# GRAPH控件

GRAPH控件可以用来显示图形化的数据。它的典型应用是显示的测量值或曲线图。可以同时显示多条曲线。可用水平和垂直刻度可以用来标记曲线。在背景上可以显示一个具有不同的水平和垂直网格间距的网格，如果数组值超出了控件的可见范围之内，该控件可以自动显示滚动条，通过滚动条可以查看到更多的数据。

1. GRAPH控件的结构

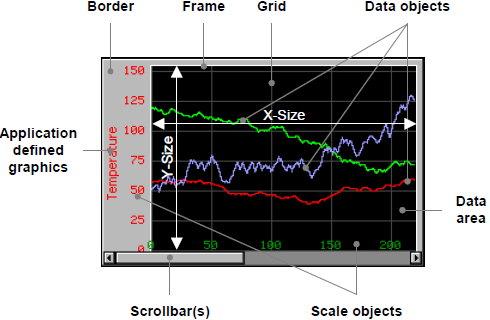
一个GRAPH控件包含不同类型的对象

●控件可附加自身的数据对象和比例尺对象

●可选一个或多个数据对象

●可选一个或多个比例尺对象

下图显示了GRAPH控件的详细结构



下表是对上图的详细说明

|  |  |
| --- | --- |
| 构件 | 描述 |
| Border | 可选的边框是GRAPH控件的一部分 |
| Frame | 环绕数据区的细线，GRAPH控件的一部分 |
| Grid | 显示在数据区的背景，GRAPH控件的一部分 |
| Data area | 一个区域，网格和数据对象可以显示在里面 |
| Data object(s) | 对于每一条曲线，可以给控件增加一个数据对象 |
| Application defined graphic | 一个定义回调函数的应用可以用来绘制任何应用程序定义的文本和/或图形。 |
| Scrollbar(s) | 当数据对象的范围超出了GRAPH控件的数据区域，控件可以自动显示水平和/或垂直的滚动条 |
| Scale object(s) | 可以附加在GRAPH控件的水平和垂直刻度 |
| X-Size | 数据区域的水平范围 |
| Y-Size | 数据区域的垂直范围 |

1. 创建和删除一个GRAPH控件

GRAPH控件的创建应按如下步骤

●创建GRAPH控件，并设置所需的属性

●创建数据对象

●将数据对象关联到GRAPH控件

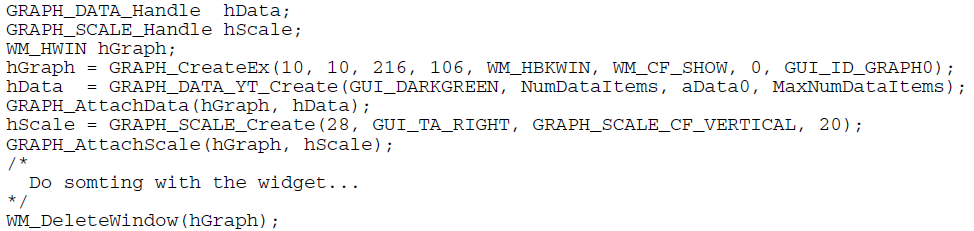
●创建可选的刻度尺对象

●将刻度尺对象关联到GRAPH控件

数据对象和刻度尺对象一旦关联到GRAPH控件，不需要在应用中删除他们，GRAPH控件会自动完成。

**举例**

下面的程序显示了如何创建和删除一个GRAPH控件



1. 绘图过程

如上所述GRAPH控件包含不同的不见和对象，下面解释该控件绘图的过程

●使用背景色填充背景

●调用可选的回调函数，这使它可以绘制诸如用户定义的网格

●绘制网格（如果使能）

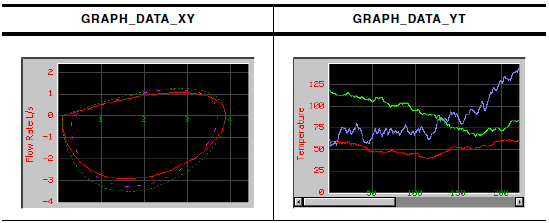
●绘制数据对象和边框区域

●绘制刻度尺对象

●调用可选的回调函数，这使它可以绘制诸如用户定义的刻度或一些附加文字和/或图形

1. 波形支持的类型

对于显示连续更新的测量值的要求，不同于显示一个带有X/Y坐标的函数波形的要求，因此该控件目前支持两种类型的数据对象，如下图



GRAPH\_DATA\_XY

数据对象用来显示包含一系列点组成的曲线，目标数据被绘制成一个多线段，一个典型应用是绘制函数波形。

GRAPH\_DATA\_YT

数据对象用来显示对于每一个X位置都带有一个Y值的波形，使用该数据对象的典型应用绘制连续更新的测量值波形。

1. 配置选项

5.1 GRAPH 控件

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **宏** | **缺省值** | **说明** |
| N | GRAPH\_BKCOLOR\_DEFAULT | GUI\_BLACK | 数据区域的缺省背景色 |
| N | GRAPH\_BORDERCOLOR\_DEFAULT | 0xC0C0C0 | 边框的缺省背景色 |
| N | GRAPH\_FRAMECOLOR\_DEFAULT | GUI\_WHITE | 框体线的缺省颜色 |
| N | GRAPH\_GRIDCOLOR\_DEFAULT | GUI\_DARKGRAY | 网格的缺省颜色 |
| N | GRAPH\_GRIDSPACING\_X\_DEFAULT | 50 | 网格的水平缺省间隔 |
| N | GRAPH\_GRIDSPACING\_Y\_DEFAULT | 50 | 网格的垂直缺省间隔 |
| N | GRAPH\_BORDER\_L\_DEFAULT | 0 | 左边框的缺省大小 |
| N | GRAPH\_BORDER\_T\_DEFAULT | 0 | 上边框的缺省大小 |
| N | GRAPH\_BORDER\_R\_DEFAULT | 0 | 右边框的缺省大小 |
| N | GRAPH\_BORDER\_B\_DEFAULT | 0 | 下边框的缺省大小 |

5.2 刻度尺对象

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **宏** | **缺省值** | **说明** |
| N | GRAPH\_SCALE\_TEXTCOLOR\_DEFAULT | GUI\_WHITE | 缺省文字颜色 |
| S | GRAPH\_SCALE\_FONT\_DEFAULT | &GUI\_Font6x8 | 缺省字体 |

1. 按键响应

该控件不能获得输入的焦点，也不能对按键输入做出响应

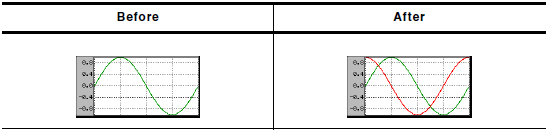
1. GRAPH API

下表按字母顺序列出可用的uC/GUI相关的函数，详细描述如下

|  |  |
| --- | --- |
| **函数** | **说明** |
| **公共函数** | |
| GRAPH\_AttachData() | 将数据对象附加到一个现有的GRAPH控件 |
| GRAPH\_AttachScale() | 将刻度尺对象附加到一个现有的GRAPH控件 |
| GRAPH\_CreateEx() | 创建一个GRAPH控件 |
| GRAPH\_SetBorder() | 设置边框（上、下、左、右）的尺寸 |
| GRAPH\_SetColor() | 设置GRAPH控件的颜色 |
| GRAPH\_SetGridDistX() | 设置水平网格间距 |
| GRAPH\_SetGridDistY() | 设置垂直网格艰巨 |
| GRAPH\_SetGridFixedX() | 在X轴上对网格修正 |
| GRAPH\_SetGridVis() | 使能网格绘制 |
| GRAPH\_SetLineStyleH() | 设置水平网格线的线型 |
| GRAPH\_SetLineStyleV() | 设置垂直网格线的线型 |
| GRAPH\_SetVSizeX() | 设置GRAPH控件的水平范围 |
| GRAPH\_SetVSizeY() | 设置GRAPH控件的垂直范围 |
| GRAPH\_SetUserDraw() | 设置用户回调函数 |
| **GRAPH\_DATA\_YT相关函数** | |
| GRAPH\_DATA\_YT\_AddValue() | 向数据对象增加一个数据项 |
| GRAPH\_DATA\_YT\_Create() | 创建一个GRAPH\_DATA\_YT对象 |
| GRAPH\_DATA\_YT\_SetOffY() | 设置数据绘制的垂直偏移量 |
| **GRAPH\_DATA\_XY相关函数** | |
| GRAPH\_DATA\_XY\_AddPoint() | 增加一个数据对象指针 |
| GRAPH\_DATA\_XY\_Create() | 创建一个GRAPH\_DATA\_XY对象 |
| GRAPH\_DATA\_XY\_SetLineStyle() | 设置用来绘制数据的线型 |
| GRAPH\_DATA\_XY\_SetOffX() | 设置绘制数据的水平偏移量 |
| GRAPH\_DATA\_XY\_SetOffY() | 设置绘制数据的垂直偏移量 |
| GRAPH\_DATA\_XY\_SetPenSize() | 设置绘制数据的笔的粗细 |
| **刻度尺相关函数** | |
| GRAPH\_SCALE\_Create() | 创建一个刻度尺对象 |
| GRAPH\_SCALE\_SetFactor() | 设置一个计算因子，用于像素转换到所需单位的计算 |
| GRAPH\_SCALE\_SetFixed() | 避免滚动刻度 |
| GRAPH\_SCALE\_SetFont() | 设置显示数字的字体 |
| GRAPH\_SCALE\_SetNumDecs() | 设置小数部分的位数 |
| GRAPH\_SCALE\_SetOff() | 设置对数量增加的一个可选的偏移量 |
| GRAPH\_SCALE\_SetPos() | 设置刻度水平或垂直的位置 |
| GRAPH\_SCALE\_SetTextColor() | 设置刻度的文字颜色 |
| GRAPH\_SCALE\_SetTickDist() | 设置刻度之间的距离（像素） |

7.1公共函数

GRAPH\_AttachData()

****

**描述**

向现有的一个GRAPH控件附加一个数据对象

**原型**

void GRAPH\_AddGraph(GRAPH\_Handle hObj, GRAPH\_DATA\_Handle hData);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| hData | 要的增加到控件的数据对象句柄，数据对象应使用GRAPH\_DATA\_YT\_Create() 或 GRAPH\_DATA\_XY\_Create()函数创建 |

**附加信息**

一旦附加到一个GRAPH控件，应用程序不需要破坏数据对象，GRAPH控件在自身被删除的时候会删除所有的附加数据对象，

关于如何创建数据对象的细节，请参考函数GRAPH\_DATA\_YT\_Create() 和 GRAPH\_DATA\_XY\_Create().

GRAPH\_AttchScale()



**描述**

向现有的一个GRAPH控件附加一个刻度尺对象

**原型**

void GRAPH\_AttchScale(GRAPH\_Handle hObj, GRAPH\_SCALE\_Handle hScale);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| hScale | 要的增加的刻度尺句柄 |

**附加信息**

一旦附加到一个GRAPH控件，应用程序不需要破坏刻度尺对象，GRAPH控件在自己被删除的时候会删除所有的附加数据对象，

关于如何创建刻度尺对象的细节，请参考函数GRAPH\_SCALE\_Create()。

GRAPH\_CreateEx()

**描述**

创建一个新的GRAPH控件

**原型**

GRAPH\_Handle GRAPH\_CreateEx(int x0, int y0,int xsize, int ysize, WM\_HWIN hParent,int WinFlags, int ExFlags, int Id);

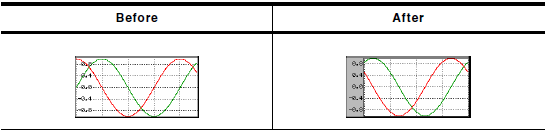
|  |  |
| --- | --- |
| **参数** | **含义** |
| X0 | 控件最左边的像素（相对父坐标） |
| Y0 | 控件最上边的像素（相对父坐标） |
| xsize | 控件的水平宽度（像素） |
| ysize | 控件的垂直高度（像素） |
| hParent | 父窗口的句柄，如果为0，则作为桌面（最顶层窗口）的子窗口 |
| winFLags | 窗口创建标志，典型值为WM\_CF\_SHOW使得控件立即可见，取值范围可参考窗口管理相关内容 |
| ExFlags | 见下面的表格 |
| Id | 控件的窗口Id |

|  |  |
| --- | --- |
| 参数x允许的取值 | |
| GRAPH\_CF\_GRID\_FIXED\_X | 在x轴上修复网格，即如果使用的水平滚动条，网格会保持在它的位置 |

**返回值**

所创建控件的句柄，如果创建失败，返回0

GRAPH\_SetBorder()

****

**描述**

设置给定的GRAPH控件的左、右、上、下边框

**原型**

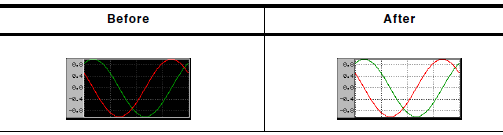
void GRAPH\_SetBorder(GRAPH\_Handle hObj,unsigned BorderL, unsigned BorderT,unsigned BorderR, unsigned BorderB);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| BorderL | 左边框大小像素值 |
| BorderT | 上边框大小像素值 |
| BorderR | 右边框大小像素值 |
| BorderB | 下边框大小像素值 |

**附加信息**

边框的大小是控件和数据区之间的像素值，如果边框大小至少有一个像素值，围绕着数据区的框架、细线是唯一可见的，有关如何设置边框颜色和瘦框架参考函数GRAPH\_SetColor()。

GRAPH\_SetColor()

****

**描述**

给GRAPH控件设置所需的颜色

**原型**

GUI\_COLOR GRAPH\_SetColor(GRAPH\_Handle hObj, GUI\_COLOR Color,unsigned Index);

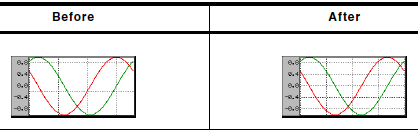
|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| Color | 控件所需的颜色 |
| Index | 见下表 |

|  |  |
| --- | --- |
| 参数Index允许的取值 | |
| GRAPH\_CI\_BK | 设置背景色 |
| GRAPH\_CI\_BOERDER | 设置边框区域的颜色 |
| GRAPH\_CI\_FRAME | 设置细框架线的颜色 |
| GRAPH\_CI\_GRID | 设置网格的颜色 |

**返回值**

返回所需条目之前的颜色

GRAPH\_SetGridDistX(),GRAPH\_SetGridDistY()

****

**描述**

设置网格线之间的距离

**原型**

unsigned GRAPH\_SetGridDistX(GRAPH\_Handle hObj, unsigned Value);

unsigned GRAPH\_SetGridDistY(GRAPH\_Handle hObj, unsigned Value)

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| Value | 网格线之间的距离，缺省为50像素 |

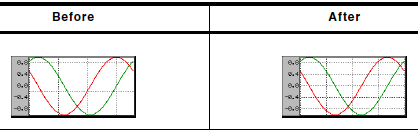
**返回值**

返回之前的线间距

**附加信息**

第一条竖线画在数据区最左边的位置，水平线画在数据区最底部的位置，除非使用了偏移量

GRAPH\_SetGridFixedX ()

****

**描述**

对X轴网格进行修正

**原型**

unsigned GRAPH\_SetGridFixedX(GRAPH\_Handle hObj, unsigned OnOff);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| onoff | 1=在X轴上对网格修正，0=不处理（缺省） |

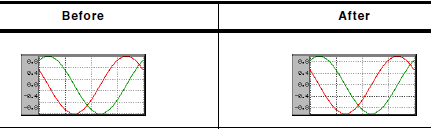
**返回值**

之前使用的值

**附加信息**

在某些情况下，在X坐标上修正网格是很有用的，一个典型的应用实在YT-图形，即连续的新值不断增加需要水平滚动条，这种情况下可以通过修正背景里的网格来解决，有关如何激活滚动条，请参考函数GRAPH\_SetVSizeX() and GRAPH\_SetVSizeY()。

GRAPH\_SetGridOffY ()

****

**描述**

增加一个偏移量用于显示水平方向的网格线

**原型**

unsigned GRAPH\_SetGridOffY(GRAPH\_Handle hObj, unsigned Value);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| Value | 偏移量 |

**返回值**

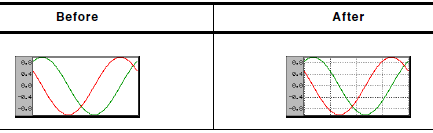
返回之前的偏移量

**附加信息**

渲染网格时，控件从数据区底部开始使用当前的间距画网格线，当纵坐标的零点位于Y轴的中间时，可能会出现中间没有网格线的情况，此时可以用此函数增加一个偏移量来进行Y轴上的转移，正值向下转移，负值向上转移。

有关网格间距的设置可参考函数GRAPH\_SetGridSpacingX()和 GRAPH\_SetGridSpacingY。

GRAPH\_SetGridVis ()

****

**描述**

设置网格线为可见

**原型**

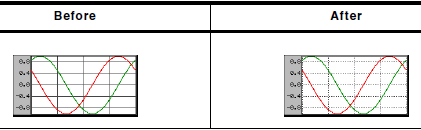
unsigned GRAPH\_SetGridVis(GRAPH\_Handle hObj, unsigned OnOff);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| OnOff | 1=网格可见，0=不可见（缺省） |

**返回值**

返回之前的可见值

GRAPH\_SetLineStyleH ()，GRAPH\_SetLineStyleH ()

****

**描述**

设置水平和垂直网格线的线型

**原型**

U8 GRAPH\_SetLineStyleH(GRAPH\_Handle hObj, U8 LineStyle);

U8 GRAPH\_SetLineStyleV(GRAPH\_Handle hObj, U8 LineStyle);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| LineStyle | 使用的线型，查看支持的线型可参考函数GUI\_SetLineStyle(),默认为GUI\_LS\_SOLID |

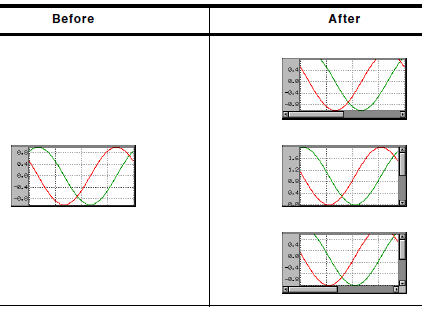
**返回值**

返回之前的线型

**附加信息**

注意：使用GUI\_LS\_SOLID之外的其他线型，绘制网格时需要花费更多的时间。

GRAPH\_SetVSizeX ()，GRAPH\_SetVSizeY()

****

**描述**

设置X轴和Y轴的虚拟区域大小

**原型**

unsigned GRAPH\_SetVSizeX(GRAPH\_Handle hObj, unsigned Value);

unsigned GRAPH\_SetVSizeY(GRAPH\_Handle hObj, unsigned Value);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| Value | X轴和Y轴的虚拟大小 |

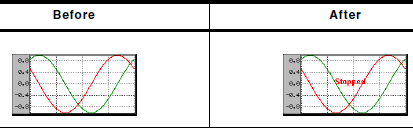
**返回值**

返回之前X轴或Y轴虚拟大小

**附加信息**

如果部件虚拟尺寸大于数据区的可视尺寸，部件自动显示滚动条。例如如果通过函数GRAPH\_DATA\_YT\_Create()创建一个数据对象， 包含的数据量超出数据区域显示显示范围，可使用GRAPH\_SetVSizeX()可以来使能滚动条，函数GRAPH\_SetVSizeX(NumDataItems)使能滚动条，提供了超出可视区域的X轴上的数据量。

GRAPH\_SetUserDrau()

****

**描述**

设置用户绘图函数，此函数可以在绘图过程中被控件调用，绘制出用户定义的数据。

**原型**

void GRAPH\_SetUserDraw(GRAPH\_Handle hObj,void (\* pUserDraw)(WM\_HWIN hObj, int Stage));

|  |  |
| --- | --- |
| **参数** | **含义** |
| hObj | 控件的句柄 |
| pUserDraw | 指向函数的指针，给函数在绘图过程中被控件调用 |

|  |  |
| --- | --- |
| 参数Stage允许的取值 | |
| GRAPH\_DRAW\_FIRST | 在填充背景色后调用，如绘制用户定义的网格 |
| GRAPH\_DRAW\_LAST | 在绘制波形之后调用，如用户定义的数据 |

**附加信息**

用户绘图函数在填充背景色之后被调用，以及绘制完所有的波形之后被调用，首次调用的裁剪区域被限制在数据区域内，最后调用被显示在除有效框架以外的全部图形控件区域。

**举例**

static void \_UserDraw(WM\_HWIN hWin, int Stage) {

switch (Stage) {

case GRAPH\_DRAW\_FIRST:

/\* Draw for example a user defined grid... \*/

break;

case GRAPH\_DRAW\_LAST:

/\* Draw for example a user defined scale or additional text... \*/

break;

}

}

static void \_CreateGraph(void) {

WM\_HWIN hGraph;

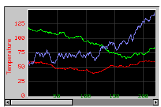
hGraph = GRAPH\_CreateEx(10, 10, 216, 106, WM\_HBKWIN, WM\_CF\_SHOW, 0, GUI\_ID\_GRAPH0);

GRAPH\_SetUserDraw(hGraph, \_UserDraw); /\* Enable user draw \*/

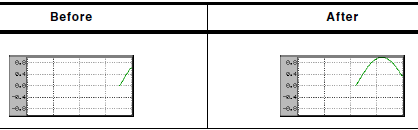
...

}

7.2 GRAPH\_DATA\_YT 相关函数

****

GRAPH\_DATA\_YT\_AddValue()

****

**描述**

向GRAPH\_DATA\_YT对象增加新的数据

**原型**

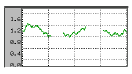
void GRAPH\_DATA\_YT\_AddValue(GRAPH\_DATA\_Handle hDataObj, I16 Value);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hDataObj | 数据对象的句柄 |
| Value | 增加到数据对象的值 |

**附加信息**

给定的值被添加到数据对象，如果数据对象是满的，意味着他包含了创建时在参数MaxNumItems中定义的同样多的数据，在增加一个新值前先转移数据，因此向一个满的对象增加一个新数据的时候，第一条数据就被转移出去了。

值0x7FFF可用来处理无效的数据，这些值在绘图时被排除在外，下面的截图显示了带有2个间隙的无效数据的波形。



GRAPH\_DATA\_YT\_Create()

**描述**

创建一个GRAPH\_DATA\_YT类型的数据对象

**原型**

GRAPH\_DATA\_Handle GRAPH\_DATA\_YT\_Create(GUI\_COLOR Color,unsigned MaxNumItems,I16 \* pData,unsigned NumItems);

|  |  |
| --- | --- |
| **参数** | **含义** |
| Color | 绘制数据的颜色 |
| MaxNumItems | 数据的最大数量 |
| pData | 指向添加到对象的数据的指针，该指针应该指向一个I16值的数组 |
| NumItems | 被添加的数据数量 |

**返回值**

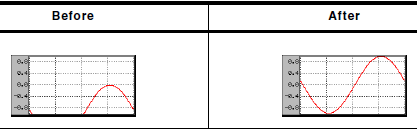
创建成功返回数据对象的句柄，否则返回0

**附加信息**

最后的数据显示在数据区域的最右列，如果一个数据对象包含比能够显示在数据区域更多的数据，可以调用GRAPH\_SetVSizeX()来显示一个滚动条，是的有可能在更大的数据对象滚动。

一旦关联到一个GRAPH控件，数据对象不需要被应用删除，当删除GRAPH控件的时候会被自动删除。

GRAPH\_DATA\_YT\_SetOffY()

****

**描述**

设置用来绘制对象数据的偏移量

**原型**

void GRAPH\_DATA\_YT\_SetOffY(GRAPH\_DATA\_Handle hDataObj, int Off);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hDataObj | 数据对象的句柄 |
| Off | 绘制数据的垂直偏移量 |

**附加信息**

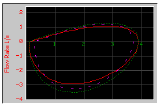
被显示的数据对象的垂直数据范围为（0）至（数据区的Y坐标大小-1），在使用滚动条的情况下当前的滚动位置被增加到范围之内。

**举例**

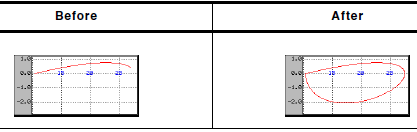
例如，可视数据范围应为-200到-100的数据需要向正方向转换200个点。

GRAPH\_DATA\_YT\_SetOffY(hDataObj, 200);

7.3 GRAPH\_DATA\_XY 相关函数



GRAPH\_DATA\_XY\_AddPoint()

****

**描述**

向一个GRAPH\_DATA\_XY对象增加一个新的数据

**原型**

void GRAPH\_DATA\_XY\_AddPoint(GRAPH\_DATA\_Handle hDataObj, GUI\_POINT \* pPoint);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hDataObj | 数据对象的句柄 |
| pPoint | 指向一个增加到数据对象的GUI\_PIONT结构体的指针 |

**附加信息**

给出的点被增加到数据对象，如果数据对象是满的，也就是说包含了创建时在参数MaxNumItems中定义的数量一样多的数据，这时在增加之前首先对数据进行一次转换，所以当向一个已满的对象增加新点的时候，原有的第一个点就被转换出去了。

GRAPH\_DATA\_XY\_Create()

**描述**

创建一个GRAPH\_DATA\_XY类型的数据对象

**原型**

GRAPH\_DATA\_Handle GRAPH\_DATA\_XY\_Create(GUI\_COLOR Color,unsigned MaxNumItems,GUI\_POINT \* pData,unsigned NumItems);

|  |  |
| --- | --- |
| **参数** | **含义** |
| Color | 用于绘制数据的颜色 |
| MaxNumItems | 最大的点数 |
| pData | 指向被增加到对象的数据指针，该指针应该指向GUI\_POINT类型的数组 |
| NumItems | 被增加的点的数量 |

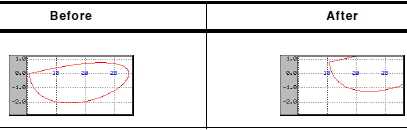
**返回值**

如果创建成功，返回数据对象的指针，否则返回0

**附加信息**

一旦关联到一个GRAPH控件，数据对象不需要被应用删除，在删除控件的时候会被自动删除。

GRAPH\_DATA\_XY\_SetOffX()，GRAPH\_DATA\_XY\_SetOffY()

****

**描述**

设置绘制曲线的垂直和水平偏移量

**原型**

void GRAPH\_DATA\_XY\_SetOffX(GRAPH\_DATA\_Handle hDataObj, int Off);

void GRAPH\_DATA\_XY\_SetOffY(GRAPH\_DATA\_Handle hDataObj, int Off);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hDataObj | 数据对象的句柄 |
| Off | 显示曲线的水平/垂直偏移量 |

**附加信息**

数据对象的数据范围为（0,0）—(数据对象的X轴范围-1，数据对象的Y轴范围-1)，当使用滚动条时，当前的滚动位置被增加到各自的范围内。若要使数据可视范围之外数据可见，应当设置偏移量，这样数据就位于可视范围之内了。

**举例**

**例如，可视数据范围为（100，-1200）—（200，-1100）应按如下设置偏移量**

GRAPH\_DATA\_XY\_SetOffX(hDataObj, -100);

GRAPH\_DATA\_XY\_SetOffY(hDataObj, 1200);

GRAPH\_DATA\_XY\_SetLineStyle()

**描述**

设置绘制曲线使用的线型

**原型**

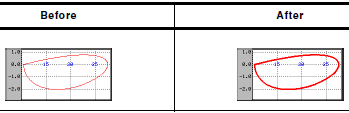
void GRAPH\_DATA\_XY\_SetLineStyle(GRAPH\_DATA\_Handle hDataObj, U8 LineStyle);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hDataObj | 数据对象的句柄 |
| LineStyle | 使用新的线型，所支持的线型请参考函数GUI\_SetLineStyle() |

**使用限制**

请注意，如果线宽大于1，只能使用GUI\_LS\_SOLID(缺省值)。

GRAPH\_DATA\_XY\_SetPenSize()

****

**描述**

设置绘制曲线使用的线宽

**原型**

void GRAPH\_DATA\_XY\_SetPenSize(GRAPH\_DATA\_Handle hDataObj, int PenSize);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hDataObj | 数据对象的句柄 |
| PenSize | 绘制曲线使用的线宽 |

7.4 比例尺相关函数

GRAPH控件支持用于标注的水平和垂直比例尺，使用比例尺的函数见下面介绍

GRAPH\_SCALE\_Create()

**描述**

创建一个比例尺对象

**原型**

GRAPH\_SCALE\_Handle GRAPH\_SCALE\_Create(int Pos, int Align,unsigned Flags, unsigned TickDist);

|  |  |
| --- | --- |
| **参数** | **含义** |
| Pos | 相对于GRAPH控件左侧和顶侧边缘的位置 |
| TextAlign | 绘制数字的文字对齐方式，详细信息请参考函数GUI\_SetTextAlign() |
| Flags | 见下表 |
| TickDist | 一个标注到另一个的距离 |

|  |  |
| --- | --- |
| **参数Flags的取值范围** | |
| GRAPH\_SCALE\_CF\_HORIZONTAL | 创建一个水平比例尺对象 |
| GRAPH\_SCALE\_CF\_VERTICAL | 创建一个垂直比例尺对象 |

**返回值**

如果创建成功，返回比例尺对象的句柄，否则返回0

**附加信息**

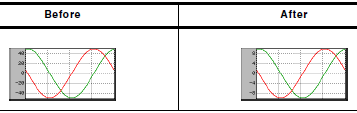
水平比例尺对象从数据区的底部到顶部显示，垂直比例尺对象从左边到右边显示，当第一个位置是0点时，参数TickDist标识了两个数字之间的距离。

参数Pos表示，使用水平比例尺时，控件顶部到比例尺文字之间的距离（像素），使用垂直比例尺时，这个参数表示从控件左侧边缘到水平文字位置之间的距离，请注意实际的文字位置还依赖于在TextAlign参数中定义的文字对齐方式。

比例尺对象在每一个位置上绘制一个数字，使用水平比例尺时有一个例外，如果第一个数字是0，则不会在对应的位置上绘制。

一旦附加到一个GRAPH上，比例尺对象不需要被应用删除，当删除GRAPH控件的时候，它会被自动删除。

GRAPH\_SCALE\_SetFactor()

****

**描述**

设置计算所绘制的数字的比例因子

**原型**

float GRAPH\_SCALE\_SetFactor(GRAPH\_SCALE\_Handle hScaleObj, float Factor);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hScaleObj | 比例尺对象的句柄 |
| Factor | 比例因子 |

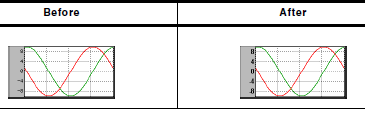
**返回值**

返回之前使用的比例因子

**附加信息**

如果不使用比例因子，比例尺对象的单位是像素，因此设置的比例因子应当转换像素到所需的单位。

GRAPH\_SCALE\_SetFont()

****

**描述**

设置绘制比例尺所使用的字体

**原型**

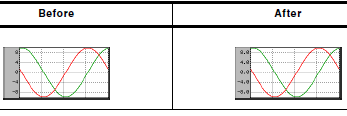
const GUI\_FONT \* GRAPH\_SCALE\_SetFont(GRAPH\_SCALE\_Handle hScaleObj,const GUI\_FONT \* pFont);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hScaleObj | 比例尺对象的句柄 |
| pFont | 使用的字体 |

**返回值**

返回之前使用的字体。

GRAPH\_SCALE\_SetNumDecs()

****

**描述**

设置显示的小数点后的位数

**原型**

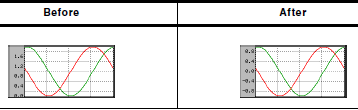
int GRAPH\_SCALE\_SetNumDecs(GRAPH\_SCALE\_Handle hScaleObj, int NumDecs);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hScaleObj | 比例尺对象的句柄 |
| NumDecs | 小数点后的位数 |

**返回值**

返回之前设置的小数点后位数。

GRAPH\_SCALE\_SetOff()

****

**描述**

设置比例尺对象向正或负方向进行偏移的偏移量

**原型**

int GRAPH\_SCALE\_SetOff(GRAPH\_SCALE\_Handle hScaleObj, int Off);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hScaleObj | 比例尺对象的句柄 |
| Off | 绘制比例尺的偏移量 |

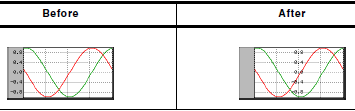
**返回值**

返回之前使用的偏移量。

**附加信息**

如同在函数GRAPH\_SCALE\_Create()函数中的描述，一个水平比例尺于是从数据区的底部边缘到顶部，一个垂直比例尺是从左侧边缘到右侧，当第一个位置是0点，很多情况下是不可取的，如果比例尺应当向正方向转换，则使用正的偏移量，同样，若想负方向转换，则使用负的偏移量值。

GRAPH\_SCALE\_SetPos

****

**描述**

设置在GRAPH控件里显示比例尺对象的位置

**原型**

int GRAPH\_SCALE\_SetPos(GRAPH\_SCALE\_Handle hScaleObj, int Pos);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hScaleObj | 比例尺对象的句柄 |
| Pos | 比例尺显示的位置 |

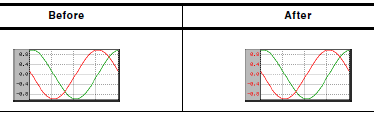
**返回值**

返回之前的比例尺对象的位置

**附加信息**

参数Pos表示在使用水平比例尺的情况下，从控件顶部到比例尺文字之间的距离（像素值），使用垂直比例尺时，该参数表示从控件左侧边缘到水平文字位置之间的距离（像素），请注意，实际的文字位置还依赖于比例尺对象的位子对齐方式设置。

GRAPH\_SCALE\_SetTextColor

****

**描述**

设置显示数字的文字颜色

**原型**

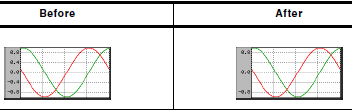
GUI\_COLOR GRAPH\_SCALE\_SetTextColor(GRAPH\_SCALE\_Handle hScaleObj,GUI\_COLOR Color);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hScaleObj | 比例尺对象的句柄 |
| Color | 显示数字的颜色值 |

**返回值**

返回之前使用的颜色。

GRAPH\_SCALE\_SetTickDist

****

**描述**

设置一个数字到另一个数字之间的距离

**原型**

unsigned GRAPH\_SCALE\_SetTickDist(GRAPH\_SCALE\_Handle hScaleObj,unsigned Dist);

|  |  |
| --- | --- |
| **参数** | **含义** |
| hScaleObj | 比例尺对象的句柄 |
| Dist | 两个数字之间的距离值（像素） |

**返回值**

返回之前的距离。