCAT 操作版的特殊输入可能性。

如果你选择了 Text input of variable 'Textdisplay'（文本输入变量'文本显示'），则联机模式时，你有可能在该可视化元素中，通过按压<Enter>（回车），并在显示的 Variables（变量）分类中的 Textdisplay（文本显示）字段中，为变量输入一个写入值。

选择滚动条中包含各种在以后联机模式时所可能有的输入。Text（文本）：将打开一个编辑字段，你可以在此输入值。Numpad（数字键盘）或 Keypad（小键盘）：将打开一个显示数字键或字符键的图像窗口，你可以在此通过激活相应的按键元素而输入某个值。该特性在必须通过触摸屏操作时非常有用。输入值的有效范围可以通过编辑字段 Min:（最小）和 Max:（最大）进行限制。

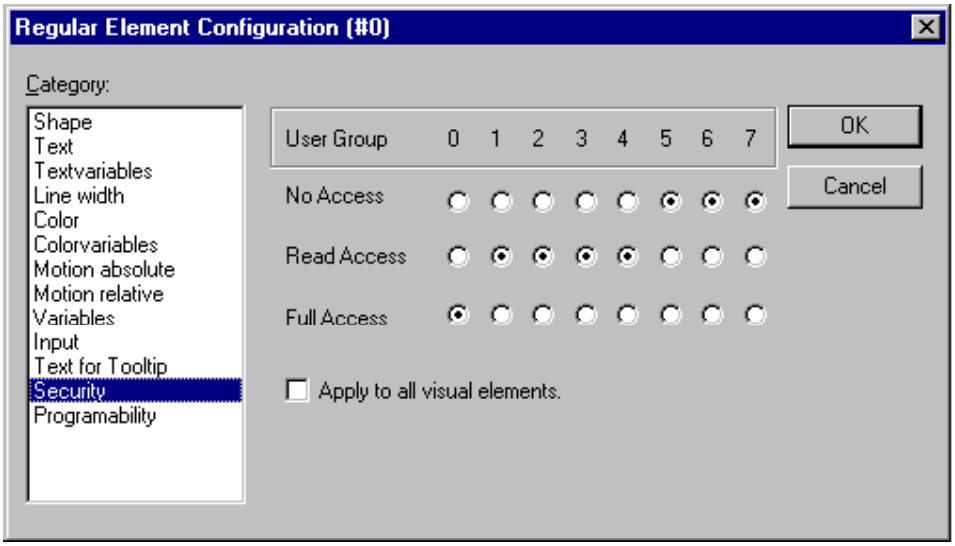
12.5.15 ToolTip（工具条提示）

用于工具条文本的对话框为文本提供了一个输入字段，在联机模式时，只要鼠标光标经过对象，将显示该文本字段内容。该文本可以使用行回车符格式化，可以使用键组合<Ctrl> + <Enter>。

12.5.16 Security（安全性）

为不同的用户组分配不同的操作权限和显示可视化的能力是非常有用的。可以通过给特定的可视化元素分配不同的操作权限实现。你可以在 TwinCAT 中分配八个用户组(参见'Project'（项目） 'Object'（对象） 'Properties'（属性）以及 'Project'（项目） 'User Group Passwords'（用户组密码）)。操作权限的分配可以通过激活可视化元素中'Access rights'（操作权限）配置对话框的相应选项实现。

可视化元素配置对话框(Security（安全性）分类)



可视化元素的操作权限以及它们在联机模式下的效果：

No Access（无权限）	元素不可见
Read Access（读权限）	元素可见，但不可操作(不允许输入)
Full Access（全部权限）	元素可见，并且可操作

如果你也想为可视化对象的所有其他元素分配操作权限，只要激活选项按钮 Apply（应用），即可对所有的可视化元素分配操作权限。

注意: 请注意，在'Project'（项目）'Object'（对象） 'Properties'（属性）对话框中为可视化对象设置的操作权限，独立于那些特定的可视化元素！

12.5.17 Programability（编程）

可视化元素的属性不仅可以通过静态或“普通”的项目变量进行设置，而且也可以通过结构变量成员进行设置，该方法被广泛地用于可视化元素的编程。

为此，结构 VisualObjectType 作为一个软件库存放于 SysLibVisu.lib 中。其成员可以用于定义大多数的元素属性。

注意: 在多重定义元素属性的情况下，“普通”项目的变量值将被结构变量值所覆盖，并且两者都将覆盖静态定义。

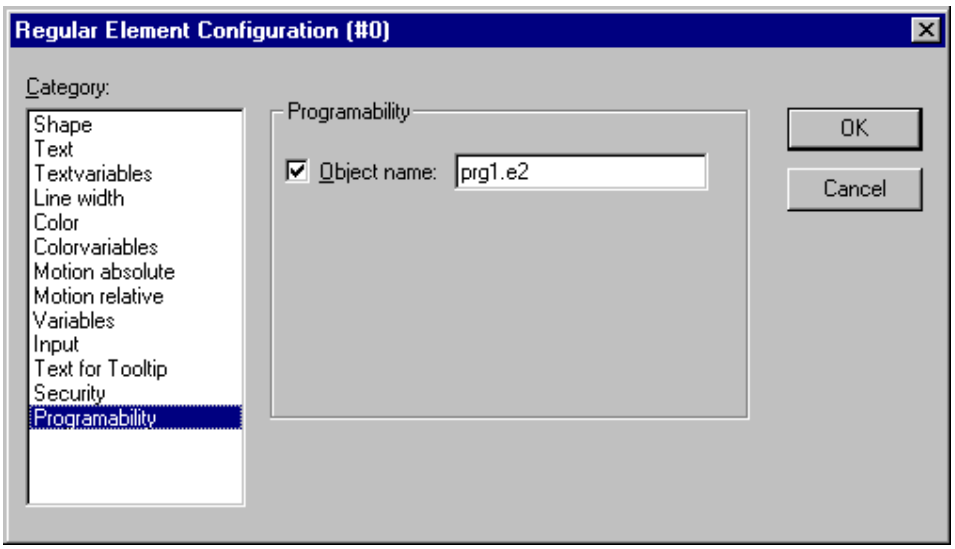
为了使用结构变量配置元素属性，请按照下列方法进行：

打开配置对话框的'Programmability'（编程）分类，并在 Object Name（对象名）字段中输入一个新的、唯一的变量名。为此，你必须通过鼠标点击检查框以激活该选项。该变量被自动声明为 VisualObjectType 类型，该结构包含在软件库 SysLibVisu.Lib 中。变量声明过程是隐式完成的，用户不可见。请确认该库已包含在库管理器中。

当在下一次编译之后，这个新分配的结构变量可以在项目中使用。(提示：在 Editor（编辑器）分类中，激活项目选项的智能功能'List components' （组件列表），以便在变量名后，通过输入点操作符可以看到结构组件的选项列表)。

示例：如果你已经为可视化元素定义了一个对象名为 "visu1_line" 的对象，则可以对该元素的线宽度进行编程；例如："visu1_line.nLIneWidth:=4"。

可视化元素配置对话框 (Programmability（编程分类）)



VisualObjectType 结构：

下表显示了该结构的所有成员以及在配置对话框中不同分类参考的相应项目：

在成员名的开始处插入下面的数据类型缩写：

n INT
dw DWORD
b BOOL
st STRING

成员(+数据类型)	效果	示例 (对象名 "vis1" 已经为元素进行了定义)	配置对话框相应项目
nXOffset : INT;	X-方向移动元素	vis1.nXOffset:=val2; (元素位置设置为 X=val2)	- 绝对移动：X-偏移
nYOffset : INT;	Y-方向移动元素	vis1.nYOffset:=val2; (元素位置设置为 Y=val2)	- 绝对移动：Y-偏移
nScale : INT;	改变大小	vis1.nScale:=plc_prg.scale_var; (元素大小跟随 plc_prg.scale_var 值的变化而线性变化)	- 绝对移动：比例
nAngle : INT;	围绕元素中心旋转	vis1.anglevar:=15; (元素顺时针方向旋转 15 度)	- 绝对移动：角度
bInvisible : BOOL;	元素可见/不可见	vis1.visible:=TRUE; (元素不可见)	- 颜色：内部无颜色 + 框架无颜色 - 颜色变量：Fillcolor + Framecolor
stTextDisplay : STRING;	元素显示文本	vis1.TextDisplay:='ON / OFF'; 元素显示该文本	- 文本：'Content' 中输入
bToggleColor : BOOL;	当从 TRUE 翻转为 FALSE 时颜色改变	vis1.bToggleColor:=alarm_var;(一旦 alarm_var 变为 TRUE，该元素通过成员 dwFillColorAlarm 变为所定义的颜色，同样 dwFrameColorAlarm 通过'Colorvariables' 或 'Color' 进行设置。	- 输入：变量翻转 + - 变量：改变颜色
bInputDisabled: BOOL;	如果是 FALSE：Input 被忽略	vis1.bInputDisabled:=FALSE; (元素的 input 被禁止)	- 变量：'Disable Input'
stTooltipDisplay: STRING;	工具条文本	vis1.stTooltipDisplay:='开关 .....';	- 工具条文本：'Content' 中输入
dwTextFlags: DWORD;	文本位置： 1 居左 2 居右 4 水平居中 8 居上 10 居下 20 垂直居中 注意：始终只能设置一个水平**和**垂直位置(附加值)!	vis1.dwTextFlags:=24; (文本将被放置在元素的中心 (4 + 20)	- 文本：水平、垂直选项 - 文本变量：Textflags
dwTextColor : DWORD;	文本颜色 (定义颜色，参见该表相关项)	vis1.dwTextColor := 16#00FF0000; (文本为蓝色)	- 文本: 字体 | 颜色 - 文本变量: 文本颜色
nFontHeight : INT;	像素表示的字体高度。范围：10-96	vis1.nFontHeight:=16; (文本高度 16 pt)	- 文本: 字体 | 灰色' - 文本变量: 文本高度
dwFontFlags : DWORD;	字体显示可以使用的标志： 1 斜体 2 粗体 4 下划线 8 取消	vis1.dwFontFlags:=10; (文本粗体显示被取消)	- 文本：字体 | 粗体 - 文本变量：字体标志

	+ 和值组合		
stFontName : STRING;	改变字体	vis1.stFontName:='Arial'; (使用 Arial)	- 文本: 字体 | 粗体 - 文本变量: 字体名
nLineWidth : INT;	框架线宽度(像素)	vis1.nLWidth:=3; (框架宽度为 3 像素)	- 线宽度
dwFillColor : DWORD;	填充颜色 (定义颜色，参见该表相关项)	vis1.dwFillColor":=16#00FF0000; (元素为 "标准" 蓝颜色)	- 颜色: 颜色 | 内部 - 颜色变量: 内部
dwFillColorAlarm : DWORD;	当 bToggleColor 变为 TRUE 时，填充颜色，参见上面) (定义颜色，参见该表相关项)	vis1.dwFillColorAlarm:=16#00808080; (只要变量切换为 TRUE 时，该元素将显示灰色)	- 颜色: 报警颜色 | 内部 - 颜色变量: 报警内部颜色
dwFrameColor: DWORD;	框架颜色(定义颜色，参见该表相关项)	vis1.dwFrameColor:=16#00FF0000; (框架为蓝颜色)	- 颜色: 颜色 | 框架 - 颜色变量: 框架
dwFrameColorAlarm: DWORD;	只要 bFrameColor 变为 TRUE，即填充颜色，参见上面(定义颜色，参见该表相关项)	vis1.dwFrameColorAlarm:=16#00808080; (只要变量 vis1.bToggleColor 变为 TRUE，该框架将显示为灰色)	- 颜色: 报警颜色 | 框架 - 颜色变量: 框架报警颜色
dwFillFlags: DWORD;	由颜色变量定义的颜色，可以显示或忽略 0 = 显示颜色 >0 = 忽略设置	vis1.dwFillFlags:=1; (element gets invisible)	- 颜色: 内部无颜色 + 框架无颜色 - 颜色变量: 填充标志
dwFrameFlags: DWORD;	框架显示: 0 完整 1 虚线 (---) 2 点线 ( ) 3 点划线(_._._) 4 双点划线(_.._..) 8 线外闪烁	vis1.FrameFlags:=1; (框架将以虚线显示)	- 颜色变量: 框架标志

定义颜色值：

示例：e1.dwFillColor := 16#00FF00FF;

颜色值采用 16 进制数表示，并由 Blue/Green/Red (RGB) 分量组成。在各种情况下，16# 之后的最前面两个零必须进行设置，以便使其填充后成为 DWORD 大小。每个颜色值可以使用 256 种颜色(0-255)。

FF Blue（蓝色）分量

00 Green （色）分量

FF Red （红色）分量

可视化元素闪烁示例

在矩形配置中定义一个类型为 VisualObjectType 的全局变量 'blink1'。则在功能块程序中，该值可以使用结构进行修改。

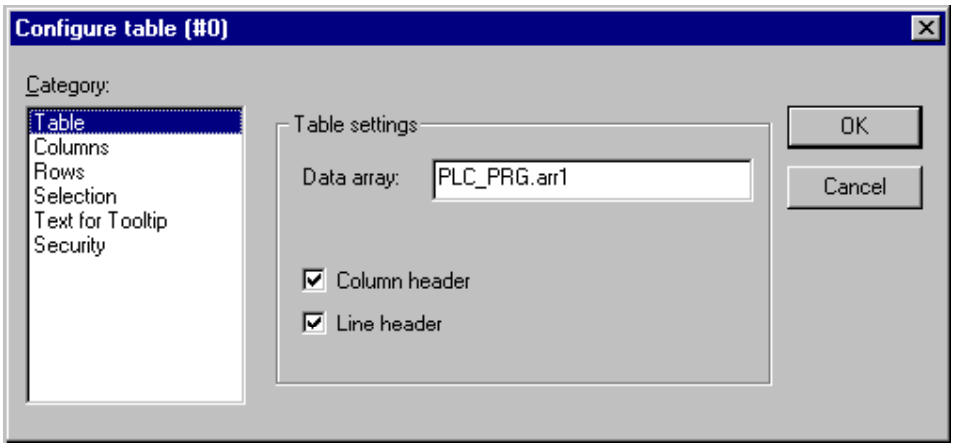
```
PROGRAM PLC_PRG
VAR
n:INT:=0;
bMod:BOOL:=TRUE;
END_VAR
(* 元素闪烁 *)
n:=n+1;
bMod:= (n MOD 20) > 10;
IF bMod THEN
blinker.nFillColor := 16#00808080; (* 灰色 *)
ELSE
blinker.nFillColor := 16#00FF0000; (* 蓝色 *)
END_IF
```

12.5.18 Table（表格）

一旦表格作为数组形式的可视化插入，将打开 Table（表格）配置对话框。此外，'Tooltip'（工具条）和'Security'（安全性）分类也可以用于其它的可视化元素，下面的分类可以配置表格显示和表格内容：

Table（表格分类）：

Table（表格）配置对话框，Table（表格）分类

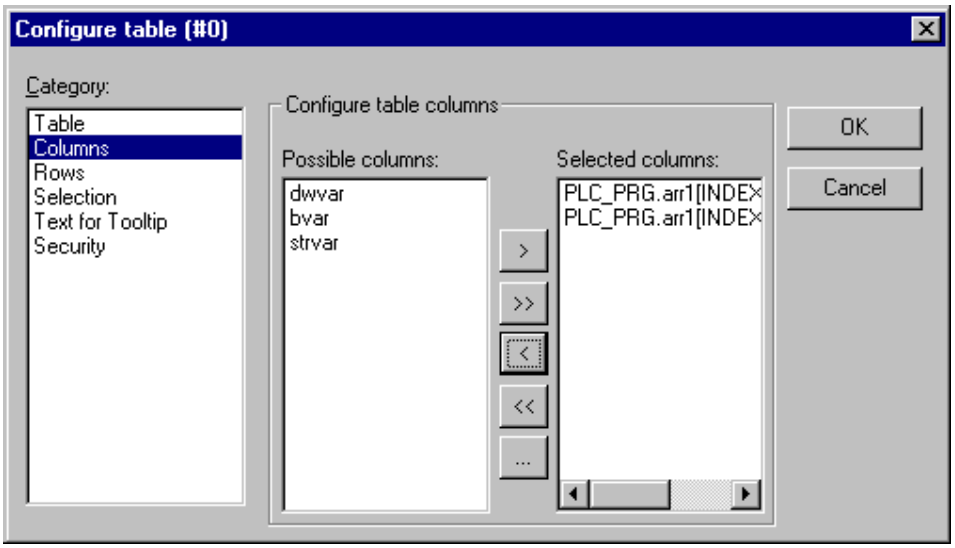


可以对表格进行以下设置：
Data array（数据数组）：插入想要显示在表格中的数组名。推荐使用 input assistant（输入助手）(<F2>) 以及智能功能。

Column header（列标题），Line header（行标题）：如果你想在表格中显示标题，激活这些选项。行标题反映的是数组索引(表格的第一列)，列标题可以在'Columns'（列）分类中定义。

Columns（列）分类：

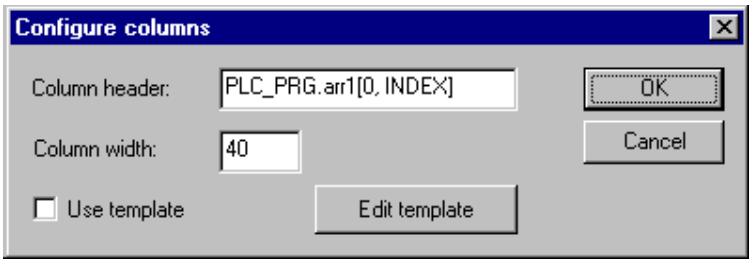
表格配置对话框，Columns （列）分类



在这里你可以定义表格元素。左侧窗口是所有元素的列表，并作为数组的每个索引处理。在结构数组的情况下，这些是结构成员。

使用箭头按钮 > ，你可以从左侧窗口向右侧窗口传送选择的成员，即你设置定义表格将要显示的元素。按压按钮 >>，所有的元素可以通过一次单击动作传送。同样，你也可以从已定义的设置中删除元素(<，<<)。为了修改有关表格列中一个元素显示的缺省设置，可以在右侧部分的窗口中双击想要的栏目，或按按钮 '...' 打开'Configure columns'（列配置）对话框：

表格配置对话框，列分类，列属性



编辑列标题和列宽度：
初始时，该编辑字段的列标题将包含一个自动生成的标题(例如："MAIN.arr1[INDEX].iNo"，在结构数组的情况下，列代表结构成员 "iNo")。并可以设置改变列宽度为其它值(字符个数)。

编辑一列中所有元素的配置参数：

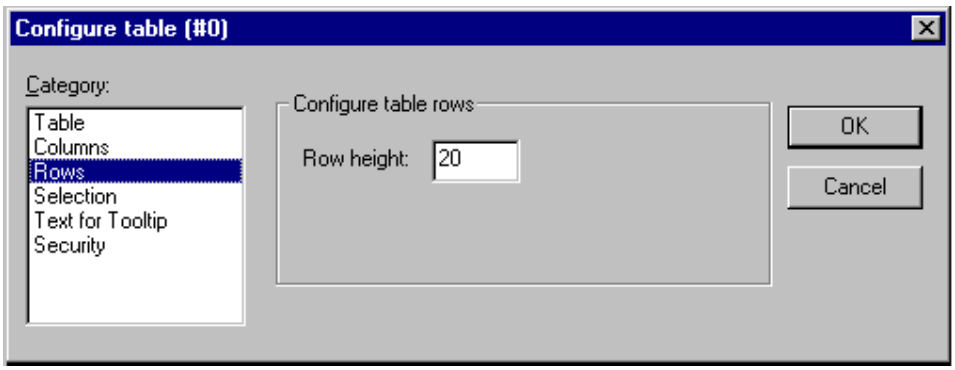
缺省情况下，表格字段显示为简单的矩形，其条目不允许编辑。但是，如果你激活选项：Use template（使用模板），则可以使用预定义的参数设置，例如：线宽度，文本输入的可能性等。为了编辑这些参数，通用配置对话框如：rectangles（矩形），bitmaps（位图），buttons（按钮）等均可以使用，并可以通过点击鼠标按钮编辑模板。

为了定义一列中所有条目的联机动作，即：数组维数中一个维数的所有数组元素，当在配置对话框中输入数组名时应使用占位符 [INDEX] (例如：MAIN.arr1[INDEX]，就像缺省列标题中所使用的一样)。

示例：你想使数组"arr1 [0..2] of BOOL" (表格只有一列) 可视化，并且在联机模式时，通过鼠标点击表格单元时，使其颜色变为红色，并且相应的数组元素自动翻转，反之亦然。为此，激活配置对话框中用于列栏目的'Use template'（使用模板）选项，并按照下述定义模板：'Input'（输入）分类，'Toggle variable'（翻转变量）动作："MAIN.arr1[INDEX]。'Colors'（颜色）分类：报警颜色设置为红颜色。'Variables'（变量）分类，'Change color'（改变颜色）动作："MAIN.arr1[INDEX]。

Rows（行）分类：

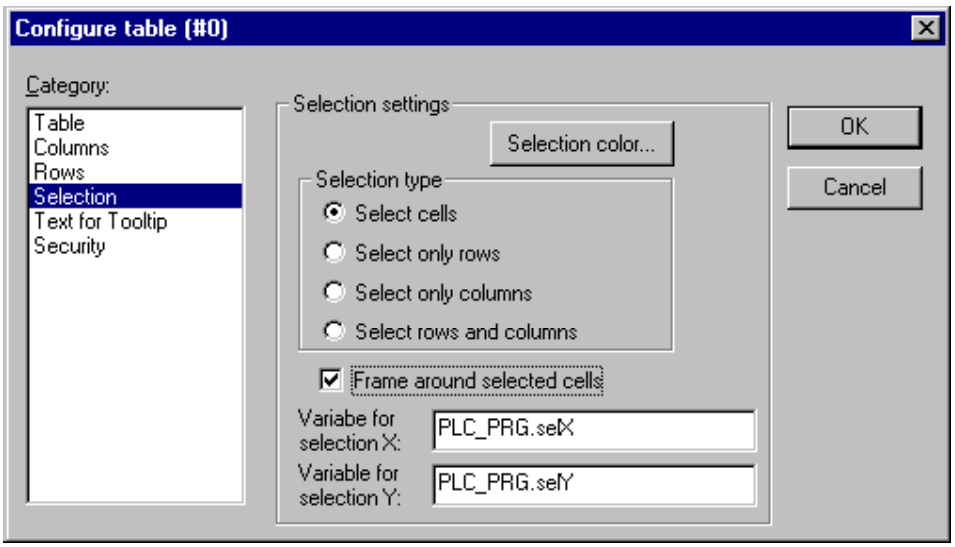
表格配置对话框，Rows（行）分类



Row height（行高度）：插入所希望的以像素为单位的高度。

Selection（选择）分类：

表格配置对话框，Selection（选择）分类



在这里，你可以设置下列有关选择表格特性的参数：

Selection color（选择颜色）：按压该按钮，可以打开标准的颜色对话框，并对选择的单元选择颜色。

Selection type（选择类型）：联机模式时，当你在表格的一个字段中点击鼠标时，定义表格的哪个部分将被选择：

Select single cells（选择一个单元）：只有单元可以被选择。

Select only rows（选择行）：整个行将被选择。

Select only columns（选择列）：整个列将被选择。

Select rows and columns（选择行和列）：所有行和列将被选择。

Frame around selected cells（框架环绕选择的单元）：选择的单元将由一个框架环绕。

Variable for selection X（选择 X 变量），Variable for selection Y（选择 Y 变量）：在这里，你可以分别输入项目变量，并分别显示所选择的表格单元 X- 与 Y-索引。

示例：创建一个表格元素可视化数组结构：

定义下面的结构：

```
TYPE strucTab :

STRUCT

iNo  : INT;
bDigi : BOOL;
sText:STRING;
byDummy: BYTE;

END_STRUCT
END_TYPE
```

在 MAIN 中定义下面的数组：

```
arr1:ARRAY [1..5] OF strucTab;
```

以及下面的变量：

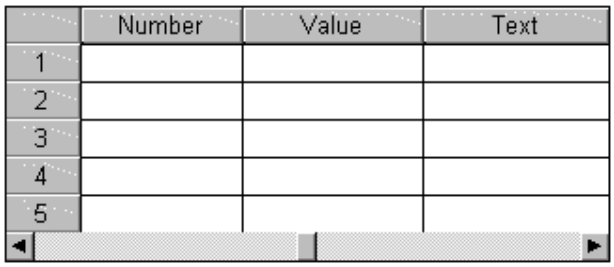
```
selX:INT;

selY:INT;
```

创建一个可视化对象，并插入一个表格元素。按照下面进行配置：

Table（表格）分类：数据数组："MAIN.arr1"

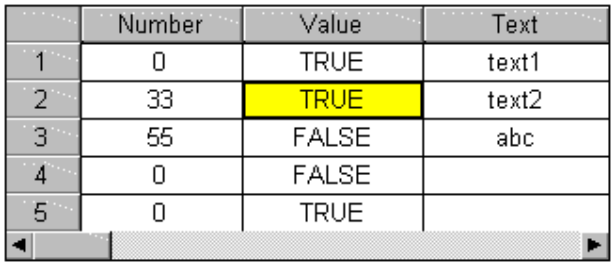
Columns（列）分类：传送成员 iNo，bDigi，sText 到右侧窗口 – 在右侧窗口中的第一个条目处进行一次双击(MAIN.arr1[INDEX].iNo)，系统将打开一个对话框，并使用 "Number" 替代缺省的标题。用 OK 确认，并且也可以为另外两个条目定义新的列标题(例如："Value" 和 "Text")。

'Selection'（选择）分类：在这里分别输入 'Variable Selection X'（变量选择 X）："MAIN.selX" 和 'Variable Selection Y'（变量选择 Y）：MAIN.selY"。激活选项 'Frame around selected cells'（框架环绕选择单元）。按压按钮 'Selection color'（选择颜色），并选择'黄'颜色。使用 OK 关闭配置对话框。则表格元素应按如下显示：



最左侧显示的数据是数组的索引，最上部是所选结构成员的标题。你可以通过将光标放置在两个纵向栏目之间的分隔点上，当鼠标变为水平双向箭头时，移动鼠标修改纵向栏目的宽度。

联机模式时，数组元素的当前值将在表格单元中显示。一旦你使用鼠标点击选择表格的一个单元，它将用黄色显示，并伴随有一个环绕的外形框。示例：



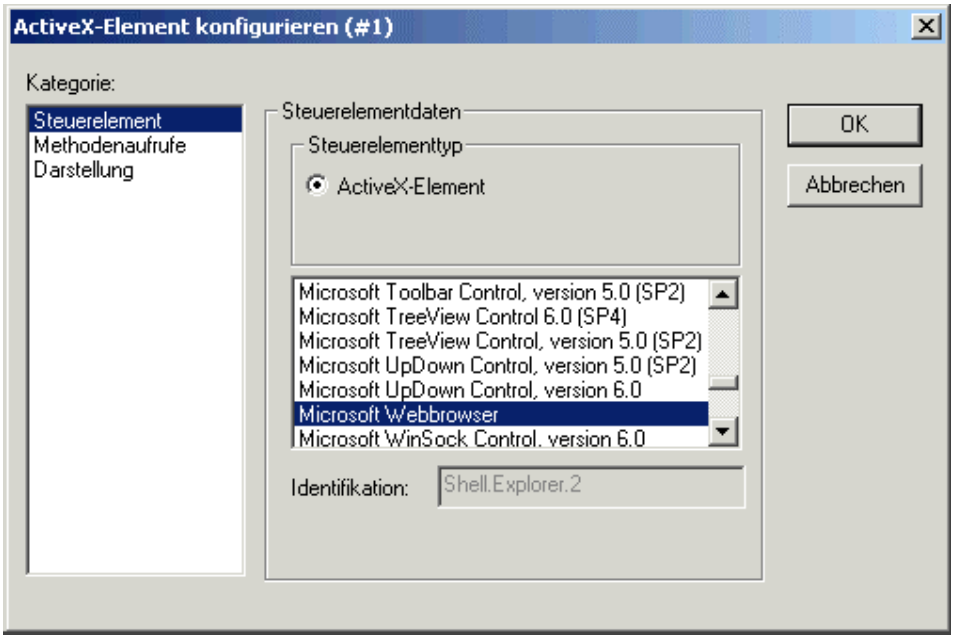
12.5.19 ActiveX-Element（ActiveX-元素）

ActiveX-Element（ActiveX-元素）在一个可视化中被服务用于显示 ActiveX 控制。该元素可用于 Windows32 系统 TwinCAT HMI 以及 TwinCAT Target - Visualization（目标-可视化）。

双击插入的元素可以打开配置对话框，并提供有三个子对话框，分别用于选择 control （控制）类型、定义 method calls（方法调用）和配置 display（显示）：

1. Control（控制）：

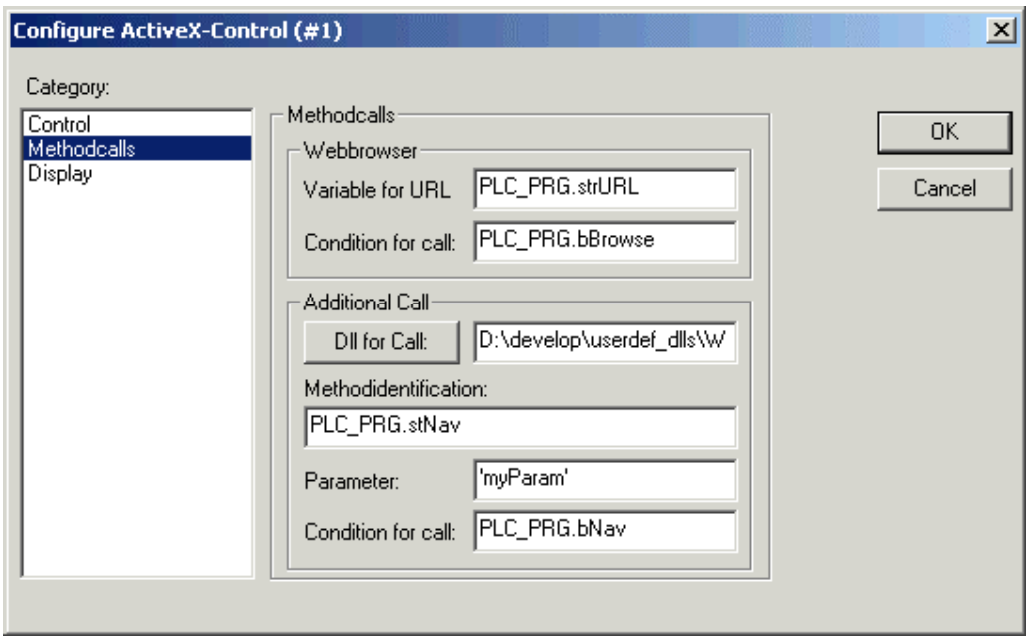
ActiveX-Element（ActiveX-元素）配置对话框，Control（控制）分类



在该对话框中，你可以从你计算机上已经注册的 ActiveX 控件选择列表中标记所希望的 ActiveX 控件。

2. Methodcalls（方法调用）：

ActiveX-Element（ActiveX-元素）配置对话框，Methodcalls （方法调用）分类



在这里，你可以对所选择的 ActiveX 控件配置方法调用：

Webbrowser（万维网流览器）：

如果你配置的一个控件元素支持 IWebBrowser 接口，这些输入字段才允许编辑。(例如：Internet 流览器或 Mozilla 流览器)。此时，TwinCAT 可以直接调用 Navigate （定位）方法(其它方法调用必须通过用户定义的 Dll 进行控制，参见下面'Additional Call'（其他调用）)。

在 URL 变量字段中输入 URL 的参数值 (输入使用反斜线分隔的字符串)或使用类型为 STRING 的项目变量定义一个 URL。只要条件字段输入的变量变为 TRUE（上升沿触发），流览器即被调用。如果这里没有定义任何条件，则在 Target-Visualization（目标可视化）中，流览器将在可视化任务的每次周期中进行调用！

Additional Call（其他调用）：

通过用户定义的 Windows-Dll，你可以为 ActiveX 控件定义方法调用，以便调用时控制控件的行为。为此，你必须在 Dll for Call（Dll 调用）字段中输入 Dll。如果你按压该按钮，将打开文件对话框流览一个 Dll。如果该 Dll 位于项目选项中指定的可视化目录中，只需输入相对此目录的路径即可，否则，需要输入完整的文件路径。

注意: 如果 Dll 用于目标可视化的实时运行系统，该动态库必须隐含地拷贝到那里。当控件在目标可视化中调用时，只考虑包含在路径中的文件名。

只要条件字段输入的变量变为 TRUE（上升沿触发），该 Dll 即被调用。如果这里没有定义任何条件，则在 Target-Visualization（目标可视化）中，它将在可视化任务的每次周期中进行调用！

当创建相应的 Dll 时，应考虑下列因素：

该 Dll 必须输出一个方法名为 "ExecuteActiveXCall" 的函数原型：

void ExecuteActiveXCall(IUnknown* pUnk, char* pszId, char* pszParam, char* pszReturnBuffer, int nReturnBufferSize, DWORD* pdwReturnFlag);

该函数将使用下列在配置对话框中定义的参数调用：

pszId : 字符串或方法标识符字段中指定的字符串变量。

pszParam : 参数字段中指定的值。

参数 pUnk 允许查询其他的 Com(ActiveX-)接口。通过这些接口，你可以调用自己的 ActiveX 控件中使用字符串封装的任何方法！

参数 pszReturnBuffer, nReturnBufferSize 和 pdwReturnFlag 目前未使用。

Dll 源代码示例：

```
#include "stdafx.h"
#include <unknwn.h>
#include <exdisp.h>
BOOL APIENTRY DllMain(  HANDLE hModule,
                        DWORD ul_reason_for_call,
                        LPVOID lpReserved)
{
    return TRUE;
}
```

```
extern "C" __declspec (dllexport) void ExecuteActiveXCall(IUnknown* pUnk, char* pszId,
char* pszParam, char* pszReturnBuffer, int nReturnBufferSize, DWORD* pdwReturnFlag)
{
    if (strcmp(pszId, "IWebBrowser|GoBack") == 0)
    {
        IUnknown* pNewUnk;
        IWebBrowser* pwb;
        pUnk->QueryInterface(IID_IWebBrowser, (void**) &pNewUnk);
        pwb = (IWebBrowser*) pNewUnk;
        if (pwb)
        {
            pwb->GoBack();
            pwb->Release();
        }
    }
    else if (strcmp(pszId, "IWebBrowser|GoForward") == 0)
    {
        IUnknown* pNewUnk;
        IWebBrowser* pwb;
        pUnk->QueryInterface(IID_IWebBrowser, (void**) &pNewUnk);
        pwb = (IWebBrowser*) pNewUnk;
        if (pwb)
        {
            pwb->GoForward();
            pwb->Release();
        }
    }
}
```

3. Display（显示）：

ActiveX-Element（ActiveX-元素）配置对话框，Display（显示）分类

在该对话框中，你可以指定变量的定义位置(X-Offset（X-偏移），Y-Offset（Y-偏移），参见 'Motion absolute'（绝对运动）分类的描述，以及控制元素的可见性(Invisible（不可见）)。

12.5.20 Meter（仪表）

Meter（仪表）元素配置对话框

只要你在可视化对象中插入一个仪表，该对话框将自动打开。预览是该对话框的一部分，以便立即显示当前设置参数的元素外观：

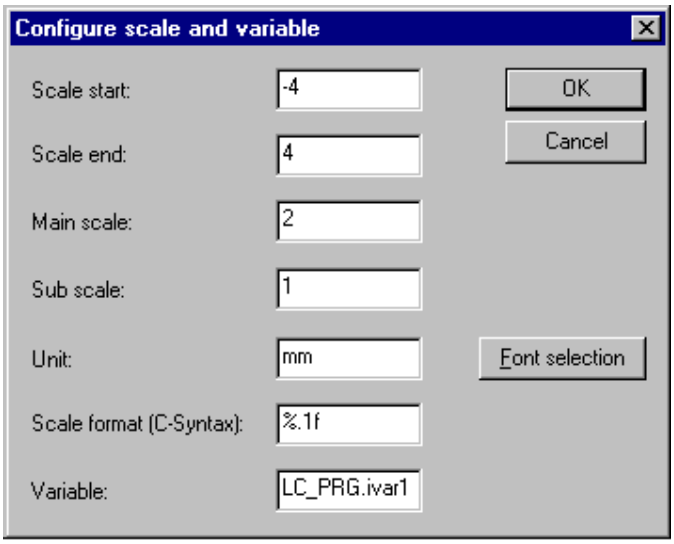
Arrowtype（箭头类型）：定义仪表中指向当前值的箭头类型。包括的类型有：Normal arrow（普通箭头），Thin arrow（细箭头），Wide arrow（宽箭头），Thin needle（细指针）。

Arrow start（箭头起始点），Arrow end（箭头终止点）：在这里，你可以定义以度数（角度）表示的虚拟圆弧起始位置和终止位置。(示例：起始角度 180°，终止角度 0° 将定义一个向上的半圆)。

Arrow color（箭头颜色）：该按钮打开一个选择颜色的标准对话框。定义指针的颜色。

Variable/Scale（变量/比例）：该按钮打开一个配置比例和变量的对话框：

配置 Meter（仪表）元素 Scale（比例）和 Variable（变量）的对话框



Scale start（比例起始），Scale end（比例终止）：比例尺中的最大和最小值，例如："-4" 和 "4"。

Main scale（主项比例）：定义比例尺上所有将被标记的间隔，即比例尺将给出的间隔和标签。例如，如果你插入 "2"，将显示间隔为 2 的整数。

Sub scale（子项比例）：除主项比例(标签+长间隔线)以外，你还可以在此定义一个子项比例，显示时不带任何标签的短间隔线。

Unit（单位）：在此定义比例单位，例如："cm" 或 "sec"。该单位在指针的原点用标签表示。

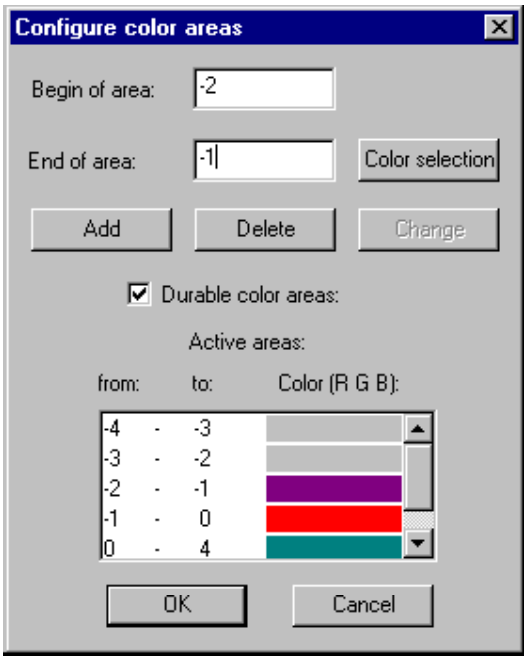
Scale format（比例格式）(C-语法)：按照 C-语法，你可以定义比例标签的显示格式；参见'Text'（文本）分类的相关描述。示例：如果你插入 "%.1f"，该比例值将显示为一个浮点数，即小数点后保留一位值(例："12.0")

Variable（变量）：这里，你可以定义指针位置的变量。(例如："PLC_PRG.posvar")

Font selection（字体选择）：该按钮将打开 Meter（仪表）元素中定义字体的标准对话框。

Color areas（颜色区域）：该按钮将打开颜色区域配置对话框。在这里，你可以给比例尺中的各个部分定义不同的颜色。

Meter（仪表）颜色区域配置对话框



Begin of area（起始区域），End of area（终止区域）：在这里可以插入比例尺各部分的起始和终止值，并使用下面定义的颜色：

Color selection（颜色选择）：该按钮将打开选择颜色的标准对话框，并使用 OK 按钮确认你的选择，然后关闭该对话框，按压按钮 Add（添加），将选定的颜色分配给指定的比例尺区域，并添加到'Active areas'（有效区域）窗口。为了删除一个已有的区域，请选择一个条目，并按压 Delete（删除）按钮。

如果选项 Durable color areas（保留颜色区域）有效，则定义的颜色范围将永久显示，否则，联机模式时，只显示包含当前值的比例尺部分的颜色。

Label（标签）：取决于哪个选项有效(inside（内部）或 outside（外部）)，该比例尺标签将放置在比例尺圆弧的内部或外部。

Additional settings（其它设置）：

Frame inside（框架内部），Frame outside（框架外部）：如果这些选项的一个或两个有效，内部或外部框架将被添加到该比例尺圆弧中。

Additional arrow（其它箭头）：除主指针外，还有一个小箭头将在比例尺上显示当前值。

12.5.21 Bar Display（棒图显示）

Bar Display（棒图显示）元素配置对话框

只要你在可视化对象中插入一个棒图元素，该对话框将自动打开。预览是该对话框的一部分，以便立即显示当前设置参数的元素外观：

Diagram type（绘图类型）：提供的选项有：'Scale beside bar'（比例尺在棒图旁边），'Scale inside bar'（比例尺在棒图内部）以及'Bar inside scale'（棒图在比例尺内部）。

Orientation（方向）：定义选项：Horizontal（水平）或 Vertical （垂直）棒图。

Running direction（运行方向）：选择棒图的动态长度与分配的变量对应关系是 Left – Right（从左到右），还是 Right – Left （从右到左）方向。

Bar color（棒图颜色）：该按钮打开一个选择颜色的标准对话框。定义正常情况下（无报警）的棒图颜色。如果选项 'Use color areas'（使用颜色区域）(参见下面)有效，则该选项将被禁止。

Alarm color（报警颜色）：该按钮打开报警配置对话框，在这里，你可以定义棒图报警颜色显示值和报警颜色：在编辑字段中插入所需要的极限值，并使大于或小于条件中的一个有效，以便定义值在高于还是低于极限值时设置报警的开/关。按压按钮 Alarm color （报警颜色），可以打开选择报警颜色的标准颜色对话框。两个对话框可以使用 OK 按钮关闭，以便确认设置并返回到棒图显示配置主对话框。如果选项 'Use color areas'（使用颜色区域）(参见下面)有效，则该选项将被禁止。

Variable/Scale（变量/比例）：该按钮将打开配置比例和变量的对话框，与 Meter（仪表）元素中的用法相对应。

Element frame（元素框架）：如果该选项有效，则有一个框架环绕棒图显示。

Bar background（棒图背景）： 如果该选项有效，则整个显示范围将为一个带黑色背景的当前值的棒图，否则，仅显示当前值的棒图。

Use color areas（使用颜色区域）：如果该选项有效，则在'Bar color'（棒图颜色）和'Alarm color'（报警颜色）(参见上面)对话框中进行的所有设置均无效。此时，将使用颜色区域定义，即通过'Configure color areas'（颜色区域配置）对话框完成。该对话框可以通过按压按钮 'Color areas'（颜色区域）(参见下面)打开。

Color areas（颜色区域）：该按钮打开配置颜色区域对话框，你可以在此对比例尺的各个部分定义不同的颜色。这些定义只有在选项 'Use color areas'（使用颜色区域）(参见上面)有效时激活。其用法请参见 Meter（仪表）元素的描述。

12.5.22 Histogram（历史趋势图）

历史趋势图元素可用于数组的可视化。数组元素值将由相邻的棒条或线段表示，并通过它们的高度指示该元素的当前值。

Histogram（历史趋势图）配置对话框

只要你在可视化对象中插入一个历史趋势图元素，该对话框将自动打开。预览是该对话框的一部分，以便立即显示当前设置参数的元素外观：

Presentation（表示形式）：激活选项 Barchart（棒图）、Lines（直线）或 Curve（曲线）。

Show horizontal lines（显示水平线）：如果该选项有效，则将附加显示跨越图的等分比例水平线。

Alarm color（报警颜色）：该按钮打开报警配置对话框，你可以在此定义报警颜色显示值及报警颜色：在编辑字段中插入所需要的门槛值，并使大于或小于条件中的一个有效，以便定义值在高于或低于极限值时开/关报警。按压按钮 Alarm color（报警颜色）可打开选择报警颜色的标准颜色对话框。OK 可关闭两个对话框，以便确认设置并返回到历史趋势图的配置主对话框。

Variable/Scale（变量/比例）：该按钮打开比例和变量配置对话框，其用法请参见 Meter（仪表）元素的描述。

Color areas（颜色区域）：该按钮打开颜色区域配置对话框：你可以在此定义比例尺各个部分的不同颜色。参见 Meter（仪表）相同对话框部分的描述。

Bar color（棒图颜色）：该按钮打开选择颜色的标准对话框。定义正常状态时的棒图颜色(无报警)。

定义显示数组的哪个范围：

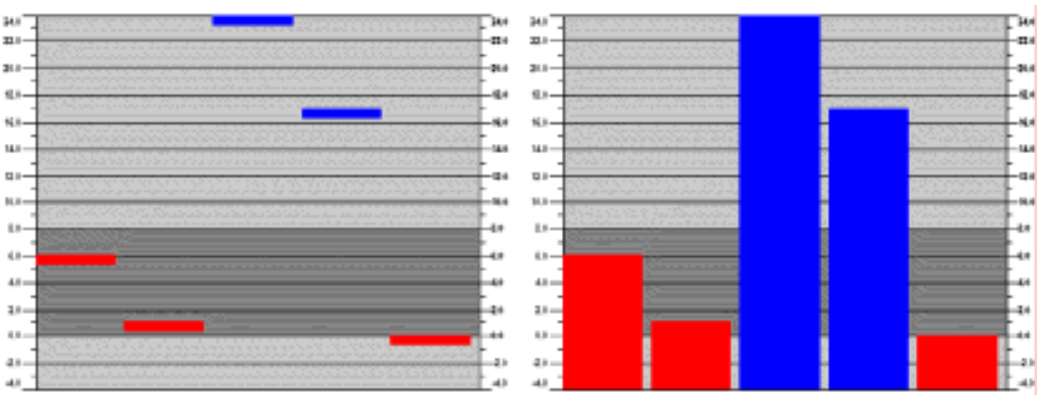
Arraybegin（数组起始）：显示第一个数组元素(索引)。

Arrayend（数组结束）：显示最后一个数组元素(索引)。

Barwidth（棒图宽度）：定义棒图占整个宽度的百分比。

示例：

参见下图联机模式示例，历史趋势图中显示的(棒图、线段)表示数组：arr1 [0..4] of INT。arraybegin（数组起始）设置为 "0"，arrayend（数组结束）为 "4"，scale start（比例起始）为 "–4"，scale end（比例结束）为 "24"，主分度尺设置为 "2"，子分度尺为 "1"，比例尺范围为 0 – 8，比例尺的其它颜色分配为(深灰)。而且，当数组元素相应的值超过 “8” 时，该棒图的报警颜色为(蓝色)。你可以看到数组元素 arr1[2] 和 arr1[3] 当前处于报警状态：



12.5.23 Alarm table（报警表）

'Alarm table'（报警表）元素用于可视化报警，但必须在 TwinCAT 报警配置中预先进行定义。

请注意: 取决于目标系统，也存在着在 PLC 中处理报警的可能性。

只要该元素插入到可视化对象中，将打开 'Configure alarm table'（配置报警表）对话框。除已知的工具条和安全配置分类之外，表格中还有有关显示和选择的分类：

Alarm table（报警表分类）：

配置报警表对话框，Alarm table（报警表）分类

定义你想在报警表中显示的内容：

改变报警组：按压该按钮，进入选择报警配置树型结构，其中有当前定义的所有报警组。选择所希望的组 (即使只有一个报警)。

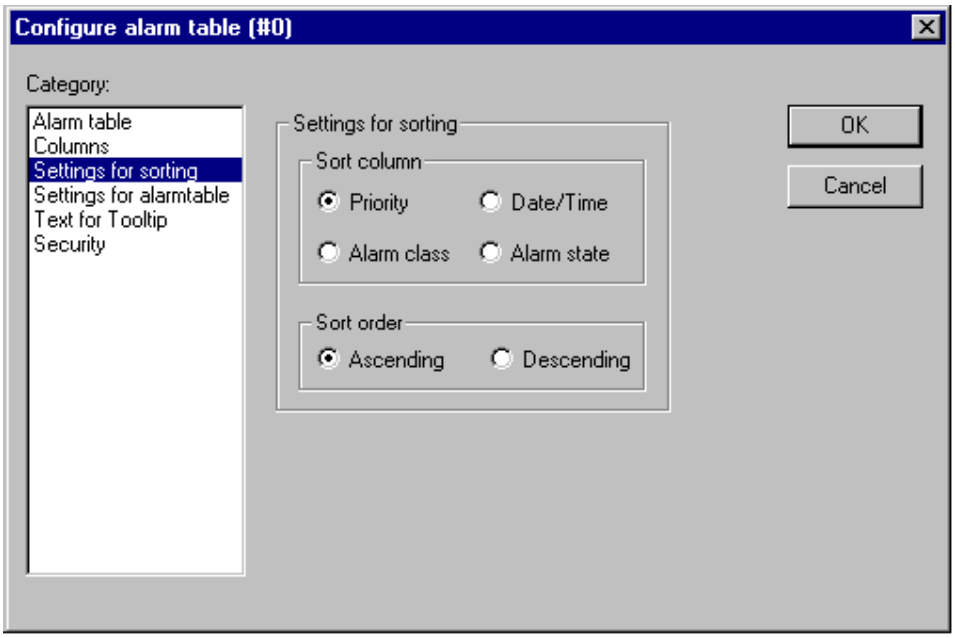
Priority（优先级）：定义你想显示的所有报警的优先级。允许的范围：0 到 255。

Alarm classes（报警等级）：标记你想显示的等级，并按压按钮 Add （添加）将该等级添加到 'Alarm classes'（报警等级）窗口列表中。 所有需要的等级均可照此办法进行。为了从等级窗口中删除一个标记的条目，可按压 Delete（删除）按钮。

如果需要在报警表中显示表头，可以分别激活选项 Column heading（列表头）和 Row heading（行表头）。

Settings for sorting（排序设置）分类：

配置报警表对话框，Settings for sorting（排序设置）分类



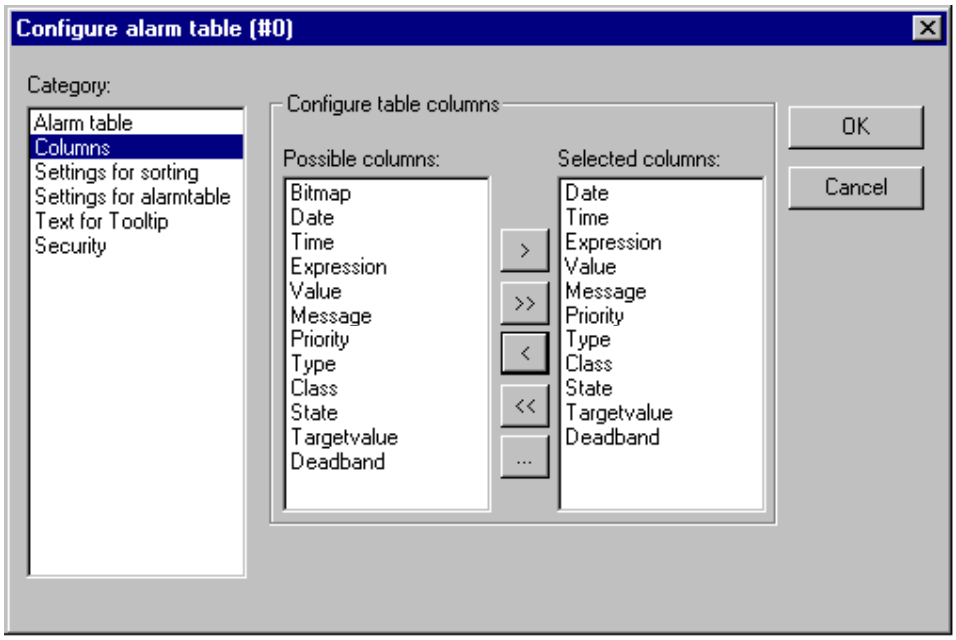
可在这里按照某种规则定义排序报警表：

Sort column（栏目排序）：按照 Priority（优先级）、Alarm class（报警等级）、Date/Time（日期/时间）或 Alarm state（报警状态）排序。

Sort order（排序顺序）：Ascending（升序）或 Descending（降序）；示例：按照优先级升序排列，即该表将从报警优先级 0 开始(如果存在)，后跟较高号码的优先级。

Columns（栏目）分类：

配置报警表对话框，Columns（栏目）分类



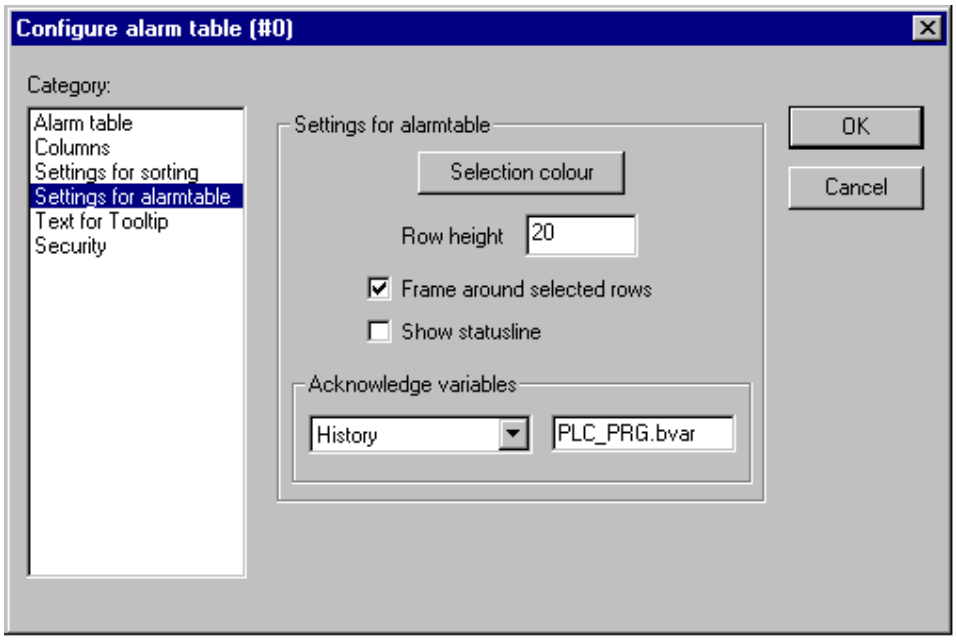
可以在此定义，哪个栏目 (报警参数) 将在报警表中显示：可以定义参数 – 日期和时间除外 (报警发生时刻)，以及报警组配置中的报警状态：Bitmap（位图）、Date（日期）、Time（时间）、Expression（表达式）、Value（值）、Message（信息）、Priority（优先级）、Type（类型）、Class（等级）、State（状态）、Target value（目标值）(用于报警类型 DEV+ 和 DEV-)、Deadband（死区带）。

使用按钮 ">"，">>"，只需单次点击即可将所有的参数从窗口左侧移动到窗口右侧。右侧窗口所定义的选项将显示在报警表中。使用按钮 "<" 和 "<<"，可以删除所选择的条目。

对于每个栏目，你可以通过双击右侧窗口条目打开 'Configure columns' （配置栏目）对话框。并在该对话框中定义 Column header（栏目头）和 Column width（栏目宽度）。

settings for alarm table（报警表设置）选项分类：

报警表配置对话框，'Settings for alarmtable'（报警表设置）分类



可以在此对所选择的表字段显示进行设置：

Selection color（颜色选择）：该按钮将打开选择颜色的标准对话框。并对选择的字段定义显示颜色。

Row height（行高度）：以像素表示的表格行高度。

Slider size（滑动条尺寸）：表格底部的滑动条高度 (像素)。

Frame around selected rows（环绕选择行的框）：如果该选项有效，所选表格的行将环绕一个框。

Show statusline（显示状态行）：如果该选项有效，则在该报警表的底部将显示一个状态行，并提供如下按钮用于联机模式操作：

Acknowledge（应答）：报警表中所有被标记的条目均被应答。

Acknowledge all（全部应答）：报警表中列举的所有报警条目均被应答。

History（历史）：如果按压该按钮，该表将显示一个到目前为止，已经发生的、完整的、所有事件的列表，并以此内容取代当前的状态行。该列表不能进行应答! 任何新发生的事件将被添加作为当前的事件。

如果你已经定义了一个记录文件，你也可以在那里找到这些所有报警等级的历史记录，只要动作 'Save' （保存）已被激活即可。

Start（启动）：取消 Stop（停止）(参见下面)

Stop（停止）：最新发生事件的当前列表更新将被停止，直到通过按压按钮'Start'（启动）时为止。

Acknowledge variables（变量应答）：该选项只有当没有选择选项 'Show statusline' （显示状态行）(参见上面)时有效。如果该选项有效，上面描述的状态行字符串功能的按钮可以通过变量控制。为了定义这些变量，从选择列表中选择一个功能，并在编辑字段中分配

一个项目变量。例如，对联机模式时所有报警的应答可以通过所分配的变量产生上升沿来完成。

12.5.24 Trend（趋势图）

. Trend（趋势图）元素配置对话框

Trend（趋势图）元素可用于记录联机模式时的变量值时间相关特性。与跟踪功能相似。并在联机模式时使用图形表示，在使用文本文件记录的情况下，每个值都采用一个独立的行记录。

请注意: 取决于目标系统，也存在将趋势图数据保存在 PLC 中的可能性。

在配置可视化元素的'Trend'（趋势图）分类对话框中，你可以进行下面的设置：

Curve（曲线）：X/t，horizontal axis（水平轴）= 时间轴，vertical axis（垂直轴）= 数值比例

Orientation（方向）：Left-right（从左到右）或 Right.-left（从右到左）：最后一个值将显示在左/右侧；

Axis（轴）：

Horizontal Axis（水平轴）：

趋势图元素水平轴配置对话框

Division lines（分度线）：激活选项 'visible'（可视），水平分度线将按照比例尺拉长显示。此时，应定义 'Scale'（比例尺）：给定的数值将定义水平轴的分度线间隔。类型(normal（普通线）___，dashed（虚线） _ _ _，dotted（点线） .....，dashdotted（点划线） _ . _ .)和线的颜色可以在对话框中进行定义，当你使用鼠标在相应的矩形框中点击时将打开该对话框，并显示当前设置的线类型和颜色。

Scale（比例尺）：这里显示的比例尺范围由 Duration（持续时间）决定。例如，如果定义了 "T#20s0ms"，则该比例尺将显示 20 秒的范围。Main（主）比例尺和 Sub scale（子比例尺）分度，将通过相同语法定义的长/短标记进行显示。

Degree of accuracy（精度）：在这里定义 (时间的标准格式，例如：T#5ms)当前显示值的间隔变量。

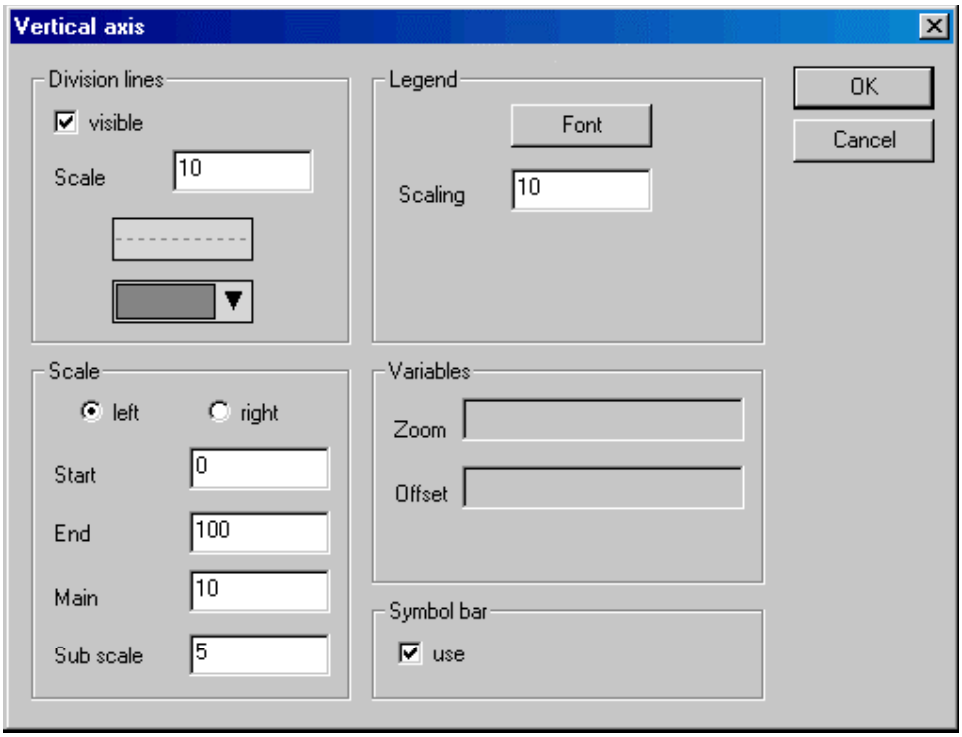
Legend（图例）：在这里可以定义显示的图例。通过按钮 Font（字体）可以打开设置字体的标准对话框。Scaling（比例尺）定义比例尺中显示文本的间隔距离(例如：T#4ms，如果比例尺标记将要显示 4 毫秒的文本间隔)。该文本可以包含时间或日期，它取决于哪个选项被激活。所需要的格式可以分别在'Format'（格式）字段中定义。

Variables（变量）：在这里你可以定义项目变量，即分别用于水平比例尺的缩放值和偏移量。例如，一旦偏移变量被赋值为 10，则水平轴的显示范围偏移量将被设置为 10。

Symbol bar（符号条）：如果选项 use（使用）有效，则在该元素的底部将添加一个水平符号条，即用于联机模式时滚动和缩放的按钮。单箭头按钮将按步在时间轴上移动显示范围，双箭头按钮则将记录移动到起始点或终止点。Zoom（缩放）按钮允许在水平比例尺方向按步缩放。可以通过它将有关缩放和偏移恢复到其原始值，定义 vertical（垂直）符号条可以进行'home'（初始状态）操作。

. Vertical axis（垂直轴）：

趋势图元素垂直轴配置对话框



Division lines（分度线）：与水平轴相同(参见上面)

Scale（比例尺）：定义是否按比例显示趋势图的左或右边界。可以选择比例尺的 Start（起始）值(低端)和 End（终止）值(高端），以及比例尺的 Main（主项）和 Sub（子项）分度线(这里定义的长/短标记将被显示)。

Legend（图例）：Font（字体）和 divisions（分度）；参见上面的水平轴。

Variables（变量）：参见上面水平轴

Symbol bar（符号条）：参见上面，horizontal axis（水平轴），另外，还有一个"home"（初始复位） 按钮用于恢复与该轴有关的缩放和偏移标准设置。

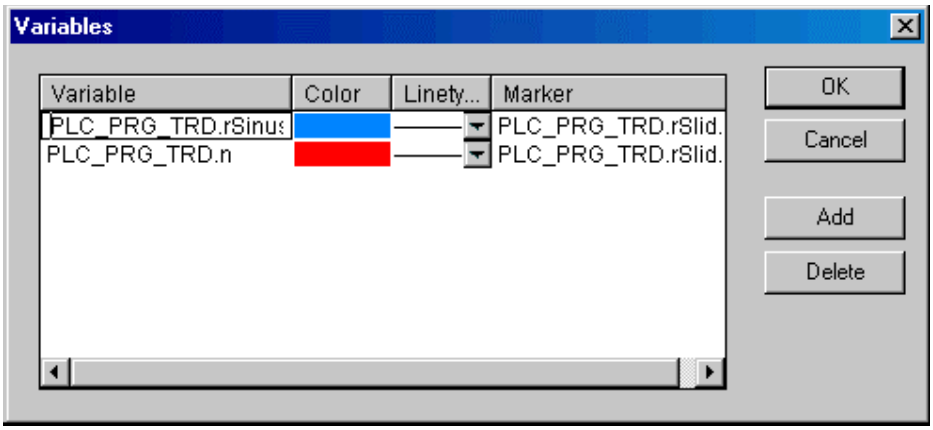
Recording（记录）：在这里定义该趋势图是否‘联机’记录，即：与时间相关的变量值特性将使用所选择的比例范围进行显示，或者，该记录是否保存到历史记录文件中，可以通过按压 History（历史）按钮进行配置。用于配置报警记录文件的相应对话框将被打开。

请注意: 如果目标指定选项 'Store trend data in the PLC' （趋势数据存储到 PLC）有效，则这里定义的路径将被忽略，记录文件将被存储到 PLC 的下载目录中。

在记录文件中，每一次测量都会写入包括名称和所需观测的所有变量值的一个分隔行。每行都使用 DWORD 格式的唯一标识符开始，并由日期表示。

Choose variable（选择变量）：按压该按钮可以打开变量对话框，在这里可以配置变量，确定趋势图记录动作，以及如何显示：

趋势图元素的变量配置对话框。



在 Variable（变量）列中输入一个项目变量 (鼠标点击该字段将打开一个编辑框)。推荐使用输入助手<F2> 或智能功能。

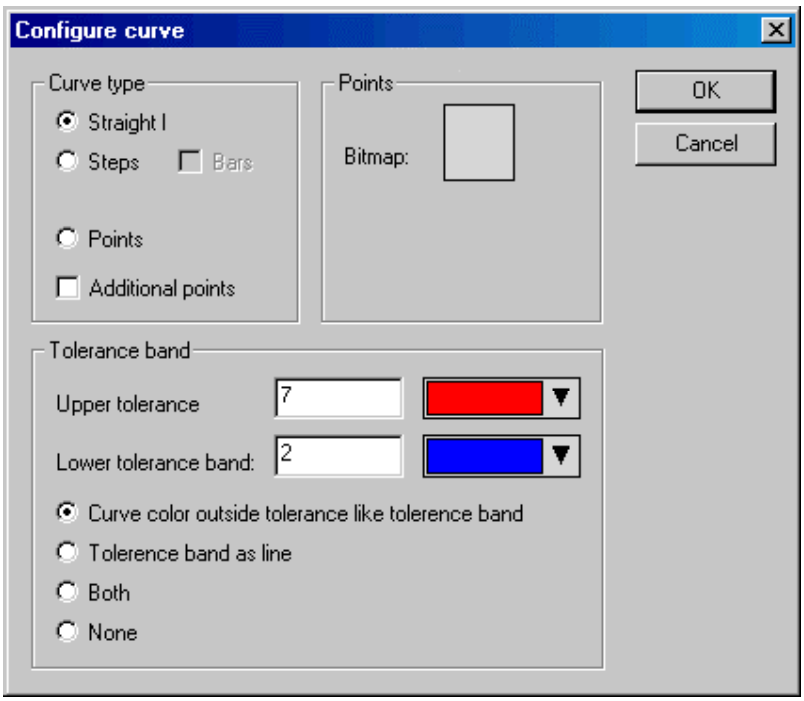
记录中显示变量的 Color（颜色）和 Line type（线型），可以通过鼠标点击相应的 Color（颜色）列(选择颜色的标准对话框) 和 Line type（线型）(实线 ___，下划线 _ _ _，点线 .....，点划线_ . _ .)列字段分别进行定义。

当联机使用标记功能时，Marker（标记）列定义的变量可提供当前的记录值。该标记将在趋势图的左上角显示为一个小灰色三角形。当你点击该三角形并保持鼠标压下时，则可以沿着水平时间轴移动垂直标记线。该变量定义为'marker'（标记），并从相关联的项目变量记录曲线中读出相应的变量值。

设置你想记录的所有变量。通过按钮 Add（添加），将在列表的最后添加一行。Delete（删除）按钮可以删除一行。

Curve configuration（曲线配置）：该按钮将打开曲线配置对话框。在这里可以进行趋势图曲线的相关设置：

趋势图元素的曲线配置对话框



Curve type（曲线类型）：选择项 Straight line（直线），Steps（台阶）或 Points（点）。对于前两种显示类型可以定义 Additional points（附加点）。对于一个点的显示可以定义位图，否则将使用填充矩型(与曲线颜色相同)作为点符号。按压 Bitmap（位图）旁的矩形可以打开选择位图文件的标准对话框。通过 Delete（删除）按钮，当前设置的位图可以从配置中删除。

Tolerance band（误差带）：你可以在垂直轴上定义显示误差带的上限和下限值。每个误差带均可以定义一个颜色(按压颜色矩形可以打开选择对话框)。如果联机模式时想显示该误差带，可以激活选项 Tolerance band as line（误差带显示为直线）。如果你希望该曲线在超出误差值时显示分别定义的误差带颜色，请激活 Curve color outside tolerance like tolerance band（误差带之外的曲线颜色显示为所定义的误差带颜色）。如果你想同时激活上面描述的两者或无显示选项，可以激活 Both（两者）或 None（无）。

示例：联机模式时显示趋势图元素：

PLC_PRG 程序中声明：

```
VAR
n: INT;
rSinus:REAL;
rValue:REAL;
rSlider1:REAL; (* 用于标记功能 *)
rSlider2:REAL; (* 用于标记功能 *)
END_VAR
```

PLC_PRG 中的程序：

```
n:=n+1;
rValue := rValue + 0.01;
rSinus:=SIN(rValue)*50 + 50;
IF n>100 THEN
n:=0;
END_IF
```

可视化中配置趋势图元素：

方向从左到右，激活历史趋势。

Horizontal axis（水平轴）：Division lines（分度线）：T#2s，Duration（持续时间）：T#10s，Main（主分度线）：T#1s，Sub scale（子分度线）：T#500ms，Degree if accuracy（精度度数）：T#200ms，Legend（图例）：时间格式('hh':'mm':'ss')，分辨率 T#2s。符号条必须激活。

Vertical axis（垂直轴）：Division lines visible（分度线可见），Scale（标定线）：10，点线，灰色；左标定，Start（起始）：0，End（结束）：100，Main（主分度线）：10，Sub scale（子分度线）：5；Legend（图例）：10；符号条必须激活。

变量：

1. 变量 PLC_PRG.rsinus，蓝色线，Marker（标记）：PLC_PRG_TRD.rSlider1；

2. 变量 PLC_PRG.n，红色线，Marker（标记）：PLC_PRG_TRD.rSlider2

Curve configuration（曲线配置）：直线，无误差带

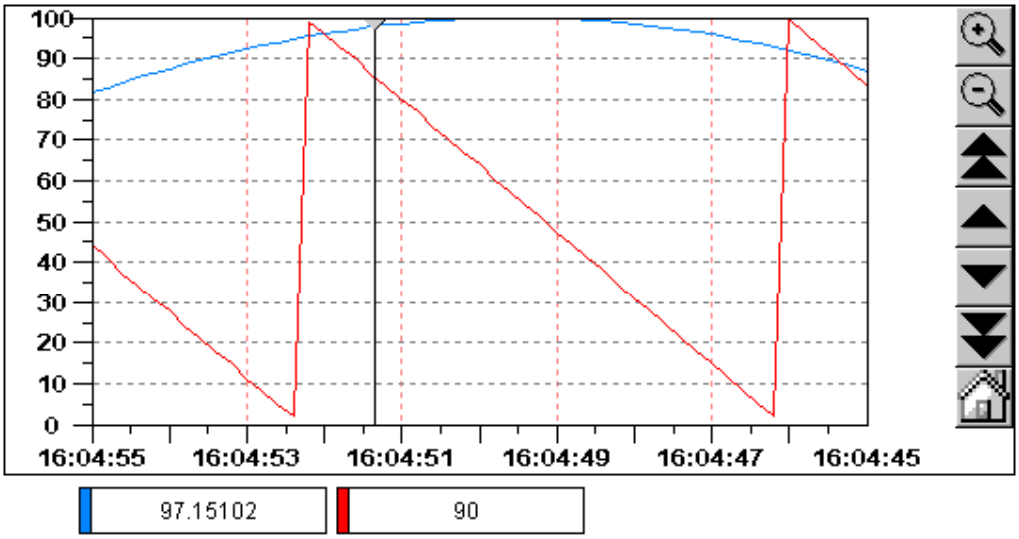
由标记变量提供的当前记录值的两个显示字段的配置：

Rectangle element 1（矩形元素 1）：Category Text（文本分类）：内容字段中插入"%s"；Category Variables（变量分类）：文本显示字段中插入 PLC_PRG.rSlider1

Rectangle element 2（矩形元素 2）：Category Text（文本分类）：内容字段中插入"%s"；Category Variables（变量分类）：文本显示字段中插入 PLC_PRG.rSlider2

(另外，在矩形元素 1 和 2 的边框左侧分别插入一个矩形元素，分别显示相应记录变量的曲线颜色)

系统录入并且启动程序之后，联机模式时的效果图：



该记录从左向右运行；最后的值显示在最左侧的位置；每隔 200 毫秒当前值被添加到该显示中。符号条中的箭头按钮允许移动显示的时间范围。使用单箭头按钮可以一步一步移动，使用双箭头按钮可以分别到达记录的起始或结束位置。示例：你可以按压指向左侧的双箭头按钮到达记录的起始位置，并查看以前的值。如果你沿着时间轴移动标记(左上角的灰色三角形)，则你可以在该图下面的两个矩形元素中分别读出两个记录变量的准确值。

12.5.25 Bitmap（位图）

你可以在可视化元素配置对话框的 **Bitmap（位图）**分类中输入位图选项。

在位图字段中输入位图文件和路径。你可以使用 ... 按钮打开标准窗口浏览对话框，并从中选择所需要的位图。

下列项目将影响位图的外形框架：

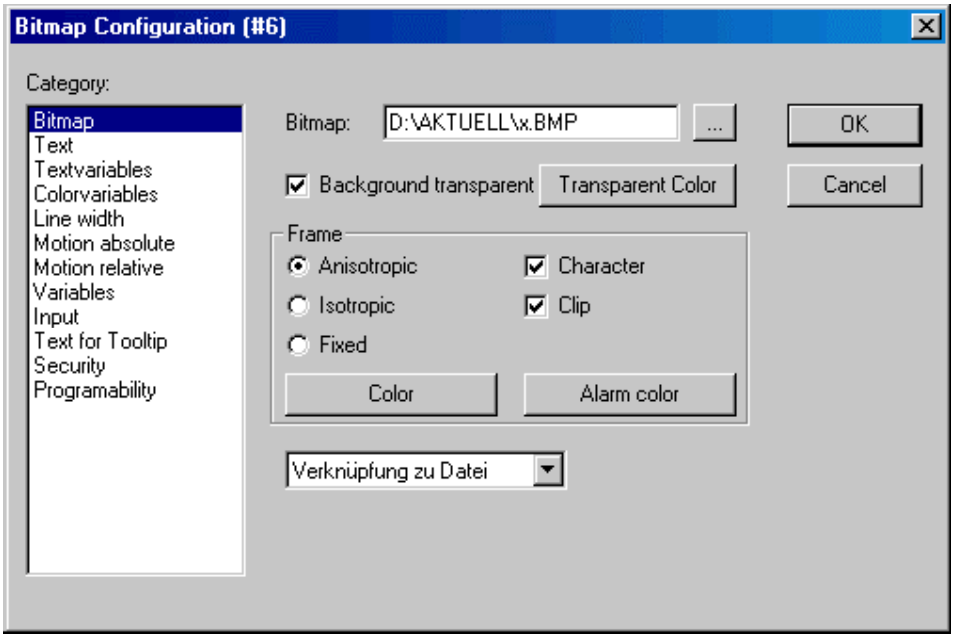
通过选择 Anisotropic（非正交），Isotropic（正交）或 Fixed（固定）方式，你可以指定位图如何影响边框的大小。Anisotropic（非正交）表示：位图保持与框架相同的尺寸，并允许分别改变位图的高度和宽度。Isotropic（正交）表示：位图保持相同的比例，即使整个尺寸发生变化时也是如此(即：高度与宽度的比例关系保持不变)。如果选择 Fixed（固定）方式，位图将保持其原始的尺寸大小，而不管框架的尺寸大小如何。

如果 Clip（夹子）选项与 Fixed（固定）设置方式同时被选择，则只有包含在框架中的那部分位图才能被显示。

如果你选择 Draw（绘图）选项，该框架将以 Color（颜色）和 Alarm Color（报警颜色）按钮的颜色对话框中所选定的颜色显示。报警颜色只有当变量分类中的 Change Color（改变颜色）字段中的变量值为 TRUE 时才起作用。

在对话框下部的选项列表中，你可以定义该位图是插入到项目中(插入)、还是仅仅创建（链接文件）与外部位图文件进行链接(路径输入在上部的'Bitmap'（位图）字段中)。将位图文件保持在项目的目录中是合理的，因为你只需要输入一个相对路径即可。否则，你需要输入一个绝对路径，并且当你想将该项目传送到其它的工作环境中时往往会导致问题的发生。

可视化元素配置对话框(位图分类)



12.5.26 Visualization（可视化）

当你将一个可视化插入到另一个可视化之中时，则创建了该可视化的一个“参考”。

该参考的配置可以在可视化元素配置对话框的可视化分类中完成。

可视化元素配置对话框 (可视化分类)

在可视化字段中输入将要插入的可视化对象名。使用 ... 按钮打开该项目中所包含的可视化。除当前所使用的可视化之外，任何可视化均可以被使用。

有关可视化框架的设置与位图中的描述相同。

Placeholder（占位符）按钮将打开'Replace placeholder'（替换占位符）对话框。在'Placeholder'（占位符）栏中将列出所有在配置对话框“母板”可视化中所插入的占位符，并在'Replacements'（替换）栏中提供将被替换的当前参考确定值。采取何种替换形式将取决于'Extras'（附加）'Placeholder list'（占位符）对话框中的“母板”可视化是否预定为组合方式。如果是这种情况，它将显示为一个组合框用于选择。如果没有进行任何预定义，双击替换栏中的相应字段时，可以在打开的编辑字段中填写任何所需要的内容。

另外一种替换占位符参考直接发生的可能性是：在配置对话框('Input'（输入）分类)中，通过 Zoom to vis（缩放到可视化）选项字段定义调用一个可视化。

注意: 无按年编排的控制替换顺序是可能的！尽管没有占位符替换文本，但它仍然包括占位符！

注意: 当使用占位符时，不再可能对项目编译时立即检查可视化元素配置的无效输入。因此，相应的错误信息只能在联机模式时发出(...无效监视表达式..)。

应用占位符概念的示例：

功能块实例可以借助于相同的可视化参考轻易实现显示。例如，配置可视化 visu 时，需要观察功能块变量，可以在每个变量之前使用占位符 FUB (例如：FUB.a)。当使用 visu 的参考时(在另一个可视化中插入 visu，或通过 'Zoom to vis.'（缩放到可视化）调用)，则在配置参考时，占位符 FUB 可以使用需要观察的功能块实例名替换。

该过程可参见下面：

在项目中，定义包含如下声明的功能块：

```
FUNCTION_BLOCK fu
VAR_INPUT
changecol : BOOL; (* 改变可视化的颜色 *`)
END_VAR
```

在 PLC_PRG 中定义两个'fu' 的实例：

```
inst1_fu : fu;
inst2_fu : fu;
```

创建一个可视化对象 'visu'。插入一个元素并打开配置对话框 'Variables'（变量）分类。在 'Change color'（改变颜色）字段中输入："FUB.changecol"。打开'Input'（输入）分类并在'Tap Variable'（变量塞）字段中输入"FUB.changecol"。打开'Text'（文本）分类并输入"FUB - change color "。在'Colors'（颜色）分类中定义一个报警颜色。

创建另外一个可视化对象 'visu1'。

在 'visu1' 中插入两个 'visu' 可视化(两个'visu' 参考)。

标记第一个 'visu' 参考，并打开 'Visualization'（可视化）分类的配置对话框。按压按钮 'Placeholder'（占位符），这样可显示占位符列表。并在此将 'FUB' 替换为 'PLC_PRG.inst1_fu'。

现在，标记第二个 'visu' 参考，并将(按照第一个描述的过程) 'FUB' 替换为 'PLC_PRG.inst2_fu'。

现在，在联机模式时，两个使用 'fu' 实例配置的变量值将在相应的 'visu' 参考中可见。

当然，占位符 FUB 可以在 'visu' 的任何变量或文本字符串处配置使用。

注意: 可视化参考的联机特性：如果你插入一个可视化，并选择和配置该参考，则它将被看作为一个单一对象，联机模式时的输入响应与相应的配置相对应。与此相反：如果你没有配置该参考，则联机模式时这些特定可视化元素的响应与原始的可视化对象相同。

12.6 可视化联机模式

12.6.1 可视化联机模式

请注意下面有关联机模式的可视化项目：

- **评估次序：**
  - o 动态定义的元素属性（通过变量）将覆盖由配置对话框选项定义的（静态）基本设置。
  - o 如果一个元素属性是通过“普通”项目变量定义的，或者是通过结构变量成员（编程特性）(Programmability)定义的，则应考虑项目变量的初始值。
  - o 可视化可以在联机模式时仅通过键盘输入的方式进行操作配置。这对于使用 TwinCAT HMI 可视化、目标可视化或 Web 可视化而言是非常重要的特性。
- 显示（Display），框架（Frame），语言（Language）的配置设置也可以在联机模式时进行编辑。
- 只要可视化“参考”没有明确地进行配置，在联机模式时该参考的特殊元素的输入特性和其原始的可视化相同。（参考“源”）。
- 当你切换语言（‘附加’（'Extras'）‘设置’（'Settings'））时，仅当该显示在联机模式时才有效。
- 可视化可以联机模式打印。

12.6.2 键盘操作 – 联机模式

为了能够独立于鼠标或触摸屏操作，将可视化完全配置为键盘操作是非常有用的：

下面每个键的缺省设置（组合键）在联机的任何时候都将有效（无需特殊配置）：

按压制表符**<Tab>**键，则选择元素列表中有输入配置的第一个元素。每次按压该键将移到列表中的下一个元素。如果是在表内，则跳到表内的下一个字段。按压该键的同时再按压换档键<Shift>，则选择前一个元素。

箭头键可以在任何方向从一个选择的元素跳到另一个相邻的元素上。

空格键**<Spacebar>**用于执行所选可视化元素的动作。如果该元素有一个文本输出变量，或者，如果它是一个表格字段，则将打开一个文本输入字段，并分别显示该字段变量的文本内容。按压回车键**<Enter>**，则写入该值。

其它用于联机操作的键（组合）可以在‘键盘使用法’（’Keyboard usage’）配置对话框中定义。并且，制表符<Tab>，空格<Space>和回车<Enter>键除具有上面描述的标准使用法之外，还可以分配其它的功能。

联机模式时参考元素的单个特性与可视化的参考特性相同。因此，它们的响应与单个元素通过鼠标和键盘的输入与操作方法相同；工具条的参考显示也与元素相关。当处理该元素列表时，与处理实例一样，当使用制表符从一个输入元素跳到下一个时，处理所有单个元素参考的进程将从该元素列表之前的参考位置跳到该元素列表的下一个。

注： 联机模式时的键盘操作是非常重要的，尤其是在使用 TwinCAT HMI 可视化，以及目标可视化或 Web 可视化时更是如此！

12.6.3 “文件”（'File'）“打印”（'Print'） 联机模式

‘文件’（'File'） ‘打印’（'Print'）用于联机模式时可视化窗口内容的打印输出。当可视化正在移动元素时，被拉伸到窗口边界之外的可视化将产生不连续的打印部分。

12.7 可视化库

可视化也可以保存在库中，并以库形式的 POU 用于项目。它们可以作为参考插入，或在项目的其它可视化输入配置中通过命令“缩放到可视化”（”Zoom to vis.”）进行调用。

注： 项目中使用的可视化应具有唯一的名称。如果来自库中作为可视化形式的实例被调用或参考时，与项目中已存在的对象名相同，则会出现问题。因为，程序在处理参考或可视化调用时，首先处理的是项目中的可视化，之后才装入库形式的可视化。

12.8 隐含变量

下面隐含创建的系统变量可以作为可视化程序使用：

隐含生成的变量	数据类型	功能	当前使用			
			HMI	Simulation	Target Visu	WebVisu
CurrentVisu	String[40]	当前打开的可视化名称。如果该名称发生变化，则改变将传送到其它可视化中。请注意：名称字符串必须用大写字符定义。取决于目标系统，该变量可以在可视化类别的目标设置对话框 中设置有效/失效。	x	x	x	x
CurrentCaller	String[40]	前次打开的可视化名称。用于调用缩放 ZOOMTOCALLER 功能。仅在目标可视化中设置和修改。	-	-	x	-
CurrentLanguage	String[40]	当前设置的语言，存放于语言文件中。语言必须使用大写字符写入。仅在目标可视化中设置和修改。	-	-	x	-
CurrentUserLevel	INT	当前设置的用户等级0..7。该值只能在 TwinCAT 对话框中修改。	x	x	x	x
CurrentPasswords[0 .. 7]	ARRAY [0..7] of String[20]	所有的密码都在 TwinCAT 的用户组密码（"Usergroup passwords“）中定义。该值只能在 TwinCAT 对话框中修改。	x	x	x	x

12.9 专用可视化键组合

下表显示了所有可能的、由键盘使用法所支持的、专用可视化变体的键组合。
使用下面的缩写：
C 用于 TwinCAT / TwinCAT HMI **TV** 用于目标可视化
WV 用于 Web-可视化
如果列中有缩写，则表示有对应的键组合支持相关的可视化变体。
一些特定行的注释可以在表下面找到。

	无修饰符	Shift	Ctrl	Shift+Ctrl	注解
VK_TAB	C	C	C	C	*K4*
VK_RETURN	C / TV	C / TV	C / TV	C / TV	
VK_SPACE	C / WV	C / WV	C / WV	C / WV	*K4;K5*
VK_ESCAPE	C / TV / WV	C / TV / WV			*K3*
VK_INSERT	C / TV	C / TV	C / TV	C / TV	
VK_DELETE	C / TV	C / TV	C / TV	C / TV	
VK_HOME	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_END	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_PRIOR	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_NEXT	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_LEFT	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_RIGHT	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_UP	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_DOWN	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F1	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	*K1*
F2	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F3	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F4	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F5	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F6	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F7	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F8	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F9	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F10	C / TV	C / TV	C / TV	C / TV	*K2*
F11	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F12	C / WV	C / WV	C / WV	C / WV	
0	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
1	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
2	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
3	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	

4	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
5	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
6	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
7	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
8	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
9	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
A	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
B	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
C	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
D	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
E	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
F	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
G	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
H	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
I	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
J	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
K	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
L	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
M	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
N	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
O	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
P	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
Q	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
R	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
S	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
T	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
U	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
V	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
W	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
X	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
Y	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
Z	C / TV / WV	C / TV / WV	C / TV / WV	C / TV / WV	
VK_NUMPAD0	C / TV / WV	C / TV / WV			
VK_NUMPAD1	C / TV / WV	C / TV / WV			

K1 在 TwinCAT 中有其它的联机帮助显示
K2 在 Web-可视化和 IE 浏览器中，文件菜单将成为焦点
K3 Ctrl/Esc 打开启动菜单，Shift/Ctrl/Esc 打开任务管理器
K4 目标可视化中制表符（Tab）和空格（Space）键有不同的功能
K5 换档（Shift）/空格（Space） 打开 JAVA 配置小程序对话框

第十三章 CX1000 CE.NET HMI

13.1 概述

从 TwinCAT 2.10 Build 1240 以上软件版本开始，全面支持 CX1000 CE.NET 的 HMI 功能。并且，第十二章中有关 TwinCAT HMI 的绝大多数功能都支持 CX1000 CE.NET。但是，CX1000 CE.NET 设备要正确实现功能，应具备如下条件：

- CX1000 CE.NET 应配备相应的可视化系统软件映像
- CX1000 CE.NET Target Visu 软件包（该软件需授权）
- TwinCAT 2.10 Build 1240 以上软件版本（该软件需授权）

13.2 安装 CX1000 CE.NET Target Visu 软件

选择 Beckhoff CD 上包含 CX1000 CE.NET Target Visu 软件的文件夹位置，见下图 13-2-1。

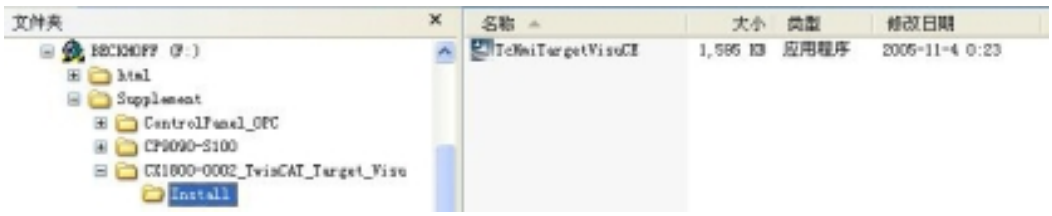


图 13-2-1 CX1800-0002_TwinCAT_Target_Visu 文件夹

运行 TcHmiTargetVisuCE.exe 程序，并按照要求输入授权号，该程序正确执行完成后将在指定的位置生成如下文件，参见图 13-2-2。

图 13-2-2 CX1800-0002_HMI_TcTargetVisuCE.I586 文件所在位置

使用以下几种方法将 CX1800-0002_HMI_TcTargetVisuCE.I586 文件传入 CX1000 CE.NET 到系统中。

- FTP Server 服务
- FLASH Card 读写器
- ActiveSync 软件

文件传送成功后，立刻在 CX1000 CE.NET 系统中双击该文件进行安装，至此，CX1000 CE.NET

系统具备 TwinCAT CE Visu 的 HMI 软件环境及要求。

13.3 TwinCAT 软件配置

TwinCAT PLC Control.ini 文件的编程环境应增加如下配置：

[TwinCAT PLC Control]
VisuHeight=768
VisuWidth=1024

可选参数：640 x 480， 800 x 600， 1024 x 768

13.4 资源任务配置

需要在 Project\Options\TwinCAT 中对 CX1000 CE.NET HMI 进行如下配置，参见图 13-4-1。

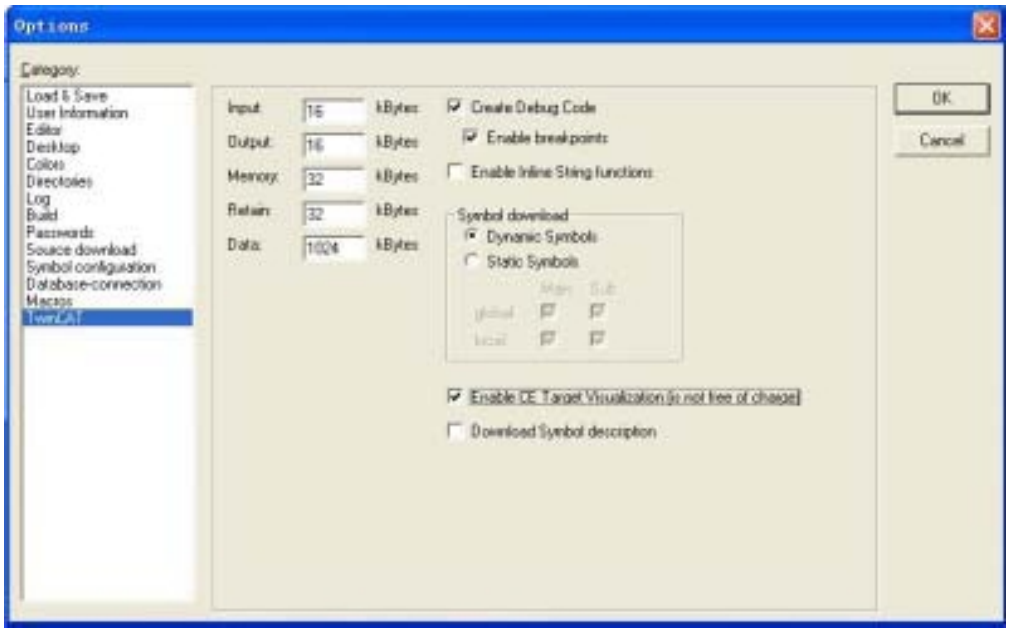


图 13-4-1 TwinCAT 选项设置

即：选中 Enable CE Target Visualization (is not free of charge) 选项，TwinCAT 系统将自动在资源的任务配置中生成两个相对应的任务，参见图 13-4-2。

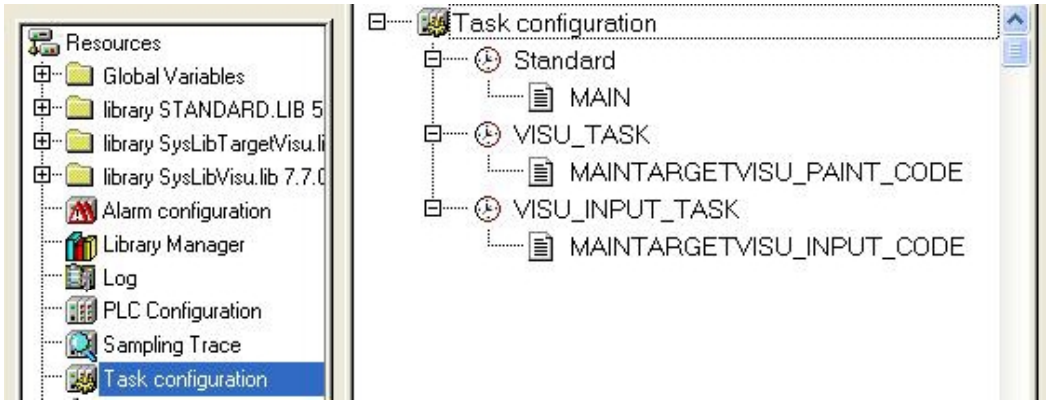


图 13-4-2 TwinCAT 系统生成的两个对应任务

至此，CX1000 CE.NET 的 HMI 开发的全部要件已经就绪，可以参照第十二章的内容进行 HMI 的组态与开发。

第十四章 使用 Visual C++ 开发 HMI

14.1 概述

TwinCAT 软件提供了 ADS-DLL、ADS-OCX 和 ADS-OPC 接口用于 HMI 软件的链接。使用 Visual C++ 开发 HMI 时，可以使用 ADS-DLL 方式。

要实现 CX1000 HMI 程序和外界设备通讯的功能，如：TwinCAT 软件，必须了解和掌握 TcAdsDll 动态库的使用方法；TcAdsDll 提供和其它 ADS 设备通讯的功能。其特点如下：

- 通过 TwinCAT 信息路由管理器和本地 TwinCAT 系统管理器或远程 TwinCAT 系统管理器通讯。
- Win32 系统 (无需安装 TwinCAT) 通过 TCP/IP 和远程的 TwinCAT 系统管理器通讯。

TcAdsDll 提供 TwinCAT ADS 客户端功能。这些功能可通过两种方法实现：

- C 语言中调用 API 函数
- 调用 COM 接口

TcAdsDll 提供和其它 ADS 设备通讯的功能，并通过 TwinCAT 路由器和 C 语言 API 函数实现。可以在 TwinCAT ADS 中找到与 ADS 相关的详细信息。

所有的示例都用 Visual C++ 进行说明。并且他们都不复杂；因此，不需要特别高级的 C/C++ 知识。

在 Visual C++ 中进行链接和程序开发时所需要的文件：

- TcAdsDll.dll – 动态链接库
- TcAdsDll.lib – 使用 TcAdsDll.dll 的函数库
- TcAdsApi.h – 声明 ADS 功能的头文件
- TcAdsDef.h – 声明结构和常数的头文件

TcAdsDll.dll 存放在 Windows NT/2000/XP 'System32' 目录下。TcAdsDll.lib, TcAdsApi.h 和 TcAdsDef.h 存放在 TwinCAT 'Ads_API\TcAdsDll\Include' 目录下。

包含头文件

要在项目中使用 TcAdsDll 功能，必须在项目中包含 TcAdsApi.h 和 TcAdsDef.h 头文件。示例如下：

```
#include "C:\TwinCAT\ADS Api\TcAdsDll\Include\TcAdsApi.h"
#include "C:\TwinCAT\ADS Api\TcAdsDll\Include\TcAdsDef.h"
```

在项目中添加库

要使用 TcAdsDll 功能，必须在项目中包含 TcAdsDll.lib。在 Visual Studio 开发软件包中选择菜单项：项目 | 设置。在项目设置对话框中选择设置范围：所有组态。为包含该软件库，必须在对象/库模块文本框中添加 TcAdsDll.Lib 文件的路径。

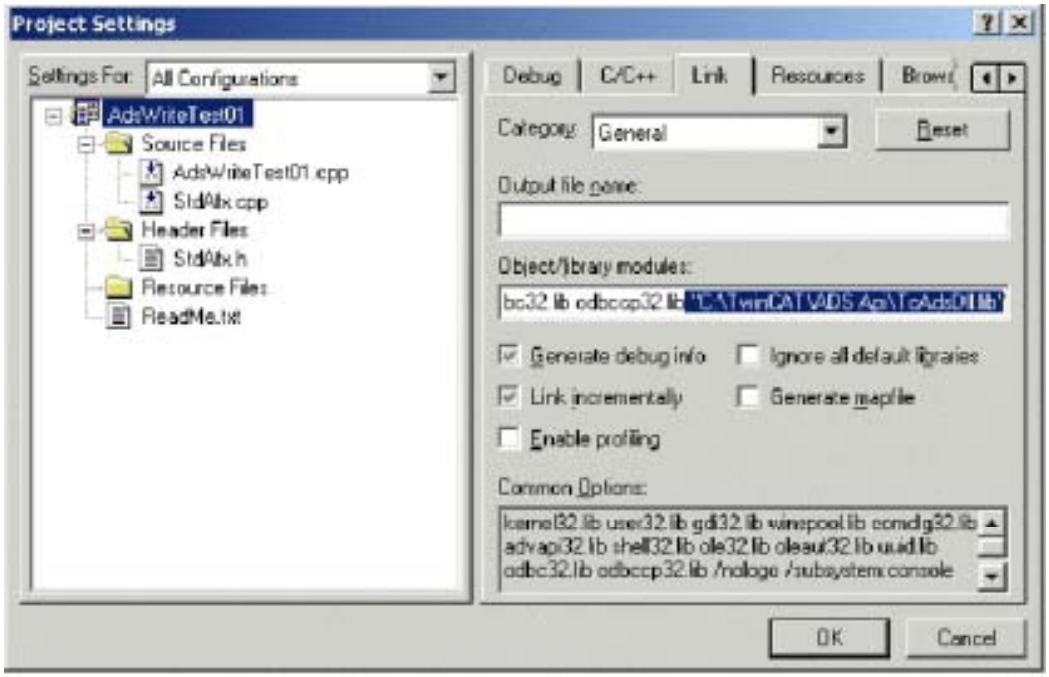


图 14-1-1 Visual C++ 环境设置

ADS 返回代码

16 进制值	10 进制值	说明
0x000	0	无错误
0x001	1	内部错误
0x002	2	无实时运行核
0x003	3	分配锁存内存错误
0x004	4	插入邮箱错误
0x005	5	接收 HMSG 错误
0x006	6	目标端口未找到
0x007	7	目标设备未找到
0x008	8	未知命令 ID
0x009	9	非法的任务 ID
0x00A	10	无 IO
0x00B	11	未知的 AMS 命令
0x00C	12	Win32 错误
0x00D	13	端口未连接
0x00E	14	无效 AMS 长度
0x00F	15	无效 AMS 网络标识
0x010	16	较低的安装等级
0x011	17	无调试信息
0x012	18	端口禁用
0x013	19	端口已连接
0x014	20	Win32 AMS 同步错

0x015	21	AMS 同步超时
0x016	22	AMS 同步 AMS 错误
0x017	23	AMS 同步无索引映像
0x018	24	无效 AMS 端口
0x019	25	无内存
0x01A	26	TCP 发送错误
0x01B	27	不能访问主机
0x500	1280	路由器：无锁定内存
0x502	1282	路由器：邮箱满
0x700	1792	错误等级 <设备错误>
0x701	1793	服务器不支持该服务
0x702	1794	无效索引组
0x703	1795	无效索引偏移
0x704	1796	读/写不允许
0x705	1797	参数长度不正确
0x706	1798	无效参数值
0x707	1799	设备不在就绪状态
0x708	1800	设备忙
0x709	1801	无效上下文(必须在 Window 中)
0x70A	1802	内存溢出
0x70B	1803	无效参数值
0x70C	1804	未找到(文件，…)
0x70D	1805	命令或文件语法错误
0x70E	1806	对象不匹配
0x70F	1807	对象已经存在
0x710	1808	符号未找到
0x711	1809	符号版本无效
0x712	1810	服务器为无效状态
0x713	1811	AdsTransMode 不支持
0x714	1812	通知句柄无效
0x715	1813	通知客户端未注册
0x716	1814	没有更多的通知句柄
0x717	1815	监视的容量太大
0x718	1816	设备未初始化
0x719	1817	设备超时
0x740	1856	错误等级 <客户端错误>
0x741	1857	无效服务参数
0x742	1858	轮循表为空
0x743	1859	变量连接已被使用
0x744	1860	调用 ID 已被使用

0x745	1861	超时时间到
0x746	1862	Win32 子系统错误
0x748	1864	ADS 端口未打开
0x750	1872	内部 ADS 同步错误
0x751	1873	HASH 表溢出
0x752	1874	HASH 中未找到键值
0x753	1875	内存中无更多符号

14.2 ADS-DLL API 功能

14.2.1 AdsPortOpen

建立 TwinCAT 信息路由器连接 (通讯口)。

```
LONG AdsPortOpen( void );
```

参数

-

返回值

返回值为 ADS 路由器分配的端口号。

示例

参见示例 2。

14.2.2 AdsPortClose

关闭 TwinCAT 信息路由器的连接(通讯口)。

```
LONG AdsPortClose( void );
```

参数

-

返回值

返回值为功能错误状态。

示例

参见示例 2。

14.2.3 AdsGetLocalAddress

返回本地 NetId 和端口号

```
LONG AdsGetLocalAddress( PAmsAddr pAddr );
```

参数

pAddr

[输出] AmsAddr 结构类型指针。

返回值

返回值为功能错误状态。

示例

参见示例 2。

14.2.4 AdsSyncWriteReq

同步写数据到一个 ADS 设备

```
LONG AdsSyncWriteReq(
PAmsAddr pAddr,
ULONG nIndexGroup,
ULONG nIndexOffset,
ULONG nLength,
PVOID pData );
```

参数
pAddr
[输入] ADS 服务器带 NetId 和端口号的结构。
nIndexGroup
[输入] 索引组
nIndexOffset
[输入] 索引偏移
nLength
[输入] 写入 ADS 服务器的数据长度，用字节表示。
pData
[输入] 写入 ADS 服务器数据的指针。
返回值
返回值为功能错误状态。
示例
参见示例 2。

14.2.5 AdsSyncReadReq

从 ADS 服务器中同步读数据。

```
LONG AdsSyncReadReq(
PAmsAddr pAddr,
ULONG nIndexGroup,
ULONG nIndexOffset,
ULONG nLength,
PVOID pData );
```

参数
pAddr
[输入] ADS 服务器带 NetId 和端口号的结构。
nIndexGroup
[输入] 索引组
nIndexOffset
[输入] 索引偏移
nLength
[输入] 数据长度，用字节表示。

pData
[输出] 接收数据的数据缓冲器指针。
返回值
返回值为功能错误状态。
示例
参见示例 3。

14.2.6 AdsSyncReadWriteReq

同步写入数据到 ADS 服务器中并从 ADS 设备接收返回的数据。

```
LONG AdsSyncReadWriteReq(
PAmsAddr pAddr,
ULONG nIndexGroup,
ULONG nIndexOffset,
ULONG nReadLength,
PVOID pReadData,
ULONG nWriteLength,
PVOID pWriteData );
```

参数
pAddr
[输入] ADS 服务器带 NetId 和端口号的结构。
nIndexGroup
[输入] 索引组
nIndexOffset
[输入] 索引偏移
nReadLength
[输入] 数据长度，用字节表示，由 ADS 设备返回。
pReadData
[输出] 由 ADS 设备返回的数据缓冲器。
nWriteLength
[输入] 数据长度，用字节表示，写入到 ADS 设备中。
pWriteData
[输出] 写入到 ADS 设备的数据缓冲器。
返回值
返回值为功能错误状态。
示例
参见示例 7。

14.2.7 AdsSyncReadDeviceInfoReq

读 ADS 服务器的标识和版本号。

```
LONG AdsSyncReadDeviceInfoReq(
PAmsAddr pAddr,
PCHAR pDevName,
```

```
PAdsVersion pVersion );
```

参数
pAddr
[输入] ADS 服务器带 NetId 和端口号的结构。
pDevName
[输出] 接收 ADS 设备名的字符串指针。
pVersion
[输出] AdsVersion 类型变量的地址，即接收版本号、再版号和创建版本号。
返回值
返回值为功能错误状态。
示例
参见示例 5。

14.2.8 AdsSyncWriteControlReq

改变 ADS 服务器的 ADS 状态和设备状态。

```
LONG AdsSyncWriteControlReq(
PAmsAddr pAddr,
USHORT nAdsState,
USHORT nDeviceState,
ULONG nLength,
PVOID pData
);
```

参数
pAddr
[输入] ADS 服务器带 NetId 和端口号的结构。
nAdsState
[输入] 新的 ADS 状态。
nDeviceState
[输入] 新的设备状态。
nLength
[输入] 用字节表示的数据长度。
pData
[输入] 发送到另外一个 ADS 设备的数据指针。
返回值
返回值为功能错误状态。
注：为改变 ADS 状态和设备状态，可以发送数据到 ADS 服务器和传送其它信息。对当前的 ADS 设备 (PLC, NC, ...),该数据无效果。

任何 ADS 设备都可以通知另一个 ADS 设备的当前状态。这里给出了设备本身的状态 (DeviceState) 和 ADS 设备 (AdsState) 的 ADS 接口状态之间的区别。这里采用的 ADS 接口状态来自 ADS 规范。
示例
参见示例 6。

14.2.9 AdsSyncReadStateReq

从 ADS 服务器中读 ADS 状态和设备状态。

```
LONG AdsSyncReadStateReq(
PAmsAddr pAddr,
USHORT *pAdsState,
PUSHORT pDeviceState );
```

参数

pAddr

[输入] ADS 服务器带 NetId 和端口号的结构。

pAdsState

[输出] 接收 ADS 状态的变量地址 (参见数据类型 ADSSTATE)。

pDeviceState

[输出] 接收设备状态的变量地址

返回值

返回值为功能错误状态。

注：任何 ADS 设备都可以通知另一个 ADS 设备的当前状态。这里给出了设备本身的状态 (DeviceState) 和 ADS 设备 (AdsState) 的 ADS 接口状态之间的区别。这里采用的 ADS 接口状态来自 ADS 规范。

示例 11 说明了如何用回调函数监测状态改变。

示例

参见示例 4。

14.2.10 AdsSyncAddDeviceNotificationReq

通知在 ADS 服务器中进行定义 (如 PLC)。当某个事件发生时，该功能 (回调功能) 在 ADS 客户端进行调用 (C 程序)。

```
LONG AdsSyncAddDeviceNotificationReq(
PAmsAddr pAddr,
ULONG nIndexGroup,
ULONG nIndexOffset,
PAdsNotificationAttrib pNoteAttrib,
PAdsNotificationFunc pNoteFunc,
ULONG hUser,
PULONG pNotification );
```

参数

pAddr

[输入] ADS 服务器带 NetId 和端口号的结构。

nIndexGroup

[输入] 索引组

nIndexOffset

[输入] 索引偏移

pNoteAttrib

[输入] 包含其它信息的结构指针。

pNoteFunc
[输入] 回调函数名。
hUser
[输入] 传递给回调函数的 32-位值。
pNotification
[输出] 接收通知的句柄变量地址。
返回值
返回值为功能错误状态。
Remarks
如果 TwinCAT 路由器被停止并重新启动，该通知将变为无效。可以使用 AdsAmsRegisterRouterNotification() 函数捕捉该事件。
示例
参见示例 8。

14.2.11 AdsSyncDelDeviceNotificationReq

从 ADS 服务器中删除以前定义的通知。

```
LONG AdsSyncDelDeviceNotificationReq(
PAmsAddr pAddr,
ULONG hNotification );
```

参数
pAddr
[输入] ADS 服务器带 NetId 和端口号的结构。
hNotification
[输出] 包含通知句柄变量的地址。
返回值
返回值为功能错误状态。
示例
参见示例 8。

14.2.12 AdsSyncSetTimeout

设置 ADS 功能超时的变量。标准值为 5000 ms。

```
LONG AdsSyncSetTimeout( LONG nMs );
```

参数
nMs
[输入] 用毫秒表示的超时时间。
返回值
返回值为功能错误状态。
示例
-

14.3 ADS-DLL API 数据结构

14.3.1 AmsAddr

完整的 ADS 设备地址可以存储在该结构中。

```
typedef struct {
AmsNetId netId;
USHORT port;
} AmsAddr, *PAmsAddr;
```

成员
NetId
网络标识
port
端口号。

14.3.2 AmsNetId

ADS 设备的网络标识可用该结构表示。

```
typedef struct {
UCHAR b[6];
} AmsNetId, *PAmsNetId;
```

成员
b[6]
网络标识，由6个数字组成。
注
该结构由 UCHAR 类型的 6 个元素数组组成。数组中的每个元素取值范围为 1 到 255。网络标识使用 TwinCAT 系统服务进行设置。

14.3 3 AdsVersion

该结构包含版本号、再版号和创建版本号。

```
typedef struct {
UCHAR version;
UCHAR revision;
USHORT build;
} AdsVersion, *PAdsVersion;
```

成员
version
版本号
revision
再版号
build
创建版本号

14.3.4 AdsNotificationAttrib

该结构包括所定义的通知所有属性。

```
typedef struct {
ULONG cbLength;
ADSTRANSMODE nTransMode;
ULONG nMaxDelay;
ULONG nCycleTime;
} AdsNotificationAttrib, *PAdsNotificationAttrib;
```

成员

cbLength

传送到回调函数的数据长度。

nTransMode

ADSTRANS_SERVERCYCLE: 该通知的回调函数被周期调用。

ADSTRANS_SERVERONCHA: 该通知的回调函数仅当数据变化时被调用。

nMaxDelay

该通知的回调函数至少在经过该延时时间后被调用。其单位为 100 ns。

nCycleTime

ADS 服务器在经过该时间间隔后再检查变量是否变化。其单位为 100 ns。

注解

ADS DLL 通过 FIFO 缓冲完成实时传送。TwinCAT 首先将每个要传送的值通过回调函数写入 FIFO 中。如果缓冲器满，或 nMaxDelay 延时时间到，则回调函数将被每个入口点调用。参数 nTransMode 将按如下方式影响其过程：

ADSTRANS_SERVERCYCLE

变量值在 nCycleTime 时间周期中被写入到 FIFO 中。nCycleTime 可取的最小值是 ADS 服务器的周期时间；对PLC而言，它是任务循环时间。 循环时间可以在 1ms 的时间内进行处理。如果输入的循环时间值为 0 ms，则变量值将在每个周期中写入 FIFO 中。

ADSTRANS_SERVERONCHA

变量值仅在其发生变化时才写入到 FIFO 中。实时采样在给定的 nCycleTime 时间之内执行。循环时间可以在 1ms 的时间内进行处理。如果输入的循环时间值为 0 ms，则变量值将在每次变量发生变化时写入 FIFO 中。

警告：如果读操作过多，将加重系统负荷；使用户接口降低速度。

提示：将循环时间设为最合适的值，并切断那些不再需要的连接。

14.3.5 AdsNotificationHeader

该结构用于回调函数。

```
typedef struct {
ULONG hNotification;
__int64 nTimeStamp;
ULONG cbSampleSize;
UCHAR data[ANYSIZE_ARRAY];
} AdsNotificationHeader, *PAdsNotificationHeader;
```

成员

hNotification

通知的句柄。当通知被定义时指定；

nTimeStamp
FILETIME 格式的时间值
cbSampleSize
传送数据的字节数。
data[ANY_SIZE_ARRAY]
传送数据的数组。
注解
时间值变换为 FILETIME 格式。FILETIME 是 64-位变量，所表示的时间和日期用100 ns 为单位的值，其起始值为 01.01.1601。不考虑本地时间的偏移；并使用一致的通用时间(UTC)。如果要存取各个成员值 (日, 月, 年, 时, 分, 秒)，则必须将该时间值从 FILETIME 格式变换为 SYSTEMTIME 格式，然后再计算时间，并将本地的时间偏移考虑进去。
示例
参见示例 8。

14.4 枚举类型数据

14.4.1 ADSSTATE

```
typedef enum nAdsState {
ADSSTATE_INVALID = 0,
ADSSTATE_IDLE = 1,
ADSSTATE_RESET = 2,
ADSSTATE_INIT = 3,
ADSSTATE_START = 4,
ADSSTATE_RUN = 5,
ADSSTATE_STOP = 6,
ADSSTATE_SAVECFG = 7,
ADSSTATE_LOADCFG = 8,
ADSSTATE_POWERFAILURE = 9,
ADSSTATE_POWERGOOD = 10,
ADSSTATE_ERROR = 11,
ADSSTATE_MAXSTATES
} ADSSTATE;
```

14.4.2 ADSTRANSMODE

```
typedef enum nAdsTransMode {
ADSTRANS_NOTRANS = 0,
ADSTRANS_CLIENTCYCLE = 1,
ADSTRANS_CLIENT1REQ = 2,
ADSTRANS_SERVERCYCLE = 3,
ADSTRANS_SERVERONCHA = 4
} ADSTRANSMODE;
```

14.5 PLC 中有关基地址的分配表

PLC 地址类型	基地址	变量类型
MD0	16#4020	DWORD
ID0	16#F020	DWORD
QD0	16#F030	DWORD
MW0	16#4020	WORD
IW0	16#F020	WORD
QW0	16#F030	WORD
MB0	16#4020	BYTE
IB0	16#F020	BYTE
QB0	16#F030	BYTE
MX0.0	16#4021	BOOL
IX0.0	16#F021	BOOL
QX0.0	16#F031	BOOL

14.6 应用示例

14.6.1 概述

说明	执行文件的源代码
示例 1: 读取 DLL 版本	ADS-DLL Sample01.exe
示例 2: 同步写入标志到 PLC 中	ADS-DLL Sample02.exe
示例 3: 从 PLC 中同步读标志	ADS-DLL Sample03.exe
示例 4: 读 ADS 状态	ADS-DLL Sample04.exe
示例 5: 读 ADS 信息	ADS-DLL Sample05.exe
示例 6: PLC 启动/停止控制	ADS-DLL Sample06.exe
示例 7: 存取 PLC 中的数组	ADS-DLL Sample07.exe
示例 8: 事件驱动方式读数据	ADS-DLL Sample08.exe

14.6.2 示例 1：读 DLL 版本

该示例完成读 DLL 文件的版本。

```
#include <iostream.h>
#include <conio.h>
#include <windows.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void main()
{
long nTemp;
AdsVersion* pDLLVersion;
nTemp = AdsGetDllVersion();
pDLLVersion = (AdsVersion *)&nTemp;
cout << "Version: " << (int)pDLLVersion->version << '\n';
cout << "Revision: " << (int)pDLLVersion->revision << '\n';
cout << "Build: " << pDLLVersion->build << '\n';
cout.flush();
getch();
}
```

14.6.3 示例 2: 同步写入标志到 PLC 中

该示例程序，将用户输入的数据写入到标志 MD0 中。

```
#include <iostream.h>
#include <windows.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void main()
{
long nErr, nPort;
AmsAddr Addr;
PAmsAddr pAddr = &Addr;
DWORD dwData;
// 打开本地 PLC 的通讯端口 (实时运行系统1)
nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(pAddr);
if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';
pAddr->port = AMSPORT_R0_PLC_RTS1;
// 读用户将写入 PLC 中的数据
cout << "Value: ";
cin >> dwData;
// 写入数据到 MD0 中
nErr = AdsSyncWriteReq( pAddr, 0x4020, 0x0, 0x4, &dwData );
if (nErr) cerr << "Error: AdsSyncWriteReq: " << nErr << '\n';
// 关闭通讯端口
nErr = AdsPortClose();
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';
}
```

14.6.4 示例 3: 从 PLC 中同步读标志

本示例程序读 PLC 中的标志 MD0 ，并将其显示在屏幕上。

```
#include <iostream.h>
#include <windows.h>
#include <conio.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void main()
{
long nErr, nPort;
AmsAddr Addr;
PAmsAddr pAddr = &Addr;
DWORD dwData;
//打开本地 PLC 的通讯端口 (实时运行系统1)
nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(pAddr);
if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';
pAddr->port = AMSPORT_R0_PLC_RTS1;
// 读数据 MD0 并进行显示
do
{
nErr = AdsSyncReadReq(pAddr, 0x4020, 0x0, 0x4, &dwData);
if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';
cout << dwData << '\n';
cout.flush();
}
while (getch() == '\r'); // 按回车键结束程序
//关闭通讯端口
nErr = AdsPortClose();
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';
}
```

14.6.5 示例 4：读 ADS 状态

本程序读 PLC 的状态。变量类型 ADSSTATE 中包含有：PLC 是运行状态、还是停止状态等信息。

```
#include <iostream.h>
#include <windows.h>
#include <conio.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void main()
{
ADSSTATE nAdsState;
USHORT nDeviceState;
long nErr, nPort;
AmsAddr Addr;
PAmsAddr pAddr = &Addr;
//打开本地 PLC 的通讯端口 (实时运行系统1)
nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(pAddr);
if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';
pAddr->port = AMSPORT_R0_PLC_RTS1;
do
{
nErr = AdsSyncReadStateReq(pAddr, &nAdsState, &nDeviceState);
if (nErr)
cerr << "Error: AdsSyncReadStateReq: " << nErr << '\n';
else
{
cout << "AdsState: " << nAdsState << '\n';
cout << "DeviceState: " << nDeviceState << '\n';
}
cout.flush();
}
while ( getch() == '\r'); // 按回车键继续，其余键结束。
//关闭通讯端口
nErr = AdsPortClose();
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';
}
```

14.6.6 示例 5: 读 ADS 信息

每个 ADS 设备都包含一个版本号和识别号。本示例程序从 PLC 中读出这些信息，并将其显示在屏幕上。

```
#include <iostream.h>
#include <windows.h>
#include <conio.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void main()
{
LONG nErr, nPort;
AdsVersion Version;
AdsVersion *pVersion = &Version;
char pDevName[50];
AmsAddr Addr;
PAmsAddr pAddr = &Addr;
//打开本地 PLC 的通讯端口 (实时运行系统1)
nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(pAddr);
if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';
pAddr->port = AMSPORT_R0_PLC_RTS1;
nErr = AdsSyncReadDeviceInfoReq(pAddr, pDevName, pVersion);
if (nErr)
cerr << "Error: AdsSyncReadDeviceInfoReq: " << nErr << '\n';
else
{
cout << "Name: " << pDevName << '\n';
cout << "Version: " << (int)pVersion->version << '\n';
cout << "Revision: " << (int)pVersion->revision << '\n';
cout << "Build: " << pVersion->build << '\n';
}
cout.flush();
getch();
//关闭通讯端口
nErr = AdsPortClose();
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';
}
```

14.6.7 示例 6: PLC 启动/停止控制

本程序控制 PLC 中运行系统 1 的启动和停止。

```
#include <iostream.h>
#include <windows.h>
#include <conio.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void main()
{
USHORT nAdsState;
USHORT nDeviceState = 0;
long nErr, nPort;
int ch;
void *pData = NULL;
AmsAddr Addr;
PAmsAddr pAddr = &Addr;
//打开本地 PLC 的通讯端口 (实时运行系统1)
nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(pAddr);
if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';
pAddr->port = AMSPORT_R0_PLC_RTS1;
cout << "(R) -> PLC Run\n";
cout << "(S) -> PLC Stop\n";
cout.flush();
ch = getch();
ch = toupper(ch);
while ( (ch == 'R') || (ch == 'S') )
{
switch (ch)
{
case 'R':
nAdsState = ADSSTATE_RUN;
break;
case 'S':
nAdsState = ADSSTATE_STOP;
break;
}
nErr = AdsSyncWriteControlReq (pAddr, nAdsState, nDeviceState, 0, pData);
if (nErr) cerr << "Error: AdsSyncWriteControlReq: " << nErr << '\n';
ch = getch();
ch = toupper(ch);
}
//关闭通讯端口
```

```
nErr = AdsPortClose();
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';
}
```

14.6.8 示例 7: 存取 PLC 中的数组

PLC 中的数组可通过读命令将其数据读出。此时，变量地址是以变量名进行编址的。该过程与读离散型变量稍有不同。数组的总长度提供给功能函数 AdsSyncReadReq()。数组中的第一个元素作为该变量的地址。

```
#include <iostream.h>
#include <windows.h>
#include <conio.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void main()
{
long nErr, nPort;
AmsAddr Addr;
PAmsAddr pAddr = &Addr;
unsigned long lHdlVar;
int nIndex;
short Data[10];
char szVar []={"MAIN.PLCVar"};
//打开本地 PLC 的通讯端口 (实时运行系统1)
nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(pAddr);
if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';
pAddr->port = AMSPORT_R0_PLC_RTS1;
//获取 PLC 变量的句柄
nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP_SYM_HNDBYNAME, 0x0, sizeof(lHdlVar),
&lHdlVar, sizeof(szVar), szVar);
if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n';
//读 PLC 变量的数据值 (通过句柄)
nErr = AdsSyncReadReq(pAddr, ADSIGRP_SYM_VALBYHND, lHdlVar, sizeof(Data),
&Data[0]);
if (nErr)
cerr << "Error: AdsSyncReadReq: " << nErr << '\n';
else
{
for (nIndex = 0; nIndex < 10; nIndex++)
cout << "Data[" << nIndex << "]: " << Data[nIndex] << '\n';
}
cout.flush();
getch();
```

```
//关闭通讯端口
nErr = AdsPortClose();
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';
}
```

14.6.9 示例 8: 事件驱动方式读数据

如果想要 PLC 或 NC 中的数据在用户接口中进行连续显示，使用 AdsSyncReadReq() 方法会感觉到很不方便，因为要周期性的调用该功能。通过使用我们所知的通知(消息)进行定义，TwinCAT 服务器可以将数据从一个 ADS 传送到另一个 ADS 设备中。该过程中有两种方式，即 TwinCAT 服务器是采用周期性地传送数据、还是当数据发生变化时再传送数据。通知是从调用功能函数 AdsSyncAddDeviceNotificationReq() 开始。之后，其回调函数自动由 TwinCAT 进行调用。

功能函数 AdsSyncDelDeviceNotificationReq() 用于停止该通知。由于通知的数量有限，应尽量删除那些程序不再使用的通知。可以在 AdsNotificationAttrib 的数据结构中查阅更详细的信息。本示例程序在 PLC 的标志 MD0 中启动了一个通知消息。当 PLC 的变量每次变化时，都会调用回调函数。回调函数接收到一个类型为 AdsNotificationHeader() 的参数变量。该结构中包括所需的全部信息(数据值、时间值等)。

```
#include <iostream.h>
#include <windows.h>
#include <conio.h>
#include <winbase.h>
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
void Callback(AmsAddr*, AdsNotificationHeader*, unsigned long);
void main()
{
long nErr, nPort;
AmsAddr Addr;
PAmsAddr pAddr = &Addr;
ULONG hNotification, hUser;
AdsNotificationAttrib adsNotificationAttrib;
//打开本地 PLC 的通讯端口 (实时运行系统1)
nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(pAddr);
if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';
pAddr->port = AMSPORT_R0_PLC_RTS1;
//指定通知的属性
adsNotificationAttrib.cbLength = 4;
adsNotificationAttrib.nTransMode = ADSTRANS_SERVERONCHA;
adsNotificationAttrib.nMaxDelay = 20000000; // 2 秒
adsNotificationAttrib.nCycleTime = 10000000; // 1 秒
// 传递给回调函数的 32-位变量值 (包括指针)
```

```
hUser = 3474573467;
//开始传送 PLC 变量
nErr = AdsSyncAddDeviceNotificationReq(pAddr, 0x4020, 0, &adsNotificationAttrib,
Callback, hUser, &hNotification);
if (nErr) cerr << "Error: AdsSyncAddDeviceNotificationReq: " << nErr << '\n';
cout << "Notification: " << hNotification << "\n\n";
cout.flush();
// 等待用户按压任意一键
getch();
//完成传送 PLC 变量
nErr = AdsSyncDelDeviceNotificationReq(pAddr, hNotification);
if (nErr) cerr << "Error: AdsSyncDelDeviceNotificationReq: " << nErr << '\n';
//关闭通讯端口
nErr = AdsPortClose();
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';
}

//回调函数
void Callback(AmsAddr* pAddr, AdsNotificationHeader* pNotification, ULONG hUser)
{
int nIndex;
static ULONG nCount = 0;
SYSTEMTIME SystemTime, LocalTime;
FILETIME FileTime;
LARGE_INTEGER LargeInteger;
TIME_ZONE_INFORMATION TimeZoneInformation;
cout << ++nCount << ". Call:\n";
//变量值输出
cout << "Value: " << *(ULONG *)pNotification->data << '\n';
cout << "Notification: " << pNotification->hNotification << '\n';
//将时间值转换为 SYSTEMTIME 格式
LargeInteger.QuadPart = pNotification->nTimeStamp;
FileTime.dwLowDateTime = (DWORD)LargeInteger.LowPart;
FileTime.dwHighDateTime = (DWORD)LargeInteger.HighPart;
FileTimeToSystemTime(&FileTime, &SystemTime);
//计算本地格式表示的时间值
GetTimeZoneInformation(&TimeZoneInformation);
SystemTimeToTzSpecificLocalTime(&TimeZoneInformation, &SystemTime, &LocalTime);
//输出时间值
cout << LocalTime.wHour << ":" << LocalTime.wMinute << ":" << LocalTime.wSecond <<
'.' << LocalTime.wMilliseconds <<
" den: " << LocalTime.wDay << '.' << LocalTime.wMonth << '.' << Local-
Time.wYear << '\n';
//用字节表示的缓冲区
```

```
cout << "SampleSize: " << pNotification->cbSampleSize << '\n';
// 用 AddNotification 表示的 32-位变量(包括指针) // (参见 main)
cout << "hUser: " << hUser << '\n';
//输出发送端的 ADS 地址
cout << "ServerNetId: ";
for (nIndex = 0; nIndex < 6; nIndex++)
cout << (int)pAddr->netId.b[nIndex] << ".";
cout << "\nPort: " << pAddr->port << "\n\n";
cout.flush();
}
```

14.7 Visual C++ 开发实例

14.7.1 ModbusTCP 示例

ModbusTCP 协议实现了在以太网 TCP/IP 中完成 Modbus 协议的功能，其基本要素包括 IP 地址和 ModbusTCP 的访问端口 502。本示例实现的是 ModbusTCP 客户端功能。

14.7.1.1 CMySocket 派生类

本示例从 CSocket 类派生出 CMySocket 类完成该功能，包括一个 200 字节的接收缓冲区成员变量和接收信息消息的虚拟函数，示例运行界面参见图 14-7-1。

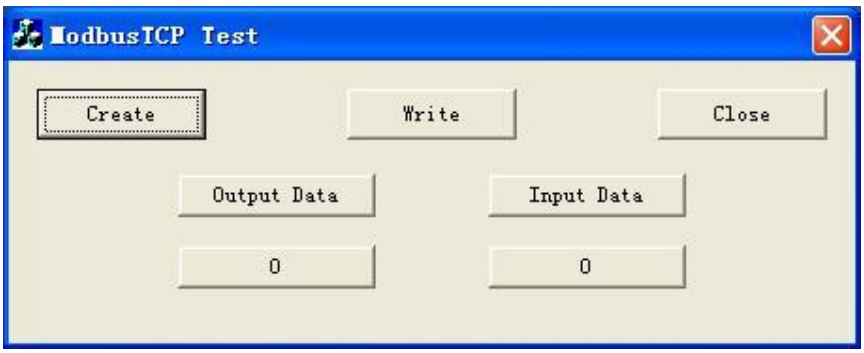


图 14-7-1：ModbusTCP 演示程序示例界面图。

相关代码如下：

```
class CMySocket : public CSocket
{
// Attributes
public:
    unsigned char    m_RecBuffer[200];
// Operations
public:
    CMySocket();
    virtual ~CMySocket();

// Overrides
public:
```

```
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMySocket)
    public:
    virtual void OnReceive(int nErrorCode);
    //}}AFX_VIRTUAL

    // Generated message map functions
    //{{AFX_MSG(CMySocket)
    // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG

// Implementation
protected:
};

void CMySocket::OnReceive(int nErrorCode)
{
    CSocket::OnReceive(nErrorCode);

    int Rec = Receive(m_RecBuffer,100,0);
    m_RecBuffer[Rec] = 0x0;
}
```

14.7.1.2 创建 ModbusTCP 客户端链接

```
void CModbusDlg::OnModbusCreate()
{
    if (m_pSocket != NULL)
    {
        m_pSocket->Close();
        delete m_pSocket;
        m_pSocket = NULL;
    }

    m_pSocket    = new CMySocket();

    if (!m_pSocket->Create())
    {
        delete m_pSocket;
        m_pSocket = NULL;
        AfxMessageBox("Create the Socket failed!");
        return;
    }

    if (!m_pSocket->Connect(m_IPAddress,502))
    {
```

```
            delete m_pSocket;
            m_pSocket = NULL;
            AfxMessageBox("Connect the Socket failed!");
            return;
        }

        m_ConnectOK = true;
        ResetWD();
        SetTimer(1,200,NULL);
//      AfxMessageBox("Connection established!");
}
```

14.7.1.3 发送读 ModbusTCP 命令

```
void CModbusDlg::OnModbusRead()
{
    if (m_pSocket != NULL)
    {
        m_SendBuffer[5]  = 0x06;  // message length.
        m_SendBuffer[6]  = 0x01;  // unit ID
        m_SendBuffer[7]  = 0x04;  // function code
        m_SendBuffer[8]  = 0x00;  // start addr. high
        m_SendBuffer[9]  = 0x00;  // start addr. low    0x0000 %IB128
        m_SendBuffer[10] = 0x00;  // data length high
        m_SendBuffer[11] = 0x01;  // data length low   (total 16 data bytes)

        m_pSocket->Send(&m_SendBuffer[0],12,0);
    }
}
```

14.7.1.4 发送写 ModbusTCP 命令

```
void CModbusDlg::OnModbusWrite()
{
    if (m_pSocket != NULL && m_ConnectOK)
    {
        m_SendBuffer[5]  = 0x06;          // message length.
        m_SendBuffer[6]  = 0x01;          // unit ID
        m_SendBuffer[7]  = 0x06;          // function code
        m_SendBuffer[8]  = 0x00;          // start addr. high
        m_SendBuffer[9]  = m_offset;      // start addr. low    0x0000 %QB128
        m_SendBuffer[10] = m_value/256;   // data value high
        m_SendBuffer[11] = m_value%256;   // data value low

        m_pSocket->Send(&m_SendBuffer[0],12,0);
    }
```

```
    m_value++;
}
```

14.7.1.5 客户端接收 ModbusTCP 消息

```
void CModbusDlg::ModbusReceive()
{
    CString str;
    int i;
    if (m_pSocket != NULL)
    {
        if (m_pSocket->m_RecBuffer[7] <= 0x10)
                                    // function code 3:read output, 4:read input
        {
            for (i=0;i<m_pSocket->m_RecBuffer[8];i++)      // data length.
            {
                switch(m_pSocket->m_RecBuffer[7])
                {
                case 0x4:
                    m_InData[i] = m_pSocket->m_RecBuffer[9+i];
                    break;
                case 0x3:
                    m_OuData[i] = m_pSocket->m_RecBuffer[9+i];
                    break;
                default:
                    m_InData[i] = 0x0;
                    m_OuData[i] = 0x0;
                }
            }
            for (i=1;i<m_pSocket->m_RecBuffer[8];i+=2)    // exchange H & L byte
            {
                m_InDataSN[i-1] = m_InData[i];
                m_OuDataSN[i-1] = m_OuData[i];
            }
            for (i=0;i<m_pSocket->m_RecBuffer[8];i+=2)
            {
                m_InDataSN[i+1] = m_InData[i];
                m_OuDataSN[i+1] = m_OuData[i];
            }
        }
        else
        {
            m_SendBuffer[5]   = 0x06;   // message length.
            m_SendBuffer[6]   = 0x01;   // unit ID
            m_SendBuffer[7]   = 0x06;   // function code
            m_SendBuffer[8]   = 0x11;   // watchdog addr. high
```

```
            m_SendBuffer[9]   = 0x21;   // watchdog addr. low
            m_SendBuffer[10] = 0xBE;   // data high
            m_SendBuffer[11] = 0xCF;   // data low      1 data
                                       // first time write data: 0xBECF
            if (m_pSocket != NULL)
                m_pSocket->Send(&m_SendBuffer[0],12,0);
            Sleep(50);
            m_SendBuffer[5]   = 0x06;   // message length.
            m_SendBuffer[6]   = 0x01;   // unit ID
            m_SendBuffer[7]   = 0x06;   // function code
            m_SendBuffer[8]   = 0x11;   // watchdog addr. high
            m_SendBuffer[9]   = 0x21;   // watchdog addr. low
            m_SendBuffer[10] = 0xAF;   // data high
            m_SendBuffer[11] = 0xFE;   // data low      1 data
                                       // second time write data: 0xAFFE
            if (m_pSocket != NULL)
                m_pSocket->Send(&m_SendBuffer[0],12,0);
        }
    }
    else
    {
        for (i=0;i<16;i++)
        {
            m_InData[i] = 0x0;
            m_OuData[i] = 0x0;
        }
    }
}
```

14.7.1.6 关闭 ModbusTCP 客户端链接

```
void CModbusDlg::OnModbusClose()
{
    if (m_pSocket != NULL)
    {
        m_pSocket->Close();
        delete m_pSocket;
        m_pSocket = NULL;
    }
}
```

全部示例代码请参见 CD 光盘。

14.7.2 Modem 远程通讯示例

Modem 远程通讯 HMI 示例与 5.4.5 章节的 PLC 程序相对应，实现 PC 通过 Modem 远程控制 PLC 的 HMI 界面功能；Modem 远程通讯 HMI 示例界面运行效果图参见图 14-7-2。

14.7.2.1 建立 Modem 通讯连接命令

该命令发送 Modem AT 指令 ATD#目标电话号码，完成与远程 PLC 设备的通讯链接，相关示例代码如下：

```
void CView20::Modem_connect()
{
    for (int i=0;i<5;i++)
        m_Flag[i] = FALSE;
    m_Index  = 0;
    m_Index1 = 0;

    m_sendcmd[0] = 'A';              // AT command ATD#14
    m_sendcmd[1] = 'T';
    m_sendcmd[2] = 'D';
    m_sendcmd[3] = '#';
    m_sendcmd[4] = '1';
    m_sendcmd[5] = '4';              // Target telphone number: #14
    m_sendcmd[6] = 0x0D;             // MSG end
    m_sendcmd[7] = 0x00;             // MSG end
    m_Port.WriteToPort((char *)m_sendcmd);
}
```

14.7.2.2 写数据命令

该命令发送写数据指令，将 PC 中的数据写入到远程 PLC 设备，相关示例代码如下：

```
void CView20::Modem_writedata()
{
    char CRC;
    CRC = 0x00;
    m_sendcmd[0] = 0x03;
    m_sendcmd[1] = 0x12;
    m_sendcmd[2] = 0x57;        // write command
    m_sendcmd[3] = m_WriteIndex;
    m_sendcmd[4] = m_WriteValue % 256;
    m_sendcmd[5] = m_WriteValue / 256;
    for (int i=0;i<= 5;i++)
        CRC = CRC ^ m_sendcmd[i];
    m_sendcmd[6] = CRC;
    m_sendcmd[7] = 0x0D;
    m_sendcmd[8] = 0x00;        // MSG end

    m_writeData[m_WriteIndex-1] = m_WriteValue;
    m_Port.WriteToPort((char *)m_sendcmd);
}
```

14.7.2.3 读数据命令

该命令发送读数据指令，将远程 PLC 设备中的数据读入到 PC 中，相关示例代码如下：

```
void CView20::Modem_readdata()
{
    char CRC;
    CRC = 0x00;
    m_sendcmd[0] = 0x03;
    m_sendcmd[1] = 0x12;
    m_sendcmd[2] = 0x52;        // read command
    m_sendcmd[3] = 0x01;
    m_sendcmd[4] = 0x11;
    m_sendcmd[5] = 0x12;
    for (int i=0;i<= 5;i++)
        CRC = CRC ^ m_sendcmd[i];
    m_sendcmd[6] = CRC;
    m_sendcmd[7] = 0x0D;
    m_sendcmd[8] = 0x00;        // MSG end
    m_Port.WriteToPort((char *)m_sendcmd);
}
```

14.7.2.4 读报警状态命令

该命令发送读数据指令，将远程 PLC 设备中的报警状态读入到 PC 中，相关示例代码如下：

```
void CView20::Modem_alarmdata()
```

```
{
    char CRC;
    CRC = 0x00;
    m_sendcmd[0] = 0x03;
    m_sendcmd[1] = 0x12;
    m_sendcmd[2] = 0x54;        // read alarm message from remote PLC
    m_sendcmd[3] = 0x01;
    m_sendcmd[4] = 0x11;
    m_sendcmd[5] = 0x22;
    for (int i=0;i<= 5;i++)
        CRC = CRC ^ m_sendcmd[i];
    m_sendcmd[6] = CRC;
    m_sendcmd[7] = 0x0D;
    m_sendcmd[8] = 0x00;        // MSG end
    m_Port.WriteToPort((char *)m_sendcmd);
}
```

14.7.2.5 读报警状态命令

该命令发送读报警状态指令，将远程 PLC 设备中的报警状态读入到 PC 中，相关示例代码如下：

```
void CView20::Modem_alarmdata()
{
    char CRC;
    CRC = 0x00;
    m_sendcmd[0] = 0x03;
    m_sendcmd[1] = 0x12;
    m_sendcmd[2] = 0x54;        // read alarm message from remote PLC
    m_sendcmd[3] = 0x01;
    m_sendcmd[4] = 0x11;
    m_sendcmd[5] = 0x22;
    for (int i=0;i<= 5;i++)
        CRC = CRC ^ m_sendcmd[i];
    m_sendcmd[6] = CRC;
    m_sendcmd[7] = 0x0D;
    m_sendcmd[8] = 0x00;        // MSG end
    m_Port.WriteToPort((char *)m_sendcmd);
}
```

14.7.2.6 关闭 Modem 远程 PLC 通讯链接命令

发送 Modem AT 指令： ATH0，关闭远程 PLC Modem 通讯状态，相关示例代码如下：

```
void CView20::Modem_terminate()
{
    m_sendcmd[0] = '+';                 // AT command +++ATH0
    m_sendcmd[1] = '+';
    m_sendcmd[2] = '+';
```

```
        m_sendcmd[3] = 'A';
        m_sendcmd[4] = 'T';
        m_sendcmd[5] = 'H';
        m_sendcmd[6] = '0';
        m_sendcmd[7] = 0x0D;
        m_sendcmd[8] = 0x00;        // MSG end
        m_Port.WriteToPort((char *)m_sendcmd);
        for (int i=0;i<5;i++)
              m_Flag[i] = FALSE;
        m_Index  = 0;
        m_Index1 = 0;
}
```

全部示例代码请参见 CD 光盘。

14.7.3 使用 Visual C++ 实现 HMI 实例

本示例代码实现了一个实际的 HMI 功能，并且完成了双缓冲区的动画显示，消除了动态画面闪屏的现象。示例中未包括使用 ADS-DLL 通讯部分，用户可在此基础上自行扩展功能，完成多画面切换、数据通讯和其他功能。该示例程序运行界面效果图参见图 14-7-3 和图 14-7-4。

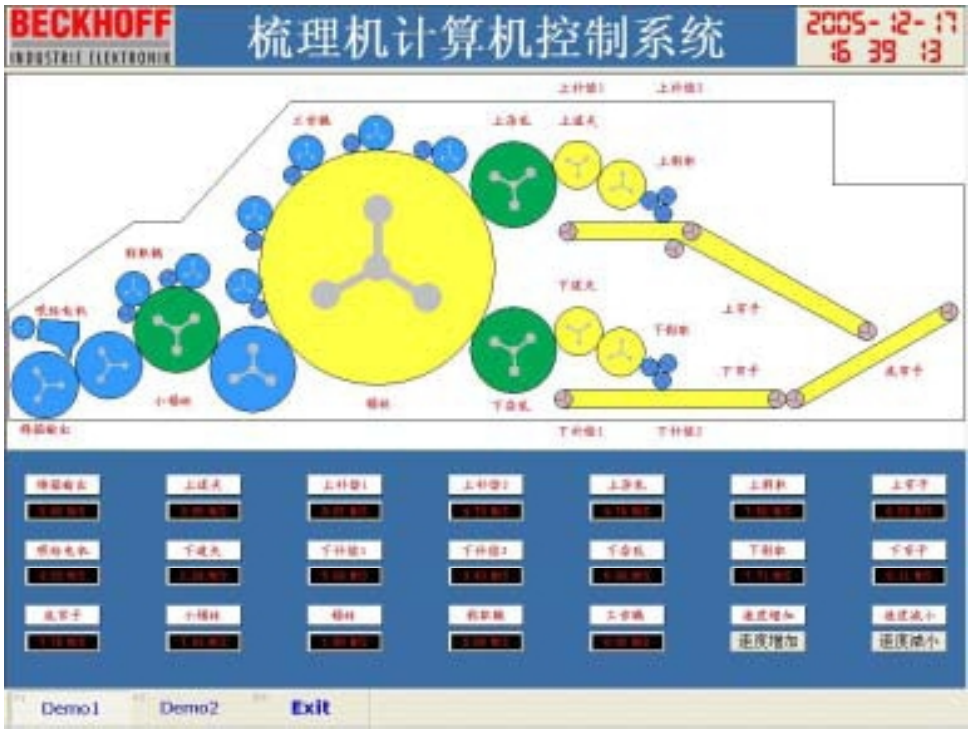


图 14-7-3 Visual C++ 实现动态 HMI 界面示例画面一

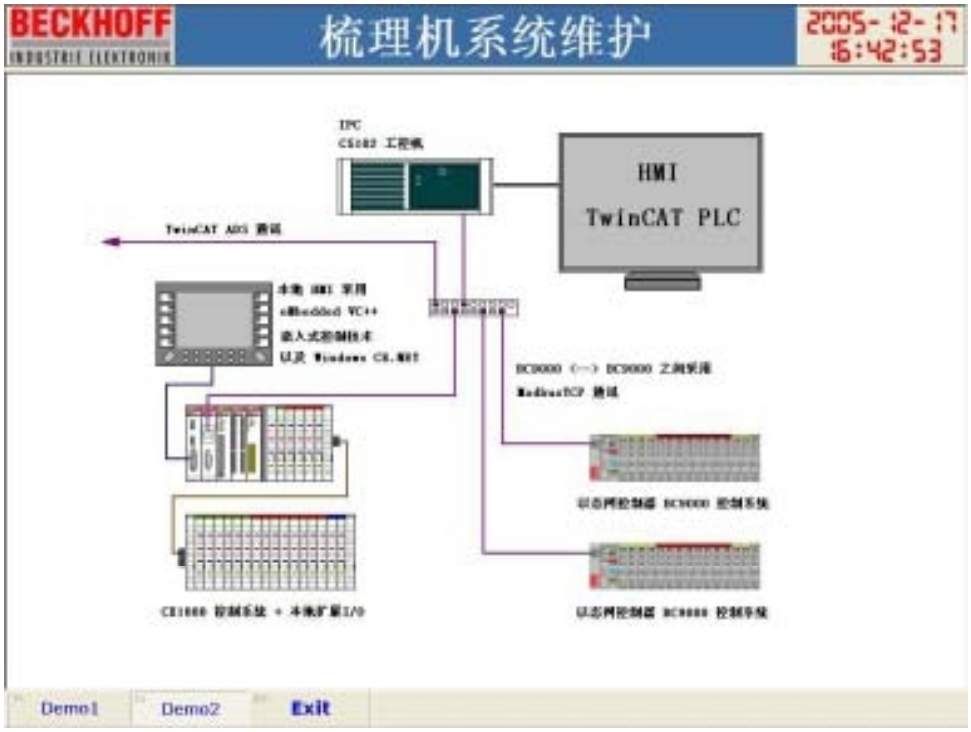


图 14-7-4 Visual C++ 实现动态 HMI 界面示例画面二

14.7.3.1 双缓冲区动态显示画面程序代码

采用面向对象程序设计方式，从 CStatic 类派生出新类 CCircle。具体代码如下：

```
void CCircle::OnPaint()
{
    CPaintDC dc(this);
    CRect rc;
    CDC MemDC,MemDC1[19];
    CBitmap cBitmap,cbmp1[19];
    CBrush br[9],*pbr;
    CPen pen[9],*ppen,*ppen1;
    POINT ptArray[7];
    int i,j;
    float angle0    = m_angle;
    float PI        = 3.1415926f;
    float temp          = 2.0f * PI;
    float alph      = (2.0f *PI) / 3.0f;
    float angleF;

    rc = m_rect;
    MemDC.CreateCompatibleDC(&dc);
    cBitmap.CreateCompatibleBitmap(&dc,rc.Width(),rc.Height());
    MemDC.SelectObject(&cBitmap);
    COLORREF GradColor = RGB(192,192,192);
    br[0].CreateSolidBrush(RGB(255,255,255));
```

```
br[1].CreateSolidBrush(RGB(53,154,255));
br[2].CreateSolidBrush(RGB(0,166,83));
br[3].CreateSolidBrush(RGB(255,255,70));
br[4].CreateSolidBrush(GradColor);
br[5].CreateSolidBrush(RGB(255,0,0));
br[6].CreateSolidBrush(RGB(64,0,0));
br[7].CreateSolidBrush(RGB(0,255,0));
br[8].CreateSolidBrush(RGB(0,64,0));

pbr = MemDC.SelectObject(&br[0]);

pen[0].CreatePen(PS_SOLID,1,m_BkColor);
pen[1].CreatePen(PS_SOLID,1,m_FgColor);
pen[2].CreatePen(PS_SOLID,1,m_LineColor);
pen[3].CreatePen(PS_SOLID,1,RGB(0,0,0));
pen[4].CreatePen(PS_SOLID,1,GradColor);
pen[5].CreatePen(PS_SOLID,2,GradColor);
pen[6].CreatePen(PS_SOLID,3,GradColor);
pen[7].CreatePen(PS_SOLID,10,GradColor);
pen[8].CreatePen(PS_SOLID,2,RGB(255,0,0));

ppen = MemDC.SelectObject(&pen[0]);

MemDC.Rectangle(&rc);

cbmp1[0].LoadBitmap(IDB_V20LABEL01);
for (i=0;i<19;i++)
      MemDC1[i].CreateCompatibleDC(&MemDC);

MemDC1[0].SelectObject(&cbmp1[0]);
// 3 - x0, 232 - y0, 53 - width, 13 - high;    11 - x1, 4 - y1
// 53, 13 为文字宽度 x 高度; 11, 4 为文字在位图中位置，保持不变
// 3, 232 为画图位置，可调整
// 3, 232 均乘 1.6 取整
MemDC.BitBlt(13,371,53,13,&MemDC1[0],11,4,SRCCOPY); // 棉箱输出

cbmp1[1].LoadBitmap(IDB_V20LABEL11);
MemDC1[1].SelectObject(&cbmp1[1]);
MemDC.BitBlt(30,245,55,13,&MemDC1[1],12,4,SRCCOPY); // 喂给电机

cbmp1[2].LoadBitmap(IDB_V20LABEL24);
MemDC1[2].SelectObject(&cbmp1[2]);
MemDC.BitBlt(125,184,41,13,&MemDC1[2],18,4,SRCCOPY);// 剥取辊
cbmp1[3].LoadBitmap(IDB_V20LABEL22);
```

```
MemDC1[3].SelectObject(&cbmp1[3]);
MemDC.BitBlt(157,341,40,13,&MemDC1[3],18,4,SRCCOPY);// 小锡林

cbmp1[4].LoadBitmap(IDB_V20LABEL25);
MemDC1[4].SelectObject(&cbmp1[4]);
MemDC.BitBlt(304,43,40,13,&MemDC1[4],17,4,SRCCOPY); // 工作辊

cbmp1[5].LoadBitmap(IDB_V20LABEL23);
MemDC1[5].SelectObject(&cbmp1[5]);
MemDC.BitBlt(381,344,27,13,&MemDC1[5],24,4,SRCCOPY);// 锡林

cbmp1[6].LoadBitmap(IDB_V20LABEL05);
MemDC1[6].SelectObject(&cbmp1[6]);
MemDC.BitBlt(516,43,41,13,&MemDC1[6],18,4,SRCCOPY); // 上杂乱

cbmp1[7].LoadBitmap(IDB_V20LABEL02);
MemDC1[7].SelectObject(&cbmp1[7]);
MemDC.BitBlt(587,43,41,13,&MemDC1[7],18,4,SRCCOPY); // 上道夫

cbmp1[8].LoadBitmap(IDB_V20LABEL03);
MemDC1[8].SelectObject(&cbmp1[8]);
MemDC.BitBlt(586,8,48,12,&MemDC1[8],15,4,SRCCOPY);   // 上补偿 1

cbmp1[9].LoadBitmap(IDB_V20LABEL04);
MemDC1[9].SelectObject(&cbmp1[9]);
MemDC.BitBlt(691,8,48,12,&MemDC1[9],14,4,SRCCOPY);   // 上补偿 2

cbmp1[10].LoadBitmap(IDB_V20LABEL06);
MemDC1[10].SelectObject(&cbmp1[10]);
MemDC.BitBlt(691,86,41,13,&MemDC1[10],18,4,SRCCOPY);// 上剥取

cbmp1[11].LoadBitmap(IDB_V20LABEL07);
MemDC1[11].SelectObject(&cbmp1[11]);
MemDC.BitBlt(760,243,41,13,&MemDC1[11],19,4,SRCCOPY);// 上帘子

cbmp1[12].LoadBitmap(IDB_V20LABEL15);
MemDC1[12].SelectObject(&cbmp1[12]);
MemDC.BitBlt(516,346,40,13,&MemDC1[12],18,4,SRCCOPY);// 下杂乱

cbmp1[13].LoadBitmap(IDB_V20LABEL12);
MemDC1[13].SelectObject(&cbmp1[13]);
MemDC.BitBlt(586,218,40,13,&MemDC1[13],18,4,SRCCOPY);// 下道夫

cbmp1[14].LoadBitmap(IDB_V20LABEL16);
```

```
	MemDC1[14].SelectObject(&cbmp1[14]);
	MemDC.BitBlt(686,266,40,13,&MemDC1[14],18,4,SRCCOPY);// 下剥取

	cbmp1[15].LoadBitmap(IDB_V20LABEL13);
	MemDC1[15].SelectObject(&cbmp1[15]);
	MemDC.BitBlt(586,373,47,13,&MemDC1[15],14,4,SRCCOPY);// 下补偿 1

	cbmp1[16].LoadBitmap(IDB_V20LABEL14);
	MemDC1[16].SelectObject(&cbmp1[16]);
	MemDC.BitBlt(691,373,47,13,&MemDC1[16],14,4,SRCCOPY);// 下补偿 2

	cbmp1[17].LoadBitmap(IDB_V20LABEL17);
	MemDC1[17].SelectObject(&cbmp1[17]);
	MemDC.BitBlt(760,309,40,13,&MemDC1[17],19,4,SRCCOPY);// 下帘子

	cbmp1[18].LoadBitmap(IDB_V20LABEL21);
	MemDC1[18].SelectObject(&cbmp1[18]);
	MemDC.BitBlt(933,309,41,13,&MemDC1[18],19,4,SRCCOPY);// 底帘子
	MemDC.SelectObject(&pen[3]);

	MemDC.MoveTo(2,368);
	MemDC.LineTo(2,251);
	MemDC.LineTo(134,158);
	MemDC.LineTo(184,158);
	MemDC.LineTo(302,29);
	MemDC.LineTo(878,29);
	MemDC.LineTo(878,118);
	MemDC.LineTo(1018,118);  //1013
	MemDC.LineTo(1018,368);
	MemDC.LineTo(2,368);
//------------------------------ 边框---------------------
	ptArray[0].x = 597;
	ptArray[0].y = 158;
	ptArray[1].x = 730;
	ptArray[1].y = 158;
	ptArray[2].x = 920;
	ptArray[2].y = 266; //267
	ptArray[3].x = 912;
	ptArray[3].y = 283; //280
	ptArray[4].x = 712;
	ptArray[4].y = 176;
	ptArray[5].x = 597;
	ptArray[5].y = 176;
	MemDC.SelectObject(&br[3]);
```

```
        MemDC.Polygon(ptArray,6);
//------------------------------- 上帘子-----------------
        ptArray[0].x = 592;
        ptArray[0].y = 336;
        ptArray[1].x = 819;
        ptArray[1].y = 336;
        ptArray[2].x = 819;
        ptArray[2].y = 355;
        ptArray[3].x = 594;
        ptArray[3].y = 355;
        MemDC.Polygon(ptArray,4);
//------------------------------- 下帘子 1 ----------------
        ptArray[0].x = 834;
        ptArray[0].y = 339;
        ptArray[1].x = 997;
        ptArray[1].y = 245;
        ptArray[2].x = 1006;
        ptArray[2].y = 261;
        ptArray[3].x = 843;
        ptArray[3].y = 354;

        MemDC.Polygon(ptArray,4);
//------------------------------- 下帘子 2 ----------------
        ptArray[0].x = 32;
        ptArray[0].y = 282;
        ptArray[1].x = 35;
        ptArray[1].y = 261;
        ptArray[2].x = 77;
        ptArray[2].y = 264;
        ptArray[3].x = 77;
        ptArray[3].y = 282;
        ptArray[4].x = 70;
        ptArray[4].y = 290;
        ptArray[5].x = 67;
        ptArray[5].y = 304;
        ptArray[6].x = 53;
        ptArray[6].y = 288;
        MemDC.SelectObject(&br[1]);
        MemDC.Polygon(ptArray,7);
//------------------------------- 棉箱输出-----------------
        for (i=0;i<40;i++)
        {
            switch (i)
            {
```

```
case 2:
case 5:
case 15:
    MemDC.SelectObject(&pen[3]);
    MemDC.SelectObject(&br[2]);
    break;
case 4:
case 6:
case 7:
case 16:
case 17:
    MemDC.SelectObject(&pen[3]);
    MemDC.SelectObject(&br[3]);
    break;
case 11:
case 12:
case 13:
case 14:
case 21:
case 22:
case 23:
case 24:
    MemDC.SelectObject(&pen[3]);
    MemDC.SelectObject(&br[4]);
    break;
default:
    MemDC.SelectObject(&pen[3]);
    MemDC.SelectObject(&br[1]);
}
MemDC.Ellipse(delx[i]-lx[i],dely[i]-lx[i],delx[i]+lx[i],dely[i]+lx[i]);
angleF    = Kangle[i] * angle0 * PI / 180.0f;

if (ArcFlag[0])
{
    ppen1 = MemDC.SelectObject(&pen[8]);
    MemDC.SelectObject(ppen1);
    ArcFlag[0] = false;
}

if (rot[i])
{
    x[i][0] = (int)((float)lx[i] * cos(angleF));
    y[i][0] = (int)((float)lx[i] * sin(angleF));
```

```
        x[i][1] = (int)((float)lx[i] * cos(alph+angleF));
        y[i][1] = (int)((float)lx[i] * sin(alph+angleF));

        x[i][2] = (int)((float)lx[i] * cos(2.0f*alph+angleF));
        y[i][2] = (int)((float)lx[i] * sin(2.0f*alph+angleF));
    }
    else
    {
        x[i][0] = (int)((float)lx[i] * cos(-angleF));
        y[i][0] = (int)((float)lx[i] * sin(-angleF));

        x[i][1] = (int)((float)lx[i] * cos(alph-angleF));
        y[i][1] = (int)((float)lx[i] * sin(alph-angleF));

        x[i][2] = (int)((float)lx[i] * cos(2.0f*alph-angleF));
        y[i][2] = (int)((float)lx[i] * sin(2.0f*alph-angleF));
    }

    for (j=0;j<3;j++)
    {
        MemDC.SelectObject(&br[4]);
        MemDC.SelectObject(&pen[4]);

        MemDC.Ellipse(delx[i]+x[i][j]/2-lr[i],dely[i]+y[i][j]/2-lr[i],
            delx[i]+x[i][j]/2+lr[i],dely[i]+y[i][j]/2+lr[i]);

        MemDC.Ellipse(delx[i]-lr[i]+1,dely[i]-lr[i]+1,
            delx[i]+lr[i]-1,dely[i]+lr[i]-1);

        if (lr[i] > 0)
        {
            switch(lr[i])
            {
            case 1:
            case 2:
                MemDC.SelectObject(&pen[4]);
                break;
            case 3:
            case 4:
                MemDC.SelectObject(&pen[5]);
                break;
            case 6:
                MemDC.SelectObject(&pen[6]);
                break;
```

```
                    case 16:
                        MemDC.SelectObject(&pen[7]);
                        break;
                    }

                    MemDC.MoveTo(delx[i],dely[i]);
                    MemDC.LineTo(delx[i] + x[i][j]/2,dely[i] + y[i][j]/2);
                }
                else
                {
                    MemDC.SelectObject(&pen[2]);
                    MemDC.SelectObject(&br[0]);

                    MemDC.MoveTo(delx[i],dely[i]);
                    MemDC.LineTo(delx[i] + x[i][j],dely[i] + y[i][j]);
                }
            }
        }
        ArcFlag[0] = true;
        dc.BitBlt(0,0,rc.Width(),rc.Height(),&MemDC,0,0,SRCCOPY);
        dc.SelectObject(ppen);
        dc.SelectObject(pbr);

        for (i=0;i<9;i++)
        {
            pen[i].DeleteObject();
            br[i].DeleteObject();
        }

        for (i=0;i<19;i++)
        {
            MemDC1[i].DeleteDC();
            cbmp1[i].DeleteObject();
        }
        MemDC.DeleteDC();
        cBitmap.DeleteObject();
        dc.DeleteDC();
}
```

全部示例代码请参见 CD 光盘。

第十五章 构建 Internet/Intranet 信息系统

15.1 概述 Microsoft Active Server Pages（ASP）是服务器端脚本环境，可用来创建交互式 Web 页并建立强大的 Web 应用程序。当服务器收到对 ASP 文件的请求时，它处理包含在用于构建发送给浏览器的 Web 页文件中的服务器端脚本。除服务器端脚本外，ASP 文件也可以包含 HTML（包括相关的客户端脚本）和 COM 组件调用，这些组件可执行不同任务，如连接到数据库或处理商业逻辑。因此，通过使用 COM/ActiveX 技术与 TwinCAT ADS-DLL 有机结合，运用 ASP 服务器端脚本环境打造功能强大的 Internet/Intranet 信息系统。

15.2 Active Server Pages 简介

15.2.1 对于 HTML 作者

用 ASP 编写服务器端脚本可使创建复杂、实用的 Web 应用程序变得十分简单。如果希望将 HTML 表单信息存储在数据库中、根据访问者的自选项自定义 Web 站点或对不同的浏览器使用不同的 HTML 功能，将会发现 ASP 提供了优异的解决方案。

例如，从前要在 Web 服务器上处理用户输入，必须首先学习用 Perl 或 C 等语言建立传统的公共网关接口(CGI)应用程序。而使用 ASP 后，仅通过在 HTML 文档中直接嵌入的简单服务器端脚本，便可以收集 HTML 表单信息，并传递到数据库。

15.2.2 对于高级 Web 脚本编写者

由于 ASP 使用了中性语言，因此如果对 VBScript、JScript 或 PERL 等脚本语言十分熟悉，那么已经了解了 Active Server Pages 的使用方法。再有，在 ASP 页中，可以使用已经装有 COM 脚本兼容编辑引擎的任何脚本编辑语言。ASP 使用 VBScript 和 JScript 脚本引擎，但仍可安装用于 PERL、REXX 和 Python 的脚本引擎，它们可从第三方供应商获得。

15.2.3 对于 Web 开发和编程人员

如果使用 Visual Basic、C++或 Java 等编程语言开发过后端 Web 应用程序，将会发现 ASP 是创建 Web 应用程序灵活而快速的方法。除了添加脚本为应用程序创建迷人的 HTML 界面之外，还可以建立自己的 COM 组件。可以将应用程序的商业逻辑封装在可重复使用的模块中，以便在脚本、其他组件或其他程序中调用。

15.2.4 Active Server Pages 模型

当浏览器向 Web 服务器请求.asp 文件时，服务器端脚本便开始运行。Web 服务器于是调用 ASP，用它从头至尾处理所请求的文件、执行脚本命令，并将 Web 页发送到浏览器。

因为脚本运行于服务器而不是客户端，所以生成发送到浏览器的 HTML 页等工作便由 Web 服务器负责。服务器端脚本无法被预先复制，因为返回到浏览器的只是脚本的运行结果。用户无法得知创建其所查看的页面使用的脚本命令。

15.3 ASP 的新特性

15.3.1 新的流控制能力

ASP 的 Server 对象具有两种可用来控制程序流的新方法：“Server.Transfer”和“Server.Execute”。与重定向请求（需要往返于客户端）不同，可以使用这些方法将请求直接传送到.asp 文件，而不需要离开服务器。

15.3.2 错误处理

ASP 具有新的错误处理能力，可以使用自定义的错误消息.asp 文件来捕捉错误。可以使用新的“Server.GetLastError”方法来显示有用信息，如错误描述或发生错误的行号。

15.3.3 无脚本

ASP 由于静态内容的处理速度通常快于服务器端内容，因此以前只将.asp 文件扩展名指派给包含 ASP 功能性的文件。无论何时，如果需要在静态.html 文件中添加 ASP，只能手工添加.asp 文件扩展名并修正相关超级链接。在 ASP 的最新版本中，不包含服务器端功能性的.asp 文件的处理速度比以前快了许多。因此，如果正在创建展开的 Web 应用程序并且其中的文件最终可能需要 ASP 功能性，现在就可以很方便地为这些文件指派.asp 文件扩展名，而不必考虑它们是否包含静态或服务器端内容。

15.3.4 性能增强的对象

ASP 现在提供流行的可安装组件的性能增强版本。这些对象能够可靠地适用于各种 Web 发布环境。

15.3.5 XML 集成

扩展标记语言(XML)允许描述复杂的数据结构或文档。可以在各种应用程序、客户端和服务器之间共享此信息。使用 Microsoft Internet Explorer 4.0 或更高版本附带的 Microsoft XML Parser，可以创建服务器端应用程序，该应用程序允许 Web 服务器与 Internet Explorer 4.0（或更高版本）或任何包含 XML 解析能力的服务器交换 XML 格式的数据。

15.3.6 Windows 脚本组件

ASP 支持 Microsoft 强大的脚本新技术-Windows 脚本组件。现在可以将商业逻辑脚本过程转换为可重复使用的 COM 组件，该组件可用于 Web 应用程序和其他组件对象模型(COM)适用的程序。

15.3.7 确定浏览器能力的新方法

ASP 具有可确定浏览器准确能力的新特性。当浏览器发送能描述其能力的 cookie（可通过使用简单的客户端脚本来安装这样的 cookie）时，可以创建一个“浏览器能力组件”实例，以便检索随 cookie 返回的浏览器属性。可以使用此特性来确定浏览器能力并对应用程序作相应调整。

15.3.8 ASP 自调整

ASP 现在可以检测执行请求何时被外部资源阻断，并自动提供更多线程以便同时执行附加请求和继续正常处理。如果 CPU 负担过重，ASP 将减少线程数量，以便减少因太多非阻断请求同时执行而产生的持续不断的交换。

15.3.9 服务器端包含（使用 SRC 属性）

现在可以使用 HTML <SCRIPT>...</SCRIPT> 标签的 SRC 属性来完成服务器端包含。当使用 SRC 属性指定虚拟或相对路径并使用 RUNAT=SERVER 属性表示服务器端执行时，可以完成与“#Include”命令一样的功能性。

15.3.10 编码的 ASP 脚本

以前，Web 开发人员无法禁止他人查看隐藏在脚本后面的逻辑。ASP 现在支持 Microsoft Visual Basic Scripting Edition (VBScript)和 Microsoft JScript 5.0 附带的新的脚本编码实用程序。Web 开发人员可以对客户端和服务器端脚本应用编码方案，以便使程序逻辑（使用标准 ASCII 字符）不可读。已编码的脚本在运行时由脚本引擎解码，因此不需要单独的实用程序。虽然此特性不是专门的安全加密解决方案，但可防止大多数用户无意中查看或复制脚本。

15.4 Windows 2000/XP 中安装 IIS

Windows 2000 Server 在安装的过程中会自动安装 IIS 5.0，而 Windows 2000 Professional 和 Windows XP 则不会，必须用添加 Windows 组件的方式另行安装。下面以 Windows XP 为例，介绍安装 IIS 5.1 的过程。

15.4.1 安装 IIS

安装 IIS 的步骤如下：

① 从桌面的“开始”菜单中选择“控制面板”，打开“控制面板”。双击“添加或删除程序”图标，打开“添加或删除程序”对话框。

② 用鼠标单击左边的“添加/删除 Windows 组件”图标，打开“Windows 组件向导”(图 15-4-1)。

图 15-4-1　Windows 组件向导

选中“组件”列表中的“Internet 信息服务（IIS）”选项，单击“下一步”按钮，组件向导即开始安装所选组件。在安装向导的最后一页单击“完成”按钮，完成组件的安装。

15.4.2 Internet 信息服务器

在“控制面板”中双击“管理工具”图标，打开“管理工具”对话框，在其中双击“Internet 信息服务”图标，打开 Internet 信息服务器窗口，如图 15-4-2 所示。

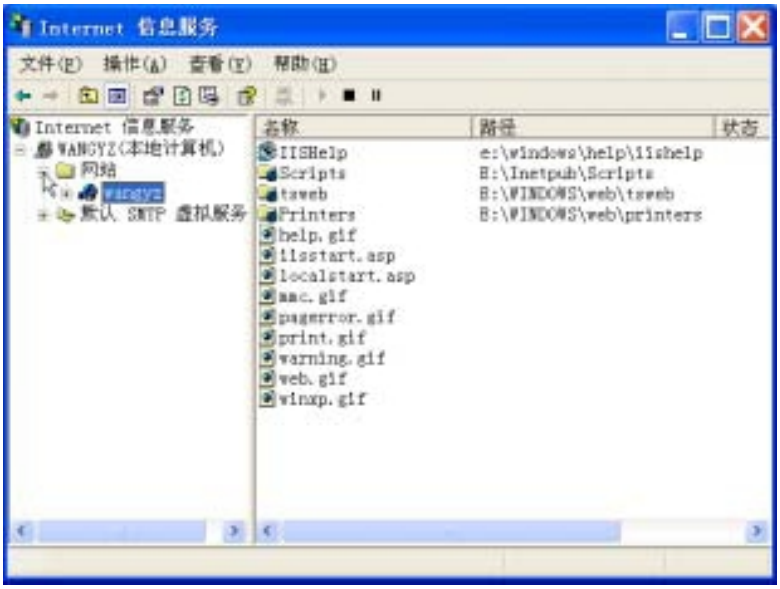


图 15-4-2 Internet 信息服务器窗口

15.4.3 设置虚拟目录

在“Internet 信息服务器窗口”中，用鼠标右击默认网站“mypc”，在弹出的快捷菜单中选择“新建”/“虚拟目录”，打开“虚拟目录创建向导”，如图 15-4-3 所示。

图 15-4-3 虚拟目录创建向导

依次输入“虚拟目录别名”、在“目录”栏中输入或通过“浏览”按钮找到要发布到网站上的内容的位置——服务器中的真实目录（工作目录：H:\ASP 例）、然后选择该目录开放的权限，选中“执行”复选框，这样可以使服务器能够运行 ASP 应用程序。

15.4.4 测试 IIS

关闭“Internet 信息服务器窗口”。

激活浏览器，在地址栏中输入本机的网址，如 http://localhost/或 http://mypc/，其中 mypc

为安装 Windows 系统时设置的本机名称，http://localhost/是系统默认的计算机名称。

如果网址输入正确，浏览器将打开 IIS 默认的网页，如图 15-4-4 所示，并同时打开 IIS 5.1 帮助文档。

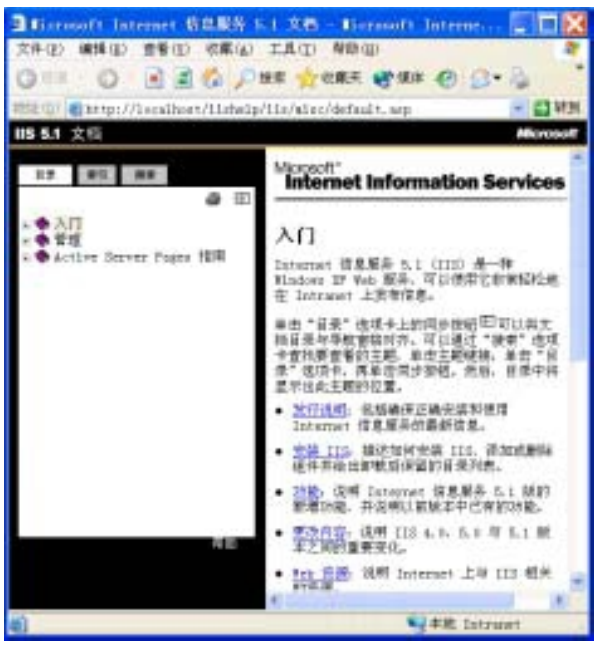


图 15-4-4 IIS 默认网页

激活浏览器，选择“工具”菜单中的“Internet 选项”，打开“Internet 选项”对话框，在“安全”选项卡中单击“自定义级别”按钮，打开“安全设置”对话框，如 15-4-5 所示。

在“安全设置”对话框的“设置”列表中找到最后一项：用户验证。选择其中的“自动使

用当前用户名和密码登录”，然后单击“确定”按钮返回浏览器。

在地址栏中输入系统默认的 IP 地址 http://127.0.0.1，同样能够打开 IIS 默认网页。

图 15-4-5 “安全设置”对话框

15.5 设置 Global.asa 服务器启动环境

Global.asa 文件可用于设置相关虚拟目录下的服务器启动运行环境，可以在此定义 TwinCAT 相关的服务，示例如下。

```
<OBJECT
    RUNAT="Server"
    SCOPE="Application"
    ID="TcPLC"
    PROGID="TcScript.TcScriptSync">
</OBJECT>
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
    Sub Application_OnStart()
        Call TcPLC.ConnectTo("", 801)
    End Sub
</SCRIPT>
```

15.6 创建 COM/ActiveX 控件

使用 Visual C++ 编程环境，可以创建功能强大的 COM/ActiveX 控件。通过 ASP 或 HTML 使用这些控件，可以实现用户所需要的各种功能。下面以 BKMytime 定时器为例，说明创建 COM/ActiveX 控件的过程，该定时器的缺省时间间隔为 100 ms，并可以通过 TimeTick

属性动态修改该时间间隔值。相关代码如下：

15.6.1 创建定时器

```
int CBKMyTimeCtrl::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (COleControl::OnCreate(lpCreateStruct) == -1)
        return -1;

    m_nTimer = SetTimer(5,m_timeTick,NULL);
    return 0;
}
```

15.6.2 动态修改时间间隔

```
void CBKMyTimeCtrl::OnTimeTickChanged()
{
    KillTimer(m_nTimer);
    m_nTimer = SetTimer(5,m_timeTick,NULL);
    SetModifiedFlag();
}
```

15.6.3 触发时间间隔事件

```
void FireOnTimeTick()
    {FireEvent(eventidOnTimeTick,EVENT_PARAM(VTS_NONE));}
```

15.6.4 销毁定时器

```
void CBKMyTimeCtrl::OnDestroy()
{
    COleControl::OnDestroy();
    KillTimer(m_nTimer);
}
```

完整的定时器代码参见光盘示例。

15.7 使用 VB Script

在 ASP 或 HTML 中，通过使用 COM/ActiveX 控件，可以轻松扩展 ASP 和 HTML 的功能，使 TwinCAT ADS-DLL 能够无缝集成到 ASP/HTML 之中，示例代码如下。

```
<HTML>
  <HEAD>
    <TITLE>My COM</TITLE>
      <META http-equiv=Content-Type content="text/html; charset=gb2312">
      <META content="Microsoft FrontPage 4.0" name=GENERATOR>
  </HEAD>
  <BODY>
    <BR><BR>
    <p align="center"><b>
    <font size="6" face="楷体_GB2312" color="#FF0000">测试画面 6</font></b>
    <p>
    <OBJECT
        aling="center"
        id=MyTime01
        height=1 width=1
        classid=CLSID:18DB8F46-AF32-4A3B-BB67-D9DB605403C8>
        <param name="_Version" value="65536">
        <param name="_ExtentX" value="26">
        <param name="_ExtentY" value="26">
        <param name="_StockProps" value="0">
        <param name="TimeTick" value= "500">
    </OBJECT>
    <SCRIPT FOR="MyTime01" EVENT="onTimeTick" LANGUAGE="VBScript">
        MyLabel01.SetText = MyTime01.OutValue
        MyLabel02.SetText = MyTime01.OutValue1
        MyLabel03.SetText = MyTime01.OutValue2
        MyLabel04.SetText = MyTime01.OutValue3
        MyLabel05.SetText = MyTime01.OutValue4
        MyLabel06.SetText = MyTime01.OutValue5
        MyLabel07.SetText = MyTime01.OutValue6
        MyLabel08.SetText = MyTime01.OutValue7
        MyLabel11.SetText = MyTime01.OutValue
        MyLabel12.SetText = MyTime01.OutValue1
        MyLabel13.SetText = MyTime01.OutValue2
        MyLabel14.SetText = MyTime01.OutValue3
        MyLabel15.SetText = MyTime01.OutValue4
        MyLabel16.SetText = MyTime01.OutValue5
        MyLabel17.SetText = MyTime01.OutValue6
        MyLabel18.SetText = MyTime01.OutValue7
        MyLabel19.SetText = MyTime01.OutValue8
        MyLabel20.SetText = MyTime01.OutValue9
    </SCRIPT>
    </p>
    <table border="1" width="100%" cellspacing="0" cellpadding="0">
```

```
  <tr>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 01</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 02</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 03</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 04</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 05</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 06</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 07</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 08</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 09</font></td>
    <td width="10%" align="center">
          <font face="楷体_GB2312" size="2">测试数据 10</font></td>
  </tr>
  <tr>
    <td width="10%" align="center">
      <p align="center">
<OBJECT
     aling="center"
     id=MyLabel01
     height=15
     width=50
     classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
     <param name="_Version" value="65536">
     <param name="_ExtentX" value="1323">
     <param name="_ExtentY" value="397">
     <param name="_StockProps" value="0">
     <param name="SetText" value="10">
     <param name="BackColor" value="255">
     <param name="TextColor" value="16777215">
</OBJECT>
     </td>
     <td width="10%" align="center">
       <p align="center">
<OBJECT
     aling="center"
```

```
    id=MyLabel02
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="20">
    <param name="BackColor" value="0">
    <param name="TextColor" value="65280">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel03
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H808080">
    <param name="TextColor" value="&H00FFFF">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel04
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&HC0C0C0">
    <param name="TextColor" value="&HFF00FF">
</OBJECT>
```

```
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel05
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&HFFFFFF">
    <param name="TextColor" value="&H000000">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel06
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H80">
    <param name="TextColor" value="&HFFFFFF">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel07
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
```

```
    <param name="SetText" value="10">
    <param name="BackColor" value="&H008080">
    <param name="TextColor" value="16777215">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel08
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H008000">
    <param name="TextColor" value="16777215">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel09
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="0">
    <param name="TextColor" value="16777215">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel10
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
```

```
      <param name="_Version" value="65536">
      <param name="_ExtentX" value="1323">
      <param name="_ExtentY" value="397">
      <param name="_StockProps" value="0">
      <param name="SetText" value="10">
      <param name="BackColor" value="0">
      <param name="TextColor" value="16777215">
</OBJECT>
      </td>
    </tr>
</table>
      <table border="0" width="100%">
        <tr>
          <td width="50%">
                
           <SCRIPT FOR="SetText1" EVENT="onClick" LANGUAGE="VBScript">
                MyLabel09.SetText = "100"
                MyLabel10.SetText = "200"
           </SCRIPT>
          </td>
          <td width="10%">
                
           <SCRIPT FOR="SetText2" EVENT="onClick" LANGUAGE="VBScript">
                MyLabel09.SetText = "200"
                MyLabel10.SetText = "100"
           </SCRIPT>
          </td>
          <td width="10%">
          </td>
          <td width="10%">
          </td>
          <td width="10%">
           <input type="button" name="SetText1" value="设置数据 1">
          </td>
          <td width="10%">
           <input type="button" name="SetText2" value="设置数据 2">
          </td>
        </tr>
      </table>
<table border="1" width="100%" cellspacing="0" cellpadding="0">
    <tr>
      <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 11</font></td>
      <td width="10%" align="center">
```

```
            <font face="楷体_GB2312" size="2">测试数据 12</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 13</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 14</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 15</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 16</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 17</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 18</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 19</font></td>
        <td width="10%" align="center">
            <font face="楷体_GB2312" size="2">测试数据 20</font></td>
      </tr>
      <tr>
        <td width="10%" align="center">
          <p align="center">
<OBJECT
        aling="center"
        id=MyLabel11
        height=15
        width=50
        classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
        <param name="_Version" value="65536">
        <param name="_ExtentX" value="1323">
        <param name="_ExtentY" value="397">
        <param name="_StockProps" value="0">
        <param name="SetText" value="10">
        <param name="BackColor" value="8388608">
        <param name="TextColor" value="16777215">
</OBJECT>
        </td>
        <td width="10%" align="center">
          <p align="center">
<OBJECT
        aling="center"
        id=MyLabel12
        height=15
        width=50
        classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
```

```
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="20">
    <param name="BackColor" value="&H800000">
    <param name="TextColor" value="65280">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel13
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H800000">
    <param name="TextColor" value="&H00FFFF">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel14
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H800000">
    <param name="TextColor" value="&HFF00FF">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
```

```
    id=MyLabel15
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="8388608">
    <param name="TextColor" value="&amp;HFFFFFF">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel16
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H800000">
    <param name="TextColor" value="&HFFFFFF">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel17
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H800000">
    <param name="TextColor" value="16777215">
</OBJECT>
```

```
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel18
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H800000">
    <param name="TextColor" value="16777215">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel19
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
    <param name="SetText" value="10">
    <param name="BackColor" value="&H800000">
    <param name="TextColor" value="16777215">
</OBJECT>
    </td>
    <td width="10%" align="center">
<OBJECT
    aling="center"
    id=MyLabel20
    height=15
    width=50
    classid=CLSID:BAB0BAFB-92C8-4563-A67D-52653932FC80>
    <param name="_Version" value="65536">
    <param name="_ExtentX" value="1323">
    <param name="_ExtentY" value="397">
    <param name="_StockProps" value="0">
```

```
      <param name="SetText" value="10">
      <param name="BackColor" value="&H800000">
      <param name="TextColor" value="16777215">
</OBJECT>
      </td>
   </tr>
</table>
<BR>
<table border="0" width="100%">
   <tr>
      <td width="50%" align="center"><b>
            <font face="楷体_GB2312" color="#FF00FF">温度趋势图</font></b></td>
      <td width="50%" align="center"><b>
            <font face="楷体_GB2312" color="#FF00FF">流量趋势图</font></b></td>
   </tr>
</table>
<table border="0" width="100%">
   <tr>
      <td width="50%">
          <p align="center">
           <OBJECT
                aling="center"
                id=MyDemo01
                height=252
                width=454
                classid=CLSID:5C15B422-63E9-4CB0-8B66-050ABE2FAE7C>
               <param name="_Version" value="65536">
               <param name="_ExtentX" value="12012">
               <param name="_ExtentY" value="6668">
               <param name="_StockProps" value="0">
               <param name="BackColor" value="16777215">
               <param name="ForeColor" value="255">
               <param name="MarkColor" value="12632256">
               <param name="TimeElapse" value="200">
           </OBJECT>
          </p>
      </td>
      <td width="50%">
          <p align="center">
           <OBJECT
                aling="center"
                id=MyDemo02
                height=252
                width=454
```

```
                classid=CLSID:5C15B422-63E9-4CB0-8B66-050ABE2FAE7C>
                <param name="_Version" value="65536">
                <param name="_ExtentX" value="12012">
                <param name="_ExtentY" value="6668">
                <param name="_StockProps" value="0">
                <param name="BackColor" value="16777215">
                <param name="ForeColor" value="16711680">
                <param name="MarkColor" value="32768">
                <param name="TimeElapse" value="400">
            </OBJECT>
            </p>
        </td>
    </tr>
  </table>
 </BODY>
</HTML>
```

15.8 ASP 实例

光盘中包含一个使用 ASP/HTML 与 TwinCAT ADS-DLL 和 COM/ActiveX 控件技术实现 Internet/Intranet 信息系统的实例及框架，用户可以在此基础上进行扩展，并结合 ASP 中的数据库技术，轻松实现 Internet/Intranet 上的过程控制和信息管理系统。下面是该示例运行的效果图例。

图 15-8-1 测试画面 1

图 15-8-2 测试画面 2

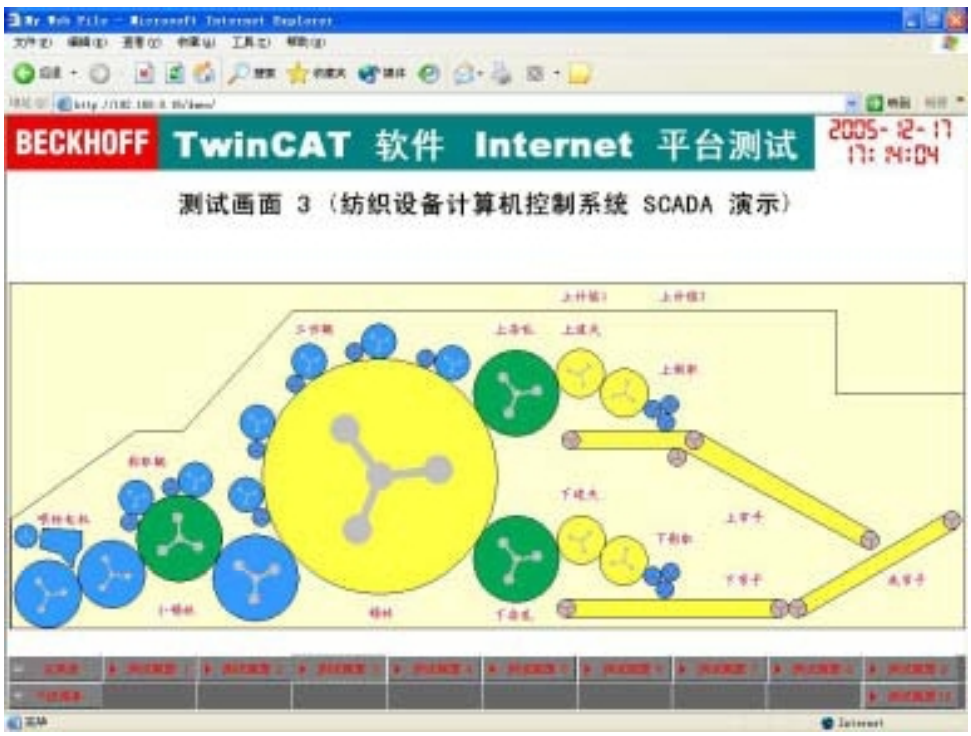


图 15-8-3 测试画面 3

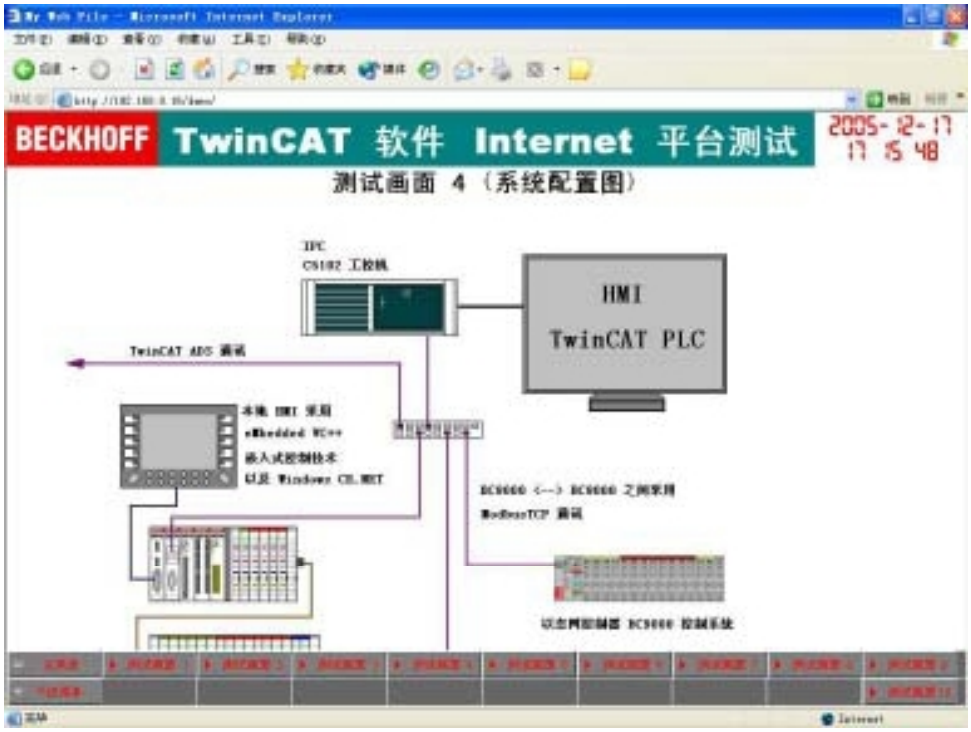


图 15-8-4 测试画面 4

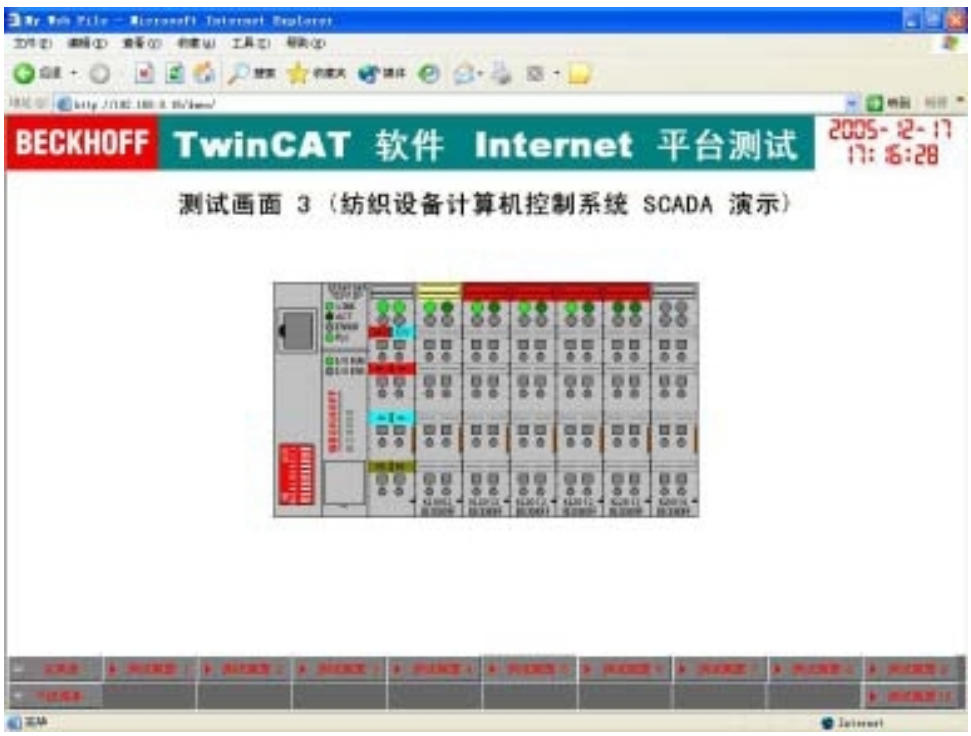


图 15-8-5 测试画面 5

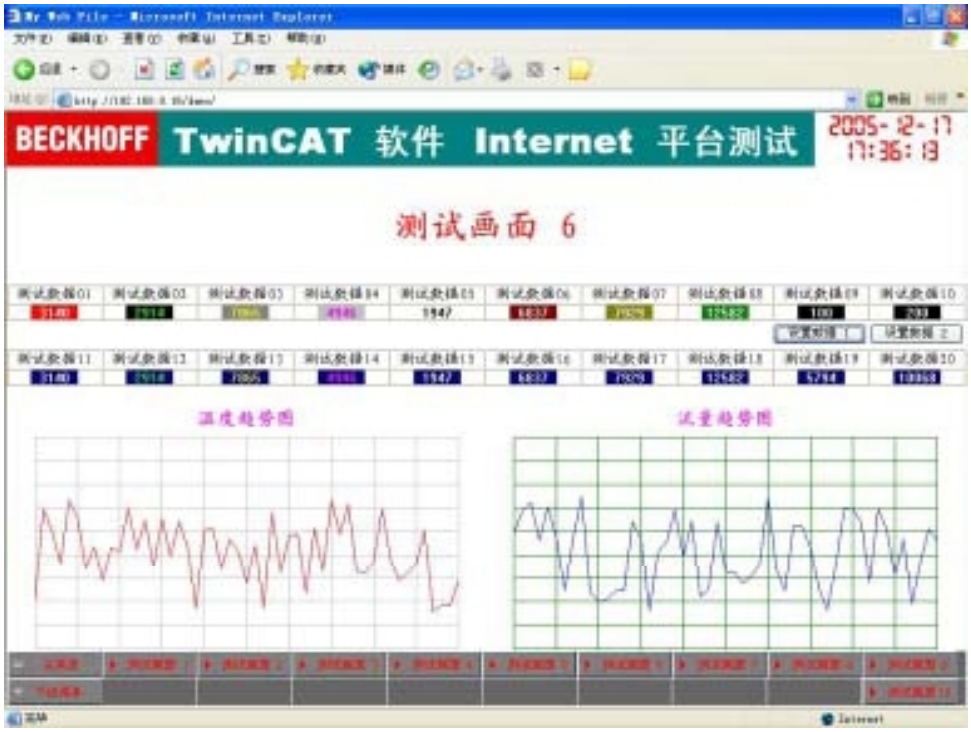


图 15-8-6 测试画面 6

图 15-8-7 测试画面 7

附录

A.1 Modem AT 指令常用 ASCII 编码对照表

编码	20H	21H	22H	23H	24H	25H	26H	27H
符号	SPACE	!	“	#	$	%	&	‘
编码	28H	29H	2AH	2BH	2CH	2DH	2EH	2FH
符号	(	)	*	+	,	-	.	/
编码	30H	31H	32H	33H	34H	35H	36H	37H
符号	0	1	2	3	4	5	6	7
编码	38H	39H	3AH	3BH	3CH	3DH	3EH	3FH
符号	8	9	:	;	<	=	>	?
编码	40H	41H	42H	43H	44H	45H	46H	47H
符号	@	A	B	C	D	E	F	G
编码	48H	49H	4AH	4BH	4CH	4DH	4EH	4FH
符号	H	I	J	K	L	M	N	O
编码	50H	51H	52H	53H	54H	55H	56H	57H
符号	P	Q	R	S	T	U	V	W
编码	58H	59H	5AH	5BH	5CH	5DH	5EH	5FH
符号	X	Y	Z	[	\	/	^	_
编码	60H	61H	62H	63H	64H	65H	66H	67H
符号	`	a	b	c	d	e	f	g
编码	68H	69H	6AH	6BH	6CH	6DH	6EH	6FH
符号	h	i	j	k	l	m	n	o
编码	70H	71H	72H	73H	74H	75H	76H	77H
符号	p	q	r	s	t	u	v	w
编码	78H	79H	7AH	7BH	7CH	7DH	7EH	7FH
符号	x	y	z	{	|	}	~	▌

A.2 BC8000 相关参数

地址：
0 ：保留地址
1 – 98 ：编程模式(从站模式)
99 ：BC8000 主站运行模式

BC8000 通讯相关参数表：

寄存器	特性	缺省值
32	波特率	2
	0：38400 bps	
	1：19200 bps	
	2：9600 bps	x
	3：57600 bps	
	4：1200 bps	
	5：2400 bps	
	6：4800 bps	
33	模式	2
	0：7 位数据位，偶校验	
	1：7 位数据位，奇校验	
	2：8 位数据位，无校验	x
	3：8 位数据位，偶校验	
	4：8 位数据位，奇校验	
34	停止位	0
	0：1 位停止位	x
	1：2 位停止位	

注：该参数表仅在 BC8000 为主站模式(地址为：99)时有效。

BC8000 技术参数

名称	RS-485 | BC8000
总线端子数量	64
最大现场总线字节数量	512 字节输入和 512 字节输出
最大端子字节数量	512 字节输入和 512 字节输出
最大可提供给 K-Bus 电源电流	1750 mA
程序内存	32 / 96 千字节
数据内存	32 / 64 千字节
保持型标志	512 字节

BC8000 从站模式通讯参数：

BC8000 工作于从站模式(地址：1-98)时，其 RS-485 的通讯参数为固定值，即：
19200 bps，8 位数据位，偶校验，1 位停止位

A.3 BK8000 相关参数

地址：
0 ：BK8000 主站运行模式
1 – 99 ：编程模式(从站模式)

BK8000 从站模式时的通讯参数：

BK8000 工作于从站模式(地址：1-99)时，其 RS-485 的通讯参数为：
19200 bps，8 位数据位，偶校验，1 位停止位

注：38400 bps 为缺省波特率

上位机/BC8000 主站命令请求格式：

请求说明	字节	值范围
起始标识	0	‘P’ (0x50)
过程数据输出字数量	1	0 - 255
信息标识	2	0 - 255
多节点地址	3	0 - 99
过程数据输出低字节(选项)	4 + 2 x n (n=0,1,2,…,125)	0 - 255
过程数据输出高字节(选项)	5 + 2 x n (n=0,1,2,…,125)	0 - 255
检验和 (CheckSum)	6 + 2 x n	0 - 255

BK8000 从站命令响应格式：

请求说明	字节	值范围
起始标识	0	‘p’ (0x70)
过程数据输入字数量	1	0 - 255
信息标识	2	0 - 255
多节点地址	3	0 - 99
状态	4	0 - 255
过程数据输入低字节(选项)	5 + 2 x n (n=0,1,2,…,125)	0 - 255
过程数据输入高字节(选项)	6 + 2 x n (n=0,1,2,…,125)	0 - 255
检验和 (CheckSum)	7 + 2 x n	0 - 255

BK8000 从站命令响应状态字节意义：

状态字节	错误 (位值 = 1)
Status.0	端子总线错误：与端子进行数据通讯时，发生错误
Status.1	组态错误：参见错误代码 1 和 2
Status.2	--
Status.3	--
Status.4	无效的过程数据输出长度：接收的过程数据输出字长度与 K-Bus 已有的物理数据长度不相等
Status.5	--
Status.6	--
Status.7	--

A.4 Modem 通讯连接示例

Modem 通讯连接示例		
PC1 笔记本 (内置 Modem)	← 电话交换机 →	PC2 台式机 (外置 Modem)
COM3 (Modem)		COM2 (Modem)
电话号码：#16		电话号码：#18
Windows XP 专业版		Windows 98 SE
超级终端配置		
	名称	
Test1		Test2
	57600 bps	
	8 位数据位	
	无校验	
	1 位停止位	
	数据流控制：硬件	
ATD#18 <CR>		显示：RING 输入：ATA <CR>
	显示：CONNECT 57600	
输入：1234567890	→	显示：1234567890
显示：abcdefgh	←	输入：abcdefgh
	挂断命令 (图标)	

注：电话交换机 (8 门) 分机电话号码：
PC1 : #16
PC2 : #18