ProviewR

OPEN SOURCE PROCESS CONTROL

Ge Designer's Guide

2024-02-25
Version 6.1.5

1

2 Introduction

Ge is an editor to build operator graphics in ProviewR. The editor is integrated in the ProviewR development environment and started from the Configurator. The graphs are built of base objects in shapes of rectangles, circles, lines and texts, and of subgraphs, i.e. a kind of composed object. Ge contains a collection of subgraphs for some common components: valves, motors, pumps etc. Other components, the constructor can easily draw himself. The shapes are stored as vectors, which makes it possible to scale and rotate the objects. In runtime, a graph is opened from the Xtt operator environment or from the web operator environment.

3 Glossary

Graph

A window from where the operator can supervise and control the process.

Subgraf

A collection of base objects can be stored as a subgraph. A subgraph works as a kind of class or pattern, and in a graph you can create instances of the subgraph, that is subgraph objects. Stored subgraphs are displayed in the subgraph palette.

Base objects

Base object are objects of type rectangles, ellipses, lines, polylines, texts, connection points and annotations.

Complex objects.

More complex objects, e.g. bar, trend, axis.

Annotation

Text in a subgraph that can be unique for each instance.

Connection

A line that connects two subgraphs. The line can be straight or contain breakpoints that divide it in vertical and horizontal parts.

Connection point

A point in a subgraph to which a connection can be connected.

Signal

Signal is used as a common name for an attribute of an object in the ProviewR realtime database. Often its the value for a Di, Do, Dv, Ai, Ao or Av, but it can be an attribute in an arbitrary object.

MB1

The left mouse button.

MB2

The middle mouse button or the scroll wheel.

MB3

The right mouse button.

Rtdb

The ProviewR realtime database.

4 Work with the editor

4.1 Base objects

We will start with looking at simple objects as lines, rectangles, circles and polylines. These objects are called base objects. The base objects is found to the left in the bottom row of the tool panel.

Fig Buttons to create base objects

4.1.1 Rectangle

You draw a rectangle by activating the rectangle button in the tool panel, push the right mouse button where a corner of the rectangle should be, drag with the mouse button pressed to the opposite corner and release the mouse button. With the default settings, a rectangle with black border is now created.

Fig Rectangle

We shall now have a look at how the appearance of the rectangle can be changed. In the tool panel there are buttons for border, fill and 3D. They work in the way that if an object is selected, the property of the selected object is changed. At the same time the current setting is the default value for new objects that are created.

Object editor

By doubleclicking on the rectangle, a window is opened where you can set various attributes of the rectangle. An attribute is changed by selecting the attributes, pressing key arrow right (or Ctrl+Q) and inserting the new value.

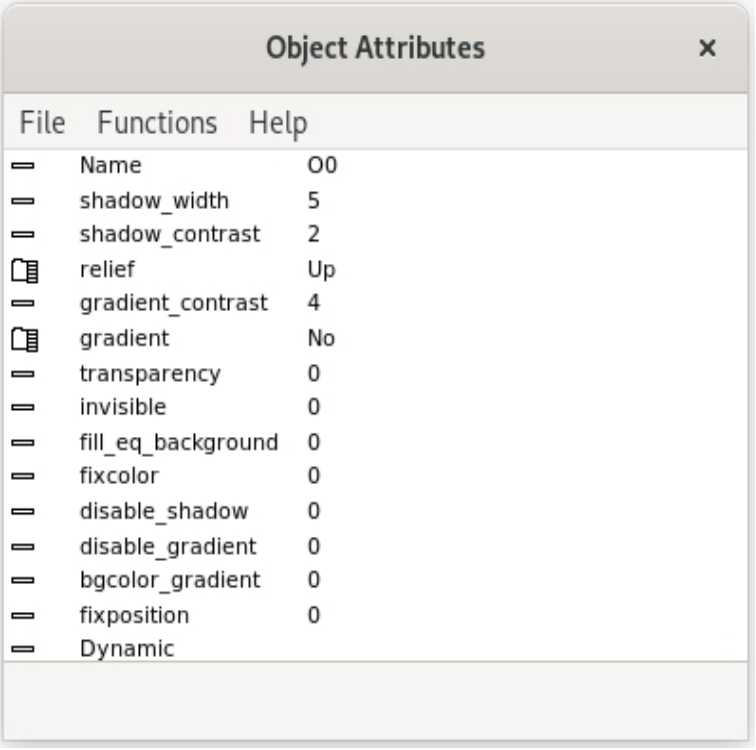


Fig Object editor

Fill color

We can fill the rectangle with color by clicking on the Fill button in the tool panel.

The color is changed by first selecting the rectangle by clicking on it, and the click on the desired color in the color palette.

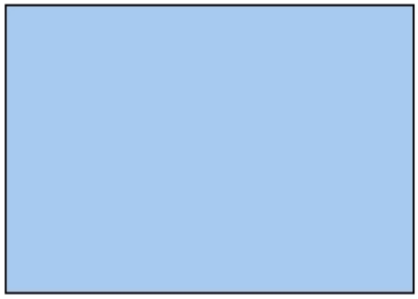


Fig Rectangle with fill color

Border

The border of the rectangle can be removed with the Border button in the tool panel. To remove the border the rectangle has to be either filled or have a 3D frame. If the border is removed, it can be added with the border button.

Fig Rectangle without border

Border width

The width of the border can be changed with the LineWidth menu in the tool panel. Select the rectangle and set desired line width from the tool panel. The unit of the line width is pixel, and a width between 1 and 8 can be set.

Fig Rectangle with broader border

Border color

Also the color of the border can be modified. Select the rectangle, activate 'Border' in the color palette and click on the desired color.

Fig Modified border color

3D

With the 3D button in the tool panel, a frame with relief effect is drawn on the rectangle.

Fig Rectangle with 3D

The width of the relief can be changed with the attribute shadow_width. Open the object editor by double clicking on the rectangle, and insert a new value in shadow_width. The unit of shadow_width is in percentage of the width or height of the rectangle (the smallest).

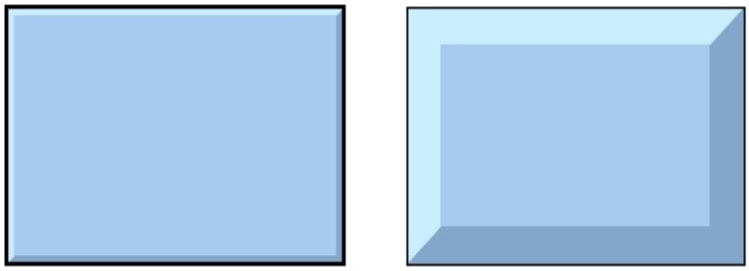


Fig Width 3 % to the left and 15 % to the right

You can also change the contrast of the relief frame with shadow_contrast. By increasing the value to 3 the shade will be darker and the light border will be lighter. Valid values are 1 - 3.

Whether the relief is outwards or inwards depends on the attribute relief. With Up the relief is outwards, ie the upper border is light and the lower dark. With Down the upper border is dark and the lower is light, which gives the effect of an inward relief. You will get the greatest relief effect if the relief is drawn with the same color as the background.

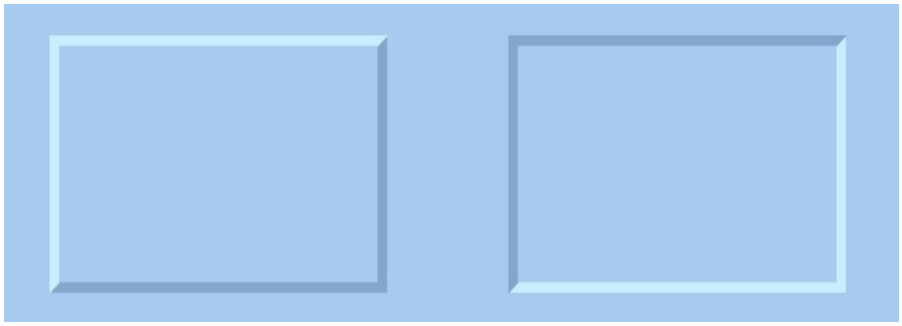


Fig Relief Up to the left and Down to the right

Gradient

With the three gradient buttons you can set a color gradient on the rectangle from lighter to darker color tone.

Fig Rectangles with various gradients

In the gradient menu there are 17 additional variations of gradients to select. Below Horizontal Tube2 is shown that is suitable for cylindric items.

Fig Rectangle with gradient Horizontal Tube2

The contrast of the gradient can be affected by the gradient_contrast attribute in the object editor. The contrast can have a value between 0 and 10.

Gradients with two different colors can be created by using the object background color.
- Draw the rectangle with a fill color.
- Attach a gradient to the rectangle.
- Open the object editor and set bgcolor_gradient to 1.
- Set the background color by selecting the rectangle and click with Shift+Ctrl MB1 on a color in the color palette.

Now the gradient will be drawn from the fill color to the background color.

Fig Gradient between two colors

Set transparency

Transparency of an object is set from the object editor. Transparency is a value between 0 and 1, where 0 is no transparency, and 1 is full transparency.

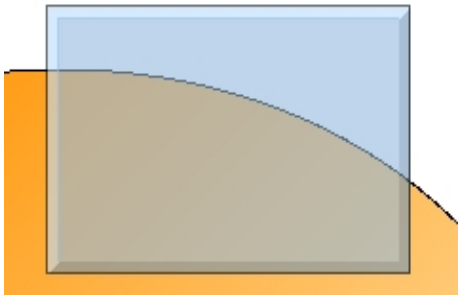


Fig Rectangle with transparency set to 0.5

Draw a square

To draw a square with equal width and height you press Ctrl+E (Scale equal) before you draw the rectangle. Reset Equal scale by right clicking in the work area.

Draw several rectangles

If you are going to draw several rectangles you can press the Shift key when clicking on the rectangle button in the tool panel. Now you can draw several rectangles without clicking on

the rectangle button for each new rectangle. The function is reset by right clicking in the work area.

4.1.2 Line

You draw a line by activating the line button in the tool panel, press the right mouse button where the line is to start, drag with the mouse button pressed to the end position and there release the mouse button.

Fig Line

Line width

The line width is changed by selecting the line and choose a width in the Linewidth menu.

Fig Linewidth set to 8

Color

The color is changed by selecting the line, clicking on Border in the color palette, and clicking on the desired color.

Fig Colored line

Line type

There is a number of different line types to choose between. Select the line and activate the desired type in the Linetype menu.

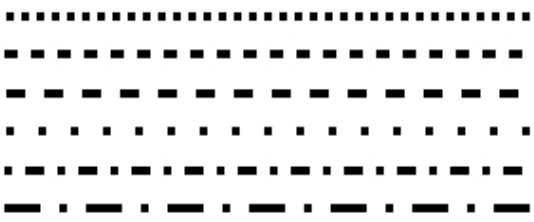


Fig Various line types

Horizontal and vertical lines

To draw a horizontal line you press Ctrl+H (MoveRestrictions Horizontal) before you draw the line. A vertical line is drawn by first pressing Ctrl+G (MoveRestrictions Vertical). MoveRestrictions are reset by right clicking in the work area.

Draw several lines

If you will draw several lines you can press the Shift key when clicking on the line button in the tool panel. Now you can draw several lines without clicking on the line button for each new line. The function is reset by right clicking in the work area.

4.1.3 Ellipse and circle

Draw an ellipse by selecting the ellipse button in the tool panel, and press the left mouse button, drag with the button pressed, and finally release the button.

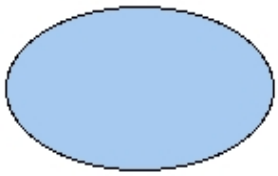


Fig Ellipse

As for the rectangle, you can set properties for border, fill, border color, fill color, 3D and gradient. Some examples are displayed below.

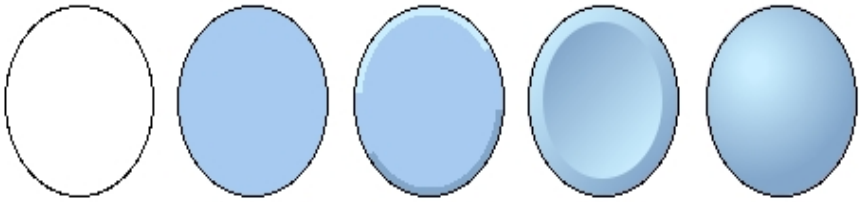


Fig Circle with various combinations of border, fill, 3D and gradient

Arc

You can also draw an arc by specifying angle1 and angle2 in the object editor. angle1 is the angle from the horizontal axis to the beginning of the arc, and angle2 is the angle for the extension of the arc.

Draw several ellipses

By pressing the Shift key when you click on the arc button in the tool panel, you can draw several arcs ellipses without clicking on the arc button for every new ellipse. The function is reset by right clickning in the work area.

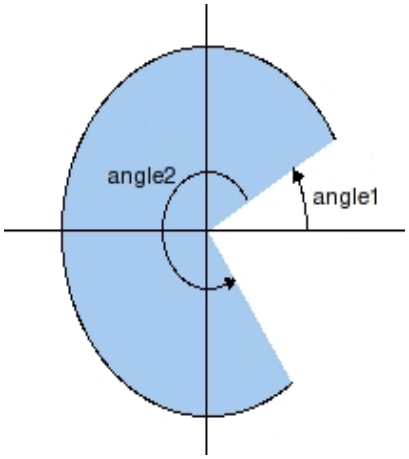


Fig Arc with angle1=30 and angle2=270

4.1.4 Polylinje and polygon

A polyline is drawn by clicking on the polyline button in the tool panel, press left mouse button at the starting point, keep the mouse button pressed and drag to the next breakpoint and release the mouse button, press again and drag to the next breakpoint etc. When the last line is drawn, right click in the work area to finish the polyline.

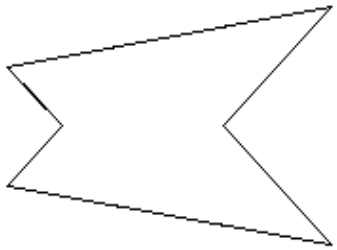


Fig Polyline

Various combinations of border, fill, 3D, gradient, linewitdh and color can be used for polylines.

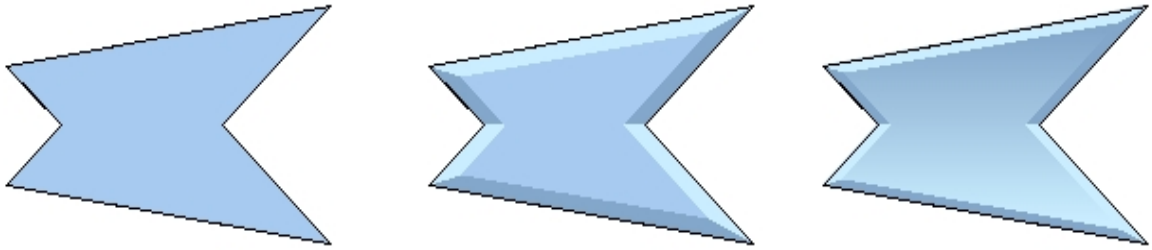


Fig Some combinations of fill, 3D and gradient

3D

When applying the 3D effect on polylines, the polyline should be closed, ie the start point should be the same as the endpoint. To draw a closed polyline, activate the Snap to grid in the tool panel before drawing the polyline.

Modify a polyline

To move a breakpoint in a polyline, select the polyline and activate Edit/Edit Polyline in the menu. The breakpoints and endpoints are now sensitive and can be moved by dragging with the left mouse button.

Vertical and horizontal lines

Often you only want the polyline to consist of vertical and horizontal lines. By pressing Ctrl+H (MoveRestrictions horizontal) the first line will be horizontal, the second vertical etc. If you press Ctrl+G (MoveRestrictions vertical) the first line will be vertical and the second horizontal etc. Reset MoveRestrictions by right clicking in the work area.

4.1.5 Text

Texts are drawn by activating the text button in the tool panel. Click in the work area and insert the text into the input field at the bottom line in the editor.

Pump control

Fig Text

Object attributes

By doubleclicking on the text the object editor is opened, displaying the object attributes. The attribute Adjustment is set for left, center or right adjustment of the text.

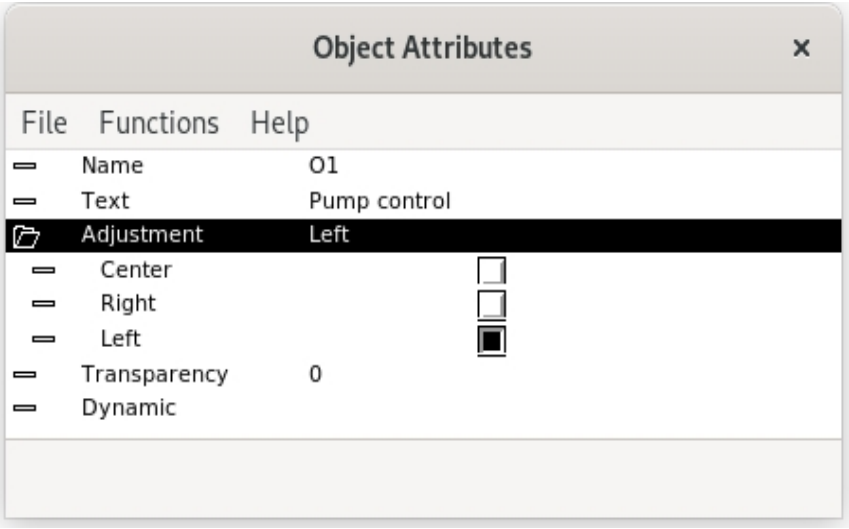


Fig The object editor with adjustment attribute

Text size

You change the text size from the Textsize menu in the tool panel. Select the text and set desired size in the menu. You can also use the scale function to change the size.

Font

The font is selected with the font menu in the tool panel, You can choose between the fonts Helvetica, Times, New Century Schoolbook, Courier and Lucida Sans. Select the text and specify the wanted font in the menu.

Color

Text color is selected by activating the Text button in the color palette, and the clicking on a clolor.

Pump control

Fig Colored text

Text rotation

A text can be rotated in steps of 90 degrees by using the rotation button in the tool panel. Only text with center adjustment can be rotated, ie the attribute Adjustment has to be set to Center in the object editor.

Fig Rotated text

Change a text

To change a text you select the text and press Ctrl+T (Change Text). An input field is opened in the bottom line of the editor where you can insert the new text.

Bitmap fonts

The normal fonts are scalable and antialiased. In some cases you might prefer bitmap fonts. They only exist in certain sizes between 8 and 24 pixel, but requires considerably less CPU capacity to handle. Bitmap fonts are selected by setting BitmapFonts in Graph Attributes to 1.

Window manager font settings

Some window managers have settings that affects the fonts. In Gnome you find the setting in System menu where you can choose between various rendering.

4.2 Editing

In this section we will have a closer look at various editing functions, for example how to move, scale and color objects.

Create an object

How to create a base object is described in the Base object section above. Normally you create and object by clicking on the symbol in the tool panel, and then click or drag with the left mouse button in the work area. Subgraphs are created by selecting a subgraph in the subgraph palette and click with MB2 in the work area.

Select an object

You select an object with Click MB1. A selected object is marked with red color. If you want to select several object click on them with Shift/Click MB1.

To select objects in the certain area you can also drag with MB1 in the work area The objects inside the selection rectangle will be selected. As objects are also moved by drag MB1 you should avoid to hit an object when starting the drag. If this is hard to avoid you can use Shift/Drag MB1 that adds objects to the select list.

Note that when you change the color of a selected object the red marking is removed, as you then want to se the new color instead. The object is though still selected.

By right clicking in the work area the select list is emptied.

Delete an object

To delete an object you select the object and press the Delete key. You can also delete an object by double clicking with MB2 on the object. With doubleclick MB2 in some empty space in the work area you delete all selected objects.

Move an object

An object is moved with drag MB1. If several object should be moved concurrently, you select them and drag one of the objects. All selected object will then follow. If the movement should be vertical or horizontal, you can use Functions/MoveRestrictions/Vertical (Ctrl+G) or Functions/MoveRestrictions/Horizontal (Ctrl+H). These functions are active until you reset with right click.

If Snap to grid is activated in the tool panel, the position of an object is adjusted to nearest grid point when the object is moved.

You can also move an object with the move command. This will give a more precise movement than if you move with the mouse. You can move by supplying relative or absolute coordinates. The command to move the selected object to the point (1,1) is

```
ge> move selected /absx=1 /absy=1
```

Object with fix position

Large objects in the background are easy to move by mistake when you edit. For some objects as rectangles, ellipses and polylines you can set the attribute 'fixposition' that locks the object at a certain position.

Copy an object

You copy objects by selecting them and activating Edit/Copy (Ctrl+C) and Edit/Paste (Ctrl+V) in the menu. After paste the objects are following the cursor, and by clicking with MB1 you position them in the work area. When pasting you can select vertical (Ctrl+G) or horizontal (Ctrl+H) movement.

Scale an object

An object is scaled by selecting the object, activating Scale in the tool panel, and then stretch the rectangle that surrounds the objects. Terminate with click MB3. If you want the proportions between width and hight to be preserved, press first Ctrl+E (Scale Equal).

There is also a button in the tool panel to double the size of an object, and one button to half the size.

Rotate an object

An object can be rotated with the rotation button in the tool panel. Selected objects will

be rotated in steps of 90 degrees.

I you want another rotation angle you activate Edit/Rotate in in the menu. Note that rectangles and ellipses only can be rotated in steps of 90 degrees, and to rotate texts the Adjustment attribute for the text has to be set to Center.

Mirror

There are two buttons in the tool panel to mirror, one for horizontal mirroring an one for vertical mirroring. The mirroring it executed on the selected objects.

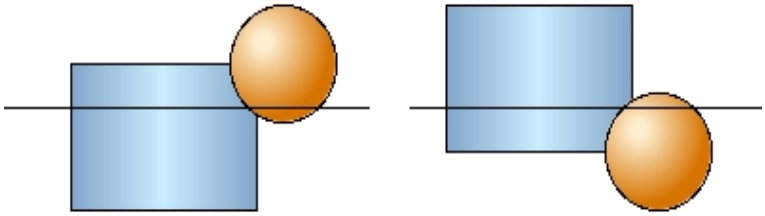


Fig Horizontal mirroring

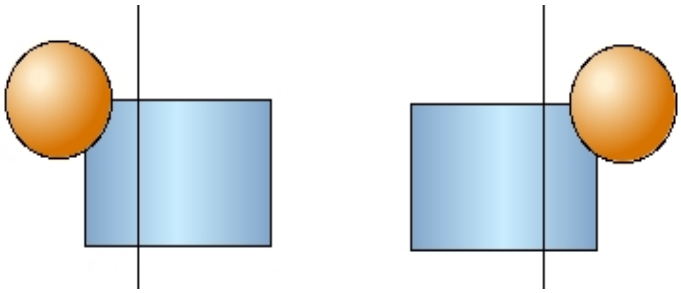


Fig Vertical mirroring

Change object color

There are three different types of colors that can be selected in the color palette, fill color, border color and text color. In the palette you find the buttons Fill, Border and Text where you mark which color type should be selected. When you then select a color in the palette the color of the selected objects are changed. The selected color combination is also applied to new objects that are created.

Fill color
To set the fill color of an object, you select the object, check that Fill is active in the color palette, or else you click on Fill. Then you select the color you want to set on the object. In the example below the fill color is changed from blue to green.

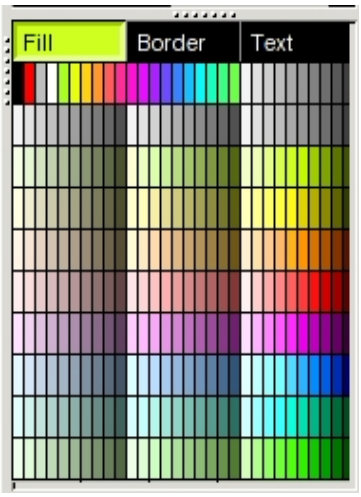


Fig Fill is marked and a green tone is selected.

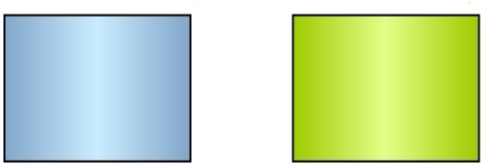


Fig Fill color is change from blue to green

Border color

The border color is changed in a similar way. Now you mark Border in the color palette, and then select the border color that should be set to the selected objects.

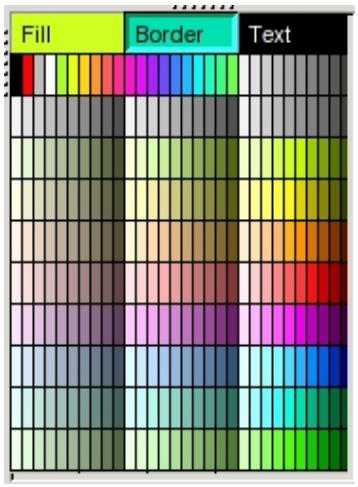


Fig Border is marked and a bluegreen tone is selected for border color

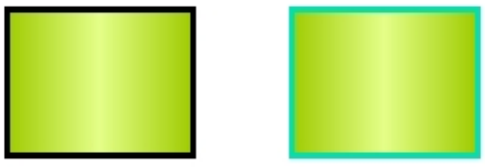


Fig Border color changed from black to bluegreen

Text color

To change the text color, you activate Text in the color palette, and then choose the text color that should be set to the selected objects.

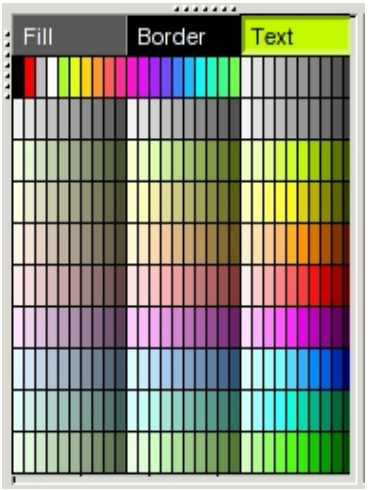


Fig Text is marked and a green tone is selected

Fig The text color is changed from black to green

Background color

You set the background color the the graph by selecting a fill color in the color palette and activate Functions/Set Background Color in the menu.

Grid

The two buttons Show grid and Snap to grid, together with the Gridsize menu handles the grid function. Show grid shows all grid points, and with the Gridsize menu you can set the distance between the grid points. When Snap to grid is activated, the coordinates for objects that are created or moved will be adjusted to nearest grid point.

The gridsize can be set to 1.0, 0.5, 0.25 or 0.10 in the Gridsize menu. With the 'set gridsize' command it is also possible to set other values on the grid size. To set the grid size to 0.20 enter the command

```
ge> set gridsize 0.20
```

Order objects

There are a number of buttons in the tool panel to adjust the position of number of objects to place them on the same horizontal or vertical level.

You can also adjust the position so objects will be placed on equal distance from each other.

Groups

You create a group of a number of objects by selecting the objects and pressing the Group button in the tool panel. The group is then handled as if it where an individual object, for example when you select, move or scale it. If you want to make a change in a member object, the group has to be dissolved by the Ungroup button, then you can make the change and group again.

Groups have the properties dynamic and action, ie you can connect them to signals in the database and for example change color of the group dependent on the status of a signal. You can also make them sensitive for click, and set signal values when they are clicked on. These are properties they share with subgraphs, that the base objects lacks. If you want to set dynamic on a base object, eg a rectangle, you have to create a group with only the rectangle, and then you will get the possibility to set dynamic and action on the rectangle. In this way you can create an indicator, pushbutton or a bar from the rectangle.

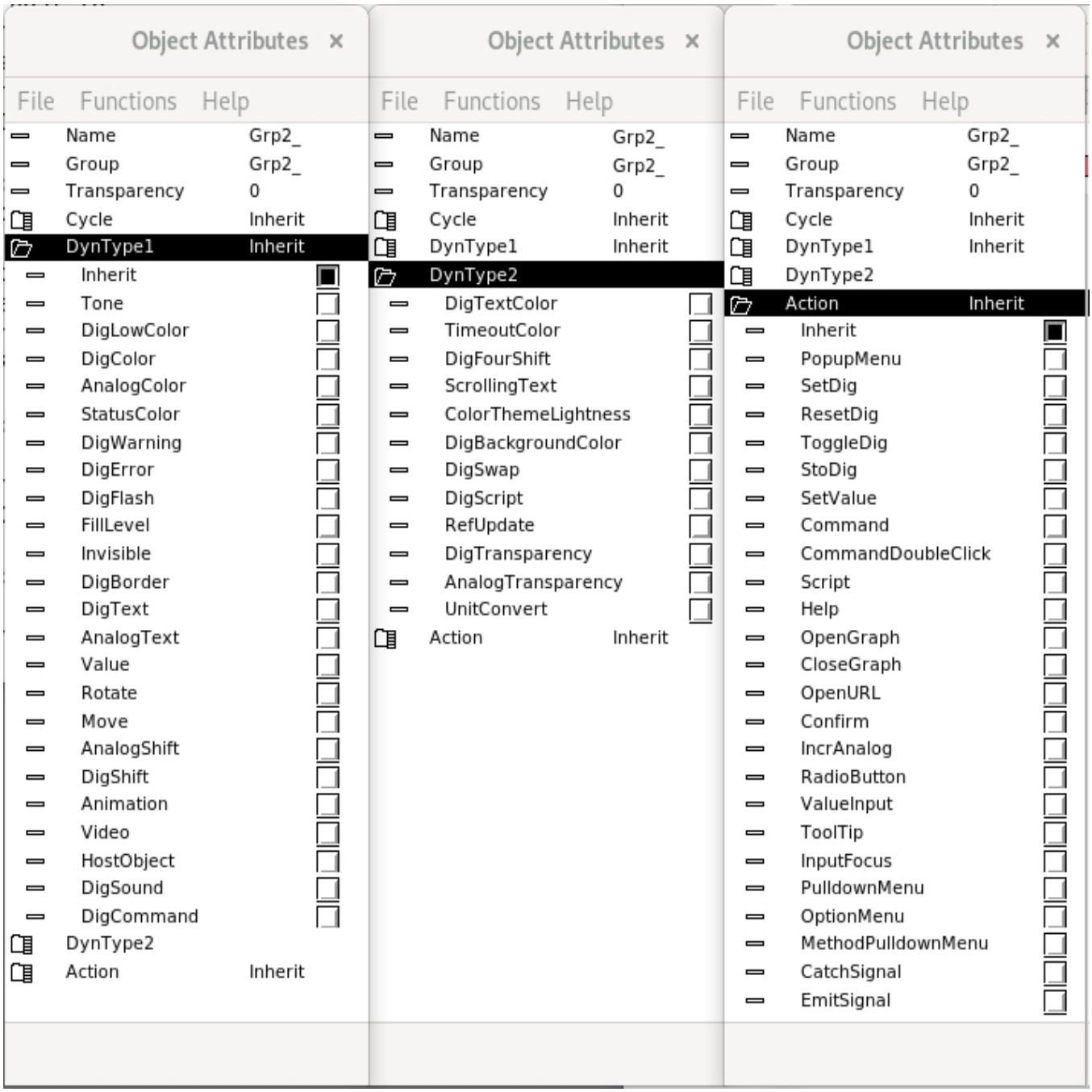


Fig The object editor for a group with attributes for dynamic and action

4.3 Subgraphs

A subgraph is a graphic component built by base objects. It often symbolizes a certain element in the plant, eg a valve or a pump, but it can also be a pushbutton or an input field.

Create a subgraph

To the right in the editor, the subgraph palette is found, and you create a subgraph by

selecting a subgraph in the palette and click with MB2 in the work area. In the figure below a Valve under the Process map is selected.

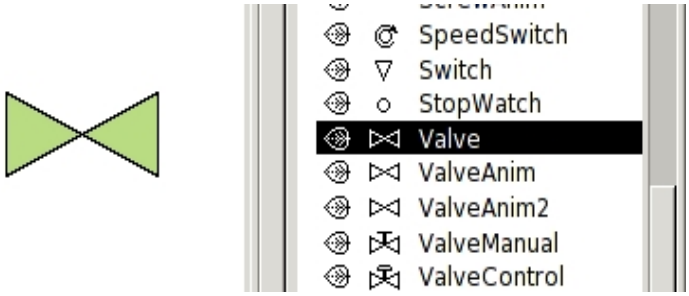


Fig A valve subgraph

On most subgraphs you can apply 3D and different kinds of gradients. There are though some subgraphs that are not adapter for this.

Fig Valve with 3D to the left, and 3D + gradient to the right

Change color

The valve in the figure is a monochrome subgraph, and you can change the color by selecting a fill color in the color palette in the same way as for base objects. You can also modify the border color and border width.

Fig Blue colored valve

Polychrome subgraphs

On subgraphs that contain different color, you change the color with the functions for color shift or with the color tone palette. Let us create a burner (Process/Burner) that contains the colors yellow and orange.

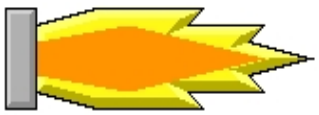


Fig Polychrome subgraph

If we set a fill color we will lose the color drawing and the burner will be monochrome. Instead we use the buttons for color shift in the tool panel (the right buttons in the figure below). Select the subgraph and click on the arrows to shift the color.

Fig Buttons to shift color

Now the colors are rotated on the color wheel so that the subgraphs is still drawn with two colors, and the color contrast is kept. You can also use the buttons for changing intensity and lightness, positioned the right of the color shift buttons, to adjust the subgraph color.

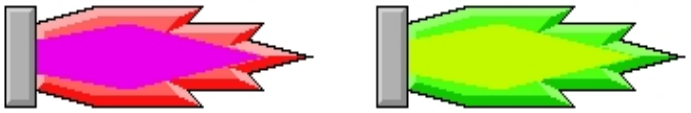


Fig Example of burners with shifted colors

Another way of changing the color of a polychrome subgraph is to use the color tone palette (the lower part of the color palette in the figure below). Select the subgraph and click on a color tone in the palette.

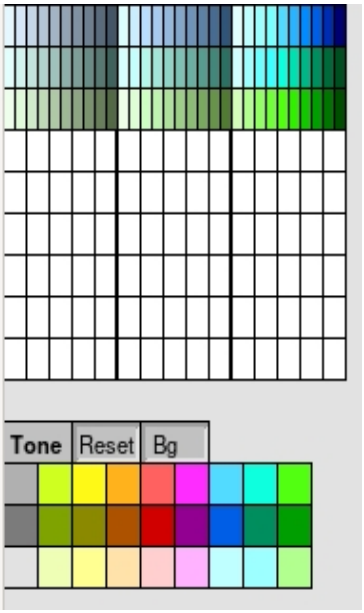


Fig Färgtons-paletten

The result is that the different parts have the same color tone, but the lightness contrast between the different parts is kept.

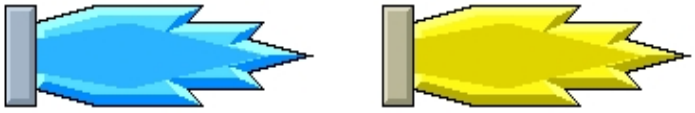


Fig Burners with various color tones applied

Connection points

Some subgraphs contains connection points that makes it possible to draw connections between the subgraphs. You create a connection by dragging MB2 from a connection point in a subgraph, and release MB2 at a connection point in another subgraph. For the connections, line width, color and 3D can be set. If you release a connection in the work area, a junction is created, from which connections in different directions can be drawn.

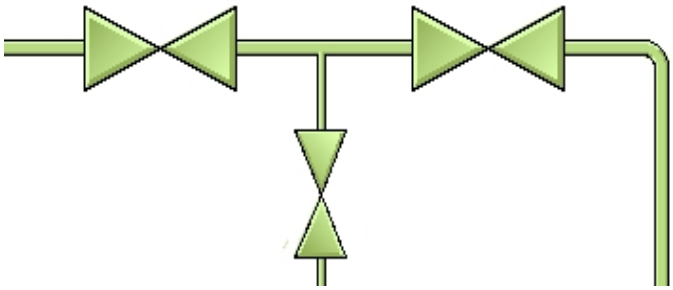


Fig Subgraphs with connections

Dynamic and action

Subgraphs have, like groups, the properties dynamic and action, which makes is possible to connect them to signals in the database and change color and shape dependent on the status of a signal. It is also possible to make them sensitive for mouse clicks and set signals when they are clicked on. Furthermore, subgraphs have often a preprogrammed dynamic or action. An indicator has for example the dynamic to change color as default, so you only have to connect it to a signal to make it work.

Let us create an indicator and look at how to change the color. We mark an indicator Indicator/IndRoundMetalFrame in the subgraph palette and click with MB2 in the work area to create the indicator. To connect it to a signal in the database we click on the navigator button in the tool panel. Now the plant hierarchy is displayed to the left. We want the indicator to show status of the Dv H1-Dv3, and find the Dv in the plant hierarchy. By selecting H1-Dv3 and click with Ctrl+Doubleclick MB1 on the indicator the connection is made.

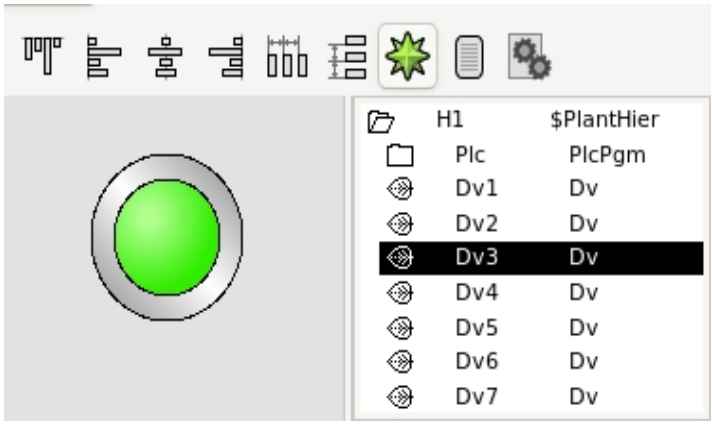


Fig A signal is connected to the indicator with Ctrl+Dubbleclick MB1

By opening the object editor for the indicator we can check that the connections is done.

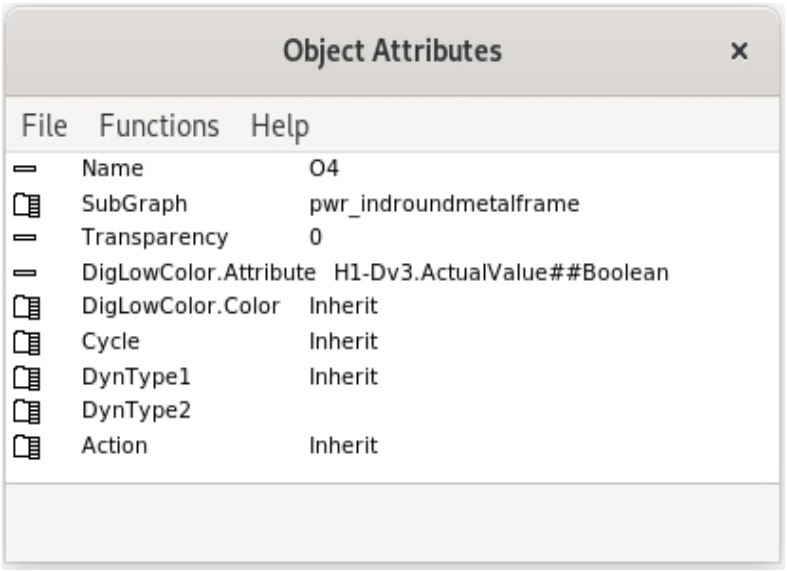


Fig The indicator object editor

We can see that the indicator has the default dynamic DigLowColor, that shifts between two colors, and DigLowColor.Attribute is set to H1-Dv3.ActualValue.

When we open the graph in rt_xtt we can se how the indicator shifts between green and dark gray when Dv3 shifts between 1 and 0.

Fig The indicator when the Dv is 1 to the left and 0 to the right

If we take a look at the valve above, it has a slightly more advanced dynamic. It can shift between three different colors, and thus has to be connected to two signals. One signal colors the valve red to indicate a fault, and the other colors the valve white to indicate closed valve.

If we open the object editor for the valve we can see that DigError.Attribute should be connected to a signal indicating fault, and DigLowColor.Attribute to a signal indicating that the valve is open.

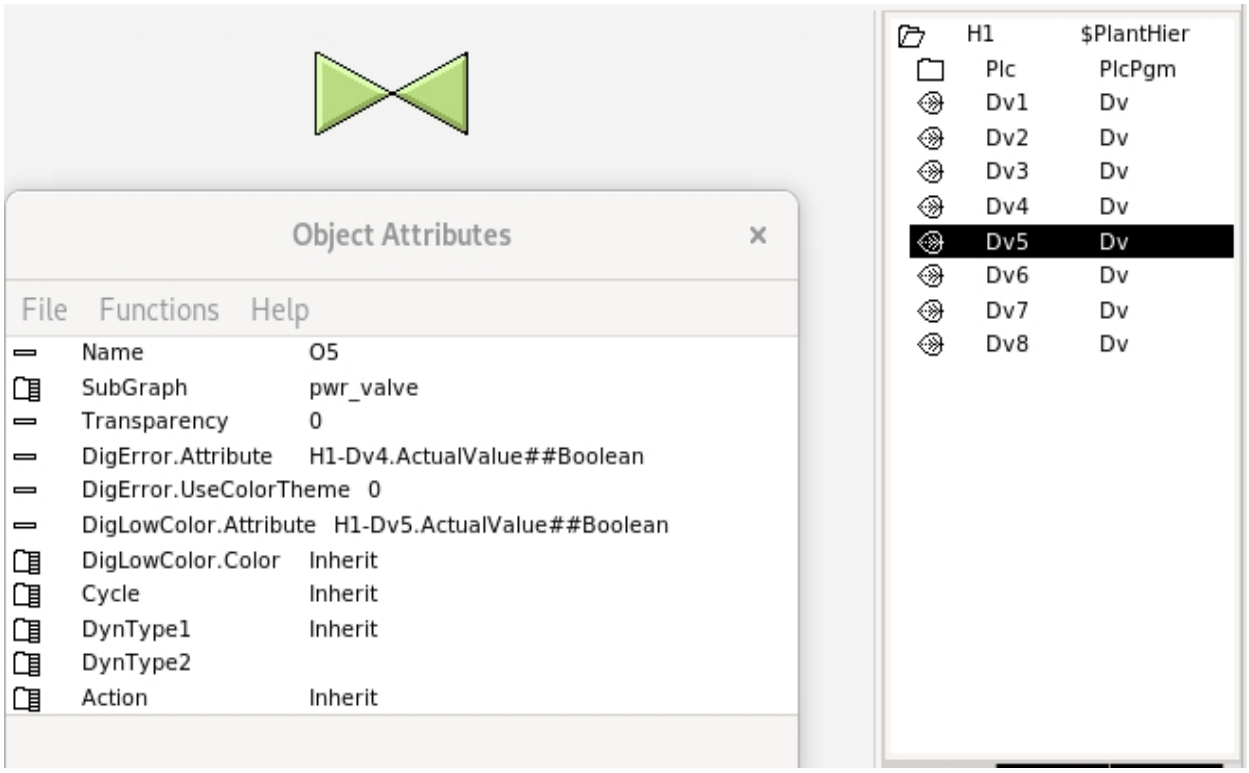


Fig Valve connected to two signals

In this case we will make the connection in the object editor, and select H1-Dv4 in the plant hierarchy. By clicking with Ctrl+Doubleclick MB1 on the attribute DigError.Attribute, H1-Dv4 in inserted. In the same way, H1-Dv5 is inserted into DigLowColor.Attribute. We can see the result in the figure below. When Dv4 is 0, ie the valve is closed, it is colored white. When Dv4 is high, ie the valve is opened, it is colored green. If we set the fault signal Dv4 it is colored red independent of the value of Dv5. This is due to the fact that DigError has higher priority than DigLowColor. In the object editor, the dynamics are ordered by priority, with the highest priority on top and the lowest priority at the bottom.

Fig Closed valve to the left, open valve in the middle and fault signal to the right

If we want to mark closed valve with dark gray instead of white, we change the color in DigLowColor.LowColor. The default value is Inherit, ie a white preprogrammed color will be chosen. By choosing GrayHigh9 instead, the color when the signal is low will be dark gray. If you have difficulties with identifying the colors, you can select desired color as fill color in the color palette, and click with Ctrl+Doublelclick MB1 on DigLowColor.LowColor to insert the color.

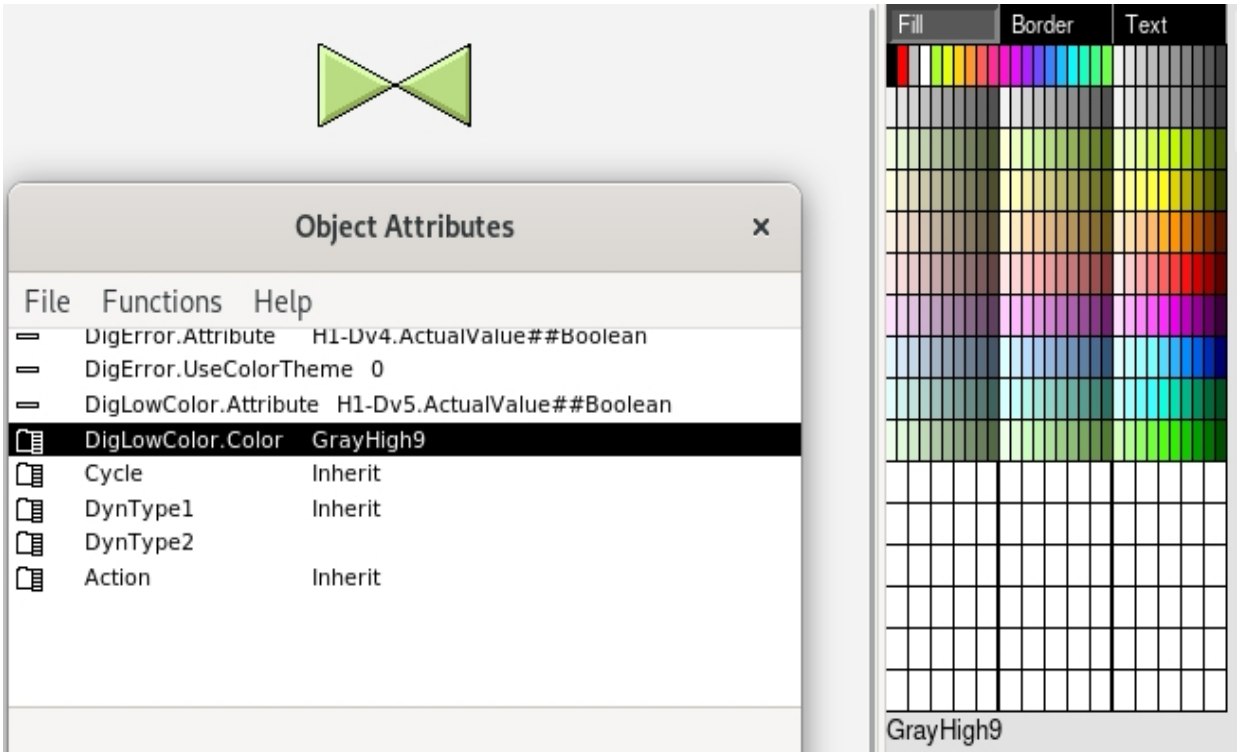


Fig LowColor is changed from Inherit to dark gray

Fig Closed valve to the left, open valve in the middle and fault signal to the right

4.4 Special objects

Besides base objects and subgraphs there are a collection of more complex objects. These objects are found in the subgraph palette. The objects are

Window
TabbedWindow
Slider
Trend
Bar
BarArc
Pie
BarChart
XYCurve
FastCurve
Axis
Table
PulldownMenu
OptionMenu
MethodToolbar

4.4.1 Window

The window object is a frame that displays a graph in a specified area in another graph. The graph is displayed with or without scrollbars. If is also possible to shift the graph displayed in the window by executing a command from a push button. You can also display object graphs for various objects in the database.

Here are some fields of application for the window object

- you have a table like part of the graph the takes to much space. This can be put in a window object and by utilizing the scrollbar of the window object you will gain space.
- you want to show information about a database object that already has an object graph.
- it is possible to build a graph with a window object that displays different graph dependent on a set of pushbuttons or menu alternatives. See also TabbedWindow.

Display a graph in a window object

The window object is found under the Other map in the subgraph palette. Create the object by middle click in the work area and scale the object to desired size.

Fig Window object

Open the object editor and supply the name of the graph that is to be displayed in Window.Filename. Note that the graph is read from $pwrp_exe, and as graphs in window objects often doesn't have an XttGraph object, they will not be copied automatically from $pwrp_pop to $pwrp_exe. The copying then has to be done by hand or by a make file.

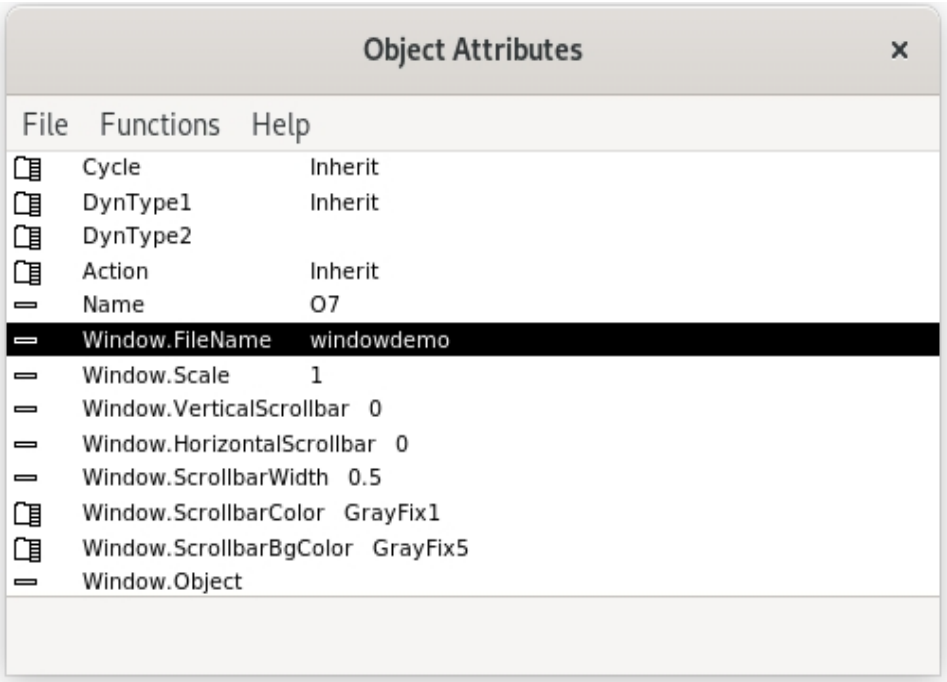


Fig The object editor for a window object with supplied filename

When the filename is inserted and the file exist on $pwrp_exe the graph will be displayed in the window object.

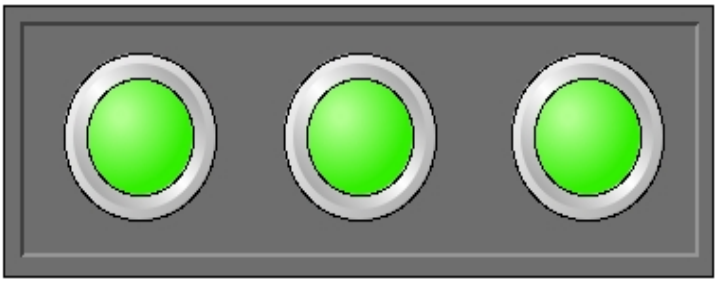


Fig Window object with inserted filename

Display an object graph in a window object

Several database objects have an object graph that is opened from 'Object Graph' in the popup menu for the object. Also object graphs can be viewed in window objects by specifying the name of the graph file. Also the database object that the graph should be connected to has to be specified in Window.Object.

If the object class is a part of the ProviewR release, the filename for the object graph is $pwr_exe/pwr_c_'classname'.pwg. For an Av object this will be $pwr_exe/pwr_c_av.pwg. You have to specify the path in the filename as the file is located in the default directory $pwrp_exe.

If the class reside in a class volume in the project, the filename is $pwrp_exe/'classname'.pwg. In this case there is no need to specify the path.

In the example below the object graph for the Av object H1-Av1 is displayed. The graph is scaled to 3/4 of the original size by setting Window.Scale to 0.75.

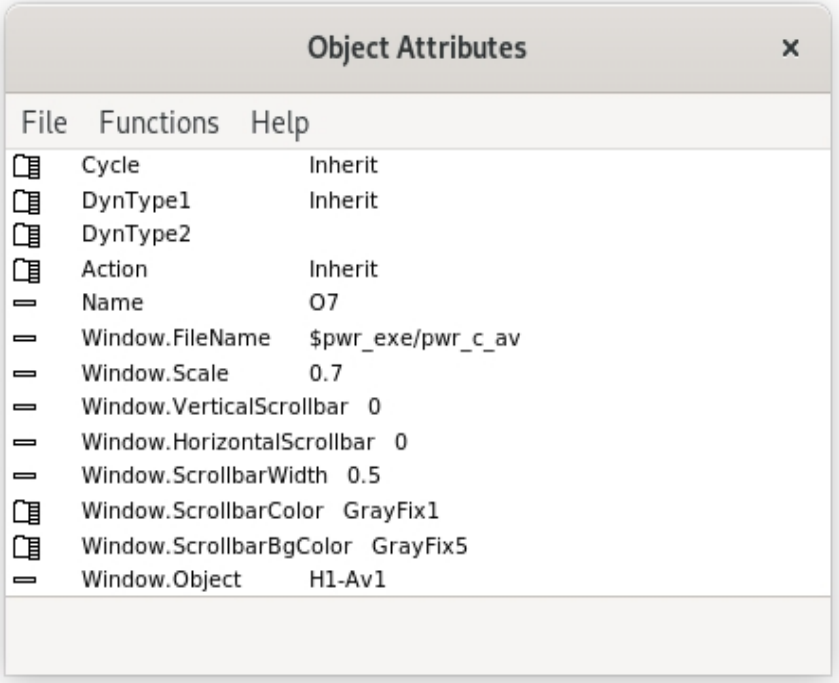


Fig Attributes for a window object displaying an object graph

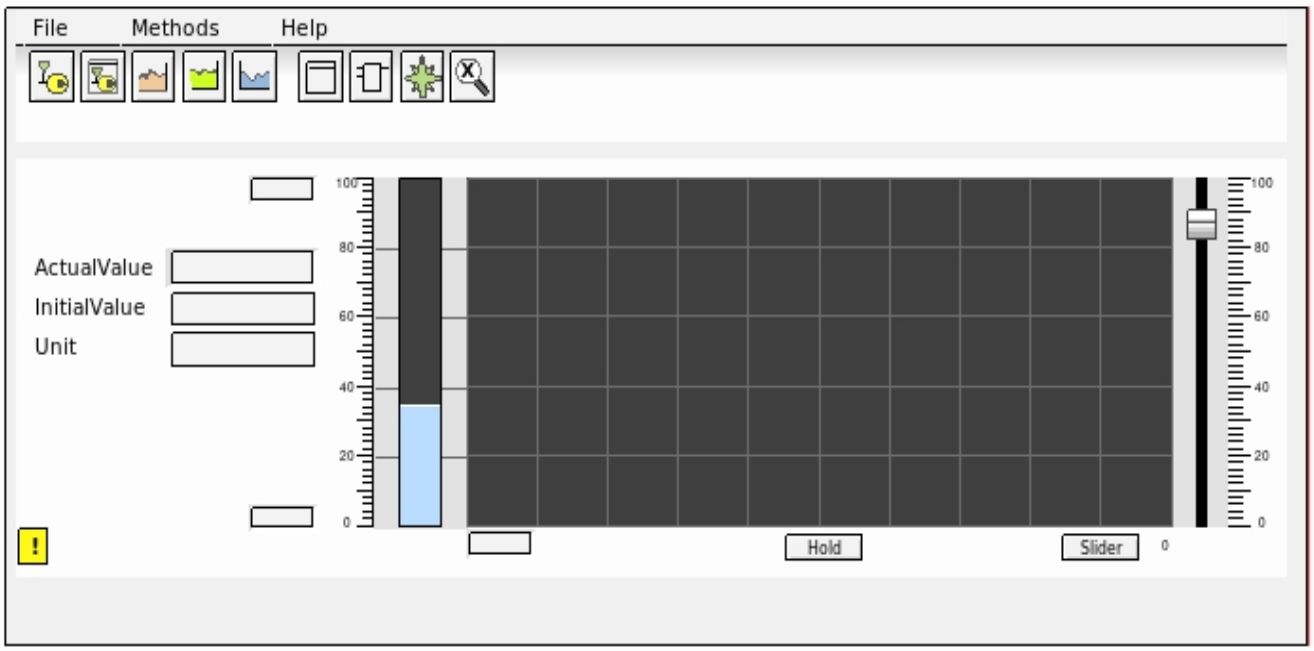


Fig Object graph for an Av displayed in the window object

Shift between two different graphs

There is an xtt command that displays a specific graph in a window object. By setting this command in a number of pushbuttons you can shift between different graphs in one window object. The function is similar to a TabbedWindow. The difference is that you have a more free layout of the push buttons, and that there is no limitation in the number of graphs that can be displayed.

The command is

```
set subwindow 'graphname' /name= /source= [/object=]
```

where grapname is the name of the graph that contains the window object. In /name the name of the window object is specified, and in /source the filename of the graph that is to be displayed. /object is used when an object graph is displayed, and specifies the database object the graph should be connected to.

In the following example, the object graphs for a number of Dv are displayed in a window object. You can insert a start graph in Window.FileName, but you can also let the window object be empty until one of the buttons are activated. In this case you should insert "_no_" in Window.FileName to avoid and error output.

The graph containing the window object is saved with the name 'dvdisplay'. It also contains 6 pushbuttons of type CommandButtonCenter. The window object is named 'DvWindow' by activating Edit/Change Name in the menu. In the figure below the object editor for a pushbutton is viewed with the command inserted. Note that the filename has to be surrounded by quotation marks as it contains a slash.

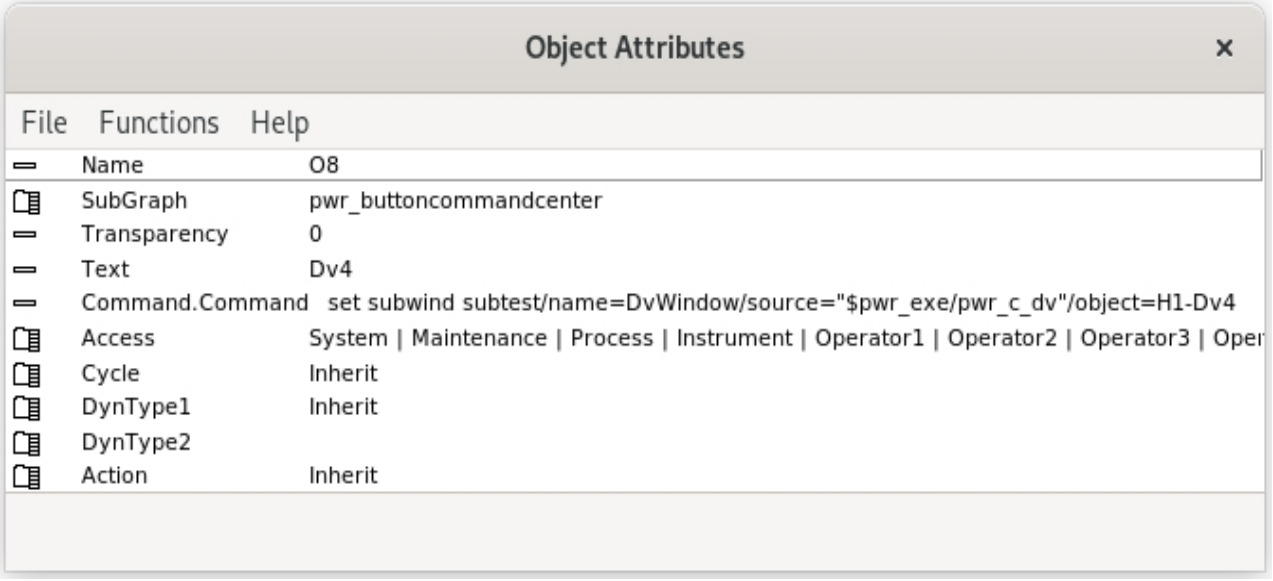


Fig Command 'set subwind' in a pushbutton

The final graph will look like this with Dv4 selected.

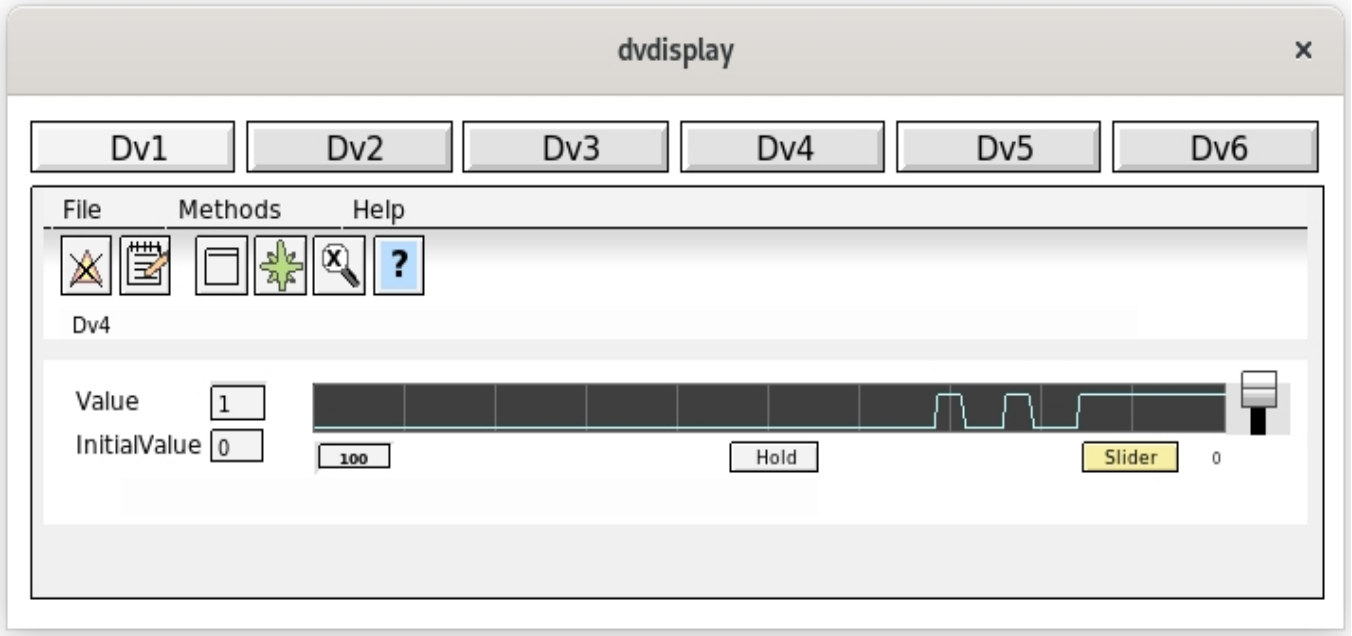


Fig The object graph for Dv4 is displayed after a click on the Dv4 button

4.4.2 TabbedWindow

TabbedWindow is a Window object that allows multiple graphs to be viewed in one window, using tabs to switch between graphs. For each tab a specific graph is stated, and when the graph is activated the stated graph is displayed in the window.

You can display object graphs or other graphs in a tabbed window. We will have a look at how to display a set of object graphs, as in the window example above.

Fig A tabbed window object

We will have 6 tabs and each tab will display the object graph of a Dv object. When the TabbedWindow is created, we open the object editor and set Folder.NumberOfFolder to 6. We also increase the tab size by changing Folder.HeaderHeight to 1. Then we insert FileName, Text and Object for the 6 first tabs. Filename is the filename of Dv object graph, ie $pwr_exe/pwr_c_dv. In Text we set Dv1, Dv2 etc and Object contains the object name of the different Dv objects, H1-Dv1, H2-Dv2 etc. We also increase the text size of the tabs by selecting the window object and set TextSize to 14 in the tool panel.

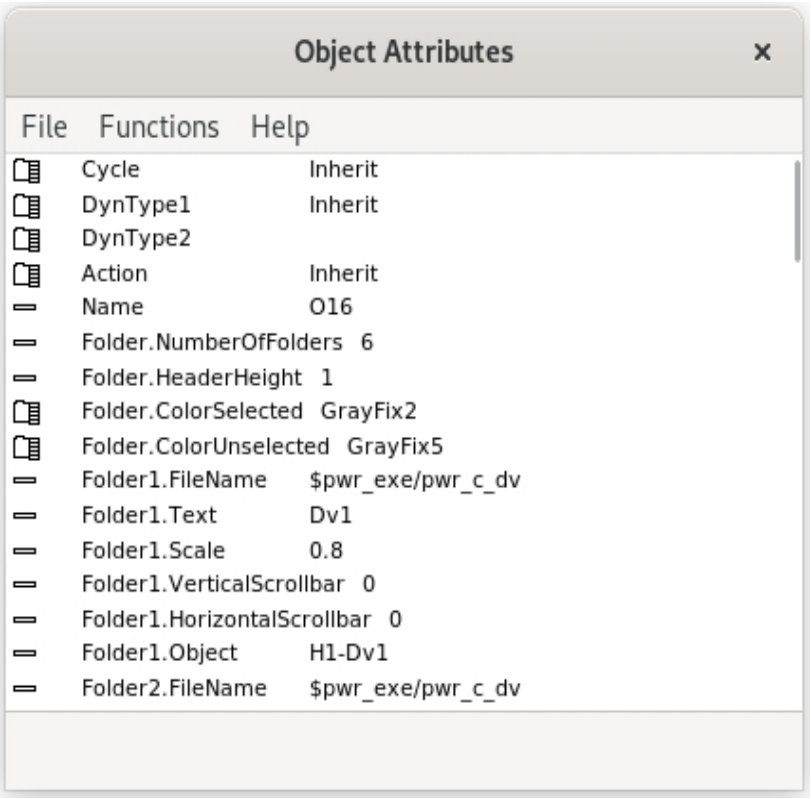


Fig The object editor of the tabbed window

4.4.3 Slider

A slider is a movable object connected to an analog signal in the database. The position of the slider states the value of the signal. The slider can be moved either horizontal or vertical between two endpoints. The position of the endpoints can be specified by two different methods. Either by placing a special background object of type SliderBackground,

or by specifying the coordinates for the endpoints in the object editor.

Slider with background object

Under the Slider map in the subgraph palette you will find slider and slider background objects. We select a SliderBackground1 and place a Slider1 on top of it. To get a horizontal slider we rotate both object 270 degrees.

Fig Slider with background object

The next step is to connect the slider to an analog signal in the database. We select the Av object H1-Av in the plant hierarchy and click with Ctrl+Doubleclick MB1 on the slider object. We also have to state the range in signal units that the slider movement corresponds to. The signal range is 0 - 1000 and we set Slider.MaxValue to 1000 in the object editor. Note that we don't have to set the min and max position as they are fetched from the slider background object.

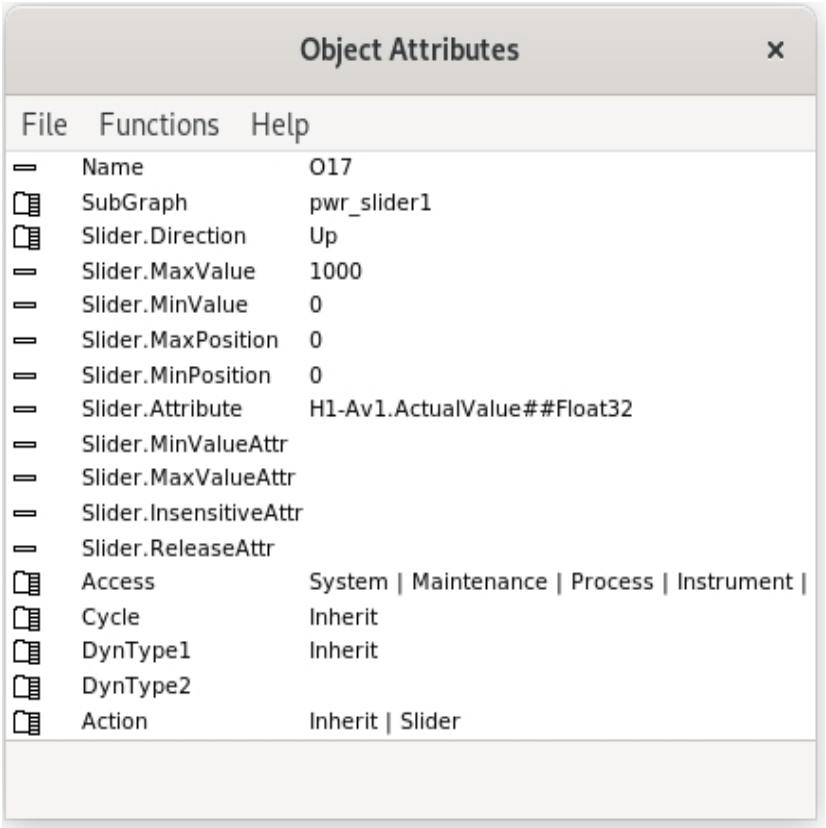


Fig The range for the signal is stated in Slider.MinValue and Slider.MaxValue

Slider without background object

For a slider without background object you have to state the direction of the slider movement and the min and max position of the movement.

Fig Slider without background object

We create a slider of type Arrow2 that is to be moved along an axis with a scale of range 0 - 100. 100 is positioned on the y-coordinate 0, and 0 is positioned on the y-coordinate 30. The slider should be placed on the position with the lowest y-coordinate, which is at the value 100 on the scale. Then we should state the movement range of the slider and measure the coordinate at the top of the slider object. This gives the y coordinates -1 in the upper position of the slider, and 29 in the lower position. We insert these coordinates in Slider.MinPosition (-1) and Slider.MaxPosition (29). We also set Slider.Direction to Up as a motion upwards in the graph will give increasing signal value.

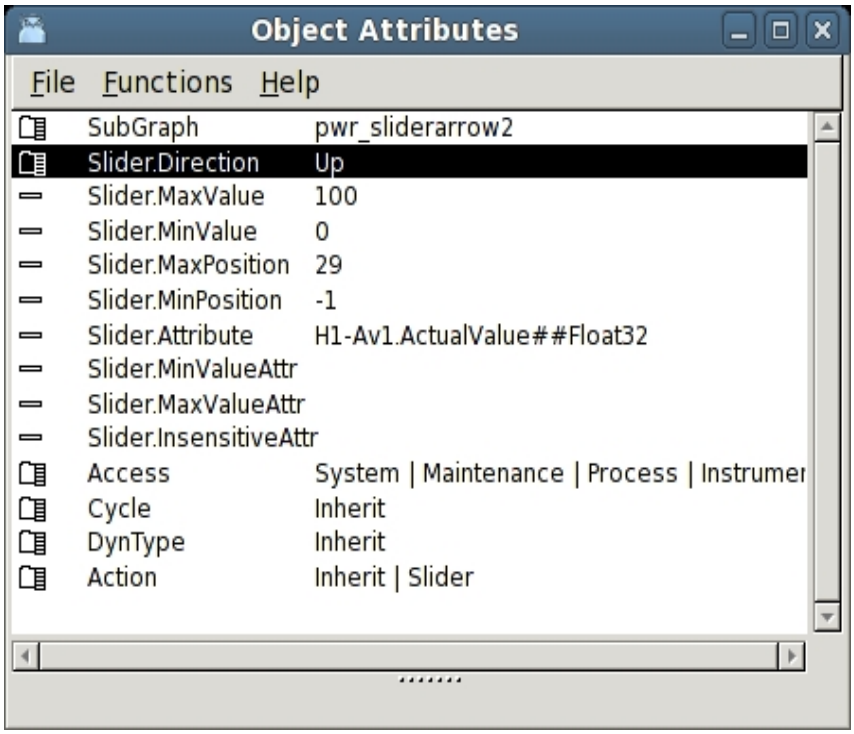


Fig The object editor of the slider

A horizontal slider is placed on the lowest x-coordinate, that is in the rightmost position in the graph. In the example below Slider.Direction is set to Left.

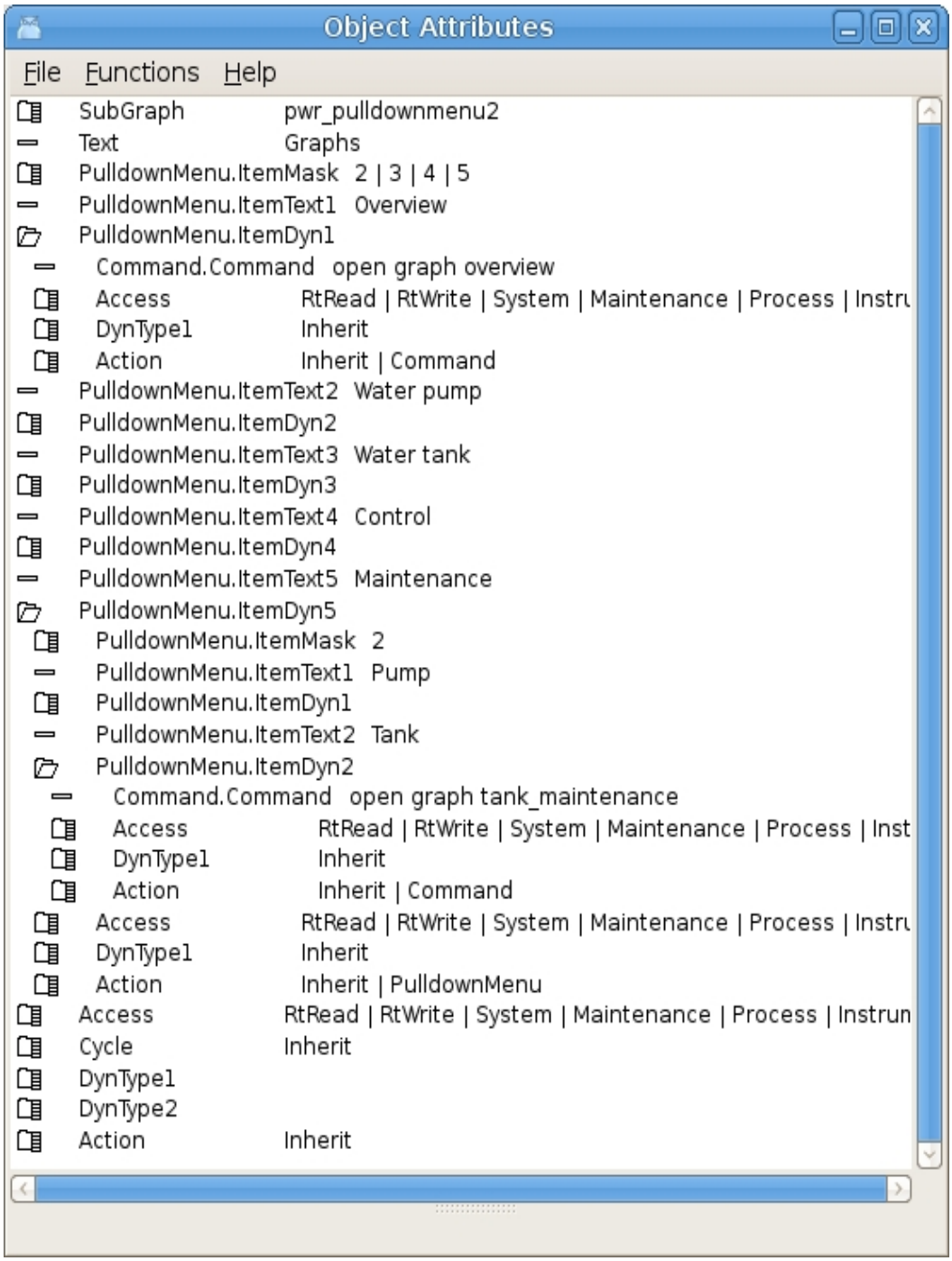


Fig Horizontal slider

4.4.4 Bar

A bar displays the value of an analog signal. The bar is found under the Analog map in the subgraph palette.

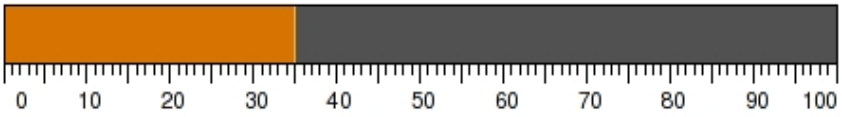


Fig Bar

The bar is configured by connecting it to an analog signal, and insert the range for the bar in Bar.MinValuie and Bar.MaxValue. The axis object in the figure is not a part of the bar object.

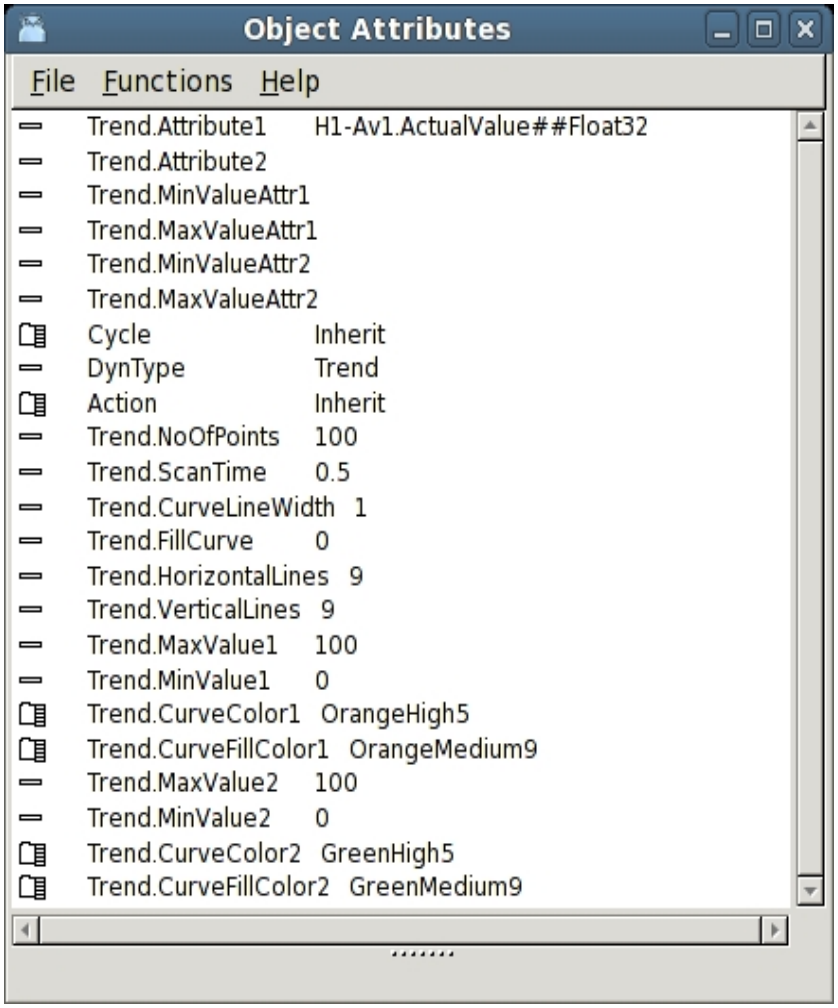


Fig Bar object attributes

If the bar is a part of an object graph, the range can differ between different instances of the database object that the graph shows. It is then possible to connect Bar.MinValueAttr and Bar.MaxValueAttr to the attributes in the database that contains the min and max value for the range of the signal.

Note that a bar also can be accomplished by a rectangle that is grouped and provided with FillLevel dynamics. The difference is that the bar object has a border line with separate color between the bar and the background.

4.4.5 BarArc

A bar arc is a bar in a circular shape. It is found under the Analog map in the subgraph palette.

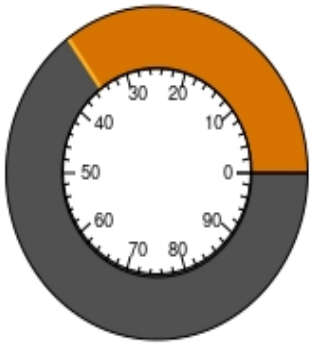


Fig BarArc

In addition to the linear bar properties, also the bar width, the start angel and the extension angle can be specified.

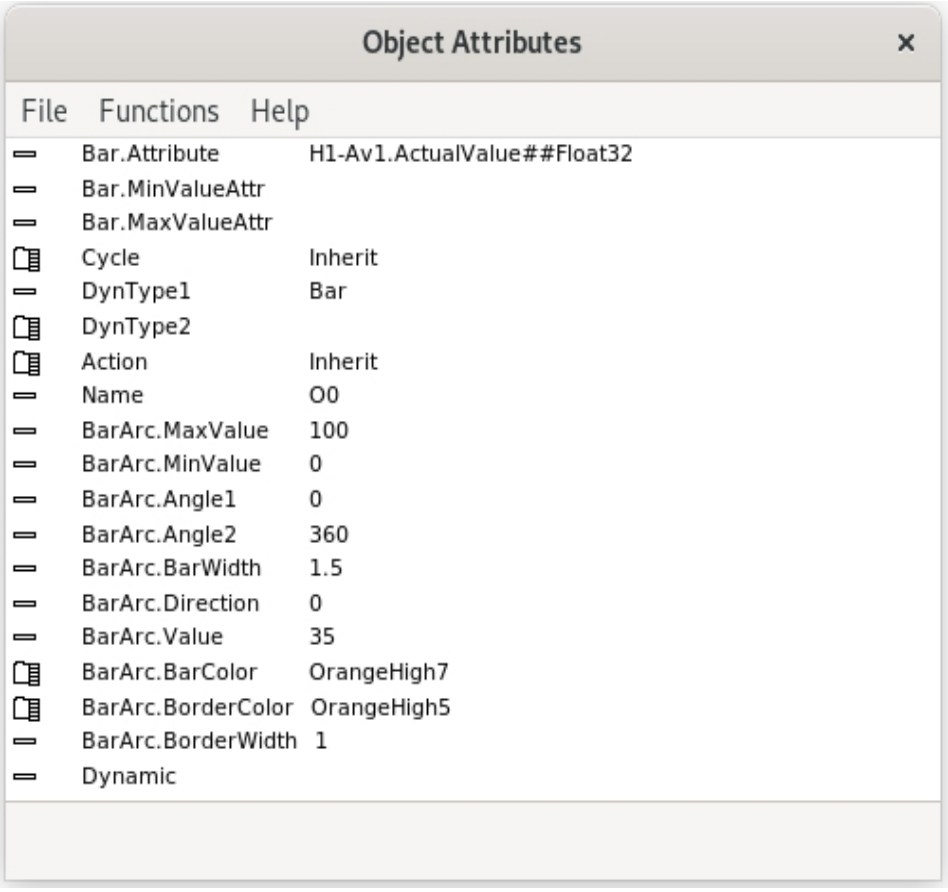


Fig BarArc object attributes

4.4.6 Pie

The pie chart can be divided in up to 12 sectors with different color, each sector connected to an analog signal that specifies the size of the sector.

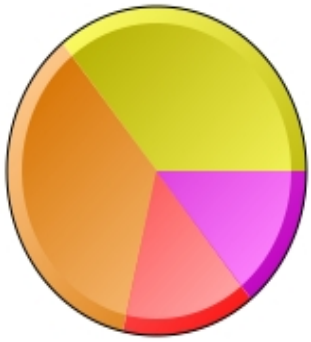


Fig Pie

In addition to the linear bar properties, also the bar width, the start angel and the extension angle can be specified.

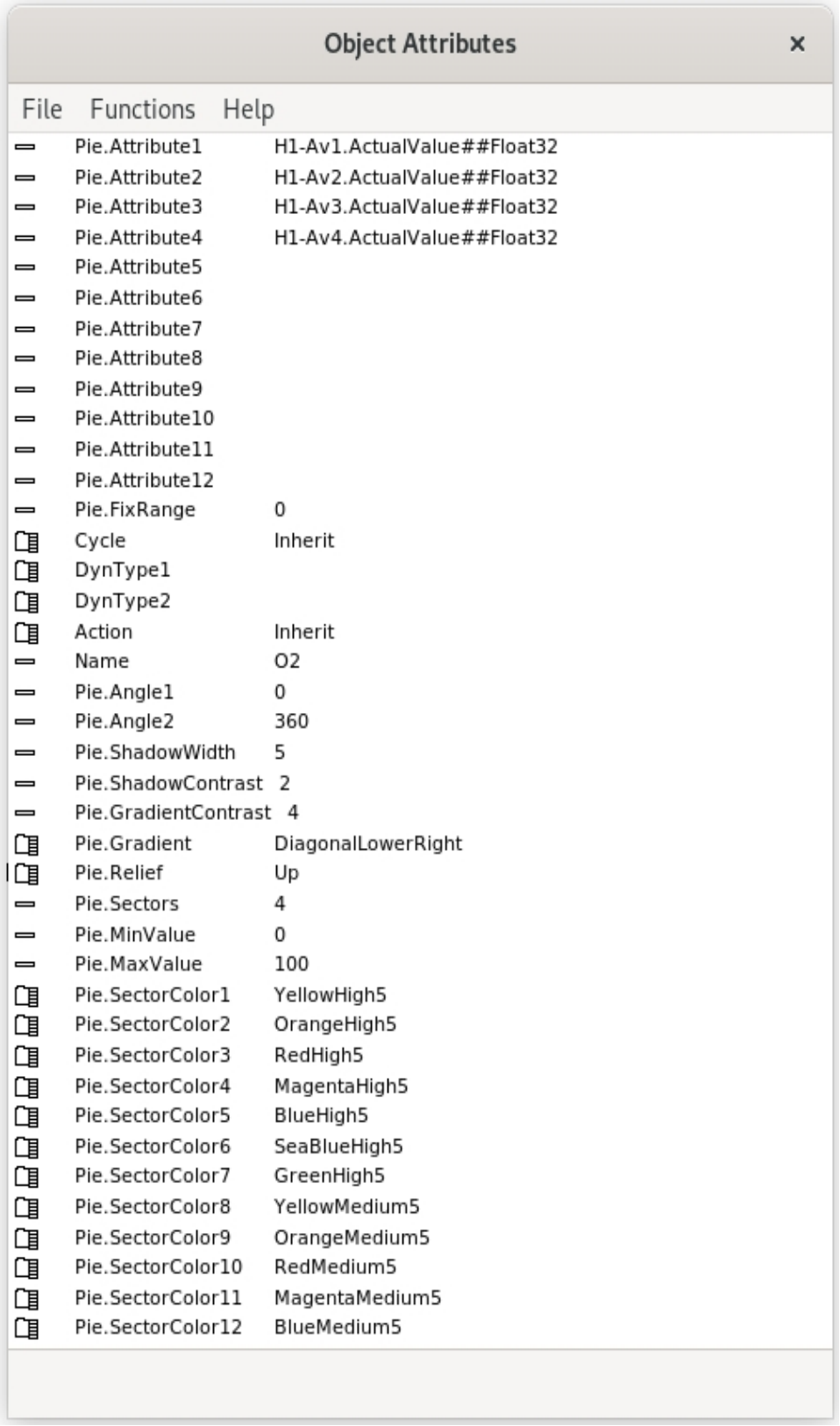


Fig Pie object attributes

4.4.7 BarChart

The bars of the bar chart can be divided in up to 12 segments with different color, each segment is connected to an analog array that specifies the size of the segments.

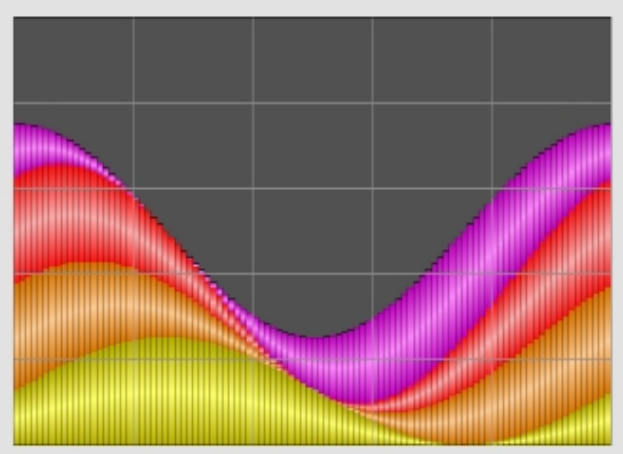


Fig BarChart

In the example above with hundred bars and four segments, the four first segments are connected to arrays of size 100. BarChart.Bars is set to 100 and BarChart.Segments is set to 4.

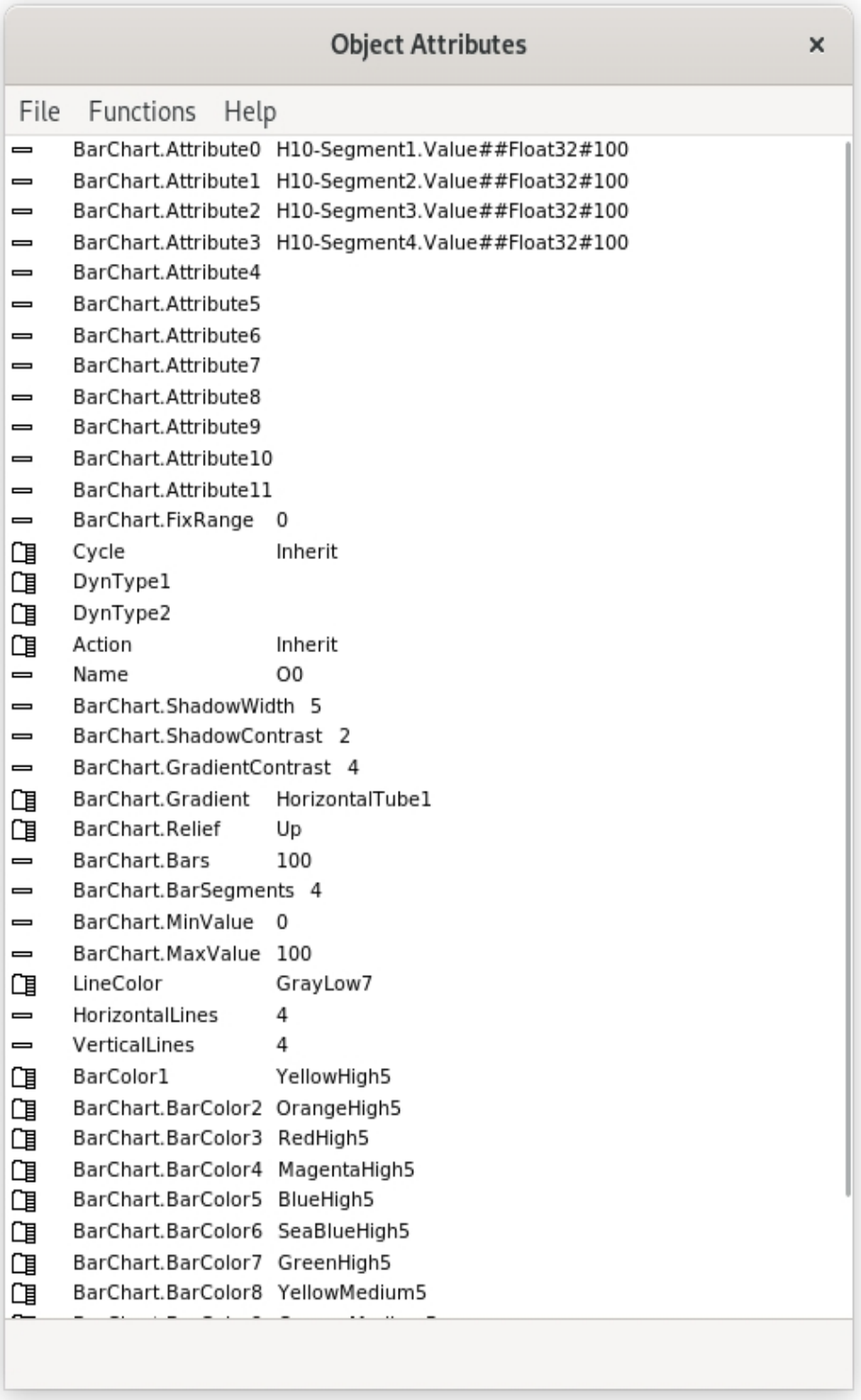


Fig BarChart object attributes

4.4.8 Trend

A trend displays a curve of the value of one or two signals. The signals can be analog, digital or integer. Trend is found under the Analog map in the subgraph palette.

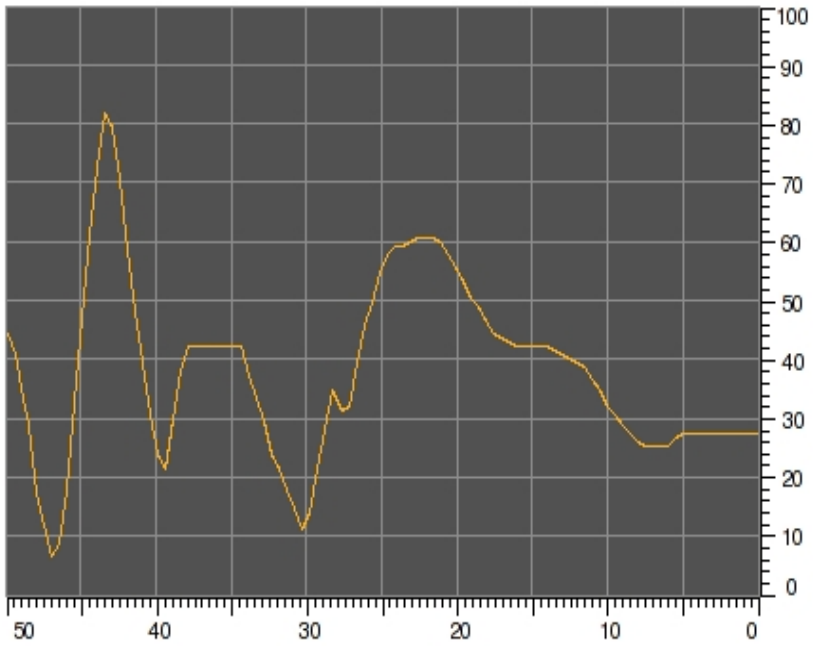


Fig Trend curve

The trend object above is configured with an analog signal. The number of vertical and horizontal lines has been increased by setting Trend.HorizontalLines and Trend.VerticalLines to 9. The axes are not included in the trend object, but has to be created with axis objects. The vertical axis has the range 0-100 as Trend.MinValue1 is 0 and Trend.MaxValue1 is 100. The time axis range is 50 s as Trend.ScanTime is 0.5 and the number of stored points, Trend.NoOfPoints is 100.

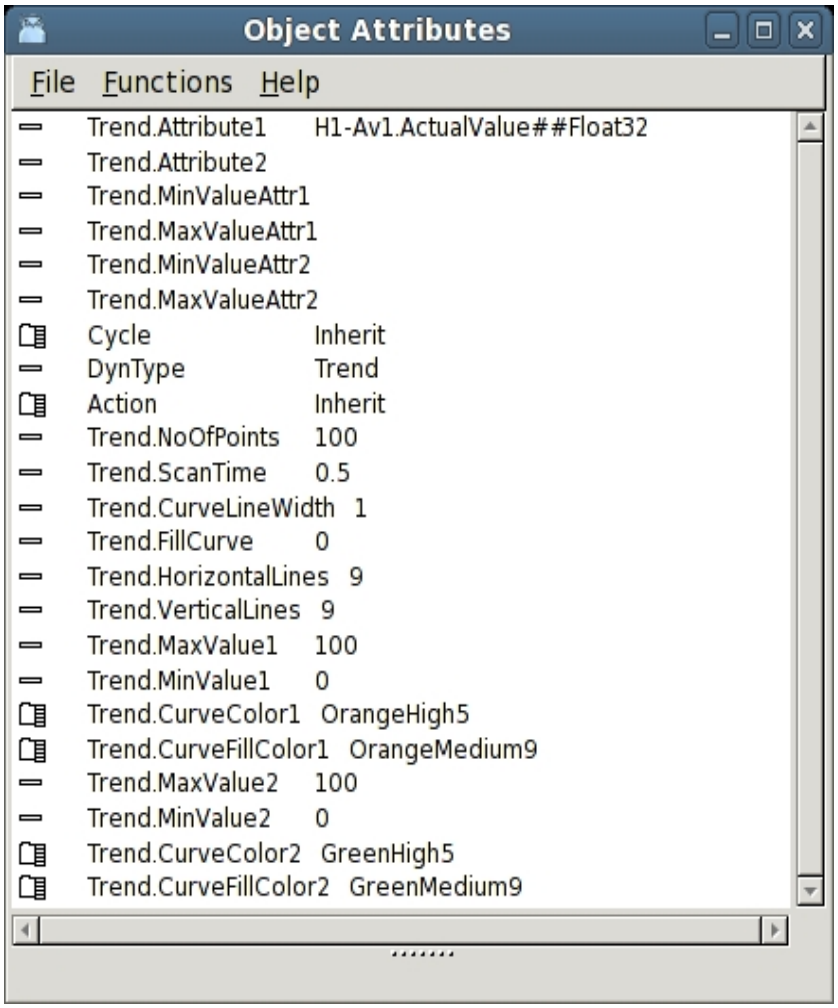


Fig Attributes for a trend object

If the trend is a part of an object graph, the range can differ between different instances of the database object that the graph shows. It is then possible to connect Trend.MinValueAttr and Trend.MaxValueAttr to the attributes in the database that contains the min and max value for the range of the signal.

4.4.9 XYCurve

The XYCurve plots a curve specified with two arrays containing the x and y values for points in the curve. There is a special XYCurve class that is convenient to use, with all the necessary attributes and two arrays with 100 elements.

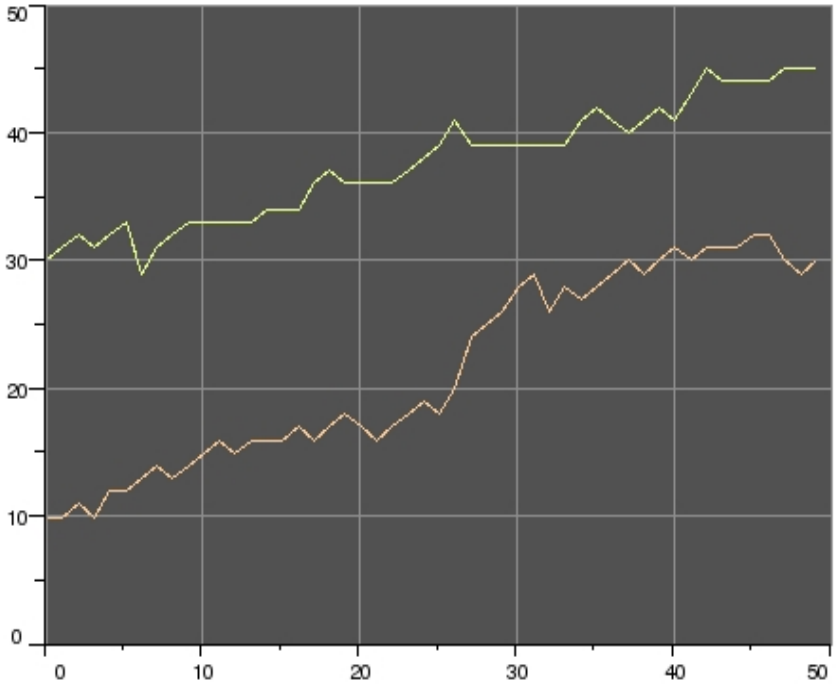


Fig XYCurve

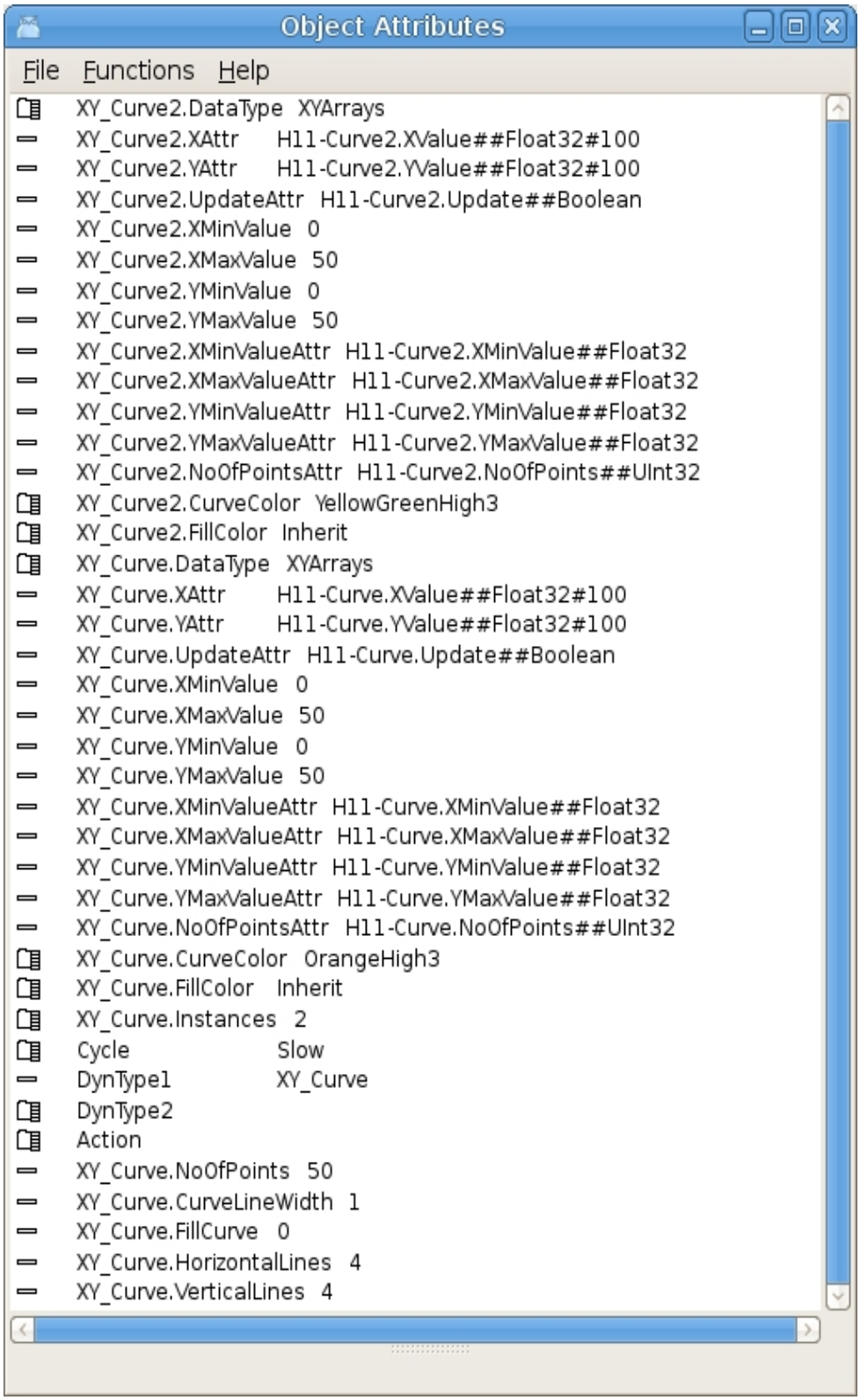


Fig Attributes for an XYCurve object

4.4.10 FastCurve

The FastCurve displays a curve specified by a DsFastCurve object. A DsFastCurve can contain 10 curves, but the FastCurve object can only show two of them. Which curve are specified with the FastCurve.FastIndex attributes.

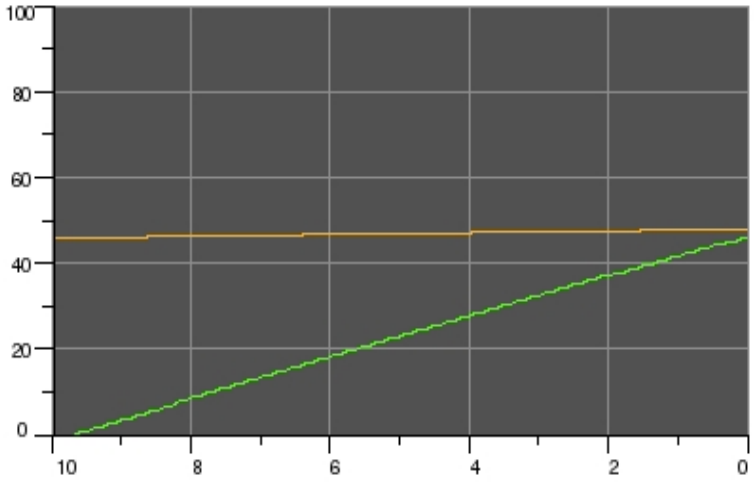


Fig FastCurve

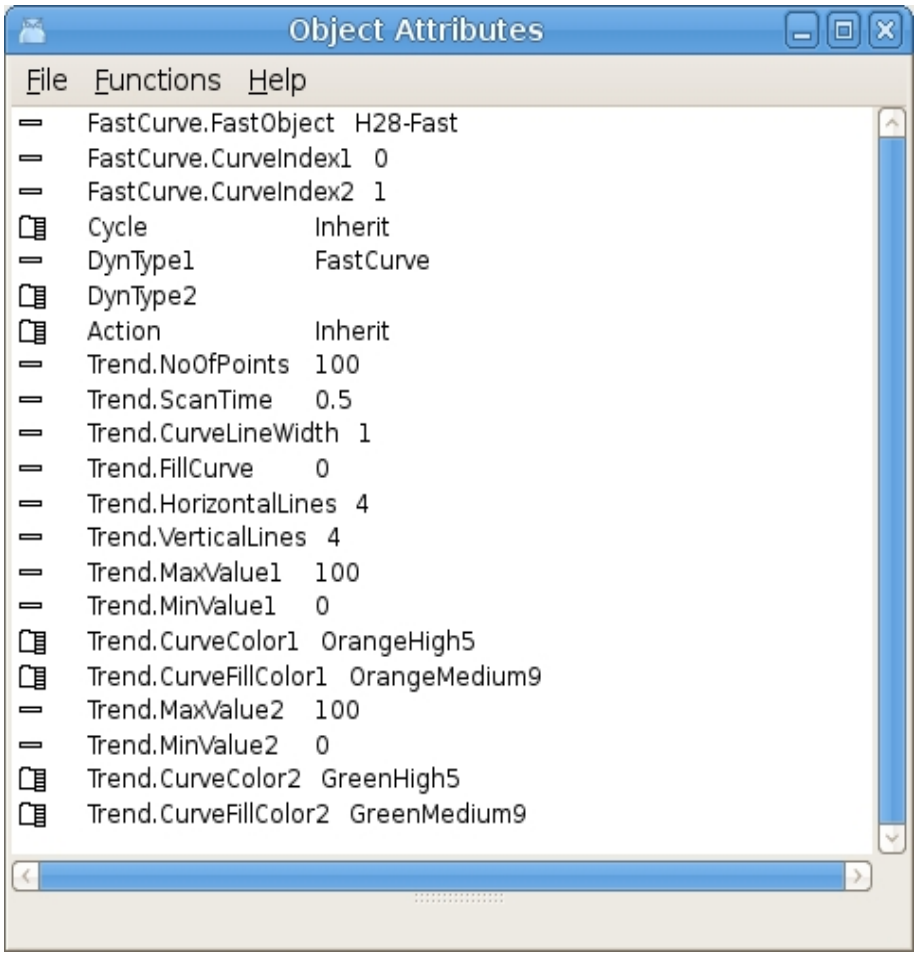


Fig Attributes for a FastCurve object

4.4.11 Axis and AxisArc

An axis object draws a scale with a specified range. Axis objects don't have any dynamics and the scale is fix. There are two variants, a straight axis and a circular or elliptic axis.

Straight axis

A straight axis is an Axis object fetched from the Analog map and scaled to desired size.

The size of the digits are adjusted from the TextSize menu in the tool panel. The range of the axis is determined by MinValue and MaxValue that in the example below are set to 0 and 50. If the scale should be in the opposite direction you shift the values in MinValue and MaxValue. It is not required that MaxValue should be greater than MinValue.

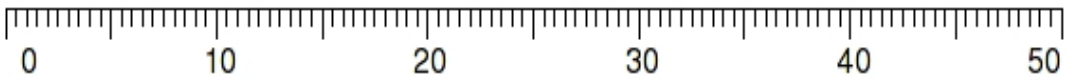


Fig Straight axis

The number of perpendicular lines are stated in the Lines attribute, that is set to 101. As every tenth line should be a little longer, LongQuotient is set to 10. Every twentieth line should be marked with a value, thus ValueQuotient is set to 20.

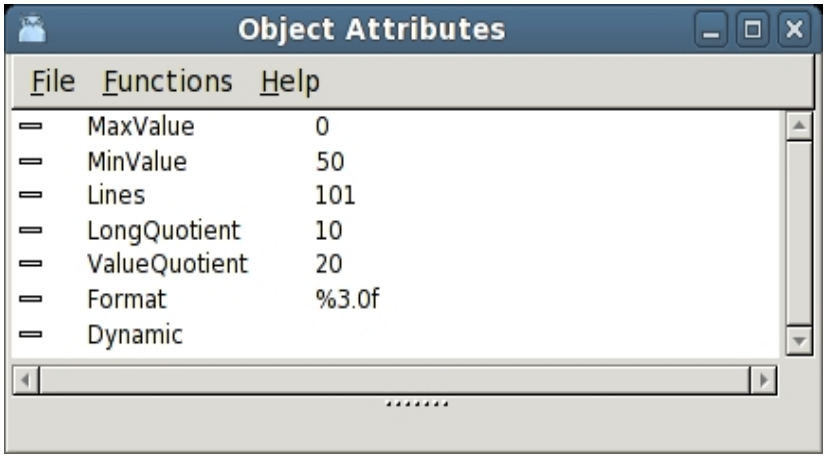


Fig Attributes of an axis object

It is also possible to affect the format of the values with the Format attribute, and if no values should be drawn, Format can be cleared. The format should be of type %5.1f where 5 is the total number of characters, decimal point included, and 1 number of decimals.

Elliptic axis

An AxisArc object is a circular or elliptic axis. The object is scaled to suitable size, and the text size is adjusted with TextSize in the tool panel.

The axis consists of an arc, and from the object editor Angle1 and Angle2 can be specified to control the size and position of the arc. Angle1 is the angle from the horizontal axis to the start of the arc, and Angle2 is the angle for the extention of the arc.

You can also affect the length of the lines in LineLength. The length is specified in relation to the radius, so a LineLength of 0.1 will give a length on the longest lines of 10 % of the radius.

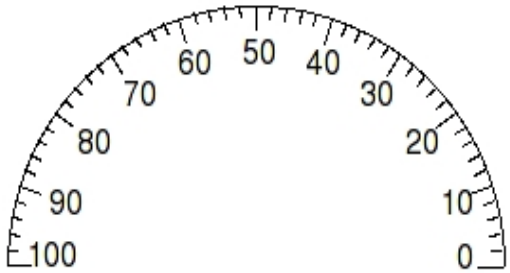


Fig Elliptic axis

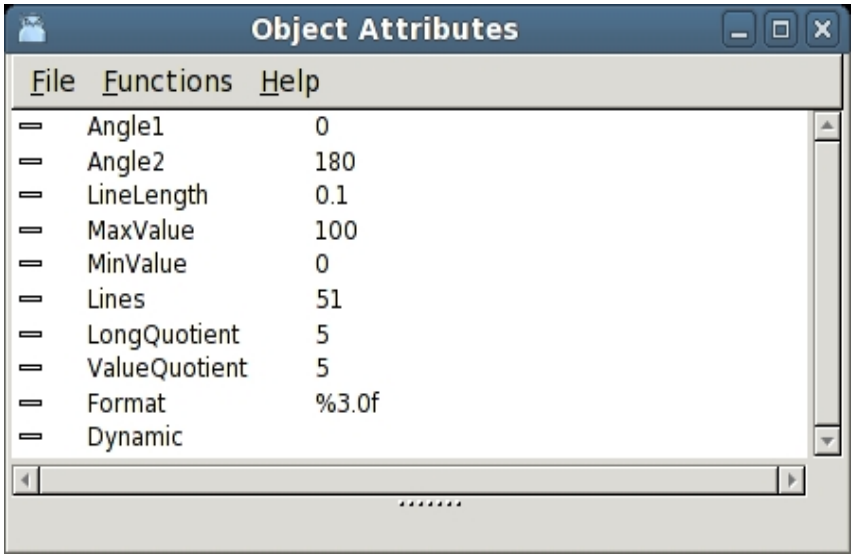


Fig Attribute of an axisarc object

4.4.12 Table

Table is a lucid disposition of data, organized in rows and columns, often provided with headers. Table is found under the Other map in the subgraph palette.

10.2	2.10
12.5	2.20
9.4	2.10
9.8	2.40
12.4	2.50
7.8	1.80
9.2	1.70
10.1	2.00
8.9	2.20
8.9	1.80

Fig Table without header

The data for one column is usually gathered in an array attribute in the database. There are a number of classes with arrays of different type, eg AArray100 and AArray500 the contains arrays with 100 and 500 elements respectively of type Float32. Corresponding objects also exist for boolean, integer and strings, DArray, IArray and SArray. In the example below data is gathered in 3 AArray100 objects and one SArray100.

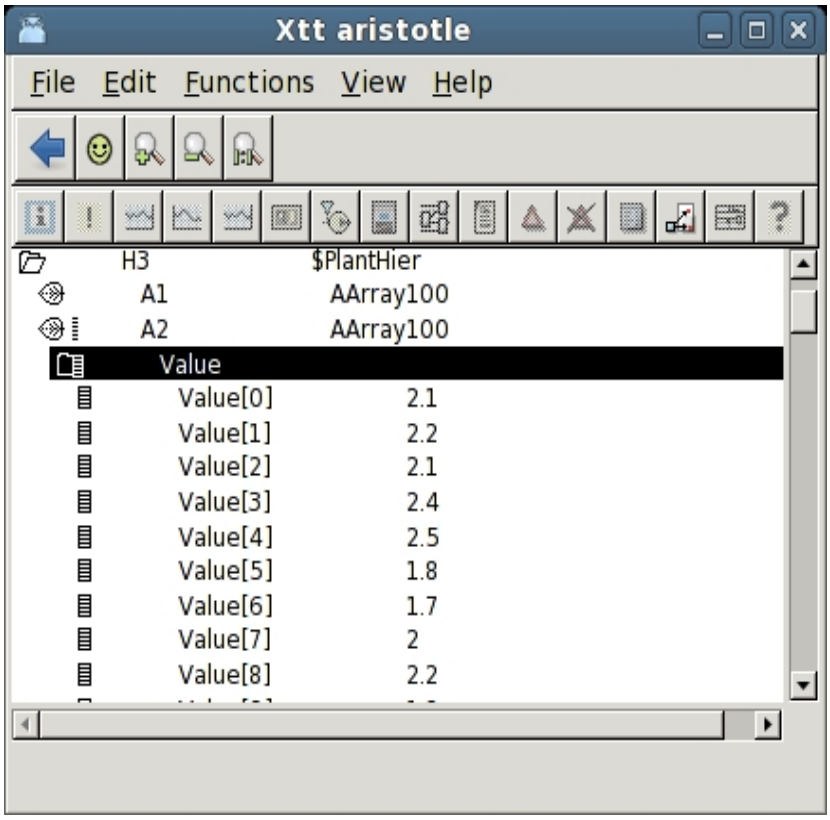


Fig Array objects contains table data

If we look at the simple table in Fig Table without header above, two columns are displayed. The number of columns are stated in Table.Columns. Column1.Attribute and Column2.Attribute are connected to two array attributes containing that data that is to be displayed in the columns, H3-A1.Value and H3-A2.Value. The syntax is H1-A1.Value##Float32#100 which means that the attribute is an array of type Float32 with 100 elements. You also has to specify the output format for the values in the columns, by inserting Column1.Format and Column2.Format. '%6.1f' means a float value with 6 characters and 1 decimal.

Some other details are modified in table object,
- The font is set to Lucida Sans, by selecting the table, and set Lucida Sans in the tool bar.
- The text size is increased to 12 from Textsize in the tool panel.
- The row height is adapted to the larger text size by setting Table.RowHeight to 0.8.

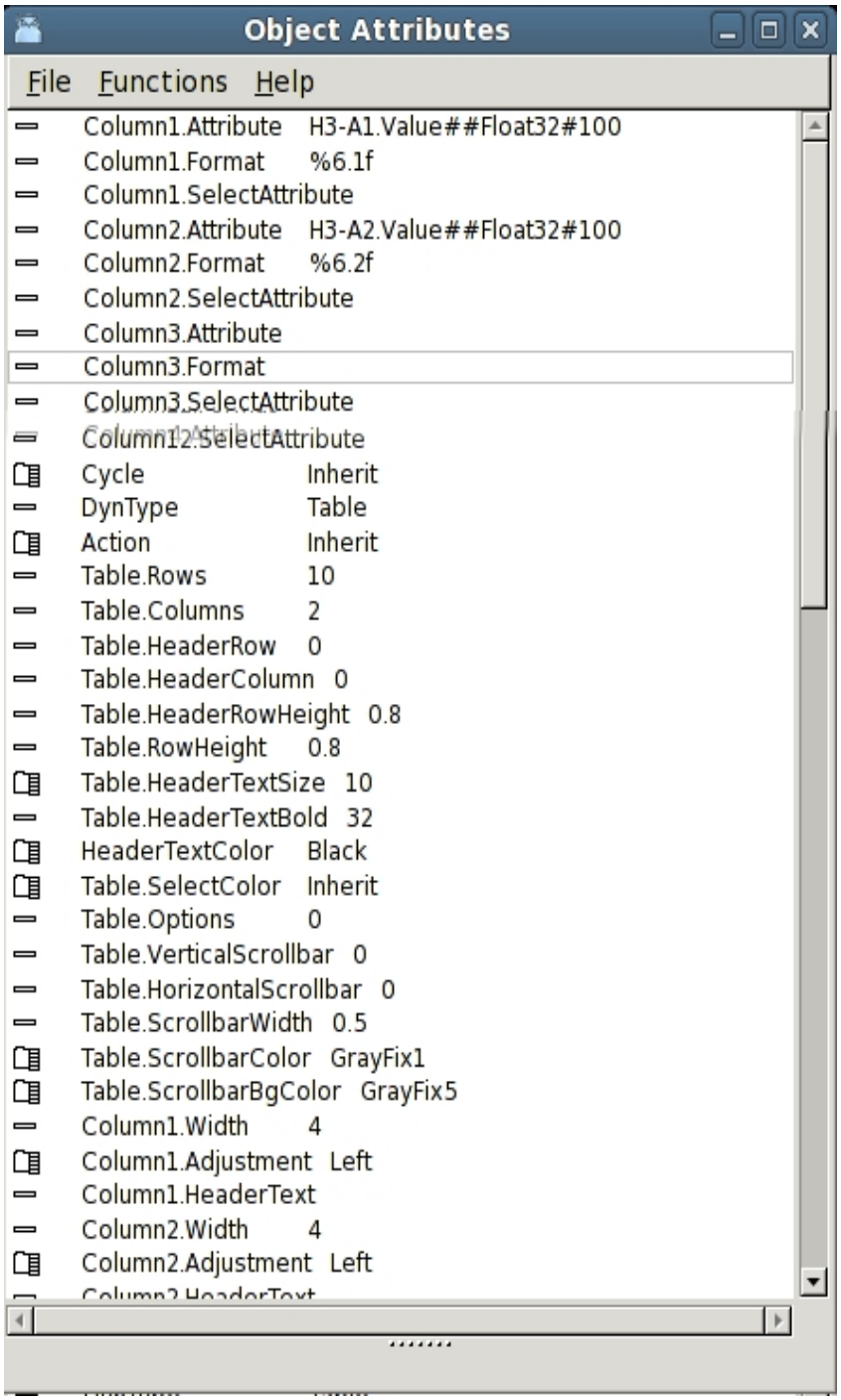


Fig Table object attributes

The figure below shows a table with a header. Here the Table.HeaderRow is set to 1. The text size for the header, Table.HederTextSize set set to 12 and Table.HeaderTextBold is set to 0 for normal text.

Length	Width
10.2	2.10
12.5	2.20
9.4	2.10
9.8	2.40
12.4	2.50
7.8	1.80
9.2	1.70
10.1	2.00
8.9	2.20
8.9	1.80

Fig Table with header

In the table below a vertical scrollbar is added by setting Table.VerticalScrollbar to 1. Note that the header row is not scrolled but always visible.

The table object displays 3 columns, the first is connected to a string array of type SArray100.Value, and the format Column1.Format is set to %s. The first column is also marked as a header column, by setting Table.HeaderColumn to 1. It implies that the border line between the first and second column is a little thicker. It also affects the horizontal scrolling which we will see in the next example.

Fig Table with vertical scrollbar and header column

The figure below shows how a header column works with a horizontal scrollbar. The horizontal scrollbar is viewed when Table.HorizontalScrollbar is set to 1. Note that the scrollbar does not cover the first column. The first column is always visible.

Fig Table with horizontal scrollbar and header column

Table displaying objects of the same type

Now we will look at an example where the data is not organized in arrays, but in a number of objects of the same class. Under H3-Data in the plant hierarchy there is a number of objects of class Thing. Thing has the attributes Length, Height and Width. We want to show the content of all the Thing objects in a table, with one object on each row. To do this we have to create an array of type Objid that contains the identity of each object that is to be displayed. Thus we create an object of type OidArray and insert the identities of all Thing objects.

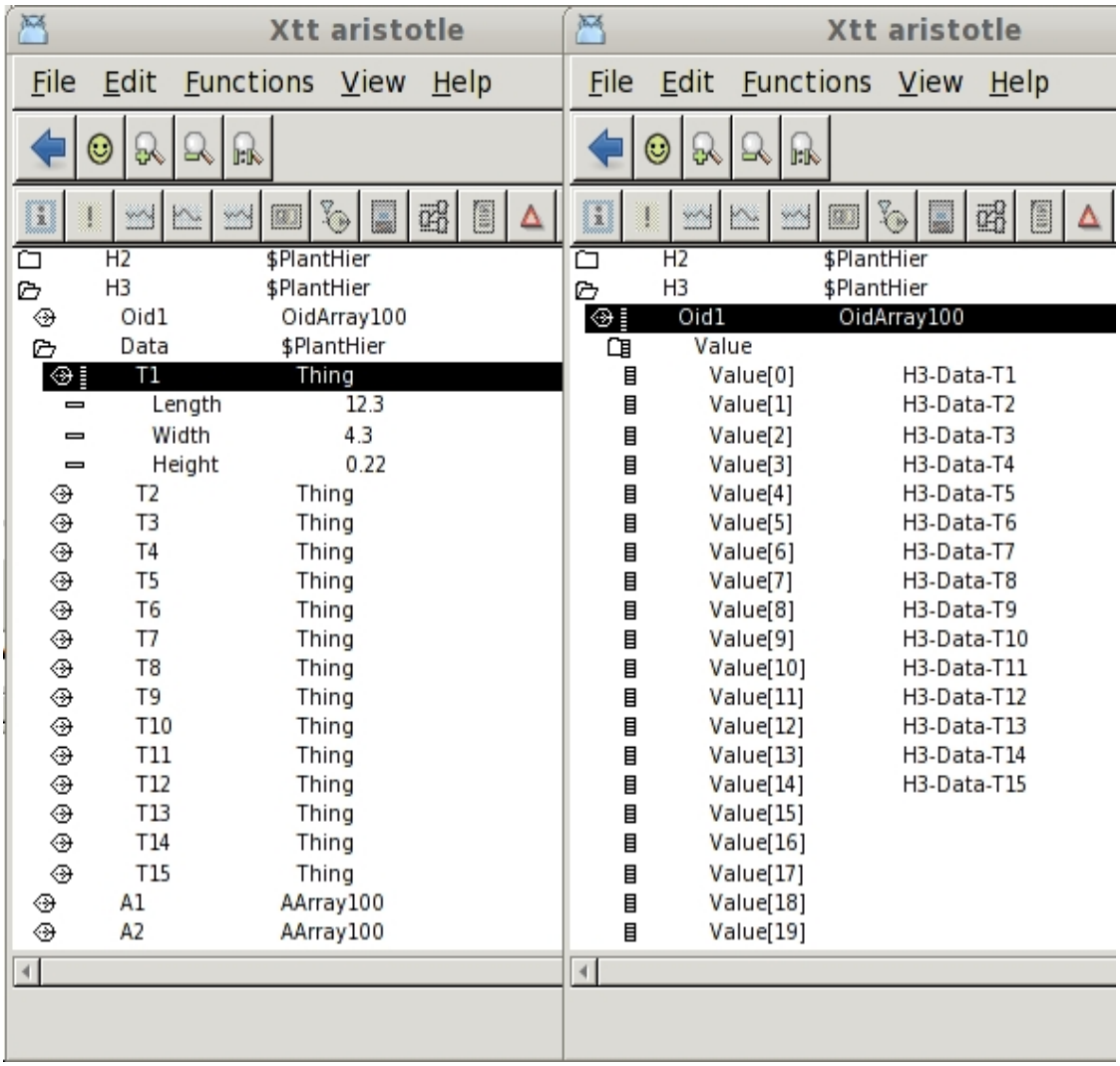


Fig The objects that will be displayed to the left, an array of the objid to the right

We create a table with four columns. The first column will show the name of the object, The second the Length attribute, the third Width and the fourth Height. The table is configured by connecting the array with the object identities to Column1.Attribute. In Column2.Attribute '$header.Length##Float32' is stated. $header points at the object displayed in the same row in the header column, and the Lengh attribute of this object will be displayed in this column. In the same way we state '$header.Width##Float32' in Column3.Attribute and '$header.Height##Float32' in Column4.Attribute.

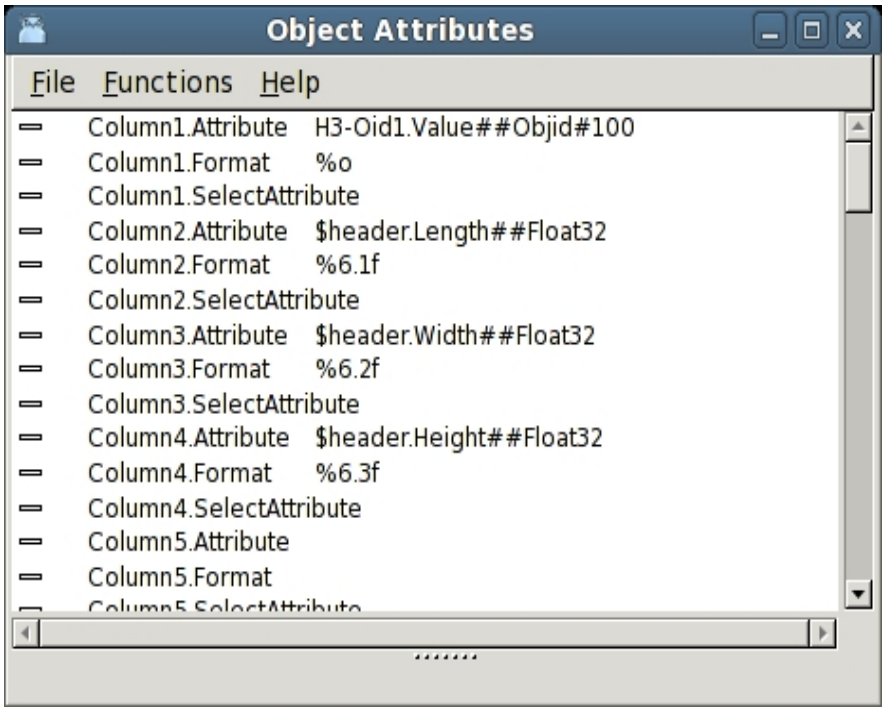


Fig The attributes of the table columns

The result is shown in the figure below. In the first column the name of the objects is displayed, and in the same row the content of the object.

Object	Length	Width	Height
T1	12.3	4.30	0.220
T2	12.2	4.50	0.220
T3	9.3	4.00	0.210
T4	11.1	3.20	0.190
T5	13.9	5.30	0.190
T6	16.4	3.90	0.220
T7	12.5	4.50	0.240
T8	11.6	3.20	0.190
T9	12.5	4.90	0.340
T10	12.0	5.10	0.210

Fig Table showing the object name and the content of the objects

Table with selection function

Sometimes you want to be able to select a cell in a table, and execute something in the plc program with the selected data or object. You do this by connecting a boolean array to the attribute SelectAttribute for a column. In the example below the array H3-Select1.Value is connected to Column1.SelectAttribute.

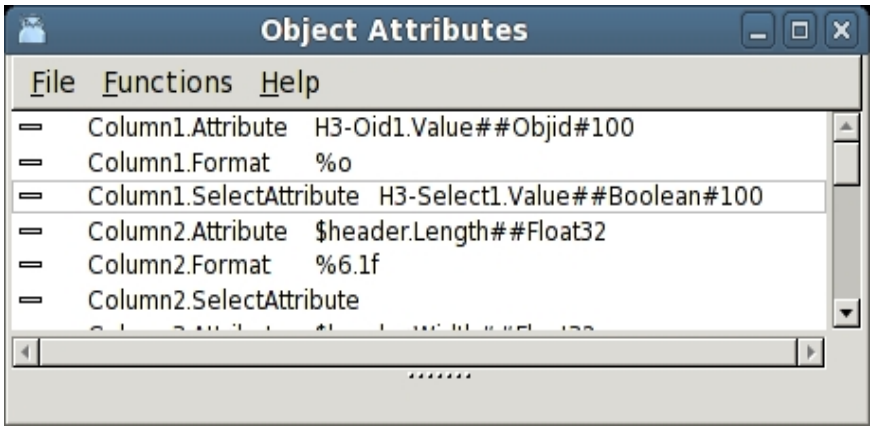


Fig A selection array is supplied for the first column

The first column will now be click sensitive, and the selected cell will be marked with a distinctive color stated in Table.SelectColor.

Object	Length	Width	Height
T1	12.3	4.30	0.220
T2	12.2	4.50	0.220
T3	9.3	4.00	0.210
T4	11.1	3.20	0.190
T5	13.9	5.30	0.190
T6	16.4	3.90	0.220
T7	12.5	4.50	0.240
T8	11.6	3.20	0.190
T9	12.5	4.90	0.340
T10	12.0	5.10	0.210

Fig The fourth row in the first column is selected

The element in the array that corresponds to the selected object will be set to 1. In the figure above, the forth row is selected, which implies that the fourth element in the select array will be set, ie H3-Select1.Value[3].

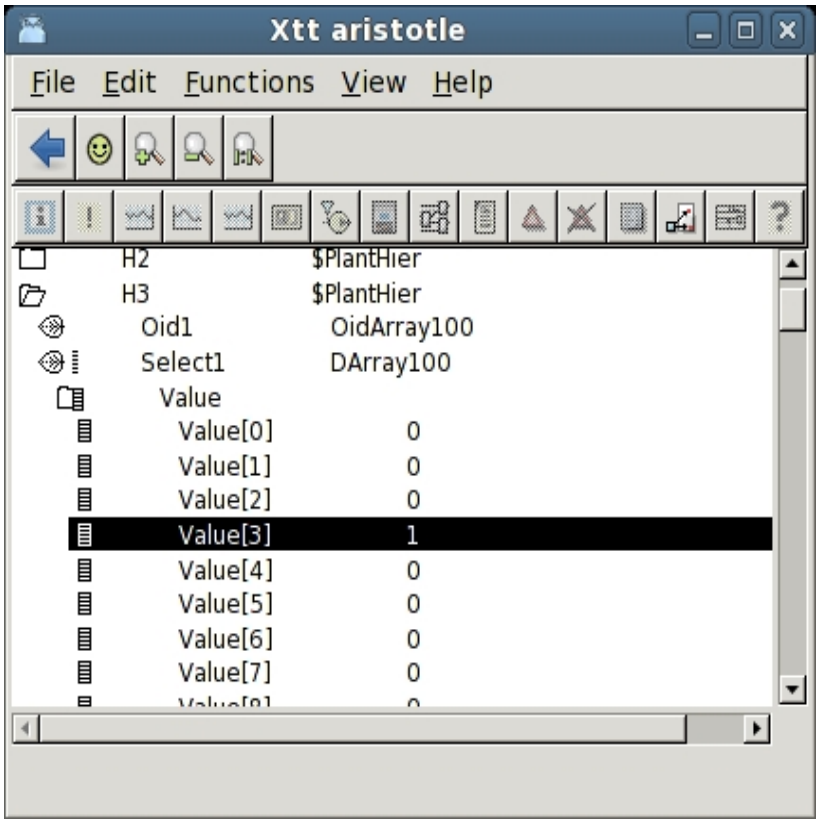


Fig The corresponding array element of the selected row is set

4.4.13 PulldownMenu

The pulldown menu contains a number of menu entries that can either open a new pulldown menu, or execute and action. Actions that can be defined for other buttons also are available for menu entries. With the action PulldownMenu submenus can be configured.

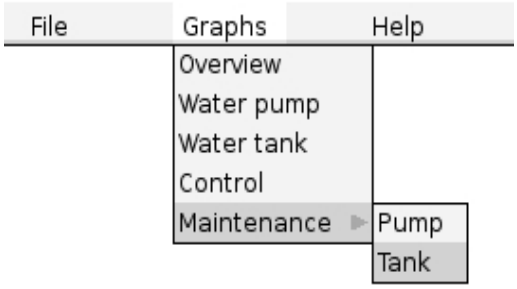


Fig Pulldown menu

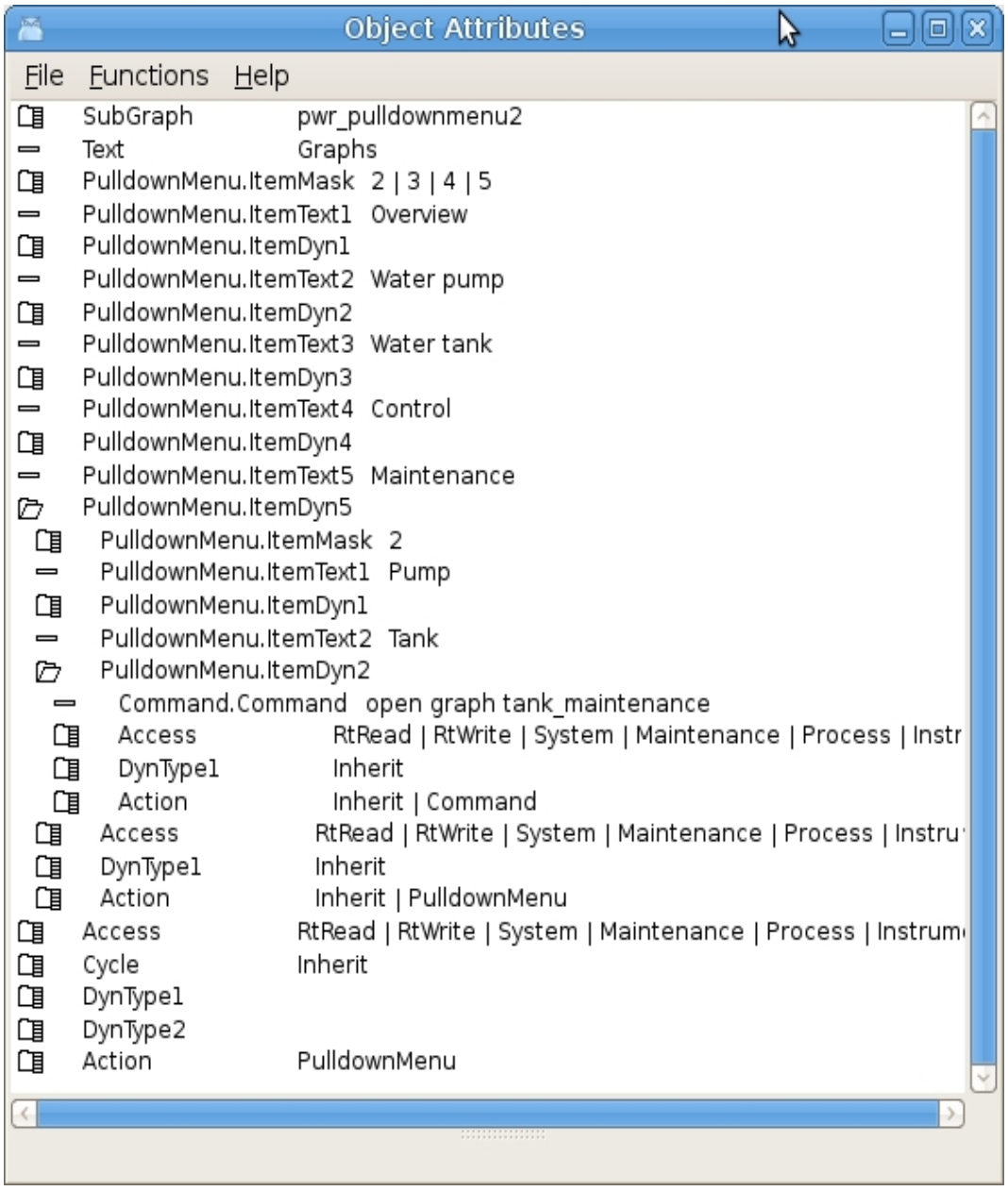


Fig The attributes for the pulldown menu

MethodPulldownMenu

The method pulldown menu contains menu entries to activate the methods of an object. It is usually use in object graphs. With the help menu options, only the help methods is displayed.

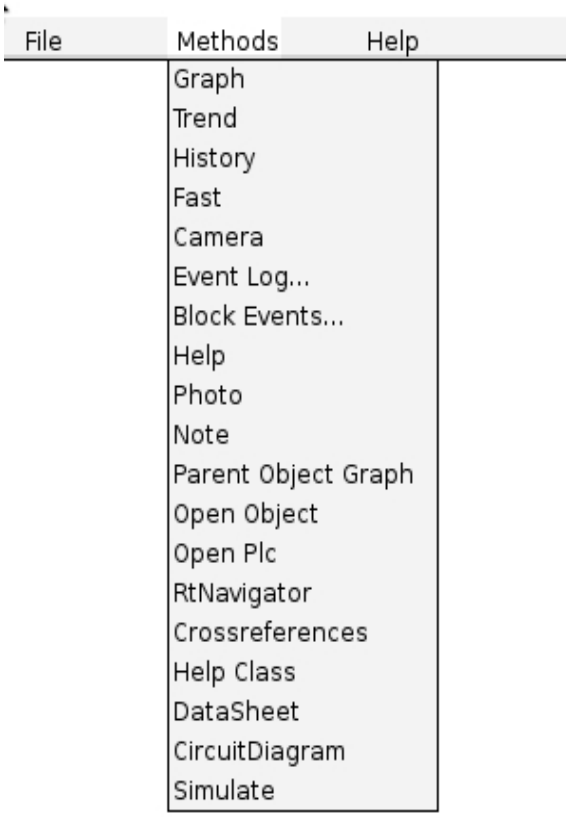


Fig Method pulldown menu

By setting the action to MethodPulldownMenu a method pulldown menu is configured.

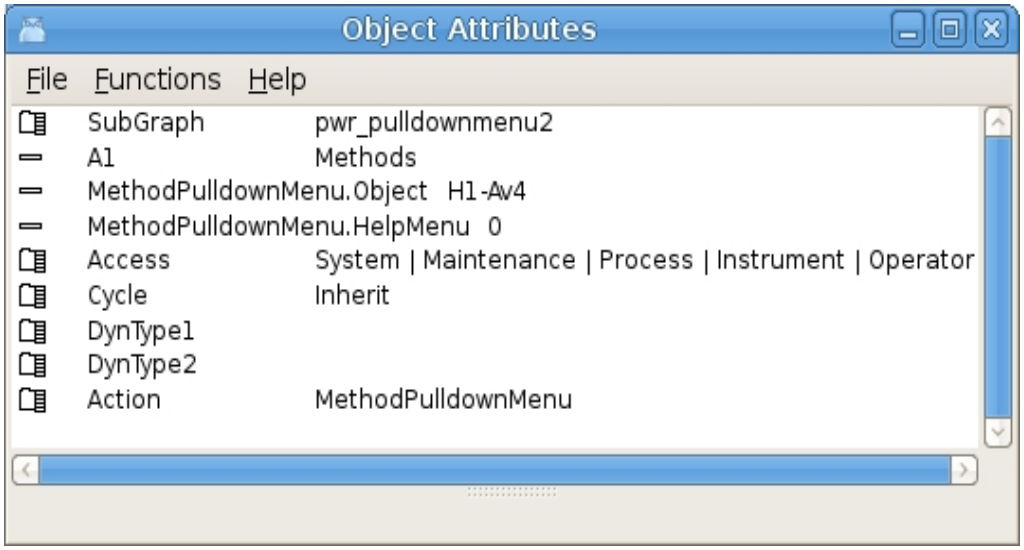


Fig The attributes for a method pulldown menu

4.4.14 OptionMenu

Option menu is menu where you select an alternative in a list of alternatives. The chosen alternative is displayed in the menu component. When you click on the component, the list of alternatives is displayed. When an alternative is chosen, the list is closed, and the selected alternative is displayed in the component.

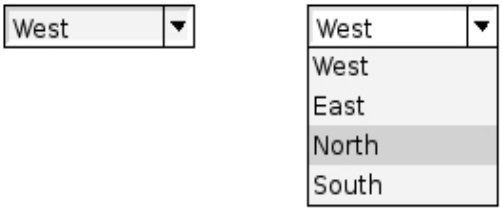


Fig Optionmenu

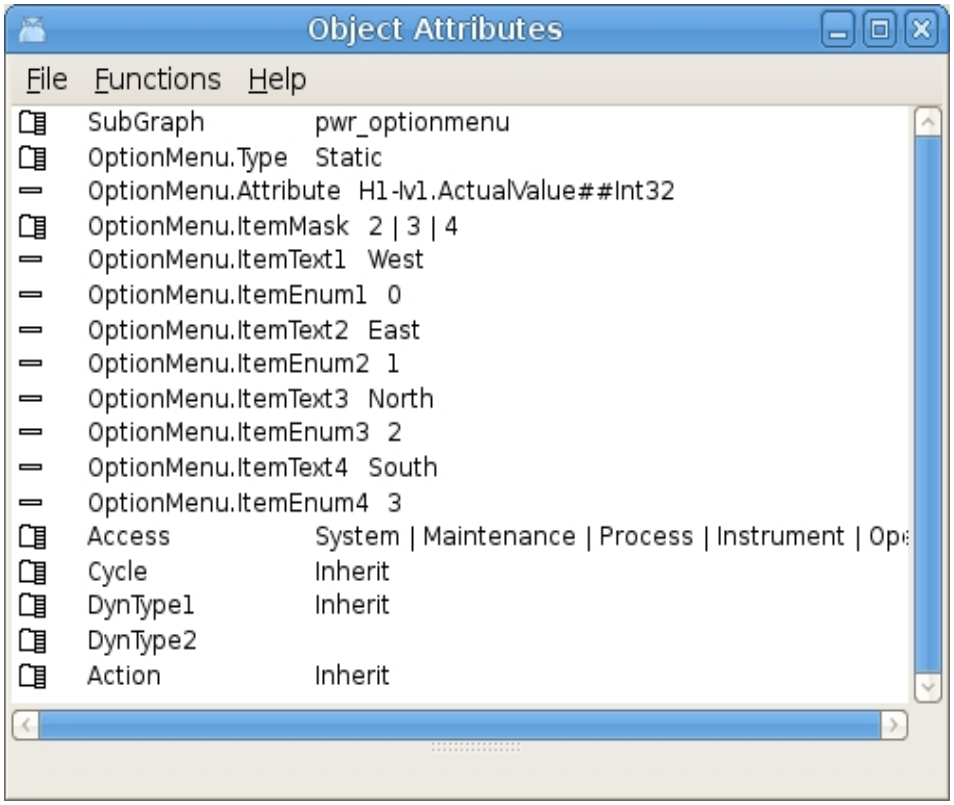


Fig The attributes for the option menu

4.4.15 MethodToolbar

The methodtoolbar displays a set of buttons to activate the methods for a specific object. Only methods that are configured for the object are displayed, and buttons that the current user doesn't have access to, are dimmed.

The methodtoolbar is usually used in object graphs.

Fig MethodToolbar

4.5 Dynamic and action

A process graph has basically two tasks, to show the status of the process to the operator, and to make it possible for the operator to affect the process.

To show the status of the process you insert dynamics in the graphs. The dynamic can be to

- change the color of an object
- make objects invisible or insensitive
- draw texts
- change the shape of an object
- show analog value
- move, scale or rotate objects
- Sound and commands

To make it possible for the operator to affect the process, you insert actions to objects in the process graph. By adding action to an object it can also be possible for the operator to gain more information about the object, from a popup menu or by opening additional graphs or curves. An action can be

- set values from pushbuttons
- popup menu
- open graphs
- execute commands
- tooltip
- help texts
- input focus

4.5.1 Change the color of an object

Shift between two colors

We will look at how to change the color of an object dependent on a digital signal in the database. There are a number of different dynamics available, DigLowColor, DigColor, DigError and DigWarning. We will start with a DigLowColor example.

DigLowColor shifts bewteen two colors, the first is the one you apply to the object in the editor, and this will be the color when the signal is high. The second color is specified in DigLowColor.LowColor. We will draw an orange indicator, that should be orange when the signal is high, and dark gray when the signal is low. Thus we draw the indicator with orange color in the editor, and set DigLowColor.Color to dark gray.

We begin by drawing a circle, and fill it with orange color. We also set the gradient to GradientGlobe.

Fig Orange indicator

We create an additional frame with a metallic look to the indicator by drawing a somewhat larger circle with gray fill color. On the frame, fixcolor in the object editor is set to 1, as the frame should not change color with the indicator lamp. We also set 3D and select the gradient DiagonalDownTube.

Fig Indicator frame

From the object editor gradient_contrast is increased to 8 and shadow_width to 9.

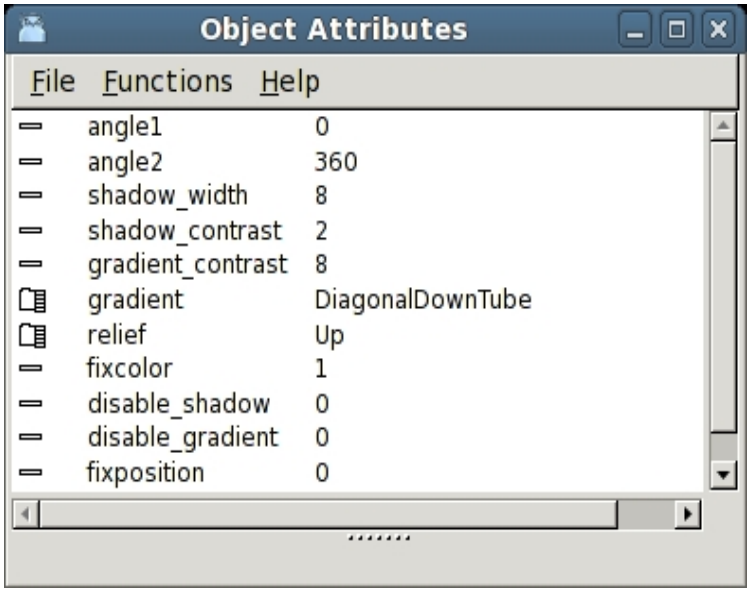


Fig Fixcolor is set on the frame

To be able to set dynamics we group the two circles. When we open the object editor for the group the attribute DynType is available and we select DigLowColor. We set DigLowColor.Color to a dark gray tone and connect DigLowColor.Attribute to a Dv, H1-Dv1.

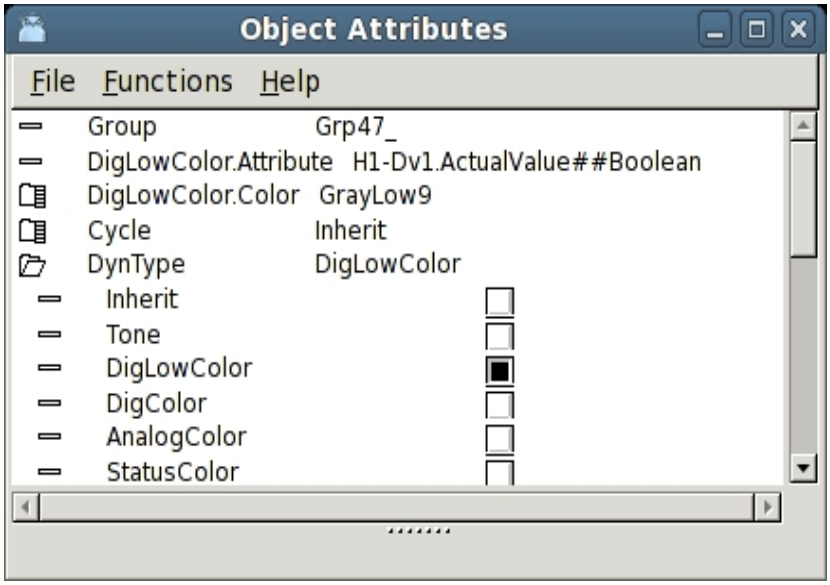


Fig Dynamic DigLowColor

The result is shown in the figure below. When the signal is high, the indicator is orange, and when the signal is low, the indicator is dark gray.

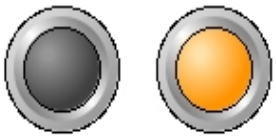


Fig Indicator with low signal to the left and high to the right

The dynamic type DigColor is similar to DigLowColor, and it could also be used in this example. The difference is that the function of the signal is inverted. In DigLowColor you specify the color the object will get when the signal is low, while in DigColor you specify the color the object will get when the signal is high.

Shift between several colors

We will now see how you can shift between several colors. We start with the indicator in the example above and add the dynamic DigColor to DynType. In the object editor we now have the attributes DigColor.Attribute and DigColor.Color. DigColor.Attribute is connected to a signal in the database, H1-Dv2, and in DigColor.Color the color for high signal is stated. We insert a blue color.

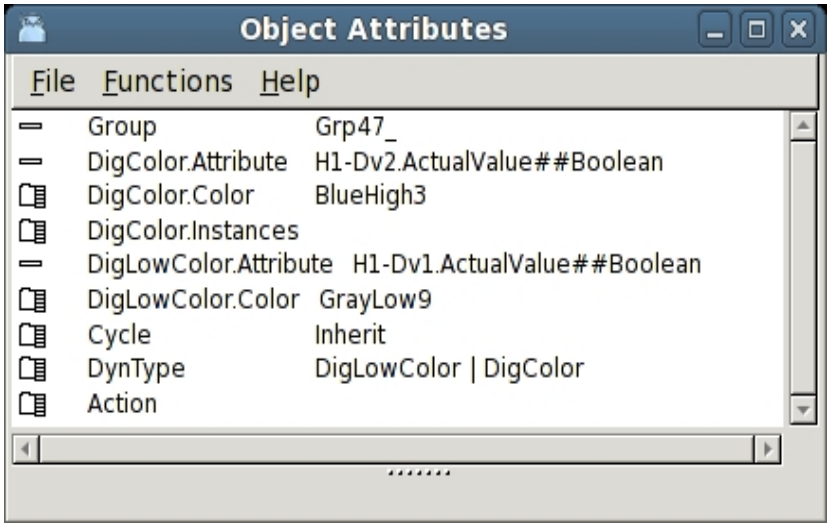


Fig A third color added with DigColor

The indicator will now shift between three different colors. DigLowColor shifts as before between dark gray and orange dependent on H1-Dv1, and DigColor sets blue color when H1-Dv2 is hign. Note that DigColor has higher priority than DigLowColor. You can see this in the object editor where the dynamics are ordered in priority order with highest priority at the top and lowest priority at the bottom. The higher priority for DigColor implies that when H1-Dv2 is high, the indicator is blue regardless of the value of H1-Dv1.

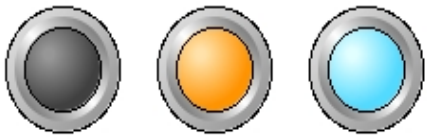


Fig The indicator is orange when H1-Dv1 is high and blue when H1-Dv2 is high

We will now add yet another color and study the Instance function. Some dynamic and action types can occur in several copies, or several instances on the same object. For DigColor this means that when we add yet another instance, you can set yet another color on the object. As there are 32 instances available you can set 32 different colors, where each color is connected to a signal in the database.

You add a DigColor instance by opening DigColor.Instance and mark instance 2. Now the attributes for this instance are displayed, DigColor2.Attribute and DigColor2.Color. We Connect DigColor2.Attribute to the signal H1-Dv3 and state a green color in DigColor2.Color. Note that instances with higher instance number has higher prioriy, ie H1-Dv3 will color the indicator green regardless of the value of H1-Dv1 and H1-Dv2.

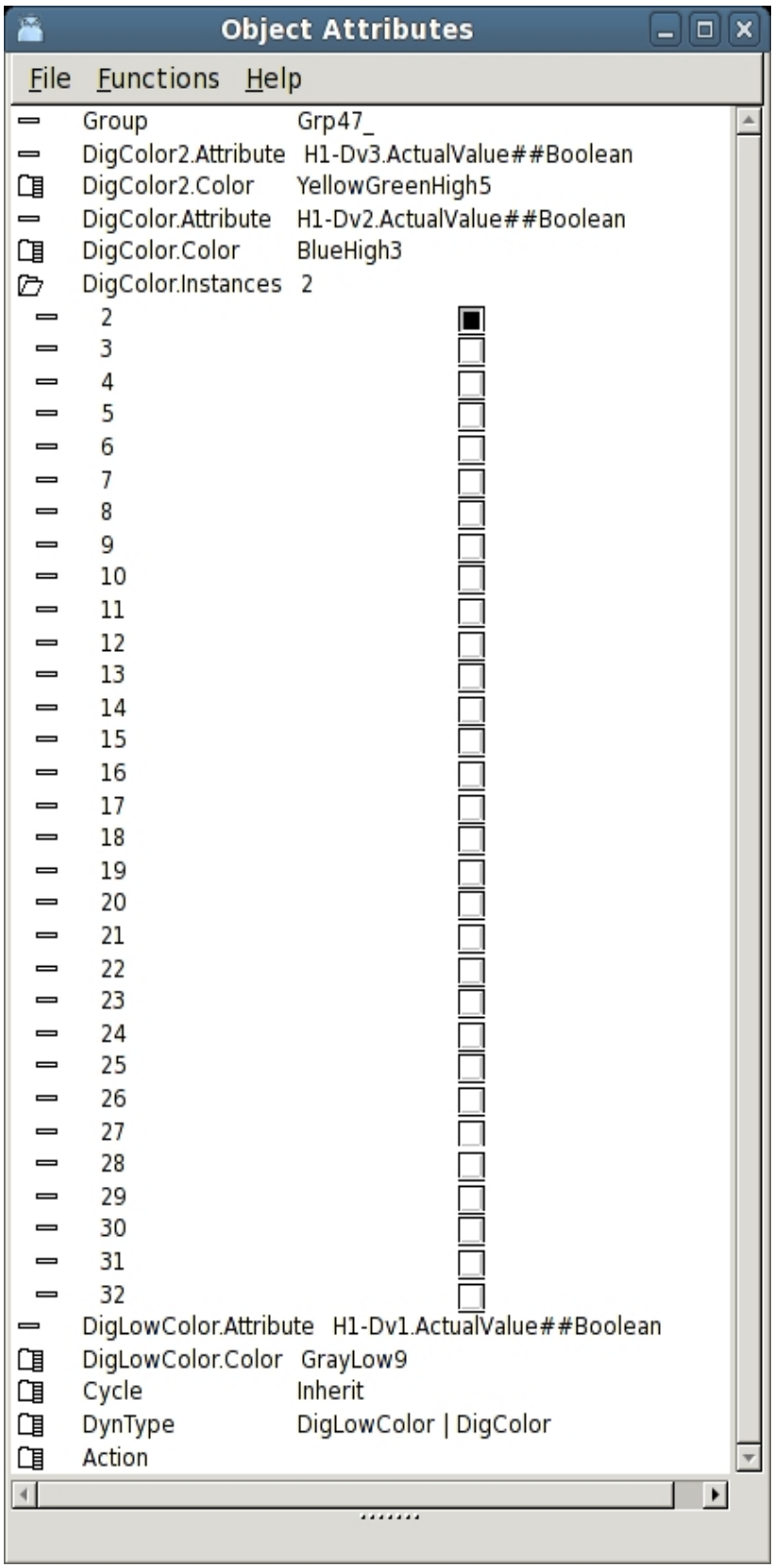


Fig A fourth color is added by DigColor instance 2

The result is displayed in the figure below. You can now shift between four different colors. Beyond the three previous, dark gray, orange and blue, the indicator is now colored green when H1-Dv3 is high.

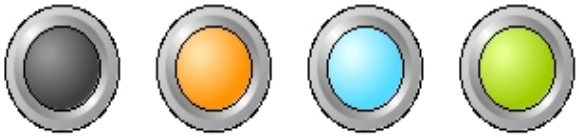


Fig The indicator with four different colors

Colors for warning and error

In ProviewR the colors yellow and red are consistently used for warning and error respectively. There are two types of dynamic, DigWarning and DigError, that sets yellow and red color. The advantage compared to DigLowColor or DigColor is that you don't have to specify the color. You just have to connect the dynamic to a digital signal in the database.

We use the indicator in the example above, and color it dark gray. This will be the color when neither of the signals for DigWarning or DigError is high. We connect DigWarning.Attribute to H1-Dv4 and DigError.Attribute to H1-Dv5.

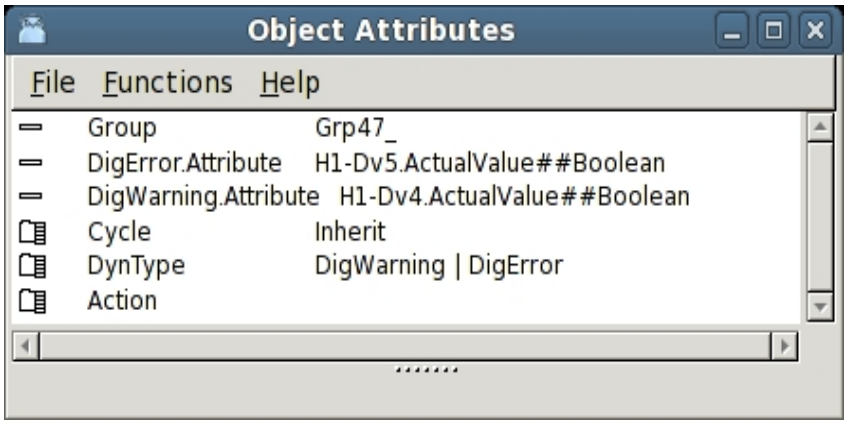


Fig Dynamic DigWarning and DigError

When no signal is set the indicator has the original color dark gray. When H1-Dv4 is set it is colored yellow, and when H1-Dv5 is set it is colored red. DigError has higher priority than DigWarning, thus it is red when H1-Dv5 is set, regardless of the value of H1-Dv4.

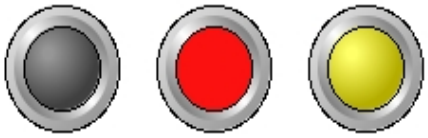


Fig Yellow color for warning and red for error

Flashing dynamic

To get maximum attention to an object you can set DigFlash dynamic on it. DigFlash means that the object will flash between two colors, when a signal is high. The flash rate equals the scan time for the object. If Cycle is Slow, the flashing will be done with ScanTime for the graph, and if Cycle is Fast with FastScanTime.

We set DigFlash dynamic to the indicator and can now state two colors that the object will flash with, DigFlash.Color and DigFlash.Color2. We set DigFlash.Color to red and DigFlash.Color2 to black, and connect DigFlash.Attribute to the signal H1-Dv6.

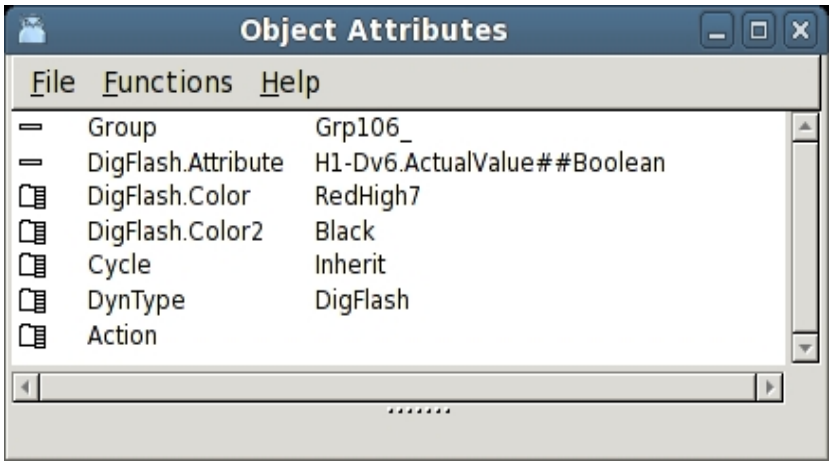


Fig DigFlash dynamic

When the signal H1-Dv6 is low the indicator has the original color dark gray. When H1-Dv6 is high it starts to flash with the colors red and black.

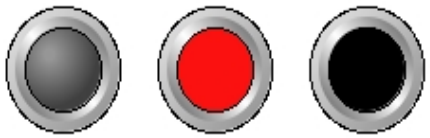


Fig The indicator flashes with red and black when the signal is high

Change the color of an analog signal

Now we will look at how to change the color of an object dependent on the value of an analog signal. There are to different types of dynamic, FillColor and AnalogColor, that are connected to analog signals. FillColor will color a portion of an object and to what extent depends on the signal value. Analog color shifts the color of an object when the signal value goes below or beyond a limit value.

Color up to a certain level

The dynamic FillColor colors an object up to a certain border line dependent on an analog signal. It is similar to the bar object, with the difference that it can be applied to an object of arbitrary shape. We will apply FillColor to a tank object fetched from Hydraulics/Tank2 in the subgraph menu.

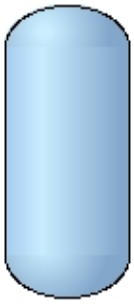


Fig A tank

The tank doesn't have any default dynamic. We mark FillColor in DynType and connect it to the Av object H4-Av1. We also set a dark gray color in FillLevel.BackgroundColor.

FillLevel.MinValue and FillLevel.MaxValue specifies the range of the signal. The default range is 0 - 100, implying that when the signal value is zero, the tank will be colored with the background color, and when the value is 100, is is colored with the original blue color we have set in the editor.

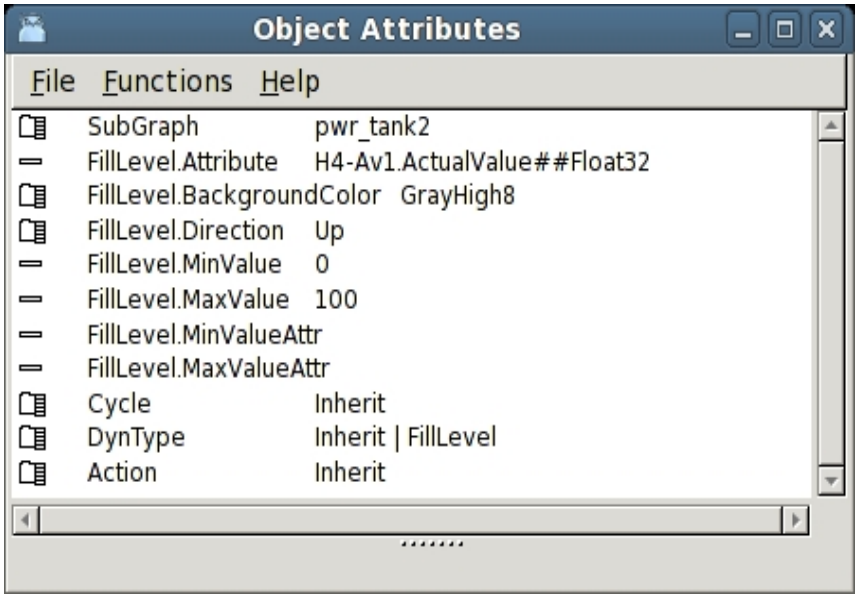


Fig Tank with FillColor dynamic

In the figure below H4-Av1 has the value 70 and the tank is filled to this level.

Fig The tank filled to a certain level

The tank can be filled in other directions by modifying FillColor.Direction. If the FillColor dynamic is used in an object graph, also the range can be connected to attributes for min and max values in the database.

Shift color beyond or below a certain limit

Often you want to display that an analog signal has gone beyond or below a certain value. Suppose that in the tank example above, the level normally should be below 80, and if it goes beyond 90 the situation is critical. We will use the dynamic AnalogColor to change the color of the tank when the value passes the two limit values.

First we want to color the tank yellow when the signal H4-Av1 goes beyond 80. We add AnalogColor to DynType and now we can set AnalogColor.Limit to 80. The type of limit value, AnalogColor.LimitType is already GreaterThan, so we don't need to change it. We must though specify the yellow color that will be set when the signal is greater than 80, and insert a yellow tone in AnalogColor.Color.

To be able to add yet another limit value at 90, we add instance 2 in AnalogColor.Instances. Now the attributes for AnalogColor2 appears and we set AnalogColor2.Limit to 90 and AnalogColor2.Color to a red color. Note that you don't connect a new signal to instance 2, but all instances uses the same signal.

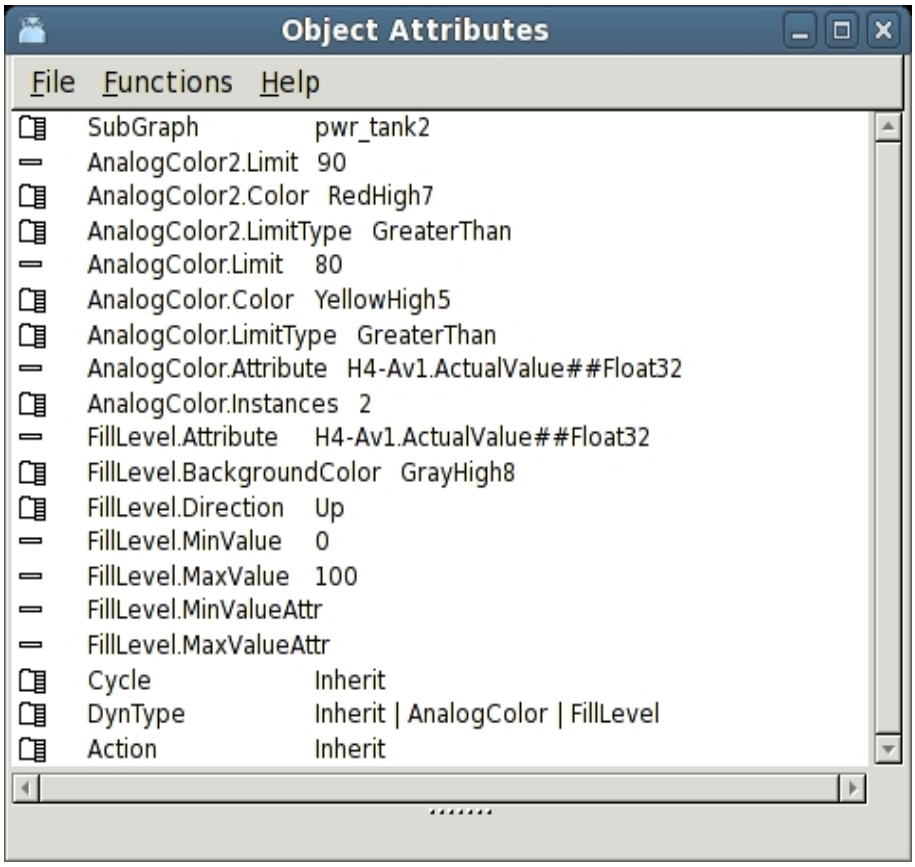


Fig AnalogColor dynamic

The result is viewed in the figure below. As long as the value of H4-Dv1 is less than 80 the tank is blue. When the value is over 80, it's yellow, and when 90 is passed it's red.

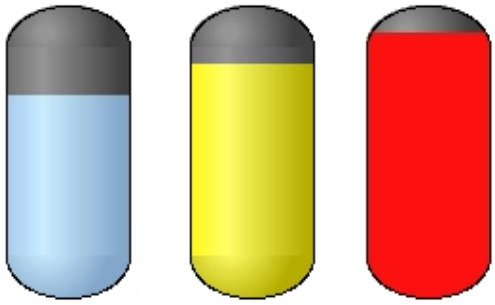


Fig At 80 the color is shifted to yellow, and at 90 to red

By setting LimitType to LessThan you can also add limit value for minimum levels, for example color the tank yellow if the value is less than 20, and red if the value is less than 10.

4.5.2 Making objects invisible or insensitive

To make objects invisible or dimmed you use the dynamic Invisible. To demonstrate this we create a pushbutton from Button/Button2MetalFrame in the subgraph menu.

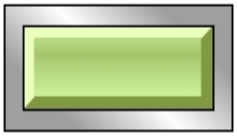


Fig A pushbutton

Invisible

The pushbutton has as default the attributes for DigToggle, and by adding Invisible to DynType we also see the attributes for Invisible. By connecting a digital signal to Invisible.Attribute the pushbutton will be invisible when the signal is high.

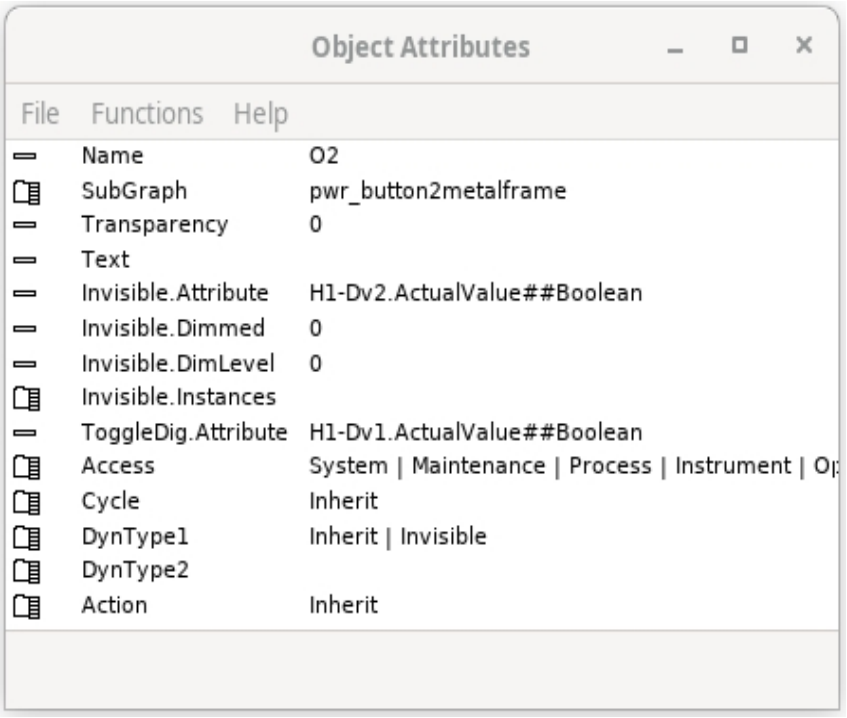


Fig Invisible attributes

We connect Invisible.Attribute to H1-Dv2, and the result is that when H1-Dv2 is high the pushbutton is invisible.

Dimmed

If you only want the pushbutton to be insensitive for click and dimmed, you set Invisible.Dimmed to 1.

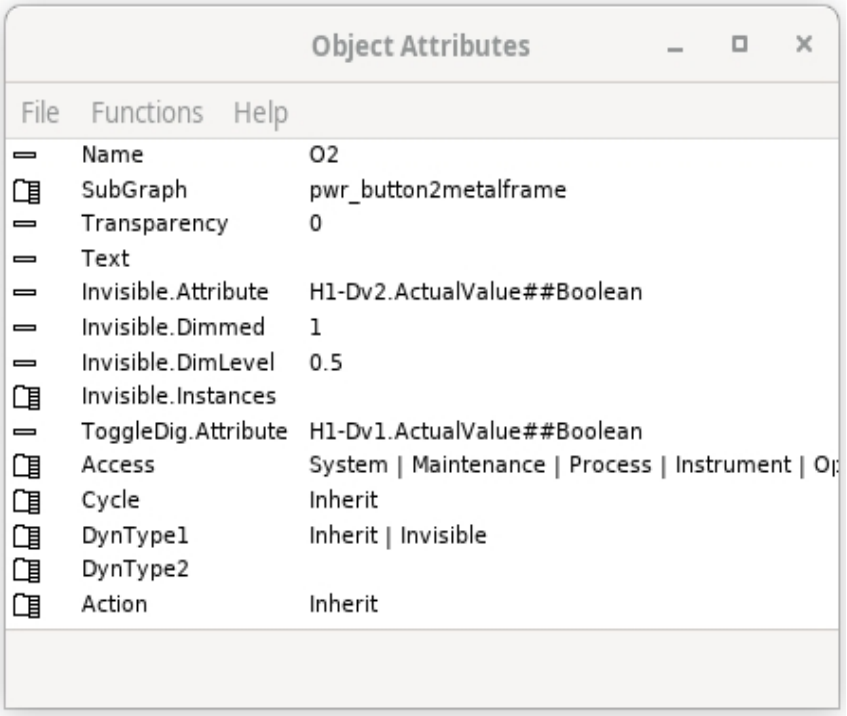


Fig Invisible.Dimmed and Invisible.DimLevel configured

When H1-Dv2 goes high the pushbutton is no longer sensitive for mouse clicks. This is marked by the drawing the button dimmed. In addition it's possible to set a transparency level on the dimmed button by setting Invisible.DimLevel to a value between 1 and 0.

Fig The pushbutton in original and dimmed state, and with DimLevel 0.5 to the right

4.5.3 Draw texts

To draw a dynamic text, you have to use a subgraph that contains an Annotation. For example almost all pushbuttons contains an Annotation to be able to print a text on the button. Also Value and InputValue subgraphs contains an annotation.

An annotation is a place in the subgraph where a text can be printed. The text either be specified in the editor, or be supplied by some different types of dynamics that is connected to signal in the database and will modify the text in runtime.

The different types of text dynamics are
- DigText, where you shift between two different texts dependent on a digital signal.
- AnalogText, where you shift between several different texts dependent on an analog signal.
- Value. When connected to a string attribute in the database, the string value will be displayed.

Shift between two texts on a digital signal

The dynamic DigLowText shifts between two texts dependent on a digital signal.
We create a pushbutton of type ButtonRoundMetalFram and opens the object editor.

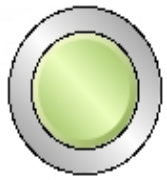


Fig Pushbutton with an annotation

In Text you can specify a text that is diplayed on the pushbutton, and we insert the text 'Off'. By adding the dynamic DigText, the text specified in Text will be replaced by the text in DigText.LowText when the signal we connect to DigText.Attribute is low. We insert the text 'On' in DigText.LowText and connect DigText.Attribute to the Dv H1-Dv1. We also connect the same signal to ToggleDig.Attribute to easy be able to change the value by clicking on the pushbutton.

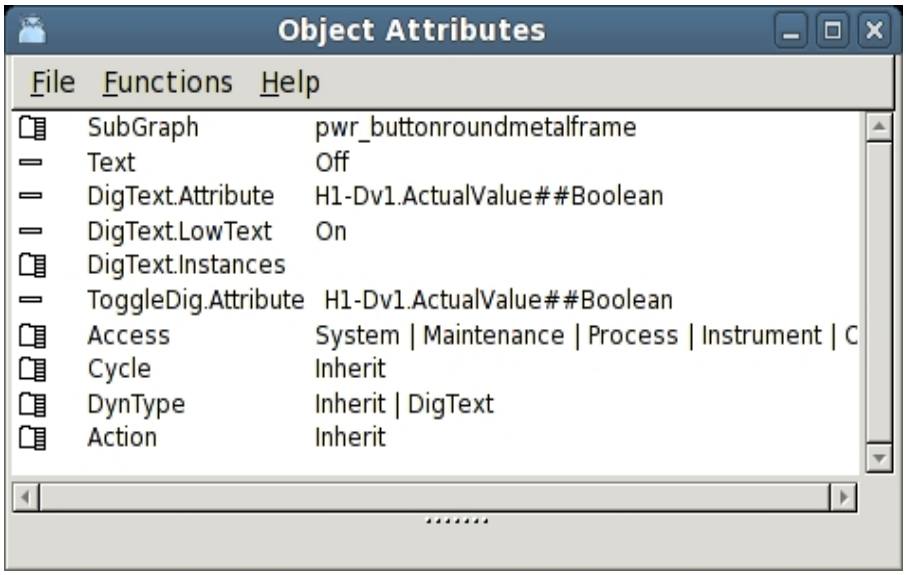


Fig Dynamik DigLowText

The dynamic will now work in this way. When H1-Dv1 is low the text 'On' is displayed on the button, this is the text in DigText.LowText, and when the signal is high the text 'Off' is displayed, this is the text we inserted in the Text attribute.

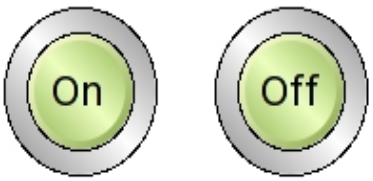


Fig Low signal to the left and high signal to the right

DigText has sevral instances and for each instance a text can be specified and it will be connected to a digital signal. Note that the function is somewhat different for the other instances. For the first instance you specify a LowText, ie a text displayed when the value is low. For the other instances you specify a HighText, ie a text that is displayed when the signal is high.

Shift between several texts dependent on an analog signal

For the dynamic AnalogColor you can specify up to 32 different texts, and which of the texts

that is printed depends on the value of analog signal that is connected to the dynamic.

We use a ValueLargeCenter to demonstrate the dynamic.

Fig A ValueLargeCenter subgraph

ValueLargeCenter has Value dynamic as default, and we remove this by removing Inherit in DynType, and mark AnalogText instead.

In AnalogText.TextMask you mark the texts that is to be used. We will shift between four different texts and mark 2, 3 and 4. The texts 'Low level, 'Normal level', 'High level' and 'Very high level' is filled in in the text attributes. We keep the enumeration values, which implies that AnalogText.Text1 will be displayed when the signal is 0, AnalogText2 when the signal is 1 etc. Actually the text will shift at the value 0.5, 1.5, and 2.5. If the signal value is larger than 3.5 the annotation will be emptied.

We also connect the Av H4-Av1 to AnalogText.Attribute.

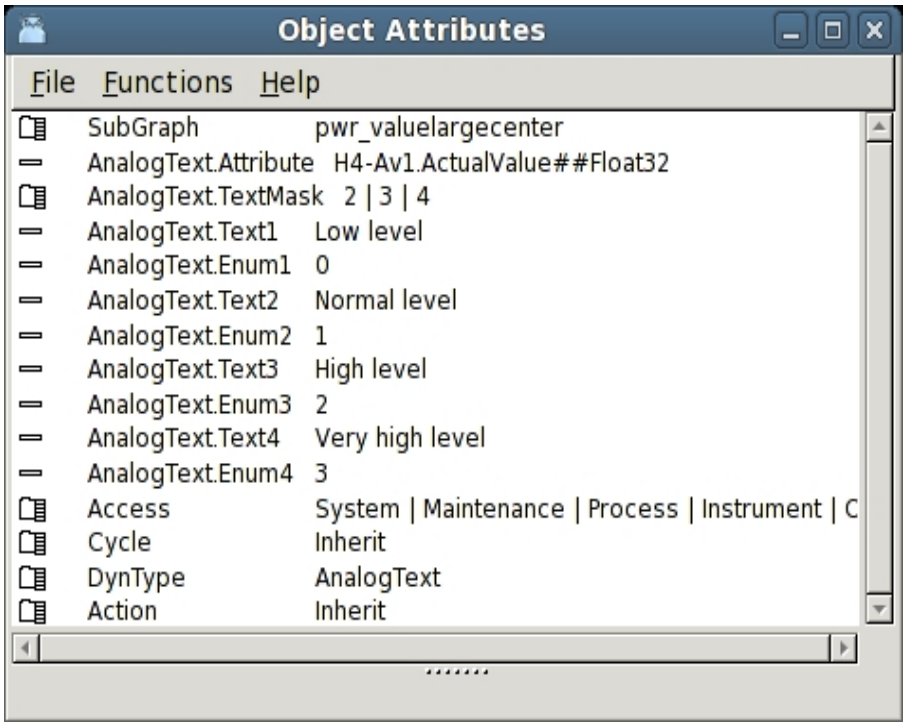


Fig The Value dynamic replaced by AnalogText

In the figure below the value object is displayed when H4-Av1 has the values 0, 1, 2 and 3 with the 0 value at the bottom.

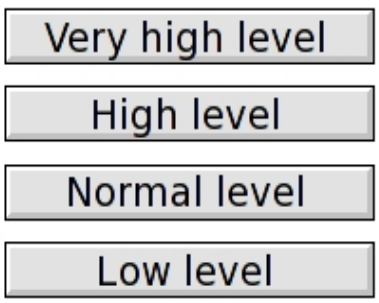


Fig Text at different values of H4-Av1

Display the text of a string attribute

To display the content of a string attribute in the database you use the Value dynamic. As in the previous example we create a ValueLargeCenter object.

Fig A ValueLargeCenter subgraph

The subgraph has Value as default dynamic, and we only have to connect Value.Attribute to a string attribute, and insert the string format '%s' in Value.Format. We connect Value.Attribute to the Sv object H1-Sv.

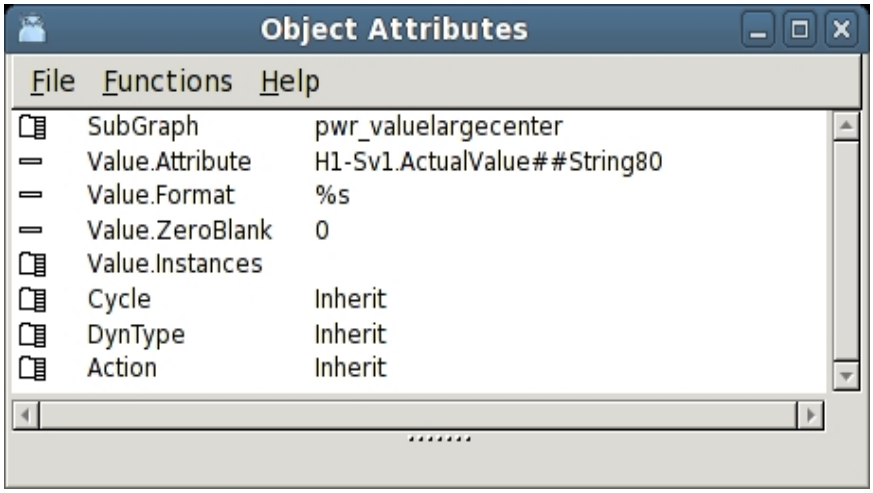


Fig Value dynamic

To show the result we insert the text 'Starting up' in H1-Sv1 in Xtt.

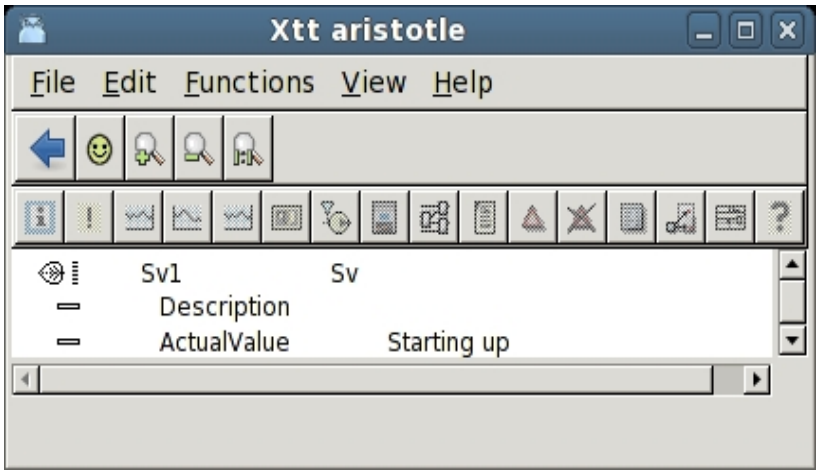


Fig Set a text in the Sv attribute

The text will be displayed in the Value object.

Fig The text displayed in the value object

4.5.4 Modify the shape

In many cases the best way to show the state of the process is to change the shape of an object. It can be a gate that is viewed open or closed, or a padlock that is locked or unlocked. This can be achieved by using subgraphs with two pages or more.

One example of an subgraph with several pages is Smiley.

Subgraphs with several pages are handled by the dynamic types DigShift, DigLowShift, AnalogShift and Animation.

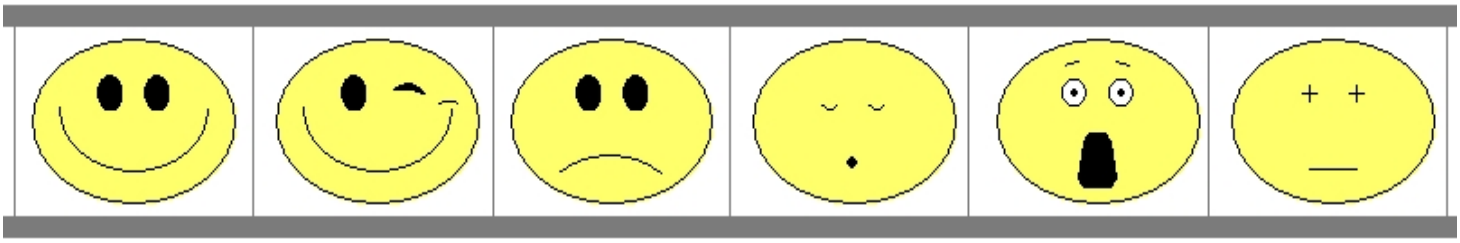


Fig Example of an subgraph with several pages

- DigShift shifts between two different pages dependent on a digital signal.
- AnalogShift shifts between several different pages dependent on an analog signal.
- Animation plays a number of pages with a certain speed so you get the impression of motion.

Shift between two pages

The dynamic DigShift can be used for a subgraph with several pages. It is connected to a digital signal and shifts between the first and the last page dependent on if the signal is high or low.

We use the Smiley subgraph to demonstrate DigShift. Smiley has the dynamic AnalogShift as default, and we first have to remove this by resetting Inherit in DynType. Instead we set DigShift in DynType, and connect the signal H1-Dv1 to DigShift.Attribute.

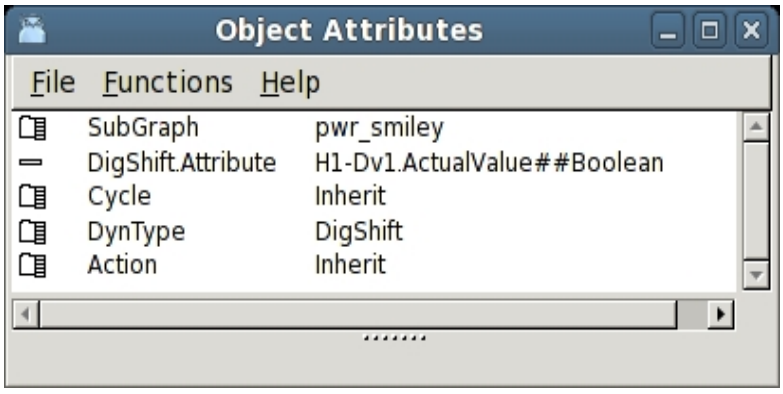


Fig Dynamic DigShift selected

The result is displayed in the figure below. When the signal is low the first page of the subgraph is viewed, and when the signal is high the last page is viewed.

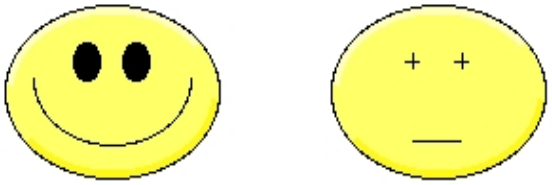


Fig Low signal to the left and high signal to the right

Shift between several pages

AnalogShift is connected to an analog signal, and can shift between several pages. The value of the analog signal will specify the index for the displayed page. When signal value is 0 the first page is displayed, when the value is 1 the second page is displayed etc.

The subgraph Smiley has AnalogShift as default. All we have to do is to connect it to an analog signal. We connect it to H4-Av1.

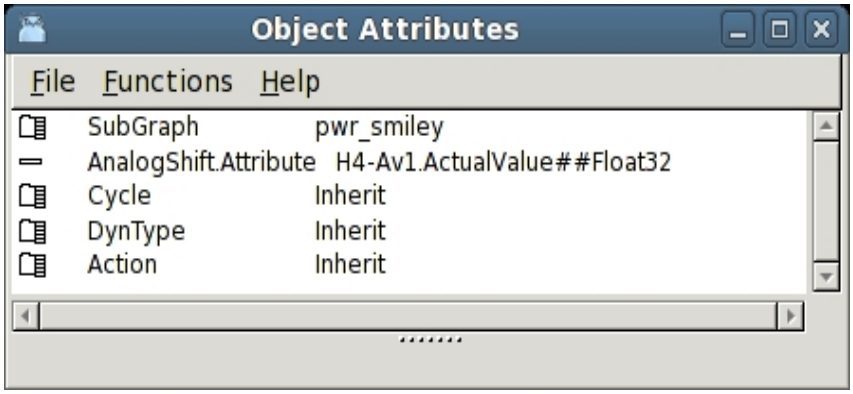


Fig Dynamic AnalogShift

Smiley contains 6 different pages. When the signal is 0 or less than 0, the first page is displayed, and when the signal is 1 the second page is diplayed etc. When the signal is 5 or greater than 5, the last page is displayed.

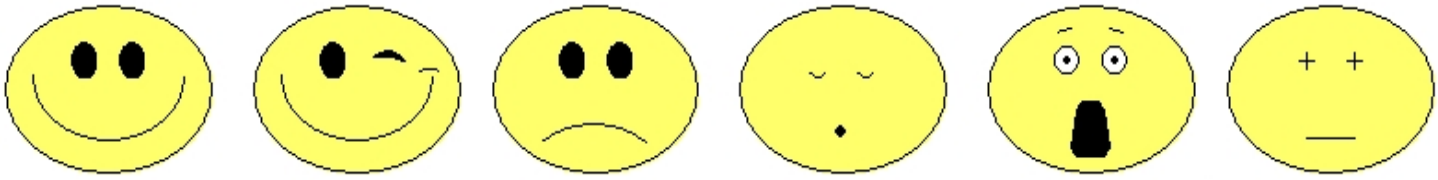


Fig The value of the signal is 0 to the left and is incremented by 1 for each picture

Animation

TODO...

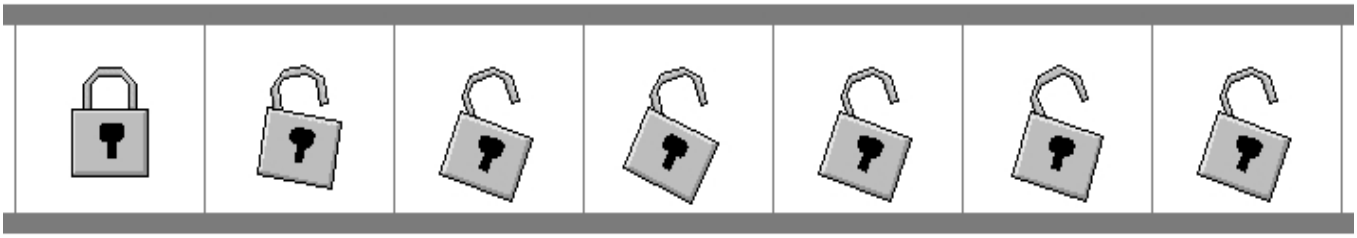


Fig Animation to open/close a padlock

4.5.5 Display analog values

Display analog value as figures

To display analog values as figures you use the dynamic Value. Value is connected to an analog signal and the value is converted to suitable format by the format specification.

The value dynamic can be used on subgraphs that contains an annotation, ie a place where a text can be printed. One example of such a subgraph is ValueLargeCenter.

Fig ValueLargeCenter

We create a ValueLargeCenter and connect it to the analog signal H4-Av1. We also have to specify the format in Value.Format. '%f' is the format for a float, and you can also specify the number of characters and decimals in the figure. The format '%7.2f' states that the format is 7 characters with the decimal point, and two of these are decimals.

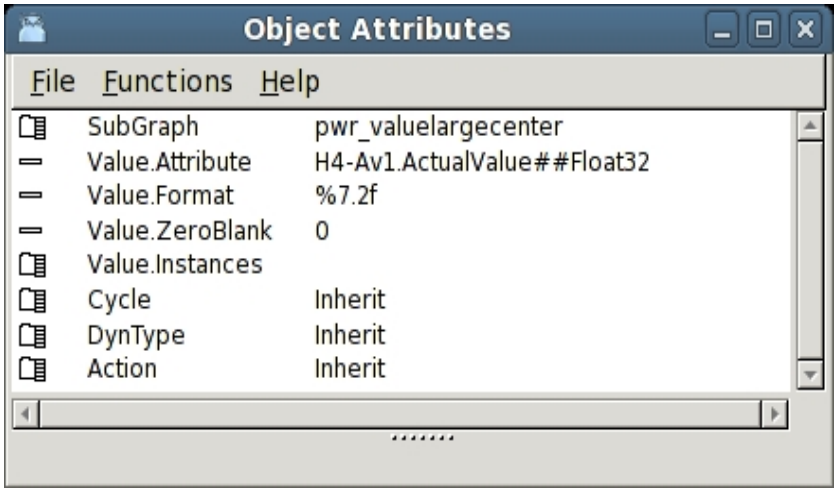


Fig Value dynamic with format specifier

The result is shown in the figure below, The analog value is written with two decimals.

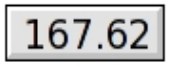


Fig Value dynamic

Note that the Value dynamic also can be used to show the content of other types of signals, eg integer, string, time etc.

By creating several instances of Value dynamic for one object, it is possible to show serveral analog value is the same object. This requires that the subgraph contain several annotations.

Analog values can also be displayed in the shape of curves and bars. This is described in the chapter 'Special objects'.

4.5.6 Move, scale and rotate objects

Move an object

To move and scale an object you use the dynamic Move. To begin with we will look at how to move an object. An object can be moved in x and y direction, and for each direction an analog signal is connected to the object, specifying the movement relative to the original position. The signal value can be transformed to a movement in Ge coordinates by one factor in each direction (Move.XFactor and Move.YFactor) and one offset in each direction (Move.XOffset and Move.YOffset).

In this example we will move a rectangle in the x direction. We draw the rectangle and create a group with only the rectangle.

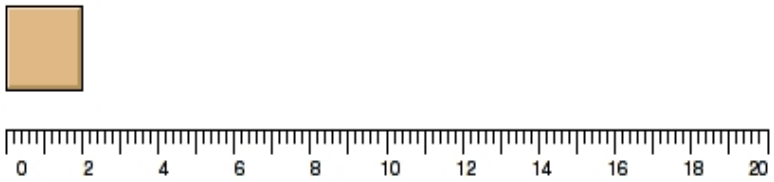


Fig A rectangle

When the rectangle is grouped, we can set Move in DynType and the attributes for Move is displayed in the object editor. As we are going to move in x direction, we connect Move.XAttribute to the signal H4-Av1 in the database. We measure the range where the rectangle is to be moved, and concludes that when the signal varies between 0-100 the rectangle should be move 20 Ge units in the graph. By placing the rectangle on the zero point, we don't have to specify any offset, but we have to scale down the signal value by 20/100 = 0.2 and this value is insert into Move.XFactor. In the example we have also drawn an x axis to show the magnitude of the movement.

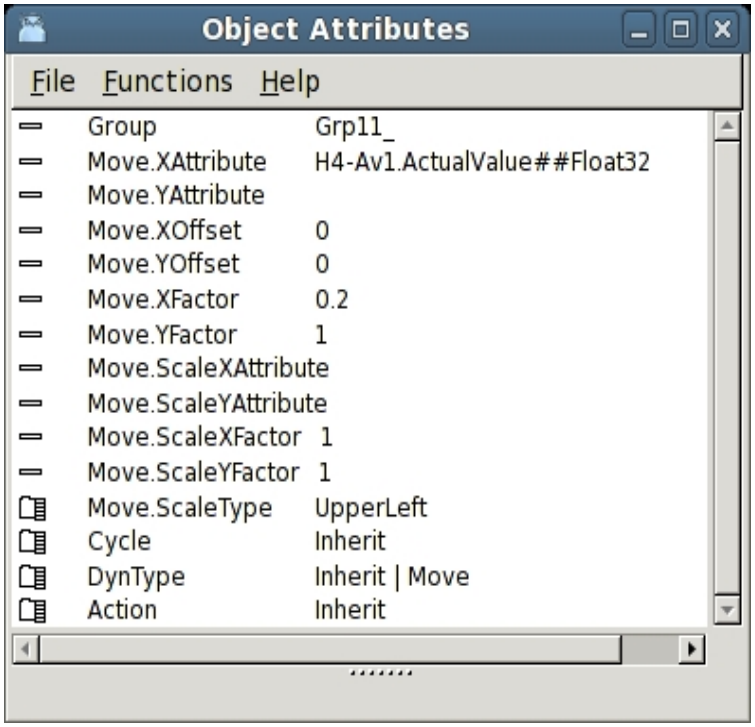


Fig Move dynamic on the rectangle

The result is that when the signal is 0, the rectangle is still positioned in the original

position. At the value 60 the rectangle is moved 12 units, se the figure below, and when the value is 100 the rectangle is moved 20 units.

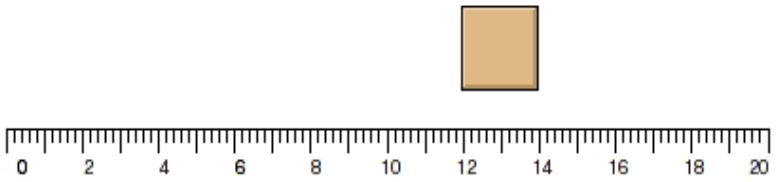


Fig The rectangle moved 12 Ge units in x direction

Scale an object

The move dynamic can also be used to scale an object. The scale can be performed in x and y direction, and for each direction you connect an analog signal specifying the magnitude of the scaling.

We use the rectangle in the example above, and add scale attributes to the Move dynamic. Move.ScaleXAttribute is connected to the signal H4-Av2. We also have to calculate a scale factor. The rectangle has the width 2 units and with a scale factor of 0.1 a signal value of 100 will result in a width of 20 units (0.1 * 100 * 2 units).

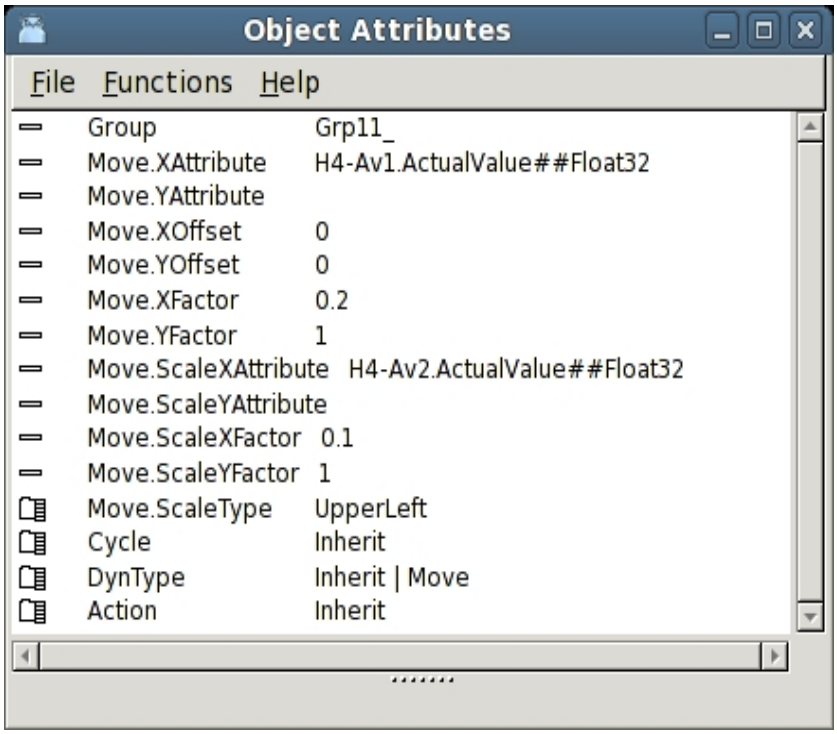


Fig Signal and scale factor stated

The result is that when the signal value is 0, the rectangle has no width at all, and is drawn as a line. At the signal value 25 is looks as in the figure below, the rectangle is scaled 2.5 times, and when the signal value is 100 the rectangle is scaled 10 times and covers the whole x axis.

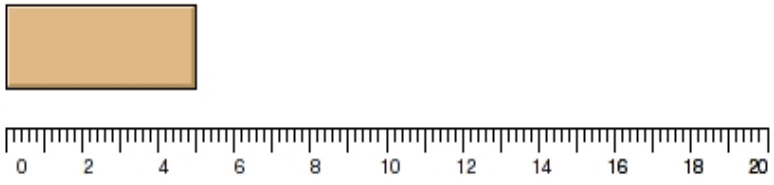


Fig The signal value 25 results in 2.5 times scaling

Rotate an object

To rotate an object you use the dynamic Rotate. Rotate is connected to an analog signal that contains the rotation in degrees.

Rotate has certain limitations as rectangles and ellipses only can be rotated in steps of 90 degrees. For that reason you preferably use polylines, lines an circles with Rotate.

We draw a simpel needle with an arc and a polyline. We group the objects and set Rotate in DynType. We also draw a circular axis with an AxisArc object.

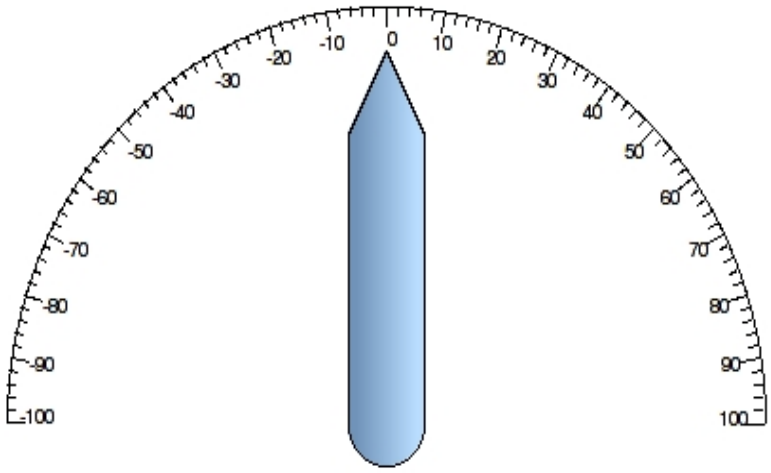


Fig Needle with rotation dynamic

The needle is connected to the anlog signal H4-Av1. We also have to specify the point on which the object will turn. In this case it is the center of the arc that has the x coordinate 9 and y coordinate 1. The turning point is inserted into Rotate.x0 and Rotate.y0.

The axis we have drawn has the range -100 - 100, which is the range of the signal H4-Av1. This has to be converted to degrees and the corresponding rotation is -90 - 90 degrees, ie the scale factor is 0.9. We insert 0.9 into Rotate.Factor.

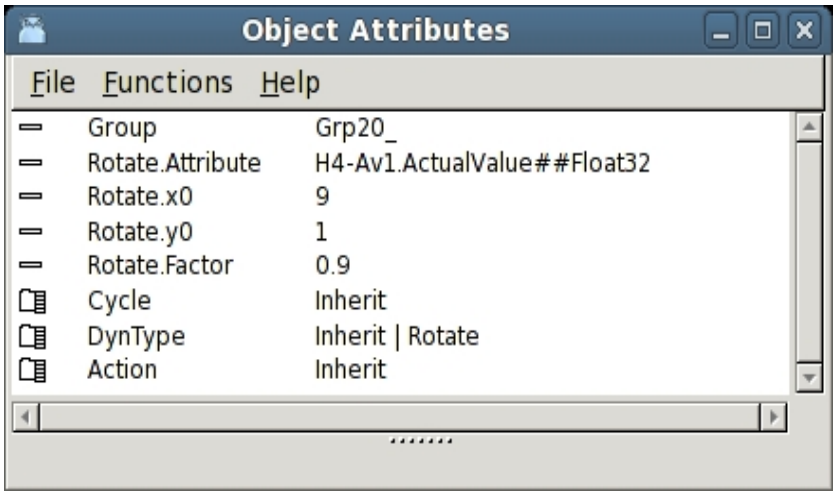


Fig The needle is connected to an analog signal and the scale factor is set to 0.9

The result is displayed in the figure below.

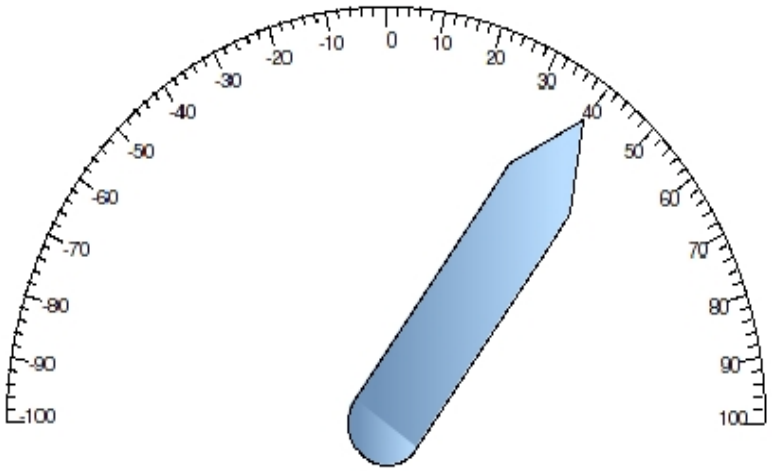


Fig The value 40 will cause a rotation of 36 degrees

4.5.7 Sounds and commands

TODO...

4.5.8 Set value with push buttons

Set a digital value

The Action SetDig sets the value of a digital signal to 1 when you click on an object.

To demonstrate this we create a button from a rectangle. By setting 3D and increase shadow_width to 15 the rectangle will get a characteristic button look. Furthermore we add the gradient DiagonalLowerRight with gradient_contrast 1 to increase the 3D effect. Finally se create a group of the rectangle and are now able to specify an Action.

Fig A rectangle shaped as pushbutton

To set set a digital signal to 1 when pressing the button, we set SetDig in Action.
This means that is the signal value is 0, the value will be set to 1. If the value already is 1 there will be no change.

We connect SetDig.Attribute to the signal H1-Dv1.

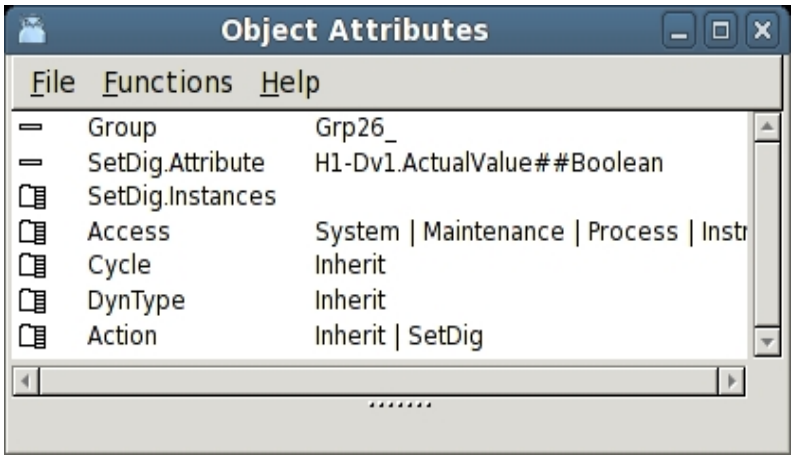


Fig Action SetDig on the button

The result is that when we click the button H1-Dv1 is set to 1.

It is possible to add several instances of SetDig and for each new instance you connect a new signal that is set to 1 when the button is clicked on. In this way several signals can be set with one mouseclick.

Reset a digital value

ResetDig works as SetDig with the difference that the signal value is set to 0.

We set Action ResetDig and connect to the signal H1-Dv1.

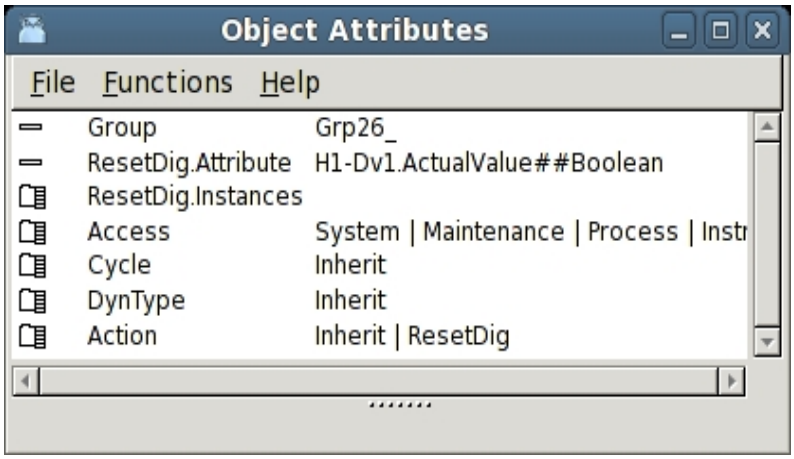


Fig Action ResetDig on the button

The result is that when we click the button H1-Dv1 is set to 0.

Toggle a digital value

With Action ToggleDig you change the value of a digital signal when clicking on the object. If the value of the signal is 1, it is set to 0, and if the value is 0, it is set to 1.

We connect ToggleDig.Attribute to the signal H1-Dv1.

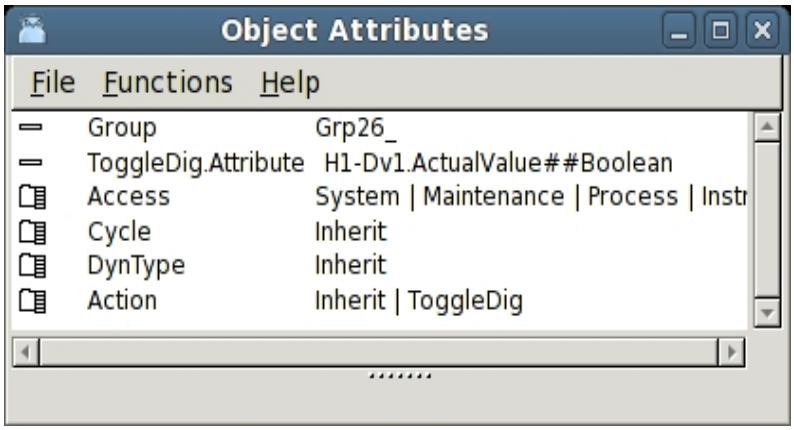


Fig Action ToggleDig on the button

The result is the the value of H1-Dv1 is inverted when the button is clicked on.

Set a digital value as long as the buttons in pressed

With Action StoDig you set the value of a digital signal to 1 when the button is pressed, and reset to 0 when the button is released.

We set Action StoDig on the button an connect to H1-Dv1.

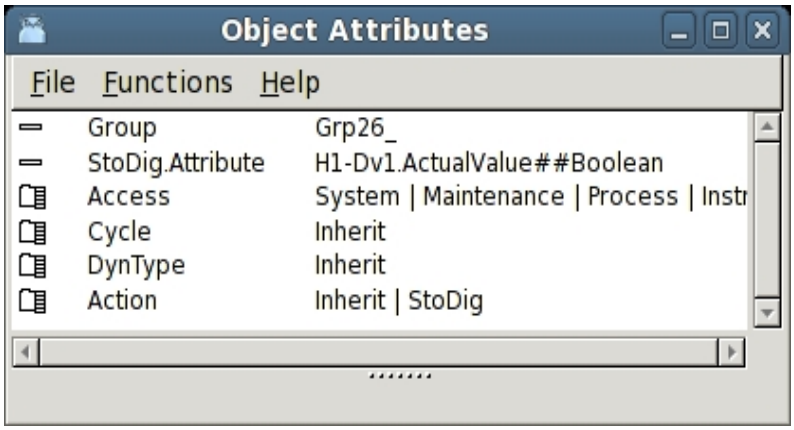


Fig Action StoDig

The result is that when we press the button, H1-Dv1 is set to 1, and when we release the button H1-Dv1 is set to 0.

Pushbutton with confirmation

If you add Action Confirm to a pushbutton, a confirmation from the user is required before the action is executed. When you click the button a popup dialog is displayed where you either can confirm or cancel. The text in the window is stated in Confirm.Text.

Set an analog value

To set a value to an analog signal the Action SetValue is used. SetValue is connected to an analog signal, and when clicking on the object a specified value is set to the signal.

We set Action SetValue on the button, and connect to the Av object H4-Av1. We want to set the value 4.5 and insert 4.5 into SetValue.Value.

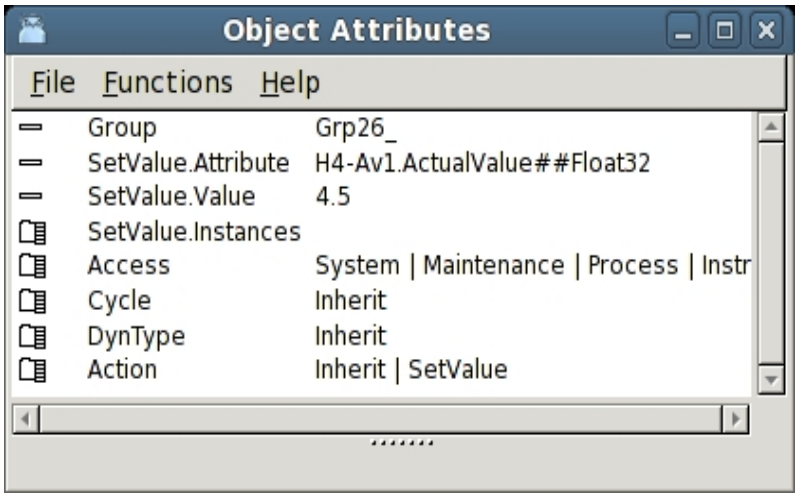


Fig SetValue

The result is that when we click the button the value 4.5 is set in H4-Av1.

Several instances of SetValue are implemented, and for each instance you specify a value and connect the instance to an analog signal. It is then possible to set different values into a number of analog signals with one mouse click.

Increase/Decrease buttons

With action IncrAnalog you can increase or decrease the value of an analog signal with mouse clicks. IncrAnalog is connected to an analog signal, and in IncrAnalog.Increment is stated the amount of the increment or decrement for each click. You can also state minimum and maximum values for the signal.

We create a ButtonUp and a ButtonDown object from the Pushbutton map in the subgraph palette.

Fig ButtonUp and ButtonDown

Both have SetDig as default. We remove SetDig by unmarking Inherit in Action, and mark IncrAnalog instead. We connect both buttons to the signal H4-Av1. The ButtonUp button should increase the value with 1 for each click, and we set IncrAnalog.Increment to 1. ButtonDown should decrease the value with 1, and we set IncrAnalog.Increment to -1. We also set min and max values and set IncrAnalog.MaxValue to 10.

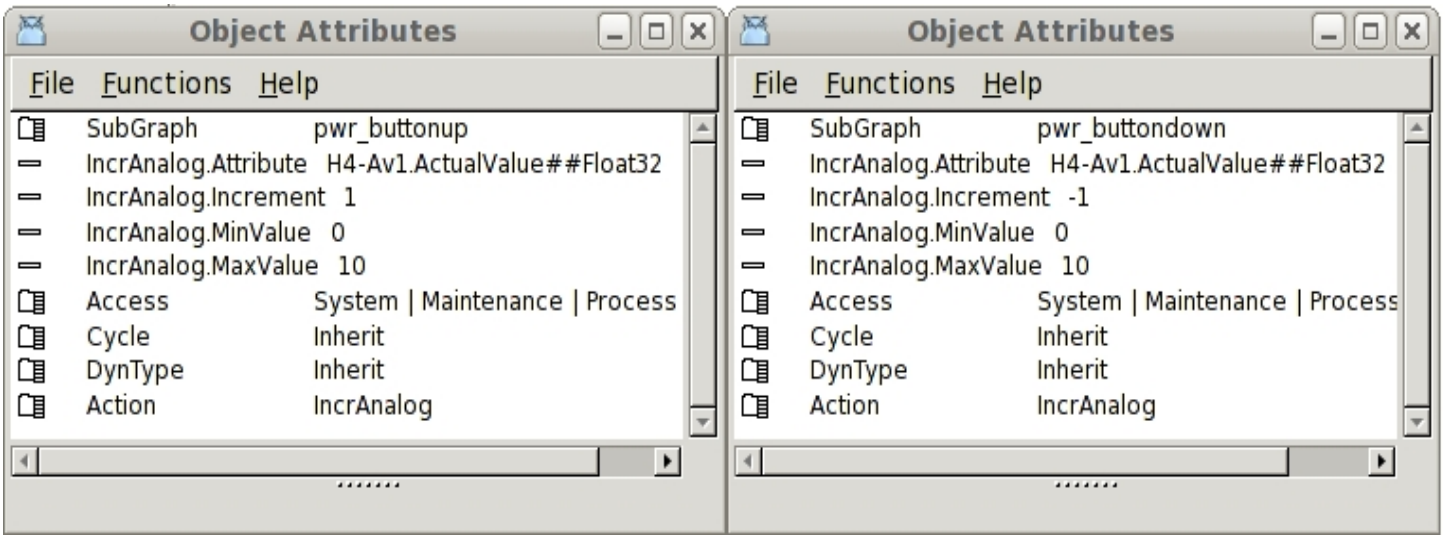


Fig ButtonUp to the left and ButtonDown to the right

The result is that when we click on ButtonUp the value of H4-Av1 is increased with 1 for each click until the max value 10 is reached. When we click on ButtonDown, the value is decreased with 1 until the min value 0 is reached.

Radio buttons

Radio buttons are used when you want to select one alternative from a number of alternatives. For each alternative you create a radio button and connect it to a digital signal. Then you group the radio buttons. The action RadioButton work in the way that the signal for a radio button that is clicked on will be set, while other radio buttons in the group will be reset. This means that at most one alternative is selected, and only one of the signals are 1.

We create four radio buttons of type RadioButtonRelief.

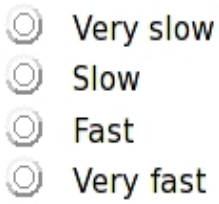


Fig Four radio buttons

They already have the action RadioButton as default. We connect the every radio button to one digital signal each, H1-Dv1, H1-Dv2, H1-Dv3 and H1-Dv4. Then we group the four radio buttons and add some suitable texts to describe the alternatives.

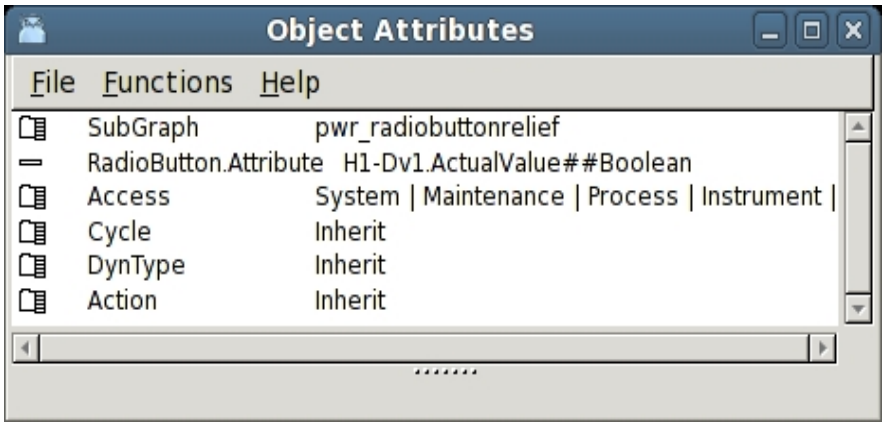


Fig The first radiobutton is connected to H1-Dv1

When we open the graph with the radio buttons in the operator environment, we can select one of the alternatives. The selected one is marked with a black dot, while the other are reseted. In the figure below the second alternative is marked, ie H1-Dv1 is 1 while Dv1, Dv3 and Dv4 is 0.

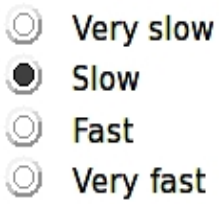


Fig A radio button is marked

Checkboxes

Checkboxes are used to mark a number of alternatives in a list. Contrary to radio buttons several alternatives can be selected concurrently. A checkbox has the action ToggleDig combined with the dynamic DigShift.

We create for checkboxes of type Checkbox2, and insert descriptive text close by.

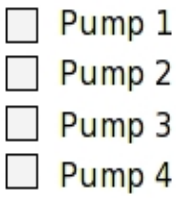


Fig Checkboxes

The checkboxes are connected to the signals H1-Dv1, H1-Dv2, H1-Dv3 and H1-Dv4. Note that both DigShift.Attribute and ToggleDig.Attribute are connected to the same signal.

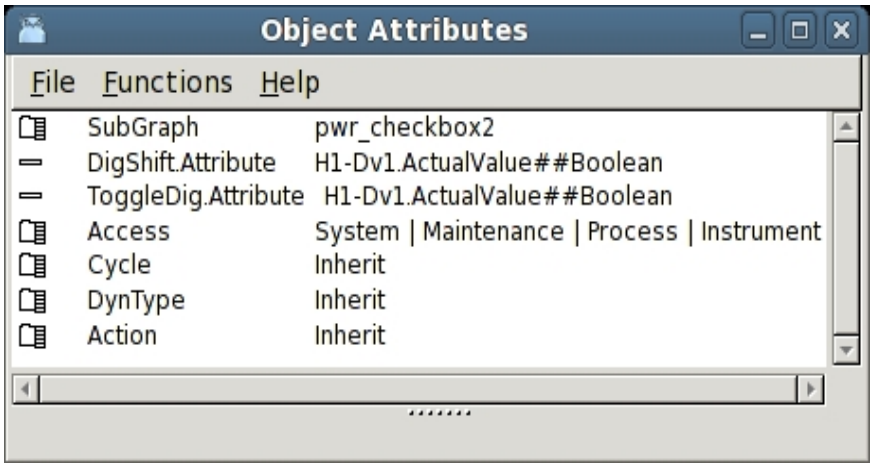


Fig Object editor for the first checkbox

In the figure below the first and third checkbox are marked, ie H1-Dv1 and H1-Dv3 are set to 1.

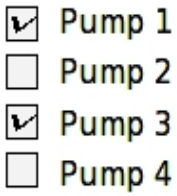


Fig Checkboxes marked

4.5.9 Popup menu

With the action PopupMenu you can by right clicking on an object, open a popup menu. The popup menu is connected to an object in the database, and displays the methods for this object.
The methods viewed depends on how the object is configured in the database, and on which methods are configured for this object type.

Let us begin with a simple example where we show the value of an Av object in a Value field.
We create a ValueLargeCenter, and connect it to H4-Av1 and insert the format %7.2f into Value.Format.

Fig ValueLargeCenter

Now we also add PopupMenu in Action, and insert the Av object into PopupMenu.ReferenceObject. Note that the references is to the whole object and not to a specific attribute.

We also have to open File/Graph Attributes in the Ge menu and set MB3Action to PopupMenu, as the default MB3 action is to close the graph.

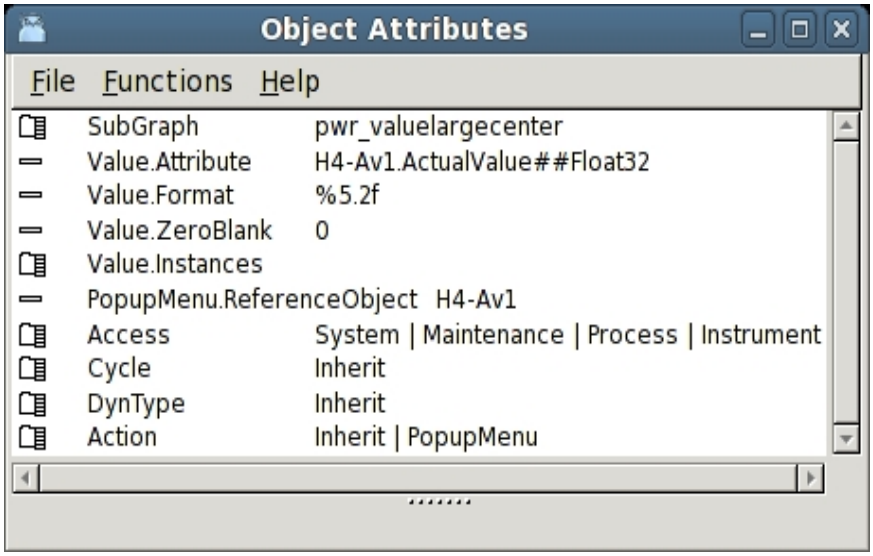


Fig Action PopupMenu added

When we now open the graph in the operator environment and right click on the Value field the popup menu is opened.

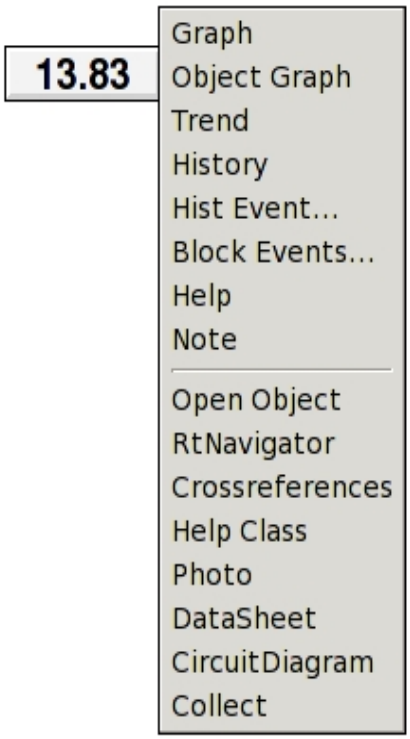


Fig The popupmenu

Which methods are present in the popup menu depends on which class the referred object belongs to and how the object is configured. In the figure below the configuration of the Av object is displayed, and we can see the attributes DefGraph, DefTrend, HelpTopic, DataSheet, CircuitDiagram, Photo and Note that all are associated with different methods.

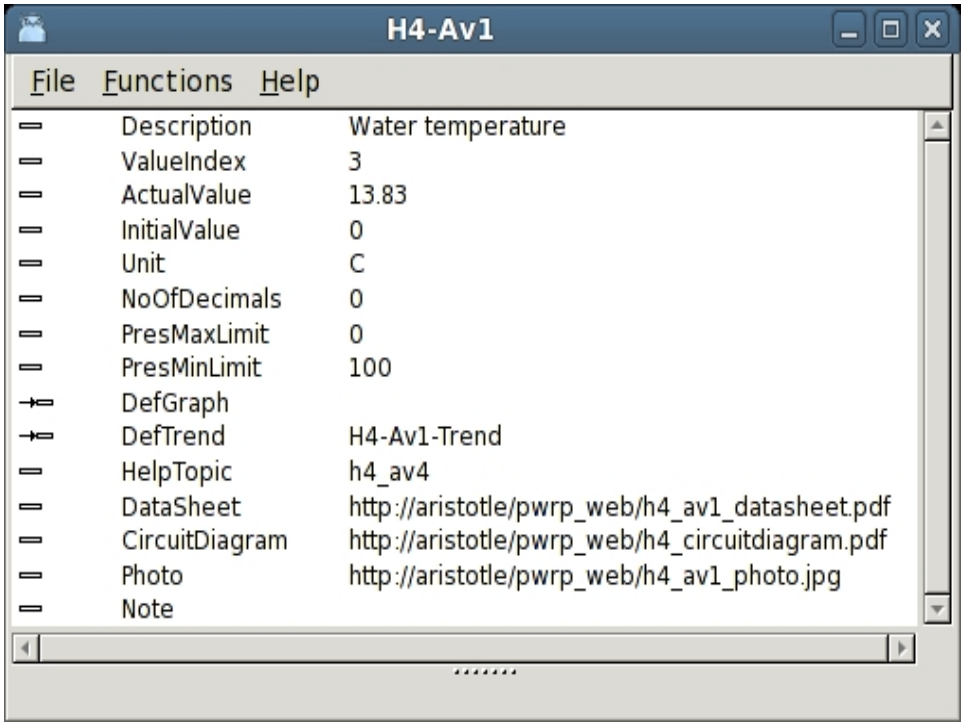


Fig The configuration of H4-Av1

We will now go over some of the alternatives in the popup menu and see how they are configured or what they depend on.

Graph
For Graph you specify a graph that shows information about the object or about the plant part where the object is found. The configuration is made by inserting an XttGraph object in the DefGraph attributes. For H4-Av1 DefGraph is not filled in, and then the graph if fetched from the closest ancestor with DefGraph specified. In this case it is the hierarchy object H4 that has a default graph stated.

Object Graph
Object Graph opens the object graph for the object. The object graph is a graph that displays information about an object of a specific class. If there is an object graph availabel for the present class this alternative is viewed in the popup menu.

Trend
Shows a trend for the object. This requires that a trend is configured with a DsTrend object, and the trend object is inserted in the DefTrend attribute. DefTrend can also refere to a DsFastCurve or PlotGroup object.

History
Opens a curve with process history for the object. This requires that storage of data is configured by a SevHist object positioned below the current object in the object tree.

Hist Event
Hist Event displays the alarm and event history for the current object.

Block Events
Block Event opens a dialog where it is possible to block alarms of different priority.

Help
Help displays a help text for the object. Help text are written in a help text file, $pwrp_cnf/xtt_help.dat, where each text embraces a topic. By inserting a topic into the attribute HelpTopic, the text for this topic will be displayed when the Help entry in the popup menu is activated.

Note
With Note you can write a text that is stored in the Note attribute of the object. The Note text is shown in the object graph of the object.

Open Object
Open Object displays the object attributes and the object content.

RtNavigator
RtNavigator searches the object in the runtime navigator.

Crossreferences
Shows a list with references to the object in plc code and graphs.

Help Class
Help Class opens a help text for the object class.

Photo
Photo displays a photo of the object in the plant. It is configured by inserting an URL to the photo into the attribute Photo.

DataSheet
If a data sheet for the object is available, you can insert an URL to the data sheet in the attribute DataSheet.

CircuitDiagram
A URL to a circuit diagram can be inserted into the CircuitDiagram attribut.

4.5.10 Open a graph

To open a graph by clicking on an object, you normally use the action OpenGraph. You can also use the Command action with the command 'open graph'.

Open a graph with a mouse click

We create a pushbutton to open a graph, with a rectangle and a text that is grouped.

Fig Pushbutton to open a graph

Action OpenGraph

In the first example we set action to OpenGraph. OpenGraph requires that you have configured an XttGraph object in the database. The XttGraph object is inserted into OpenGraph.GraphObject.

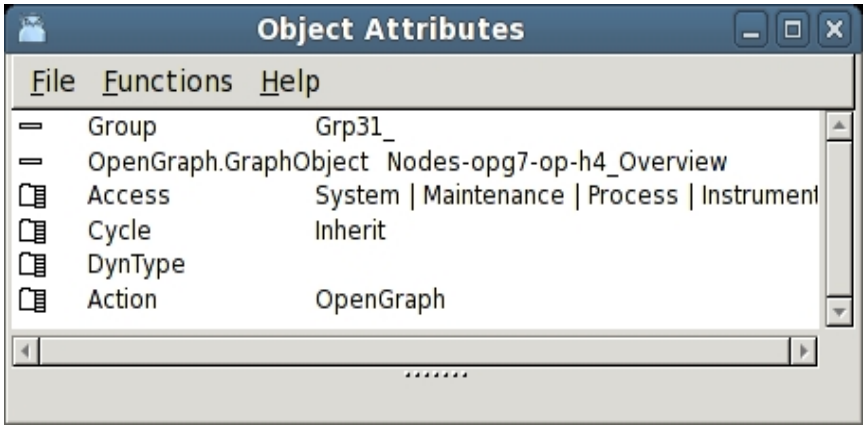


Fig Action OpenGraph

If the graph should be able to be opened on different nodes, and you prefer to use local XttGraph object, you can replace the name of the node object by '$node'.

```
$node-op-h4_Overview
```

$node will be replaced by the name of the node object for the current node. You also have to see to that XttGraph object with the corresponding name exist on all the nodes in question.

Action Command

It is also possible to open a graph with the action Command and the command

```
open graph 'graphname' [/width=] [/height=]
```

The command to open an object graph is

```
open graph /class /instance=
```

In this example we have set Command in Action and inserted a command to open the object graph for the Av object H4-Av1.

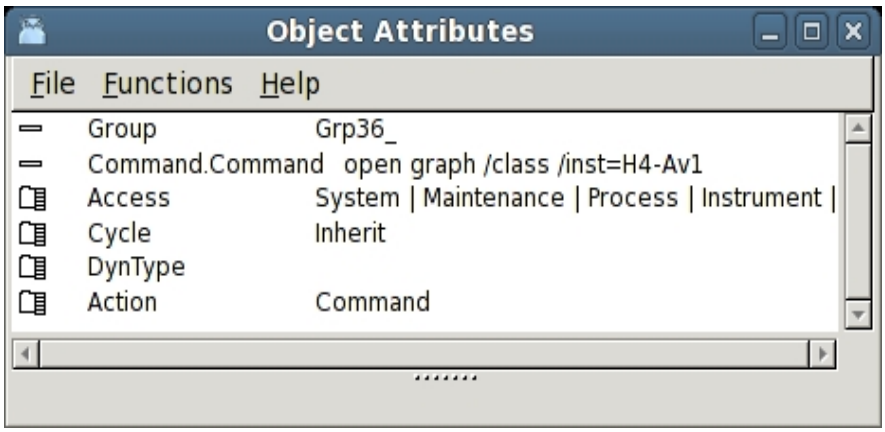


Fig Open an object graph

If the action PopupMenu is present and you in addition want to open the object graph when you click on the object, it is sufficient to add OpenGraph to action. You don't have to fill in an object, as this will be fetched from PopupMenu.ReferenceObject by default.

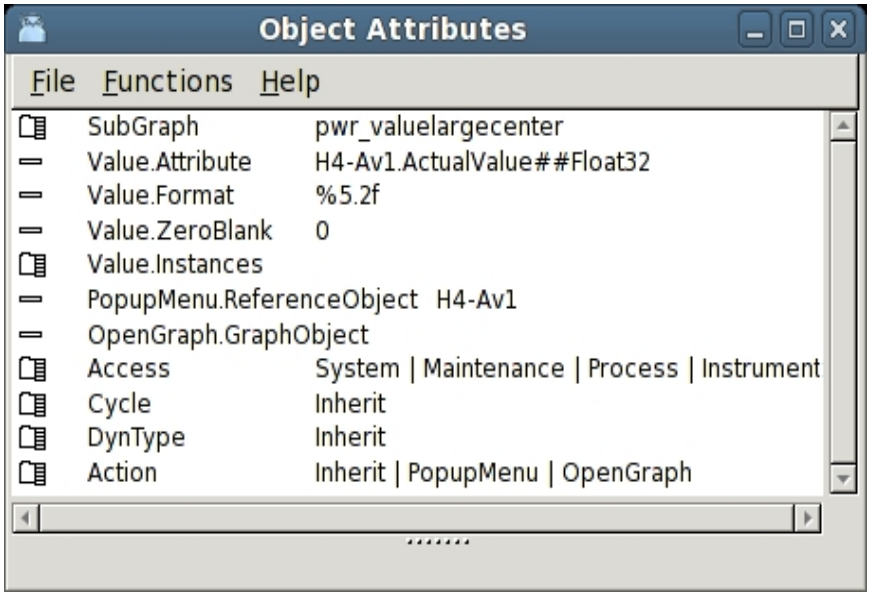


Fig Open the object graph when PopupMenu is configured

If you you want to set input focus to a specific input field in the graph when the graph is opened, you use the command 'open graph /focus'. You also have to specify the name of the input object that should receive input focus. The name can be set from Edit/Change name in the menu.

```
open graph /object=*-Graphs-SomeXttGraph /focus="TempSetValue"
```

Close a graph

To close a graph the action CloseGraph is used.

Fig Pushbutton to close a graph

We create a pushbutton with the text 'Close' and set Action to CloseGraph.

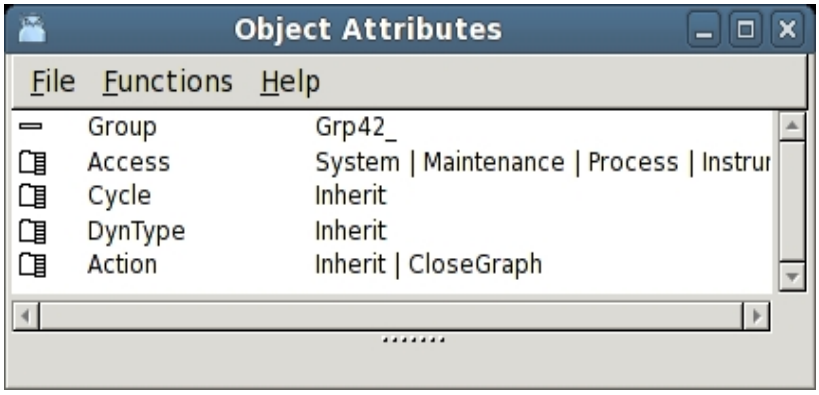


Fig Action CloseGraph

4.5.11 Execute commands

With action Command you can specify an xtt command that is executed when an object is clicked on. There are a number of xtt commands to open graph, show curves, call methods etc. The xtt commands are described in the Operator's Guide. You can also execute scripts with Command.

In this example we will open a trend curve from a push button.
We create a rectangle, add a suitable text and make a group fo the recangle and the text.

Fig Command pushbutton

In Action we mark Command and in Command.Command we insert the xtt command. We want to open a trend curve configured with the DsTrend object H4-Av1-Trend, and the command to open the curve is

```
open trend H4-Av1-Trend
```

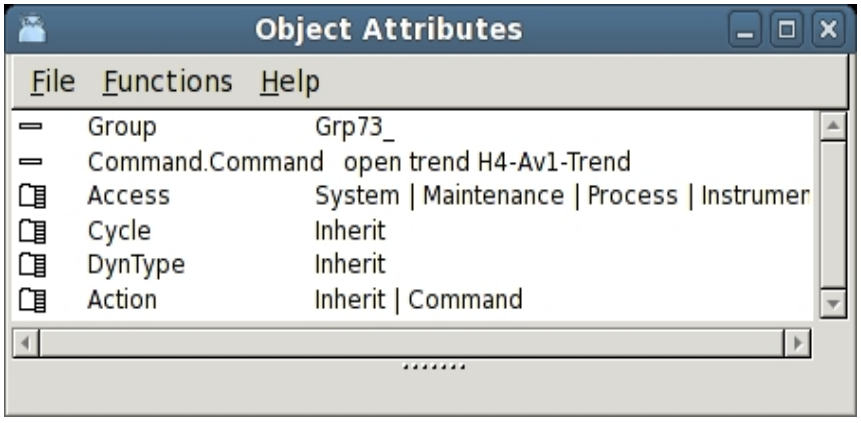


Fig Command to open a trend

Execute a script

With a script you can execute several commands at the same time, but you can also read attributes in the database, test on different conditions and set attributes.

A script is executed with an xtt command consisting of a '@' sign followed by the filename of the script, for example

```
@$pwrp_exe/my_script
```

This will execute the script my_script.rtt_com on the directory $pwrp_exe.

Below is an example of a script that opens the graph g1 if H1-Dv1 is set, else the graph g2 is opened.

```
main()
  int sts;

  if ( GetAttribute( "H1-Dv1.ActualValue", sts))
    open graph g1
  else
    open graph g2
  endif
endmain
```

4.5.12 Help and info

It is of course important that an operator quickly and easily can get help on how to handle graphs and processes. One way to solve this is to create help and info buttons in the graph. You can, for example, use the subgraphs ButtonHelp or ButtonInfo. The action type for those are Help, and you supply a topic in Help.Topic, and any bookmark in Help.Bookmark. The help function fetches help text written in the file $pwrp_cnf/xtt_help.dat, and a bookmark makes it possible to position in a specific row of the text.

4.5.13 Navigate from the keyboard

To enter data and activate objects without using the mouse, you use the action type InputFocus. If an object has input focus, key events are sent to that object. For a ValueInput object, this means that you can enter a value, for a pushbutton it means that you can activate it by pressing Return.

Focus input is traversed between the objects with the Tab and arrow keys. How the input focus is traversed, is configured with the attributes NextHorizontal, NextVertical and NextTab. Here you state the object name of the object that input focus is traversed to by arrow right, arrow down and Tab keys.

An object that has input focus is usually marked with a relief that surrounds the object.

4.6 Images

You insert a png, gif or jpg image in the following way. Copy the image file to $pwrp_pop. It is now displayed under the folder Local/Images in the subgraph palette. Select the picture and click with MB2 in the working area.

Change color

The functions to change colors for subgraphs, i.e. color tone, lightness, intensity and color shift, also applies to image objects.

Dynamics

You can not set any dynamic on the image object directly. You can though let them be a member of a group or subgraph, and state dynamics for the group or subgraph. If you dynamically want to change the color of an image object, you should use a dynamic type that changes the color tone, e.g. DigTone.

4.7 Custom colors

The standard color palette contains 300 colors, but sometimes the tone you desire isn't found there. Then you can use the custom colors. These are found below the standard palette, and are displayed by increasing the color palette window, or by scrolling down.

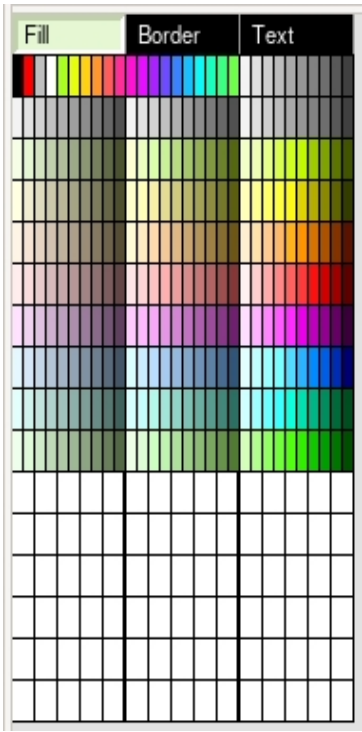


Fig Custom color palette

Create a custom color by double clicking on an empty entry in the custom color palette.

Fig Color selection

Now the Color selection dialog is opened. Either you can select one of the predefined colors, or you can press the '+' button to select a more specific color.

Fig Color selection

Select the color hue with the slider to the left, and click in color area to select brightness and saturation. The bottom slider for transparency is not implemented yet. Press the Select button to insert the color into the custom color palette. The color can now be used as fill, border or text color.

4.8 Color themes

The operator can select a color theme in the operator environment, and by drawing a graph with color theme colors, also the process grapics will follow the color themes. The color theme loads a set of color into the custom color palette, and each color is used for a specific purpose. The first color for example should be used as background color. To follow the color theme, you should also use the subgraphs under ColorTheme in the subgraph palette. These are drawn with color theme colors.

This is the way to create a simple graph with color theme colors.

Set ColorTheme in File/Graph Attributes to $default.

Select a color theme in File/ColorTheme/Select, for example Sand. Now the custom color palette is filled with the Sand color theme colors.

Fig Sand colors loaded

The first custom color should be the background color. By hovering over a color with the cursor, the pupose of the color is written below the custom color palette. Click on the color to set the fill color, and activate Functions/Set background color.

To create a button, select a button under ColorTheme, for example ColorTheme/Pushbuttons/ButtonSmallToggle. Connect and enter a text, eg 'Start' or 'Stop'.

Create indicators from ColorTheme/Indicators, eg IndSquareGreen or IndSquareRed.

Draw texts by setting the fifth color in the custom color palette, 'Text/Lines on background', as text color. Then create the text objects.

Draw the delimiter line between the push buttons and indicators by setting the fourth color, 'Delimiter line', as border color, and draw the line.

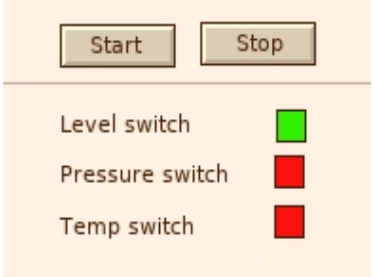


Fig Simple graph with color theme

With File/ColorTheme/Next (Ctrl+Alt+T) the colortheme is switched and we can examine how the graph look for the other themes.

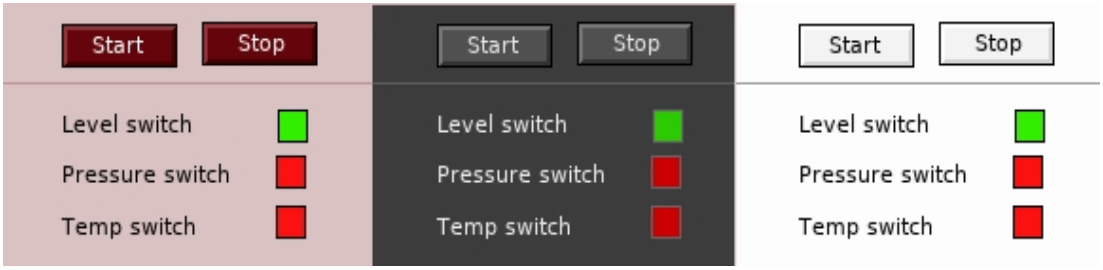


Fig Simple graph with different themes

4.9 Object tree

The object tree displays the graphical objects in the graph in a tree structure. It is opened from View/View Object Tree. Objects can be selected with click MB1 or with the Arrow up and down keys. The properties for an object is opened by clicking on the leaf or with the arrow right key. It is often a convinient way set set properties of objects, especially when there is a set of similar objects, or for objects inside a group.

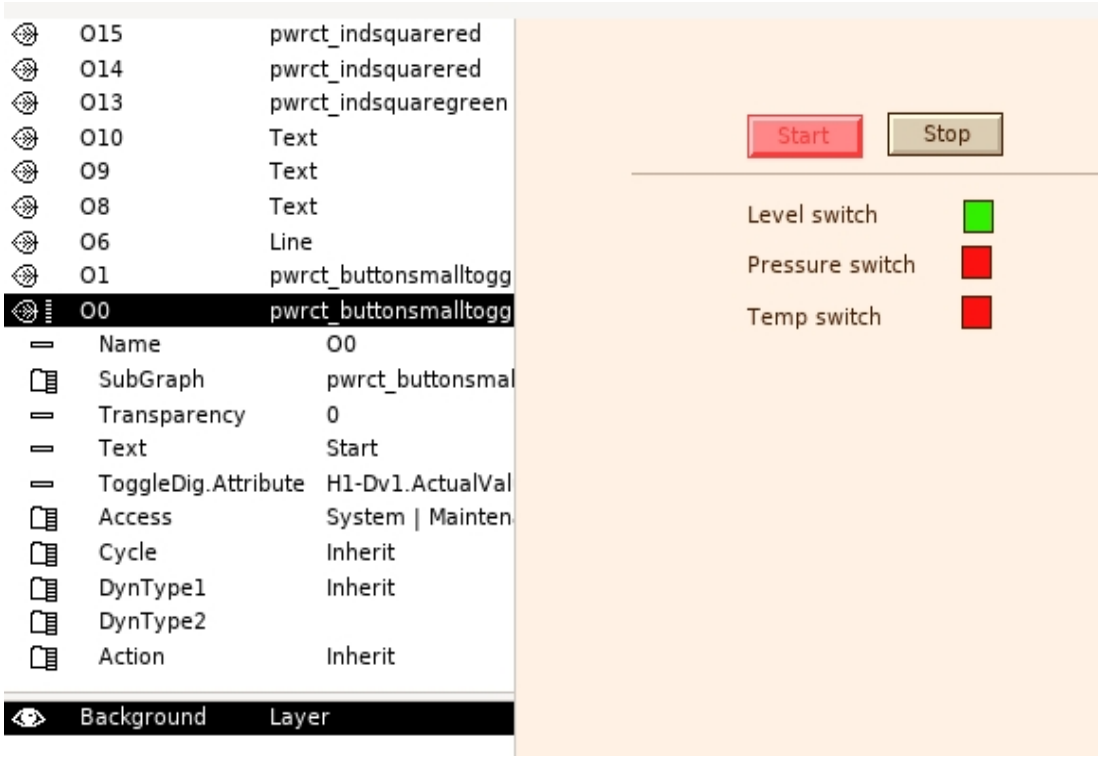


Fig Object tree

4.10 Layers

The graphical objects can be created in layers. This is a way to structure the graph and to simplify the editing. It's only possible to edit objects in the currently active layer. Thus there is no risk that objects in other layers accidently is moved or modified.

A layer is created from 'Layers/Create layer' in the menu. When a layer is created it is visible in the layer list below the object tree. When you want to create or edit objects in a layer, you make the layer active by selecting it in the layer list. By clicking on the eye symbol the layer can be set invisible or visible.

The layer is also displayed in the object tree. It will be the parent of all the objects in the layer. Also the properties for the layer can be displayed and editor from the object tree.
By clicking on the leaf or map, or with the Arrow right key, the properties for the layer are displayed. By clicking with Shift + Click MB1 on the map, the objects in the layer are displayed.

As an example we will create the simple graph above with layers, and put the buttons, texts and indicators in different layers.

We create a layer from 'Layers/Create layer' in the menu, open the properties for the layer in the object tree, and change the name to 'Buttons'. Then we select the Buttons layer in the layer list to make it active. The two buttons that we now create will be inserted into the Buttons layer.

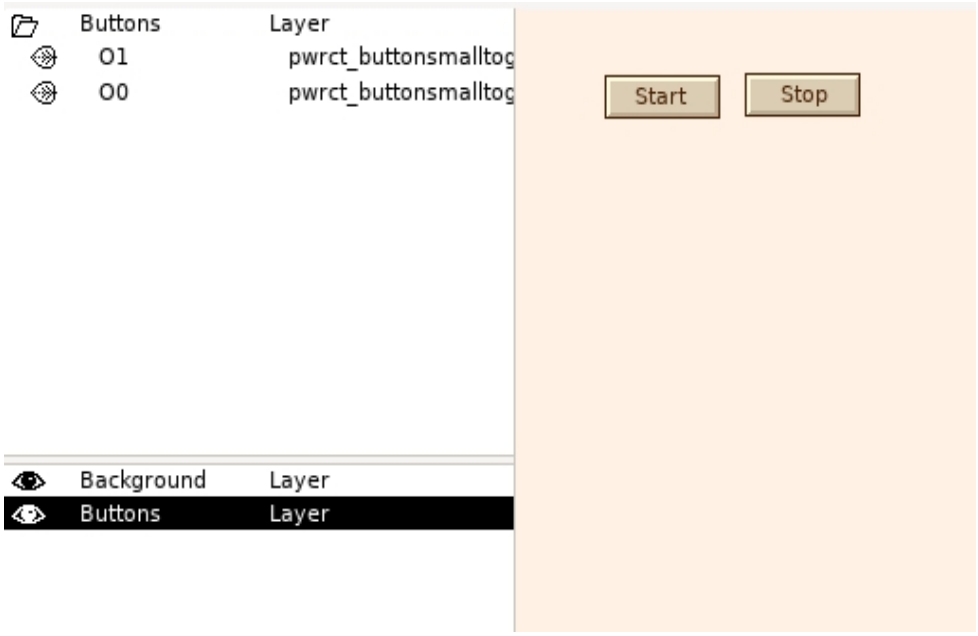


Fig Buttons layer

In the same way we create the Texts layer with the texts, and the Indicators layer with the indicators. Note how objects that doesn't belong to the active layer are insensible, and how layers can be hidden by clicking on the eye symbol in the layer list.

The delimiter line is created in the background layer that is the default layer. Objects in the background layer are displayed on the top level in the object tree.

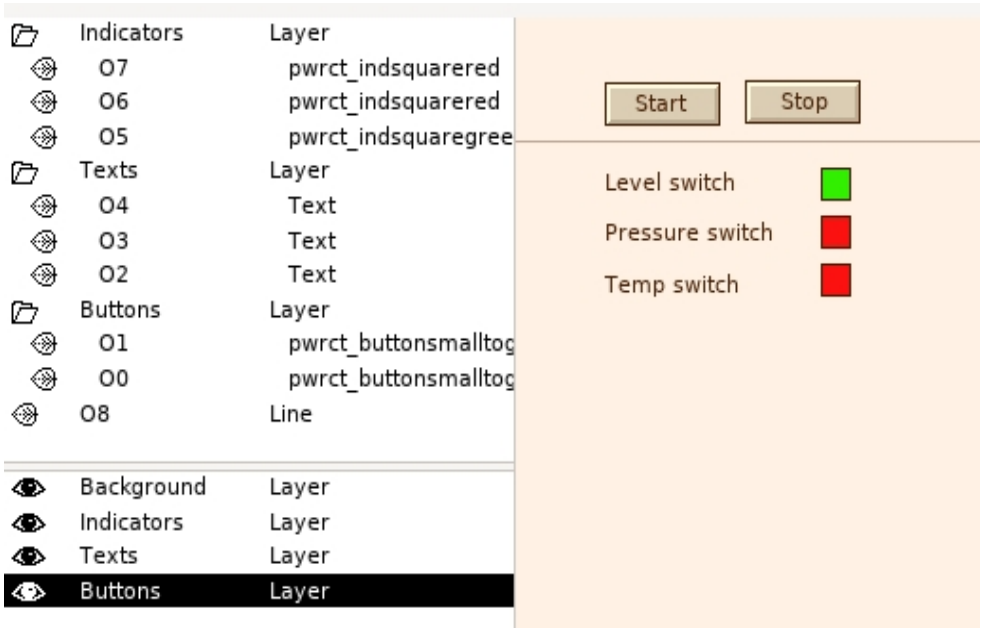


Fig Layers

It's also possible to set dynamics on a layer. By setting invisible dynamics with dimmed on the Buttons layer, all buttons in the layer will be dimmed and insensitive when the connected signal is set.

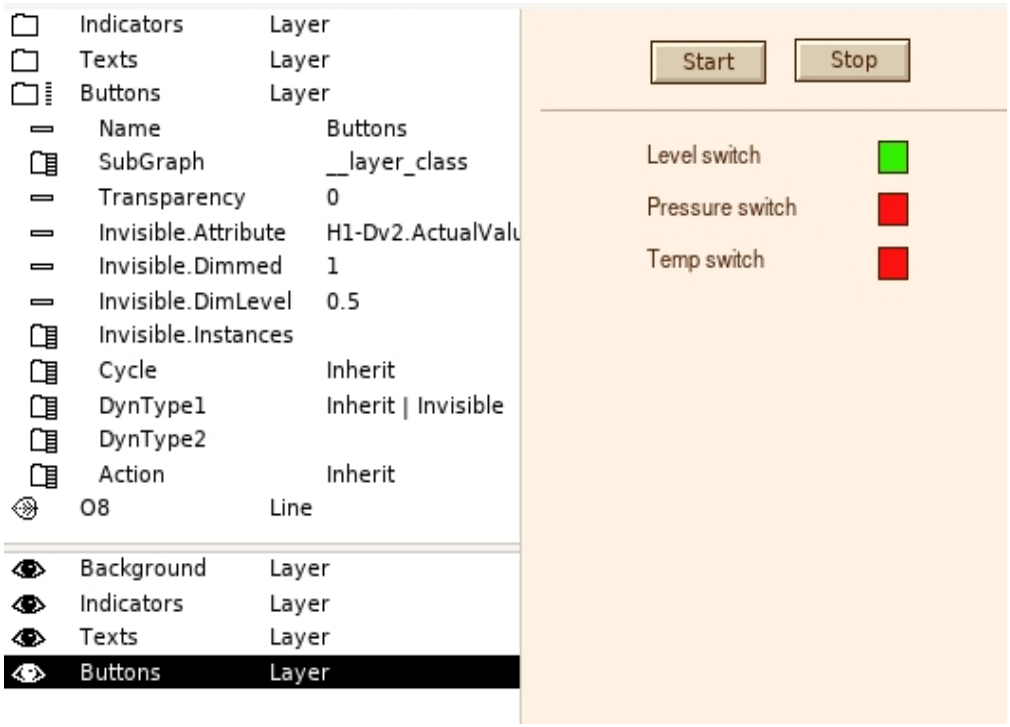


Fig Layer with dynamics

4.11 Draw a subgraph

A subgraph is constructed by drawing base objects in the working area and then save these as a subgraph. You can also draw other subgraphs in the working area, but these can't have any

dynamics of its own, but will follow the dynamic of the main subgraph. There are two special objects that are used in subgraphs: connection points and annotations.

Connection points makes it possible to draw connections between objects. They are created by activating connection point in the tool panel and clicking MB1 in the working area. In the object editor for the connection point you can state the direction (Direction) of the point, i.e. the direction a connection out from the point will get. Connectionpoints on the left side of an object should have Direction Left, on the upper side Up, on the right side Right and on the lower side Down.

Annotation is a place for a text, that can vary for different instance of the subgraph. They are created by selecting An in the tool panel and click with MB1 in the working area. When you create instances of the subgraph, the annotations are displayed as attributes in the object editor for the subgraphs object as A1, A2 etc., and you kan in this way put texts in the annotations.

When the subgraph is ready you open 'File/GraphAttributes' and enter 1 in Subgraph. You can also enter default values for dynamic type or action, e.g. DigLowColor. Then you save it with a suitable name.

You create instances of the subgraphs by selecting it under the folder Local/Subgraphs in the subgraph palette, and click with MB2 in the working area.

A subgraph with several pages

Some types of dynamic, e.g. animations, presume that you hava a subgraph with several pages. When building an animation you create several pages that slightly differs from each other, and the pages are run through, it gives an illusion of movement. The dynamic types DigShift and AnalogShift also uses different pages, and shifts between pages depending on the value of a digital or analog signal.

When to draw a subgraph with several pages, you draw the first page in the usual way. When you have saved it as a subgraph, you activate 'File/Page/Create next page' (Shift+Ctrl/N). Now the second page is created, which is marked in the title of the window. When the changes on page 2 is drawn, this is saved and you again activate 'Create next page' to create the third page etc.

You can shift between pages with 'File/Page/Next page' (Ctrl/N) and 'File/Page/Previous page' (Ctrl+J). When drawing animations, this is a valuable way to see differences between pages and evaluate how the animation works.

Slider

A slider is constructed with two separate subgraphs, one background, and one mobile part. The slider should always be drawn vertically, i.e. with vertical movement direction.

For the background subgraph you measure the y coordinate for the endpoints of the slider movement, and enter the values in the attributes y0 and y1 in 'Graph attributes'. As DynamicType you state SliderBackground.

For the mobile subgraph you measure the y coordinate for the point that is to be adjusted against the signal value, and put this in y0. The attribute Slider should be set to 1.

FillLevel

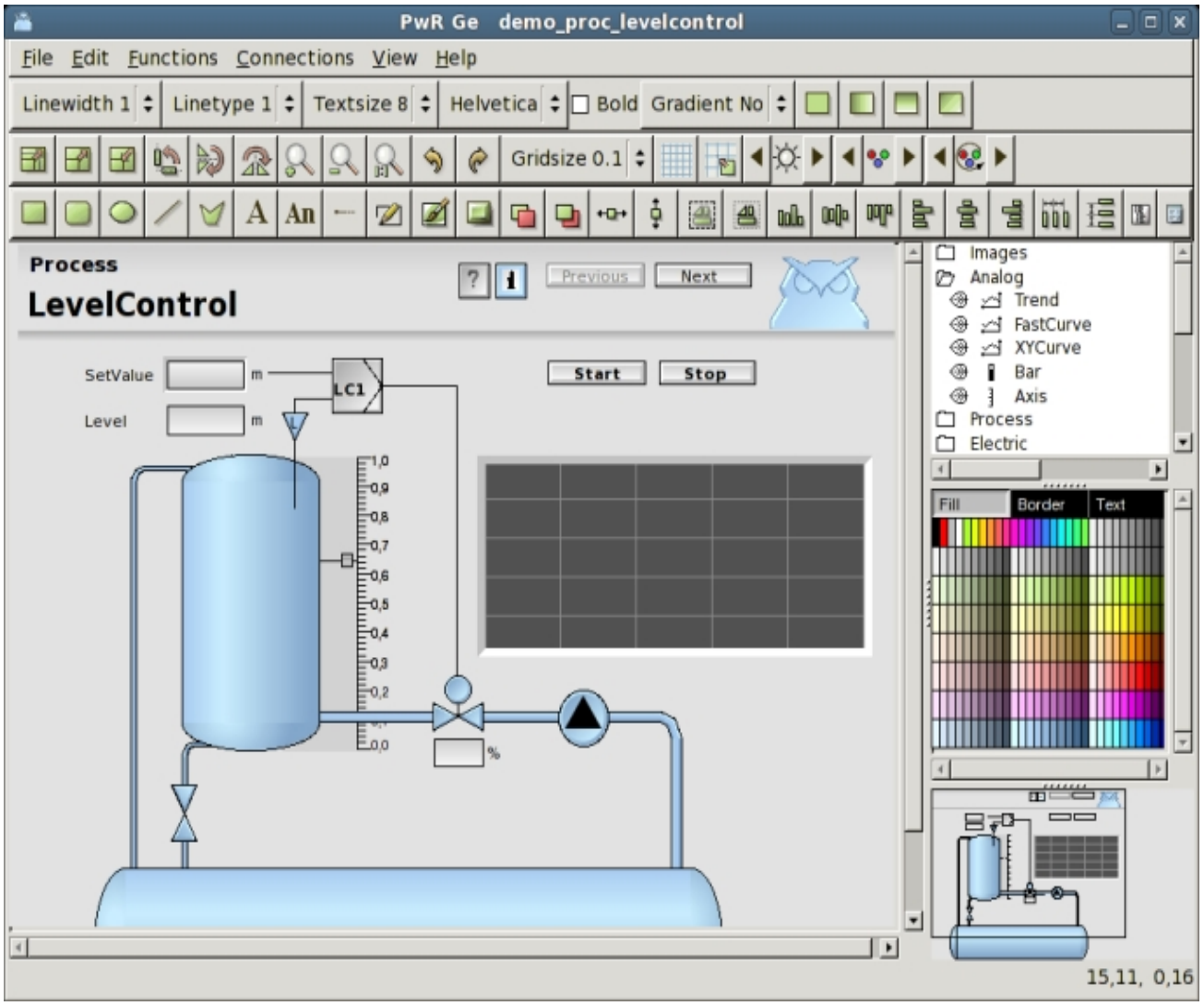
For a subgraph with dynamic FillLevel, e.g. a thermometer, you often want to set a minimum and maximum limit for the level. By measuring the coordinates for the maximum and minium level and put these in y0 and y1 in 'Graph attributes' you achieve this function. The subgraph should

always be drawn in an upright position, so that the level is changed up or down.

Animation

For subgraphs with dynamic Animation, you select a suitable AnimationSequence in 'Graph Attributes'. AnimationSequence can be Cyclic, Dig or ForwBack.

5 Windows



In ge, there are a number of palettes and other windows to choose objects, subgraphs, colors etc.

Working area

In the working area the graph is edited by clicking and dragging the mouse.

Mouse click	Function
Click MB1	Select an object, or create a base object.
Shift/Click MB1	Add an object to the select list.
Press MB1	Select objects in an area, or move objects.
Chift/Press MB1	Add objects in an area to the select list.
DoubleClick MB1	Open the object editor for the object.
Ctrl/DoubleClick MB1	Connect the first attribute in a subgraph object to a rtdb-object.
Shift/Ctrl/DoubleClick MB1	Connect the second attribute in a subgraph object to a

	database object.
Shift/DoubleClick MB1	Set selected fill color in LowColor in a subgraph object.
Click MB2	Create a subgraph object.
Press MB2	Create a connection.

Navigation window

Down to the left there is a navigation window that contains a copy of the working area in reduced scale. From the navigation window you can scroll (MB1) or zoom (MB2) the graph in the working area.

The subgraph palette

The subgraph palette consists of a number of folders with subgraphs. Under the Local/Subgraphs folder there are subgraphs that is created within the project. Other folder contain subgraphs that comes with ProviewR.

The color palette

The color palette contains 300 colors in 10 rows with 30 colors in each row. In the first row there are back, white and intense signal colors, and a collection of gray tones. Then follows a row with the gray scale and after that 8 rows with different color tones: yellowgreen, yellow, orange, violet, blue, seablue and green. Within each color group the 10 first has low saturation, the 10 middle medium saturation and the 10 last high saturation.

The three buttons above the palette displays the current fill, border and text colors. New objects will be created with these colors. To set a new current fill color, set the palette in fill color mode by pressing the Fill button and select a color in the palette. To set a fill color on an existing object, select the object, make sure the palette is in fill color mode, and select the desired color. In the same way the palette can be set in border or text mode by pressing the Border and Text buttons, and the border and text colors of object can be handled.

Object also have a background color, and this is handled differently from the other colors. To set the background color of an object, select the object and click with Ctrl+Shift MB1 on the desired color in the palette.

To set the background color of the graph, set the desired color as current fill color, and activate 'Functions/Set background color' in the menu.

Custom colors

In addition to the predefined colors there are space for 90 custom defined colors below the predefined colors. To set a custom color, enter the color selector by double clicking on an entry in the custom color palette. Press the '+' button to specify a more specific color than the predefined ones. Select the color hue with the slider to the left, and click in color area to select brightness and saturation. The bottom slider for transparency is not implemented yet. Press the Select button to insert the color into the custom color palette. The color can now be used as fill, border or text color.

Fig Color selection

To use a custom color palette in another graph, save the palette from 'File/CustomColors/Save' in the menu. The palette is stored in a pwgc file in $pwrp_pop. Load the file into another graph by activating 'File/CustomColor/Load' and select the file.

It's also possible to use a common color file for several graphs in a project. If a color file is specified in ColorTheme in Graph attributes, this file will be loaded automatically when the graph is opened. When a color is adjusted and stored, it will affect all object drawn with this color in all graphs with this color file. In this way the colors in the graphs can easily be adjusted.

For graphs drawn with color themes, the custom color palette is used for the theme colors and can not be used for custom colors.

Color tones

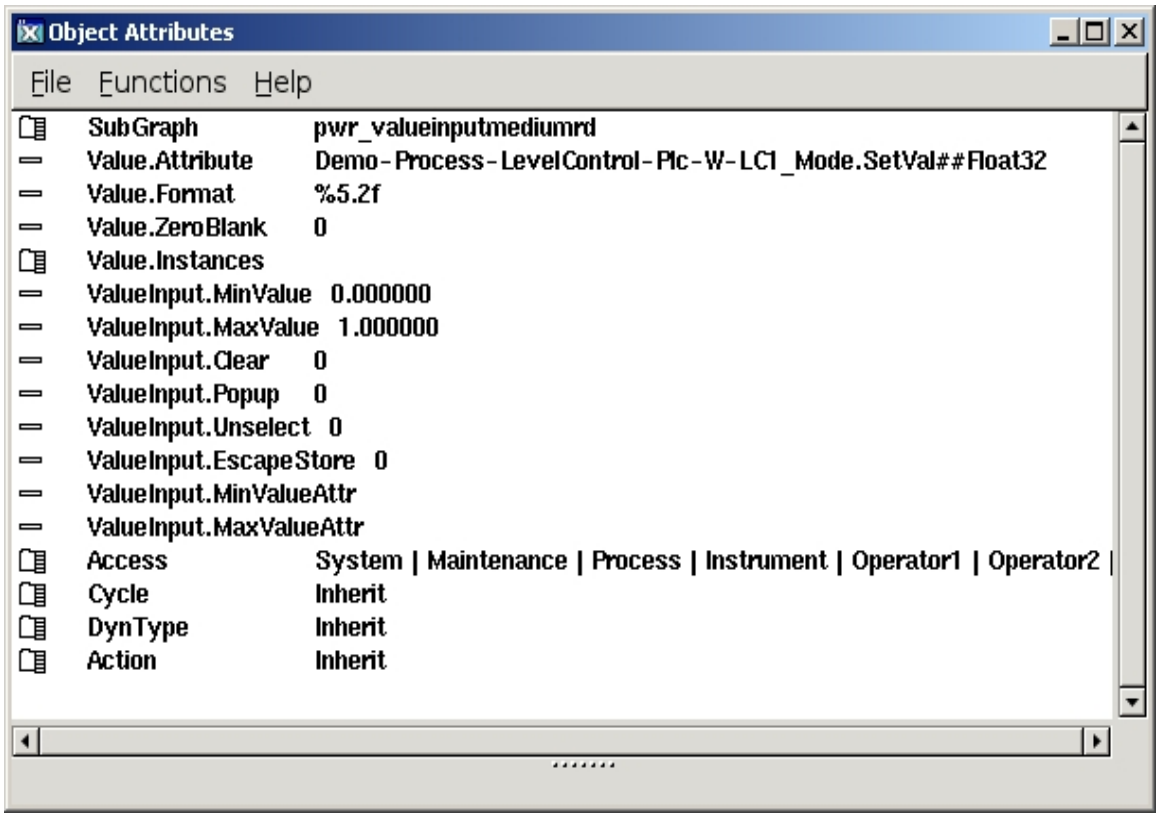
Beneath the color palette there is also a color tone palette in which you can select a color ton for subgraph objects.

Here is also a reset button to return to the original color of an object.

Tool panel

The tool panel consists of a number of pushbuttons to create rectangles, circles, lines etc. There are also functions to rotate and scale, to set line width, font size, to modify color of subgraphs etc.

The object and attribute editor



The object and attribute editor makes it possible to enter properties to objects, graphs and subgraphs.

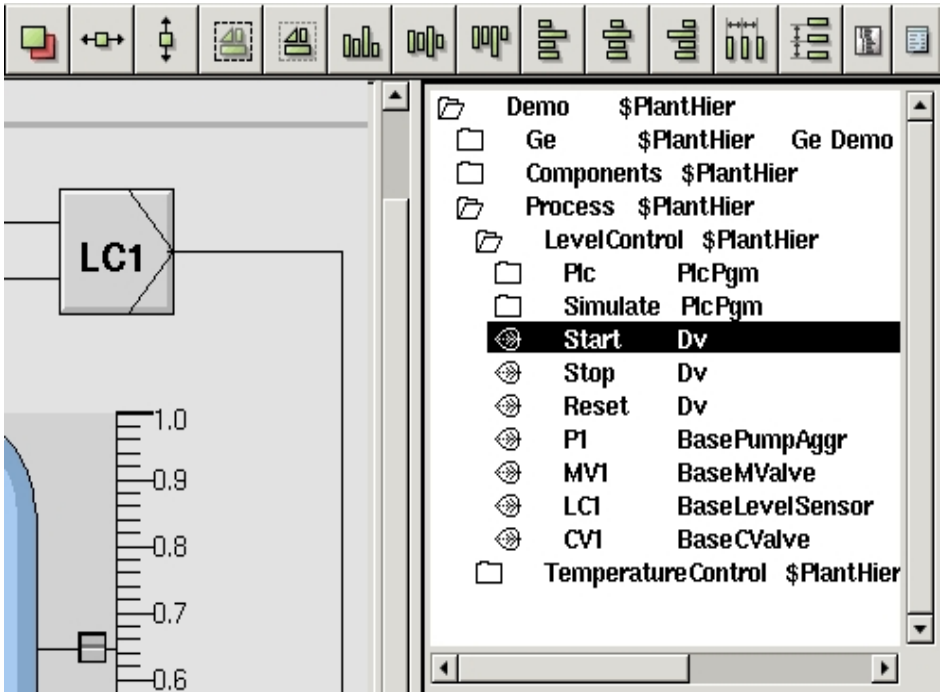
For an object, this is opend by doubleclicking on the object, for graph or subgraphs, it is opened from the menu 'File/Graph attributes'.

Values in the object editor are changed by selecting the attribute and press ArrowRight, or from the menu with 'Functions/Change value'.

Bitmasks, enum attributes and arrays are opened with ArrowRight or DoubleClick MB1. Checkboxes are changed with ArrowRight or by clicking in the checkbox.

The fastest way is to select an attributes with ArrowUp and ArrowDown, and use ArrowRight to open and change the attribute, and ArrowLeft to close the attribute.

Plant hierarchy



If Ge is started from the navigator, the plant hierarchy is displayed in a window below the color palette. It is used to connect dynamic objects to signals in the database. You connect a subgraph object to a signal by selecting the object (or an attribute in the object) in the plant hierarchy, and then click with Ctrl/Doubleclick MB1 on the subgraph object (or on an attribute in the object editor). If there is a second attribute that is to be connected to a signal, you can connect this in the same way with Shift/Ctrl/Doubleclick MB1.

6 Color Themes

Colorthemes provides a number of color combinations for the operator environment, to satisfy the operators preferences and adapt to the brightness of the external environment. The operator can select a color theme from the menu in the operator window, and will affect the appearence of the tools and graph in the ProviewR base system.

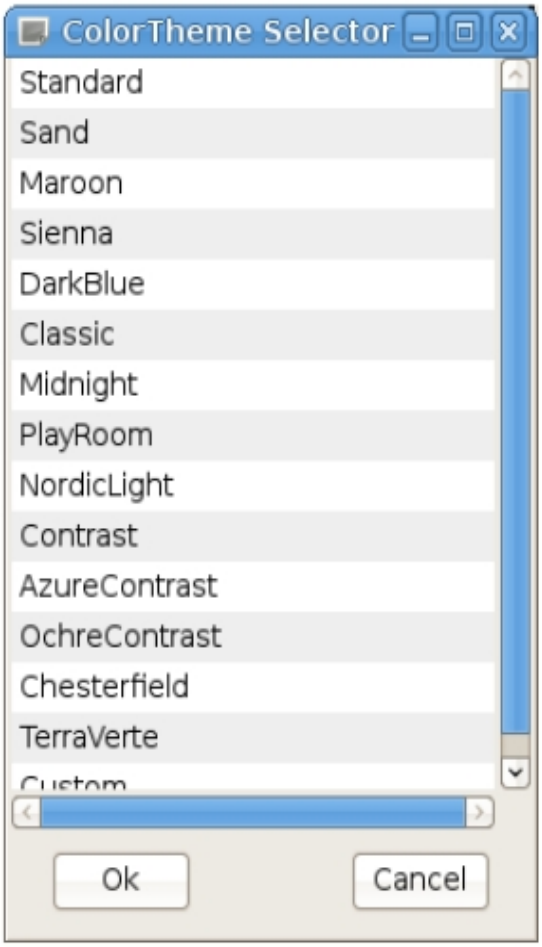
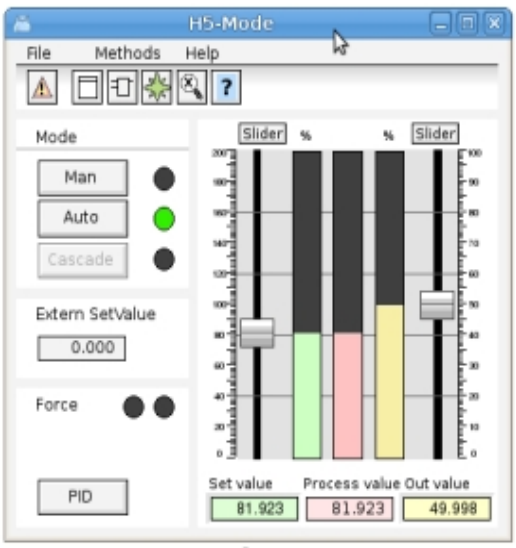


Fig Color theme selection

Fig Operator window with sand color theme



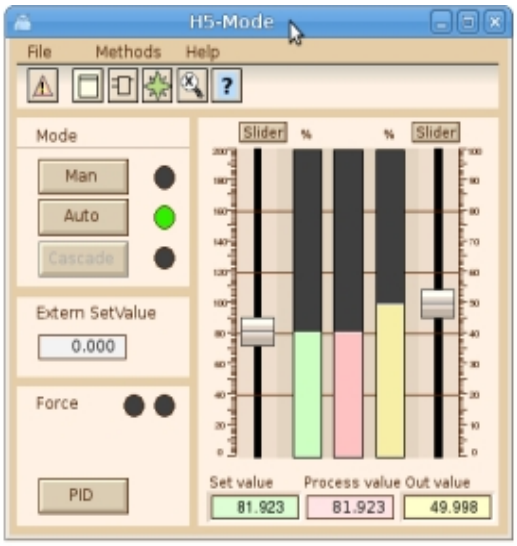


Fig Color themes Standard and Sand

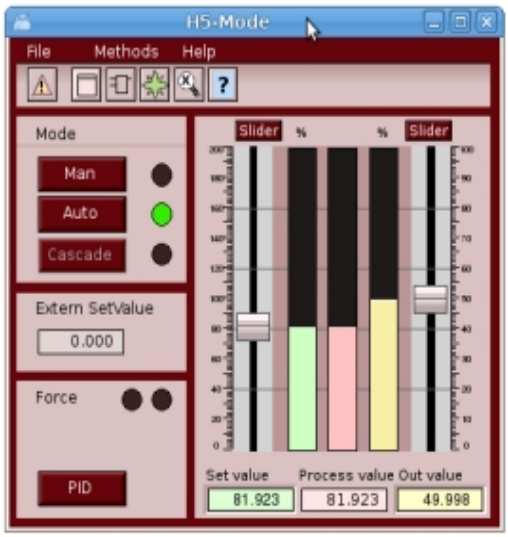


Fig Color themes Marron and Sienna

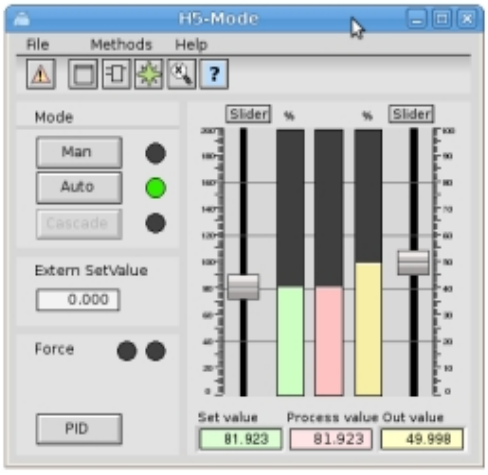


Fig Color themes DarkBlue and Classic

Fig Color themes Midnight and Playroom

Fig Color themes NordicLight and Contrast

Fig Color themes AzurContrast and OchreConstrast

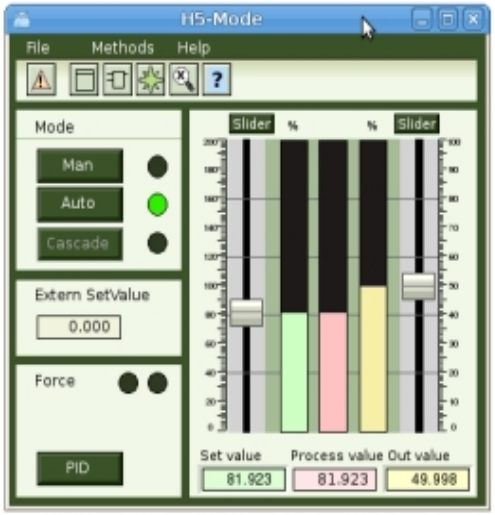


Fig Color themes Chesterfield and TerraVerte

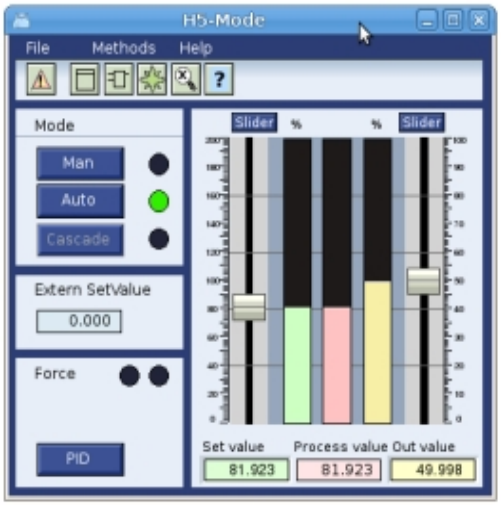


Fig Color theme Polar

The color themes can also be used when drawing graphs in a project. These graphs will follow the appearance of the color theme selected by the operator, but it requires that specific colors in the custom color palette is used when drawing different objects. A color theme is loaded from File/ColorTheme/Select in the Ge menu. The colors for the theme is then loaded into the custom colors palette. Every color in the custom color palette is adjusted for a specific purpose. The first color for example is the background color for the graph, and the fifth color should be set on texts on the background. By placing the cursor on a color in the palette, the purpose of this color will be written in a field below the palette.

The color theme has support for specific elements in a graph. In the example below the background color is lightgreen. The dark green lines are called delimiter areas. In this case it is rectangles the divides the graph into different sections. It can also be used for larger areas with text and indicators. At the top there is a menu bar with pulldown menues, and below this a gradient to make a shadow below the menu. Object graphs should contain a method toolbar positioned below the menu. Other elements are bars, axises, buttons, indicators, texts on the background, and texts on a delimiter area, sliders, value fields and input fields, tables, diagrams etc.

For more complex elements that are are not suited for a change of fill color, as the gray actuator and valve below, the dynamic ColorThemeLightness can be used to adapt the lightness of these elements to the lightness of the theme.

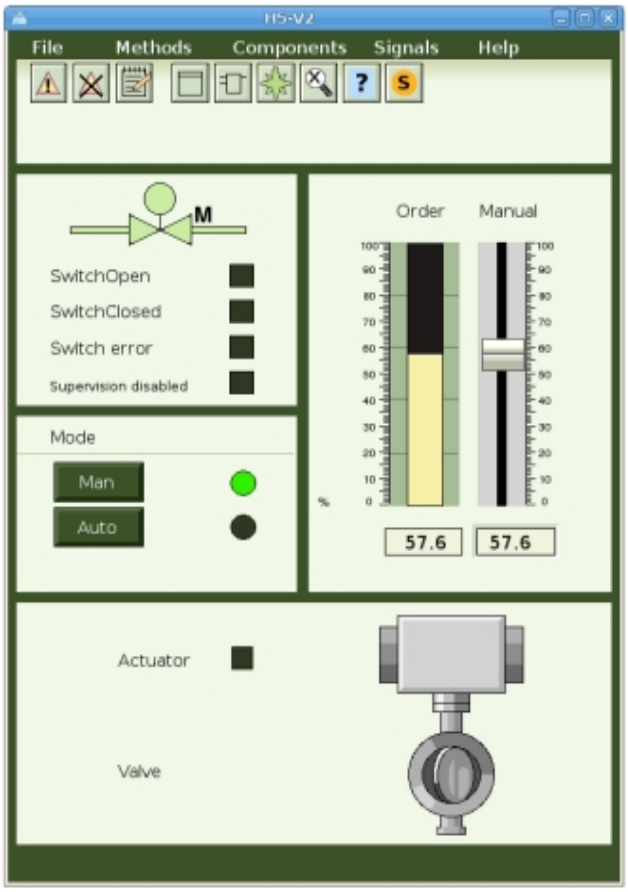


Fig Graph drawn with color theme colors

Description of color theme colors

	Name	Description
1	Background	Should be set as background color for the graph. Light green in the example above.
2	Background gradient	Can be used for gradients to the background color, eg the gradient under the menu above.
3	Delimiter area	Color of delimiter areas. The dark green areas.
4	Delimiter lines	Color of lines that divides the background, eg the line under Mode above.
5	Text/Lines on background	Color of texts and lines on the background.
6	Input field fill color	Fill color for ValueInput objects.
7	Input field border color	Border color for ValueInput objects.
8	Input field text color	Text color for ValueInput objects.
16	Indicator border color	Should be set as border color on indicators.
17	Indicator low color	Low color for indicator
18	Indicator on delimiter low color	Low color for an indicator on a delimiter area.
19	Slider color	Color for sliders.
20	Slider background color	Color for slider background.
21	Value field fill color	Fill color for Value objects.
22	Value field border color	Border color for Value objects.
23	Value field text color	Text color for Value objects.
31	Limit switch high color	Color for limit switch when it's high.
32	Limit switch low color	Color for limit switch when it's low.

33	Limit switch border color	Color for limit switch border.
34	Text/Lines on delimiter	Color for text or lines on a delimiter area.
35	Button active color	Color to indicate that a button is active.
36	Button fill color	Fill color for buttons.
37	Button border color	Border color for buttons
38	Button text color	Text color for buttons
39	Button insensitive text color	Text color set when button is set insensitive by Invisible dynamics.
40	Button insensitive border color	Border color when button is set insensitive by Invisible dynamics.
46	Symbol fill color	Color that can be used component symbols.
47	Symbol border color	Color for symbol border.
48	Symbol low color	Color when symbols is low.
49	Symbol empty color	Color that can be used to indicate that a valve in closed or empty.
50	-	
51	Bar bar color	Color for the value part of a bar object.
52	Bar background color	Color for background part of a bar object.
53	Bar bar limit color	Color for edge of bar.
54	Bar background area	Color used for surrounding area for bar objects. Dark green in the example above.
55	Bar background lines	Color for lines on the surrounding area for bar objects.
61	Menu fill color	Fill color for menu bars and menus.
62	Menu text color	Color for menu texts.
63	Toolbar fill color	Color for method toolbar in object graphs
64	Toolbar border color	Border color for method toolbar.
65	Toolbar text color	Text color for method toolbar.
66	Diagram fill color	Fill color for diagrams.
67	Diagram border color	Border color for diagrams.
68	Diagram curve color	Default curve color for diagram.
69	Axis border color	Color for axis objects.

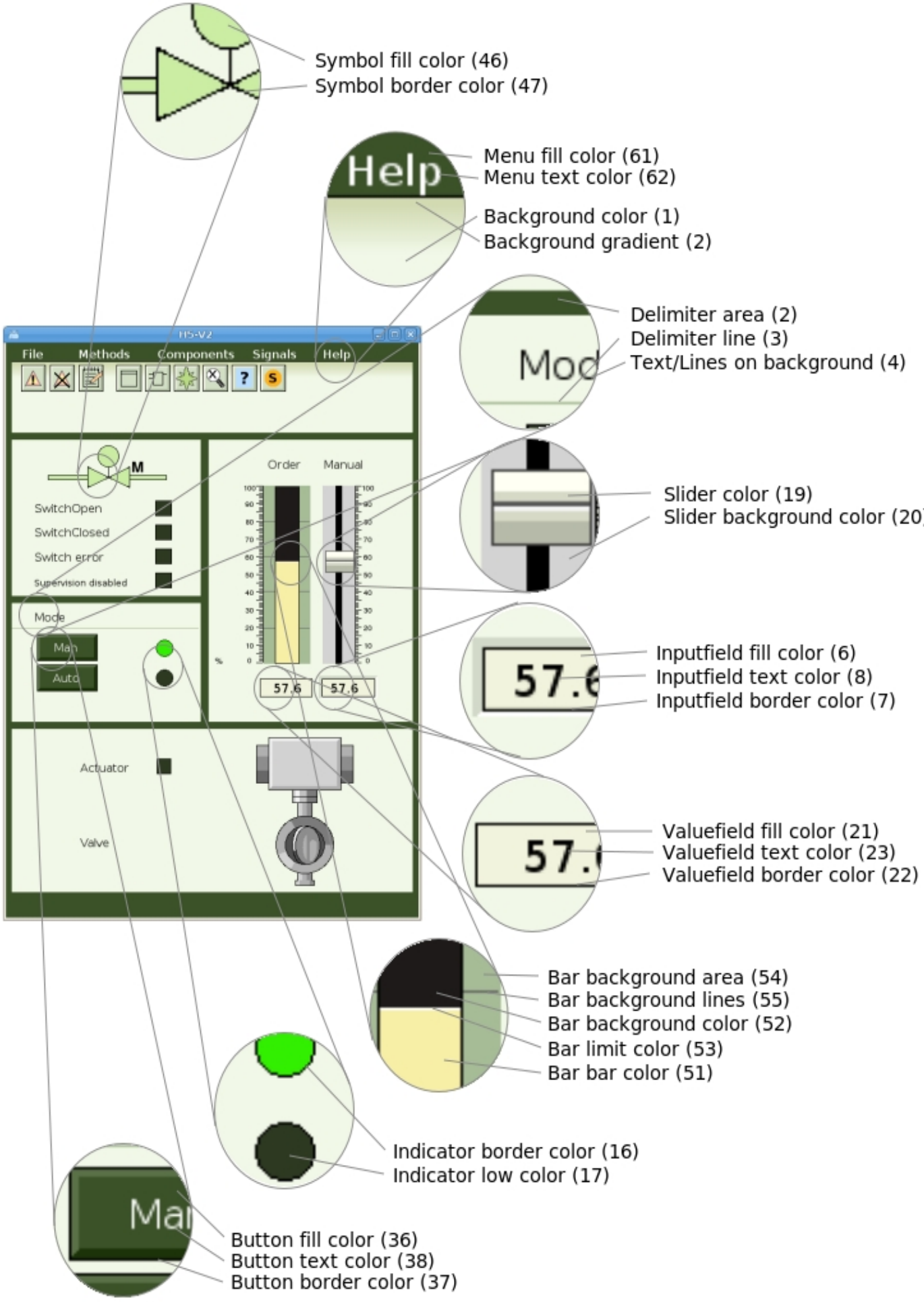

Symbol fill color (46)
Symbol border color (47)
Help
Menu fill color (61)
Menu text color (62)
Background color (1)
Background gradient (2)
Mod
Delimiter area (2)
Delimiter line (3)
Text/Lines on background (4)
H5-V2
File
Methods
Components
Signals
Help
M
SwitchOpen
SwitchClosed
Switch error
Supervision disabled
Order
Manual
Mode
Man
Auto
57.6
57.6
Actuator
Valve
Slider color (19)
Slider background color (20)
57.6
Inputfield fill color (6)
Inputfield text color (8)
Inputfield border color (7)
57.
Valuefield fill color (21)
Valuefield text color (23)
Valuefield border color (22)
Bar background area (54)
Bar background lines (55)
Bar background color (52)
Bar limit color (53)
Bar bar color (51)
Indicator border color (16)
Indicator low color (17)
Mar
Button fill color (36)
Button text color (38)
Button border color (37)

Fig Examples of some color theme colors

Custom color theme

The color theme selector contains an entry with name Custom. To create a custom theme, load a color theme into the custom color palette, double click on the colors to open the color selector and modify the colors. When the desired color are modified, save the theme from /File/CustomColors/Save with name 'pwr_colortheme100'. The color theme is stored in the file $pwrp_pop/pwr_colorthem100.pwgc. Copy this file to $pwrp_exe and make sure it is distributed to the operator stations.

To set the custom color theme in the operator environment, open the color theme selector from 'Functions/View/Color theme' in the operator window menu, and select 'Custom'.

7 Objects

An object is created by selecting the object in the tool panel, and draw it in the working area with MB1. The object is drawn with the current values of fill, fillcolor, border, bordercolor and linewidth. If the properties of an existing object is to be changed, you select it and activate the color, linewidth etc. that should be applied to the object. The object editor for an object is opened by double clicking on the object. Below is described, under the title 'Properties', the properties that can be changed from the tool panel or color palette, and under the title 'Attributes' the attributes found in the object editor.

Base objects

Rectangle
Rounded rectangle
Circle
Line
Polyline
Text
Annotation
Connection point

Complex objects

Bar
BarChart
Pie
Trend
FastCurve
DsTrend
DsTrendCurve
Axis
AxisArc
Window
TabbedWindow
Table
XYCurve

7.1 Rectangle

A rectangle is created by selecting rectangle in the tool panel and drag with MB1 in the working area. If you first activate Functions/ScaleEqual a square is drawn.

Properties

- Fill
- Fillcolor
- Border
- Bordercolor
- Linewidth
- 3D
- Gradient
- Transparency
- Backgroundcolor

Attributes

Attribute	Description
shadow_width	The with of the 3D-shadow in % of width or length (the smallest).
shadow_contrast	Contrast of the 3D-shadow. A value in the range 1 - 3.
gradient_contrast	Contrast of the gradient. A value in the range 0 - 10.
gradient	Type of gradient.
transparency	Level of transparency, A value in the range 0.0 - 1.0.
invisible	The object is invisible but sensitive for mouseclicks. This is used in a subgraph to increase the area of sensitivity for the subgraph.
fill_eq_background	Indicates that the fill color is drawn with the specified background color for the object, instead of the current fill color.
fixcolor	The color is fix. This is used when the object is a part of a subgraph and should not be affected by color setting of the subgraph.
relief	Up gives a shadow on the lower side, and Down gives a shadow on the upper side.
disable_shadow	Disables the 3D function. This is used when the object is a part of a subgraph and the object should not be drawn with 3D although 3D is specified for the subgraph.
disable_gradient	Disables the gradient function. This is used when the object is a part of a subgraph and the object should not be drawn with gradient although gradient is specified for the subgraph.
bgcolor_gradient	The color of the gradient goes from the background color to the fill color.
fixposition	The object can not be moved.
Dynamic	Not implemented.

7.2 Rounded rectangle

A rectangle with rounded corners are created by selecting rounded rectangle in the tool panel and drag with MB1 in the working area. If you first activate Functions/ScaleEqual a square is drawn.

Properties

- Fill
- Fillcolor
- Border
- Bordercolor
- Linewidth
- 3D
- Gradient
- Transparency

Attributes

Attribute	**Description**
round_amount	The amount of the rounded corners in % of length or width (the smallest).
shadow_width	The with of the 3D-shadow in % of width or length (the smallest).
shadow_contrast	Contrast of the 3D-shadow.
relief	Up gives a shadow on the lower side, and Down gives a shadow on the upper side.
gradient_contrast	Contrast of the gradient. A value in the range 0 - 10.
gradient	Type of gradient.
transparency	Level of transparency, A value in the range 0.0 - 1.0.
disable_shadow	Disables the 3D-function. Is used when the object is a part of a subgraph an not should be drawn with 3D when 3D is chosen for the instance of the subgraph.
disable_gradient	Disables the gradient function. This is used when the object is a part of a subgraph and the object should not be drawn with gradient although gradient is specified for the subgraph.
fixposition	The object can not be moved.
Dynamic	Not implemented.

7.3 Ellipse

An ellipse is created by selecting circle in the tool panel and drag with MB1 in the working area. With angle1 and angle2 a segment of the circle is drawn. If you first activate Functions/ScaleEqual a circle is drawn.

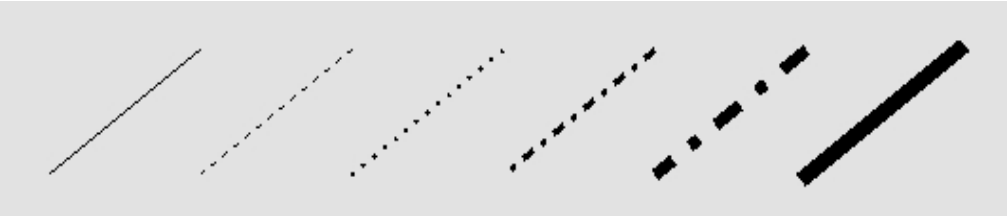
Properties

- Fill
- Fillcolor
- Border
- Bordercolor
- Linewidth
- 3D
- Gradient
- Transparency
- Backgroundcolor

Attributes

Attribute	Description
angle1	Angle from the x-axis in degrees to the start of the segment.
angle2	Angle in degrees that states the size of the segment.
shadow_width	The with of the 3D-shadow in % of width or length (the smallest).
shadow_contrast	Contrast of the 3D-shadow.
gradient_contrast	Contrast of the gradient. A value in the range 0 - 10.
gradient	Type of gradient.
transparency	Level of transparency, A value in the range 0.0 - 1.0.
relief	Up gives a shadow on the lower side, and Down gives a shadow on the upper side.
fixcolor	The color is fix. This is used when the object is a part of a subgraph and should not be affected by color setting of the subgraph.
disable_shadow	Disables the 3D-function. Is used when the object is a part of a subgraph an not should be drawn with 3D when 3D is chosen for the instance of the subgraph.
disable_gradient	Disables the gradient function. This is used when the object is a part of a subgraph and the object should not be drawn with gradient although gradient is specified for the subgraph.
fixposition	The object can not be moved.
fill_eq_background	Indicates that the fill color is drawn with the specified background color for the object, instead of the current fill color.
Dynamic	Not implemented.

7.4 Line



A line is created by selecting line in the tool panel and drag with MB1 in the working area. By activating Functions/MoveRestrictions/Horizontal or Vertical you draw horizontal or vertical lines.

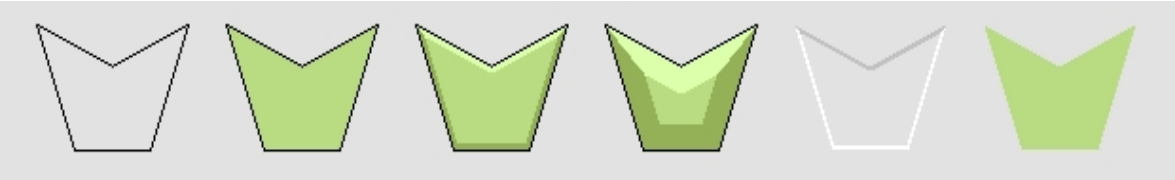
Properties

- Border
- Bordercolor
- Linewidth
- Linetype
- Transparency

Attributes

Attribute	Description
transparency	Level of transparency, A value in the range 0.0 - 1.0.
Dynamic	Not implemented.

7.5 Polyline



A polyline is created by selecting polyline in the tool panel and drag MB1 in the working area. When MB1 is released, the first line segment is created. By dragging MB1 additional line segments are created. When all the line segments are drawn, the drawing sequence is terminated with MB3.

If you want to move a single breakpoint in a polyline, you select the polyline and activate 'Edit/Edit polyline' in the menu. The editing is terminated with MB3.

By activating 'Functions/MoveRestrictions/Horizontal' or 'Vertical' you draw horizontal and vertical lines. Horizontal or Vertical states the direction of the first line segment. The following segments are drawn perpendicular to the previous segment. MoveRestrictions can be disabled when drawing a polyline with 'Functions/MoveRestrictions/No'.

Properties

- Fill
- Fillcolor
- Border
- Bordercolor
- Linewidth
- 3D
- Gradient
- Transparency
- Backgroundcolor

Attributes

Attribute	Description
shadow_width	The with of the 3D-shadow in % of width or length (the smallest).
shadow_contrast	Contrast of the 3D-shadow.
gradient_contrast	Contrast of the gradient. A value in the range 0 - 10.
relief	Up gives a shadow on the lower side, and Down gives a shadow on the upper side.
gradient	Type of gradient.
transparency	Level of transparency, A value in the range 0.0 - 1.0.
disable_shadow	Disables the 3D-function. Is used when the object is a part of a subgraph an not should be drawn with 3D when 3D is chosen for the instance of the subgraph.
disable_gradient	Disables the gradient function. Is used when the object is a part of a subgraph an not should be drawn with gradient when gradient is chosen for the instance of the subgraph.
fill_eq_border	Indicates that the fill color is drawn with the specified border color instead of the current fill color. This is used for subgraphs where certain elements, e.g. arrows, have more in common with the border color than the fill color.

fill_eq_background	Indicates that the fill color is drawn with the specified background color for the object, instead of the current fill color.
fill_eq_light	If the object is a part of a subgraph, it will have the light 3D-color, when 3D is chosen for the subgraph.
fill_eq_shadow	If the object is a part of a subgraph, it will have the same color as a 3D shadow, when 3D is chosen for the subgraph.
fill_eq_bglight	If the object is a part of a subgraph, it is drawn the light 3D-tone of the background color when 3D is chosen for the subgraph.
fill_eq_bgshadow	If the object is a part of a subgraph, it drawn with the dark 3D-tone of the background color, when 3D is chosen for the subgraph.
fixcolor	The color is fix. This is used when the object is a part of a subgraph and should not be affected by color setting of the subgraph.
fixposition	The object can not be moved.
Dynamic	Not implemented.

7.6 Text

A text is created by selecting text in the tool panel and clicking MB1 in the working area. An input field is opened, where the text is entered. The text is modified from 'Edit/Change text' in the menu or from the object editor.

The currently available font is Helvetica, normal or bold. The maximum number of character in the text string is 79.

Properties

- Font size
- Normal or bold text
- Text color
- Transparency

Attributes

Attribute	**Description**
Text	Actual text, maximum 79 characters.
Adjustment	Text adjustment, left, right or center.
Transparency	Level of transparency, A value in the range 0.0 - 1.0.
Dynamic	Not implemented.

7.7 Annotation

An annotation is a text in a subgraph, that is unique for each instance of the subgraph. Examples of annotations are the text of a pushbutton, or the analog value of an update field. When editing the subgraph, the position for the text is stated. The actual text is stated in the object editor for the instance of the subgraph (or handled by certain dynamic functions).

An annotation is created by selecting annotation (An) in the tool panel and clicking MB1 in the working area.

Annotations can only be created in a subgraph. There can be 10 annotations in one subgraph. Each annotation is given a number that is unique within the subgraph.

Note that some types of dynamics (Value, SetDig etc) requires specific numbers of annotations.

Properties

- Normal or bold text
- Text color

Attributes

Attribute	**Description**
TextSize	Text size 0-7
Number	Numbering of annotations within the subgraph.
Adjustment	Text adjustment, left, right or center.
Font	Font
Type	OneLine of MultiLine.
Protected	The text is not displayed. It is marked with asterisks. Use for input of passwords.

7.8 Connection point

A connection point in a subgraph makes it possible to draw connections between instances of the subgraph. In the subgraph, the position of the connection point is stated, as well as the direction in which a connection of the connection point is to be drawn.

A connection point is created by selecting connection point in the tool panel and clicking MB1 in the working area.

Connection points can only be created in subgraphs.

A connection point is marked with a circle when editing the subgraph. In an instance of the subgraph, the connection point is not visible.

If the connection is to be connected to connections of type grafcet or routed, the direction has to be given. It is stated from the object editor.

Each connection point has a number that is unique within a subgraph.

Properties

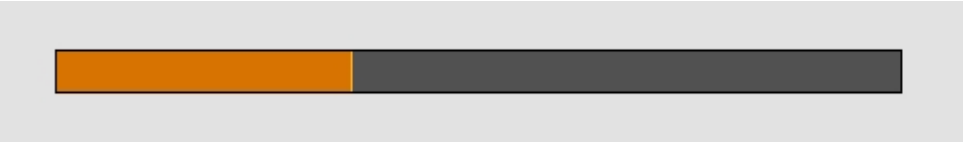
- Direction

Attributes

Attribute	**Description**
Number	Numbering or connection points within a subgraph.
Direction	Direction: up, down, right or left.

7.9 Complex objects

7.9.1 Bar



Bar is one way to display the value of an analog signal.
The properties fill, fillcolor, bordercolor etc. states properties for the background rectangle.
The properties for the bar itself, are stated in the object editor.
A bar is created by selecting bar in the subgraph palette, 'Analog/Bar', and clicking MB2 in the working area.

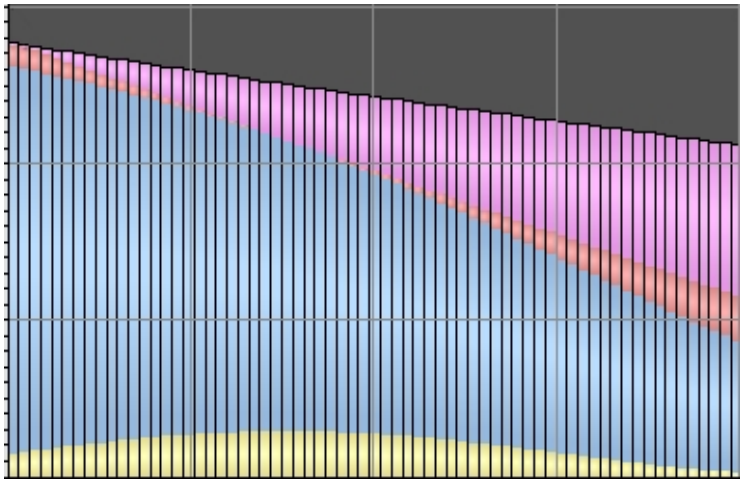
Properties

- Fill (for the background rectangle)
- Fillcolor (for the background rectangle)
- Border (for the background rectangle)
- Bordercolor (for the background rectangle)
- Linewidth (for the background rectangle)

Attributes

Attribute	Description
Bar.Attribute	The name of a signal in the database.
Bar.MaxValue	Signal value that corresponds to full length of the bar.
Bar.MinValue	Signal value that corresponds to zero length of the bar.
Bar.Value	Can be used as a test value in the editor.
Bar.BarColor	Fillcolor for the bar (the value Inherit implies that the border color of the background rectangle is chosen).
Bar.BorderColor	Border color for the upper border line of the bar (the value Inherit implies that that the bordercolor from the background rectangle is chosen).
Bar.BorderWidth	Line width of the borderline (1-8)
Dynamic	Not implemented

7.9.2 BarChart



BarChart displays a number of bars in a chart. Each bar can be divided in segments with different colors. The maximum number of segments is 12. The size of each segment corresponds an to an array of float values, where each element in the array states the segments size in different bars.

The properties fill, fillcolor, bordercolor etc. states properties for the background rectangle.
The properties for the bars and segments are stated in the object editor.
A barchart is created by selecting BarChart in the subgraph palette, 'Analog/BarChart', and clicking MB2 in the working area.

Properties

- Fill (for the background rectangle)
- Fillcolor (for the background rectangle)
- Border (for the background rectangle)
- Bordercolor (for the background rectangle)
- Linewidth (for the background rectangle)

Attributes

Attribute	Description
BarChart.Attribute0	The name of an array in the database. The array values states the of the first segment in the different bars.
BarChart.Attribute1	Array for the second segment.
...	
BarChart.Attribute11	Array for the twelveth segment.
BarChart.Bars	Number of bars.
BarChart.BarSegments	Number of segments in each bar.
BarChart.MaxValue	Value that corresponds to full length of a bar.
BarChart.MinValue	Signal value that corresponds to zero length of a bar.
BarChart.BarColor1	Fillcolor for the first segments.
BarChart.BarColor2	Fillcolor for the second segments.
...	
BarChart.BarColor12	Fillcolor for the twelveth segments.
Dynamic	Not implemented

7.9.3 Pie

Pie is a circular chart with a number of sectors in different color. The maximum number of sectors is 12. The size of each sector corresponds a to a signal value.

The properties fill, fillcolor, bordercolor etc. states properties for the background circle.
The properties for the sectors are stated in the object editor.
A Pie is created by selecting Pie in the subgraph palette, 'Analog/Pie', and clicking MB2 in the working area.

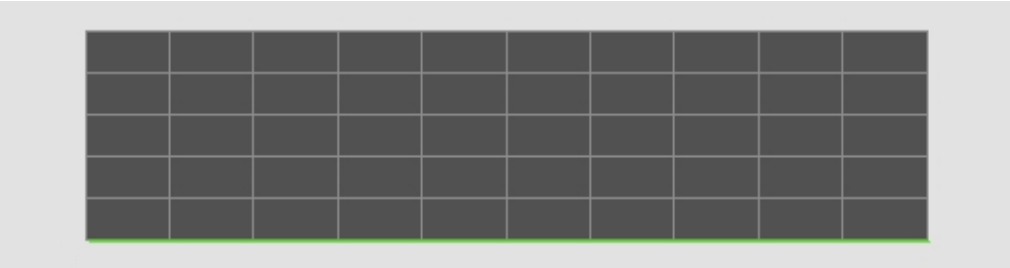
Properties

- Fill (for the background circle)
- Fillcolor (for the background circle)
- Border (for the background circle)
- Bordercolor (for the background circle)
- Linewidth (for the background circle)

Attributes

Attribute	Description
Pie.Attribute0	Database attribute of type Float that influences the size of the first sector.
Pie.Attribute1	Database attribute for the second sector.
...	
Pie.Attribute11	Database attribute for the twelfth sector.
Pie.Angle1	Angle from the x-axis in degrees to the start of the chart.
Pie.Angle2	Angle in degrees that states the size of the chart.
Pie.Sectors	Number of sectors.
Pie.MaxValue	Maximum value for the diagram range.
Pie.MinValue	Minimum value for the diagram range.
Pie.SectorColor1	Fillcolor for the first sector.
Pie.SectorColor2	Fillcolor for the second sector.
...	
Pie.SectorColor12	Fillcolor for the twelveth sector.
Dynamic	Not implemented

7.9.4 Trend



Trend displays the value of one or two analog values as curves.

The cuves can be drawn with or without fill, where fill means that that area between the curve and the time axis is filled. The curves are drawn with the second curve on top.

A number of horizontal and vertical lines can be drawn with the border color in the trend object. The number of lines are configured from the object editor.

The properties fill, fillcolor, bordercolor etc., states the properties for the background rectangle of the curve. The properties of the curve itself are stated from the object editor.

A trend is created by selecting trend in the subgraph palette, 'Analog-Trend', and clicking MB2 in the working area.

Properties

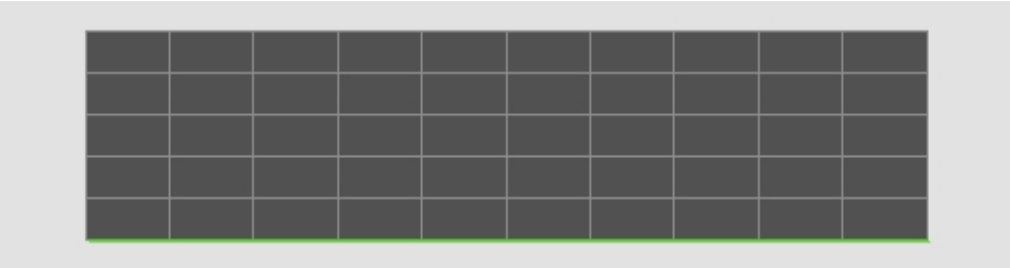
- Fill (for the background rectangle)
- Fillcolor (for the background rectangle)
- Border (for the background rectangle)
- Bordercolor (for the background rectangle)
- Linewidth (for the background rectangle)

Attributes

Attribute	Description
Trend.Attribute1	Signal for curve number 1.
Trend.Attribute2	Signal for curve number 2.
Trend.MinValueAttr1	Signal for minimum value curve number 1. Is used when the min value is dynamic. For static value, Trend.MinValue1 is used.
Trend.MaxValueAttr1	Signal for maximum value curve number 1. Is used when the max value is dynamic. For static value, Trend.MaxValue1 is used.
Trend.MinValueAttr2	Signal for minimum value curve number 2. Is used when the min value is dynamic. For static value, Trend.MinValue1 is used.
Trend.MaxValueAttr2	Signal for maximum value curve number 2. Is used when the max value is dynamic. For static value, Trend.MaxValue1 is used.
Trend.HoldAttr	Signal to freeze the curve.
Trend.TimeRangeAttr	Signal of type Float32 for the time range of the curve.
Trend.Mark1Attr	Database attribute for the position of the first horizonal marker line.
Trend.Mark2Attr	Database attribute for the position of the second horizontal marker line.

Trend.Mark1Color	Color of the first marker line.
Trend.Mark2Color	Color of the second marker line.
Trend.NoOfPoints	Number of points in the curve.
Trend.ScanTime	Time interval between two points.
Trend.CurveLineWidth	Linewidth for the curve (1-8)
Trend.FillCurve	States that the area between the curve and the time axis is to be filled.
Trend.HorizontalLines	Number of horizontal lines
Trend.VerticalLines	Number or vertical lines
Trend.MaxValue1	Maximum value of Attribute1
Trend.MinValue1	Minimum value of Attribute1
Trend.CurveColor1	Border color for the curve of Attribute1
Trend.CurveFillColor1	Fill color for the curve of Attribute1
Trend.MaxValue2	Maximum value of Attribute2
Trend.MinValue2	Minimum value of Attribute2
Trend.CurveColor2	Border color for the curve of Attribute2
Trend.CurveFillColor2	Fill color for the curve of Attribute2
Trend.Dynamic	Not implemented

7.9.5 FastCurve

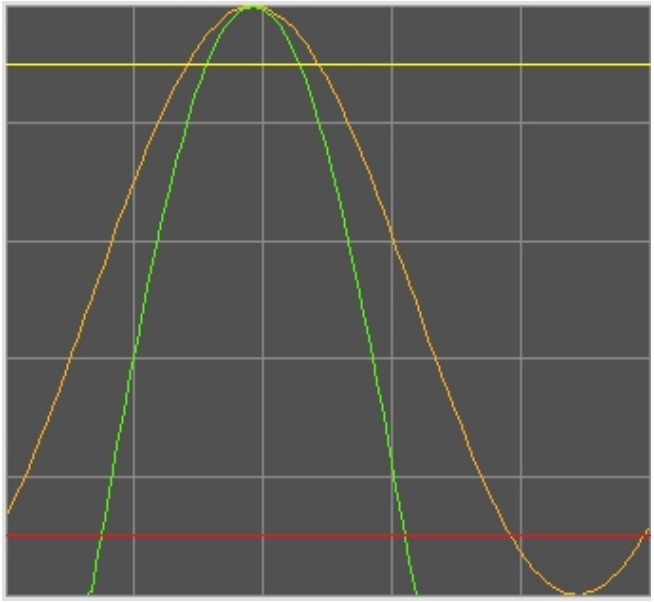


The FastCurve displays one or two curves configured with a DsFastCurve object. A Fast curve is the measurement of a signal for a certain period of time. The measurement is triggered by some event, and when the measurement is done, the curve is displayed in the FastCurve object.

Attributes

Attribute	Description
FastCurve.FastObject	Database object of class DsFastCurve.
FastCurve.CurveIndex1	Index in the DsFastCurve object for first curve.
FastCurve.CurveIndex2	Index in the DsFastCurve object for second curve.
FastCurve.NoOfPoints	Number of points in the curve.
FastCurve.CurveLineWidth	Line width.
FastCurve.FillCurve	Fill the area between the curve and the x-axis.
FastCurve.HorizontalLines	Number of horizontal lines in the diagram.
FastCurve.VerticalLines	Number of vertical lines.
FastCurve.MaxValue1	Maximum value in range for curve 1.
FastCurve.MinValue1	Minimum value in range for curve 1.
FastCurve.CurveColor1	Color for curve 1.
FastCurve.CurveFillColor1	Fill color for curve 1.
FastCurve.MaxValue2	Maximum value in range for curve 2.
FastCurve.MinValue2	Minimum value in range for curve 2.
FastCurve.CurveColor2	Color for curve 2.
FastCurve.CurveFillColor2	Fill color for curve 2.

7.9.6 DsTrend

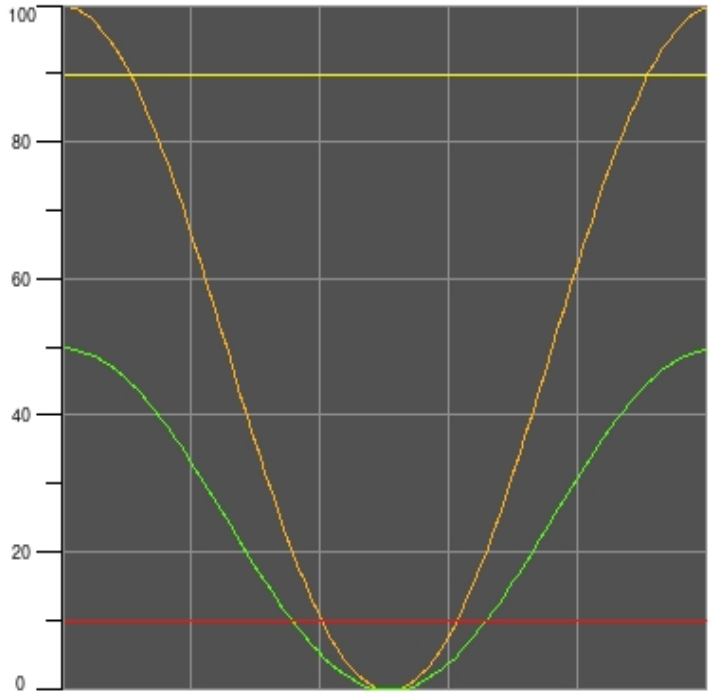


The DsTrend displays one or two curves configured with DsTrend objects.

Attributes

Attribute	Description
DsTrend.Object1	Database object of class DsTrend for curve 1.
DsTrend.Object2	Database object of class DsTrend for curve 2. Optional.
DsTrend.MinValueAttr1	Attribute for minimum value of curve 1. Optional.
DsTrend.MaxValueAttr1	Attribute for maximum value of curve 1. Optional.
DsTrend.MinValueAttr2	Attribute for minimum value of curve 2. Optional.
DsTrend.MaxValueAttr2	Attribute for maximum value of curve 2. Optional.
DsTrend.HoldAttr	Attribute to hold updating of the curve. Optional.
DsTrend.Mark1Attr	Attribute for marker 1. Optional.
DsTrend.Mark2Attr	Attribute for marker 2. Optional.
DsTrend.Mark1Color	Marker 1 color.
DsTrend.Mark2Color	Marker 2 color.
DsTrend.NoOfPoints	Number of points in the curve.
DsTrend.CurveLineWidth	Line width.
DsTrend.FillCurve	Fill the area between the curve and the x-axis.
DsTrend.HorizontalLines	Number of horizontal lines in the diagram.
DsTrend.VerticalLines	Number of vertical lines.
DsTrend.MaxValue1	Maximum value in range for curve 1.
DsTrend.MinValue1	Minimum value in range for curve 1.
DsTrend.CurveColor1	Color for curve 1.
DsTrend.CurveFillColor1	Fill color for curve 1.
DsTrend.MaxValue2	Maximum value in range for curve 2.
DsTrend.MinValue2	Minimum value in range for curve 2.
DsTrend.CurveColor2	Color for curve 2.
DsTrend.CurveFillColor2	Fill color for curve 2.
DsTrend.Direction	Direction of curve movement, Right or Left.

7.9.7 DsTrendCurve



The DsTrendCurve displays one or two curves configured with a DsTrendCurve object.

Attributes

Attribute	Description
DsTrendCurve.Object	Database object of class DsTrendCurve.
DsTrendCurve.MinValueAttr1	Attribute for minimum value of curve 1. Optional.
DsTrendCurve.MaxValueAttr1	Attribute for maximum value of curve 1. Optional.
DsTrendCurve.MinValueAttr2	Attribute for minimum value of curve 2. Optional.
DsTrendCurve.MaxValueAttr2	Attribute for maximum value of curve 2. Optional.
DsTrendCurve.HoldAttr	Attribute to hold updating of the curve. Optional.
DsTrendCurve.Mark1Attr	Attribute for marker 1. Optional.
DsTrendCurve.Mark2Attr	Attribute for marker 2. Optional.
DsTrendCurve.Mark1Color	Marker 1 color.
DsTrendCurve.Mark2Color	Marker 2 color.
DsTrendCurve.NoOfPoints	Number of points in the curve.
DsTrendCurve.CurveLineWidth	Line width.
DsTrendCurve.FillCurve	Fill the area between the curve and the x-axis.
DsTrendCurve.HorizontalLines	Number of horizontal lines in the diagram.
DsTrendCurve.VerticalLines	Number of vertical lines.
DsTrendCurve.MaxValue1	Maximum value in range for curve 1.
DsTrendCurve.MinValue1	Minimum value in range for curve 1.
DsTrendCurve.CurveColor1	Color for curve 1.
DsTrendCurve.CurveFillColor1	Fill color for curve 1.
DsTrendCurve.MaxValue2	Maximum value in range for curve 2.
DsTrendCurve.MinValue2	Minimum value in range for curve 2.
DsTrendCurve.CurveColor2	Color for curve 2.

DsTrendCurve.CurveFillColor2	Fill color for curve 2.
DsTrendCurve.Direction	Direction of curve movement, Right or Left.

7.9.8 Axis

Axis shows the range for a curve or bar in x or y direction. There are two variants of Axis, one with static min and max values, and one with dynamic min and max values, where min and max are connected to signals in the database. For a dynamic Axis object, the values of Lines, LongQuotient, ValueQuotient and Format are calculated automatically in runtime.

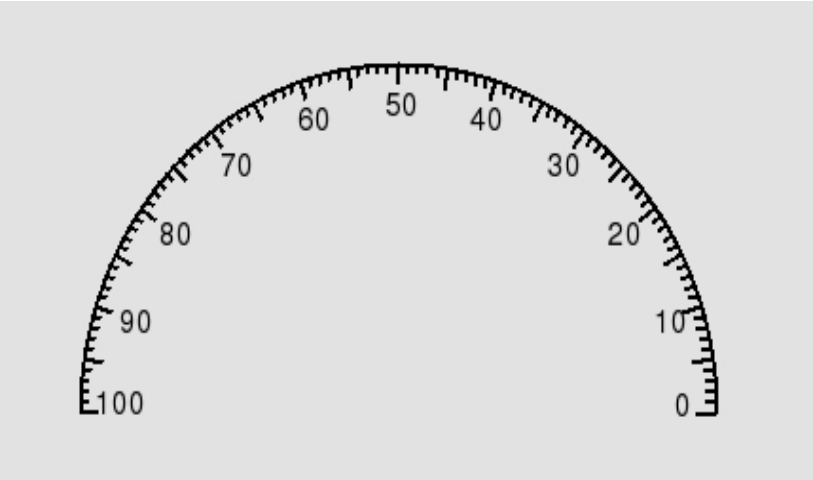
Attributes

Attribute	**Description**
MaxValue	Maximum value for the range.
MinValue	Minimum value for the range.
Lines	Number of lines perpendicular to the axis.
LongQuotient	Specification of lines that is a bit longer. For example 4 implies that every fourth line is a bit longer.
ValueQuotient	How often a value is to be written. For example 4 impies that a value is written at every fourth line.
Format	Format in c syntax of written values.
Dynamic	Not implemented.

Dynamic Axis

Axis.MinValueAttr	Min value signal of type Float32 or Int32.
Axis.MaxValueAttr	Max value signal of type Float32 of Int32.
Axis.KeepSettings	If 0, new values of Lines, LongQuotient, ValueQoutient and Format are calculated when the Min or Max value is dymaically changed. If KeepSettings are 1, the original values will be kept.

7.9.9 AxisArc



AxisArc is an arc shaped scale that displays the range for a needle.
As for axis there are two variants of AxisArc, one with static min and max values, and one with dynamic min and max values, where min and max are connected to signals in the database. For a dynamic AxisArc object, the values of Lines, LongQuotient, ValueQuotient and Format are calculated automatically in runtime.

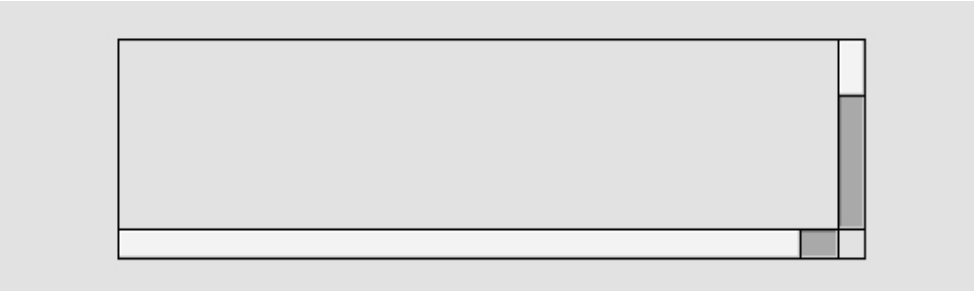
Attributes

Attribute	**Description**
Angle1	Angle from the x-axis in degrees to the start of the axis.
Angle2	Angle in degrees that states the size of the axis.
LineLength	Length of lines perpendicular to the axis.
MaxValue	Maximum value for the range.
MinValue	Minimum value for the range.
Lines	Number of lines perpendicular to the axis.
LongQuotient	Specification of lines that is a bit longer. For example 4 implies that every fourth line is a bit longer.
ValueQuotient	How often a value is to be written. For example 4 impies that a value is written at every fourth line.
Format	Format in c syntax of written values.
Dynamic	Not implemented.

Dynamic AxisArc

Axis.MinValueAttr	Min value signal of type Float32 or Int32.
Axis.MaxValueAttr	Max value signal of type Float32 of Int32.
Axis.KeepSettings	If 0, new values of Lines, LongQuotient, ValueQoutient and Format are calculated when the Min or Max value is dymaically changed. If KeepSettings are 1, the original values will be kept.

7.9.10 Window



Window displays a separate graph in a specified area in another graph. The graph is displayed with or without scrollbars.

Limitations

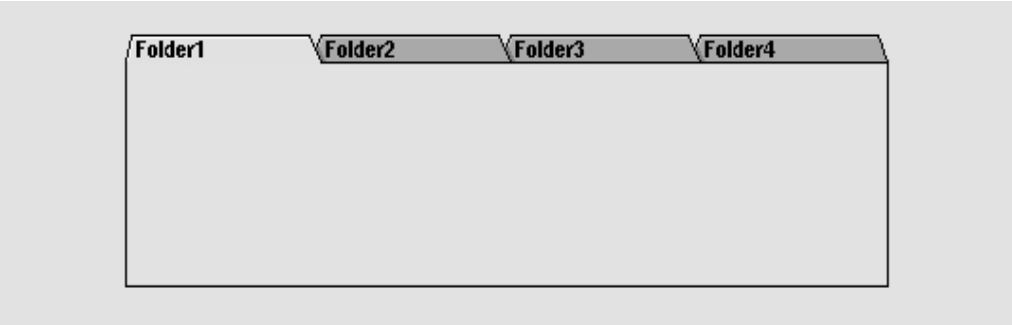
Connections within the window object are not scaled properly.

A window object grapbs all events inside the window area, and click sensitive objects can not be placed on top of the window. If the window temporary should be covered by sensitive objects, the xtt command 'set subevents' can be used to disable the window event handling.

Attributes

Attribute	Description
Window.FileName	Name of the graph to be displayed in the window.
Window.Scale	Scale of the graph.
Window.VerticalScrollbar	A vertical scrollbar is displayed in the window.
Window.HorizontalScrollbar	A horizontal scrollbar is displayed in the window.
Window.ScrollbarWidth	Width of scrollbar.
Window.ScrollbarColor	Fill color of the mobile part of the scrollbar.
Window.ScrollbarBgColor	Fill color of the background part of the scrollbar.

7.9.11 TabbedWindow



TabbedWindow is a window object with folders. For each folder a separate graph is stated, and when that folder is activated, the stated graph is displayed in the window.

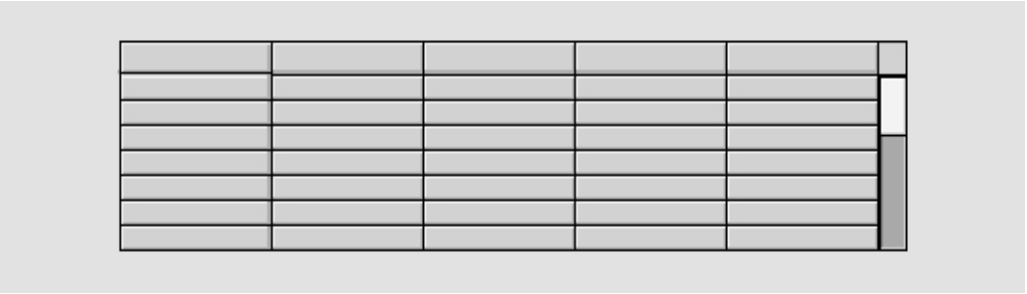
Limitations

Sliders within the window object are disabled.
You can not state an hierarchy or class object.

Attributes

Attribute	Description
Folder.NumberOfFolders	Number of folders.
Folder.HeaderHeight	Folder height.
Folder.ColorSelected	Color of the active folder.
Folder.ColorUnselected	Color of inactive folders.
Folder.ScrollbarWidth	Width of scrollbar.
Folder.ScrollbarColor	Fill color of the mobile part of a scrollbar.
Folder.ScrollbarBgColor	Fill color of the background part of a scrollbar.
Folder1.FileName	Name of the graph that is displayed in the first folder.
Folder1.Text	Text of the first folder.
Folder1.Scale	Scale of the graph in folder number 1.
Folder1.VerticalScrollbar	A vertical scrollbar is displayed for folder 1.
Folder1.HorizontalScrollbar	A horizontal scrollbar is displayed for folder 1.

7.9.12 Table



A table object displays a table with a number of rows and columns. Each column is connected to an array attribute in the database, and the values of the array elements are displayed in the column.

A cell in the table is selected by clicking on it, or by the arrow keys (if action type InputFocus is configured for the table). To each column you can connect a select attribute, i.e. an array attribute in the database of type Boolean. When a cell is selected, the corresponding element in the select attribute is set, and the previous selected is reset.

If the connected array attribute to a column is of type Objid, you can open a popup menu with the methods of the objects displayed in the column.

The array attributes should be stated with data type and size, for example VWX-P1-Table.TabVect##Float32#100, where Float32 is the datatype and 100 the number of elements in the array.

The table object consists of the following components:
- Vertical and horizontal scrollbar.
- A header row with a title for each column. The header row is not comprised by the vertical scrollbar.
- A header column. The left most column can be configured as a header column, i.e. it is not comprised by the horizontal scrollbar.

Limitations

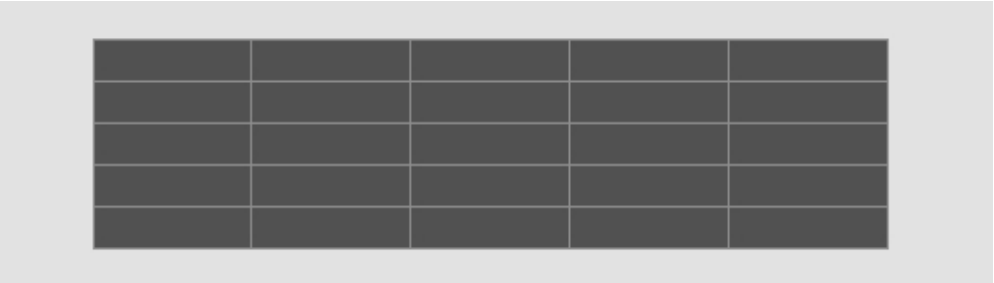
A table object grapbs all events inside the table area, and click sensitive objects can not be placed on top of the table. If the table temporary should be covered by sensitive objects, the xtt command 'set subevents' can be used to disable the table event handling.

Attributes

Attribute	Description
Table.Rows	Number of rows in the table (header row excluded).
Table.Columns	Number of columns in the table (header column included).
Table.HeaderRow	A header row with title for each column is drawn.
Table.HeaderColumn	The leftmost column is not comprised by the horizontal scrollbar.
Table.RowHeight	Row height.
Table.HeaderTextSize	Text size of the text in the header row.
Table.HeaderTextBold	Bold text in the header row.

Table.HeaderTextColor	Text color in the header row.
Table.Options	Bitmask with options 1: If the cell in the leftmost column is empty, all the cells on that row is drawn empty.
Table.VerticalScrollbar	Vertical scrollbar in the table.
Table.HorizontalScrollbar	Horizontal scrollbar in the table.
Table.ScrollbarWidth	Scrollbar width.
Table.ScrollbarColor	Fill color of the mobile part of the scrollbar.
Table.ScrollbarBgColor	Fill color of the background part of the scrollbar.
Column1.Attribute	Array attribute. The value of each element is displayed in the corresponding row in the column. The attribute should be stated with data type and size, see above.
Column1.Format	Format in c syntax.
Column1.SelectAttribute	Array attribute of type Boolean, When a cell in the column is selected, the corresponding element in the array is set. The attribute should be stated with data type and size, see above.
Column1.Width	Width of the column.
Column1.HeaderText	Column title. The title is written in the header row.

7.9.13 XYCurve



XYCurve displays curves defined with x and y coordinates. 10 curves can be displayed in one XYCurve object. For each curve, the object is connected to two float arrays containing the x and y coordinates for the curve. An UpdateAttr is also connected to indicate when new coordinate values are present and the curves should be redrawn. The ranges in x and y direction can be stated with fix values, or connected to attributes in the database. Also the number of points can be defined statically with a connection to a database attribute.

The XYCurve can be connected to any attributes in the database. There is though a specific class, XyCurve, that can be used to store a curve with 100 points. It contains two arrays with 100 elements for the coordinates, together with attributes for the ranges, number of points and update indication.

Attributes

Attribute	Description
XY_Curve.XAttr	Database array containing the x coordinates for the curve.
XY_Curve.YAttr	Database array containing the y coordinates for the curve.
XY_Curve.UpdateAttr	Boolean database attribute that indicates that the curves should be updated.
XY_Curve.XMinValue	Minimum value of range in x direction.
XY_Curve.XMaxValue	Maximum value of range in x direction.
XY_Curve.YMinValue	Minimum value of range in y direction.
XY_Curve.YMaxValue	Maximum value of range in y direction.
XY_Curve.XMinValueAttr	Database attribute that contains the minimum value of range in x direction.
XY_Curve.XMaxValueAttr	Database attribute that contains the maximum value of range in x direction.
XY_Curve.YMinValueAttr	Database attribute that contains the minimum value of range in y direction.
XY_Curve.YMaxValueAttr	Database attribute that contains the maximum value of range in y direction.
XY_Curve.NoOfPointsAttr	Integer database attribute with number of points in the curve.
XY_Curve.CurveColor	Color of curve.
XY_Curve.FillColor	Fill color of curve.
XY_Curve.Instances	Adding new instances makes it possible to display several curves in the same diagram. Maximum 10 curves is possible to show.
XY_Curve.NoOfPoints	Number of points in curve.
XY_Curve.CurveLineWidth	Line width of curve.
XY_Curve.FillCurve	If 1 the curve is filled.

XY_Curve.HorizontalLines	Number of horizontal lines in the diagram.
XY_Curve.VerticalLines	Number of vertical lines in the diagram.
XY_Curve.HorizontalPadding	The curve outside the x value range will be drawn with horizonal lines from the first or last point.
XY_Curve.XMark1Attr	Database attribute for the x position of the first vertical marker line.
XY_Curve.XMark2Attr	Database attribute for the x position of the second vertical marker line.
XY_Curve.YMark1Attr	Database attribute for the y position of the first horizonal marker line.
XY_Curve.YMark2Attr	Database attribute for the y position of the second horizontal marker line.
XY_Curve.Mark1Color	Color of the first marker lines (vertical and horizontal).
XY_Curve.Mark2Color	Color of the second marker lines (vertical and horizontal).

8 Subgraphs

A subgraph is a class or pattern that is built of a number of base objects.

An instance of the subgraph is created by selecting the subgraph in the subgraph palette, and clicking MB1 in the working area. In the editor, the instances are handled as other objects, and the fillcolor, bordercolor and linewidth of the instances can be set.

Color

You can set the fill color and border color of a subgraph object in the same way as the base objects. It implies that these properties are set to all the base objects of the subgraph.
If you set the fill color of a subgraph that is drawn with shadows and reflections to give a 3D effect, this effect will disappear and the objects is totally flattened out. To modify the color of this kind of subgraph you use the functions for lightness, color intensity and color tone in the tool panel.

Lightness
The function for lightness is found in the tool panel and marked with sun. Here are increase and decrease buttons that makes the fill color of the object lighter or darker. The lightness can be modified in seven steps.

Intensity
The function for color intensity is found in the tool panel and marked with three color points. The intensity can be modified in three steps: strong intensity, medium and gray.

Shift
Color shift implies that every color in the subgraph object is rotated on the color circle.
An object that contains blue and green colors will, for example, after four shift steps contain orange and red colors. The colors keeps their original lightness and intensity, and also their internal color contrast.

The function is found in the tool panel and marked with color points surrounded by an circular arrow. The colors of the first row of the color palette are not affected by the shift function.

Tone
Toning of an object, implies that all the color of the subgraph object gets the same color tone. They keep their lightness and intensity. There are nine different color tones: gray, yellowgreen, yellow, orange, red, violet, blue, seablue and green.

The function for tone is found in the color palette below the selection of fill, border and text color. Here is also a button to reset the color of an objects to the original colors.

Animation and shift

Subgraphs with several pages
Animation and Shift are types of dynamics that don't just change the color of the subgraph, they also change the shape. This is done by creating a subgraph with several pages.

To create additional pages for a subgraph you activate 'File/Page/Create next page'. This function will create a new subgraph with the suffix '__px', where x is the page number, and it will link this to the previous page by putting the name in the attribute NextSubgraph in the graph attributes. For the last page, NextSubgraph is empty. When a page is created and saved, you can easily shift to the next or previous page from 'File/Page/NextPage' and 'File/Page/PreviousPage' in the menu.

You should save the different pages with the same zoom factor, otherwise you might get one pixels displacement of pages in runtime.

Shift

Shift between different pages in runtime is done with the dynamic types DigShift, DigFourShift and AnalogShift. DigShift shifts between the first and second page depending on a digital value. DigFourShift shifts between four pages depending on the values of three digital signals. AnalogShift is connected to an analog signal, where the value corresponds the the index for the page to be displayed (the first page has index 0).

Animation

When an animation is run, the pages are displayed, one after another and gives the impression of movement.

The interval between each page shift, normally has to be much shorter than the normal update interval. For a graph, you can state the animation scan time in 'File/Graph attributes' in the menu. Note that the time is common for all subgraphs in the graph, and can not be stated for each subgraph individually. A suitable value for the animation scan time is 0.2 seconds.

If an animation should rest on the same page for several animation cycles, you can enter the number of cycles in AnimationCount in Graph attributes for the page.

Animations are handled by the dynamic type Animation. It is connected to a digital signal that controls start and stop of the animation. It also has the attribute Sequence which tells in which order, and on what condition, the animation is run. Sequence can have the value Cyclic, Dig or ForwBack.

If Sequence is Cyclic, the first page is displayed when the value of the signal is 0. When the value of the signal is 1, the animation is active. It starts at page 2 and continues to the last page, after that it jumps to the second page again and continues to the last page etc. This continues until the value of the signal becomes 0.

The sequence ForwBack is similar, but here, the animation runs continuously forward and back between the first and last page, as long as the signal value is 1.

If Sequence is Dig the animation is run only when the signal value is changed. When the value is changed to 1, the animation is run from the first to the last page, and stays on the last page, when the value is changed to 0, the animation is run back to the first page, and rests there.

8.1 Database connection

Subgraphs are connected to attributes in the database which makes it possible to change color or shape of an object when an attribute value is changed. For example the dynamic

DigColor has the property Attribute, where the database attribute that affects the color is specified.

The attribute specification consists of the attribute name type, for example

```
H1-H2-Start.ActualValue##Boolean
```

Usually the attribute is inserted by selecting the attribute in the database navigator, and clicking on the attribute property in the object editor with Ctrl+Double click. If only one attribute is present you can click directly on the subgraph with Ctrl+Double click.

Invert signal
A signal is inverted by placing an exclamation mark in front of the attribute name. Only attributes of type Boolean can be inverted. Inverted signal can only be used in dynamics, not in actions.

```
!H1-H2-Start.ActualValue##Boolean
```

The exclamation mark will invert the signal before it is used in the dynamics.

Object and hierarchy graphs
When drawing object graphs, the instance object or hierarchy object is denoted by the string $object. For example if the attribute

```
$object.ActualValue##Float32
```

occurs in an object graph, and the object graph is opened for the object H1-H2-Temperature, $object is replace by the instance object name and the connection will be made to

```
H1-H2-Temperature.ActualValue##Float32
```

$object is also used in hierarchy graphs in a similar way.

```
$object-Temperature.ActualValue##Float32
```

with the hierarchy object H1-H2 will give

```
H1-H2-Temperatyre.ActualValue##Float32
```

Parent object
The parent object to the $object can be referenced with '.<' or '-<'. These operators will remove the last attribute or the last object to the closest preceding '.' or '-'. H1-H2-<.Description will be H1.Description. If you have an hierarchy graph for H1-H2, the description in the parent object H1 can be displayed with

```
$object-<.Description##String80
```

For attribute objects, where attribute names is removed instead, '.<' is used. H1-Plates.Data.<.Temperature will be H1-Plates.Temperature. Thus in the object graph for Data the Temperature attribute in parent object Plates can be displayed by

```
$object.<.Temperature##Float32
```

Attribute references
If an object contains attributes of type AttrRef of Objid, values in the objects that these attribute points at, can be displayed with the &() syntax.

```
&(H1-H2-Motor.Object).ActualValue##Float32
```

means that H1-H2-Motor.Object contains a pointer to an object, and it is to the ActualValue attribute of this object the connection is made.

The connection is made when the graph is opened, and if the pointer in H1-H2-Motor.Object is changed, the connection will not be automatically changed. The dynamic DigSwap though can be used to reconnect all connections in a graph.

Variable array index
Array indexes can be specified with attributes of type Int32 or UInt32. In

```
H1-H2-Array[&(H1-H2-CurrentIndex.ActualValue)].Value##Float32
```

H1-H2-CurrentIndex.ActualValue is an Int32 attribute, and the value of this attribute will be fetched and inserted as array index.

The index is evaluated when the graph is opened and if the index value is changed, the connection will not be automatically changed. The dynamic DigSwap though can be used to reconnect all connections in a graph.

Node object
$node will be replaced by the node object. This can for example be used to reference XttGraph objects where the same graph are used on different nodes. For example

```
$node-Graphs-Overview
```

will work for any node that has an XttGraph object named Overview under a Graph hierarchy under the node object.

Local database
Local variables in a graph can be used to communicate between objects in a graph. They are specified with the prefix $local. followed by a varable name and variable type, eg

```
$local.HoldTrend##Boolean
```

A local variable can for example be used to hold a trend by beeing used in Trend.HoldAttr. With a toggle button that uses the same local variable in ToggleDig.Attribute, trend hold can be turned on and off by pressing the button.

Local variables though are not implemented for all types of dynamics and actions.

Arrays
In some dynamic types as Table and XYCurve whole arrays are connected. The syntax to connect an array is 'attributename'##'type'#'length', for example

```
H1-H2-A1.Value##Float32#100
```

where 100 is the number of elements in the array.

Array elements
To connect an element in an array, the syntax is 'attributename'##'type'#'length'['index'], eg

```
H1-H2-A1.Value##Float32#100[2]
```

Attribute types

The type of the attribute is specified after the attribute name. The following types are supported

##Boolean
##Float32
##Float64
##Char
##Int8
##Int16
##Int32
##Int64
##UInt8
##UInt16
##UInt32
##UInt64
##Objid
##Time
##DeltaTime
##Status
##NetStatus
##Enum
##Mask
##DataRef
##String
##Bit

String type
The String type should be followed by the size of the string, eg ##String80.

Bit type
The bit type makes it possible to connect to an individual bit in a mask or integer attribute.

The syntax is ##Bit#'size'['bitnumber'] or ##Bit['bitnumber'].

Size is the number of bits in the mask, and bitnumber the number where the first bit has number 0.

Example

```
H1-H2-Mode.ActualValue##Bit[2]
```

8.2 Dynamics

You can apply a number of different types of dynamic for the subgraphs. The dynamic defines how signals in the database should influence for example the color, or to print out an analog value.

There are a number of predefined types of dynamics, e.g shift between two colors or between two texts. When you draw the subgraph you state the type of dynamic that is proper for this subgraph, but this is only a default value for each instance, and can be replace by another type.

Dynamic	Signal type	Description
Inherit	-	The dynamic for the instance is inherited from the subgraph.

Tone	-	States that dynamics that changes the color of the component should change the color tone instead of the fill color.
DigLowColor	Boolean	Set the specified fill color when the signal is low.
DigColor	Boolean	Set the specified fill color when the signal is high.
AnalogColor	Float32 Int32	Set specified fill color when the signal goes beyond or below the specified limit.
StatusColor	Status	Set fill color dependent on a status signal severity.
DigError	Boolean	Set red fill color when the signal is high.
DigWarning	Boolean	Set yellow fill color when the signal is high.
DigFlash	Boolean	Flash with specified color when the signal is high.
FillLevel	Float32	Changes partly the fill color of the component. The level for the borderline of the colored part is determined by the value of the signal.
Invisible	Boolean String	Make the component invisible when the signal is high, or when the string is empty.
DigBorder	Boolean	Set specified border color when the signal is low.
DigBackgroundColor	Boolean	Set specified background color when signal is high.
TimeoutColor	Boolean	Set specified color when at subscription timeout.
DigTextColor	Boolean	Set specified text color when the signal is high.
DigText	Boolean	Set specified text when the signal is low.
AnalogText	Float32 Int32	Set up till 32 different texts depending on the value of an analog signal.
Value	Arbitrary	Write the value of an attribute.
Rotate	Float32 Int32	Rotate the component.
Move	Float32 Int32	Move and scale the component is x and y direction.
DigShift	Boolean	Shift between two pages in the subgraph (first and last page).
DigLowShift	Boolean	Shift between two pages in the subgraph (first and last page).
DigFourShift	Boolean	Shift between four pages in the subgraph.
AnalogShift	Float32 Int32	Shift between different pages in the subgraph. The value of the signal determines the page number.
Animation	Boolean	Different types of animation.
Video	-	Display a continuously updated image file.
SliderBackground	-	Indicates that the subgraph is background to a slider component.
DigCommand	Boolean	Executes a command when signal gets high.
DigScript	Boolean	Executes a script when signal gets high.
ScrollingText	Boolean	Display a scrolling text.
DigSound	Boolean	Play a sound.
DigTransparency	Boolean	Shift transparency depending on the value of a digital signal.
AnalogTransparency	Float32	Set transparency depending on the value of an analog signal.
ColorThemeLightness	-	Adapt lightness of component to color theme.
DigSwap	Boolean	Restart all subscriptions in the graph.
UnitConvert	Float32	Convert value unit before displaying in value field.
HostObject	Object	Complex component dynamics.

Inherit

For an instance of a subgraph, Inherit means, that the dynamic is inherited from the subgraph class, that is the default dynamic stated when the subgraph was edited. If also the default dynamic is inherit, the subgraph lacks dynamics.

The default dynamic for an instance, is displayed by opening the object editor for the instance and there open the subgraph folder.

Xtt-commands

There are a number of pushbuttons that executes xtt-commands. The most usual is to open another graph, but you can also open trace or trends etc. Here follows some usable xtt-commands.

```
open graph GraphName [/width=][/height=][/scrollbar][/navigator]

open trace WindowName [/center=]
```

Access

All dynamics that makes it possible to influence the value of a signal in the database, has the attribute Access. Access states the privileges that are required for a user to be allowed to change a value. There are 15 different privileges that are of interest for graphs, 10 for various operators, and 4 for different professionals. Furthermore there is RtRead, read access in runtime, that a user that is not logged in is granted. A user grants one or several privileges, and if one of his privileges is present in Access, he has the right to influence the object.

Default access is all privileges except RtRead. Some operations, as opening other graphs, should also accept RtRead.

Privileges in runtime	**Description**
RtRead	Read access.
System	System manager.
Maintenance	Maintenance technicians.
Process	Process technicians.
Instrument	Instrument technicians.
Operator1	Different operator or operator places.
Operator2	"
Operator3	"
Operator4	"
Operator5	"
Operator6	"
Operator7	"
Operator8	"
Operator9	"
Operator10	"

8.2.1 DigLowColor

Set specified fill color when the signal is low.

The object is connected to a digital signal in the database. If the value is 1, the object is drawn with the normal fill color, if the value is 0, it is drawn with DigLowColor.Color which is stated in the object editor.

Compare to DigColor below, that set the specified color when the signal is high instead of low. If to use DigLowColor or DigColor is a question how the component is to be drawn in the editor: The way is will appear when the signal is high or low, i.e. if you want to draw the graph as it will look when the plant is running or stopped.

Attribute	**Description**
DigLowColor.Attribute	Signal in the database of type Boolean that should influence the component.
DigLowColor.Color	Fill color of the component when the signal is 0.

8.2.2 DigColor

Set specified fill color when the signal is high.

The object is connected to a digital signal in the database. If the value is 1 the object is drawn with DigColor.Color which is stated from the object editor, if the value is 0 it is drawn with the normal fillcolor.

DigColor can exist in several instances, which makes it possible to change between several colors (up to 32 colors). Instances with higher number has higher priority, i.e. if the signal for a higher instance is high, it determines the color independent of the values of signals connected to lower instances.

Attribute	Description
DigColor.Attribute	Signal in the database of type boolean that should influence the component.
DigColor.Color	Fill color of the component when the signal is 1.
DigColor.Instance	States the number of instances that is created.
DigColor2.Attribute	Signal for instance number 2.
DigColor2.Color	Fill color for instance number 2.
DigColor3.Attribute	Signal for instance number 3.
DigColor3.Color	Fill color for instance number 3.

8.2.3 AnalogColor

AnalogColor is connected to an analog signal, and sets the specified fill color when the signal goes below or beyond a limit. The limit is specified in AnalogColor.Limit and AnalogColor.LimitType determines if it is an upper or lower limit. If LimitType is GreaterThan, the fillcolor is set, when the value of the signal exceeds the limit, and when LimitType is LessThan, the fillcolor is set when the signal value is below the limit. Other limit types are GreaterEqual, LessEqual and Equal.

AnalogColor can exist in several instances, which makes it possible to set several limit values and to shift between several colors (up to 32 colors).

Either all instances work with the same signal, or each signal has it's own attribute. This is configured with AnalogColor.CommonAttribute.

Note! If there are several instances with equal LimitType, a higher instance has to have a higher limit, when LimitType is GreaterThan or GreaterEqual. If LimitType is LessThan or LessEqual, a higher instance has to have a lower limit.

Attribute	**Description**
AnalogColor.Attribute	Signal in the database of type Float32 or Int32. The value is compared to the limit, and if it is beyond/below the limit the color of the component is changed.
AnalogColor.Color	Fill color of the component value when the signal has passed the limit.
AnalogColor.Border	Set border color instead of fill color.
AnalogColor.CommonAttribute	The same attribute is used for all instances. Otherwise each instance will have it's own attribute.
AnalogColor.Limit	Limit value.
AnalogColor.LimitType	Type of limit. Can be GreaterThan, GreaterEqual, LessThan, LessEqual or Equal.
AnalogColor.Instances	Instance mask where instances can be added or removed.
AnalogColor2.Color	Fill color for instance number 2.
AnalogColor2.Limit	Limit for instance number 2.
AnalogColor2.LimitType	Type of limit for instance number 2.
AnalogColor3.Color	Fill color for instance number 3.

8.2.4 StatusColor

StatusColor is connected to an attribute of type Status. A status is defined by a text and a severity, where the severity is one of the categories Success, Info Warning, Error and Fatal. The status value can also be 0, NoStatus. StatusColor set fill color from the status severity.

Success, Info	Original color.
Warning	Yellow.
Error	Red.
Fatal	Flashing red.

Attribute	**Description**
StatusColor.Attribute	Signal in the database of type Status. The color of the component is set from the severity of the status value.
StatusColor.NoStatusColor	Color when the status value is 0.
StatusColor.UseColorTheme	When set to 1, color theme colors will be used.

8.2.5 DigWarning

Set fill color to yellow when the signal is high.

The object is connected to a digital signal in the database. If the value is 1, the object is drawn with yellow fill color, if the value is 0, it is drawn with the normal fill color.

Attribute	Description
DigWarning.Attribute	Signal in the database of type Boolean that should influence the object.
DigWarning.UseColorTheme	When set to 1, color theme colors will be used.

8.2.6 DigError

Set the fill color to red when the signal is high.

The object is connected to a digital signal in the database. If the value is 1, the object is drawn with red fill color, if the value is 0, it is drawn with the normal fill color.

Attribute	Description
DigError.Attribute	Signal in the database of type Boolean that should influence the object.
DigError.UseColorTheme	When set to 1, color theme colors will be used.

8.2.7 DigFlash

Flash with specified colors when the signal is high.

DigFlash changes the color of the object every cycle. The color is shifted between the colors specified in Color and Color2. When the signal is low, the object is drawn with the original color.

The flash frequency is determined by the cycle time of the cycle of the object.

Attribute	Description
DigFlash.Attribute	Signal in the database that should influence the object.
DigFlash.Color	First flash color.
DigFlash.Color2	Second flash color.

8.2.8 FillLevel

Draw the object with a specified fill color, up to a certain level.

FillLevel draws the original color (or the color determined by other dynamics) up to a certain border line. On the other side of the borderline the object is drawn with the specified background color. The position of the borderline is determined from the value of the analog signal. If the value is less then MinValue, the whole object is drawn with background color. If the value is greater than MaxValue the object is drawn with the original color.

Direction states the direction of the movement of the borderline. If Direction is Up the border is move upwards with increasing value, if Down downwards, if Left to the left and if right to the right.

Attribute	**Description**
FillLevel.Attribute	Signal in the database of type Float32.
FillLevel.BackgroundColor	Background color.
FillLevel.Direction	Direction Up, Down, Left or Right.
FillLevel.MinValue	Min value. When the signal value equals MinValue, the whole object is drawn with background color.
FillLevel.MaxValue	Max value. When the signal value equals MaxValue, the whole object is drawn with original color.
FillLevel.MinValueAttr	Signal for minimum value.
FillLevel.MaxValueAttr	Signal for maximum value.

8.2.9 Invisible

Makes the object invisible (or dimmed) when the signal is high.
The signal is usually connected to a digital signal, but can also be connected to a string.
In this case the object is invisible when the string is empty. Invisible can also be used with the xtt-command 'check' to, for example, check if a method is defined for an object. In Attribute you then write $cmd('command'), for example

```
$cmd(check method/method=Note/object=VK-Valve)
```

Attribute	Description
Invisible.Attribute	Signal in the database of type Boolean that should influence the object.
Invisible.Dimmed	1 dims the object, 0 makes the object invisible.
Invisible.DimLevel	A transparency value between 0 and 1 to add transparency to the dimmed appearance.
Invisible.Instances	Instance mask where instances can be added or removed.

8.2.10 DigBorder

Set specified border color when the signal is low.

The object is connected to a digital signal in the database. If the value is 1, the object is drawn with the normal border color, if the value is 0, it is drawn with DigBorder.Color that is stated from the object editor.

Attribute	Description
DigBorder.Attribute	Signal in the database of type Boolean the should influence the component.
DigBorder.LowColor	Border color of the component when the signal is 0.

8.2.11 DigBackgroundColor

The DigBackgroundColor dynamics makes it possible to set dynamics on the background color. DigBackgroundColor works analogous with DigColor, but will changes the background color instead of the fill color. The DigBackgroundColor has several instances and thus can be connected to several digital signals, and change between several colors.
DigBackground color makes it possible to modify the color of two parts of a subgraph or group independently of each other. In the example below, the gray rectangle is drawn with fixed gray color, the left circle has the fill color, and the right background color.
Drawing a circle with background color is achieved by setting the fill_eq_background property of the circle.

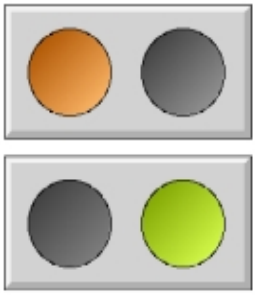


Fig Background color dynamics

When setting dynamic DigColor and DigBackgroundColor to the group, the DigColor dynamic will affect the left circle drawn with fill color, and the DigBackgroundColor dynamics will affect the right circle drawn with background color.

Attribute	Description
DigBackgroundColor.Attribute	Signal in the database of type Boolean the should influence the component.
DigBackgroundColor.Color	Background color of the component when the signal is 1.
DigBackgroundColor.Instances	Instance mask where instances can be added or removed.

8.2.12 TimeoutColor

Set specified color at subscription timeout.

Checks the last arrival for the subscription, and sets the fill color to the specifed color if the timeout time is exceeded.

TimeoutColor has to be combined with another dynamic specifying the attribute for the subscription. If there are several dynamics, the subscription for the dynamic with the highest priority is supervised.

Attribute	**Description**
TimeoutColor.Time	Timeout time in seconds.
TimeoutColor.Color	Fill color of the component at timeout.

8.2.13 DigTextColor

Set specified text color when the signal is high.

The object is connected to a digital signal in the database. If the value is 1, the object is drawn with DigTextColor.Color, if the value is 0, it is drawn with normal text color.

Attribute	Description
DigTextColor.Attribute	Signal in the database of type Boolean the should influence the component.
DigTextColor.Color	Text color of the component when the signal is 1.

8.2.14 DigText

Shift between two texts.

The object is connected to a digital signal in the database. If the value is 1, the original text is written, if the value is 0, the text in LowText is written.

With serveral instances, the object can be connected to further signals, and further texts can be displayed. For instance 2 and upwards, the specified HighText is written when the corresponding signal is high.

DigText operates on annotation 1.

Attribute	Description
DigText.Attribute	Signal in the database of type Boolean.
DigText.LowText	Text to write when the signal is low.
DigText.Instances	Instance mask where instances can be added or removed.
DigText2.Attribute	Signal in the database of type Boolean.
DigText2.HighText	Text to write when the signal is high.

8.2.15 AnalogText

Shift between several texts depending on the value of an analog signal.

The object is connected to a database attribute of type float or int. To each text you specify an Enum value. The text which Enum value matches the value of the database attribute, is displayed in the component.

Attribute	Description
AnalogText.Attribute	Database attribute of type Float32, Float64, Int32, UInt32, Int16, UInt16, Int8 or UInt8. The text which Enum matches the value of the database attribute is displayed.
AnaogText.TextMask	Mask that states number of texts.
AnalogText.Text1	Text number 1.
AnalogText.Enum1	Enumeration value for text number 1.
AnalogText.Text2	Text number 2.
AnalogText.Enum2	Enumeration value for the second text.

8.2.16 Value

Writes the value of a signal. The value is written in a text field.

The object is connected to an attribute in the database of arbitrary type. In format, the format of the conversion to text is specified, in c-syntax.

It is possible to create additional instances. This requres that the subgraph has several annotations to write out values in. The value of an instance are written out in the annotation with the corresponding number.

Attribute	Description
Value.Attribute	Signal in the database or arbitrary type, that is written out in the text field.
Value.Format	Format in c-printf syntax.
Value.ZeroBlank	When value is zero, the value field is blank.
Value.DecimalsAttr	Integer signal for number of decimals.
Value.DecimalsDecrease	The number of decimals will be decremented the specified amount.
Value.Instances	Instance mask where instances can be added or removed.

Format

For the following attributes types, some additional format strings are defined.

pwr_tTime

%t	Date and time. Eg 24-MAY-2007 11:33:43.91
%1t	Only time, no hundredth. Eg 11:33:43
%2t	Only time, with hundredth. Eg 11:33:43:91
%3t	Compressed date and time. Eg 07-05-24 11:33:43
%4t	Date only. Eg 24-MAY-2007
%5t	Compressed date. Eg 07-05-24
%6t	Time and date. Eg 01:00:00 30/01/87

pwr_tDeltaTime

%t	Time with hundredth. Ex 1:23:45.43
%1t	Time without hundredth. Ex 1:23:45

pwr_tEnum

%d	The enumeration value is displayed as in UInt32.
%s	The enumeration value will be converted to the corresponding string.

pwr_tObjid

%o	Object name (last segment).
%1o	Path and object name.
%2o	Volume, path and object name.

pwr_tAttrRef and pwr_tDataRef

%o	Object name (last segment) and attribute name.
%1o	Path, object name and attribute name.

%2o	Volume, path, object name and attribute name.

pwr_tStatus and pwr_tNetStatus

%m	The status value will be converted to the corresponding string. Eg "GDH-E-NOSUCHOBJ, No such object".
%1m	Only the text part of the string is displayed. Eg "No such object".

pwr_tMask

%d	The mask is displayed as an UInt32.
%b	The mask is displayed as a 32-bit binary value.
%16b	The mask is displayed as a 16-bit binary value. Eg 001111100011001.

8.2.17 Rotate

Rotate an object.

The rotation is made from the position the object has in the editor. The value of the signal is multipied by Factor, that gives the rotation in degrees.

x0 and y0 specifies the rotation point. If x0 and y0 is zero, origin in the coordinate system of the subgraph is used as rotation point. Subgraph that has Rotate as default dynamics, usually are drawn in a way that the rotation points doesn't has to be specified.

Attribute	**Description**
Rotate.Attribute	Signal in the database of type Float32. Supplies the rotation of the object.
Rotate.x0	x coordinate for the rotation point.
Rotate.y0	y coordinate for the rotation point.
Rotate.Factor	Factor that, multiplied by the value of the signal, gives the value in degrees.
Rotate.Offset	Angel offset value.
Rotate.MinAngle	Min value for angle.
Rotate.MaxAngle	Max value for angle.

8.2.18 Move

Moves and scales an object. Can be connected to four analog signals that moves the object in x and y direction, and scales it in x and y direction.

The movement is done from the position the object is given in the editor. The calculation from the values of the signals in XAttribute and YAttribute to Ge coordinates, are influenced by XOffset, YOffset and Factor. XOffset and YOffset are the values that corresponds to the starting point in x and y direction. Factor specifies the conversion from XAttribute and YAttribute values to Ge coordinates, according to the formula:

dx = (XAttribute - XOffset) * Factor

dy = (YAttribute - YOffset) * Factor

The scaling in x and y direction is done from the values of the signals in ScaleXAttribute and ScaleYAttribute. These are multiplied by ScaleFactor and gives the scale factor, where 1.0 implies no scaling, values greater than 1 gives enlargement, and values less then 1 gives reduced scale.

Attribute	**Description**
Move.XAttribute	Signal in the database of type Float32. Gives movement in x direction.
Move.YAttribute	Signal in the database or type Float32. Gives movement in y direction.
Move.XFactor	Conversion factor for XAttribute.
Move.YFactor	Conversion factor for YAttribute.
Move.XOffset	Value for XAttribute that corresponds to the start position for the object.
Move.YOffset	Value for YAttribute that corresponds to the start position for the object.
Move.ScaleXAttribute	Signal in the datbase of type Float32. Gives scaling in x direction.
Move.ScaleYAttribute	Signal in the database of type Float32. Gives scaling in y direction.
Move.ScaleXFactor	Conversion factor for ScaleXAttribute.
Move.ScaleYFactor	Conversion factor for ScaleYAttribute.
Move.ScaleType	Type of scale. From where the scale emanates, the center or on of the corners.

8.2.19 DigShift

Shift between two pages in the subgraph. The subgraph has to contain at least two pages.

The object is connected to a digital signal in the database. If the value is 0, the object is drawn with the first page, if the value is 1, it is drawn with the last page.

Attribute	Description
DigShift.Attribute	Signal in the database or type Boolean that shifts page. When the value is 0, the first page is displayed, when value is 1 the last page.

8.2.20 DigLowShift

Shift between two pages in the subgraph. The subgraph has to contain at least two pages. Equal to DigShift but will show last page on low signal instead of high.

The object is connected to a digital signal in the database. If the value is 0, the object is drawn with the last page, if the value is 1, it is drawn with the first page.

Attribute	**Description**
DigLowShift.Attribute	Signal in the database or type Boolean that shifts page. When the value is 0, the last page is displayed, when value is 1 the first page.

8.2.21 DigFourShift

Shift between four pages in the subgraph. The subgraph has to contain at least four pages.

The object is connected to three digital signals in the database.
If the value is 1 on Attribute1, the first page is drawn, if Attribute2 is 1 the second page is drawn, and if Attribute3 is 1 the third page is drawn. If the value is 0 on all signals, the last page is drawn. If several attributes it set, higher page number has higher priority.

Attribute	Description
DigFourShift.Attribute1	Signal in the database or type Boolean that shifts page. Activates drawing of the first page.
DigFourShift.Attribute2	Signal in the database or type Boolean that shifts page. Activates drawing of the second page.
DigFourShift.Attribute3	Signal in the database or type Boolean that shifts page. Activates drawing of the third page.

8.2.22 AnalogShift

Shift between several pages in the subgraph. The subgraph has to contain at least two pages.

The object is connected to an analog signal in the database. The value of the analog attribute determines the index for the page to display. Value 0 displays the first page, value 1 the second etc.

Attribute	Description
AnalogShift.Attribute	Signal in the database of type Float32 or Int32 that shifts page of the subgraphs. The value determines which page is to be displayed. Value 0 displays the first page.

8.2.23 Animation

Dynamic for animation subgraphs, i.e. subgraphs with several pages. The object is connected to a digital signal that activates the animation.

The animation sequence is determined by Sequence, that can have the value Cyclic, ForwBack or Dig.

Cyclic

Alter between resting and cyclic animation. The animation is done from page 2 to the last page, and continues on page 2 to the last page etc.

If the signal value is 0 the object is drawn with the first page, if the value is 1 the cyclic animation starts between the second and the last page. At animation the pages from second to the last page are run through, and after that you jump to the second page again, animating to the last page etc.

ForwBack

Shift between resting and cyclic animation. The animation runs forward and back between the first and the last page. If the signal value is 0 the object is drawn with th first page, if the value is 1, the cyclic animation starts forward and back between the first and the last page.

Dig

Shift between two resting positions. When the signal value is changed, an animation between the resting positions is performed.

If the signal value is 0, the object is drawn with the first page, if the value is 1, it is draw with the last page. When the values changes from 0 to 1 the animation is run from the first page to the last page. When the value changes from 1 to 0, the animation is run from the last page to the first page.

Attribute	Description
Animation.Attribute	Signal in the database of type Boolean that activates the animation.
Animation.Sequence	Animation sequence. Can be Cyclic, ForwBack or Dig.

8.2.24 Video

Displays an image file that is continuously updated by a web camera.
Also other image files can be used. When the file is modified, the image is redrawn.

Video has no attributes.

8.2.25 DigCommand

Execute specified command when the signal goes from low to high, or when the signal is high.

The object is connected to a digital signal in the database. The command is executed either on positive edge of the signal, or cyclically when the signal is high.

DigCommand can exist in several instances, which makes it possible to execute several commands.

Attribute	**Description**
DigCommand.Attribute	Signal in the database of type boolean that should influence the component.
DigCommand.Level	If Level is 0, the command is executed when the signal changes from 0 to 1. If Level is 1, the command is executed cyclically when the signal is high.
DigCommand.Command	Xtt command that will be executed.
DigCommand.Instances	Instance mask where instances can be added or removed.
DigCommand2.Attribute	Signal for instance number 2.
DigCommand2.Level	Command for instance number 2.
DigCommand2.Command	Command for instance number 2.

8.2.26 DigScript

Execute a script when the signal goes from low to high, or when the signal is high.

The object is connected to a digital signal in the database. The script is executed either on positive edge of the signal, or cyclically when the signal is high.

Attribute	Description
DigScript.Attribute	Signal in the database of type boolean that should influence the component.
DigScript.Arguments	Arguments that can be used as p1-p9 in the script. The arguments will be translated as a commmand (see action Command), for exa $object is replace by the current instance for an object graph.
DigScript.Level	If level is 0, the script is executed when the signal changes from 0 to 1. If level is 1, the script is executed cyclically when the signal is high.
DigScript.Script	Xtt script that will be executed.

8.2.27 ScrollingText

Display a scrolling text.

The object is connected to a string attribute in the database, and the text in the attribute is displayed.

Attribute	Description
ScrollingText.Attribute	Attribute in the database or type String.
ScrollingText.Direction	Scrolling direction, right, left, up or down.
ScrollingText.Speed	Scrolling speed.
ScrollingText.Bounce	The text will bounce at the borders.

8.2.28 DigSound

Play a sound when the signal goes from low to high, or when the signal is high.

The object is connected to a digital signal in the database. The sound is played either once on positive edge of the signal, or cyclically when the signal is high. Also a sound object of class Sound or SoundSequence has to be specified.

Attribute	Description
DigSound.Attribute	Signal in the database of type boolean that should influence the component.
DigSound.SoundObject	Sound object of class Sound or SoundSequence.
DigSound.Level	If level is 0, the sound is played when the signal changes from 0 to 1. If level is 1, the sound is played cyclically when the signal is high.
DigSound.Interval	Cycle time for cyclic play.
DigSound.Instances	Instance mask where instances can be added or removed.

8.2.29 DigTransparency

Shift transparency depending on a digital signal. When the signal is low, the transparancy is set to the value specified in LowValue, and when the signal is high, to the value in HighValue.

Attribute	Description
DigTransparency.Attribute	Signal in the database of type boolean that should influence the component.
DigTransparency.LowValue	Transparancy set when the signal is low. A value between 0 and 1.
DigTransparency.HighValue	Transparancy set when the signal is high. A value between 0 and 1.
DigTransparency.SmoothTransition	Gradual transition between low and high value.

8.2.30 AnalogTransparency

The transparency for en object is linked to an analog signal. The transparency depends on the value of the analog signal.

Attribute	Description
AnalogTransparency.Attribute	Signal in the database of type Float32.
AnalogTransparency.MinValue	Signal value that corresponds to minimal transparency.
AnalogTransparency.MaxValue	Signal value that corresponds to maximal transparency.

8.2.31 ColorThemeLightness

A color theme has a lightness property that is negative for dark themes and positive for light themes. The theme Midnight has for example lightness -5 while NordicLight has 1. Adding dynamic ColorThemeLightness to an object will increase or decrease the lightness of the object dependent on the colortheme lightness value.

The ColorThemeLightness dynamic has no properties.

8.2.32 DigSwap

DigSwap will disconnect all subscriptions in the graph and reconnect them again. This is useful when subscriptions by reference is used with the &() syntax, eg &(H1-Plate.Ref).Length##Float32. In this example H1-Plate.Ref is an attribute reference that points to an object containing the Length attribute. If the reference in H1-Plate.Ref is changed, the subscriptions has to be reconnected to the new reference, and this is done by DigSwap. DigSwap is connected to a digital signal that will activate the swap.

Attribute	Description
DigSwap.Attribute	Signal in the database of type Boolean that will trigger the swap.
DigSwap.ResetValue	When a trigger is detected the connected signal is reset by the DigSwap function.

8.2.33 UnitConvert

Unit conversion of a value displayed in a value field.
UnitConvert should be combined with dynamic Value and is implemented for attributes of type Float32. When combined with ValueInput, minimum and maximum input limits should be set in the converted unit.

Implemented entities and units

Entity	Units
Acceleration	m/s2, ft/s2, in/s2
Angle	Rad, degree, min, mrad, percent, sec
Area	m2, a, ac, cm2, ha, km2, mm2, sq in, sq mi, sq ft, sq yd
Energy	J, cal, kJ
Force	N, kN, kp, MN, p
Frequency	Hz, kHz, MHz, rad/min, rad/s, RPM
General	1:1, Tera, Giga, Mega, Kilo, Hecto, Deca, Deci, Centi, Milli, Micro, Nano, Pico
Length	m, cm, dm, ft, in, km, mi, mm, nm, NM, um, yd
Mass	kg, g, hg, kt, lb, mg, oz, st, t, ug
MassFlow	kg/s, g/min, g/s, kg/min, lb/min, lb/s, mg/min, oz/min, oz/s
Power	W, hp, GW, kW, mW, MW, TW, uW
Pressure	Pa, atm, b, kPa, lb/ft2, lb/in2, mb, mPa, mmHg, Mpa
Speed	m/s, ft/min, ft/s, in/min, in/s, km/h, kn, m/min, mm/s, mi/h, yd/min, yd/s
Temperature	K, C, F, R
Time	s, d, h, min, ms, ns, us, wk, y
Volume	m3, cm3, dm3, ft3, in3, mm3
VolumeFlow	m3/s, cl/s, in3/min, in3/s, ft3/min, ft3/s, l/s, m3/min, mm3/s, yd3/min, yd3/s

Attribute	Description
UnitConvert.Entity	Entity of value.
UnitConvert.DbUnit	Unit for value in the database.
UnitConvert.DisplayUnit	Unit for value in display.

8.2.34 HostObject

HostObject is a dynamic that is used for subgraphs that is connected to components. Components often contains several values that should affect the subgraph, and with HostObject it is possible to program a complex dynamic that only requires that the user makes one single connection to the component object.

Attribute	Description
HostObject.Object	Name of a database object.

The dynamic is programmed in graph attributes for the subgraph. Here subgraph is set to 1 and DynType1 is set to HostObject. HostObject.DynType1, HostObject.DynType2 and HostObject.Action can now be used to set dynamics on the subgraph. The components that will be connected to each intance is denoted '$hostobject'.

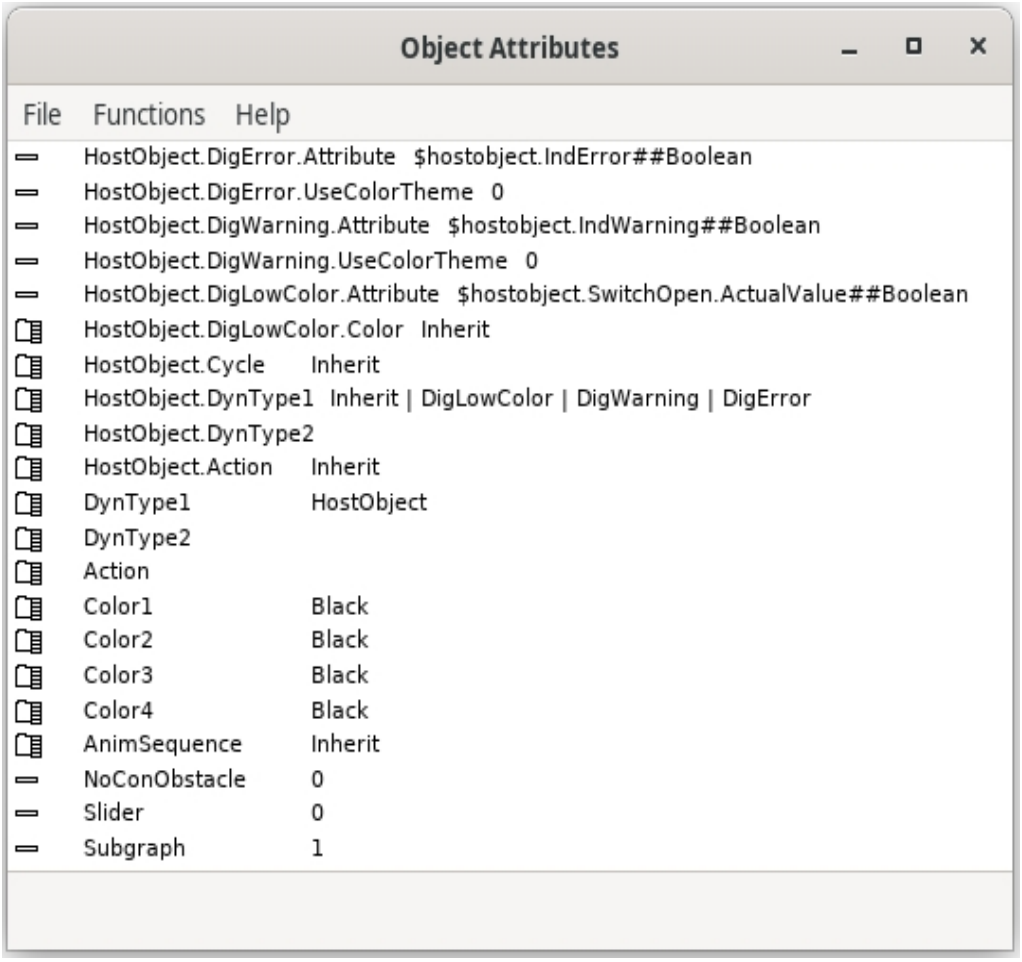


Fig Subgraph with default dynamics HostObject

Recursive dynamics
By default subgraphs inside other subgraphs can not contain dynamics, but by setting RecursiveDynamics in graph attributes for the subgraph to 1, it is possible to use other subgraphs as indicators and pushbuttons, and to use '$hostobject' to connect to the component object.

In the example below, the three indicators are subgraphs of type pwr_indsquare. They are

connected to attributes in the component object with $hostobject syntax.

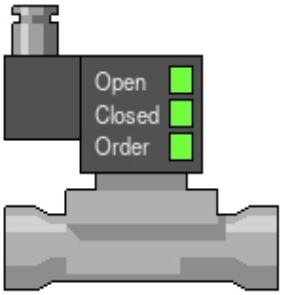


Fig Subgraph with recursive dynamics containing other subgraphs

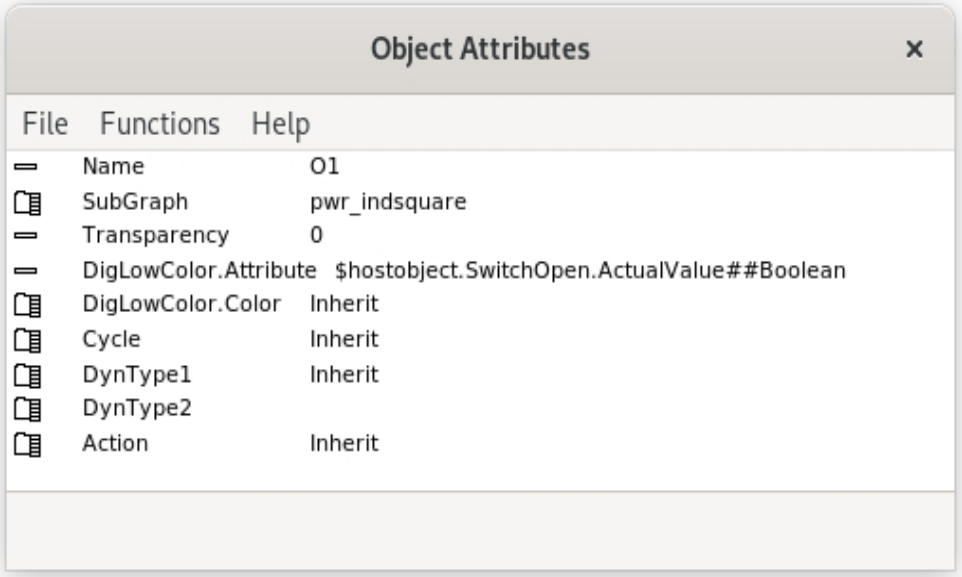


Fig Properties for indicator

8.3 Action

Action defines what happens when you activate an object, for example by clicking on it. You can set or reset signals in the database, open another graph, display a help text or a tooltip text etc.

Action	Signal type	Description
Inherit	-	Action for the instance is inherited from the subgraph.
PopupMenu	Object	A popup menu can be opened from the component by clicking MB3.
SetDig	Boolean	Set the value of a signal to true with Click MB1.
ResetDig	Boolean	Set the value of a signal to false with Click MB1.
ToggleDig	Boolean	Change the value of a signal with Click MB1.
StoDig	Boolean	Set the value of a signal as long as the button is pressed.
SetValue	Int, Float, String	Set the value of a signal to a specified value.
Command	-	Execute an xtt command with Click MB1.
CommandDoubleClick	-	Execute an xtt command with Doubleclick MB1.
Script	-	Execute an xtt script with Click MB1, or when graph is opened or c
Confirm	-	Confirm before the action is executed.
Help	-	Display a help text with Click MB1.
OpenGraph	-	Open a graph with Click MB1.
CloseGraph	-	Close the current graph.
OpenURL	-	Open an URL in a suitable web browser on Click MB1.
IncrAnalog	Float32	Increase or decrease an analog value.
RadioButton	Boolean	Action for radio buttons.
ValueInput	Arbitrary	Input field.
ToolTip	-	Display a tooltip text.
InputFocus	-	Set input focus.
OptionMenu	-	Action for option menus.
PulldownMenu	-	Action for pulldown menus.
MethodPulldownMenu	-	Action for pulldown menus for object methods.
EmitSignal	-	Emit a signal.
CatchSignal	-	Catch a signal.
ContextMenu	-	Open en popup menu.

8.3.1 PopupMenu

An popup menu is opened by clicking MB3. The popup menu contains the methods for the object specified in RefObject (Se chapter Methods in the Xtt manual).

Attribute	Description
PopupMenu.ReferenceObject	Object in the database, which methods are displayed and activated from the popup menu.

8.3.2 SetDig

Set the value of a digital signal to 1 by clicking MB1 on the object.

SetDig can exist in several instances, and then sets the value of several signals.

Attribute	Description
SetDig.Attribute	Signal in the database that is set to 1 with Click MB1 on the object.
SetDig.Instances	Instance mask where instances can be added or removed.

8.3.3 ResetDig

Set the value of a digital signal to 0 by clicking MB1 on the object.

ResetDig can exist in several instances, and then sets the value of several signals to 0.

Attribute	Description
ResetDig.Attribute	Signal in the database of type Boolean that is set to 0 with Click MB1 on the object.
ResetDig.Instances	Instance mask where instances can be added or removed.

8.3.4 ToggleDig

Toggle the value of a digital signal by clicking MB1 on the object.

Attribute	Description
ToggleDig.Attribute	Signal in the database of type Boolean that is toggled with Click MB1 on the object.

8.3.5 StoDig

Set the value of a digital signal to 1 for as long the the button is pressed.

Attribute	Description
StoDig.Attribute	Signal in the database of type Boolean that is set 1 as long as the button is pressed.

8.3.6 SetValue

Set the value of a signal to specified value.
By using several instances, values can be set into several attributes
with the same button.

Attribute	Description
SetValue.Attribute	Signal in the database of type Float32, Int32, UInt32, Int16, UInt16, Int8, UInt8 or String, that it set with Click MB1 on the object.
SetValue.Value	The value that is set.
SetValue.Instances	Instance mask where instances can be added or removed.

8.3.7 Command

Execute an xtt command by clicking MB1 on the object.

Attribute	Description
Command.Command	Xtt command that is executed when the object is activated.

Additions to the xtt command syntax

$object

$object will be replaced by the current object in an object graph or by the stated hierarchy in an hierarchy graph.

$hostobject

A special syntax for HostObject dynamics where $hostobject will be replaced by the current host object.

&(attribute-reference)

The string for an attribute reference of type &(attribute-reference) will be replaced by the content of the attribute reference. Suppose there is an array, H1-ObjectList.ObjArray, with attriute references that points to objects, which object graphs is going to be shown. With the command

```
> open graph /class /instance=&(H1-ObjectList.ObjArray[0])
```

the object graph is opened for the object in the first element. It is also possible to state double steps of attribute references with the syntax '&(&(attribute-reference))'.

Array index replacement
An Int32 or UInt32 attribute can be used as array index with the syntax [&(integer-attribute)]. With the command

```
> open graph /class /instance=H1-Plates[&(H1-CurrentIndex.ActualValue)]
```

the index array of H1-Plates will be fetched from the current value of H1-CurrentIndex.ActualValue.

Parent object references
The .< syntax will remove the preceding attribute in an attribute name. H1-Plates.Data.<.CurrentIndex will result in H1-Plates.CurrentIndex. This can be used in object graphs in combination with $object. For example

```
> open graph /class /instance=$object.<
```

will open the object graph for the object the contains the current object.

The -< syntax is similar but works for object hierarchies. H1-Pumps-P1<- will result in H1-Pumps. This makes it possible to address parent objects in hierarchy graphs. For example

```
> open graph /class /instance=$object-<
```

will open the object graph for the parent object.

8.3.8 CommandDoubleClick

Execute an xtt command by doubleclicking MB1 on the object.

Attribute	Description
CommandDoubleClick.Command	Xtt command that is executed when the object is activated.

8.3.9 Script

Execute an xtt script by clicking MB1 on the objectj, or when the graph is opened or closed.

Attribute	Description
Script.TriggerEvent	Event that trigger execution of the script. ClickMB1, when MB1 is clicked. Open, when the graph is opened. Close, when the graph is closed.
Script.Arguments	Arguments that can be used as p1-p9 in the script. The arguments will be translated as a commmand (see action Command), for exa $object is replace by the current instance for an object graph.
Script.Script	Xtt script that is executed when the object is activated.

Scripts can use extern variables to store status values and for interaction between scripts. The extern variables can be declared in a script triggered by the Open event, and deleted in a script triggered by the Close event.

Extern variables can also be used in object graphs, where several object graphs can be opened concurrently. Mix-up of the variables is avoided by setting the namespace for extern variables to the instance object name. Supply the instance object as argument and set the namespace with set_namespace(p1).

8.3.10 Confirm

Confirm requires a confirmation from the user before an action is executed. A dialog box with specified text, Ok and Cancel buttons, is displayed before any other action element is executed.

Attribute	Desciption
Confirm.Text	Text displayed in the confirm window.
Confirm.OnSet	Used with action ToggleDig when only set of value to 1 should be Set of value to 0 is executed without confirm.
Confirm.OnReset	Used with action ToggleDig when only set of value to 0 should be Set of value to 1 is executed without confirm.

8.3.11 Help

Open the help window with specified topic by clicking MB1 on the object.

Attribute	Description
Help.Topic	Help text topic that is to be displayed.
Help.Bookmark	Bookmark in the help text. Optional.

8.3.12 OpenGraph

Open another graph via a XttGraph object by clicking MB1 on the object.

Attribute	Description
OpenGraph.GraphObject	Object of class XttGraph.

8.3.13 CloseGraph

Close the current graph.

CloseGraph has no attributes.

8.3.14 OpenURL

Open an URL in a suitable webbrowser by clicking MB1 on the object.

Attribute	Description
OpenURL.URL	An URL

8.3.15 IncrAnalog

Increase or decrease the value of an analog signal.

The object is connected to an analog signal. When clicking MB1 on the object the value is incremented by the value specified in Increment. Minimum and maximum values for the signal value can be specified.

Attribute	Description
IncrAnalog.Attribute	Signal in the database of type Float32. When the object is activated, the value of the signal is incremented by the value in Increment.
IncrAnalog.Increment	Value that the signal value is incremented by.
IncrAnalog.MinValue	Minimum value for the signal.
IncrAnalog.MaxValue	Maximum value for the signal.

8.3.16 RadioButton

Sets the value of a digital signal when the button is activated, and resets the value for the other RadioButton objects in the group.

The object has to be a member of a group with other objects of type RadioButton.

Attribute	Description
RadioButton.Attribute	Signal in the database of type Boolean. With Click MB1 the value of the signal is set, and the other RadioButton objects in the same group are reset.

8.3.17 ValueInput

Input field. Makes it possible to enter the value of a signal, in a component with Value dynamics specified. The value is changed by clicking MB1 on the object.

The signal is specified in Value.Attribute.

Attribute	Description
ValueInput.MinValue	Minimum input value.
ValueInput.MaxValue	Maximum input value.
ValueInput.Clear	The field is cleared when opened for input.
ValueInput.Popup	Input is done in a popup box.
ValueInput.Unselect	The text in the field is not selected when the field is opened for input.
ValueInput.EscapeStore	Normally a value is stored into database when Enter is pressed, but with EscapeStore, the storage is also done when when input focus is lost. I will also be stored by the xtt command 'set graph /escapestore'.
ValueInput.MinValueAttr	Signal for minimum input value.
ValueInput.MaxValueAttr	Signal for maximum input value.
ValueInput.KeyboardType	Type of virtual keyboard that is opened when the field has input focus. Requires that VirtualKeyboards is enbled for the operator place.
ValueInput.UpdateOpen	Update the field if the database value is changed also when the field is open.

8.3.18 ToolTip

Views a ToolTip text for a component. ToolTip is a white text box, that is displayed when the cursor is resting on a component for a certain time. When the cursor leaves the component, the textbox is removed. In 'Text' the tooltip text is usually specified. If the text is to be modified, you can connect it to a string attribute by specifying the string attribute in 'Text' with the prefix '&', for example

```
&A1-B1- Sv1.ActualValue##String80
```

Attribute	Description
ToolTip.Text	ToolTip text.

8.3.19 InputFocus

Input focus is a function that makes it possible to influence components in the graph from the keyboard instead using the mouse. Most functions executed with cursor and mouse can also be performed with the keyboard.

A component is selected with the arrow and Tab keys (or by clicking on it), to receive input focus.
This means that all input from the keyboard are directed to the component. A pushbutton is, for example activated by pressing Return, an option menu by selecting the desired alternative with the arrow keys and pressing Return, a ValueInput by entering desired value and pressing Return.

Only components with the action type InputFocus can receive input focus, the exception is ValueInput that also can receive input focus by clicking on it.

You shift input focus between components with the arrow keys or the tab key. The order of the components are configured by linking them in three lists, one horizontal, one vertical and one Tab list. In the horizontal list you move with the ArrowLeft and ArrowRight keys, in the vertical with the ArrowUp and ArrowDown, and in the Tab list with the Tab key. This is configured with the attributes InitialFocus, NextHorizontal, NextVertical and NextTab. In InitialFocus you can state that an object is first or last in a list. You can also state that a component should receive input focus when the graph is opened. In NextHorizontal, NextVertical and NextTab you specify the object name of the next component in the lists.

You use to divide the components in Tab groups. Between the objects within a Tab group you navigate with the arrow keys, and between Tab groups you navigate with the Tab key. This is achieved by setting NextTab for all the objects in one Tab group, to the first object in the next Tab group.

Attribute	**Description**
InputFocus.InitalFocus	Bitmask with the following bits: - InitialFocus The component has input focus when the graph is opened. - FirstHorizontal The object is the first object in the horizontal list. - FirstVertical The object is the first object in the vertical list. - FirstTab The object is the first object in the Tab list. - LastHorizontal The component is the last object in the horizontal list. - LastVertical The component is the last object in the vertical list.
InputFocus.NextHorizontal	Object name of the next component in the horizontal list.
InputFocus.NextVertical	Object name of the next component in the vertical list.
InputFocus.NextTab	Object name of the next component in the Tab list.

8.3.20 PulldownMenu

Action for a pulldown menu, or for a submenu in a pulldown menu.

A pulldown menu consists of a number of menu alternatives that either are pushbuttons or submenus.
The number of alternatives in the menu is configured in ItemMask, a menu can contain up to 32 menu alternatives. For each alternative you choose an action type. For pushbuttons you can choose between SetDig, ResetDig, ToggleDig, Command, Help, OpenGraph, CloseGraph, OpenURL and IncrAnalog. Submenus should have action type PulldownMenu, and for this you configure the number of menu alternatives and the actiontypes in the same way as for a pulldown menu. Submenus can be built in an unlimited number of levels.

Attribute	**Description**
Text	Pulldown menu text.
PulldownMenu.ItemMask	Mask that states the number of menu alternatives.
PulldownMenu.ItemText1	Text of the first menu alternative.
PulldownMenu.ItemDyn1	Dynamic for the first menu alternative. Contains action type, and possible access, for the alternative.
PulldownMenu.ItemText2	Text of the second menu alternative.
PulldownMenu.ItemDyn2	Dynamic for the second menu alternative.

8.3.21 OptionMenu

Option menu is menu where you select an alternative in a list of alternatives. The chosen alternative is displayed in the menu component. When you click on the component, the list of alternatives is displayed. When an alternative is chosen, the list is closed, and the selected alternative is displayed in the component.

The option menu is connect to a database attribute of type float or int. Each alternative in the list corresponds to an enumeration value, and when an alternative is selected, the value is inserted into the database attribute. The component is continuously reading the value of the databas attribute, and displays the corresponding text in the text field.

The option men can be static or dynamic. For a static meny you state the texts and enumeration values in different items. For a dynamic meny the meny text are fetched from a database attribute of type array of String80. The number of texts in the menu is fetched from an attribute of type Int32, and further more you should state an attribute that indicates that new texts should be loaded. This attribute should be of type Boolean and the new texts are loaded when the value is changed from 0 to 1.

Attribute	**Description**
OptionMenu.Type	Type of option menu. Static or dynamic.
Static menu	
OptionMenu.Attribute	Database attribute of type Float32, Float64, Int32, UInt32, Int16, UInt16, Int8 or UInt8. The value in ItemEnum for a selected alternative inserted into the database attribute.
OptionMenu.ItemMask	Mask that states the number of menu alternatives.
OptionMenu.ItemText1	Text of the first menu alternative.
OptionMenu.ItemEnum1	Enumeration value of the first menu alternative.
OptionMenu.ItemText2	Text of the second menu alternative.
OptionMenu.ItemEnum2	Enumeration value of the second menu alternative.
Dynamic menu	
OptionMenu.Attribute	Database attribute of type Float32, Float64, Int32, UInt32, Int16, UInt16, Int8 or UInt8. The value in ItemEnum for a selected alternative inserted into the database attribute.
OptionMenu.TextAttribute	Database attribute of type array of String80. The array contains the texts that are displayed in the menu.
OptionMenu.SizeAttribute	Database attribute of type Int32. States the number of alternative in the menu.
OptionMenu.UpdateAttribute	Database attribute of type Boolean. Should be set to 1 when new texts are present in the TextAttribute. New texts are loaded when the value is changed from 0 to 1.

8.3.22 MethodPulldownMenu

Action for a method pulldown menu, with menu alternatives for the methods of an object.
Only the configured methods are displayed in the menu.

Attribute	Description
MethodPulldownMenu.Object	Object which methods are diplayed in the menu.
MethodPulldownMenu.MenuType	Type of menu. Object, Help or Simulate.

8.3.23 EmitSignal

Action to emit a signal. The signal is emitted on Click MB1.

Attribute	Description
EmitSignal.SignalName	Signal name.
EmitSignal.Global	If global is set, the signal is emitted to all open graphs and multiviews, Otherwise only to the current graph.

8.3.24 CatchSignal

Action to catch a signal. When the signal is caught, the click actions for the current object is executed.

Attribute	Description
CatchSignal.SignalName	Signal name.

8.3.25 ContextMenu

An popup menu is opened by clicking MB3. The popup menu can contain up to 10 menu items. For each item an Xtt command can be specified that will be executed when the item is activated.

Attribute	Description
ContextMenu.ItemText[0]	Text of first menu item.
ContextMenu.ItemAction[0]	Xtt command to execute when the menu item is activated.
ContextMenu.ItemText[1]	Text of second menu item.
ContextMenu.ItemAction[1]	Xtt command to execute when the menu item is activated.
...	

8.4 Create a subgraph

You create a subgraph by drawing base objects in the working area. Here you also have access to connection points and annotations.

Mark that the graph is a subgraph by opening 'File/Graph attributes' and set 'Subgraph' to 1.

Attribute	Description
DynType	Type of dynamic. The instances will inherit this as their default dynamic.
Action	Type of action. The instances will inherit this as their default action.
Color1	The first color that the instances will inherit as default.
Color2	The second color that the instances will inherit as default.
Color3	The third color that the instances will inherit as default.
Color4	The fourth color that the instances will inherit as default.
AnimSequence	Type of animation when DynType is Animation.
NoConObstacle	Indicates that connections should ignore instances of this subgraph. Connections of type 'routed' normally avoids subgraph objects, though if NoConObstacle is set, connections can cross the object.
Slider	Indicates that the subgraph is a slider.
Subgraph	Should be 1 for a subgraph.
AnimationCount	Used for animations. States the number of cycles the animation stays at this page.
JavaName	When exported as a java bean, the subgraph will be exported as java class with this name.
Cycle	Default value for cycle, slow or fast, that instances will inherit as default.
x0	In (x0, y0) and (x1, y1) you can state the max size of a subgraph in the case where the subgraph has pages with different extension.
y0	See x0. Also used by some subgraphs with DynType Slider, Sliderbackground and FillLevel.
x1	See x0.
y1	See x0. Also used by some subgraphs with DynType SliderBackground and FillLevel.
InputFocusMark	How objects are marked when they have input focus.
RecursiveDynamic	Used for dynamic HostObject where objects in the subgraph have individual dynamics.
Dynamic	Not implemented.

Save the subgraph with 'File/SaveAs...'. Copy the .pwsg file that is created to $pwrp_exe. The subgraph should now be visible under the Local/Subgraphs folder in the subgraph palette.

Its advisable to draw subgraphs close to origin in the working area, because when creating a subgraph, the position of the mouse click corresponds to origin. Connection points should be positioned on grid points, if rectangular connections are used to connect the instances. You then avoid notches in the connections lines.

External och internal subgraphs

When you for the first time in a graph create an instance of a certain subgraph, this is loaded from the .pwsg-file. As default it is internal, which means that the subgraph is saved together with the graph, and the .pwsg-file is not needed any more, i.e. you don't need to concern yourself with copying it to operator and process nodes. This will work, until you need to change the subgraph. As long as the subgraph is internal, the changes will not come along.

To get in the changes, the subgraph has to be external. When the graph then is saved, the subgraph will not be saved together with the graph, and when reading the graph next time, the subgraph is loaded from the .pwsg-file. When the changed version is loaded, you can again reset to internal.

To rearrange a subgraph from internal to external and vice versa, you activate 'Loaded Subgraphs' under 'File' in the menu. Here you get a list of all loaded subgraph and can set them as external or internal.

Slider

A slider is a specific type of subgraph. A slider is moved horizontally or vertically between two endpoints, and the position is transferred, after conversion, to an analog value in the database.

With background

The easiest way to configure a slider, is to put a SliderBackground object behind the slider. Then, the only thing you have to do, is to connect the slider to an analog signal.

Without background

If you absolutely don't want to have any background to the slider, you have to state the direction, and minimum and maximum position for the movement. A horizontal slider is configured in the following way (vertical configuration inside parentesis).

Position the slider on the lowest y (x) coordinate for the slider movement. Remember the origin is positioned in the upper left corner.
Measure the minimum and maximum values for the movement of the slider, relative to the upper (left) side of the slider by placing the cursor there. The position of the cursor is present in the lower right corner of the window. Enter these values in the attributes MaxPosition and MinPosition from the object editor for the object. State also the direction, Up indicates a vertical slider with increasing value upwards, Right a horizontal slider with increasing value to the right etc.

Attribute	**Description**
Attribute	Signal in the database of type Float32 or Int32 that is changed when the slider is dragged.
Access	Privileges required to change the value of the signal.
Direction	Only has to be stated if there is no SliderBackground object. Direction of the slider. Up implies a vertical slider with increasing value upwards, Right a horizontal slider with increasing value to the right.
MaxValue	Value of the signal that corresponds to the max position of the slider.
MinValue	Value of the signal that corresponds to the min position of the slider.
MaxPosition	Only has to be stated if there is no SliderBackground object. x or y coordinate for the max position of the slider (see above).
MinPosition	Only has to be stated if there is no SliderBackground object. x or y coordinate for the min position of the slider (see above).

9 Groups

A group is a number of objects that appear and behave as a single object when moved, scaled, rotated etc. A group is created by selecting the object that should belong to the group, and activating 'Functions/Group' in the menu. If you want to resolve a group, you select the group and activate 'Functions/Ungroup' in the menu.

There are some limitations in objects that can be a part of a group, connections for example are not allowed in groups, and sliders will loose their sensitivity.

Dynamics

One property for groups is that you can specify dynamic to them. All types of dynamics and actions available for subgraphs, are also available for groups.

The types of dynamic that includes a change of color or color tone, will affect all the objects in the group that has no dynamic of its own. Subgraphs that should follow the color of the group, should have No as dynamic type (xtt and java here work a bit different).
Subgraphs and subgroups within the group that has its own dynamic, will role over the color itself.

When you resolve a group, that has some kind of dynamic type, the data of the dynamic is lost. To simplify the situations when you temporary resolve a group to do some modifications of a group member, there is a function that tries to recover the dynamic. When you resolve the group, the dynamic data is saved in a recall buffer with the group name as key. Furthermore the group name is also saved in all the members of the group. When you later regroup the objects, you look for the group that the majority of the objects has been a member of, and try to find the dynamic for this group in the recall buffer. If it is found, it is inserted as dynamic for the new group. There are cases when the restoring of dynamic doesn't work, so it is advisable to check that the dynamic is correctly recovered after having temporary dissolved of the group. If something has gone wrong, you can usually find the dynamics in the recall buffer, that is handled from the object editor with 'Functions/Recall' and 'Functions/Recall previous'.

10 Images

An image is an object that reads an image from an image file of type png, jpg or gif.

Create an image

You create an image in the same way as a subgraph. png, jpg and gif images are found under the 'Images' folder, and under the 'Local/Images' folder. If you want to read a specific image file you copy it to $pwrp_pop. The file is then found under the 'Local/Images' folder. By selecting the image and clicking MB2 in the work area you create the image object.

An image can be scaled and rotated in steps of 90 degrees. Functions for color tone, lightness, intensity and color shift also applies to images.

Dynamics

To put dynamics behind an image, you first have to make a subgraph of the image, or let it be a member of a group. If you want to change the color dynamically, you should use a dynamic type that changes the color tone, e.g. DigTone. Change of fill color of an image will have no effect.

11 Connections

The subgraph objects that contains connection points can be connected with connections. Connections are drawn by dragging MB2 from one object to the other. The connection point that is closest to the cursor when you press or release the mouse is selected.

Connections are characterized by type, line width, 3D and color.

Type

Type is selected from the menu under 'Connections'.

Straight
A straight line connection between the connection points.

StraightOneArrow
A straight line connection with one arrow.

Routed
Connection with horizontal and vertical lines. The route and breakpoints of the connection are caculated to avoid collision with other subgraph objects (that don't has the attribute NoConObstacle set).

Grafcet
There are a number of connections used when drawing grafcet sequences: StepDivergence, StepConvergence, TransDivergence and TransConvergence. These can be used, together with the subgraphs under the folder Grafcet in the subgraph palette, if you want to view a grafcet sequence in a graph.

Round corners

For connections of type routed, you can choose to have rounded corners, and you can also specify the radius of the rounding. This is chosen in the menu, 'Connections/Corners' and 'Connections/CornerRoundAmount'.

Color

The color of the connection is selected as fill color, that is with MB1 in the color palette.

Border

With 'Border', the connections is draw with black borderlines.

3D

With '3D', a lighter shadow is drawn on the upper side, and a darker on the lower side. Only implemented for connections of type Routed and round corners.

Width

The width is chosen in the tool panel in the same way as for lines.

Ramification

If a connection is drawn from a component and released in the working area, a component of type ConGlue is created at the end of the connection. ConGlue has four connection points and works either as a termination, or as a ramification of the connection, or as a possibility to influence the route of the connection. By drawing new connections from ConGlue, new connections are created, and the ConGlue object adapts its color and shape after its connected connections.

ConGlue is also found in the subgraph palette under 'Other'.

12 Editing

Ge contains a number of functions to create objects, modify them, order them etc. The functions are found in the tool panel or in the menu, and som are activated with mouse clicks.

Create objects

Base objects are created by choosing an object type in the tool panel, and then dragging or clicking in the working area with MB1. If you press the Shift key while the object type is chosen, you can create several objects without choosing object type again. A more detailed description of how to create each object type is found in the chapter Object.

An instance of a subgraph or a complex object, is created by selecting the object in the subgraph palette an clicking MB1 in the working area.

Create connections

Connections are created by dragging MB2 between two subgraph object that contains connection points.
First you set the proper connection type from 'Connections' in the menu.

Reset

By clicking with MB3 you reset or terminate most functions: clear the list of selected objects, end drawing of a polyline, end scaling, end polyline editing.

Select objects

An object or connection that is selected is marked with red color. This is not valid when you change the color of the selected objects. Then the object is drawn with the new color so that the effect of the color change can be examined.

Objects are selected in the following ways

- Click on the object with MB1. Previously selected objects are removed from the select list. If the object already is selected, the select list is cleared.
- Click on the object with Shift MB1. The object is added to the select list. If the object already is selected it is removed from the select list.
- Drag with MB1. A rectangle is drawn and objects positioned totally inside the rectangle becomes selected. Previously selected objects are remove from the select list. Note that as you also move objects with drag MB1, you should not hit any object when starting to drag. If there is an object covering the background, you can use Shift MB1 instead (se below).
- Drag with Shift MB1. Object inside the rectangle are added to the list of selected objects.
- 'Select all objects' under 'Edit' in the menu selects all objects.
- 'Select all connections' selects all connections.

The list of selected objects is cleared by clicking MB1 in an empty space in the working area or by clicking MB3 in the working area.

Move objects

An object is moved by MB1. If you want to move several objects simultaneously, you select the objects. If you now move one of the selected objects the other selected objects will follow.

If you want to move objects vertically or horizontally you first select 'Move Restrictions' under 'Edit' in the menu. Move Restrictions are reset by MB3.

Cut, copy and paste

You copy objects by selecting the objects that are to be copied, and copy them to the paste buffer with Ctrl+C. With Ctrl+V the objects are copied to the working area, and follows the movement of the cursor until you click MB1 to fasten them. Cut works in the same way with Ctrl+X.

Grid

Grid and grid size are chosen from the tool panel.

Scaling

The scaling function is activated from the tool panel and affects selected objects. The selected objects are framed by a rectangle. By dragging a corner or side of the rectangle, the selected objects are scaled. If you drag a side, the scaling is vertical or horizontal, if you drag a corner the scaling is unlimited. By activating 'Functions/ScaleEqual' you get a conform scaling in x and y direction.

Rotate

Selected objects are rotated 90 degrees clockwise when the rotate function in the tool panel is activated. The 90 degrees rotation works för all objects except texts. If you want to totate with another angle, this is done from 'Edit/Rotate' in the menu. Here you can enter an arbitrary angle. Note that only lines, polylines and circles can handle arbitrary rotations.
If you want subgraphs to be able to handle arbitrary rotation, you have to build them with lines and polylines.

Order objects

Vertically

Objects that are placed vertically can be left aligned, right aligned or centered, by using 'Align/vertical' under 'Functions' in the menu. Select the object that are to be aligned and activate the menu entry.

Horizontally

Objects that are placed horizontally can be aligned regarding the upper side, the lower side or the center, by using 'Align/horizontal' under 'Functions' in the menu. Select the objects that is to be aligned and activate the menu entry.

Concentric circles

Concentric circles are achieved by first activating 'Align/vertical/center' and then 'Align/horizontal/center'.

Equal distance between objects

Objects that are placed vertically or horizontally, often should have the same distance between them. This is achieved by 'Equidistance' under 'Functions' in the menu. The Equidistance function for horizontal objects, keeps the position of the leftmost and rightmost objects, and adjusts the position of the objects between, to get the same distance between them. The distance is measured from the left border, right border or the center. In a similar way vertically objects are ordered with 'Equidistance/Vertical'.

Over or under

If objects overlap, you can move them over or under with 'Pop' and 'Push' under 'Functions' in the menu. Pop puts an object on top of all other objects in the graph, and Push puts the object beneath all other objects. If you want to put an object between two objects, some thinking is required to activate the pop and push in the correct order.

Colors

Background color

The background color in the graph is set by selecting a suitable fill color in the color palette, and then activating 'Set background color' under 'Functions' in the menu.

Object colors

Object colors are divided in fill color and border color. The filled part of the objects is drawn with the fill color and the border is drawn with the border color. Some objects, for example lines, are only drawn with border colors. The colors are selected in the color palette, fill color is selected with MB1 and border color with MB2. The currently selected colors are displayed in rectangles topmost in the palette, fill color to the left and border color in the middle. When base objects are created, the object are given the currently selected colors in the palette. If you want to change the color of an existing object, you select it and select a new fill or border color in the palette. Note that the objects after this operation are not drawn with red color, though they are selected. For that reason you should always clear the select list by clicking MB3 after the operation.

Subgraph object colors

You can change the color of a subgraph object in the same way as you change the color of a base object, by selecting them, and state a border or fill color. Object that is drawn with shadows and 3D effects might loose these effects if you change the fill color. There are special functions to change the color of subgraph objects, color tone, lightness, intensity and color shift. In the chapter Subgraphs this is described in more detail.

Connection to the database

When a subgraph object is to influence or be influenced by a signal in the database, you have to state this in the attributes of the subgraph. One way is to enter the signal name by hand in the object editor. Another and smoother way is to use the plant hierarchy that is found over the navigation window. In the plant hierarch you select the object you want to connect the subgraph object to, and then you click on the subgraph object with Ctrl/Doubleclick MB1. If there are two database attributes in the subgraph object, the first is connected in this way, and the second with Ctrl/Shift/Doubleclick MB1. You can also open the object editor and click on the attribute row with Ctrl/Doubleclick MB1 to insert the signal. If you are connecting a signal it is enough to select the object in the plant hierarchy, the attribute ActualValue is selected automatically. If it is another type of object, you have to state which attribute to connect to the subgraph object. This is done by opening the object with Doubleclick MB1, and the select the desired attribute.

You can also select the object in the configurator.

Suffix

To the signal name, a suffix should be added, that tells the type of the attribute. Some common types are ##Boolean, ##Float32, ##Int32, ##UInt32 and ##String80.

Graph attributes

Under File in the menu you find 'Graph attributes' and here you can enter attributes for the graph.

Here you state which part of the working area that is to be displayed in runtime. The coordinates for the upper left corner and lower right corner are measured with the cursor and entered in x0, y0 and x1, y1 respectively.

Attribute	Description
subgraph	States that the graph is a subgraph. In this case the attributes below are not valid.
x0	x-coordinate for the upper left corner of the graph.
y0	y-coordinate for the upper left corner of the graph.
x1	x-coordinate for the lower right corner of the graph.
y1	y-coordinate for the lower right corner of the graph.
Scantime	Cycle time in seconds for update objects that runs with the slow cycle.
FastScantime	Cycle time in seconds for update objects that runs with the fast cycle.
AnmationScantime	Cycle time in second for animations in the graph.
BackgroundImage	Name of a gif or jpeg image for the background image.
MB3Action	Action for right click in the graph.
Translate	All dynamic and static texts are translated.
BitmapFonts	Bitmap fonts are used instead of scalalbe fonts.
HotIndication	How sensitive objects are marked when they are hot.
TooltipTextsize	Size of text in tooltip.
ColorTheme	Color theme file. '$default' denotes the colortheme choosen by the operator.
WindowResize	-
Dashboard	Graph is a dashboard.

Object and hierarchy graphs

This chapter describes how to create Ge graphs that displays the content of an object of a certain class, and is able to display all object of that class. It also describes how you can construct common graphs for similar hierarchies in the ProviewR database.

Object graph

For some classes that belongs to ProviewR baseclasses, there are so called object graphs, i.e. graphs that can be opened for each instance of the class. Object graphs exist for example for the classes PID, Mode, Av, Ai, Ao, Di, Do and Dv, and is opened with the command 'open graph/instance=' where you in /instance supply the name of the object. You can also open the object graph by selecting the object and activate 'Functions/Open object graph' in the xtt menu.

If you have created a user class, you can draw an object graph for this class. You edit the graph as a normal ge graph. The difference is when you are about to connect the dynamic objects to database objects. Where you normally write an object name, you instead write '$object'. Data for an indicator connected to the attribute ActualValue might look like this.

Attribute	**Value**
SubGraph	pwr_indsquare
LowColor.Attribute	$object.ActualValue##Boolean
LowColor.Color	Inherit
Cycle	Inherit
DynType	Inherit
Action	Inherit

The graph is saved with the same name as the class, but with lower case.

The command to open the object graph for the object H1-H2-MyObject of class MyClass is

```
open graph /classgraph /instance=H1-H2-MyObject
```

Hierarchy graph

Often there are plant parts in a system that are identical and that are configured with nearly identical hierarchies. You can for example have 30 identical frequency converters, and there is an possibility to, instead of drawing 30 identical graphs, draw one generic graph that can display all the frequency converters. The procedure is similar to object graphs above. The hierarchy that is supplied in /instance in the open graph command when the graph is opened, will replace all occurrences of $object in dynamic connections in the graph.

Data for an indicator that is connected to the Dv Start can look like this

Attribute	**Value**
SubGraph	pwr_indsquare
LowColor.Attribute	$object-Start.ActualValue##Boolean
LowColor.Color	Inherit
Cycle	Inherit
DynType	Inherit
Action	Inherit

Then the picture is opened with the command

```
open graph my_fo_graph /instance=H1-H2-Fo1.53
```

the indicator is connected to the attribute H1-H2-Fo1-Start.ActualValue.

Also in commands that is executed by pushbuttons in the graph, the string $object will be replaced by the hierarchy name. This makes it possible to open object graph or trends of objects within the hierarchy from pushbuttons. The command to open the object graph for a Mode object could be

```
open graph /classgraph /instance=$object-Plc-W-Mode0/name=$object
```

Example of a command to open a trend from a plotgroup in the hierarchy

```
open trend $object-Plot
```

13 Dashboard

The dashboard is a simplified graph with a very limited number of building blocks. The building blocks are called dashcells that can display values in the shape of indicators, value fields, bars, trends, gauges etc.

The dashboard can be created in the Ge editor, or in the operator environment. The building blocks and how to create a dashbord in the operator environment is described in Operator's Guide.

In the Ge editor, a dashbord is created in the following way.

- Open Graph Attributes and set Dashboard to 1.
- Open Graph Attributes again and set the size of the dashboard in DashRows and DashColumns.
- Dash cells are drawn with colortheme colors, thus select a colortheme from File/Colortheme/Select in the menu.
- Set the background color to the first color in the custom color palette.
- If the dashboard should follow the color theme selected by the operator, set ColorTheme in Graph Attributes to $default.
- Create a DashCell from Other/DashCell in the subgraph palette.
- Double click on the cell to set properties and size, and connect the cell to a signal.
- Save and build the dashboard.

The dashboard is saved in a pwd file on $pwrp_pop, that is copied to $pwrp_exe when it's built. This file should be distributed to the operator station.

The dashboard can be opened from Functions/Dashboard/Open i the Xtt menu, or with the command 'open dashboard'.

14 Testing

Xtt

By starting xtt in the development environment, you can easily test the graph. The only requirement is that the .pwg file for the graph is present on $pwrp_exe. Use the build button in the Ge editor tools panel to copy the graph to $pwrp_exe.

A graph is opened with the command

```
xtt> open graph 'graphname' [/width=][/height=][/scrollbar][/navigator]
```

All the available graphs are listed with command

```
xtt> show graph
```

A graph can be opened from the list by clicking on the graph icon, or selecting a graph and pressing key arrow right.

Preview

There is a function to test the graph immediately in the editor. One requirement is that you have to start the ProviewR runtime environment on the development station, to be able to find signals that occurs in the graph. From 'View/Preview' in the menu, the editor connects to the database and starts executing the dynamic of the graph. Preview is closed with 'View/Preview close' and then you can continue to edit the graph.

15 Development environment

This is a short description of directories and files that are used by Ge, and what files are generated.

File types

Ge reads graph files and save graph files in the directory $pwrp_pop.

A graph is saved as a file of type .pwg. A subgraph is also saved as a .pwg-file, that is used when the subgraph is edited, but it is also saved as a .pwsg-file. It is the .pwsg-file that is loaded when you create instances of the subgraph.

If you export the graph as a java application (frame) you create a .java-file with the name specified as java name.

If you export the graph as a java applet, a java-file is created with the name stated as java name with the suffix _A, e.g. Oversikg_A.java. Furthermore a html files is created with the class name as name.

Setup-files

Ge uses two setup files, one for the color palette, and one for the subgraphs palette.

Color palette

If you want to modify the colors in the color palette, this is possible by creating av file with the name ge_colors.dat. This should contain the rgb-codes for the colors in the palette. There is a template in ...

Subgraph palette

The subgraph palette contains a collection of fix menus and subgraphs, and the folder Local that displayes the subgraphs and images in $pwrp_pop. Most of the fix menus and subgraphs are configured in the file pwr_ge_setup.dat. This file is common for all projects but can be modified by the system manager.

16 Commands

Ge contains a command line that is opened from the menu or by Ctrl/b. The command is mainly considered to be used in ge scrips, but can also be executed from the command line. From the command line ge script is also executed by writing the name of the file preceded by a '@'.
Note that a change of functions in the editor made via a command is not viewed in the optionmenus or check boxes. When a scrip is run the setting can differ from what is viewed.

17 Script

The script handling in Ge is described in the chapter Ge script and commands. Here some examples are viewed to get an idea of what scripts can do.

17.1 Draw a graph with script in Xtt

In addition to drawing a graph in the Ge editor, it is also possible to to write a ge script that draws the graph. The script can call Ge script functions to draw graphical elements in the graph, but it also call also xtt script functions. This makes it possible to get information from the database and adapt the graph to the current state.

The script graph is opened as an ordinary graph but the graph name is replaced by the script name preceded by a '@', eg 'open graph @myscript'. Scripts are by default read from $pwr_exe or $pwrp_exe, and for any other location the path should be added, eg 'open graph @"$pwrp_login/myscript"'.

Example
A simple script that draws a rectangle and a push button

```
main()
  int id;
  float x1;
  float y1;
  float x2;
  float y2;
  float width;
  float height;

  SetDraw(0);
  SetBackgroundColor(eDrawType_Color66);

  # Draw rectangle
  x1 = 1;
  y1 = 1;
  width = 18;
  height = 5;
  id = CreateRectangle(x1, y1, width, height);
  SetObjectFillColor(id, eDrawType_Color74);
  SetObjectFill(id, 1);
  SetObjectBorder(id, 0);

  # Draw pushbutton
  x1 = 7.5;
  y1 = 2;
  x2 = 10.5;
  y2 = 3.5;
  id = CreateObject("pwr_buttontogglecenter", x1, y1, x2, y2);
```

```
  SetObjectAttribute(id, "Text", "Toggle");
  SetObjectAttribute(id, "ToggleDig.Attribute", "H1-Dv1.ActualValue##Boolean");

  # Set graph size
  SetGraphAttribute("x1", 20.0);
  SetGraphAttribute("y1", 7.0);

  SetDraw(1);
endmain
```

17.2 Modify graphs in the editor

A script is executed from the command line in Ge by setting a '@' before the filename, for example

```
ge> @my_script
```

The command line is opened from Functions/Command (Ctrl+B).

Example
This example shows how to change a subgraph in several graphs. A subgraph is normally intern, and a modification does not take effect until you have performed the procedure to

- set the subgraph extern
- save the graph
- read the graph, now with the modified subgraph
- set the subgraph intern again
- save the graph

If you often modify you subgraphs you can with advantage write a script that replaces a subgraph in all graphs of the project.

```
!
! Replace a subgraph
!
function int process(string graph)
  int sts;
  printf( "Processing graph %s\n", graph);
  open 'graph'
  sts = SetExtern("my_subgraph");
  if ( sts)
    printf( "Changing my_subgraph\n");
    save
    open 'graph'
    SetIntern( "my_subgraph");
    save
  endif
endfunction

main()
  process( "nu4_alla_platar");
  process( "nu4_status_trp");
  process( "nu4_trp_hy1");
  process( "nu4_trp_rb_ut");
```

```
  process( "nu4_buffhog");
  process( "nu4_status_ugn");
  process( "nu4_trp_lu1");
  process( "nu4_trp_start");
  process( "nu4_inlagg");
  process( "nu4_trend");
  process( "nu4_trp_rb1");
  process( "nu4_ugn_start");
endmain
```

18 Web graphs

Configure a web site

To be able to open the varoius process graphs in a system it is suitable to have web page with a menu to the left and actual graph to the right. With some configuration you can generate such a page from 'Generate web' in the menu.

This is how the configuration is done:
Create a WebHandler object under the node object in the node hierarchy. The WebHandler object will cause the start of a server process that supplies the graphs with the dynamic information from the realtime database. In the object you also state if you should be able to log in as a ProviewR user. Under the WebHandler object you create a WebGraph object for each graph that is to be opened from the menu.
Under the WebHandler object, you can also create WebLink objects. Each WebLink object give rise to a menu entry that is connected to an URL, e.g. documents for working instructions, function specifications etc.

All files required for the web site are gathered in the directory $pwrp_web. By copying these files, together with the files $pwr_lib/pwr_rt_client and $pwr_lib/pwr_jop.jar, to a suitable directory in the web server, the operator station is available on the net.

19 Runtime

When the graph is completed it is to be displayed in an operator station, an process station or on the intranet.

Maintenance graphs in xtt

Maintenance graphs can be opened without configuration from the command line in xtt, which is described in the chapter Test. This works as long as you yourself and other informed persons should have access to the graphs. If a broader group should have access to them, you can build a menu tree in xtt. This is done by writing commands in the xtt setup file. The setup file ~/xtt_setup.rtt_com is executed each time xtt is started. Here you can define symbols and add commands to create menus. Below is an example of a symbol file that creates a simple maintenance menu.

```
create item/text="Maintenance"/menu/dest=DataBase/before
create item/text="Overview"/command="open graph overview" /menu
                  /dest=Maintenance /firstchild
create item/text="Blast"/command="open graph blast" /menu
                  /dest=Maintenance /firstchild
create item/text="Roller bed"/command="open graph rollerbed" /menu
                  /dest=Maintenance /firstchild
```

Operator graphics in xtt

If the graphs is to be operator graphics you build an operator place with an OpPlace object, an User object and XttGraph objects. Xtt is started with the name of the OpPlace object as argument.

Web graphs

Ge grahps can also be viewed in a web browser with the web operator environment. How this is configured is described in Designer's Guide, chapter Web operator evironment. The web enviroment though has limited functionality for Ge graphs,

- Dynamics DigScript, Script, EmitSignal, CatchSignal and PopupMenu are not implemented.
- Only a few Xtt commands, used in dynamics Command and DigCommand, are implemented. Here is a list of implemented commands
  - open graph
  - open url
  - open trend
  - open fast
  - set subwindow
  - help
  - check
  - call

20 Ge script och commands

20.1 Introduction to script

In Ge there is a collection of commands to create graphic elements as rectangles, texts, polylines and subgraphs. These commands can be executed from the command line in Ge, or executed from script files that is executed from the command line.

The script files can furthermore contain a language similar to c, that contains operators för calculations, if statements and for loops etc. There are also functions to fetch information from the development database. See the manual for Wtt Script for more information.

Ge script can for example be used to convert graph from other systems to ge graphs, or to automatically generate forms or graphs from objects in the database.

Create objects

Base objects as rectangles, lines etc. are created by the create command. Before the object is created you set the disired properties of the editor (fill, border, fill color, text size etc.) and then you create the object.

Rectangle

An example of a filled rectangle

```
set fill
set linewidth 1
set bordercolor Black
set fillcolor BlueHigh7
create rect /x1=10 /y1=1 /width=4  /height=2
```

Circle

An example of a half circle

```
set nofill
set linewidth 2
set bordercolor Black
create arc /x1=1 /y1=1 /x2=3 /y2=2 /angle1=0 /angle2=180
```

Line

Line example

```
set linewidth 1
set bordercolor Black
create line /x1=0 /y1=0 /x2=10 /y2=0
```

Polylinje

To create to polyline, you create the first segment with 'create', and the following segments with 'add'.

Polyline example

```
set nofill
set linewidth 1
set bordercolor Black
create polyline /x1=5 /y1=5 /x2=6 /y2=6
add polyline /x1=5 /y1=7
add polyline /x1= 6/y1=8
add polyline /x1=5 /y1=9
add polyline /x1= 6 /y1=10
```

Text

Example

```
set textsize 14
set textfont LucidaSans
set textcolor RedHigh8
set bold
create text/text="Example" /x1=3 /y1=5
```

Subgraph object

Creating subgraphs object is a bit different. Here you create the object first, and the set the attributes of the object.

Valve example

```
create object /sub=pwr_valve /x1=1 /y1= 1 /x2=3 /y2=2
set current fillcolor YellowGreenMedium4
set current attr DigLowColor.Attribute "Rt-Dv1.ActualValue##Boolean"
set current attr DigLowColor.Color GrayHigh8
```

Pushbutton example

```
create object /sub=pwr_buttonset /x1=5 /y1=1
! Change type to SetDig with Confirm
set current attr Dyntype1 DigLowColor
set current attr Action Inherit|Confirm
set current attr annotation "Start"
set current attr Confirm.Text "Do you really want to..."
set current attr SetDig.Attribute "rt-Dv1.ActualValue##Boolean"
set current attr DigLowColor.Attribute "rt-Dv2.ActualValue##Boolean"
set current attr DigLowColor.Color YellowGreenMedium4
set current attr Access System|Operator1
```

Graph attributes

Before saving the graph, you should state the borders in x and y direction for the graph, and some other attributes. This is done with the command 'set graphattributes'.

Example

```
set graph x0 -3
set graph y0 -3
set graph x1 40
set graph y1 37
set graph AnimationScanTime 0.2
set graph BackgroundImage "corrado.gif"
```

Subgraph attributes

If the graph is to be saved as a subgraph, you also use 'set graphattributes'. You first set the attribute 'subgraph' and thereafter the other attributes that belongs to a subgraph.

Example

```
set graph subgraph 1
set graphattr Action ToggleDig
set graphattr DynType1 DigLowColor
set graphattr Color1 YellowGreenMedium4
set graphattr NoConObstacle 1
```

20.2 Commands

Below is a description of commands in Ge

add
    add polyline
build
create
    create rectangle
    create arc
    create line
    create polyline
    create rectangle
    create text
    create subgraph
    create bar
exit
filter navigator
group
move
    move currentobject
    move selectedobject
new
open
quit
replace
rotate
    rotate currentobject
    rotate selectedobject
save
scale

scale currentobject
scale selectedobject
search object
select
select currentobject
select clear
set
set verify
set noverify
set fill
set nofill
set border
set noborder
set shadow
set noshadow
set grid
set nogrid
set linewidth
set gridsize
set textsize
set textfont
set bold
set nobold
set backgroundcolor
set fillcolor
set bordercolor
set textcolor
set currentobject fillcolor
set currentobject colortone
set currentobject colorlightness
set currentobject colorintensity
set currentobject colorshift
set currentobject gradient
set currentobject attributes
set currentobject annotation
set graphattributes
show
show version

20.2.1 add

20.2.2 add polyline

Add a breakpoint to the latest created polyline.
A polyline is created by 'create polyline' that also creates the first segment. The following segments are created by 'add polyline'.

Syntax

```
ge> add polyline /x1= /y1=
```

/x1	x-coordinate for the added segment.
/y1	y-coordinate for the added segment.

20.2.3 build

Build the current graph.
The pwg-file is copied to $pwrp_exe.

Syntax

```
ge> build
```

20.2.4 create

Create graphic object in the graph.

20.2.5 create rectangle

Create a rectangle.

Syntax

```
ge> create rectangle /x1= /y1= /width= /height=
```

/x1	x-coordinate of upper left corner
/y1	y-coordinate of upper left corner
/widht	Width of the rectangle.
/height	Height of the rectangle.

20.2.6 create arc

Create a circle or an ellipse, or a segment of a circle or ellipse.

Syntax

```
ge> create arc /x1= /y1= /x2= /y2= /angle1= /angle2=
```

/x1	x-coordinate of upper left corner of a rectangle the surrounds the ellipse.
/y1	y-coordinate of upper left corner of a rectangle the surrounds the ellipse.
/x2	x-coordinate of lower right corner of a rectangle the surrounds the ellipse.
/y2	y-coordinate of lower right corner of a rectangle the surrounds the ellipse.
/angle1	Angle that states the start of a circle segment in degrees. Default value 0.
/angle2	Angle that states the size of a circle segment in degrees. Default value 360 degrees.

20.2.7 create line

Create a straight line between two points.

Syntax

```
ge> create line /x1= /y1= /x2= /y2=
```

/x1	x-coordinate for the first end point.
/y1	y-coordinate for the first and point.
/x2	x-coordinate för the second end point.
/y2	y-coordinate för the second end point.

20.2.8 create polyline

Create a polyline, i.e. a line that consists of several straight line segments.
'create polyline' creates the first line segment. The following segments are created by 'add polyline'. The point (x1, y1) is the starting point for the polyline. The next segment

is added to the point (x2, y2).

Syntax

```
ge> create polyline /x1= /y1= /x2= /y2=
ge> add polyline /x1= /y1=
ge> add polyline /x1= /y1=
ge> ...
```

/x1	x-coordinate for the starting point.
/y1	y-coordinate for the starting point.
/x2	x-coordinate for the ending point of the first segment.
/y2	y-coordinate for the ending point of the first segment.

20.2.9 create text

Create a text.

Syntax

```
ge> create text /x1= /y1= /text=
```

/x1	x-coordinate.
/y1	y-coordinate.
/text	Text surrounde by quoutes.

20.2.10 create subgraph

Create a subgraph object.

Syntax

```
ge> create subgraph /x1= /y1= /subgraph= [/x2= /y2=]
```

/x1	x-coordinate for the upper left corner.
/y1	y-coordinate for the upper left corner.
/subgraph	The name of the subgraph. If the subgraph is a part of the ProviewR base system is always has the prefix 'pwr_' and is written with lower case, e.g. 'Valve' has the name pwr_valve.
/x2	If the point (x2, y2) is supplied, the subgraph is scaled to fit inside a rectangle with the corner points (x1, y1) and (x2, y2).
/y2	

20.2.11 create bar

Create a bar object.

Syntax

```
ge> create bar /x1= /y1= [/x2= /y2=]
```

/x1	x-coordinate for the upper left corner.
/y1	y-coordinate for the upper left corner.
/x2	If the point (x2, y2) is supplied, the bar is scaled to fit inside a rectangle with the corner points (x1, y1) and (x2, y2).
/y2	

20.2.12 exit

Save the current graph and close Ge.

Syntax

```
ge> exit ['filename']
```

20.2.13 filter navigator

Filter objects in the object navigator.
Filtering can be made on object name or subgraph with
wildcard pattern.

Syntax

```
ge> filter navigator /type= /pattern=
ge> filter navigator /reset
```

/reset	Reset filter and show all objects.
/type	'name' or 'class'. Name will filter on object name and class on subgraph.
/pattern	Filter pattern. Can contain wildcard (*).

Example

```
ge> filter navigator /type=class /pattern=*button*
```

20.2.14 group

Create a group of the selected objects.

20.2.15 move

20.2.16 move currentobject

Move the current object. The movement can be specified with relative coordinates (x and y) or absolute coordinates (absx and absy) or combinations.

Syntax

```
ge> move currentobject /x= /y=
ge> move currentobject /absx= /absy=
```

/x	Relative movement in x direction.
/y	Relative movement in y direction.
/absx	Movement to absolute coordinate in x direction.
/absy	Movement to absolute coordinate in y direction.

20.2.17 move selectedobject

Move the selected object. The movement can be specified with relative coordinates (x and y) or absolute coordinates (absx and absy) or combinations.

Only one object can be selected.

Syntax

```
ge> move selectedobject /x= /y=
ge> move selectedobject /absx= /absy=
```

/x	Relative movement in x direction.
/y	Relative movement in y direction.
/absx	Movement to absolute coordinate in x direction.
/absy	Movement to absolute coordinate in y direction.

20.2.18 new

Clear and reset the working area.

20.2.19 open

Open a graph.

Syntax

```
ge> open ['filename']
```

20.2.20 quit

Close without save.

Syntax

```
ge> quit
```

20.2.21 replace attribute

String replacement in dynamic and action properties for selected objects.

Syntax

```
ge> replace attribute /from= /to= [/strict]
```

/from	String that is to be replaced.
/to	Replacement string.
/strict	Search of from string is case sensitive.

20.2.22 rotate

20.2.23 rotate currentobject

Rotates the current object around the center of the object.

Syntax

```
ge> rotate currentobject /angle=
```

/angle	Angle in degrees that the object is rotated.

20.2.24 rotate selectedobject

Rotates the selected object around the center of the object.

Syntax

```
ge> rotate selectedobject /angle=
```

/angle	Angle in degrees that the object is rotated.

20.2.25 save

Save a graph.

Syntax

```
ge> save ['filename']
```

20.2.26 scale

20.2.27 scale currentobject

Scale the current object.

Syntax

```
ge> scale currentobject /scalex= /scaley= [/x= /y=]
```

/scalex	Scale factor in x direction.
/scaley	Scale factor in y direction.
/x	x coordinate for the reference point of the scaling.
/y	y coordinate for the reference point of the scaling.

20.2.28 scale selectedobject

Scale the selected object.

Syntax

```
ge> scale selectedobject /scalex= /scaley= [/x= /y=]
```

/scalex	Scalefactor in x direction.
/scaley	Scalefactor in y direction.
/x	x coordinate for the reference point of the scaling.
/y	y coordinate for the reference point of the scaling.

20.2.29 search object

Search for an object.
The found object is selected after the search.

Syntax

```
ge> search object /name=
```

/name	Object name.

20.2.30 select clear

Clear the select list.

20.2.31 select

20.2.32 select currentobject

Put the current object in the list of selected objects.

20.2.33 select clear

Clear the select list.

20.2.34 set

20.2.35 set verify

Set veriy mode, i.e. all executed rows in a script are printed in the terminal window.

20.2.36 set noverify

Reset verify mode.

20.2.37 set fill

Set 'fill' in the editor. Created objects will in the future be created with the fill property set.

20.2.38 set nofill

Reset 'fill' in the editor. Created objects will in the future be created with the fill property reset.

20.2.39 set border

Set 'border' in the editor. Created objects will in the future be created with the border property set.

20.2.40 set noborder

Reset 'border' in the editor. Created objects will in the future be created with the border property reset.

20.2.41 set shadow

Set 'shadow' or '3D' in the editor. Created objects will in the future be created with the shadow property set.

20.2.42 set noshadow

Reset 'shadow' or '3D' in the editor. Created objects will in the future be created with the shadow property reset.

20.2.43 set grid

Set 'grid' in the editor. Created objects are positioned to closest grid point.

20.2.44 set nogrid

Reset grid in the editor.

20.2.45 set linewidth

Set linewidth in the editor. Created objects will in the future be created with specified line width. The line width has a value in the interval 1-8.

Syntax

```
ge> set linewidth 'linewidth'
```

20.2.46 set gridsize

Set gridsize in the editor.

Syntax

```
ge> set gridsize 'gridsize'
```

20.2.47 set textsize

Set 'textsize' in the editor. All text objects will in the future be created with this text size. The text size can be 8, 10, 12, 14, 18 or 24.

Syntax

```
ge> set textsize 'textsize'
```

20.2.48 set textfont

Set 'textfont' in the editor. All text objects will in the future be created with this font. The font can be 'Helvetica', 'Times', 'New Century Schoolbook', 'Curier' or 'LucidaSans'.

Syntax

```
ge> set textfont 'font'
```

20.2.49 set bold

Set 'bold' in the editor. Textobject will in the future be created with bold.

20.2.50 set nobold

Reset bold.

20.2.51 set backgroundcolor

Set background color.

Syntax

```
ge> set backgroundcolor 'color'
```

20.2.52 set fillcolor

Set fill color in the color palette. The object will in the future be created with this fill color.

Syntax

```
ge> set fillcolor 'color'
```

20.2.53 set bordercolor

Set border color in the color palette. Objects will in the future be created with this border color.

Syntax

```
ge> set bordercolor 'color'
```

20.2.54 set textcolor

Set text color in the color palette. Text objects will in the future be created with this text color.

Syntax

```
ge> set textcolor 'color'
```

20.2.55 set currentobject fillcolor

Set fill color of the current object.

Syntax

```
ge> set currentobject fillcolor 'color'
```

20.2.56 set currentobject colortone

Set color tone of the current object.

Syntax

```
ge> set currentobject colortone 'tone'
```

20.2.57 set currentobject colorlightness

Set lightness of the current object. The lightness is an integer. Positive value gives lighter colors, negative darker colors.

Syntax

```
ge> set currentobject colorlightness 'lightness'
```

20.2.58 set currentobject colorintensity

Set color intensity of the current object. The intensity is an integer. Positive value gives more intense colors, negative less intense.

Syntax

```
ge> set currentobject colorintensity 'intensity'
```

20.2.59 set currentobject colorshift

Shift color of the current object. The color tones of the subgraph are rotated in the color circle, but the colors keep their internal color contrast. Colorshift is an integer that states the number of steps to shift the color. Positive value rotates in direction yellowgreen->

yellow->orange->red->violet->blue->seablue->green. Negative in the opposite direction.

Syntax

```
ge> set currentobject colorshift 'shift'
```

20.2.60 set currentobject gradient

Set color gradient on the current object.
The gradient can be No,
HorizontalUp, HorizontalDown, HorizontalTube1, HorizontalTube2,
VerticalLeft, VerticalRight, VerticalTube1, VerticalTube2,
DiagonalUpperLeft, DiagonalLowerLeft, DiagonalUpperRight, DiagonalLowerRight,
DiagonalUpTube, DiagonalDownTube,
Globe, RadialCenter,
RadialUpperLeft, RadialLowerLeft, RadialUpperRight or RadialLowerRight.

Syntax

```
ge> set currentobject gradient 'gradient'
```

20.2.61 set currentobject attributes

Set attributes of the current object.

Syntax

```
ge> set currentobject attributes 'attrname' 'value'
```

The name the attributes depends on which type of dynamic is applied on the subgraph. The dynamic type has to be stated first, and thereafter the attributes for that dynamic type is stated.

Enumeration types
For attributes of enumeration type, where you can select one of several alternatives, the value is set to the name of the enumeration value, eg

```
 set currentobject attr Cycle Slow
```

Bitmask types
For attributes of bitmask type, where you can select several alternatives, the names of the desired alternatives are specifed separated with '|', eg

```
 set currentobject attr Action ToggleDig|Confirm|Tooltip
 set currentobject attr DigColor.Instances 2|3|4
```

Example

```
 set currentobject attr Cycle Fast
 set currentobject attr DynType1 DigColor|DigWarning|DigError
 set currentobject attr Action Tooltip|OpenGraph
 set currentobject attr DigColor.Instances 2|3
 set currentobject attr DigColor3.Attribute H1-Dv3##Boolean
 set currentobject attr DigColor3.Color MagentaHigh3
 set currentobject attr DigColor2.Attribute H1-Dv2##Boolean
 set currentobject attr DigColor2.Color OrangeHigh4
 set currentobject attr DigColor.Attribute H1-Dv1##Boolean
```

```
set currentobject attr DigColor.Color BlueHigh3
set currentobject attr DigError.Attribute H1-Dv4##Boolean
set currentobject attr DigWarning.Attribute H1-Dv5##Boolean
set currentobject attr OpenGraph.GraphObject Nodes-MyNode-Op-Overview
set currentobject attr ToolTip.Text "Temperature switch indicator"
```

20.2.62 set currentobject annotation

Insert a string in the annotation in the current object. The text is put in annotation number 1.

Syntax

```
ge> set currentobject annotation 'text'
```

20.2.63 set graphattributes

Set an attribute for the current graph or subgraph.

Graph attributes

Name	Type
subgraph	Boolean
x0	Float
y0	Float
x1	Float
y1	Float
ScanTime	Float
AnimationScanTime	Float
JavaWidth	Float
IsJavaApplet	Boolean
IsJavaFrame	Boolean
BackgroundImage	String
BackgroundIiled	Boolean
DoubleBuffered	Boolean
MB3Action	Enumeration: No, Close, PopupMenu or Both
Translate	Boolean
BitmapFonts	Boolean
HotIndication	Enumeration: No, LineWidth, DarkColor or LightColor
TooltipTextsize	Integer
AppMotion	Enumeration: Scroll, Slider or Both

Subgraph attributes

Name	Type
DynType1	Mask: Inherit, Tone, DigLowColor, DigColor, AnalogColor, StatusColor, DigWarning, DigError, DigFlash, FillLevel, Invisible, DigBorder, DigText, Valule, Rotate, Move, AnalogShift, DigShift, Animation, Bar, Trend, FastCurve, XY_Curve, SliderBackground, Video, Table, HostObject, DigSound, DigCommand
DynType2	Mask: DigTextColor, TimeoutColor
Action	Mask: Inherit, PopupMenu, SetDig, ResetDig, ToggleDig, StoDig, SetValue, Command, CommandDoubleClick, Help, OpenGraph, CloseGraph, OpenURL, Confirm, IncrAnalog, RadioButton, ValueInput, ToolTip, InputFocus, PulldownMenu, OptionMenu, MethodPulldownMenu, Slider
Color1	A color or color tone

Color2	A color or color tone
AnimSequence	Enumeration: Cyclic, Dig or ForwBack
NoConObstacle	Boolean
Slider	Boolean
NextSubgraph	String
AnimationCount	Int
JavaName	String
Cycle	Enumeration: Fast or Slow
x0	Float
y0	Float
x1	Float
y1	Float
InputFocusMark	Enumeration: Relief or No.
RecursiveDynamic	Boolean

Syntax

```
ge> set graphattributes 'name' 'value'
```

Example graph

```
 set graphattributes x0 0
 set graphattributes y0 0
 set graphattributes x1 15
 set graphattributes y1 20
 set graphattributes DoubleBuffered 1
 set graphattributes MB3Action PopupMenu
 set graphattributes TooltipTextsize 14
```

Example subgraph

```
 set graphattributes Subgraph 1
 set graphattributes DynType1 DigLowColor|DigWarning|DigError
 set graphattributes Action PopupMenu|OpenGraph
 set graphattributes Color1 GrayLow9
 set graphattributes Cycle Fast
```

20.2.64 show

20.2.65 show version

Show Ge version.

21 Script

Ge script is a way to program ge commands. In the script function there are also possibilities to perform calculations, if statements, loop statements, declarations of variables and functions.

A script is started with '@' followed by the name of the script file and possible arguments.

A ge-script should have the filetype .ge_com.

Example

```
ge> @my_script
```

In Wtt script there is a description of the script language and a description of built in functions. Here follows a list of build in functions that also can be used in Ge script.

Ge functions

Function	Description
BuildGraph	Build the current graph.
ClearAll	Remove all objects.
CreateArc	Create an Arc object.
CreateAxis	Create an Axis object.
CreateAxisArc	Create an AxisArc object.
CreateBar	Create a Bar object.
CreateBarArc	Create a BarArc object.
CreateDsTrend	Create a DsTrend object.
CreateDsTrendCurve	Create a DsTrendCurve object.
CreateFastCurve	Create a FastCurve object.
CreateImage	Create an Image object.
CreateLayer	Create a Layer.
CreateLine	Create a Line object.
CreateObject	Create a subgraph object.
CreatePie	Create a Pie object.
CreatePolyLine	Create a PolyLine object.
CreateRectangle	Create a Rectangle object.
CreateRecRounded	Create a rounded rectangle object.
CreateSevHist	Create a SevHist object.
CreateText	Create a Text object.
CreateToolbar	Create a MethodsToolbar object.
CreateTrend	Create a Trend object.
CreateWindow	Create a Window object.
CreateXYCurve	Create an XYCurve object.
DashInsertObject	Insert an object into a DashCell.
DeleteObject	Delete an object.
GetCurrentObject	Returns the identity of the last created object.
GetFirstObject	Get the first object.
GetGraphAttribute	Get an attribute value of a graph.
GetGraphConfig	Get value of GraphConfiguration attribute.
GetGraphName	Get name of current graph.
GetInstanceObject	Get the instance object of the graph.
GetNextObject	Get the next object.

GetModified	Check if graph is modified.
GetObjectAttribute	Get an attribute value of an object.
GetObjectBorder	Get object border.
GetObjectBorderColor	Get the border color of an object.
GetObjectClass	Get object class.
GetObjectDynType	Get dynamic and action type of an object.
GetObjectFill	Get object fill.
GetObjectFillColor	Get fill color of an object.
GetObjectGradient	Get gradient of an object.
GetObjectName	Get object name.
GetObjectShadow	Get shadow of an object.
GetObjectText	Get object text.
GetObjectTextColor	Get text color of an object.
GetObjectTransparency	Get object transparency.
GetObjectType	Get object type.
GetRgbColor	Get rgb values of a color.
GetTextExtent	Calculate the extent of a text.
GetUI_Env	Get UI environment.
GetWindowSize	Get size in pixels of the current window.
GetWindowDimension	Get size in Ge units of the current window.
GroupGetFirstObject	Get first object in a group.
GroupGetNextObject	Get next object in a group.
GroupSelected	Group the selected objects.
LayerGetFirstObject	Get first object in a layer.
LayerGetNextObject	Get next object in a layer.
LayerResetActiveAll	Set all layers inactive.
LayerSetActive	Set a layer active or inactive.
Layout	Calculate a layout.
MeasureObject	Get object extent.
MergeVisibleLayers	Merge visible layers.
MergeAllLayers	Merge all layers.
MoveAbsObject	Move an object to a position.
MoveObject	Move an object a distance.
MoveSelectToLayer	Move selected objects to active layer.
PopSeleted	Pop the selected objects.
PushSeleted	Push the selected objects.
OpenGraph	Open a graph.
PolyLineAdd	Add a segment to a PolyLine.
Reload	Read current graph from file.
RotateSelected	Rotate selected objects.
SaveGraph	Save the current graph.
ScaleObject	Scale an object.
SelectAdd	Add an object to select list.
SelectClear	Clear select list.
SetBackgroundColor	Set graph background color.
SetColorTheme	Set color theme.
SetCurrentObject	Set CurrentObject.
SetDraw	Set drawing on or off.
SetExtern	Set subgraph extern.
SetExternAll	Set all subgraphs extern.
SetIntern	Set subgraph intern.
SetInternAll	Set all subgraph intern.
SetGraphAttribute	Set an attribute value of a graph.
SetGraphName	Set name of current graph.
SetGraphOptions	Set options for the current graph.
SetObjectAttribute	Set an attribute value of an object.
SetObjectBackgroundColor	Set background color of an object.
SetObjectBorder	Set object border.

SetObjectBorderColor	Set border color of an object.
SetObjectClass	Replace the subgraph of an object.
SetObjectFill	Set object fill.
SetObjectFillColor	Set fill color of an object.
SetObjectGradient	Set gradient of an object.
SetObjectShadow	Set object shadow.
SetObjectTextColor	Set text color of an object.
SetObjectTransparency	Set object transparency.
SetRgbColor	Set rgb values of a color.
SetSelectTextBold	Set text bold on selected objects.
SetSelectTextFont	Set text font on selected objects.
SetSelectTextSize	Set text size on selected objects.
TranslateObjectName	Translate Ge syntax name to database name.

Input and output

Function	Description
ask	Print a question and read an answere.
say	Print a text.
printf	Formatted print.
scanf	Formatted read.

File handling

Function	Description
fclose	Close a file
felement	Extract one element from the last read line.
fgets	Read a line from a file.
file_search	Search for file or files.
fopen	Open a file.
fprintf	Formatted write to file.
fscanf	Formatted read from file.
translate_filename	Replace environment variables in a file name.

Handling of strings

Function	Description
edit	Removes leading and trailing spaces and tabs, and replaces multiple tabs and spaces with a single space.
element	Extract one element from a string of elements.
extract	Extracts the specified characters from a string.
sprintf	Formatted print to a string variable.
strchr	Find the first occurrence of a character in a string.
strlen	The length of a string.
strrchr	Find the last occurrence of a character in a string.
strstr	Find the first occurrence of a character sequence in a string.
tolower	Convert a string to lower case.
toupper	Convert a string to upper case.

Database functions

Function	Description
CutObjectName	Extract the last segment of an object name.
GetAttribute	Fetch the value of an attribute.
GetChild	Get the first child of an object.
GetNextSibling	Get next sibling of an object.
GetNextVolume	Get next volume.

GetParent	Get the parent of an object.
GetObjectClass	Get the class of an object.
GetRootList	Get first object in the root list.
GetVolumeClass	Get the class of a volume.

System functions

Function	Description
exit	Terminate the execution of a script.
system	Execute a shell command.
terminate	Terminate the process.
time	Fetch the system time.
verify	Set verify on or off.

21.1 Predefined variables

Object types

eObjectType_Rect
eObjectType_Line
eObjectType_Arc
eObjectType_ConPoint
eObjectType_Annot
eObjectType_PolyLine
eObjectType_SubGraph
eObjectType_Text
eObjectType_Bar
eObjectType_Trend
eObjectType_Slider
eObjectType_Image
eObjectType_Group
eObjectType_Axis
eObjectType_RectRounded
eObjectType_ConGlue
eObjectType_Menu
eObjectType_Window
eObjectType_Table
eObjectType_Folder
eObjectType_XYCurve
eObjectType_AxisArc
eObjectType_Pie
eObjectType_BarChart
eObjectType_Toolbar

Dynamic and action types

DynType1
mDynType1_No
mDynType1_Inherit
mDynType1_Tone
mDynType1_DigLowColor
mDynType1_DigColor
mDynType1_DigError

mDynType1_DigWarning
mDynType1_DigFlash
mDynType1_Invisible
mDynType1_DigBorder
mDynType1_DigText
mDynType1_Value
mDynType1_AnalogColor
mDynType1_Rotate
mDynType1_Move
mDynType1_AnalogShift
mDynType1_DigShift
mDynType1_Animation
mDynType1_Bar
mDynType1_Trend
mDynType1_SliderBackground
mDynType1_Video
mDynType1_FillLevel
mDynType1_FastCurve
mDynType1_AnalogText
mDynType1_Table
mDynType1_StatusColor
mDynType1_HostObject
mDynType1_DigSound
mDynType1_XY_Curve
mDynType1_DigCommand
mDynType1_Pie
mDynType1_BarChart

DynType2
mDynType2_No
mDynType2_Axis
mDynType2_DigTextColor
mDynType2_TimeoutColor
mDynType2_DigFourShift
mDynType2_ScrollingText
mDynType2_ColorThemeLightness
mDynType2_DigBackgroundColor

ActionType1
mActionType1_No
mActionType1_Inherit
mActionType1_PopupMenu
mActionType1_SetDig
mActionType1_ResetDig
mActionType1_ToggleDig
mActionType1_StoDig
mActionType1_Command
mActionType1_CommandDoubleClick
mActionType1_Confirm
mActionType1_IncrAnalog
mActionType1_RadioButton
mActionType1_Slider
mActionType1_ValueInput
mActionType1_TipText
mActionType1_Help
mActionType1_OpenGraph
mActionType1_OpenURL
mActionType1_InputFocus

mActionType1_CloseGraph
mActionType1_PulldownMenu
mActionType1_OptionMenu
mActionType1_SetValue
mActionType1_MethodToolbar
mActionType1_MethodPulldownMenu
mActionType1_Script

Fonts

eFont_Helvetica
eFont_Times
eFont_NewCenturySchoolbook
eFont_Courier
eFont_LucidaSans

Directions

eDirection_Center
eDirection_Right
eDirection_Left
eDirection_Up
eDirection_Down

Colors

Standard palette
eDrawType_Color1 = 0
eDrawType_Color2 = 1
eDrawType_Color3 = 2
...
eDrawType_Color300 = 299

Custom palette
eDrawType_CustomColor1 = 310
eDrawType_CustomColor2 = 314
eDrawType_CustomColor3 = 318
...
eDrawType_CustomColor90 = 666

Access

mAccess_RtRead
mAccess_RtWrite
mAccess_System
mAccess_Maintenance
mAccess_Process
mAccess_Instrument
mAccess_Operator1
mAccess_Operator2
mAccess_Operator3
mAccess_Operator4
mAccess_Operator5
mAccess_Operator6
mAccess_Operator7
mAccess_Operator8
mAccess_Operator9
mAccess_Operator10

mAccess_RtEventsAck
mAccess_RtPlc
mAccess_RtNavigator
mAccess_DevRead
mAccess_DevPlc
mAccess_DevConfig
mAccess_DevClass
mAccess_RtEventsBlock
mAccess_Administrator
mAccess_SevRead
mAccess_SevAdmin
mAccess_AllRt
mAccess_RtDefault
mAccess_AllOperators
mAccess_AllSev
mAccess_AllPwr
mAccess_Default

21.2 BuildGraph()

void BuildGraph()

Description

Build the current graph.

Example

```
BuildGraph();
```

21.3 ClearAll()

void ClearAll()

Description

Remove all objects in the current graph.

Example

```
ClearAll();
```

21.4 CreateArc()

int CreateArc(float x1, float y1, float x2, float y2, int angle1, int angle2)

Description

Create an arc.
Color, fill, border and other properties are fetched from
the current editor settings.

Argument

float	x1	X coordinate for upper left corner.
float	y1	Y coordinate for upper left corner.
float	x2	X coordinate for lower right corner.
float	y2	Y coordinate for lower right corner.
int	angle1	Start angle in degrees. Zero is 3 o'clock.
int	angle2	Arc extension in degrees clockwise from the start angle.

Returns the id of the created object.

Example

```
int id;

id = CreateArc(2.0, 2.0, 3.0, 3.0, 0, 180);
```

21.5 CreateAxis()

int CreateAxis(float x1, float y1 [,float x2, float y2, int colortheme, int dynamic, int direction])

Description

Create an axis object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.
int	dynamic	Dynamic axis (1) or static (0). Optional.
int	direction	Direction up, down, left or right. Specified with symbols eDirection_Up, eDirection_Down, eDirection_Left and eDirection_Right. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateAxis(1.0, 1.0, 10.0, 2.0, 0, 0, eDirection_Down);
SetObjectAttribute(id, "MaxValue", 100.0);
SetObjectAttribute(id, "MinValue", -100.0);
```

21.6 CreateAxisArc()

int CreateAxisArc(float x1, float y1 [,float x2, float y2, int colortheme, int dynamic])

Description

Create an axis arc object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.
int	dynamic	Dynamic axis (1) or static (0). Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateAxisArc(1.0, 1.0, 5.0, 4.0, 0, 0);
SetObjectAttribute(id, "MaxValue", 100.0);
SetObjectAttribute(id, "MinValue", -100.0);
SetObjectAttribute(id, "Angle1", -45);
SetObjectAttribute(id, "Angle2", 270);
```

21.7 CreateBar()

int CreateBar(float x1, float y1 [,float x2, float y2, int colortheme, int direction])

Description

Create a bar object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.
int	direction	Direction up, down, left or right. Specified with symbols eDirection_Up, eDirection_Down, eDirection_Left and eDirection_Right. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateBar(2.0, 2.0, 2.5, 5.0, 1, eDirection_Down);
SetObjectAttribute(id, "Bar.Attribute", "H1-Av1.ActualValue##Float32");
SetObjectAttribute(id, "Bar.MaxValue", 100.0);
SetObjectAttribute(id, "Bar.MinValue", -100.0);
```

21.8 CreateBarArc()

int CreateBar(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create an BarArc object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateBarArc(2.0, 2.0, 5.0, 4.0, 1);
SetObjectAttribute(id, "Bar.Attribute", "H1-Av1.ActualValue##Float32");
SetObjectAttribute(id, "BarArc.MaxValue", 100.0);
SetObjectAttribute(id, "BarArc.MinValue", -100.0);
SetObjectAttribute(id, "BarArc.Angle1", -45);
SetObjectAttribute(id, "BarArc.Angle2", 270);
```

21.9 CreateDsTrend()

int CreateDsTrend(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a DsTrend object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateDsTrend(2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "DsTrend.Object1", "H1-Trend1");
SetObjectAttribute(id, "DsTrend.MaxValue1", 100.0);
SetObjectAttribute(id, "DsTrend.MinValue1", -100.0);
SetObjectAttribute(id, "DsTrend.ScanTime", 0.5);
```

21.10 CreateDsTrendCurve()

int CreateDsTrendCurve(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a DsTrendCurve object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateDsTrendCurve(2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "DsTrendCurve.Object", "H1-TrendCurve");
SetObjectAttribute(id, "DsTrendCurve.MaxValue1", 100.0);
SetObjectAttribute(id, "DsTrendCurve.MinValue1", -100.0);
SetObjectAttribute(id, "DsTrendCurve.ScanTime", 0.5);
```

21.11 CreateFastCurve()

int CreateFastCurve(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a FastCurve object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateFastCurve(2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "FastCurve.Object", "H1-FastCurve");
SetObjectAttribute(id, "FastCurve.MaxValue1", 100.0);
SetObjectAttribute(id, "FastCurve.MinValue1", -100.0);
```

21.12 CreateImage()

int CreateImage(string image, float x1, float y1 [,float x2, float y2])

Description

Create an image object.

Argument

string	image	Image file name. The file should be present on $pwrp_exe or $pwr_exe.
float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the image should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the image should be scaled. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateImage("pwr_logofully.png", 2.0, 2.0);
```

21.13 CreateLine()

int CreateLine(float x1, float y1, float x2, float y2)

Description

Create a straight line.
Color, width and line type are fetch from the current editor settings.

Argument

float	x1	X coordinate for start point.
float	y1	Y coordinate for start point.
float	x2	X coordinate for end point.
float	y2	Y coordinate for end point.

Returns the id of the created object.

Example

```
int id;

id = CreateLine(2.0, 2.0, 3.0, 3.0);
```

21.14 CreateObject()

int CreateObject(string subgraph, float x1, float y1 [,float x2, float y2])

Description

Create an subgraph object.

Argument

string	subgraph	Subgraph name.
float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the subgraph should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the subgraph should be scaled. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateObject("pwr_valve", 2.0, 2.0);
```

21.15 CreatePie()

int CreatePie(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a pie object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreatePie(2.0, 2.0, 2.5, 5.0, 1);
SetObjectAttribute(id, "Pie.Sectors", 3);
SetObjectAttribute(id, "Pie.FixRange", 1);
SetObjectAttribute(id, "Pie.Attribute1", "H1-Av1.ActualValue##Float32");
SetObjectAttribute(id, "Pie.Attribute2", "H1-Av2.ActualValue##Float32");
SetObjectAttribute(id, "Pie.Attribute3", "H1-Av3.ActualValue##Float32");
SetObjectAttribute(id, "Pie.MinValue", 0.0);
SetObjectAttribute(id, "Pie.MaxValue", 300.0);
```

21.16 CreatePolyLine()

int CreatePolyLine(float x1, float y1, float x2, float y2)

Description

Create the first segment of a polyline. Other segments are added
with PolyLineAdd().
Color, width and line type are fetch from the current editor settings.

Argument

float	x1	X coordinate for start point.
float	y1	Y coordinate for start point.
float	x2	X coordinate for end point.
float	y2	Y coordinate for end point.

Returns the id of the created object.

Example

```
int id;

id = CreatePolyLine(2.0, 2.0, 3.0, 3.0);
PolyLineAdd(id, 4.0, 2.0);
PolyLineAdd(id, 5.0, 3.0);
```

21.17 CreateRectangle()

int CreateRectangle(float x, float y, float width, float height)

Description

Create a rectangle.
Color, fill, border and other properties are fetch from
the current editor settings.

Argument

float	x	X coordinate for upper left corner.
float	y	Y coordinate for upper left corner.
float	width	Rectangle width.
float	height	Rectangle height.

Returns the id of the created object.

Example

```
int id;

id = CreateRectangle(2.0, 2.0, 5.0, 3.0);
```

21.18 CreateRectRounded()

int CreateRectRounded(float x, float y, float width, float height)

Description

Create a rectangle with rounded corners.
Color, fill, border and other properties are fetch from
the current editor settings.

Argument

float	x	X coordinate for upper left corner.
float	y	Y coordinate for upper left corner.
float	width	Rectangle width.
float	height	Rectangle height.

Returns the id of the created object.

Example

```
int id;

id = CreateRectRounded(2.0, 2.0, 5.0, 3.0);
```

21.19 CreateSevHist()

int CreateSevHist(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a SevHist object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateSevHist(2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "SevHist.Object1", "H1-Av1-Hist");
SetObjectAttribute(id, "SevHist.MaxValue1", 100.0);
SetObjectAttribute(id, "SevHist.MinValue2", -100.0);
SetObjectAttribute(id, "SevHist.TimeRange", 120.0);
```

21.20 CreateText()

int CreateText(string text, float x, float y)

Description

Create a text object. Font, size, color and other properties are fetched from the current editor settings.

Argument

string	text	Text.
float	x	X coordinate.
float	y	Y coordinate.

Returns the id of the created object.

Example

```
int id;

id = CreateText("Start", 2.0, 2.0);
```

21.21 CreateTrend()

int CreateTrend(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a Trend object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateTrend(2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "Trend.Attribute1", "H1-Av1.ActualValue##Float32");
SetObjectAttribute(id, "Trend.MaxValue1", 100.0);
SetObjectAttribute(id, "Trend.MinValue1", -100.0);
SetObjectAttribute(id, "Trend.ScanTime", 0.5);
```

21.22 CreateXYCurve()

int CreateXYCurve(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a XYCurve object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateXYCurve(2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "XY_Curve.XAttr", "H1-X1.Value##Float32#100");
SetObjectAttribute(id, "XY_Curve.YAttr", "H1-Y1.Value##Float32#100");
SetObjectAttribute(id, "XY_Curve.XMaxValue", 100.0);
SetObjectAttribute(id, "XY_Curve.XMinValue", 0.0);
SetObjectAttribute(id, "XY_Curve.YMaxValue", 100.0);
SetObjectAttribute(id, "XY_Curve.YMinValue", -100.0);
SetObjectAttribute(id, "XY_Curve.UpdateAttr", "H1-Update.ActualValue##Boolean");
```

21.23 CreateWindow()

int CreateWindow(float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a Window object.

Argument

float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateWindow(2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "Window.FileName", "subwindow.pwg");
```

21.24 CreateToolbar()

int CreateToolbar(string name, float x1, float y1 [,float x2, float y2])

Description

Create a MethodToolbar object.

Argument

string	name	Object name.
float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateToolbar("T1", 2.0, 2.0, 7.0, 5.0);
```

21.25 CreateTable()

int CreateTable(string name, float x1, float y1 [,float x2, float y2, int colortheme])

Description

Create a Table object.

Argument

string	name	Object name.
float	x1	X coordinate of upper left corner.
float	y1	Y coordinate of upper left corner.
float	x2	X coordinate of lower right corner if the object should be scaled. Optional.
float	y2	Y coordinate of lower right corner if the object should be scaled. Optional.
int	colortheme	1 if colortheme colors should be used, else 0. Optional.

Returns the id of the created object.

Example

```
int id;

id = CreateTable("T1", 2.0, 2.0, 7.0, 5.0, 1);
SetObjectAttribute(id, "Table.Columns", 2);
SetObjectAttribute(id, "Table.Rows", 20);
SetObjectAttribute(id, "Column1.Attribute", "H1-Array1.Value##Float32#100");
SetObjectAttribute(id, "Column1.Format", "%6.2f");
SetObjectAttribute(id, "Column2.Attribute", "H1-Array2.Value##Float32#100");
SetObjectAttribute(id, "Column2.Format", "%6.2f");
```

21.26 CreateLayer()

int CreateLayer()

Description

Create a Layer.

Returns the id of the created object.

Example

```
int id;

id = CreateLayer();
```

21.27 DashInsertObject()

void DashInsertObject(int dashoid, int oid)

Description

Insert an object into a dashcell.

Argument

int	dashoid	DashCell object identity.
int	oid	Object identity.

Example

```
int doid;
int oid;

DashInsertObject(doid, oid);
```

21.28 DeleteObject()

void DeleteObject(int oid)

Description

Delete an object.

Argument

int	oid	Object identity.

Example

```
int oid;

DeleteObject(oid);
```

21.29 GetCurrentObject()

int GetCurrentObject()

Description

Returns the identity of the last created object.

Example

```
int id;

id = GetCurrentObject();
```

21.30 GetFirstObject()

int GetFirstObject()

Description

Fetches the identity for the first object in a graph. Other objects can be fetched with GetNextObject().

Returns an integer with the object identity.

Example

```
int oid;

oid = GetFirstObject();
while ( oid != 0)
  ...
  oid = GetNextObject(oid)
endwhile
```

21.31 GetGraphAttribute()

int GetGraphAttribute(string attribute, (arbitrary type) value)

Description

Returns the value of an attribute of the current graph. The attribute name is the same as in the graph attributes editor.

Argument

string	attribut	Attribute name.
(arbitrary type)	value	Fetched attribute value

Returns the status of the operation.

Example

```
int sts;
float scantime;

sts = GetGraphAttribute( oid, "Scantime", scantime);
if ( !(sts & 1))
  printf( "Couldn't get scan time\n");
endif
```

21.32 GetGraphConfig()

int GetGraphConfig()

Description

Fetches the value of the GraphConfiguration attribute of the current instance object.

Only implemented in Xtt environment. Returns 0 in other environments.

Returns the configuration.

Example

```
int conf;

conf = GetGraphConfig();
```

21.33 GetGraphName()

string GetGraphName()

Description

Fetches the name for the current graph.

Returns a string with the name.

Example

```
string name;

name = GetGraphName();
```

21.34 GetGraphName()

string GetInstanceObject()

Description

Fetches the instance object for an object graph.

Returns a string with the name.

Example

```
string name;

name = GetInstanceObject();
```

21.35 GetModified()

int GetModified()

Description

Returns 1 if the graph is modified, else 0.

Example

```
int mod;

mod = GetModified();
```

21.36 GetNextObject()

int GetNextObject(int oid)

Description

Fetches the identity for the next object in a graph. The first object is fetched with GetFirstObject() and the other with GetNextObject().

Argument

int	oid	Object identity.

Returns an integer with the object identity.

Example

```
int oid;

oid = GetFirstObject();
while ( oid != 0)
  ...
  oid = GetNextObject( oid)
endwhile
```

21.37 GetObjectAttribute()

int GetObjectAttribute(int oid, string attribute, (arbitrary type) value)

Description

Returns the value of an attribute of en object. The attribute name is the same as in the object editor for the object.

Argument

int	oid	Object identity.
string	attribut	Attribute name.
(arbitrary type)	value	Fetched attribute value

Returns the status of the operation.

Example

```
int sts;
int oid;
int color;

sts = GetObjectAttribute( oid, "DigColor.Color", color);
if ( !(sts & 1))
  printf( "Couldn't get color\n");
endif
```

21.38 GetObjectBorder()

int GetObjectBorder(int oid)

Description

Get the border property of an object.

Returns the border value, 0 or 1.

Argument

int	oid	Object identity.

Example

```
 int oid;
 int border;

 border = GetObjectBorder( oid);
```

21.39 GetObjectBorderColor()

int GetObjectBorderColor(int oid)

Description

Get the border color of an object.

Argument

int	oid	Object identity.

Returns the border color.

Example

```
int oid;
int bcolor;

bcolor = GetObjectBorderColor( oid);
if ( bcolor == eDrawType_Color33)
  SetObjectBorderColor( oid, eDrawType_CustomColor24);
endif
```

21.40 GetObjectClass()

string GetObjectClass(int oid)

Description

Get the name of the subgraph of an object.

Argument

int	oid	Object identity.

Returns the name.

Example

```
int oid;
string class;

class = GetObjectClass( oid);
if ( class == "pwr_roundind")
  ...
endif
```

21.41 GetObjectDynType()

int GetObjectDynType(int oid, int dyntype1, int dyntype2, int actiontype1, int actiontype2)

Description

Get the dynamic type and action type of an object.

Argument

int	oid	Object identity.
int	dyntype1	Returns the value of DynType1.
int	dyntype2	Returns the value of DynType2.
int	actiontype1	Returns the value of ActionType1.
int	actiontype2	Returns the value of ActionType2.

Returns status of the operation.

See Predefined variables for dyntype and actiontype values.

Example

```
int sts;
int oid;
int dyntype1;
int dyntype2;
int actiontype1;
int actiontype2;

sts = GetObjectDynType( oid, dyntype1, dyntype2, actiontype1, actiontype2);
if ( dyntype1 & mDynType1_Value)
  ...
endif
```

21.42 GetObjectFill()

int GetObjectFill(int oid)

Description

Get the fill property of an object.

Argument

int	oid	Object identiry.

Returns the fill value, 0 or 1.

Example

```
int oid;
int fill;

fill = GetObjectFill( oid);
```

21.43 GetObjectFillColor()

int GetObjectFillColor(int oid)

Description

Get the fill color of an object.

Argument

int	oid	Object identity.

Returns the fill color.

Example

```
int oid;
int fcolor;

fcolor = GetObjectFillColor( oid);
if ( fcolor == eDrawType_Color67)
  SetObjectFillColor( oid, eDrawType_CustomColor23);
endif
```

21.44 GetObjectGradient()

int GetObjectGradient(int oid)

Description

Get the gradient of an object.

Argument

int	oid	Object identity.

Returns the gradient value, ie the type of gradient. Value 0 means no gradient.

Example

```
int oid;
int grad;

grad = GetObjectGradient( oid);
```

21.45 GetObjectName()

string GetObjectName(int oid)

Description

Get the object name.

Argument

int	oid	Object identity.

Returns the name.

Example

```
int oid;
string oname;

oname = GetObjectName( oid);
```

21.46 GetObjectShadow()

int GetObjectShadow(int oid)

Description

Get the shadow property of an object.

Argument

int	oid	Object identity.

Returns the shadow, 0 or 1.

Example

```
int oid;
int shadow;

shadow = GetObjectShadow( oid);
```

21.47 GetObjectText()

string GetObjectText(int oid)

Description

Fetches the text of a text object.

Argument

int	oid	Object identity.

Returns the text.

Example

```
int oid;
string text;

text = GetObjectText( oid);
```

21.48 GetObjectTextColor()

int GetObjectTextColor(int oid)

Description

Get the text color of an object.

Argument

int	oid	Object identity.

Returns the text color.

Example

```
int oid;
int tcolor;

tcolor = GetObjectTextColor( oid);
if ( tcolor == eDrawType_Color49)
  SetObjectTextColor( oid, eDrawType_CustomColor5);
endif
```

21.49 GetObjectTransparency()

float GetObjectTransparency(int oid)

Description

Get the level of transparency of an object.

Argument

int	oid	Object identity.

Returns the tranparency. A float value in the range 0.0 - 1.0.

Example

```
float transparency;

transparency = GetObjectTransparency( oid);
```

21.50 GetObjectType()

int GetObjectType(int oid)

Description

Get the type of an object.

Argument

int	oid	Object identity.

Returns the object type.
See Predefined variables for object type values.

Example

```
int oid;
int type;

type = GetObjectType( oid);
if ( type == eObjectType_SubGraph)
  SetObjectFillColor( oid, eDrawType_CustomColor22);
endif
```

21.51 void GetRgbColor()

void GetRgbColor(int color, float red, float green, float blue)

Description

Fetches rgb values for a color. The color is stated with color index of with color symbol, see below. The rgb valus are float values in the range 0 - 1.

Argument

int	color	Color.
float	red	Returns the red value.
float	green	Returns the green value.
float	red	Returns the blue value.

The colors of the standard palette are defined with eDrawType_Color1 - eDrawType_Color300, and the colors of the custom palette with eDrawType_CustomColor1 - eDrawType_CustomColor90.

The corresponding color indexes can also be used, for the standard palette 0 - 299, and for the custom palette 310 to 669. For the custom palette the base colors are positioned on every fourth index, ie 310, 314, 318, ... 666.

Example

```
int i;
float r;
float g;
float b;

printf( "Idx  Red   Green Blue\n");
for ( i = eDrawType_CustomColor1; i <= eDrawType_CustomColor90; i += 4)
  GetRgbColor( i, r, g, b);

  printf( "%3d  %5.3f %5.3f %5.3f\n", i, r, g, b);
endfor
```

21.52 GetTextExtent()

int GetTextExtent(string text, int textsize, int font, int bold, float width [,
float height, float descent])

Description

Calculates the extent of a text.
Returns the width of the text in the width argument, and if the arguments height and descent are supplied, also height and descent in these arguments.

Argument

string	text	Text which size is to be measured.
int	textsize	Text size index (0-5).
int	font	Text font.
int	bold	Bold (1) or normal (0) text.
float	width	The text width is returned in this variable.
float	height	The height is returned in this variable.
float	descent	The descent is returned in this variable.

Example

```
float width;

GetTextExtent("This is a text", 3, eFont_LucidaSans, 1, width);
```

21.53 GetUI_Env()

int GetUI_Env()

Description

Get the current environment. The environment can be eUI_Env_Web, eUI_Env_Xtt or eUI_Env_Development.

returns the environment.

Example

```
int env

env = GetUI_Env();
if (env == eUI_Env_Web)
...
```

21.54 GetWindowSize()

string GetWindowSize(float width, float height)

Description

Fetches the size in pixel of the current window.

Attributes

float	width	The window width is returned in this variable.
float	height	The window height is returned in this variable.

Example

```
float width;
float height;

name = GetWidowSize(width, height);
```

21.55 GetWindowDimension()

string GetWindowDimension(float width, float height)

Description

Fetches the size of the current window.

Attributes

float	width	The window width is returned in this variable.
float	height	The window height is returned in this variable.

Example

```
float width;
float height;

name = GetWidowDimension(width, height);
```

21.56 GroupGetFirstObject()

int GroupGetFirstObject(int group)

Description

Get the first object in a group. Other objects can be fetched with GroupGetNextObject().

Argument

int	group	Group identity.

Returns in integer with the object identity.

Example

```
int goid;
int oid;

# Get all objects
goid = GetFirstObject();
while ( goid != 0)
  type = GetObjectType( goid);
  if ( type == eObjectType_Group)
    # This is a group, get all objects in the group
    oid = GroupGetFirstObject( goid);
    while ( oid != 0)
      ...
      oid = GroupGetNextObject( goid, oid)
  endwhile
endwhile
```

21.57 GroupGetNextObject()

int GroupGetNextObject(int group, int oid)

Description

Get the next object in a group, The first object is fetched with GroupGetFirstObject() and the other objects with GroupGetNextObject().

Argument

int	oid	Object identity.

Returns an integer with the object identity.

21.58 GroupSelected()

int GroupSelected()

Description

Create a group of the selected objects.

Returns an integer with the group identity.

Example

```
int id1;
int id2;
int gid;

SelectAdd(id1);
SelectAdd(id2);
gid = GroupSelected();
```

21.59 LayerGetFirstObject()

int LayerGetFirstObject(int lid)

Description

Get the first object in a layer.

Argument

int	lid	Layer identity.

Returns the id of the first object.

Example

```
int lid;
int oid;

oid = LayerGetFirstObject(lid);
while (oid)
  ...
  oid = LayerGetNextObject(lid, oid);
endwhile
```

21.60 LayerGetNextdObject()

int LayerGetNextObject(int lid, int oid)

Description

Get the next object in a layer.

Argument

int	lid	Layer identity.
int	oid	Object identity.

Returns the id of the next object.

21.61 LayerSetActive()

void LayerSetActive(int id, int active)

Description

Set a layer active or inactive.

Argument

int	oid	Layer identity.
int	active	1 will set layer active, 0 inactive.

Example

```
int id;

LayerSetActive(id, 1);
```

21.62 LayerResetActive()

void LayerResetActive()

Description

Set all layers inactive.

21.63 Layout()

void Layout(float width, float height, int prio[], int top[],
fint down[], int left[], int right[], float pref_width, fload pref_height,
float fix_width, float fix_height, float calc_x, float, calc_y,
float calc_width, float calc_height)

Description

Calculates the layout for the modules of the current window.

Argument

float	width	Window width
float	height	Window height
int	prio	Prio array
int	top_neighbour	Top neighbour array
int	down_neighbour	Down neighbour array
int	left	Left neighbour array
int	right	Right neighbour array
float	pref_width	Preferred width
float	pref_height	Preferred height
int	fix_width	Fix width
int	fix_height	Fix height
float	calc_x	Calculated x coordinate
float	calc_y	Calculated y coordinate
float	calc_width	Calculated width
float	calc_heigh	Calculated height

21.64 MeasureObject()

void MeasureObject(int oid, float ll_x, float ll_y, float ur_x, float ur_y)

Description

Get the extention of an object.
Returns the x and y coordinates for lower left and upper right limits.

Argument

int	oid	Object identity.
float	ll_x	Returned x coordinate for lower left.
float	ll_y	Returned y coordinate for lower left.
float	ur_x	Returned x coordinate for upper right.
float	ur_y	Returned y coordinate for upper right.

Example

```
int oid;
float ll_x;
float ll_y;
float ur_x;
float ur_y;

MeasureObject( oid, ll_x, ll_y, ur_x, ur_y);
```

21.65 MergeAllLayers()

void MergeAllLayers()

Description

Merge all layers.

21.66 MergeVisibleLayers()

void MergeVisibleLayers()

Description

Merge visible layers.

21.67 MoveAbsObject()

void MoveAbsObject(int oid, float x, float y)

Description

Move an object to coordinates x and y.

Argument

int	oid	Object identity.
float	x	x coordinate.
float	y	y coordinate.

Example

```
MoveAbsObject( oid, 0.5, o.5);
```

21.68 MoveObject()

void MoveObject(int oid, float dx, float dy)

Description

Move an object a distance.

Argument

int	oid	Object identity.
float	dx	Distance to move in x direction.
float	dy	Distance to move in y direction.

Example

```
MoveObject( oid, 0.5, 0.5);
```

21.69 MoveSelectToLayer()

void MoveSelectToLayer()

Description

Move the selected objects to the active layer.

21.70 PopSelected()

void PopSelected()

Description

Pop the selected objects.

21.71 PushSelected()

void PushSelected()

Description

Push the selected objects.

21.72 OpenGraph()

void OpenGraph(string name)

Description

Open a graph.

Argument

string	name	Name of graph.

Example

```
OpenGraph("overview");
```

21.73 PolyLineAdd()

int PolyLineAdd(int id, float x, float y)

Description

Add a segment to a polyline.

Argument

float	x	X coordinate for end point.
float	y	Y coordinate for end point.

Example

```
int id;

id = CreatePolyLine(2.0, 2.0, 3.0, 3.0);
PolyLineAdd(id, 4.0, 2.0);
PolyLineAdd(id, 5.0, 3.0);
```

21.74 Reload()

void Reload()

Description

Reads the current graph from file.

Example

```
# Update subgraph pwr_roundind
SetExtern("pwr_roundind");
save
Reload();
SetIntern("pwr_roundind");
save
```

21.75 RotateSelected()

void RotateSelected(float angle, int rotationpoint)

Description

Rotate selected object around the rotation point.
The rotation point can be

0 lower left corner
1 lower right corner
2 upper right corner
3 uppder left corner
4 center (default)

Arguments

float	angle	Angle in degrees for the rotation.
int	rotationpoint	Rotation point. Default center.

Example

```
int id;

SelectAdd(id);
RotateSelected(90.0);
```

21.76 SaveGraph()

void SaveGraph([string name])

Description

Save the current graph.

Argument

string	name	Name of graph. Optional.

Example

```
SaveGraph();
```

21.77 ScaleObject()

void ScaleObject(int oid, float scalex, float scaley)

Description

Scale an object.

Argument

int	oid	Object identity.
float	scalex	Scale factor in x direction.
float	scaley	Scale factor in y direction.

Example

```
ScaleObject( oid, 1.5, 1.5);
```

21.78 SelectAdd()

int SelectAdd(int objectid)

Description

Insert an object in the list of selected objects.

Argument

int	objectid	Object identity.

Example

```
int id;

id = GetCurrentObject();
SelectAdd( id);
```

21.79 SelectClear()

int SelectClear()

Description

Clear the list of selected objects.

21.80 SetBackgroundColor()

void SetBackgroundColor(int color)

Description

Set the background color for the current graph.

Argument

int	color	Background color.

Example

```
SetBackgroundColor(eDrawType_CustomColor1);
```

21.81 SetColorTheme()

void SetColorTheme([int colortheme])

Description

Set the colortheme for the current graph.
In runtime also ColorTheme in GraphAttributes has to be set to $default. In this case SetColorTheme() should be called without argument.

Argument

int	colortheme	Color theme index. Optional.

Example

```
SetGraphAttribute("ColorTheme", "$default");
SetColorTheme();
```

21.82 SetCurrentObject()

void SetCurrentObject(int oid)

Description

Set CurrentObject.

Argument

int	oid	Object identity.

Example

```
SetCurrentObject( oid);
```

21.83 SetGraphOptions()

int SetGraphOptions(int options)

Description

Set graph options.

Argument

int	options	Graph options.

21.84 SetDraw()

int SetDraw(int on)

Description

Set drawing on or off.

Argument

int	on	If 0, drawing is turned off. If 1, drawing is turned on.

Example

```
SetDraw(0);

for ( i = 0; i < 10000; i++)
  x2 = x + 0.55;
  y2 = y + 0.55;
  create object/sub=pwr_indsquare/x1='x'/y1='y'/x2='x2'/y2='y2'
  set current attr DigLowColor.Attribute H1-DArray.Val['i']##Boolean
  x += 0.65;
  if ( i - (i / 100) * 100 == 99)
    y += 0.65;
    x = 0;
  endif
endfor

SetDraw(1);
```

21.85 SetExtern()

int SetExtern(string name)

Description

Set a subgraph extern.

Argument

string	name	Name of the subgraph.

Example

```
SetExtern("MySubgraph");
```

21.86 SetExternAll()

void SetExternAll()

Description

Set all subgraphs extern.

Example

```
SetExternAll();
```

21.87 SetIntern()

int SetIntern(string name)

Description

Set a subgraph intern.

Argument

string	name	Name of the subgraph.

Example

```
SetIntern("MySubgraph");
```

21.88 SetInternAll()

void SetInternAll()

Description

Set all subgraphs intern.

Example

```
SetInternAll();
```

21.89 SetGraphAttribute()

int SetGraphAttribute(string attribute, (arbitrary type) value)

Description

Set the value of an attribute in the current graph. The attribute name is the same as in the graph attributes editor.

Quotation marks inside a string can be set with \", and new line in attributes with several lines with \\n.

Argument

string	attribut	Attribute name.
(arbitrary type)	value	Attribute value.

Returns the status of the operation.

Example

```
int sts;

sts = SetObjectAttribute( "Scantime", 0.2);
```

21.90 SetGraphName()

string SetGraphName()

Description

Set the name for the current graph.

Argument

string	name	Name of the graph.

Example

```
string name = "overview";

SetGraphName(name);
```

21.91 SetObjectAttribute()

int SetObjectAttribute(int oid, string attribute, (arbitrary type) value)

Description

Set the value of an attribute in an object. The attribute name is the same as in the object editor for the object.

Quotation marks inside a string can be set with \", and new line in attributes with several lines with \\n.

Argument

int	oid	Object identity.
string	attribut	Attribute name.
(arbitrary type)	value	Attribute value.

Returns the status of the operation.

Example 1

```
 int sts;
 int oid;

 sts = SetObjectAttribute( oid, "DigColor.Color", eDrawType_Color244);
```

Example 2

Script attribute with several lines and strings with quotes.

```
  SetObjectAttribute(id, "Script.Script", "\
int a;\\n\
a = GetAttribute(\"H1-Dv1.ActualValue\");\\n\
if (a)\\n\
  SetSubwindow(\"$current\",\"W1\",\"@plant1\",\"\", 0);\\n\
else\\n\
  SetSubwindow(\"$current\",\"W1\",\"@plant2\",\"\", 0);\\n\
endif");
```

21.92 SetObjectBackgroundColor()

void SetObjectBackgroundColor(int oid, int color)

Description

Set the background color of an object.

Argument

int	oid	Object identity.
int	color	Background color.

Example

```
SetObjectBackgroundColor( oid, eDrawType_CustomColor1);
```

21.93 SetObjectBorder()

void SetObjectBorder(int oid, int border)

Description

Set the border property of an object.

Argument

int	oid	Object identity.
int	border	border property. 1 will display the border, 0 will not.

Example

```
SetObjectBorder( oid, 1);
```

21.94 SetObjectBorderColor()

void SetObjectBorderColor(int oid, int color)

Description

Set the border color of an object.

Argument

int	oid	Object identity.
int	color	Border color.

Example

```
SetObjectBorderColor( oid, eDrawType_CustomColor1);
```

21.95 SetObjectClass()

void SetObjectClass(int oid, string subgraph)

Description

Replace the subgraph of an object.

Argument

int	oid	Object identity.
string	subgraph	The name of the new subgraph.

Example

```
int oid;
string class;

class = GetObjectClass( oid)
if ( class == "pwr_indsquare")
  SetObjectClass( oid, "pwr_indround");
endif
```

21.96 SetObjectFill()

void SetObjectFill(int oid, int fill)

Description

Set the fill property of an object.

Argument

int	oid	Object identity.
int	fill	Fill property. 1 will draw with fill, 0 will not.

Example

```
SetObjectFill( oid, 1);
```

21.97 SetObjectFillColor()

void SetObjectFillColor(int oid, int color)

Description

Set the fill color of an object.

Argument

int	oid	Object identity.
int	color	Fill color.

Example

```
SetObjectFillColor( oid, eDrawType_CustomColor1);
```

21.98 SetObjectGradient()

void SetObjectGradient(int oid, int gradient)

Description

Set the gradient property of an object.

Argument

int	oid	Object identity.
int	gradient	Gradient property. 0 means no gradient.

Example

```
SetObjectGradient( oid, 0);
```

21.99 SetObjectShadow()

void SetObjectShadow(int oid, int shadow)

Description

Set the shadow property of an object.

Argument

int	oid	Object identity.
int	shadow	Shadow property, 1 will draw the object with shadow, 0 will not.

Example

```
SetObjectShadow( oid, 1);
```

21.100 SetObjectTextColor()

void SetObjectTextColor(int oid, int color)

Description

Set the text color of an object.

Argument

int	oid	Object identity.
int	color	Text color.

Example

```
SetObjectTextColor( oid, eDrawType_CustomColor5);
```

21.101 SetObjectTransparency()

void SetObjectTransparency(int oid, float transparency)

Description

Set the level of transparency of an object.

Argument

int	oid	Object identity.
float	transparency	Level of transparency. A value in the range 0.0 - 1.0.

Example

```
SetObjectTransparency(oid, 0.5);
```

21.102 void SetRgbColor()

void SetRgbColor(int color, float red, float green, float blue)

Description

Set rgb values for a color in the custom palette. The color is stated with color index of with color symbol, see below. The rgb valus are float values in the range 0 - 1.

Argument

int	color	Color.
float	red	Red value.
float	green	Green value.
float	red	Blue value.

The colors of the standard palette are defined with eDrawType_Color1 - eDrawType_Color300, and the colors of the custom palette with eDrawType_CustomColor1 - eDrawType_CustomColor90.

The corresponding color indexes can also be used, for the standard palette 0 - 299, and for the custom palette 310 to 669. For the custom palette the base colors are positioned on every fourth index, ie 310, 314, 318, ... 666.

Example

```
float red = 0.731;
float green = 0.224;
float blue = 0.328;

SetRgbColor( eDrawType_CustomColor1, red, green, blue);
```

21.103 void SetSelectTextBold()

void SetSelectTextBold(int font)

Description

Set bold to selected objects.

Argument

int	bold	1 for bold and 0 for normal text.

Example

```
int id;

SelectAdd(id);
SetSelectTextBold(1);
```

21.104 void SetSelectTextFont()

void SetSelectTextFont(int font)

Description

Set font to selected objects.

Argument

int	font	Text font.

Example

```
int id;

SelectAdd(id);
SetSelectTextFont(eFont_LucidaSans);
```

21.105 void TranslateObjectName()

void TranslateObjectName(string ing, string out)

Description

Translate an object or attribute name with Ge connection syntax
to database name.

Argument

string	in	Name with Ge connection syntax.
string	out	Database name.

Example

```
string in;
string out;

TranslateObjectName(in, out);
```

21.106 void SetSelectTextSize()

void SetSelectTextSize(int size)

Description

Set text size to selected objects.

Argument

int	size	Size index value in the range 0-5.

Example

```
int id;

SelectAdd(id);
SetSelectTextSize(2);
```

21.107 Example

Example 1

```
!
! Draw some simple objects
!
main()
  string name;

  verify(1);


  ! Draw a rectangle
  set fill
  set shadow
  set linewidth 1
  set bordercolor Black
  set fillcolor BlueHigh8
  create rect /x1=10 /y1=1 /width=20 /height=12
  set current attr shadow_width 2
  set current attr gradient Globe

  ! Draw a circle
  set fill
  set shadow
  set fillcolor MagentaHigh4
  set linewidth 2
  create arc /x1=1/y1=1/x2=6/y2=6
  set current attr shadow_width 12
  set current attr gradient DiagonalLowerLeft

  ! Draw a polyline
  set fill
  set shadow
  set fillcolor YellowGreenHigh5
  set linewidth 1
  create polyline /x1=5 /y1=5 /x2=10 /y2=10
  add polyline /x1=10 /y1=20
  add polyline /x1=5 /y1=20
  add polyline /x1=5 /y1=5
  set current attr gradient DiagonalLowerLeft

  ! Print a text
  set bold
  set textsize 24
  set textfont LucidaSans
  set textcolor RedHigh8
  create text/text="Example" /x1=3 /y1=5
```

```
  ! Create a subgraph
  set fillcolor YellowGreenMedium3
  create object/sub=pwr_valve/x1=1/y1=1/x2=3/y2=2
  set current attr DigLowColor.Attribute "Rt-Dv1.ActualValue##Boolean"
  set current attr DigLowColor.Color GrayHigh9

  ! Create a pushbutton
  create object /sub=pwr_buttonsetcenter /x1=5 /y1=1
  set current annotation "Start"
  set current attr SetDig.Attribute "rt-dv1.actualvalue##Boolean"
  set current attr Access System|Operator1

  ! Set graph attributes
  set graphattr x0 -3
  set graphattr y0 -3
  set graphattr x1 40
  set graphattr y1 25

  save
endmain
```

Example 2

```
!
! This example finds all dv objects in an hierarchy,
! prints the name and displayes the value in an indicator.
! A frame is drawn around the objects.
!
main()

  string name;
  string segname;
  string attr;
  float x;
  float y;
  float x_ind;
  float y_ind;
  float x1;
  float x2;
  float y1;
  float y2;
  float width;
  float height;
  float t_width;
  string class;

  verify(1);

  x_ind = 2;
  x = 4;
  y_ind = 1.5;
```

```
  y = 2;
  name = GetChild( p1);

  if ( p1 == "")
    printf("usage : test3 'parent'\n");
    exit();
  endif

  set bold
  set textsize 12
  while ( name != "")
    class = GetObjectClass( name);
    if ( class == "Dv")
      create object/sub=pwr_indsquare/x1='x_ind'/y1='y_ind'
      attr = name + ".ActualValue##Boolean";
      set current attr DigLowColor.Attribute "'attr'"
      segname = CutObjectName( name, 1);
      create text/text="'segname'"/x1='x'/y1='y'
      GetTextExtent( segname, 12, 1, t_width);
      if ( t_width > width)
        width = t_width;
      endif
      y += 1;
      y_ind += 1;
    endif
    name = GetNextSibling( name);
  endwhile

  x1 = x_ind - 1;
  x2 = x + width + 1;
  y1 = 0;
  y1 = y;
  width = x2 - x1;
  height = y2 - y1;

  set fillcolor GrayLow3
  set shadow
  set noborder
  set nofill
  create rectangle/x1='x1'/y1='y1'/width='width'/height='height'
  set current attr shadow_width 2
  set current attr relief Down

  x1--;
  x2++;
  y1--;
  y2++;
  set background GrayLow3
  set graph x0 'x1'
  set graph y0 'y1'
  set graph x1 'x2'
  set graph y1 'x2'
endmain
```