SIEMENS

WinCC

全局脚本

手册

6AV6392-1XA04-0AH0

1997年8月版

我们已检查了本手册的内容，使其与所描述的硬件和软件相一致。由于差错是不可避免的，所以我们不能保证完全的一致性。然而我们会经常审定手册中的数据并在以后的版本中做必要的修正。欢迎提出改进建议。

技术数据修改后生效

6AV6392-1XA04-0AH0

中华人民共和国印刷

Siemens Aktiengesellschaft

目录表

前言

该手册的目的

该手册介绍每个软件组元的功能和操作格式。可以在手册和在线帮助中利用内容表格或索引很快地找到需要的信息。

总览和组态实例

作为视窗控制中心(WinCC)软件包的一部分，“使用入门”手册包含了WinCC总览和组态实例。通过该组态实例，可以了解WinCC中各组成部分的基本功能。

附加支持

如果存在技术问题，请与当地的西门子服务机构联系。可以在系统手册、产品样本和CompuServe中找到当地西门子服务机构的地址。此外，可以拨打热线电话011-49-911-895-7000(Fax 7001)。

可以通过internet得到有关信息，地址为
www.aut.siemens.de/coros/html_00/coros.htm。

关于SIMATIC产品的信息

可以通过下列方法得到关于SIMATIC产品的最新信息：

* Internet地址：http://www.aut.siemens.de/
* 传真：08765-93 02 77 95 00

此外，通过SIMATIC客户支持可提供和下载，有关SIMATIC产品使用的信息：

* Internet地址：
http://www.aut.siemens.de/support/html_00/index.shtml
* SIMATIC客户支持信箱，电话：+49（911）895-7100
* 访问信箱，请使用高于V.34 （28.8 kBaud）的调制解调器，参数设置如下：8,N,1,ANSI或通过ISDN拨号(x.75, 64 kBit)。

SIMATIC客户支持的电话为+49 (911) 895-7000，传真为
+49 (911) 895-7002。可以通过Internet信箱或上述地址查询。

1 简介

全局脚本是用于C函数和动作的常规术语，根据不同的动作类型，动作可以在一个项目内或几个项目之间使用。

可以区别下列类型：

项目函数
可以创建新的项目函数或改变现有的项目函数。项目函数创建后是唯一的。

标准函数
可以创建新的标准函数或改变现有的标准函数。可越过项目识别标准函数。

内部函数
不能创建或改变内部函数。可越过项目识别内部函数。

动作
动作作为全局脚本，可以创建和改变。这些动作创建后是唯一的。

在下列区域使用项目函数、标准函数和内部函数：

- 相应对象的C动作
- 动态对话框中创建的相应对象动作

可使用项目函数、标准函数和内部函数使下列区域变为动态：
过程值归档
用户归档
压缩归档

运行时，可以在过程控制中执行全局脚本动作。开始前，必须激活项目和启动运行系统。

激活项目，参见控制中心，2.1.4节。

启动运行系统，参见控制中心，6.3节。

注意：动作将被解释处理。当需要执行多个动作时，应该考虑系统的负荷情况。因此，建议将多个动作分开放置于DLL(动态链接库)中。

通过可用的DLL，可以实现WinCC函数和动作的功能。为此使用下列命令：

```
#pragma code("<NAME>.dll")
#include "<NAME>.h"
#pragma code()
```

备注：如果在与项目或标准函数的连接中定义自己的结构，下列结构将引起翻译错误：

```
struct MyOwnStruct;
{
....
}
void MyFunction (struct MyOwnStruct...);
{
...
}
```

此函数的原型传送到文件“ap_pbib.h”或“ap_glob.h”，但其结构丢失。

为了识别结构的传送和返回功能，按下列步骤进行：

在项目目录(项目路径\library)，在‘#include “ap-pbib.h”’结构前插入“apdefap.h”。

```
struct MyOwnStruct;
{
....
}
#include "ap-pbib.h"
```

同样相应地：

```
#pragma code ("pdlcsapi.dll")
#include "pldcsapi.h"
#prgma code()

#include "ap_pbib.h"
```

这样，当项目函数的原型被传送时，对象识别了MyOwnStruct和返回函数的结构。
在项目函数本身，写入：

```
#include "apdefap.h"
```

2 项目函数

项目函数是用户可编程和修改的C函数。它们创建后对于当前项目是唯一的。可以使用项目函数使图形对象和归挡动态化。同样可以在其它项目函数和全局脚本动作中使用它们。在控制中心启动全局脚本编辑器，通过全局脚本编辑器创建项目函数。

按下列步骤创建项目函数：

1. 制订函数
2. 添加函数信息
3. 编译函数
4. 保存函数，如果需要，可更改名称。
5. 根据需要，可生成头文件.

完成创建项目函数。

第一次创建项目函数，头文件apdefap.h位于项目文件夹...\<Projectname>\LIBRARY中。文件包含来自项目库ap_plib.h的头文件ap_glob.h。

头文件ap_plib.h包含项目函数的定义。如果项目函数被装载，并且缺省值或其它项目函数在此处使用，必须集成头文件apdefap.h或ap_glob.h。

如何制订函数：

1. 在控制中心启动全局脚本。“全局脚本”对话框打开。

2. 在函数浏览器中函数类型(项目函数、标准函数或动作)上单击鼠标右键，打开弹出式菜单。放置光标或单击“新建”打开子菜单，并选择“函数”子菜单。
   或
   使用位于工具栏中的  按钮。
   此选择打开已经包含C函数概要的窗口。

3. 制订函数。

可以使用下列编辑功能：

功能	键操作
新行	Enter
向右删除字符	Delete
向左删除字符	Backpace
跳至行的起始	Home
跳至行的末尾	End
跳至文本起始	Ctrl+Home
跳至文本末尾	Ctrl+End
移动光标	Cursor Keys
剪切选择的文本	Ctrl+X
复制选择的文本	Ctrl+C
从剪贴板上粘贴文本	Ctrl+V

功能	鼠标
使文本高亮度	用鼠标左键
使字高亮度	双击鼠标左键
移动插入点	用鼠标左键

附加的编辑功能包括下列内容：

- 写入模式是“Insert”。
- 高亮度的文本被键盘上键入的下一个字符代替。
- 扩展高亮度：

  使一个区域高亮度：
  将光标放在希望高亮度的区域起始处。按住Shift键，将光标移动到希望高亮度的区域末尾。

  扩展高亮度的范围：
  按住Shift键，将光标移动到希望高亮度的区域末尾。

用户编辑的函数中，可以在光标位置处添加项目函数、标准函数和内部函数，也可以用函数代替选择的文本。按下列步骤进行：

1. 定位光标或将希望代替的文本高亮度。
2. 在函数浏览器中，打开适当的文件夹并在期望的函数上单击鼠标右键打开弹出式菜单。
3. 在弹出式菜单中选择“发送至...”菜单条目。
4. 从子菜单列表中选择适当的编辑窗口。如果已打开多个窗口，子菜单包含用于每个窗口的条目。
5. 如果需要，更新粘贴的函数的参数。

单击  按钮，输入标签名。此动作打开“选择标签”对话框，可以选择期望的标签。如果需要，打开文件夹。

如何添加关于函数的信息：

如果已打开多个编辑窗口，使包含期望函数的窗口成为工作窗口。

1. 单击  按钮打开“描述”对话框。

   创建日期，改变日期，版本已经被输入。创建和改变日期的格式(从左到右)是“日”，“月”，“年”。

2. 输入或改变“创建者”和“修改者”对话框。
3. 在“注释”对话框中输入注释，例如，函数的简短说明。
4. 如果想给函数分配口令以免被更改，单击“口令”复选框来激活“更改”按钮。
   单击“更改”按钮，在“输入口令”方框中输入口令。
   重新输入口令，并且确认。
   单击“确定”按钮，关闭对话框。现在已经用口令保护函数。
5. 单击“确定”按钮，关闭“描述”对话框。该项被应用。

如何编译函数/动作：

如果已打开多个编辑窗口，将包含期望函数/动作的窗口作为工作窗口。

单击  按钮开始编译。

工作编辑窗口被分为两部分。上半部包含函数/动作和显示来自编译器信息的按钮。

如何保存函数：

如果创建新的函数或动作，它们从系统得到缺省名字。

如果希望用当前名字保存函数或动作，按下列步骤进行：

1. 单击  按钮。编辑窗口依然打开。
2. 在“文件”菜单中选择“保存”命令。编辑窗口依然打开。
3. 在编辑窗口单击“关闭”方框。此动作打开一个方框，告诉存在没有被保存的变化。然后可以选择保存函数/动作，拒绝改变，或取消此操作。如果取消此操作，编辑窗口依然打开，否则关闭。

如果希望用其他名称保存函数/动作，在“文件”菜单选择“另存为...”。此选择打开一个对话框，可以在方框内设置名字和保存位置。

通常，保存位置按下列步骤设置：

- 对于项目函数：
  在项目文件夹的“library”子文件夹中
- 对于动作：
  在项目文件夹的“Pas”子文件夹中
- 对于标准函数：
  在“Aplib”子文件夹的WinCC系统文件夹中。

如何产生新的头文件：

如果函数自行生成，将它们复制到适当的文件夹可以使它们适用于项目。

这样使得函数可用，但对于系统是未知的。

使用  按钮或选择“产生头文件”菜单条目来重新产生头文件。这样系统就可以识别新的或改变过的函数。

按同样方法，删除函数也使得系统不能识别它。

当新的头文件被产生，新函数显示在全局脚本编辑器的函数浏览器中。

3 标准函数

系统使得标准函数可用。这些函数可以根据需要改变。此外，可以自己创建标准函数。

标准函数可越过项目识别。

使用标准函数,可以使图形对象和归档文件动态化。同样可以在项目函数、其它标准函数和全局脚本中使用它们。

使用**全局脚本**编辑器创建标准函数。在**控制中心**启动编辑器。

必须按以下步骤创建标准函数：

- 制订函数
- 扩展函数信息
- 编译函数
- 保存函数如果需要，将其改名。
- 如果需要，生成头文件。

创建标准函数完成。

集成了头文件ap_glob.h的头文件apdefap.h包含标准函数的定义。

文件apdefap.h和ap_glob.h位于文件夹...\APLIB。

如果新的标准函数被创建，并且其它标准函数在此使用，必须集成头文件apdefap.h或ap_glob.h.

新的标准函数被加入服务器中位于WinCC安装文件夹...\APLIB下的标准函数中。

系统所提供标准函数可分为：

报警
图形
标签记录
WINCC
WINDOWS

基于选项软件包(例如，基本过程控制，高级过程控制)，系统提供附加的标准函数。

这些函数分为下列类别：

选项
报告
分割画面管理器
用户归档

包含在“用户归档”中的标准函数，被更详细的解释在Simatic-WinCC-手册“标签记录用户归挡”。

3.1 报警（标准函数）

这些函数的返回值为下列出错代码：

- TRUE: =函数执行正确。
- FALSE: =产生错误

BOOL **GMsgFunction(**char* pszMsgData**)**
功能：

此函数表达了用于单个信息的全局函数。它提供单个信息的信息数据，如果“触发动作”参数被设置，它可以被激活。

参数：

pszMsgData =存储信息记录的缓冲器指针。

BOOL **OnBtnArcLong(**char* pszMsgWin**)**
BOOL **AXC_OnBtnArcLong(**char* pszMsgWin**)**
¹¦ÄÜ：

通过图形对象完成外部信息窗口的操作。此函数显示顺序归挡（显示顺序归挡）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnArcLong)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnArcLong)

BOOL **OnBtnArcShort(**char* pszMsgWin**)**
BOOL **AXC_OnBtnArcShort(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数显示短期归挡（显示短期归挡）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnArcShort)
pszMsgWin =在OLE-控制-单元上的指针
(AXC_OnBtnArcShort)

BOOL **OnBtnComment(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数显示注释（注释）。

参数：

pszMsgWin =信息窗口名称的指针

BOOL **OnBtnEmergAckn(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数打开确认对话框(紧急确认/重新设置)。

参数：

pszMsgWin =信息窗口名称的指针

BOOL **OnBtnHornAckn(**char* pszMsgWin**)**
BOOL **AXC_OnBtnHornAckn(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数确认报警器信号（报警器确认）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnHornAckn)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnHornAckn)

BOOL **OnBtnInfo(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数显示信息文本（信息文本）。

参数：

pszMsgWin =信息窗口名称的指针

BOOL **OnBtnLanguage(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数允许你改变信息窗口操作的语言（语言开关）。

参数：

pszMsgWin =信息窗口名称的指针

此函数不支持以WinCCV4.0启动。

BOOL **OnBtnLock(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数激活“设置锁定”对话框。

参数：

pszMsgWin =信息窗口名称的指针

BOOL **OnBtnLoop(**char* pszMsgWin**)**
BOOL **AXC_OnBtnLoop(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数触发选择的信息的“循环报警”函数。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnLoop)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnLoop)

BOOL **OnBtnMsgFirst(**char* pszMsgWin**)**
BOOL **AXC_OnBtnMsgFirst(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数切换信息列表的起始位置（第一列表）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnMsgFirst)
pszMsgWin =在OLE-控制-单元上的指针
(AXC_OnBtnMsgFirst)

BOOL **OnBtnMsgLast(**char* pszMsgWin**)**
BOOL **AXC_OnBtnMsgLast(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数切换信息列表的末尾位置（最后列表）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnMsgLast)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnMsgLast)

BOOL **OnBtnMsgNext(**char* pszMsgWin**)**
BOOL **AXC_OnBtnMsgNext(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数切换信息列表中的下一个信息（下一个信息）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnMsgNext)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnMsgNext)

BOOL **OnBtnMsgPrev(**char* pszMsgWin**)**
BOOL **AXC_OnBtnMsgPrev(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数切换信息列表中的前一个信息（前一个信息）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnMsgPrev)
pszMsgWin =在OLE-控制-单元上的指针
(AXC_OnBtnMsgPrev)

BOOL **OnBtnMsgWin(**char* pszMsgWin**)**
BOOL **AXC_OnBtnMsgWin(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数调用过程信息窗口（调用过程信息窗口）。

备注：
过程信息窗口包含没有被确认的当前信息。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnMsgWin)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnMsgWin)

BOOL **OnBtnPrint(**char* pszMsgWin**)**
BOOL **AXC_OnBtnPrint(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数激活报表(报表函数)。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnPrint)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnPrint)

BOOL **OnBtnScroll(**char* pszMsgWin**)**
BOOL **AXC_OnBtnScroll(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数激活水平和垂直滚动函数(自动滚动开/关)。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnScroll)
pszMsgWin =在OLE-控制-单元上的指针
(AXC_OnBtnScroll)

BOOL **OnBtnSelect(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数激活“设置选择”对话框。

参数：

pszMsgWin =信息窗口名称的指针

BOOL **OnBtnSinglAckn(**char* pszMsgWin**)**
BOOL **AXC_OnBtnSinglAckn(**char* pszMsgWin**)**
功能：

通过图形对象完成外部信息窗口的操作。此函数执行信息的确认（单个确认）。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnSinglAckn)
pszMsgWin =在OLE-控制-单元上的指针(AXC_OnBtnSinglAckn)

BOOL **OnBtnVisibleAckn(**char* pszMsgWin**)**
BOOL **AXC_OnBtnVisibleAckn(**char* pszMsgWin**)**

功能：

通过图形对象完成外部信息窗口的操作。此函数执行所有在信息窗口中可见信息的确认(组确认)。

参数：

pszMsgWin =信息窗口名称的指针(OnBtnVisibleAckn)
pszMsgWin =在OLE-控制-单元上的指针
(AXC_OnBtnVisibleAckn)

3.2 画面(标准函数)

void **OpenPicture(**Picture Picture Name**)**
功能：

用你设置的画面名称打开图形。

参数：

画面名称 = 画面名称

3.3 标签记录（标准函数）

模板
工具栏按钮

工具栏按钮函数的返回值为下列出错代码：

- TRUE： 函数被正确执行。
- FALSE：产生错误

3.3.1 模板

int **TlgGetColumnPosition(**char* lpszTemplate**)**
返回值：

在制表窗口中列指针的当前位置。

功能：

在制表窗口中给列指针当前位置。制表窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate =在制表窗口名称上的指针

int TlgGetNumberOfColumns(char* lpszTemplate)
返回值：

在制表窗口中列的编号。

功能：

在制表窗口中给列的编号。制表窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate =在制表窗口名称上的指针

int TlgGetNumberOfRows(char* lpszTemplate)
返回值：

在制表窗口中行的编号。

功能：

在制表窗口中给行的编号。制表窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate =在制表窗口名称上的指针

int TlgGetNumberOfTrends(char* lpszTemplate)
返回值：

在趋势窗口中可见趋势的编号。

功能：

在趋势窗口中给可见趋势编号。趋势窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate =在趋势窗口名称上的指针

int **TlgGetRowPosition(**char* lpszTemplate**)**

返回值：

在制表窗口中行指针的当前位置。

功能：

在制表窗口中给行指针当前位置。制表窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate	=在制表窗口名称上的指针

char* **TlgGetRulerArchivNameTrend(**char* lpszTemplate, int nTrend**)**

返回值：

趋势窗口中标尺位置上趋势的归挡名称。

功能：

在趋势窗口中标尺位置上给趋势的归挡名称一个编号“n趋势”。趋势窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate	=在趋势窗口名称上的指针
n趋势	=趋势的编号 (0<=n趋势<=可见趋势的编号-1)

SYSTEMTIME **TlgGetRulerTimeTrend(**char* lpszTemplate, int nTrend**)**

返回值：

趋势窗口中标尺位置上趋势的时间。

功能：

在趋势窗口中标尺位置上给趋势的时间一个编号“n趋势”。趋势窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate	=在趋势窗口名称上的指针
n趋势	=趋势的编号 (0<=n趋势<=可见趋势的编号-1)

double **TlgGetRulerValueTrend(**char* lpszTemplate, int nTrend**)**
返回值：

趋势窗口中标尺位置上趋势的值。

功能：

在趋势窗口中标尺位置上给趋势的值一个编号“n趋势”。趋势窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate	=在趋势窗口名称上的指针
n趋势	=趋势的编号 (0<=n趋势<=可见趋势的编号-1)

char* **TlgGetRulerVariableNameTrend(**char* lpszTemplate, int nTrend**)**
返回值：

趋势窗口中标尺位置上趋势的标签名称。

功能：

在趋势窗口中标尺位置上给趋势的标签名称一个编号“n趋势”。趋势窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate	=在趋势窗口名称上的指针
n趋势	=趋势的编号 (0<=n趋势<=可见趋势的编号-1)

char* **TlgGetTextAtPos(**char* lpszTemplate, int nColumn, int nLine**)**
返回值：

作为文本的制表窗口单元的内容

功能：

对于过程值归档和用户归档。作为文本给制表窗口单元内容单元由n列和n行指定。制表窗口的名称以参数lpszTemplate传送。

参数：

lpszTemplate	=在制表窗口名称上的指针
nColumn	=列的编号
nLine	=行的编号

3.3.2 工具栏按钮

BOOL **TlgTableWindowPressEditRecordButton(**char* lpszTemplateName**)**
功能：

允许/不允许编辑制表窗口(切换函数)

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressFirstButton(**char* lpszTemplateName**)**
功能：

在制表窗口的显示区域中显示第一个数据记录。显示的记录的编号基于组态的时间范围
参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressHelpButton(**char* lpszTemplateName**)**
功能：

在制表窗口中显示在线帮助
参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressInsertRecordButton(**char* lpszTemplateName**)**
功能：

插入数据记录

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressLastButton(**char* lpszTemplateName**)**
功能：

在制表窗口的显示区域中显示最后一个数据记录。显示的记录的编号基于组态的时间范围

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressNextButton(**char* lpszTemplateName**)**
功能：

在制表窗口的当前显示区域后面显示数据记录。显示的记录的编号基于组态的时间范围

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressNextItemButton(**char* lpszWindowName**)**
功能：

制表窗口的列向左移动一列，左列的位置被右列取代。

参数：

lpszWindowName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressOpenArchiveVariableSelectionDlgButton(**char* lpszWindowName**)**
功能：

打开一个方框，用来将表格的列连接到每个归挡名称和标签上。

参数：

lpszWindowName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressOpenDlgButton(**char* lpszTemplateName**)**
功能：

制表窗口的在线组态对话框出现

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL TlgTableWindowPressOpenItemSelectDlgButton(char* lpszWindowName)
功能：

打开用于选择可见列的对话框。

参数：

lpszWindowName　=在应用软件窗口名称上的指针

BOOL TlgTableWindowPressOpenTimeSelectDlgButton(char* lpszWindowName)
功能：

打开用于将表格列连接到显示数值的每个时间范围上的对话框。

参数：

lpszWindowName　=在应用软件窗口名称上的指针

BOOLTlgTableWindowPressPrevButton(char* lpszTemplateName)
功能：

在制表窗口的当前显示区域后面显示数据记录。显示的记录的编号基于组态的时间范围

参数：

lpszTemplateName　=在应用软件窗口名称上的指针

BOOL TlgTableWindowPressPrevItemButton(char* lpszWindowName)
功能：

制表窗口的列向右移动一列，右列的位置被左列取代。

参数：

lpszWindowName　=在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressRemoveRecordButton(**char* lpszTemplateName**)**
功能：

删除数据记录

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTableWindowPressStartStopButton(**char* lpszTemplateName**)**
功能：

制表窗口的更新被打开/关闭（切换函数）

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL**TlgTrendWindowPressFirstButton(**char* lpszTemplateName**)**
功能：

在趋势窗口的显示区域中显示第一个数据记录。显示的记录的编号基于组态的时间范围

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressHelpButton(**char* lpszTemplateName**)**
功能：

显示趋势窗口的在线帮助

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressLastButton(**char* lpszTemplateName**)**
功能：

在趋势窗口的显示区域中显示最后一个数据记录。显示的记录的编号基于组态的时间范围

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressLinealButton(**char* lpszTemplateName**)**
功能：

打开/关闭趋势窗口的标尺（切换函数）。标尺可以通过左右光标键移动。

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressNextButton(**char* lpszTemplateName**)**
功能：

在趋势窗口的当前显示区域后面显示数据记录。显示的记录的编号基于组态的时间范围

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressNextItemButton(**char* lpszWindowName**)**
功能：

将趋势窗口中所有趋势向前移动一个层面。前面的趋势向后移动。

参数：

lpszWindowName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressOneToOneButton(**char* lpszTemplateName**)**
功能：

趋势窗口将回复到放大镜打开前的状态。放大镜被关闭。只能用鼠标选择比例缩放区域。参见TlgTrendWindowPressZoomInButton

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressOpenArchiveTagSelectionDlg Button(**char* lpszWindowName**)**
功能：

打开一个方框，用来将趋势窗口的趋势连接到每个归挡名称和标签上。

参数：

lpszWindowName　=在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressOpenDlgButton(**char* lpszTemplateName**)**
功能：

趋势窗口的在线组态对话框出现

参数：

lpszTemplateName　=在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressOpenItemSelectDlgButton(**char* lpszWindowNumber**)**
功能：

打开用于选择可见趋势和将在前面显示的趋势的对话框。

参数：

lpszWindowName　=在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressOpenTimeSelectDlgButton(**char* lpszWindowNumber**)**
功能：

打开用于将趋势连接到显示数值的每个时间范围上的对话框。

参数：

lpszWindowName　=在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressPrevButton(**char* lpszTemplateName**)**
功能：

在趋势窗口的当前显示区域后面显示数据记录。显示的记录的编号基于组态的时间范围

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressPrevItemButton(**char* lpszWindowName**)**
功能：

将趋势窗口中所有趋势向后移动一个层面。后面的趋势向前移动。

参数：

lpszWindowName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressStartStopButton(**char* lpszTemplateName**)**
功能：

趋势窗口的更新被打开/关闭（切换函数）

参数：

lpszTemplateName =在应用软件窗口名称上的指针

BOOL **TlgTrendWindowPressZoomInButton(**char* lpszTemplateName**)**
功能：

放大镜被打开。只能用鼠标选择比例缩放区域。

参数：

lpszTemplateName =在应用软件窗口名称上的指针

3.4 WINCC(标准函数)

void **OnErrorExecute(**CCAPErrorExecute ErrorExecute**)**
功能：

当执行动作或函数时产生错误，调用出错执行。这使你能够执行自己的错误处理，并确定错误的原因。

参数：

ErrorExecute = 告知你有关产生的错误的结构。

结构定义：

```
typedef struckt{
    DWORD dwCurrentThreadID;        //当前线程中的线程ID
    DWORD dwErrorCode1;             //错误代码1
    DWORD dwErrorCode2;             //错误代码2
    BOOL bCycle;                    //周期/非周期
    char* szApplicationName;        //应用软件的名称
    char* szFunctionName;           //函数的名称
    char* szTagName;                //标签的名称
    LPVOID lpParam;                 //动作堆栈的指针
    DWORD dwParamSize;              //动作堆栈的尺寸
    DWORD dwCycle;                  //变量的周期
    CMN_ERROR* pError;              //CMN_ERROR的指针
    }CCAPErrorExecute;
```

单个错误标记的含义，以及基于它们的被传送结构单元的含义，在下列表格中描述：

dwErrorCode1	dwErrorCode2	bCycle	szApplicationName	szFunctionName	szTagname	lpParam	dwParamSize	dwCycle	pError	描述
1007001	0	x	x	x		x	x			动作中的例外
1007001	1	x	x	x		x	x			访问返回结果时的例外
1007001	4097	x	x	x		x	x			执行动作时的堆栈溢出
1007001	4098	x	x	x		x	x			在动作中以0区分
1007001	4099	x	x	x		x	x			访问不在动作中的符号
1007001	4100	x	x	x		x	x			在动作中访问超出
1007004	0	x	x	x						未知的的函数
1007005	1	x	x							动作包含无 P-代码
1007005	2	x	x							函数名称错误
1007005	4	x	x	x		x	x			返回值类型无效
1007005	32768ff	x	x	x		x	x			装载动作时Ciss编译器中的错误
1007006	0	x	x	x	x	x	x	x		没有定义标签
1007006	1	x	x	x	x	x	x	x		标签超时
1007006	2	x	x	x	x	x	x	x	x	在期望的格式中不能提供标签
1007006	3	x	x	x	x	x	x	x	x	标签表明状态超出，状态位于 CMN_ERROR.dwError1
1007007	1	x	x	x		x	x		x	PDLRTGetProp上的错误
1007007	2	x	x	x		x	x		x	PDLRTSetProp上的错误
1007007	3	x	x	x		x	x		x	DM-调用上的错误

3.5 WINDOWS(标准函数)

unsigned int **ProgramExecute(**char* Programm_Name**)**
功能：

用提供的名称启动程序。

参数：

Program_Name =程序名称的指针。

3.6 选项(标准函数)

BOOL **GetCSigPicture(**LPCSTR lpcPictureName, LPCSTR lpcObjectName, LPCSTR lpcPropertyName, LPSTR lpPictureName**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

双字**GetSignificantMask(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

BOOL **PASSCheckAreaPermission(**LPCTSTR areaname**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=有权限存在
FALSE=无权限存在

功能：

确定用户是否产生用于操作指定系统的权限。

参数：

areaname =系统名称

BOOL **PASSCheckAreaLevelPermission(**LPCTSTR areaname, DWORD level**)**
此函数是高级过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=有权限存在
FALSE=无权限存在

功能：

检查用户是否具有当授权级别被传送时用于已传送系统的权限。

参数：

areaname =系统名称
level =作为编号数值的授权级别

void **PASSLoginDialog(**TCHAR ch**)**
此函数是基本过程控制选项软件包的一部分。

功能：

显示登录对话框，并在登录成功时将用户数据下载至共享存储器。

参数：

ch =登录用的监视器的规格
对话框将打开(对于监视器1，输入“1”字符）。

3.7 报告(标准函数)

void **ReportJob(**LPSTR lpJobName, LPSTR lpMethodName**)**
此函数将被函数RPTJobPreview和RPTJobPrint替代，因此不能继续使用。

功能：

基于参数值lpMethodName，启动打印作业或打印预览。

参数：

lpJobName = 在打印作业上的指针
lpMethodName = 在函数模式上的指针
“打印”打印作业将被执行
“预览”打印预览被启动

BOOL **RPTJobPreview(**LPSTR lpJobName**)**
功能：

启动打印作业的预览。

参数：

lpJobName = 在打印作业名称上的指针

BOOL **RPTJobPrint(**LPSTR lpJobName**)**
功能：

启动打印作业。

参数：

lpJobName = 在打印作业名称上的指针

3.8 分割画面管理器(标准函数)

BOOL **AcknowledgeAllPicture(**LPCTSTR pictName**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

void **GetASVarIndex(**int nVarIndex, int nClassIndex, BOOL* bActive, BOOL* bQuit**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

int **GetIndexFromMask(**DWORD dwMask**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

void **GetMessageClassFromVar(**TCHAR* VarName, int nClassIndex, BOOL* bActive, BOOL* bQuit**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

BOOL **GetCountPicture(**LPTSTR aktName, LPTSTR newName, int nNewNameLength, DWORD dwBitMask**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

BOOL **LoopInAlarm(**TCHAR* TagName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：
出错代码
TRUE=有权限存在
FALSE=无权限存在

功能：
此函数显示提供用于标签名称得原型图形，以及连接到标签名称上的原型图形。

参数：
TagName =标签名称的指针
void **profile(**long pos_no, long value, LPCTSTR lpszPictureName**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

BOOL **PTMUnload()**
此函数是基本过程控制选项软件包的一部分。

功能：

此函数卸载图形树管理器。

只在内部使用。

void **reset_hoer(**Tag var**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

void **SetASVarIndex(**int nVarIndex, int nClassIndex, BOOL bActive, BOOL bQuit**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

void **SetMessageClassToVar(**TCHAR* VarName, int nClassIndex, BOOL bActive, BOOL bQuit**)**
此函数是基本过程控制选项软件包的一部分。

只在内部使用。

BOOL **SFCLoopInAlarm(**TCHAR* TagInfo**)**
此函数是高级过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=函数被执行
FALSE=函数没有被执行，或输入了错误的参数

功能：

此函数打开在度量点上触发信息的SFC计划的浏览。

参数：

TagInfo = 信息缓冲器的指针关于度量点被存储

BOOL **SSMChangeButtonField(**char Screen, char* PictureName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在按钮区显示被改变
FALSE=错误，显示没有被改变

功能：

名为lptPictureName的图形显示在以画面描述的监视器的按钮区。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 名称的指针，包括扩展名将在按钮区显示的图形

BOOL **SSMChangeOverviewField(**char Screen, char* PictureName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在总览区显示被改变
FALSE=错误，显示没有被改变

功能：

名为lptPictureName的图形显示在以画面描述的监视器的总览区。

参数：

Screen = 包含作为字符串的监视器编号

PictureName = 名称的指针，包括扩展名
将在总览区显示的图形

BOOL **SSMChangeWorkField(**char Screen, char* PictureName, BOOL Store**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在总览区显示被改变
FALSE=错误，显示没有被改变

功能：

文件的内容显示在以画面描述的监视器的工作空间。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 名称的指针，包括扩展名
将被显示的PDL文件。如果名称被指定
在PictureName 中，包括在体系中
“图形树管理器”，区域名为
自动传送和保存。你可以访问区域名，通过
“SSMGetAreaFromPicturePath”函数。
如果在PictureName 指定的文件不在体系中，
（图形原型），你可以设置区域名，通过
调用SSMRTSetAreaToPicturePath函数。
Store = 如果TRUE，更新图形存储。（缺省值设置）

BOOL **SSMCheckWorkFieldDown(**char Screen**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在下面第一个节点中的图形存在
FALSE=错误或在下面第一个节点中的图形不存在。

功能：

此函数确定在画面工作空间中哪个图形被显示，并图形是否存在于“图形树管理器”体系的相关下部节点中。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMCheckWorkFieldLeft(**char Screen**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在左面节点中的图形存在
FALSE=错误或左面节点中的图形不存在。

功能：

此函数确定在画面工作空间中哪个图形被显示，并图形是否存在于“图形树管理器”体系的相关左面节点中。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMCheckWorkFieldRight(**char Screen**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在右面节点中的图形存在
FALSE=错误或右面节点中的图形不存在。

功能：

此函数确定在画面工作空间中哪个图形被显示，并图形是否存在于“图形树管理器”体系的相关右面节点中。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMCheckWorkFieldUp**(char Screen)
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在上面下一个节点中的图形存在
FALSE=错误或在上面下一个节点中的图形不存在。

功能：

此函数确定在画面工作空间中哪个图形被显示，并图形是否存在于“图形树管理器”体系的相关上部节点中。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMChgWorkFieldDown**(char Screen)
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在工作空间显示被改变
FALSE=错误，在工作空间显示没有被改变。

功能：

此函数决定哪个图形显示在监视器工作空间中。从“图形树管理器”体系的下面第一个节点得到适当的图形，并且在画面工作空间显示此新图形。图形堆栈自动更新。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMChgWorkFieldLeft(**char Screen**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在工作空间显示被改变
FALSE=错误，在工作空间显示没有被改变。

功能：

此函数决定哪个图形显示在画面工作空间中。从“图形树管理器”体系的左面节点得到适当的图形，并且在画面工作空间显示此图形。图形堆栈自动更新。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMCheckWorkFieldRight(**char Screen**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在工作空间显示被改变
FALSE=错误，在工作空间显示没有被改变。

功能：

此函数决定哪个图形显示在画面工作空间中。从“图形树管理器”体系的右面节点得到适当的图形，并且在画面工作空间显示此图形。图形堆栈自动更新。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMCheckWorkFieldUp**(char Screen)
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=在工作空间显示被改变
FALSE=错误，在工作空间显示没有被改变。

功能：

此函数决定哪个图形显示在画面工作空间中。从“图形树管理器”体系的上面第二个节点得到适当的图形，并且在画面工作空间显示此图形。
图形堆栈自动更新。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMDeleteUserSettings**(LPCTSTR lpctUserName)
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=相关用户设置被删除。
FALSE=错误

功能：

使用此函数删除所有在标准文件夹lpctUserName中由lpctUserName创建的“分离画面管理器”中的所有文件。属于其它用户或其它编辑器的文件不受影响。
属于当前用户的，在其它文件夹中的文件也不受影响。

参数：

lpctUserName = 设置被删除的用户名的指针

BOOL **SSMGetAreaFromPath(**char* PicturePath, char* AreaName, int len**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=相关用户设置被删除。
FALSE=错误

功能：

此函数得到指定图形路径的系统分配。此函数不使用“图形树”。图形路径必须包含在工作空间或顶部区域中显示的图形的完整路径。

参数：

PicturePath = 图形名称的指针，包括路径。你不用输入扩展名。
AreaName = 存储区域/系统名称的缓冲器的指针。
len = AreaName缓冲器的最大长度

BOOL **SSMGetAreaFromWorkField(**char Screen, char* AreaName, int nAreaNameLen**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数得到在指定的监视器中的工作空间的系统分配。

参数：

Screen = 包含作为字符串的监视器编号
AreaName = 存储区域/系统名称的缓冲器的指针。
nAreaNamelen = AreaName缓冲器最大长度

BOOL **SSMGetAutoLoadSettings()**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=自动装载激活
FALSE=错误

功能：

此函数确定用户登录后是否自动装载用户指定的设置到运行系统中。

BOOL **SSMGetContainerToPicture(**char* PictureName, char* ReturnContainer, int len**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到容器
FALSE=错误

功能：

此函数使用图形树管理器来得到适用于指定图形的容器。

参数：

PictureName = 图形名称
ReturnContainer = 存储容器名称的缓冲器的指针。
len = ReturnContainer缓冲器的最大长度

BOOL **SSMGetContPict(**int area_no, int subarea_no, char* ReturnPictureName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到图形名称
FALSE=错误

功能：

此函数得到由系统编号和子系统编号指定的容器的图形名称。如果子系统编号为 0，函数得到系统容器的图形名称。

参数：

area_no = 系统编号
subarea_no = 子系统编号
ReturnPictureName = 存储图形名称的缓冲器的指针。

BOOL **SSMGetContainer(**int area_no, int subarea_no, char* ReturnName**)**此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到容器名称
FALSE=错误

功能：

此函数得到由系统编号和子系统编号指定的容器的容器名称。如果子系统编号为 0，系统得到系统容器的容器名称。

参数：

area_no = 系统编号
subarea_no = 子系统编号
ReturnName = 存储容器名称的缓冲器的指针。

BOOL **SSMGetRootToPicture(**char* PictureName, char* ReturnRootContainer**)**此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE = 得到系统容器名称
FALSE = 错误

功能：

此函数使用图形名称来得到系统容器名称。

参数：

PictureName = 图形名称
ReturnRootContainer = 存储系统容器名称的缓冲器的指针。

char **SSMGetScreen(**char* lpszPictureName**)**
此函数是基本过程控制选项软件包的一部分。

功能：
此函数指定显示选择的图形的监视器。

参数：
lpszPictureName = 将被检查的图形的指针

BOOL **SSMGetWorkFieldPicture(**char Screen, char* ReturnPictureName, int len**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到图形名称
FALSE=错误

功能：

此函数得到当前在画面工作空间中显示的图形名称（包括扩展名）。

参数：

Screen = 包含作为字符串的监视器编号
ReturnPictureName = 存储图形名称缓冲器的指针。
len = ReturnPictureName缓冲器的长度

BOOL **SSMGetWorkFieldCoordinates(**TCHAR cMonitor, int* pLeft, int* pTop, int* pWidth, int* pHeight**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到工作空间的大小和位置。
FALSE=错误

功能：

此函数得到用于cMonitor监视器工作空间的定位和位置，并将它装载到pLeft，pTop，pWidth和pHeight参数。

参数：

cMonitor = 包含作为字符串的监视器编号。
pLeft = 装载工作空间的X坐标的存储位置的指针。
pTop = 装载工作空间的Y坐标的存储位置的指针。
pWidth = 装载工作空间宽度的存储位置的指针。
pHeight = 装载工作空间高度的存储位置的指针。

void **SSMGetWorkFieldPath(**char Screen, char* ReturnBaseName, int Length**)**
此函数是基本过程控制选项软件包的一部分。

功能：

此函数从指定监视器得到位于工作空间的图形的完整路径。

参数：

Screen = 包含作为字符串的监视器编号
ReturnBaseName = 装载路径规定的按钮的指针
Length = ReturnBaseName缓冲器的长度

BOOL **SSMLoadCurrentFields(**LPCTSTR lpctSettingsName, LPCTSTR lpctUserName)此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到路径设置
FALSE=错误

功能：

函数从在lpctSettingsName中指定的文件中装载用户指定信息，其长度等于在lpctUserName中指定的，相当于存储在文件中的用户名的名称长度。下列信息被装载，用于每个监视器。

在总览区域中的图形名称
在工作空间中的图形名称
在按钮区中的图形名称
在此处所有在过程显示窗口中显示的图形的名称和位置（X、Y、宽度、高度）。
图形堆栈
图形存储
在此处所有可见区域的名称和位置(X、Y、宽度、高度)

参数：

lpctSettingsName = 存储用户指定信息的文件名称的指针。
如果lpctSettingsName不包含扩展名，
扩展名“.SSM”被附加。如果
lpctSettingsName不包含路径规定 函数搜索它
在lpctUserName用户的SSM标准文件夹中。
如果lpctSettingsName包含相对或绝对的路径规定，路径被使用不作任何改变（不推荐）。

lpctUserName = 设置被装载的用户名称的指针。
如果在lpctUserName中没有输入用户名（ZERO或空字符串），
当前登录的用户名称被使用

BOOL **SSMLoadSettings(**TCHAR cMonitor**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到路径设置
FALSE=错误

功能：

此函数在对话框窗口显示用户指定的设置。下列信息被指定，用于指定的监视器：

在总览区域中的图形名称
在工作空间中的图形名称
在按钮区中的图形名称
在此处所有在过程显示窗口中显示的图形的名称和位置(X，Y，宽度，高度)。
图形堆栈
图形存储
在此处所有可见区域的名称和位置(X，Y，宽度，高度)

参数：

cMonitor = 作为字符串的监视器编号选择对话框被显示在字符串上。

BOOL **SSMOpenSpecField(**char Screen, char* FieldName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数在指定监视器上打开特殊区域。

参数：

Screen = 包含作为字符串的监视器编号
FieldName = 存储特殊区域名称的缓冲器的指针。

BOOL **SSMOpenTopField(**char Screen, char* PictureName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=打开的过程显示窗口
FALSE=错误或不存在顶部区域

功能：

此函数打开变量大小的过程显示窗口，并使用它来显示由PictureName指定的图形。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 将被显示在过程显示窗口中的图形名称的指针。

BOOL **SSMOpenTopFieldFixedSize(**char Screen, char* PictureName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=打开的过程显示窗口
FALSE=错误

功能：

此函数打开设置大小的过程显示窗口，并使用它来显示由PictureName指定的图形。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 将被显示在过程显示窗口中的图形名称的指针。

BOOL **SSMPictureStoreGet(**char Screen, char* PictureName, int len**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=已编辑的图形存储
FALSE=错误

功能：

此函数从指定监视器的图形存储中装载图形名称。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 存储在图形存储中的图形名称的指针。
len = PictureName缓冲器的长度

int **SSMPictureStoreNum(**char Screen**)**
此函数是基本过程控制选项软件包的一部分。

功能：

此函数指定位于图形存储中的图形的编号。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMPictureStoreSet(**char Screen, char* PictureName**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=已编辑的图形存储
FALSE=错误

功能：

此函数将图形名称存储在指定监视器的图形存储中。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 存储在图形存储中的图形名称的指针。

unsigned int **SSMProgramExecute(**char Screen, char * szCommandLine**)**
此函数是基本过程控制选项软件包的一部分。

功能：

此函数在指定监视器上启动应用软件。

参数：

Screen = 包含作为字符串的监视器编号
szCommandLine = 命令行（程序名称包括参数)

BOOL **SSMPictureMemoryInquire(**char Screen, char* PictureName, int len, int* ReturnCount**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数得到图形堆栈中顶部存储区域的图形名称。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 存储在图形堆栈中的图形名称的指针。
len = PictureName缓冲器的长度
ReturnCount = 在监视器画面图形存储中的图形(单元)编号的指针。

int **SSMPictureMemoryNum(**char Screen**)**
此函数是基本过程控制选项软件包的一部分。

功能：

此函数指定在指定监视器图形堆栈中放置多少图形。在图形堆栈中最多可存储 8个图形。

参数：

Screen = 包含作为字符串的监视器编号

BOOL **SSMPictureMemoryStore(**char Screen, char* PictureName, int* ReturnCount**)**此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数将图形从适当的监视器中保存到图形堆栈中。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 存储在图形堆栈中的图形名称的指针。
ReturnCount = 监视器图形堆栈的图形（元素）的号码指针。

BOOL **SSMPictureMemoryRestore(**char Screen, char* PictureName, int len, int* ReturnCount**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数将图形从图形堆栈中装载到适当的监视器中。然后存储器又存在了。

参数：

Screen = 包含作为字符串的监视器编号
PictureName = 存储在图形存储中的图形名称的指针。
len = 缓冲器的长度
ReturnCount = 监视器Screen图形堆栈中图形编号（单元）的指针。

BOOL **SSMSetAreaNameToPicture(**char* PicturePath, char* AreaName**)**此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数为图形路径设置区域名称。区域名称不由函数测试，但是被直接存储。

参数：

PicturePath = 图形名称的指针，包括路径设置。不需要给扩展名。
AreaName = 连接到图形路径的区域名称的指针

BOOL **SSMSetLanguage(**DWORD dwLanguage**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数设置在运行方式中使用的语言。

参数：

dwLanguage = 作为编号值在运行中使用的语言

BOOL **SSMStoreCurrentFields(**LPCTSTR lpctSettingsName, LPCTSTR lpctUserName**)**此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到路径设置
FALSE=错误

功能：

此函数创建在lpctSettingsName中指定的文件，并将用户指定的信息保存在内。下列信息被存储，用于每个监视器。

在总览区域中的图形名称
在工作空间中的图形名称
在按钮区中的图形名称
在此处所有在过程显示窗口中显示的图形的名称和位置(X，Y，宽度，高度)。
图形堆栈
图形存储
在此处所有可见区域的名称和位置(X，Y，宽度，高度)

参数：

lpctSettingsName = 存储用户指定信息的文件名称的指针。如果在lpctSettingsName中指定的文件已经存在，它被覆盖。没有关于文件名称的限制。如果你提供扩展名，它必须是“.SSM”。如果lpctSettingsName不包含扩展名，扩展名“.SSM”被增加。如果lpctSettingsName不包含路径规定，文件被设置在lpctUserName用户的SSM标准文件夹中。如果lpctSettingsName包含相对或绝对的路径规 定，使用路径时不作任何改变（不推荐）。

lpctUserName = 设置被装载的用户名的指针如果在lpctUserName中没有输入用户名（ZERO或空字符串），当前登录的用户名称被使用

BOOL **SSMStoreSettings(**TCHAR cMonitor**)**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=得到路径设置
FALSE=错误

功能：

此函数将用户指定设置存储在通过选择对话框选择的文件中。
下列信息被存储，用于指定的监视器：

在总览区域中的图形名称
在工作空间中的图形名称
在按钮区中的图形名称
在此处所有在过程显示窗口中显示的图形的名称和位置(X、Y、宽度、高度)。
图形堆栈
图形存储
在此时所有可见区域的名称和位置(X、Y、宽度、高度)

参数：

cMonitor = 作为显示选择对话框的字符串的监视器编号。

BOOL **SSMUnload()**
此函数是基本过程控制选项软件包的一部分。

返回值：

出错代码
TRUE=无错误
FALSE=产生错误

功能：

此函数卸载分离画面管理器。

只在内部使用。

4 内部函数

使用内部函数可使图形对象和归档动态化。也可将其用在项目函数、标准函数、和全局脚本动作中。
内部函数分成下列各种类别：

报警
包含了启动与停止接收单个信息以及指定过滤器等函数。

分配
包含了保留和激活工作存储器等函数。

c_bib
包含了源于C语言标准库的函数。

图形
包含了读出和设置图形对象属性的函数。

标签
包含了写入和读出过程标签的函数。

视窗控制中心
包含了语言转换、停止运行和退出视窗控制中心等函数。

4.1 报警 (内部函数)

BOOL **MSRTSetMsgFilter (**DWORD dwServiceID,
LPMSG_FILTER_STRUCT lpMsgFilter, LPCMN_ERROR lpError**);**
功能:

为所显示的服务设置新的过滤器。

参数：

dwServiceID = 服务，用于将要设置的过滤器
lpMsgFilter = 要使用的过滤器
lpError = 获得扩充的指针
错误信息

BOOL **MSRTStartMsgService (**LPDWORD lpdwServiceID,
MSG_SERVICE_NOTIFY_PROC lpfnNotifyProc,
LPMSG_FILTER_STRUCT lpMsgFilter, DWORD dwNotifyMask,
LPVOID lpvUser, LPCMN_ERROR lpError**);**
功能:

启动服务以便接收单个信息。可创建信息窗口。

参数：

lpdwServiceID = 包含了函数后面的服务识别码已被成功的调用。
lpfnNotifyProc = 通知函数，它用于将信息传输给服务
lpMsgFilter = 过滤条件的指针：(NULL=所有信息)
dwNotifyMask = 所发送信息的详细说明。
lpvUser = 发送给回叫信号的用户数据。
lpError = 获得扩充的指针错误信息

BOOL **MSRTStopMsgService (**DWORD dwServiceID,
LPCMN_ERROR lpError**);**
功能:

停止用于接收单个信息的服务。

参数：

dwServiceID = 要停止的服务的ID号码
lpError = 获得扩充的指针错误信息

4.2 分配

“分配”函数类包含了保留和激活工作存储器的函数。

void **SysFree(**void* lpFree**);**
功能:

用“**SysMalloc**”函数释放所保留的存储器。

参数：

lpFree =指针

void* **SysMalloc(**unsigned long int size**);**
功能:

为动作保留存储器。将存储器范围分配给动作。当动作完成并且结果被送出时，系统释放存储器。使用“**SysFree**”函数可自己释放存储器。

实例：
保留用于返回的动作的value的存储器。

```
char* main(...);
{
char* returnwert;
char text[17];
returnwert=SysMalloc(17);
strcpy(returnwert,&text[0];
return returnwert;
}
```

参数：

size =以BYTE 为单位的存储器的尺寸

4.3 c_bib

函数类c_bib包含了来自于C语言库的C语言函数且分成下列几个区域：

字符类型
数学符号
存储器
标准IO
标准库
字符串
时间

标准IO本身又分成下列几个区域：

字符IO
直接IO
错误
文件
文件位置
输出

在相关的技术文献中可找到有关该函数的描述。

4.4 图形

图形函数类分成下列几类：

获取函数

该类函数用于发送属性值。获取函数本身分成下列函数组：

axes
color
fill
flash
focus
font
general
geometry
i_o
limits
link
miscs
ole_control
pictures
property
state
style

获取函数需要下述的两个参数：

lpszPictureName：图象名称(不带扩展名pdl)
lpszObjectName：对象名

设置函数

该类函数用于设置属性值。设置函数本身分成下列函数组：

axes
color
fill
flash
focus
font
geometry
i_o
limits
link
miscs
ole_control
pictures
property
state
style

设置函数需要下述的三个参数：

lpszPictureName：图象名称(不带扩展名pdl)
lpszObjectName：对象名
3.参数：属性值

设置函数的返回值是下述的错误代码：
TRUE：函数执行时无错误。
FALSE：有错误产生。

4.4.1 图形获取轴线函数

BOOL **GetAlignment (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

TRUE：文本在棒图的右边。
FALSE：文本在棒图的左边。

double **GetAxisSection (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

轴线区段：在两个邻近轴线标志之间的值的差。

BOOL **GetExponent (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

TRUE表示坐标轴上的数字是指数格式。
FALSE表示坐标轴上的数字是小数格式。

long int **GetLeftComma (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

坐标轴上的小数点左边的数位。

BOOL **GetLongStrokesBold (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

TRUE表示以粗体字显示棒图标尺上的长轴部分。
FALSE表示正常显示棒图标尺上的长轴部分。

BOOL **GetLongStrokesOnly (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

TRUE表示只有长轴部分才在棒图标尺上显示。
FALSE表示长轴部分和其子段均在棒图标尺上显示。

long int **GetLongStrokesSize(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

控制长轴部分的长度的编号值

long int **GetLongStrokesTextEach (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

控制长轴部分的文本标签的编号值

实例：
返回值=1->单个长轴线区段被标记。
返回值=2->各个其它长轴线区段被标记。

long int **GetRightComma (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

坐标轴上的小数点右边的数位。

long int **GetScaleTicks (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

以棒图整个高度的某个百分比的值来标记标尺。

BOOL **GetScaling (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

TRUE表示某个附加标尺。
FALSE表示不带有附加标尺。

long int **GetScalingType(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值

确定棒图定标类型的编号值
0： 线性的(同一权重)
1： 对数的(侧重于低限值)
2： 负对数的(侧重于高限值)
3： 自动的(线性的)
4： 正切的(侧重于高限和低限值)
5： 平方的(侧重于高限值)
6： 立方的(侧重于高限值)

4.4.2 图形获取颜色函数

long int **GetBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

对象背景颜色的编号值

注意:
如果函数调用与整个图象有关，则设置参数lpszObjectName = 空。

long int **GetBackColor2 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象的棒图颜色的编号值

long int **GetBackColor3 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

用于棒图对象的棒图背景颜色的编号值

long int **GetBackColorBottom(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

用于设置滑块对象的底部右边的背景颜色的编号值。

long int **GetBackColorTop(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

用于设置滑块对象的顶部左边的背景颜色的编号值。

long int **GetBorderBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

线条或边框背景颜色的编号值

long int **GetBorderColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

线条或边框颜色的编号值

long int **GetBorderColorBottom (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

用于三维边框(阴影)的右部和底部颜色的编号值

long int **GetBorderColorTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

用于三维边框的左部和顶部颜色的编号值

long int **GetButtonColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

滑块对象上的按钮颜色的编号值

long int **GetColorBottom (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

底部/右滑块位置颜色的编号值(低限)

long int **GetColorTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

顶部/左滑块位置颜色的编号值(高限)

long int **GetFillColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

填充模式颜色的编号值
注意:
如果函数调用与整个图象相关，则设置参数lpszObjectName =空。

long int **GetForeColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

字体颜色的编号值

long int **GetGridColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图形编辑器工作空间的网格颜色的编号值

long int **GetItemBorderBackColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“文本列表”对象中的分隔线背景颜色。

long int **GetItemBorderColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“文本列表”对象中的分隔线颜色。

long int **GetScaleColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的标尺颜色的编号值

long int **GetSelBGColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“文本列表”对象中浏览器列表里的所选条目的背景颜色。

long int **GetSelTextColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“文本列表”对象中浏览器列表里的所选条目的字体颜色。

long int **GetTrendColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的趋势颜色的编号值

long int **GetUnselBGColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“文本列表”对象中浏览器列表里的未选条目的背景颜色。

long int **GetUnselTextColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“文本列表”对象中浏览器列表里的未选条目的文本颜色。

4.4.3 图形获取填充函数

BOOL **GetFilling (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示背景颜色的动态填充属性是激活的。具有实心边界(长方形、圆角长方形、圆形、椭圆形、饼图形、扇形、多边形、文本、输入/输出域、及其它)的对象可被填充。
FALSE表示背景颜色的动态填充属性不是激活的。

long int **GetFillingIndex (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当前填充量(百分比)的编号值

4.4.4 图形获取闪烁函数

long int **GetBackFlashColorOff (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

闪烁属性关闭时背景颜色的编号值

long int **GetBackFlashColorOn (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当闪烁属性作为编号值打开时的背景颜色

long int **GetBorderFlashColorOff (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当闪烁属性关闭时边框颜色或线条颜色的编号值

long int **GetBorderFlashColorOn (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当闪烁属性打开时边框颜色或线条颜色的编号值

BOOL **GetFlashBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示背景闪烁是激活的。
FALSE表示背景闪烁没有激活。

BOOL **GetFlashBorderColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示边框或线条闪烁是激活的。
FALSE表示边框或线条闪烁没有激活。

BOOL **GetFlashForeColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示文本闪烁是激活的。
FALSE表示文本闪烁没有激活。

long int **GetFlashRateBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

背景闪烁频率。

long int **GetFlashRateBorderColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

线条或边框闪烁频率。

long int **GetFlashRateForeColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

文本闪烁频率。

long int **GetForeFlashColorOff (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当闪烁属性关闭时文本颜色的编号值

long int **GetForeFlashColorOn (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当闪烁属性打开时文本颜色的编号值

4.4.5 图形获取焦点函数

char* **Get_Focus();**
返回值：

对象名放置在中心上或已被放置在最后

4.4.6 图形获取字体函数

long int **GetAlignmentLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

文本水平对齐 (左对齐、居中、右对齐)的编号值

long int **GetAlignmentTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

文本垂直对齐 (顶部对齐、居中、底部对齐)的编号值

BOOL **GetFontBold (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示“粗体字”文本属性已设置。
FALSE表示“粗体字”文本属性未设置。

BOOL **GetFontItalic (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示“斜体字”文本属性已设置。
FALSE表示“斜体字”文本属性未设置。

char* **GetFontName (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当前设置字体名称的指针。

long int **GetFontSize (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当前字体尺寸
lpszPictureName =图象名
lpszObjectName =对象名

BOOL **GetFontUnderline (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示“下划线”文本属性已设置。
FALSE表示“下划线”文本属性未设置。

BOOL **GetOrientation (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示文本定位是水平的。
FALSE表示文本定位是垂直的。

char* **GetText (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

静态文本的指针。对于单选和复选框、以及多边形和折线，必须在调用**获取文本**函数以前用**设置下标**函数设置元素或点。

4.4.7 图形获取常规函数

long int **GetLayer (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

提供对象所在的图象层。

4.4.8 图形获取几何函数

long int **GetActualPointLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

与图象原点相关的多边形和折线对象类型的角度点的当前水平位置 (X 轴)。
设置下标函数设置多边形的当前点。

long int **GetActualPointTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

与图象原点相关的多边形和折线对象类型的角度点的当前垂直位置(Y 轴)。
设置下标函数设置多边形的当前点。

long int **GetBoxCount (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

在某个选项组区域中的复选框区或选项按钮上的所选复选框的编号。

long int **GetDirection (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象里的棒图方向(向上、向下、向左、或向右)的编号值

long int **GetEndAngle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

饼图和扇形以及圆弧和椭圆弧的终端角

BOOL **GetGrid (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图形编辑器的工作空间里的网格开通。
FALSE表示图形编辑器的工作空间里的网格关闭。

long int **GetGridHeight (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图形编辑器的工作空间里的网格的高度

long int **GetGridWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图形编辑器的工作空间里的网格的宽度

long int **GetHeight (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

包围某个对象的长方形的高度

注意:
如果函数的调用与整个图象相关，则lpszObjectName参数必须=空。

long int **GetLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

包围某个对象的长方形的左上角在X轴上的当前位置
long int **GetPointCount (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

多边形或折线的角度的编号

long int **GetRadius (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

圆形、饼图形、或圆弧的半径

long int **GetRadiusHeight (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

椭圆、扇形、或椭圆弧的垂直半径

long int **GetRadiusWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

椭圆、扇形、或椭圆弧的水平半径

long int **GetReferenceRotationLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于直线、多边形和折线
返回值：

旋转的参考X值(对象旋转的参考点)

long int **GetReferenceRotationTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于直线、多边形和折线。
返回值：

旋转的参考Y值(对象旋转的参考点)

long int **GetRotationAngle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于直线、多边形和折线。
返回值：

对象绕旋转点的顺时针旋转角(度数)

long int **GetRoundCornerHeight (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

圆角长方形的角的垂直半径

long int **GetRoundCornerWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

圆角长方形的角的水平半径

long int **GetStartAngle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

饼图和扇形以及圆弧和椭圆弧的起始角

long int **GetTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

包围某个对象的长方形的左上角的当前Y轴值

long int **GetWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

包围某个对象的长方形的宽度

注意：
如果函数的调用与整个图象相关，则lpszObjectName参数必须=空。

long int **GetZeroPoint (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象里的零点

4.4.9 图形获取I/O函数

char* **GetAssignments(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

文本赋值的值的范围依赖于列表的类型。

BOOL **GetAssumeOnExit (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

TRUE表示系统一旦退出区域就接受输入值。
FALSE表示系统一旦退出区域就不接受输入值。

BOOL **GetAssumeOnFull (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

TRUE表示系统一旦完成输入(所输入字符的指定编号)就自动地退出输入域(不须使用Tab键或回车键)并接受输入。
FALSE表示系统一旦完成输入不会自动地退出输入域和接受输入。

long int **GetBitNumber(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“位”列表类型里的输出值中的相应位。

BOOL **GetClearOnError (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

TRUE表示当无效输入时自动地删除输入域。
FALSE表示当无效输入时不会删除输入域。

BOOL **GetClearOnNew (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

TRUE表示输入域在启动时被删除。
FALSE表示在启动时输入域不会被删除。

long int **GetDataFormat (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

区域内容的数据类型(二进制、十进制、十六进制、或字符串)的编号值

BOOL **GetHiddenInput (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

TRUE表示输入值在输入期间是隐含的。所显示的星号(*)可代表每个字符。
FALSE表示输入值在输入期间不是隐含的。

char* **GetInputValueChar (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

输入值的指针

double **GetInputValueDouble (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

输入值

long int **GetListType(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示文本列表的类型。

可能有下述的列表类型：
0：　　十进制
1：　　二进制
2：　　位

long int **GetNumberLines(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示“文本列表”对象中的浏览器列表里包含了多少可见直线。

注意:
如果所配置文本的编号大于可见直线的编号，则浏览器列表将具有垂直滚动条。

char* **GetOutputFormat (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

输出格式的指针

char* **GetOutputValueChar (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

输出值的指针

double **GetOutputValueDouble (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于输入/输出域
返回值：

输出值

4.4.10 图形获取界限函数

double **GetAlarmHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象里的报警高限

double **GetAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象里的报警低限

BOOL **GetCheckAlarmHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“报警高限”的限位值被监控到。
FALSE表示“报警高限”的限位值没有监控到。

BOOL **GetCheckAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“报警低限”的限位值被监控到。
FALSE表示棒图对象中的“报警低限”的限位值没有监控到。

BOOL **GetCheckLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“保留4”的高限值被监控到。
FALSE表示棒图对象中的“保留4”的高限值没有被监控到。

BOOL **GetCheckLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中“保留5”的高限值被监控到。
FALSE表示棒图对象中的“保留5”的高限值没有被监控到。

BOOL **GetCheckLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“保留4”的低限值被监控到。
FALSE表示棒图对象中的“保留4”的低限值没有被监控到。

BOOL **GetCheckLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“保留5”的低限值被监控到。
FALSE表示棒图对象中的“保留5”的低限值没有被监控到。

BOOL **GetCheckToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“公差高限”的限位值被监控到。
FALSE表示棒图对象中的“公差高限”的限位值没有被监控到。

BOOL **GetCheckToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“公差低限”的限位值被监控到。
FALSE表示棒图对象中的“公差低限”的限位值没有被监控到。

BOOL **GetCheckWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“报警高限”的限位值被监控到。
FALSE表示棒图对象中的“报警高限”的限位值没有被监控到。

BOOL **GetCheckWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中的“报警低限”的限位值被监控到。
FALSE表示棒图对象中的“报警低限”的限位值没有被监控到。

long int **GetColorAlarmHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“报警高限”的限位值达到时棒图颜色的编号值

long int **GetColorAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“报警低限”的限位值达到时棒图颜色的编号值

long int **GetColorLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“保留4”的高限值达到时棒图颜色的编号值

long int **GetColorLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“保留5”的高限值达到时棒图颜色的编号值

long int **GetColorLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“保留4”的低限值达到时棒图颜色的编号值

long int **GetColorLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“保留5”的低限值达到时棒图颜色的编号值

long int **GetColorToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“公差高限”的限位值达到时棒图颜色的编号值

long int **GetColorToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“公差低限”的限位值达到时棒图颜色的编号值

long int **GetColorWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“报警高限”的限位值达到时棒图颜色的编号值

long int **GetColorWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

当“报警低限”的限位值达到时棒图颜色的编号值

double **GetLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“保留4”的高限值

double **GetLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“保留5”的高限值

double **GetLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“保留4”的低限值

double **GetLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“保留5”的低限值

double **GetLimitMax (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

输入/输出域的高限值

double **GetLimitMin (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

输入/输出域的低限值

BOOL **GetMarker (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示限位值显示为棒图对象里的标尺值。
FALSE表示棒图对象中没有显示限位值。

double **GetToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“公差高限”的限位值

double **GetToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“公差低限”的限位值

BOOL **GetTypeAlarmHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“报警高限”的限位值设置为百分比的形式。
FALSE表示在棒图对象中“报警高限”的限位值设置为绝对值。

BOOL **GetTypeAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“报警低限”的限位值设置为百分比形式。
FALSE表示在棒图对象中“报警低限”的限位值设置为绝对值。

BOOL **GetTypeLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“保留4”的高限值设置为百分比形式。
FALSE表示在棒图对象中“保留4”的高限值设置为绝对值。

BOOL **GetTypeLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“保留5”的高限值设置为百分比形式。
FALSE表示在棒图对象中“保留5”的高限值设置为绝对值。

BOOL **GetTypeLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“保留4”的低限值设置为百分比形式。
FALSE表示在棒图对象中“保留4”的低限值设置为绝对值。

BOOL **GetTypeLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“保留5”的低限值设置为百分比形式。
FALSE表示在棒图对象中“保留5”的低限值设置为绝对值。

BOOL **GetTypeToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“公差高限”的限位值设置为百分比形式。
FALSE表示在棒图对象中“公差高限”的限位值设置为绝对值。

BOOL **GetTypeToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“公差低限”的限位值设置为百分比形式。
FALSE表示在棒图对象中“公差低限”的限位值设置为绝对值。

BOOL **GetTypeWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“报警高限”的限位值设置为百分比形式。
FALSE表示在棒图对象中“报警高限”的限位值设置为绝对值。

BOOL **GetTypeWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示在棒图对象中“报警低限”的限位值设置为百分比形式。
FALSE表示在棒图对象中“报警低限”的限位值设置为绝对值。

double **GetWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“报警高限”的限位值

double **GetWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象中的“报警低限”的限位值

4.4.11 图形获取链接函数

BOOL **GetLink(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName,LPCTSTR lpszPropertyName, LPLINKINFO() pLink**);**

返回值：

询问对象属性的当前标签连接

参数：

结构：

标签结构由LinkTyp、dwCycle和szLinkName等参数组成。

LinkType：

0	对象属性不是动态的
1	直接标签连接
2	间接标签连接
3	C语言动作
4	通过动态向导所创建的C语言动作

dwCycle：

0	图象周期
1	窗口周期
2	一旦改变
3	250ms
4	500ms
5	1s
6	2s
7	5s
8	10s
9	1min
10	5min
11	10min
12	1hr
13-17	用户自定义周期1至5

szLinkName：
如果有直接或间接标签连接，则传送标签名称。

实例：

```
{       LINKINFO linkinfo;

        GetLink(.....,&linkinfo);
        printf(„%d,%d,%s“,linkinfo.LinkType,linkinfo.dwCycle,
        linkinfo.szLinkName);
}
```

4.4.12 图形获取其它函数

BOOL **GetAdaptBorder (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于静态文本、输入/输出域、按钮、复选框和选项按钮。
返回值：

TRUE表示边框动态地去匹配文本尺寸。
FALSE表示边框没有动态地去匹配文本尺寸。

BOOL **GetAdaptPicture (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于图象窗口。修改图象以匹配图象窗口的尺寸。
返回值：

TRUE表示图象被修改以匹配窗口尺寸。
FALSE表示图象没有被修改以匹配窗口尺寸。

BOOL **GetAdaptSize (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于图象窗口。
返回值：

TRUE表示修改窗口尺寸以便与图象尺寸相匹配。
FALSE表示没有修改窗口尺寸以便匹配图象尺寸。

BOOL **GetAverage (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于棒图对象
返回值：

TRUE表示将求最后15个值的平均值。
FALSE表示将不会求最后15个值的平均值。

long int **GetBoxType (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

输入/输出对象(输入域、输出域、输入/输出域)的域类型

BOOL **GetCaption (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图象窗口带有标题。
FALSE表示图象窗口不带有标题。

BOOL **GetCloseButton (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图象窗口可以被关闭。
FALSE表示图象窗口不能被关闭。

BOOL **GetColorChangeType (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示当界限达到时整个棒图中均要进行颜色的改变。
FALSE表示当界限达到时在个别棒图段中要进行颜色的改变。

BOOL **GetCursorControl (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示打开输入/输出域里的光标控制(在退出当前域之后某个光标以栏目移动次序跳转到下域)。
FALSE表示关闭输入/输出域里的光标控制。

BOOL **GetCursorMode (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图象的光标模式是光标顺序。
FALSE表示图象的光标模式是栏目次序排列。

BOOL **GetEditAtOnce (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示将输入/输出域的立即输入属性设置为“是”(在访问输入/输出域时可用Tab键进行立即输入，而不用执行任何进一步的动作)。
FALSE表示将输入/输出域的立即输入属性设置为“否”。

BOOL **GetExtendedOperation (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示将滑块的扩充操作属性设置为“是”(通过单击当前滑块设置外部的区域将滑块设置给某个新的适当最小/最大限位值)
FALSE表示将滑块的扩充操作属性设置为“否”。

long int **GetHotkey (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

按钮的热键或组合键。

BOOL **GetHysteresis (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象的显示滞后发生。
FALSE表示棒图对象的显示不用滞后发生。

double **GetHysteresisRange (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图对象显示的滞后(以显示值的百分比体现的滞后)

char* **GetLanguageSwitch(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示文本列表对象是否用一种或多种语言配置。

TRUE　多种语言文本列表
FALSE　单语言文本列表

char* **GetLastChange (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图象上次改变的日期。

double **GetMax (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

带有棒图和滑块对象的完整值视图的确定值

BOOL **GetMaximizeButton (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图象或应用软件窗口可以最大化。
FALSE表示图象或应用软件窗口不能最大化。

double **GetMin (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图和滑块对象的最小值视图的绝对值

BOOL **GetMoveable (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图象或应用软件窗口可以移动。
FALSE表示图象或应用软件窗口不能移动。

long int **GetOffsetLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图象距图象窗口的左边缘窗口的水平距离

long int **GetOffsetTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图象距图象窗口的顶部边缘窗口的垂直距离

BOOL **GetOnTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图象或应用软件窗口总是位于前台。
FALSE表示图象或应用软件窗口不是一直位于前台。

BOOL **GetOperation (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示对象可被控制。
FALSE表示对象不能被控制。

注意:
如果函数的调用与整个图象相关，则必须令参数lpszObjectName=空。

BOOL **GetOperationMessage (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于输入/输出域、复选框、选项按钮和滑块对象
返回值：

TRUE表示当操作成功的实现时输出一条信息。
FALSE表示当操作成功的实现时没有信息输出。

BOOL **GetOperationReport (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于除了应用软件窗口、图象窗口和OLE控制以外的所有对象
返回值：

TRUE表示报告操作员活动的原因。
FALSE表示不报告操作员活动的原因。

注意:
如果函数的调用与整个图象相关，则必须令参数lpszObjectName=空。

long int **GetPasswordLevel (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于除了应用软件窗口、图象窗口和OLE控制以外的所有对象
返回值：

控制对象的授权等级

注意:
如果函数的调用与整个图象相关，则必须令参数lpszObjectName=空。

char* **GetPictureName (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于图象对象
返回值：

图象对象里所包含的图象名的指针

double **GetProcess (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
适用于滑块、棒图、复选框和选项组对象
返回值：

对于滑块和棒图：
将被显示的为过程值所预置的值。

对于复选框和选项组：
所选框

BOOL **GetScrollBars(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示图象窗口带有滚动条。
FALSE表示图象窗口不带有滚动条。

char* **GetServerName(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

显示对象名(OLE控制和OLE对象)，对象在其名下注册在视窗里。

BOOL **GetSizeable (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示应用软件或图象窗口的尺寸可被改变。
FALSE表示应用软件或图象窗口的尺寸不能改变。

long int **GetSmallChange (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

通过鼠标器单击动作移动滑块的步距数目

BOOL **GetTrend (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示棒图对象中显示趋势。
FALSE表示棒图对象中将不显示趋势。

long int **GetUpdateCycle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

整个图象的更新周期的编号值

BOOL **GetVisible (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示显示对象。
FALSE表示不显示对象。

注意:
如果函数的调用与整个图象相关，则必须令参数lpszObjectName=空。

BOOL **GetWindowBorder (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示应用软件或图象窗口显示时带有边框。
FALSE表示应用软件或图象窗口显示时不带有边框。

double **GetZeroPointValue (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图显示里的零点的绝对值

long int **GetZoom (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图象窗口的缩放因子

4.4.13 图形获取OLE控制函数

long int **GetPosition(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

作为某个编号值的OCX滑块的滑块位置

long int **GetRangeMax(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

作为某个编号值的OCX滑块的最大设置范围。

long int **GetRangeMin(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

作为某个编号值的OCX滑块的最小设置范围。

4.4.14 图形获取图象函数

char* **GetPictureDeactivated(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

“已释放”状态的图象名称
位图文件(*.bmp，*.dib)以及图元文件(*.emf，*.wmf)已链接。

char* **GetPictureDown(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

“打开/已按下”状态的图象。
位图文件(*.bmp，*.dib)以及图元文件(*.emf，*.wmf)已链接

char* **GetPictureUp(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

“关闭/没有按下”状态的图象名称
位图文件(*.bmp，*.dib)以及图元文件(*.emf，*.wmf)已链接。

BOOL **GetPicDeactReferenced(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示参考“已释放”状态的位图文件。
FALSE表示将位图文件的内容复制给图形(对于“已释放”状态)。

long int **GetPicDeactTransparent(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

“已释放”状态的透明颜色

注意:
此函数只适用于位图图形。

BOOL **GetPicDeactUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示使用“已释放”状态的透明颜色。
FALSE表示没有使用“已释放”状态的透明颜色。

BOOL **GetPicDownReferenced(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示参考“打开/已按下”状态的位图文件。
FALSE表示将位图文件复制给图形(对于状态“打开/已按下”)

long int **GetPicDownTransparent(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

“打开/已按下”状态的透明颜色

注意:
此函数只适用于位图图形。

BOOL **GetPicDownUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示使用了“打开/已按下”状态的透明颜色。
FALSE表示没有使用“打开/已按下”状态的透明颜色。

BOOL **GetPicReferenced(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示参考位图文件。
FALSE表示将位图文件的内容复制给图形对象。

long int **GetPicTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

图形对象的背景图象的透明颜色

注意:
此函数只适用于位图图形。

BOOL **GetPicUpReferenced(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示参考“关闭/未按下”状态的位图文件。
FALSE表示将位图文件的内容复制给图形(对于“关闭/未按下”状态)。

long int **GetPicUpTransparent(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

“关闭/未按下”状态的透明颜色

注意:
此函数只适用于位图图形。

BOOL **GetPicUpUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示使用了“关闭/未按下”状态的透明颜色。
FALSE表示没有使用“关闭/未按下”状态的透明颜色。

BOOL **GetPicUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示使用了图形对象的背景图象的透明颜色。
FALSE表示没有使用图形对象的背景图象的透明颜色。

4.4.15 图形获取属性函数

BOOL **GetPropBOOL(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName**)**
返回值：

类型BOOL 的属性值。

参数：

lpszPropertyName　　=属性的OLE自动化名称

char* **GetPropChar(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName**)**
返回值：

类型字符型的属性值上的指针。

参数：

lpszPropertyName　　=属性的OLE自动化名称

double **GetPropDouble(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName**)**
返回值：

类型double 的属性值。

参数：

lpszPropertyName　　=属性的OLE自动化名称

long int **GetPropWord(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName**)**
返回值：

类型long int 的属性值。

参数：

lpszPropertyName　　=属性的OLE自动化名称

4.4.16 图形获取状态函数

long int **GetBasePicTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

原始图象的透明颜色

注意:
此函数只适用于位图图形。

BOOL **GetBasePicUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示使用了原始图象的透明颜色。
FALSE表示没有使用原始图象的透明颜色。

char* **GetBasePicture(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

状态显示的原始图象名称

BOOL **GetBasePicReferenced(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示参考位图文件。
FALSE表示将位图文件的内容复制给状态显示。

BOOL **GetFlashFlashPicture(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示闪烁图象动态地活动。
FALSE表示闪烁图象静态地活动。

char* **GetFlashPicture(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

闪烁图象名称(图形的文件名称)

long int **GetFlashPicTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

闪烁图象的透明颜色

注意:
此函数只适用于位图图形。

BOOL **GetFlashPicUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示使用了闪烁图象的透明颜色。
FALSE表示没有使用闪烁图象的透明颜色。

BOOL **GetFlashPicReferenced(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

TRUE表示参考位图文件。
FALSE表示将位图文件的内容复制给状态显示。

long int **GetFlashRateFlashPic (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

闪烁图象的闪烁频率的编号值

0： 无闪烁频率
1： 闪烁频率慢(大约0.5赫兹)
2： 闪烁频率中等(大约2赫兹)
3： 闪烁频率快(大约8赫兹)

注意:
因为闪烁是一种软件过程，故其频率取决于系统和硬件(对象数目、处理器、存储器、更新时间等)。

long int **GetIndex(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

多边形或折线的当前下标。

4.4.17 图形获取样式函数

long int **GetBackBorderWidth(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

三维边框和滑块对象的边框权重的编号值

long int **GetBorderEndStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

线尾样式的类型的编号值

long int **GetBorderStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

线条样式或边框样式的编号值

long int **GetBorderWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

作为编号值的线条权重或边框线条的权重

long int **GetBoxAlignment (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

在复选框或选项按钮，左对齐或右对齐中分配控制单元的编号值

long int **GetFillStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

填充模式的类型的编号值

注意:
如果函数的调用与整个图象相关，则必须令参数lpszObjectName=空。

long int **GetFillStyle2 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

棒图显示里的棒图填充模式的编号值

long int **GetItemBorderStyle(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

用于“文本列表”对象的分隔行的样式

long int **GetItemBorderWidth(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
返回值：

用于“文本列表”对象的分隔行的权重

BOOL **GetPressed(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于循环按钮
返回值：

TRUE表示打开/已按下的开关设置。
FALSE表示关闭/未按下的开关设置。

BOOL **GetToggle(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只用于循环按钮
返回值：

TRUE表示开关设置没有锁存。
FALSE表示开关设置已锁存。

BOOL **GetWindowsStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**
只能使用位于按钮上的此函数。
返回值：

TRUE与“视窗样式”相对应，表示按钮按照视窗标准显示。
FALSE与“非视窗样式”相对应，表示可自己确定按钮的显示外貌。

4.4.18 图形设置轴线函数

BOOL **SetAlignment (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAlignment**);**
只用于棒图对象
功能:

设置文本相对于棒图的位置(右/左)。

参数：

bAlignment　　=文本位于棒图的右边或左边。

TRUE：文本位于棒图的右边。
FALSE：文本位于棒图的左边。

BOOL **SetAxisSection (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dAxisSection**);**
只用于棒图对象
功能:

设置坐标轴上的量度单位(在两个邻近轴线标志之间的值的差)。

参数：

dAxisSection　　=轴线区段

BOOL **SetExponent (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bExponent**);**
只用于棒图对象
功能:

确定显示数字时是否应带有指数或小数。

参数：

bExponent　　=用指数/小数编号的坐标轴

TRUE表示坐标轴上显示的数字带有指数。
FALSE表示坐标轴上显示的数字带有小数。

BOOL **SetLeftComma (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lLeftComma**);**
只用于棒图对象
功能:

设置小数点左边数字的位数。

参数：

lLeftComma　　=小数点左边数字的位数

BOOL **SetLongStrokesBold (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bLongStrokesBold**);**
只用于棒图对象
功能:

设置标尺表里的长轴部分是显示为粗体字还是标准字。

参数：

bLongStrokesBold　　=长轴部分为粗体字/标准字

TRUE表示棒图的标尺表中的长轴部分是粗体的。
FALSE表示棒图的标尺表中的长轴部分是标准的。

BOOL **SetLongStrokesOnly (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bLongStrokesOnly**);**
只用于棒图对象
功能:

确定棒图标尺上是只使用长轴部分或是也使用子部分。

参数：

bLongStrokesOnly =专门使用长轴部分是/否。

TRUE表示棒图标尺上只使用长轴部分。
FALSE表示棒图标尺上使用长轴部分和子部分。

BOOL **SetLongStrokesSize(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lLongStrokesSize**);**
只用于棒图对象
功能:

设置棒图标尺上的长轴部分的长度。

参数：

lLongStrokesSize =长轴部分的长度

BOOL **SetRightComma (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRightComma**);**
只用于棒图对象
功能:

设置坐标轴刻度上的小数点右边数字的位数(0至20)。

参数：

lRightComma =小数点右边数字的位数

BOOL **SetScaleTicks (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lScaleTicks**);**
只用于棒图对象
功能:

将棒图内的标尺记号设置为整个棒图高度的百分比形式。

参数：

lScaleTicks = 整个棒图高度的百分比形式的标尺记号

BOOL **SetScaling (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bScaling**);**
只用于棒图对象
功能:

打开和关闭标尺。

参数：

bScaling =标尺打开/关闭

TRUE表示“带有标尺”
FALSE表示“不带有标尺”

BOOL **SetScalingType(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lScalingType**);**
只用于棒图对象
功能:

设置棒图标尺的类型。

参数：

lScalingType　　=棒图标尺的类型

0： 线性的(同一权重)
1： 对数的(侧重于低限)
2： 负对数的(侧重于高限)
3： 自动的(线性的)
4： 正切的(侧重于高限和低限)
5： 平方的(侧重于高限)
6： 立方的(侧重于高限)

4.4.19 图形设置颜色函数

BOOL **SetBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColor**);**
功能:

设置对象的背景颜色。

参数：

lBackColor　　=对象背景颜色的编号值

注意:
如果函数的调用与整个图象相关，则必须令参数lpszObjectName=空。

BOOL **SetBackColor2 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColor2**);**
功能:

设置棒图对象的棒图颜色。

参数：

lBackColor2　　=棒图对象的棒图颜色的编号值

BOOL **SetBackColor3 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColor3**);**
功能:

设置棒图对象的棒图背景的颜色。

参数：

lBackColor3　　= 棒图对象的棒图背景颜色的编号值

BOOL **SetBackColorBottom(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColorBottom**);**
功能:

设置滑块对象的底部/右边的背景颜色。

参数：

lBackColorBottom　　=棒图背景颜色的编号值

BOOL **SetBackColorTop(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackColorTop**);**
功能:

设置滑块对象顶部/左边的背景颜色。

参数：

lBackColorTop =棒图背景颜色的编号值

BOOL **SetBorderBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderBackColor**);**
功能:

设置线条或边框的背景颜色。

参数：

lBorderBackColor = 线条或边框背景颜色的编号值

BOOL **SetBorderColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderColor**);**
功能:

设置线条或边框颜色。

参数：

lBorderColor =线条或边框颜色的编号值

BOOL **SetBorderColorBottom (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderColorBottom**);**
功能:

设置三维边框的右边和底部部分(阴影)的颜色。

参数：

lBorderColorBottom =三维阴影颜色的编号值

BOOL **SetBorderColorTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderColorTop**);**
功能:

设置三维边框左边和顶部部分的颜色。

参数：

lBorderColorTop =三维边框左边和顶部颜色的编号值

BOOL **SetButtonColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lButtonColor**);**
功能:

设置滑块对象的按钮颜色。

参数：

lButtonColor =滑块对象中的按钮颜色的编号值

BOOL **SetColorBottom (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorBottom**);**
功能:

设置滑块底部/右部的颜色。

参数：

lColorBottom = 滑块底部/右部颜色的编号值

BOOL **SetColorTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorTop**);**
功能:

设置顶部/左部滑块位置的颜色。

参数：

lColorTop = 顶部/左部滑块位置颜色的编号值

BOOL **SetFillColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillColor**);**
功能:

设置填充模式颜色。

参数：

lFillColor　　=填充模式颜色的编号值

注意:
如果函数的调用与整个图象相关，则必须令参数lpszObjectName=空。

BOOL **SetForeColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lForeColor**);**
功能:

设置对象里的文本的颜色。

参数：

lForeColor　　=文本颜色的编号值

BOOL **SetItemBorderBackColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lItemBorderBackColor**);**
功能:

设置“文本列表”对象的分隔行的背景颜色。

参数：

lItemBorderBackColor　=背景颜色的编号值

BOOL **SetItemBorderColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lItemBorderColor**);**
功能:

设置“文本列表”对象的分隔行颜色。

参数：

lItemBorderColor =分隔行颜色的编号值

BOOL **SetScaleColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lScaleColor**);**
功能:

设置棒图对象的标尺颜色。

参数：

lScaleColor　　=棒图对象中的标尺颜色的编号值

BOOL **SetSelBGColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lSelBGColor**);**
功能:

设置“文本列表”对象里的浏览器列表中的所选条目的背景颜色。

参数：

lSelBGColor　　= 所选条目的背景颜色

BOOL **SetSelTextColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lSelTextColor**);**
功能:

设置“文本列表”对象里的浏览器列表中的所选条目的文本颜色。

参数：

lSelTextColor　　= 所选条目的文本颜色

BOOL **SetTrendColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lTrendColor**);**
功能:

设置棒图对象的趋势颜色。

参数：

lTrendColor　　=棒图对象里的趋势颜色的编号值

BOOL **SetUnselBGColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lUnselBGColor**);**
功能:

设置“文本列表”对象里的浏览器列表中的未选条目的背景颜色。

参数：

lUnselBGColor = 浏览器列表中的未选条目的背景颜色

BOOL **SetUnselTextColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lUnselTextColor**);**
功能:

设置“文本列表”对象里的浏览器列表中的未选条目的文本颜色。

参数：

lUnselTextColor = 文本颜色的编号值

4.4.20 图形设置填充函数

BOOL **SetFilling (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFilling**);**
功能:

激活/释放带有背景颜色的动态填充。

参数：

bFilling = 激活的/未激活的带有背景颜色的动态填充

TRUE表示带有背景颜色的动态填充是激活的。
FALSE表示带有背景颜色的动态填充未激活。

BOOL **SetFillingIndex (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillingIndex**);**
功能:

设置填充索引。

参数：

lFillingIndex =填充索引的编号值

4.4.21 图形设置闪烁函数

BOOL **SetBackFlashColorOff (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackFlashColorOff**);**
功能:

当闪烁属性关闭时设置背景颜色。

参数：

lBackFlashColorOff = 背景颜色的编号值，当闪烁属性关闭时

BOOL **SetBackFlashColorOn (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackFlashColorOn**);**
功能:

当闪烁属性打开时设置背景颜色。

参数：

lBackFlashColorOn = 背景颜色的编号值，当闪烁属性打开时

BOOL **SetBorderFlashColorOff (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderFlashColorOff**);**
功能:

当闪烁属性关闭时设置边框或线条颜色。

参数：

lBorderFlashColorOff =边框或线条颜色的编号值，当闪烁属性关闭时

BOOL **SetBorderFlashColorOn (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderFlashColorOn**);**
功能:

当闪烁属性打开时设置边框或线条颜色。

参数：

lBorderFlashColorOn=边框或线条颜色的编号值，当闪烁属性打开时

BOOL **SetFlashBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashBackColor**);**
功能:

激活或释放背景闪烁。

参数：

bFlashBackColor　=背景闪烁为激活的/未激活的

TRUE表示背景闪烁是激活的。
FALSE表示背景闪烁是未激活的。

BOOL **SetFlashBorderColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashBorderColor**);**
功能:

激活闪烁边框或线条。

参数：

bFlashBorderColor　=闪烁边框或线条为激活的/未激活的

TRUE表示边框或线条闪烁是激活的。
FALSE表示边框或线条闪烁是未激活的。

BOOL **SetFlashForeColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashForeColor**);**
功能:

激活文本闪烁。

参数：

bFlashForeColor　=文本闪烁为激活的/未激活的

TRUE表示文本闪烁是激活的。
FALSE表示文本闪烁是未激活的。

BOOL **SetFlashRateBackColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateBackColor**);**
功能:

设置背景闪烁频率。
参数：

lFlashRateBackColor　=背景闪烁频率

BOOL **SetFlashRateBorderColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateBorderColor**);**
功能:

设置线条或边框闪烁频率。

参数：

lFlashRateBorderColor =线条或边框闪烁频率

BOOL **SetFlashRateForeColor (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateForeColor**);**
功能:

设置文本闪烁频率。

参数：

lFlashRateForeColor =文本闪烁频率

BOOL **SetForeFlashColorOff (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lForeFlashColorOff**);**
功能:

当闪烁属性关闭时设置文本颜色。

参数：

lForeFlashColorOff = 文本颜色的编号值，当闪烁属性关闭时

BOOL **SetForeFlashColorOn (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lForeFlashColorOn**);**
功能:

当闪烁属性打开时设置文本颜色。

参数：

lForeFlashColorOn = 文本颜色的编号值，当闪烁属性打开时

4.4.22 图形设置焦点函数

BOOL **Set_Focus(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName**);**

功能:

将中心放置在指定的对象上

4.4.23 图形设置字体函数

BOOL **SetAlignmentLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lAlignmentLeft**);**
功能:

设置文本的水平对齐 (左对齐、居中、右对齐)。

参数：

lAlignmentLeft = 文本的水平对齐的编号值(左对齐、居中、右对齐)

BOOL **SetAlignmentTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lAlignmentTop**);**
功能:

设置文本的垂直对齐 (顶对齐、居中、底对齐)。

参数：

lAlignmentTop = 文本垂直对齐的编号值(顶对齐、居中、底对齐)

BOOL **SetFontBold (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontBold**);**
功能:

关闭和打开“粗体字”文本属性。

参数：

bFontBold =“粗体字”打开/关闭
TRUE表示“粗体字”打开。
FALSE表示“粗体字”关闭。

BOOL **SetFontItalic (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontItalic**);**
功能:

打开和关闭“斜体字”文本属性。

参数：

bFontItalic　　=“斜体字”打开/关闭

TRUE表示“斜体字”打开。
FALSE表示“斜体字”关闭。

BOOL **SetFontName (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szFontName**);**
功能:

设置字体。

参数：

szFontName　　=字体名称的指针

BOOL **SetFontSize (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFontSize**);**
功能:

设置字体尺寸。

参数：

lFontSize　　=字体尺寸

BOOL **SetFontUnderline (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFontUnderline**);**
功能:

打开和关闭“下划线”文本属性。

参数：

bFontUnderline　=文本样式“下划线”打开/关闭

TRUE表示“下划线”打开。
FALSE表示“下划线”关闭。

BOOL **SetOrientation (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOrientation**);**
功能:

设置文本方向(水平/垂直)。

参数：

bOrientation　　=水平/垂直文本方向

TRUE表示水平文本方向。
FALSE表示垂直文本方向。

BOOL **SetText (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szText**);**
功能:

设置静态文本的指针。对于单选和复选框、以及多边形和折线，单元或指针的设置必须在调用**SetText**函数之前用**SetIndex**函数进行。

参数：

szText　　=静态文本的指针

4.4.24 图形设置几何函数

BOOL **SetActualPointLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lActualPointLeft**);**
功能:

设置与图象原点相关的多边形或折线的角点的当前水平位置。设置多边形的当前点是用**SetIndex**函数。

参数：

lActualPointLeft = 多边形或折线的角点的X值 (水平位置)

BOOL **SetActualPointTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lActualPointTop**);**
功能:

设置与图象原点相关的多边形或折线的角点的当前垂直位置。设置多边形的当前点是用**SetIndex**函数。

参数：

lActualPointTop = 多边形或折线的角点的Y值(垂直位置)

BOOL **SetBoxCount (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBoxCount**);**
功能:

设置复选框区里的复选框或选项组里的选项按钮的编号。

参数：

lBoxCount = 位于复选框区里的复选框的编号或选项组里的选项按钮的编号。

BOOL **SetDirection (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lDirection**);**
功能:

设置棒图对象里的的棒图方向(向上、向下、向左、或向右)。

参数：

lDirection = 棒图对象里的棒图方向的编号值(向上、向下、向左、或向右)

BOOL **SetEndAngle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lEndAngle**);**
功能:

设置饼图和扇形以及圆弧和椭圆弧的终止角。

参数：

lEndAngle = 饼图和扇形以及圆弧和椭圆弧的终止角

BOOL **SetHeight (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lHeight**);**
功能:

设置包围某个对象的长方形的高度。

参数：

lHeight =包围某个对象的长方形的高度

注意:
如果函数的调用与整个图象相关，则必须令lpszObjectName参数=空。

BOOL **SetLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lLeft**);**
功能:

设置包围某个对象的长方形的左上角在X轴上的位置。

参数：

lLeft = (X轴上的位置)包围某个对象的长方形的左上角的X值

BOOL **SetPointCount (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPointCount**);**
功能：

设置多边形或折线的角的编号。

参数：

lPointCount =角的编号

BOOL **SetRadius (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRadius**);**
功能:

设置圆形、饼图形、或圆弧的半径。

参数：

lRadius =半径

BOOL **SetRadiusHeight (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRadiusHeight**);**
功能:

设置椭圆形、扇形、或椭圆弧的垂直半径。

参数：

lRadiusHeight =垂直半径

BOOL **SetRadiusWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRadiusWidth**);**
功能：

设置椭圆形、扇形、或椭圆弧的水平半径。

参数：

lRadiusWidth =水平半径

BOOL **SetReferenceRotationLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lReferenceRotationLeft**);**
适用于线条、多边形和折线。
功能:

设置旋转的参考X值(对象旋转所围绕的参考点)。

参数：

lReferenceRotationLeft =旋转的参考X值

BOOL **SetReferenceRotationTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lReferenceRotationTop**);**
适用于线条、多边形和折线
功能:

设置旋转的参考Y值(对象旋转所围绕的参考点)。

参数：

lReferenceRotationTop =旋转的参考Y值

BOOL **SetRotationAngle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRotationAngle**);**
适用于线条、多边形和折线
功能:

设置对象绕旋转点的顺时针旋转角(度数)。

参数：

lRotationAngle =旋转角

BOOL **SetRoundCornerHeight (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRoundCornerHeight**);**
功能:

设置圆角长方形的角的垂直半径。

参数：

lRoundCornerHeight =垂直半径

BOOL **SetRoundCornerWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRoundCornerWidth**);**
功能:

设置圆角长方形的角的水平半径。

参数：

lRoundCornerWidth =水平半径

BOOL **SetStartAngle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lStartAngle**);**
功能:

设置饼图和扇形以及圆弧和椭圆弧的起始角。

参数：

lStartAngle　　=起始角

BOOL **SetTop (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lTop**);**
功能:

设置包围某个对象的长方形的左上角的Y轴值。

参数：

lTop　　=　包围某个对象的长方形的左上角的Y轴值

BOOL **SetWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lWidth**);**
功能:

设置包围某个对象的长方形的宽度。

参数：

lWidth　　=包围某个对象的长方形的宽度

注意:
如果函数的调用与整个图象相关，则必须令lpszObjectName参数=空。

BOOL **SetZeroPoint (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lZeroPoint**);**
功能:

设置棒图对象里的零点。

参数：

lZeroPoint　　=棒图对象里的零点

4.4.25 图形设置I/O函数

BOOL **SetAssumeOnExit (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAssumeOnExit**);**
只用于输入/输出域
功能：

将系统设置为一退出域就接受输入(使用Tab键或回车键)。

参数：

bAssumeOnExit = 一旦从域中退出，就马上接受输入：是/否

TRUE表示系统一旦从域中退出就马上接受输入。
FALSE表示系统一旦从域中退出，也不会接受输入。

BOOL **SetAssumeOnFull (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAssumeOnFull**);**
只用于输入/输出域
功能：

将系统设置为一旦完成输入(字符的指定编号被输入)就自动地从输入域中退出(不使用Tab键或回车键)并且马上接受输入。

参数：

bAssumeOnFull = 一旦输入完成就接受数值：是/否

TRUE表示一旦完成输入，系统将自动地接受输入。
FALSE表示一旦完成输入，系统不会自动地接受输入。

BOOL **SetBitNumber(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBitNumber**);**
功能：

设置“位”列表类型里的输出值的相应位。

参数：

lBitNumber =输出值的相应位

BOOL **SetClearOnError (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bClearOnError**);**
只用于输入/输出域
功能:

将输入域里的域输入条目设置为当有非法输入时能自动地被删除。

参数：

bClearOnError = 当有非法输入时，自动地删除输入域里的域输入条目：是/否

TRUE表示当有非法输入时能自动地删除域输入条目。
FALSE表示当有非法输入时不能自动地删除域输入条目。

BOOL **SetClearOnNew (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bClearOnNew**);**
只用于输入/输出域
功能:

将输入域里的域输入条目设置为当有新建条目时能将其删除。

参数：

bClearOnNew = 当有新建的条目时，删除输入域里的域输入条目：是/否

TRUE表示当有新建的条目时系统删除域输入条目。
FALSE表示当有新建的条目时系统不删除域输入条目。

BOOL **SetHiddenInput (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bHiddenInput**);**
只用于输入/输出域
功能:

在输入期间隐含输入值。所显示的星号(*)用于每个字符。

参数：

bHiddenInput =隐含输入：是/否

TRUE表示输入被隐含。
FALSE表示输入没有被隐含。

BOOL **SetNumberLines(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lNumberLines**);**
功能:

设置“文本列表”对象里的浏览器列表中的可见行的编号。

注意:
如果所配置文本的编号大于可见行的编号，则浏览器列表将获得垂直滚动条。

参数：

lNumberLines　=可见行的编号

BOOL **SetOutputValueChar (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szOutputValueChar**);**
只用于输入/输出域
功能:

设置输出值的指针。

参数：

szOutputValueChar　=输出值的指针

BOOL **SetOutputValueDouble (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dOutputValueDouble**);**
只用于输入/输出域
功能:

设置输出值。

参数：

dOutputValueDouble　=输出值

4.4.26 图形设置界限函数

BOOL **SetAlarmHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dAlarmHigh**);**
功能：

设置棒图对象里的报警高限

参数：

dAlarmHigh　　=报警高限

BOOL **SetAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dAlarmLow**);**
功能:

设置棒图对象里的报警低限。

参数：

dAlarmLow　　=报警低限

BOOL **SetCheckAlarmHigh (**LPCTSTRlpszPictureName,LPCTSTRlpszObjectName,BOOLbCheckAlarmHigh**);**
功能:

设置“报警高限”的限位值的监控函数。

参数：

bCheckAlarmHigh =监控：是/否

TRUE表示“报警高限”的限位值被监控。
FALSE表示“报警高限”的限位值没有被监控。

BOOL **SetCheckAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckAlarmLow**);**
功能:

设置“报警低限”的限位值的监控函数。

参数：

bCheckAlarmLow =监控：是/否

TRUE表示“报警低限”的限位值被监控。
FALSE表示“报警低限”的限位值没有被监控。

BOOL **SetCheckLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitHigh4**);**
功能:

设置棒图对象里的“保留4”的高限值的监控函数。

参数：

bCheckLimitHigh4 =监控：是/否

TRUE表示“保留4”的高限值被监控。
FALSE表示“保留4”的高限值没有被监控。

BOOL **SetCheckLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitHigh5**);**
功能:

设置棒图对象里的“保留5”的高限值的监控函数。

参数：

bCheckLimitHigh5 =监控：是/否

TRUE表示“保留5”的高限值被监控。
FALSE表示“保留5”的高限值没有被监控。

BOOL **SetCheckLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitLow4**);**
功能:

设置棒图对象里的“保留4”的低限值的监控函数。

参数：

bCheckLimitLow4 =监控：是/否

TRUE表示“保留4”的低限值被监控。
FALSE表示“保留4”的低限值没有被监控。

BOOL **SetCheckLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckLimitLow5**);**
功能:

设置棒图对象的“保留5”的低限值的监控函数。

参数：

bCheckLimitLow5 =监控：是/否

TRUE表示“保留5”的低限值被监控。
FALSE表示“保留5”的低限值没有被监控。

BOOL **SetCheckToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckToleranceHigh**);**
功能:

设置棒图对象的“公差高限”的限位值的监控函数。

参数：

bCheckToleranceHigh =监控：是/否

TRUE表示“公差高限”的限位值被监控。
FALSE表示“公差高限”的限位值没有被监控。

BOOL **SetCheckToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckToleranceLow**);**
功能:

设置棒图对象的“公差低限”的限位值的监控函数。

参数：

bCheckToleranceLow =监控：是/否

TRUE表示“公差低限”的限位值被监控。
FALSE表示“公差低限”的限位值没有被监控。

BOOL **SetCheckWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckWarningHigh**);**
功能：

设置棒图对象的“报警高限”的限位值的监控函数。
参数：

bCheckWarningHigh =监控：是/否

TRUE表示“报警高限”的限位值被监控。
FALSE表示“报警高限”的限位值没有被监控。

BOOL **SetCheckWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCheckWarningLow**);**
功能:

设置棒图对象的“报警低限”的限位值的监控函数。

参数：

bCheckWarningLow =监控：是/否

TRUE表示“报警低限”的限位值被监控。
FALSE表示“报警低限”的限位值没有被监控。

BOOL **SetColorAlarmHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorAlarmHigh**);**
功能:

设置“报警高限”的限位值达到时的棒图颜色。

参数：

lColorAlarmHigh =棒图颜色的编号值

BOOL **SetColorAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorAlarmLow**);**
功能:

设置“报警低限”的限位值达到时的棒图颜色。

参数：

lColorAlarmLow =棒图颜色的编号值

BOOL **SetColorLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitHigh4**);**
功能:

设置棒图对象里的“保留4”的高限值达到时的棒图颜色。

参数：

lColorLimitHigh4 =棒图颜色的编号值

BOOL **SetColorLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitHigh5**);**
功能:

设置棒图对象里的“保留5”的高限值达到时的棒图颜色。

参数：

lColorLimitHigh5 =棒图颜色的编号值

BOOL **SetColorLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitLow4**);**
功能:

设置棒图对象里的“保留4”的低限值达到时的棒图颜色。

参数：

lColorLimitLow4 =棒图颜色的编号值

BOOL **SetColorLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorLimitLow5**);**
功能:

设置棒图对象里的“保留5”的低限值达到时的棒图颜色。

参数：

lColorLimitLow5 =棒图颜色的编号值

BOOL **SetColorToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorToleranceHigh**);**
功能：

设置棒图对象里的“公差高限”的限位值达到时的棒图颜色。

参数：

lColorToleranceHigh　　=棒图颜色的编号值

BOOL **SetColorToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorToleranceLow**);**
功能：

设置棒图对象里的“公差低限”的限位值达到时的棒图颜色。

参数：

lColorToleranceLow　=棒图颜色的编号值

BOOL **SetColorWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorWarningHigh**);**
功能：

设置棒图对象里的“报警高限”的限位值达到时的棒图颜色。

参数：

lColorWarningHigh　=棒图颜色的编号值

BOOL **SetColorWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lColorWarningLow**);**
功能：

设置棒图对象里的“报警低限”的限位值达到时的棒图颜色。

参数：

lColorWarningLow =棒图颜色的编号值

BOOL **SetLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitHigh4**);**
功能：

设置棒图对象里的“保留4”的高限值。

参数：

dLimitHigh4　　=“保留4”的高限值

BOOL **SetLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitHigh5**);**
功能:

设置棒图对象里的“保留5”的高限值。

参数：

dLimitHigh5　　=“保留5”的高限值

BOOL **SetLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitLow4**);**
功能:

设置棒图对象里的“保留4”的低限值。

参数：

dLimitLow4　　=“保留4”的低限值

BOOL **SetLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitLow5**);**
功能：

设置棒图对象里的“保留5”的低限值。

参数：

dLimitLow5　　=“保留5”的低限值

BOOL **SetLimitMax (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitMax**);**
功能:

设置输入/输出域的高限值。

参数：

dLimitMax　　=高限值

BOOL **SetLimitMin (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dLimitMin**);**
功能:

设置输入/输出域的低限值。

参数：

dLimitMin　　=低限值

BOOL **SetMarker (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bMarker**);**
功能：

设置棒图对象里的限位值的显示。

参数：

bMarker　　=限位值显示：打开/关闭

TRUE表示限位值显示为棒图对象里的标尺值。
FALSE表示棒图对象里没有显示限位值。

BOOL **SetToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dToleranceHigh**);**
功能：

设置棒图对象里的“公差高限”的限位值。

参数：

dToleranceHigh　=“公差高限”的限位值

BOOL **SetToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dToleranceLow**);**
功能：

设置棒图对象里的“公差低限”的限位值。

参数：

dToleranceLow　=“公差低限”的限位值

BOOL **SetTypeAlarmHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeAlarmHigh**);**
功能：

设置在棒图对象里提供“报警高限”的限位值的方式。

参数：

bTypeAlarmHigh= 百分比形式或绝对值形式的“报警高限”的限位值

TRUE表示以百分比形式提供“报警高限”的限位值。

FALSE表示以绝对值形式提供“报警高限”的限位值。

BOOL **SetTypeAlarmLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeAlarmLow**);**
功能：

设置棒图对象里提供“报警低限”的限位值的方式。

参数：

bTypeAlarmLow = 百分比形式或绝对值形式的“报警低限”的限位值

TRUE表示以百分比形式提供的“报警低限”的限位值。
FALSE表示以绝对值形式提供的“报警低限”的限位值。

BOOL **SetTypeLimitHigh4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitHigh4**);**
功能：

设置棒图对象里提供“保留4”的高限值的方式。

参数：

bTypeLimitHigh4 = 百分比形式或绝对值形式的“保留4”的高限值

TRUE表示以百分比形式提供的“保留4”的高限值。
FALSE表示以绝对值形式提供的“保留4”的高限值。

BOOL **SetTypeLimitHigh5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitHigh5**);**
功能：

设置在棒图对象里提供“保留5”的高限值的方式。
参数：

bTypeLimitHigh5 = 百分比形式或绝对值形式的“保留5”的高限值

TRUE表示以百分比形式提供的“保留5”的高限值。
FALSE表示以绝对值形式提供的“保留5”的高限值。

BOOL **SetTypeLimitLow4 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitLow4**);**
功能：

设置在棒图对象里提供“保留4”的低限值的方式。

参数：

bTypeLimitLow4 =百分比形式或绝对值形式的“保留4”的低限值

TRUE表示以百分比形式提供的“保留4”的低限值。
FALSE表示以绝对值形式提供的“保留4”的低限值。

BOOL **SetTypeLimitLow5 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeLimitLow5**);**
功能：

设置在棒图对象里提供“保留5”的低限值的方式。

参数：

bTypeLimitLow5 = 百分比形式或绝对值形式的“保留5”的低限值

TRUE表示以百分比形式提供的“保留5”的低限值。
FALSE表示以绝对值形式提供的“保留5”的低限值。

BOOL **SetTypeToleranceHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeToleranceHigh**);**
功能：

设置在棒图对象里提供“公差高限”的限位值的方式。

参数：

bTypeToleranceHigh = 百分比形式或绝对值形式的“公差高限”的限位值

TRUE表示以百分比形式提供的“公差高限”的限位值。
FALSE表示以绝对值形式提供的“公差高限”的限位值。

BOOL **SetTypeToleranceLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeToleranceLow**);**
功能：

设置在棒图对象里提供“公差低限”的限位值的方式。

参数：

bTypeToleranceLow = 百分比形式或绝对值形式的“公差低限”的限位值

TRUE表示以百分比形式提供的“公差低限”的限位值。
FALSE表示以绝对值形式提供的“公差低限”的限位值。

BOOL **SetTypeWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeWarningHigh**);**
功能：

设置在棒图对象里提供“报警高限”的限位值的方式。

参数：

bTypeWarningHigh = 百分比形式或绝对值形式的“报警高限”的限位值

TRUE表示以百分比形式提供的“报警高限”的限位值。
FALSE表示以绝对值形式提供的“报警高限”的限位值。

BOOL **SetTypeWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTypeWarningLow**);**
功能：

设置在棒图对象里提供“报警低限”的限位值的方式。

参数：

bTypeWarningLow = 百分比形式或绝对值形式的“报警低限”的限位值

TRUE表示以百分比形式提供的“报警低限”的限位值。
FALSE表示以绝对值形式提供的“报警低限”的限位值。

BOOL **SetWarningHigh (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dWarningHigh**);**
功能：

设置棒图对象里的“报警高限”的限位值。

参数：

dWarningHigh =“报警高限”的限位值

BOOL **SetWarningLow (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dWarningLow**);**
功能:

设置棒图对象里的“报警低限”的限位值。

参数：

dWarningLow =“报警低限”的限位值

4.4.27 图形设置链接函数

BOOL **SetLink(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName,LPCTSTR lpszPropertyName, LPLINKINFO() pLink**);**

功能：

创建对象属性的标签链接。

参数：

lpszPropertyName =对象属性
pLink =标签链接结构的地址条目

结构：

标签结构由LinkTyp、dwCycle和szLinkName等参数组成。

LinkTyp：
1 直接标签链接
2 间接标签链接
dwCycle：
0 一旦改变
1 250ms
2 500ms
3 1s
4 2s
5 5s
6 10s
7 1min
8 5min
9 10min
10 1hr
11-15 用户自定义周期1至5

dwCycle：
标签名称

实例：

```
{       LINKINFO linkinfo;

        linkinfo.lLinkType=1;
        linkinfo.cwCycle=3;
        strcpy(linkinfo.szLinkName,"Tag1");

        SetLink(....,&linkinfo);
}
```

4.4.28 图形设置其它函数

BOOL **SetAverage (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bAverage**);**
功能：

确定棒图对象里的最后15个值是否需要求其平均值。

参数：

bAverage　　=求值的平均值：是/否

TRUE表示将计算最后15个值的平均值。
FALSE表示不计算最后15个值的平均值。

BOOL **SetBoxType (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBoxType**);**
功能：

设置输入/输出对象的域类型(输入域、输出域、输入/输出域)。

参数：

lBoxType　　= 输入/输出对象的域类型(输入域、输出域、输入/输出域)

BOOL **SetColorChangeType(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bColorChangeType**);**
功能：

设置棒图颜色变化的类型。

参数：

bColorChangeType　=颜色变化的类型：全部/部分

TRUE表示部分棒图的颜色变化。
FALSE表示整个棒图的颜色变化。

BOOL **SetCursorControl (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCursorControl**);**
功能：

设置输入/输出域的光标控制。
参数：

bCursorControl　=光标控制：打开/关闭

TRUE表示输入/输出域里的光标控制被打开(在退出当前域后光标以栏目顺序跳转到下域)。
FALSE表示输入/输出域里的光标控制被关闭。

BOOL **SetCursorMode (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bCursorMode**);**
功能：

设置图象的光标模式。

参数：

bCursorMode　= 光标模式：栏目次序排列/光标顺序

TRUE表示图象的光标模式为光标顺序。
FALSE表示图象的光标模式为栏目次序排列。

BOOL **SetEditAtOnce (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bEditAtOnce**);**
功能：

设置输入/输出域的输入。

参数：

bEditAtOnce　=立即输入：是/否

TRUE表示将立即输入属性设置为输入/输出域里的“是”(在用TAB键访问输入/输出域时可进行立即输入，而不用执行任何其它的进一步的动作)。
FALSE表示将立即输入属性设置为输入/输出域里的“否”。

BOOL **SetExtendedOperation (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bExtendedOperation**);**
功能：

设置滑块的扩充操作。

参数：

bExtendedOperation =扩充操作：是/否

TRUE表示将滑块的扩充操作属性设置为“是”(通过单击当前滑块设置的外部区域为滑块设置新的合适的最小/最大限位值)
FALSE表示将滑块的扩充操作属性设置为“否”。

BOOL **SetHysteresis (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bHysteresis**);**
功能：

设置棒图对象的显示(滞后或不滞后)。

参数：

bHysteresis =滞后或不滞后显示

TRUE表示产生滞后的显示。
FALSE表示产生不滞后的显示。

BOOL **SetHysteresisRange (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dHysteresisRange**);**
功能：

设置棒图对象显示的滞后 (滞后表示为显示值的某个百分比)。

参数：

dHysteresisRange =滞后

BOOL **SetMax (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dMax**);**
功能：

设置棒图和滑块对象的完整值视图的绝对值。

参数：

dMax =最大值

BOOL **SetMin (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dMin**);**
功能：

设置棒图和滑块对象的最小值视图的绝对值。

参数：

dMin =最小值

BOOL **SetOffsetLeft (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lOffsetLeft**);**
功能：

设置图象窗口里的图象距窗口左边缘的水平距离。

参数：

lOffsetLeft =图象偏移量

BOOL SetOffsetTop (LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lOffsetTop);
功能：

设置图象窗口里的图象距窗口顶部边缘的垂直距离。

参数：

lOffsetTop =图象偏移量

BOOL **SetOperation (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOperation**);**
功能：

确定对象是否受控制。

参数：

bOperation =对象可被控制：是/否

TRUE表示对象可被控制。
FALSE表示对象不能被控制。

注意:
如果函数的调用与整个图象相关，则必须令lpszObjectName参数=空。

BOOL **SetOperationMessage (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOperationMessage**);**
适用于输入/输出域、复选框、选项按钮和滑块对象
功能:

确定当操作成功的实现时是否输出信息。

参数：

bOperationMessage = 当操作成功的实现时输出信息：是/否

TRUE表示当操作成功的实现时输出信息。
FALSE表示当操作成功的实现时不输出任何信息。

BOOL **SetOperationReport (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bOperationReport**);**
适用于除了应用软件窗口、图象窗口和OLE控制以外的所有对象。
功能：

确定是否报告对象中的操作员活动的原因。

参数：

bOperationReport = 操作员活动的原因被报告：是/否

TRUE表示报告操作员活动的原因。
FALSE表示不报告操作员活动的原因。

注意:
如果函数的调用与整个图象相关，则必须令lpszObjectName参数=空。

BOOL **SetPasswordLevel (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPasswordLevel**);**
适用于除了应用软件窗口、图象窗口和OLE控制以外的所有对象。
功能：

设置对象的控制权限级别。

参数：

lPasswordLevel =授权级别
注意:
如果函数的调用与整个图象相关，则必须令lpszObjectName参数=空。

BOOL **SetPictureName (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureName**);**
功能：

设置图象对象里所包含的图象名称的指针。

参数：

szPictureName =图象名称的指针

BOOL **SetProcess (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dProcess**);**
适用于滑块、棒图、复选框和选项组对象。
功能：

对于滑块和棒图：
设置将要显示的过程值的预置值。

对于复选框和选项组：
所选框。

参数：

dProcess =缺省值

BOOL **SetSmallChange (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lSmallChange**);**
功能：

设置通过鼠标器单击来移动滑块的步距数。

参数：

lSmallChange =步距数设置

BOOL **SetTrend (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bTrend**);**
功能：

设置棒图对象里的趋势显示。

参数：

bTrend =显示趋势：是/否
TRUE表示棒图对象里显示趋势。
FALSE表示棒图对象里不显示趋势。

BOOL **SetVisible (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bVisible**);**
功能：

确定是否显示对象。

参数：

bVisible =对象显示：是/否

TRUE表示显示对象。
FALSE表示不显示对象。

注意:
如果函数的调用与整个图象相关，则必须令lpszObjectName参数=空。

BOOL **SetZeroPointValue (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, double dZeroPointValue**);**
功能：

设置棒图显示的零点的绝对值。

参数：

dZeroPointValue =零点的绝对值

BOOL **SetZoom (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lZoom**);**
功能：

设置图象窗口的缩放因子。

参数：

lZoom =缩放因子

4.4.29 图形设置OLE控制函数

BOOL **SetPosition(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPosition**)**;
功能：

设置OCX滑块的滑块位置。

参数：

lPosition　　=OCX滑块的滑块位置

BOOL **SetRangeMax(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRangeMax**);**
功能：

设置OCX滑块的最大设置范围。

参数：

lRangeMax　　=OCX滑块的最大设置范围

BOOL **SetRangeMin(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lRangeMin**);**
功能：

设置OCX滑块的最小设置范围。

参数：

lRangeMin　　=OCX滑块的最小设置范围

4.4.30 图形设置图象函数

BOOL **SetPictureDeactivated(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureDeactivated**);**
功能：

设置循环按钮的“已释放”状态的图象名称。
位图文件(*.bmp，*.dib)以及图元文件(*.emf，*.wmf)已链接。

参数：

szPictureDeactivated =“已释放”状态的图象名称

BOOL **SetPictureDown(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureDown**);**
功能：

设置循环按钮的“打开/已按下”状态的图象名称。
位图文件(*.bmp，*.dib)以及图元文件(*.emf，*.wmf)已链接。

参数：

szPictureDown =“打开/已按下”状态的图象名称

BOOL **SetPictureUp(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, char* szPictureUp**);**
功能：

设置循环按钮的“关闭/未按下”状态的图象名称。
位图文件(*.bmp，*.dib)以及图元文件(*.emf，*.wmf)已链接。

参数：

szPictureUp =“关闭/未按下”状态的图象名称

BOOL **SetPicDeactTransparent(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPicDeactTransparent**);**
功能：

设置循环按钮的“已释放”状态的透明颜色。

注意:
此函数只适用于位图图形。
参数：

lPicDeactTransparent =“已释放”状态的透明颜色

BOOL **SetPicDeactUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPicDeactUseTransColor**);**
功能：

控制循环按钮的“已释放”状态的透明颜色

参数：

bPicDeactUseTransColor =透明颜色是/否

TRUE 使用“已释放”状态的透明颜色。
FALSE 不使用“已释放”状态的透明颜色。

BOOL **SetPicDownTransparent(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPicDownTransparent**);**
功能：

设置循环按钮的“打开/已按下”状态的透明颜色。

注意:
此函数只适用于位图图形。

参数：

lPicDownTransparent =“打开/已按下”状态的透明颜色

BOOL **SetPicDownUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPicDownUseTransColor**);**
功能:

控制循环按钮的“打开/已按下”状态的透明颜色。

参数：

bPicDownUseTransColor =透明颜色是/否

TRUE 使用“打开/已按下”状态的透明颜色。
FALSE 不使用“打开/已按下”状态的透明颜色。

BOOL **SetPicTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPicTransColor**);**
功能：

设置图形对象的背景图象的透明颜色。

注意:
此函数只适用于位图图形。

参数：

lPicTransColor　=背景图象的透明颜色

BOOL **SetPicUpTransparent(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lPicUpTransparent**);**
功能：

设置循环按钮的“关闭/未按下”状态的透明颜色。

注意:
此函数只适用于位图图形。

参数：

lPicUpTransparent　=“关闭/未按下”的透明颜色

BOOL **SetPicUpUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPicUpUseTransColor**);**
功能：

控制循环按钮的“关闭/未按下”状态的透明颜色。

参数：

bPicUpUseTransColor =透明颜色是/否

TRUE 使用“关闭/未按下”状态的透明颜色。
FALSE 不使用“关闭/未按下”状态的透明颜色。

BOOL **SetPicUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPicUseTransColor**);**
功能：

控制图形对象的背景图象的透明颜色。
参数：

bPicUseTransColor =透明颜色是/否

TRUE 使用背景图象的透明颜色。
FALSE 不使用背景图象的透明颜色。

4.4.31 图形设置属性函数

BOOL **SetPropBOOL(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, BOOL bValue**)**
功能：

将属性设置为带有值bValue。

参数：

lpszPropertyName =属性的OLE自动化名称
bValue =值(TRUE，FALSE)

实例：

SetPropBOOL("Picture1","CustomizedObject1","Visible1",FALSE);
//返回类型为BOOL

图象“Picture1”里的自定义对象“CustomizedObject1”的OLE自动化名称“Visible1”的自定义属性被设置为FALSE。

BOOL **SetPropChar(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, char* szValue**)**
功能：

将属性值设置给所指向的指针szValue 。

参数：

lpszPropertyName =属性的OLE自动化名称
szValue =值上的Zeiger

BOOL **SetPropDouble(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, double dValue**)**
功能：

将属性设置为带有值dValue。

参数：

lpszPropertyName =属性的OLE自动化名称
dValue =值

BOOL **SetPropWord(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, long lValue**)**
功能：

将属性设置为带有值lValue。

参数：

lpszPropertyName =属性的OLE自动化名称
lValue =值

4.4.32 图形设置状态函数

BOOL **SetBasePicTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBasePicTransColor**);**
功能：

设置状态显示的原始图象的透明颜色。

注意:
此函数只适用于位图图形。

参数：

lBasePicTransColor =原始图象的透明颜色

BOOL **SetBasePicUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bBasePicUseTransColor**);**
功能：

控制状态显示的原始图象的透明颜色。

参数：

bBasePicUseTransColor =透明颜色是/否

TRUE 使用原始图象里的透明颜色。
FALSE 不使用原始图象里的透明颜色。

BOOL **SetFlashFlashPicture(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashFlashPicture**);**
功能：

控制状态显示里的闪烁图象的闪烁。

参数：

bFlashFlashPicture =透明颜色是/否

TRUE 动态的闪烁图象
FALSE 静态的闪烁图象

BOOL **SetFlashPicTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashPicTransColor**);**
功能：

设置状态显示的闪烁图象的透明颜色。

注意：
此函数只适用于位图图形。

参数：

lFlashPicTransColor =闪烁图象里的透明颜色

BOOL **SetFlashPicUseTransColor(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bFlashPicUseTransColor**);**
功能：

控制状态显示的闪烁图象的透明颜色。

参数：

bFlashPicUseTransColor =透明颜色是/否

TRUE 使用闪烁图象里的透明颜色。
FALSE 不使用闪烁图象里的透明颜色。

BOOL **SetFlashRateFlashPic(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFlashRateFlashPic**);**
功能：

设置状态显示里的闪烁图象的闪烁频率。

参数：

lFlashRateFlashPic =闪烁图象的闪烁频率

0： 无闪烁频率
1： 闪烁频率慢(大约0.5赫兹)
2： 闪烁频率中等(大约2赫兹)
3： 闪烁频率快(大约8赫兹)

注意：
因为闪烁是由软件完成的，故频率取决于系统和硬件(对象数目、处理器、存储器、更新时间等)。

BOOL **SetIndex(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lIndex);
功能：

设置多边形或折线的下标，并由此设置了对象的当前点。

参数：

lIndex =下标值

4.4.33 图形设置样式函数

BOOL **SetBackBorderWidth(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBackBorderWidth**);**
功能：

设置三维边框和滑块对象的边框的权重。

参数：

lBackBorderWidth =边框权重

BOOL **SetBorderEndStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderEndStyle**);**
功能：

设置线尾样式。

参数：

lBorderEndStyle =线尾样式的编号值

BOOL **SetBorderStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderStyle**);**
功能：

设置线条或边框的样式。

参数：

lBorderStyle =线条或边框样式的编号值

BOOL **SetBorderWidth (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBorderWidth**);**
功能：

设置线条或边框的权重。

参数：

lBorderWidth =线条或边框权重的编号值

BOOL **SetBoxAlignment (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lBoxAlignment**);**
功能：

向左或向右对齐复选框或选项按钮里的控制单元。

参数：

lBoxAlignment = 向左或向右对齐复选框或选项按钮里的控制单元的编号值

BOOL **SetFillStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillStyle**) ;**
功能：

设置填充模式。

参数：

lFillStyle =填充模式的类型的编号值

注意:
如果函数的调用与整个图象相关，则必须令lpszObjectName参数=空。

BOOL **SetFillStyle2 (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long int lFillStyle2**);**
功能：

设置棒图显示里的棒图填充模式。

参数：

lFillStyle2 = 棒图显示里的棒图填充模式的编号值

BOOL **SetItem BorderStyle(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long ind lItemBorderStyle**);**
功能：

设置“文本列表”对象的分隔线类型。

参数：

lItemBorderStyle =分隔线的编号值

BOOL **SetItem BorderWidth(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, long ind lItemBorderWidth**);**
功能：

设置“文本列表”对象的分隔线权重。

参数：

lItemBorderWidth =分隔线权重的编号值

BOOL **SetPressed(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bPressed**);**
只适用于循环按钮
功能：

控制循环按钮的开关设置。

参数：

bPressed =循环按钮的开关设置

TRUE 开关设置“打开/已按下”
FALSE 开关设置“关闭/未按下”。

BOOL **SetToggle(**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bToggle**);**
只适用于循环按钮
功能：

控制开关函数的锁存或未锁存。

参数：

bToggle　　=开关函数锁存或未锁存

TRUE　开关函数未锁存
FALSE　开关函数已锁存

BOOL **SetWindowsStyle (**LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, BOOL bWindowStyle**);**
只适用于按钮
功能：

打开或关闭“视窗样式”。
参数：

bWindowStyle　=“视窗样式”打开/关闭

TRUE对应于“视窗样式”，它表示按钮所显示的样式是合乎视窗标准的样式(无边框的灰色按钮)。
FALSE对应于“非视窗样式”，它表示可自己设置按钮的外观。

4.5 标签

标签函数类分成下述的几种功能:

获取函数

General get functions
get state functions
get wait functions
get state wait functions

设置函数

General set functions
set state functions
set wait functions
set state wait functions

等待函数不同于其它函数的原因在于标签value直接在过程里访问。标签没有被登录到控制中心中。
通过执行合适的获取等待或获取状态等待函数来代替获取或获取状态函数可明显的减少通讯网络的负载，因为在此情况下将不再周期性地搜索标签的变化。
由于系统将一直等待到等待函数被执行，因此可使过程更好的同步。

设置函数的返回值是错误代码
TRUE：无错误
FALSE：有错误产生

4.5.1 标签获取函数

short int **GetTagBit (**Tag Tag_Name**);**
返回值：

在“short int ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

BYTE **GetTagByte (**Tag Tag_Name**);**
返回值：

“BYTE ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

char* **GetTagChar (**Tag Tag_Name**);**
返回值：

“字符”数据类型里的过程标签的value的指针

参数：

Tag_Name =过程标签的名称

double **GetTagDouble (**Tag Tag_Name**);**
返回值：

“double ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

DWORD **GetTagDWord (**Tag Tag_Name**);**
返回值：

“DWORD ”数据类型里的过程标签的值
参数：

Tag_Name =过程标签的名称

float **GetTagFloat (**Tag Tag_Name**);**
返回值：

“float ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

BOOL **GetTagRaw (**Tag Tag_Name, BYTE* pValue, DWORD size**);**
返回值：

错误代码
TRUE=无错误
FALSE=有错误产生

功能:

从原始数据类型中获取value。

参数：

Tag_Name =过程标签的名称
pValue =包含原始数据的BYTE 框的BYTE 指针
size =以BYTE 为单位的框的尺寸

signed char **GetTagSByte(**Tag Tag_Name**);**
返回值：

“signed char ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

long int **GetTagSDWord(**Tag Tag_Name**);**
返回值：

“long int ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

short int **GetTagSWord(**Tag Tag_Name**);**
返回值：

“short int ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称

BOOL **GetTagValue (**LPDM_VARKEY lpdmVarKey, LPDM_VAR_UPDATE_STRUCT lpdmresult, LPCMN_ERROR lpdmError**);**
返回值：

错误代码
TRUE=无错误
FALSE=有错误产生

功能:

激活变量形式的value的传送。
获取包含值的结果结构的指针。

参数：

lpdmVarKey = DM_VARKEY结构的指针
lpdmresult = “变量”数据类型的值的指针
lpdmError = 包含错误描述的结构的指针

WORD **GetTagWord (**Tag Tag_Name**);**
返回值：

“字”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

4.5.2 标签获取状态函数

short int **GetTagBitState (**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“short int ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

BYTE **GetTagByteState (**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“BYTE ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

char* **GetTagCharState (**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“字符”数据类型里的过程标签的value的指针

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

double **GetTagDoubleState (**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“double ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

DWORD **GetTagDWordState (**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“DWORD ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

float **GetTagFloatState (**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“float ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

BOOL **GetTagRawState (**Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate**);**
返回值：

错误代码
TRUE=无错误
FALSE=有错误产生

功能：

原始数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称
pValue =包含原始数据的BYTE 框的BYTE 指针
size =以BYTE 为单位的框的尺寸
lp_dwstate =标签的状态

signed char **GetTagSByteState(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“signed char ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

long int **GetTagSDWordState(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“long int ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

short int **GetTagSWordState(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“short int ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

WORD **GetTagWordState (**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“字”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

4.5.3 标签获取等待函数

VARIANT_BOOL **GetTagBitWait(**Tag Tag_Name**);**
返回值：

“VARIANT_BOOL ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称

BYTE **GetTagByteWait(**Tag Tag_Name**);**
返回值：

“BYTE ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

char* **GetTagCharWait(**Tag Tag_Name**);**
返回值：

“字符”数据类型里的过程标签的value的指针

参数：

Tag_Name =过程标签的名称

double **GetTagDoubleWait(**Tag Tag_Name**);**
返回值：

“double ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

DWORD **GetTagDWordWait(**Tag Tag_Name**);**
返回值：

“DWORD ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

float **GetTagFloatWait(**Tag Tag_Name**);**
返回值：

“float ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

BOOL **GetTagMultiWait(**const char* pFormat,...**)**
返回值：

TRUE =无错误产生
FALSE =有错误产生

功能：

在相关地址上以已指明的格式确定和存储多种标签的值。

参数：

pFormat = 类似函数打印文件的格式描述。
跟着的是标签名称和地址的值。

实例：

```
DWORD dwTag1Value;
char* szTag2Value; // Storage for the tag value is created by the function with SysMalloc
BOOL ok;
ok=GetTagMultiWait("%d%s,"Tag1",&dwTag1Value,"Tag2",
&szTag2Value);
```

BOOL **GetTagRawWait(**Tag Tag_Name , BYTE pValue[], DWORD size**);**
返回值：

TRUE=无错误
FALSE=有错误产生
功能：

从原始数据类型中获取value。

参数：

Tag_Name =过程标签的名称
pValue =包含原始数据的BYTE 框的BYTE 指针
size =以BYTE 为单位的框的尺寸

signed char **GetTagSByteWait(**Tag Tag_Name**);**
返回值：

“signed char ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

long int **GetTagSDWordWait(**Tag Tag_Name**);**
返回值：

“long int ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称

short int **GetTagSWordWait(**Tag Tag_Name**);**
返回值：

“short int ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称

BOOL **GetTagValueWait(**LPDM_VARKEY lpdmVarKey, LPDM_VAR_UPDATE_STRUCT lpdmresult, LPCMN_ERROR lpdmError**);**
返回值：

TRUE=无错误
FALSE=有错误产生

功能:

允许以变量的形式发送value。
获取包含值的结果结构的指针。

参数：

lpdmVarKey = DM_VARKEY结构的指针
lpdmresult = “变量”数据类型的值的指针
lpdmError = 包含错误描述的结构的指针

WORD **GetTagWordWait(**Tag Tag_Name**);**
返回值：

“字”数据类型里的过程标签的值

参数：

Tag_Name　=过程标签的名称

4.5.4 标签获取状态等待函数

变量_BOOL **GetTagBitStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“VARIANT_BOOL ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

BYTE **GetTagByteStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“BYTE ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

char* **GetTagCharStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“字符”数据类型里的过程标签的value的指针。

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

double **GetTagDoubleStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“double ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

DWORD **GetTagDWordStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“DWORD ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

float **GetTagFloatStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“float ”数据类型里的过程标签的值。

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

BOOL **GetTagMultiStateWait**(DWORD* pdwState, const char* pFormat**)**
返回值：

TRUE =无错误产生
FALSE =有错误产生

功能:

在相关地址上以已指明的格式确定和存储多种标签的值和状态。

参数：

pdwState = 域，其上存储标签状态
pFormat = 类似函数打印文件的格式描述。
跟着的是标签名称和地址的值。

实例：

```
DWORD dwSate[2];
DWORD dwTag1Value;
char* szTag2Value; //Storage for the tag value is created by
带有SysMalloc的函数
BOOL ok;
ok=GetTagMultiStateWait("%d%s,&dwSate,"Tag1",
&dwTag1Value,"Tag2",&szTag2Value);
```

BOOL **GetTagRawStateWait(**Tag Tag_Name, BYTE pValue[], DWORD size, PDWORD lp_dwstate**);**
返回值：

TRUE=无错误
FALSE=有错误产生

功能：

确定原始数据类型的值。

参数：

Tag_Name =过程标签的名称
pValue =包含原始数据的BYTE 框的BYTE 指针
size =以BYTE 为单位的框的尺寸
lp_dwstate =标签的状态

signed char **GetTagSByteStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“signed char ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

long int **GetTagSDWordStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“long int ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

short int **GetTagSWordStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“short int ”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

字**GetTagWordStateWait(**Tag Tag_Name, PDWORD lp_dwstate**);**
返回值：

“字”数据类型里的过程标签的值

参数：

Tag_Name =过程标签的名称
lp_dwstate =标签的状态

4.5.5 标签设置函数

BOOL **SetTagBit (**Tag Tag_Name, short int value**);**
功能：

设置带有“short int ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagByte (**Tag Tag_Name, BYTE value**);**
功能：

设置带有“BYTE ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagChar (**Tag Tag_Name, LPSTR value**);**
功能：

设置带有“字符”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签值的指针

BOOL **SetTagDouble (**Tag Tag_Name, double value**);**
功能：

设置带有“double ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagDWord (**Tag Tag_Name, DWORD value**);**
功能：

设置带有“DWORD ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagFloat (**Tag Tag_Name, float value**);**
功能：

设置带有“float ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagRaw (**Tag Tag_Name, BYTE* pValue, DWORD size**);**
功能：

设置过程标签的value，该过程标签所带有的值通过指针pValue来寻址且具有“size”的长度。

参数：

Tag_Name = 标签的名称
pValue = 包含原始数据标签的值的BYTE 框的指针
size = 框的尺寸

BOOL **SetTagSByte(**Tag Tag_Name, signed char value**);**
功能：

设置带有“signed char ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagSDWord(**Tag Tag_Name, long value**);**
功能：

设置带有“long int ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagSWord(**Tag Tag_Name, short value**);**
功能：

设置带有“short int ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagValue (**LPDM_VARKEY lpdmVarKey, LPVARIANT lpdmValue, PDWORD dwState, LPCMN_ERROR lpdmError**);**
功能：

激活以变量形式的value的传送。
设置“变量”数据类型的值的指针。

参数：

lpdmVarKey = “DM_VARKEY”结构的指针
lpdmValue = “变量”数据类型的值的指针
dwState = 标签的状态。状态返回发生在函数执行之后。
lpdmError = 包含错误描述的结构的指针

BOOL **SetTagWord (**Tag Tag_Name, WORD value**);**
功能：

设置带有“字”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

4.5.6 标签设置状态函数

BOOL **SetTagBitState (**Tag Tag_Name, short int value, PDWORD lp_dwstate**);**
功能：

设置带有“short int ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagByteState (**Tag Tag_Name, BYTE value, PDWORD lp_dwstate**);**
功能：

设置带有“BYTE ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagCharState (**Tag Tag_Name, LPSTR value, PDWORD lp_dwstate**);**
功能：

设置带有“字符”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagDoubleState (**Tag Tag_Name, double value, PDWORD lp_dwstate**);**
功能：

设置带有“double ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagDWordState (**Tag Tag_Name, DWORD value, PDWORD lp_dwstate**);**
功能：

设置带有“DWORD ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagFloatState (**Tag Tag_Name, float value, PDWORD lp_dwstate**);**
功能：

设置带有“float ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagRawState (**Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate**);**
功能：

将过程标签的value设置给通过“pValue”指针寻址并具有“size”长度的某个值。
参数：

Tag_Name =标签的名称
pValue =值的指针
size =值的长度
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagSByteState(**Tag Tag_Name, signed char value, PDWORD lp_dwstate**);**
功能：

设置带有“signed char ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagSDWordState(**Tag Tag_Name, long value, PDWORD lp_dwstate**);**
功能：

设置带有“long int ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagSWordState(**Tag Tag_Name, short value, PDWORD lp_dwstate**);**
功能：

设置带有“short int ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagWordState (**Tag Tag_Name, WORD value, PDWORD lp_dwstate**);**
功能：

设置带有“字”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

4.5.7 标签设置等待函数

BOOL **SetTagBitWait(**Tag Tag_Name, short value**);**
功能：

设置带有“short int ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagByteWait(**Tag Tag_Name, BYTE value**);**
功能：

设置带有“BYTE ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagCharWait(**Tag Tag_Name, LPSTR value**);**
功能：

设置带有“LPSTR”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签值的指针

BOOL **SetTagDoubleWait(**Tag Tag_Name, double value**);**
功能：

设置带有“double ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagDWordWait(**Tag Tag_Name, DWORD value**);**
功能：

设置带有“DWORD ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagFloatWait(**Tag Tag_Name, float value**);**
功能：

设置带有“float ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagMultiWait(**const char* pFormat,...**)**
返回值：

TRUE =无错误产生
FALSE =有错误产生

功能：

以已指明的格式设置多种标签的值。

参数：

pFormat = 类似函数打印文件的格式描述。
跟着的是标签名称和相关的值。

实例：

```
BOOL ok;
ok=SetTagMultiWait("%d%s,"Tag1",34,
"Tag2","newValue");
```

BOOL **SetTagRawWait(**Tag Tag_Name, BYTE pValue[], DWORD size**);**
功能：

设置过程标签的value，该过程标签带有的值可通过“pValue”指针寻址且具有“size”长度。

参数：

Tag_Name = 标签的名称
pValue = 包含原始数据标签的值的BYTE 框的指针
size = 框的尺寸

BOOL **SetTagSByteWait(**Tag Tag_Name, signed char value**);**
功能：

设置带有“signed char ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagSDWordWait(**Tag Tag_Name, long value**);**
功能：

设置带有“long int ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagSWordWait(**Tag Tag_Name, short value**);**
功能：

设置带有“short int ”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

BOOL **SetTagValueWait(**LPDM_VARKEY lpdmVarKey, LPVARIANT lpdmValue, PDWORD dwState, LPCMN_ERROR lpdmError**);**
功能：

允许以变量的形式发送value。
设置“变量”数据类型的值的指针。

参数：

lpdmVarKey = DM_VARKEY结构的指针
lpdmValue = “变量”数据类型的值的指针
dwState = 在函数执行之后返回的标签状态
lpdmError = 包含错误描述的结构的指针

BOOL **SetTagWordWait(**Tag Tag_Name, WORD value**);**
功能：

设置带有“字”数据类型值的过程标签值。

参数：

Tag_Name =标签的名称
value =标签的值

4.5.8 标签设置状态等待函数

BOOL **SetTagBitStateWait(**Tag Tag_Name, short value, PDWORD lp_dwstate**);**
功能：

设置带有“short int ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagByteStateWait(**Tag Tag_Name, BYTE value, PDWORD lp_dwstate**);**
功能：

设置带有“BYTE ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagCharStateWait(**Tag Tag_Name, LPSTR value, PDWORD lp_dwstate**);**
功能：

设置带有“LPSTR”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签值的指针
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagDoubleStateWait(**Tag Tag_Name, double value, PDWORD lp_dwstate**);**
功能：

设置带有“double ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagDWordStateWait(**Tag Tag_Name, DWORD value, PDWORD lp_dwstate**);**
功能：

设置带有“DWORD ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagFloatStateWait(**Tag Tag_Name, float value, PDWORD lp_dwstate**);**
功能：

设置带有“float ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagMultiStateWait(**DWORD* pdwState, const char* pFormat,...**)**
返回值：

TRUE =无错误产生
FALSE =有错误产生
功能：

以已指明的格式存储多种标签的值和状态。

参数：

pdwState = 域，其上存储标签的状态
pFormat = 类似函数打印文件的格式描述。
跟着的是标签名称和值。

实例：

```
DWORD dwState[3];
BOOL ok;
ok=SetTagMultiStateWait("%d%s%f,&dwState[0],"Tag1",34,
"Tag2","newValue",
"Tag3",4.67);
```

BOOL **SetTagRawStateWait(**Tag Tag_Name, BYTE pValue[], DWORD size, PDWORD lp_dwstate**);**
功能：

设置过程标签的value，该过程标签带有的值可通过“pValue”指针来寻址且具有“size”长度。

参数：

Tag_Name =标签的名称
pValue =值的指针
size =值的长度
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagSByteStateWait(**Tag Tag_Name, signed char value, PDWORD lp_dwstate**);**
功能：

设置带有“signed char ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagSDWordStateWait(**Tag Tag_Name, long value, PDWORD lp_dwstate**);**
功能：

设置带有“long int ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagSWordStateWait(**Tag Tag_Name, short value, PDWORD lp_dwstate);
功能：

设置带有“short int ”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

BOOL **SetTagWordStateWait(**Tag Tag_Name, WORD value, PDWORD lp_dwstate**);**
功能：

设置带有“字”数据类型值的过程标签值。“lp_dwstate”包含了函数执行之后的标签状态。

参数：

Tag_Name =标签的名称
value =标签的值
lp_dwstate =函数执行之后的标签状态

4.6 视窗控制中心(WinCC)

“视窗控制中心”函数类分成下述的几种功能:

system

函数的返回值是下述的错误代码中的：
TRUE：函数执行时不带有错误。
FALSE：有错误产生。

4.6.1 视窗控制中心（WinCC）系统

BOOL **DeactivateRTProject ();**
功能：

释放当前正在运行的项目。

BOOL **ExitWinCC ();**
功能：

退出视窗控制中心。

DWORD **GetLanguage();**

返回值：

在运行时所设置语言的语言标识符。

功能：

传送所设置的语言。

采用下述的分配(语言标识符的十六进制代码)：
德语(德国)=0x407
英语(美国)=0x409
法语(法国)=0x40C

对于附加的语言标识符参见6.1节。

DWORD *InquireLanguage(DWORD* dwCount);

返回值：

包含已发送语言标识符的框的指针。

功能:

传送运行时的文本库里所配置的所有语言。使用dwCount来指规定已发送的语言标识符的编号应存储在何处。

采用下述的分配(语言标识符的十六进制代码)：
德语(德国)=0x407
英语(美国)=0x409
法语(法国)=0x40C

对于附加的语言标识符参见6.1节。

参数：

dwCount =已发送的语言标识符编号的指针

BOOL SetLanguage(DWORD dwLocaleID);
功能：

通过所提供的语言标识符设置运行时的当前语言。

采用下述的分配(语言标识符的十六进制代码)：
德语(德国)=0x407
英语(美国)=0x409
法语(法国)=0x40C

对于附加的语言标识符参见6.1节。

参数：

dwLocaleID =语言标识符

5 动作

全局脚本动作是指可以自己创建和修改的那些动作。只在创建它们的实际项目中才有效。

在运行时使用全局脚本动作来对过程进行控制。通过触发执行。

启动控制中心里的全局脚本编辑器，可以创建和编辑全局脚本动作。

使用下列步骤可创建一个全局脚本动作：

1. 制定函数
2. 扩展函数信息
3. 设置启动触发事件。
4. 编译动作
5. 设置操作员授权。
6. 保存动作，如有必要，将其重新命名。

完成创建全局脚本动作。

对全局脚本动作可进行输出和输入。所输入的动作将完全代替活动窗口里的动作。

可给动作分配一个授权等级。

如果创建了新的动作，则在项目文件夹...\<Projectname>\LIBRARY中自动地集成头文件apdefap.h。如果没有任何可用的项目函数，则集成来自文件夹...\APLIB的头文件。这可使动作中的标准函数和项目函数变成可用的函数。

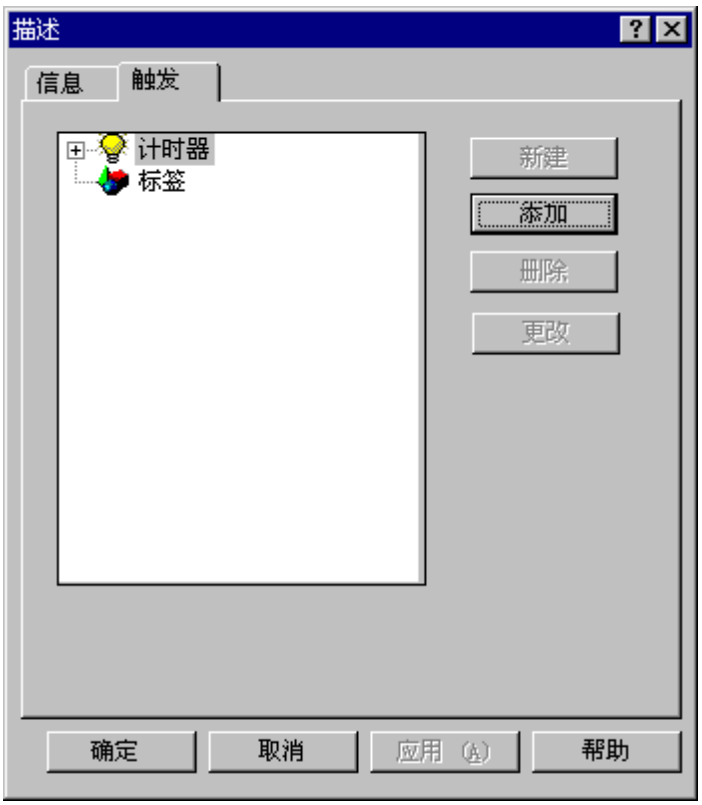
对于多用户系统，在全局动作和局部动作之间存在着差异。

全局动作载入项目文件夹不依赖于计算机(...\<Projectname>\PAS)。它们在任何计算机上均是激活的。

局部动作则根据项目目录(...\<Projectname>\<Computername>\PAS)中的计算机来载入。也就是说局部动作只在相关的计算机上才是激活的。

如何设置触发：

1. 如果同时打开了多个编辑窗口，则选择期望动作所位于的窗口，使该窗口激活。
2. 单击  按钮可打开“描述”对话框。
3. 转换到“触发”栏。



可从下述各种触发类型中选择：

- 非周期的：动作只在所定义的时间点上触发。
- 周期性的：动作可在所设置的一个时帧里连续地触发。
- 标签：当某个确定标签的值变化时动作才被触发。

如何选择触发类型：

1. 选择“描述”对话框里的期望触发类型并单击“添加...”按钮。

   或

   在触发类型上单击鼠标右键并选择弹出菜单里的“添加”。

   如果必要的话，打开“定时器”文件夹。

   打开“添加触发”对话框。它包括说明所选触发类型的内容。

2. 作出下列附加的选择：

   非周期的：

   设置触发的日期和时间。输入可选的触发名称。触发名称和图标显示在触发浏览器的“描述”对话框里。如果没有输入触发名称，则所创建触发的日期和时间与图标一同显示。

   周期性的：

   从“周期”列表中选择所期望的时间，即两个动作触发之间所经历的时间。输入一个可选触发的名称。触发名称和图标显示在触发浏览器的“描述”对话框里。。如果没有输入触发名称，则所设置的周期时间与图标一同显示。

   标签：

   a. 在“标签名称”文本行里，输入当值发生变化时用作触发的标签名称。

   b. 单击“应用”按钮将标签传送给标签列表。

   单击  按钮打开“选择标签”对话，从中选择一个标签。

   单击“确定”关闭“添加触发”对话框。

3. 单击“确定”按钮关闭“描述”对话框。

所有输入到触发浏览器里的触发均是有效的。各个单个触发事件均可启动动作。

实例：

- 在“标签”触发类型上，输入作为触发的多个标签。其中的某个标签的值一发生变化，动作就启动。
- 在“周期性的”触发类型上，输入多个触发周期。其中的各个触发均能启动动作本身。如果几个触发事件并存，则动作启动的次数与各时间点的触发事件启动动作的次数相同。

注意：由于将多重触发事件分配给动作，因此单个触发与动作运行时的相互作用可能导致不期望出现的后果。分配触发时应将此考虑进去。

修改所输入的某个触发，可通过在触发浏览器里选择并单击“修改”按钮(或使用弹出菜单里的“修改”菜单项)来进行。

删除所输入的触发，可通过在触发浏览器里选择并单击“删除”按钮(或使用弹出菜单里的“删除”菜单项)来进行。

通过单击“确定”按钮来关闭“描述”框。

如何设置操作员授权：

1. 如果同时打开多个编辑窗口，则可选择期望分配授权等级的窗口，此选择可激活该窗口。
2. 单击  按钮打开“授权等级”对话框。
3. 从授权等级列表中选择所期望的授权等级。
4. 单击“确定”按钮关闭对话框。

完成设置操作员权限。

6 属性的值定义

属性名称：

- 语言标识
- 颜色
- 线尾风格
- 线风格
- 闪烁频率
- 文本定位
- 栏定位
- 输入/输出域、域类型
- 输入/输出域、域内容的数据类型
- 复选框和单选框里的单元定位

在C函数中使用符号名或相应编号值作为参数。

6.1 语言标识符

视窗控制中心只支持视窗规定的各种缺省语言。

符号描述	值(十六进制)
LANG_ARABIC	0x0401
LANG_AFRIKAANS	0x0436
LANG_ALBANIAN	0x041C
LANG_BASQUE	0x042D
LANG_BULGARIAN	0x0402
LANG_BYELORUSSIAN	0x0423
LANG_CATALAN	0x0403
LANG_CHINESE	0x0404
LANG_CROATIAN	0x041A
LANG_CZECH	0x0405
LANG_DANISH	0x0406
LANG_DUTCH	0x0413
LANG_ENGLISH	0x0409
LANG_ESTONIAN	0x0425
LANG_FAEROESE	0x0438
LANG_FARSI	0x0429
LANG_FINNISH	0x040B
LANG_FRENCH	0x040C
LANG_GERMAN	0x0407
LANG_GREEK	0x0408
LANG_HEBREW	0x040D
LANG_HUNGARIAN	0x040E
LANG_ICELANDIC	0x040F
LANG_INDONESIAN	0x0421
LANG_ITALIAN	0x0410
LANG_JAPANESE	0x0411
LANG_KOREAN	0x0412
LANG_LATVIAN	0x0426
LANG_LITHUANIAN	0x0427
LANG_NORWEGIAN	0x0414
LANG_POLISH	0x0415
LANG_PORTUGUESE	0x0416
LANG_ROMANIAN	0x0418
LANG_RUSSIAN	0x0419
LANG_SLOVAK	0x041B
LANG_SLOVENIAN	0x0424
LANG_SORBIAN	0x042E
LANG_SPANISH	0x040A
LANG_SWEDISH	0x041D
LANG_THAI	0x041E
LANG_TURKISH	0x041F
LANG_UKRAINIAN	0x0422

6.2 颜色

颜色	符号名	值(十六进制)
黑色	CO_BLACK	00000000
白色	CO_WHITE	00FFFFFF
红色	CO_RED	000000FF
深红色	CO_DKRED	00000080
绿色	CO_GREEN	0000FF00
深绿色	CO_DKGREEN	00008000
蓝色	CO_BLUE	00FF0000
深蓝色	CO_DKBLUE	00800000
黄色	CO_YELLOW	0000FFFF
深黄色	CO_DKYELLOW	00008080
青色	CO_CYAN	00FFFF00
深青色	CO_DKCYAN	00808000
绛红	CO_MAGENTA	00FF00FF
深绛红	CO_DKMAGENTA	00800080
浅灰色	CO_LTGRAY	00C0C0C0
深灰色	CO_DKGRAY	00808080

6.3 线尾风格

线尾	符号名	值
圆	LE_NO	0
	LE_HOLLOW_ARROW	1
	LE_FULL_ARROW	2
	LE_CFULL_ARROW	3
	LE_LINE	4
	LE_HOLLOW_CIRCLE	5
	LE_FULL_CIRCLE	6

实例：

```
long value;
value = LE_HOLLOW_ARROW;
value <<= 16;                    //左端是虚箭头
value += LE_HOLLOW_CIRCLE;       //右端是虚心圆的
```

6.4 线风格

线风格	符号名	值
———	LS_SOLID	0
— —	LS_DASH	1
- - - - -	LS_DOT	2
- - —	LS_DASHDOT	3
- - - —	LS_DASHDOTDOT	4
隐藏形状	LS_INVISIBLE	5

6.5 闪烁频率

编号值	闪烁频率
0	闪烁关闭
1	0.5赫兹
2	2赫兹
3	8赫兹

因为闪烁是采用软件技术来实现的，所以频率依赖于系统和硬件(对象的编号、CPU、存储器、更新周期等)。因此，上述表格里的值仅供参考。

6.6 文本定位

定位	编号值
居左	0
居中	1
居右	2

6.7 栏定位

栏定位	编号值
顶部	0
底部	1
左边	2
右边	3

6.8 输入/输出域、域类型

类型	编号值
输出	0
输入	1
输出和输入	2

6.9 输入/输出域、域内容的数据类型

数据类型	编号值
二进制	0
十进制	1
字符串	2
十六进制	3

6.10 单选框和复选框里的单元定位

定位	编号值
左	0
右	1

Index

K

L

M

Q

S

T

X

Y

Z